SQL

- DML
- TCL
- DDL
- Indici
- View
- Progetto di riferimento
  - https://github.com/egalli64/mpjp mySql *(modulo 4)*

INSERT

INSERT INTO table (columns…) VALUES (values…);

```
insert into regions(region_id, region_name)
values (11, 'Antarctica');
```

- I valori NULLABLE, se NULL, sono impliciti

```
insert into regions(region_id) values (12);
```

- Il nome delle colonne è opzionale (cfr. DESCRIBE)

```
insert into regions values (13, null);
```

UPDATE (WHERE!)

UPDATE table

SET column = value [, column2 = value2 ...]

[WHERE condition];

```
update regions
set region_name = concat('Region ', region_id)
where region_id > 10;
```

DELETE (WHERE!)

DELETE FROM table [WHERE condition];

```
delete from regions
where region_id > 10;
```

Transazioni

- Inizio: prima istruzione DML (INSERT, UPDATE, DELETE) in assoluto, o dopo la chiusura di una precedente transazione
- Fine: COMMIT, ROLLBACK, istruzione DDL, DCL, EXIT (implicano COMMIT o ROLLBACK in caso di failure)
- Buona norma: COMMIT o ROLLBACK esplicite
  - Eclipse Database Development: Window, Preferences, Data Management, SQL Development, SQL Editor, SQL Files / Scrapbooks, Connection Commit Mode → Manual
  - MySQL Workbench Query → Auto-Commit Transactions

COMMIT, ROLLBACK, SAVEPOINT

SAVEPOINT: punto intermedio in una transazione

```
insert into regions(region_id, region_name) values (11, 'Antarctica');
savepoint sp;

insert into regions(region_id, region_name) values (12, 'Oceania');

rollback to sp; -- keep Antarctica, rollback Oceania

commit; -- persist Antarctica
```

Livelli di isolamento nelle transazioni

- Transazioni concorrenti possono causare problemi in lettura:
  - **Phantom read**: T1 SELECT su più righe; T2 INSERT o DELETE nello stesso intervallo; T1 riesegue la stessa SELECT, nota un fantasma (apparso o scomparso) nel risultato
  - **Non repeatable read**: T1 SELECT, T2 UPDATE, T1 SELECT non ripetibile
  - **Lost update**: T1 UPDATE, T2 UPDATE. Il primo update è perso
  - **Dirty read**: T1 UPDATE, T2 SELECT, T1 ROLLBACK, valore per T2 è invalido
- Garanzie fornite da DBMS

  READ UNCOMMITTED: tutti comportamenti leciti

  READ COMMITTED: impedisce solo dirty read

  REPEATEBLE READ: phantom read permesse ← default MySQL

  SERIALIZABLE: nessuno dei problemi indicati ← default SQL

CREATE TABLE (on ME)

- Nome tabella, nome e tipo colonne, constraint, …

```
create table items (
    item_id integer primary key,
    status char,
    name varchar(20),
    coder_id integer);
```

CREATE TABLE AS SELECT

- Se si hanno i privilegi in lettura su una tabella si possono copiare dati e tipo di ogni colonna

  (GRANT SELECT ON … TO …)

```
create table coders
as
   select employee_id as coder_id, first_name, last_name, hire_date, salary
   from employees
   where department_id = 60;
```

ALTER TABLE

- ADD / DROP COLUMN

  alter table items add counter decimal(38, 0);

  alter table items drop column counter;

- ADD CONSTRAINT CHECK / UNIQUE

  alter table items add constraint items_status_ck check(status in ('A', 'B', 'X'));

  alter table coders add constraint coders_name_uq unique(first_name, last_name);

- ADD CONSTRAINT PRIMARY KEY / senza o con AUTO_INCREMENT

  alter table coders add constraint primary key(coder_id);

  alter table coders modify coder_id int primary key auto_increment;

CREATE TABLE con CONSTRAINT

```
create table details (
    detail_id integer primary key
        constraint detail_id_ck check (mod(detail_id, 2) = 1),
    status char default 'A'
       constraint detail_status_ck check (status in ('A', 'B', 'X')),
    -- alternativa: status enum('A', 'B', 'X') default 'A'
    name varchar(20),
        -- not null,
        -- unique,
    coder_id integer,

    constraint details_coder_fk foreign key(coder_id) references coders(coder_id), -- on delete cascade / set null
    constraint details_name_status_uq unique(name, status)
);
```

TRUNCATE / DROP TABLE

MySQL Workbench ha “safe mode” che limita le funzionalità standard
(Edit → Preferences → SQL Editor → Safe Updates)

- delete from table_name; -- DML → rollback
- truncate table table_name; -- no rollback!

- drop table table_name; -- no rollback!

- Negli script che si pensa possano essere eseguiti più volte è spesso utile fare un check sull’esistenza della tabella prima della sua eliminazione
  - drop table if exists table_name;

INDEX

- Possono velocizzare l’accesso alle tabelle, riducendo gli accessi alla memoria di massa
- B-Tree by default

```
-- indice semplice
create index coders_last_name_ix on coders(last_name);
-- indice composto
create index coders_name_ix on coders(first_name, last_name);

drop index coders_last_name_ix on coders;
```

VIEW

- Query predefinita su una o più tabelle, acceduta come se fosse una tabella
- Semplifica e controlla l'accesso ai dati

```
create or replace view odd_coders_view as
select * from coders
where mod(coder_id, 2) = 1;

drop view odd_coders_view;
```

Esercizi

- Coders
  - Inserire come assunti oggi:
    - 201, Maria Rossi, 5000€ e 202, Franco Bianchi, 4500€
  - Cambiare il nome da Maria a Mariangela
  - Aumentare di 500€ i salari minori di 6000€
  - Eliminare Franco Bianchi
  - Committare i cambiamenti