NUMERO: 1809A - ANNO: 2015

A P P U N T I

STUDENTE: Tosti Michela

MATERIA: Basi di dati (appunti + esercizi + esami) - Prof. Farinetti

Base di Dati - dalma Fanfinetti -

BASE DI DATI: collezione di dati che rappresenta le informazioni di interesse per un sistema informativo.

↳ DEFINIZIONE GENERALE

BASE DI DATI: collezione di dati gestita da un DBMS (data base menagment system)

(HA IL MINIMO POSSIBILE DI DATI, I QUALI NON SONO MAI RIPETUTI)

↳ DEFINIZIONE TECNICA

DBMS: sistema che GESTISCE COLLEZIONI DI DATI che sono:

- GRANDI (molti dati)
- CONDIVISI (tanti utenti devono poterci accedere contemporan.)
- PERSISTENTI (devono durare molto)

assicurando:

- AFFIDABILITA' (non devono essere persi)
- PRIVATEZZA (non tutti devono poterci accedere)

caratterizzato da:

- EFFICIENZA (lavora in tempi accettabili)
- EFICACIA (capacità di rendere produttiva l'attività degli utenti)

Mediante le BASE DI DATI posso sapere DOVE STANNO e COME USARE i DATI di interesse

- elementari
- complessi (es: ARRAY)

⇨ Il modello di dati più diffuso è il MODELLO RELAZIONALE che è l'insieme delle TABELLE (RELAZIONI) che lo costituiscono.

BASE DATI RELAZIONALI; divisibile in:

1) SCHEMA descrive la STRUTTURA DEI DATI, costituito dall'INTESTAZIONE DELLA TABELLA: ATTRIBUTO
2) ISTANZA costituita dal CONTENUTO di ogni TABELLA, cioè dagli effettivi valori dei dati

TIPI DI MODELLO

1) CONCETTUALE → Permette di rappresentare i dati in modo indipendente dal modello logico
2) LOGICO → descrive la struttura dati nel DBMS

INDIPENDENZA DATI

INDIPENDENZA FISICA consente di interagire con il DBMS in modo indipendente dalla struttura dei dati

INDIPENDENZA LOGICA: Dipende di interagire con il livello esterno in modo indipendente dal modello logico

CHIAVE: PROPRIETÀ DI UNO SCHEMA

DEFINIZIONE: Un INSIEME DI K ATTRIBUTI È CHIAVE DI UNA RELAZIONE SE:

1) la relazione non contiene 2 RIGHE CON GLI STESSI VALORI per K → UNIVOCITÀ
2) non ESISTONO SOTTOINSIEMI PROPRI DI K ancora UNIVOCI → K MINIMALE

N.B.: SE È VERIFICATA SOLO LA 1° PROPRIETÀ, K È UNA SUPERCHIAVE MA non una CHIAVE

⇨ SE UNA CHIAVE PUÒ ASSUMERE IL VALORE NULL SI perde la PROPRIETÀ DI UNIVOCITÀ DELLA CHIAVE

↳ QUINDI: SI limita la presenza di VALORI NULL così:

(vedi pg 38)

1- SI DEFINISCE UNA CHIAVE PRIMARIA, essa
   - non ammette valori null
   - PERMETTE IL RIFERIMENTO TRA I DATI IN ≠ RELAZIONI
2- LE ALTRE CHIAVI SONO LE CHIAVI CANDIDATE, esse possono ammettere valori null (pg 39)

2) VINCOLO D'INTEGRITÀ REFERENZIALE: se ho dei VALORI che DEVONO SERVIRE PER FARE RIFERIMENTO AD ALTRE TABELLE, TALI VALORI DOVRANNO ESSERE PRESENTI ANCHE NELLE ALTRE TABELLE

(vedi pg 36)

⇨ date 2 RELAZIONI
- R (REFERENZIATA) ⇒ a cui faccio RIFERIMENTO
- S (REFERENZIANTE) ⇒ che FA RIFERIMENTO ↳ MEDIANTE L'INSIEME DI ATTRIBUTI X

$S \xrightarrow{X} R$

I VALORI ASSUNTI DALL'INSIEME X DI S possono essere esclusivamente VALORI ASSUNTI EFFETTIVAMENTE DALLA CHIAVE PRIMARIA DI R

↳ L'INSIEME DEGLI ATTRIBUTI X DI S COSTITUISCE UNA CHIAVE ESTERNA DI S.

3) VINCOLO DI DOMINIO: esprime CONDIZIONI SUL VALORE ASSUNTO DA UN singolo attributo di una tupla

(pg 36)

↳ • può essere un'espressione booleana (AND, OR, NOT) di predicati semplici

4) VINCOLO DI TUPLA: esprime CONDIZIONI sul valore assunto da singole tuple, in modo indipendente dalle altre tuple della relazione

(pg 36)

↳ • può essere un'espressione booleana
  • può correlare attributi ≠

ESEMPIO: Prezzo = Costo + PercIVA*Costo

PROIEZIONE estrazione VERTICALE dalla RELAZIONE

$R = \Pi_L A$ → LISTA DEGLI ATTRIBUTI

⇨ genera una relazione R

- AVENTE COME SCHEMA LA LISTA DI ATTRIBUTI DI L
- CONTENENTE TUTTE LE TUPLE DI A

esempio: TROVARE IL nome dei docenti

$R = \Pi_{nomeDOC}(DOCENTI)$ oppure

R
=
$\Pi_{nome\ docenti}$
|
DOCENTI

⇨ SONO ELIMINATE LE RIGHE = (DOPO LA PROIEZIONE)

esempio: TROVA I nomi DEI DIPARTIMENTI con almeno un professore

MATDOCENTE	NOMEDOC	DIPARTIM.
01	LUCA	INFO
02	MARTA	GEOM
03	GIGI	INFO

DOCENTI

$R = \Pi_{DIPARTIMENTO}(DOCENTI)$ RISULTATO:

DIPARTIM.
INFO
GEOM
~~INFO~~

SELEZIONE + PROIEZIONE

es: TROVARE i nomi DEI CORSI del II SEMESTRE

1° individuo i corsi del II semestre
2° estraggo i nomi

$R = \Pi_{nome\ corso}(\sigma_{semestre=2}(CORSI))$ oppure

R
=
$\Pi_{nome\ corso}$
|
$\sigma_{semestre=2}$
|
CORSI

=> RICORDA! SI FA prima la selezione poi la proiezione, altrimenti PERDO INFORMAZIONI.

LEZIONE 3
3/10/2014

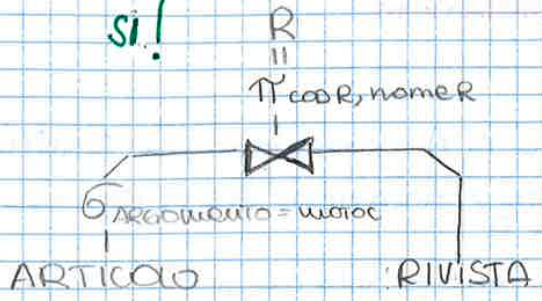
ESERCIZIO:

1) RIVISTA (CODR, NomeR, Editore)
ARTICOLO (CODA, Titolo, Argomento, CODR) → chiave esterna che mi dice in che rivista è stato pubblicato.

A) TROVA CODICE E nome RIVISTA che ha pubblicato almeno un ARTICOLO DI ARGOMENTO MOTOCICLISTICO

SOLUZIONE:

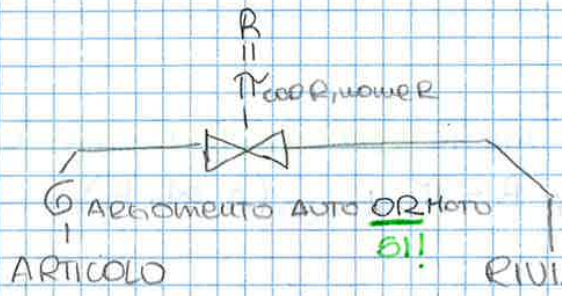
SI!



RICORDA:

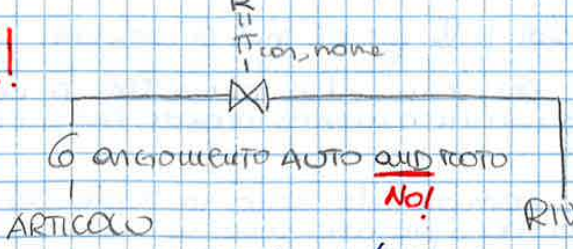
NO!

B) TROVA CODICE e nome RIVISTE che hanno pubblicato almeno 1 ARTICOLO DI motociclismo O AUTO.



C) TROVA CODICE E nome RIVISTE che hanno pubblicato almeno UN ARTICOLO DI MOTOCICLISTICO E AUTO

NO!!!



Lo RIPRENDIAMO PIÙ AVANTI....

↳ Perché altrimenti in una cosella ci sarebbe + di un valore e questo non può succedere!

RIPRENDENDO IL THETA JOIN, se per esempio volessi prendere un docente con almeno 2 corsi DATA:

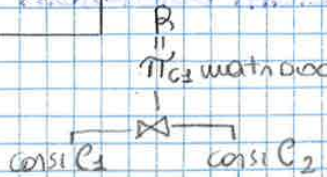
$C_1 \Rightarrow$

CODICE	INSEG.	CORSO
~	~	~
~	~	~

DEVO Duplicare la TABELLA (dando alla seconda nome C_2) E UNIRE le 2 TABELLE

↳ POI DEVO Tenere solo le INFO che voglio

$P = C_1.matr\,doc = C_2.matr\,doc \wedge C_1.codice <> C_2.codice$

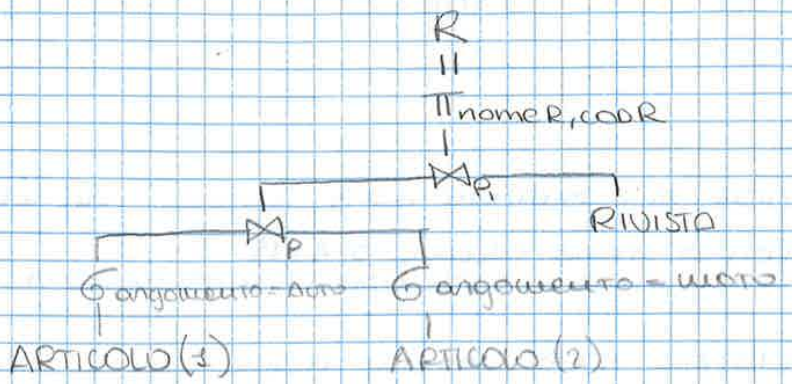


FULL OUTER-JOIN

$R = A \overline{\bowtie} B$ — schema: unione schemi A e B

=> COMMUTATIVO

RIPRENDENDO ORA IL PUNTO (e) DELL'ESERCIZIO

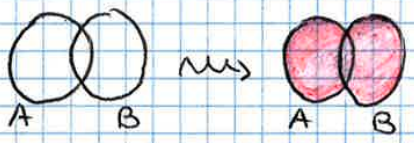


- P_1 = ARTICOLO(1).CODR = ARTICOLO(2).CODR
- P_2 = ARTICOLO(2).CODR = RIVISTA.CODR

LEZIONE 4
6/10/2014

UNIONE

d'unione di due relazioni seleziona gli elementi presenti in almeno una delle 2.



(ELIMINA I DUPLICATI)

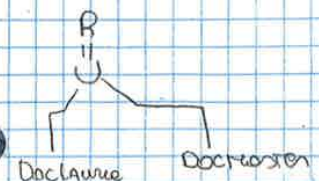
Definizione: $R = A \cup B$ (ha tutte le righe di entrambi con l'eliminazione dei duplicati)

- d'unione di 2 RELAZIONI A e B genera una relazione R:
  - AVENTE STESSO SCHEMA DI A e B
  - CONTENENTE TUTTE LE TUPLE DI A e DI B

COMPATIBILITÀ:
- LE RELAZIONI DEVONO AVERE LO STESSO SCHEMA

⇨ - COMMUTATIVA
- ASSOCIATIVA

Esempio: TRADUCI INTO relative ai docenti dei corsi di Laurea (e) di Master



R = DocLaurea ∪ DocMaster

ANTI-JOIN

L'ANTI-JOIN seleziona tutte le TUPLE DI A 'SEMANTICAMENTE NON LEGATE' a B.

↳ LE INFO DI B non compaiono nel risultato.

$$R = A \bar{\bowtie}_p B$$

USABILE al POSTO DELLA DIFFERENZA

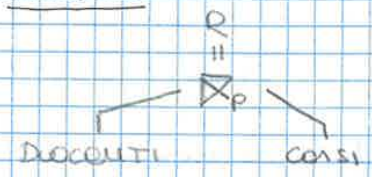
DEFINIZIONE: L'ANTI-JOIN DI 2 relazioni A e B GENERA una relazione R:

- CON LO STESSO SCHEMA DI A
- CONTENENTE TUTTE le TUPLE DI A per cui non esiste alcuna TUPLA in B per cui è vero il predicato p.

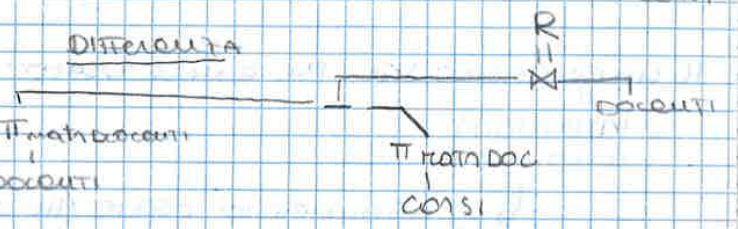
⇨ • NON COMMUTATIVO
• NON ASSOCIATIVO

ESEMPIO: TROVA MATRICOLA, NOME e DIPARTIMENTO docenti che NON tengono corsi

ANTI-JOIN



DIFFERENZA



p: Docenti.MatrDocenti = Corsi.MatrDocenti

DIVISIONE

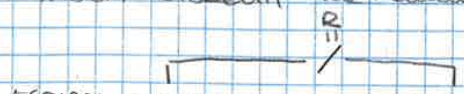
$$R = A / B$$

DEFINIZIONE: DIVISIONE della relaz. A per la relaz B genera una relaz R:

- AVENTE SCHEMA = SCHEMA (A) - SCHEMA (B)
- CONTENENTE TUTTE le TUPLE DI A tali che per ogni tupla (Y: y) presente in B ESISTE una TUPLA (X: x, Y: y) in A

⇨ • NON COMMUTATIVA
• NON ASSOCIATIVA

ESEMPIO: TROVA STUDENTI che hanno SUPERATO TUTTI I CORSI DEL I ANNO



R = E.Superati / C. Primo Anno

- DEFINIZIONE DI STRUTTURE DATI ACCESSORIE PER RECUPERARE EFFICIENTEMENTE I DATI
  - CREAZIONE, CANCELLAZIONE INDICI
    - CREATE, DROP INDEX
- DEFINIZIONE PRIVILEGI ACCESSO UTENTI
  - CONCESSIONE / REVOCA PRIVILEGI SULLE RISORSE
    - GRANT, REVOKE
- DEFINIZIONE DI TRANSAZIONI
  - TERMINAZIONE DI UNA TRANSAZIONE
    - COMMIT, ROLLBACK

NOTAZIONE SQL:

- Parole chiave → caratteri maiuscoli e colore blu scuro
- Termini variabili → corsivo

GRAMMATICA SQL

PARAMETRI SINGOLARI < >
(ISOLANO IL TERMINE DELLE SINTASSI)

PARENTESI QUADRE [ ]
(INDICANO CHE IL TERMINE È OPZIONALE)

PARENTESI GRAFFE { }
(INDICANO CHE IL TERMINE PUÒ NON COMPARIRE O ESSERE RIPETUTO n VOLTE)

BARRA VERTICALE |
(INDICANO CHE DEVE ESSERE SCELTO UNO DEI TERMINI)

Esempio:

DB FORNITURE PRODOTTI

- Tabella P descrive FORNITURE PRODOTTI
  - CHIAVE PRIMARIA codP
- Tabella F descrive i FORNITORI
  - CHIAVE PRIMARIA codF
- Tabella FP descrive FORNITURE, METTENDO IN RELAZIONE i PRODOTTI con i FORNITORI che li FORNISCONO
  - CHIAVE PRIMARIA (codF, codP)

P

codP	NomeP	Colore	Taglia
...	...	...	...

F

codF	NomeF	N.Soci
...	...	...

FP

codF	codP	QTà
...	...	...

Clausola WHERE

→ Permette di esprimere condizioni di selezione applicate singolarmente ad ogni tupla

→ espressione Booleana di predicati

→ predicati semplici
- espressioni di confronto tra attributi e costanti
- Ricerca testuale
- Valori NULL

es: trova codice fornitori Milano

```
SELECT codF,
FROM F
WHERE sede = 'Milano';
```

es2: Trovare il codice dei fornitori di Milano con più di 2 soci

```
SELECT codF
FROM F
WHERE sede = 'Milano' AND NSoci > 2;
```

OPERATORE LIKE

Nome Attributo LIKE Stringa di Caratteri

- il carattere '_' rappresenta un singolo carattere qualsiasi (obbligatoriamente presente)
- il carattere % rappresenta una sequenza qualsiasi di n caratteri (anche vuota).

es: trova codice e nome dei prodotti il cui nome inizia con la C

```
SELECT CodP, NomeP
FROM P
WHERE NomeP LIKE 'C%'
```

esempio 2: il codice del fornitore è pari a 2 e
- → preceduto da un carattere qualsiasi
- → costituito esattamente da 2 caratteri

```
CodF LIKE '_2'
```

esempio 3: l'attributo magazzino non contiene una e in 2ª posizione

```
Magazzino NOT LIKE '_e%'
```

NOTA Bene: se in qualche riga ho la presenza di NULL non posso usare >, <, = perché il NULL non viene da essi considerato.

OPERATORE IS

Nome Attributo IS [NOT] Null

esempio: trova nome e codice prodotti per cui la Tg non è indicata

```
select Nome, codP
From P
where Taglia IS Null
```

Esempio: trovare coppie e codici fornitori entrambi i fornitori abbiano sede nella stessa città

```
select FX.codF, FY.codF
from F AS FX, F AS FY
where FX.sede = FY.sede
```

ma sono presenti • coppie di valori =
• permutazioni delle stesse coppie di valori

↳ `where FX.sede = FY.sede AND FX.CodF <> FY.CodF;`

Elimino coppie di valori =

↳ `where FX.Sede = FY.sede AND FX.CodF < FY.CodF`

Elimina permutazione stessa coppia di valori

SINTASSI ALTERNATIVA JOIN

```
Select [DISTINCT] attributi
From Tabella tipo JOIN JOIN Tabella ON Condizione di Join
Where condizione di tupla
```

TIPO JOIN = < INNER | [FULL | LEFT | RIGHT] OUTER >

ESEMPIO INNER JOIN

Trova nome fornitori che forniscono almeno 1 prodotto rosso

```
Select NomeF
FROM F INNER JOIN FP ON PCod/P = FP.Cod.P
       INNER JOIN F.CodF = FP.CodF
where P.Colore = 'Rosso';
```

ESEMPIO OUTER JOIN

Trova COD e NOME fornitori insieme al codice dei relativi prodotti forniti, visualizzando anche i fornitori che non hanno forniture.

```
Select F.CodF, NomeF, CodP
FROM LEFT OUTER JOIN FP ON F.CodF = FP.CodF
```

FUNZIONI AGGREGATE

- operano su insiemi di valori
- produce come risultato un unico valore → AGGREGATO
- è INDICATA NEL select

↳ possono essere richieste più funzioni aggregate contempor.

ELENCO

COUNT	conteggio elementi in un attributo
SUM	somma valori di un attributo
AVG	media valori di un attributo
MAX	massimo valore di un attributo
MIN	minimo valore di un attributo

OPERATORE GROUP BY

ESEMPIO 1: PER OGNI PRODOTTO, TROVA QUANTITÀ TOTALE PEZZI FORNITI

```
SELECT CodP, Sum(Qta)
FROM FP
GROUP BY CodP;
```

⇒ clausola di raggruppamento

GROUP BY Elenco attributi di raggruppamento
{ ordine ininfluente

⇒ nella clausola SELECT possono comparire solo

- Attributi presenti nella clausola group by
- Funzioni aggregate

ESEMPIO 2: Per ogni prodotto trova la quantità totale dei pezzi forniti da fornitori con sede a Milano

```
Select CodP, Sum(Qta)
From FP, F
Where F.codF = FP.CodF and Sede = 'Milano'
Group By CodP;
```

Ricorda: i prodotti senza fornitore non sono inclusi

⇒ Artificio sintattico: Gli attributi univocamente determinati da attributi già presenti nella clausola GROUP BY possono essere aggiunti senza alterare il Risultato.

SCHEMA:

```
SELECT [DISTINCT] Elenco attributi da visualizzare
FROM Elenco tabelle da usare
[WHERE condizione di tupla]
[GROUP BY Elenco attributi di raggruppamento]
[ORDER BY Elenco attributi di ordinamento]
```

ESEMPIO 3: Trova quantità totale pezzi forniti per i prodotti per cui sono forniti in totale almeno 600 pezzi

```
Select CodP, SUM(Qta)
From FP
Group By CodP
Having SUM(Qta) >= 600;
```

Clausola **HAVING** ➡ permette di specificare condizioni su funzioni aggregate

ESEMPIO 4: CODICE PRODOTTI ROSSI FORNITI DA PIÙ DI 1 FORNITORE

```
SELECT FP.CodP
From P, FP
Where FP.CodP = P.CodP And Colore = 'Rosso'
GroupBy FP.CodP
Having Count(*) > 1
```

OPERATORI IN e NOT IN

IN

ESEMPIO1: TROVA NOME FORNITORI che FORNISCONO P2

```
SELECT  NOMEF
FROM    F
WHERE   CODF (IN) ( SELECT  CODF
                    FROM    FP
                    WHERE   CODP = 'P2');
```

↓ Appartenenza all'insieme

⇒ OPERATORE **IN**: ESPRIME CONCETTO DI APPARTENENZA A UN INSIEME DI VALORI

NomeAttributo IN (INTERROGAZIONE MODIFICATA)

⇒ PERMETTE DI ESPRIMERE L'INTERROGAZIONE

- SCOMPONENDO IL PROBLEMA IN SOTTO PROBLEMI
- SEGUENDO UN PROCEDIMENTO "BOTTOM-UP"

→ QUASI SEMPRE È POSSIBILE REALIZZARE UNA FORMULAZIONE = CON IL JOIN

ESEMPIO2: NOME FORNITORI CHE FORNISCONO ALMENO UN PRODOTTO ROSSO

<NIDIFICATO>

```
SELECT  NomeF
FROM    F
WHERE   CODF IN (SELECT CODF
                 FROM   FP
                 WHERE  CODP IN (SELECT CODP
                                 FROM   P
                                 WHERE  Colore = 'Rosso');
```

<JOIN>

```
SELECT  NomeF
FROM    F, FP, P
WHERE   F.CODF = FP.CODP
AND     P.CODP = FP.CODP
AND     Colore = 'Rosso';
```

NOT IN

ESEMPIO 1: Trova nome Fornitori che non Forniscono P_2

```
SELECT  Nome F
FROM    F
WHERE   CodP (NOT IN) (Select CodP
                       FROM   FP
                       WHERE  CODP = 'P2');
```

↓ non appartenenza all'insieme

NON esprimibile con JOIN

⇒ OPERATORE **NOT IN**: esprime il concetto di esclusione da un insieme di valori

Nome Attributo NOT IN (INTERROGAZIONE MODIFICATA)

⇒ RICHIEDE DI INDIVIDUARE IN MODO APPROPRIATO L'INSIEME DA ESCLUDERE

- DEFINITO DALL'INTERROGAZIONE MODIFICATA

OPERATORE EXIST

TROVARE IL NOME DEI FORNITORI DEL PRODOTTO P2

↳ TROVARE NOME FORNITORI PER CUI ESISTE UNA FORNITURA DEL PRODOTTO P2

```
SELECT NOME F
FROM F
WHERE EXIST ( SELECT *
              FROM FP
              WHERE CODP = 'P2' AND FP.CODF = F.CODF );
```

↳ CONDIZIONE DI CORRELAZIONE

IL PREDICATO CONTENENTE EXIST È

- VERO, SE RESTITUISCE ALMENO UNA TUPLA
- FALSO, SE RESTITUISCE L'INSIEME VUOTO

- NELL'INTERROGAZIONE INTERNA A EXIST, **la clausola SELECT** È OBBLIGATORIA, MA IRRILEVANTE, PERCHÉ GLI ATTRIBUTI NON SONO VISUALIZZATI.
- LA CONDIZIONE DI CORRELAZIONE LEGA L'ESECUZIONE DELL'INTERROGAZIONE INTERNA AI VALORI DI ATTRIBUTI DELLA TUPLA CORRENTE NELL'INTERROGAZIONE ESTERNA

VISIBILITÀ ATTRIBUTI:

UN INTERROGAZIONE MODIFICATA:

a) PUÒ FARE RIFERIMENTO AD ATTRIBUTI DEFINITI IN INTERROGAZIONI PIÙ ESTERNE
b) NON PUÒ FARE RIFERIMENTO AD ATTRIBUTI NEFELIBRATI:
   - IN INTERROGAZIONE MODIFICATA AL SUO INTERNO
   - IN INTERROGAZIONE ALLO STESSO LIVELLO

OPERATORE NOT EXIST

TROVARE IL NOME DEI FORNITORI CHE NON FORNISCONO P2

```
Select NOME F
FROM F
WHERE NOT EXIST ( select *
                  FROM FP
                  WHERE CODP = 'P2' AND FP.CODF = F.CODF );
```

CONDIZIONE DI CORRELAZIONE

IL PREDICATO CONTENENTE NOT EXIST È:

- VERO SE L'INTERROGAZIONE INTERNA RESTITUISCE L'INSIEME VUOTO
- FALSO SE " " " ALMENO UNA TUPLA

] OPPOSTO DI EXIST

- LA CONDIZIONE DI CORRELAZIONE LEGA L'ESECUZIONE DELL'INTERROGAZ. INTERNA AL VALORE DI ATTRIBUTI DELLA TUPLA CORRENTE NELL'INTERROGAZIONE ESTERNA

ESEMPIO 1 TROVARE IL CODICE DEI FORNITORI CHE FORNISCONO TUTTI I PRODOTTI

```
Select CODF
FROM FP
GROUP BY CODF
HAVING COUNT (*) = (select count (*)
                    FROM P);
```

NUMERO TOTALE PRODOTTI

ESEMPIO 2 TROVARE IL CODICE DEI FORNITORI CHE FORNISCONO ALMENO TUTTI I PRODOTTI FORNITI DAL FORNITORE F2

```
select CODF
FROM FP
WHERE CODP IN ( select CODP
                FROM FP
                WHERE CODF = 'F2')

GROUP BY CODF
HAVING COUNT (*) = ( select count (*)
                     FROM FP
                     WHERE CODF = 'F2' );
```

N° TOTALE PRODOTTI DI F2

TABLE FUNCTIONS

ESEMPIO STUDENTE (MATRICOLA, ANNOISCRIZIONE)

ESAMESUPERATO (MATRICOLA, CODC, DATA, VOTO)

→ TROVA MEDIA MAX DI UNO STUDENTE

SI DIVIDE IN: TROVA MEDIA DI OGNI STUDENTE ①
TROVA IL VALORE MAX DELLA MEDIA ②

```
select MAX(MEDIA STUDENTI)                                  } ②
FROM (select Matricola, AVG (VOTO) AS MEDIA STUDENTI        } ①
      FROM ESAMESUPERATO
      GROUP BY MATRICOLA) AS MEDIE)
```

TABLE FUNCTION

DEFINIZIONE: E' LA DEFINIZIONE DI UNA tabella temporanea CHE PUO' essere USATA per ULTERIORI OPERAZIONI DI CALCOLO

- HA LA STRUTTURA DI UN SELECT
- DEFINITA ALL'INTERNO DELLA clausola FROM
- PUO' ESSERE REFERENZIATA COME UNA NORMALE TABELLA

permette DI:

- Calcolare più livelli di AGGREGAZIONE
- FORMULARE IN MODO EQUIVALENTE LE INTERROGAZIONI che richiedono la CORRELAZIONE

ESEMPIO Trovare le città sede di fornitori e sede magazzino prodotti

```
select sede
from F
intersect
select Magazzino
from P;
```

NOTA BENE: l'operazione di intersezione può essere eseguita anche mediante
- JOIN
- IN

ESEMPIO EQUIVALENZA CON JOIN

- la clausola FROM contiene le relazioni interessate dall'intersezione
- la clausola WHERE contiene condizioni di join tra gli attributi presenti nella clausola SELECT delle espressioni relazionali A e B

→ Trova città sia sede di fornitori, sia sede di magazzini

```
select sede
from F, P
where F.sede = P.magazzino;
```

ESEMPIO EQUIVALENZA CON IN

- Una delle 2 espressioni relazionali diviene una interrogazione modificata mediante l'operatore IN
- Gli attributi nella clausola SELECT esterna, uniti da un costruttore di tupla, costituiscono la parte sx dell'operatore IN

→ Trova città sia sede di fornitori, sia sede di magazzini

```
select Magazzino
from P
where Magazzino IN (select sede
                    from F);
```

---

3. EXCEPT

Operatore insiemistico di differenza (A − B) — except

Sottrae l'espressione relazionale B all'espressione relazionale A

↳ Richiede compatibilità di schema tra A e B

ESEMPIO Trova città sede di fornitori, ma non magazzino di prodotti

```
Select Sede
from F
except
Select Magazzino
from P;
```

NOTA BENE: l'operazione di differenza può essere eseguita anche mediante l'operatore NOT IN

→ L'espressione B è modificata mediante il NOT IN
→ Gli attributi della clausola SELECT di A, uniti da un costruttore di tupla costituiscono la parte sx del NOT IN

ESEMPIO 2 Inserire il prodotto P8 con città: Genova, Taglia: 42

```
INSERT INTO P (CodP, Citta, Taglia)
VALUES('P8', 'Genova', 42)
```

È inserita nella tabella P una nuova tupla con i valori specificati
↳ a NomeP e Colore è assegnato il valore NULL

Per tutti gli attributi, il cui valore non è specificato, il Dominio dell'attributo deve consentire il valore NULL

ESEMPIO 3 → INSERIMENTO: INTEGRITÀ REFERENZIALE

Inserire una nuova Fornitura relativa al Fornitore F20, prodotto P20 e QUANTITÀ 1000

VINCOLO D'INTEGRITÀ REFERENZIALE: è necessario che P20 e F20 siano già presenti rispettivamente in P e F
↳ se non soddisfatto, impossibile l'inserimento

```
INSERT INTO FP (CodF, CodP, Qta)
VALUES ('F20', 'P20', 1000)
```

Inserimento di più RECORD

```
INSERT INTO NomeTabella
[(ElencoColonne)];
InterrogazioneSQL;
```

- Sono inserite in NomeTabella tutte le tuple selezionate dall'interrogazione InterrogazioneSQL
- InterrogazioneSQL è una istruzione SELECT arbitraria
  ↳ NON può contenere la clausola ORDER BY

ESEMPIO

FORNITURE-TOTALI (CodP, TotQta)

→ Per ogni prodotto, inserire in Forniture-totali la quantità totale fornita

```
INSERT INTO FORNITURE-TOTALI (CodP, TOT QTA)
(select CodP, SUM(Qta)
 from FP
 GROUP BY CodP);
```

3. Update

```
UPDATE Nome Tabella
SET colonna = Espressione
{ colonna = espressione }
[WHERE predicato] ;
```

⇒ Tutti i RECORD della tabella Nome Tabella che SODDISFANO i predicati sono MODIFICATI in base alle assegnazioni colonna = Espressione nella clausola SET

ESEMPIO 1 AGGIORNARE le caratteristiche del prodotto P1: assegnare Giallo al colore, incrementare la taglia di 2 e assegnare NULL a città

```
UPDATE P
set Colore = 'Giallo'
    TAGLIA = Taglia+2,
    CITTA' = Null
WHERE CodP = 'P1' ;
```

→ E' aggiornata la TUPLA individuata nel codice P1

AGGIORNAMENTO MULTIPLO

ESEMPIO 2 AGGIORNARE N° Soci al doppio del valore per TUTTI i FORNITORI di Milano

```
UPDATE F
set NSoci = 2* N.Soci
WHERE Citta' = 'Milano';
```

→ Sono aggiornate tutte le TUPLE indicate dal predicato nella clausola WHERE

AGGIORNAMENTO CON SOTTOINTERROGAZIONE

ES 3 AGGIORNA A 10 LE QUANTITA' FORNITE x TUTTI i FORNITORI di Milano

```
UPDATE FP
SET QTA = 10
WHERE CodF IN ( Select CodF
                FROM F
                WHERE Citta' = 'Milano' ; )
```

AGGIORNAMENTO DI PIÙ TABELLE

ESEMPIO 4 MODIFICARE con il valore F9 il CODICE del FORNITORE F2

```
UPDATE F
SET CodF = 'F9'
WHERE CodF = 'F2'
```

Se in FP esistono FORNITURE che fanno RIFERIMENTO ai CODICI dei FORNITORI aggiornati, è VIOLATO IL VINCOLO DI INTEGRITA' REFERENZIALE

↳ VANNO aggiornate anche in FP

```
UPDATE F
SET CodF = 'F9'
WHERE CodF = 'F2';

UPDATE FP
SET CODF = 'F9'
WHERE CodF = 'F2'
```

NUMERICI ESATTI

- NUMERIC [(Precisione, Scala)]
- DECIMAL [(Precisione, Scala)]
- INTEGER
- SMALLINT

↓ numeri in base decimale

Precisione:
- → n° totale di cifre
- → per il dominio NUMERIC la precisione rappresenta il valore esatto
- → per il dominio DECIMAL la precisione costituisce un requisito minimo

Scala: n° cifre dopo la virgola

ESEMPIO: 133.45 Precisione = 3, Scala = 2

NUMERICI APPROSSIMATI

- FLOAT [(n)] → specifico la precisione
- REAL
- DUBLE PRECISION

'n'

- specifica il n° di bit utilizzati per memorizzare la mantissa di un n float rappresentato in notazione scientifica
- valore tra 1 e 53
- valore di default 53

INTERVAL Prima unità di tempo [TO ultima unità di tempo]

→ le unità di tempo sono divise in 2 gruppi
- anno, mese
- giorno, ora, minuti, secondi

ESEMPIO INTERVAL Year TO month
- memorizza periodo di tempo usando i campi anno e mese

TIME STAMP [(Precisione)] [WITH TIME ZONE]

- Memorizza i valori che specificano anno, mese, giorno, ora, minuti, secondi
- 19 caratteri + quelli per la precisione
- YYYY-MM-DD hh:mm:ss:p

ESEMPI ⇒ ISTRUZIONE ALTER TABLE

ES 1 AGGIUNGERE LA COLONNA - N° DIPENDENTI - ALLA TABELLA DEI FORNIT.

```
ALTER TABLE F
ADD COLUMN NDipendenti SMALLINT;
```

ES 2 ELIMINA COLONNA NSoci DAL FORNITORI

```
ALTER TABLE F
DROP COLUMN NSoci RESTRICT;
```

ES 3 Aggiungere il valore di default 0 alla colonna QUANTITÀ della tab delle Forniture

```
Alter TABLE FP
Alter column Qta SET DEFAULT 0;
```

CANCELLAZIONE DI UNA TABELLA

```
DROP TABLE Nome Tabella
           [RESTRICT / CASCADE]
```

→ Tutte le righe della tabella sono eliminate insieme a essa

RESTRICT → LA TABELLA non è rimossa se è presente in qualche definizione di tabella, vincolo, viste o opzione di default

CASCADE → Se la tabella compare in qualche definizione di viste; anche queste è rimossa

ESEMPIO ⇒ cancellazione tabelle : cancella tabella Fornitori

```
DROP TABLE F;
```

DIZIONARIO DEI DATI

⇨ I metadati sono informazioni sui dati
- possono essere memorizzati in tabelle della base di dati

⇨ Il dizionario dei dati contiene i metadati di una base di dati relazionale
- contiene INFO sugli oggetti della base di dati
- è gestito direttamente dal **DBMS** relazionale
- può essere interrogato con istruzioni **SQL**

⇨ CONTIENE DIFFERENTI INFO
- DESCRIZIONE DI TUTTE le STRUTTURE della base di dati
- STORED PROCEDURE SQL
- PRIVILEGI DEGLI UTENTI
- STATISTICHE
  - sulle tabelle della base di dati
  - sugli indici della base di dati
  - sulle viste della base di dati
  - sulla crescita della base di dati

INFO sulle Tabelle

⇨ IL DIZIONARIO dei dati contiene per ogni TABELLA della BASE DI DATI
- NOME della tabella e STRUTTURA FISICA del FILE in cui è memorizzata
- NOME e TIPO di dato per ogni attributo
- NOME di TUTTI GLI INDICI creati sulla tabella
- VINCOLI D'INTEGRITÀ

Procedure Applicative

⇨ All'interno di ogni applicazione sono previste tutte le verifiche di correttezza necessarie

⇨ Vantaggi: Approccio molto efficiente

⇨ Svantaggi:
- Possibile aggirare le verifiche interagendo direttamente con il DBMS
- Un errore di codifica può avere un effetto significativo sulla BDD
- La conoscenza di regole di correttezza è tipicamente "nascosta" nelle applicazioni

Vincoli integrità sulle Tabelle

⇨ Sono:
- definiti nelle istruzioni Create o Alter Table
- memorizzati nel dizionario dati del sistema

⇨ Durante l'esecuzione di qualunque operazione di modifica dei dati il DBMS verifica automaticamente che i vincoli siano osservati

① VANTAGGI
- Definizione dichiarativa dei vincoli, la cui verifica è affidata al sistema
- Dizionario dei dati descrive tutti i vincoli presenti nel sistema
- Unico pt centralizzato di verifica → impossibile aggirare la verifica dei vincoli

② SVANTAGGI
- possono rallentare l'esecuzione delle applicazioni
- non è possibile definire tipologie arbitrarie di vincoli

TRIGGER

⇨ Procedure automatiche quando si verificano opportune modifiche dei dati
- definiti dall'istruzione CREATE TRIGGER
- memorizzati nel dizionario dati del sistema

① VANTAGGI
- Permettono di definire vincoli di tipo complesso
- Unico pt centralizzato di verifica

② SVANTAGGI
- Applicativamente complessi
- possono rallentare l'esecuzione delle applicazioni

AMMISSIBILITÀ DEL VALORE NULLO

→ NULL, indica omessa INFO
→ quando è OBBLIGATORIO specificare sempre un valore per l'attributo

Nome Attributo	Dominio	NOT NULL

→ valore nullo non è ammesso

Esempio:

```
CREATE TABLE F (CodF CHAR(5),
                NomeF CHAR(20) NOT NULL,
                NSoci SMALLINT,
                Sede CHAR(15));
```

UNICITÀ: un attributo, o un insieme di attributi non può assumere lo stesso valore in righe ≠ della tabella

Per un solo ATTRIBUTO:

Nome Attributo	Dominio	UNIQUE

Per 1 o + ATTRIBUTI:

UNIQUE (Elenco attributi)

→ AMMESSA: ripetizione valore null

(Pg 3) CHIAVE CANDIDATA
- insieme di attributi, potrebbe assumere il ruolo di C. PRIMARIA
- univoca
- permette valore nullo

N.B.
I) La combinazione UNIQUE NOT NULL permette di definire una chiave candidata che non ammette valori Nulli

Nome Attributo	Dominio	UNIQUE	NOT NULL

ESEMPIO UNICITÀ

```
CREATE TABLE (CodP CHAR(6),
              NomeP CHAR(20) NOT NULL UNIQUE,
              Colore CHAR(6)
              Tg SMALLINT,
              Magazzino CHAR(15));
```

ESEMPIO → GESTIONE VINCOLI

Tabella FP (REFERENZIANTE)

- INSERT (NUOVA TUPLA) → NO
- UPDATE (COD_F) → NO
- DELATE (tupla) → OK

TABELLA F (REFERENZIATA)

- INSERT (NUOVA TUPLA) → OK
- UPDATE (COD F) → aggiornare in cascata (CASCADE)
- DELATE (tupla) → aggiornare in cascata (CASCADE)
  IMPEDIRE L'AZIONE (NO ACTION)

ESEMPIO 2 → GESTIONE VINCOLI

- IMPIEGATI (Matr, NomeI, Residenza, DNum)
- DIPARTIMENTI (DNUM, DNome, Sede)

IMPIEGATI (REFERENZIANTE)

- INSERT (nuova tupla) → NO
- UPDATE (DNUM) → NO
- DELATE (TUPLA) → OK

DIPARTIMENTI (REFERENZIATA)

- INSERT (NUOVA TUPLA) → OK
- UPDATE (DNUM) → aggior. IN CASCATA (CASCADE)
  " " " IMPEDIRE AZIONE (NO ACTION)
  " " " IMPOSTARE A VALORE NULLO (SET NULL)
  " " " IMPOSTARE A VALORE DI DEFAULT (SET DEFAULT)
- DELATE (TUPLA) →

POLITICHE DI GESTIONE DEI VINCOLI

→ I vincoli d'integrità sono verificati dopo ogni istruzione SQL che potrebbe causarne la violazione

→ NON sono ammesse operazioni di inserimento e modifica della tabella referenziante che violano il vincolo

OPERAZIONI DI MODIFICA/CANCELLAZIONE DELLA TABELLA REFERENZIATA CAUSANO SULLA TABELLA REFERENZIANTE

- CASCADE: PROPAGAZIONE DELL'OPERAZIONE DI AGGIORNAMENTO O CANCELLAZIONE
- SET NULL / DEFAULT: NULL O VALORE DEFAULT IN TUTTE LE COLONNE DELLE TUPLE CHE HANNO VALORI NON + PRESENTI NELLA TAB. REFERENZIATA
- NO ACTION: NON ESEGUE L'AZIONE IN VIOLAZIONE

GESTIONE VISTE

- una VISTA è una tabella "virtuale"
  - il contenuto (tuple) è definito mediante un'interrogazione SQL sulla BDD
  - il contenuto della vista dipende dal contenuto delle altre tabelle presenti nella BDD
  - contenuto NON memorizzato fisicamente nella BDD
  - la VISTA è un oggetto della BDD
  - utilizzabile nelle interrogazioni come se fosse una TABELLA

ESEMPIO 1 . DEFINIZIONE della VISTA

- Definizione della VISTA 'piccoli fornitori' → tutti i fornitori con meno di 3 soci
- LA VISTA 'piccoli fornitori' contiene
  - codice
  - nome
  - numero di soci
  - sede

```
CREATE VIEW Piccoli_Fornitori AS
SELECT CodF, NomeF, NSoci, Sede
FROM F
WHERE NSoci < 3
```

(CREATE VIEW Piccoli_Fornitori AS → Nome della vista; SELECT ... WHERE NSoci < 3 → interrogazione associata alla vista)

. INTERROGAZIONE

- Visualizza .codice dei piccoli fornitori di Torino
  - nome
  - sede
  - N° soci

① Posso risolvere senza l'uso di viste

```
SELECT *
FROM F
WHERE NSoci < 3 AND Sede = 'Torino'.
```

② L'interrogazione può essere risolta con la VISTA di prima

```
SELECT *
FROM Piccoli_Fornitori
WHERE Sede = 'Torino'
```

↳ USATA come TABELLA

. Riscrittura dell'interrogazione

→ se l'interrogazione fa riferimento a 1 vista, deve essere riscritta dal DBMS prima dell'esecuzione

→ Riscritture è svolta AUTOMATICAMENTE

~ Visualizza il codice, il nome, sede, n° soci dei piccoli fornitori di B

```
SELECT *
FROM Piccoli_Fornitori
WHERE Sede = 'Torino';
```

Piccoli_Fornitori → F

```
SELECT CodF, NomeF, Sede, NSoci
FROM F
WHERE NSoci < 3 AND Sede = 'Torino'
```

→ Si rendono espliciti gli attributi

Drop View Nome Vista

CANCELLAZIONE VISTA

- La cancellazione di una tabella a cui fa riferimento una vista può avere i seguenti effetti:
  - ELIMINAZIONE AUTOMATICA DELLE VISTE ASSOCIATE
  - INVALIDAZIONE " " " " "
  - DIVIETO DI esecuzione dell'operazione di cancellaz. della tabella

L'effetto dipende dal DBMS usato

ALTER VIEW Nome Vista [(elenco attributi)]
AS Interrogazione SQL;

AGGIORNABILITA' VISTE

- È possibile eseguire operazioni d'aggiornamento dei dati presenti in 1 vista solo per alcune tipologie di viste
  - Standard SQL-92 → Aggiornabili le viste in cui 1 sola riga di ciascuna tabella di base corrisponde a 1 sola riga della vista
    - corrispondenza univoca tra le tuple della vista e della tabella
    - possibile propagare le modifiche su ogni tabella in cui è definita la vista

- non è possibile aggiornare una vista che, nel blocco più esterno dell'interrogazione che la definisce:
  - non contiene la chiave primaria della tabella su cui è definita
  - contiene join che rappresentano corrispondenze 1-molti o molti-molti
  - contiene funzioni aggregate
  - contiene DISTINCT

ESEMPIO 1 Vista FORNITORE_SEDE

```
CREATE VIEW Fornitore_sede AS
SELECT CodF, Sede
FROM F;
```

INSERIMENTO (inserimento fornitore sede di [illegible])
→ inserisce in F una [illegible]
→ [illegible]

CHECK OPTION

→ per le VISTE AGGIORNABILI può essere USATA la clausola WITH CHECK OPTION → LIMITA AGGIORNAMENTI POSSIBILI

```
CREATE VIEW Nome Vista [(Elenco Attributi)]
AS Interrogazione SQL
[WITH [LOCAL | COSCADED] CHECK OPTION];
```

→ DOPO UN AGGIORNAMENTO le TUPLE DEVONO ANCORA APPARTENERE alla VISTA

→ permette di INSERIRE UNA NUOVA TUPLA nella VISTA se e solo se la TUPLA SODDISFA i VINCOLI PRESENTI nella DEFINIZIONE della VISTA

ESEMPIO 1

```
CREATE VIEW Prodotti_Taglia_Media_a_Grande
(CodP, NomeP, Taglia) AS
SELECT CodP, NomeP, Taglia
FROM P
WHERE Taglia >= 42
WITH CHECK OPTION
```

∑ LA VISTA È AGGIORNABILE → non si possono aggiornare i valori della VISTA con Tg < 42 (valori di)

→ QUANDO 1 VISTA È DEFINITA IN TERMINI DI ALTRE VISTE

- SE SI SPECIFICA **LOCAL**
  - la correttezza dell'aggiornamento è verificata solo sulla VISTA più esterna
- SI SI SPECIFICA **CASCADED**
  - la correttezza dell'aggiornamento è verificata su tutte le VISTE coinvolte dall'aggiornamento
  - OPZIONE DI DEFAULT

ESEMPIO 2 → LIBRO APPUNTI (LIBRO 1° → GESTIONE VISITE → PG 16-17)

GESTIONE PRIVATEZZA

le VISTE permettono di INDIVIDUARE SOTTOINSIEMI DI DATI → INDIVIDUATI DA 1 ESPRESSIONE SELECT

∑ ASSEGNANDO a 1 UTENTE l'accesso a SPECIFICHE VISTE SI LIMITANO
- la sua VISIBILITÀ su TABELLE ESISTENTI
- le OPERAZIONI che può ESEGUIRE

GESTIONE TRANSAZIONI

ESEMPIO APPLICATIVO

OPERAZIONI BANCARIE → operazioni **prelievo**
→ operazioni **versamento**

prelievo: operazioni svolte: specifica importo, verifica disponibilità ecc...

versamento: operazioni svolte: verifica importo, memorizza movimento ecc...

Le operazioni devono essere tutte svolte correttamente altrimenti prelievo/versamento non va a buon fine

- la BDD BANCARIA è un AMBIENTE MULTI UTENTE → diversi operatori
- la GESTIONE corretta delle INFO richiede
  - ↳ meccanismi per la gestione dell'accesso concorrente alla BDD
  - ↳ meccanismi per il ripristino dello STATO corretto della BDD in caso di guasti

---

GESTIONE TRANSAZIONI

- Necessaria quando più utenti possono accedere contemporaneamente ai DATI
- Offre meccanismi efficienti per:
  - Gestire l'accesso concorrente ai DATI
  - Effettuare il recovery a seguito di un malfunzionamento

TRANSAZIONE ⇨ sequenza di operazioni che:
- rappresenta un'UNITÀ elementare di lavoro
- può concludersi con successo o insuccesso

SISTEMA TRANSAZIONALE:

→ Sistema che mette a disposizione un meccanismo per la definizione e l'esecuzione di transazioni viene detto: SISTEMA TRANSAZIONALE

→ I DBMS contengono blocchi architetturali che offrono servizi di gestione delle transazioni

---

TRANSAZIONI IN SQL

UNA TRANSAZIONE È:
- un'UNITÀ logica di lavoro, non ulteriormente scomponibile
- una sequenza di operazioni (istruzioni SQL) di modifica dei DATI, che porta la BDD da 1 STATO consistente ad un altro STATO consistente

INIZIO TRANSAZIONE:

Per definire l'inizio di 1 transazione, il linguaggio SQL prevede l'istruzione **START TRANSACTION**

↳ di solito viene omessa

- l'inizio è implicito

→ quando si verifica la violazione di 1 vincolo il sistema interviene
- Per annullare la transazione
- oppure, per modificare lo stato della BDD eliminando la violazione del vincolo

ISOLAMENTO → eseguita come se fosse unica
→ d'esecuzione di 1 transazione indipendente dalla contemporanea esecuzione di altre transazioni
→ Gli effetti di 1 transazione non sono visibili dalle altre transazioni finché esso non è terminato
- si evita visibilità di stati intermedi non stabili
  - ↳ uno stato intermedio può essere annullato da un ROLLBACK successivo
    - ↳ in caso di ROLLBACK è necessario effettuare ROLLBACK delle altre transazioni che hanno osservato lo stadio intermedio (Effetto domino)

PERSISTENZA → rende permanenti le modifiche
→ L'effetto di una transazione che ha effettuato il commit è memorizzato in modo permanente
- Le modifiche dei dati eseguite da 1 transazione terminata con successo sono permanenti dopo il COMMIT

→ Garantisce affidabilità delle operazioni di modifica dei dati
- I DBMS offrono meccanismi di ripristino dello stato corretto della BDD dopo che si è verificato 1 guasto

APPLICAZIONI E SQL

→ Per risolvere problemi reali non è quasi mai sufficiente eseguire singole istruzioni SQL

→ Servono applicazioni per:

- Acquisire e gestire i dati forniti in ingresso
  - scelte dell'utente, parametri
- Gestire la logica applicativa
  - Flusso di operazioni da eseguire
- Restituire i risultati all'utente in formati diversi
  - Rappresentazione non relazionale dei dati
  - Visualizzazione complessa delle info

Integrazione tra SQL e Applicazioni

→ Le applicazioni sono scritte in linguaggio di programmazione tradizionale ad alto livello (C, C++, Java, C#)

↳ Linguaggio denominato: **linguaggio ospite**

→ Le istruzioni SQL sono usate nelle applicazioni per accedere alla BDD

- Interrogazioni
- Aggiornamenti

→ È necessario integrare il linguaggio SQL e i linguaggi di programmazione

- SQL → linguaggio dichiarativo
- Linguaggi di programmazione → Tipicamente procedurali

Conflitto di impedenza

⇨ Conflitto di impedenza

- Le interrogazioni SQL operano su una o più tabelle → risultato è tabella
  - Approccio **set oriented**
- I linguaggi di programmazione accedono alle righe di una tabella leggendole una a una
  ↳ Approccio **tuple oriented**

⇨ Soluzioni possibili per risolvere il conflitto

- Uso cursori
- Uso linguaggi che dispongono in modo naturale di strutture di tipo "insieme di righe"

ESEMPIO 2

→ Visualizza nome e n° soci dei fornitori di Torino

```
select NSoci, NomeF
from   F
where  Sede = 'Torino';
```

→ Restituisce un insieme di tuple

→ Necessario definire un **cursore** per leggere separatamente le tuple del risultato

→ Definizione cursore mediante la sintassi del linguaggio PL/SQL di Oracle

```
CURSOR FornitoriTorino IS
SELECT NomeF, NSoci
FROM   F
WHERE  Sede = 'Torino';
```

IL CURSORE

- Permette di leggere singolarmente le tuple che fanno parte del risultato di un'interrogazione
- Ogni interrogazione SQL che può restituire un insieme di tuple deve essere associata a un cursore

Non necessitano di cursori:

- Interrogazioni SQL con restituzione di max 1 tupla
- Selezioni sulla chiave primaria
- operazioni di aggregazione senza clausola GROUP BY
- I comandi di aggiornamento e DDL → non generano tuple come risultato

AGGIORNABILITÀ

→ È possibile aggiornare o cancellare la tupla corrente puntata dal cursore

↳ Più efficiente rispetto all'esecuzione di 1 istruzione SQL separata di aggiornamento

→ L'aggiornamento di una tupla tramite cursore è possibile solo se è aggiornabile la vista che corrisponde all'interrogazione associata

↳ Deve esistere una corrispondenza 1 a 1 tra le tuple puntate dal cursore e la tupla da aggiornare nella tabella della BDD

ESEMPIO CURSORE NON AGGIORNABILE

Si supponga l'esistenza del cursore DATIFORNITORI associato all'interrogazione

```
select DISTINCT CodF, NomeF, NSoci
from   F, P, FP
where  F.CodF = FP.CodF AND P.CodP = FP.CodP AND Colore = 'Rosso';
```

→ Il cursore DATI Fornitore non è aggiornabile

→ Se la STESSA INTERROGAZIONE Dinamica DEVE essere eseguita più VOLTE nella STESSA SESSIONE DI LAVORO.

- E' POSSIBILE RIDURRE I TEMPI D'ESECUZIONE
- SI EFFETTUA UNA SOLA VOLTA la COMPILAZIONE e la scelta DEL PIANO D'ESECUZIONE
- SI ESEGUE L'INTERROGAZIONE più VOLTE

EMBEDDED SQL

JDBC: INTERAZIONE CON DBMS

Caricamento del driver specifico per il DBMS utilizzato

Creazione di 1 connessione

Esecuzione istruzioni SQL

- Creazione di 1 STATEMENT
- Richiesta esecuzione istruzione
- elaborazione risultato in caso di interrogazioni

- Chiusura statement
- chiusura connessione

CARICAMENTO DEL DBMS DRIVER

→ Il driver è specificato per il DBMS utilizzato
→ Il caricamento avviene tramite l'instanziazione dinamica della classe associata al driver

OBJECT Class.forName(String /nomeDriver)

↓ contiene nome classe da instanziare

→ è la 1ª operazione da effettuare
→ non è necessario conoscere in fase di compilazione del codice quale DBMS è usato
  a la lettura del nome del driver può avvenire a runtime da 1 file di configurazione

CREAZIONE DI 1 CONNESSIONE

invocazione del metodo del getConnection della classe DriverManager

CONNECTION DriverManager.getConnection(STRING url, STRING user, STRING password)

- url → contiene info necessarie per identificare il DBMS a cui si vuole collegare
  → Formato legato al driver utilizzato
- USER E PASSWORD → credenziali di autenticazione

ESECUZIONE ISTRUZIONI SQL

l'esecuzione d'una istruzione SQL richiede l'uso di una interfaccia specifica → denominata **STATEMENT**

↳ ogni oggetto statement

→ è associato a 1 connessione
→ è creato mediante il metodo CREATE STATEMENT della classe CONNECTION

Statement Create Statement()

b) ESECUZIONE (PREPARATA dell'INTERROGAZIONE)

→ Utile quando si deve eseguire la stessa ISTRUZIONE SQL, più volte nella stessa sessione di lavoro → RIDUCE TEMPO ESECUZIONE → COMPILATA 1 SOLA VOLTA

→ L'istruzione SQL è compilata una volta sola e il suo piano d'esecuzione è memorizzato nel DBMS

↳ è inoltre eseguita più volte nella stessa sessione

perciò:

→ È COMPILATA 1 SOLA VOLTA → All'inizio dell'esecuzione dell'applicazione

→ ESEGUITA PIÙ VOLTE → prima di ogni esecuzione è necessario specificare il valore corrente dei parametri

PREPARAZIONE dello STATEMENT

- Si utilizza un oggetto del tipo **PREPARED STATEMENT**
  - ↳ creato con il metodo `preparedStatement (String ISTRUZIONE SQL)`
    - contiene comando SQL da eseguire
    - dove si vuole specificare la presenza di un parametro è presente il simbolo ?

ESEMPIO

```
PreparedStatement pstmt;
pstmt = conn.prepareStatement
("SELECT CodF, NSoci
FROM F WHERE Sede = '?' ");
```

IMPOSTAZIONE PARAMETRI

→ SOSTITUZIONE SIMBOLI ? per l'esecuzione corrente
→ Si EVOCA su un oggetto PreparedStatment uno dei seguenti metodi

- `void setInt (int numeroParametro, int valore)`
- `void setString (int numeroParametro, String valore)`

numeroParametro → INDICA LA POSIZIONE DEL PARAMETRO DA ASSEGNARE
- Possono esserci più parametri nella stessa istruzione SQL
- Il 1° parametro è associato al n° uno

valore → INDICA IL VALORE DA ASSEGNARE AL PARAMETRO

RESULT SET AGGIORNABILE

→ È possibile creare un **Result Set Aggiornabile**

- L'esecuzione di aggiornamenti della BDD è più efficiente
- È simile a 1 cursore aggiornabile
  - Necessaria esistenza di corrispondenza 1 a 1 tra tuple del risultato e tuple della tabelle presenti nel DBMS

DEFINIZIONE DI TRANSAZIONE

→ Le connessioni avvengono implicitamente in modalità **AUTOCOMMIT MODE**

- Dopo l'esecuzione con successo di ogni istruzione SQL, è eseguito automaticamente **COMMIT**

→ Quando è necessario eseguire COMMIT solo dopo aver eseguito con successo 1 seq. di istruzioni SQL

- Si esegue 1 solo **COMMIT** alla fine dell'esecuzione di tutte le istruzioni
- Il **COMMIT** deve essere gestito in modo non automatico

GESTIONE TRANSAZIONI

→ Gestione della modalità di **COMMIT** invocando il metodo **setAutoCommit()** sulla connessione

VOID **setAutoCommit** (Boolean autoCommit)

- TRUE: se si vuole abilitare l'autocommit (default)
- FALSE: se si vuole disabilitare l'autocommit

SE SI DISABILITA L'AUTOCOMMIT

- le operazioni di **Commit** e **Rollback** devono essere chieste esplicitamente
  - COMMIT → VOID COMMIT()
  - ROLLBACK → VOID ROLLBACK()
- I metodi sono invocati sulla connessione interessata

CONTROLLO ACCESSO

SICUREZZA DATI

→ PROTEZIONE DEI DATI DA

- LETTURE NON AUTORIZZATE
- ALTERAZIONE & DISTRUZIONE

→ IL DBMS FORNISCE STRUMENTI per realizzare le protezioni, che sono definite dall'amministratore della BASE DI DATI (DBA)

→ IL CONTROLLO DELLA SICUREZZA VERIFICA CHE GLI UTENTI SIANO AUTORIZZATI a eseguire le operazioni

→ la SICUREZZA È GARANTITA MEDIANTE 1 INSIEME DI VINCOLI

- SPECIFICATI DAL DBA IN 1 OPPORTUNO LINGUAGGIO
- MEMORIZZATI NEL DIZIONARIO DI DATI DEL SISTEMA

RISORSE E PRIVILEGI

→ Qualsiasi componente dello schema di una BDD è una risorsa

- TABELLA
- VISTA
- ATTRIBUTO ALL'INTERNO DI 1 TABELLA O VISTA
- DOMINIO
- PROCEDURA
- ...

→ le risorse sono protette mediante la definizione di **privilegi d'accesso**

privilegi d'accesso:

⇨ DESCRIVONO I DIRITTI D'ACCESSO ALLE RISORSE DEL SISTEMA

⇨ SQL offre meccanismi di controllo dell'accesso flessibili, mediante i quali è possibile specificare

- le risorse a cui possono accedere gli utenti
- le risorse che devono essere mantenute private

CARATTERISTICHE DEI PRIVILEGI

→ Ogni privilegio è caratterizzato dalle seguenti INFO

- Risorsa a cui si riferisce
- TIPO DI PRIVILEGIO
- UTENTE che concede il privilegio
- " RICEVE " "
- FACOLTÀ DI TRASMETTERE IL PRIVILEGIO A altri UTENTI

GRANT

GRANT Elenco Privilegi ON nome risorsa TO elenco utenti [WITH GRANT OPTION]

- specifica l'elenco dei privilegi
  - ALL PRIVILEGES
    - Parola chiave per identificare tutti i privilegi
- Specifica la risorsa sulla quale si vuole concedere il privilegio
- Specifica gli utenti a cui viene concesso il privilegio
  - Se vogliamo dare il privilegio a tutti quanti esiste TO PUBLIC
- FACOLTÀ di trasferire il privilegio ad altri utenti

ESEMPIO 1

GRANT All PRIVILEGES ON P TO Bianchi e Neri

→ A gli utenti Bianchi e Neri sono concessi tutti i privilegi sulla tabella P

ESEMPIO 2

GRANT Select on F TO Rossi WITH GRANT OPTION

→ All'utente Rossi è concesso il privilegio di select sulla tabella F
→ Rossi può anche trasferire il privilegio ad altri

REVOKE

REVOKE Elenco Privilegi ON nome risorsa FROM elenco utenti [RESTRICT | CASCADE]

Il comando REVOKE può TOGLIERE

- Tutti i Privilegi che erano stati concessi
- Un sottoinsieme dei privilegi concessi

- RESTRICT
  - Il comando non deve essere eseguito qualora la revoca dei privilegi all'utente comporti qualche altra revoca di privilegi
  - VALORE DI DEFAULT
- CASCADE
  - Revoca anche tutti i privilegi che erano stati propagati
  - Per ogni privilegio revocato sono revocati a cascata tutti i privilegi concessi, tranne tutti gli elementi della BDD che erano stati creati sfruttando questi privilegi.

ESEMPIO 1

REVOKE UPDATE ON P FROM Bianchi

→ A Bianchi è TOLTO il privilegio di UPDATE di P

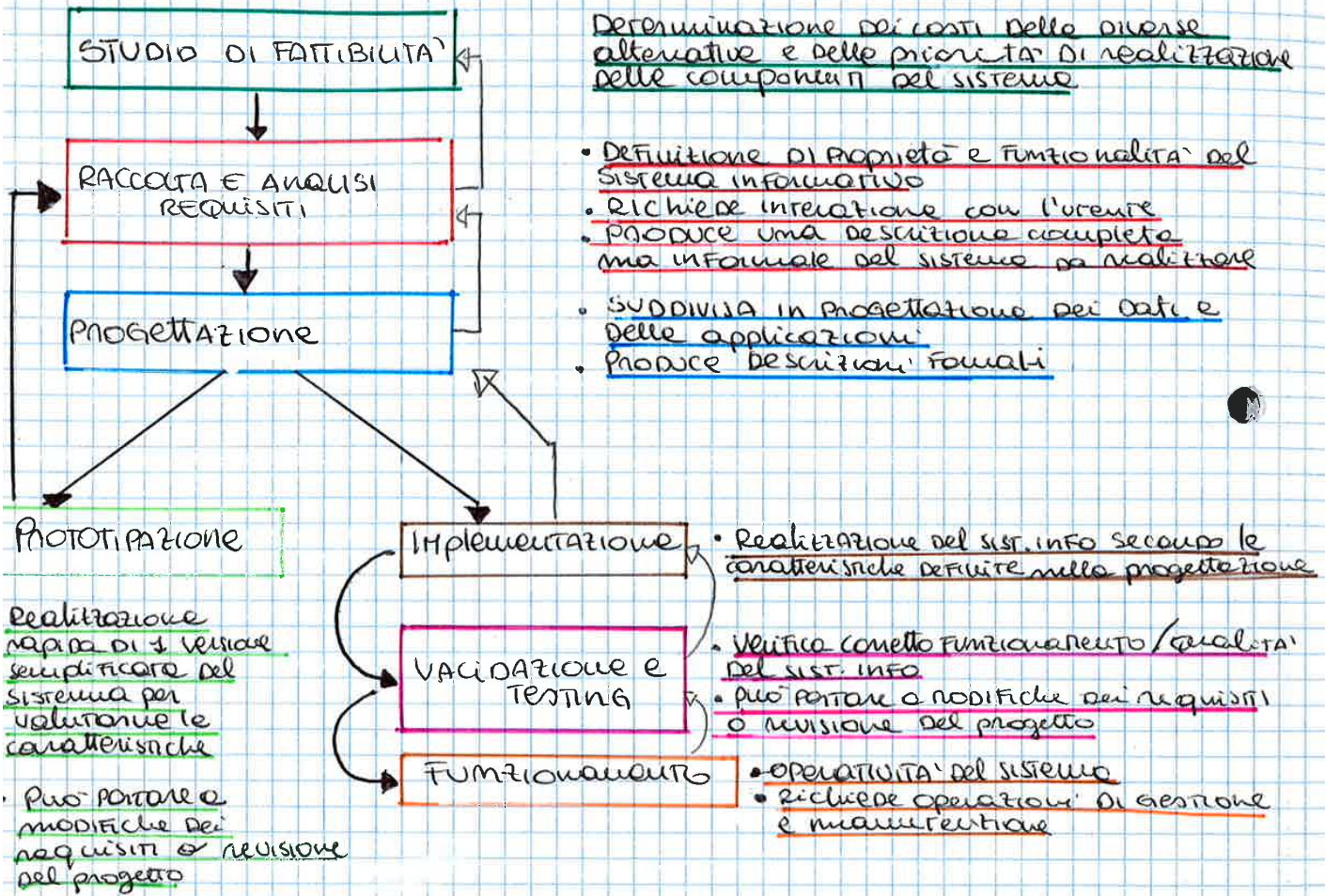
→ Il comando non è eseguibile se comporta la revoca del privilegio ad altri utenti

MODELLO ENTITÀ - RELAZIONE

Ciclo di vita di un sistema informativo

La progettazione di una BDD è una delle attività del processo di sviluppo di un sistema informativo

↳ va inquadrata come "ciclo di vita del sist. informativo"



Determinazione dei costi delle diverse alternative e delle priorità di realizzazione delle componenti del sistema

- Definizione di proprietà e funzionalità del sistema informativo
- Richiede interazione con l'utente
- Produce una descrizione completa ma informale del sistema da realizzare
- Suddivisa in progettazione dei dati e delle applicazioni
- Produce descrizioni formali

Realizzazione rapida di 1 versione semplificata del sistema per valutarne le caratteristiche

- Può portare a modifiche dei requisiti o revisione del progetto

- Realizzazione del sist. info secondo le caratteristiche definite nella progettazione

- Verifica corretto funzionamento/qualità del sist. info
- Può portare a modifiche dei requisiti o revisione del progetto

- Operatività del sistema
- Richiede operazioni di gestione e manutenzione

PROGETTAZIONI DI BDD

- La BDD costituisce un componente importante del sistema complessivo
- La metodologia di progettazione consiste in:
  1. Decomposizione dell'attività di progetto in passi successivi indipendenti tra loro
  2. Strategie da seguire nei vari passi e criteri per la scelta delle strategie
  3. Modelli di riferimento per descrivere i dati in ingresso e in uscita dalle varie fasi

ESEMPIO: Preparazione atletica

1. Dati in ingresso: peso attuale, % di grasso corporeo
   Dati in uscita: modello della struttura corporea della persona in forma
2. Dati in ingresso: modello persona in forma
   Dati in uscita: = struttura corporea dell'atleta medio

COSTRUTTI PRINCIPALI DEL MODELLO E-R

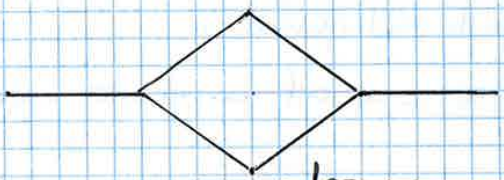
ENTITÀ
RELAZIONI
ATTRIBUTI
IDENTIFICATORI
GENERALIZZAZIONI E SOTTOINSIEMI

ENTITÀ

NOME ENTITÀ

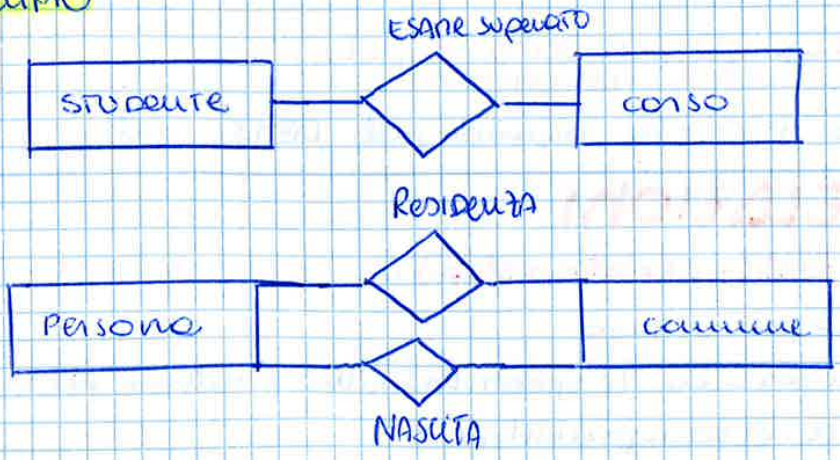
- rappresenta classi di oggetti del mondo reale che hanno
  - proprietà comuni
  - esistenza autonoma
- un occorrenza di 1 ENTITÀ è un oggetto della classe che l'entità rappresenta

RELAZIONE



- rappresenta un legame logico tra 2 o più entità
- DA non confondere con la relazione del modello Relazionale

ESEMPIO



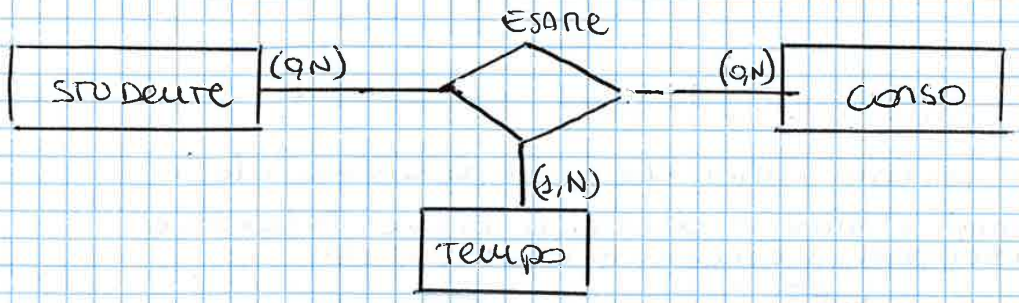
LIMITE DI UNA RELAZIONE BINARIA

→ non è possibile che 1 STUDENTE sostenga 2 volte lo stesso esame

→ ciò è possibile solo in TEMPI DIVERSI.

es: S_1 C_1 $T_{(1)}$ } ≠ TEMPO DIVERSO
S_1 C_1 $T_{(2)}$

Relazione TERNARIA:

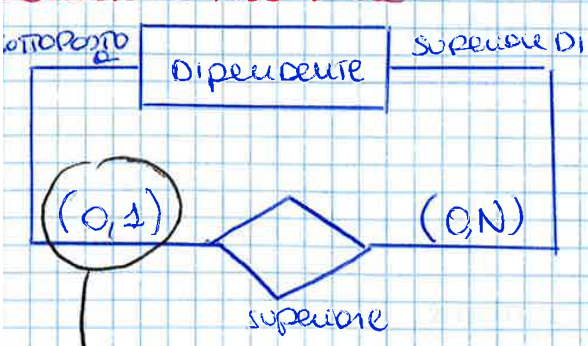


→ la CARDINALITÀ MINIMA RARAMENTE SONO 1 per TUTTE le ENTITÀ COINVOLTE IN 1 RELAZIONE

→ la CARDINALITÀ MASSIMA SONO QUASI SEMPRE N

↳ se la PARTECIPAZIONE DI 1 ENTITÀ E ha CARDINALITÀ MAX 1 è POSSIBILE ELIMINARE la RELAZIONE n-aria e LEGARE l'ENTITÀ E con le ALTRE MEDIANTE RELAZIONI BINARIE

Relazione RICORSIVA



⇨ RELAZIONE DI 1 ENTITÀ CON SE STESSA

⇨ se la RELAZIONE non è SIMMETRICA, OCCORRE DEFINIRE 2 RUOLI DELL'ENTITÀ

↳ UN SOTTOPOSTO POTREBBE AVERE PIÙ SUPERIORI (0,N)

IDENTIFICATORI

→ Specificati per OGNI ENTITÀ

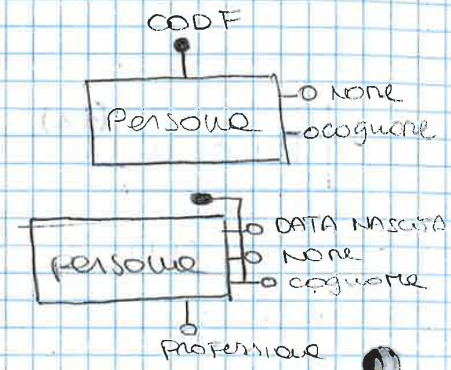
→ Descrive i concetti dello schema che permettono di individuare in modo univoco le occorrenze delle entità

- OGNI ENTITÀ DEVE AVERE ALMENO 1 IDENTIFICATORE
- Può ESISTERE PIÙ DI 1 IDENTIFICATORE Appropriato per un ENTITÀ

IDENTIFICATORE INTERNO:

SEMPLICE: COSTITUITO DA 1 solo attributo

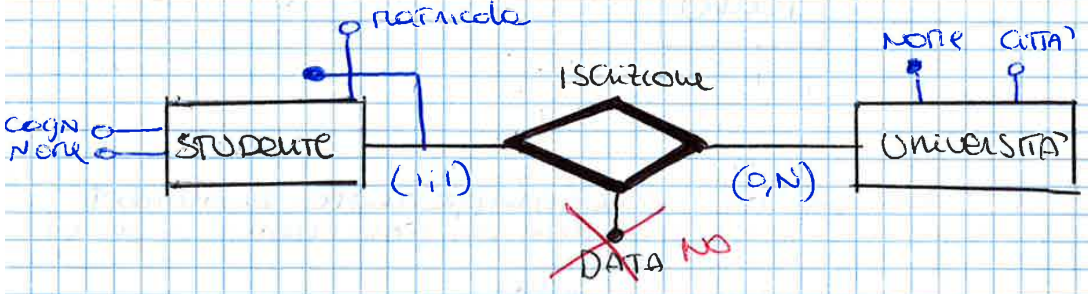
COMPOSTO: " " = più attributi



IDENTIFICATORE ESTERNO:

→ L'entità che non dispone internamente di attributi sufficienti per definire 1 identificatore è denominata ENTITÀ Debole

→ L'entità debole deve partecipare con cardinalità (1,1) in ognuna delle relazioni che forniscono parte dell'identificatore



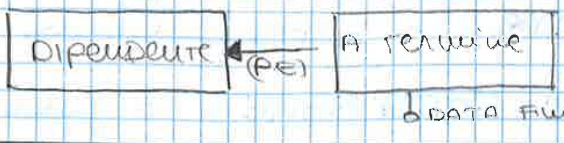
Inoltre: UN IDENTIFICATORE ESTERNO può coinvolgere 1 entità o sua volta IDENTIFICATA esternamente

↳ non si devono generare cicli di IDENTIFICAZIONE

J.B. le relazioni NON hanno IDENTIFICATORI!

NOTA BENE: SE SI GENERA 1 SOLA ENTITÀ FIGLIA, È SEMPRE PARZIALE e ESCLUSIVA

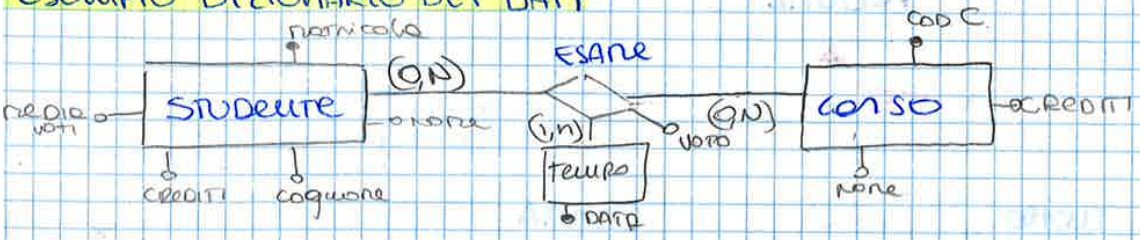
ESEMPIO



DOCUMENTAZIONE DI SCHEMI E-R

DIZIONARIO DEI DATI → permette di ARRICCHIRE lo schema E-R con descrizione in linguaggio naturale di entità, relazioni e attributi

ESEMPIO DIZIONARIO DEI DATI



DIZIONARIO:

ENTITÀ	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORE
STUDENTE	STUD UNIVERSITÀ	MATRICOLA, NOME, COGNOME, CDD, MEDIA	MATRICOLA
CORSO	VARI CORSI	NOME, CREDITI, COD C	COD C
TEMPO	DATE ESAMI	DATA	DATA

RELAZIONE	DESCRIZIONE	ENTITÀ COINVOLTE	ATTRIBUTI
ESAME	ASSOCIA LO STUDENTE AI SUOI ESAMI E SALVA I VOTI	STUDENTE (0,N) CORSO (0,N) TEMPO (1,N)	VOTO

VINCOLI D'INTEGRITÀ SUI DATI → NON SEMPRE possono essere INDICATI esplicitamente in 1 schema E-R
→ Possono essere descritti in LINGUAGGIO naturale

ESEMPIO

VINCOLI D'INTEGRITÀ	
R_1	Il VOTO di un esame può assumere esclusivamente valorine tra 1 e 30
R_2	OGNI STUDENTE non può superare 2 volte con esito positivo lo stesso esame

ESEMPIO PROGETTAZIONE

Si vuole rappresentare una base dati per la gestione di un sistema di prenotazioni di esami medici all'interno di una Azienda Sanitaria Locale (ASL), tenendo conto delle informazioni seguenti. Ciascun paziente è caratterizzato da numero della tessera sanitaria, nome, cognome, indirizzo, data di nascita, luogo di nascita e età. Gli ospedali della ASL sono caratterizzati da un codice numerico, da un nome e un indirizzo. Ogni ospedale è suddiviso in reparti identificati da un codice numerico univoco all'interno dell'ospedale di appartenenza e caratterizzati dal nome del reparto e numero di telefono. Il personale del reparto è identificato attraverso il codice fiscale. Sono noti inoltre il nome, il cognome e l'indirizzo di domicilio. Tra il personale, nel caso dei medici del reparto è noto l'elenco delle specializzazioni conseguite, mentre per il personale volontario è noto il nome dell'associazione di appartenenza, se disponibile. Gli esami medici che possono essere eseguiti sono caratterizzati da un codice numerico e da una descrizione testuale (ad esempio radiografia, ecc.) Nel caso di esami specialistici si memorizzano inoltre il medico che effettua la visita e la descrizione della dieta da seguire (se necessaria). I laboratori che eseguono gli esami sono identificati da un codice univoco all'interno di un ospedale della ASL e sono caratterizzati dal nome del laboratorio, dal piano di ubicazione e dal numero di stanza. Per ogni componente del personale di laboratorio si memorizzano le giornate e i laboratori in cui presta servizio. Si tenga presente che nel corso della stessa giornata ogni componente del personale può prestare servizio presso più laboratori.

Per effettuare un esame è necessario eseguire una prenotazione. Per ogni prenotazione di un esame da parte di un paziente si vuole memorizzare la data e l'ora dell'esame, il laboratorio presso cui è eseguito, il costo del ticket e se tale esame è prescritto con urgenza. Si tenga presente che ogni paziente può effettuare più prenotazioni dello stesso esame in date diverse. Si noti inoltre che lo stesso esame non può essere ripetuto nello stesso giorno dallo stesso paziente, neppure in laboratori divers. Ogni medico può assumere ruoli diversi nel corso della sua carriera (ad esempio assistente, primario, ecc.). Si vuole tenere traccia dei ruoli assunti da ogni medico nel corso di tutta la sua carriera e dei periodi di tempo in cui ha assunto tali ruoli (data di inizio, data di fine). Si tenga presente che ogni medico non può assumere contemporaneamente più ruoli, mentre può assumere lo stesso ruolo in periodi di tempo diversi.

○ OGGETTI COMPLESSI ● ENTITÀ FIGLIE ● TEMPO

PROGETTAZIONE LOGICA RELAZIONALE

PASSI DELLA PROGETTAZIONE LOGICA

SCHEMA E-R
↓
RISTRUTTURAZIONE SCHEMA
↓
SCHEMA E-R SEMPLIFICATO
↓
TRADUZIONE
↓
SCHEMA LOGICO RELAZIONALE

RISTRUTTURAZIONE SCHEMA E-R

→ TIENE CONTO DI ASPETTI REALIZZATIVI → non è più uno schema concettuale

→ OBIETTIVI
- ELIMINAZIONE costrutti per cui non esiste una rappresentazione diretta nel modello relazionale
- TRASFORMAZIONI volte ad aumentare l'EFFICIENZA delle operazioni di accesso ai dati

ATTIVITÀ DELLA RISTRUTTURAZIONE

ANALISI RIDONDANZE

Eliminazione GENERALIZZAZIONI

PARTIZIONAMENTO E ACCORPAMENTO ENTITÀ e relazioni

Scelta IDENTIFICATORI PRINCIPALI

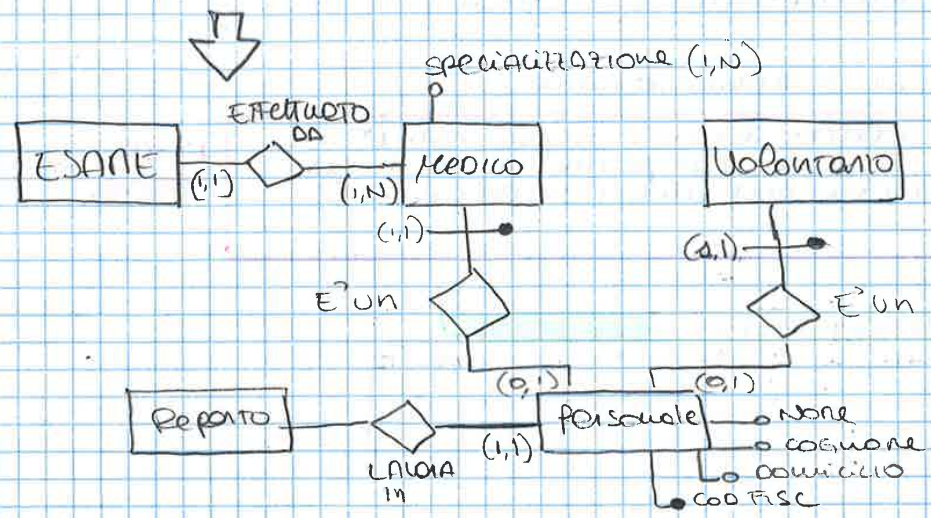
Analisi Ridondanze

RAPPRESENTANO INFO SIGNIFICATIVE MA DERIVABILI DA ALTRI CONCETTI

EFFETTI delle RIDONDANZE sullo SCHEMA LOGICO
- SEMPLIFICAZIONE e velocizzazione delle INTERROGAZIONI
- MAGGIOR COMPLESSITÀ e RALLENTAMENTO AGGIORNAMENTI
- MAGGIOR OCCUPAZIONE DI SPAZIO

ESEMPIO ⇒ SOSTITUZIONE CON RELAZIONI

SCHEMA PARTENZA VEDI ESEMPIO 1 e 2



↳ Soluzione più generale e sempre Applicabile

VALUTAZIONE ALTERNATIVE

ACCORPAMENTO NEL PADRE: è appropriato quando

- LE ENTITÀ FIGLIE INTRODUCONO DIFFERENZIAZIONI NON ESSENZIALI
- LE OPERAZIONI DI ACCESSO NON DISTINGUONO TRA OCCORRENZE DI ENTITÀ PADRE E FIGLIE

ACCORPAMENTO NELLE FIGLIE:

- LA GENERALIZZAZIONE È TOTALE
- LE OPERAZIONI DI ACCESSO DISTINGUONO TRA OCCORRENZE DELLE DIVERSE ENTITÀ FIGLIE

SOLUZIONI "MISTE":

- LE OPERAZIONI D'ACCESSO DISTINGUONO TRA OCCORRENZE DI ALCUNE ENTITÀ FIGLIE

→ Per le generalizzazioni a + livelli si procede nello stesso modo partendo dal livello inferiore

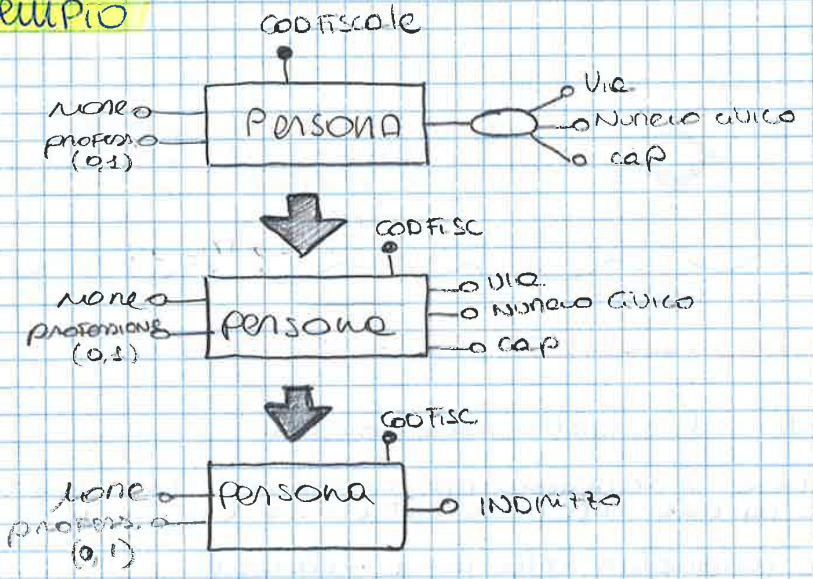
ELIMINAZIONE ATTRIBUTI COMPOSTI E SCELTA DEGLI IDENTIFICATORI PRIMARI

→ Non rappresentabili nel modello relazionale

→ Due alternative

- Si rappresentano in modo separato gli attributi componenti
  - ↳ adatto se necessario accedere separatamente a ciascun attributo
- Si introduce un unico attributo che rappresenta la concatenazione degli attributi componenti
  - ↳ adatta se sufficiente l'accesso all'info complessiva

ESEMPIO



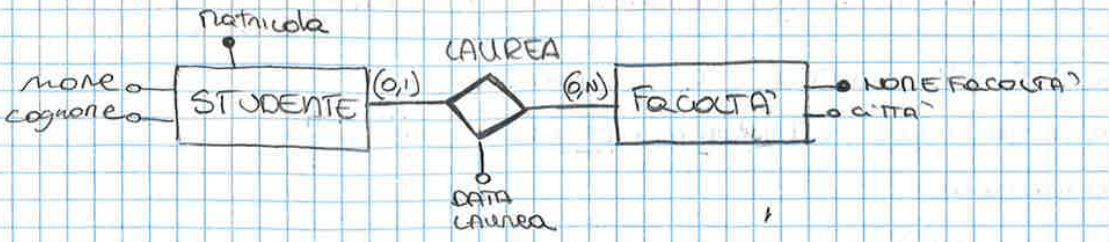
→ Necessario per definire la chiave primaria delle tabelle

→ Un BUON IDENTIFICATORE

- NON assume valore nullo
- è costituito da pochi attributi
- possibilmente è intero
- utilizzato da molte operazioni d'accesso

→ può essere opportuno riprodurre codici identificativi

ESEMPIO → RELAZIONE BINARIA 1 a MOLTI



Soluzione A

STUDENTE (matricola, nome, cognome)
FACOLTÀ (nomefacoltà, città)
LAUREA (matricola, nome facoltà, DATA LAUREA)

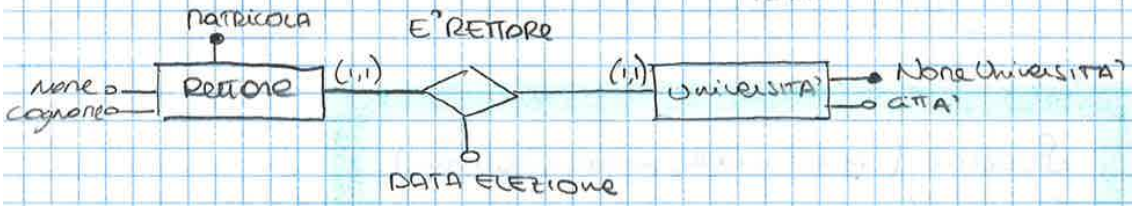
soluzione B

STUDENTE (matricola, nome, cognome, nome facoltà*, DATA LAUREA)
FACOLTÀ (nomefacoltà, città)

TRADUZIONE DEL MODELLO RELAZIONALE: UNO A UNO

- POSSIBILI PIÙ TRADUZIONI
  - DIPENDE DAL valore della CARDINALITÀ minima

ESEMPIO → RELAZIONE BINARIA 1 a 1



ALTERNATIVA 1:

Rettore (matricola, nome, cognome, nome università, DATA ELEZIONE)
UNIVERSITÀ (nome università, città)

ALTERNATIVA 2:

Rettore (matricola, nome, cognome)
UNIVERSITÀ (nome università, città, matricola, data elezione)

ALTERNATIVA 3:

RETTORE (matricola, nome, cognome)
UNIVERSITÀ (nome università, città)
E' RETTORE (matricola, nome università, DATA ELEZIONE)

PARTECIPAZIONE OPZIONALE DA ENTRAMBI I LATI, QUINDI POTEVA QUALCHE ISTANZA NON AVERE NOME UNIVERSITÀ la chiave primaria

oppure
RETTORE (matricola, nome, cognome)
UNIVERSITÀ (nome università, città, matricola*, DATA ELEZIONE*)

RIPRENDO L'ESEMPIO DI PROGETTAZIONE:

→ lo schema relazionale corretto sarà:

PAZIENTE (CodTes, Nome, Cognome, Indirizzo, LuogoNascita, DataNascita)

OSPEDALE (CodO, Nome, Indirizzo)

ESAME (CodE, Descrizione, DescrizioneDieta*, Tipo, CodFisc*)

PERSONALE (CodFisc, Nome, Cognome, Domicilio, Associazione*, Tipo, CodR, CodO)

LABORATORIO (CodLab, CodO, NomeLab, Piano, Stanza)

REPARTO (CodR, CodO, Nome, Telefono)

PRENOTAZIONE (CodTes, CodE, Data, Ora, Tiket, Urgenza, CodLab, CodO)

RuoloMedico (CodFisc, DataInizio, DataFine*, Ruolo)

HaSpecializzazione (CodFisc, Specializzazione)

DiServizioIn (CodFisc, CodLab, CodO, Data, OraInizio, Durata)

→ con vincoli d'integrità referenziale

ESAME (CodFisc) REFERENCES Personale (CodFisc)

Personale (CodR, CodO) REFERENCES Reparto (CodR, CodO)

Reparto (CodO) REFERENCES Ospedale (CodO)

Laboratorio (CodO) REFERENCES Ospedale (CodO)

Prenotazione (CodTes) REFERENCES Paziente (CodTes)

Prenotazione (CodE) REFERENCES Esame (CodE)

Prenotazione (CodLab, CodO) REFERENCES Laboratorio (CodLab, CodO)

RuoloDelMedico (CodFisc) REFERENCES Personale (CodFisc)

HaSpecializzazione (CodFisc) REFERENCES Personale (CodFisc)

DiServizioIn (CodFisc) REFERENCES Personale (CodFisc)

DiServizioIn (CodLab, CodO) REFERENCES Laboratorio (CodLab, CodO)

RIDONDANZA

- Un'unica relazione è utilizzata per rappresentare info eterogenee
  - Alcuni dati sono ripetute in tuple ≠ senza l'aggiunta di nuove info → dati ridondanti

Anomalie

- le info ridondanti devono essere aggiornate in modo atomico (tutte contemporaneamente)
- la cancellazione di una tupla comporta la cancellazione di tutti i concetti in essa rappresentati → inclusi quelli che potrebbero essere ancora validi
- l'inserimento di una nuova tupla è possibile solo se esiste almeno l'info completa relativa alla chiave primaria → non è possibile inserire la parte di tupla relativa ad un solo concetto

Forma normale di Boyce Codd

Dipendenza funzionale

→ È un tipo particolare di vincolo d'integrità

→ Descrive legami di tipo funzionale tra gli attributi di 1 relazione

Esempio: la residenza è unico per ogni studente

- ogni volta che compare lo stesso studente, il valore è ripetuto
- il valore di Matr Studente determina il valore di residenza

→ Una relazione soddisfa una dipendenza funzionale $X \to Y$ se, per ogni coppia t_1, t_2 di tuple di r aventi gli stessi valori per gli attributi in X, t_1 e t_2 hanno gli stessi valori anche per gli attributi in Y

- X determina Y (in R)

Esempi: Matr Studente → Residenza

Matr Studente CodCorso → Nome Corso

Dipendenza non banale

→ La dipendenza: Matr Studente → CodCorso è banale perché CodCorso fa parte di entrambi i lati

→ Una dipendenza funzionale $X \to Y$ è NON BANALE se nex attributo in X compare tra gli attributi in Y

Dipendenze funzionali e chiavi

→ Data una chiave K di una relazione r $K \to$ qualsiasi altro attributo di R

Esempi:

Matr Studente CodCorso → Residenza

Matr Studente CodCorso → Voto

→ Ovvero: Gli attributi comuni formano 1 chiave per almeno 1 delle relazioni decomposte

Esempio

R_1 (Impiegato, Stipendio) R_2 (Categoria, Stipendio)

∴ X_1 = Impiegato, Stipendio
X_2 = Categoria, Stipendio
X_0 = Stipendio

↳ non soddisfa la condizione di decomposizione snz perdita perché non è chiave primaria

→ Se fosse stato X_0 = Impiegato → soddisfa

Conservazione delle dipendenze

→ Una decomposizione conserva le dipendenze se ciascuna delle dipendenze funzionali dello schema originario è presente in 1 delle relazioni decomposte

→ È opportuno che le dipendenze siano conservate, in modo da garantire che nello schema decomposto siano soddisfatti gli stessi vincoli dello schema originario

Esempio

R(Impiegato, Categoria, Stipendio)

Decomposizione basata sulle dipendenze funzionali

Impiegato → Categoria
Categoria → Stipendio

↳ (Impiegato → Stipendio) implicito → non perde alcuna dipendenza

⚠ Decomposizione deve sempre soddisfare → Ricomposizione senza perdita
→ Conservazione dipendenze

PARADIGMA CLIENT-SERVER

→ CLIENT: UTENTE DEL SERVIZIO

→ SERVER: FORNITORE DEL SERVIZIO

→ CLIENTE E SERVER NON HANNO SENSO SENZA 1 PROTOCOLLO DI COMUNICAZIONE CHE:

- DEFINISCA LE POSSIBILI INTERAZIONI TRA CLIENT E SERVER
- SPECIFICHI I PATTI GLI A OGNI INTERAZIONE
- DEFINISCA LE CONDIZIONI DI ERRORE E LE AZIONI DA SVOLGERE IN CONSEGUENZA

ARCHITETTURA CLIENT/SERVER

→ 2 LIVELLI

- THICK CLIENT → CONTIENE DELLA LOGICA OPERATIVA
- DBMS SERVER → CONSENTE ACCESSO AI DATI

→ 3 LIVELLI

- THIN CLIENT → BROWSER
- APPLICATION SERVER → - IMPLEMENTA LA LOGICA APPLICATIVA
  - SOVENTE È ANCHE WEB SERVER
- DBMS SERVER → CONSENTE L'ACCESSO AI DATI

ESECUZIONE SQL

COMPILE AND GO

1. QUERY INVIATA A SERVER
2. QUERY COMPILATA → GENERAZIONE PIANO D'ESECUZIONE
3. QUERY ESEGUITA
4. RISULTATO RESTITUITO

EFFICACE PER ESECUZIONE DI QUERY NON RIPETITIVE

COMPILE AND STORE

1. QUERY INVIATA AL SERVER
2. QUERY COMPILATA → GENERAZIONE PIANO D'ESECUZIONE, MEMORIZZAZIONE " " PER USO FUTURO
3. QUERY ESEGUITA
4. RISULTATO RESTITUITO

→ EFFICACE PER QUERY RIPETUTE