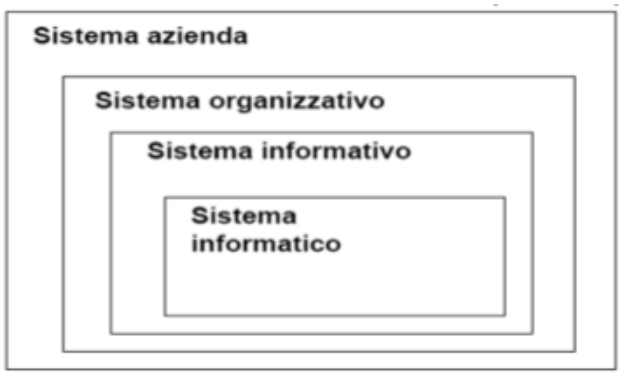
Riassunto

giovedì 30 giugno 2016 11:03

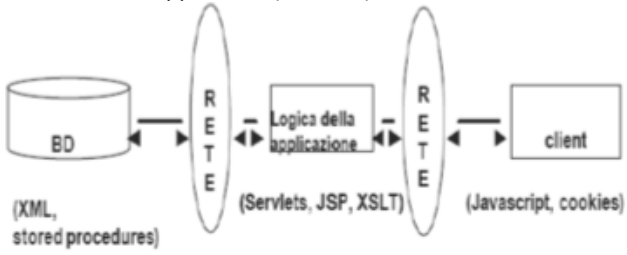
1. Introduzione

- Sistema informativo: componente del sistema organizzativo che **acquisisce, elabora, conserva, produce le informazioni di interesse**. Costituito da **risorse e regole** per lo svolgimento coordinato di attività (processi) per perseguire gli scopri propri di un'organizzazione.

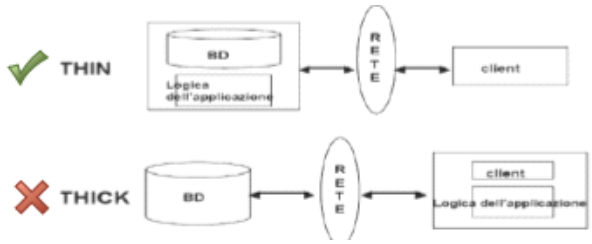


- Base di dati: **insieme organizzato di dati** utilizzati per rappresentare le informazioni di interesse per un **sistema informativo**. È gestito da un DBMS
  - DBMS, garantisce:
    - privatezza
      - Autorizzazioni agli utenti
    - affidabilità
      - Resistenza hw e sw
    - efficienza
      - In memoria
      - Tempo di esecuzione e risposta
    - efficacia
      - Funzionalità articolate, potenti e flessibili
      - DB ben progettato
- Transazione: insieme di operazioni elementari sulla base di dati da considerare indivisibile.
  - Atomicità: sequenza di operazioni che viene eseguita per intero o per niente.
- Serializzabilità: l'effetto dell'esecuzione di transazioni concorrenti dev'essere equivalente alla loro esecuzione in sequenza.
- Commit: conclusione positiva di una transazione.
- Architetture
  - Single tier

  - Two tier (client-server)
  - Three tier
    Secondo server applicativo (internet)



- Struttura del client



- Scalare
  - orizzontalmente: aggiunta di cloud servers
  - verticalmente: migliorare hardware di un singolo cloud server
- NoSQL
  - Auto-sharding (auto-ridistrubuzione) senza perdere di efficienza nell'esecuzione di query complesse
  - Replicazione estesa di nodi anche a caldo (scaling orizzontale)
  - Impossibile attuare controlli sull'integrità per garantire consistenza
  - Non esiste uno standard per memorizzazione e accesso ai dati

2. Modello relazionale

- Schema di relazione: insieme di attributi R(A1, …, An)
  - Istanza di relazione su uno schema R(X): insieme r di ennuple su X
- Schema di base di dati: insieme di schemi di relazione R = {R1(X1), …, Rk(Xk)}
  - Istanza di base di dati su uno schema R = {R1(X1), …, Rk(Xk)}: insieme di relazioni r = {r1, …, rn}
- Ennupla su un insieme di attributi X: funzione che associa a ciascun attributo A in X un valore del dominio di A
- t[A]: valore della ennupla t sull'attributo A

- Ennupla = riga
- Attributo = colonna
  - Dominio = tipo di colonna (INT, CHAR, …)

```
CREATE DOMAIN Voto
  AS SMALLINT DEFAULT NULL
  CHECK ( value >=18 AND value <= 30 )
```

- Tabella = relazione
  - Una tabella rappresenta una relazione se:
    - I valori di ogni colonna sono fra loro omogenei
    - Le righe sono diverse
    - Le intestazioni sono diverse

RELAZIONE
Schema
Istanza

```
CREATE TABLE Impiegato(
  Matricola CHAR(6) PRIMARY KEY,
  Nome CHAR(20) NOT NULL,
  Cognome CHAR(20) NOT NULL,
  Dipart CHAR(15),
  Stipendio NUMERIC(9) DEFAULT 0,
  FOREIGN KEY(Dipart) REFERENCES "fa riferimento"
      Dipartimento(NomeDip),
  UNIQUE (Cognome,Nome) non ci sono omonimi
)
```

Chiavi (identificazione ennupla)

- **Superchiave**: insieme di attributi (anche ridondante) che identificano univocamente una ennupla
  - **Chiave** (***canditate key***, **UNIQUE KEY**): superchiave minimale (da cui non si può togliere niente)
    In presenza di valori nulli, non è possibile realizzare riferimenti con altre relazioni
    - **Chiave primaria**: non sono ammessi valori nulli (notazione sottolineata)

Conseguenza

- Non esistono due ennuple (righe) uguali
  - Quindi ogni relazione (tabella) ha come superchiave l'insieme degli attributi (colonne) su cui è definita

Vincoli

- **Dipendenza funzionale** (il valore di un insieme di attributi, implica il valore di un insieme di altri attributi)
  Date due insiemi di attributi X e Y, si dice che X determina Y, oppure Y dipende funzionalmente da X (**X -> Y**), se e solo se **t1[X] = t2[X] => t1[Y] = t2[Y]**

Impiegato	Stipendio	Progetto	Bilancio	Funzione
Rossi	20 000	Marte	2000	tecnico
Verdi	35 000	Giove	15 000	progettista
Verdi	35 000	Venere	15 000	progettista
Neri	55 000	Venere	15 000	direttore
Neri	55 000	Giove	15 000	consulente
Neri	55 000	Marte	2000	consulente
Mori	48 000	Marte	2000	direttore
Mori	48 000	Venere	15 000	progettista
Bianchi	48 000	Venere	15 000	progettista
Bianchi	48 000	Giove	15 000	direttore

Consideriamo ancora la relazione in Figura 11.1. Abbiamo osservato che lo stipendio di ciascun impiegato è unico e quindi, ogni volta che in una tupla della relazione compare un certo impiegato, il valore del suo stipendio rimane sempre lo stesso. Possiamo cioè dire che il valore dell'attributo Impiegato *determina* il valore dell'attributo Stipendio o, in maniera più precisa, che esiste una funzione che associa a ogni elemento del dominio dell'attributo Impiegato che compare nella relazione, un solo elemento del dominio dell'attributo Stipendio. Un di-

- Vincoli di chiave

- Vincoli di integrità
  - Intrarelazionali: condizioni su valori di ciascuna ennupla
    - NOT NULL
    - UNIQUE definisce chiavi
    - PRIMARY KEY: chiave primaria (una sola, implica NOT NULL)
      - Di dominio: un solo attributo
        (Voto ≥ 18) AND (Voto ≤ 30)
        o su più domini:
        (Voto = 30) OR NOT (Lode = "e lode")
  - Interrelazionali = **Referenziali**
    Impone ai valori su un insieme di attributi della tabella (foreign key) diversi da NULL di comparire come valori della chiave primaria di un'altra tabella.

```
FOREIGN KEY(Provincia, Numero)
        REFERENCES Auto(Provincia, Numero)
        ON DELETE SET NULL    Provincia e numero vengono messi a NULL.
        ON UPDATE CASCADE )   Provincia e numero vengono aggiornati
```

    - Per valori nulli:

Integrità referenziale e valori nulli

Impiegati

Matricola	Cognome	Progetto
34321	Rossi	IDEA
53524	Neri	XYZ
64521	Verdi	NULL
73032	Bianchi	IDEA

Progetti

Codice	Inizio	Durata	Costo
IDEA	01/2000	36	200
XYZ	07/2001	24	120
BOH	09/2001	24	150

57

FOREIGN KEY(Progetto) REFERENCES Progetti(Codice)

    - Violazioni
      - Sulla tabella interna
        - Modificare l'attributo referente
        - Inserire una nuova riga (senza corrispettivo nell'altra tabella)

Infrazioni

Codice	Data	Vigile	Prov	Numero
34321	1/2/95	3987	MI	39548K
53524	4/3/95	3295	TO	E39548
64521	5/4/96	3295	PR	839548
73321	5/2/98	9345	PR	839548

Auto

Prov	Numero	Cognome	Nome
MI	E39548	Rossi	Mario
TO	F34268	Rossi	Mario
PR	839548	Neri	Luca

      - Sulla tabella esterna
        - Modificare l'attributo riferito
          Reazioni:

        - Cascade (propaga la modifica nella tab. interna)
        - Set null (pone null l'attributo referente, così non è obbligato a comparire nella tabella esterna)
        - Set default
    - Eliminare una ennupla (le foreign keys non compaiono più nella tabella esterna)
      Reazioni:
        - Cascade (elimina la relativa riga nella tab. interna)
        - Set null
        - Set default
        - No action

3. 4. 5. 6. SQL (temporaneamente saltati)

Creazione

Mostrata prima

Modifica

- ALTER/DROP DOMAIN/TABLE

- INSERT INTO Tabella[(Attributi)] VALUES(Valori)
  - Ordinamento significativo
  - Se la lista degli attributi non contiene tutti gli attributi della selezione, viene inserito NULL o un valore di default
- DELETE FROM Tabella [WHERE Condizione]
  - DELETE FROM Tabella: elimina tutte le tuple
  - DROP TABLE tabella: elimina la tabella
- UPDATE Tabella
  SET Attributo = < Espressione | SELECT … | NULL | DEFAULT >
  [WHERE Condizione]

Lettura

- LIKE: pattern matching su strunghe
  - _ singolo carattere non specificato
  - % gruppo di caratteri non specificato

7. Progetto (temporaneamente saltato)

- Entità: classe di oggetti
  - Occorrenza di entità = istanza di entità
- Relationship = relazione = correlazione = associazione: legame logico fra due o più entità
  - Occorrenza di relationship binaria: coppia di occorrenze di entità
- Attributo: proprietà elementare di un'entità o di una relationship

8. Modello logico (temporaneamente saltato)

9. Algebra relazionale

Data manipulation language: tipi

- Query
  - Procedurali
    Specificano le modalità di generazione del risultato
    - Algebra relazionale
  - Dichiarativi
    Specificano le proprietà del risultato
    - Calcolo relazionale
    - SQL (parzialmente)
- Update

Algebra relazionale (linguaggio procedurale)

- Algebra
  a. Dati
     - Relazioni
  b. Operatori:

- Su insiemi
  Le relazioni sono insiemi.
  Queste operazioni sono possibili tra relazioni con stessi attributi: i risultati sono relazioni.
  Si possono applicare solo tra relazioni definite sugli stessi attributi, in modo che il risujltato sia una relazione sugli stessi attributi.
  - Unione
  - Intersezione
  - Differenza
- Su relazioni
  - Ridenominazione *"AS, ON"*
    ρ_nuovo<-vecchio(Relazione)
  - Selezione *"WHERE"* ↔
    σ_expression(Relazione) = RelazioneRisultato
  - Proiezione *"SELECT DISTINCT"* ↕
    π_attributi(Relazione)
  - Join
    - Join naturale
      Relazione1 ▷◁ Relazione2
      Su attributi dello stesso nome
      - Join esterno
        Estende, con valori nulli, le tuple che verrebbero tagliate fuori da un join interno
        - Sinistro
          Impiegati $\triangleright\!\triangleleft_{\text{LEFT}}$ **Reparti**
          Mantiene le tuple del primo operando, estendendole con valori nulli
        - Destro
          Impiegati $\triangleright\!\triangleleft_{\text{RIGHT}}$ **Reparti**
        - Completo
          Impiegati $\triangleright\!\triangleleft_{\text{FULL}}$ **Reparti**
    - Prodotto cartesiano
    - Theta-join (join interno)
      $R_1 \triangleright\!\triangleleft_F R_2$
      Join su condizione, con attributi di nome diverso
      - Equi-join
        La condizione è di uguaglianza

Equivalenze

- $\sigma_{A=10}(R_1 \triangleright\!\triangleleft R_2) = R_1 \triangleright\!\triangleleft \sigma_{A=10}(R_2)$
- $\pi_{X_1 Y_2}(R_1 \triangleright\!\triangleleft R_2) = R_1 \triangleright\!\triangleleft \pi_{Y_2}(R_2)$

Relazioni derivate

Relazioni il cuo contenuto è funzione del contenuto di altre relazioni.

- Viste materializzate
  Relazione derivata memorizzata nel DB.
  - Immediatamente disponibili
  - Ridondanti
  - Appesantiscono gli update
  - Raramente supportate dai DBMS
- Viste [virtuali]
  Supervisione =
  $\pi_{\text{Impiegato, Capo}}(\text{Afferenza} \triangleright\!\triangleleft \text{Direzione})$
  Viene ricalcolata ad ogni query sulla lista
  - Non influiscono sull'efficienza
  - Supportate dai DBMS
  - Supervisione =
    $\pi_{\text{Impiegato, Capo}}(\text{Afferenza} \triangleright\!\triangleleft \text{Direzione})$

Estensioni

- Join esterno che permette di trattare valori null
- Proiezione generalizzata

$\pi_{Cliente,Credito-Spese}(Conto)$

- Operatori aggregati
  - $sum_{Spese}(Conto)$
  - $count_{Cliente}(Conto)$
  - $max_{Credito}(Conto)$
  - $count\text{-}distinct_{Cliente}(Conto)$
- Raggruppamento

  $Cliente\,\mathcal{G}_{sum(Credito)}(Conto)$

  SELECT SUM(Credito)
  FROM Conto
  GROUP BY Ciente

10. Calcolo relazionale

- Specifica la proprietà del risultato.
- Versioni:
  - Su domini

    $\{ A1: x1, \ldots, An: xn \mid f \}$
  - Su ennuple con dichiarazioni di range

    $\{ x1.Z1, \ldots, xn.Zn \mid xi(R1), \ldots, xj(Rm) \mid f \}$

11. Forme normali

- Schemi relazionali -> forma normale

Dipendenze funzionali

- ad ogni chiave K di R corrisponde una dipendenza funzionale in R da K verso tutti gli attributi della relazione
- Un gruppo di dipendenze funzionali (F) può implicare un'altra dipendenza funzionale

  $F \Rightarrow X \rightarrow Y$

  se per ogni istanza r di R che ver.ifica tutte le dipendenze funzionali in F, risulta verificata anche X -> Y
- **Chiusura** di un insieme di dipendenze funzionali (F): insieme di tutte le dipendenze funzionali implicate da F.

  $F^+ = \{ X \rightarrow Y \mid F \Rightarrow X \rightarrow Y \}$
- Istanza legale (r) di una relazione (R): se l'istanza r di R soddisfa F, allora soddisfa anche tutte le dipendenze funzionali logicamente implicate da F.
- Superchiave (X): se X -> Z (tutti gli attributi) è logicamente implicata da F, **cioè se X -> Z è in F+**.
  È chiave se è una superchiave minimale.
- **Regole di inferenza di Armstrong**

  Permettono di derivare tutte le dipendenze funzionali che sono implicate da un dato insieme iniziale, cioè F+.

  1. Riflessività: Se $Y \subseteq X$, allora $X \rightarrow Y$
  2. Additività (o espansione): Se $X \rightarrow Y$, allora $XZ \rightarrow YZ$, per qualunque Z
  3. Transitività: Se $X \rightarrow Y$ e $Y \rightarrow Z$, allora $X \rightarrow Z$

  Derivate:

4. Regola di unione
   $\{X \to Y, X \to Z\} \Rightarrow X \to YZ$
5. Regola di pseudotransitività (o aggiunta sinistra)
   $\{X \to Y, WY \to Z\} \Rightarrow XW \to Z$
6. Regola di decomposizione
   Se $Z \subseteq Y, X \to Y \Rightarrow X \to Z$

- Equivalenza: F e G sono equivalenti se F+=G+, ovvero, per ogni X –>Y ∈ F, deve essere X->Y ∈ G+ e, viceversa, per ogni X->Y ∈ G, deve essere X->Y ∈ F+
- X->Y ∈ F+, se e solo se Y ⊆ X+
  - Algoritmo per il calcolo di X+ (cioe' dell'insieme di attributi che dipendono da X)

    CalcolaChiusura(X,F)=
    { $X^+ = X$;
    Ripeti:
    – Fine = true;
    – Per tutte le FD in F = $\{V_i \to W_i\}$:
    – Se $V_i \subseteq X^+$ e $W_i \not\subset X^+$ allora: $\{X^+ = X^+ \cup Wi$; Fine = false$\}$
    Fino a che Fine = true o $X^+ = Z$
    }

    dove Z e' l'insieme di tutti gli attributi.
- Copertura minimale di F: insieme minimale di dipendenze funzionali senza le seguenti ridondanze:
  (pdf11 pag39)
  La forma standard prevede un singolo attributo sulla destra.
  - Algoritmo per la copertura minimale
    - Calcolo di M minimale per un insieme di FD F :
      – M = F
      – ogni FD del tipo $X \to \{A1, A2, \ldots, An\}$ è sostituita da $X \to A1, X \to A2, \ldots, X \to An$
      – ogni FD $X \to A$ in M, se, per ogni attributo $B \in X$, $\{\{M - \{X \to A\}\} \cup \{(X - \{B\}) \to A\}\} \equiv M$, è sostituita da $(X - \{B\}) \to A$
        - $A \subseteq (X - \{B\})^+$
    - ogni rimanente $X \to A$ in M, se $\{\{F - \{X \to A\}\} \equiv M$ è rimossa
      - $A \subseteq X^+$ anche in $\{\{F - \{X \to A\}\}$
- Proiezione di F su X

  CalcolaProiezione(F,X):=
  { *result* = ∅;
  per ogni sottoinsieme proprio S di X, per ogni attributo A in X tale che A non è in S, e tale che non esiste alcun sottoinsieme S' di S tale che $S' \to A$ è in *result*,
  if A è in CalcolaChiusura(S,F) allora *result* = *result* ∪ {$S \to A$};
  }

Forme normali

- **BCNF** (Boyce-Codd normal form): per ogni dipendenza funzionale (non banale) X->Y definita su di essa, X contiene una chiave K di r.
  - Avversaria della BCNF: dipendenza funzionale X->Y non banale, tale che X non e' una superchiave di R.
    Rende quindi una relazione non in BCNF.
    - Teorema dell'avversaria sconfitta: se F non ha avversarie, nemmeno F+ ne ha.
      Per verificare se una relazione e' BCNF, occorre verificare che tutte le dipendenze funzionali in F siano:
      - Banale, oppure
      - Tali da avere una superchiave come membro sinistro
  - Per portare una relazione in una BCNF, la si decompone sulla base delle dipendenze funzionali (al fine di separare i concetti).
    - Perdite
      - Perdita sul join: si perdono informazioni quando si fa il join sulle tabelle decomposte.

Non si hanno perdite sul join con il seguente algoritmo per la decomposizione.
    - Perdita di dipende\nze. Si evitano se ciascuna delle dipendenze funzionali dello schema orinigario coinvolge attributi che compaiono tutti insieme in uno degli schemi decomposti.
  - Algoritmo per la decomposizione

Assumiamo (senza perdita di generalità) che ogni volta che chiamiamo l'algoritmo descritto sotto ogni dipendenza funzionale in F abbia un unico attributo come membro destro, e che U sia l'insieme di tutti gli attributi di R

```
Decomponi(R,F):=
{ if esiste un'avversaria X → A di BCNF in F
     then  { sostituisci R con una relazione R1 con
             attributi U-A, ed una relazione R2 con
             attributi X∪A;
Decomponi(R1,F_{U-A});
Decomponi(R2,F_{X∪A})
           }
}
```

Quest'algoritmo produce:
    - Relazioni in BCNF
    - Assenza di perdita nel join
- **3NF** (Third normal form): per ogni dipendenza funzionale (non banale) X->Y definita su una relazione r, e' verificata almeno una delle seguenti condizioni:
  - X e' superchiave di r
  - Ogni attributo in Y e' contenuto in almeno una chiave di r

  Algoritmo di decomposizione:

- Dati R ed F minimale, si usa Decomponi(R,F) ottenendo gli schemi $R_1(X_1), R_2(X_2),\ldots, R_n(X_n)$ in BCFN ciascuno con dipendenze funzionali F_{X_i}
- Sia l'insieme N di dipendenze funzionali in F che non sono preservate in $R_1,R_2,\ldots,R_n$, cioè che non sono incluse nella chiusura dell'unione dei vari F_{X_i}
  - Per ogni dipendenza funzionale $X \rightarrow A$ in N, aggiungiamo lo schema relazionale X A alla decomposizione, con le corrispondenti dipendenze funzionali

12. Schema fisico (saltato temporaneamente)

13. Transazioni

- Properietà delle transazioni:
  - Atomicità e durabilità

    Assicurate dal gestore dell'affidabilità, che gestisce l'esecuzione dei comandi transazionali
    - B (begin)
    - C (commit)
    - A (aborto)
  - Consistenza
  - Isolamento
- Nel log viene salvato solo uno stato consistente, per cui:
  - Undo(undo(A)) = undo(A)
  - Redo(redo(A)) = redo(A)
- Un guasto prima di un commit, porta ad un undo di tutte le azioni.
- **Schedule**: sequenza di operazioni di input/output di transazioni concorrenti.
- Ogni schedule conflict-serializable e' anche view-serializable.

Algebra estesa

- Join esterno che tratta i valori null, per modellare info mancanti
- Proiezione generalizzata

  $\pi_{Cliente,Credito-Spese}(Conto)$
- Funzioni aggregate

- sum, count, min, max
  - sum_{Spese} (Conto)
  - $count_{Cliente}$ (Conto)
  - $max_{Credito}$ (Conto)
  - $count\text{-}distinct_{Cliente}$ (Conto)
- Raggruppamento

  $Cliente\,G_{sum(Credito)}(Conto)$

  SELECT SUM(Credito)
  FROM Conto
  GROUP BY Cliente)