12. Uso avanzato di....

12.1 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...

12.1.1 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:

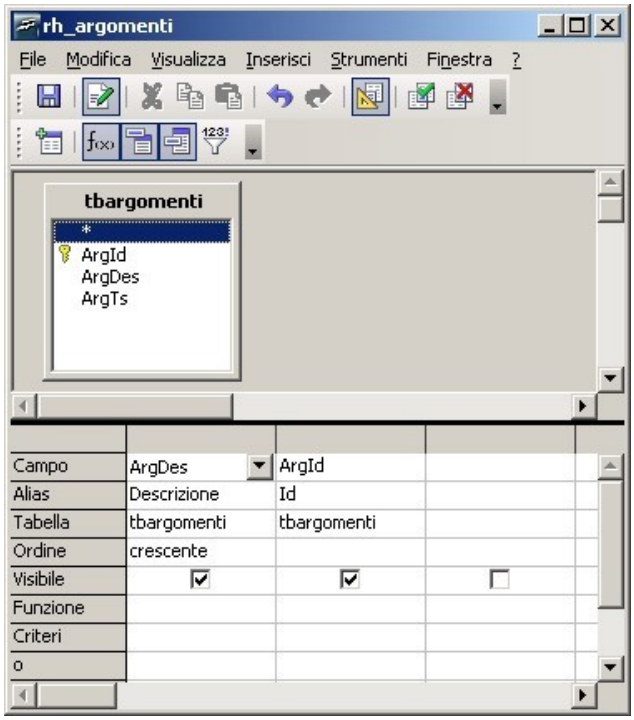


Figura 12.1.1: Ricerca su una Tabella singola

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

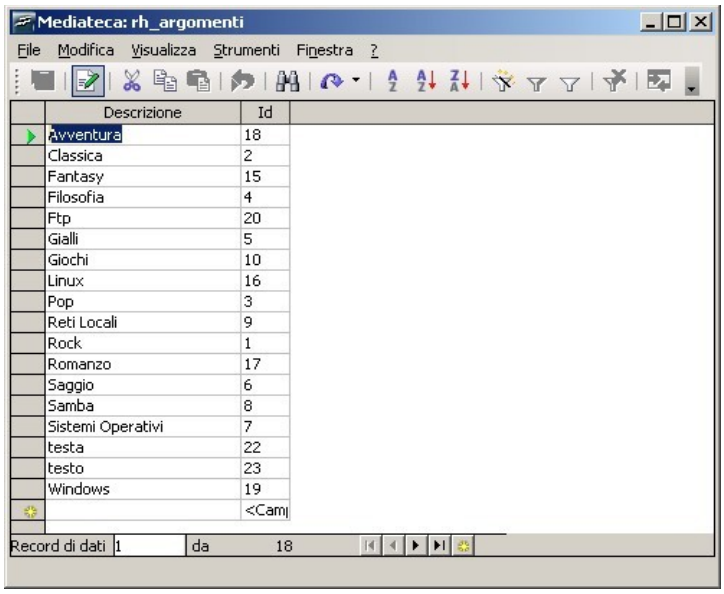


Figura 12.1.2 Il risultato della Query

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 il token ***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 :

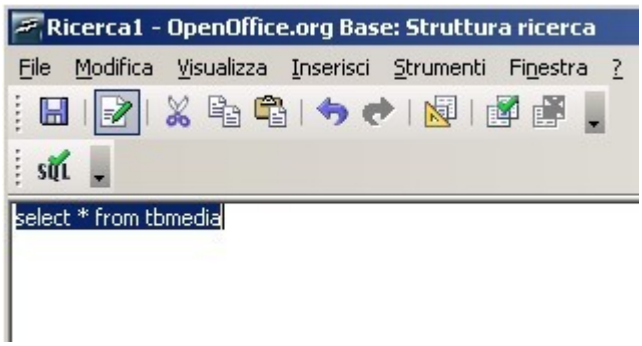


Figura 12.1.3: Una Ricerca in vista SQL

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	Springste	1	1		0,00	11/05/0

Record di dati 1 da 15 *

Figura 12.1.4: Risultato della Ricerca su TbMedia

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

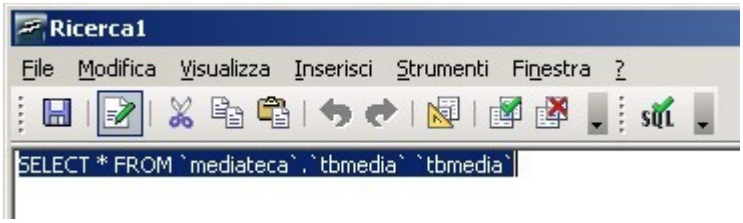


Figura 12.1.5: La Ricerca "rielaborata" da OOo

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.

12.1.2 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:

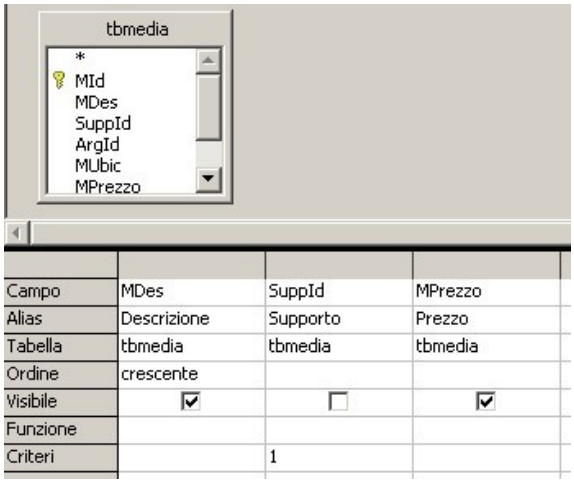


Figura 12.1.6: Ricerca con criteri

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 12.1.7: 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. In fondo 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 12.1.8: Ricerca con la richiesta del parametro

In questo modo, all'esecuzione avremo questa maschera :

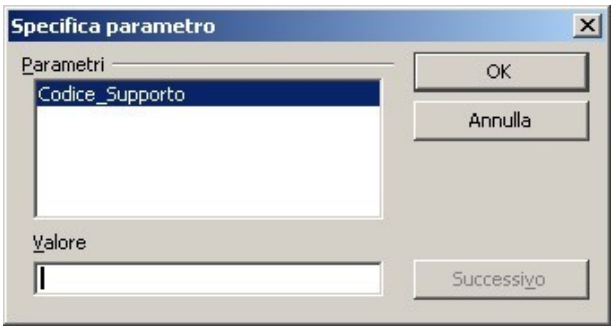


Figura 12.1.9: Richiesta del parametro

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....

12.1.3 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 query:

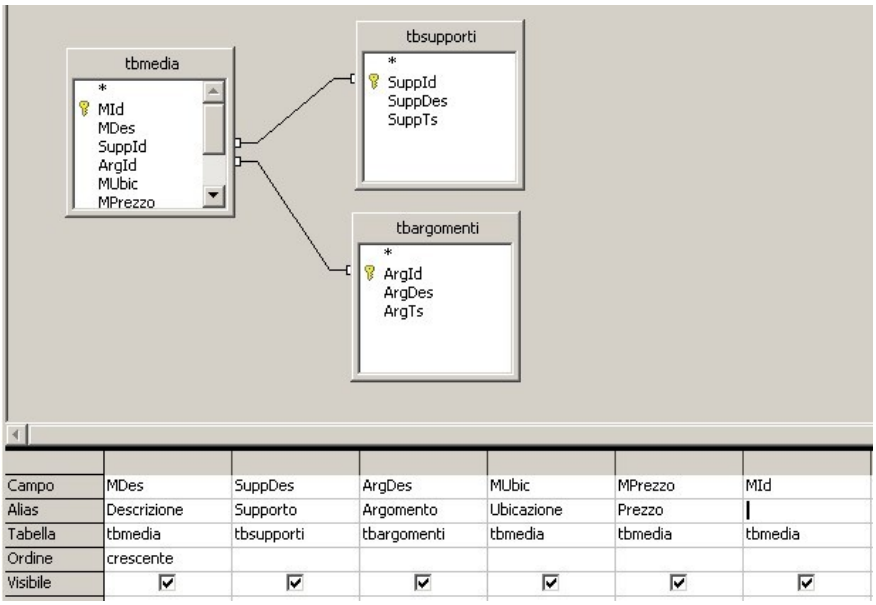


Figura 12.1.10 La ricerca per la prima Scheda del nostro Formulario

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 12.1.11: 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.

12.1.4 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 12.1.12: Ricerca con più Parametri

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

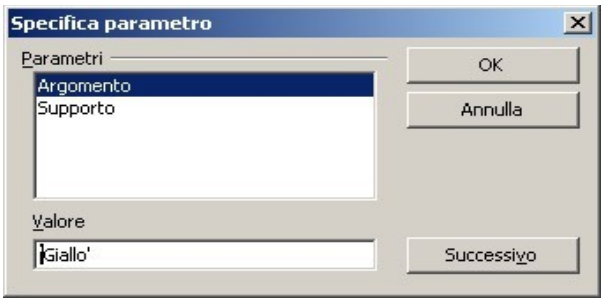


Figura 12.1.13: Richiesta di più parametri

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 12.1.14: 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.

12.1.5 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.

Tips

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:

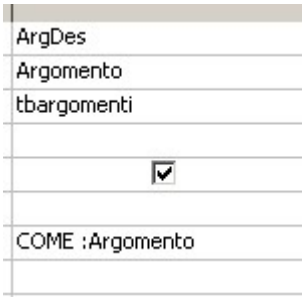


Figura 12.1.15:
Parametri con wildcards

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....

12.1.6 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 12.1.16: 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 12.1.17: Ricerche a sola lettura

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

12.1.7 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):

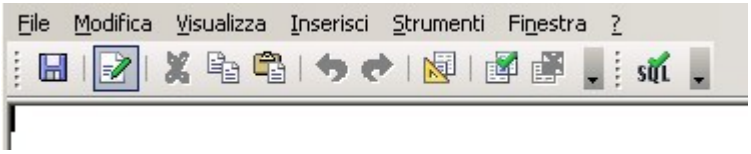


Figura 12.1.18: Barra per l'esecuzione di comandi SQL

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

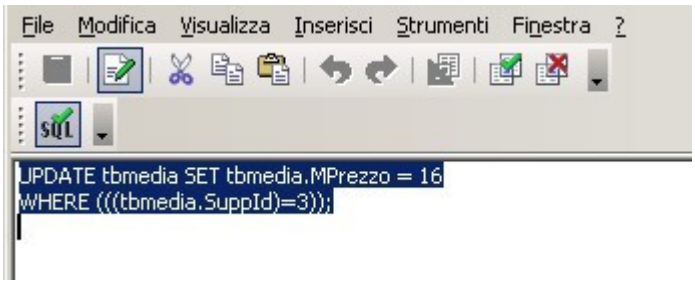


Figura 12.1.19: Esecuzione diretta di un Comando SQL

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 12.1.20: 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.

Tips

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....

12.1.8 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 di OOo Beta che stiamo utilizzando, però, questo tipo di elaborazione è praticamente impossibile, perché la sintassi dell'istruzione SQL non viene generata in modo corretto.* Useremo quindi la 1.1.4, perché in sostanza non cambia praticamente nulla. Supponiamo di desiderare una Ricerca che conti quanti Media abbiamo, suddividendoli per Argomento. Potremmo costruire qualcosa del genere:

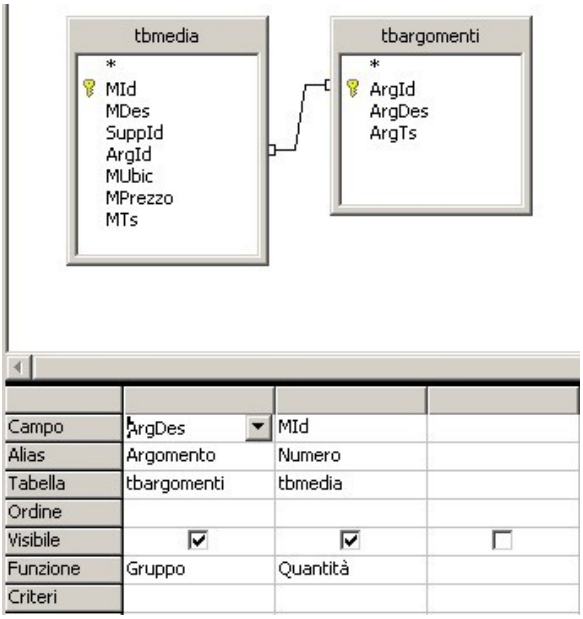


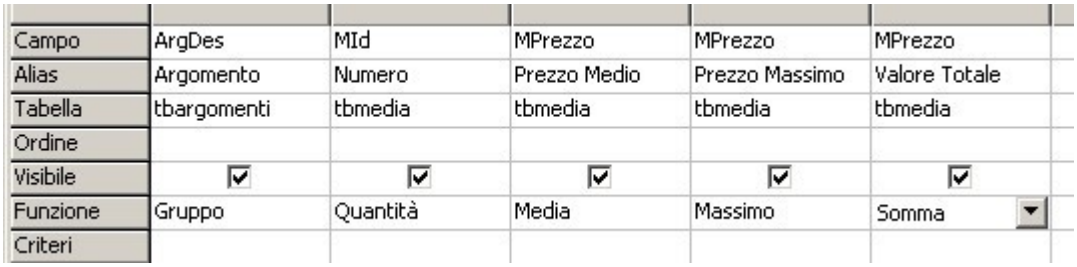
Figura 12.1.21: Ricerca con uso di Gruppi

Per ottenere:

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

Figura 12.1.22: 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 12.1.23: 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 12.1.24: 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.

12.1.9 Funzioni nei campi

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

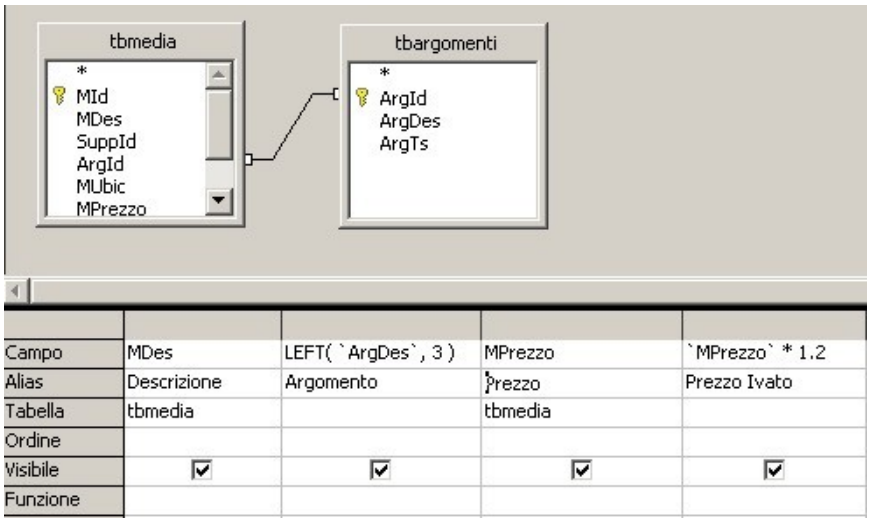


Figura 12.1.25: Ricerca con funzioni nei campi

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 12.1.26: 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`)
```

12.1.10 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:

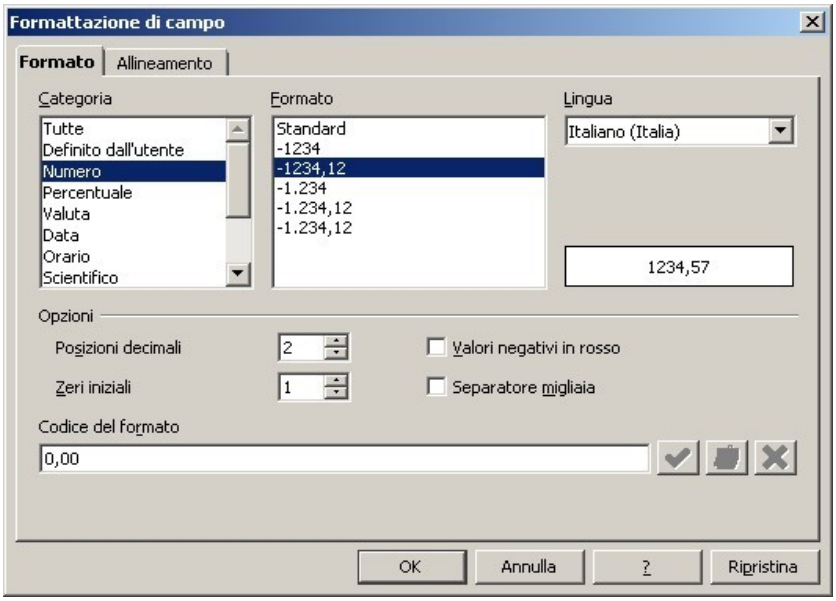


Figura 12.1.27: Formattazione di una Colonna

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.

12.2 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	*

Campo	Tipo di Campo	Commenti	Idx
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 12.2.1: 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 12.2.2: 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 12.2.3: 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 bel 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:

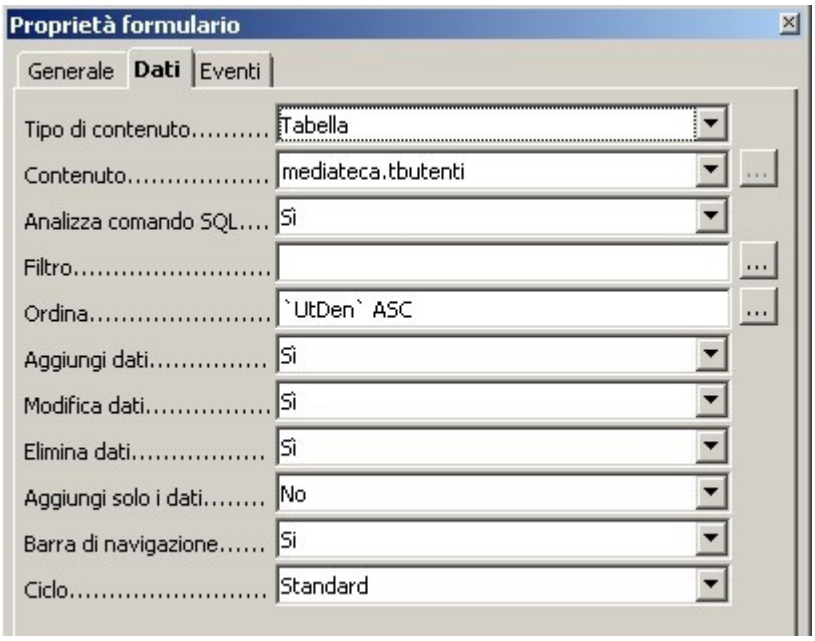


Figura 12.2.4: Proprietà del Formulario

Tips

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 12.2.5: Campo di Testo

Ma queste sono cose che abbiamo già visto, quindi procediamo senza indugio....

12.2.1 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:

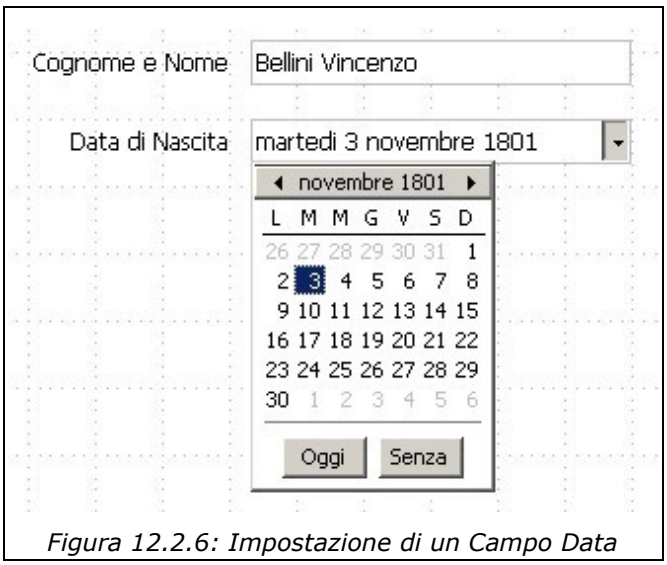


Figura 12.2.6: Impostazione di un Campo Data

12.2.2 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 12.2.7: Tab Dati di una Casella di Controllo

Tips

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".

12.2.3 Campi a Maschera

Nei Formulari è a volte indispensabile stabilire delle "Maschere di Immissione" per alcuni tipi di dati. Queste "Maschere" stabiliscono 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 12.2.8: 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.

12.2.4 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'autocomposizione, 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 capi 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 12.2.9: Pulsante di scelta per il campo "Maschio"

Per il campo "*Femmina*" basta cambiare il "*Valore di riferimento (on)*" in "*F*".

Figura 12.2.10

12.2.5 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 12.2.11: 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.

12.2.6 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.

12.3 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'autocomposizione, e gli interventi successivi possibili sono troppo limitati.

Torneremo a parlare dei Rapporti probabilmente quando sarà disponibile la versione 2.0 definitiva.