1.**ASSIOMI DI ARMSTRONG**

Sono una serie di *regole di inferenza* per le dipendenze funzionali:

**-Riflessività: se y <= x, allora x→ y**

Questa regola afferma che un insieme di attributi determina sempre se stesso o uno qualsiasi dei suoi sottoinsiemi.

**-Arricchimento: se x→ y , w<= t, allora xw → yw**

Questa regola sostiene che aggiungendo lo stesso insieme di attributi alla parte sinistra e alla parte destra di una dipendenza si ottiene un’altra dipendenza valida.

**-Transitività: se x → y , y→ z, allora x→ z**

Secondo questa regola le dipendenze funzionali sono transitive.

**2. QUERY FOR ALL**

Dato che il linguaggio SQL, non esiste un modo per esprimere il “per ogni”, modifichiamo opportunamente il costrutto EXISTS

La condizione è vera solo se esiste almeno una tupla che la verifica. Spesso il risultato di una query dipende dall’esistenza di certe righe in altre tabelle.

∀ x p ↔ ∄ x p:

Per ogni x ↔ Non esiste x solo per qualcuno

***Esempio:***

***Voglio Tutti gli studenti che hanno fatto la stessa carriera di Mario Rossi. ↔ Voglio gli studenti per i quali non esiste nessun esame diverso da Mario Rossi.***

***3.* CONCORRENZA IN SQL**

Una delle funzionalità di un DBMS SQL è di consentire l' esecuzione *CONCORRENTE* di più transazioni , evitando interferenze quando esse lavorano sugli stessi dati.

Un classico esempio di concorrenza è quello in cui , come conseguenza, si ha una perdita di modifiche . Un modo semplice per risolvere le interferenze è che per ogni coppia di transazioni Ti, Ts, le azioni di Ti, precedono quelle di Ts.

4. **DIFFERENZA TRA I VARI JOIN**

II JOIN è un operatore di giunzione che consente di unire in un' unica tupla ,il contenuto di più tabelle.

*Tipi di JOIN:*

**EQUI - JOIN** : presente nella condizione di John , l'operatore di uguaglianza

Formalmente: R ⨝ S , Ai = Bj

**THETA - JOIN** : Presenta un operatore di controllo

Formalmente: R ⨝ S , Ai Ѳ Bj

**NATURAL – JOIN**: Consente l' eliminazione di uno dei due attributi uguali , quando uniamo le tabelle .

Formalmente: R\*S

**LEFT-RIGHT JOIN** : Viene proiettata rispettivamente le tabelle di sinistra o destra .

**5. SPECIALIZZAZIONE CONCETTUALE – RELAZIONALE**

Per tradurre una specializzazione possiamo optare 3 modi diversi :

**1. TABELLA UNICA** in cui inseriamo tutti i dati di entrambi le sottoclassi , ma potremmo incorrere in uno spreco dei dati nel caso le sottoclassi siano disgiunte

**2. PARTIZIONAMENTO VERTICALE** si crea una tabella per ogni entità ( classe madre + 2 sottoclassi) , con chiave esterna comune alle sottoclassi (che funge anche da chiave primaria . SVANTAGGIO : sgretolamento dei dati

**3. PARTIZIONAMENTO ORIZZONTALE** si creano sempre tante Tabelle quanti sono le entità , ma non si usano chiavi esterne • SVANTAGGIO : Non si può riconoscere che le sottoclassi provengono dalla classe madre . In più non è applicabile quando c'E una relazione che coinvolge la stessa madre.

**6. CHIAVI**

Una chiave identifica univocamente l' entità d'appartenenza.Ogni entità ha uno o piu attributi principali .

Un' insieme di attributi costituisce una *SUPERCHIAVE* se eliminando un attributo da questo insieme si perde l'univocità.

**Esempio:**

**Superchiave → matricola-cognome, eliminando matricola si perderebbe l’univocità.**

Può capitare che esistono più chiavi per un entità , che prendono il nome di CHIAVI CANDIDATE . Scegliamo queste in base a diversi criteri.

ES. anche se un attributo può essere una chiave , potrebbe essere opzionale , quindi sicuramente non sceglieremo quello .

**7.ENTITÀ DEBOLE**

Molto spesso può capitare che una chiave non basti per identificare un entità , quindi le uniamo a quella dell' entità associata , andando a creare un entità debole dotata di chiave debole ( chiave + chiave entità associata).

**8. TIPO UNIONE**

Un tipo unione si ha quando più sottoclassi si uniscono a fare parte di unica classe : *SUPERCLASSE*

***9. ASSERZIONE***

Le asserzioni permettono di specificare dei vincoli . In pratica permettono all' utente di annullare un operazione che causa violazione.

CREATE ASSERTION <nome>

<vincolo>

*Il vincolo è soddisfatto se nessuna combinazione di Tuple viola il vincolo .*

**10 . NORMALIZZAZIONE**

Consiste in una serie di test che facciamo al nostro DB affinche vengano eliminate ridondanze e dipendenze funzionali anomale .

**-1NF :** Si vogliono evitare attributi multi -valore , usando attributi atomici e singoli quindi non scomponibili

**-2NF :**SI verifica che in caso di chiavi composte , ogni attributo non primo , dipende per forza da entrambe le chiavi primarie .

Quindi si basa sul concetto di dipendenza funzionale ( ✗→ Y se rimuovendo qualsiasi attributo da x, la dipendenza non sussiste più )

**-3NF:** Gestisce le dipendenze funzionali anomale, ossia quando un attributo non prima dipende da un altro attributo non primo.

*ESEMPIO : Data \_ nascita→ età [non primi] DIPENDENZA FUNZIONALE ANOMALA.*

**11 . CONFLITTO DI IMPEDENZA**

È l' espressione usata per indicare problemi che si verificano a causa delle differenze tra il modello del DB e il modello del linguaggio di programmazione

ES . EMBEDDED SQL e API soffrono di tale conflitto poiché in essi l ' SQL non è NATIVO.

**12. DIFFERENZE TRA PROCEDURE E TRIGGER**

**- Le procedure:**

vengono dichiarate permanentemente bel DB , e sono blocchi di codice che non restituiscono alcun valore , ma automatizzano alcune operazioni del DB come l' inserimento o le modifica di tuple.

**- I Trigger:**

sono blocchi di codice che si attivano in occasioni di alcuni eventi come BEFORE INSERT all interno di essi è descritta una condizione e sono nel caso essa si verifica , l' evento termine correttamente.

**13 . CARDINALITÀ DELLE RELAZIONI**

Indicano il numero minimo e massimo delle occorrenze di un entità in un associazione.

**14.TRADUZIONE IN RELAZIONALE DELLE ASSOCIAZIONI**

◦ 1 - N : la chiave esterna dell' entità 1 , va su N .

◦ 1 - 1 : E preferibile inserire la chiave esterna dove c' è la totalità.

◦ M -N : si crea una nuova tabella che contiene le chiavi di entrambe le associazioni.

**15.VINCOLI STATICI E DINAMICI**

Rispettare i vincoli vuol dire conferire integrità al nostro DB

- STATICI → valgono in ogni istante di vita del DB

- DINAMICI → mutano in base all' evoluzione del DB

**16 . POLITICHE DI REAZIONE**

Usando chiavi esterne si creano problemi circa l'aggiornamento e eliminazione della Tabelle con chiave primaria . Per questo vengono adottate POLITICHE DI REAZIONE

Alcuni esempi :

ON DELETE CASCADE ,

ON DELETE SET NULL ,

ON UPDATE CASCADE,

ON UPDATE SET NULL.

**17. A. C. I . D .**

-Atomica → Ogni transazione deve essere atomica

-Coerente → Una transazione deve far rimanere valido il database , che dopo di esse deve comunque rispettare tutti i vincoli e Trigger

- Isolamento → Garantisce che l'esecuzione simultanea delle operazioni , lasci il DB nello stesso stato In cui il DB sarebbe stato , se le azioni fossero state eseguite in sequenza.

- Durabilità → il DB deve persistere nel tempo con le sue azioni .

**18 . DIFFERENZA TRA DATO E INFORMAZIONE**

Il dato è un insieme di bit . Di per sé , il dato non ha alcun senso , ma se associato ad un contesto , assume un significato , ossia l' Informazione , quindi:

DATO + CONTESTO = INFORMAZIONE

Una base di dati , è un insieme di dati che nasce in un contesto specificato il fine di uno scopo.

**19 . DIPENDENZE FUNZIONALI**

È un vincolo tra due insiemi di attributi nel database Il vincolo è che per ogni coppia di Tuple T1, T2 in R allora: T1 [x] - T2lx] → T1 [y] - T2[y]

ESEMPIO : Matricole 1 = Matricole 2 → Nome1 = Nome2

***Nome dipende funzionalmente da Matricola .***

**20 . SCHEMI RELAZIONALI**

Il modello relazionale è uno schema logico che si basa sulla teoria degli insiemi , e visualizza le entità del diagramma per come dovranno apparire nel DB.

**21 . TRADUZIONE CONCETTUALE - RELAZIONALE**

Trasformiamo le nostre entità in tabelle che contengono gli attributi a cui si aggiungono i tipi dell' SQL . le associazioni avverranno attraverso le chiavi esterne con le opportune regole da seguire per le cardinalità.