



Basi di dati

Maurizio Lenzerini

***Dipartimento di Ingegneria Informatica
Automatica e Gestionale “Antonio Ruberti”
Università di Roma “La Sapienza”***

Anno Accademico 2023/2024

<http://www.dis.uniroma1.it/~lenzerini/?q=node/44>



2. Il modello relazionale

2.1 Struttura dei dati relazionali

- | | |
|------------|---------------------------------------|
| 2.1 | struttura dei dati relazionali |
| 2.2 | algebra relazionale |
| 2.3 | vincoli di integrità |



Il modello relazionale

- Proposto da E. F. Codd nel 1970 per favorire l'indipendenza fisica dei dati (ovvero per rendere il modo in cui si usano i dati a livello logico indipendente dalla loro memorizzazione fisica)
- Disponibile come modello logico in DBMS reali nel 1981 (10 anni di incubazione)
- Si basa sul concetto matematico di **relazione** (ma con importanti varianti)
- Le relazioni hanno una rappresentazione naturale per mezzo di tabelle



Relazione matematica

- Siano D_1, D_2, \dots, D_n n insiemi, non necessariamente distinti
- il **prodotto cartesiano** $D_1 \times D_2 \times \dots \times D_n$ è l'insieme di tutte le n -uple ordinate (d_1, d_2, \dots, d_n) tali che $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$
- una relazione matematica sugli insiemi D_1, D_2, \dots, D_n è un **sottoinsieme del prodotto cartesiano** $D_1 \times D_2 \times \dots \times D_n$
- se R è una relazione matematica sugli insiemi D_1, D_2, \dots, D_n , tali insiemi sono detti i **domini** della relazione R . Una relazione su n domini si dice che ha **grado** (o **arità**) n
- il numero di n -uple contenute in una relazione è la **cardinalità** della relazione stessa



Relazione matematica: esempio

- $D_1 = \{\text{Conti}, \text{Totti}, \text{Mancini}\}$
- $D_2 = \{\text{Genoa}, \text{Roma}\}$
- prodotto cartesiano $D_1 \times D_2 \longrightarrow$

Conti	Genoa
Conti	Roma
Totti	Genoa
Totti	Roma
Mancini	Genoa
Mancini	Roma

- una relazione: $\text{HaGiocato} \subseteq D_1 \times D_2 \longrightarrow$

Conti	Roma
Conti	Genoa
Totti	Roma



Relazione matematica: proprietà

Una **relazione matematica** è quindi un insieme di ennuple ordinate (dette anche **n-ple**, o **tuple**) su D_1, D_2, \dots, D_n , dove ogni tupla è una sequenza di valori della forma (d_1, \dots, d_n) tale che $d_1 \in D_1, \dots, d_n \in D_n$. Attenzione: la tupla (d_1, \dots, d_n) viene anche scritta come $\langle d_1, \dots, d_n \rangle$.

Si noti che una relazione è un **insieme** di tuple e quindi:

- non è definito alcun ordinamento fra le sue tuple
- le sue tuple sono tutte distinte una dall'altra (non ci sono duplicati)

Ciascuna tupla di una relazione è **ordinata** nel senso che

- il valore del primo elemento della tupla proviene dal primo dominio
- il valore del secondo elemento della tupla proviene dal secondo dominio
- ...
- il valore dell' n -esimo elemento della tupla proviene dall' n -esimo dominio



Relazione nel modello relazionale dei dati

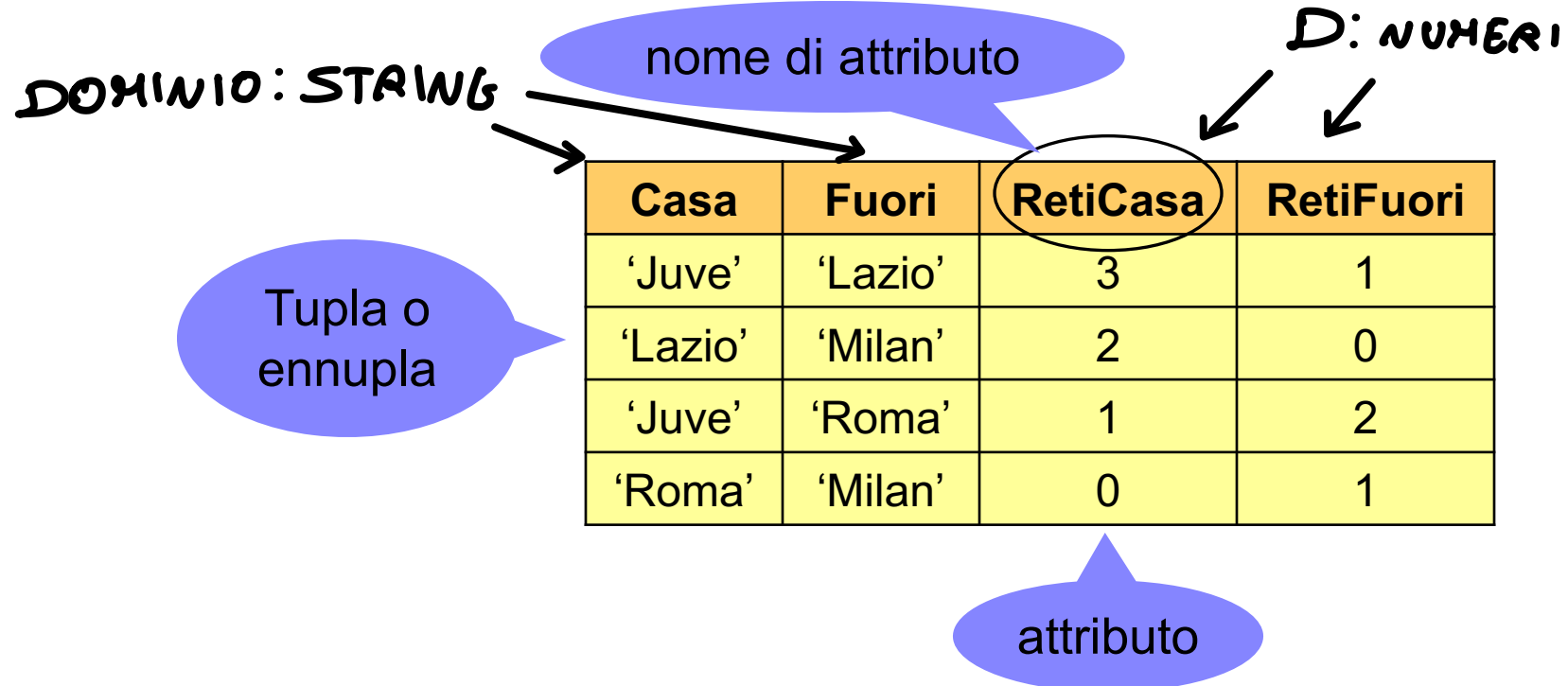
- Una **relazione nel modello relazionale** è simile ad una relazione matematica, ma con le seguenti differenze:
 - le posizioni che determinano le varie componenti delle tuple sono dette **attributi**,
 - ogni attributo è caratterizzato da un nome ed un insieme di valori **atomici**, quest'ultimo detto **dominio** dell'attributo; si noti che un valore atomico è un valore semplice (ad esempio, un intero o una stringa), non costituito da una struttura complessa
 - non può succedere che due attributi della stessa relazione abbiano lo stesso nome
- Da ora in poi, quando parliamo di relazione intendiamo “relazione nel modello relazionale”, e quando vogliamo parlare di relazione matematica useremo il termine “relazione matematica”



Relazione nel modello relazionale dei dati

- Una relazione nel modello relazionale si può quindi rappresentare come una **tabella** in cui gli attributi corrispondono alle colonne ed i nomi degli attributi (tutti diversi l'uno dall'altro, come detto prima) sono usati come intestazioni delle colonne
- Poiché ad ogni colonna della relazione è associato il nome di un attributo, l'ordinamento delle colonne nella tabella è irrilevante: in altre parole, la struttura, al contrario della relazione matematica, è **non posizionale**
- Ogni riga della tabella corrisponde ad una tupla della relazione, e siccome la relazione è un **insieme** di tuple (cioè una collezione senza ripetizioni), non ci possono essere due righe uguali nella tabella.

Relazione nel modello relazionale dei dati: esempio



- In questa relazione gli attributi sono 4, e nella rappresentazione tabellare essi corrispondono alle colonne della tabella. Le tuple, invece, corrispondono alle righe.
- I domini degli attributi sono *string* per Casa e Fuori, ed *integer* per RetiCasa e RetiFuori. Per non appesantire la notazione, essi non vengono mostrati nella rappresentazione tabellare

Relazione nel modello relazionale dei dati: esempio

C	F	Casa	Fuori
‘Juve’	‘Lazio’	3	1
‘Lazio’	‘Milan’	2	0
		3	
		5	
‘Juve’	‘Roma’	1	2
‘Roma’	‘Milan’	0	1

(1)

A	B	A
5	2	3
5	4	2

(2)

valore non
atomico

- Quella mostrata in (1) **non** è una relazione, perché nell'attributo Casa della seconda tupla compare un array di interi, che non è un valore atomico.
- Quella mostrata in (2) **non** è una relazione, perché il suo schema contiene due attributi con lo stesso nome.

Notazioni

- Sia X l'insieme degli attributi di una relazione R . Se t è una tupla di R , cioè una tupla su X , e se $A \in X$, allora
 $t[A]$ oppure anche $t.A$
detta la “restrizione” di t su A , indica **il valore che la tupla t ha in corrispondenza dell'attributo A**
- Se t è la tupla che compare per prima nella tabella dell'esempio precedente, allora si ha che $t[\text{Fuori}] = \text{Lazio}$
- La stessa notazione è estesa anche ad insiemi di attributi:
 $t[\text{Fuori}, \text{RetiFuori}]$ indica la restrizione della tupla t su due attributi, che sarà una tupla sui due attributi Fuori e RetiFuori
- Riferendoci alla tupla t vista prima, si ha che
 $t[\text{Fuori}, \text{RetiFuori}] = \langle \text{Lazio}, 1 \rangle$
- Per completezza definiamo anche $t[]$ (cioè la restrizione della tuple t sull'insieme vuoto di attributi, e stabiliamo che $t[]$ sia la cosiddetta “tupla vuota”, cioè una tupla senza componenti).



Esempi di restrizioni di tuple su attributi

	Casa	Fuori	RetiCasa	RetiFuori
t1 →	'Juve'	'Lazio'	3	1
t2 →	'Lazio'	'Milan'	2	0
t3 →	'Juve'	'Roma'	1	2
	'Roma'	'Milan'	0	1

- $t1[Fuori] = \text{'Lazio'}$
- $t2[RetiCasa] = 2$
- $t3[Casa, RetiCasa] = \langle \text{'Juve'}, 1 \rangle$
- $t1[RetiFuori, Fuori] = \langle 1, \text{'Lazio'} \rangle$
- $t3[Fuori, Casa, RetiFuori] = \langle \text{'Roma'}, \text{'Juve'}, 2 \rangle$
- $t3[Fuori, Retifuori, Retifuori] = \langle \text{'Roma'}, 2, 2 \rangle$



Altra notazione

- La specifica del nome R di una relazione, degli attributi A_1, A_2, \dots, A_n e dei domini di tali attributi D_1, D_2, \dots, D_n forma il cosiddetto **schema di relazione**, che si denota come

$$R(A_1 : D_1, A_2 : D_2, \dots, A_n : D_n)$$

oppure semplicemente come (nel caso non interessi esplicitare i domini degli attributi)

$$R(A_1, A_2, \dots, A_n)$$

- Di conseguenza, una tupla di R con $t[A_1] = a, t[A_2] = b, \dots, t[A_n] = c$ si può denotare anche come $\langle A_1:a, A_2:b, \dots, A_n:c \rangle$
- In altre parole, una tupla di una relazione si può rappresentare come “**tupla etichettata**”, in cui le etichette sono gli attributi della relazione, ed i valori associati alle etichette sono i valori in corrispondenza dei vari attributi



Relazioni: riassunto

- Sottolineiamo che in una relazione del modello relazionale:
 - i valori di ciascuna colonna sono fra loro omogenei, cioè appartengono allo stesso dominio
 - le righe (cioè le tuple) sono tutte diverse fra loro (perché la relazione **è un insieme di tuple, non un multiinsieme**)
 - le intestazioni delle colonne (attributi) sono tutte diverse tra loro
- Inoltre, nella rappresentazione tabellare della relazione
 - l'ordinamento tra le righe è irrilevante
 - l'ordinamento tra le colonne è irrilevante
- Sottolineiamo anche che **il modello relazionale è basato sui valori**: eventuali riferimenti fra due relazioni diverse sono espressi per mezzo di valori che compaiono nelle tuple di entrambe le relazioni



Studente

Matricola	Cognome	Nome	Data di nascita
6554	Rossi	Mario	05/12/1978
8765	Neri	Paolo	03/11/1976
9283	Verdi	Luisa	12/11/1979
3456	Rossi	Maria	01/02/1978

Esame

Studente	Voto	Corso
3456	30	04
3456	24	02
9283	28	01
6554	26	01

Corso

Codice	Titolo	Docente
01	Analisi	Mario
02	Chimica	Bruni
04	Chimica	Verdi



Studente

Matricola	Cognome	Nome	Data di nascita
6554	Rossi	Mario	05/12/1978
8765	Neri	Paolo	03/11/1976
9283	Verdi	Luisa	12/11/1979
3456	Rossi	Maria	01/02/1978

Esame

Studente	Voto	Corso
3456	30	04
3456	24	02
9283	28	01
6554	26	01

collegamento tra
le due tabelle
rappresentato dallo
stesso valore presente
nelle due tabelle

Corso

Codice	Titolo	Docente
01	Analisi	Mario
02	Chimica	Bruni
04	Chimica	Verdi



Definizioni: schemi e istanze

Schema di relazione: un **nome di relazione** R con un insieme non vuoto di **attributi** A_1, \dots, A_n ed eventualmente anche i corrispondenti domini

$$R(A_1, \dots, A_n) \quad \text{oppure} \quad R(A_1 : D_1, A_2 : D_2, \dots, A_n : D_n)$$

Istanza di relazione su uno schema $R(X)$: insieme r delle tuple sull'insieme di attributi X

Relazione: schema Z di relazione + istanza di relazione su Z

Schema di base di dati: insieme non vuoto di schemi di relazione (con nomi diversi) $\mathbf{R} = \{R_1(A_1 A_2 \dots A_n), \dots, R_m(B_1 B_2 \dots B_p)\}$

Istanza di base di dati su uno schema $\mathbf{R} = \{R_1(X_1), \dots, R_m(X_m)\}$: insieme di relazioni $\mathbf{r} = \{r_1, \dots, r_m\}$, dove per ogni $i = 1, \dots, m$, r_i è una istanza relazione sullo schema $R_i(X_i)$

Base di dati: schema S di basi di dati + istanza di basi di dati su S

Esempio di basi di dati relazionale

Basi di dati B

Schema S della base di dati B:

schema della
relazione Studente

{ StudenteLavoratore(Matricola),
Studente(Matricola,Cognome,Nome,DataDiNascita)
}

nome di
relazione

nome di
attributo

Istanza della base di dati B sullo schema S:

StudenteLavoratore

Matricola
6554
3456

relazione
Studente

Studente

istanza della
relazione
Studente

Matricola	Cognome	Nome	DataDiNascita
6554	Rossi	Mario	05/12/1978
8765	Neri	Paolo	03/11/1976
9283	Verdi	Luisa	12/11/1979
3456	Rossi	Maria	01/02/1978



2. Il modello relazionale

2.2 Algebra relazionale

- | | |
|------------|--------------------------------|
| 2.1 | struttura dei dati relazionali |
| 2.2 | algebra relazionale |
| 2.3 | vincoli di integrità |



Linguaggi per basi di dati

- Operazioni sullo schema:
DDL: data definition language
- Operazioni sui dati:
DML: data manipulation language
 - interrogazioni ("query language")
 - aggiornamenti



Linguaggi di interrogazione (query language) per basi di dati relazionali

Tipologia:

- **Dichiarativi**: specificano le proprietà del risultato ("*che cosa*")
- **Procedurali**: specificano le modalità di generazione del risultato ("*come*")

Rappresentanti più significativi:

- **Algebra relazionale**: procedurale
 - **Calcolo relazionale**: dichiarativo
 - **SQL** (Structured Query Language): parzialmente dichiarativo (linguaggio ormai standard **usato in pratica**)
 - **QBE** (Query by Example): dichiarativo
- Noi studieremo **Algebra relazionale** (base teorica di tutti i linguaggi di query) ed **SQL** (standard de facto)



Algebra relazionale

I suoi operatori

- sono definiti su relazioni
- producono relazioni
- e possono essere composti

Operatori dell'algebra relazionale:

- **insiemistici**: unione, intersezione, differenza
- **ridenominazione**
- **selezione**
- **proiezione**
- **join in diverse versioni**: join naturale, prodotto cartesiano, theta-join



Operatori insiemistici

- a livello estensionale (cioè delle istanze), le relazioni sono insiemi di tuple, e quindi è sensato definire per essi gli operatori insiemistici
- sono operatori binari (due operandi) e i risultati dell'applicazione di tali operatori sono a loro volta relazioni (vista la proprietà di chiusura delle algebre)
- è possibile applicare **unione** (simbolo \cup), **intersezione** (simbolo \cap), **differenza** (simbolo $-$) solo a relazioni che hanno uguali definizioni intensionali (ovvero stessi attributi)
- la semantica (ovvero il significato) degli operatori insiemistici è quella classica, e quindi diamo per scontato che non sia necessario ripeterla

Unione

Laureato

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Dirigente

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Laureato \cup Dirigente

INTERSEZIONE \rightarrow

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45
9297	Neri	33

HANNO GLI
STESSI ATTRIBUTI

QUINDI SI POSSONO
UNIRE



Intersezione

Laureato

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Dirigente

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Laureato \cap Dirigente

Matricola	Nome	Età
7432	Neri	54
9824	Verdi	45



Differenza

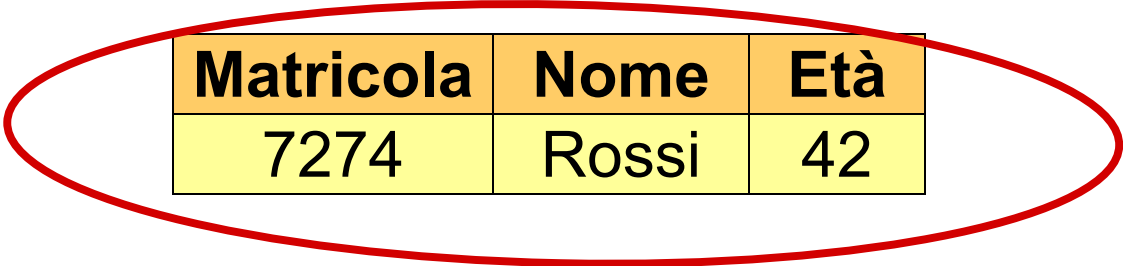
Laureato

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Dirigente

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Laureato – Dirigente



Matricola	Nome	Età
7274	Rossi	42

DOV'È IL RISULTATO?



Importante osservazione sull'algebra relazionale

- L'algebra relazionale è un impianto matematico formato da operatori che lavorano su relazioni. Le relazioni sono quindi i “valori” su cui gli operatori agiscono (esattamente come l'algebra elementare lavora sui valori interi)
- Il risultato di una interrogazione espressa in algebra relazionale è esso stesso una relazione e di tale relazione è rilevante sia l'insieme degli attributi sia, ovviamente, le tuple che lo compongono, ma **ma non è, invece, significativo il nome**. Potremmo scegliere l'espressione sintattica che ha generato il risultato come nome, ma poiché la relazione risultato non fa parte della base di dati (è semplicemente un valore calcolato nel sistema algebrico), tale nome è irrilevante.
- Quando passeremo ad analizzare i sistemi di basi di dati (che, al contrario dell'algebra, non sono solo formalismi matematici, ma sono veri e propri sistemi software), ci occuperemo del **modo in cui vengono gestiti i risultati di interrogazioni** (ad esempio, per memorizzarli in nuove relazioni della base di dati, o per trasferirli all'esterno del sistema)

Un nuovo operatore: la ridenominazione

Un'unione sensata ma impossibile in algebra: vogliamo i genitori ed i loro figli

Paternità

Padre	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

Maternità

Madre	Figlio
Eva	Abele
Eva	Set
Sara	Isacco

Paternità \cup Maternità
??

impossibile
eseguire l'unione:
le due relazioni
hanno differenti
insiemi di attributi

Ridenominazione

- operatore monadico (con un argomento)
- "modifica lo schema" (cambiando il nome di uno o più attributi) lasciando inalterata l'istanza dell'operando

ridenomina l'attributo
"Padre" in "Genitore"

Paternità

Padre	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

$REN_{Genitore \leftarrow Padre}$ (Paternità)

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

Ridenominazione: sintassi e semantica

Sintassi:

$\text{REN}_{A1 \leftarrow B1, A2 \leftarrow B2, \dots, An \leftarrow Bn} (\text{Operando})$

o anche

$\text{REN}_{A1, A2, \dots, An \leftarrow B1, B2, \dots, Bn} (\text{Operando})$

Semantica:

La schema della relazione rappresentata da “Operando” viene modificato sostituendo al nome di attributo B1 il nome A1, al nome di attributo B2 il nome A2, ... , e al nome di attributo Bn il nome An.

Nota: non ci devono essere duplicati negli attributi risultanti dalla ridenominazione (cioè A1,...,An devono essere tutti diversi tra loro)



Paternità

Padre	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

REN_{Genitore ← Padre} (Paternità)

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

Maternità

Madre	Figlio
Eva	Abele
Eva	Set
Sara	Isacco

REN_{Genitore ← Madre} (Maternità)

Genitore	Figlio
Eva	Abele
Eva	Set
Sara	Isacco



$REN_{Genitore \leftarrow Padre}$ (Paternità)

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

$REN_{Genitore \leftarrow Padre}$ (Paternità)



$REN_{Genitore \leftarrow Madre}$ (Maternità)

$REN_{Genitore \leftarrow Madre}$ (Maternità)

Genitore	Figlio
Eva	Abele
Eva	Set
Sara	Isacco

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco
Eva	Abele
Eva	Set
Sara	Isacco

Impiegati

Cognome	Ufficio	Stipendio
Rossi	Roma	55
Neri	Milano	64

Operai

Cognome	Fabbrica	Salario
Bruni	Monza	45
Verdi	Latina	55

REN Sede, Retribuzione ← Ufficio, Stipendio **(Impiegati)**

∪
REN Sede, Retribuzione ← Fabbrica, Salario **(Operai)**

Cognome	Sede	Retribuzione
Rossi	Roma	55
Neri	Milano	64
Bruni	Monza	45
Verdi	Latina	55



Selezione

- operatore monadico (cioè con un argomento) che prevede la formulazione di condizione (una espressione booleana)
- produce un risultato che
 - ha lo stesso schema dell'operando
 - contiene un sottoinsieme delle tuple dell'operando, quelle che soddisfano la condizione espressamente indicata nell'operatore

Impiegato

Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Roma	55
5998	Neri	Milano	64
9553	Milano	Milano	44
5698	Neri	Napoli	64

- impiegati che
 - guadagnano più di 50 >
 - guadagnano più di 50 e lavorano a Milano > &
 - hanno lo stesso nome della filiale presso cui lavorano



Selezione: sintassi e semantica

Sintassi:

SEL *Condizione* (*Operando*)

Condizione: espressione booleana (come quelle dei linguaggi di programmazione, in cui gli operandi utilizzano gli attributi delle relazioni in gioco)

Semantica:

la relazione risultato ha gli stessi attributi dell'operando e contiene le tuple dell'operando che soddisfano la condizione specificata

- impiegati che guadagnano più di 50

Impiegato

Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Roma	55
5998	Neri	Milano	64
5698	Neri	Napoli	64

SEL_{Stipendio > 50} (Impiegato)

- impiegati che guadagnano più di 50 e lavorano a Milano

Impiegato

Matricola	Cognome	Filiale	Stipendio
5998	Neri	Milano	64

SEL_{Stipendio > 50 AND Filiale = 'Milano'} (Impiegato)

- impiegati che hanno lo stesso nome della filiale presso cui lavorano

Impiegato

Matricola	Cognome	Filiale	Stipendio
9553	Milano	Milano	44

SEL $\text{Cognome} = \text{Filiale}(\text{Impiegato})$

Selezione e proiezione

Sono due operatori "ortogonali"

- **selezione:**
 - decomposizione orizzontale
- **proiezione:**
 - decomposizione verticale





Proiezione

- operatore monadico
- produce un risultato che
 - ha parte degli attributi dell'operando
 - contiene tuple cui contribuiscono tutte le tuple dell'operando: sono le tuple ottenute dall'operando eliminando gli attributi che non compaiono nella lista espressamente indicata nell'operatore



Impiegato

Matricola	Cognome	Filiale	Stipendio
7309	Neri	Napoli	55
5998	Neri	Milano	64
9553	Rossi	Roma	44
5698	Rossi	Roma	64

- per tutti gli impiegati:
 - matricola e cognome
 - cognome e filiale



Proiezione: sintassi e semantica

Sintassi:

PROJ *ListaAttributi* (Operando)

Semantica:

- la relazione risultato ha i soli attributi contenuti in *ListaAttributi*, e contiene le tuple ottenute da tutte le tuple dell'operando ristrette agli attributi nella lista

- matricola e cognome di tutti gli impiegati

Impiegato

Matricola	Cognome
7309	Neri
5998	Neri
9553	Rossi
5698	Rossi

PROJ Matricola, Cognome (Impiegato)



- cognome e filiale di tutti gli impiegati

Impiegato

Cognome	Filiale
Neri	Napoli
Neri	Milano
Rossi	Roma

PROJ Cognome, Filiale **(Impiegato)**



Cardinalità delle proiezioni

- una proiezione
 - contiene al più tante tuple quante l'operando
 - può contenerne di meno, a causa di eliminazione di duplicati
- Nota:
vedremo che se X contiene un insieme di attributi “superchiave” per R , allora $PROJ_X(R)$ contiene esattamente tante tuple quante R



Selezione e proiezione

- Combinando selezione e proiezione, possiamo estrarre interessanti informazioni da una relazione

Restituire matricola e cognome degli impiegati che guadagnano più di 50

Impiegato

Matricola	Cognome
7309	Rossi
5998	Neri
5698	Neri

PROJ_{Matricola,Cognome} (SEL_{Stipendio > 50} (Impiegato))



Esercizio 4 (con soluzioni)

Concerto

idconc	idart	anno	città
209	73	1999	Roma
298	73	2001	Napoli
253	59	2020	Milano
299	95	2020	Roma
276	59	2018	Firenze
223	56	1995	Firenze

Artista

idart	nome	tipo
73	S&G	duo
59	Neil	solo
95	Bruce	solo
56	RolSto	band
40	Beatles	band

Evento

ideve	idconc	voto
2	209	9
4	253	4
8	276	6
2	223	10
8	299	9

Esercizio 4 – query 1

Trovare id del concerto (idconc) e id dell'artista (idart) dei concerti tenuti a Roma.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

PROJ IDCONC, IDART (SEL città = "ROMA" (CONCERTO))

PROJ E SEL NON COMMUTATIVI



Esercizio 4 – query 1

Trovare id del concerto (idconc) e id dell'artista (idart) dei concerti tenuti a Roma.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

Soluzione:

PROJ_{idconc,idart}(SEL_{città='Roma'}(Concerto))



Esercizio 4 – query 1

Trovare id del concerto (idconc) e id dell'artista (idart) dei concerti tenuti a Roma.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

Soluzione:

PROJ_{idconc,idart}(SEL_{città='Roma'}(Concerto))

Risultato:

idconc	idart
209	73
299	95

Esercizio 4 – query 2

Calcolare l'idconc di tutti i concerti presenti nella base di dati.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

$PROJ_{IDCONC} (CONCERTO) \cup PROJ_{IDCONC} (EVENTO)$



Esercizio 4 – query 2

Calcolare l'idconc di tutti i concerti presenti nella base di dati.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

Soluzione:

$\text{PROJ}_{\text{idconc}}(\text{Concerto}) \cup \text{PROJ}_{\text{idconc}}(\text{Evento})$



Esercizio 4 – query 2

Calcolare l'idconc di tutti i concerti presenti nella base di dati.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

Soluzione:

$\text{PROJ}_{\text{idconc}}(\text{Concerto}) \cup \text{PROJ}_{\text{idconc}}(\text{Evento})$

Risultato:

idconc
209
298
253
299
276
223

Esercizio 4 – query 3

Trovare id degli artisti (idart) che, secondo quanto riportato nella base di dati, non hanno mai tenuto concerti a Napoli.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

$PROJ_{IDART}(ARTISTA) - PROJ_{IDART}(SEL_{CITTÀ = 'NAPOLI'}(CONCERTO))$



Esercizio 4 – query 3

Trovare id degli artisti (idart) che, secondo quanto riportato nella base di dati, non hanno mai tenuto concerti a Napoli.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

Soluzione (assumendo che non possano esistere valori di idart nella relazione Concerto che non sono presenti nella relazione Artisti):

PROJ_{idart}(Artista) - PROJ_{idart}(SEL_{città='Napoli'}(Concerto))

Esercizio 4 – query 3

Trovare id degli artisti (idart) che, secondo quanto riportato nella base di dati, non hanno mai tenuto concerti a Napoli.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

Soluzione (assumendo che non possano esistere valori di idart nella relazione Concerto che non sono presenti nella relazione Artisti):

PROJ_{idart}(Artista) - PROJ_{idart}(SEL_{città='Napoli'}(Concerto))

Risultato:

idart
56
95
59



Esercizio 4 – query 3

Trovare id degli artisti (idart) che, secondo quanto riportato nella base di dati, non hanno mai tenuto concerti a Napoli.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

Soluzione (caso generale: non si assume che non possano esistere valori di idart nella relazione Concerto che non sono presenti nella relazione Artisti):

$(\text{PROJ}_{\text{idart}}(\text{Artista}) \cup \text{PROJ}_{\text{idart}}(\text{Concerto}))$

—

$\text{PROJ}_{\text{idart}}(\text{SEL}_{\text{città}='Napoli'}(\text{Concerto}))$



Il problema del “mondo chiuso”

La formulazione della query 2 nel testo dell’esercizio 4 recita:

“... secondo quanto riportato nella base di dati ...”

Una base di dati, in effetti, rappresenta un **mondo chiuso**, ossia un sistema in cui vale questo principio (closed world assumption):

“se una tupla non è presente nella base di dati, allora il fatto rappresentato da tale tuple è falso nel mondo corrispondente alla base di dati”.

Da ora in poi, quindi, eviteremo di specificare “... secondo quanto riportato nella base di dati ...”, perché lo riterremo implicito.

Nell’ambito dell’esercizio 4, questo principio implica, ad esempio, che stiamo interrogando un mondo in cui non esista alcun altro concerto che non sia un concerto con idconc presente nella base di dati (analogamente per gli artisti).

Si noti che esistono anche sistemi di rappresentazione dell’informazione che operano con un’assunzione di mondo aperto, ma questi esulano da quanto affrontato nel nostro corso.



Il problema della negazione in algebra relazionale

La query 2 dell'esercizio 4 ci consente di sottolineare una proprietà fondamentale dell'algebra relazionale: l'unico modo per esprimere condizioni di negazione (che non si riducano a semplici selezioni) è utilizzare l'operatore di differenza.

Da questo segue che un principio importante da tenere presente per scrivere query corrette è:

L'insieme degli elementi di A che **non** soddisfano la proprietà B è uguale all'insieme che otteniamo da A **togliendo** gli elementi che soddisfano la proprietà B:

$$\{ x \mid x \in A \wedge \neg B(x) \} = A - \{ x \mid B(x) \}$$

non c'è in algebra relazionale

c'è in algebra relazionale

Questo principio è esattamente quello che ci ha suggerito la soluzione alla query 2 dell'esercizio 4.



Join

Studiamo un operatore fondamentale: il join, che permette di correlare dati in relazioni diverse.

Esempio: corsi e linguaggi di programmazione

- Ogni docente insegna uno o più corsi, e nei corsi si insegnano i linguaggi di programmazione

1	Basi di dati
3	Reti
1	Prog. sw
2	Prog. sw

Basi di dati	SQL
Basi di dati	Java
Prog. sw	UML
Prog. sw	Java

- Quali docenti insegnano quali linguaggi?

1	SQL
1	UML
1	Java
2	UML
2	Java

con gli operatori che abbiamo finora illustrato non si può rispondere alla domanda e quindi non si può calcolare questo risultato



Join

- combinando selezione e proiezione, possiamo estrarre informazioni da **una** relazione
- non possiamo però correlare informazioni presenti in relazioni diverse
- il **join** è l'operatore più interessante dell'algebra relazionale
- come detto prima, permette di correlare dati in relazioni diverse
- esistono diverse versioni del join: join naturale, theta-join, equijoin, join esterno, ecc.



Join naturale

- operatore binario (generalizzabile secondo quanto specificato più avanti)
- produce un risultato
 - il cui schema ha l'unione (quindi con un risultato che non ha duplicati) degli attributi degli operandi
 - le cui tuple sono costruite ciascuna a partire da due tuple che si «combinano», una di un operando ed una di un altro operando

Join naturale: sintassi e semantica

- In queste slides, se X_1 e X_2 sono due insiemi, l'espressione X_1X_2 denota la loro unione
- Siano $R_1(X_1)$, $R_2(X_2)$ due schemi di relazioni
- $R_1 \text{ JOIN } R_2$ è una relazione su X_1X_2 il cui insieme di tuple è:

l'unione di X_1 e X_2

$\{ t \text{ su } X_1X_2 \mid \text{esistono due tuple } t_1 \in R_1 \text{ e } t_2 \in R_2 \text{ tali che } t[X_1] = t_1 \text{ e } t[X_2] = t_2 \}$

Join naturale: sintassi e semantica

- Definizione formale del join:

$R_1 \text{ JOIN } R_2$ è una relazione su $X_1 X_2$ il cui insieme di tuple è:

$$\{ t \text{ su } X_1 X_2 \mid \text{esistono } t_1 \in R_1 \text{ e } t_2 \in R_2 \\ \text{ tali che } t[X_1] = t_1 \text{ e } t[X_2] = t_2 \}$$

- Diciamo che $t_1 \in R_1$ e $t_2 \in R_2$ sono **combinabili** dal join naturale se $t_1[X_1 \cap X_2] = t_2[X_1 \cap X_2]$. Dalla definizione segue che ogni tupla nel join tra R_1 ed R_2 proviene da due tuple combinabili dal join. Ed in effetti $t_1 \in R_1$ e $t_2 \in R_2$ tali che $t_1[X_1 \cap X_2] = t_2[X_1 \cap X_2]$ si combinano per ottenere la tupla t tale che $t[X_1] = t_1$ e $t[X_2] = t_2$



Insegna

Join naturale: esempio

Tratta

Docente	Corso
1	BD
2	PS
3	Reti
1	PS

Corso	Ling
BD	SQL
BD	Java
PS	Java
PS	UML

Insegna join Tratta

L'insieme degli
attributi del risultato è
 $\{\text{Docente}, \text{Corso}, \text{Ling}\}$
 $=$
 $\{\text{Docente}, \text{Corso}\}$
 \cup
 $\{\text{Corso}, \text{Ling}\}$

Docente	Corso	Ling
1	BD	SQL
1	BD	Java
1	PS	Java
1	PS	UML
2	PS	Java
2	PS	UML



Insegna

Join naturale: esempio

Tratta

Docente	Corso
1	BD
2	PS
3	Reti
1	PS

Corso	Ling
BD	SQL
BD	Java
PS	Java
PS	UML

Insegna join Tratta

L'insieme degli attributi del risultato è
 $\{\text{Docente}, \text{Corso}, \text{Ling}\}$
 $=$
 $\{\text{Docente}, \text{Corso}\}$
 \cup
 $\{\text{Corso}, \text{Ling}\}$

Docente	Corso	Ling
1	BD	SQL
1	BD	Java
1	PS	Java
1	PS	UML
2	PS	Java
2	PS	UML



Insegna

Join naturale: esempio

Tratta

Docente	Corso
1	BD
2	PS
3	Reti
1	PS

Corso	Ling
BD	SQL
BD	Java
PS	Java
PS	UML

Insegna join Tratta

L'insieme degli attributi del risultato è
 $\{\text{Docente}, \text{Corso}, \text{Ling}\}$
 $=$
 $\{\text{Docente}, \text{Corso}\} \cup \{\text{Corso}, \text{Ling}\}$

Docente	Corso	Ling
1	BD	SQL
1	BD	Java
1	PS	Java
1	PS	UML
2	PS	Java
2	PS	UML



Insegna

Join naturale: esempio

Tratta

Docente	Corso
1	BD
2	PS
3	Reti
1	PS

Corso	Ling
BD	SQL
BD	Java
PS	Java
PS	UML

Insegna join Tratta

L'insieme degli
attributi del risultato è
 $\{\text{Docente}, \text{Corso}, \text{Ling}\}$
 $=$
 $\{\text{Docente}, \text{Corso}\}$
 \cup
 $\{\text{Corso}, \text{Ling}\}$

Docente	Corso	Ling
1	BD	SQL
1	BD	Java
1	PS	Java
1	PS	UML
2	PS	Java
2	PS	UML



Insegna

Join naturale: esempio

Tratta

Docente	Corso
1	BD
2	PS
3	Reti
1	PS

Corso	Ling
BD	SQL
BD	Java
PS	Java
PS	UML

Insegna join Tratta

L'insieme degli attributi del risultato è
 $\{\text{Docente}, \text{Corso}, \text{Ling}\}$
 $=$
 $\{\text{Docente}, \text{Corso}\} \cup \{\text{Corso}, \text{Ling}\}$

Docente	Corso	Ling
1	BD	SQL
1	BD	Java
1	PS	Java
1	PS	UML
2	PS	Java
2	PS	UML



Join completo

Un **join completo** è un join in cui ogni tupla contribuisce al risultato. Questo è un esempio di join naturale completo:

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
A	Mori
B	Bruni

Impiegato	Reparto	Capo
Rossi	A	Mori
Neri	B	Bruni
Bianchi	B	Bruni



Un join naturale non completo

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
B	Mori
C	Bruni

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori



Un join il cui risultato è l'insieme vuoto di tuple

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
D	Mori
C	Bruni

Impiegato	Reparto	Capo



Un join naturale completo con $n \times m$ tuple

Impiegato	Reparto
Rossi	B
Neri	B

Reparto	Capo
B	Mori
B	Bruni

Impiegato	Reparto	Capo
Rossi	B	Mori
Rossi	B	Bruni
Neri	B	Mori
Neri	B	Bruni



Cardinalità del join naturale

- $R_1(A,B), R_2(B,C)$
- In generale, il join di R_1 e R_2 contiene un numero di tuple compreso fra zero ed il prodotto di $|R_1|$ e $|R_2|$:
$$0 \leq |R_1 \text{ JOIN } R_2| \leq |R_1| \times |R_2|$$

Prodotto cartesiano

- Ricordiamo la definizione di join naturale:

$R_1 \text{ JOIN } R_2$ è una relazione su $X_1 X_2$ il cui insieme di tuple è:

$$\{ t \text{ su } X_1 X_2 \mid \text{esistono } t_1 \in R_1 \text{ e } t_2 \in R_2 \\ \text{tali che } t[X_1] = t_1 \text{ e } t[X_2] = t_2 \}$$

e questo significa che $t_1 \in R_1$ e $t_2 \in R_2$ tali che $t[X_1 \cap X_2] = t_2[X_1 \cap X_2]$ si combinano per ottenere la tupla t tale che $t[X_1] = t_1$ e $t[X_2] = t_2$. Ricordiamo che se t è una qualunque tupla, la sua restrizione sull'insieme vuoto di attributi è la tupla vuota. Si evince quindi che il join naturale su relazioni senza attributi in comune si riduce al prodotto cartesiano

- In questo caso il risultato contiene sempre un numero di tuple pari al prodotto delle cardinalità degli operandi (perché le tuple sono tutte combinabili a coppie)



Impiegato

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto

Codice	Capo
A	Mori
B	Bruni

Impiegato JOIN Reparto

Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Rossi	A	B	Bruni
Neri	B	A	Mori
Neri	B	B	Bruni
Bianchi	B	A	Mori
Bianchi	B	B	Bruni

Theta-join

- Siano R_1 ed R_2 due relazioni **senza attributi in comune** tali quindi che il loro join naturale corrisponda al prodotto cartesiano. Ma il prodotto cartesiano ha senso solo (o quasi solo) se combinato con una selezione:

$$SEL_{condizione} (R_1 \text{ JOIN } R_2)$$

- La combinazione delle due operazioni viene chiamata **theta-join**, richiede appunto che R_1 ed R_2 non abbiano attributi in comune e viene indicata con

$$R_1 \text{ JOIN}_{condizione} R_2$$

forma compatta di
 $SEL_{condizione} (R_1 \text{ JOIN } R_2)$

- La condizione di selezione è spesso una congiunzione (**AND**) di atomi di confronto $A_1 \vartheta A_2$ dove ϑ (da cui deriva il nome) è uno degli operatori di confronto ($=, >, <, \dots$).
- Se il solo operatore di confronto usato nella condizione ϑ è l'uguaglianza ($=$), allora si parla di **equi-join**.



Impiegato

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto

Codice	Capo
A	Mori
B	Bruni

Impiegato JOIN_{Reparto=Codice} Reparto

equi-join

Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Neri	B	B	Bruni
Bianchi	B	B	Bruni



Join naturale ed equi-join

Per riformulare un join naturale usando un equi-join occorre procedere a ridenominazioni.

Impiegato

Impiegato	Reparto
------------------	----------------

Reparto

Reparto	Capo
----------------	-------------

Join naturale: **Impiegato JOIN Reparto**

Equi-join:

PROJ_{Impiegato,Reparto,Capo} (

Impiegato JOIN_{Reparto=Codice} **REN**_{Codice ← Reparto} **(Reparto)**

Equi-join

)



Join naturale ed equi-join

Anche per riformulare un equi-join usando un join naturale occorre procedere a ridenominazioni.

Impiegato

Impiegato	Reparto
-----------	---------

Reparto

Codice	Capo
--------	------

Equi-join:

PROJ_{Impiegato,Reparto,Capo} (

Impiegato JOIN_{Reparto=Codice}**REN**_{Codice ← Reparto} (**Reparto**)
)

Join naturale:

Impiegato JOIN REN_{Reparto ← Codice}(**Reparto**)

Join naturale

Esercizio 4 – query 4

Trovare nome e tipo degli artisti che hanno tenuto almeno un concerto prima del 2000.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

$PROJ_{nome, tipo} (SEL_{ANNO < 2000} (CONCERTO JOIN ARTISTA))$

$PROJ_{nome, tipo} (ARTISTA JOIN_{ANNO < 2000 \text{ AND } Q=IDART}$

$REN_{Q \leftarrow IDART} (CONCERTO))$



Esercizio 4 – query 4

Trovare nome e tipo degli artisti che hanno tenuto almeno un concerto prima del 2000.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

Soluzione:

PROJ_{nome, tipo}(Artista JOIN SEL_{anno < 2000}(Concerto))



Esercizio 4 – query 4

Trovare nome e tipo degli artisti che hanno tenuto almeno un concerto prima del 2000.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

join naturale su attributo
comune idart

Soluzione:

PROJ_{nome, tipo}(Artista JOIN SEL_{anno < 2000}(Concerto))

Gli attributi del risultato del join naturale sono:
idart, nome, tipo, idconc, anno, città

Risultato:

nome	tipo
RolSto	band
S&G	duo

Esercizio 4 – query 5

Trovare l'id e nome degli artisti che hanno tenuto almeno un concerto svoltosi in una città diversa da Roma nell'ambito di un evento nel quale ha ottenuto un voto ~~minore di 6~~. > 4

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

PROJ_{nome, idart} (ARTISTA JOIN

(SEL_{voto > 4, città ≠ "ROMA"} (CONCERTO JOIN EVENTO)))

Esercizio 4 – query 5

Trovare l'id e nome degli artisti che hanno tenuto almeno un concerto svoltosi in una città diversa da Roma nell'ambito di un evento nel quale ha ottenuto un voto minore di 6.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

Soluzione:

```
PROJidart,nome( (Artista JOIN SELcittà!='Roma'(Concerto))  
                JOIN  
                SELvoto>4(Evento)  
            )
```

join naturale su attributo
comune idart

join naturale su
attributo
comune idconc

Esercizio 4 – query 5

Trovare l'id e nome degli artisti che hanno tenuto almeno un concerto svoltosi in una città diversa da Roma nell'ambito di un evento nel quale ha ottenuto un voto minore di 6.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

Soluzione:

**PROJ_{idart,nome} ((Artista JOIN SEL_{città!='Roma'}(Concerto))
JOIN
SEL_{voto>4}(Evento))**

join naturale su attributo comune idart. Gli attributi del risultato del join naturale sono:
idart, nome, tipo, idconc, anno, città

join naturale su attributo comune idconc. Gli attributi sono:
idart, nome, tipo, idconc, anno, città, ideve, voto

Risultato:

idart	nome
59	Neil

Esercizio 4 – query 6

Trovare l'id degli artisti che hanno tenuto almeno due concerti nella stessa città in anni diversi.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

$$\text{PROJ}_{\text{IDART}} \left(\text{CONCERTO JOIN}_{\text{IDART} = a \text{ AND } \text{CITTÀ} = x \text{ AND } \text{ANNO} \neq n} \right)$$

$$\text{REN}_{c \leftarrow \text{IDCONC}, a \leftarrow \text{IDART}, n \leftarrow \text{ANNO}, x \leftarrow \text{CITTÀ} (\text{CONCERTO})}$$



Esercizio 4 – query 6

Trovare l'id degli artisti che hanno tenuto almeno due concerti nella stessa città in anni diversi.

Concerto(idconc, idart, anno, città)

Artista(idart, nome, tipo)

Evento(ideve, idconc, voto)

Soluzione: si deve calcolare il join di Concerto con se stesso

PROJ_{idart} (**REN**_{i1←idconc, t1←idart, a1←anno, c1←città} (**Concerto**)

Theta-join

JOIN_{idart=t1 and città=c1 and anno<>a1}

Concerto

)

Gli attributi del risultato sono:
i1, t1, a1, c1, idconc, idart, anno, città



Il self-join in algebra

La query 6 dell'esercizio appena visto contiene un equi-join in cui la stessa relazione (in quel caso Concerto) compare sia come operando sinistro sia come operando destro. Quando una operazione di join coinvolge la stessa relazione due volte, si definisce un **self-join**.

Ovviamente, siccome l'equi-join prevede che i due operandi non abbiano attributi in comune, quando si applica un self-join in uno dei due operandi occorre ridenominare tutti gli attributi della relazione, cosa che abbiamo fatto nell'esercizio menzionato.



Esercizio 5 (con soluzioni)

Impiegato

Matricola	Nome	Età	Stipendio
7309	Rossi	34	45
5998	Bianchi	37	38
9553	Neri	42	35
5698	Bruni	43	42
4076	Mori	45	50
8123	Lupi	46	60

Supervisione

Impiegato	Capo
7309	5698
5998	5698
9553	4076
5698	4076
4076	8123




Trovare matricola, nome, età e stipendio degli impiegati che guadagnano più di 40 milioni.

Soluzione:

SEL_{Stipendio>40}(Impiegato)

Matricola	Nome	Età	Stipendio
7309	Rossi	34	45
5698	Bruni	43	42
4076	Mori	45	50
8123	Lupi	46	60



Trovare matricola, nome ed età degli impiegati che guadagnano più di 40 milioni.

PROJ MATRICOLA, NOME, ETÀ (SEL STIPENDIO > 40 (IMPIEGATO))



Trovare matricola, nome ed età degli impiegati che guadagnano più di 40 milioni.

Soluzione:

PROJ_{Matricola, Nome, Età} (**SEL**_{Stipendio>40}(Impiegato))

Matricola	Nome	Età
7309	Rossi	34
5698	Bruni	43
4076	Mori	45
8123	Lupi	46



Trovare le matricole dei capi di quegli impiegati che guadagnano più di 40 milioni.

Impiegato

Matricola

Nome

Età

Stipendio

Supervisione

Impiegato

Capo

$$\text{PROJ}_{\text{CAPO}}(\text{SUPERVISIONE JOIN}_{\text{IMPI}=\text{MATR}}(\text{SEL}_{\text{STIP}>40}(\text{IMPIEGATO})))$$



Trovare le matricole dei capi di quegli impiegati che guadagnano più di 40 milioni.

Impiegato

Matricola	Nome	Età	Stipendio
-----------	------	-----	-----------

Supervisione

Impiegato	Capo
-----------	------

Soluzione:

PROJ_{Capo} (

Supervisione JOIN_{Impiegato=Matricola} (SEL_{Stipendio>40}(Impiegato))
)

Trovare nome e stipendio dei capi di quegli impiegati che guadagnano più di 40 milioni.

Impiegato

Matricola	Nome	Età	Stipendio
-----------	------	-----	-----------

Supervisione

Impiegato	Capo
-----------	------

$PROJ_{nome, stipendio} (IMPIEGATO \Join_{MATRICOLA = CAPO}$

$PROJ_{CAPO} (SUPERVISIONE \Join_{IMPI = MATR} (SEL_{STIP > 40} (IMPIEGATO))))$



Trovare nome e stipendio dei capi di quegli impiegati che guadagnano più di 40 milioni.

Impiegato

Matricola	Nome	Età	Stipendio
------------------	-------------	------------	------------------

Supervisione

Impiegato	Capo
------------------	-------------

Soluzione:

```
PROJNome,Stipendio (  
  Impiegato JOINMatricola=Capo  
    PROJCapo(Supervisione  
      JOINImpiegato=Matricola (SELStipendio>40(Impiegato))))
```



Trovare gli impiegati che guadagnano più del proprio capo, mostrando matricola, nome e stipendio dell'impiegato e del capo.

Impiegato

Matricola

Nome

Età

Stipendio

Supervisione

Impiegato

Capo

$PROJ_{MATR, NOME, STIPENDIO, m, n, s} (SEL_{STIP > s} ($

$REN_{m, n, s, e} \leftarrow MATR, NOM, STIP, ETÀ (IMPIEGATO)$

$JOIN_{m=CAPO} (SUPERVISIONE JOIN_{IMP=MATR} IMPIEGATO)))$



Trovare gli impiegati che guadagnano più del proprio capo, mostrando matricola, nome e stipendio dell'impiegato e del capo.

Impiegato

Matricola	Nome	Età	Stipendio
------------------	-------------	------------	------------------

Supervisione

Impiegato	Capo
------------------	-------------

Soluzione:

```
PROJMatr, Nome, Stip, MatrC, NomeC, StipC
  (SELStipendio > StipC
RENMatrC, NomeC, StipC, EtàC ← Matr, Nome, Stip, Età (Impiegato)
  JOINMatrC = Capo
  (Supervisione JOINImpiegato = Matricola Impiegato)))
```



Trovare la matricola degli impiegati che **non** hanno un capo

Impiegato

Matricola	Nome	Età	Stipendio
-----------	------	-----	-----------

Supervisione

Impiegato	Capo
-----------	------

PROJ_{MATRICOLOLA} (IMPIEGATO) -

PROJ_{MATRICOLOLA} (SUPERVISIONE JOIN_{MATR=IMPIE} IMPIEGATO)

NO



Trovare la matricola degli impiegati che **non** hanno un capo

Impiegato

Matricola	Nome	Età	Stipendio
-----------	------	-----	-----------

Supervisione

Impiegato	Capo
-----------	------

Idea per la soluzione: per trovare gli elementi di un insieme S che **non** soddisfano la condizione A possiamo **togliere** da S gli elementi che soddisfano A:

$$\{ x \mid x \in S \wedge \neg A(x) \} = S - \{ x \mid A(x) \}$$

non c'è in algebra relazionale

c'è in algebra relazionale



Trovare la matricola degli impiegati che **non** hanno un capo

Impiegato

Matricola	Nome	Età	Stipendio
-----------	------	-----	-----------

Supervisione

Impiegato	Capo
-----------	------

Soluzione:

PROJ_{Matricola} (**Impiegato**)

—

REN_{Matricola} ← **Impiegato**(**PROJ**_{Impiegato} (**Supervisione**))



Trovare matricola ed età degli impiegati che **non** hanno un capo

Impiegato

Matricola	Nome	Età	Stipendio
-----------	------	-----	-----------

Supervisione

Impiegato	Capo
-----------	------

PROJ_{MATRICOLO, ETÀ} (IMPIEGATO JOIN

(PROJ_{MATRICOLO} (IMPIEGATO) - REN_{MATRICOLO ← IMPIEGATO}

(PROJ_{IMPIEGATO} (SUPERVISIONE))))



Trovare matricola ed età degli impiegati che **non** hanno un capo

Impiegato	Matricola	Nome	Età	Stipendio
------------------	------------------	-------------	------------	------------------

Supervisione	Impiegato	Capo
---------------------	------------------	-------------

Soluzione:

**PROJ_{Matricola, Età}(Impiegato
JOIN**

(PROJ_{Matricola} (Impiegato)

—

REN_{Matricola} ← Impiegato (PROJ_{Impiegato} (Supervisione))))

Trovare le matricole dei capi i cui impiegati guadagnano **tutti** più di 40 milioni.

Impiegato

Matricola	Nome	Età	Stipendio
-----------	------	-----	-----------

Supervisione

Impiegato	Capo
-----------	------

PROJ_{CAPO} (SUPERVISIONE) -

PROJ_{CAPO} (SUPERVISIONE JOIN_{IMP=MATR}

(SEL_{STIP > 40} (IMPIEGATO)))



Trovare le matricole dei capi i cui impiegati guadagnano **tutti** più di 40 milioni.

Impiegato

Matricola

Nome

Età

Stipendio

Supervisione

Impiegato

Capo

Idea per la soluzione: per trovare i capi i cui impiegati soddisfano **tutti** la condizione A possiamo **togliere** dall'insieme dei capi quelli che hanno **almeno** un impiegato che **non** soddisfa A

$$\{ x \mid x \in S \wedge \forall y (R(x,y) \rightarrow A(y)) \} = S - \{ x \mid \exists y R(x,y) \wedge \neg A(y) \}$$

non ci sono in algebra relazionale

ci sono in algebra relazionale



Trovare le matricole dei capi i cui impiegati guadagnano **tutti** più di 40 milioni.

Impiegato

Matricola	Nome	Età	Stipendio
-----------	------	-----	-----------

Supervisione

Impiegato	Capo
-----------	------

Soluzione:

**PROJ_{Capo} (Supervisione) -
PROJ_{Capo} (Supervisione
JOIN Impiegato=Matricola
(SEL_{Stipendio} ≤ 40 (Impiegato)))**



Informazione incompleta

- Il modello relazionale impone ai dati una struttura rigida:
 - l'informazione è rappresentata per mezzo di tuple
 - le tuple ammesse sono dettate dagli schemi di relazione
- Nella pratica, però, i dati disponibili possono non corrispondere esattamente al formato previsto, per varie ragioni

Esempio:

- di Firenze non conosciamo l'indirizzo della prefettura
- Tivoli non è provincia: non ha prefettura
- Prato è “nuova” provincia: ha la prefettura?

Prefettura

città	indirizzo
Roma	Via IV Novembre
Firenze	
Tivoli	
Prato	



Informazione incompleta: soluzioni?

Spesso si utilizzano valori ordinari del dominio (0, stringa nulla, “999”, etc) per rappresentare informazione mancante. Questo però è un errore, per vari motivi:

- potrebbero non esistere valori “non utilizzati”
- valori “non utilizzati” fino ad un certo momento potrebbero diventare significativi in seguito
- in fase di utilizzo (ad esempio, nei programmi) è necessario ogni volta tener conto del “significato” di questi valori speciali e questo richiede di mettere d’accordo diversi progettisti del software e programmatori, cosa non semplice o impossibile nella pratica



Informazione incompleta nel modello relazionale

- Nel modello relazionale si adotta una tecnica rudimentale, ma per certi versi efficace:
 - viene introdotto il cosiddetto **valore nullo** (denotato da **NULL**): esso è un valore che il sistema riconoscere essere particolare e che denota l'assenza di un valore del dominio (il valore nullo non è un valore del dominio, anche se può comparire come valore di qualunque attributo definito su qualunque dominio)
- Formalmente, è sufficiente estendere il concetto di tupla: se t è una tupla, allora $t[A]$, per ogni attributo A , è un valore del dominio $\text{dom}(A)$ oppure il valore nullo **NULL**
- Al momento di definire uno schema di basi di dati, si possono poi imporre restrizioni sulla presenza di valori nulli nei vari attributi delle relazioni



Interpretazioni del valore nullo

- (almeno) tre casi differenti
 - valore sconosciuto: esiste un valore del dominio, ma non è noto (nell'esempio precedente: Firenze)
 - valore inesistente: non esiste un valore del dominio (nell'esempio precedente: Tivoli)
 - valore senza informazione: non è noto se esista o meno un valore del dominio (nell'esempio precedente: Prato)
- I DBMS non distinguono i tipi di valore nullo (e quindi implicitamente adottano l'interpretazione “**senza informazione**“)



Il valore nullo nell'algebra relazionale

Una condizione in un'espressione nell'algebra è vera **solo per valori non nulli**

Esempio sulla selezione:

*NULL NON SODDISFA
ALCUNA CONDIZIONE
↪ SEMPRE FALSO*

Impiegato

Matricola	Cognome	Filiale	Età
7309	Rossi	Roma	32
5998	Neri	Milano	45
9553	Bruni	Milano	NULL



SEL $Età > 40$ (Impiegato)



Condizioni speciali sul valore nullo

- per riferirsi ai valori nulli esistono forme apposite di condizioni:

IS NULL

IS NOT NULL

- Esempio sulla selezione:

SEL $Età > 40$ (Impiegati)

la condizione atomica è vera solo per valori non nulli

- L'espressione

SEL $Età > 40$ OR $Età$ IS NULL (Impiegati)

è vera anche per le tuple in cui $Età$ contiene il valore nullo



Impiegato

Matricola	Cognome	Filiale	Età
5998	Neri	Milano	45
9553	Bruni	Milano	NULL

SEL (Età > 40) OR (Età IS NULL) **(Impiegato)**



Join: un'osservazione

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
B	Mori
C	Bruni

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

Come visto prima, alcune tuple possono non contribuire al risultato nel join, ossia vengono "tagliate fuori» dal risultato



Join esterno

- Il **join naturale esterno** estende, con valori nulli, le tuple che verrebbero tagliate fuori da un join (interno)
- esiste in tre versioni:
 - **sinistro**: mantiene tutte le tuple del primo operando, estendendole con valori nulli, se necessario
 - **destro**: ... del secondo operando ...
 - **completo**: ... di entrambi gli operandi ...



Join naturale esterno sinistro

Impiegato

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto

Reparto	Capo
B	Mori
C	Bruni

Impiegato JOIN Reparto

Supponiamo di voler sapere di ogni impiegato in quale reparto lavora e chi è il capo del reparto.

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

Una ovvia soluzione può essere il calcolo del join naturale

Ma in questo modo non otteniamo le informazioni sull'impiegato Rossi e sul reparto C



Join naturale esterno sinistro

Impiegato

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto

Reparto	Capo
B	Mori
C	Bruni

Supponiamo di voler sapere di ogni impiegato in quale reparto lavora e chi è il capo del reparto e non vogliamo perdere informazioni sugli impiegati

La soluzione è il join esterno sinistro

Impiegato JOIN_{LEFT} Reparto

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori
Rossi	A	NULL



Join naturale esterno destro

Impiegato

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto

Reparto	Capo
B	Mori
C	Bruni

Supponiamo di voler sapere di ogni impiegato in quale reparto lavora e chi è il capo del reparto e non vogliamo perdere informazioni sui reparti e i loro capi

La soluzione è il join esterno destro

Impiegato JOIN_{RIGHT} Reparto

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori
NULL	C	Bruni



Join naturale esterno completo

Impiegato

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto

Reparto	Capo
B	Mori
C	Bruni

Supponiamo di voler sapere di ogni impiegato in quale reparto lavora e chi è il capo del reparto e non vogliamo perdere informazioni né sugli impiegati né sui reparti

La soluzione è il join esterno completo

Impiegato JOIN_{FULL} Reparto

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori
Rossi	A	NULL
NULL	C	Bruni



Ultima osservazione sull'algebra: equivalenza di espressioni

- Due espressioni sono **equivalenti** se producono lo stesso risultato qualunque sia l'istanza della base di dati sulla quale vengono valutate
- L'equivalenza è importante in pratica perché i DBMS cercano di eseguire espressioni equivalenti a quelle date, ma meno "costose" della espressione originaria

Un'equivalenza importante per i DBMS

- Effettuare le selezioni il prima possibile (**push selections**)

Esempio: se A è attributo di R_2

$$SEL_{A=10}(R_1 \text{ JOIN } R_2) = R_1 \text{ JOIN } SEL_{A=10}(R_2)$$

- Le selezioni tipicamente riducono in modo significativo la dimensione del risultato intermedio (e quindi il costo dell'operazione)



2. Il modello relazionale

2.3 Vincoli di integrità

- | | |
|------------|--------------------------------|
| 2.1 | struttura dei dati relazionali |
| 2.2 | algebra relazionale |
| 2.3 | vincoli di integrità |



Vincoli di integrità: introduzione

Esistono istanze di basi di dati che, pur sintatticamente corrette, non rappresentano dati che descrivono situazioni possibili per l'applicazione di interesse.

Studente

Matricola	Cognome	Nome	Nascita
276545	Rossi	Maria	23/04/1968
276545	Neri	Anna	23/04/1972
788854	Verdi	Fabio	12/02/1972

Esame

Studente	Voto	Lode	Corso
276545	28	e lode	01
276545	32		02
788854	23		03
200768	30	e lode	03

Corso

Codice	Titolo	Docente
01	Analisi	Giani
03	NULL	NULL
02	Chimica	Belli

Vincolo di integrità

Definizione di vincolo di integrità

- 1° APPUNTO**
- Un vincolo di integrità (o semplicemente vincolo) è una condizione che si esprime a livello di schema e che si intende debba essere soddisfatta da tutte le istanze della base di dati, perché individua una condizione necessaria per tutte quelle istanze della base di dati che rappresentano situazioni corrette per l'applicazione
 - Ogni vincolo può essere visto come una funzione booleana (o un predicato) che associa ad ogni istanza della base di dati il valore **vero** (nel caso in cui il vincolo sia soddisfatto) o **falso** (altrimenti)
- Ad ogni schema di basi di dati si associa un insieme di vincoli e si considerano **corrette** (diciamo anche lecite, legali, valide, ammissibili) solo le istanze che soddisfano tutti i vincoli



Vincoli di integrità: motivazioni

- risultano utili al fine di descrivere la realtà di interesse in modo più accurato di quanto le sole strutture permettano;
- forniscono un contributo verso la “qualità dei dati”
- costituiscono uno strumento di ausilio alla progettazione
- sono utilizzati dal sistema nella scelta della strategia di esecuzione delle interrogazioni

Nota:

- non tutte le proprietà di interesse sono rappresentabili per mezzo di vincoli esprimibili direttamente



Vincoli di integrità: classificazione

- Intrarelazionali → SINGOLA RELAZIONE
 - di tupla
 - di dominio
 - di chiave
 - altri (che per il momento ignoriamo)
- Interrelazionali → TRA RELAZIONI DIVERSE
 - di integrità referenziale (anche dette di foreign key)
 - di inclusione
 - altri (che per il momento ignoriamo)

Vincoli intrarelazionali: vincoli di tupla

- Esprimono **condizioni sui valori di ciascuna tupla** di una relazione, indipendentemente dalle altre tuple
- Un vincolo di tupla su una relazione R si esprime come un'espressione booleana (con AND, OR e NOT) costruita su atomi che confrontano valori di attributi (della relazione R) o espressioni aritmetiche su di essi
- Un vincolo di tupla che coinvolge un solo attributo si dice **vincolo di dominio**

Esempio di vincolo di tupla: in una tabella relativa agli stipendi, in cui appaiano gli attributi Lordo, Ritenute e Netto, si esprimerà la seguente condizione che li lega

$$\text{Lordo} = (\text{Ritenute} + \text{Netto})$$



Esercizio 6

Sulla relazione Esame(Studente,Voto,Lode,Corso), esprimere i seguenti vincoli:

$$VOTO \geq 18 \quad AND \quad VOTO \leq 30$$

1.il vincolo che stabilisce che ogni valore intero che si trova nell'attributo Voto è compreso tra 18 e 30.

$$VOTO \neq 30 \rightarrow LODE \neq "e lode"$$

2.Il vincolo che stabilisce che in ogni tupla di esame, nell'attributo Lode compare la stringa "e lode" solo se il corrispondente valore dell'attributo Voto è 30.

$$LODE = '' \quad OR \quad VOTO < 30$$

3.Il vincolo che stabilisce che in ogni tupla di esame, nell'attributo Lode compare la stringa vuota se il valore dell'attributo Voto è diverso da 30.

$$LODE = "e lode" \rightarrow VOTO = 30$$



Esercizio 6: soluzione

Si noti che la seconda condizione equivale a: se il voto non è 30, allora il valore dell'attributo Lode non è "lode"; questa condizione a sua volta equivale a: il voto è 30 oppure il valore dell'attributo Lode non è "lode".

Si noti che la terza condizione equivale a: se il voto è diverso da 30, allora il valore dell'attributo Voto è "".

Soluzioni:

- | | |
|-------------------------------|--|
| <i>1. Vincolo di dominio:</i> | $\text{Voto} \geq 18 \text{ AND } \text{Voto} \leq 30$ |
| <i>2. Vincolo di tupla:</i> | $(\text{NOT } \text{Lode} = \text{"e lode"}) \text{ OR } \text{Voto} = 30$ |
| <i>3. Vincolo di tupla:</i> | $\text{Voto} = 30 \text{ OR } \text{Lode} = \text{" "}$ |



Vincoli intrarelazionali: vincoli di chiave

Studente	Matricola	Cognome	Nome	Corso	Nascita
	27655	Rossi	Mario	Ing Inf	5/12/78
	78763	Rossi	Mario	Ing Inf	3/11/76
	65432	Neri	Piero	Ing Mecc	10/7/79
	87654	Neri	Mario	Ing Inf	3/11/76
	67653	Rossi	Piero	Ing Mecc	5/12/78

- il numero di matricola identifica gli studenti, nel senso che, qualunque sia l'istanza della relazione **Studente**, non ci possono essere due tuple in tale istanza con lo stesso valore sull'attributo **Matricola**

Vedremo che questa condizione si impone definendo un vincolo di chiave sulla relazione.

Vincoli intrarelazionali: le condizioni di superchiave e chiave

Se K è un insieme non vuoto di attributi di una relazione, si dice che K **soddisfa la condizione di superchiave in quella relazione** se non esistono due tuple t_1 e t_2 della relazione tali che i valori che t_1 ha negli attributi in K sono gli stessi che t_2 ha in tali attributi.

Più precisamente:

- Sia R uno schema di relazione con insieme di attributi X , sia K un sottoinsieme non vuoto di X , e sia r una istanza di R
- Sia $K = \{A_1, \dots, A_n\}$. Allora K soddisfa la **condizione di superchiave** in r se r non contiene due tuple distinte t_1 e t_2 tali che $t_1[A_1] = t_2[A_1], \dots, t_1[A_n] = t_2[A_n]$. In questo caso si dice che K è una **superchiave in r**
- K soddisfa invece la **condizione di chiave** in r (nel qual caso diciamo che K è una **chiave in r**) se K è una superchiave minimale in r , ossia se K è una superchiave in r e se nessun sottoinsieme proprio di K è una superchiave in r



Il concetto di superchiave: esempi

R

Matricola	Cognome	Nome	Corso	Nascita
27655	Rossi	Mario	Ing Inf	5/12/78
78763	Rossi	Mario	Ing Inf	3/11/76
65432	Neri	Piero	Ing Mecc	10/7/79
87654	Neri	Mario	Ing Inf	3/11/76
67653	Rossi	Piero	Ing Mecc	5/12/78

- **Matricola** soddisfa la condizione di superchiave in R (cioè è superchiave in R), infatti non esistono due tuple in R con il valore di Matricola uguale
- Cognome non soddisfa la condizione di superchiave in R, e lo stesso vale per Nome e anche per Nascita
- **Cognome, Nome, Nascita** è superchiave in R, infatti non esistono due tuple in R con il valore della terna <Cognome, Nome, Nascita> uguale



Il concetto di chiave: esempi

R

Matricola	Cognome	Nome	Corso	Nascita
27655	Rossi	Mario	Ing Inf	5/12/78
78763	Rossi	Mario	Ing Inf	3/11/76
65432	Neri	Piero	Ing Mecc	10/7/79
87654	Neri	Mario	Ing Inf	3/11/76
67653	Rossi	Piero	Ing Mecc	5/12/78

- **Matricola** soddisfa la condizione di chiave in R (cioè è chiave in R), infatti:
 - Matricola è superchiave in R
 - contiene un solo attributo e quindi è minimale
- **Cognome, Nome, Nascita** soddisfa la condizione di chiave in R, infatti:
 - l'insieme Cognome, Nome, Nascita è superchiave in R
 - nessuno dei suoi sottoinsiemi è superchiave
- Cognome, Nome, Nascita, Corso è superchiave, ma non soddisfa la condizione di chiave, ovvero non è una chiave



Esistenza di superchiavi e chiavi

- poiché le relazioni sono insiemi, una relazione non può contenere tuple uguali fra loro:
 - ne segue che in ogni relazione, l'insieme di tutti gli attributi (insieme che per definizione è non vuoto) su cui essa è definita soddisfa il vincolo di superchiave
- poiché l'insieme di tutti gli attributi è una superchiave in ogni relazione, ogni relazione ha almeno una superchiave
- siccome ogni relazione ha almeno una superchiave, ci sarà almeno un sottoinsieme di tale superchiave che è minimale; ne segue che per ogni relazione esiste almeno un insieme di attributi che soddisfano la **condizione di chiave**



Vincolo di superchiave

- Un **vincolo di superchiave** per K su uno schema di relazione R è un'asserzione che specifica che l'insieme K non vuoto di attributi è una superchiave in **ogni istanza** della relazione R ; in altre parole, in nessuna istanza di R esistono due tuple della relazione R che coincidono negli attributi in K
- Ad esempio, se sullo schema di relazione $R(A,B,C,D)$ dichiaro un vincolo di superchiave per $\{A,B\}$, sto asserendo che l'insieme degli attributi $\{A,B\}$ è una superchiave (cioè soddisfa la condizione di superchiave) in ogni istanza della relazione R
- Ovviamente su uno schema di relazione si possono dichiarare più vincoli di chiave



Vincolo di chiave

- Un **vincolo di chiave** per K su uno schema di relazione R è un'asserzione che specifica che l'insieme K non vuoto di attributi di R gode di queste proprietà:
 1. K è una superchiave **in ogni istanza** della relazione R ;
 2. per nessun sottoinsieme proprio non vuoto di K vale la condizione 1.
- Ad esempio, se sullo schema di relazione $R(A,B,C,D)$ dichiaro un vincolo di chiave per $\{A,B\}$, sto asserendo che
 1. $\{A,B\}$ è una superchiave in ogni istanza della relazione R ;
 2. né $\{A\}$ e né $\{B\}$ è una superchiave in ogni istanza di R .
- Ovviamente su uno schema di relazione si possono dichiarare più vincoli di chiave



Individuazione delle chiavi

Come si individuano i vincoli di chiave per uno schema di relazione?

- considerando le proprietà che i dati devono soddisfare nell'applicazione (il “frammento di mondo reale di interesse”)
- notando per quali insiemi di attributi vale la proprietà che, per ogni istanza della base di dati, non possono esistere due tuple con gli stessi valori per gli attributi di questi insiemi
- e individuando i sottoinsiemi minimali di tali insiemi che conservano la proprietà suddetta

Esempio:

Su Studenti(Matricola, Cognome, Nome, Corso, Nascita) deve essere definito il vincolo di chiave che asserisce che Matricola è una chiave per ogni relazione dello schema



Chiavi e valori nulli

Si noti che un vincolo di chiave può riguardare attributi che contengono valori nulli.

Esempio: supponiamo di avere definito sullo schema S il vincolo di chiave per $\{A,B\}$ e consideriamo la seguente istanza R di S .

R	A	B	C
	a1	b1	c1
	a1	NULL	c1
	a2	b2	d
	a3	NULL	d

Ricordando che il valore NULL non soddisfa mai alcuna condizione (compresa l'uguaglianza), è facile verificare che l'istanza di relazione mostrata qui sopra soddisfa il vincolo di chiave, perché non esistono due tuple in questa istanza con gli stessi valori per gli attributi A e B.

Chiavi e valori nulli

Abbiamo appena visto che il concetto di chiave ha perfettamente senso anche in presenza di valori nulli. Tuttavia, in presenza di valori nulli, i valori degli attributi che formano la chiave:

- non permettono di identificare le tuple
- né permettono di realizzare facilmente i riferimenti da altre relazioni

ed entrambe queste proprietà possono essere problematiche.

Studente

Matricola	Cognome	Nome	Corso	Nascita
NULL	Neri	Mario	Ing Inf	5/12/78
78763	Rossi	Mario	Ing Civile	3/11/76
65432	Bianchi	Piero	Ing Mecc	10/7/79
87654	Neri	Mario	Ing Inf	NULL
NULL	Bianchi	Mario	Ing Inf	5/12/78

Nell'esempio qui sopra, la relazione soddisfa il vincolo che asserisce che l'attributo Matricola è una chiave, ma tale attributo non può essere usato per identificare le tuple della relazione colorate in rosa.

Chiave primaria

- La presenza di valori nulli nelle chiavi può quindi essere problematica
- Soluzione pratica: per ogni schema di relazione dichiariamo un vincolo di chiave primaria, ossia scegliamo una particolare chiave (la **chiave primaria**) su cui non ammettiamo valori nulli.
- Notazione per la chiave primaria: gli attributi che la compongono sono sottolineati

<u>Matricola</u>	Cognome	Nome	Corso	Nascita
27655	Rossi	Mario	Ing Inf	5/12/78
78763	Rossi	Mario	Ing Civile	3/11/76
65432	Neri	Piero	Ing Mecc	10/7/79
87654	Neri	Mario	Ing Inf	NULL
67653	Rossi	Piero	NULL	5/12/78



Vincolo di chiave primaria

- Un **vincolo di chiave primaria** su uno schema S di relazione è un'asserzione che specifica che K (insieme di attributi in S) è la chiave primaria della relazione, cioè che
 - K è una chiave per S e
 - in tutte le istanze di D non si ammettono i valori nulli per gli attributi in K
- Un solo vincolo di chiave primaria è ammessa per una relazione.
- Si può dimostrare che (α) se K è la chiave primaria di una relazione R , allora per ogni istanza r di R , prese due tuple t_1 e t_2 in r , si ha $t_1[K] \neq t_2[K]$ e questo implica che K può essere usato per identificare tutte le tuple di R
- Si noti che (α) non è necessariamente vera se K è una chiave non primaria, a causa della possibile presenza di valori nulli – la condizione in quel caso è che non esistono due tuple t_1 e t_2 in r tali che $t_1[K] = t_2[K]$



Esercizio 7

Partita

Codice	SqCasa	SqTrasferta	GoalCasa	GoalTrasf	Schedina

Non ci sono due partite con lo stesso codice. Gli attributi **SqCasa** e **SqTrasferta** dicono chi sono le squadre che hanno giocato la partita. **GoalCasa** dice quanti goal sono stati segnati dalla squadra di casa, mentre **GoalTrasf** quanti goal sono stati segnati dalla squadra in trasferta. **Schedina** codifica il risultato della partita in termini di 1, 2 o X. I domini degli attributi sono specificati come segue:

- **Codice**, **SqCasa**, **SqTrasferta** sono stringhe
- **GoalCasa** e **GoalTrasf** sono interi
- **Schedina** è un carattere.

Quali vincoli di integrità intrarelazionali definireste su questo schema?



Esercizio 7: soluzione

Partita

Codice	SqCasa	SqTrasferta	GoalCasa	GoalTrasf	Schedina

Questi sono i vincoli di integrità intrarelazionali da definire sullo schema:

- Codice è chiave primaria della relazione Partita – **vincolo di chiave primaria**
- SqCasa \neq SqTrasferta – **vincolo di tupla**
- GoalCasa ≥ 0 – **vincolo di dominio**
- GoalTrasf ≥ 0 – **vincolo di dominio**
- Schedina = '1' OR Schedina = '2' OR Schedina = 'X' – **vincolo di dominio**
- (NOT Schedina = '1') OR (GoalCasa > GoalTrasf) – **vincolo di tupla**
- (NOT Schedina = '2') OR (GoalCasa < GoalTrasf) – **vincolo di tupla**
- (NOT Schedina = 'X') OR (GoalCasa = GoalTrasf) – **vincolo di tupla**

Vincoli interrelazionali: integrità referenziale

- Informazioni in relazioni diverse sono correlate attraverso valori comuni, in particolare attraverso valori delle chiavi (primarie, di solito – ma non necessariamente). Il meccanismo con cui si realizza questa idea è il vincolo di integrità referenziale
- Un **vincolo di integrità referenziale** (detto anche vincolo di “**foreign key**”) fra una sequenza non vuota X di n attributi di una relazione R_1 ed una sequenza Y di n attributi **che formano una chiave** di una relazione R_2 è una asserzione che impone che per ogni istanza B della base di dati ogni combinazione di valori su X che non contengono NULL e che sono presenti nella istanza di R_1 in B compaia come combinazione di valori su Y nella istanza di R_2 in B
- Se la sequenza Y è chiave primaria di R_2 , si dice anche che il **vincolo di integrità referenziale sussiste fra X di R_1 e R_2**



Vincoli di integrità referenziale: esempio

<u>Codice</u>	<u>Comune</u>	<u>Data</u>	<u>Vigile</u>	<u>Targa</u>
34321	Roma	1/2/95	3987	39548K
53524	Milano	4/3/95	3295	E39548
64521	Napoli	5/4/96	3295	H39548
73321	Roma	5/2/98	9345	H39548

Infrazioni

*Ogni valore diverso da null
che compare nell'attributo
Vigile di Infrazioni
deve comparire
anche nell'attributo
Matricola di Vigili
→ vincolo di integrità
referenziale fra Vigile
in Infrazioni e Matricola
in Vigili (si dice anche
fra Vigile di Infrazioni e Vigili)*

Vigili

<u>Matricola</u>	<u>Cognome</u>	<u>Nome</u>
3987	Rossi	Luca
3295	Neri	Piero
9345	Neri	Mario
7543	Mori	Gino

Vincoli di integrità referenziale: esempio

<u>Codice</u>	<u>Comune</u>	Data	Vigile	Targa
34321	Roma	1/2/95	3987	39548K
53524	Milano	4/3/95	3295	E39548
64521	Napoli	5/4/96	3295	H39548
73321	Roma	5/2/98	9345	H39548

Infrazioni

Ogni coppia di valori che non contiene NULL e che compare in <Codice,Comune> di Pagamenti compare anche in <Codice,Comune> di Infrazioni → vincolo di integrità referenziale fra <Codice,Comune> di Pagamenti a <Codice,Comune> di Infrazioni (oppure fra <Codice,Comune> di Pagamenti a Infrazioni)

Pagamenti

<u>Transazione</u>	<u>Codice</u>	<u>Comune</u>
101	34321	Roma
295	53524	Milano
300	NULL	NULL
236	64521	Napoli



Vincoli di integrità referenziale: esempio di violazione

<u>Codice</u>	<u>Comune</u>	Data	Vigile	Targa
34321	Roma	1/2/95	3987	39548K
53524	Milano	4/3/95	3295	E39548
64521	Napoli	5/4/96	3295	H39548
73321	Roma	5/2/98	9345	H39548

Infrazioni

Pagamenti

<u>Transazione</u>	<u>Codice</u>	<u>Comune</u>
101	34321	Napoli
295	53524	Roma
300	73321	Roma
236	64521	Napoli

Le tuple colorate in rosso rappresentano violazioni del vincolo di integrità referenziale fra <Codice, Comune> di Pagamenti e Infrazioni

Integrità referenziale e valori nulli

Eventuali valori nulli contenuti in un attributo coinvolto in un vincolo di integrità referenziale non costituiscono una violazione del vincolo stesso. Nell'esempio qui sotto il vincolo di integrità referenziale da Progetto di Impiegati a Codice di Progetti è soddisfatto (anche se è presente il valore nullo)

Impiegati

<u>Matricola</u>	Cognome	Progetto
34321	Rossi	IDEA
53524	Neri	XYZ
64521	Verdi	NULL
73032	Bianchi	IDEA

Progetti

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



Esercizio 8

Partita

<u>Codice</u>	SqCasa	SqTrasferta	GoalCasa	GoalTrasf	Schedina

Squadra

<u>Nome</u>	AnnoFondazione	Città

Quali vincoli di integrità interrelazionali definireste su questo schema (si noti che **Nome** è la chiave primaria della relazione **Squadra**)?



Esercizio 9: soluzione

Partita

<u>Codice</u>	SqCasa	SqTrasferta	GoalCasa	GoalTrasf	Schedina

Squadra

<u>Nome</u>	AnnoFondazione	Città

Occorre definire

- 1.un vincolo di integrità referenziale tra l'attributo SqCasa e la relazione Squadra
- 2.un vincolo di integrità referenziale tra l'attributo SqTrasferta e la relazione Squadra



Vincoli interrelazionali: vincoli di inclusione

- Un **vincolo di inclusione** fra un sequenza non vuota X di n attributi di una relazione R_1 ed una sequenza Y di n attributi di una relazione R_2 è una asserzione che impone che per ogni istanza B della base di dati ogni combinazione di valori su X che non contengono NULL e che sono presenti nella istanza di R_1 in B compaia come combinazione di valori su Y nella istanza di R_2 in B
- È immediato verificare che la nozione di vincolo di integrità referenziale (foreign key) è una specializzazione della nozione di vincolo inclusione. Infatti **un vincolo di integrità referenziale non è nient'altro che un vincolo di inclusione in cui l'insieme di attributi in Y formano una chiave per R_2** . Detto in altro modo, la nozione di vincolo di inclusione adatta la nozione di vincolo di integrità referenziale al caso in cui l'insieme di attributi in Y non formino una chiave per R_2 (magari formano un superchiave di R_2 oppure non formano nemmeno una superchiave).