Database

Cos'è un DBMS?

Un **Database Management System** è un sistema software che permette di gestire un database, ovvero un insieme di dati strutturati e correlati tra loro.

Spesso ci si riferisce ai **DBMS** utilizzando impropriamente il termine database, a causa dell'accoppiamento stretto tra l'archivio dati (databse) e il software di gestione (DBMS).

Funzionalità

I DBMS sono progettati per essere utilizzati da più utenti contemporaneamente, e permettono di gestire in modo efficiente e sicuro i dati, garantendo la loro integrità e la loro persistenza nel tempo.

Il sistema di sicurezza di un DBMS permette di gestire i permessi di accesso ai dati, impedendo ad utenti non autorizzati di accedere alle informazioni.

Per esempio un database di impiegati può contenere tutti i dati riguardanti un singolo soggetto e un gruppo di utenti può essere autorizzato a vedere solamente i dati riguardanti lo stipendio, mentre altri utenti possono essere autorizzati a vedere solamente le informazioni che riguardano la sua storia lavorativa e la situazione sanitaria.

Funzionalità

I DBMS permettono di mantenere l'integrità dei dati, ovvero di garantire che i dati siano sempre corretti e coerenti, ad esempio impedendo che un utente possa inserire un valore non valido in un campo di tipo numerico, o che più utenti possano modificare contemporaneamente lo stesso dato.

Le query sono un altro aspetto fondamentale dei DBMS, permettono di interrogare il database per estrarre informazioni specifiche, ad esempio è possibile estrarre tutti i dati relativi agli impiegati che hanno un determinato stipendio.

Tipi di DBMS

Esistono diversi tipi di DBMS, i più comuni sono i **relazionali**, i **documentali**, i **gerarchici** e i **grafici**.

I DBMS relazionali sono i più diffusi, sono basati sul modello relazionale e utilizzano il linguaggio SQL per interrogare il database.

I DBMS documentali sono basati sul modello a documenti, i dati sono organizzati in documenti e sono accessibili tramite chiave, degli esempi di DBMS documentali sono MongoDB e firebase.

I DBMS gerarchici sono basati sul modello gerarchico, i dati sono organizzati in una struttura ad albero, un esempio di DBMS gerarchico è IMS.

I DBMS grafici sono basati sul modello a grafo, i dati sono organizzati in nodi e archi, un esempio di DBMS grafico è Neo4j.

DBMS relazionali

I DBMS relazionali sono basati sul modello relazionale, i dati sono organizzati in tabelle, le tabelle sono composte da righe e colonne, le colonne rappresentano gli attributi e le righe rappresentano le istanze.

I DBMS relazionali utilizzano il linguaggio SQL per interrogare il database, SQL è originariamente un linguaggio dichiarativo, ovvero un linguaggio che permette di descrivere il risultato che si vuole ottenere, senza specificare come ottenerlo, ma si è successivamente evoluto con l'introduzione di costrutti procedurali, istruzioni per il controllo di flusso, tipi di dati definiti dall'utente e varie altre estensioni del linguaggio.

DBMS relazionali

Essendo un linguaggio dichiarativo, SQL non richiede la stesura di sequenze di operazioni (come i linguaggi imperativi), ma di specificare le proprietà logiche delle informazioni ricercate. Esso si divide in:

- **DDL** (Data Definition Language), utilizzato per definire la struttura del database, ovvero le tabelle, gli attributi e le relazioni tra le tabelle.
- **DML** (Data Manipulation Language), utilizzato per manipolare i dati, ovvero per inserire, modificare e cancellare i dati.
- **QL** (Query Language), utilizzato per interrogare il database, ovvero per estrarre informazioni specifiche.
- **DCL** (Data Control Language), utilizzato per gestire i permessi e gli utenti.

SQL

Noi ci concentreremo sul linguaggio SQL, che è il linguaggio più diffuso per interrogare i database relazionali.

```
SELECT *
FROM tabella
WHERE condizione
GROUP BY attributo
HAVING condizione
ORDER BY attributo
```

SQL

Per poter eseguire una interrogazione (query) del database utilizziamo il comando SELECT

Con questo comando è possibile decidere quali colonne visualizzare (**proiezione**), quali righe visualizzare (**selezione**) e quali tabelle unire (**congiunzione**)

Per capire come si interroga un database partiamo da un database in cui ci sono 2 tabelle collegate da una associazione 1:N

IDStudente	Nome	Cognome	DataNascita	CittaNascita	Altezza
1	Ryan	Gosling	21/1/1980	Londra	184
2	Patrick	Bateman	25/12/1978	New York	182
3	Bruce	Wayne	13/9/1997	Gotham City	182
4	Walter	White	14/9/1997	Albuquerque	179
5	Tyler	Durden	23/2/1995	New York	180

IDCompito	Materia	Data	Voto	Tipo	IDstudente
1	matematica	13/10/2014	7	scritto	1
2	storia	23/10/2014	8	scritto	3
3	matematica	13/11/2014	6	orale	1
4	matematica	13/11/2014	7	orale	4
5	geografia	15/5/2013	10	orale	4
6	inglese	13/4/2013	2	scritto	2
7	storia	3/10/2016	4	scritto	1

Proiezione

Per sapere i nomi e i cognomi di tutti gli studenti possiamo usare la query:

```
SELECT Nome, Cognome
FROM Studente
```

Il risultato sarà:

Nome	Cognome
Ryan	Gosling
Patrick	Bateman
Bruce	Wayne
Walter	White
Tyler	Durden

Possiamo ad esempio invertire le colonne e ordinare i risultati in ordine alfabetico per cognome:

```
SELECT Cognome, Nome
FROM Studente
ORDER BY Cognome
```

Il risultato sarà:

Cognome	Nome
Bateman	Patrick
Durden	Tyler
Gosling	Ryan
Wayne	Bruce
White	Walter

Possiamo oppure ordinare i risultati in ordine alfabetico discendente per cognome e poi per nome:

```
SELECT Cognome, Nome
FROM Studente
ORDER BY Cognome DESC, Nome DESC
```

Cognome	Nome
White	Walter
Wayne	Bruce
Gosling	Ryan
Durden	Tyler
Bateman	Patrick

Possiamo selezionare anche ogni colonna della tabella:

```
SELECT Nome, Cognome, DataNascita, CittaNascita, Altezza
FROM Studente
```

O meglio:

```
SELECT *
FROM Studente
```

Che ritornerà tutti i dati della tabella Studente iniziale

Se vogliamo conoscere tutte le città di nascita degli studenti senza ripetizioni possiamo usare la parola chiave DISTINCT:

```
SELECT DISTINCT CittaNascita
FROM Studente
```

CittaNascita
Londra
New York
Gotham City
Albuquerque

Possiamo anche rinominare le colonne:

```
SELECT CittaNascita AS Città
FROM Studente
```

Città
Londra
New York
Gotham City
Albuquerque

Selezione

Spesso vogliamo selezionare solo alcune righe della tabella, si impongono quindi delle condizioni che devono essere soddisfatte per selezionare una riga, utilizzando gli operatori di confronto `=` `<` `>` `<=` `>=` `<>` e gli operatori logici `AND` `OR` `NOT`

Per esempio vogliamo conoscere i nomi e i cognomi degli studenti che sono nati a New York:

```
SELECT Nome, Cognome, CittaNascita AS Città
FROM Studente
WHERE CittaNascita = 'New York'
```

Nome	Cognome	Città
Patrick	Bateman	New York
Tyler	Durden	New York

Possiamo anche selezionare gli studenti che sono nati a New York o a Gotham City:

```
SELECT Nome, Cognome, CittaNascita AS Città
FROM Studente
WHERE CittaNascita= 'New York' OR CittaNascita = 'Gotham City'
```

Nome	Cognome	Città
Patrick	Bateman	New York
Bruce	Wayne	Gotham City
Tyler	Durden	New York

Possiamo anche selezionare gli studenti che sono nati a New York e che sono alti più di 175 cm:

```
SELECT Nome, Cognome, CittaNascita AS Città, Altezza
FROM Studente
WHERE CittaNascita = 'New York' AND Altezza > 180
```

Nome	Cognome	Città	Altezza
Patrick	Bateman	New York	182

Possiamo anche selezionare gli studenti che sono nati a New York o a Gotham City e che sono alti più di 175 cm:

```
SELECT Nome, Cognome, CittaNascita AS Città, Altezza
FROM Studente
WHERE (CittaNascita= 'New York' OR CittaNascita = 'Gotham City') AND Altezza > 180
```

O meglio:

```
SELECT Nome, Cognome, CittaNascita AS Città, Altezza
FROM Studente
WHERE CittaNascita IN ('New York', 'Gotham City') AND Altezza > 180
```

Nome	Cognome	Città	Altezza
Patrick	Bateman	New York	182
Bruce	Wayne	Gotham City	182

Per specificare la lettera iniziale di un nome o di un cognome possiamo usare l'operatore LIKE:

```
SELECT Nome, Cognome
FROM Studente
WHERE Nome LIKE 'R%'
```

Nome	Cognome
Ryan	Gosling

Il carattere `%` indica che possono esserci 0 o più caratteri dopo la R, invece il carattere `_` indica che deve esserci un solo carattere dopo la R.

Se vogliamo sapere gli studenti che fanno parte della generazione dei millenials, possiamo usare l'operatore BETWEEN:

```
SELECT Nome, Cognome, DataNascita
FROM Studente
WHERE DataNascita BETWEEN '1/1/1981' AND '31/12/1996'
```

Nome	Cognome	DataNascita
Bruce	Wayne	13/9/1997
Walter	White	14/9/1997
Tyler	Durden	23/2/1995

Se vogliamo avere la top 3 degli studenti più alti possiamo usare la parola chiave TOP:

```
SELECT TOP 3 Nome, Cognome, Altezza
FROM Studente
ORDER BY Altezza DESC
```

Nome	Cognome	Altezza
Ryan	Gosling	184
Patrick	Bateman	182
Bruce	Wayne	182

Aggregazione

Possiamo anche effettuare delle operazioni di aggregazione, come la somma, la media, il conteggio, il minimo e il massimo, utilizzando le funzioni `SUM` `AVG` `COUNT` `MIN` `MAX`

Ad esempio vogliamo sapere la media delle altezze degli studenti:

```
SELECT AVG(Altezza)
FROM Studente
```

Altezza
181.4

Ovviamente non possiamo scrivere `SELECT AVG(Altezza), nome FROM Studente`

Possiamo anche usare più funzioni di aggregazione contemporaneamente:

```
SELECT AVG(Altezza) AS Media, MIN(Altezza) AS Minimo, MAX(Altezza) AS Massimo
FROM Studente
```

Media	Minimo	Massimo
181.4	179	184

Cosa succederebbe se mettessimo tutti gli studenti uno sopra l'altro?

```
SELECT SUM(Altezza) AS Somma
FROM Studente
```

Somma
907

Possiamo anche raggruppare i risultati in base al valore di una colonna, utilizzando la parola chiave GROUP BY:

```
SELECT CittaNascita AS Città, COUNT(*) AS NumeroStudenti
FROM Studente
GROUP BY CittaNascita
```

Città	NumeroStudenti
Albuquerque	1
Gotham City	1
Londra	1
New York	2

Possiamo anche filtrare i risultati raggruppati, utilizzando la parola chiave HAVING:

```
SELECT CittaNascita AS Città, COUNT(*) AS NumeroStudenti
FROM Studente
GROUP BY CittaNascita
HAVING COUNT(*) <= 1
```

Città	NumeroStudenti
Albuquerque	1
Gotham City	1
Londra	1

Congiunzione

Possiamo anche unire due tabelle, se scriviamo `SELECT * FROM Studente, Compito` otterremo il prodotto cartesiano tra le due tabelle, ovvero ogni riga della prima tabella verrà accoppiata con ogni riga della seconda tabella. Otterremmo quindi in totale 35 righe (7 * 5). Questo tipo di prodotto non ha molto senso, ma possiamo utilizzare la parola chiave WHERE per filtrare i risultati:

```
SELECT *
FROM Studente, Compito
WHERE Studente.IDStudente = Compito.IDStudente
```

IDStudente	Nome	Cognome	DataNascita	CittaNascita	Altezza	IDCompito	Materia	Data	Voto	Tipo	IDstudente
1	Ryan	Gosling	21/1/1980	Londra	184	1	matematica	13/10/2014	7	scritto	1
1	Ryan	Gosling	21/1/1980	Londra	184	3	matematica	13/11/2014	6	orale	1
1	Ryan	Gosling	21/1/1980	Londra	184	7	storia	3/10/2016	4	scritto	1
2	Patrick	Bateman	25/12/1978	New York	182	6	inglese	13/4/2013	2	scritto	2
3	Bruce	Wayne	13/9/1997	Gotham City	182	2	storia	23/10/2014	8	scritto	3
4	Walter	White	14/9/1997	Albuquerque	179	4	matematica	13/11/2014	7	orale	4
4	Walter	White	14/9/1997	Albuquerque	179	5	geografia	15/5/2013	10	orale	4

Durante questa congiunzione abbiamo perso qualche studente, infatti non tutti gli studenti hanno svolto un compito.

Per selezionare anche gli studenti che non hanno svolto un compito possiamo usare la parola chiave LEFT JOIN, magari scrivendo solamente i voti:

```
SELECT Nome, Cognome, Voto
FROM Studente LEFT JOIN Compito ON Studente.IDStudente = Compito.IDStudente
```

Nome	Cognome	Voto
Ryan	Gosling	7
Ryan	Gosling	6
Ryan	Gosling	4
Patrick	Bateman	2
Bruce	Wayne	8
Walter	White	7
Walter	White	10
Tyler	Durden	

Possiamo visualizzare la media dei voti di ogni studente:

```
SELECT Nome, Cognome, AVG(Voto) AS Media
FROM Studente LEFT JOIN Compito ON Studente.IDStudente = Compito.IDStudente
GROUP BY Nome, Cognome
ORDER BY Media DESC
```

Nome	Cognome	Media
Walter	White	8.5
Bruce	Wayne	8
Ryan	Gosling	5.67
Patrick	Bateman	2
Tyler	Durden	