# SINTASSI DEI COMANDI SQL

- Creazione di domini

```
CREATE DOMAIN nome_dominio [AS] tipo [CHECK (condizione)];

es: CREATE DOMAIN CIFRA AS  CHAR(1)  CHECK (VALUE BETWEEN '0' AND '9');
    CREATE DOMAIN VOTO  AS  CHAR(13) CHECK (VALUE IN ('insuff','suff'));
    CREATE DOMAIN POS   AS  INTEGER  CHECK (VALUE > 0);
```

- Creazione di tabelle

```
CREATE TABLE nome_tabella (
        nome_colonna (tipo | dominio) [NOT NULL|[PRIMARY|FOREIGN] KEY]
        {, nome_colonna (tipo | dominio) [NOT NULL|[PRIMARY|FOREIGN] KEY]}
        {, vincolo} );

Vincoli:
PRIMARY KEY (lista_nomi_colonne)   dichiara la chiave primaria
UNIQUE (lista_nomi_colonne)        dichiara una o più chiavi candidate
FOREIGN KEY (lista_nomi_colonne)   dichiara una o più chiavi esterne
   REFERENCES nome_tabella         NB: lista_nomi_colonne deve essere la
                                       chiave primaria della tabella re-
                                       ferenziata
CHECK (condizione)                 dichiara un vincolo interno alla tabella

es: CREATE TABLE ARTICOLI (
        CodArt INTEGER NOT NULL,
        Prezzo INTEGER NOT NULL,
        PRIMARY KEY (CodArt),
        CHECK (Prezzo > 0)
    );
```

- Creazione di asserzioni

```
CREATE ASSERTION nome_vincolo CHECK (condizione);

es: CREATE ASSERTION ALMENO_UN_ORD CHECK
       NOT EXIST
          (SELECT *
           FROM CLIENTI AS C
           WHERE IDCl NOT IN
               (SELECT IDCl
                FROM ORDINI AS O
                WHERE C.IDCl=O.IDCl)
           );
```

- Modifica e cancellazione dello schema

```
ALTER TABLE nome_tabella ADD attr1 (tipo | dominio) [AFTER|BEFORE] attr2;
ALTER TABLE nome_tabella DROP nome_attr;
ALTER TABLE nome_tabella CHANGE nome_attr1 nome_attr2 TIPO(valore);
DROP TABLE nome_tabella;
DROP ASSERTION nome_asserzione;
DROP VIEW nome_vista;
DROP INDEX nome_indice;

es: ALTER TABLE CLIENTI ADD E_Mail CHAR(30); ALTER TABLE CLIENTI DROP E_Mail;
    ALTER TABLE DIPENDENTI CHANGE stipendio salario INTEGER(13);
    DROP TABLE CLIENTI;
    DROP ASSERTION ALMENO_UN_ORD;
    DROP VIEW nome_vista;
    DROP INDEX matricola_index;
```

- Creazione di viste

```
CREATE VIEW nome_vista AS query;

es: CREATE VIEW ART_ASSENTI AS
    SELECT CodArt,Desc
    FROM ARTICOLI
    WHERE Giacenza=0;
```

- Operatore Like

```
attr LIKE 'format_string'

NB: Nella format_string si usano dei caratteri speciali:
    %   indica una qualsiasi stringa
    _   indica un qualsiasi carattere

es: SELECT *
    FROM ARTICOLI
    WHERE Desc LIKE '%DOC___%1995%';
```

- Funzioni di aggregazione

```
OpAggreg ([DISTINCT] attr)

OpAggreg può essere:
   COUNT: numero degli elementi
   MIN: valore minimo
   MAX: valore massimo
   SUM: somma
   AVG: media aritmetica

NB: DISTINCT fa sì che l'operazione sia applicata alla colonna priva di
    eventuali duplicazioni.

es: Quanti tipi di articolo sono stati ordinati?
    SELECT COUNT (DISTINCT CodArt)
    FROM DETT_ORD;
```

- Istruzione SELECT

```
PARAMETERS attr [, attr]
SELECT [DISTINCT] (expr | attr [AS Alias] [, expr | attr [AS Alias]] | *)
FROM nome_tabella [AS Alias] [, nome_tabella [AS Alias]]
WHERE attr (=|<>|>|<|>=|<=|like) (attr|stringa|valore|[SOME|ALL|ANY]insieme)
      [(AND | OR) attr (EXIST | [NOT] IN) insieme]
GROUP BY attr [, attr]
HAVING cond_sul_gruppo [(AND | OR) cond_sul_gruppo]
ORDER BY attr [, attr]
[(UNION | INTERSECT | EXCEPT) [ALL] insieme2];

NB: Insieme è un altro SELECT che può anche far riferimento agli attributi
    Del SELECT soprastante (SELECT correlato). NOT IN → ≠ALL e IN → =ANY.

es: PARAMETERS MaxNum
    SELECT I.IDImp,SUM(Stipendio)/1000
    FROM IMPIEGATO AS I, REPARTO AS R
    WHERE I.IDImp=R.IDImp
    GROUP BY I.IDImp
    HAVING COUNT(*)>=MaxNum
    ORDER BY I.IDImp [ASC | DESC];
```

- I JOIN

```
SELECT [DISTINCT] (expr | attr [AS Alias] [, expr | attr [AS Alias]] | *)
FROM TAB1 (JOIN|NATURAL JOIN|INNER JOIN|[LEFT|RIGHT|FULL]OUTER JOIN) TAB2
      [ON TAB1.Attr(=|<>|>|<|>=|<=) TAB2.Attr];

Significato:
   JOIN: prodotto cartesiano
   NATURAL JOIN: tra 2 tabelle con un attributo uguale
   INNER JOIN: tra 2 tab con attributi da confrontare, da specificare in ON
   OUTER JOIN: come il NATURAL solo che preserva le righe con campi nulli
      LEFT     combina tutte le righe di tab1 con quelle di tab2
      RIGHT    combina tutte le righe di tab2 con quelle di tab1
      FULL     combina tutte le righe di tab1 con tutte quelle di tab2
```

- Comandi per la modifica dell'istanza di una tabella

```
1. INSERT INTO nome_tabella
   [(lista_attr)]
   VALUES (lista_valori) | insieme;

es: INSERT INTO DIPART
    (NomeDip,Città)
    VALUES ('Produzione','TO');

    INSERT INTO PROD_MI
    (SELECT Cod,Desc
     FROM PROD
     WHERE Luogo='MI');

2. UPDATE nome_tabella
   SET attr=(expr_scalare|valore) [,attr=(expr_scalare|valore)]
   WHERE cond;

es: UPDATE IMP
    SET Stip=Stip*1.1
    WHERE Stip < 30;

3. DELETE FROM nome_tabella
   WHERE cond;

es: DELETE FROM IMP
    WHERE Stip < 30;
```

- Creazione di TRIGGER

```
CREATE TRIGGER nome_trigger FOR nome_tabella
(BEFORE|AFTER)(INSERT|UPDATE|DELETE)
BEGIN
    Istruzioni_SQL
    [IF (new|old).attr (=|<>|>|<|>=|<=)(new|old).attr THEN
         RAISE EXCEPTION ...
     ENDIF]
    [Istruzioni_SQL]
END;

es:
CREATE TRIGGER CancellaOrdine FOR
ORDINI
BEFORE DELETE BEGIN
    DELETE DETT_ORD
    WHERE
DETT_ORD.NumOrd=ORDINI.NumOrd
END;

    CREATE TRIGGER Modifica_Anni FOR
    PERSONE
    AFTER UPDATE BEGIN
        IF NEW.Anni < OLD.Anni THEN
            RAISE EXCEPTION ...
        ENDIF
    END;
```

- Creazione di INDICI

```
CREATE [UNIQUE] INDEX nome_indice ON nome_tabella (lista_attr);

NB: UNIQUE è usato per le chiavi primarie.
```

- Assegnamento di PRIVILEGI

```
GRANT  lista_privilegi ON nome_tabella TO (lista_utenti|PUBLIC);
REVOKE lista_privilegi ON nome_tabella TO (lista_utenti|PUBLIC);
```

| *LISTA_PRIVILEGI* | *EFFETTI* |
|---|---|
| **SELECT** | Possibilità di lettura delle tuple |
| **INSERT**(lista_attrs) | Possibilità di usare INSERT su lista_attr |
| **UPDATE**(lista_attrs) | Possibilità di usare UPDATE su lista_attr |
| **DELETE** | Possibilità di cancellare delle tuple |
| **REFERENCES**(lista_attrs) | Possibilità di citare i campi lista_attrs in vincoli di integrità |

- Comandi MySQL

  - *Avvio del server*

```
mysqld --standalone --datadir=path\mysql\data --skip-innodb
```

  - *Chiusura del server*

```
mysqladmin shutdown
```

  - *Salvataggio di un db*

```
mysqldump nomedb > path\nomedb.sql
```

  - *Ripristino di un db*

```
mysql nomedb < path\nomedb.sql
```

  - *Comandi MySQL client*

```
SHOW databases;            → mostra i db disponibili
SHOW tables;               → mostra le tabelle disponibili
CREATE database nomedb;    → crea un db
USE nomedb;                → usa il db nomedb
DESCRIBE nometab;          → mostra lo schema della tabella
LOAD DATA LOCAL INFILE "path\\nomefile.ext" INTO TABLE nome_tab;
```

- Note

- La maggior parte dei comandi funziona con MySQL anche se TRIGGER, VISTE, o ASSERZIONI potrebbero non essere accettate dal suo dialetto.
- La clausola PARAMETERS del comando SELECT non è presente in alcuni dialetti sql (ma ad esempio è accettata da Access), essa permette di formulare la semplice richiesta di uno o più dati all'utente prima di eseguire la query.
- Ho preferito mantenere alcune istruzioni e/o clausole anche se non potrebbero funzionare con MySQL per permettere a chi ne ha l'occasione di poterle provare lo stesso con programmi differenti (Le incompatibilità sono dovute alla mancata implementazione di alcune funzioni nei vari programmi ed ai vari dialetti sql che essi possono adottare).
- La parola PATH è intesa come percorso di directory, ad es: 'programmi\bin' oppure 'x:\documenti\bd' dove 'x' è la lettera dell'unità.
- Nel caricamento dei dati da un file in una tabella il PATH utilizza come separatore di directory la doppia barra '\\'.