DATABASE

Il ***DATABASE*** è una struttura fisica che consente di effettuare in modalità permanente lo storage di dati, fino alla loro eventuale rimozione fisica.

Esistono 3 tipologie di DATABASE:

- ***DATABASE RELAZIONALE*** (mySQL, Oracle, Postgress, SQL Server)
  La struttura più elementare di storage dei dati è rappresentata dalla tabella.

Ogni tabella è composta da Record (righe) e Field (colonne o campi).
TABELLA = l'unità più elementare che consente di effettuare 4 operazioni:
OPERAZIONI DI CRUD:

- **C** = **CREATION** (INSERIMENTO DI NUOVE INFORMAZIONI)
- **R** = **READ** (LETTURA DI DATI GIA' ESISTENTI)
- **U** = **UPDATE** (MODIFICA DI DATI GIA' ESISTENTI)
- **D** = **DELETE** (CANCELLAZIONE DI DATI GIA' ESISTENTI)

si chiama Relazionale perchè le tabelle possono essere eventualmente relate tra di loro secondo tre diversi tipi di relazione:

1. **OneToOne** (1-1) : al record (riga) di una tabella è associato il record di un'altra tabella (auto e proprietario).
2. **OneToMany** (1-M) : ad un record di una tabella sono associati N record di una tabella (corso e studenti).
   [Per importare una relazione 1-M bisogna creare due tabelle di cui una rappresenta la tabella padre e l'altra rappresenta la tabella figlia, nella quale è necessario assegnare ad un campo il vincolo di ForeignKey verso la PrimaryKey della tabella padre.]
   La 1-M è bidirezionale, contenente due versi:

- OneToMany : ad un record, più record (tabella padre);
- ManyToOne : a più record, un record (tabella figlia).

3. **ManyToMany** (M-M) : a più record di una tabella, sono associati più record di un'altra tabella (user e ruoli) (clienti-prodotti).
   [Per importare una relazione M-M tra due tabelle, c'è bisogno di una creazione di una terza tabella (JOIN TABLE) contenente almeno 2 colonne(campi), che farà da

tabella di unione per le altre due.]
[Nella terza tabella:]

- **[Ogni colonna è una FOREIGN KEY verso la PRIMARY KEY di una delle due tabelle;]**
- **[Entrambe le colonne della JOIN TABLE rappresentano una PRIMARY KEY.]**

ESEMPIO DELLA CREAZIONE DI UNA RELAZIONE **M-M** :

```
//creazione delle 3 tabelle

CREATE TABLE customer ( id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100)
NOT NULL, email VARCHAR(100) NOT NULL );

CREATE TABLE product ( id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100)
NOT NULL, price DECIMAL(10, 2) NOT NULL );

CREATE TABLE customer_product ( customer_id INT, product_id INT, PRIMARY KEY
(customer_id, product_id), FOREIGN KEY (customer_id) REFERENCES
customer(id), FOREIGN KEY (product_id) REFERENCES product(id) );
```

```
//creazione dei parametri dentro le tabelle

insert into customer(name,email)values('Gianni','g@libero.it'); insert into
customer(name,email)values('Giulia','g@virgilio.it'); insert into
product(name,price)values('PC',1800.56); insert into
product(name,price)values('Tablet',500.87); insert into
customer_product(customer_id,product_id)values(1,1); insert into
customer_product(customer_id,product_id)values(1,2); insert into
customer_product(customer_id,product_id)values(2,1); insert into
customer_product(customer_id,product_id)values(2,2)
```

OneToMany e **ManyToMany** sono le più utilizzate. (LE CHIEDONO SEMPRE AI COLLOQUI).

DBMS = **DATABASE MANAGEMENT SYSTEM** = SOFTWARE CON INTERFACCIA GRAFICA CHE CONSENTE LA GESTIONE DI UN DATABASE (CREAZIONE E MANUTENZIONE).

SE NE DISTNGUONO 2 TIPI: RDBMS E NORDBMS.

RDBMS = **RELATIONAL DBMS** = SOFTWARE CHE CONSENTE TRAMITE INTERFACCIA GRAFICA LA GESTIONE DI UN DATABASE RELAZIONALE

NORDBMS = **NO RELATIONAL DBMS** = SOFTWARE CHE CONSENTE TRAMITE INTERFACCIA GRAFICA LA GESTIONE DI UN DATABASE NON RELAZIONALE

JDBC = **JAVA DATABASE CONNECTIVITY** = I[NSIEME DI API FORNITE DA JAVA SE CHE **CONSENTONO L'IMPLEMENTAZIONE DI TUTTE LE OPERAZIONI DI CRUD SU UN DATABASE RELAZIONALE**.]
(ALL'INTERNO DI UNA QUALSIASI APPLICAZIONE JAVA SIA STAND ALONE SIA ENTERPRISE)
(AGGIUNGERE APPUNTTI DEL 28)

> **(al colloquio finale di solito "sistemi" le chiede, non a livello pratico, ma a livello implementativo)**

- ***DATABASE NON RELAZIONALE*** (Mongo DB, Raven DB)
  In un database non relazionale, la struttura più elementare di storage dei dati è rappresentata dal *DOCUMENT*.
  I dati vengono salvati in un Document, sottoforma di file json (ogni json viene chiamata ***Collection***)
- ***DATABASE A GRAFI*** (NeO4J)
  La struttura più elementare di storage dei dati è rappresentata dal nodo di un grafo.
  Il contenuto del nodo rappresenta le informazioni vere e proprie

Tutte le operazioni di CRUD sul database relazionale, devono essere eseguite tramite linguaggio SQL (Structured Query Language).

SQL è un linguaggio universalmente comprensibile da tutti i database relazionali per l'esecuzione delle 4 operazioni di CRUD

Dialetti : SQL = PL-SQL, T-SQL

PL-SQL = Procedure Language Structured Query Language -> >>>> Database Oracle
T-SQL = Transection Structured Query Language -> >>>> Azienda Microsoft

All'interno di ogni tabella di un database relazionale è possibile impostare uno o più campi, rappresentanti un vincolo (constraint) di nome PK (Primary Key)

PK (Primary Key) --> >>>> il campo che rappresenta la PK ha valore univoco per ogni singolo record della tabella non è possibile avere informazioni ridondanti all'interno della tabella

---

La *JUnit* è un API JAVA SE che consente di effettuare test unitari all'interno di qualsiasi applicazione, ovvero testare singoli metodi.

Quando si vuole utilizzare la JUnit per testare i metodi, c'è bisogno di:

1. Creare una classe nel package "test" (tasto destro su "java" -> new -> Class);
2. Dare un nome alla classe ("CrudTest");
3. Mettere l'annotazione "@Test";
4. Invocare il metodo da testare;
5. Run del CrudTest (tasto destro sulla classe creata).

ESEMPIO :

```
import com.sistemi.informativi.connection.ConnectionManager;
import com.sistemi.informativi.dao.FacadeDAO;
import com.sistemi.informativi.dao.FacadeDAOImpl;
import org.junit.jupiter.api.Test;

import java.sql.Connection;
import java.sql.SQLException;

public class CrudTest {

    @Test
    public void removeCourseTest() throws SQLException,
ClassNotFoundException {

        FacadeDAO facadeDAO = new FacadeDAOImpl();
        Connection con = ConnectionManager.getConnection();

        //invocazione del metodo da testare
        facadeDAO.removeCourse(3);

        ConnectionManager.closeConnection();

    }

}
```

H2 = ***Database Relazionale Embedded*** (Interno all'Applicazione).
Infatti non occorre utilizzare software esterni come XAMPP, dato che il Database "vive" all'interno dell'Applicazione.
E' un Database usato come Database di *TEST*.
Ed è un Database Volatile, quindi al Restart dell'Applicazione, i dati vengono cancellati.

starters DI H2 (nell'applicazione):
SPRING WEB = SPRING CORE + SPRING MVC + SPRING RESTFUL
SPRING DATA JPA = JPA + SPRING DATA JPA + HIBERNATE