INFORMATICA
U1 L1 - Introduzione ai database

La memorizzazione di dati in file di record mediante un linguaggio di programmazione è un'operazione complessa, laboriosa e pesante da un punto di vista computazionale e man mano che si va avanti, continueranno ad aumentare le dimensioni delle strutture dati, andando così a peggiorare la situazione, per questo motivo si è arrivati ad una soluzione efficace, l'utilizzo dei database.

Un database è una raccolta di dati progettati in modo tale da poter essere utilizzati in maniera ottimizzata da differenti applicazioni e da utenti diversi.

Le informazioni necessarie a un'organizzazione sono gestite da un sistema informativo.
Un sistema informativo è un insieme organizzato di strumenti automatici, procedure manuali, risorse umane e materiali, norme organizzative, orientato alla gestione delle informazioni rilevanti per un'organizzazione.

Un sistema informatico è un sottoinsieme del sistema informativo che si dedica alla gestione automatica delle informazioni, rappresentate mediante dati digitali.
Esso è costituito dagli archivi elettronici in cui sono memorizzati tutti i dati relativi all'azienda

- Archivi e applicazioni (Software)
- Supporti fisici e strumentazione (Hardware)

DBMS: sistema di gestione del database visto nel suo complesso. Si preoccupa di gestire interamente i dati, compresa la loro definizione e il modo in cui vengono fisicamente archiviati.

Problemi legati all'uso degli archivi:

- Ridondanza dei dati
- Incongruenza dei dati: se un dato di un articolo viene modificato, tale modifica andrà effettuata su tutti i record che contengono quell'articolo
- Inconsistenza: possibilità che ci siano due valori diversi per lo stesso dato senza poter risalire al valore corretto
- Impossibilità di modificare la struttura senza generare un insieme di modifiche a cascata(Dipendenza logica)
- Scarsa flessibilità in caso di nuove esigenze che potrebbero essere irrealizzabili a causa della struttura degli archivi(Dipendenza fisica)
- I dati contenuti negli archivi possono essere trattati solo elaborando i file di record

Caratteristiche dei DBMS:

- Gestione efficiente:
  Gestire grandi quantità di dati
- Condivisione e gestione della concorrenza:
  Garantire la condivisione dei dati che devono poter essere usati da applicazioni e utenti diversi secondo proprie modalità
- Persistenza e affidabilità:
  Garantire la persistenza dei dati che devono durare nel tempo e memorizzare i dati su memoria secondaria.
  Garantire l'affidabilità delle operazioni(transizioni, operazioni fisiche elementari sul database che generano una variazione del suo stato) che devono operare sui dati rispettando le proprietà ACID:
- Sicurezza:

Gestire il controllo degli accessi per assicurare che i dati siano visibili solo da particolari utenti o gruppi di utenti in possesso dei diritti necessari.

Proprietà ACID

- Atomicity: L'operazione o viene eseguita nella sua interezza o non è eseguita
- Consistency preservation: Un'esecuzione corretta della transazione deve portare il DB in un nuovo stato consistente, così come quello da cui è partito.
- Isolation: non si deve permettere di visionare i risultati di operazioni intermedie ad altre transazioni. ma solo alla sua normale e corretta terminazione si possono vedere i risultati che essa ha prodotto
- Durability(Persistenza): Alla terminazione di una transazione su un DB i risultati devono essere permanenti

U1 L2 - Modelli classici di database e tecniche di progetto

Progettazione di un database - passi principali:

1. Analisi del problema
2. Progettazione concettuale del database (Modello E-R / a Oggetti)
3. Progettazione logica del database(Schema logico)
4. Progettazione fisica e implementazione
5. Realizzazione delle applicazioni

Il modello concettuale descrive cosa deve essere rappresentato, mentre il modello logico descrive come sono organizzati i dati.

Progettazione logica:

Modello logico dei dati: insieme di costrutti utilizzati per organizzare i dati di interesse e descriverne la dinamica.
Nel corso degli anni sono stati proposti molti modelli logici che hanno anche seguito l'evoluzione dell'informatica

Modello gerarchico
Modello reticolare
Modello relazionale
Modello a oggetti
XML
Schema-less

Modello relazionale, ha l'obiettivo di non duplicare inutilmente le informazioni e consiste in un insieme di tabelle che possono essere connesse tra loro mediante relazioni.
Ogni tabella è composta da righe(Record/Tuple) e colonne(Campi/Proprietà)
Vantaggi:

- Indipendenza dei dati
- Rappresentazione logica non fa alcun riferimento a quella fisica
- Collocazione dell'informazione nei campi e non in strutture fisiche come i puntatori

E' attualmente il modello logico di dati più utilizzato

Linguaggio SQL

- DDL: Data Definition Language, definisce schemi delle relazioni, vincoli tra i dati, domini dei dati ecc
- DML: Data Manipulation Language, legge, modifica, elimina i dati contenuti nel DB
- DCL: Data Control Language, fornisce o revoca i permessi agli utenti
- QL: Query Language, estrae informazioni dal database, interrogato mediante query

U2 L1 - Elementi di algebra relazionale

Il modello relazionale usa il concetto di relazione matematica come suo componente elementare e ha il suo fondamento teorico nella teoria degli insiemi e nella logica dei predicati del primo ordine.

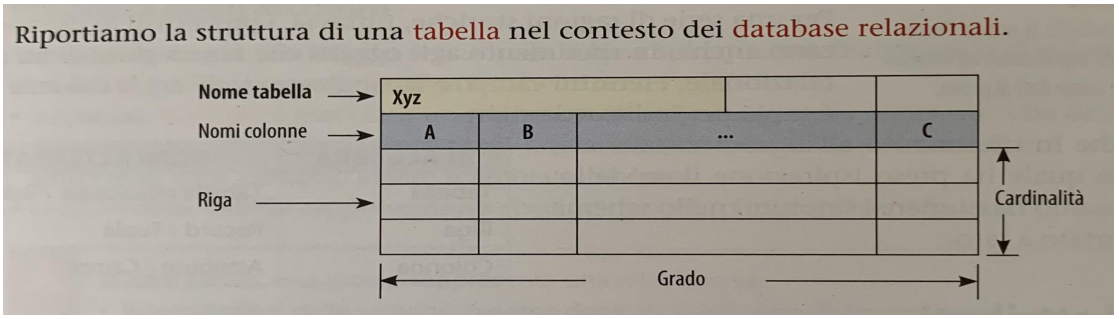
5 operazioni di base:
- Selezione(σ): Seleziona un sottoinsieme di righe della relazione
- Proiezione(π): Cancella colonne non desiderate dalla relazione
- Prodotto cartesiano(x): Consente di combinare due relazioni
- Differenza insiemistica(-): Tuple presenti nella relazione 1, ma non nella relazione 2
- Unione(u): Tuple presenti nella relazione 1 e nella relazione 2.

Selezione e proiezione sono chiamati operatori unari dell'algebra relazionale in quanto si applicano a una relazione e restituiscono una relazione.
LEFT JOIN(=⋈): conserva tutte le righe della tabella di sinistra
RIGHT JOIN(⋈=): conserva tutte le righe della tabella di destra
FULL JOIN(=⋈=): conserva tutte le righe di entrambe le tabelle

U2 L2 - Il modello relazionale: attributi e chiavi

L'organizzazione dei dati nelle tabelle relazionali è nota come vista logica
Il modo in cui il software del database salva fisicamente i dati sul disco fisso di un computer è detto vista interna

Con schema di un database si intende la descrizione della struttura che viene rappresentata nell'intestazione di ogni tabella, composta dal nome della tabella stessa e dai nomi delle colonne.



- Colonne: Contengono informazioni sulla tabella
- Righe: riportano le istanze degli oggetti rappresentati dalla tabella
- Dominio: insieme dei valori che possono essere presenti in una colonna
- Grado: numero delle colonne
- Cardinalità: numero delle righe

Dominio degli attributi:
- Tipo di dato: es. decimale, int, char
- Lunghezza
- Intervallo

Proprietà degli attributi:

Una tabella rappresenta una relazione se:

- Le intestazioni delle colonne, o nome degli attributi, sono diverse tra loro.
- I valori di ciascuna colonna sono fra loro omogenei, appartengono allo stesso dominio.
- Le righe sono diverse fra loro, quindi ogni riga è univoca.

Il campo identificatore(Chiave)

I vincoli di chiave si esprimono a livello di schema, sulla base di un'analisi della realtà che si vuole modellare: classici campi chiave sono quelli che per loro natura sono unici in un insieme, come il numero di matricola, il codice fiscale, la partita IVA, la targa ecc…
L'esistenza delle chiavi garantisce l'accessibilità a ciascun dato della tabella, in quanto ogni singolo valore è univocamente individuato da nome della relazione, valore della chiave e nome dell'attributo.
In qualche tabella potrebbe anche non essere presente un attributo che da solo svolge la funzione di chiave, mentre per tale scopo si potrebbero utilizzare due attributi presi "assieme": in tal caso parliamo di chiave composta.

Nei casi in cui non si possa garantire per nessun attributo la disponibilità di valori differenti tra loro, è necessario introdurre un nuovo attributo, cioè un particolare "codice contatore" che svolga le funzioni di campo chiave.
Questa chiave prende il nome di chiave artificiale e generalmente viene dato a questo attributo il nome di ID(identificatore)

Le chiavi sono lo strumento principale attraverso il quale vengono correlati i dati in relazioni diverse nel modello relazionale.

Chiave primaria e candidata
In una relazione potrebbero anche esserci più campi chiave: tutti questi campi sono candidati a diventare l'attributo identificativo(chiave primaria)
Nello schema della tabella relazionale generalmente questo attributo viene collocato "in prima posizione" e viene evidenziato/sottolineato/caratterizzato dalla sigla pk.

Chiave esterna
Una base di dati è costituita da molte tabelle e spesso le informazioni contenute in più relazioni sono correlate tra loro, questi attributi prendono il nome di chiavi esterne e sono attributi che assumono il ruolo di chiave primaria nella tabella che deve essere messa in relazione con la seconda tabella.

U2 L3 - I vincoli di integrità intra e inter relazionali

Nei DBMS viene messo in atto un insieme di regole che cercano di evitare l'inserimento all'interno dell'archivio di dati errati e di prevenire situazioni di inconsistenza quando si eseguono operazioni di modifica e/o cancellazione di informazioni in una tabella che magari sono collegate ad altre informazioni presenti in altre tabelle.
Due categorie di vincoli di integrità:

- Intrarelazionale: vincoli su singoli attributi della tabella in questione
  Vincoli sugli attributi o di tupla(Statici - con controllo solo all'inserimento/variazione, Dinamici - con controllo "periodico")
  Vincoli di chiave
- Interrelazionale: vincoli su attributi che coinvolgono più relazioni presenti sul db
  Vincoli sui valori a causa di legami che vengono espressi nel modello concettuale da cui derivano

U4 L1 - Progettazione concettuale: i diagrammi E-R

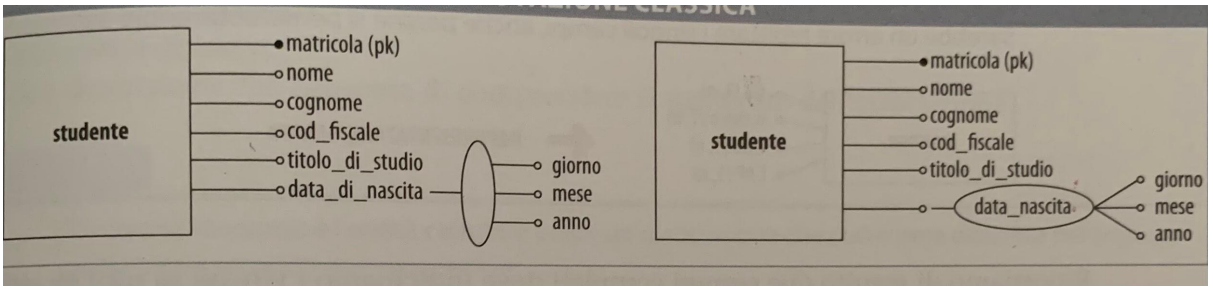
L'utilità del modello E-R è così riassumibile:

- I costrutti del modello possono essere impiegati facilmente per la definizione dei database relazionali
- è semplice e facile da capire con un minimo di guida. perciò può essere usato dai progettisti dei database per comunicare la struttura agli utenti finali
- può essere usato come piano di lavoro per gli sviluppatori del database per implementare un modello di dati in uno specifico software di gestione di database

Entità

Sono classi di oggetti(fatti, persone, cose) che hanno proprietà comuni e con esistenza autonoma che richiama la progettazione delle classi della OOP.

Nella rappresentazione grafica delle entità vengono indicati gli attributi e la chiave primaria, che viene rappresentata con una grafica differente in modo da essere facilmente riconoscibile.



Le associazioni tra più entità

Nel modello relazionale un db è una collezione di tabelle, ciascuna descritta mediante una notazione chiamata schema, dove gli attributi di ciascuna tupla sono legati in una relazione.

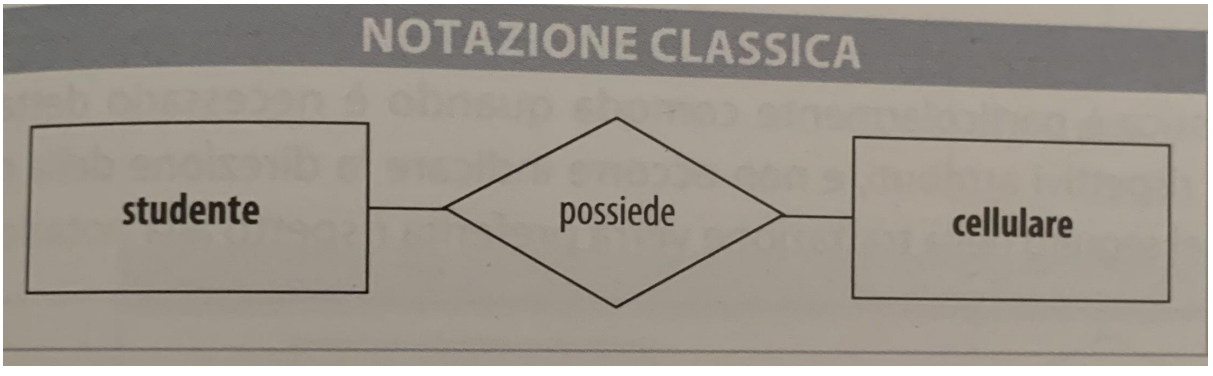
Le tabelle possono essere in collegamento logico tra di loro, con le tuple legate tramite coppie "chiavi primarie-chiavi esterne".

Una associazione tra due tabelle è spesso esprimibile tramite un verbo.

Ogni associazione ha due versi con specifici significati e ogni verso si compone di:

- Un'entità di partenza
- Un'entità di arrivo
- Una descrizione

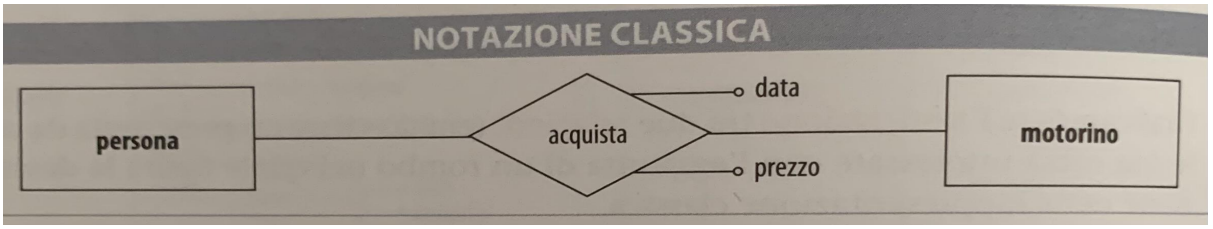
Es.



Attributi delle relazioni

Anche le relazioni, come le entità, possono avere degli attributi che le caratterizzano

Es.



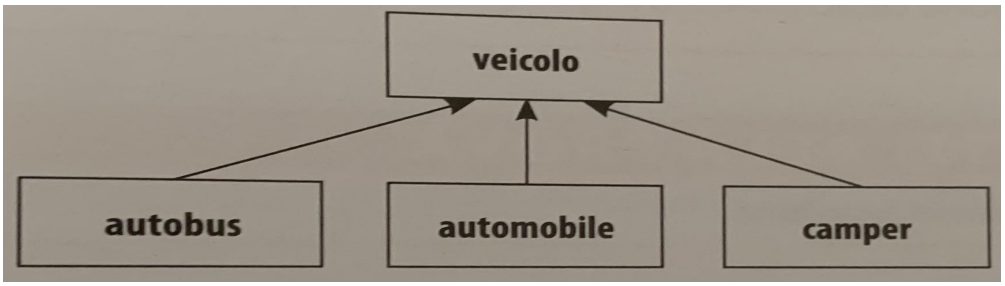
Classificazione delle relazioni
Grado:
Il grado di una relazione è il numero di entità associate alla relazione es. binarie, ternarie.

Relazioni ricorsive:
Una relazione è ricorsiva quando esiste una relazione tra un'entità e se stessa

Cardinalità:
Per ogni entità che partecipa a un'associazione bisogna indicare il numero minimo e massimo di istanze della relazione a cui un'istanza dell'entità può partecipare.
L'indicazione del numero delle istanze delle entità associate a una relazione prende il nome di cardinalità.

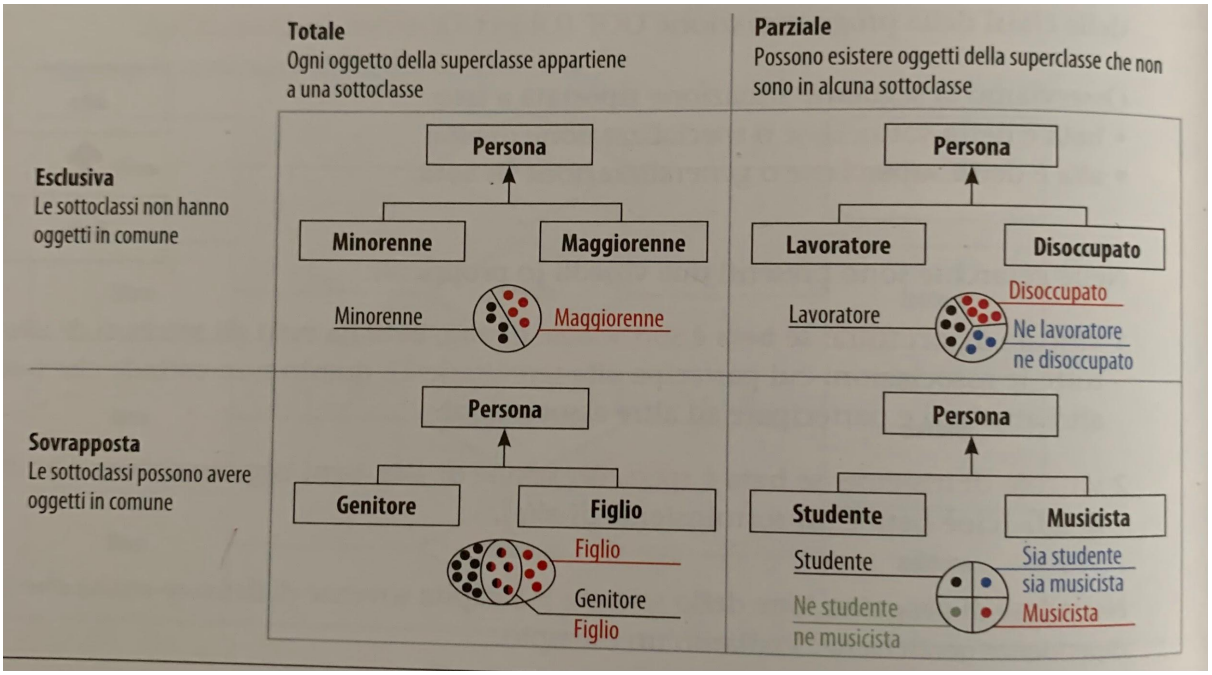
- Uno a uno(1,1) si ottiene solo quando al massimo una istanza dell'entità A viene associata a una sola istanza dell'entità B
  es. Ogni nazione(A) ha una capitale(B)
- Uno a molti(1,N) si ottiene quando per una istanza dell'entità A ci sono zero, una, o molte istanze dell'entità B, ma per una istanza dell'entità B c'è solo una istanza dell'entità A
  es. Un ufficio(A) ha molti impiegati(B), e ogni impiegato(B) è associato a un solo ufficio(A).
- Molti a molti(N,N) si ottiene quando per una istanza dell'entità A ci sono zero, una, o molte istanze dell'entità B e viceversa
  es. Un insegnante(A) insegna a più classi(B) e ogni classe(B) ha più insegnanti(A)
- Esistenza opzionale(0,N / 0,1) se l'istanza di un'entità non è richiesta
- Esistenza obbligatoria(1,N / 1,1) se l'istanza di un'entità deve necessariamente esserci perchè un'entità sia inclusa in una relaziona

Generalizzazione:
Esistono situazioni dove tra le entità può essere stabilita una gerarchia.



Le generalizzazioni si caratterizzano per "due dimensioni indipendenti";
1) Confronto fra unione delle specializzazioni e classe generalizzata:
   - Totale, se ogni oggetto della superclasse appartiene a una sottoclasse
   - Parziale, se possono esistere oggetti della superclasse che non sono in alcuna sottoclasse
2) Confronto fra le classi specializzate:
   - Esclusiva, se le sottoclassi non hanno oggetti in comune
   - Sovrapposta, se le sottoclassi possono avere oggetti in comune

Tabella generalizzazioni:



U5 L1 - La normalizzazione delle tabelle

La normalizzazione è una tecnica di progettazione ampiamente utilizzata nel disegno di database relazionali, con l'obiettivo di scomporre le tabelle in tabelle più piccole e senza anomalie, in modo da avere relazioni più strutturate e ridurre al minimo la ridondanza dei dati.

Un db non normalizzato presenta problemi di:

- Ridondanza
- Inefficienza
- Complessità
- Perdita di informazioni

Prima forma normale:

Una relazione si dice in prima forma normale se e solo se tutti i suoi attributi sono valori atomici, ovvero non sono formati da campi composti

Seconda forma normale:

Una tabella in seconda forma normale è una tabella in prima forma normale in cui tutte le colonne non chiave sono completamente dipendenti dall'intera chiave primaria: non sono cioè ammesse colonne che dipendono funzionalmente solo da una parte della chiave primaria composta.
In una tabella dove la chiave primaria è semplice non ci sono problemi inerenti alla seconda forma normale.

Terza forma normale:

La terza forma normale richiede che la tabella sia già in seconda forma normale e che tutte le colonne in una tabella relazionale siano dipendenti solamente dalla chiave primaria.

SQL Views

Le viste sono un elemento utilizzato dalla maggior parte dei DBMS. Si tratta, come suggerisce il nome, di "modi di vedere i dati".

Una vista è rappresentata da una query (SELECT), il cui risultato può essere utilizzato come se fosse una tabella.

Le viste generalmente vengono utilizzate per semplificare le query. Se il database è realmente relazionale, leggere un insieme di dati avente un significato potrebbe essere complesso, perché potrebbe richiedere eccessive JOIN tra tabelle; con una vista è possibile semplificare molto la stesura di query che leggono le informazioni.

Es.

```
CREATE VIEW [Products Above Average Price] AS
SELECT ProductName, Price
FROM Products
WHERE Price > (SELECT AVG(Price) FROM Products);
```

SQL Trigger

Il trigger (grilletto in italiano) è un insieme di comandi SQL che viene eseguito automaticamente a causa di eventi che modificano il database. L'attivazione di un trigger è provocata da un evento che fa scattare un controllo sulla base di dati. Se le condizioni del controllo sono verificate, i comandi SQL programmati nel trigger vengono eseguiti; se la condizione controllata non si verifica non accade nulla. Per questa ragione i trigger sono indicati con il termine di regole
Es.

Mostriamo, con un esempio, come usare un trigger per impedire che lo stipendio di un dipendente possa essere diminuito o che il dipendente ottenga un aumento maggiore del 30%.

```
CREATE TRIGGER VariazioneStipendio
AFTER UPDATE ON Impiegati                          Evento UPDATE di Stipendio
FOR EACH ROW
BEGIN
   IF  NEW.Stipendio < OLD.Stipendio OR            Condizione di attivazione
       NEW.Stipendio > OLD.Stipendio*1.3 THEN      del trigger

   UPDATE Impiegati                                Comando SQL
   SET Stipendio = OLD.Stipendio                   eseguito se la condizione
   WHERE ID = NEW.ID;                              è verificata

   END IF
END;
```

SQL Permessi

-- crea utente che può accedere da qualsiasi host

CREATE USER utentemaga IDENTIFIED BY 'pwd';

se non lascia ricreare l'utente usare FLUSH PRIVILEGES;

-- visualizza i privilegi

SHOW GRANTS for utentemaga;

--assegna tutti i privilegi a utentemaga sul db magazzino

GRANT ALL PRIVILEGES ON magazzino.* TO utentemaga;

-- rimuove tutti i privilegi a utentemaga sul db magazzino

REVOKE ALL PRIVILEGES ON magazzino.* FROM utentemaga;

– assegna il privilegio di select a utentemaga sul db magazzino;

GRANT SELECT ON magazzino.* TO utentemaga;

--cancellazione utente

DROP USER utentemaga;

MongoDB

MongoDB è un DBMS non relazionale, orientato ai documenti.

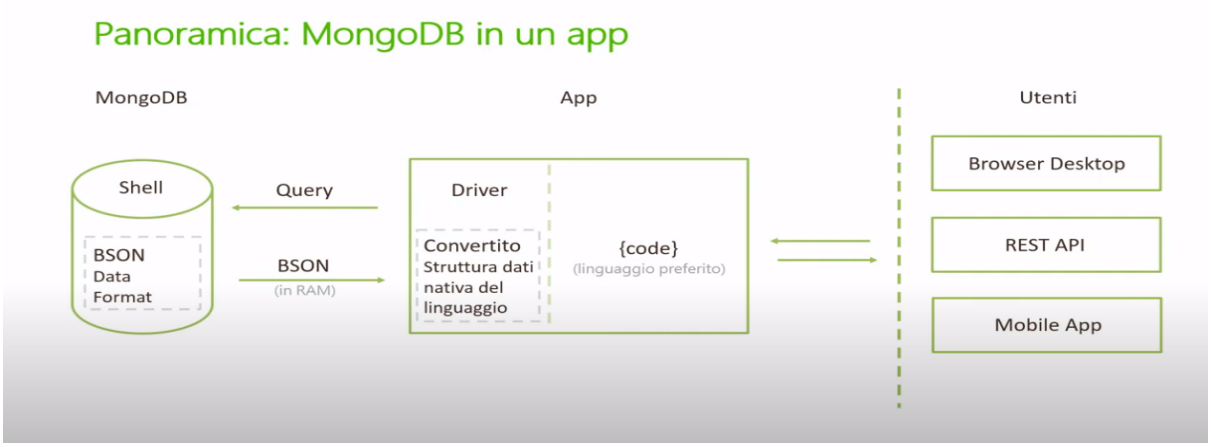
Classificato come un database di tipo NoSQL, MongoDB si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in favore di documenti in stile JSON con schema dinamico (MongoDB chiama il formato BSON), rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce.

I database NoSQL (noti anche come “non solo SQL”) non sono tabulari e memorizzano i dati in modo diverso rispetto alle tabelle relazionali. I database NoSQL sono disponibili in una varietà di tipologie in base al loro modello di dati. I tipi principali sono di documenti, di valore chiave, wide-column e grafici, e forniscono schemi flessibili e facilmente scalabili con grandi quantità di dati e carichi elevati degli utenti.

Il suo obiettivo è la scalabilità e le sue caratteristiche chiave sono:

- servizi di alta disponibilità, dal momento che la replicazione di un database (Replica Set) può avvenire in modo molto semplice;
- garantisce la scalabilità automatica, ossia la possibilità di distribuire (Sharding) le collezioni in cluster di nodi, in modo da supportare grandi quantità di dati senza influire pesantemente sulle performance.

MongoDB si adatta a molti contesti, in generale quando si manipolano grandi quantità di dati eterogenei e senza uno schema. Non è invece opportuno quando si devono gestire molte relazioni tra oggetti, e si vuole garantire l'integrità referenziale tra essi (ad esempio in un ERP).



Gli indici

Un indice è una tabella ausiliaria associata a un campo (o più campi) della tabella primaria, e costruita quindi sui valori assunti da quel campo, che permette di velocizzare gli accessi ai valori del campo stesso.
Associare un indice a un campo significa, quindi, poter effettuare ricerche sui valori che quel campo assume con tempi ridotti rispetto a una ricerca con accesso sequenziale, in quanto i valori di un indice sono sempre mantenuti ordinati ed è perciò possibile utilizzare un accesso per chiave che guidi la ricerca direttamente alla zona della tabella primaria in cui si trova eventualmente il dato richiesto.

Nella gestione degli indici va comunque ricordato che, trattandosi di strutture dati di appoggio alle tabelle, occupano spazio e richiedono un notevole carico di lavoro da parte del DBMS, non solo in fase di creazione ma soprattutto quando si aggiornano i valori del campo su cui l'indice è costruito.
È quindi evidente che la scelta dei campi da indicizzare riveste una notevole importanza nella fase di progettazione fisica di un database.
Certamente devono essere indicizzati i campi definiti chiave primaria di una tabella.
Quando conviene indicizzare un campo:

- conviene sempre indicizzare i campi che sono utilizzati in operazioni di giunzione tra tabelle, perché possono velocizzarne l'esecuzione. Andranno quindi indicizzati non solo i campi con funzione di chiave primaria ma anche quelli con funzione di chiave esterna;
- conviene indicizzare i campi ai quali si fa accesso con maggior frequenza nelle interrogazioni più utilizzate;
- conviene indicizzare i campi ai quali accede un maggior numero di interrogazioni;

Quando NON conviene indicizzare un campo:

- non conviene indicizzare i campi appartenenti a tabelle che non raggiungono grandi dimensioni, in quanto non apportano miglioramenti alle prestazioni del database;
- non conviene indicizzare campi il cui dominio è costituito da pochi valori. Infatti un indice può velocizzare un'interrogazione solo quando questa fornisce un numero modesto di dati, altrimenti non si apportano tangibili miglioramenti;
- non conviene indicizzare un campo che è regolarmente coinvolto da operazioni di aggiornamento, in quanto porterebbe a un notevole aumento del carico di lavoro per aggiornare gli indici;
- non conviene creare troppi indici se si hanno problemi di occupazione di memoria. Ricordiamo infatti che gli indici occupano spazio all'interno del database e più indici si gestiscono, più aumenta lo spazio richiesto per la loro memorizzazione.