**1)Com’è formato un sistema aziendale?**

Sistema organizzativo, informativo e informatico

**2)Cos’è un DBMS? Per cosa si differenziano dai file system?**

Sistema software per gestire collezioni di dati (basi di dati) che siano grandi, condivise e persistenti. I file system applicano il tutto o niente, mentre i DBMS forniscono più servizi in maniera integrata

**3)Come sono le BD?**

**Grandi:** il limite è solo quello fisico del dispositivo

**Condivise:** ogni organizzazione (specie se grande) è divisa in settori o comunque svolge diverse attività

**Persistenti:** hanno una vita indipendente dalle singole esecuzioni dei programmi che le utilizzano

**4)Quali sono le strategie di progettazione che possiamo applicare nella modellazione concettuale**

**Top-down:** Lo schema concettuale viene prodotto tramite una serie di raffinamenti successivi. Dallo schema iniziale astratto, trasformando lo schema aggiungendo i dettagli.

(Schema iniziale---->schema intermedio---->schema finale)

**Inside-Out:** (strategia particolare della bottom-up)  
Si individuano solo inizialmente alcuni concetti importanti e si procede da questi a macchia d'olio, espandendoli man mano. Si rappresentano prima i concetti in relazione con i concetti iniziali per poi muoversi verso quelli più lontani. (uno per volta)

**Bottom Up:** In questa strategia le specifiche iniziali vengono suddivise in componenti sempre più piccoli fino a quando questi elementi non descrivono un frammento elementare della realtà d'interesse. (Divido il problema in più sotto-problemi per isolarli e risolverli più semplicemente)

**5)Fasi di progettazione concettuale, logica e fisica**

**Concettuale:** a partire dai requisiti informativi viene creato uno schema concettuale cioè una descrizione funzionalizzata e integrata delle esigenze aziendali, espresso in modo indipendente dal DBMS adattato.

L’uso degli schemi concettuali permette:

* il coinvolgimento e il controllo da parte del committente del risultato
* maggiore chiarezza degli obiettivi
* stabilità della progettazione
* maggiore facilità di integrazione di nuovi dati
* migliore possibilità di confronto fra sistemi informativi

**Fase di raccolta e analisi dei requisiti:** È la fase in cui si raccolgono e analizzano le specifiche informali ed eterogenee che i vari utenti danno delle procedure da automatizzare mediante un DBMS

Attività principali:

* Costruzione glossario dei termini
* Eliminazione delle ambiguità (sinonimi, omonimi)
* Raggruppamento dei requisiti “omogenei”

**Logica:** consiste nella traduzione dello schema concettuale nel modello dei dati del DBMS. Il risultato è uno schema Logico, espresso nel DDL dal DBMS. (Gerarchico e reticolare utilizzano riferimenti espliciti (puntatori) fra record)

In questa fase si considerano anche aspetti come:

* efficienza
* integrità e coesistenza (vincoli).

La progettazione logica si articola in due sottofasi:

* ristrutturazione dello schema concettuale
* traduzione verso il modello logico

Fase di ristrutturazione:

* Analisi delle ridondanze
* Eliminazione delle generalizzazioni
* Partizionamento o Accorpamento di alcuni concetti di entità o associazioni
* Scelta degli identificatori principali

**Fisica:** in questa fase si operano scelte spesso legate strettamente al DBMS utilizzato. Il risultato lo schema fisico, che descrive le strutture di memorizzazione e accesso ai dati

**6)Cos'è e come è composto il modello ER?**

Il modello ER un grafo, ossia uno schema che costituisce una fotografia del fenomeno. (è composto da entità e relazioni, caratterizzate da attributi)

**7)Cosa distingue un’entità debole da una forte?**

L'entità è un insieme di oggetti di cui sono note alcune caratteristiche. Ogni entità ha un nome che la identifica univocamente.

**Entità forti:** entità che possiedono un identificatore che le specifichi

**Entità deboli:** entità per le quali non si trova un identificatore

Per queste ultime l’identificatore è ottenuto utilizzando altre entità, ovvero con un identificatore esterno (le entità deboli hanno cardinalità (1,1))

**8)Cos'è una relazione?**

Una relazione è un legame logico tra due o più entità. Si trova indicata anche come correlazione o associazione.

**9)Cosa sono e a cosa servono le cardinalità?**

Dicono quante volte, in una relazione tra entità, un'occorrenza di una di queste entità può essere legata a occorrenze delle altre entità coinvolte. (l'occorrenza o istanza è un elemento della classe non sono i dati) Nella maggior parte dei casi si utilizzano tre valori: 0, 1 e N

**10)Che cos'è e a cosa serve l'identificatore?**

Vengono specificati per ogni entità di uno schema e descrivono i concetti (attributi o entità) dello schema che permette di identificare in maniera univoca le occorrenze delle entità.

Spesso uno o più attributi sono sufficienti a individuare un identificatore: si parla così di identificatore interno o chiave

**11)Attributi**

Una riga con un pallino finale rappresenta gli attributi di un’entità o di una relazione. Associa ad ogni occorrenza d'identità o di relazione un valore appartenente a un insieme detto dominio dell'attributo.

Attributi:

* atomici vs composti
* A valore singolo vs multivalore

**Attributo atomico**: è un attributo che non è composto a sua volta da altri attributi

**Attributo composto:** è un attributo che si basa sull'unione di più attributi messi insieme

**Attributo valore singolo:** sono quegli attributi unici nell’istante in cui vengono considerati

**Attributo multivalore:** sono quegli attributi che possono essere molteplici nell’istante in cui vengono considerati

**Attributo derivabile:** è possibile avere un attributo derivabile se esso deriva da altri attributi

**Attributo esterno:** si ottiene quando si utilizza un identificatore di un entità per utilizzarlo su un’entità che non la possiede

**12)Sottoclasse e superclasse**

Un tipo entità può avere dei sotto raggruppamenti delle sue entità in base a particolari caratteristiche. In alcuni contesti può essere interessante o necessario rappresentare tali raggruppamenti, per esempio memorizzare informazioni specifiche

L'entità della sottoclasse rappresenta l'entità della superclasse

**13)Generalizzazione e specializzazione**

Una generalizzazione è **totale** su ogni occorrenza dell’entità genitori e in un'occorrenza di almeno un'entità figlia. Sennò si definisce **parziale**.

L'entità figlie non solo hanno gli attributi del padre e i propri, ma partecipano anche alle sue relazioni

Infatti, l'identificatore dei figli è sempre lo stesso del padre

Una generalizzazione può essere:

**Esclusiva/Disgiuntiva:** se ogni occorrenza del padre è al più un’occorrenza di un’entità figlia, altrimenti si dice sovrapposta

**Sovrapposta:** un elemento del padre può appartenere a più di un figlio

La specializzazione è il processo di definire un insieme di sottoclassi in un tipo entità, detto superclasse delle specializzazioni sulla base di particolari caratteristiche

**14)Come si ristruttura una generalizzazione**

14.1) Le entità figlie vengono eliminate e le loro proprietà vengono accorpate all’entità genitore con l'aggiunta di un attributo "tipo"//PARZIALE

14.2) Accorpamento del genitore nelle figlie eliminando l'entità genitore. Le sue proprietà vengono aggiunte alle entità figlie (possibile solo se di tipo totale)

14.3) Sostituzione della generalizzazione con delle associazioni. L'entità genitore, quindi viene collegato tramite relazioni 1:1 alle entità figlie (consigliata quando non è totale) //ESCLUSIVA

**15)Cos'è una relazione ricorsiva?**

Quando lo stesso tipo di entità partecipa ad una relazione più di una volta con ruoli diversi, viene detta ricorsiva

**16)Cos'è e come si utilizza una reificazione?**

Per reificazione si intende trasformare una relazione con cardinalità massima n:n in entità debole con chiave esterna con cardinalità 1:1 in relazione con le due altre entità base con cardinalità x:n

La relazione viene reificata quando ha come cardinalità massima n:n

**17)Business Rules**

Le business rules si definiscono in fase di programmazione concettuale.

Le business rules sono uno degli strumenti più usati dagli analisti. Si tratta di regole dal particolare dominio applicativo:

* **concetto:** descrizione di un concetto rilevante per l'applicazione
* **un vincolo di integrità sui dati dell'applicazione:** sia esso la documentazione di un vincolo espresso con qualche costrutto del modello ER o la descrizione di un vincolo non esprimibile con i concetti del modello
* **Una derivazione:** ovvero un concetto può essere ottenuto attraverso un inferenza o un calcolo aritmetico

Le BR possono essere:

* Descrittive => dizionario dei dati:
* tabella delle entità dello schema
* tabella delle relazioni
* Non descrittive =>
* tabelle delle regole di vincolo
* tabelle delle regole di derivazione

**18)Mapping**

Si ottiene applicando allo schema concettuale ristrutturato un insieme di regole di traduzione.

Nell’**associazione uno a uno con partecipazione obbligatoria** si deve passare la chiave di una delle due entità nell’altra come attributo semplice più l’attributo della relazione

Nell’**associazione uno a uno con partecipazione opzionale** per un’entità si deve passare la chiave dell’entità con cardinalità (0,1) all’entità con cardinalità (1,1) come attributo semplice più l’attributo della relazione

Nell’**associazione a uno a uno con partecipazione opzionale per entrambe le entità** si deve passare la chiave di una delle due entità all’altra come attributo semplice e viene passato anche l’attributo della relazione, entrambi con un \* (indica che l’attributo può essere nullo)

Oppure le chiavi di entrambe le entità vengono passate alla relazione come chiavi esterne

Nell’**associazione binaria molti a molti** si devono passare le loro chiavi nella relazione, entrambe come chiavi esterne

Nell’**associazione** **ternaria molti a molti** si devono passare le loro chiavi nella relazione, come chiavi esterne

**Associazione uno a molti con partecipazione obbligatoria** si passa la chiave dell’entità con cardinalità (x,n) all’entità con cardinalità (1,1) più l’attributo semplice

**Associazione uno a molti con partecipazione opzionale** la chiave dell’entità con cardinalità (0,1) passa alla relazione come chiave esterna, mentre la chiave dell’entità con cardinalità (x,n) come attributo semplice

oppure

la chiave dell’entità con cardinalità (x,n) viene passata all’entità con cardinalità (0,1) assieme all’attributo della relazione, entrambi con un \* (Associazione uno a uno con partecipazione opzionale per entrambe le entità)

**Associazione con identificatore esterno** la chiave dell’entità con cardinalità (x,n) viene passata come componente della chiave primaria dell’altra entità (con cardinalità (1,1)) più l’attributo della relazione

**19)Che cos'è il carico applicativo?**

Le informazioni necessarie per progettare il database.

È formata dalla tavola dei dati + la tavola delle operazioni

**20)Cosa si fa in caso di ridondanza**

L'analisi delle ridondanze, visione delle tabelle e decisione se mantenere o meno l'attributo derivabile.

**21)Cosa si fa in fase di ristrutturazione?**

* Analisi delle ridondanze
* Eliminazione delle generalizzazioni
* Partizionamento o Accorpamento di alcuni concetti di entità o associazioni
* Scelta degli identificatori principali

**22)Tavola dei volumi, tavola delle operazioni, tavola degli accessi**

**Tavola dei volumi:** tavola dove indichiamo le entità e relazioni con il loro rispettivo volume, cioè il numero massimo di occorrenze che possono essere associate a quelle entità e relazioni.

**Tavola delle operazioni:** operazioni che vogliamo effettuare

Interattiva: un dato in input.

Batch: invece no.

**Tavola degli accessi:** gli accessi per ogni operazione, ci dà il numero di letture e di scritture effettuate per quell’operazione e l’occupazione di memoria.

**23)Algebra relazionale**

L'algebra relazionale è un linguaggio di interrogazione teorico delle basi dati nel modello relazionale.

**La proiezione:** è un operatore unario ortogonale dell'algebra relazionale che seleziona alcuni attributi (colonne) di una relazione (tabella). Il simbolo della proiezione è il p greco ∏ con in pedice la lista degli attributi e tra parentesi il nome della r. Il risultato della proiezione è una tabella contenente al più tutte le ennuple della relazione in ingresso senza duplicazioni ma soltanto alcune colonne.

**La selezione:** è un operatore unario ortogonale dell'algebra relazionale che seleziona le tuple di una relazione (tabella) che soddisfano una particolare condizione. Il simbolo della selezione è sigma σ o SEL. Il risultato della selezione è una tabella con lo stesso schema della relazione r(X) in ingresso. Dove X è l'insieme degli attributi.

**Il prodotto cartesiano:** è un operatore binario insiemistico dell'algebra relazionale che, date due relazioni r1 e r2, crea una relazione composta da tutte le combinazioni possibili delle tuple della prima con le tuple della seconda. Il simbolo del prodotto x

**La ridenominazione:** è un operatore unario dell'algebra relazionale che permette di cambiare i nomi degli attributi di una relazione R. Il simbolo è ρ.

ρ nuovo nome <— nome attributo (relazione)

L'operazione di ridenominazione cambia soltanto i nomi degli attributi nello schema della relazione e non i dati nelle tuple.

**L'unione:** è un operatore insiemistico binario dell'algebra relazionale che unisce due relazioni r1 e r2 (tabelle) in un'unica relazione r3. Il simbolo dell'unione è U. Il risultato è una relazione unione r3 composta da tutte le tuple delle due relazioni, senza duplicazioni.

**L'intersezione:** è un operatore insiemistico binario dell'algebra relazionale che seleziona le tuple in comune in due relazioni r1 e r2 (tabelle). Il simbolo dell'intersezione è ⋂. Il risultato dell'intersezione è una relazione composta dalle tuple presenti in entrambe le relazioni, senza duplicazioni.

**La differenza:** è un operatore insiemistico binario dell'algebra relazionale che contiene le tuple presenti nella relazione r1 ma non nella relazione r2. Il risultato della differenza è una relazione composta dalle tuple presenti nella prima relazione ma non nella seconda.

**La divisione:** è un operatore binario dell'algebra relazionale che contiene le tuple della prima relazione A tali che per ogni tupla della seconda relazione B ci sia una tupla nella prima relazione A.

A / B = {<x> | ∀ <y>∈ B, <x, y>∈A}

**24)Join**

L'operatore join in algebra relazionale correla i dati contenuti in relazioni diverse confrontando i valori. Esistono diversi join:

**Join completo:** Il join si dice completo se ogni tupla di ciascuna relazione degli operandi contribuisce almeno a una tupla del risultato.

**Join incompleto:** Il join si dice incompleto se almeno una tupla delle relazioni degli operandi non contribuisce al risultato.

**Join vuoto:** Il join si dice vuoto se nessuna tupla delle relazioni degli operandi contribuisce al risultato finale.

**Inner join:** Una inner-join crea una nuova tabella combinando i valori delle due tabelle di partenza (A and B) basandosi su una certa regola di confronto. Nella inner-join fanno parte:

**Join naturale:** correla i dati in due relazioni diverse sulla base dei valori uguali negli attributi con

lo stesso nome. Il simbolo del join naturale è una farfalla. ⋈

**Il join condizionale (theta join):** correla i dati in due relazioni diverse sulla base di una condizione booleana. È anche detto theta join. Il simbolo del join condizionale è una farfalla seguita da una C. r 1 ⋈ C r 2

L'operatore theta join legge le tuple della prima relazione e verifica quali tuple della seconda relazione soddisfano la condizione di join.

**L’equi-Join:** Se la condizione è composta da operatori di uguaglianza (=), eventualmente in congiunzione (and) tra loro, il join condizionale è detto equi join. Di fatto l'equi join è simile a un join naturale in cui nella condizione di join sono ammessi gli attributi.

**Outer Join**: Una outer join non richiede che ci sia corrispondenza esatta tra le righe di due tabelle. La tabella risultante da una outer join trattiene tutti quei record che non hanno alcuna corrispondenza tra le tabelle. Le outer join si suddividono in:

**Left Join:** Il risultato di una query left outer join (o semplicemente left join) per le tabelle A e B contiene sempre tutti i record della tabella di sinistra ("left") A, mentre vengono estratti dalla tabella di destra ("right") B solamente le righe che trovano corrispondenza nella regola di confronto della join.

**Right Join:** Una right outer join (o right join) semplicemente ricalca il funzionamento della left outer join, ma

invertendo l'ordine delle tabelle interessate.

**Full Join:** Una full outer join combina i risultati delle due tabelle A e B tenendo conto di tutte le righe delle tabelle, anche di quelle che non hanno corrispondenza tra di loro. (Conviene usarla quando si vogliono produrre anche i valori nulli)

**25)Operatori fondamentali dell'algebra**

FONDAMENTALI: unione, differenza, prodotto cartesiano, selezione, proiezione, ridenominazione

DERIVATI: intersezione, join, divisione

**26)Cos’è una forma normale? Cos’è la normalizzazione? Quali sono i tipi di normalizzazione?**

È una proprietà di uno schema relazionale che ne garantisce la “qualità”, cioè l’assenza di determinati difetti.

Una relazione non normalizzata: presenta ridondanze, si presta a comportamenti poco desiderabili durante gli aggiornamenti.

Le forme normali sono di solito definite sul modello relazionale, ma hanno senso anche in altri contesti, ad esempio nel modello E/R

L’attività che permette di trasformare schemi non normalizzati in schemi che soddisfano una forma normale è detta normalizzazione.

La normalizzazione va utilizzata come tecnica di verifica dei risultati della progettazione di una base di dati. Non costituisce quindi una metodologia di progettazione.

**La dipendenza funzionale:**

Un’istanza r di uno schema R(X)

Due sottoinsiemi (non vuoti) di attributi Y e Z di X

Diciamo che in r vale la dipendenza funzionale (FD)Y→Z (Y determina funzionalmente Z) se per ogni coppia di tuple t1 e t2 di r con gli stessi valori su Y, t1 e t2 hanno gli stessi valori anche su Z

**Prima Forma Normale (1NF):** Richiede semplicemente che tutti gli attributi dello schema abbiano domini “atomici” (ovvero non siano composti o multivalore)

**Seconda Forma Normale (2NF):** Uno schema R(X) è in seconda forma normale se e solo se ogni attributo non-primo (ovvero non appartenente a nessuna chiave) dipende completamente da ogni chiave (ovvero non dipende solamente da una parte di chiave)

**Forma Normale di Boyce-Codd (BCNF):** Uno schema R(X) è in forma normale di Boyce e Codd se e solo se, per ogni dipendenza funzionale (non banale) Y → Z definita su di esso, Y è una super chiave di R(X)

Si noti che, come al solito, i vincoli si riferiscono allo schema, in quanto dipendono dalla semantica degli attributi. Un’istanza può pertanto soddisfare “per caso” il vincolo, ma ciò non garantisce che lo schema sia normalizzato. In altri termini, le FD (Dipendenze Funzionali) non si “ricavano” dall’analisi dei dati, ma ragionando sugli attributi dello schema.

Una forma normale meno restrittiva della BCNF è la **Terza Forma Normale (3NF):** Uno schema R(X) è in terza forma normale se e solo se, per ogni dipendenza funzionale (non banale) Y → Z definita su di esso, Y è una superchiave di R(X) oppure ogni attributo in Z è primo (cioè contenuto in almeno una chiave di R(X).

**Definizione:** Uno schema R(X) è in terza forma normale se e solo se ogni attributo non-primo non dipende transitivamente da nessuna chiave.

**27)Cos’è un vicolo? Tipologie di vincoli**

Un vincolo è una funzione booleano (predicato) che associa ad ogni istanza il valore vero o falso.

**Vincoli di integrità:** regole che vengono applicate alle tabelle e che vincolano i valori che possono essere inseriti. Si dividono in:

* **Vincoli intrarelazionali:** il suo soddisfacimento è definito su singole relazioni
* **Vincoli interrelazionali:** se coinvolge più relazioni

**Vincoli su valori (o di dominio):** specificano l’attributo associato a un dominio deve assumere valori in quel dominio

**Vincoli di n-pla/Vincolo di tupla:** Vincolo con singoli valori è detto vincolo di dominio perché limita l’attributo

**Vincoli di chiave:** sono i più importanti e necessitano di un attributo univoco

**28)Cosa sono gli attributi di relazione**

Caratteristica della relazione che dipende dalle due entità della relationship.

**29)Cosa sono le viste?**

Rappresentazione di tabelle “temporanee”. Possono essere di due tipi:

* **Viste materializzate:**
  + **vantaggi:** subito disponibili per le query
  + **svantaggi:** ridondanti, appesantiscono gli aggiornamenti sono raramente supportate dai DBMS
* **Viste virtuali:** Sono supportate da tutti i DBMS

Un interrogazione su una vista viene eseguita “ricalcolando la funzione”

Ben pochi sono gli aggiornamenti ammissibili sulle viste

Le tuple della vista sono il risultato di una query che viene valutata dinamicamente ogni volta che si fa riferimento alla vista.

Le viste vengono create per diversi scopi:

* Permette agli utenti dia vere una visione personalizzata del DB
* Far fronte a modifiche dello schema logico
* Semplificare la scrittura di query complesse
* Possono essere usate come meccanismo per il controllo degli accessi

**30)MYSQL**

Il linguaggio SQL (Structured Query Language), a dispetto del nome, non è solo un linguaggio di interrogazione (query) per database basati sul modello relazionale. In base alle funzioni eseguite sui dati, è possibile distinguere i seguenti gruppi di comandi SQL:

* **DDL (Data Definition Language):** consente di creare e modificare schemi di database;
* **DML (Data Manipulation Language):** consente di inserire, modificare e gestire i dati memorizzati;
* **DQL (Data Query Language):** consente di interrogare i dati memorizzati;
* **DCL (Data Control Language):** consente di creare e gestire strumenti di controllo ed accesso ai dati;
* **TCL (Transaction Control Language):** consente di avviare, concludere e gestire le transazioni.

**I comandi di tipo DDL** (Data Definition Language) consentono di creare, modificare o cancellare gli oggetti in un database, intervenendo a livello di struttura degli oggetti stessi dello schema.

**CREATE, ALTER, DROP, RENAME**

**I comandi di tipo DML** (Data Manipulation Language) riguardano le operazioni di aggiornamento dei dati; consentono quindi di inserire, modificare o cancellare i dati delle tabelle, senza intervenire a livello di struttura degli oggetti stessi dello schema.

**INSERT, UPDATE, DELETE**

**I comandi di tipo DQL** (Data Query Language) riguardano le operazioni di interrogazione dei dati; consentono quindi di recuperare i dati delle tabelle.

**SELECT, JOIN, SUBQUERY**

**GROUP BY**: è una clausola che raggruppa le righe che hanno gli stessi valori in righe di riepilogo, viene spesso utilizzata con funzioni aggregate (COUNT, MAX, MIN, SUM, AVG) per raggruppare il set di risultati in base a una o più colonne.

L’operatore **LIKE**, mediante il \_ (un carattere arbitrario) e % (una stringa arbitraria), permette di esprimere dei “pattern” su stringhe

L’operatore **BETWEEN** permette di esprimere condizioni di appartenenza a un intervallo

L’operatore **IN** permette di esprimere condizioni di appartenenza a un insieme

Per ordinare il risultato di una query secondo i valori di una o più colonne si introduce la clausola **ORDER BY**, e per ogni colonna si specifica se l’ordinamento è per valori “ascendenti” (**ASC**) o “discendenti” (**DESC**)

**Operatori insiemistici:**

L’istruzione SELECT non permette di eseguire unione, intersezione e differenza di tabelle.

Ciò che si può fare è combinare in modo opportuno i risultati di due istruzioni SELECT, mediante gli operatori: **UNION, INTERSECT, EXCEPT**

In tutti i casi gli elementi delle SELECT list devono avere tipi compatibili e gli stessi nomi se si vogliono colonne con un’intestazione definita. L’ordine degli elementi è importante (notazione posizionale).

Il risultato è in ogni caso privo di duplicati, per mantenerli occorre aggiungere l’opzione **ALL**

In SQL è possibile fare uso di subquery.

* Una subquery che restituisca al massimo un valore è detta scalare, e per essa si possono usare i soliti operatori di confronto
* Le forme **ANY** e **ALL** si rendono necessarie quando la subquery può restituire più valori
* Il quantificatore esistenziale **EXISTS** è soddisfatto quando il risultato della subquery non è vuoto (e **NOT EXISTS** quando è vuoto)

**31) Funzioni aggregate**

COUNT (conteggio delle righe), MAX (massimo valore), SUM (somma delle righe di una colonna), MIN (minimo valore), AVG (media).

**COUNT**: effettua il conteggio di tutte le righe presenti nel gruppo, indipendentemente dai valori assunti. Può essere usato anche senza GROUP BY, per calcolare le righe totali presenti in una tabella: Se tra le parentesi specifichiamo un’espressione, verranno contate solo le righe che hanno quell’espressione non nulla. Se prima dell’espressione indichiamo la parola chiave DISTINCT verranno conteggiate solo le espressioni non nulle e distinte.

**SUM:** somma i valori dei campi trovati nel gruppo. I valori nulli vengono ignorati.

**MIN, MAX e AVG (media):** con queste funzioni si possono ottenere i valori massimo e minimo di una colonna, in base al criterio di ordinamento predefinito (ad esempio per il testo verrà usato un ordinamento alfabetico). I valori nulli vengono ignorati. Naturalmente più funzioni possono essere usate nella stessa query

**32) Differenza tra WHERE e HAVING**

Esprimono entrambe delle condizioni… HAVING (funzioni aggregate) viene utilizzato per i gruppi come insieme, mentre WHERE (attributi atomici) tiene conto delle righe.

(La WHERE effettua il filtraggio solo sull’attributo. La HAVING effettua il filtraggio sulle funzioni di aggregazione)

**33)Valori nulli**

Valori senza informazione

Valori inesistenti

Valore sconosciuto (non sa da dove prenderlo)