



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Un esempio di RDBMS: MySQL

Basi di Dati

Corso di Laurea in Informatica per il Management

Alma Mater Studiorum - Università di Bologna

Prof. Marco Di Felice

Dipartimento di Informatica – Scienza e Ingegneria

marco.difelice3@unibo.it

MySQL: Overview del tool

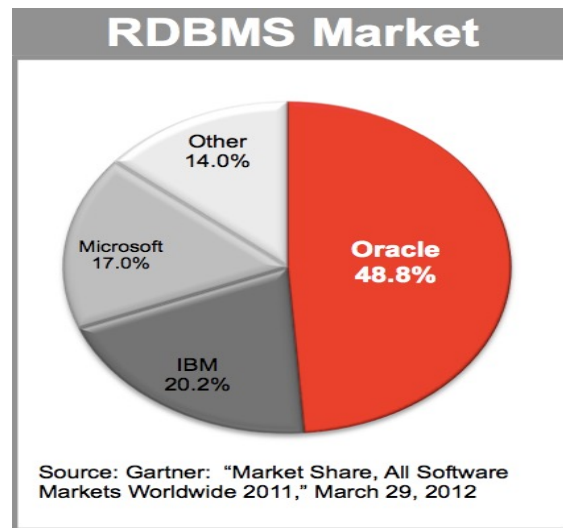
- **MySQL** → DBMS basato sul modello relazionale (RDBMS)
- Sviluppato nel 1995 della MySQL AB, dal 2008 di proprietà della SUN, ora di proprietà di **Oracle Corporation**.
- Disponibile in **diverse licenze/versioni** (Enterprise Edition, Community Server, Cluster, ...)
- Ultima versione: **8.0.31** (Community Server)
- **Multiplatforma**: Windows/Linux/Mac OSX/...

MySQL: Overview del tool

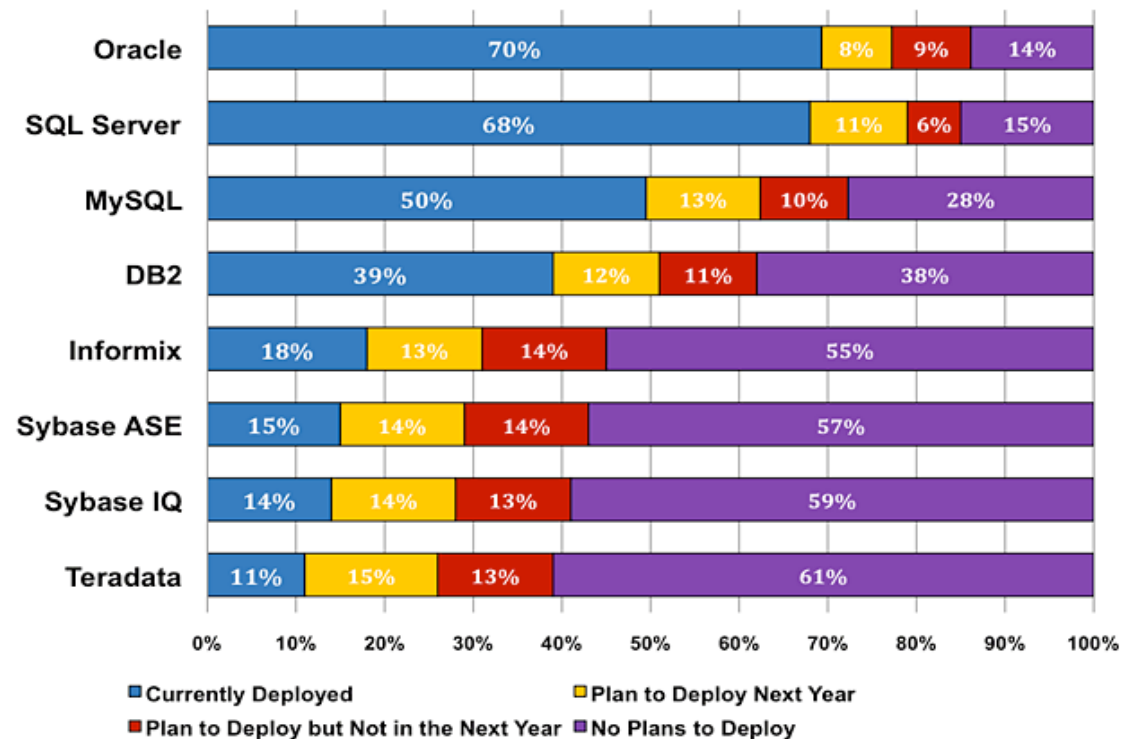
LICENZE D'USO

- MySQL è un tool **Open Source** (dal 2000).
- Esistono **diversi fork** del progetto (es. MariaDB).
- Due possibili **licenze**:
 - **GNU/GPL** (progetti open-source)
 - **Commerciale**

MySQL: Overview del tool

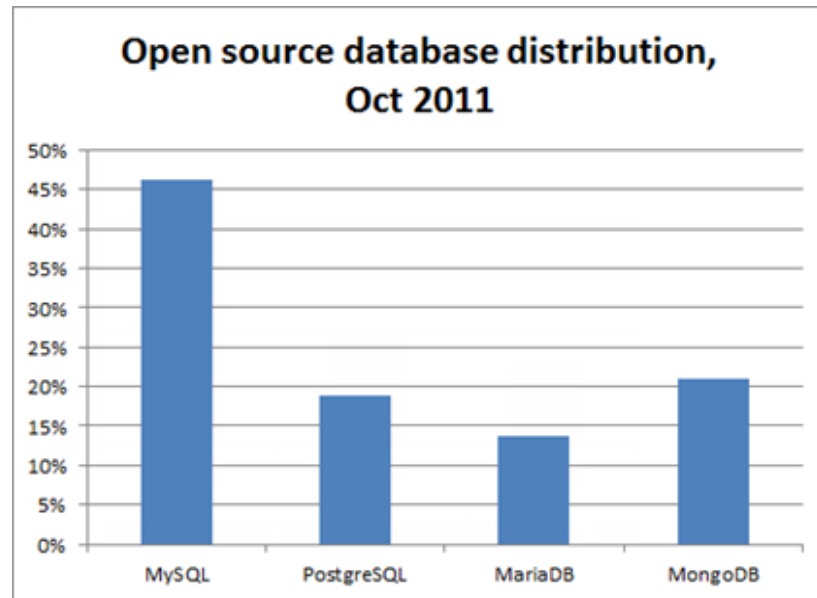


Database installations and deployment plans
Gartner study, 2008



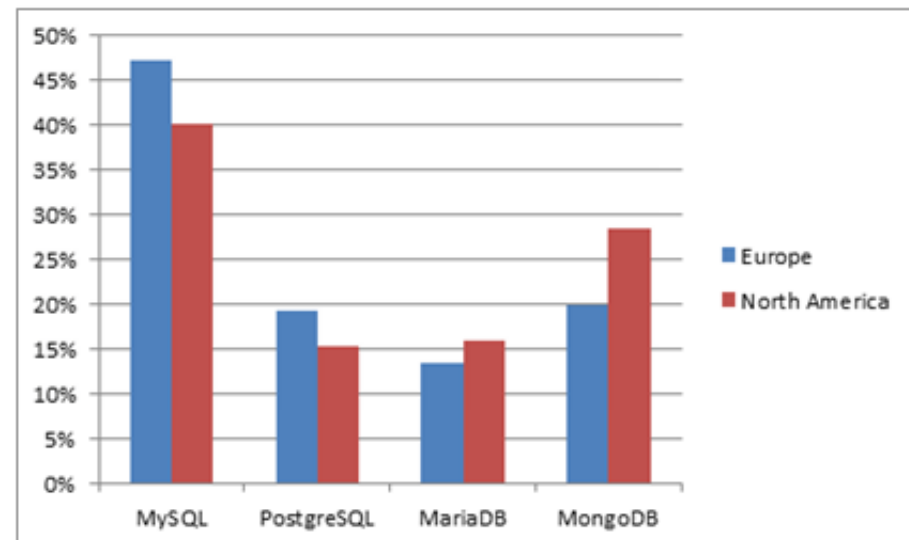
MySQL: Overview del tool

MySQL nel mercato dei DBMS open source ...



Source: Hosting statistics from Jelastic.com. Attribution required.

Open source database market share by region. October 2011.



Source: Hosting statistics from Jelastic.com. Attribution required.

MySQL: Overview del tool

- **Utilizzato** internamente come DBMS da molte aziende:

Google, Yahoo!, CERN, MIT, Bank of Canada Deutsche, Post Ministère de la Justice, NASA, Swedish National Police, United Nations, FAO, Braun, Daimler, Chrysler, Epson, Yamaha, BBC SEAT, The Weather Channel, PHP-Nuke, BMC, Dell, 3COM, Nokia ...

- **Integrato** in prodotti software venduti da terze parti

- **Integrato** in applicazioni WIS (Web Information System)

MySQL: Overview del tool

- Supporta gran parte dei costrutti del **linguaggio SQL 2.0** (viste, query annidate, vincoli di chiave, etc), con trigger e viste aggiornabili.
- Supporta **l'esecuzione di transazioni** su un tipo particolare di tabelle (**INNODB**).
- Supporta **molte tipologie di dati** numerici, testuali (es. VARCHAR), temporali (es. DATE) e binari (es. file di dati).
- Dispone di un proprio **linguaggio di estensione procedurale** per definire le **stored procedures**.

MySQL: Overview del tool

- MySQL **NON** ha limiti espliciti sulla **dimensione massima** di un database e sul numero di tabelle (eccetto per INNODB → 4 miliardi).
- Il **numero massimo di righe** in una tabella **dipende dai vincoli imposti dal sistema operativo** sulla dimensione max di un file.

Sistema Operativo	Dimensione Max File
Linux (ext4)	16 TB
Windows 8 (NTFS)	16 TB, up to 256 TB
MacOs X (HFS+)	8 EB

MySQL: Overview del tool

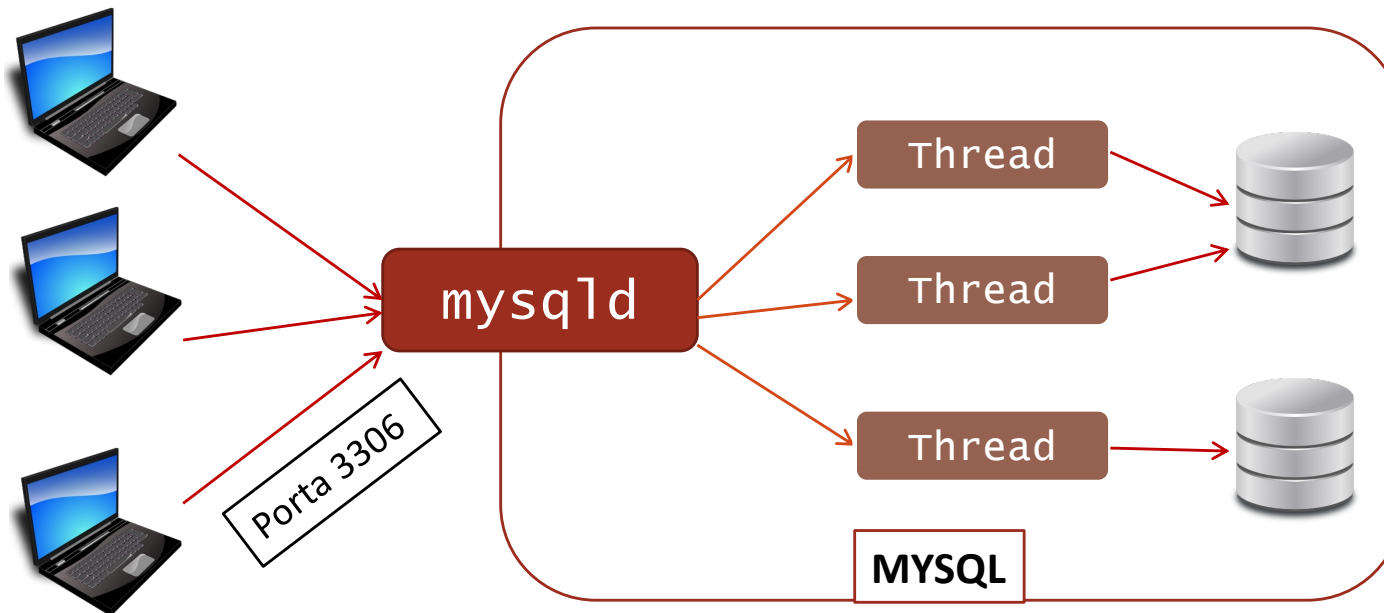
- Non esistono problemi dal punto di vista della concorrenza in termini di **numero massimo di connessioni simultanee** al server MySQL.

max_user_connection [0: 4294967295]

- ... Tuttavia, i problemi emergono –ovviamente- dal punto di vista delle **risorse** (es. memoria), per cui il **numero effettivo di connessioni simultanee supportate** dipende dalle capacità e dalle risorse hardware dell'elaboratore.

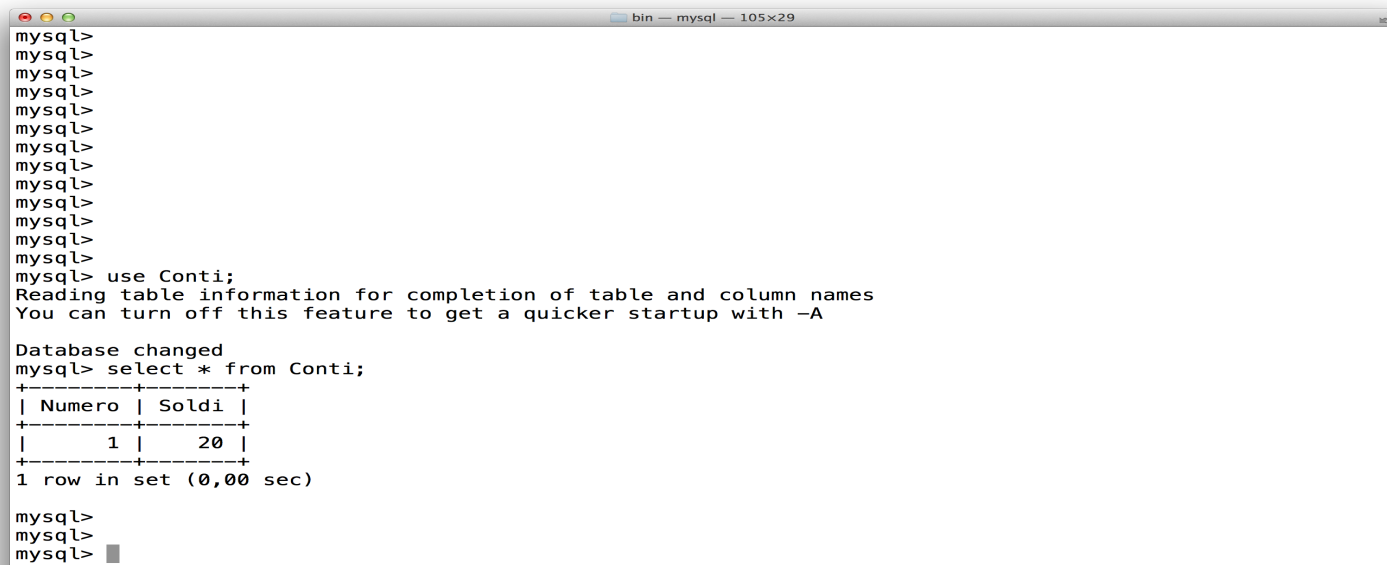
MySQL: Architettura Software

Architettura Client-Server del sistema software



MySQL: Architettura Software

- Una possibilità di interazione con il tool MySQL è attraverso il **terminale SQL** a riga di comando ... (comandi → istruzioni SQL)



```
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> use Conti;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from Conti;
+-----+-----+
| Numero | Soldi |
+-----+-----+
|      1 |    20 |
+-----+-----+
1 row in set (0,00 sec)

mysql>
mysql>
mysql>
```

MySQL: Autenticazione

- Connessione via shell:
mysql -u utente -p password

```
mysql -u root -p root
```

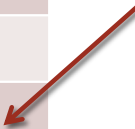
```
mysql -h localhost -u marco -P 9999 < script.sql -p
```

MySQL: Autenticazione

- L'autenticazione avviene attraverso la combinazione: **nomeutente@host**.
- La gestione dei dati degli utenti avviene attraverso la tabella `mysql.user`.

Host	User
%	marco
localhost	michele
localhost	

E' possibile lasciare vuoto il campo user.



```
SELECT * FROM mysql.USER;
```

MySQL: Autenticazione

- **Creare** un nuovo utente (locale):
`CREATE USER nome@localhost;`
- **Impostare la password** di un utente:
`SET PASSWORD FOR nome@localhost=PASSWORD('passwd');`
- Creare un nuovo utente con password:
`CREATE USER nome@localhost
IDENTIFIED BY 'passwd';`
- **Cancellare un utente**:
`DROP USER nome@localhost;`

MySQL: Creazione di database

- Per **creare un nuovo database**:
CREATE DATABASE [IF NOT EXIST] nome_db;
- Per **rimuovere un database**:
DROP DATABASE [IF EXISTS] nome_db;

In MySQL, un database e' una sottodirectory della directory dei dati (/Applications/MAMP/db/mysql/). Aggiungere un nuovo database corrisponde alla creazione di una nuova directory.

MySQL: Creazione di database

- Per vedere quali **db sono presenti** nel sistema:
SHOW databases;
- Per impostare il **db corrente**:
USE nome_database;
- **Due modi** per **assegnare/rimuovere un privilegio** ad un determinato utente:
 - ✧ Comandi SQL: REVOKE | INSERT
 - ✧ Aggiornare tabelle *mysql.user*, *mysql.db*, *mysql.tables_priv* attraverso INSERT/UPDATE.

MySQL: Creazione di tabelle

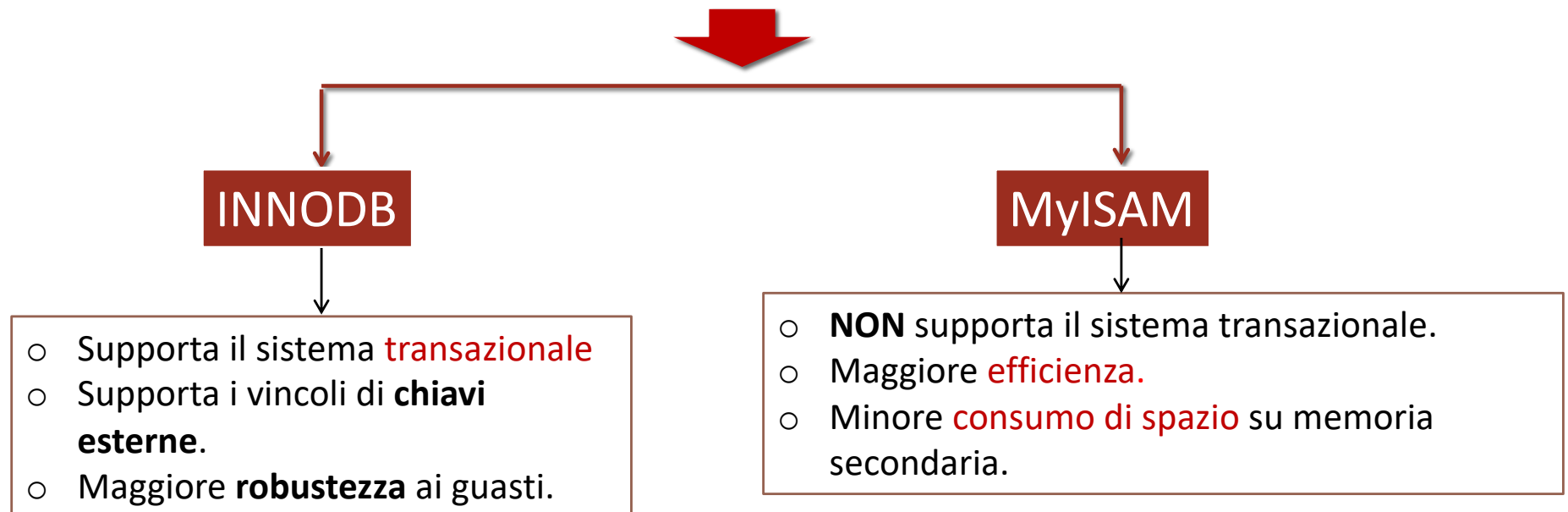
- Per **creare** una tabella:

```
CREATE [TEMPORARY] TABLE  
    nome_tabella | nome_db.nome_tabella  
    [definizione attributi]  
    [opzioni]  
    [select]
```

- E' possibile generare una tabella valida solo per la **sessione corrente** (opzione TEMPORARY).
- E' possibile **popolare** la tabella con il risultato di una query SELECT da altre tabelle.

MySQL: Creazione di tabelle

MySQL supporta diversi tipi di “**storage engine**” (in pratica, **tipi di tabelle**), tra cui i principali sono:



MySQL: Creazione di tabelle

- Per **creare** una tabella:

```
CREATE [TEMPORARY] TABLE  
    nome_tabella | nome_db.nome_tabella  
    [definizione attributi]  
    [opzioni]  
    [select]
```

- E' possibile generare una tabella valida solo per la **sessione corrente** (opzione TEMPORARY).
- E' possibile **popolare** la tabella con il risultato di una query SELECT da altre tabelle.

MySQL: Creazione di tabelle

Esempi di **opzioni** valide sulle tabelle:

- **ENGINE = tipo_tabella (ISAM|INNODB)**
- **AUTO_INCREMENT = valore**
- **AVG_ROW_LENGTH = valore**
- **CHECKSUM = { 0 | 1 }**
- **COMMENT = stringa**
- **MAX_ROWS= valore**

MySQL: Creazione di tabelle

- Per **creare** una tabella:

```
CREATE [TEMPORARY] TABLE  
    nome_tabella | nome_db.nome_tabella  
    [definizione attributi]  
    [opzioni]  
    [select]
```

- E' possibile generare una tabella valida solo per la **sessione corrente** (opzione TEMPORARY).
- E' possibile **popolare** la tabella con il risultato di una query SELECT da altre tabelle.

MySQL: Creazione di tabelle

Sintassi per specificare una **colonna della tabella**.

Nome_colonna TIPO
[NOT NULL | NULL] [DEFAULT valore]
[AUTO_INCREMENT]
[UNIQUE | PRIMARY KEY]
[COMMENT 'commento']

MySQL: Creazione di tabelle

- Per definire i vincoli di **integrità referenziale**:


```
FOREIGN KEY (nome_colonna_interna)
REFERENCES      nome_tabella_esterna
(nome_colonna_esterna)
[ON DELETE | ON UPDATE
  RESTRICT | CASCADE | SET NULL |
  NO ACTION ]
```

Funziona SOLO con tabelle di tipo INNODB ...

MySQL: Creazione di tabelle

Sintassi per specificare una **colonna della tabella**.

Nome_colonna **TIPO**
[NOT NULL | NULL] [DEFAULT valore]
[AUTO_INCREMENT]
[UNIQUE | PRIMARY KEY]
[COMMENT 'commento']



Esaminiamo alcuni tipi di dato supportati da MySQL ...

MySQL: Creazione di tabelle

- Tipi di **dato numerici** supportati da MySQL:

BIT

TINYINT [UNSIGNED][ZEROFILL]

SMALLINT [UNSIGNED][ZEROFILL]

MEDIUMINT [UNSIGNED][ZEROFILL]

INT [UNSIGNED][ZEROFILL]

BIGINT [UNSIGNED][ZEROFILL]

FLOAT [UNSIGNED][ZEROFILL]

DOUBLE [UNSIGNED][ZEROFILL]

DECIMAL [UNSIGNED][ZEROFILL]

MySQL: Creazione di tabelle

- Tipi di **dato temporali** supportati da MySQL:

DATE

DATETIME

TIMESTAMP [M]

TIME

YEAR [(2,4)]

- Per conoscere **data/timestamp correnti**:

SELECT NOW();

SELECT CURTIME();

MySQL: Creazione di tabelle

- (Alcuni) Tipi di dato **stringa di caratteri o byte:**

CHAR(M) [BINARY | ASCII | UNICODE]

VARCHAR(M) [BINARY]

BINARY(M)

VARBINARY(M)

TINYBLOB

TINYTEXT

BLOB(M)

TEXT(M)

LONGBLOB

MySQL: Creazione di tabelle

Esempio di creazione di una tabella in **MYSQL**

```
CREATE TABLE IMPIEGATI (  
  Codice smallint auto_increment primary key,  
  Nome varchar(200) not null,  
  Cognome varchar(100) not null,  
  Salario double default 1000,  
  Anno date)  
engine=innodb;
```

MySQL: Comandi CRUD

- **Popolamento di dati** attraverso l'**INSERT** :

```
INSERT      [LOW_PRIORITY|DELAY|HIGH_PRIORITY]
[INTO] nome_tabella [(nome_colonne,...)]
VALUES ({espressione | DEFAULT}, ...)
[ON DUPLICATE KEY
  UPDATE nome_colonna=espressione, ...]
```

- E' possibile specificare una **priorità** dell'inserimento dei dati, nel caso in cui la tabella sia usata da altri processi.

MySQL: Comandi CRUD

- **Popolamento di dati** attraverso la **REPLACE**:


```
REPLACE [LOW_PRIORITY | DELAYED]  
[INTO] nome_tabella [(nome_colonna, ...)]  
VALUES ({espressione | DEFAULT}, ...)
```

- **Estensione** (MySQL) del costrutto di INSERT.
- Consente di **rimpiazzare** delle righe persistenti con delle nuove righe, qualora si verifichi un problema di inserimento con chiave doppia.

MySQL: Comandi CRUD

- **Popolamento di dati** attraverso la **LOAD**:

```
LOAD DATA [LOCAL] INFILE 'file.txt'  
[REPLACE | IGNORE]  
INTO TABLE nome_tabella  
[FIELDS  
  [TERMINATED BY 'stringa']  
  [ENCLOSED BY 'stringa']  
  [ESCAPED BY 'stringa'] ]  
[LINES  
  [STARTING BY 'stringa']  
  [TERMINATED BY 'stringa']]  
[IGNORE numero LINES]
```



Popolo la tabella
a partire dai dati presenti
in “file.txt”, specificando
i separatori delle colonne
ed eventualmente le righe
da filtrare ...

MySQL: Comandi CRUD

- Ricerca di dati attraverso il comando di **SELECT**:

```
SELECT [ALL | DISTINCT | DISTINCTROW]
lista_colonne
[INTO OUTFILE 'nome_file' |
 INTO DUMPFILE 'nome_file' ]
FROM lista_tabelle
[WHERE condizione]
[GROUP BY {nome_colonna}]
[HAVING condizione]
[ORDER BY {nome_colonna}]
[LIMIT [offset,] numero_righe]
```


MySQL: Comandi CRUD

- **Cancellazione di dati** attraverso il comando **DELETE**:

```
DELETE [LOW_PRIORITY][IGNORE][QUICK]  
FROM nome_tabella  
[WHERE condizione]  
[LIMIT numero_righe]
```

- **Rimuovere tutto il contenuto** attraverso il comando **TRUNCATE**:

```
TRUNCATE nome_tabella
```

- **Aggiornamento di dati** attraverso il comando di **UPDATE**:

```
UPDATE [LOW_PRIORITY][IGNORE]  
SET {nome_colonna=espressione, ...}  
WHERE condizione
```

MySQL: Creazione di TRIGGER

- **Creazione** di regole attive attraverso il costrutto di **TRIGGER**

```
CREATE TRIGGER nome tipo  
ON tabella FOR EACH ROW istruzioniSQL
```

- Il **tipo** specifica l'evento che attiva il trigger:

```
BEFORE INSERT  
BEFORE UPDATE  
BEFORE DELETE  
AFTER INSERT  
AFTER UPDATE  
AFTER DELETE
```

MySQL: Creazione di TRIGGER

Esempio di definizione di trigger in MySQL

```
CREATE TRIGGER upd_check
BEFORE INSERT ON Impiegati
FOR EACH ROW
BEGIN
    IF NEW.Salario > 300 THEN
        SET NEW.Salario=300;
    END IF;
END;
```

MySQL: Creazione di VISTE

- Creazione di viste attraverso il comando **VIEW**

```
CREATE [OR REPLACE]
[ALGORITHM = (UNDEFINED | MERGE | TEMPTABLE)]
VIEW nome [(lista colonne)]
AS selectSQL
[WITH [CASCADED|LOCAL] CHECK OPTION]
```

- E' possibile definire **viste aggiornabili** (attraverso la clausola **WITH CHECK OPTION**).

MySQL: Creazione di STORED PROC

- **Creazione** di **stored procedures** in MySQL:

```
CREATE PROCEDURE nomeProcedura  
([IN|OUT] nomeParametro tipo)  
BEGIN  
[dichiarazione di variabili locali]  
[istruzioni SQL]  
END;
```

- Insieme di istruzioni SQL memorizzate nel DBMS, cui e' associato un **nome**
- Puo' ricevere **parametri in input**, puo' restituire piu' di un **valore in output**. Il corpo contiene istruzioni SQL

MySQL: Creazione di STORED PROC

Esempio di definizione di stored procedure in MYSQL

```
CREATE PROCEDURE nomeImpiegato  
(IN cod INT, OUT nomeI VARCHAR(200))  
BEGIN  
  SELECT NOME AS NOMEI  
  FROM IMPIEGATI  
  WHERE (CODICE=cod);  
END;
```

```
CALL nomeImpiegato(200,@var);  
SELECT @var;
```

MySQL: Creazione di STORED PROC

- Dichiarazione di **variabili locali**:

```
DECLARE a INT DEFAULT 0;
```

- Costrutti di **selezione** (IF THEN ELSEIF ELSE):

```
IF Condizione THEN  
    IstruzioniSQL  
[ELSE IstruzioniSQL]  
ENDIF;
```

- Costrutti **iterativi** (WHILE/LOOP/REPEAT):

```
[nome] WHILE Condizione DO  
    IstruzioniSQL  
END WHILE [nome];
```

MySQL: Creazione di STORED PROC

- Dichiarazione di **cursori di query SQL**:

```
DECLARE nomeCursore CURSOR FOR selectSQL;  
OPEN nomeCursore  
FETCH nomeCursore INTO nomeVariabili;  
CLOSE nomeCursore
```

- I cursori consentono di eseguire query SQL e salvare il risultato (**result set**) in una lista.
- La lista risultante puo' essere iteratamente **visitata** attraverso il comando di FETCH.

MySQL: Creazione di STORED PROC

Esempio di di stored procedure con cursori in MYSQL (1/2)

```
CREATE PROCEDURE nomeImpiegato  
(IN salarioMax INT, OUT valido BIT)  
BEGIN  
    DECLARE fine INT DEFAULT 0;  
    DECLARE cur CURSOR FOR  
    SELECT salario FROM IMPIEGATI;  
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET  
fine=1;
```

MySQL: Creazione di STORED PROC

Esempio di di stored procedure con cursori in MYSQL (2/2)

```
SET valido=1;
OPEN cur;
ciclo: WHILE NOT fine DO
    FETCH cur INTO salarioCor;
    IF salarioCor > salarioMax THEN
        valido=0;
    END IF;
END WHILE ciclo;
END;
```

MySQL: Transazioni

- Gestione delle **transazioni** per tabelle **INNODB**:
 - Di default, la modalità **autocommit** è abilitata, quindi tutti gli aggiornamenti sono effettuati immediatamente sul database.
 - Nel caso in cui gli autocommit siano disabilitati, è necessario indicare l'inizio della transazione (**START TRANSACTION**) e terminarla con un comando di **COMMIT** o **ROLLBACK**.

MySQL: Transazioni

Esempio di transazione in MYSQL

```
SET AUTOCOMMIT = 0;  
START TRANSACTION  
INSERT INTO IMPIEGATO (Nome, Cognome, Salario)  
VALUES ('Michele','Rossi',1200);  
INSERT INTO IMPIEGATO (Nome, Cognome, Salario)  
VALUES ('Carlo','Bianchi',1000);  
COMMIT
```

MySQL: Transazioni

MySQL offre quattro livelli di **isolamento**:

- **READ UNCOMMITTED** → sono visibili gli aggiornamenti non consolidati fatti da altri.
- **READ COMMITTED** → aggiornamenti visibili solo se consolidati (ossia solo dopo COMMIT).
- **REPEATABLE READ** → tutte le letture di un dato operate da una transazione leggono sempre lo stesso valore (comportamento di **default**).
- **SERIALIZABLE** → lettura di un dato blocca gli aggiornamenti fino al termine della transazione stessa che ha letto il dato (lock applicato ad ogni SELECT).

MySQL: Transazioni

MySQL offre quattro livelli di **isolamento**:

- **READ UNCOMMITTED** → sono visibili gli aggiornamenti non consolidati fatti da altri.
- **READ COMMITTED** → sono visibili solo gli aggiornamenti consolidati (ossia quelli che hanno superato il commit).
- **REPEATABLE READ** → da una transazione leggiamo sempre lo stesso valore (comportamento di default).
- **SERIALIZABLE** → lettura di un dato blocca gli aggiornamenti fino al termine della transazione stessa che ha letto il dato (lock applicato ad ogni SELECT).

```
SET [GLOBAL|SESSION] TRANSACTION ISOLATION LEVEL  
{ READ UNCOMMITTED |  
  READ COMMITTED |  
  REPEATABLE READ |  
  SERIALIZABLE }
```

MySQL: Utilities

- Il tool `mysqldump` consente di effettuare **backup** del contenuto di un database (o di tutti).

Backup di tutti i database con tabelle INNODB

```
mysqldump --single-transaction --all-database > nomefile
```

Backup di uno specifico database con tabelle INNODB

```
mysqldump --single-transaction nomeadb > nomefile
```

Ripristino di un database (o tutti) da un file di backup

```
mysql [nomeadb] < nomefile
```

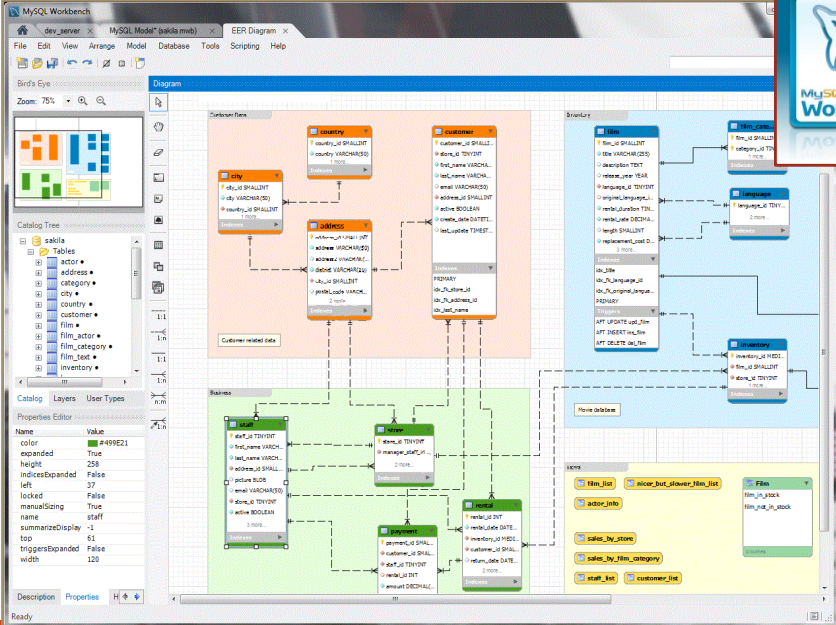
MySQL: Utilities

L'installazione di MySQL include molte **utilities**:

- `mysqladmin` → Amministrazione di MySQL
- `mysqlcheck` → Check di tabelle
- `mysqldump` → Dump di database su file
- `mysqlshow` → Mostra la struttura di tabelle
- `myisampack` → Comprime le tabelle su disco di tipo MyISAM (tabelle in sola lettura)
- ...

MySQL: Utilities

Il tool **MySQL Workbench** fornisce una GUI (Graphical User Interface) per l'utilizzo di MySQL.



The screenshot displays the MySQL Workbench interface. On the left, the 'Catalog Tree' shows a list of tables: actor, address, category, city, country, customer, film, film_actor, film_category, film_text, and inventory. The main workspace shows an Entity-Relationship (EER) diagram with entities like 'customer', 'address', 'film', 'inventory', and 'actor'. Relationships are indicated by dashed lines. A 'Properties Editor' on the left shows details for the 'actor' table, including columns like 'actor_id', 'first_name', 'last_name', 'email', 'phone', 'password', 'photo', 'bio', 'notes', and 'address_id'. A 'MySQL Workbench' logo is overlaid on the top right of the screenshot.

TOOL INTEGRATO:

- Amministrazione di database
- Sviluppo di SQL
- Modellazione e progettazione

BASI DI DATI

PROF. MARCO DI FELICE – CORSO DI LAUREA IN INFORMATICA PER IL MANAGEMENT

MySQL: Controfronto con altri RDBMS

- Principali **differenze** tra **MySQL** e **Oracle**

	MySQL	Oracle
Costi	Free (Community Ed.)	Pagamento (Enterprise)
Autenticazione & Sicurezza	Basata su host+username+password	Meccanismi multipli di autenticazione, ruoli
Gestione della Concorrenza	Lock a livello di tabella	Supporta lock a livello di singola riga
Supporto SQL e stored procedures	SQL base + estensioni procedurali (limitate)	SQL base + estensioni procedurali (PL/SQL)
Backup	Pochi tool di backup	Molti tool di backup
Supporto XML	Limitato	Supporto SQL/XML
Tipi di dati	Solo 2 tipi di dato char	4 Tipi di dato char