Introduzione

lunedì 24 settembre 2018 09.46

Un **sistema informativo** (**SI**) è un componente di una organizzazione il cui scopo è gestire le informazioni utili per gli scopi dell'organizzazione stessa dove con gestire si intende acquisire, elaborare, conservare, produrre e distribuire i dati.

Un SI gestisce informazioni, ma ciò non implica che faccia ricorso a strumenti automatici. La parte automatizzata di un SI viene più propriamente denominata **sistema informatico**.

DBMS

lunedì 24 settembre 2018 09.51

Il modo più comune con cui un sistema informatico gestisce le informazioni è attraverso la rappresentazione codificata dei dati tecnicamente detta **Base di dati** (DB).
Un DB viene gestito da un **DBMS** (Data Base Management System).

Un DBMS è in grado di:

- Gestire collezioni di dati condivise da più applicazioni e utenti
- Garantire la persistenza anche a fronte di guasti
- Garantire elevate prestazioni anche con grandi quantità di dati
- Indipendenza fisica
- Indipendenza logica
- Preservare l'integrità dei dati

Nel corso si tratteranno i **RDBMS** ovvero quei sistemi che supportano il **modello relazionale** dei dati (rappresentazione in forma tabellare).
In questo tipo di rappresentazione dei dati si distinguono:

- Lo *schema*, che descrive la struttura dei dati (**parte intensionale**)
- L'*istanza*, l'informazione vera e propria (**parte estensionale**)

Un DBMS inoltre mette a disposizione diversi linguaggi per interagire con la base di dati come:

- **DDL** (Data Definition Language)
  Per definire gli schemi e la struttura
- **DML** (Data Manipulation Language)
  Per interrogare e modificare le istanze del DB
- **DCL** (Data Control Language)
  Comandi di vario tipo come ad es. il controllo degli accessi

Il Modello Relazionale

lunedì 24 settembre 2018 16.51

La principale differenza tra il modello relazionale (1970) e i preesistenti gerarchico e reticolare sta nel modo con cui si rappresentano i legami tra diverse strutture: mentre gerarchico e reticolare lavorano con i puntatori, nel modello relazionale si fa esclusivamente uso dei valori.

Relazione matematica
Una relazione su $D_1,D_2,\dots,D_n$ è una qualunque sottoinsieme del prodotto cartesiano $D_1\times D_2\times\dots\times D_n$.
Una relazione è un insieme di n-uple ordinate ($D_1\times D_2 \neq D_2\times D_1$) su domini non necessariamente distinti.

Relazione nel modello relazionale
Ad ogni occorrenza di dominio si associa un nome univoco nella relazione, detto **attributo**.

Studenti

Matricola	Cognome	Nome	DataNascita	Email
29323	Bianchi	Giorgio	21/06/1978	gbianchi@alma.unibo.it
35467	Rossi	Anna	13/04/1978	anna.rossi@yahoo.it
39654	Verdi	Marco	20/09/1979	mverdi@mv.com
42132	Neri	Lucia	15/02/1978	lucia78@cs.ucsd.edu

Lo schema di un DB relazionale è un insieme di schemi di relazioni con nomi distinti:
$R = \{R_1(X_1), R_2(X_2), \dots, R_m(X_m)\}$
L'istanza di un DB con schema R è un insieme di istranze di relazioni:
$r = \{r_1, r_2, \dots, r_m\}$

Chiavi e superchiavi
Dato uno schema R(X), un insieme di attributi $K\subseteq X$ è

- **superchiave** se e solo se
  in ogni istanza ammissibile r di R(X) non esistono due tupe t_1 e t_2 tali che $t_1[K]=t_2[K]$.
  NB. {Matricola, Cognome} e {CodiceFiscale, Nome} sono superchiavi
- **chiave** se e solo se
  è una superchiave minimale, ovvero non possiamo ridurre ulteriormente il numero di attributi che costituiscono la superchiave.
  NB. {Matricola} e {CodiceFiscale} sono chiavi

Le chiavi sono lo strumento attraverso il quale vengono correlati i dati in relazioni diverse.

Matricola	CodiceFiscale	Cognome	Nome	DataNascita
NULL	NULL	Bianchi	Giorgio	21/06/1978
35467	RSSNNA78D13A	Rossi	Anna	13/04/1978
NULL	VRDMRC79I20A	Verdi	Marco	20/09/1979
42132	NULL	Verdi	Marco	20/09/1979

Nel caso in questione però:

- La prima tupla non è identificable in alcun modo
- La terza e la quarta non sappiamo se si riferiscano o meno allo stesso studente

Per evitare di incorrere in tali problemi è necessario scegliere una **chiave primaria**, dove cioè non si ammettono valori nulli (generalmente gli attributi della chiave primaria vengono sottolineati).
Nel caso in cui nessuna chiave possa garantire la disponibilità di valori è necessario introdurre un attributo che svolga la funzione di codice univco.

Chiave importata
Si considerino due schemi $R_1(X_1)$ e $R_2(X_2)$ e sia $Y\subseteq X_2$, l'insieme Y viene detto **foreign key** o chiave importata se l'insieme dei valori di Y in r_2 è un sottoinsieme dei valori della chiave primaria di $R_1(X_1)$ nell'istanza r_1.

I vincoli di chiave sono di tipo **intra-relazionale** in quanto interessano una relazione alla volta mentre i vincoli di foreign key sono di tipo **inter-relazionale** in quanto individuano correlazioni tra più relazioni.

Algebra Relazionale

venerdì 28 settembre 2018 14.18

Un linguaggio di manipolazione (**DML**) permette di interrogare e modificare istanze di DB.
Linguaggi formali:

- **Calcolo relazionale**
  linguaggio dichiarativo basato su logica dei predicati del primo ordine
- **Algebra relazionale**
  linguaggio procedurale di tipo algebrico i cui operandi sono relazioni

L'algebra relazionale è costituita da un insieme di operatori:

- Operatori di base unari: *selezione, proiezione, ridenominazione*
- Operatori di base binari: *join, unione, differenza*

Operatore di selezione σ
E' un operatore unario, permette di selezionare un sottoinsieme di tuple di una relazione applicando a ciascuna di esse una formula booleana.
$\sigma_F(R)$

schema	R(X)	X
istanza	r	$\sigma_F(r) = \{t \mid t \in r \text{ AND } F(t) = true\}$
	input	output

Operatore di proiezione π
E' un operatore unario, permette di selezionare un sottoinseieme Y degli attributi di una relazione.
$\pi_Y(R)$

schema	R(X)	Y
istanza	r	$\pi_Y(r) = \{t[Y] \mid t \in r\}$
	input	output

NB. La carinalità di $\pi_Y(r)$ è in generale minore o uguale a quella di r poichè la proiezione "elimina i duplicati". L'uguaglianza è garantita se e solo se Y è una superchiave di R(X).

Operatore di join naturale ⋈
E' un operatore binario, combina le tuple di due relazioni sulla base dell'uguaglianza dei valori degli attributi comuni alle due relazioni.
$R_1 \bowtie R_2$

schema	$R_1(X_1), R_2(X_2)$	X_1, X_2
istanza	r_1, r_2	$r_1 \bowtie r_2 = \{t \mid t[X_1] \in r_1 \text{ AND } t[X_2] \in r_2\}$
	input	output

$0 \leq |r1 \bowtie r2| \leq |r1| \times |r2|$

Operatore di unione U
E' un operatore binario.
$R_1 \cup R_2$

schema	$R_1(X), R_2(X)$	X
istanza	r_1, r_2	$r_1 \cup r_2 = \{t \mid t \in r_1 \text{ OR } t \in r_2\}$
	input	output

Operatore di differenza -
E' un operatore binario
$R_1 - R_2$

schema	$R_1(X), R_2(X)$	X
istanza	r_1, r_2	$r_1 - r_2 = \{t \mid t \in r_1 \text{ AND } t \notin r_2\}$
	input	output

Algebra Relazionale

martedì 2 ottobre 2018 12.59

Operatore di ridenominazione ρ

E' un operatore unario che modifica lo schema di una relazione, cambiando i nomi di uno o più attributi

$\rho_{Y \leftarrow X}(R)$

schema	$R_1(XZ)$	YZ
	input	output

Operatori derivati

Gli operatori sinora visti definiscono completamente l'AR. Tuttavia, per praticità, ètalvolta utile ricorrere ad altri operatori "derivati":

Operatore di divisione ÷

La divisione di r_1 per r_2 , con r_1 su $R_1(X_1X_2)$ e r_2 su $R_2(X_2)$, è il più grande insieme di tuple con schema X_1 tale che, facendo il prodotto Cartesiano con r_2, ciò che si ottiene è una relazione contenuta in r_1.

$R_1 \div R_2$

schema	$R_1(X_1X_2), R_2(X_2)$	X_1
istanza	r_1, r_2	$r_1 \div r_2 = \{t \mid \{t\} \bowtie r_2 \subseteq r_1\}$
	input	output

Voli

Codice	Data
AZ427	21/07/2001
AZ427	23/07/2001
AZ427	24/07/2001
TW056	21/07/2001
TW056	24/07/2001
TW056	25/07/2001

Linee

Codice
AZ427
TW056

Voli ÷ Linee

Data
21/07/2001
24/07/2001

Operatore di theta-join

E' la combinazione di prodotto cartesiano e selezione.

$r_1 \bowtie_F r_2 = \sigma_F(r_1 \bowtie r_2)$

con r_1 e r_2 senza attributi in comune e F composta di "predicati di join", ossia del tipo A θ B, con $A \in X_1$ e $B \in X_2$.

Se F è una congiunzione di uguaglianza si parla più propriatamete di **equi-join**.

Ricercatori

Nome	CodProgetto
Rossi	HK27
Verdi	HAL2000
Bianchi	HK27
Verdi	HK28
Neri	HAL2000

Progetti

Sigla	Responsabile
HK27	Bianchi
HAL2000	Neri
HK28	Verdi

Ricercatori $\triangleright\triangleleft_{CodProgetto=Sigla}$ Progetti

Nome	CodProgetto	Sigla	Responsabile
Rossi	HK27	HK27	Bianchi
Verdi	HAL2000	HAL2000	Neri
Bianchi	HK27	HK27	Bianchi
Verdi	HK28	HK28	Verdi
Neri	HAL2000	HAL2000	Neri

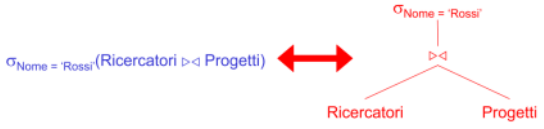
Ricercatori $\triangleright\triangleleft_{(CodProgetto=Sigla)\ AND\ (Nome \neq Responsabile)}$ Progetti

Nome	CodProgetto	Sigla	Responsabile
Rossi	HK27	HK27	Bianchi
Verdi	HAL2000	HAL2000	Neri

Espressioni

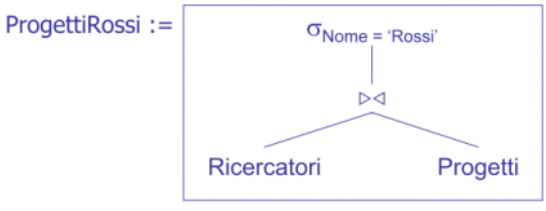
Gli operatori dell'AR si possono liberamente combinare tra loro, avendo cura di rispettare le regole stabilite per la loro applicabilità.

Oltre alla rappresentazione lineare è anche possibile adottare una rappresentazione grafica ad albero di tipo "bottom-up".



Viste

Le viste non sono altro che espressioni a cui viene assegnato un nome. E' quindi possibile utilizzare le viste all'interno di altre espressioni per semplificare la scrittura di espressioni complesse.



La sintassie è V := E

DDL

martedì 2 ottobre 2018 13.41

Il DDL di SQL permette di definire, modificare ed eliminare schemi di relazioni (table) e permettedi specificare vincoli sia a livello di riga che di tabella.
Inoltre si possono definire tabelle virtuali dette viste (view).

CREATE TABLE
L'istruzione CREATE TABLE permette di definire lo schema di una tabella e ne definisce un'istanza vuota.

```
CREATE TABLE nome_tabella (
    nome_attributo1 <dominio>,
    [...]
    [ON DELETE  NO ACTION | CASCADE | SET NULL]
    [ON UPDATE  NO ACTION | CASCADE | SET NULL]
)
```

← Operazione non concessa in DB2

DROP TABLE
L'istruzione DROP TABLE permette di eliminare lo schema di una tabella e conseguentemente la relativa istanza.

```
DROP TABLE nome_tabella
```

Tipi di dato più comuni

CHAR(n)	Stringhe di lunghezza fissa n	[1 ≤ n ≤ 254]
VARCHAR(n)	Stringhe di lunghezza variabile ≤ n	[n ≤ 32672]
INT	Interi a 4 byte	
SMALLINT	Interi a 2 byte	
DEC(p,s)	Numeri decimali - p: precisione (numero totale di cifre) [1 ≤ p ≤ 31] - s: scala (numero di cifre a destra del punto decimale) [0 ≤ s ≤ p]	
REAL	Numeri reali a 32 bit	
DOUBLE	Numeri reali a 64 bit	
DATE	Data in formato DD-MM-YYYY	[10 byte]
TIME	Orario in formato HH.MM.SS	[8 byte]
TIMESTAMP	Data e orario in formato YYYY-MM-DD-HH-MM-SS	[26 byte]
BOOLEAN	Valore binario (non può essere usato su DB2 come tipo di attributo)	

Opzioni per la definizione degli attributi
vincoli sul valore
con la clausola CONSTRAINT si può nominare una certa condizione

```
CF char(16) NOT NULL
Anno smallint DEFAULT 1
Stipendio dec(7,2) NOT NULL CHECK (Stipendio > 0)
Stipendio dec(7,2) NOT NULL CONSTRAINT StipendioPos CHECK (Stipendio > 0)
```

valori univoci / primary key

```
CF char(16) NOT NULL UNIQUE
Matricola char(5) NOT NULL PRIMARY KEY
```

ATTENZIONE: In DB2 è obbligatorio aggiungere il vincolo NOT NULL , ovvero non si possono dichiarare Primary Key con valori nulli.

foreign key
Nel caso in cui non si specifichino gli attributi destinazione si fa riferimento all'intera chiave primaria

```
CodCorso int REFERENCES Corsi(CodCorso)
CodCorso int REFERENCES Corsi
```

DDL

giovedì 11 ottobre 2018 10.33

ALTER TABLE

L'istruzione ALTER TABLE permette di modificare lo schema di una tabella, in particolare aggiungendo o rimuovendo attibuti e/o vincoli.

```
ALTER TABLE nome_tabella (
     ADD COLUMN nome_attributo <dominio>,
     ADD CONSTRAINT nome_vincolo CHECK (<condizione>)
     DROP UNIQUE (nome_attributo_x, nome_attributo_y)
     DROP CONSTRAINT nome_vincolo_x
     [...]
)
```

DML

lunedì 8 ottobre 2018 16.41

SELECT
E' l'istruzione che permette di eseguire interrogazioni sul DB.

```
SELECT [DISTINCT] [nome_tabella.]nome_attributo | * | <espressione> [AS <alias>] [, ...]
FROM nome_tabella [AS <alias>] [, ...]
[[CROSS | LEFT | RIGHT | FULL] JOIN nome_tabella_x ON (<espressione_di_match>) [...]]
[WHERE <condizione> [AND | OR | NOT <condizione>]]
[ORDER BY <criterio> [DESC] [, ...]]
```

DISTINCT	*elimina le righe duplicate*
<espressione>	*un' espressione del tipo:* nome_attributo/3
<espressione_di_match>	*un' espressione del tipo:* nome_tabella_x.attr_FK = nome_tabella.attr_PK
DESC	*ordine decrescente*

Operatori per le stringhe:

CONCAT	permette di concatenare stringhe es: SELECT Matricola, Nome CONCAT ' ' CONCAT Cognome ...
UPPER	Converte una stringa in caratteri maiuscoli
LOWER	Converte una stringa in caratteri minuscoli

Operatori <condizione>:

> | < | = | != (o <>)	operatori comuni
IS | IS NOT	operatore di test per valori NULL
LIKE	operatore per le stringhe che , mediante le wildcard: • _ *un carattere arbitrario* • % *una stringa arbitraria* permette di trovare stringhe che soddifano un certo pattern
BETWEEN <val> AND <val>	operatore per valori numerici che verifica che tale valore sia compreso nell'intervallo specificato
IN	

L'istruzione SELECT però non permette di eseguire unione, intersezione e differenza di tabelle. Per far ciò occorre combinare in modo opportuno i risultati di due istruzioni SELECT mediante gli operandi:

- **UNION [ALL]**
- **INTERSECT [ALL]**
- **EXCEPT [ALL]**

Clausola ALL necessaria se si vuole mantenere i duplicati durante le operazioni

DML

giovedì 11 ottobre 2018 11.00

Istruzioni di aggiornamento dei dati
Le istruzioni che permettono di aggiornare il DB sono:

- **INSERT**
- **DELETE**
- **UPDATE**

INSERT
L'istruzione insert permette di inserire una o più tuple specificando i valori per i singoli attributi o selezionandoli da un'altra tabella.

```
INSERT INTO nome_tabella (att_1, att_2, ..., att_n)
VALUES   (val_1, val_2, ..., val_n),
         [...]
```

oppure

```
INSERT INTO nome_tabella_dst (att_1, att_2, att_3)
SELECT att_x, att_y, att_z
FROM nome_tabella_src
```

N.B. Si possono omettere i nomi degli attributi se vengono inseriti tutti e nell'ordine in cui sono stati definiti nella tabella.

DELETE
L'istruzione DELETE può far uso di una condizione per specificare le tuple da cancellare.

```
DELETE FROM nome_tabella
[WHERE <condizione>]
```

UPDATE
L'istruzione UPDATE può far uso di una condizione per specificare le tuple da modificare e di espressioni per determinare i nuovi valori.

```
UPDATE nome_tabella
SET  nome_attributo_1 = val_1,
     ...                     ,
     nome_attributo_n = val_n
WHERE <condizione>
```

Raggruppamenti

venerdì 12 ottobre 2018 14.11

In molti casi è utile ottenere dal DB informazioni (di sintesi) che caratterizzano gruppi di tuple.
A tale proposito SQL mette a disposizione due strumenti:

1. **Funzioni aggregate**

MIN	minimo
MAX	massimo
SUM	somma
AVG	media aritmetica
COUNT	contatore

N.B.
- Il COUNT non considera i record in cui i valori selezionati sono tutti nulli
- Ricordarsi del casting quando si usa la funzione AVG:
```
SELECT AVG(CAST(Stipendio AS Decimal(6,2))) AS AvgStip
FROM Imp
```

2. **Clausola GROUP BY**

```
SELECT Sede, COUNT(*) AS NumProgrammatori
FROM Imp
WHERE Ruolo = 'Programmatore'
GROUP BY Sede
```

Osservazione: Oltre alle funzioni aggregate, nella clausola SELECT, possono essere presenti gli attributi presenti nella clausola GROUP BY ma non altre.

HAVING
Ha una funzione simile a quella che ha la clausola WHERE per le tuple ma ha effetto sui gruppi.

N.B. L'unico caso in cui la funzione HAVING ha lo stesso effetto di WHERE è quando vado a fare condizioni sugli attributi nel GROUP BY:

```
SELECT Sede, COUNT(*) AS NumImp FROM
Imp
GROUP BY Sede
 HAVING Sede <> 'S01'
```

oppure

```
SELECT Sede, COUNT(*) AS NumImp FROM
Imp
WHERE Sede <> 'S01'
GROUP BY Sede
```

Subquery

lunedì 15 ottobre 2018 16.44

E' possibie esprimere delle condizioni che si basano sul risultato di altre interrogazioni (subquery)

```
SELECT CodImp
FROM Imp
WHERE Sede IN (    SELECT Sede
                   FROM Sedi
                   WHERE Citta = 'Milano')
```

Gli operatori di confronto =, <, … si possono usare solo quando la subquery restituisce non più di una tupla (**subquery scalare**)

```
SELECT CodImp
FROM Imp
WHERE Stipendio = (    SELECT MIN(Stipendio)
                       FROM Imp)

SELECT CodImp
FROM Imp
WHERE Stipendio =  ANY | ALL (  SELECT Stipendio
                                FROM Imp)
```

ANY La relazione vale per almeno uno dei valori ALL La relazione vale per tutti i valori La forma =ANY equivale a IN La forma <>ALL equivale a NOT IN

Attenzione!

R1

A	B
a_1	b_1
a_2	b_2

R2

B	C
b_1	c_1
b_2	c_2
NULL	c_3

```
SELECT *
FROM R1
WHERE B <>ALL (SELECT B
                FROM R2
                WHERE B IS NOT NULL)
```

Senza la clausola WHERE, nel caso incontrasse un valore null la subquery l'espressione non saprebbe se valutarla uguale o diversa a B.

Subquery correlate

```
SELECT Sede
FROM Sedi S
WHERE EXISTS (    SELECT * FROM Imp
                  WHERE Ruolo = 'Programmatore'
                  AND Sede = S.Sede)
```

Per ogni tupla del blocco esterno, considera il valore di S.Sede e risolvi la query innestata.

Viste

venerdì 26 ottobre 2018 14.21

Mediante l'istruzione CREATE VIEW si definisce una vista ovvero una "tabella virtuale"

```
CREATE VIEW ProgSedi(CodProg, CodSede)
AS  SELECT P.CodProg, S.Sede
    FROM Prog P, Sedi S
    WHERE P.Citta = S.Citta
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

Tale tabella è interrogabile come una qualsiasi tabella
La clausola WITH CHECK OPTION serve a far in modo che le tuple che non rispettano le specifiche della vista non possano essere inserite nella stessa.

Common table expression
L'idea alla base delle "common table expressions" è definire una "vista temporanea" che può essere usata in una query come se fosse a tutti gli effetti una VIEW:

```
WITH SediStip(Sede,TotStip)
AS ( SELECT Sede,SUM(Stipendio)
    FROM
    Imp GROUP BY Sede)
SELECT Sede
FROM SediStip
WHERE TotStip = ( SELECT MAX(TotStip)
                  FROM SediStip )
```

Interrogazioni ricorsive
Nel caso in cui il DB è aciclico (prima o poi l'esecuzione è garantita terminare) e non è necessaria nessuna informazione aggiuntiva per i "percorsi" trovati il pattern è:

```
WITH Antenati(Persona, Avo)
AS (( SELECT Figlio, Genitore          -- subquery base
      FROM Genitori)
     UNION ALL                        -- sempre UNION ALL!
    ( SELECT G.Figlio, A.Avo          -- subquery ricorsiva
      FROM Genitori G, Antenati A
      WHERE G.Genitore = A.Persona))
SELECT Avo
FROM Antenati
WHERE Persona = 'Anna'
```

Se invece è importante l'informazione sulla lunghezza del percorso (ad esempio il grado di parentela):

- *Se esso è univoco, cioè posso raggiungere un dato solo mediante un percorso*

```
WITH Antenati(Persona, Avo, Len)
AS (( SELECT Figlio, Genitore, 1          -- caso base: Len = 1
      FROM Genitori)
     UNION ALL
    ( SELECT G.Figlio, A.Avo, A.Len+1
      FROM Genitori G, Antenati A
      WHERE G.Genitore = A.Persona))
SELECT *
FROM Antenati
WHERE Persona = 'Anna'
```

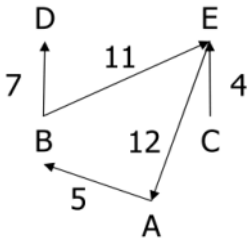
Viste

lunedì 29 ottobre 2018 17.22

- *Se esso NON è univoco, cioè posso raggiungere un dato mediante più percorsi*

```
WITH Percorsi(Composto, Componente, Qty)
AS (( SELECT Parte,Subparte, Qta
      FROM Parti)
      UNION ALL
     ( SELECT H.Composto, P.Subparte, H.Qty*P.Qta
       FROM Parti P, Percorsi H
       WHERE H.Componente = P.Parte))
SELECT Composto, Componente, SUM(Qty) AS QtaTot
FROM Percorsi
GROUP BY Composto, Componente
```

Nel caso in cui il DB è ciclico occorre inoltre prevedere una condizione di stop:



Paesi

From	To	Km
A	B	5
B	D	7
B	E	11
C	E	4
E	A	10

Percorsi

Da	A	Lungh
...	...	...
A	D	12
A	E	16
C	A	16
E	B	17

```
WITH Percorsi(Da, A, TotKm)
AS (( SELECT From, To, Km
  FROM Paesi)
  UNION ALL
  ( SELECT P.Da, S.To, P.TotKm + S.Km
    FROM Paesi S, Percorsi P
    WHERE P.A = S.From
    AND P.Da <> S.To
    AND P.TotKm + S.Km <= 17))
SELECT * FROM
FROM Percorsi
```

Autorizzazioni

lunedì 5 novembre 2018 16.49

In SQL ogni operazione deve essere autorizzata, ovvero l'utente che esegue l'operazione deve avere i privilegi necessari. Mediante GRANT e REVOKE si controllano le autorità, ovvero il diritto ad eseguire azioni amministrative di un certo tipo. La clausola WITH GRANT OPTION autorizza l'utente interessato a passare l'autorità ad altri utenti.

```
GRANT | REVOKE  ALL | <lista_di_privilegi>
ON SCHEMA <schema_name>
TO USER <utente>  |  PUBLIC [, ...]
[ WITH GRANT OPTION ]
```

<lista_di_privilegi>	CREATEIN: *creare oggetti* ALTERIN: *modificare struttura oggetti* DROPIN: *eliminare oggetti*

```
GRANT | REVOKE  ALL | <lista_di_privilegi>
ON [TABLE] <table_name>
TO USER <utente>  |  PUBLIC [, ...]
[ WITH GRANT OPTION ]
```

<lista_di_privilegi>	CONTROL: *tutti i privilegi. Permette di conferire tali privilegi anche ad altri utenti. Può essere conferito solo da SYSADM o DBADM. ALL è diverso da CONTROL: definisce tutti i permessi che l'utente può conferire ma in ogni caso non CONTROL.* ALTER : *modificare la definizione di una tabella* DELETE: *cancellare righe di una tabella* INDEX : *creare un indice sulla tabella* INSERT: *inserire righe nella tabella* REFERENCES: *diritto di definire foreign key in altre tabelle che referenziano la tabella* SELECT: *eseguire query sulla tabella* UPDATE: *modificare righe della tabella*

N.B. Per effettuare REVOKE bisogna avere l'autorità di SYSADM, DBADM oppure privilegio di CONTROL sulla relazione.

Trigger

lunedì 26 novembre 2018 17.13

Un DBMS si dice **attivo** quando dispone di un sottosistema integrato per definire e gestire regole.
I **trigger** sono un caso specifico di regole di tipo ECA (Evento, Condizione, Azione).
I trigger vengono tipicamente usati per gestire vincoli di integrità, calcolare dati derivati, gestire eccezioni, ...

Un trigger può attivarsi prima o dopo l'evento corrispondente:

- BEFORE viene usato per condizionare l'esito dell'operazione oppure bloccarla segnalando errore
  - SELECT selezionare
  - SIGNAL segnalare un errore
  - SET modificare uno o più campi di una tupla
- AFTER viene usato per reagire alla modifica del DB mediate opportune azioni

Un trigger può attivarsi:

- FOR EACH ROW per ogni tupla
- FOR EACH STATEMENT una volta per istruzione

Parole chiave:

- OLD valore tupla prima della modifica
- OLD_TABLE ipotetica tabella che contiene tutte le tuple modificate con i valori prima della modifica
- NEW valore tupla dopo della modifica
- NEW_TABLE ipotetica tabella che contiene tutte le tuple modificate con i valori dopo la modifica

Progettazione di basi di dati

venerdì 9 novembre 2018 14.14

Le fasi di progrettazione di un sistema informativo seguono uno schema a cascata:

La progettazione riguarda 2 aspetti:

- Progettazione dei dati
- Progettazione delle applicazioni

Per progettare una base di dati di buona qualità è necessario seguire una **metodologia di progettazione** che:

- definisca le fasi in cui l'attività di progettazione si articola
- fornisca dei criteri per scegliere tra diverse alternative
- sia supportata da dei modelli di rappresentazione
- sia di applicabilità generale e facile da utilizzare

Fasi di progettazione

1. REQUISITI DEL SISTEMA INFORMATIVO
2. PROGETTAZIONE CONCETTUALE
   schema concettuale
3. PROGETTAZIONE LOGICA
   schema logico
4. PROGETTAZIONE FISICA
   schema fisico

Schema concettuale

Lo schema concettuale consiste in una descrizione formalizzata e integrata delle esigenze aziendali espressa in modo indipendente dal DBMS. Il modello più noto per rappresentare i dati in tale modo è l'**Entity-Relationship** (ER).

Schema logico

Lo schema logico consiste nella traduzione dello schema concettuale nel modello dei dati del DBMS. In questa fase si considerano anche gli aspetti legati a:

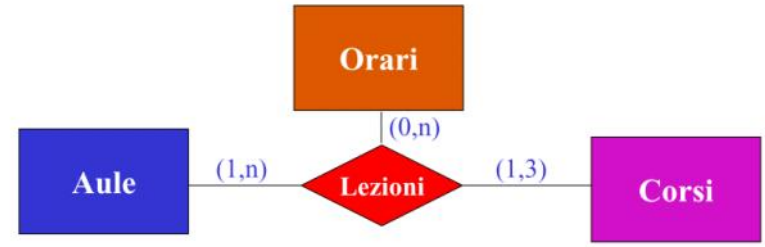
- Vincoli di integrità e consistenza dei dati
- Efficienza delle operazioni

Modello ER di base

venerdì 9 novembre 2018 14.42

Concetti di base

Entità	*Insieme (classe) di oggetti della realtà di interesse che possiedono caratteristiche comuni e che hanno esistenza "autonoma".*
Associazione	*Rappresenta un legame logico tra entità rilevante nella realtà che si sta considerando.*
Attributo	*Un atttributo è una proprietà elementare di un'entità o di un'associazione. Ogni attributo è definito su un dominio di valori.*
Vincolo di cardinalità	*Un vincolo è una coppia di valori* (min-card,max-card) *associati a ogni entità che partecipa a un'associazione, che specifica il numero minimo e massimo di istanze dell'associazione a cui un'istanza dell'entità può partecipare.*
Identificatore	*Un identificatore permette l'individuazione univoca delle istanze di un'entità.* *Se il numero di elementi (attributi o entità) che costituiscono l'identificatore è pari a 1 si parla di identificatore* ***semplice****, altrimenti l'identificatore è* ***composto****.*



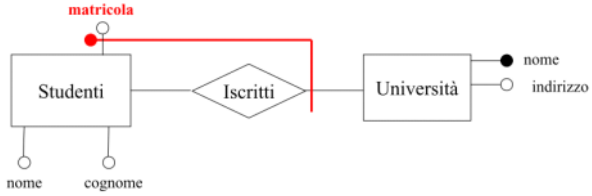
Ogni aula ospita da 1 a n lezioni settimanali
Ogni corso ha da 1 a 3 lezioni settimanali
In ogni ora si tengono da 0 a n lezioni settimanali

Modello ER avanzato

lunedì 12 novembre 2018 17.34

Identificatore esterno

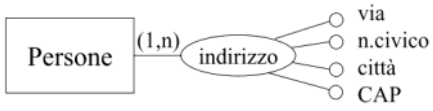
Oltre a poter identificare un'entità E mediante uno o più attributi (identificatore interno), nel modello E/R è prevista la possibilità di identificare E mediante altre (una o più) entità, collegate a E da associazioni, più eventuali attributi di E.



OSS: Se E è identificata esternamente attraverso l'associazione A, allora si ha sempre max-card(E,A) =1

Attributi ripetuti e composti

Nel caso di presenza di più attributi multivalore, la creazione di un attributo composto può rendersi necessaria per evitare ambiguità.



Generalizzazione e specializzazione

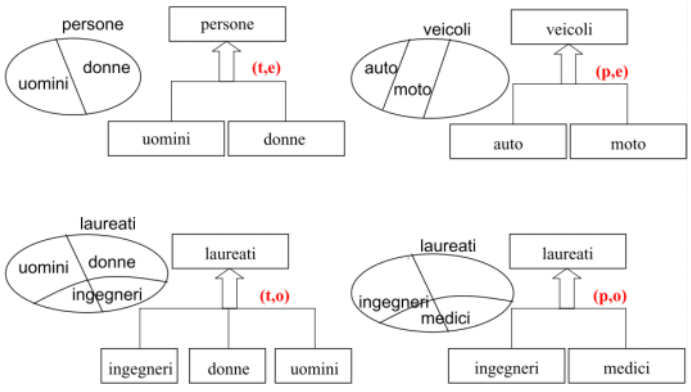
Parliamo di generalizzazione se a partire da più sottoclassi definiamo una superclasse. Specializzazione è il procedimento inverso.

Una generalizzazione è:

- **t***otale* se la classe generalizzata è l'unione delle specializzazioni
- **p***arziale* se la classe generalizzata contiene l'unione delle specializzazioni

Una specializzazione è:

- **e***sclusiva* se le specializzazioni sono fra loro disgiunte
- *sovrapposta* (**o***verlapped*) se può esistere una intersezione non vuota fra le specializzazioni



Subset

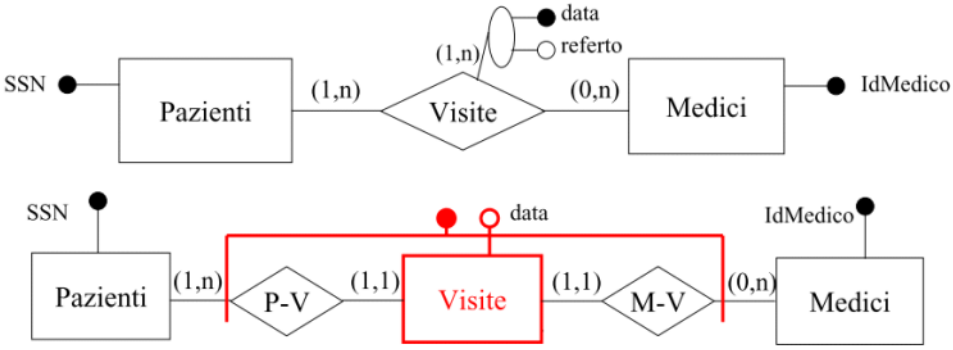
È un caso particolare di gerarchia in cui si evidenzia una sola classe specializzata.

ER pattern di progettazione

venerdì 16 novembre 2018 13.59

Non prendere per tipologie (e quindi per specializzazioni di un'entità) quelle che sono solo istanze dell'entità.
Ad esempio "Le aree del campeggio possono essere : spiaggia, centrale, ingresso …" non si riesce a rappresentare con una gerarchia di aree ma con un attributo semplice.

Nell' esempio "Un paziente può essere visitato da diversi medici, e anche più volte dallo stesso medico, ma in questo caso in giorni diversi" per una corretta rappresentazione occorre **reificare** (trasformare l'associazione in entità):



Progettazione logica: regole di traduzione

venerdì 23 novembre 2018 13.26

Entità

Ogni entità è tradotta con una relazione con gli stessi attributi:

- La chiave primaria coincide con l'identificatore dell'entità. Nel caso di più identificatori i criteri per scegliere la chiave in ordine sono: assenza di opzionalità, semplicità e utilizzo più frequente nelle operazioni altrimenti si può introdurre un codice aggiuntivo che soddisfa lo scopo.
- Se un attributo è opzionale usiamo l'asterisco (*) per indicarlo

Caso identificazione esterna

Oltre all'identificatore dell'entità si importa nella chiave primaria l'identificatore della/e entità esterna/e. L'associazione compresa tra le entità viene inglobata.

Caso gerarchie di entità

- *collasso verso l'alto*: si definisce un attributo Type per le varie tipologie di sotto-entità e si definiscono check per mantenere l'integrità delle relazioni associate alle sotto-entità. L'attributo Type:
  - (t, e) : Type ha N valori quante sono le entità figlie
  - (p, e): Type ha N+1 valori; il valore in più serve per le istanze non considerate
  - (t, o) - (p, o): Type è un boolean nelle sotto-entità che indica l'appartenenza dell'entità alla sotto-entità
- *collasso verso il basso*: si replicano gli attributi dell'entità nelle sotto-entità. E' valida solo nel caso (t, e).
- *introducendo associazioni*: si introduce un'associazione tra l'entità e la sotto-entità con identificazione esterna

Associazioni

Ogni associazione è tradotta con una relazione con gli stessi attributi, cui si aggiungono gli identificatori di tutte le entità che essa collega:

- gli identificatori delle entità collegate costituiscono una superchiave
- la chiave primaria dipende dalle cardinalità massime delle entità nell'associazione:
  - **Caso molti a molti**
    La chiave primaria coincide con l'unione degli identificatori delle entità collegate
  - **Caso uno a molti**
    La chiave primaria coincide con l'identificatore dell'entità che partecipa con cardinalità massima 1
    alternativa: si ingloba l'associazione nella relazione dell'entità che partecipa con cardinalità massima 1 *
  - **Caso uno a uno**
    La chiave primaria è uno dei due identificatori, l'altro diventa una chiave alternativa
    in più: si ingloba l'associazione nella relazione di una delle due entità (sconsigliato se la cardinalità minima è 0 per il fatto che la relazione finale conterrà molti valori nulli).

    alternativa:si accorpano associazione e due entità in un'unica relazione in cui:
    - nel caso min-card(E1,R) = min-card(E2,R) = 1 la primary key è una delle due
    - nel caso min-card(E1,R) = 0 e min-card(E2,R) = 1 la primary key è quella con cardinalità minima 1
    - nel caso min-card(E1,R) = 0 e min-card(E2,R) = 0 occorre introdurre un codice per primary key

* PRO:
Si semplifica la scrittura di query SQL e si riduce il numero di join da eseguire migliorando le prestazioni
CONTRO:
Se l'entità che ingloba l'associazione partecipa con cardinalità minima 0 l'operazione è vantaggiosa solo se i valori che possono non partecipare sono relativamente pochi altrimenti si ha uno spreco di spazio. E in più occorre garantire che i valori della foreign key siano o tutti definiti o tutti nulli.

Caso associazioni n-arie

Valgono le considerazioni precedenti per ogni entità collegata all'associazione

Caso associazioni ad anello

Valgono le considerazioni precedenti

Attributi

Caso attributi composti : vengono tradotti ricorsivamente nelle loro componenti e utilizzando un prefisso comune.
Caso attributi multivalore : si aggiungono alla chiave primaria dell'entità e ne differenziano le istanze.

Progettazione logica: ottimizzazione

venerdì 7 dicembre 2018 14.21

Il **carico di lavoro** (costo) del DB dipende da:

- il volume dei dati in gioco
- le principali operazioni che il DB dovrà supportare

Per la valutazione del costo:

1. Si definisce la **tavola dei volumi** che specifica il numero stimato di istanze per ogni entità/associazione
2. Si definisce lo **schema di navigazione** (la parte dello schema E/R interessata dall'operazione)
3. Si costruisce una **tavola degli acessi** basata sullo schema di navigazione che, per ogni entità/associazione, riporta il tipo e il numero di accessi ricavato da stime sulla tavola dei volumi.

Analisi delle ridondanze

Una ridondanza in uno schema E/R è una informazione significativa ma derivabile da altre.
Le possibili ridondanze riguardano attributi derivabili da altri o associazioni in cascata.
E' importante in fase di progettazione decidere se mantenere tali ridondanze per semplificare il carico di lavoro a discapito di spazio occupato e appesantimento degli aggiornamenti dei dati.

Progettazione logica: normalizzazione

giovedì 17 gennaio 2019 19.14

Una **forma normale** è una proprietà di uno schema relazionale che ne garantisce la qualità, cioè l'assenza di determinati difetti.
Una relazione non normalizzata:

- presenta ridondanze
- esibisce comportamenti poco desiderabili durante gli aggiornamenti

Per formalizzare il processo di normalizzazione si introduce un nuovo tipo di vincolo, la **dipendenza funzionale (FD)**: Dato uno schema R(X, Y, Z) si ha $X \longrightarrow Y$ (FD) se e solo $\forall$ r istanza di R(X) non esistono due tuple distinte t_1 e t_2 tali che $t_1[X] = t_2[X]$ e $t_1[Y] \neq t_2[Y]$. Ovvero se t_1 e t_2 hanno gli stessi valori su X, allora hanno gli stessi valori anche su Y.
Si parla di **dipendenza funzionale banale** $X \longrightarrow Y$ se $Y \subseteq X$.

Prima forma normale (1NF)
Uno schema R(X) è in 1NF se:

- ogni campo rappresenta informazioni elementari e non multiple
- esiste una chiave primaria

Seconda forma normale (2NF)
Uno schema R(X) è in 2NF se:

- è in prima forma normale
- non esistono dipendenze funzionali parziali; cioè gli attributi non facenti parte della chiave devono dipendere dall'intera chiave

Terza forma normale (3NF)
Uno schema R(X) è in 3NF se:

- è in seconda forma normale
- per ogni dipendenza funzionale (non banale) $Y \longrightarrow Z$ definita su di esso, Y è una superchiave di R(X) oppure ogni attributo in Z è contenuto in almeno una chiave di R(X).

Forma normale di Boyce-Codd (BCNF)
Uno schema R(X) è in BCNF se

- è in terza forma normale
- per ogni dipendenza funzionale (non banale) $Y \longrightarrow Z$ definita su di esso, Y è una superchiave di R(X).

Per portare uno schema in BCNF occorre fare una **decomposizione senza perdita** ovvero:

- se si effettua join sulle tabelle ottenute dalla scomposizione il risultato non deve avere record aggiuntivi rispetto alla tabella di partenza
- devono essere mantenute le dipendenze funzionali originarie