Il linguaggio SQL e Access

1 Linguaggio SQL

Caratteristiche generali

L'SQL (structured query language) è un linguaggio utilizzato per *definire*, *gestire*, *controllare* e *reperire* dati di un DBMS; prevede funzioni di DDL (Data Description Language), DML (Data Manipulating Language) e QL (Query Language). Esso può anche essere usato come linguaggio ospite (embedded SQL) di linguaggi procedurali.

Nasce nel 1981 per opera dell'IBM con il nome di SQL/DS (Data System) come prodotto per database; le sue origini risalgono però al 1973 quando l'IBM creò il linguaggio SEQUEL. Dal 1981 sono nate numerose versioni fino a quella del 1986 prodotta dall'ISA. In pratica tutti i RDBMS (dBASE IV, Oracle, Paradox, Informix, Access) consentono di utilizzare istruzioni in SQL.

Il linguaggio SQL presenta due diverse modalità di utilizzo dei comandi:

- Il modo interattivo
- Il modo programmatico (embedded SQL)

Il modo *interattivo* è un ambiente in cui è possibile introdurre comandi SQL attraverso la tastiera. I comandi agiscono su tabelle e l'effetto è immediato. In genere i DBMS mettono a disposizione delle librerie che consentono di applicare le istruzioni SQL a fonti di dati esterne. Microsoft utilizza ODBC

Il modo *Embedded* consente di inglobare il codice SQL nel codice di un linguaggio ospite e vengono utilizzati i normali meccanismi del linguaggio per il passaggio dei parametri e l'utilizzo dei risultati. Normalmente il codice così prodotto viene prima convertito da un pre-processore e in seguito compilato dal compilatore del linguaggio ospite.

Le implementazioni di SQL variano l'una dall'altra; alcune escludono delle istruzioni o parti di istruzione, altre ne aggiungono. Molti prodotti riconoscono solo l'istruzione SELECT.

- Identificatori e tipi di dato

Gli identificatori: sequenza di caratteri (max 18). I tipi sono standard. Leggere a pag. 134

I comandi

- Comandi per la creazione del database

Per la creazione di un database è a disposizione il comando CREATE DATABASE

```
CREATE DATABASE {percorso} nome data base
```

- Comandi per la creazione di tabelle

Per la **creazione di una tabella** è a disposizione il comando CREATE TABLE:

```
CREATE TABLE nometab
(nome colonna1   tipo   [NOT NULL],
 nome colonna2   tipo   [NOT NULL],
……)
```

▶ Esempio:

```
CREATE TABLE Studenti
   (nome CHAR(15) NOT NULL,
   cognome CHAR(15) NOT NULL,
   classe SMALLINT NOT NULL,
   sezione CHAR(1) NOT NULL,
   anno-nascita SMALLINT)
CREATE TABLE Docenti
   (nome CHAR(15) NOT NULL,
   cognome CHAR(15) NOT NULL,
   materia CHAR(20) NOT NULL)
```

I tipi disponibili sono del tipo elementare: Char (n), Integer, Float(Reali), SmallInt e altri ancora. Alcune implementazioni consentono tipi particolari per la data, per la valuta, per il n° di telefono.

La clausola **NOT NULL** indica che il valore da specificare è obbligatorio.

- Comandi per la manipolazione di tabelle

È possibile manipolare tabelle per aggiungere/eliminare (ADD/DROP) una colonna

- Per aggiungere l'attributo *Nascita* nella tabella docenti

```
ALTER TABLE Docenti
   ADD Nascita date;
```

L'istruzione CREATE INDEX viene utilizzata per creare un nuovo indice su una tabella (utile per velocizzare gli accessi).

```
CREATE UNIQUE INDEX IDoc
   ON Docenti(cognome, nome);
```

Per eliminare una tabella DROP TABLE nome tabella

```
DROP TABLE Docenti
```

- Comandi per la manipolazione dei dati

Mediante i comandi INSERT, UPDATE e DELETE è possibile manipolare i dati in una tabella.

Per l'**inserimento dati** è a disposizione il comando:

```
INSERT INTO  nometabella
   (valori)
```

- Esempio:

```
INSERT INTO Studenti
   ("Mario", "Rossi", 3, "A", 1974
   "Franco", "Bianchi", 4, "A", 1973
   "Sara", "Verdi", 5, "B", 1972)
```

Per **aggiornare** un valore è possibile utilizzare:

```
UPDATE nome tabella
   SET nome campo = valore
   WHERE (condizione per individuare la tupla)
```

- Vedi esempi a pag. 138

Per eliminare alcune tuple si utilizza DELETE FROM

```
DELETE FROM  nometabella
   WHERE (condizione per individuare le tuple da eliminare)
```

Comando SELECT

Consente di porre interrogazioni alla base di dati. Racchiude in se la "*proiezione*" e la "*selezione*".

```
SELECT [DISTICT|ALL] [*|colonna1, colonna2, ....][AS nomecolonna]
FROM tabella1, tabella2, ....
[WHERE condizione]
[GROUP BY colonna1, colonna2, ... [HAVING condizione]]
[ORDER BY colonna1, colonna2, ... [DESC]]
```

Seleziona dalla/e tabella/e specificata/e le righe secondo particolari clausole, permette inoltre di operare una scelta anche delle colonne. Se non è specificato nulla il risultato sarà la visualizzazione di tutta la tabella. Le parole SELECT e FROM sono le uniche sempre presenti.

- *Interrogazione semplice*: conoscere cognome, nome e classe dalla tabella studenti

```
SELECT cognome, nome, classe
FROM studenti;
```

Il comando opererà la selezione di tutte le tuple della tabella *studenti* visualizzando solo gli attributi specificati.

- *Interrogazione completa*: visualizzare tutti gli attributi della tabella studenti

```
SELECT *
FROM studenti;
```

- Predicato ALL/DISTINCT

L'opzione **ALL** (default) conserva un numero di righe pari al numero di righe ottenute dall'interrogazione, le righe duplicate non sono soppresse. **DISTINCT** causa una soppressione delle righe duplicate.

- *Interrogazione con duplicazione* (defualt): conoscere classe e sezione di tutti gli studenti

```
SELECT ALL classe, sezione
FROM studenti;
```

- *Interrogazione senza duplicazione*: conoscere solo le classi e le sezioni degli studenti

```
SELECT DISTINCT classe, sezione
FROM studenti;
```

Nel primo caso il numero di tuple corrisponde al numero originale di elementi nella tabella, nel secondo compariranno le tuple ripetute solo una volta.

- *Interrogazione senza duplicazione*: visualizzare le funzioni presenti nella tabella impiegato

```
SELECT DISTINCT funzione
FROM impiegato;
```

- Predicato **AS**

L'opzione **AS** consente di rinominare una colonna (alias) o di attribuire un nome a una colonna derivante da un calcolo (in seguito)

- *Interrogazione con nuovo nome alla colonna*: per ottenere dall'attributo *prov* della tabella studenti l'elenco delle province di provenienza degli studenti, intestando la colonna con "Provincia"

```
SELECT DISTINCT prov AS Provincia
FROM studenti;
```

- *Interrogazione con nuova colonna calcolata*: visualizzare accanto agli stipendi attuali, quali sarebbero i nuovi stipendi con un aumento del 5% senza modificare il valore attuale

```
SELECT cognome, nome, stipbase AS Attuale, stipbase*1,05 AS Nuovo
FROM impiegati;
```

Clausola WHERE

La clausola **WHERE** consente di introdurre condizioni che permettono di selezionare le tuple.

- *Interrogazione con selezione di righe*: per conoscere tutti gli insegnati che insegnano "*italiano*" dalla tabella insegnanti:

```
SELECT cognome, nome
FROM docenti
WHERE materia = "italiano";
```

È possibile anche un inserimento parametrico

- *Interrogazione parametrica*: per conoscere tutti gli insegnati che insegnano una materia inserita datastiera:

```
SELECT cognome, nome
FROM docenti
WHERE materia = [Materia insegnata];
```

- Condizioni di ricerca

Con la clausola WHERE è possibile usare **operatori logici** (AND, OR, NOT) e **di relazione** (=, <,>, <>, >=, <=)

- *Interrogazione con operatore AND*: per conoscere tutti gli *impiegati* della provincia *BS* dalla tabella dipendente:

```
SELECT cognome, nome
FROM dipendente
WHERE funzione =”impiegato” AND prov=”BS”;
```

L'operatore **BETWEEN/AND** serve per specificare un intervallo

- *Interrogazione con operatore BETWEEN/AND*: per conoscere gli impiegati con stipendio base compreso tra 1000 e 1500 euro dalla tabella dipendente:

```
SELECT cognome, nome
FROM dipendente
WHERE funzione =”impiegato” AND stipbase BETWEEN 1000 AND 1500;
```

L'operatore **LIKE** confronta un attributo con un valore. È utilizzabile al posto dell'operatore **=** inoltre consente l'uso dei caratteri jolly: **_** (underscore) per indicare un singolo carattere in quella posizione nella stringa, **%** (percento) per indicare una qualsiasi sequenza di caratteri in quella posizione.

- *Interrogazione con operatore LIKE*: per conoscere tutti i dipendenti il cui cognome inizia con 'M':

```
SELECT cognome, nome
FROM dipendente
WHERE cognome LIKE ‘M%’;
```

L'operatore **IN** controlla l'appartenenza di un attributo a un insieme di valori.

- *Interrogazione con operatore IN*: per conoscere tutti i dipendenti delle province di 'BS', 'CR', 'MI'

```
SELECT cognome, nome
FROM dipendente
WHERE prov IN (“BS”,”CR”,”MI”);
```

L'operatore IS NULL o IS NOT NULL possono essere usati per selezionare righe con attributi nulli o non nulli:

- *Interrogazione con operatore IS NOT NULL*: per conoscere solo i dipendenti con indirizzo e-mail:

```
SELECT cognome, nome, email
FROM dipendente
WHERE email IS NOT NULL;
```

Funzioni di aggregazione

Funzioni che agiscono sui valori contenuti nelle righe: COUNT, SUM, AVG, MIN, MAX

- Funzione **COUNT**

Conta il numero di righe presenti in una tabella.

La sintassi prevede di specificare come argomento della funzione il **nome di un attributo** oppure il carattere '*****'.

Nel primo caso non vengono conteggiate le righe che hanno valore **Null** nella colonna dell'attributo specificato, nel secondo caso si calcola il numero delle righe incluse quelle con campi nulli.

```
SELECT COUNT(attributo)
FROM nometabella;

conta numero tuple da nometabella
con attributo indicato non nullo
```

```
SELECT COUNT(*)
FROM nometabella;

conta numero tuple da nometabella
```

- Conoscere il numero di dipendenti con indirizzo email:

```
SELECT COUNT(email)
FROM dipendente;
```

- Conteggio con clausola

Con la clausola WHERE è possibile specificare una condizione.

- *Conteggio con clausola*: per contare gli studenti di 'BS':

```
SELECT COUNT(*)
FROM studente
WHERE prov=”BS”;
```

- Funzione SUM, AVG, MIN e MAX

SUM restituisce la somma di tutti i valori contenuti in una colonna specificata come argomento.

- *Funzione SUM*: per calcolare la somma delle rette degli studenti di una classe

```
SELECT SUM(retta)
FROM studente
WHERE classe=3;
```

AVG restituisce la media di tutti i valori contenuti in una colonna specificata come argomento.

- *Funzione AVG*: per calcolare la media delle rette degli studenti di una classe

```
SELECT AVG(retta)
FROM studente
WHERE classe=3;
```

MIN e MAX restituiscono il valore minimo e massimo di un attributo.

- *Funzione MIN e MAX*: per calcolare la retta più bassa e più alta degli studenti di una classe

```
SELECT MIN(retta), MAX(retta)
FROM studente
WHERE classe=3;
```

Ordinamenti e Raggruppamenti

- Odinamento delle righe

La clausola ORDER BY consente di ottenere i risultati di una interrogazione ordinati secondo valori contenuti in una o più colonne.

- *Clausola ORDER BY*: per ordinare i dipendenti per stipendio base decrescente e per cognome

```
SELECT cognome, stipbase
FROM dipendente
ORDER BY stipbase DESC, cognome;
```

ATTENZIONE: in Access bisogna stare attenti all'ordine delle colonne.

- Raggruppamenti

La clausola GROUP BY serve per raggruppare un insieme di righe aventi lo stesso valore nelle colonne indicate: questa opzione produce un riga di risultato per ogni raggruppamento.

- *Clausola GROUP BY*: per ottenere la lista delle funzioni dei dipendenti con la somma degli stipendi e il numero di dipendenti appartenenti alla funzione:

```
SELECT funzione, SUM(stipbase), COUNT(*)
FROM dipendente
GROUP BY funzione;
```

In genere GROUP BY è associata a funzioni di conteggio o di somma.

ATTENZIONE: tutti gli attributi che compaiono nella lista accanto a SELECT devono essere inclusi nella clausola GROUP BY oppure devono essere argomenti di una funzione di aggregazione.

- *Clausola GROUP BY*: per conoscere classe per classe, sezione per sezione, il numero di studenti e la retta media pagata:

```
SELECT classe, sez, Avg(retta) AS Mediaretta, Count(cognome) AS Studenti
FROM studente
GROUP BY classe, sez;
```

- *Clausola GROUP BY*: per conteggiare il numero di studenti della provincia di "BS" classificati per retta pagata:

```
SELECT retta, Count(cognome) AS Studenti
FROM studente
WHERE prov="BS"
GROUP BY retta;
```

- Clausola HAVING

Introduce una condizione sul raggruppamento

- *Clausola HAVING*: per conoscere le classi con più di 10 studenti:

```
SELECT classe, sez, Count(cognome) AS Studenti
FROM studente
GROUP BY retta;
```

```
HAVING Count(cognome)>=10;
```

- *Clausola HAVING*: per ottenere l'elenco delle filiali nelle quali ci sono più di 10 dipendenti con la funzione 'Impiegato':

```
SELECT filiale, COUNT(filiale) AS conteggio
FROM personale
WHERE funzione="Impiegato"
GROUP BY filiale
HAVING COUNT(*)>10;
```

Interrogazioni su più tabelle

Il comando SELECT può operare su più tabelle, in tal caso i nomi degli attributi sono preceduti dal nome della tabella e nella clausola WHERE si deve specificare i nomi degli attributi che corrispondono nelle due tabelle (**congiunzione**).

```
SELECT tabella1.attributo1, tabella1,attributo2,...,tabella2.attributo1,...
FROM tabella1,tabella2
WHERE tabella1.attributoi = tabella1.attributoj
```

- Conoscere docente per docente la materia che insegna, *IDdoc* è attributo di congiunzione

```
SELECT docente.cognome, docente.nome, materia.nome
FROM docente, materia
WHERE docente.IDdoc = materia.IDdoc;
```

Viene stabilito un **equi-join**.

Con Access la clausola WHERE deve essere specificata nel seguente modo:

```
WHERE docente INNER JOIN materia ON docente.Iddoc = materia.IDdoc;
```

- Tipi di Join

Esistono tre tipi di Join.

In Access è possibile scegliere i tipi di JOIN modificando la relazione direttamente nella finestra struttura delle query.

- Equi Join (**INNER JOIN**) : include nell'interrogazione solo le righe i cui campi collegati sono eguali.

  Nell'esempio precedente vengono visualizzati solo i *docenti* che sono stati abbinati ad una *materia*.

- Left Join (**LEFT JOIN**) : include nell'interrogazione tutti i record della prima tabella e solo i record della seconda i cui campi collegati sono eguali.

  ```
  WHERE docente LEFT JOIN materia ON docente.Iddoc = materia.IDdoc;
  ```

  In questo caso vengono visualizzati anche i *docenti* che non sono stati abbinati ad una *materia*.

- Right Join (**RIGHT JOIN**) : include nell'interrogazione tutti i record della seconda tabella e solo i record della prima i cui campi collegati sono eguali.
  ```
  WHERE docente RIGHT JOIN materia ON docente.Iddoc = materia.IDdoc;
  ```

  In questo caso vengono visualizzati soli i *docenti* con una *materia* abbinata e vengono inclusi anche i record della tabella *materia* non abbinate ad alcun docente.

- Self Join

  Nell'esempio seguente viene creato un SELF JOIN, necessario quando in una interrogazione si fa riferimento ad attributi della medesima tabella.

Supponiamo che nella tabella *personale* ci siano alcuni dipendenti con funzione dirigente che dirigono gli altri dipendenti. Ciò è possibile perché nella tabella è stato aggiunto un campo "*dirigente*" che contiene il codice del dirigente. Si vuole conoscere per ogni dipendente qual è il suo dirigente.

```
SELECT tab1.cognome, tab1.nome, tab2.cognome
FROM personale AS tab1, personale AS tab2, materia
WHERE tab1.dirigente = tab2.IDper;
```

- Esercizi:

2 *Access*

Breve ripasso su Access e funzioni

Le Query

- La finestra delle query

Impostare una query in visualizzazione struttura.

Trascinare i singoli campi o tutti '*'.

Eseguire una q.

Introdurre semplici criteri.

- Tipi di query

Query "di selezione", "con parametri", "a campi incrociati", "di comando"

- Query di selezione

È il tipo più comune e consente di recuperare e visualizzare dati provenienti da più tabelle. Si possono fare raggruppamenti e calcoli.

- Query con parametri

È una query di selezione nella quale viene visualizzata una finestra di dialogo che richiede delle informazioni (inserimento di una data o di un certo valore)

Nel database *classifica.mdb* creare una query con parametri che consenta di visualizzare i dischi di un certo distributore

- Query a campi incrociati

Consente di estrarre valori di riepilogo come somme, conteggi e medieda un campo di una tabella e li visualizza in una tabella a doppia entrata, ponendo sulla riga di intestazione la prima chiave di raggruppamento e sulla colonna a sinistra la seconda chiave di raggruppamento.

Ad esempio si vuole suddividere gli artisti per prezzo del CD prodotto e per casa distributrice

```
Creare nuova query
Selezionare nel menù "query" la voce "query a campi incrociati"
Trascinare il campo "PrezzoCD" e selezionare nella casella campi incrociati
"Intestazione riga"
Trascinare il campo "Distri" e selezionare nella casella campi incrociati
"Intestazione colonna"
Trascinare il campo "Artista" e selezionare nella casella formula l'operazione
"conteggio" e nella casella campi incrociati "Valore"
Eseguire la query
Provare a invertire le intestazioni
```

- Query di comando

Apporta modifiche a gruppi di record con una sola operazione (incrementare la retta agli studenti, cancellare gruppi di studenti ecc.).

I comandi sono 4:

eliminazione: per eliminare un gruppo di record

aggiornamento: per apportare modifiche globali a gruppi di record

creazione di tabelle: per ad esempio esportare dati

accodamento: per aggiungere record

- Query di eliminazione

Eliminare dalla tabella classifica, o da una sua copia, tutte le righe il cui distributore è "emi"

```
Creare nuova query
Selezionare nel menù "query" la voce "query di eliminazione"
Doppio click su "*" per trascinare tutti i campi
Doppio click sul campo "Distri"
Nella casella criteri scrivere "emi"
```

```
Eseguire la query leggendo i messaggi riportati (ATTENZIONE)
```

- Queri di aggiornamento

È utile per aggiornare alcuni campi della tabella in base a criteri prestabiliti

Esempio: nella tabella *classifica* si vuole aumentare di % euro il prezzo dei CD ditribuiti dalla "Sony"

```
Creare nuova query
Selezionare nel menù "query" la voce "query di aggiornamento"
Doppio click su "PrezzoCD" e nel campo "Aggiorna a" scrivere
"classifica.PrezzoCD+5"
Doppio click sul campo "Distri"
Nella casella criteri scrivere "sony"
Eseguire la query leggendo i messaggi riportati (ATTENZIONE)
```

- Query di creazione tabella

La query serve per creare una nuova tabella con i dati provenienti da un'altra tabella. Nella nuova tabella si possono riportare solo alcuni campi, anche le righe possono soddisfare opportuni criteri. La nuova tabella può essere creata nel medesimo database oppure in un altro. L'operazione è utile ad esempio per esportare dati.

Esempio: a partire dai dati contenuti nella tabella *"classifica"* si vuole creare la tabella denominata *"Nuova"* contenente solo i dischi distribuiti da "Sony"

```
Creare nuova query
Selezionare nel menù "query" la voce "query di creazione tabella" e nella
finestra di dialogo "Creazione tabella" specificare "Nuova" come nome della
tabella, lasciando attivo "Database corrente".
Selezionare la tabella "classifica" e trascinare tutti i campi
Nella casella criteri del campo "Distri" scrivere "sony"
Eseguire la query leggendo i messaggi riportati (ATTENZIONE)
```

- Query di accoramento

Serve per aggiungere nuove righe in una tabella con dati provenienti da un'altra tabella:

- se la tabella di provenienza ha grado > allora bisogna selezionare le colonne da introdurre
- se il grado è < le colonne mancanti vengono riempite con il valore Null (0)

Esempio si vuole aggiungere alla tabella "Nuova" tutti i dischi distribuiti da "emi"

```
Creare nuova query
Selezionare nel menù "query" la voce "query di accodamento" e nella finestra di
dialogo "Accodamento" specificare "Nuova" come nome della tabella, lasciando
attivo "Database corrente".
Selezionare la tabella "classifica" e trascinare tutti i campi
Nella casella criteri del campo "Distri" scrivere "emi"
Eseguire la query leggendo i messaggi riportati (ATTENZIONE)
```

ATTENZIONE al campo contatore. Non includerlo se già presente nella tabella di destinazione

- Assegnare esercizi da svolgere

Query di raggruppamento

Servono per ottenere altre informazioni in genere legate a operazioni di conteggio. Per i raggruppamenti gli ordinamenti si utilizza la clausola GROUP BY.

Esempio: contare il numero di artisti per distributore

```
Creare nuova query con fonte dati "classifica"
Trascinare i campi "Distri" e "Artista"
Ciccare sul simbolo Σ e nel campo "formula:" selezionare "raggruppamento" per
il campo "Distri" e "conteggio" per il campo "Artista"
```

Esempio: mostrare il numero di artisti per settimane di presenza nella classifica

```
Creare nuova query con fonte dati "classifica"
Trascinare i campi "NumSett" e "Artista"
```

Ciccare sul simbolo Σ e nel campo “formula:” selezionare “raggruppamento” per il campo “NumSett” e “conteggio” per il campo “Artista”

Esempio: mostrare la media delle settimane di permanenza in classifica calcolata per distributore

Creare nuova query con fonte dati “classifica”
Trascinare i campi “Distri” e “NumSett”
Ciccare sul simbolo Σ e nel campo “formula:” selezionare “raggruppamento” per il campo “Distri” e “media” per il campo “NumSett”

È possibile specificare anche una condizione (clausola having)

- Le interrogazioni in SQL

Di seguito si riporta il codice delle interrogazioni degli esempi precedenti.

```
SELECT classifica.PosAtt, classifica.Titolo, classifica.Artista
FROM classifica;

SELECT PosAtt, PosPrec, Titolo, Artista, Distri
FROM classifica
WHERE (((classifica.Distri)=[Distributore?]));

TRANSFORM Count(classifica.Artista) AS ConteggioDiArtista
SELECT classifica.PrezzoCD
FROM classifica
GROUP BY classifica.PrezzoCD
PIVOT classifica.Distri;


DELETE copia.*, copia.Distri
FROM copia
WHERE (((copia.Distri)="emi"));


UPDATE copia SET copia.PrezzoCD = copia.PrezzoCD+5
WHERE (((copia.Distri)="sony"));


SELECT PosAtt,PosPrec,NumSett,Titolo,Etichetta,Artista,Distri,PrezzoCD INTO
Nuova
FROM classifica
WHERE (((classifica.Distri)="sony"));


INSERT INTO Nuova (PosAtt,PosPrec,NumSett,Titolo,Etichetta,Artista, Distri,
PrezzoCD)
SELECT PosAtt, PosPrec, NumSett, Titolo, Etichetta, Artista, Distri, PrezzoCD
FROM classifica
WHERE (((classifica.Distri)="emi"));


SELECT Distri, Count(Artista) AS ConteggioDiArtista
FROM classifica
GROUP BY classifica.Distri;


SELECT NumSett, Count(Artista) AS ConteggioDiArtista
FROM classifica
GROUP BY classifica.NumSett;


SELECT Distri, Avg(NumSett) AS MediaDiNumSett
FROM classifica
GROUP BY Distri;
```

Esercizio

L'esercizio proposto ha lo scopo di familiarizzare con query, maschere e immagini. Si tratta di gestire un catalogo di case suddivise per zona. File **`archivio abitazioni.mbd`**.

- Aggiungere un'immagine o un oggetto che viene modificato da un record all'altro

È possibile aggiungere due tipi di immagine o di oggetto: **un' immagine o un oggetto non associato che rimane invariato per tutti i record oppure un'immagine o un oggetto associato, diverso per i vari record**. Un'immagine o un oggetto può inoltre essere incorporato o collegato.

- Per memorizzare immagini in un database di Microsoft Access, creare un **campo Oggetto OLE** nella struttura della tabella.

Non è possibile visualizzare il contenuto di un campo Oggetto OLE o di una colonna di tipo image in una pagina di accesso ai dati.

- Creare la maschera e aprirla in visualizzazione Struttura.
- Creare una cornice di oggetto associato  e in **`dati>origine`** controllo specificare il campo OLE per l'immagine.
- Passare alla **visualizzazione maschera**

Se l'applicazione dalla quale si sta copiando supporta il trascinamento della selezione OLE, invece di utilizzare il comando **Oggetto**, è possibile trascinare il file direttamente da Risorse del computer o dal desktop nel campo Oggetto OLE in un database di Microsoft Access. In caso contrario, effettuare le seguenti operazioni:

- Spostarsi sul record in cui si desidera inserire l'oggetto e fare clic sul campo Oggetto OLE di un database di Microsoft Access (mdb) oppure nella colonna di tipo image di un progetto di Microsoft Access (adp).
- Scegliere **Oggetto** dal menu **Inserisci**.
- Nella finestra di dialogo **Inserisci oggetto** selezionare **Crea da file**, quindi specificare il percorso del file. Se non si conosce il percorso scegliere **Sfoglia**.
- Selezionare la casella di controllo **Mostra come icona** se si desidera che l'oggetto venga visualizzato come icona anziché come oggetto. La visualizzazione come icona può risultare utile quando un oggetto contiene informazioni supplementari che non è necessario visualizzare. La visualizzazione di un oggetto come icona richiede inoltre una quantità di spazio su disco notevolmente inferiore.
- Scegliere **OK**.

- Query per conoscere le case disponibili in una certa zona

Si vuole realizzare una maschera che permette di visionare le case di una determinata zona

- Creare una query di selezione sulla tabella **`case`** e successivamente creare con l'autocomposizione una maschera sulla query appena creata attribuendo un nome diverso dalla query (assegnato per defualt) ad esempio **`elenco case`**.
- Aprire la maschera in visualizzazione struttura, aggiungere una Casella combinata che mostri l'elenco delle zone e memorizzare il codice nel campo **`idz`** nella maschera. Attribuire alla Casella combinata un nome per esempio **`inzona`**.
- Aprire la query creata in visualizzazione struttura e nei criteri di **`idz`** inserire la frase **`[forms]![elenco case]![inzona]`**
- Creare una macro con l'azione **`rieseguiquery`** abbinare alla Casella Combinata **`inzona`** l'azione la macro **`rieseguiquery`** sull'evento **`Dopo Aggiornamento`**.
- Provare la maschera

È possibile introdurre altri criteri come costo inferiore a o superficie superiore a …Mostrare l'esempio.