Basi di Dati
Modulo Laboratorio

Lezione 1: PostgreSQL

DR. SARA MIGLIORINI

A.A. 2024-2025

PostgreSQL

- È un Relational Data Base Management System (RDBMS o DBMS) con alcune funzionalità orientate agli oggetti.
- È multipiattaforma (Linux,Mac OS X, Windows, ecc.), di pubblico dominio, gestita da un gruppo di volontari.
- Il sito ufficiale è http://www.postgresql.org
- Implementa gran parte dello standard SQL:
  - la versione 16 implementa quasi tutte le funzioni di SQL:2023. Vedere https://www.postgresql.org/docs/16/features.html
  - https://www.postgresql.org/docs/16/unsupported-features-sql-standard.html indica le funzioni NON implementate di SQL:2023.

PostgreSQL

- In PostgreSQL l'integrità dei dati e l'affidabilità sono la priorità.
- Rappresenta un'alternativa open source ai DBMS di Oracle, Microsoft, etc.
- Un confronto tra PostgreSQL e Oracle è disponibile:
  - http://db-engines.com/en/system/Oracle%3BPostgreSQL
  - http://www.sql-workbench.net/dbms_comparison.html

PostgreSQL

- L'interazione tra un utente (programmatore/utente finale) e le basi di dati gestite da PostgreSQL avviene secondo il modello client-server.
- Server
  - Il daemon postmaster supervisiona tutti i processi specifici di PostgreSQL.
- Client(s)
  - Qualsiasi programma in grado di gestire l'iterazione con postmaster: input comandi, invio al postmaster, visualizzazione esito comando, ecc.
  - Il client standard è psql, un'applicazione a linea di comando.
  - Un'alternativa interessante è pgAdmin IV, http://www.pgadmin.org/, client grafico più intuitivo.

PostgreSQL – In laboratorio

- Per le esercitazioni in laboratorio si usa il server PostgreSQL dbserver.scienze.univr.it.
- Il server è un PostgreSQL 16.
- Ogni studente iscritto al III anno ha a disposizione una base di dati personale per poter svolgere le esercitazioni.
- La base di dati personale è accessibile usando le proprie credenziali GIA e deve essere attivata prima di poterla usare.
- Accedere al sito web http://db-srv.di.univr.it con le proprie credenziali GIA e seguire le istruzioni per attivare la propria base di dati (Accesso tramite VPN o rete universitaria).
- La base di dati personale è disponibile fino al 28/02/2026.

PostgreSQL – In laboratorio

Ci si connette al server tramite l'applicazione psql:

- `psql -h <server> -U <username> [-d] <database>`

Esempio – connessione tramite psql da parte dell'utente id05156

```
$ psql -h db-srv.di.univr.it -U id051563 id051563
Inserisci la password per l'utente id051563:
psql (16.0)
connessione SSL (protocollo: TLSv1.2, cifrario: ECDHE-RSA-AES256-GCM-SHA384,
bit: 256, compressione: disattivato)
Digita "help" per avere un aiuto.
id051563 =>
```

PostgreSQL - Applicativo psql

- **psql** è un interprete iterativo da terminale.
- **psql** accetta due tipi di comando: comandi di tipo Structured Query Language (SQL) o comandi interni.
  - Comandi SQL: permettono di creare/gestire e interrogare le basi di dati.
  - Comandi interni (riconoscibili perché iniziano con '\') permettono di configurare il client o avere aiuto per scrivere comandi SQL.

PostgreSQL – Applicativo psql

Esempio – comando help, unico che non inizia con '\'

```
id051563 => help
Stai utilizzando psql , l' interfaccia a riga di comando di PostgreSQL.
Digita: \ copyright per le condizioni di distribuzione
\h per la guida sui comandi SQL
\? per la guida sui comandi psql
\g o termina con punto e virgola per eseguire la query
\q per uscire
id051563 =>
```

PostgreSQL – Applicativo psql

Esempio – comando interno \l: elenca tutti le basi di dati

```
id051563 => \l
                  List of databases
Name       | Owner    | Encoding | Collate         | Ctype
-----------+----------+----------+-----------------+--------------
template0 | postgres | UTF8     | it_IT .UTF -8 | it_IT .UTF -8
Template1 | postgres | UTF8     | it_IT .UTF -8 | it_IT .UTF -8
id051563  | id051563 | UTF8     | it_IT .UTF -8 | it_IT .UTF -8

...

id051563 =>
```

PostgreSQL – Applicativo psql

Esempio – comando SQL CREATE

```
id051563 => CREATE TEMP TABLE inutile (id INTEGER , inutile VARCHAR );
CREATE TABLE
id051563 =>
```

Structured Query Language (SQL)

SQL

- SQL è il linguaggio di riferimento per le basi di dati relazionali
- Originariamente SQL era l'acronimo di "Structured Query Language" (ora "nome proprio")
- SQL è un linguaggio con varie funzionalità:
  - Linguaggio per la definizione delle strutture dati e dei vincoli di integrità: Data Definition Language (DDL)
  - Linguaggio per manipolare i dati: inserimento, aggiornamento e cancellazione: Data Manipulation Language (DML)
  - Linguaggio per interrogare: Data Query Language (DQL o QL)
- Ne esistono varie versioni

SQL

Figura 4.1
Evoluzione
dello standard SQL.

Nome informale	Nome ufficiale	Caratteristiche
SQL base	SQL-86	Costrutti base
	SQL-89	Integrità referenziale
SQL-2	SQL-92	Modello relazionale Vari costrutti nuovi 3 livelli: entry, intermediate, full
SQL-3	SQL:1999	Modello relazionale a oggetti Organizzato in diverse parti Trigger, funzioni esterne, ...
	SQL:2003	Estensioni del modello a oggetti Eliminazione di costrutti non usati Nuove parti: SQL/JRT, SQL/XML, ...
	SQL:2006	Estensione della parte XML
	SQL:2008	Lievi aggiunte (per esempio, trigger `instead of`)
	SQL:2011	Ricco supporto per dati temporali
	SQL:2016	Gestione del formato JSON

Cosa può fare SQL

- Definizione dello schema di una base di dati
- Modifica dello schema di una base di dati
- Inserimento/modifica/cancellazione dei dati da una base di dati
- Specificare interrogazioni semplici sulla base di dati
- Specificare interrogazioni complesse:
  - Interrogazioni annidate
  - Raggruppamenti
  - E molto altro…

SQL - Introduzione

In questo modulo considereremo solo una parte di SQL, sufficiente per creare e gestire semplici basi di dati:

- Costrutti fondamentali del Data Definition Language (DDL), come CREATE TABLE, ALTER TABLE,. . .
- Costrutti fondamentali del Data Manipulation Language (DML), come INSERT, UPDATE, DELETE,. . .
- Costrutti fondamentali del Data Query Language (DQL o QL), come SELECT, SELECT con JOIN, SELECT innestate,. . .

- Manuale PostgreSQL: http://www.postgresql.org/docs/16/interactive/

SQL – Comandi base
Creazione di una base di dati

Creazione di una tabella: CREATE TABLE

```
CREATE TABLE tabella (
      attributo dominio [ valoreDefault ] { vincolo }
      {, attributo dominio [ valoreDefault ] { vincolo } }
      {, vincoloTabella }
]);
```

- Notazione: [e] indicano parti opzionali (il termine può non comparire o comparire una sola volta); { e } indicano che il termine può non comparire oppure comparire un numero arbitrario di volte.
- L'istruzione CREATE TABLE definisce uno schema di relazione e ne crea un'istanza vuota: attributi, domini e vincoli.

Creazione di una tabella: ESEMPIO

```
CREATE TABLE Impiegato (
     Matricola CHARCTER(6) PRIMARY KEY,                  -- attributo
     Nome CHARCTER VARYING(20) NOT NULL,                 -- attributo
     Cognome CHARCTER VARYING(20) NOT NULL,              -- attributo
     Dipart CHARCTER VARYING(15),                        -- attributo
     Stipendio NUMERIC(9) DEFAULT 0,                     -- attributo
     FOREIGN KEY(Dipart)
              REFERENCES Dipartimento(NomeDip),          -- vincolo di tabella
     UNIQUE (Cognome,Nome)                               -- vincolo di tabella
 )
```

Gli Identificatori

- Già con l'istruzione CREATE TABLE si devono conoscere i concetti fondamentali del DDL: identificatori, domini (tipo) e vincoli.

- Identificatori SQL: stringhe che iniziano con una lettera (UTF-8) o un underscore (_) e che possono contenere anche cifre (0-9).

- Gli identificatori SQL non sono sensibili al minuscolo/maiuscolo:
  - idPersona e idpersona rappresentano un solo identificatore.

- Tutti i nomi di tabella, attributo, ecc. sono identificatori SQL.

Domini

- Domini elementari (predefiniti)
  - caratteri: singoli caratteri o stringhe anche di lunghezza variabile
  - bit: bit singoli (booleani) o stringhe di bit
  - numerici esatti e approssimati
  - istanti temporali
  - intervalli temporali

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

Domini elementari: Tipo carattere

- Permette di rappresentare singoli caratteri oppure stringhe.
- Un carattere o stringa di caratteri sono rappresentati tra apici (’):
  - ’a’, ’Questa è una stringa’
- La lunghezza delle stringhe può essere fissa o variabile:
  `CHARACTER [VARYING][(lunghezza)]`
  - CHARACTER: carattere singolo.
  - CHARACTER(20): stringa di lunghezza fissa. Se si assegna una stringa di lunghezza inferiore a 20, la stringa viene riempita di spazi fino a rendere la stringa lunga 20.
  - CHARACTER VARYING(20): stringa di lunghezza variabile (max 20).
  - TEXT: stringa di lunghezza variabile senza limite fissato. Questa è un’estensione PostgreSQL.

Domini elementari: Tipo Boolean (SQL-3)

- Permette di rappresentare i valori booleani TRUE/FALSE, rappresentati anche come t/f, 1/0, yes/NO, più lo stato unknown (valore nullo), rappresentato come NULL.

- `BOOLEAN`: valore booleano singolo.

Domini elementari: Tipo numerici esatti

- Permettono di rappresentare valori esatti: interi o con una parte decimale di lunghezza prefissata
- Valori interi:
  - `SMALLINT`: valori interi (2 bytes: da -32.768 (2^{15}) a +32.767 (2^{15}-1)).
  - `INTEGER`: valori interi (4 bytes: da -2147483648 (2^{31}) to +2147483647 (2^{31}-1)).
- Valori decimali:
  - `NUMERIC [(precisione [, scala])]`: precisione è il numero totale di cifre significative (cioè di cifre a sinistra e a destra della virgola), scala è il numero di cifre dopo la virgola (vuoto = 0).
  - `DECIMAL [(precisione [, scala])]`: equivalente al precedente.
  - Esempio: NUMERIC(5,2) permette di rappresentare valori come 100.01, 100.999 (arrotondato a 101.00), ma non valori come 1000.01!

Domini elementari: Tipo numerici approssimati

- Permettono di rappresentare valori in virgola mobile:
- `REAL`: una precisione di 6 cifre decimali.
- `DOUBLE PRECISION`: una precisione di 15 cifre decimali.
- Ciascun numero è rappresentato da una coppia di valori: la mantissa e l'esponente.
  - La mantissa è un numero frazionario, l'esponente è un numero intero
  - Il valore si ottiene moltiplicando la mantissa per la potenza di 10 con grado pari all'esponente
  - 0.17E16 = 1,7 x 1015 -0.4E-6 = -4 x 10-7
- Nota:
  - REAL e DOUBLE PRECISION sono comunque valori approssimati.
  - Se si devono rappresentare importi di denaro che contengono anche decimali, MAI usare questi tipi ma usare NUMERIC(precisione)!

Domini elementari: Istanti temporali

- Permettono di rappresentare istanti di tempo.
- `DATE`: istanti del tipo (anno, mese, giorno).
  - Un valore DATE si rappresenta tra apici (') e lo standard ISO prescrive il formato YYYY-MM-DD.
- `TIME [(precisione)][WITH TIME ZONE]`: istanti del tipo (hour, minute, second).
  - precisione: numero cifre decimali per rappresentare frazioni del secondo;
  - WITH TIME ZONE: permette di specificare anche [+-]hour:minute, che rappresentano la differenza con l'ora di Greenwich, o nomeFusoOrario.
- `TIMESTAMP [(precisione)][WITH TIME ZONE]`: date + time.

Domini elementari: Intervalli temporali

- Permettono di rappresentare intervalli di tempo, come la durata di un evento
  `INTERVAL` *`PrimaUnitàDiTempo`* `[TO` *`UltimaUnitàDiTempo`*`]`
  - *`PrimaUnitàDiTempo`* e *`UltimaUnitàDiTempo`* definiscono le unità di misura che devono essere utilizzata, della più precisa alla meno precisa;
  - INTERVAL year TO month: indica un intervallo di tempo misurato in numero di anni e di mesi
- L'insieme delle unità di misura è diviso in due insiemi distinti:
  - da **year** a **month**, e
  - da **day** a **second**
  - Poiché non è possibile paragonare esattamente giorni e mesi (un mese può avere da 28 a 31 giorni!)
- Si può specificare una precisione per la *PrimaUnitàDiTempo*: numero di cifre in base 10 per la rappresentazione
- Se l'*`UltimaUnitàDiTempo`* è **second**, si può specificare una precisione che rappresenta il numero di cifre decimali dopo la virgola
  - INTERVAL year(5) to month intervalli della durata fino a 99 999 anni a 11 mesi
  - INTERVAL day(4) to second(2) intervalli della durata fino a 9999 giorni, 23 ore, 59 minuti e 59,99 secondi

Domini definiti dall'utente

- Istruzione CREATE DOMAIN:
  - definisce un dominio, utilizzabile in definizioni di relazioni, anche con vincoli e valori di default
  - il dominio può essere definito a partire da un dominio elementare oppure da un altro dominio definito dall'utente
  - `CREATE DOMAIN nome AS tipoBase [valoreDefault][vincolo]`

  - ```
    CREATE DOMAIN giorniSettimana AS CHARACTER (3)
       CHECK (VALUE IN ('LUN', 'MAR', 'MER', 'GIO', 'VEN', 'SAB', 'DOM'));
    ```
    ```

Domini definiti dall'utente: Esempio

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

Nome dominio

Tipo di dato (dominio base)

Valore di default

Vincolo

Vincoli intrarelazionali

- `NOT NULL`: il valore nullo non è ammesso per l'attributo, quindi il valore deve sempre essere specificato, o si assegna un valore di default;
- `UNIQUE`: definisce (super)chiavi
- `PRIMARY KEY`: chiave primaria (una sola, implica not null)
- `CHECK`

Vincolo NOT NULL e DEFAULT

Il vincolo NOT NULL determina che il valore nullo non è ammesso come valore dell'attributo.

- Nel caso di vincoli NOT NULL può essere utile specificare un valore di default per l'attributo. L'istruzione DEFAULT valore specifica un valore di default per un attributo quando un comando di inserimento dati non specifica nessun valore per l'attributo.

Esempio

```
nome VARCHAR(20) NOT NULL
cognome VARCHAR(20) NOT NULL DEFAULT 'VUOTO'
```

Vincolo UNIQUE

Il vincolo UNIQUE impone che righe differenti della tabella non possono avere gli stessi valori: concetto di superchiave il quale però richiede anche il vincolo di NOT NULL!.

- Si può definire su:
  - un solo attributo;
  - un insieme di attributi.

UNIQUE su una coppia ≠ UNIQUE sui due attributi	

`nome VARCHAR(20),`
`cognome VARCHAR(20),`
`UNIQUE(nome, cognome)`
	`('n','c'), ('n', NULL), ('c', NULL),`
	`(NULL, NULL), (NULL, NULL).`

`nome VARCHAR(20) UNIQUE,` `cognome VARCHAR(20) UNIQUE`	`('n','c'), ('n', NULL), (NULL, 'c'),` `(NULL, NULL), (NULL, NULL).`

Vincolo PRIMARY KEY

Il vicolo PRIMARY KEY identifica l'attributo che rappresenta la chiave primaria della relazione:

- Si usa una sola volta per tabella.
- Implica il vincolo NOT NULL.
- Due forme di specifica:
  - nella definizione dell'attributo, se è l'unico componente della chiave primaria:

```
matricola CHAR(6) PRIMARY KEY;
```

  - come definizione separata a livello di tabella (vincolo di tabella), se invece la chiave primaria è composta di più attributi.

```
nome VARCHAR(20),
cognome VARCHAR(20),
PRIMARY KEY(nome , cognome )
```

Vincolo CHECK

Il vincolo CHECK specifica un vincolo generico che devono soddisfare le tuple della tabella.

- Un vincolo CHECK è soddisfatto se la sua espressione è vera o nulla.

Esempio

```
CREATE TABLE Impiegato (
  matricola CHAR(6) PRIMARY KEY,
  nome VARCHAR(20) NOT NULL,
  cognome VARCHAR(20) NOT NULL,
  qualifica VARCHAR(20),
  stipendio NUMERIC(8,2) DEFAULT 500.00 NOT NULL
  CHECK( stipendio >= 0.0 ), --check di attributo
  UNIQUE( cognome, nome ),
  CHECK( nome <> cognome ) --check di tabella
);
```

Vincoli interrelazionali (tra relazioni diverse)

- Un vincolo di integrità referenziale (= interrelazionale) crea un legame tra i valori di un attributo (o di un insieme di attributi) A della tabella corrente (detta interna/slave) e i valori di un attributo (o di un insieme di attributi) B di un'altra tabella (detta esterna/master):
- Impone che, in ogni tupla della tabella interna, il valore di A, se diverso dal valore nullo, sia presente tra i valori di B nella tabella esterna.
- L'attributo B della tabella esterna deve essere soggetto a un vincolo UNIQUE (o PRIMARY KEY).
  - Nota: L'attributo (o insieme di attributi) B può anche **non** essere la chiave primaria, ma deve essere identificante per le tuple della tabella esterna.

Vincoli interrelazionali (tra relazioni diverse)

- REFERENCES e FOREIGN KEY permettono di definire vincoli di integrità referenziale
- Due sintassi:
  - Per singoli attributi
  - Se è coinvolto un solo attributo, basta usare il costrutto sintattico REFERENCES nella dichiarazione dell'attributo, specificando la tabella esterna e il relativo attributo coinvolto.
  - Su più attributi
  - In alternativa, e necessariamente quando il legame coinvolge più attributi, si usa il costrutto FOREIGN KEY alla fine della definizione degli attributi, elencando gli attributi coinvolti e seguito dalla definizioni dei corrispondenti attributi della tabella esterna, sempre grazie a REFERENCES.

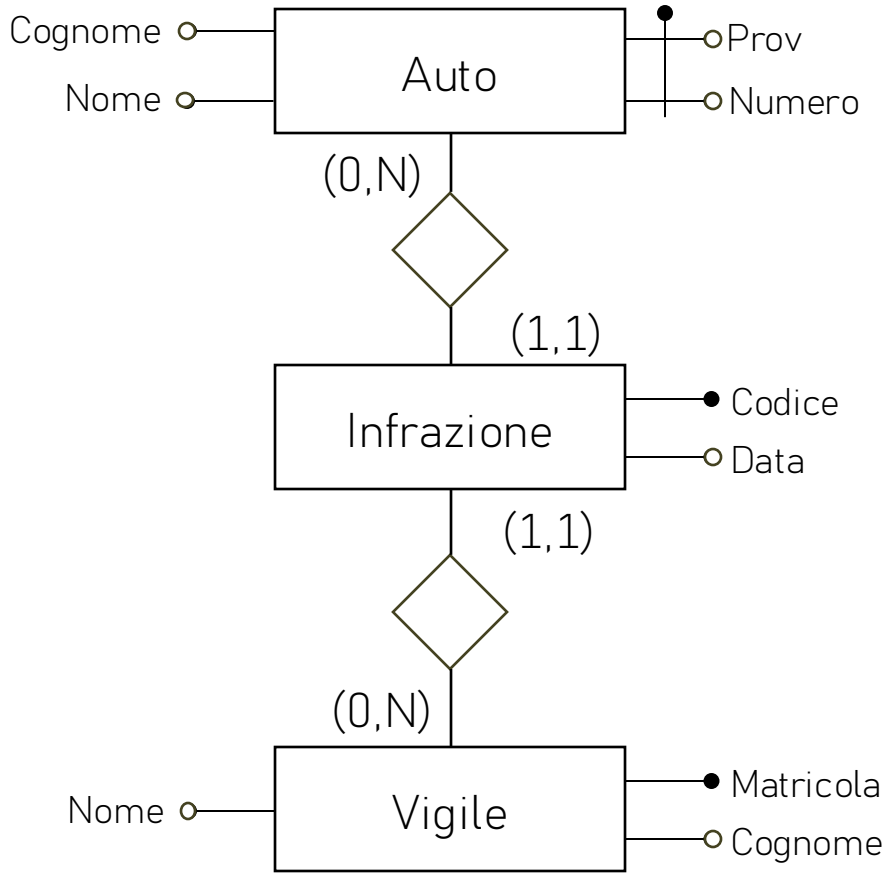
REFERENCES e FOREIGN KEY

```
CREATE TABLE Impiegato (
      Matricola   CHARACTER(6) primary key,
      Nome        CHARACTER VARYING(20)   not null,
      Cognome     CHARACTER VARYING(20)   not null,
      Dipart      CHARACTER VARYING(15)
                        REFERENCES Dipartimento(NomeDip),
      Ufficio     NUMERIC(3),
      Stipendio   NUMERIC(9) default 0,
      UNIQUE (Cognome, Nome)
)
```

REFERENCES e FOREIGN KEY

```
CREATE TABLE Impiegato (
       Matricola  CHARACTER(6) primary key,
       Nome       CHARACTER VARYING(20)  not null,
       Cognome    CHARACTER VARYING(20)  not null,
       Dipart     CHARACTER VARYING(15) REFERENCES Dipartimento(NomeDip),
       Ufficio    NUMERIC(3),
       Stipendio  NUMERIC(9) default 0,
       FOREIGN KEY (Nome, Cognome) REFERENCES Anagrafica (Nome, Cognome)
)
```

CREATE TABLE: Esempio



Infrazione (Codice, Data, Vigile, Prov, Numero)

Vigile (Matricola, Cognome, Nome)

Auto (Prov, Numero, Cognome, Nome)

CREATE TABLE: Esempio

```
CREATE TABLE Infrazione(
        Codice CHAR(6) PRIMARY KEY,
        Data DATE NOT NULL,
        Vigile INTEGER NOT NULL
                REFERENCES Vigile(Matricola),
        Provincia CHAR(2),
        Numero CHAR(6) ,
        FOREIGN KEY(Provincia, Numero)
                REFERENCES Auto(Provincia, Numero)
 )
```

Modifiche degli schemi

- SQL offre delle primitive per modificare schemi già introdotti, attraverso l'uso dei comandi

- ALTER: effettua modifiche (a domini, tabelle…), ad esempio posso aggiungere una colonna a una relazione,

- DROP (DOMAIN, TABLE): permette di rimuovere i vari componenti di domini e tabelle, ad esempio posso cancellare una tabella.

Modifica struttura tabella

La struttura di una tabella si può modificare dopo la sua creazione con il comando ALTER TABLE.
Di seguito, le modifiche più comuni:

- Aggiunta di un nuovo attributo (ADD COLUMN):
  - `ALTER TABLE tabella ADD COLUMN attributo tipo;`
  - `ALTER TABLE impiegato ADD COLUMN stipendio NUMERIC(8,2);`
- Rimozione di un attributo (DROP COLUMN):
  - `ALTER TABLE tabella DROP COLUMN attributo;`
- Modifica valore di default di un attributo (ALTER COLUMN):
  - `ALTER TABLE tabella ALTER COLUMN attributo { SET DEFAULT valore | DROP DEFAULT };`
  - `ALTER TABLE impiegato ALTER COLUMN stipendio SET DEFAULT 1000.00;`

Modifica struttura tabella

La struttura di una tabella si può modificare dopo la sua creazione con il comando ALTER TABLE.
Di seguito, le modifiche più comuni:

- Aggiunta di un nuovo attributo (ADD COLUMN):
  - `ALTER TABLE tabella ADD COLUMN attributo tipo;`
  - `ALTER TABLE impiegato ADD COLUMN stipendio NUMERIC(8,2);`
- Rimozione di un attributo (DROP COLUMN):
  - `ALTER TABLE tabella DROP COLUMN attributo;`
- Modifica valore di default di un attributo (ALTER COLUMN):
  - `ALTER TABLE tabella ALTER COLUMN attributo { SET DEFAULT valore | DROP DEFAULT };`
  - `ALTER TABLE impiegato ALTER COLUMN stipendio SET DEFAULT 1000.00;`

Modifica struttura tabella

La struttura di una tabella si può modificare dopo la sua creazione con il comando ALTER TABLE.
Di seguito, le modifiche più comuni:

- Aggiunta di un nuovo attributo (ADD COLUMN):
  - `ALTER TABLE tabella ADD COLUMN attributo tipo;`
  - `ALTER TABLE impiegato ADD COLUMN stipendio NUMERIC(8,2);`
- Rimozione di un attributo (DROP COLUMN):
  - `ALTER TABLE tabella DROP COLUMN attributo;`
- Modifica valore di default di un attributo (ALTER COLUMN):
  - `ALTER TABLE tabella ALTER COLUMN attributo { SET DEFAULT valore | DROP DEFAULT };`
  - `ALTER TABLE impiegato ALTER COLUMN stipendio SET DEFAULT 1000.00;`

Cancellazione dati in una tabella

Le tuple di una tabella vengono cancellate con il comando DELETE

```
DELETE FROM tabella [ WHERE condizione ];
```

- condizione è una espressione booleana che seleziona quali tuple cancellare.
- Se WHERE non è presente, tutte le tuple saranno cancellate.
  - `DELETE FROM impiegato WHERE matricola = 'A001 ‘;`

Una tabella viene cancellata con il comando DROP TABLE:

```
DROP TABLE tabella;
```