Basi di Dati

Appunti del corso Basi di Dati 1

▼ Lezione del 27/09/2023

Definizioni Generali introduttive

Un database è un'insieme di file organizzati da strutture dati che facilitano la loro creazione, accesso e aggiornamento.

L'accesso agli elementi delle strutture dati è possibile mediante query o interrogazioni.

Un Sistema di Informazioni (Information System) è un insieme di dati organizzati fisicamente e organizzati tramite software in modo da poterli creare, visionare ed aggiornare.

Esistono diversi modelli per organizzare database, i due principali sono:

-Modelli **Logici** indipendenti dalle strutture fisiche

-Modelli **Concettuali** indipendenti dai modelli di realizzazione (struttura dati)

Esempi pratici di modelli di organizzazione dei database:

Mesh Model

I dati sono rappresentati come una collezione di record di dati omogenei, le relazioni binarie sono rappresentate come collegamenti (implementati con puntatori), sono rappresentati quindi come una struttura a grafo.

Hierarchical Model

Un tipo ristretto di Mesh Model, rappresenta un grafico mediante grafi implementati tramite alberi (o foreste).

Relational Model

I dati e le relazioni sono rappresentati come valori, non ci sono espliciti collegamenti tra i dati (es non ci sono puntatori).

Un esempio di realizzazione di database mediante modello relazionale può essere la seguente:

STUDENTS

Matric	Last name	Name	Birthday
276545	Smith	Mary	25/11/1980
485745	Black	Anna	23/04/1981
200768	Greens	Paolo	12/02/1981
587614	Smith	Lucy	10/10/1980
937653	Brown	Mavis	01/12/1980

COURSES

Code	Title	Tutor
01	Physics	Grant
03	Chemistry	Beale
04	Chemistry	Clark

EXAMS

Stud	Vote	Course
276545	C	01
276545	B	04
937653	B	01

I tre livelli astratti di un Database:

- Schema Esterno:

  Descrizione di una porzione del database in un modello logico attraverso "viste" parziali o derivate che possono fornire organizzazioni dei dati diverse da quelle utilizzate nello schema logico e che riflettono le esigenze e i privilegi di accesso di tipi specifici di utenti; più di uno schema esterno può essere associato a uno schema logico.

- Schema Logico:

  Descrizione dell'intero database nel modello logico principale del DBMS, ad esempio la struttura delle tabelle.

- Schema Fisico:

  Rappresentazione dello schema logico attraverso strutture di archiviazione fisiche, cioè file.

Indipendenza dei dati:

- Indipendenza Fisica:

  I livelli logici ed esterni sono indipendenti dal livello fisico, ciò significa che le relazioni usate nel database sono indipendenti dalla loro realizzazione fisica (organizzazione dei file o la loro allocazione) inoltre l’implementazione fisica può cambiare senza avere ripercussioni sul programma.

- Indipendenza Logica:

  Il livello esterno è indipendente dal livello logico, ciò significa che sono possibili aggiunte o modifiche alle viste non richiedono modifiche al livello logico.

  Le modifiche allo schema logico che non influenzano lo schema esterno rimangono trasparenti.

In ogni Database esistono:

- Lo schema: sostanzialmente invariabile nel tempo che descrive la struttura del database stesso(nel modello relazionale rappresenta l’intestazione delle tabelle).
- Le istanze: i valori che correnti dei dati archiviati che possono cambiare nel tempo.

NAME	SURNAME	BIRTH	TOWN
Piero	Naples	22-10-63	Bari
Marco	Bianchi	01-05-54	Rome
Maria	Rossi	09-02-68	Milan
Maria	Bianchi	07-12-70	Bari
Paolo	Sossi	15-03-75	Palermo

Parte in bianco: Schema | Parte in giallo: istanze.

Integrità e Sicurezza

Integrità:

I dati devono soddisfare "vincoli" che esistono nel contesto di interesse:

- Uno studente risiede in una sola città (vincoli di dipendenza funzionale).
- Il numero di matricola identifica univocamente uno studente (vincoli di chiave).
- Un voto è un numero intero positivo compreso tra 18 e 30 (vincoli di dominio).
- Le ore straordinarie di un dipendente sono date dal prodotto del numero di ore e del salario orario.
- Il salario di un dipendente non può diminuire (vincoli dinamici).

Sicurezza

- I dati devono essere protetti dall'accesso non autorizzato.

L'amministratore del database (DBA) deve prendere in considerazione:

- Il valore attuale delle informazioni per l'organizzazione.
- Chi può accedere a quali dati e in quale modo.

Successivamente, il DBA deve prendere decisioni riguardanti:

- La regolamentazione dell'accesso.
- Gli effetti di una violazione.

Modello Relazionale

Introduzione

Le relazioni, che sono un insieme di coppie ordinate i cui elementi appartengono a domini differenti, si traducono naturalmente in tabelle.

I dati e le relazioni/associazioni tra dati di diversi insiemi (relazioni/tabelle) sono rappresentati come valori.

Bisogna fare attenzione alle diverse accezioni di "relazione" e "relazione" nell'ambito matematico e informatico.

Definizione n°1, Il Dominio

💡 **Dominio**: un insieme (possibilmente infinito) di valori.
Esempi:

- L'insieme degli interi è un dominio.
- L'insieme dei numeri decimali è un dominio.
- L'insieme delle stringhe di caratteri di lunghezza = 20 è un dominio.
- {0,1} è un dominio

Definizione n°2

💡 Siano D1, D2, ..., Dk domini, non necessariamente distinti; il prodotto cartesiano di questi domini, indicato da:

D1 x D2 x.. Dk

ed è l'insieme:

{(V1,V2,..Vk) | V1 ∈ D1, V2 ∈ D2,.. Vk ∈ Dk}

Definizione n°3

Una relazione matematica è qualsiasi sottoinsieme del prodotto cartesiano di uno o più domini.

Una relazione che è un sottoinsieme del prodotto cartesiano di k domini è detta di grado k.

Gli elementi di una relazione sono chiamati tuple (o ennuple): il numero di tuple di una relazione è la sua cardinalità.

Ciascuna tupla di una relazione di grado k ha k componenti ordinate (il valore i-esimo proviene dal dominio i-esimo), ma non c'è un ordine tra le tuple.

Le tuple di una relazione sono tutte distinte (almeno per un valore)

Esempio:

Supponiamo k=2

D1={bianco,nero}, D2={0,1,2}

D1 x D2= {(bianco,0),(bianco,1),(bianco,2),(nero,0),(nero,1),(nero,2)}

Quindi {(bianco,0),(nero,0),(nero,2)} è una relazione di grado 2 (poiché due elementi per coppia) di cardinalità 3 (poiché presenti 3 coppie).

Il numero massimo di relazioni per quest’ esempio è di 2^6 .

Notazione

Data **r,** una relazione di grado k

Data **t**, una tupla in **r**

Se i è un intero in {1,…,k}

t[i] indica l’ iesimo componente di t.

Esempio:

r = {(0,a),(0,c),(1,b)}

t = (0,a)

t[2] = a

t[1,2] = (0,a)

Relazioni e Tabelle

Un attributo è definito dal nome A e dal suo dominio, che denotiamo con dom(A).

Supponiamo che R sia un insieme di attributi. Una tupla su R è una funzione definita su R che associa a ciascun attributo A in R un elemento di dom(A).

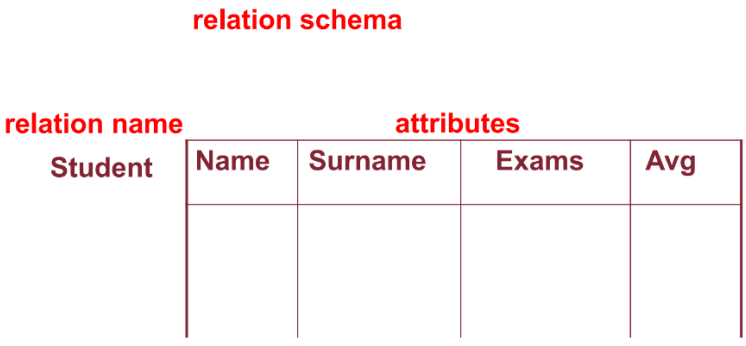
Se t è una tupla in R e A è un attributo in R, allora con t(A) denotiamo il valore assunto dalla funzione t sull'attributo A. In altre parole, t(A) rappresenta il valore dell'attributo A nella tupla t.

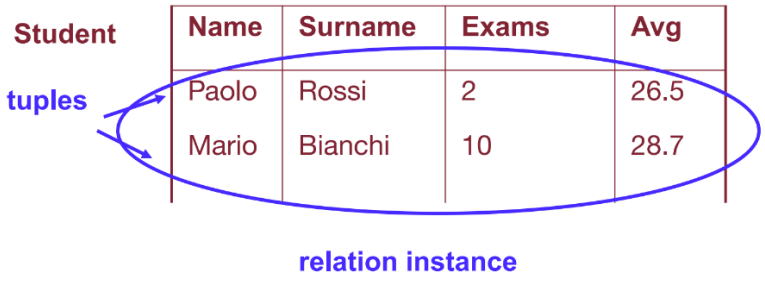
Le tuple sono un sottoinsieme dell’istanza dello schema.

Gli attributi rappresentano il “titolo” di ogni colonna della mia tabella.

Il relation name rappresenta il tipo di dato che sto immagazzinando nel database

L’istanza è l’insieme di dati che sono contenuti nelle varie tuple.





Dato t1={name,surname,exams,avg} questo insieme rappresenta gli attributi del mio database, il cui dominio è formato da String ∪ String ∪ Int ∪ Reale.

Riassumendo

Una relazione può essere considerata come una tabella in cui ogni riga rappresenta una tupla distinta e ogni colonna corrisponde a un componente (con valori omogenei, cioè provenienti dallo stesso dominio).

Le colonne corrispondono ai domini D1, D2, ..., Dk e hanno nomi univoci che sono autoesplicativi all'interno della tabella.

La coppia (nome attributo, dominio) è chiamata attributo; l'insieme di attributi di una relazione è chiamato schema.

Se R è una relazione e i suoi attributi sono:

A1,A2,..,Ak, lo schema è spesso indicato come:

R(A1, A2, ..., Ak)

Schemi e Istanze

Schema di una relazione: un nome di relazione R con un insieme di nomi di attributi.

R(A1, A2, ..., Ak)

Lo schema di una relazione è invariante nel tempo e descrive la sua struttura (aspetto intenzionale).

L'istanza di una relazione con schema R(X) è rappresentata da un insieme r di tuple su X.

L'istanza contiene i valori attuali, che possono cambiare rapidamente (aspetto estensionale).

In altre parole, lo schema fornisce la struttura concettuale della relazione, mentre l'istanza rappresenta i dati concreti e attuali contenuti nella relazione.

Relazioni e Database

Schema del database:

> un insieme di schemi di relazione con nomi diversi. In altre parole, il database schema è la descrizione complessiva della struttura e delle relazioni tra le tabelle (relazioni) all'interno di un database, e ciascun schema di relazione definisce la struttura specifica di una tabella all'interno del database.

Schema del database relazionale: un insieme di schemi R1, R2, ..., Rn di relazioni.

Un database relazionale con lo schema R1, R2, ..., Rn è definito come un insieme di relazioni

r1, r2, ..., rn, dove ogni ri è un'istanza di relazione con schema Ri.

Nella definizione di un modello relazionale, i componenti di una relazione vengono indicati dai nomi degli attributi, piuttosto che dalla loro posizione.

t[Ai] indica il valore dell'attributo con il nome Ai della tupla t.

Se t è la seconda tupla nell'esempio precedente, allora t[Region] = Lombardia.

Se Y è un sottoinsieme degli attributi dello schema X di una relazione ($Y \subseteq X$), allora t[Y] è il sottoinsieme dei valori nella tupla t che corrispondono agli attributi in Y (chiamato anche restrizione di t).

Nel modello relazionale, i riferimenti tra dati in diverse relazioni vengono rappresentati mediante i valori di dominio che compaiono nelle tuple.

▼ Lezione del 28/09/2023

Il valore Null

> 💡 **Definizione**
> I valori **NULL** rappresentano la mancanza di informazioni o il fatto che l'informazione non è applicabile.

Ad esempio, per un numero di telefono posso utilizzare il valore Null poichè:

- La persona non ha un telefono.
- Non so se la persona ha un telefono.
- La persona ha un telefono, ma non conosco il numero.

In alcune situazioni, è necessario inserire un valore (la tupla deve essere conforme allo schema). In questi casi, è possibile definire un valore predefinito.

Tuttavia, è importante notare che alcuni attributi non dovrebbero mai assumere valori NULL, come l'ID studente o i voti degli esami, poiché questi dovrebbero essere sempre presenti e noti.

Il problema delle "unused" (non utilizzate) valori di dominio è importante da considerare nel design di un database.

Questi valori potrebbero effettivamente essere utilizzati in futuro, e potrebbero influenzare calcoli come il salario di un lavoratore.

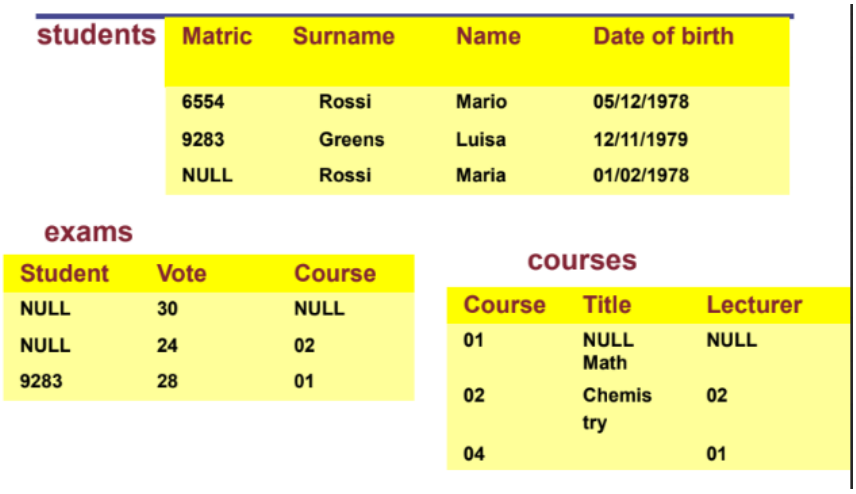
Pertanto, è importante decidere attentamente come trattare tali valori nel contesto specifico del tuo sistema.

Il valore speciale NULL è un concetto importante nei database relazionali. Viene utilizzato come un valore polimorfico che **non appartiene a nessun dominio** specifico ma può sostituire valori in qualsiasi dominio.

È importante notare che due valori NULL, anche nello stesso dominio, sono considerati diversi. Questo significa che due valori NULL in una colonna possono rappresentare concetti diversi o mancanza di informazioni diversificate.

La gestione dei valori NULL e dei valori non utilizzati dovrebbe essere pianificata attentamente per garantire che il database rappresenti accuratamente i dati e possa eseguire le operazioni desiderate in modo efficace.

> Ricorda Null NON E' 0



students

Matric	Surname	Name	Date of birth
6554	Rossi	Mario	05/12/1978
9283	Greens	Luisa	12/11/1979
NULL	Rossi	Maria	01/02/1978

exams

Student	Vote	Course
NULL	30	NULL
NULL	24	02
9283	28	01

courses

Course	Title	Lecturer
01	NULL Math	NULL
02	Chemis try	02
04		01

In questo Database sono presenti troppi valori null.

Integrity Contraints

Integrity constraints sono regole o condizioni che vengono applicate ai dati all'interno di un database relazionale per garantire l'integrità dei dati e la coerenza delle informazioni.

Queste regole vengono imposte per garantire che i dati soddisfino determinati standard di qualità e correttezza.

Queste proprietà devono essere soddisfatte da ogni istanza nel Database.

Esistono due tipi di contraints:

- intra-relational contraints: definiti su singoli domini o su valori di stesse tuple o tuple di stesse istanze.
- interrelational contraints: definite tra più relazioni.

Keys - Chiavi

Le chiavi sono fondamentali per identificare le tuple di un istanza.

Definizione informale:

Una chiave relazionale (non necessariamente unica) è un attributo o un insieme di attributi che identificano unicamente una tupla.

Definizione formale:

💡 Un insieme X di attributi di una relazione R è una chiave di R se soddisfa le seguenti condizioni:

1. Per ogni istanza di R, non esistono due tuple distinte t1 e t2 che abbiano gli stessi valori per tutti gli attributi in X, cioè tali che t1[X] = t2[X].
2. Non esiste un sottoinsieme proprio di X che soddisfa la condizione 1).

Definizione matematica:

💡 1) $X \subseteq R(A1,\ldots,An)$
$\forall\ t1,t2 \in R$
$t1[x]=t2[x] \Rightarrow t1=t2$

2) $\forall x' \subseteq X$
$\exists\ t1,t2 \in r$ tale che:
$t1[x]=t2[x] \wedge t1 \neq t2$

TAXCODE	CODE	SURNAME	NAME	ROLE	HIRING
CSR...	COD1	Rossi	Mario	Analyst	1995
BA...	COD2	Bianchi	Peter	Analyst	1990
NRI...	COD3	Neri	Paolo	Admin	1985

Una chiave è tale quindi se non esistono 2 attributi uguali al suo interno, in Role ho 2 volte Analyst e per questo NON è una chiave.

La definizione delle chiavi in questo schema (ID, SURNAME, NAME, HIRING) vale però solo per questa istanza d’esempio poiché sarebbe plausibile trovare in un vero Database più persone con lo stesso nome.

Una relazione può avere diverse chiavi alternative.

Tuttavia, tra queste chiavi alternative, se ne sceglie una come chiave primaria (primary key). La chiave primaria è solitamente quella più utilizzata o composta da un numero minore di attributi rispetto alle altre chiavi alternative.

La chiave primaria ha alcune caratteristiche importanti:

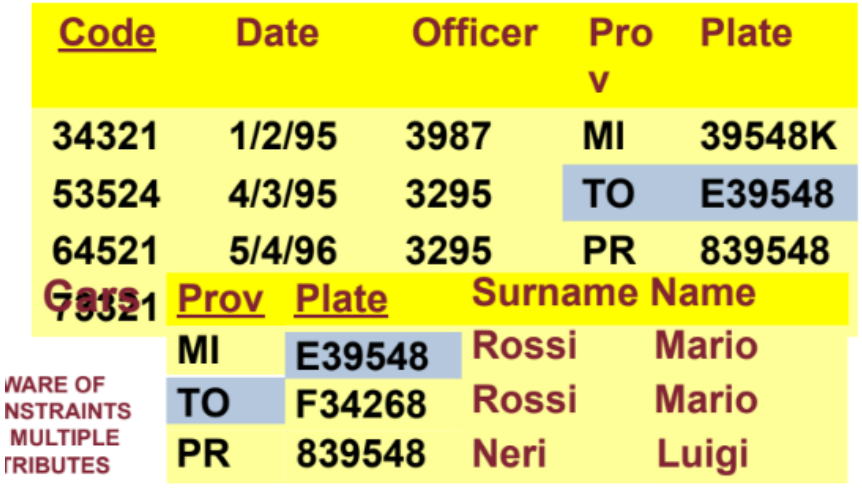
1. **Non Ammette Valori NULL**: La chiave primaria non consente valori NULL. Ogni tupla all'interno della relazione deve avere un valore univoco per ciascun attributo della chiave primaria.
2. **Identificazione Unica**: La chiave primaria deve essere in grado di identificare univocamente ogni tupla nella relazione. Questo significa che non possono esistere due tuple con gli stessi valori per tutti gli attributi della chiave primaria.
3. **Almeno una Chiave**: Ogni relazione deve avere almeno una chiave, poiché non possono esistere tuple identiche all'interno di una relazione. Quindi, anche se non viene esplicitamente definita una chiave primaria, ce ne sarà sempre almeno una.

Le chiavi, inclusa la chiave primaria, sono importanti perché consentono di stabilire relazioni tra le tabelle in un database. Ad esempio, le chiavi esterne in una tabella possono fare riferimento alla chiave primaria in un'altra tabella, consentendo di collegare dati tra tabelle diverse.

Interrelational Contraints

I vincoli di integrità referenziale tra le relazioni "Road violations" e "Officers" e tra "Road violations" e "Cars" possono essere definiti come segue:

- L'attributo "Officer" nella relazione "Road Violations" e l'attributo "ID" (che è la chiave) nella relazione "Agenti".
- Gli attributi "Prov" e "Plate" delle "Road Violations" e gli attributi "Prov" e "Plate" (che costituiscono la chiave) della relazione Cars.



Svolgono un ruolo “chiave” nel concetto di un "modello basato su valori" (il modello relazionale).

Nella presenza di valori NULL, i vincoli possono essere resi meno restrittivi.

È possibile definire "azioni" compensatorie in seguito a violazioni.

Dipendenze funzionali

Una dipendenza funzionale stabilisce un collegamento semantico tra due insiemi non vuoti di attributi, X e Y, appartenenti a uno schema R.

·Questo vincolo viene scritto come X → Y e viene letto come: "X determina Y".

> 💡 **Definizione**
> Sono dei modi per esprimere vincoli, fondamentali nella teoria relazionale.

Esempio di dipendenze funzionali:

Supponiamo di avere uno schema di relazione "Flights" (Codice, Giorno, Pilota, Ora) con i seguenti vincoli basati sul buon senso:

- Un volo con un determinato codice parte sempre alla stessa ora.
- C'è solo un volo con un dato pilota, in un dato giorno, a un dato orario.
- C'è solo un pilota per un dato volo in un dato giorno.

Questi vincoli generano le seguenti dipendenze funzionali:

- Code → Time (Il codice determina l'ora di partenza del volo.)
- {Day, Pilot, Time} → Code (Il giorno, il pilota e l'orario determinano il codice del volo.)
- {Code, Day} → Pilot (Il codice e il giorno determinano il pilota del volo.)

Diciamo che un' istanza r con schema R soddisfa la dipendenza funzionale X → Y se:

💡 1)
La dipendenza funzionale X → Y è applicabile a R nel senso che sia X che Y sono sottoinsiemi dell'insieme di attributi R, cioè entrambi X e Y sono attributi presenti nello schema R.

2)
Nelle tuple r che sono identiche sull'insieme di attributi X, devono anche essere identiche sull'insieme di attributi Y.

$$t1[X] = t2[X] \Rightarrow t1[Y] = t2[Y].$$

> La freccia verso destra significa che se le tuple sono uguali su X allora devono essere uguali su Y.

X e Y sono sottoinsiemi di attributi di tutti gli attributi dello schema.

▼ Lezione del 11/10

L’algebra relazionale

Definiamo come algebra relazionale una **notazione algebrica specifica** utilizzata per realizzare **query** su un database relazionale, composta da un insieme di operatori unari e binari che, se applicati rispettivamente ad una o due istanze di relazioni, generano una **nuova istanza**.
In particolare, individuiamo i seguenti quattro tipi di operatore:

1. **Rimozione** di specifiche parti di una relazione (Proiezione e Selezione)
2. **Operazioni insiemistiche** (Unione, Intersezione e Differenza)
3. C**ombinazione delle tuple di due relazioni** (Prodotto cartesiano e Join)
4. **Ridenominazione** di attributi.

L’algebra relazionale è un **linguaggio procedurale**, ossia descrive l’esatto ordine con cui
gli operatori devono essere applicati.

Proiezione e Selezione

Proiezione

Definizione
Sia R una relazione con istanza r e sia A1, . . . ,Ak un insieme di attributi.
Una **proiezione**
su R è un operatore unario che effettua una restrizione con A1, . . . ,Ak su tutte
le tuple di R:

$$\pi_{A1,...,Ak}(R) := \{t[A1, \ldots, Ak] \mid t \in r\}$$

In altre parole, una proiezione effettua un "**taglio verticale**" su una relazione, selezionando
solo le colonne A1, . . . ,Ak

Esempio:

- Supponiamo di voler ottenere una lista di tutti i clienti presenti in questa relazione:

Customers		
Name	**C#**	**Town**
Rossi	C1	Roma
Rossi	C2	Milano
Rossi	C3	Milano
Bianchi	C4	Roma
Verdi	C5	Roma

- Proviamo ad effettuare una proiezione π Name(Customers), il cui output sarà:

$\pi_{\text{Name}}(\text{Customers})$
Name
Rossi
Bianchi
Verdi

- La nuova relazione generata dalla proiezione segue comunque le regole dell'insiemistica, dunque non possono esserci tuple uguali. Difatti, notiamo come nell'output sia presente una sola tupla contente il nome "Rossi", nonostante nella relazione iniziale ve ne fossero tre.
- Per prevenire tale perdita di informazione, proviamo ad effettuare la proiezione π Name, Town(Customers), il cui output sarà:

$\pi_{\text{Name, Town}}(\text{Customers})$

Name	Town
Rossi	Roma
Rossi	Milano
Bianchi	Roma
Verdi	Roma

- La situazione è migliorata rispetto alla prima proiezione, tuttavia vi è stata comunque una perdita di informazione.
- Per prevenire totalmente la perdita di informazione, quindi, sfruttiamo l'unicità della chiave C#, effettuando la proiezione π Name, C#(Customers), il cui output sarà:

$\pi_{\text{Name, C\#}}(\text{Customers})$

Name	C#
Rossi	C1
Rossi	C2
Rossi	C3
Bianchi	C4
Verdi	C5

- Abbiamo quindi ottenuto una lista completa dei nostri clienti senza alcuna perdita di informazioni

Selezione

Definizione

Data una relazione R con istanza r e schema R(X), una selezione su R è un operatore unario che data una proposizione logica ' seleziona tutte le tuple di R per cui ' è rispettata:

σ(R) := {t 2 r | ' è valida}

Le proposizioni logiche associate ad una selezione corrispondono ad un composto di espressioni Booleane (dunque utilizzando operatori come ^, _ e ¬) i cui termini appaiono nelle forme AØB o AØa, dove:

• A,B 2 R(X) | dom(A) = dom(B)

• A 2 R(X), a 2 dom(A)

• Ø è un operatore di comparazione (<,=,-,>)

In altre parole, una selezione effettua un "taglio orizzontale" su una relazione, selezionando solo alcune tuple di essa.

Esempio:

- Supponiamo di voler ottenere tutte le informazioni riguardo i clienti provenienti da Roma presenti in questa relazione:

Customers		
Name	**C#**	**Town**
Rossi	C1	Roma
Rossi	C2	Milano
Rossi	C3	Milano
Bianchi	C4	Roma
Verdi	C5	Roma

- Possiamo effettuare una selezione #Town=’Roma’(Customers) per ottenere le tuple richieste:

$\sigma_{\texttt{Town='Roma'}}(\texttt{Customers})$		
Name	**C#**	**Town**
Rossi	C1	Roma
Bianchi	C4	Roma
Verdi	C5	Roma

- Vogliamo ora ottenere tutte le informazioni riguardo i clienti chiamati "Rossi" provenienti da Roma. Possiamo effettuare una selezione σNome=’Rossi’^Town=’Roma’(Customers) per ottenere le tuple richieste:

$\sigma_{\text{Nome='Rossi'}\wedge\text{Town='Roma'}}(\text{Customers})$

Name	C#	Town
Rossi	C1	Roma

Unione, Intersezione e Differenza

Compatibilità di unione

Data una relazione R1 con schema R1(A1, . . . ,Ak) ed una relazione R2 con schema
R2(a1, . . . ,Ah), tali relazioni vengono dette compatibili in unione se e solo se:

- k = h, ossia hanno lo stesso numero di attributi
- ∀i ∈ [1, k], dom(R1.Ai) = dom(R2.Ai), ossia ogni attributo corrispondente ha lo stesso dominio

Unione

> 💡 **Definizione**
> Date due relazioni compatibili in unione R1 e R2 con rispettive istanze r1 e r2,
> l’unione tra R1 e R2 è un operatore binario che restituisce una nuova relazione
> contenente tutte le tuple presenti in almeno una relazione tra R1 e R2.
> $R1 \cup R2 := \{t \mid t \in r1 \vee t \in r2\}$

Affinché sia possibile utilizzare l’operatore di unione, non è necessario che gli attributi corrispondenti delle due relazioni abbiano lo stesso nome, ma solo lo stesso dominio (nonostante spesso non abbia alcun senso unire due relazioni aventi attributi con nomi diversi).

Esempio:

- Consideriamo le seguenti due relazioni, descriventi gli insegnanti e i responsabili di vari dipartimenti di una scuola:

Teachers		
Name	**Code**	**Department**
Rossi	C1	Math
Rossi	C2	Italian
Bianchi	C3	Math
Verdi	C4	English

Admins		
Name	**AdminCode**	**Department**
Esposito	C1	English
Riccio	C2	Math
Pierro	C3	Italian
Verdi	C4	English
Bianchi	C5	English

- Effettuando l'unione tra *Teachers* e *Admins*, otteniamo una nuova relazione contenente tutti i membri dello staff:

Teachers ∪ Admins		
Name	**Code**	**Department**
Rossi	C1	Math
Rossi	C2	Italian
Bianchi	C3	Math
Verdi	C4	English
Esposito	C1	English
Riccio	C2	Math
Pierro	C3	Italian
Bianchi	C5	English

- Consideriamo le seguenti due relazioni:

Teachers		
Name	**Code**	**Department**
Rossi	C1	Math
Rossi	C2	Italian
Bianchi	C3	Math
Verdi	C4	English

Admins			
Name	**Code**	**Department**	**Salary**
Esposito	C1	English	1250
Riccio	C2	Math	2000
Pierro	C3	Italian	1000

- È impossibile applicare l'operatore unione tra le due relazioni *Teachers* e *Admins*, poiché non possiedono lo stesso numero di attributi
- Per risolvere il problema, possiamo effettuare prima una proiezione su *Admins*, per poi effettuare l'unione con *Teachers*:

$\texttt{Teachers} \cup \pi_{\texttt{Name, AdminCode, Department}}(\texttt{Admins})$		
Name	**Code**	**Department**
Rossi	C1	Math
Rossi	C2	Italian
Bianchi	C3	Math
Verdi	C4	English
Esposito	C1	English
Riccio	C2	Math
Pierro	C3	Italian

- Consideriamo ora le seguenti due relazioni:

Teachers		
Name	**Code**	**Department**
Rossi	C1	Math
Rossi	C2	Italian
Bianchi	C3	Math
Verdi	C4	English

Admins		
Name	**Code**	**ServiceYears**
Esposito	C1	3
Riccio	C2	13
Pierro	C3	7

- È impossibile applicare l'operatore unione tra le due relazioni *Teachers* e *Admins*, poiché dom(Teachers.Department) ≠ dom(Admins.ServiceYears). Tuttavia, possiamo effettuare una proiezione su entrambe le relazioni, per poi unirle:

$\pi_{\text{Name, Code}}(\text{Teachers}) \cup \pi_{\text{Name, Code}}(\text{Admins})$

Name	Code
Rossi	C1
Rossi	C2
Bianchi	C3
Verdi	C4
Esposito	C1
Riccio	C2
Pierro	C3

Intersezione

💡 **Definizione**
Date due relazioni **compatibili** in unione R1 e R2 con rispettive istanze r1 e r2,
l'intersezione tra R1 e R2 è un **operatore binario** che restituisce una nuova relazione
contenente tutte le tuple presenti sia in R1 sia in R2.

$$R1 \cap R2 := \{t \mid t \in r1 \land t \in r2\}$$

Differenza

💡 **Definizione**
Date due relazioni compatibili in unione R1 e R2 con rispettive istanze r1 e r2,
la differenza tra R1 e R2 è un operatore binario che restituisce una nuova relazione
contenente tutte le tuple presenti in R1 ma non in R2.

$$R1 - R2 := \{t \mid t \in r1 \wedge t \notin r2\}$$

Contrariamente da unione e intersezione, l'operatore di **differenza non è commutativo.**

Esempio:

- Consideriamo le seguenti due relazioni, descriventi gli insegnanti e i responsabili di
  vari dipartimenti di una scuola:

Teachers		
Name	**Code**	**Department**
Rossi	C1	Math
Rossi	C2	Italian
Verdi	C3	English
Bianchi	C4	English

Admins		
Name	**AdminCode**	**Department**
Esposito	C1	English
Riccio	C2	Math
Verdi	C3	English
Bianchi	C4	English

- Effettuando **l'intersezione** tra *Teachers* e *Admins*, otteniamo una nuova relazione

contenente tutti gli insegnanti che sono anche responsabili:

Teachers ∩ Admins		
Name	**Code**	**Department**
Verdi	C4	English
Bianchi	C5	English

- Effettuando la **differenza** tra *Teachers* e *Admins*, otteniamo una nuova relazione contenente tutti gli insegnanti che non sono responsabili:

Teachers − Admins		
Name	**Code**	**Department**
Rossi	C1	Math
Rossi	C2	Italian

- Analogamente, effettuando la differenza tra *Admins* e *Teachers*, otteniamo una nuova relazione contenente tutti i responsabili che non sono insegnanti:

Admins − Teachers		
Name	**AdminCode**	**Department**
Esposito	C1	English
Riccio	C2	Math

Ridenominazione

💡 **Definizione**
Sia R una relazione con istanza r e schema R(A1, . . . ,Ak). Una **ridenominazione** su R è un operatore unario che restituisce una nuova relazione R' con istanza r' e schema R0(B1, . . . ,Bk), dove le tuple di r' sono identiche alle tuple di r:

$$\rho B1,...,Bk \leftarrow A1,...,Ak(R) := \{t' \mid t'[Bi] = t[Ai], t \in r, \forall i \in [1, k]\}$$

In altre parole, una ridenominazione **modifica** il nome di un attributo della relazione.

Prodotto Cartesiano

💡 **Definizione**
Siano R1 ed R2 due relazioni con rispettive istanze r1 e r2. Il **prodotto cartesiano** di R1 e R2 è un operatore binario che restituisce una relazione contenente tutte le possibili combinazioni tra le tuple di r1 e le tuple di r2:

$$R1 \times R2 := \{(t1, t2) \mid t1 \in r1, t2 \in r2\}$$

Esempio:

- Consideriamo le seguenti relazioni:

Customers		
Name	**C#**	**Town**
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Orders			
O#	**C#**	**A#**	**Qnty**
O1	C1	A1	100
O2	C2	A2	200
O3	C3	A2	150
O4	C4	A3	200
O1	C1	A2	200
O1	C1	A3	100

- Vogliamo ottenere l’elenco di tutti i clienti e gli ordini da loro effettuati.
- Prima di poter effettuare il prodotto cartesiano tra le due relazioni, è necessario ridenominare uno dei due attributi C# presenti in entrambe le relazioni

$$\text{OrdersR} = \boldsymbol{\rho}\text{C\#} \leftarrow \text{CC\#}(\text{Orders})$$

- Successivamente, effettuando il prodotto cartesiano Customers ×OrdersR, otteniamo:

Customers × OrdersR						
Name	**C#**	**Town**	**O#**	**CC#**	**A#**	**Qnty**
Rossi	C1	Roma	O1	C1	A1	100
Rossi	C1	Roma	O2	C2	A2	200
Rossi	C1	Roma	O3	C3	A2	150
Rossi	C1	Roma	O4	C4	A3	200
Rossi	C1	Roma	O1	C1	A2	200
Rossi	C2	Milano	O1	C1	A1	100
Rossi	C2	Milano	O2	C2	A2	200
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Verdi	C4	Roma	O4	C4	A3	200
Verdi	C4	Roma	O1	C1	A2	200

- A questo punto, però, notiamo la presenza di alcune incorrettezze. Ad esempio, il cliente "Rossi", avente codice *C1*, non ha mai effettuato l'ordine "(O2, C2, 200)", il quale invece è stato effettuato dal cliente avente codice *C2*.
- Possiamo risolvere tale problema effettuando una selezione dopo aver effettuato il
  prodotto cartesiano:

$\sigma_{C\#=CC\#}$(Customers × OrdersR)						
Name	**C#**	**Town**	**O#**	**CC#**	**A#**	**Qnty**
Rossi	C1	Roma	O1	C1	A1	100
Rossi	C1	Roma	O1	C1	A2	200
Rossi	C1	Roma	O1	C1	A3	100
Rossi	C2	Milano	O2	C2	A2	200
Bianchi	C3	Roma	O3	C3	A2	150
Verdi	C4	Roma	O4	C4	A3	200

- Infine, per via del select svolto, le colonne *C#* e *CC#* risultano essere uguali, dunque
  potremmo rimuovere una delle due effettuando una proiezione.
  La query finale, quindi risulta essere:

OrdersR = **ρ**C# ← CC#(Orders)

CustomerOrders = **π**Name, C#, Town, O#, A#, Qnty(**σ**C#=CC#(Customers ***x***OrdersR))

CustomerOrders					
Name	**C#**	**Town**	**O#**	**A#**	**Qnty**
Rossi	C1	Roma	O1	A1	100
Rossi	C1	Roma	O1	A2	200
Rossi	C1	Roma	O1	A3	100
Rossi	C2	Milano	O2	A2	200
Bianchi	C3	Roma	O3	A2	150
Verdi	C4	Roma	O4	A3	200

Join Naturale

💡 **Definizione**
Siano R1 e R2 due relazioni con rispettive istanze r1 e r2 e rispettivi schemi R1(X) e R2(Y). Il join naturale tra R1 e R2 è un operatore binario equivalente a:
- R1 ⋈ R2 = **π**X,(Y −X)(**σφ**(R1 × R2))
dove dato un insieme di attributi A1, . . . ,Ak | 8i 2 [1, k], Ai 2 X \ Y si ha che:
- **φ**:= R1.A1 = R2.A1 ∧ . . . ∧ R1.Ak = R2.Ak

In altre parole, il **join naturale** tra R1 ed R2 restituisce l'insieme di tutte le combinazioni tra tuple di r1 ed r2 che sono uguali per i loro attributi in comune.

Esempi:

- 1-Riprendiamo l'esempio visto per il prodotto cartesiano: date le seguenti due relazioni, vogliamo ottenere un elenco di tutti i clienti e gli ordini da

loro effettuati

Customers		
Name	**C#**	**Town**
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Orders			
O#	**C#**	**A#**	**Qnty**
O1	C1	A1	100
O2	C2	A2	200
O3	C3	A2	150
O4	C4	A3	200
O1	C1	A2	200
O1	C1	A3	100

- Notiamo come la soluzione già vista sia equivalente ad un join naturale tra le due relazioni:

Customers ⋈ Orders					
Name	**C#**	**Town**	**O#**	**A#**	**Qnty**
Rossi	C1	Roma	O1	A1	100
Rossi	C1	Roma	O1	A2	200
Rossi	C1	Roma	O1	A3	100
Rossi	C2	Milano	O2	A2	200
Bianchi	C3	Roma	O3	A2	150
Verdi	C4	Roma	O4	A3	200

- 2-Oltre alle due precedenti relazioni, consideriamo anche la seguente relazione:

Articles		
A#	**Label**	**Price**
A1	Plate	3
A2	Glass	2
A3	Mug	4

- Vogliamo ottenere una lista dei nomi e delle città dei clienti che hanno ordinato più di 100 pezzi di articoli che costano più di due euro.
- Prima di tutto, effettuiamo un join naturale tra le tre relazioni:

$$\texttt{CustOrdArt} = (\texttt{Customers} \bowtie \texttt{Orders}) \bowtie \texttt{Articles}$$

CustOrdArt							
Name	**C#**	**Town**	**O#**	**A#**	**Qnty**	**Label**	**Price**
Rossi	C1	Roma	O1	A1	100	Plate	3
Rossi	C1	Roma	O1	A2	200	Glass	2
Rossi	C1	Roma	O1	A3	100	Mug	4
Rossi	C2	Milano	O2	A2	200	Glass	2
Bianchi	C3	Roma	O3	A2	150	Glass	2
Verdi	C4	Roma	O4	A3	200	Mug	4

- Successivamente, selezioniamo le tuple che rispettano le condizioni che ci interessano:

$\sigma_{\texttt{Qnty}>100 \wedge \texttt{Price}>2}(\texttt{CustOrdArt})$							
Name	**C#**	**Town**	**O#**	**A#**	**Qnty**	**Label**	**Price**
Verdi	C4	Roma	O4	A3	200	Mug	4

- Infine, effettuiamo una proiezione sul nome e la città delle tuple ottenute:

$\pi_{\texttt{Name, Town}}(\sigma_{\texttt{Qnty}>100 \wedge \texttt{Price}>2}(\texttt{CustOrdArt}))$	
Name	**Town**
Verdi	Roma

- 3- Oltre alla soluzione appena vista, possiamo ottenere lo stesso risultato selezionando prima le tuple che rispettano le condizioni e i

dati che ci interessano, per poi effettuare il join tra loro, rendendo così la query più efficiente, poiché la mole di dati su cui vengono applicati gli operatori è minore:

$$\pi_{Name,Town}((Customer \rhd\lhd \sigma_{Qnty>100}(Order)) \rhd\lhd \sigma_{Price>2}(\pi_{A\#,Price}(Article)))$$

Casi Speciali join naturale

Siano R1 e R2 due relazioni con rispettivi schemi R1(X) e R2(Y).

1. Se R1 ed R2 hanno un insieme di attributi in comune ma nessun valore in comune
   per tali attributi, allora il risultato sarà un insieme vuoto:
   $$Z \subseteq X \cap Y, \nexists t1 \in R1 \wedge t2 \in R2 \mid t1[Z] = t2[Z] \Rightarrow R1 \rhd\lhd R2 = \varnothing$$
2. Se R1 ed R2 non hanno degli attributi in comune, allora il join naturale degenererà
   in un prodotto cartesiano:
   $$X \cap Y = \varnothing \Rightarrow R1 \rhd\lhd R2 = R1 \times R2$$
   Posso usare la rinomina.

▼ Lezione del 12/10

Theta Join

Definizione

Siano R1 e R2 due relazioni con rispettive istanze r1 e r2 e rispettivi schemi R1(X) e R2(Y).

Il join naturale tra R1 e R2 è un operatore binario equivalente a:

$$R1 \triangleright\!\triangleleft_{A\theta B} R2 = \sigma_{A\theta B}(R1 \times R2)$$

dove:

- $A \in R1(X), B \in R2(Y)$
- $dom(A) = dom(B)$
- θ è un operatore di confronto (<,=,>)

In altre parole, un theta join tra R1 ed R2 restituisce tutte le combinazioni tra le tuple di r1 e r2 che rispettano una condizione su un attributo in comune

In alcuni casi può essere necessario effettuare il join tra una relazione e se stessa (**self join**), in modo da ottenere combinazioni di tuple della stessa relazione.

Esempio:

- Data la seguente relazione, vogliamo ottenere una lista dei codici e dei nomi dei dipendenti aventi un salario maggiore dei loro supervisori

Employees				
Name	**C#**	**Section**	**Salary**	**Supervisor#**
Rossi	C1	B	100	C3
Pirlo	C2	A	200	C3
Bianchi	C3	A	500	NULL
Verdi	C4	B	200	C2
Neri	C5	B	150	C1
Tosi	C6	B	100	C1

- In tal caso, la soluzione migliore risulta essere una selezione effettuata su self join
  di *Employees*, in modo da poter accoppiare ogni dipendente al suo supervisore.
  Possiamo ottenere un self join eseguendo una delle seguenti query:

– Creiamo una copia della relazione per poi effettuare un theta join tra il codice dei dipendenti e il codice dei loro supervisori, specificando la relazione di appartenenza di ognuno dei due attributi confrontati:

$$\text{EmployeesC} = \text{Employees}$$

$$\text{EmpSup1} = \text{EmployeesC} \rhd\lhd \text{EmployeesC.Supervisor\#} = \text{Employees.C\#}\ \text{Employees}$$

– Effettuiamo un theta join tra la relazione ed una sua copia rinominata, senza dover specificare la relazione di appartenenza degli attributi confrontati:

$$X = \{\text{Name, C\#, Section, Salary, Supervisor\#}\}$$

$$Y = \{\text{CName, CC\#, CSec, CSal, CSup\#}\}$$

$$\text{EmpSup2} = \text{Employees} \rhd\lhd \text{Supervisor\#} = \text{C\#}\ \boldsymbol{\rho} X \leftarrow Y\ (\text{Employees})$$

- Attenzione: utilizzare il join naturale al posto del theta join in una delle tre soluzioni genererebbe una relazione identica a *Employees*, poiché ogni tupla verrebbe joinata con se stessa scartando automaticamente gli attributi doppioni.
- Successivamente, eseguiamo la selezione richiesta, mantenendo solo le tuple dove il
  salario del dipendente è maggiore del salario del suo supervisore:

  $$\sigma \text{Employees.Salary} > \text{EmployeesC.Salary}(\text{EmpSup1})$$

  oppure:

  $$\sigma \text{Salary} > \text{CSalary}(\text{EmpSup2})$$

$\sigma_{\texttt{Salary > CSal}}(\text{EmpSup}_2)$									
Name	**C#**	**Section**	**Salary**	**Supervisor#**	**CName**	**CC#**	**CSec**	**CSal**	**CSup#**
Verdi	C4	B	200	C2	Pirlo	C2	A	200	C3
Neri	C5	B	150	C1	Rossi	C1	B	100	C3
Tosi	C6	B	100	C1	Rossi	C1	B	100	C3

- Infine, effettuiamo una proiezione sul nome e il codice del dipendente:

πEmployees.Name, Employees.C#(σEmployees.Salary > EmployeesC.Salary(EmpSup1))
oppure:
πName, C#(#Salary > CSal(EmpSup2))

$\pi_{\texttt{Name, C\#}}(\sigma_{\texttt{Salary > CSal}}(\text{EmpSup}_2))$	
Name	**C#**
Verdi	C4
Neri	C5
Tosi	C6

Quantificazione universale

Fino ad ora, abbiamo visto solo query inerenti la **quantificazione esistenziale** (indicata col simbolo ∃), ossia la selezione di oggetti che soddisfino almeno una volta una determinata condizione.
Tuttavia, utilizzando solo gli operatori visti precedentemente, non abbiamo un modo per poter effettuare query inerenti alla **quantificazione universale** (indicata col simbolo ∀), ossia la selezione di oggetti che soddisfino **sempre** una determinata condizione.

Nella logica del primo ordine, la negazione di "Per ogni oggetto x la condizione ' è vera" non corrisponde a "Per ogni oggetto x la condizione ' è falsa", bensì corrisponde a "Esiste almeno un oggetto x per cui la condizione ' è falsa".

In simboli, diremmo che:

$$\neg(\forall x, \varphi(x)) \neq \forall x, \neg\varphi(x)$$

ma bensì:

$$\neg(8x, \varphi(x)) = \exists x, \neg\varphi(x)$$

Per selezionare tutte le tuple di una relazione per cui una determinata condizione ' è sempre valida, quindi, ci basta scartare tutte le tuple per cui almeno una volta la condizione non è valida.

Esempi:

- 1- Data la seguente relazione, vogliamo ottenere un elenco di tutti i nomi e la città di provenienza dei clienti che hanno sempre effettuato un ordine di più di 100 pezzi.

Customers		
Name	**C#**	**Town**
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Orders			
O#	**C#**	**A#**	**Qnty**
O1	C1	A1	100
O2	C2	A2	200
O3	C3	A2	150
O4	C4	A3	200
O1	C1	A2	200
O1	C1	A3	100

- Per ottenere l’elenco richiesto, ci basta scartare l’elenco dei nomi e delle città che almeno una volta non hanno acquistato più di 100 pezzi dall’elenco totale dei nomi e delle città:

$$\texttt{ElencoNonValidi} = \pi_{\texttt{Name, Town}}(\sigma_{\neg(\texttt{Qnty}>100)}(\texttt{Customers} \bowtie \texttt{Orders}))$$

$$R = \pi_{\texttt{Name, Town}}(\texttt{Customers}) - \texttt{ElencoNonValidi}$$

oppure, direttamente:

$$R = \pi_{\texttt{Name, Town}}(\texttt{Customers}) - \pi_{\texttt{Name, Town}}(\sigma_{\neg(\texttt{Qnty}>100)}(\texttt{Customers} \bowtie \texttt{Orders}))$$

R	
Name	**Town**
Bianchi	Roma
Verdi	Roma

- 2-Data la seguente relazione, vogliamo ottenere una lista dei codici e dei nomi dei supervisori aventi un salario maggiore di tutti i loro dipendenti

Employees				
Name	**C#**	**Section**	**Salary**	**Supervisor#**
Rossi	C1	B	100	C3
Pirlo	C2	A	200	C3
Bianchi	C3	A	500	NULL
Verdi	C4	B	200	C2
Neri	C5	B	150	C1
Tosi	C6	B	100	C1

- Anche in questo caso, per ottenere l'elenco richiesto ci basta scartare i supervisori aventi il salario minore di almeno un dipendente:

$$\texttt{EmployeesC} = \texttt{Employees}$$

$$\texttt{EmpSup} = \texttt{EmployeesC} \bowtie_{\texttt{EmployeesC.Supervisor\# = Employees.C\#}} \texttt{Employees}$$

$$\texttt{Invalid} = \pi_{\texttt{Name, C\#}}(\sigma_{\neg(\texttt{Employees.Salary < EmployeesC.Salary})}(\texttt{EmpSup}))$$

$$R = \pi_{\texttt{Name, C\#}}(\texttt{EmpSup}) - \texttt{Invalid}$$

R	
Name	**C#**
Bianchi	C3

▼ Lezione del 19/10

Teoria della Normalizzazione

Dipendenze funzionali

Sia R uno schema con istanza r e siano X, Y ⊆ R.
Definiamo come **dipendenza funzionale** tra X e Y , indicata come X → Y e letta "X determina Y ", un vincolo di integrità che impone ad ogni coppia di tuple in r aventi valori uguali su X di avere valori uguali anche su Y :

$$\forall t1, t2 \in r, t1[X] = t2[X] \Rightarrow t1[Y] = t2[Y]$$

Attenzione: notiamo come nella condizione non vi sia un "*se e solo se*", bensì solo un "*se*". Dunque, X → Y non implica che ogni coppia di tuple in r aventi valori uguali su Y debba avere valori uguali anche su X:

Esempi:

- Supponiamo di avere il seguente schema *Flights(Code, Day, Pilot, Time)*. Viene naturale considerare i seguenti vincoli:

- Un volo con un certo codice parità sempre allo stesso orario
- C’è un solo volo con un certo pilota ad un certo orario in un certo giorno
- C’è un solo pilota di un certo volo in un certo giorno

Dunque,imponiamo le seguenti dipendenze funzionali sullo schema:

- *Code → Time*
- *(Day, Pilot, Time)→Code*
- *(Code, Day) → Pilot*

Istanza legale

Dato uno schema R e un insieme F di dipendenze funzionali definite su R, diciamo che
un’istanza di R è **legale su F** se soddisfa tutte le dipendenze funzionali in F

Esempi:

- Consideriamo la seguente relazione su cui sono definite le seguenti dipendenze funzionali:

$$F = \{A \rightarrow B\}$$

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2
a_2	b_2	c_1	d_3

Tale istanza è *legale* su F, poiché soddisfa le dipendenze funzionali in F:
tutte le tuple aventi t[A] = a1 hanno anche t[B] = b1, così come tutte le tuple (nonostante sia solo una in questo caso) aventi t[A] = a2 hanno anche t[B] = b2

- Consideriamo la seguente situazione:

$$F = \{A \rightarrow B\}$$

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_1	c_2	d_2
a_2	b_2	c_1	d_3

Tale istanza è *illegale* su F, poiché non soddisfa le dipendenze funzionali in F: la seconda e la terza tupla hanno lo stesso valore in A ma non in B

Chiusura di F

💡 Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.
Definiamo come **chiusura di F**, indicata con F+, l'insieme di **tutte** le dipendenze funzionali, incluse quelle non in F, soddisfatte da ogni istanza di R legale su F.

F+ = ∩ r∈L {f dipendenza funzionale | r soddisfa f}

dove L = {r istanza di R | r legale su F}. In generale, quindi, si ha che F ⊆ F+.

Chiavi

💡 Dato uno schema di relazione R e un insieme F di dipendenze funzionali definite su R:
Un sottoinsieme K di uno schema di relazione R è una chiave di R se:

1. K→R ∈ F+
2. non esiste un sottoinsieme proprio K'⊆ K tale che K'→R

Chiave Primaria

💡 Dato uno schema R e un insieme F di dipendenze funzionali, è possibile che ci siano diverse chiavi di R. In SQL, ne verrà scelta una come chiave primaria (la quale non può essere assegnata un valore nullo).

Esempio:

- Consideriamo la seguente relazione su cui sono definite le seguenti dipendenze funzionali:

$$F = \{A \to B, B \to C\}$$

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_2
a_2	b_2	c_1	d_3

Tale istanza è legale su F, poiché soddisfa tutte le dipendenze funzionali in F. Inoltre, è soddisfatta anche la dipendenza funzionale A → C, che tuttavia non è in F. Dunque, si ha che A → B,B → C,A → C ∈ F+

Dato uno schema R e un insieme F di dipendenze funzionali definite su R, si ha che:

$$Y \subseteq X \subseteq R \Rightarrow X \rightarrow Y \in F+$$

Tali dipendenze funzionali vengono dette **banali**, poiché soddisfatte da ogni istanza di R.

Preposizione:

Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.
Dati X, Y $\subseteq$ R, si ha che:

$$X \rightarrow Y \in F+ \leftrightarrow \forall A \in Y, X \rightarrow A \in F+$$

Dimostrazione:

- Siano X, Y $\subseteq$ R, dove Y = {A1, . . . ,Ak} .
- Data r una qualsiasi istanza di R, si ha che:

$$\forall A_i \in Y, X \rightarrow A \in F^+ \iff \begin{cases} \forall t_1, t_2 \in r, t_1[X] = t_2[X] \implies t_1[A_1] = t_2[A_1] \\ \vdots \\ \forall t_1, t_2 \in r, t_1[X] = t_2[X] \implies t_1[A_k] = t_2[A_k] \end{cases} \iff$$

$$\iff \forall t_1, t_2 \in r, t_1[X] = t_2[X] \implies t_1[\{A_1, \ldots, A_k\}] = t_2[\{A_1, \ldots, A_k\}]$$

$$\iff \forall t_1, t_2 \in r, t_1[X] = t_2[X] \implies t_1[Y] = t_2[Y] \iff X \rightarrow Y \in F^+$$

Esempio:

- Consideriamo la seguente istanza di uno schema R = {A,B,C}:

A	B	C
a_1	b_1	c_1
a_1	b_1	c_1
a_2	b_2	c_1

Dato F un insieme di dipendenze funzionali definite su R, notiamo facilmente, che tutte le tuple di tale istanza soddisfano la dipendenza ABC → ABC ∈F+.

Notiamo inoltre che tutte le tuple di tale istanza in cui A e B sono uguali anche A è uguale, dunque AB → A ∈ F+.

Procedendo analogamente, in definitiva si ha che:

$$\left.\begin{array}{l} ABC \to ABC, ABC \to AB, ABC \to AC, ABC \to BC, \\ ABC \to A, ABC \to B, ABC \to C, AB \to AB, AB \to A, \\ AC \to A, AC \to C, BC \to A, BC \to C, A \to A, B \to B, C \to C \end{array}\right\} \in F^+$$

▼ Lezione del 25/10

Assiomi di Armstrong

Sia *R* uno schema e sia *F* un insieme di dipendenze funzionali definite su R.
Definiamo come F^A l'insieme di tutte le dipendenze funzionali ottenivili partendo da *F* applicando i seguenti **Assiomi di Armstrong:**
- Inclusione iniziale ($F \subseteq F^A$):

$$X \to \in F \Rightarrow x \to \in F^A$$

-Assioma di riflessività:

$$Y \subseteq X \subseteq R \Rightarrow X \to Y \in F^A$$

-Assioma di aumento:

$$Z \subseteq R, X \to Y \in F^A \Rightarrow XZ \to YZ \in F^A$$

-Assioma di transitività:

$$X \to Y \in F^A \wedge Y \to Z \in F^A \Rightarrow X \to Z \in F^A$$

Regole secondarie di Armstrong

Dato uno schema R e un insieme F di dipendenze funzionali definite su R, tramite gli assiomi di Armstrong è possibile ricavare le seguenti regole aggiuntive:

-**Regola dell'unione:**

$$X \to Y \in F^A \land X \to Z \in F^A \Rightarrow X \to YZ \in F^A$$

-Regola della decomposizione:

$$Z \subseteq Y, X \to Y \in F^A \Rightarrow X \to Z \in F^A$$

-**Regola della pseudo-transitività:**

$$X \to Y \in F^A \land WY \to Z \in F^A \Rightarrow WX \to Z \in F^A$$

Dimostrazione:

- **Regola dell'unione:**
  - Siano $X \to Y, X \to Z \in F^A$.
  - Per assioma di aumento, si ha che:

$$X \subseteq R, X \to Y \in F^A \implies XX \to XY = X \to XY \in F^A$$

  - Analogamente, si ha che:

$$Y \subseteq R, X \to Z \in F^A \implies XY \to ZY = XY \to YZ \in F^A$$

  - Infine, per assioma di transitività si ha che:

$$X \to XY \in F^A \land XY \to YZ \in F^A \implies X \to YZ \in F^A$$

- **Regola della decomposizione:**

- Sia $Z \subseteq Y \subseteq R$ e sia $X \to Y \in F^A$.
- Per assioma di riflessività, si ha che:

$$Z \subseteq Y \subseteq R \implies Y \to Z \in F^A$$

- Infine, per assioma di transitività si ha che:

$$X \to Y \in F^A \land Y \to Z \in F^A \implies X \to Z \in F^A$$

- **Regola della pseudo-transitività:**
  - Sia $X \to Y, YW \to Z \in F^A$.
  - Per assioma di aumento, si ha che:

$$W \subseteq R, X \to Y \in F^A \implies XW \to YW \in F^A$$

  - Infine, per assioma di transitività si ha che:

$$XW \to YW \in F^A \land YW \to Z \in F^A \implies XW \to Z \in F^A$$

Proposizione

Dato uno schema R e un insieme F di dipendenze funzionali definite su R, si ha che:

$$X \to Y \in F^A \iff \forall A \in Y, X \to A \in F^A$$

Dimostrazione:

- Siano $X, Y \subseteq R$, dove $Y = \{A_1, \ldots, A_k\}$.
- Per la regola dell'unione, si ha che:

$$\forall A_i \in Y, X \to A \in F^A \implies X \to \{A_1, \ldots, A_k\} = Y \in F^A$$

- Per la regola della decomposizione, invece si ha che:

$$X \to Y \in F^A \implies \forall A \in Y, X \to A \in F^A$$

Chiusura di X

Chiusura di un insieme di attributi

Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.
Dato $X \subseteq R$, definiamo come **chiusura di X rispetto ad F,** indicata con X_F^+ (o solo con X^+ se F è l'unico insieme di dipendenze su R), il seguente insieme:

$$X_F^+ = \{ A \in R | X \to A \in F^A \}$$

Dove $A \in R$ implica che A sia un singolo attributo di R

Rappresenta quindi l'insieme degli elementi funzionalmente determinati da X anche dopo aver applicato i teoremi di Armstrong

Lemma 1

📖 Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.
Dato $X, Y \subseteq R$, si ha che:

$$X \to Y \in F^A \iff Y \subseteq X^+$$

Dimostrazione:

- Dato $Y = \{A_1, \ldots, A_k\}$, si ha che:

$$X \to Y \in F^A \iff X \to A \in F^A, \forall A \in Y \iff A \in X^+, \forall A \in Y \iff Y \subseteq X^+$$

Corollario 1

⚠️ Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.
Dato $X \subseteq R$, si ha che $X \to X \in F^A$. Dunque ne segue che:

$$X \to X \in F^A \iff X \subseteq X^+$$

In altre parole, **ogni insieme di attributi è un elemento della sua chiusura.**

$$F^+ = F^A$$

Teorema 1: $F^+ = F^A$

💡 Dato uno schema R e un insieme F di dipendenze funzionali definite su R, si ha che:

$$F^+ = F^A$$

Dimostrazione:

DIMOSTRAZIONE PER INDUZIONE SU NOTE QUA SOTTO RIPORTATO ESEMPIO DI EXYSS

- Dimostriamo che $F^A \subseteq F^+$:
  - **Caso base** $(n = 0)$: se $X \to Y \in F^A$ senza aver applicato alcun assioma di Armstrong, allora l'unica possibilità è che:

    $$X \to Y \in F^A \iff X \to Y \in F$$

    Siccome $X \to Y \in F$, allora

    $$X \to Y \in F \implies X \to Y \in F^+$$

– **Ipotesi induttiva forte**: ogni dipendenza funzionale in F^A ottenuta da F applicando $k \leq n$ assiomi di Armstrong è anche in F^+

$$X \to Y \in F^A \text{ tramite } k \leq n \text{ assiomi} \implies X \to Y \in F^+$$

– **Passo induttivo** ($n > 0$): è necessario dimostrare che se $X \to Y \in F^A$ dopo aver applicato $n+1$ assiomi di Armstrong, allora $X \to Y \in F^+$.

È possibile ritrovarsi in uno dei seguenti tre casi:

1. Se l'$(n+1)$-esimo assioma applicato è l'assioma di riflessività, allora l'unica possibilità è che $X \to Y \in F^A \iff Y \subseteq X \subseteq R$.

   Dunque, poiché, $Y \subseteq X \subseteq R$, per ogni istanza legale di R si ha che:

   $$\forall t_1, t_2 \in r, t_1[X] = t_2[X] \implies t_1[Y] = t_2[Y]$$

   da cui ne segue automaticamente che $X \to Y \in F^+$:

2. Se l'$(n+1)$-esimo assioma applicato è l'assioma di aumento, allora è obbligatoriamente necessario che:

   * $\exists V, W \subseteq R \mid \exists V \to W \in F^A$, ottenuta applicando $j \leq n$ assiomi di Armstrong
   * $\exists Z \subseteq R \mid X := VZ, Y := WZ$

   affinché si abbia che:

   $$Z \subseteq R, V \to W \implies VZ \to WZ = X \to Y \in F^A$$

   Siccome per ipotesi induttiva si ha $V \to W \in F^A \implies V \to W \in F^+$ e siccome $Z \subseteq Z \implies Z \to Z \in F^+$, si vede facilmente che:

   $$\begin{cases} V \to W \in F^+ \\ Z \to Z \in F^+ \end{cases} \implies \begin{cases} \forall t_1, t_2 \in r, t_1[V] = t_2[V] \implies t_1[W] = t_2[W] \\ \forall t_1, t_2 \in r, t_1[Z] = t_2[Z] \implies t_1[Z] = t_2[Z] \end{cases} \implies$$

   $$\implies \forall t_1, t_2 \in r, t_1[VZ] = t_2[VZ] \implies t_1[WZ] = t_2[WZ] \implies$$

   $$\implies \forall t_1, t_2 \in r, t_1[X] = t_2[X] \implies t_1[Y] = t_2[Y] \implies X \to Y \in F^+$$

3. Se l'$(n+1)$-esimo assioma applicato è l'assioma di transitività, allora è obbligatoriamente necessario che $\exists X \to Z, Z \to Y \in F^A$, ottenute con $k \leq n$ assiomi di Armstrong, affinché si abbia che:

   $$X \to Z \in F^A \wedge Z \to Y \in F^A \implies X \to Y \in F^A$$

Siccome per ipotesi induttiva $X \to Z \in F^A \implies X \to Z \in F^+$ e $Z \to Y \in F^A \implies Z \to Y \in F^+$, si vede facilmente che:

$$\begin{cases} X \to Z \in F^+ \\ Z \to Y \in F^+ \end{cases} \implies \forall t_1, t_2 \in r, t_1[X] = t_2[X]$$

$$\implies t_1[Z] = t_2[Z] \implies t_1[Y] = t_2[Y] \implies X \to Y \in F^+$$

- Dimostriamo che $F^+ \subseteq F^A$:
  - Sia $X \subseteq R$ e sia r istanza di $R(X^+, R - X^+)$ tale che

X^+			$R - X^+$		
A_1	$\cdots$	A_i	A_j	$\cdots$	A_n
1	...	1	1	...	1
1	...	1	0	...	0

    dunque tale che $\forall t_1, t_2 \in r$ si ha:
    - $t_1[X^+] = (1, \ldots, 1) = t_2[X^+]$
    - $t_1[R - X^+] = (1, \ldots, 1) \neq (0, \ldots, 0) = t_2[R - X^+]$
  - Notiamo che $\forall V, W \subseteq R \mid V \to W \in F$ si ha che:
    - Se $V \cap R - X^+ \neq \emptyset$ (dunque anche se $V \subseteq R - X^+$) allora $t_1[V] \neq t_2[V]$, dunque r soddisfa $V \to W \in F$
    - Se invece $V \subseteq X^+$, per il lemma precedentemente visto si ha che

$$V \subseteq X^+ \iff X \to V \in F^A$$

      Siccome $V \to W \in F \implies V \to W \in F^A$, per transitività si ha che

$$X \to V \in F^A \land V \to W \in F^A \implies X \to W \in F^A \iff W \subseteq X^+$$

      Dunque, siccome $V, W \subseteq X^+$, in definitiva si ha che

$$t_1, t_2 \in r, t_1[V] = (1, \ldots, 1) = t_2[V] \land t_1[W] = (1, \ldots, 1) = t_2[W]$$

      e quindi r soddisfa ogni $V \to W \in F$
  - Siccome in entrambi i casi r soddisfa ogni $V \to W \in F$, allora r è legale.
  - A questo punto, una qualsiasi dipendenza $X \to Y \in F^+$ deve essere soddisfatta da qualsiasi istanza legale di R, inclusa r stessa
  - Poiché $X \subseteq X^+$, ne segue che la dipendenza non può essere soddisfatta a vuoto poiché $t_1[X] = t_2[X]$. Dunque, l'unica possibilità affinché $X \to A \in F^+$ sia soddisfatta da r è che $Y \subseteq X^+$ in modo che si abbia $t_1[Y] = t_2[Y]$
  - A questo punto, per il lemma si ha che $Y \subseteq X^+ \iff X \to Y \in F^A$
  - Dunque, siccome $X \to Y \in F^+ \implies X \to Y \in F^A$, concludiamo che $F^+ \subseteq F^A$

Osservazione

💡 Poiché $F^+ = F^A$, per **calcolare** F^+ ci basta applicare gli assiomi di Armstrong sulle dipendenze in F in modo da trovare F^A.
Tuttavia, calcolare $F^+ = F^A$ richiede **tempo esponenziale** quindi $\Omega(2^R)$: considerando anche solo l'assioma di riflessività, siccome ogni possibile sottoinsieme di R genera una dipendenza e siccome i sottoinsiemi possibili di R sono 2^R allora ne segue che $|F^+|>>2^R$

▼ Lezione del 26/10

Terza Forma Normale (3NF)

A questo punto, possiamo sfruttare la definizione di dipendenza funzionale per dare una definizione più rigorosa di chiave:

Chiave e Primo

💡 Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.
Definiamo il sottoinsieme di attributi $K \subseteq R$ come **Chiave** di R se:
- $K \to R \in F^+$
- $\nexists K' \subset K | K' \to R \in F^+$

Se K è una chiave di R, ogni attributo $A \in K$ viene detto **primo.**

Esempio:

- Consideriamo lo schema `Student(Matr, LastName, FirstName, BirthD)`
- In questo caso, è ovvio imporre la seguente dipendenza funzionale in F:

$$\texttt{Matr} \to \{\texttt{LastName, FirstName, BirthD}\} \in F \subseteq F^+$$

  poiché ogni tupla avente matricola uguale deve anche avere informazioni uguali.
- Siccome $\texttt{Matr} \subseteq \texttt{Matr}^+ \iff \texttt{Matr} \to \texttt{Matr} \in F^+ = F^A$, per unione si ha che:

$$\texttt{Matr} \to \{\texttt{Matr, LastName, FirstName, BirthD}\} \in F^+ = F^A$$

  dunque `Matr` è superchiave di `Student` poiché determina tutto il suo schema
- Siccome non esiste alcun sottoinsieme di `Matr`, allora possiamo concludere che `Matr` sia chiave di `Student`

Superchiave

Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.
Definiamo il sottoinsieme di attributi $K \subseteq R$ come **Superchiave** di R se:

- $K \to R \in F^+$
- $\exists K' \subseteq K | K' \to R \in F^+ \wedge \nexists K'' \subset K' | K'' \to R \in F^+$
  ossia contiene una chiave

Osservazione

Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.
Se $X \subseteq R$ è chiave di R, allora essa è anche superchiave, poiché $\exists X \subseteq X$ tale che X chiave di R

X chiave di R$\Rightarrow X$ superchiave di R

Terza Forma Normale (3NF)

Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.
Lo schema R viene detto in **terza forma normale (3NF)** se:

$$\forall X \to A \in F^+, A \in R - X, \exists K \subseteq R \text{ chiave} \mid K \subseteq X \vee A \in K$$

In altre parole, uno schema viene detto in terza forma normale se per ogni dipendenza funzionale **non banale** $X \to A \in F^+$, il determinante di X è superchiave o il determinante di A è primo.
Se uno scema è in 3NF, la quantità di **anomalie** e di **ridondanze** dei dati è **estremamente ridotta.**

Esempio:

1.
   - Sia $R = ABCD$ uno schema e sia $F = \{A \to B, B \to CD\}$ un insieme dipendenze funzionali su R
   - Applicando gli assiomi di Armstrong, si ha che:
     - Per riflessività:
       $$A \subseteq A \implies A \to A \in F^A$$
     - Per transitività:
       $$A \to B, B \to CD \in F^A \implies A \to CD \in F^A$$
     - Per unione:
       $$A \to A, A \to B, A \to CD \in F^A \implies A \to ABCD = R \in F^A$$

- Dunque, siccome $A \to R \in F^A = F^+$ e siccome A non ha sottoinsiemi, allora A è chiave di R (in particolare, A è l'unica chiave di R)
- Verifichiamo quindi se R sia in 3NF:
  - $A \to B \in F^+$ rispetta la definizione di 3NF, poiché il determinante A è chiave (e quindi anche una superchiave di se stessa)
  - $B \to CD \in F^+$ non è una dipendenza da controllare, poiché CD sono un sottoinsieme di attributi e non un singolo attributo
  - Tuttavia, per decomposizione abbiamo che $B \to CD \in F^A = F^+ \implies B \to C, B \to D \in F^A = F^+$
  - Per entrambe si ha che B non è superchiave, poiché $A \not\subseteq B$, mentre C e D non sono primi, poiché $C, D \notin A$, dunque concludiamo che R non sia in 3NF

2.
- Sia $R = ABCD$ e sia $F = \{AC \to B, B \to AD\}$ un insieme di dipendenze funzionali su R.
- Applicando gli assiomi di Armstrong, si ha che:
  - Per riflessività:

$$A, C, AC \subseteq AC \implies AC \to A, AC \to C, AC \to AC \in F^A$$

$$B, C, BC \subseteq BC \implies BC \to B, BC \to C, BC \to BC \in F^A$$

  - Per transitività:

$$AC \to B, B \to AD \in F^A \implies AC \to AD \in F^A$$

$$BC \to B, B \to AD \in F^A \implies BC \to AD \in F^A$$

  - Per unione:

$$AC \to C, AC \to B, AC \to AD \in F^A \implies AC \to ABCD = R \in F^A$$

$$BC \to B, BC \to C, BC \to AD \in F^A \implies BC \to ABCD \in F^A$$

  - Per aumento:

$$AC \to ABCD = R \in F^A \implies ABC \to ABBCD = ABCD = R \in F^A$$

- Deduciamo quindi che AC e BC siano chiave di R, mentre ABC è una superchiave di R
- Verifichiamo quindi se R sia in 3NF:
  - $AC \to B \in F^A = F^+$ rispetta la definizione di 3NF, poiché AC è chiave
  - $B \to AD \in F^A = F^+$ non va controllato, ma per decomposizione si ha che $B \to A, B \to D \in F^A = F^+$

- $B \to A \in F^+$ rispetta la definizione di 3NF, poiché $A \in AC$ e dunque primo, mentre $B \to D \in F^+$ non rispetta la definizione di 3NF, poiché né B è superchiave né D è primo, dunque concludiamo che R non sia in 3NF

3.
   - Sia $R = ABCD$ uno schema e sia $F = \{AB \to CD, BC \to A, D \to AC\}$ un insieme di dipendenze funzionali su R
   - In tal caso si ha che AB, BC e BD sono chiavi di R
   - Verifichiamo quindi se R sia in 3NF:
     - $AB \to CD \in F^+$ rispetta la definizione di 3NF, poiché AB è chiave
     - $BC \to A \in F^+$ rispetta la definizione di 3NF, poiché BC è chiave e A è primo
     - $D \to AC \in F^+ \implies D \to A, D \to C \in F^+$, i quali rispettano entrambi la definizione di 3NF, poiché A e C sono entrambi primi
     - Dunque, concludiamo che R sia in 3NF

Dipendenza parziale

Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.
Definiamo $X \to A \in F^+$, dove $A \notin X$, come **dipendenza parziale** su R se:
- A **non** è primo
- $\exists K \subseteq R$ chiave di R tale che $X \subset K$ (quindi in particolare X≠K)

Dipendenza transitiva

💡 Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.
Definiamo $X \rightarrow A \in F^{+}$,dove $A \notin X$, come **dipendenza transitiva** in R se:
- A **non** è primo
- $\forall K \subseteq R$ chiave di R si ha che $X \not\subset K$ e K - X ≠0

Corollario 2: Definizione alternativa di 3NF

💡 Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.
Lo schema di R viene detto **in terza forma normale (3NF)** se non esistono dipendenze parziali o transitive su F

$\nexists X \rightarrow Y \in F | X \rightarrow Y$ dipendenza parziale o transitiva.