INTRODUZIONE AI DATABASE RELAZIONALI

Base dati e tecniche di utilizzo in ambito scientifico – parte 2

Opus Facere – INAF Bologna

nicastro@iasfbo.inaf.it

http://ross.iasfbo.inaf.it/opusfacere/

Quale usare – gratuiti

en.wikipedia.org/wiki/Relational_database

- mysql.com / dev.mysql.com

- mariadb.org

- postgresql.org

- sqlite.org

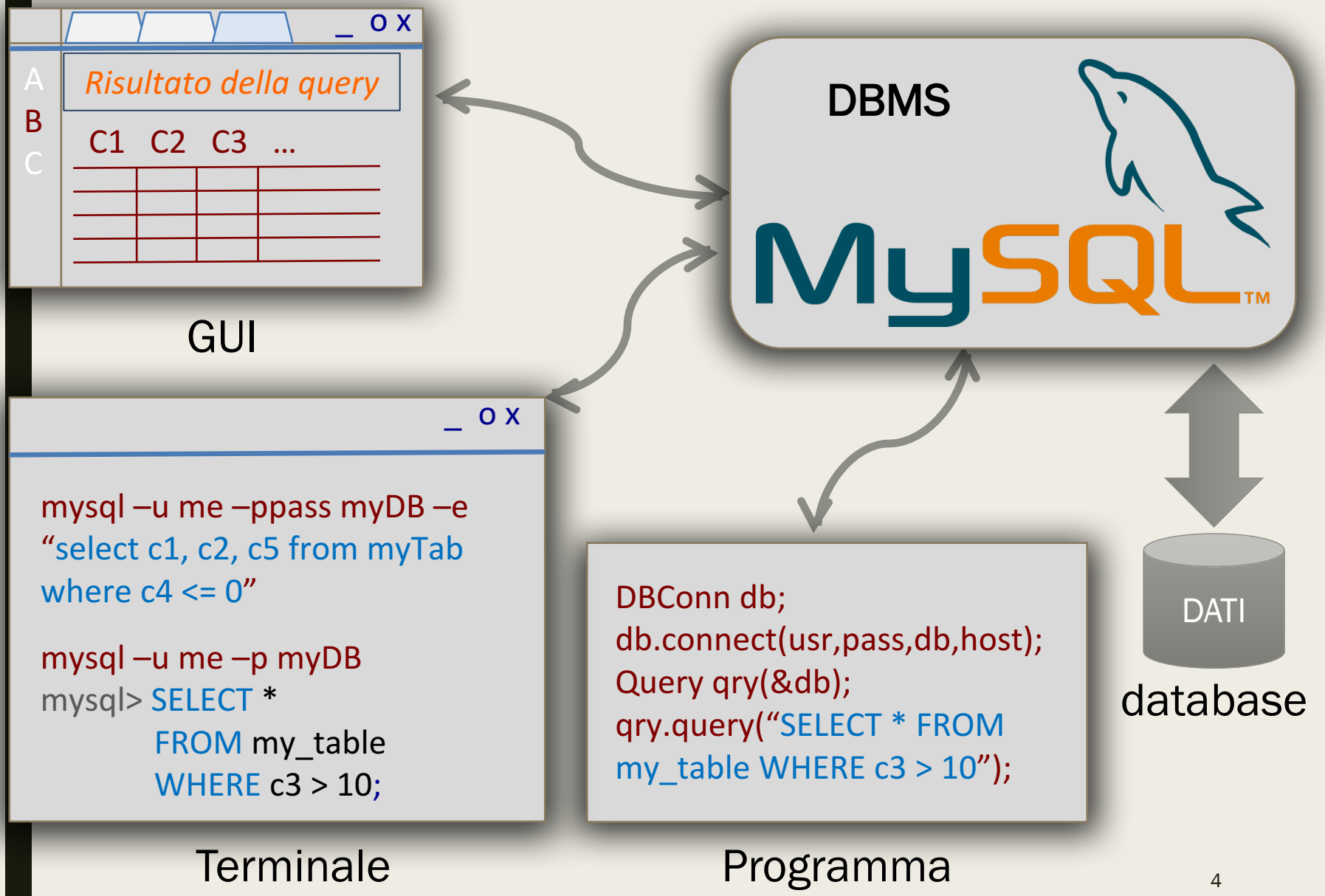
Introduzione a MySQL

Il più usato RDBMS **Open Source** *

- **Gratuito**: versione corrente MySQL 5.7, scaricabile da dev.mysql.com
- E' un sistema di gestione di basi di dati relazionali (RDBMS) → composto da un insieme di relazioni
- E' stabile e veloce
- Comprende tutte le caratteristiche considerate più importanti dalla comunità delle basi di dati, quali:
  - Transazioni
  - Vincoli a livello di record
  - Chiavi esterne
  - Interrogazioni annidate
  - Ricerca testuale

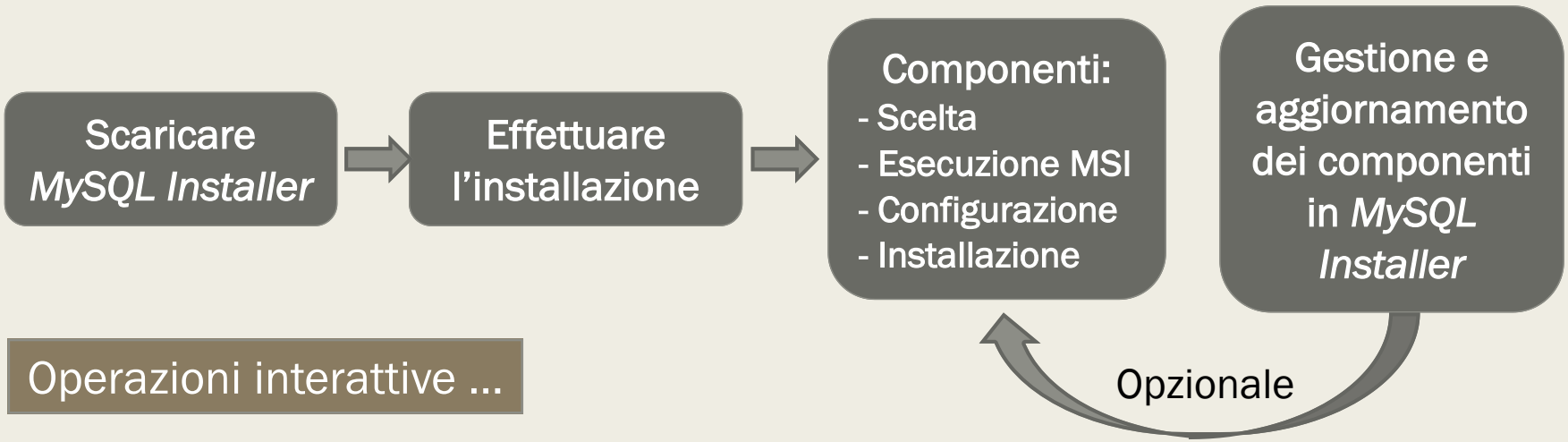
* Ottobre 2017

Introduzione a MySQL



Installazione MySQL

- Come per altri applicativi, ogni Sistema Operativo prevede un'installazione ad hoc.
  - **Windows**
    https://dev.mysql.com/doc/refman/5.7/en/mysql-installer.html
    https://dev.mysql.com/doc/refman/5.7/en/mysql-installer-setup.html



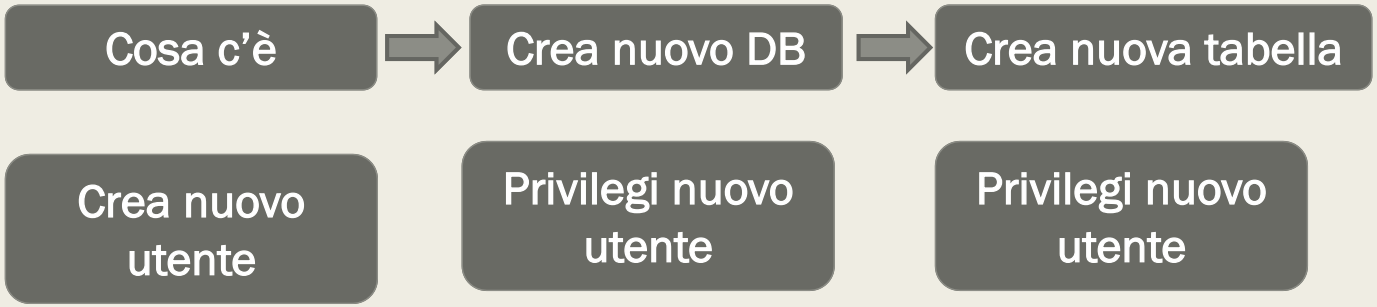
Avvio MySQL

- Avvio del *Server* (vedi operazione di installazione)
  - Può essere avviato in automatico o a richiesta
- Avvio del *Client*
  - Start → Programs → MySQL → MySQL 5.7 … Client

```
mysql>
```

Si inizia come utente *root* (amministratore)

Prime operazioni …



Comandi base

- Vediamo cosa c'è

```
mysql> show databases;
```

- Creare un database

```
mysql> create database nome;
```

- Es.

```
mysql> create database opus2017;
```

- Posizionarsi nel database (verifica con `show databases;`)

```
mysql> use nome_database;
```

- Es.

```
mysql> use opus2017;
```

Comandi base

- Creare una tabella. Le parentesi [] significa “opzionale”.

```
mysql> create [temporary] table [if not exist]
nome_tabella [(definizioni_create)] [opzioni
tabella] [comandi di selezione];
```

- Es.

```
mysql> create table utenti(
Id int not null AUTO_INCREMENT,
Utente varchar(20) not null default ‘nessuno’,
Iscritto datetime,
PRIMARY KEY (Id)
);
```

Notare AUTO_INCREMENT. Si chiede che ad ogni nuovo inserimento il valore di questo campo (Id) venga incrementato di 1, indipendentemente da quello che si inserisce.

Comandi base

- Inserire dati in una tabella (stringhe e date tra apici !)

```
mysql> INSERT INTO nome_tabella VALUES
(contenuto_primo_campo,'contenuto secondo campo',
'terzo campo', …), (…), …;
```

- Es. (notare il primo campo sempre 0. OK qualsiasi numero.)

```
mysql> insert into utenti values
(0, 'Tizio', '2017-10-17 17:04:00'),
(0, 'Caio', '2017-10-17 17:04:10'),
(0, 'Sempronio', '2017-10-17 17:04:20');
```

- Interrogazione

```
mysql> SELECT * FROM utenti;
```

Comandi base

- Sostituzione dei dati nella tabella

```
mysql> REPLACE INTO nome_tabella VALUES
(contenuto_primo_campo,'contenuto secondo campo',
'terzo campo', …);
```

- Modifica dei dati

```
mysql> UPDATE nome_tabella SET campo1=valore1,
campo2=valore2, … [WHERE condizione];
```

- Cancellazione dei dati

```
mysql> DELETE FROM nome_tabella [WHERE ()];
```

- Cancellazione della tabella

```
mysql> DROP TABLE nome_tabella;
```

Per gli altri comandi: ross.iasfbo.inaf.it/opusfacere (sezione Documenti)

Comandi MySQL (lato *client*)

```
mysql> ?
Note that all text commands must be first on line and end with ';'

?         (\?) Synonym for `help'.
clear     (\c) Clear the current input statement.
connect   (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter.
edit      (\e) Edit command with $EDITOR.
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
nopager   (\n) Disable pager, print to stdout.
notee     (\t) Don't write into outfile.
pager     (\P) Set PAGER [to_pager]. Print the query results via PAGER.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.rehash    (\#) Rebuild completion hash.
source    (\.) Execute an SQL script file. Takes a file name as an argument.
status    (\s) Get status information from the server.
system    (\!) Execute a system shell command.
tee       (\T) Set outfile [to_outfile]. Append everything into given outfile.
use       (\u) Use another database. Takes database name as argument.
charset   (\C) Switch to another charset. Might be needed for processing binlog with ...
warnings  (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.
resetconnection(\x) Clean session context.

For server side help, type 'help contents'
```

… alcuni suggerimenti / info

- usare nomi descrittivi che riflettono il soggetto della tabella o i dati;
- non usare nomi specifici per dati che potrebbero assumere una forma più generale;
- evitare abbreviazioni e acronimi;
- non usare punteggiatura o spazi nei nomi;
- usare nomi delle tabelle plurali, nomi dei campi singolari;
- tabelle di relazione hanno di solito come nome la combinazione dei nomi delle tabelle che collegano.

Nota 1: tutti i comandi e i nomi dei campi sono *case insensitive*.

Nota 2: i nomi dei DB e delle Tabelle "*possono*" essere *sensitive*, a seconda del Sistema Operativo e della configurazione del server.

I concetti base del DB relazionale

Per approfondimenti usare **internet**

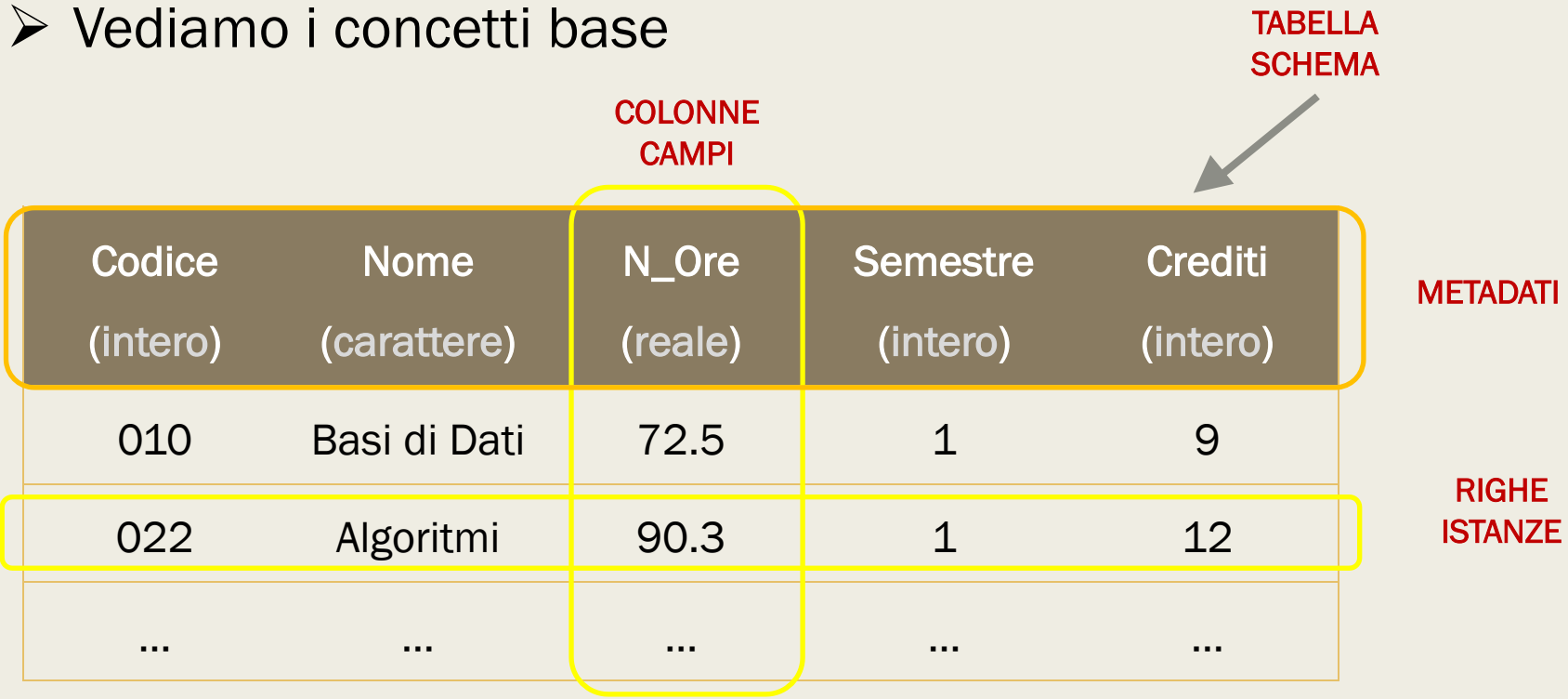
- Nomenclatura base:
  - **tabella**
  - **record**
  - **campi**
  - **chiavi**
  - **query**
  - **SQL**
  - **DBMS**

Terminologia

- **tabella**
- E' una raccolta di dati organizzati in righe e colonne.
- Ogni tabella, in genere, rappresenta un raccolta di informazioni su uno specifico argomento o tematica.
- Ad esempio possiamo avere una tabella per gli studenti, una per i corsi e una per i docenti.
- **record** (riga, istanza, tupla)
- E' una singola riga di una tabella.
- Ci consente di identificare un determinato insieme di dati, all'interno di tutti quelli che sono contenuti nella tabella.
- **campi**
- Sono le celle di una colonna che compongono la tabella e ne definiscono la struttura; si può specificare il tipo (formato) di ciascuna colonna (testo, numerico, intero e floating, ecc.) e si può imporre una certa serie di regole, ad es. che un campo non può essere vuoto (null) o deve avere un massimo numero di caratteri.

Le tabelle

- Vediamo i concetti base



- Ad es. chiamiamo la tabella sopra "**corsi**"
- I più comuni tipi di dati sono: intero, reale (float), carattere (stringa), data (timestamp).

Terminologia

- **chiavi** (primarie, esterne)

- E' l'insieme dei campi che permette di identificare in modo univoco ed inequivocabile un singolo record all'interno della tabella.
- Ogni tabella può avere una o più chiavi.
- Non si possono avere in tabella due record distinti con lo stesso valore del campo chiave.
- Non è possibile avere chiavi contenenti valore NULL.
- Una chiave può essere composta da uno o più campi.
- E' un elemento fondamentale per i database relazionali.

Terminologia

➢ query

- E' un'interrogazione sul Database utile per per estrarre, riorganizzare, riclassificare i dati.
- Fornisce come risultato un insieme di dati che soddisfano le condizioni imposte in essa.
- Normalmente negli strumenti per la gestione dei DB si ha la possibilità di creare la query.
- Sia con apposite interfacce facilitate, sia scrivendola direttamente in un linguaggio apposito.

I concetti base del DB relazionale

- **SQL** (*Structured Query Language - pronunciato "ES-Q-EL"*)

- E' il linguaggio utilizzato per accedere e gestire i dati di un (R)DBMS, per creare e modificare la struttura del database e per gestire l'accesso agli oggetti contenuti.
- Sebbene SQL sia uno standard ANSI e ISO, molti RDBMS estendono supportano estensioni al linguaggio (dialetti). La prima versione fu sviluppata da IBM nei primi anni '70 da D. D. Chamberlin e R. F. Boyce col nome di SEQUEL. IBM brevettò questa versione del linguaggio nel 1985 chiamandolo SQL, nel 1986 fu formalmente standardizzato dall'American *National Standards Institute*. Le versioni successive vennero rilasciate come standard dall'*International Standardization Organization* (ISO). Attualmente lo standard è SQL:2008.

SQL: Structured Query Language

- SQL è un linguaggio di interrogazione e manipolazione della base dati e delle informazioni in essa contenute.
- Non è un linguaggio imperativo/procedurale, è un linguaggio dichiarativo: manca dei concetti di variabile e di struttura di controllo algoritmica, tipica dei linguaggi imperativi e della programmazione strutturata.
- E' costituito da tre insiemi di istruzioni:
  - **DDL** (*Data Definition Language*): per definire la struttura della base dati
  - **DCL** (*Data Control Language*): per gestire i criteri di protezione e di accesso ai dati
  - **DML** (*Data Manipulation Language*): per operare sui dati
  - **TCL** (*Transactional Control Language*): per gestire le varie transazioni nel DB. Es. `COMMIT`, `ROLLBACK`.

SQL: Structured Query Language

➢ Query e SQL

Una query scritta in SQL può avere questa forma:

`Select – From – Where`

– **SELECT**: per indicare i campi richiesti (* per tutti)

– **FROM**: per indicare su quali tabelle si deve effettuare la query

– **WHERE**: per indicare i vincoli imposti

Ad esempio:

```
SELECT Nome, N_Ore, Crediti
FROM Corsi
WHERE Codice=22;
```

➢ Può essere usato per costruire scripts, funzioni e procedure.

➢ I comandi più usati: `SELECT`, `INSERT`, `DELETE`, `UPDATE`.

Il catalogo di Messier

Metadati

Id	Type	Const	Mag	Ra_h	Ra_m	Dec_d	Dec_p	Dist	App_size
Int	char	char	float	float	float	float	float	char	char
1	BN	Tau	8.2	05	34.5	+22	01.0	6.3 kly	6'x4'
2	GC	Aqu	6.3	21	33.5	00	-49.0	36.2 kly	12.9'
3	GC	CVn	6.3	13	42.2	+28	23.0	30.6 kly	16.2'
4	GC	Sco	6.4	16	23.5	-26	31.5	6.8 kly	26.3'
5	GC	Ser	6.2	15	18.6	+02	05.0	22.8 kly	17.4'
6	OC	Sco	4.2	17	40.4	-32	13.8	2 kly	33'
7	OC	Sco	4.1	17	53.9	-34	47.0	800 ly	80.0'
8	BN	Sag	6.0	18	04.1	-24	18.0	5200 ly	90'x40'
9	GC	Oph	7.3	17	19.2	-18	31.0	26.4 kly	9.3'

– Dati –

- *Dati + metadati = file*, ad es. *excel, xml, FITS, ecc.*
- *Dati + metadati + software = DATABASE !*

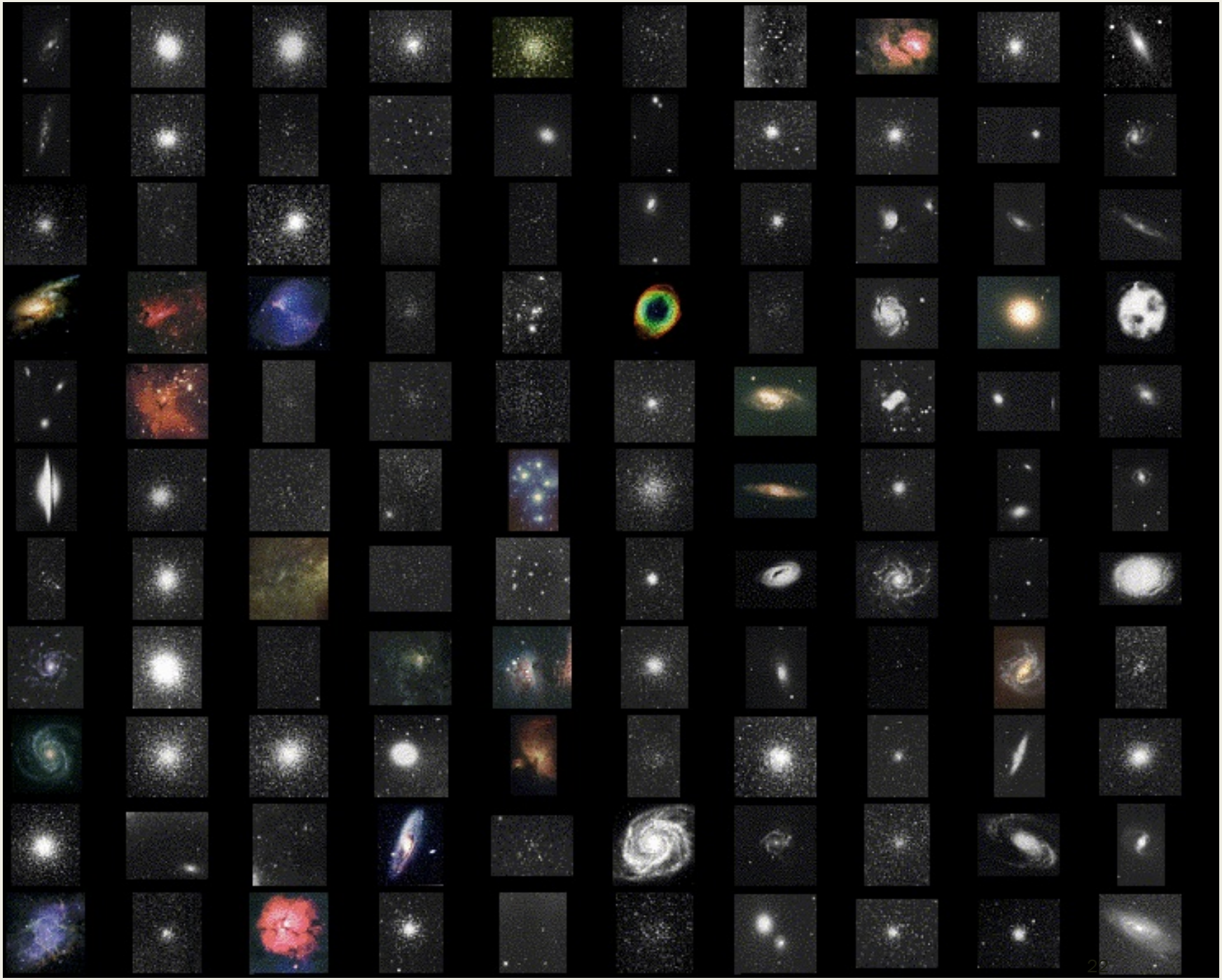
11

10

...

2

1



Data Definition Language

- E' il set di istruzioni di SQL per la definizione della struttura della base dati.
- Sono tre le istruzioni principali del DDL:
  - `create`: per la creazione di database, tabelle, indici, viste, ecc.
  - `alter`: per la modifica della struttura di una tabella o di altri oggetti interni ad una base dati
  - `drop`: per l'eliminazione di una tabella, di un intero database o di altri oggetti
  - vedere anche `truncate`, `comment`, `rename`
- Creazione di un database:

```
create database corsi2017;
```

- Cancellazione di un database:

```
drop database corsi2017;
```

DDL: Dominio di definizione degli attributi

- Definire una tabella significa definirne gli attributi e il dominio degli stessi attributi.
- I domini sono quelli tipici di ogni linguaggio di programmazione:
  - numeri interi (`int`, `integer`, `decimal`)
  - numeri a virgola mobile – floating point (`float`, `real`, `double`)
  - stringhe di caratteri (`char, varchar, text`)
  - date (`date`, `time`, `timestamp`, *year, datetime*)
- Altri tipi possono essere definiti sulla base di specifiche tipiche di un determinato RDBMS.
- Ogni attributo può essere anche caratterizzato da un valore di default e una serie di vincoli (es. "`not null`" per i campi obbligatori di una tabella).
- Tra i vincoli vi è la possibilità di aggiungere un attributo alla chiave primaria della tabella o l'attributo di chiave esterna referenziando uno o più campi chiave di un'altra tabella.

Data Control Language

- E' il set di istruzioni di SQL per la definizione dei permessi di accesso sui database e sulle tabelle e per la gestione degli account utente.
- Sono due le istruzioni principali del DCL:
  - `grant`: per assegnare un determinato permesso ad un utente
  - `revoke`: revocare un determinato permesso ad un utente
- Concessione di permessi:

  `grant` *privilegi* `on` *risorsa* `to` *utenti* [`with grant option`];
- Esempio:

```
grant select on corsi to rossi@localhost;
```

Data Control Language – 2

- Gestione account: utenti, permessi, verifica stato, ecc.
  - `CREATE USER`, `DROP USER`, `RENAME USER`, `SET PASSWORD`, ecc.
- Gestione delle tabelle:
  - `ANALYSE TABLE`, `CHECK TABLE`, `REPAIR TABLE`
- Plugin e User-Defined Function Statements:
  - `CREATE/DROP FUNCTION`, `INSTALL/UNINSTALL PLUGIN`

Data Manipulation Language

- E' il set di istruzioni di SQL per la gestione delle informazioni presenti nella base dati
- Sono quattro le istruzioni principali del DML:
  - `insert`: per inserire dati in una tabella
  - `select`: selezionare in base a determinati criteri o condizioni i dati presenti in una o più tabelle
  - `update`: per modificare i dati di una tabella sulla base di un determinato criterio di selezione
  - `delete`: per eliminare i record di una tabella corrispondenti ad una determinata condizione
  - vedere anche `call`, `lock table`, `replace/merge`, `do`, `handler`, `load data infile`, …

SQL: utilità

- Visualizzare lo stato del server, delle connessioni, ecc.:
  - `SET`, `SHOW`, `DESCRIBE` (`SHOW COLUMNS FROM`), `EXPLAIN`, `HELP`, `USE`, `FLUSH`, `KILL`, `RESET`, es..
  - `SHOW item` – dà informazioni per una lunga lista di cose: database, tabelle, view, colonne, funzioni, stato, riguardo al server, ecc.
  - `EXPLAIN` item – riporta come verrà eseguita la query, il tipo di accesso dei data, ecc. Es.

```
EXPLAIN SELECT la_tua_query;
```

- Alcune clausole comuni nel `SELECT`:
  - `LIMIT`, `HAVING`, `GROUP BY`, `ORDER BY`, `INTO OUTFILE`, `BETWEEN`, `IN`

Aggregazione

SQL: funzioni

- La maggior parte sono SQL standard, ma altre sono specifiche dei DBMS. Per MySQL:
  - Operatori (con conversione automatica di tipo nelle espressioni)
  - Funzioni di controllo di flusso (`IF()`, `CASE`, ...)
  - Funzioni su Stringhe, Funzioni e Operatori numerici
  - Funzioni su Date and Tempo
  - Funzioni di ricerca *Full-Text*
  - Funzioni e Operatori di *Cast*
  - Funzioni *XML*
  - Funzioni *Bit*

SQL: funzioni – 2

- Funzioni di Encriptazione e Compressione
- Funzioni informative e simili
- Funzioni e Modificatori da usare con clausole `GROUP BY`
- Estensioni Spaziali (2-d)
- Matematica di precisione

➢ `Es.:`

- `COUNT`, `MIN`, `MAX`, `SUM`, `AVG`, STDDEV

Funzioni di aggregazione

SQL: importanza dell'indicizzazione

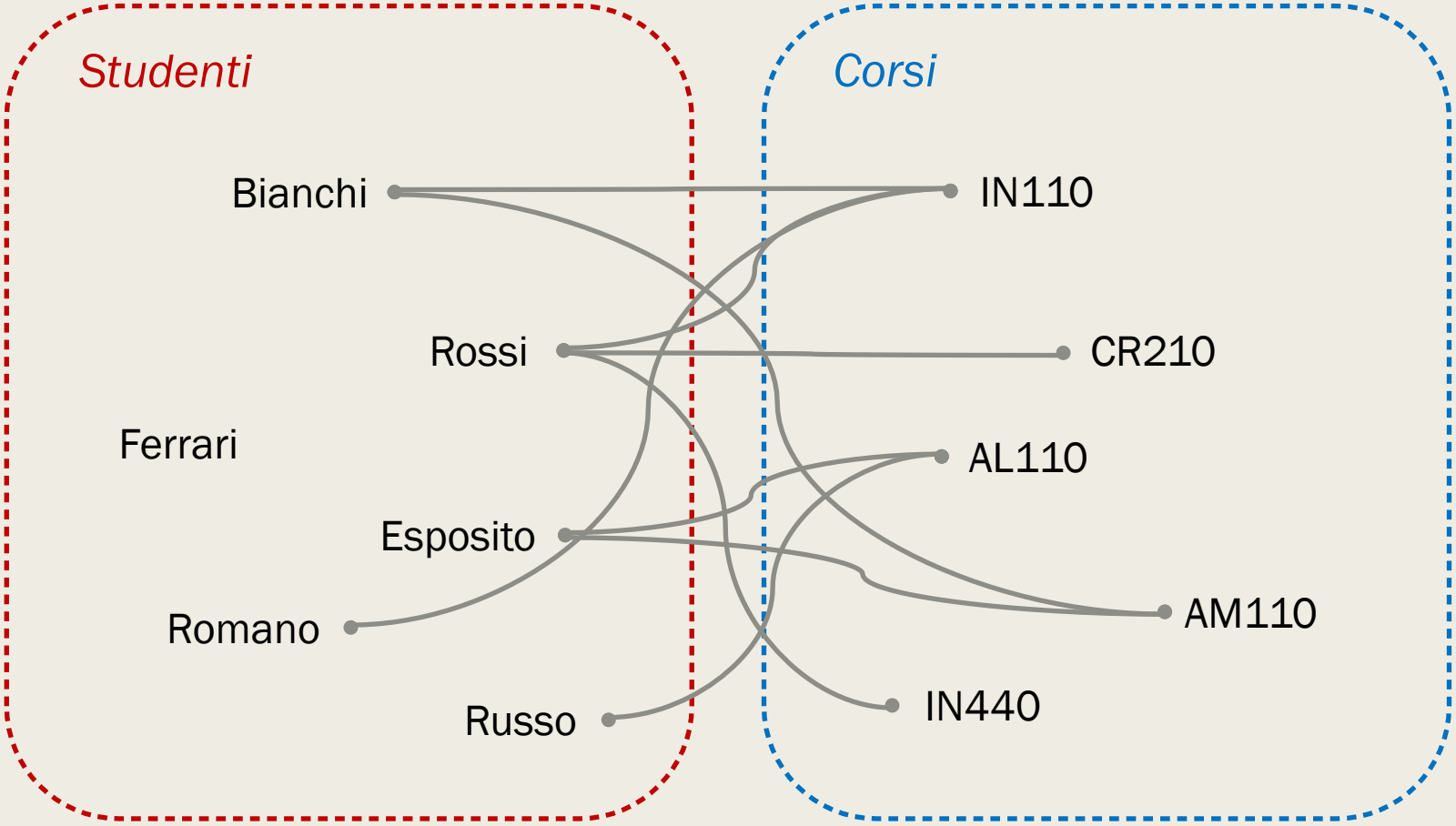
- Ogni tabella può avere 1 o più indici (chiavi). Possono anche non essere univoci ma identificare sottoinsiemi di righe.
- Fondamentale per le operazioni relazionali → `JOIN`
- Es.: tutte le lingue parlate in Oceania:

```
SELECT * FROM Country
JOIN CountryLanguage
ON Country.Code=CountryLanguage.CountryCode
WHERE Continent='Oceania';
```

Tipi di relazione

- Uno a Molti (1 – •)
  - Relazione che mette in rapporto le informazioni di due tabelle secondo uno schema gerarchico *genitore-figlio*. Per generarla è necessario che la tabella genitore abbia una chiave primaria (campo univoco) e la tabella figlio abbia una chiave esterna (campo indice con duplicati ammessi) dello stesso formato.
- Uno a Uno (1 – 1)
  - Relazione che mette in rapporto le informazioni di due tabelle secondo una relazione biunivoca. Ad ogni record di una tabella può corrispondere un solo record di quella correlata. Per generarla è necessario che entrambe le tabelle dispongano di una chiave primaria (campo univoco) e dello stesso formato.
- Molti a Molti (• – •)
  - Si tratta in realtà di una relazione fittizia. Apparentemente ad un record di una tabella possono corrispondere molti record di un'altra tabella e viceversa, in realtà questa relazione è costituita da una tabella di collegamento correlata alle tabelle principali per mezzo di due relazioni uno a molti.

Tipi di relazione – visione grafica



Tipi di relazione – costruzione

- Un prerequisito alla costruzione delle relazioni è garantire che ogni tabella abbia una chiave univoca per identificare le singole righe della tabella.
- Qualunque campo esistente contenente un valore univoco è un candidato accettabile per essere utilizzato come chiave.
- Una soluzione migliore è quella di aggiungere un campo arbitrario ad ogni tabella con il compito specifico di fungere da campo chiave.
- Tale campo deve contenere un valore univoco, anche senza un particolare significato per i dati contenuti nella tabella.
- Il valore della chiave è tipicamente un intero assegnato ad ogni record inserito nella tabella e mai ripetuto.
- Le relazioni tra tabelle possono essere costruite usando le chiavi delle singole tabelle.

DBMS: componenti – *continua...*