Basi Di Dati e di conoscenza

MySQL- SQL DDL e DML

Contenuti della lezione

- SQL e DBMS
- DBMS MySQL
- Installazione MySQL
- DDL
- DML
- Vincoli d'integrità

SQL

- originariamente "**Structured Query Language**", ora "nome proprio"
- linguaggio con varie funzionalità:
  - contiene sia il DDL sia il DML
- ne esistono varie versioni
- vediamo gli aspetti essenziali, non i dettagli

SQL: un po' di storia

- prima proposta SEQUEL (1974);
- prime implementazioni in SQL/DS e Oracle (1981)
- dal 1983 ca. "standard di fatto"
- standard (1986, poi 1989, 1992, 1999, 2003, 2006, 2008, 2011, 2016 …)
  - recepito solo in parte (!! Vedi http://troels.arvin.dk/db/rdbms/ per un confronto)

DBMS

Un DataBase Management System è un sistema di gestione il cui obiettivo generale è **mantenere le informazioni** (ovvero qualsiasi cosa sia ritenuta interessante da un individuo o da una organizzazione a cui il sistema è rivolto) e **renderle disponibili su richiesta**.

J. Date

DBMS

Un DBMS deve garantire:

- Condivisione dei dati
- Database persistenti
- Affidabilità
- Privatezza
- Efficienza
- Efficacia

Contenuti della lezione

- SQL e DBMS
- DBMS MySQL
- Installazione MySQL
- DDL
- DML
- Vincoli d'integrità

MySQL

- **MySQL**: sistema relazione di gestione per basi di dati (DBMS). E' un RDBMS open source (di Oracle), tra i più utilizzati. Incorpora funzioni non open nelle versioni enterprise (a pagamento).

- **SQL (Structured Query Language):** linguaggio per l'interrogazione dei dati

- **MySQL Workbench**: software per il disegno e modellazione del database

DBMS MySQL

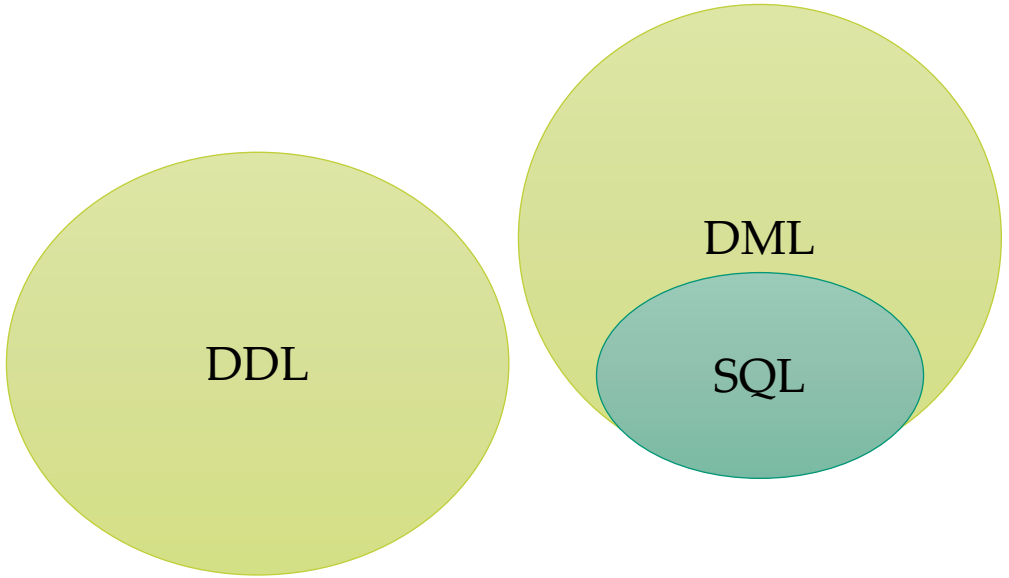
Caratteristiche di MySQL:

- Basato su routine ISAM, scritto in C e C++
- Engine preferibile: InnoDB (transazioni, fk, lock a livello di record)
- Interfacciabile da diversi linguaggi di programmazione (C, Java, Python, ecc.)
- Non ha limiti espliciti sulla dimensione max di un database e sul numero di tabelle (numero max righe dipende dai vincoli dell'os)
- Esistono diversi fork del progetto (MariaDB)

3 linguaggi:

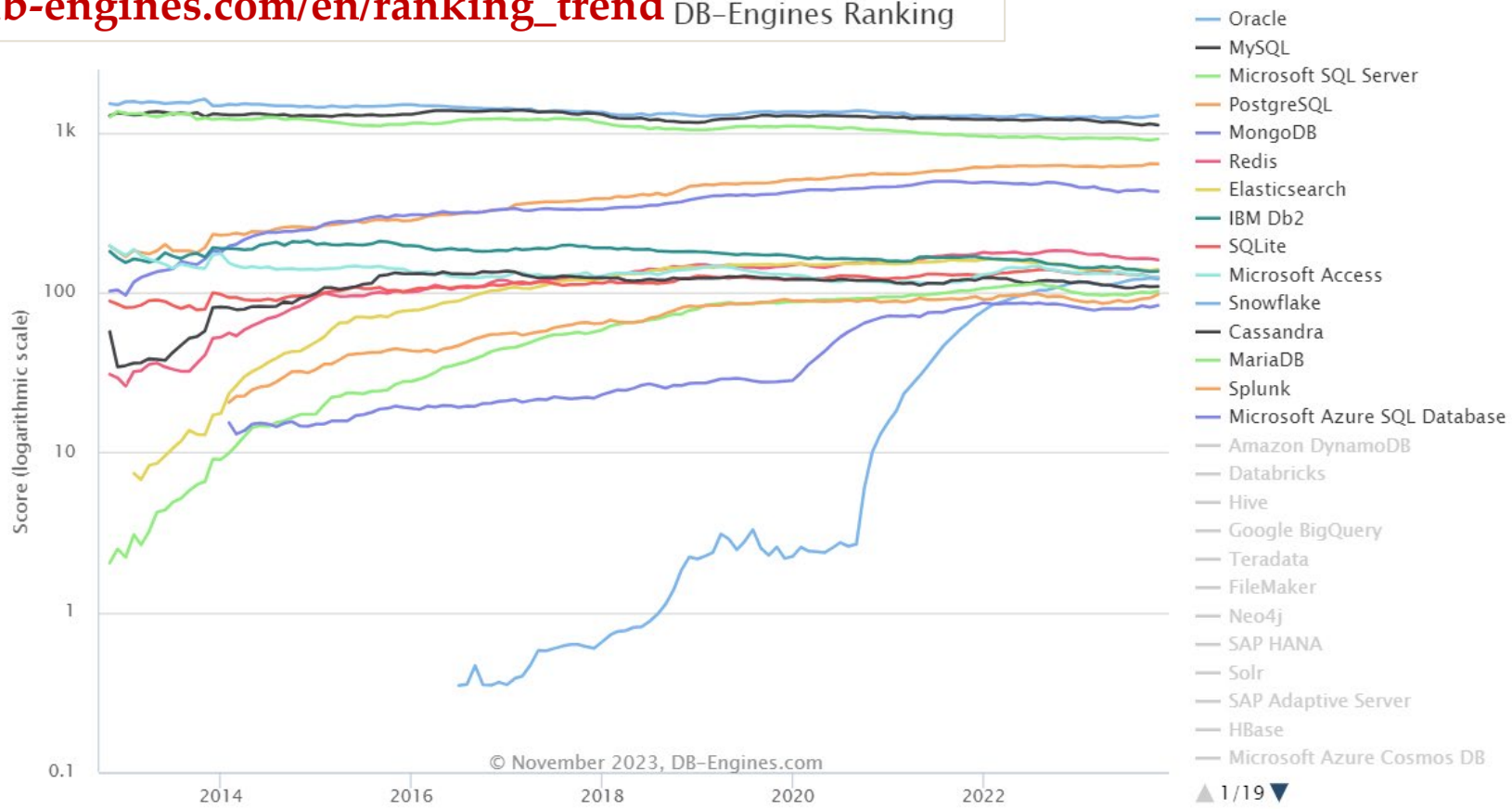
- DDL: data definition language
- DML: data manipulation language
- SQL: structured query language

SQL include funzionalità DDL e DML



DB-Engines Ranking - Trend Popularity

https://db-engines.com/en/ranking_trend DB-Engines Ranking



Contenuti della lezione

- SQL e DBMS
- DBMS MySQL
- Installazione MySQL 
- DDL
- DML
- Vincoli d'integrità

Installazione

- https://www.mysql.com/it/downloads/
- MySQL Community (GPL) Downloads »
- MySQL Installer for Windows

SQL

- L'interfaccia principale del tool è costituita da un terminale SQL a riga di comando dove i comandi sono le istruzioni SQL.

- Per connettersi (da terminale):

```
mysql –u utente –p password [-P porta –h host …]
```

Contenuti della lezione

- SQL e DBMS
- DBMS MySQL
- Installazione MySQL
- DDL
- DML- in MySQL

SQL- Istruzioni base

- `Show databases;` → **visualizza tutti i database**
- `Use <nomedb>;` → **usa il database con il nome specificato**
- `Create database <nomedb>;` → **crea un nuovo database con il nome specificato**
- `Drop database <nomedb>;` → **cancella database con il nome specificato**
- `Exit;` → **esci**

`N.B.: ;` chiude le istruzioni ed è necessario

Domini

- Domini elementari (predefiniti)
- Domini definiti dall'utente (semplici, ma riutilizzabili)

Domini elementari

- Carattere: singoli caratteri o stringhe, anche di lunghezza variabile
- Numerici, esatti e approssimati
- Data, ora, intervalli di tempo
- Introdotti in SQL:1999:
  - Boolean
  - BLOB, CLOB (binary/character large object): per grandi immagini e testi

SQL – Tipi di dato

Principali tipi di dato numerici:

- **`tinyint`**: da –128 a 127 unsigned 0-255
- **`smallint`**: da –32768 a 32767 unsigned 0-65535
- **`int`**: da –2.147.483.648 a 2.147.483.647 unsigned 0-4.294.967.295
- **`float(M,D):`** dove M numero cifre parte intera e D numero cifre parte decimale
- **`double(M,D):`** come float ma a doppia precisione

SQL – Tipi di dato

Principali tipi di dato **alfanumerici**:

- **`char(x)`:** Stringa di max 255 char di lunghezza fissa
- **`varchar(x)`:** Stringa di max 255 char di lunghezza variabile
- **`text`:** file 65535 byte
- **`blob`:** file o immagine (Binary Large Object)

non creare indici su text e blob

SQL – Tipi di dato

Principali tipi di dato **temporali**:

- **`datetime:`** aaaa-mm-gg hh:mm:ss
- **`date`**: aaaa-mm-gg
- **`time`**: hh:mm:ss
- **`year`**: aaaa
- **`timestamp(x):`** Variabile a seconda di x che varia da 2 a 14

Definizione di domini

- Istruzione **CREATE DOMAIN**:
  - definisce un dominio (semplice), utilizzabile in definizioni di relazioni, anche con vincoli e valori di default

```
CREATE DOMAIN Voto
  AS SMALLINT DEFAULT NULL
  CHECK ( value >=18 AND value <= 30 )
```

SQL-DLL

Sintassi **creazione e cancellazione** tabella:

```
create table <nome> (
     nome_attributo tipo opzioni,
     nome_attributo tipo opzioni,
     …) opzioni;

drop table <nome>;
```

- Una tabella è inizialmente vuota e chi la crea possiede tutti i **diritti su di essa**

SQL-Gestione tabelle

MySQL supporta diversi tipi di **storage engine** tra cui i principali sono:

- **InnoDB**
  - Supporta le transazioni
  - Supporta i vincoli di chiave esterna
  - Maggiore robustezza ai guasti
- **MyISAM**
  - Non supporta il sistema transazionale
  - Maggiore efficienza
  - Minore consumo di spazio in memoria secondaria

SQL-DLL

Per creare una tabella **InnoDB**

```
create table <nome> (
     nome_attributo tipo opzioni,
     nome_attributo tipo opzioni,
     …) opzioni engine=innodb;

drop table <nome>;
```

- Una tabella è inizialmente **vuota** e chi la crea possiede tutti i **diritti su di essa**

Create: esempio

```
Create table emp(
      empno int not null primary key
      auto_increment,
      ename varchar(30) not null,
      ……,
      sal float(4,2),
      ……);
Drop table emp;
```

SQL – DDL Gestione tabelle

- Si possono specificare **valori di default** che vengono assunti da un attributo quando, al momento della creazione di una nuova tupla, in un campo non viene inserito alcun valore

```
default < GenericoValore | null >
```

  - `GenericoValore` rappresenta un valore compatibile col dominio
  - `null` corrisponde al valore di default di base

  - Un nuovo default (es. nella definizione di un tipo a partire da un altro) sostituisce il precedente

Esempio `NumeroFigli smallint default 0`

impone il valore 0 quando non viene specificato il valore dell'attributo.

SQL – DDL Creazione tabella

Esempio creazione e cancellazione tabella :

```
create table emp (
      emp_id int not null primary key auto_increment,
      dept_id char(2) not null,
      emp_name varchar(30) not null,
      deptno int not null,
      sal float(4,2) not null,
      foreign key (deptno) references dept(dptno) on update cascade on
      delete no action
) engine=innodb;

drop table emp;
```

SQL – DDL Modifica schema

Sintassi altre operazioni:

- `alter table <nome> add nome_attributo tipo;` → modifica tabella esistente
- `alter table <nome> rename <nuovo_nome>;` → rinomina una tabella
- `create table if not exist <nome>…` → crea tabella se già non esiste

SQL – DDL Modifica schema

Sintassi modifica degli schemi

```
alter table NomeTabella <
      alter column NomeAttributo
       < set default NuovoDefault |
             drop default > |
              add constraint DefVincolo |
        drop constraint NomeVincolo |
        add column NomeAttributo |
              drop column NomeAttributo
          >
```

- **N.B. Quando si inserisce un nuovo vincolo, questo deve essere soddisfatto dai dati già presenti**

SQL – DDL Modifica schema

Sintassi modifica degli schemi

Il comando `drop` permette di rimuovere dei componenti

```
drop < schema | domain | table | view | assertion > NomeElemento
  [ restrict | cascade ]
```

`restrict` specifica di non eseguire il comando in presenza di oggetti non vuoti.

`cascade` implica che gli oggetti specificati vengano rimossi. Inoltre vengono rimossi tutti gli oggetti da essi dipendenti.
Quindi, **ATTENZIONE**!

Definizione degli indici

- è rilevante dal punto di vista delle prestazioni
- ma è a livello fisico e non logico
- in passato era importante perché in alcuni sistemi era l'unico mezzo per definire chiavi
- **`CREATE INDEX`**

SQL –Indici

Sintassi creazione e cancellazione indice:

```
create index <nome> on <nome_tabella>(lista nomi
colonne);
```

Esempio:

```
create index indnome on emp(ename);

drop index indnome on emp;
```

Contenuti della lezione

- SQL e DBMS
- DBMS MySQL
- Installazione MySQL
- DDL
- DML
- Vincoli d'integrità

Manipolazione dei dati (SQL-DML)

- I comandi SQL che permettono di modificare il contenuto di una base di dati sono:
  - **`insert`**
  - **`delete`**
  - **`update`**

insert

- **insert** ha la seguente sintassi

```
insert into NomeTabella [ListaAttributi]
< values (ListaDiValori) | SelectSQL >
```

- Nel primo caso si inserisce una singola riga all'interno della tabella. Nel secondo si possono aggiungere degli insiemi di righe, estratti dal contenuto della base di dati.

SQL – DML **insert**: esempi

```
insert into Dipartimento(NomeDip, Città)
                  values ('Produzione', 'Torino')
```

- Si utilizza in genere mediante una maschera (*form*) per consentire agli utenti di inserire dati. L'ordinamento degli attributi (se presente) e dei valori è significativo e le due liste devono avere lo stesso numero di elementi

```
insert into ProdottiMilanesi
     (select codice, descrizione
          from Prodotto
          where LuogoProd = 'Milano')
```

- Se in un inserimento non vengono inseriti tutti i dati di una riga si usa o il default (se esiste) o il valore nullo. La corrispondenza fra valori inseriti e attributi è per posizione

SQL – DML **insert**

Esempio:

```
insert into persone VALUES ('mario',25,52);

insert into persone(nome, eta, reddito)
       values('pino',25,52);

insert into persone(nome, reddito)
       values('lino',55);

insert into persone ( nome )
select padre
from paternita
where padre NOT IN (select nome
                    from persone);
```

SQL – DML **update**

Sintassi:

```
update <nome_tabella>
        set <nome attributo> = <valore>,
                <nome attributo> = <valore>,...
        where <condizioni>;
```

aggiorna uno o più attributi delle righe di **nome_tabella** che soddisfano l'eventuale Condizione. Se non c'è condizione la modifica avviene per tutte le righe.

Il nuovo valore può essere:

- il risultato di un'espressione valutata sugli attributi della tabella
- il risultato di una generica interrogazione SQL
- il valore nullo
- il valore di default

SQL – DML **update**: esempio

```
update emp
      set job = "salesman",
            sal=1,1* sal
      where ename ="Wilson";
```

aggiorna una sola riga

```
update emp
      set sal = (select 2*avg(sal) from emp where job='salesman')
      where job='salesman';
```

aggiorna un insieme di righe

Nella valutazione delle espressioni bisogna ricordarsi che SQL ha una natura orientata agli insiemi e non alle tuple. Quindi non è possibile pensare all'esecuzione di un comando come esecuzione riga per riga, ma come un'operazione effettuata contemporaneamente su tutto un insieme

SQL – DML **update**: esempio

Vogliamo aumentare del 10% gli stipendi degli impiegati con stipendio inferiore ai 30 milioni e del 15% quello degli impiegati con stipendio superiore ai 30 milioni.

Se scriviamo

```
update Impiegato set Stipendio = Stipendio * 1.1
where Stipendio <= 30
update Impiegato set Stipendio = Stipendio * 1.15
where Stipendio > 30
```

ad alcuni impiegati lo stipendio viene aumentato 2 volte!!!
Se invertiamo l'ordine la procedura risulta corretta.
Ma non sempre è così facile……..

SQL – DML **delete**

Sintassi:

```
delete from <nome tabella>
where <condizioni>;
```

Esempi:

```
delete from emp
where ename ="Wilson";

delete from emp
where job in (select …);
```

`delete from Dipartimento;` distrugge *il contenuto della tabella*

- se **`where`** non è specificato tutte le righe della tabella vengono eliminate. Se esiste un vincolo di integrità referenziale con politica di cascade … ATTENZIONE!
- Possibile usare interrogazioni nidificate

Contenuti della lezione

- SQL e DBMS
- DBMS MySQL
- Installazione MySQL
- DDL
- DML
- Vincoli d'integrità

SQL – DDL Vincoli

- I più semplici **vincoli intrarelazionali** predefiniti sono
- **`not null`** indica che il valore nullo non è ammesso su uno specifico attributo. Quindi richiede che sia inserito un valore, salvo che non sia già definito un valore di default. SQL non distingue i diversi tipi di valore nullo.
- **`unique (Attributo {, Attributo})`** indica che l'insieme di attributi deve essere una superchiave per la tabella.
- **`primary key (Attributo {, Attributo})`** definisce **la chiave primaria**.
  - Deve essere essere una sola.
  - Tutti gli attributi sono **`not null.`**

  .

SQL – DDL Vincoli

Esempio

```
Nome  char(20) not null,
Cognome char(20) not null,
unique (Nome, Cognome)
```

- impone che non ci sia una riga con sia `Nome` che `Cognome` uguali

Esempio

```
Nome  char(20) not null unique,
Cognome char(20) not null unique
```

- impone che sia `Nome` che `Cognome` siano diversi in tutte le righe

SQL – DDL Vincoli

Vincoli di integrità referenziale

- In SQL si usa il vincolo di **foreign key** (**chiave esterna**) per creare un legame fra i valori di un attributo della tabella **corrente** (**interna**) e un attributo di un'altra **tabella** (**esterna**).
- **Impone che per ogni riga della tabella interna il valore dell'attributo sia presente nel corrispondente attributo della tabella esterna.**

- L'attributo della tabella esterna deve essere **unique**. Si usa la keyword **`foreign key`**.

SQL – DDL Vincoli

Vincoli di integrità referenziale - Sintassi

```
on < delete | update >
     < cascade | set null | set default | no action >
```

- subito dopo la specifica del riferimento

SQL – DDL Vincoli

Esempio

```
create table Impiegato
(
      Matricola  char(6) primary key,
      Nome   char(20) not null,
      Cognome      char(20) not null,
      Stipendio  int default 0,
      unique (Cognome, Nome),
      foreign key(Nome, Cognome)
                  references Anagrafica(Nome, Cognome)
) engine=innodb;
```

SQL – DDL Vincoli

Vincoli di integrità referenziale

- Per i vincoli intra-relazionali, l'inserimento di un valore che li viola viene semplicemente impedito.
- Per i vincoli d'integrità referenziale, quando la violazione avviene per un cambiamento apportato alla tabella esterna, si hanno diverse possibili reazioni…

SQL – DDL Vincoli

Vincoli di integrità referenziale Modifica (comando update):

- **`cascade`** il nuovo valore dell'attributo della tabella esterna viene riportato su tutte le corrispondenti righe della tabella interna.
- **`set null`** all'attributo referente viene assegnato il valore nullo.
- **`set default`** all'attributo referente viene assegnato il valore di default.
- **`no action`** la modifica non viene consentita.

SQL – DDL Vincoli

Vincoli di integrità referenziale - Cancellazione (comando delete):

- **`cascade`** tutte le corrispondenti righe della tabella interna vengono cancellate.
- **`set null`** all'attributo referente viene assegnato il valore nullo.
- **`set default`** all'attributo referente viene assegnato il valore di default.
- **`no action`** la cancellazione non viene consentita.

Vincoli di integrità generici

Con i costrutti visti sinora, non è sempre possibile definire tutti i possibili vincoli di integrità.

Per questo esiste l'istruzione

check (*Condizione*)

La condizione specificata deve essere verificata da tutte le tuple in ogni momento.

E' possibile con check definire tutti i vincoli predefiniti.

Però è meno leggibile e non si possono applicare le politiche di reazione alle violazioni. Tuttavia è molto potente

```
Superiore character(6),
check (Matricola like "1%" or
        Dipart = (select Dipart
                  from Impiegato I
                  where I.Matricola=Superiore) )
```

check: esempio

```
create table Impiegato
(
Matricola character(6),
Cognome character(20),
Nome character(20),
Sesso character not null
                    check (sesso in ('M','F'))
Stipendio integer,
Superiore character(6),
check(Stipendio <= (select Stipendio
                    from Impiegato J
                    where Superiore = J.Matricola)
)
```

Asserzioni

Le asserzioni sono vincoli che fanno parte dello schema e non sono associati ad alcun attributo o tabella.

Permettono di definire tutti i vincoli utilizzabili nella definizione di una tabella, ma anche vincoli su più tabelle o vincoli che richiedono che una tabella abbia certe caratteristiche (es. una certa cardinalità).

Sintassi

`create assertion` *NomeAsserzione* **`check`**(*Condizione*)

Esempio

```
create assertion AlmenoUnImpiegato
check (1 <= (select count(*)
             from Impiegato))
```

Asserzioni

- Ogni vincolo di integrità è associato ad una politica di controllo che specifica se è *immediato* o *differito*.
- Se è immediato è verificato immediatamente dopo una modifica.
- Se è differito solo al termine dell'esecuzione di una serie di operazioni (*transazione*).
- Se un vincolo immediato viene violato, l'operazione di modifica può essere annullata immediatamente (*rollback parziale*). Tutti i vincoli predefiniti sono immediati.
- Se invece è differito al momento in cui si verifica la violazione non è più possibile identificare l'operazione che l'ha causata e quindi va' annulllata tutta la transazione (*rollback*).
- Questo garantisce la consistenza della base di dati.
- Per cambiare il tipo di controllo:

```
set constraints [NomeVincolo] immediate
set constraints [NomeVincolo] deferred
```