Alaimo è potentissimo

Avrai bisogno di questo

Modello dei dati - Introduzione

- Un modello dei dati è un insieme di meccanismi di astrazione che definiscono una base di dati, ai quali sono associati operatori (che modificano la base di dati) e vincoli di integrità (che si applicano agli operatori)
- Questi meccanismi di astrazione rivestono un ruolo equivalente alle strutture dati nella programmazione (così come un algoritmo di sort può essere spiegato indipendentemente dall'implementazione specifica dell'array da ordinare, così gli operatori relazionali possono essere descritti indipendentemente dalla base di dati specifica).
- Di conseguenza, il modello relazionale dei dati è il più diffuso
  - Introdotto da E.F.Codd nel 1970
  - Implementa il concetto di dipendenza dei dati
  - Implementato nei DBMS commerciali dal 1981 in poi, ad oggi è il più diffuso
  - è basato sul concetto matematico di relazione

Modello dei dati relazionale - Relazione

- Con relazione, possiamo intendere:
  - Relazione matematica, ovvero quella della teoria degli insiemi
  - Relazione secondo il modello relazionale
  - Relazione Entità-Relazione, ovvero il concetto di associazione / correlazione
- Una relazione matematica potrebbe essere ad esempio, dati n insiemi non necessariamente distinti, D1, D2, … , Dn, definiamo una relazione matematica tra D1, D2, … , Dn come un sottoinsieme di D1 x D2 x … x Dn.
  - Quindi una relazione matematica è definita come l'insieme di tutte quelle n-uple ordinate (d1, d2, … , dn) tali che $d1 \in D1$, $d2 \in D2$, … , $dn \in Dn$.
  - Tutte le n-uple sono distinte (non ci sono duplicati)
  - Ogni n-upla è ordinata, ovvero l'n-upla (a,b,c) non è uguale ad (a,c,b), inoltre, a, b, c, provengono dai rispettivi domini
  - Tra di loro, le tuple non sono ordinate, ovvero non viene definito un ordinamento tra gli elementi della relazione

Modello dei dati relazionale - Posizionale e non posizionale

Juve	Lazio	3	1
Lazio	Milan	2	0
Juve	Roma	0	2
Roma	Milan	0	1

- Se prendessimo in considerazione la relazione “Partite”, definita come sottoinsieme di str x str x int x int (dove “str” corrisponde all’insieme di tutte le stringhe, “int” all’insieme dei numeri naturali, e “x” è l’operazione di prodotto cartesiano.
- Ogni dominio ha ruoli diversi, che distinguiamo attraverso la posizione dell’elemento nella tupla. Viene quindi definita una struttura posizionale.
- Se invece associassimo a ciascun dominio un nome unico nella tabella, che descriva il ruolo:

Casa	Fuori	RetiCasa	RetiFuori
Juve	Lazio	3	1
Lazio	Milan	2	0
Juve	Roma	0	2
Roma	Milan	0	1

- Questo è uno schema posizionale

Modello dei dati relazionale - schema di relazione

- Uno schema di relazione R : {T} è definito come una coppia *nome relazione (R)* / *tipo relazione (T)*
- Definiamo “tipi primitivi” quei tipi quali Interi, Reali, Booleani, Stringhe
- Dati T1, T2, …, Tn tipi primitivi, anche uguali e A1, A2, …, An etichette distinte, allora (A1:T1, A2:T2, …, An:Tn) è un tipo n-upla di grado n.
- Essenzialmente stiamo creando un nuovo tipo a partire dai tipi primitivi esistenti
- Dato un tipo n-upla T, chiamiamo {T} tipo relazione (ovvero l’insieme di n-uple T)
- Uno Schema relazionale è costituito da:
  - Un insieme di schemi di relazione R0:{T0}, R1:{T1}, …, Rn:{Tn}
  - Un insieme di vincoli di integrità relativi a tali schemi
- Uno schema relazionale è responsabile per l’aspetto intensionale del database, ovvero il significato di un dato.
- L’aspetto estensionale del dato è invece il valore stesso dello specifico dato
- Una Relazione R è quindi un insieme finito di n-uple di uno stesso tipo

Modello dei dati relazionale - Schema relazionale

- Data la relazione rappresentata dalla tabella:
- Essa è una relazione di tipo:

Nome	Matricola	Indirizzo	Telefono
Mario Rossi	123456	Via Etnea 18	777777
Maria Bianchi	234567	Via Roma 2	888888
Giovanni Verdi	345678	Via Etnea 18	999999
Enzo Gialli	456789	Via Catania 3	444444

- { (Nome : str) , (Matricola : int) , (Indirizzo : str) , (Telefono : int)}
- Dove ogni riga è una n-upla della relazione
- Indichiamo dom(Ai) il dominio dei valori dell'attributo Ai
  - Ad esempio, dom(Nome) potrebbero essere tutti i nomi registrati all'anagrafe, mentre dom(Telefono) potrebbe essere l'insieme dei numeri telefonici registrati
- Se chiamata questa relazione "studenti" volessimo definirla, potremmo omettere il tipo degli attributi qualora fosse facilmente evincibile dal nome stesso delle etichette, ad esempio:
  - Studenti (Nome, Matricola, Indirizzo, Telefono)

Modello dei dati relazionale - Vincoli di integrità

- I vincoli di integrità servono a migliorare la qualità delle informazioni contenute in una base di dati, oltre che ad assicurarsi che i valori contenuti in un certo dominio siano coerenti con l'aspetto intensivo della relazione.
- La differenza tra dominio e vincoli di integrità è che un vincolo di integrità può anche essere dipendente da diversi valori dell'n-upla:
  - Consideriamo EsitoEsame(Matricola, Corso, Materia, Voto, Lode)
  - Restringere il Voto tra 0 e 30 ad esempio potrebbe essere interpretato come un semplice cambio di dominio, ma un vincolo di integrità necessario è ad esempio che Lode può essere True solo se Voto == 30.
  - Oppure che Matricola deve fare parte delle matricole registrate ad un certo Corso, e che Materia faccia parte delle materie offerte da un certo Corsoù
- Esistono 2 tipi di vincoli:
  - Intrarelazionali:
    - Specificano quali attributi devono essere non NULL
    - Specificano quali attributi sono chiave
  - Interrelazionale
    - Specificano quali attributi sono chiavi esterne

Modello dei dati relazionale - Chiavi

- Un insieme X di attributi di uno schema di relazione R è detto superchiave dello schema se ogni istanza valida dello schema non contiene 2 o più n-uple distinte, t1 e t2, con t1[X] == t2[X]
  - Essenzialmente un insieme di attributi è superchiave se per ogni combinazione valida di questi attributi, posso risalire ad una sola n-upla della relazione
- Una chiave è una superchiave minimale, ovvero un insieme X di attributi (x1, x2, …, xn) tali che, tolto un qualsiasi attributo xi, X non sia più una superchiave.
- Per esempio, considerata la seguente relazione:

Nome	Matricola	Indirizzo	Telefono
Mario Rossi	123456	Via Etnea 18	777777
Maria Bianchi	234567	Via Roma 2	888888
Giovanni Verdi	345678	Via Etnea 18	999999
Enzo Gialli	456789	Via Catania 3	444444

- Sebbene la coppia { Nome, Matricola } sia una superchiave, non è una chiave, infatti mi basta anche soltanto { Matricola }, che è una chiave.

Modello dei dati relazionale - Esistenza chiavi, chiave primaria

- Data una relazione R:{A1, A2, …, An}, sicuramente {A1, A2, …, An} è una superchiave. Quindi esiste sempre almeno una chiave (che nel caso peggiore, è proprio {A1, A2, …, An}.
- Le chiavi hanno un ruolo fondamentale nelle basi di dati:
  - L'esistenza delle chiavi garantisce l'accesso univoco a ciascun dato della base di dati
  - Il modello relazionale è basato sui valori, e le chiavi permettono di correlare i dati tra loro
- In presenza di valori nulli, i valori della chiave non sempre permettono:
  - L'identificazione delle varie n-uple in maniera univoca
  - La semplice realizzazione di riferimenti che provengono da altre relazioni
- Quando progettiamo uno schema relazionale è bene scegliere una chiave primaria:
  - Non ammette valori NULL
  - Nella rappresentazione dello schema, essa viene sottolineata.

Nome	Matricola	Indirizzo	Telefono
Mario Rossi	123456	Via Etnea 18	777777
Maria Bianchi	234567	Via Roma 2	888888
Giovanni Verdi	345678	Via Etnea 18	999999
Enzo Gialli	456789	Via Catania 3	444444

Modello dei dati relazionale - Chiavi esterne

- Un insieme di attributi {A1, A2, …, An} appartenenti ad uno schema relazionale R può essere definito chiave esterna che riferisce una chiave primaria {B1, B2, …, Bn} di un altro schema S se:
  - Per ogni n-upla r di R esiste un n-upla s di S tale che r.Ai = s.Bi per i = 1, 2, …, n.
  - In questo caso, si dice che s è riferita da r.
- Notiamo una chiave esterna con una sottolineatura tratteggiata nello schema.
- Ad esempio:

Esami

Corso	Matricola	Voto
Programmazione 1	345678	27
Architettura	123456	30
Matematica discreta	234567	19
Basi di Dati	345678	28

Studenti

Nome	Matricola	Indirizzo	Telefono
Mario Rossi	123456	Via Etnea 18	777777
Maria Bianchi	234567	Via Roma 2	888888
Giovanni Verdi	345678	Via Etnea 18	999999
Enzo Gialli	456789	Via Catania 3	444444

- {Matricola} è una chiave esterna rispetto alla chiave primaria di Studenti, in quanto per ogni valore di Esami[Matricola], esiste anche in Studenti[Matricola]

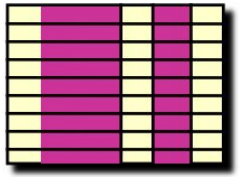
Algebra Relazionale

- Il modello dell'algebra relazionale è definito da:
  - Un set di operatori che:
    - Sono definiti sulle relazioni
    - Producono come risultato una relazione
    - Possono essere combinati per formare espressioni complesse
- Gli operatori primitivi dell'Algebra Relazionale sono:
  - Ridenominazione
  - Unione
  - Differenza
  - Proiezione
  - Restrizione o Selezione
  - Prodotto
- Per convenzione i simboli:
  - R, S, … denotano relazioni
  - A, B, … denotano attributi
  - X, Y, … denotano insiemi di attributi

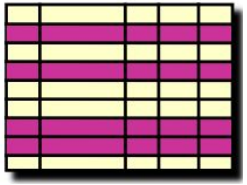
Algebra relazionale - Ridenominazione, Unione, Differenza, Intersezione

- L’operatore di ridenominazione viene indicato con la lettera greca δ.
  - Data una tabella, restituisce una tabella identica a quella data, in cui il nome di un attributo è stato modificato.
  - La sintassi è: $\delta_{\text{Attributo} \rightarrow \text{Nuovo nome attributo}}(\text{Relazione})$
- Gli operatori di Unione, Differenza e Intersezione sono fondamentalmente uguali a quelli che conosciamo nell’algebra degli insiemi:
  - Unione: $R \cup S = \{t | t \in R \vee t \in S\}$
  - Differenza: $R - S = \{t | t \in R \wedge t \notin S\}$
  - Intersezione: $R \cap S = \{t | t \in R \wedge t \in S\}$
- Se per esempio volessimo unire la tabella Madri(Mamma,Figlio) con Padri(Papà, Figlio), nonostante le tabelle siano formate in maniera simile, non possiamo unirle così come sono. Dobbiamo prima rinominare gli attributi in maniera che si chiamino allo stesso modo. Solo a quel punto possiamo unire le due relazioni.

Algebra Relazionale - Proiezione e Selezione/Restrizione



- L'operazione di proiezione viene indicata con π
- E' definito come $\pi_{A1, A2, \ldots, An}(R) = \{ t[A1, A2, \ldots, An] \mid t \in R \}$
  - Essenzialmente, data una Relazione R, siano A1, A2, …, An un sottoinsieme dei suoi attributi, definiamo proiezione di R negli attributi A1, A2, …, An quella relazione π A1 X A2 X … X An tale che per ogni tupla in π ne esiste una in R che abbia gli stessi A1, A2, …, An
  - La cardinalità di π è sempre minore o uguale a quella della relazione cui è applicata



- L'operazione di Selezione o Restrizione viene indicata con σ
- E' definita come $\sigma_{\lambda}(R) = \{t \mid t \in R \land \lambda(t) = TRUE\}$
  - Essenzialmente data una Relazione R, sia λ(t) una formula proposizionale costruita a partire dagli atomi del tipo (AθB) con connettivi $\land, \lor, \sim$.
    - A e B sono attributi di R, costanti o risultati di altri atomi, mentre θ = { =, <, >, ≠, ≤, ≥ }

Algebra relazionale - Prodotto (Cartesiano)

- L’operazione di prodotto cartesiano viene indicata con X
- E’ definita come R X S = { tu | t ∈ R ∧ u ∈ S }
  - Essenzialmente, definite due relazioni R(A1:T1, A2:T2, …, An:Tn) e S(B1:U1, B2:U2, …, Bm:Um), creiamo una relazione Z(A1:T1, A2:T2, …, An+m:Tn+m) che abbia come tuple ogni possibile tupla di R unita ad ogni possibile tupla di S
  - Una differenza sostanziale tra il prodotto cartesiano algebrico e quello relazionale è che quello relazionale non da luogo a tuple di elementi dei suoi operandi.

Employees

Employee	Project
Smith	A
Black	A
Black	B

Projects

Code	Name
A	Venus
B	Mars

Employes X Projects

Employee	Project	Code	Name
Smith	A	A	Venus
Black	A	A	Venus
Black	B	A	Venus
Smith	A	B	Mars
Black	A	B	Mars
Black	B	B	Mars

Algebra relazionale - Operatori Derivati

- Ha senso classificare alcuni operatori molto utili che si possono esprimere in funzione di quelli primitivi
- Intersezione:
  - Dati R, S relazioni dello stesso tipo (ovvero con le stesse etichette)
  - $R \cap S = \{t | t \in R \land t \in S\}$
  - Si può esprimere in funzione degli operatori primitivi come $R \cap S = R - (R - S)$
- Join:
  - Forse l'operatore più importante dell'algebra relazionale, combina tuple da relazioni diverse in base al valore degli attributi
  - Esistono due tipi principali di join:
    - Natural join
    - Theta join
    - Varianti dei precedenti

Algebra relazionale - Natural JOIN

- Sia R con attributi XY ed S con attributi YZ
- Si definisce natural join di R ed S, e si indica come R ⋈ S quella relazione di attributi XYZ costituita dalle n-uple di t tali che t[XY] ∈ R e t[YZ] ∈ S
  - Quindi: R ⋈ S = { t | [XY] ∈ R e t[YZ] ∈ S }
  - Ovvero le n-uple del risultato sono ottenute combinando le n-uple di R e S che hanno gli stessi valori negli attributi con lo stesso nome

Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

Maternity

Mother	Child
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

Paternity ⋈ Maternity

Father	Child	Mother
Adam	Cain	Eve
Abraham	Isaac	Sarah
Abraham	Ishmael	Hagar

- Ciò che succede nell’esempio è che in base all’attributo in comune (Child) vengono create le n-uple a partire da quelle che hanno lo stesso valore:
  - Ad esempio “Adam” e “Eve”, avendo nel campo Child “Cain”, anche se di tabelle diverse, vanno assieme

Algebra relazionale - Theta-JOIN ed Equi-JOIN

- E’ una versione modificata del natural join, in cui viene specificato un predicato per la selezione delle n-uple
- E’ quindi un operatore derivato: Lo indichiamo con $\bowtie_F$, dove F è un predicato
  - $R \bowtie_F S = \sigma_F(R \times S)$
- Se F è una congiunzione di uguaglianze, viene chiamata equi-Join
  - Siano R(A1:T1, A2:T2, …, An:Tn) e S(An+1:Tn+1, …, An+m,Tn+m), tali che abbiano almeno un etichetta in comune
  - Poniamo allora $R \bowtie_{Ai=Ak} S = \{ tu \mid t \in R, u \in S, t.A_i == u.A_k \}$
  - Dove i va da 1 a n e k va da n+1 a n+m
  - Possiamo scrivere l’equi-JOIN come operazione derivata:
    - $R \bowtie_{Ai=Ak} S = \sigma_{Ai = Ak}(R \times S)$
- Possiamo quindi verificare che la natural join è un operatore derivato, infatti la natural join di R ed S.
  - Se rinominiamo gli attributi di S come Y’ (in modo che siano con nomi diversi da quelli di R), poi facciamo la EquiJoin del tipo $R \bowtie_{Y = Y'} S'$ e poi rimuoviamo Y’, ridondante, con una proiezione

Algebra Relazionale - Query 1

- L’algebra relazionale può quindi essere usata per interrogare una base di dati
- Definiamo query una funzione espressa attraverso gli operatori dell’algebra relazionale, che va da un’istanza di un database (ovvero un insieme di relazioni) ad una relazione.
- Sia data:
  - Trovare numero, nome ed età degli impiegati che guadagnano almeno 40mila euro all’anno.
  - Prima SELEZIONIAMO gli impiegati con almeno 40mila euro all’anno, poi prendiamo solo i campi che vogliamo.
    - Nota che non ci serve per forza usare tutte le tabelle

Employees

Number	Name	Age	Salary
101	Mary Smith	34	40
103	Mary Bianchi	23	35
104	Luigi Neri	38	61
105	Nico Bini	44	38
210	Marco Celli	49	60
231	Siro Bisi	50	60
252	Nico Bini	44	70
301	Steve Smith	34	70
375	Mary Smith	50	65

Supervision

Head	Employee
210	101
210	103
210	104
231	105
301	210
301	231
375	252

Algebra relazionale - Query 2

- Sia data:
  - Trovare il numero dei responsabili degli impiegati che guadagnano più di 40mila euro.
  - SELEZIONIAMO gli impiegati che hanno almeno 40mila euro, facciamo una equijoin con tra il risultato della SELEZIONE e la tabella Supervision, confrontando DOVE Number == Employee. Infine PROIETTIAMO solo Head.

Employees

Number	Name	Age	Salary
101	Mary Smith	34	40
103	Mary Bianchi	23	35
104	Luigi Neri	38	61
105	Nico Bini	44	38
210	Marco Celli	49	60
231	Siro Bisi	50	60
252	Nico Bini	44	70
301	Steve Smith	34	70
375	Mary Smith	50	65

Supervision

Head	Employee
210	101
210	103
210	104
231	105
301	210
301	231
375	252

- Trovare nome e salario dei responsabili degli impiegati che guadagnano più di 40mila euro.
  - Partendo dal risultato di prima, facciamo una equijoin DOVE Head == Number. Infine SELEZIONIAMO nome e salario

Algebra relazionale - Query 3

- Trovare gli impiegati che guadagnano più dei loro responsabili e visualizzare numero, nome e salario sia dell'impiegato che del responsabile:

Employees

Number	Name	Age	Salary
101	Mary Smith	34	40
103	Mary Bianchi	23	35
104	Luigi Neri	38	61
105	Nico Bini	44	38
210	Marco Celli	49	60
231	Siro Bisi	50	60
252	Nico Bini	44	70
301	Steve Smith	34	70
375	Mary Smith	50	65

Supervision

Head	Employee
210	101
210	103
210	104
231	105
301	210
301	231
375	252

- Effettuiamo una Theta JOIN tra Employees e Supervision, DOVE Number == Employee, e proiettiamo tutto tranne Number ed Age. RINOMINIAMO NAME -> ENAME, SALARY->ESALARY. Effettuiamo adesso una EQUIJOIN tra il risultato ed Employees, DOVE Head == Number. PROIETTIAMO tutto tranne Number ed age, e RINOMINIAMO NAME -> HNAME, SALARY ->HSALARY. Infine, selezioniamo DOVE ESALARY > HSALARY.

Algebra relazionale - JOIN incompleti

- Nel caso in cui alcuni valori tra gli attributi comuni non coincidono, alcune n-uple non partecipano al JOIN
  - Esse vengono definite “dangling” n-uple (“pendenti”)
  - Un caso estremo potrebbe essere che nessuna n-upla trovi un corrispettivo cui JOINare
  - L’esatto opposto è quando ogni n-upla della prima relazione si combina con quella della second (effettivamente diventando una sorta di prodotto cartesiano, cui viene proiettato tutto tranne che una delle copie degli attributi della natural join)
- Definiamo OUTER JOIN una variante del join che mantiene nel risultato le n-uple che non partecipano al join, riempendo i campi non matchati con NULL. Ne esistono 3 tipi diversi:
  - Left: solo dangling n-uple del primo operando
  - Right: solo dangling n-uple del secondo operando
  - Full: tutte le dangling n-uple
- Si definisce giunzione esterna con il simbolo di natural join, con una freccia con la punta rivolta verso il senso della giunzione (doppia punta nel caso della Full)

Algebra Relazionale - Proprietà del JOIN

- Il JOIN e':
  - Commutativo: $R \bowtie S = S \bowtie R$, qualsiasi sia R, S
  - Associativo: $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$, qualsiasi sia R, S, T
- Ne consegue che possiamo scrivere sequenze di JOIN del tipo $R \bowtie S \bowtie \ldots \bowtie T$, senza alcun rischio di ambiguità, in quanto cambiando l'ordine degli operandi il risultato sarebbe comunque lo stesso.
- Il JOIN è anche definito quando non ci sono attributi comuni fra le relazioni, nel qual caso, non essendoci vincoli sulle tuple da selezionare, otteniamo un prodotto cartesiano.
- Inoltre, Date R:{T} ed S:{T}, il JOIN tra R ed S è uguale all'intersezione delle stesse. Infatti, essendo gli attributi uguali, vengono controllati tutti, essenzialmente confrontando l'intera n-upla.

Algebra Relazionale - Semi Join, Unione Esterna

- Siano R con attributi XY, S con attributi YZ.
- $R \ltimes S$ è definito come semigiunzione tra R ed S, ed è costituito da tutte le n-uple di R che partecipano ad $R \bowtie S$.
  - In pratica: $R \ltimes S = \pi_{XY}(R \bowtie S)$, quindi è un operazione derivata
- Definiamo invece come unione esterna (indicata col simbolo di unione con sopra una doppia freccia) e si ottiene essenzialmente aggiungendo gli attributi mancanti ad ogni tabella (nel nostro caso, Z ad R, X ad S), riempendoli con null, e fare quindi una unione.
- Definiamo Giunzione esterna, con il simbolo di Join ed una freccia, eventualmente doppia sopra, la giunzione naturale che includa tutti gli elementi della tabella di destra, sinistra o entrambe (in base al senso della freccia), riempendo i campi mancanti con NULL

Algebra Relazionale - Problemi con valori nulli

- Quando si effettuano operazioni che contengono un predicato, come ad esempio una SELEZIONE o una Thetha JOIN, possiamo incorrere nel caso dove vengono comparati valori nulli.
    - Questo può causare un comportamento indesiderato della query: se ad esempio selezionassimo tutte le n-uple da una tabella di Persone, tale che l'età della persona sia minore di 30, e gli aggiungessimo tutte le persone tale che la loro età sia maggiore o uguale a 30, le persone con un età settata a NULL, non sarebbero incluse nel risultato, nonostante intuitivamente la nostra query dovrebbe selezionare l'intera tabella.
- Per ovviare a questo motivo, esistono degli operatori di confronto "IS NULL", e la sua forma negata, "IS NOT NULL"
- Sarebbe anche possibile usare una logica a tre valori, True, False, Unknown, ma nella pratica risulta inutile e scomodo nell'implementazione.

Algebra relazionale: Divisione

- Siano R(XY), S(Y) due schemi di relazione, e siano r, s, due istanze di R, S.
- Definiamo divisione tra r ed s, indicata come r : s, la relazione le cui tuple, se estese con una qualunque tupla del secondo operando, producono una tupla del primo operando.
  - Più propriamente, $r:s = \{ t \mid \{t\} \times s \subseteq r \}$
  - Essenzialmente, proietto il primo operando su X, ovvero sull'insieme di quelle etichette non presenti in S. Costruisco poi l'insieme universo U, formato come il risultato della proiezione X S. Elimino poi da U tutte quelle tuple del tipo (U / R), proiettate su X. U è il risultato della divisione.
- La divisione è molto utile, in quanto permette di rispondere a query del tipo: trova tutte le n-uple di R, associate a tutte le n-uple di S
  - Potrebbe ad esempio servire a considerare tutti gli studenti che hanno superato un certo set di esami, necessario per iscriversi ad un certo corso

Algebra relazionale - Viste / relazioni derivate

- Una vista è una rappresentazione diversa per gli stessi dati
- Essenzialmente, possiamo considerarlo come una sorta di copia
- Le relazioni derivate sono relazioni il cui contenuto è funzione del contenuto di altre relazioni, anche derivate, definito per mezzo di query precedenti.
  - Ad esempio, il risultato di una selezione su una lista di impiegati
- Le relazioni di base invece sono quelle relazioni il cui contenuto è autonomo, in un certo senso “primitivo”.
  - Ad esempio, la lista di tutti i dipendenti di una compagnia
- L’utilità delle viste è che possiamo usarle come una sorta di variabili per costruire query sequenziali più facilmente scrivibili e mantenibili, evitando espressioni ripetute
- Inoltre le viste permettono di fare in modo che l’utente abbia accesso ad uno schema esterno ad hoc, anche ristretto in base alle eventuali autorizzazioni
- Inoltre, l’uso di viste non influisce sull’efficienza delle interrogazioni

Algebra Relazionale - Viste Materializzate, Viste Virtuali

- Le viste possono anche essere materializzate o virtuali:
  - Viste materializzate: Sono quelle relazioni che, una volta computate, vengono memorizzate nella base di dati.
  - Presentano vantaggi e svantaggi:
    - Sono immediatamente disponibili per le interrogazioni senza ulteriori computazioni
    - Sono spesso ridondanti
    - Appesantiscono di molto il processo di aggiornamento
    - Sono poco supportate dai DBMS
  - Viste virtuali: (o semplicemente viste): Sono relazioni virtuali, definito solo dalla definizione, che viene memorizzata dalla base di dati. Ogni interrogazione che accede ad una vista virtuale, ricalcola l'intera vista sostituendo la definizione.
  - Presentano vantaggi e svantaggi:
    - Sono facilmente salvabili, occupando poco spazio
    - Sono supportate da praticamente tutti i DBMS
    - Se fatto frequente accesso, il continuo ricalcolo può diventare gravoso per il sistema

Algebra relazionale - Espressioni equivalenti

- In algebra relazionale, due espressioni si dicono espressioni equivalenti se producono lo stesso risultato, a prescindere dall'istanza della base di dati sulla quale sono eseguiti.
  - Due espressioni equivalenti possono però avere costi di calcolo anche significativamente differenti. Ha senso quindi cercare quali sono quei modi per ottenere una query "ottimizzata" a partire da una data.
- Ad esempio:
  - $\sigma_{A=10}\,(R1 \bowtie R2) = R1 \bowtie \sigma_{A=10}\,(R2)$
  - Nella prima espressione, la natural join può produrre una tabella anche molto grossa, con la seconda espressione invece, riduciamo sin da subito la grandezza della join.
- Esistono regole specifiche per l'ottimizzazione di certe espressioni.

Algebra relazionale - Espressioni Equivalenti

- $\sigma_{c(X)}(\sigma_{c\,(Y)}(E)) = \sigma_{c(X) \wedge c(Y)}(E)$
  - Ovvero: Doppia selezione è meno efficiente di una singola selezione che controlla entrambe le condizioni
- $\sigma_{c(X)}(\pi_Y(E)) = \pi_Y(\sigma_{c(X)}(E))$ se $X \subseteq Y$
  - Ovvero: La selezione di una proiezione è equivalente alla proiezione della selezione, se $X \subseteq Y$
- $\pi_Y(\sigma_{c(X)}(\pi_{XY}(E))) = \pi_Y(\sigma_{c(X)}(E))$ se $X \not\subseteq Y$
  - Ovvero: Proiettare la selezione di una proiezione maggiore di quella originale, è come lasciare soltanto la proiezione minore
- $\sigma_{c(Y)}(E \times F) = \sigma_{c(Y)}(E) \times F$ (se Y è un attributo di E, altrimenti basta invertire E ed F)
  - Ovvero: Possiamo anticipare la selezione di un attributo prima rispetto al prodotto, ed è conveniente in quanto riduce da subito la cardinalità del prodotto.
- $\sigma_{c(Y)\wedge c(Z)}(E \times F) = \sigma_{c(Y)}(E) \times \sigma_{c(Z)}(F)$
  - Ovvero: Conviene sempre anticipare la selezione di un attributo, anche scomponendola nel caso sia un espressione complicata.
- $\sigma_{c(W)\wedge c(Y)\wedge c(Z)}(E \times F) = \sigma_{c(W)}(\sigma_{c(Y)}(E) \times \sigma_{c(Z)}(F))$
  - Ovvero: Possiamo scomporre il predicato di una selezione anche non distribuendolo del tutto.

Algebra Relazionale - Espressioni equivalenti

- $\pi_X(\pi_Y(E)) = \pi_X(E)$ se $X \subseteq Y$.
  - Ovvero, una proiezione seguita da una più stringente, è come fare soltanto quella più selettiva
- $\pi_X(E) = E$ se $X = attr(E)$
  - Ovvero, una proiezione su tutti gli attributi di una tabella la lascia invariata
- $\pi_{XY}(E \times F) = \pi_X(E) \times \pi_Y(F)$ se $X \subseteq attr(E)$ e $Y \subseteq attr(F)$
  - Ovvero: Visto che il prodotto di proiezioni è equivalente alla proiezione del prodotto, anticipiamo le proiezioni rispetto al prodotto, in quanto otterremo molte colonne nel risultato, aumentando l'efficienza della query.

Algebra Relazionale - Algoritmo di Ottimizzazione

- Per prima cosa si anticipa σ rispetto a π usando la formula (vista prima):
  - $\sigma_{c(X)}(\pi_Y(E)) = \pi_Y(\sigma_{c(X)}(E))$ se $X \subseteq Y$
- Si raggruppano le restrizioni, usando la formula:
  - $\sigma_{c(X)}(\sigma_{c(Y)}(E)) = \sigma_{c(X) \wedge c(Y)}(E)$
- Si anticipano le restrizioni prima dei prodotti:
  - $\sigma_{c(Y)}(E \times F) = \sigma_{c(Y)}(E) \times F$
  - $\sigma_{c(Y)\wedge c(Z)}(E \times F) = \sigma_{c(Y)}(E) \times \sigma_{c(Z)}(F)$
  - $\sigma_{c(W)\wedge c(Y)\wedge c(Z)}(E \times F) = \sigma_{c(W)}(\sigma_{c(Y)}(E) \times \sigma_{c(Z)}(F))$
- Una volta che non possiamo più effettuare nessuno degli step precedenti, usiamo i seguenti metodi:
  - Eliminazione di proiezioni superflue => $\pi_X(E) = E$ se $X = attr(E)$
  - Raggruppamento delle proiezioni => $\pi_X(\pi_Y(E)) = \pi_X(E)$ se $X \subseteq Y$
  - Anticipazione di π rispetto al prodotto => $\pi_{XY}(E \times F) = \pi_X(E) \times \pi_Y(F)$ se $X \subseteq attr(E)$ e $Y \subseteq attr(F)$
  - Anticipazione #2 di π rispetto al prodotto => $\pi_Y(\sigma_{c(X)}(\pi_{XY}(E))) = \pi_Y(\sigma_{c(X)}(E))$ se $X \not\subseteq Y$

SQL - Introduzione

- Sql viene sviluppato da IBM nel 1973, ed è lo standard moderno per tutti i sistemi commerciali ed open source (Oracle, Informix, MySql, etc…)
- Esistono anche sistemi commerciali con interfacce di tipo QBE:ACCESS. Queste però hanno anche sistemi per la traduzione in SQL
- I comandi SQL non sono case sensitive (anche se per convenzione di solito vengono scritti in maiuscolo)
- I comandi SQL possono essere spezzati su più linee, ma non nel mezzo di una parola chiave
    - Di solito, si va a capo quando c'è una nuova clausola

SQL - SELECT

- L’operatore di SELECT ha la seguente sintassi
- SELECT [DISTINCT] *Attributi*
  FROM *Tabelle*
  [WHERE *Condizione*]
  - Ciò che sta in parentesi quadrate è opzionale. I nomi in corsivo sono quelli che vengono sostituiti con i nomi di eventuali attributi, costanti o viste.
- Come funziona?
  - Essenzialmente prima di tutto viene calcolato il prodotto cartesiano tra le tabelle in *Tabelle*
  - Poi vengono selezionate solo le righe che soddisfano una specifica condizione
  - Infine vengono valutate le espressioni specificate nella target list *Attributi*
- L’operazione di SELECT implementa assieme gli operatori dell’algebra relazionale:
  - Ridenominazione
  - Proiezione
  - Selezione
  - Join

SQL - SELECT: Un esempio

Nome	Cognome	Matricola
Alberto	Angela	2
Marco	Senatori	3
Maria	Bianchi	4
Marco	Marchetti	5
Bianca	Marchetti	6
Mario	Rossi	1

- Dato il DB formato dall'unica tabella:
- Usiamo la seguente query:
  - SELECT DISTINCT Nome, Cognome FROM studenti WHERE Matricola <= 3;
- Il risultato che otteniamo è il seguente:

Nome	Cognome
Alberto	Angela
Marco	Senatori
Mario	Rossi

- Qui il distinct non fa alcuna differenza, ma nella query:
  - SELECT DISTINCT Cognome FROM studenti WHERE Matricola >= 3;
- La differenza è che il risultato non contiene il "Marchetti" duplicato

SQL - SELECT: Altri quirk

- Se durante un'operazione di select, tra gli attributi viene specificata un espressione aritmetica e/o funzionale che coinvolge uno o più attributi precedenti, la tabella ritornata vedrà una colonna che ha come nome l'operazione passata, e come valori il risultato della stessa.
- Ad esempio, usando l'operatore CONCAT(*Str1*, *Str2*, …):
  - SELECT Nome, Cognome, concat(Nome, " ", Cognome) FROM studenti;

Nome	Cognome	concat(Nome, " ", Cognome)
Alberto	Angela	Alberto Angela
Marco	Senatori	Marco Senatori
Maria	Bianchi	Maria Bianchi
Marco	Marchetti	Marco Marchetti
Bianca	Marchetti	Bianca Marchetti
Mario	Rossi	Mario Rossi

- Qualora volessimo rinominare gli attributi possiamo usare l'operatore "AS" (che può anche essere sottointeso. E' necessario usare le virgolette in caso di nome attributo contenente spazi.
  - SELECT Nome, Cognome, CONCAT(Nome, " ", Cognome) AS "Nome Completo" FROM studenti;

Nome	Cognome	Nome Completo
Alberto	Angela	Alberto Angela
Marco	Senatori	Marco Senatori
Maria	Bianchi	Maria Bianchi
Marco	Marchetti	Marco Marchetti
Bianca	Marchetti	Bianca Marchetti
Mario	Rossi	Mario Rossi

SQL - SELECT : Problemi con i nomi e alias

- Quando esistono attributi con lo stesso nome in tabelle diverse, essi devono essere espressi in base a quale tabella vogliamo referenziare:
  - Ad esempio, se una tabella professore ha il campo “Nome”, ma anche la tabella studente lo ha, in una query che usa il nome di questi attributi, è necessario specificare di quale stiamo parlando.
  - Se volessimo risolvere questo problema definitivamente, dobbiamo trovare un altro nome per gli attributi duplicati.

SQL - Predicati : BETWEEN, IN, LIKE, IS NULL

- Between:
  - *Espr1* [NOT] BETWEEN *Espr2* AND *Espr3*
  - Equivale a: [NOT] (*Espr2* <= *Espr1* AND *Espr1* <= *Espr3*)
- In:
  - *Espr1* [NOT] IN (*Value1*, *Value2*, …)
  - Equivale a controllare se [NOT] *Espr1* è uguale ad almeno uno dei *Value1*, *Value2*, …
- Like:
  - *Espr1* [NOT] LIKE “*stringvalue*”, dove *stringvalue* contiene wildcards: % e _.
  - % Indica che può essere sostituito con qualsiasi sequenza di caratteri, anche vuota.
  - _ Indica che può essere sostituito con un solo carattere, necessariamente non nullo.
  - Si usa una escape sequence “\%” ed “_” per cercare i simboli così come sono.
- Is Null;
  - *Espr* IS [NOT] NULL:
  - Equivale a controllare se *Espr* sia NULL o meno.

SQL - Predicati : AND, OR, NOT e precedenza

- And:
  - *Espr1* AND *Espr2*
- Or:
  - *Espr1* OR *Espr2*
- Not:
  - NOT *Espr*
- In SQL, a meno dell'ordine imposto dalle parentesi, gli operatori seguono un ordine ben specifico:
  - Per prima cosa, vengono eseguiti tutti gli operatori di confronto
  - NOT
  - AND
  - OR

SQL - ORDER BY, JOIN, Prodotto Cartesiano

- Order by:
  - ORDER BY *Espr1, Espr2, …* [ASC / DESC]:
  - Ordina la lista in base all'attributo o espressione passata. Se non scpecificato, l'ordine è ascendente, altrimenti ASC => ascendente, DESC => discendente
  - Una volta creata un alias per un attributo, è possibile usarla nell'*Espr* dell'ordinamento.
  - L'ordinamento può essere fatto su più espressioni, che vengono valutate in caso di uguaglianza di quelle più significative.
  - L'ordinamento può coinvolgere anche colonne non incluse nella target list.
- Join:
  - Esprimiamo la join come una SELECT
  - Select (Tabelle_da_Joinare)
    From (Tabelle_da_Joinare)
    Where (condizioni…)
  - La condizione del join va, ovviamente, nella WHERE clause. Ricorre l'uso dei prefissi di tabella in caso di ambiguità dei nomi degli attributi.
- Prodotto Cartesiano:
  - Quando non viene specificata una condizione di join, o tutte le righe di tutte le tabelle soddisfano la condizione specificata, la join diventa equivalente ad un prodotto cartesiano.

SQL - Vari tipi di JOIN

- Join esplicito:
  - Select *Attr1*, *Attr2*, …, FROM *Tabella* JOIN *Tabella2* ON *Condition*, … [WHERE …]
  - Essenzialmente, viene usato per specificare tipi particolari di JOIN. Di per se, è equivalente all'usare il where
- Left, Right, Full Join:
  - Esattamente equivalenti ai join esterni destri, sinistri o interi dell'algebra relazionale
- Cross Join:
  - Il prodotto cartesiano
- Union Join … On …:
  - E' l'unione esterna piena dell'algebra relazionale.
- Natural Join:
  - La natural join
- Join … Using …:
  - La natural join sugli attributi specificati (equijoin -> proiezione)
- [LEFT | RIGHT | FULL]:
  - Usati per specificare quale tipo di comportamento usare nelle join con p

Progettazione

- La progettazione della base di dati è quel passo fondamentale per lo sviluppo di una base di dati, nel quale si separano le decisioni relative a cosa rappresentare in una base di dati da quella relative a come implementarle
- La progettazione si divide in 3 fasi:
  - Progettazione concettuale
  - Progettazione Logica
  - Progettazione Fisica

Progettazione concettuale

- Lo scopo della progettazione concettuale è tradurre una descrizione informale della realtà in uno schema formale, indipendente dal criterio di rappresentazione del DBMS specifico usato.
- Di solito, si fa un analisi dei requisiti del DB prima di fare la progettazione concettuale.
- La descrizione formale fa riferimento ad un modello concettuale, cioè un insieme di concetti e notazioni standard adatti alla rappresentazione del dominio applicativo.

Progettazione Logica

- Consiste nel tradurre lo schema concettuale in un modello logico di dati, anche in base al DBMS che si vuole usare. Il risultato è lo schema logico.
- Un passo fondamentale della progettazione logica è l’ottimizzazione della rappresentazione in base alle operazioni eseguite (ad esempio il processo di normalizzazione).

Progettazione fisica

- La progettazione fisica consiste nell’individuare i parametri fisici di memorizzazione dei dati. Si produce lo schema fisico che fa riferimento ad un certo modello fisico dei dati che dipende dal DBMS scelto.

Progettazione concettuale - Analisi dei requisiti

- L’analisi dei requisiti ha come scopo:
  - individuare proprietà e funzionalità del sistema
  - produrre una descrizione dei dati coinvolti e delle operazioni su di essi
  - individuare i requisiti software ed hardware del sistema
- Questa produrrà come output:
  - Studio di fattibilità:
    - Stima dei costi e impegno personale
    - Inefficienze dovute al cambio di sistema
    - Benefici in termini di riduzione dei tempi di lavoro ed efficienza dei processi aziendali
  - Piano di sviluppo:
    - Priorità di realizzazione
    - Tempi di realizzazione
- Per effettuare una buona analisi serve:
  - Analizzare il sistema informativo esistente
  - Intervistare i responsabili del settore
  - Produrre una bozza dei requisiti, assieme ad un glossario
  - Analizzare ed eliminare eventuali ambiguità di descrizione

Progettazione concettuale - Formazione del glossario

- Per prima cosa, si raggruppano frasi e termini relativi alle categorie di dati e alle operazioni inerenti lo sviluppo del DB, nonché il settore in cui viene applicato.
  - Per ogni termine, vengono individuati significato, sinonimi e termini collegati
- Si verifica la completezza, ovvero di non aver trascurato nessun aspetto importante
- Si verifica la consistenza, ovvero:
  - Tutti i termini del glossario sono stati definiti
  - Tutti i termini del glossario sono stati usati in almeno un operazione (per evitare spreco di spazio)
  - Tutte le operazioni definite danno riferimento a termini definiti
-

Progettazione concettuale: Il modello Entità-Relazione (E-R)

- Il modello Entità-Relazione è un modello concettuale di dati che contiene alcuni costrutti atti a descrivere la realtà in maniera semplice, indipendentemente dall'organizzazione dei dati nel computer.
- I suoi costrutti sono:
  - Entità (Class)
  - Associazioni (Relazioni Class)
  - Proprietà (attributi delle Classi)
  - Cardinalità (len)
  - Identificatori
  - Generalizzazioni
  - Sottoinsiemi
- Le entità, associazioni e proprietà sono riguardanti una conoscenza "astratta". La conoscenza concreta di un valore è incete attribuita ad una "Istanza" dell'entità o della associazione

Progettazione concettuale, modello ER - Entità, Associazioni

- Un’entità è una classe di oggetti che hanno proprietà comuni ai fini dell’applicazione.
- Un‘istanza di entità è una specifica istanza, che assume valori concreti
- Un’associazione rappresenta i legami logici tra due o più entità dell’applicazione.
  - Possono anche esserci più relazioni fra le stesse entità (associazioni ricorsive).
- Un’istanza di associazione è una specifica relazione tra specifiche istanze di entità
- Ogni istanza (di entità o associazione) ha le proprie proprietà (attributi), i quali descrivono le caratteristiche di queste. Le proprietà assumono dei valori ben specifici.
-

Rappresentazione del modello E-R - schema scheletro

- Il modello E-R usa simboli grafici per favorire la comprensione dello schema.
- Possono comunque essere aggiunte frasi di commento per rappresentare le caratteristiche non modellabili (ad esempio vincoli complessi).
- Per le entità, si usa un rettangolo col nome dell'entità
- Per le associazioni si usa un rombo col nome dell'associazione
- Per gli attributi, si usa un pallino collegato all'entità o associazione
- Usando questi tre elementi grafici è possibile rappresentare lo schema scheletro del DB, ovvero una prima rappresentazione di massima di ciò che sarà nel DB.
- Lo schema scheletro potrebbe anche contenere associazioni potenzialmente ridondanti, quando queste hanno le stesse entità collegate.
- Per risolvere ambiguità di modellazione, si possono creare associazioni che legano più di due entità diverse

Rappresentazione del modello E-R - proprietà

- Per rappresentare una proprietà, si adopera un pallino collegato all'entità cui è parte.
- Si può indicare anche il numero minimo e massimo di valori per relazione (ottenendo qualcosa del tipo (n,m) , con n il minimo ed m il massimo)
- Quindi per una proprietà di cui deve esistere ed essere unico il valore (per esempio un campo obbligatorio in un form) si usa (1,1), che di norma viene sottointeso nel momento in cui non venga specificato nulla.
- Per dire che la proprietà possa anche essere null, n deve essere 0.
- Per dire che la proprietà può avere un numero qualsiasi di valori, bisogna lasciare la m come massimo (0,m) -> Opzionale, quanti ne vuoi
- Possono essere specificate anche proprietà derivate: queste sono definite come proprietà non strettamente intrinseche ma calcolabili con un metodo, che viene specificato a parte.

Rappresentazione del modello E-R: proprietà e chiavi

- Quando una proprietà è chiave di un entità o associazione, si annerisce il pallino che la rappresenta.
  - Sicuramente è del tipo (1,1)
  - Può essere composta
  - Di solito non è modificata durante il ciclo di vita del DB
- Se la chiave è composta, si uniscono con un pallino annerito ed una stanga le proprietà che compongono la chiave composta.
- Se sono presenti più pallini anneriti, essi rappresentano chiavi alternative.
- La scelta della chiave può essere effettuata in funzione di diversi fattori di tipo pratico (es. velocità di digitazione), comodità d'uso, velocità di controllo di validità etc…
- Spesso, si usano metodi di auto incremento per assegnare nuove istanze di entità, che garantiscono l'unicità della stessa.
- In un associazione, la chiave non deve essere esplicitata in quanto corrisponde sempre all'insieme delle chiavi delle entità partecipanti.

Rappresentazione del modello E-R - Associazioni N:M, Auto associazioni

- Anche nelle associazioni, si possono usare le indicazioni di cardinalità (n,m):
  - (1,1) : obbligatoria, unica
  - (1,n) : obbligatoria, almeno una volta, massimo n volte
  - (0,1) : opzionale, unica
  - (0,n) : opzionale, n volte
  - (1,m) con m non assegnato: almeno 1
  - (0,m) con m non assegnato: quanti se ne vuole, anche zero
- Un’associazione può anche avere come partecipanti istanze provenienti dalla stessa entità. In questo caso si parla di associazioni unarie o ad anello.
  - Ad esempio dato un database di persone, l’associazione “è sposato con” è unaria, in quanto riguarda due istanze della stessa classe (Persona).

Rappresentazione del modello E-R - Identificazioni esterne

- In alcuni casi una entità può essere identificata da altre ad essa collegate.
- Viene indicata con un simbolo di chiave che parte dal collegamento stesso fino all’entità
- Le identificazioni esterne sono tutte di tipo (1,1). ovvero associazioni binarie
- Possono comunque essere coinvolte entità che vengono coinvolte esternamente, a patto però che non si creino cicli di identificazione.
-

Rappresentazione modello E - R: Gerarchie

- Spesso nell'analisi di un settore aziendale possono essere identificate entità molto simili, con minime differenze tra di loro.
- Si potrebbe quindi pensare di creare una gerarchia concettuale. Dove si definiscono i legami logici tra un entità padre E e le sue entità figlie $E_1E_2…E_n$, dove queste ultime sono dette specializzazioni di E, sono tutte istanze di E, e non è detto che tutte le istanze di E facciano parte di $E_1E_2…E_n$.
- I tipi di gerarchie sono:
  - t : totale - per ogni istanza del padre, deve essere almeno parte di una delle figlie
  - nt : non totale - per ogni istanza del padre, può essere almeno parte di una delle figlie
  - e : esclusiva - per ogni istanza del padre, che è anche istanza di un figlio En, allora non è istanza di altri figli
  - ne : non esclusiva - per ogni istanza del padre, che è anche istanza di un figlio En. allora può essere istanza di altri figli
- Le proprietà vengono ereditate dal padre al figlio
- Le gerarchie concettuali vengono chiamate ISA (X is a Y)

Documentazione di uno schema E-R

- Lo schema E-R deve essere corredato con una documentazione di supporto per facilitare l'interpretazione dello stesso, nonchè corredare eventuali informazioni non direttamente esprimibili nello schema.
- Vengono quindi inserite le Business Rules:
  - Descrizione di un concetto
    - Viene espresso in linguaggio naturale
  - Vincoli di integrità
    - Viene espresso in un linguaggio formale scelto
  - Derivazione (un concetto che può essere ottenuto tramite calcoli su altri concetti).
    - Viene espresso in un linguaggio formale scelto.
- La documentazione finale è quindi composta da 4 tabelle:
  - Il dizionario dei dati, diviso in:
    - Tabella entità
    - Tabella relazioni
  - Regole di Vincolo
  - Regole di derivazione

Strategie di progettazione - Top Down

- La prima strategia di progettazione è chiamata Top-Down.
- Essa parte dalla specifica, che dopo essere stata trasformata in uno schema iniziale viene successivamente raffinata fino ad ottenere uno schema finale.
- Le operazioni sono di “complicazione”:
  - Dato un singolo concetto dello schema, viene trasformato in una struttura maggiormente complessa ma con migliore capacità e precisione espressiva
- Le regole sono:
  - T1: Un’entità che descrive due concetti legati tra di loro, diventa tue entità collegate da un’associazione
  - T2: Un’entità cui esistono entità figlie diventa composta da sottoentità distinte
  - T3: Una relazione che descrive più concetti tra stesse entità viene divisa in due relazioni
  - T4: Una relazione che descrive un concetto autonomamente definito, viene sostituito dalle entità partecipanti e la relazione stessa
  - T5: L’aggiunta di attributi ad entità
  - T6: L’aggiunta di attributi a relazioni

Strategie di progettazione - Top Down

- La prima strategia di progettazione è chiamata Top-Down.
- Essa parte dalla specifica, che dopo essere stata trasformata in uno schema iniziale viene successivamente raffinata fino ad ottenere uno schema finale.
- Le operazioni sono di “complicazione”:
  - Dato un singolo concetto dello schema, viene trasformato in una struttura maggiormente complessa ma con migliore capacità e precisione espressiva
- Le regole sono:
  - T1: Un’entità che descrive due concetti legati tra di loro, diventa tue entità collegate da un’associazione
  - T2: Un’entità cui esistono entità figlie diventa composta da sottoentità distinte
  - T3: Una relazione che descrive più concetti tra stesse entità viene divisa in due relazioni
  - T4: Una relazione che descrive un concetto autonomamente definito, viene sostituito dalle entità partecipanti e la relazione stessa
  - T5: L’aggiunta di attributi ad entità
  - T6: L’aggiunta di attributi a relazioni
- Questa strategia è comoda per piccoli progetti, ma poco prona al lavoro in gruppo per progetti più grandi.

Strategie di progettazione - Bottom Up

- La strategia di progettazione Bottom Up consiste nell'individuare varie componenti separate a partire dallo schema, svilupparle in maniera autonoma, unirle ed ottenere lo schema finale.
- Le operazioni sono:
  - T1: L'introduzione di un'entità a partire da un concetto
  - T2: L'individuazione di un'associazione tra due o più entità definite
  - T3: L'individuazione di una generalizzazione tra entità: in questo caso si forma una gerarchia
  - T4: L'individuazione di un entità a partire dagli attributi che li aggrega
  - T5: L'individuazione di un'associazione a partire dagli attributi che li aggrega
- Il principale vantaggio è che si adatta ad un lavoro modulare, dove diversi progettisti possono occuparsi di parti separate del DB che possono poi essere assemblate successivamente.
- Comporta comunque una notevole difficoltà nell'implementazione di sistemi concettualmente diversi

Ulteriori strategie

- Inside-out:
  - Una variante della bottom up, in cui si sviluppano schemi di supporto che vengono usati per poi estendere le definizioni a macchia d’olio.
- Mista:
  - Si cercano di combinare i vantaggi top-down e bottom up. Viene prima definito uno schema scheletro che contiene i principali concetti dell’applicazione, di modo da garantire una visione completa dell’intero progetto, in modo da guidare i progettisti durante lo sviluppo dei sottoschemi.

Progettazione : Metodologia generale

- Si parte da un’analisi dei requisiti che possa:
  - Ottenere requisiti minimi, raggruppandoli in insiemi omogenei
  - Analizzare ed eliminare qualsiasi ambiguità (interpretazioni, omonimie, etc…)
  - Costruire un glossario dei termini
- Poi viene costruito uno schema scheletro, questo deve:
  - Individuare i concetti più rilevanti, in particolare le entità, associazioni ed eventualmente le gerarchie.
- Poi decomposto se necessario lo schema scheletro in sottoschemi, in base alle specifiche.
- Viene quindi continuamente raffinato ogni sottoschema aggiungendo raggruppando e gerarchizzando concetti, per meglio esprimere i requisiti
- Infine vengono riuniti tutti i sottoschemi in un unico schema generale, la cui qualità deve essere quindi analizzata

Progettazione : Qualità dello schema concettuale

- La qualità dello schema concettuale dipende dalla sua:
  - Correttezza: Vengono usati propriamente i costrutti? Esistono errori sintattici o semantici?
  - Completezza: Vengono rappresentati tutti i dati di interesse? Tutte le operazioni possono essere eseguite a partire dai concetti dello schema?
  - Leggibilità: Lo schema rappresenta in maniera semplice i requisiti? Sono disposti al centro i costrutti con più legami? Sono evitate al meglio linee intersecate? I figli sono disposti sopra i padri?
  - Minimalità: Le informazioni sono tutte rappresentate una sola volta? Esistono ridondanze, e se lo sono, sono dovute ad una scelta voluta di progettazione (ad esempio per risparmiare tempo su una determinata operazione), o sono dovute ad un errore di progettazione?
- Una volta analizzata la qualità dello schema E-R, qualora essa sia ritenuta soddisfacente, si passa alla progettazione logica

Progettazione Logica

- La progettazione logica segue quella concettuale.
- Essa è divisa in due fasi:
  - Ristrutturazione dello schema E-R
  - Traduzione in Modello Logico
- La ristrutturazione dello schema è indipendente dal modello logico stesso e si basa su criteri di ottimizzazione e semplificazione dello schema.
- La traduzione fa riferimento ad uno specifico modello logico (ad esempio quello relazionale), e include ulteriori ottimizzazioni specifiche al modello scelto, come ad esempio la normalizzazione nel caso della logica relazionale.