Database
Il linguaggio SQL

RDBMS

- Sebbene di solito i dati di un DB siano memorizzati in un file o in un insieme di file, un'applicazione non interagisce direttamente con essi
- Esiste un'entità chiamata RDBMS (Relational Data Base Management System) che gestisce la base di dati e con cui l'applicazione colloquia in modo astratto
- Un RDBMS può essere una semplice libreria oppure un applicazione molto complessa
- Il DB può risiedere localmente o su un'altra macchina in rete
- L'applicazione invia dei comandi, sotto forma di stringhe di testo
- Questi comandi costituiscono l'unico modo per interagire con il database
- E' un modello di interazione di tipo cliente-servitore

SQL

- Una caratteristica importante del modello relazionale è la presenza di un linguaggio di interrogazione e manipolazione dei DB
- Come abbiamo detto questo linguaggio rappresenta l'unico modo per interagire con un RDBMS
- Il linguaggio di gran lunga più diffuso è SQL (Structured Query Language)
- E' uno standard – definito dall'ANSI - di cui esistono diverse versioni SQL 89, SQL 92, SQL 99
- In pratica i vari RDBMS (Oracle, MS SQL Server, SYBASE, DB2, PostrgreSQL, MySQL ecc.) utilizzano propri dialetti più o meno aderenti agli standard
- Attenzione: è un linguaggio case-insensitive: non fa differenza fra maiuscole e minuscole

Cosa si può fare con SQL

- SQL è sostanziamente un insieme di comandi (statement), con una sintassi ben definita
- E' costituito da due parti:
  - DDL (Data Definition Language):
    - permette di creare basi di dati e modificarne la struttura: tabelle, vincoli di integrità, diritti di accesso ecc.
  - DML (Data Manipulation Language): permette di fare due tipi di operazioni
    - Modificare i dati in un DB: inserire, cancellare e aggiornare tuple
    - Interrogare il DB ed estrarre le informazioni volute
- Il termine query viene utilizzato spesso come sinonimo di comando. Si parla quindi di Query SQL

Esploriamo SQL

- **Per esaminare i vari comandi SQL utilizziamo l'esempio delle carriere studenti che ci è servito nella descrizione del modello relazionale**

Studenti

Matricola	Cognome	Nome	Nascita
276545	Rossi	Maria	25/11/1981
485745	Neri	Anna	23/04/1982
200768	Verdi	Fabio	12/02/1982
587614	Rossi	Luca	10/10/1981
937653	Bruni	Mario	01/12/1981

Esami

Studente	Voto	Lode	Corso
276545	28	N	1
276545	27	N	4
200768	30	S	1
587614	24	N	4

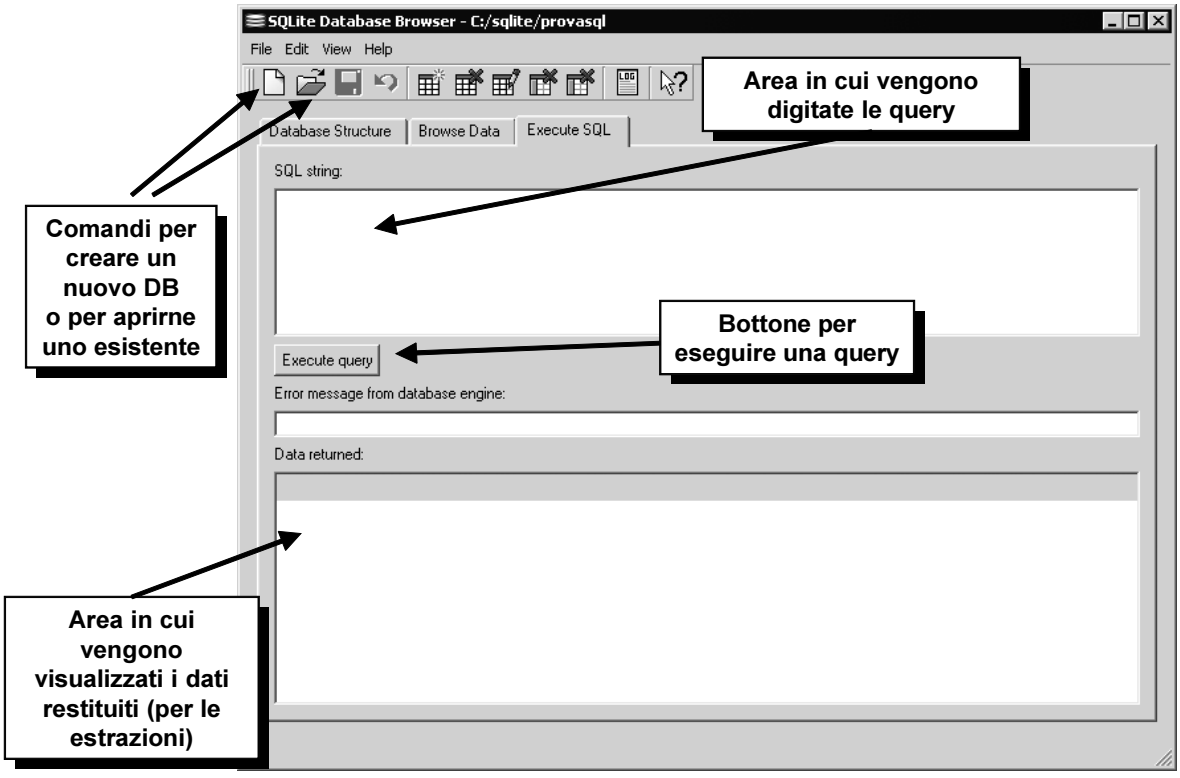
Corsi

Codice	Titolo	Crediti
1	Analisi	6
3	Fisica	6
4	Lab. Fisica	3

Strumenti

- **Per i nostri semplici esempi utilizzeremo un RDBMS molto semplice e gratuito chiamato SQLite**
- **SQLite è implementato sotto forma di una libreria che permette creare e gestire DB.**
- **Ogni DB è un singolo file che viene memorizzato sul disco locale del PC**
- **Permette di operare con un sottoinsieme abbastanza ricco di ANSI SQL 92**
- **Come ogni RDBMS anche per SQLite esistono applicazioni che permettono di lavorare sui DB con comandi SQL**
- **Utilizzeremo l'applicazione: SQLite Database Browser**

SQLite Database Explorer



DDL – Create table

- Immaginiamo di partire da zero con un database completamente vuoto
- La prima cosa da fare è creare le tabelle
- Utilizzeremo il comando DDL CREATE TABLE che ha la seguente sintassi:

```
CREATE TABLE nome_tabella
( attributo tipo [Vincolo di attributo], ...
    [, Vincoli di integrità di tupla]
    [, Vincoli di integrità referenziale]
)
```

- Mediante questo comando possiamo quindi definire
  - Nome della tabella
  - Attributi
  - Tipo degli attributi (il dominio)
  - Eventuali vincoli di integrità

I tipi di dati SQL

- SQL definisce un certo numero di tipi per gli attributi (come abbiamo detto un tipo equivale ad un dominio)
- Per semplicità ne prendiamo in considerazione solo alcuni tra i più comuni, riportando l' equivalente tipo Java

SQL	Java	Note
VARCHAR(n)	String	n = dimensione max della stringa
CHAR(1)	char	Carattere
INTEGER	int	Numero intero
FLOAT	float, double	Numero reale

- Abbiamo escluso i booleani perché molti DB non li prevedono
- Per le stringhe (varchar) bisogna indicare una dimensione massima (come in C)

Creazione delle tabelle studenti e corsi

- Creiamo innanzitutto la tabella studenti e la tabella corsi specificando le chiavi primarie:

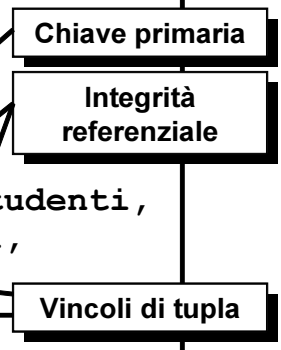
```
CREATE TABLE Studenti
(
  Matricola VARCHAR(6)NOT NULL,
  Cognome VARCHAR(50) NOT NULL,
  Nome VARCHAR(50) NOT NULL,
  Nascita VARCHAR(10),
  PRIMARY KEY (Matricola)
)
```

```
CREATE TABLE Corsi
(
    Codice INTEGER NOT NULL,
    Titolo VARCHAR(50),
    Crediti INTEGER,
    PRIMARY KEY (Codice)
)
```

Creazione tabella esami

- Creiamo la tabella esami con tutti i vincoli necessari: chiave primaria, vincoli di tupla e integrità referenziale

```
CREATE TABLE Esami
  (
    Studente VARCHAR(6) NOT NULL,
    Voto INTEGER,
    Lode CHAR(1),
    Corso INTEGER NOT NULL,
    PRIMARY KEY (Studente, Corso),
    FOREIGN KEY (Studente) REFERENCES studenti,
    FOREIGN KEY (Corso) REFERENCES corsi,
    CHECK (VOTO>18 AND VOTO<30),
    CHECK((NOT(Lode="S")) OR (Voto=30))
  )
```



Altri comandi DDL

- DDL mette a disposizione anche altri comandi:
- Per cancellare una tabella:

```
DROP TABLE nome_tabella
```

- Per rinominare una tabella

```
ALTER TABLE nome_tabella RENAME nuovo_nome
```

- Per modificarla aggiungendo, eliminando o rinominando colonne:

```
ALTER TABLE nome_tabella
  ADD COLUMN nome_attributo tipo [NOT NULL]
ALTER TABLE nome_tabella
  DROP COLUMN nome_attributo
ALTER TABLE nome_tabella
 CHANGE COLUMN column_definition
```

DML

- DML è la parte di SQL che consente di effettuare le seguenti operazioni
- Inserimento di una nuova riga in una tabella (comando INSERT)
- Cancellazione di una o più righe da una tabella (comando DELETE)
- Modifica di uno o più campi appartenenti a una o più righe ddi una tabella (comando UPDATE)
- Estrazione di dati da una o più tabelle (comando SELECT)
- Per esaminare il funzionamento di questi comandi continuiamo con il nostro esempio, popolando la base dati che abbiamo appena costruito

INSERT

- L'inserimento di una nuova riga viene fatto con il comando INSERT la cui sintassi è:

```
INSERT INTO tabella
 (attributo1,attributo2…,attributoN)
VALUES
 (valore1,valore2…,valoreN)
```

- Nel nostro esempio per inserire una riga nella tabella studenti scriveremo:

```
INSERT INTO Studenti
 (Matricola,Cognome,Nome,Nascita)
VALUES
 (276545,'Rossi','Maria','25/11/1981')
```

DELETE

- L'inserimento di una nuova riga viene fatto con il comando DELETE la cui sintassi è

  ```
  DELETE FROM tabella
  WHERE <Condizione>
  ```

- Attenzione: ricordarsi la clausola WHERE, se manca vengono cancellate tutte le righe della tabella!
- Esempi:

```
DELETE FROM Studenti
WHERE Matricola = 276545

DELETE FROM Esami
WHERE Corso=04 AND Voto<28
```


UPDATE

- L'inserimento di una nuova riga viene fatto con il comando UPDATE la cui sintassi è

  ```
  UPDATE Tabella
  SET Attributo1 = <Espressione>, Attributo2 =
  <Espressione>...
  WHERE <Condizione>
  ```

- Esempi:

```
UPDATE Esami
SET Voto = Voto + 1
WHERE Corso = 1

UPDATE Corsi
SET  Nome = 'FISICA LA', Crediti = 9
WHERE Codice=3
```

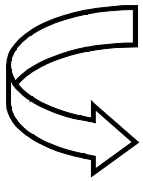
SELECT – Estrazione semplice

- Il comando SELECT permette di fare estrazioni di datai da un database a partire da una o più tabelle
- E' il più ricco e complesso fra i comandi
- Nella sua versione più semplice la sua sintassi è:

```
SELECT * FROM Tabella
WHERE <Condizione>
```

- Esempio

```
SELECT * FROM Studenti
WHERE Cognome = 'Rossi'
```



Matricola	Cognome	Nome	Nascita
276545	Rossi	Maria	25/11/1981
587614	Rossi	Luca	10/10/1981

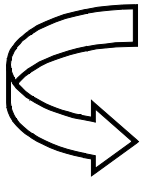
SELECT – Scelta campi

- Possiamo scegliere le colonne che vogliamo estrarre con la sintassi:

```
SELECT colonna1, colonna 2... FROM Tabella
WHERE <Condizione>
```

- Esempio:

```
SELECT Cognome, Nome  FROM Studenti
WHERE Cognome = 'Rossi'
```



Cognome	Nome
Rossi	Maria
Rossi	Luca

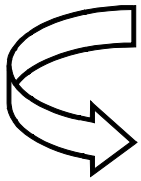
SELECT – Ordinamento

- Possiamo ordinare le righe usando una clausola order by:

```
SELECT colonna1, colonna2... FROM Tabella
WHERE <Condizione>
ORDER BY colonnaA, colonnaB
```

- Esempio:

```
SELECT Cognome, Nome, Nascita FROM Studenti
WHERE Cognome = 'Rossi'
ORDER BY Nome, Nascita
```

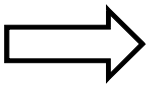


Cognome	Nome	Nascita
Rossi	Luca	10/10/1981
Rossi	Maria	25/11/1981

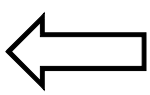
Join

- Finora abbiamo visto estrazioni da una sola tabella
- SQL permette di fare un'operazione chiamata Join che permette di utilizzare congiuntamente le informazioni contenute in più tabelle.
- In SQL un join si può formulare utilizzando i costrutti visti finora (FROM – WHERE), che permettono di compiere prodotti cartesiani e selezioni.
- Per esempio vogliamo vedere gli esami con accanto i nomi e cognomi degli studenti che li hanno sostenuti:

Da Studenti



Matricola	Cognome	Nome	Voto	Lode
276545	Rossi	Maria	28	N
276545	Rossi	Maria	27	N
200768	Verdi	Fabio	30	S
587614	Rossi	Luca	24	N



Da Esami

Esempio Join

- Scriveremo un comando SELECT in cui
  - La clausola FROM contiene due tabelle
  - La clausola WHERE contiene una condizione che coinvolge i campi corrispondenti delle due tabelle

```
SELECT Matricola, Cognome, Nome, Voto, Lode
FROM Studenti, Esami
WHERE Studenti.Matricola = Esami.Studente
```

- La clausola WHERE può naturalmente contenere altre condizioni per limitare la selezione:

```
SELECT Matricola, Cognome, Nome, Voto, Lode
FROM Studenti, Esami
WHERE Studenti.Matricola = Esami.Studente
AND Cognome = 'Rossi'
```


Operatori aggregati - 1

- Nell'elenco deglle colonne di un comando possiamo avere anche espressioni che calcolano valori a partire da insiemi di n-uple:
- Le funzioni sono `COUNT, MIN, MAX, AVG, TOT`
- La sintassi è

  `FUNZIONE(*)`
- Oppure

  `FUNZIONE(Attributo)`
- Per esempio se vogliamo contare quanti studenti hanno il cognome Rossi scriveremo:

```
SELECT COUNT(*) FROM Studenti
WHERE Cognome = 'Rossi'
```

Operatori aggregati - 2

- Se invece vogliamo calcolare la media dei voti di Rossi Maria (Matricola 276545), scriveremo:

```
SELECT AVG(Voto) FROM Esami
WHERE Studente = 276545
```

- Infine se vogliamo calcolare i crediti accumulati da Rossi Maria useremo un join e la funzione SUM:

```
SELECT SUM(Crediti) FROM Esami,Corsi
WHERE Esami.Corso=Corsi.Codice
AND Studente = 276545
```