

BASI DI DATI

Prof. Vaglini

INDICE VAGLINI-TEORIA+ESERCIZI

Pagina 7
<ul style="list-style-type: none">• DBMS• ARCHITETTURA CLIENT-SERVER
Pagina 8
<ul style="list-style-type: none">• DATABASE RELAZIONALI E DATABASE NoSQL
Pagina 9
<ul style="list-style-type: none">• MODELLO RELAZIONALE
Pagina 10
<ul style="list-style-type: none">• VINCOLI• SEMANTICA, CALCOLO RELAZIONALE E ALGEBRA RELAZIONALE
Pagina 11
<ul style="list-style-type: none">• OTTIMIZZAZIONE ALGEBRICA• CALCOLO SU DOMINI, CALCOLO SU TUPLE, EQUIVALENZE CALCOLO-ALGEBRA
Pagina 12
<ul style="list-style-type: none">• SCHEMI DI BASI DI DATI• MODELLO CONCETTUALE E-R
Pagina 14
<ul style="list-style-type: none">• VINCOLI DI INTEGRITÀ GENERICI
Pagina 15
<ul style="list-style-type: none">• TRIGGER• PRIVILEGI
Pagina 16
<ul style="list-style-type: none">• RISTRUTTURAZIONE
Pagina 19
<ul style="list-style-type: none">• TRADUZIONE DA E-R VERSO IL MODELLO LOGICO-RELAZIONALE
Pagina 21
<ul style="list-style-type: none">• DIPENDENZE FUNZIONALI E CONCETTO DI NORMALIZZAZIONE
Pagina 22
<ul style="list-style-type: none">• CONCETTI DI CHIUSURA, IMPLICAZIONE E SUPERCHIAVE
Pagina 23
<ul style="list-style-type: none">• CONCETTI DI EQUIVALENZA DI ESPRESSIONI, CHIUSURA SULL'INSIEME DI ATTRIBUTI E MINIMALITÀ• FORME NORMALI
Pagina 26
<ul style="list-style-type: none">• TRANSAZIONI
Pagina 28
<ul style="list-style-type: none">• GESTORE AFFIDABILITÀ E TIPI DI MEMORIE• LOG, CHECKPOINT E DUMP
Pagina 30
<ul style="list-style-type: none">• GUASTI, RIPRESA A CALDO E RIPRESA A FREDDO• ANOMALIE DURANTE LA CONCORRENZA• SCHEDULE, SCHEDULER E CONTROLLO DI CONCORRENZA
Pagina 31
<ul style="list-style-type: none">• SCHEDULE SERIALE E SERIALIZZABILE• VIEW-EQUIVALENZA, VIEW-SERIALIZZABILITÀ E VSR• CONFLICT-SERIALIZZABILITÀ
Pagina 32
<ul style="list-style-type: none">• LOCK
Pagina 34
<ul style="list-style-type: none">• 2PL
Pagina 35
<ul style="list-style-type: none">• S2PL

INDICE VAGLINI-TEORIA+ESERCIZI

Pagina 36
• TS
Pagina 37
• TS CONFRONTATO CON 2PL, TIMEOUT E STARVATION
Pagina 38
• GRAFICO RIEPILOGATIVO PER LE TECNICHE DI GESTIONE DELLA CONCORRENZA
PARTE DI ESERCITAZIONE – PREPARAZIONE ESERCIZI
Pagina 39
• UNIONE
• INTERSEZIONE
• DIFFERENZA
• RIDENOMINAZIONE
Pagina 40
• SELEZIONE
• CONDIZIONI CON NULL
Pagina 41
• PROIEZIONE
• PRODOTTO CARTESIANO
• JOIN NATURALE
Pagina 42
• THETA JOIN, EQUI-JOIN
• UTILIZZO DEL MULTI JOIN
• ESEMPIO DI QUERY IN ALGEBRA RELAZIONALE
Pagina 44
• TRASFORMAZIONI IN ESPRESSIONI EQUIVALENTI
• CALCOLO SUI DOMINI
Pagina 45
• CALCOLO SU TUPLE
• ESEMPI DI CALCOLO SU DOMINI E SU TUPLE
• QUANTIFICATORI ESISTENZIALI E UNIVERSALI
Pagina 46
• ESERCIZIO ALGEBRA RELAZIONALE
• ESERCIZIO ALGEBRA RELAZIONALE-CALCOLO SU DOMINI
Pagina 47
• ESERCIZIO ALGEBRA RELAZIONALE-CALCOLO SU TUPLE
• ESERCIZIO CALCOLO SUI DOMINI
Pagina 48
• ESERCIZIO ALGEBRA RELAZIONALE
• LEFT OUTER JOIN
• RIGHT OUTER JOIN
• FULL OUTER JOIN
Pagina 49
• ESERCIZIO JOIN ESTERNO
Pagina 50
• PROIEZIONE GENERALIZZATA
• OPERATORI DI AGGREGAZIONE: SOMMA, CONTEGGIO, CONTEGGIO SENZA DUPLICATI, MINIMO E MASSIMO
• ESEMPI DI OPERATORI DI AGGREGAZIONE
Pagina 51
• RAGGRUPPAMENTO
• DIVISIONE
• VISTA

INDICE VAGLINI-TEORIA+ESERCIZI


Pagina 52
<ul style="list-style-type: none">• VISTA 'MULTIPLA' E RIDENOMINAZIONE SU UNA VISTA• ESERCIZIO CON QUANTIFICATORI
Pagina 53
<ul style="list-style-type: none">• ESERCIZIO: NON APPARTENENZA IN ALGEBRA RELAZIONALE• ESERCIZIO: DIVERSITÀ NEL JOIN IN ALGEBRA RELAZIONALE• ESERCIZIO: DIVERSITÀ NEL JOIN SUL CALCOLO DEI DOMINI• ESERCIZIO: DIVERSITÀ NEGLI ATTRIBUTI SUL CALCOLO DEI DOMINI• ESERCIZIO: NON ESISTENZA ALGEBRA RELAZIONALE-CALCOLO DEI DOMINI
Pagina 54
<ul style="list-style-type: none">• ESERCIZIO: RIDENOMINAZIONI ALGEBRA RELAZIONALE-CALCOLO DEI DOMINI• TRADUZIONE GENERALIZZAZIONI
Pagina 55
<ul style="list-style-type: none">• INSERIMENTO CARDINALITÀ NELL'E-R
Pagina 57
<ul style="list-style-type: none">• TAVOLA DEI VOLUMI: OCCORRENZE DEL PADRE DI UNA GENERALIZZAZIONE
Pagina 58
<ul style="list-style-type: none">• TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ENTITÀ (CARDINALITÀ NON CONSIDERATA)• TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ENTITÀ CON LE OCCORRENZE DELL'ASSOCIAZIONE MANCANTI
Pagina 59
<ul style="list-style-type: none">• TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ASSOCIAZIONE (CARDINALITÀ NON CONSIDERATA)
Pagina 60
<ul style="list-style-type: none">• TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ASSOCIAZIONE CON RICORSIONE (CARDINALITÀ NON CONSIDERATA)
Pagina 61
<ul style="list-style-type: none">• TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ASSOCIAZIONE AVENTE UNA CARDINALITÀ (1,1)
Pagina 62
<ul style="list-style-type: none">• TAVOLA DEI VOLUMI: OCCORRENZE IN MEDIA TRA DUE ENTITÀ (CARDINALITÀ NON CONSIDERATA)
Pagina 63
<ul style="list-style-type: none">• TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ASSOCIAZIONE E DUE PERCORSI (CARDINALITÀ NON CONSIDERATA)
Pagina 65
<ul style="list-style-type: none">• COSTO OPERAZIONI DI LETTURA E SCRITTURA• OPERAZIONI ELEMENTARI TRA DUE ENTITÀ (CARDINALITÀ NON CONSIDERATA)
Pagina 66
<ul style="list-style-type: none">• OPERAZIONI ELEMENTARI TRA TRE ENTITÀ (CARDINALITÀ NON CONSIDERATA)
Pagina 68
<ul style="list-style-type: none">• CALCOLO TOTALE DELLE OPERAZIONI ELEMENTARI (LETTURA/SCRITTURA)
Pagina 69
<ul style="list-style-type: none">• FREQUENZA NEL CALCOLO OPERAZIONI ELEMENTARI (LETTURA/SCRITTURA)• CONFRONTO FREQUENZE NEL CALCOLO OPERAZIONI ELEMENTARI (LETTURA/SCRITTURA)
Pagina 70
<ul style="list-style-type: none">• FREQUENZA NEL CALCOLO OPERAZIONI ELEMENTARI (LETTURA/SCRITTURA) E RIDONDANZE ATTRIBUTI
Pagina 71
<ul style="list-style-type: none">• REGOLE DI ARMSTRONG
Pagina 72
<ul style="list-style-type: none">• TROVARE LA CHIAVE CON LE REGOLE DI ARMSTRONG• CALCOLO EQUIVALENZA CON LE REGOLE DI ARMSTRONG

INDICE VAGLINI-TEORIA+ESERCIZI

Pagina 74
• CHIUSURA SULL'INSIEME DI ATTRIBUTI DI UNA RELAZIONE X^+
Pagina 75
• EQUIVALENZA CON LA CHIUSURA SULL'INSIEME DI ATTRIBUTI
Pagina 76
• RIDONDANZE
Pagina 77
• DIPENDENZA FUNZIONALE SEMPLICE (STANDARD)
• ATTRIBUTI ESTRANEI
Pagina 78
• ALGORITMO DI MINIMIZZAZIONE
Pagina 79
• ESERCIZIO MINIMIZZAZIONE
• CHIAVE DI UN INSIEME DI DIPENDENZE (CON CHIUSURA)
Pagina 81
• VERIFICA BCNF (CON DECOMPOSIZIONE)
Pagina 84
• RECORD DI UNA TRANSAZIONE (FILE DI LOG)
Pagina 85
• UNDO E REDO
• RIPRESA A CALDO
Pagina 86
• APPLICAZIONE RIPRESA A CALDO
Pagina 87
• RIPRESA A FREDDO
• APPLICAZIONE RIPRESA A FREDDO
Pagina 88
• VIEW-EQUIVALENZA E VIEW-SERIALIZZABILITÀ TRA DUE SCHEDULE
• APPLICAZIONE VIEW-EQUIVALENZA E VIEW-SERIALIZZABILITÀ TRA DUE SCHEDULE
Pagina 89
• CONFLICT-SERIALIZZABILITÀ E CONFLICT-EQUIVALENZA DI UNO SCHEDULE
Pagina 90
• APPLICAZIONE CONFLICT-SERIALIZZABILITÀ E CONFLICT-EQUIVALENZA DI UNO SCHEDULE
Pagina 91
• CSR, VSR E CONFLICT-EQUIVALENZA DI UNO SCHEDULE
Pagina 92
• CSR, VSR, VIEW-EQUIVALENZA E CONFLICT-EQUIVALENZA TRA DUE SCHEDULE
• APPLICAZIONE CSR, VSR, VIEW-EQUIVALENZA E CONFLICT-EQUIVALENZA TRA DUE SCHEDULE
Pagina 98
• TIMESTAMP
Pagina 99
• ESEMPIO DI TIMESTAMP
Pagina 100
• TIMESTAMP SU SCHEDULE
Pagina 101
• APPLICAZIONE TIMESTAMP SU SCHEDULE
Pagina 102
• S2PL
Pagina 105
• APPLICAZIONE S2PL

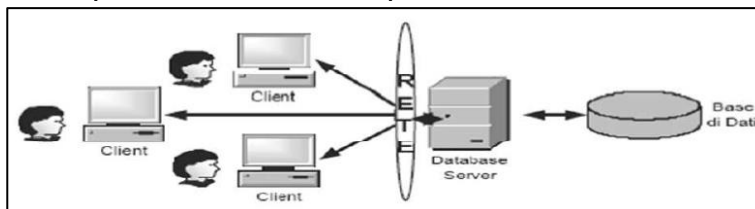
VAGLINI-NOZIONI DI TEORIA

DBMS

- Il **DBMS** è il gestore della base di dati:
 - Si trova tra i programmi e il sistema operativo
- 
- Il diagramma mostra una gerarchia a quattro livelli. In alto c'è un rettangolo con "software applicativo". Sotto di esso, un rettangolo con "DBMS". Sotto quello, un rettangolo con "sistema operativo". In fondo, un cilindro con "database (unico)".
- Fa interagire i programmi e il sistema operativo attraverso i loro dati, i quali sono contenuti in un **catalogo** (o **dizionario**), il quale a sua volta, contiene la descrizione dei dati stessi
- Il DBMS garantisce:
 - Privatezza
 - Efficienza
 - Efficacia
 - Affidabilità
- Il **DBMS relazionale** è sempre **transazionale**, poiché offre supporto alla **transazione**, la quale gode della proprietà **ACID**:
 - Di conseguenza anche i DBMS relazionali transazionali garantiscono le proprietà ACID

ARCHITETTURA CLIENT-SERVER

- Un'**architettura distribuita** è un collegamento tra due macchine (computer), per mezzo dell'utilizzo della rete
- È un'architettura distribuita composta, da:
 - Un processo **CLIENT** che richiede i servizi tramite interrogazioni (ruolo attivo, che genera tante richieste)
 - Da un processo **SERVER** che offre i servizi eseguendo le interrogazioni generate dai CLIENT (ruolo passivo, si limita a rispondere alle richieste del CLIENT)



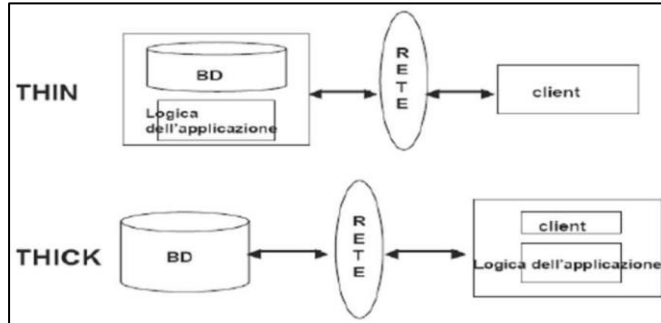
- I processi del CLIENT e del SERVER in genere risiedono su macchine diverse, che sono però collegate in rete
- Il SERVER solitamente è unico, ma se ce n'è più di uno:
 - Si parla quindi di architettura completamente distribuita
- Nell'architettura a due livelli (**two tier**), c'è un livello CLIENT e un livello SERVER



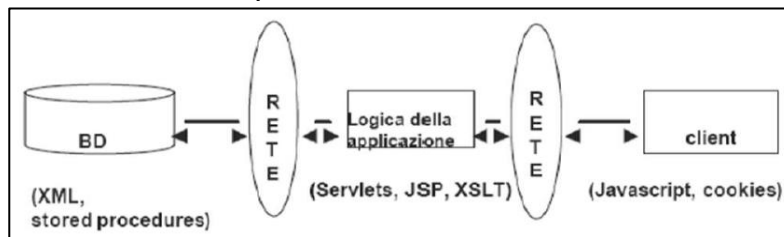
VAGLINI-NOZIONI DI TEORIA

...CONTINUO ARCHITETTURA CLIENT-SERVER

- Il CLIENT dell'architettura a due livelli può essere:
 - **Thin**, dove le applicazioni risiedono nel SERVER (più diffuso)
 - **Thick** dove le applicazioni risiedono nel CLIENT



- Nell'architettura a tre livelli (**three tier**), c'è un secondo SERVER, detto **server applicativo**:
 - Esso sta tra il CLIENT e il SERVER, e separa le applicazioni tra questi due e gestisce la logica applicativa sui CLIENT
 - Un grande vantaggio è quello di connettere sistemi eterogenei, con SERVER non dello stesso tipo (alta scalabilità)



DATABASE RELAZIONALI E DATABASE NoSQL

- I database scalano:
 - In maniera **verticale**, cioè su server centralizzato e potente
 - In maniera **orizzontale**, cioè su server distribuiti ma di tipologia standard
- OSSERVAZIONE: Nella scalabilità **orizzontale** si potrebbero aggiungere anche server potenti, ma in generale l'importante è che i server aumentino in quantità, in modo da avere una rete di server, tutti più o meno equivalenti tra loro
- I **database relazionali** si basano sul concetto di relazione matematica e scalano verticalmente
- I **database NoSQL** sono database non relazionali e scalano orizzontalmente
- I database NoSQL sono costruiti in modo da tollerare i fallimenti e recuperare i dati, dato che ogni tanto i server vanno in crash
- Anche se i database NoSQL scalano su più server, le applicazioni che ne fanno utilizzo vedono un solo database distribuito
- I database relazionali garantiscono la consistenza e la disponibilità, sacrificando la scalabilità (più affidabili e meno scalabili rispetto ai sistemi NoSQL)
- I database NoSQL garantiscono la disponibilità e la scalabilità (anche se per il Teorema CAP) sarebbe garantita la consistenza

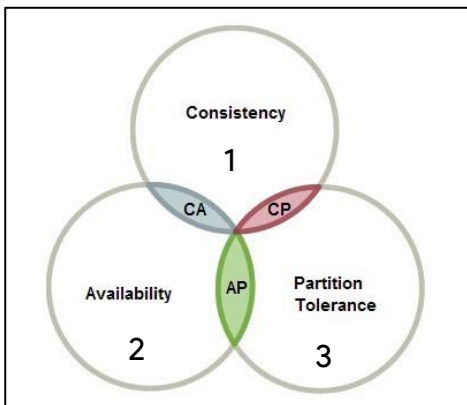


VAGLINI-NOZIONI DI TEORIA

...CONTINUO DATABASE RELAZIONALI E DATABASE NoSQL

- Per il database NoSQL la consistenza si esclude dato che questi database 'prima o poi possono essere consistenti', quindi per la semplicità del database, ci si conduce alla mancanza dei controlli fondamentali sull'integrità dei dati
- Per i database NoSQL non esiste uno standard universale che li faccia interagire tutti tra loro:
 - Ciò porta ogni database non relazionale ad avere le **api**, cioè programmi che utilizzano metodi personali per accedere ai propri dati
- I database NoSQL permettono di realizzare tabelle con le righe di dimensione qualsiasi (con campi di illimitata lunghezza), perché ciò è dovuto al fatto dell'assenza di uno schema (come c'è per i database relazionali)

TEOREMA CAP (DI BREWER)



1. CONSISTENZA
2. DISPONIBILITÀ
3. POSSIBILITÀ DI REPLICA
(Scalabilità sui server)

In un database distribuito è possibile mantenere solo 2 di queste 3 caratteristiche

- I database relazionali privilegiano 1) e 2) ma non 3)
- I database NoSQL privilegiano 3) e alcuni mantengono 1) come **MongoDB** e altri mantengono 2) come **Cassandra**

OSSERVAZIONE: In sede d'esame si preferisce il fatto che ci sia il mantenimento su della scalabilità e della disponibilità, piuttosto che la consistenza

MODELLO RELAZIONALE

- Il **dominio** di una tabella rappresenta gli insiemi dei tipi per ogni riga della tabella, come ad esempio: int, string ecc...
- Il **prodotto cartesiano** dei domini è la combinazione di tutte le righe dei domini stessi
- Una riga della tabella è detta **tupla** (oppure **n-upla**)
- La **relazione** su un insieme di domini è:
 - Un sottoinsieme del prodotto cartesiano dei domini
 - Descritta per mezzo di una tabella, cioè la tabella stessa
- L'**attributo** descrive il ruolo di un dominio di una tabella
- L'**istanza** di una relazione su uno schema X è un insieme di tuple di quello schema X

VAGLINI-NOZIONI DI TEORIA

VINCOLI

- Un **vincolo di integrità** è un predicato che associa ad ogni istanza della base di dati, il valore vero o falso, che corrisponde alla correttezza di una relazione, creata in fase di progettazione e riferente alla realtà interessata
- Vincoli **intrarelazionali** possono essere:
 - Vincoli **di tupla** (sui valori di ciascuna tupla)
 - Vincoli **di dominio** (su un singolo attributo)
- **Chiave**: Insieme di attributi che identificano univocamente una singola tupla di una relazione, e da questi attributi non possono togliere nulla (si definisce in fase DDL)
- La chiave di ogni tupla di una tabella fa sì che, tutte le tuple siano diverse fra loro
- **Chiave Primaria**: chiave in cui non sono ammessi valori nulli (P.K.) ed è sottolineata!
- Vincoli **di chiave**: sono le **dipendenze funzionali**
- Vincoli **interrelazionali** sono dei vincoli fra tabelle diverse e quindi:
 - Si chiamano anche **Vincoli di integrità referenziale** (V.I.R.)
- I V.I.R. permettono di collegare i valori di un attributo di una tabella con quelli di un attributo di un'altra tabella, o della stessa tabella (attraverso le P.K.)
→ Il fatto che ci si possa riferire alla stessa tabella, lo si può notare con un vincolo di integrità referenziale presente in un'associazione ricorsiva
- Ci sono dei vincoli di P.K. che identificano la chiave primaria, come ad esempio sul valore NULL, il quale non va messo su una P.K.
- L'utilizzo del V.I.R. può fare sì che la P.K. di una tabella si formi attraverso l'attributo di questa tabella e la P.K. di un'altra tabella
- I V.I.R. vengono valutati dal DBMS prima di ogni tentativo di modifica di dati contenuti in tabelle interessate dai vincoli o di inserzioni di nuovi elementi in tali tabelle
→ Un vincolo può intersecare più tabelle (legate tra loro tramite politica di vincolo) e un inserimento/modifica su una tabella, di conseguenza comprometterebbe anche le altre → Politica **CASCADE**
- Si producono violazioni sui vincoli a seguito di modifiche, in una tabella interna, oppure a seguito di modifiche/cancellazioni da una tabella interna ad una esterna
→ Politica **CASCADE**
- La reazione alla violazione di un vincolo è il rifiuto del tentativo di fare l'operazione

SEMANTICA, CALCOLO RELAZIONALE E ALGEBRA RELAZIONALE

- Nella semantica di un linguaggio di programmazione consideriamo, quello di tipo **procedurale** e quello di tipo **dichiarativo**
- L'**algebra relazionale** è di tipo procedurale, perché ci dice come un'interrogazione viene fatta, cioè in che ordine vengono eseguite le operazioni, e quali operatori si adoperano
- Il **calcolo relazionale** invece, è di tipo dichiarativo, perché ci dice solamente quale valore è atteso come risultato, ma non ci dice come si ottiene
- L'effettiva semantica di SQL è data dal metodo dichiarativo, che implica il calcolo relazionale basato su n-uple (tuple)

VAGLINI-NOZIONI DI TEORIA

OTTIMIZZAZIONE ALGEBRICA

- La 1° fase dell'ottimizzazione algebrica si basa sul concetto di **equivalenza**
- L'equivalenza è rappresentata dall'euristica che dice, di confrontare un'espressione così com'è, con una in cui si anticipano:
 - Le selezioni (**pushing selections down**)
 - Le proiezioni (**pushing projections down**)
- Due espressioni algebriche sono **equivalenti**, se producono lo stesso risultato, cioè lo stesso schema (dato dagli stessi attributi) e le stesse tuple, qualunque sia l'istanza attuale della base di dati
- Per dire se due espressioni algebriche siano equivalenti o meno, bisognerebbe valutare i loro passaggi intermedi, che potrebbero far variare il costo delle operazioni

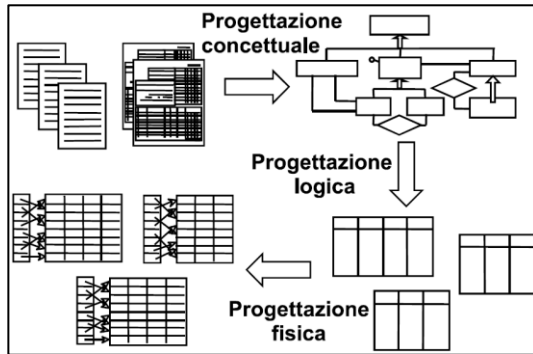
CALCOLO SU DOMINI, CALCOLO SU TUPLE, EQUIVALENZE CALCOLO-ALGEBRA

- Il calcolo relazionale comprende:
 - Calcolo **su domini**
 - Calcolo **su tuple**
- Questi due non possono essere usati l'uno al posto dell'altro per specificare qualsiasi interrogazione (vedi operatore di unione \cup)
- Il calcolo dei domini non richiede variabili diverse per i vari nomi diversi delle singole relazioni:
 - A prescindere le variabili sono tutte diverse per qualsiasi relazione
- Nel calcolo relazionale si possono esprimere espressioni 'senza senso', cioè:
 - **Dipendenti dal dominio degli attributi**, ma anche indipendenti
- Nell'algebra relazionale invece, tutte le espressioni hanno senso, cioè:
 - **Indipendenti dal dominio degli attributi**
- Le espressioni del calcolo relazionale dipendenti dal dominio di attributi, non sono rappresentabili nell'algebra relazionale, a causa del discorso sull'indipendenza in quest'ultima
- Invece, le espressioni del calcolo relazionale che siano indipendenti dal dominio, fanno sì, che esistano espressioni dell'algebra relazionale equivalenti ad esse; vale lo stesso discorso per il viceversa, cioè dall'algebra relazionale al calcolo relazionale
- Ci sono però, alcune interrogazioni che non sono esprimibili, né con il calcolo relazionale, né con l'algebra relazionale e sono:
 - Le funzioni ricorsive come ad esempio la **chiusura transitiva**

VAGLINI-NOZIONI DI TEORIA

SCHEMI DI BASI DI DATI

- Gli schemi di una base di dati sono:
 - Lo **schema concettuale** (E-R)
 - Lo **schema logico** (Modello logico-relazionale strutturato in tabelle)
 - Lo **schema fisico** (Dischi fisici)

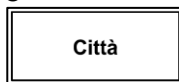


MODELLO CONCETTUALE E-R

- Esistono vari costrutti per rappresentare i concetti in uno schema E-R e sono i seguenti:

1. **Entità**:

- Rappresenta una classe di oggetti (fatti/persone/cose), i quali concetti hanno una proprietà significativa propria
- Una **occorrenza di entità** (oppure **istanza di entità**) rappresenta l'elemento o gli elementi della classe di oggetti

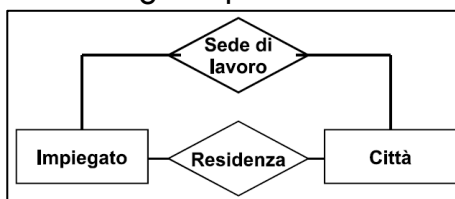


2. **Associazione** (oppure **relazione** / **relationship**):

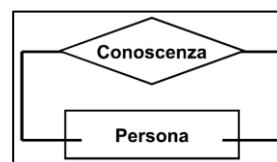
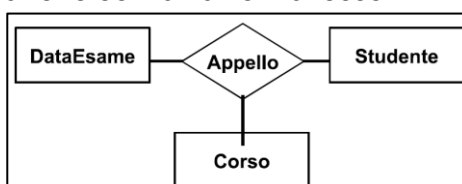
- Rappresenta il legame logico fra due o più entità



- Questo legame può non essere unico sulle due entità e può avere ruoli diversi



- Una **occorrenza di associazione** rappresenta una tupla di occorrenze tra le entità coinvolte nell'associazione
- Esiste l'**associazione ternaria** e l'**associazione ricorsiva** e volendo, esistono anche combinazioni di esse

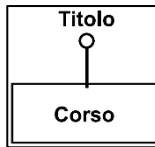


VAGLINI-NOZIONI DI TEORIA

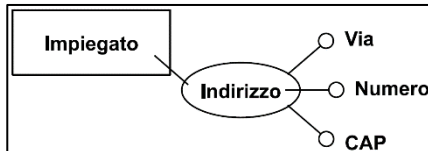
...CONTINUO MODELLO CONCETTUALE E-R

3. *Attributo:*

- È la proprietà del concetto che assume un'entità o un'associazione



- Può esistere un **attributo composto**, cioè a sua volta l'attributo è composto da altri attributi, che ne specificano l'utilizzo in maniera specifica



4. *Cardinalità di associazioni:*

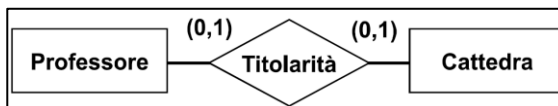
- La cardinalità minima usa i simboli 0 e 1
- La cardinalità massima usa i simboli 1 e N
- Associazione “molti a molti”



- Associazione “uno a molti”

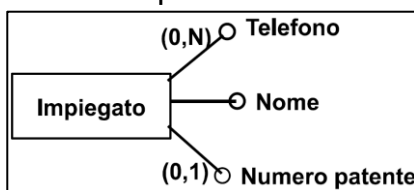


- Associazione “uno a uno”



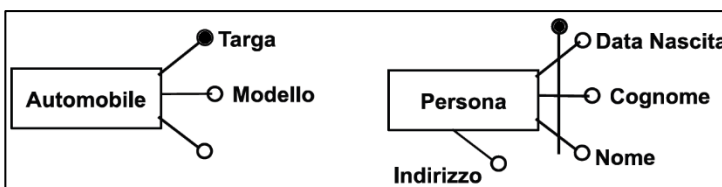
5. *Cardinalità di attributi:*

- Molto utile per indicare attributi opzionali, oppure multivalore



- Per gli attributi abbiamo gli **identificatori** che si dividono in:

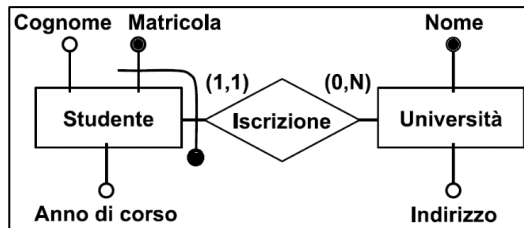
- **Identificatori interni**



VAGLINI-NOZIONI DI TEORIA

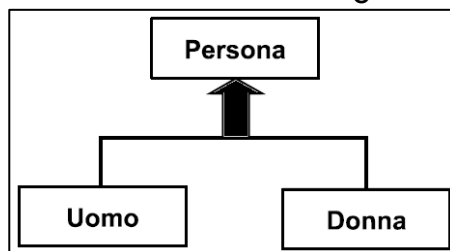
...CONTINUO MODELLO CONCETTUALE E-R

- **Identificatori esterni** con cardinalità obbligatoria (1,1)



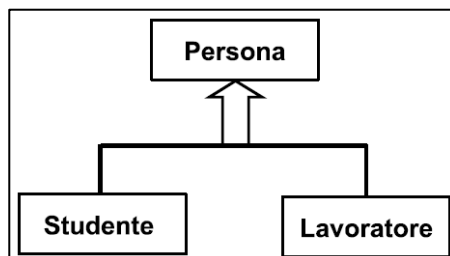
6. Generalizzazione:

- Utilizza il concetto di ereditarietà, cioè che le proprietà dell'entità genitore vengono ereditate all'entità figlie e ci sono vari tipi di generalizzazione
- **Generalizzazione totale**, se ogni occorrenza del genitore è occorrenza di almeno una delle entità figlie e, si mette la freccia piena



//Una persona può essere uomo o donna

- **Generalizzazione parziale**, se possono esistere delle occorrenze del genitore che non sono nessuna delle entità figlie e, si mette la freccia vuota



//Una persona può non essere né studente né lavoratore

- **Generalizzazione esclusiva**, se con le occorrenze dell'entità figlia copro totalmente le occorrenze dell'entità genitore
- **Generalizzazione esclusiva**, se la stessa occorrenza del genitore la trovo in entrambe le entità figlie

VINCOLI DI INEGRITÀ GENERICI (ASSERZIONI)

- Sono costituiti da vincoli, posti allo stesso livello della definizione delle tabelle, chiamate **asserzioni**
- Si rappresentano attraverso la dicitura:

```
CREATE ASSERTION NomeAsserzione  
CHECK (Condizione)
```

- La clausola **check** permette di restringere il dominio su cui si verifica la sua condizione (solitamente la condizione è quella che compare nella WHERE)



VAGLINI-NOZIONI DI TEORIA

...CONTINUO VINCOLI DI INEGRITÀ GENERICI (ASSERZIONI)

- Costrutto per verificare il tipo controllo associato ad un vincolo:

```
SET CONSTRAINTS NomeAsserzione (IMMEDIATE | DEFERRED)
```

- Costrutto per cancellare l'asserzione:

```
DROP NomeAsserzione
```

TRIGGER

- Il **trigger** è un vincolo dinamico che reagisce a ciò che capita, durante la computazione
- È caratterizzato dal paradigma **evento-condizione-azione** suddiviso così:
 - L'evento che lo fa partire (solitamente modifiche e cancellazioni)
 - La condizione che si deve verificare perché esso parta
 - L'azione da eseguire a seguito della verifica della condizione (solitamente degli aggiornamenti SQL)
- Il trigger viene eseguito prima (**BEFORE**) o dopo (**AFTER**) di un evento
- I trigger sullo stesso evento possono andare in conflitto per questo:
 - Si eseguono prima tutti i trigger di tipo BEFORE e poi quelli di tipo AFTER
- I trigger si possono trovare:
 - Tutti di tipo BEFORE
 - Tutti di tipo AFTER
- L'ordine di esecuzione dei trigger, BEFORE o AFTER che siano, è definito dal loro **timestamp** di creazione

PRIVILEGI

- Possono essere concessi o negati a:
 - Risorse riferite
 - Tipologie di utenti
 - Azioni da svolgere sul database
- Costrutto di concessione di privilegio:

```
GRANT < PRIVILEGES | ALL PRIVILEGES > ON  
RESOURCE  
TO USERS [WITH GRANT OPTION]
```

 - La clausola **grant** indica se il privilegio può essere trasmesso o meno ad altri utenti
- Costrutto di revoca di privilegio:

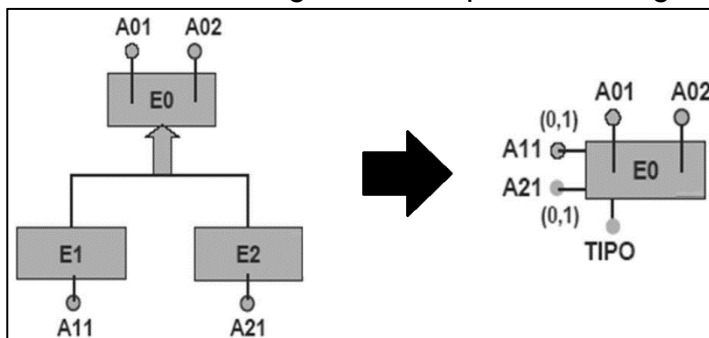
```
REVOKE PRIVILEGES ON  
RESOURCE  
FROM USERS [RESTRICT | CASCADE]
```

VAGLINI-NOZIONI DI TEORIA

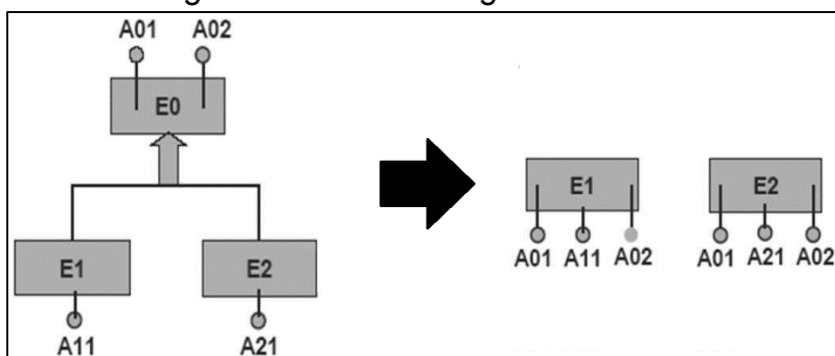
RISTRUTTURAZIONE

- Per la ristrutturazione ci sono alcuni passi da seguire e sono:
 - L'eliminazione delle generalizzazioni
 - L'eliminazione degli attributi multivalore
 - L'analisi e l'eventuale eliminazione delle ridondanze
 - Il partizionamento/accorpamento di entità e associazioni

- Il primo tipo di eliminazione delle generalizzazioni riguarda il fatto di accorpare le entità figlie della generalizzazione nell'entità genitore:
 - Nell'entità genitore si inseriscono gli attributi di ambedue i figli, i quali attributi non riguardano tutte le tuple, ma una parte (per ciascun figlio)
 - Ci sarà quindi un attributo ulteriore che rappresenterà a quale delle entità figlie ci si riferisce
 - Questo conviene se gli accessi al padre e alle figlie sono contemporanei



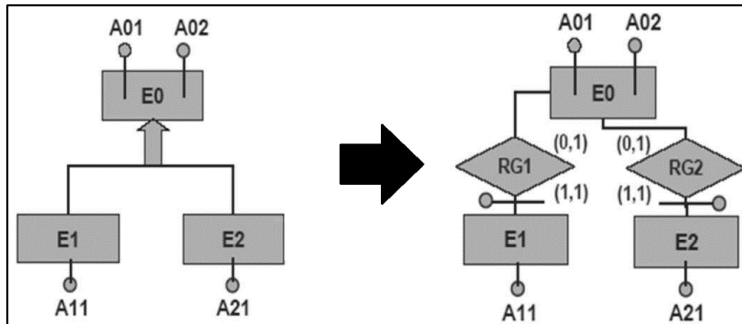
- Il secondo tipo di eliminazione delle generalizzazioni riguarda il fatto di accorpare il genitore della generalizzazione nelle entità figlie:
 - Levando l'entità padre, i suoi attributi li dovrò duplicare nelle sottoclassi, che riguardano le figlie
 - Conviene se gli accessi sono alle figlie e sono distinti dall'una all'altra



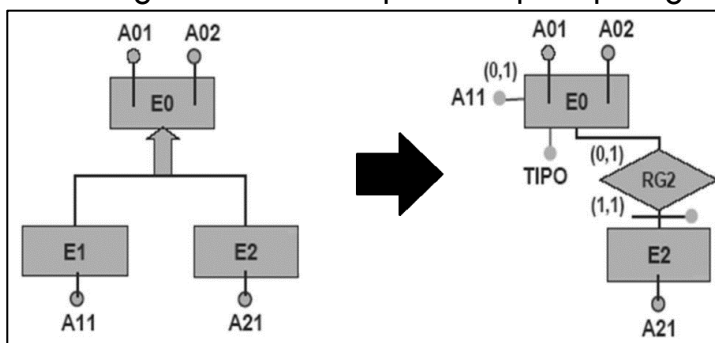
VAGLINI-NOZIONI DI TEORIA

...CONTINUO RISTRUTTURAZIONE

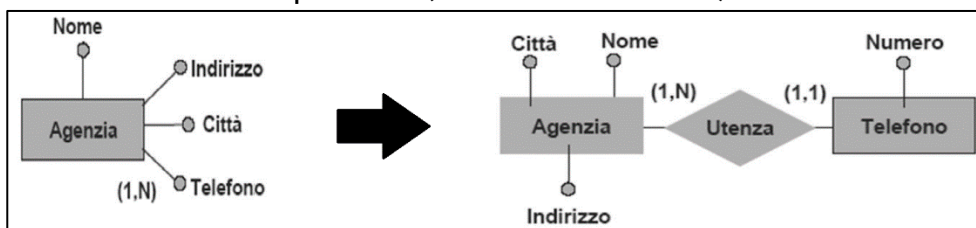
- Il terzo tipo di eliminazione delle generalizzazioni riguarda il fatto di sostituire la generalizzazione con relazioni:
 - Esse legano le sottoclassi con l'entità padre, e non introducono mai ridondanza
 - Conviene se si effettuano accessi separati alle entità figlie e al padre



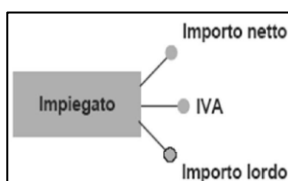
- Sono possibili anche delle soluzioni ibride che accorpano una delle entità figlie sul padre e lasciano separata l'altra entità figlia:
 - L'entità figlia che rimane separata è quella più significativa



- Per l'eliminazione degli attributi multivalore
 - Si trasforma questo attributo in un'entità, la quale si collega tramite una relazione all'entità di partenza (dove era multivalore)



- Per l'analisi/eventuale eliminazione di ridondanze
 - Si considerano da togliere gli attributi derivabili da altri attributi oppure le associazioni derivabili da altre associazioni



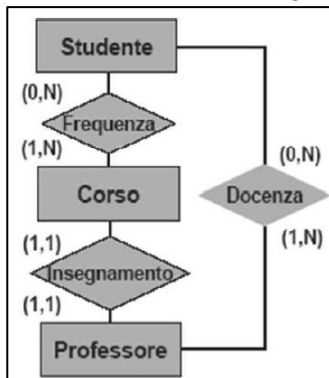
//Qualunque coppia tra questi tre mi porta a trovare il valore rimanente



VAGLINI-NOZIONI DI TEORIA

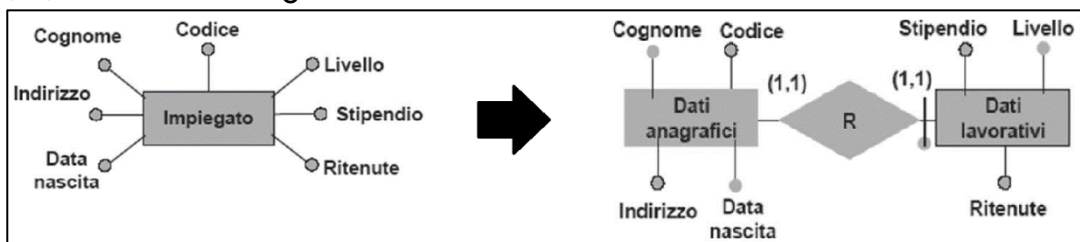
...CONTINUO RISTRUTTURAZIONE

- Si considerano da togliere le associazioni derivabili da altre associazioni



//Docenza è ridondante, perché ottenibile dai collegamenti:
Studente → Frequenza → Corso → Insegnamento → Professore

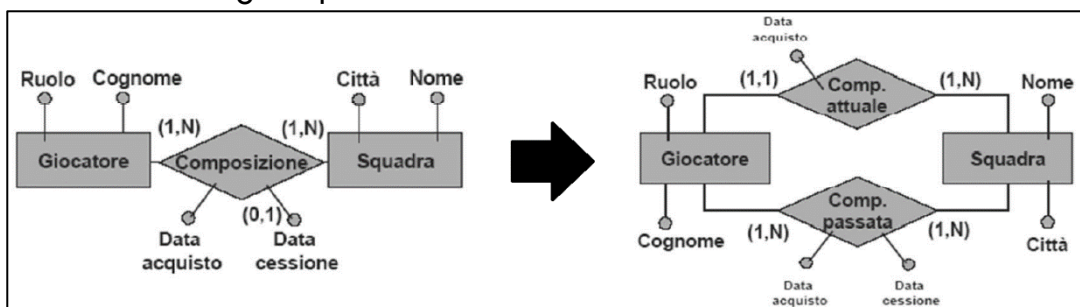
- Per il partizionamento/accorpamento di entità
 - Posso avere il partizionamento di un'entità in due entità tramite associazione (1,1) verso entrambi gli estremi



- Posso accorpare un'entità Y che ha la partecipazione obbligatoria (1,1) nell'altra entità X legata ad essa, e in X ci finiscono tutti gli attributi di Y



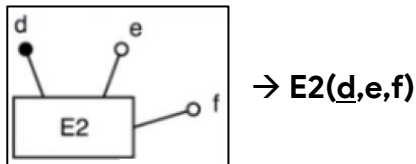
- Posso inoltre, scomporre un'associazione tra due entità in due associazioni diverse che collegano queste due entità



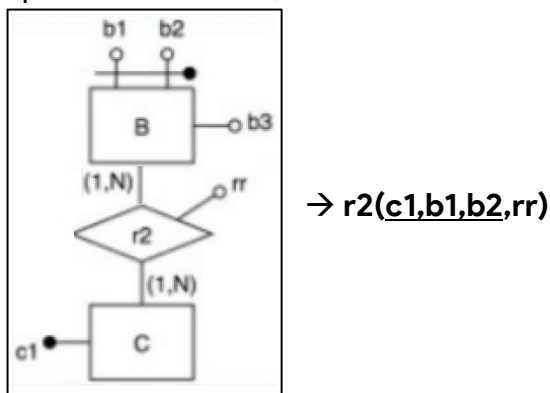
VAGLINI-NOZIONI DI TEORIA

TRADUZIONE DA E-R VERSO IL MODELLO LOGICO-RELAZIONALE

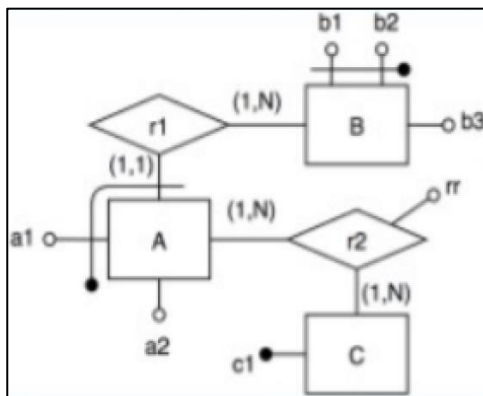
- Le entità diventano tabelle (relazioni) con i relativi attributi



- Le associazioni diventano tabelle (relazioni) con:
 - La chiave primaria composta dagli attributi chiave delle tabelle coinvolte nell'associazione
 - I relativi attributi (che non fanno parte della chiave primaria di quest'associazione)



- Un vincolo di chiave primaria rappresenta l'unione di tutti gli attributi che fanno parte della chiave primaria e viene contato come 1
- Un vincolo di chiave esterna (V.I.R.) rappresenta il collegamento, da un'entità verso l'altra, tramite la chiave esterna, la quale è rappresentata da uno più attributi, contando come 1, per ciascun vincolo esterno
- Questo/i attributo/i (V.I.R.) farà/faranno parte della tabella dove si rappresenta la traduzione di un'associazione tra entità, una delle quali utilizza questo/i attributo/i come chiave primaria
- La tabella che ha la chiave esterna deve avere verso l'associazione obbligatoriamente la cardinalità (1,1)



$A(\underline{a1}, b1, b2, a2) \rightarrow 1$ V.I.R. rappresentato dalla coppia b1, b2 riferente alla tabella B e di conseguenza include anche l'associazione r1, perché non avrà una tabella separata

$B(\underline{b1}, b2, b3) \rightarrow 0$ V.I.R.

$C(\underline{c1}) \rightarrow 0$ V.I.R.

$r2(\underline{a1}, b1, b2, c1, rr) \rightarrow 2$ V.I.R. rappresentati da a1, b1, b2 per la tabella A e c1 per la tabella C

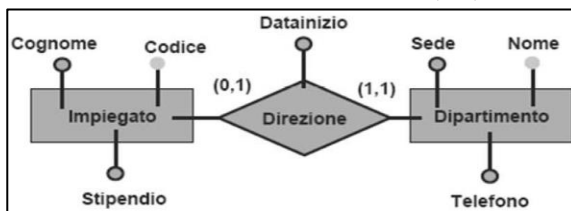
TOTALE: 3 V.I.R.



VAGLINI-NOZIONI DI TEORIA

...CONTINUO TRADUZIONE DA E-R VERSO IL MODELLO LOGICO-RELAZIONALE

- Le relazioni 1 a 1 fra due entità E1 e E2 ammettono più di un tipo di traduzione nel modello logico relazionale, perché a seconda dei seguenti casi si possono verificare situazioni diverse:
 - (1,1)-(0,1)
 - (1,1)-(1,1)
 - (0,1)-(0,1)
- Quando ho un'entità X, un'associazione e un'entità Y, con cardinalità (0,1) da X verso Y e cardinalità (1,1) da Y verso X:
 - La tabella referente all'associazione scompare, e i suoi attributi finiscono nella tabella Y
 - Rimangono solo le tabelle delle entità X e Y, mettendo dentro Y, il nome della tabella X (oppure un altro nome che lo può riguardare) come attributo
 - Si crea un V.I.R. tra questo attributo appena messo e la chiave primaria della tabella X
 - CONVIENE ACCORPARE VERSO IL LATO IN CUI LA CARDINALITÀ MINIMA TROVIAMO 1, CIOÈ DOVE HO (1,1)

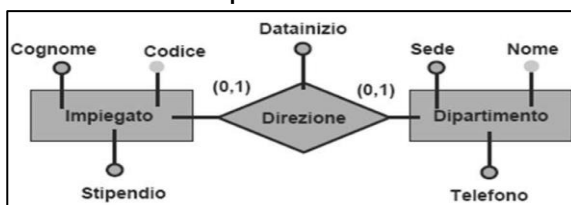


Impiegato(Codice, Cognome, Stipendio)

Dipartimento (Nome, Sede, Telefono, Direttore, DataInizio)

//Direttore si riferisce alla tabella Impiegato, dopo l'accorpamento

- Accorpare la 'partecipante opzionale', che ha cardinalità minima 0, dipende dal fatto se si creano troppi, o pochi valori nulli nella base di dati
- Quando ho un'entità X, un'associazione e un'entità Y, con cardinalità (0,1) da X verso Y e cardinalità (0,1) da Y verso X:
 - Conviene tenere la tabella referente all'associazione per non avere troppi valori nulli da nessuna parte



Impiegato(Codice, Cognome, Stipendio)

Direzione(Direttore, Dipartimento, DataInizio)

Dipartimento (Nome, Sede, Telefono)

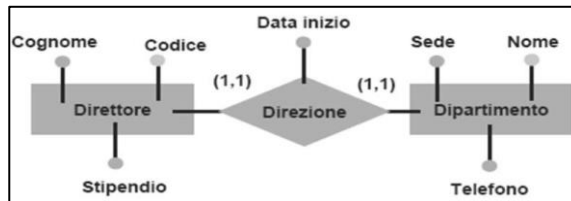
//Direttore si riferisce alla tabella Impiegato



VAGLINI-NOZIONI DI TEORIA

...CONTINUO TRADUZIONE DA E-R VERSO IL MODELLO LOGICO-RELAZIONALE

- Quando ho un'entità X, un'associazione e un'entità Y, con cardinalità (1,1) da X verso Y e cardinalità (1,1) da Y verso X:
 - La tabella referente all'associazione scompare
 - Rimangono solo le tabelle delle entità X e Y, mettendo dentro X, il nome della tabella Y come attributo (OPPURE VICEVERSA)
 - Si crea un V.I.R. tra questo attributo appena messo e la chiave primaria della tabella Y
 - SI PUÒ SCEGLIERE DI ACCORPARE O DA UNA PARTE O DALL'ALTRA



Direttore(Codice,Cognome, Stipendio,SedeDipartimento,DataInizio)

Dipartimento (Nome,Sede,Telefono)

//SedeDipartimento si riferisce alla tabella Dipartimento, dopo l'accorpamento sulla tabella X

- L'algoritmo di traduzione di un diagramma E-R verso il modello logico relazionale può decidere se creare o meno la tabella corrispondente all'associazione R(X,Y) in base alle cardinalità di R, verso le due entità X e Y
- Questo algoritmo di traduzione può:
 - Ottimizzare l'occupazione di memoria per la tabella corrispondente all'associazione R(X,Y) quando la cardinalità di R è (1,1) per X o per Y
 - Può non ottimizzarla quando la cardinalità di R è (0,1) per X o per Y
- Un'associazione ternaria del modello E-R con cardinalità (0,1) su una delle entità coinvolte:
 - Ogni istanza di quell'entità può partecipare ad al massimo una terna dell'associazione, dato che la cardinalità massima di (0,1) è 1

DIPENDENZE FUNZIONALI E CONCETTO DI NORMALIZZAZIONE

- Le **dipendenze funzionali** (FD) esprimono un legame semantico fra due gruppi di attributi di una relazione, la quale deve avere uno stato valido da cui partire per poi poter trovare le varie FD
- Le **forme normali** (FN) sono le caratteristiche di uno schema di relazione, per cui valgono le dipendenze funzionali
- Esistono vari tipo di FN, le quali, devono poter garantire l'assenza di alcuni difetti che sono presenti negli schemi di relazione
- La **normalizzazione** è la procedura che permette di portare uno schema relazionale in una determinata forma normale



VAGLINI-NOZIONI DI TEORIA

...CONTINUO DIPENDENZE FUNZIONALI E CONCETTO DI NORMALIZZAZIONE

- La dipendenza funzionale è rappresentabile tramite una relazione matematica $X \rightarrow Y$, cioè che esiste una dipendenza funzionale da X a Y, i valori di X determinano quelli di Y per una determinata relazione R(Z), su:
 - Z=Insieme di attributi
 - X, Y=Sottoinsiemi di Z
- Ad ogni chiave K della relazione R(Z) corrisponde una dipendenza funzionale $K \rightarrow Z$

ESEMPIO DI DIPENDENZA FUNZIONALE:

"Ogni impiegato ha il suo stipendio"

Impiegato \rightarrow Stipendio

(l'impiegato in questo caso determina lo stipendio)

- Si dice **dipendenza funzionale banale**, una FD sempre soddisfatta, cioè che a SX e a DX della freccia ho qualcosa sempre uguale \rightarrow Ciò mi rappresenta il fatto che quello che sta a DX della freccia è sempre compreso anche a sinistra

ESEMPIO DI DIPENDENZA FUNZIONALE BANALE:

Impiegato Progetto \rightarrow Progetto

(dato un impiegato e un progetto su cui lavora, essi determinano un progetto)

CONCETTI DI CHIUSURA, IMPLICAZIONE E SUPERCHIAVE

- Considerando:
 - F come insieme di FD definite su R(Z)
 - La FD: $X \rightarrow Y$ \rightarrow si dice che F **implica** $X \rightarrow Y$ e si scrive $F \Rightarrow X \rightarrow Y$ quando si verifica che:
 - Per ogni istanza della relazione R che verifica tutte le dipendenze in F, risulta verificata anche $X \rightarrow Y$
- La **chiusura transitiva** di F è l'insieme di tutte le FD implicate da F e si indica con F^+
 - \rightarrow Quindi è l'insieme di tutti i legami tra attributi (FD) appartenenti ad F, con ciascuno dei quali che, preso singolarmente, porta all'unione con tutti i rimanenti legami (FD) dell'insieme F, tranne sé stesso
 - \rightarrow Dato un qualunque insieme di dipendenze, posso costruire tutte le dipendenze implicate
- La **superchiave** è rappresentata da uno o più attributi e si può determinare considerando se:
 - Data F^+
 - Se vi è contenuta una dipendenza che, dato un valore K ottengo tutti gli attributi della relazione \rightarrow Allora K è superchiave

VAGLINI-NOZIONI DI TEORIA

CONCETTI DI EQUIVALENZA DI ESPRESSIONI, CHIUSURA SULL'INSIEME DI ATTRIBUTI E MINIMALITÀ

- Dati due insiemi di FD, F e $G \rightarrow$ Si dice che sono **equivalenti** se hanno la stessa chiusura, cioè $F^+ = G^+ \rightarrow$ Devo poter ottenere lo stesso insieme di dipendenze F partendo da G , oppure il viceversa, cioè ottenere G partendo da F
- Denotiamo con X^+ la **chiusura sull'insieme di attributi** di una relazione $R(Z)$
- Essa permette di verificare se una dipendenza, è implicata da un insieme
 \rightarrow Quella dipendenza sta nella chiusura di un insieme F dato, se e solo se
 \rightarrow La parte DX è ottenibile partendo dalla parte SX e usando le regole di F
- In generale, trovando tutti gli attributi, questo tipo di chiusura può servire per trovare la chiave (o superchiave)
- Un insieme F di FD si dice **minimale** quando non posso togliere attributi a SX , oppure quando non posso togliere dipendenze da F , dopo che è stato posto in forma standard, cioè con un solo attributo a DX
- La **copertura minimale** di un insieme F di FD è un insieme equivalente ad esso, ma con minor complessità

FORME NORMALI

- Una relazione è in **forma normale di Boyce-Codd (BCNF)** se per ogni dipendenza definita su di essa (non banale), la parte SX di ciascuna dipendenza è superchiave, e si verifica così:
 - Si prende ciascuna dipendenza
 - Si considera la sua parte SX e se ne fa la chiusura
 - Se per ciascuna parte SX considerata, si trovano tutte le dipendenze, allora questa forma sarà in BCNF
 - Altrimenti non ho una forma BCNF per la relazione
- La complessità di verificare il BCNF è **polinomiale** \rightarrow Ciò è dovuto al fatto che basta guardare se, per ogni dipendenza, alla sua SX è presente una superchiave
- Se una relazione non è in BCNF si rimpiazza con un'altra relazione che lo sia, usando la **decomposizione** sulle FD che consiste nel:
 - Separare gli attributi di una determinata tabella, sulla base delle FD
 - Esempio: “Data la tabella e le successive FD”

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Impiegato \rightarrow Sede
Progetto \rightarrow Sede



VAGLINI-NOZIONI DI TEORIA

...CONTINUO FORME NORMALI

- Decompongo sulla base delle dipendenze ottenendo

Impiegato	Sede	Progetto	Sede
Rossi	Roma	Marte	Roma
Verdi	Milano	Giove	Milano
Neri	Milano	Saturno	Milano
		Venere	Milano

- Ciò comporta che per entrambe le dipendenze trovate, a SX non ho una superchiave, dato che la superchiave è formata da Impiegato, Progetto
- Provo allora a riunire ottenendo

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano
Verdi	Saturno	Milano
Neri	Giove	Milano

- Che è diversa dalla relazione iniziale
- Incombo in quella che si chiama **decomposizione con perdita sul JOIN**
- Esiste un algoritmo per la decomposizione in BCNF che:
 - Mi garantisce la non perdita sul JOIN
 - Non mi garantisce la non perdita sulle dipendenze
- Per verificare una decomposizione senza perdita di dipendenze si deve:
 - Fare tutti i possibili partizionamenti
 - Calcolare le rispettive chiusure, vedendo se sono equivalenti alle chiusure di partenza

OSSERVAZIONE: Non è sempre possibile ottenere un BCNF che conservi le dipendenze e il JOIN

- Una relazione è in **terza forma normale (3NF)** se, per ogni dipendenza definita su di essa
 1. Può avere la parte SX di ciascuna dipendenza come superchiave (come BCNF)
 2. Oppure, può avere anche gli attributi della parte DX che sono contenuti in almeno una chiave della relazione→ Deve verificarsi o l'una o l'altra situazione
- La complessità di verificare la 3NF è *esponenziale* → Ciò è dovuto al fatto che non solo devo guardare per ogni dipendenza, alla sua SX è presente una superchiave
→ Ma posso vedere se, di ogni dipendenza che considero, la parte DX è contenuta in almeno una chiave della relazione → Quindi devo vedere tutti i possibili sottoinsiemi delle chiavi



VAGLINI-NOZIONI DI TEORIA

...CONTINUO FORME NORMALI

- Esempio: "Data la tabella e le successive FD"

<u>Dirigente</u>	<u>Progetto</u>	<u>Sede</u>
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Progetto Sede → Dirigente
Dirigente → Sede

- Considerando Progetto Sede → Dirigente, posso dire che a SX ho la superchiave (punto 1 di pag. precedente)
 - Considerando invece Dirigente → Sede, posso dire che Sede è contenuto nella chiave (punto 2 di pag. precedente)
- Esempio: "Data la seguente relazione e le successive FD"

R(<u>A</u> , <u>B</u> ,C,D,E,F,G)
AB→CDEF
C→F
F→G

Si generano gli schemi:

R1(A,B,C,D,E,F) R2(C,F) R3(F,G)

- R1 non è 3NF perché all'interno esiste una FD tra C ed F dove non c'è né a SX una superchiave, né a DX qualcosa contenuto nella chiave
- Inoltre A,B che va in F, nella prima FD è ridondante, perché ci può arrivare passando tramite C, e di conseguenza andrebbe minimizzato così: AB→CDE
- Da una forma non in 3NF, si può sempre ottenere una decomposizione 3NF, senza perdite sul JOIN e inoltre, che conserva le dipendenze e ci sono due tipologie di decomposizione:
 - Si garantisce l'assenza di perdita sul JOIN e poi si conservano le dipendenze
 - Si conservano le dipendenze e poi si risolve l'eventuale perdita sul JOIN→ Quindi è sempre raggiungibile rispetto alla BCNF
- Quando viene effettuata la decomposizione in 3NF, se verifico che a SX ho un'unica chiave per tutte le parti DX delle FD, allora essa è anche superchiave
→ Di conseguenza viene verificata automaticamente che la forma 3NF è uguale alla forma BCNF
- La 3NF rispetto al BCNF è meno restrittiva e ammette relazioni con alcune anomalie e ridondanze
- Una base di dati ottenuta da un processo di progettazione non è necessario che tutte le relazioni siano in forma normale, qualsiasi essa sia → La normalizzazione si può usare durante la progettazione concettuale, sulla verifica di ridondanze, partizionamenti di entità/relazioni, ma non è obbligatoria
- Ogni insieme di dipendenze in forma normale può essere minimizzato e quindi diventare minimale



VAGLINI-NOZIONI DI TEORIA

...CONTINUO FORME NORMALI

- Ogni insieme minimale di dipendenze può essere in forma normale perché:
 - Insieme minimale → Dispone di una regola di ottimizzazione sintattica: non ci sono né lettere né dipendenze che si possono togliere
 - La normalità → Mi dice invece, che a SX di una dipendenza ci deve essere una superchiave oppure a DX ogni elemento è contenuto in una chiave
→ Se prendo un insieme in forma normale e lo minimizzo (insieme minimale) rimane in forma normale
- Ogni volta che vengono definite delle relazioni si controlla se queste siano in forma normale per:
 - Verificare la qualità dello schema logico-relazionale
 - Verificare la qualità dello schema concettuale, sia sulle ridondanze, sia sul partizionamento di entità/relazioni → Tramite la normalizzazione, facciamo in modo di ridurre le inconsistenze che si potrebbero creare, come ad esempio la perdita di vincoli di integrità originari, dopo un eventuale partizionamento, e così via...

TRANSAZIONI

- Il concetto di **multitasking** rappresenta l'esecuzione contemporanea di più processi in un calcolatore, e i processi possono essere eseguiti in modalità:
 - **Interleaved** → Alternati fra loro, concorrentemente
 - Parallela → Solo se sono presenti più CPU
- La **transazione** è l'unità elementare di lavoro utilizzata dalle applicazioni → Cioè un programma, rappresentato da una serie di comandi e operazioni
- Essa svolge una serie di operazioni utili in una base di dati come:
 - Inserimenti
 - Cancellazioni
 - Modifiche
 - Interrogazioni
- Il sistema che utilizza il meccanismo delle transazioni e che ne controlla la concorrenza → È chiamato **sistema transazionale**
- Il programma che caratterizza la transazione è dotato di:
 - Un comando iniziale → **begin-transaction**
 - Un comando finale → **end-transaction**
- Questi due comandi rappresentano un **cammino di esecuzione**, dove vi deve essere eseguito una sola volta uno dei due seguenti comandi:
 - **Commit work** → Per la terminazione corretta della transazione
 - **Rollback work** → Per abortire la transazione
→ Entrambi i comandi rappresentano la fine della transazione
→ Quindi la transazione può contenere all'interno del proprio cammino di esecuzione o uno o l'altro comando
- Le transazioni godono di una proprietà fondamentale che ne garantisce il modo di esecuzione → Cioè la proprietà **ACID**



VAGLINI-NOZIONI DI TEORIA

...CONTINUO TRANSAZIONI

- **A → ATOMICITÀ**
 - La base di dati non può rimanere in uno stato intermedio → Quindi gli stati intermedi di una transazione non devono essere visibili
 - Un guasto o un errore prima del commit → Fa sì che si ritorni all'inizio
 - Un guasto o un errore dopo il commit → Fa sì che vengano ripetute le operazioni fatte fino al commit stesso
 - **C → CONSISTENZA**
 - Rispetta i vincoli di integrità
 - Stato iniziale transazione corretto porta a → Stato finale transazione corretto → Ciò deve verificarsi indipendentemente da ciò che accade nel durante
 - **I → ISOLAMENTO**
 - Transazioni non risentono gli effetti di altre transazioni eseguite concorrentemente → L'esecuzione concorrente tra transazioni deve produrre risultati ottenibili, come se ci fosse un'esecuzione sequenziale (con una priorità tra una transazione ed un'altra)
 - Le singole operazioni all'interno di una transazione non devono essere visibili all'esterno
 - **D → DURABILITÀ (PERSISTENZA)**
 - Effetti di transazioni andate in commit devono essere salvati → In modo da poterli recuperare, a seguito di un guasto al database per poter riaverne lo stato consistente
 - I gestori delle transazioni sono
 - **Gestore dell'affidabilità** → Atomicità / Durabilità
 - **Gestore della concorrenza** → Isolamento
 - **Gestore dell'integrità a tempo di esecuzione** → Consistenza
 - La transazione si deve trovare sempre in uno dei seguenti stati:
 - **Active** → Dopo lo stato iniziale siamo in uno stato in cui si possono eseguire operazioni di lettura o scrittura (**R/W**)
 - **Partially committed** → Stato prima della fine della transazione, dove avviene un controllo da parte del Gestore dell'affidabilità → Il quale verifica se verrà salvato lo stato della transazione in modo permanente (per la durabilità) → Se ciò avviene, allora si passerà allo stato successivo
 - **Committed** → Stato in cui la transazione è finita
 - **Failed** → Stato in cui l'esecuzione si blocca a causa di un errore SW → Gestore dell'affidabilità fallisce
 - **Aborted** → Stato dopo che la transazione ha subito un rollback → Il database viene ripristinato allo stato precedente → Dell'inizio della transazione
- OSSERVAZIONE: Il gestore dell'affidabilità principalmente garantisce che gli effetti di una transazione andata in commit non vadano perduti (persistenza)

VAGLINI-NOZIONI DI TEORIA

GESTORE AFFIDABILITÀ E TIPI DI MEMORIE

- Il gestore si occupa di alcuni malfunzionamenti che possono esserci come:
 - Malfunzionamento del disco → Vengono perse le informazioni risiedute sopra (in memoria secondaria)
 - Malfunzionamento di alimentazione → Vengono perse le informazioni della memoria centrale e dei registri → Come, ad esempio, risultati di transazioni che hanno fatto commit → Ciò invece, non accade nella memoria secondaria dove c'è il database
 - Errore software → Otteniamo risultati scorretti e quindi delle inconsistenze
- I comandi transazionali eseguiti dal gestore di affidabilità sono:
 - **start transaction**
 - **commit work**
 - **rollback work**
 - **warm restart**
 - **cold restart**

→ Le operazioni vanno fatte con uno stato consistente del database, oppure devono portare il database in uno stato consistente

→ Le ultime due servono per il ripristino in caso di guasti → Ciò avviene grazie ad un archivio permanente in cui si salvano le operazioni svolte → **LOG**
- Considerando il ripristino le memorie possono essere classificate:
 - Memoria volatile → Perdita di informazioni in caso di spegnimento di sistema
 - Memoria non volatile → Mantenimento di informazioni in caso di spegnimento di sistema, ma perdita per altri malfunzionamenti
 - Memoria stabile → Mantenimento sempre delle informazioni per ogni guasto
- La persistenza delle memorie cambia a seconda di quale si considera:
 - Memoria centrale → Non persistente
 - Memoria di massa → Persistente ma danneggiabile
 - Memoria stabile → Memoria 'indistruttibile'

OSSERVAZIONE:

- L'ultima tipologia di memoria è un'astrazione, perché non esiste in natura, ma si può progettare una memoria solida che la porti ad essere stabile e sopravvivere a qualunque guasto
- La memoria stabile serve per memorizzare le strutture dati che servono per garantire la persistenza del database

LOG, CHECKPOINT E DUMP

- Il LOG:
 - Si trova in memoria stabile (secondaria)
 - È replicato in tante copie → Quindi grazie a queste repliche garantisce la sua sopravvivenza
 - Riporta tutte le operazioni in ordine di una transazione
 - Serve a ricostruire le operazioni che hanno prodotto un risultato, che non sappiamo più se è affidabile così com'è memorizzato



VAGLINI-NOZIONI DI TEORIA

...CONTINUO LOG, CHECKPOINT E DUMP

- Il **checkpoint** e **dump**, gestiti dal controllore dell'affidabilità, servono per la ricostruzione delle operazioni, a seguito di un guasto → A partire dal punto più vicino del guasto → Senza dover ripartire dall'inizio
- Il checkpoint registra le transazioni che sono attive in un certo istante
→ Confermando quali transazioni non sono iniziate oppure quali sono finite
- Prima dell'inserimento del record di checkpoint nel LOG, il gestore dell'affidabilità:
 - Può rifiutare nuovi begin-transaction → Aspettando che tutte le transazioni iniziate eseguano commit/abort
 - Può rifiutare nuovi commit
- Nel momento del checkpoint, invece, si procede così:
 1. Sospende l'accettazione di commit/abort dalle transazioni
 2. Forza la scrittura in memoria secondaria, sulle pagine del buffer
→ Le quali sono state modificate dalle transazioni che hanno fatto commit
 3. Forza la scrittura nel LOG di un record che contiene il nome delle transazioni attive
 4. Si riprende ad accettare tutte le operazioni da parte delle transazioni
→ Garantisco la persistenza delle transazioni che hanno fatto commit
→ Di conseguenza i dati sono memorizzati per qualunque guasto
- Il dump:
 - È una copia completa della base di dati
 - È salvato in memoria stabile come *backup*
 - Se si trova dentro un LOG rappresenta il momento in cui il LOG viene effettuato
- Ci sono due regole per la modifica del LOG da parte delle transazioni:
 1. Regola **Write-Ahead-Log (WAL)** → I record di LOG devono essere scritti prima della scrittura dei corrispondenti record sul database
 2. Regola **Commit-Precedenza** → I record di LOG devono essere scritti prima dell'esecuzione di commit
- La scrittura del LOG nella base di dati avviene in 3 modalità:
 1. Immediata → Prima del commit, ad ogni modifica metto il record nel LOG e poi scrivo nel database
 2. Differita → Dopo il commit, inserisco il record nel LOG e scrivo nel database
 3. Mista → A volte scrivo dopo una modifica e a volte scrivo dopo il commit

OSSERVAZIONE:

- Scrittura sul LOG prima della scrittura sulla base di dati, cioè sulla memoria secondaria → Perché garantisce la consistenza in caso di guasto
- Scrittura sulla base di dati avviene in qualsiasi momento → Può avere quindi maggior efficienza in entrambi i casi, dato che il gestore porta i dati in memoria secondaria al momento, a lui, più opportuno
- Il commit è considerato effettuato quando il corrispondente record di LOG viene scritto

VAGLINI-NOZIONI DI TEORIA

GUASTI, RIPRESA A CALDO E RIPRESA A FREDDO

- Una transazione può fallire per qualche guasto (tramite un abort) e, tutti gli stati intermedi fra il begin e l'abort, vanno disfatti → Ci sono più tipi di guasti:
 - Guasto **soft** → Errore di sistema / di caduta di tensione → Si ricorre alla **RIPRESA A CALDO** (warm restart)
 - Guasto **hard** → Perdita della memoria secondaria/guasto HW (ma non del LOG, che può essere un evento 'catastrofico') → Si ricorre alla **RIPRESA A FREDDO** (cold restart):
 - Prima recupera gli oggetti danneggiati
 - Fa una ripresa a caldo
- Considerando la RIPRESA A CALDO e la RIPRESA A FREDDO, le quali basano la fase di recovery sugli insiemi UNDO e REDO → A seconda di come siano gli insiemi di UNDO e REDO → Ci possono essere, o meno, operazioni da rifare e disfare
- Quando una transazione abortisce (abort), si possono verificare due casi per le operazioni derivanti dal rollback della transazione
 1. Si inseriscono nell'insieme UNDO e poi si fanno al momento del recovery
 2. Si eseguono al momento dell'abort e si inseriscono nell'insieme REDO, al momento del recovery da un guasto

ANOMALIE DURANTE LA CONCORRENZA

- Le anomalie tramite possono avvenire sull'esecuzione di due transazioni nei casi in cui una delle due, o entrambe, in un caso vogliano scrivere; se invece, entrambe vogliono leggere non si verificano le anomalie.
- Siano la lettura **R** e la scrittura **W** i tipi di anomalie che troviamo sono:
 - Perdita di aggiornamento → Per W-W
 - Lettura sporca → Per R-W oppure per W-W
 - Lettura inconsistente → Per R-W
 - Aggiornamento fantasma → Per R-W
 - Inserimento fantasma → Per R-W

SCHEDULE, SCHEDULER E CONTROLLO DI CONCORRENZA

- **Schedule** → È la funzione fondamentale della concorrenza → Sequenza di R/W relative all'insieme delle transazioni concorrenti in un certo istante, le quali operazioni sono ordinate dentro lo schedule in ordine temporale di esecuzione della base di dati
- Esempio di schedule:

$S_1 : r_1(x) \ r_2(z) \ w_1(x) \ w_2(z)$

 - $r_1(x)$ → Lettura dell'oggetto **x** da parte della transazione **t1**
 - $r_2(z)$ → Lettura dell'oggetto **z** da parte della transazione **t2**
 - $w_1(x)$ → Scrittura dell'oggetto **x** da parte della transazione **t1**
 - $w_2(z)$ → Scrittura dell'oggetto **z** da parte della transazione **t2**
- **Scheduler** → Esegue il controllo della concorrenza, tenendo traccia di tutte le operazioni eseguite sulla base di dati, da parte delle transazioni e decide se accettare o rifiutare le operazioni che vengono richieste

VAGLINI-NOZIONI DI TEORIA

SCHEDULE SERIALE E SERIALIZZABILE

- **Schedule seriale** → Si ha se, per ogni transazione t , tutte le azioni di t compaiono in sequenza senza interruzioni da parte di azioni che compaiono da altre transazioni
- Esempio di schedule seriali:

○ $S2: r0(x)r0(y)w0(x)r1(y)r1(x)w1(y)r2(x)r2(y)r2(z)w2(z)$

→ è uno schedule seriale poiché delle seguenti transazioni tutte le loro azioni sono eseguite in sequenza $t0, t1, t2$

○ $S6: w0(x)r2(x)w2(x)w2(z)r1(x)$

→ è uno schedule seriale poiché delle seguenti transazioni tutte le loro azioni sono eseguite in sequenza $t0, t2, t1$

OSSERVAZIONE: L'ordine delle transazioni non è importante che sia in sequenza come nel primo esempio, ma le loro azioni non devono essere interrotte

- **Schedule serializzabile** → Si ha quando uno schedule S_i produce lo stesso risultato di un qualunque schedule seriale S_j , il quale è definito dalle stesse transazioni di S_i
- In uno schedule la sequenza delle operazioni che vengono completate rappresentano degli schedule serializzabili, equivalenti a qualche schedule seriale

VIEW-EQUIVALENZA, VIEW-SERIALIZZABILITÀ E VSR

- La relazione **legge-da** → Vuole dire che viene fatta un'operazione di lettura su un oggetto, solamente se è preceduta da un'operazione di scrittura sullo stesso oggetto
- La **scrittura finale** → È l'ultima scrittura fatta su un oggetto
- Due schedule si dicono **view-equivalenti** → Se entrambi hanno la stessa relazione "legge-da" e le stesse "scritture finali" sull'oggetto
- Uno schedule è **view-serializzabile** → Se è view-equivalente ad un qualche schedule seriale → Quindi se ha la stessa relazione "legge-da" di uno schedule seriale → Quindi, due schedule view-equivalenti possono essere view-serializzabili
- **VSR** → È l'insieme degli schedule view-serializzabili
- La verifica per la:
 - View-equivalenza → POLINOMIALE
 - View-serializzabilità → NP-COMPLETO

CONFLICT-SERIALIZZABILITÀ

- Due operazioni si dicono in **conflitto** → Se operano sullo stesso oggetto, e almeno una di esse è in scrittura, su quell'oggetto e ci sono:
 - Conflitto **read-write** (rw o wr)
 - Conflitto **write-write** (ww)
- Due schedule si dicono **conflict-equivalenti** → Se hanno le stesse operazioni (da parte delle transazioni) ed ogni coppia di operazioni in conflitto ($rw/wr/ww$) compare in entrambi nello stesso ordine
- Uno schedule è **conflict-serializzabile** → Se è conflict-equivalente ad un qualche schedule seriale
- **CSR** → È l'insieme degli schedule conflict-serializzabili



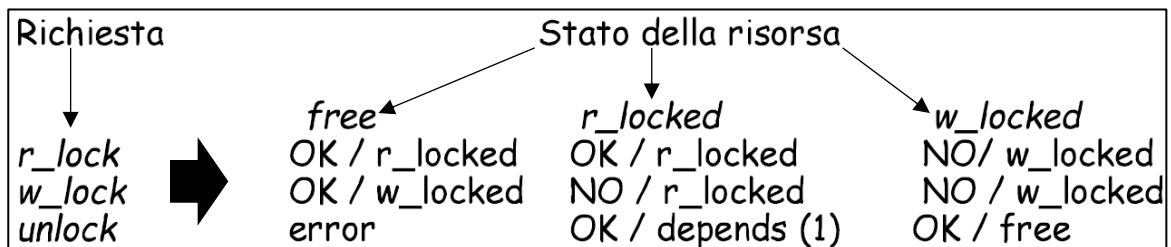
VAGLINI-NOZIONI DI TEORIA

...CONTINUO CONFLICT-SERIALIZZABILITÀ

- CSR implica VSR → Ogni schedule conflict-serializzabile è anche view-serializzabile, però
- VSR non implica CSR → Perché ci sono degli schedule VSR che non sono CSR
- La verifica della conflict-serializzabilità avviene tramite il **grafo dei conflitti** → Cioè trasformato ciò che è successo nello schedule in un grafo
- Dalla precedente, ne scaturisce un TEOREMA:
Uno schedule è in CSR se e solo se il grafo è aciclico
→ Questo comporta nel dire che due schedule sono conflict-equivalenti se hanno lo stesso grafo dei conflitti
- Verifica per la conflict-serializzabilità → LINEARE

LOCK

- Il **lock** è usato dai sistemi operativi per l'accesso alle risorse condivise
- Tutte le letture e scritture sono precedute da:
 - **lock** → Rappresenta l'esclusività dell'operazione su un oggetto
- Inoltre, sono seguite da:
 - **unlock** → Rappresenta rilascio dell'esclusività dell'operazione su un oggetto
- In poche parole, con il lock si prende possesso di una variabile
- Il gestore del lock → **Lock manager** → Riceve le richieste di lock dalle transazioni e le accoglie o rifiuta
- I lock sono di tipo diverso:
 - **Condiviso** → In lettura → "Finché leggo una variabile di un oggetto la può leggere chiunque"
 - **Esclusivo** → In scrittura → "Se scrivo, invece lo devo fare solo io"
- **r_lock** → Precede tutte le letture, cioè le **read**
- **w_lock** → Precede le scritture (con l'esclusività), cioè le **write**
- **unlock** → Segue letture e scritture (con l'esclusività sulla scrittura)
- L'accoglimento o richiesta di lock da parte delle transazioni è gestita attraverso
→ La **tavola dei conflitti** → Realizza la politica per la gestione dei conflitti



- **r_locked** → Qualcuno detiene in lock in lettura
- **w_locked** → Qualcuno detiene in lock in scrittura
- I NO rappresentano i conflitti tra letture-scritture / scritture-letture
- Ci sono 3 tipi di richieste di lock, e ognuna di esse rappresenta una configurazione

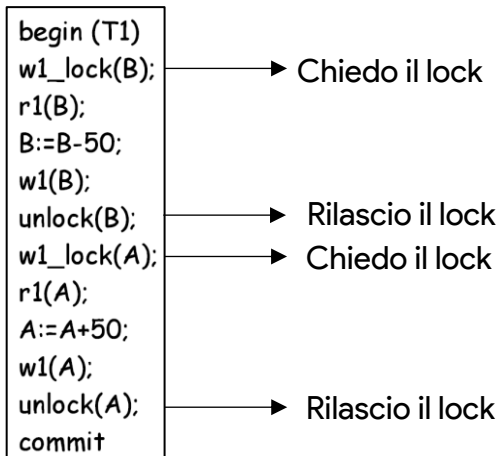


VAGLINI-NOZIONI DI TEORIA

...CONTINUO LOCK

- Per la richiesta `r_lock` controllo se:
 - a. Lo stato della risorsa è `free` (libera) → OK, la lettura si può fare
 - b. Lo stato della risorsa è `r_locked` → OK, la lettura si può fare lo stesso
 - c. Lo stato della risorsa è `w_locked` → NO, l'operazione in lettura viene rifiutata e lo stato di risorsa rimane quello che è
- Per la richiesta `w_lock` controllo se:
 - a. Lo stato della risorsa è `free` (libera) → OK, la lettura si può fare
 - b. Lo stato della risorsa è `r_locked` → NO, l'operazione in scrittura viene rifiutata e lo stato di risorsa rimane quello che è
 - c. Lo stato della risorsa è `w_locked` → NO, l'operazione in lettura viene rifiutata e lo stato di risorsa rimane quello che è
- Per la richiesta di `unlock` controllo se:
 - a. Lo stato della risorsa è `free` (libera) → Errore perché non ha senso sbloccare una richiesta libera
 - b. Lo stato della risorsa è `r_locked` → Si può fare `unlock` guardando il contatore dei lettori (1), che tiene il conto di quante risorse devono leggere:
 - i. Se questo è 0 allora la risorsa viene rilasciata
 - ii. Se questo non è 0 allora, il numero di lettori diminuisce di 1, ma rimane sempre lockato in lettura finché non si giunge al punto precedente
 - c. Lo stato della risorsa è `w_locked` → OK, si libera il lock dalla scrittura
- Se le risorse non vengono concesse, le transazioni richiedenti vengono messe in coda (lista di attesa), e si tirano fuori poi, una transazione alla volta dalla coda di attesa, non appena la risorsa viene rilasciata

-
- Esempio: "Lock e unlock"



VAGLINI-NOZIONI DI TEORIA

2PL

- Il **two-phase locking (2PL)** garantisce la conflict-serializzabilità, quindi

→ 2PL implica CSR

→ CSR non implica 2PL

- Per la condizione che CSR implica VSR allora ho che:

→ 2PL implica VSR

→ VSR non implica 2PL

OSSERVAZIONE: Se avessi due schedule risultanti dall'esecuzione delle richieste delle transazioni sulla base del protocollo 2PL → Essi non sarebbero sempre view-equivalenti / conflict- equivalenti tra loro, ma lo sarebbero soltanto se riguardassero le stesse transazioni

- Fasi del 2PL:

1. Impone di avere sempre prima il lock e dopo l'unlock

2. Vincolo sul rilascio dei lock → Una transazione, acquisisce tutti i lock necessari, poi uno ad uno li rilascia, e finchè non li ha rilasciati tutti non può acquisire nient'altro

→ Quindi, ho queste "due fasi", una di acquisizione e una di rilascio

Esempio:

<pre>begin (T1) w1_lock(B); r1(B); B:=B-50; w1(B); w1_lock(A); r1(A); unlock(B); A:=A+50; w1(A); unlock(A); commit</pre>	<p>————→ Chiedo il lock</p> <p>————→ Chiedo il lock</p> <p>————→ Rilascio il lock</p> <p>————→ Rilascio il lock</p>
--	---

- Il 2PL presenta alcune anomalie:

1. **Letture sporche** → Il fallimento (abort) di una transazione che ha scritto una risorsa causa il fallimento di tutte le transazioni che hanno letto quel valore → Aborto in cascata tra le transazioni (**cascading rollbacks**) → La prima transazione abortisce, la seconda abortisce, e così via...

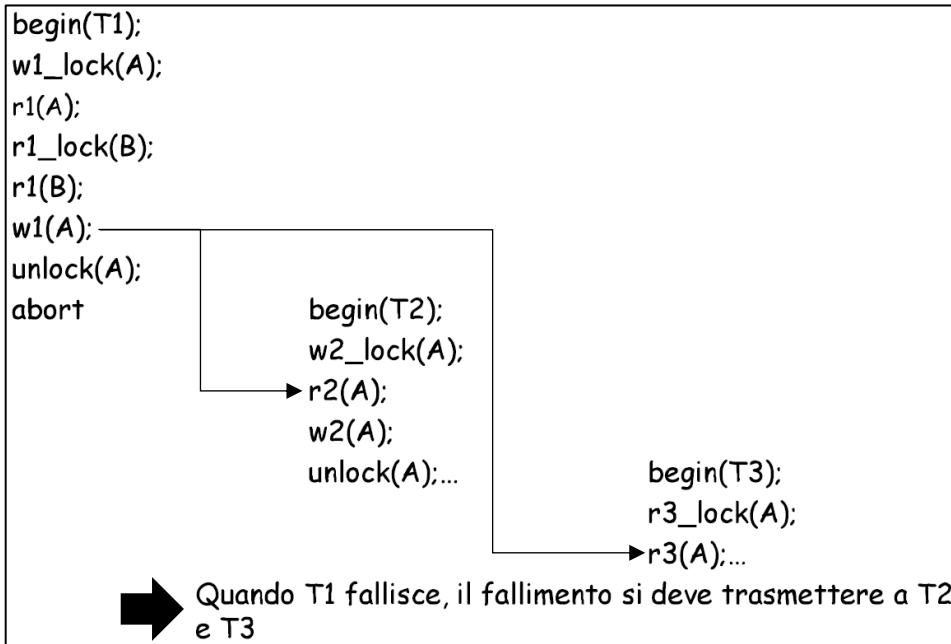
2. **Deadlock** (attese incrociate) → Avviene quando ho una transazione che attende che si liberi una risorsa da parte di un'altra transazione, la quale a sua volta, attende che si liberi la risorsa da parte della prima transazione → Essendo che entrambe si trovano in una coda di attesa → Ciò ci porta ad avere questo blocco di risorse tramite queste 'attese incrociate', e a non ripartire mai



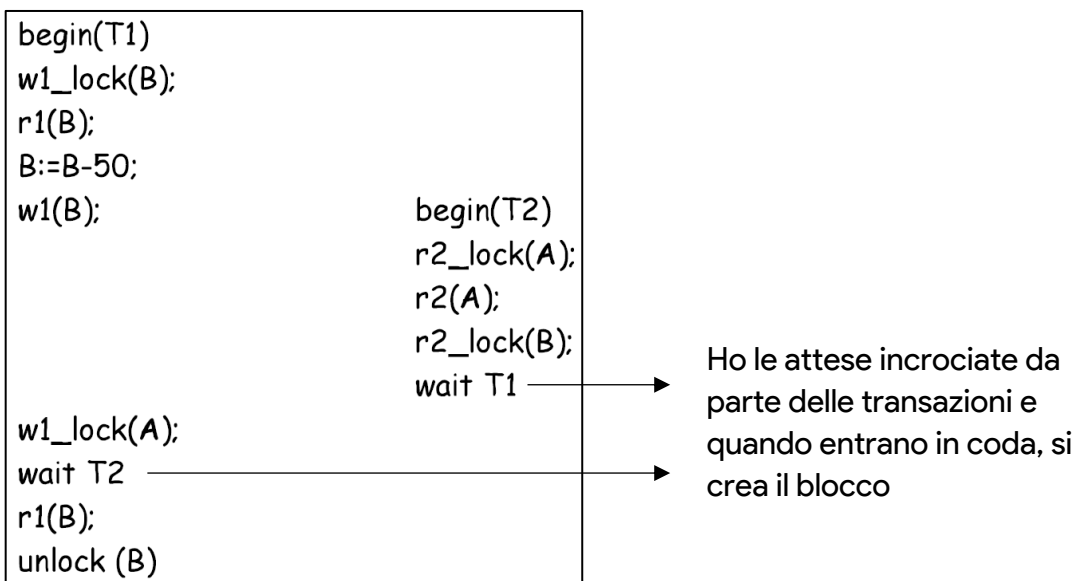
VAGLINI-NOZIONI DI TEORIA

...CONTINUO 2PL

Esempio di letture sporche:



Esempio di deadlock:



S2PL

- Lo **strict two-phase locking (S2PL)** elimina l'anomalia di "letture sporche"
→ Di conseguenza risolve il problema del "cascading rollback" (aborto in cascata)
- Per questo protocollo, tutti i lock effettuati da una transazione possono essere rilasciati solamente dopo aver fatto il commit (è un po' più restrittivo del 2PL)
- Poiché S2PL è un sottoinsieme di 2PL abbiamo che:
 - S2PL implica 2PL
 - 2PL non implica S2PL



VAGLINI-NOZIONI DI TEORIA

...CONTINUO S2PL

- Poiché 2PL implica CSR ho che:
→ S2PL implica CSR
- Per la condizione che CSR implica VSR, allora, ho anche che:
→ S2PL implica VSR
- Per i due casi precedenti non vale il viceversa:
→ CSR non implica S2PL
→ VSR non implica S2PL

OSSERVAZIONE: Se avessi uno schedule risultante dall'esecuzione delle richieste delle transazioni sulla base del protocollo 2PL ed uno risultante dall'esecuzione delle richieste delle transazioni sulla base del protocollo S2PL → Essi non sarebbero sempre view-equivalenti / conflict-equivalenti tra loro, ma lo sarebbero soltanto se riguardassero le stesse transazioni

Esempio:

```
begin (T1)
w1_lock(B);
r1(B);
B:=B-50;
w1(B);
w1_lock(A);
r1(A);
A:=A+50;
w1(A);
commit
unlock(B);
unlock(A);
```

→ Soltanto dopo il commit
vengono rilasciati i lock

TS

- Il **timestamp (TS)** → È un'alternativa al 2PL e rappresenta un identificatore che definisce l'ordinamento su ogni transazione
- TS rappresenta l'istante d'inizio di una transazione → Cioè l'attivazione della transazione

Esempio: "Nel momento che nel sistema parte il BEGIN di una transazione T, se nel sistema erano già presenti 3 transazioni → Allora T come timestamp prende 4 → Cioè rappresenta il fatto che T sia partita come 4°"

- Per questa tecnica si accetta come schedule 'corretto' uno schedule non seriale, ma che sia equivalente ad uno seriale → In cui le transazioni sono costruite secondo l'ordine introdotto dal timestamp, riferito all'inizio delle transazioni
Esempio: "Ho un ordinamento di transazioni che è T1 T2 T3 e so che è equivalente a T2 T1 T3 → Per un 2PL potrebbe essere accettato ma per il TS non conta l'equivalenza → T2 T1 T3 deve essere rifiutato e deve essere accettato un ordinamento del tipo T1 T2 T3 → Che è quello compatibile con l'ordinamento delle transazioni"
- L'ordine seriale delle transazioni viene fissato prima del BEGIN, senza accettare altri ordinamenti



VAGLINI-NOZIONI DI TEORIA

...CONTINUO TS

- Dal discorso precedente, si afferma che se uno schedule è eseguito con il TS
→ Esiste un unico schedule seriale corrispondente, che rispetta l'ordine del BEGIN
- Per il discorso precedente abbiamo che:
→ TS implica CSR
- Per la condizione che CSR implica VSR, allora, ho anche che:
→ TS implica VSR
- Per i due casi precedenti non vale il viceversa:
→ CSR non implica TS
→ VSR non implica TS
- L'alternatività tra 2PL e TS porta ad avere che:
→ TS e 2PL sono incomparabili
- Lo schedule usato per il timestamp ha due contatori per ogni oggetto:
 - **RTM(x)** → Contatore di lettura
 - **WTM(x)** → Contatore di scrittura
- Inoltre, riceve richieste di letture /scritture con indicato il TS della transazione
 - **read(x, ts)** → Richiesta di lettura
 - **write(x, ts)** → Richiesta di scrittura

Esempio: "Schedule di un TS"

read(x,1)	RTM(x)=1
write(x,1)	WTM(x)=1
read(z,2)	RTM(z)=2
read(y,1)	RTM(y)=1
write(y,1)	WTM(y)=1
read(x,2)	RTM(x)=2
write(z,2)	WTM(z)=2

→ Più alto è l'identificativo del contatore più recente è la transazione che arriva

TS CONFRONTATO CON 2PL, TIMEOUT E STARVATION

- In TS ho che:
 - Le transazioni quando non possono prendere il lock, vengono "uccise" e poi rilanciate, come nuove transazioni → Non può causare deadlock
- In 2PL invece ho che:
 - Le transazioni quando non possono prendere il lock, vengono messe in attesa → Causa deadlock
- Il fatto di uccidere le transazioni e rilanciarle → Ha un costo maggiore del metterle in attesa, proprio perché le ripartenze sono più costose
- Le transazioni che rimangono in deadlock possono avere il problema dello **stallo**
→ Che si può risolvere con il **timeout**:
 - Una specie di timer impostato sulle risorse chieste da parte delle transazioni, prima di decidere, per quest'ultime, se abortirle o meno
 - L'abort si fa quando passa più tempo del timeout e la risorsa ancora non viene concessa

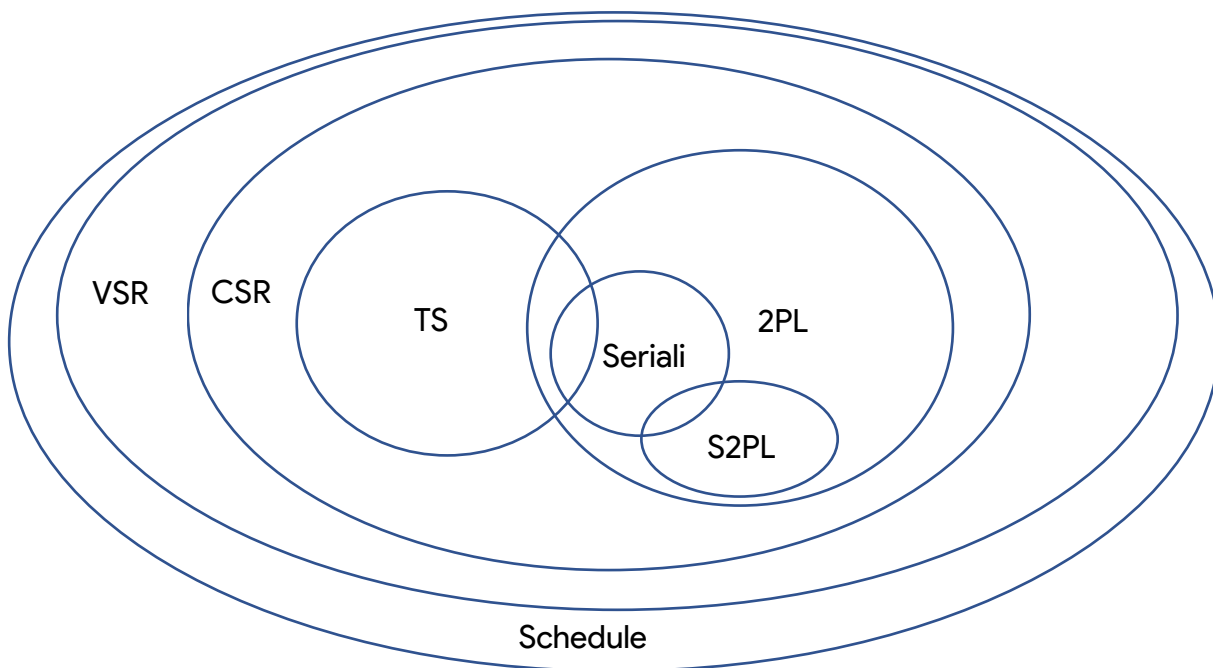


VAGLINI-NOZIONI DI TEORIA

...CONTINUO TS CONFRONTATO CON 2PL, TIMEOUT E STARVATION

- Per scegliere le transazioni da uccidere si può pensare a:
 1. Uccidere le transazioni appena fanno richiesta di una qualche risorsa
→ Per non interrompere le transazioni che stanno lavorando
 2. Uccidere le transazioni che hanno le risorse e che hanno svolto meno lavoro
→ Cioè quelle che si sono bloccate subito
- Il secondo punto ci può portare ad avere una uccisione sulla stessa transazione, per più volte → Cioè, se la transazione viene uccisa, e poi si trova di nuovo nella stessa situazione quando riparte → Essa viene riuccisa, e questa cosa può riaccadere di nuovo → Se si verifica ripetutamente si può andare incontro al fenomeno detto **starvation**
- Se considero due transazioni che vengono eseguite concorrentemente → Queste non hanno mai il TS uguale

GRAFICO RIEPILOGATIVO PER LE TECNICHE DI GESTIONE DELLA CONCORRENZA



VAGLINI-PREPARAZIONE ESERCIZI

UNIONE

\cup → Tuple di una relazione unite alle tuple di un'altra relazione

INTERSEZIONE

\cap → Tuple in comune tra due relazioni

DIFFERENZA

$-$ → Tuple della prima relazione che non sono nella seconda relazione

OSSERVAZIONE:

- Nell'unione, se ho valori ripetuti (venenti fuori tra l'unione di due relazioni) ne inserisco solo una di tuple! *//No duplicati*
- Rimane lo stesso numero di attributi per tutti e tre gli operatori
- Le tuple possono essere sia uguali che diverse per numero
- Gli schemi sono diversi ma gli attributi dello stesso dominio

RIDENOMINAZIONE

$\rho_{A1, A2, \dots, An \leftarrow B1, B2, \dots, Bn}(R)$

- $A1, A2, \dots, An \rightarrow$ Nuovi attributi
- $B1, B2, \dots, Bn \rightarrow$ Vecchi attributi
- $R \rightarrow$ Argomento

OSSERVAZIONE:

- La ridenominazione si fa prima degli attributi che vanno proiettati oppure quando ci serve ridenominare gli attributi per una condizione di JOIN, solitamente il SELF JOIN

ESEMPIO

“Ridenominare l'attributo padre in genitore”

Paternità	
Padre	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

- Ridenominando

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

→ Ottengo il seguente schema:

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

VAGLINI-PREPARAZIONE ESERCIZI

SELEZIONE

$\sigma_{F1, \dots, Fn}(R)$

- $F1, \dots, Fn \rightarrow$ Condizioni su attributi

//Orizzontale su tuple \rightarrow

OSSERVAZIONE:

- La condizione nella selezione è vera per valori non nulli
- La selezione mi può buttare via delle righe

ESEMPIO (Una condizione)

"Impiegati che guadagnano più di 50000 euro"

- Faccio la selezione

$\sigma_{\text{Stipendio} > 50000}(\text{Impiegati})$

\rightarrow Potrei ottenere possibilmente il seguente schema:

Impiegati			
Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Roma	55000
5998	Neri	Milano	64000
5698	Neri	Napoli	64000

ESEMPIO (Più condizioni)

"Impiegati che guadagnano più di 50000 euro e che lavorano a Milano"

- Faccio la selezione

$\sigma_{(\text{Stipendio} > 50000) \text{ AND } (\text{Filiale} = \text{'Milano'})}(\text{Impiegati})$

\rightarrow Potrei ottenere possibilmente il seguente schema:

Impiegati			
Matricola	Cognome	Filiale	Stipendio
5998	Neri	Milano	64000

CONDIZIONI CON NULL

IS NULL / IS NOT NULL \rightarrow Riferimento con la selezione a valori nulli

ESEMPIO

"Impiegati che hanno più di 45 anni, oppure che non hanno l'età"

- Faccio la selezione

$\sigma_{(\text{Età} > 40) \vee (\text{Età IS NULL})}(\text{Impiegati})$

\rightarrow Potrei ottenere possibilmente il seguente schema:

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

// \vee è l'OR logico (è vera se una delle due è vera!)

VAGLINI-PREPARAZIONE ESERCIZI

PROIEZIONE

$\pi_{A_1, \dots, A_n}(R)$

- $A_1, \dots, A_n \rightarrow$ Attributi da proiettare

//Verticale su attributi ↑

ESEMPIO

“Matricola e cognome di tutti gli impiegati”

- Faccio la proiezione

$\pi_{\text{Matricola, Cognome}}(\text{Impiegati})$

→ Potrei ottenere possibilmente il seguente schema:

Matricola	Cognome
7309	Neri
5998	Neri
9553	Rossi
5698	Rossi

PRODOTTO CARTESIANO

X → Unione degli schemi di tutte le tuple

ESEMPIO

Impiegato	Reparto		Reparto	Capo		R.Impiegato	R.Reparto	Q.Cap	Q.Reparto
Rossi	A	X	B	Mori	➔	Rossi	A	Mori	B
Neri	B		C	Bruni		Rossi	A	Bruni	C
Bianchi	B					Neri	B	Mori	B
						Neri	B	Bruni	C
						Bianchi	B	Mori	B
						Bianchi	B	Bruni	C

JOIN NATURALE

R1 \bowtie **R2** → Unione degli schemi di tutte le tuple di due relazioni tramite l'attributo

in comune, inserendolo una volta sola

//No sovrapposizione

OSSERVAZIONE :

- Se ci sono relazioni senza attributi comuni il JOIN NATURALE diventa PRODOTTO CARTESIANO!

ESEMPIO

Impiegati \bowtie **Reparti**

Impiegato	Reparto		Reparto	Capo		Impiegato	Reparto	Capo
Rossi	A	⋈	B	Mori	➔	Neri	B	Mori
Neri	B		C	Bruni		Bianchi	B	Mori
Bianchi	B							

VAGLINI-PREPARAZIONE ESERCIZI

THETA JOIN, EQUI-JOIN

R1 \bowtie_F **R2** → Join con condizione, su un determinato attributo

- $F \rightarrow$ Condizione con operatori di confronto ($>$, $<$, $=$, ecc...) anche su attributi di relazioni diverse

OSSERVAZIONE:

- Se l'operatore di confronto, tra due condizioni (per esempio A,B) è sempre ' $=$ ', cioè:

R1 $\bowtie_{A=B}$ **R2**

→ si chiama **EQUI-JOIN**

- A differenza del NATURAL JOIN, gli attributi su cui si fa join vengono ripetuti, quindi vengono ripetute le colonne, mentre prima no → Si prendeva solo la colonna in comune

ESEMPIO

Impiegati $\bowtie_{\text{Reparto}=\text{Codice}}$ **Reparti**

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

UTILIZZO DEL MULTI JOIN

Se mi servono attributi da più relazioni nella proiezione finale, faccio più THETA-JOIN di seguito con le condizioni sulle chiavi primarie così:

R1 \bowtie_{F1} (**R2** \bowtie_{F2} **R3**)

OSSERVAZIONE:

- Se mi riferisco in **maniere** diverse ad una stessa tabella (in questo caso $R3 = R1$) posso fare il "join consecutivo" con ovviamente ridenominazioni precedenti riferendomi alla stessa tabella con una di tramite che ne descrive la **maniera**

R1 \bowtie_{F1} (**R2** \bowtie_{F2} **R1**)

ESEMPIO DI QUERY IN ALGEBRA RELAZIONALE

Impiegati	Matricola	Nome	Età	Stipendio	Supervisione	Impiegato	Capo
	7309	Rossi	34	45000		7309	5698
	5998	Bianchi	37	38000		5998	5698
	9553	Neri	42	35000		9553	4076
	5698	Bruni	43	42000		5698	4076
	4076	Mori	45	50000		4076	8123
	8123	Lupi	46	60000			

// La tabella Impiegati assume 2 significati diversi: come impiegato e come capo



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO ESEMPIO DI QUERY IN ALGEBRA RELAZIONALE

“Trovare gli impiegati che guadagnano più del loro capo mostrando matricola nome e stipendio del capo e dell'impiegato”

```

π Matr, Nome, Stip, MatrC, NomeC, StipC
(σ Stipendio > StipC (
ρ MatrC, NomeC, StipC, EtàC ← Matr, Nome, Stip, Età (Impiegati)
▷◁ MatrC=Capo
(Supervisione ▷◁ Impiegato=Matricola Impiegati)))
    
```

SPIEGAZIONE CODICE:

1° passo:

```

(Supervisione ▷◁ Impiegato=Matricola Impiegati)
    
```

→ Faccio il theta-join tra queste due tabelle sulla condizione Impiegato=Matricola, ottenendo la seguente tabella (in modo da avere gli impiegati con il capo associato)

Matricola	Nome	Età	Stipendio	Impiegato	Capo
7309	Rossi	34	45000	7309	5698
5998	Bianchi	37	38000	5998	5698
9553	Neri	42	35000	9553	4076
5698	Bruni	43	42000	5698	4076
4076	Mori	45	50000	4076	8123

2° passo:

```

ρ MatrC, NomeC, StipC, EtàC ← Matr, Nome, Stip, Età (Impiegati)
    
```

→ Dalla nella tabella Impiegato effettuo una ridenominazione degli attributi (in modo da avere una tabella che si riferisce al capo, visto che impiegati e capo sono nella stessa tabella ma, tramite matricola si distinguerà poi il capo, usando la tabella Supervisione). Ottengo

MatrC	NomeC	EtàC	StipC
7309	Rossi	34	45000
5998	Bianchi	37	38000
9553	Neri	42	35000
5698	Bruni	43	42000
4076	Mori	45	50000
8123	Lupi	46	60000

3° passo:

```

▷◁ MatrC=Capo
    
```

→ Combino le tabelle trovate ai passi 1° e 2° sulla condizione MatrC=Capo

MatrC	NomeC	EtàC	StipC	Matricola	Nome	Età	Stipendio	Impiegato	Capo
5698	Bruni	43	42000	7309	Rossi	34	45000	7309	5698
5698	Bruni	43	42000	5998	Bianchi	37	38000	5998	5698
4076	Mori	45	50000	9553	Neri	42	35000	9553	4076
4076	Mori	45	50000	5698	Bruni	43	42000	5698	4076
8123	Lupi	46	60000	4076	Mori	45	50000	4076	8123



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO ESEMPIO DI QUERY IN ALGEBRA RELAZIONALE

4° passo:

$\sigma_{\text{Stipendio} > \text{StipC}}$

→ Seleziono le tuple che hanno lo stipendio (dell'impiegato) maggiore dello stipendio del capo, sulla tabella del 3° passo

MatrC	NomeC	EtàC	StipC	Matricola	Nome	Età	Stipendio	Impiegato	Capo
5698	Bruni	43	42000	7309	Rossi	34	45000	7309	5698

5° passo:

$\pi_{\text{Matr, Nome, Stip, MatrC, NomeC, StipC}}$

→ Proietto i dati richiesti dal problema sulla tabella del 4° passo, che ne danno la soluzione

Matricola	Nome	Stipendio	MatrC	NomeC	StipC
7309	Rossi	45000	5698	Bruni	42000

TRASFORMAZIONI IN ESPRESSIONI EQUIVALENTI

Se F è una condizione con un attributo riguardante R2:

$\sigma_F(R1 \bowtie R2)$ diventa $\rightarrow R1 \bowtie \sigma_F(R2)$ // JOIN sulla selezione

Se A è un attributo riguardante R2:

$\pi_A(R1 \bowtie R2)$ diventa $\rightarrow R1 \bowtie \pi_A(R2)$ // JOIN sulla proiezione

OSSERVAZIONE:

- In una sequenza di selezioni si anticipano quelle più selettive (che 'buttano via' più roba)
- Anticipare il più possibile le proiezioni, anche introducendone di nuove

CALCOLO SUI DOMINI

$\{A_1:X_1, A_2:X_2, \dots, A_n:X_n \mid R(A_1:X_1, A_2:X_2, \dots, A_n:X_n) \wedge \theta\}$

(Cosa cerco)

(Dove lo cerco)

(Unito ad una formula)

- $A_i \rightarrow$ Attributi (Noti) richiesti dal testo
- $X_i \rightarrow$ Nomi riferenti agli attributi (Da assegnare)
- $| \rightarrow$ Separatore
- $R \rightarrow$ Nome relazione (Nota) da cui prendere gli attributi, con gli attributi e i loro nomi associati $A_i:X_i$
- $\wedge \rightarrow$ AND (logica) di unione per le formule oppure di unione tra tabelle sugli attributi comuni, cioè le chiavi primarie (quindi i vari JOIN)!
- $\theta \rightarrow$ Se è formula, riguarda gli X_i e non si mettono le parentesi!
- $\{, : () \} \rightarrow$ Sono di sintassi

VAGLINI-PREPARAZIONE ESERCIZI

CALCOLO SU TUPLE

$\{N_1.(X^1_1, \dots, X^1_n), \dots, N_n.(X^n_1, \dots, X^n_n) \mid N_1(R_1), \dots, N_n(R_n) \mid N_i.(X^i_1, \dots, X^i_n) \text{ su } \theta\}$

(Cosa cerco, notazione a punto) (Dove lo cerco) (Formula, notazione a Punto)

- $N_i \rightarrow$ Nomi riferenti alle relazioni (R_i) dopo il primo separatore, degli attributi che si vogliono in uscita
- $X^i_1, \dots, X^i_n \rightarrow$ Attributi riferenti a quella i -esima relazione che si cercano in uscita
- $| \rightarrow$ Separatore
- $R_i \rightarrow$ Relazione
- $\theta \rightarrow$ Condizione da soddisfare in base ai determinati attributi espressi nel testo dell'esercizio (SENZA PARENTESI), anche qua ho gli AND logici che mi uniscono le condizioni espressi nella formula

OSSERVAZIONE:

- Se nel calcolo di tuple volessi tutti gli attributi di una relazione nella parte "cosa cerco" ci metto
 $N_i.*i$ // Ovviamente considerando una determinata i -esima relazione
- Data un'espressione in algebra, se in questa c'è l'unione tra due tabelle diverse \rightarrow Non può essere espressa nel calcolo su tuple!
- Se nel calcolo relazionale ho un'unione $U \rightarrow$ Nel calcolo su domini ho \vee
- L'unione non si può esprimere nel calcolo su tuple

ESEMPI DI CALCOLO SU DOMINI E SU TUPLE

"Trovare matricola e nome degli impiegati che guadagnano più di 40 milioni"

Sui DOMINI:

$\{ \text{Matricola: } m, \text{ Nome: } n \mid \text{Impiegati (Matricola: } m, \text{ Nome: } n, \text{ Et\`a: } e, \text{ Stipendio: } s) \wedge s > 40 \}$

Sulle TUPLE:

$\{ i.(\text{Matricola}, \text{Nome}) \mid i(\text{Impiegati}) \mid i.\text{Stipendio} > 40 \}$

QUANTIFICATORI ESISTENZIALI E UNIVERSALI

$\exists \quad \forall \quad \neg \rightarrow$ Sono interscambiabili tra loro // $\exists x(f)$ posso scriverlo $\rightarrow \neg(\forall x(\neg(f)))$

- Nel calcolo dei domini posso dire $\exists / \forall / \neg x . \rightarrow$ Il punto rappresenta la dicitura "tale che"

VAGLINI-PREPARAZIONE ESERCIZI

ESERCIZIO ALGEBRA RELAZIONALE

- Base di dati da considerare per l'esercizio

- Film(CodiceFilm, Titolo, CodiceRegista, Anno)
- Produzione(CasaProduzione, Nazionalità, CodiceFilm, Costo, Incasso1annoSala)
- Artista(CodiceAttore, Cognome, Nome, Sesso, DataDiNascita, Nazionalità)
- Interpretazione(CodiceFilm, CodiceAttore, Personaggio, SessoPersonaggio)
- Regista(CodiceRegista, Cognome, Nome, Sesso, DataDiNascita, Nazionalità)
- Noleggio(CodiceFilm, Incasso1annoVideo, Incasso1annoDVD)

Esercizio: "Titoli dei film i cui attori sono tutti dello stesso sesso"

$$\begin{aligned} & (\pi_{\text{Titolo}}(\text{Film}) - \pi_{\text{Titolo}}(\text{Film} \bowtie (\pi_{\text{CF}}(\sigma_{\text{Sesso}='M'}(\text{Artista}) \bowtie \text{Interpretazione})))) \cup \\ & (\pi_{\text{Titolo}}(\text{Film}) - \pi_{\text{Titolo}}(\text{Film} \bowtie (\pi_{\text{CF}}(\sigma_{\text{Sesso}='F'}(\text{Artista}) \bowtie \text{Interpretazione})))) \end{aligned}$$

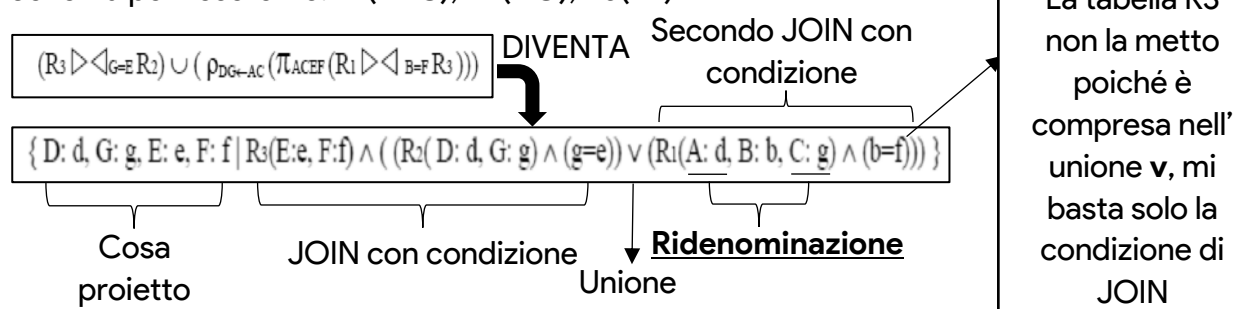
1

2

- A tutti i titoli ci tolgo quelli i cui attori sono solo maschi → Ci rimangono quindi femmine
 - U → UNITO A
- A tutti i titoli ci tolgo quelli i cui attori sono solo femmine → Ci rimangono quindi i maschi
→ Ho quindi i titoli dei film i cui attori sono tutte femmine unito a quello i cui attori sono tutti maschi

ESERCIZIO ALGEBRA RELAZIONALE-CALCOLO SU DOMINI

Schema per l'esercizio: R1(ABC), R2(DG), R3(EF)



- L'unione con OR viene fatta sulle tabelle che sono in comune a quella in AND esterna
- La tabella che è in comune va in AND a quelle altre due
- Le condizioni aggiuntive 'g=e' e 'b=f' si fanno dentro le parentesi riguardanti R2 e R1, poiché si riferiscono a R3 esterno

VAGLINI-PREPARAZIONE ESERCIZI

ESERCIZIO ALGEBRA RELAZIONALE-CALCOLO SU TUPLE

Schema per l'esercizio: R1(ABC), R3(EF)

$(\rho_{DG \leftarrow AC}(\pi_{ACEF}(R_1 \triangleright \triangleleft_{B=F} R_3)))$ **DIVENTA**

$\{i.(D,G), s.(E,F) \mid i(R_1), g(R_1), s(R_3) \mid$
 $i.B=g.B=s.F \wedge i.G=g.C \wedge i.D=g.A\}$

// g(R1) tabella aggiuntiva con gli attributi utili necessari da proiettare, questa tabella non sta nel risultato ma mi serve per ridenominare gli attributi di i(R1)

// Osserva il JOIN che va fatto anche sugli attributi della tabella aggiuntiva

ESERCIZIO CALCOLO SUI DOMINI

- Base di dati da considerare per l'esercizio

- Film(CodiceFilm, Titolo, CodiceRegista, Anno)
- Produzione (CasaProduzione, Nazionalità, CodiceFilm, Costo, Incasso1annoSala)
- Artista (CodiceAttore, Cognome, Nome, Sesso, DataDiNascita, Nazionalità)
- Interpretazione (CodiceFilm, CodiceAttore, Personaggio, SessoPersonaggio)
- Regista (CodiceRegista, Cognome, Nome, Sesso, DataDiNascita, Nazionalità)
- Noleggio (CodiceFilm, Incasso1annoVideo, Incasso1annoDVD)

“Titoli dei film i cui attori sono tutti dello stesso sesso”

$\{\text{Titolo: } t \mid \text{Film}(\text{CodiceFilm: } cf, \text{Titolo: } t, \dots) \wedge \neg \exists$
 $ca1, c1, n1, s1, nz1, dn1, ca2, c2, n2, s2, nz2, dn2, p1, p2, sp1, sp2$
 $(\text{Artista}(\text{CodiceAttore: } ca1, \text{Cognome: } c1, \text{Nome: } n1,$
 $\text{Sesso: } s1, \text{DataDiNascita: } dn1,$
 $\text{Nazionalità: } nz1) \wedge (\text{Artista}(\text{CodiceAttore: } ca2,$
 $\text{Cognome: } c2, \text{Nome: } n2, \text{Sesso: } s2, \text{DataDiNascita: } dn2,$
 $\text{Nazionalità: } nz2) \wedge \text{Interpretazione}(\text{CodiceFilm: } cf,$
 $\text{CodiceAttore: } ca1, \text{Personaggio: } p1,$
 $\text{SessoPersonaggio: } sp1) \wedge \text{Interpretazione}(\text{CodiceFilm: } c$
 $f, \text{CodiceAttore: } ca2, \text{Personaggio: } p2,$
 $\text{SessoPersonaggio: } sp2) \wedge s1 \neq s2)\}$

Non esistono due attori che abbiano sesso diverso (*)

Lista attributi dei due attori da valutare

(*) Condizione di sesso diverso

// Guardo i film uno per uno (TUTTI), andando a vedere se hanno attori dello stesso sesso e se ne trovo due con sesso diverso → Quel film non lo metto nel risultato

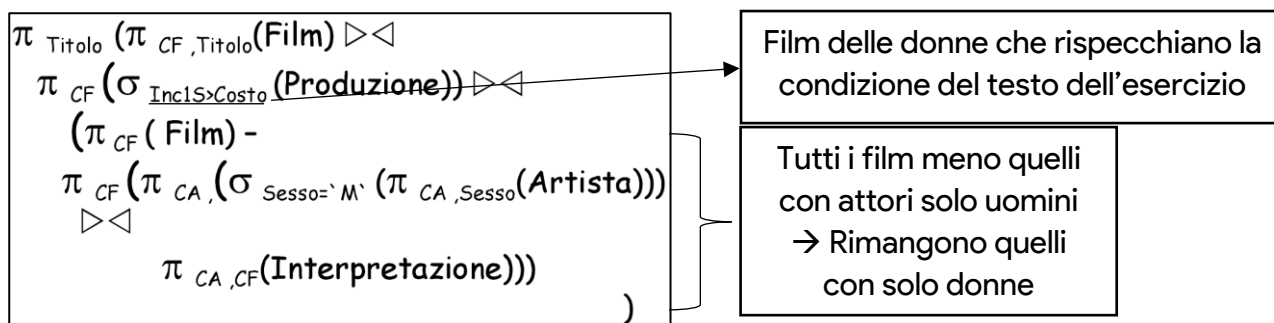
VAGLINI-PREPARAZIONE ESERCIZI

ESERCIZIO ALGEBRA RELAZIONALE

- Base di dati da considerare per l'esercizio

- Film(CodiceFilm, Titolo, CodiceRegista, Anno)
- Produzione(CasaProduzione, Nazionalità, CodiceFilm, Costo, Incasso1annoSala)
- Artista(CodiceAttore, Cognome, Nome, Sesso, DataDiNascita, Nazionalità)
- Interpretazione(CodiceFilm, CodiceAttore, Personaggio, SessoPersonaggio)
- Regista(CodiceRegista, Cognome, Nome, Sesso, DataDiNascita, Nazionalità)
- Noleggio(CodiceFilm, Incasso1annoVideo, Incasso1annoDVD)

"Titoli di film con solamente attori donna che abbiano incassato in sala più del proprio costo"



LEFT OUTER JOIN

$(= \bowtie \leftarrow)$ → Il join esterno sinistro mantiene le tuple che non si combinano della prima relazione (di SX) riempiendo con NULL la seconda relazione (di DX)

RIGHT OUTER JOIN

$(\bowtie \leftarrow =)$ → Il join esterno destro mantiene le tuple che non si combinano della seconda relazione (di DX) riempiendo con NULL la prima relazione (di SX)

FULL OUTER JOIN

$(= \bowtie \leftarrow =)$ → Il join esterno completo mantiene le tuple che non si combinano di entrambe le relazioni riempiendole con NULL

VAGLINI-PREPARAZIONE ESERCIZI

ESERCIZIO JOIN ESTERNO

- Considero le seguenti tabelle

Impiegati	Matricola	Nome	Età	Stipendio	Supervisione	Impiegato	Capo
	7309	Rossi	34	45000		7309	5698
	5998	Bianchi	37	38000		5998	5698
	9553	Neri	42	35000		9553	4076
	5698	Bruni	43	42000		5698	4076
	4076	Mori	45	50000		4076	8123
	8123	Lupi	46	60000			

“Trovare la matricola dei capi degli impiegati che guadagnano tutti più di 40000 euro”

$\pi_{\text{Capo}} (\sigma_{\text{Matricola}=\text{Null}} ($
Supervisione = $\triangleright \triangleleft$ $\text{Impiegato}=\text{Matricola}$
 $\sigma_{\text{Stipendio} \leq 40000}(\text{Impiegati}))$

//Da osservare la selezione iniziale = NULL

1° passo

$\sigma_{\text{Stipendio} \leq 40000}(\text{Impiegati})$

→ Seleziono le tuple degli impiegati che guadagnano meno o uguale a 40000

Impiegati	Matricola	Nome	Età	Stipendio
	5998	Bianchi	37	38000
	9553	Neri	42	35000

2° passo

Supervisione = $\triangleright \triangleleft$ $\text{Impiegato}=\text{Matricola}$

→ Faccio il JOIN ESTERNO SX, dove c'è la combinazione tra SUPERVISIONE e il risultato del passo 1°, sulla condizione Impiegato=Matricola

Le tuple di SUPERVISIONE che non si combinano con il risultato al passo 1° le mantengo, mettendoci a DX NULL (cioè alle tuple del risultato del passo 1°)

Impiegato	Capo	Matricola	Nome	Età	Stipendio
5998	5698	5998	Bianchi	37	38000
9553	4076	9553	Neri	42	35000
5698	4076	NULL	NULL	NULL	NULL
4076	8123	NULL	NULL	NULL	NULL
7309	5698	NULL	NULL	NULL	NULL

3° passo

$\sigma_{\text{Matricola}=\text{Null}}$

→ Seleziono dal risultato del 2° passo gli impiegati che hanno la Matricola = NULL (che sarebbero quelli che hanno lo stipendio sopra i 40000)

Impiegato	Capo	Matricola	Nome	Età	Stipendio
5698	4076	NULL	NULL	NULL	NULL
4076	8123	NULL	NULL	NULL	NULL
7309	5698	NULL	NULL	NULL	NULL



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO ESERCIZIO JOIN ESTERNO

4° passo

π_{Capo}

→ Dal 3° passo ne proietto il capo che rappresenta la soluzione al testo dell'esercizio

Capo
4076
8123
5698

PROIEZIONE GENERALIZZATA

$\pi_{F_1, \dots, F_n}(R)$

- $F_i \rightarrow$ Espressioni aritmetiche di attributi di R

ESEMPIO

$\pi_{\text{Cliente}, \text{Credito-Spese}}(\text{Conto})$

//Credito-Spese = Nuovo attributo che ha come valore il risultato di quell'espressione algebrica

OPERATORI DI AGGREGAZIONE: SOMMA, CONTEGGIO, CONTEGGIO SENZA DUPLICATI, MINIMO E MASSIMO

$\text{sum}_{A_i}(R)$

$\text{count}_{A_i}(R)$

$\text{count-distinct}_{A_i}(R)$

$\text{min}_{A_i}(R)$

$\text{max}_{A_i}(R)$

→ Sono operatori aggregati che si possono usare su singoli attributi di una tabella

ESEMPI DI OPERATORI DI AGGREGAZIONE

$\text{sum}_{\text{Spese}}(\text{Conto})$

$\text{count}_{\text{Cliente}}(\text{Conto})$

$\text{max}_{\text{Credito}}(\text{Conto})$

$\text{count-distinct}_{\text{Cliente}}(\text{Conto})$

VAGLINI-PREPARAZIONE ESERCIZI

RAGGRUPPAMENTO

$A^x_1, \dots, A^x_n \text{ } G_{\text{sum/count/min/max}}(A^y_i) (R)$

- $A^x_i \rightarrow$ Attributo su cui si fa il raggruppamento
- $\text{sum / count / min / max} \rightarrow$ Funzioni aggregate che si applicano all'attributo A^y_i (e possono essercene tante di funzioni)
- $R \rightarrow$ Relazione su cui si applica il tutto

ESEMPIO

$\text{Cliente } G_{\text{sum}(\text{Credito})}(\text{Conto})$

// 1 riga per ogni cliente e il risultato associato è la somma di tutti i suoi crediti

DIVISIONE

$:$ \rightarrow Operatore derivato utile per le interrogazioni di tipo universale, quando ci si riferisce a tutto il database

ESEMPIO

“Trovare i nomi dei clienti che hanno un conto corrente in tutte le filiali di banca di Pisa”

$\Pi_{CN,BN}(\text{depositor} \triangleright \triangleleft \text{account}) \div$
 $\Pi_{BN}(\sigma_{BC='Pisa'}(\text{branch}))$

// Tutti i clienti che hanno il conto in qualche filiale DIVISO tutte le banche di Pisa e ci viene fuori \rightarrow Clienti che hanno un conto in tutte le filiali della banca di Pisa

VISTA

NomeVista = F \rightarrow Serve a dare un nome ad un'espressione (F) e si può richiamare quante volte vogliamo

OSSERVAZIONE:

- Quando la vista viene chiamata all'esecuzione si ri-esponde (senza accorgersene)
- Con le viste posso evitare di utilizzare l'operatore di ridenominazione

ESEMPIO

Supervisione =
 $\pi_{\text{Impiegato, Capo}}(\text{Afferenza} \triangleright \triangleleft \text{Direzione})$

// SUPERVISIONE è il nome della vista e il resto dopo l'uguale è l'espressione
 \rightarrow Viene chiamata eseguendola così:

$\sigma_{\text{Capo}='Leoni'}(\text{Supervisione})$

VAGLINI-PREPARAZIONE ESERCIZI

VISTA 'MULTIPLA' E RIDENOMINAZIONE SU UNA VISTA

```
V = NOLEGGI ▷ ◁Auto=Targa AUTOVETTURE  
V1 = V ▷ ◁(Cliente=Cliente') ∧ (Modello=Modello') (ρX'←X V)  
πCliente NOLEGGI - πCliente (V1)
```

V è una creazione della prima vista e ci metto i noleggi di autovetture, che mi serve per prenderne il modello

V1 è la creazione della seconda vista alla quale gli viene assegnato una combinazione in join sulla prima vista e su un'espressione riguardante quest'ultima, si prendono i noleggi da parte di un cliente e si fa il SELF JOIN su questo, rinominandone gli attributi e si considera che

- Il cliente sia lo stesso
- Il modello sia differente

→ Ho quindi i clienti che hanno fatto almeno più di un noleggio, con auto differenti

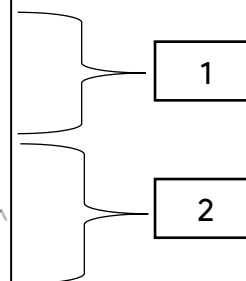
La query finale invece mi dà la proiezione su tutti i clienti dei noleggi meno i clienti che hanno fatto almeno due noleggi con auto diverse

→ Mi rimangono quindi quelli che non hanno noleggiato macchine di modello diverso, cioè solamente di quel modello

ESERCIZIO CON QUANTIFICATORI

Definire in algebra relazionale una query che produca la lista dei titoli dei film che "Marcello Mastroianni" ha interpretato senza "Sofia Loren".

```
{Titolo: t |  
  Film(CodFilm: cf, Titolo: t, CodRegista: cr, Anno: a) ∧  
  Attore(CodAttore: ca, Cognome: c, Nome: n, Sesso: s, DataNascita: dn, Nazionalità: nz) ∧  
  Interpretazione(CodFilm: cf, CodAttore: ca, Personaggio: p, SessoPersonaggio: sp) ∧  
  (c = "Mastroianni") ∧ (n = "Marcello") ∧  
  ¬ (∃ ca' (∃ p' (∃ sp' (∃ c' (∃ n' (∃ dn' (∃ nz' (  
    Film(CodFilm: cf', Titolo: t', CodRegista: cr', Anno: a') ∧  
    Attore(CodAttore: ca', Cognome: c', Nome: n', Sesso: s', DataNascita: dn', Nazionalità: nz') ∧  
    Interpretazione(CodFilm: cf', CodAttore: ca', Personaggio: p', SessoPersonaggio: sp') ∧ c' = "Loren" ∧ n' = "Sofia Loren"))))))))
```



1- Tabelle dei film, dell'attore e dell'interpretazione, il cui attore è Marcello Mastroianni

2- Impongo che non esistano gli attributi inerenti alle tabelle dei film, dell'attore e dell'interpretazione, la cui attrice è Sofia Loren

→ Ho quindi i titoli dei film dove è presente **solo** Marcello Mastroianni

VAGLINI-PREPARAZIONE ESERCIZI

ESERCIZIO: NON APPARTENENZA IN ALGEBRA RELAZIONALE

Elencare tutte le tuple che **non hanno** X

→ Equivale a dire **tutte le tuple** – tutte le tuple che hanno X

ESERCIZIO: DIVERSITÀ NEL JOIN IN ALGEBRA RELAZIONALE

$\pi_{\text{nome}, \text{cognome}}(\pi_{\text{numero}}(\sigma_{\text{nome}=\text{"Bilancio"}}(\text{Commissione})))$
 $\bowtie_{\text{commissione} \neq \text{numero}}$
 $\pi_{\text{commissione}, \text{nome}, \text{cognome}}(\text{Deputato})$

Elencare i deputati membri di una commissione **diversa** dalla Commissione Bilancio

ESERCIZIO: DIVERSITÀ NEL JOIN SUL CALCOLO DEI DOMINI

$\{\text{nome:n, cognome:c} \mid \text{Commissione}(\text{numero: num, nome: nc, presidente:pr}) \wedge \text{Deputato}(\text{codice:cd, nome:n, cognome:c, commissione:num', provincia:pv, collegio:cg}) \wedge \text{nc}=\text{'Esteri'} \wedge \text{num} \neq \text{num'}\}$

Elencare i deputati presenti in commissioni **diverse** dalla Commissione Esteri

ESERCIZIO: DIVERSITÀ NEGLI ATTRIBUTI SUL CALCOLO DEI DOMINI

$\{\text{nome:np} \mid \text{Provincia}(\text{sigla: sg, nome: np, regione: cr}) \wedge \text{Deputato}(\text{codice:cd, nome:n, cognome:c, commissione:num, provincia: sg, collegio: cg}) \wedge \text{Regione}(\text{codice: cr, nome:nr}) \wedge (\text{nr} \neq \text{'Sicilia'} \wedge \text{nr} \neq \text{'Toscana'})\}$

Elencare i nomi delle province **non** della Sicilia **né** della Toscana in cui c'è almeno un deputato eletto

ESERCIZIO: NON ESISTENZA ALGEBRA RELAZIONALE-CALCOLO DEI DOMINI

1. $\pi_{AB}(R) - \pi_{AB}(R \bowtie_{C=D} S)$
2. $\{A:a, B:b \mid \text{not exists } e, c . R(A:a, B:b, C:c) \text{ and } S(D:c, E:e) \}$

1. $\pi_{AB}(R) - \pi_{AB}(R \bowtie_{C=D} S)$
2. $\{A:a, B:b \mid R(A:a, B:b, C:c) \text{ and not exists } e . S(D:c, E:e) \}$

AND e NOT EXISTS Sono intercambiabili, però attenzione:

- Nel 2° non esiste la variabile della relazione S quindi il JOIN va a vuoto senza produrre nulla
- Nel 1° il JOIN va a VUOTO perché non esistono entrambe le variabili della relazione S

VAGLINI-PREPARAZIONE ESERCIZI

ESERCIZIO: RIDENOMINAZIONI

ALGEBRA RELAZIONALE -CALCOLO DEI DOMINI

1° CASO

$$(R_3 \bowtie_{G=E} R_2) \cup (\rho_{DG \leftarrow AC}(\pi_{ACEF}(R_1 \bowtie_{B=F} R_3)))$$

$$\{ D: d, G: g, E: e, F: f \mid R_3(E: e, F: f) \wedge ((R_2(D: d, G: g) \wedge (g=e)) \vee (R_1(A: d, B: b, C: g) \wedge (b=f))) \}$$

2° CASO

$$\rho_{E,F \leftarrow A,B}(\pi_{A,B,D}(R_1 \bowtie R_2))$$

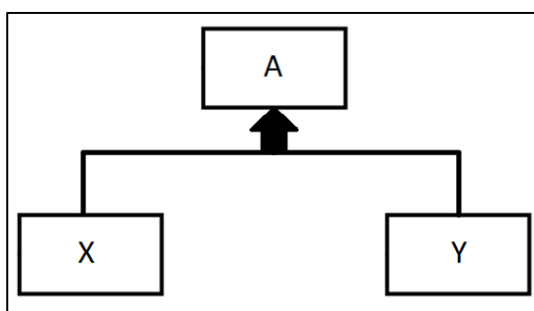
$$\{ D: d, E: a, F: b \mid R_1(A: a, B: b, C: c, D: d) \text{ and } R_2(B: b, D: d) \}$$

→ Nel 1° caso la ridenominazione $DG \leftarrow AC$ la faccio nella relazione R_1 interna dopo il ' \mid ', perché, il JOIN è fatto su B e F e quindi non ho bisogno di A e C e posso cambiarli subito

→ Nel 2° caso invece la ridenominazione $EF \leftarrow AB$ è fatta prima di ' \mid ', perché il JOIN è naturale ed è fatto sugli attributi comuni a R_1 e R_2 , e R_1 ed R_2 hanno in comune d , che non ha bisogno di ridenominazione e b , che va ridenominato; se lo ridenominavo dopo il ' \mid ' il NATURAL JOIN non viene fatto e quindi sarebbe un errore

TRADUZIONE GENERALIZZAZIONI

- Data la seguente generalizzazione



Considererò per la suddivisione o l'accorpamento della generalizzazione i casi di:

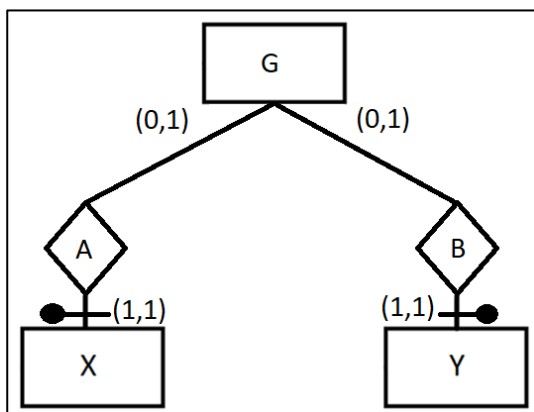
1. Separare l'entità padre dalle figlie tramite delle associazioni
2. Accorpate le entità figlie nell'entità padre



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO TRADUZIONE GENERALIZZAZIONI

1. La generalizzazione diventa la seguente

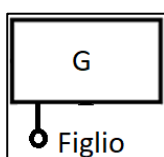


→ Metto la cardinalità da G verso le associazioni A e B come $(0,1)$ → Perché G può essere o l'uno o l'altro

→ Metto la cardinalità da X e da Y come $(1,1)$ verso G → Perché sia X, sia Y, devono necessariamente essere parte dell'entità padre G, dato che ne ereditano gli attributi

→ Metto la chiave esterna perché altrimenti non saprei a cosa mi riferisco

2. La generalizzazione diventa la seguente

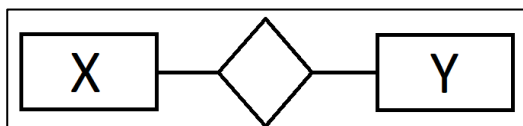


→ Aggiungo un attributo, che mi andrà a rappresentare uno dei due figli, ai quali mi riferirò al momento dell'utilizzo

OSSERVAZIONE: Per questi schemi non tengo conto il fatto che possono avere tanti attributi (li ometto per semplicità)

INSERIMENTO CARDINALITÀ NELL' E-R

Considero le X e le Y entità collegate da un'associazione e, a seconda dei casi dati dal testo dell'esercizio, metto la cardinalità



VAGLINI-PREPARAZIONE ESERCIZI

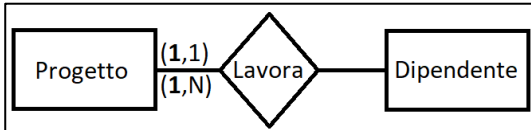
...CONTINUO INSERIMENTO CARDINALITÀ NELL' E-R

CASO: AD X FA PARTE ALMENO UN Y

→ La cardinalità minima da X verso Y è 1 → Posso avere quindi (1,1) / (1,N)

ESEMPIO

"In un progetto lavora almeno un dipendente"

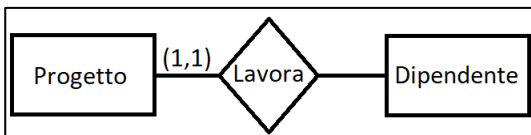


CASO: X HA A CHE FARE CON UNO ED UN SOLO Y

→ La cardinalità da X verso Y è (1,1)

ESEMPIO

"In un progetto lavora uno ed un solo un dipendente"

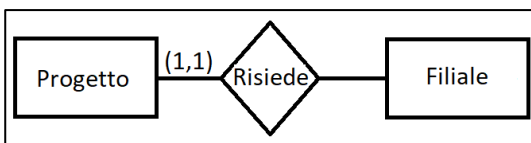


CASO: UN VALORE DI X HA A CHE FARE CON UN VALORE Y

→ La cardinalità da X verso Y è (1,1)

ESEMPIO

"Un progetto ha sede in una filiale"

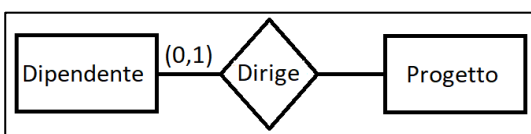


CASO: X PUÒ AVERE A CHE FARE CON AL PIÙ UN SOLO Y

→ La cardinalità da X verso Y è (0,1)

ESEMPIO

"Un dipendente può dirigere al più un progetto"



VAGLINI-PREPARAZIONE ESERCIZI

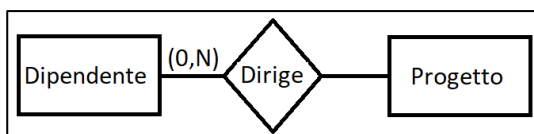
...CONTINUO INSERIMENTO CARDINALITÀ NELL' E-R

CASO: X PUÒ AVERE A CHE FARE CON CERTO NUMERO DI VALORI DI Y

→ La cardinalità da X verso Y è (0,N)

ESEMPIO

“Un dipendente può dirigere tanti progetti”

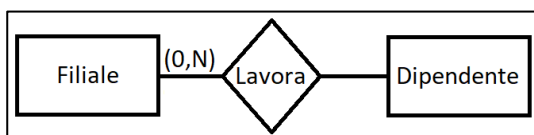


CASO: AD X PUÒ NON FARE PARTE NESSUN VALORE DI Y, OPPURE AD X POSSONO FARE PARTE TANTI VALORI DI Y

→ La cardinalità da X verso Y è (0,N)

ESEMPIO

“In una filiale può non lavorare nessun dipendente e vi possono lavorare più dipendenti”



CASO: X PUÒ AVERE A CHE FARE CON UN CERTO NUMERO DI VALORI DI Y, MA CON ALMENO UNO SICURAMENTE AVRÀ A CHE FARE

→ La cardinalità da X verso Y è (1,N)

ESEMPIO

“Un dipendente può lavorare in più progetti, ma lavora in almeno un progetto”

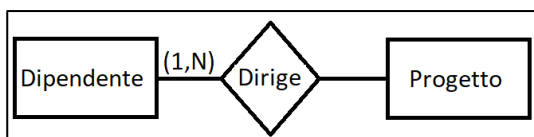
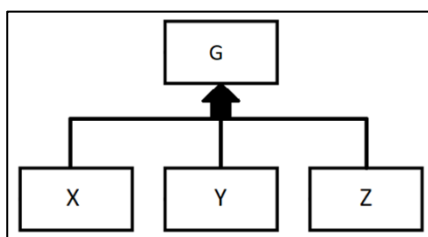


TAVOLA DEI VOLUMI: OCCORRENZE DEL PADRE DI UNA GENERALIZZAZIONE

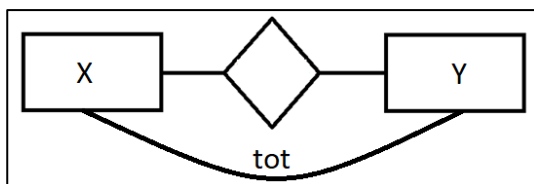


- Per trovare le occorrenze di A basta che sommi le occorrenze di X, Y e Z

→ **A=X+Y+Z**

VAGLINI-PREPARAZIONE ESERCIZI

TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ENTITÀ (CARDINALITÀ NON CONSIDERATA)



CASO: VENGONO USATI IN MEDIA UN CERTO NUMERO DI VALORI (TOT) DI Y PER CIASCUN VALORE DI X

- X: nota
- tot: nota
- Y: incognita

- Per trovare le occorrenze di Y devo prendere i valori di X e il valore che lega X a Y, cioè tot, e devo moltiplicarli tra loro

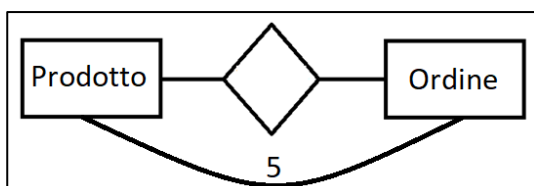
→ Questo perché, se considero che per 1 valore di X se ne usano (tot*1) di Y

→ Per tutti i valori di X se ne usano (tot*X) di Y

→ **$Y = \text{tot} * X$**

ESEMPIO

“Vengono effettuati in media 5 ordini per ciascun prodotto”

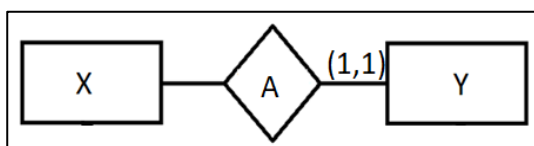


- X=Prodotto=600
- tot=5
- Y=Ordine

→ Per trovare l'occorrenza di Ordine considero che, se 1 prodotto mi dà 5 ordini, allora 600 prodotti mi daranno (600*5) ordini

→ Ordine=5*600=3000

TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ENTITÀ CON LE OCCORRENZE DELL'ASSOCIAZIONE MANCANTI



- X: nota
- A: **non** nota
- Y: incognita



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ENTITÀ CON LE OCCORRENZE DELL'ASSOCIAZIONE MANCANTI

- Per trovare le occorrenze di Y devo prendere i valori di X e devo combinarli con tutti i suoi possibili valori tranne il primo che prendo in considerazione

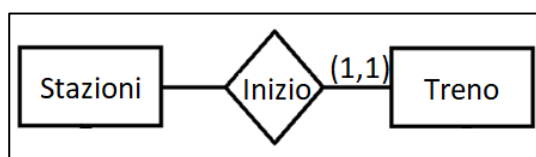
→ Ciò è dovuto alla cardinalità (1,1) sulla tabella Y

→ $Y = X * (X - 1)$

//Di conseguenza anche le occorrenze di A saranno quelle dell'entità Y

ESEMPIO

"Trovare le occorrenze di Treno"

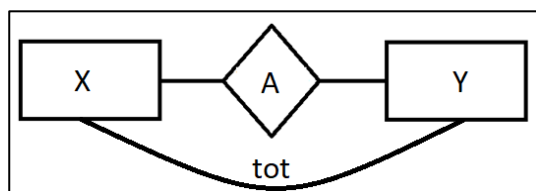


- $X = \text{Stazioni} = 100$

→ Le occorrenze dell'entità Treno sono date dalle combinazioni sulle Stazioni

→ $\text{Treno} = 100 * 99 = 9900$

TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ASSOCIAZIONE (CARDINALITÀ NON CONSIDERATA)



CASO: CIASCUN VALORE DI Y UTILIZZA IN MEDIA UN CERTO NUMERO DI VALORI DIVERSI (TOT) DI X

- X: nota (superflua ai fini del calcolo)
- tot: nota
- Y: nota
- A: incognita

- Per trovare le occorrenze di A devo prendere i valori di Y e il valore che lega X a Y, cioè tot, e devo moltiplicarli tra loro

→ Questo perché, se considero che per 1 valore di Y se ne usano (tot*1) di X

→ Per tutti i valori di Y se ne usano (tot*Y) di X

→ $A = \text{tot} * Y$

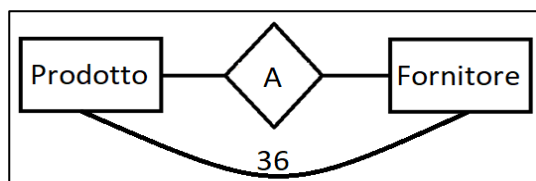


VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ASSOCIAZIONE (CARDINALITÀ NON CONSIDERATA)

ESEMPIO

"Ciascun fornitore vende in media 36 prodotti diversi"

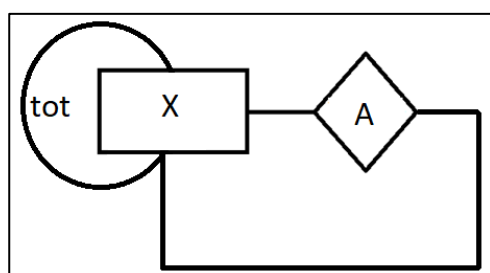


- $X = \text{Prodotto} = 600$
- $\text{tot} = 36$
- $Y = \text{Fornitore} = 50$

→ Per trovare l'occorrenza dell'associazione A, che posso chiamare *Acquista*, considero che, se 1 fornitore in media vende 36 prodotti diversi, allora 50 fornitori mi daranno $(50 \cdot 36)$ prodotti diversi

→ $A = 50 \cdot 36 = 1800$

TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ASSOCIAZIONE, CON RICORSIONE (CARDINALITÀ NON CONSIDERATA)



CASO: CIASCUN VALORE DI X UTILIZZA IN MEDIA UN CERTO NUMERO DI VALORI (TOT) DI ABBINAMENTI

- X: nota
- tot: nota
- A: incognita

- Per trovare le occorrenze di A devo prendere i valori di X e il valore che lega X a sé stesso, cioè tot, e devo moltiplicarli tra loro

→ Questo perché, se considero che per 1 valore di X se ne usano $(\text{tot} \cdot 1)$ di sé stesso

→ Per tutti i valori di X se ne usano $(\text{tot} \cdot X)$ di sé stesso

→ **$A = \text{tot} \cdot X$**

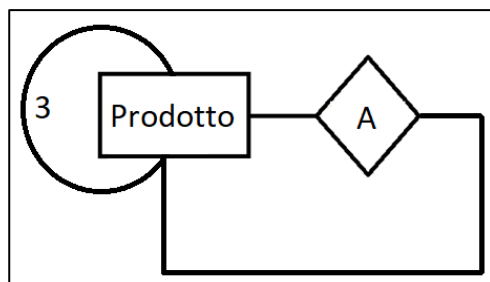


VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ASSOCIAZIONE, CON RICORSIONE (CARDINALITÀ NON CONSIDERATA)

ESEMPIO

"Ciascun prodotto ha in media 3 abbinamenti"

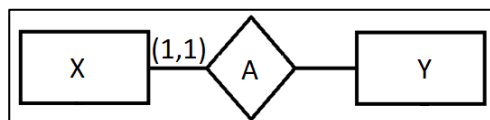


- $X = \text{Prodotto} = 600$
- $\text{tot} = 3$
- $Y = \text{Fornitore} = 50$

→ Per trovare l'occorrenza dell'associazione A, che posso chiamare *Consiglia*, considero che, se 1 prodotto in media è abbinato a sé stesso 3 volte, allora 600 prodotti sono abbinati a sé stessi ($600 \cdot 3$) volte

→ $A = 600 \cdot 3 = 1800$

TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ASSOCIAZIONE AVENTE UNA CARDINALITÀ (1,1)



CASO: CIASCUN VALORE DI X APPARTIENE AD UN SOLO (OPPURE AD UNO, ED UNO SOLO) VALORE DI Y

- X: nota
- Y: nota (superflua ai fini del calcolo)
- A: incognita

- Quando ho cardinalità (1,1) e devo trovare le occorrenze dell'associazione A, da X verso Y

→ Le occorrenze di A sono le stesse di quelle di X, **indipendentemente dalla cardinalità dall'entità Y all'associazione A**

→ **A=X**

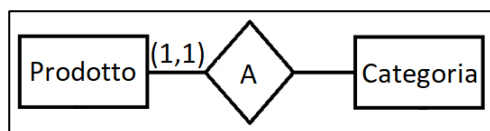


VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ASSOCIAZIONE AVENTE UNA CARDINALITÀ (1,1)

ESEMPIO

Ciascun prodotto appartiene ad una, ed una sola, categoria

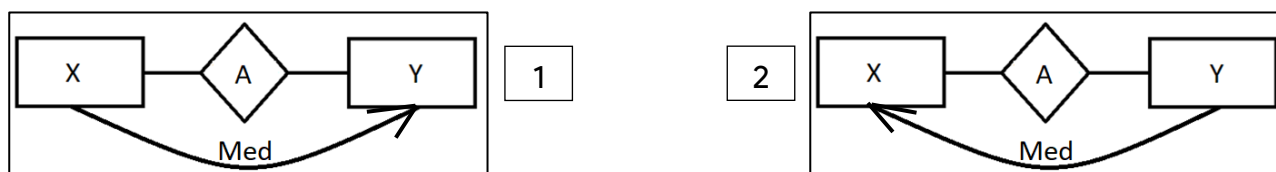


- $X = \text{Prodotto} = 600$
- $Y = \text{Categoria} = 30$

→ Per trovare l'occorrenza dell'associazione **A**, che posso chiamare *Appartiene*, considero che, se 1 prodotto appartiene ad una sola categoria, allora tra il prodotto e la categoria ci sono tanti legami (coppie tra le due entità), quante sono le occorrenze dell'entità **Prodotto**

→ $A = X = 600$

TAVOLA DEI VOLUMI: OCCORRENZE IN MEDIA TRA DUE ENTITÀ (CARDINALITÀ NON CONSIDERTA)



CASO: SI DEVONO TROVARE, IN MEDIA (MED), QUANTI VALORI DI X SONO COLLEGATI, TRAMITE L'ASSOCIAZIONE A ALL'ENTITÀ Y (1) O VICEVERSA, CIOÈ DA Y A X (2)

- **X**: nota (superflua ai fini del calcolo)
- **Y**: nota
- **A**: nota
- **Med**: incognita di 2 tipi, a seconda del testo dell'esercizio

- Per trovare la media delle occorrenze tra un'entità di partenza **X** e un'entità di arrivo **Y**

→ Devo innanzitutto, considerare le coppie che si formano nell'associazione, quindi le occorrenze dell'associazione

→ Poi, devo prendere queste coppie e vedere a quanti valori sono associati sull'entità di arrivo **Y**, facendoci una divisione

→ **Med = A/Y** - **Med = A/X**

//TIPO-1 e TIPO-2

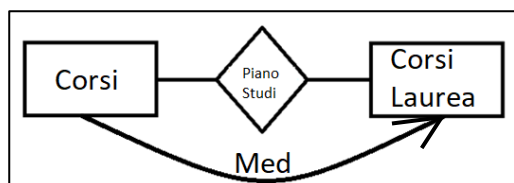


VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO TAVOLA DEI VOLUMI: OCCORRENZE IN MEDIA TRA DUE ENTITÀ (CARDINALITÀ NON CONSIDERTA)

ESEMPIO

“Quanti corsi ci sono in media nel piano di studi di un corso di laurea? (TIPO-1)”



- $X = \text{Corsi} = 50$
- $Y = \text{CorsiLaurea} = 5$
- $A = \text{PianoStudi} = 75$

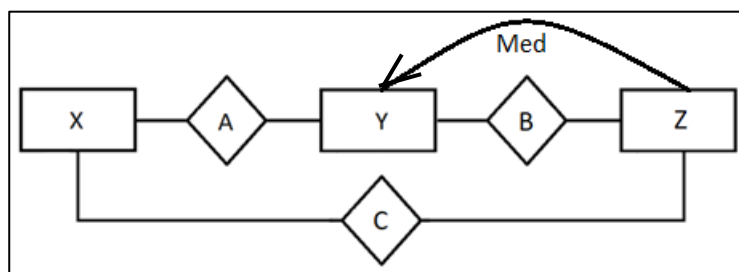
- Per trovare l'occorrenza media (Med) considero le coppie che si formano nell'associazione PianoStudi, che sono 75

→ Di queste coppie, prendo quelle collegate ai corsi di laurea dividendo per le occorrenze dell'entità CorsiLaurea, cioè 5

→ $\text{Med} = 75 / 5 = 15$

//Ciascun corso di laurea ha in media nel proprio piano di studi 15 corsi

TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ASSOCIAZIONE E DUE PERCORSI (CARDINALITÀ NON CONSIDERTA)



CASO: SI DEVONO TROVARE LE OCCORRENZE DELL'ASSOCIAZIONE C CHE COLLEGA L'ENTITÀ X ALL'ENTITÀ Z

- X: nota
- Y: nota
- Z: nota (superflua ai fini del calcolo)
- A: nota (superflua ai fini del calcolo)
- B: nota
- Med: incognita intermedia (TIPO-2)
- C: incognita finale

- Per trovare le occorrenze dell'associazione C, si deve suddividere il problema in due parti separate



VAGLINI-PREPARAZIONE ESERCIZI

... CONTINUO TAVOLA DEI VOLUMI: OCCORRENZE DI UN'ASSOCIAZIONE E DUE PERCORSI (CARDINALITÀ NON CONSIDERTA)

1°PARTE

- Dobbiamo innanzitutto considerare l'entità Z e fare un conto intermedio darà il collegamento dall'entità Z a quella Y
→ Si devono trovare, in media (Med), quanti valori di Z sono collegati, tramite l'associazione B all'entità Y

→ **Med=B/Y**

//Come nella spiegazione precedente, solo che qua è da DX verso SX e non da SX verso DX

2°PARTE

- Dato che l'entità Y è di tramite per arrivare a Z, devo prendere tutte le occorrenze di X e moltiplicarle per il collegamento intermedio, ottenuto su Y al punto precedente (Med)

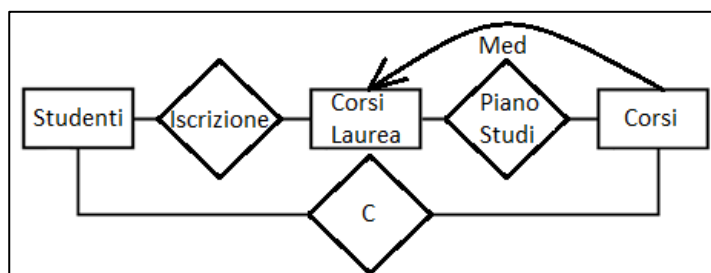
→ Ciò mi permetterà di arrivare a Z, tramite l'entità intermediaria Y

→ **C=X*Med**

//che sarebbe $X*(B/Y)$

ESEMPIO

"Quante occorrenze ci sono nell'associazione Partecipazione?"



- X=Studenti=1000
- Y=CorsiLaurea=5
- Z=Corsi=50
- A=Iscrizione=1000
- B=PianoStudi=75

- Per trovare le occorrenze dell'associazione C, chiamata *Partecipazione*, divido il problema in due parti:

1°PARTE

- Per prima cosa, trovo quanti corsi ci sono in media nel piano di studi di un corso di laurea

→ $Med=75/5=15$

2°PARTE

- Considerando che uno studente può o meno essere iscritto ad un corso di laurea (indipendentemente dalla cardinalità), devo vedere tutti gli studenti a quanti corsi di laurea sono iscritti

→ Si fa ciò, moltiplicando le occorrenze di studente, che sono 1000, per la media dei corsi di ciascun piano di studi, di un corso di laurea 15 (Med)

→ $C=1000*15=15000$

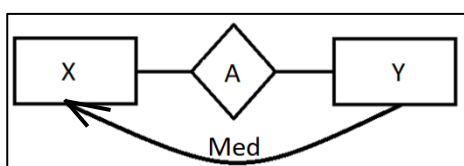
VAGLINI-PREPARAZIONE ESERCIZI

COSTO OPERAZIONI DI LETTURA E SCRITTURA

- Nel calcolo del costo delle singole operazioni di lettura e scrittura possiamo dire che
 - Ciascuna operazione di lettura si conta come una singola operazione → **L=1**
 - Ciascuna operazione di scrittura si conta come una doppia operazione → **S=2**

OSSERVAZIONE: Quando arrivo a modificare uno specifico attributo di un'entità, su quell'entità, devo prima leggerci l'attributo → **1 OPERAZIONE di L + 1 OPERAZIONE di S**, (S mi rappresenta il fatto di scrivere su quell'attributo); invece quando modifico un'entità e basta ci leggo direttamente, facendo → **1 OPERAZIONE di S**

OPERAZIONI ELEMENTARI TRA DUE ENTITÀ (CARDINALITÀ NON CONSIDERTA)



- X: nota
- Y: nota (superflua ai fini del calcolo)
- A: nota
- Med: incognita (TIPO-2)

- Per prima cosa vedo da dove devo partire per guardare le operazioni elementari, in base al testo dato dall'esercizio

→ Per esempio, considero di partire dall'entità X facendoci un accesso diretto in L/S

→ **1 OPERAZIONE (L/S) su X**

→ Conto quanti valori di Y ci sono in media (Med), passando dall'associazione A, all'entità X

→ **Med=A/X**

→ Accedo per Med volte sull'associazione A

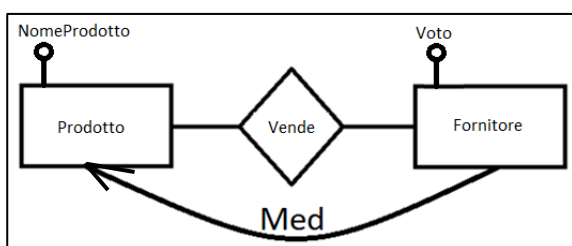
→ **Med volte di OPERAZIONE (L/S) su A**

→ Accedo all'entità Y direttamente con lo stesso valore (Med), dato che Y è collegato Med volte all'associazione A

→ **Med volte di OPERAZIONE (L/S) su Y**

ESEMPIO

“Elenco dei fornitori con voto medio più alto tra quelli che hanno un dato prodotto x presente in magazzino”



VAGLINI-PREPARAZIONE ESERCIZI

...COTINUO OPERAZIONI ELEMENTARI TRA DUE ENTITÀ (CARDINALITÀ NON CONSIDERTA)

- $X = \text{Prodotto} = 600$
- $Y = \text{Fornitore} = 50$
- $A = \text{Vende} = 1800$
- Med: incognita

- Devo partire dall'entità Prodotto, per sapere il nome del prodotto e devo arrivare alla tabella Fornitore per saperne il voto

- Faccio 1 accesso su Prodotto per recuperarne il nome e, controllo se questo è uguale a X
→ 1 OPERAZIONE di LETTURA su PRODOTTO

- Conto quanti fornitori vendono in media quel determinato prodotto X

→ $\text{Med} = 1800 / 600 = 3$

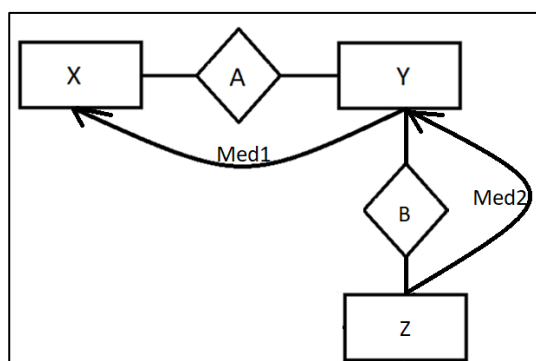
→ Quindi quel prodotto X è venduto in media da 3 fornitori

→ 3 OPERAZIONI di LETTURA su VENDE

- Per conoscere il voto richiesto nel testo, faccio un accesso diretto su Fornitore per 3 volte, dato che è legato su 3 fornitori, che vendono in media quel determinato prodotto X

→ 3 OPERAZIONI di LETTURA su FORNITORE

OPERAZIONI ELEMENTARI TRA TRE ENTITÀ (CARDINALITÀ NON CONSIDERTA)



- X: nota
- Y: nota (superflua ai fini del calcolo)
- A: nota
- Med1: incognita (TIPO-2)
- B: nota
- Med2: incognita (TIPO-2)
- Z: nota (superflua ai fini del calcolo)

- Per prima cosa devo decidere da dove devo partire e dove devo arrivare, per trovare e collegare tutti i dati necessari, richiesti dal testo dell'esercizio

- Suddivido il problema in tre parti in cui, nella prima parte mi collego da X a Y, nella seconda parte collego Z ad Y ed infine, nella terza parte collego i due collegamenti precedenti



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO OPERAZIONI ELEMENTARI TRA TRE ENTITÀ (CARDINALITÀ NON CONSIDERTA)

1°PARTE

- In questo caso parto dall'entità X e ci faccio un accesso diretto
- **1 OPERAZIONE (L/S) su X**
- Conto quanti valori di Y ci sono in media (Med1) passando dall'associazione A all'entità X
- **Med1=A/X**
- Accedo per Med1 volte sull'associazione A
- **Med1 volte di OPERAZIONE (L/S) su A**

2°PARTE

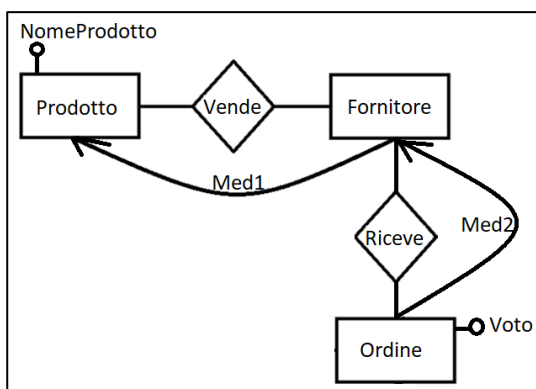
- Conto quanti valori di Z ci sono in media (Med2) passando dall'associazione B all'entità Y
- **Med2=B/Y**
- Accedo per Med2 volte sull'associazione B

3°PARTE

- Per ogni media di valori di Z collegati ad Y (Med2) multiplico per la media dei valori di Y sono collegati ad X e trovo gli accessi che devo fare sull'associazione B che chiamerò tot
- **tot=Med2*Med1**
- Accedo per tot volte sull'associazione B
- **tot volte di OPERAZIONE (L/S) su B**
- Accedo all'entità Z direttamente con lo stesso valore (tot), dato che Z è collegato tot volte all'associazione B
- **tot volte di OPERAZIONE (L/S) su Z**

ESEMPIO

"Elenco dei fornitori con voto medio più alto tra quelli che hanno un dato prodotto x presente in magazzino"



- X=Prodotto=600
- Y=Fornitore=50
- A=Vende=1800
- B=Riceve=3000
- Z=Ordine=3000

- Devo partire dall'entità Prodotto per sapere il nome del prodotto e devo arrivare alla tabella Ordine per saperne il voto



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO OPERAZIONI ELEMENTARI TRA TRE ENTITÀ (CARDINALITÀ NON CONSIDERTA)

1ª PARTE

- Faccio 1 accesso su Prodotto per recuperare il nome del prodotto e vedere se equivale a X
→ 1 OPERAZIONE di LETTURA su PRODOTTO
- Conto quanti fornitori vendono in media quel determinato prodotto X
→ $\text{Med1} = 1800 / 600 = 3$
→ Quindi quel prodotto X è venduto in media da 3 fornitori
→ 3 OPERAZIONI di LETTURA su VENDE

2ª PARTE

- Controllo quanti ordini vengono fatti in media dai fornitori
→ $\text{Med2} = 3000 / 50 = 60$
→ Quindi ogni fornitore riceve in media 60 ordini

3ª PARTE

- Dato che ogni fornitore riceve in media 60 ordini, e i fornitori sono in media 3 per prodotto
→ Per avere gli ordini ricevuti dai fornitori, che hanno quel determinato prodotto X in magazzino, dovrò moltiplicare 60 per 3
→ $\text{tot} = 60 * 3 = 180$
→ 180 OPERAZIONI di LETTURA su RICEVE
- Per conoscere il voto richiesto nel testo, faccio un accesso diretto su Ordine per 180 volte, dato che è legato alle 180 ricezioni di ordini da parte dei 3 fornitori, che vendono in media un determinato prodotto X
→ 180 OPERAZIONI di LETTURA su ORDINE

OSSERVAZIONE: Se, da Fornitore a Riceve, avessi avuto cardinalità (1,1) → Med2 sarebbe stato 1 (50/50), perché Riceve sarebbe stato 50 → Quindi avrei avuto che 1 fornitore avrebbe ricevuto in media 1 ordine

CALCOLO TOTALE DELLE OPERAZIONI ELEMENTARI (LETTURA/SCRITTURA)

- Per il totale delle operazioni elementari, sommo le operazioni di L/S, laddove siano maggiori di 1 dove ho solo letture, oppure anche =1 dove ci sono pure le scritture
→ **TOTALE OPERAZIONI NECESSARIE = Somma di tutte le operazioni (L/S)**

ESEMPIO

Considerando l'esempio delle OPERAZIONI ELEMENTARI TRA DUE ENTITÀ ho:

- 1 OPERAZIONE di LETTURA su PRODOTTO
- 3 OPERAZIONI di LETTURA su VENDE
- 3 OPERAZIONI di LETTURA su FORNITORE
- Totale operazioni necessarie: $3 + 3 = 6$



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO CALCOLO TOTALE DELLE OPERAZIONI ELEMENTARI (LETTURA/SCRITTURA)

ESEMPIO

Considerando l'esempio delle OPERAZIONI ELEMENTARI TRA TRE ENTITÀ ho:

- 1 OPERAZIONE di LETTURA su PRODOTTO
- 3 OPERAZIONI di LETTURA su VENDE
- 180 OPERAZIONI di LETTURA su RICEVE
- 180 OPERAZIONI di LETTURA su ORDINE
- Totale operazioni necessarie: $3+180+180=163$

OSSERVAZIONE: Laddove ho attributi su cui scriverci, valgono le singole letture, altrimenti se non ho attributi su cui scrivere, ma direttamente sull'entità, le singole letture non contano

FREQUENZA NEL CALCOLO OPERAZIONI ELEMENTARI (LETTURA/SCRITTURA)

- Data una certa frequenza ORARIA/GIORNALIERA/SETTIMANALE/MENSILE/ANNUALE, a seconda di come ci viene chiesto nel problema, basta che facciamo

- **FREQUENZA TOTALE = Frequenza data * totale delle operazioni elementari necessarie (L/S)**

ESEMPIO

Considerando che la soluzione dell'esempio delle OPERAZIONI ELEMENTARI TRA DUE ENTITÀ, sia ripetuta 10 volte al giorno

- La frequenza giornaliera sarà data da
- 6 Operazioni Elementari * 10 volte al giorno = 60 operazioni elementari al giorno

OSSERVAZIONE: Se di questa operazione ne volessi la frequenza mensile, basta che moltiplichi la frequenza giornaliera per i 30 giorni del mese, e otterrei la frequenza mensile

CONFRONTO FREQUENZE NEL CALCOLO OPERAZIONI ELEMENTARI (LETTURA/SCRITTURA)

- Se una serie di operazioni X viene svolta con una frequenza F (in più o in meno, dipendente dal segno di F), rispetto ad una serie di operazioni Y, allora il totale di operazioni elementari necessarie, con il relativo confronto tra frequenze, è dato da:

- **OPERAZIONI NECESSARIE CONFRONTATE = $Y + FX$**

ESEMPIO

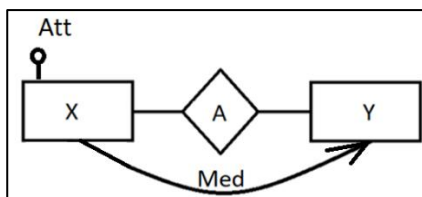
Considerando che le operazioni necessarie dell'esempio delle OPERAZIONI ELEMENTARI TRA DUE ENTITÀ (X), siano compiute 10 volte più frequentemente (F), rispetto alle operazioni necessarie dell'esempio delle OPERAZIONI ELEMENTARI TRA TRE ENTITÀ (Y)

- $163 + 6*10 = 193$ Totale operazioni elementari

// $Y+FX$

VAGLINI-PREPARAZIONE ESERCIZI

FREQUENZA NEL CALCOLO OPERAZIONI ELEMENTARI (LETTURA/SCRITTURA) E RIDONDANZE ATTRIBUTI

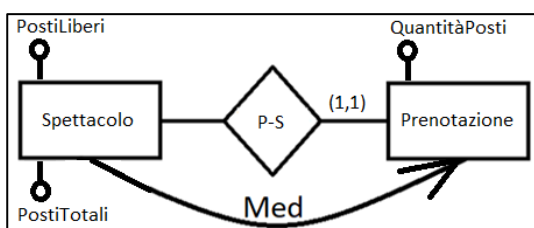


- X: nota
- Y: nota
- A: nota/da trovare
- Med: da trovare (TIPO-1)
- Att: già inserito oppure ottenibile da altra/e tabelle

- Per trovare il numero di operazioni elementari necessarie considero la frequenza data e, in base alle operazioni di L/S vedo a quali tabelle accedere con e senza attributo

ESEMPIO

"Inserimento di una prenotazione per un certo spettacolo, con frequenza giornaliera $f_2=10000$ "



- X=Spettacolo=1000
- Y=Prenotazione=1000000
- Att=PostiLiberi, già inserito oppure ottenuto da:
→ PostiTotale - (Somma di tutti i valori di QuantitàPosti) (****)

- Distinguo 2 casi: 1° caso - PostiLiberi non è presente, 2° caso - PostiLiberi è presente
- Presuppongo di aver già trovato P-S=100000, dato che ho cardinalità (1,1) su Prenotazione

1°CASO

- Parto dall'entità Spettacolo per leggere PostiTotale (utile per il calcolo (****))

→ 1 OPERAZIONE di LETTURA su SPETTACOLO

- Conto quanti spettacoli ci sono in media per ciascuna prenotazione, sull'associazione P-S

→ $Med = 1000000 / 100000 = 1$

- L'associazione P-S mi collega le due entità; quindi, dentro ho anche le chiavi di Prenotazione, su cui devo scriverci per Med volte

→ 1 OPERAZIONE di SCRITTURA su P-S

- Per conoscere QuantitàPosti, utile nella somma, faccio un accesso diretto su Prenotazione, scrivendo per 1 volta, dato che è legato mediamente 1 volta con lo Spettacolo

→ 1 OPERAZIONI di SCRITTURA su PRENOTAZIONE



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO FREQUENZA NEL CALCOLO OPERAZIONI ELEMENTARI (LETTURA/SCRITTURA) E RIDONDANZE ATTRIBUTI

→ TOTALE OPERAZIONI 1° CASO = $1S+1S$

//Valgono doppio

→ 4 Operazioni elementari * 10000 volte al giorno

→ 40000 Operazioni elementari

//La lettura iniziale non è considerata

2°CASO

- Parto dall'entità Spettacolo per leggere l'attributo PostiLiberi e anche scriverci sopra

→ 1 OPERAZIONE di LETTURA + 1 OPERAZIONE di SCRITTURA su SPETTACOLO

- Conto quanti spettacoli ci sono in media per ciascuna prenotazione, sull'associazione P-S

→ $Med=1000000/100000=1$

- L'associazione P-S mi collega le due entità; quindi, dentro ho anche le chiavi di

Prenotazione, su cui devo scriverci per Med volte

→ 1 OPERAZIONE di SCRITTURA su P-S

- Infine, faccio un accesso diretto su Prenotazione, scrivendo per 1 volta, dato che è legato mediamente 1 volta con lo Spettacolo

→ 1 OPERAZIONI di SCRITTURA su PRENOTAZIONE

TOTALE OPERAZIONI 2° CASO = $1L+1S+1S+1S$

//S Valgono doppio

→ 7 Operazioni elementari * 10000 volte al giorno

→ 70000 Operazioni elementari

//La lettura iniziale adesso è considerata

OSSERVAZIONE: Quando devo leggere/scrivere su attributi, oppure su attributi calcolati considero due casi:

1. Se parto dall'entità di SX verso quella di DX, il numero medio di occorrenze sarà del TIPO-1
2. Se parto dall'entità di DX verso quella di SX, il numero medio di occorrenze sarà del TIPO-2

REGOLE DI ARMSTRONG

- Le **regole di inferenza di Armstrong** permettono di derivare tutte le FD che sono implicate da un dato insieme iniziale:

1. **Riflessività**: "Se Y è sottoinsieme di X allora vale $X \rightarrow Y$ "

Se prendo un qualunque sottoinsieme banale di X, posso scrivere una dipendenza, che rappresenta il fatto che X determina questo sottoinsieme

2. **Additività** (oppure **Espansione**): "Se $X \rightarrow Y$, allora $XZ \rightarrow YZ$ (per qualunque Z)"

Se ho un insieme X con un certo numero di attributi, ed un insieme Y con un certo numero di attributi, nel caso aggiungessi uno o più attributi a DX o a SX della dipendenza, questa continua a valere

3. **Transitività**: "Se $X \rightarrow Y$ e $Y \rightarrow Z$ allora $X \rightarrow Z$ "

Pseudo-transitività, applicata tra due FD



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO REGOLE DI ARMSTRONG

- Per le regole di Armstrong ci sono proprietà da rispettare, tramite alcuni teoremi:
 - Teorema della correttezza di un insieme: Applicando le regole di inferenza di Armstrong, ottengo solo FD implicate da qualunque insieme di partenza
 - Teorema della completezza: Si possono ottenere tutte le FD implicate, quindi l'insieme completo
 - Teorema della minimalità: Non posso togliere nessuna di queste regole, altrimenti perderei la completezza
- Ci sono altre regole provenienti da quelle di inferenza e si chiamano **regole derivate di Armstrong** e sono:
 4. **Regola di unione**: "Se ho $X \rightarrow Y$ e $X \rightarrow Z$ posso scrivere $X \rightarrow YZ$ "
 5. **Regola di pseudotransitività** (oppure **di aggiunta a sinistra**): "Se ho $X \rightarrow Y$ e $WY \rightarrow Z$, allora $XW \rightarrow Z$ "
L'aggiunta sulla transitività è W
 6. **Regola di decomposizione**: "Se ho Z sottoinsieme di Y, e $X \rightarrow Y$, allora $X \rightarrow Z$ "

TROVARE LA CHIAVE CON LE REGOLE DI ARMSTRONG

- Dato l'insieme F di FD composto così:
- i. Impiegato \rightarrow Stipendio
 - ii. Progetto \rightarrow Bilancio
 - iii. Impiegato Progetto \rightarrow Funzione
- 1) Uso la regola 2 sulle dipendenze i e ii, quindi ottengo
- i. Impiegato Progetto \rightarrow Stipendio Progetto
(dove Z sarebbe in questo caso Progetto)
 - ii. Progetto Impiegato \rightarrow Bilancio Impiegato
(dove Z sarebbe in questo caso Impiegato)
 - iii. Impiegato Progetto \rightarrow Funzione (rimane così com'è)
- 2) Uso la regola 4 ottenendo:
Impiegato Progetto \rightarrow Stipendio Progetto Impiegato Bilancio Funzione
(dove in questo caso come X consideriamo entrambi gli attributi Impiegato Progetto)
 \rightarrow Possiamo stabilire inoltre, che **Impiegato Progetto** sono chiave!

CALCOLO EQUIVALENZA CON LE REGOLE DI ARMSTRONG

"Date le seguenti DF:

- 1) $F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$
- 2) $G = \{A \rightarrow CD, E \rightarrow AH\}$

Verificare se F e G sono equivalenti"

\rightarrow Quindi va dimostrato che le DF in F sono derivabili dalle FD in G, e viceversa tramite le regole di Armstrong



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO CALCOLO EQUIVALENZA CON LE REGOLE DI ARMSTRONG

- Comincio col verificare che le FD in F sono derivabili da quelle in G

- 1) Prendo $A \rightarrow CD$ e ci applico la regola 6 di Armstrong, considerando prima C e poi D sottoinsieme di CD ed ottengo: $A \rightarrow CD \Rightarrow A \rightarrow C, A \rightarrow D$

OSSERVAZIONE: Per la regola di Armstrong ho considerato A come la X, CD come la Y e, separatamente come Z, prima considero la C e poi considero la D

- La FD $A \rightarrow C$ appartiene all'insieme F; quindi, è derivabile da G

- 2) Prendo $A \rightarrow CD$ e ci applico la regola 2 di Armstrong, aggiungendo C a SX della freccia e ottengo: $A \rightarrow CD \Rightarrow AC \rightarrow CD$

OSSERVAZIONE: Per la regola di Armstrong ho considerato A come X, CD come Y e C come Z, il quale è sottoinsieme di Y (cioè di CD)

Applico ora la regola 6 di Armstrong, considerando prima C e poi D come sottoinsieme di CD ed ottengo: $AC \rightarrow CD \Rightarrow AC \rightarrow C, AC \rightarrow D$

OSSERVAZIONE: Per la regola di Armstrong ho considerato AC come la X, CD come la Y e, separatamente come Z, prima considero la C e poi considero la D

- La FD $AC \rightarrow D$ appartiene all'insieme F; quindi, è derivabile da G

- 3) Prendo $E \rightarrow AH$ e ci applico la regola 6 di Armstrong, considerando prima A e poi H come sottoinsieme di AH ed ottengo: $E \rightarrow AH \Rightarrow E \rightarrow A, E \rightarrow H$

OSSERVAZIONE: Per la regola di Armstrong ho considerato E come la X, AH come la Y e, separatamente come Z, prima considero la A e poi considero la H

- La FD $E \rightarrow H$ appartiene all'insieme F; quindi, è derivabile da G

- 4) Prendo $E \rightarrow AH$ e ci applico la regola 6 di Armstrong, considerando prima A e poi H come sottoinsieme di AH ed ottengo: $E \rightarrow AH \Rightarrow E \rightarrow A, E \rightarrow H$

OSSERVAZIONE: Per la regola di Armstrong ho considerato E come la X, AH come la Y e, separatamente come Z, prima considero la A e poi considero la H

Considero ora $E \rightarrow A$, ci applico la regola 3 di Armstrong, considerando $A \rightarrow D$ ottengo: $E \rightarrow A, A \rightarrow D \Rightarrow E \rightarrow D$

OSSERVAZIONE: Per la regola di Armstrong ho considerato E come la X, A come la Y e, D come Z

Considero ora $E \rightarrow A, E \rightarrow D$ e ci applico la regola 4 di Armstrong ottenendo:

$E \rightarrow A, E \rightarrow D \Rightarrow E \rightarrow AD$

OSSERVAZIONE: Per la regola di Armstrong ho considerato E come la X, A come la Y e, D come Z

- La FD $E \rightarrow AD$ appartiene all'insieme F; quindi, è derivabile da G ed inoltre, ho verificato che tutte le dipendenze funzionali in F sono derivabili da G



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO CALCOLO EQUIVALENZA CON LE REGOLE DI ARMSTRONG

- Adesso verifico che le FD in G sono derivabili da quelle in F

- 1) Prendo $A \rightarrow C$ e ci applico la regola 3 di Armstrong, considerando $C \rightarrow D$ ottengo:

$$A \rightarrow C, C \rightarrow D \Rightarrow A \rightarrow D$$

OSSERVAZIONE: Per la regola di Armstrong ho considerato A come la X, C come la Y e, D come Z

Considero ora $A \rightarrow C$, $A \rightarrow D$ e ci applico la regola 4 di Armstrong ottenendo:

$$A \rightarrow C, A \rightarrow D \Rightarrow A \rightarrow CD$$

OSSERVAZIONE: Per la regola di Armstrong ho considerato A come la X, C come la Y e, D come Z

- La FD $A \rightarrow CD$ appartiene all'insieme G, quindi è derivabile da F

- 2) Prendo $E \rightarrow AD$ e ci applico la regola 6 di Armstrong, considerando prima A e poi D come sottoinsieme di AD ed ottengo: $E \rightarrow AD \Rightarrow E \rightarrow A, E \rightarrow D$

OSSERVAZIONE: Per la regola di Armstrong ho considerato E come la X, AD come la Y e, separatamente come Z, prima considero la A e poi considero la D

Prendo dall'insieme di dipendenze $E \rightarrow H$, ed $E \rightarrow A$ appena trovato ed applico la regola 4 di Armstrong ottenendo: $E \rightarrow A, E \rightarrow H \Rightarrow E \rightarrow AH$

- La FD $E \rightarrow AH$ appartiene all'insieme G; quindi, è derivabile da F ed inoltre, ho verificato che tutte le dipendenze funzionali in G sono derivabili da F

→ F e G sono equivalenti!

CHIUSURA SULL'INSIEME DI ATTRIBUTI DI UNA RELAZIONE: X^+

PASSAGGI:

- 1) Considero X, il primo attributo dell'insieme di dipendenze dato in ingresso
- 2) Inizialmente faccio rappresentare il valore X come un insieme, e lo chiamo *chiusura di X* → $X^+ = X$
- 3) Ciclicamente, per ogni dipendenza della relazione:
 - a. Prendo la dipendenza
 - b. Controllo se il valore che ha alla sua SX sta nell'insieme di chiusura di X
 - c. Se è presente, di questa dipendenza ne prendo la parte DX e la metto nell'insieme di chiusura di X
Altrimenti salto il valore della dipendenza e vedo se lo trovo più tardi
OSSERVAZIONE: In X^+ ci metto solamente la parte di DX (considerata al punto c), che manca, senza ripetere gli attributi
 - d. Mi fermo solamente quando nella chiusura di X ci sono già **tutte** le parti DX delle dipendenze della relazione che mi interessa mettere!

ESEMPIO

“Dato il seguente insieme di DF:

$F = \{A \rightarrow B, BC \rightarrow D, B \rightarrow E, E \rightarrow C\}$

Calcoliamo A^+ ”

→ Ovvero l'insieme degli attributi che dipendono da A



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO CHIUSURA SULL'INSIEME DI ATTRIBUTI DI UNA RELAZIONE: X^+

PASSAGGI:

- 1) Considero A come primo attributo
- 2) Rappresento A come un insieme cioè la chiusura di A $\rightarrow A^+=A$
- 3) Ciclo:
 - a. Prendo $A \rightarrow B$
 - b. Il valore a SX di questa dipendenza appartiene ad A^+
 - c. Metto la parte DX della dipendenza, cioè B in A^+ , quindi $A^+=AB$

 - a. Prendo $BC \rightarrow D$
 - b. Il valore C che sta a SX di questa dipendenza non appartiene ad A^+ lo salto, finché C, non apparirà in A^+

 - a. Prendo $B \rightarrow E$
 - b. Il valore a SX di questa dipendenza appartiene ad A^+
 - c. Metto la parte DX della dipendenza, cioè E in A^+ , quindi $A^+=ABE$

 - a. Prendo $E \rightarrow C$
 - b. Il valore a SX di questa dipendenza appartiene ad A^+
 - c. Metto la parte DX della dipendenza, cioè C in A^+ , quindi $A^+=ABEC$

 - a. Riprendo $BC \rightarrow D$
 - b. Il valore a SX di questa dipendenza adesso appartiene ad A^+
 - c. Metto la parte DX della dipendenza, cioè D in A^+ , quindi $A^+=ABECD$
 - d. Mi fermo perché ho visto tutte le dipendenze, dicendo che A è superchiave, oltre ad essere chiave, perché da essa dipendono tutti gli attributi

EQUIVALENZA CON LA CHIUSURA SULL'INSIEME DI ATTRIBUTI

PASSAGGI:

- Date due insiemi di dipendenze in ingresso G e F, devo verificare se tutte le dipendenze in F dipendono dagli attributi di G e viceversa:

- 1) Se $X \rightarrow Y$ vale in F, devo verificare che valga nella chiusura sull'insieme di attributi $(X)^{+G} \rightarrow$ Se vale allora Y è contenuto nella chiusura $(X)^{+G}$
- 2) Se $W \rightarrow Z$ vale in G, devo verificare che valga nella chiusura sull'insieme di attributi $(W)^{+F} \rightarrow$ Se vale allora Z è contenuto nella chiusura $(W)^{+F}$

ESEMPIO

"Dati i seguenti insiemi di DF:

- $F=\{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$
- $G=\{A \rightarrow CD, E \rightarrow AH\}$

Devo verificare se sono equivalenti"



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO EQUIVALENZA CON LA CHIUSURA SULL'INSIEME DI ATTRIBUTI

- Per l'equivalenza devo verificare se:

- (1) Gli attributi dell'insieme F sono inclusi nella chiusura rispetto all'insieme G
- (2) Gli attributi dell'insieme G sono inclusi nella chiusura rispetto all'insieme F

(1)

- Considero $A \rightarrow C$ e ne considero la chiusura di A rispetto a G
- Risulta quindi $(A)^{+G} = ACD$
- Di conseguenza C appartiene a $(A)^{+G}$

-
- Considero $AC \rightarrow D$ e ne considero la chiusura di AC rispetto a G
 - Risulta quindi $(AC)^{+G} = ACD$
 - Di conseguenza D appartiene a $(AC)^{+G}$

-
- Considero $E \rightarrow AD$ e ne considero la chiusura di E rispetto a G
 - Risulta quindi $(E)^{+G} = EACDH$
 - Di conseguenza AD appartiene a $(E)^{+G}$

-
- Considero $E \rightarrow H$ e ne considero la chiusura di E rispetto a G
 - Risulta quindi $(E)^{+G} = EACDH$
 - Di conseguenza H appartiene a $(E)^{+G}$
 - Posso concludere che gli attributi dell'insieme F sono inclusi nella chiusura rispetto all'insieme G

(2)

-
- Considero $A \rightarrow CD$ e ne considero la chiusura di A rispetto a F
 - Risulta quindi $(A)^{+F} = ACD$
 - Di conseguenza CD appartiene a $(A)^{+F}$

-
- Considero $E \rightarrow AH$ e ne considero la chiusura di E rispetto a F
Risulta quindi $(E)^{+F} = EACDH$
 - Di conseguenza AH appartiene a $(E)^{+F}$
 - Posso concludere che anche gli attributi dell'insieme G sono inclusi nella chiusura rispetto all'insieme F
 - Gli insiemi di FD sono EQUIVALENTI!

RIDONDANZE

- Un insieme di FD può contenere **dipendenze ridondanti** ottenibili da altre dipendenze
- **Ridondanza a DX** \rightarrow Possiamo avere una FD che ha la parte DX ottenibile da altre FD dell'insieme
- **Ridondanza a SX** \rightarrow Possiamo avere una FD che ha la parte SX ottenibile da altre FD dell'insieme



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO RIDONDANZE

ESEMPIO

“Dato $F=\{A \rightarrow C, E \rightarrow AC, E \rightarrow H\}$ verifichiamo la ridondanza a DX della FD $E \rightarrow AC$ ”

- Consideriamo $E \rightarrow AC$ e la togliamo, spezzandola in più FD, diventando $F=\{A \rightarrow C, E \rightarrow A, E \rightarrow C, E \rightarrow H\}$
- A questo punto, notiamo che posso togliere $E \rightarrow C$, perché è *ridondante*, cioè è ottenibile passando dalle FD: $E \rightarrow A$ e poi $A \rightarrow C$
- Ottengo quindi $F=\{A \rightarrow C, E \rightarrow A, E \rightarrow H\}$

ESEMPIO

“Dato $F=\{A \rightarrow B, B \rightarrow C, AC \rightarrow D\}$ verifichiamo la ridondanza a SX della FD $AC \rightarrow D$ ”

- Consideriamo $AC \rightarrow D$ e la togliamo, spezzandola in più FD, diventando $F=\{A \rightarrow B, B \rightarrow C, A \rightarrow D, A \rightarrow C\}$
- A questo punto, notiamo che posso togliere $A \rightarrow C$, perché è *ridondante*, cioè è ottenibile passando dalle FD: $A \rightarrow B$ e poi $B \rightarrow C$
- Ottengo quindi $F=\{A \rightarrow B, B \rightarrow C, A \rightarrow D\}$

OSSERVAZIONE: Quando tolgo una ridondanza, considero la dipendenza che fa il percorso più corto

DIPENDENZA FUNZIONALE SEMPLICE (STANDARD)

- Per portare un insieme di FD in forma **standard** → Dobbiamo avere a DX un unico attributo

ESEMPIO

“Dato $F=\{AC \rightarrow BD, AB \rightarrow DE\}$ portarla in forma standard”

→ Ottengo: $F=\{AC \rightarrow B, AC \rightarrow D, AB \rightarrow D, AB \rightarrow E\}$

ATTRIBUTI ESTRANEI

- Gli **attributi estranei** sono degli attributi ‘inutili’ che si trovano sul lato SX delle FD ed eseguo i seguenti passaggi:

- Per ogni attributo presente sul lato SX, per verificare che esso sia estraneo → Si fa la chiusura rispetto ad ogni attributo considerato.
Data $XY \rightarrow W$ si fa X^+ e Y^+
- Se per ogni attributo, facendoci la chiusura sopra, trovo che l’attributo a W è incluso nella chiusura → Allora l’attributo considerato **non** è estraneo e si tiene
- Se per ogni attributo, facendo la chiusura non trovo che l’attributo W è incluso nella chiusura → Allora l’attributo considerato è estraneo e si toglie

ESEMPIO

“Dato $F=\{AB \rightarrow C, A \rightarrow B\}$ togliere gli attributi estranei”

- Calcolo le chiusure di A^+ e B^+



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO ATTRIBUTI ESTRANEI

(Passaggi del calcolo chiusura per A^+)

- 1) Considero A come primo attributo
- 2) Rappresento A come un insieme cioè la chiusura di $A \rightarrow A^+ = A$
- 3) Ciclo:
 - a. Prendo $AB \rightarrow C$
 - b. Il valore B che sta a SX di questa dipendenza non appartiene ad A^+ lo salto, finché B, non apparirà in A^+

 - a. Prendo $A \rightarrow B$
 - b. Il valore a SX di questa dipendenza appartiene ad A^+
 - c. Metto la parte DX della dipendenza, cioè B in A^+ , quindi $A^+ = AB$

 - a. Riprendo $AB \rightarrow C$
 - b. Il valore a SX di questa dipendenza adesso appartiene ad A^+
 - c. Metto la parte DX della dipendenza, cioè C in A^+ , quindi $A^+ = ABC$ e mi fermo perché ho visto tutte le dipendenze

(Passaggi del calcolo chiusura per B^+)

- 1) Considero B come primo attributo
- 2) Rappresento B come un insieme cioè la chiusura di $B \rightarrow B^+ = B$
- 3) Non faccio nessun ciclo, perché si può notare che da B non ho nessuna dipendenza, e inoltre, a sua volta esso dipende da A

→ Posso concludere che in questo caso l'attributo B è **estraneo** e lo tolgo

→ Quindi, l'insieme di dipendenze diventa: $F = \{A \rightarrow C, A \rightarrow B\}$

ALGORITMO DI MINIMIZZAZIONE

- 1) Considero l'insieme F di FD e porto tutte le dipendenze ad avere **un solo** valore alla parte DESTRA (forma *standard*)
- 2) Poi porto le dipendenze in forma *canonica*, togliendo a SX gli attributi estranei
OSSERVAZIONE: Data la dipendenza $AB \rightarrow X$ si possono verificare due casi, scorrendo i successivi attributi:
 - a. $A \rightarrow B$, cioè B dipende da A e si toglie dalla dipendenza iniziale la lettera B, quindi ottengo $A \rightarrow X, A \rightarrow B, \dots$
 - b. $B \rightarrow A$, cioè A dipende da B e si toglie dalla dipendenza iniziale la lettera A, quindi ottengo $B \rightarrow X, B \rightarrow A, \dots$
- 3) Controllo se ci sono dipendenze RIDONDANTI, per poi toglierle eventualmente (potrei usare anche il fatto di verificare se per ogni dipendenza dell'insieme F, la parte DX di ciascuna dipendenza appartiene alla chiusura della parte di SX)
- 4) Se necessario riunisco alcune dipendenze
→ Ciò dipende a seconda di come sono poste le risposte dei testi degli esercizi perché potrei avere
 - a. Caso in cui le dipendenze dopo esser state spezzate si riuniscano
 - b. Caso in cui le dipendenze rimangano divise

VAGLINI-PREPARAZIONE ESERCIZI

ESERCIZIO MINIMIZZAZIONE

ESEMPIO

“Dato l'insieme di dipendenze funzionali: $F=\{AB \rightarrow C, B \rightarrow A, C \rightarrow E, AB \rightarrow E\}$ trovare la copertura minimale”

Applicazione algoritmo:

- 1) **$AB \rightarrow C, B \rightarrow A, C \rightarrow E, AB \rightarrow E$** , rimane così com'è perché a DX ho solo un elemento
- 2) Posso notare che A dipende da B su due dipendenze, quindi, tolgo le A dalle dipendenze nelle parti SX, ed ottengo: **$B \rightarrow C, B \rightarrow A, C \rightarrow E, B \rightarrow E$**
- 3) Partendo da B per arrivare ad E, la strada più corta si fa tramite la dipendenza $B \rightarrow E$ che è una ridondanza, quindi la tolgo ed ottengo **$B \rightarrow C, B \rightarrow A, C \rightarrow E$**
- 4) Poiché l'esercizio ha le soluzioni con le dipendenze divise lo lascio così com'è, e quindi, la dipendenza equivalente minima ottenuta è **$B \rightarrow C, B \rightarrow A, C \rightarrow E$**

ESEMPIO

“Dato l'insieme di dipendenze funzionali: $F=\{A \rightarrow BCD, CD \rightarrow E, B \rightarrow D, A \rightarrow E\}$ trovare la copertura minimale”

Applicazione algoritmo:

- 1) **$A \rightarrow B, A \rightarrow C, A \rightarrow D, CD \rightarrow E, B \rightarrow D, A \rightarrow E$**
- 2) Nell'unica dipendenza che ha 2 attributi alla sua sinistra non ho dipendenze che dipendano da altre; quindi, lascio così com'è al punto 1) e cioè:
 $A \rightarrow B, A \rightarrow C, A \rightarrow D, CD \rightarrow E, B \rightarrow D, A \rightarrow E$
- 3) Mi accorgo che ci sono due strade partendo da A per arrivare ad E, quindi tolgo quella più corta $A \rightarrow E$, che è una ridondanza. Mi accorgo inoltre, che ci sono anche due strade partendo da A per arrivare ad D, quindi tolgo quella più corta $A \rightarrow D$, che è una ridondanza
Ottengo quindi l'insieme di dipendenze così: **$A \rightarrow B, A \rightarrow C, CD \rightarrow E, B \rightarrow D$**
- 4) Poiché l'esercizio ha le soluzioni con le dipendenze unite, unisco $A \rightarrow B$, e $A \rightarrow C$ ed ottengo la dipendenza equivalente minima: **$A \rightarrow BC, CD \rightarrow E, B \rightarrow D$**

CHIAVE DI UN INSIEME DI DIPENDENZE (CON CHIUSURA)

- Dato un insieme F di FD

- 1) Controllo gli attributi delle parti SX che non compaiono a DX di ciascuna dipendenza, perché sono quelli che non dipendono da nessuno (o insieme o separati)
- 2) Prendo gli attributi considerati al punto precedente e ne faccio chiusura (o insieme o separati)
- 3) Posso avere due casi per la chiave
 - a) Gli attributi della chiusura, tutti insieme, coprono tutte le dipendenze
→ Allora quegli attributi rappresentano una *superchiave*
 - b) Gli attributi della chiusura, tutti insieme non coprono tutte le dipendenze, ma lo fanno separatamente → Allora quegli attributi rappresentano una *chiave*
 - c) Si può presentare un caso che gli attributi producono, tramite delle determinate combinazioni, più chiavi



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO CHIAVE DI UN INSIEME DI DIPENDENZE (CON CHIUSURA)

CASO SUPERCHIAVE

“Dato l'insieme di FD $F=\{A \rightarrow B, BC \rightarrow HD, ED \rightarrow H, AB \rightarrow G\}$ trovare la chiave”

Applico i passaggi:

1) In questo caso ottengo A, C, E (insieme)

2) Faccio la chiusura ACE^+

a) **$ACE^+=ACE$** //inizialmente

b) Prendo la dipendenza $A \rightarrow B$ e verifico se A appartiene ad $ACE^+ \rightarrow$ A appartiene alla chiusura \rightarrow Metto il valore a DX, cioè B nella chiusura \rightarrow **$ACE^+=ACEB$**

c) Prendo la dipendenza $BC \rightarrow HD$ e verifico se BC appartiene ad $ACE^+ \rightarrow$ BC appartiene alla chiusura \rightarrow Metto il valore a DX, cioè HD nella chiusura \rightarrow **$ACE^+=ACEBHD$**

d) Prendo la dipendenza $ED \rightarrow H$ e verifico se ED appartiene ad $ACE^+ \rightarrow$ ED appartiene alla chiusura \rightarrow Non metto il valore a DX, cioè H nella chiusura, perché è già presente

e) Prendo la dipendenza $AB \rightarrow G$ e verifico se AB appartiene ad $ACE^+ \rightarrow$ AB appartiene alla chiusura \rightarrow Metto il valore a DX, cioè G nella chiusura \rightarrow **$ACE^+=ACEBHDG$**

3.a) ACE è superchiave di F

CASO 1 - CHIAVE

“Dato l'insieme di FD $F=\{A \rightarrow B, A \rightarrow C, D \rightarrow E\}$ trovare la chiave”

Applico i passaggi:

1) In questo caso ottengo A, D (separati)

2) Non posso fare la chiusura AD^+ ma la faccio separatamente prima con A^+ e poi con D^+

a) **$A^+=A$** //inizialmente

b) Prendo la dipendenza $A \rightarrow B$ e verifico se A appartiene ad $A^+ \rightarrow$ A appartiene alla chiusura \rightarrow Metto il valore a DX, cioè B nella chiusura \rightarrow **$A^+=AB$**

c) Prendo la dipendenza $A \rightarrow C$ e verifico se A appartiene ad $A^+ \rightarrow$ A appartiene alla chiusura \rightarrow Metto il valore a DX, cioè C nella chiusura \rightarrow **$A^+=ABC$**

d) Prendo la dipendenza $D \rightarrow E$ e verifico se D appartiene ad $A^+ \rightarrow$ D non appartiene alla chiusura \rightarrow Mi fermo e passo con la chiusura rispetto a D, indicando A come parte della chiave

a) **$D^+=D$** //inizialmente

b) Prendo la dipendenza $D \rightarrow E$ e verifico se D appartiene ad $D^+ \rightarrow$ D appartiene alla chiusura \rightarrow Metto il valore a DX, cioè E nella chiusura \rightarrow **$D^+=DE$** \rightarrow Mi fermo indicando D come parte della chiave

3.b) AD è chiave di F

CASO 2 - CHIAVE

“Dato l'insieme di FD $F=\{A \rightarrow B, C \rightarrow AD, AF \rightarrow E\}$ trovare la chiave”

Applico i passaggi:

1) In questo caso ottengo C, F (separati)

2) Non posso fare la chiusura CF^+ ma la faccio separatamente prima con C^+ e poi con F^+



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO CHIAVE DI UN INSIEME DI DIPENDENZE (CON CHIUSURA)

- a) $C^+ = C$ //inizialmente
- b) Prendo la dipendenza $A \rightarrow B$ e verifico se A appartiene ad $C^+ \rightarrow A$ non appartiene alla chiusura \rightarrow La salto e ci torno dopo
- c) Prendo la dipendenza $C \rightarrow AD$ e verifico se C appartiene ad $C^+ \rightarrow C$ appartiene alla chiusura \rightarrow Metto il valore a DX, cioè AD nella chiusura $\rightarrow C^+ = CAD$
- d) Prendo la dipendenza $AF \rightarrow E$ e verifico se AF appartiene ad $C^+ \rightarrow AF$ non appartiene alla chiusura, la salto e ritorno a considerare la dipendenza al punto a)
- e) Riprendo la dipendenza $A \rightarrow B$ e verifico se A appartiene ad $C^+ \rightarrow A$ appartiene alla chiusura \rightarrow Metto il valore a DX, cioè B nella chiusura $\rightarrow C^+ = CADB \rightarrow$ Mi fermo e passo con la chiusura rispetto a F , indicando C come parte della chiave

-
- a) $F^+ = F \rightarrow$ Mi accorgo che da sola non fa nulla quindi la devo legare alla parte della chiave trovata prima, cioè C , e alla parte con cui è legato nella FD , cioè A
 \rightarrow Devo fare quindi
 - i. CF^+
 - ii. AF^+

-
- i. $CF^+ = CADBFE \rightarrow$ Perché \rightarrow La chiusura di C mi porta a $CADB \rightarrow$ La chiusura di F mi dà $F \rightarrow$ Ma avendo A dentro la chiusura di $C \rightarrow$ Tramite AF arrivo ad EC , del quale mi prendo solo E visto che C è già dentro la chiusura stessa

-
- ii. $AF^+ = CADBFE \rightarrow$ Perché \rightarrow La chiusura di F mi dà $F \rightarrow F$ insieme ad A mi produce $EC \rightarrow$ Tramite C ottengo la $D \rightarrow A$ di per sé mi produce B

3.c) AF e CF sono chiavi di F

//Considerate separatamente

VERIFICA BCNF (CON DECOMPOSIZIONE)

- Data una relazione R , con le corrispondenti FD (+ la chiave) \rightarrow Per verificare che R sia BCNF si procede così:

1. Si prendono le varie FD e si controllano se **tutte** alla loro SX hanno una *superchiave*
2. Se ciò accade, allora la relazione è BCNF
3. Se ciò non accade, per avere la BCNF si procede con la decomposizione in tabelle della relazione, in cui ciascuna tabella è determinata dalla corrispondente FD
4. Dopo averle decomposte, si inserisce una tabella aggiuntiva che ha come attributi \rightarrow Tutte le parti SX delle FD ottenute al punto 2), in modo che formano una chiave, da poter garantire la non perdita sul JOIN



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO VERIFICA BCNF (CON DECOMPOSIZIONE)

CASO-AGGIUNTA DI TABELLA

“Data la seguente relazione Sito

Sito (Categoria, Codice, NumeroOggettiDisponibili, Prezzo, IndirizzoWebNegozio, Fornitore, IndirizzoFornitore, TipoPagamento, NazionalitàFornitore, PIVA)

e le corrispettive dipendenze (con chiave annessa)

IndirizzoWebNegozio → TipoPagamento
Categoria, Codice, Fornitore → Prezzo
Categoria, Codice → NumeroOggettiDisponibili
Fornitore → IndirizzoFornitore, NazionalitàFornitore, PIVA

K = Categoria, Codice, Fornitore, IndirizzoWebNegozio

verificare se è in forma BCNF”

Passaggi:

1. Posso già notare che alla loro SX tutte le FD non hanno una superchiave
2. Salto questo punto perché non è una BCNF
3. Decompongo in tabelle
4. Aggiungendo un'ulteriore relazione che contiene tutta la chiave k

Negozio (IndirizzoWebNegozio, Tipo Pagamento)
Oggetto (Categoria, Codice, Fornitore, Prezzo)
Magazzino (Categoria, Codice, NumeroOggettiDisponibili)
Fornitore (Fornitore, IndirizzoFornitore, NazionalitàFornitore, PIVA)
Localizzazione (Categoria, Codice, Fornitore, IndirizzoWebNegozio)

Tabella
aggiuntiva

CASO-NON AGGIUNTA DI TABELLA

“Data la seguente relazione Sito

Sito (Categoria, Codice, NumeroOggettiDisponibili, Prezzo, IndirizzoWebNegozio, Fornitore, IndirizzoFornitore, TipoPagamento, NazionalitàFornitore, PIVA)

e le corrispettive dipendenze (con chiave annessa)

Categoria, Codice → IndirizzoWebNegozio, TipoPagamento
Categoria, Codice, Fornitore → Prezzo, NumeroOggettiDisponibili
Fornitore → IndirizzoFornitore, NazionalitàFornitore, PIVA

K = Categoria, Codice, Fornitore

verificare se è in forma BCNF”



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO VERIFICA BCNF (CON DECOMPOSIZIONE)

Passaggi:

1. Posso già notare che alla loro SX tutte le FD non hanno una superchiave, ma la seconda sì
2. Salto questo punto perché non è una BCNF
3. Decompongo in tabelle
4. Non aggiungo una ulteriore tabella che mi contenga la chiave, perché la seconda dipendenza alla sua SX ha tutta la chiave k

Assortimento (Categoria, Codice, IndirizzoWebNegozio , Tipo Pagamento) Oggetto (Categoria, Codice, Fornitore , Prezzo, NumeroOggettiDisponibili) Fornitore (Fornitore, IndirizzoFornitore, NazionalitàFornitore, PIVA)
--

CASO-ATTRIBUTO NON PRESENTE NELLE FD

“Data la seguente relazione *RivisteScientifiche*

RivisteScientifiche (TitoloRivista, Direttore, Editore, Numero, Anno, NumeroPagineRivista, NumeroCopieRivista, TitoloArticolo, AutoreArticolo, NumeroPagineArticolo, ArgomentoArticolo)

e le corrispettive dipendenze (con chiave annessa)

TitoloRivista → Direttore, Editore TitoloRivista, Numero, Anno → NumeroPagineRiviste, NumeroCopieRivista TitoloArticolo → NumeroPagineArticolo, ArgomentoArticolo K= <u>TitoloArticolo, TitoloRivista, Numero, Anno, AutoreArticolo</u>
--

verificare se è in forma BCNF”

Passaggi:

1. Posso già notare che alla loro SX tutte le FD non hanno una superchiave e inoltre, l'attributo *AutoreArticolo* della relazione non è presente in nessuna delle FD, di conseguenza dovrà far parte della chiave
2. Salto questo punto perché non è una BCNF
3. Decompongo in tabelle
4. Aggiungendo un'ulteriore relazione che contiene tutta la chiave k, compreso l'attributo mancante

Riviste (<u>TitoloRivista</u> , Direttore, Editore) NumeroRiviste (<u>TitoloRivista, Numero, Anno</u> , NumeroPagineRiviste, NumeroCopieRivista) Articoli (<u>TitoloArticolo</u> , NumeroPagineArticolo, ArgomentoArticolo) Pubblicazioni (<u>TitoloArticolo, TitoloRivista, Numero, Anno, AutoreArticolo</u>)
--

Tabella
aggiuntiva



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO VERIFICA BCNF (CON DECOMPOSIZIONE)

CASO-UNIONE DI FD NELLA DECOMPOSIZIONE

“Data la seguente relazione *DEGENTE*

DEGENTE (Cofice-Fiscale, Nome-Degente, Cognome-Degente, Data, Reparto, Capo-Reparto, Stanza, Infermiere-di-Stanza)

e le corrispettive dipendenze (con chiave data da $K=\text{CodiceFiscale, Data}$)

Codice-Fiscale -> Nome-degente, Cognome-degente
Reparto -> Capo-Reparto
Stanza -> Reparto
Stanza -> Infermiere-di-stanza
Codice-Fiscale, Data -> Stanza

verificare se è in forma BCNF”

Passaggi:

1. Posso già notare che alla loro SX tutte le FD non hanno una superchiave
2. Salto questo punto perché non è una BCNF
3. Decompongo in tabelle, unendo la 3° e 4° FD, dato che hanno a SX lo stesso attributo, cioè *Stanza*
4. Non aggiungo una ulteriore tabella che mi contenga la chiave, perché l'ultima dipendenza alla sua SX ha tutta la chiave k

DEGENTE (Codice-Fiscale, Nome-degente, Cognome-degente)
REPARTO (Reparto, Capo-Reparto)
STANZA (Stanza, Reparto, Infermiere-di-Stanza)
DEGENZA (Codice-Fiscale, Data, Stanza)

RECORD DI UNA TRANSAZIONE (FILE DI LOG)

- Il recordo del LOG è composto da:

- Record di transazione, in cui si trovano i seguenti comandi:

B(T)	begin
I(T, O, AS)	insert
D(T, O, BS)	delete
U(T, O, BS, AS)	update
C(T)	commit
A(T)	abort

→dove indichiamo con:

- T → Il nome dell'oggetto su cui si lavora (variabile/transazione)
 - O → Il valore dell'oggetto
 - AS → “AFTER STATE”, cioè lo stato successivo all'operazione su un oggetto
 - BS → “BEFORE STATE”, cioè lo stato precedente all'operazione su un oggetto
- Record di sistema:

DUMP	dump
CK(T1, ... ,Tn)	checkpoint

VAGLINI-PREPARAZIONE ESERCIZI

UNDO E REDO

- Per recuperare lo stato di una transazione (oppure un oggetto O) si fanno le operazioni di

UNDO	"Disfare"
REDO	"Rifare"

- Se uso UNDO su una transazione:

- update / delete → Devo recuperarne lo stato che aveva prima di UNDO, cioè il BS
- insert → Devo cancellare la transazione

- Se uso REDO su una transazione:

- insert / update → Devo recuperare lo stato che aveva dopo di REDO, cioè AS
- delete → Devo cancellare la transazione

RIPRESA A CALDO

- Passaggi:

1. Parto dal **GUASTO** e vado a ritroso fino al primo checkpoint che trovo (**CK**)
2. Considero le transazioni attive **T1...Tn** in CK, che non hanno fatto commit/abort
3. Incomincio a costruire gli insiemi di **UNDO**, **REDO**, guardando **B(T)**, **A(T)** e **C(T)**:
 - a) In UNDO devo mettere le transazioni T1...Tn trovate al momento del CK
4. Vado in avanti fino al GUASTO e controllo in quale dei seguenti casi mi trovo:
 - a) Transazioni che iniziano e non fanno commit → Finiscono nell'insieme UNDO
 - b) Transazioni che erano in UNDO e non hanno fatto commit → Rimangono nell'insieme UNDO
 - c) Transazioni che erano in UNDO e hanno fatto commit → Finiscono in REDO
 - d) Transazioni dell'insieme UNDO che fanno abort → Rimangono in UNDO, visto che vanno disfatte
5. Considero le transazioni da disfare dell'insieme UNDO e ne riempio la tabella così:
 - a) Parto dal fondo ed eseguo, andando all'indietro, le loro azioni
6. Considero le transazioni da rifare dell'insieme REDO e ne riempio la tabella così:
 - a) Parto dall'inizio ed eseguo, andando avanti, le loro azioni

CASO AZIONI SVOLTE CON UNDO

- a) Insert → I(T, O, AS) → **delete O** → Disfare un inserimento comporta la cancellazione dell'oggetto
- b) Update → U(T, O, BS, AS) → **O = BS** → Disfare un aggiornamento comporta che l'oggetto prenda il suo before state
- c) Delete → D(T, O, BS) → **insert O=BS** → Disfare un inserimento comporta l'inserimento dell'oggetto

CASO AZIONI SVOLTE CON REDO

- a) Insert → I(T, O, AS) → **insert O = AS** → Rifare un inserimento comporta che l'oggetto venga inserito con il suo after state
- b) Update → U(T, O, BS, AS) → **O = AS** → Rifare un aggiornamento comporta che l'oggetto prenda il suo after state
- c) Delete → D(T, O, BS) → **delete O** → Rifare una cancellazione, comporta la cancellazione dell'oggetto

VAGLINI-PREPARAZIONE ESERCIZI

APPLICAZIONE RIPRESA A CALDO

“Considerando il seguente LOG applicare la ripresa a caldo”

B(T1), B(T2), I(T1, O1, A1), D(T2, O2, B2), B(T3),
B(T4), U(T3, O3, B3, A3), C(T2), CK(T1,T3,T4),
U(T1, O4, B4, A4), A(T3), B(T5), D(T4, O5, B5),
C(T1), C(T4), I(T5, O6, A6), GUASTO

- 1) Parto dal GUASTO e arrivo fino a → **CK(T1,T3,T4)**
- 2) Le transazioni attive sono → **T1, T3, T4**
- 3) Costruisco gli insiemi
 - **UNDO={T1, T3, T4}**
 - **REDO={ }**
- 4) Inizio la percorrenza in avanti dal CK fino al GUASTO:
 - **B(T5)** → Caso 4.a) di pagina precedente, gli insiemi diventano:
 - **UNDO={T1, T3, T4, T5}**
 - **REDO={ }**
 - **C(T1)** → Caso 4.c) di pagina precedente, gli insiemi diventano:
 - **UNDO={T3, T4, T5}**
 - **REDO={T1}**
 - **C(T4)** → Caso 4.c) di pagina precedente, gli insiemi diventano:
 - **UNDO={T3, T5}**
 - **REDO={T1,T4}**
- 5) Considero l'insieme UNDO={T3,T5}, partendo dal fondo e andando all'indietro, svolgo le azioni relative alle transazioni:
 - **I(T5, O6, A6)** → CASO AZIONI SVOLTE CON UNDO.a):
→ **delete O6**
 - **U(T3, O3, B3, A3)** → CASO AZIONI SVOLTE CON UNDO.b):
→ **O3=B3**
- 6) Considero l'insieme REDO={T1,T4}, partendo dall'inizio e andando avanti, svolgo le azioni relative alle transazioni:
 - **I(T1, O1, A1)** → CASO AZIONI SVOLTE CON REDO.a):
→ **insert O1=A1**
 - **U(T1, O4, B4, A4)** → CASO AZIONI SVOLTE CON REDO.b):
→ **O4=A4**
 - **D(T4, O5, B5)** → CASO AZIONI SVOLTE CON REDO.c):
→ **delete O5**

VAGLINI-PREPARAZIONE ESERCIZI

RIPRESA A FREDDO

- La ripresa a freddo si applica sulle azioni che riguardano gli oggetti **O1...On**, dati in input, con i seguenti passaggi:

1. Parto dal **GUASTO** e vado all'indietro fino al **DUMP** più recente
2. Mi sposto in avanti eseguendo le *azioni* della parte deteriorata degli O1...On oggetti, fino a che arrivo al GUASTO
3. Si esegue una RIPRESA A CALDO

CASO AZIONI SVOLTE CON RIPRESA A FREDDO

- a) Insert $\rightarrow I(T, O, AS) \rightarrow \text{insert } O = AS \rightarrow$ L'oggetto deve essere inserito con il suo after state
- b) Update $\rightarrow U(T, O, BS, AS) \rightarrow O = AS \rightarrow$ L'oggetto deve essere aggiornato con il suo after state
- c) Delete $\rightarrow D(T, O, B) \rightarrow \text{delete } O \rightarrow$ Cancellazione dell'oggetto
- d) Commit $\rightarrow C(T) \rightarrow \text{commit}(T)$
- e) Abort $\rightarrow A(T) \rightarrow \text{abort}(T)$

APPLICAZIONE RIPRESA A FREDDO

“Considerando il seguente LOG applicare la ripresa a freddo sugli oggetti O1, O2, O3”

DUMP, B(T1), B(T2), I(T1, O1, A1), D(T2, O2, B2), B(T3),
B(T4), U(T3, O3, B3, A3), C(T2), CK(T1,T3,T4),
U(T1, O4, B4, A4), A(T3), B(T5), D(T4, O5, B5),
C(T1), C(T4), I(T5, O6, A6), GUASTO

- 1) Parto dal GUASTO e arrivo fino al **DUMP** più recente (che sarebbe l'unico che è presente)
- 2) Spostandomi in avanti, fino al guasto, considero le azioni degli oggetti O1, O2, O3
 - $I(T1, O1, A1) \rightarrow$ CASO AZIONI SVOLTE CON RIPRESA A FREDDO.a)
 $\rightarrow \text{insert } O1=A1$
 - $D(T2, O2, B2) \rightarrow$ CASO AZIONI SVOLTE CON RIPRESA A FREDDO.c)
 $\rightarrow \text{delete } O2$
 - $U(T3, O3, B3, A3) \rightarrow$ CASO AZIONI SVOLTE CON RIPRESA A FREDDO.b)
 $\rightarrow O3=A3$
 - $C(T2) \rightarrow$ CASO AZIONI SVOLTE CON RIPRESA A FREDDO.d)
 $\rightarrow \text{commit}(T2)$
 - $A(T3) \rightarrow$ CASO AZIONI SVOLTE CON RIPRESA A FREDDO.e)
 $\rightarrow \text{abort}(T3)$
 - $C(T1) \rightarrow$ CASO AZIONI SVOLTE CON RIPRESA A FREDDO.d)
 $\rightarrow \text{commit}(T1)$
- 3) RIPRESA A CALDO (che è la stessa dell'esempio di pagina precedente)

VAGLINI-PREPARAZIONE ESERCIZI

VIEW-EQUIVALENZA E VIEW-SERIALIZZABILITÀ TRA DUE SCHEDULE

- Per la **view-equivalenza** tra due schedule **S1** e **S2** verifico se hanno entrambi:
 - La stessa legge-da → Cioè se entrambi hanno le stesse letture sullo stesso oggetto, le quali letture sono precedute dalle stesse scritture (sempre sullo stesso oggetto) indipendentemente dall'ordine in cui vengono fatte!
 - La stessa scrittura finale sullo stesso oggetto
- Per la **view-serializzabilità** verifico in ordine:
 1. Se ho un determinato schedule **X** che è seriale
 2. Se ho uno schedule **S** che è view-equivalente ad **X**

APPLICAZIONE VIEW-EQUIVALENZA E VIEW-SERIALIZZABILITÀ TRA DUE SCHEDULE

“Considerati i seguenti schedule, dire quali sono view-equivalenti, e view-serializzabili”

S3: w0(x)r2(x)r1(x)w2(x)w2(z)
S4: w0(x)r1(x)r2(x)w2(x)w2(z)
S5: w0(x)r2(x)w2(x)r1(x)w2(z)
S6: w0(x)r2(x)w2(x)w2(z)r1(x)

CASO VIEW-EQUIVALENZA

- 1) Considero S3 con S4:
 - Con S4 ha la stessa “legge-da”, perché hanno entrambi r1 e r2 che leggono su x dopo la scrittura di w0 → Anche se entrambi leggono in ordine differente va bene lo stesso!
 - Inoltre, sia S3 che S4, hanno entrambi le stesse “scrittura finali” di w2 su z e w2 su x
→ **Sono view-equivalenti**
- 2) Considero S3 con S5:
 - Con S5 non ho la stessa “legge-da”, perché in S5 ho che r1 legge ciò che ha scritto w2, mentre in S3 ho che r1 legge ciò che ha scritto w0, allora posso già affermare senza andare a vedere la “scrittura finale” che:
→ **Non sono view-equivalenti**
- 3) Considero S3 con S6:
 - Con S6 non ho la stessa “legge-da”, perché in S6 ho che r1 legge ciò che ha scritto w2, mentre in S3 ho che r1 legge ciò che ha scritto w0, allora posso già affermare senza andare a vedere la “scrittura finale” che:
→ **Non sono view-equivalenti**
- 4) Considero S4 con S5:
 - Con S5 non ho la stessa “legge-da”, perché in S5 ho che r1 legge ciò che ha scritto w2, mentre in S4 ho che r1 legge ciò che ha scritto w0, allora posso già affermare senza andare a vedere la “scrittura finale” che:
→ **Non sono view-equivalenti**



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO APPLICAZIONE VIEW-EQUIVALENZA E VIEW-SERIALIZZABILITÀ TRA DUE SCHEDULE

5) Considero S4 con S6:

- Con S6 non ho la stessa “legge-da”, perché in S6 ho che r1 legge ciò che ha scritto w2, mentre in S4 ho che r1 legge ciò che ha scritto w0, allora posso già affermare senza andare a vedere la “scrittura finale” che:

→ **Non sono view-equivalenti**

6) Considero S5 con S6:

- Con S6 ha la stessa “legge-da”, perché hanno entrambi r2 che legge su x dopo la scrittura di w0, e r1 che legge su x dopo la scrittura di w2
- Inoltre, sia S5 che S6, hanno entrambi la stessa “scrittura finale” di w2 su z

→ **Sono view-equivalenti**

CASO VIEW-SERIALIZZABILITÀ

1. Dato che per valere la view-serializzabilità deve valere la view-equivalenza ad uno schedule seriale → Dimostro che, come schedule seriali, ho solamente S4 ed S6:

a) S4 è formata dalle transazioni t0,t1,t2,t2,t2 → Delle quali nessuna viene interrotta
→ Quindi, **S4 è seriale**

b) S6 è formata dalle transazioni → t0,t2,t2,t2,t1 → Delle quali nessuna viene interrotta
→ Quindi, **S6 è seriale**

OSSERVAZIONE: Le ripetizioni delle transazioni come in 1.a) e in 1.b) su t2 potrei anche evitarle e metterne una sola di t2

2. Escludo, la non-equivalenza tra i vari schedule, e considerando gli schedule seriali S4 e S6 → Dai punti precedenti tolgo i confronti degli schedule dei punti 2), 3), 4) e 5):

- Tra S3 e S4 → Essendo S4 seriale, ed S3 view-equivalente ad S4
→ Allora **S3 è view-serializzabile**
- Tra S5 e S6 → Essendo S6 seriale, ed S5 view-equivalente ad S6
→ Allora **S5 è view-serializzabile**

CONFLICT-SERIALIZZABILITÀ E CONFLICT-EQUIVALENZA DI UNO SCHEDULE

1. Si considera uno schedule **S** e si costruisce il grafo dei conflitti così:

- a) Si inserisce ogni nodo per ogni transazione **t_i**
- b) Si crea una lista di operazioni (da parte delle relative transazioni **t_i**) per ogni oggetto, secondo l'ordine in cui sono presenti nello schedule
- c) Si considerano, tutte le possibili coppie tra le operazioni delle transazioni su ogni oggetto (considerando l'ordine di ciascuna operazione con le successive):

- Se sono in conflitto (rw/wr/ww) → Si fa un arco orientato da un nodo del grafo che riguarda l'operazione della 1° transazione in conflitto → Ad un altro nodo del grafo, che riguarda l'operazione della 2° transazione in conflitto

- Se non sono in conflitto (rr) oppure se c'è su sé stesso → Non si fa nessun arco

OSSERVAZIONE: Se nell'inserire archi, ce l'ho già presente (inserito prima)

→ Non lo reinserisco



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO CONFLICT-SERIALIZZABILITÀ E CONFLICT-EQUIVALENZA DI UNO SCHEDULE

2. Se il grafo non presenta cicli → Cioè se a partire da un nodo non esiste un cammino che mi riporti lì (senza considerare un cammino diretto su sé stesso) → Allora lo schedule è **conflict-serializzabile** ad un qualche schedule seriale da trovare nel punto successivo
3. Troviamo adesso un possibile schedule seriale equivalente ad esso, guardando il grafico e vedendo da dove si parte e dove si arriva, rispettando gli archi → Si verifica quindi la **conflict-equivalenza** ad uno schedule seriale

APPLICAZIONE CONFLICT-SERIALIZZABILITÀ E CONFLICT-EQUIVALENZA DI UNO SCHEDULE

“Considerato il seguente schedule dire se è CSR”

$S = r1(x)w2(x)r3(x)r1(y)w2(y)r1(v)w3(v)r4(v)w4(y)w5(y)$

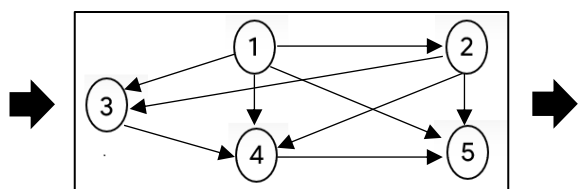
1. 1 2 3 4 5

1.a) Scrivo le operazioni, in ordine per ogni oggetto:

- $x \rightarrow r1, w2, r3$
- $y \rightarrow r1, w2, w4, w5$
- $v \rightarrow r1, w3, r4$

1.b) Considero adesso, in ordine le coppie di operazioni da parte delle transazioni su ogni oggetto:

- Su x:
 - $r1, w2 \rightarrow$ **Sono in conflitto** → 1.c) Arco orientato **da 1 a 2**
 - $r1, r3 \rightarrow$ NON Sono in conflitto → 1.c) Non inserisco niente
 - $w2, r3 \rightarrow$ **Sono in conflitto** → 1.c) Arco orientato **da 2 a 3**
- Su y:
 - $r1, w2 \rightarrow$ Sono in conflitto → 1.c) Arco orientato da 1 a 2 → Non lo inserisco perché è già presente
 - $r1, w4 \rightarrow$ **Sono in conflitto** → 1.c) Arco orientato **da 1 a 4**
 - $r1, w5 \rightarrow$ **Sono in conflitto** → 1.c) Arco orientato **da 1 a 5**
 - $w2, w4 \rightarrow$ **Sono in conflitto** → 1.c) Arco orientato **da 2 a 4**
 - $w2, w5 \rightarrow$ **Sono in conflitto** → 1.c) Arco orientato **da 2 a 5**
 - $w4, w5 \rightarrow$ **Sono in conflitto** → 1.c) Arco orientato **da 4 a 5**
- Su v:
 - $r1, w3 \rightarrow$ **Sono in conflitto** → 1.c) Arco orientato **da 1 a 3**
 - $r1, r4 \rightarrow$ NON Sono in conflitto → 1.c) Non inserisco niente
 - $w3, r4 \rightarrow$ **Sono in conflitto** → 1.c) Arco orientato **da 3 a 4**



2. Il grafo è aciclico quindi è **conflict-serializzabile** ad uno schedule seriale da trovare al prossimo punto



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO APPLICAZIONE CONFLICT-SERIALIZZABILITÀ E CONFLICT-EQUIVALENZA DI UNO SCHEDULE

3. Un possibile ordinamento per questo grafo potrebbe essere T1, T2, T3, T4, T5

→ Ragionamento:

- T1 precede T2, T3 e T4
- T2 precede T3, T4 e T5
- T3 precede T4
- T4 precede T5

→ Ho trovato quindi uno schedule seriale che rende lo schedule di partenza **conflict-equivalente** ad esso, per valere la conflict-serializzabilità

CSR, VSR E CONFLICT-EQUIVALENZA DI UNO SCHEDULE

“Considerato il seguente schedule dire se è CSR o VSR, e se è serializzabile mostrare uno schedule seriale equivalente”

S: r1(y) w3(z) r1(z) r2(z) w3(x) w1(x) w2(x) r3(y)

- Inizio con il provare se lo schedule è CSR

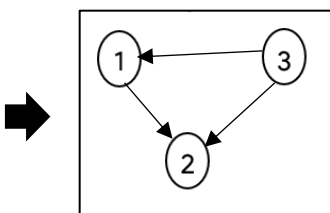
1. (1) (2) (3)

1.a) Scrivo le operazioni, in ordine per ogni oggetto:

- x → w3, w1, w2
- y → r1, r3
- z → w3, r1, r2

1.b) Considero adesso, in ordine le coppie di operazioni da parte delle transazioni su ogni oggetto:

- Su x:
 - w3, w1 → Sono in conflitto → 1.c) Arco orientato da 3 a 1
 - w3, w2 → Sono in conflitto → 1.c) Arco orientato da 3 a 2
 - w1, w2 → Sono in conflitto → 1.c) Arco orientato da 1 a 2
- Su y:
 - r1, r3 → NON Sono in conflitto → 1.c) Non inserisco niente
- Su z:
 - w3, w1 → Sono in conflitto → 1.c) Arco orientato da 3 a 1 → Non lo inserisco perché è già presente
 - r1, r2 → NON Sono in conflitto → 1.c) Non inserisco niente



2. Il grafo è aciclico quindi è **conflict-serializzabile (CSR)**, ad un qualche schedule seriale (da trovare nel punto successivo)
Il fatto che sia conflict-serializzabile implica, dalle proprietà → Che è anche **view-serializzabile (VSR)**



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO CSR E VSR DI UNO SCHEDULE

3. Un possibile ordinamento per questo grafo potrebbe essere **T3, T1, T2**, ragionando così:

- T3 precede T1 e T2
- T1 precede T2

→ Ho trovato quindi uno schedule seriale che rende lo schedule di partenza

conflict-equivalente ad esso, per valere la conflict-serializzabilità

CSR, VSR, VIEW-EQUIVALENZA E CONFLICT-EQUIVALENZA TRA DUE SCHEDULE

1. Comincio col provare che i due schedule siano conflict-equivalenti tra loro → Costruisco i grafi dei due schedule e controllo che siano uguali tra loro → Cioè se rispettano lo stesso ordine → Se lo sono allora sono **conflict-equivalenti**
2. Per provare che gli schedule siano conflict-serializzabili, devo verificare che i due schedule siano conflict-equivalenti ad uno schedule seriale ed ho due casi:
 - i. Caso in cui analizzo gli schedule separatamente → Se trovo uno schedule seriale equivalente, per ciascuno di essi → Allora ciascuno di essi è **conflict-serializzabile**
 - ii. Caso in cui analizzo gli schedule insieme tra loro → Se confrontando gli schedule, quello non seriale a quello seriale e vedo che vale la conflict-equivalenza → Allora essi sono **conflict-serializzabili**
3. Se i due schedule sono conflict-serializzabili tra loro (punto 2.ii)
→ Sono automaticamente **view-serializzabili**, altrimenti devo passare al punto 4.
4. Provo che i due schedule siano **view-equivalenti** tra loro
5. Provo che i due schedule siano la **view-serializzabili**, ad un qualche schedule seriale
→ Posso farlo sia separatamente, che per entrambi

APPLICAZIONE CSR, VSR, VIEW-EQUIVALENZA E CONFLICT-EQUIVALENZA TRA DUE SCHEDULE

ESERCIZIO 1

“Dati i seguenti schedule, dire se sono view-equivalenti o conflict-equivalenti.

Dire inoltre, se sono VSR o CSR”

S1=w2(x) r2(x) w1(x) r1(x) w2(y) r2(y) w1(x) w2(z)

S2= w1(x) r1(x) w2(x) r2(x) w1(x) w2(y) r2(y) w2(z)

1. Inizio con il costruire il grafo di S1

- Inserisco un nodo per ogni transazione → Ho **t1** e **t2**



- Scrivo le operazioni, in ordine per ogni oggetto:

- x → w2, r2, w1, r1, w1
- y → w2, r2
- z → w2

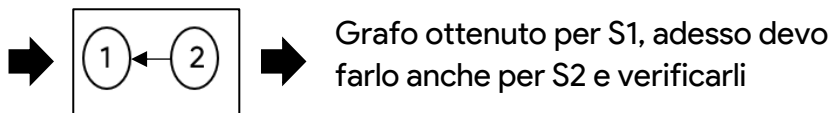


VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO APPLICAZIONE CSR, VSR, VIEW-EQUIVALENZA E CONFLICT-EQUIVALENZA TRA DUE SCHEDULE

- Considero adesso, in ordine le coppie di operazioni da parte delle transazioni su ogni oggetto:

- Su x:
 - $w_2, r_2 \rightarrow$ Sono in conflitto \rightarrow L'arco orientato da 2 a 2 non si mette
 - $w_2, w_1 \rightarrow$ **Sono in conflitto** \rightarrow Arco orientato **da 2 a 1**
 - $w_2, r_1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 1 \rightarrow Non lo inserisco perché è già presente
 - $w_2, w_1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 1 \rightarrow Non lo inserisco perché è già presente
 - $r_2, w_1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 1 \rightarrow Non lo inserisco perché è già presente
 - $r_2, r_1 \rightarrow$ NON Sono in conflitto \rightarrow Non inserisco niente
 - $r_2, w_1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 1 \rightarrow Non lo inserisco perché è già presente
 - $w_1, r_1 \rightarrow$ Sono in conflitto \rightarrow L'arco orientato da 1 a 1 non si mette
 - $w_1, w_1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 1 \rightarrow Non lo inserisco perché è già presente
 - $r_1, w_1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 1 \rightarrow Non lo inserisco perché è già presente
- Su y:
 - $w_2, r_2 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 2 \rightarrow Non inserisco nulla
- Su z: non faccio nulla



1. Inizio con il costruire il grafo di S2, perché va confrontato con quello di S1

- Inserisco un nodo per ogni transazione \rightarrow Ho **t1** e **t2**



- Scrivo le operazioni, in ordine per ogni oggetto:

- $x \rightarrow w_1, r_1, w_2, r_2, w_1$
- $y \rightarrow w_2, r_2$
- $z \rightarrow w_2$

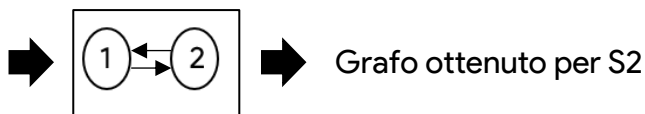


VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO APPLICAZIONE CSR, VSR, VIEW-EQUIVALENZA E CONFLICT-EQUIVALENZA TRA DUE SCHEDULE

- Considero adesso, in ordine le coppie di operazioni da parte delle transazioni su ogni oggetto:

- Su x:
 - $w1, r1 \rightarrow$ Sono in conflitto \rightarrow L'arco orientato da 1 a 1 non si mette
 - $w1, w2 \rightarrow$ **Sono in conflitto** \rightarrow Arco orientato **da 1 a 2**
 - $w1, r2 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 2 \rightarrow Non lo inserisco perché è già presente
 - $w1, w1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 1 \rightarrow Non inserisco nulla
 - $r1, w2 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 2 \rightarrow Non lo inserisco perché è già presente
 - $r1, r2 \rightarrow$ NON Sono in conflitto \rightarrow Non inserisco niente
 - $r1, w1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 1 \rightarrow Non inserisco nulla
 - $w2, r2 \rightarrow$ Sono in conflitto \rightarrow L'arco orientato da 2 a 2 non si mette
 - $w2, w1 \rightarrow$ **Sono in conflitto** \rightarrow Arco orientato **da 2 a 1**
 - $r2, w1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 1 \rightarrow Non inserisco nulla
- Su y:
 - $w2, r2 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 2 \rightarrow Non inserisco nulla
- Su z: non faccio nulla



\rightarrow Gli schedule hanno grafi diversi, e tra loro \rightarrow **Non sono conflict-equivalenti**

2. Verifico la conflict-serializzabilità:

ii. Essendo che tra loro non vale la conflict-equivalenza allora

\rightarrow **Non sono conflict-serializzabili (non CSR)**

3. Questo punto lo salto, perché non ho la conflict-serializzabilità

4. Provo la view-equivalenza:

- Per S1 e per S2 vale “legge-da” perché:
 - Su S1 ho che:
 - Su x \rightarrow r2 legge subito dopo che ha scritto w2, e r1 legge subito dopo che ha scritto w1, la scrittura finale w1 non ha importanza
 - Su y \rightarrow r2 legge subito dopo che ha scritto w2
 - Su S2 ho che:
 - Su x \rightarrow r1 legge subito dopo che ha scritto w1, e r2 legge subito dopo che ha scritto w2, la scrittura finale w1 non ha importanza
 - Su y \rightarrow r2 legge subito dopo che ha scritto w2
- \rightarrow S1 e S2 hanno la stessa ‘legge-da’, perché per entrambi viene rispettata la regola w-r, sia per la transazione 1 che per la transazione 2, indipendentemente se viene fatta prima o una o l'altra \rightarrow Lo stato finale di x,y,z è uguale in tutte e tre le transazioni



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO APPLICAZIONE CSR, VSR, VIEW-EQUIVALENZA E CONFLICT-EQUIVALENZA TRA DUE SCHEDULE

- Inoltre, sia su S1 che S2 ho le stesse scritture finali (che non vengono lette) sia su x che su z $\rightarrow w1(x), w2(z)$

\rightarrow Allora sono **view-equivalenti**

5. Considerando lo schedule S2 se sposto tutte le operazioni di T2 prima di T1 lo stato degli oggetti non cambia, e diventa identico ad S1 \rightarrow Vigendo la stessa "legge-da" e, in questo caso la stessa ultima lettura su x di w1 (view-equivalenza) \rightarrow Essendo entrambi view-equivalenti ad uno schedule seriale T2T1 allora \rightarrow Sono **view-serializzabili (VSR)**

ESERCIZIO 2

"Dati i seguenti schedule, dire se sono view-equivalenti o conflict-equivalenti.

Dire inoltre, se sono VSR o CSR"

S1: r1(x) r2(x) w1(x) r2(y) w1(y) w2(z) r3(z) r1(z) w3(x) w1(z) w3(z) r3 (y) w3(y)
S2: r1(x) r2(x) w1(x) r2(y) w1(y) w2(z) r1(z) w1(z) r3(z) w3(x) w3(z) r3 (y) w3(y)

1. Inizio con il costruire il grafo di S1

- Inserisco un nodo per ogni transazione \rightarrow Ho **t1, t2 e t3**



- Scrivo le operazioni, in ordine per ogni oggetto:

- x \rightarrow **r1, r2, w1, w3**
- y \rightarrow **r2, w1, r3, w3**
- z \rightarrow **w2, r3, r1, w1, w3**

- Considero adesso, in ordine le coppie di operazioni da parte delle transazioni su ogni oggetto:

- Su x:
 - r1, r2 \rightarrow NON Sono in conflitto \rightarrow Non inserisco niente
 - r1, w1 \rightarrow Sono in conflitto \rightarrow L'arco orientato da 1 a 1 non si mette
 - r1, w3 \rightarrow **Sono in conflitto** \rightarrow Arco orientato **da 1 a 3**
 - r2, w1 \rightarrow **Sono in conflitto** \rightarrow Arco orientato **da 2 a 1**
 - r2, w3 \rightarrow **Sono in conflitto** \rightarrow Arco orientato **da 2 a 3**
 - w1, w3 \rightarrow Sono in conflitto \rightarrow Arco orientato da 1 a 3 \rightarrow Non lo inserisco perché è già presente



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO APPLICAZIONE CSR, VSR, VIEW-EQUIVALENZA E CONFLICT-EQUIVALENZA TRA DUE SCHEDULE

- Su y:
 - $r_2, w_1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 1 \rightarrow Non lo inserisco perché è già presente
 - $r_2, r_3 \rightarrow$ NON Sono in conflitto \rightarrow Non inserisco niente
 - $r_2, w_3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 3 \rightarrow Non lo inserisco perché è già presente
 - $w_1, r_3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 3 \rightarrow Non lo inserisco perché è già presente
 - $w_1, w_3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 3 \rightarrow Non lo inserisco perché è già presente
 - $r_3, w_3 \rightarrow$ Sono in conflitto \rightarrow L'arco orientato da 3 a 3 non si mette
- Su z:
 - $w_2, r_3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 3 \rightarrow Non lo inserisco perché è già presente
 - $w_2, r_1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 1 \rightarrow Non lo inserisco perché è già presente
 - $w_2, w_1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 1 \rightarrow Non lo inserisco perché è già presente
 - $w_2, w_3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 3 \rightarrow Non lo inserisco perché è già presente
 - $r_3, r_1 \rightarrow$ NON Sono in conflitto \rightarrow Non inserisco niente
 - $r_3, w_1 \rightarrow$ **Sono in conflitto** \rightarrow Arco orientato da 3 a 1
 - $r_3, w_3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 3 a 3 \rightarrow Non inserisco nulla
 - $r_1, w_1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 1 \rightarrow Non inserisco nulla
 - $r_1, w_3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 3 \rightarrow Non lo inserisco perché è già presente
 - $w_1, w_3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 3 \rightarrow Non lo inserisco perché è già presente



1. Inizio con il costruire il grafo di S2, perché va confrontato con quello di S1

- Inserisco un nodo per ogni transazione \rightarrow Ho **t1**, **t2** e **t3**



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO APPLICAZIONE CSR, VSR, VIEW-EQUIVALENZA E CONFLICT-EQUIVALENZA TRA DUE SCHEDULE

- Scrivo le operazioni, in ordine per ogni oggetto:

- $x \rightarrow r1, r2, w1, w3$
- $y \rightarrow r2, w1, r3, w3$
- $z \rightarrow w2, r1, w1, r3, w3$

- Considero adesso, in ordine le coppie di operazioni da parte delle transazioni su ogni oggetto:

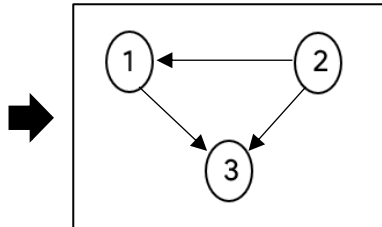
- Su x:
 - $r1, r2 \rightarrow$ NON Sono in conflitto \rightarrow Non inserisco niente
 - $r1, w1 \rightarrow$ Sono in conflitto \rightarrow L'arco orientato da 1 a 1 non si mette
 - $r1, w3 \rightarrow$ **Sono in conflitto** \rightarrow Arco orientato **da 1 a 3**
 - $r2, w1 \rightarrow$ **Sono in conflitto** \rightarrow Arco orientato **da 2 a 1**
 - $r2, w3 \rightarrow$ **Sono in conflitto** \rightarrow Arco orientato **da 2 a 3**
 - $w1, w3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 3 \rightarrow Non lo inserisco perché è già presente
- Su y:
 - $r2, w1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 1 \rightarrow Non lo inserisco perché è già presente
 - $r2, r3 \rightarrow$ NON Sono in conflitto \rightarrow Non inserisco niente
 - $r2, w3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 3 \rightarrow Non lo inserisco perché è già presente
 - $w1, r3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 3 \rightarrow Non lo inserisco perché è già presente
 - $w1, w3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 3 \rightarrow Non lo inserisco perché è già presente
 - $r3, w3 \rightarrow$ Sono in conflitto \rightarrow L'arco orientato da 3 a 3 non si mette
- Su z:
 - $w2, r1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 1 \rightarrow Non lo inserisco perché è già presente
 - $w2, w1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 1 \rightarrow Non lo inserisco perché è già presente
 - $w2, r3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 3 \rightarrow Non lo inserisco perché è già presente
 - $w2, w3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 2 a 3 \rightarrow Non lo inserisco perché è già presente
 - $r1, w1 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 1 \rightarrow Non inserisco nulla
 - $r1, r3 \rightarrow$ NON Sono in conflitto \rightarrow Non inserisco niente
 - $r1, w3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 3 \rightarrow Non lo inserisco perché è già presente
 - $w1, r3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 3 \rightarrow Non lo inserisco perché è già presente



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO APPLICAZIONE CSR, VSR, VIEW-EQUIVALENZA E CONFLICT-EQUIVALENZA TRA DUE SCHEDULE

- $w1, w3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 1 a 3 \rightarrow Non lo inserisco perché è già presente
- $r3, w3 \rightarrow$ Sono in conflitto \rightarrow Arco orientato da 3 a 3 \rightarrow Non inserisco nulla



Grafo ottenuto per S1, adesso devo farlo anche per S2 e verificarli

\rightarrow Gli schedule hanno grafi diversi, e tra loro \rightarrow **Non sono conflict-equivalenti**

2. Verifico la conflict-serializzabilità:

i. Provo prima per S1 e poi per S2:

- Per S1 noto che c'è un ciclo, quindi \rightarrow **Non è conflict-serializzabile (non CSR)** ad uno schedule seriale
- Per S2, per come è costruito il grafo, non ci sono cicli, quindi \rightarrow **È conflict-serializzabile (CSR)** ad uno schedule seriale T2, T1, T3 trovato così:
 - T2 precede T1 e T3
 - T1 precede T3

3. S2, essendo conflict-serializzabile è anche **view-serializzabile (VSR)**

4. Provo la view-equivalenza tra loro:

- Posso già dire che **non sono view-equivalenti** \rightarrow Perché non hanno la stessa "LEGGE-DA":
 - In S1, per esempio per l'oggetto z, ho r3 che legge ciò che ha scritto w2, mentre in S2 ho r3 che legge ciò che ha scritto w1!

5. Provo soltanto che lo schedule S1 sia view-equivalente a qualche schedule seriale:

- Considerando il grafo ottenuto da S1 posso provare l'ordine delle transazioni T2 T1 T3 oppure T2 T3 T1 \rightarrow Noto che per entrambi sulle operazioni x, y, z non vale la "LEGGE-DA" con lo schedule S1 \rightarrow Quindi S1 **non è view-serializzabile (non VSR)**

TIMESTAMP

- Per questa tecnica considero le richieste di lettura e scrittura, per ogni oggetto, e per ciascuna di esse vado a vederne il valore di timestamp, assumendo x oggetto generico:

- Richiesta di lettura \rightarrow **read(x,ts)** \rightarrow Confronto x con il valore di $WTM(x)$, e se vale:
 - $ts < WTM(x) \rightarrow$ Cioè se l'oggetto x ha un timestamp minore dell'ultima transazione che ha scritto su quell'oggetto \rightarrow **La richiesta viene respinta e la transazione uccisa** (come l'abort)
 - $ts \geq WTM(x) \rightarrow$ **La richiesta viene accolta e $RTM(x)$ prede come nuovo valore il valore di ts**

OSSERVAZIONE: Se confronto una lettura con un'altra lettura non vanno in conflitto e si può eseguire e come RTM prendo il valore più alto



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO TIMESTAMP

- Richiesta di scrittura \rightarrow **write(x,ts)** \rightarrow Vado a confrontare x con il valore di WTM(x) oppure con il valore RTM(x), a seconda dell'ultimo TS inserito su quell'oggetto, e se vale:
 - $ts < WTM(x)$ oppure $ts < RTM(x)$ \rightarrow **La richiesta viene respinta oppure la transazione viene uccisa**
 - $ts \geq WTM(x)$ oppure $ts \geq RTM(x)$ \rightarrow **La richiesta viene accolta e WTM(x) prede come nuovo valore il valore di ts**

OSSERVAZIONE:

- Inizialmente i contatori valgono RTM e WTM sugli oggetti sono 0
- Le transazioni che vengono fatte abortire \rightarrow Ripartono con un valore di ts maggiore all'ultimo valore di ts, che ha ogni oggetto

ESEMPIO DI TIMESTAMP

<u>RICHIESTA</u>	<u>NUOVO VALORE</u>
read(x,1)	RTM(x)=1
write(x,1)	WTM(x)=1
read(z,2)	RTM(z)=2
read(y,1)	RTM(y)=1
write(y,1)	WTM(y)=1
read(x,2)	RTM(x)=2
write(z,2)	WTM(z)=2

- Inizialmente RTM e WTM sugli oggetti x, y, z sono 0
- 1° Richiesta \rightarrow read(x,1) \rightarrow Verifico $ts < WTM(x)$ $\rightarrow 1 < 0 \rightarrow$ No \rightarrow Sono nel caso $ts \geq WTM(x) \rightarrow$ **RTM(x) assume valore 1**
- 2° Richiesta \rightarrow write(x,1) \rightarrow Verifico la lettura, cioè $ts < RTM(x)$ dato che è l'ultimo TS che ho per l'oggetto x $\rightarrow 1 < 1 \rightarrow$ No \rightarrow Sono nel caso $ts \geq RTM(x) \rightarrow$ **WTM(x) assume valore 1** (anche se già ce l'aveva)
- 3° Richiesta \rightarrow read(z,2) \rightarrow Verifico $ts < WTM(z)$ $\rightarrow 2 < 0 \rightarrow$ No \rightarrow Sono nel caso $ts \geq WTM(z) \rightarrow$ **RTM(z) assume valore 1**
- 4° Richiesta \rightarrow read(y,1) \rightarrow Verifico $ts < WTM(y)$ $\rightarrow 1 < 0 \rightarrow$ No \rightarrow Sono nel caso $ts \geq WTM(y) \rightarrow$ **RTM(y) assume valore 1**
- 5° Richiesta \rightarrow write(y,1) \rightarrow Verifico la lettura, cioè $ts < RTM(y)$ dato che è l'ultimo TS che ho per l'oggetto y $\rightarrow 1 < 1 \rightarrow$ No \rightarrow Sono nel caso $ts \geq RTM(y) \rightarrow$ **WTM(y) assume valore 1** (anche se già ce l'aveva)
- 6° Richiesta \rightarrow read(x,2) \rightarrow Verifico $ts < WTM(x)$ $\rightarrow 2 < 1 \rightarrow$ No \rightarrow Sono nel caso $ts \geq WTM(x) \rightarrow$ **RTM(x) assume valore 2**
- 7° Richiesta \rightarrow write(z,2) \rightarrow Verifico la lettura, cioè $ts < RTM(z)$ dato che è l'ultimo TS che ho per l'oggetto z $\rightarrow 2 < 2 \rightarrow$ No \rightarrow Sono nel caso $ts \geq RTM(z) \rightarrow$ **WTM(z) assume valore 2** (anche se già ce l'aveva)

VAGLINI-PREPARAZIONE ESERCIZI

TIMESTAMP SU SCHEDULE

- Dato uno schedule **S** si assume che il valore di **ts** per ogni sia rappresentato da ogni transazione **T** su quel determinato oggetto
- Considero le richieste di lettura e scrittura, per ogni oggetto, e per ciascuna di esse vado a vederne il valore di timestamp, assumendo **x** oggetto generico:

- Richiesta di lettura $\rightarrow r(x) \rightarrow$ Vado a confrontare il valore di **T** per **r** con **WTM(x)**, e se vale:
 - $T < WTM(x) \rightarrow$ Cioè se l'oggetto **x** ha un timestamp minore dell'ultima transazione che ha scritto su quell'oggetto \rightarrow **La richiesta viene respinta e la transazione uccisa** (come l'abort)
 - $T \geq WTM(x) \rightarrow$ **La richiesta viene accolta e RTM(x) prende come nuovo valore il valore di T**

OSSERVAZIONE: Se confronto una lettura con un'altra lettura non vanno in conflitto e si può eseguire e come RTM prendo il valore più alto

- Richiesta di scrittura $\rightarrow w(x) \rightarrow$ Vado a confrontare il valore di **T** per **w** con il valore di **WTM(x)** oppure con il valore **RTM(x)**, a seconda dell'ultimo **T** inserito su quell'oggetto, e se vale:
 - $T < WTM(x)$ oppure $T < RTM(x) \rightarrow$ **La richiesta viene respinta oppure la transazione viene uccisa**
 - $T \geq WTM(x)$ oppure $T \geq RTM(x) \rightarrow$ **La richiesta viene accolta e WTM(x) prende come nuovo valore il valore di T**

OSSERVAZIONE:

- Inizialmente i contatori valgono RTM e WTM sugli oggetti sono **0**
- Le transazioni che vengono fatte abortire \rightarrow **Ripartono** (RESTART) con valore di **T** maggiore all'ultimo valore di **T**, che ha ogni oggetto, ed ho due casi:
 - CASO A: Metto la nuova transazione **T** subito dopo l'ultimo valore considerato se, dopo non ho operazioni che hanno lo stesso **ts**, cioè con lo stesso **T** del nuovo **T**
 - CASO B: Se invece dopo l'ultimo valore considerato ho operazioni con lo stesso **ts**, cioè con lo stesso **T** del nuovo **T** \rightarrow Metto la nuova transazione **T** in coda a tutte le operazioni
- Si costruirà una **sequenza finale** \rightarrow Senza considerare tutte le operazioni riguardanti le transazioni che hanno abortito (anche prima dell'ABORT)

VAGLINI-PREPARAZIONE ESERCIZI

APPLICAZIONE TIMESTAMP SU SCHEDULE

“Dato il seguente schedule applicare il TS”

S: r1(y) w3(z) r1(z) r2(z) w3(x) w1(x) w2(x) r3(y)



<u>RICHIESTA</u>	<u>NUOVO VALORE</u>
r1(y)	RTM(y)=1
w3(z)	WTM(z)=3
r1(z)	abort, restart T1 come T4
r4(y)	RTM(y)=4
r4(z)	RTM(z)=4
r2(z)	abort, restart T2 come T5
r5(z)	RTM(z)=5
w3(x)	WTM(x)=3
r3(y)	RTM(y)=4
w4(x)	WTM(x)=4
w5(x)	WTM(x)=5

- Inizialmente RTM e WTM sugli oggetti x, y, z sono 0

- 1° Richiesta → r1(y) dove $T=1$ → Verifico $T < WTM(y)$ → $1 < 0$ → No → Sono nel caso $T \geq WTM(y)$ → **RTM(y) assume valore T, cioè 1**
- 2° Richiesta → w3(z) dove $T=3$ → Verifico sia lettura che scrittura, anche se non ho operazioni prima su z, cioè $T < RTM(z)$ e $T < WTM(z)$ → $3 < 0$ → No → Sono nel caso $T \geq RTM(z)$ oppure $T > WTM(z)$ → **WTM(z) assume valore 3**
- 3° Richiesta → r1(z) dove $T=1$ → Verifico $T < WTM(z)$ → $1 < 3$ → Sì → **ABORT T1, RESTART T1 come T4** → Mi trovo nel CASO B, visto che ho una transazione T4 subito dopo → Quindi, per questo caso, andrò nella 5° Richiesta
- 4° Richiesta → r4(y) dove $T=4$ → Poiché su y ho solo operazioni di lettura, questa non va in conflitto e come RTM prende il valore maggiore → **RTM(y) assume valore T, cioè 4**
- 5° Richiesta → r4(z) dove $T=4$ → Questa è la transazione che è ripartita → Verifico $T < WTM(z)$ → $4 < 3$ → No → Sono nel caso $T \geq WTM(z)$ → **RTM(z) assume valore T, cioè 4**
- 6° Richiesta → r2(z) dove $T=2$ → Verifico $T < WTM(z)$ → $2 < 3$ → Sì → **ABORT T2, RESTART T2 come T5** → Mi trovo nel CASO B, visto che ho una transazione T3 subito dopo → Quindi, per questo caso, andrò nella 7° Richiesta
- 7° Richiesta → r5(z) dove $T=5$ → Verifico $T < WTM(z)$ → $5 < 3$ → No → Sono nel caso $T \geq WTM(z)$ → **RTM(z) assume valore T, cioè 5**
- 8° Richiesta → w3(x) dove $T=3$ → Verifico sia lettura che scrittura, anche se non ho operazioni prima su x, cioè $T < RTM(x)$ e $T < WTM(x)$ → $3 < 0$ → No → Sono nel caso $T \geq RTM(x)$ oppure $T > WTM(x)$ → **WTM(x) assume valore 3**
- 9° Richiesta → r3(y) dove $T=3$ → Poiché su y ho solo operazioni di scrittura ma solo letture, questa non va in conflitto e come RTM prende il valore maggiore → **RTM(y) assume valore T, cioè 4** (cioè rimane lo stesso)



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO APPLICAZIONE TIMESTAMP SU SCHEDULE

- 10° Richiesta $\rightarrow w_4(x)$ dove $T=4 \rightarrow$ Verifico la scrittura, cioè $T < WTM(x)$ dato che è l'ultimo TS che ho per l'oggetto $x \rightarrow 4 < 3 \rightarrow$ No \rightarrow Sono nel caso $T \geq RTM(x) \rightarrow WTM(x)$ assume valore 4
- 11° Richiesta $\rightarrow w_5(x)$ dove $T=5 \rightarrow$ Verifico la scrittura, cioè $T < WTM(x)$ dato che è l'ultimo TS che ho per l'oggetto $x \rightarrow 5 < 4 \rightarrow$ No \rightarrow Sono nel caso $T \geq RTM(x) \rightarrow WTM(x)$ assume valore 5

- La sequenza finale senza le operazioni delle transazioni T_1 e T_2 è data da:

$\rightarrow w_3(z) \ r_4(y) \ r_4(z) \ w_3(x) \ r_3(y) \ r_5(z) \ w_3(x) \ r_3(y) \ w_4(x) \ w_5(z)$

S2PL

- Per utilizzare S2PL, dobbiamo tenere sott'occhio alcune cose della tavola dei conflitti:

Richiesta	Stato della risorsa		
r_lock	$free$	r_locked	w_locked
w_lock	OK / r_locked	OK / r_locked	NO / w_locked
$unlock$	OK / w_locked	NO / r_locked	NO / w_locked
	error	OK / depends (1)	OK / free

- Imponendo che il "contatore dei lettori" inizialmente sia 0, per un dato oggetto generico x

$\rightarrow cr(x)=0$

- Considero adesso, una serie di CASI che si sviluppano dalle operazioni di richiesta sulle risorse, scorrendo lo scheduler

1. Se ho una prima richiesta di lettura da parte di una transazione, che supponiamo essere T_1 , sull'oggetto x , che è libero:

- La lettura si fa $\rightarrow r_1_lock(x) \rightarrow$ E si incrementa di 1 il contatore delle letture su quell'oggetto $\rightarrow cr(x)=1$

2. Avendo già avuto una richiesta di lettura da parte di una transazione, che supponiamo essere T_1 , sull'oggetto x (es. $cr(x)=1$), se ho un'altra richiesta di lettura che la segue, da parte di un'altra transazione, che supponiamo essere T_2 , sullo stesso oggetto x :

- La lettura si fa senza nessun conflitto $\rightarrow r_2_lock(x) \rightarrow$ E si incrementa di 1 il contatore delle letture su quell'oggetto $\rightarrow cr(x)=2$

OSSERVAZIONE: Gli stessi passaggi del punto 2, si ripetono ogni volta che incontro altre richieste di lettura, da parte delle transazioni sullo stesso oggetto $x \rightarrow$ Quindi altre letture \rightarrow Altri incrementi sul contatore dei lettori

3. Avendo già avuto una richiesta di lettura da parte di una transazione, che supponiamo essere T_1 , sull'oggetto x (es. $cr(x)=1$), se ho una "prima" richiesta di lettura che la segue, da parte di un'altra transazione, che supponiamo essere T_2 , o della stessa transazione T_1 , però su un altro oggetto, che supponiamo essere y libero:

- La lettura si fa senza conflitti $\rightarrow r_2_lock(y)$ oppure $r_1_lock(y) \rightarrow$ Se ne incrementa di 1 il relativo contatore delle letture, su quell'oggetto che da 0 va a 1 $\rightarrow cr(y)=1$

OSSERVAZIONE: Se non è una "prima" richiesta di lettura, e il contatore ha già fatto altre letture, aumenterà di 1 il contatore aggiungendolo al valore che già possiede



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO S2PL

-
4. Se ho una prima richiesta di scrittura da parte di una transazione, che supponiamo essere **T1**, sull'oggetto **x**, che è libero:
- La scrittura si fa normalmente → **w1_lock(x)**
-
5. Avendo già avuto una richiesta di lettura da parte di una transazione, che supponiamo essere **T1**, sull'oggetto **x** (es. **cr(x)=1**), se ho una richiesta di scrittura che la segue, da parte della stessa transazione **T1** sullo stesso oggetto **x**:
- La scrittura si fa utilizzando il "lock esclusivo" → **upgradelock w1_lock(x)**
-
6. Avendo già avuto più richieste di richiesta di lettura da parte di più transazioni, che supponiamo essere **T1** e **T2** consecutivamente, sull'oggetto **x**, quindi con contatore di lettori uguale a 2 → **cr(x)=2** → Se ho una richiesta di scrittura che la segue, da parte di una transazione **T1** sullo stesso oggetto **x**:
- Anche se è presente **T1**, → **w1(x)** non si fa, e si mette in attesa questa operazione con tutte le altre successive operazioni di **T1**, in attesa che **T2** si concluda → **WAIT T1 T2**
-
7. Avendo già avuto una richiesta di lettura da parte di una transazione, che supponiamo essere **T1** sull'oggetto **x** (es. **cr(x)=1**), se ho una richiesta di scrittura che la segue, da parte di una transazione **T2** sullo stesso oggetto **x**:
- **w2(x)** non si fa, e si mette in attesa questa operazione con tutte le altre successive operazioni di **T2**, in attesa che **T1** si concluda → **WAIT T2 T1**
-
8. Avendo già avuto una richiesta di scrittura da parte di una transazione, che supponiamo essere **T1** sull'oggetto **x**, se ho una richiesta di scrittura che la segue, da parte di una transazione **T2** sullo stesso oggetto **x**:
- **w2(x)** non si fa, e si mette in attesa questa operazione con tutte le altre successive operazioni di **T2**, in attesa che **T1** si concluda → **WAIT T2 T1**
-
9. Avendo già avuto una richiesta di scrittura da parte di una transazione, che supponiamo essere **T1** sull'oggetto **x**, se ho una richiesta di lettura che la segue, da parte di una transazione **T2** sullo stesso oggetto **x**:
- **r2(x)** non si fa, e si mette in attesa questa operazione con tutte le altre successive operazioni di **T2**, in attesa che **T1** si concluda → **WAIT T2 T1**
-



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO S2PL

10. Avendo già avuto una richiesta di scrittura da parte di una transazione, che supponiamo essere **T1** sull'oggetto **x**, se ho una richiesta di scrittura che la segue, da parte di una transazione **T2** su un altro oggetto **y**, per quella y devo vedere, se prima dell'operazione di T1 scorrendo lo scheduler all'indietro, ho:

- Nessun'operazione su y, quindi è libera → **Caso nel caso 4**
- Una lettura della stessa transazione T2 su y → **Caso nel caso 5**
- Più letture da parte di più transazioni, come ad esempio T2 e T1, su y → **Caso nel caso 6**

OSSERVAZIONE: Per applicare il punto 6, rispetto a questo punto 9 considero T2 prima di T1 come transazioni consecutive

- Una lettura da parte di un'altra transazione, come ad esempio T1 su y → **Caso nel caso 7**
- Una scrittura da parte di un'altra transazione, come ad esempio T2 su y → **Caso nel caso 8**

11. Se viene fatto un commit **c1** su una transazione **T1** che può aver lavorato per esempio su un oggetto **x**, oppure su più oggetti, come ad esempio **x,y,z**:

- Si sbloccano, oggetto oppure gli oggetti che la transazione ha usato → **unlock(x) / unlock(x,y,z)**
- Inoltre, se sull'oggetto / sugli oggetti ci sono state delle letture da parte di T1, si decrementa il contatore / i contatori relativi a quell'oggetto/quegli oggetti
→ Considerando, per esempio, che $cr(x)=2$ con il rilascio decremento di 1 → **$cr(x)=1$**
→ Considerando, per esempio, che $cr(x)=2$, $cr(y)=3$, $cr(z)=4$ con il rilascio decremento di 1 → **$cr(x)=1$, $cr(y)=2$, $cr(z)=3$**

12. Se, in precedenza, si è già verificata un'attesa da parte di una transazione **T1** verso una transazione **T2** (**WAIT T1 T2**), e adesso si verifica un'attesa da parte della transazione **T2** verso la transazione **T1** (**WAIT T2 T1**) → Sono in uno stallo:

- Quindi si verifica l'attesa incrociata → **deadlock**
- Si abortisce la transazione che in questo momento ha causato il deadlock → **abort t2**
- Di questa transazione T2 si annullano tutte le operazioni, di lettura e scrittura, fatte in precedenza a questo momento:
 - Per le letture, si decrementano i contatori, relativi a tutti gli oggetti riguardanti la transazione T2 → Per esempio se ho fatto una lettura su x, e $cr(x)=2$ → **$cr(x)=1$**
- Si toglie dalla lista di attesa la transazione che non ha causato il deadlock, in questo caso T1, e se ne eseguono tutte le operazioni che la riguardano → **dequeue T1**
- Appena si conclude l'operazione T1, che libera le risorse necessarie a T2 → **Si eseguono tutte le operazioni che riguardano T2, sia quelle precedenti, che le successive**



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO S2PL

13. Se si è verificato il caso precedente e ho un caso precedente a questo in cui un'altra transazione, supponiamo essere **T3**, che era in attesa di utilizzare **T1** (**WAIT T3 T1**):
- Dobbiamo aspettare che si concluda il caso precedente → **Che si liberi T2, in modo che rilasci le operazioni necessarie a T1**
 - Poi dobbiamo aspettare che concluda anche T1, in modo che sblocchi le risorse necessarie a T3
 - Eseguiamo l'operazione di T1

- Si esegue la SEQUENZA FINALE escludendo:
- Le operazioni delle transazioni quando vengono bloccate e messe in attesa
→ **Di quelle transazioni si prendono le operazioni, quando ripartono dopo l'attesa e vengono portate a termine, fino al commit**
 - La parte in cui le transazioni abortiscono, quindi anche tutte le operazioni precedenti al momento dell'abort → **Di quelle transazioni si prendono, quindi, solo le operazioni che ripartono e vengono portate a termine, fino al commit**

APPLICAZIONE S2PL

"Dato il seguente schedule applicare il S2PL"

r1(x), w1(x), w3(x), r2(y), r3(y), w3(y), c3, w1(y), c1, r2(x), c2

1°	r1(x)	r1_lock(x)	cr(x)=1			
2°	w1(x)	upgradelock w1_lock(x)				
3°	w3(x)	WAIT T3 T1				
4°	r2(y)	r2_lock(y)	cr(y)=1			
5°	w1(y)	WAIT T1 T2				
6°	r2(x)	WAIT T2 T1	deadlock	abort t2	cr(y)=0	dequeue T1
7°	w1(y)	w1_lock(y)				
8°	c1	unlock(x,y)	cr(x)=0			
9°	r2(y)	r2_lock(y)	cr(y)=1			
10°	r2(x)	r2_lock(x)	cr(x)=1			
11°	c2	unlock(x,y)	cr(x)=0, cr(y)=0			
12°	w3(x)	w3_lock(x)				
13°	r3(y)	r3_lock(y)	cr(y)=1			
14°	w3(y)	upgradelock w3_lock(y)				
15°	c3	unlock(x,y)	cr(y)=0			
16°	r1(x), w1(x), w1(y), c1, r2(y), r2(x), c2, w3(x), r3(y), w3(y), c3					

VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO APPLICAZIONE S2PL

1° PASSO:

- Mi trovo come nel CASO 1 (S2PL) → T1 'lockato' in lettura su x → Aumento il contatore dei lettori di 1 per x

2° PASSO:

- Mi trovo come nel CASO 5 (S2PL) → Prendo il lock esclusivo, visto che ho solo T1 che legge su x

3° PASSO:

- Mi trovo come nel CASO 8 (S2PL) → T3, insieme alle sue operazioni successive rimane in attesa della risorsa usata da T1

4° PASSO:

- Mi trovo come nel CASO 1 (S2PL) → T1 'lockato' in lettura su y → Aumento il contatore di lettori di 1 per y

5° PASSO:

- Mi trovo come nel CASO 7 (S2PL) → T1, insieme alle sue operazioni successive rimane in attesa della risorsa usata da T2

6° PASSO:

- Mi trovo come nel CASO 6 (S2PL) → L'attesa è dovuta perché si è verificato un CASO 9 (S2PL) → T2, insieme alle sue operazioni successive rimane in attesa della risorsa usata da T1 → Ciò causa deadlock → Quindi cancello l'operazione fatta da T2, togliendo anche l'operazione precedente → Decremento il contatore dei lettori su y per il quale ci avevo fatto una sola lettura con T2 → Sblocco le operazioni di T1 da dove erano rimaste ed eseguo tutte quelle che ha, nel resto dello schedule → Rieseguirò poi T2 non appena finisce T1, e poi T3 che, anch'essa, era in attesa di T1

7° PASSO:

- Mi trovo come nel CASO 4 (S2PL) → Essendo adesso y libera → Ci metto in lock semplice in scrittura per T1

8° PASSO:

- Mi trovo come nel CASO 11 (S2PL) → Rilascio x e y, e sulla x decremento il contatore dei lettori, perché in precedenza T1 ci aveva fatto la lettura
- Adesso rieseguo immediatamente la transazione T2 che avevo abortito, insieme a tutte le sue operazioni

9° PASSO:

- Mi trovo come nel CASO 1 (S2PL) → T2 'lockato' in lettura su y → Aumento il contatore dei lettori di 1 per y



VAGLINI-PREPARAZIONE ESERCIZI

...CONTINUO APPLICAZIONE S2PL

10° PASSO:

- Mi trovo come nel CASO 1 (S2PL) → T2 'lockato' in lettura su x → Aumento il contatore dei lettori di 1 per x

11° PASSO:

- Mi trovo come nel CASO 11 (S2PL) → Rilascio x e y, e per entrambe decremento i contatori dei lettori, perché le letture fatte nei due passaggi precedenti da T2
- Adesso rieseguo immediatamente la transazione T3 che era in attesa per T1, ma è slittata a causa di T2, la quale andava fatta il prima possibile, come nel CASO 13 (S2PL)

12° PASSO:

- Mi trovo come nel CASO 4 (S2PL) → Essendo adesso x libera → Ci metto in lock semplice in scrittura per T3

13° PASSO:

- Mi trovo come nel CASO 1 (S2PL) → T3 'lockato' in lettura su y → Aumento il contatore dei lettori di 1 per y

14° PASSO:

- Mi trovo come nel CASO 5 (S2PL) → Prendo il lock esclusivo, visto che ho solo T3 che legge su y

15° PASSO:

- Mi trovo come nel CASO 11 (S2PL) → Rilascio x e y, e sulla y decremento il contatore dei lettori, perché in precedenza T3 ci aveva fatto la lettura

16° PASSO:

- LETTURA FINALE escludendo le richieste ai PASSI: 3°, 4°, 5°, 6°