

# Basi di dati 2017-2018

## Introduzione

La metodologia di progettazione è un insieme di strumenti che permette di suddividere in fasi lo sviluppo, fornisce strategie e criteri di scelta con modelli di riferimento, e garantisce generalità rispetto ai problemi da affrontare, qualità, efficienza (velocità) e facilità d'uso.

Una base di dati è una risorsa composta da una collezione di dati utilizzata per rappresentare l'informazione di interesse di diversi settori di una organizzazione, condivisa fra le varie applicazioni software, che permette di gestire attività diverse e multi-utente su dati condivisi ma rappresentati una sola volta, controllando le concorrenze. Il sistema software di gestione delle basi di dati si chiama DBMS (database management system), e garantisce le caratteristiche di affidabilità, sicurezza ed efficienza. È in grado di gestire richieste di utenti e applicazioni software relative a interrogazioni, transazioni, per accedere a una o più basi di dati che siano grandi, condivise e persistenti.

Il DBMS a sua volta utilizza due linguaggi: il DDL (data description language) per descrivere le strutture degli schemi, e il DML (data manipulation language) per esprimere le interrogazioni per trovare i dati e le istruzioni per aggiornarli.

Le elaborazioni delle basi di dati coinvolgono la CPU, la memoria centrale e la memoria di massa: le tempistiche sono dell'ordine di  $10^{-8}$  dalla CPU alla memoria centrale, e  $10^{-2}$  dalla CPU alla memoria di massa. La base di dati risiede nella memoria di massa.

Ci sono due livelli di indipendenza dei dati: indipendenza fisica, cioè l'indipendenza del livello logico ed esterno da quello fisico; indipendenza logica, del livello esterno da quello logico, aggiunte o modifiche agli schemi esterni non richiedono modifiche al livello logico e viceversa (trasparenza delle modifiche allo schema logico).

Lo sviluppo di una base di dati segue le seguenti fasi: studio di fattibilità, raccolta e analisi dei requisiti, progettazione, implementazione, validazione e collaudo, funzionamento. Queste fasi vengono raggruppate in:

- progettazione concettuale (schema concettuale), traduce i requisiti in una descrizione e crea una rappresentazione grafica. Partendo da una realtà in linguaggio naturale, si rappresentano le informazioni tramite tabelle (modello relazionale).
- progettazione logica, traduzione dello schema concettuale nel modello dati, che produce uno schema logico espresso nel DDL (alto livello);
- progettazione fisica, rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione (files).

## Modello E/R

Usato nella progettazione concettuale, permette di rappresentare i dati di interesse per l'applicazione in modo indipendente da ogni sistema DBMS.

Un'entità (occorrenza) è una classe di oggetti con proprietà comuni ed esistenza autonoma, della quale si registrano fatti specifici. Ogni entità ha un nome univoco ed è rappresentata graficamente da un rettangolo nel diagramma.

Le entità hanno identificatori univoci, insieme di proprietà che permettono di identificare univocamente le loro istanze (chiavi). Gli identificatori possono essere interni, cioè formati da attributi interni alla tabella, ed esterni; sono formati da uno o più attributi (pallino nero o pallini bianchi uniti con una linea che termina con un pallino nero).




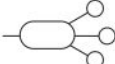



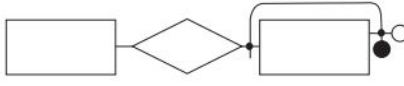


Un'istanza è un oggetto della classe che l'entità rappresenta (il dato vero e proprio, identificato univocamente), non viene rappresentato nello schema concettuale.

Un attributo è una proprietà locale di un'entità, definita su un dominio di valori; associa a ogni istanza di entità un valore nel corrispondente dominio (es. il nome di una persona specifica. Possono essere obbligatori o opzionali. Vengono specificati nella documentazione.

Analogamente, gli attributi di una relazione sono proprietà locali di una relazione: descrivono proprietà del legame logico e associano a ogni istanza di relazione un valore appartenente all'insieme detto dominio dell'attributo.

Gli attributi composti si realizzano raggruppando attributi di una stessa entità o relazione con affinità nel significato. Il dominio è un record, cioè un insieme di campi elementari (es. indirizzo, con via, numero civico, CAP). Gli attributi multivalore possono avere più valori per ogni istanza dell'entità associata. Gli attributi calcolati sono ottenuti da valori di altri attributi tramite calcoli.

Una relationship descrive un'azione o una situazione e stabilisce legami logici tra istanze di entità; possono essere tra più di due entità, e il numero di esse coinvolte determina il grado della relazione. È identificata univocamente da un sostantivo e rappresentata con un rombo. Non ha identificatori univoci propri: utilizza quelli delle entità collegate.

Construct	Graphical representation
Entity	
Relationship	
Simple attribute	
Composite attribute	
Cardinality of a	
Cardinality of an attribute	
Internal identifier	
External identifier	
Generalization	
Subset	

Una relazione di grado maggiore di 2 si dice n-aria, altrimenti binaria. Più relazioni possono essere definite anche sulle stesse entità.

Una relazione che coinvolge più volte un'unica entità è chiamata relazione ad anello (esempio: persona, amico) e può essere simmetrica, riflessiva, transitiva.

Un concetto verrà modellato come un'entità se le sue istanze sono significative indipendentemente da altre istanze, o potrebbe avere relazioni con altri concetti; come un attributo se non è significativo e rappresenta proprietà locali di un altro concetto; come una relazione se le sue istanze rappresentano insiemi di altre istanze.

Due entità possono avere istanze in comune: in questo caso la relazione si definisce IS-A o relazione di sottoinsieme, in cui ogni istanza dell'entità padre è anche istanza dell'entità figlia, ed eredita le sue relazioni (principio dell'ereditarietà). Le sottoclassi devono derivare da uno stesso criterio di classificazione delle istanze delle superclasse; le entità figlie non hanno identificatore interno, ereditano quello del padre. Non sempre è necessario inserire nel database il complemento di un'entità figlia.

Se l'entità padre generalizza diverse sottoentità indipendenti rispetto a un unico criterio si parla di generalizzazione (es. persona raggruppa studente e docente). Può essere completa (freccia nera), cioè che raggruppa tutte le entità con quel criterio, o non completa (bianca).

Un vincolo di cardinalità tra un'entità e una relazione esprime un limite minimo e un limite massimo di istanze della relazione a cui può partecipare ogni istanza dell'entità. Il limite massimo non è necessariamente infinito.

Cardinalità delle relazioni:

- 0/1 per la cardinalità minima, dove 0 significa partecipazione opzionale (valori null), 1 significa partecipazione obbligatoria;
- 1/n per la cardinalità massima, dove 1 significa che le istanze partecipano al più una volta, n significa che le istanze partecipano un numero qualsiasi (non costante ma senza limite) di volte alla relazione;
- 1 : 1 quando le cardinalità massime delle entità sono <1, 1>, l'identificatore esterno si può mettere solo con la cardinalità 1 : 1 (es. una sala in un cinema);
- 1 : n (uno a molti) quando le cardinalità massime delle entità sono <1, n>;
- 0 : n quando non c'è un vincolo di cardinalità;
- n : n (molti a molti) quando le cardinalità massime delle entità sono <n, n>.

## **Progettazione concettuale**

La progettazione concettuale è la prima fase della progettazione di una base di dati: vengono raccolti i requisiti degli utenti (in linguaggio naturale, quindi ambiguo) e a partire da essi, organizzandoli e analizzandoli, viene creata una rappresentazione astratta, cioè uno schema concettuale. Ci sono diverse strategie di progettazione concettuale, ognuna con vantaggi e svantaggi. Dallo schema concettuale vengono realizzati il modello relazionale, il modello ER e da questi è effettuata la progettazione fisica del database.

Lo schema concettuale deve rispettare alcune qualità:

- Correttezza, rispetto ai requisiti richiesti e al modello, la base di dati dev'essere funzionale e progettata secondo le regole di entità e relazioni;
- Completezza: tutti i requisiti devono essere rappresentati nello schema;
- Pertinenza, non ci devono essere concetti che non compaiono nei requisiti;
- Minimalità, non ci devono essere più concetti che rappresentano gli stessi requisiti (per evitare ridondanza tra le entità e i loro attributi);
- Leggibilità, lo schema deve rappresentare i requisiti in modo comprensibile, grafica (del diagramma) e concettuale (scelta dello schema più compatto e semplice).

L'analisi dei dati è la raccolta dei requisiti per eliminare le ambiguità, rendere comprensibile i termini utilizzati e la progettazione concettuale. I requisiti possono provenire dagli utenti, dalla documentazione fornita o da quella esistente come i regolamenti e le normative.

L'acquisizione dei dati non è sempre facile: utenti diversi possono fornire informazioni diverse, e avere visioni più o meno dettagliate dell'insieme, e occorre rappresentare il riferimento tra termini nel contesto. È necessario effettuare periodicamente verifiche di comprensione, mancanza di incompletezze e coerenza, e richiedere eventuali chiarimenti su definizioni e classificazioni.

Un maggior livello di strutturazione rende più semplice la progettazione del modello concettuale. Bisogna scegliere il livello di astrazione più corretto, standardizzare le fasi e identificare le operazioni richieste dai dati. Per organizzare termini e concetti (es. omonimi) si utilizza un glossario che individua sinonimi e riferimenti tra termini.

I requisiti vengono strutturati in gruppi di frasi omogenee, in modo da semplificare la progettazione concettuale: trovare entità, relazioni e attributi risulta più facile. Le frasi sono di carattere generale, relative a categorie di utenti nel database oppure alle attività da gestire.

I design pattern sono strutture utilizzate comunemente per rappresentare le basi di dati. Il loro scopo è uniformare la progettazione concettuale mantenendo l'integrità, la velocità e l'adattabilità ai cambiamenti delle basi di dati. Si usa un procedimento di reificazione, creazione di un modello di dati basato su un concetto astratto predefinito.

Per lavorare in modo strutturato con numerose specifiche sono state adottate strategie. Le strategie sono semplici quando danno indicazioni per scegliere tra attributo, entità, relazione, IS-A o generalizzazione (scelta dei singoli concetti, basandosi sul modello ER); complesse se esprimono un'idea su come percorrere il processo di progettazione.

Top-down: da un piccolo schema astratto si arriva a schemi dettagliati (raffinamenti). Non è sempre ottenibile in casi complessi perché richiede una visione globale ma non necessita particolari ristrutturazioni, e in ogni momento il progettista rappresenta sempre l'intero insieme dei requisiti.

Bottom-up: da tanti schemi con pochi concetti e proprietà comuni si ottiene la loro integrazione in uno schema completo. Permette una ripartizione delle attività e della progettazione tra diversi gruppi, ma richiede ristrutturazioni.

Inside-out: propagandosi da un concetto verso tutti gli altri. Non richiede fase di integrazione ma a ogni passo vanno esaminate le specifiche, quindi è poco pratico per casi con numero di specifiche molto elevato.

Mista, un insieme delle precedenti strategie, permette di unificare i vantaggi delle altre. È una strategia ibrida in cui, prima di ogni altra scelta progettuale, si individuano i concetti principali (più importanti) realizzando uno schema scheletro; poi essi vengono organizzati in uno schema concettuale.

Dopo aver individuato lo schema scheletro si può seguire una strategia top-down e raffinare/aggiungere concetti fino al raggiungimento della qualità e completezza richiesta, oppure mista, che decompone i requisiti in sottoschemi e li integra in uno schema complessivo facendo riferimento allo schema scheletro.

## **Modello relazionale**

Un modello di dati è un insieme di concetti per organizzare i dati e descriverne la struttura. La componente fondamentale sono i meccanismi di strutturazione, o costruttori di tipo.

Ci sono due tipi principali di modelli: logici (relazionale), basati su strutture astratte e sono utili per l'organizzazione dei dati, e concettuali che permettono di rappresentare i dati in modo indipendente dal sistema e descrivere i concetti (ER). Lo schema logico viene poi rappresentato in uno schema fisico attraverso file.

Il modello relazionale è formato da un insieme di tabelle, composte da righe e colonne; la prima riga rappresenta i campi della tabella, le successive le istanze. Il suo costruttore è la relazione, che permette di definire insieme di record (righe) omogenei. La struttura è basata su valori: l'utente finale vede gli stessi dati (tabelle) dei programmatori, le strutture fisiche sono indipendenti e i dati sono portabili da un sistema all'altro.

Una relazione è rappresentata da una tabella dove le righe rappresentano specifici record e le colonne corrispondono ai campi dei record. È basata sui valori, permettendo una maggiore indipendenza dei dati. Le colonne non sono interscambiabili.

In ogni base di dati esistono lo schema (invariato nel tempo) che ne descrive la struttura, e l'istanza (valore attuale) che rappresenta l'aspetto estensionale e può cambiare rapidamente.

Una n-upla su un insieme di attributi è una funzione che associa a ciascun attributo nell'insieme un valore del dominio dell'attributo. Un'istanza di relazione è un insieme di ennuple sull'insieme. Un insieme di relazioni è un'istanza di base di dati.

Le relazioni sono di tre tipi:

- Relationship, rappresenta una classe di fatti nel modello ER;

- Relazione matematica, rappresentata con un sottoinsieme del prodotto cartesiano (dominio). I campi delle n-uple sono ordinati ma non c'è ordinamento tra le n-uple;
- Relazione su cui si basa il modello relazionale dei dati.

I domini devono essere semplici: non si possono ridurre a strutture più semplici (array, liste). Un dominio non ulteriormente decomponibile (atomico) è in prima forma normale.

La seconda forma normale invece stabilisce, oltre al database in prima forma normale, che gli attributi che non sono chiavi devono dipendere dall'insieme delle chiavi (e non da una sola di esse).

Una base di dati è in terza forma normale se è in seconda forma normale e tutti gli attributi che non sono chiavi dipendono esclusivamente dall'insieme delle chiavi, e non da qualche altro attributo.

A ciascun dominio si associa un nome (attributo). I valori di ogni colonna sono tra loro omogenei, le righe sono diverse tra loro. L'ordinamento tra righe e colonne è irrilevante.

I riferimenti fra dati in relazioni diverse sono rappresentati per mezzo di valori dei domini che compaiono nelle n-uple. Le relazioni sono rappresentate con un nome (istanza) e il relativo insieme di attributi, e l'insieme delle relazioni è contenuto nello schema della base di dati.

Le strutture nidificate contengono altre strutture al loro interno. Generalmente la stessa realtà può essere rappresentata in modi equivalenti, cioè che contengono le stesse informazioni, ma per evitare ridondanza è bene definire ogni entità rilevante separatamente. Le tabelle hanno relazioni gerarchiche, e attributi precisi.

L'integrità dei dati rappresenta la possibilità della situazione reale descritta nella base di dati. I dati devono essere coerenti con la realtà, le chiavi esterne devono essere coerenti con gli attributi delle altre tabelle nelle relazioni.

Tutte le istanze devono soddisfare il vincolo di integrità, una funzione booleana che associa a ogni istanza la sua correttezza. I vincoli permettono una rappresentazione più accurata della realtà e sono utili per la qualità dello schema nella progettazione. Sono intrarelazionali (all'interno di una relazione) o interrelazionali (definiti tra due o più relazioni); definiti su singoli valori, su n-uple o relazioni. L'univocità dei valori chiave deve sempre essere rispettata.

Una chiave (formalmente) è un insieme di attributi che identifica le n-uple di una relazione. Un insieme di attributi è superchiave per una relazione se la relazione non contiene due n-uple distinte con chiave uguale. Se la superchiave non contiene altre superchiavi (per esempio è un solo attributo) allora è anche una chiave o superchiave minimale. La superchiave deve contenere la chiave (es. codice fiscale e nome) ma non deve contenere valori opzionali (null).

I vincoli sono proprietà dello schema: sono valide le istanze che soddisfano tutti i vincoli. Una relazione non può contenere n-uple distinte ma uguali; ogni relazione ha almeno una chiave.

Null (estensione del dominio) significa nessuna informazione: i campi possono essere opzionali. Not null significa che il campo è obbligatorio. Servono per rappresentare anche dati imprecisi o mancanti pur rispettando le strutture rigide del modello relazionale.

I valori nulli sono sconosciuti, inesistenti o senza informazione (non distinti dal DBMS).

Le chiavi non possono avere valori nulli, altrimenti la n-upla non sarebbe identificata univocamente. La notazione della chiave primaria è generalmente la sottolineatura.

Ogni entità deve avere almeno un identificatore (chiave); le relazioni non hanno identificatore, sono identificate dalle entità collegate a esse (occorrenze, coppie di chiavi).

La chiave può essere univoca all'interno della relazione, ma non del database (es. la stessa matricola corrispondere a più università), e viene indicato come identificatore esterno nel modello ER.

L'integrità referenziale esprime la proprietà per cui informazioni con stesso significato in relazioni diverse sono correlate, rispettando la proprietà per cui non è possibile che un oggetto della realtà (e quindi un valore), contemporaneamente:

- Sia rappresentato nella sua relazione con un altro oggetto;
- Non sia rappresentato e identificato nella tabella che lo descrive nativamente.

Un vincolo di integrità referenziale (foreign key, chiave esterna) tra un insieme di attributi di una relazione  $R_1$  e un'altra relazione  $R_2$  impone ai valori degli attributi (che coincidono con la chiave primaria) in  $R_1$  di comparire come valori della chiave primaria di  $R_2$ . Questo vincolo esprime una proprietà strutturale degli schemi di relazione e delle istanze corrispondenti in una base di dati. I vincoli sono intrarelazionali o interrelazionali.

- Dominio (es. voto maggiore di 30L), violazione del dominio dell'attributo;
- Tupla (es. 28L), violazione all'interno di una tupla;
- Chiave (ID duplicato), violazione all'interno di un'istanza di relazione;
- Referenziale, violazione inter relazione (nella corrispondenza).

Se una n-upla viene cancellata violando l'integrità referenziale è necessario effettuare azioni compensative come l'eliminazione in cascata, il rifiuto dell'operazione o l'introduzione di valori nulli. L'ordine degli attributi è significativo.

L'esistenza delle chiavi garantisce l'accessibilità a ciascun dato della base di dati. La chiave primaria è usata per stabilire corrispondenze tra le tabelle, se nessun attributo può averne il ruolo se ne definisce uno apposito. Nel modello relazionale le chiavi sono sottolineate, i vincoli di integrità referenziale sono rappresentati da frecce.

In generale la foreign key e la chiave primaria possono includere attributi con nomi diversi, ma possono far parte della stessa relazione su insiemi di attributi diversi. I vincoli di integrità non si riferiscono necessariamente a chiavi primarie.

## **Progettazione logica**

La progettazione logica è la fase tra la progettazione concettuale e quella fisica: traduce uno schema dal modello ER al modello relazionale. L'input è costituito dallo schema concettuale, le informazioni sul carico applicativo (interrogazioni, transazioni e operazioni con relative frequenze) e il modello logico: questi sono utilizzati per produrre uno schema logico corretto ed efficiente, e la relativa documentazione.

Correttezza significa rappresentare la stessa realtà, efficienza implica il ridotto uso di risorse elaborative (tempo di esecuzione e spazio di memoria) per interrogazioni e transazioni.

Lo spazio viene misurato con una tavola dei volumi, che descrive il numero delle istanze di entità e relazioni; il tempo con una tavola degli accessi, che descrive il numero di istanze di entità e relazioni (lettura o scrittura) visitate dall'operazione.

Alcune strutture dello schema concettuale non sono direttamente rappresentabili nello schema logico. Ristrutturazione del modello ER: eliminazione dallo schema di tutti i costrutti che non possono essere direttamente rappresentati nel modello logico target. Lo scopo è rendere semplice e ottimizzata la traduzione del modello relazionale.

- Eliminazione delle ridondanze;
- Eliminazione delle generalizzazioni;
- Partizionamento o accorpamento di entità e associazioni;
- Scelta degli identificatori primari (una sola chiave).

Una ridondanza in uno schema ER è un concetto dello schema derivabile da altri. Le principali ridondanze individuabili sono attributi derivabili (calcolabili) e relazioni derivabili, cioè con istanze calcolabili con composizione di altre relazioni. Con l'eliminazione delle ridondanze, il carico applicativo e lo spazio si riducono.

Gli attributi che si possono ottenere tramite query (come il numero di abitanti di una città o il salario lordo) vengono eliminati, perché generalmente il costo in assenza di ridondanza sulla tavola degli accessi è trascurabile.

La partizione di un'associazione avviene quando per esempio alcuni campi non cambiano, e altri sono mutabili. Per ottimizzare le query si creano due tabelle con una relazione 1 : 1.

Le generalizzazioni vengono modificate portando tutte le entità figlie (e i relativi attributi) come attributi nell'entità padre (es. tipo, oppure un attributo per ogni tipo). La cardinalità può essere (0, 1) perché sono ammessi valori null.

Altre possibilità sono il trasferimento degli attributi dell'entità padre nei figli, con la stessa chiave e le stesse relazioni (raddoppiate), oppure la sostituzione delle generalizzazione con relazioni. La scelta tra queste regole viene fatta in base alla loro convenienza rispetto agli accessi e alle operazioni.

I partizionamenti e gli accorpamenti sono ristrutturazioni effettuate per rendere più efficienti le operazioni. Gli accessi si riducono separando gli attributi di un concetto che vengono acceduti separatamente da molte operazioni e raggruppando gli attributi di concetti diversi che sono acceduti insieme da molte operazioni.

- Partizionamento verticale di entità (es. dati anagrafici e lavorativi);



- Eliminazione di attributi multivalore;
- Accorpamento di entità/relationship;
- Partizionamento orizzontale di relationship (es. squadra corrente e passata).

La scelta degli identificatori primari è vincolata dall'assenza di opzionalità degli attributi che costituiscono l'identificatore, quindi la scelta dev'essere su attributi strutturalmente semplice (pochi) utilizzati nelle operazioni più frequenti e importanti. Se non esistono attributi del genere viene introdotto un codice generato appositamente.

Le entità sono tradotte in relazioni definite sugli stessi attributi e aventi come chiave primaria l'identificatore principale, per individuare le singole tuple.

Le relazioni molti a molti hanno come chiave primaria gli identificatori delle entità coinvolte, ed eventuali attributi propri. Tra le relazioni sono definiti vincoli di integrità referenziale. Associazioni definite sulla stessa entità diventano relazioni in modo analogo.

Per le relazioni binarie, ci saranno tante relazioni quante sono le entità; una relazione per l'associazione con chiave data dall'unione degli identificatori, e tanti vincoli di integrità referenziale quante le entità. Le cardinalità 1 : 1 vengono fuse.

La partecipazione opzionale viene rappresentata con valori nulli.

Riassumendo, le relazioni nel modello ER si traducono in questo modo:

- 1 : 1, trasformando le due entità e la relazione in un'unica entità, oppure accorpando la relazione in una delle due entità;
  - (0, 1) : (1, 1), accorpando gli attributi nel lato senza valori nulli;
  - (0, 1) : (0, 1), gestibili anche con tre relazioni;
- 1 : n, la relazione tra le entità va tenuta, con chiave primaria l'unione delle chiavi primarie delle entità più eventuali altri attributi;
- (1, 1) : (1, n), aggiungendo la chiave primaria dell'entità lato n agli attributi dell'entità del lato 1, in modo da poter eliminare la relazione;
  - In presenza di una chiave esterna si procede analogamente, ma l'attributo aggiunto dev'essere parte della chiave primaria;
  - Se l'entità dalla quale viene presa la chiave esterna ha una chiave esterna a sua volta, vanno inserite entrambe (come chiave primaria);
- (0, n), si può assumere che gli attributi dell'entità restino invariati;
- Relazione ternaria (o più), con qualsiasi cardinalità, si presuppone che tutte le entità partecipino, quindi ogni chiave è chiave nella relazione;
- Relazione (n : m) definita sulla stessa entità, con una coppia di chiavi primarie prese entrambe dall'entità originale;
  - (1, 1) : (1, n) sulla stessa entità, con una sola tabella;
- Relazione ternaria con (1, 1), accorpabile nell'entità lato (1, 1) con le chiavi delle altre entità (1, n);
- Più relazioni tra due entità, accorpabili in un'entità ma è necessario rinominare le chiavi in modo da mantenere l'univocità;

- Generalizzazione, accorpendo i figli al padre con un attributo tipo e cardinalità minima 0, accorpendo il padre ai figli, oppure aggiungendo una relazione tra le entità;
  - $(1, n) : (1, n)$  in una generalizzazione, tradotta con una relazione ad anello (in caso i figli vengano accorpati) con cardinalità  $(0, n) : (0, n)$ ;
  - Relazione tra padre e figli, aggiungendo la chiave esterna del padre ai figli, con cardinalità  $(1 : 1), (0 : 1)$ .

## Data Base Management System

È un sistema software che facilita il processo di definizione, costruzione e manipolazione di una base di dati, garantendone la persistenza e consentendo l'accesso concorrente ai dati in essa contenuti da parte di utenti e applicazioni.

Le tipologie più usate di DBMS sono l'Object-Oriented DBMS e NoSQL, struttura a tabelle senza relazioni. Ogni espressione dell'algebra relazionale può essere tradotta in SQL.

La coerenza dei dati viene mantenuta rispetto alla struttura e nel tempo, facilitando l'accesso delle applicazioni ai dati e il controllo delle transazioni rispettando i principi ACID (atomicità, coerenza, isolamento e durabilità).

La quantità dei dati gestiti è grande, i dati sono persistenti e condivisi. Esiste una parte server e una parte client, e meccanismi di ottimizzazione delle richieste e affidabilità dei dati.

MySQL offre crittografia dei dati, gestione multiutente, funzionalità di backup e ripristino, funzionalità di progettazione, gestione e ricerca. Si sviluppa su tre livelli: servizi di rete, query e motori di storage.

Lo storage engine è transazionale: InnoDB consente il full-text search, l'utilizzo di chiavi esterne nel rispetto dell'integrità referenziale e il lock a livello di record.

Un singolo processo server si occupa di ascoltare su una porta socket (3306) e lanciare i thread per le sessioni utente. I database sono directory che contengono uno o più file per ogni tabella.

MySQL prevede tre sotto parti SQL:

- DCL (Data Control Language), come `GRANT`, `REVOKE`, `SHOW TABLES`;
- DDL (Data Definition Language), come `CREATE`;
- DML (Data Manipulation Language), come `SELECT`.

## SQL-2 e DDL

I linguaggi delle basi di dati servono per istruzioni di create, read, update e delete (CRUD).

SQL è un linguaggio standard per la definizione e l'interrogazione di database relazionali. Solitamente i DBMS implementano in parte lo standard e lo estendono per supportare sintassi differenti e funzionalità specifiche.

SQL adotta la logica a 3 valori dell'algebra relazionale: nasce dall'esigenza di gestire i valori nulli, e introduce Unknown nelle tabelle di verità per poter fare confronti su campi nulli.

Notazione SQL:

- < x > usate per indicare un termine x;
- [ x ] indicano che il termine x è opzionale;
- { x } indicano che x può essere ripetuto zero o più volte;
- | separa opzioni alternative (or).

Uno schema di basi di dati è una collezione di oggetti, ogni schema ha un nome e un proprietario ed è definito dalla sintassi. Se il nome dello schema o del proprietario vengono omessi, di default sono il nome dell'utente che ha lanciato il comando CREATE. È possibile assegnare delle autorizzazioni.

Una tabella è costituita da una collezione ordinata di attributi e da un insieme (eventualmente vuoto) di vincoli. Può esserci la duplicazione di righe.

L'istruzione CREATE definisce uno schema di relazione e ne crea un'istanza vuota. Per ogni attributo va specificato il dominio, un eventuale valore di default ed eventuali vincoli a livello di tabella o tra più attributi.

I domini di default in SQL-2 sono i valori bit, i caratteri, i valori numerici esatti o approssimati, data/ora e intervalli temporali. Ulteriori domini possono essere definiti dagli utenti.

Il tipo BIT può essere di lunghezza fissa o variabile, se la lunghezza non è specificata corrisponde a 1 (flag, usato nei valori booleani).

Il tipo char può essere di lunghezza fissa o variabile (massima), vengono definiti con CHARACTER o VARCHAR.

I valori numerici esatti rappresentano interi o decimali in virgola fissa, con una precisione. Si definiscono con INTEGER, SMALLINT, NUMERIC e DECIMAL (con precisione maggiore).

I valori numerici approssimati sono FLOAT, DOUBLE PRECISION e REAL; composti da una mantissa e un esponente in un range.

Data e ora descrivono informazioni temporali, ciascuno di questi domini è strutturato e decomponibile in un insieme di campi (anno, mese, giorno, minuti, secondi). Esiste un valore TIMESTAMP che rappresenta la combinazione di data e ora.

L'intervallo temporale è rappresentato con INTERVAL, un range di partenza e un eventuale range di arrivo, indicati con X TO Y dove X, Y sono campi di data/ora (tranne mesi e giorni).

I valori BLOB servono per immagazzinare un numero elevato di byte (files).

Un vincolo è una regola che specifica delle condizioni sui valori di un elemento dello schema del database. Un vincolo può essere associato a una tabella, a un attributo o a un dominio. Sono intrarelazionali o interrelazionali.

Esempi di vincoli sono NOT NULL, DEFAULT, PRIMARY KEY, UNIQUE per valori unici che non siano chiavi primarie e CHECK per condizioni complesse (maggiore, minore). I valori

NULL sono utilizzabili in caso di valore sconosciuto, inesistente oppure senza informazioni; può comparire su diverse righe senza violare vincoli.

AUTO\_INCREMENT serve per dichiarare una colonna di tipo intero a incremento automatico. È possibile settare un valore iniziale.

I domini definiti dagli utenti sono costruiti tramite la primitiva CREATE DOMAIN e le relative informazioni. L'utilità è di associare dei vincoli a un nome di dominio, per ripetere la stessa definizione di attributo su diverse tabelle.

I vincoli interrelazionali coinvolgono più relazioni e tabelle. Possono essere definiti attraverso i costrutti sintattici REFERENCES, FOREIGN KEY e CHECK. Si devono riferire ad attributi della tabella padre che costituiscono una chiave; devono essere sempre presenti.

Se si omettono gli attributi di destinazione, vengono assunti quelli della chiave primaria; quando si hanno più attributi da riferire, si utilizza sempre FOREIGN KEY.

A seguito di una violazione, il comando di aggiornamento viene rifiutato segnalando l'errore all'utente. Per i vincoli di integrità referenziale si possono introdurre violazioni operando sulla tabella interna o esterna; se le modifiche (cancellazioni o update) sono sulla tabella padre vengono assegnati valori nulli o di default, o la modifica viene propagata o impedita.

A fini diagnostici e di documentazione è spesso utile sapere quale vincolo è stato violato, a tale scopo è possibile associare dei nomi ai vincoli.

Per modificare gli schemi si utilizzano ALTER e DROP: RESTRICT impedisce DROP se gli oggetti comprendono istanze non vuote, CASCADE applica DROP agli oggetti collegati (potenziale reazione a catena).

L'azione da eseguire è definita con i costrutti ON UPDATE e ON DELETE.

## Algebra relazionale

L'algebra relazionale è un insieme di operatori e di proprietà che vengono applicati su un determinato insieme di valori: gli operatori producono relazioni e possono essere composti tra loro. L'algebra relazionale viene applicata al modello relazionale. Le operazioni sono unarie o binarie (operazioni insiemistiche e join).

Operatori insiemistici: unione, intersezione e differenza, che sono applicabili solo su relazioni definite sugli stessi attributi, quindi potrebbe esserci necessità di rinominare colonne.

- Unione: tutte le n-uple di una relazione oppure dell'altra, indicata con  $\cup$ ;
- Intersezione: le n-uple che appartengono a entrambe le relazioni, indicata con  $\cap$ ;
- Differenza: n-uple che appartengono alla prima relazione ma non alla seconda, con  $-$ .

Ridenominazione  $\rho$ : modifica lo schema lasciando inalterata l'istanza della relazione. È utile per definire lo stesso dominio nelle operazioni insiemistiche.

La sintassi è `REN NuovoAttributo ← Vecchio (Tabella)`. Se gli attributi da cambiare sono più di uno si separano con virgole: `REN Nuovo1, Nuovo2 ← Vecchio1, Vecchio2 (Tabella)`.

Selezione  $\sigma$ : operatore monadico, permette di selezionare il sottoinsieme delle n-uple che soddisfano una certa condizione. Produce un risultato che ha lo stesso schema dell'operando (è un sottoinsieme).

La sintassi è SEL Condizione (Operando) e le condizioni possono essere combinate con AND, OR o NOT.

Le interrogazioni non possono dare un risultato sotto forma di valore booleano in presenza di un campo NULL (tranne nel caso in cui il controllo sia per valori NOT NULL o IS NULL), quindi le corrispondenti n-uple non vengono mai incluse nell'output.

Proiezione  $\pi$ : decomposizione verticale, permette di selezionare determinate colonne da una tabella. È definita su una parte degli attributi dell'operando, la cardinalità è il numero delle n-uple. Le n-uple duplicate vengono eliminate.

La sintassi è PROJ Attributi (Operando).

Join  $\bowtie$ : permette di correlare dati che si trovano in relazioni diverse. Ha dominio sul prodotto cartesiano delle tabelle. Tutte le coppie di n-uple sono combinabili tra di loro.

La theta-join è preceduta da una SEL, con sintassi SEL condizione ( $R_1 \bowtie R_2$ ), con qualsiasi condizione arbitraria..

L'equi-join è un tipo di theta-join con operatore di uguaglianza espresso nella condizione.

Il join naturale (binario) unisce n-uple a partire da una n-upla di ognuno degli operandi. Le n-uple devono avere gli stessi valori negli attributi comuni. Il join naturale rimuove le colonne duplicate dal risultato.

Quando ogni n-upla delle tabelle contribuisce al risultato si parla di join completo, altrimenti di join non completo.

Il join esterno estende con valori nulli le n-uple escluse dal join naturale. Può essere a destra, sinistra o completo: il join a sinistra mantiene le n-uple del primo operando, a destra quelle del secondo operando, completo di entrambi.

Il self-join unisce una tabella con se stessa, viene usato nelle query dove è necessario contare attributi (è necessaria la ridenominazione).

Una vista è una relazione derivata a partire dalle relazioni definite nello schema delle basi di dati. La derivazione è espressa tramite una normale interrogazione SQL ed eredita i nomi degli attributi citati nella SELECT. Le interrogazioni sulle viste vengono eseguite sostituendo alla vista la sua definizione.

La sintassi è <Nome vista> = interrogazione SQL. Le viste materializzate sono relazioni derivate memorizzate nella base di dati, immediatamente disponibili ma ridondanti; le viste virtuali vengono ricalcolate ogni volta.

## **DML**

SELECT seleziona una specifica colonna della tabella specificata nella WHERE. La WHERE definisce condizioni sull'output, con eventuali operatori logici.

Per unire due campi nella SELECT si usa tabella1.campo + ' ' + tabella2.campo (in questo modo i due campi vengono inseriti in una colonna sola separati da uno spazio).

Quando in SELECT ci potrebbero essere campi non univoci è utile inserire anche una chiave per garantire l'univocità delle righe.

L'istruzione JOIN in SQL ha la sintassi [tipo] JOIN ON condizione; se il tipo non è specificato viene eseguita la INNER JOIN.

L'operatore OUTER JOIN aggiunge al risultato le tuple che non hanno partecipato al JOIN, completandole con NULL. Esistono tre tipologie di OUTER JOIN, che sono LEFT, RIGHT e FULL (funzionano allo stesso modo rispetto all'algebra relazionale).

SELF JOIN permette di duplicare una singola tabella (es. trovare i nonni da una tabella con genitore e figlio).

Quando il JOIN dev'essere eseguito su più tabelle (oppure per indicarlo implicitamente) si utilizza WHERE con la condizione della JOIN ed eventuali AND.

DISTINCT seleziona una volta sola eventuali risultati ripetuti. IN funziona come un OR in un sottoinsieme di valori. EXISTS restituisce TRUE se esiste almeno una n-upla.

AS (alias) permette di rinominare le tabelle di output alla SELECT. Nella FROM è possibile duplicare tabelle e assegnare loro nomi con AS implicito (es. persone p, persone f).

ORDER BY [colonna] serve per dare un ordinamento ASC (ascendente) o DESC (discendente) alle colonne della tabella, secondo una specifica colonna.

Gli operatori di aggregazione sono COUNT, SUM, MAX, MIN, AVG, GROUP BY.

COUNT(\*) conta il numero di tuple nella tabella, se ha valore nullo la n-upla viene omessa.

Non è possibile utilizzare gli operatori di aggregazione violando la cardinalità delle tabelle (es. nome e cognome di MAX(stipendio), seleziona comunque tutti i nomi e i cognomi).

Non è possibile utilizzare gli operatori di aggregazione in WHERE, perché WHERE confronta n-upla per n-upla e il valore richiesto potrebbe non essere nei campi della SELECT.

GROUP BY viene usata con le funzioni di aggregazione per raggruppare i risultati. Definisce gruppi omogenei di tuple, specificando una o più colonne su cui i valori sono raggruppati secondo i valori uguali. Per contare secondo n attributi è sufficiente inserirli nella GROUP BY.

La GROUP BY può essere utilizzata solo se include tutte le colonne della SELECT tranne quelle su cui è stata definita una funzione di aggregazione.

HAVING verifica condizioni (espressioni booleane) su gruppi di tuple, intervenendo sugli operatori di aggregazione (es. HAVING COUNT > 50). La WHERE permette la verifica solo riga per riga. HAVING viene applicata sul risultato della GROUP BY.

Per concatenare query SQL formando un'unica query vengono usati gli operatori insiemistici. La sintassi è SELECT query [ < UNION | INTERSECT | EXCEPT > [ALL] SELECT

query }.

Non c'è la possibilità di mantenere i duplicati a meno che non ci sia all.

Per semplificare le query si possono utilizzare le query annidate: vengono usate per determinare uno o più valori di confronto generalmente nella clausola WHERE.

Nella clausola WHERE possono comparire predicati che confrontano un attributo, insieme a < any | all > che sono vere rispettivamente se almeno una riga soddisfa il confronto, e tutte le righe soddisfano il confronto. ANY con uguaglianza funziona come IN.

Gli operatori di confronto si possono usare solo se la subquery restituisce non più di una tupla (es. la media).

È possibile selezionare più di una colonna tramite una sotto-interrogazione: in tal caso è necessario usare un costruttore di tupla, apponendo parentesi tonde alla lista delle colonne da confrontare nella WHERE.

Interpretazione semplice: l'interrogazione interna viene eseguita prima di quella esterna. Il risultato può essere salvato in una tabella temporanea.

Query correlate: la query più interna fa riferimento a una variabile definita nella query esterna, quindi a ciascuna riga del prodotto cartesiano delle tabelle viene applicata WHERE. Le sotto-query vengono eseguite una volta per ciascuna n-upla dell'interrogazione esterna.

Per poter riferire le colonne delle n-uple della query esterna si fa uso degli alias (AS); seleziona n-uple dalla stessa relazione definita in entrambe le query. Se un'interrogazione possiede interrogazioni nidificate allo stesso livello, le variabili di una non possono essere usate nell'altra.

In una subquery non si possono usare operatori insiemistici; può comparire solo come operando destro in un predicato, e possono comparire anche in SELECT o HAVING. In una subquery scalare non ci possono essere GROUP BY e HAVING.

Una vista è una tabella che contiene il risultato di una query salvata come una tabella virtuale. Le viste sono utili per mantenere le informazioni private e per scrivere interrogazioni più chiare, per esempio inserendo output di join multiple.

Le viste in SQL si creano con CREATE VIEW con il nome della vista seguito da AS e la query relativa. Il nome della vista dev'essere univoco (e generalmente significativo); la vista deve contenere un eventuale insieme di attributi oppure clausole di raggruppamento.

È possibile definire viste tramite interrogazioni che contengono espressioni o funzioni. Quando si specifica un'interrogazione su una vista, il sistema sostituisce nell'interrogazione la vista con la sua definizione. La query viene rieseguita ogni volta.

Le viste si utilizzano con funzioni di aggregazione in cascata (operazione illegale in SELECT per questioni di cardinalità). La vista deve sempre essere definita separatamente prima di essere utilizzata, quindi non è corretto inserirla nel FROM.

## Query

mysql.server start

Avvio del server MySQL.

mysql -u root -p

Password nulla di default, avvio dell'ambiente MySQL.

SHOW DATABASE nome;

Elenco dei database.

USE nomedb;

Elenco delle tabelle.

CREATE DATABASE nome;

Creazione un database con il nome specificato.

DROP DATABASE nome;

Eliminazione dell database corrispondente.

```
CREATE TABLE nome (  
    attributo1 dominio [default] [vincoli],  
    [attributo2 dominio [default] [vincoli]],  
    [altri vincoli]);
```

Crea una tabella con un insieme di attributi e i rispettivi vincoli.

```
CREATE DOMAIN nome AS tipo (  
    DEFAULT valore  
    NOT NULL);
```

Creazione di un dominio.

```
ALTER DOMAIN nome <  
    SET DEFAULT valore |  
    DROP DEFAULT |  
    ADD CONSTRAINT vincolo |  
    DROP CONSTRAINT vincolo >;
```

Modifica di dominio.

```
ALTER TABLE nome <  
    ALTER COLUMN colonna |  
    SET DEFAULT default |  
    DROP COLUMN colonna |  
    ADD COLUMN colonna |  
    DROP CONSTRAINT vincolo |  
    ADD CONSTRAINT vincolo >;
```

Modifica di tabella.

```
DROP <schema, dominio, tabella, view> nome  
    [RESTRICT | CASCADE];
```

Eliminazione di oggetti DDL.



```
SELECT colonne [AS nome], SUM/COUNT/...  
FROM tabella1 JOIN tabella2 ON tabella1.campo1 = tabella2.campo2  
WHERE condizione [AND/OR/NOT condizione]  
GROUP BY colonne  
HAVING condizione  
ORDER BY colonne;  
Interrogazione di un database.  
  
LOAD DATA INFILE file.csv INTO TABLE tabella;  
Importazione di un file csv.
```