1=Join, 2= Forme normali, 3=Algebra, 4=Funzioni di aggregazione, 5=Vincoli(B_rules,entitaDeboli), 6=Ristrutturazione e strategie, 7=def attributi cardina 0

Join:

L'operatore join in algebra relazionale correla i dati contenuti in relazioni diverse confrontando i valori. Esistono due tipi di join: il join naturale e il join condizionale (theta 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:

Il 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. E' 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.

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

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 outer join***, ***right outer join***, e ***full outer join***,

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 (Conviene usarla quando si vogliono produrre anche i valori nulli)

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.

Il risultato di una query *full outer join* per le tabelle A e B contiene sempre tutti i record della tabella di sinistra ("left") A, estraendo dalla tabella di destra ("right") B solamente le righe che trovano corrispondenza nella regola di confronto della join inoltre verranno estratti tutti i record della tabella di sinistra ("left") A che non trovano corrispondenza nella tabella di destra ("right") B impostando a NULL i valori di tutte le colonne della tabella B e tutti i record della tabella di destra ("right") B che non trovano corrispondenza nella tabella di sinistra ("left") A impostando a NULL i valori di tutte le colonne della tabella A.

Una forma normale

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 sotto insiemi (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
Seconda Forma Normale (2NF)
Per evitare le anomalie si può introdurre la
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)

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

Ma per evitare altre anomalie possiamo affidarci anche alla
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 si definisce come segue:
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.
Differenza con la Boyce codde la terza forma normale ammette relazioni con alcune anomalie ed e sempre raggiungibile

Algebra:

Cos'è l'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 la pi 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.
P 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.**
$A / B = \{ < x > \mid \forall < y > \in B , < x , y > \in A\}$

Funzioni di aggregazione

Le funzioni di aggregazione sono particolari funzioni che operano su più righe.

Le funzioni più comuni sono:

COUNT: per effettuare conteggi nel gruppo;
SUM :per le somme;
MAX e MIN per il massimo e il minimo;
AVG :per calcolare la media.

La funzione **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.

La funzione **SUM**
Questa funzione somma i valori dei campi trovati nel gruppo. I valori nulli vengono ignorati.

Le funzioni **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.

Filtraggio sul raggruppamento (having)
A differenza di WHERE, che agisce a livello di singola riga, la parola chiave **HAVING** permette di effettuare un filtraggio sul raggruppamento. Questa clausola si inserisce subito dopo la GROUP BY. Il criterio di filtraggio può contenere qualsiasi funzione di raggruppamento.

GROUP BY di SQL è utilizzata per raggruppare i valori identici presenti in una o più colonne.
raggruppa le righe che hanno gli stessi valori in righe di riepilogo

Subquery è una select all interno di un altra istruzione

La where effettua il filtraggio solo sulla attributo mentre
La having effettua il filtraggio sulle funzioni di aggregazione.

Vincoli di integrità

I vincoli di integrità servono per mantenere la consistenza della base di dati. Infatti, non tutte le istanze della base di dati sono lecite. Sono ammissibili solo quelle che soddisfano tutti i vincoli di integrità specificati per la base di dati.

Questi sono:

Vincoli di dominio:

specificano che un attributo associato ad un certo dominio deve assumere valori in quel dominio. Ad esempio, l'inconsistenza numero 2 viola tale vincolo.

Vincoli di chiave:

specificano che una chiave deve avere valori univoci e una chiave primaria deve avere valori univoci e non nulli. Ad esempio, le inconsistenze numero 1 e 3 violano questi vincoli.

Vincoli di chiave esterna:

specificano che i valori *non nulli* di della chiave esterna devono corrispondere a valori della chiave riferita. Si noti che il vincolo deve essere soddisfatto solo per il valori non nulli. Inoltre, la chiave riferita dalla chiave esterna può essere primaria o candidata. Ad esempio, le inconsistenze numero 5 e 6 violano questi vincoli.

Vincolo intrarelazionale, Vincolo su valori o dominio, Vincolo di n-pla, Vincolo interrelazionale

Ulteriori vincoli di integrità possono essere specificate mediante le **regole aziendali** contenute nello schema concettuale.

Business Rules o Regole Aziendali

Sono uno degli strumenti più usati dagli analisti di sistemi informativi come ad es: un impiegato non può guadagnare più del proprio datore di lavoro questo contribuisce alle business rules.

1) descrizione di un concetto rilevante per l applicazione **ovvero la definizione precisa di un entità, di un attributo, o di una relazione del modello E-R**

2)un vincolo di integrità sui dati dell'applicazione, sia esso la documentazione di un vincolo espresso con qualche costrutto del modello E-R(p**er esempio la cardinalità di una relazione**) o la descrizione di un vincolo non esprimibile direttamente con i costrutti del modello;

3)Una Derivazione, **ovvero un concetto può essere ottenuto, attraverso un' inferenza o un calcolo aritmetico**, da altri concetti dello schema(per esempio un attributo Costo il cui valore pro essere ottenuto dalla somma degli attributi Costo Netto e Tasse)

Business rules

Regola del particolare dominio applicativo

-descrizione di un concetto (descrittiva)

-vincolo di integrità (non descrittiva)

-derivazione (non descrittiva)

VINCOLI DI INTEGRITA

Osservazione atomiche dichiarative

- <concetto> deve/non deve <espressione su concetti>

Es. Il direttore di un dipartimento deve afferire a tale dipartimento

DERIVAZIONE

<concetto> si ottiene <operazione su concetti>

Tecniche di documentazione

Business rule descrittive => dizionario dei dati

Tabella delle entita dello schema e tabella delle relazioni

Business Rules non descrittiva =>

Tabella delle regole di vincolo e tabella delle regole di derivazione

Entità deboli

Le **entità deboli** sono delle entità che non hanno nessun insieme di attributi che sia sufficiente a identificare univocamente le singole istanze di quell'entità e quindi non c'è nessun insieme di attributi candidabile come come chiave.
In questi casi **l'entità debole è di norma collegata a un'altra entità non debole** (ovvero forte) tramite un'associazione uno a uno, in questi casi la chiave dell'entità debole sarà formata da una chiave composta dagli attributi della chiave dell'entità forte (**chiave esterna**) più eventualmente uno o più attributi dell'entità debole (talvolta chiamati chiave parziale).
Un'altra possibilità per avere una chiave nelle entità deboli è quella di aggiungervi un ulteriore attributo (generalmente un identificatore incrementale) che abbia le caratteristiche per poter essere una chiave.

Ristrutturazione di schemi E-A

Una serie di passi da effettuare in sequenza:
1. Eliminazione degli attributi multivalore
2. Eliminazione degli attributi composti
3. Eliminazione delle generalizzazioni
4. Scelta degli identificatori primari (per quelle entità che ne hanno più di uno)

Ogni **attributo composto** viene sostituiti con gli attributi componenti

È necessario trasformare le generalizzazioni in entità e/o associazioni
Tre metodi:
1. Accorpamento del genitore nei figli
2. Accorpamento dei figli nel genitore
3. Sostituzione della generalizzazione con associazioni

1. Accorpamento del genitore nei figli:

Per l'ereditarietà, attributi e associazioni del genitore vanno aggiunti a tutti i figli
Possibile solo se la generalizzazione è totale ed esclusiva

2. Accorpamento dei figli nel genitore:

Si aggiunge al genitore un attributo che indichi il "tipo" dell'individuo (a quale entità figlia eventualmente appartiene)
Si aggiungono al padre gli attributi e le associazioni dei figli, che diventano opzionali
Sempre applicabile, ma comporta la presenza di attributi e associazioni opzionali e l'aggiunta di vincoli

3. Sostituzione della generalizzazione con associazioni:

Si aggiungono nuove associazioni che rappresenta la generalizzazione
I figli partecipano obbligatoriamente all'associazione, il genitore opzionalmente
L'alternativa 1 conviene quando le operazione non fanno molta distinzione tra le occorrenze e tra gli attributi di E0, E1, ed E2.
L'alternativa 2 è applicabile quando la generalizzazione è totale.
L'alternativa 3 è applicabile quando la generalizzazione non è totale, e ci sono operazioni che fanno distinzione tra entità padre ed entità figlie.

Strategie di Progettazione

Per affrontare progetti complessi è opportuno adottare uno specifico modo di procedere, ovvero una strategia di progettazione
I casi notevoli sono:
Strategia **top-down**:
Si parte da uno schema iniziale molto astratto ma completo, che viene successivamente raffinato fino ad arrivare allo schema finale
Strategia **bottom-up**
Si suddividono le specifiche in modo da sviluppare semplici schemi parziali ma dettagliati, che poi vengono integrati tra loro
Strategia **inside-out**
Lo schema si sviluppa "amacchiad'olio", partendo dai concetti più importanti, aggiungendo quelli ad essi correlati, e così via.

Definizioni di attributi

Attributo Multivalore: attributo che può avere per ogni istanza dell'entità associata più valori
Si trasforma facendo diventare un entita l attributo e mettendoci la relazione
Attributo Composto: costituito da più attributi correlati e si gestisce collegando gli attributi atomici che lo compongono direttamente all entità
Attributo Opzionale: è un attributo che può non avere un valore
Attributo Derivabile: è un attributo i cui valori sono calcolabili da altri attributi della stessa entità o da attributi di altre entità o relazioni. Si gestisce facendo l analisi delle ridondanze e decidendo se tenerlo o eliminarlo.

La cardinalizia minima 0, si gestisce come una cardinalità 1,1 con asterisco
R(0,1)(0,1) E1(a, b, c), E2(g, h, i) = E1(a, b, c, g*) E2 (g, h, i) / E1(a,b,c),E2(g,h,i)R(a,g)

Differenza tra where e having
Where effete un filtraggio o una condizione atomici sulle tuple mentre l having effettua anche essa un filtraggio o una condizione ma su una funzione aggregata es sum count ecc.
La clausola GROUP BY serve a specificare quali sono i campi su cui effettuare i raggruppamenti: il motore di query, per ogni riga esaminerà tali campi e la classificherà nel gruppo corrispondente.

Regole di derivazione
(1,1)(x,n) la chiave primaria della n viene portata dentro 1 a 1 come chiave sterna e se c è un attributo sulla relazione la 1a1 prende pure quella
(1,1)(1,1) la chiave di una delle due passa nell altra
(1,1)(0,1) 1 a 1 contiene 0 a 1
(0,1)(0,1) fai come se fosse una normale 1 a 1 ma l attributo che prendi si mette l isterico
(N,n) la relazione prende le chiavi delle due relazioni
(1,n) con attributo sulla relazione porti la chiave di n in 1 e porti anche l attributo della relazione.

Un sistema di gestione di base di dati, abbreviato in DBMS, costituisce insieme a una base di dati un sistema di database ,è un software che definisce il modello di un sistema di database
Il DBMS può essere suddiviso in tre componenti principali: il dizionario dei dati, il linguaggio di definizione dati e il linguaggio di manipolazione dati.

- **Dizionario dei dati**: il dizionario dei dati (Data Dictionary) consiste in una raccolta di metadati che c**ontengono informazioni sul contenuto dei vari dati presenti nell'archivio**. Essi forniscono informazioni anche in merito all'autorizzazione e all'uso dei set di dati e alla loro rappresentazione fisica. In parole povere, il dizionario contiene tutte le informazioni rilevanti sui dati memorizzati in un database.
- Linguaggio di definizione dati: il linguaggio di definizione dati, in inglese Data Definition Language (DDL), **è progettato per strutturare il contenuto di un database**. Singoli oggetti come riferimenti, relazioni o diritti dell'utente possono essere modificati, cancellati e creati utilizzando tale linguaggio.
- Linguaggio di manipolazione dati: questo linguaggio, in inglese Data Manipulation Language (DML), **si usa per cancellare, inserire, modificare e leggere i dati contenuti in un database**. Inoltre, permette di comprimere ed estrarre i dati.

Esistono diversi modelli che si differenziano principalmente per la strutturazione dei dati. Scegliere un DBMS equivale quindi a definire un modello di database specifico. Sono disponibili i seguenti modelli di database:

- Relazionale
- Gerarchico
- Orientato alla rete
- Orientato agli oggetti
- Orientati ai documenti

Tuttavia, il DBMS, come qualsiasi altro software, presenta anche diversi punti deboli, come dimostra chiaramente il seguente elenco.
Vantaggi di un sistema di gestione di base di dati:

- semplice amministrazione di grandi set di dati
- accesso semplice ed efficace ai dati memorizzati
- elevata flessibilità
- integrità e coerenza dei dati
- controllo degli accessi per gli utenti (sicurezza e protezione dei dati)
- alta disponibilità

Svantaggi di un sistema di gestione di base di dati:

- investimento iniziale relativamente costoso (inclusi i costi aggiuntivi per l'hardware)
- per software speciali piuttosto meno efficienti
- dipendenti qualificati necessari (amministratori di database)
- maggiore vulnerabilità a causa della centralizzazione dei dati

Progettazione di un database
Progettazione concettuale, logica e fisica
Progettazione concettuale: permette di rappresentare i dati in modo indipendente da ogni sistema
• cercano di descrivere i concetti del mondo reale
• sono utilizzati nelle fasi preliminari di progettazione
il più noto è il modello Entity-Relationship
Disegno concettuale
schema concettuale: descrizione ad alto livello della struttura del database indipendente dal dbms
Modello Concettuale: linguaggio usato per descrivere lo schema concettuale
Disegno Logico: schema logico e modello logico
Schema logico: descrizione della struttura di un database elaborato dal software DBMS
Modello Logico: linguaggio usato per specificare uno schema logico

Progettazione concettuale in contemporanea
Fase di raccolta e analisi dei requisiti
Fase di ristrutturazione
Fase Logica
Fase Fisica

Il ciclo di vita di un sistema informativo è formato da
Studio di fattibilità
Raccolta e analisi dei requisiti
Progettazione
Implementazione
Validazione e collaudo
Funzionamento

- Studio di fattibilità. Serve a definire, in maniera per quanto possibile precisa, i costi delle varie alternative possibili e a stabilire le priorità di realizzazione delle varie componenti del sistema.
- Raccolta e analisi dei requisiti. Consiste nell'individuazione e nello studio delle proprietà e delle funzionalità che il sistema informativo dovrà avere. Questa fase richiede un'interazione con gli utenti del sistema e produce una descrizione completa, ma generalmente informale, dei dati coinvolti (anche in termini di previsione sul carico applicativo) e delle operazione su di essi (anche in termini di previsione sulla loro frequenza). Vengono inoltre stabiliti i requisiti software e hardware del sistema informativo
- Progettazione. Si divide generalmente in progettazione dei dati e progettazione delle applicazioni. Nella prima si individua la struttura e l'organizzazione che i dati dovranno avere, nell'altra si definiscono le caratteristiche dei programmi applicativi. Le due attività sono complementari e possono procedere in parallelo o in cascata.
  Le descrizioni dei dati e delle applicazioni prodotte in questa fase sono formali e fanno riferimento a specifici modelli.
- Implementazione. Consiste nella realizzazione del sistema informativo secondo la struttura e le caratteristiche definite nella fase di progettazione. Viene costruita e popolata la base di dati e viene prodotto il codice dei programmi.
- Validazione e collaudo. Serve a verificare il corretto funzionamento e la qualità del sistema informativo. La sperimentazione deve prevedere, per quanto possibile. tutte le condizioni operative.
- Funzionamento. In questa fase il sistema informativo diventa operativo ed esegue i compiti per i quali era stato originariamente progettato. Se non si

verificano malfunzionamenti o revisioni delle funzionalità del sistema, questa attività richiede solo operazioni di gestione e manutenzione.

Progettazione concettuale: il suo scopo è quello di rappresentare le specifiche informali della realtà di interesse in termini di una descrizione formale e compie a ma indipendente dai criteri di rappresentazione utilizzati nei sistemi di gestione di basi di dati. Il prodotto di questa fase viene chiamato schema concettuale e f riferimento a un nodello concettuale dei dati Come abbiamo accennato nel Paragrafo 1.3. i modelli concettuali ci consentono di descrivere l'organizzazione dei vi. In questa fase infatti, il progettista deve cercare di rappresentare il connesso informativo della base di dati, senza preoccuparsi né delle modalità con le quali queste informazioni verranno codificate in un sistema reale, né dell'efficienza dei programmi che faranno uso di queste informazioni.

Fase di progettazione
Fase di progettazione concettuale:
A partire dai requisiti informativi viene creato uno schema concettuale cioè una descrizione formalizzata e contagiata dalle esigenze aziendali esprime in modo indipendente dal DBMS adottato.
A tale scopo si adotta un modello concettuale che permette di fruire descrizioni ad alto livello indipendentemente dal implementazione
Lo schema concettuale è indipendente anche dal tipo di DBMS che sarà utilizzato (relazionale, gerarchico etc.).

Fase di progettazione logica
Consiste nella traduzione dello schema concettuale nel modello di dati del DBMS
Il risultato è uno schema logico espresso nel Del del DBMS
In Questa fase si considerano anche aspetti legati a integrità consistenza (vincoli)
La progettazione logica si articola in due sotto-fasi
-ristrutturazione dello schema concettuale
-traduzione il modello logico

Fase di progettazione Fisica

Modello Gerarchico relazionale e reticolare
Gerarchico e reticolare utilizzano riferimenti espliciti(puntatori)fra record.
Il modello relazionale invece è basato su valori, anche riferimento su dati e strutture diverse(relazioni) sono rappresentati per mezzo dei valori stessi.
(Cioè lavora direttamente sulle entità e le relazioni)
Quanti tipi di valori nulli?
Senza informazione = non si hanno informazioni per quel valore
Inesistente = un valore che non si ha all interno del database
Sconosciuto = hai un valore ma non sai da dove deriva

Per conoscere il valore massimo tavole dei volumi
Per conoscere il caso medio Tavola degli accessi

Definizione di mapping si rappresenta un istanza

DATABASE CICLICI
Nel caso di database ciclici occorre prestare **particolare attenzione** è infatti necessario prevedere **una condizione di stop altrimenti il rischio è la non terminazione**!!!
Occorre inserire nella subquery ricorsiva un predicato che prima o poi, sia falso per tutte le nuove tuple che si andrebbero ad aggiungere alla vista
Casi tipici: 1) limitazione nella lunghezza massima di percorsi, 2) limitazione sul "costo"

Una subquery non è altro che una SELECT all'interno di un'altra istruzione.
Gli operatori di confronto si possono usare solo se la Subquery restituisce non più di una tripla (subquery scalare)
Sì la Subquery può restituire più di un valore si dice si possono usare le forme

Vincoli intrarelazionali.

All'interno delle tabelle si utilizzano i vincoli intarelazionali, chiamati anche constraints(costrizioni), per verificare l'input dei dati di una tabella. Questi vincoli coinvolgono una sola relazione e sono:

-**NOT NULL** e significa che l'attributo non può essere lasciato vuoto
-**UNIQUE** che si utilizzano per garantire che non vengano immessi valori duplicati in colonne specifiche che non fanno parte di una chiave primaria.
A differernza della PRIMARY KEY, il vincolo UNIQUE può essere applicato a più campi nella stessa tabella.

-**PRIMARY KEY** che definisce uno o più campi i cui valori identificano in modo univoco ciascun record della tabella e deve sempre essere di tipo NOT NULL.
Vincoli interrelazionali.

Il vincolo interrelazionale

rappresenta un vincolo tra due dati presenti in due tabelle diverse; si tratta di una chiave primaria (PRIMARY KEY) e di una chiave esterna (FOREIGN KEY).
ESEMPIO: vogliamo relazionare le tabelle voti e alunni in modo tale che a ogni record immesso nella tabella voti corrisponda un allievo della tabella alunni.
In tal caso si deve usare un vincolo interrelazionale, che permette appunto di verificare che la tabella alunni possieda effettivamente quel particolare alunno.
Ciò consente di verificare l'esistenza di un elemento della tabella correlata e prende il nome di "integrità referenziale"→si intende l'insieme dei vincoli interrelazioni tesi ad assicurare che le relazioni, tra i record delle tabelle correlate, siano valide e che i dati tra loro collegati non vengano eliminati o modificati per errore.

VINCOLO DI TUPLA

Un vincolo di tupla o record è un vincolo che può essere valutato su ciascuna tupla indipendentemente dalle altre
Un esempio è il vincolo *not NULL*, usato per dichiarare che un dato attributo non può assumere il valore nullo.