Dato: singola informazione codificabile o codificata; unità elementare dell'informazione.
Sistema informativo: componente di un'organizzazione che gestisce le informazioni di interesse
Ogni organizzazione ha un sistema operativo, eventualmente non esplicitato in struttura.
Gestione delle informazioni:

- Raccolta, acquisizione
- Archiviazione, conservazione
- Elaborazione, trasformazione, produzione
- Distribuzione, comunicazione, scambio

Il sistema informativo è un sistema atto a organizzare e gestire i dati in maniera tale da permettere la loro facile elaborazione/ memorizzazione per ottenere informazioni.
Sistema informativo: gestisce le informazioni con tecnologia informatica.
All'interno del sistema informativo, la collezione di dati è chiamata Database, il cui compito è di memorizzare i dati e di rappresentare le relazioni tra di essi.
Il software all'interno del sistema informativo atto a gestire i dati è detto Sistema di Gestione della base di dati o Database Management System.
Ogni programma ha acceso al file system gestito dal sistema operativo per creare uno o più file.
Ogni file è un insieme di registrazioni (record) all'interno dei quali sono memorizzati i dati elementari. La condivisione di dati tra più programmi può essere fatta tramite l'uso di file condivisi.
I file possono avere diversi formati incompatibili tra di loro, i programmi si devono adeguare a diverse convenzioni anche a distanza di parecchio tempo. Ciò rende la condivisione dei dati attraverso applicazioni differenti difficoltosa.
I dati se non memorizzati su file condivisi sono replicati con spreco di risorse di memorizzazione e possibili problemi legati a inconsistenza.
DBMS (Database Management System): adozione di un sistema di gestione dei dati indipendente dalle applicazioni e specializzato in tale funzione. I dati sono gestiti da un DBMS che offre un'interfaccia comune a tutte le applicazioni. Si interpone fra le applicazioni e la memoria di massa. I dati non appartengono ad un'applicazione, ma esse vi accedono tramite il DBMS.
Le basi di dati sono condivise, ciò aiuta a ridurre la ridondanza e l'inconsistenza.
DBMS: sono componenti software specializzati nel gestire grandi quantità di dati e implementano procedure basate sulle best-practices per la gestione di efficacia ed efficienza, grosse dimensioni di dati, persistenza (tempo di vita non limitato).
Vantaggi DBMS: affidabilità (backup e informazioni di controllo), concorrenza (implementano delle metodologie per garantire il corretto accesso concorrente ai dati minimizzandone l'impatto sulle prestazioni di accesso), privatezza, riduzione del tempo di sviluppo, semplificazione e standardizzazione dello sviluppo.
Svantaggi DBMS: sono costosi, integrazione di servizi non sempre utili.
I DBMS sono prodotti commerciali. Esistono DBMS open-source.
I DBMS sono software in grado di gestire dati eterogenei (con differenti DB).
Schema di dati: per creare e gestire la base di dati; è costruito secondo un modello di dati. Tramite lo schema si fornisce al DBMS una rappresentazione dei dati.
Modello di dati: collezione di costrutti usati per descrivere dati di interesse e la loro struttura in modo che essa risulti comprensibile ad un calcolatore. Con il modello di dati si descrivono lo schema di dati, le loro relazioni e i vincoli di consistenza.
Modello relazionale: descrive lo schema di una base di dati attraverso la specifica delle relazioni che i dati stessi hanno tra di loro. Il costrutto base è la relazione o tabella relazionale, identificata da un nome. I legami fra tabelle vengono espressi in valori.
Schema = struttura di un database. Raramente cambia durante la vita del DBMS.
Istanza: è l'insieme delle informazioni contenute in un database in un certo istante, cambia continuamente nel tempo.
Vantaggio di DBMS: supporto di astrazioni sui dati. Tre diversi livelli di astrazione.
Livello logico: rappresentato dal modello di dati; associato allo schema logico che fornisce una descrizione dell'intera base di dati.
Livello fisico: livello più basso di astrazione, interno al database; associato allo schema interno descrive come i dati rappresentati nello schema logico sono memorizzati nelle strutture fisiche di memorizzazione.
Livello esterno: livello di astrazione più alto che permette di esporre agli utenti finali delle viste parziali; lo schema esterno costituisce la descrizione della porzione della base di dati di interesse.
I livelli di astrazione permettono di garantire l'indipendenza dei dati.
Indipendenza fisica: gli utenti interagiscono con il DBMS indipendentemente dalla struttura fisica usata per memorizzare i dati.
Indipendenza logica: gli utenti interagiscono con le viste in maniera indipendente dal livello logico.
I linguaggi si distinguono in categorie in base allo scopo:

- Data Definition Language (DDL, linguaggio di definizione dei dati): per definire gli schemi e le autorizzazioni per l'accesso ai vari utenti
- Data Manipulation Language (DML, linguaggio di manipolazione dei dati): per interrogare o aggiornare/modificare un'istanza della base di dati
- Data Storage definition Language (SDL, linguaggio di definizione delle strutture di memorizzazione: per specificare le strutture di memorizzazione ed i metodi di accesso della base di dati.

Definito lo schema logico della base di dati, la progettazione della base di dati avviene ricorrendo ai DDL.
L'accesso ai dati in SQL avviene tramite un'interfaccia interattiva.
Gli utenti di un DB possono essere suddivisi in diverse tipologie, in base al ruolo e alle informazioni a cui vogliono accedere:

- Utenti che hanno il controllo completo del sistema:
  - DBA (Data Base Administrator): definisce gli schemi, le strutture dati e metodi di accesso, modifica gli schemi e l'organizzazione fisica, definisce le politiche di accesso e concede permessi, specifica i vincoli di integrità, implementa politiche di backup e recovery, garantisce le prestazioni.
- Utenti che possono essere classificati sulla base dell'esperienza d'uso e sulle funzionalità:
  - Progettisti e sviluppatori di applicazioni: usano il linguaggio di manipolazione, possono aver accesso allo schema logico o ad una vista per accedere ai dati
  - Utenti "casuali": interagiscono direttamente con la base di dati
  - Utenti finali: utilizzano la base di dati indirettamente

Progettazione della base di dati: attività del processo di sviluppo dei sistemi informativi.

Ciclo di vita dei sistemi informatici: insieme e sequenzializzazione delle attività svolte dai vari utenti, attività iterativa (quindi ciclo).
Fasi del ciclo di vita: studio di fattibilità, raccolta e analisi dei requisiti, progettazione, realizzazione, validazione e collaudo, funzionamento/manutenzione.
Progettazione della base di dati: definire i vari schemi per definire come i dati reali verranno memorizzati in essa. Tre fasi:

- Progettazione concettuale: rappresentare le specifiche della realtà di interesse in termini di una descrizione formale e completa, ma indipendente dai criteri di rappresentazione utilizzati nel DBMS; in questa fase si descrive la ealtà che la base di dati dovrà rappresentare, il suo contenuto informativo.
- Progettazione logica: traduzione dello schema concettuale nel modello di rappresentazione dei dati adottato dal sistema di gestione della base di dati a disposizione; in questa fase si progetta lo schema della base di dati a partire dalla descrizione concettuale della realtà, in maniera indipendente da come poi i dati verranno fisicamente memorizzati.
- Progettazione fisica: lo schema logico viene completato con la specifica dei parametri fisici di memorizzazione dei dati; in questa fase si definisce come i dati verranno fisicamente memorizzati

Modelli logici: utilizzati nei DBMS esistenti per l'organizzazione dei dati

→ Progettazione concettuale.

Modelli concettuali: permettono di rappresentare i dati in modo indipendente da ogni sistema, esempio modello Entità-Relazione.
La definizione di un modello concettuale come fase iniziale permette di concentrarsi inizialmente sugli aspetti più importanti dei dati, sulle loro relazioni, su quali dati sono importanti.
Modello ER:

- Permette di descrivere i dati coinvolti in un'organizzazione del mondo reale in termini di oggetti e delle loro relazioni
- Ha una rappresentazione grafica, utile anche come documentazione di un progetto
- Fornisce una serie di costrutti che consentono di descrivere la realtà di interesse in modo semplice da comprendere.
- Costrutti principali:
  - Entità:
    - Rappresentano classi di oggetti di interesse per la realtà esaminata che hanno proprietà comuni e esistenza autonoma
    - Possono essere concrete o astratte
    - Un'istanza (o occorrenza) di un'entità è un'oggetto della classe che l'entità rappresenta
    - Ogni entità ha un nome che la identifica univocamente nello schema
  - Relazioni:
    - Sono associazioni tra due o più entità
    - Rappresentano legami logici significativi per la realtà considerata tra due o più entità
    - Istanze di relazioni sono univocamente individuate dalle istanze delle entità partecipanti
    - Ogni relazione (relationship) ha un nome che la identifica univocamente nello schema
    - Grado di una relazione: è il numero di entità coinvolte nella relazione
    - Relazioni ricorsive possono coinvolgere anche entità appartenenti allo stesso insieme di entità
  - Attributi:
    - Descrivono le proprietà elementari di relazioni e entità che sono di interesse
    - Tutte le istanze di una data entità o relazione hanno gli stessi attributi
    - Un'istanza di un'entità o di una relazione è completamente descritta dagli attributi
    - Il dominio di un attributo rappresenta l'insieme dei valori assunti dall'attributo
    - Semplici: se possono assumere un solo valore
    - Multipli o multivalore: se possono assumere più valori contemporaneamente
    - Composti: se è possibile scomporli in più attributi
    - Opzionali: quando possono non avere un valore definito
  - Vincoli:
    - Di integrità: proprietà che istanze di entità e relazioni devono soddisfare per essere considerate valide
      - Cardinalità sulle relazioni: impongono un limite minimo e un limite massimo al numero di occorrenze della relazione a cui un'entità può partecipare (quante volte in una relazione un'istanza di un'entità può essere legata ad occorrenze delle altre entità coinvolte)
      - Cardinalità sugli attributi: numero minimo e massimo di valori dell'attributo associati a ogni occorrenza di un'entità o relazione.
      - Identificatori delle entità: specificati per ciascuna entità, permettono l'identificazione delle occorrenze delle entità. L'insieme degli attributi selezionati come identificatori sono chiamati anche identificatori chiave.
        Sono entità deboli se possono essere identificate solamente considerando la chiave di un'altra entità. Non avendo un identificatore interno, le entità deboli devono usare attributi di altre entità per formare un identificatore esterno.
        L'identificatore esterno è formato dalla chiave del proprietario e da un discriminatore (insieme di attributi dell'entità debole che permette di distinguere le entità deboli tra loro)

→ Progettazione logica.
Modello relazionale:

- Si basa sul concetto matematico di relazione (in senso algebrico, relation). Ogni relazione è rappresentata graficamente sotto forma di tabella. Un DB viene rappresentato nel modello relazionale come una collezione di tabelle a ciascuna delle quali è assegnato un unico nome.
- L'associazione tra tabelle è realizzata tramite valori.
- Vantaggi modello relazionale: non si fa riferimento ad aspetti fisici quindi si realizza l'indipendenza del modello logico da quello fisico, sono rappresentate solo le informazioni rilevanti per l'utente, è più facile realizzare la migrazione da un sistema ad un altro.
- Relazione/tabella: insieme di n-uple ordinate rispetto ai domini. Non c'è ordinamento fra le n-uple, che sono distinte. Ciascuna tupla è ordinata, l'i-esimo valore proviene dall'i-esimo dominio. Una tabella rappresenta una relazione se le righe sono diverse tra loro, le intestazioni delle colonne sono diverse tra loro e i valori di ogni colonna sono fra loro omogenei.
  Lo schema di una relazione definisce la sua struttura attraverso il nome della relazione, l'insieme dei domini e i nomi assegnati a ciascun dominio (attributi) che descrivono il ruolo giocato dal dominio stesso.

master copy

- Istanza di una relazione: tabella riempita secondo il duo schema; è l'insieme di righe (tuple) che popolano la tabella
- Schema di una base di dati: insieme degli schemi di relazioni che lo compongono
- Istanza di base di dati: insieme delle istanze delle relazioni che la compongono
- I legami fra dati in relazioni diverse vengono rappresentati tramite valori a comune nelle tuple
- Il valore NULL si aggiunge al dominio; troppi valori null in una tabella possono rendere l'informazione inconsistente
- Le strutture del modello permettono di definire come i dati verranno organizzati
- Vincoli di integrità sui dati di una relazione: proprietà che devono essere soddisfatte dalle istanze.
  - Vincolo intrarelazionale: vincoli il cui soddisfacimento è definito rispetto alle singole relazioni della base di dati, quindi possono riguardare valori di una singola tupla (vincolo di tupla) o di più tuple appartenenti alla stessa relazione. Se riguardano il singolo attributo sono vincoli di valore/dominio (es not null).
    - Vincolo di tupla: esprime condizioni sui valori di ciascuna tupla indipendentemente dalle altre
    - Vincolo di dominio: i valori nelle tuple devono essere del dominio (tipo) specificato nello schema, è un vincolo di base
    - Vincolo di chiave: quando una superchiave di una relazione è una chiave (superchiave minimale), ossia se non esiste un'altra superchiave composta da un sottoinsieme dei suoi attributi che sia ancora superchiave. Un insieme di attributi è definito superchiave di una relazione se non esistono due tuple distinte con gli stessi valori degli attributi componenti la superchiave della tabella.
    - Vincolo di chiave primaria: chiave a cui si aggiunge il vincolo NOT NULL al fine di poterla garantire l'identificazione univoca di tutte le tuple. (ogni relazione ha almeno una chiave)
  - Vincolo interrelazionale: se il vincolo coinvolge più relazioni
    - Vincoli di integrità referenziale (foreign key): permette di imporre che i riferimenti tra tabelle siano validi

Finita la fase di progettazione, la base di sati con le sue tabelle è pronta per ospitare istanze di dati, quindi gli utenti sono pronti a fare interrogazioni e manipolare i dati tramite il linguaggio di manipolazione (DML)

Interrogazione: data un'istanza di una base di dati, produce una relazione

Aggiornamento/manipolazione: data un'istanza di una base di dati, produce una nuova istanza sullo stesso schema.

Esistono due tipologie di linguaggi:

- Procedurali: in cui si descrive al sistema il dato che l'utente vuole attraverso l'elenco delle procedure da eseguire per ottenerlo
- Dichiarativi: in cui vengono enunciate direttamente le proprietà del risultato senza alcun accenno alle procedure da eseguire per produrlo.

SQL è a carattere misto.

Algebra relazionale: linguaggio procedurale formale di tipo algebrico i cui operandi sono relazioni. È basata su un insieme di operatori che accettano una o due tabelle come input e producono una tabella come output. Tramite gli operatori definiti si descrivono le procedure di interrogazione per prendere delle tabelle di input e/o generare delle tabelle di output.

Operatori di base:

- Operatori di arità 1 (accettano come argomento una sola tabella):
  - Selezione SEL predicato (relazione): seleziona le tuple di una relazione che soddisfano un predicato producendo un sottoinsieme delle stesse.
  - Proiezione PROJ lista attributi (relazione): è costituita dalla tabella generata dagli attributi specificati, contenente tutte le tuple della tabella di partenza (tuple duplicate ignorate)
  - Ridenominazione REN nome nuovo ← nome (relazione): permette di ottenere una nuova tabella con i nuovi nomi per gli attributi modificati e che ha le stesse tuple della tabella originale
- Operatori di arità 2:
  - Unione U: ignora i duplicati, le due tabelle devono vere lo stesso numero di attributi, i tipi degli attributi corrispondenti devono essere uguali, se hanno nome di attributi diversi vengono usati i nomi della prima tabella, le tabelle devono avere lo stesso grado, il grado della relazione risultato è uguale al grado delle relazioni operandi.
  - Differenza insiemistica – : tabella che contiene le tuple della prima tabella che non sono presenti nella seconda tabella, le relazioni devono avere lo stesso grado e gli attributi compatibili, il grado della relazione risultato è uguale al grado delle relazioni operandi
  - Intersezione ∩: tabella che contiene le tuple che sono presenti nella prima e nella seconda tabella, stesse condizioni di unione
  - Prodotto cartesiano X: tabella con schema la somma degli schemi (duplicati ripetuti), tabella risultato composta da tutte le coppie possibili composte dagli elementi appartenenti alle due relazioni
- Operazioni derivate dalle precedenti:
  - JOIN ><: permette di correlare dati contenuti in relazioni diverse confrontando i valori comuni contenuti in esse.
    - Join naturale (JOIN, ><): correla dati in relazioni diverse sulla base dei valori uguali negli attributi a comune. Se ciascuna tupla degli operandi contribuisce ad almeno una tupla del risultato il join si dice completo, se così non fosse le tuple che non contribuiscono si dicono dondolanti
    - Join esterno: prevede che tutte le tuple diano un contributo al risultato eventualmente estese con il valore null ove non vi siano controparti opportune
    - Theta-join (inner join): permette l'esecuzione di un prodotto cartesiano e della selezione sul risultato.
    - Self-join: join di una tabella con se stessa applicando la ridenominazione

Gli operatori si possono comporre. Più predicati si possono unire con OR, AND e NOT. Gli operatori sono =, <, >, <=, >=, ≠.

Affinché il predicato sia valido i suoi attributi devono appartenere allo schema della relazione, l'operatore deve avere senso nel dominio, se si utilizzano costanti queste devono essere compatibili col dominio.

Viste: relazione risultato di un'interrogazione; hanno la possibilità di semplificare la scrittura di interrogazioni complesse.

master COPY

Il linguaggio SQL (Structured Query Language) è un linguaggio dichiarativo (non-procedurale) ovvero non specifica la sequenza di operazioni da compiere per ottenere il risultato. È "relazionalmente completo" perché ogni espressione dell'algebra relazionale può essere tradotta in un corrispondente espressione in SQL. Funzionalità:

- Data Manipulation Language (DML): linguaggio di manipolazione dei dati, è l'insieme di istruzioni dedicate alla manipolazione di dati (istruzioni per l'inserimento, la cancellazione e la modifica di dati)
- Data Definition Language (DDL): linguaggio di definizione dei dati, è l'insieme di istruzioni dedicate alla definizione di dati e tabelle
- Data Control Language (DCL): linguaggio di controllo degli accessi, è l'insieme di istruzioni che perette di definire i permessi necessari per accedere alle tabelle della base di dati

Alla base del modello relazionale ci sono le tabelle con i relativi attributi. Ogni attributo appartiene ad un dominio che deve essere ben definito, a tal fine SQL definisce vari tipi di dato.
Un tipo di dato è costituito da un insieme di valori e da un insieme di operazioni definite sul tipo. I tipi di dato o (domini) sono classificati in:

- Numerici:
  - Esatti: valori interi e valori decimali in virgola fissa (il separatore è ".")
    - NUMERIC (precisione, scala): precisione: numero totale di cifre (>0), scala: (≥0 e <precisione) è il numero di cifre alla destra del separatore decimale.
    - DECIMAL (precisione, scala) abbr DEC: identico a NUMERIC, con l'unica differenza che la precisione rappresenta il valore minimo con cui poi i numeri verranno memorizzati
      In mySQL DECIMAL e NUMERIC sono sinonimi
    - BIGINT: rappresenta i valori interi (senza cifre decimali) (in C2 su 64 bit)
    - INTEGER abbr INT: rappresenta i valori interi, non è possibile dimensionarlo (in C2 su 32 bit)
    - SMALLINT: rappresenta i valori interi, richiedono minore spazio di memorizzazione (in C2 su 16 bit)
  - Approssimati: valori reali in virgola mobile
    - REAL: valori a singola precisione (dipende dalla specifica implementazione di SQL) in virgola mobile.
    - DOUBLE PRECISION o DOUBLE: valori a doppia precisione in virgola mobile. (su 64 bit)
      In mySQL REAL e DOUBLE sono sinonimi
    - FLOAT: permette di richiedere la precisione che si desidera (il parametro p indica il numero di cifre da rappresentare per la mantissa). Il formato è FLOAT(p). Ha un'implementazione non standard. (su 32 bit)
- Carattere: singoli caratteri o stringhe, anche di lunghezza variabile
  - CHARACTER (CHAR): permette di definire stringhe di carattere di lunghezza fissata n. CHAR(n)
  - CHARACTER VARYING (VARCHAR): permette di definire stringhe di carattere di lunghezza variabile con lunghezza massima predefinita n. La specifica è della forma VARCHAR(n). La dimensione è obbligatoria e va da 1 a 4000 (65536 in mySQL).
- Temporali: per permettere la gestione di date ('anno-mese-giorno') e ore ('ore-minuti-secondi')
  - DATE: rappresenta le date espresse come anno (4 cifre), mese (2 cifre comprese tra 1 e 12) e giorno (2 cifre comprese tra 1 e 31 con ulteriori restrizioni a seconda del mese)
  - TIME: rappresenta i tempi espressi come ore (2 cifre), minuti (2 cifre) e secondi (2 cifre) – sono utilizzati sia per rappresentare tempi che intervalli temporali
  - DATETIME: rappresenta una concatenazione dei due tipi di dato precedenti con una precisazione al microsecondo, sono specificati nella forma 'anno-mese-giorno ore:minuti:secondi.microsecondi' ma sono memorizzati 'YYYY:MM:GG HH:MM:SS' dove HH:MM:SS deve essere un tempo e non un intervallo
- Bit (singoli booleani o stringhe)
  - Per rappresentare appositamente i valori booleani, che generalmente occupano solo 1 bit
  - TRUE, FALSE, UNKNOWN dove true e false solo alias per 1 e 0
- Booleani
- Valore NULL: presente in tutti i DBMS, rappresenta l'assenza di informazione. È compatibile con ogni tipo di dato. Nelle espressioni aritmetiche restituisce sempre NULL. Per verificare se un valore è NULL devo usare IS NULL o IS NOT NULL (non = e !=). Operatori e funzioni su valori null restituiscono null.
- Tipi di dato definiti dall'utente a partire dai domini elementari:
  - CREATE DOMAIN nome_dominio AS tipo [DEFAULT value] [CHECK (vincolo)]

SQL definisce diversi tipi di carattere utilizzati per rappresentare stringhe (sequenze) di caratteri. Assegnare un valore più lungo della lunghezza specificata nella definizione del tipo di dato costituisce un errore a tempo di esecuzione (run time error).
Sintassi: tipocarattere(dimensione)
Sui vari tipi sono definiti un certo numero di operatori, gli operandi devono essere dello stesso tipo altrimenti il risultato ha poco o nullo significato. Gli operatori hanno precedenza e associatività. Sono definite anche un certo numero di funzioni atte a manipolare i tipi.
Si usano con i tipi numerici +, -, /, *, %, DIV per divisione intera
Operatori di confronto: =, !=, >, <, >=, <=, IS / IS NOT boolean_value – test per boolean
Il confronto si applica ai tipi numerici, alle stringhe (case insensitive in mySQL) e alle date.
Operatori logistici: AND, OR, NOT

master COPY

Funzioni scalari per le stringhe:

- CONCAT str1, str2, ... : restituisce una stringa che è la concatenazione degli argomenti
- LOWER (str) e UPPER (str): restituiscono una stringa con i caratteri tutti minuscoli o maiuscoli
- CHAR_LENGTH(str): restituisce la lunghezza della stringa

Funzioni scalari per le date:

- NOW(): restituisce la data e l'ora corrente
- CURTIME(): restituisce l'ora corrente
- CURDATE(): restituisce la data corrente
- DATEDIFF(expr1, expr2): restituisce la differenza in giorni tra due date
- TIMEDIFF(expr1, expr2)(exp1-expr2): restituisce la differenza tra due intervalli temporali (time)
- TIMESTAMPDIFF(unit, datetime_expr1, datetime_expr2): restituisce ila differenza espressa in unit (second, minute, hour, day, week, month, year)
- ADDTIME(expr1, expr2) e SUBTIM(expr1, expr2): dove expr1 è TIME o DATETIME e expr2 è TIME; restituisce expr2 +/- expr1
- Date +/- INTERVAL (expr, unit): aggiunge o sottrae intervalli, con date è datetime e unit è l'unità di tempo
- DATE_FORMAT(date, format): estrarre i campi da una data; format è una stringa che specifica il formato
- TIME_FORMAT(time, format)
- year(date), month(date), day(date): estrae anno/mese/giorno dalla data
- STR_TO_DATE(str, format): trasforma le stringhe in date, datetime o time

Commenti su SQL: -- se il commento è su un'unica riga, /* */ se il commento è su più righe.
I risultati delle interrogazioni possono essere: valori singoli (una sola tupla con una sola colonna), liste (una sola colonna), tabella.
Query optimizer: è un componente che analizza l'interrogazione e forma un'interrogazione equivalente nel linguaggio procedurale interno al sistema (nascosto all'utente).

SELECT lista attributi FROM lista tabelle WHERE condizioni
Lista attributi: cosa si vuole come risultato, è la lista delle colonne che si vuole estrarre, l'ordine e il nome che si vuoe assegnare
Lista tabelle: da dove si prende, è la lista delle tabelle dalle quali verranno estratti i dati
Condizioni: definisce una condizione su cui verranno filtrati i record da estrarre
SELECT ALL(duplicati inclusi)/DISTINCT(duplicati esclusi) expr AS rename : corrisponde ad un'operazione di proiezione e ridenominazione
Se non ci sono tuple che soddisfano la condizione, l'interrogazione restituisce un insieme vuoto.
Nel select in expr posso inserire operazioni e funzioni.
In expr specifico gli attributi che devono comparire nel risultato
La clausola as consente di rinominare la colonna (può anche essere omesso AS).

FROM: specifica le tabelle su cui si fa la query, costruisce la tabella intermedia a partire da una o più tabelle su cui operano le clausole select e where. Con la clausola AS posso specificare un alias della tabella per convenienza (non è obbligatorio).
Gli attributi con lo stesso nome di tabelle diverse si distinguono indicandoli con NomeTabella.NomeAttributo
Forma estesa per referenziare una colonna NomeTabella.NomeColonna

WHERE: permette di filtrare le tuple in base a una condizione; consiste in un'espressione logica di predicati combinati con operatori logivi AND, OR, NOT, matematici, di comparazione, operatori sulle stringhe. È opzionale.
WHERE (condizione) dove condizione è un'espressione booleana ottenuta combinando predicati semplici con gli operatori NOT, AND, OR (in ordine di precedenza). I predicati semplici sono costruiti utilizzando gli operatori di confronto.

Qualsiasi confronto logico con il valore NULL porta a UNKNOWN/NULL come risultato. L'applicazione di funzioni e/o operatori in cui un operando è null restituiscono null.

LIKE per il confronto tra stringhe
_ corrisponde a un carattere arbitrario
% corrisponde a una stringa arbitraria

BETWEEN value AND value: confronto numerico, permette di esprimere condizioni di appartenenza ad un intervallo (estremi inclusi)
IN/NOT IN (expr): appartenenza o meno a un insieme che può essere un insieme di valori numerici, stringhe o altri tipi di dato
ORDER BY (expr): per ordinare il risultato di una query secondo uno o più valori di una o più colonne, ASC (default) o DESC. L'attributo rispetto a cui si ordina non deve comparire necessariamente nel select.

Funzioni di gruppo: permettono di calcolare espressioni/valori su un insieme di righe, operano su più righe ma restituiscono un unico valore.
COUNT(* |attributo|ALL attributo| DISTINCT attributo): permette di contare il numero di tuple della tabella risultante.
(*) conta anche null, (attr | ALL attr) conta numero righe con valori di attributo diversi da null con duplicati, (DISTINCT attr) senza duplicati
In count possono apparire più attributi.
MIN(expr): seleziona il minimo, argomento su cui è definito un ordinamento
MAX(expr): seleziona il massimo, argomento su cui è definito un ordinamento
AVG(expr): seleziona la media, argomento numerico
SUM(expr): somma totale dei valori, argomento numerico

GROUP BY ListaAttr: consente di realizzare l'operazione di aggregazione su sottoinsiemi delle tuple. Le colonne usate per il raggruppamento sono le sole che possono apparire anche nella clausola SELECT e ORDER BY. La clausola va inserita dopo il WHERE. Il raggruppamento considera come gruppo anche le tuple che hanno sull'attributo di raggruppamento il valore null.

HAVING expr: può contenere funzioni di aggregazione, introduce predicati su gruppi di tuple. Deve essere scritta dopo GROUP BY. Permette di specificare condizioni sui gruppi di tuple generate da GROUP BY in modo da selezionare solo i gruppi che rispettano certi criteri.

Ordine: SELECT - FROM - WHERE - GROUP BY - HAVING

DELETE FROM Tab WHERE ...
UPDATE Tab SET Campo=Val WHERE ...

INSERT INTO Tab (Campo1, Campo2, ...) VALUES (Val1, ...) / SELECT ...