OpenOffice.org 2.0 e i Database

Introduzione all'uso dei Database con OpenOffice.org 2.0

Versione 0.99 – Dicembre 2005

Soft.Com

Indice Generale

1. Introduzione

OpenOffice.org è un potente software di produttività personale composto da moduli dedicati ad aspetti diversi della elaborazione delle informazioni, ma ben integrati tra loro. La versione attualmente disponibile (al momento della stesura di questo documento) è la 2.0.0, ma lo sviluppo prosegue velocemente. OpenOffice (nel seguito **OOo**) comprende i tre moduli principali necessari all'uso *office* , cioè una SW di Videoscrittura (***Writer***), un Foglio di Calcolo (***Calc***), un SW di Presentazione (***Impress***). La suite inoltre dispone di un modulo di Disegno (***Draw***) che può servire ad integrare immagini anche mediamente complesse nei propri documenti. Nella versione 2.0 è stato aggiunto un nuovo modulo chiamato ***Base*** e dedicato alla gestione degli Archivi. OOo *Base*, oltre a possedere un proprio Database interno, ha anche la possibilità di "interagire" con archivi esterni in modo da poter utilizzare *dati* disponibili in molti formati diversi.

Rispetto ad altre soluzioni, anche molto diffuse, OOo ha due importanti vantaggi, a parità di funzionalità e potenza disponibile:

- **è disponibile a basso costo o gratuitamente** , perché il codice sorgente è *libero* (rilasciato sotto una Licenza che ne permette la variazione, l'integrazione e la distribuzione);
- **è multi piattaforma**, cioè può essere usato in modo esattamente identico su molti sistemi operativi, garantendo sempre e comunque lo scambio senza modifiche dei documenti.

Scopo di questa guida è illustrare in modo semplice ed accessibile, anche a chi non ha una grande dimestichezza con gli aspetti più complessi dell'informatica, quali possibilità offre OOo per leggere, modificare, integrare, stampare dati provenienti da "basi di dati" (o ***Database***, abbreviato **Db**) esterni al programma stesso. A tal fine useremo prodotti anch'essi gratuiti, ***liberi*** e multi piattaforma come ***MySql*** e ***PostgreSQL***, che sono appunto dei "database server" (e vedremo più avanti che cosa significa). Spazio sarà anche dedicato a due soluzioni proprietarie, ***Ms SQL Server 2005 Express*** e ***Oracle10g Express***, che sono disponibili in uso gratuito e liberamente scaricabili dalla rete. Questo anche per dimostrare che OpenOffice.org può essere utilizzata senza limitazioni anche con prodotti non liberi.

TIP

In realtà la Licenza di *MySql*, pur essendo classificata come Open Source, pone dei limiti all'utilizzo commerciale del pacchetto. Nel caso doveste usarlo in ambienti di produzione (cioè in azienda), sarebbe meglio dare uno sguardo a www.mysql.com e leggere con attenzione i termini di Licenza. Questo non accade con gli altri prodotti esaminati.

Nel testo saranno evidenziate due sezioni particolari:

- i **TIP** saranno suggerimenti o scorciatoie utili a velocizzare e razionalizzare il lavoro;
- la parte **tecnica** servirà ad approfondire alcuni argomenti più *difficili*, e può essere *saltata* senza problemi da chi non è interessato.

OOo è liberamente scaricabile dal Sito www.openoffice.org, dove è anche possibile consultare una ottima sezione in Italiano.

Tutto è perfettibile, quindi mi scuso in anticipo per eventuali inesattezze presenti nel testo; suggerimenti, correzioni, integrazioni saranno benvenuti, e potranno servire a migliorare questa documentazione. Buona lettura.

1.1 Meglio però non illudersi....

Bisogna precisare subito che *OpenOffice* NON E' uno strumento *RAD* (cioè di sviluppo rapido di applicazioni) e nemmeno vuole esserlo. Ogni paragone quindi con prodotti specifici è fuorviante, perciò non aspettatevi di trovare caratteristiche avanzate di gestione delle interfacce o strumenti di reporting di livello professionale. In ambito Office però sono molte le cose che si possono realizzare, e, se anche OOo non è adatto a scrivere un gestionale, è sufficientemente potente per applicazioni non molto complesse.

Lasciatemi comunque dire, a scanso di equivoci, che, nonostante l'inclusione di uno specifico modulo "*Base*" nella versione 2.0, ogni paragone con *Microsoft Access* è fuorviante. Infatti è mia opinione che la scelta di includere *Ms Access* come componente di Office è soprattutto dovuta a problemi di marketing, più che a scelte tecniche o logiche. *Access*, infatti, è un prodotto molto diverso dagli altri componenti della suite. Si tratta, come ben sanno quelli che lo usano in modo intensivo, sostanzialmente di uno strumento *RAD* (*Rapid Application Developement*) orientato ai Database, tra l'altro poco integrabile ad esempio con *Word* o *Excel*. Inoltre, in barba alle molte *auto composizioni*, è piuttosto difficile da utilizzare e comprendere, e risultati accettabili si ottengono solo se usato da professionisti con ampio ricorso al linguaggio di programmazione.

Ovviamente, *un utente medio può fare più o meno le stesse cose con OpenOffice e con Access*. Infine, il termine di paragone di OpenOffice non è Ms Office, ma... se stesso. Sono due prodotti diversi, che pur volendo fare più o meno la stessa cosa la fanno con filosofie e modalità diverse. Quindi discutere di quale sia migliore è aria fritta. Ognuno sceglierà quello che ritiene più adatto ai propri scopi, e, poi, noi abbiamo già scelto.....

1.2 Configurazione utilizzata

Tutti i documenti di esempio sono stati creati e modificati sulle seguenti configurazioni:

Ambiente **Windows** (client):

Pc Amd Athlon 2200+, 512 Mb di Ram, Hd 80 Gb, Ati Radeon 9250
Sistema Operativo Windows XP Service Pack 2
OpenOffice Versione 2.0.0 Italiano

Ambiente **Windows** (server) :

Pc Amd Athlon 2600+, 512 Mb di Ram, Hd 80 Gb, Ati Radeon 9250
Sistema Operativo Windows XP Service Pack 2
MySQL versione 5.0.15
PostgreSQL versione 8.0.3

Ambiente **Linux** (Server)

Pc Amd Athlon 1700+, 512 Mb di Ram, Hd 40 Gb, Matrox Millennium
Sistema Operativo Linux SuSe 9.3
MySql Server 4.1.10a

Ambiente **Linux** (Client)

Pc Portatile Toshiba S1800-400
Sistema Operativo Linux SuSe OSS 10.0

La maggior parte dei concetti e delle soluzioni presentati nel seguito sono indipendenti dal motore di Database utilizzato. Questo significa che, in generale, la sola presenza di un driver ODBC oppure JDBC affidabile permette l'utilizzo di un qualsiasi prodotto alternativo (come Ms Access, oppure Sybase etc.).

1.3 Installazione delle Applicazioni

L'installazione di OpenOffice, sia in ambiente Windows che Linux, non comporta grosse difficoltà. In **Appendice** troverete notizie dettagliate sulle procedure da seguire. Per i server di Database consultate i capitoli ad essi dedicati.

2. Il mondo dei Database

2.1 Primi passi....

Questa sezione serve ad introdurre alcuni concetti indispensabili per le persone che non sono molto esperte di Database. Se la cosa vi annoia, saltatela pure.

La traduzione letterale di ***Database*** è ***Base di dati***, e serve almeno per farsi un'idea dell'argomento che andiamo ad affrontare. Più semplicemente, potremo indicare un Database con la generica parola **archivio**. Ora, un archivio è una idea semplice, nulla di astratto, usiamo ogni tipo di archivio molto più spesso di quanto si creda. Basta un elenco telefonico per capire cos'è un archivio, basta guardare la rubrica del cellulare....

Potremmo definire un a*rchivio* (*database*) come un insieme di informazioni, organizzate in una struttura logica, spesso ordinate secondo una propria necessità, con uno o più caratteristiche in comune. Guardiamo appunto una rubrica telefonica: la caratteristica in comune è "archivio dei numeri della rubrica di Giuseppe", la struttura comprende due informazioni (il nome ed il numero), l'ordine è di solito quello alfabetico.

Nome	**Numero**
Carla	340 1234
Elisa	06 54678
Giuditta	02 987456

Questa, nel gergo dei Database, è una **Tabella** (***table***), che comprende due **Campi** *(**field**)*, *Nome* e *Numero*. La Tabella comprende tre **Righe** chiamate anche **Schede** *(**row**, **record**)*. Un **Database** è di solito un ***insieme di Tabelle*** che possono essere messe in ***relazione*** tra loro tramite connessioni logiche (presenza in più tabelle della stessa informazione). Da questa definizione deriva il nome di **Database Relazionale** *(RDBMS, Relational Database Management System)* assegnato alla tipologia di prodotti che stiamo esaminando. Attenzione, spesso si indica con lo stesso nome (*Database*) sia il *motore* cioè il software che mi permette di gestire le informazioni, sia le *informazioni* stesse. Questo, in generale, non è corretto.

Tecnica

Esiste una *definizione* molto precisa dei database relazionali. Quindi un Database, per poter essere correttamente definito *relazionale*, deve soddisfare una serie di regole che qui sarebbe lungo ed improduttivo descrivere. Secondo alcuni, ad esempio, *MySql*, siccome non possiede TUTTE queste caratteristiche, NON E' a rigore un Database relazionale. Questo però a noi interessa davvero poco....

2.2 Progettiamo un Database

Abbiamo quindi definito la **Tabella** come ***l'unità logica di un Database***. Ovviamente possiamo avere Database composti da una sola Tabella, ma direi che sono casi particolari. Ora siamo perciò di fronte al nostro amato PC e vogliamo cominciare a creare il nostro primo Database; la prima cosa da fare è quindi... *un bel passo indietro*.

Infatti l'approccio più sbagliato che possa esistere è partire con la struttura di un Db senza averci prima ben riflettuto... magari davanti ad un bel foglio bianco e con una cara e vecchia penna in mano. Voglio dire che in generale è fondamentale "progettare" il Database *PRIMA* di mettere la manine sulla tastiera, ed quindi il buonsenso consiglia di porsi in anticipo le seguenti domande:

1. di quante e quali unità logiche (*tabelle*) deve essere composto il mio Db ?
2. per ogni tabella, quali informazioni (*campi*) devo comprendere ?
3. per ogni campo di ogni tabella, che tipo di informazioni devo archiviare ?
4. quali sono i campi su cui sarà necessario eseguire un *ordinamento* ?
5. che *relazioni* ci sono tra le varie tabelle ?
6. che *operazioni* desidero eseguire su ogni tabella ?

Avrete quindi capito che, siccome "chi ben comincia etc. etc.", rispondere a queste domande all'inizio del lavoro vi eviterà problemi nel seguito. Questo non significa che in corso d'opera non si potranno fare variazioni, ma cambiare la struttura di Db complessi quando già è iniziato il caricamento dei dati è davvero complicato. Inoltre, non prevedere qualche piccolo dettaglio può portare a risultati pericolosi, vi ricordate l'affare dell'anno 2000 ?

2.3 La Tabella

Una **Tabella**, abbiamo visto, è composta da **Campi**. Possiamo pensare al *Campo* come l'intestazione di colonna di una lista, ma in realtà è molto di più di una descrizione. In genere, infatti un *Campo* incorpora numerose *proprietà*, cioè *caratteristiche* che, una volta impostate, determineranno in modo preciso il tipo di informazione che quel *Campo* può contenere.

Esaminiamo un po' più in dettaglio queste caratteristiche:

- Il **nome** del Campo è in genere la *descrizione* dell'informazione (ad es. *numero di telefono*). Non ci sono molti vincoli sul nome, vi consiglio però di non sceglierlo troppo lungo, ma allo stesso tempo abbastanza esplicativo. Nel nostro caso andrebbe ad es. bene *NumTel*. Se il Db comprende più Tabelle in cui appare lo stesso campo (se ho, ad esempio, la Tabella Clienti e quella Fornitori), e non sono in relazione tra loro, può essere utile anteporre al nome del campo un indicativo della Tabella stessa, ad es. "CNumTel" e "FNumTel". In genere si possono usare gli *underscore* (Num_Tel) ma io lo trovo poco pratico.

- Il **tipo** indica la tipologia della informazione da archiviare nel campo. Le tre categorie principali sono ***stringa***, ***numero*** e ***data/ora***. All'interno di ogni categoria possiamo scegliere molte ulteriori tipologie, ed inoltre, come vedremo, esistono *tipi* particolari (come *binario* o *timestamp*) non riconducibili facilmente agli altri.
- La **lunghezza** misura la dimensione massima che vogliamo assegnare all'informazione contenuta nel campo. Per le *stringhe* si indicano i caratteri (ad es max 30 caratteri per il campo *nome*), per i *valori numerici* il discorso è un po' più complesso (lo vedremo in seguito), per *data/ora* esistono formati standard con diverse occupazioni di memoria a seconda dell'intervallo dei valori che si vuole archiviare.
- **Ammetti null** è un segnalatore che indica la possibilità di archiviare nel campo anche valori *nulli* (a prima vista può sembrare inutile, invece la scelta è assai importante).
- **Valore predefinito** è il valore che dovrà assumere in automatico il campo all'immissione di una nuova riga se non viene modificato dall'utente. Ad esempio, se la maggior parte dei nostri clienti è della provincia di Milano, potremmo volere che il campo Provincia sia riempito dalla stringa *MI*. Se nel valore predefinito non si specifica niente, di solito il campo assume il valore *null*. Per i campi numerici è particolarmente importante assegnare come valore predefinito lo *zero* (come vedremo tra poco).
- **Chiave** oppure i**ndice** stabilisce se il campo debba essere indicizzato e se l'indice è univoco.

Queste *proprietà del campo* si ritrovano quasi identiche in praticamente tutte le varietà di Db Server. Ogni prodotto, poi, implementa *aggiunte* magari non standard, ma egualmente importanti. Ora descriveremo più in dettaglio le principali tipologie di Dati, in modo da capire meglio come strutturare le nostre Tabelle.

2.4 Indici

Alla domanda: *data una Tabella, quali campi è necessario indicizzare?* la risposta è *dipende...* . Cioè in pratica non si risponde, perché non c'è una regola generale universalmente condivisa. Ognuno, in base alla propria esperienza, decide in modo autonomo. Inoltre gestire poche centinaia oppure alcune migliaia di record sono cose ben diverse, come pure usare da soli il proprio Db oppure far parte di una rete con decine di macchine. Questo non significa che non vi posso dare qualche buon consiglio....

Innanzi tutto cerchiamo di capire **cos'è un indice**. Una volta definita la struttura della nostra tabella, ogni volta che aggiungiamo una riga, il nostro motore di Db *accoda* sul disco le informazioni all'archivio aperto. Tornando al nostro semplice esempio della Rubrica del cellulare, quando aggiungiamo i nomi, di certo non lo facciamo in ordine alfabetico; ma comunque, nella consultazione, l'ordine alfabetico ci è molto comodo. Quindi il software del nostro cellulare crea un *indice* sul campo *nome*, in modo da poterci fornire le informazioni nell'ordine più logico.

Quindi un *indice* è, in generale, un *ordinamento* creato su uno o più *campi* in modo che il reperimento delle informazioni contenute nell'indice stesso sia molto rapido. Non è il caso di spiegare in dettaglio come questo avvenga a livello di programmazione, accontentiamoci dei risultati. Allora....

Consiglio 1: i motori di Db funzionano meglio se in ogni tabella esiste un *indice univoco*, cioè un valore *specifico* e *diverso* per ogni riga. Questo campo di solito si chiama ***Chiave Primaria*** (***Primary Key***). Possiamo perciò definire la "*chiave primaria*" come *un campo che assume un valore diverso per ogni riga della tabella, e quindi identifica univocamente la riga stessa*.

Consiglio 2: i campi *Interi ad incremento automatico* sono (come vedremo) i candidati ideali per le *Chiavi Primarie*. Infatti i motori di Db hanno prestazioni ottimali sugli indici numerici interi.

Consiglio 3: create un indice sui campi che desiderate siano ordinati sui prospetti di stampa (report). Se ho un archivio clienti, e mi serve una stampa in ordine alfabetico, dovrò creare un indice sulla *Denominazione*.

Consiglio 4: create un indice sui campi che userete per cercare delle informazioni. Se ho un elenco di libri, una buona idea è creare un indice sul campo *autore* (cercherò sicuramente i libri per autore)

Consiglio 5: se avete due tabelle correlate (messe in relazione) tra loro, create indici sui campi comuni (questo, se non vi è chiaro, lo capirete più avanti)

Consiglio 6: troppi indici rallentano il sistema. Molti utenti ancora inesperti sono portati ad indicizzare tutto l'indicizzabile. Sbagliato. Ogni operazione di modifica dei dati, comporta, se il campo è indicizzato, anche l'aggiornamento degli indici stessi. Inoltre gli indici occupano memoria e spazio su disco. Quindi, mi raccomando, parsimonia.

Consiglio 7: l'efficacia di un indice è inversamente proporzionale alla lunghezza del campo. Più il campo è piccolo, più l'indice è efficiente. Se avete, nella vostra tabella, un campo *note* di 200 caratteri, creare un indice non è una buona idea.

Consiglio 8: evitate gli indici su campi che assumono solo pochi valori. Se ho un campo che può contenere ad esempio solo "*S*" per Si e "*N*" per No, un indice peggiora le prestazioni e non serve a niente.

Consiglio 9: i campi scelti come indice non dovrebbero contenere il valore *null*, perché i motori Db gestiscono male questa situazione. Ovviamente neppure la chiave primaria deve ammettere valori *null*.

TIP

Spesso, negli ambiti più vari, può risultare impossibile gestire Tabelle che non dispongono di una chiave primaria. Vi consiglio, quindi, di utilizzarla sempre nella progettazione dei vostri Db.

2.5 Principali Tipi di Dati

Ogni Server di Database possiede un lungo elenco di tipologie di Dati gestibili. In realtà i tipi più comuni (numeri, testo e date) sono molto simili, come sarà chiaro più avanti, in tutti i *motori* di Db.

2.5.1 Campi di tipo Stringa

Una **Stringa** è in generale una informazione alfanumerica di lunghezza variabile (un nome, una via, un titolo etc.). Questa informazione può essere archiviata in due modi:

1. posso stabilire un numero massimo di caratteri, e riservare *sempre* un numero di byte fisici equivalenti nel mio archivio (tipo ***char***)
2. posso stabilire un numero massimo di caratteri, ma archiviare solo quelli *effettivamente digitati* dall'utente (tipo ***varchar***)

Nel primo caso saranno aggiunti sempre tanti spazi quanto servono a raggiungere la lunghezza specificata, nel secondo invece lo spazio occupato è in funzione dei caratteri digitati. Ad esempio supponiamo di avere il campo *nome* di lunghezza max 20 caratteri; se scrivo nel campo la stringa *Carlo*, nel primo caso avrò usato 20 byte, nel secondo solo 5.

Allora, vi chiederete, visto che è più conveniente, dal lato dell'occupazione di spazio, usare il tipo *varchar*, il *char* che ci sta a fare ? In effetti il *char* si usa quando l'informazione è formata da pochi caratteri e magari deve essere anche indicizzata. Quindi il *char*, per esempio, va bene per archiviare il campo *CAP* di un indirizzo, oppure un *Codice Cliente* di 5 caratteri (che sicuramente andrà indicizzato). Il fatto è che i campi a lunghezza fissa come *char* sono più veloci da manipolare da parte del motore Db, e quindi anche più indicati nella creazione di indici.

TIP

La lunghezza di un campo stringa andrebbe attentamente valutata; da una parte, se si imposta a troppo pochi caratteri, si rischia di non poter archiviare informazioni lunghe, dall'altra ogni carattere in più significa spazio sprecato e velocità abbassata. La mia regola (ovviamente opinabile) è che se la lunghezza è inferiore a 10 caratteri uso *char*, altrimenti *varchar*. Per i campi di denominazione (nome e cognome, nome di aziende etc.) direi che 50 caratteri sono sufficienti. Negli altri casi un po' di sperimentazione non guasta.

In generale il massimo numero di caratteri archiviabile in un campo ***char*** o ***varchar*** è di 255 (meglio, la lunghezza totale del campo non può eccedere 255 byte). Se abbiamo la necessità di archiviare stringhe più lunghe è possibile usare campi di tipo ***text*** che sono molto più *capienti* (anche fino a 2^32, cioè 4.294.967.295 o 4GB).

2.5.2 Campi di tipo Numerico

I campi di tipo numerico si dividono in due grandi sotto categorie: **interi** e **decimali**.

I campi di tipo ***numerico intero***, a seconda dell'intervallo di valori che possono contenere, si dividono in varie classi. In generale i tipi più comuni occupano due (***smallint***), quattro (***integer*** o ***int***) oppure otto byte (***bigint***) di spazio. Il numero *Intero* classico occupa *quattro* byte e di solito può contenere un range numerico abbastanza ampio. Ad esempio in *MySql* un campo ***int*** può contenere da -2.147.483.648 a 2.147.483.647. Questi numeri *possono* (***signed***) oppure *non possono* (***unsigned***) contenere il segno, e quindi assumere valori negativi. Ogni motore di Db indica gli interi in modo leggermente diverso, quindi documentarsi non fa certo male.

I numeri **decimali**, a loro volta, possono dividersi in ulteriori due classi: il tipo ***numeric*** (chiamato anche ***decimal***), che comporta calcoli esatti in quanto il numero di cifre decimali è fisso, ed il tipo ***real*** o ***double precision*** che comporta calcoli arrotondati perché eseguiti in virgola mobile. Questi ultimi sono una implementazione dello standard IEEE 754 per l'aritmetica in virgola mobile. In generale è sempre opportuno usare, per i numeri decimali, il tipo *numeric.* Nella definizione di questi tipologie di dati di solito si specificano *due* valori : la *precisione* e la *scala*. La *scala* è il numero di cifre decimali da considerare, mentre la *precisione* è il numero totale di cifre che il numero può contenere (contando la parte intera più la parte decimale). Così, ad esempio, il numero 123,4567 ha una *scala* di quattro ed una *precisione* di sette. Alcuni motori di Db hanno anche un campo di tipo ***currency*** o ***money*** per l'archiviazione dei valori di *valuta*. Dove manca o dove non è utilizzabile si può usare ad esempio un tipo ***numeric(16,4)***. Se si vuole arrotondare i calcoli a due cifre decimali, va bene anche ***numeric(14,2)***.

TIP

Sarebbe sempre meglio controllare bene l'intervallo dei valori archiviabili nei tipi di dati numerici del motore di database che vogliamo usare. Ad esempio il tipo *Intero* di Ms Access (Jet) può contenere valori tra -32.728 e 32.767, e quindi equivale a *smallint* di MySql. Allo stesso modo, il tipo *currency.* o *valuta.* di Ms Access corrisponde al tipo *Decimal* con precisione 19 e scala 4 di MySql. La differenza è che mentre il campo di Access occupa sempre 8 byte, quello di MySql occupa *un numero di byte uguale alla precisione* (nel nostro caso quindi 19).

2.5.3 Campi di Tipo Data/Ora

Tutti i motori di Db hanno tipologie specifiche di campi per la manipolazione di date ed orari. I tipi più comuni si chiamano ***Time***, ***Date***, ***DateTime*** e possono, come è facilmente intuibile, contenere rispettivamente un *Orario*, una *Data* ed una *Data/Orario* completa.

Ogni prodotto però implementa diverse varianti, dall'intervallo di date che è possibile archiviare, a vari sottogruppi di tipologie (*SmallDateTime*), al formato di memorizzazione ed all'uso di *Timezone*. Quindi documentarsi è indispensabile.

2.5.4 Campi di Tipo Booleano

Questa tipologia di campi può contenere solo due "stati" logici, di solito indicati come ***true*** (vero) o ***false*** (falso). Altre rappresentazioni possibili per *true* sono *1, t, y, yes*, e per *false*: *0, f, n, no*. La rappresentazione interna è un numero intero che può assumere il valore di 0 o 1 (oppure -1). Uno stato non definito si indica col valore *null*.

2.5.5 Campi di tipo Binario

Un campo ***Binario*** può, appunto, contenere dati di tipo binario, cioè sequenze di byte. Sono usati per archiviare informazioni di tipo "grezzo" (ad esempio immagini o documenti in formato nativo), che occupano molto spazio. Non sono indicizzabili, ed andrebbero usati con cautela.

2.5.6 Campi particolari: Intero ad incremento automatico

Come abbiamo visto nel paragrafo dedicato agli indici, può essere a volte utile che il valore di un campo sia stabilito in automatico dal motore Db, secondo una progressione numerica. Questo campo può, ad esempio, rappresentare un codice univoco da assegnare alla scheda senza che noi dobbiamo preoccuparci di fare niente. Campi di questo tipo sono i candidati migliori per la definizione di *Chiavi Primarie*. In generale si tratta di un campo di tipo ***Integer*** associato alla proprietà di ***auto_increment*** (incremento automatico) o ***Identità***. Alcuni Server definiscono questo tipo di campo come ***Serial***, ma la sostanza non cambia.

2.5.7 Campi particolari: Timestamp

Il ***Timestamp*** è in generale un tipo di campo ***Data/Ora*** oppure ***Float*** aggiornato dal sistema. In ambienti multiutente è particolarmente utile per gestire le modifiche concorrenti sulla stessa tabella. Siccome alcuni *front end* per database funzionano male se in ogni tabella non è presente almeno un campo *Timestamp*, noi lo aggiungeremo sempre, tanto non costa niente. Ad esempio, *Ms Access* si rifiuta di funzionare bene su Tabelle *MySql* che non hanno il *Timestamp*. Questo, in funzione di OOo potrebbe essere ininfluente, ma certamente non vogliamo che il nostro Db in futuro non sia leggibile da qualsiasi applicazione, giusto ?

2.6 Vincoli (Constraint)

Ad un singolo Campo, oppure ad un'intera tabella, è di solito possibile applicare dei "*vincoli*" (in inglese ***constraints***). Ogni motore di Db ha delle particolarità, ma in generale possiamo avere:

- **vincoli *check***: servono a controllare i valori immessi in uno o più campi di una Tabella; ad esempio potrei definire un vincolo check che mi impedisce di immettere valori negativi in un campo numerico;
- **vincolo *not null*** : impedisce l'immissione di valori nulli nel campo;
- **vincolo *unique*** : di solito collegato ad un indice, impedisce di duplicare un valore in più righe di una stessa tabella;
- **vincolo *primary key*** : individua la chiave primaria di una tabella; include le caratteristiche di unique e not null; è di solito il campo che individua in modo univoco la singola riga;
- **vincolo *foreign keys*** : indica le chiavi esterne della tabella, come vedremo nel paragrafo dedicato all'integrità referenziale;

i vincoli vengono controllati, durante le fasi di immissione e variazione dei dati, direttamente dal motore di Db, e sono un ottimo strumento di verifica della coerenza del Database.

2.7 Il nostro Database di esempio

Per meglio illustrare i concetti che andremo ad introdurre, abbiamo bisogno di un piccolo Archivio di esempio, non troppo complesso ma nemmeno banale. Dopo una lunga (e sofferta) riflessione, ho deciso che gestiremo una **Mediateca**. Ora vi chiederete che sarà mai questo oggetto sconosciuto: bene, è un impasto di Videoteca, Biblioteca, Discoteca, Emeroteca etc. In pratica tutto quello che finisce in -teca ...

Vogliamo, cioè, creare un Archivio che ci permetta di gestire in modo flessibile qualsiasi tipo di *Media* desideriamo catalogare. Inoltre visto che alle nostre cose ci teniamo, ho stabilito che potremmo anche usare un piccolo promemoria per sapere a chi e quando abbiamo (sconsideratamente) fatto un prestito.

Trattandosi di un esempio didattico, non andremo troppo per il sottile, quindi guru dei Db trattenete commenti inopportuni. Chi vuole poi potrà studiare e migliorare....

2.7.1 Struttura degli Archivi

Cominciamo con una piccola analisi preliminare delle nostre esigenze, in modo da definire le principali informazioni che andremo a gestire. L'Archivio principale destinato a contenere le informazioni sui nostri Media (***TbMedia***) potrebbe essere strutturato così:

Campo	**Descrizione**
Id	Chiave primaria – Intero ad incremento automatico
Desc	Descrizione
TipoSupp	Tipo del supporto di memorizzazione (CD, DVD, Rivista, File, etc.)
Argom	Argomento o classificazione

Campo	Descrizione
Ubicazione	Ubicazione (Scaffale, Scatolo, Num Rivista, Percorso su Hd etc.)
Prezzo	Prezzo (di acquisto, di vendita, di prestito...), forse non necessario, ma sicuramente didattico
Note	Annotazioni libere
Prestato	Si / No

Bene, dobbiamo ora definire il tipo di campo da assegnare ad ogni informazione della nostra Tabella. La cosa più immediata, ad esempio per il campo "*TipoSupp*"o "*Argom*", sarebbe ovviamente *stringa*, magari di lunghezza 20 caratteri. Così, ad esempio, una parte del nostro archivio potrebbe essere :

Desc	TipoSupp	Argom
Pearl Jam - Binaural	CD Audio	Rock
Fromm – Avere o Essere ?	Libro	Filosofia
Harry Potter e la camera dei segreti	DVD Film	Fantasy
Peter Gabriel – Growing Up Live	DVD Musicale	Rock
Condividere risorse con Samba 3	Rivista	Linux / Samba
Rossini – 10 Ouvertures – Chailly	CD Audio	Classica

Siccome è bene, prima di progettare qualunque Database, preparare uno schema *reale* di quello che la Tabella dovrà contenere, che cosa possono suggerirci queste *righe* di esempio ?

1. I campi *TipoSupp* e *Argom* contengono stringhe ripetitive, cioè molte righe avranno ad esempio in *TipoSupp* la stringa *DVD Musicale*; è uno spreco enorme di spazio.
2. Se scriviamo *manualmente* in *TipoSupp* le varie classificazioni, sarà facile sbagliare e quindi inserire ad esempio per errore *DVD Musocale*; se devo poi selezionare tutti i *DVD Musicali*, quella riga sarà scartata.
3. Come facciamo a ricordare a memoria tutte le classificazioni usate? Se la mediateca è ampia, è quasi impossibile.

Allora come fare ? Semplice, basta creare un'altra Tabella da usare come *classificatore*, del tipo:

TbSupporti -> Classificazione dei Media per Tipo di Supporto

Identificatore	Descrizione
1	DVD Film
2	DVD Musicale
3	CD Audio
4	Rivista

Identificatore	Descrizione
5	Libro

Così il nostro Archivio apparirà in questa forma :

Desc	TipoSupp	Argom
Pearl Jam - Binaural	3	Rock
Fromm – Avere o Essere ?	5	Filosofia
Harry Potter e la camera dei segreti	1	Fantasy
Peter Gabriel – Growing Up Live	2	Rock
Condividere risorse con Samba 3	4	Linux / Samba
Rossini – 10 Ouvertures – Chailly	3	Classica

I vantaggi di questa organizzazione sono subito evidenti :

1. risparmio di spazio (numeri interi al posto di stringhe);
2. velocità di immissione (devo solo selezionare, nel campo *TipoSupp*, un valore da una Tabella);
3. velocità operativa (un indice su "*TipoSupp*" sarà molto più efficiente);
4. riduzione della possibilità di errore.

Ho dunque eseguito, magari senza saperlo, quella che si chiama una **normalizzazione** del Database. Tutto quanto spiegato vale anche per il campo *Argom*, ed in generale per tutti i campi destinati a contenere informazioni riconducibili ad un insieme ben definito e non troppo ampio. Nella fase operativa vedremo come mettere in *relazione* le due tabelle, e quindi sfruttare al meglio il Db normalizzato. In Appendice troverete lo schema completo del nostro Database di esempio.

TIP

In questo esempio abbiamo usato un campo di tipo *intero* per collegare le informazioni contenute nelle due Tabelle. Questo non è obbligatorio. Ci sono molti casi in cui i campi collegati sono di tipo *stringa*. La regola è che comunque tutti i campi usati nelle relazioni DEVONO essere indicizzati (come vedremo più avanti) e quindi gli *interi* sono più efficienti

Tecnica

Quella illustrata è una relazione tra Tabelle di tipo *uno a molti*, cioè un singolo valore di una riga di *TbSupporti* ("3", equivalente a "CD Audio") appare in molte righe di *TbMedia*. Altri tipi di relazione sono *uno a uno* e *molti a molti*. Comprendere come

> funzionano le relazioni tra Tabelle nei Database può sembrare complicato, ma alla fine i risultati sono assai soddisfacenti. Nel nostro caso il campo *TipoSupp* di *TbMedia* fa riferimento alla *chiave primaria* di un'altra Tabella, cioè *TbSupporti*, quindi *TipoSupp* è una **chiave esterna** (**foreign key**) per *TbMedia*. Una delle regole che indicano se un motore di Db può definirsi relazionale o meno riguarda proprio le chiavi esterne. Ne riparleremo più avanti nell'esaminare la gestione dell'integrità referenziale.

Ora descriveremo nel dettaglio la struttura dei nostri archivi. Notate che:

- in ogni tabella esiste un campo di tipo **id** (*identificatore*) di tipo **integer auto_increment** *assegnato come Chiave Primaria;*
- i *nomi* dei campi hanno un prefisso che indica la tabella di appartenenza (ad es. *MDes*, dove M sta per Media e Des per descrizione), salvo quelli che fanno riferimento ad altre Tabelle (*chiavi esterne*) che assumono lo stesso nome della Tabella di appartenenza, e sono indicati in corsivo; questo non è obbligatorio, ma evita di fare confusione nell'uso del linguaggio SQL;
- il *tipo di campo* viene indicato con la sintassi standard SQL; alcuni motori di Db potrebbero avere tipologie leggermente diverse;
- in ogni tabella abbiamo aggiunto un campo di tipo *timestamp*;
- l'ultima colonna indica se al campo è associato un indice; **pk** sta per *Primary Key*, cioè quel campo è la chiave primaria.

2.7.2 Tabella TbMedia -> Archivio Principale

Dunque la Tabella **TbMedia** sarà il nostro Archivio principale. In base alle considerazioni precedenti, la struttura potrebbe essere:

Campo	Tipo di Campo	Commenti	Idx
MId	Integer auto_increment	Identificatore - Chiave primaria	Pk
MDes	Varchar(100)	Descrizione max 100 caratteri	*
SuppId	Integer	Identificatore del Supporto	*
ArgId	Integer	Identificatore dell'Argomento	*
MUbic	Varchar(50)	Ubicazione max 50 caratteri	
MPrezzo	Decimal(14,2)	Prezzo in Euro (max due cifre dec)	
MTs	Timestamp	Timestamp per la tabella	

2.7.3 Tabella TbSupporti -> Tipologie dei Supporti

Questa Tabella associa ad un numero intero la descrizione del Supporto corrispondente. Struttura di **TbSupporti**:

Campo	Tipo di Campo	Commenti	Idx
SuppId	Integer auto_increment	Identificatore - Chiave primaria	Pk
SuppDes	Varchar(30)	Descrizione max 30 caratteri	*
SuppTs	Timestamp	Timestamp per la tabella	

2.7.4 Tabella TbArgomenti -> Tipologie degli Argomenti

Questa Tabella associa ad un numero intero la descrizione dell'Argomento corrispondente. Struttura di **TbArgomenti**:

Campo	Tipo di Campo	Commenti	Idx
ArgId	Integer auto_increment	Identificatore - Chiave primaria	Pk
ArgDes	Varchar(30)	Descrizione max 30 caratteri	*
ArgTs	Timestamp	Timestamp per la tabella	

2.7.5 Tabella TbUtenti -> Archivio degli Utenti dei Prestiti

Siccome desideriamo gestire anche eventuali prestiti per la nostra Mediateca, può essere opportuno archiviare le informazioni degli Utenti, secondo questo schema (**TbUtenti**):

Campo	Tipo di Campo	Commenti	Idx
UtId	Integer auto_increment	Identificatore - Chiave primaria	Pk
UtDen	Varchar(50)	Denominazione max 50 caratteri	*
UtVia	Varchar(50)	Indirizzo max 50 caratteri	
UtCit	Varchar(100)	Città max 100 Caratteri	
UtTel	Varchar(20)	Telefono max 20 Caratteri	
UtTs	Timestamp	Timestamp per la tabella	
UtDNas	Data	Data di Nascita	
UtImg	Immagine	Foto dell'Utente	
UtCodFis	Varchar(16)	Codice Fiscale	
UtSesso	Char(1)	Sesso (M o F)	
UtTessera	Integer(1)	Tesserato (valore Logico, 0 o 1)	

Non tutti i campi sarebbero strettamente necessari, ma alcuni sono indispensabili per comprendere meglio l'uso avanzato dei Formulari.

2.7.6 Tabella TbPrestiti -> Archivio dei prestiti e delle restituzioni

Questa Tabella contiene l'elenco cronologico dei prestiti e delle restituzioni, collegato con la Tabella dei Media e con la Tabella degli Utenti. Struttura di **TbPrestiti**:

Campo	Tipo di Campo	Commenti	Idx
PreId	Integer auto_increment	Identificatore - Chiave primaria	Pk
PreData	Date	Data del prestito	*
UtId	Integer	Identificativo Utente (TbUtenti)	*
MId	Integer	Identificativo Media (TbMedia)	*
PreDataR	Date	Data di restituzione	
PreTs	Timestamp	Timestamp per la tabella	

2.8 Integrità Referenziale

In questo paragrafo cercheremo di approfondire un aspetto dei motori di Db assai importante, che ogni sviluppatore (od anche un semplice utente) dovrebbe conoscere bene per evitare che il proprio Database incorra in seri problemi nella gestione in ambienti di produzione: l'*integrità referenziale*.

tbmedia.MId	tbmedia.MDes	tbmedia.ArgId	tbargomenti.ArgId	tbargomenti.ArgDes
2	Joss Stone - The Soul Session	1	1	Rock
3	Bad Boys II	18	18	Avventura
4	KDE 3.2 - Novità Major Release	16	16	Linux
5	Montalbano - Il Ladro di merendina	5	5	Gialli
6	Benni - La compagnia dei celestini	17	17	Romanzo
7	Guccini - Ritratti	3	3	Pop
454	Il Signore degli Anelli	15	15	Fantasy
455	Montalbano - La forma dell'acqua	5	5	Gialli
456	Pearl Jam - Ten	1	1	Rock
457	Cornwell - L'Ultimo Distretto	5	5	Gialli

Figura 2.8.1: Relazione tra TbMedia e TbArgomenti

Per chiarire i termini della questione, diamo un'occhiata al nostro Db di test (*Mediateca*). La Tabella *TbMedia*, che contiene l'archivio dei nostri supporti, per la classificazione per argomento fa riferimento alla Tabella *TbArgomenti*, attraverso il campo *ArgId*.

Consideriamo, per semplicità, il nostro Archivio con alcuni dati inseriti, come nella figura precedente: i campi di *TbMedia* sono sulla sinistra, quelli di *TbArgomenti* sulla destra. Quella che vedete in *gergo* Db si chiama ***query***, e viene anche indicata nella terminologia di OpenOffice come ***ricerca***.

Nel nostro esempio se si specifica il valore "5" nel Campo *ArgId* di *TbMedia*, la riga sarà associata alla voce "Gialli" che compare in *TbArgomenti*. *Ora provate a pensare che per errore*

venga cancellato il Record con ArgId "5" in TbArgomenti. In questo caso tutti i Record che hanno come riferimento "Gialli" in TbMedia non avrebbero più un argomento associato, con conseguenze imprevedibili sulla coerenza dei dati di tutto il Db. Infatti *ArgId* è una **chiave esterna** (**foreign key**) per la Tabella *TbMedia*, collegata a *TbArgomenti*. Quindi *TbArgomenti* si indica anche come tabella **parent** (*genitore*) e *TbMedia* come tabella **child** (*figlio*); *questo sta anche a significare che per ogni chiave ArgId "parent" in TbArgomenti, possono esistere più chiavi "child" in TbMedia (cioè molti supporti possono essere classificati come "Gialli"), tipica relazione uno a molti*.

Sarebbe, inoltre, MOLTO opportuno che nel campo *ArgId* di *TbMedia* possano essere immessi solo valori GIA' PRESENTI in *TbArgomenti*, cioè non dovrebbe essere consentito inserire argomenti inesistenti, o, se volete, figli senza genitori.

Molti motori di Db hanno a disposizione un meccanismo di controllo che impedisce la cancellazione o la modifica delle chiavi esterne, o comunque permette di stabilire regole precise per la gestione di queste eventualità. Questi motori si occupano perciò di **preservare l'integrità referenziale del Database**.

2.9 Server di Database

A questo punto dovrebbe essere chiaro che cosa intendiamo quando si parla di *Database Relazionale*. Forse è meno chiaro come sfruttare i concetti che abbiamo appena illustrato, ovvero quale prodotto usare, casomai volessimo creare ad esempio la nostra Mediateca. In realtà quello dei Database è un mondo estremamente vasto e vario: esistono software di tutte le taglie e per tutte le tasche che, più o meno, possono ricondursi alla categoria oggetto del nostro interesse. Una prima classificazione può essere fatta in base alla "vocazione" del prodotto: **server** o **desktop**.

In generale un **Database Server** è un programma che rimane in "ascolto" su una porta del PC, pronto ad eseguire i "comandi" impartiti da altre applicazioni. Quindi, di solito, se un Server è in funzione sul PC potreste tranquillamente non notare nulla di strano, salvo magari qualche icona nella tray bar. Per usare queste tipologie di prodotti è necessario un programma *client* che permetta di interagire col *server*. Tutti i Server di Db sono dotati di una serie di tool *client,* quasi sempre a linea di comando. Alcuni si servono anche di apposite interfacce grafiche che semplificano la vita degli utenti. Sono disponibili molti prodotti commerciali di questo tipo, alcuni davvero ottimi, ma dal costo non sempre accessibile: **Oracle**, **Sybase**, **Microsoft SQL Server**. Sul versante *Open Source* le notizie sono, per una volta, molto buone: esistono almeno due prodotti che hanno poco da invidiare alle soluzioni a pagamento, **MySql** e **PostgreSql**. Recentemente si sta affermando anche un altro prodotto Open Source chiamato **Firebird**, che è una "rielaborazione" del vecchio motore *Interbase* di Borland.

Un **Database per Desktop** in genere ha un approccio diverso: per prima cosa quasi sempre include un suo motore di Db, ma spesso può collegarsi anche anche ad altri prodotti;

poi dispone della possibilità di sviluppare facilmente *interfacce* verso i dati (maschere, ricerche e selezioni, report etc.); infine a volte include un vero e proprio sistema di sviluppo di applicazioni, completo di linguaggio di alto livello. Esempi di questa tipologia di software possono essere Microsoft **Access**, **Filemaker**, **Foxpro**, **Paradox** ma anche il nostro **OpenOffice Base**. Questi prodotti possono anche essere usati come *ponte* verso altri motori di Db, compresi molti Server, ed è questo che vedremo nel seguito con riferimento appunto ad OpenOffice.

2.10 Modalità di accesso ai Dati

Un Database Server deve prevedere delle interfacce di accesso ai propri dati, in modo che applicazioni di terze parti possano *interagire* col server stesso. Queste interfacce sono di tipo anche molto diverso tra loro. Abbiamo già detto dei programmi a linea di comando. Esistono poi appositi *moduli*, sviluppati in vari linguaggi di programmazione, che permettono alle applicazioni l'accesso e la manipolazione dei dati. Un esempio è il Modulo di *PHP* per *MySql*, ma ne troviamo decine per gli ambienti di programmazione più diversi.

Sono inoltre disponibili dei protocolli standard, sviluppati sul concetto di *Driver* (un po' come per le Schede Grafiche). In questo momento esistono due principali classi di driver: **ODBC** e **JDBC**.

ODBC (Open DataBase Connectivity) è lo standard più usato: esistono versioni per Windows e Linux e le prestazioni sono soddisfacenti. La presenza di un *Driver ODBC* per un Server garantisce l'accesso ai Dati a qualunque applicazione sia conforme allo standard (a cominciare da OpenOffice per finire ai vari moduli di Microsoft Office compreso Access).

JDBC (Java DataBase Connectivity) è più o meno la stessa cosa, rivolta però ad ambienti di sviluppo Java. Questo non toglie che driver JBDC siano utilizzabili anche da altre applicazioni, come OOo.

Esistono ancora altre modalità di interconnessione specifiche per ambienti Windows (come *ADO Db*) di cui però mi sembra inutile ora occuparsi.

2.11 Componenti di un Database

Abbiamo già descritto due elementi essenziali (anzi direi indispensabili) di ogni Database: le *Tabelle* e gli *Indici*. A questi ogni motore di Db aggiunge caratteristiche proprie che permettono una migliore gestione delle Basi di Dati. Parliamo di *Viste*, *Triggers*, *Stored Procedure*. Non è il caso qui di approfondire molto questi argomenti, ma forse è il caso di parlarne brevemente.

2.11.1 View (Vista)

Una ***Vista*** (*view*) è una particolare struttura che raccoglie ed organizza campi provenienti da una oppure più tabelle messe in relazione tra di loro. Si tratta, come vedremo,

sostanzialmente di una query di selezione archiviata sul Server, che però può essere usata come una Tabella reale (quindi a volte è anche aggiornabile). Una *Vista* è particolarmente utile quando è necessario organizzare i dati secondo precise regole, lasciando fare tutto il lavoro al Server (così da ottenere velocità e precisione). A titolo di esempio si potrebbe creare una *vista* che elenca tutti gli elementi di *TbMedia* completi di *Argomento* e *Supporto* ordinata per Descrizione; in questo caso quindi, tre tabelle, collegate tra loro da relazioni, che vengono raggruppate in una sola entità. Il risultato sarebbe più o meno:

MId	MDes	SuppDes	ArgDes
459	Afterhours - Ballate per piccole iene	CD Audio	Rock
3	Bad Boys II	DVD Video	Avventura
6	Benni - La compagnia dei celestini	Libro	Romanzo
457	Cornwell - L'Ultimo Distretto	Libro	Giallo
458	De Gregori - Pezzi	CD Audio	Rock
7	Guccini - Ritratti	CD Audio	Pop
454	Il Signore degli Anelli	DVD Video	Fantasy
2	Joss Stone - The Soul Session	CD Audio	Pop
4	KDE 3.4 - Novità Major Release	Rivista	Linux
463	Ligabue - Roma Stadio Olimpico	CD Audio	Rock
462	Marillion - Marbles on the Roads	DVD Video	Rock
5	Montalbano - Il Ladro di merendine	Libro	Giallo
455	Montalbano - La forma dell'acqua	Libro	Giallo
456	Pearl Jam - Ten	CD Audio	Rock
460	Springsteen - Devils & Dust	DVD Video	Rock

Figura 2.11.1: Una Vista che comprende tre Tabella

2.11.2 Trigger

Un ***Trigger*** è un'azione eseguita al verificarsi di una condizione riguardante una Tabella oppure un singolo Campo. Il *Trigger* è comunque un oggetto del Database, quindi deve possedere un nome univoco. Di solito, nella definizione, si indica, oltre all'oggetto a cui si applica, l'azione che attiva il Modificatore (***insert***, ***delete*** o ***update***) ed il momento di esecuzione (***before*** oppure ***after***). Così, ad esempio, se nella mia tabella ho un campo "*Prezzo Iva esclusa*" ed uno "*Prezzo Iva inclusa*", posso creare un *Trigger* che, ad ogni modifica del primo campo calcola immediatamente il secondo, senza ulteriori interventi da parte dell'utente.

2.11.3 Stored Procedure

Una ***Stored Procedure*** è una subroutine scritta in un linguaggio di programmazione (generalmente SQL) ed archiviata sul Server, che può essere richiamata in modo semplice da un Client. In generale una Stored Procedure può accettare parametri, ed il vantaggio (ovvio) è che tutte le elaborazioni avvengono "lato Server" senza appesantire il Client ed il traffico di rete. In alcuni casi le *Stored Procedure* sono particolarmente utili: ad esempio quando programmi client scritti in linguaggi diversi e funzionanti su piattaforme diverse devono

eseguire le stesse operazioni; in questo caso possono limitarsi ad invocare la Stored Procedure disponibile sul Server.

2.12 Parliamo un po' di Structured Query Language

Il Linguaggio *SQL*, nelle sue caratteristiche di base, è uno Standard ANSI/ISO. Come spesso accade, però, ogni azienda fornitrice di Db server ritiene opportuno aggiungere *estensioni* allo standard, in base alle caratteristiche del proprio prodotto. Come è facile immaginare, queste *estensioni*, non avendo una base concordata, spesso sono incompatibili tra loro, quindi un comando *SQL* valido, ad esempio, per *Sybase* non funzionerà con *MySql*, per la gioia di tutti gli sviluppatori. Quando si studia un motore di Db è quindi necessario capire bene quali sono le particolarità della sintassi *SQL* che si va ad utilizzare. In ogni caso una base comune esiste, quindi quanto scriverò nel seguito è in generale valido per tutti i motori di Db.

Come tutti i linguaggi di programmazione, anche *SQL* ha una serie di regole per quanto riguarda la sintassi, che sarebbe inutile descrivere qui.

SQL è stato inizialmente progettato per eseguire "ordini" digitati da una linea di comando. Quindi una istruzione *SQL* esegue *immediatamente* una operazione sul Database selezionato. Possiamo, per comodità, suddividere istruzioni in gruppi, a seconda delle scopo a cui sono dedicate, e quindi :

1. Amministrazione del Database
2. Definizione dei dati
3. Manipolazione dei dati
4. Gestione delle repliche
5. Gestione delle transazioni

Ai nostri fini, considerato che la parte amministrativa può essere benissimo gestita con uno qualsiasi degli strumenti grafici disponibili, e che le repliche e le transazioni certo esulano dallo scopo di questo manuale, converrà discutere solo della definizione e manipolazione dei dati.

Le istruzioni per la **definizione dei dati** (meglio sarebbe dire per la definizione della *struttura* dei dati) comprendono tutti gli strumenti adatti a ***creare***, ***modificare*** e ***cancellare tabelle***, ***indici*** e ***chiavi esterne***. Anche queste operazioni, come abbiamo già visto, possono essere eseguite tramite interfacce grafiche, quindi l'uso di questi istruzioni dirette è raro, almeno nel nostro ambito. A titolo di esempio, vi riporto l'istruzione SQL che crea la tabella *TbMedia* del nostro Db, completa di indici e di chiave esterna in *MySql*:

```
CREATE TABLE `tbmedia` (

  `MId` int(11) unsigned NOT NULL auto_increment,
  `MDes` varchar(100) NOT NULL default '',
  `SuppId` int(10) unsigned NOT NULL default '0',
```

```
  `ArgId` int(10) unsigned NOT NULL default '0',
  `MUbic` varchar(100) default '',
  `MPrezzo` decimal(14,2) unsigned NOT NULL default '0.00',
  `MTs` timestamp(14) NOT NULL,

  PRIMARY KEY  (`Mid`),

  KEY `Supp` (`SuppId`),
  KEY `Des` (`MDes`),
  KEY `Arg` (`ArgId`),

  CONSTRAINT `tbmedia_ibfk_1` FOREIGN KEY (`ArgId`) REFERENCES `tbargomenti`
    (`ArgId`)

  CONSTRAINT `tbmedia_ibfk_2` FOREIGN KEY (`SuppId`) REFERENCES `tbsupporti`
    (`SuppId`)

) TYPE=InnoDB;
```

A titolo di esempio, vi riporto anche la stessa istruzione, ma relativa a **PostgreSQL**:

```
CREATE TABLE "TbMedia" (

  "MId" int4 NOT NULL DEFAULT nextval('public."TbMedia_MId_seq"'::text),
  "MDes" varchar(100),
  "SuppId" int4 NOT NULL DEFAULT 0,
  "ArgId" int4 NOT NULL DEFAULT 0,
  "MUbic" varchar(50),
  "MPrezzo" numeric(14,2) NOT NULL DEFAULT 0,
  "MTs" timestamp,

  CONSTRAINT "Pk" PRIMARY KEY ("MId"),

  CONSTRAINT "ExtKArg" FOREIGN KEY ("ArgId") REFERENCES "TbArgomenti"
    ("ArgId") ON UPDATE RESTRICT ON DELETE RESTRICT,

  CONSTRAINT "ExtKSupp" FOREIGN KEY ("SuppId") REFERENCES "TbSupporti"
    ("SuppId") ON UPDATE RESTRICT ON DELETE RESTRICT
)
WITHOUT OIDS;
```

Le differenze non sono poi molte, ma quanto basta per rendere ***incompatibili*** i due comandi. Come si può notare, un comando (statement) *SQL* è, di solito, composto da una parola chiave dal nome sufficientemente esplicativo, seguita da uno o più parametri. L'uso delle parentesi serve a "passare" informazioni, separate da virgole, al motore di Db (come se fossero i parametri di una chiamata di funzione). Le stringhe sono contenute in *virgolette singole o doppie (dette anche apici)*. L'istruzione si chiude, di solito, con un punto e virgola (;).

Così, ad esempio, se volessi modificare il tipo di dati del campo *MUbic* da *varchar(100)* a *varchar(50),* l'istruzione sarebbe, in *MySql*:

```
ALTER TABLE `mediateca`.`tbmedia` MODIFY COLUMN `MUbic` VARCHAR(50);
```

In questo caso è indicato esplicitamente il Database (*Mediateca*) che contiene la tabella. Gli statement per la definizione dei dati non sono molti, ma in alcuni casi possono raggiungere una notevole complessità. **Se possibile, usate l'interfaccia grafica.**

Per la manipolazione dei dati, invece, i comandi principali sono soltanto quattro e precisamente:

```
DELETE
INSERT
UPDATE
SELECT
```

Non vi tedierò con la sintassi precisa di ogni istruzione, e dal nome si intuisce facilmente lo scopo a cui sono dedicate. Voglio però farvi notare che di tutte, la sola che ritorna un risultato è ***SELECT***.

DELETE, ***INSERT*** e ***UPDATE*** servono rispettivamente a cancellare, aggiungere, modificare i dati presenti ne Db. Così ad esempio l'istruzione di *MySql*:

```
DELETE FROM tbmedia WHERE ArgId=98;
```

cancella dalla tabella TbMedia tutti i record con ArgId uguale a 98. Notate anche che, *se non ci sono ambiguità e non avete usato caratteri speciali, gli apici possono essere omessi*. Io, per comodità, uso le maiuscole per gli elementi della sintassi, e le minuscole per gli argomenti ed i parametri, ma ognuno può fare come vuole. Grande importanza in *SQL* assume la clausola ***WHERE*** ("*DOVE*"), che permette di limitare l'ambito di applicazione dell'istruzione ad un sottoinsieme di record che rispettano determinate condizioni. Se eseguiamo lo statement precedente, come vedremo, dalla linea di comando, *MySql* risponderà con qualcosa del tipo "*XX Rows Affected*", cioè col numero di Record cancellati. *Quindi in generale* ***DELETE***, ***INSERT***, *ed* ***UPDATE*** *modificano il contenuto del Db, e ritornano solo il numero di record modificati*.

SELECT, invece, è tutta un'altra storia. ***SELECT*** serve a "selezionare" una parte di record ed a mostrare solo le informazioni che ci servono. Ad esempio :

```
SELECT  MDes, ArgId, MId FROM tbmedia
WHERE ArgId=5;
```

ha come risultato :

MDes	ArgId	MId
Montalbano - Il Ladro di merendine	5	5
Montalbano - La forma dell'acqua	5	455
Cornwell - L'Ultimo Distretto	5	457

Figura 2.12.1Risultato della Query

abbiamo cioè chiesto di selezionare (***SELECT***) i campi *Mdes*, *ArgId*, *MId* dalla tabella *TbMedia*, ma solo quelli che hanno *ArgId uguale a 5*. Da quanto detto, a seconda del contesto, ***SELECT*** può "ritornare" (cioè appunto selezionare) anche migliaia di record, ed è lo strumento ideale per interrogare il Db. L'esempio che abbiamo fatto è, in sostanza, banale, perché ***SELECT*** ha una sintassi abbastanza complessa; infatti, dal manuale di *MySql* :

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr, ...
    [INTO OUTFILE 'file_name' export_options
      | INTO DUMPFILE 'file_name']
    [FROM table_references
      [WHERE where_definition]
      [GROUP BY {col_name | expr | position}
        [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_definition]
      [ORDER BY {col_name | expr | position}
        [ASC | DESC] , ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      [PROCEDURE procedure_name(argument_list)]
      [FOR UPDATE | LOCK IN SHARE MODE]]
```

e da quello di *PostgreSQL*:

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    * | expression [ AS output_name ] [, ...]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY expression [, ...] ]
    [ HAVING condition [, ...] ]
    [ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]
    [ ORDER BY expression [ ASC | DESC | USING operator ] [, ...] ]
    [ LIMIT { count | ALL } ]
    [ OFFSET start ]
    [ FOR UPDATE [ OF table_name [, ...] ] ]
```

quindi, ad esempio, possiamo specificare più tabelle legate da relazioni, raggruppamenti, ordinamenti, limiti e molto altro. Vi consiglio perciò di studiare l'argomento in modo approfondito, se volete divertirvi con i server di Database.

Tecnica

In sostanza una istruzione ***SELECT*** non è che una "richiesta" al motore di Db, e quindi in inglese può indicarsi anche col termine ***QUERY***. Più precisamente una ***SELECT*** viene indicata come **QUERY DI SELEZIONE**. Per estensione, ***UPDATE***, ***DELETE*** ed ***INSERT*** vengono indicate come **QUERY DI COMANDO**. Per inciso, è questa la terminologia adottata da *MS Access*. *OpenOffice* invece indica una ***SELECT*** col termine di **RICERCA**, ed è possibile con lo stesso strumento eseguire anche *QUERY DI COMANDO*.

3. Finalmente OOo....

3.1 Come OpenOffice si collega ai Database

Fino alla *versione 1.X.X*, OOo ha permesso la consultazione, la modifica e l'integrazione nei documenti dei Db solo attraverso apposite interfacce che il programma chiama "**sorgenti dati**". In pratica, se si vogliono usare i dati di un qualsiasi Db, si deve prima creare una apposita "*sorgente dati*", che indica ad OpenOffice come "dialogare" col Db stesso. Una volta stabilita questa connessione, il programma integra alcuni interessanti strumenti che permettono di ottenere risultati notevoli anche senza conoscere (quasi) nulla del motore di Db utilizzato. Dalla **versione 2.0**, OOo integra anche un motore di Db interno, basato sul prodotto Open Source **HSQL,** che evita il collegamento con sorgenti dati esterne.

OpenOffice 2.0 utilizza due fondamentali modalità di accesso a Database Server esterni: **ODBC** (*Open DataBase Connectivity*) e **JDBC** (*Java DataBase Connectivity*). In genere per ogni tipologia di Server sono disponibili entrambi i Driver: quale usare dipende (come vedremo) dal contesto. I Driver *ODBC* sono di solito più veloci e performanti, ma non sempre possono essere utilizzati in Linux; *JBDC* si appoggia invece sulle macchine virtuali Java, quindi è disponibile senza problemi in Linux ma è assai meno performante del collega.

3.2 ODBC

Lo standard *ODBC* prevede, per l'accesso ad uno specifico Database, la creazione di un "**DSN**", o "**Data Source Name**". In pratica si tratta di una interfaccia di configurazione di sistema che contiene informazioni sull'Archivio a cui bisogna collegarsi. Infatti i server di Db, possono gestire contemporaneamente molti Database diversi, ed è quindi necessario indicare, oltre al tipo di "motore" che vogliamo usare, anche esattamente "a cosa" vogliamo avere accesso.

3.2.1 ODBC in Windows

In *Windows*, i DSN vengono gestiti da un apposito programma presente nel Pannello di Controllo, nella voce "*Strumenti di Amministrazione*": "**Origine Dati ODBC**". Alla partenza, il modulo mostra una finestra come quella in figura.

Come si vede, esistono tre tipi di *DSN*: "**utente**", "**di sistema**" e "**su file**". Tralasciando l'ultimo (poco importante ai nostri fini), è importante notare la differenza tra i primi due. Il "*DSN Utente*" ha valore solo per l'utente che lo crea, mentre quello "*di sistema*" vale per tutti gli utenti del Computer. Per il resto la procedura da seguire è assolutamente identica.

Per ottenere una connessione funzionante ad un Database, è necessario disporre del Driver apposito. Un elenco dei Driver disponibili sul sistema è contenuta nella Tab **Driver**.

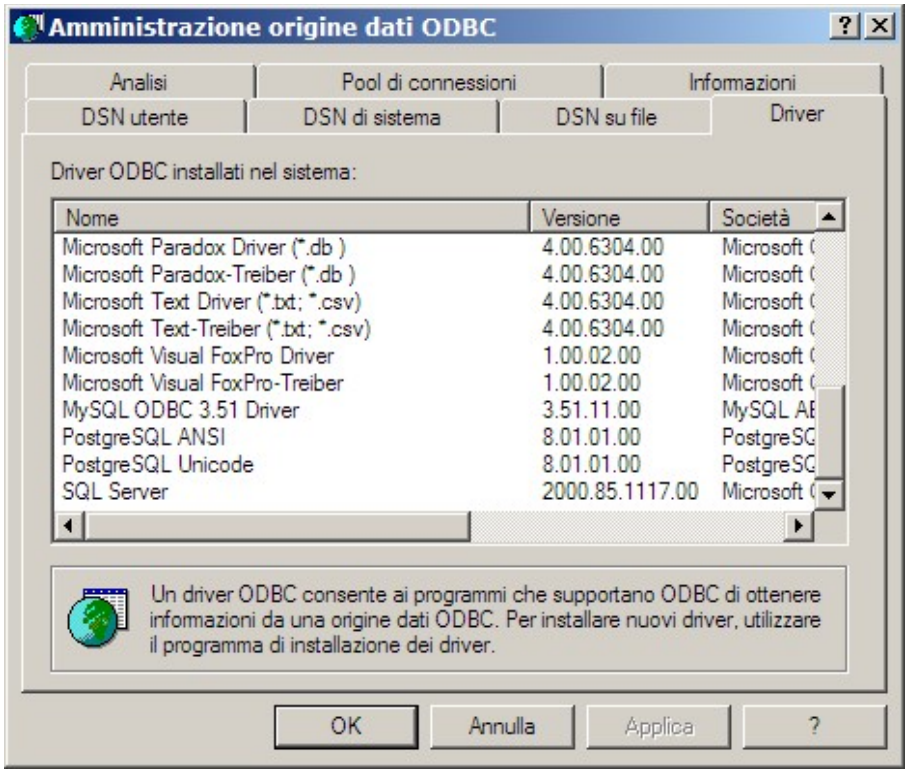


Figura 3.2.1: Driver ODBC in Windows

I parametri da inserire per un DSN sono diversi per ogni tipologia di Database Server disponibile. In figura, ad esempio, un DSN per MySQL.

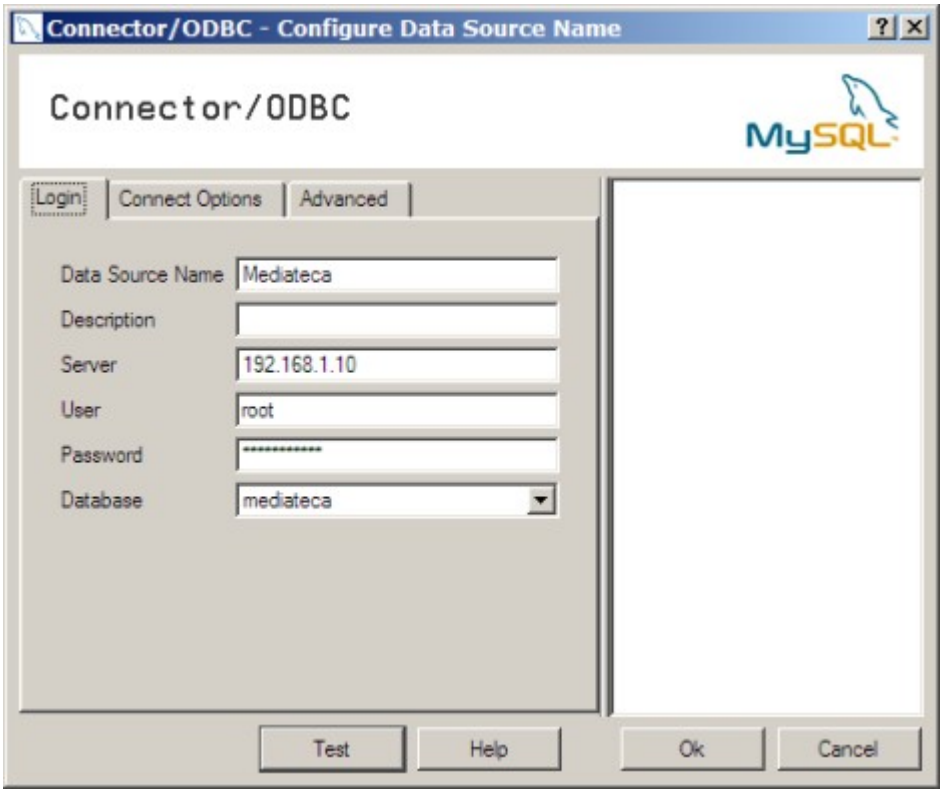


Figura 3.2.2: DSN per MySQL

3.2.2 ODBC in Linux

In Linux la configurazione è simile. Il pacchetto si chiama **unixODBC** è può essere scaricato da Sourceforge, se non è disponibile nella vostra distro. Serve sia l'RPM delle librerie (***unixODBC-2.2.XX-1.i386***), sia l'interfaccia grafica, che semplifica molto la configurazione (***unixODBC-gui-qt-2.2.XX.i386***). Dopo l'installazione, il programma da lanciare è ***/usr/bin/ODBCConfig*** che si presenta in modo molto simile alla sua controparte Windows.

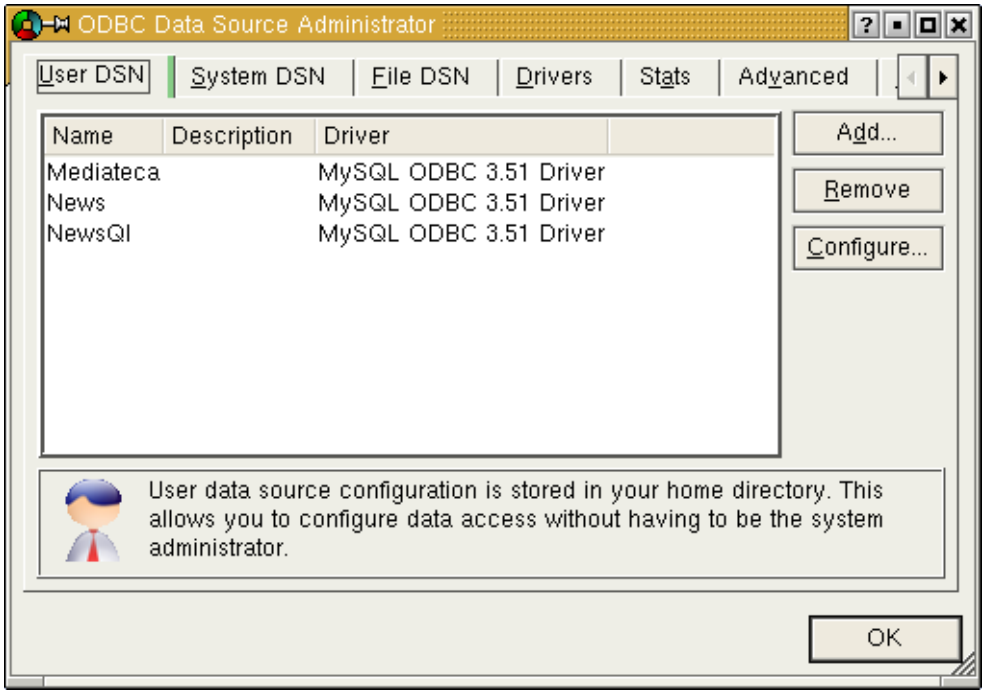


Figura 3.2.3: Configurazione del DSN in Linux

Più avanti vedremo come configurare i DSN per ogni Server di Database.

3.3 JDBC

Un Driver **JDBC**, di solito, si presenta come un semplice file .jar, che può essere aggiunto alla configurazione *JAVA* del sistema (file *java.ini*), oppure usato in modo autonomo da OOo. Il sistema più semplice è creare una Cartella che contiene i file JDBC, e quindi configurare OpenOffice per l'uso: vedremo in dettaglio questa procedura nei prossimi capitoli.

3.4 Il Wizard del Modulo "Base"

Se da qualsiasi altro Modulo di OOo 2.0 scegliamo "*File -> Nuovo -> Database*", oppure se dal Menu dei Programmi scegliamo il Modulo "*Base*", il programma avvia un Wizard che permette l'impostazione guidata di un nuovo archivio. La prima scelta che ci viene proposta è quella tra "*Crea un nuovo Database*" e "*Collega ad un Database esistente*". Nel primo caso sarà

creato un documento vuoto, con la possibilità di gestire i dati col motore interno *HSQL*, nel secondo dovremo scegliere il tipo di Dati da gestire.

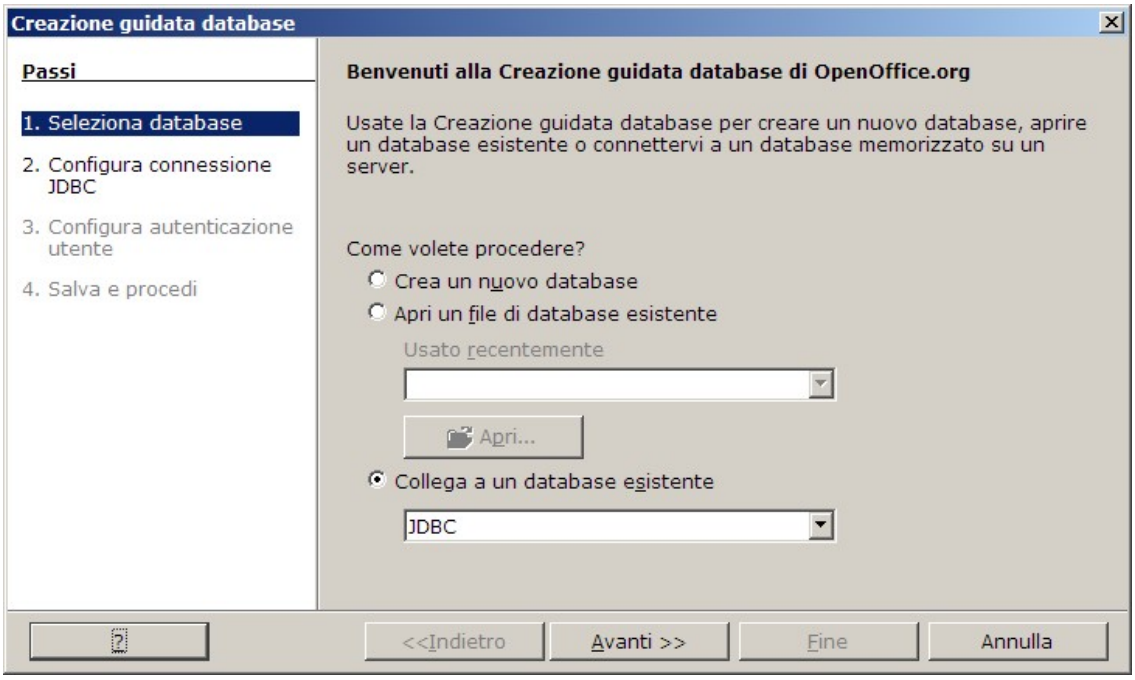


Figura 3.4.1: Creazione guidata del Database

Le tipologie di Database a cui è possibile collegarsi sono molte, come si vede in figura. A seconda della scelta, OOo apre la finestra necessaria all'impostazione dei parametri di collegamento.

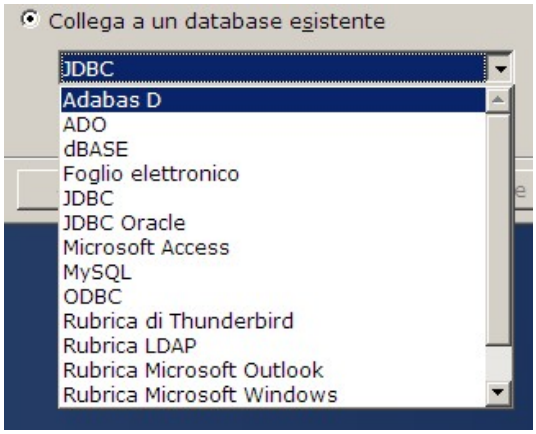


Figura 3.4.2: Collegamenti a sorgenti dati esterne

Qualunque sia la tipologia di dati che vogliamo gestire, alla fine del Wizard OOo ci chiederà se vogliamo ***registrare*** il *Database* (o meglio la *Sorgente Dati*). La registrazione implica che il Db viene riconosciuto come sorgente dati valida dagli altri moduli di OpenOffice, e quindi viene elencato alla pressione del tasto **F4** (Sorgenti Dati), ad esempio in Writer.

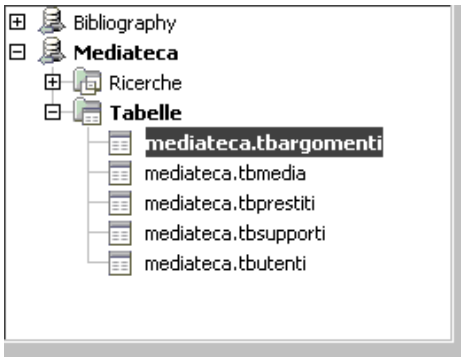


Figura 3.4.3 La sorgente Dati registrata

TIP

Cosa fare se abbiamo **dimenticato di *"registrare"*** il Database in OpenOffice ? In questo caso alla pressione del tasto F4 il Db non comparirà nell'elenco e non potrà essere utilizzato come sorgente dati. Ma niente paura.... Basta scegliere la voce "Opzioni" dal Menu "Strumenti" e, nell'elenco a sinistra, selezionare "OpenOffice Base" e poi "Database". Dalla finestra a destra potremo "registrare" la nostra sorgente dati semplicemente usando il pulsante "Nuovo..." e specificando il nome del file del nostro Db.

In ogni caso, alla fine del Wizard, dovrebbe apparire la finestra principale del Modulo Base:

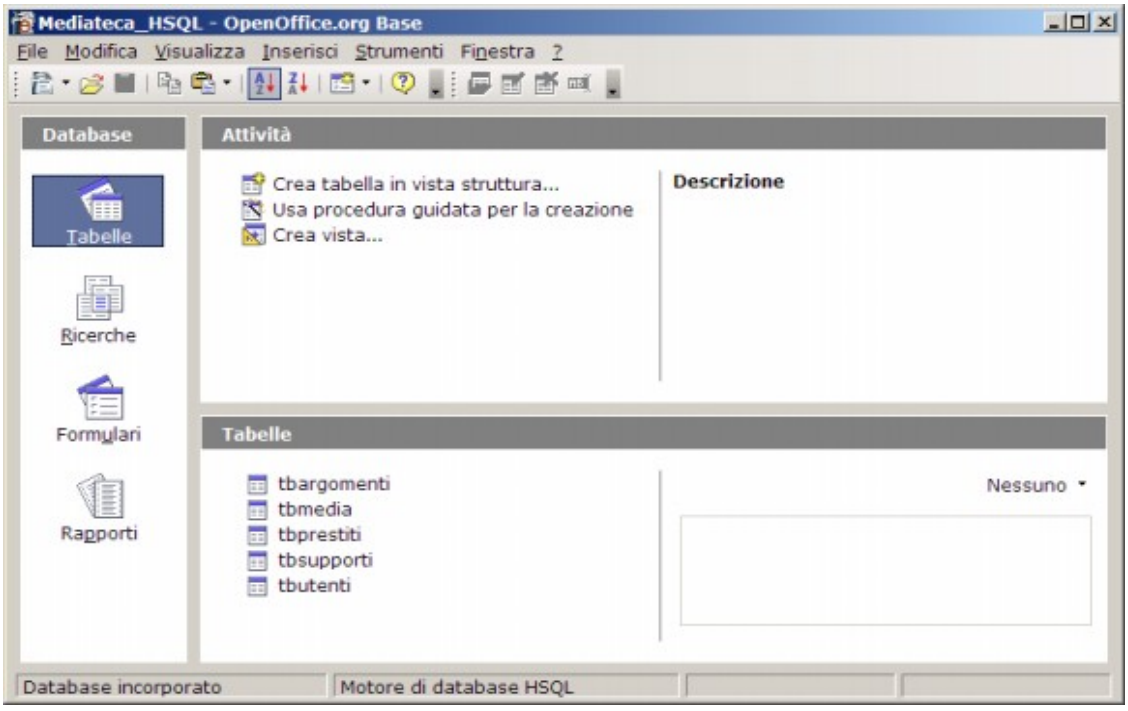


Figura 3.4.4: Finestra principale del Modulo Base

4. Motore di Database HSQL

Dalla versione 2.0, OOo ha deciso di dotarsi di un proprio motore di Db, chiamato **HSQL**. **HSQLDb** (http://hsqldb.org/) è un motore Db Open Source scritto in Java. Questo motore viene attivato automaticamente se, nella procedura di creazione di un Documento *Base*, non si sceglie una sorgente dati esterna. In questo caso tutte le tabelle (ed i dati in esse contenuti) non risiedono "all'esterno", ma *nel* Documento stesso.

Questo approccio limita molto l'utilizzo delle informazioni in modalità multiutente, perchè non è possibile aprire lo stesso Documento da più posti di lavoro senza avere seri problemi. Inoltre uno delle regole principali della corretta gestione dei Database (quella di tenere strettamente separati *Dati* ed *Applicazioni*) viene ovviamente violata. HSQL usa Java, *quindi è necessario un JRE installato e funzionante*.

4.1 Tipi di Campo

4.1.1 Campi di Tipo Stringa

HSQL implementa quasi tutte le tipologie riconducibili a "stringa". Abbiamo ***char*** e ***varchar***, oltre ad un "peculiare" ***varchar_ignorecase*** che dovrebbe trattare le stringhe senza badare al maiuscolo / minuscolo (utile negli ordinamenti ?). Associata alla descrizione ***Memo***, troviamo anche una ***longvarchar***, ma non chiedetemi quanti caratteri al massimo può contenere....

4.1.2 Campi di Tipo Numerico

I campi di tipo **numerico intero**, a seconda dell'intervallo di valori che possono contenere, si dividono in ***TinyInt***, ***SmallInt***, ***Int***, ***BigInt***. Il range dei valori disponibili non è riportato nella Guida di OOo, e non è facile da trovare neanche sul sito di HSQL, quindi non posso aiutarvi. Analogamente, i campi di tipo **numerico decimale** in HSQL possono essere definiti con un numero di decimali fisso (***Decimal***, ***Numeric***) oppure variabile (***Float***, ***Double***, ***Real***). In particolare il tipo *Decimal* è indicato per la manipolazione di valori "valutari", quello che in altri motori di Db viene definito campo "*Money*" o "*Currency*".

TIP

Il valore realmente memorizzato nelle colonne definite ***Float***, ***Double*** o ***Real*** dipende, stranamente, da come viene formattata la colonna stessa in modalità visualizzazione. OOo assume per default un numero di decimali pari a due, e, se non si specifica altrimenti, arrotonda a due cifre decimali qualsiasi valore inserito dall'utente. Questo, ovviamante, può creare grossi problemi.

4.1.3 Campi di Tipo Booleano

Definito in italiano come Si/No equivale al classico ***boolean***. Nella rappresentazione interna, un valore uguale a zero dovrebbe essere considerato "*falso*", diverso da zero "*vero*".

4.1.4 Campi di Tipo Data/Ora

HSQL usa ***Date*** e ***Time***, dal significato piuttosto intuitivo. Mi pare che il range di "Date" sia compreso tra 01-01-100 e 31-12-9999. Se si immette l'anno con sole due cifre, i valori da 01 a 29 vengono considerati successivi al 2000, da 30 a 99 precedenti (esempio 31/12/29 equivale a 31/12/2029 – 31/12/30 a 31/12/1930).

4.1.5 Campi di Tipo Binario

Abbiamo una scelta ampia: ***binary***, ***varbinary***, ***longvarbinary***. Anche in questo caso non sono riuscito a trovare la lunghezza massima del campo.

4.1.6 Campi particolari: Intero ad incremento automatico

Possiamo usare un campo "***Integer***" con la caratteristica "***Valore Automatico***" settata.

4.1.7 Campi particolari: Timestamp

Il ***Timestamp*** di HSQL è definito come ***Data/Ora*** e non viene aggiornato automaticamente dal motore di Db.

4.2 Creazione e Modifica di Tabelle

Possiamo aggiungere o modificare Tabelle con le voci apposite del pannello *Attività* dopo aver scelto *Tabelle* nel pannello Database. La finestra di gestione della struttura delle Tabelle è concepita in modo assai semplice.

Il *Tipo di Campo* può essere scelto da una casella a discesa, e nella parte inferiore è possibile assegnare ulteriori proprietà. Per quanto sia presente un *Esempio di Formato*, non sempre poi viene rispettato nella visualizzazione. In fase di salvataggio, OOo chiede se desiderate aggiungere una chiave primaria: è importante rispondere **No** se avete già definito nella struttura un campo ***Intero ad incremento automatico*** (come ***Mid***). Infatti in questo modo OOo utilizzerà la chiave definita da voi. La finestra di gestione degli indici si richiama con l'apposito pulsante della barra degli strumenti e non comporta difficoltà particolari. HSQL supporta anche le **viste** (***view***): la creazione di una vista è molto simile (se non uguale...) a quella di una *ricerca* (che vedremo in seguito). C'è da chiedersi, visto che l'archiviazione dei dati è interna, a cosa può servire mai una *vista*...

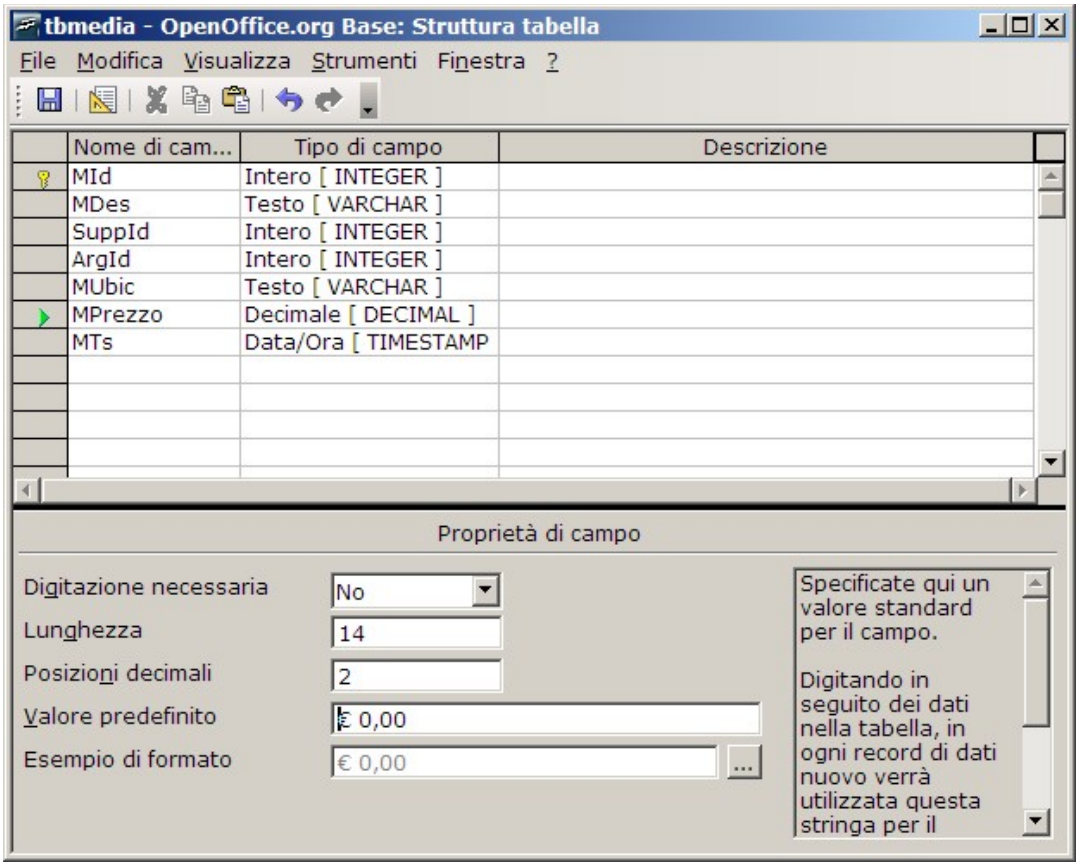


Figura 4.2.1: Modifica della struttura di una Tabella

Da questa finestra è possibile richiamare la funzionalità di manutenzione degli Indici con la voce **Strumenti -> Struttura Indice**.

TIP

Tenete presente che le variazioni agli archivi di HSQL non avvengono in maniera "fisica" ma "logica". Se ad esempio aggiungo 1000 righe ad una Tabella e quindi decido di cancellarle, la dimensione fisica del Db resta la stessa. Questo meccanismo ricorda un po' le Tabelle di Dbase che ogni tanto avevano bisogno del comando *pack*. Per ottenere la stessa cosa in HSQL è necessario scegliere *Strumenti -> SQL* dal Menu del modulo Base ed eseguire il comando **shutdown compact**.

4.3 Integrità referenziale

HSQL supporta l'utilizzo di chiavi esterne, nel modo classico dei Database Server. Per impostare le relazioni è necessario scegliere dal Menu la voce **Strumenti -> Relazioni**. La crezione di una relazione è semplice: basta aggiungere le Tabelle alla finestra, e trascinare il campo della Tabella su cui si desidera *applicare* una chiave esterna sul corrispondente campo

della Tabella che *contiene* la chiave. HSQL supporta i più comuni tipi di relazione: per una trattazione più approfondita, vedere il paragrafo relativo a MySql. Ecco, ad esempio, le relazioni impostate per la Mediateca:

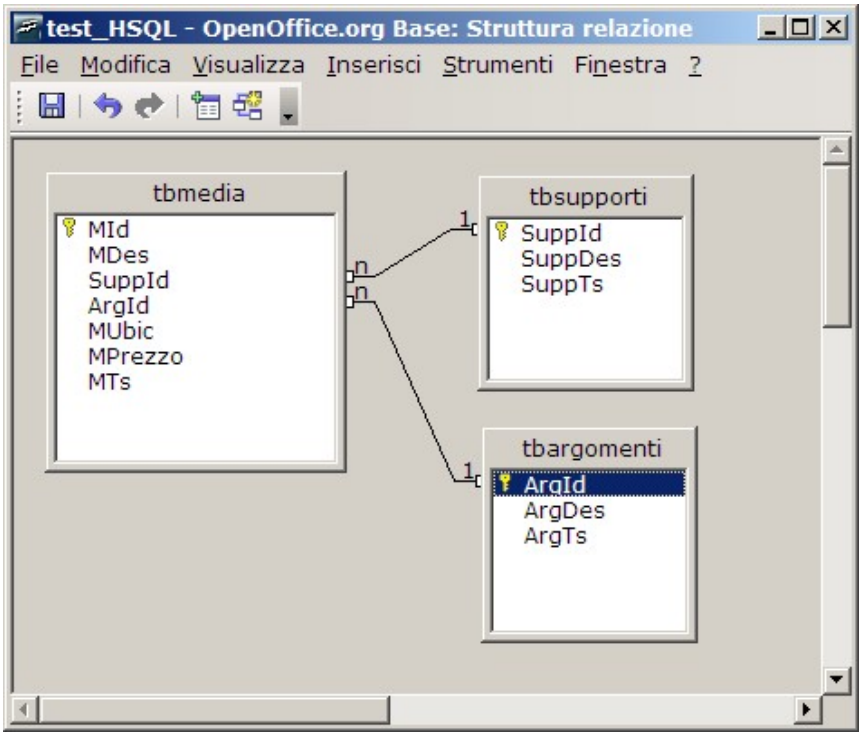


Figura 4.3.1: Relazioni in HSQL per la Mediateca

4.4 Importazione di Archivi esterni

Non credo che esista un sistema semplice per importare archivi esistenti in HSQL. Infatti non disponiamo di un Driver ODBC per HSQL, e non ho trovato nessuna procedura di importazione per altri formati (magari almeno in fixed text, che diamine...). Dopo un po' di prove, la soluzione più semplice è questa:

- creare una Database OOo collegato alla sorgente dati da importare (ad es. MySql)
- creare un Database OOo HSQL
- aprire i due file
- trascinare le tabelle da trasferire dal primo Database al secondo, oppure usare copia/incolla

A questo punto si apre una finestra di auto composizione che aiuta nel processo. Attenzione: non sempre è possibile ottenere risultati accettabili; per esempio non c'è verso di importare Tabelle di Access, mente MySql funziona quasi sempre.

4.5 I Limiti di HSQL

La versione usata in OOo Base, l'abbiamo detto, è "embedded", cioè non si appoggia ad un motore esterno, e questo è un grosso problema. Vediamo perché.

- **Formato di File non manipolabile**. Nonostante il formato di OOo Base (.odb) sia uno standard OASIS, i dati vengono "inclusi" nel file XML in forma binaria e non in XML. Questo comporta che non siano accessibili dall'esterno in modo semplice.
- **Assenza della multiutenza.** HSQL in versione OOo non è multiutente, quindi non è permesso l'accesso contemporaneo ai dati da parte di più utenti.
- **Assenza di qualunque Driver**. I dati di un file *.odb* HSQL possono essere letti solo direttamente da OOo. Non esiste infatti alcun Driver per l'accesso esterno. Quindi niente ODBC, oppure JDBC oppure OLEDB.
- **Difficoltà di importazione da altre fonti**. Importare Database da altri formati non è semplice. Non esiste una procedura guidata, i formati di Data e Ora sono difficilmente manipolabili, ed anche un semplice file .csv crea problemi ad un utente non tecnico.
- **Dipendenza da JAVA**. HSQL non funziona se non è installata e funzionante una macchina virtuale JAVA, che, lo ricordo, non è un prodotto Open Source.
- **Prestazioni**. Il motore è lento, già con relativamente pochi record mostra la corda. Con Archivi di un certo "spessore" è inusabile.

Basta questo per definire, almeno per ora, HSQL un prodotto non adatto in ambito professionale.

5. Database Server MySql

MySql (www.mysql.com) è probabilmente il più usato Database Server Open Source disponibile. La sua caratteristica principale è sempre stata la velocità di risposta, ed in nome dell'efficienza gli sviluppatori hanno probabilmente trascurato alcuni aspetti che dovrebbero essere presenti in un prodotto che aspira a fare concorrenza a nomi blasonati come *Microsoft Sql Server* o *Oracle*. La versione stabile attualmente disponibile è la 5.0.XX che introduce alcune novità largamente richieste dall'utenza, come le *stored procedure* ed i *trigger*. Mancheranno però ancora delle caratteristiche che molti ritengono essenziali, ma che gli sviluppatori di MySql valutano non compatibili con un prodotto snello ed efficiente.

Nei paragrafi che seguono ci occuperemo di alcuni aspetti interessanti del Db, utili anche in contesti non strettamente collegati ad OpenOffice.

5.1 Installazione Windows

5.1.1 Il Server

Avere in pochi minuti MySql attivo e funzionante in Windows è molto semplice. I file da scaricare dal sito www.mysql.com sono:

mysql-essential-5.0.XX-win32	il **server**, per circa 17 Mb – la versione *essential* è più che sufficiente ai nostri scopi
myODBC-3.51.XX-win	**Driver ODBC**, per circa 4 Mb (versione attuale 3.51.11)
mysql-connector-java-3.1.XX	**Driver JDBC** in formato Zip per circa 8 Mb, se si desidera connettersi al Db con Java
mysql-administrator-1.1.X-win	**MySql Administrator**, per circa 5 Mb, un tool che permette la creazione e la manutenzione dei Database e del Server MySql (versione attuale 1.1.4)
mysql-query-browser-1.1.XX-win	**MySql Query browser**, per circa 5 Mb, permette la modifica dei Dati di Database Sql ed il test delle Query

In realtà il *Query Browser* non è strettamente necessario, perché la modifica dei dati avviene di solito con programmi *client* (nel nostro caso OOo), ma torna utile in molti casi. Assolutamente indispensabile è invece *Administrator*, a meno che non vogliate impazzire con le utility a riga di comando. Tutte le operazioni di installazione vanno fatte da un utente con privilegi di amministratore. Il primo file (il **Server**) è un installer windows che è necessario lanciare nel modo tradizionale. Si avvia un Wizard e, tra le prime opzioni di installazione, la configurazione "*typical*" può andar bene nella maggior parte dei casi. La schermata successiva chiede di creare un account di registrazione su *MySql.com* ma si può evitare scegliendo "*skip*

sign-up". Alla fine il programma chiede se si desidera configurare il server appena installato, e questa è senz'altro una buona idea.

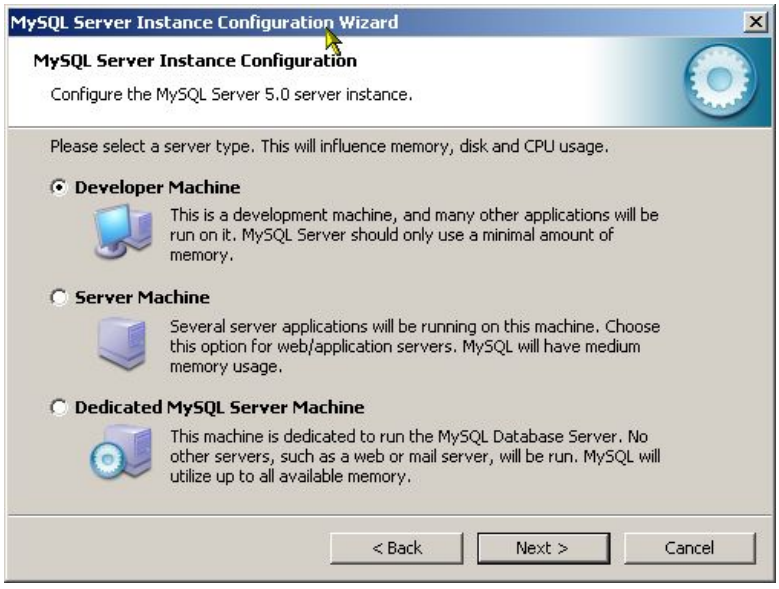


Figura 5.1.1: Configurazione dell'istanza di MySql Server

La prima scelta riguarda il tipo di configurazione (*standard* o *dettagliata*): si può selezionare ***dettagliata***, anche per farsi un'idea delle varie opzioni disponibili. Poi si passa al tipo di utilizzo che intendiamo fare di MySql (vedi figura). Di solito va bene anche "**Developer Machine**". Il passo successivo riguarda i "motori" interni da selezionare per l'archiviazione dei dati. In questo caso è opportuno selezionare "**Multifunctional Database**" in modo da disporre sia di *MyIsam* che di *InnoDb* (del cui utilizzo parleremo in seguito). Poi dobbiamo scegliere il tipo di server in funzione delle connessioni simultanee: se non ci sono esigenze particolari, va bene "**DSS**" che è la prima opzione (vedi figura).

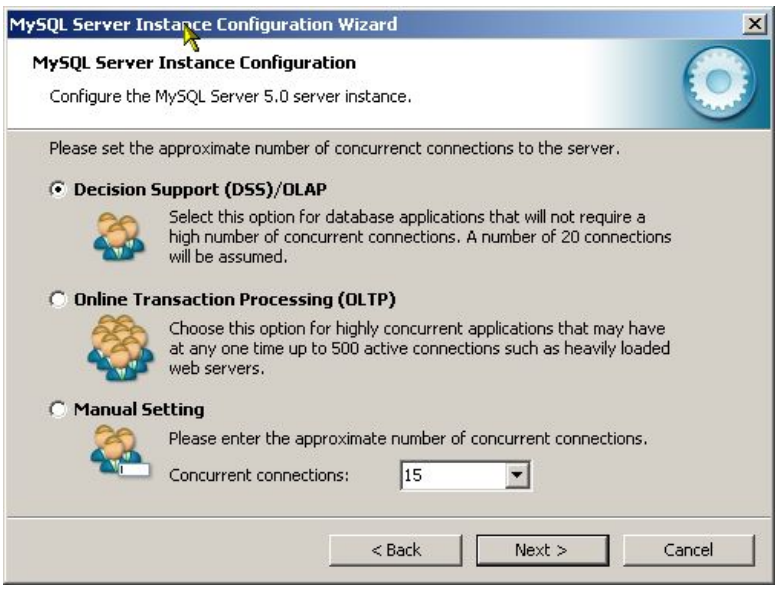


Figura 5.1.2: Numero di client previsti per MySql

Quindi si passa alla configurazione di Rete. Se intendiamo usare il Server sulla rete (come di solito accade) dobbiamo **abilitare il TCP/IP** e selezionare la **porta di ascolto** (3306 è quella standard). Dopo aver confermato il **Set di Caratteri** (quello proposto va bene), si sceglie di **avviare il Server come servizio** e (se volete usare le utility a linea di comando) di includere la **cartella Bin** nel path di Windows. Quindi è necessario impostare la **password** per l'Amministratore (che in MySql, come in Linux, si chiama **root**), scegliere se *root* può collegarsi anche dalle macchine in rete (dipende da voi, di solito si preferisce che *root* possa eseguire il logon solo sulla macchina che ospita il server), e se abilitare l'**accesso anonimo** (vivamente sconsigliato).

Figura 5.1.3: Impostazione della sicurezza in MySql

A questo punto la configurazione termina, e siete pronti ad usare il Server.

Tecnica

MySql può essere installato in Windows 2000 e XP come "***servizio***" ad esecuzione automatica. Se abbiamo scelto invece la partenza "*manuale*", per avviare il Server basta richiamare il pannello di controllo e nella sezione "*strumenti di amministrazione*", e scegliere la voce "*servizi*". Avremo una lista dei servizi installati sul PC, quindi click col tasto destro sul servizio *MySql* e dal Menu contestuale selezionare la voce "*Avvia*". Viceversa, se desideriamo che MySql non si avvii più in automatico, doppio click sul servizio *MySql* e selezionare nella casella a discesa "*Tipo di Avvio*" la voce "*Manuale*".

Il resto dei programmi citati può anche non essere installato sulla macchina che ospita il server in quanto si tratta essenzialmente di strumenti *client*. Questo significa che possono (ed a volte devono) essere caricati sui PC che si occuperanno della manutenzione del Server e delle Basi di Dati, e che eseguiranno l'accesso ai Dati stessi.

TIP

La configurazione del Server può essere modificata in qualsiasi momento con la voce *Start -> Programmi -> MySQL -> MySQL Server 5.0 -> MySQL Server Instance Config Wizard*

5.1.2 Il Driver MyODBC

Il file, l'abbiamo detto è ***myODBC-3.51.XX.msi*** (dove XX sta per il numero di versione corrente). Basta eseguire l'installazione ed avremo il nostro driver pronto all'uso. Per verificare che tutto sia andato bene, aprendo il *Pannello di Controllo -> Strumenti di Amministrazione -> Origine dati (ODBC) -> Driver* dovrebbe apparire :

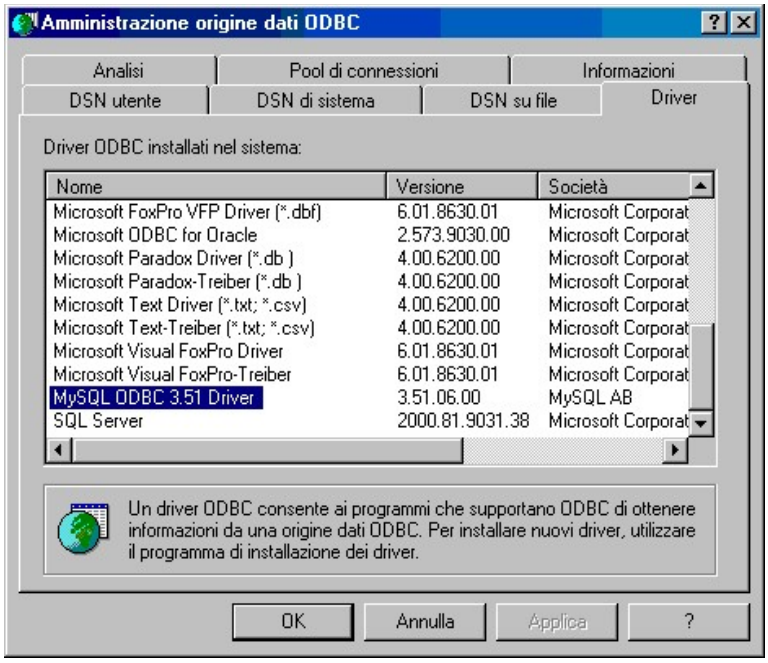


Figura 5.1.4 : Driver ODBC

5.1.3 Configurazione del DSN

Bisogna scegliere, innanzi tutto, se si desidera creare un DSN Utente (quindi valido solo per l'utente corrente) oppure di sistema (per tutti gli utenti). Quindi si preme il pulsante Aggiungi, si seleziona il driver MySql e quindi il pulsante Fine. A questo punto è possibile configurare la connessione, come in figura.

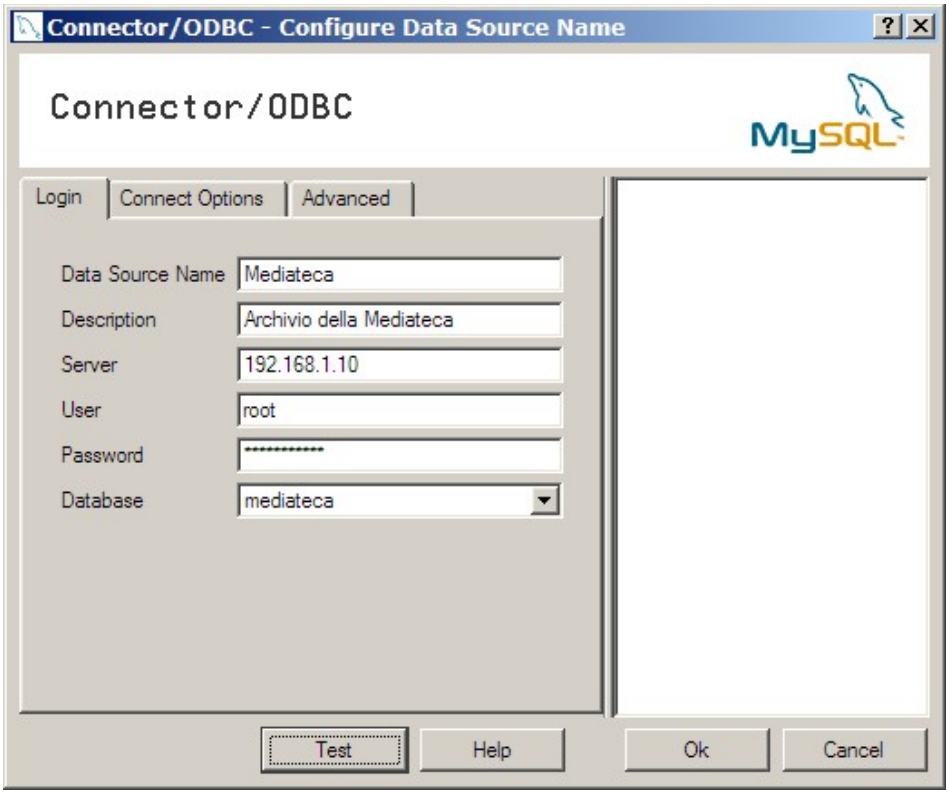


Figura 5.1.5: Configurazione del DSN per MySql

Si deve assegnare un *nome* al DSN, indicare *l'indirizzo IP* oppure il *nome* del Server, immettere le *credenziali* (nome utente e password) e, volendo, indicare un *Database* per la gestione. Se la connessione avviene sulla macchina che ospita il server, si può indicare *localhost* al posto dell'indirizzo IP. Il pulsante **Test** permette di provare la connessione. Nelle Tab *Options* ed *Advanced* è possibile impostare alcuni altri parametri, ma di solito non è necessario modificare quelli di default. Se volete approfondire, fate riferimento alla documentazione di MySQL.

5.1.4 Driver JDBC

E' necessario scaricare, dal sito di MySql, il file **mysql-connector-java-3.1.11.zip**. Una volta scompattato, copiare il file **mysql-connector-java-3.1.11-bin.jar** in una cartella a piacere (io uso *c:\programmi\JDBC*).

5.1.5 I programmi Client

Per la gestione del Server e delle Basi di Dati abbiamo già detto che sono disponibili due programmi: **Administrator** e **Query Browser**. L'installazione non comporta alcuna difficoltà e le rispettive voci compariranno nel Menu dei Programmi. Administrator installa anche una piccola utility che si chiama **System Tray Monitor** che serve anche ad avviare ed arrestare il Servizio, e che può tornare utile.

5.2 Installazione Linux

5.2.1 Il Server

Quasi tutte le moderne distribuzioni prevedono la possibilità di installare MySQL nelle sue varie versioni. Utilizzando i tool presenti nelle distribuzioni è dunque semplice avere immediatamente il server funzionante. Se proprio desiderate l'ultima versione, è disponibile anche in formato RPM sul sito di MySql.com (attenzione alle dipendenze!). In questo caso, la procedura di installazione e configurazione è ben spiegata nella documentazione ufficiale, quindi non mi dilungherò oltre.

TIP

Di solito le versioni Linux, appena installate, hanno una gestione della sicurezza particolarmente debole. Infatti la password dell'amministratore di MySql è vuota, anche se è possibile collegarsi al Database solo dalla macchina che lo ospita (*localhost*). La prima cosa da fare è dunque configurare la sicurezza.

5.2.2 Il Driver MyODBC

In molte distribuzioni il Driver è già incluso nell'elenco dei pacchetti, ma vi consiglio di scaricare l'ultima versione in formato RPM da MySql.com, nel nostro caso **MyODBC-3.51.11-2.i586**. Per funzionare il Driver ha bisogno del pacchetto ***UnixODBC***, disponibile su *Sourceforge*. Possiamo scaricare, ad esempio, **unixODBC-2.2.10-1.i386** (RPM delle librerie) e **unixODBC-gui-qt-2.2.10-1.i386** (interfaccia grafica di configurazione). La giusta sequenza di installazione prevede *unixODBC*, *unixODBC-qt* e quindi *MyOdbc*. Dopo l'installazione, il programma da lanciare è ***/usr/bin/ODBCConfig***. Sarebbe anche possibile installare il driver modificando con un editor il file *odbc.ini*, ma questo direi che esula dallo scopo di questa documentazione.

In ogni caso, il programma si presenta molto simile alla sua controparte Windows. Anche in questo caso è possibile aggiungere un *DSN* selezionando il tipo di Driver. La configurazione successiva della connessione è esattamente uguale a quella vista per i Sistema Operativo di Microsoft.

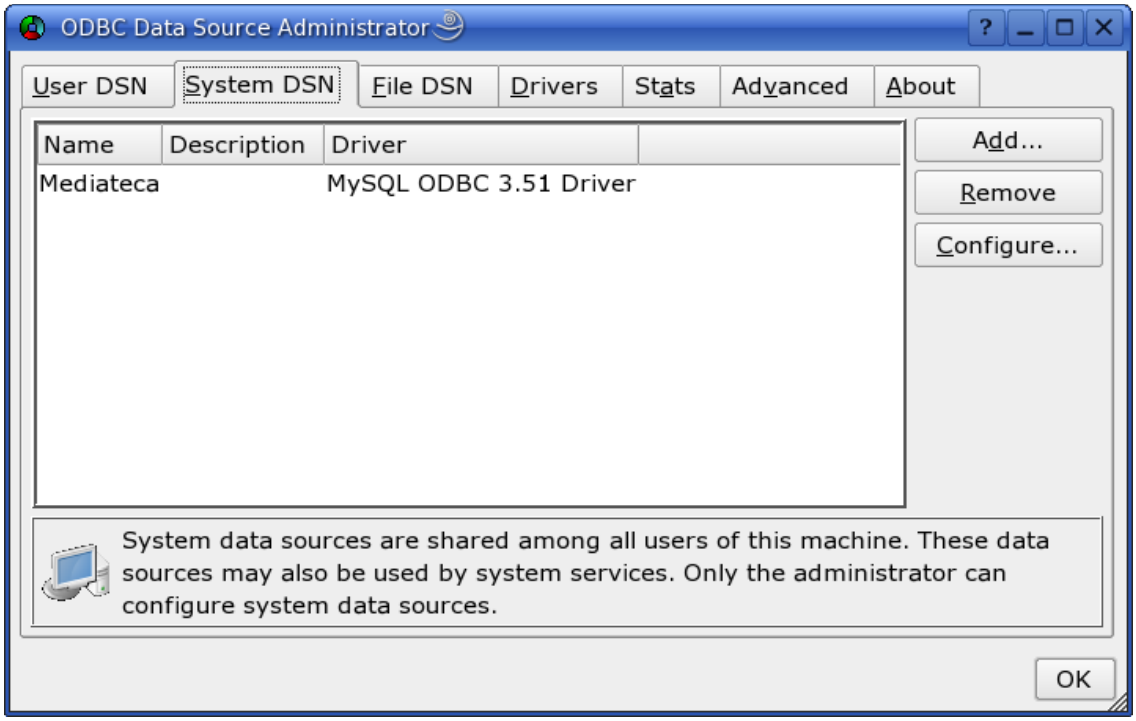


Figura 5.2.1: DSN in Linux (SUSE 9.3)

5.2.3 Il Driver JDBC

E' necessario scaricare, dal sito di MySql, il file **mysql-connector-java-3.1.11.zip**. Una volta scompattato, copiare il file **mysql-connector-java-3.1.11-bin.jar** in una cartella a piacere (io uso /opt/JBDC/).

5.2.4 I programmi Client

Dal sito MySql.com è possibile scaricare in formato RPM sia ***Administrator*** che ***Query Browser***. L'installazione, per le distribuzioni che prevedono un RPM manager è semplice, salvo problemi con le dipendenze. In alternativa, è anche possibile ricompilare dai sorgenti.

5.3 I Tipi di Dati

Ogni Server di Database possiede un lungo elenco di tipologie di Dati gestibili. In realtà i tipi più comuni (numeri, testo e date) sono molto simili in tutti i "motori" SQL. Credo però sia opportuno indicare con precisione, per ogni Server, la denominazione esatta della tipologia e le caratteristiche della informazione che può essere archiviata.

5.3.1 Campi di Tipo Stringa

MySql implementa i classici ***char*** e ***varchar***; una "*n*" davanti al tipo di campo ("***nchar***", "***nvarchar***") significa che il campo stesso usa il set di caratteri internazionali predefinito (opzione comunque attiva di default).

Tipo di Dato	*Lunghezza*	*Definizione*
char / **nchar**	Da 0 a 255	char(X) dove X è il numero di caratteri
varchar / **nvarchar**	Da 0 a 255 Da 0 a 65.535	varchar(X) dalla versione 5.0.3

5.3.2 Campi di Tipo Numerico

Qui troviamo qualche utile variante allo standard. I campi di tipo **numerico intero**, a seconda dell'intervallo di valori che possono contenere, si dividono in ***TinyInt***, ***SmallInt***, ***MediumInt***, ***Int***, ***BigInt***. Analogamente, i campi di tipo **numerico decimale** si possono definire ***Float***, ***Double***, ***Decimal***. In particolare il tipo *Decimal* è indicato per la manipolazione di valori "valutari", quello che in altri motori di Db viene definito campo ***Money*** o ***Currency***. L'attributo **unsigned** elimina il segno e quindi permette la gestione di soli numeri positivi. **zerofill** invece riempie a sinistra il numero con zeri fino alla dimensione massima definita.

Tipo di Dato	*Intervallo di valori*	*Note*
bit	Da 0 a 255	Introdotto dalla versione 5.0.3 – sinonimo di tinyint[1]
tinyint	Da -128 a 127 signed Da 0 a 255 unsigned	Occupa 1 byte di spazio
smallint	Da -32.768 a 32.767 signed Da 0 65.535 unsigned	Occupa 2 byte di spazio
mediumint	Da -8.388.608 a 8.388.607 signed Da 0 a 16.777.215 unsigned	Occupa 3 byte di spazio
int	Da -2.147.483.648 a 2.147.483.648 Da 0 a 4.294.967.295 unsigned	Occupa 4 byte di spazio
bigint	Da -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 signed da 0 a 18.446.744.073.709.551.615 uns	Occupa 8 byte di spazio
float(M,D)	Virgola mobile singola precisione	M è il numero massimo di cifre da visualizzare, D il numero di cifre decimali
double(M,D)	Virgola mobile doppia precisione	
decimal(M,D)	Numero a virgola fissa	Vedi riquadro tecnica

Tecnica

La definizione (e la gestione) del tipo **Decimal** sono ***cambiate*** in *MySql 5.X*. Nella vecchia versione il valore era memorizzato fisicamente come una stringa, mentre ora si utilizza un formato binario. Questo potrebbe causare problemi con le applicazioni che accedono ai dati, come, ad esempio, OpenOffice.

5.3.3 Campi di Tipo Booleano

Definito come **boolean**, è l'equivalente di ***TinyInt[1]*** (quindi un valore numerico intero). Come tipo di dati è presente dalla versione 4.1.0 di MySql. Un valore uguale a zero viene considerato "*falso*", diverso da zero "*vero*".

5.3.4 Campi di Tipo Data/Ora

MySql usa ***Date***, ***DateTime***, ***Time***, ***Year***, dal significato piuttosto intuitivo. Notate che il range di "Date" va dal 01-01-1000 al 31-12-9999. Siccome quasi sempre le date vengono memorizzate nel formato AAAAMMGG e visualizzate invece nel formato americano MM-GG-AAAA, bisogna fare molta attenzione ai formati di input che si assegnano.

5.3.5 Campi di Tipo Binario / Text

In questo caso MySql usa una variante del tipo *text* chiamata ***blob*** (binay long object) con alcune tipologie sempre collegate alla dimensione massima archiviabile (***tinyblob***, ***mediumblob***, ***longblob***). Per l'archiviazione di dati di tipo *testo* abbiamo i corrispondenti ***tinytext***, ***text***, ***mediumtext***, ***longtext***. Le dimensioni massime archiviabili sono elencate al capitolo 11.5 della guida di MySQL.

5.3.6 Campi particolari: Intero ad incremento automatico

In MySql possiamo usare un campo ***Int*** con la caratteristica ***auto_increment*** settata.

TIP

Si noti che in MySql esiste anche un tipo ***SERIAL*** equivalente a ***BIGINT*** **UNSIGNED NOT NULL AUTO_INCREMENT**. Siccome in altri motori di Db il tipo *SERIAL* viene utilizzato per le chiavi primarie, se non avete particolari esigenze, per lo stesso scopo può essere usato il tipo ***INT*** **UNSIGNED NOT NULL AUTO_INCREMENT**, che risparmia 4 byte ed è più veloce nelle elaborazioni.

5.3.7 Campi particolari : Timestamp

Il ***Timestamp*** di MySql è di tipo "classico", cioè viene aggiornato automaticamente dal motore di Db. In una Tabella è possibile definire più campi Timestamp: quelli successivi al primo sono a disposizione dell'utente.

5.4 MySql Administrator

Ogni Database che si rispetti avrebbe bisogno di uno strumento di amministrazione semplice e potente, adatto anche a chi non vuole usare la pur potente riga di comando. Dall'*estate del 2004* è disponibile una nuova interfaccia di amministrazione per i server MySql, chiamata appunto **MySql Administrator**, scaricabile, per tutti i sistemi operativi supportati, da *www.mysql.com*. Con questo Tool è possibile gestire tutti gli aspetti della configurazione del nostro Db Server; è inoltre possibile modificare, aggiungere, cancellare Tabelle, Campi ed Indici, il tutto con una semplicità encomiabile. Dopo aver avviato il programma è necessario scegliere quale Server SQL vogliamo gestire, fornendo anche le eventuali credenziali di autenticazione, nella maschera in figura:

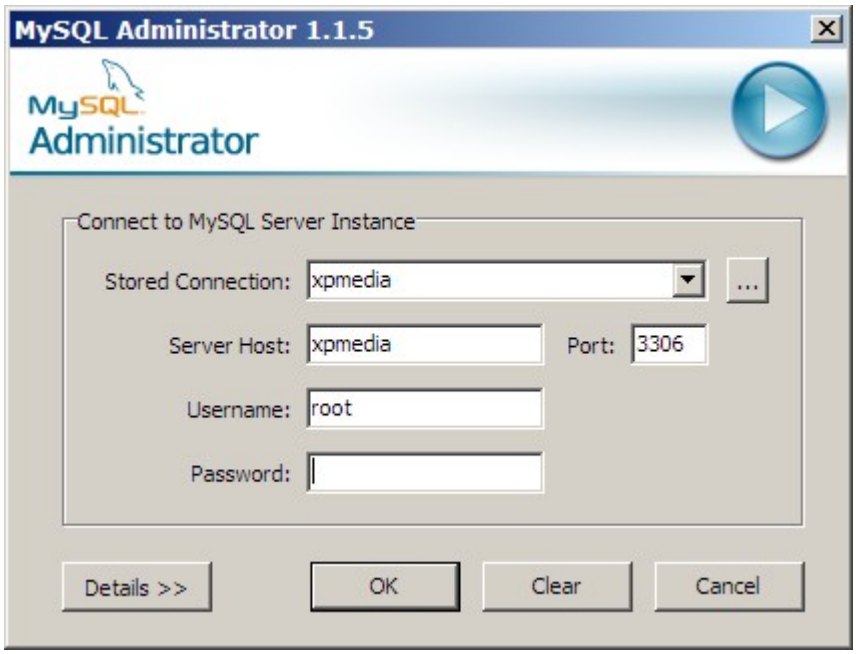


Figura 5.4.1: Connessione a un Db con Administrator

Vi ricordo che se MySql gira sulla stessa macchina che state usando, è sufficiente inserire come Hostname "*localhost*", in caso contrario è necessario indicare l'indirizzo Ip del server. A questo punto vi troverete con la finestra principale finestra attiva.

Non è ovviamente questa la sede per illustrare in dettaglio tutte le funzionalità di *MySql Administrator,* mi limiterò quindi a segnalarvi gli aspetti più utili. Selezionando la voce

desiderata nell'elenco a sinistra, il programma mostra sulla destra i pannelli di gestione della voce stessa.

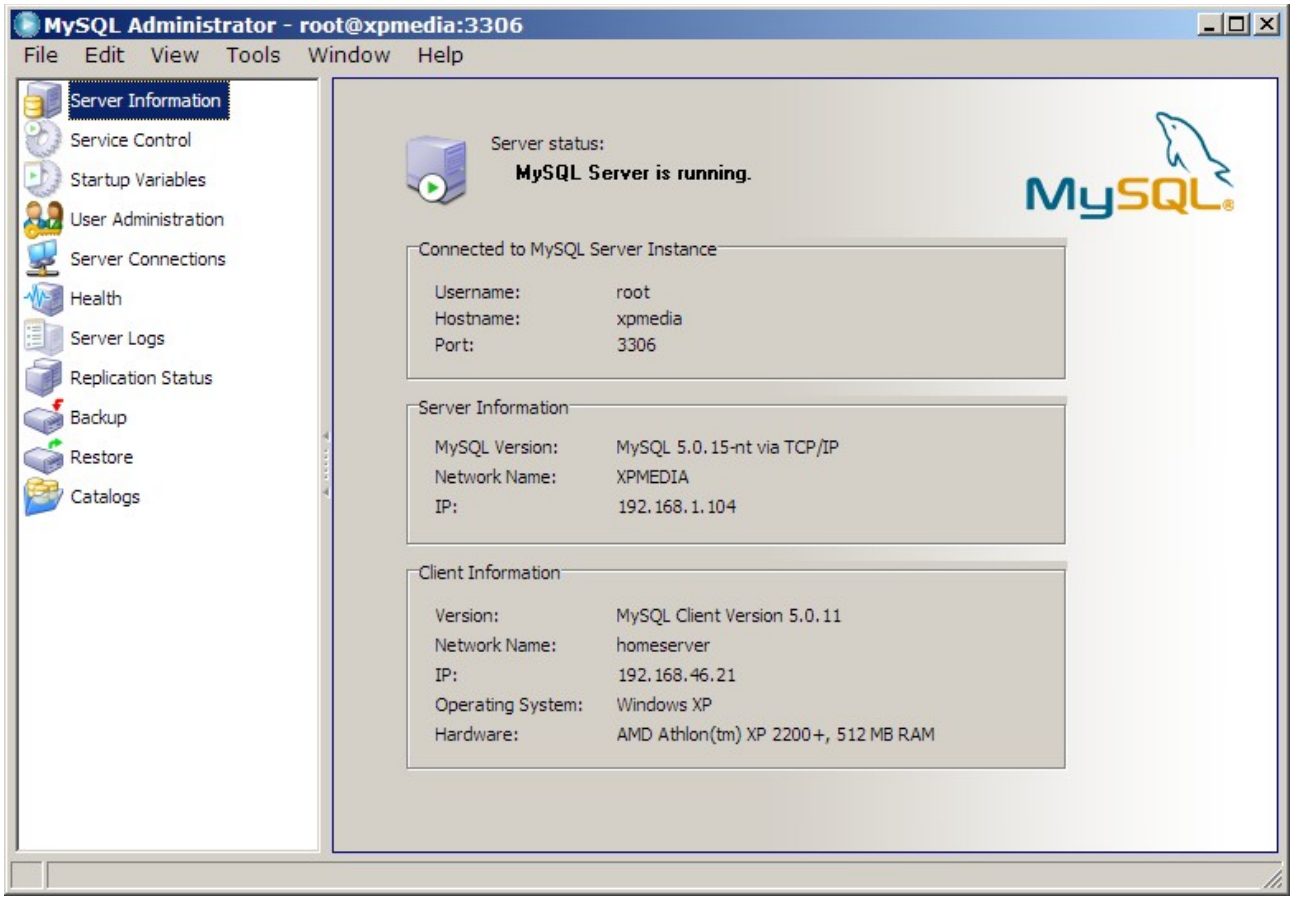


Figura 5.4.2: Finestra principale di MySql Administrator

La voce **Service Control** permette di stabilire le modalità di funzionamento del Server; può infatti essere utile decidere che il servizio sia eseguito in automatico dal Sistema Operativo all'avvio, ed in effetti questo è il caso più comune. Tra le altre opzioni, vi ricordo che *Support for InnoDb* (cioè la possibilità di usare le Tabelle di Tipo InnoDb) è attivo di default dalla Versione 4.0, quindi va eventualmente abilitato solo per le versioni precedenti; che *Support for BDB* serve solo se volete usare quel tipo di Tabelle (vi ricordo che viene ancora dichiarato in Beta Test); che *Named Pipes* permette l'accesso al Db sul Localhost (cioè in locale) attraverso questa modalità operativa al posto del Tcp/Ip (l'uso delle *Named Pipes* è comunque, a mio parere, una opzione inutile, anzi direi dannosa).

Startup Variables permette il *fine tuning* (configurazione dettagliata) dell'ambiente MySql, fornendo un'interfaccia grafica di accesso a tutti i parametri di configurazione specificati nel file *my.ini*. Per quanto una trattazione più approfondita sarebbe interessante, non è questa la sede adatta. Vi consiglio però di fare modifiche solo se ben consapevoli di quello che volete ottenere.

User Administration serve a gestire gli account Utente ed i privilegi di accesso al Server ed ai Database. Di questo ci occuperemo in dettaglio in uno dei prossimi paragrafi.

Server Connections si occupa di monitorare in dettaglio gli accessi di basso livello al server: è utile soprattutto nelle fasi di controllo del carico e delle connessioni attive.

Health propone un quadro molto dettagliato dello "stato di salute" del nostro Server attraverso grafici aggiornati in tempo reale su vari aspetti del funzionamento di MySql. Molto utile ed anche "scenografico".

Server Logs permette l'accesso a tutti i file di Log del Server.

Replication Status fornisce informazioni sulle eventuali "repliche" dei dati presenti nel Server; per gli scopi di questo documento è un aspetto che davvero non ci interessa.

Backup e **Restore** forniscono un strumento semplice per effettuare copie di sicurezza dei nostri Db, quindi ne parlerò in dettaglio in uno dei prossimi paragrafi.

Catalogs permette la gestione completa e semplice della struttura dei nostri Database; in pratica da questa opzione si può modificare un Db in ogni suo aspetto, ed è per questo che ne parleremo in modo esteso nel prossimo paragrafo.

5.4.1 Parametri di avvio del Server

Dopo l'installazione, è possibile stabilire una volta per tutte alcuni parametri di avvio del Server che vengono salvati in Windows nel File *My.ini* nella Dir di MySql.

TIP

I parametri di avvio sono modificabili solo se MySql Administrator viene eseguito sulla macchina Server, e solo se l'utente ha diritti di Amministratore sul Computer (quindi root per Linux). Tra le opzioni della Tab *General Parameters* di *Startup Variables* figura anche un *Disable Networking* che in pratica limita il funzionamento del Server alla macchina locale esclusivamente attraverso le *Named Pipes* (in Windows). A meno che non ci siano stringenti esigenze di sicurezza, non disabilitate ovviamente mai l'accesso tramite Tcp/Ip.

5.4.2 Gestione del Database

Se scegliamo **Catalogs**, ci appare in basso l'elenco degli ***Schemi*** cioè dei *Database* presenti nel Server. Selezionando con un click uno degli schemi (ad esempio *Mediateca*) ci ritroveremo nella parte destra, nella Tab **Schema Tables**, l'elenco delle Tabelle contenute nel Database, insieme ad una serie di utili informazioni sulle stesse (vedi figura). Da questa finestra è possibile inserire, modificare e cancellare i campi e le tabelle del Db. La seconda Tab, **Schema Indices**, contiene un elenco di tutti gli indici assegnati a tutte le tabelle. Le Tab **Views** e **Stored Procedures** sono specifiche della versione 5.X di MySql e saranno esaminate in seguito. Se si seleziona una Tabella nello *Schema Tables*, un click sul pulsante in basso ***Details*** ci fornisce informazioni più approfondite sulla Tabella stessa. Il pulsante ***Maintenance***

consente invece di ottimizzare le Tabelle e di recuperare i dati in caso di problemi. Tra le tre opzioni disponibili per la manutenzione, la più interessante (ma anche quella che richiede più tempo di elaborazione) è senza dubbio ***Optimize*** che in un colpo solo ripara (se necessario) la Tabella, ricostruisce gli indici ed aggiorna le statistiche di accesso. L'operazione si può eseguire anche su più tabelle, selezionando le tabelle stesse con le *SHIFT* (per quelle contigue) o con il *CONTROL* per quelle non contigue.

TIP

Nella Figura compare già il nostro Database di esempio "*mediateca*", ma nel seguito vedremo come costruirlo passo dopo passo. MySql indica col nome di "*Schema*" quello che in altri motori di Db viene indicato come "*Database*"; così la tradizionale "struttura del Database" cioè l'insieme delle definizioni delle Tabelle, dei Campi e degli Indici diventa "*Struttura dello Schema*"

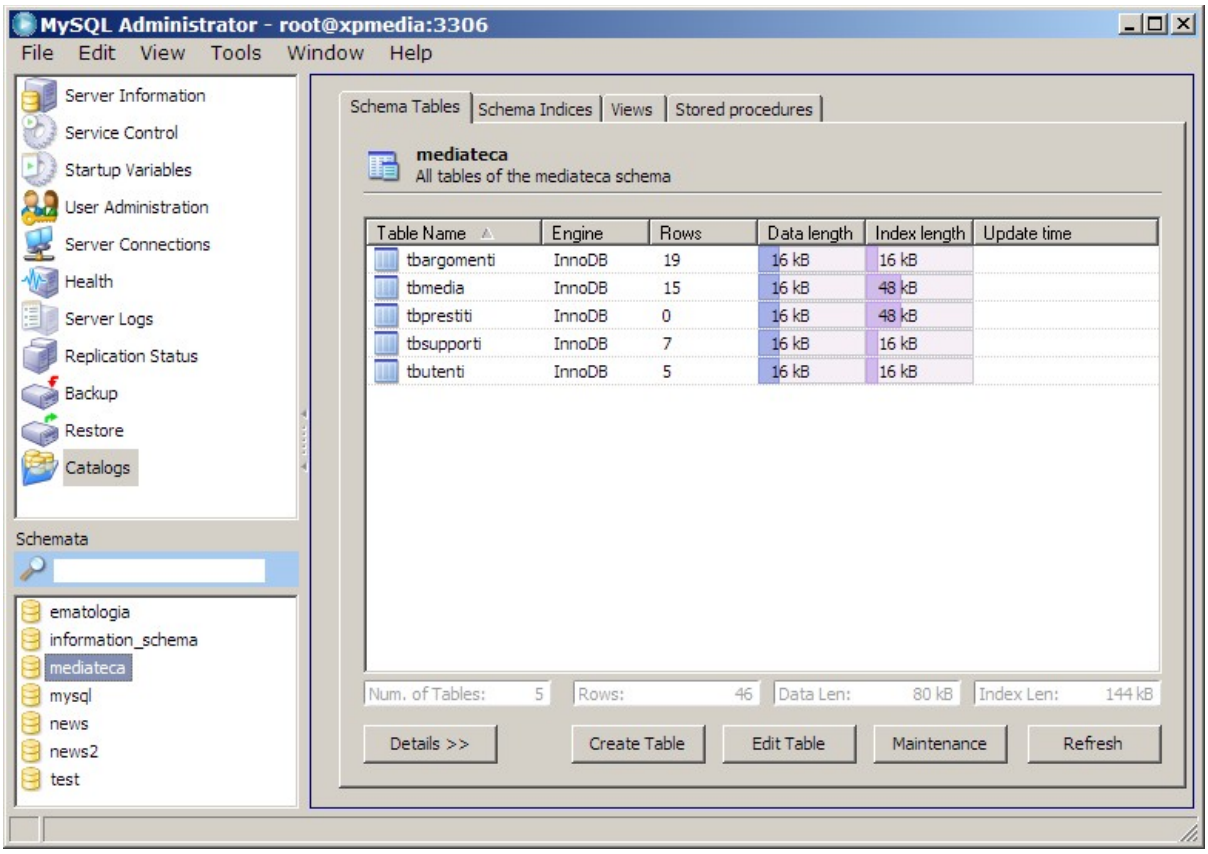


Figura 5.4.3: La gestione dei Catalog

5.5 Gestione delle Tabelle

MySql, a differenza di altri software simili, pur avendo un formato "nativo", può "appoggiarsi" ad altri prodotti Open Source per quanto riguarda il formato di archiviazione delle informazioni. Il formato "originario" del motore si chiama **MyIsam**, ed è in concreto una implementazione piuttosto efficiente del classico *B-Tree*. Nelle versioni più recenti MySql può usare anche Tabelle di tipo **InnoDb** e **BDB** (Berkley Db). In pratica però solo il supporto ad *InnoDb* è dichiarato "stabile", mentre *BDB* è in Beta. Parleremo perciò solo di *MyIsam* e *InnoDb*.

MyIsam, come abbiamo detto, è il formato "nativo" di MySql. Si tratta di Tabelle archiviate in singoli file con estensione ***.MYD*** per i dati e ***.MYI*** per gli indici. Si tratta di una soluzione affidabile e veloce, semplice da gestire anche per quanto riguarda i Backup. Purtroppo questo tipo di tabelle non supporta alcune caratteristiche che sono comuni nei moderni Database, e che possono rivelarsi utili in applicazioni di media complessità. In particolare *MyIsam non prevede la gestione dell'integrità referenziale* (come vedremo tra poco).

Il tipo **InnoDb** è, come abbiamo detto, una "aggiunta" al MySql, nel senso che è un motore di Db di terze parti che è stato inglobato in MySql. Il motore è di concezione più moderna rispetto a *MyIsam*, ed usa, invece dei singoli file, un unico "*tablespace*", più alcuni file di log. Le caratteristiche migliorative di *InnoDb* rispetto a *MyIsam* sono, in breve :

- supporto per le **transazioni**, cioè la possibilità di tornare indietro ad uno stato precedente se l'aggiornamento del Db non va a buon fine;
- **Row Level Locking**, cioè blocco del Record sulla singola riga, utile in ambienti multiutente;
- supporto per le **Foreign Keys**, (cioè le chiavi esterne) che è un primo passo verso l'integrità referenziale;
- **velocità di risposta**, soprattutto in ambiente Windows.

L'uso di un tipo piuttosto che l'altro dipende, quindi, dal tipo di applicazione e di dati che bisogna gestire. Per Db semplici, magari in Linux, va benissimo anche *MyIsam*. Per applicazioni complesse, ed in ambiente Windows, va meglio *InnoDb*. Comunque il passaggio da un tipo all'altro è sempre possibile, e nello stesso Db possono convivere Tabelle di tipi diversi.

Tecnica

Il supporto alle Tabelle di tipo *InnoDb* è incluso nella installazione standard di MySql dalla versione 4.0 in poi. Se non viene utilizzato, è comunque escludibile modificando il file di configurazione con l'opzione *skip-innodb*. Nelle versioni precedenti del motore di Db è necessario, invece, che sia abilitato seguendo le istruzioni dettagliate del manuale dell'applicazione.

5.5.1 Modifica della struttura delle Tabelle

Il primo passo nella gestione di un Db è la creazione dello *Schema*. Bisogna posizionarsi nel pannello dei *Catalogs* in basso a sinistra e quindi, dal menu contestuale del tasto destro del mouse selezionare ***Create New Schema*** e scegliere il nome da dare al "nascituro". Allo stesso modo, per creare una nuova *Tabella*, basta selezionare lo *Schema* e scegliere il pulsante ***Create Table*** in basso.

Allo stesso modo si può modificare la struttura di una Tabella esistente con ***Edit Table***. La finestra che si apre è divisa in tre Tab, nella parte alta. ***Columns and Indices*** si occupa dei campi e degli indici associati; ***Table Options*** permette di stabilire alcuni parametri riguardanti la Tabella; ***Advanced Option*** non è interessante per noi in questo momento.

La Tab ***Columns and Indices*** visualizza nella parte centrale l'elenco e le caratteristiche dei campi: un doppio click sulla parte che ci interessa permette la modifica del parametro.

TIP

Notate che nella colonna *Datatype* non c'è una casella a discesa per la selezione del tipo ma il parametro va specificato per esteso (scrivendo fisicamente ad es. INTEGER). In verità il programma usa una specie di completamento automatico, e chiude anche le parentesi: all'inizio ci si sente disorientati, ma dopo un po' sembra una buona idea.

Nella parte bassa troviamo una Tab (***Indices***) per la gestione degli indici associati alla Tabella. A *sinistra* sono elencati tutti gli indici presenti, ed è possibile aggiungere o cancellare nuovi indici con i pulsanti **"+"** e **"-"** che vedete in basso. Nella parte *centrale* è possibile modificare il nome ed il tipo dell'indice. Nella parte *destra* vengono elencati i campi che fanno parte dell'indice. Per aggiungere un campo è necessario selezionarlo nell'elenco in alto e premere il pulsante **"+"** sull'estrema destra. In alternativa è possibile usare il drag'n'drop, ma a me non sempre riesce.

La Tab ***Foreign Keys*** sarà esaminata in dettaglio tra un attimo.

La Tab ***Column Details*** contiene, in una forma diversa, le stesse informazioni dell'elenco dei campi in alto.

Con la Tab in alto ***Table Options*** è possibile stabilire il *Tipo* di Tabella che desideriamo gestire. Abbiamo parlato nel Paragrafo precedente di ***MyIsam*** e ***InnoDb***: bene qui scegliamo quale *motore* gestirà i dati. Le Opzioni sono molte (ben sette!), ma a noi interessano solo le prime due.

Una caratteristica interessante di *MySql Administrator* è che alla fine delle modifiche, premendo il pulsante ***Apply Changes*** viene mostrato l'elenco dei comandi SQL che dovranno essere eseguiti sul Db. Questo è estremamente "didattico", perché traduce in SQL quello che abbiamo richiesto, permettendoci di capire meglio quello che accade.

5.5.2 Aggiungere le Tabelle a Mediateca

Seguendo le indicazioni sulla struttura dei Dati fornite nel Capitolo "*Il nostro Database di esempio*" non dovrebbe essere difficile creare la struttura di Mediateca. In figura i campi di TbMedia.

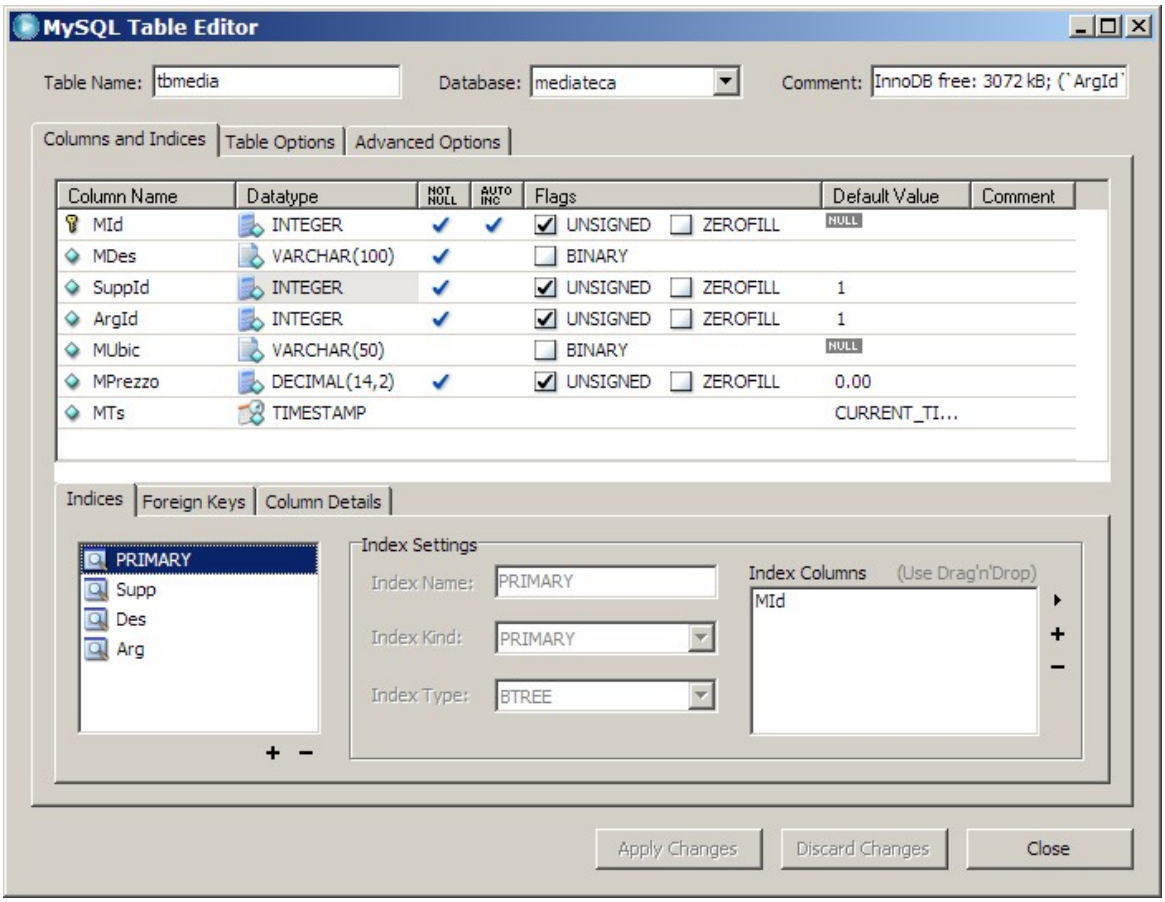


Figura 5.5.1: Modifica della struttura di una Tabella

TIP

La proprietà **valore predefinito** (*default value*) è importante. Se definiamo un campo numerico qualsiasi, non è saggio, per i motivi spiegati in precedenza, permettere l'archiviazione di valori nulli. Nel caso di aggiunta di una riga, però, dovremmo manualmente scrivere il valore 0 nel campo numerico. Basta quindi assegnare al campo il valore predefinito di 0 per toglierci il fastidio. *Quindi ai campi numerici è sempre opportuno assegnare il valore iniziale uguale a zero.* I Campi stringa possono anche *ammettere* il *null*, a meno che non si tratti di un indice, dove il *null* è sconsigliato. Anche in questo caso, se il *null* non è ammesso, può essere comodo assegnare un valore iniziale, magari uno spazio. Le ultime versioni di MySql Administrator sono un po' schizzinose sulla definizione dei valori predefiniti: se siete indecisi e non si tratta di un campo numerico assegnate pure *null*.

Per i Campi di tipo **Decimal**, la lunghezza va impostata secondo la forma ***"M,D"***, dove ***M*** è il numero complessivo di cifre desiderato e ***D*** il numero di Decimali. Il campo "*Mprezzo*" è stato definito come "14,2".

Tecnica

Per chi non ha mai approfondito l'argomento, il concetto di ***null*** può non essere chiaro. In effetti *null* significa "vuoto" cioè letteralmente "senza alcun valore di alcun tipo", quindi "privo di valore". Questo significa che un valore *null* è cosa diversa dallo *zero*, ed anche diverso dalla stringa vuota, cioè "". Per definizione un valore *null* non è confrontabile, e non può essere usato in operazioni aritmetiche. Perciò, ad esempio 0+*null* è scorretto (può dare un errore di sistema o ancora un *null*), e neppure ha senso dire che "pippo" viene prima o dopo *null*. Quindi occhio, amici miei, che questo è un punto fondamentale per ottenere, alla fine, dal nostro archivio valori sensati.

Notate che Administrator è abbastanza "intelligente" da capire che se chiamo il primo campo di una Tabella *MId* (quindi il nome contiene la stringa "Id", cioè "identifier") ed assegno un valore *Integer*, allora quel campo potrebbe essere una chiave primaria; perciò l'indice viene creato in automatico. Gli altri indici invece vanno creati manualmente, come spiegato nei paragrafi precedenti.

Nella Tab ***Table Option*** scegliamo il tipo di motore che desideriamo usare (per comodità nell'esempio useremo *InnoDb*, visto che ci interessa anche l'integrità referenziale).

5.5.3 Integrità Referenziale

Molti motori di Db hanno a disposizione un meccanismo di controllo che impedisce la cancellazione o la modifica delle chiavi esterne, o comunque permette di stabilire regole precise per la gestione di queste eventualità. Questi motori si occupano perciò di *preservare l'integrità referenziale del DataBase*. Per molto tempo gli sviluppatori di MySql hanno sostenuto che *MyIsam*, per mantenere le caratteristiche di leggerezza e velocità, non doveva occuparsi dell'integrità referenziale, che quindi era lasciata completamente sotto la responsabilità dell'utente. Fortunatamente la cose con *InnoDb* sono un po' cambiate.

Le Tabelle InnoDb supportano la gestione delle chiavi esterne, non tanto nelle query di relazione (cioè quando si usano in una query più tabelle), quanto appunto nel controllo dell'integrità referenziale. Per applicare ad una tabella una chiave esterna, è necessario che :

- le tabelle coinvolte siano tutte di tipo *InnoDb*;
- i campi da mettere in relazione siano dello stesso tipo e della stessa lunghezza;
- il campo nella tabella che *contiene la chiave esterna* (nel nostro caso *TbArgomenti*) deve avere un indice univoco e deve preferibilmente essere la chiave primaria;

- il campo nella tabella che *fa riferimento alla chiave esterna* (*TbMedia*) deve essere indicizzato;

Se tutte le condizioni precedenti sono verificate, possiamo impostare la Foreign Key.

5.5.4 Impostazione dell'Integrità Referenziale

MySql Administrator è, a quanto mi risulta, il primo Tool grafico ufficiale (cioè direttamente supportato da MySql Ab) che permette la gestione delle chiavi esterne. Se i prerequisiti che abbiamo già elencato sono tutti verificati, nella finestra di modifica della struttura della Tabella (nel nostro caso TbMedia), in basso, troviamo una Tab chiamata ***Foreign Keys*:**

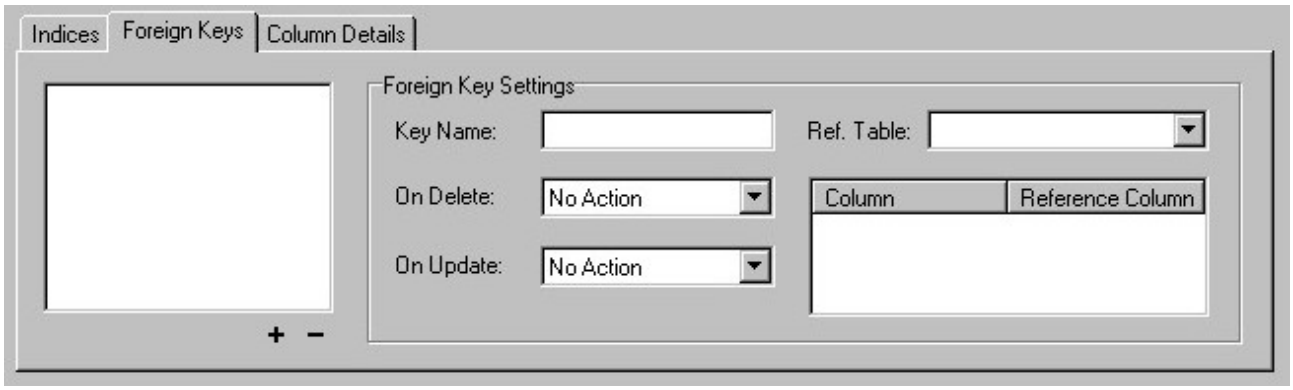


Figura 5.5.2 Finestra di gestione delle Chiavi Esterne

Vi ricordo che stiamo definendo una chiave esterna (*ArgId*) per la Tabella *TbMedia*, che faccia riferimento al Campo *ArgId* della Tabella *TbArgomenti*. Per aggiungere la chiave è necessario fare click sul pulsante **"+"** in basso a sinistra. Dopo aver assegnato un nome a piacere alla chiave, bisogna selezionare dalla casella a discesa ***Ref. Table*** qual'è la tabella che contiene il campo collegato (nel nostro caso *TbArgomenti*). Se esistono campi comuni tra le due tabelle (*ArgId*), il programma li aggiunge direttamente alla relazione (***Column***, ***Reference Column***) come in figura:

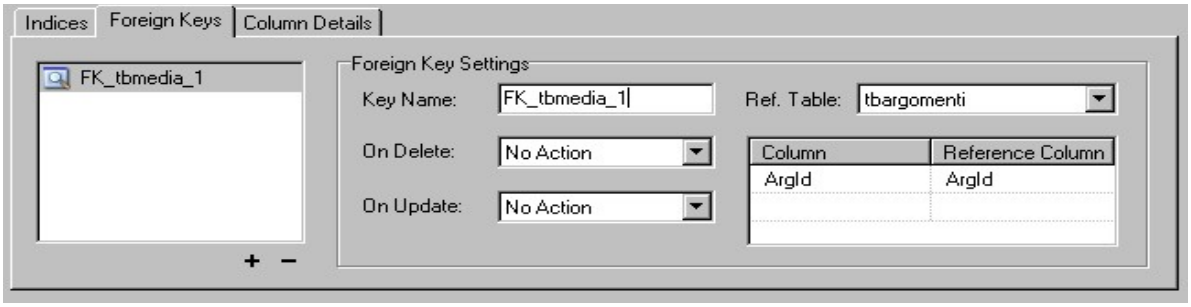


Figura 5.5.3: Chiave esterna degli Argomenti

Con il pulsante ***Apply Changes*** salviamo la chiave appena creata, e, se volete, date anche uno sguardo al comando SQL che *Administrator* si appresta ad eseguire:

```
ALTER TABLE `mediateca`.`tbmedia`
ADD FOREIGN KEY `FK_tbmedia_1` (`ArgId`)
REFERENCES `tbargomenti` (`ArgId`);
```

La sintassi è piuttosto comprensibile, quindi evito altri commenti. Quali risultati abbiamo ottenuto ? Per prima cosa se tentassimo di cancellare il Record con *Id* uguale a 5 (Gialli); otterremo questo messaggio di errore di questo tipo :

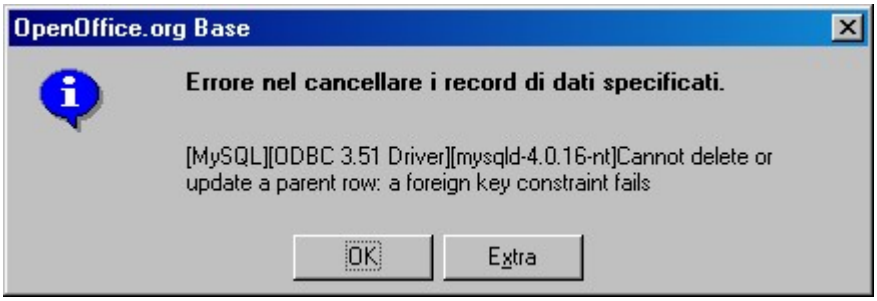


Figura 5.5.4 Messaggio di errore di integrità violata

Cioè il Motore di Db mi impedisce di cancellare una Chiave esterna referenziata in un'altra Tabella (ne nostro caso TbMedia). Significa, in parole povere, che finché in TbMedia esisterà un Record con ArgId uguale a 5, il Record con ArgId uguale a 5 di TbArgomenti NON POTRA' ESSERE CANCELLATO. In altre parole, non posso eliminare un genitore se esistono figli (e questo per chi crede che l'informatica manchi di etica...). Questo è già un buon risultato, ma andiamo avanti.

Se invece cercassimo di aggiungere a *TbMedia* un Record con un *ArgId* non presente nella Tabella *TbArgomenti* (ad esempio un valore 99, inesistente), il messaggio di errore sarebbe:

Figura 5.5.5 Errore di violazione dell'integrità referenziale

Cioè il Motore di Db mi impedisce di aggiungere un Record con un valore nel campo ArgId NON PRESENTE in TbArgomenti. Cioè non è possibile inserire figli senza genitore. Era quello che volevamo, no?

Torniamo però un passo indietro. Nella finestra di impostazione della chiave esterna, si possono notare due altre opzioni, precisamente ***ON UPDATE*** e ***ON DELETE***, entrambe settate su ***NO ACTION***. Bene, se quello che abbiamo appena visto è il comportamento standard del motore di Db (*NO ACTION*), possiamo comunque eventualmente scegliere delle opzioni alternative ne caso si modifichi (*ON UPDATE*) o si cancelli (*ON DELETE*) la chiave esterna (cioè il genitore di una serie di record presenti nella Tabella "figlia"). In particolare :

CASCADE estende la variazione della chiave esterna alla Tabella "figlia"; nel caso di ***ON UPDATE CASCADE***, se, ad esempio, in *TbArgomenti* modifichiamo *ArgId* da 5 a 55, tutti i record di *TbMedia* con *ArgId* uguale a 5 saranno aggiornati di conseguenza. Nel caso di ***ON DELETE CASCADE***, però, la cancellazione di una chiave esterna NON VIENE PIU' IMPEDITA, e vengono eliminati a cascata anche tutti i record della Tabella "figlia". *Cioè se cancello "5" in TbArgomenti, non avrò più nessun Giallo in TbMedia.*

SET NULL invece si limita ad impostare a Null tutte le chiavi figlie del genitore modificato.

RESTRICT infine, è equivalente a *NO ACTION*, cioè controlla l'applicazione dell'integrità.

Come vedete l'integrità referenziale è uno strumento molto potente, perché permette di stabilire regole a livello di motore di Db che non possono essere scavalcate da manovre errate dell'utente. Queste regole però vanno impostate con cognizione di causa, e, soprattutto, in fase di progettazione *iniziale* del Database (o, adottando la nomenclatura di MySql, della struttura del Catalogo). Applicare una nuova regola di integrità ad una Tabella già colma di dati infatti non sempre è possibile, perché alcuni dei record immessi potrebbero già non rispettare la regola stessa.

5.6 Backup degli Archivi MySql

Una sana politica di Backup, come ben sapete, ci mette al riparo da disastri che provocano giornate di lavoro perse e mal di testa cronici. Per quanto la natura umana sia pervasa dalla convinzione che "a me non succederà mai", la logica e la Legge di Murphy ci dicono che invece sicuramente prima o poi accadrà. Quindi, di fronte all'ineluttabile, meglio essere preparati (stiamo parlando di informatica, forse è meglio non generalizzare...).

La domanda da porsi, quindi, è : *come si fa un Backup degli Archivi di MySql ?*

Fortunatamente la risposta è abbastanza semplice: *basta trasferire periodicamente i file dei Dati su un altro supporto*. Già, ma *quali file ?*

Per le Tabelle ***MyIsam***, per ogni "catalogo" (cioè Database) esiste una cartella nella dir dei Dati definita per il Server. In Windows basta cercare in c:\programmi\mysql\MYSQL Server X.X\data\. Un Backup dell'intero percorso mette al sicuro tutti i cataloghi. In Linux, a seconda

dei parametri di installazione, esiste una struttura equivalente di solito in /usr/local/mysql. *Ovviamente questo metodo funziona quando il Server non è in esecuzione.*

Se utilizziamo anche Tabelle ***InnoDb***, a quanto detto prima vanno aggiunti i file di Dati e di Log di InnoDb, e sono tutti quelli che iniziano per "ib" (ib*).

Se non è possibile disattivare il Server, il manuale di MySql elenca con dovizia di particolari tutta una serie di metodi alternativi, alcuni basati su tool da linea di comando, per ottenere lo stesso risultato senza grossi sforzi. Non credo però che sia il caso di approfondire, anche perché abbiamo, volendo, il supporto di una ottima interfaccia grafica....

5.6.1 Backup dei Dati

Alla voce ***Backup*** di MySql Administrator, basta selezionare ***New Project*** col pulsante in basso e fornire i parametri del nostro Backup. In particolare, è possibile selezionare più di un Catalogo, oppure, all'interno di un Catalogo, solo alcune Tabelle. Il pulsante ***Save Project*** permette il salvataggio dei parametri di Backup, in modo da non doverli reinserire in seguito.

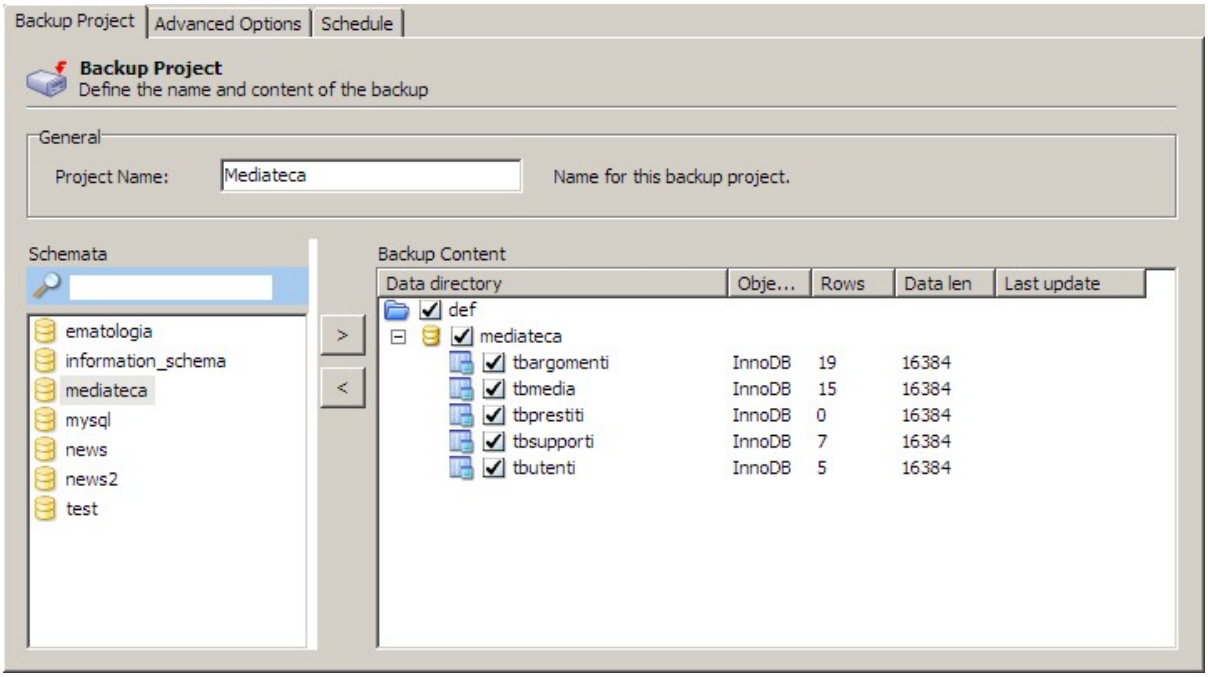


Figura 5.6.1: Backup con MySql Administrator

La Tab ***Advanced Options*** serve a definire alcune opzioni utili. Se le tabelle sono di tipo **MyIsam**, selezionate, nel ***Backup Execution Method*** la voce ***Lock All Tables***, che "blocca" le Tabelle per permettere un Backup sicuro dei dati. Se invece avete usato **InnoDb**, selezionate l'opzione ***InnoDb OnLine Backup***, più adatta allo scopo.

In ***Output File Options*** c'è poco da selezionare, e direi che è saggio lasciare le opzioni di default, anche se le scelte sono abbastanza intuitive. Il ***Backup Type*** permesso, almeno per questa versione, è unicamente ***Sql file***; questo significa che tutto il Db (o le tabelle che avete selezionato) sarà salvato come un file di testo con estensione *.sql* contenente i comandi SQL

necessari a ricreare il Db da zero. Non si tratta perciò di una copia "binaria" dei dati, ma della "trasposizione" SQL dell'intera struttura e del contenuto del Catalogo.

Non ci credete ? Allora date un'occhiata all'inizio del File creato dal Backup della Tabella tbArgomenti di Mediateca :

```
[.....]
--
-- Create schema mediateca
--

CREATE DATABASE /*!32312 IF NOT EXISTS*/ mediateca;
USE mediateca;

--
-- Table structure for table `mediateca`.`tbargomenti`
--

DROP TABLE IF EXISTS `tbargomenti`;
CREATE TABLE `tbargomenti` (
  `ArgId` int(10) unsigned NOT NULL auto_increment,
  `ArgDes` varchar(30) NOT NULL default '',
    `ArgTs` timestamp NOT NULL default CURRENT_TIMESTAMP on update
CURRENT_TIMESTAMP,
  PRIMARY KEY  (`ArgId`),
  KEY `Arg` (`ArgDes`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `mediateca`.`tbargomenti`
--

/*!40000 ALTER TABLE `tbargomenti` DISABLE KEYS */;
INSERT INTO `tbargomenti` (`ArgId`,`ArgDes`,`ArgTs`) VALUES
 (1,'Rock','2004-04-08 12:00:19'),
 (2,'Classica','2004-04-08 12:00:22'),
 (3,'Pop','2004-04-11 11:03:22'),
 (4,'Filosofia','2004-04-08 12:00:32'),
 (5,'Giallo','2005-04-26 21:33:31'),
 (6,'Saggio','2004-09-16 19:02:14'),
 (7,'Sistemi Operativi','2004-04-08 12:01:02'),
 (8,'Samba','2004-04-08 12:01:05'),
 (9,'Reti Locali','2005-05-05 20:16:53'),
 (10,'Giochi','2004-04-10 17:34:09'),
 (15,'Fantasy','2004-04-12 18:35:26'),
 (16,'Linux','2004-04-13 18:41:30'),
 (17,'Romanzo','2004-04-16 09:35:02'),
 (18,'Avventura','2004-04-16 09:38:18'),
 (19,'Windows','2005-04-29 11:42:59'),
 (20,'Ftp','2005-04-07 12:23:05');
/*!40000 ALTER TABLE `tbargomenti` ENABLE KEYS */;
[....]
```

Questo metodo ha dei vantaggi. Siccome abbiamo un file di testo con comandi SQL (quasi) standard, potrebbe essere usato, ad esempio, per il trasferimento dei dati ad altri motori di Db. Inoltre questo tipo di Backup è l'ideale per il passaggio di dati tra server MySql diversi, o tra Cataloghi diversi nello stesso Server. Infatti...

5.6.2 Restore dei Dati

La Tab ***Restore*** di MySql Administrator è un esempio di semplicità. Per prima cosa, col pulsante ***Open Backup File*** si seleziona il Backup da recuperare, quindi si può scegliere, in ***Target Schema*** se si desidera recuperare i dati nella posizione originale oppure in un altro catalogo. Se il Db non esiste, si può richiedere di crearlo da zero. Infine nella Tab ***Restore Content*** il pulsante ***Analyze Backup*** ci permette di selezionare solo alcune tabelle per il restore. Un lavoro davvero ottimo.

5.7 La sicurezza : Utenti e Diritti

MySql è un Db particolare e forse la cosa sta lentamente insinuandosi nelle vostre sveglie menti, dopo aver letto fin qui. "Particolare" significa che su alcuni aspetti questo software ha un approccio direi "nordico", vista l'origine. Il "nordico" va inteso nel senso che, per certi versi, gli sviluppatori hanno una visione delle cose assai personale e non vogliono cambiarla. Quelli di voi che hanno usato altri Db ed hanno a cuore la sicurezza dei propri dati, continuando nella lettura di questo paragrafo potrebbero storcere il naso (io direi meglio "sentire un sudore freddo su per la spina dorsale"), ma che volete, questi di MySql passano metà dell'anno al buio....

Allora, tanto per cominciare, ecco tre succose informazioni che servono a farsi un'idea :

1. ogni utente del Db è identificato dal **nome**, **dall'host** da cui si connette e dalla **password**; questo vuol dire che il nostro amico Alfred, che si collega sia dalla macchina che ospita il Server, sia dal Client in Rete deve avere DUE ACCOUNT;
2. il concetto di **Gruppi di Utenti** è ***sconosciuto*** a MySql, per cui non si possono assegnare diritti ad un Gruppo e quindi semplicemente aggiungere un Utente al Gruppo; sarebbe troppo comodo, e quindi dovrete assegnare i diritti singolarmente AD OGNI UTENTE;
3. appena dopo aver installato il Server, l'utente **root** (cioè l'amministratore) ha ***password vuota***; come se non bastasse, potrebbe esistere anche un account che permette l'accesso anonimo, dalla macchina che ospita il Server (localhost), con i poteri di Amministratore (in Windows) e di Ospite (in Unix / Linux), sempre con la password vuota (questo è un po' cambiato nelle ultime versioni: l'installer Windows permette di configurare alcuni importanti aspetti della sicurezza prima di avviare effettivamente il Server)

Bene, se finora pensavate che forse non era una cattiva idea migrare in MySql quel Db con una cinquantina di utenti che avete in Azienda, ora forse qualche dubbio in verità vi assale...

Ma non disperate: mai fermarsi alla prima impressione, e poi di tutto bisogna accettare sia i pregi che i difetti. In verità questa parte relativa alla sicurezza mi sembra un po' trascurata in MySql, e quindi se vogliamo mettere su un Server sicuro c'è un po' da lavorare. Cominciamo quindi con i concetti di base.

Il modello di sicurezza usato da MySql è, in realtà, assai semplice. I livelli di autenticazione (e quindi di assegnazione dei diritti) sono due; il primo permette l'accesso al Server e concede i **privilegi globali**, il secondo si occupa di gestire i diritti sui **singoli Database** (o Schema) presenti nel Server stesso. Inoltre si può scendere fino al dettaglio della singola colonna (o campo) di una Tabella (ad esempio stabilendo che per un utente quel campo deve essere in sola lettura). Allora, vediamo quali sono i diritti assegnabili :

Diritto	**Contesto di assegnazione**
ALTER	tabelle
DELETE	tabelle
INDEX	tabelle
INSERT	tabelle
SELECT	tabelle
UPDATE	tabelle
CREATE	cataloghi, tabelle, o indici
DROP	cataloghi o tabelle
GRANT	cataloghi o tabelle
REFERENCES	NON USATO
CREATE TEMPORARY TABLES	amministrazione del server
EXECUTE	NON USATO
FILE	accesso ai file
LOCK TABLES	amministrazione del server
PROCESS	amministrazione del server
RELOAD	amministrazione del server
REPLICATION CLIENT	amministrazione del server
REPLICATION SLAVE	amministrazione del server
SHOW DATABASES	amministrazione del server
SHUTDOWN	amministrazione del server
SUPER	amministrazione del server

Come vedete, in generale il nome è identico alla corrispondente istruzione SQL, quindi il significato è lampante. Il privilegio di GRANT permette di trasferire i propri diritti ad un altro Utente. *Un **privilegio globale** vale per tutti gli Schemi (Database) gestiti dal Server, e per il Server stesso.*

Abbiamo già detto che ogni **utente** viene identificato da tre parametri: il ***nome***, ***l'host*** da cui si connette e la ***password***. Per "host da cui si connette" intendiamo il **nome** o **l'indirizzo IP** del Computer che richiede la connessione. La macchina su cui gira il Server è identificata sempre dalla stringa "*localhost*". Quindi l'utente pippo@testws è diverso da pippo@multimedia. *Diverso* significa che può avere password diversa e privilegi differenti. Per fortuna possiamo

almeno usare delle wildcards (ma solo nel campo host), quindi ad esempio ***pippo@%*** significa "*l'utente Pippo, da qualsiasi workstation si connetta*". Se si indica solo il nome di host, invece, per MySql significa "*accesso anonimo dall'host*", meglio ancora "*tutti gli utenti dell'host*". Però questo implica che, ad esempio, ***@localhost*** significa "accesso anonimo dal localhost", ***@testws*** "accesso anonimo da testws", ***@%*** "accesso anonimo da qualunque PC collegato in rete" (con tanti ringraziamenti da parte di un eventuale intruso).

Ora, se vi sentite confusi, sappiate di essere in buona compagnia; ma non disperate, non è ancora finita. Supponiamo di voler affidare all'utente *pippo* la gestione del Database *Mediateca*; vogliamo inoltra che pippo possa collegarsi da tutti gli host della nostra rete. Quindi aggiungiamo l'utente *pippo@%* e concediamo tutti i diritti su *Mediateca*. Pippo si collega senza problemi e svolge il suo lavoro. Per caso abbiamo inoltre, sulla nostra rete, un PC che si chiama *testws*, e desideriamo che tutti gli utenti di questo host accedano al Database *news2*; quindi creiamo l'utente anonimo *@testws* ed assegniamo i diritti sul catalogo *news2*. La situazione attuale sarebbe, quindi :

Utente	Significato	database	diritti
pippo@%	pippo, collegato da qualsiasi host	mediateca	gestione
@testws	qualsiasi utente anonimo dell'host testws	news2	gestione

A questo punto *pippo* si siede al Pc *testws* e chiede l'accesso a *mediateca*: MySql rifiuta la connessione. Questo vuol dire che *i diritti di accesso anonimo dell'host invalidano quelli del singolo utente*. Perciò, **O** consentiamo l'accesso anonimo da *testws* anche a *mediateca*, ma questo apre il db anche agli altri utenti, **OPPURE** aggiungiamo un utente pippo@testws con i diritti appropriati.

A questo punto *pippo*, come utente di *testws*, vuole accede a *news2*, quindi fornisce le sue credenziali (nome e password) e MySql gli *rifiuta la connessione*. Cioè *per accedere a news2 da testws l'utente deve essere anonimo*. In caso contrario MySql controlla i diritti di pippo, e se non è specificato che può accedere a news2, nega la connessione.

Tutto questo ha una logica (anche se per me piuttosto oscura) ed è abbastanza ben spiegato ai capitoli 5.4 e 5.5 del manuale di MySql. Quindi coraggio e buona lettura. Vi risparmio anche i dettagli di come tutte queste informazioni siano archiviate nel catalogo (Database) di nome MySql (non sono un sadico, e voi interrompereste la lettura dopo due righe). Siccome però è probabile che (come me) avete capito poco, qui di seguito vi passo qualche semplice consiglio che viene fuori dalla mia esperienza di utente. Allora:

1. dopo l'installazione di MySql, per prima cosa assegnate una *password* all'utente **root**; ricordate che gli utenti root sono DUE, uno per la connessione da localhost, l'altro per la

connessione dalla rete; se desiderate che root possa collegarsi SOLO dal localhost, eliminate l'utente **root@%;**

2. eliminate o regolamentate l'***accesso anonimo*** al server; una buona soluzione è cancellare del tutto gli utenti **@%** e **@localhost;**
3. se non serve controllare l'host di collegamento, aggiungete solo utenti del tipo **utente@%**; questo garantisce comunque la verifica dell'autenticazione senza specificare da quale workstation la connessione debba avvenire;
4. se invece volete controllare l'host di collegamento, tenete presente che il nome host viene passato direttamente dal sistema (cioè voi non potete specificarlo);
5. il modello di sicurezza di MySql non si basa su quello del Sistema Operativo su cui è in esecuzione, quindi aggiungere troppi utenti complica la gestione; piuttosto, se sul server sono presenti più database, può essere utile seguire un modello "orientato al catalogo". Ad esempio, per l'accesso a mediateca, posso aggiungere un utente **mediateca@%,** con i soli privilegi su questo db;
6. ricordate che di solito l'accesso a MySql avviene attraverso la configurazione di un **DSN** (Data Source Name) di tipo ODBC sia in Linux che in Windows; i DSN possono essere specifici di un utente o di una macchina; potrei voler assegnare un DSN già configurato all'utente od alla macchina, senza rivelare all'utente la password di accesso a MySql; quindi posso aggiungere un DSN con nome utente "mediateca", uno con nome utente "news2" etc.; in questo modo all'utente non verrà MAI chiesta una password di connessione; questa soluzione però non funziona in Linux, perché nel DSN non si può specificare un nome utente ed una password;
7. limitate l'assegnazione di privilegi globali solo agli account che ne hanno effettivamente bisogno; limitate l'accesso di ogni utente ad un singolo Database, salvo casi eccezionali;
8. le password di MySql sono lunghe fino a sedici caratteri, ma non è possibile impostare alcuna policy (numero minimo di caratteri, scadenza etc.) su di esse; quindi è fondamentale la scelta di password non banali.

Bene, ora che abbiamo studiato la teoria, passiamo alla pratica.

TIP

Se, nei parametri del Server, avete disabilitato la gestione dei nomi allora tutti i permessi devono far riferimento all'indirizzo fisico della macchina. Non è perciò consentito (o almeno non funziona) un utente del tipo fc@homeserver. Questo, se ad esempio usiamo un server DHCP per l'assegnazione automatica degli indirizzi, è molto scomodo perché in teoria la stessa macchina potrebbe nel tempo assumere indirizzi IP diversi.

5.7.1 Gestione della sicurezza da linea di comando

Tutta la parte di gestione di un Server MySql (e quindi anche la sicurezza) può essere eseguita tramite tool a linea di comando presenti nella cartella **..\mysql\bin**, a cominciare dal già citato **mysql**. Credo non sia questa l'occasione per approfondire l'argomento. Sappiate almeno però che a livello SQL l'assegnazione e la revoca di diritti avviene tramite gli statements *GRANT* e *REVOKE*. A titolo di esempio ecco la sintassi di *GRANT* presa dal manuale di MySql:

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON {tbl_name | * | *.* | db_name.*}
    TO user [IDENTIFIED BY [PASSWORD] 'password']
        [, user [IDENTIFIED BY [PASSWORD] 'password']] ...
    [REQUIRE
        NONE |
        [{SSL| X509}]
        [CIPHER cipher [AND]]
        [ISSUER issuer [AND]]
        [SUBJECT subject]]
    [WITH [GRANT OPTION | MAX_QUERIES_PER_HOUR count |
                          MAX_UPDATES_PER_HOUR count |
                          MAX_CONNECTIONS_PER_HOUR count]]
```

Inoltre, poiché tutta la struttura è memorizzata nelle tabelle user, db e host del catalogo mysql, è possibile usare anche semplici istruzioni INSERT, DELETE od UPDATE, purché si sappia quello che si sta facendo. Ma a noi va bene anche la GUI di MySQL Administrator...

5.7.2 MySql Administrator : la sicurezza

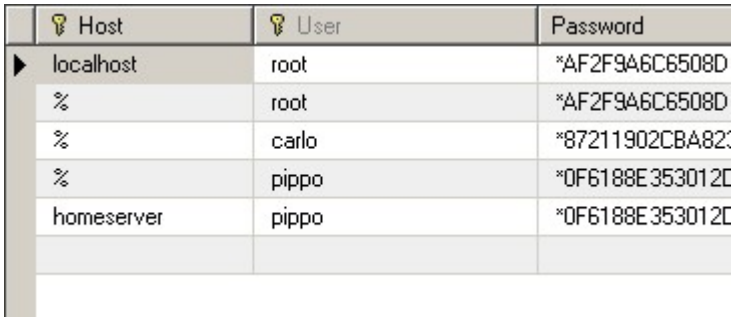
Selezionando la voce User Administration, nella finestra in basso a destra vengono elencati gli utenti autorizzati per il server, come in figura:

Figura 5.7.1: Gestione utenti

Ogni utente è identificato da un nome, e, se necessario, anche dagli hosts specificati per il suo accesso. *Se non viene indicato alcun host, si intende "ogni host"*. Nella sintassi di mysql, quindi il "carlo" della figura sarebbe "carlo@%". Se sono elencati altri host, ognuno rappresenta un utente diverso. Quindi nel nostro caso avremo ***pippo@%***, ***pippo@homeserver***.

Con un menu contestuale attivato dal tasto destro in questa finestra è possibile *aggiungere* un *nuovo utente*. Sempre con un menu contestuale, selezionato un utente si possono eventualmente *aggiungere nuovi host di connessione*. Questa rappresentazione può sembrare

lineare, ma secondo me non lo è affatto. Infatti un nuovo utente viene aggiunto sempre nella forma utente@%, cioè è possibile il collegamento da qualsiasi host. Anche se si aggiunge un host specifico (come per pippo@homeserver), non viene eliminato pippo@%. Questa situazione si vede bene se si apre la tabella ***user*** del Database di sistema ***mysql,*** come in figura:



Host	User	Password
localhost	root	*AF2F9A6C6508D
%	root	*AF2F9A6C6508D
%	carlo	*87211902CBA82
%	pippo	*0F6188E353012D
homeserver	pippo	*0F6188E353012D

Figura 5.7.2: Utenti di un Server MySql

qui si comprende assai meglio che in effetti ***pippo@%*** e ***pippo@homeserver*** sono proprio DUE UTENTI DIVERSI. Per fare in modo che *pippo* si colleghi SOLO da *homeserver* è necessario cancellare manualmente l'utente *pippo@%;* il problema è che, però, in questo caso l'utente viene visualizzato in Administrator solo come pippo (direi che questo modulo ha ancora bisogno di una limatura da parte dei programmatori...). In ogni caso, all'utente è possibile assegnare i privilegi con un comodo sistema mostrato in figura. Abbiamo le tab in alto riguardanti i privilegi globali, i Database (schema), le Tabelle e le colonne.

TIP

In realtà, nella installazione standard di MySql Administrator, nelle Tab compare solo la voce "*schema privileges*"; per abilitare le altre due, nella voce di menu Tools->*Options* è necessario abilitare le caselle di spunta alla voce "*User Administration*".

La gestione è abbastanza semplice; una volta selezionati l'utente e la sezione che ci interessa, tutto si risolve spostando i privilegi elencati a video dalla colonna ***available*** (*disponibile*) a quella ***assigned*** (*assegnato*), come in figura.

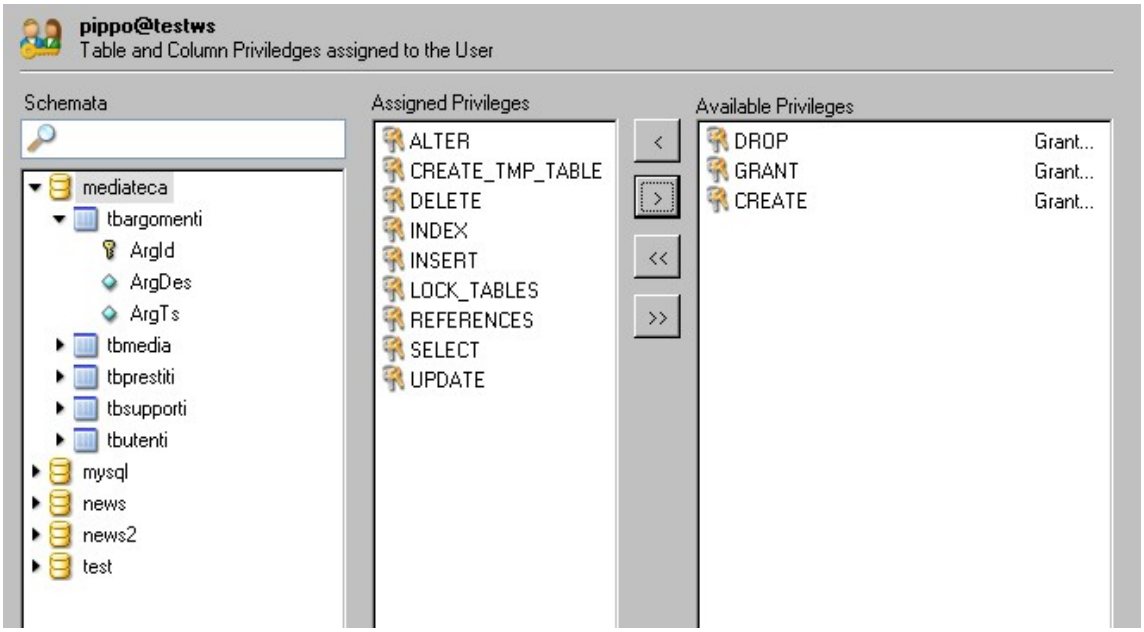


Figura 5.7.3 Assegnazione diritti ad un Utente

5.8 Importazione dei Dati da altri Db

Quando si decide di migrare da un Server di Database ad un altro è sempre abbastanza complicato spostare i dati da una parte all'altra. Nel caso di MySQL è possibile usare più di un sistema (compreso il "transito" attraverso il modulo Base di OOo), ma quello più semplice è l'uso di Migration Toolkit.

5.8.1 Migration Toolkit

Il **Migration Toolkit** è scaricabile da sito di MySql, ed è un Wizard che permette l'importazione di dati da ***Ms Access***, ***Ms Sql Server***, ***Oracle***, un driver ***JBDC*** generico oppure un'altra istanza di MySQL. Il funzionamento è piuttosto intuitivo, quindi vi risparmio i dettagli.

Con *Ms Access* funziona egregiamente: i tipi di dati vengono conservati e, quando non disponibili in MySQL, "tradotti" con intelligenza (ad es. *decimal(19,4)* per il tipo *valuta* oppure *tinyint(1)* per *Si/No*). Si può trasferire un intero Database oppure singole Tabelle; in definitiva si tratta di un ottimo strumento.

5.9 Novità della Versione 5

La Versione 5 di MySQL ha introdotto caratteristiche lungamente attese dalla comunità degli Utenti. In breve, riportando solo quelle essenziali, abbiamo:

- un nuovo tipo di dato **BIT** (equivale a *tinyint(1)*)

- la presenza dell'**Information Schema**, una sorta di catalogo generale di tutti gli oggetti del Database
- incremento della precisione matematica dei calcoli
- due nuovi motori di archiviazione: ARCHIVE e FEDERATED
- gestione di **Stored Procedure** e **Stored Function**
- maggiore conformità allo standard SQL
- supporto (ancora limitato) ai **trigger**
- aumento della capacità di archiviazione del tipo VARCHAR fino a 65.532 caratteri
- supporto per le Viste (**Views**)
- aumento generalizzato della velocità di elaborazione

I progressi sono notevoli, e permetteranno sicuramente a MySQL di acquisire nuovi utenti a scapito della concorrenza.

5.9.1 Creazione di Viste

La creazione di ***viste*** in MySQL 5 è abbastanza semplice: si tratta di utilizzare il comando SQL **CREATE VIEW**. Ad esempio, per la nostra mediateca, il comando potrebbe essere:

```
CREATE VIEW 'mediateca`.`media` AS
SELECT
    `tbmedia`.`MId` AS `Mid`,
    `tbmedia`.`MDes` AS `Mdes`,
    `tbsupporti`.`SuppDes` AS `SuppDes`,
    `tbargomenti`.`ArgDes` AS `ArgDes`
FROM
    ((`tbsupporti` join `tbmedia`) join `tbargomenti`)
WHERE
    ((`tbsupporti`.`SuppId` = `tbmedia`.`SuppId`)
    and (`tbargomenti`.`ArgId` = `tbmedia`.`ArgId`))
ORDER BY `tbmedia`.`MDes`;
```

La sintassi è molto simile alla **SELECT** normale, bisogna solo specificare il nome della vista nella forma ***database.nome***. Una volta creata, la vista può ovviamente essere modificata o cancellata con appositi comandi SQL. In teoria le viste dovrebbero poter essere gestite anche con l'interfaccia grafica di MySQL Administrator, ma la versione attuale (la 1.1.5) non è affidabile sotto questo aspetto.

5.10 Esecuzione di comandi SQL

Potremmo ora domandarci come fare ad "ordinare" al Server l'esecuzione di un comando SQL. I metodi sono sostanzialmente due:

1. usare l'utility a linea di comando **mysql** inclusa nel server;
2. trasferire al server il comando attraverso un Tool esterno (magari ad interfaccia grafica);

Se scegliamo la prima soluzione, dobbiamo lanciare il programma **mysql** contenuto nella dir **..\bin** dell'installazione del Server (in Windows di solito ***c:\mysql\bin***). Si tratta di una shell a linea di comando, quindi ci troveremo più o meno nella situazione in figura:

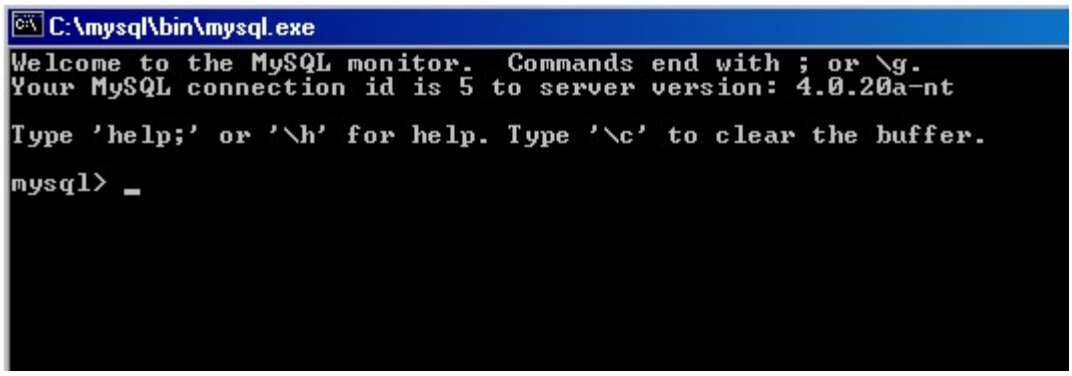


Figura 5.10.1 Il Client MySql

Per noi che abbiamo usato il DBase II la cosa è abbastanza familiare; per voi abituati alle interfacce grafiche potrebbe essere disorientante. Comunque non spaventatevi. La prima cosa da fare è selezionare il Db da usare quindi :

```
use mediateca
```

A questo punto possiamo scrivere qualunque comando SQL valido, con l'accortezza di terminare l'istruzione con un punto e virgola. Ad esempio :

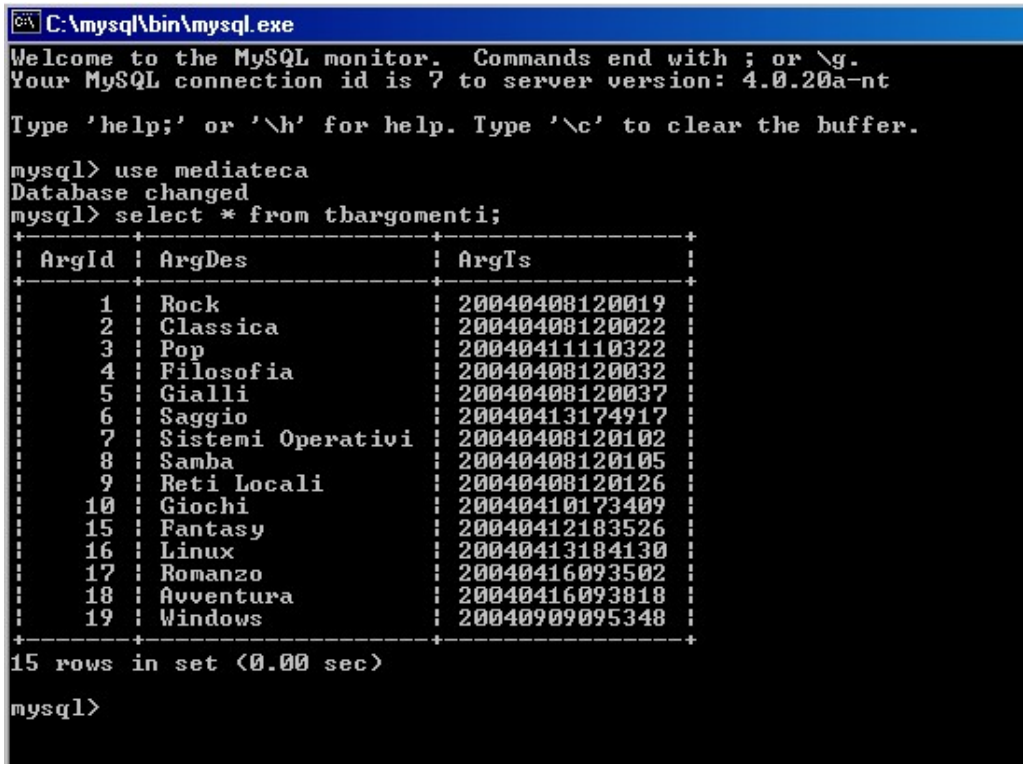


Figura 5.10.2 Il Comando Select

Se premiamo INVIO senza chiudere con il punto e virgola, è possibile continuare la scrittura sulla riga successiva, il che è comodo perché molti statements SQL sono lunghi e complessi. Perciò :

Figura 5.10.3 Ancora un Comando Select

Con questo semplice tool possiamo passare al Server *qualsiasi comando SQL*. Per uscire, un semplice **EXIT** chiude le operazioni. Con un po' di pratica ***MYSQL Monitor*** (nome ufficiale del tool) si rivela abbastanza immediato e semplice da usare. In più, può accettare in input anche un file di testo comprendente più istruzioni SQL, quindi si rivela utile in molti casi. Esistono però varie alternative, come adesso vedremo.

5.11 Altri Tools Grafici per la gestione di MySql

MySql Administrator, come abbiamo visto, è il nuovo strumento di amministrazione ufficiale per MySql. Sul sito www.mysql.com è comunque disponibile un altro Tool, e precisamente il **Query Browser**. Si tratta di un programma molto interessante orientato alla costruzione grafica di query SQL, non tanto nella maniera utilizzata in modalità Disegno di OOo, quanto nel supporto alla sintassi di TUTTI i comandi SQL (e quindi non solo delle SELECT). Inoltre dispone di uno strumento di scripting per la scrittura e l'esecuzione di Procedure SQL, anteprima di quello che sarà il supporto alle Stored procedure di MySQL 5.0.

Altro programma da considerare è **Db Manager Professional** di Db Tools, scaricabile gratuitamente dal sito www.dbtools.com.br. Questo software permette di fare tutte le cose essenziali necessarie all'amministrazione di un Db in modo semplice con una Gui efficace e senza fronzoli. Inoltre può essere usato anche con server Postgres, quindi due piccioni....

Se proprio volete affrontare il mondo dei Db server da professionisti, quello che fa per voi allora è l'ottimo **Fabforce Db Designer 4**, un progetto OpenSource scaricabile da www.fabforce.net. Si tratta di un "disegnatore" di Database di alto livello, che concentra l'attenzione sulla struttura logica delle basi di dati, svincolandosi dal singolo motore di Db. Si interfaccia con MySql e con Oracle, ma anche con ODBC, quindi con quasi tutto. Tools dedicato a chi di Db è già esperto, e quindi sconsigliato ai deboli di cuore.

6. Database Server PostgreSQL

PostgreSQL è l'alternativa a *MySql* per quanto riguarda i Server Open Source. A differenza di MySql, Postgres segue una politica di Licensing molto più canonica, e perfettamente aderente alle regole GNU. Inoltre per certi versi può essere considerato più "professionale" perché implementa caratteristiche (come le *viste* e le *stored procedure*) comuni a molti server di Db commerciali, ma assenti in MySql (fino alla versione 5). Ovviamente questo appesantisce un po' il codice, quindi viene da alcuni considerato meno performante del "collega". D'altro canto la maggiore aderenza agli standard SQL e l'intrinseca "robustezza" lo fa preferire in molte applicazioni "mission critical" dove l'integrità dei dati è più importante delle pure prestazioni.

Fino a poco tempo fa Postgres non poteva considerarsi realmente multi piattaforma, perché la versione Windows utilizzava uno strato di "emulazione" che ne penalizzava molto le performance. Dalla versione 8.0 è invece disponibile un eseguibile "nativo" per il Sistema Operativo di Zio Bill, quindi anche questa difficoltà sembra superata.

6.1 Installazione Windows

6.1.1 Il Server

Dalla versione 8.0 è disponibile un comodo installer che rende estremamente semplice avere un Server Postgres attivo è funzionante in pochi minuti. Per i nostri scopi useremo la nuova versione 8.1. Basta scaricare da http://www.postgresql.org/ il file Zip in formato Win32, nel nostro caso *postgresql-8.1.0.zip,* decomprimere e lanciare il file in formato *.msi*.

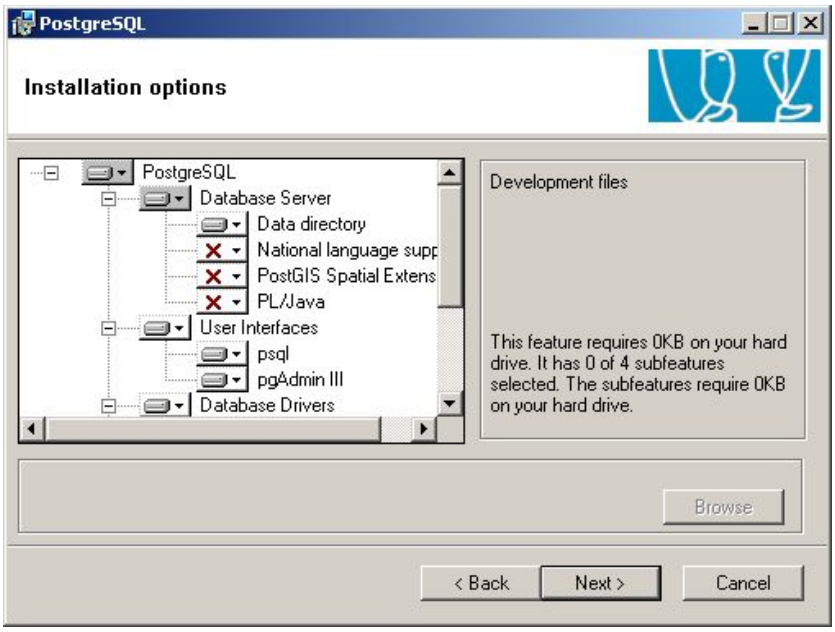


Figura 6.1.1: Scelta dei componenti per l'installazione

La procedura permette di scegliere quali componenti installare tra: ***Database Server*** (con relative estensioni), ***User Interfaces*** (*psql*, a linea di comando e *pgAdmin III* con interfaccia grafica), ***Database driver*** (ODBC, JDBC, .NET e OLEDB). Se si desidera solo accedere ad un server Postgres su un'altra macchina, possono essere sufficienti i *Database driver* e magari le *User Interfaces*. In caso contrario sarà utile lasciare le impostazioni consigliate, a meno di esigenze particolari.

Se è nostra intenzione installare un Server, nella fase successiva dovremo decidere se definire Postgres come Servizio (scelta consigliata), magari eseguito automaticamente all'avvio. In questo caso è necessario creare un utente, sulla macchina Windows che esegue il Server, che abbia i diritti di avviare il servizio stesso. Il passaggio può essere eseguito automaticamente dall'installer.

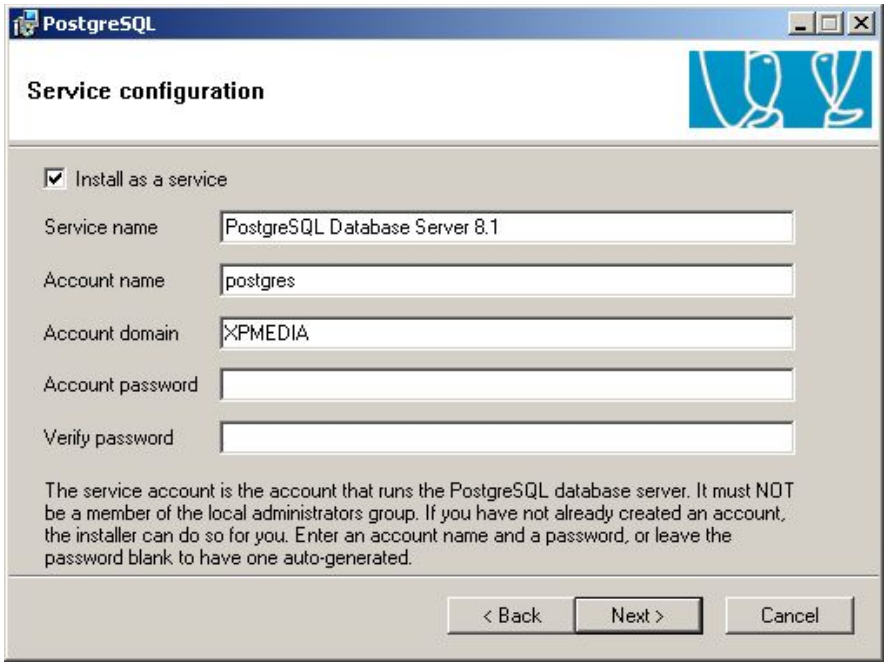


Figura 6.1.2: Avvio del Server come servizio

TIP

Se nel PC è stata installata in precedenza una versione di PostgreSQL, probabilmente l'utente *postgres* è già esistente. In questo caso per completare in modo corretto questo passaggio è necessario preventivamente *cancellare* il vecchio utente *postgres* per permettere la creazione del nuovo.

Poi è necessario selezionare il linguaggio ed il sistema di encoding per la memorizzazione delle stringhe. Il programma di installazione propone SQL_ASCII, ma è opportuno consultare la documentazione per capire se possono esserci problemi. In ogni caso, questa scelta può essere modificata alla creazione di un nuovo Db. Inoltre è bene assegnare una password

all'amministratore del Server di Db (*superuser*), il cui nome viene proposto come *postgres* (sappiamo bene che la fantasia non è una dota degli sviluppatori...).

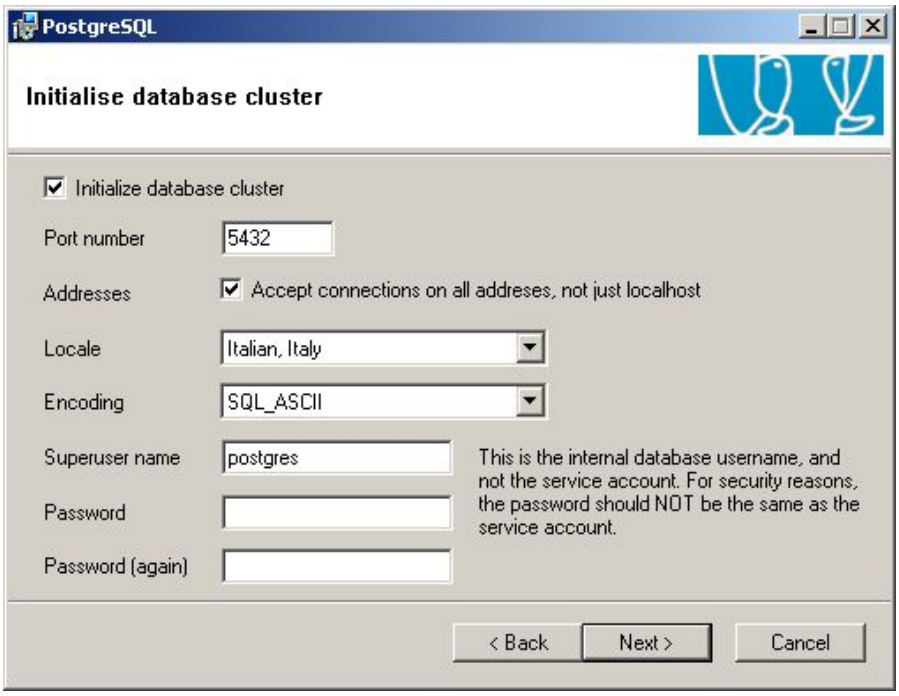


Figura 6.1.3: Le opzioni per il Server PostgreSQL

Completati questi step, avremo, nel Menu Programmi, le voci in figura.

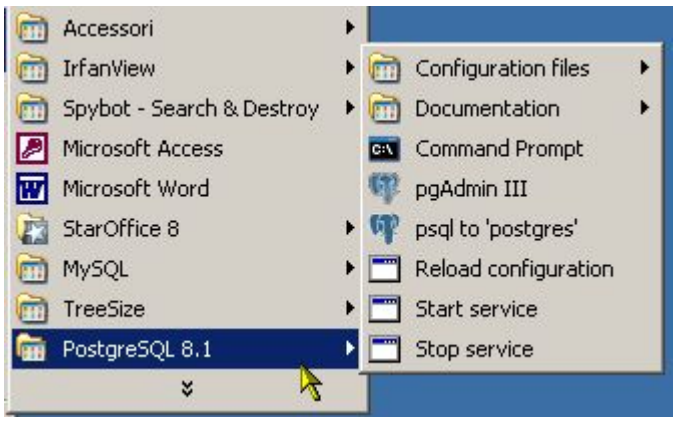


Figura 6.1.4: Il Menu di PostgreSQL

Le ultime tre voci del Menu servono a gestire il Servizio (cioè l'avvio e lo stop del server) e le modifiche alla configurazione (come vedremo tra poco).

TIP

Vi ricordo che l'avvio automatico o meno di un servizio in Windows è definibile nel *Pannello di controllo -> Strumenti di Amministrazione -> Servizi*; una volta individuato il Servizio (nel nostro caso PostgreSQL Database Server 8.1), un doppio click apre la

scheda delle proprietà, dove alla voce "Tipo di avvio" si può scegliere tra "Manuale" ed "Automatico".

PgAdmin III è invece l'interfaccia grafica di configurazione, invero assai ben fatta. Ma prima di partire è necessario ancora mettere mano ai ...

6.1.2 File di configurazione

Postgres, da buon prodotto proveniente dal mondo Unix, utilizza dei semplici file di testo per la propria configurazione. Un'ottima idea è stata quella di aggiungere delle semplici chiamate al menu per la modifica di questi file, come in figura:

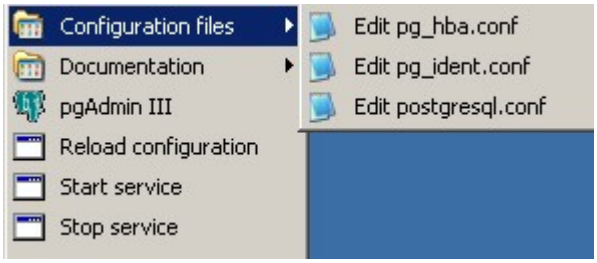


Figura 6.1.5: file di configurazione in Windows

Non è questa ovviamente la sede per esaminare in dettaglio le opzioni previste da *Postgres*, piuttosto ci interessa che il Server appena installato faccia due cose semplici : ascolti le richieste su tutte le interfacce di rete, e permetta l'accesso Client anche dagli altri Computer della Lan. Per la prima opzione, dobbiamo assicurarci che nel file **postgres.conf** siano presenti le seguenti righe:

```
....
listen_addresses = '*'  # what IP interface(s) to listen on;
                        # defaults to localhost, '*' = any
port = 5432
........
```

in pratica si richiede al server di *ascoltare* le richieste su qualsiasi interfaccia IP presente sul sistema utilizzando la porta *5432*. Questo ovviamente può non essere saggio in alcune configurazioni (potremmo avere una interfaccia WAN, su cui *ascoltare* è inutile e pericoloso...), quindi regolatevi secondo le vostre esigenze. La seconda opzione prevede che nel file **pg_hba.conf** siano presenti le righe:

```
......
# TYPE      DATABASE    USER  CIDR-ADDRESS      METHOD
# IPv4 local connections:
host        all         all   127.0.0.1/32      md5
host        all         all   192.168.1.1/24    md5
.......
```

In particolare la seconda riga *host* indica che i PC con indirizzo nel segmento di rete 192.168.1.1/24 possono accedere con le credenziali di qualunque user a qualunque database. Notate l'estrema semplicità con cui è possibile limitare l'accesso al server con un editor di testo. Anche in questo caso l'effettiva configurazione dipende dalla vostra LAN e dal compito assegnato al vostro Server.

Tecnica

Se sul PC che ospita il Server Postgres è installato un Firewall, controllate che la porta 5432 non sia bloccata. Questo è il caso, per esempio di Windows XP con Service Pack 2. Se la vostra interfaccia di rete serve solo al collegamento LAN locale, può essere una buona idea disabilitare questo tipo di Firewall (la cui efficacia è comunque tutta da dimostrare...)

6.1.3 Il Driver ODBC

Il Driver ODBC può essere installato assieme al Server oppure scaricando dal sito di PostgreSQL il file *psqlodbc-08_01_0101.zip*. La configurazione del DSN non comporta grosse difficoltà: i Driver disponibili sono due, quello *ANSI* e quello *Unicode*, da scegliere in base alle proprie esigenze. Questa è la finestra dei parametri di connessione:

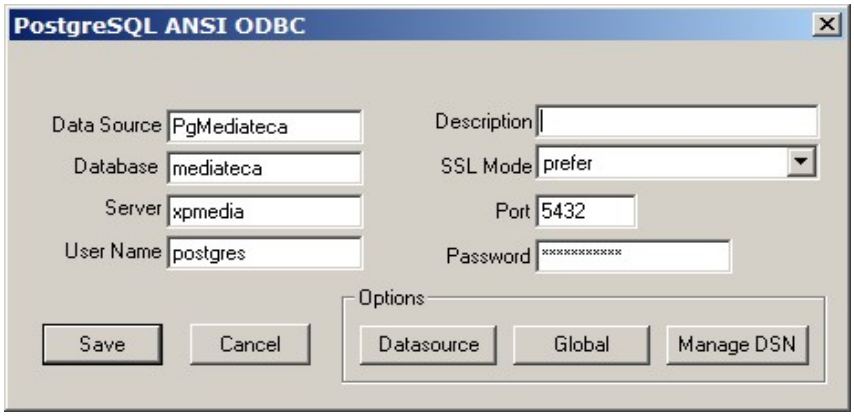


Figura 6.1.6: DSN per Postgres

6.1.4 Il Driver JDBC

Anche in questo caso i Driver JDBC sono inclusi nel Server, e possono essere recuperati nella cartella *c:\programmi\PostgreSQL\8.1.\jdbc*. Versioni aggiornate possono essere scaricate da http://jdbc.postgresql.org/download.html. Notate che sono disponibili diverse build, a seconda della versione di JDK utilizzata. Il file *.jar* può poi essere copiato in una posizione più comoda di vostra scelta.

6.1.5 I programmi Client

Nel Server sono inclusi sia ***psql***, cioè il Client a linea di comando, sia l'interfaccia grafica di amministrazione ***pgAdmin III*** (nella versione 1.4.0). Versioni aggiornate di *pgAdmin* possono essere scaricate da http://www.pgadmin.org/.

6.2 Installazione in Linux

Per Linux, in questo momento, sono disponibili versioni binarie aggiornate (in formato RPM) solo per *Fedora* e *Red Hat*. In alternativa è possibile ovviamente scaricare e compilare il Sorgente (niente di eccessivamente complicato ma bisogna avere una certa confidenza con Linux). Comunque quasi tutte le maggiori distribuzioni hanno il pacchetto disponibile nei CD di installazione, anche se potrebbe trattarsi di versioni non proprio recentissime.

Il discorso è identico per *pgAdmin III* e per i *Driver ODBC*: la soluzione migliore è il download e la compilazione, ma mi rendo conto che questi passaggi non sono alla portata di tutti. Quanto detto sui **file di configurazione** vale comunque anche per gli altri sistemi operativi supportati (oltre Linux e Windows, davvero un bell'elenco...), quindi nessuna ulteriore difficoltà. Ad esempio una installazione in Mandrake 10.1 da rpm posiziona i file in ***/var/lib/pgsql/data***.

6.3 Tipi di Dati

Ogni Server di Database possiede un lungo elenco di tipologie di Dati gestibili. In realtà i tipi più comuni (numeri, testo e date) sono molto simili in tutti i *motori* SQL. Credo però sia opportuno indicare con precisione, per ogni Server, la denominazione esatta della tipologia e le caratteristiche dell'informazione che può essere archiviata. *Postgres* è uno dei Server di Database con la maggiore varietà di tipologie di dati gestibili. Oltre alle definizioni più o meno standard, come quelle che seguono, sono disponibili ad esempio tipi *Geometrici* per i Database Gis, *Indirizzi di Rete* di vario tipo oltre ad un completo set di Arrays. Avete quindi spazio per approfondimenti interessanti....

6.3.1 Campi di tipo Stringa

Postgres implementa le classiche tre tipologie, ***char***, ***varchar*** e ***text***. Nella definizione di char(n) e varchar(n), il numero di caratteri disponibili non è limitato a 256 come in altri prodotti ma a circa 1 Gb. Il tipo text, invece, è molto generico e permette la gestione di stringhe di lunghezza non definita a priori.

6.3.2 Campi di tipo numerico

Il Tipo di dati gestibile può essere riassunto nella tabella seguente:

Nome	Spazio	Descrizione	Intervallo di valori
smallint	2 byte	Intero piccolo	-32.768 <=> +32.767
int	4 byte	Intero	-2.147.483.648 <=> +2.147.483.647
bigint	8 byte	Intero Lungo	-9223372036854775808 <=> 9223372036854775807
decimal	Variabile	Precisione definibile, calcoli esatti	Non limitato
real	4 byte	Precisione variabile, calcoli inesatti	Precisione massima di 6 decimali
double	8 byte	Precisione variabile, calcoli inesatti	Precisione massima di 15 decimali

6.3.3 Campi di tipo Data/Ora

Postgres usa ***date***, e ***time*** oltre ad un tipo ***interval*** che serve ad archiviare gli intervalli di date. Se è necessario manipolare campi che contengono insieme sia date che orari, si può usare il tipo ***timestamp***.

6.3.4 Campi di tipo booleano

Definito come ***boolean***, può assumere i valori di *True*, *False* ma anche *Unknown* rappresentato dal valore *null*. Notate che non si tratta di un campo numerico ma carattere. Così un valore predefinito non può essere 0 (zero) ma '0' o 'f' (con gli apici).

6.3.5 Campi di tipo binario

Un campo binario in Postgres viene definito come ***bytea***, cioè una sequenza binaria di byte.

6.3.6 Campi particolari: Intero ad incremento automatico

In Postgres esiste una definizione particolare per questo tipo di campo, riassunta nella tabella:

Nome	Spazio	Descrizione	Intervallo di valori
Serial	4 byte	Intero incremento aut	1 <=> 2.147.483.647
Bigserial	8 byte	Intero incremento aut	1 <=> 9.223.372.036.854.775.807

6.3.7 Campi particolari : Timestamp

Il ***Timestamp*** di Postgres NON viene aggiornato automaticamente dal motore di Db, e può invece essere usato per l'archiviazione di valori Data/Orario.

6.4 pgAdmin III

pgAdmin è una ottima interfaccia grafica per l'amministrazione di Server Postgres, disponibile per le principali piattaforme. La versione Windows del Server permette di avere immediatamente disponibile il software, ma è possibile scaricare da Internet eventuali versioni più recenti. All'avvio il programma presenta un'interfaccia pulita e razionale. Lo schermo è diviso in tre sezioni: la parte sinistra mostra una struttura ad albero contenente tutti gli elementi del Server (o dei server) selezionati; la finestra superiore della parte destra contiene tutte le informazioni relative all'elemento selezionato; la finestra inferiore contiene il "reverse engineering" in linguaggio SQL dell'elemento stesso (questo sarà più chiaro tra un attimo).

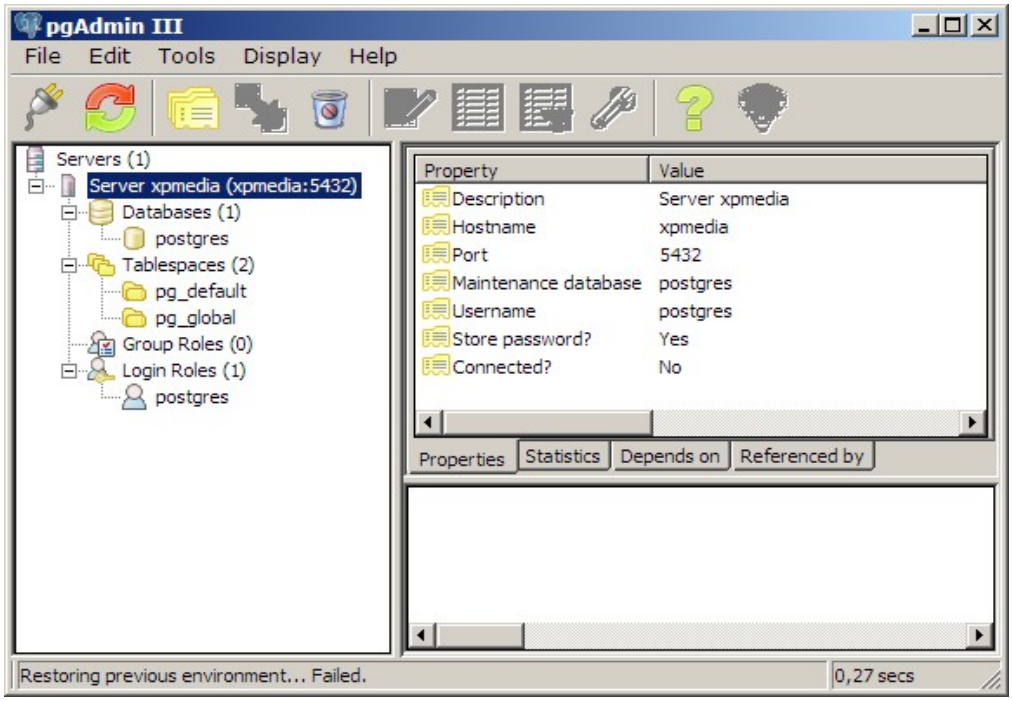


Figura 6.4.1: pgAdmin III Versione 1.4.0

La prima cosa da fare, se non si è già fatto, è configurare una connessione con un Server: si sceglie dal Menu File la voce "Add Server" e si compila la form seguente:

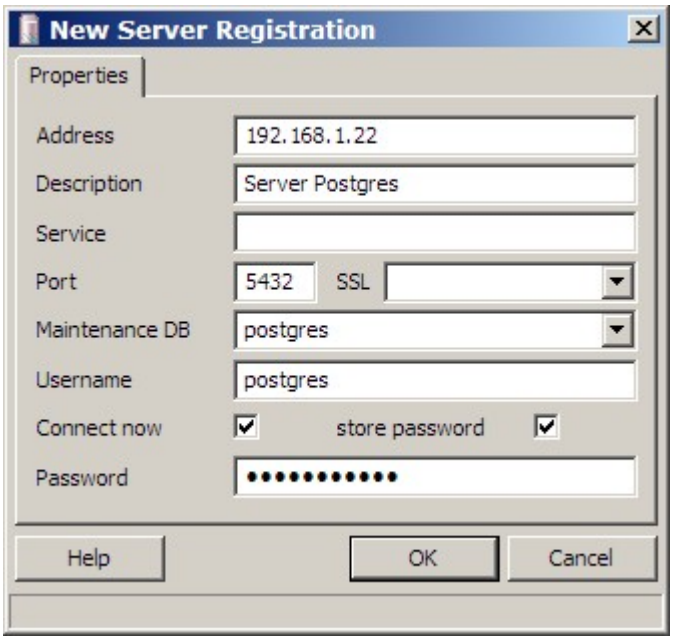


Figura 6.4.2: Registrazione di un nuovo Server

L'*indirizzo* è l'IP del Server, ma si può usare *localhost* se si opera in locale; la *Descrizione* identifica il Server; la *porta* è quella definita per l'ascolto sul Server (default 5432); se la connessione è criptata, si può scegliere la tipologia *SSL*; il *servizio* può rimanere in bianco; il *Db iniziale* è quello che desideriamo gestire, ma può andare bene anche *postgres*; il *nome utente* è quello con cui desideriamo connetterci (*postgres* se vogliamo amministrare il Server); se vogliamo che *pgAdmin* conservi la password di connessione, in modo da non doverla reinserire ad ogni collegamento, selezionate *store password*.

Una volta collegati al Server, possiamo espandere l'albero sulla sinistra per elencare tutti gli elemento disponibili (che sono davvero tanti...).

La figura può dare un'idea delle possibilità del prodotto e di perché sia considerato adatto anche ad applicazioni complesse. Notate come nella finestra in basso a destra compaia la "traduzione" SQL dell'elemento selezionato: questo è molto "didattico" oltre che molto utile, come vedremo in seguito.

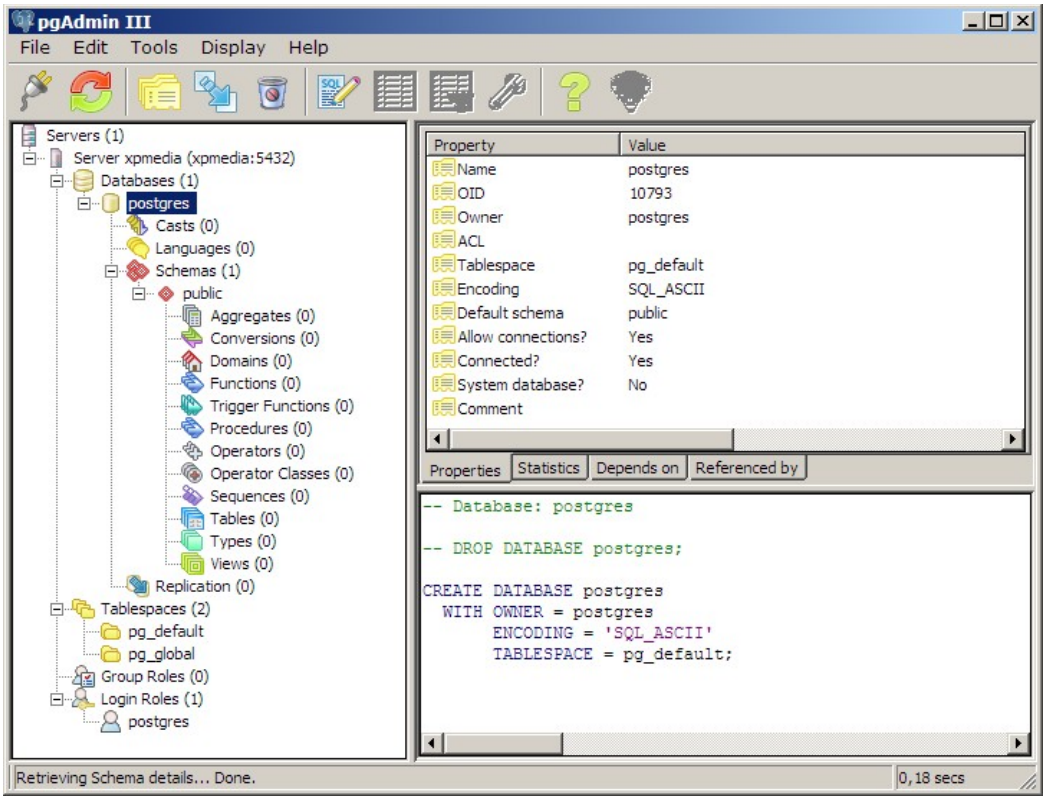


Figura 6.4.3: pgAdmin III

6.4.1 pgAdmin: Creazione del Database di esempio

L'aggiunta di un nuovo Database al Server è una operazione semplice: dopo aver selezionato la voce ***Databases*** sulla sinistra, col menu contestuale del tasto destro scegliamo *New Database*: per i nostri scopi basta immettere il nome (*Mediateca*) e lasciare tutti gli altri parametri così come sono.

6.5 Gestione delle Tabelle

6.5.1 pgAdmin: aggiunta delle Tabelle

Una volta creato il Database, l'aggiunta delle Tabelle è piuttosto banale. Si seleziona sulla sinistra l'elemento ***Tables*** e con il menu contestuale richiamato dal tasto destro del mouse si sceglie ***New Table***. Nella prima Tab possiamo limitarci ad immettere il nome della Tabella.

TIP

Con Postgres è possibile creare Tabelle con ***OIDs***, come opzione. *L'OIDs* è un campo a gestione automatica utilizzato dalle Versioni precedenti del Server. Sebbene ne sia sconsigliato l'uso, i driver *ODBC* funzionano male con i campi di tipo Serial se la tabella

non possiede l'OIDs. *In linea di massima consiglio di aggiungere sempre questa caratteristica alle nuove tabelle.*

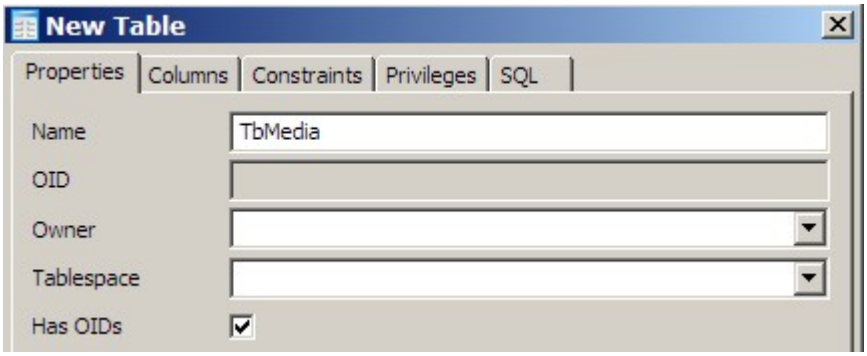


Figura 6.5.1: Creazione di una nuova Tabella

Selezionando la Tab *Columns* avremo appunto la possibilità di aggiungere i Campi della Tabella; in basso si trova infatti il pulsante *Add* che apre una form come questa:

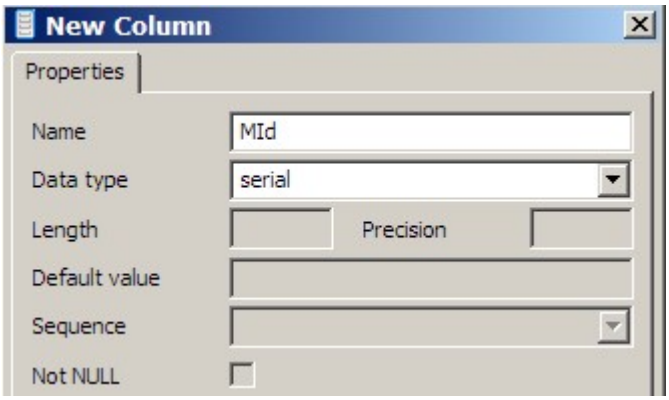


Figura 6.5.2: Aggiunta di una colonna

A seconda del tipo di campo selezionato sarà possibile immettere i parametri relativi; in questo caso ***MId*** è un contatore, quindi abbiamo scelto il tipo ***serial***. Alla fine del lavoro, per *TbMedia* dovremmo avere più o meno questa situazione:

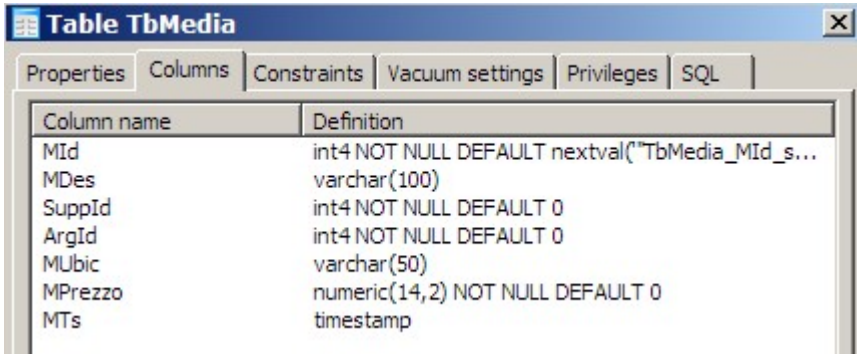


Figura 6.5.3: Struttura della Tabella

Notate che:

- pur avendo definito ***MId*** come ***serial***, Postgres "traduce" la tipologia in un linguaggio SQL più canonico (intero con riferimento ad un contatore), ma il risultato non cambia;
- Postgres usa anche degli alias nelle definizioni (così, tanto per confondere un po' le idee...), per cui ad esempio ***int4*** sta per ***integer*** e ***int2*** vuol dire ***smallint***; comunque la cosa è sufficientemente intuitiva;
- una volta definito, un campo non può cambiare tipologia (ad esempio da char a integer); è necessario cancellare ed aggiungere una nuova definizione, con conseguente perdita del dati
- non è qui che si può impostare la *chiave primaria*....

6.5.2 pgAdmin: impostazione della Chiave Primaria

La *Chiave Primaria* (come anche una Chiave Esterna) è considerata da Postgres un *Constraints* (*Vincolo*), quindi abbiamo una Tab apposita (appunto *Constraints*). In basso è possibile selezionare il tipo di vincolo da applicare con la pressione del pulsante *Add*:

Figura 6.5.4: Aggiunta di una Chiave Primaria

Otterremo una form dove per prima cosa è necessario assegnare un nome al vincolo; ATTENZIONE: non possono esistere, all'interno dello stesso Db, vincoli con lo stesso nome quindi sceglietelo con cura:

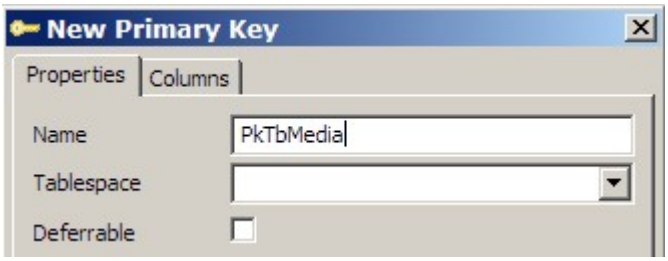


Figura 6.5.5: Nome del Vincolo

A questo punto dovremo definire la colonna (o le colonne) che saranno la nostra chiave primaria: nella tab *Columns* si seleziona in basso il nome del campo e si preme il pulsante *Add*.

Figura 6.5.6: MId come Chiave Primaria

Dovremmo ottenere più o meno questo:

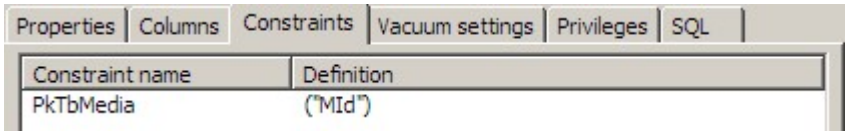


Figura 6.5.7: Chiave Primaria definita

Da quanto esposto dovrebbe essere semplice aggiungere le altre Tabelle della Mediateca.

6.5.3 PgAdmin: definizione degli Indici

Se si espande sulla sinistra un "rametto" relativo ad una tabella possiamo farci un'idea di quanti elementi siano ad essa associati:

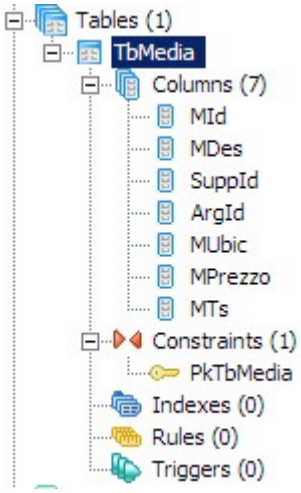


Figura 6.5.8: Elementi di una tabella

Tra le voci (alcune davvero interessanti ma ovviamente non trattabili in questa sede) troviamo anche gli *Indici (Indexes)*. Col solito metodo del menu contestuale, scegliamo *New Index*, e dopo aver assegnato il nome (anche questo univoco per il Database) possiamo selezionare la colonna (o le colonne) che ne faranno parte:

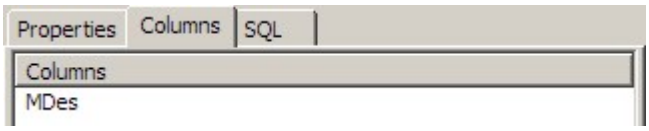


Figura 6.5.9: Indice sulla Descrizione

TIP

In ognuna di queste form, la tab chiamata "SQL" esplicita in Sql quello che stiamo facendo. Anche questo è molto didattico. Per l'azione precedente, ad esempio, avremo:

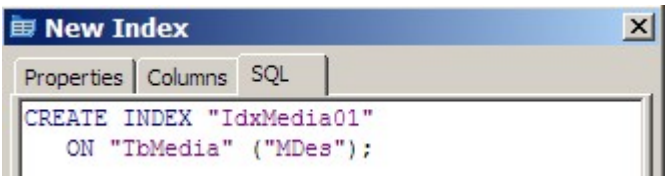


Figura 6.5.10: La traduzione in SQL

6.5.4 pgAdmin: Integrità Referenziale

Nei Capitoli precedenti abbiamo già parlato dell'*integrità referenziale*, quindi il concetto dovrebbe essere già abbastanza chiaro. Si tratta in pratica di stabilire che un determinato campo di una Tabella (ad esempio ***ArgId*** di ***TbMedia***) fa riferimento (anzi DEVE fare riferimento) ad un campo simile di un'altra Tabella (***ArgId*** di ***TbArgomenti***) in modo che il "legame" sia solido e continuo. Il motore di Db deve cioè controllare che il nuovo Media inserito faccia riferimento ad un Argomento già presente nella Tabella degli Argomenti. Inoltre non deve essere possibile cancellare da *TbArgomenti* un valore già utilizzato in *TbMedia*.

Questo, per Postgres, è un *Vincolo* della Tabella (come la *Chiave Primaria*), quindi va appunto immesso come "*caratteristica*" della Tabella stessa. Richiamata col solito metodo la Form relativa alle proprietà della Tabella; alla Tab ***Constraints*** in basso è possibile aggiungere una *Chiave Esterna*:

Figura 6.5.11: Chiave Esterna

Nella Form di aggiunta bisogna impostare come prima cosa il nome, che come al solito deve essere univoco all'interno del Database; quindi è necessario indicare la Tabella che contiene la Chiave Esterna:

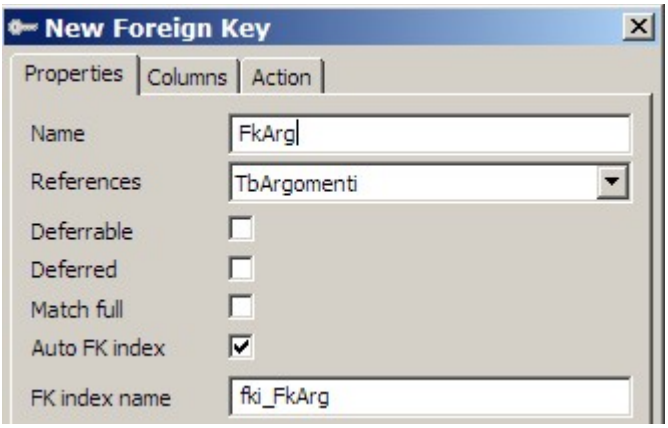


Figura 6.5.12: Parametri per la chiave esterna

Ancora: in *Local* va immesso il nome del Campo della Tabella a cui stiamo aggiungendo il vincolo (*TbMedia*), in *Referenced* il nome del Campo della Tabella contenente la Chiave Esterna (*TbArgomenti*):

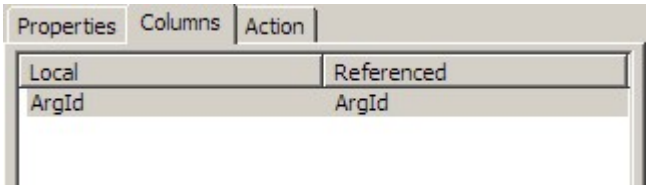


Figura 6.5.13: Definizione della chiave

Infine nella Tab *Action* bisogna definire il "comportamento" dell'integrità:

Figura 6.5.14: Regole dell'integrità

In questa Tab, si possono notare due opzioni, precisamente ***Su Aggiornamento*** (**ON UPDATE**) e ***Su Cancellazione*** (**ON DELETE**), entrambe settate su ***RESTRICT***. Bene, se quello che abbiamo appena visto è il comportamento standard del motore di Db (*RESTRICT*), possiamo comunque eventualmente scegliere delle opzioni alternative ne caso si modifichi (*ON UPDATE*) o si cancelli (*ON DELETE*) la chiave esterna (cioè il genitore di una serie di record presenti nella Tabella "figlia"). In particolare :

CASCADE estende la variazione della chiave esterna alla Tabella "figlia"; nel caso di *ON UPDATE CASCADE*, se, ad esempio, in *TbArgomenti* modifichiamo *ArgId* da 5 a 55, tutti i record di *TbMedia* con *ArgId* uguale a 5 saranno aggiornati di conseguenza. Nel caso di *ON DELETE CASCADE*, però, la cancellazione di una chiave esterna NON VIENE PIU' IMPEDITA, e vengono eliminati a cascata anche tutti i record della Tabella "figlia". *Cioè se cancello "5" in TbArgomenti, non avrò più nessun Giallo in TbMedia.*

SET NULL invece si limita ad impostare a Null tutte le chiavi figlie del genitore modificato.

SET DEFAULT infine, imposta le chiavi figlie al valore di DEFAULT eventualmente specificato.

6.6 Backup degli Archivi Postgres

I database Postgres risiedono fisicamente in una Directory dell'Hard Disk chiamata *data.* Postgres definisce questa entità un ***Database Cluster***. A seconda del Sistema Operativo, ed anche delle scelte dell'utente, questa cartella può trovarsi in percorsi diversi. In Windows, l'installazione standard posiziona la dir nel percorso c:\Programmi\PostgreSQL\8.1\data. In Linux è in effetti l'utente in fase di installazione che sceglie il percorso, col comando initdb. Spesso la scelta cade su `/usr/local/pgsql/data` oppure su `/var/lib/pgsql/data` .

Fatta questa doverosa premessa, Postgres suggerisce tre metodi principali per eseguire un Backup degli archivi:

- l'uso del programma di servizio ***pg_dump*** e del quasi equivalente ***pg_dumpall***; queste utility creano una copia del DB in formato testo, che a sua volta può essere compresso ad esempio in formato Zip; questo metodo è in effetti quello consigliato, perché può essere usato senza arrestare il Server
- la copia fisica, con qualunque metodo, della cartella data; questo implica però l'arresto del Server e permette la copia di tutti i Database presenti
- quello che Postgres chiama ***On-line backup and point-in-time recovery (PITR)***, basato sulla presenza di un ***write ahead log (WAL)*** cioè di un Log delle transazioni; è un metodo interessante, ma direi estraneo allo scopo di questo documento

6.6.1 PgAdmin: Backup e Restore degli Archivi

PgAdmin può essere usato come interfaccia grafica verso *pgdump*. Selezionando sulla sinistra il Database, od anche una singola Tabella, dal menu contestuale (tasto destro del Mouse) è possibile scegliere la voce *Backup*, che apre una form simile:

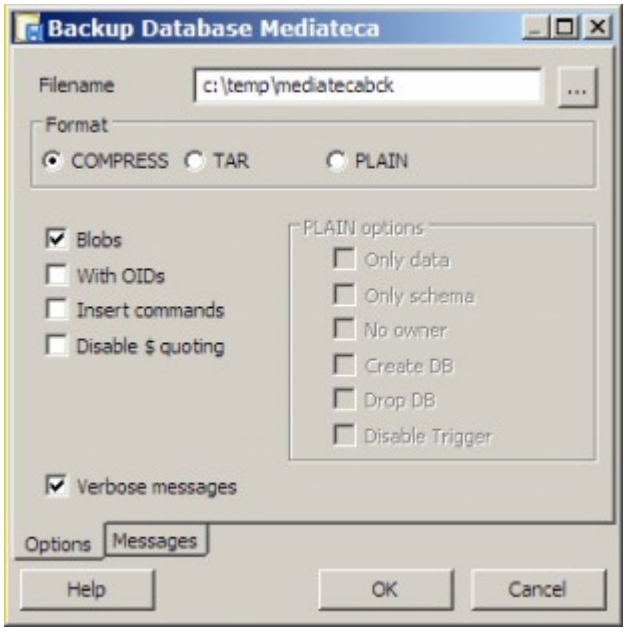


Figura 6.6.1: Backup con pgAdmin

Le opzioni sul formato riguardano:

- ***compress***, un formato proprietario compresso che può essere gestito solo con *Postgres*, *pgAdmin* ed i programmi a linea di comando; efficiente e flessibile
- ***tar***, formato simile al precedente, ma compresso attraverso programmi standard
- ***plain***, cioè il dump del Database in Testo "liscio" (***plain***), utile se si desidera manipolare i dati con altri programmi ma sconsigliato perché occupa davvero molto spazio

Analogamente è possibile richiamare la form per il restore, che presenta alcune intuitive opzioni, come in figura:

Figura 6.6.2: restore con pgAdmin

6.7 La sicurezza: utenti e diritti

Postgres presenta una gestione della sicurezza abbastanza "classica". A differenza di *MySql,* è possibile definire *Gruppi* di Utenti ed associare privilegi al Gruppo piuttosto che al singolo utente. Ad un *Utente* oppure ad un *Gruppo* possono essere assegnati, come vedremo, diritti abbastanza "granulari", cioè specifici, sugli oggetti gestiti dai motore di Db.

6.7.1 pgAdmin : creazione di un nuovo utente

Le cose, nella versione 8.1 ed in pgAdmin 1.4, sono un po' cambiate. L'istruzione classica Sql **CREATE USER** è diventata sinonimo di **CREATE ROLE** (crea ruolo). In sostanza, per aggiungere un nuovo utente, è necessario scegliere la voce ***Login Roles*** e, col tasto destro, ***New Login Role***. La form è quella in figura:

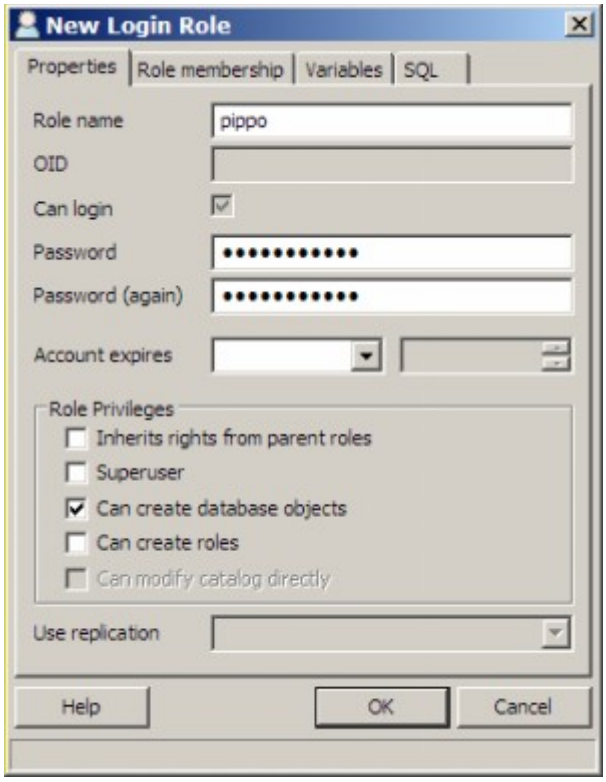


Figura 6.7.1: Creazione di un nuovo utente

In questa fase è possibile assegnare anche alcuni privilegi globali alla nuova identità. A differenza di *MySql*, l'utente può effettuare il login da qualsiasi macchina abilitata sulla rete.

6.7.2 pgAdmin: creazione di un nuovo Gruppo

I Gruppi di utenti possono essere creati alla voce ***Group Roles***, al solito con il tasto destro selezionando ***New Group Role***:

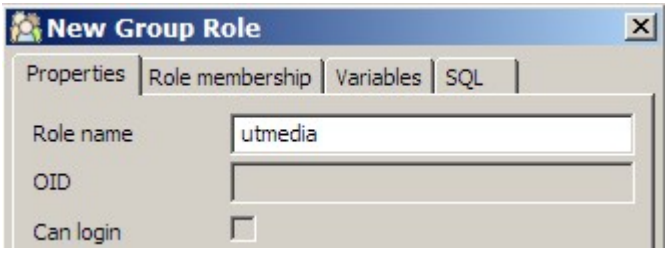


Figura 6.7.2: Creazione di un Gruppo

la Tab ***Role Membership*** delle proprietà del *Login Role* (Utente) permette di aggregare un Utente al Gruppo:

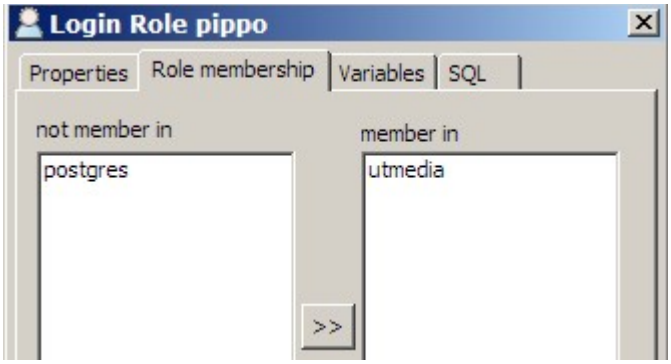


Figura 6.7.3: Assegnazione dell'Utente al Gruppo

6.7.3 pgAdmin: assegnazione dei diritti

Anche la fase di assegnazione dei diritti è abbastanza semplice. Ovviamente il tipo di diritto cambia a seconda dell'oggetto che vogliamo "proteggere". Ad esempio, nella Tab ***Privileges*** delle proprietà di un Database è possibile solo assegnare i diritti: *All*, *Create*, *Temp* (cioè *tutto*, *creare nuovi oggetti*, *creare tabelle temporanee*), mentre a livello di singola tabella le cose si fanno più complesse:

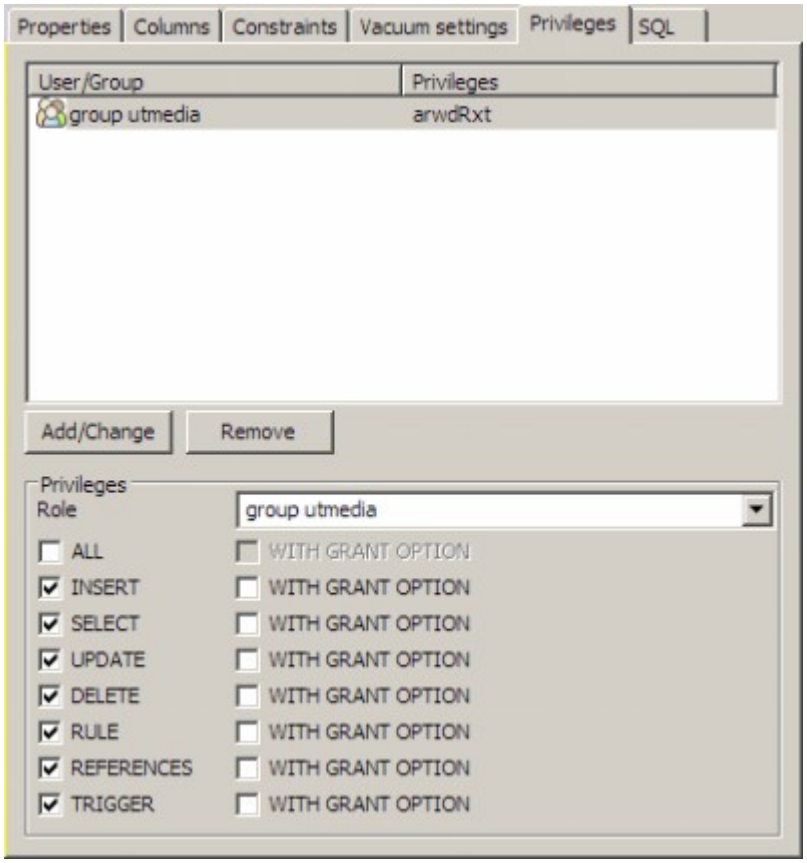


Figura 6.7.4: Diritti Utente su una Tabella

pgAdmin permette l'assegnazione di diritti ***solo ai Gruppi*** , a meno che non sia selezionata la voce ***Show Users for Privileges*** nella Tab ***Preferences*** del Menu ***File -> Options***. Vi consiglio, per approfondire, la lettura dell'ottima documentazione a corredo del prodotto.

TIP

La versione 1.4.0 di pgAdmin ha qualche problema nell'assegnazione dei Diritti Utente. Innanzi tutto il voler indicare ogni cosa con la parola ***ROLE*** crea confusione. Per essere concreti, ***LOGIN ROLES*** sono i classici *Utenti*, ***GROUP ROLES*** sono i *Gruppi*. Quindi, di solito, si aggiunge un Utente ad un Gruppo, e non il contrario, ricordatelo. La procedura giusta perciò è: ***a.*** creare un *Gruppo*; ***b.*** creare gli *Utenti*; ***c.*** assegnare gli Utenti al Gruppo; ***d.*** stabilire i *Privilegi* per il Gruppo. Nella Tab di assegnazione diritti, poi, *pgAdmin non permette di selezionare i Gruppi con la casella a discesa*: prima di impazzire, provate a scrivere manualmente *group* ed il nome del gruppo. Infine usate, per gli Utenti ed i Gruppi, solo *nomi con lettere minuscole*, altrimenti potreste avere strani malfunzionamenti.

6.8 Importazione di Dati da altri Db

PostgreSQL non dispone di uno strumento dedicato al trasferimento dei dati da altri Db. Il sistema più semplice (!?) è creare un file di testo CVS (Comma Separated Value) ed importarlo con il comando SQL COPY FROM.

6.9 Caratteristiche avanzate

PostgreSQL è un Db molto potente, con caratteristiche simili a prodotti commerciali di alto lignaggio. Non è ovviamente questa la sede adatta a descrivere in dettaglio queste feature: cito soltanto i Tipi definiti dall'Utente, le funzioni di aggregazione personalizzabili, i Trigger, le Stored Procedure ed, ovviamente, le Viste (View).

6.9.1 Creazione di Viste (Views)

Come abbiamo già avuto modo di dire, una vista è un'istruzione SQL molto simile ad una **SELECT**. Nel pannello destro selezioniamo ***Views***, quindi col tasto destro ***New View***, ed avremo:

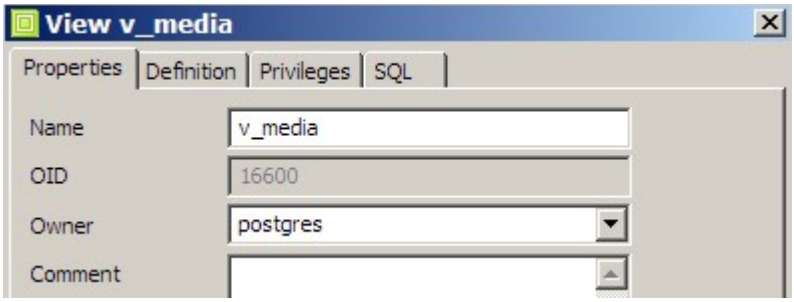


Figura 6.9.1: Creazione di una Vista

Dobbiamo assegnare un nome, e, nella Tab Definition, l'istruzione SQL che definisce la vista. Nel nostro caso la classica:

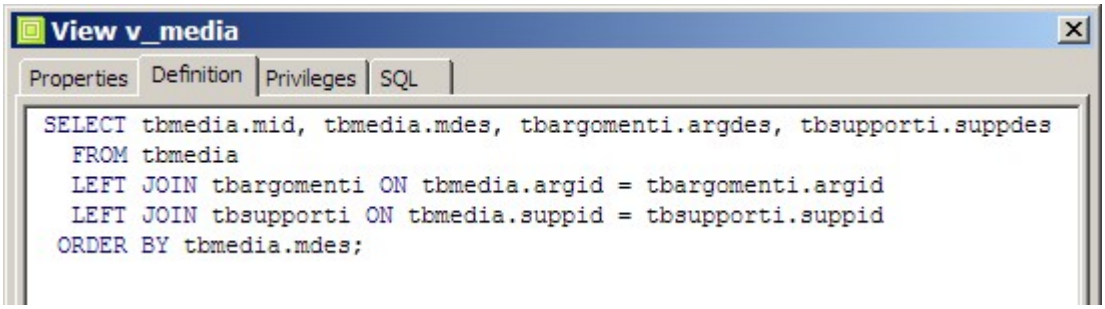


Figura 6.9.2: Istruzione SQL per la vista

Attenzione: *le Viste, in PostgreSql, sono sempre di sola lettura: non è possibile, perciò modificare le righe.*

7. Database Server Microsoft SQL 2005 EE

In questi ultimi mesi del 2005 due protagonisti del mercato dei Db, *Microsoft* ed *Oracle*, hanno deciso di rilasciare versioni gratuite (il che non significa certo "*libere*" nel senso di Open Source) dei loro prodotti di punta. In verità Microsoft permette l'uso gratuito anche della versione precedente di SQL Server, con un pacchetto chiamato *MSDE* (Microsoft Data Engine), però troppo limitato per un uso serio e con qualche problema riguardante la Licenza.

Questa volta, invece, la *Express Edition* ha solo limitazioni non molto vincolanti: la dimensione massima del Db (credo sia 4 GB), la memoria centrale usata (Max 1 Gb) ed il numero di Processori utilizzabili (solo 1). Quindi può essere un'alternativa concreta per chi è abituato ad ambienti Windows e desidera un'applicazione facilmente integrabile con gli altri prodotti dell'Azienda di Zio Bill.

7.1 Installazione

Microsoft permette il download diretto dal proprio sito Web sia del Server che di una serie di prodotti accessori che possono tornare utili. In particolare:

- Il ***Server***: *SQLEXPR.exe*, circa 55 Mb;
- il ***Software di Gestione***: *SQLServer2005_SSMSEE.msi*, SQL Server Management Studio Express , circa 31 Mb.
- La ***Documentazione***: *SqlServer2K5_BOL*, circa 116 Mb
- I ***Driver ODBC***: *sqlncli.msi*, circa 3,5 Mb
- I ***Driver JDBC***: *sqljdbc_1.0.419.102_enu.zip*, 609 Kb

Per installare il Server occorre lanciare il primo file, ed assicurarsi di avere un PC recente con almeno 512 Mb di RAM. L'installer è piuttosto semplice: l'unica scelta importante (comunque modificabile in qualsiasi momento) è utilizzare l'autenticazione integrata di Windows oppure quella classica con la gestione utenti separata dal Sistema Operativo. Alla fine, come al solito, avremo un Servizio di Windows, ad avvio Automatico o Manuale, che è il nostro Server di Db.

TIP

In realtà i Servizi sono due. Uno è il *Server*, l'altro si chiama *SQL Server Browser*: non ho ancora capito bene a cosa serve e più che altro crea problemi. Meglio, come vedremo, lasciarlo disabilitato.

Terminata questa prima fase, assieme al Server avremo a disposizione due piccole Utility: la prima si chiama SQL Server Configuration Manager, la seconda SQL Server Surface Area Configuration. Siccome, come spesso accade, i prodotti Microsoft sembrano progettati dal *Servizio Complicazione Cose Semplici* (*SCCS*), quelli di voi che credono di avere già tutto ben configurato e pronto all'uso dovranno ricredersi.

7.2 Configurazione di Rete

Appena installato, SQL Server Express Edition (nel seguito SSEE) accetta connessioni solo in locale e solo attraverso le Named Pipes. Quindi il primo passo è fare in modo che ci si possa connettere dalla Rete e col TCP/IP. In verità i passi sono due...

Per prima cosa bisogna utilizzare la *SQL Server Surface Area Configuration*. Si lancia il Tool, si sceglie *Surface Area Configuration for Service and Connection* ed, alla voce *Database Engine -> Remote Connections* si seleziona la voce mostrata in figura:

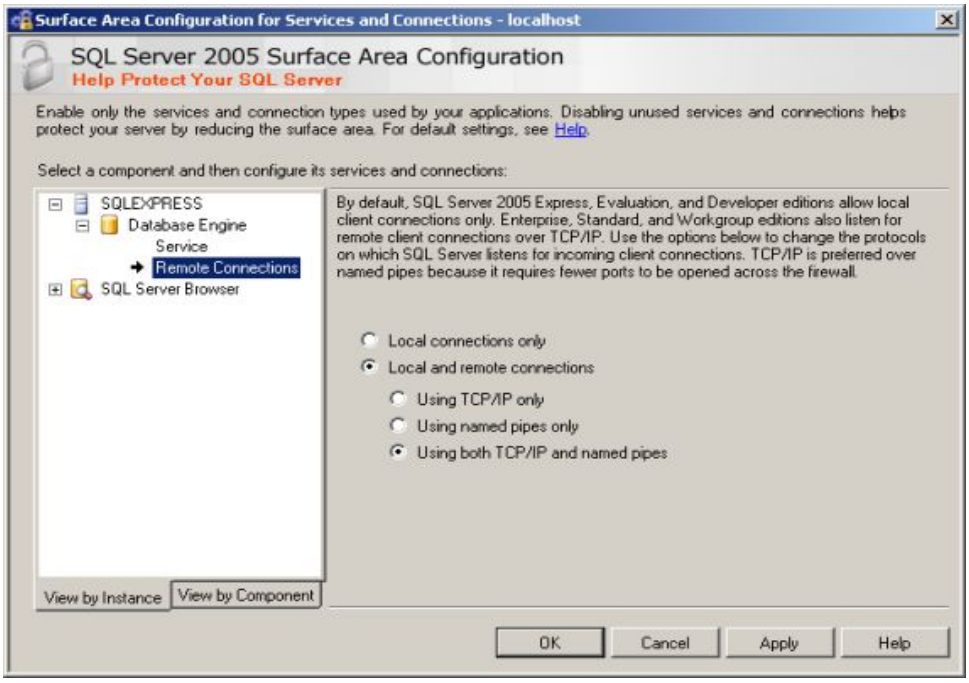


Figura 7.2.1: Protocollo di connessione per SQL Server

Secondo Microsoft questo dovrebbe essere sufficiente, ma io, dopo aver diligentemente eseguito questa operazione, lo stesso non sono riuscito a collegarmi da nessun Client. Dopo un po' di indagini, ho capito che era necessario un altro passaggio (che però non era scritto da nessuna parte...). Ecco quello che ho fatto.

- Ho richiamato il tool SQL *Server Configuration Manager*;
- ho scelto la voce *TCP/Ip* dai protocolli del Server, come in Figura;
- ho impostato le proprietà del protocollo *TCP/Ip*

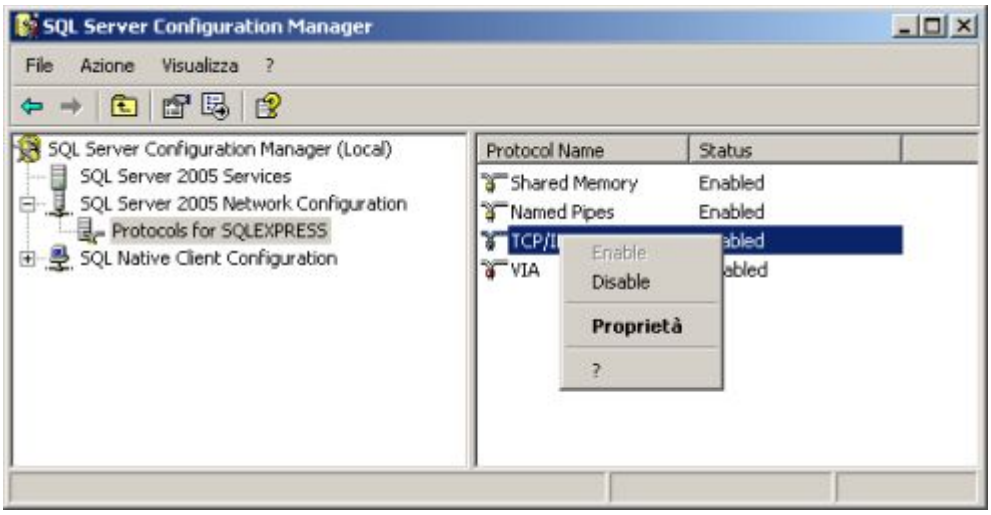


Figura 7.2.2: Impostazioni delle proprietà del Protocollo

In sostanza, secondo Microsoft, ***se*** viene abilitato anche il Servizio *SQL Server Browser*, dovrebbe funzionare una specie di assegnazione dinamica delle porte di collegamento: cioè dovrebbe essere il servizio stesso a comunicare al Client su quale porta del PC Server è abilitato l'ascolto (di solito la *1433*). A parte il fatto che questo servizio è disabilitato nell'installazione standard, anche attivandolo, per qualche motivo sulle mie macchine il collegamento non funziona, ed è necessario indicare manualmente la porta *1433*, come in figura:

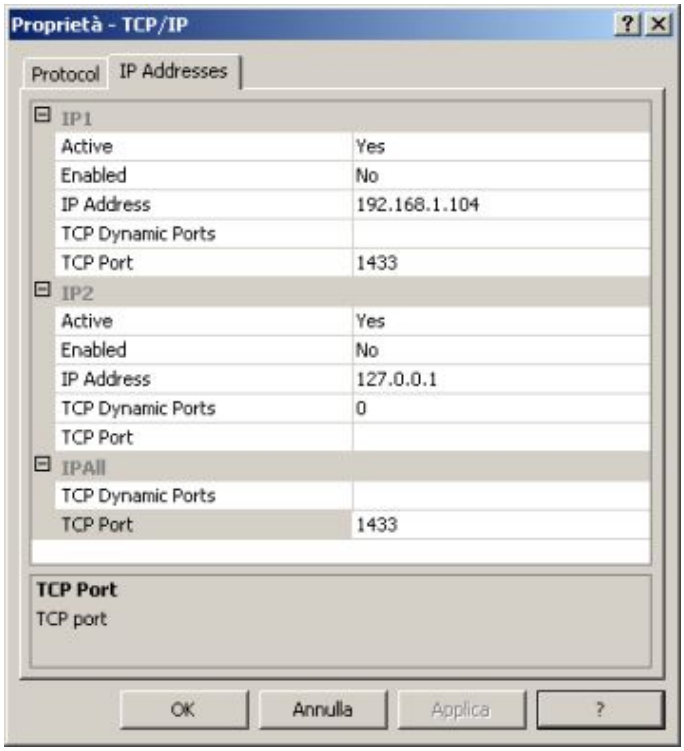


Figura 7.2.3: Configurazione delle porte

E' importante notare che i parametri devono essere impostati per ***Ip1***, cioè la scheda di rete del Server, e per ***IpAll*** (tutte le schede). Il campo ***TCP Dinamic Ports*** deve essere ***vuoto***, ed il campo ***TCP Port*** deve contenere il ***numero della porta scelta***. Dopo le modifiche è necessario riavviare il Server. Probabilmente a quelli di Microsoft tutta questa storia sarà sembrata il massimo della flessibilità: a me pare solo un'inutile complicazione. Comunque ora siamo pronti a partire.

7.3 SQL Server Management Studio Express

Il Tool di Gestione fornito per SSEE è indispensabile per la gestione del Db. Come struttura somiglia a *pgAdmin* di PostgreSQL, ed ha più o meno le stesse funzionalità. Per chi ha usato l'Enterprise Manager della versione precedente (peraltro non fornito con la versione gratuita MSDE), le cose sono cambiate, in verità non tutte in meglio.

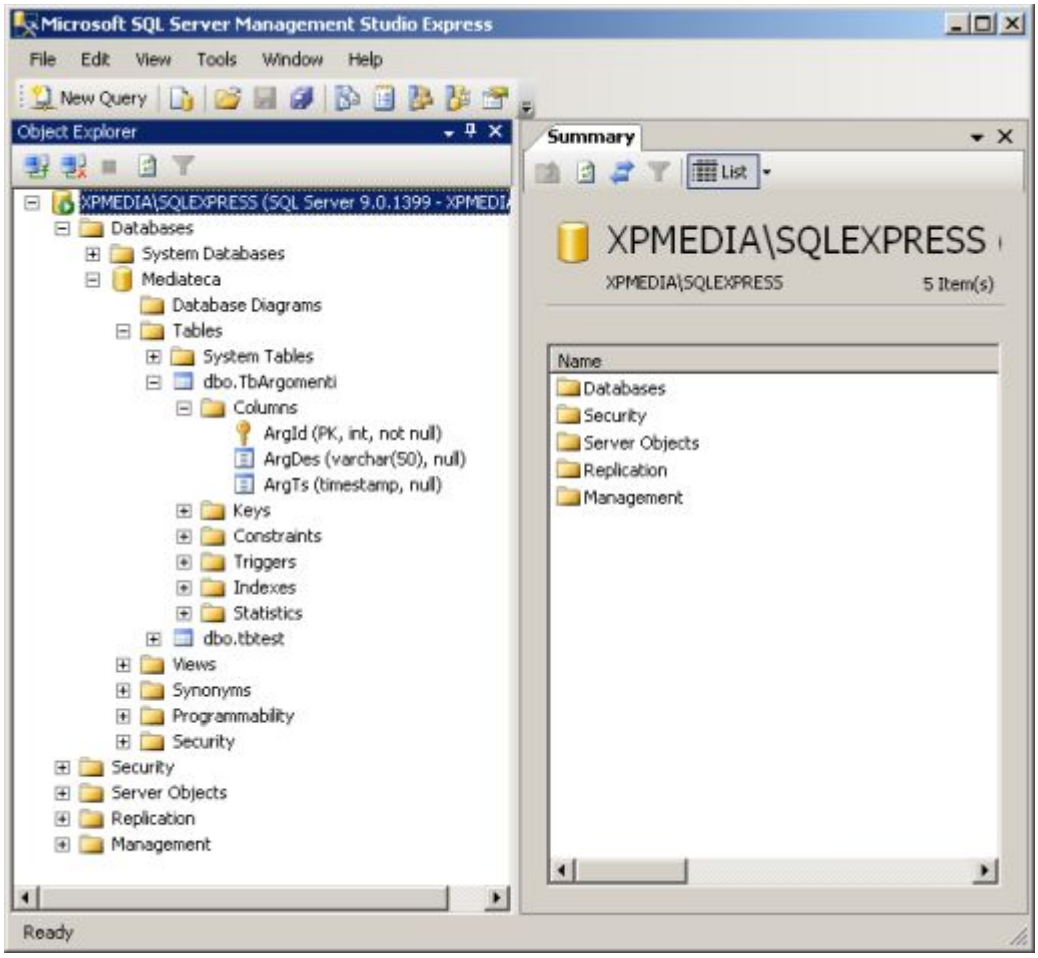


Figura 7.3.1: Studio Express

Non è questa la sede per illustrare il funzionamento del Programma: sappiate però che non sempre l'interfaccia è semplice e, soprattutto nella modifica della struttura delle Tabelle, ci vuole un po' di pratica per capire come procedere.

7.4 Tipi di Dati

7.4.1 Campi di Tipo Stringa

MS Sql implementa i classici ***char*** e ***varchar***; una "*n*" davanti al tipo di campo ("***nchar***", "***nvarchar***") significa che il campo stesso usa il set di caratteri internazionali predefinito .

Tipo di Dato	*Lunghezza*	*Definizione*
char / **nchar**	Da 1 a 8.000	char(X) dove X è il numero di caratteri
varchar / **nvarchar**	Da 0 a 8.000	varchar(X)

7.4.2 Campi di Tipo Numerico

I campi di tipo **numerico intero**, a seconda dell'intervallo di valori che possono contenere, si dividono in ***TinyInt***, ***SmallInt***, ***MediumInt***, ***Int***, ***BigInt***. Analogamente, i campi di tipo **numerico decimale** si possono definire ***Float***, ***Real***, ***Decimal***. Disponiamo inoltre di un tipo ***Money*** ed anche di ***SmallMoney***. Vi rimando alla documentazione ufficiale per notizie dettagliate sull'intervallo di valori ammessi.

7.4.3 Campi di Tipo Data / Ora

Abbiamo ***DateTime*** e ***SmallDateTime***, a seconda del valore da archiviare. Non esiste un tipo *Date* oppure un Tipo *Time*. Il valore deve contenere comunque una Data ed un Orario.

7.4.4 Campi di Tipo Booleano

Non esiste un campo *boolean*: al suo posto può essere usato un campo di tipo ***bit***: si noti che il valore *Vero* corrisponde a *1*, mentre, al solito, *Falso* a *zero*.

7.4.5 Campi di Tipo Binario / Testo

Ms Sql dispone del tipo ***Binary*** (campo binario a dimensione fissa) e ***Varbinary*** (a dimensione variabile). Inoltre abbiamo ***Text*** ed ***NText*** (campi di tipo testo di dimensione fino a 2 Gb) e ***Image*** (campo binario fino a 2 Gb, per l'archiviazione di immagini).

7.4.6 Campi particolari: Intero ad Incremento Automatico

Il campo deve essere definito come ***Int*** (o ***BigInt***) e deve essere abilitata la proprietà ***identity***; è possibile specificare il valore di partenza e l'incremento. Esiste un altro tipo di colonna, chiamata ***Uniqueidentifier***, che permette la memorizzazione automatica di un valore *certamente* univoco: se non usate la replica oppure il merge di database, risulta poco utile. L'assegnazione di un campo identità ad una Tabella non crea automaticamente la *chiave*

primaria che deve essere appositamente definita con la voce ***Table Designer -> Set Primary Key.***

7.4.7 Campi particolari: Timestamp

Il ***Timestamp*** di Ms SQL è di tipo "classico", cioè viene aggiornato automaticamente dal motore di Db.

7.5 Caratteristiche avanzate

SSEE è un Db di derivazione Enterprise e la versione FULL viene definita da Microsoft, con la solita umiltà, "*il miglior Database disponibile al mondo*". Questo, ovviamente, può non essere vero: però in ogni caso troviamo tutte le caratteristiche necessarie all'utente avanzato. Sono così disponibili: Integrità Referenziale, Viste, Trigger, Stored Procedure, Funzioni e Tipi definiti dall'Utente, etc. etc.

Non è questa la sede per approfondire questi argomenti: vi rimando alla abbondante documentazione disponibile direttamente da Microsoft.

7.6 Driver ODBC

L'installazione del Driver ODBC non comporta alcuna difficoltà. La creazione del DSN prevede la selezione del Driver ***SQL Native Client***, ed è una specie di procedura guidata che lascia poco spazio alle personalizzazioni; in pratica bisogna specificare: il nome del DSN, il nome o l'indirizzo IP del Server, il tipo di autenticazione, il Database predefinito. Un comodo pulsante di nome *Test Data Source* permette di controllare che tutto funziona.

8. Database Server Oracle 10g Express Edition

Alla fine del 2005 Oracle ha reso disponibile una versione ridotta del proprio Database Server 10g, chiamata *Express Edition*. Questo prodotto, pur non essendo Open Source, è liberamente scaricabile, utilizzabile e distribuibile anche con applicazioni commerciali. Le limitazioni, rispetto alla versione completa, ne fanno comunque una soluzione allettante anche per le Medie Imprese: fino a 4 Gb di spazio per i Db Utente, possibilità di usare una sola istanza ed una sola CPU per Server, memoria massima utilizzabile di 1Gb. Per il resto si tratta del classico ***10g*** confezionato per un'utenza non specializzata, e questo non è detto che sia uno svantaggio...

8.1 Installazione Windows

L'installer è scaricabile dal sito Oracle e pesa circa 150 Mb. L'installazione è semplicissima: l'unico parametro da specificare è la password dell'utente ***system*** (cioè l'amministratore del Database). Alla fine il Server è pronto all'uso: sono stati configurati una serie di servizi (alcuni ad avvio automatico) ed anche un mini server web alla porta 8080 per la gestione del Database. Il menu disponibile è quello in figura.

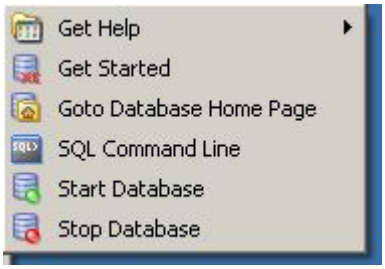


Figura 8.1.1: Menu di Oracle XE

La documentazione a corredo è ottima ed abbondante, integrata anche con molto materiale on line. In particolare vi segnalo le due Guide "***2 Day Dba***" e "***2 Day Developer***" che sono scritte bene ed illustrano in pochi capitoli tutte le informazioni essenziali alla gestione del Db.

8.2 Tipi di Dati

Oracle prevede un numero sorprendentemente basso di tipologie di Dati. Questo può essere disorientante per chi proviene da altri prodotti simili, ma è un vantaggio per chi non vuole eccessive complicazioni. Esistono poi una serie di tipi di campo abbastanza particolari su cui mi sembra inutile soffermarsi in questa sede.

8.2.1 Campi di Tipo Stringa

Il Server implementa i classici ***char*** (max 2000 byte) e ***varchar*** (che viene indicato come ***varchar2*** ed ha una capienza massima di 4000 byte); una "*n*" davanti al tipo di campo ("***nchar***", "***nvarchar2***") significa che il campo stesso usa il set di caratteri internazionali predefinito.

8.2.2 Campi di Tipo Numerico

Piuttosto che fare differenze tra interi e decimali, Oracle usa un solo tipo chiamato ***number***. A seconda di come si dichiara, può essere *intero* o *decimale*. La sintassi è la solita ***number(p,s)*** dove ***p*** rappresenta la ***precisione*** e ***s*** la ***scala***. Perciò, ad esempio, un numero intero di 9 cifre può essere definito come *number(9,0)* oppure *number(9),* mentre la classica valuta come *number(14,2)*. Se si desidera utilizzare numeri in virgola mobile con precisione binaria, sono disponibili i tipi ***binary_float*** e ***binary_double***.

8.2.3 Campi di tipo Data / Ora

Esiste un solo Tipo, chiamato ***date***, che può archiviare un ampio intervallo di valori. Se non si specifica per esteso l'anno (con quattro cifre), viene assunto appartenere al secolo corrente. Per cui, ad esempio, 31/12/92 per Oracle è il 31/12/2092 e non il 31/12/1992.

8.2.4 Campi di Tipo Booleano

Non esiste in Oracle una definizione esplicita per il tipo Boolean. Può essere utilizzato il tipo number(1) assegnando a falso il valore 0; è comunque possibile definire tipi di dati addizionali.

8.2.5 Campi di tipo binario

Qui abbiamo invece una scelta abbastanza ampia. Il Tipo più generico è il classico ***blob***. Ma disponiamo anche di ***raw***, ***clob*** etc.

8.2.6 Campi particolari: Intero ad Incremento automatico

Un tipo specifico come ***serial*** non è previsto in Oracle. Possiamo definire invece un tipo number(X) (quindi intero) associato ad una sequenza mediante un Trigger. Se la cosa sembra complicata, posso rassicurarvi: l'interfaccia grafica di gestione prevede una procedura automatica per la definizione di questo tipo di campi.

8.2.7 Campi particolari: Timestamp

Il ***Timestamp*** di Oracle NON viene aggiornato automaticamente dal motore di Db, e può invece essere usato per l'archiviazione di valori Data/Orario.

8.3 Interfaccia grafica di gestione

Abbiamo già detto che Oracle XE possiede una completa interfaccia grafica di gestione raggiungibile da un qualsiasi browser all'indirizzo:

```
http://server:8080/htmldb/
```

dove ***server*** è *l'indirizzo* oppure il *nome* del PC che esegue il motore di Db. Se è il nostro stesso PC, si può usare *localhost*. Ci accoglie la schermata di *login*; per entrare come Amministratore è necessario immettere come nome utente ***system*** e come password quella specificata in fase di installazione. Dopo il login abbiamo un'accoglienza molto diversa da quella del fratello maggiore:

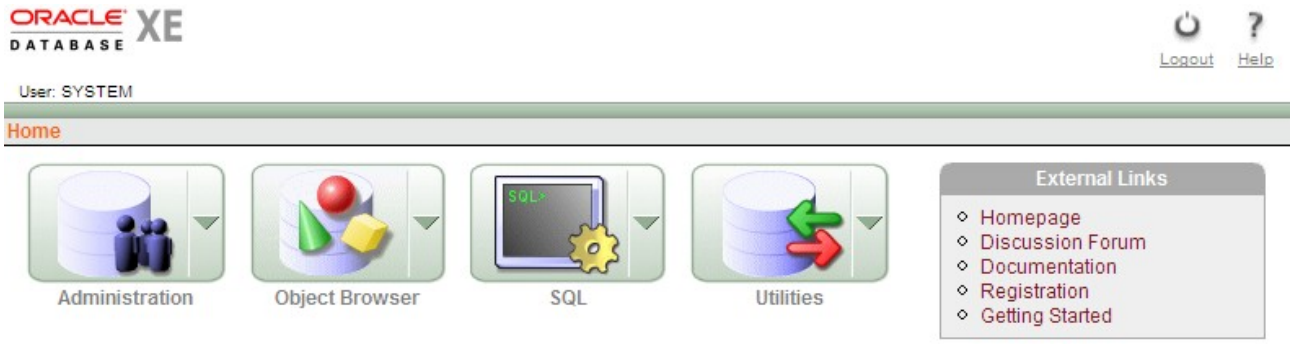


Figura 8.3.1: Interfaccia di Gestione di Oracle

Infatti, per la versione XE, Oracle ha completamente ridisegnato l'interfaccia di HTMLDB, rendendo tutto molto più semplice ed intuitivo. Da questo menu è possibile eseguire tutti i compiti amministrativi, su cui, ovviamente, non ci soffermeremo. Un utilizzatore proveniente da un altro prodotto simile (ad esempio MySql) potrebbe chiedersi, a questo punto, come si fa a creare nuovi schemi e nuove tabelle, visto che non c'è nessuna voce che prevede questa possibilità; è una domanda legittima, ma in Oracle, prima di poter creare questa tipologia di oggetti è necessario....

8.3.1 Creare un nuovo Utente

Il concetto di Database (come insieme logico di Tabelle, Viste etc.) in Oracle è strettamente connesso ad altri due definizioni: il ***Tablespace*** e ***l'Utente***. Il *Tablespace* è la struttura logica su cui vanno inseriti tutti gli oggetti di un Server. Il *Tablespace* deputato a contenere i dati si chiama ***USERS***. Quando si crea un nuovo utente, si genera anche una nuova struttura di database con lo stesso nome nel Tablespace assegnato all'utente stesso. Quindi, se io volessi creare la struttura per Mediateca, dovrei prima creare un utente di nome ***mediateca***.

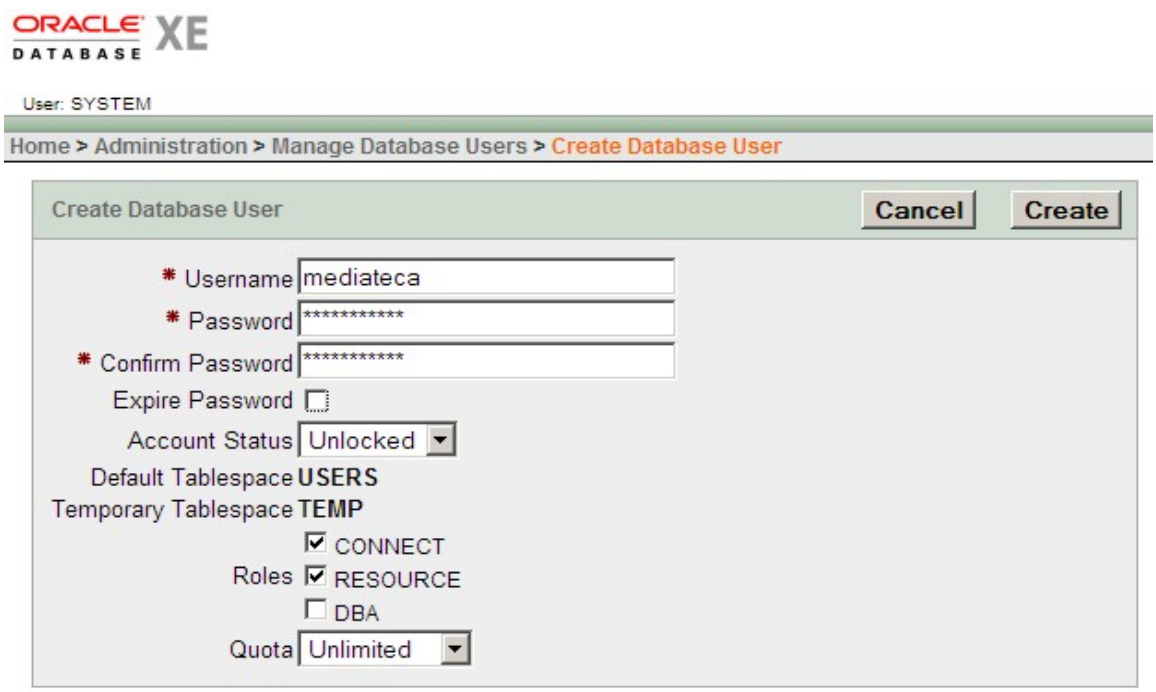


Figura 8.3.2: Creazione di un nuovo utente

L'*Account Status* deve ovviamente essere ***unlocked***, ed i ruoli da garantire sono almeno ***connect*** e ***resource***. A questo punto è possibile disconnettersi come *system* e rientrare come utente *mediateca*. Nel Menu *Object browser* compare la voce *create*, che permette finalmente di gestire il nostro db.

8.3.2 Creare una Tabella

Oracle prevede una specie di procedura guidata per la creazione di Tabelle.

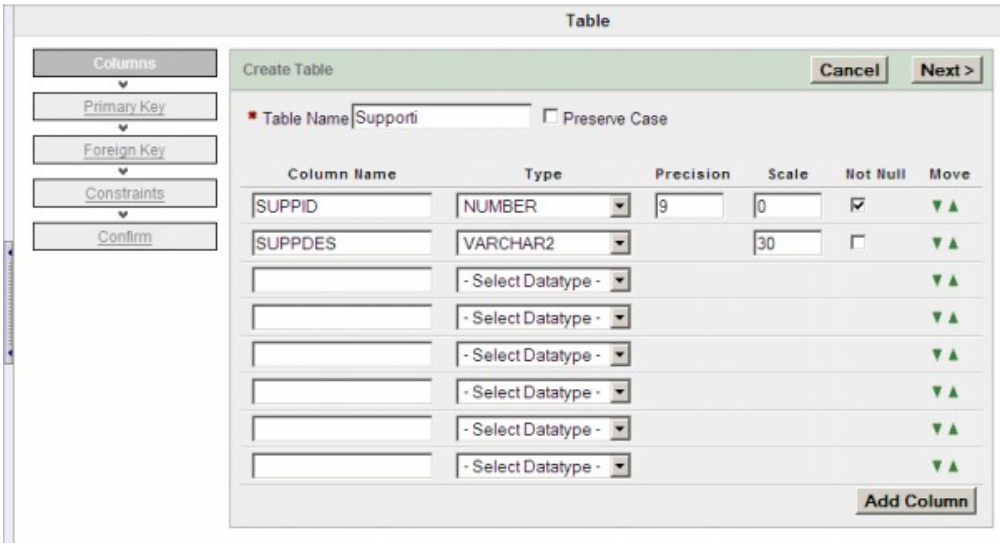


Figura 8.3.3: Creazione di Tabelle

Sulla sinistra sono elencati i vari passaggi da compiere, sulla destra la struttura dei dati. Il tipo di campo si può scegliere da una casella a discesa, e quindi è poi possibile indicare la grandezza. Tutto è molto semplice anche per utenti non particolarmente esperti. Col pulsante *next* si passa alla fase successiva.

La scelta della chiave primaria è particolarmente delicata:

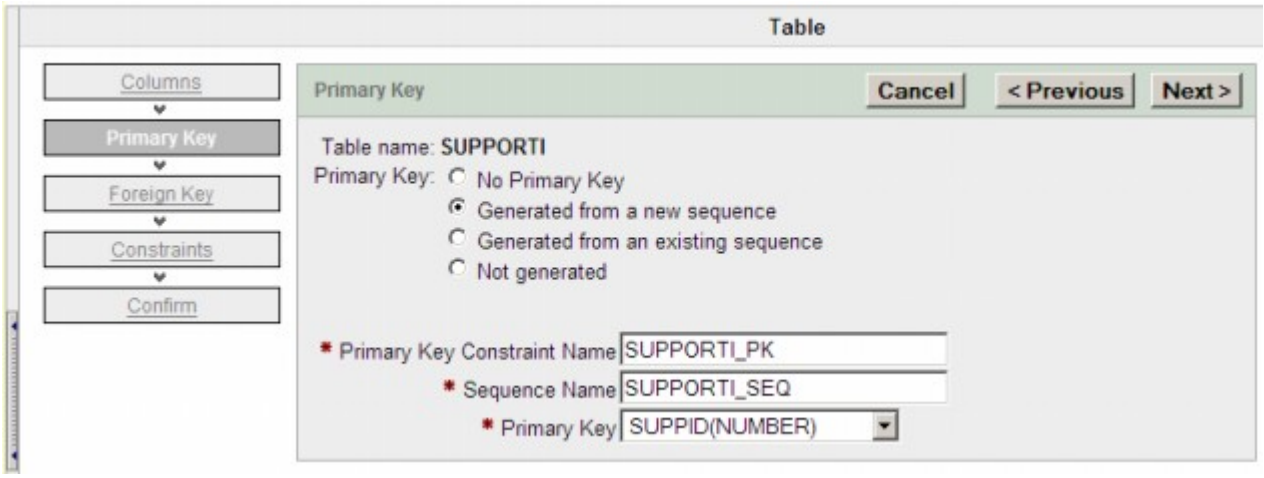


Figura 8.3.4: Scelta della Chiave Primaria

Se desideriamo un comportamento simile a quello di un campo ***serial*** classico, dobbiamo selezionare "*Generated from a new sequence*" ed indicare il campo della *Primary key*, come in figura.

Nel passaggio successivo è possibile definire *chiavi esterne*. Infine possiamo aggiungere direttamente dei *Vincoli* (ovviamente diversi dal *not null*, che si specifica direttamente nella struttura). Missione compiuta.

Con la stessa, semplice, interfaccia è possibile in qualsiasi momento modificare ogni aspetto della struttura del nostro Db. Davvero un ottimo lavoro.

8.4 Caratteristiche avanzate

Oracle è sicuramente il più completo e professionale motore di Db disponibile, quindi supporta tutte le caratteristiche necessarie ad un uso professionale dei Database. Per lo stesso motivo, un utente alle prime armi potrebbe sentirsi disorientato, anche se un grande sforzo è stato fatto per rendere *XE* semplice ed accessibile. Ovviamente, se il vostro scopo è archiviare le ricette di cucina o la rubrica degli amici potete sicuramente usare un altro prodotto. Se invece avete esigenze più importanti e l'affidabilità è fondamentale, la disponibilità di un prodotto come *XE* è un grosso vantaggio, anche se non è Open Source.

8.5 Driver ODBC

In attesa di un Client "leggero" specifico per XE, è necessario utilizzare il Client classico per Oracle e perciò siamo messi piuttosto male. Infatti di solito i programmi Client Oracle sono una suite corposa di strumenti, alcuni davvero interessanti, che occupano quasi un CD intero. In ogni caso è possibile scaricare anche una versione "ridotta" del client, esattamente l'***OLE Db Provider per Microsoft .NET***, che installa anche il Driver ODBC (sono circa 180 Mb). Sul sito è anche presente un "instant client" di pochi Mb, ma non sono riuscito a farlo funzionare.

TIP

Tra i Driver ODBC forniti con Windows Xp troviamo anche un "*Microsoft ODBC for Oracle*", che però ha bisogno *comunque* dell'infrastruttura di rete fornita dal Client nativo, quindi da solo non funziona. Anche molte Software House indipendenti forniscono Driver per Oracle, ma sono tutti a pagamento e la maggior parte ha lo stesso problema di quello di Microsoft.

L'installazione di OLE Db provider è piuttosto semplice: l'installer è il classico Oracle, e gli strumenti sono quelli essenziali per lo sviluppo di applicazioni su piattaforma Windows.

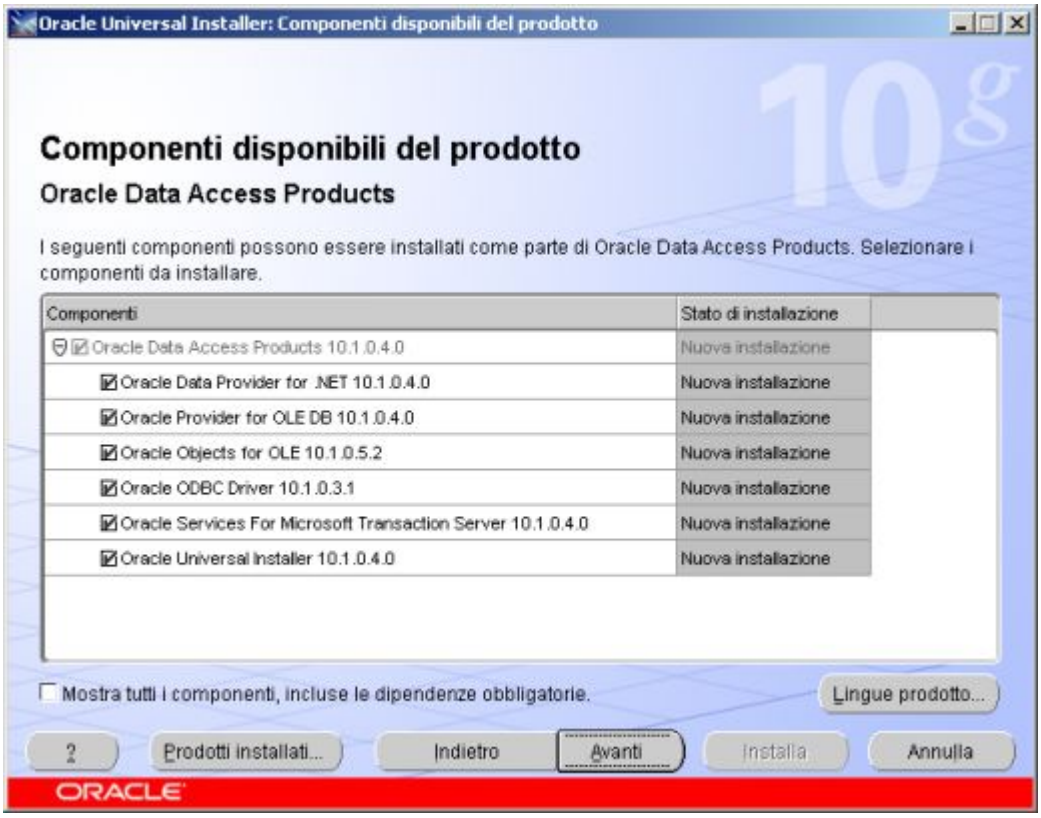


Figura 8.5.1: Oracle Installer

Alla fine dell'installazione parte in automatico l'assistente di configurazione di *Oracle Net*, che raccoglie le informazioni necessarie al collegamento ad un server: il *nome di servizio* (che, lo ricordo, è ***XE***), il *protocollo* (***TCP***), il *nome host* (o l'indirizzo IP) del server e la *porta* (***1521*** quella standard). Terminata questa fase, è possibile configurare un DSN: si sceglie il Driver Oracle, e la finestra di configurazione è questa:

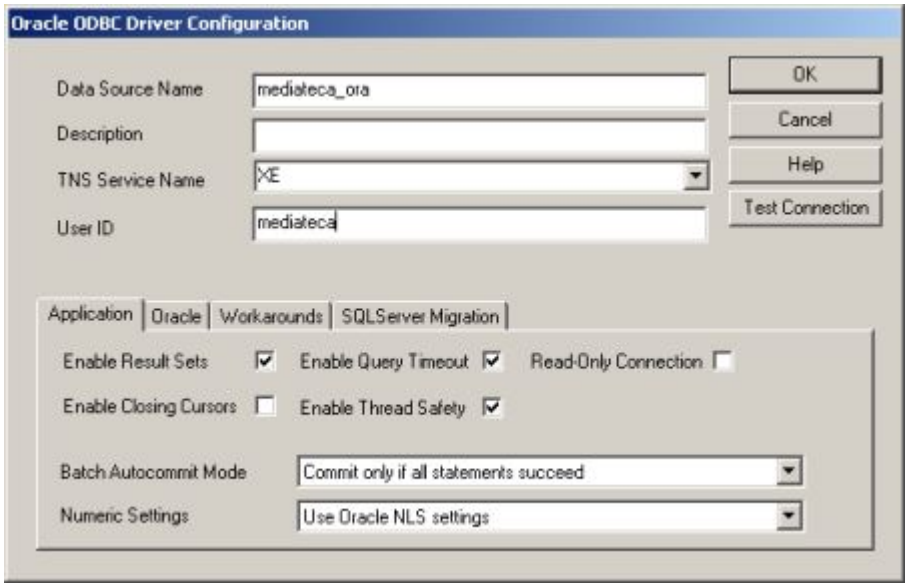


Figura 8.5.2: Configurazione del DSN

In sostanza è necessario solo specificare il nome del *Servizio TNS* precedentemente configurato (default ***XE***) ed un nome utente valido per l'accesso. Il pulsante *Test Connection* permette di verificare che tutto funziona bene.

TIP

Vi ricordo che il nome del servizio per un Server Oracle classico è ***orcl*** e non ***XE***. Quindi è necessario specificare ***orcl*** in tutti i parametri di collegamento ad un ***10g***.

8.6 Driver JDBC

Qui le cose vanno meglio, perché è possibile scaricare Driver per Windows, per Linux ed anche per Mac Os X. Si tratta del classico file .jar, nel nostro caso esattamente ***ojdbc14.jar***, da sistemare, al solito, in una cartella di nostra scelta.

9. Usare OpenOffice.org con...

9.1 MySql

In generale l'utilizzo di OOo con MySql è abbastanza agevole. Si può scegliere se usare *ODBC* o *JDBC*, e vedremo le peculiarità ed i problemi che si possono verificare, sia in Windows che in Linux.

9.1.1 ODBC

Abbiamo già visto, nel Capitolo dedicato a *MySql*, come si imposta un ***DSN*** (*Data Source Name*): per il collegamento ODBC questo è un passaggio preliminare che non può essere evitato. Per creare un Documento Base OOo collegato a MySql, nella prima schermata del Wizard è necessario scegliere ***Collega a un Database esistente -> MySql***. Nella finestra successiva, ovviamente, ***Connetti con ODBC***. Quindi è necessario indicare il nome della sorgente dati ***DSN*** preventivamente creata: il pulsante *Sfoglia* può essere d'aiuto. Nel passo successivo è necessario indicare un *Nome Utente* solo se le credenziali non sono state associate al *DSN*; il pulsante *Test Connection* permette di verificare che il Server risponda in modo corretto. Infine si può o meno registrare il Database, ma è necessario ovviamente assegnare un *Nome* al File. La procedura in ***Linux*** è esattamente la stessa. Sarà bene tener presente che:

- Con *MySql 5.0* i **campi *decimal*** vengono gestiti in modo non corretto con i Drivers ODBC ***precedenti*** alla versione ***3.51.12***; come abbiamo già avuto modo di dire, MySql 5.0 non considera più il Tipo *decimal* come testo, ma come valore binario; questo impedisce la memorizzazione corretta dei dati da parte di OOo. Se il campo *decimal* è definito *Not Null*, OOo ritorna un messaggio di errore, in caso contrario qualunque cosa si scriva assegna il valore *null*. La cosa viene superata con Driver aggiornati
- In generale *non è conveniente modificare la struttura delle Tabelle direttamente con OOo*; infatti i tipi di campo non sempre corrispondono perfettamente, ed il Tipo TimeStamp viene interpretato come Data/Ora [DateTime]. Inoltre assegnare valori predefiniti a campi numerici non porta a risultati affidabili. Infine non è possibile assegnare ai campi un esempio di formato.
- Per quanto MySql sia un Db relazionale, almeno se si usano Tabelle InnoDb, il Menu ***Strumenti -> Relazioni*** sostiene il contrario.
- L'***Amministrazione degli utenti*** in OOo è piuttosto rudimentale (Menu *Strumenti -> Amministrazione degli Utenti*); se si crea un nuovo Utente viene archiviato nella forma ***utente@%*** quindi può collegarsi da qualsiasi Host; inoltre nella finestra di gestione non

è possibile assegnare alcun diritto, e quelli elencati comunque fanno riferimento alle sole Tabelle dello Schema.

- E' possibile, se si usa MySql 5.0, creare ***viste*** direttamente da OOo, anzi è sicuramente più semplice che in MySql Administrator; l'interfaccia è uguale alla creazione di Ricerche, quindi comoda ed efficiente. Una volta archiviata, però, la struttura di una Vista non è più modificabile direttamente da OOo; inoltre i Dati sono in sola lettura

La velocità di risposta con il driver ODBC è soddisfacente anche con archivi corposi.

9.1.2 JDBC

Abbiamo visto come scaricare ed installare i Driver JDBC per MySql. Il file .jar deve essere archiviato in una cartella a piacere, e deve essere aggiunto alla *Classpath* di OOo con *Strumenti -> Opzioni -> Java -> Classpath*. Nel Wizard del Database si deve selezionare, al solito, ***Collega a un Database esistente -> MySql***. Nella finestra successiva, ovviamente, ***Connetti con JDBC***. Quindi avremo una finestra come quella in figura. Bisogna immettere il *Nome del Database*, l'*URL del Server* (volendo anche come indirizzo IP) ed accertarsi che sia indicata la giusta *Classe JDBC*. Il pulsante *Classe di Prova* permette di testare l'esistenza della Classe stessa. Andando avanti, è necessario specificare un *Nome Utente* e selezionare il check *Password Required*. Fatto. In ***Linux*** la procedura è esattamente la stessa.

Con *JDBC* ogni volta che si apre una documento Base è necessario specificare la password per accedere ai dati, perché la password stessa non viene archiviata con la connessione.

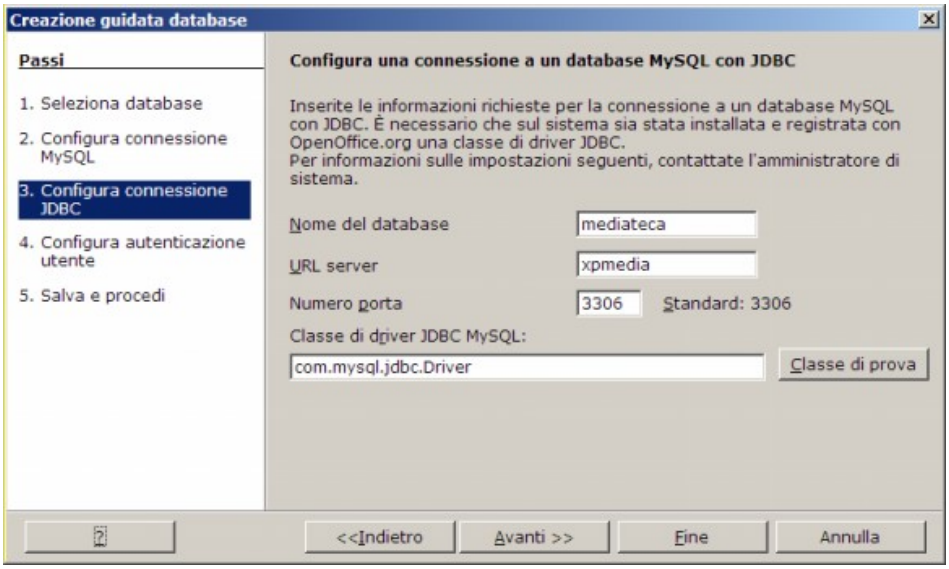


Figura 9.1.1: Configurazione connessione JDBC con MySql

Tenete presente che:

- Con il Driver *JDBC* attuale (versione 3.1.12) è ***impossibile*** aprire Tabelle che contengano tipi di dati ***Decimal,*** se il Server è *MySql 5.0*, probabilmente per lo stesso motivo già visto con ODBC.
- L'accesso ai dati è lento, quindi questa modalità è sconsigliata se gli archivi da gestire sono consistenti.

9.2 PostgreSQL

Il problema più grosso con PostgreSQL, è la pessima gestione che OpenOffice fa dei campi di tipo ***Serial*** (***Integer Auto Increment***) con il Driver *ODBC*. Questo in parte è dovuto al fatto che, in Postgres, il tipo *Serial* non è nativo, ma viene gestito attraverso il riferimento ad una *Sequenza* (lo stesso problema si verifica con *Firebird*, altro Server Db Open Source). C'è anche da dire, però, che una issue sul bug è aperta da svariati anni senza che gli sviluppatori si siano davvero impegnati a correggere l'errore. Questo rende l'utilizzo di *PostgreSQL* con OpenOffice attraverso ODBC assai problematico. Nonostante tutto *Postgres* è un buon prodotto, e vale la pena di descrivere come si integra nei nostri documenti Base, se non altro per permettere l'accesso in lettura a queste tipologie di dati.

9.2.1 ODBC

Al solito, è necessario avere installato il Driver ODBC ed aver creato l'opportuno DSN. Per creare un Documento Base OOo collegato a PostgreSQL, nella prima schermata del Wizard è necessario scegliere ***Collega a un Database esistente -> ODBC***. Quindi è necessario indicare il nome della sorgente dati ***DSN*** preventivamente creata: il pulsante *Sfoglia* può essere d'aiuto. Andando avanti, è necessario specificare un *Nome Utente* e selezionare il check *Password Required*. Infine si può o meno registrare il Database, ma è necessario ovviamente assegnare un Nome al File. Attenzione: il nome del Database nel DSN è case-sensitive, per cui *Mediateca* e *mediateca* sono due cose diverse, anche sotto Windows.

Non ho testato il Driver *ODBC* PostgreSQL in ***Linux***, non essendo disponibile un RPM. Comunque:

- non è possibile modificare la struttura delle Tabelle direttamente da OOo; meglio: è possibile aggiungere colonne, ma non modificare quelle esistenti; in ogni caso, come abbiamo già avuto modo di dire, è sempre meglio, per queste funzionalità, utilizzare gli strumenti nativi.
- Per quanto PostgreSQL sia un Db relazionale, il Menu ***Strumenti -> Relazioni*** sostiene il contrario.
- L'Amministrazione degli Utenti direttamente da OOo non è supportata.

- Il problema con i campi ***serial*** può essere risolto solo immettendo manualmente l'identificatore; se la colonna si lascia vuota, OOo ritorna un errore ma la riga viene comunque aggiunta.
- Il numero di decimali visualizzato per i campi ***float*** è due; è comunque possibile modificare il formato del numero selezionando la colonna e scegliendo dal menu contestuale la voce proprietà.
- Il formato dei campi di tipo ***time*** è *hh.mm.ss* ed il valore va immesso con i *punti*; se si usa il tastierino numerico, OOo aggiunge una virgola ed il valore non viene riconosciuto.
- E' possibile creare ***viste*** direttamente da OOo, anzi è sicuramente più semplice che con *pgAdmin*; l'interfaccia è uguale alla creazione di *Ricerche*, quindi comoda ed efficiente. Una volta archiviata, però, la struttura di una Vista non è più modificabile direttamente da OOo; inoltre i Dati sono in sola lettura.

9.2.2 JDBC

Abbiamo visto come scaricare ed installare i Driver *JDBC* per PostgreSQL. Il file .jar deve essere archiviato in una cartella a piacere, e deve essere aggiunto alla *classpath* di OOo con *Strumenti -> Opzioni -> Java -> Classpath*. Per la creazione di un nuovo documento Base di tipo JBDC Postgres, bisogna scegliere, nel Wizard, la voce JDBC. Nella finestra successiva è necessario specificare l'URL del collegamento nella forma:

```
jdbc:postgresql://server/database
```

dove *server* è il nome o l'indirizzo IP del server e *database* è il nome del Db da utilizzare. Nel campo Classe Driver è necessario immettere la stringa:

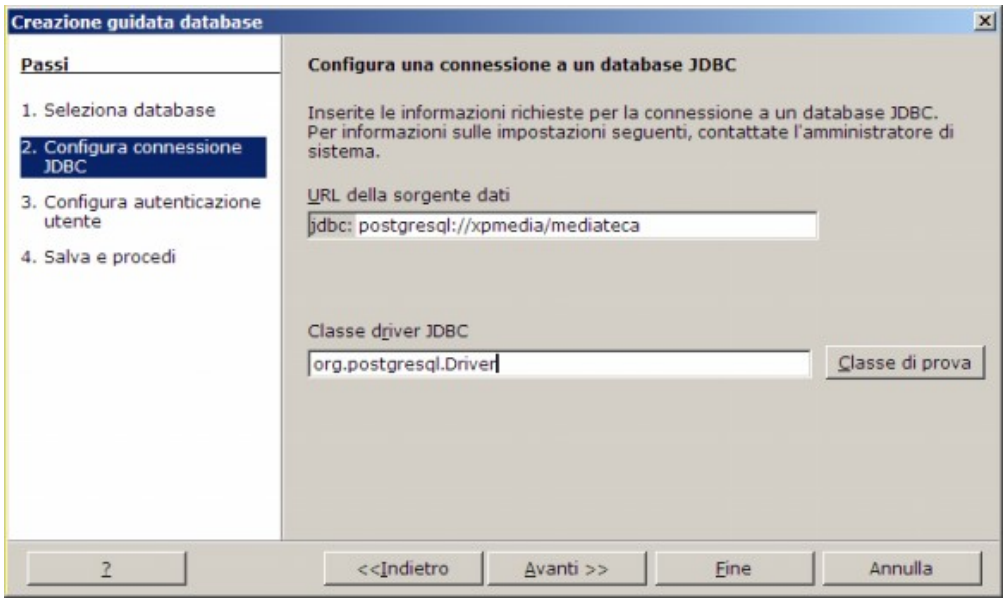


Figura 9.2.1: Connessione JDBC ad un server PostgreSQL

```
org.postgresql.Driver
```

Il pulsante *Classe di Prova* permette di testare l'esistenza della Classe stessa. Andando avanti, è necessario specificare un *Nome Utente* e selezionare il check *Password Required*. Si noti che:

- il nome del Database è *case-sensitive* (cioè è necessario scrivere esattamente il nome corretto, e *Mediateca* è diverso da *mediateca*).
- Come con ODBC, non è possibile modificare la struttura delle Tabelle direttamente da OOo.
- È possibile visualizzare e modificare le ***Relazioni*** dal Menu Strumenti->Relazioni.
- L'Amministrazione degli Utenti direttamente da OOo non è supportata.
- La gestione del tipo ***serial*** *funziona egregiamente* con JDBC, almeno col driver *8.1-404.jdbc3*, sia con tabelle complete di OIDS che senza.
- In compenso non viene riconosciuto il tipo ***bool***, ed il valore (vero o falso) deve essere assegnato esplicitamente.
- Purtroppo neppure il tipo ***decimal*** viene riconosciuto in modo corretto, e non è possibile immettere alcun valore.
- Per il resto vale quanto detto a proposito del Driver ODBC.

9.3 Microsoft SQL 2005 Express Edition

L'utilizzo di SQL 2005 Express Edition non comporta grosse difficoltà, almeno con *ODBC* in Windows.

9.3.1 ODBC

Al solito, è necessario avere installato il Driver ODBC ed aver creato l'opportuno DSN (vi ricordo che il Driver si chiama *SQL Native Client*). Per creare un Documento Base OOo collegato a MSEE, nella prima schermata del Wizard è necessario scegliere ***Collega a un Database esistente -> ODBC***. Quindi è necessario indicare il nome della sorgente dati ***DSN*** preventivamente creata: il pulsante *Sfoglia* può essere d'aiuto. Andando avanti, è necessario specificare un *Nome Utente* e selezionare il check *Password Required*, solo se le credenziali di connessione non sono state specificate nel DSN. ne Infine si può o meno registrare il Database, ma è necessario ovviamente assegnare un Nome al file.

Nell'elenco delle Tabelle OOo mostra anche quelle di sistema, quindi è comodo impostare un filtro. Le Tabelle utente sono nel "ramo" *dbo*. Considerazioni:

- pochi problemi con i tipi di dati: i *serial* funzionano senza problemi (definiti come *int* con l'opzione *identity*); lo stesso i *money*, i *bool* (definiti come *bit*), ed il tipo *real*.

- Il tipo *float*, invece non viene gestito in modo corretto; nel campo, infatti, immettendo un numero con la virgola, lo stesso viene arrotondato ad intero; bisogna quindi inserire dati decimali col punto, che, tra l'altro, vengono formattati piuttosto male.
- I tipi *datetime* e *smalldatetime* si comportano bene, salvo che l'immissione di un orario senza data è piuttosto problematica.
- E' impossibile (com'era prevedibile) gestire gli utenti direttamente da OOo.
- Pare sia consentito invece impostare relazioni e chiavi esterne con OOo, ma sinceramente non so quanto possa essere affidabile.
- Non si può, da OOo, modificare la struttura delle Tabelle.
- E' possibile creare viste, ma nel salvare non deve essere indicato il nome del Db.

9.3.2 JDBC

Nonostante sia possibile scaricare un driver JDBC dal sito di Microsoft (ancora in beta), non sono riuscito a stabilire un collegamento al server con OOo: in pratica pare che OpenOffice passi dei parametri non corretti al Driver nella stringa di connessione e quindi, nonostante il classpath sia regolarmente riconosciuto, si ottiene solo solo un messaggio di errore. Ovviamente questo preclude ogni possibilità di utilizzo di questa tipologia di dati da Desktop Linux (mi sarei meravigliato del contrario...).

9.4 Oracle 10i Express Edition

Oracle è il dominatore della categoria dei Database professionali. La lunga presenza sul mercato e la proverbiale affidabilità ne fanno un prodotto difficile da battere in ambito "mission critical"; allo stesso tempo è un Software complesso ed assai sofisticato, abbastanza difficile da gestire e non adatto ai "principianti". Non è certo questa l'occasione per descrivere in dettaglio le caratteristiche di Oracle: ci limiteremo alle informazioni essenziali ai nostri scopi.

Vi dico subito che i Db Oracle possono essere *consultati*, ma non *aggiornati,* almeno in modo affidabile, da OOo.

9.4.1 ODBC

Abbiamo gia visto come configurare un DSN per Oracle. Per creare un Documento Base OOo collegato a Oracle XE, nella prima schermata del Wizard è necessario scegliere ***Collega a un Database esistente -> ODBC***. Quindi è necessario indicare il nome della sorgente dati ***DSN*** preventivamente creata: il pulsante *Sfoglia* può essere d'aiuto. Andando avanti, è necessario specificare un *Nome Utente* e selezionare il check *Password Required*. Infine si può o meno registrare il Database, ma è necessario ovviamente assegnare un Nome al File.

Il funzionamento di OOo con Oracle XE in ODBC potrebbe definirsi "bizzarro". Infatti il modulo Base elenca correttamente le Tabelle del nostro Db, comprese quelle di sistema. Ma, se

si fa un doppio click per aprire una Tabella, non si vede.... nulla. In pratica si apre una finestra vuota che non mostra neppure i nomi dei campi. Il bello è che, se creiamo una ricerca, come per magia tutto funziona. Oddio, tutto funziona è eccessivo... Si apre una finestra dove è possibile consultare i dati, ma quanto ad aggiungere o modificare, scordatevelo. I risultati sono totalmente inaffidabili.

9.4.2 JDBC

Il modo più semplice per connettersi ad un Db Oracle è senza dubbio JDBC. Basta scaricare il Driver dal sito di Oracle e copiare il file *ojdbc14.jar* in una dir di vostra scelta. A questo punto, se non volete modificare a mano il file *java.ini*, è opportuno aggiungere la classe del Driver ad OpenOffice. Questo si ottiene con *Strumenti -> Opzioni -> Java*: nella finestra, alla pressione del tasto **Classpath...**, avremo la possibilità di aggiungere l'archivio di classi a OOo, come in figura.

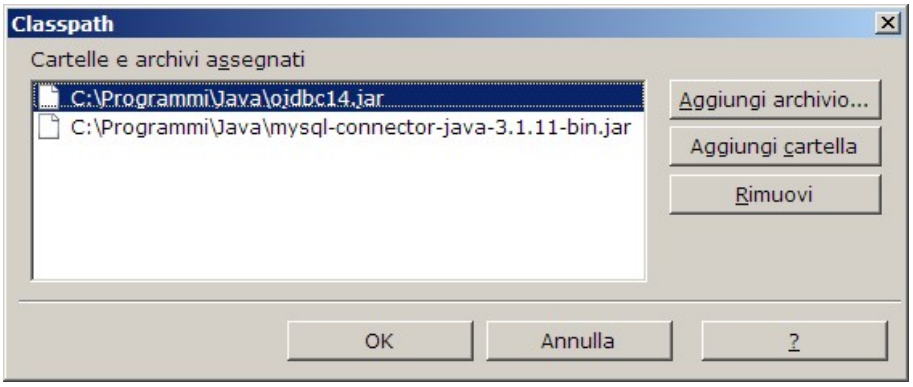


Figura 9.4.1: Aggiunta di una Classe a OOo

Dopo questo passaggio è necessario riavviare OOo (compreso il quickstart), e saremo pronti a stabilire una connessione. All'apertura del modulo Base, scegliamo "*Collega ad una Database esistente*" e quindi, dalla casella a discesa, "*Oracle JDBC*". Nella finestra successiva è necessario indicare il *nome del Database* (normalmente **xe**), *il nome o l'indirizzo IP del Server* (nel nostro caso **asus**), ed *il numero di porta* (**1521** è quella standard). Nella schermata successiva è poi possibile specificare un nome utente, ma non una password (che sarà richiesta ad ogni ulteriore connessione).

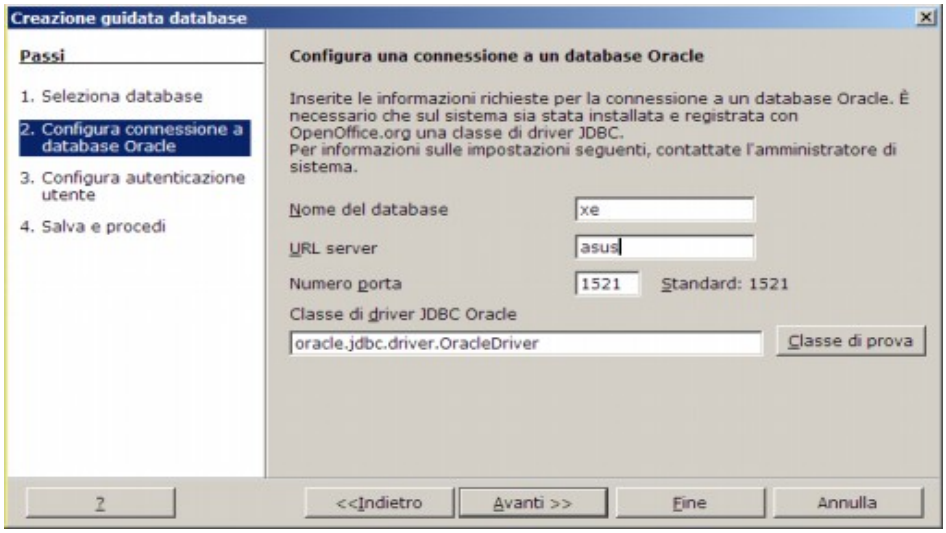


Figura 9.4.2: Connessione JDBC a Oracle

Le connessioni JDBC a Tabelle Oracle XE sono inaffidabili se si cerca di aggiungere o modificare i dati. In sostanza ogni aggiornamento di informazioni porta quasi inevitabilmente ad errori di vario tipo, perciò direi che possono essere usate esclusivamente in lettura.

9.5 Sybase SQL Anywhere

Sybase, con una linea di prodotti Server piuttosto nutrita, è un altro dei contendenti al ruolo di miglior Db professionale in commercio. In realtà non si tratta di un prodotto molto diffuso, e ne parliamo solo per un motivo: è stato scelto come strumento di sviluppo per il SW SISSI in rete, cioè il sistema di automazione delle Segreterie Scolastiche fornito dal Ministero. Quindi lo troviamo installato nell'80 % delle Scuole Italiane, e magari a qualcuno può tornare utile sapere come si accede ai dati di SISSI (solo in lettura, mi raccomando...). Nelle Scuole di solito viene installato SQL Anywhere Studio 8.0.X, ed è a questo prodotto che faremo riferimento.

9.5.1 ODBC

Il Driver viene istallato dal CD del Server, e non è possibile scaricarne una versione autonoma da Internet. Quindi sul PC che fa da Server avremo il prodotto completo, sulle Stazioni di Lavoro solo la parte Client. SISSI installa già un DSN su ogni PC per l'accesso alla base dati, ed è di questo che ci serviremo.

Per accedere al Db, basta selezionare un collegamento ODBC, e specificare il DSN SISSI. Di solito il DSN è già fornito di credenziali: se così non fosse, il nome utente è SISSI e la password è SISSI (ogni commento è superfluo....).

Se tutto fila liscio, avremo nella sezione Tabelle del nostro documento Base tutte le Tabelle del Db, comprese quelle di sistema.

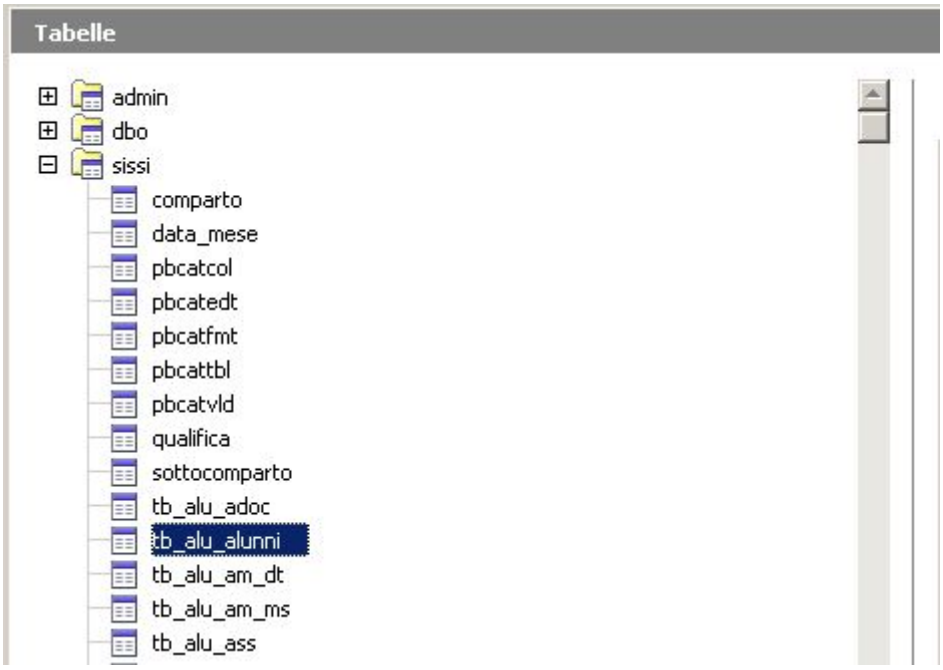


Figura 9.5.1: Le Tabelle di SISSI in Rete in un Documento Base

Ovviamente è sconsigliato modificare i dati direttamente da OOo, mentre si può benissimo estrarre informazioni con Ricerche ed usarle, ad esempio, per un Mail Merge. Non è possibile modificare la struttura delle Tabelle e neppure le Relazioni: se proprio volete sperimentare, è possibile usare Sybase Central, strumento di gestione fornito insieme al Server.

9.6 Microsoft Access

Il formato .mdb è molto diffuso in ambiente Ms Windows ed è strettamente correlato a Microsoft Access. Tanto per essere chiari, *è possibile usare database in formato .mdb esclusivamente con Windows*. Infatti non esiste un driver ODBC affidabile in Linux, ed anche il Progetto *MdbTools* non ha ottenuto finora risultati apprezzabili (ritengo soprattutto perché non esiste un reale interesse ad usare questo formato su piattaforme unix like). Un file .mdb può contenere, oltre alle tabelle ed agli indici, anche tutti gli altri elementi che compongono un Database Access: Query, Maschere, Report, Macro e Moduli.

9.6.1 Connessioni

Le ***versioni Windows di OOo*** prevedono la gestione di Db *.mdb* con tre modalità di accesso: *diretta*, *ODBC* e *ADO*. Nella modalità ***diretta*** è possibile scegliere dalla casella a discesa *Collega ad un Database esistente* la voce *Microsoft Access* e quindi specificare nella schermata successiva il percorso del file da aprire. Se invece si opta per ***ADO***, la procedura è un po' più complessa, perché si apre la finestra di selezione di connessione ADO da cui scegliere la voce *Microsoft Jet 4.0 OleDb Provider*. Infine la modalità ***ODBC*** prevede la

creazione preventiva di una sorgente dati collegata al Driver Microsoft Access, così come avviene, ad esempio, per MySql. Consiglio, ovviamente, la modalità *diretta* che equivale comunque ad *ADO* (OleDb provider).

9.6.2 Tabelle e Query

Una volta collegato il Database .mdb ad OOo, alla voce Tabelle compaiono sia le Tabelle vere e proprie sia quelle che Access chiama *Query*. Anzi, direttamente da OOo è possibile creare nuove Query di Access utilizzando la voce *Crea Vista* dal Pannello *Attività*. Peccato però che una volta create, le Viste non siano più modificabili...

9.6.3 Tipo di Campo

La corrispondenza tra i Tipi di Campo è abbastanza rispettata. Tenete però presente che:

- la creazione / modifica di Tabelle direttamente da OOo non è perfetta e spesso il risultato è un criptico messaggio di errore
- non c'è verso di assegnare una dimensione ai *campi di testo*; la scelta del tipo è obbligatoria in ***Varchar***, e se anche impostato, il valore della dimensione viene convertito da Access in 255
- in genere nessun problema per i campi *Valuta*, *Logico*, *Intero*, *Intero ad incremento automatico, Data ed Ora*
- in Access non esiste un equivalente del Tipo *Timestamp*, perfettamente inutile in questo contesto
- se si usa il Driver *ODBC*, la struttura delle Tabelle *non può essere modificata* ed OOo non riconosce gli indici e la chiave primaria; in compenso non vengono elencate le Tabelle *MSys*

9.6.4 Diretta / ADO - OleDb

Con questo tipo di connessione tra le Tabelle vengono elencate anche quelle di sistema, che hanno il nome che inizia con *Msys*... . Normalmente in Access questi elementi sono nascosti, e credo sia inutile dire che non è una buona idea aprirli con OOo.

Purtroppo OpenOffice soffre di un ***bug***: non è infatti possibile immettere, in Tabelle Access, valori corretti in campi definiti come "*Decimale*"; ogni tentativo di immissione di valori, ad esempio, come 12,44 porta all'archiviazione di 1244,00.

9.6.5 ODBC

Se si usa il Driver *ODBC*, la struttura delle Tabelle *non può essere modificata* ed OOo non riconosce gli indici e la chiave primaria; in compenso non vengono elencate le Tabelle *Msys.*

Però i campi Decimali funzionano. In lettura i dati sono affidabili, ed in scrittura anche, almeno per quanto sia possibile verificare in un test non approfondito.

9.7 Microsoft SQL Server 2000

Sql Server è il Database "professionale" di Microsoft, quello con cui la Società americana ritiene di poter essere competitiva sul mercato Enterprise. Il Software viene distribuito in varie release, addirittura una versione ridotta (fino a 25 utenti e senza strumenti di gestione) è scaricabile gratuitamente dal sito di Microsoft. Per testare OOo abbiamo usato una build 8.00.760 (2000 Sp3 Developer). Come per Access, anche in questo caso non è possibile accedere ai dati da PC che non siano forniti di una versione di Ms-Windows.

Ms Sql Server è un prodotto (almeno nelle versioni FULL) assai sofisticato e di non facile gestione. Supporta i *Trigger* e le *Stored Procedure* oltre a varie forme di replica e di distribuzione del carico. L'interfaccia di amministrazione si chiama *Enterprise Manager*, e permette di controllare abbastanza agevolmente tutti i Server presenti sulla propria rete.

9.7.1 Connessioni

Al solito, è possibile collegare Tabelle e Viste di Ms Sql attraverso ***ADO*** oppure ***ODBC***. Con *ODBC* è necessario creare preventivamente una sorgente dati con le solite modalità. La connessione *ADO* è invece possibile scegliendo *OLE Db provider for SQL Server* nella lista delle proprietà del Data Link. Nella Tab *Connessione* è necessario indicare: il nome o l'indirizzo IP del Server; la modalità di autenticazione; il nome del Database a cui collegarsi.

Ms Sql ha due modalità di *gestione della sicurezza*, impostate una tantum sul Server: può autenticare gli utenti attraverso la protezione integrata di Windows oppure con una sua gestione autonoma. Nel primo caso non è necessario impostare nome utente e password: se l'admin ha deciso che potete accedere, porte aperte, altrimenti nisba. Nel secondo caso dovete conoscere le credenziali: nelle prime versioni di SQL Server 2000, *l'admin* era automaticamente impostato a *sa* ed era privo di password (davvero un bell'esempio di cura per la sicurezza....).

9.7.2 ODBC

Se si accede tramite *ODBC* fortunatamente non compaiono le Tabelle di sistema (il cui nome comincia per *sys*), però non è possibile modificare la struttura del Db. Inoltre è praticamente *impossibile* gestire valori dichiarati di tipo *Money*, ed anche i numeri *Float* soffrono di errori di arrotondamento. Questa modalità può essere usata in modo accettabile ***solo in lettura***, ad esempio se si desidera eseguire un Mail Merge oppure trasferire dati al Foglio elettronico.

9.7.3 ADO

In questo caso compaiono le Tabelle di sistema, e la struttura del Db è modificabile. Purtroppo però quando si apre una Tabella, il primo record viene duplicato su tutte le righe, rendendo praticamente ***inusabile*** il Database.

9.8 dBase

Perdonatemi una breve (ma forse utile) digressione, amici miei. E' infatti con gli occhi lucidi ed un groppo alla gola che mi accingo a parlarvi di un vecchio amico, che molti di voi neppure conosceranno, il dBase. **dBase II** è stato il primo SW di gestione DB che ha avuto successo nell'arcano mondo di Ms-Dos, ai tempi remoti che Berta filava... Stiamo parlando della metà degli anni ottanta, quando ogni PC che voleva definirsi tale doveva avere a bordo un SW di elaborazione testi (WordStar), un foglio elettronico (Lotus 123) ed appunto dBase II. La leggenda dice che dBase I non sia mai esistito e che gli sviluppatori partirono dal 2 per dare un'aria di "vissuto" al programma. Fatto sta che lo standard "dBase" ha conosciuto da allora un enorme successo, e ce lo ritroviamo pari pari nel nostro aggiornatissimo OOo.

dBase è un semplice formato di archiviazione che prevede un file dati con estensione **.dbf** per ogni tabella, associato ad uno o più file con estensione **.ndx** per gli indici. Non si tratta quindi di un server Db, quanto piuttosto di una specifica di formato che ognuno può utilizzare come vuole. Nonostante l'età, le prestazioni sono più che dignitose, ed ancora può essere usato per scopi non troppo complessi.

Per farla breve, non vi parlerò delle evoluzioni che dBase ha avuto nel tempo (dBase III, poi IV, poi il Clipper, uno dei compilatori più diffusi in passato), ma solo di come usarlo con OOo, perché può tornare utile in più di una occasione.

9.8.1 Gestire un Db

Per usare un Db di tipo *dBase* abbiamo solo bisogno di una cartella sul Disco Rigido. Nella procedura guidata, infatti, se si sceglie *dBase* come connessione ad un Database esistente viene richiesto solo il percorso dove sono archiviate (o bisognerà archiviare) le Tabelle. Se le Tabelle esistono, saranno riportate nel nostro documento Base, in caso contrario sarà possibile crearle direttamente da OOo.

9.8.2 Tipi di Dati

Non abbiamo molto da scegliere: *Char*, *Varchar*, *Booleano*, *Memo* (una sorta di *BLOB*) e *Data* (che comprende anche l'ora). Per i Campi numerici abbiamo solo *Decimale*, e si può specificare la lunghezza ed il numero di posizioni decimali. Niente *intero ad incremento automatico* e neppure *timestamp*.

Il nome del campo è limitato a dieci caratteri, in omaggio al vecchio dBase, con cui è evidentemente necessario mantenere la compatibilità, quindi bisogna tenerne conto se si importano Tabelle da altre applicazioni.

Per gli *indici*, nulla da segnalare, se non che ogni indice è archiviato in un file separato con estensione *.ndx* ed associato alla tabella tramite un file *.inf*. Se nella struttura è compreso un campo di tipo *memo*, avremo anche un file *.dbt*.

9.8.3 Considerazioni.....

Svantaggi:

- ✔ manca la gestione del tipo *Serial*, quindi i campi di chiave primaria vanno riempiti a mano
- ✔ non è un Db relazionale, quindi niente chiavi esterne ed integrità referenziale

Vantaggi rispetto a HSQL:

- ✔ è multiutente, basta archiviare i dati in una Cartella condivisa
- ✔ è molto più veloce e molto meno avido di risorse

In conclusione, se le esigenze non sono complesse e non richiedono l'integrità referenziale, meglio *dBase* che *HSQL*.

9.9 Rubrica di Thunderbird

Thunderbird è il programma di Posta Elettronica della Mozilla Foundation che, assieme al "cugino", più famoso, browser *Firefox* sta conquistando un'ottima base di utenti nel mondo. Sarà per questo che gli sviluppatori di OpenOffice si sono preoccupati di garantire l'accesso alla *Rubrica* di Thunderbird in modo semplice e diretto.

Basta infatti scegliere *Rubrica di Thunderbird* dalla casella a discesa di *Collega ad un Database esistente* per avere subito disponibili i dati. Infatti, nel pannello Tabelle, sono elencate sia le *Rubriche* (possono essere più di una, quella principale si chiama *Rubrica Personale*) sia le *Liste di Invio*. I campi ci sono tutti, ma ***le informazioni sono in sola lettura***. In compenso è possibile creare *ricerche*, *formulari* e *rapporti*.

10. Il Modulo "Base"

Supponiamo che, seguendo le istruzioni precedenti, abbiate creato il nostro Database di prova completo di Tabelle, Indici e Relazioni. Nel seguito faremo riferimento a Tabelle collegate ad un Db MySql, ma la maggior parte delle cose che diremo può essere applicata a qualsiasi motore di Db. Dovremmo avere una situazione simile a questa:

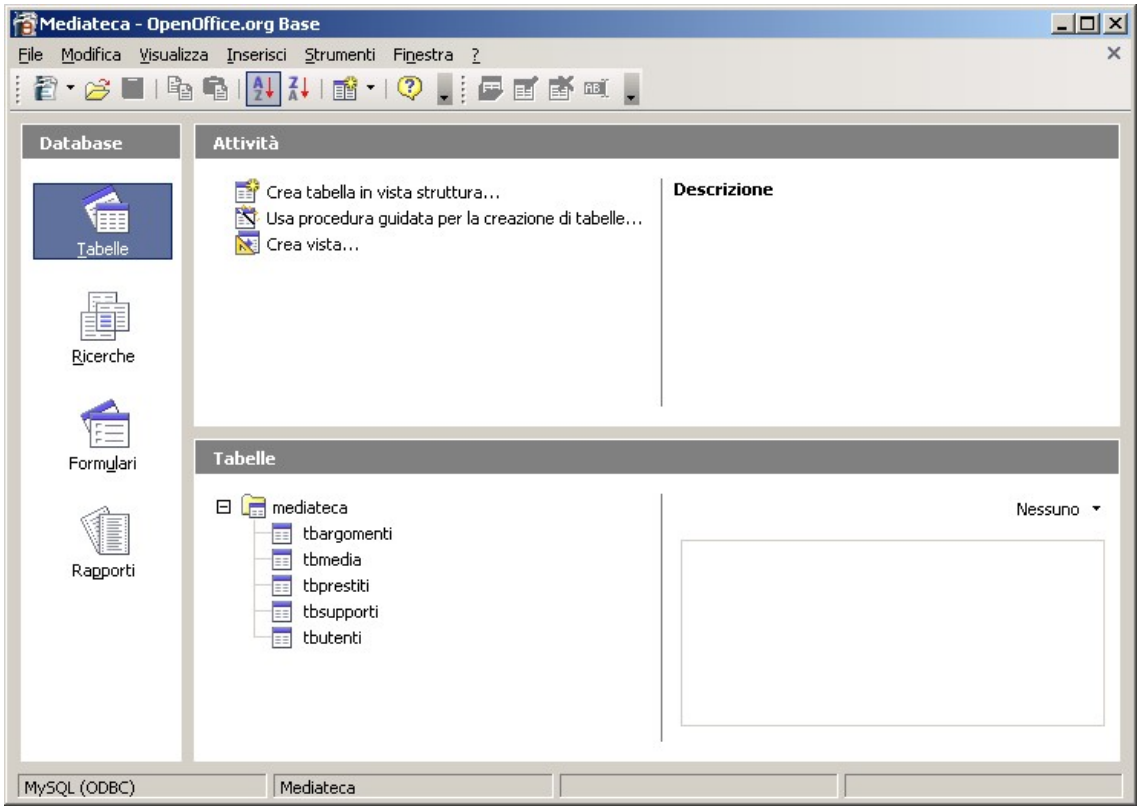


Figura 10.1 Il Modulo Base

In effetti l'aspetto ha qualcosa di familiare per chi ha usato altri prodotti simili, e questo certo facilita le cose. Sulla sinistra abbiamo gli "oggetti" che fanno parte del nostro Documento. Si parte dalle ***Tabelle***, che sono quelle presenti nel nostro Db. Poi abbiamo le ***Ricerche***, che sono in pratica delle *Query*, cioè particolari selezioni che possiamo creare sui nostri dati aggregandoli come più ci conviene. Quindi i ***Formulari***, maschere di immissione e di modifica dei Dati. Infine i ***Rapporti***, da usare per la stampa. Cominciamo a caricare i Dati.

10.1 Tabelle

Un doppio click su una Tabella apre una nuova finestra dove è possibile inserire o modificare i dati in forma, appunto, tabellare. Quindi, ad esempio:

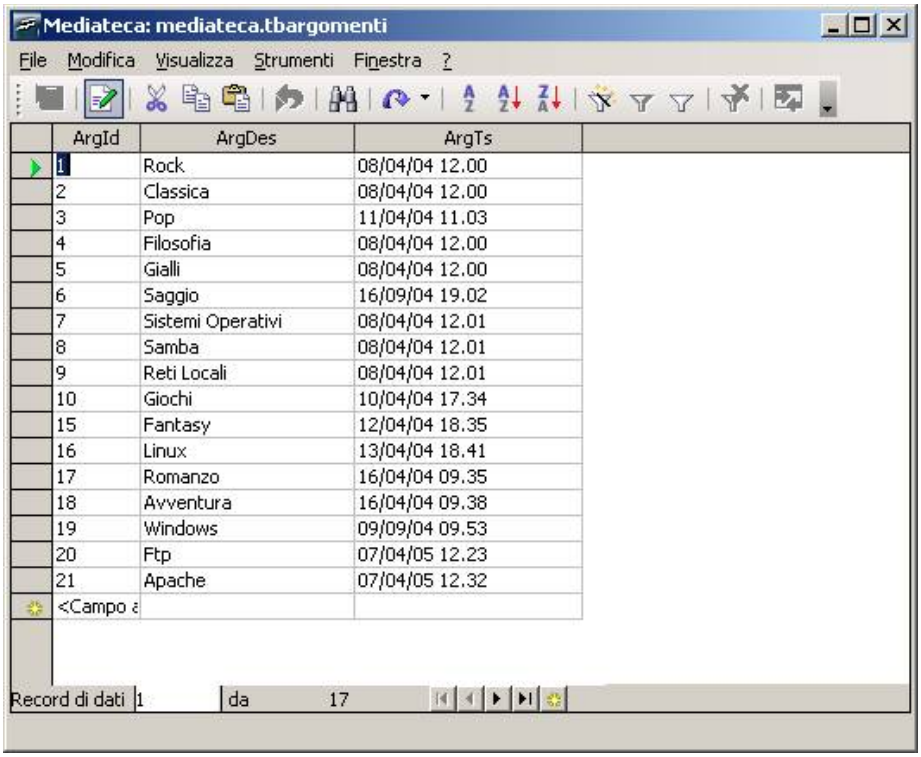


Figura 10.1.1 Modifica dei Dati di una Tabella

per comodità, nel seguito tutti gli esempi faranno riferimento ad un Db MySql collegato tramite ODBC. Nella gestione dei campi di tipo serial e del timestamp ci possono essere differenze notevoli, come abbiamo visto, tra i vari motori di Db.

In questo caso l'unica colonna che possiamo modificare è quella relativa ad *ArgDes*, perché *ArgId* è un contatore ad incremento automatico e *ArgTs* è un valore aggiornato dal sistema. Conviene, per poter meglio seguire gli esempi successivi, immettere alcuni valori (come quelli in figura) sia nella Tabella *tbargomenti* che in quella *tbsupporti*. Si noti come, man mano che si aggiungono delle righe, il sistema assegni il numero progressivo di *ArgId*, ed aggiorni con la data e l'ora corrente il timestamp *ArgTs*.

TIP

OOo però non impedisce la modifica dei valori nelle colonne *ArgId* e *ArgTs*, e quindi se per errore si immettono manualmente dati anche in queste colonne i risultati sono imprevedibili. Per questo motivo, in presenza di Campi di Tipo *Contatore* o *Timestamp* è opportuno usare metodi di modifica dei dati alternativi, come vedremo nel seguito.

In questa finestra è anche possibile ordinare i dati, effettuare ricerche ed impostare filtri. Il tutto è ben spiegato nella Guida di OOo, perciò andiamo avanti senza indugio mostrandovi alcuni dati di prova che potete caricare nella Tabella *tbsupporti*.

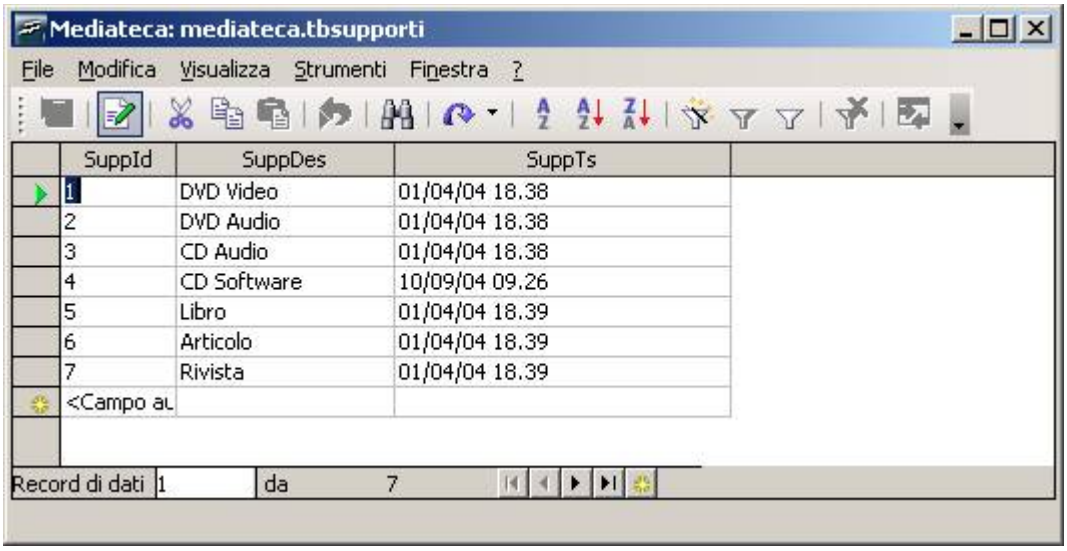


Figura 10.1.2 Dati della Tabella Supporti

10.2 Modifica della struttura di un Database

OOo permette anche di modificare la struttura delle Tabelle di un Database. Se infatti selezioniamo ad esempio la tabella *tbargomenti* e scegliamo, dal Menu contestuale richiamato dal tasto destro del mouse, la voce "*modifica*" avremo:

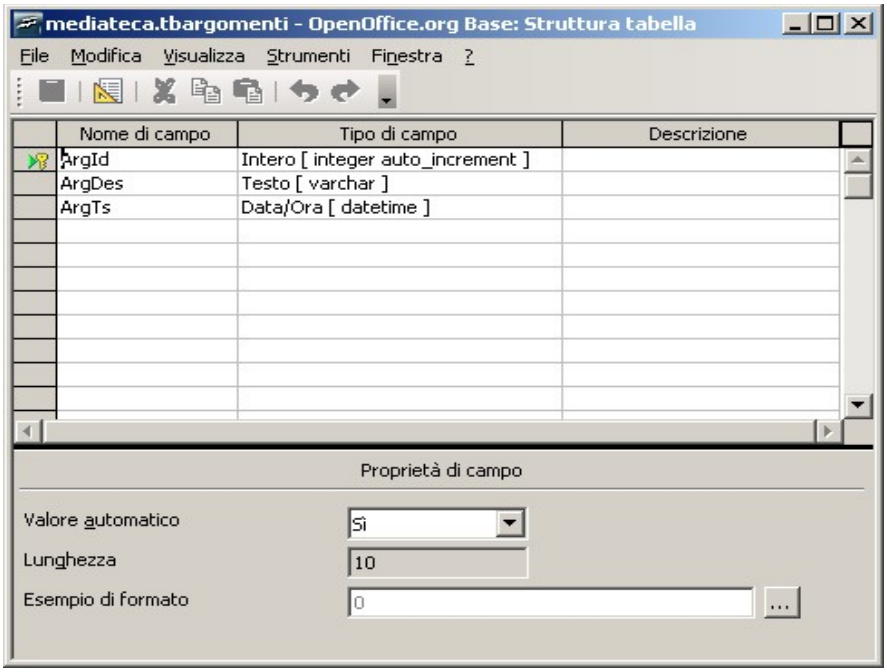


Figura 10.2.1 Modifica della struttura di una Tabella

Qui si può variare il nome, la natura ed altre caratteristiche dei campi. Dovendo prevedere la gestione di molti Database diversi, OOo cerca di adattarsi, ed il tipo di dati disponibile da assegnare ad un campo varia molto in funzione della sorgente dati.

Una pressione sull'apposito pulsante della barra sotto il Menu apre la finestra di gestione degli indici:

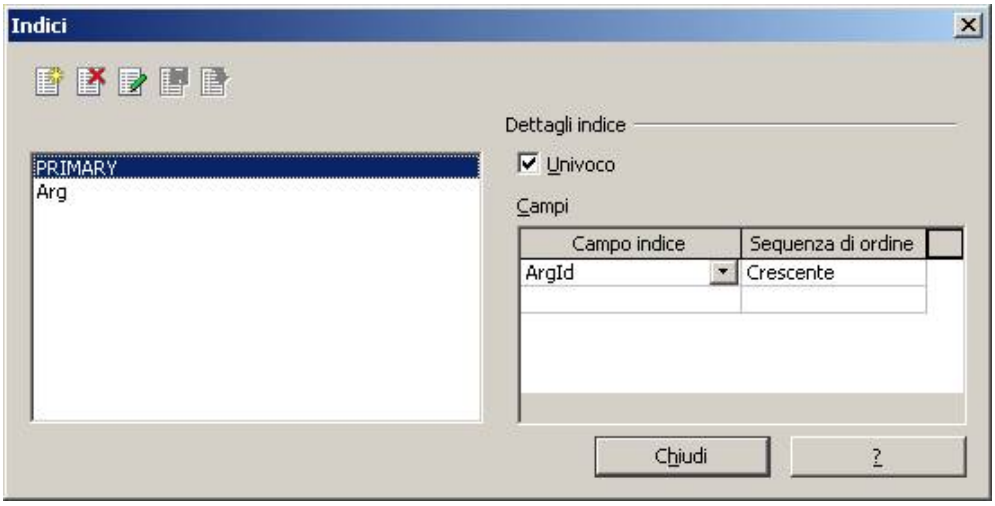


Figura 10.2.2 Modifica degli Indici

Il funzionamento è piuttosto intuitivo, quindi non ci dilungheremo sull'argomento. Tenete però presente che da questa finestra non è possibile assegnare una chiave primaria.

Infatti, SALVO EMERGENZE, è **assolutamente sconsigliato, se si usa un motore di Db esterno, creare o modificare tabelle direttamente da OOo**: la strada migliore è senza dubbio utilizzare gli strumenti messi a disposizione dai Db Server.

10.3 Ricerche

Il termine inglese sarebbe ***query***, ed infatti in altri programmi sono chiamate così. Però *query* è un po' generico, in quanto viene usato in molti contesti diversi, con significati non proprio identici. Quindi ***ricerca*** ci va bene.

Una *ricerca* è una particolare *vista* dei dati presenti in una o più tabelle del nostro Db. Si presenta come una nuova tabella, ma non deve necessariamente rispettare una struttura fisica esistente. Infatti alcune *colonne* possono essere frutto di calcoli, oppure *l'ordine* non è quello naturale della tabella di origine. Per fare un esempio, per motivi di comodità potremmo volere un elenco degli elementi compresi nella tabella *tbargomenti*, in ordine alfabetico. Uno dei modi è creare una *ricerca*. Questa struttura è anche utile per scartare dalle elaborazioni quei campi che non hanno una utilità immediata, come i *timestamp*. Alle ricerche è dedicato un'apposita voce nel pannello a sinistra del Modulo Base. Per ogni Documento possiamo creare una o più ricerche, a seconda delle nostre esigenze.

Per creare una nuova ricerca, abbiamo, in sostanza, tre opzioni elencate nella finestra *Attività*. Possiamo perciò:

- creare una Ricerca in *vista struttura*; questa è la modalità più comoda, perchè permette un controllo completo della struttura e delle proprietà della query

- usare una *procedura guidata*; per chi non è molto esperto, ma è utile solo per cose semplici
- creare una Ricerca in *vista SQL*; se siete in buoni rapporti con SQL....

Noi, anche per motivi didattici, useremo la prima opzione. Vogliamo, ad esempio, creare una *ricerca* che elenca le voci della tabella *tbargomenti* in ordine alfabetico. La prima cosa che ci viene richiesta è, appunto, di scegliere la tabella (o le tabelle) che faranno parte della nostra *ricerca*; scegliamo *tbargomenti*, ed avremo questo risultato:

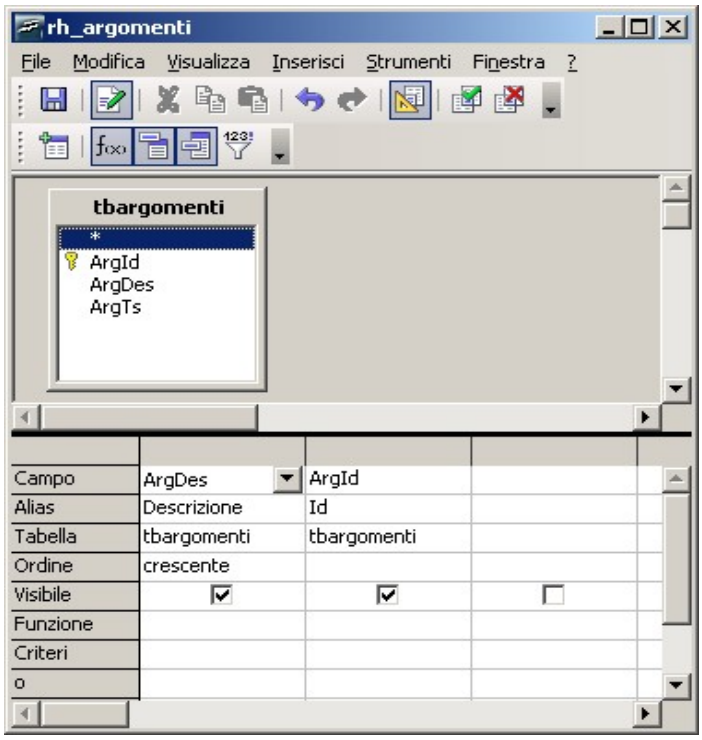


Figura 10.3.1: Struttura di una Ricerca

La parte superiore della finestra contiene le tabelle selezionate; qui, come vedremo, si possono definire anche le relazioni tra le tabelle. La parte inferiore riporta i *campi*, e le caratteristiche degli stessi, che dovranno "comporre" la nostra ricerca. Si può aggiungere un *campo* alla ricerca con un doppio click sul nome, nel pannello superiore. Si può trascinare un *campo* dalla tabella alla griglia della parte bassa. Infine si può selezionare il nome del campo dalla casella a discesa. I campi possono essere disposti in qualsiasi ordine, a seconda di come desideriamo vengano visualizzati. Per ogni campo è possibile selezionare, oltre al ***nome*** ed alla ***tabella*** di appartenenza : un ***alias***, cioè una intestazione di colonna che sarà usata nella visualizzazione; un ***ordine***, cioè uno o più criteri di ordinamento della ricerca; un flag di ***visibile***, che indica se il campo deve essere visibile; uno o più ***criteri***, cioè condizioni di selezione per il contenuto dei campi (su questo torneremo in seguito).

TIP

Siccome nel seguito useremo la stessa ricerca per costruire una casella a discesa per la selezione dell'argomento, è **OBBLIGATORIO** posizionare la Descrizione come prima colonna.

Dei pulsanti sulla barra sotto il menù, ***esegui ricerca***  , ci mostra un'anteprima del risultato delle nostre impostazioni ed è utile per verificare al volo la correttezza delle nostre scelte. Invece ***attiva/disattiva vista disegno***  è particolarmente importante, perché permette di visualizzare la ricerca con la sintassi del linguaggio SQL. In questo modo è anche possibile "spedire" ordini direttamente in SQL al motore di Database, per chi di voi è in grado di farlo. Nel nostro caso, sarebbe:

```
SELECT `ArgId` AS `Id`, `ArgDes` AS `Descrizione`
FROM `mediateca`.`tbargomenti` `tbargomenti`
ORDER BY `Descrizione` ASC
```

cioè in pratica: *SELEZIONA (elenco dei campi) DA (elenco delle tabelle) ORDINATO PER (nome di campo)*. Facile, no ?

Tecnica

SQL sta per Structured Query Language, ed è una sintassi definita anche da uno standard ANSI per la creazione, la modifica e l'interrogazione di basi di dati. Si basa su relativamente poche parole chiave o "istruzioni" (come SELECT) con una sintassi di solito semplice e comprensibile. Gli standard ANSI per SQL sono ANSI 92 e ANSI 99. Ovviamente ANSI 99 è molto più ampio del predecessore. MySql è sostanzialmente conforme ad ANSI 92, ma non implementa completamente ANSI 99. Inoltre esistono "estensioni" a SQL caratteristiche di MySql, che potreste non ritrovare in altri prodotti di Database. Questo degli standard è un problema annoso, ancora in parte non risolto, per cui esistono molti "dialetti" SQL, anzi tutti i produttori di Server di Db hanno una propria "versione" del linguaggio, ovviamente parzialmente incompatibile con le altre. Per quanto riguarda OOo, il programma "traduce", come abbiamo visto, le ricerche create col tool visuale in istruzioni SQL. Purtroppo non sempre queste istruzioni sono corrette per il Db a cui sono destinate, e quindi alcune ricerche potrebbero non funzionare come ci si aspetta. Fortunatamente è anche possibile "spedire" istruzioni SQL direttamente al motore Db, senza "traduzione" di OpenOffice, risolvendo così in parte i problemi.

Dopo aver salvato la *ricerca*, ad esempio col nome ***rh_argomenti***, la stessa comparirà nell'elenco delle *ricerche* disponibili. Un doppio click ci permette di "aprire" la ricerca, che appare più o meno così :

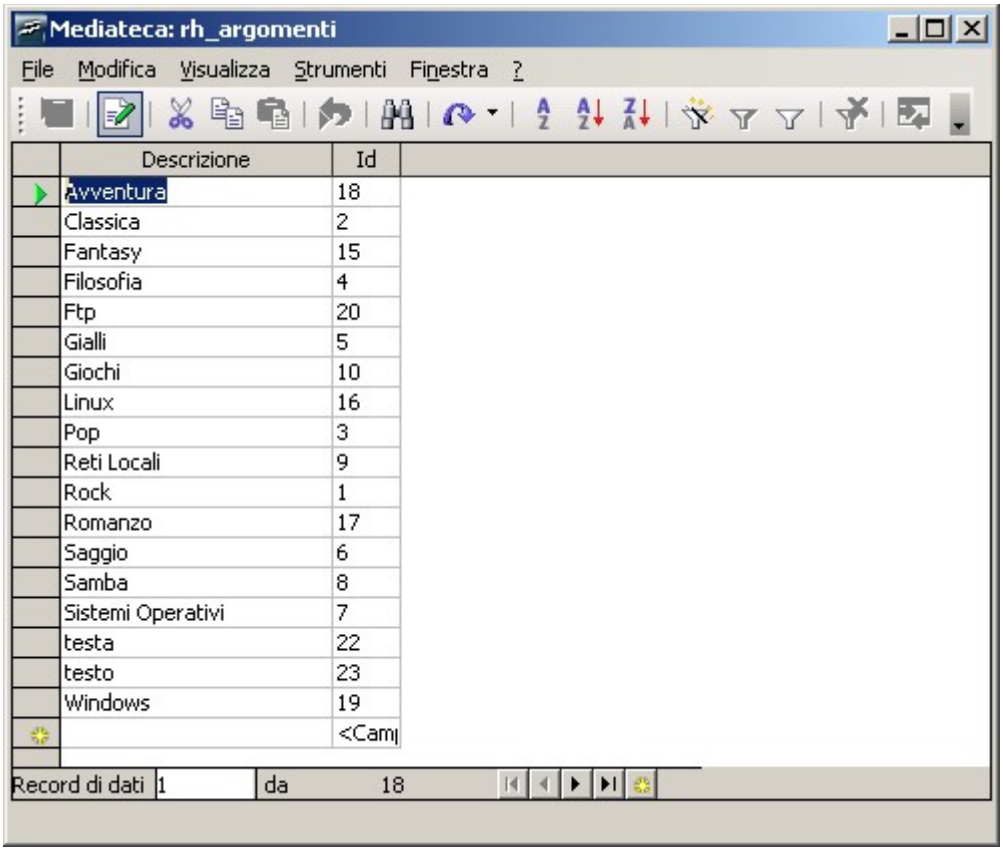


Figura 10.3.2 Una Ricerca

La finestra è molto simile a quella di gestione dei dati di una tabella, ed in effetti questo tipo di *ricerche* può proprio essere visto come una tabella "personalizzata". In questa finestra è anche possibile aggiungere nuove righe, o cancellarne alcune. Se il Database è stato ***registrato*** la *ricerca* comparirà anche nell'elenco a sinistra della finestra delle ***Sorgenti Dati*** richiamabile con ***Tasto F4*** dagli altri moduli di OpenOffice.

Con lo stesso procedimento sarà opportuno creare una ricerca simile anche per la tabella *Supporti*, perché ci sarà utile in seguito. Ricordate che dovrà contenere solo la *Descrizione* e *l'Id*, dovrà essere ordinata per Descrizione ed avere il nome di ***rh_supporti***.

10.4 Com'era... com'è...

Per chi ha già usato l'interfaccia ai Db con la versione 1.X.X di OOo, l'aggiornamento alla 2.X modifica la filosofia di gestione delle basi di dati. In precedenza le *sorgenti dati* erano essenzialmente di supporto agli altri moduli della suite, mentre questa volta si cerca di dare un po' di vita propria al modulo ***base***. Questo diventa particolarmente chiaro se diamo uno sguardo al pannello delle "sorgenti dati" di OOo 1.X.X, e cioè

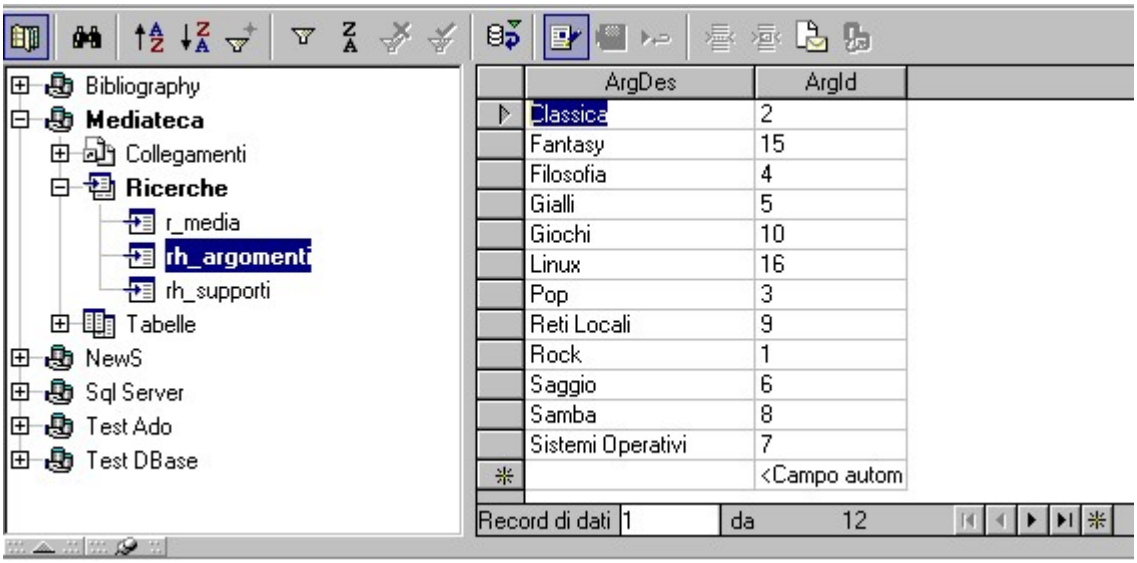


Figura 10.4.1 Sorgente Dati della 1.X.X

e lo confrontiamo con quello della 2.X:

Figura 10.4.2 Sorgente Dati della 2.0

Nella 1.X.X ogni sorgente dati prevede **tre** elementi, cioè le *Tabelle*, le *Ricerche*, ed i *Collegamenti*. Un formulario (una scheda) od un rapporto sono documenti "esterni", che possono essere *collegati* alla sorgente dati. Nella 2.0 sono spariti i *collegamenti* perchè formulari e rapporti sono elementi di un nuovo tipo di documento, chiamato ***base***.

Non solo: il documento di tipo ***base*** è portatile, cioè può essere trasferito con (speriamo) facilità da un PC all'altro, e può essere semplicemente "registrato" come sorgente dati di OOo con la voce di menu *Strumenti -> Opzioni -> OpenOffice.org Base -> Database*. Per questo motivo in OOo 2.0 manca la gestione diretta delle Sorgenti Dati.

Tutto il resto rimane identico, comprese le varie modalità di inclusione di dati esterni nei vari moduli della suite.

11. La Mediateca

11.1 Filosofie a confronto

Quando si parla di interfacce utente ognuno ha una opinione diversa dagli altri, e le discussioni si sprecano. C'è chi ad esempio ritiene che, per un programma gestionale, il mouse sia solo di impaccio e la tastiera rende il lavoro migliore; c'è chi preferisce avere il desktop ingombro di icone e programmi, chi sceglie la pulizia assoluta e quindi focalizza un compito alla volta. Siccome il nostro argomento sono i Database, fortunatamente c'è poco da scegliere: *tabella o scheda ?*

Se scegliamo un archivio qualsiasi, la vista a **tabella** (tabellare, nella terminologia di OOo) è simile a quella di un foglio elettronico, con a video una lista di *record* (cioè righe) e la possibilità di scorrere l'archivio di molti record alla volta. La vista a **scheda** (a righe o a colonne), più tradizionalmente, mostra appunto una scheda alla volta ed i campi sono sistemati sul video in modo da facilitare l'immissione dei dati. Scegliere una o l'altra modalità di interazione con l'utente dipende, appunto, dai gusti personali, ed anche dalla quantità e qualità dei dati da gestire. La *tabella* consente di avere a schermo un numero elevato di record, e facilita gli spostamenti nell'Archivio e l'individuazione di righe specifiche, se ad esempio l'archivio è ordinato. La *scheda* rende agevole l'immissione e la variazione dei dati di un *singolo* record.

Io uso solo la vista tabellare se i record da gestire sono pochi; invece, come vedremo, una combinazione tra le due per archivi complessi. Ma andiamo con ordine.

11.2 Il Formulario

Cominceremo col creare dei piccoli moduli per la gestione delle tabelle "di contorno" del nostro Database, cioè ***tbargomenti*** e ***tbsupporti***. Nonostante si potrebbe facilmente "popolare" di dati queste tabelle, vista la loro semplice struttura, direttamente dalla sezione *Tabelle* del Documento, costruiremo dei formulari appositi per esse. Dovremo aver già creato delle *Ricerche* per le due tabelle, secondo quanto già spiegato nei paragrafi precedenti. In particolare, le due ricerche comprenderanno solo i campi *Id* e *Descrizione*, e saranno ordinate appunto per *Descrizione*.

Il ***Formulario*** è una particolare *vista* dei nostri dati che dovrebbe permettere la gestione delle informazioni nel modo più semplice possibile. In altri prodotti di Db la stessa entità è chiamata anche *Scheda*, tanto per essere chiari. Attraverso un *Formulario* è possibile visualizzare e modificare i dati contenuti nelle *Tabelle* che fanno parte del nostro Archivio. Nel Modulo *OOo Base* esiste una funzione di auto composizione dei Formulari, ma qui preferiremo la procedura manuale, che è anche più didattica.

Perciò dalla sezione di sinistra scegliamo *Formulari* e quindi in alto, tra le Attività, *Crea formulario in vista struttura*. Si apre una bella finestra vuota, opportunamente suddivisa con una griglia, dove potremo dare il giusto sfogo alla nostra creatività...

Se non sono visibili, con la voce di Menù *Visualizza -> Barre dei Simboli* scegliamo di rendere esplicite la *Struttura del Formulario* :

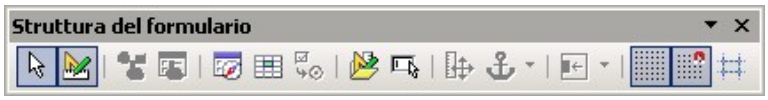


Figura 11.2.1 La Barra "Struttura del Formulario"

e la *Controlli per Formulario*:

Figura 11.2.2 La Barra "Controlli"

che scopriremo essere molto utili.

Non vogliamo in questa fase aiuti automatici, quindi ***disattiviamo il pilota*** (ultimo pulsante a destra della seconda riga dei *Controlli per Formulario*). Siccome intendiamo usare un'interfaccia di tipo tabellare, scegliamo lo strumento ***Campo di controllo Tabella*** dai *Controlli per Formulario,* e tracciamo un bel rettangolo sul nostro foglio bianco. Otterremo più o meno questo:

Figura 11.2.3 La Tabella appena creata

Bene, abbiamo creato la base per la nostra tabella. Siamo in modalità ***bozza***, in pratica stiamo *disegnando* la maschera, e ciò è anche indicato dall'attivazione dell'apposito pulsante sulla barra delle funzioni del formulario ( ).

In modalità *bozza* è possibile cambiare l'aspetto e le caratteristiche del nostro documento. Ogni ***elemento*** del formulario ha delle ***proprietà*** che possono essere modificate secondo le nostre necessità. Ad esempio, selezionando la tabella creata, col tasto destro del mouse possiamo scegliere la voce ***Formulario...*** per stabilire per prima cosa quali dati vogliamo gestire. Nella finestra delle proprietà, nella tab ***Generale*** assegniamo il nome del formulario (*Argomenti*); nella tab ***Dati*** dobbiamo specificare i parametri di collegamento come in figura:

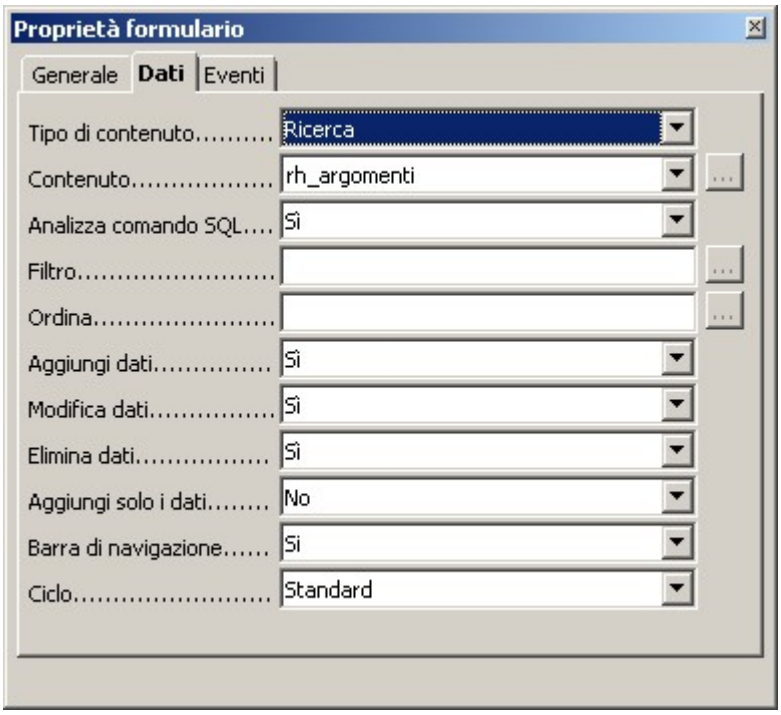


Figura 11.2.4 Le proprietà del Formulario

Quindi: *Tipo di contenuto* ***ricerca***, *contenuto* ***rh_argomenti*** (che è il nome della ricerca). Bene, ora dobbiamo *popolare* la nostra tabella, quindi posizioniamo il cursore sul bordo superiore della tabella stessa e col tasto destro del mouse scegliamo ***Inserisci colonna*** e quindi ***Campo di testo***. Siccome ci servono due colonne, ripetiamo l'operazione due volte, ottenendo questo risultato:

Figura 11.2.5 Le due colonne del Formulario

Dobbiamo ora *personalizzare* le colonne, cioè fare in modo che contengano i dati che ci servono. Selezioniamo *Colonna1* ed ancora col tasto destro del mouse scegliamo la voce ***Colonna...***. In questo modo si apre la finestra delle proprietà della colonna. Il primo campo da visualizzare per la nostra tabella è l'identificatore, valore numerico intero assegnato dal motore di Db, quindi nel Tab ***Generale*** possiamo immettere questi valori:

Figura 11.2.6 Le proprietà della Colonna 1 (Id)

Quindi: *Nome della colonna* ***Id***, *Titolo visualizzato* ***Id,*** *Abilitato* ***No***, *Sola lettura* ***Sì***. Questi due ultimi valori fanno in modo che il campo non sia modificabile. Se volete informazioni sulle altre opzioni presenti, la pressione del tasto **F1** vi porta all'ottima guida in linea di OOo. Nella Tab ***Dati*** dobbiamo indicare quale campo desideriamo sia visualizzato sulla colonna, quindi nel *Campo di Dati* selezioniamo ***Id*** dalla casella a discesa.

Figura 11.2.7 I Dati collegati alla Colonna Id

Passiamo ora alla colonna di descrizione. Quindi *Colonna2*, tasto destro, ***Colonna...*** ed impostiamo, nella tab **Generale**, *nome* ***Descrizione***, *Titolo* ***Descrizione***, *Abilitato* ***Sì***, *Sola Lettura* ***No***, perché il campo deve essere modificabile. Nella Tab **Dati**, *Campo di Dati* ***Descrizione***. Siccome la colonna di *Descrizione* è un po' stretta, allarghiamola trascinandola per il bordo, proprio come faremmo con una colonna di un foglio elettronico. Benissimo, siamo alla fine, una bella pressione sul pulsante ***bozza***, et voila:

Id	Descrizione
18	Avventura
2	Classica
15	Fantasy
4	Filosofia
20	Ftp
5	Gialli
10	Giochi
16	Linux
3	Pop
9	Reti Locali
1	Rock
17	Romanzo
6	Saggio
8	Samba
7	Sistemi Operativi
19	Windows
automatico>	

Record di dati 1 da 16

Figura 11.2.8 Il nostro primo formulario

Abbiamo creato il nostro primo formulario. Tutto qui ? Ma non è la stessa cosa della Tabella o della Ricerca ?? In effetti così sembra, ma NON E' AFFATTO la stessa cosa. Ci sono molte importanti differenze, e ne cito ora solo due, tanto per cominciare. Primo, i formulari sono completamente personalizzabili per quanto riguarda la formattazione, e questa non viene persa una volta chiuso il documento. Secondo, in un formulario si possono specificare una serie di parametri (come il "sola lettura") che nelle Tabelle o nelle Ricerche non sono applicabili.

TIP

Date uno sguardo alla colonna *Id* del nostro formulario. Abbiamo definito l'Id come un intero ad incremento automatico. Allora, perché alcuni numeri (ad es. 11) non compaiono ? Perché se si cancella una riga, il numero assegnato non viene più riutilizzato; quindi se cancello il record 11, il motore riparte dal 12.

Tornando in modalità *bozza*, è possibile variare molti parametri relativi alla visualizzazione dei dati, quindi possiamo abbellire il formulario secondo i nostri gusti. Trattandosi un pratica di una specie di documento di testo, ad inizio pagina possiamo anche assegnare un bel titolo, magari *Tabella degli Argomenti*, e salvare. Non ci resta che preparare un formulario anche per la Tabella *Supporti*, che dovrebbe apparire, alla fine, più o meno così:

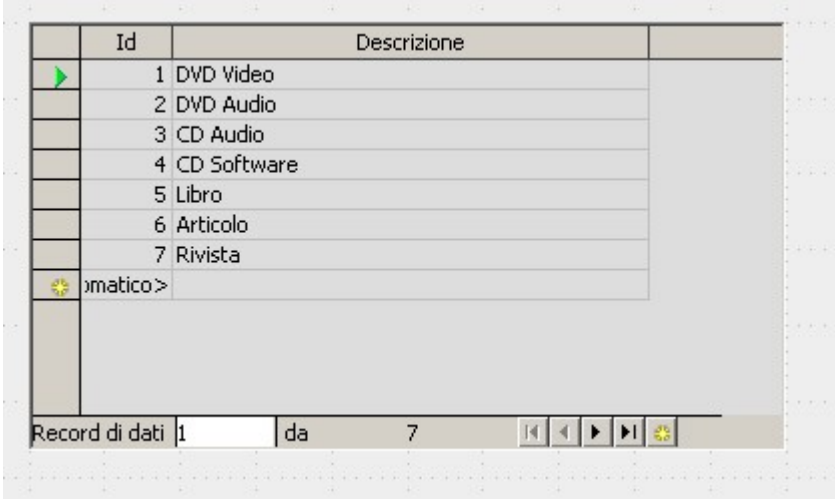


Figura 11.2.9 Formulario per la Tabella Supporti

In questo caso ho usato il ***pilota automatico***, che è raggiungibile alla voce *File -> Pilota Automatico -> Formulario*; questa auto composizione rende abbastanza lineare la creazione di formulari non molto complessi, e può essere usata senza problemi se si ha fretta. Inoltre nulla vieta di modificare in seguito i formulari generati dall'auto composizione secondo le nostre preferenze.

TIP

Il passaggio dalla modalità di modifica della struttura del formulario alla modalità dei gestione dei dati avviene sempre con la pressione del pulsante *bozza*. E' l'equivalente del passaggio in Ms Access da vista "struttura" a vista "scheda", per quelli di voi che hanno confidenza col prodotto Microsoft.

Forse è il caso di ricapitolare i vari passaggi che ci hanno portato fin qui:

- Abbiamo creato la struttura dei nostri archivi con *MySql Administrator*, nell'Archivio *Mediateca* di un Server MySql
- Abbiamo generato un *DSN* (Data Source Name) per la gestione dell'archivio tramite ODBC
- Abbiamo creato un nuovo documento di tipo "Base" collegato con il DSN e lo abbiamo registrato come *Sorgente Dati* di OOo
- Abbiamo creato due ***Ricerche*** relative alle tabelle *tbargomenti* e *tbsupporti*, scartando il campo di tipo *timestamp* ed ordinate per *Descrizione* (*rh_argomenti* e *rh_supporti*)
- infine abbiamo creato due ***Formulari*** per la gestione delle due tabelle.

A questo punto nel Pannello *Formulari* del nostro Documento, saranno presenti due voci; un doppio click apre la scheda per permettere l'immissione o la modifica dei dati. Il tasto destro apre un Menu contestuale che permette, tra le altre cose, di modificare la struttura della Scheda.

11.3 Il Formulario per la Mediateca

Siamo arrivati al momento di costruire il modulo di gestione del nostro archivio principale, cioè ***tbmedia***. Per spiegare meglio le varie possibilità disponibili, potremmo usare *due tipi di formulario diversi*: il primo più semplice, basato su una struttura a tabella simile a quelle già viste. Il secondo più complesso, che mostra una struttura mista scheda / tabella, più adatta all'immissione dei dati.

Cominciamo col creare una nuova ricerca (***r_media***) che sarà la base del nostro formulario. Includiamo ovviamente solo i campi che ci interessano, quindi :

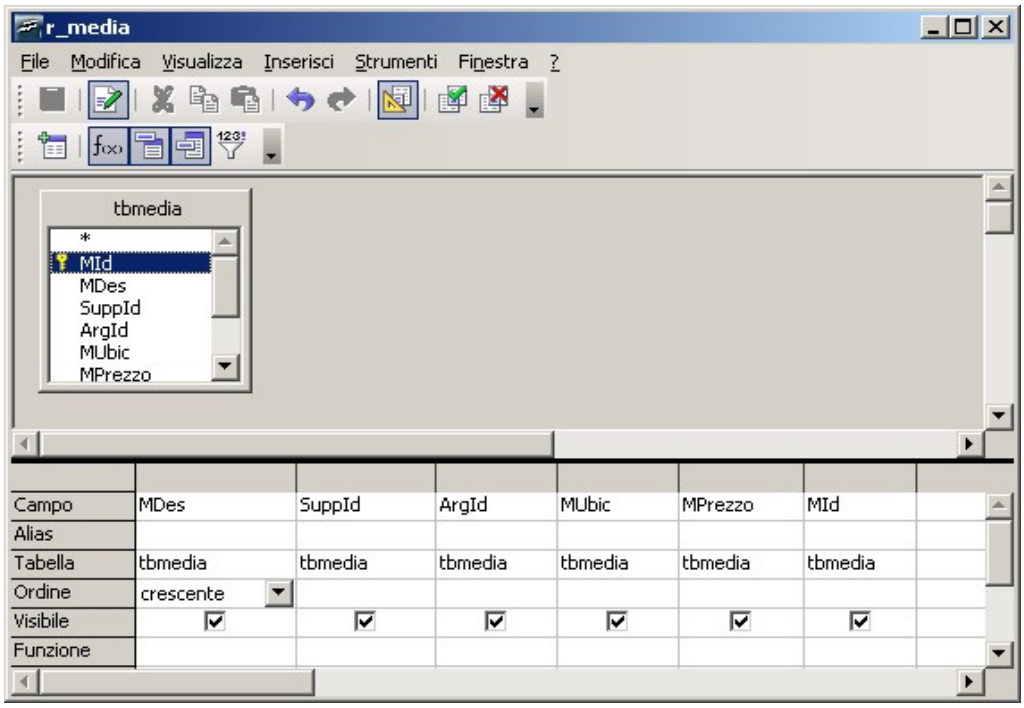


Figura 11.3.1 La Ricerca per la Mediateca

Abbiamo scelto di ordinare la ricerca per descrizione, e di includere anche il campo *MId* (la chiave primaria). A rigore questo campo (intero ad incremento automatico) non era necessario, però ***se non si include la chiave primaria le ricerche di OOo risultano di sola lettura***, cioè non si possono modificare i dati. I Campi *MTs* (cioè il timestamp) e *MNote* in questa fase non ci interessano. Creata la ricerca, generiamo il formulario seguendo le procedure che abbiamo già visto (manualmente o col pilota automatico, fate voi). Dovremmo ottenere più o meno questo risultato:

Descrizione	Supporto	Argomento	Ubicazione	Prezzo
Bad Boys II	4	18		€ 15,00
Benni - La compagnia dei celestini	5	17		€ 0,00
Cornwell - L'Ultimo Distretti	5	5		€ 0,00
Guccini - Ritratti	3	3		€ 0,00
Il Signore degli Anelli	5	18		€ 0,00
Joss Stone - The Soul Session	3	3		€ 0,00
KDE 3.4 - Novità Major Release	7	16	Linux & C. - 38	€ 15,00
Montalbano - Il Ladro di merendine	3	3		€ 0,00
Montalbano - La forma dell'acqua	5	5		€ 0,00
Pearl Jam - Ten	2	1		€ 0,00

Record di dati 11 da 11

Figura 11.3.2: La Tabella di Mediateca

Questo tipo di formattazione è stato semplicemente ottenuto sfruttando le proprietà delle varie colonne, ed adeguando la larghezza alle nostre esigenze. Notate che, ad esempio, la colonna *Prezzo* può essere formattata come *euro* modificando la proprietà *formattazione*.

Per un corretto funzionamento del formulario è necessario impostare,nelle proprietà delle colonne *Tipo Supporto*, *Argomento*, *Prezzo*, la proprietà ***Valore Standard*** a ***zero*** (0), per fare in modo che venga assegnato in automatico un valore. Comunque ancora non ci siamo.

Infatti immettere a mano il valore numerico di corrispondenza per la Tabella *Tipo Supporto* e *Argomento* certo non è accettabile. Sarebbe meglio usare una casella a discesa, da cui selezionare in campo che ci interessa. Dobbiamo perciò cambiare ancora qualcosa. Torniamo in modo *bozza*, selezioniamo ad esempio la colonna *Argomento* e con il tasto destro del mouse scegliamo ***Sostituisci con...*** e poi ***Casella di riepilogo***. Sempre col tasto destro selezioniamo ***Colonna...*** e nella tab dati impostiamo questi valori:

Figura 11.3.3 Colonna di tipo Casella di riepilogo

Con questa impostazione, stiamo dicendo ad OOo che desideriamo compilare il Campo Dati *ArgId* attraverso il riferimento ad una ricerca (*rh_argomenti*), ed il campo collegato (cioè il valore da trasferire al campo dati) è il ***SECONDO*** della ricerca (cioè, appunto, ***ArgId***). Infatti i campi della ricerca che desideriamo collegare sono ***numerati a partire da zero***, e quindi 0 rappresenta ***ArgDes***. In verità questa impostazione non è molto logica, ed anche un po' limitata, perché ad esempio nella casella a discesa potrò mostrare sempre solo una colonna. La regola è dunque che, a seconda del valore impostato in "campo collegato", nella casella a discesa sarà mostrata la colonna A SINISTRA del campo scelto, cominciando a contare le colonne da zero. Per cui siccome nella mia ricerca ho solo due colonne *ArgDes* ed *ArgId*, sarà mostrata *ArgDes* e sarà trasferito il valore di *ArgId*, cioè appunto quello che volevamo ottenere. Adesso dovrebbe essere anche chiaro perché abbiamo posizionato *ArgDes* come prima colonna della ricerca. Questo meccanismo non è molto ben spiegato nella guida in linea, e si ci arriva con un po' di tentativi. Comunque alla fine tutto si risolve, e, applicando la stessa procedura anche a *Tipo Supporto*, avremo :

Descrizione	Supporto	Argomento	Ubicazione	Prezzo
Bad Boys II	DVD Video	Avventura		€ 15,00
Benni - La compagnia dei celestini	Libro	Romanzo		€ 0,00
Cornwell - L'Ultimo Distretti	Libro	Giallo		€ 0,00
De Gregori - Pezzi	CD Audio	Rock		€ 0,00
Guccini - Ritratti	CD Audio	Pop		€ 0,00
Il Signore degli Anelli	DVD Video	Avventura		€ 0,00
Joss Stone - The Soul Session	CD Audio	Pop		€ 0,00
KDE 3.4 - Novità Major Release	Rivista	Linux	Linux & C. - 38	€ 15,00
Montalbano - Il Ladro di merendine	CD Audio	Pop		€ 0,00
Montalbano - La forma dell'acqua	Libro	Giallo		€ 0,00
Pearl Jam - Ten	CD Audio	Rock		€ 0,00

Record di dati 1 da 11

Figura 11.3.4 il Formulario nella versione finale

Abbiamo costruito un formulario per la nostra mediateca che:

- è ordinato per descrizione, quindi la ricerca di un titolo è agevole

- ci permette di caricare i dati in modo comodo, con caselle a discesa per i campi che fanno riferimenti ad altre tabelle
- è facilmente navigabile

Possiamo, anche in questo caso, scrivere un bel titolo per il documento (ad es. MEDIATECA) e salvare il formulario col nome di ***f_media***.

11.4 Qualche informazione in più

OOo ha altre interessanti caratteristiche quando adoperiamo un formulario. Innanzi tutto vi sarete accorti che il *Campo di Controllo Tabella* ha alla base una piccola barra degli strumenti.

Figura 11.4.1 Barra degli strumenti alla base della tabella

Questa barra è del tutto simile a quella che compare nelle schede di Ms Access, e serve a navigare nell'archivio. L'ultimo pulsante sulla destra permette anche l'immissione immediata di un nuovo record. Più interessante è, però, la barra dei pulsanti che compare alla base della finestra di OpenOffice quando usiamo il formulario.

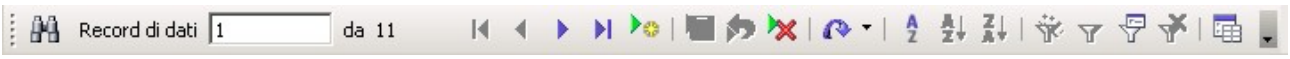


Figura 11.4.2 Barra degli strumenti alla base del Formulario

Oltre ai pulsanti già visti, questa barra offre una serie di altre possibilità. In primo luogo il pulsante ***aggiorna***, che permette di "ricaricare" i dati del formulario, utile in caso di aggiornamenti contemporanei da più stazioni di lavoro. Poi i pulsanti di ordinamento, non molto sfruttati: con i formulari basati su ricerche ordinate, infatti, non funzionano. Il più utile è senz'altro quello di ***Filtro Automatico***, che permette di selezionare molto velocemente i dati sulla base del contenuto di un campo. Funziona così: basta portare il cursore sul campo che contiene il valore da selezionare, e premere il pulsante. Ad esempio, se voglio selezionare tutti i CD Audio del mio archivio, posiziono il cursore sul campo *supporto* in una riga che contiene "Cd Audio" ed applico il filtro automatico. OOo mi mostrerà tutti e solo i Cd Audio. Per tornare alla situazione precedente, basta usare il pulsante ***Rimuovi filtro/ordine***, che si trova tre posizioni più a destra. Questa barra prevede molte altre possibilità, e vi consiglio di approfondire l'argomento con l'aiuto della Guida di OpenOffice.

12. Uso dei Formulari

Nel Capitolo precedente abbiamo visto come costruire un semplice Formulario tabellare per l'immissione dei dati della Mediateca. In particolare abbiamo costruito delle Maschere di immissione di tipo tabellare per i Supporti, gli Argomenti e gli elementi della Mediateca. Ora invece, costruiremo un documento un po' più complesso, formato da due formulari dedicati ognuno ad uno scopo, ma collegati tra loro attraverso dei pulsanti. In particolare, il primo servirà a consultare l'archivio, ed il secondo a modificare un singolo record. Vi ricordo che il tutto deve intendersi in un'ottica didattica, quindi sarebbe inopportuno, da parte vostra, ritenere troppo ingenue alcune scelte. Inoltre questo esempio ci farà conoscere ed apprezzare alcune insospettabili caratteristiche di OOo che la documentazione ufficiale riesce a celare in modo quasi perfetto (devo dire che in questo gli estensori dell'Help di OOo sono degli autentici maestri...).

12.1 Partiamo dalla fine

Credo sia meglio, per essere più chiari, partire dal risultato finale. Allora, ci proponiamo di ottenere un formulario che, all'apertura, ci mostri qualcosa del genere :

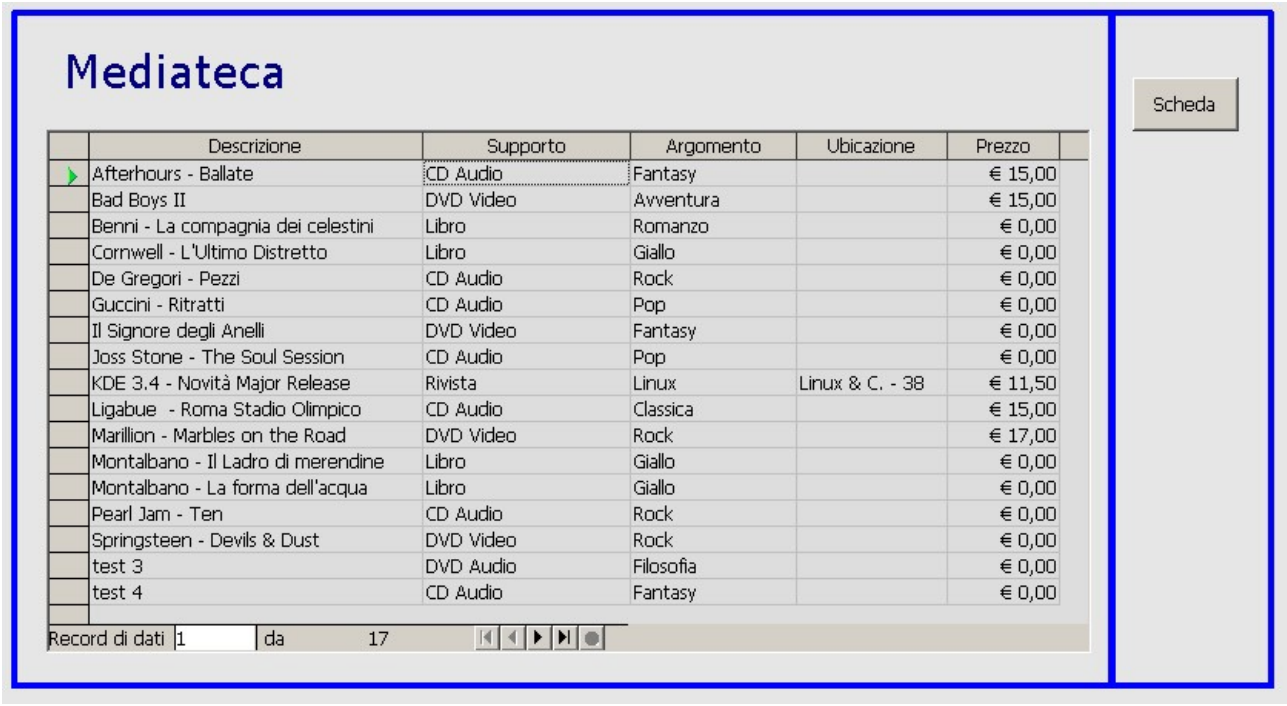


Figura 12.1.1 La Tabella

Si tratta di un Formulario tabellare di sola lettura che ci permette di consultare l'Archivio ordinato per descrizione e di impostare filtri sul Tipo di Supporto e sull'Argomento attraverso i pulsanti presenti sulla barra che OOo mostra in basso sullo schermo. Si noti che in questa

maschera non è possibile modificare alcun dato dell'Archivio. Alla pressione del Pulsante "Scheda", si passa a questa maschera :

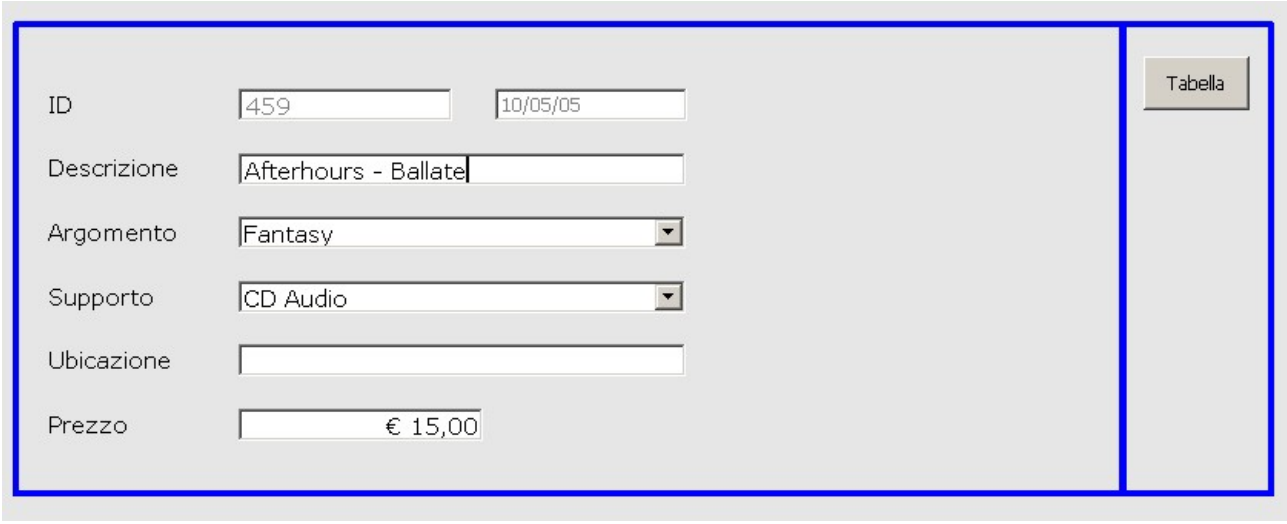


Figura 12.1.2 La Scheda

Qui invece è possibile modificare i Dati della scheda in una maniera più comoda, secondo uno schema sequenziale. Il Pulsante Tabella riporta alla maschera precedente.

Ora guarderemo da vicino le peculiarità di ogni maschera, soffermandoci sui dettagli importanti e tralasciando le cose più ovvie che abbiamo già descritto nella prima parte.

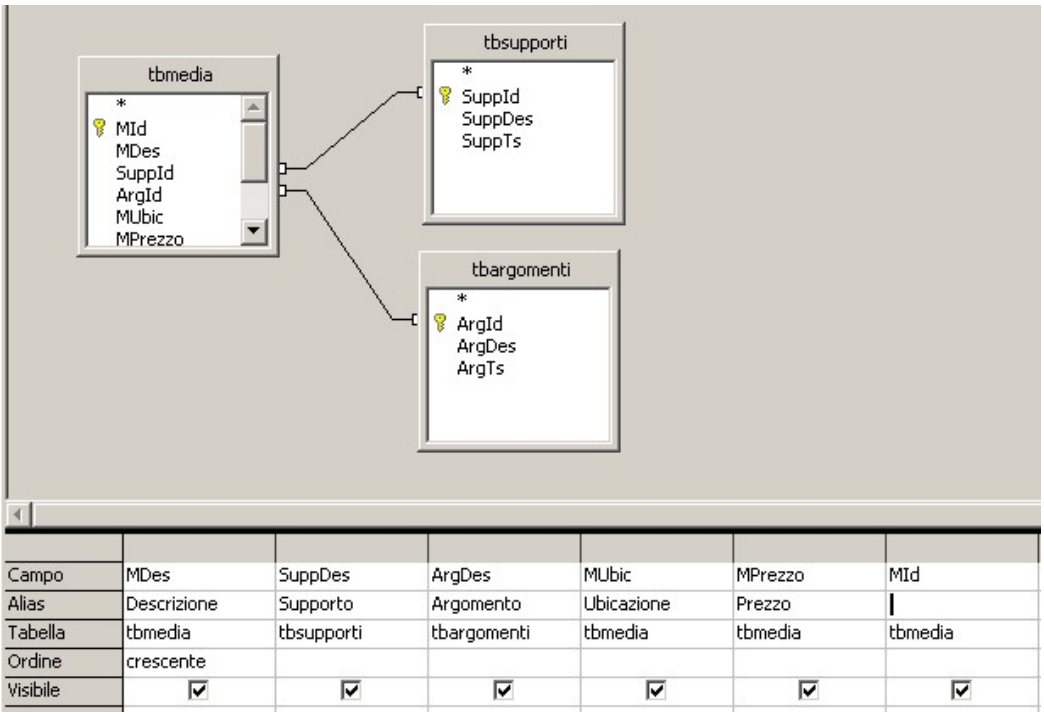
12.2 La prima parte – la Tabella

La prima cosa che ci serve è dunque una tabella che ci mostri i dati del nostro archivio principale, ma non solo. Deve essere anche "esplicativa", cioè "interpretare" i codici di ArgId e SuppId riportando la corretta descrizione. Infine deve essere di sola lettura. Cominciamo quindi a costruire una ricerca che faccia a caso nostro.

12.2.1 Una Ricerca più complessa...

In figura potete vedere una Ricerca che contiene campi provenienti da più tabelle. Dalla finestra di modifica delle *ricerche* è possibile aggiungere *tabelle* col pulsante "**Aggiungi Tabella**" in alto a sinistra. Per ottenere questo tipo di Ricerca bisogna anche dire ad OOo come "collegare" le varie tabelle; è, cioè, necessario indicare quali campi in comune il programma deve considerare per costruire la ricerca.

In pratica devo spiegare ad OpenOffice che *il campo SuppId della Tabella tbmedia corrisponde al campo SuppId della tabella tbsupporti, quindi, se il valore è ad esempio 3, la descrizione da riportare è 'Cd Audio'*. Il nome tecnico di questa operazione è "**join**": sto eseguendo un **join tra tabelle**.



Campo	MDes	SuppDes	ArgDes	MUbic	MPrezzo	MId
Alias	Descrizione	Supporto	Argomento	Ubicazione	Prezzo	
Tabella	tbmedia	tbsupporti	tbargomenti	tbmedia	tbmedia	tbmedia
Ordine	crescente					
Visibile	☑	☑	☑	☑	☑	☑

Figura 12.2.1 La ricerca per la prima Scheda del nostro Formulario

In OOo ottengo il risultato ***trascinando*** con il mouse il campo *SuppId* di *tbmedia* sul campo *SuppId* di *tbsupporti*. La stessa operazione devo fare per il campo *ArgId*. Quindi sceglierò i campi che dovranno apparire nella ricerca; notate inoltre che il campo Mdes è ordinato.

Tecnica

Stabilire delle *relazioni* in una *ricerca* non è ovviamente lo stesso che stabilire delle *relazioni* a livello di database. Queste ultime, come abbiamo visto, hanno la funzione principale di garantire l'integrità referenziale dei dati, e vengono archiviate in modo permanente nella struttura del database stesso. Una *relazione* impostata in una *ricerca* ha il solo scopo di recuperare informazioni coerenti da più tabelle, ma non ha alcun effetto sull'integrità dei dati.

Le *relazioni* possono essere "manipolate" semplicemente con un doppio clic sul collegamento tra le tabelle, e cancellate col menu contestuale attivato dal tasto destro. Inoltre ogni *relazione* può essere di tipo *interna*, *sinistra* o *destra*, e su questo argomento torneremo più avanti. Bene, salviamo la nostra ricerca col nome, ad esempio di ***r_media_2a*** e passiamo alla creazione del nostro formulario.

12.2.2 Un nuovo Formulario....

Potremmo iniziare daccapo, ma abbiamo una Scheda appena creata che può tornarci utile (*f_media*), e con una semplice operazione di copia ed incolla avremo la nostra nuova scheda col nome ***f_media_2***. E' necessario, ovviamente, modificare alcune cosucce, quindi sotto col lavoro.

Per prima cosa, dobbiamo dire ad OOo che, per questo nuovo documento, i dati di partenza sono nella ricerca ***r_media_2a***, quindi selezioniamo la tabella e col menu contestuale scegliamo *formulario*...

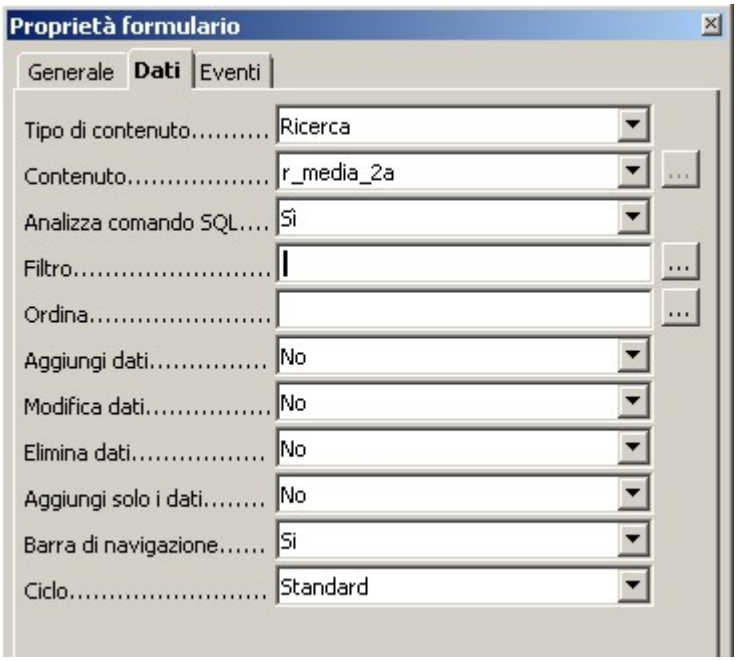


Figura 12.2.2 Le proprietà del formulario

Inoltre vogliamo che questa tabella sia di sola lettura, quindi settiamo a ***no*** le voci di *Aggiungi*, *Modifica* ed *Elimina Dati*. Poi non ci servono più le caselle a discesa per la scelta degli *Argomenti* e dei *Supporti*: selezioniamo una colonna alla volta, col tasto destro scegliamo ***sostituisci con -> campo di testo***, e nella finestra delle proprietà scegliamo come ***Campo di Dati*** rispettivamente *SuppDes* e *ArgDes*. Fatto.

Riepilogo e spiegazione : ci serve una semplice lista non modificabile dell'archivio dei *Media*, che però contenga anche alcune informazioni presenti in altre Tabelle (*Supporti* ed *Argomenti*); perciò abbiamo creato una ricerca che mette in relazione le tabelle e riporta le informazioni desiderate; rispetto al formulario precedente, in questo non servono più le caselle a discesa di selezione (perché di sola lettura) di *Argomenti* e *Supporti*, quindi abbiamo riportato le colonne a semplici caselle di testo, "agganciandole" ai campi giusti (*SuppDes* ed *ArgDes*); chiaro ? Spero di si....

TIP

In realtà impostare il formulario come "sola lettura" modificando le proprietà in questo caso non è necessario. In OOo, infatti, i formulari basati su ricerche contenenti più tabelle non sono comunque modificabili.

A parte i campi collegati ai dati, un formulario è un documento di testo, quindi con qualche abbellimento, potremo avere un risultato come questo:

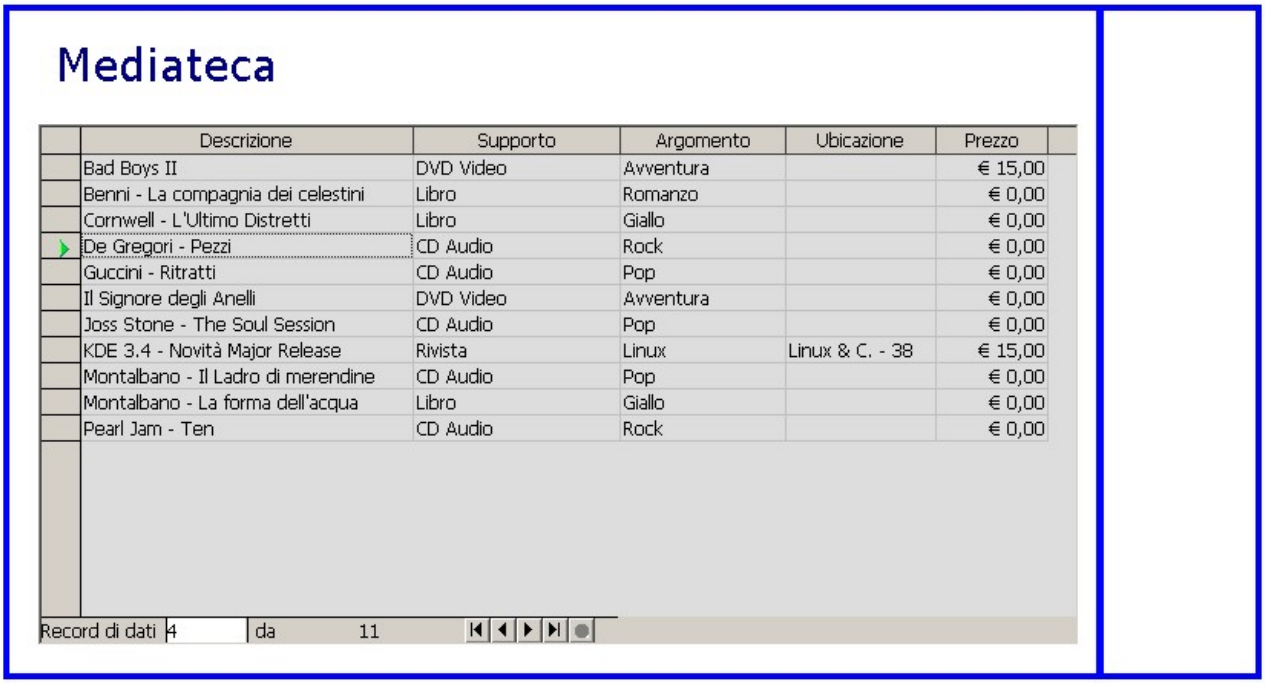


Figura 12.2.3 La prima Scheda per Mediateca

Selezioniamo la tabella appena creata, ed assegniamo ad esempio il nome ***F_Principale***.

12.3 La seconda parte - la Scheda

Bene, ora ci serve una Scheda per modificare i dati. Dobbiamo cioè aggiungere un altro *formulario* (un *sottoformulario*) collegato a quello principale. Per fare questo, cominciamo a prendere confidenza con una piccola finestra, quella del ***navigatore del formulario***. Questa finestra si attiva con l'apposito pulsante della Barra *struttura del formulario*, e nel nostro caso si presenta così :

Figura 12.3.1 Il Navigatore del Formulario

La finestra, al pari dell'omologa, ad esempio, dei Documenti di Testo, presenta la struttura del ***documento formulario*** e permette la navigazione veloce. Da qui si capisce subito che ogni *documento formulario* è in effetti composto da molti elementi diversi, che possono ben integrarsi tra loro. Per creare un sottoformulario, selezioniamo con il tasto destro la voce *MainForm* e, dal menu contestuale, scegliamo ***nuovo->formulario*** ed assegniamo il nome ***S1***, come in figura.

Figura 12.3.2 Il Sottoformulario S1

Ora per prima cosa dovremo dire a OOo quali dati dovrà gestire ***S1***, quindi apriamo le proprietà ed scriviamo nel campo "*contenuto*" il valore ***mediateca.tbmedia***, come in figura:

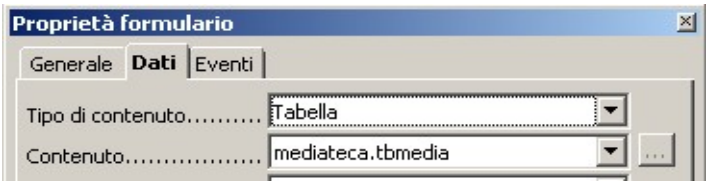


Figura 12.3.3 Proprietà Dati del Formulario S1

Da ora in poi, tutti i controlli associati ad ***S1*** saranno collegati ai dati della Tabella *tbmedia*.

La nostra intenzione è creare una nuova scheda, non tabellare, per modificare un singolo record. Avremo perciò bisogno di *Testi fissi* e *Campi di Testo*. Cominciamo a fare spazio; siccome un formulario è un po' come un documento di testo, con una nutrita serie di "invio" allunghiamo la nostra pagina. Quindi dalla barra *controlli per formulario* scegliamo ***testo fisso*** e tracciamo un quadrato come in figura:

Figura 12.3.4 Il controllo "testo fisso"

A differenza di altri software, OOo non permette la modifica diretta del testo, quindi dovremo usare la finestra delle proprietà. Inseriamo come titolo *Descrizione*.

Figura 12.3.5 Le proprietà del testo fisso

Allo stesso modo ancora dalla barra *controlli per formulario* scegliamo ***campo di testo*** e tracciamo un altro quadrato. Nelle proprietà del campo, come *nome* scegliamo *Descrizione*, e, nella tab *dati*, associamo il controllo al campo *Mdes*, come in figura:

Figura 12.3.6 Proprietà della casella di testo

In sostanza abbiamo appena creato il primo campo della maschera. Se usciamo dal *modo bozza*, dovremmo avere un risultato simile a questo:

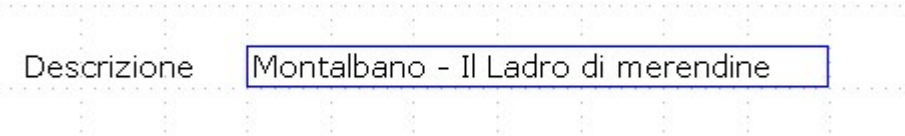


Figura 12.3.7 Campo completo

Tornando al modo bozza, il nostro navigatore del formulario dovrebbe apparire così :

Figura 12.3.8 Vista del Navigatore Formulario

Quindi ogni formulario ha i suoi campi, secondo una struttura ben definita. Ora bisogna procedere, e con un po' di "copia ed incolla" creiamo gli altri campi della scheda.

TIP

Nella versione di OOo che stiamo usando, per ottenere una corretta disposizione dei campi nella scheda, è necessario selezionare, per ogni elemento, la proprietà ***ancoraggio*** al valore ***pagina***, e questo PRIMA di eseguire qualsiasi "copia ed incolla". In caso contrario, il campo "incollato" apparirà in fondo alla pagina, ed alla riapertura del formulario avremo i campi messi alla rinfusa.

Con un po' di lavoro, comunque, dovremmo riuscire ad ottenere un risultato di questo tipo:

Figura 12.3.9 La maschera di immissione per Mediateca

Sarà però il caso di spiegare un po' meglio qualche dettaglio....

1. Il Campo ***Id***, siccome è un contatore automatico, deve essere definito, nella finestra delle proprietà, come "*abilitato* -> ***no***" e "*sola lettura* -> ***si***".
2. Nei campi di testo, come ***Descrizione*** ed ***Ubicazione***, va definita la *lunghezza max. del testo* all'effettivo numero di caratteri definito per il campo, perché OOo non associa questo valore automaticamente; quindi, ad esempio, per ***Descrizione***, *lunghezza max. del testo* -> ***100.***
3. ***Argomento*** è una Casella di Riepilogo, definita con : *Campo dei Dati* -> ***ArgId***, *Tipo del contenuto della lista* -> ***ricerca***, *Contenuto elenco* -> ***rh_argomenti***, *Campo collegato* -> ***1***, *Apribile* -> ***Si***; notate che se non si abilita questa ultima opzione, la lista non si apre (potremmo chiederci a cosa serve una casella a discesa che non si apre...); in modo simile è possibile definire il campo ***Supporto***, cambiando ovviamente l'origine dei dati a *rh_supporti*; Attenzione: nelle caselle a discesa è SEMPRE necessario specificare un valore quando aggiungiamo nuovi record, anche se sembra che OOo ne abbia già assegnato uno;
4. ***Prezzo***, è invece un *campo formattato*; come dice il nome, è in definitiva un campo di testo a cui è possibile associare una formattazione (in questo caso euro); possiamo anche assegnare una valore standard, in modo da non doverlo sempre digitare; la proprietà è appunto *valore standard* -> ***0***; OOo prevede anche un *campo valuta*, sostanzialmente simile, peccato non si possa allinearlo a destra...
5. il *campo alla destra di Id*, che contiene una Data, è collegato al Timestamp (***Mts***) della tabella; in effetti il Timestamp dovrebbe essere automaticamente aggiornato dal motore di Db, ma, nel caso di aggiunta di record, in OOo questo non accade; siamo costretti dunque ad aggiungerlo alla Scheda, definirlo come *campo data*, assegnare alla proprietà *Data Standard* un valore qualsiasi, in modo che sia assegnato una valore in automatico; in caso contrario OOo si rifiuta di aggiungere un nuovo Record; è inoltre opportuno scegliere anche *abilitato* -> ***no*** e *sola lettura* -> ***si***;

12.4 Completare il lavoro...

Bene, ora abbiamo due formulari, uno di tipo tabellare ed a sola lettura, uno di tipo a scheda con la possibilità di modificare i dati. Il nostro scopo è fare in modo che selezionando un Record nella Tabella, lo stesso sia visualizzato nella scheda. Dobbiamo cioè mettere in relazione i due formulari. Questo si ottiene selezionando il formulario ***S1*** nel Navigatore, e (nelle proprietà) premendo il pulsante accanto alla casella *Collega da...* o *Collega per...*. La finestra che si apre:

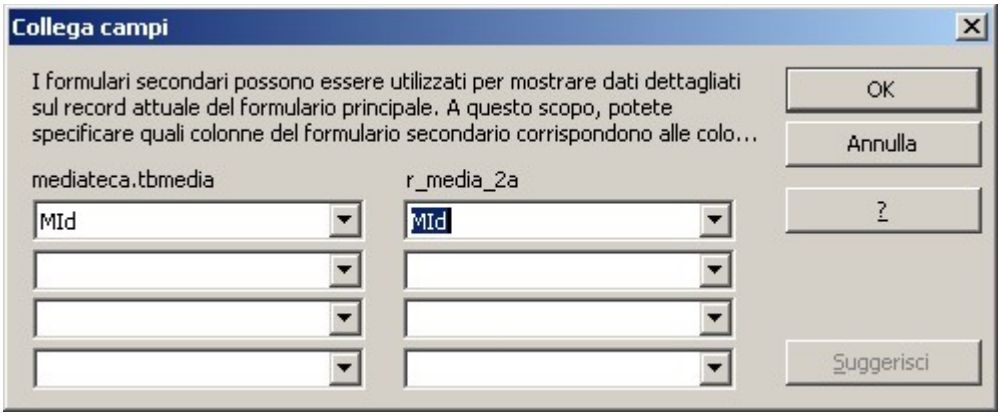


Figura 12.4.1 Collegamento tra formulari

permette di selezionare qual'è il campo che "aggancia" i due formulari; nel nostro caso ***MId***, ma in generale dovrebbe comunque essere un campo indicizzato in tutte e due le tabelle. Si noti anche che è possibile selezionare più di un campo, e questo torna utile in altre situazioni.

Bene, a questo punto scorrendo la tabella superiore OOo aggiorna anche la scheda inferiore, quindi è possibile "spostarsi" tra un formulario e l'altro senza problemi. Per *aggiungere un nuovo record*, basta selezionare un campo qualsiasi del formulario ***S1*** e scegliere nella barra di navigazione il pulsante *Nuovo record di Dati* per avere una Scheda su cui scrivere.

TIP

Quando si modifica la scheda, i cambiamenti non vengono riportati nella tabella. Per ottenere questo risultato è necessario utilizzare il pulsante ***aggiorna*** della *barra di navigazione del formulario*.

Scorrere però su e giù il documento però è piuttosto antipatico, quindi potremmo usare un paio di pulsanti per facilitare il lavoro. Ricordiamo allora che, in fondo, un Documento Formulario è anche un Documento di testo, quindi ne conserva tutte le caratteristiche. Impostiamo dunque, alla bisogna, due segnalibri. Il primo proprio all'inizio del documento, con la voce di Menu *Inserisci -> Segnalibri*, denominato *tabella*. Il secondo alla fine del documento, cioè alla base della seconda scheda, e dal nome, appunto, *scheda*. A questo punto accanto alla tabella inseriamo un pulsante (con l'apposito controllo) come in figura:

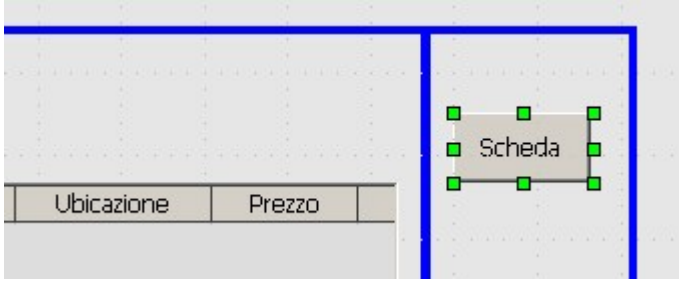


Figura 12.4.2 Il pulsante Scheda

Assegniamo le proprietà *Nome* -> ***B1*** e *Titolo* -> ***Scheda***. Quindi passiamo alla parte più interessante :

Figura 12.4.3 Proprietà del Pulsante Scheda

Al pulsante possiamo associare una "operazione" tra una serie di scelte predefinite; quella che ci interessa è *Apri Documento*, non certo per aprire un nuovo documento, ma perché vogliamo spostare il cursore sulla nostra scheda; ed infatti come URL si immette *#scheda* che è appunto il nome del segnalibro impostato in precedenza. Allo stesso modo aggiungiamo un pulsante *Tabella* alla scheda, come in figura:

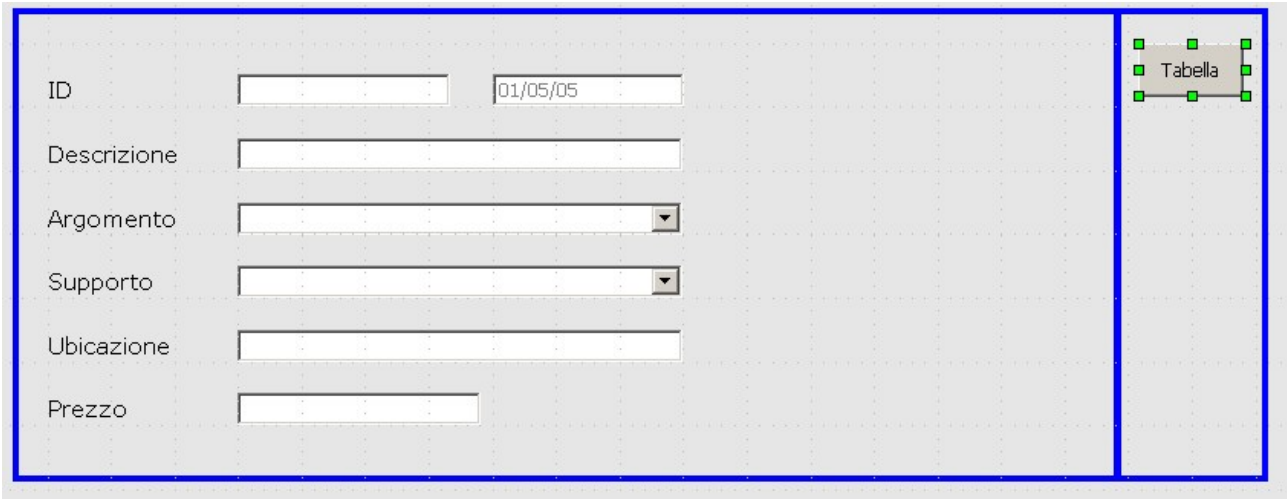


Figura 12.4.4 Il pulsante per la Scheda

Questa volta impostiamo la proprietà URL a *#tabella*, ed il gioco è fatto.

In sostanza abbiamo solo creato due pulsanti che ci permettono di spostarci agevolmente sul formulario; tutti gli altri compiti possono essere invece demandati alla Barra di Navigazione presente in fondo alla pagina. Su altre simpatiche proprietà dei campi di formulario torneremo tra poco.

13. Gestione dei Rapporti

Con la parola ***Rapporto*** intendiamo la produzione di "riassunti" del contenuto di un archivio, in forma di solito non modificabile, e formattati per la stampa. Quindi una cosa diversa dal *Formulario*, che invece è orientato alla modifica dei dati. Il *Rapporto* (*Report* per gli anglofoni e per Microsoft) è un aspetto molto interessante ed utile della gestione di un Archivio; nell'ottica di OOo, direi che è lo strumento più utile per accedere a dati che non sono disponibili direttamente da altri programmi. Supponiamo infatti di avere un Database esterno, proveniente dalla procedura gestionale dell'Azienda; potrei semplicemente voler stampare tutti i Clienti di un certo Agente con un fatturato minore di una soglia arbitraria. E' una tipica situazione dove usare OOo, senza scomodare il sistemista che si occupa del Software. Inoltre, disponendo di un accesso in sola lettura, sicuramente non posso fare danni.

13.1 I Rapporti del Modulo Base

Come ogni Software di Db che si rispetti, anche OOo 2.0 ha il suo bel modulo di Reporting; peccato che sia ancora un po' da migliorare, ma siamo sulla strada giusta. Cominciamo a chiarire un paio di piccole cose senza le quali potremmo credere che le cose funzionino piuttosto male....

1. l'unico modo per creare un nuovo *Rapporto* è *l'auto composizione*; a differenza delle altre entità del Modulo Base, non si può (almeno in questa versione) costruirsi un *Report* "a mano"
2. l'auto composizione ha ampi margini di miglioramento, quindi non tutto fila liscio al primo tentativo

fatte queste dovute premesse, mettiamoci al lavoro...

Come per il *Formulario*, anche un *Rapporto* deve "appoggiarsi" ad una struttura di dati esistente, che può essere una *Tabella* o una *Ricerca*. Siccome è nostra intenzione generare un semplice Report per elencare il contenuto della Mediateca, possiamo usare la *Ricerca* già creata per il *Formulario* del Capitolo precedente, cioè "*r_media_2a*". Selezioniamo dunque il Modulo dei Rapporti e partiamo con l'auto composizione. La prima cosa da fare è la scelta della struttura dei dati da usare, che, nel nostro caso è appunto "*r_media_2a*"; quindi bisogna stabilire quali campi dovranno fare parte del nostro Rapporto, e quindi avremo una situazione come in Figura:

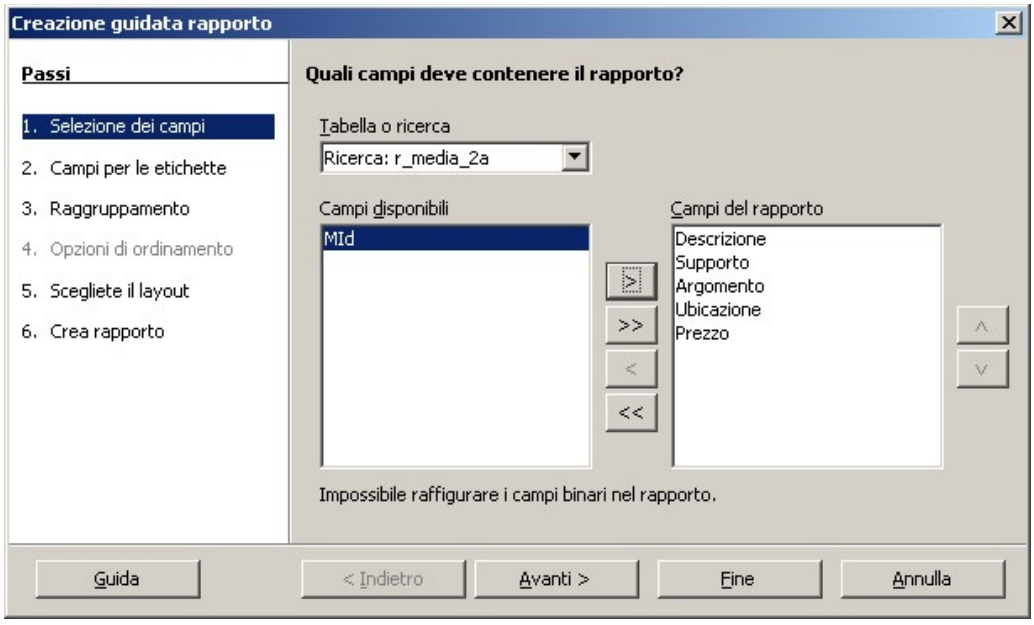


Figura 13.1.1: Auto composizione Rapporto, Passo 1

Il ***Passo 2*** ci chiede di scegliere una descrizione per le etichette di campo; siccome la nostra Ricerca era già "etichettata", basta confermare, come in figura:

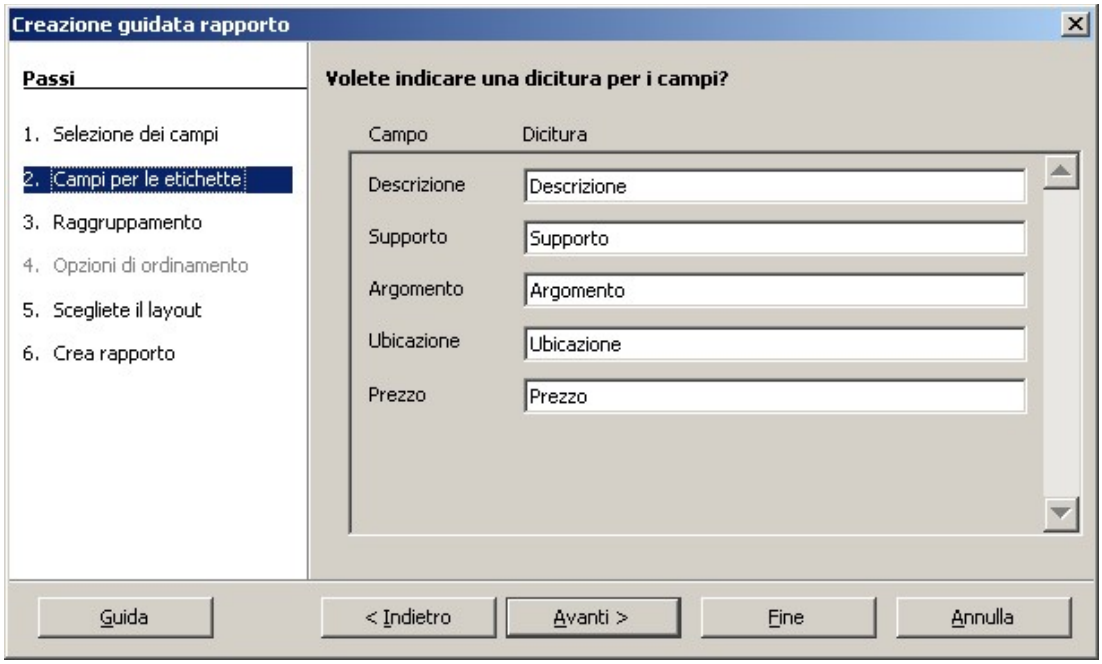


Figura 13.1.2: Auto composizione Rapporto, Passo 2

Il ***Passo 3*** riguarda eventuali opzioni per il raggruppamento, che nel nostro caso non sono utili, quindi passiamo oltre. Il ***Passo 4*** riguarda le Opzioni di ordinamento, ma siccome stiamo usando già una *Ricerca ordinata*, l'auto composizione lo salta. Il ***Passo 5*** permette di scegliere la formattazione del rapporto, come in figura:

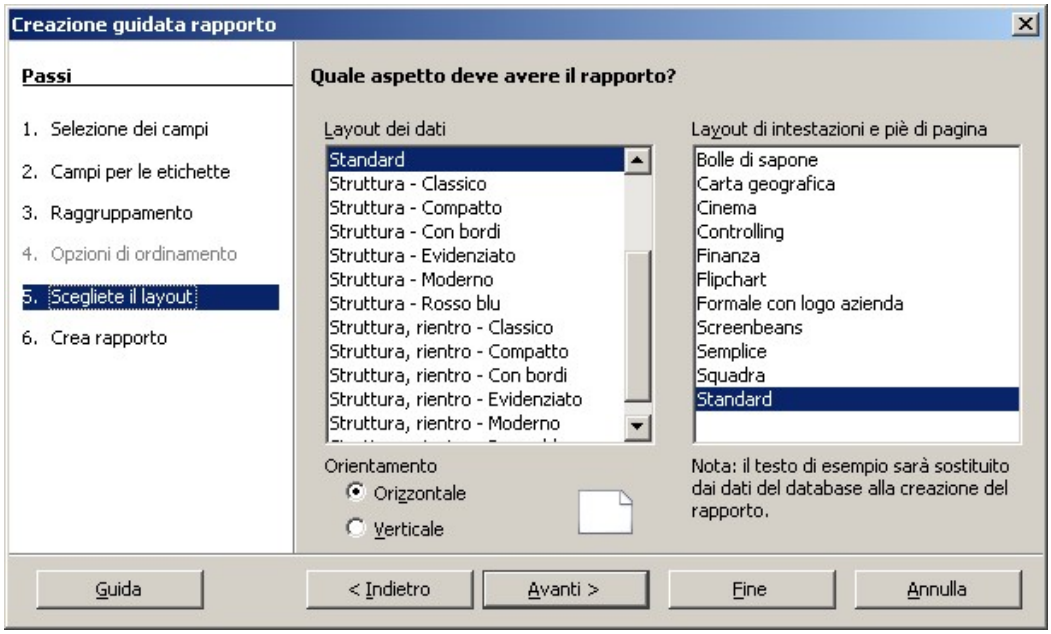


Figura 13.1.3: Auto composizione Rapporto, Passo 5

Lasciamo "*standard*" ed andiamo avanti. Al ***Passo 6*** possiamo (finalmente) concludere la nostra fatica e scegliere le ultime opzioni:

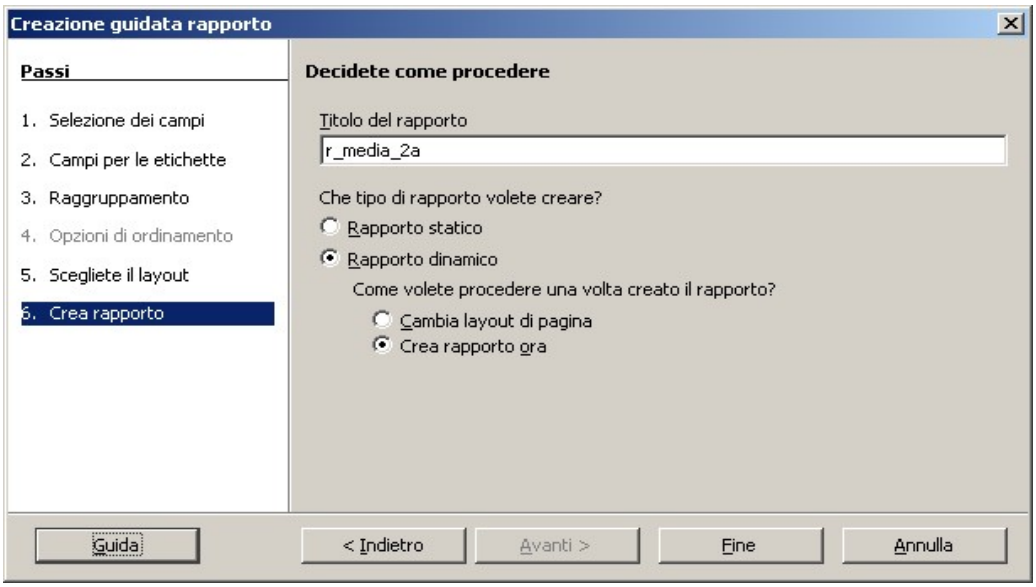


Figura 13.1.4: Auto composizione Rapporto, Passo 6

A parte il *Nome*, che potremo stabilire banalmente in "*Mediateca*", l'opzione successiva non è molto chiara. Secondo la Guida, un *Rapporto Statico* "fotografa" la situazione del Db in quel momento, quindi ogni successiva modifica ai Record non dovrebbe essere riportata; un *Rapporto Dinamico*, invece, è sempre aggiornato alla situazione attuale. Posto che per natura ci piaccioni più i *Rapporti Dinamici*, direi di scegliere questa opportunità e di salvare il lavoro.

Viene fuori una cosa del genere, che a dir poco non è il massimo. Forse dobbiamo cambiare qualcosa...

Titolo:
Autore: Filippo Cerulo
Data: 16/05/05

Descrizione	*Supporto*	*Argomento*	*Ubicazione*
Afterhours - Ballate per piccole iene	CD Audio	Fantasy	
Bad Boys II	DVD Video	Avventura	
Benni - La compagnia dei celestini	Libro	Romanzo	
Cornwell - L'Ultimo Distretto	Libro	Giallo	
De Gregori - Pezzi	CD Audio	Rock	
Guccini - Ritratti	CD Audio	Pop	
Il Signore degli Anelli	DVD Video	Fantasy	

Figura 13.1.5: I Rapporto per la Mediateca

13.2 All'interno di un Rapporto....

... ci sono sempre cose che non vanno, dice il saggio. Allora o "rompiamo" (e lasciamo perdere OOo), o ci diamo da fare per capire dov'è il problema. In verità, nel nostro caso, i problemi abbondano, ma il Rapporto è giovane, lasciamolo crescere. Dopo questa divagazione filosofica, facciamo uno sforzo per tornare seri e proseguiamo.

Apriamo in modifica il Rapporto *Mediateca*, ed avremo a video :

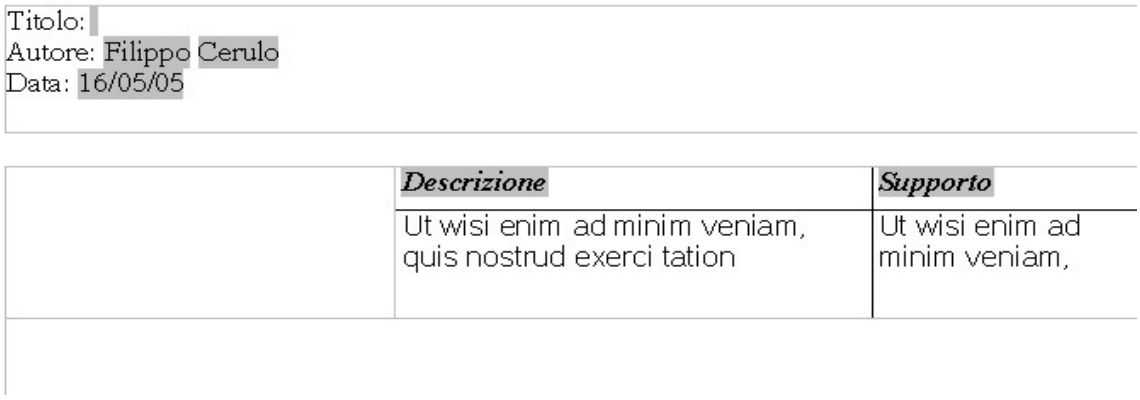


Figura 13.2.1: Rapporto in modalità struttura

In sostanza si tratta ancora una volta di un documento di testo con alcune particolarità. Per questo motivo si possono usare i noti strumenti di formattazione senza porsi molti problemi. In particolare, l'intestazione (Titolo, Autore etc.) è fatta di Comandi di Campo facilmente modificabili. Il *corpo*, invece, è una Tabella ed anche qui possiamo *aggiustare* quasi tutto. Cominciamo. Per prima cosa notate che l'auto composizione preferisce creare i *Rapporti* con

impostazione *orizzontale*, ma con *Formato -> Pagina* possiamo modificare questa opzione, se serve. Nel nostro caso abbiamo molte informazioni, quindi *orizzontale* va bene. Piuttosto spostiamo i campi a sinistra, eliminando lo spazio vuoto: l'auto composizione si ostina a spostare le colonne sulla destra, ma non mi sembra tanto utile. Quindi cambiamo un po' il formato dei caratteri.

TIP

Notate che qui non è possibile passare al *modo bozza*, perché siamo già in *modo bozza;* e neppure è possibile passare ad una anteprima del risultato. Se vogliamo vedere "come viene", dobbiamo salvare ed aprire col classico doppio click.

Ogni modifica compiuta sul formato del contenuto della seconda riga della Tabella (per intenderci quella in latino) sarà applicata a tutti i record in modalità *visualizzazione*, quindi possiamo impostare un bel "*Verdana 10 pt*"; anche il formato del campo *Prezzo* può essere utilmente trasformato in "*Valuta*" con la voce "*Formato Numero*" del Menu contestuale. Avremo così :

Titolo:
Autore: Filippo Cerulo
Data: 16/05/05

Descrizione	Supporto	Argomento	Ubicazione	Prezzo
Afterhours - Ballate per piccole iene	CD Audio	Fantasy		€ 15,00
Bad Boys II	DVD Video	Avventura		€ 15,00
Benni - La compagnia dei celestini	Libro	Romanzo		€ 0,00
Cornwell - L'Ultimo Distretto	Libro	Giallo		€ 0,00
De Gregori - Pezzi	CD Audio	Rock		€ 0,00
Guccini - Ritratti	CD Audio	Pop		€ 0,00
Il Signore degli Anelli	DVD Video	Fantasy		€ 0,00
Joss Stone - The Soul Session	CD Audio	Pop		€ 0,00
KDE 3.4 - Novità Major Release	Rivista	Linux	Linux & C. - 38	€ 11,50
Ligabue - Roma Stadio Olimpico	CD Audio	Classica		€ 15,00
Marillion - Marbles on the Road	DVD Video	Rock		€ 17,00
Montalbano - Il Ladro di merendine	Libro	Giallo		€ 0,00
Montalbano - La forma dell'acqua	Libro	Giallo		€ 0,00
Pearl Jam - Ten	CD Audio	Rock		€ 0,00
Springsteen - Devils & Dust	DVD Video	Rock		€ 0,00
test t	CD Audio	Fantasy		€ 0,00

Figura 13.2.2: Il Rapporto modificato

che mi sembra vada già meglio....

13.3 Rapporto di Gruppo

Purtroppo per voi non sto cambiando argomento; vorrei solo spiegare come si può facilmente raggruppare informazioni all'interno di un Rapporto di OOo. Supponiamo che l'elenco appena ottenuto ci piaccia di più raggruppato per *Argomento*. Però la Ricerca di prima

non è più utilizzabile, perché per *raggruppare* devo prima *ordinare*: cioè per avere tutti i miei *gialli* uniti insieme, la Ricerca deve essere ordinata per *Argomento*, altrimenti nisba (provare per credere). Quindi modifichiamo la *Ricerca*, chiamata *r_media_2b*, secondo questo schema:

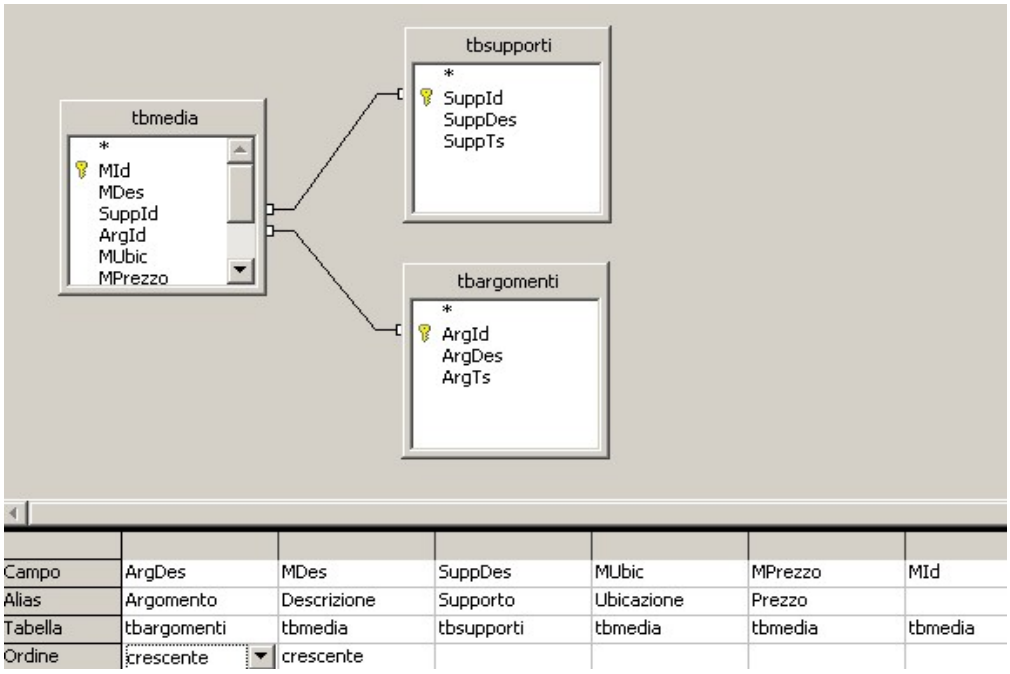


Figura 13.3.1: La nuova Ricerca

In pratica ho solo spostato il campo *Argomento* in prima posizione, scegliendo un ordinamento crescente. Ho anche lasciato l'ordinamento su *Descrizione*, in modo da avere, all'interno del gruppo, i Record in ordine alfabetico. Andiamo sui *Rapporti* e ripartiamo con l'auto composizione; i passi iniziali sono gli stessi di prima (a parte il nome della Ricerca), quindi, al ***Passo 3***, avremo:

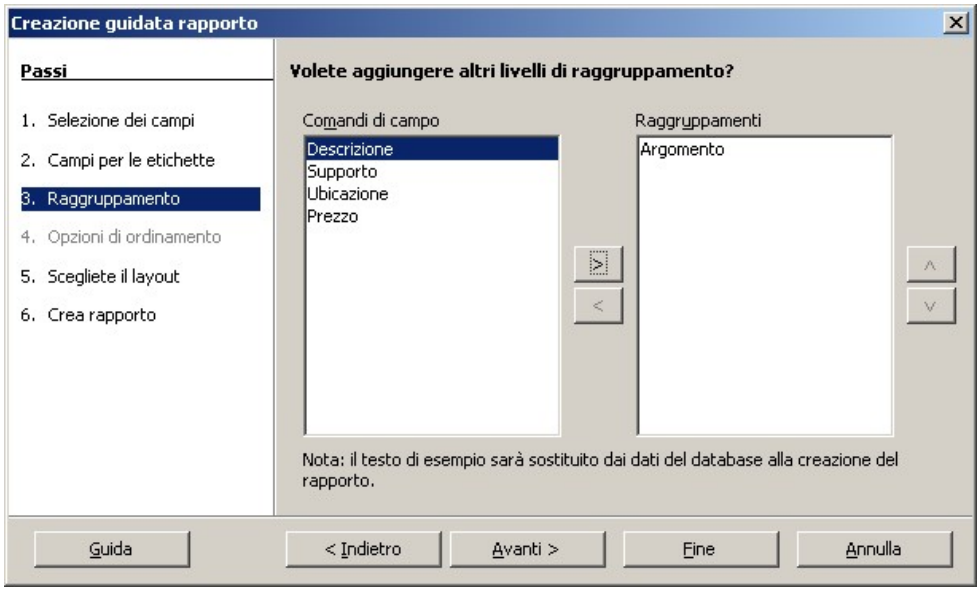
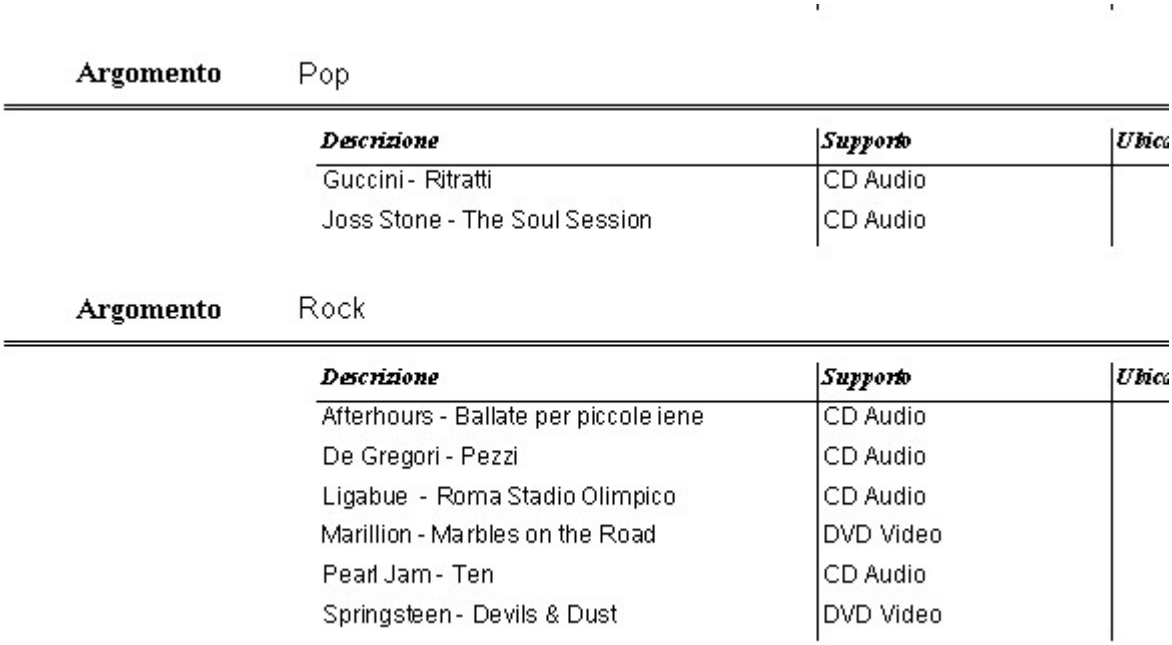


Figura 13.3.2: Raggruppamento per Argomento

scegliamo cioè di raggruppare per *Argomento*. Andiamo avanti ed il risultato sarà :



Argomento Pop

Descrizione	*Supporto*	*Ubica*
Guccini - Ritratti	CD Audio	
Joss Stone - The Soul Session	CD Audio	

Argomento Rock

Descrizione	*Supporto*	*Ubica*
Afterhours - Ballate per piccole iene	CD Audio	
De Gregori - Pezzi	CD Audio	
Ligabue - Roma Stadio Olimpico	CD Audio	
Marillion - Marbles on the Road	DVD Video	
Pearl Jam - Ten	CD Audio	
Springsteen - Devils & Dust	DVD Video	

Figura 13.3.3: Il Rapporto con i dati Raggruppati

Carino, vero ? Anche qui, aprendo il Rapporto in modifica, è possibile apportare tutte le migliorie "estetiche" desiderate, con risultati piuttosto incoraggianti.

13.4 Alterare un Rapporto ovvero...

... perché le cose semplici devono essere celate così bene...

Come abbiamo visto, in fondo il Rapporto è un Documento di Testo, quindi *l'apparenza* si modifica abbastanza facilmente. Ma la *sostanza* ? Meglio: le informazioni riguardanti i Dati contenuti nel *Rapporto* (Ricerca, Nomi dei Campi, Raggruppamenti etc.) dove sono ?

Per chi proviene da altri ambienti (ma anche considerando il già visto *Formulario*) sarebbe logico pensare che il nome del Campo di riferimento di una Colonna (ad es *Mdes* per *Descrizione*) sia tra le proprietà della Colonna stessa. Purtroppo, però, *non esistono le Proprietà della Colonna*. Stessa cosa per l'*origine* del Rapporto: noi abbiamo usato due *Ricerche*, ma se volessimo cambiare ed usare un'altra Tabella senza riscrivere il Rapporto daccapo e ripassare dall'auto composizione, che cosa mai dovremmo modificare ?

Confesso che capirci qualcosa non è stato facile, e la soluzione è saltata fuori quasi per caso. Se apriamo in modifica il primo rapporto che abbiamo creato (*Mediateca*) è possibile, al pari degli altri moduli, richiamare dalla barra *Struttura del Formulario* la finestra del *Navigatore*:

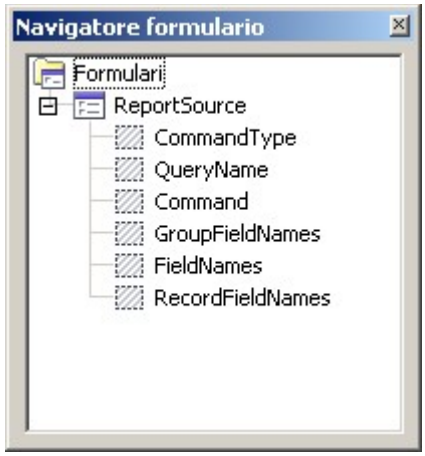


Figura 13.4.1: Il Navigatore del Formulario

Qui, ben nascosto, abbiamo tutto ciò che ci serve. All'interno della struttura di *ReportSource* ci sono degli *Hidden Controls*, cioè dei Controlli Nascosti che contengono le informazioni che ci interessano. Per visualizzare queste informazioni, è necessario scegliere la voce *Proprietà* dal menu contestuale di ogni controllo. Cominciamo.

Il primo controllo interessante è *QueryName*: si tratta appunto del nome della Ricerca (Query) a cui il Rapporto fa riferimento:

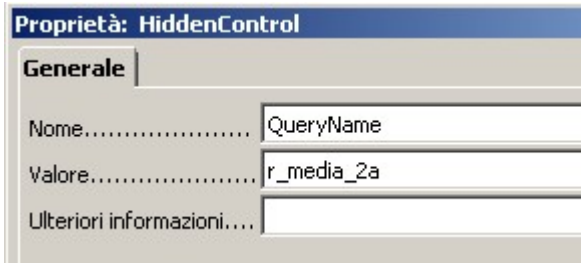


Figura 13.4.2: Il Controllo QueryName

I Controlli *FieldNames* e *RecordFieldNames* all'apparenza contengono gli stessi dati, cioè l'elenco dei Campi che fanno parte del Rapporto:

Figura 13.4.3: Il Controllo FieldNames

In realtà questo è vero solo se il Rapporto non contiene *Raggruppamenti*. Se invece esistono campi raggruppati, come per il nostro Rapporto *mediateca_2*, il controllo *FieldNames* contiene l'elenco completo dei Campi (Argomento;Descrizione;Supporto;Ubicazione;Prezzo) mentre *RecordFieldNames* contiene solo i campi che non fanno parte del Raggruppamento (Descrizione;Supporto;Ubicazione;Prezzo).

Il controllo *CommandType* regola le caratteristiche della "sorgente dati" del Rapporto: se il campo ***Valore*** è ***1***, significa che il Rapporto si basa su una *Ricerca* oppure su una *Tabella* il cui nome è specificato nel controllo *QueryName*; se il ***Valore*** è ***0***, allora verrà eseguito il *Comando SQL* presente nel controllo *Command*.

Modificando opportunamente questi valori è possibile alterare il funzionamento del *Rapporto* senza crearlo daccapo con l'auto composizione.

14. Rapporti .... Old Style

Il Modulo *Base* di OOo è, a tutti gli effetti, l'ultimo arrivato. OpenOffice ha sempre "sofferto" della mancanza di un Modulo di Database; la versione 1.X suppliva con le Sorgenti Dati ed i Formulari, ma la "concorrenza" (vedi Access) aveva davvero poco da temere. Ora le cose sono cambiate, ma certo non si può pretendere di recuperare in una sola volta tutto il tempo perduto.

Del nuovo modulo *Base*, la parte *Rapporti* è la meno "dotata", quindi con questo strumento non è che si possa fare molto. Abbiamo forse un'alternativa ? Certo, basta usare **Calc**.

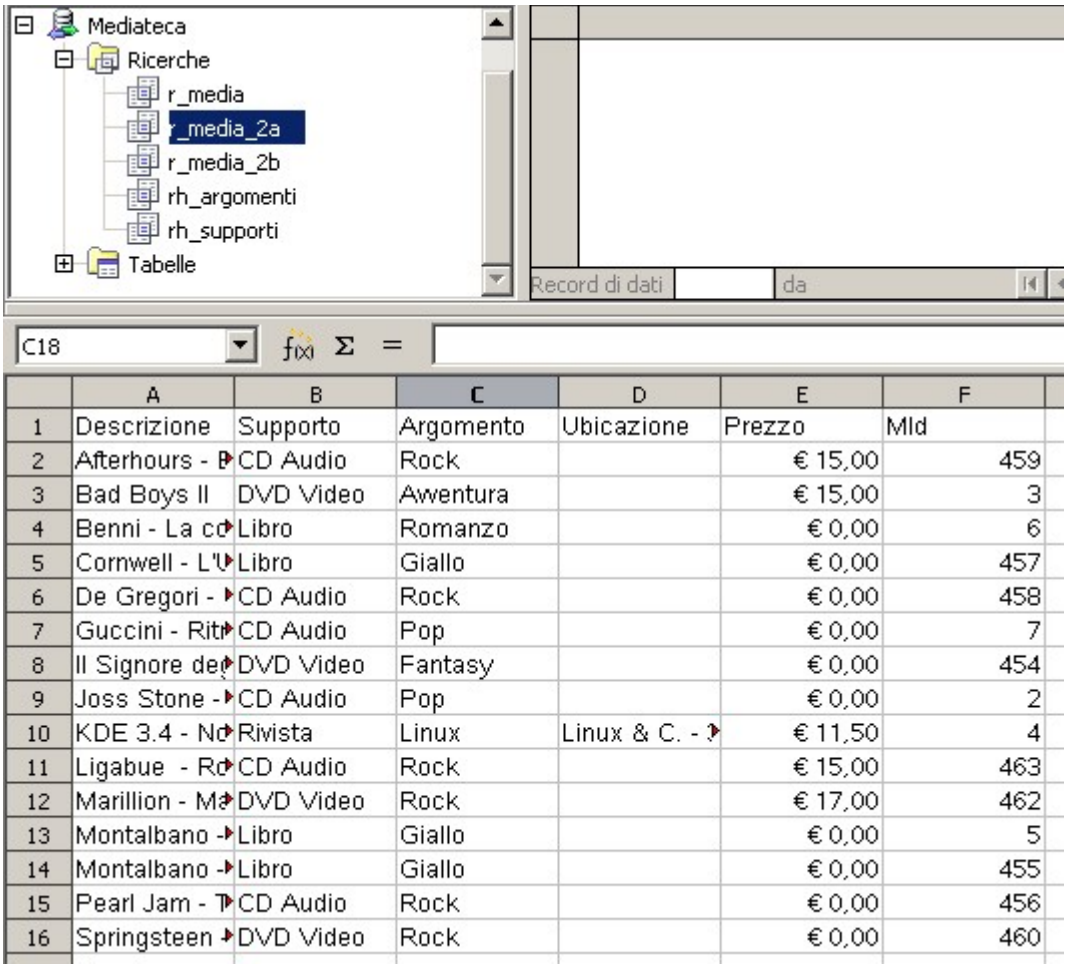
14.1 Calc per generare e stampare Report

Calc (il Modulo Spreadsheet di OOo) infatti possiede una utile caratteristica: è possibile collegare alle celle del foglio una Tabella o una Ricerca, e fare in modo che ogni modifica ai dati si rifletta nel foglio stesso. Spieghiamoci meglio: non è possibile modificare i dati direttamente dalle celle di *Calc* (o meglio le modifiche non vengono trasferite all'archivio), ma ogni modifica eseguita "a monte" può essere riportata nel foglio elettronico. Questo significa che, una volta creato il "link", anche se alla tabella sono state aggiunte 100 righe, le ritroveremo pari pari nel nostro Foglio Elettronico. Inoltre l'area di "collegamento" può essere formattata a nostro piacimento, e nel foglio è possibile aggiungere intestazioni e piè di pagina esattamente come in un normale foglio elettronico.

Quello che faremo sarà dunque creare una *struttura* per i dati che intendiamo comprendere nel Rapporto, e collegare un foglio elettronico alla *struttura* stessa. A titolo di esempio genereremo un Rapporto per il contenuto della nostra mediateca, e come struttura useremo una Ricerca. Al lavoro.

14.2 Il Report in Calc

In effetti le cose sono molto più semplici di quanto pensiate. Il primo passo è quello di collegare la Sorgente Dati (cioè il file .odb, nel nostro caso *Mediateca*) alla nostra installazione di OOo. Questo si ottiene, come abbiamo già visto, con la voce *OpenOffice.Org Base -> Database* di *Strumenti->Opzioni*. In questo modo, alla pressione del tasto ***F4***, *Mediateca* dovrebbe comparire tra le nostre Sorgenti Dati. Creiamo un nuovo documento Calc, apriamo le sorgenti dati con *F4*, selezioniamo la nostra ricerca *r_media_2a* (che abbiamo usato in precedenza) e "trasciniamola" nella prima casella del foglio elettronico. Avremo questo risultato:



	A	B	C	D	E	F
1	Descrizione	Supporto	Argomento	Ubicazione	Prezzo	Mld
2	Afterhours - P	CD Audio	Rock		€ 15,00	459
3	Bad Boys II	DVD Video	Awentura		€ 15,00	3
4	Benni - La co	Libro	Romanzo		€ 0,00	6
5	Cornwell - L'U	Libro	Giallo		€ 0,00	457
6	De Gregori -	CD Audio	Rock		€ 0,00	458
7	Guccini - Rit	CD Audio	Pop		€ 0,00	7
8	Il Signore de	DVD Video	Fantasy		€ 0,00	454
9	Joss Stone -	CD Audio	Pop		€ 0,00	2
10	KDE 3.4 - No	Rivista	Linux	Linux & C. -	€ 11,50	4
11	Ligabue - Re	CD Audio	Rock		€ 15,00	463
12	Marillion - Ma	DVD Video	Rock		€ 17,00	462
13	Montalbano -	Libro	Giallo		€ 0,00	5
14	Montalbano -	Libro	Giallo		€ 0,00	455
15	Pearl Jam - T	CD Audio	Rock		€ 0,00	456
16	Springsteen	DVD Video	Rock		€ 0,00	460

Figura 14.2.1: Una Ricerca trasferita in Calc

Cioè la Ricerca riportata per righe e colonne nel foglio elettronico. Bene, dal menu *Dati* scegliamo *Definisci Area*. Noterete che è stata creata una nuova area di nome *Importa1*; selezioniamola e premiamo il pulsante *EXTRA*.

Come in Figura, sarà necessario impostare tutte e quattro le *opzioni*. In particolare *Contiene Intestazioni Colonne* viene abilitato in automatico. *Inserisci/Elimina Celle* dice ad OpenOffice di aggiornare l'Area in base alle modifiche dei Dati usati come sorgente (quindi aggiungere o togliere righe). *Mantieni Formattazione* significa che tutte le opzioni di formattazione eseguite sul Foglio Elettronico saranno mantenute nel tempo. *Non salvare i dati importati* serve ad evitare che ogni volta i dati siano salvati insieme al Foglio (in questo caso si salva solo il "collegamento" alla tabella). A seconda delle esigenze, la seconda e la quarta opzione possono essere disabilitate, se ad esempio desideriamo che il Foglio contenga un'immagine "statica" degli archivi, magari da spedire per posta elettronica.

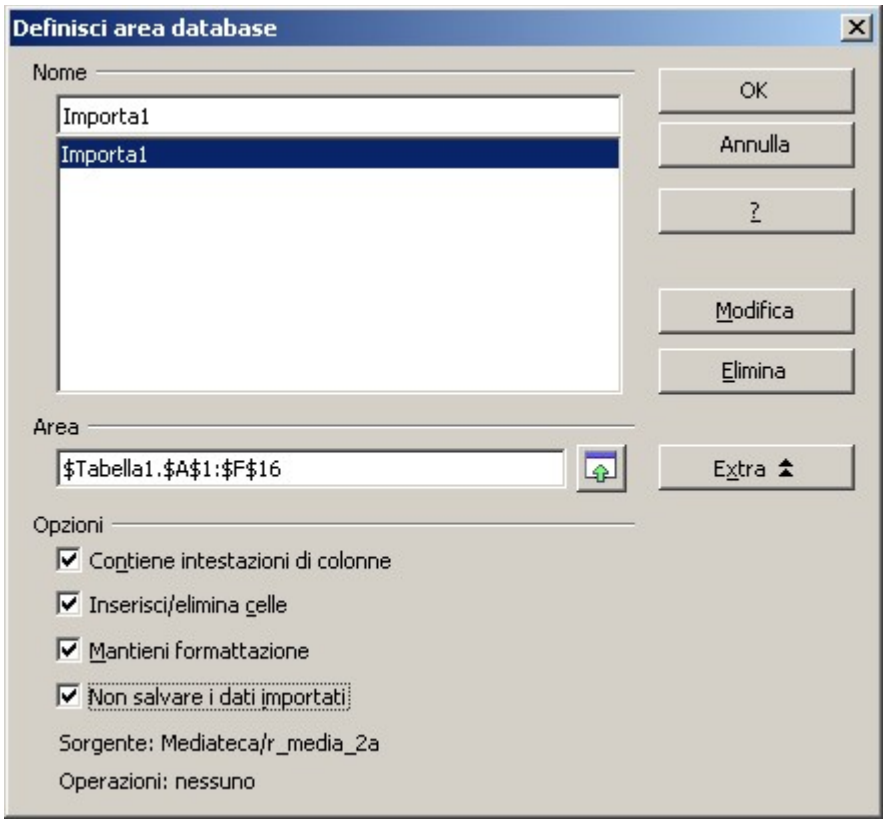


Figura 14.2.2: Le Opzioni per l'Area Dati

Ad ogni riapertura del Documento, OOo ci richiederà se vogliamo aggiornare le ricerche (bisogna ovviamente rispondere di si). Se invece, a foglio aperto, desideriamo ricaricare *manualmente* i Dati in modo da avere sotto mano le ultime modifiche eseguite sull'archivio, basterà usare la voce del menù *Dati->Aggiorna Area*, dopo aver selezionato una cella qualsiasi del nostro Report.

A questo punto possiamo formattare l'area come meglio ci aggrada, ad esempio così:

	A	B	C	D	E
1	**Descrizione**	**Supporto**	**Argomento**	**Ubicazione**	**Prezzo**
2	Bad Boys II	DVD Video	Gialli		€ 0,00
3	Benni - La compagnia dei celestini	Libro	Romanzo		€ 0,00
4	Cornwell - L'Ultimo Distretto	Libro	Gialli		€ 0,00
5	Guccini - Ritratti	CD Audio	Pop		€ 0,00
6	Il Signore degli Anelli	DVD Video	Fantasy		€ 0,00
7	Joss Stone - The Soul Session	CD Audio	Rock		€ 0,00
8	KDE 3.2 - Novità Major Release	Articolo	Linux	Linux & C. - 38	€ 0,00
9	Montalbano - Il Ladro di merendine	Libro	Gialli		€ 0,00
10	Montalbano - La forma dell'acqua	Libro	Gialli		€ 0,00
11	Pearl Jam - Ten	CD Audio	Rock		€ 0,00

Figura 14.2.3: L'area Dati dopo un piccolo ritocco estetico

Un'ultima ritoccata all'area di stampa, quindi impostiamo la pagina con Intestazioni, Margini ed orientamento et voila... il nostro Rapporto è pronto per essere consultato e stampato quando vogliamo.

TIP

Ricordate che le modifiche sui dati che potreste eseguire nel foglio non vengono riportate nel Database, quindi sono inutili. Per modificare i dati è necessario, in OOo, un *Formulario*.

15. Uso avanzato di....

15.1 Ricerche

Le Ricerche (*query*) sono un argomento all'apparenza complicato. Infatti, pur con un'interfaccia grafica abbastanza intuitiva, non si riesce del tutto a "mascherare" la potenza dell'istruzione ***SELECT*** di SQL. In OOo le Ricerche sono usate essenzialmente come *query di selezione* (e chi proviene da altri ambienti sa di cosa parlo), cosa sostanzialmente diversa da una *query di comando*.

Nel primo caso, infatti, lo scopo e quello di creare, da una o più tabelle di partenza, una *selezione* (***SELECT***) di dati con alcune caratteristiche aggiuntive (come gli *alias* e *l'ordinamento*), magari scelti anche in modo da soddisfare specifici *criteri*. Questo tipo di elaborazioni *non modificano* i dati del Db, si limitano a mostrarli in una forma diversa.

Una *query di comando* è invece una istruzione SQL che altera i dati del Db (eseguendo, ad es. dei calcoli) e di solito usa l'istruzione Sql ***UPDATE***. Ma andiamo con ordine...

15.1.1 Ricerche su Tabelle singole

Questo è il caso più semplice, perché è coinvolta una sola tabella. Abbiamo già visto come si crea questo tipo di ricerca nel paragrafo di introduzione al modulo *Base*. Se torniamo sulla Figura precedente:

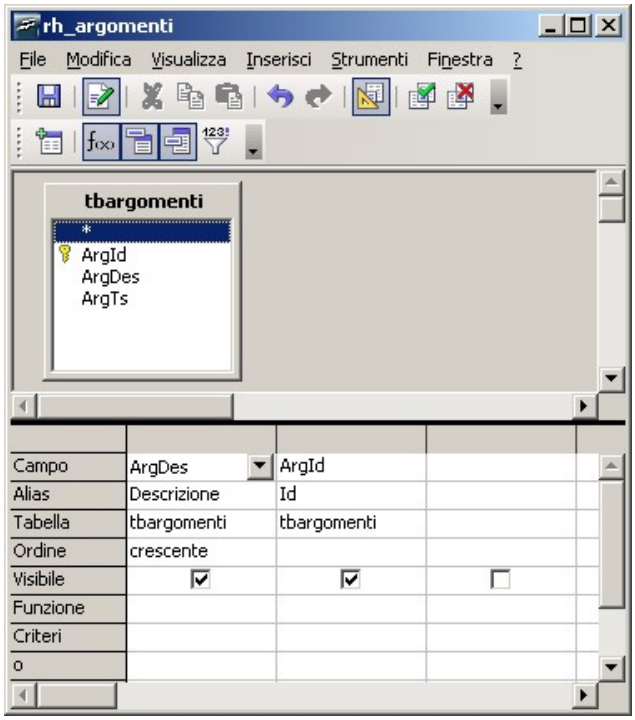


Figura 15.1.1: Ricerca su una Tabella singola

ci rendiamo conto che non si tratta di nulla di complicato. Basta selezionare i campi, assegnare (se vogliamo) un alias, stabilire un ordine et voila...

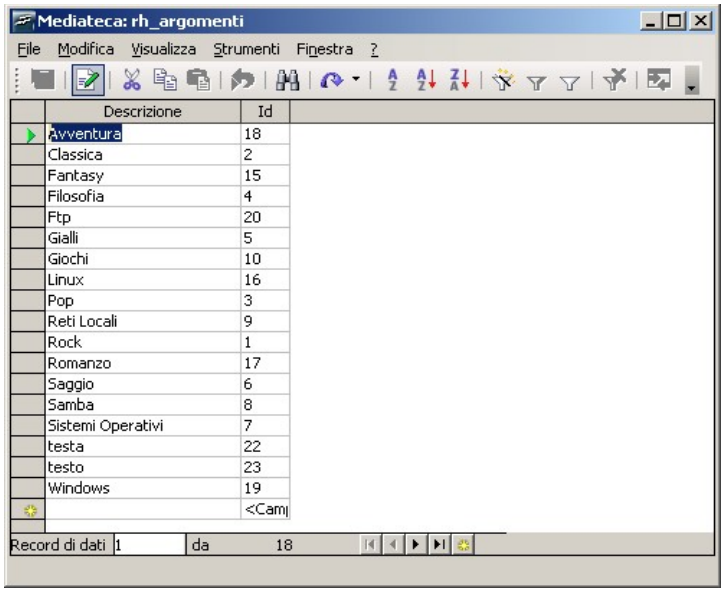


Figura 15.1.2 Il risultato della Query

Possiamo anche esaminare facilmente l'istruzione SQL, che è questa:

```
SELECT `ArgDes` AS `Descrizione`, `ArgId` AS `Id`
FROM `mediateca`.`tbargomenti` `tbargomenti`
ORDER BY `Descrizione` ASC
```

Notate che:

- l'alias viene specificato subito dopo il ***SELECT*** con la parola chiave ***AS***
- nella riga del ***FROM***, OOo usa un alias anche per il nome tabella (*tbargomenti* per *mediateca.tbargomenti*) e questo non sarebbe strettamente necessario
- la riga ***ORDER BY*** indica l'ordinamento ed usa l'alias per indicare il nome di campo, concludendo con un ***ASC*** che significa ***ASCENDING*** (cioè ascendente)

OOo permette anche la scrittura diretta della query in SQL, commutando la finestra col pulsante *Modalità Struttura On/Off* della barra degli strumenti; se scriviamo direttamente l'istruzione SQL, in fase di salvataggio comunque OOo "aggiusta" la sintassi secondo i propri standard, e questo non sempre è piacevole. Ma facciamo un piccolo esempio.

Supponiamo di voler creare una Ricerca che mi elenchi in modo completo i dati della Tabella *TbMedia*. Sul pannello di sinistra selezioniamo *Ricerche* e dal riquadro delle Attività in alto *Crea Ricerca in vista SQL*. Quindi :

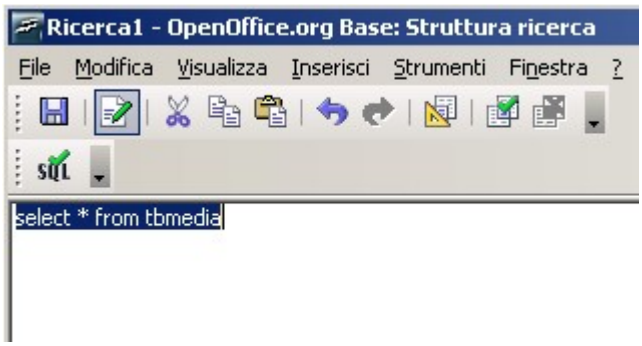


Figura 15.1.3: Una Ricerca in vista SQL

Se eseguiamo la Ricerca, il risultato sarà:

	MId	MDes	SuppId	ArgId	MUbic	MPrezzo	MTs
▶	2	Joss Ston	3	3		0,00	10/05/0
	3	Bad Boys	1	18		15,00	11/05/0
	4	KDE 3.4 -	7	16	Linux & C.	11,50	11/05/0
	5	Montalbar	5	5		0,00	11/05/0
	6	Benni - La	5	17		0,00	09/09/0
	7	Guccini - F	3	3		0,00	12/05/0
	454	Il Signore	1	15		0,00	11/05/0
	455	Montalbar	5	5		0,00	08/09/0
	456	Pearl Jam	3	1		0,00	06/05/0
	457	Cornwell -	5	5		0,00	11/05/0
	458	De Gregor	3	1		0,00	06/05/0
	459	Afterhour	3	1		15,00	16/05/0
	460	Springstee	1	1		0,00	11/05/0

Record di dati 1 da 15 *

Figura 15.1.4: Risultato della Ricerca su TbMedia

Ora salviamo la Ricerca. Se in seguito chiediamo di modificare la Query, il risultato sarà:

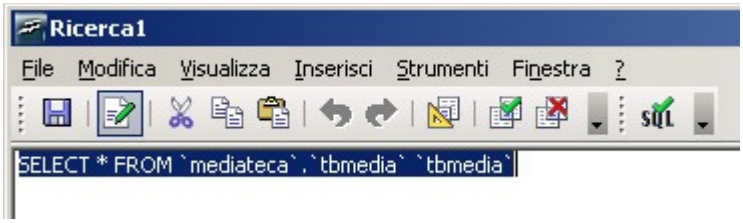


Figura 15.1.5: La Ricerca "rielaborata" da OOo

Quindi OOo ha aggiunto gli apici, ed un alias non richiesto. Questo strano comportamento limita un po' la scrittura diretta di Ricerche in SQL, e bisogna tenerne conto perché non tutti i Server di Db gradiscono queste modifiche di sintassi.

15.1.2 Ricerche con parametri

Una delle cose più interessanti delle Ricerche è che possiamo fornire dei *Criteri* in modo da "filtrare" il risultato a nostro piacimento. Supponiamo di volere una lista di tutti i DVD (quindi

con *SuppId=1*), ordinati per Descrizione con accanto il prezzo. Potremmo creare una Ricerca come questa:

Figura 15.1.6: Ricerca con criteri

che ritorna questo risultato:

	Descrizione	Prezzo
▶	Bad Boys II	15,00
	Il Signore degli Anelli	20,00
	Marillion - Marbles on the Roads	17,00
	Springsteen - Devils & Dust	18,00

Figura 15.1.7: I nostri DVD

e che in SQL è riportata così:

```
SELECT `MDes` AS `Descrizione`, `MPrezzo` AS `Prezzo`
FROM `mediateca`.`tbmedia` `tbmedia`
WHERE ( ( `SuppId` = 1 ) )
ORDER BY `Descrizione` ASC
```

In fondo niente di complicato; abbiamo aggiunto il valore desiderato alla voce *criteri* della colonna *SuppId*, abbiamo anche escluso questa colonna dal risultato togliendo il segno di spunta su *visibile*. Dal punto di vista SQL, notate che le colonne da mostrare sono elencate subito dopo la ***SELECT***, e che è stata aggiunta la clausola ***WHERE***, che permette appunto di specificare un criterio. Abbiamo ottenuto quello che volevamo, ma il risultato è un po' "grezzo". Volendo ad esempio elencare i CD Audio (*SuppId=3*) dovremmo modificare la ricerca. Ma a tutto c'è rimedio.....

Infatti sarebbe opportuno che, ad ogni apertura, la Ricerca richiedesse il *parametro* da considerare. Ma questo è facile, basta indicare nel campo *criteri* un nome a piacere (magari indicativo della richiesta) preceduto dal simbolo ":" (*due punti*) come in figura:

Campo	MDes	MPrezzo	SuppId
Alias	Descrizione	Prezzo	
Tabella	tbmedia	tbmedia	tbmedia
Ordine	crescente		
Visibile	☑	☑	☐
Funzione			
Criteri			:Codice_Supporto

Figura 15.1.8: Ricerca con la richiesta del parametro

In questo modo, all'esecuzione avremo questa maschera :

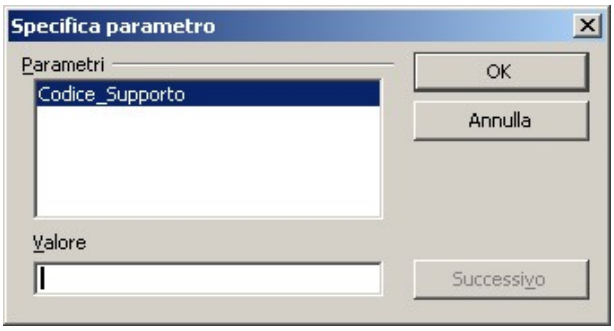


Figura 15.1.9: Richiesta del parametro

dove è possibile immettere il valore desiderato. Dal punto di vista SQL:

```
SELECT `MDes` AS `Descrizione`, `MPrezzo` AS `Prezzo`
FROM `mediateca`.`tbmedia` `tbmedia`
WHERE ( ( `SuppId` = :Codice_Supporto ) )
ORDER BY `Descrizione` ASC
```

non è poi cambiato molto....

15.1.3 Ricerche con più Tabelle

Come abbiamo già visto, è possibile includere più di una Tabella nella Ricerca. Per il nostro formulario, in precedenza, avevamo preparato questa Ricerca:

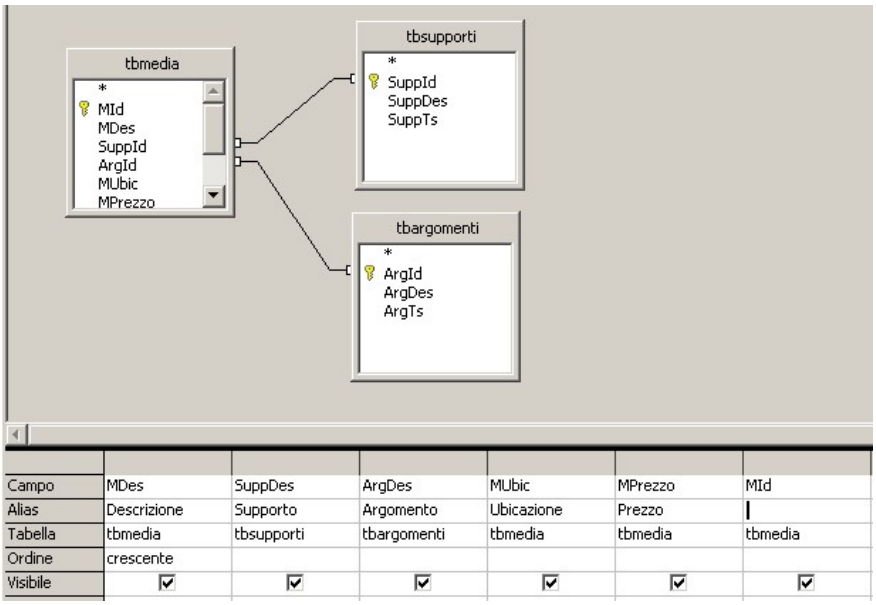


Figura 15.1.10 La ricerca per la prima Scheda del nostro Formulario

Lo scopo è quello di includere anche la Tabella dei *Supporti* e quella degli *Argomenti* in modo da avere la descrizione "estesa" dei campi corrispondenti. Infatti il risultato è:

Descrizione	Supporto	Argomento	Ubicazione	Prezzo	Id
Afterhours - Ballate per piccole iene	CD Audio	Rock		15,00	459
Bad Boys II	DVD Video	Avventura		15,00	3
Benni - La compagnia dei celestini	Libro	Romanzo		8,00	6
Cornwell - L'Ultimo Distretto	Libro	Giallo		10,00	457
De Gregori - Pezzi	CD Audio	Rock		18,00	458
Guccini - Ritratti	CD Audio	Pop		16,00	7
Il Signore degli Anelli	DVD Video	Fantasy		20,00	454
Joss Stone - The Soul Session	CD Audio	Pop		15,00	2
KDE 3.4 - Novità Major Release	Rivista	Linux	Linux & C. - 38	11,50	4
Ligabue - Roma Stadio Olimpico	CD Audio	Rock		15,00	463
Marillion - Marbles on the Roads	DVD Video	Rock		17,00	462
Montalbano - Il Ladro di merendine	Libro	Giallo		8,00	5
Montalbano - La forma dell'acqua	Libro	Giallo		8,00	455
Pearl Jam - Ten	CD Audio	Rock		27,00	456
Springsteen - Devils & Dust	DVD Video	Rock		18,00	460

Figura 15.1.11: Ricerca con più Tabelle

Per usare più Tabelle in una ricerca è necessario *mettere in relazione* le Tabelle stesse, cioè indicare quali sono i campi comuni alle strutture dei Dati. Nel nostro caso, *TbMedia* si lega a *TbArgomenti* attraverso il campo comune *ArgId*, ed a *TbSupporti* con *SuppId*. In questo esempio i campi citati sono, come abbiamo visto, anche *chiavi esterne* per *TbMedia*, ma questo non è indispensabile. Se osserviamo l'istruzione SQL per la Ricerca:

```
SELECT
    `tbmedia`.`MDes` AS `Descrizione`,
    `tbsupporti`.`SuppDes` AS `Supporto`,
    `tbargomenti`.`ArgDes` AS `Argomento`,
    `tbmedia`.`MUbic` AS `Ubicazione`,
    `tbmedia`.`MPrezzo` AS `Prezzo`,
```

```
    `tbmedia`.`MId` AS `Id`
FROM
    `mediateca`.`tbargomenti` `tbargomenti`,
    `mediateca`.`tbmedia` `tbmedia`,
    `mediateca`.`tbsupporti` `tbsupporti`
WHERE
    ( `tbargomenti`.`ArgId` = `tbmedia`.`ArgId`
    AND `tbsupporti`.`SuppId` = `tbmedia`.`SuppId` )
ORDER BY `Descrizione` ASC
```

possiamo notare che:

- nella ***SELECT*** ogni campo è preceduto dal nome della Tabella a cui appartiene;
- il ***FROM*** elenca tutte le Tabelle coinvolte;
- ***WHERE*** indica le relazioni da considerare;

in fondo mi sembra tutto abbastanza chiaro. Ci sarebbe da dire qualcosa sul *tipo* di relazioni che si possono stabilire tra Tabelle in una ricerca, ma rimandiamo l'argomento di qualche paragrafo.

15.1.4 Ricerche con più parametri

Possiamo indicare più di un criterio in una ricerca, sia *fisso* che *variabile*. Nel caso precedente, volendo estrarre i dati in base sia al *Supporto* che *all'Argomento*, modificheremo la Ricerca in questo modo:

Campo	MDes	SuppDes	ArgDes	MUbic	MPrezzo	MId
Alias	Descrizione	Supporto	Argomento	Ubicazione	Prezzo	Id
Tabella	tbmedia	tbsupporti	tbargomenti	tbmedia	tbmedia	tbmedia
Ordine	crescente					
Visibile	☑	☑	☑	☑	☑	☑
Funzione						
Criteri		:Supporto	:Argomento			

Figura 15.1.12: Ricerca con più Parametri

all'apertura, OOo ci mostra la richiesta dei Parametri in questo modo:

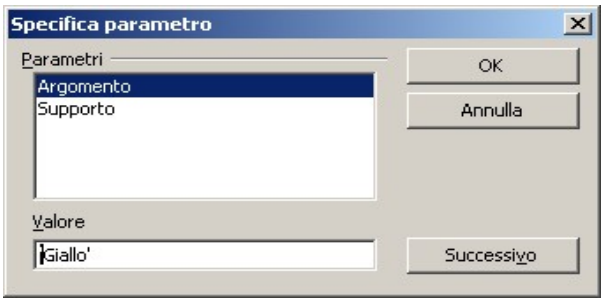


Figura 15.1.13: Richiesta di più parametri

possiamo perciò specificare due valori da utilizzare per la selezione (ad esempio "*Giallo*" e "*Libro*") per ottenere il risultato voluto:

Descrizione	Supporto	Argomento	Ubicazione	Prezzo	Id
Cornwell - L'Ultimo Distretto	Libro	Giallo		10,00	457
Montalbano - Il Ladro di merendine	Libro	Giallo		8,00	5
Montalbano - La forma dell'acqua	Libro	Giallo		8,00	455

Figura 15.1.14: Risultato della Ricerca

Tradotto in SQL:

```
SELECT
    `tbmedia`.`MDes` AS `Descrizione`,
    `tbsupporti`.`SuppDes` AS `Supporto`,
    `tbargomenti`.`ArgDes` AS `Argomento`,
    `tbmedia`.`MUbic` AS `Ubicazione`,
    `tbmedia`.`MPrezzo` AS `Prezzo`,
    `tbmedia`.`MId` AS `Id`
FROM
    `mediateca`.`tbargomenti` `tbargomenti`,
    `mediateca`.`tbmedia` `tbmedia`,
    `mediateca`.`tbsupporti` `tbsupporti`

WHERE

(`tbargomenti`.`ArgId`=`tbmedia`.`ArgId` AND `tbsupporti`.`SuppId` =
`tbmedia`.`SuppId` )

   AND

((`tbsupporti`.`SuppDes` = :Supporto AND `tbargomenti`.`ArgDes` =
:Argomento ))

ORDER BY `Descrizione` ASC
```

notate l'aggiunta tramite un ***AND*** alla clausola ***WHERE*** della richiesta dei parametri. Attenzione: nella richiesta dovrete indicare ESATTAMENTE i valori desiderati (cioè "*Giallo*" e non, ad es. "*Gialli*", ma questo è ovvio), e soprattutto TUTTI i valori, in caso contrario la Ricerca non ritornerà alcuna riga.

15.1.5 Criteri...

La parola chiave ***WHERE*** di SQL (e di conseguenza la casella dei *criteri* nelle ricerche) permette di fare molte più cose di quante abbiamo finora avuto modo di vedere. Innanzi tutto possiamo disporre di tutti gli operatori relazionali più comuni (>,<,<> etc.), per cui, se odiamo i gialli, possiamo scrivere ***<>'Giallo'*** nei criteri del campo *ArgDes*. Ma ci sono cose più interessanti....

Quelli che di voi hanno usato il buon Ms-Dos (ma anche qualsiasi SHELL da linea di comando) sanno cosa sono le ***wildcards***. Si tratta di simboli speciali che rappresentano gruppo di caratteri (*****) oppure un carattere singolo (**?**). Per cui, se scrivo ***Sil*,*** intendo tutte le parole

che iniziano per ***Sil*** (*Silvia*, Silvana, *silenzio* etc.); se scrivo ***d?re*** intendo *dire* o *dare* e così via. OOo permette l'uso delle *wildcards* nei criteri assieme alla parola chiave ***COME*** . Così, scrivendo nei criteri di *ArgDes* ad esempio ***COME 'G*'***, avremo i *Gialli* ed i *Giochi*; invece ***COME 'Sa???'*** elencherà i Record con argomento *Samba* e *Saggi*.

Tecnica

La parola chiave di OOo ***COME*** corrisponde al token SQL ***LIKE***; i caratteri usati nelle wildcards, però, sono diversi; in particolare ***'*'*** di OOo equivale a ***'%'*** di SQL e ***'?'*** equivale a ***'_'***. Per cui, ***COME 'G*'*** verrà "tradotto" con ***LIKE 'G%'*** e ***COME 'Sa???'*** con ***LIKE 'Sa___'***.

Un'altra opzione interessante è ***TRA*** (cioè ***BETWEEN*** in SQL); un'espressione come ***TRA x E y*** (***BETWEEN x AND y***) permette di selezionare un intervallo di valori presenti in un campo.

Discorso un po' diverso per le Date: nei criteri vanno indicate preferibilmente **tra due cancelletti (#)**, per cui ***#18/03/2003#*** significa appunto il 18 Marzo 2003; stranamente la parola chiave ***TRA*** non funziona con le Date, per cui per specificare un intervallo dobbiamo scrivere ***>= #01/03/2003# E <= #31/03/2003#*** ed avremo tutte le date di Marzo 2003.

TIP

Domanda: si possono usare le wildcards anche con i parametri ? La risposta è **SI**, ma solo se si usano quelle di SQL. Mi spiego meglio. In una delle Ricerche precedenti, nei criteri del Campo *ArgDes* abbiamo immesso il valore *:Argomento*, in modo che OOo chiedesse, in fase di apertura della query, l'argomento da selezionare. Se si volesse usare una wildcard anche in questo caso dovremmo modificare i criteri come in figura:

Figura 15.1.15:
Parametri con wildcards

a questo punto alla richiesta del Parametro, sareste tentati di scrivere ***'G*'*** per avere i *Gialli* ed i *Giochi*; **ERRORE** !! dovrete scrivere ***'G%'***, ma non chiedetemi perché....

15.1.6 Ricerche modificabili

Abbiamo già visto l'importanza delle Ricerche, e come esse siano di solito alla base di Formulari e Rapporti. Dobbiamo chiederci, perciò, *quali caratteristiche deve avere una ricerca per permettere anche la modifica dei dati ?* La domanda non è banale, perché se agganciamo un Formulario ad una Ricerca, dobbiamo essere sicuri che i Dati siano modificabili (cioè sia possibile aggiungere, cancellare e cambiare intere righe o singole informazioni). Nel Capitolo dedicato all'introduzione alla Mediateca abbiamo usato una Ricerca (*rh_argomenti*) per costruire un Formulario per la gestione della Tabella degli Argomenti. La Ricerca comprende i Campi *ArgId* (identificatore, Chiave Primaria) e *ArgDes* (Descrizione). Siccome *ArgId* è un contatore ad incremento automatico (quindi assegnato dal sistema) potreste chiedervi come mai è stato incluso nella Ricerca. La risposta è semplice: ***OOo permette di modificare i Dati di una Ricerca SOLO se è inclusa, ed è visibile, la Chiave primaria***. Cioè, questa ricerca permette la modifica:

Campo	ArgDes	ArgId
Alias		
Tabella	tbargomenti	tbargomenti
Ordine	crescente	
Visibile	☑	☑
Funzione		

Figura 15.1.16: Una Ricerca modificabile

Ma queste sono a sola lettura (in quella a sinistra manca la Chiave primaria ed in quella a destra la stessa non è visibile) :

Campo	ArgDes	
Alias		
Tabella	tbargomenti	
Ordine	crescente	
Visibile	☑	☐
Funzione		

Campo	ArgDes	ArgId
Alias		
Tabella	tbargomenti	tbargomenti
Ordine	crescente	
Visibile	☑	☐
Funzione		

Figura 15.1.17: Ricerche a sola lettura

Analogamente, in linea di massima, ***le Ricerche composte da più Tabelle non permettono mai la modifica dei Dati***.

15.1.7 Esecuzione diretta di Comandi SQL

Le Ricerche sono molto utili per la selezione delle Informazioni, ma come fare per eseguire modifiche alle Tabelle senza scrivere manualmente i valori ? Supponiamo di voler assegnare a tutti i CD Audio presenti nella nostra Mediateca il prezzo convenzionale di 16 Euro (si, lo so,

costano di più, ma io sono ottimista...). Potremmo costruire una Ricerca che elenca solo i CD Audio, e scrivere a mano il valore di *16* nel campo *MPrezzo*. Operazione lunga e noiosa, quindi troviamo un altro sistema.

OOo permette di creare Ricerche composte da Comandi SQL che vengono inviati direttamente al Server. Questo si ottiene con il pulsante *Esegui direttamente Comando SQL*, che è l'ultimo a destra della Barra visualizzata in modalità SQL (cioè NON in vista struttura):

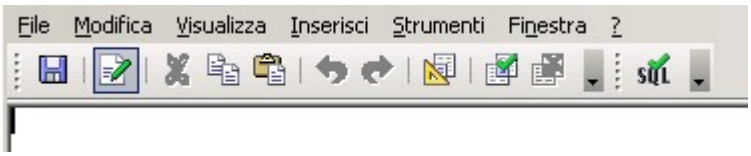


Figura 15.1.18: Barra per l'esecuzione di comandi SQL

Basta scrivere un comando SQL valido, che OOo passerà al Server per l'esecuzione. Nel nostro caso :

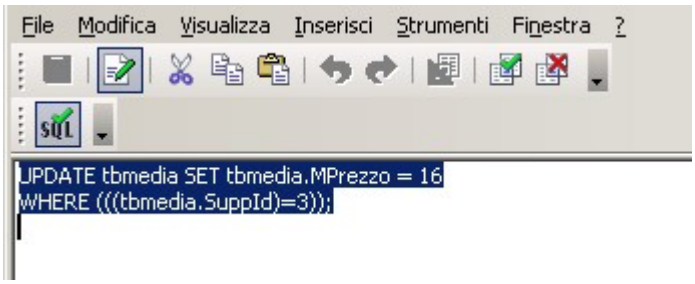


Figura 15.1.19: Esecuzione diretta di un Comando SQL

Che, scritta un po' meglio viene:

```
UPDATE tbmedia
    SET tbmedia.MPrezzo = 16
    WHERE (((tbmedia.SuppId)=3));
```

Quindi si usa l'istruzione ***UPDATE***, con ***SET***, si indica il campo da modificare e, con ***WHERE***, il campo di applicazione (*SuppId* uguale a 3, cioè "*CD Audio*"): niente di difficile. Una volta salvata la Ricerca, si manda in esecuzione con un doppio click, e OOo ci informa sul fatto che :

Figura 15.1.20: Esecuzione di una Query di comando

Questo è ovvio: non stiamo usando una query di selezione (***SELECT***), ma una di "comando" (***UPDATE***) quindi non c'è nessun insieme di Record come risultato (*no ResultSet*); in ogni caso l'istruzione è stata eseguita correttamente. Questo tipo di Ricerche permette l'esecuzione di qualsiasi comando SQL, quindi basta conoscere un po' il linguaggio per ottenere con pochi clic risultati notevoli.

TIP

Come abbiamo già avuto modo di dire, ogni Server SQL ha un suo *dialetto* che differisce in modo più o meno marcato dallo standard. Siccome in questo caso OOo non *traduce* l'istruzione, bisogna essere certi che la sintassi sia valida in funzione del Server che si sta utilizzando. Con comandi semplici, come quello che abbiamo visto, di solito non ci sono problemi, ma non sempre è così. Inoltre le query di comando modificano i dati del Database senza produrre messaggi di avvertimento, quindi attenzione a quello che fate....

15.1.8 Raggruppamenti e formule matematiche

In alcuni casi è necessario *raggruppare* le righe di una Tabella in funzione di un valore presente in uno dei campi, soprattutto per calcolare totali o conteggi di altro tipo. Questo è il compito della riga ***Funzione*** nella visualizzazione struttura della Ricerca.

Nella versione 2.0.0 di OOo che stiamo utilizzando, però, questo tipo di elaborazione è praticamente impossibile, perché la sintassi dell'istruzione SQL non viene generata in modo corretto. Il Bug è stato eliminato con la versione 2.0.1.

Supponiamo di desiderare una Ricerca che conti quanti Media abbiamo, suddividendoli per Argomento. Potremmo costruire qualcosa del genere:

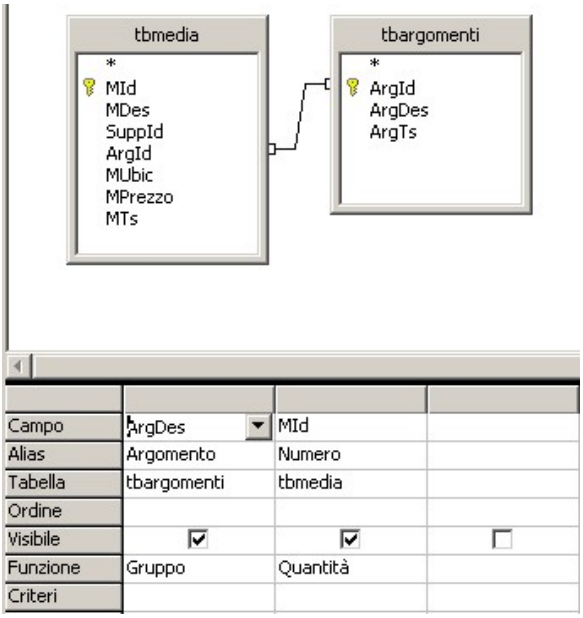


Figura 15.1.21: Ricerca con uso di Gruppi

Per ottenere:

	Argomento	Numero
▶	Avventura	1
	Fantasy	1
	Giallo	3
	Linux	1
	Pop	2
	Rock	6
	Romanzo	1

Figura 15.1.22: Risultato della Ricerca

In pratica sulla prima colonna abbiamo messo il valore su cui effettuare il raggruppamento (in questo caso *ArgDes*, che è più descrittivo, ma poteva andare anche *ArgId*), e nella colonna successiva il campo su cui vogliamo eseguire l'operazione matematica. La Funzione usata è *Quantità*, ma avremmo potuto scegliere anche tra *Massimo*, *Minimo*, *Media* e *Somma*. Una Ricerca con abbastanza funzioni, ad esempio, potrebbe essere:

Campo	ArgDes	MId	MPrezzo	MPrezzo	MPrezzo
Alias	Argomento	Numero	Prezzo Medio	Prezzo Massimo	Valore Totale
Tabella	tbargomenti	tbmedia	tbmedia	tbmedia	tbmedia
Ordine					
Visibile	☑	☑	☑	☑	☑
Funzione	Gruppo	Quantità	Media	Massimo	Somma
Criteri					

Figura 15.1.23: Ricerca con più funzioni di Raggruppamento

il cui risultato è:

Argomento	Numero	Prezzo Medio	Prezzo Massimo	Valore Totale
Avventura	1	16	16	16
Fantasy	1	16	16	16
Giallo	3	8,67	10	26
Linux	1	11,5	11,5	11,5
Pop	2	16	16	32
Rock	6	15,25	16	91,5
Romanzo	1	8	8	8

Figura 15.1.24: Risultato della Ricerca

Ora, diamo uno sguardo al codice SQL generato da OOo:

```
SELECT

    `tbargomenti`.`ArgDes` AS `Argomento`,
    COUNT( `tbmedia`.`MId` ) AS `Numero`,
    AVG( `tbmedia`.`MPrezzo` ) AS `Prezzo Medio`,
    MAX( `tbmedia`.`MPrezzo` ) AS `Prezzo Massimo`,
    SUM( `tbmedia`.`MPrezzo` ) AS `Valore Totale`

FROM `mediateca`.`tbargomenti` `tbargomenti`,
    `mediateca`.`tbmedia` `tbmedia`

WHERE ( `tbargomenti`.`ArgId` = `tbmedia`.`ArgId` )

GROUP BY `tbargomenti`.`ArgDes`
```

Le Funzioni di Raggruppamento, in SQL sono appunto ***COUNT***, ***AVG*** etc. Notate anche la clausola ***WHERE*** che contiene la relazione, e l'istruzione ***GROUP BY*** che consente di raggruppare secondo i valori contenuti in un campo.

15.1.9 Funzioni nei campi

Nelle espressioni di campo si possono usare anche funzioni, purché siano compatibili con il motore di database. Ad esempio:

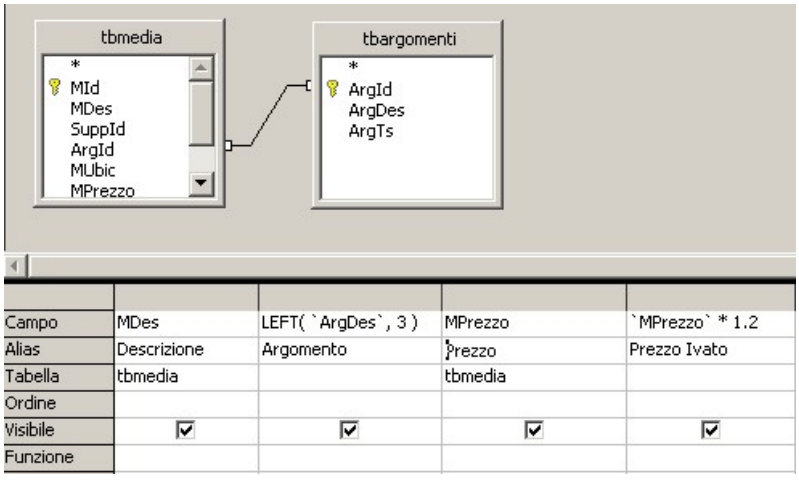


Figura 15.1.25: Ricerca con funzioni nei campi

ha come risultato:

	Descrizione	Argomento	Prezzo	Prezzo Ivato
▶	Joss Stone - The Soul Session	Pop	16,00	19,2
	Bad Boys II	Avv	16,00	19,2
	KDE 3.4 - Novità Major Release	Lin	11,50	13,8
	Montalbano - Il Ladro di merendine	Gia	8,00	9,6
	Benni - La compagnia dei celestini	Rom	8,00	9,6
	Guccini - Ritratti	Pop	16,00	19,2
	Il Signore degli Anelli	Fan	16,00	19,2
	Montalbano - La forma dell'acqua	Gia	8,00	9,6
	Pearl Jam - Ten	Roc	16,00	19,2

Figura 15.1.26: Risultato della Ricerca

Nel campo "*Argomento*" abbiamo riportato solo i primi tre caratteri grazie alla funzione ***LEFT***, e poi abbiamo calcolato il *Prezzo Iva Inclusa* (assolutamente ipotetico, a solo titolo di esempio) moltiplicando il campo *Mprezzo* per 1.2 (20%). In SQL:

```
SELECT

    `tbmedia`.`MDes` AS `Descrizione`,
    LEFT(`ArgDes`,3) AS `Argomento`,
    `tbmedia`.`MPrezzo` AS `Prezzo`,
    `MPrezzo` * 1.2 AS `Prezzo Ivato`

FROM `mediateca`.`tbargomenti` `tbargomenti`, `mediateca`.`tbmedia` `tbmedia`

WHERE (`tbargomenti`.`ArgId` = `tbmedia`.`ArgId`)
```

15.1.10 Trattamento estetico

Un aspetto senz'altro positivo del Modulo *Base* è che permette di "trattare" dal punto di vista estetico le nostre Ricerche. Non è che si possa fare moltissimo, ma tutte le modifiche vengono salvate, ed alla riapertura la nostra Ricerca ci riappare bella come prima.

La prima cosa che si fa, di solito, è adattare la la larghezza delle colonne al contenuto: la Ricerca si presenta come un piccolo Foglio Elettronico, e si procede esattamente allo stesso modo, cioè trascinando i margini della colonna stessa nella intestazione. Procedendo per analogia, un doppio clic sull'intestazione regola automaticamente la larghezza in base al contenuto più ampio, e questo torna utile.

Per selezionare una colonna basta un solo clic sempre sull'intestazione; dal menu contestuale si può scegliere la voce "*Formattazione Colonna*", che apre una maschera come questa:

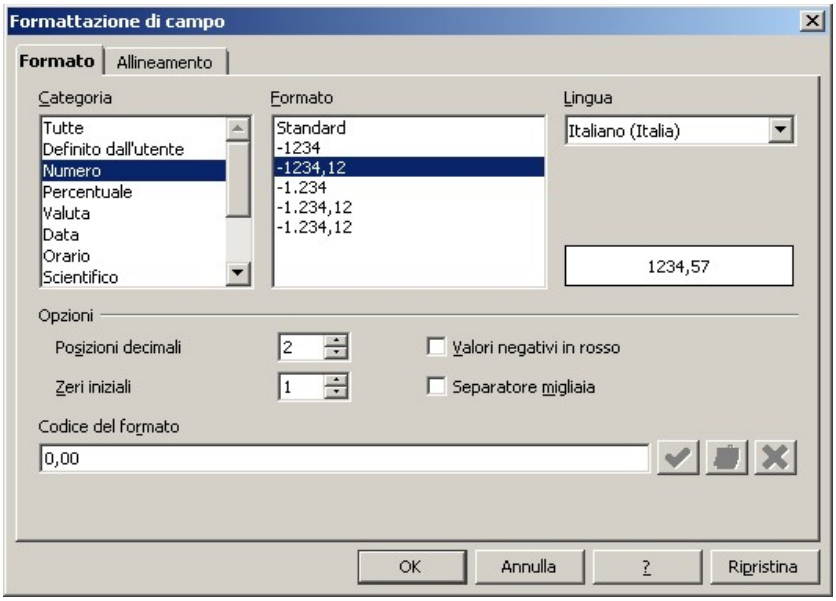


Figura 15.1.27: Formattazione di una Colonna

In realtà per le *Colonne di Testo* si può fare poco, ma per i *Numeri* abbiamo tutto quello che serve. Anche l'altezza della riga è modificabile, col solito sistema di trascinare il bordo. Infine si può nascondere una o più colonne. Non esiste modo (o almeno io non l'ho trovato) di intervenire sul formato del carattere, quindi dovremo accontentarci dei Font di sistema.

15.2 Formulari

La modalità di progettazione dei Formulari in OOo è abbastanza potente, anche se alcune interessanti funzionalità sono così ben nascoste da poterle trovare solo per caso. Per illustrare meglio l'uso avanzato dei Formulari useremo la Tabella degli *Utenti*, che, lo ricordo, abbiamo definito così:

In ***MySql***:

```
CREATE TABLE `tbutenti` (

  `UtId` int(10) unsigned NOT NULL auto_increment,
  `UtDen` varchar(50) NOT NULL default '',
  `UtVia` varchar(50) default NULL,
  `UtCit` varchar(100) default NULL,
  `UtTel` varchar(20) default NULL,
  `UtTs` timestamp NOT NULL default CURRENT_TIMESTAMP
       on update CURRENT_TIMESTAMP,
  `UtDNas` date NOT NULL default '0000-00-00',
  `UtImg` mediumblob,
  `UtCodFis` varchar(16) default NULL,
  `UtSesso` char(1) default 'M',
  `UtTessera` tinyint(1) NOT NULL default '0',

  PRIMARY KEY  (`UtId`),
  KEY `Den` (`UtDen`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Oppure, in ***PostGres*** :

```
CREATE TABLE "TbUtenti"
(
  "UtId" int4 NOT NULL DEFAULT nextval('public."TbUtenti_UtId_seq"'::text),
  "UtDen" varchar(50) NOT NULL DEFAULT ' '::character varying,
  "UtVia" varchar(50),
  "UtCit" varchar(100),
  "UtTel" varchar(20),
  "UtTs" timestamp,
  "UtDNas" date,
  "UtImg" bytea,
  "UtCodFis" varchar(16),
  "UtSesso" char(1) DEFAULT 'M'::bpchar,
  "UtTessera" int2 NOT NULL DEFAULT 0,
  CONSTRAINT "PKey" PRIMARY KEY ("UtId")
)
WITH OIDS;
```

cioè, tradotto in linguaggio umano:

Campo	Tipo di Campo	Commenti	Idx
UtId	Integer auto_increment	Identificatore - Chiave primaria	Pk
UtDen	Varchar(50)	Denominazione max 50 caratteri	*
UtVia	Varchar(50)	Indirizzo max 50 caratteri	
UtCit	Varchar(100)	Città max 100 Caratteri	
UtTel	Varchar(20)	Telefono max 20 Caratteri	
UtTs	Timestamp	Timestamp per la tabella	
UtDNas	Data	Data di Nascita	
UtImg	Immagine	Foto dell'Utente	
UtCodFis	Varchar(16)	Codice Fiscale	
UtSesso	Char(1)	Sesso (M o F default M)	
UtTessera	Integer(1)	Tesserato (valore Logico, 0 o1)	

Cominciamo a mostrare il risultato finale, in modo da semplificare il processo di costruzione del Formulario. Dunque, la Maschera per i nostri Utenti dovrebbe essere più o meno questa:

Figura 15.2.1: La Maschera per l'Archivio Utenti

Ovviamente era necessario un archivio di prova, ed ho pensato a qualche grande musicista del passato decisamente impossibilitato a richiedere i diritti d'autore per l'uso dell'immagine (*sempre che il Parlamento Europeo oppure il nostro Governo non abbiano approvato l'allungamento del periodo a 500 anni, cosa, visti i tempi che corrono, tutt'altro che improbabile*). Come potete constatare il risultato è discreto; vediamo come è stato ottenuto descrivendo brevemente i campi che compongono la nostra maschera ed i passi seguiti per costruirla.

I *Formulari* in OOo Base possono essere creati attraverso una procedura automatica oppure *manualmente*. La prima possibilità è comoda, ma dopo un po' si preferisce (almeno a me è successo così) fare da se. Dunque, si sceglie dal pannello sinistro la voce *Formulari* e quindi dal pannello superiore *Crea Formulario in vista struttura*. Ci ritroveremo con un bel foglio vuoto, opportunamente *grigliato* (cioè munito di griglia), dove costruire la nostra maschera.

Vi ricordo che per aggiungere campi ad una maschera è necessario selezionare prima il tipo di campo desiderato dalla Barra dei simboli *Controlli per formulario*.

Figura 15.2.2: La Barra dei Controlli per il Formulario

Questa Barra elenca i controlli più comuni; controlli aggiuntivi possono essere selezionati col pulsante *Altri Campi di Controllo* della stessa Barra (quello alla destra di *ABC*):

Figura 15.2.3: Altri Campi di Controllo per il Formulario

La prima cosa da fare è creare un bel campo di testo per il *Cognome* e *Nome*, quindi selezioniamo *Campo di Testo* e tracciamo un rettangolo sul foglio vuoto. Però abbiamo forse saltato un passaggio.... Non dovremmo prima collegare la maschera all'archivio, in modo da poter specificare le corrispondenze tra campi maschera e campi del Database ? Giusto, quindi selezioniamo il rettangolo appena tracciato e col tasto destro scegliamo *Formulario...*: le impostazioni necessarie sono mostrate in figura:

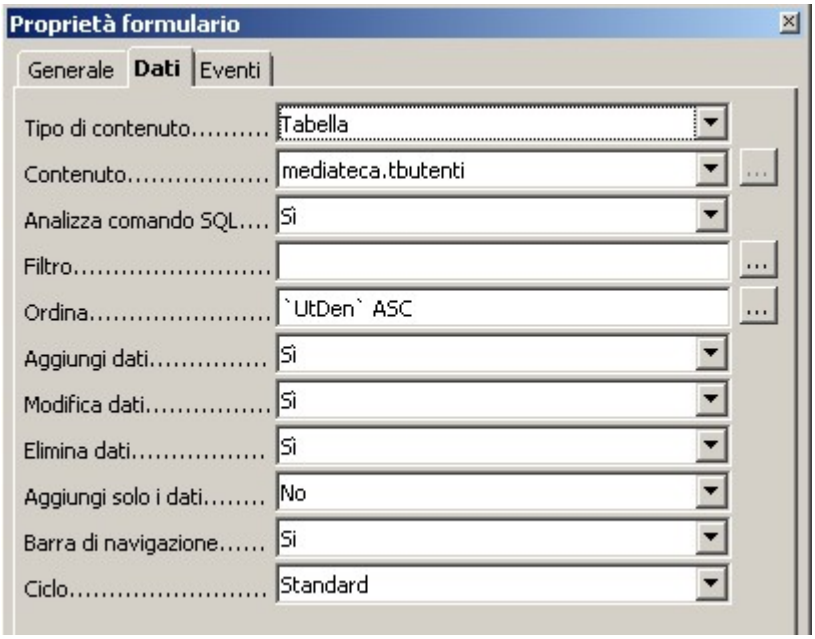


Figura 15.2.4: Proprietà del Formulario

TIP

Nelle Proprietà del Formulario possono essere stabilite alcune caratteristiche interessanti; la più utile è quella che permette di stabilire un ordinamento nella visualizzazione dei record senza usare una query. Il pulsante di auto composizione accanto alla voce *Ordina* permette di scegliere facilmente il campo od i campi da usare per l'ordinamento, come in figura.

Il campo *Cognome e Nome* è un semplice Campo di Testo; con il tasto destro e la voce *Campo di Controllo* apriamo la finestra delle proprietà: dobbiamo, nel pannello *Dati* assegnare il *Campo di Dati* (in questo caso UtDen), nella pannello *Generale* stabilire un nome (ad

esempio *MUtDen*) e, soprattutto, *la lunghezza massima della stringa di input,* che dovrebbe coincidere con la lunghezza del campo del Database (50 caratteri).

Nello stesso pannello è opportuno anche assegnare un tipo di carattere leggibile: *Tahoma* o *Verdana* a 10 punti direi che vanno bene. Ora associamo un'etichetta descrittiva, selezionando il tipo di campo *Testo Fisso*: il procedimento è semplice, quindi evito di dilungarmi. Il risultato è:

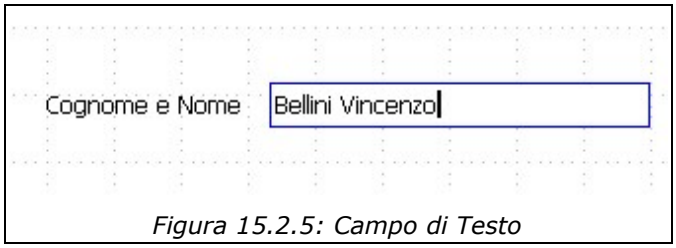


Figura 15.2.5: Campo di Testo

Ma queste sono cose che abbiamo già visto, quindi procediamo senza indugio....

15.2.1 Campi Data

Ci serve un campo per la Data di Nascita, ed i campi data si selezionano sulla Barra "*Altri Campi di controllo*". Tracciamo, come prima, un bel rettangolo e guardiamo le proprietà; quelle interessanti sono:

- il **Controllo Formato**, che se impostato a *Sì* (consigliato), evita l'immissione, nelle date, di caratteri non ammessi
- l'intervallo **Data Min / Data Max**, impostato tra il 1800 ed 2200, di solito più che sufficiente, ma non nel nostro caso, perché *Rossini* è nato nel *1792*
- il **Formato Data**, da me impostato su "*Standard Lungo*" (mostra anche il giorno della settimana), ma dipendente dai vostri gusti
- la proprietà **Apribile** che permette la selezione della data con un mini calendario che viene attivato da un pulsante sulla destra

Risultato:

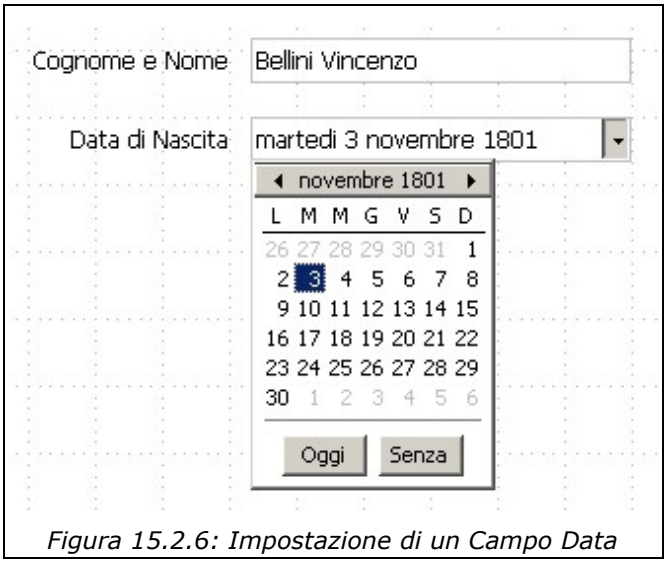


Figura 15.2.6: Impostazione di un Campo Data

15.2.2 Caselle di Controllo

Una *Casella di Controllo* (***CheckBox***) è il tipico quadratino col segno di spunta, che di solito significa *Si* o *No*. Nel nostro esempio ho introdotto un campo di nome *Tesserato* che dovrebbe indicare il possesso di una Tessera per accedere alla nostra Mediateca. Il controllo si trova nella Barra *Controlli per Formulario*. Nel pannello *Dati* è necessario impostare esplicitamente il valore per lo stato *On* (selezionato) e per quello *Off* (deselezionato), come in figura.

Figura 15.2.7: Tab Dati di una Casella di Controllo

TIP

Per convenzione i valori booleani possono essere rappresentati con un numero intero: il valore *Zero* significa *Falso* o *No*, qualsiasi valore diverso da *Zero* (di solito *1*, ma a volte *-1*) significa *Vero* o *Si*.

15.2.3 Campi a Maschera

Nei Formulari è a volte indispensabile stabilire delle *Maschere di Immissione* per alcuni tipi di dati. Queste *Maschere* (potremmo anche chiamarli *formati di immissione*) stabiliscono appunto il *formato* dell'informazione digitata. Nel nostro esempio abbiamo il *Codice Fiscale*, che, come sappiamo tutti, ha un formato del tipo

CCC CCC NNCNN CNNNC

dove con C indichiamo un carattere alfabetico e con N una cifra da 0 a 9. Il controllo *Campo a Maschera* permette di specificare appunto quella che OOo chiama una *Maschera di digitazione* per agevolare la scrittura del dato ed evitare errori. Per il *Codice Fiscale*:

Figura 15.2.8: Un Campo a maschera

Le regole per costruire una Maschera di digitazione sono piuttosto semplici, e ben illustrate nella Guida, quindi non spreco altro spazio.

15.2.4 Gruppi di Opzioni e Pulsanti di Scelta

Per la scelta del *Sesso*, per motivi didattici ho optato per un *Gruppo di Opzioni*. Questo controllo si trova sulla Barra *Altri controlli per il Formulario*, e, se è attiva l'auto composizione, fa partire un comodo Pilota Automatico che vi guida attraverso la corretta definizione dei parametri. Questa strada è senz'altro comoda, ma alla fine vi accorgerete che:

- il controllo *Gruppo di Opzioni* non è che un quadrilatero con un titolo, senza collegamenti ai campi dati
- all'interno del quadrilatero possono essere inseriti altri controlli collegati ai campi del Database

Nel nostro esempio abbiamo un *Gruppo di Opzioni* (*sesso*) che contiene due *Pulsanti di Scelta*, esattamente *Maschio* e *Femmina*. Un *Pulsante di scelta* è un po' come una *Casella di controllo*, che permette di assegnare un valore diverso nel caso sia *On* oppure *Off*. Noi abbiamo in precedenza stabilito che il campo *UtSesso* può assumere i valori "*M*" o "*F*". Quindi, per *Maschio*:

Figura 15.2.9: Pulsante di scelta per il campo "Maschio"

Per il campo *Femmina* basta cambiare il *Valore di riferimento (on)* in "*F*".

Figura 15.2.10

15.2.5 Controllo Immagine

In alcuni casi è necessario archiviare nei Database informazioni che non sono riconducibili al puro testo. Molti Server di Db, come abbiamo visto, usano una tipologia particolare di Dati per questo tipo di informazioni: in MySql ***mediumblob***, in PostgreSQL ***bytea***. Questi campi possono, in generale, ospitare qualsiasi sequenza di caratteri, ma sono particolarmente utili nel caso di immagini.

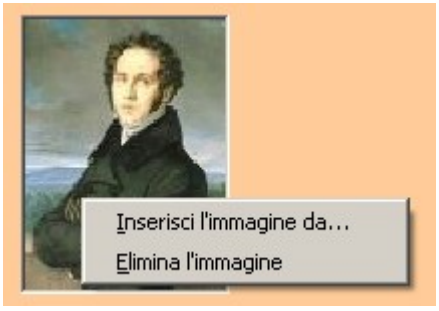


Figura 15.2.11: Menu contestuale per il controllo immagine

OOo Base dispone di un controllo chiamato *Campo di controllo immagine* che può, in questi casi, tornare utile. Il controllo è un semplice rettangolo che può essere posizionato ovunque nella maschera. Possiede caratteristiche banali: l'unica degna di nota è *Scala*, che permette di adeguare l'immagine allo spazio che abbiamo riservato. Deve essere ovviamente collegato, nel nostro caso, al Campo dati *UtImg*. In fase di immissione dei dati, l'immagine può essere caricata nel Database con un doppio clic sul controllo: si apre un selettore di file ed è possibile specificare un file grafico. Per quanto dovrebbe essere possibile utilizzare qualsiasi formato

grafico supportato dal sistema, quello che crea meno problemi è *jpg*. In alternativa è disponibile un menu contestuale aperto col tasto destro con il mouse posizionato sul campo.

A differenza di altri prodotti concorrenti, il controllo non supporta l'OLE, e quindi non è possibile "incollare" una immagine "tagliata" da un'altra applicazione.

15.2.6 Altri Controlli per il Formulario

Per l'immissione di alcune tipologie di dati OOo prevede dei controlli dedicati: *Campo Numerico*, *Campo di Valuta*, *Campo Formattato*. L'uso è piuttosto semplice, quindi eviterò di parlarne lasciandovi il piacere della scoperta.

OOo dispone inoltre una serie di altri controlli non collegabili direttamente ai Campi di Db che permettono però un buon grado di interattività nella progettazione dei Formulari: vari tipi di *Pulsanti*, una *Barra di Scorrimento*, la *Barra di Navigazione* etc. Questi controlli vanno di solito associati ad eventi e macro, quindi saranno descritti in modo più approfondito nei paragrafi seguenti.

15.3 Rapporti

In questa fase di sviluppo di OpenOffice.org 2.0, la parte dedicata ai Rapporti del Modulo Base è quella meno avanzata. Come abbiamo visto, un Rapporto si può costruire solo con l'auto composizione, e gli interventi successivi possibili sono troppo limitati.

Torneremo a parlare dei Rapporti probabilmente quando sarà disponibile la versione 2.0 definitiva.

16. Base e i suoi Colleghi....

Una Suite da ufficio che si rispetti dovrebbe garantire un buon grado di interattività tra i suoi moduli. Microsoft sfrutta a fondo l'OLE, ma questa è una prerogativa unica di Windows che mal si adatta ad un prodotto multi piattaforma. Nei paragrafi seguenti quindi vedremo come è possibile utilizzare il Dati provenienti da documenti di tipo Base con gli altri moduli di OpenOffice.org.

16.1 Registrare il Database

Come abbiamo già avuto modo di dire, l'accesso ai dati dai moduli di OOo avviene aprendo il pannello delle ***Sorgenti Dati*** (*Visualizza -> Sorgenti Dati*) oppure utilizzando il tasto funzione **F4**. Perché una Sorgente Dati sia *visualizzata (*e quindi utilizzabile*),* è però necessario che sia stata preventivamente *registrata*; in pratica si tratta di indicare ad OOo qual'è il Documento Base che utilizzeremo come sorgente. Questo si ottiene con la voce *Strumenti -> Opzioni -> OpenOffice.org Base -> Database*; qui, col pulsante *Nuovo*, è possibile indicare il percorso del File *Base* da registrare. A questo punto, con il tasto **F4** avremo accesso alle Tabelle ed alle Ricerche del nostro Db.

16.2 Writer

Writer è il modulo di *Elaborazione Testi* di OpenOffice, ed è utile, insieme al Database, sia per il Mail Merge (Lettere personalizzate) sia come tool di *Reporting* (purtroppo però solo statico). Vediamo....

16.2.1 Lettere Personalizzate

L'uso di Writer come strumento per Lettere Personalizzate è assai intuitivo. Esiste anche un'auto composizione, ma secondo me è più facile fare "a mano". Per creare questo tipo di Documenti, in effetti, basta seguire questi semplici passi:

- ✔ Registrare il Documento OOo Base che serve da sorgente dati;
- ✔ aprire un documento di testo vuoto;
- ✔ aprire il pannello delle *Sorgenti Dati* con ***F4***, e selezionare la Ricerca o la Tabella che intendiamo utilizzare;
- ✔ trascinare col mouse, dal pannello di destra, la *colonna* che ci serve nel punto del Documento di Testo dove vogliamo che appaia il campo.

Il tutto è visibile nella figura in basso, dove possiamo usare l'anagrafica degli Utenti per scrivere una Lettera personalizzata.

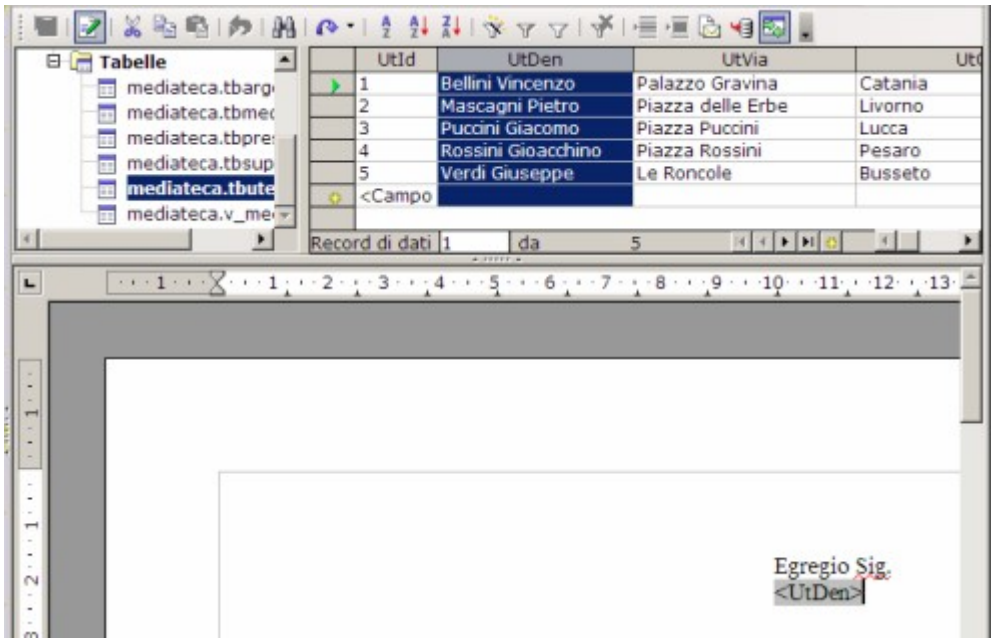


Figura 16.2.1: Trascinare una colonna nel documento di testo

Ovviamente è possibile trascinare quante colonne si desidera nel Documento di Testo. Se vogliamo vedere "come viene", possiamo selezionare un Record e premere il pulsante ***Dati in Campi*** della barra degli strumenti della Sorgente Dati ( ), come in Figura.

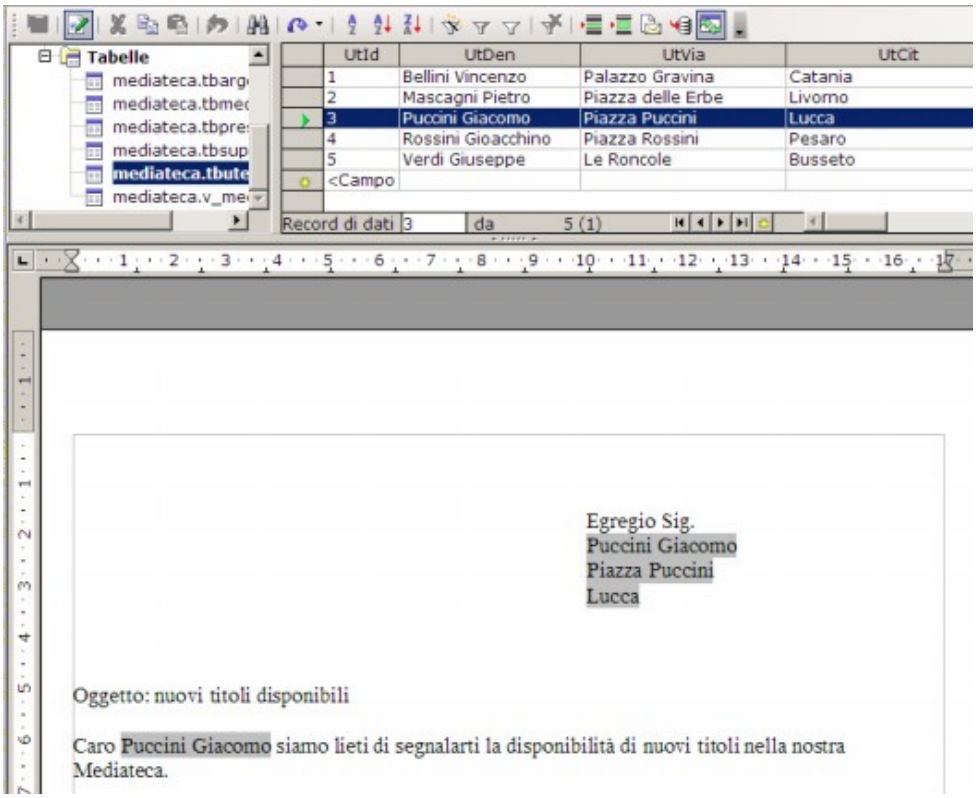


Figura 16.2.2: L'anteprima della Stampa in Serie

Quando si avvia una stampa, OOo avverte che si tratta di una Lettera Personalizzata e chiede se si vuole eseguire una stampa in serie: le opzioni sono piuttosto intuitive, quindi non mi dilungo oltre.

16.2.2 Stampa in serie da Ricerche

Se è necessario filtrare i dati di una Tabella per una Stampa in Serie, la cosa più saggia è usare una Ricerca, per evitare di impostare complesse opzioni nel Documento di Testo. Creiamo, perciò una Ricerca nel nostro Documento Base per la stampa di un messaggio ai soli Tesserati (*UtTessera=1*), col nome di ***Lettere_Tesserati***.

Campo	UtDen	UtVia	UtCit	UtSesso	UtTessera
Alias					
Tabella	tbutenti	tbutenti	tbutenti	tbutenti	tbutenti
Ordine	crescente				
Visibile	☑	☑	☑	☑	☑
Funzione					
Criteri					<> 0

Figura 16.2.3: La Ricerca Lettere_Tesserati

Riapriamo il nostro documento, e con la Voce di Menu *Modifica -> Scambia Database* selezioniamo la nuova Sorgente Dati. Fatto, ora siamo pronti a spedire il documento solo ai Tesserati.

16.2.3 Stampa in Serie con Campi Condizionali

Siccome siamo molto esigenti, vorremmo che, a seconda del *Sesso* del nostro Tesserato, la formula di saluto fosse diversa: "Egregio Signor" per i maschietti e "Gentile Signora" per le femminucce. Questo si può ottenere inserendo un Comando di Campo di tipo Testo Condizionale: abbiamo, infatti, opportunamente aggiunto il campo *UtSesso* alla ricerca. La cosa è piuttosto semplice (tenete presente che, in questo esempio, il Documento Base si chiama Mediateca_My5) :

- ✔ Scegliamo la Voce di Menu *Inserisci -> Comando di Campo -> Altro*;
- ✔ selezioniamo la Tab *Funzioni* ed il tipo di *Campo Testo Condizionale*;
- ✔ alla voce ***Condizione***, scriviamo: *Mediateca_My5.Lettere_Tesserati.UtSesso EQ "M"*;
- ✔ alla voce ***Poi*** : *Egregio Signor*
- ✔ alla voce ***Altrimenti*** : *Gentile Signora*.

Quindi in *Condizione* è possibile fare riferimento ad un Campo della Tabella o della Ricerca nella forma *Database.Tabella.NomeCampo;* nel nostro caso

Mediateca_My5.Lettere_Tesserati.UtSesso

seguito dall'operatore (***EQ*** sta per uguale, ma ce ne sono molti altri) e dal valore da confrontare ("***M***" per maschio).

Figura 16.2.4: Impostazione di Testo Condizionale

Il risultato è soddisfacente.....

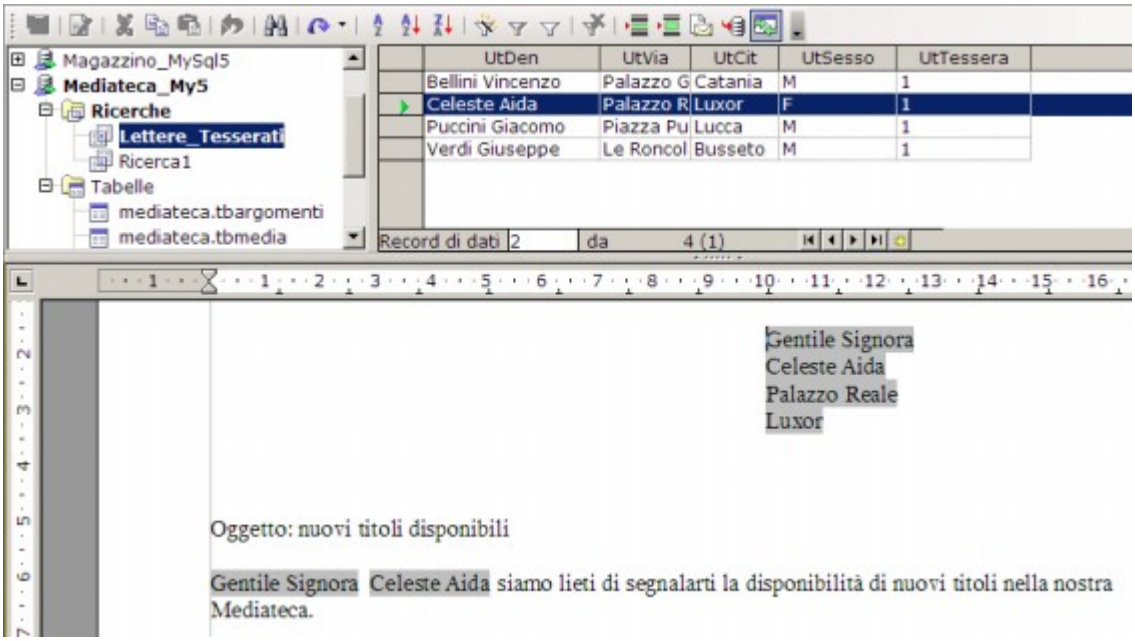


Figura 16.2.5: Stampa in serie con Testo Condizionato

16.2.4 Aggiunta di Tabelle da Ricerche

Supponiamo ora di voler elencare, all'interno della nostra Lettera per Stampa in Serie, una Tabella contenente, ad esempio, tutti i titoli relativi ai *CD Audio Rock*. Come al solito la prima cosa da fare è crearsi una Ricerca nel Documento Base che soddisfi le nostre esigenze. Un esempio potrebbe essere quello in figura, che ho chiamato ***CD_Rock***.

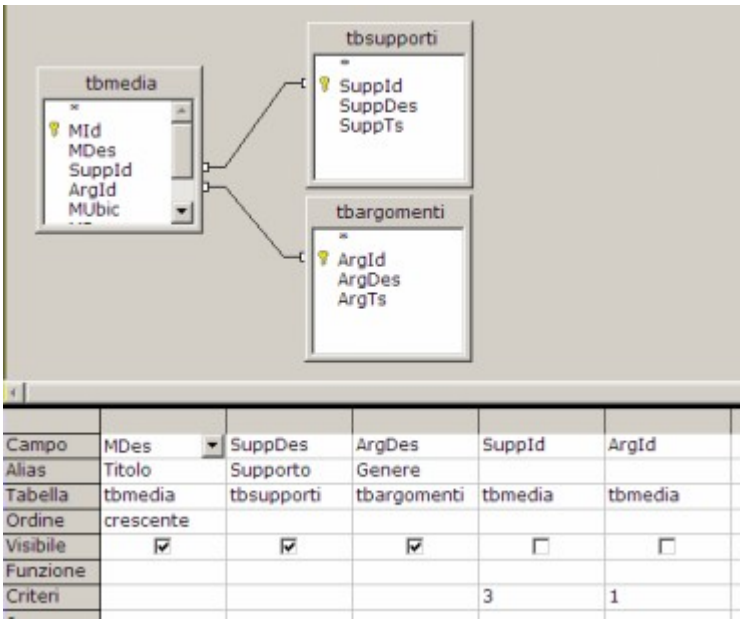


Figura 16.2.6: Ricerca dei Cd Rock

Ok, allora torniamo al nostro documento e, dopo aver aperto il pannello Sorgenti Dati con ***F4***, trasciniamo col mouse la nuova ricerca nel nostro documento. Si apre una finestra dove è necessario impostare alcuni parametri:

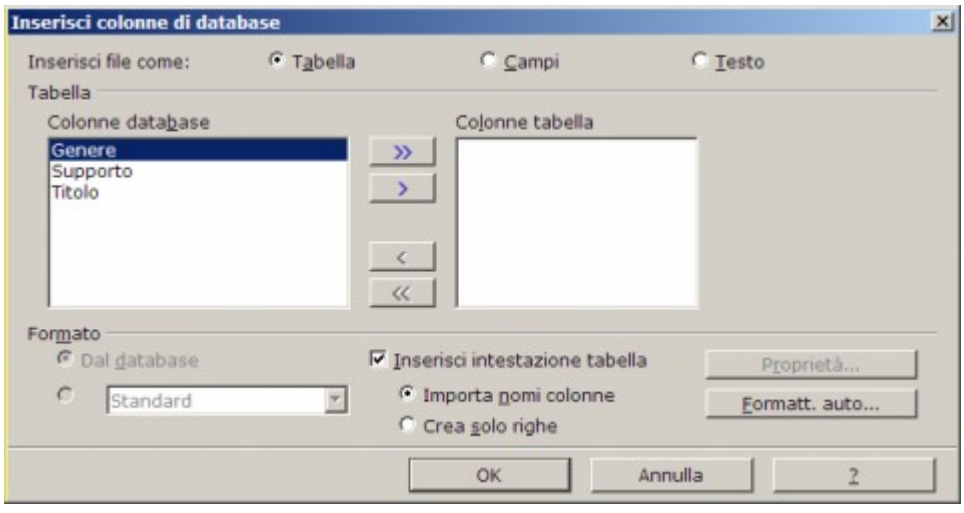


Figura 16.2.7: Inserire una Ricerca come Tabella

Notate che il *nome dei Campi* è riportato in ordine alfabetico, quindi è necessario spostare gli elementi che ci servono nella colonna a destra nella giusta sequenza (nel nostro caso *Titolo*, *Supporto*, *Genere*).

Otterremo più o meno questo:

Gentile Signora
Celeste Aida
Palazzo Reale
Luxor

Oggetto: nuovi titoli disponibili

Gentile Signora Celeste Aida siamo lieti di segnalarLe la disponibilità di nuovi titoli nella nostra Mediateca.

Titolo	*Supporto*	*Genere*
Afterhours - Ballate per piccole iene	CD Audio	Rock
De Gregori - Pezzi	CD Audio	Rock
Ligabue - Roma Stadio Olimpico	CD Audio	Rock
Pearl Jam - Ten	CD Audio	Rock

Figura 16.2.8: La Tabella ricavata da una Ricerca

Un punto deve essere chiaro: i *Campi* immessi come *Comandi di Campo* sono ***variabili***, cioè sono collegati al Database (e quindi alla Ricerca) di origine; le *Tabelle* trascinate nei documenti sono statiche, cioè rispecchiano la situazione di quel preciso momento. Perciò se in futuro aggiungeremo un Tesserato, potremo utilizzare lo stesso Documento senza modifiche, mentre se disporremo di nuovi Cd Rock sarà necessario trascinare di nuovo la Tabella nel Documento.

TIP

Quando si apre un Documento Base nel Pannello delle Sorgenti Dati, lo stesso risulta in uso al Documento Writer (o Calc) a cui è collegato. Perciò, ad esempio, se apro *anche* Mediateca_My5 (il documento Base) in un'altra finestra, OOo mi impedisce di modificare la struttura delle Tabelle e delle Ricerche. In questo caso è necessario riavviare OpenOffice, compreso il QuickStart, per risolvere il problema. Questo credo sia un piccolo bug che sarà corretto nelle prossime versioni.

16.3 Calc

Abbiamo già visto, nel capitolo *Rapporti... Old Style*, come collegare una Sorgente Dati con un Foglio Elettronico. Con la stessa tecnica è possibile ottenere buoni risultati anche per rappresentare in forma grafica informazioni di un Database, oppure per calcolare Totali Parziali su Raggruppamenti.

16.3.1 Grafici da Sorgenti Dati

Riprendiamo la Ricerca, già creata al Paragrafo *Raggruppamenti e Formule Matematiche* del Capitolo *Uso Avanzato di....*, che abbiamo chiamato ***Analisi_Argomenti*** e definito così:

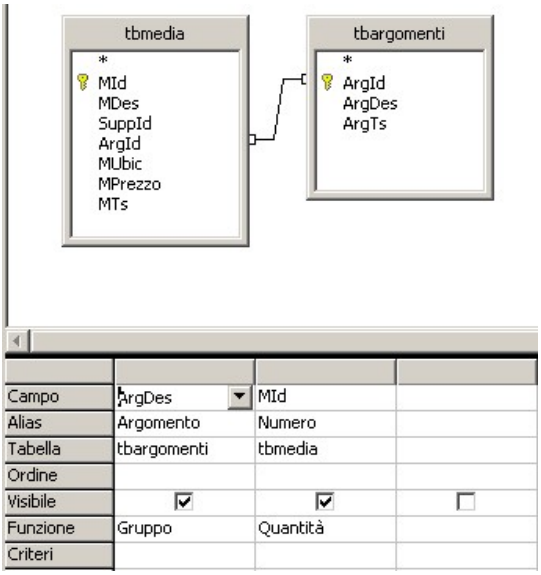


Figura 16.3.1: Ricerca con uso di Gruppi

Se "trasciniamo" la Ricerca su un Foglio elettronico vuoto e selezioniamo i parametri già visti per l'*Area di Importazione (Dati->Definisci Area)*, è abbastanza semplice creare un Grafico associato ai Dati della Ricerca stessa, come in figura. Notate che al variare delle informazioni contenute nel Documento Base, varierà di conseguenza anche il Grafico.

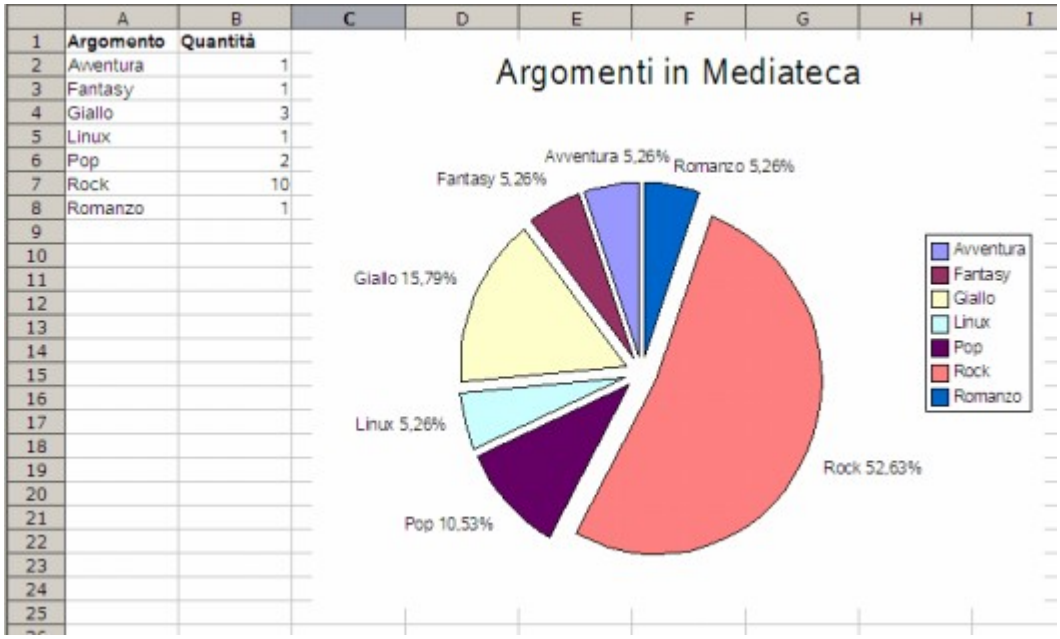


Figura 16.3.2: Rappresentazione Grafica dei Dati

16.3.2 Calcolo di Subtotali

Supponiamo di necessitare di un elenco del nostri Media, ordinato per Tipo di Supporto, che calcoli anche il costo totale per ogni tipologia (lo so che mi invento cose improbabili, ma serve solo da esempio...). Cominciamo, al solito, a costruire la nostra ricerca, che potrebbe essere:

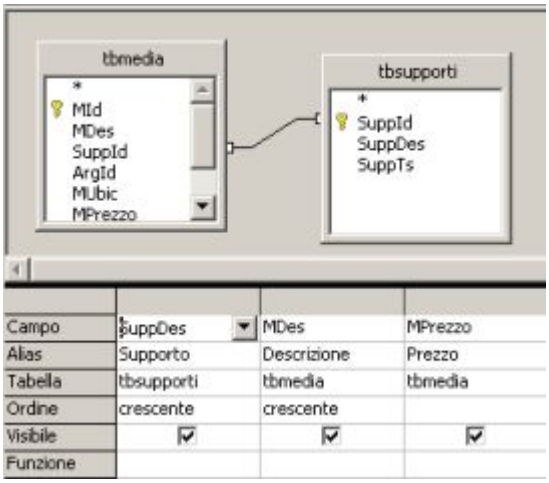


Figura 16.3.3: Ricerca per i SubTotali

Al solito, "trasciniamo" la Ricerca su un Foglio di Calc, impostando nel modo corretto i parametri per l'Area; avremo:

	A	B	C
1	**Supporto**	**Descrizione**	**Prezzo**
2	CD Audio	Afterhours - Ballate per piccole iene	€ 16,00
3	CD Audio	De Gregori - Pezzi	€ 16,00
4	CD Audio	Guccini - Ritratti	€ 16,00
5	CD Audio	Joss Stone - The Soul Session	€ 16,00
6	CD Audio	Ligabue - Roma Stadio Olimpico	€ 16,00
7	CD Audio	Pearl Jam - Ten	€ 16,00
8	DVD Video	Bad Boys II	€ 16,00
9	DVD Video	Il Signore degli Anelli	€ 16,00
10	DVD Video	Marillion - Marbles on the Roads	€ 16,00
11	DVD Video	Springsteen - Devils & Dust	€ 13,50
12	Libro	Benni - La compagnia dei celestini	€ 8,00
13	Libro	Cornwell - L'Ultimo Distretto	€ 10,00
14	Libro	Montalbano - Il Ladro di merendine	€ 8,00
15	Libro	Montalbano - La forma dell'acqua	€ 8,05
16	Rivista	KDE 3.4 - Novità Major Release	€ 11,50

Figura 16.3.4: La Ricerca nel Foglio di Calc

A questo punto basta selezionare l'Area (che, lo ricordo, di default si chiama ***Importa1***) e con la voce di Menu *Dati -> Subtotali* definiamo i parametri giusti come in figura:

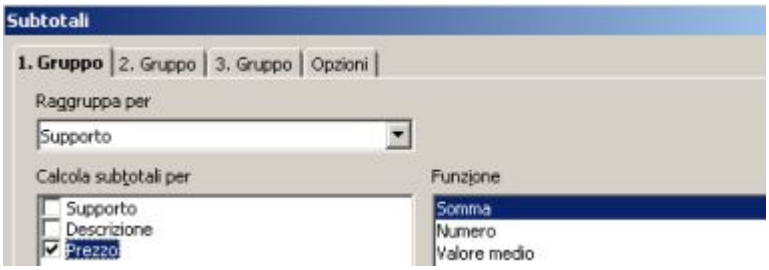


Figura 16.3.5: Impostazione dei Subtotali

per avere :

	A	B	C
1	**Supporto**	**Descrizione**	**Prezzo**
2	CD Audio	Afterhours - Ballate per piccole iene	€ 16,00
3	CD Audio	De Gregori - Pezzi	€ 16,00
4	CD Audio	Guccini - Ritratti	€ 16,00
5	CD Audio	Joss Stone - The Soul Session	€ 16,00
6	CD Audio	Ligabue - Roma Stadio Olimpico	€ 16,00
7	CD Audio	Pearl Jam - Ten	€ 16,00
8	***CD Audio Somma***		***€ 96,00***
9	DVD Video	Bad Boys II	€ 16,00
10	DVD Video	Il Signore degli Anelli	€ 16,00
11	DVD Video	Marillion - Marbles on the Roads	€ 16,00
12	DVD Video	Springsteen - Devils & Dust	€ 13,50
13	***DVD Video Somma***		***€ 61,50***
14	Libro	Benni - La compagnia dei celestini	€ 8,00
15	Libro	Cornwell - L'Ultimo Distretto	€ 10,00
16	Libro	Montalbano - Il Ladro di merendine	€ 8,00
17	Libro	Montalbano - La forma dell'acqua	€ 8,05
18	***Libro Somma***		***€ 34,05***
19	Rivista	KDE 3.4 - Novità Major Release	€ 11,50
20	***Rivista Somma***		***€ 11,50***
21	***Totale***		***€ 203,05***

Figura 16.3.6: Subtotali con Calc

Anche in questo caso il risultato è "dinamico": ogni modifica ai Dati del Db comporta il corretto aggiornamento del Foglio Elettronico. Vi ricordo che col modulo integrato per i Rapporti di *Base* è, attualmente, *impossibile* utilizzare in modo flessibile i Subtotali, ma *Calc* è davvero un'ottima alternativa.

16.3.3 Ricerche a Campi incrociati

Lo so, lo so, il Modulo *Base* prevede solo Ricerche semplici, chiamate anche Query di Selezione in altri prodotti analoghi, quindi direttamente possiamo fare poco. Però abbiamo Calc, ed il suo DataPilot, quindi non tutto è perduto....

Cominciamo, per una volta, dalla fine: voglio un prospetto dove sia indicato *quanti* Supporti abbiamo di tipo diverso per ogni Argomento, cioè una cosa del genere:

	A	B	C	D	E	F	G	H	I	J
1	ContaNumeri - Descrizione	Argomento								
2	Supporto	Avventura	Classica	Fantasy	Giallo	Linux	Pop	Rock	Romanzo	**Totale Risultato**
3	CD Audio						3	4		**7**
4	DVD Video	1		1				2		**4**
5	Libro				3				1	**4**
6	MP3		1							**1**
7	Rivista					1				**1**
8	**Totale Risultato**	**1**	**1**	**1**	**3**	**1**	**3**	**6**	**1**	**17**

Figura 16.3.7: Una Query a Campi incrociati

Questo tipo di analisi dei dati viene definita "a campi incrociati" perché si indica un Campo di raggruppamento per le righe, uno per le colonne, ed i valori vengono calcolati in base ad una formula matematica. Si tratta di un sistema assai utile, ad esempio, quando si voglia calcolare l'andamento mensile di determinati valori (come le vendite oppure le scorte di magazzino). Ma torniamo indietro...

Il primo passo, al solito, è la creazione di una Ricerca adatta all'uopo; dobbiamo includere i campi che devono essere "incrociati", magari in ordine, ed un valore che possa essere "contato", quindi:

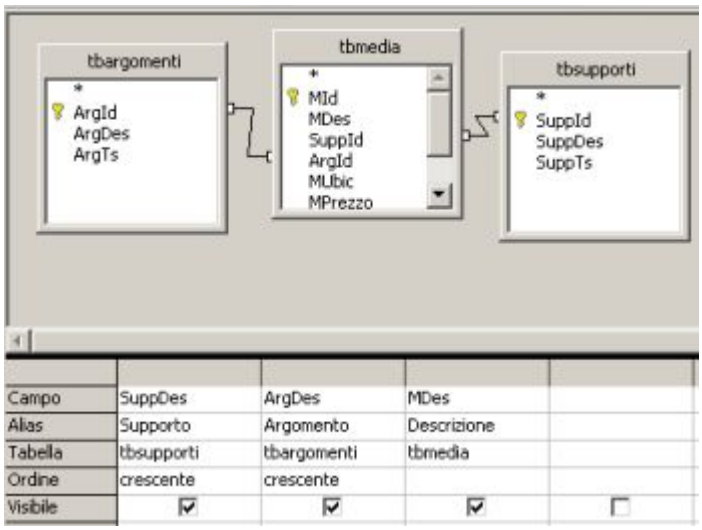


Figura 16.3.8: Ricerca per il DataPilot

Al solito, trasciniamo la Ricerca in un Foglio vuoto e definiamo l'area come abbiamo visto in precedenza. Avremo più o meno questo risultato:

	A	B	C
1	**Supporto**	**Argomento**	**Descrizione**
2	CD Audio	Pop	Guccini - Ritratti
3	CD Audio	Pop	Rolling Stones - A Bigger Bang
4	CD Audio	Pop	Joss Stone - The Soul Session
5	CD Audio	Rock	Ligabue - Roma Stadio Olimpico
6	CD Audio	Rock	Pearl Jam - Ten
7	CD Audio	Rock	Afterhours - Ballate per piccole iene
8	CD Audio	Rock	De Gregori - Pezzi
9	DVD Video	Awentura	Bad Boys II
10	DVD Video	Fantasy	Il Signore degli Anelli
11	DVD Video	Rock	Marillion - Marbles on the Roads
12	DVD Video	Rock	Springsteen - Devils & Dust
13	Libro	Giallo	Cornwell - L'Ultimo Distretto
14	Libro	Giallo	Montalbano - Il Ladro di merendine
15	Libro	Giallo	Montalbano - La forma dell'acqua
16	Libro	Romanzo	Benni - La compagnia dei celestini
17	MP3	Classica	Mozart - Sonate
18	Rivista	Linux	KDE 3.4 - Novità Major Release

Figura 16.3.9: La Ricerca nel Foglio Elettronico

Ok; ora selezioniamo l'area e scegliamo dal Menu ***Dati -> Datapilot -> Avvia***. La maschera di immissione dei parametri si presenta così:

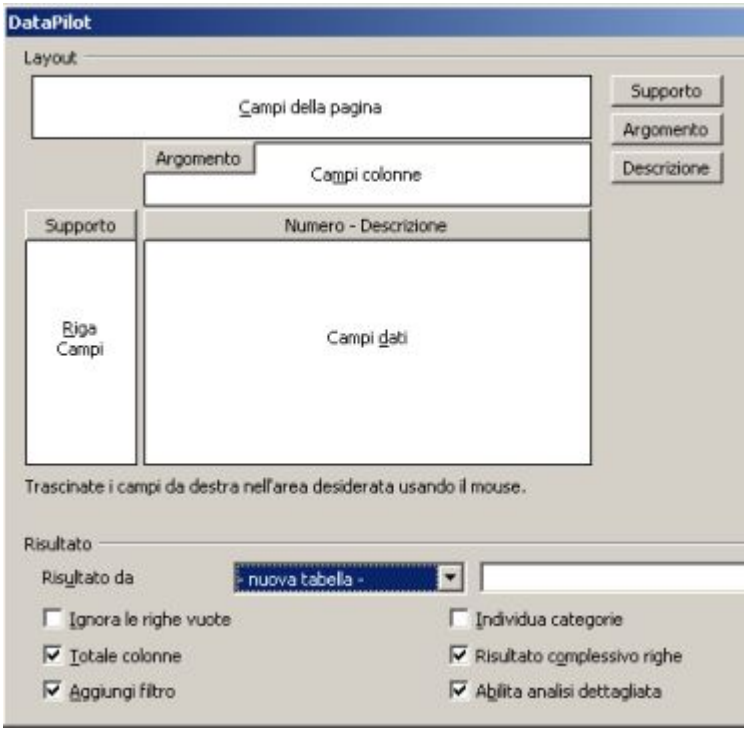


Figura 16.3.10: Impostazione di un DataPilot

Sulla destra ci sono i "pulsanti" che rappresentano i Campi della Tabella: dobbiamo trascinare i Campi al posto giusto. Così *Supporto* va sulla riga, *Argomento* sulla Colonna e *Descrizione* nel Campo dati. Ci resta solo da stabilire il metodo di calcolo per i Dati: con un doppio click su *Descrizione*, indichiamo *Conteggio*, come in figura:

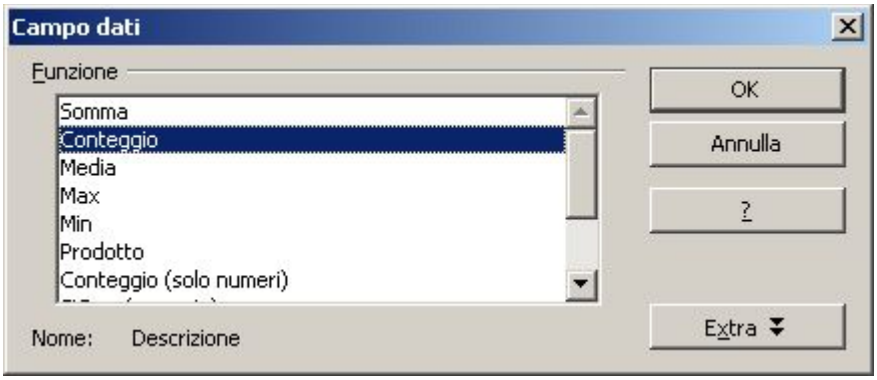


Figura 16.3.11: Indicazione del metodo di calcolo

Infine dobbiamo decidere dove "costruire" la nuova Tabella, e, per evitare confusioni, ho indicato nella voce *Risultato da* il valore *nuova tabella* (in alternativa è possibile usare un'altra area della stessa tabella). Quello che si ottiene è mostrato nella prima immagine di questo paragrafo.

Vi faccio ancora notare che il collegamento è "dinamico", cioè ogni modifica ai dati nel Db aggiorna anche i valori del foglio elettronico.

TIP

Se tutto è stato definito per bene, ad ogni apertura del Foglio di Calcolo OOo chiede se aggiornare le Ricerche: ovviamente bisogna rispondere di si. La Tabella ottenuta col DataPilot va, invece, aggiornata manualmente: basta posizionarsi su una cella qualsiasi dell'area e scegliere dal menu contestuale la voce Aggiorna.

17. Appendice A – Installazione di OpenOffice

17.1 Installazione in Windows

Dal Sito www.openoffice.org è possibile scaricare il pacchetto in formato eseguibile che permette l'installazione della versione 2.0 su tutte le versioni di Windows. La procedura è semplice, e permette l'utilizzo di OOo al solo Utente che esegue l'installazione o a tutti gli Utenti del Computer (scelta, ovviamente, consigliata).

17.2 Installazione in Linux

OOo 2.0 è ormai presente in tutte le più recenti distribuzioni Linux, quindi le difficoltà sono davvero minime. In ogni caso è possibile scaricare le versioni più aggiornate, purtroppo solo in formato RPM. Per le distro che non supportano questo formato, come Debian, è necessario affidarsi a strumenti di conversione.

18. Appendice B – Tipo di Dati di MySql

Riporto in questa appendice un riassunto dei tipi di dati utilizzabili con MySql 4.X. Vi ricordo che con ***M*** si indica il numero di caratteri (massimo 255) utilizzati per la visualizzazione. Con ***D***, per i campi numerici Decimali, si indica appunto il numero di cifre dopo la virgola: il massimo è 30, ma non può essere ovviamente più grande di M-2. Con ***UNSIGNED*** si indica che il numero può essere solo positivo. Con ***ZEROFILL*** si chiede al motore di Db di completarlo con cifre 0 a sinistra fino alla lunghezza massima. Ricordate che se un numero viene definito ZEROFILL, diventa anche UNSIGNED.

18.1 Valori di Tipo Numerico

TINYINT [(M)] [UNSIGNED] [ZEROFILL]
Il numero Intero più piccolo, con un range da **-128** a **127**. Il valore unsigned va da **0** a **255**.

BIT
BOOL
BOOLEAN
Queste definizioni sono sinonimi di **TINYINT(1)**. Il tipo **BOOLEAN** è stato aggiunto dalla versione 4.1.0.

SMALLINT [(M)] [UNSIGNED] [ZEROFILL]
Numero Intero breve. Il range va da **-32.768** a **32.767**. Il valore unsigned da **0** a **65.535**.

MEDIUMINT [(M)] [UNSIGNED] [ZEROFILL]
Numero Intero Medio. Il range va da **-8.388.608** a **8.388.607**. Il valore unsigned da **0** a **16.777.215**.

INT [(M)] [UNSIGNED] [ZEROFILL]
Valore Intero "classico". Il range va da **-2.147.483.648** to **2.147.483.647**. Il valore unsigned da **0** a **4.294.967.295**.

INTEGER[(M)] [UNSIGNED] [ZEROFILL] - Sinonimo per **INT**.

BIGINT [(M)] [UNSIGNED] [ZEROFILL]
Intero lungo. Il range va da **-9.223.372.036.854.775.808** a **9.223.372.036.854.775.807**. Il valore unsigned da **0** a **18.446.744.073.709.551.615**.

FLOAT(precisione) [UNSIGNED] [ZEROFILL]
Numero a virgola mobile. **precisione** può assumere un range da 0 a 24 per valori a precisione singola, e da 25 a 53 per valori a precisione doppia. Questi tipi di dati sono equivalenti a **FLOAT** e **DOUBLE** descritti subito dopo. **FLOAT(precisione)** possiede lo stesso range dei tipi corrispondenti **FLOAT** e **DOUBLE**, ma il numero di cifre da visualizzare ed il numero di decimali non sono definiti. Notate che l'uso di **FLOAT** potrebbe causare alcuni problemi, perchè i calcoli sono eseguiti sempre in precisione doppia. In effetti questa tipologia è prevista solo per compatibilità con ODBC.

FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]

Numero a virgola mobile in precisione singola. Il range va da **-3.402823466E+38** a **-1.175494351E-38**, **0**, e **1.175494351E-38** fino a **3.402823466E+38**. Se viene specificato **UNSIGNED**, i valori negativi non sono permessi.

DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]

Numero a virgola mobile in precisione doppia. I valori consentiti sono da **-1.7976931348623157E+308** a **-2.2250738585072014E-308**, **0**, e da **2.2250738585072014E-308** a **1.7976931348623157E+308**. Se viene specificato **UNSIGNED**, i valori negativi non sono permessi.

DOUBLE PRECISION[(M,D)] [UNSIGNED] [ZEROFILL]

REAL[(M,D)] [UNSIGNED] [ZEROFILL]

Sinonimi per **DOUBLE**.

DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]

Un numero "esplicito", con numero di cifre decimali fisse. Si mostra come una colonna di tipo **CHAR** : "esplicito" significa che il numero viene memorizzato come una stringa (quindi non trasformato), usando un carattere per ogni cifra del valore. Il punto decimale ed il carattere "-" per i numeri negativi non sono compresi nel valore **M**, ma il loro spazio è riservato automaticamente. Se **D** è zero, i valori non hanno né punto né cifre decimali. Il range massimo per **DECIMAL** è lo stesso del tipo **DOUBLE**, ma ovviamente può essere influenzato dalla scelta di **M** e **D**. Se viene specificato **UNSIGNED**, i valori negativi non sono permessi. Se **D** viene omesso, il default è 0. Se **M** è omesso, il default è 10.

DEC[(M[,D])] [UNSIGNED] [ZEROFILL]

NUMERIC[(M[,D])] [UNSIGNED] [ZEROFILL]

FIXED[(M[,D])] [UNSIGNED] [ZEROFILL]

Queste definizioni sono sinonimi di **DECIMAL**. Il tipo **FIXED** è stato aggiunto dalla versione 4.1.0 per compatibilità con altri server.

18.1.1 Intervallo dei valori ammessi per il tipo numerico intero

Tipo	byte	Valore Minimo	Valore Massimo
TINYINT	1	-128	127
SMALLINT	2	-32.768	32.767
MEDIUMINT	3	-8.388.608	8.388.607
INT	4	-2.147.483.648	2.147.483.647
BIGINT	8	-9.223.372.036.854.775.808	9.223.372.036.854.775.807

18.2 Valori di Tipo Date e Time

DATE

Valore Data. Il range (in anno,mese,giorno) va da **'1000-01-01'** a **'9999-12-31'**. MySQL mostra i valori **DATE** nel formato **'AAAA-MM-GG'**, ma permette di assegnare i valori alle colonne di tipo **DATE** usando sia stringhe che numeri.

DATETIME

Una combinazione di valori DATE (Data) e TIME (cioè orario). Il range va da **'1000-01-01 00:00:00'** a **'9999-12-31 23:59:59'**. MySQL mostra i valori **DATETIME** nel formato **'AAAA-MM-GG HH:MM:SS'**, ma permette di assegnare i valori alle colonne di tipo **DATETIME** usando sia stringhe che numeri.

TIMESTAMP[(M)]

Un valore di tipo timestamp. L'intervallo è **'1970-01-01 00:00:00'** fino ad un momento indefinito dell'anno **2037**. Una colonna **TIMESTAMP** torna utile per archiviare il momento di una operazione di **INSERT** o **UPDATE** sul record. La prima colonna **TIMESTAMP** di una Tabella è automaticamente riempita con la data e l'orario della modifica più recente eseguita sul record stesso, se non viene specificato un altro valore dall'utente. Potete inoltre riempire una qualsiasi colonna **TIMESTAMP** alla data e ora corrente semplicemente assegnando un valore **null**. L'argomento **M** indica solo come la colonna **TIMESTAMP** viene visualizzata, e non la lunghezza del valore archiviato, che è sempre 4 byte.

TIME

Un valore orario. L'intervallo va da **'-838:59:59'** to **'838:59:59'**. MySQL mostra i valori **TIME** nel formato **'HH:MM:SS'**, ma permette di assegnare i valori alle colonne usando sia stringhe che numeri.

YEAR[(2|4)]

Un valore che indica un Anno, nel formato a due o a quattro cifre. Il valore predefinito è quattro cifre. I valori permessi vanno dal **1901** al **2155**, nel formato a quattro cifre, e da **70** a **69**, che rappresentano gli anni dal 1970 al 2069, nel formato a due cifre. MySQL mostra le colonne di tipo **YEAR** nel formato **AAAA**, ma permette di assegnare i valori alle colonne usando sia stringhe che numeri.

18.3 Valori di Tipo Stringa

[NATIONAL] CHAR(M) [BINARY | ASCII | UNICODE]

Una stringa di lunghezza fissa, che viene comunque riempita a destra di spazi, dal motore di Db, fino a raggiungere la lunghezza massima. **M** rappresenta il numero massimo di caratteri che la colonna può contenere. Il range di **M** va da 0 a 255 caratteri. Gli spazi aggiunti vengono rimossi quando si accede al valore memorizzato. I valori **CHAR** vengono ordinati e comparati **NON** tenendo conto delle maiuscole e minuscole (**case-insensitive**), in accordo con il set di caratteri predefinito, a meno che non venga specificata la parola chiave **BINARY**. Dalla versione 4.1.0, una colonna di tipo **CHAR** con una lunghezza specificata maggiore di 255 caratteri viene convertita nel più piccolo valore di tipo **TEXT** che può contenere stringhe di quella lunghezza. **CHAR** è una abbreviazione di **CHARACTER**. Il tipo **NATIONAL CHAR** (o la sua equivalente forma breve, **NCHAR**) è il modo standard per SQL di definire che una colonna di tipo **CHAR** deve usare il set di caratteri di default, e questa è l'opzione predefinita per MySQL. Dalla versione 4.1.0, l'attributo **ASCII** può essere specificato. Questa opzione assegna il set di caratteri **latin1** ad una colonna di tipo **CHAR**. Dalla versione 4.1.1 può essere specificato anche l'attributo **UNICODE**. MySQL permette la creazione di una colonna di tipo **CHAR(0)**. Ciò è utile soprattutto per motivi di compatibilità con vecchie applicazioni che prevedono l'esistenza di una colonna, ma non accedono mai ai valori contenuti nella colonna stessa.

CHAR

Sinonimo di **CHAR(1)**.

[NATIONAL] VARCHAR(M) [BINARY]

Una stringa di lunghezza variabile. **M** rappresenta il numero massimo di caratteri contenuti. Il range di **M** è compreso tra 0 e 255 caratteri. I valori **VARCHAR** vengono ordinati e comparati **NON** tenendo conto delle maiuscole e minuscole (**case-**

insensitive), a meno che non venga specificata la parola chiave **BINARY**. Dalla versione 4.1.0, una colonna di tipo **VARCHAR** con una lunghezza specificata maggiore di 255 caratteri viene convertita nel più piccolo valore di tipo **TEXT** che può contenere stringhe di quella lunghezza. **VARCHAR** è una abbreviazione di **CHARACTER VARYING**.

TINYBLOB
TINYTEXT
Un valore di tipo **BLOB** o **TEXT** con una lunghezza massima di **255** (2^8 - 1) caratteri.

BLOB
TEXT
Un valore di tipo **BLOB** o **TEXT** con una lunghezza massima di **65.535** (2^16 -1) caratteri.

MEDIUMBLOB
MEDIUMTEXT
Un valore di tipo **BLOB** o **TEXT** con una lunghezza massima di **16.777.215** (2^24 - 1) caratteri.

LONGBLOB
LONGTEXT
Un valore di tipo **BLOB** o **TEXT** con una lunghezza massima di **4.294.967.295** o 4GB (2^32 - 1) caratteri.

ENUM('value1','value2',...)
Una lista di scelta. In sostanza un valore stringa che può avere un solo valore scelto tra duna lista predefinita (**'value1', 'value2', ...**), oppure **NULL**, oppure il carattere speciale **''** che rappresenta un valore errato. Una lista **ENUM** può comprendere un massimo di 65,535 valori distinti.

SET('value1','value2',...)
Un set di valori. Una stringa che può contenere zero o più valori, ognuno dei quali deve essere scelto dalla lista specificata (**'value1', 'value2', ...**). Un **SET** può avere al massimo 64 membri.