SQL (4)

- DML
- TCL
- DDL
- Indici
- View
- Progetto di riferimento
  - https://github.com/egalli64/mjd – folder oracle *(modulo 4)*

INSERT

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

```
insert into clients (client_id, name) values (20, 'Mordor');
```

Colonne con un default (NULL o altro) sono implicite

```
insert into clients (client_id) values (21);
```

Il nome delle colonne è opzionale, ma vanno indicati tutti i valori nel giusto ordine

```
insert into clients values (22, 'Rohan');
```

UPDATE (WHERE!)

UPDATE table
SET column = value [, column2 = value2 ...]
[WHERE condition];

```
update clients
set name = 'Client ' || client_id
where client_id > 10;

update coders
set first_name = 'Timmy', last_name = 'Eyes'
where coder_id = 108;
```

DELETE (WHERE!)

DELETE FROM table

[WHERE condition];

```
delete from clients
where client_id = 42;

delete from clients
where client_id > 10;
```

Per ridurre il rischio di DELETE/UPDATE inattese, converrebbe usare condizioni nel WHERE sulla PK (o eventuali altri campi che permettano una selezione univoca della riga)

Transazioni

- Vogliamo eseguire più comandi DML in un unico blocco ACID
  - **A**tomic **C**onsistent **I**solated **D**urable
- È possibile usare la modalità autocommit: ogni DML è in una propria transazione
  - SET AUTOCOMMIT ON/**OFF** per specificare il comportamento nella sessione corrente
- Molti tool di sviluppo hanno come default autocommit on
  - Eclipse Database Development: Window, Preferences, Data Management, SQL Development, SQL Editor, SQL Files / Scrapbooks, Connection Commit Mode → Manual
- SET TRANSACTION inizia esplicitamente una nuova transazione
  - Implicito alla prima istruzione DML o in seguito alla chiusura di una precedente transazione
- COMMIT, ROLLBACK terminano esplicitamente una transazione
  - implicitamente alla prima istruzione DDL, DCL, o alla terminazione della connessione

COMMIT, ROLLBACK, SAVEPOINT

SAVEPOINT: punto intermedio in una transazione

```
insert into clients (client_id, name) values (31, 'Mordor');

savepoint sp;

insert into clients (client_id, name) values (32, 'Rohan');

rollback to sp; -- keep Mordor, rollback Rohan

commit; -- persist Mordor
```

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 e T2 SELECT, 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 ← default Oracle DB

  SERIALIZABLE: nessuno dei problemi indicati ← default SQL
- Impostazione del comportamento per sessione o globale: SET TRANSACTION ISOLATION LEVEL
- È possibile usare “lock” per cambiare il comportamento di una transazione

CREATE TABLE

- 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 execs
as
   select employee_id as exec_id, first_name, last_name, hire_date, salary
   from hr.employees
   where department_id = 90;
```

ALTER TABLE

- ADD / DROP COLUMN

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

  alter table items drop column counter;

- ADD /DROP CONSTRAINT CHECK / UNIQUE

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

  alter table execs add constraint execs_name_uq unique(first_name, last_name);

  alter table items drop constraint items_status_ck;

- ADD / DROP CONSTRAINT PRIMARY KEY

  alter table execs modify exec_id number(6) primary key;

  alter table execs drop primary key;

CREATE TABLE con CONSTRAINT

```
create table details (
    detail_id integer,
    status char default 'A',
--      name varchar(20),
        name varchar(20) not null,
    exec_id integer,
    constraint details_pk primary key(detail_id),
    constraint detail_id_ck check (mod(detail_id, 2) = 1),
    constraint detail_status_ck check (status in ('A', 'B', 'X')),
    constraint details_coder_fk foreign key(exec_id) references execs(exec_id),
--    constraint details_coder_fk foreign key(exec_id) references execs(exec_id) on delete cascade,
--    constraint details_execs_fk foreign key(exec_id) references execs(exec_id) on delete set null,
    constraint details_name_status_uq unique(name, status)
);
```

TRUNCATE / DROP TABLE

Tre comandi SQL dal risultato simile ma con differenze sostanziali

- **delete from** table_name;
  - Elimina tutte le righe dalla tabella specificata
  - DML → rollback
- **truncate** table table_name;
  - Come sopra, ma è un comando DDL → no rollback
- **drop** table table_name;
  - Elimina l'intera tabella, struttura e contenuto
  - DDL → no rollback

INDEX

- Possono velocizzare l’accesso alle tabelle, riducendo gli accessi alla memoria di massa
- Andrebbero creati in un proprio tablespace, informazioni in USER_INDEXES
- B-Tree

  consigliato per colonne con valori unici, usato da Oracle per PK

  – create index execs_last_name_ix on execs(last_name); -- indice semplice

  -- indice composto

  create index execs_name_ix on execs(first_name, last_name);

  drop index execs_last_name_ix;
- Bitmap
  - più efficienti per colonne con pochi valori
  - create bitmap index execs_gender_ix on execs(gender);

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;
```

SEQUENCE

- Oggetto di database che genera una sequenza di interi

  create sequence my_seq; -- inizia da 1, incremento 1

- nextval: incrementa e ritorna il valore della sequenza

  select my_seq.nextval from dual;

- currval: ritorna il valore corrente, senza incremento

  select my_seq.currval from dual;

- Le sequenze possono essere modificate o eliminate

  alter sequence my_seq increment by 2;

  drop sequence my_seq;

SEQUENCE /2

- Sequenza con custom start e increase:

  create sequence my_seq start with 201 increment by 2;

- Altre proprietà definibili su di una sequenza:

  minvalue, maxvalue, cycle, order, etc.

- PK: si delega alla sequenza la generazione di valori univoci

  insert into execs

  values(my_seq.nextval, 'Bertrand', 'Meyer', SYSDATE, 20000);

- Info nella tabella USER_SEQUENCES

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