

Presentazione del corso-20220301 0808-1

Con il termine Base di Dati si intende un insieme organizzato di dati che viene utilizzato come supporto ad attività specifiche di un'organizzazione. Caratteristiche di una Base di Dati:

- Grande: Gigabyte, anche Terabyte nel caso di applicazioni scientifiche.
- Persistente: i dati non devono sparire al termine dell'esecuzione di un'applicazione, ma devono essere trovati anche in tempi successivi.
- Condivisa: in genere più attività, applicazioni, fanno riferimento agli stessi dati.

Un DBMS, come dice il termine, è un sistema per la gestione di basi di dati. Esempi di DBMS:

- Access
- DB2
- Oracle
- MySQL
- PostgreSQL
- SQLServer

In un Ateneo due applicazioni gestiscono informazioni sull'orario delle lezioni e sull'orario di ricevimento dei docenti, ciascuna con il proprio archivio dati. Sicuramente i due archivi conterranno informazioni duplicate, necessarie anche per la produzione di report e per inviare comunicazioni agli studenti (nome e cognome del docente e relativi insegnamenti). ⇒ Ridondanza dei dati archiviati. I due archivi può darsi non siano allineati, certi aggiornamenti sui dati comuni può darsi siano stati inseriti in un archivio e non nell'altro. ⇒ Le informazioni prodotte dalle due applicazioni possono essere fra loro inconsistenti.

Gestione dati con DBMS

Tutte le applicazioni fanno riferimento alla stessa collezione di dati memorizzata su disco. ⇒ Es: il nome e il cognome dei docenti vengono memorizzati una volta sola. I DBMS permettono di gestire grandi quantità di dati in modo condiviso e persistente, assicurando affidabilità e privatezza. Una precisazione: i DBMS utilizzano a loro volta file per la memorizzazione dei dati, però ne permettono una organizzazione più sofisticata.

Obiettivi di un DBMS

Modalità di accesso generale ad archivi integrati. Accesso concorrente ai dati. Privatezza dei dati: meccanismi di protezione (es. chiavi di accesso). Integrità dei dati: vincoli di consistenza (condizioni che devono essere soddisfatte). Ripristino dei dati: backup (salvataggio), giornale, log (diario di tutte le modifiche fatte a partire dall'ultimo backup).

Modello, Schema e Istanza

Modello dei dati: il modello relazionale (inizio anni 70) è attualmente il più diffuso e utilizza il concetto di relazione per organizzare i dati in insiemi di record a struttura fissa; un modello è un insieme di concetti per organizzare i dati di interesse e descriverne la struttura; fornisce un insieme di costruttori che permettono di definire nuovi tipi a partire da tipi elementari, es: tipo base, tipo record, tipo tabella. altri tipi di modelli: gerarchico (anni 60), reticolare (inizio anni 70), a oggetti (anni 80), XML (anni 90), NoSQL (recente). Schema: descrizione della organizzazione dei dati di interesse in accordo al modello considerato, es: descrizione delle tabelle che vogliamo usare in accordo al modello relazionale. Istanza: contenuto corrente della base di dati, es: contenuto effettivo delle tabelle.

Dizionario dei dati/metadati

Il sistema mantiene un catalogo di tutte le strutture definite nel data base con le loro caratteristiche: ad esempio, l'elenco di tutte le tabelle definite, ognuna con i nomi delle colonne che la compongono, oppure l'elenco di tutti gli indici definiti. Il dizionario è strutturato secondo il modello logico: ad esempio strutturato in tabelle. Il dizionario è accessibile con gli stessi strumenti con cui si accede ai dati: ad esempio, con una query SQL si può avere l'elenco dei nomi delle tabelle definite.

Architettura di un DBMS: livelli di astrazione

Livello logico: modello di dati gerarchico, reticolare, relazionale, a oggetti; schema logico: descrizione della struttura dei dati secondo il modello utilizzato; esempio: modello relazionale, descrizione della struttura delle tabelle. Livello fisico (o interno): schema fisico, rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione; esempio: descrizione degli indici costruiti. Livello esterno: descrizione di parte della base di dati per mezzo di un modello logico (viste parziali o

DOCENTE	INSEGNAMENTI	ORARIO
Paola Bianchi	Algoritmi	lunedì 15:00-16:00
Nicola Bruni	Programmazione	martedì 9:30-10:30
Anna Neri	Statistica	martedì 11:30-12:30
Mario Rossi	Fisica I	giovedì 10:30-12:30
Luigi Verdi	Analisi I	mercoledì 14:00-15:30
:	Basi di dati	:
	Sistemi Informativi	:

derivate che riflettono il punto di vista di un particolare utente o insieme di utenti); il livello esterno può coincidere con il livello logico: la vista non è stata definita, l'applicazione vede direttamente lo schema logico

Livelli di astrazione: indipendenza logica e fisica dei dati

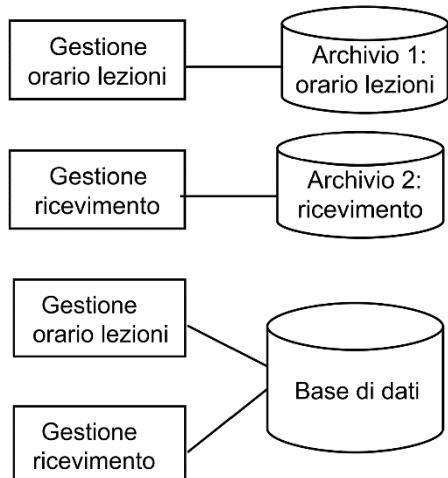
Indipendenza fisica: deriva dall'uso di un modello logico dei dati; gli utenti e le applicazioni interagiscono con il data base secondo la sua struttura logica; è possibile cambiare la struttura fisica senza modificare l'applicazione, ad es. è possibile aggiungere o togliere un indice al data base senza cambiare il programma di interrogazione dei dati. Indipendenza logica: deriva dall'uso delle viste; se un'applicazione interagisce con il data base attraverso le viste è possibile modificare il livello logico senza modificare l'applicazione, purché le viste rimangano inalterate. Ad es. è possibile suddividere una tabella in due tabelle purché per l'applicazione sia definita una vista che vede una sola tabella.

Linguaggi per basi di dati

Data Definition Language, DDL: definizione dello schema logico; definizione schema fisico, schema esterno (viste); autorizzazioni per l'accesso. Data Manipulation Language, DML: inserimento, modifica, cancellazione, interrogazioni; opera sulla istanza di una base di dati. SQL è un linguaggio che offre le due funzionalità.

Utenti delle Basi di Dati

Amministratore della base di dati: responsabile della progettazione, controllo e amministrazione; gestisce le autorizzazioni per l'accesso ai dati; è responsabile dell'affidabilità (backup e recovery); cura la manutenzione del sistema. Programmatori di applicazioni: progettano e implementano le applicazioni usate dagli utenti per accedere alla base di dati. Utenti finali: interagiscono con la base di dati tramite le applicazioni. Utenti casuali: accedono ai dati in modo non predefinito, tipicamente in modo interattivo.



Esistono tanti esempi di sistemi, ne ho elencato alcuni dei più conosciuti sistemi di gestione di basi di dati: Access, DB2, Oracle, postgresql. Una volta imparato a utilizzare uno di questi sistemi la logica è esattamente la stessa, questo perché appunto il modello dei dati che sta dietro a questi sistemi è lo stesso, perché il modello relazionale.

Per capire un po' meglio che cos'è un sistema di gestione di dati partiamo appunto a fare un esempio di gestione dati in assenza di una base di dati e immaginiamo di avere due applicazioni in un ateneo e immaginiamo di averne una che gestisce informazioni sull'orario delle lezioni e una invece che gestisce informazioni sull'orario di ricevimento dei docenti, immaginiamo che ciascuna faccia riferimento a degli archivi di dati diversi separati, ora è chiaro che questi due archivi conterranno delle informazioni duplicate perché tutti e due le applicazioni hanno bisogno di sapere il nome e il cognome del docente e gli insegnamenti

quindi da una parte abbiamo il problema della ridondanza dei dati, ovvero la stessa informazione che viene archiviata più volte, e l'altro problema, forse ancor più grave, è che questi due archivi potrebbero non essere allineati, perché certi aggiornamenti che vengono fatti su uno dei due archivi tramite una delle applicazioni potrebbero non essere inseriti poi nell'altro, per cui le due applicazioni si troverebbero ad avere a che fare con informazioni inconsistenti.

INSEGNAMENTO	DOCENTE	AULA	ORARIO
Algoritmi	Paola Bianchi	A2	8:30-10:30
Analisi I	Mario Rossi	A1	8:30-10:30
Basi di dati	Luigi Verdi	A2	10:30-12:30
Fisica I	Anna Neri	A1	12:30-14:30
Programmazione	Paola Bianchi	A1	10:30-12:30
Sistemi Informativi	Luigi Verdi	A3	8:30-10:30
Statistica	Nicola Bruni	A2	12:30-13:30
:	:	:	:

Immaginiamo di avere un archivio che contiene queste informazioni: insegnamento, il docente, l'aula e l'orario, quindi una prima applicazione che fa riferimento a un archivio organizzato con queste informazioni, l'altra applicazione potrebbe invece fare riferimento a un archivio che contiene il riferimento al docente e agli insegnamenti che il docente tiene e all'orario di ricevimento, quindi in assenza di una base di dati siamo in una situazione che può essere rappresentata da questa da questa figura, quindi abbiamo le due applicazioni ognuna che

fa riferimento ad un archivio separato con i problemi che abbiamo detto, ridondanza ed inconsistenza. L'obiettivo invece dei dei sistemi di gestione di basi di dati è quello di avere un unico archivio con le informazioni e fare in modo che le applicazioni facciano riferimento a questa unica base di dati in modo da evitare sia la ridondanza delle informazioni sia appunto l'inconsistenza.²

Apro una parentesi: in realtà il fatto di avere un'unica base di dati non ci garantisce che non ci sia ridondanza perché insomma se la base di dati è progettata male la ridondanza può essere presente però vedremo durante il corso tutta una serie di tecniche e procedimenti che ci permettono di progettare una base di dati che questi problemi non li ha.

Avendo una gestione tramite basi di dati tutte le applicazioni faranno riferimento alla stessa collezione di dati che è memorizzata su disco e quindi ad esempio il nome e il cognome dei docenti verranno memorizzati solo una volta, si riesce ad evitare il problema della ridondanza.

I sistemi di gestione di basi di dati sono strumenti complessi e permettono di gestire grandi quantità di dati e permettono di fare questa operazione anche garantendo che le operazioni possono essere fatte in modo persistente assicurando l'affidabilità quindi se ci sono dei guasti il sistema di gestione di basi di dati è in grado di assicurare comunque che le cose poi possano tornare a funzionare correttamente e quindi a non perdere informazioni e permettono poi la gestione condivisa da parte di più utenti, è chiaro che anche i sistemi di gestione di basi di dati utilizzano allora molto file degli archivi per memorizzare i dati che vogliono organizzare però permettono un'organizzazione molto più sofisticata di questi file è che garantisce tutte le proprietà che abbiamo detto quindi comunque si tratta di file organizzati in modo da garantire tutte queste caratteristiche.

Gli obiettivi di una base di dati: quello che vogliamo è fare in modo che sia possibile accedere a archivi integrati in modo concorrente quindi più utenti fanno riferimento a questi dati, devono essere previsti dei meccanismi di protezione, privatezza dei dati, in modo da consentire a certi utenti di fare certe operazioni e ad altri invece di farne altri, l'amministratore della base di dati avrà certi privilegi, l'utente invece ne avrà altri.

Il sistema deve garantire quella che si chiama l'integrità dei dati, cioè ci sono dei vincoli di integrità che devono essere soddisfatti, si tratta di condizioni che devono essere soddisfatti dalla base di dati. Un altro aspetto importante è quello del ripristino dei dati quindi qui si parla soprattutto di backup e di possibilità di mantenere i giornali il log della base di dati in modo da poter risalire a quello che era stata fatto prima di un eventuale guasto.

Ci sono tre concetti importanti che sono quello di modello, schema e istanza.

Che cos'è il modello dei dati? Il modello di dati è il modello che seguiremo per realizzare la nostra base di dati, noi faremo riferimento al modello relazionale, tale modello è stato introdotto all'inizio degli anni 70 e continua ad essere il modello di dati ancora più diffuso e utilizzato. Come mai questo? Perché si basa su un concetto matematico, quello di relazione che permette di organizzare i dati insieme di record a struttura fissa e quindi il modello è da questo punto di vista è piuttosto semplice e ed è molto rigoroso proprio perché si basa su un concetto matematico. Non è l'unico modello che è stato utilizzato nella letteratura prima del modello relazionale sono stati introdotti anche il modello gerarchico e il modello reticolare, questi sono modelli che organizzano i dati utilizzando delle strutture gerarchiche, come alberi oppure grafi, a differenza del modello relazionale in cui la struttura di base è una tabella. Sono stati proposti negli anni modelli che fanno riferimento a dati di tipo XML però il modello relazionale è quello più utilizzato. L'ultimo inventato, nosql non è un vero e proprio modello ma un insieme di modelli che via via vengono presentati quando il modello relazionale è troppo vincolante per rappresentare informazioni che mal rappresentano con un formato così rigido come quello tabellare, soprattutto quando si ha a che fare con big data, una grande quantità di informazioni che difficilmente si riescono a memorizzare in modo tabellare ma in molti casi poi questi modelli si riconducono in parte al modello relazionale quindi un informatico non può prescindere dalla conoscenza del modello relazionale.

Che cos'è lo schema? Lo schema di una base di dati è la descrizione dei dati di interesse in accordo col modello di dati che si è scelto quindi il modello relazionale in cui i dati sono organizzati in tabelle, quindi lo schema della base di dati è rappresentato dalla forma delle tabelle cioè dalle informazioni che vogliamo andare a descrivere con e tramite questa struttura.

Istanza della base di dati? L'istanza rappresenta il contenuto vero e proprio della base di dati, quindi i record che andiamo a memorizzare nella base di dati utilizzando quel formato particolare che poi vedremo.

Lo schema della basi di dati:

ORARIO LEZIONI			
INSEGNAMENTO	DOCENTE	AULA	ORARIO

Sunto: come modello dei dati abbiamo il modello relazionale, lo schema è la base di dati organizzata secondo il modello dei dati utilizzato, in altre parole la forma che diamo al nostro contenitore e poi il contenuto, cioè l'insieme dei record che andiamo ad inserire e che formano l'istanza della base di dati.

Esempio: facendo riferimento alla tabella che abbiamo visto prima su l'orario delle lezioni, lo schema è definito specificando
L'istanza della basi di dati:

ORARIO LEZIONI			
INSEGNAMENTO	DOCENTE	AULA	ORARIO
Algoritmi	Paola Bianchi	A2	8:30-10:30
Analisi I	Mario Rossi	A1	8:30-10:30
Basi di dati	Luigi Verdi	A2	10:30-12:30
Fisica I	Anna Neri	A1	12:30-14:30
Programmazione	Paola Bianchi	A1	10:30-12:30
Sistemi Informativi	Luigi Verdi	A3	8:30-10:30
Statistica	Nicola Bruni	A2	12:30-13:30
:	:	:	:

il nome della tabella e i nomi delle colonne degli attributi che caratterizzano questa struttura, mentre l'istanza della base di dati corrisponde a tutto il suo contenuto, quindi all'insieme di record che andiamo ad inserire nella struttura.

Ovviamente l'insieme istanza è un qualcosa di dinamico che cambia, mentre lo schema una volta definito è una cosa rigida e rimane così com'è. L'istanza è qualcosa di dinamico che può variare nel tempo, in seguito ad inserimenti, cancellazioni, modifiche che si vanno a fare nella base di dati.

Un altro concetto è quello del dizionario dei dati, ovvero viene mantenuto un catalogo di tutte le strutture definite nel database con

le loro caratteristiche, quindi ad esempio è l'elenco di tutte le tabelle definite, i nomi delle colonne che le compongono ecc. Questo catalogo è strutturato secondo lo stesso modello logico quindi abbiamo la possibilità di vedere l'organizzazione presente sulle basi di dati che sono state create e con una strutturazione che è esattamente quella del modello relazionale, abbiamo una base di dati a livello di sistema che ci dà informazioni su tutte le basi di dati con cui abbiamo a che fare. Questo catalogo è accessibile utilizzando gli stessi strumenti con cui si accede poi normalmente ai dati di ogni base di dati e quindi si può vedere la struttura della base di dati.

Quando si parla di un sistema di gestione di base di dati si possono individuare dei livelli di astrazione con cui questo con cui il modello è stato progettato. Il primo livello di astrazione è quello logico quindi il livello logico è quello che fa riferimento al modello dei dati che si sta considerando quindi al modello relazionale, reticolare, a oggetti e quindi a questo livello abbiamo lo schema logico ovvero la descrizione della base di dati secondo il modello dei dati che stato scelto. Un esempio a questo livello quindi troviamo il modello relazionale con la descrizione della struttura delle tabelle che formano il base la base di dati, questo è quello che viene comunemente indicato come livello logico.

Il livello fisico è la rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione quindi qui si entra un po' più nel merito dell'organizzazione interna della base di dati. Ad esempio in questo livello abbiamo la descrizione degli indici che si costruiscono per le basi di dati delle strutture dati che sono necessarie per memorizzare questi indici.

L'ultimo livello è quello esterno che è quello che si vedono gli utenti poi che utilizzano la base di dati corrisponde quindi alla descrizione di parte della base di dati per mezzo di un modello logico e come vedremo nel modello relazionale esiste uno strumento che è la vista e che permette di rappresentare porzioni della base di dati e in modo da far vedere agli utenti porzioni diverse della base dei dati.

Il livello esterno potrebbe coincidere con il livello logico se non c'è nessuna vista e quindi le applicazioni che si interfacciano alla base di dati fanno vedere esattamente tutto, però in generale quello che potrà succedere è che invece a livello esterno un'applicazione vada a interfacciarsi soltanto su una parte della base di dati e questo è possibile tramite l'uso di viste. Questo è uno strumento che permette di ricreare un sottoinsieme del modello logico che in qualche modo a partire da quello iniziale. Questa organizzazione in livelli di astrazione nei sistemi di gestione di base di dati ha due caratteristiche importanti e che sono la prima l'indipendenza fisica.

Cosa si intende per indipendenza fisica? Questa indipendenza fa sì che sia possibile modificare certi dettagli a livello fisico della base di dati senza dover ad esempio modificare o aggiungere o togliere un indice senza per questo modificare il programma di interrogazione.

L'altra caratteristica importante è l'indipendenza logica che deriva dall'uso delle viste, quindi questo strumento permette di vedere porzioni della base di dati come se fossero a tutti gli effetti delle tabelle nel database. Questo carattere è importante perché se ho un'applicazione che interagisce con il database attraverso le viste è possibile modificare il modello logico della base di dati senza modificare l'applicazione purché le viste rimangano inalterate quindi questo è una caratteristica importante quando appunto si hanno necessità di modificare, di ristrutturare la base di dati a livello logico ma non si vogliono bonificare le applicazioni. Questo in alcuni casi non sempre è possibile grazie all'uso delle viste quindi alla possibilità quindi di ricreare in qualche modo a una vista del un sottoinsieme o ritrovare le tabelle che nella nuova organizzazione non sono più presenti tramite l'uso di questo strumento

Quindi immaginiamo di avere una base di dati con queste due tabelle, abbiamo il riferimento a dei corsi, quindi l'insegnamento, il docente e l'aula e in un'altra tabella invece abbiamo il riferimento alle aule quindi l'edificio e al piano, immaginiamo che siano tabelle reali quindi a ognuna di queste tabelle corrisponderà un file un archivio organizzato secondo il sistema di gestione di base di dati. Tramite lo strumento della vista ad esempio è possibile creare delle tabelle virtuali, ad esempio uno potrebbe creare una vista corsi-sedi che mette insieme le informazioni delle due tabelle precedenti, in questa nuova tabella c'è riferimento sia al corso che all'edificio e al piano in cui l'aula si trova. Questa tabella che vedete potrebbe anche non esistere e quindi non esistere un file nella base di dati che corrisponde esattamente a queste informazioni ma con lo strumento della vista è possibile creare degli oggetti di questo tipo e quindi anche un'eventuale ristrutturazione della base di dati a livello logico potrebbe non causare problemi a un'eventuale applicazione che si interfaccia la base di dati perché comunque abbiamo la possibilità di creare delle tabelle virtuali che fanno riferimento ai dati.

L'indipendenza fisica e logica sono due caratteristiche importanti nei sistemi di gestione di basi di dati ed è il motivo per cui dall'introduzione del modello relazionale negli anni 70 poi all'effettivo utilizzo dei sistemi di gestione di basi di dati siano passati poi diversi anni perché non è stato facile implementare queste caratteristiche.

Linguaggi per basi di dati: per gestire le informazioni e poi per interrogare una base di dati utilizzeremo il linguaggio SQL e questo linguaggio prevede la possibilità di utilizzare tutto un insieme di istruzioni di tipo DDL ovvero di istruzioni che permettono di definire lo schema logico e lo schema fisico, ma anche le viste quindi lo schema esterno, e anche di definire le eventuali autorizzazioni per l'accesso. Lo stesso linguaggio permette anche di definire invece le istruzioni che vengono dette di DML quindi le istruzioni che servono invece per la manipolazione dei dati e ovvero istruzioni che permettano inserimento, modifica, cancellazione, interrogazioni e che quindi operano sulla istanza della base di dati. I linguaggi delle basi di dati hanno questa doppia caratteristica quindi da una parte ci permettono di definire lo schema logico e fisico, e poi abbiamo anche la possibilità di manipolare i dati per operazioni di inserimento modifica cancellazione e per interrogazioni. Il linguaggio SQL offre tutte e due queste funzionalità ed è abbastanza semplice poi creare e utilizzare una base di dati con questo strumento.

SQL: un esempio di operazione DDL: un esempio di creazione di tabella

```
CREATE TABLE Orario (
    insegnamento CHAR(20),
    docente CHAR(20),
    aula CHAR(4),
    ora TIME)
```

Abbiamo l'istruzione create table, l'istruzione che permette di creare tabelle nel modello relazionale tramite il linguaggio SQL, in questo caso andiamo a creare la tabella che si chiama orario, e poi tra parentesi tonde specifichiamo un insieme di attributi specificando anche un tipo dell'attributo, quindi abbiamo l'insegnamento, il docente, l'aula che in questo caso sono stati tutti definiti di tipo Char, quindi delle stringhe di lunghezza variabile (nel senso che cambia a seconda dell'attributo), e poi abbiamo un attributo ora invece che in questo caso è stato definito di tipo time, quindi questo è un semplice esempio di utilizzo del linguaggio per la creazione di una tabella, e ovviamente una volta creata la tabella ci dovremo preoccupare di popolarla, quindi di inserire record nella tabella, eventualmente modificarli, cancellarli ecc.

Fare delle interrogazioni perché utilizzando il modello relazionale è possibile interrogare la base di dati e ricavare delle informazioni anche non banali. Abbiamo due tabelle che

CORSI		
INSEGNAMENTO	DOCENTE	AULA
Algoritmi	Paola Bianchi	A2
Analisi I	Mario Rossi	A1
Sistemi Informativi	Luigi Verdi	A3
Statistica	Nicola Bruni	A2

AULE		
NOME	EDIFICIO	PIANO
A1	Plesso Morgagni	I
A2	Ex Farmacologia	I
A3	Plesso Morgagni	II

Trovare i corsi tenuti in aule al I piano.

```
SELECT Insegnamento, Aula, Piano
FROM Aule, Corsi
WHERE Nome = Aula
AND Piano = 'I'
```

INSEGNAMENTO	AULA	PIANO
Algoritmi	A2	I
Analisi I	A1	I
Statistica	A2	I

tramite i linguaggi SQL si può andare a trovare i corsi tenuti in aule al primo piano. Per poter rispondere a questa interrogazione è necessario mettere insieme le informazioni che si trovano nelle due. Questa operazione si può fare tramite un'operazione di selezione, una select, ma soprattutto in questo caso la cosa importante che si deve fare è una congiunzione delle due tabelle, si devono congiungere le due tabelle in modo tale che valore dell'attributo aula nella tabella corsi corrisponda al nome dell'aula nella tabella delle aule, quindi vedete in questo esempio ho una selezione in cui qui

specifico quali sono gli attributi che poi voglio in risposta alla mia interrogazione, in questo caso sono l'insegnamento, l'aula e il piano, from che fa parte dell'istruzione ed elenca quali sono le tabelle che sono coinvolte nell'interrogazione, quindi in questo caso sono le aule e i corsi, where è la condizione che deve essere verificata e qui abbiamo una condizione di due tipi, perché ne abbiamo una vedete nome = aula è la condizione che ci permette di congiungere le due tabelle e poi invece l'altra parte della condizione è quella che fa riferimento al fatto che vogliamo le aule gli insegnamenti che si trovano al primo piano quindi La necessità di specificare qual è il piano di interesse. Con una selezione di questo tipo abbiamo una risposta sempre in forma tabellare che contiene gli insegnamenti che hanno questa caratteristica.

Gli utenti delle basi di dati sono possono essere di diversi tipi: l'amministratore della base di dati è quello che è responsabile della progettazione, del controllo e dell'amministrazione della base di dati, è quello che gestisce le autorizzazioni per l'accesso, è responsabile anche dell'affidabilità, ovvero di tutte le operazioni di backup e recovery. Un'altra tipologia di utenti è quella dei programmatore di applicazioni, le quali applicazioni devono accedere a base di dati. Poi ci sono gli utenti finali, quelli utilizzano la base dei dati interagendo tramite le applicazioni che accedono alla base di dati.

Modello relazionale-20220302 0801-1

Tre modelli logici tradizionali:

- gerarchico
- Reticolare
- relazionale.

Più recenti:

- a oggetti (poco diffuso)
- basato su XML

Gerarchico e reticolare: utilizzano riferimenti esplicativi (puntatori) fra record

Relazionale è basato su valori: anche i riferimenti fra dati in strutture (relazioni) diverse sono rappresentati per mezzo dei valori stessi

Il modello relazionale: Proposto da E. F. Codd nel 1970 per favorire l'indipendenza dei dati. Disponibile in DBMS reali nel 1981 (non è facile implementare l'indipendenza con efficienza e affidabilità). Si basa sul concetto matematico di relazione (con una variante). Le relazioni hanno naturale rappresentazione per mezzo di tabelle.

Relazione: tre accezioni

Relazione matematica: come nella teoria degli insiemi.

Relazione secondo il modello relazionale dei dati.

Relazione (dall'inglese relationship) che rappresenta una classe di fatti, nel modello Entity-Relationship; tradotto anche con associazione o correlazione.

Relazione matematica

D_1, \dots, D_n (n insiemi anche non distinti)

prodotto cartesiano $D_1 \times \dots \times D_n$: l'insieme di tutte le n -uple (d_1, \dots, d_n) tale che $d_1 \in D_1, \dots, d_n \in D_n$

relazione matematica r su $D_1 \times \dots \times D_n$: un sottoinsieme di $D_1 \times \dots \times D_n$

$r \subseteq D_1 \times \dots \times D_n$ D_1, \dots, D_n sono i domini della relazione

Relazione matematica: proprietà

Una relazione matematica è un insieme di n -uple ordinate (d_1, \dots, d_n) tali che $d_1 \in D_1, \dots, d_n \in D_n$. Una relazione è un insieme: non c'è ordinamento fra le n -uple. le n -uple sono distinte ciascuna n -upla è ordinata: l' i -esimo valore proviene dall' i -esimo dominio.

Negli insiemi i duplicati vengono rimossi quindi nel concetto di insiemi non ci possono essere duplicati.

Tabelle e relazioni

In una tabella che rappresenta una relazione l'ordinamento tra le righe è irrilevante l'ordinamento tra le colonne `e irrilevante. Una tabella rappresenta una relazione se le righe sono diverse fra loro le intestazioni delle colonne sono diverse tra loro i valori di ogni colonna sono fra loro omogenei.

Il modello è basato su valori

Il riferimenti fra dati in relazioni diverse sono rappresentati per mezzo di valori dei domini che compaiono nelle ennuple. Altri modelli (sia quelli storici, reticolare e gerarchico, sia quello a oggetti) prevedono riferimenti esplicativi, gestiti dal sistema.

Struttura basata su valori: vantaggi

Indipendenza dalle strutture fisiche che possono cambiare dinamicamente. Si rappresenta solo ciò che rilevante dal punto di vista dell'applicazione I dati sono portabili più facilmente da un sistema ad un altro. I puntatori sono direzionali.

Definizioni

Schema di relazione: un nome R con un insieme di attributi $X = \{A_1, \dots, A_n\} : R(X) = R(A_1, \dots, A_n)$; n `e il grado della relazione.

Schema di base di dati: insieme di schemi di relazione: $R = \{R_1(X_1), \dots, R_k(X_k)\}$

Una ennupla su un insieme di attributi X `e una funzione che associa a ciascun attributo A in X un valore del dominio di A; $t[A]$ denota il valore della ennupla t sull'attributo A;

istanza di relazione su uno schema $R(X)$: insieme r di ennuple su X; $|r|$ `e la cardinalità dell'istanza di relazione. istanza di base di dati su uno schema $R = \{R_1(X_1), \dots, R_n(X_n)\}$: insieme di relazioni $r = \{r_1, \dots, r_n\}$ (con r_i relazione su R_i)

Informazione incompleta

Il modello relazionale impone ai dati una struttura rigida: le informazioni sono rappresentate per mezzo di ennuple; solo alcuni formati di ennuple sono ammessi, quelli che corrispondono agli schemi di relazione. I dati disponibili possono non corrispondere al formato previsto.

Informazione incompleta: soluzioni?

Non conviene (anche se spesso si fa) usare valori del dominio (0, stringa nulla, 99, ...): potrebbero non esistere valori non utilizzati; valori non utilizzati potrebbero diventare significativi; in fase di utilizzo (nei programmi) sarebbe necessario ogni volta tener conto del significato di questi valori.

Informazione incompleta nel modello relazionale

Tecnica rudimentale ma efficace: si introduce il concetto di valore nullo: denota l'assenza di un valore del dominio (ma non `e un valore del dominio); $t[A]$, per ogni attributo A, `e un valore del dominio $\text{dom}(A)$ oppure il valore nullo (che indichiamo qui con NULL). Si possono (e debbono) impostare restrizioni sulla presenza di valori nulli

Tipi di valore nullo

(Almeno) tre casi differenti: valore sconosciuto valore inesistente valore senza informazione I DBMS non distinguono i tipi di valore nullo.

Vincoli di integrità

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

Proprietà che deve essere soddisfatta dalle istanze che rappresentano informazioni corrette per l'applicazione.

Un vincolo `e una funzione booleana (un predicato): associa ad ogni istanza il valore vero o falso.

Vincoli di integrità, perché?

Descrizione più accurata della realtà. Contributo alla qualità dei dati. Utili nella progettazione (vedremo). Usati dai DBMS nella esecuzione delle interrogazioni.

Vincoli di integrità, nota

Alcuni tipi di vincoli (ma non tutti) sono supportati dai DBMS: possiamo quindi specificare vincoli di tali tipi nella nostra base di dati e il DBMS ne impedisce la violazione. Per i vincoli non supportati, la responsabilità della verifica è dell'utente o del programmatore.

Tipi di vincoli

- Vincoli intrarelazionali: vincoli su valori (o di dominio); vincoli di ennupla.
- Vincoli interrelazionali.

Vincoli di ennupla

Esprimono condizioni sui valori di ciascuna ennupla, indipendentemente dalle altre ennupla.

Caso particolare: Vincoli di dominio: coinvolgono un solo attributo.

Sintassi ed esempi

Una possibile sintassi: espressione booleana di atomi che confrontano valori di attributo o espressioni aritmetiche su di essi

Vincolo di dominio: $(Voto \geq 18) \text{ AND } (Voto \leq 30)$

Vincolo di tupla: $\text{NOT}(Lode = \text{si}) \text{ OR } (Voto = 30)$ (la lode è ammessa solo se il voto è 30)

Il concetto di chiave

Insieme di attributi che identificano le ennupla di una relazione. Formalmente: un insieme K di attributi è superchiave per r se r non contiene due ennupla distinte t1 e t2 con $t1[K] = t2[K]$ K è chiave per r se è una superchiave minimale per r (cioè non contiene un'altra superchiave).

Vincoli, schemi e istanze

I vincoli corrispondono a proprietà del mondo reale modellato dalla base di dati. Interessano a livello di schema (con riferimento cioè a tutte le istanze). Ad uno schema associamo un insieme di vincoli e consideriamo corrette (valide, ammissibili) le istanze che soddisfano tutti i vincoli. Un'istanza può soddisfare altri vincoli (per caso).

Esistenza delle chiavi

Una relazione non può contenere ennupla distinte ma uguali. Ogni relazione ha come superchiave l'insieme degli attributi su cui è definita. E quindi ha (almeno) una chiave.

Importanza delle chiavi

L'esistenza delle chiavi garantisce l'accessibilità a ciascun dato della base di dati. Le chiavi permettono di correlare i dati in relazioni diverse: il modello relazionale è basato su valori.

Chiavi e valori nulli

In presenza di valori nulli, i valori della chiave non permettono: di identificare le ennupla; di realizzare facilmente i riferimenti da altre relazioni.

La presenza di valori nulli nelle chiavi deve essere limitata.

Chiave primaria

Chiave su cui non sono ammessi nulli. Notazione: sottolineatura

Vincoli interrelazionali:

integrità referenziale Informazioni in relazioni diverse sono correlate attraverso valori comuni. In particolare, valori delle chiavi (primarie). Le correlazioni debbono essere coerenti.

Vincolo di integrità referenziale

Un vincolo di integrità referenziale, foreign key, fra gli attributi X di una relazione R1 e un'altra relazione R2 impone ai valori su X in R1 di comparire come valori della chiave primaria di R2. Esempi di vincoli di integrità referenziale fra: l'attributo Vigile della relazione INFRAZIONI e la relazione AGENTI; gli attributi Prov e Numero di INFRAZIONI e la relazione AUTO.

Vincoli di integrità referenziale: commenti

Giocano un ruolo fondamentale nel concetto di modello basato su valori. In presenza di valori nulli i vincoli possono essere resi meno restrittivi. Sono possibili meccanismi per il supporto alla loro gestione (azioni compensative a seguito di violazioni). Attenzione ai vincoli su più attributi.

Azioni compensative

Esempio: viene eliminata una ennupla causando una violazione. Comportamento standard: rifiuto dell'operazione. Azioni compensative: eliminazione in cascata; introduzione di valori nulli.

Vincoli multipli su più attributi

Vincoli di integrità referenziale fra: gli attributi ProvA e NumeroA di INCIDENTI e la relazione AUTO; gli attributi ProvB e NumeroB di INCIDENTI e la relazione AUTO. L'ordine degli attributi `e significativo.

I modelli gerarchico e reticolare fanno riferimenti esplicativi tra i record ovvero per collegare le informazioni di tipologie diverse utilizzano proprio dei riferimenti fisici, dei puntatori, che permettono di collegare le varie informazioni. La forza invece del modello relazionale è che si basa su valori. I riferimenti tra i dati che si trovano nelle strutture che costituiscono la base di dati, quindi nelle tabelle, di fatto che vanno a costituire la base dei dati e sono ottenuti per mezzo di valori, quindi non per mezzo di riferimenti a strutture fisiche, ma tramite valori.

STUDENTI			
Matricola	Cognome	Nome	Nascita
6554	Rossi	Mario	05/12/1995
8765	Neri	Paolo	03/11/1994
3456	Rossi	Maria	01/02/1992
9283	Verdi	Luisa	12/11/1995

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

CORSI		
Codice	Titolo	Docente
01	Analisi	Mario
02	Algoritmi	Bruni
04	Algoritmi	Verdi

Abbiamo la tabella studenti con quattro attributi la matricola, il cognome, il nome e la data di nascita e poi abbiamo una tabella degli esami, in cui compaiono gli attributi studente, voto e il codice del corso, che evidentemente corrisponde all'esame sostenuto e infine abbiamo una tabella dei corsi contente i dettagli dei corsi, quindi il codice, il titolo e il docente del corso. Queste tre tabelle perché siano utilizzabili è necessario poterle collegare. In questo esempio ci sono dei riferimenti a dei numeri di matricola, che poi compaiono anche nella tabella degli esami, così come il codice del corso che compare nella tabella dei corsi e nella tabella degli esami. Tramite questi riferimenti incrociati è possibile congiungere

queste tabelle e ottenere delle informazioni che tengono conto sia degli studenti che degli esami che dei corsi.

Il modello relazionale è stato introdotto nel 1970 da Codd. I sistemi di gestione di basi di dati che utilizzavano questo modello dei dati sono stati resi disponibili solo diversi anni dopo, nel 1981, perché l'implementazione di questi modelli non è stata semplice. L'indipendenza fisica e logica, caratteristiche di questi sistemi, non sono di facile realizzazione.

Il modello relazionale si basa su un concetto matematico, che è quello di relazione (con una piccola variante). Nel contesto di questo corso parleremo di relazione da tre punti di vista un po' diversi. Prima di tutto parleremo di relazione matematica che è quella da cui parte tutto quindi il concetto base del modello, ed è un concetto che nasce nella teoria degli insiemi, poi parleremo invece di relazione secondo il modello relazionale dei dati, che è molto simile al concetto precedente di relazioni matematiche ma con una piccola variante, che è utile per la gestione dei dati. Un terzo concetto è quello di relazione, dall'inglese relationship, che rappresenta un'associazione tra oggetti, quindi ne parleremo quando introdurremo la progettazione concettuale di una base di dati, e parleremo del modello entità relazione (in questi casi utilizzerò il termine associazione per non fare confusione con il termine relazione che viene tipicamente utilizzato per definire le tabelle del modello relazionale).

- $D_1 = \{a, b\}$, $D_2 = \{x, y, z\}$
- prodotto cartesiano $D_1 \times D_2$

a	x
a	y
a	z
b	x
b	y
b	z

- una relazione $r \subseteq D_1 \times D_2$

a	x
a	z
b	y

ciascun dominio della relazione si associa un nome unico nella tabella, che descrive il ruolo di quell'attributo. In questo modo si passa da una struttura posizionale ad una struttura che non è più posizionale, perché il nome che io ho associato a quell'attributo mi permette di capire il significato di quell'informazione indipendentemente dalla sua posizione. Questa è la prima differenza tra il concetto di relazione matematica, che è semplicemente un sottoinsieme di un prodotto cartesiano. Mentre nel modello relazionale si introduce questa informazione aggiuntiva, ovvero il nome degli attributi, per passare quindi da una notazione di tipo posizionale ad una di tipo non posizionale. Questa è la distinzione fra questi due concetti. Nel modello relazionale, la relazione si rappresenta in modo naturale come una tabella. Una tabella per presentare una relazione però deve avere delle caratteristiche (non è che ciascuna tabella corrisponde a una relazione). In una tabella che rappresenta una relazione le righe devono essere diverse, non possono esserci duplicati, le intestazioni delle colonne devono essere diverse, i valori che andiamo che compaiono in corrispondenza di ogni colonna devono essere omogenei quindi devono appartenere alla stessa tipologia.

STUDENTI			
Matricola	Cognome	Nome	Data
6554	Rossi	Mario	05/12/1995
8765	Neri	Paolo	03/11/1994
3456	Rossi	Maria	01/02/1992
9283	Verdi	Luisa	12/11/1995

ESAMI			CORSI		
Studente	Voto	Corso	Codice	Titolo	Docente
3456	30	04	01	Analisi	Mario
3456	24	02	02	Algoritmi	Bruni
9283	28	01			

STUDENTI			
Matricola	Cognome	Nome	Data
6554	Rossi	Mario	05/12/1995
8765	Neri	Paolo	03/11/1994
3456	Rossi	Maria	01/02/1992
9283	Verdi	Luisa	12/11/1995

ESAMI			CORSI		
Studente	Voto	Corso	Codice	Titolo	Docente
↔	30	↔	01	Analisi	Mario
↔	24	↔	02	Algoritmi	Bruni
↔	28	↔			
↔	26	↔	04	Algoritmi	Verdi

che cos'è innanzi tutta una relazione matematica? un esempio: consideriamo due insiemi, un insieme costituito da due elementi {a,b} e l'altro costituito da tre elementi {x,y,z}. possiamo definire il prodotto cartesiano fra questi due insiemi. che cos'è il prodotto cartesiano? è costituito da un insieme di coppie, vado a prendere un elemento del primo insieme e accoppiandolo con ciascuno degli elementi del secondo quindi vedete a-x, a-y, a-z, poi così via con l'elemento b, quindi questo è quello che si definisce prodotto cartesiano. una relazione matematica è semplicemente un sottoinsieme di un prodotto cartesiano.

Nella relazione matematica abbiamo un'interpretazione delle n-uple che deve tener conto della loro posizione, è una struttura di tipo posizionale. Per evitare qualsiasi tipo di ambiguità è sicuramente più chiaro associare a ognuna delle colonne, che corrispondono ai vari termini della relazione, un attributo, ovvero un nome, quindi a

ciascun dominio della relazione si associa un nome unico nella tabella, che descrive il ruolo di quell'attributo. In questo modo si passa da una struttura posizionale ad una struttura che non è più posizionale, perché il nome che io ho associato a quell'attributo mi permette di capire il significato di quell'informazione indipendentemente dalla sua posizione. Questa è la prima differenza tra il concetto di relazione matematica, che è semplicemente un sottoinsieme di un prodotto cartesiano. Mentre nel modello relazionale si introduce questa informazione aggiuntiva, ovvero il nome degli attributi, per passare quindi da una notazione di tipo posizionale ad una di tipo non posizionale. Questa è la distinzione fra questi due concetti. Nel modello relazionale, la relazione si rappresenta in modo naturale come una tabella. Una tabella per presentare una relazione però deve avere delle caratteristiche (non è che ciascuna tabella corrisponde a una relazione). In una tabella che rappresenta una relazione le righe devono essere diverse, non possono esserci duplicati, le intestazioni delle colonne devono essere diverse, i valori che andiamo che compaiono in corrispondenza di ogni colonna devono essere omogenei quindi devono appartenere alla stessa tipologia.

Perché il modello viene detto basato su valori? Nelle relazioni che sono rappresentate da tabelle possono essere messe in relazione tra di loro tramite i valori che compaiono nelle ennuple. A differenza di altri modelli come il reticolare e gerarchico in cui i riferimenti sono invece esplicativi e gestiti poi dal sistema.

Tornando all'esempio iniziale: queste tre tabelle rappresentano delle relazioni. Se io voglio sapere quali esami ha fatto Rossi Maria dovrò andare a vedere qual è la matricola di Maria, la matricola la trovo nella tabella degli esami, vedo che ha preso 30 al corso con codice quattro, e poi vado a scoprire che questo corso è il corso di algoritmi tenuto dal docente verdi. I riferimenti tra le varie ennuple sono fatti attraverso dei valori.

Nei modelli gerarchico e reticolare questi riferimenti sarebbero fatti attraverso dei puntatori, quindi questi valori delle ennuple negli studenti dovrebbero essere collegate fisicamente. È chiaro che questo poi complica la gestione delle informazioni.

In particolare avere una struttura basata su valori come quella del modello relazionale ha come vantaggio l'indipendenza delle strutture fisiche, ovvero le strutture fisiche che stanno dietro al

modello possono anche cambiare perché quello che conta è che ci permette di lavorare sui dati e su tutti i legami logici tra le varie tabelle sono i valori, che non dipendono da strutture fisiche. Una base di dati realizzata con un modello di questo tipo è facilmente portabile da un sistema all'altro perché non si fa riferimento esplicito a strutture fisiche ma soltanto al modello logico dei dati.

Sunto: la relazione del modello relazionale è un una piccola variante rispetto al concetto di relazione matematica e la caratteristica principale del modello relazionale è che è basato su valori.

Cos'è lo schema di relazione? Lo schema di relazione è in pratica la forma della tabella, è definito dal nome della tabella e da un insieme di attributi, il numero di attributi viene detto il grado della relazione.

Uno schema di base di dati è un insieme di schemi di relazione, non è altro che un insieme che contiene un numero k di schemi di relazione, quindi si sta definendo qual è la forma delle tabelle e di conseguenza qual è la struttura della base di dati.

Istanza di relazione è un insieme di record definiti sull'insieme X degli attributi e la cardinalità di questo insieme di record è la cardinalità dell'istanza della relazione, quindi l'istanza corrisponde all'insieme di record che vado a inserire nella tabella

Istanza di basi di dati con riferimento invece all'insieme delle istanze di relazioni che corrispondono ai vari schemi. L'importante è distinguere quindi tra lo schema di relazione che rappresenta la forma della tabella e l'istanza di basi di dati che invece rappresenta l'insieme delle ennuple che costituiscono una particolare istanza della relazione

Relazioni su singoli attributi

STUDENTI			
Matricola	Cognome	Nome	Data
6554	Rossi	Mario	05/12/1995
8765	Neri	Paolo	03/11/1994
3456	Rossi	Maria	01/02/1992
9283	Verdi	Luisa	12/11/1995

LAVORATORI	
Matricola	
6554	
9283	

stati ordinati e il loro costo e anche il costo totale. Cerchiamo di capire come dovremmo organizzare una base di dati di tipo relazionale per poter contenere queste informazioni allora quindi la parte che adesso vedete evidenziata rossa è la parte variabile della fattura mentre appunto le altre sono parti costanti. Allora una prima cosa che può venire in mente di fare è organizzare quel informazioni in modo tabellare, anche se questa ancora non è la forma che ci piace, perché abbiamo una tabella dentro una tabella, quindi non è esattamente la forma richiesta nel modello relazionale. Qui abbiamo semplicemente

Da Pippo		
via XYZ 1, Firenze		
Ricevuta fiscale		
100	del	18-2-2014
3	Coperti	3,00
2	Antipasti	6,20
3	Primi	12,00
2	Bistecche	18,00
	Totale	39,20

Da Pippo		
via XYZ 1, Firenze		
Ricevuta fiscale		
200	del	20-2-2014
2	Coperti	2,00
2	Antipasti	7,00
2	Primi	8,00
2	Orate	20,00
2	Caffè	2,00
	Totale	39,00

Esempio: una tabella studenti e una tabella lavoratori. Abbiamo una tabella definita su un singolo attributo. L'idea è di andare a selezionare in qualche modo gli studenti che hanno anche la caratteristica di essere lavoratori, quindi possono esserci delle situazioni in cui anche tabelle costituite da una singola colonna possono essere di interesse.

Cercando di capire come da un problema reale si possa passare a definire le informazioni che sono di interesse, utilizzando il modello relazionale allora qui abbiamo un esempio di ricevute di un ristorante e vedete c'è la parte in azzurro contiene riferimento a ristorante, il numero della ricevuta e la data e poi abbiamo invece le informazioni su quello che è stato ordinato, quindi sul numero di coperti, sul numero di piatti che sono

evidenziato in un'unica tabella qual è appunto il numero della ricevuta, la data, il totale è però qui abbiamo la quantità la descrizione l'importo di ciascun di ciascuno ordine e e evidenziati insieme questo ancora non rappresenta una tabella del modello relazionale no perché abbiamo appunto una è come se avessimo una tabella annidata dentro un'altra quindi per poter arrivare ad avere le informazioni invece non formato ti che quello relazionale e possiamo fare organizzare le informazioni in questo modo quindi abbiamo vedere che possiamo inserire i riferimenti alla numero della ricevuta la data al totale in una tabella ricevute ad esempio no con attributi numero data e totale e poi il dettaglio il dettaglio della ricevuta lo troviamo invece in un'altra tabella che

appunto vi è stata chiamata dettaglio in cui compare vedete il riferimento un valore e lo stesso valore 100 nel compare nella tabella ricevute quindi il collegamento può essere fatto facilmente con poi i riferimenti alla quantità la descrizione e all'importo quindi questo potrebbe essere una prima soluzione no quindi qui abbiamo le informazioni che sono contenute nelle a definire due fatture però organizzate secondo il modello relazionale quindi pensate ad un programma il ristorante che tutto memorizza le informazioni giardia che riceve gli ordini e poi ci sarà un'applicazione che le stampa andando a prendere le informazioni da queste da queste tabelle e ecco la domanda che vi faccio e con un'organizzazione di questo tipo una eventuale applicazione che poi va a stampare le ricevute riuscirebbe a stampare la ricevuta in questo modo cioè avendo le due tabelle così come le ho organizzate riusciamo a stampare e le ricevute ricostruendolo esattamente come se le avessi scritte a mano quindi qui vedete qui mi immaginiamo che insomma queste siano state scritte a mano quindi prima ho scritto il numero di coperti poi gli antipasti primi e poi il secondo e così via no nell'organizzazione che io ho dato beh manca un dettaglio per poter ricostruire esattamente la ricevuta così come ve l'ho presentata perché qui vedete io sto dicendo che nella ricevuta numero 100 compaiono questi elementi ma non sto dicendo in che ordine compaiono perché abbiamo detto le cupole l'ordine con cui i record compare in una tabella non è significativo il fatto che adesso siano che l'ordine adesso sia esattamente quello che compare nella ricevuta e è casuale nel modello relazionale non c'è un ordine predefinito tra i record quindi se voglio invece mantenere quell'ordine e ho bisogno e ho bisogno di utilizzare un'altra informazione quindi ho bisogno di un'altra attributo di un altro attributo che mi tenga conto della riga della fattura del momento in cui quell'elemento è stato ordinato no questo mi permette di ricostruire esattamente la fattura tenendo conto anche appunto del momento in cui 1 1 certo piatto è stato ordinato ora questo ovviamente è un esempio insomma e ci si può arrivare a questa a questa conclusione insomma ragionandoci insomma essendo un esempio semplice poi ovviamente non è così che si fa va bene per se si ha bisogno di definire una base di dati e è necessario fare una progettazione ma quello di tensione concettuale.

Il primo problema che deve essere affrontato è quello delle informazioni incomplete. Il modello relazionale ha una struttura rigida perché i record hanno numero definito di campi e questi devono essere tutti ovviamente presenti. Quando si vanno a

inserire i record nella base di dati a volte però qualche informazione non ce l'abbiamo. Esempio: immaginate di avere questa tabella presidente, con i nomi di presidenti degli stati uniti, c'è un attributo nome, un attributo secondo nome e un cognome. Per il primo presidente compare il secondo nome mentre per gli altri non compare, è chiaro che ci possono essere delle situazioni in cui ho bisogno di rappresentare un'informazione che per alcuni record è presente e per altri non è presente, oppure in altri casi potrei proprio non sapere se è presente oppure no, quindi per poter gestire questa mancanza di informazione come posso fare? Una soluzione, che non è però utilizzata, potrebbe essere quella di utilizzare dei valori particolari del dominio, quindi ad esempio in un attributo di tipo numerico di indicare con il valore 0 o 99 l'assenza di informazione, oppure se ho delle stringhe, utilizzare la stringa nulla per indicare che l'informazione manca. Questa è una soluzione che andrebbe evitata perché i valori utilizzati per rappresentare questi casi particolari poi potrebbero invece diventare valori effettivamente presenti nelle ennuple cioè potrebbero diventare significativi. La soluzione che viene utilizzata è quella di introdurre il concetto di valore nullo, quindi un valore che indica l'assenza di un valore nel dominio, quindi in corrispondenza di un certo attributo, il record può assumere un qualsiasi valore del dominio oppure il valore nullo, se l'informazione non la conosciamo. Tipicamente il valore nullo si indica con null. Bisogna imporre delle restrizioni sull'uso di valori nulli perché altrimenti le cose diventano complicate e quindi come vi dicevo ci possono essere almeno tre casi di valori nulli: valore sconosciuto, valore inesistente oppure valore senza informazioni. Quando compare il valore nullo non si fa distinzione tra questi casi diversi.

STUDENTI			
Matricola	Cognome	Nome	Data
6554	Rossi	Mario	05/12/1995
8765	Neri	Paolo	03/11/1994
3456	Rossi	Maria	01/02/1992
NULL	Verdi	Luisa	12/11/1995

ESAMI		
Studente	Voto	Corso
NULL	30	NULL
NULL	24	02
9283	28	01
6554	26	01

CORSI		
Codice	Titolo	Docente
01	Analisi	Mario
02	NULL	NULL
04	Algoritmi	Verdi

che lo studente ha sostenuto, senza questo valore questo incrocio di informazioni non riesco a farlo. Nella tabella degli esami ho il riferimento a due studenti senza sapere quali sono le loro matricole, quindi è chiaro che non riesco a risalire allo studente. Nel primo caso ho un doppio problema perché non solo non riesco a risalire allo studente ma addirittura non so a che corso ha preso 30. Nella tabella dei corsi, il titolo e il docente sono tutte e due nulli, anche questa è una ennupla completamente priva di significato. Per evitare una base di dati con valori nulli che non hanno nessun significato e che poi non riesco a gestire è necessario introdurre nella base di dati quelli che si chiamano vincoli di integrità. Possono esistere nelle istanze di base di dati che anche se sono sintatticamente corrette però non rappresentano informazioni. Devo aggiungere alla definizione delle mie tabelle dei vincoli dei vincoli di integrità. Questi vincoli permettono di risolvere alcuni dei problemi sui valori nulli che abbiamo visto ma sono in generale utili per diversi altri problemi.

ESAMI			
Studente	Voto	Lode	Corso
6554	32		01
6554	30	si	02
8765	27	si	03
5432	24		04

STUDENTI		
Matricola	Cognome	Nome
6554	Rossi	Mario
8765	Neri	Paolo
3456	Rossi	Maria
3456	Verdi	Luisa

Esempio: la tabella esami in cui questa volta ho aggiunto anche una colonna lode e cerchiamo di capire perché questa base di dati è scorretta. Compare in corrispondenza del primo record un valore 32, nel sistema universitario italiano non può esistere, i voti sono fino al 30. L'altro problema è che uno studente ha preso 27 e anche la lode, ed è chiaro che non va bene. Un altro problema è che non è presente il riferimento allo studente, che qui è evidenziato in verde, quello con matricola 5432, il problema è che questo studente non compare nella tabella successiva, quindi ho un riferimento a uno studente che non è stato inserito nella base di dati. Un altro problema che è presente in questo esempio è il fatto che gli ultimi due studenti Rossi Maria e Verdi Luisa hanno lo stesso numero di matricola, e anche questo non va bene. Questo è un esempio di base di dati che dal punto di vista sintattico soddisfa le caratteristiche che abbiamo dato però non dal punto di vista semantico. È sbagliata perché o non contiene informazioni o sono scorrette. Devo impedire quando definisco una base di dati che si verificano queste situazioni, e quindi che cos'è un vincolo di integrità? Un vincolo di integrità è una proprietà che deve essere soddisfatta dalle istanze della base di dati, in maniera un pochino più formale è un predicato che associa ad ogni istanza della base di dati e che deve essere verificato dalle ennuple. I vincoli permettono di fare una descrizione più accurata della realtà contribuendo quindi alla qualità dei dati e sono importanti nell'esecuzione delle interrogazioni. Introdurre un vincolo vuol dire che poi se io vado ad inserire nella base di dati una ennupla che non soddisfa quel vincolo, il sistema di gestione di base di dati deve impedirmi di inserire quell'oggetto perché viene violato un vincolo, quindi inserire dei vincoli vuol dire che poi il sistema in qualche modo dovrà reagire a questa violazione.

situazioni, e quindi che cos'è un vincolo di integrità? Un vincolo di integrità è una proprietà che deve essere soddisfatta dalle istanze della base di dati, in maniera un pochino più formale è un predicato che associa ad ogni istanza della base di dati e che deve essere verificato dalle ennuple. I vincoli permettono di fare una descrizione più accurata della realtà contribuendo quindi alla qualità dei dati e sono importanti nell'esecuzione delle interrogazioni. Introdurre un vincolo vuol dire che poi se io vado ad inserire nella base di dati una ennupla che non soddisfa quel vincolo, il sistema di gestione di base di dati deve impedirmi di inserire quell'oggetto perché viene violato un vincolo, quindi inserire dei vincoli vuol dire che poi il sistema in qualche modo dovrà reagire a questa violazione.

Esistono due tipi di vincoli, i vincoli intrarelazionali e i vincoli interrelazionali.

I vincoli intra relazionali sono quelli che fanno riferimento ad una singola tabella e che possono coinvolgere o valori oppure intere ennuple.

I vincoli interrelazionali invece sono quelli forse più importanti di tutti perché sono quelli proprio che ci permettono di correlare le informazioni che si trovano in tabelle distinte.

Il concetto di identificazione di ennupla: in una tabella del modello relazionale è importante poter distinguere ogni record da un altro. In questo esempio, la matricola permette di fare questa operazione, perché non ci sono due studenti che hanno lo stesso valore di matricola. Se andiamo a vedere con attenzione non è l'unico attributo che permette di fare questa distinzione, perché se andiamo a considerare il cognome e il nome e la data di nascita e vediamo che in questa istanza non ci sono due studenti che hanno queste tre stesse caratteristiche uguali. I primi due record corrispondono tutte e due Rossi Mario, però poi la data di nascita è diversa. È importante che in ogni tabella sia possibile identificare in maniera univoca un record dall'altro, ma non è detto che ci sia un unico modo per fare questa identificazione.

Questo mi permette di introdurre il concetto di chiave. Una chiave è un insieme di attributi che mi permettono di identificare i record della relazione in maniera univoca, quindi ogni record è identificato da un insieme di attributi che viene definito chiave della relazione. È bene fare distinzione fra i due concetti che sono la super chiave e la chiave, una chiave è quella che si chiama una super chiave minimale, cioè, per essere una chiave, un insieme di attributi deve essere l'insieme minimo che ha questa caratteristica, cioè se io tolgo un attributo dall'insieme perdo questa caratteristica. Una chiave non deve contenere un'altra super chiave.

Esempio: nella tabella degli studenti che abbiamo già visto, matricola è sicuramente una chiave, perché ha soltanto un attributo e quindi non posso toglierne nessuno, quindi è sia super chiave e che chiave, e quindi è una super chiave perché mi permette di distinguere in maniera univoca ogni studente ma è anche chiave perché essendo appunto contenendo solo attributo è sicuramente minimale. La terna cognome, nome e nascita in questa istanza rappresenta un'altra chiave perché ho tre attributi che mi permettono di distinguere ogni record dall'altro, quindi la terna è super chiave. Per dimostrare che in questo caso la terna rappresenta una chiave, ovvero che è una super chiave minimale, quello che devo fare è controllare cosa succede se provo a togliere uno di questi tre attributi, e se io provo ad eliminare ad esempio la nascita da questa terna vedo che ci sono due studenti che hanno lo stesso cognome e nome, quindi perdo la caratteristica che sto cercando, se invece tolgo il riferimento al nome allo stesso modo ho due studenti che hanno lo stesso cognome e la stessa data di nascita, e infine se provo a togliere il cognome ho ancora due studenti che hanno lo stesso nome e la stessa data di nascita. Quindi la terna cognome, nome e nascita è una chiave, perché è una super chiave, ma è anche minimale. È importante che il concetto di chiave mi permetta di distinguere un record all'altro, senza che situazioni particolari possano creare problemi. Quindi, anche se è raro che due studenti abbiano lo stesso nome e la stessa data di nascita, potrebbe capitare, quindi è chiaro che tra le due chiavi che abbiamo visto, la matricola è sicuramente da preferire, difatti questo è un codice che sicuramente posso generare in maniera diversa per ogni studente. Quindi bisogna far distinzione tra quelle che risultano chiavi casualmente nell'istanza e quelle che invece si prestano ad essere definite come tali. Quando decidiamo quale insieme di attributi costituisce la chiave della relazione, in quel momento, noi stiamo introducendo un vincolo di integrità nella tabella, ciò significa che quando proviamo ad inserire due record che hanno nello stesso valore il sistema ci impedirà di fare questo inserimento. Quindi tutte le volte che inseriamo una chiave nella tabella stiamo introducendo un vincolo di integrità nella tabella. Quando si parla di relazione abbiamo a che fare con un insieme e quindi in una relazione non possono esserci record distinti che sono uguali, cioè non posso avere due record che hanno tutti gli attributi uguali. Questo è importante perché ci garantisce che qualsiasi sia la mia relazione, una chiave esiste sempre, ed è costituita dalla totalità degli attributi. Quindi se io prendo tutti gli attributi della tabella quella sicuramente rappresenta una super chiave per la relazione e questo mi garantisce che sia sempre possibile avere una chiave di una tabella secondo il modello relazionale.

I vincoli d'integrità tra tabelle diverse: le chiavi in una tabella sono importanti perché sono quelle che poi ci permettono di correlare i dati che si trovano in relazioni diverse.

Un altro concetto che è importante chiarire: abbiamo già visto nell'esempio degli studenti che c'erano due record che avevano valore nullo in corrispondenza della matricola, è chiaro che se ho un valore nullo in corrispondenza di attributi che fanno parte della chiave, io perdo la caratteristica della chiave, cioè quella di poter distinguere un record dall'altro, quindi è necessario aggiungere dei vincoli d'integrità. È necessario imporre che sulle chiavi non ci possano essere dei valori nulli. In altro esempio con l'istanza leggermente cambiata, se l'attributo matricola Assume valore nullo io non riesco più a distinguere uno studente dall'altro e in questo caso ho anche valori nulli in corrispondenza degli altri attributi che formano una super chiave, allora lo studente che sto evidenziando non riesco a distinguergli né tramite la matricola né tramite la terna cognome,

nome e corso e data di nascita e così come nel caso dei primi due studenti non riesco a capire se si tratta della stessa persona oppure no, allora quindi si passa dal concetto di super chiave e chiave a quello di chiave primaria.

Che cos'è una chiave primaria? La chiave primaria è una chiave su cui non sono ammessi valori nulli, quindi abbiamo un ulteriore vincolo e la notazione si utilizza per indicare un insieme di attributi che formano la chiave primaria è la sottolineatura. Questa notazione ci indica che quello non solo è la chiave della tabella ma che su quell'insieme di attributi non possiamo ammettere valori nulli, quindi nei sistemi di gestione di basi di dati si possono fare due cose, cioè si possono definire degli attributi in modo che questi formano una chiave primaria o semplicemente una chiave. Quindi nel caso di chiave primaria ovviamente non solo il sistema impedirà l'inserimento di record che hanno lo stesso valore in corrispondenza della matricola ma impedirà anche l'inserimento di record che hanno valore nullo.

I vincoli, che sono forse quelli più importanti, perché sono quelli che ci permettono di correlare informazioni che si trovano in relazioni diverse. Ho due relazioni, una dei una tabella infrazioni che contiene riferimenti ad infrazioni al codice della strada, ed è composta dall'ora, la data, l'agente, l'articolo che è stato violato, la notazione della targa vecchio stile con provincia e numero, poi abbiamo la tabella degli agenti con le informazioni degli agenti: la matricola, il codice fiscale, il cognome e il nome. È chiaro che le informazioni che sono presenti nella tabella delle infrazioni e in particolare quelle in riferimento all'agente hanno senso solo se ogni valore che compare nella colonna agente lo ritrovo nella tabella degli agenti in corrispondenza dell'attributo matricola, quindi non può esistere un valore nella colonna agenti che non sia presente nella tabella agente in corrispondenza della matricola, quindi questo è qualcosa che devo imporre nel sistema il sistema di gestione di basi di dati. Mi deve dare lo strumento perché io possa vincolare le entità della tabella infrazione a quelle della tabella agente in modo che tutte le volte che io introduco una nuova infrazione l'agente faccia riferimento ad un agente effettivamente presente. Quindi questo è il concetto di vincolo di integrità referenziale. Nella tabella delle infrazioni è presente anche un riferimento a una targa di auto, di cui ho informazione in una terza tabella, quella delle auto, e anche qui quello che deve succedere è che la targa dev'essere tra quelle che compaiono nella tabella delle auto e questo deve valere

per tutte per tutte le infrazioni, quindi la tabella delle infrazioni è vincolata in due modi, sia con la tabella degli agenti, perché deve il codice essere quello di un agente che è presente nella tabella, sia con la tabella delle auto. Dal punto di vista delle infrazioni quindi è necessario che ogni targa che compaia sia tra le auto che sono elencate, non vale però il viceversa: ci potrebbe essere un'auto, come quella di Verdi Piero, che è memorizzata nella tabella delle auto ma che non è coinvolta al momento in nessuna infrazione. Quindi in una direzione è importante che ci sia il vincolo ma in questo caso non è necessario che ci sia nella tabella delle auto, in cui posso inserire anche record che non sono coinvolti in infrazioni.

Modello relazionale-20220303 0935-1

I vincoli intra-relazionali si introducono a livello di singola relazione, di singola tabella. Il vincolo di chiave primaria è sicuramente uno dei vincoli più importanti che permette tramite ad un insieme di attributi di identificare in maniera univoca ogni record della relazione della tabella e garantisce che non ci siano valori nulli su quell'attributo, perché una chiave primaria non può assumere i valori nulli. Questo ha un riscontro pratico nei sistemi di gestione di basi di dati quando inseriamo una chiave infatti il sistema poi ha il compito di controllare che non ci siano violazioni a questo vincolo (duplicati o valori nulli).

INFRAZIONI					
Codice	Data	Agente	Articolo	Prov	Numero
1	25/10/14	567	44	FI	4E5432
2	26/10/14	456	34	FI	4E5432
3	26/10/14	638	34	FI	2F7643
4	15/10/14	456	53	MI	2F7643
5	12/10/14	567	44	MI	2F7643

I vincoli interrelazionali, cioè vincoli che ci permettono di congiungere le informazioni di tabelle diverse. La caratteristica principale del modello relazionale è che basato su valori, quindi i riferimenti che possiamo creare tra relazioni diverse avvengono tramite valori, quindi il fatto che nella prima riga della tabella infrazioni compaia il codice 567, ci fa capire che l'agente che ha rilevato questa infrazione è rossi Mario, e tale informazione la troviamo nella tabella degli agenti. Per far sì che queste informazioni che sono presenti nella tabella infrazioni abbiano valore è necessario che il codice che compare in corrispondenza dell'attributo agente si effettivamente il codice di un agente che è presente, e per garantire questo si utilizza il concetto di vincolo di integrità interrelazionale (vincolo di integrità referenziale). nella tabella auto abbiamo due attributi sottolineati ad indicare il fatto che in questo caso la chiave primaria è costituita da due attributi e per fare in modo che le informazioni nella tabella delle infrazioni abbiano una logica è necessario che la targa la si ritrovi nella tabella delle auto. tutte le volte che io ho bisogno di collegare tramite valori delle informazioni che si trovano in tabelle diverse devo creare in modo esplicito dei vincoli di questo tipo per garantire quando vado a fare le

AUTO			
Prov	Numero	Proprietario	Indirizzo
FI	2F7643	Verdi Piero	Via Tigli
FI	1A2396	Verdi Piero	Via Tigli
FI	4E5432	Bini Luca	Via Aceri
MI	2F7643	Bianchi Gino	Via Aceri

operazioni di congiunzione, cioè a congiungere le informazioni che si trovano nelle varie tabelle, questa operazione si possa fare senza difficoltà.

Un vincolo di integrità referenziale, foreign key, fra gli attributi X di una relazione R1 e un'altra relazione R2 impone ai valori su X in R1 di comparire come valori della chiave primaria di R2.

Esempi di vincoli di integrità referenziale fra: l'attributo Vigile della relazione INFRAZIONI e la relazione AGENTI; gli attributi Prov e Numero di INFRAZIONI e la relazione AUTO.

In realtà i sistemi di gestione di basi di dati permettono di creare vincoli di questi tipi anche su attributi che non sono la chiave primaria, ma una chiave deve essere comunque, però non necessariamente una chiave primaria, quindi in realtà questo concetto è più generale di quello sembra. Per questo motivo però quando dobbiamo in qualche modo indicare che in una relazione sono presenti dei vincoli, il consiglio che do è sempre quello di specificare anche per la tabella R2 quali sono gli attributi coinvolti perché non necessariamente si tratta sempre della chiave, quindi per essere precisi specifichiamo sempre il nome delle due tabelle e gli attributi che sono coinvolti.

Violazione di vincolo di integrità referenziale

INFRAZIONI					
Codice	Data	Agente	Articolo	Prov	Numero
1	25/10/14	567	44	FI	4E5432
2	26/10/14	456	34	FI	4E5432
3	26/10/14	638	34	FI	2F7643
4	15/10/14	456	53	MI	2F7643
5	12/10/14	567	44	TO	2F7643

AUTO			
Prov	Numero	Proprietario	Indirizzo
FI	2F7643	Verdi Piero	Via Tigli
FI	1A2396	Verdi Piero	Via Tigli
TO	4E5432	Bini Luca	Via Aceri
MI	2F7643	Bianchi Gino	Via Aceri

Nel caso del vincolo sulla targa abbiamo un vincolo che coinvolge due attributi perché la chiave di auto è costituita dalla provincia e dal numero. Attenzione in questo caso abbiamo questa coppia Torino e 2F7643 che singolarmente compaiono nella tabella delle auto, però compaiono in corrispondenza di righe diverse. Quello che si vuole è che la coppia compaia nella stessa tupla della tabella auto. Il fatto che ci sia già la provincia torino e che ci sia casualmente lo stesso numero in corrispondenza di un'altra targa non ha nessun significato, quindi quando sono coinvolti più attributi, come in questo caso, dobbiamo fare attenzione a definire il vincolo in modo che la coppia provincia e numero della tabella infrazioni corrisponda esattamente alla coppia provincia e numero della tabella auto.

Vincoli di integrità referenziale:

- Giocano un ruolo fondamentale nel concetto di modello basato su valori.
- In presenza di valori nulli i vincoli possono essere resi meno restrittivi.
- Sono possibili meccanismi per il supporto alla loro gestione (azioni compensative a seguito di violazioni).
- Attenzione ai vincoli su più attributi.

Integrità referenziale e valori nulli

Abbiamo due tabelle: una tabella impiegati, in cui abbiamo la chiave primaria matricole, poi abbiamo il cognome e il riferimento a un progetto, un'altra tabella progetti, in cui abbiamo il codice del progetto, la data di inizio, la durata e il costo. Ci si aspetta che il progetto faccia riferimento ad uno dei progetti della tabella progetti, quindi ci possiamo immaginare un vincolo di integrità tra l'attributo progetto di impiegati e l'attributo codice di progetti. In questo esempio in corrispondenza della terza tupla c'è un valore nullo. Un valore di questo tipo è ragionevole che ci possa essere? Il sistema dovrebbe impedire la presenza di un valore nullo? Si ha un valore nullo in assenza di informazione, difatti Null rappresenta il fatto che un'informazione o non ce l'ho o in quel momento non la conosco. Quindi in questo caso dell'impiegato Verdi non si sa niente relativamente a eventuali progetti in cui è coinvolto. Per la definizione di vincolo di integrità questa è una situazione possibile, perché non sto violando nessuna particolare condizione dicendo che queste informazioni in questo momento ad esempio non ce l'ho oppure quel impiegato proprio non ha niente a che fare con i progetti e che quindi quell'informazione non ce l'avrà mai. Quindi il fatto di inserire un vincolo di integrità tra attributi di tabelle diverse non impedisce poi che in corrispondenza della tabella R1, quella in corrispondenza della quale creò il vincolo, non possa inserire dei valori null, è una cosa che è permessa e che i sistemi riescono a gestire senza problemi.

IMPIEGATI		
Matricola	Cognome	Progetto
34321	Rossi	IDEA
64521	Neri	XYZ
53524	Verdi	NULL
73032	Bianchi	IDEA

PROGETTI			
Codice	Inizio	Durata	Costo
IDEA	01/2012	36	200
BOH	07/2012	24	120
XYZ	09/2013	24	150

Azioni compensative

- Esempio: viene eliminata una tupla causando una violazione.
- Comportamento standard: rifiuto dell'operazione.

- Azioni compensative: eliminazione in cascata; introduzione di valori nulli.

Le azioni compensative che possono essere svolte quando ad esempio si vanno a fare delle operazioni di cancellazione, di modifica che coinvolgono record che sono coinvolti in vincoli e di questo tipo. L'esempio: abbiamo sempre gli impiegati e i progetti. Immaginiamo questa volta che si voglia eliminare, per qualche motivo, una riga della tabella progetti quindi il progetto xyz non mi interessa più mantenerlo, qui però c'è un problema, perché nella tabella degli impiegati c'è l'impiegato neri che fa riferimento a questo progetto, quindi è chiaro che se io lo cancello vi creo un problema per il fatto che ho creato un vincolo che mi dice che se inserisco un progetto, quello deve essere presente nella tabella dei progetti, quindi a seconda di quello che si decide di fare, si possono scegliere delle azioni compensative, ovvero se si cancella una riga della tabella dei progetti quello che si può fare è far sì che in corrispondenza dell'attributo progetto nel record neri compaia il valore null, quindi tutte le volte io faccio una cancellazione nella tabella progetti, come azione compensativa, dato che il vincolo viene violato a quel punto, posso introdurre il valore nullo in corrispondenza del valore che è coinvolto dal vincolo, quindi questa è una possibile azione che si può fare e che ogni sistema di gestione di base di dati secondo il modello relazionale vi permette di implementare.

Una soluzione più drastica, però in alcune situazioni può anche essere conveniente, è l'eliminazione a cascata. Eliminando una tupla nella tabella progetti automaticamente vengono cancellate tutte le tuple della tabella impiegati che contengono il valore del progetto che viene eliminato.

Le possibilità sono l'eliminazione a cascata oppure inserire il valore nullo oppure di default impedire la modifica o la cancellazione, quindi siccome ho un vincolo non posso cancellare nessun progetto. Attenzione: posso cancellare liberamente un impiegato, perché il problema ce l'ho in una direzione e non nell'altra. Invece sui progetti devo decidere che politica voglio utilizzare.

Vincoli multipli su più attributi

Vincoli di integrità referenziale fra:

- gli attributi ProvA e NumeroA di INCIDENTI e la relazione AUTO;
- gli attributi ProvB e NumeroB di INCIDENTI e la relazione AUTO.

L'ordine degli attributi è significativo.

INCIDENTI					
Codice	Data	ProvA	NumeroA	ProvB	NumeroB
343211	25/10/14	FI	E39548	MI	39548K
645211	26/10/14	TO	839548	FI	E39548

AUTO				
Prov	Numero	Cognome	Nome	
FI	E39548	Verdi	Piero	
MI	39548K	Bini	Luca	
TO	839548	Bianchi	Gino	

In una stessa tabella posso inserire più vincoli, come abbiamo già visto peraltro nella tabella delle infrazioni. Ho una tabella degli incidenti in cui sono coinvolte due macchine, quindi ho due targhe, e in questo caso quello che voglio è che, sia la prima targa che la seconda targa, appartengano ad auto che compaiono nella tabella Auto. Questo è un caso un po' particolare perché devo utilizzare nomi diversi per gli attributi che identificano le due targhe, perché poi devo fare attenzione a non confondere i valori degli attributi, però il fatto di utilizzare dei nomi diversi per distinguerli ci aiuta, non posso mescolare la provincia della prima targa con il numero della seconda. Voglio che la coppia provinciaA-numeroA e provinciaB-numeroB facciano riferimento a una targa di auto.

Algebra relazionale

Insieme di operatori su relazioni che producono relazioni e possono essere composti. Gli operatori dell'algebra relazionale si indicano con delle lettere greche.

Operatori dell'algebra relazionale: unione, intersezione, differenza, ridenominazione, selezione, proiezione, join (join naturale, prodotto cartesiano, theta-join).

Operatori insiemistici: le relazioni sono insiemi; i risultati devono essere relazioni; è possibile applicare unione, intersezione, differenza solo a relazioni definite sugli stessi attributi.

Una volta organizzate queste informazioni, ho bisogno di fare delle interrogazioni sulla base di dati (ricercare informazioni, congiungere informazioni che sono in tabelle diverse ecc). L'algebra relazione è costituita dall'insieme degli operatori che sono messi a disposizione dal modello relazionale per rispondere a delle interrogazioni su una base di dati definita secondo questo modello. Si tratta di operatori che agiscono sulle relazioni, quindi sulle tabelle della base di dati, e che producono come risultato relazioni, quindi prendono in input una o più tabelle e restituiscono una tabella. Questi operatori possono

essere composti e riescono a formulare anche interrogazioni molto complesse. I primi sono gli operatori della insiemistica, quindi unione, intersezione e differenza. Poi abbiamo invece degli operatori che sono proprio caratteristici del modello relazionale, che sono la ridenominazione, la selezione e la proiezione e poi quello più importante di tutti è l'operatore di join, l'operatore che ci permette di congiungere le informazioni che si trovano in tabelle diverse. Quando parleremo di join faremo distinzione tra join naturale, prodotto cartesiano e join, che sono tutte versioni un po' diverse dello stesso concetto.

Una relazione è un insieme, che rappresentiamo in forma di tabella, quindi è naturale che si possano utilizzare gli operatori d'insieme: unione, intersezione, differenza. Essendo relazioni un po' particolari, in cui abbiamo i nomi degli attributi (a differenza della relazione matematica), queste operazioni d'insieme si possono fare purché si lavori con tabelle che hanno lo stesso schema.

LAUREATI			SPECIALISTI		
Matricola	Nome	Eta	Matricola	Nome	Eta
7274	Rossi	42	9297	Neri	33
7432	Neri	54	7432	Neri	54
9824	Verdi	45	9824	Verdi	45

LAUREATI \cup SPECIALISTI		
Matricola	Nome	Eta
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

LAUREATI			SPECIALISTI		
Matricola	Nome	Eta	Matricola	Nome	Eta
7274	Rossi	42	9297	Neri	33
7432	Neri	54	7432	Neri	54
9824	Verdi	45	9824	Verdi	45

LAUREATI \cap SPECIALISTI		
Matricola	Nome	Eta
7432	Neri	54
9824	Verdi	45

Esempio: ho due tabelle, quella dei laureati con attributi matricola, nome, età e quella degli specialisti con attributi matricola, nome, età. Posso fare l'unione di queste due relazioni e il risultato di questa unione è una nuova relazione definita con lo stesso schema che conterrà le tuple che sono sia da una parte che dall'altra. Se ci sono dei duplicati il record compare una volta sola (nella teoria degli insiemi non ci sono duplicati).

L'intersezione di due relazioni è una relazione definita sempre sullo stesso schema delle due iniziali. Si prendono i record che sono sia da una parte che dall'altra, quindi in questo caso abbiamo neri e verdi che appartengono sia ai laureati che agli specialisti.

LAUREATI			SPECIALISTI		
Matricola	Nome	Eta	Matricola	Nome	Eta
7274	Rossi	42	9297	Neri	33
7432	Neri	54	7432	Neri	54
9824	Verdi	45	9824	Verdi	45

LAUREATI – SPECIALISTI		
Matricola	Nome	Eta
7274	Rossi	42

Differenza: possiamo fare quest'operazione se le relazioni sono definite sugli stessi schemi e per ottenere il risultato si prendono tutti i record della tabella dei laureati e a questi record vado a togliere quelli che compaiono anche nella tabella degli specialisti, quindi in questo caso rimane soltanto rossi, perché appartiene alla tabella laureati ma non compare nella tabella degli specialisti.

Un'unione sensata ma impossibile

PATERNITÀ		MATERNITÀ	
Padre	Figlio	Madre	Figlio
Adamo	Abele	Eva	Abele
Adamo	Caino	Eva	Set
Adamo	Isacco	Sara	Isacco

PATERNITÀ \cup MATERNITÀ

??

Abbiamo due relazioni: una paternità con attributi padre e figlio e poi una relazione maternità che fa riferimento agli attributi madre e figlio. Voglio avere informazioni in generale sul genitore e sul figlio. In questo caso ho due relazioni che hanno degli schemi diversi, quindi non posso fare l'unione. In questo caso nel modello relazionale posso utilizzare è un operatore specifico del modello che permette di cambiare il nome degli attributi, ovvero l'operatore ridenominazione, con tale operatore riesco in maniera formale precisa a definire l'unione anche in queste situazioni. Applico l'operatore ad un'istanza di relazione e devo specificare quando applico l'operatore quali sono gli attributi a cui cambio il nome.

Ridenominazione

Operatore monadico (con un argomento). Modifica lo schema lasciando inalterata l'istanza dell'operando.

Data r di schema $R(A_1, \dots, A_k)$ e un insieme di attributi B_1, \dots, B_k

l'operatore di ridenominazione $\rho_{B_1 \dots B_k \leftarrow A_1 \dots A_k}(r)$ oppure $REN_{B_1 \dots B_k \leftarrow A_1 \dots A_k}(r)$

produce una relazione di schema $R(B_1, \dots, B_k)$ che contiene una tupla t' per ogni tupla t contenuta nella relazione originaria in modo tale che $t'[B_i] = t[A_i] \forall i$.

PATERNITÀ		$\rho_{\text{Genitore} \leftarrow \text{Padre}}(\text{PATERNITÀ})$	
Padre	Figlio	Genitore	Figlio
$\rho_{\text{Genitore} \leftarrow \text{Padre}}(\text{PATERNITÀ})$		$\rho_{\text{Genitore} \leftarrow \text{Madre}}(\text{MATERNITÀ})$	
Genitore	Figlio	Genitore	Figlio
Adamo	Abele	Eva	Abele
Adamo	Caino	Eva	Set
Adamo	Isacco	Sara	Isacco

$\rho_{\text{Genitore} \leftarrow \text{Padre}}(\text{PATERNITÀ}) \cup \rho_{\text{Genitore} \leftarrow \text{Madre}}(\text{MATERNITÀ})$	
Genitore	Figlio
Adamo	Abele
Adamo	Caino
Adamo	Isacco
Eva	Abele
Eva	Set
Sara	Isacco

In questa relazione, tramite l'operatore ridenominazione, posso modificare l'attributo padre in genitore. Applico l'operatore alla tabella paternità e la trasformazione che faccio è quella che padre diventa genitore, quindi se io applico questo operatore ottengo una nuova relazione, che a differenza degli operatori di unione, intersezione e differenza (le quali restituiscono una relazione definita sullo stesso schema ma con istanze diverse che dipendono dal tipo di operazione applicata), contiene esattamente le stesse tuple di prima, quindi non vado a modificare il contenuto, ma modifichiamo lo schema. Ho un operatore che mi permette di modificare lo schema della relazione.

Risoluzione del problema dell'unione di queste due tabelle: posso ridenominare padre in genitore in paternità e madre in genitore in maternità, a questo punto mi sono ricondotto a due tabelle che hanno lo stesso schema e quindi posso farne l'unione.

L'operatore di ridenominazione mi permette di andare a modificare il nome di uno o più attributi dello schema. Questo operatore ha un ruolo fondamentale quando si fanno operazioni di congiunzione tra tabelle.

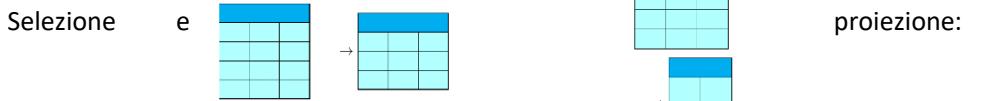
IMPIEGATI		
Cognome	Ufficio	Stipendio
Rossi	Pisa	55
Neri	Firenze	64
OPERAI		
Cognome	Fabbrica	Salario
Bruni	Lucca	45
Verdi	Siena	55

Un esempio di unione tra queste due relazioni impiegati e operai. La prima con attributi cognome, ufficio e stipendio mentre la seconda con attributi cognome, fabbrica e salario. È chiaro che ufficio e stipendio - fabbrica e salario sono concetti che si corrispondono. È stata fatta l'unione di queste due relazioni dopo aver ridenominato nella prima relazione la coppia ufficio-stipendio in sede e retribuzione e nella seconda tabella invece fabbrica-salario sempre in sede e retribuzione.

$\sigma_{Sede, Retribuzione = Ufficio, Stipendio}(\text{IMPIEGATI}) \sqcup \sigma_{Sede, Retribuzione = Fabbrica, Salario}(\text{OPERAI})$		
Cognome	Sede	Retribuzione
Rossi	Pisa	55
Neri	Firenze	64
Bruni	Lucca	45
Verdi	Siena	55

operatori ortogonali

- Selezione: decomposizione orizzontale
- Proiezione: decomposizione verticale



La selezione e la proiezione sono due operatori importanti dell'algebra relazionale. Definiti ortogonali, nel senso che il primo fa una decomposizione orizzontale della relazione mentre il secondo esegue una decomposizione verticale della relazione. Tutti e due questi operatori si applicano ad una singola relazione, quindi in input vogliono una relazione e restituiscono una relazione. L'operatore di selezione restituire un sottoinsieme delle tuple della tabella che soddisfano una certa condizione, per cui si parla di decomposizione orizzontale. Invece la proiezione restituisce un sottoinsieme degli attributi, quindi sono coinvolte tutte le tuple, ma di questi record non ci interessano tutte le informazioni ma soltanto quelli che andiamo a considerare, quindi facciamo proprio un taglio verticale.

Selezione

- Operatore monadico: si applica ad una singola tabella e il risultato è una tabella.
- Produce un risultato che ha lo stesso schema dell'operando.
- Contiene un sottoinsieme delle ennuple dell'operando, ovvero quelle che soddisfano una condizione.

(Per poter specificare una selezione ho bisogno di specificare una condizione che mi permette di selezionare certi record piuttosto che altri) Formula proposizionale per schema R(A₁, ..., A_k):

- A_i θ A_j con θ ∈ {=, ≠, >, <, ≤, ≥} è una formula;
- A_i θ c con c ∈ dom(A_i) è una formula;
- se F₁ e F₂ sono formule, allora F₁ and F₂, F₁ or F₂, not F₁ sono formule;

Data r di schema R(A₁, ..., A_k) e F formula proposizionale l'operatore di selezione

$\sigma_F(r)$ oppure $SEL_F(r)$

produce una relazione sugli attributi di R che contiene le tuple di r su cui F è vera.

Ho la tabella degli impiegati: ad esempio potrei essere interessata a sapere quali sono gli impiegati che guadagnano più di 50, quindi prendo un sottoinsieme delle tuple che soddisfano questa condizione, oppure potrei essere interessato a sapere quali sono gli impiegati che guadagnano più di 50 e lavorano a prato, quindi una condizione in questo caso che coinvolge più attributi, oppure sono interessata ad avere gli impiegati che hanno lo stesso nome della filiale presso cui lavorano, quindi in questo caso ce n'è uno che è il terzo impiegato. Impiegati che guadagnano più di 50: In questo caso sto utilizzando una formula proposizionale del tipo A_i θ c, cioè confronto un attributo con uno dei valori che l'attributo può assumere, cioè richiedo nello stipendio sia maggiore di 50.

• Impiegati che guadagnano più di 50.

IMPIEGATI			
Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Firenze	55
5998	Neri	Prato	64
9553	Prato	Prato	44
5698	Neri	Pisa	64

• Impiegati che guadagnano più di 50 e lavorano a Prato.

IMPIEGATI			
Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Firenze	55
5998	Neri	Prato	64
9553	Prato	Prato	44
5698	Neri	Pisa	64

$\sigma_{Stipendio > 50}(\text{IMPIEGATI})$

$\sigma_{Stipendio > 50 \text{ and Filiale} = \text{Prato}}(\text{IMPIEGATI})$			
Matricola	Cognome	Filiale	Stipendio
5998	Neri	Prato	64

Impiegati che guadagnano più di 50 e lavorano a prato: in questo caso la condizione coinvolge più attributi, per scrivere in maniera formale l'operazione di selezione dovrà utilizzare l'operatore sigma specificando come condizione la seguente: stipendio maggiore di 50 and filiale uguale a prato. ho una condizione composta tramite and logico.

Impiegati che hanno lo stesso nome della filiale presso cui lavorano: io posso esprimere con l'operatore sigma con condizione: il cognome sia uguale a filiale. Sto utilizzando una formula del tipo $\sigma_{Cognome=Filiale}(r)$.

• Impiegati che hanno lo stesso nome della filiale presso cui lavorano.

IMPIEGATI			
Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Firenze	55
5998	Neri	Prato	64
9553	Prato	Prato	44
5698	Neri	Pisa	64

$\sigma_{Cognome=Filiale}(IMPIEGATI)$			
Matricola	Cognome	Filiale	Stipendio
9553	Prato	Prato	44

Proiezione

- Operatore monadico.
- Produce un risultato che ha parte degli attributi dell'operando.
- Contiene ennuple cui contribuiscono tutte le ennuple dell'operando.

Dato l'insieme di attributi $X = \{A_1, \dots, A_k\}$, la relazione r di schema $R(X)$ e un sottoinsieme

$Y \subset X$ l'operatore di proiezione (dobbiamo specificare quali sono gli attributi rispetto ai quali vogliamo effettuare questa proiezione)

$\pi_Y(r)$ oppure $PROJ_Y(r)$

produce una relazione su Y ottenuta dalla tuple di r considerando solo i valori su Y .

Cardinalità delle proiezioni

Una proiezione contiene al più tante ennuple quante l'operando ma può contenerne di meno. Se X è una superchiave (insieme di attributi che include una chiave) di R , allora $\pi_X(R)$ contiene esattamente tante ennuple quante R .

L'operatore di proiezione permette di fare un taglio verticale delle tabelle.

- Matricola e cognome di tutti gli impiegati.

IMPIEGATI			
Matricola	Cognome	Filiale	Stipendio
7309	Neri	Firenze	55
5998	Neri	Prato	64
9553	Rossi	Pisa	44
5698	Rossi	Pisa	64

$\pi_{Matricola,Cognome}(IMPIEGATI)$

$\pi_{Matricola,Cognome}(IMPIEGATI)$	
Matricola	Cognome
7309	Neri
5998	Neri
9553	Rossi
5698	Rossi

- Cognome e filiale di tutti gli impiegati.

IMPIEGATI			
Matricola	Cognome	Filiale	Stipendio
7309	Neri	Firenze	55
5998	Neri	Prato	64
9553	Rossi	Pisa	44
5698	Rossi	Pisa	64

$\pi_{Cognome,Filiale}(IMPIEGATI)$

$\pi_{Cognome,Filiale}(IMPIEGATI)$	
Cognome	Filiale
Neri	Firenze
Neri	Prato
Rossi	Pisa
Rossi	Pisa

Consideriamo la tabella degli impiegati e applichiamo le operazioni di proiezione: non sono interessata ad avere tutte le caratteristiche degli impiegati ma mi interessa soltanto avere l'informazione sulla matricola e sul cognome, quindi uso l'operatore pi greco π con pedice matricola, cognome applicato ad impiegato, oppure voglio il cognome e la filiale di tutti gli impiegati. Attenzione: quando faccio l'operazione in questo esempio in cui ci sono due tuple, la terza e la quarta, che coincidano, ovvero ci sono due impiegati che hanno lo stesso cognome e che lavorano nella stessa filiale. Inizialmente considero tutte le tuple come richiesto, infine però il risultato che ottengo avrà un numero di tuple inferiore a quello della tabella iniziale perché la coppia rossi-pisa comparirà soltanto una volta, perché le relazioni sono insiemi. Quando fate una proiezione su una tabella il risultato può avere un numero di record che è inferiore a quello della tabella da cui siamo partiti. Ci sono delle proiezioni che sicuramente mantengono la stessa cardinalità della tabella originale? Se ho una proiezione che coinvolge una chiave primaria della tabella è chiaro che non potrà succedere che compaiano dei record uguali, quindi il numero di tuple del risultato potrà essere inferiore se vado a fare una proiezione su attributi che non contengono una chiave della relazione. [una chiave è una super chiave minimale, la super chiave ha un insieme di attributi che mi permette di distinguere un record all'altro ma che contiene delle informazioni ridondanti, cioè posso togliere qualche attributo e continuare ad avere la stessa caratteristica, invece in una chiave questo non succede, quindi una chiave è una super chiave.]

tuple, la terza e la quarta, che coincidano, ovvero ci sono due impiegati che hanno lo stesso cognome e che lavorano nella stessa filiale. Inizialmente considero tutte le tuple come richiesto, infine però il risultato che ottengo avrà un numero di tuple inferiore a quello della tabella iniziale perché la coppia rossi-pisa comparirà soltanto una volta, perché le relazioni sono insiemi. Quando fate una proiezione su una tabella il risultato può avere un numero di record che è inferiore a quello della tabella da cui siamo partiti. Ci sono delle proiezioni che sicuramente mantengono la stessa cardinalità della tabella originale? Se ho una proiezione che coinvolge una chiave primaria della tabella è chiaro che non potrà succedere che compaiano dei record uguali, quindi il numero di tuple del risultato potrà essere inferiore se vado a fare una proiezione su attributi che non contengono una chiave della relazione. [una chiave è una super chiave minimale, la super chiave ha un insieme di attributi che mi permette di distinguere un record all'altro ma che contiene delle informazioni ridondanti, cioè posso togliere qualche attributo e continuare ad avere la stessa caratteristica, invece in una chiave questo non succede, quindi una chiave è una super chiave.]

Selezione e proiezione

Combinando selezione e proiezione, possiamo estrarre interessanti informazioni da una relazione. Non possiamo però correlare informazioni presenti in relazioni diverse, né informazioni in ennuple diverse di una stessa relazione.

- Matricola e cognome degli impiegati che guadagnano più di 50.

IMPIEGATI	
Matricola	Cognome
7309	Rossi
5998	Neri
5698	Neri

$\pi_{\text{Matricola}, \text{Cognome}}(\sigma_{\text{Stipendio} > 50}(\text{IMPIEGATI}))$

L'esempio è sempre quello degli impiegati: voglio la matricola e il cognome degli impiegati che guadagnano più di 50. È chiaro che sono coinvolte sia la selezione che la proiezione perché voglio che lo stipendio sia maggiore di 50 ma degli impiegati non mi interessano tutti i dettagli ma soltanto quelli riferiti alla matricola e al cognome, quindi faccio la selezione rispetto alla condizione stipendio maggiore di 50 e poi applico l'operatore di proiezione andando a selezionare la matricola e il cognome (è chiaro che a volte queste operazioni si possono anche invertire nell'ordine). In questo caso devo fare prima la selezione e poi la proiezione. Il risultato che si ottiene è una relazione che contiene soltanto la matricola e il cognome di quegli impiegati che hanno uno stipendio maggiore di 50.

Join

Join è l'operatore fondamentale dell'algebra relazionale perché è l'operatore che ci permette di congiungere informazioni che si trovano in tabelle diverse.

Numero	Voto
1	25
2	13
3	27
4	28

Numero	Candidato
1	Mario Rossi
2	Luca Verdi
3	Bianca Russo
4	Rosa Neri

Numero	Candidato	Voto
1	Mario Rossi	25
2	Luca Verdi	13
3	Bianca Russo	27
4	Rosa Neri	28

Esempio: Nelle prove scritte in un concorso pubblico i compiti sono anonimi e ad ognuno è associata una busta chiusa con il nome del candidato. Ciascun compito e la relativa busta vengono contrassegnati con uno stesso numero. Immaginiamo di essere in una situazione di questo tipo: 4 candidati che hanno fatto quattro compiti, quindi ho corretto questi compiti facendo riferimento a un numero 1, 2, 3 e 4, ma non so chi sono effettivamente le persone. Poi invece apro le buste chiuse e scopro chi sono i candidati che corrispondono a questi numeri. Ho una situazione di questo tipo: da una parte ho il voto con riferimento a un contrassegno e dall'altra ho il riferimento esplicito al candidato che ha eseguito il compito, allora quello che è necessario fare è congiungere queste informazioni.

L'operazione è quella di congiunzione, cioè prendere il primo record che si trova nella prima tabella e andare a vedere qual è il numero che gli corrisponde e poi andare a vedere nella seconda chi è il candidato che corrisponde a quel numero, quindi devo incrociare le informazioni delle due tabelle sulla base del valore che viene assunto in questi due casi dell'attributo numero. Il risultato che ottengo è una nuova tabella che, in questo caso, è definita sull'unione degli attributi, il numero compare una volta sola e poi compare il candidato e il voto. Questo è un esempio di quello che si chiama join naturale. Lo schema della nuova relazione che ottengo è definito sull'unione degli attributi. Facendo la congiunzione di queste due tabelle: ha uno schema che è definito sull'unione, se prendo una qualsiasi tupla di questa nuova relazione, ad esempio, la prima (1 - mario rossi - 25) e faccio la proiezione di questa tupla sugli attributi numero e voto (quindi 1 e 25), ritrovo un record della prima tabella. Se invece vado a fare una proiezione su numero e candidato trovo 1 e mario rossi, che è un record della seconda tabella. Quindi la caratteristica che hanno questi due record è quella di avere lo stesso valore in corrispondenza dell'attributo che è comune a tutte e due le relazioni, questa è una conseguenza della definizione di join naturale. Le tuple del risultato sono ottenute combinando le tuple degli operatori che hanno valori uguali sugli attributi che sono in comune alle due relazioni. Le situazioni che possono verificarsi sono le seguenti allora generalmente quando si applica un join naturale le due relazioni hanno attributi in comune, questa è l'uso classico che si fa di questo operatore però niente vieta di applicare l'operatore anche a situazioni invece estreme che sono quelle in cui le due relazioni sono completamente disgiunte (diventa un'altra operazione: prodotto cartesiano), quindi non ci sono attributi in comune oppure viceversa quando applicare l'operatore di join naturale a due relazioni in cui invece che hanno esattamente lo stesso schema, quindi in cui gli attributi si coincidono.

Join naturale, sintassi e semantica

- Operatore binario (generalizzabile)
- Produce un risultato sull'unione degli attributi degli operandi, con ennuple costruite ciascuna a partire da una ennupla di ognuno degli operandi.
- Dati R1(X1) e R2(X2) il join naturale R1 \bowtie R2 oppure R1 JOIN R2 è una relazione sull'unione X1 \cup X2 :

$$R1 \bowtie R2 = \{t \in X1 \cup X2 \mid \exists t1 \in R1 \text{ e } t2 \in R2 \text{ t.c. } t[X1] = t1 \text{ e } t[X2] = t2\}$$

Le tuple del risultato di un join naturale sono ottenute combinando tuple degli operandi con valori uguali sugli attributi. Infatti, se $X1,2 = X1 \cap X2$, le condizioni $t[X1] = t1$ e $t[X2] = t2$ implicano $t[X1,2] = t1[X1,2]$ e $t[X1,2] = t2[X1,2]$, pertanto $t1[X1,2] = t2[X1,2]$.

- Se X1 e X2 hanno attributi in comune si ha la definizione tradizionale di join naturale.
- Se X1 e X2 sono disgiunti si ha la definizione di prodotto cartesiano (vedi dopo).
- Se X1 e X2 coincidono si ha la definizione dell'intersezione.

Esempio: join naturale e intersezione

LAUREATI		
Matricola	Nome	Eta
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

SPECIALISTI		
Matricola	Nome	Eta
9297	Neri	33
7432	Neri	54
9824	Verdi	45

$$\text{LAUREATI} \bowtie \text{SPECIALISTI} = \text{LAUREATI} \cap \text{SPECIALISTI}$$

LAUREATI \bowtie SPECIALISTI		
Matricola	Nome	Eta
7432	Neri	54
9824	Verdi	45

Esempio: laureati-specialisti, ho due schemi, questa volta, identici: gli attributi sono gli stessi matricola, nome ed età. Se voglio fare il join naturale di queste due relazioni, il risultato è una relazione che è definita sull'unione degli attributi ma l'unione degli attributi e coincide esattamente. Nel risultato ci devo mettere delle tuple con la caratteristica che se vado a proiettare questa tupla sugli attributi della prima trovo una tupla della prima relazione e che se vado a proiettare questa tupla sulla gli attributi della seconda relazione trovo una tupla della seconda. In questa situazione particolare in cui tutti gli attributi sono uguali le uniche tuple che hanno questa caratteristica sono quelle che appartengono all'intersezione.

Join naturale, con attributi in comune

Date r, s definite su insiemi di attributi non disgiunti $R(A_1, \dots, A_k, \dots, A_n)$ e $S(A_1, \dots, A_k, B_1, \dots, B_m)$ il risultato di $r \bowtie s$ è una relazione definita su $A_1, \dots, A_n, B_1, \dots, B_m$

$$Z(A_1, \dots, A_n, B_1, \dots, B_m)$$

che contiene il seguente insieme di tuple $\{t \mid t[A_1, \dots, A_n] \in r \text{ e } t[A_1, \dots, A_k, B_1, \dots, B_m] \in s\}$.

Esempio: join completo

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
A	Mori
B	Bruni

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

Ogni tupla contribuisce al risultato.

risultato di questo join per farvi capire la definizione che abbiamo dato all'inizio: se io prendo un record della tabella risultante rossi-A-mori e vado a considerare la proiezione su impiegato e reparto trovo una tupla che è rossi-A che si trova nella prima relazione, se io di questo record invece vado a prendere la proiezione sul reparto e capo trovo il record A-mori che è una tupla della seconda. Perché questo succeda è necessario che queste due tuple siano uguali sull'attributo che le due relazioni hanno in comune. Questo in particolare si chiama join completo perché ogni tupla contribuisce al risultato del join, quindi non ci sono tuple nel risultato che sono state tagliate fuori dall'operazione. Tutte le tuple della prima e della seconda in qualche modo sono coinvolte nel risultato, però non è sempre così.

Esempio: join non completo

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
B	Mori
C	Bruni

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

Le tuple che non contribuiscono al risultato si dicono **dondolanti** (dangling).

seconda, non danno nessun contributo al risultato.

Esempio di join che non è completo: ho sempre riferimento a impiegati e riparti e poi reparto e capo. La situazione è diversa per due motivi: innanzitutto ho è il primo record che è quello rossi che lavora in reparto A, ma di questo reparto non so chi è il capo, quindi il primo record non lo congiungo con niente. Invece gli impiegati neri e bianchi possono essere congiunti con mori che è il capo del reparto B. C'è anche il record C che fa riferimento al capo bruni ma anche questo non è coinvolto nel risultato, perché non c'è nessun impiegato che lavora nel reparto C. Esempio di join in cui due record, rossi nella prima e bruni nella

Esempio: join vuoto

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

Un altro esempio è il join vuoto: gli impiegati lavorano nei reparti A e B, però conosco soltanto i capi delle reparti C e D, quindi quando vado a fare la congiunzione di queste due relazioni ottengo una relazione vuota.

Un join completo, con $n \times m$ ennuple

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

ogni record della prima relazione lo posso congiungere con tutti quelli della seconda, ho due impiegati che lavorano entrambi nel reparto B, ma questo reparto ha due capi, sia Mori che Bruni. quando eseguo la congiunzione avrò come risultato del join quattro record perché il primo record Rossi lo potrà congiungere sia con Mori che con Bruni e la stessa cosa vale per quanto riguarda Neri. il risultato di questa operazione è un insieme che coinvolge ogni tupla è stata congiunta con tutte quelle della seconda relazione.

Cardinalità del join

- Il join di R1 e R2 contiene un numero di ennuple compreso fra zero e il prodotto di $|R1|$ e $|R2|$;
- se il join coinvolge una chiave di R2, allora il numero di ennuple è compreso fra zero e $|R1|$;
- se il join coinvolge una chiave di R2 e un vincolo di integrità referenziale, allora il numero di ennuple è pari a $|R1|$.

Date $R1(A,B)$, $R2(B, C)$

- in generale si ha $0 \leq |R1 \bowtie R2| \leq |R1| \times |R2|$
- se B è chiave in R2 $0 \leq |R1 \bowtie R2| \leq |R1|$
- se B è chiave in R2 ed esiste vincolo di integrità referenziale fra B (in R1) e R2: $|R1 \bowtie R2| = |R1|$.

Considerazioni generali sulla cardinalità dell'operazione di join: in generale se io ho due tabelle R1 e R2 che vado a congiungere tramite un join naturale è quello che mi devo aspettare è che un join vuoto oppure viceversa tutte le righe della prima le posso congiungere con quelle della seconda, quindi è chiaro che in generale quello che succede è che la cardinalità di un join naturale è compreso tra zero, nel caso in cui join è vuoto, oppure è il prodotto delle cardinalità delle due relazioni.

STUDENTI			ELABORATI	
Nome	Corso	Progetto	Progetto	Tutor
Rossi	BDSI	BD1	BD1	Mori
Bianchi	BDSI	BD1	BD2	Bruni
Verdi	BDSI	BD2	A3	Viola
Neri	ASD	A3	A4	Grigio

Vincolo di integrità referenziale tra STUDENTI.Progetto e ELABORATI.Progetto.

Ogni tupla di STUDENTI contribuisce al risultato una volta.

$$|STUDENTI \bowtie ELABORATI| = |STUDENTI|$$

ASD, A4] non contribuisce.

$$|STUDENTI \bowtie ELABORATI| \leq |STUDENTI|$$

Ci sono dei casi particolari perché se il join che vado a fare coinvolge una chiave della relazione R2 allora il numero è compreso fra zero e la cardinalità di R1. Un esempio di join che coinvolge una chiave di R2: c'è un vincolo di integrità referenziale tra le due allora il risultato dei join avrà un numero di tuple pari alla cardinalità di R1. Abbiamo una tabella studenti che ha come attributi il nome, il corso e il progetto e poi una tabella degli elaborati in cui per ogni progetto si sa chi è il tutor. Se vado a fare il join naturale tra queste due tabelle che cosa succede? La chiave della seconda tabella è progetto, quindi quando si fa la congiunzione, è chiaro che ogni record della prima tabella lo posso congiungere al più con un record della seconda, quindi rossi lo congiungo con mori, bianchi lo congiungo con mori, verdi lo congiungo con bruni, neri con viola. C'è rossi che ha un progetto A4 e di questo non conosco il tutor. Le

situazioni che possono verificarsi sono due: non la congiungo o la congiungo al più una volta, perché progetto è chiave della seconda relazione. In questo caso il join naturale mi darà un numero di tuple che è minore o uguale a quelle della prima tabella, quindi una tupla che è quella di rossi non sarebbe coinvolta nel join, quindi parto da una tabella che ha 5 record ma nel risultato ne avrei soltanto 4.

Se richiedo che nella tabella studenti progetto debba comparire per forza come progetto della tabella elaborati, cioè non ci può essere un record che contenga riferimento ad un progetto che non esiste, quindi ho un vincolo di integrità referenziale. In questo caso è chiaro che quello che succede è che ogni tupla della tabella studenti la congiungo con una tupla della tabella elaborati soltanto in un modo, quindi la cardinalità del risultato è esattamente pari alla cardinalità della tabella studenti. Niente vieta che nella tabella degli elaborati non esistano altri record che fanno riferimento a progetti che non sono coinvolti nella tabella degli studenti.

Queste considerazioni sono importanti perché fare un join è costoso.

Join, una difficoltà

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
B	Mori
C	Bruni

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

Alcune ennuple non contribuiscono al risultato:
vengono tagliate fuori.

Join esterno

Il join esterno estende, con valori nulli, le ennuple che verrebbero tagliate fuori da un join (interno).

Esiste in tre versioni:

- sinistro: mantiene tutte le ennuple del primo operando, estendendole con valori nulli, se necessario;
- destro: ... del secondo operando ...
- completo: ... di entrambi gli operandi ...

Join esterno, in modo formale

Date r, s definite su insiemi di attributi non disgiunti $R(A_1, \dots, A_k, \dots, A_n)$ e $S(A_1, \dots, A_k, B_1, \dots, B_m)$ il risultato di $r \star_{\text{FULL}} s$ è una relazione definita su $A_1, \dots, A_n, B_1, \dots, B_m$

$$Z(A_1, \dots, A_n, B_1, \dots, B_m)$$

definita come segue

$$r \star_{\text{FULL}} s = r \star s \cup (r - \pi_{A_1, \dots, A_n}(r \star s)) \times \{B_1 = \text{null}, \dots, B_m = \text{null}\} \cup \{A_{k+1} = \text{null}, \dots, A_n = \text{null}\} \times (s - \pi_{A_1, \dots, A_k, B_1, \dots, B_m}(r \star s))$$

Le tuple che non contribuiscono al join naturale vengono unite con tuple nulle.

IMPIEGATI	
Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

REPARTI	
Reparto	Capo
B	Mori
C	Bruni

IMPIEGATI \bowtie_{LEFT} REPARTI	
Impiegato	Reparto
Reparto	Capo
Neri	B
Bianchi	B
Rossi	A

Permette di inserire nel risultato del join naturale anche tutto o un sottoinsieme delle tuple che altrimenti verrebbero tagliate fuori. Esistono tre versioni quel del join: il join sinistro, quello destro e quello completo. Il join sinistro mantiene tutte le tuple del primo operando estendendole eventualmente con valori nulli quando non ho un corrispondente nella tabella a destra, e il join destro fa esattamente la stessa cosa con la tabella di destra, quello completo invece fa tutte e due le cose, quindi cerca di mantenere sia a sinistra che quelle a destra. Se io faccio è un join sinistro oltre alle due tuple che già naturalmente inserite perché hanno lo stesso valore di reparto, facendo un join sinistro andrà anche a considerare il record rossi-A che si trova nella

prima relazione di cui non ho l'informazione sul capo e quindi ci introduce un valore nullo. Se vado a fare un join destro invece succede la stessa cosa con la tupla che fa riferimento a bruni, quindi avrò i due impiegati che hanno reparto B e poi inserisco bruni con riferimento null in corrispondenza dell'impiegato.

IMPIEGATI	
Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

REPARTI	
Reparto	Capo
B	Mori
C	Bruni

IMPIEGATI \bowtie_{FULL} REPARTI	
Impiegato	Reparto
Reparto	Capo
Neri	B
Bianchi	B
Rossi	A
NULL	C

IMPIEGATI	
Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

REPARTI	
Reparto	Capo
B	Mori
C	Bruni

IMPIEGATI \bowtie_{RIGHT} REPARTI	
Impiegato	Reparto
Reparto	Capo
Neri	B
Bianchi	B
NULL	C

il join completo oltre a considerare i due record che hanno lo stesso valore in reparto, aggiunge sia la tupla che fa riferimento al reparto A e la tupla che fa riferimento al capo Bruni del reparto C, quindi questa è la situazione più generale che si può avere. in questo caso non ho perso nessuna informazione. comunque può essere utile se voglio eventualmente aggiungere le informazioni in un secondo momento.

Algebra relazionale-20220315 0801-1

PRODOTTO

Date r, s con schemi $R(A_1, \dots, A_n)$ e $S(B_1, \dots, B_m)$ il risultato di $r \times s$ è una relazione definita su $A_1, \dots, A_n, B_1, \dots, B_m$

$$Z(A_1, \dots, A_n, B_1, \dots, B_m)$$

le cui tuple sono ottenute concatenando ogni tupla di r con tutte le tuple di s ottenendo l'insieme

$$\{t \mid t = uv \text{ con } u \in r \text{ e } v \in s\}$$

NOTA: il prodotto è un operatore primitivo, insieme a ridenominazione, unione, differenza, selezione e proiezione. Invece, per l'intersezione si ha $r \cap s = r - (r - s)$, quindi anche con la differenza.

PRODOTTO e JOIN

- Un join naturale su relazioni senza attributi in comune coincide con il prodotto delle due relazioni.
- La condizione che le due tuple debbono avere gli stessi valori sugli attributi comuni degenera in una condizione sempre verificata.
- Un prodotto contiene sempre un numero di ennuple pari al prodotto delle cardinalità degli operandi (le ennuple sono tutte combinabili).

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

IMPIEGATI × REPARTI			
Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Rossi	A	B	Bruni
Neri	B	A	Mori
Neri	B	B	Bruni
Bianchi	B	A	Mori
Bianchi	B	B	Bruni

Il prodotto cartesiano è una relazione definita su schema che contiene tutti gli attributi, quindi a differenza del join naturale che è definito sull'unione degli attributi, il prodotto cartesiano è definito sulla totalità degli attributi concatenando ogni record della prima tabella con tutti quelli della seconda, quindi in tutti i modi possibili. Il prodotto è un operatore primitivo, nel senso che è non riesco a definirlo tramite altri operatori. Esempio di prodotto cartesiano: ho le stesse tabelle di prima ma questa volta gli attributi hanno nomi diversi, quindi nel primo caso si parla di reparto, nel secondo di codice, quindi il prodotto cartesiano è rappresentato da una relazione definita su tutti gli attributi che definiscono le mie tabelle e che contiene tante tuple quante sono quelle della prima tabella moltiplicate per quelle della seconda. Fare un join naturale su

relazioni che non hanno attributi in comune è esattamente coincidente con il prodotto cartesiano tra le due tabelle.

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

IMPIEGATI × REPARTI		
Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

IMPIEGATI REPARTI		
Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Bruni
Verdi	A	Bini

π _{Impiegato, Reparto} (IMPIEGATI × REPARTI)		π _{Reparto, Capo} (IMPIEGATI × REPARTI)	
Impiegato	Reparto	Reparto	Capo
Neri	B	B	Mori
Bianchi	B		

IMPIEGATI REPARTI		
Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Bruni
Verdi	A	Bini

π _{Impiegato, Reparto} (IMPIEGATI × REPARTI) × π _{Reparto, Capo} (IMPIEGATI × REPARTI)		
Impiegato	Reparto	Capo
Neri	B	Mori
Neri	B	Bruni
Bianchi	B	Mori
Bianchi	B	Bruni
Verdi	A	Bini

Un'osservazione su join e proiezione: consideriamo queste due tabelle impiegati e reparto, in questo caso abbiamo un attributo in comune e andiamo a fare il join naturale, il risultato sono queste due tuple e se a questo punto faccio la proiezione di questa nuova tabella che ho trovato rispetto agli attributi delle due tabelle iniziali, quindi la proiezione su impiegato e riparto e poi su reparto e capo. Le tabelle che ottengo sono diverse da quelle iniziali, quindi questo per dire che quando faccio un'operazione di join non ottengo la stessa situazioni di partenza, perché le tuple che non sono state considerate le ho perse.

Una considerazione analoga vale anche nella direzione opposta: se parto da una tabella che ha tre attributi impiegato, reparto e capo e faccio prima la proiezione su impiegato e reparto e poi sul reparto e capo ottenendo due tabelle che hanno tre record ciascuna. Se a questo punto di queste due relazioni faccio il join naturale ottengo una relazione che non è uguale a quella iniziale perché tutte le volte in cui compare il reparto B questo lo congiungo due volte con il record della seconda tabella. Bisogna essere consci di questa caratteristica.

Join e proiezioni

- $R1(X1), R2(X2)$

$$\pi X_1 (R1 \bowtie R2) \subseteq R1$$

- $R(X), X = X1 \cup X2$

$$(\pi X_1 (R)) \bowtie (\pi X_2 (R)) \supseteq R$$

θ-join

- Il prodotto cartesiano, in pratica, ha senso solo se seguito da selezione:

$$\sigma_{\text{Condizione}} (R1 \times R2)$$

- L'operazione viene chiamata theta-join e indicata con $R1 \bowtie_{\text{Condizione}} R2$
- La condizione è spesso una congiunzione (AND) di atomi di confronto $A_1 \theta A_2$ dove θ è uno degli operatori di confronto ($=, >, <, \geq, \leq$).
- Se l'operatore di confronto nel theta-join è sempre l'uguaglianza ($=$) allora si parla di equi-join.

θ -join, in modo formale

Date r, s con schemi $R(A_1 \dots, A_n)$ e $S(B_1, \dots, B_m)$, $A_i \neq B_j$ il risultato di $r \bowtie_{A_i \theta B_j} s$ è una relazione definita su $A_1, \dots, A_n, B_1, \dots, B_m$

$$Z(A_1, \dots, A_n, B_1, \dots, B_m)$$

che contiene il seguente insieme di tuple $\{t \mid t = uv \text{ con } u \in r, v \in s, u[A_i] \theta v[B_j]\}$

Si ha $r \bowtie_{A_i \theta B_j} s = \sigma_{A_i \theta B_j}(r \times s)$

Il theta join è un prodotto cartesiano seguito da una selezione. Questa operazione è indicata con lo stesso simbolo che si usa per il join naturale però specificando una condizione (quando avete un'espressione algebrica con \bowtie senza niente quello è un join naturale). Un esempio: abbiamo le due tabelle impiegati e reparti, questa volta vogliamo fare un congiungere queste due

IMPIEGATI	
Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

REPARTI	
Codice	Capo
A	Mori
B	Bruni

sorelle però sulla base utilizzando come condizione reparto uguale a codice. Del prodotto cartesiano vado a selezionare quelle in cui il valore di reparto e il valore di codice sono uguali.

IMPIEGATI	
Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

REPARTI	
Codice	Capo
A	Mori
B	Bruni

Il join naturale può essere simulato per mezzo della ridenominazione, dell'equi-join e della proiezione.

$$\pi_{\text{Impiegato}, \text{Reparto}, \text{Capo}}(\sigma_{\text{Reparto}=\text{Codice}}(\text{Impiegati} \bowtie \rho_{\text{Codice} \leftarrow \text{Reparto}}(\text{Reparti})))$$

Join naturale ed equi-join

- si ridenominano gli attributi in modo da avere relazioni su schemi disgiunti;
- si effettua l'equi-join, con condizioni di uguaglianza sugli attributi corrispondenti;
- si effettua una proiezione per eliminare gli attributi doppi.

Applicazione ripetuta degli operatori

- Gli operatori $U, \cap, -, \rho, \sigma, \pi, \bowtie$ e $\bowtie_{\text{Condizione}}$ si applicano a tabelle e danno come risultato una tabella.
- È possibile quindi applicare un operatore al risultato prodotto da un altro.
- Molte interrogazioni possono essere risolte con una opportuna applicazione degli operatori visti.

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

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

Utilizzo degli operatori dell'algebra relazionale per rispondere a interrogazioni: consideriamo queste due tabelle, impiegati che contiene informazioni sulla matricola, il nome, l'età e lo stipendio e supervisione che ci dice per ogni impiegato qual è il suo capo. in questo caso sia l'impiegato che il capo sono impiegati, quindi sarebbe opportuno inserire un vincolo di integrità referenziale per impostare che questi codici dei capi corrispondano a codici di impiegati.

Interrogazione: trovare matricola, nome, età e stipendio degli impiegati che guadagnano più di 40. quando si tratta di fare una interrogazione la prima cosa che dovete fare è cercare di capire quale o quali sono le tabelle coinvolte. riguarda soltanto la tabella degli impiegati perché tutte le informazioni le troviamo in questa tabella e quindi dovremmo andare a fare una selezione rispetto alla condizione stipendio maggiore di 40 della tabella impiegati.

$$\sigma_{\text{Stipendio} > 40}(\text{Impiegati})$$

Interrogazione: voglio trovare la matricola, il nome e l'età degli impiegati che guadagnano più di 40, quindi è la stessa di prima, però questa volta non mi interessa avere l'indicazione sullo stipendio, quindi dovrò fare la stessa selezione che ho fatto nel caso precedente, ma dovrò far seguire questa selezione da una proiezione, perché mi interessano soltanto tre degli attributi che sono indicati e non tutti quindi

$$\pi_{\text{Matricola}, \text{Nome}, \text{Eta}}(\sigma_{\text{Stipendio} > 40}(\text{Impiegati}))$$

Capi degli impiegati che guadagnano più di 40

$\sigma_{\text{Stipendio} > 40}(\text{IMPIEGATI})$				SUPERVISIONE	
Matricola	Nome	Eta	Stipendio	Impiegato	Capo
7309	Rossi	34	45	7309	5698
5698	Bruni	43	42	5998	5698
4076	Mori	45	50	9553	4076
8123	Lupi	46	60	5698	4076
				4076	8123

$\text{SUPERVISIONE} \bowtie_{\text{Impiegato} = \text{Matricola}} (\sigma_{\text{Stipendio} > 40}(\text{IMPIEGATI}))$					
Impiegato	Capo	Matricola	Nome	Eta	Stipendio
7309	5698	7309	Rossi	34	45
5698	4076	5698	Bruni	43	42
4076	8123	4076	Mori	45	50

$\pi_{\text{Capo}}(\text{Supervisione} \bowtie_{\text{Impiegato} = \text{Matricola}} (\sigma_{\text{Stipendio} > 40}(\text{Impiegati})))$

Nome e stipendio dei capi degli impiegati che guadagnano più di 40

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

IMPIEGATI $\bowtie_{\text{Matricola} = \text{Capo}}$ CAPI				
Matricola	Nome	Eta	Stipendio	Capo
5698	Bruni	43	42	5698
4076	Mori	45	50	4076
8123	Lupi	46	60	8123

Impiegati che guadagnano più del proprio capo, mostrando matricola, nome e stipendio dell'impiegato e del capo

R=SUPERVISIONE $\bowtie_{\text{Impiegato} = \text{Matricola}}$ IMPIEGATI					
Impiegato	Capo	Matricola	Nome	Eta	Stipendio
7309	5698	7309	Rossi	34	45
5998	5698	5998	Bianchi	37	38
9553	4076	9553	Neri	42	35
5698	4076	5698	Bruni	43	42
4076	8123	4076	Mori	45	50

$\rho_{\text{MatrC}, \text{NomeC}, \text{EtaC}, \text{StipC} \leftarrow \text{Matricola}, \text{Nome}, \text{Eta}, \text{Stipendio}}(\text{IMPIEGATI}) \bowtie_{\text{MatrC}} \text{Capo}$					
MatrC	NomeC	EtaC	StipC	Impiegato	Capo
5698	Bruni	43	42	7309	5698
5698	Bianchi	43	42	5998	5698
4076	Mori	45	50	9553	4076
4076	Mori	45	50	4076	5698
8123	Lupi	46	60	8123	4076

ambiguità quando io vado a congiungere questa relazione con il risultato della congiunzione che ho fatto in precedenza perché se qui non facessi questa operazione avrei matricola che compare due volte.

$\pi_{\text{Matr}, \text{Nome}, \text{Stip}, \text{MatrC}, \text{NomeC}, \text{StipC}}(\sigma_{\text{Stipendio} > \text{StipC}}(\rho_{\text{MatrC}, \text{NomeC}, \text{EtaC}, \text{StipC} \leftarrow \text{Matricola}, \text{Nome}, \text{Eta}, \text{Stipendio}}(\text{IMPIEGATI}) \bowtie_{\text{MatrC}} \text{Capo}(\text{Supervisione} \bowtie_{\text{Impiegato} = \text{Matricola}} \text{IMPIEGATI))))$

Quello che voglio mettere in evidenza è l'uso della stessa tabella, cioè la tabella impiegati viene utilizzata due volte, e l'uso della ridenominazione proprio per consentire di utilizzarla due volte, perché se non facessi la ridenominazione non avrei modo di distinguere i vari attributi tra di loro.

Introduzione su SQL e MySQL-20220309 0807-1

Linguaggi per basi dati

Algebra relazionale: linguaggio procedurale, una espressione dell'algebra specifica come calcolare il risultato.

SQL (Structured Query Language): linguaggio dichiarativo, una espressione SQL specifica quale risultato si vuole raggiungere.

Come sistema di gestione per le basi di dati utilizzeremo MySQL (la versione corrente è la 8, MySQL Community Server).

Cos'è SQL?

Linguaggio di riferimento per le basi di dati relazionali

Acronimo di Structured Query Language Comprende:

Interrogazione: trovare i capi degli impiegati che guadagnano più di 40. Le indicazioni sui capi le abbiamo nella tabella supervisione, quindi per risolvere questa query è congiungere queste due tabelle in modo tale che la matricola dell'impiegato sia uguale al valore che trovo in corrispondenza dell'attributo impiegato in supervisione. Una selezione degli impiegati che guadagnano più di 40 che deve essere messa in join con la tabella supervisione, in particolare un teta join, in cui la condizione è che l'impiegato sia uguale alla matricola, e poi siccome nella query richiede di trovare i capi devo fare la proiezione sull'attributo (siccome sto facendo un teta join nel risultato di questa operazione mi compaiono tutti gli attributi, quindi sei attributi, perché sono quattro della prima tabella e due della seconda).

Interrogazione: trovare nome e stipendio dei capi degli impiegati che guadagnano più di 40. Devo fare più o meno quello che ho fatto prima però quei codici che trovo poi li devo di nuovo utilizzare perché ho bisogno di trovare le informazioni e dei capi visti come impiegati quindi trovare il nome e lo stipendio, quindi devo fare due join. In questo caso voglio sapere qual è il nome e lo stipendio di questi capi, questi valori li devo congiungere di nuovo con la tabella degli impiegati, quindi ho bisogno di un altro join.

$\pi_{\text{Nome}, \text{Stipendio}}(\text{Impiegato} \bowtie_{\text{Matricola} = \text{Capo}} (\pi_{\text{Capo}}(\text{Supervisione} \bowtie_{\text{Impiegato} = \text{Matricola}} (\sigma_{\text{Stipendio} > 40}(\text{Impiegati}))))$

Interrogazione: trovare gli impiegati che guadagnano più del proprio capo, mostrando matricola, nome e stipendio dell'impiegato e del capo. Vogliamo trovare gli impiegati che guadagnano più del loro capo, quindi in questo caso si inverte la richiesta, però la cosa rende non proprio banale l'esercizio è che si vogliono informazioni quali la matricola, il nome e lo stipendio sia dell'impiegato che del capo, quindi ho bisogno di utilizzare sicuramente la tabella impiegato due volte, una volta per quanto riguarda l'impiegato e una volta per quanto riguarda il capo. Ho utilizzato la ridenominazione per modificare il nome degli attributi della tabella impiegati in modo tale che poi non ci siano ambiguità quando io vado a congiungere questa relazione con il risultato della congiunzione che ho fatto in precedenza perché se qui non facessi questa operazione avrei matricola che compare due volte.

$\pi_{\text{Matr}, \text{Nome}, \text{Stip}, \text{MatrC}, \text{NomeC}, \text{StipC}}(\sigma_{\text{Stipendio} > \text{StipC}}(\rho_{\text{MatrC}, \text{NomeC}, \text{EtaC}, \text{StipC} \leftarrow \text{Matricola}, \text{Nome}, \text{Eta}, \text{Stipendio}}(\text{IMPIEGATI}) \bowtie_{\text{MatrC}} \text{Capo}(\text{Supervisione} \bowtie_{\text{Impiegato} = \text{Matricola}} \text{IMPIEGATI))))$

- un Data Definition Language (DDL), ovvero un insieme di comandi per la definizione dello schema di una base di dati relazionale; quell'insieme di comandi che ci permette di definire lo schema della base di dati relazionali no quindi il creare le tabelle con gli vincoli e quindi di creare lo schema quello che abbiamo definito come lo schema della base di dati
- un Data Manipulation Language (DML), ovvero un insieme di comandi per la modifica e l'interrogazione dell'istanza di una base di dati. per la manipolazione dei dati che servono per modificare e interrogare un'istanza di una base di dati quindi quando poi appunto una volta che la base dei dati è stata creata e andrà a popolata e a quel punto potremo eseguire interrogazioni eventualmente fare delle modifiche alla base di dati
- un Data Control Language (DCL), ovvero un insieme di comandi che consentono di controllare o proibire l'accesso ai vari oggetti della base di dati. quindi di controllo dei dati che permettono di controllare o proibire l'accesso a vari oggetti nella base di dati ora su questo aspetto vedremo non vedremo niente perché appunto lavoriamo poi in un ambiente in cui non cose di questo tipo non non ci servono

Un po' di storia su SQL

Prima proposta SEQUEL, Structured English Query Language, 1974: viene sviluppato presso i laboratori IBM di San Jose, con l'obiettivo di farne il linguaggio per la manipolazione dei dati del DBMS relazionale System R.

La diffusione di SQL è dovuta in buona parte alla intensa opera di standardizzazione dedicata a questo linguaggio, svolta principalmente nell'ambito degli organismi ANSI (l'organismo nazionale statunitense degli standard) e ISO (l'organismo internazionale che coordina i vari organismi nazionali).

Cos'è MySQL?

E il più diffuso sistema RDBMS (Relational Database Management System) open source. Queste le sue principali caratteristiche:

- è un software libero fornito in doppia licenza (General Public License (GPL) e licenza commerciale, nel caso si voglia inglobarlo in un sistema commerciale), supportato dalla società MySQL AB, attualmente di proprietà di Oracle;
- è ottimizzato per garantire velocità di esecuzione; questo lo rende particolarmente adatto ad applicazioni via web;
- è disponibile per un'ampia varietà di sistemi operativi: Windows, Linux, MacOS, Solaris, etc;
- è affiancato da un'ampia gamma di software che lo rendono interfacciabile con i più comuni linguaggi e ambienti di programmazione: C, PHP, Java, Python, etc,

Primi passi con MySQL

MySQL è un sistema client-server. Il server mantiene i dati memorizzati in memoria di massa e risponde alle interrogazioni (sia a quelle di selezione che a quelle di modifica dei dati). L'utente non ha una interazione diretta col server, ma accede a quest'ultimo, per via indiretta, adoperando altri programmi, detti client.

Il client è un programma che fa da interfaccia tra l'utente della base di dati e il server. Può essere un client testuale se accetta comandi dall'utente in formato testo (tipicamente comandi SQL) o grafico se presenta una interfaccia visuale.

Convenzioni sintattiche del linguaggio

La sintassi delle espressioni MySQL usa le seguenti convenzioni:

- Le barre verticali | indicano opzionalità.
- Le parentesi quadre [,] indicano che l'espressione all'interno è opzionale. (vuol dire che non siamo obbligati ad inserirla)
- Le parentesi graffe {, } indicano che l'espressione all'interno deve comparire.

Lettere maiuscole e minuscole

Le parole chiave di SQL non sono case-sensitive. Questo è uno standard delle basi di dati. Tuttavia, in MySQL, si deve prestare attenzione all'uso di maiuscole e minuscole quando si fa riferimento ai nomi delle basi di dati e delle tabelle. Infatti, generalmente, ogni base di dati risiede in una propria directory e ciascuna tabella in uno o più file. I nomi di queste directory e di questi file seguono regole diverse a seconda del sistema operativo in cui si trovano. Con Windows non importa se usate lettere maiuscole o minuscole per scrivere i nomi delle basi di dati e delle tabelle. In un sistema basato su Unix la cosa cambia.....

Si deve fare attenzione all'uso di maiuscole e minuscole quando si fa riferimento ai nomi delle basi di dati o ai nomi delle tabelle perché quelli corrispondono a dei file, quindi come tali seguono regole diverse a seconda del sistema operativo in cui stiamo creando la base di dati. È una buona cosa adottare una regola quindi scrivete sempre in maiuscolo in minuscolo in modo che poi non ci siano problemi qualora si volesse cambiare sistema. I nomi degli attributi, degli indici, delle procedure e dei trigger in MySQL non dipendono MAI dall'uso di maiuscole e minuscole.

Il lessico del linguaggio

- Stringhe: sequenza di caratteri (anche caratteri escape) racchiusa tra apici o doppi apici.
- Numeri: sequenza di cifre, eventualmente preceduta da + e da -. Il separatore decimale è il punto.
- Valori Booleani: TRUE e FALSE, sono valutati 1 e 0 rispettivamente.
- Valore NULL: significa assenza di dato. Tale valore è rappresentato da \N nelle operazioni di importazione o esportazione di dati.

Identifieri in MySQL

Un identificatore è semplicemente il nome di una base di dati, di una tabella, di un attributo, di un alias oppure di un indice. Può contenere qualsiasi carattere con le seguenti eccezioni:

- i nomi di basi di dati possono contenere tutti i caratteri che si possono usare per un nome di directory eccetto i caratteri \, / e .
- i nomi di tabella possono contenere tutti i caratteri che si possono usare per un nome di file eccetto i caratteri \, / e .

Tutti gli identifieri, eccetto gli alias, possono avere fino ad un massimo di 64 caratteri. I nomi degli alias possono contenere fino a 255 caratteri.

Una strana regola degli identifieri in MySQL è il fatto che sono ammesse anche parole riservate a patto che siano messe tra `backtick`. Ad esempio, è possibile avere una tabella chiamata TABLE. Esiste anche una lista di parole riservate che è possibile usare come identifieri in MySQL. Gli esempi più comuni sono DATE e TIME usati come nomi di attributi. Il fatto che sia possibile usare parole chiave non significa che sia una buona regola farlo.....

Identifieri multipli: per fare riferimento alla colonna COL NAME della tabella TAB NAME si può usare l'identificatore multiplo TAB NAME.COL NAME. Se non ci sono ambiguità, non è necessario usare gli identifieri multipli.

Commenti: alla fine di una linea: iniziano con # o --spazio su più linee o all'interno di una linea: iniziano con /* e terminano con */

Domini elementari in MySQL

Tipi numerici esatti int, intero a 4 byte, può variare tra -2147483648 e 2147483647, oppure tra 0 e 4294967295. tinyint, intero a 1 byte (sinonimo di bit, bool e boolean), può variare tra -128 e 127, oppure 0 e 255; il valore 0 è considerato falso, un valore non-zero è considerato vero; smallint, mediumint, bigint interi a 2, 3 e 8 byte. numeric[(M,D)], decimal[(M,D)]: numeric e decimal sono sinonimi e sono usati per memorizzare tipi esatti con la virgola e di solito vengono scelti per le valute. M rappresenta il numero totale di cifre e D quello dei decimali.

ecco una cosa che dico tutti tutti gli anni perché poi tanto viene sempre qualcuno l'ha sbaglia sempre allora quando quando si fa riferimento a Tipo intero int enne non indica un'intero AN byte va bene Inter come parole chiave fa sempre riferimento all'intero quattro byte eh quindi li l'indicazione n che compare tra parentesi influenza soltanto la la lunghezza minima che viene che visualizzabile per per quel valore quindi anche in questo caso è appunto è è una questione di visualizzazione e Lenny mi compare tra parentesi non ha niente a che fare con con il numero di byte dedicati a questo a questo tipo è invece tipicamente se io chiedo fate scrivete definito un attributo so di un come un intero la A quattro quattro byte spesso insomma mi viene scritto in 3 4 oppure in un altro valore d'accordo e questa questa opzione poi appunto la cercheremo di evidenziarlo questa questa cosa e se e se il l'attributo definito con questo tipo è stato definito anche con l'opzione zero Phil è quello che succede appunto quando si specifica il valore n dopo Inter che in fase di visualizzazione quel quel valore viene riempito a sinistra con degli zeri qualora il numero delle cifre sia inferiore ad n

I tipi numerici possono essere seguiti dalla parola chiave UNSIGNED e/o ZEROFILL (il numero verrà visualizzato con degli zeri che precedono le cifre significative). Attenzione: int(n) non indica un intero a n byte. L'indicazione del parametro n sugli interi non influisce sui valori memorizzati, ma rappresenta la lunghezza minima visualizzabile per il dato. Se il valore occupa meno

cifre, viene riempito a sinistra con degli zeri, se è stata usata l'opzione ZEROFILL. A partire da MySQL 5.0.3, esiste anche il tipo BIT: BIT(M) permette di memorizzare M bit ($1 \leq M \leq 64$).

Tipi numerici approssimati

- FLOAT: indica un numero reale a virgola mobile e precisione semplice; valori possibili sono tra -3.402823466E+38 e -1.175494351E-38, 0, e tra 1.175494351E-38 e 3.402823466E+38.
- DOUBLE: indica un numero reale a virgola mobile e precisione doppia; valori possibili sono tra -1.7976931348623157E+308 e -2.2250738585072014E-308, 0, e tra 2.2250738585072014E-308 e 1.7976931348623157E+308

Tipi stringa e testo

- CHAR: viene usato per memorizzare stringhe di lunghezza fissa, di solito seguito da una indicazione di lunghezza (senza indicazione di lunghezza corrisponde a char(1)). La lunghezza massima è di 255 caratteri.
- VARCHAR: viene usato per memorizzare stringhe a lunghezza variabile, deve essere seguito da una indicazione di lunghezza massima.
- TEXT,BLOB: servono per memorizzare testi più lunghi di quanto si possa fare con i tipi precedenti. Il tipo BLOB andrebbe usato per memorizzare dati binari più che testi veri e propri e inoltre è case-sensitive, a differenza del tipo TEXT. Sono entrambi a lunghezza variabile e hanno diversi sottotipi. Hanno lunghezza massima $2^{16}-1 = 65535$.
- ENUM('value1','value2',...): permette di definire una stringa che può assumere un valore tra 'value1', 'value2', ..., NULL o error (la stringa vuota).
- SET('value1','value2',...): permette di definire una stringa che può assumere zero o più valori tra 'value1', 'value2', ...

Tipi data e tempo

- DATE: memorizza una data nel formato AAAA-MM-GG.
- TIME: memorizza un dato temporale come HH:MM:SS.
- DATETIME: è una combinazione dei due precedenti. Il formato è AAAA-MM-GG HH:MM:SS.
- TIMESTAMP: in corrispondenza di questo tipo viene automaticamente memorizzato il momento in cui la riga è stata inserita o modificata l'ultima volta. Un dato di questo tipo viene visualizzato nel formato di DATETIME.
- YEAR: questo tipo memorizza un anno a 2 o 4 cifre (YEAR(2) o YEAR(4)). In particolare, valori negli intervalli 00-69 e 70-99 sono convertiti in valori in 2000-2069 e 1970-1999.

Valori DEFAULT

Nella definizione di tipi di dati si può includere la clausola DEFAULT che indica il valore di default per quell'attributo: tale valore deve essere una costante, non può essere una funzione o una espressione; l'unica eccezione ammessa è il valore CURRENT_TIMESTAMP per una colonna di tipo TIMESTAMP.

Se la clausola non viene specificata, MySQL determina il valore di default seguendo le seguenti regole:

- se l'attributo accetta il valore NULL, questo è preso come default;
- per i tipi numerici lo 0, eccetto per i numeri definiti con l'attributo AUTO INCREMENT per i quali il default è il valore successivo nella sequenza;
- per i tipi data, lo 0 appropriato;
- per le stringhe, la stringa vuota;
- per il tipo enum, il primo valore dell'elenco;
- BLOB e TEXT non possono avere un valore default.

Definizione di domini

MySQL rispetta quasi totalmente lo standard SQL. Cercheremo di evidenziare le situazioni in cui questo non è vero. Ad esempio, lo standard SQL permette di definire un dominio a partire da domini predefiniti: CREATE DOMAIN AS TipodiDato [ValoreDiDefault] [Vincolo] CREATE DOMAIN AS VOTOSAME DEFAULT NULL CHECK (VALUE >= 18 AND VALUE <= 30) Questa istruzione non è supportata da MySQL, vedremo più avanti come sia possibile realizzare un vincolo di questo tipo.

Creazione di una base di dati in MySQL

- SHOW DATABASES; mostra le basi di dati presenti nel sistema.

- CREATE DATABASE [IF NOT EXISTS] Nomedb; crea un nuovo database. Notare che la creazione di un database aggiunge una nuova directory in c:\ProgramData\MySQL\Data.
- DROP DATABASE Nomedb; cancella un database.
- USE Nomedb; usa un database.
- SELECT DATABASE(); visualizza il nome del database corrente.

Nello standard SQL esistono invece istruzioni del tipo CREATE SCHEMA Nomeschema, DROP SCHEMA Nomeschema.

Creazione di tabelle

L'istruzione CREATE: definisce uno schema di relazione e ne crea un'istanza vuota; specifica attributi, domini e vincoli; la sintassi più semplice `e:

```
CREATE TABLE [IF NOT EXISTS ] NomeTb
  (NomeCol Dominio[Default][Vincoli], . . . )
  [ENGINE = ];

CREATE TABLE Dipartimento (
  Nome CHAR(20) PRIMARY KEY,
  Indirizzo CHAR(50),
  Sede CHAR(20));
```

Vincoli intrarelazionali

- NOT NULL: questo vincolo indica che il valore nullo non `e ammesso come valore dell'attributo: Cognome CHAR(20) NOT NULL;
- UNIQUE: questo vincolo si applica ad un attributo o un insieme di attributi di una tabella e impone che i valori dell'attributo siano una (super)chiave, cio`e righe differenti della tabella non possono avere gli stessi valori:

```
Matricola CHAR(6) UNIQUE;
Nome CHAR(20),
Cognome CHAR(20),
UNIQUE(Cognome, Nome)
```

- PRIMARY KEY: questo vincolo definisce la chiave primaria di una relazione:

```
Matricola CHAR(6) PRIMARY KEY;
```

oppure, in modo equivalente:

```
Matricola CHAR(6);
. . . ,
PRIMARY KEY(Matricola);
```

- Il vincolo PRIMARY KEY `e spesso usato insieme al modificatore di tipo AUTO INCREMENT, che pu`o essere utilizzato per generare una sequenza di numeri. Il nuovo valore generato automaticamente sar`a sempre maggiore di una unit`a rispetto al valore pi` grande presente in quel momento nella tabella. La prima riga inserita avrà valore 1.

```
IdDipartimento
INT AUTO INCREMENT PRIMARY KEY;

· CREATE TABLE Prova (
  · i INT(3) ZEROFILL,
  · j INT(6) ZEROFILL,
  · k INT(11) ZEROFILL);
  · INSERT INTO Prova (i, j, k) VALUES (123, 456, 789); SELECT * FROM Prova;
```

Si inseriscono dei valori interi in questa tabella per vedere il risultato: quello che vedremo è che cambia essenzialmente la visualizzazione di questi numeri, nel primo caso trattandosi di numeri con tre cifre verrà semplicemente visualizzato 123, nel secondo caso invece 456 sarà preceduto da tre zeri, mentre 789 sarà preceduto da 8+3 quindi 8 zeri per arrivare a 11. Una prima istruzione di inserimento: insert into, quindi inseriamo nella tabella prova e specifichiamo gli attributi che vogliamo andare ad inserire nella tabella. select* from prova, select è l'istruzione che ci permette di eseguire interrogazioni sulla base di dati, quindi select è istruzione fondamentale del linguaggio SQL. Questo * in questo caso indica tutto, un modo abbreviato per indicare che vogliamo vedere tutto il contenuto della tabella.

- CREATE TABLE Studente(
- Matricola INT,
- Cognome VARCHAR(20),
- Nome VARCHAR(20),
- Nascita DATE,
- Corso ENUM('Informatica','Matematica','Fisica'));

ho creato una tabella studente in cui ho una matricola di tipo intero, cognome e nome sono tutti e due varchar di lunghezza massima 20, poi ho l'attributo nascita che è di tipo date e poi corso frequentato dallo studente, in questo caso ho utilizzato un tipo relativo che mi permette di scegliere tra informatica matematica e fisica.

L'istruzione SELECT

Permette di selezionare tuple da una o più tabelle.

La sintassi semplificata è: SELECT lista attributi o espressioni [FROM lista tabelle] [WHERE condizioni]

Può essere usata anche come operatore di calcolo nelle forme: SELECT espressioni

L'istruzione è fondamentale per fare interrogazioni e permette anche di fare selezioni di record sia da una singola tabella che da più tabelle. Dopo select dobbiamo specificare la lista di attributi o anche delle espressioni poi abbiamo una parte from lista tabelle, quindi una clausola from che ci permette di specificare quali sono le tabelle che vogliamo utilizzare per l'interrogazione, e poi c'era un'altra clausola fondamentale che è quella di where in cui andiamo a specificare le condizioni. Sia la clausola from che la clausola where sono tra parentesi quadre, quindi possiamo utilizzare il comando select anche come un operatore di calcolo senza specificare tabelle, quindi condizioni. Possiamo semplicemente utilizzare l'operatore nella forma select specificando un'espressione e l'effetto è quella di calcolare di valutare l'espressione in corrispondenza.

Algebra relazionale-20220315 0801-1

IMPIEGATI			
Matricola	Cognome	Filiale	Eta
7309	Rossi	Firenze	34
5998	Bianchi	Roma	45
9553	Neri	Milano	NULL

$\sigma_{Eta > 40}(Impiegati)$

La condizione atomica è vera solo per valori non nulli: come viene considerato l'impiegato Neri di cui non si conosce l'età?

maniera esplicita quando si fanno queste selezioni, quindi viene restituito semplicemente il record che corrisponde a bianchi, che è l'unico in questa selezione a soddisfare la condizione sull'età.

Un risultato non desiderabile

$\sigma_{Eta > 40}(Impiegati) \cup \sigma_{Eta \leq 40}(Impiegati) \neq Impiegati$

Perchè? Perché le selezioni vengono valutate separatamente!

$\sigma_{Eta > 40} \text{ or } \sigma_{Eta \leq 40}(Impiegati) \neq Impiegati$

Perchè? Perché anche le condizioni atomiche vengono valutate separatamente!

Valori nulli in algebra relazionale

Si usa la tradizionale algebra a due valori. Un valore nullo rende falso ogni predicato atomico $A \theta B$, $A \theta \text{cost}$. Introduzione di nuovi prediciati:

- A IS NULL: assume valore vero su una tupla t se il valore di t su A è nullo.

Abbiamo una tabella impiegati in cui compare un valore nullo in corrispondenza dell'età, quindi per l'impiegato neri non sappiamo l'età. Nei sistemi di gestione di basi di dati l'uso del valore in un punto permette di gestire situazioni in cui non si conosce un certo valore oppure non lo si sa ancora, quindi l'assenza di informazione. Che cosa succede in presenza di valori nulli quando si vanno a fare delle interrogazioni? Se richiedo una selezione su una tabella di questo tipo, con condizione età maggiore di 40, quello che succede in corrispondenza di valori nulli è che questi non vengono considerati in

- A IS NOT NULL: assume valore vero su una tupla t se il valore di t su A è specificato.

IMPIEGATI			
Matricola	Cognome	Filiale	Eta
7309	Rossi	Firenze	34
5998	Bianchi	Roma	45
9553	Neri	Milano	NULL

$\sigma_{(Eta > 40) \text{ or } Eta \text{ is null}}$ (IMPIEGATI)			
Matricola	Cognome	Filiale	Eta
5998	Bianchi	Roma	45
9553	Neri	Milano	NULL

esempio: se io faccio una selezione sua età maggiore di 40 oppure età è null, questa selezione mi selezionerà sia bianchi che neri e per fare in modo di avere la tabella intera

$\sigma_{Eta > 40}(\text{Impiegati}) \cup \sigma_{Eta \leq 40}(\text{Impiegati}) \cup \sigma_{Eta \text{ is null}}(\text{Impiegati}) = \sigma_{Eta > 40 \text{ or } Eta \leq 40 \text{ or } Eta \text{ is null}}(\text{Impiegati}) = \text{Impiegati}$

i valori nulli hanno un trattamento particolare quindi per poterli gestire dobbiamo utilizzare questi due predicati diversi dagli operatori classici.

Equivalenza di espressioni

Due espressioni sono equivalenti se producono lo stesso risultato qualunque sia l'istanza attuale della base di dati. L'equivalenza è importante in pratica perché i DBMS cercano di eseguire espressioni equivalenti a quelle date, ma meno costose.

Un'equivalenza importante

Anticipare la selezione rispetto al join.

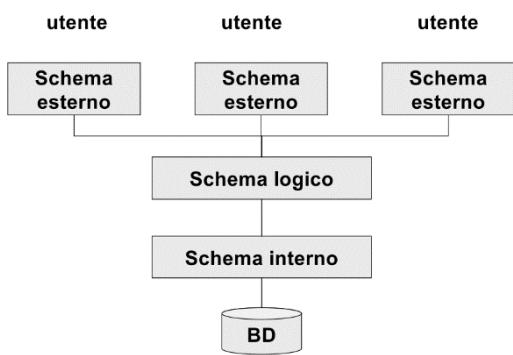
Esempio (se A è attributo di R2): $\sigma_{A=10}(R1 \bowtie R2) = R1 \bowtie \sigma_{A=10}(R2)$

Riduce in modo significativo la dimensione del risultato intermedio (e quindi il costo dell'operazione). È chiaro che la seconda soluzione è più efficiente perché vado a fare un join su una tabella che ha un numero di record inferiore.

Viste, o relazioni derivate

- Rappresentazioni diverse per gli stessi dati (schema esterno).
- Relazioni di base: contenuto autonomo.
- Relazioni derivate: relazioni il cui contenuto è funzione del contenuto di altre relazioni (definito per mezzo di interrogazioni).
- Le relazioni derivate possono essere definite su altre derivate.

Le realzioni, quindi tabelle, possono essere viste come delle relazioni di base. Possono esistere anche delle relazioni derivate cioè relazioni il cui contenuto è funzione del contenuto di altre relazioni e in particolare il contenuto di queste relazioni derivate è definito per mezzo di interrogazioni che si fanno sulle relazioni di base.



Qual è l'architettura secondo la quale organizzato un sistema di gestione di basi di dati? Nella base di dati ci sono tre livelli, nel primo livello abbiamo lo schema interno (lo schema fisico, quindi entrano in gioco proprio le strutture fisiche con cui è realizzata la base di dati), poi lo schema logico cioè lo schema del modello logico (nel nostro caso del modello relazionale) che si sta utilizzando e infine c'è uno schema finale che è quello esterno, che può essere diverso a seconda dell'utente che va a utilizzare la base di dati. In questo terzo livello entrano in gioco le viste, perché le viste sono uno strumento che permette di realizzare delle relazioni derivate da quelle dello schema principale e che permettono quindi di ottenere dei sottoschemi per utenti diversi.

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

Immaginiamo di avere queste due tabelle reali, quindi relazioni base. Afferenza definita sulla coppia di attributi impiegato e reparto, poi una relazione direzione definita su reparto e capo. Sono due relazioni e quindi da un punto di vista pratico ci saranno due file che corrispondono ad ognuna di queste tabelle. Che cos'è la vista? Ad esempio ad un certo utente non mi interessa far vedere l'indicazione sul reparto ma voglio far vedere per ogni impiegato qual è il suo capo, quindi tramite lo strumento vista posso creare una tabella virtuale, in cui evidenzio qual è la relazione tra impiegato e capo tramite un join naturale tra afferenza e direzione, seguito da una proiezione su impiegato e capo, quindi in questo caso andrei a coniugare Rossi con Mori, Neri con Bruni e Bianchi con Bruni tramite il

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

evidenzio qual è la relazione tra impiegato e capo tramite un join naturale tra afferenza e direzione, seguito da una proiezione su impiegato e capo, quindi in questo caso andrei a coniugare Rossi con Mori, Neri con Bruni e Bianchi con Bruni tramite il

valore dell'attributo reparto. La vista è un oggetto virtuale, quindi non c'è una tabella che corrisponde a questa definizione quando eseguiamo la creazione di una vista, ma semplicemente facciamo un'interrogazione e tale interrogazione assume le sembianze di una tabella, quindi a livello di lettura la possiamo trattare come fosse una qualsiasi altra tabella, a livello di scrittura le cose sono un pochino più delicate. Il concetto di vista ci serve soprattutto per semplificare la scrittura delle interrogazioni vedete quando io definisco un'interrogazione in pratica le posso associare di un nome, come in questo caso supervisione, quindi rappresentare espressioni dell'algebra in modo più semplice.

Viste virtuali e materializzate

Due tipi di relazioni derivate:

- viste materializzate (viene considerata come una tabella);
- relazioni virtuali (o viste).

La differenza con la tabella normale è che il contenuto di una vista si deve aggiornare in base all'interrogazione, quindi se cambiano le tabelle di base deve cambiare anche il contenuto della derivata.

Viste materializzate

- Relazioni derivate memorizzate nella base di dati.
- Vantaggi: immediatamente disponibili per le interrogazioni.
- Svantaggi: ridondanti, appesantiscono gli aggiornamenti, sono raramente supportate dai DBMS.

Viste virtuali (o viste)

- Sono supportate dai DBMS (tutti)
- Una interrogazione su una vista viene eseguita ricalcolando la vista.

Quindi tutte le volte che si fa riferimento ad una vista, l'interrogazione deve essere ricalcolata, quindi il contenuto della vista è dinamico e dipende dal valore di tali tabelle che ci stanno dietro.

Interrogazioni sulle viste

Sono eseguite sostituendo alla vista la sua definizione. Ad esempio: $\sigma_{Capo=Bruni}(Supervisione)$ viene eseguita come $\sigma_{Capo=Bruni}(\pi_{Impiegato,Capo}(\text{Afferenza } \bowtie \text{ Direzione}))$

Viste, motivazioni

- Schema esterno: ogni utente vede solo ciò che gli interessa e nel modo in cui gli interessa, senza essere distratto dal resto; ciò che è autorizzato a vedere (autorizzazioni).
- Strumento di programmazione: si può semplificare la scrittura di interrogazioni: espressioni complesse e sottoespressioni ripetute. Utilizzo di programmi esistenti su schemi ristrutturati.
- Invece: L'utilizzo di viste non influisce sull'efficienza delle interrogazioni.

I motivi per introdurre il concetto di vista: a livello di schema esterno questo strumento permette di far sì che ogni utente vedrà soltanto una porzione della base di dati, quella che gli compete, e sono anche uno strumento di interrogazione, perché permettono di semplificare la scrittura delle interrogazioni sia a livello di algebra ma anche quando si usa il linguaggio. Sono uno strumento importante quando esistono dei programmi che si interfacciano su basi di dati che decidiamo per esempio di ristrutturare quindi se lo schema viene ristrutturato il programma può in alcune situazioni continuare a funzionare perché è possibile creare una vista che ricrea la struttura dello schema originale. Se quel programma avrà bisogno di accedere alla base di dati in lettura è facile, se ci accede in scrittura la situazione è un pochino più delicata.

Viste come strumento di programmazione

Trovare gli impiegati che hanno lo stesso capo di Rossi.

- Senza vista:

$$\pi_{Impiegato} ((Afferenza \bowtie Direzione) \bowtie \rho_{ImpR,RepR \leftarrow Imp,Reparto} (\sigma_{Impiegato=Rossi}(Afferenza \bowtie Direzione)))$$

- Con la vista:

$$\pi_{Impiegato} (Supervisione \bowtie \rho_{ImpR \leftarrow Imp} (\sigma_{Impiegato=Rossi}(Supervisione)))$$

Viste e aggiornamenti

AFFERENZA		DIREZIONE	
Impiegato	Reparto	Reparto	Capo
Rossi	A	A	Mori
Neri	B	Bruni	
Verdi	A	C	Bruni

SUPERVISIONE	
Impiegato	Capo
Rossi	Mori
Neri	Bruni
Verdi	Bruni

Aggiornare una vista: modificare le relazioni di base in modo che la vista, ricalcolata rispecchi l'aggiornamento. L'aggiornamento sulle relazioni di base corrispondente a quello specificato sulla vista deve essere univoco (altrimenti crea ambiguità). In generale però non è univoco! Ben pochi aggiornamenti sono ammissibili sulle viste. Una vista su una singola tabella è una vista aggiornabile, tramite la vista posso fare delle modifiche anche sulla tabella base.

Vogliamo inserire, nella vista, il fatto che Lupi ha come capo Bruni; oppure che Belli ha come capo Falchi; come facciamo? Non conosciamo l'informazione relativa al Reparto!

Algebra relazionale: limiti

- Ci sono interrogazioni interessanti non esprimibili.
- Calcolo di valori derivati: possiamo solo estrarre valori, non calcolarne di nuovi.
- Calcoli di interesse: a livello di ennupla o di singolo valore (conversioni, somme, differenze, etc.) su insiemi di ennupla (somme, medie, etc.) Le estensioni sono ragionevoli, le vedremo in SQL.
- Interrogazioni inerentemente ricorsive, come la chiusura transitiva.

Chiusura transitiva

Data r di schema $R(X,Y)$, la chiusura transitiva r^* di r è la relazione che si ottiene aggiungendo, fino a che è possibile, alle tuple in r la coppia (a, b) se esiste un valore c tale che le coppie (a, c) e (c, b) sono in r o sono state aggiunte precedentemente.

- Per ogni impiegato, trovare tutti i superiori (cioè il capo, il capo del capo, e così via).

SUPERVISIONE		SUPERVISIONE2	
Impiegato	Capo	Impiegato	Superiore
Rossi	Lupi	Rossi	Lupi
Neri	Bruni	Neri	Bruni
Lupi	Falchi	Lupi	Falchi
		Rossi	Falchi

- In questo esempio, basta il join della relazione con se stessa, previa opportuna ridenominazione.

Che cos'è la chiusura transitiva? È una relazione che si ottiene unendo alle tuple di R tutte le tuple del tipo (a,b) che io riesco a creare a partire dalle tuple (a,c) e (c,b) che sono eventualmente già presenti nella tabella e vado avanti finché posso. Esempio: Per ogni impiegato voglio trovare tutti i superiori, cioè il capo del capo del capo e così via, e quindi abbiamo la tabella supervisione con attributi impiegato e capo, e quindi per ogni impiegato io so qual è il suo capo. Rossi ha come capo Lupi, Lupi ha capo Falchi, quindi Rossi ha come superiori sia Lupi che Falchi. Voglio esprimere un'interrogazione che mi permetta di avere questo risultato. Creo

una copia della tabella supervisione, chiamata supervisione2, cioè una vista supervisione che va a riprendere esattamente lo stesso contenuto della tabella originale. Siccome ho due coppie Rossi-Lupi e Lupi-Falchi, devo coniugare queste due coppie e ottenere la coppia Rossi-Falchi, quindi aggiungo Rossi-Falchi e in questo caso questa tabella che vedete a destra corrisponde al risultato della chiusura transitiva, perché non ci sono altre coppie che posso coniugare. Possiamo scrivere questa interrogazione in algebra in questo modo:

$\text{Copiasupervisione} = \rho_{\text{ImpX}, \text{CapoX} \leftarrow \text{Impiegato}, \text{Capo}}(\text{Supervisione})$

$\text{SuperSuper} = \pi_{\text{Impiegato}, \text{CapoX}}(\text{Supervisione} \bowtie_{\text{Capo}=\text{ImpX}} \text{Copiasupervisione})$

$\text{Supervisione2} = \rho_{\text{Superiore} \leftarrow \text{Capo}}(\text{Supervisione}) \cup \rho_{\text{Superiore} \leftarrow \text{CapoX}}(\text{SuperSuper})$

Ho creato prima una copia di supervisione e l'ho chiamata Copiasupervisione, in cui ho considerato la relazione supervisione e ho ridenominato gli attributi in impX e capoX. Ho creato una vista che ho chiamato SuperSuper in cui ho fatto una coniugazione tra supervisione e la copia di supervisione con condizione capo uguale impX. La vista supervisione2 corrisponde a una ridenominazione di supervisione in cui ho cambiato capo in superiore e a un'unione delle tuple che erano in supervisione e la vista SuperSuper, che è quella che contiene le tuple che corrispondono al join naturale della tabella supervisione con se stessa.

SUPERVISIONE	
Impiegato	Capo
Rossi	LUPI
Neri	Bruni
LUPI	FALCHI
FALCHI	LEONI

SUPERVISIONE2	
Impiegato	Superiore
Rossi	LUPI
Neri	Bruni
LUPI	FALCHI
FALCHI	LEONI
Rossi	Rossi
LUPI	LUPI
LEONI	LEONI
LEONI	LEONI

In questo esempio, i join necessari sarebbero due!

Non posso sapere in anticipo non conoscendo qual è il contenuto della tabella quanti join devo fare per poter realizzare la chiusura ricorsiva. Non esiste una query in algebra relazionale che mi permetta di risolvere in generale il problema. Quando si scrivono interrogazione in algebra si scrive indipendentemente dal contenuto cioè io devo scrivere un'espressione mi permetta di risolvere quell'interrogazione qualsiasi sia il contenuto della tabella invece in questo caso non lo posso fare. È un limite della dell'algebra relazionale, che ad esempio si può risolvere tramite l'uso di strutture, che possono essere che l'uso del ciclo nel linguaggio SQL.

Chiusura transitiva, impossibile!

Non esiste in algebra la possibilità di esprimere l'interrogazione che, per ogni relazione binaria, ne calcoli la chiusura transitiva. Per ciascuna relazione, è possibile calcolare la chiusura transitiva, ma con un'espressione ogni volta diversa. Quanti join servono? Non c'è limite!

Esercizi

Disco			
disco_id	titolo	casa_disc	genere
1	Titolo1	Casa1	Generel
2	Titolo2	Casa1	Gener2
3	Titolo3	Casa2	Gener2
4	Titolo4	Casa1	Gener1
5	Titolo5	Casa3	Gener2
6	Titolo6	Casa1	Gener3
7	Titolo7	Casa3	Gener2
8	Titolo8	Casa2	Gener3
9	Titolo9	Casa3	Gener1
10	Titolo10	Casa2	Gener1

Autore		
autore_id	nome	nazionalita
1	Nome1	Nazione1
2	Nome2	Nazione2
3	Nome3	Nazione3
4	Nome4	Nazione1
5	Nome5	Nazione2
6	Nome6	Nazione3
7	Nome7	Nazione1
8	Nome8	Nazione2
9	Nome9	Nazione3
10	Nome10	Nazione1

Composto_da	
disco	autore
1	2
2	1
2	2
3	7
3	10
4	4
5	8
6	1
7	2
7	6
8	9
9	2
10	5
10	7
10	9

Negozio			
p_iva	nome	indirizzo	citta
Piva1	Negozi1	Indirizzo1	Citta1
Piva2	Negozi2	Indirizzo2	Citta2
Piva3	Negozi3	Indirizzo3	Citta1
Piva4	Negozi4	Indirizzo4	Citta3
Piva5	Negozi5	Indirizzo5	Citta2
Piva6	Negozi6	Indirizzo6	Citta3

Vendita		
p_iva	disco_id	copie
Piva1	1	3
Piva1	3	2
Piva2	2	1
Piva3	4	4
Piva4	5	2
Piva5	6	3
Piva1	7	1
Piva2	8	5
Piva3	9	3
Piva4	10	2
Piva5	2	6
Piva2	3	7
Piva3	6	5
Piva4	7	3
Piva5	3	5
Piva5	10	5

Abbiamo una base di dati che contiene queste 5 tabelle, disco, autore, composto da, negozio e vendita. Per ogni disco abbiamo un attributo disco_id, che rappresenta la chiave della relazione, titolo, la casa discografica e il genere. Per gli autori abbiamo il codice dell'autore, il nome e la nazionalità. Una relazione composta da che ci dice per ogni disco qual è l'autore o quali sono gli autori. Per i negozi abbiamo il riferimento alla partita iva, il nome, l'indirizzo e la città. Una tabella vendita che per ogni negozio e per ogni disco ci dice qual è il numero delle copie che sono state vendute di quel disco. Ovviamente ci sono dei vincoli di integrità che non ho esplicitato ma che sono presenti naturalmente con questo schema e ovviamente in composto da, disco si riferisce al codice del disco, e autore all' ide dell'autore e nella vendita, la partita iva è la partita iva di un negozio e il codice del disco è un disco.

Nome degli autori di nazionalità Nazione1: si tratta di fare una selezione dalla tabella autore con condizione nazionalità uguale a nazione1 e poi fare la proiezione, quindi proiettiamo sul nome perché è quello che è richiesto.

$\pi_{\text{Nome}}(\sigma_{\text{nazionalita}=\text{'Nazione1'}}(\text{Autore}))$

Partita iva dei negozi che del disco di codice 3 hanno venduto più di 2 copie: Primo passo: capire quali sono le tabelle coinvolte nell'interrogazione, qui si tratta di trovare la partita iva dei negozi che del disco di codice tre hanno venduto più di due copie e siccome nella tabella della vendita abbiamo la partita iva del negozio, il codice del disco e le copie, quindi con questa tabella possiamo rispondere all'interrogazione, quindi non abbiamo bisogno di andare a prendere altre tabelle. basta fare una selezione con condizione disco_di uguale a tre e copie maggiori di due dalla tabella vendita e poi la proiezione sulla partita iva.

$\pi_{\text{p_iva}}(\sigma_{\text{disco_id}=3 \text{ and } \text{copie}>2}(\text{Vendita}))$

Nome, indirizzo e città dei negozi che del disco di codice 3 hanno venduto più di 2 copie: Simile al precedente questa volta però non ci basta avere il codice del negozio e vogliamo anche il nome, l'indirizzo e la città, quindi è chiaro che in questo caso le informazioni che sono presenti nella tabella vendita non sono più sufficiente. Abbiamo bisogno anche delle informazioni su il negozio, quindi in questo caso è necessario fare un join naturale. La tabella che ci restituisce la partita iva dei negozi che del disco 3 hanno venduto più di due copie la mettiamo in join con la tabella negozio. Ci va bene un join naturale in questo caso perché negozio e vendita hanno proprio lo stesso attributo partita iva per cui se le voglio congiungere

Per trovare la chiusura ricorsiva che cosa dovrei fare? Dovrei aggiungere alle tuple già presenti in supervisione quali altre tuple? Sicuramente la stessa di prima, quindi Rossi-Falchi. La tabella supervisione contiene anche il record falchi-leoni. Lupi ha come capo Falchi e falchi ha come capo Leoni, ciò vuol dire che Lupi ha come superiore Leoni, però questo mi dice anche che Rossi ha come superiore leoni. Mentre le tuple Rossi-falchi e Lupi-leoni si trovano facendo un join della tabella supervisione con se stessa, per determinare la coppia Rossi-leoni devo fare più join.

posso fare un join naturale. Poi faccio la proiezione sul nome, indirizzo e città. Ho scritto l'interrogazione in maniera un po' diversa: fare la selezione su disco_id=tre e copie maggiori due della tabella vendita senza fare la proiezione e poi congiungere con negozio e infine fare la proiezione successiva (è un esempio di due espressioni equivalenti perché danno luogo allo stesso risultato però sono ottenute in modo leggermente diverso).

$$\pi_{\text{nome}, \text{indirizzo}, \text{città}}(\text{Negozio} \bowtie \pi_p \text{iva}(\sigma_{\text{disco id}=3 \text{ and } \text{copie}>2}(\text{Vendita})))$$

$$\pi_{\text{nome}, \text{indirizzo}, \text{città}}(\text{Negozio} \bowtie \sigma_{\text{disco id}=3 \text{ and } \text{copie}>2}(\text{Vendita}))$$

Nome, indirizzo e città dei negozi che hanno venduto più di 2 copie di un disco di titolo Titolo3: ho un riferimento al titolo del disco ma le congiunzioni tra le tabelle le posso fare tramite il codice del disco. Devo utilizzare anche la tabella che contiene le informazioni sul disco in modo tale che dal titolo risalgo al codice e poi mi riconduco ai casi precedenti, quindi faccio il join naturale tra disco e vendita, in questo caso le due tabelle condividono l'attributo disco_id e quindi possono di nuovo fare un join naturale e questa volta ho fatto una selezione sul titolo e il numero delle copie. Poi ho fatto la proiezione sulla partita iva e poi ho fatto esattamente come prima andando a congiungere con negozio, per poi fare la proiezione.

$$\pi_{\text{nome}, \text{indirizzo}, \text{città}}(\text{Negozio} \bowtie \pi_p \text{iva}(\sigma_{\text{titolo}=\text{'Titolo3'}} \text{ and } \text{copie}>2)(\text{Disco} \bowtie \text{Vendita}))$$

Partita iva dei negozi che hanno venduto più di 4 copie di un qualche disco.

$$\pi_p \text{iva}(\sigma_{\text{copie}>4}(\text{Vendita}))$$

Partita iva dei negozi che hanno venduto 4 copie, o meno di 4 copie, di un qualche disco.

$$\pi_p \text{iva}(\sigma_{\text{copie}\leq 4}(\text{Vendita}))$$

Partita iva dei negozi che di ogni disco che hanno venduto, ne hanno vendute più di 2 copie: Innanzitutto devo capire bene che cosa voglio trovare: voglio i negozi che hanno la caratteristica particolare di aver venduto un numero di copie maggiore di due se hanno venduto un disco. Voglio esprimere un'interrogazione mi permetta di trovare i negozi 10 e 20, cioè i due negozi che soddisfano la condizione, ma non voglio trovare il negozio 30, perché non tutti i dischi che 30 ha venduto soddisfano la condizione sul numero delle copie. Come si fa a rispondere a un'interrogazione di questo tipo? L'idea è che io devo riuscire a trovare i negozi che sicuramente non soddisfano la condizione in modo che poi per differenza io trovo tutti gli altri, quindi devo trovare i negozi che hanno venduto almeno un disco con un numero di copie inferiore a soglia prefissata e quel negozio lo dovrò sottrarre da tutti gli altri. Devo fare una selezione dalla tabella vendita con copie minore uguale di due, cioè io vado a vedere quali sono i record che corrispondono al numero di copie minori uguali di due. I negozi che corrispondono a questi record li devo togliere dalla lista complessiva di negozi, quindi quando vado a fare la differenza tra la relazione vendita e la selezione sempre da vendita con copia minori uguali due, ottengo soltanto i record che corrispondono a selezioni che soddisfano la condizione. Alla fine dovrò fare semplicemente una proiezione sulla partita iva per trovare quali sono i negozi che hanno questa caratteristica. Attenzione quando fate delle differenze, perché le differenze deve essere fatta su schemi uguali, non posso fare la differenza tra due tabelle che hanno una forma diversa.

$$\pi_p \text{iva}(\text{Vendita}) - \pi_p \text{iva}(\sigma_{\text{copie}\leq 2}(\text{Vendita}))$$

Da tutti i negozi vengono tolti quelli che di qualche disco hanno venduto 2 copie o meno.

I titoli dei dischi di cui almeno un autore è di nazionalità Nazione2: Voglio sapere quali sono i dischi che hanno la caratteristica di avere almeno un autore di nazionalità nazione2, quindi devo fare diverse congiunzioni perché l'informazione sul titolo del disco è nella tabella disco e l'informazione sulla nazionalità è nella tabella autore e poi ho la tabella composto da, che è la tabella che mi dice di ogni disco quali sono gli autori, quindi ho bisogno di queste tre tabelle. In corrispondenza di autore ho fatto la selezione con nazionalità uguale a nazione2 e poi la congiunzione tra queste tabelle è stata fatta con un teta join perché gli attributi disco e autore hanno dei nomi diversi nella tabella composto da. La condizione che almeno un autore sia di nazionalità nazione2 la risolvo semplicemente facendo una congiunzione perché se ce n'è almeno uno che ha questa caratteristica io con questa operazione lo seleziono sicuramente.

$$\pi_{\text{titolo}}(\text{Disco} \bowtie_{\text{disco id}} (\text{Composto da} \bowtie_{\text{autore=autore id}} (\sigma_{\text{nazionalità}=\text{'Nazione2'}}(\text{Autore}))))$$

I codici dei dischi di cui almeno un autore ha nazionalità diversa da Nazione2: Lo stesso succede se voglio sapere quali sono i codici dei dischi di cui almeno un autore ha nazionalità diversa da nazione2, quindi rispetto a prima la query è più semplice perché non faccio riferimento al titolo ma direttamente al codice del disco e anche in questo caso basta che io vado a fare una congiunzione tra autore e composto da, e dopo faccio una selezione con nazionalità diverso nella nazione2.

$$\pi_{\text{disco id}} (\text{Composto da} \bowtie_{\text{autore=autore id}} (\sigma_{\text{nazionalità}\neq\text{'Nazione2'}}(\text{Autore})))$$

I codici dei dischi di cui tutti (quindi si risolve tramite differenza) gli autori hanno nazionalità Nazione2: Voglio selezionare un disco solo se tutti gli autori hanno quella nazionalità. Questo è un tipo di interrogazione si risolve tramite differenza, l'idea è simile a quella di prima, ovvero trovo i dischi che sicuramente non hanno questa caratteristica e li sottraggo da tutti quelli invece che mi rimangono. Come faccio a trovare i dischi che sicuramente non hanno questa caratteristica? Nella parte che vado a togliere è la proiezione sul codice del disco della congiunzione tra composto da e autore con selezione nazionalità diverse da nazione2, in pratica vado a vedere quali sono i dischi in cui almeno un autore ha una nazionalità diversa e quindi quel disco sicuramente non mi va bene. Tutto quello che ottengo facendo questa differenza invece sicuramente soddisfa la condizione che sto richiedendo. Da tutti i rischi vengono tolti quelli per cui almeno un autore ha nazionalità diversa.

$$\pi_{\text{disco}}(\text{Composto da}) - \pi_{\text{disco}}(\text{Composto da} \bowtie_{\text{autore}=\text{autore id}} (\sigma_{\text{nazionalita} \neq \text{'Nazione2'}}(\text{Autore})))$$

Da tutti i dischi vengono tolti quelli per cui almeno un autore ha nazionalità diversa da Nazione2.

Algebra relazionale-20220316 0804-1

Si vuole il valore massimo che compare nella colonna copie della tabella Vendita: Esercizio particolare che si riesce a fare con gli operatori dell'algebra relazionale è quello del calcolo del massimo o del minimo e sono due problemi completamente identici. La base di dati di riferimento è quella di negozi che vendono dischi. Quello che vogliamo fare è trovare il valore massimo che compare nella tabella vendita. Si tratta di trovare una caratteristica del valore massimo che permetta poi di individuarlo tra tutti gli altri valori, allora la caratteristica del valore massimo è che è un valore che non è minore di nessun altro. L'idea è proprio quella di andare a trovare i valori che invece non hanno questa caratteristica, cioè i valori che sono minori di qualcun altro valore e tra questi valori sicuramente non ci sarà il massimo, e poi per differenza si può trovare il valore massimo.

$$\pi_{\text{copie}}(\text{Vendita}) - \pi_{\text{copie}}(\text{Vendita} \bowtie_{\text{copie} < \text{Xcopie}} (\rho_{\text{Xp}_1 \text{iva}, \text{Xdisco id}, \text{Xcopie} \leftarrow \text{p}_1 \text{iva}, \text{disco id}, \text{copie}}(\text{Vendita})))$$

Da tutti i valori presenti nella colonna si tolgono quelli che sono minori di qualcun altro.

$$\text{XVendita} = \rho_{\text{Xp}_1 \text{iva}, \text{Xdisco id}, \text{Xcopie} \leftarrow \text{p}_1 \text{iva}, \text{disco id}, \text{copie}}(\text{Vendita}) \pi_{\text{copie}}(\text{Vendita}) - \pi_{\text{copie}}(\text{Vendita} \bowtie_{\text{copie} < \text{Xcopie}} \text{XVendita})$$

Vendita		
p_iva	disco_id	copie
10	1	3
10	2	4
10	3	4
20	2	5
30	1	2
30	3	6

XVendita		
Xp_iva	Xdisco_id	Xcopie
10	1	3
10	2	4
10	3	4
20	2	5
30	1	2
30	3	6

Vorrei discutere la query su un'istanza particolare della relazione vendita. Immaginiamo di essere in questa situazione con questi valori che compaiono in corrispondenza dell'attributo copie, quindi quello che vogliamo fare in questo caso particolare è una query che ci permetta di avere il valore sei come risultato. Come si può fare? Tutti gli altri valori, ad esclusione del sei, hanno la caratteristica di essere minori di qualcun altro. Come faccio a individuare questi valori che sono diversi dal sei? Posso fare una copia della tabella vendita, chiamata Xvendita (quindi si tratta di creare una vista che è rappresenti esattamente la tabella vendita, quindi che riporti esattamente tutte le caratteristiche sia strutturali che il contenuto della tabella vendita). In questo caso ho rinominato gli attributi della Xvendita in modo da distinguere gli attributi di una tabella dall'altra. A questo punto quello che si deve fare è mettere in join queste due relazioni in modo da determinare i valori che compaiono in copie che sono minori di un altro, cioè possiamo fare un teta join tra la tabella vendita e la tabella Xvendita in cui la condizione è che il valore che compare in corrispondenza dell'attributo copie sia minore dell'attributo che compare in corrispondenza dell'attributo xcopie. Facendo questo teta join ottengo questa tabella in cui nella corrispondenza della colonna copie ho valori che sono sicuramente minori di quelli che compaiono in xcopie, cioè nella colonna copie il valore 6 non può comparire, perché sei non è mai minore di un altro. A questo punto devo fare una proiezione sull'attributo copie di questa tabella (ovviamente quando faccio la proiezione duplicati si eliminano) e quindi la proiezione su quell'attributo mi restituisce proprio valori 2,3,4,5, cioè tutti i valori ad esclusione del valore massimo. Per trovare il valore massimo sarà sufficiente fare una differenza, quindi

faccio una proiezione sull'attributo copie dalla tabella vendita che mi restituisce tutti i valori, compreso il sei, e da questi valori tolgo quelli che invece non hanno la caratteristica che abbiamo detto. In questo modo riesco ad individuare il massimo. Se volessimo calcolare il minimo dovremmo fare la stessa cosa, ovviamente cambia la condizione, perché il minimo ha la caratteristica di non essere maggiore di nessuno, quindi devo andare a trovare i valori che sono maggiori di un altro invece che minori. Attenzione quando ci sono esercizi che richiedono una differenza, perché la differenza deve essere fatta tra

Vendita $\bowtie_{\text{copie} < \text{Xcopie}}$ XVendita				
p_iva	disco_id	copie	Xp_iva	Xdisco_id
10	1	3	10	2
10	1	3	10	3
10	1	3	20	2
10	1	3	30	3
10	2	4	20	2
10	2	4	30	3
10	3	4	20	2
10	3	4	30	3
20	2	5	30	3
30	1	2	10	1
30	1	2	10	2
30	1	2	30	3
30	1	2	30	2
30	1	2	30	1

X1
copie
2
3
4
5

$$X1 = \pi_{\text{copie}}(\text{Vendita} \bowtie_{\text{copie} < \text{Xcopie}} \text{XVendita})$$

faccio una proiezione sull'attributo copie dalla tabella vendita che mi restituisce tutti i valori, compreso il sei, e da questi valori tolgo quelli che invece non hanno la caratteristica che abbiamo detto. In questo modo riesco ad individuare il massimo. Se volessimo calcolare il minimo dovremmo fare la stessa cosa, ovviamente cambia la condizione, perché il minimo ha la caratteristica di non essere maggiore di nessuno, quindi devo andare a trovare i valori che sono maggiori di un altro invece che minori. Attenzione quando ci sono esercizi che richiedono una differenza, perché la differenza deve essere fatta tra

relazioni che hanno lo stesso schema, quindi anche se ci sono da fare delle proiezioni vanno fatte in modo da rendere gli schemi identici.

L'elenco di tutti i negozi (che hanno venduto dischi) insieme ai dischi venduti: Si tratta di andare a prendere i negozi che compaiono nella tabella vendita quindi si fa una proiezione sulla partita iva dei negozi che non hanno venduto dischi e i negozi che non hanno venduto dischi sono quelli che non compaiono nella tabella della vendita quindi anche questo lo possiamo fare per differenza, quindi da tutti i negozi che abbiamo tolgo quelli che hanno venduto qualcosa.

Negozi \bowtie Vendita

L'elenco di tutti i dischi insieme alle informazioni che riguardano i loro autori.

Disco $\bowtie_{\text{disco id}=\text{disco}}$ Composto da $\bowtie_{\text{autore}=\text{autore id}}$ Autore

La partita iva dei negozi che hanno venduto qualche disco.

$\pi_{p \text{ iva}}(\text{Vendita})$

La partita iva dei negozi che non hanno venduto dischi.

$\pi_{p \text{ iva}}(\text{Negozio}) - \pi_{p \text{ iva}}(\text{Vendita})$

La partita iva e la città dei negozi che non hanno venduto dischi:

$\pi_{p \text{ iva,citta}}(\text{Negozio} \bowtie (\pi_{p \text{ iva}}(\text{Negozio}) - \pi_{p \text{ iva}}(\text{Vendita})))$

La partita iva dei negozi che non hanno venduto tutti i dischi dell'elenco: Abbiamo una tabella dischi che contiene un certo numero di dischi e una tabella vendita in cui compaiono il riferimento da cui possiamo vedere quali sono i dischi che un certo negozio ha venduto. Vogliamo sapere quali sono quei negozi che non hanno venduto tutti i dischi. Ragioniamo su un esempio:

$\pi_{p \text{ iva}} \text{Negozi} \bowtie \pi_{\text{disco id}} \text{Disco}$	
p_iva	disco_id
10	1
10	2
10	3
20	1
20	2
20	3
30	1
30	2
30	3

$\pi_{p \text{ iva}, \text{disco id}} \text{Vendita}$	
p_iva	disco_id
10	1
10	2
10	3
20	2
30	1
30	3

immaginiamo di essere in queste situazioni, invece di lavorare sull'intera tabella negozio e sull'intera tabella disco ho fatto delle proiezioni perché tanto mi interessano la partita iva e il codice del disco. In questa situazione abbiamo tre negozi e tre dischi nella tabella vendita se vado a fare la proiezione su partita iva e disco io vedo per ogni negozio qual è il disco che è stato venduto, quindi in particolare il negozio 10 ha venduto tutti e tre i dischi, il negozio 20 invece ne ha venduto solo uno, il negozio 30 ne ha venduto solo due, quindi voglio come risposta alla mia interrogazione 20 e 30, perché sono i negozi che non hanno venduto tutti i dischi. Faccio il prodotto cartesiano tra la partita iva del negozio e il codice disco del disco, quindi mi creo tutte le possibili coppie di vendita, quindi come faccio a individuare i negozi che non li hanno venduti tutti? Ho bisogno di fare una differenza, quindi vado a togliere dal prodotto cartesiano le tuple che compaiono in vendita. Questa differenza mi dice che il negozio 20 non ha venduto il disco 1, che il negozio 20 non ha venduto il disco 3, che il negozio 30 non ha venduto il disco 2. Quindi è chiaro che la proiezione di questa differenza sull'attributo partita iva mi risponde proprio all'interrogazione, quindi negozi che non hanno venduto tutti i dischi sono proprio quelli che trovo facendo la differenza e facendo poi la proiezione sulla partita iva.

Negozi e Disco non hanno attributi in comune quindi il JOIN corrisponde ad un prodotto cartesiano.

La partita iva dei negozi che hanno venduto tutti i dischi dell'elenco.

$\pi_{p \text{ iva}}(\text{Negozio}) - \pi_{p \text{ iva}}(\pi_{p \text{ iva}}(\text{Negozio}) \bowtie \pi_{\text{disco id}}(\text{Disco}) - \pi_{p \text{ iva}, \text{disco id}}(\text{Vendita}))$

Negozi e Disco non hanno attributi in comune quindi il JOIN corrisponde ad un prodotto cartesiano.

La partita iva dei negozi che hanno venduto più di un tipo di disco, ovvero, la partita iva dei negozi che hanno venduto almeno due dischi diversi, ovvero, la partita iva dei negozi che compaiono nella tabella Vendita almeno due volte; l'attributo copie non interessa:

$\pi_{p \text{ iva}}(\pi_{p \text{ iva}}(\text{Negozio}) \bowtie \pi_{\text{disco id}}(\text{Disco}) - \pi_{p \text{ iva}, \text{disco id}}(\text{Vendita}))$

Vendi	
p_iva	disco_id
10	1
10	2
10	3
20	2
30	1
30	3

XVendi	
p_iva	Xdisco_id
10	1
10	2
10	3
20	2
30	1
30	3

Coppie		
p_iva	disco_id	Xdisco_id
10	1	1
10	1	2
10	1	3
10	2	1
10	2	2
10	2	3
10	3	1
10	3	2
10	3	3
20	2	2
30	1	1
30	1	3
30	3	1
30	3	3

$\sigma_{disco_id \neq Xdisco_id}$ Coppie		
p_iva	disco_id	Xdisco_id
10	1	2
10	1	3
10	2	1
10	2	3
10	3	1
10	3	2
30	1	3
30	3	1
30	3	3

Vendi = $\pi_{p_iva, disco_id}$ (Vendita)

XVendi = $\rho_{Xdisco_id \leftarrow disco_id}$ (Vendi)

Coppie = Vendita \bowtie XVendi

Noi vogliamo sapere quali sono i negozi che hanno venduto almeno due dischi, quindi vogliamo che il negozio 10 compaia nel nostro risultato finale, anche il negozio 30, non vogliamo il negozio 20. Dobbiamo riuscire a individuare una query che ci permetta di individuare 20 per poi eliminarlo da tutti quelli che invece sono presenti. Conviene creare una tabella chiamata xvendi, clone della tabella vendi, utilizzando però una ridenominazione dell'attributo disco in xdisco, perché poi voglio fare un join naturale (lascio gli altri attributi con lo stesso nome per semplificare il problema). L'operatore di ridenominazione è un operatore fondamentale perché ci permette di fare dei join di una tabella con se stessa e di risolvere problemi che insomma altrimenti non saremmo in grado di risolvere. In questo caso il join naturale viene fatto sulla base dell'attributo che le due tabelle hanno in comune in corrispondenza di partita iva e quindi nel risultato andremo a combinare tutti questi record sulla base della partita iva. 20-2 è l'unico record che come risultato finale del join mi darà luogo soltanto ha un record

mentre in corrispondenza del negozio 10 si creano 9 record, perché ogni negozio lo congino con tutti gli altri, e nel caso del negozio 30 mi comparirà quattro volte, perché ogni record lo congino con gli altri due. Questo per evidenziare il fatto che quando faccio questo join naturale il negozio 20 lo individuo facilmente, perché è l'unico che ha la caratteristica di comparire in corrispondenza di una coppia con valori uguali. Dopodiché seleziono da questa tabella il record in cui disco_id è diverso da xdisco_id perché mi permettono di trovare quei negozi che hanno venduto almeno due dischi. A questo punto ho bisogno di fare una proiezione sulla partita iva, questa proiezione ovviamente mi restituisce soltanto i negozi 10 e 30.

La tabella Coppie contiene coppie di vendite riferite allo stesso negozio, ma contiene anche la coppia che si riferisce allo stesso disco.

$\pi_{p_iva}(\sigma_{disco_id \neq Xdisco_id} (Coppie))$

La partita iva dei negozi che hanno venduto un solo tipo di disco, ovvero, la partita iva dei negozi che compaiono nella tabella Vendita una sola volta. Sono tutti i negozi (che hanno venduto qualcosa) meno quelli che hanno venduto almeno due dischi diversi.

$\pi_{p_iva}(Vendita) - \pi_{p_iva}(\sigma_{disco_id \neq Xdisco_id} (Coppie))$

$\pi_{p_iva}(Vendita) - \pi_{p_iva}(\sigma_{disco_id \neq Xdisco_id} (\text{Vendita} \bowtie \rho_{Xdisco_id, Xcopie \leftarrow disco_id, copie} (Vendita)))$

Le città dei negozi che hanno venduto un solo tipo di disco

Solodisco = $\pi_{p_iva}(Vendita) - \pi_{p_iva}(\sigma_{disco_id \neq Xdisco_id} (\text{Vendita} \bowtie \rho_{Xdisco_id, Xcopie \leftarrow disco_id, copie} (Vendita))) \pi_{citta}(\text{Negozio} \bowtie Solodisco)$

Il codice dei dischi venduti nella città Citta1 (ed eventualmente in altre citt'a).

$\pi_{disco_id}(\text{Vendita} \bowtie \sigma_{citta=Citta1}(\text{Negozio}))$

Il codice dei dischi venduti solo nella città Citta1

$\pi_{disco_id}(\text{Vendita}) - \pi_{disco_id}(\text{Vendita} \bowtie \sigma_{citta \neq Citta1}(\text{Negozio}))$

Il codice dei dischi non venduti nella città Citta1.

$\pi_{disco_id}(\text{Vendita}) - \pi_{disco_id}(\text{Vendita} \bowtie \sigma_{citta=Citta1}(\text{Negozio}))$

Il codice dei dischi venduti in più di una citta. Si costruiscono tutte le possibili coppie di vendite dello stesso disco in citt'a diverse.

VenditaNegozio = $\pi_{disco_id, citta}(\text{Vendita} \bowtie \text{Negozio})$

XVenditaNegozio = $\rho_{Xcitta \leftarrow citta}(\text{Vendita} \bowtie \text{Negozio})$

$\pi_{disco_id}(\sigma_{citta \neq Xcitta}(\text{Vendita} \bowtie \text{XVendita} \bowtie \text{Negozio}))$

Il codice dei dischi venduti in una sola città. Sono tutti i dischi venduti meno quelli venduti in più di una città.

$\text{VenditaNegozi} = \pi_{\text{disco id}, \text{citta}}(\text{Vendita} \bowtie \text{Negozi})$

$\text{XVenditaNegozi} = \rho_{\text{citta} \leftarrow \text{citta}}(\text{VenditaNegozi})$

$\pi_{\text{disco id}}(\text{Vendita}) - \pi_{\text{disco id}}(\sigma_{\text{citta} \neq \text{Xcitta}}(\text{VenditaNegozi} \bowtie \text{XVenditaNegozi}))$

Persone		
Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Paternità	
Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Maternità	
Madre	Figlio
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Altra base di dati: abbiamo tre relazioni, persone, maternità e paternità. La tabella persone che contiene informazioni sul nome, l'età e il reddito. La tabella paternità che crea un'associazione padre-figlio e maternità un'associazione madre-figlio. Immaginiamo che il nome della persona sia l'elemento distintivo per ogni persona, quindi nome rappresenta la chiave primaria della tabella persone. Gli altri attributi padre, figlio e madre fanno riferimento a il nome di una persona, quindi ci dovrebbero essere dei vincoli di integrità tra questi attributi. La coppia padre-figlio e madre-figlio formano la chiave primaria nelle altre due tabelle.

Trovare i nomi delle persone tra i 20 e i 40 anni: il riferimento all'età è nella tabella delle persone quindi si tratta di fare una selezione andando a prendere dalla tabella delle persone quelle che hanno l'età maggiore di 20 e minore di 40 e poi si fa la proiezione sul nome.

$\pi_{\text{Nome}}(\sigma_{\text{Eta} \geq 20 \text{ and } \text{Eta} \leq 40}(\text{Persone}))$

Trovare i nomi dei padri: si tratta di andare a fare una proiezione sull'attributo padre della tabella.

$\pi_{\text{Padre}}(\text{Paternita})$

Trovare i nomi delle persone con entrambi i genitori: ho bisogno di fare un join naturale fra le tabelle paternità e maternità che hanno in comune l'attributo figlio, ottenendo tutte le terne in cui del figlio si conosce sia il padre che la madre.

$\text{Paternita} \bowtie \text{Maternita}$

Trovare il nome e l'età dei figli di Anna: Ho bisogno di informazioni che sono sia nella tabella delle persone, perché c'è un riferimento all'età, sia nella tabella maternità, in cui ho il legame tra figli e madre. Un teta join tra persone e maternità con la condizione che il nome della persona corrisponda al figlio nella tabella maternità, poi la selezione sull'attributo madre uguale ad anna e poi proietto su nome ed età.

$\pi_{\text{Nome}, \text{Eta}}(\sigma_{\text{madre}=\text{Anna}}(\text{Persone} \bowtie \sigma_{\text{Nome}=\text{Figlio}} \text{Maternita}))$

Trovare i nomi delle persone che guadagnano più della propria madre, mostrando anche il reddito della madre: Abbiamo bisogno della tabella maternità, perché c'è un riferimento alla madre e della tabella persone, che ci servirà due volte, una volta perché abbiamo bisogno del nome delle persone che guadagnano più della propria madre, una seconda volta perché vogliamo anche il reddito della madre. Siccome la tabella persona è coinvolta due volte avremo bisogno di fare una ridenominazione.

$\pi_{\text{Nome}, \text{Reddito}, \text{RM}}(\sigma_{\text{Reddito} > \text{RM}}(\rho_{\text{NM}, \text{EM}, \text{RM} \leftarrow \text{Nome}, \text{Eta}, \text{Reddito}}(\text{Persone}) \bowtie_{\text{NM}=\text{Madre}} (\text{Maternita} \bowtie_{\text{Figlio}=\text{Nome}} \text{Persone})))$

Trovare i nomi delle persone con almeno due figli: Ho risolto la query con una serie di viste, ho creato una vista per paternità1 rinominando l'attributo figlio in figlio1 e ho fatto la stessa cosa per quanto riguarda la maternità rinominando l'attributo figlio in figlio1. Ho fatto un join naturale perché paternità e paternità1 hanno in comune l'attributo padre, e la coppia maternità e maternità1 hanno in comune l'attributo madre, quindi prendo quelli che hanno la caratteristica di avere figlio minore di figlio1, perché in questo modo io riesco a ritrovare quei record in cui compare più di un figlio, perché dove ce n'è solo uno in questa congiunzione troverò soltanto un record in cui un figlio è congiunto con se stesso. Faccio questa operazione sia per i padri che per le madri e poi utilizzo un'unione. Ho rinominato sia padre che madre in genitore in modo da ricondurmi a due schemi identici che quindi posso unire.

$\text{Paternita1} = \rho_{\text{Figlio1} \leftarrow \text{Figlio}}(\text{Paternita})$

$\text{Maternita1} = \rho_{\text{Figlio1} \leftarrow \text{Figlio}}(\text{Maternita})$

Padrecon2 = $\sigma_{\text{Figlio} < \text{Figlio1}}(\text{Paternita} \bowtie \text{Paternita1})$

Madrecon2 = $\sigma_{\text{Figlio} < \text{Figlio1}}(\text{Maternita} \bowtie \text{Maternita1})$

$p_{\text{Genitore}} \leftarrow \text{Padre} (\pi_{\text{Padre}}(\text{Padrecon2})) \cup p_{\text{Genitore}} \leftarrow \text{Madre} (\pi_{\text{Madre}}(\text{Madrecon2}))$

Trovare i nomi delle persone il cui padre guadagna più della madre: Utilizzato delle viste paternitàR e maternitàR, che sono del tutto simili. Per quanto riguarda la prima ho creato la congiunzione fra paternità e persone tramite la condizione padre uguale al nome e poi sono andata a proiettare soltanto sugli attributi padre, figlio e reddito, infine per evidenziare che questo reddito è quello del padre ho ridenominato l'attributo reddito in RP. Nella tabella maternità ho fatto una cosa analoga. Vado a fare il join naturale tra queste due viste rispetto all'attributo figlio che le due relazioni hanno in comune. Devo selezionare quelle tuple in cui il reddito del padre è maggiore del reddito della madre e poi faccio la proiezione sul figlio perché è quello che l'esercizio mi richiedeva.

PaternitaR = $p_{\text{RP}} \leftarrow \text{reddito} (\pi_{\text{padre}}, \text{figlio}, \text{reddito} ((\text{Paternita} \bowtie \text{padre} = \text{nome Persone})))$

MaternitaR = $p_{\text{RM}} \leftarrow \text{reddito} (\pi_{\text{madre}}, \text{figlio}, \text{reddito} ((\text{Maternita} \bowtie \text{madre} = \text{nome Persone})))$

$\pi_{\text{figlio}} (\sigma_{\text{RP} > \text{RM}}(\text{PaternitaR} \bowtie \text{MaternitaR}))$

Trovare i nomi delle donne che hanno avuto figli con padri diversi: Fare un join naturale tra paternità e maternità rispetto all'attributo figlio e poi vado a proiettare rispetto a padre e madre, quindi mi vado a creare tutte le coppie padri-madri. Questa vista la metto in join con se stessa, perché quello che voglio far vedere è se ci sono donne che hanno avuto figli con padri diversi, ovvero se una madre ha avuto figli con padri diversi nella tabella la stessa madre comparirà due volte in corrispondenza di padri diversi. Come faccio a individuarle? Si può fare un join tra pm e se stessa dopo però aver ridenominato padre in padre1 in una delle due tabelle pm. Si va a ricercare le tuple in cui padre è diverso da padre1 quelle che hanno questa caratteristica corrispondono proprio alle madri che hanno avuto figli con padri diversi.

PM = $\pi_{\text{padre}, \text{madre}} ((\text{Paternita} \bowtie \text{Maternita}))$

$\pi_{\text{madre}} (\sigma_{\text{padre} < > \text{padre1}}(p_{\text{M}} \bowtie p_{\text{padre1}} \leftarrow \text{padre} (\text{PM})))$

Trovare i nomi delle donne che hanno avuto un figlio prima dei 30 anni: Di ogni donna conosco quali sono i figli e di ogni persona conosco l'età. La tabella persone mi servirà due volte, perché una volta ho bisogno dell'indicazione dell'età della madre e una volta ho bisogno delle indicazioni dell'età del figlio. Per far questo ho creato due viste che ho chiamato personem per madre e personef per il figlio, in cui ho ridenominato gli attributi nome ed età di persone in madre ed etàm e poi proietto esattamente su questi due attributi. La stessa cosa ho fatto per i figli. Ho bisogno della tabella maternità in cui ho la relazione tra madre e figlio e questa tabella la congiungo con le due tabelle personem e personef, avendo fatto queste ridenominazioni posso utilizzare dei join naturali. Come faccio a trovare i nomi delle donne che hanno avuto un figlio prima dei trent'anni? Avendo a disposizione sia l'età della madre che l'età del figlio devo andare a verificare che l'età della madre meno l'età del figlio sia minore di 30, infine faccio la proiezione finale sull'attributo madre.

Personem = $\pi_{\text{madre}, \text{etaM}}(p_{\text{madre}}, \text{etaM} \leftarrow \text{nome}, \text{eta}(\text{Persone}))$

Personef = $\pi_{\text{figlio}, \text{etaF}}(p_{\text{figlio}}, \text{etaF} \leftarrow \text{nome}, \text{eta}(\text{Persone}))$

$\pi_{\text{madre}} ((\sigma(\text{etaM} - \text{etaF} < 30)((\text{Personem} \bowtie \text{Maternita}) \bowtie \text{Personef}))$

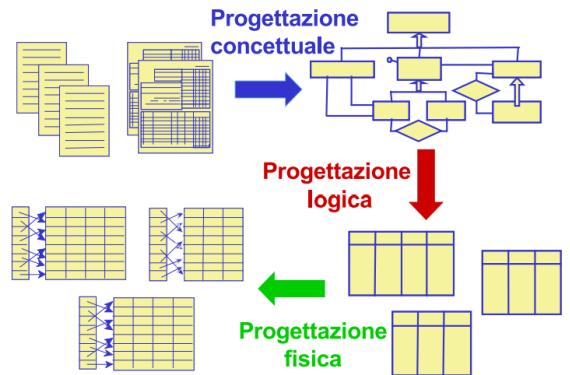
Trovare le coppie nipote/nonno-materno: In maternità voglio che il figlio giochi il ruolo di nipote e quindi ho fatto una ridenominazione figlio in nipote. In paternità invece voglio che il padre giochi il ruolo del nonno e quindi faccio una ridenominazione padre in nonno. A questo punto quello che voglio è che il figlio di paternità sia la madre del nipote e quindi faccio la congiunzione rispetto alla condizione figlio uguale madre e poi proietto rispetto agli attributi che mi interessano, che sono quello di nonno e nipote.

$\pi_{\text{nonno}, \text{nipote}} (p_{\text{nonno} \leftarrow \text{padre}}(\text{Paternita}) \bowtie_{\text{figlio} = \text{madre}} p_{\text{nipote} \leftarrow \text{figlio}}(\text{Maternita}))$

Modello Entità-Relazione-20220322 0803-1

Come si può arrivare a definire uno schema logico che sia coerente e che descriva la realtà di interesse? Iniziamo a parlare del modello entità relazione. Perché dobbiamo definire un modello diverso da quello logico per definire una base di dati? Perché il modello relazionale è abbastanza rigido, il fatto che tutte le informazioni siano rappresentate tramite delle tabelle e che i legami tra queste tabelle siano basate su valori fa sì che non sia facile iniziare a progettare una base di dati partendo direttamente da questo tipo di modello perché è difficile insomma avere subito le idee chiare. Il processo di progettazione di una base di dati è qualcosa di più ampio e che rientra in un contesto più generale, che è quello del ciclo di vita dei sistemi

informativi, ovvero un insieme di azioni che devono essere svolte da figure diverse, analisti, progettisti, utenti per arrivare a sviluppare e utilizzare un sistema informativo. Spesso sono necessarie delle modifiche, quindi si parla di ciclo perché spesso si deve fare un passo indietro e approfondire certi aspetti. Quali sono le fasi che sono previste durante la progettazione? Innanzitutto uno studio di fattibilità quindi definire i costi e quali sono le priorità (quali sono gli aspetti che ci interessa realizzare per primi), fondamentale è la raccolta e l'analisi dei requisiti, per capire cosa devo realizzare e in questa fase ci si deve confrontare con le richieste degli utenti che non sempre sono che chiare. Una volta capito e raccolte le informazioni sulla base dobbiamo poi realizzare il progetto, si entra nella fase di progettazione vera e propria, quindi si tratta di progettare sia i dati che le funzioni, quindi le procedure che su questi dati dovranno lavorare, poi dalla progettazione si passa alla realizzazione, seguono poi le fasi di validazione e collaudo, fino ad arrivare al funzionamento, ovvero il sistema diventa operativo. A partire dai requisiti della base di dati si deve procedere a fare una progettazione concettuale della base di dati, cioè dobbiamo cercare di capire che cosa è necessario fare senza però fare riferimento a nessuno schema logico particolare e ragionare in maniera più astratta. La progettazione concettuale restituisce quello che si chiama lo schema concettuale. Lo schema concettuale lo faremo utilizzando il modello entità relazione, a partire da questo modello entità relazione prodotto nella fase di progettazione concettuale, si può attraverso un'ulteriore fase di progettazione logica, arrivare allo schema logico, nel nostro caso al modello relazionale, e poi a quel punto la realizzazione vera e propria è abbastanza semplice. Il modello concettuale permette di rappresentare i dati in modo indipendente da qualsiasi modello logico che andremo a utilizzare e vengono utilizzati per descrivere il progetto nella fase iniziale.



Modelli di dati

- Modelli logici: utilizzati nei DBMS esistenti per l'organizzazione dei dati utilizzati dai programmi indipendenti dalle strutture fisiche. Esempi: relazionale, reticolare, gerarchico, a oggetti.
- Modelli concettuali: permettono di rappresentare i dati in modo indipendente da ogni sistema cercano di descrivere i concetti del mondo reale sono utilizzati nelle fasi preliminari di progettazione. Servono per ragionare sulla realtà di interesse, indipendentemente dagli aspetti realizzativi. Permettono di rappresentare le classi di oggetti di interesse e le loro correlazioni. Prevedono efficaci rappresentazioni grafiche (utili anche per documentazione e comunicazione). Il modello Entity-Relationship è il più diffuso modello concettuale. Introdotto da Chen nel 1976, poi esteso. Ne esistono molte versioni, (più o meno) diverse l'una dall'altra. I costrutti del modello ER: Entità, Relationship, Attributo, Identificatore, Generalizzazione.

Entità

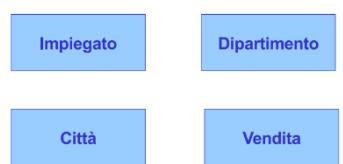
Classe di oggetti (fatti, persone, cose) della realtà di interesse con proprietà comuni e con esistenza autonoma. Esempi: impiegato, città, conto corrente, ordine, fattura.

un'entità è semplicemente una classe di oggetti che hanno proprietà comuni e che esistono autonomamente quindi indipendentemente dalla presenza di altri oggetti.

Entità: schema e istanza

- Entità: classe di oggetti, persone, ..., omogenei
- Occorrenza (o istanza) di entità: elemento singolo della classe (l'oggetto, la persona, ...)
- Nello schema concettuale rappresentiamo le entità, non le singole istanze (astrazione)
- Ogni entità ha un nome che la identifica univocamente nello schema:
 - nomi espressivi
 - opportune convenzioni: usare il singolare

Dal punto di vista della rappresentazione grafica, un'entità viene rappresentata tramite un rettangolo con un nome.

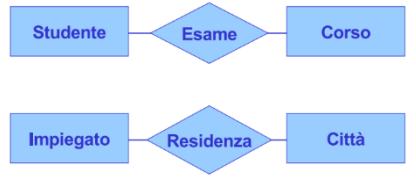


Relationship

Legame logico fra due o più entità, rilevante nell'applicazione di interesse. Esempi: Residenza (fra persona e città) che associa ogni persona alla città in cui questa risiede, Esame (fra studente e corso). Chiamata anche: relazione, correlazione, associazione. Ogni relationship ha un nome che la identifica univocamente nello schema:

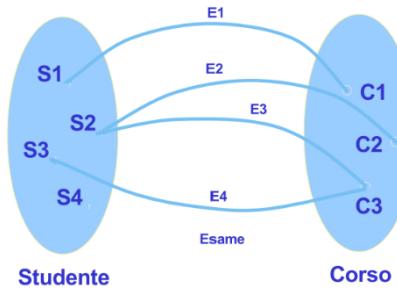
- nomi espressivi
- opportune convenzioni: singolare, sostantivi invece che verbi (se possibile)

Da un punto di vista grafico la relationship si rappresenta come un rombo che collega due o più entità, quindi ad esempio le entità studenti e corso che sono collegate dall'associazione esame e l'associazione residenza che collega impiegati e città.

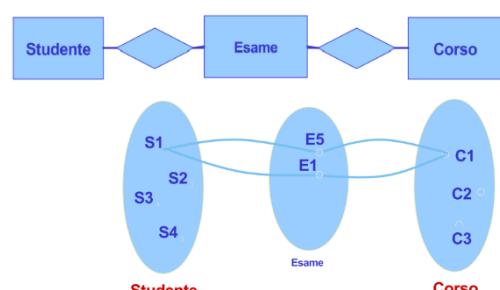
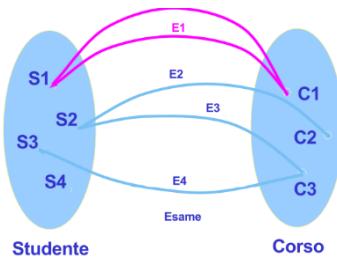


Relationship, occorrenze

Una occorrenza di una relationship binaria è una coppia di occorrenze di entità, una per ciascuna entità coinvolta. Una occorrenza di una relationship n-aria è una n-upla di occorrenze di entità, una per ciascuna entità coinvolta. Nell'ambito di una relationship non ci possono essere occorrenze (coppie, ennuple) ripetute: si tratta infatti di un sottoinsieme del prodotto cartesiano.

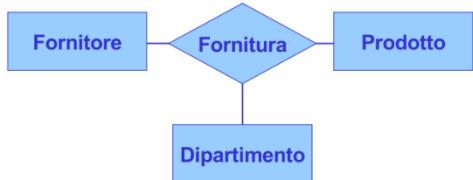


Esempio: abbiamo l'entità studente e l'entità corso, l'entità rappresenta una classe di oggetti e quindi potete immaginare un'entità come un insieme di studenti e l'entità corso come un insieme di corsi. Nell'insieme studenti sono presenti quattro studenti e nella classe corso sono presenti tre corsi. Che cos'è un'un'associazione esame tra studenti e corso? È un legame logico che mi dice che se uno studente ha sostenuto un certo corso ma mi dice anche se un corso è stato sostenuto da certi studenti. In questo esempio succede che lo studente S1 ha sostenuto il corso C1, ma possiamo anche dire che il corso C1 è stato sostenuto soltanto dallo studente. Lo studente S2 ha fatto due corsi, che sono il corso C2 e il corso C3. Lo studente S3 ha fatto il corso C3. Lo studente S4 non ha sostenuto nessun corso. Dal punto di vista dei corsi il corso C3 è stata sostenuta da due studenti e i corsi C1 e C2 sono stati sostenuti soltanto da uno, quindi l'associazione è la possibilità di rappresentare come un arco che collega un oggetto della classe studente con un oggetto della classe corso. A seconda delle situazioni da ogni oggetto possono uscire più archi e quello che si deve fare quando si va a definire uno schema in modo preciso è capire in generale quanti archi potranno uscire da ogni oggetto che appartiene alle varie entità. Dobbiamo prevederlo in modo da realizzare un modello più possibile preciso. Quindi che cos'è un'occorrenza di associazione? Si tratta di un'associazione binaria, quindi che coinvolge soltanto due entità, un'occorrenza è semplicemente una coppia di valori che rappresenta il fatto che tra quei due oggetti, uno di una classe e uno dell'altra, esiste un'associazione. Nell'ambito delle associazioni non possono esserci occorrenze ripetute, quindi la coppia S1 e C1 si rappresenta una volta e basta, non possono esserci linee diverse che collegano gli stessi oggetti. L'associazione tra entità la possiamo vedere come un sottoinsieme del prodotto cartesiano tra l'entità che sono coinvolte, quindi da una parte avete due insiemi, le coppie che potete formare sono quelle date dal prodotto cartesiano, collego tutti gli studenti con tutti gli esami e tutti gli esami con tutti gli studenti. Torniamo sempre all'associazione esami, abbiamo detto ogni coppia studente-corso io la posso rappresentare tramite quel modello soltanto una volta, se però volessi rappresentare non soltanto gli esami che sono stati superati ma tutte le prove che uno studente fa per superare l'esame, potrei avere la necessità di considerare la stessa coppia studente-corso più volte. È fondamentale perché quando voi andate a rappresentare nel modello un'associazione tra oggetti dovete essere consapevoli che se avete bisogno di rappresentare queste coppie più di una volta dovete utilizzare uno schema diverso. In questo caso quello che si fa è promuovere l'associazione in entità, cioè si introduce una nuova entità esame che collegiamo alle due entità studente e corso. L'introduzione di una di esame come entità mi permette di creare collegamenti tra S1-C1 diversi perché passo tramite un esame diverso, quindi se lo studente S1 ha provato due volte a fare l'esame C1 ci saranno due oggetti nella classe esame che fanno riferimento a quello studente in date diverse e che mi permettono di collegarmi al corso C1 in due modi diversi.





vedete che le due entità possono essere associate anche tramite legami logici diversi. Esempio: L'associazione fornitura collega il fornitore, il prodotto e il dipartimento. anche in questo caso il passaggio alla rappresentazione tramite insiemi può

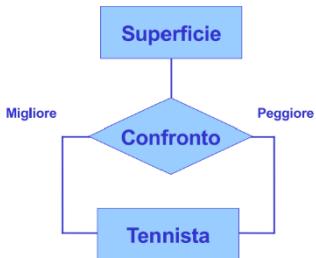


prodotto P2 al dipartimento D2 e così via.

Esistono associazioni ricorsive, ovvero una stessa entità può essere collegata a se stessa tramite un'associazione. Esempio: abbiamo l'entità impiegato e l'associazione collega, quindi per sapere qual è il collega di un impiegato devo associare in pratica un impiegato a un altro impiegato. Questa è un'associazione simmetrica perché voglio sapere che un certo impiegato ha come collega un altro impiegato, ma non c'è una precedenza tra i due impiegati perché ognuno è collega dell'altro.

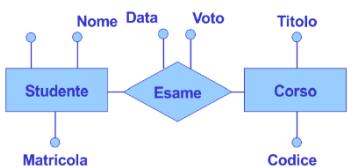
Possono esserci invece delle associazioni non simmetriche, in cui l'ordine con cui compaiono

Possono esserci invece delle associazioni non simmetriche, in cui l'ordine con cui compare gli elementi nella coppia mi danno un'indicazione, ad esempio l'entità ministro e l'associazione successione, quello che voglio rappresentare con questo schema è che il primo elemento rappresenti il successore e il secondo rappresenti il predecessore, non è simmetrico scambiare i due oggetti. Esempio in cui ho sempre un'associazione è ricorsiva e non simmetrica: voglio



del tennista 2 sul campo da gioco S1.

Attributo



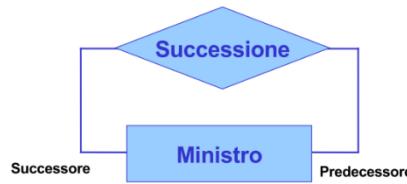
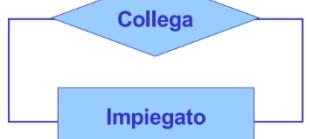
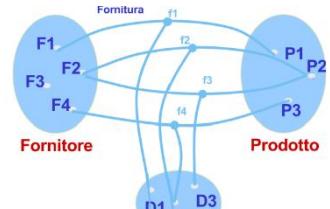
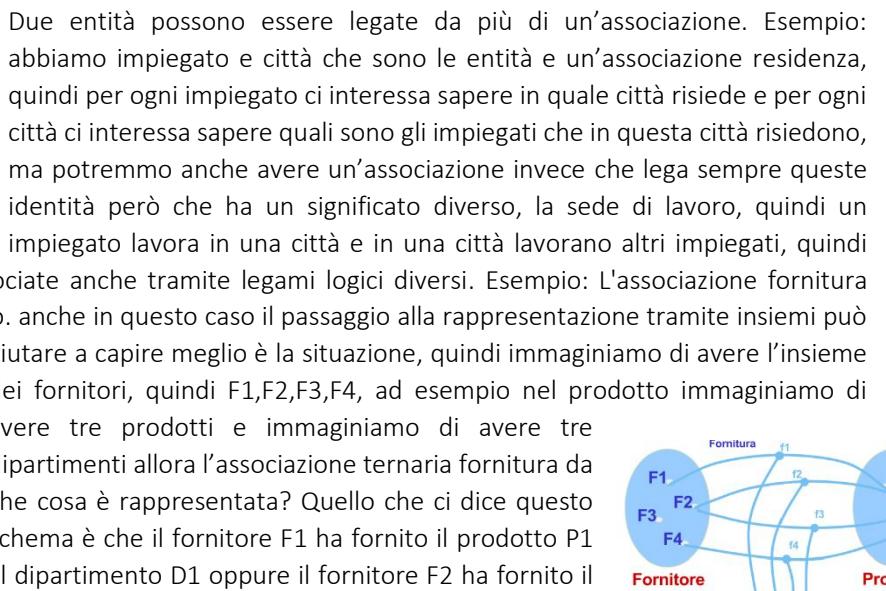
significato o uso. Esempio attributo composto: Via, Numero civico e CAP formano un Indirizzo. Si usa un ovale che rappresentare il nome dell'attributo composto e poi si fanno uscire i singoli attributi come uscite da un contenitore.

Attributi da esso invece che insomma farli uscire direttamente dall'impiegato. Gli attributi sono tutti di tipo pubblico (titolo, cognome, indirizzo, ecc.) e non si può accedere

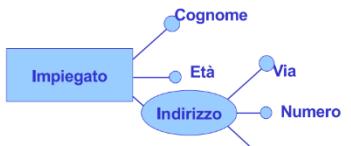
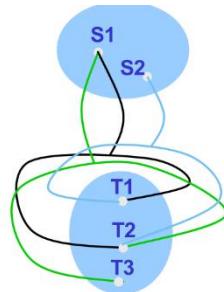
Gli attributi si rappresentano come delle linee che escono dalle entità o delle associazioni a cui si associa un nome.

Cardinalità di relationship

Coppia di valori associati a ogni entità che partecipa a una relationship. Specificano il numero minimo e massimo di occorrenze delle relationship cui ciascuna occorrenza di una entità può partecipare. Per semplicità usiamo solo tre simboli: 0 e 1 per la cardinalità minima: 0 = partecipazione opzionale, 1 = partecipazione obbligatoria, 1 e N per la massima: N non pone alcun limite. In pratica quando si definisce un'associazione tra entità è necessario definire una coppia di valori che si associano ad



confrontare tennisti sulla base della superficie su cui giocano, quindi superficie rappresenta il tipo di superficie del campo da tennis e l'associazione confronto mi permette di confrontare coppie di tennisti e quindi da ramo sinistro c'è l'indicazione migliore e sul ramo destro l'indicazione peggiore, per cui nella coppia il primo che vado a considerare è quello che ha giocato meglio dell'altro. Io posso rappresentare con degli insiemi, in questo modo abbiamo l'insieme dei tennisti, quindi devo associare ad ogni tennista un altro tennista e se hanno giocato insieme su un certo campo da tennis, quindi il tennista 1 è stato migliore del tennista 2 sul campo da gioco S2 oppure il tennista 3 stato migliore



Proprietà elementare di un'entità o di una relationship, di interesse ai fini dell'applicazione. Associa ad ogni occorrenza di entità o relationship un valore appartenente a un insieme detto dominio dell'attributo.

Attributi composti: raggruppano attributi di una medesima entità o relationship che presentano affinità nel loro

significato o uso. Esempio attributo composto: Via, Numero civico e CAP formano un Indirizzo. Si usa un ovale che rappresentare il nome dell'attributo composto e poi si fanno uscire i singoli attributi come uscite da un contenitore.

Attributi da esso invece che insomma farli uscire direttamente dall'impiegato. Gli attributi sono tutti di tipo pubblico (titolo, cognome, indirizzo, ecc.) e non si può accedere

Gli attributi si rappresentano come delle linee che escono dalle entità o delle associazioni a cui si associa un nome.

ogni entità che partecipa all'associazione. Questi valori rappresentano il minimo e il massimo di occorrenze cui ciascuna entità può partecipare nell'associazione.



Esempio: abbiamo l'associazione assegnamento che crea un legame logico tra impiegato e incarico, sui rami dell'associazione ho inserito delle coppie di valori che si riferiscono all'entità vicina, quindi 2-5 si riferisce a impiegato e 0-50 si riferisce a incarico. Utilizzeremo questo formalismo per cui la coppia si riferisce all'entità più vicina. Ogni impiegato deve essere assegnato al minimo almeno a due incarichi e al massimo a 5, cioè si rappresenta con insieme gli impiegati e gli incarichi, da ogni impiegato devono uscire almeno due linee e al massimo 5. L'incarico partecipa a questa associazione con cardinalità minima zero, cioè ci possono essere degli incarichi che non sono stati assegnati a nessuno, ma al massimo un incarico deve essere assegnato a 50. Esempio voglio rappresentare l'associazione residenza, ho un insieme di studenti in cui ogni studente deve per forza avere una residenza, quindi da ogni studente deve partire una e una sola linea, però dal punto di vista della città ci possono essere delle città in cui non risiede nessuno degli studenti presenti oppure possono esserci delle città in cui risiedono più studenti, quindi lo studente deve partecipare con cardinalità minima uno e massima uno, ogni studente deve per forza partecipare a questa associazione e deve parteciparci però una volta sola, cioè ogni studente deve essere collegato ad una e ad una sola città, la città invece, siccome possono esserci città che non corrispondono a nessuno studente, non è obbligata a partecipare, per cui la quantità minima in questo caso sarà zero e siccome in una città possono risiedere tanti studenti la cardinalità massima sarà n, ai indicare il fatto che non c'è un limite superiore al numero di occorrenze cui una città può partecipare.



Tipi di relationship

Con riferimento alle cardinalità massime, abbiamo relationship:

- Uno a uno
- Uno a molti
- Molti a molti

Attenzione al verso nelle relationship uno a molti.

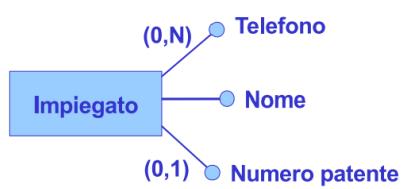


Il riferimento alla tipologia uno a molti, uno a uno e molti a molti si fa sulla base delle cardinalità massime. Esempi di associazioni molti a molti: la cardinalità massima nelle due coppie di cardinalità è sempre n, però la cardinalità minima può variare. Che differenza ci può essere nella scelta della cardinalità minima? Ho l'associazione esami di tipo molto a molti perché ogni studente può aver sostenuto tanti esami e un esame può essere stato sostenuto da tanti studenti, però ci potrebbe essere sia lo studente che non ha sostenuto nessun esame sia il corso che non è stato sostenuto da nessuno studente per cui è sensato mettere zero come cardinalità minima. Un altro esempio: abbiamo la montagna e l'alpinista e voglio creare un legame tra questi due entità tramite l'associazione scalata, è stato scelto di mettere zero come cardinalità minima di montagna perché ci potrebbero essere delle montagne che non sono state scalate da nessuno per l'alpinista e invece compare 1 nell'altra cardinalità minima perché per definirsi alpinista almeno una montagna deve averla scalata. L'altra la situazione che può capitare quando tutte e due l'entità sono obbligate a partecipare all'associazione: abbiamo un'associazione abilitazione che associa macchinisti con locomotore, quindi ogni macchinista dovrà avere l'abilitazione per un locomotore e un locomotore dovrà essere stato associato almeno a un macchinista. Associazioni di tipo uno a molti: perché la cardinalità massima di un'entità è uno e la cardinalità massima dell'altra è n. Un esempio: abbiamo comune e provincia e l'associazione ubicazione, chiaro che un comune deve essere in una provincia e una provincia deve avere almeno un comune, quindi tutte e due l'entità sono obbligate a partecipare, quindi da ogni comune parte sicuramente una e una sola linea e da una provincia ne parte una o più. L'ultima tipologia di associazioni sono quelli uno a uno, cioè tutte e due le entità che partecipano hanno come cardinalità massima 1.

Cardinalità di attributi

È possibile associare delle cardinalità anche agli attributi, con due scopi:

- indicare opzionalità (informazione incompleta)
- indicare attributi multivalue.



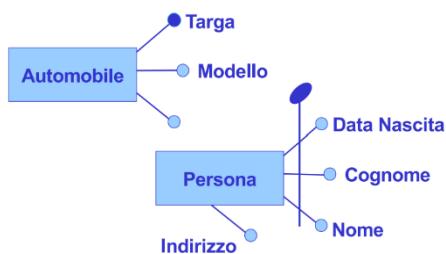
Si tratta sempre di indicare una coppia di valori, il valore minimo e il valore massimo. Questa caratteristica del modello ci permette di definire delle situazioni particolari, ad esempio: abbiamo l'entità impiegato e voglio tenere traccia del numero della patente, però non è detto che un impiegato ce l'abbia, per cui questo attributo è stato definito come 0-1, e nel caso del

telefono ho una coppia 0-n, che mi indica che ci potrebbe essere un impiegato che non ha telefono e che di un certo impiegato ho più numeri di telefono, quindi ho più informazioni della stessa tipologia che voglio associare a un certo impiegato.

Identificatore di una entità

Strumento per l'identificazione univoca delle occorrenze di un'entità costituito da:

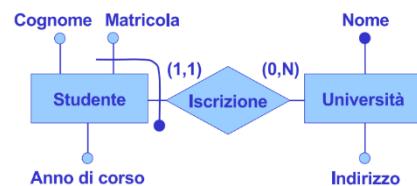
- attributi dell'entità: identificatore interno
- (attributi +) entità esterne attraverso relationship: identificatore esterno



Ogni entità deve essere identificata da uno o più attributi e gli identificatori interni in particolare sono gli attributi che dell'entità che hanno questa caratteristica, ad esempio, se ho un'entità automobile, la targa è un identificatore interno perché ogni targa mi permette di distinguere una automobile dall'altra. L'identificatore di persone è composto da una terna di valori.

Ci sono situazioni in cui gli identificatori interni non bastano per identificare un oggetto della classe, ad esempio: ho due entità che sono lo studente con un nome, matricola e anno di corso e poi università con il nome e indirizzo. Attenzione non sto parlando di una singola università ma di tutte le università italiane. Voglio creare un'associazione iscrizione che di

ogni studente universitario in Italia mi dica qual è la l'università di appartenenza. L'associazione sarà di tipo uno a molti, ogni studente per chiamarsi tale deve essere iscritto ad un'università. L'università ha come identificatore il nome, ad esempio, non ci sono università in Italia che si chiamano nello stesso modo. La matricola dello studente non necessariamente mi basta per rappresentare uno studente perché lo stesso codice matricola lo potrei trovare in università diverse, quindi per poter distinguere uno studente dall'altro non mi basta sapere qual è la sua matricola, ho bisogno di sapere anche qual è l'università di appartenenza. Quindi questo è un esempio in cui studente ha bisogno di essere identificato non soltanto da uno dei suoi attributi ma anche esternamente dall'entità che partecipa all'associazione, quindi questo è un esempio di identificatore esterno. Ogni studente è rappresentato dalla sua matricola tramite l'associazione all'università a cui appartiene. Un'osservazione importante: se ho la necessità di un identificatore esterno è chiaro che la cardinalità dell'associazione dal lato dello studente deve avere cardinalità minima 1, cioè ogni studente deve essere obbligato a partecipare all'associazione perché se non sapessi qual è l'università di appartenenza non potrei distinguerlo da un altro, quindi quando ho un identificatore esterno l'entità coinvolta deve partecipare in modo obbligato all'associazione, altrimenti non posso utilizzare l'associazione per identificare il singolo oggetto dell'entità.



Alcune osservazioni: Ogni entità deve possedere almeno un identificatore, ma può averne in generale più di uno. Una identificazione esterna è possibile solo attraverso una relationship a cui l'entità da identificare partecipa con cardinalità (1,1).

Modello Entità-Relazione-20220323 0803-1

Generalizzazione

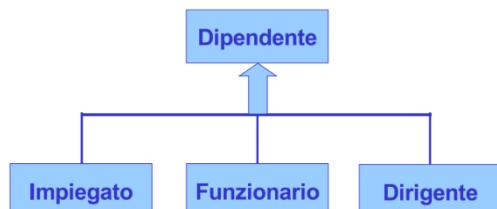
Mettere in relazione una o più entità E1, E2, ..., En con una entità E, che le comprende come casi particolari:

- E è generalizzazione di E1, E2, ..., En
- E1, E2, ..., En sono specializzazioni (o sottotipi) di E

Proprietà delle generalizzazioni

Se E (genitore) è generalizzazione di E1, E2, ..., En (figlie):

- ogni proprietà di E è significativa per E1, E2, ..., En
- ogni occorrenza di E1, E2, ..., En è occorrenza anche di E.



Questo concetto permette di mettere in relazione una o più entità con un'entità padre, che le comprende come casi particolari. Proprietà della generalizzazione: tutte le proprietà dell'entità genitore sono proprietà anche per le entità figlie, quindi tutte le caratteristiche di padre sono anche significative per le figlie e ogni occorrenza delle figlie è anche una occorrenza del padre.

Ereditarietà: tutte le proprietà (attributi, relationship, altre generalizzazioni) dell'entità genitore vengono ereditate dalle entità figlie e non rappresentate esplicitamente.

Tipi di generalizzazioni

- Totale se ogni occorrenza dell'entità genitore è occorrenza di almeno una delle entità figlie, altrimenti è parziale.
- Esclusiva se ogni occorrenza dell'entità genitore è occorrenza di al più *totale ed esclusiva* una delle entità figlie, altrimenti è sovraposta.

Altre proprietà

- Possono esistere gerarchie a più livelli e multiple generalizzazioni allo stesso livello
- Un'entità può essere inclusa in più gerarchie, come genitore e/o come figlia
- Se una generalizzazione ha solo un'entità figlia si parla di sottoinsieme

Esempio: abbiamo una generalizzazione parziale e sovrapposta, l'entità persona, che è l'entità genitore che contiene degli attributi che sono il codice fiscale, il nome e l'età, le due entità figlie, che sono il lavoratore e lo studente. Ogni persona partecipa ad un'associazione con l'entità città, persona vi partecipa con cardinalità (1,1), quindi persona è obbligata a partecipare all'associazione. Perché questa è una generalizzazione parziale? Perché potrebbero esserci delle persone che non sono né lavoratori né gli studenti, quindi nell'entità genitore esistono delle occorrenze che non fanno parte di nessuna delle due entità figlie. Perché è una generalizzazione sovrapposta? Perché potrebbero esserci degli studenti che sono anche lavoratori, quindi l'occorrenza di lavoratore potrebbe anche essere presente in studente. La generalizzazione mi dice che ci sono delle persone e tutte hanno queste caratteristiche, quindi sia i lavoratori che gli studenti ereditano le caratteristiche del padre, quindi il codice fiscale, il nome e l'età, ma anche la residenza in una città. ci sono caratteristiche invece che riguardano soltanto una classe di queste persone, in particolare i lavoratori, che hanno anche uno stipendio.

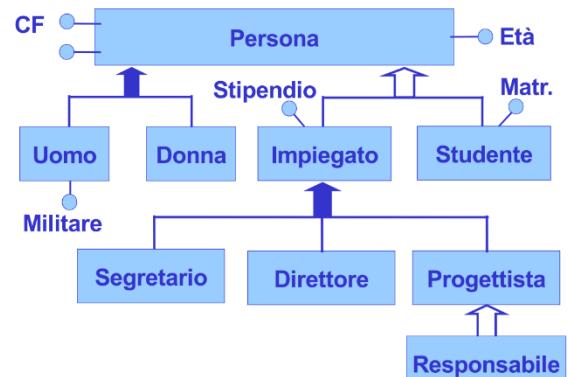
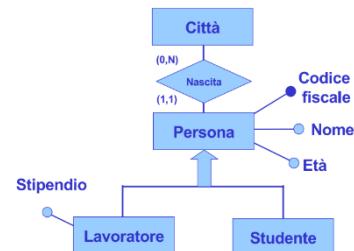
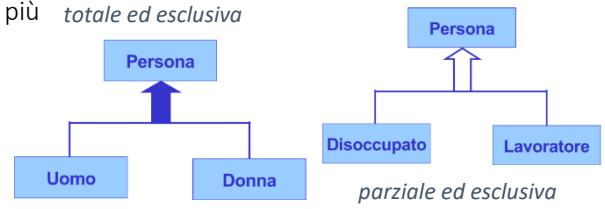
Esempio: Le persone hanno CF, cognome ed età; gli uomini anche la posizione militare; gli impiegati hanno lo stipendio e possono essere segretari, direttori o progettisti (un progettista può essere anche responsabile di progetto); gli studenti (che non possono essere impiegati) un numero di matricola; esistono persone che non sono nè impiegati nè studenti (ma i dettagli non ci interessano).

L'entità principale è persona, che ha come caratteristiche il codice fiscale, l'età e il cognome. Ci sono due generalizzazioni allo stesso livello e poi addirittura una generalizzazione di una generalizzazione. È stata fatta distinzione tra uomini e donne ed è stata fatta distinzione tra impiegati e studenti perché se andiamo a rileggere il testo e infatti si parlava di uomini hanno anche la posizione militare quindi questo mi fa capire che devo distinguere tra categoria uomo e categoria donna e tra impiegati e studenti, siccome poi nel caso degli impiegati si dettagliava anche sulla tipologia di impiegato ovvero segretario, direttore o progettista, ho una generalizzazione a un livello inferiore che mi chiarisce questo aspetto. Non tutti i progettisti lo sono, ma un sottoinsieme può essere responsabile.

Documentazione associata agli schemi concettuali

La documentazione è associata a uno schema, quindi possiamo descrivere quello che si chiama il dizionario dei dati facendo entrare in una descrizione sia dell'entità che delle associazioni che abbiamo utilizzato nel modello, poi dobbiamo esprimere anche i vincoli che con questo modello non posso rappresentare.

- Dizionario dei dati
 - entità
 - relationship
- Vincoli non esprimibili

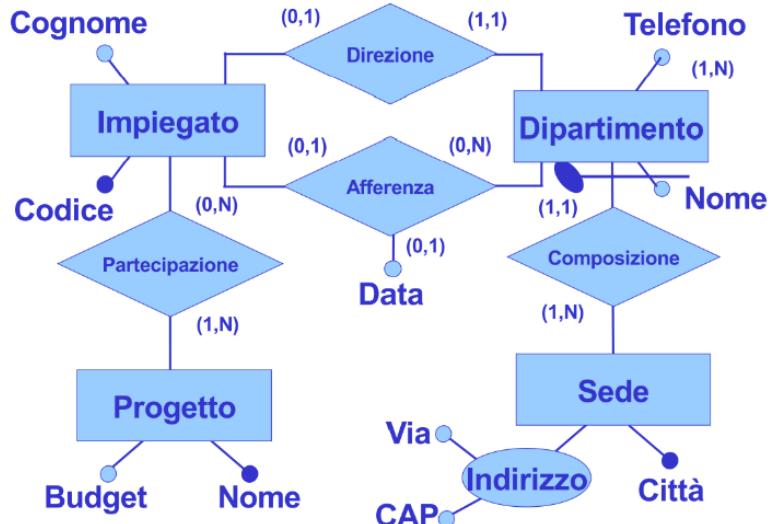


Entità	Descrizione	Attributi	Identificatore
Impiegato	Dipendente dell'azienda	Codice, Cognome, Stipendio	Codice
Progetto	Progetti aziendali	Nome, Budget	Nome
Dipartimento	Struttura aziendale	Nome, Telefono	Nome, Sede
Sede	Sede dell'azienda	Città, Indirizzo	Città

Relazioni	Descrizione	Componenti	Attributi
Direzione	Direzione di un dipartimento	Impiegato, Dipartimento	
Afferenza	Afferenza a un dipartimento	Impiegato, Dipartimento	Data
Partecipazione	Partecipazione a un progetto	Impiegato, Progetto	
Composizione	Composizione dell'azienda	Dipartimento, Sede	

Vincoli di integrità sui dati

- (1) Il direttore di un dipartimento deve afferire a tale dipartimento
- (2) Un impiegato non deve avere uno stipendio maggiore del direttore del dipartimento al quale afferisce
- (3) Un dipartimento con sede a Roma deve essere diretto da un impiegato con più di dieci anni di anzianità
- (4) Un impiegato che non afferisce a nessun dipartimento non deve partecipare a un progetto



Modellazione ER: avvertenze

- Una entità modella un insieme di oggetti
 - Se abbiamo una descrizione del tipo una ditta vuole mantenere informazioni sui suoi impiegati... non esiste nel diagramma la entità Ditta (questa rappresenterebbe un insieme di ditte).
- Una entità deve avere attributi che la caratterizzano.
- Una entità deve avere almeno un identificatore.

- Le associazioni possono avere attributi (ma non chiavi). Se una associazione ha molti attributi esaminare se è il caso di rappresentare questo concetto come entità invece che con una corrispondenza.
- Un diagramma ER modella i dati, non le operazioni che vengono effettuate sui dati.

Progettazione concettuale-20220329 0704-1

Analisi dei requisiti e progettazione concettuale

Comprende le seguenti attività interconnesse:

- acquisizione dei requisiti (in linguaggio naturale)
- analisi dei requisiti
- costruzione dello schema concettuale
- costruzione del glossario, cioè una costruzione di una documentazione che ci permette di capire meglio quali sono gli oggetti dello schema concettuale e quali sono le relazioni tra di essi

Requisiti

L'acquisizione dei requisiti è un'attività difficile e non standardizzabile. Possibili fonti:

- Utenti e committenti, attraverso interviste e documentazione apposita
- Documentazione esistente, ad esempio normative (leggi, regolamenti del settore), regolamenti interni, procedure aziendali, realizzazioni preesistenti (da rimpiazzare o che devono interagire con il nuovo sistema)
- Modulistica

Regole generali per produrre la documentazione dei requisiti

- Cercare di utilizzare sempre lo stesso stile sintattico, evitando termini troppo generici o troppo specifici
- Usare frasi semplici e brevi e separare le frasi sui dati da quelle delle procedure
- Costruire un glossario dei termini: contiene una breve descrizione di ogni termine e gli eventuali sinonimi
- Strutturare i requisiti in gruppi di frasi omogenee

Dovendo realizzare una base di dati da zero si deve partire dai requisiti, quindi da una serie di documenti e interviste degli utenti che ci permettono di capire di che cosa abbiamo bisogno, questa è la fase di progettazione concettuale quindi si progetta la base guidati dal punto di vista concettuale, ovvero da quali sono gli oggetti che sono coinvolti, quali sono le relazioni tra questi oggetti senza preoccuparci del modello logico che andremo a utilizzare. In questa fase si produce

quello che si chiama lo schema concettuale. Nella fase iniziale si deve descrivere a parole quello che è necessario fare, quindi prima di passare allo schema concettuale. Ci sono alcuni consigli di cui tener conto: prima di tutto usare frasi semplici nella descrizione, distinguere bene la parte di descrizione dei dati da quella invece delle procedure delle applicazioni sui dati e cercare sempre di utilizzare lo stesso stile, cioè evitare troppi termini generici e ambiguità nel testo. Si deve costruire il glossario dei termini, cioè una breve descrizione di tutti i concetti dei termini che sono importanti per le fasi di progettazione e anche individuare gli eventuali sinonimi. È importante anche strutturare tutte queste informazioni in gruppi di frasi omogenee, perché questo ci permette di passare allo schema concettuale in modo più semplice.

Esempio: Si vuole realizzare una base di dati per una società che eroga corsi, di cui vogliamo rappresentare i dati dei **partecipanti** ai corsi e dei docenti. Per gli **studenti** (circa 5000), identificati da un codice, si vuole memorizzare il codice fiscale, il cognome, l'età, il sesso, il **luogo** di nascita, il nome dei loro attuali datori di lavoro, i **posti** dove hanno lavorato in precedenza insieme al **periodo**, l'**indirizzo** e il **numero** di telefono, i corsi che hanno frequentato (i corsi sono in tutto circa 200) e il **giudizio finale**.

- Rappresentano lo stesso concetto
- Termini ambigui ed imprecisi

Rappresentiamo anche i **seminari** che stanno attualmente frequentando e, per ogni **giorno**, i **luoghi** e le ore dove sono tenute le lezioni. I **corsi** hanno un codice, un **titolo** e possono avere varie edizioni con date di inizio e fine e numero di partecipanti. Se gli **studenti** sono liberi professionisti, vogliamo conoscere l'area di interesse e, se lo possiedono, il **titolo**. Per quelli che lavorano alle dipendenze di altri, vogliamo conoscere invece il loro livello e la posizione ricoperta.

- Rappresentano lo stesso concetto
- Termini ambigui ed imprecisi
- Stesso termine con concetti diversi

Per gli **insegnanti** (circa 300), rappresentiamo il cognome, l'età, il **posto** dove sono nati, il **nome** del corso che insegnano, quelli che hanno insegnato nel passato e quelli che possono insegnare. Rappresentiamo anche tutti i loro **recapiti telefonici**. I **docenti** possono essere dipendenti interni della società o collaboratori esterni.

Glossario dei termini: in questa tabella è riportata una descrizione del termine e i possibili sinonimi che sono presenti nel testo. Il partecipante è la persona che partecipa ai corsi, il sinonimo che abbiamo trovato nel testo è studente ed è collegato sia a corso che a società. Docente è colui che detiene i corsi, il sinonimo è insegnante, ed è collegato al concetto di corso. Il corso è organizzato dalla società e può avere più edizioni e il sinonimo è seminario ed è collegato al docente. Infine il termine società che come sinonimo ha posti, ed è l'ente presso cui i partecipanti lavorano o hanno lavorato e il collegamento è quello al partecipante. Un glossario permette di fare chiarezza su quelli che sono i vari termini, i sinonimi e i collegamenti tra di essi. Una volta fatto questo lavoro è bene ristrutturare il testo in frasi omogenee, cercando di raggruppare quelle che riguardano i partecipanti, quelle che riguardano i corsi, i docenti e le società.

Termino	Descrizione	Sinonimi	Collegamenti
Partecipante	Persona che partecipa ai corsi	Studente	Corso, Società
Docente	Docente dei corsi. Può essere esterno.	Insegnante	Corso
Corso	Corso organizzato dalla società. Può avere più edizioni	Seminario	Docente
Società	Ente presso cui i partecipanti lavorano o hanno lavorato.	Posti	Partecipante

- Frasi di carattere generale - Si vuole realizzare una base di dati per una società che eroga corsi, di cui vogliamo rappresentare i dati dei partecipanti ai corsi e dei docenti.
- Frasi relative ai partecipanti - Per i partecipanti (circa 5000), identificati da un codice, rappresentiamo il codice fiscale, il cognome, l'età, il sesso, la città di nascita, i nomi dei loro attuali datori di lavoro e di quelli precedenti (insieme alle date di inizio e fine rapporto), le edizioni dei corsi che stanno attualmente frequentando e quelli che hanno frequentato nel passato, con la relativa votazione finale in decimi.
- Frasi relative ai datori di lavoro - Relativamente ai datori di lavoro presenti e passati dei partecipanti, rappresentiamo il nome, l'indirizzo e il numero di telefono.
- Frasi relative ai corsi - Per i corsi (circa 200), rappresentiamo il titolo e il codice, le varie edizioni con date di inizio e fine e, per ogni edizione, rappresentiamo il numero di partecipanti e il giorno della settimana, le aule e le ore dove sono tenute le lezioni.
- Frasi relative a tipi specifici di partecipanti - Per i partecipanti che sono liberi professionisti, rappresentiamo l'area di interesse e, se lo possiedono, il titolo professionale. Per i partecipanti che sono dipendenti, rappresentiamo invece il loro livello e la posizione ricoperta.

- Frasi relative ai docenti - Per i docenti (circa 300), rappresentiamo il cognome, l'et`a, la citt`a di nascita, tutti i numeri di telefono, il titolo del corso che insegnano, di quelli che hanno insegnato nel passato e di quelli che possono insegnare. I docenti possono essere dipendenti interni della societ`a di formazione o collaboratori esterni.

Dai requisiti allo schema concettuale

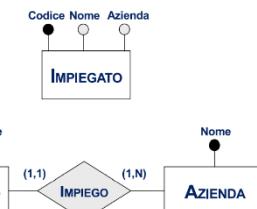
Quale costrutto del modello ER va utilizzato per rappresentare un concetto presente nelle specifiche?

- Se ha proprietà significative e descrive oggetti con esistenza autonoma: entità (es., docente e corso)
- Se `e semplice e non ha proprietà: attributo (es., et`a e citt`a)
- Se correla due o piu concetti: relationship (es., relazione tra partecipanti e corsi)
- Se `e caso particolare di un altro: generalizzazione (es., professionista e dipendente come possibile partecipante)

Esistono alcuni pattern comuni nella progettazione concettuale di una base di dati di cui `e bene tenere conto.

Reificazione di attributo di identità

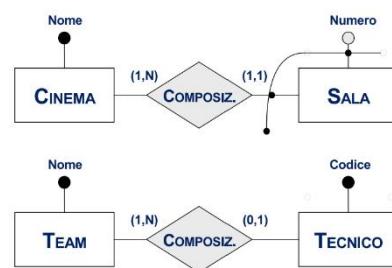
Abbiamo una entità impiegato con attributi il codice, il nome e l'azienda. A seconda del contesto che vogliamo descrivere potrebbe anche essere necessario reificare un attributo, cioè verificare se promuovere il promuovere l'attributo, in questo caso l'azienda è in entità. Creo un secondo schema ER in cui ho sempre l'impiegato con il codice e il nome, però invece di avere l'attributo azienda, ho una nuova entità azienda con un nome ed eventualmente altre caratteristiche. l'impiegato è associato all'azienda tramite un'associazione uno a molti, ovvero ad ogni impiegato è associata un'azienda, quindi dal punto di vista dell'impiegato non cambia niente rispetto a prima. cambia il fatto che adesso un'entità azienda e ad ogni azienda posso associare diversi impiegati.



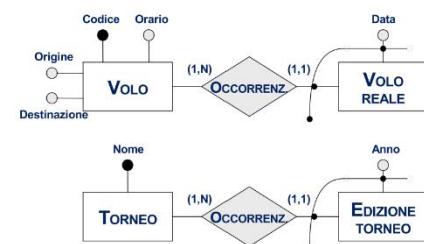
Trasformazione da concetto astratto a classe di oggetti.

Part-of

Nel secondo caso abbiamo le due entità, il team con identificatore nome e un tecnico con identificatore codice. Queste due entità sono collegate per cui ogni tecnico è associato ad un team. In questo schema ER sia il tecnico che il team hanno un'esistenza autonoma. Situazione diversa invece è quella rappresentata dallo schema iniziale: abbiamo sempre un'associazione uno a molti tra l'entità cinema e sala. In questo caso la sala è identificata esternamente dal nome del cinema a cui appartiene, quindi l'entità sala esiste in quanto parte di cinema. Sono sempre due associazioni uno a molti, però nel primo caso l'entità sala non ha un ruolo in quanto è un concetto che è parte di un altro.



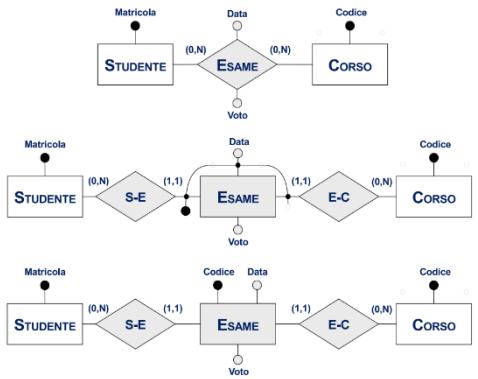
Si vuole rappresentare il fatto che un'entità è parte di un'altra entità: un tecnico ha esistenza autonoma, l'esistenza di una sala dipende dall'esistenza di un cinema.



Le occorrenze di una entità della relazione sono istanze di occorrenze dell'altra entità.

Reificazione di relazione binaria

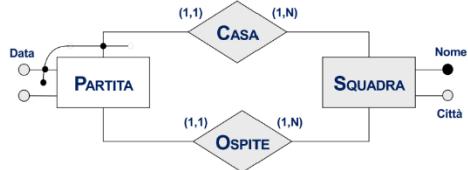
il primo schema rappresenta l'associazione esame tra studente e corso. studenti ha la matricola e corso ha il codice. questa associazione è molti a molti, tutte e due partecipano con cardinalità minima zero ovvero alcuni studenti possono non aver dato nessun esame e che un corso potrebbe non essere stato sostenuto da nessuno studente, ma uno studente può dare tanti esami così come un corso può essere stato sostenuto da tanti studenti. in corrispondenza dell'associazione abbiamo due attributi che sono la data e il voto. questa associazione non è altro che un sottoinsieme del prodotto cartesiano tra studente e corso, questo vuol dire che se io fisso uno studente particolare è un corso particolare questa coppia studente corso può comparire soltanto una volta, con questo tipo di rappresentazione quindi rappresento il superamento di un esame da parte di uno studente. Con questo modello non riesco a rappresentare eventuali bocciature dell'esame, perché una stessa coppia studente corsi o non la posso rappresentare più volte con una un'associazione rappresentare anche gli esami non superati: lo schema in cui ho reificato l'associazione esame in entità, e siccome voglio che una coppia studente-corso possa essere riconosciuta da uno studente, da un corso e dalla sua data. La terza soluzione è quella di utilizzare un identificatore esterno tramite le due entità studente-corso è stato introdotto



La prima soluzione è valida solo se ogni studente può sostenere un esame una sola volta.

Reificazione di relazione ricorsiva

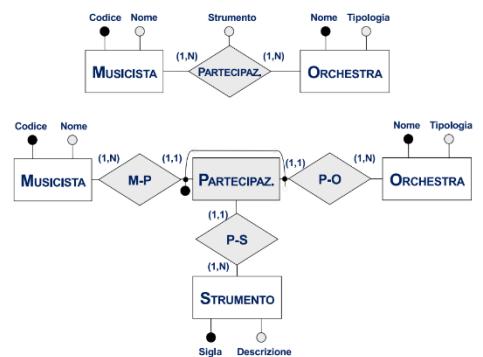
immaginiamo di voler rappresentare partite tra due squadre: si potrebbe realizzare con una semplice relazione ricorsiva tra squadra e se stessa, se faccio così però perdo un dettaglio perché non riesco a rappresentare il fatto che due squadre si possono incontrare più volte, perché in un sottoinsieme del prodotto cartesiano la stessa coppia non la posso considerare due volte. ogni partita adesso è associata alla squadra in due modi diversi tramite un'associazione casa che identifica anche la partita, quindi la partita è identificata dalla data e dalla squadra che gioca in casa, inoltre la partita è anche associata alla squadra tramite un'associazione ospite, è chiaro che in questo modo il ruolo di squadra di casa e di ospite si può invertire.



Una semplice relazione ricorsiva **Partita** tra l'entità **Squadra** (e se stessa) non permette di rappresentare il fatto che due squadre si possono incontrare più volte.

Reificazione di attributo di relazione

Ho un'entità musicista con codice e nome, un'entità orchestra con nome e tipologia e poi ho un'associazione molti a molti con un attributo strumento. Il problema è simile a quello che abbiamo visto tra studenti e corsi, un musicista può suonare strumenti diversi quando suona per orchestre diverse ma quando suona in una certa orchestra deve per forza suonare sempre lo stesso strumento perché la coppia musicista-orchestra compare solo una volta e comparirà con un particolare strumento. Se vogliamo invece dare la possibilità che in corrispondenza di una stessa orchestra un musicista suoni strumenti diversi dobbiamo utilizzare uno schema diverso. Nel secondo schema non ho più un'associazione partecipazione ma ho una promozione a entità e in questo caso l'orchestra. Lo strumento non è più un attributo nell'associazione ma è diventato un attributo dell'entità orchestra. Questo mi permette di avere anche strumenti diversi in corrispondenza di una orchestra.



La prima soluzione è corretta se il musicista può suonare strumenti diversi ma suona sempre lo stesso strumento con una stessa orchestra.

o ha un identificazione tramite il musicista e
un'entità che è associata alla partecipazione.
oppia musicista e orchestra.

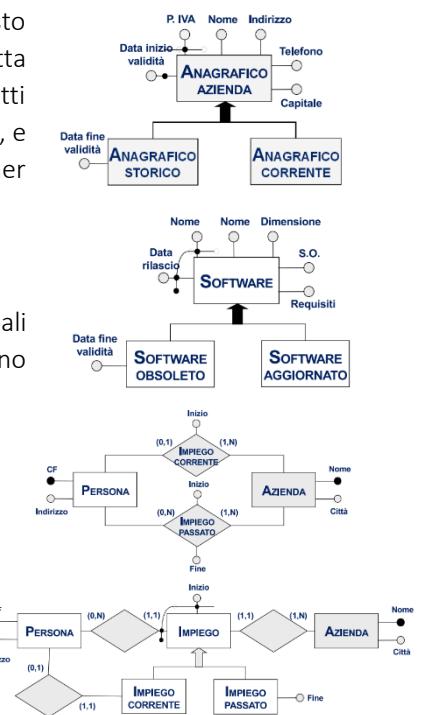
Storicizzazione di concetto

abbiamo l'anagrafica di un'azienda, quindi le informazioni relative ad un'azienda, però alcune aziende potrebbero avere cessato la loro attività e quindi è importante poter fare questa distinzione, ovvero avere l'anagrafica corrente e l'anagrafica

storica con la data fine di validità, abbiamo la generalizzazione che ci permette di fare questo tipo di distinzione. altro esempio sempre per la stessa tipologia: abbiamo il software con tutta una serie di attributi tra cui appunto il nome e la data di rilascio che ci identificano gli oggetti dell'entità e poi abbiamo come casi particolari il software obsoleto, con la data di fine validità, e il software aggiornato. quindi il la generalizzazione è spesso utilizzata quando si vuole tener traccia di cosa è successo nel passato.

Storicizzazione di concetto rappresentato da una associazione tra entità

Per una certa persona vogliamo tener conto sia dell'impiego corrente che degli eventuali impieghi passati. Nella prima soluzione abbiamo queste due entità persone e aziende che sono associate tramite due associazioni, l'impiego corrente con un attributo inizio che ad ogni persona eventualmente associa un'azienda, invece un impiego passato con due attributi che sono la data di inizio e la data di fine, in questo caso l'associazione molti a molti, perché è una persona potrebbe avere avuto tanti impieghi, Così come un'azienda potrebbe avere avuto tanti impiegati. Ci potrebbero essere delle persone che hanno lavorato in periodi diversi per la stessa azienda, in questo schema io non posso rappresentarle tramite l'associazione impiegato passato. Modifico lo schema: è stata inserita un'entità impiego tra persone e azienda, che però è rappresentata tramite una generalizzazione in impiego passato e impiego corrente. Dell'impiego passato si ha l'attributo fine e dell'impiego corrente invece si ha l'associazione con una persona, quindi ogni persona la posso legare ad un'azienda quante volte voglio perché l'entità impiego mi permette di considerare più volte una persona impiegata presso la stessa azienda.



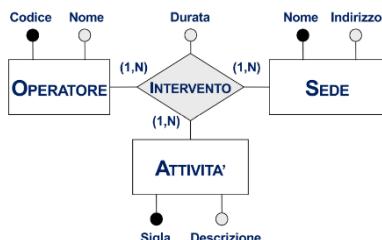
Il primo schema non rappresenta il fatto che una persona possa aver lavorato in periodi diversi per la stessa azienda.

Evoluzione di concetto nel tempo

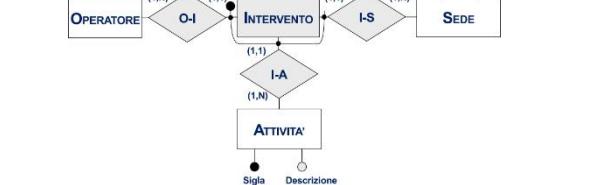
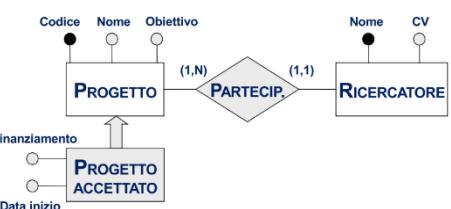
Uso la generalizzazione per rappresentare un concetto che evolve nel tempo: abbiamo l'associazione partecipazione tra ricercatori e progetto. alcuni dei progetti potranno essere finanziati e altri no, quindi c'è una generalizzazione che ci indica quali sono i progetti che sono accettati (e quindi finanziati).

Reificazione di relazione ternaria

l'associazione intervento è un sottoinsieme del prodotto cartesiano di tre entità operatore, attività e sede. voglio rappresentare il fatto che un certo operatore ha svolto una certa attività in una certa sede e poi la durata. questo modello mi permette di considerare la terna operatore, attività e sede una volta sola. altrimenti se invece ho bisogno di rappresentare anche situazioni in cui un operatore magari in una sede ha fatto tante volte tante attività diverse oppure che ha fatto la stessa attività in tante sedi diverse, ho bisogno di rappresentare questo schema in maniera più generale, quindi tramite l'introduzione di una entità intervento.



Si tratta di progetti proposti con l'obiettivo di ottenere un finanziamento.



Questo schema è equivalente al precedente perché la nuova entità risulta identificata da tutte le entità originarie. Ma cambiando l'identificazione di **Intervento** si modellano altre soluzioni per le quali la relazione ternaria non sarebbe corretta.



Solo **Attività** è identificante, lo schema si semplifica.

Strategie di progetto

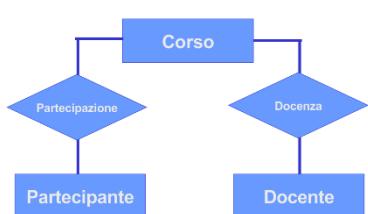
Come si fa a realizzare lo schema?

- Strategia top-down: raffinamenti successivi di uno schema iniziale che descrive tutte le specifiche con pochi concetti molto astratti. Via, via si aumenta il dettaglio dei concetti. Cerco di avere un'idea generale dello schema da rappresentare e partendo da tale schema iniziale rappresentare tutte le specifiche del progetto e poi dettagliare via via i vari concetti, quindi per fare una progettazione seguendo la strategia top-down ho bisogno di avere un'un'idea abbastanza chiara di quale sarà l'organizzazione generale della mia base di dati.
- Strategia bottom-up: le specifiche iniziali sono suddivise in componenti via via sempre più piccole, rappresentate da semplici schemi concettuali che vengono infine integrati. Inizio a rappresentare i concetti suddividendoli in componenti sempre più piccole quindi prima mi preoccupo di rappresentare lo schema di un concetto, poi

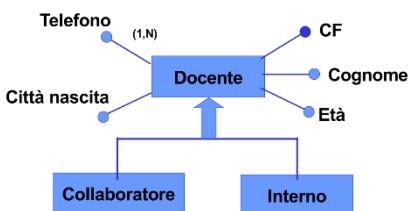
rappresentano lo schema di un altro concetto e così via, fino ad avere tanti piccoli schemi che poi vengono integrati e connessi.

- In pratica, si procede con una strategia mista: si individuano i concetti principali e si realizza uno schema scheletro (rappresenta i concetti principali che si vogliono realizzare) sulla base di questo si può decomporre, poi si raffina, si espande, si integra

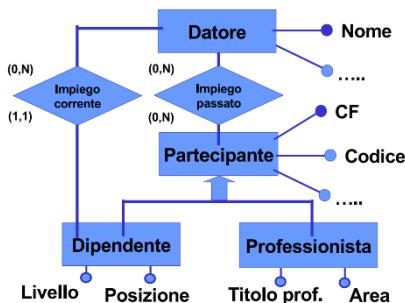
Schema scheletro



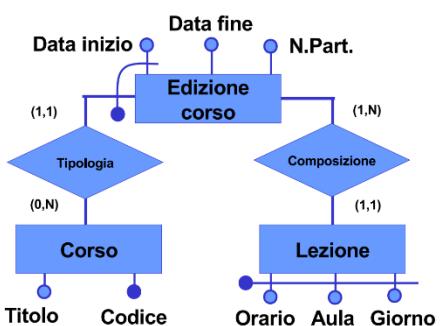
Frasi di carattere generale - Si vuole realizzare una base di dati per una società che eroga corsi, di cui vogliamo rappresentare i dati dei partecipanti ai corsi e dei docenti.



Frasi relative ai docenti



Frasi relative ai partecipanti
Frasi relative ai datori di lavoro
Frasi relative a tipi specifici di partecipanti



Frasi relative ai corsi
Da una descrizione di una società che vuole erogare corsi, abbiamo

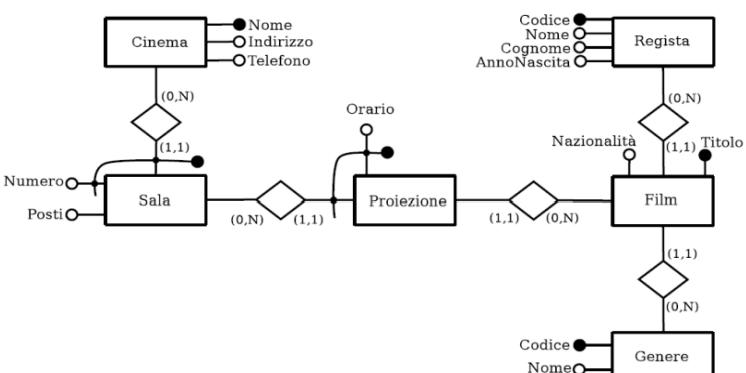
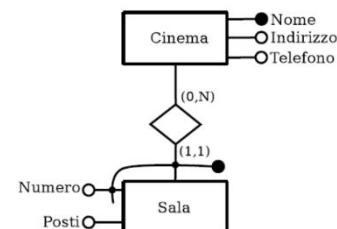
suddiviso il testo in frasi e partendo dalla frase di carattere generale uno schema scheletro che la può essere quello che vedete, in cui ho evidenziato tre entità che sono sicuramente coinvolte nella base di dati che sono i corsi, i partecipanti e i docenti, e le associazioni che devono per forza esistere quindi tra corso e il docente e quella tra corso e partecipante. cerco di dettagliarlo meglio, ad esempio: se mi concentro sui partecipanti, con riferimento alle frasi relative ai partecipanti, capisco quali sono i dettagli di cui ho bisogno, ovvero il codice fiscale ecc.

Progettazione concettuale-20220330 0701-1

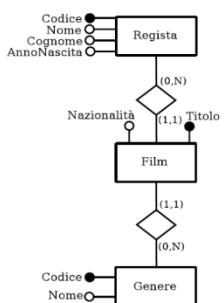
Ogni cinema ha un nome che lo identifica univocamente, un indirizzo e un numero di telefono.



Un cinema è organizzato in più sale, ognuna delle quali ha un codice che la distingue (nell'ambito del cinema) e un numero fissato di posti.

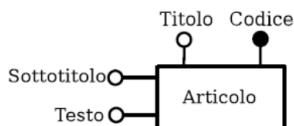


Per ogni film si registrano il titolo, il genere (codice e nome descrittivo), la nazionalità (una semplice stringa) e il regista (con codice identificativo, nome, cognome e anno di nascita).



Gli articoli hanno un titolo, un sottotitolo, uno o più autori e un testo (una stringa molto grande, ma comunque gestibile).

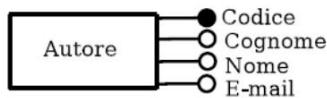
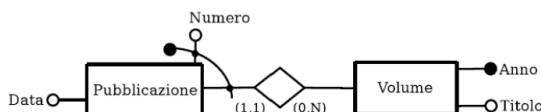
Gli autori hanno nome, cognome, email ed affiliazione (l'istituzione per la quale lavorano).



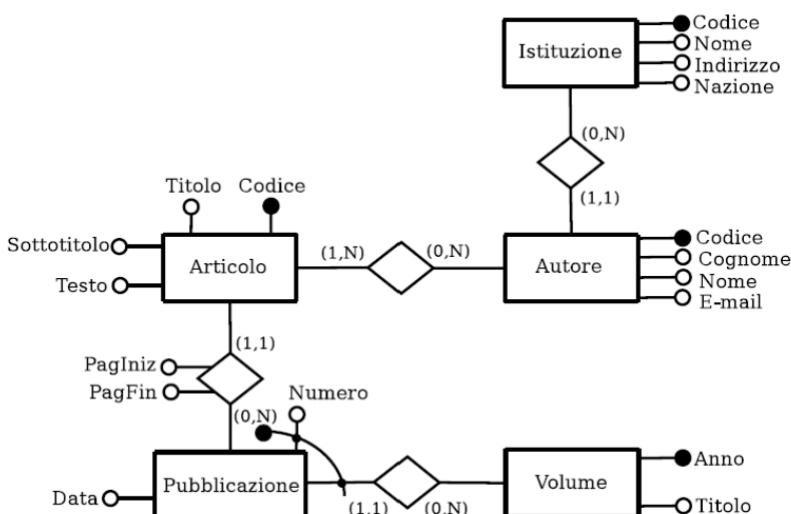
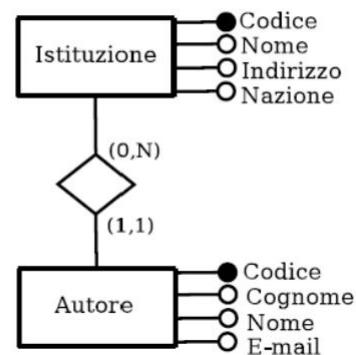
La rivista viene pubblicata un certo numero di volte in un anno. Le pubblicazioni di un anno vengono raccolte in un volume (a cui viene dato un titolo complessivo).



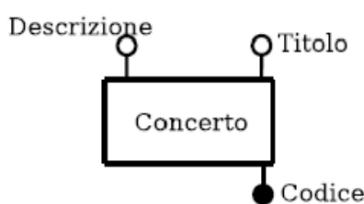
Ogni pubblicazione ha un numero, unico nel rispettivo volume, una data di pubblicazione e una serie di articoli, per ognuno dei quali viene registrata la pagina di inizio e quella di fine.



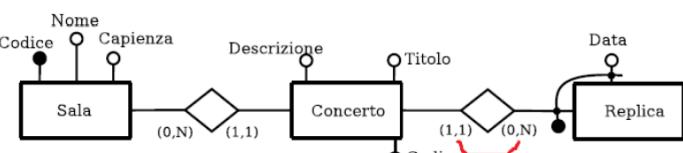
Per ogni istituzione (degli autori) sono d'interesse il nome, l'indirizzo e la nazione.



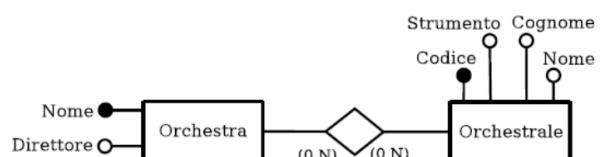
Ogni concerto ha un codice, un titolo e una descrizione ed è composto da una sequenza (ordinata) di pezzi musicali.



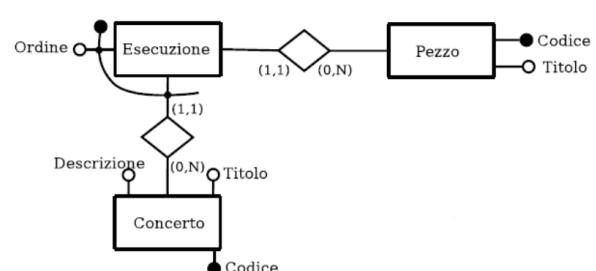
Ogni concerto è tenuto più volte, in giorni diversi, ma sempre nella stessa sala. Ogni sala ha un codice, un nome e una capienza.

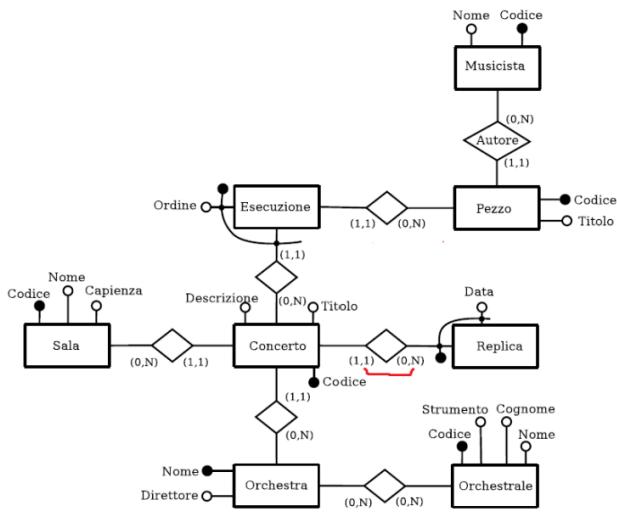


Ogni concerto è eseguito da un'orchestra: ogni orchestra ha un nome, un direttore (del quale interessano solo nome e cognome) e un insieme di orchestrali. Ogni orchestrale ha una matricola (univoca nell'ambito della base di dati), nome e cognome, può partecipare a più orchestre, in ciascuna delle quali suona uno e un solo strumento, ma in orchestre diverse può suonare strumenti diversi.

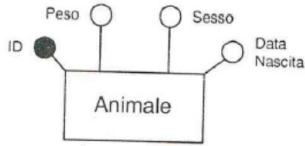


Ogni concerto ha un codice, un titolo e una descrizione ed è composto da una sequenza (ordinata) di pezzi musicali. Ogni pezzo ha un codice, un titolo e un autore (con codice e nome); uno stesso pezzo può essere rappresentato in diversi concerti.

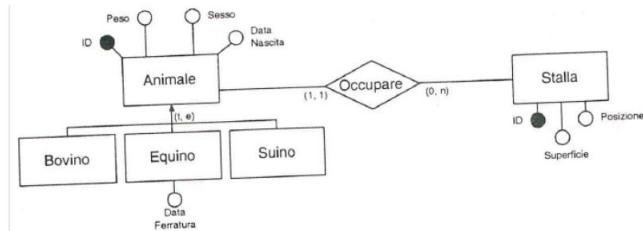




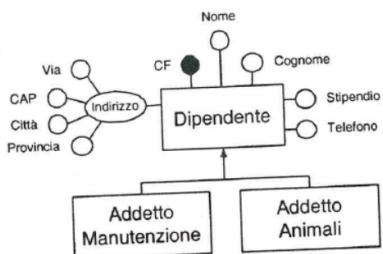
Una fattoria è specializzata nell'allevamento di animali. Di ciascun **animale** si conoscono: il codice identificativo, il peso espresso in chilogrammi, il sesso e la data di nascita.



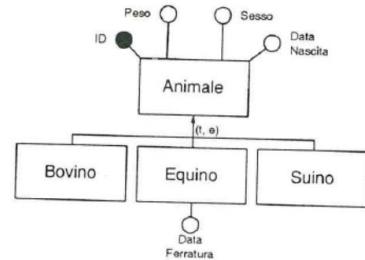
Ogni animale vive in una stalla. Di ciascuna **stalla** si conosce la posizione geografica e la superficie occupata, espressa in metri quadrati.



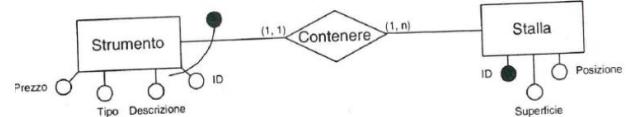
Nella fattoria lavorano diversi **dipendenti**, di cui si conosce l'anagrafica, che include il codice fiscale, nome, cognome, indirizzo (composto da via, CAP, città e provincia) e numero di telefono, e lo stipendio mensilmente percepito. I dipendenti si dividono in due categorie: gli addetti alla manutenzione e coloro che si occupano degli animali.



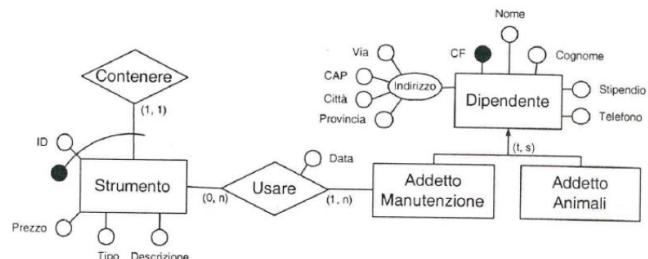
In particolare, gli animali si distinguono fra suini, bovini ed equini. Di questi ultimi si vuole tenere traccia della data di ultima ferratura, per poter programmare la prossima.



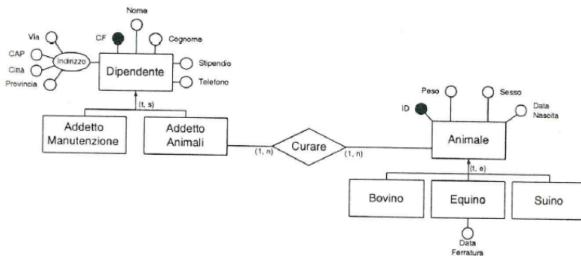
Ogni stalla è equipaggiata con gli strumenti necessari alla sua manutenzione. In particolare, di ogni **strumento** si vuole tenere traccia della descrizione, del tipo di attrezzo, del suo prezzo di acquisto e della stalla dove è collocato. Ogni strumento è identificato da un codice, unico all'interno della stalla.



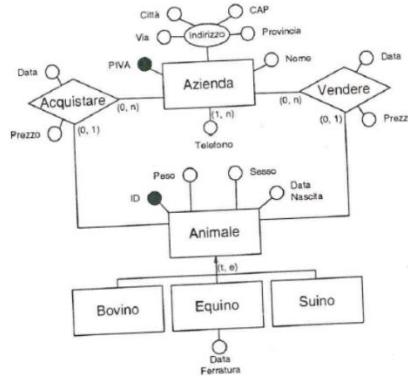
I primi, per svolgere il loro compito, utilizzano gli attrezzi siti nelle varie stalle; per ogni utilizzo degli attrezzi si vuole tenere traccia della data di utilizzo.



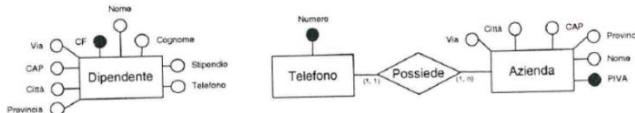
Degli altri (coloro che si occupano degli animali), invece, si vuole mantenere l'elenco degli animali di cui si occupano. Si noti che ciascun dipendente può occuparsi, in momenti diversi della giornata, sia della manutenzione sia della cura degli animali, a seconda della necessità del momento.



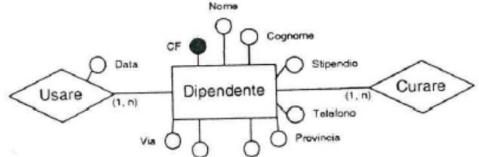
Ogni volta che un animale viene acquistato è necessario mantenere la data di acquisto e il prezzo pagato; allo stesso modo in caso di vendita si tiene traccia della data e del prezzo di vendita.



Eliminazione di attributi composti e attributi multipli.

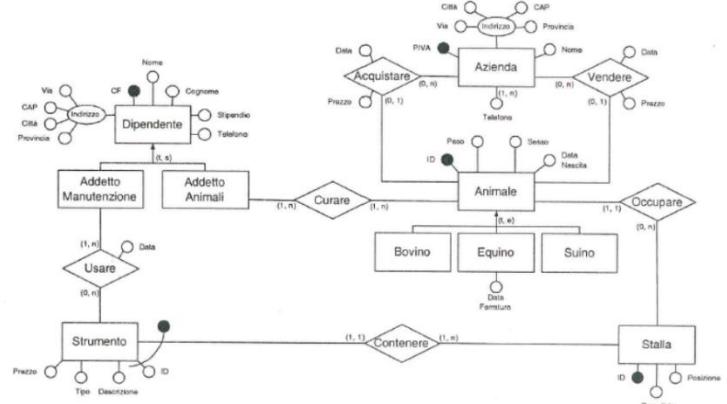
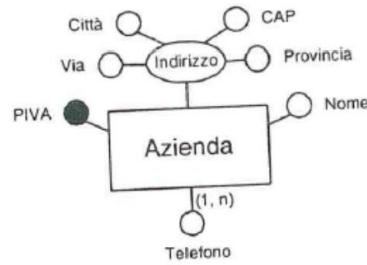


Eliminazione della generalizzazione dell'entità dipendente (totale e sovrapposta).

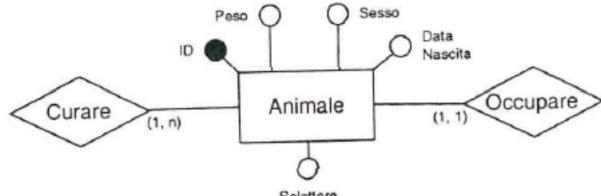


Da aggiungere anche due selettori per indicare se un dipendente si occupa della manutenzione, degli animali o di entrambi le cose.

La fattoria intrattiene rapporti commerciali con **aziende** esterne per l'acquisto e la vendita degli animali. Ciascuna di queste aziende è caratterizzata da una Partita IVA, un nome, un indirizzo (composto da via, CAP, città e provincia) e uno o più numeri telefonici.



Eliminazione della generalizzazione dell'entità animale (totale ed esclusiva).



Da aggiungere anche l'attributo opzionale data ferratura.

Schema logico risultante

Relazione	Attributi
AZIENDA	PIVA, Nome, Via, Cap, Città, Provincia
ANIMALE	ID, Peso, Sesso, DataNascita, DataFerratura, Selettore, IDStalla
STALLA	ID, Superficie, Posizione
STRUMENTO	IDStalla, ID, Descrizione, Tipo, Prezzo
DIPENDENTE	CF, Nome, Cognome, Stipendio, Telefono, Via, Cap, Città, Provincia, CuraAnimali, Manutentore
TELEFONO	Numer, PIVAAzienda
ACQUISTARE	IDAnimale, PIVAAzienda, Data, Prezzo
VENDERE	IDAnimale, PIVAAzienda, Data, Prezzo
USARE	IDStalla, IDInstrumento, CFIDipendente, Data
CURARE	IDAnimale, CFIDipendente

Chiave Esterna	Referenza
TELEFONO.PIVAAzienda	AZIENDA.PIVA
ANIMALE.IDStalla	STALLA.ID
ACQUISTARE.IDAnimale	ANIMALE.ID
ACQUISTARE.PIVAAzienda	AZIENDA.PIVA
VENDERE.IDAnimale	ANIMALE.ID
VENDERE.PIVAAzienda	AZIENDA.PIVA
STRUMENTO.IDStalla	STALLA.ID
USARE.{IDStalla, IDInstrumento}	STRUMENTO.{IDStalla, ID}
USARE.CFIDipendente	DIPENDENTE.CF
CURARE.IDAnimale	ANIMALE.ID
CURARE.CFIDipendente	DIPENDENTE.CF

Obiettivo della progettazione logica

Tradurre lo schema concettuale in uno schema logico che rappresenti gli stessi dati in maniera corretta ed efficiente. In ingresso:

- schema concettuale
- informazioni sul carico applicativo (in termini di dimensione dei dati e caratteristiche delle operazioni)
- modello logico

In uscita:

- schema logico
- documentazione associata

Ristrutturazione schema ER

Motivazioni:

- semplificare la traduzione
- ottimizzare le prestazioni

Osservazione:

- uno schema ER ristrutturato non è (più) uno schema concettuale nel senso stretto del termine

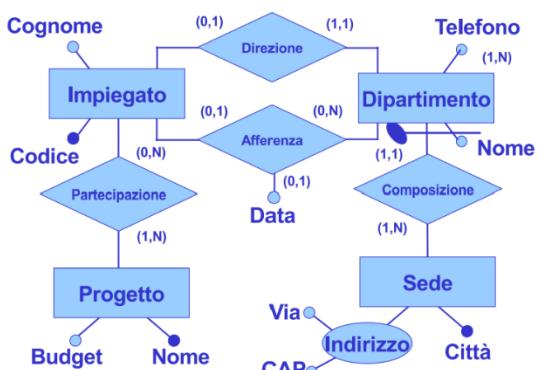
Valutazione delle prestazioni

Si considerano indicatori dei parametri che regolano le prestazioni.

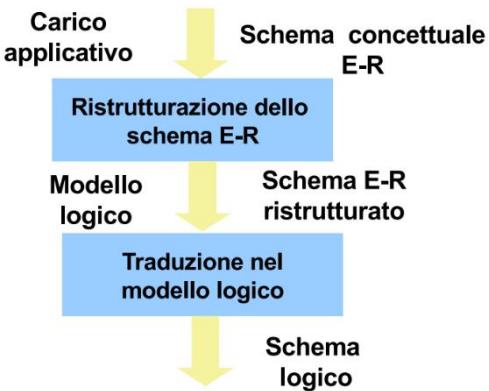
- Spazio: numero di occorrenze previste per le entità e le relazioni.
- Tempo: numero di occorrenze visitate durante un'operazione.

Nell'operazione di traduzione le entità e le associazioni possono essere tradotte facilmente nel modello ma ci sono alcune costrutti che non trovano un corrispondente diretto nel modello relazionale, soprattutto le generalizzazioni, quindi è importante quando si fa una progettazione concettuale utilizzare questo costrutto perché ci permette di chiarire quali sono i sottoinsiemi di un'entità e che caratteristiche hanno, però prima della traduzione quello che dovremmo fare è eliminare queste generalizzazioni, cioè dovremmo fare un'operazione che si chiama per ristrutturazione dello schema entità relazione, cioè eliminando le generalizzazioni, gli attributi composti e gli attributi multivaleure, perché questi costrutti non hanno un corrispondente nel modello relazionale, quindi quando si parla di progettazione logica si intende proprio fare una ristrutturazione del modello concettuale in vista poi della traduzione. Qual è l'obiettivo della progettazione logica è quello di tradurre lo schema concettuale nello schema logico che abbiamo scelto, il nostro caso nel modello relazionale, quindi in input abbiamo lo schema concettuale ma abbiamo bisogno anche di informazioni sul carico applicativo, cioè per poter fare delle scelte poi anche di ristrutturazione del modello è importante capire anche quali saranno le dimensioni dei miei dati, perché sulla base di queste informazioni posso fare delle scelte che sono più vantaggiose rispetto ad altre.

Questo schema che contiene informazioni su impiegati, dipartimenti, progetti e sede.



Traduzione del schema concettuale in uno schema logico.



Concetto	Tipo	Volume
Sede	E	10
Dipartimento	E	80
Impiegato	E	2000
Progetto	E	500
Composizione	R	80
Afferenza	R	1900
Direzione	R	80
Partecipazione	R	6000

Un dipartimento appartiene ad una ed una sola sede (Dipartimento e Composizione).

Ci sono impiegati che non afferiscono a un dipartimento (Impiegato e Afferenza).

Ogni impiegato partecipa in media a 3 progetti (Impiegato e Partecipazione).

rappresenta la dimensione che ci aspettiamo per quell'entità o per

quell'associazione, quindi nella fase di raccolta dei requisiti bisognerà capire con le interviste o dalla documentazione qual è il volume dei dati. Immaginiamo che le sedi siano 10, che ci siano 80 dipartimenti, 2000 impiegati e che i progetti in corso

siano 500. Per quanto riguarda le associazioni dobbiamo tener conto anche del tipo di associazione che è stata considerata, ad esempio l'associazione composizione è un'associazione che lega il dipartimento alla sede, ma il dipartimento è identificato esternamente dalla sede, quindi vuol dire che le occorrenze dell'associazione dovranno essere tante quante sono i dipartimenti. L'associazione afferenza di un impiegato al dipartimento: tutte e due le entità partecipano con cardinalità minima zero, quindi non tutti gli impiegati parteciperanno a questa associazione, e immaginiamo che soltanto 1900 siano parte dall'associazione. L'associazione direzione deve per forza avere come volume 80, perché ci aspettiamo che ogni dipartimento abbia un direttore. L'associazione partecipazione che lega gli impiegati al progetto, dipende da quanti progetti in media ogni impiegato dovrà seguire, quindi facciamo l'ipotesi che ogni impiegato mediamente dovrà seguire tre progetti, avevamo 2000 impiegati quindi complessivamente abbiamo un volume di 6000. Il volume delle varie entità è relativo alla dimensione dell'insieme, l'associazione rappresenta invece un prodotto cartesiano fra le entità coinvolte quindi in pratica stiamo definendo la dimensione di quel prodotto cartesiano.

se devo valutare una certa operazione devo costruire su quell'operazione la tavola degli accessi basandomi su uno schema di navigazione. Esempio: trovare tutti i dati di un impiegato, del dipartimento nel quale lavora e dei progetti ai quali partecipa. Si costruisce una tavola degli accessi basata su uno schema di navigazione. Devo considerare la parte dello schema che è coinvolta, quindi per ogni impiegato devo, tramite l'associazione afferenza, andare a vedere quale dipartimento afferisce, quali sono le informazioni del dipartimento e per ogni impiegato dovrò anche, tramite l'associazione partecipazione, andare a vedere quali sono i progetti in cui è coinvolto. Posso fare anche delle ipotesi sui tempi di questa operazione, quindi se voglio trovare gli impiegati e tutte le informazioni relative sia ai dipartimenti che ai progetti dovrò accedere all'entità in lettura e per andare a trovare l'impiegato dovrò accedere ad associazione afferenza sempre in lettura, per andare a individuare il dipartimento dovrò accedere all'entità dipartimento per andare a prendere le caratteristiche del dipartimento coinvolto e per quanto riguarda la partecipazione e il progetto devo prevedere un numero di accessi a tre perché ho previsto che ogni impiegato partecipi mediamente a tre di questi progetti, quindi vuol dire che dovrà per ogni impiegato percorre questo tratto dello schema tre volte.

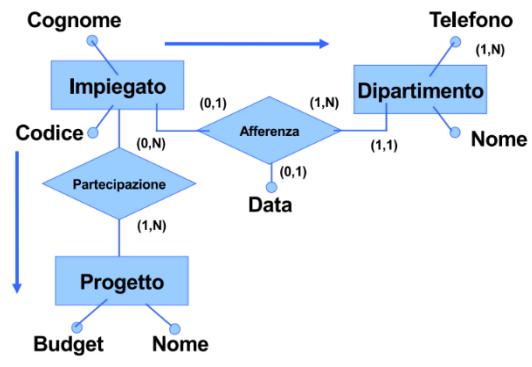
Attività della ristrutturazione

- Analisi delle ridondanze: ovvero informazioni che posso ricavare in più di un modo.
- Eliminazione delle generalizzazioni.
- Partizionamento/accorpamento di entità e relationship.
- Scelta degli identificatori primari: perché ogni entità deve avere un identificatore, tali identificatori non devono essere troppo complessi, per cui se ci sono tanti attributi che servono per identificare un oggetto allora forse vale la pena ripensarci e introdurne anche di nuovi.

Analisi delle ridondanze

Una ridondanza in uno schema ER è una informazione significativa ma derivabile da altre. In questa fase si decide se eliminare le ridondanze eventualmente presenti o mantenerle (o anche di introdurre di nuove). Per fare tale scelta posso consultare le tavole dei volumi o degli accessi.

- Vantaggi: semplificazione delle interrogazioni.
- Svantaggi: appesantimento degli aggiornamenti e maggiore occupazione di spazio



Concetto	Costrutto	Accessi	Tipo
Impiegato	E	1	Lettura
Afferenza	R	1	Lettura
Dipartimento	E	1	Lettura
Partecipazione	R	3	Lettura
Progetto	E	3	Lettura

Forme di ridondanza in uno schema ER

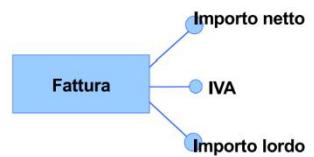
Le forme di ridondanza che possono essere presenti attributi derivabili ovvero attributi che sono derivabili da altri attributi della stessa entità o relationship, oppure da attributi di altre entità o relationship, oppure ci possono essere delle relationship che sono derivabili dalla composizione di altre.

Attributi derivabili:

- da altri attributi della stessa entità (o relationship);
- da attributi di altre entità (o relationship).

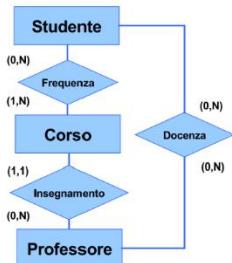
Relationship derivabili dalla composizione di altre (più in generale: cicli di relationship).

Esempio di attributo derivabile



- Uno degli attributi è deducibile dall'altro attraverso una operazione di somma o differenza.

Ridondanza dovuta a ciclo

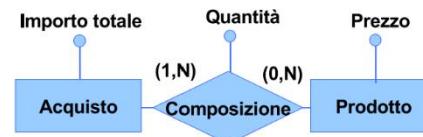


- L'associazione Docenza tra Studente e Professore può essere derivata dalle associazioni Frequenza e Insegnamento. Se l'associazione avesse un attributo Tesi allora lo schema non sarebbe ridondante.

Esempio di un'analisi di una ridondanza: abbiamo una persona, la città e l'associazione residenza di tipo uno a molti. Una persona risiede in una città e in una città possono risiedere tante persone. In corrispondenza di città compare un attributo numero abitanti.

Numero abitanti è un attributo ridondante perché per sapere quante persone ci abitano posso utilizzare l'associazione residenza e vedere quante sono le persone che partecipano a questa associazione. Meglio togliere o lasciare questo attributo? Per prendere questa decisione ho bisogno di valutare la dimensione dei dati sia di valutare le operazioni che su questi dati voglio effettuare. Ipotesi sul volume dei dati: immaginiamo che le città che voglio gestire siano 200 e di avere un milione di persone, per cui avrò lo stesso numero anche per quanto riguarda l'associazione residenza, e immaginiamo di voler fare queste due operazioni: la prima operazione consiste nel memorizzare una nuova persona con la relativa città di residenza e immaginiamo di voler fare questa operazione 500 volte al giorno, la seconda operazione invece è di tipo diverso, devo stampare tutti dati di una città incluso il numero di abitanti e voglio fare questa operazione due volte al giorno. Voglio capire quanto mi costano queste due operazioni sia in presenza di ridondanza sia togliendo l'attributo ridondante. quanto mi costa mantenere l'attributo numero abitanti? non costa tanto perché, se immaginiamo di avere un attributo che mi occupa quattro byte e di avere 200 città, per mantenere quell'attributo ho bisogno di 800 byte. quanto mi costano le due operazioni in presenza di ridondanza? se l'attributo è presente allora per l'operazione 1 devo memorizzare ogni nuova persona con la relativa città di residenza quindi devo accedere a persona in scrittura perché devo memorizzare una nuova persona e siccome devo memorizzare la sua residenza dovrò accedere in scrittura anche a residenza e dovrò accedere a Città sia in lettura per trovare la città in cui la persona risiede ma anche scrittura perché ho bisogno di andare a scrivere a incrementare il valore il numero degli abitanti di quella città, quindi con ridondanza l'entità persona dovrò accenderla in scrittura, l'entità residenza la devo accedere

Esempio di attributo derivabile da altra entità



- L'importo totale si può derivare, attraverso l'associazione Composizione, dagli attributi Prezzo e Quantità



L'attributo Numero abitanti può essere derivato contando le occorrenze dell'associazione Residenza a cui una città partecipa.

Meglio togliere o lasciare l'attributo Numero abitanti?

Concetto	Tipo	Volume
Città	E	200
Persona	E	1000000
Residenza	R	1000000

Operazione 1: memorizza una nuova persona con la relativa città di residenza (500 volte al giorno).

Operazione 2: stampa tutti i dati di una città, incluso il numero di abitanti (2 volte al giorno).

Immaginando che l'attributo Numero abitanti occupi 4 byte, avremo un'occupazione di $200 \times 4 = 800$ byte in più per mantenere questa informazione.

Operazione 1 e 2 in presenza di ridondanza

- Operazione 1: 1500 S + 500 L accessi

Concetto	Costrutto	Accessi	Tipo
Persona	E	1	Scrittura
Residenza	R	1	Scrittura
Città	E	1	Lettura
Città	E	1	Scrittura

- Operazione 2: 2 L accessi, costo trascurabile

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura

Operazione 1 e 2 in assenza di ridondanza

- Operazione 1: 1000 S accessi

Concetto	Costrutto	Accessi	Tipo
Persona	E	1	Scrittura
Residenza	R	1	Scrittura

- Operazione 2: 10002 L accessi (il numero 5000 rappresenta un valore medio ottenuto dividendo il numero di persone per il numero di città, 1000000/200)

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura
residenza	R	5000	Lettura

sempre in scrittura, città sia in lettura che scrittura siccome abbiamo detto che immaginiamo di fare questa operazione 500 volte al giorno complessivamente al giorno spenderò 1500 accessi in scrittura +500 accessi in lettura. la seconda operazione ha un costo veramente trascurabile se tengo l'attributo ridondante perché per stampare il numero di residenti dovrò semplicemente accedere all'entità città in lettura e quindi un costo praticamente nullo. invece se tolgo l'attributo ridondante dovrò semplicemente aggiungere una persona, quindi scrittura, e la sua residenza, quindi in questo caso si snellisce l'operazione perché mi bastano 1000 accessi e non ho bisogno di incrementare l'attributo. per la seconda operazione invece qui le cose cambiano in maniera sostanziale perché per stampare il numero di residenze di ogni città io dovrò accedere all'entità città in lettura e ma poi dovrò percorrere l'associazione residenza per capire quante sono le persone che ci abitano. abbiamo fatto l'ipotesi che ci siano un milione di persone per città, che sono 200, quindi mediamente ogni città avrà 5000 abitanti quindi mi posso aspettare che l'associazione contenga 5000 occorrenze e per poterle contare ho bisogno di accenderle tutte, quindi la seconda operazione mi costa 10.002 accessi in lettura. a questo punto possiamo capire quale delle due soluzioni è la più conveniente.

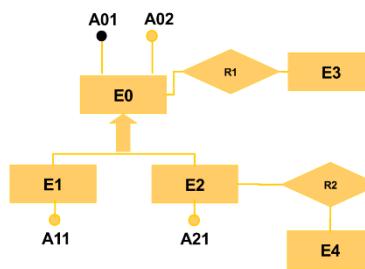
Costi delle operazioni 1 e 2 in presenza e assenza di ridondanza

- Assumiamo che un accesso in scrittura costi il doppio di quello in lettura ($1S = 2L$).
- In presenza di ridondanza: $\text{costo1} + \text{costo2} = 1500S + 500L + 2L = 3502L$ accessi al giorno In assenza di ridondanza: $\text{costo1} + \text{costo2} = 1000S + 10002L = 12002L$ accessi al giorno
- In assenza di ridondanza si fanno circa 8500 accessi in più contro un'occupazione di 800 byte in più per gestire la ridondanza.

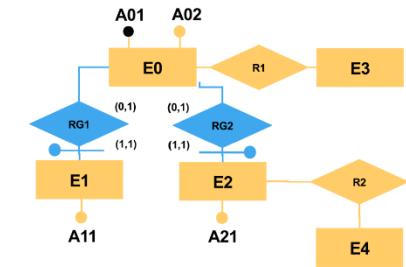
Eliminazione delle gerarchie

Il modello relazionale non può rappresentare direttamente le generalizzazioni. Entità e relationship sono invece direttamente rappresentabili. Si eliminano perciò le gerarchie, sostituendole con entità e relationship.

Esempio: abbiamo un'entità E0 che ha un identificatore, poi uno o più attributi, le due figlie E1 e E2, che avranno in generale degli attributi propri, e poi abbiamo delle associazioni sia a livello di padre sia a livello di entità figlia. Questo è uno schema abbastanza generale di generalizzazioni. Cosa vuol dire eliminare la generalizzazione? Ci sono diverse strategie che si possono utilizzare, una consiste nell'eliminare le due entità figlie accorpandole nel padre: gli attributi dell'entità E0 continuano ad essere presenti l'identificatore e gli attributi delle due entità figlie sono diventati attributi dell'entità padre, però sono attributi con cardinalità minima zero, il che vuol dire che ci sono delle occorrenze dell'entità che potranno avere un attributo e altre che ne potranno avere altri, perché E1 e E2 rappresentavano dei sottoinsiemi di E0, quindi alcuni avevano l'attributo A11 e altri A21, eventualmente anche tutti e due se la generalizzazione fosse sovrapposta, quindi questo lo risolvo semplicemente mettendoci (0,1) come cardinalità, perché non tutti possono avere questi attributi. Ho bisogno di aggiungere un attributo tipo che mi permette di dire se quell'oggetto apparteneva a un'entità figlia. Mi devo occupare ovviamente delle altre associazioni, quindi l'associazione R1 che esisteva tra E0 e E3 non viene coinvolta, però l'associazione R2 invece tra la figlia E2 è un'altra entità E4 me la devo portare in E0 e però parteciperà a questa associazione con cardinalità minima zero per lo stesso motivo di prima, potrebbero esserci delle occorrenze, in particolare quelle che facevano che concettualmente parte dell'altra entità, che non parteciperanno a questa associazione. Un'altra soluzione è accorrpare il genitore nelle figlie, quindi elimino l'entità padre e lascio le due entità figlie. Le due figlie ereditano tutti gli attributi del padre, quindi in particolare A01 e A02 che erano l'identificatore e gli attributi vengono ereditati, e poi dobbiamo creare due nuove associazioni tra E1 e l'entità E3 e lo stesso con E2. Questo tipo di trasformazione conviene se gli accessi alle figlie sono distinti, quindi se le mie operazioni fanno distinzione tra una figlia e l'altra e ovviamente se la generalizzazione totale.

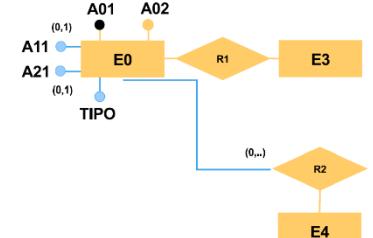


Sostituzione della generalizzazione con associazioni



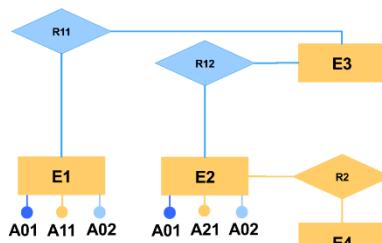
Devono essere aggiunti vincoli: ogni occorrenza di E0 o appartiene ad una occorrenza di RG1 oppure ad una di RG2. Conviene se gli accessi alle figlie sono separati dagli accessi al padre.

Accorpamento delle figlie nel genitore



L'attributo **Tipo** serve a distinguere il tipo di occorrenza di E0. A11 e A21 possono avere cardinalità 0, E0 partecipa a R2 con cardinalità minima 0 perché le occorrenze di E2 sono un sottoinsieme di quelle di E0. Conviene se le operazioni non fanno tanta distinzione tra E0, E1, E2.

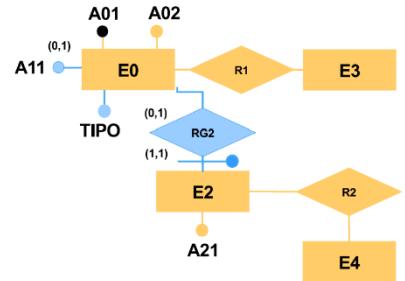
Accorpamento del genitore nella figlia



R11 e R12 sono restrizioni della relazione R1 sulle occorrenze delle entità E1 e E2. Conviene se gli accessi alle figlie sono distinti e la generalizzazione è totale.

esistono anche altre soluzioni: ad esempio quella di sostituire la generalizzazione con delle associazioni ed è la soluzione che mi permette di mantenere sia l'entità padre che le entità figlie e in questo caso quello che devo fare è creare delle associazioni uno a uno tra il padre e le figlie con partecipazione obbligatoria lato e1 ed e2 ma anche identificate esternamente da e0 perché tutte le caratteristiche del padre sono caratteristiche delle figlie. Mentre lato e0 l'associazione ha la cardinalità minima dall'associazione uguale a zero perché un'occorrenza può partecipare ad una o all'altra associazione. Questo tipo di soluzione può essere conveniente quando ho delle operazioni che fanno riferimento sia al padre che alle figlie. Ci possono essere anche delle soluzioni ibride, scelgo una soluzione per certe figlie e un'altra per altre figlie, ad esempio: ho accorpato e1 con e0 mentre e2 l'ho lasciata trasformando ovviamente la generalizzazione in associazione.

Una soluzione ibrida



E0 e E1 sono state accorpate, E2 è rimasta separata.

Partizionamento/accorpamento di entità e relationship

Ristrutturazioni effettuate per rendere più efficienti le operazioni in base a un semplice principio. Gli accessi si riducono:

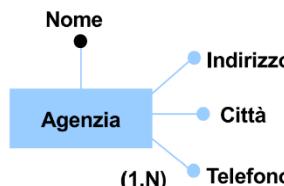
- separando attributi di un concetto che vengono acceduti separatamente;
- raggruppando attributi di concetti diversi acceduti insieme.

Casi principali:

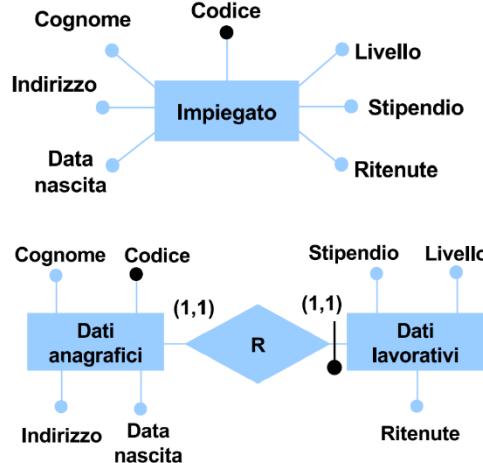
- Partizionamento verticale di entità
- Partizionamento orizzontale di relationship
- Eliminazione di attributi multivalore
- Accorpamento di entità/relationship

L'altra attività che si fa durante questa fase il partizionamento o l'accorpamento di entità e relationship. L'idea alla base di questa attività è che può essere utile separare attributi di un concetto se io accedo a questi attributi separatamente oppure raggrupparli se invece tutte le volte che accedo a un certo attributo accedo anche ad un altro. Esempio partizionamento verticale: abbiamo un impiegato e in corrispondenza di questo impiegato abbiamo tante informazioni sia sull'anagrafica dell'impiegato, quindi il cognome, l'indirizzo, la data sia informazioni di tipo diverso, quindi le chiamate al lavoro dell'impiegato, il livello, lo stipendio e le ritenute. Potrebbe essere utile trasformare quell'entità in due entità, quindi dividere praticamente le informazioni in due entità separate associate da un'associazione uno a uno, cioè ad ogni entità di tipo anagrafico deve essere associata l'informazione relativa ai dati lavorativi e i dati lavorativi sono identificati dal codice della persona tramite l'associazione.

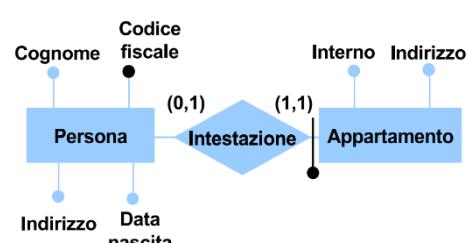
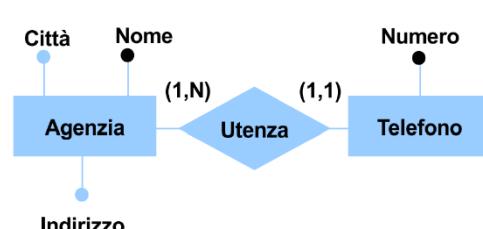
Eliminazione di attributi: quello che si può fare è trasformare quell'attributo in un'associazione, quindi aggiungo un'entità telefono con i numeri di telefono e associo questa entità tramite l'associazione utenza uno a molti all'agenzia, quindi si eliminano gli attributi e si fanno diventare delle associazioni uno a molti.

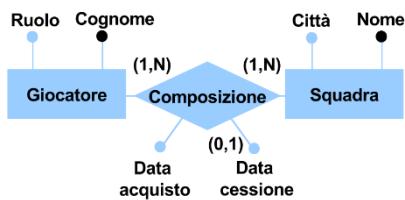


Il modello relazionale non permette di rappresentare direttamente questo tipo di attributi.



Ristrutturazione conveniente se le operazioni coinvolgono o i dati anagrafici o i dati lavorativi.

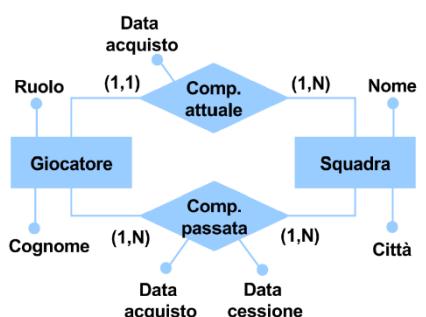




Accorpamento di entità: un'entità persona con informazioni codice fiscale, cognome, indirizzo, data di nascita e poi informazioni invece relative all'indirizzo all'appartamento in cui vive, quindi dell'appartamento conosciamo l'interno e l'indirizzo. L'associazione intestazione è un'un'associazione a uno a uno, l'appartamento ovviamente è associato a una persona mentre possono esserci delle persone che non hanno intestato nessun appartamento. Allora se l'applicazione di

cui ho bisogno tutte le volte che accede ad una persona ha bisogno di avere informazioni sul suo eventuale appartamento, posso valutare di accorpare le informazioni delle due entità in una sola entità, ovviamente l'interno e l'indirizzo saranno attributi con cardinalità zero (avere attributi con cardinalità zero vuol dire semplicemente che quell'attributo nel modello relazionale potrà assumere il valore nullo).

Esempio: abbiamo un'associazione composizione di tipo molti a molti tra giocatore e squadra, e in corrispondenza dell'associazione due attributi la data di acquisto e la data di cessione tipo (0,1), perché non è detto che il giocatore sia stato venduto. Posso partizionare l'associazione facendo distinzione tra la composizione attuale e la composizione passata, quindi lascio le due entità però invece di mantenere un'unica associazione introduco due associazioni una che tenga conto della situazione attuale, quindi con la data di acquisto e una invece che tenga conto della situazione passata con la data di acquisto e la data di cessione.



Scelta degli identificatori principali

Operazione indispensabile per la traduzione nel modello relazionale, Perché gli identificatori dell'entità che sono coinvolte poi diventeranno le chiavi delle tabelle nel modello relazionale quindi ovviamente dobbiamo scegliere come identificatori, attributi o insieme di attributi, in cui non ci sia opzionalità quindi se ci sono attributi con cardinalità minima zero quindi sicuramente non possono essere considerati, anche semplici nel senso non conviene avere degli identificatori formati da tanti attributi perché poi quando si traducono nel modello logico in corrispondenza delle chiavi primarie vengono generate degli indici delle strutture dati che permettono di

accedere in maniera efficiente alle righe della tabella e tanto più complesso l'identificatore e tanto più complessa sarà la struttura dati. Criteri:

- assenza di opzionalità (vanno esclusi attributi con valori nulli);
- semplicità (questo garantisce che gli indici siano di dimensioni ridotte);
- utilizzo nelle operazioni più frequenti o importanti.

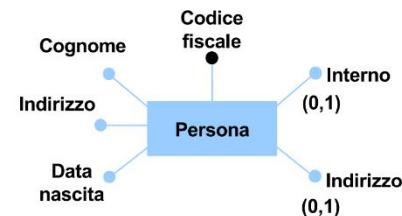
Se nessuno degli identificatori soddisfa questi requisiti, si introducono nuovi attributi (codici) contenenti valori speciali generati appositamente per questo scopo.

Traduzione verso il modello relazionale

Concluse tutte queste trasformazioni otteniamo uno schema ER ristrutturato che dobbiamo tradurre nel modello relazionale.

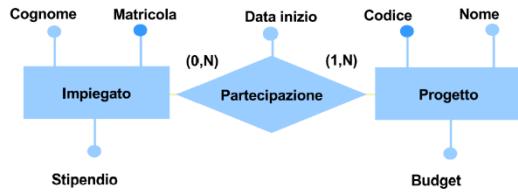
Idea di base:

- le entità diventano relazioni sugli stessi attributi;
- le relationship diventano relazioni sugli identificatori delle entità coinvolte (più gli attributi propri).



Utile se le operazioni su persona richiedono sempre anche i dati relativi all'appartamento.

Entità e relationship molti a molti



Impiegato(Matricola, Cognome, Stipendio)

Progetto(Codice, Nome, Budget)

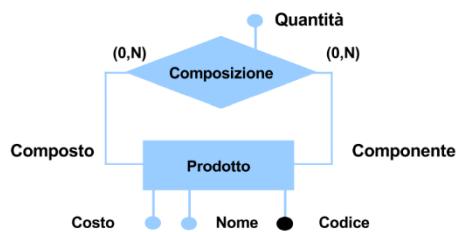
Partecipazione(Impiegato, Progetto, DataInizio)

Partecipazione.Impiegato → Impiegato.Matricola

Partecipazione.Progetto → Progetto.Codice

- La traduzione non riesce a tenere conto delle cardinalità minime delle relationship molti a molti.

e il codice del progetto e poi la data d'inizio che è l'attributo dell'associazione. Ci sono dei vincoli perché la matricola di partecipazione deve corrispondere a quella di un impiegato e il codice che compare in partecipazione deve essere il codice di un progetto. La traduzione non riesce a tener conto delle cardinalità minime delle associazioni a molti a molti, cioè il fatto che il progetto sia obbligato a partecipare mentre impiegato non lo sia in questa traduzione non compare, quindi dovrò eventualmente tenerne traccia in qualche altro modo.



Prodotto(Codice, Nome, Costo)

Composizione(Composto, Componente, Quantità)

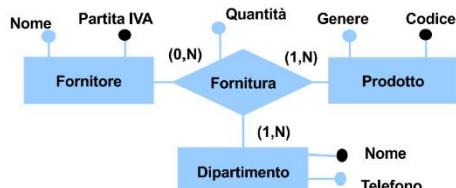
Composizione.Composto → Prodotto.Codice

Composizione.Componente → Prodotto.Codice

quindi composto e componente e poi la quantità.

Esempio di associazione molti a molti di tipo ricorsivo: ho il prodotto che è associato a se stesso tramite l'associazione composizione, quindi un'associazione ricorsiva non simmetrica in cui da una parte il prodotto che gioca il ruolo di composto e da una parte il prodotto che invece gioca il ruolo di componente. Traduzione: la strategia è la stessa di prima però in questo caso di entità c'era una sola l'entità ovvero prodotto che è associata con se stessa, quindi dovrò tradurre prodotto nella tabella con attributi codice, nome e costo con chiave codice e l'associazione composizione deve tener conto nel fatto che deve associare due prodotti. È necessario che io utilizzi due nomi diversi per indicare il ruolo diverso che i due prodotti hanno nell'associazione,

Relationship n-arie



Fornitore(PartitaIVA, Nome)

Prodotto(Codice, Genere)

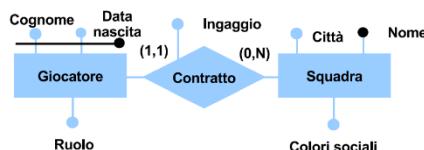
Dipartimento(Nome, Telefono)

Fornitura(Fornitore, Prodotto, Dipartimento, Quantità)

molti quella deve diventare una tabella.

Progettazione logica-20220406 0704-1

Esempio: abbiamo due entità giocatore e squadra. Giocatore ha come attributi il cognome, la data di nascita e il ruolo ed è identificato dalla coppia cognome e data di nascita. La squadra identificata da un nome e come attributo città e colori sociali. Queste due entità sono collegate da un'associazione contratto di tipo uno a molti con un attributo ingaggio, quindi un giocatore è associato ad una squadra tramite contratto e ad una singola squadra saranno associati diversi giocatori, eventualmente nessuno, perché qui è stata prevista una cardinalità zero.



Giocatore(Cognome, DataNascita, Ruolo)

Contratto(CognGiocatore, DataNascG, Squadra, Ingaggio)

Squadra(Nome, Città, ColoriSociali)

- Un giocatore ha un contratto con una sola squadra quindi Giocatore e Contratto hanno la stessa chiave ed è possibile fondere le due relazioni.

Giocatore(Cognome, DataNascita, Ruolo, Squadra, Ingaggio)

Squadra(Nome, Città, ColoriSociali)

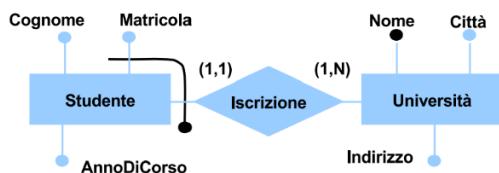
Giocatore.Squadra → Squadra.Nome

- Se la cardinalità minima della relationship è 0, allora Squadra in Giocatore deve ammettere valore nullo.
- La traduzione riesce a rappresentare efficacemente la cardinalità minima della partecipazione che ha 1 come cardinalità massima: 0 significa che il valore nullo è ammesso, 1 che valore nullo non è ammesso.

quindi un giocatore è associato ad una singola squadra, quindi vuol dire che in realtà le due tabelle giocatore e contratto hanno la stessa chiave nel senso che in contratto la terna cognome-giocatore e data di nascita e squadra in realtà è una super chiave, perché squadre ridondante dato che il giocatore appartiene sempre ad una squadra. La traduzione in questi casi può essere semplificata, quindi è inutile avere due tabelle che hanno la stessa chiave, è sufficiente averne una sola. In queste situazioni si traducono le due entità in tabelle ma l'associazione non viene tradotta in tabella ma viene portata dentro l'entità che partecipa all'associazione con cardinalità massima, quindi in questo caso l'associazione viene portata dentro all'entità giocatore.

Traduzione: il giocatore diventa una tabella che si porta dietro sia gli attributi che gli identificatori. Squadra lo stesso si porta dietro attributi e identificatori. In questa soluzione anche il contratto come una tabella identificata da due entità, quindi giocatore e squadra (attenzione quando fate queste traduzioni, in generale se una un'entità ha più di attributo che la identifica ve li dovete portare dietro, quindi in questo caso il giocatore si porta dietro sia il cognome che la data di nascita). Questa è un'associazione di tipo uno a molti

Entità con identificazione esterna



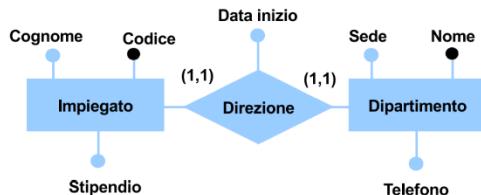
Studente(Matricola, Università, Cognome, AnnoDiCorso)

Università(Nome, Città, Indirizzo)

Studente.Università → Università.Nome

Esempio di associazione uno a molti in cui però una delle due entità è identificata esternamente dall'altra: abbiamo lo studente con cognome, anno di corso e matricola e poi l'università con nome, città e indirizzo. Ogni studente per essere identificato ha bisogno anche dell'università di riferimento. Traduzione: gli attributi della entità associata a quella che ha cardinalità massima 1 sono stati portati nella tabella corrispondente però in questo secondo caso l'identificatore esterno fa parte della chiave della tabella. Anche in questo caso non è necessario tradurre l'associazione iscrizione in una tabella.

Relationship uno a uno



Impiegato (Codice, Cognome, Stipendio, Dipartimento.Diretto, InizioDirezione)

Dipartimento (Nome, Sede, Telefono)

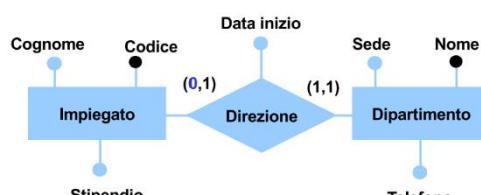
Impiegato(Codice, Cognome, Stipendio)

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

Esempio di associazioni uno a uno: abbiamo impiegato e dipartimento. Cognome, codice e stipendio gli attributi di impiegato, sede, nome e telefono quelli del dipartimento e poi la data di inizio come attributo dell'associazione direzione. Traduzione: siccome tutte e due l'entità partecipano con cardinalità massima 1, le traduzioni possibili sono due perché o si portano i riferimenti di dipartimento dentro impiegato oppure viceversa. Impiegato con attributi cognome codice stipendio più i riferimenti all'altra entità quindi il dipartimento diretto e l'inizio della direzione come attributo dell'associazione oppure viceversa quello che si può fare è definire la tabella impiegato semplicemente con gli attributi dell'entità e inserire invece in dipartimento il riferimento al direttore e all'inizio direzione dell'impiegato che corrisponde a direttore del dipartimento. Due

traduzioni sono equivalenti quindi non ce n'è una da preferire rispetto all'altra. Ci possono essere delle scelte da preferire se le cardinalità minime delle due entità sono zero o almeno uno e zero o tutte e due sono zero.

Una possibilità privilegiata, senza valori nulli



Impiegato(Codice, Cognome, Stipendio)

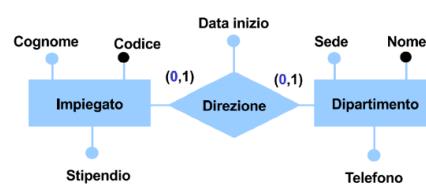
Dipartimento(Nome, Sede, Telefono, Direttore, InizioDirezione)

Dipartimento.Direttore → Impiegato.Codice

inizio. I vincoli devono essere sempre aggiunti per tener conto del fatto ci

Esempio: impiegato partecipi con cardinalità minima zero. La traduzione da fare è quella in cui si va a inserire l'associazione dentro il dipartimento, quindi dentro l'entità che partecipa con cardinalità minima e massima 1, quindi in questo caso nella tabella dipartimento ho inserito il riferimento all'impiegato che ha ruolo direttore e l'attributo data di

Un altro caso



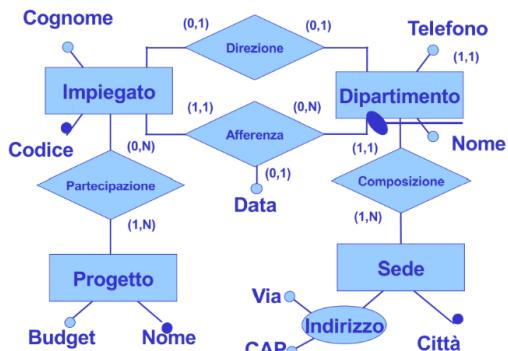
Impiegato(Codice, Cognome, Stipendio)

Dipartimento(Nome, Sede, Telefono)

Direzione(Direttore, Dipartimento, InizioDirezione)

aspettiamo che il direttore sia uno degli impiegati, quindi se una delle due cardinalità minime è zero allora l'associazione viene portata dentro quella che partecipa con cardinalità minima 1.

Esempio se sono tutte e due 0: abbiamo un'associazione a uno a uno in cui tutte e due le entità partecipano in modo non obbligato. le traduzioni possibili in questo caso sono tre perché si può portare l'associazione dentro impiegato o l'associazione dentro dipartimento ma in questo caso può valer la pena tradurre l'associazione di nuovo in tabella, quindi introdurre di nuovo la tabella direzione (non importa portarsi dietro le chiavi di tutte e due le entità perché tanto sono in corrispondenza uno a uno, quindi basta ad esempio il direttore).



Impiegato(Codice, Cognome, Dipartimento, Sede, Data*)

Dipartimento(Nome, Città, Telefono, Direttore*)

Sede(Città, Via, CAP)

Progetto(Nome, Budget)

Partecipazione(Impiegato, Progetto)

Impiegato.Dipartimento → Dipartimento.Nome

Impiegato.Sede → Sede.Città

Dipartimento.Direttore → Impiegato.Codice

Dipartimento.Città → Sede.Città

Partecipazione.Impiegato → Impiegato.Codice

Partecipazione.Progetto → Progetto.Nome

Un * indica la possibilità di avere valori nulli.

Impiegato(Codice, Cognome, Dipartimento, Data*)

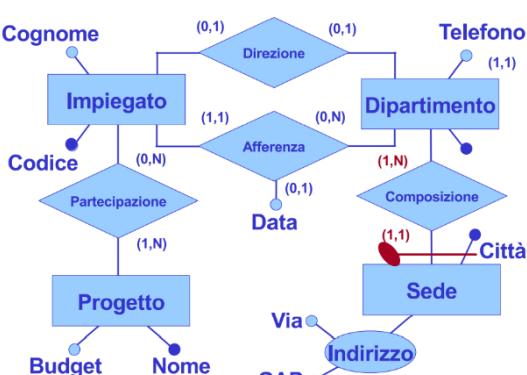
Dipartimento(Nome, Telefono, Direttore*)

Sede(Città, Dipartimento, Via, CAP)

Progetto(Nome, Budget)

Partecipazione(Impiegato, Progetto)

Un Dipartimento può essere strutturato in più sedi.



Impiegato(Codice, Cognome, Dipartimento, Data*)

Dipartimento(Nome, Telefono, Direttore*)

Sede(Città, Dipartimento, Via, CAP)

Progetto(Nome, Budget)

Partecipazione(Impiegato, Progetto)

Un Dipartimento può essere strutturato in più sedi.

Esempio: abbiamo uno schema ER da tradurre, la prima cosa che vi consiglio di fare è un conto delle tabelle ci dobbiamo aspettare sulla base delle considerazioni che abbiamo fatto. Ci deve essere sicuramente una tabella per ogni entità coinvolta, poi se ci sono delle associazioni molti a molti quelle diventano sicuramente tabelle. Se ci sono invece le associazioni uno a molti quelle devono essere inglobate in una delle due entità. Sulle associazioni uno o uno dipende dalle cardinalità minime. Abbiamo quattro entità quindi sicuramente quattro tabelle, poi abbiamo un'associazione molti a molti tra impiegato e progetto quindi una quinta tabella, e queste due associazioni uno a molti invece le dovremo inserire

dentro le tabelle corrispondenti e poi abbiamo questa associazione direzione di tipo uno a uno con cardinalità minima zero, qui possiamo decidere una delle tre strategie. Sono sicuramente 5 tabelle e potrebbero essere 6 se direzione decidiamo di tradurla come tabella (quindi se sono di meno o di più vuol dire che c'è qualcosa che state sbagliando). Impiegato si porterà dietro il cognome e il codice e poi si deve portare dietro eventuali associazioni uno a molti se partecipa con cardinalità massima 1. Impiegato partecipa all'associazione afferenza con cardinalità massima 1, quindi vuol dire che a impiegato dovremo associare l'identificatore del dipartimento e poi l'attributo data. (attenzione perché non è detto che l'identificatore dell'entità sia semplice, in questo caso l'entità dipartimento è identificata dal suo nome ma esternamente anche dalla sede, quindi vuol dire che quando faccio questa traduzione nell'impiegato mi dovrò portare dentro il nome di dipartimento e la sede della città e la data di afferenza, la data è un attributo con cardinalità (0,1) quindi è opzionale ed è stato inserito nella traduzione una* ad indicare che quell'attributo potrà assumere valori nulli). Dipartimento partecipa a questa associazione uno a molti ma dipartimento è identificato da sede, quindi si porterà dietro sicuramente la città e la coppia nome-città forniranno l'identificatore di dipartimento, poi abbiamo il telefono, e può valer la pena portarsi dietro l'associazione direzione, quindi ho inserito un attributo direttore con l'asterisco perché potrebbero esserci dei dipartimenti che non hanno direttore. Sede si porta dietro soltanto gli attributi dell'entità perché partecipa soltanto all'associazione una molti ma con cardinalità massima N. La tabella partecipazione che è quella che traduce l'associazione molti a molti e fa riferimento agli identificatori dell'entità coinvolte, quindi al codice dell'impiegato e al nome del progetto e ovviamente la coppia di questi due attributi formerà la chiave dell'associazione.

Lo stesso schema in cui però è stata fatta una piccola variazione, questa volta è la sede l'entità ad essere identificata dalla sua città e dal dipartimento corrispondente. Cambia anche la traduzione perché abbiamo sempre 5 tabelle però in questo caso è la sede che deve includere l'associazione e quindi che in sede è stata inserita la città e il dipartimento che formeranno la chiave. Nel dipartimento ovviamente non ci sono più gli attributi che c'erano prima che corrispondevano all'associazione, c'è invece sempre riferimento al

direttore perché anche in questa versione l'associazione direzione è stata portata dentro direttore.

Traduzione di associazioni molti a molti

Tipologia	Concetto iniziale	Risultati possibili
Associazione binaria molti a molti		$E_1(A_{E11}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$
Associazione ternaria molti a molti		$E_1(A_{E11}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $E_3(\underline{A_{E31}}, A_{E32})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, \underline{A_{E31}}, A_R)$

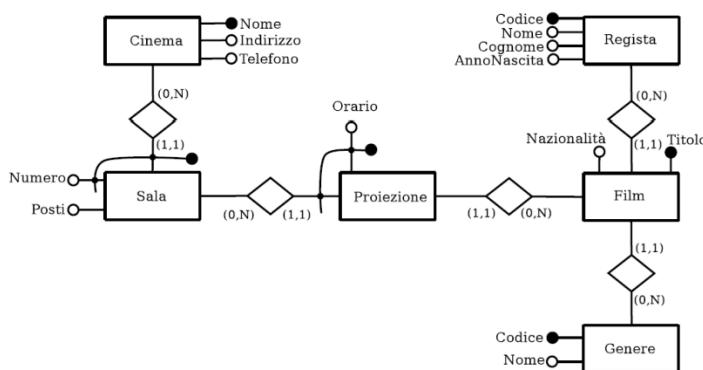
Traduzione di associazioni uno a molti

Tipologia	Concetto iniziale	Risultati possibili
Associazione uno a molti con partecipazione obbligatoria		$E_1(A_{E11}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$
Associazione uno a molti con partecipazione opzionale		$E_1(A_{E11}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(A_{E11}, A_{E21}, A_R)$ Oppure: $E_1(A_{E11}, A_{E12}, A_{E21}, A_R^*)$ $E_2(\underline{A_{E21}}, A_{E22})$
Associazione con identificatore esterno		$E_1(A_{E11}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

Traduzione di associazioni uno a uno

Tipologia	Concetto iniziale	Risultati possibili
Associazione uno a uno con partecipazione obbligatoria per entrambe le entità		$E_1(A_{E11}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ Oppure: $E_2(A_{E21}, A_{E22}, A_{E11}, A_R)$ $E_1(\underline{A_{E11}}, A_{E12})$
Associazione uno a uno con partecipazione opzionale per una entità		$E_1(A_{E11}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$
Associazione uno a uno con partecipazione opzionale per entrambe le entità		$E_1(A_{E11}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ Oppure: $E_1(A_{E11}, A_{E12}, A_{E21}, A_R^*)$ $E_2(\underline{A_{E21}}, A_{E22})$ Oppure: $E_1(A_{E11}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(A_{E11}, A_{E21}, A_R)$

Sale cinematografiche di una città



Cinema(Nome, Indirizzo, Telefono)

Sala(NomeC, Numero, Posti)

Proiezione(Cinema, NSala, Orario, TitoloFilm)

Film(Titolo, Nazionalita, Regista, Genere)

Regista(Codice, Nome, Cognome, AnnoNascita)

Genere(Codice, Nome)

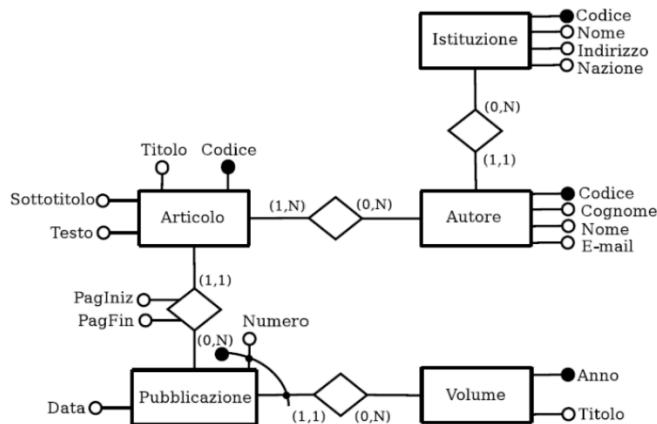
Sala.NomeC → Cinema.Nome

Proiezione.Cinema → Cinema.Nome

Proiezione.NSala → Sala.Numero

Film.Regista → Regista.Codice

Film.Genere → Genere.Codice



Articolo(Codice, Titolo, Sottotitolo, Testo, Pagl, PagF, Numero, Anno)

Pubblicazione(Numero, Anno, Data)

Volume(Anno, Titolo)

Autore(Codice, Cognome, Nome, Email, CodIstituzione)

ArticoloAutore(CodArticolo, CodAutore)

Istituzione(Codice, Nome, Indirizzo, Nazione)

(Articolo.Numero, Articolo.Anno) → (Pubblicazione.Numero, Pubblicazione.Anno)

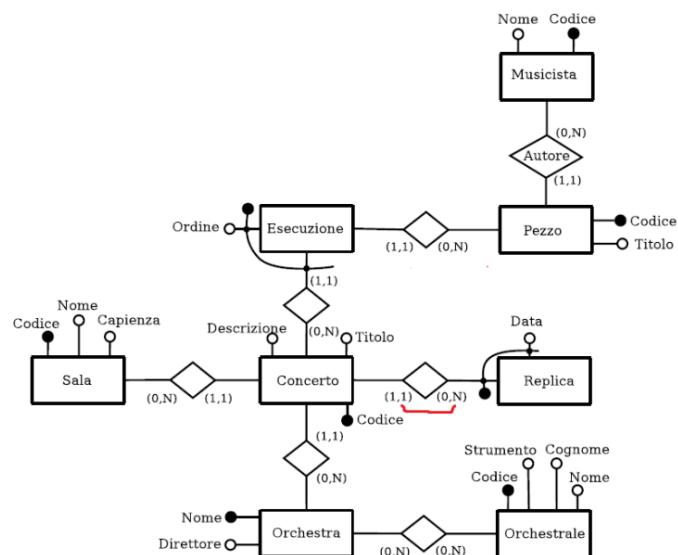
Pubblicazione.Anno → Volume.Anno

Autore.CodIstituzione → Istituzione.Codice

ArticoloAutore.CodArticolo → Articolo.Codice

ArticoloAutore.CodAutore → Autore.Codice

Concerti



Concerto(Codice, Titolo, Descrizione, NomeO, CodiceS)

Replica(CodiceC, Data)

Sala(Codice, Nome, Capienza)

Orchestra(Nome, Direttore)

Orchestrale(Codice, Nome, Cognome, Strumento)

OO(Orchestra, Orchestrale)

Esecuzione(CodiceC, Ordine, CodiceP)

Pezzo(Codice, Titolo, Musicista)

Musicista(Codice, Nome)

Introduzione sulla normalizzazione

Parleremo di:

- Ridondanze e anomalie
- Dipendenze funzionali, implicazione, regole di inferenza
- Chiusura di un insieme di attributi
- Chiave di una relazione
- Coperture, coperture ridotte

- Decomposizioni
- Decomposizioni senza perdita
- Conservazione delle dipendenze
- Forma di Boyce Codd
- Terza forma normale

ORDINE(num, fornitore, indirizzo, articolo, data, quantita)

num	fornitore	indirizzo	articolo	data	quantita
350	Rossi	Prato	penna	10/2/13	50
350	Rossi	Prato	lapis	10/2/13	40
350	Rossi	Prato	gomma	10/2/13	45
351	Bianchi	Empoli	pennarelli	10/2/13	60
360	Verdi	Empoli	quaderni	13/2/13	100
360	Verdi	Empoli	penna	13/2/13	70
362	Rossi	Prato	penna	13/2/13	10

viene inserito nuovamente il riferimento al suo indirizzo.

Un esempio: abbiamo una tabella ordine che contiene diversi attributi. Si immagina che in corrispondenza dello stesso numero ordine ci sia riferimento ad articoli diversi che sono presenti in quell'ordine, quindi ad esempio per l'ordine 350 sono stati acquistati 50 penne, 40 lapis e 45 gomme. Si osserva che per come è organizzata la tabella ci sono delle ridondanze: l'indirizzo del fornitore viene ripetuto in tutte le tuple che fanno riferimento alla stessa ordine, ma anche il fatto che se quel fornitore compare in un altro record

- ridondanza: il valore di indirizzo è ripetuto in tutte le tuple che riguardano un ordine e in tutti gli ordini dello stesso fornitore. La presenza di ridondanza crea tutta una serie di anomalie come:
 - anomalie di aggiornamento: se modifichiamo l'indirizzo di un fornitore o la data di un ordine in una tupla, dobbiamo modificare contemporaneamente anche le altre. perché se ci accorgiamo di aver sbagliato ad inserire qualche informazione ad esempio che il fornitore non si chiama Rossi ma si chiama Rosa dovrà accedere a tutte le tuple della tabella e modificare il nome del fornitore.
 - anomalie di inserzione: non possiamo inserire le caratteristiche di un nuovo fornitore senza che gli sia stato ordinato qualcosa, cioè l'aver mescolato ordini, fornitori e articoli tutto insieme fa sì che se io voglio aggiungere le indicazioni relative a un fornitore non lo posso fare fintanto che non viene fatto un ordine a quel fornitore.
 - anomalia di cancellazione: se cancelliamo le informazioni relative all'ordine 351, cancelliamo anche le informazioni che riguardano il fornitore Bianchi.

FREQ(cod_stud, nome, indirizzo, cod_corso, nome_corso, tipo, periodo)

- ogni studente può frequentare più di un corso
- ogni corso può essere frequentato da più studenti
- sono presenti ridondanze e anomalie
 - tutte le caratteristiche di uno studente vengono ripetute per ogni corso che frequenta;
 - tutte le caratteristiche di un corso vengono ripetute per ogni studente che lo frequenta;
- sono state raggruppate informazioni eterogenee.

studente ha frequentato e d'altra parte ogni corso potrà essere frequentato da più studenti per cui le informazioni di un corso saranno ripetute.

Dipendenze funzionali

Dipendenza funzionale: vincolo di integrità che lega fra loro i valori degli attributi di una relazione

- Es: in una istanza di ORDINE, se abbiamo due valori uguali per l'attributo fornitore, quelle due tuple hanno valori uguali anche per l'attributo indirizzo;
- Es: in una istanza di FREQ, se abbiamo due valori uguali per l'attributo cod_stud, quelle due tuple hanno valori uguali anche per l'attributo nome.

Ad uno schema di relazione sono in genere associate più dipendenze funzionali. Questi tipi di vincoli possono essere individuati solo considerando il significato degli attributi.

ORDINE(num, fornitore, indirizzo, articolo, data, quantità)

- un certo ordine si riferisce sempre allo stesso fornitore:
 $num \rightarrow fornitore$
- un fornitore ha sempre lo stesso indirizzo:
 $fornitore \rightarrow indirizzo$
- un ordine viene inviato in una certa data: $num \rightarrow data$
- se ci sono due righe con lo stesso numero d'ordine, allora sono uguali anche i valori di fornitore, indirizzo e data:
 $num \rightarrow fornitore, indirizzo, data$
- in un certo ordine, un certo articolo viene ordinato in una determinata quantità: $num, articolo \rightarrow quantità$
- $indirizzo \rightarrow indirizzo$ dipendenza banale, identità.

uguale y, quindi la dipendenza x determina x è presente in qualsiasi schema di relazione. Le identità sono presenti in qualsiasi schema relazionale. La presenza di dipendenze in qualche modo deve essere gestita perché la presenza di questo concetto che poi crea ridondanza. Attenzione non è che le dipendenze si possono eliminare, non si devono eliminare perché sono dei vincoli che esistono, quindi non vogliamo eliminarle, vogliamo gestirle in modo da che non diano fastidio. Una dipendenza che non crea ridondanza perché avendo x una chiave non potranno mai esserci due tuple che hanno gli stessi valori su x, quindi l'obiettivo è quello di avere delle relazioni che contengano dipendenze funzionali che hanno a sinistra una chiave

Esempio: abbiamo una tabella di frequenza in cui gli attributi sono il codice dello studente, il nome, l'indirizzo, il codice del corso, il nome del corso, il tipo e il periodo. Vogliamo rappresentare studenti e i corsi che hanno seguito però anche questa tabella presenta delle ridondanze, perché ogni studente può frequentare più di un corso, quindi vuol dire che il riferimento a uno studente, al nome e all'indirizzo sarà ripetuto tutte le volte che devo indicare un corso diverso che quello

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

$$X \subseteq \{A_1, A_2, \dots, A_n\}, Y \subseteq \{A_1, A_2, \dots, A_n\}$$

si dice che **X determina funzionalmente Y** o che Y dipende funzionalmente da X e si scrive $X \rightarrow Y$ se \forall istanza r di R , \forall coppia di tuple t_1 e t_2 in r

$$t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$$

In altre parole, in ogni istanza non possono esserci due tuple con valori uguali per X e valori diversi per Y.

Se $Y \subseteq X$ la dipendenza si dice **banale** o **ovvia** in particolare se $X = Y$ la dipendenza (banale) si chiama anche **identità**.

Esempio: se io prendo

due qualsiasi tuple che hanno gli stessi valori di fornitore, queste due tuple devono avere uguale lo stesso valore di indirizzo quindi perché ci sia una dipendenza funzionale fra insieme di attributi bisogna che tra questi succeda che se io prendo due record che hanno gli stessi valori sulla parte sinistra della dipendenza questi due record devono avere gli stessi valori anche sulla parte destra della dipendenza quindi se hanno valori uguali su x devono avere valori uguali anche sui y.

Ci sono dei casi particolari: se y è un sottoinsieme di x la dipendenza si dice banale o ovvia, queste sono le dipendenze banali che sono sempre presenti nella base di dati. Il caso particolare più banale è quello in cui x

perché la presenza di quelle dipendenze non genera ridondanza. Esempio: abbiamo una tabella impiegato e abbiamo il codice dell'impiegato, il nome dell'impiegato, lo stipendio, il progetto e la data finale. In questo schema ci sono delle dipendenze: il codice dell'impiegato determina il nome degli impiegati, il codice dell'impiegato determina lo stipendio e quindi posso anche

IMPIEGATO(codimp, nome, stipendio, progetto, data_finale)

- $\text{codimp} \rightarrow \text{nome}$
- $\text{codimp} \rightarrow \text{stipendio}$
- $\text{codimp} \rightarrow \text{nome, stipendio}$
- $\text{codimp, nome} \rightarrow \text{stipendio}$
- $\text{progetto} \rightarrow \text{data_finale}$
- $\text{codimp, progetto} \rightarrow \text{codimp, data_finale}$
- $\text{stipendio} \rightarrow \text{stipendio}$ dipendenza ovvia, identità
- $\text{nome, stipendio} \rightarrow \text{stipendio}$ dipendenza ovvia

Se un impiegato può partecipare ad un solo progetto, allora, ad esempio: $\text{codimp} \rightarrow \text{progetto}$ e $\text{codimp} \rightarrow \text{data_finale}$

dire che il codice dell'impiegato determina sia il nome che lo stipendio oppure un'altra dipendenza che esiste in questo schema è che il codice dell'impiegato e il nome dell'impiegato determinano lo stipendio. Si capisce che c'è un problema quando io vado a definire le dipendenze perché il rischio è quello di ripetere più volte la stessa informazione, cioè definire delle dipendenze significa stabilire dei vincoli tra gli attributi e rischio di definire questi vincoli più del necessario. Dipendenza tra progetto e data finale anche questa è ragionevole che sia presente nello schema e altre dipendenze che possono essere definite sono ad esempio codice dell'impiegato e il progetto determinano il codice dell'impiegato e la data finale. Sto creando delle dipendenze ridondanti perché avevo già detto prima che il progetto determina la data finale.

Questa nuova dipendenza ha sia a sinistra che a destra un attributo codice impiegato e anche questa è una dipendenza corretta perché se io vado a considerare il tuple che hanno lo stesso codice impiegato e lo stesso progetto di sicuro queste due avranno lo stesso codice impiegato e la stessa data finale. L'identità tra stipendio e stipendio è una dipendenza ovvia perché a destra ho un attributo che è un sottoinsieme degli attributi che si trovano a sinistra.

Alcune osservazioni

- Una dipendenza funzionale è un vincolo di integrità associato ad una relazione.
- Ad ogni schema viene associato un insieme di dipendenze funzionali.
- Ad ogni schema è sempre associato l'insieme delle dipendenze ovvie (anche se in genere si tralasciano perché prive di interesse pratico).
- L'insieme delle dip. funzionali può essere determinato solo conoscendo il significato degli attributi nel contesto che stiamo rappresentando (vedi $\text{codimp} \rightarrow \text{progetto}$ in IMPIEGATO).
- Uno schema di una relazione viene associato ad un insieme di dipendenze funzionali dal progettista della base di dati.
- Una particolare tabella che verifica l'insieme dei vincoli associati allo schema (e quindi anche le dip. funz. associate) si dice corretta.

Dipendenze funzionali: implicazione

Dato un insieme di dipendenze funzionali F

$$F \text{ implica } X \rightarrow Y$$

se ogni relazione r che soddisfa F soddisfa anche $X \rightarrow Y$.

L'insieme F^+ delle dipendenze implicate da F viene detto **chiusura di F**

$$F^+ = \{X \rightarrow Y \mid F \text{ implica } X \rightarrow Y\}, \quad F \subseteq F^+$$

Per gestire le dipendenze funzionali devo introdurre dei concetti: il concetto di implicazione, cioè abbiamo visto negli esempi precedenti che le stesse relazioni tra attributi le posso definire tramite diverse dipendenze. Immaginiamo di avere è una tabella con un insieme di dipendenze funzionali e a partire da queste ne possono individuare anche tramite il concetto di implicazione. L'insieme di tutte le dipendenze che sono implicate da F si chiama la chiusura di F . La chiusura di F è l'insieme di tutte le dipendenze funzionali che riesco a determinare a partire da un insieme, quindi

Esempi di implicazione di dipendenze

- Se $\{A \rightarrow B, B \rightarrow C\} \subseteq F$ allora F implica $A \rightarrow C$
Si può dimostrare: sia r una relazione che soddisfa F , siano t_1 e t_2 due tuple di r tali che $t_1[A] = t_2[A]$, poiché r soddisfa $A \rightarrow B$ si ha $t_1[B] = t_2[B]$, allora poiché r soddisfa $B \rightarrow C$ si ha $t_1[C] = t_2[C]$, quindi r soddisfa anche $A \rightarrow C$.
- **IMPIEGATO(codimp, nome, stipendio, progetto, data_finale)**
 $F = \{\text{codimp} \rightarrow \text{nome}, \text{codimp} \rightarrow \text{stipendio}, \text{codimp} \rightarrow \text{progetto}, \text{progetto} \rightarrow \text{data_finale}\}$

$$F \text{ implica codimp} \rightarrow \text{data_finale}$$

- **ORDINE(num, fornitore, indirizzo, articolo, data, quantita)**
 $F \supseteq \{\text{num} \rightarrow \text{fornitore}, \text{fornitore} \rightarrow \text{indirizzo}\}$

$$F \text{ implica num} \rightarrow \text{indirizzo}$$

formalmente la chiusura la indico con F^+ ed è costituita da un insieme di dipendenze funzionali quindi del tipo x determina y , cioè l'insieme da cui parte è contenuto in F^+ . Esempi che fanno riferimento alle tabelle, quella di impiegato e quella di ordine in cui andiamo ad applicare l'implicazione, quindi nella tabella impiegato con attributi codice-impiegato, nome, stipendio, progetto e data finale se consideriamo le dipendenze allora il codice dell'impiegato determina il nome, il codice dell'impiegato determina lo stipendio, il codice dell'impiegato determina il progetto e il progetto determina la data finale. Quale altra dipendenza può essere implicata? Abbiamo codice impiegato determina progetto e progetto determina la data finale quindi abbiamo due dipendenze in cui una ha una parte destra uguale alla parte sinistra dell'altra e quindi la dipendenza codice-impiegato determina data finale è implicata dall'insieme f . Nel caso della tabella ordine non sono state ricordate tutte le dipendenze ma soltanto due, il codice dell'ordine determina il fornitore e il fornitore determina l'indirizzo, quindi posso dire che il codice dell'ordine determina l'indirizzo.

Chiusura di F : Assiomi di Armstrong

Dato F , la chiusura F^+ può essere calcolata applicando ripetutamente a F le tre regole di inferenza dette **Assiomi di Armstrong**:

- **Riflessività** (stabilisce le dipendenze banali):

$$\text{se } Y \subseteq X \text{ allora } X \rightarrow Y$$

- **Arricchimento**:

$$\text{se } X \rightarrow Y \text{ allora } XZ \rightarrow YZ$$

- **Transitività**:

$$\text{se } X \rightarrow Y \text{ e } Y \rightarrow Z \text{ allora } X \rightarrow Z$$

Esempio di applicazione delle regole di Armstrong

IMPIEGATO(codimp, nome, stip, progetto, data_finale)

- **Riflessività**: $\{stip, progetto\} \subseteq \{stip, progetto\}$ quindi

$$stip, progetto \rightarrow stip, progetto$$

$$\{progetto\} \subseteq \{stip, progetto\} \text{ quindi } stip, progetto \rightarrow progetto$$

- **Arricchimento**: se vale $progetto \rightarrow data_finale$ allora

$$progetto, stip \rightarrow data_finale, stip$$

- **Transitività**: se $codimp \rightarrow progetto$ e $progetto \rightarrow data_finale$ allora $codimp \rightarrow data_finale$.

Dipendenze funzionali: unione e decomposizione

Dagli assiomi di Armstrong possono essere derivate altre regole di inferenza:

- **Unione**: se $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_k$ allora $X \rightarrow A_1A_2\dots A_k$

- **Decomposizione**: se $X \rightarrow A_1A_2\dots A_k$ allora $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_k$

Esempio di unione: se $codimp \rightarrow nome$ e $codimp \rightarrow stipendio$ allora $codimp \rightarrow nome, stipendio$

Esempio di decomposizione: se $codimp \rightarrow nome, stipendio$ allora $codimp \rightarrow nome$ e $codimp \rightarrow stipendio$.

Unione e decomposizione nel caso $k = 2$

- **Unione**: se $X \rightarrow A_1, X \rightarrow A_2$ allora $X \rightarrow A_1A_2$

- ① da $X \rightarrow A_1$ si ha $XA_2 \rightarrow A_1A_2$ per arricchimento
- ② da $X \rightarrow A_2$ si ha $XX \rightarrow XA_2 = X \rightarrow XA_2$ per arricchimento
- ③ da 2) e 1) si ha $X \rightarrow A_1A_2$ per transitività

- **Decomposizione**: se $X \rightarrow A_1A_2$ allora $X \rightarrow A_1, X \rightarrow A_2$

- ① si ha $A_1A_2 \rightarrow A_1$ per riflessività
- ② da $X \rightarrow A_1A_2$ e 1) si ha $X \rightarrow A_1$ per transitività
- ③ analogamente per $X \rightarrow A_2$

transitività. X determina XA_2 e XA_2 determina A_1A_2 .

L'obiettivo è trovare la chiusura di f quindi l'insieme di tutte le dipendenze funzionali che possono essere implicate. La chiusura può essere calcolata applicando in maniera ripetuta queste tre regole, dette le regole di inferenza di Armstrong o assiomi di Armstrong. Riflessività: è la regola che stabilisce le dipendenze banali e cosa ci dice questa regola che se ho un sottoinsieme y contenuto in un insieme x sicuramente x determina y , quindi tutte quelle in cui la parte destra della dipendenza è contenuta nella parte sinistra. Arricchimento: se io ho una dipendenza x determina y e in pratica quello che posso fare è andare ad aggiungere a destra e a sinistra della dipendenza un qualsiasi insieme Z quindi se vale x determina y allora sicuramente vale la dipendenza XZ determina YZ . Transitività: se la dipendenza x determina y che a sua volta determina z allora posso concludere che x determina z . Se io parto da un insieme F di dipendenze funzionali che ho individuato nel mio schema e applico ripetutamente questi assiomi determino la chiusura di F . Esempi di applicazioni delle regole: abbiamo la relazione impiegato definita sul codice impiegato, nome, stipendio, progetto e data finale e quindi dato che stipendio e progetti sono contenute in stipendi e progetto vale la dipendenza ovvia. Arricchimento quindi se ho la dipendenza tra progetto e data finale posso aggiungere a destra e a sinistra lo stesso insieme di attributi e la dipendenza continua a valere, quindi il progetto e lo stipendio determinano la data finale e lo stipendio. La transitività, quindi il codice dell'impiegato determina il progetto e il progetto determina la data finale, quindi il codice impiegato determina la data finale. Posso utilizzare anche altre due regole che possono essere derivate da questi assiomi che sono l'unione e la decomposizione. L'unione: se ci sono tante dipendenze funzionali che hanno a sinistra la stessa parte allora posso unire le parti destre di dipendenze che hanno la stessa parte sinistra e poi posso fare anche l'operazione inversa che è quella di decomposizione: se io ho una dipendenza che ha destra più di un attributo, quindi un insieme di attributi allora posso decomporre questa dipendenza. Queste regole possono essere dimostrate in maniera formale utilizzando il concetto di dipendenza funzionale, immaginiamo quindi di avere due dipendenze x determina con A_1 e x determina A_2 , quello che voglio dimostrare è che vale l'unione e parto dalla prima dipendenza, so che x determina A_1 e per la proprietà di arricchimento posso arricchire la mia dipendenza a destra e a sinistra con uno stesso insieme di attributi, quindi arricchisco con A_2 , quindi se vale x determina con A_1 allora posso dire che XA_2 determina A_1A_2 d'altra parte. Siccome x determina A_2 anche in questo caso posso fare un arricchimento aggiungendo questa volta sia a destra che a sinistra x , ma siccome a sinistra facendo x unione x in pratica la dipendenza XX determina XA_2 è equivalente a X determina XA_2 . Dalla prima ho trovato che XA_2 determina A_1A_2 , dalla seconda ho trovato che X determina XA_2 e quindi posso dire che X determina A_1A_2 per transitività.

Il problema dell'implicazione

- Un problema che si presenta spesso è quello di decidere se una dipendenza funzionale appartiene a F^+
- La sua risoluzione con l'algoritmo che consiste nell'applicare ripetutamente ed esaustivamente gli assiomi di Armstrong ha una complessità esponenziale rispetto al numero di attributi dello schema $R(T)$.
- Se n è la dimensione di T , F^+ contiene almeno le $2^n - 1$ dipendenze funzionali banali $T \rightarrow X$ dove X è un sottoinsieme non vuoto di T .
- Un metodo di complessità minore (polinomiale rispetto alle dimensioni di T e F) si basa sulla seguente considerazione: per decidere se $X \rightarrow Y \in F^+$ si può controllare se $Y \subseteq X_F^+$

Si passerà dal concetto di chiusura di un insieme di dipendenze al concetto di chiusura di un insieme di attributi. Il simbolo che utilizzeremo X_F^+ è il simbolo che utilizzeremo per indicare la chiusura di un insieme di attributi X rispetto ad un insieme di dipendenze funzionali. Per risolvere il problema dell'implicazione sarà sufficiente vedere se la parte destra della dipendenza quindi y è contenuta in questo insieme di attributi. L'obiettivo è realizzare uno schema di qualità e quindi possono essere presenti soltanto dipendenze che non generano ridondanza.

Normalizzazione-20220426 0705-1

Chiusura di un insieme di attributi

Dati $R(T)$ e F sia $X \subseteq T$

$$X_F^+ = \{A \in T \mid X \rightarrow A \in F^+\} \text{ ovvero}$$

$$X_F^+ = \{A \in T \mid F \text{ implica } X \rightarrow A\}$$

la chiusura di X rispetto a F è l'insieme degli attributi che dipendono da X (direttamente o implicitamente)

Teorema:

$$F \text{ implica } X \rightarrow Y \text{ se e solo se } Y \subseteq X_F^+$$

Questo teorema ci dà un metodo per verificare se una dipendenza è implicata da F (appartiene a F^+)

$$X \subseteq X_F^+$$

$$F \text{ implica } X \rightarrow Y \Leftrightarrow Y \subseteq X_F^+$$

• \Rightarrow

- ① sia $Y = A_1 A_2 \dots A_k$ quindi $X \rightarrow A_1 A_2 \dots A_k$
- ② per la regola di decomposizione si ha $X \rightarrow A_i$, quindi $A_i \in X_F^+$ per definizione di X_F^+ , $1 \leq i \leq k$
- ③ pertanto $Y \subseteq X_F^+$

Calcolo di X_F^+

- Input: $R(T)$, F , $X \subseteq T$
- Output: X_F^+ chiusura di X rispetto a F
- Metodo: si calcola una sequenza di insiemi X^0, X^1, \dots con i passi
 - ① $X^0 \leftarrow X$
 - ② $X^{i+1} \leftarrow X^i \cup \{A \mid Y \rightarrow Z \in F, Y \subseteq X^i, A \in Z\}$
- $X = X^0 \subseteq X^1 \dots \subseteq X^i \dots \subseteq T$ e T è finito (l'algoritmo termina)
- se $X^i = X^{i+1}$ allora $X^i = X^{i+1} = X^{i+2} = \dots$
- si può dimostrare che in tal caso $X^i = X_F^+$

Aggiungo gli attributi di una dipendenza alla volta.

sequenza di insiemi quindi tutte le volte un insieme un po' più grande del precedente. Al massimo aggiungerò tutti gli attributi di t , quindi questo mi garantisce che l'algoritmo termini. Mi fermo quando mi accorgo che un certo passo l'insieme ottenuto non è cambiato rispetto al passo precedente. A quel punto l'algoritmo si arresta e l'ultimo insieme che ho determinato rappresenta quella chiusura di x .

Per risolvere il problema in maniera più semplice dal punto di vista computazionale si introduce il concetto di chiusura di un insieme di attributi. Che cos'è un la chiusura di un insieme di attributi? Abbiamo una relazione definita sull'insieme t degli attributi e l'insieme f delle dipendenze ad essa associate e consideriamo un sottoinsieme x degli attributi, allora definisco la chiusura di x rispetto all'insieme f delle dipendenze funzionali costituita da tutti gli attributi che appartengono all'insieme t , quindi all'insieme che definisce la mia relazione, tali che esiste una dipendenza da x , l'insieme che sto considerando, mi permette di determinare A , ovvero x determina A appartiene ad F^+ , quindi è possibile determinare l'attributo A partendo da x tramite le dipendenze che sono in F . Teorema: Avendo una relazione e un insieme F di attributi allora posso dire che F implica la dipendenza x determina y se e solo se y , quindi la parte destra della dipendenza, è contenuta in X_F^+ . Vuol dire che per determinare se una certa dipendenza funzionale è implicata da un insieme F di dipendenze io posso verificare che la parte destra della dipendenza è contenuta nella chiusura di x rispetto ad f .

Per calcolare l'insieme x esistono degli algoritmi: abbiamo in input una relazione e un insieme f di dipendenze funzionali e un sottoinsieme x dell'insieme t , voglio trovare questo insieme X_F^+ . In questa prima versione dell'algoritmo per il calcolo della chiusura di x vengono aggiunti con un procedimento iterativo gli attributi di una dipendenza alla volta, si parte da un insieme X^0 che equivale ad X , quindi l'insieme di cui voglio calcolare la chiusura, e poi ad ogni step genero un nuovo insieme X^{i+1} , dato dall'unione dell'insieme che avevo generato al passo precedente e vado ad aggiungere a questo insieme gli attributi che hanno questa caratteristica, cioè deve esistere una dipendenza y determina Z in f , in cui y è costituito da attributi che trovo nell'insieme X^i che ho trovato il passo precedente e A è l'attributo nuovo che vado ad aggiungere e che deve essere un attributo della parte destra della dipendenza (un attributo nuovo che non ho già aggiunto in precedenza). Genero una sequenza di insiemi quindi tutte le volte un insieme un po' più grande del precedente. Al massimo aggiungerò tutti gli attributi di t , quindi questo mi garantisce che l'algoritmo termini. Mi fermo quando mi accorgo che un certo passo l'insieme ottenuto non è cambiato rispetto al passo precedente. A quel punto l'algoritmo si arresta e l'ultimo insieme che ho determinato rappresenta quella chiusura di x .

Esempio di calcolo di X_F^+

$R(stud, corso, prof, ora, aula, crediti)$

$$F = \{corso \rightarrow prof, ora\ aula \rightarrow corso, ora\ prof \rightarrow aula, corso\ stud \rightarrow crediti, ora\ stud \rightarrow aula\}$$

$$\text{Calcolo di } X_F^+ = (ora, aula)_F^+$$

$$X^0 = \{ora, aula\}$$

$$X^1 = \{ora, aula, corso\} \text{ per la seconda dipendenza}$$

$$X^2 = \{ora, aula, corso, prof\} \text{ per la prima dipendenza}$$

$$X^3 = \{ora, aula, corso, prof\} = X^2 = X_F^+$$

considerando soltanto gli attributi che ho nell'insieme in quel momento. Le esamino nell'ordine corso-prof = corso non compare, questa dipendenza non la posso considerare, nella successiva ho ora e aula determina corso, quindi ho una dipendenza che a sinistra a tutti gli attributi che appartengono all'insieme, quindi utilizzo questa dipendenza per determinare un nuovo attributo, in questo caso l'attributo corso che non è ancora presente, quindi lo aggiungo all'insieme e al passo successivo ricomincio e questa volta siccome corso è stato aggiunto posso utilizzare subito la prima dipendenza funzionale e quindi tramite questa aggiungere l'attributo prof, e a questo punto vado a creare l'insieme successivo che però rimane invariato rispetto al precedente, quindi l'insieme che ho trovato è proprio la chiusura di x rispetto ad f .

Esempio di calcolo di X_F^+

$R(A, B, C, D, E, G)$

$$F = \{AB \rightarrow C, D \rightarrow EG, C \rightarrow A, BE \rightarrow C, BC \rightarrow D, CG \rightarrow BD, ACD \rightarrow B, CE \rightarrow AG\}$$

$$\text{Calcolo di } X_F^+ = (BD)_F^+$$

$$X^0 = \{BD\}$$

$$X^1 = \{BDEG\}$$

$$X^2 = \{BCDEG\}$$

$$X^3 = \{ABCDEG\} = X_F^+$$

Un altro esempio di calcolo della chiusura: abbiamo questa relazione generica. questa volta voglio determinare la chiusura di BD, quindi parto dall'insieme che inizialmente contiene soltanto la coppia BD e poi devo andare a vedere se ci sono delle dipendenze che a sinistra hanno gli attributi BD. la prima non la posso considerare però la seconda D determinati EG, D è contenuto in BD quindi posso considerare questa dipendenza e quindi aggiungo tutti e due gli attributi che trovo a destra EG, perché non sono già presenti, quindi al passo dopo ricomincio da capo, la prima dipendenza non la posso considerare, questa volta però posso considerare la quarta BE determina C, quindi aggiungo l'attributo C, quindi potrò poi considerare la dipendenza C determina A. a questo punto vedete mi

devo fermare perché ho trovato tutti gli attributi della mia relazione, quindi è inutile che vada avanti non potrò aggiungere sicuramente nient'altro. in questo caso la chiusura di BD ci restituisce tutti gli attributi della relazione quindi vuol dire che BD è una super chiave della relazione perché tramite questo insieme di attributi riesco a determinare tutti gli altri e per verificare una chiave dovrei verificare che togliendo uno fra B e D riesco ad arrivare allo stesso insieme gli attributi.

Un'altra versione dell'algoritmo: ad ogni step io posso aggiungere più dipendenze alla volta andando a vedere se c'è più di

Calcolo di X_F^+ : altra formulazione

una dipendenza che possa applicare a partire dall'insieme creato al passo precedente. Quando faccio l'unione i duplicati poi vengono eliminati.

$$F \text{ implica } X \rightarrow Y \Leftrightarrow X \rightarrow Y \in F^+ \Leftrightarrow Y \subseteq X_F^+$$

Una

Il calcolo della chiusura di un insieme di attributi può quindi essere usato per determinare se una certa dipendenza è o non è implicata da F .

Usando le nozioni di dipendenza funzionale e di chiusura di un insieme di dipendenze si possono definire in modo formale le nozioni di *superchiave* e *chiave* di uno schema.

definizione in termini di dipendenze funzionali: ho una relazione definita sull'insieme t e un insieme F di dipendenze funzionali e x contenuto in t . Che caratteristiche deve avere x per essere una chiave della relazione? Bisogna che F , l'insieme delle dipendenze funzionali, implichi la dipendenza x determina t , cioè

- Dati $R(T)$ e F , $X \subseteq T$ è una chiave se
 - ① F implica $X \rightarrow T$
 - ② per nessun Y sottoinsieme proprio di X si ha F implica $Y \rightarrow T$
- Un insieme X che verifica la proprietà 1 è detto *superchiave* (una chiave è anche una superchiave). Si può dire che una chiave è una superchiave non ridondante.
- La proprietà 1 può essere verificata controllando se $T = X_F^+$.
- Dati $R(T)$ e F , T è sempre una superchiave (talvolta è anche l'unica chiave).

tramite x devo determinare tutti gli attributi della mia relazione e se voglio che x sia una chiave non deve esistere nessun sottoinsieme y di x tale che F implica y determina t , cioè x deve essere l'insieme più piccolo con questa caratteristica. Se invece ho un insieme x che verifica la prima proprietà ma non verifica la seconda posso dire che quell'insieme x è una super

chiave, però potrebbe contenere degli attributi ridondanti. Per dimostrare che x è una chiave basta che io dimostri che la chiusura di x rispetto ad F è uguale a t , perché questo corrisponde a dire che la dipendenza x determina t è implicata da f .

Verifica che K è chiave e calcolo di K_F^+ -1-

ORDINE(num, fornitore, indirizzo, articolo, data, quantita)
 $F = \{num \rightarrow fornitore\ data, fornitore \rightarrow indirizzo, num\ articolo \rightarrow quantita\}$

Chiave K : $num\ articolo$

$K^0 = \{num\ articolo\}$

$K^1 = \{num\ articolo\ fornitore\ data\}$ per la I dip.

$K^2 = \{num\ articolo\ fornitore\ data\ indirizzo\}$ per la II dip.

$K^3 = \{num\ articolo\ fornitore\ data\ indirizzo\ quantita\}$, III dip.

Quindi la coppia $(num\ articolo)$ è una superchiave.

Se ripetiamo il procedimento a partire solo da num otteniamo $\{num\ fornitore\ data\ indirizzo\}$ quindi num non è chiave.

Se ripetiamo il procedimento a partire solo da $articolo$ otteniamo $\{articolo\}$ quindi $articolo$ non è chiave.

determinare tutti gli attributi della relazione. Se considero $articolo$ non ho nessuna dipendenza che ha a sinistra soltanto quell'attributo, quindi tramite l' $articolo$ determino soltanto $articolo$. Questo mi permette di dire che K è chiave della relazione perché non solo mi permette di determinare tutti gli attributi ma non esiste nessun suo sottoinsieme che ha la stessa caratteristica.

Verifica che K è chiave e calcolo di K_F^+ -2-

FREQ(cod_stud, nome, indirizzo, cod_corso, nome_corso, tipo, periodo)
 $F = \{cod_stud \rightarrow nome\ indirizzo, cod_corso \rightarrow nome_corso\ tipo\ periodo\}$

Chiave K : $cod_stud\ cod_corso$

$K^0 = \{cod_stud\ cod_corso\}$

$K^1 = \{cod_stud\ cod_corso\ nome\ indirizzo\}$ per la I dip.

$K^2 =$

$\{cod_stud\ cod_corso\ nome\ indirizzo\ nome_corso\ tipo\ periodo\}$ per la II dip.

Quindi $(cod_stud\ cod_corso)$ è superchiave; inoltre $(cod_stud\ cod_corso)$ è chiave perché non è ridondante, infatti: da cod_stud si ottiene solo $\{cod_stud\ nome\ indirizzo\}$ da cod_corso si ottiene solo $\{cod_corso\ nome_corso\ tipo\ periodo\}$

informazioni relative allo studente. Quindi ho trovato la chiave della relazione.

Esempio: questa relazione indirizzo contiene gli attributi città, strada e cap. Le dipendenze: la coppia città-strada determina il cap ma il cap determina la città. Se considero la coppia città e strada, tramite la prima dipendenza trovo posso subito aggiungere il cap e quindi arrivo a determinare tutti gli attributi e chiaramente non posso toglierle nessuno perché non ho dipendenze che hanno a sinistra soltanto città o soltanto strada, quindi città strada è chiave. D'altra parte se considero strada e cap anche questa coppia di valori forma una chiave perché se parto da strada e cap poi tramite i cap determino la città e questa coppia di attributi non è ridondante perché è se ne considero soltanto uno non riesco ad ottenere tutti gli attributi.

Esempi di chiave

- *IMPIEGATO(codimp, nome, stipendio, progetto, data_finale)*
 $F = \{codimp \rightarrow nome\ stipendio, progetto \rightarrow data_finale\}$
Chiave: $codimp\ progetto$

- *IMPIEGATO(codimp, nome, stipendio, progetto, data_finale)*
 $F = \{codimp \rightarrow nome\ stipendio, codimp \rightarrow progetto, progetto \rightarrow data_finale\}$
Chiave: $codimp$

- *R(stud, corso, prof, ora, aula, crediti)*
 $F = \{corso \rightarrow prof, ora\ aula \rightarrow corso, ora\ prof \rightarrow aula, corso\ stud \rightarrow crediti, ora\ stud \rightarrow aula\}$
Chiave: $stud\ ora$

Abbiamo la tabella ordine e voglio dimostrare che la coppia num-articolo costituisce la chiave della relazione, per far questo è sufficiente trovare la chiusura di questa coppia di attributi. Tramite questa coppia numero articolo sono riuscita a determinare tutti gli attributi della relazione, quindi questo vuol dire che vale la prima condizione, cioè che K è sicuramente una super chiave della relazione. Per verificare che è una chiave devo ripetere il procedimento considerando soltanto num oppure considerando soltanto articolo per vedere se uno di questi due attributi ha la stessa caratteristica allora se ripeto il procedimento a partire dal num, quindi voglio trovare la chiusura dell'attributo num però vedete che articolo non lo raggiungerò mai tramite num perché non ho nessuna dipendenza che ha destra articolo e quindi sicuramente non è chiave perché non mi permette di

esempio: voglio dimostrare che la chiave di questa relazione è costituita dalla coppia codice studente e codice corso, quindi parto da questo insieme di attributi e poi vado ad applicare le dipendenze che posso applicare. Trovo che sicuramente questa coppia di attributi è super chiave ma per dimostrare che è chiave devo dimostrare che prendendo soltanto uno dei due attributi non riesco ad ottenere tutti gli attributi, ma questo si vede facilmente perché tramite codice studente riuscirò a determinare il nome e l'indirizzo ma non potrò mai considerare la dipendenza che ha a sinistra codice corso perché il codice corso non compare a destra da nessuna parte e analogamente se invece parto da codice corso riuscirò a determinare il nome del corso il tipo il periodo ma non riuscirò a determinare le

Verifica che K è chiave e calcolo di K_F^+ -3-

INDIRIZZO(citta, strada, CAP)

$F = \{citta\ strada \rightarrow CAP, CAP \rightarrow citta\}$

- ad ogni coppia $citta\ strada$ corrisponde un solo CAP
- ad ogni CAP corrisponde una sola $citta$

Chiave K : $citta\ strada$

$K_{piu} = \{citta\ strada\}$

$K_{piu} = \{citta\ strada\ CAP\}$ per la I dipendenza e la coppia $(citta\ strada)$ non è ridondante

Chiave K : $strada\ CAP$

$K_{piu} = \{strada\ CAP\}$

$K_{piu} = \{strada\ CAP\ citta\}$ per la II dipendenza e la coppia $(strada\ CAP)$ non è ridondante.

Per operare su insiemi di dipendenze è comodo trasformarli in una forma minimale. Si introducono i concetti di equivalenza e copertura. La relazione di equivalenza fra insiemi di dipendenze permette di stabilire quando due schemi di relazione rappresentano gli stessi fatti: basta controllare se gli attributi sono gli stessi e le dipendenze equivalenti.

Equivalenza tra insiemi di dipendenze funzionali

Gli insiemi di dipendenze funzionali F e G sono equivalenti, $F \equiv G$, se

- $F^+ = G^+$ ovvero
- $F \subseteq G^+$ e $G \subseteq F^+$ ovvero
- G implica ogni dipendenza in F e viceversa

Se $F \equiv G$, F è detto copertura di G e viceversa.

con g , oppure l'insieme f deve essere contenuto in g^+ e l'insieme g deve essere contenuto in f^+ oppure g deve implicare ogni dipendenza in f e viceversa f deve implicare ogni dipendenza in g . Se f e g sono equivalenti allora si dice che f è una copertura di g e viceversa.

Esempio di equivalenza

$R(\text{imp}, \text{ind}, \text{tel})$

$F = \{\text{imp} \rightarrow \text{tel}, \text{imp} \rightarrow \text{ind}\}$

$G = \{\text{imp ind} \rightarrow \text{tel}, \text{imp} \rightarrow \text{ind}\}$

- $F \subseteq G^+$ infatti $\text{imp} \rightarrow \text{tel} \notin G$ ma G implica $\text{imp} \rightarrow \text{tel}$ poiché $\text{tel} \in \text{imp}_G^+$
- $G \subseteq F^+$ infatti $\text{imp ind} \rightarrow \text{tel} \notin F$ ma F implica $\text{imp ind} \rightarrow \text{tel}$ poiché $\text{tel} \in (\text{imp ind})_F^+$

devo dimostrare è che se considero la dipendenza impiegato determina telefono che è in F ma non è in G . Devo dimostrare che questa dipendenza la riesco comunque a determinare tramite G . In questo modo dimostro che F è contenuto in G^+ . Parto dalla dipendenza impiegato determina telefono, questa dipendenza non appartiene a G , per dimostrare che questa dipendenza invece è contenuta in G^+ basta dimostrare che il telefono appartiene alla chiusura di impiegato rispetto a G . Se io sono in G e considero la chiusura di impiegato, tramite impiegato determino l'indirizzo e a quel punto tramite l'indirizzo determino il telefono, quindi il telefono appartiene alla chiusura di impiegato rispetto a G , quindi la dipendenza impiegato determina telefono non appartiene a G ma riesco a determinarla tramite le dipendenze che sono in G , quindi F è contenuto in G^+ . Poi devo dimostrare anche che G invece è contenuto in F^+ e questa volta la dipendenza che devo andare a considerare è impiegato, indirizzo determina telefono appartiene a G ma non appartiene ad F , io voglio dimostrare che questa dipendenza però la posso implicare dalle dipendenze in F . Calcolo la chiusura della coppia impiegato indirizzo tramite l'insieme di dipendenze di F e tramite impiegate aggiungo sia indirizzo che telefono e quindi telefono appartiene alla chiusura dell'insieme impiegato indirizzo tramite F e quindi la dipendenza impiegato indirizzo determina telefono non appartiene ad F ma è implicata dalle dipendenze che sono in F , quindi G è contenuto in F^+ . Quindi ho due insiemi che sono diversi ma equivalenti perché le dipendenze che differiscono riesco a determinarle tramite le dipendenze dell'altro insieme.

Insiemi di dipendenze non ridondanti e ridotti

- Un insieme F di dipendenze è **non ridondante** se
 - ➊ non esiste $f \in F$ tale che $F - \{f\}$ implica f ovvero tale che $(F - \{f\}) \equiv F$
- Un insieme F di dipendenze è **ridotto** sse
 - ➊ ogni dipendenza ha a destra un solo attributo
 - ➋ è non ridondante, cioè non ci sono dipendenze ridondanti
 - ➌ le dipendenze hanno parti sinistre non ridondanti, cioè per ogni $X \rightarrow Y \in F$ non esiste $A \subset X$ tale che

$$F - \{X \rightarrow Y\} \cup \{(X - A) \rightarrow Y\} \equiv F$$

sinistre nelle dipendenze che siano ridondanti, ovvero se io considero una dipendenza x determina y e con x costituito da più di un attributo, quindi consideriamo una dipendenza che ha a sinistra un certo insieme di attributi e mi chiedo se sono necessari oppure ce n'è qualcuno che potrei eventualmente togliere. Le dipendenze non devono avere a sinistra attributi ridondanti, ho attributi ridondanti quando togliendo da F la dipendenza che sto considerando, quindi x determina y , e

L'equivalenza tra insiemi di dipendenze funzionali: tramite questo concetto riesco a determinare quella che si chiama la copertura ridotta di un insieme di dipendenze funzionali, cioè dato un insieme di dipendenze riesco a trasformarlo in un insieme che è il più semplice possibile. Abbiamo insiemi f e g diversi ma equivalenti, nel senso che tramite essi riesco a determinare le stesse dipendenze. Tra due insiemi equivalenti scelgo quello più semplice. Due insiemi di dipendenze funzionali sono equivalenti se vale una di queste tre condizioni, allora la chiusura di f deve essere uguale alla chiusura di g , quindi tutte le dipendenze che riesco a determinare tramite f devono coincidere con tutte le dipendenze che riesco a determinare

Un esempio di questo concetto: consideriamo questa relazione, gli attributi sono impiegato, indirizzo e telefono e abbiamo due insiemi F e G . In F ho due dipendenze l'impiegato determina il telefono e l'impiegato determina indirizzo. In G invece ho queste due dipendenze da impiegato e l'indirizzo determinano il telefono e l'impiegato determina l'indirizzo. Dimostrare che questi due insiemi sono equivalenti, quindi per dimostrarlo uso questa seconda formulazione, cioè dimostro che F contenuto in G^+ e che G è contenuto in F^+ . Questi due insiemi hanno una dipendenza in comune, quindi quella in comune è inutile che vada a considerarla. Il problema sono le dipendenze che sono diverse, quindi quello che

dimostrare è che se considero la dipendenza impiegato determina telefono che è in F ma non è in G . Devo dimostrare che questa dipendenza la riesco comunque a determinare tramite G . In questo modo dimostro che F è contenuto in G^+ . Parto dalla dipendenza impiegato determina telefono, questa dipendenza non appartiene a G , per dimostrare che questa dipendenza invece è contenuta in G^+ basta dimostrare che il telefono appartiene alla chiusura di impiegato rispetto a G . Se io sono in G e considero la chiusura di impiegato, tramite impiegato determino l'indirizzo e a quel punto tramite l'indirizzo determino il telefono, quindi il telefono appartiene alla chiusura di impiegato rispetto a G , quindi la dipendenza impiegato determina telefono non appartiene a G ma riesco a determinarla tramite le dipendenze che sono in G , quindi F è contenuto in G^+ . Poi devo dimostrare anche che G invece è contenuto in F^+ e questa volta la dipendenza che devo andare a considerare è impiegato, indirizzo determina telefono appartiene a G ma non appartiene ad F , io voglio dimostrare che questa dipendenza però la posso implicare dalle dipendenze in F . Calcolo la chiusura della coppia impiegato indirizzo tramite l'insieme di dipendenze di F e tramite impiegate aggiungo sia indirizzo che telefono e quindi telefono appartiene alla chiusura dell'insieme impiegato indirizzo tramite F e quindi la dipendenza impiegato indirizzo determina telefono non appartiene ad F ma è implicata dalle dipendenze che sono in F , quindi G è contenuto in F^+ . Quindi ho due insiemi che sono diversi ma equivalenti per essere in forma ridotta è che non esistano delle parti

sostituendola con la stessa dipendenza in cui però a sinistra ho semplificato la dipendenza togliendo un certo attributo ottengo un insieme di dipendenze che è equivalente a quello di partenza.

Normalizzazione-20220427 0701-1

Esempi di insiemi di dipendenze non ridondanti e ridotti

CORSO(codice, nome, CFU)

- $F = \{codice \rightarrow nome, codice \text{ nome} \rightarrow CFU, codice \rightarrow CFU\}$
- F è ridondante, infatti
 - $F \equiv F - \{codice \rightarrow CFU\}$ ovvero
 - $(F - \{codice \rightarrow CFU\}) \text{ implica } codice \rightarrow CFU$ ovvero
 - $codice \rightarrow CFU \in (F - \{codice \rightarrow CFU\})^+$ ovvero
 - $CFU \in codice_{F - \{codice \rightarrow CFU\}}^+$
- $F_1 = \{codice \rightarrow nome, codice \text{ nome} \rightarrow CFU\}$
- F_1 non è ridotto, infatti
 - $(F_1 - \{codice \text{ nome} \rightarrow CFU\} \cup \{codice \rightarrow CFU\}) \equiv F_1$ ovvero
 - $F_1 \text{ implica } codice \rightarrow CFU$ ovvero
 - $codice \rightarrow CFU \in F_1^+$ ovvero
 - $CFU \in codice_{F_1}^+$
- $F_2 = \{codice \rightarrow nome, codice \rightarrow CFU\}$ è ridotto

ridotto perché c'è una dipendenza che a sinistra ha più di un attributo, quindi voglio dimostrare che la parte sinistra di questa dipendenza contiene un attributo ridondante, in particolare l'attributo ridondante è nome, quindi voglio dimostrare che se da f_1 tolgo questa dipendenza, ma aggiungo la stessa dipendenza in cui ho tolto a sinistra il nome, ottengo un insieme equivalente a quello iniziale, quindi devo trovare la chiusura dell'attributo codice tramite f_1 e la chiusura di codice conterrà sicuramente il nome e poi tramite la seconda dipendenza trovo cfu, quindi cfu appartiene alla chiusura di codice, ciò significa che questi due insieme sono equivalenti, per cui l'attributo nome non mi serve, è ridondante. Dopo questi due passaggi ho un nuovo insieme f_2 , costituito dalla dipendenza codice determina nome e codice determina cfu, che è un insieme equivalente all'insieme f ma è in forma ridotta.

Esempio: immaginiamo di avere una generica relazione definita su tre attributi. La prima contiene a sinistra due attributi a e b ma in realtà ne basta solo uno perché ho a determina b e b determina a , quindi se elimino b dato che a determina b riesco comunque a determinare c , analogamente se elimino a , dato che b determina a riesco comunque a determinare c , quindi vuol dire che in questa situazione particolare io posso semplificare la prima dipendenza a sinistra in due modi diversi perché posso eliminare a oppure posso eliminare b , sono due strategie equivalenti perché ottengo insiemi equivalenti all'insieme f . Le semplificazioni che posso fare non sono in generale uniche. Esempio: la prima dipendenza la posso semplificare in due modi, quindi posso eliminare il telefono e quindi ottenere questo primo insieme di dipendenze, impiegato determina ufficio, impiegato determina telefono, telefono determina impiegato oppure elimino impiegato e mi rimane telefono determina ufficio. La copertura ridotta non è unica.

Calcolo di una copertura ridotta per F

- ➊ Trasforma ogni dipendenza $X \rightarrow Y$ con $|Y| > 1$ in dipendenze che hanno a destra un solo attributo (regola di decomposizione)
- ➋ Indicato con F l'insieme di dipendenze corrente, per ogni dipendenza $X \rightarrow A \subseteq F$ con $|A| > 1$ controlla se X contiene

Attenzione all'ordine dei passi

I passi 2 e 3 non devono essere scambiati.

$R(stanza, imp, tel)$

$F = \{stanza \text{ imp} \rightarrow tel, imp \rightarrow stanza, tel \rightarrow stanza\}$

ogni impiegato può essere reperito con un certo numero di telefono, però non è detto che un telefono serva una sola persona

2 $\{imp \rightarrow tel, imp \rightarrow stanza, tel \rightarrow stanza\} \quad tel \in (imp)^+_F$

3 $\{imp \rightarrow tel, tel \rightarrow stanza\} \quad imp \rightarrow stanza$ è ridondante
è una copertura minimale

cambiando l'ordine:

3 $\{stanza \text{ imp} \rightarrow tel, imp \rightarrow stanza, tel \rightarrow stanza\}$

2 $\{imp \rightarrow tel, imp \rightarrow stanza, tel \rightarrow stanza\}$
non è una copertura minimale

Esempi: abbiamo una prima relazione e vogliamo far vedere che questo insieme è ridondante, la dipendenza codice determina cfu è di troppo, la posso eliminare perché se io considero l'insieme f tolta questa dipendenza ottengo un nuovo insieme che equivale a quello di partenza e per dimostrarlo tolgo questa dipendenza dall'insieme e dimostro che quelle che rimangono mi permettono comunque di determinare questa dipendenza. Per dimostrare che l'insieme tolto la dipendenza codice determina cfu implica la dipendenza codice determina cfu, devo calcolare la chiusura di codice rispetto alle due dipendenze che sono rimaste, tramite codice determino il nome e poi tramite codice-nome determino cfu. L'insieme che abbiamo ottenuto al passo precedente eliminando la dipendenza codice determina cfu però non è ancora

Non unicità della copertura ridotta

Un insieme di dipendenze F può avere più di una copertura ridotta

- $R(ABC)$
 $F = \{AB \rightarrow C, A \rightarrow B, B \rightarrow A\}$
- $G_1 = \{A \rightarrow C, A \rightarrow B, B \rightarrow A\}$
 $G_2 = \{B \rightarrow C, A \rightarrow B, B \rightarrow A\}$
- $R(imp, ufficio, telefono)$
 $F = \{imp \text{ telefono} \rightarrow ufficio, imp \rightarrow telefono, telefono \rightarrow imp\}$ ogni impiegato ha un telefono personale sulla scrivania
- $G_1 = \{imp \rightarrow ufficio, imp \rightarrow telefono, telefono \rightarrow imp\}$
 $G_2 = \{telefono \rightarrow ufficio, imp \rightarrow telefono, telefono \rightarrow imp\}$

L'algoritmo per il calcolo di una copertura: cerca di ottenere un insieme di dipendenza in forma ridotta. Il primo step è facile perché si tratta di trasformare tutte le dipendenze che hanno a destra un numero di attributi maggiori di uno in dipendenze che hanno a destra un solo attributo (basta applicare la regola di decomposizione), dopodiché dobbiamo eliminare le dipendenze di troppo e gli attributi a sinistra ridondanti (questa operazione deve essere fatta in questo ordine), l'ultimo passo è quello di controllare se ci sono dipendenze ridondanti, quindi elimino eventualmente una dipendenza x determina a , perché quelle rimanenti mi permettono comunque di avere un insieme equivalente a quello di partenza. È chiaro che al secondo step posso generare a seconda delle operazioni che ho fatto un insieme diverso e lo stesso anche il passo tre può essere realizzato in più modi. Attenzione all'ordine dei passi degli algoritmi cioè i passi due e tre non devono essere scambiati. Esempio: (il passo uno è già applicato perché la destra ho soltanto un attributo in tutti i casi) applicando prima il passo due e poi il passo tre: al passo due posso eventualmente

semplificare la prima dipendenza se uno di questi due attributi sinistri è ridondante, infatti l'attributo stanza può essere eliminato, e al passo tre osservo che c'è una ricorsione, l'impiegato determina il telefono e il telefono determina la stanza quindi la dipendenza impiegato determina stanza non mi serve perché è implicata delle altre due, e quella che ottengo in questo modo è una copertura minimale. Se io avessi applicato all'algoritmo prima il passo tre e poi il passo due è chiaro che al passo tre la dipendenza che io considero è quella con a sinistra sia stanza che l'impiegato, in questo caso io non posso eliminare nessuna perché non è implicata dalle altre due e quindi poi al passo due la semplificazione della prima dipendenza eliminando l'attributo stanza mi da come risultato una copertura non minimale, perché ho una dipendenza ridondante.

Esempio di calcolo di copertura ridotta

STUD(matr, nome, fascia_reddito, tasse, cdl, presidente, progetto, tutor)

$$F = \{ \text{matr} \rightarrow \text{fascia_reddito} \text{ tasse } \text{cdl} \text{ presidente}, \text{fascia_reddito} \rightarrow \text{tasse}, \text{matr presidente} \rightarrow \text{nome} \text{ cdl}, \text{cdl} \rightarrow \text{presidente}, \text{presidente} \rightarrow \text{cdl}, \text{matr progetto cdl} \rightarrow \text{tutor matr} \}$$

Una copertura ridotta è:

$$F = \{ \text{matr} \rightarrow \text{fascia_reddito}, \text{matr} \rightarrow \text{cdl}, \text{fascia_reddito} \rightarrow \text{tasse}, \text{matr} \rightarrow \text{nome}, \text{presidente} \rightarrow \text{cdl}, \text{cdl} \rightarrow \text{presidente}, \text{matr progetto} \rightarrow \text{tutor} \}$$

Attenzione: la verifica del passo 2) dell'algoritmo va fatta rispetto all'insieme F che contiene la dipendenza sotto esame (si veda ad esempio la dipendenza $\text{matr presidente} \rightarrow \text{nome}$ in cui presidente risulta ridondante).

Esempio: trovare la copertura ridotta: applicare lo step uno quindi eseguo la decomposizione numerando via via le dipendenze che si vanno a determinare, quindi nel primo caso la matricola determina la fascia di reddito, le tasse, il CDL e presidente è chiaro che la posso decomporre in quattro dipendenze che numero da uno a quattro, poi vado a considerare la successiva la fascia di reddito determina le tasse, poi matricola e presidente determina nome e CDL la posso decomporre in altre due dipendenze, cdl determina presidente la riscrivo, lo stesso quella successiva, e poi ho un'altra decomposizione che posso fare in corrispondenza dell'ultima dipendenza. Alla fine del primo step ho enumerato le dipendenze da 1 a 11 (utile perché nelle fasi successive piuttosto che dover andare a riscrivere la dipendenza per estese potete far riferimento al numero). L'ultima dipendenza matricola progetto cdl determina matricola è una dipendenza ovvia perché ho a sinistra un insieme a destra un suo sottoinsieme, quindi la posso eliminare.

Esempio: abbiamo due insiemi F di dipendenze funzionali in cui ha semplicemente cambiato l'ordine delle dipendenze all'interno dell'insieme. cosa c'entra l'ordine? c'entra perché quello che in genere si fa è applicare l'algoritmo seguendo l'ordine con cui compaiono nell'insieme F , per cui se l'ordine che si considera è il primo (il passo che posso fare è soltanto il terzo, quello di eliminazione delle dipendenze) e se io le dipendenze le vedo in quest'ordine mi accorgo che il codice fiscale determina la matricola e che la matricola determina il corso di laurea quindi il codice fiscale determina il corso di laurea la posso eliminare. Se però vedo le dipendenze in un ordine diverso posso decidere di eliminare la dipendenza matricola determina cdl.

Esempio di calcolo di copertura ridotta: passo 1

$$F = \{ \text{matr} \rightarrow \text{fascia_reddito} \text{ tasse } \text{cdl} \text{ presidente}, \text{fascia_reddito} \rightarrow \text{tasse}, \text{matr presidente} \rightarrow \text{nome} \text{ cdl}, \text{cdl} \rightarrow \text{presidente}, \text{presidente} \rightarrow \text{cdl}, \text{matr progetto cdl} \rightarrow \text{tutor matr} \}$$

- (1) $\text{matr} \rightarrow \text{fascia_reddito}$
- (2) $\text{matr} \rightarrow \text{tasse}$
- (3) $\text{matr} \rightarrow \text{cdl}$
- (4) $\text{matr} \rightarrow \text{presidente}$
- (5) $\text{fascia_reddito} \rightarrow \text{tasse}$
- (6) $\text{matr presidente} \rightarrow \text{nome}$
- (7) $\text{matr presidente} \rightarrow \text{cdl}$
- (8) $\text{cdl} \rightarrow \text{presidente}$
- (9) $\text{presidente} \rightarrow \text{cdl}$
- (10) $\text{matr progetto cdl} \rightarrow \text{tutor}$
- (11) $\text{matr progetto cdl} \rightarrow \text{matr}$ è una dipendenza ovvia

Esempio di calcolo di copertura ridotta: passo 3

- (1) $\text{matr} \rightarrow \text{fascia_reddito}$
- (2) $\text{matr} \rightarrow \text{tasse}$ si elimina per (1) e (5)
- (3) $\text{matr} \rightarrow \text{cdl}$
- (4) $\text{matr} \rightarrow \text{presidente}$ si elimina per (3) e (8)
- (5) $\text{fascia_reddito} \rightarrow \text{tasse}$
- (6) $\text{matr} \rightarrow \text{nome}$
- (7) $\text{presidente} \rightarrow \text{cdl}$
- (8) $\text{cdl} \rightarrow \text{presidente}$
- (9) $\text{presidente} \rightarrow \text{cdl}$ si elimina perché identica a (7)
- (10) $\text{matr progetto} \rightarrow \text{tutor}$

Step due dell'algoritmo: esaminare le dipendenze di questa lista che hanno a sinistra un numero di attributi maggiore di uno, quindi in questo caso le dipendenze sono la 6, 7 e 10. Devo vedere se in queste tre dipendenze c'è qualche attributo a sinistra che posso eliminare. Inizio da matricola ma non è ridondante perché se considero la chiusura di presidente tramite presidente determino il cdl ma non riesco a derivare altri attributi per cui non posso togliere, se invece considero la chiusura di matricola, essa conterrà la fascia di reddito, tasse, CDL e presidente, quindi tramite la matricola con presidente determino anche il nome, quindi posso eliminare il presidente da questa dipendenza. Nella dipendenza 7 posso eliminare la matricola perché la chiusura di presidente contiene il cdl (in questo caso anche la chiusura di matricola contiene il cdl, avrei potuto eliminare anche l'altra). Nell'ultima dipendenza posso eliminare cdl, calcolo la chiusura della coppia matricola progetto che contiene il tutor. Rimangono due attributi a sinistra nella 10, quindi uno potrebbe chiedersi se posso fare ancora qualche eliminazione però non è possibile perché il tutor non appartiene né alla chiusura di matricola né tantomeno alla chiusura di progetto. Nell'ultimo passo devo decidere che cosa eliminare, la matricola determina tasse è una dipendenza che posso eliminare, perché è implicata dalla prima matricola che determina fascia di reddito e la quinta fascia di reddito determina tasse, la matricola determina il presidente la posso eliminare perché la posso ottenere tramite la dipendenza 3 e la dipendenza 8, infine posso eliminare la 9 perché è identica alla 7. L'insieme finale è di copertura minimale,

quindi questo insieme rappresenta un insieme di dipendenze equivalente a quello di partenza però più semplice. Questo è il primo passaggio per normalizzare una base di dati, quindi questo ci servirà in tutte le fasi successive.

Per eliminare anomalie da uno schema mal definito si cerca di decomporlo in schemi più piccoli che godono di particolari proprietà (forme normali) ma sono in qualche senso equivalenti allo schema originale. Si richiede in genere che lo schema soddisfi due condizioni indipendenti fra loro: preservi i dati e preservi le dipendenze.

Decomposizione di relazioni

Data r di schema $R(T)$, siano T_1 e T_2 due sottoinsiemi di T tali che $T_1 \cup T_2 = T$

Una decomposizione d di r in due relazioni, secondo gli attributi T_1 e T_2 , è la coppia di relazioni r_1 e r_2 che si ottengono effettuando la proiezione di r su T_1 e T_2 rispettivamente

$$d = (r_1 = \pi_{T_1}(r), r_2 = \pi_{T_2}(r))$$

In generale una decomposizione di r secondo gli attributi T_1, T_2, \dots, T_k (con $T_1 \cup T_2 \cup \dots \cup T_k = T$) può essere ottenuta applicando in modo iterativo una decomposizione in due.

In seguito tratteremo la decomposizione in due.

Cosa significa decomporre uno schema di relazione: una decomposizione di uno schema non è altro che la proiezione su due sottoinsiemi uno e due della relazione di partenza, quindi decomposizione di una tabella in due sottotabelle ma posso applicare poi ad ognuna delle due tabelle lo stesso procedimento quindi in generale avremo a che fare con tabelle che posso decomporre un certo numero di volte. Ad ognuna delle tabelle che si ottengono posso riapplicare il procedimento. Una decomposizione è senza perdita se facendo il join naturale di r_1 e r_2 ottengo r . I due insiemi t_1 e t_2 non possono essere disgiunti perché altrimenti quando vado a fare poi il join naturale ottengo il prodotto cartesiano delle due relazioni, quindi è chiaro che fare una decomposizione senza perdita devo scegliere i due insiemi in maniera che l'intersezione non sia vuota.

Esempio di decomposizione con perdita

$R(forn, ind, articolo, prezzo)$,
 $F = \{forn \rightarrow ind, forn\ articolo \rightarrow prezzo\}$
 $T_1 = \{forn, ind, articolo\}, T_2 = \{articolo, prezzo\}$

forn	ind	articolo	prezzo
Rossi	Prato	libro	3
Verdi	Prato	penna	2
Rossi	Prato	penna	2
Verdi	Prato	libro	1

forn	ind	articolo
Rossi	Prato	libro
Verdi	Prato	penna
Rossi	Prato	penna
Verdi	Prato	libro

articolo	prezzo
libro	3
penna	2
libro	1

$$r \subseteq r_1 \text{ JOIN } r_2$$

Esempio: in questa tabella sono presenti ridondanze perché tutte le volte che compare lo stesso fornitore viene ripetuto il suo indirizzo e tutte le volte che un certo fornitore vende un certo articolo ritrovo lo stesso prezzo, allora decido di fare una decomposizione prendendo T_1 uguale a fornitore, indirizzo e articolo e T_2 uguale articolo e prezzo. Cosa succede quando faccio la proiezione su T_1 ? Ottengo una tabella con quattro tuple. Quando vado a fare la proiezione su T_2 ? Otterrò una tabella con tre record, perché in questo caso due record sono identici e quindi uno si elimina. Se ricalcolo il join naturale di R_1 e R_2 in questo caso l'attributo in comune delle due relazioni è articolo, quindi quando vado a ricreare il join naturale di queste due tabelle succederà che in corrispondenza di libro io avrò due nuove tuple, una con prezzo tre e una con prezzo uno, quindi ottengo una nuova relazione che contiene quella iniziale ma non è uguale. Questo è un esempio che mostra che

Condizione per decomposizioni senza perdita

Data r di schema $R(T)$, T_1 e T_2 tali che $T_1 \cup T_2 = T$, sia $X = T_1 \cap T_2$, la decomposizione $d = (r_1 = \pi_{T_1}(r), r_2 = \pi_{T_2}(r))$ è senza perdita se r soddisfa $X \rightarrow T_1$ oppure $X \rightarrow T_2$

Dati $R(T)$ e F , T_1 e T_2 tali che $T_1 \cup T_2 = T$, sia $X = T_1 \cap T_2$, ogni r che soddisfa F può essere decomposta senza perdita secondo T_1 e T_2 se

$$X \rightarrow T_1 \in F^+ \text{ oppure } X \rightarrow T_2 \in F^+$$

La condizione precedente è vera se X è superchiave di $R_1(T_1)$ oppure di $R_2(T_2)$

Si può anche dire che in questo caso la decomposizione dello schema $R(T)$ nei due schemi $R_1(T_1)$ e $R_2(T_2)$ è senza perdita

Esempio di decomposizione senza perdita

$R(forn, ind, articolo, prezzo)$,
 $F = \{forn \rightarrow ind, forn\ articolo \rightarrow prezzo\}$
 $T_1 = \{forn, ind\}, T_2 = \{forn, articolo, prezzo\}, T_1 \cap T_2 \rightarrow T_1 \in F^+$

forn	ind	articolo	prezzo
Rossi	Prato	libro	3
Verdi	Prato	penna	2
Rossi	Prato	penna	2
Verdi	Prato	libro	1

forn	ind
Rossi	Prato
Verdi	Prato

forn	articolo	prezzo
Rossi	libro	3
Verdi	penna	2
Rossi	penna	2
Verdi	libro	1

$$r = r_1 \text{ JOIN } r_2$$

prendere due insiemi che hanno intersezione non nulla non ci garantisce poi che il join naturale coincida con l'insieme di partenza.

Quello che deve succedere è che l'intersezione x deve essere chiave di una delle due relazioni che vado a creare. Quello che deve succedere è che x l'intersezione sia una super chiave per una delle due tabelle che vado a considerare. Esempio: faccio una decomposizione diversa, prendo come T_1 la coppia fornitore e indirizzo e come T_2 prendo invece il fornitore, l'articolo e il prezzo, allora l'intersezione di T_1 e T_2 in questo caso è fornitore e tramite il fornitore ad esempio determino anche l'indirizzo tramite la prima dipendenza, quindi con il fornitore determino T_1 , in altre parole fornitore è chiave della prima decomposizione. Quindi se vado a fare la decomposizione utilizzando questa scelta e poi vado a ricostruire a ritroso il join naturale ritrovo la tabella iniziale.

Esempio di decomposizione con perdita

$R(stud, corso, prof)$, $F = \{dipendenze banali\}$
 $T_1 = \{stud, corso\}$, $T_2 = \{corso, prof\}$

Questa decomposizione non è senza perdita, infatti:

$corso = \{stud, corso\} \cap \{corso, prof\}$
 $NOT(corso \rightarrow stud \ corso)$
 $NOT(corso \rightarrow corso \ prof)$

stud	corso	prof
Rossi	BDSI	Cesarini
Verdi	BDSI	Merlini

Proiezione di dipendenze

Dati $R(T)$, F e $T_1 \subseteq T$, la proiezione di F su T_1 è l'insieme delle dipendenze appartenenti a F^+ che coinvolgono gli attributi di T_1
 $\pi_{T_1}(F) = \{X \rightarrow Y \in F^+ | X \subseteq T_1, Y \subseteq T_1\}$

- **IMP(codice, qualifica, stip).**
 $F = \{codice \rightarrow qualifica, qualifica \rightarrow stip\}$
 $\pi_{codice \ stip}(F) = codice \rightarrow stip \cup \{dipendenze ovvie\}$
- **R(forn, ind, articolo, prezzo).**
 $F = \{forn \rightarrow ind, forn \ articolo \rightarrow prezzo\}$
 $\pi_{forn \ ind}(F) = \{forn \rightarrow ind\} \cup \{dipendenze ovvie\}$
 $\pi_{forn \ articolo}(F) = \{forn \rightarrow ind\} \cup \{dipendenze ovvie\}$
 $\pi_{forn \ articolo}(F) = \{dipendenze ovvie\}$
- **INDIRIZZO(città, via, CAP).**
 $F = \{città \ via \rightarrow CAP, CAP \rightarrow città\}$
 $\pi_{via \ CAP}(F) = \{dipendenze ovvie\}$
 $\pi_{città \ CAP}(F) = CAP \rightarrow città \cup \{dipendenze ovvie\}$

costituito da codice, stipendio, quindi considero codice e stipendio e vado a fare la proiezione di F su questo insieme e nessuna delle due dipendenze che sono presenti in F può essere inserita nella proiezione, perché tutti e due contengono qualifica e qualifica non fa parte dell'insieme che sto considerando, però se io le combino ed applico la regola di transitività, tramite codice determino lo stipendio, questa è una dipendenza che appartiene alla proiezione di F sull'insieme codice-stipendio, perché è costituita soltanto da attributi che hanno a destra e a sinistra attributi che sono in T_1 . Si tratta di andare a selezionare dell'insieme F quelle dipendenze che hanno a destra e a sinistra attributi di T_1 , però a volte la cosa è un pochino più complessa perché nella definizione la dipendenza appartiene alla chiusura di F , quindi non è detto che la dipendenza sia contenuta in F ma tramite l'applicazione delle dipendenze io posso trovarne una che è costituita soltanto da attributi che sono contenuti nell'insieme T_1 .

Esempio di decomposizione con perdita: Per fare la decomposizione come T_1 prendiamo stud e corso e come T_2 prendiamo corso e professore, in questo caso l'intersezione è corso ma su questo schema non esiste nessuna delle due dipendenze che mi garantirebbero che la decomposizione sia senza perdita, perché non è vero che il corso determina lo studente e non è vero che il corso determina il professore. Se vado a fare la proiezione di T_1 e T_2 e poi a fare di nuovo il join otterrei una relazione che contiene quella di partenza.

Le dipendenze iniziali le devo distribuire un po da una parte e un po dall'altra con il criterio dato dal concetto di proiezione di un insieme di dipendenze funzionali su un insieme di attributi, quindi ho una relazione definita su un insieme di attributi T , ho l'insieme di dipendenze funzionali associate ed ho un sottoinsieme T_1 di T . Che cos'è la proiezione di F su T_1 ? Questa proiezione è data dall'insieme delle dipendenze che appartengono alla chiusura di F e che coinvolgono soltanto gli attributi che sono in T_1 . Usa lo stesso simbolo dell'algebra relazionale però sto facendo una proiezione che ha un significato diverso, sto proiettando dipendenze su un insieme di attributi quindi ottengo un insieme di dipendenze funzionali che appartengono alla chiusura di F che però hanno una caratteristica particolare cioè sia la parte sinistra e la parte destra sono costituite da attributi che sono in T_1 . Esempio: consideriamo la tabella codice, qualifica e stipendio e le dipendenze sono il codice che determina la qualifica e la qualifica determina lo stipendio, voglio trovare la proiezione di F su l'insieme

nell'insieme T_1 . Quindi ecco ricordatevi questo primo esempio perché è indicativo del fatto che fare una proiezione può coinvolgere delle dipendenze che non compaiono in modo esplicito nell'insieme F . Un altro esempio: abbiamo il fornitore, l'indirizzo, l'articolo e il prezzo e in questo caso le dipendenze sono il fornitore determina l'indirizzo e il fornitore e l'articolo determinano il prezzo, ci sono vari esempi di proiezioni di dipendenze, la prima sull'insieme costituito da fornitore e indirizzo, in questo caso mi compare proprio nell'insieme F una dipendenza che ha sia a sinistra che a destra e gli attributi che sto considerando, quindi quella sicuramente farà parte della mia proiezione e posso sempre immaginare che ci siano le dipendenze ovvie. Se considero fornitore indirizzo articolo quindi

Conservazione delle dipendenze

Dati $R(T)$ e F , la decomposizione in $R_1(T_1)$ e $R_2(T_2)$ conserva le dipendenze se $F \equiv \pi_{T_1}(F) \cup \pi_{T_2}(F)$

- **Decomposizione che conserva le dipendenze:**
 $IMP(cod, nome, qualifica, stip)$
 $F = \{cod \rightarrow nome \ qualifica \ stip, qualifica \rightarrow stip\}$
 $IMP_1(cod, nome, qualifica)$
 $\pi_{T_1}(F) = \{cod \rightarrow nome \ qualifica\}$
 $IMP_2(qualifica, stip) \quad \pi_{T_2}(F) = \{qualifica \rightarrow stip\}$
- **Decomposizione che non conserva le dipendenze:**
 $IMP(cod, nome, qualifica, stip)$
 $F = \{cod \rightarrow nome \ qualifica \ stip, qualifica \rightarrow stip\}$
 $IMP_1(cod, nome, qualifica)$
 $\pi_{T_1}(F) = \{cod \rightarrow nome \ qualifica\}$
 $IMP_2(cod, stip) \quad \pi_{T_2}(F) = \{cod \rightarrow stip\}$

sicuramente la dipendenza fornitore indirizzo fa parte della proiezione e non posso aggiungere neanche in questo caso la seconda perché prezzo non è contenuto e di nuovo non riesco ad applicare queste dipendenze in modo da ottenerne altre. Se considero fornitore e articolo quindi un altro sottoinsieme addirittura nessuna delle due dipendenze la posso includere perché non ce n'è nessuna che contiene soltanto questi attributi e quindi in questo caso la proiezione coincide soltanto con l' insieme delle dipendenze ovvie. Un altro esempio: se considero come insieme via e cap la proiezione mi dà soltanto le dipendenze ovvie, perché non riesco ad includere nessuna delle due delle dipendenze presenti né riesco a considerarne dall'applicazione di F, se invece considero la coppia città e cap in quel caso potrò aggiungere alla proiezione la dipendenza cap determina città. Non voglio perdere le dipendenze cioè questa è un'altra caratteristica che deve avere della decomposizione e quello che mi aspetto dalla decomposizione è che i due schemi T1 e T2 in cui ho decomposto la mia tabella iniziale conservino le dipendenze, cioè quello che deve succedere è che se io vado a considerare l'unione delle proiezioni su T1 e T2 dell'insieme delle dipendenze funzionali devo trovare un insieme equivalente a quello di partenza. La decomposizione devono avere queste due caratteristiche, devono garantire di essere senza perdita ma devono anche garantirci la conservazione delle dipendenze.

È importante conservare le dipendenze perché se le dipendenze ci danno dei vincoli sui dati che esistono non possiamo ignorarle. Abbiamo impiegato con attributi codice, nome, indirizzo, progetto e data finale. Come dipendenze abbiamo il codice dell'impiegato determina il nome, l'indirizzo, il progetto e la data finale e poi il progetto determina la data finale. Una

Perché è importante conservare le dipendenze -1-

$IMP(cod, nome, ind, prog, data_finale)$
 $F = \{cod \rightarrow nome \ ind \ prog \ data_finale, prog \rightarrow data_finale\}$
 $R_1(cod, nome, ind, prog) \quad F_1 = \{cod \rightarrow nome \ ind \ prog\}$
 $R_2(cod, data_finale) \quad F_2 = \{cod \rightarrow data_finale\}$

cod	nome	ind	prog
3	Rossi	FI	A
6	Bianchi	MI	A
5	Verdi	TO	B
8	Neri	TO	B

cod	data_finale
3	31/12/2012
6	31/12/2012
5	31/03/2013
8	30/06/2013

Può essere fatto l'inserimento in blu senza violare nessun vincolo
Chiaramente l'inserimento non si accorda con la semantica di quella che era la tabella originaria per la dipendenza $prog \rightarrow data_finale$.

poi andrebbe in contraddizione con la dipendenza che progetto determina la data finale, perché avrei il riferimento al progetto B però con due date finali diverse. Non deve succedere perché poi arrivò ad avere dei dati che sono inconsistenti con i vincoli iniziali, quindi l'obiettivo della normalizzazione è quello di avere delle decomposizioni che siano senza perdita ma devono essere mantenute anche le dipendenze.

Normalizzazione-20220503 0705-1

Esempio che evidenzia come mai è importante che la decomposizione conservi le dipendenze: vado a decomporre in due tabelle, una con attributi città e cap e una con attributi via e cap, allora queste due tabelle hanno come intersezione cap, quindi cap è sicuramente la chiave di R1, perché cap determina città, quindi sicuramente questa decomposizione è senza perdita. D'altra parte se andiamo a fare la proiezione delle dipendenze funzionali su questi due sottoschemi, nel caso di R1 possiamo associare a questo schema la dipendenza cap determina città, più le dipendenze ovvie, quando però andiamo a considerare la coppia via e cap non possiamo aggiungere a questo schema nessuna dipendenza se non quelle ovvie, perché non esistono dipendenze che contengono soltanto attributi in T2. Se andiamo a fare il join di queste due tabelle rispetto all'attributo cap vengono fuori queste due tuple che non soddisfano la dipendenza che abbiamo perso, quindi non è più vero che città e via determinano cap.

Perché è importante conservare le dipendenze -2-

$INDIRIZZO(citta, via, CAP)$
 $F = \{citta \ via \rightarrow CAP, CAP \rightarrow citta\}$
 $R_1(citta, CAP) \quad F_1 = \{CAP \rightarrow citta\} \cup \{\text{dipendenze ovvie}\}$
 $R_2(via, CAP) \quad F_2 = \{\text{dipendenze ovvie}\}$
Se nella base di dati abbiamo definito R_1 e R_2 sono le seguenti istanze:

citta	CAP
FI	02138
FI	02139

via	CAP
Via Verdi	02138
Via Verdi	02139

Se proviamo a ricostruire una istanza di $INDIRIZZO$ con un JOIN troviamo un'istanza che viola i vincoli di $INDIRIZZO$:

citta	via	CAP
FI	Via Verdi	02138
FI	Via Verdi	02139

Decomposizioni senza (con) perdita e conservazione delle dipendenze -1-

IMP(cod, nome, progetto, budget)

$$F = \{cod \rightarrow nome \text{ progetto} \text{ budget}, \text{progetto} \rightarrow budget\}$$

$$d = \{R_1(cod, nome, progetto), R_2(cod, budget)\}$$

d è senza perdita poiché $T_1 \cap T_2 = cod$ è chiave di R_1 (e di R_2)

$$\pi_{T_1}(F) = \{cod \rightarrow nome \text{ progetto}\} \cup \{dip. ovvie\}$$

$$\pi_{T_2}(F) = \{cod \rightarrow budget\} \cup \{dip. ovvie\}$$

d non conserva le dipendenze poiché

$$(progetto \rightarrow budget) \notin (\bigcup_i \pi_{T_i}(F))^+$$

$$d = \{R_1(cod, nome, progetto), R_2(progetto, budget)\}$$

d è senza perdita poiché $T_1 \cap T_2 = progetto$ è chiave di R_2

d conserva le dipendenze poiché $F \equiv (\bigcup_i \pi_{T_i}(F))$

Decomposizioni senza (con) perdita e conservazione delle dipendenze -2-

INDIRIZZO(citta, via, CAP)

$$F = \{citta \text{ via} \rightarrow CAP, CAP \rightarrow citta\}$$

$$d = \{R_1(citta, CAP), R_2(via, CAP)\}$$

d è senza perdita poiché $T_1 \cap T_2 = CAP$ è chiave di R_1

d non conserva le dipendenze poiché

$$(\bigcup_i \pi_{T_i}(F)) = \{CAP \rightarrow citta\} \cup \{dip. ovvie\}$$

Si può verificare che d è l'unica decomposizione di *INDIRIZZO* senza perdita.

Non si può trovare una decomposizione di *INDIRIZZO* che sia senza perdita e conservi le dipendenze.

Decomposizioni senza (con) perdita e conservazione delle dipendenze -3-

Esempio di decomposizione che conserva le dipendenze ma non è senza perdita

R(forn, ind, articolo, colore)

$$F = \{forn \rightarrow ind, articolo \rightarrow colore\}$$

$$d = \{R_1(forn, ind), R_2(articolo, colore)\}$$

d non è senza perdita: $T_1 \cap T_2 = \emptyset$

d conserva le dipendenze:

$$\pi_{forn, ind}(F) \cup \pi_{articolo, colore}(F) =$$

$$\{forn \rightarrow ind\} \cup \{articolo \rightarrow colore\} = F$$

l'intersezione di queste due relazioni è l'insieme vuoto, d'altra parte però questa decomposizione conserva le dipendenze, perché ovviamente alla prima potrà associare la dipendenza fornito determina indirizzo, alla seconda potrà associare la dipendenza articolo determina colore, quindi il problema di questa decomposizione è che non mi permette di ricostruire la tabella iniziale.

Osservazioni sulle decomposizioni

Le proprietà delle decomposizioni di essere senza perdita e di conservare le dipendenze sono indipendenti tra loro. Una buona decomposizione dovrebbe sia essere senza perdita che conservare le dipendenze. Le tecniche che si utilizzano per ottenere queste composizioni si chiamano forme normali.

Relazione non 1NF

Tutte le relazioni che abbiamo visto fino adesso sono sempre in forma normale, allora per chiarire il concetto ho riportato un esempio di tabella dipartimento con tre attributi l'identificatore, il nome e le sedi. Nel caso della prima tupla il dipartimento ricerca è strutturato su tre sedi e qui sono state indicate tutte e tre,

- DIPARTIMENTO(id, nome, sedi)

id	nome	sedi
10	Ricerca	{Firenze, Roma, Napoli }
20	Amministrazione	{Pisa}
30	Sede centrale	{Roma}

Prima forma normale

Una relazione è in prima forma normale (1NF) se ogni attributo è definito su un dominio atomico.

- Le relazioni viste in questo corso sono sempre relazioni in prima forma normale.
- Altri modelli per basi di dati (ad esempio il modello a oggetti o il modello relazionale a oggetti) consentono la definizione di attributi su domini non atomici quali vettori, insiemi, identificatori di oggetto, ...

Seconda forma normale

Una relazione è in seconda forma normale (2NF) se non ci sono dipendenze parziali dalla chiave.

- Un attributo dipende parzialmente dalla chiave se dipende da un sottoinsieme proprio di essa.
- Attualmente il concetto di seconda forma normale non è utilizzato nella progettazione di basi dati relazionali, mentre sono ampiamente utilizzati i concetti di *terza forma normale e forma normale di Boyce Codd*.

Relazione non 2NF

- $R(forn, ind, articolo, prezzo)$

$F = \{forn \rightarrow ind, forn\ articolo \rightarrow prezzo\}$ la chiave è $(forn\ articolo)$ e ind dipende parzialmente da essa.

forn	ind	articolo	prezzo
10	Pisa	Lapis	1
10	Pisa	Penna	2
20	Firenze	Lapis	2
30	Lucca	Gomma	1
30	Lucca	Lapis	1

Terza forma normale

Dati $R(T)$ e F , R è in terza forma normale (3NF) se per ogni dipendenza funzionale $X \rightarrow A$ non banale definita su di essa, ovvero $\forall X \rightarrow A$ tale che $X \rightarrow A \in F^+$ and $NOT(A \subseteq X)$ vale una delle seguenti condizioni

- 1 X è superchiave
 - 2 A è un attributo primo (ovvero un attributo appartenente a una chiave)
-
- La definizione fa riferimento a dipendenze con a destra un solo attributo perché ogni dipendenza $X \rightarrow A_1 A_2 \dots A_k$ può essere decomposta nell'insieme $\{X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_k\}$
 - Si può dimostrare che ogni relazione 3NF è anche 2NF.

anche delle dipendenze che a sinistra non hanno una super chiave, quindi queste dipendenze possono generare un po' di ridondanza, quindi si ottiene una decomposizione che è quasi perfetta. Esempio di schemi che sono in terza forma normale, quindi che non hanno bisogno di essere decomposti: voglio dimostrare che questa relazione così com'è, senza fare nessuna decomposizione, è in terza forma normale. Questo schema ha due chiavi, città e via, perché città e via determinano cap, quindi determinano tutti gli attributi, ma anche cap e via è chiave, perché tramite cap determino città e quindi poi tramite città via. Perché è in terza forma normale questa relazione? Perché per

quindi questo è un esempio di schema non in prima forma normale perché ho un attributo per le sedi che è costituito da un insieme di valori e non da un singolo valore. (la nostra base di dati deriva da una progettazione in cui questa situazione non ci dovrebbe mai essere, perché ad esempio un'entità in cui compariva l'attributo telefono che poteva assumere più valori, prima di essere tradotta nel modello relazionale era necessario eliminare per tipo gli attributi aggiungendo o una un'entità o un'associazione uno a molti tra le tabelle). È un problema che si risolve facilmente, basta crearsi un'altra tabella in cui si assocerà l'identificatore le varie sedi e questo risolve il problema. La seconda forma normale richiede che nella relazione non ci siano dipendenze parziali dalla chiave. Esempio per chiarire il concetto: la chiave di questa relazione è costituita dalla coppia fornitori-articolo, perché tramite questa coppia di attributi riesco a determinare tutti gli altri, quindi tramite le dipendenze che ho a disposizione. Però in questo caso anche la prima dipendenza ha a sinistra un sottoinsieme della chiave perché a sinistra ho fornitore e ciò comporta che tutte le volte che compare un certo fornitore l'indirizzo viene ripetuto. Per essere in seconda forma bisogna che non ci siano dipendenze che hanno a sinistra un sottoinsieme della chiave. Terza forma normale: per ogni dipendenza x determina a che appartiene alla chiusura di f e a non è contenuta in x altrimenti sarebbe banale, vale una delle seguenti condizioni allora x è una super chiave della relazione oppure a è un attributo primo, ovvero un attributo appartenente ad una chiave. Quindi nella terza forma normale si richiede che le dipendenze funzionali soddisfino una di queste due condizioni. Se siamo nella prima ipotesi abbiamo vinto, cioè questo è la soluzione migliore che possiamo ottenere perché se ho delle dipendenze che hanno a sinistra una super chiave quelle non generano ridondanza, invece nel secondo caso ci possono essere anche delle dipendenze funzionali che a sinistra non hanno una super chiave ma a destra hanno un attributo primo, quindi in pratica rispetto alla seconda forma normale che si imponeva che non potessero esserci sottoinsiemi della chiave a sinistra qui si consente che ci possano essere sottoinsiemi della chiave a destra in qualche modo. Esiste un algoritmo che data una relazione che non è in terza forma normale ci permette sempre di decomporla in terza forma normale e la decomposizione mantiene sia i dati quindi senza perdita e mantiene anche le dipendenze. Il difetto di questa decomposizione sta proprio nella definizione perché la terza forma normale

Terza forma normale: esempi

ammette

INDIRIZZO(città, via, CAP)

$F = \{città\ via \rightarrow CAP, CAP \rightarrow città\}$

$K_1 = città\ via \quad K_2 = CAP\ via$

INDIRIZZO è in terza forma normale perché:

$città\ via \rightarrow CAP$ ha a sinistra una chiave

$CAP \rightarrow città$ ha a destra un attributo primo

IMP(codice, nome, ufficio, capo_ufficio)

$F = \{codice \rightarrow nome, codice \rightarrow ufficio, ufficio \rightarrow capo_ufficio\}$

$K = codice$

IMP non è 3NF perché la dipendenza

$ufficio \rightarrow capo_ufficio$ non verifica nessuna delle due condizioni.

Decomposizione in terza forma normale

- Una relazione non in terza forma normale presenta ridondanze e anomalie.
- Consideriamo ad esempio la relazione *IMP* vista sopra, se ci sono 10 impiegati nell'ufficio *progettazione*, il nome del *capo_ufficio* viene ripetuto 10 volte.
- In realtà forme di ridondanza *tollerate* possono essere presenti anche in 3NF: si veda, ad esempio, la dipendenza $CAP \rightarrow citta$ in *INDIRIZZO*: ogni volta che compare un certo *CAP* viene riportata anche la rispettiva *CITTA*.
- Una relazione non in terza forma normale può **sempre** essere decomposta, senza perdita e conservando le dipendenze, in relazioni in terza forma normale.
- La decomposizione di cui sopra può essere ottenuta con *l'algoritmo di sintesi*.

la prima dipendenza ho a sinistra una chiave, nel secondo caso cap non è chiave o superchiave della mia relazione ma città è un attributo primo. Altro esempio: la chiave di questa relazione è il codice perché tramite codice riesco a determinare tutti gli altri attributi, quindi mentre le prime due dipendenze verificano la condizione per la 3nf perché hanno a sinistra una super chiave, la terza dipendenza ufficio determina capo ufficio non verifica nessuna delle due perché a sinistra non ha una super chiave e a destra non ha un attributo primo, quindi questo è un esempio di schema che non è in terza forma normale per colpa dell'ultima dipendenza. Una relazione che non è in terza forma normale sicuramente presenta ridondanze e anomalie. Esiste un algoritmo di sintesi che permette di decomporre sempre una relazione in terza forma normale senza perdita e conservando le dipendenze, può succedere che ci siano ancora delle dipendenze che causano ridondanza però che sono tollerate perché comunque sono quelle che hanno a destra attributi primi. L'algoritmo prende in input una relazione e un insieme di dipendenze funzionali e in output restituisce un insieme di una decomposizione in n sottotabelle in cui ognuna è in terza forma normale. Il primo passo è quello di andare a calcolare una copertura ridotta G di F , quindi la prima cosa da fare per la decomposizione è semplificare F eliminando sia gli attributi ridondanti a sinistra sia le eventuali dipendenze ridondanti, fatta questa semplificazione dobbiamo unire, quindi applicare la regola di unione, quindi se abbiamo delle dipendenze che hanno a sinistra lo stesso insieme di attributi dobbiamo unirli, quindi andare a considerare la dipendenza che a destra ha l'unione di tutte quelle precedenti. A questo punto quindi avrò tutte dipendenze del tipo x determina y dove xy saranno degli insiemi di attributi e a questa dipendenza associo una tabella definita sull'unione degli attributi che sono in x e in y , e a questa tabella io associo la dipendenza x determinare y . Sicuramente x rappresenta la chiave della tabella per costruzione perché vado a costruire una tabella definita proprio sull'unione degli attributi che la costituiscono, quindi questo passo dell'algoritmo mi permette di trovare delle tabelle che sicuramente soddisfano la 3nf, perché li associo una dipendenza che ha a sinistra sicuramente la chiave della relazione. Il passo quattro devo fare per tutte le dipendenze che sono presenti nel mio schema questa operazione però

Algoritmo di sintesi per 3NF

Input: $R(T), F$

Output: decomposizione $R_1(T_1), \dots, R_n(T_n)$ con $R_i(T_i)$ in 3NF, la decomposizione è senza perdita e mantiene le dipendenze.

- ① Calcola una copertura ridotta G di F
- ② Sostituisci ogni insieme di dipendenze $X \rightarrow A_1, X \rightarrow A_2, X \rightarrow A_k$ con la dipendenza $X \rightarrow A_1 A_2 \dots A_k$
- ③ Per ogni dipendenza $X \rightarrow Y$ costruisci uno schema di attributi XY a cui viene associata la dipendenza $X \rightarrow Y$
- ④ Se uno schema ha gli attributi che sono un sottoinsieme proprio degli attributi di un altro schema eliminalo, ovvero, se abbiamo $R_i(T_i)$ e $R_j(T_j)$ con T_i sottoinsieme proprio di T_j , elimina $R_i(T_i)$ e aggiungi le dipendenze associate a $R_i(T_i)$ a quelle associate con $R_j(T_j)$
- ⑤ Se nessun T_i contiene una chiave dello schema $R(T)$, aggiungi alla decomposizione lo schema $R_w(W)$ con W chiave di $R(T)$.

Algoritmo di sintesi: osservazioni

- L'algoritmo ha complessità polinomiale, pari a quella per il calcolo della copertura ridotta.
- La decomposizione mantiene le dipendenze per come vengono costruiti gli schemi a partire dalle dipendenze stesse.
- Gli schemi sono 3NF:
 - ① Il passo 3 costruisce schemi per cui la dipendenza associata ha il membro sinistro che è una superchiave.
 - ② Il passo 5 aggiunge eventualmente uno schema che ha solo le dipendenze banali.
 - ③ Si può dimostrare che il passo 4 non contrasta l'essere gli schemi 3NF.
- La decomposizione è senza perdita
 - ① Si può dimostrare che in una decomposizione in n sottoschemi, se uno dei sottoschemi contiene una chiave della relazione di partenza allora la decomposizione è senza perdita.
 - ② La condizione precedente è assicurata dal passo 5 dell'algoritmo.

potrebbe succedere che a un certo punto determino due tabelle in cui una tabella è contenuta nell'altra, ovvero gli attributi di una sono contenuti negli attributi dell'altra, quindi si elimina ovviamente la tabella più piccola e si lascia solo la più grande. Le dipendenze funzionali che avevamo associato alla tabella che abbiamo eliminato vengono associate alla tabella che rimane ed è proprio in questo passo che può succedere che si introducano delle dipendenze in cui non ho più una superchiave a sinistra perché vengono fuori da un sottoinsieme degli attributi della relazione corrente. Nell'ultimo step devo controllare che almeno una delle tabelle che ho trovato contenga una chiave dello schema iniziale, perché se così non fosse vorrebbe dire che ho trovato una decomposizione in cui ho delle tabelle slegate che non mi permettono a ritroso di rifare il join, quindi se sono in questa situazione devo aggiungere un'ultima tabella definita sugli sull'insieme W che corrisponde ad una delle chiavi della tabella iniziale. Quest'ultimo passaggio che non sempre è necessario ma che mi serve e se nei passi tre o quattro ho creato delle tabelle che sono completamente disgiunte, quindi ho bisogno di crearne un'altra che mi permette poi di risalire alla tabella iniziale, quindi è il passo se volete che ci garantisce che la decomposizione sia senza perdita. La decomposizione

mantiene le dipendenze per costruzione perché ogni dipendenza la associa ad una tabella per questo è proprio per costruzione che mantengo le dipendenze. Esempi di come si applica: prima di applicare l'algoritmo di sintesi per convertire questa tabella in 3NF dobbiamo accertarci che questa tabella non sia terza forma normale, perché se è già in terza forma normale non devo fare niente, quindi il primo passo che dovete fare tutte le volte che c'è da fare una normalizzazione intanto è trovare la chiave della relazione. La chiave di questa relazione è data dalla coppia fornitore-articolo, perché la chiusura di questa coppia gli attributi mi restituisce tutto l'insieme. A questo punto che ne prima dipendenza ne la seconda soddisfano la 3NF, perché a sinistra non ho una super chiave e a destra non ho attributi

primi, quindi nessuna delle due condizioni è verificata, quindi posso applicare l'algoritmo di sintesi. Al primo step devo trovare una copertura ridotta dell'insieme, in questo caso l'insieme è già in forma ridotta (il primo step dell'algoritmo per la copertura ridotta prevede di andare a considerare dipendenze che hanno a destra soltanto un attributo, in questo caso non ho chiaramente attributi ridondanti a sinistra perché non ci sono dipendenze che hanno a sinistra più di un attributo e non ci sono dipendenze ridondanti perché se elimino una qualsiasi di queste tre non riesco poi a implicarla dalle altre che rimangono quindi è già in forma normale), quindi al passo due ho bisogno di unire le dipendenze che hanno la stessa parte sinistra, quindi in pratica ritorno al punto di partenza e considero una dipendenza alla volta, quindi alla dipendenza fornitore determina indirizzo telefono associo una tabella che vado a costruire proprio sull'unione degli attributi della dipendenza. Creo una tabella definita sulla coppia articolo e colore e di nuovo ho una tabella in terza forma normale in cui articolo è la chiave e al passo quattro non faccio niente perché non sono contenute una nell'altra, quindi le lascio tutte e due. In questa decomposizione in nessuna delle due ho la chiave fornitore articolo che era la chiave della mia relazione di partenza e ho due tabelle che sono completamente disgiunte quindi non riesco tramite il join a risalire alla tabella iniziale, quindi al passo 5 aggiungo alla decomposizione una terza tabella costituita sul insieme di attributi che costituiscono una chiave, in questo caso fornitore articolo. Un altro esempio: la chiave di questo schema è AB perché tramite AB determino C, tramite C determino D e tramite D determino B. La prima dipendenza soddisfa la condizione 3NF perché a sinistra ha la chiave e l'ultima dipendenza perché a destra ha un attributo primo, però la dipendenza C determina D non soddisfa né l'una nell'altra condizione perché non ha a sinistra una super chiave e non ha destra un attributo primo, quindi questo schema non è in terza forma normale, quindi applico l'algoritmo di sintesi: alla prima dipendenza associo la relazione R1, alla seconda dipendenza associo la relazione R2 e poi all'ultima la relazione associo R3. In questo caso la chiave è contenuta nella tabella R1, quindi non c'è nessuna relazione che è contenuta nell'altra, quindi non applico il passo quattro ma nemmeno il passo

Algoritmo di sintesi: esempio

- $R(A, B, C, D)$

$$F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow B\}$$

Chiave: (A, B)

Lo schema non è in 3NF a causa della dipendenza $C \rightarrow D$.

L'algoritmo di sintesi produce la decomposizione:

$$R_1(T_1) = R_1(A, B, C) \text{ con } F_1 = \{AB \rightarrow C\}$$

$$R_2(T_2) = R_2(C, D) \text{ con } F_2 = \{C \rightarrow D\}$$

$$R_3(T_3) = R_3(D, B) \text{ con } F_3 = \{D \rightarrow B\}$$

Si osservi che $F_2 = \pi_{T_2}(F)$, $F_3 = \pi_{T_3}(F)$ mentre

$$F_1 \neq \pi_{T_1}(F) = \{AB \rightarrow C, C \rightarrow B\}$$

5, perché la chiave AB è contenuta in R1. F2 e F3 sono gli stessi insiemi che avrei ottenuto se avessi trovato la proiezione di F sugli insiemi di attributi che definiscono le tabelle corrispondenti, se io vado a fare la proiezione di f su ABC questa proiezione contiene A determina C, ma siccome C determina D e D determina B, la proiezione contiene anche la dipendenza C determina B, perché la proiezione è definita come l'insieme delle dipendenze che sono implicate da un insieme F ma che contengono soltanto attributi nell'insieme che sto considerando, quindi la dipendenza C determina B è una dipendenza che è implicata da F e che contiene tutti attributi di T1, quindi quando faccio l'algoritmo di sintesi gli insiemi di dipendenze che associo alle relazioni in generale non corrispondono alle proiezioni dell'insieme di dipendenze funzionali su quelli insieme di attributi.

Dati $R(T)$ e F , R è in forma normale di Boyce Codd (BCNF) se per ogni dipendenza funzionale $X \rightarrow Y$ non banale definita su di essa X è superchiave.

- Ogni relazione in forma di Boyce Codd è anche in terza forma normale.
- Una relazione non BCNF presenta ridondanze e anomalie.

Algoritmo di sintesi: esempio sul passo 5

- $R(\text{fornitore}, \text{indirizzo}, \text{tel}, \text{articolo}, \text{colore})$

$$F = \{\text{fornitore} \rightarrow \text{indirizzo}, \text{tel}, \text{articolo} \rightarrow \text{colore}\}$$

Chiave: $(\text{fornitore}, \text{articolo})$

$$\textcircled{1} \quad G = \{\text{fornitore} \rightarrow \text{indirizzo}, \text{fornitore} \rightarrow \text{tel}, \text{articolo} \rightarrow \text{colore}\}$$

$$\textcircled{2} \quad G = \{\text{fornitore} \rightarrow \text{indirizzo}, \text{tel}, \text{articolo} \rightarrow \text{colore}\}$$

$$\textcircled{3} \quad R_1(\text{fornitore}, \text{indirizzo}, \text{tel}) \quad F_1 = \{\text{fornitore} \rightarrow \text{indirizzo}, \text{tel}\}$$

$$R_2(\text{articolo}, \text{colore}) \quad F_2 = \{\text{articolo} \rightarrow \text{colore}\}$$

$\textcircled{4}$ La decomposizione rimane inalterata

$\textcircled{5}$ Viene aggiunto alla decomposizione $R_3(\text{fornitore}, \text{articolo})$

In assenza del passo 5 avremmo avuto due schemi disgiunti.

Forma normale di Boyce Codd

È sicuramente senza perdita ma non sempre mantiene le dipendenze.

Forma normale di Boyce Codd: esempi -1-

ORDINE(*num, fornitore, indirizzo, tel, articolo, data, quantita*)

$$F = \{num \rightarrow fornitore\ data, fornitore \rightarrow\\ indirizzo\ tel, num\ articolo \rightarrow quantita\}$$

Chiave : *num articolo*

ORDINE non è in BCNF a causa della prima e della seconda dipendenza

ORDINE presenta ridondanze e anomalie

ORDINE non è neanche 3NF

IMPIEGATO(*codice, nome, indirizzo, qualifica, stipendio_base*)

$$F = \{codice \rightarrow nome\ indirizzo\ qualifica, qualifica \rightarrow\\ stipendio_base\}$$

Chiave: *codice*

IMPIEGATO non è BCNF (e neanche 3NF)

IMPIEGATO ha ridondanze e anomalie (per tutti gli impiegati con la stessa qualifica viene ripetuto lo stipendio base)

memorizzare lo stesso valore di stipendio.

Esempio: la chiave di questa tabella è rappresentata dalla coppia num-articolo. Questa relazione non è in BCNF per colpa della prima e della seconda dipendenza che a sinistra non hanno una super chiave e non è neanche in terza forma normale perché queste due dipendenze a destra non hanno attribuiti primi. Questo è un esempio di relazione che non è BCNF e non è 3NF, presenta ridondanze e anomalie. Un altro esempio: la chiave è codice perché tramite essa si determinano tutti gli attributi. La prima dipendenza soddisfa la BCNF ma la seconda non la soddisfa e non soddisfa neanche la terza forma normale. Ho uno schema che non è in nessuna delle due forme normali e anche questo è uno schema che presenta ridondanze e anomalie e in particolare tutte le volte che compare la qualifica se ho degli impiegati che hanno la stessa qualifica dovrò

Forma normale di Boyce Codd: esempi -2-

INDIRIZZO(*città, via, CAP*)

$$F = \{città\ via \rightarrow CAP, CAP \rightarrow città\}$$

INDIRIZZO non è BCNF

VISITE(*specializzazione, medico, data*)

$$F = \{specializzazione\ data \rightarrow medico, medico \rightarrow specializzazione\}$$

VISITE non è BCNF (si notino le ridondanze)

specializzazione	medico	data
oculista	Neri	10/12/13
oculista	Neri	13/12/13
oculista	Neri	14/12/13
otorino	Bianchi	10/12/13
oculista	Verdi	12/12/13
dentista	Rossi	10/12/13

esempio: città e via è sicuramente una chiave della relazione per cui questo schema è sicuramente in terza forma normale perché abbiamo una super chiave nella prima a sinistra e un attributo primo a destra nella seconda dipendenza, ma ovviamente la seconda dipendenza fa sì che questo schema non sia BCNF. Esempio: la chiave di questa relazione è la coppia specializzazione-data, quindi la prima dipendenza a sinistra ha una super chiave ma la seconda ha un attributo primo a destra, per cui questa tabella di nuovo non è BCNF ma è in terza forma normale. Sono presenti delle dipendenze dovute proprio alla seconda dipendenza, tutte le volte che compare un certo medico compare anche riferimento alla sua specializzazione. Da una parte ho l'algoritmo di sintesi che mi garantisce sempre sia che i dati siano mantenuti sia che le dipendenze siano mantenute però può generare ridondanze dall'altra ho un algoritmo che quando funziona mi genera una decomposizione di BCNF senza perdita ma non mi garantisce il mantenimento delle dipendenze. Ognuno di questi algoritmi ha un difetto.

Decomposizione in forma BCNF

- Ogni relazione non in forma normale di Boyce Codd può essere decomposta in relazioni BCNF senza perdita.
- Esistono relazioni non in forma normale di Boyce Codd che non è possibile decomporre in relazioni BCNF mantenendo le dipendenze.

Esempio: le possibili chiavi sono due città e strada e poi strada e cap. Per la seconda dipendenza questo schema non è di bcnf e però l'unica decomposizione senza perdita di questa tabella è quella che ho evidenziato in questo lucido, cioè quella che corrisponde alla coppia cap e città e poi alla coppia strada e cap, quindi in questo caso questa decomposizione è sicuramente una decomposizione di bcnf perché ho tutte le due relazioni in cui le dipendenze sono ovvie oppure hanno a sinistra una chiave della relazione, però rispetto alla relazione da cui sono partite ho perso la dipendenza città strada determina cap, però siccome questa è l'unica decomposizione senza perdita che si può trovare è chiaro

Decomposizione BCNF e dipendenze

INDIRIZZO(*città, strada, CAP*)

$$F = \{città\ strada \rightarrow CAP, CAP \rightarrow città\}$$

$K_1 : città\ strada$ $K_2 : strada\ CAP$

INDIRIZZO non è BCNF a causa della seconda dipendenza

$$IND_1(CAP, città), F_1 = \{CAP \rightarrow città\}$$

$$IND_2(strada, CAP), F_2 = \{\text{solo dipendenze ovvie}\}$$

Questa decomposizione mantiene i dati (è l'unica possibile decomposizione senza perdita)

Questa decomposizione non mantiene le dipendenze: abbiamo perso la dipendenza *città strada* → *CAP* che coinvolge tutti gli attributi.

Non è possibile trovare una decomposizione senza perdita che mantenga le dipendenze.

che questo ci fa capire che in queste situazioni l'algoritmo non può darmi una soluzione che non esiste.

Algoritmo di analisi per BCNF

Input: $R(T_1, F_1)$ con F_1 in forma ridotta (in realtà è sufficiente che non si abbiano dipendenze ridondanti e che tutti i membri sinistri siano non ridondanti)

Output: ρ , decomposizione di R in BCNF che preserva i dati

- ① $\rho = \{R(T_1, F_1)\}$ e $k = 1$
- ② while esiste $R_i(T_i, F_i) \in \rho$ non in BCNF per la dipendenza $X \rightarrow Y$ do
 - $k = k + 1$
 - $T_a = X^+$ e $F_a = \pi_{T_a}(F_i)$
 - $T_b = T_i - T_a \cup X$ e $F_b = \pi_{T_b}(F_i)$
 - $\rho = \rho - R_i(T_i, F_i) + \{R_i(T_a, F_a) + R_k(T_b, F_b)\}$
- end do

L'algoritmo ha complessità esponenziale, a causa del calcolo delle proiezioni delle dipendenze funzionali.

che fa sì che quella dipendenza non sia più BCNF e poi incremento il valore di K. Devo fare la decomposizione come la costruisco? La prima tabella che vado a costruire la determino andando a trovare la chiusura di x rispetto ovviamente all'insieme che sto considerando, quindi ho una relazione definita sull'insieme T_i e con le dipendenze F_i e la dipendenza x determina y che sicuramente non è in BCNF, calcolo la chiusura di x e l'associo a questo insieme di attributi, questa volta vado a fare la proiezione di F_i , quindi insieme dell'indipendenza che sto considerando su questo insieme di attributi determinato. Il secondo insieme di attributi lo ottengo per differenza dall'insieme iniziale quindi tolgo a T_1 gli attributi che ho trovato al passo precedente, e poi faccio Unione x, cioè la parte sinistra della dipendenza da cui ha avuto origine questa decomposizione la devo rimettere in x e questo è il passaggio che mi permette di avere delle tabelle che non sono disgiunte ma collegate, e di nuovo vado a fare la proiezione su questi insieme di attributi. Entra in gioco la variabile k che ho utilizzato, sono partita da una tabella e ne ho create due, quindi devo togliere la vecchia tabella e aggiungere le due nuove, quindi dal mio insieme di tabelle tolgo la tabella che ho appena esaminato e ne aggiungo due. Tolgo la tabella che ho decomposto e ne aggiungo una con lo stesso indice e poi aggiungo l'altra con indice K che mi sono portata dietro, per cui sarà il nuovo indice che devo considerare. L'importante è ricordare di rimetterci anche gli attributi della parte sinistra della dipendenza perché altrimenti le cose non tornano.

Normalizzazione-20220504 0705-1

Algoritmo decomposizione BCNF -1-

IMPIEGATO(codice, nome, ind, qualifica, stipendio_base)

$F = \{\text{codice} \rightarrow \text{nome}, \text{codice} \rightarrow \text{ind}, \text{codice} \rightarrow \text{qualifica}, \text{qualifica} \rightarrow \text{stipendio_base}\}$

- IMPIEGATO non è BCNF a causa dell'ultima dipendenza
- $R_1(\text{qualifica}, \text{stipendio_base}) \quad F_1 = \{\text{qualifica} \rightarrow \text{stipendio_base}\}$
- $R_2(\text{codice}, \text{nome}, \text{ind}, \text{qualifica}) \quad F_2 = \{\text{codice} \rightarrow \text{nome ind qualifica}\}$
- R_2 è BCNF quindi il procedimento termina

$R_1(\text{qualifica}, \text{stipendio_base}) \quad F_1 = \{\text{qualifica} \rightarrow \text{stipendio_base}\}$

$R_2(\text{codice}, \text{nome}, \text{ind}, \text{qualifica}) \quad F_2 = \{\text{codice} \rightarrow \text{nome ind qualifica}\}$

Si può verificare che questa decomposizione mantiene anche le dipendenze.

esempio: l'insieme di dipendenze è già in forma ridotta, verifichiamo se questo schema è BCNF oppure no, quindi allora la chiave di questo schema è costituita dalla coppia fornitore articolo. Le prime due dipendenze non sono BCNF perché a sinistra non hanno una super chiave e quindi si può partire o dall'una o dall'altra nell'applicazione dell'algoritmo. Siccome le due dipendenze hanno la stessa parte sinistra è chiaro che sia che scelga la prima sia che scelga la seconda devo sempre trovare la chiusura di fornitore, quindi la scelta in questo caso è ininfluente. La chiusura di fornitore mi restituisce il fornitore l'indirizzo e il telefono, quindi ho trovato la prima tabella a cui associo le dipendenze che ottengo facendo la proiezione. Per costruire il secondo insieme devo togliere dall'insieme di partenza gli attributi fornitore indirizzo telefono e poi aggiungere fornitore, e ottengo questa seconda relazione a cui associo di nuovo la proiezione delle dipendenze funzionali. Queste due relazioni sono tutte e due BCNF perché fornitore è chiave di R_1 e fornitore articolo è chiave di R_2 , quindi abbiamo due schemi in cui le

L'algoritmo di analisi: prende in input una relazione che è definita su un insieme T_1 di attributi e un insieme F_1 con F_1 in forma ridotta, quindi è sempre necessario avere a che fare con insiemi di dipendenze ridotte e a questo punto restituisce una decomposizione in BCNF che preserva i dati, quindi questo è quello che ci assicura sulla conservazione delle dipendenze. Come funziona l'algoritmo? Tutto si svolge all'interno di questo di questo ciclo e in cui vado tutte le volte ad esaminare le tabelle che si trovano quindi in questo insieme, che via via vado a modificare, finché esiste una tabella in questo schema che non è BCNF. Inizialmente avrò lo schema iniziale che sicuramente non è BCNF e devo individuare una delle dipendenze dello schema che fa sì che questo schema non sia BCNF, ce ne potrebbe essere più di una, quindi parto da una dipendenza associata al mio schema

Algoritmo decomposizione BCNF -2-

ELENCO(fornitore, indirizzo, tel, articolo, prezzo)

$F = \{\text{fornitore} \rightarrow \text{indirizzo}, \text{fornitore} \rightarrow \text{tel}, \text{fornitore articolo} \rightarrow \text{prezzo}\}$

- ELENCO non è BCNF a causa della prima dipendenza
- $R_1(\text{fornitore}, \text{indirizzo}, \text{tel}) \quad F_1 = \{\text{fornitore} \rightarrow \text{indirizzo tel}\}$
- $R_2(\text{fornitore}, \text{articolo}, \text{prezzo}) \quad F_2 = \{\text{fornitore articolo} \rightarrow \text{prezzo}\}$
- R_2 è BCNF quindi il procedimento termina

$R_1(\text{fornitore}, \text{indirizzo}, \text{tel}) \quad F_1 = \{\text{fornitore} \rightarrow \text{indirizzo tel}\}$

$R_2(\text{fornitore}, \text{articolo}, \text{prezzo}) \quad F_2 = \{\text{fornitore articolo} \rightarrow \text{prezzo}\}$

Si può verificare che questa decomposizione mantiene anche le dipendenze.

dipendenze sicuramente non generano ridondanza. Anche in questo caso il procedimento si arresta e questa è la decomposizione che algoritmo restituisce. La decomposizione ha mantenuto anche le dipendenze perché se facciamo l'unione di F1 e F2 otteniamo esattamente l'insieme F iniziale.

Esempio: la chiave è num-articolo, quindi questa tabella non è in forma BCNF per colpa di diverse dipendenze perché c'è sia la seconda ma anche la terza che a sinistra hanno degli attributi diversi quindi se scelgo una o l'altra la decomposizione viene fatta in maniera diversa, quindi prendiamo la prima quindi consideriamo la prima dipendenza che fa sì che lo schema non sia BCNF quindi devo calcolare la chiusura di fornitore che mi restituisce il fornitore, l'indirizzo e il telefono e la proiezione è semplicemente la dipendenza che stiamo considerando e in R2 metterò tutto quello che c'era nell'insieme degli attributi iniziale tolti l'indirizzo e il telefono perché il fornitore ce lo devo lasciare e quando vado a fare la proiezione delle dipendenze su questo insieme di attributi questa volta ci vanno a finire sia la prima dipendenza num articolo determina quantità sia la dipendenza num determina fornitore data perché sono costituite da tutti attributi che fanno parte di T2. A questo punto devo controllare se le due tabelle che ho trovato sono BCNF oppure no, la prima è chiaramente BCNF perché il fornitore è chiave ma nel secondo caso invece la chiave continua ad essere num-articolo ma la dipendenza dal num non soddisfa la BCNF, quindi a questo punto quello che devo fare è applicare di nuovo la decomposizione partendo dalla relazione R2 e ovviamente considerando F2 come insieme di dipendenze e quindi considero la dipendenza da num, calcolo la chiusura di num che mi restituisce num, fornitore e data con proiezione data dalla dipendenza stessa in questo caso e poi la seconda tabella è quella che ottengo andando a togliere dagli attributi il fornitore e la data perché num ce lo dobbiamo raggiungere, quindi ottengo questa terza tabella a cui vado ad associare la dipendenza num articolo determina quantità che corrisponde alla proiezione di F2 questa volta su T3. Le tabelle che ho ottenuto con questa seconda decomposizione sono tutte e due BCNF perché num è chiave di R2 e num articolo è chiave di R3. Ho determinato una decomposizione in tre tabelle e anche in questo caso la decomposizione mantiene le dipendenze perché se faccio l'unione dei tre insiemi F1, F2 e F3 ottengo un insieme equivalente a quello iniziale.

Algoritmo decomposizione BCNF -3-

ORDINE(num, fornitore, indirizzo, tel, articolo, data, quantita)

$F = \{num\ articolo \rightarrow quantita, fornitore \rightarrow indirizzo\ telefono, num \rightarrow fornitore\ data\}$

- ORDINE non è BCNF a causa della seconda dipendenza
- $R_1(fornitore, indirizzo, tel)$ $F_1 = \{fornitore \rightarrow indirizzo\ telefono\}$
- $R_2(num, fornitore, articolo, data, quantita)$ $F_2 = \{num\ articolo \rightarrow quantita, num \rightarrow fornitore\ data\}$
- R_2 non è BCNF a causa della seconda dipendenza
- $R'_2(num, fornitore, data)$ $F'_2 = \{num \rightarrow fornitore\ data\}$
- $R_3(num, articolo, quantita)$ $F_3 = \{num\ articolo \rightarrow quantita\}$

$R_1(fornitore, indirizzo, tel)$ $F_1 = \{fornitore \rightarrow indirizzo\ telefono\}$

$R_2(num, fornitore, data)$ $F_2 = \{num \rightarrow fornitore\ data\}$

$R_3(num, articolo, quantita)$ $F_3 = \{num\ articolo \rightarrow quantita\}$

Si può verificare che questa decomposizione mantiene anche le dipendenze.

Algoritmo di analisi: esempio

$R(T) = R(A, B, C, D)$

$F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow B\}$

Chiave: (A, B)

Lo schema non è in BCNF per $C \rightarrow D$ e $D \rightarrow B$.

Partendo dalla dipendenza $C \rightarrow D$ si ha:

$T_1 = C_F^+ = \{C, D, B\}$ quindi $R_1(C, D, B)$ con

$F_1 = \pi_{T_1}(F) = \{C \rightarrow D, D \rightarrow B\}$ che non è BCNF per $D \rightarrow B$

$T_2 = T - T_1 + C = \{A, C\}$ quindi $R_2(A, C)$ con

$F_2 = \{\text{dipendenze ovvie}\}$

$T_{11} = D_{F_1}^+ = \{D, B\}$ quindi $R_{11}(D, B)$ con

$F_{11} = \pi_{T_{11}}(F_1) = \{D \rightarrow B\}$

$T_{12} = T_1 - T_{11} + D = \{C, D\}$ quindi $R_{12}(C, D)$ con

$F_{12} = \pi_{T_{12}}(F_1) = \{C \rightarrow D\}$

La decomposizione non preserva le dipendenze.

Trovando nessuna dipendenza se non quelle ovvie, quindi la dipendenza AB determina C con questo primo step di decomposizione la perdo e quindi l'algoritmo va avanti decomponendo la relazione R1 ancora una volta e quindi lo faccio questa volta tramite la dipendenza D determina B che è quella che non soddisfa la condizione, quindi si calcola la chiusura di D, questa volta rispetto ad F1 e questo mi dà chiaramente soltanto D e B, quindi ho trovato una relazione costituita sulla coppia D-B con dipendenza associata D determina B, poi la seconda tabella la trovo per differenza e poi aggiungo D, quindi mi rimane soltanto C e D, a questa tabella assocero la proiezione costituita dalla dipendenza C determina D, quindi alla fine avrò una decomposizione fatta da queste tre tabelle che ho evidenziato in rosso, però in questo procedimento abbiamo perso la dipendenza AB determina C, quindi quella eventualmente deve essere gestita in maniera diversa, ci sarà un qualche controllo sui dati che mi permette di verificare che quella condizione continui ad esistere se vogliamo mantenere questo tipo di decomposizione.

Esempio: la chiave è AB, quindi questa volta le dipendenze che fanno sì che lo schema non sia BCNF sono C determina D oppure D determina B, quindi sono partite dalla prima dipendenza. Ho determinato la chiusura di C rispetto ad F che contiene C,D e B e quindi ottengo questa prima relazione con questi tre attributi e la proiezione su questi tre attributi è C determina D e D determina B. questa decomposizione che ho trovato non è BCNF perché la chiave di R1 è C, quindi c'è la dipendenza D determina B che continua a produrre ridondanza nella tabella, quindi dovrò poi applicare di nuovo la decomposizione a questo schema per quanto riguarda T2, devo togliere dagli attributi di T1 ma aggiungerci C, quindi la parte sinistra e mi rimane soltanto A e C, quindi la seconda tabella sarà costituita dalla coppia A-C e quando vado a fare la proiezione però delle mie dipendenze sulla coppia non

- La trasformazione in forma normale di Boyce e Codd preserva i dati ma non sempre garantisce la conservazione delle dipendenze.
- La trasformazione 3NF è meno forte della BCNF e quindi non offre le medesime garanzie di qualità per una relazione, accettando anche schemi con anomalie: ha però il vantaggio di essere sempre ottenibile e di mantenere sia i dati che le dipendenze.
- Una decomposizione tesa ad ottenere la 3NF produce in molti casi schemi BCNF.

$IMP(\text{codice}, \text{name}, \text{indirizzo}, \text{progetto}, \text{data_finale}, \text{budget})$

$$F = \{\text{codice} \rightarrow \text{name} \text{ indirizzo} \text{ progetto}, \text{progetto} \rightarrow \text{data_finale} \text{ budget}\}$$

La chiave è $K = \{\text{codice}\}$ e quindi lo schema non è né BCNF né 3NF.

Per 3NF si ha:

$$R_1(\text{codice}, \text{name}, \text{indirizzo}, \text{progetto}), F_1 = \{\text{codice} \rightarrow \text{name} \text{ indirizzo} \text{ progetto}\}$$

$$R_2(\text{progetto}, \text{data_finale}, \text{budget}), F_2 = \{\text{progetto} \rightarrow \text{data_finale} \text{ budget}\}$$

Per BCNF si ha:

$$T_1 = (\text{progetto})_F^+ = \{\text{progetto}, \text{data_finale}, \text{budget}\}$$

$R_1(T_1), F_1 = \{\text{progetto} \rightarrow \text{data_finale} \text{ budget}\}$ è BCNF

$$T_2 = T - T_1 \cup \{\text{progetto}\} =$$

$$\{\text{codice}, \text{name}, \text{indirizzo}, \text{progetto}\}$$

$R_2(T_2), F_2 = \{\text{codice} \rightarrow \text{name} \text{ indirizzo} \text{ progetto}\}$ è BCNF

$$F = F_1 \cup F_2$$

$IMP(\text{codice}, \text{name}, \text{indirizzo}, \text{progetto}, \text{data_finale}, \text{budget})$

$$F = \{\text{codice} \rightarrow \text{name} \text{ indirizzo}, \text{progetto} \rightarrow \text{data_finale} \text{ budget}\}$$

La chiave è $K = \{\text{codice}, \text{progetto}\}$ e quindi lo schema non è né BCNF né 3NF.

Per 3NF si ha:

$$R_1(\text{codice}, \text{name}, \text{indirizzo}), F_1 = \{\text{codice} \rightarrow \text{name} \text{ indirizzo}\}$$

$$R_2(\text{progetto}, \text{data_finale}, \text{budget}), F_2 = \{\text{progetto} \rightarrow \text{data_finale} \text{ budget}\}$$

$$R_3(\text{codice}, \text{progetto}), F_3 = \{\text{dipendenze ovvie}\}$$

Per BCNF si ha:

$$T_1 = (\text{progetto})_F^+ = \{\text{progetto}, \text{data_finale}, \text{budget}\}$$

$R_1(T_1), F_1 = \{\text{progetto} \rightarrow \text{data_finale} \text{ budget}\}$ è BCNF

$$T_2 = T - T_1 \cup \{\text{progetto}\} =$$

$$\{\text{codice}, \text{name}, \text{indirizzo}, \text{progetto}\}$$

$R_2(T_2), F_2 = \{\text{codice} \rightarrow \text{name} \text{ indirizzo}\}$ non è BCNF

$$T_{21} = (\text{codice})_F^+ = \{\text{codice}, \text{name}, \text{indirizzo}\}, F_{21} =$$

$$\{\text{codice} \rightarrow \text{name} \text{ indirizzo}\}$$
 è BCNF

$$T_{22} = T_2 - T_{21} \cup \{\text{codice}\} = \{\text{codice}, \text{progetto}\}, F_{22} = \{\text{dipendenze ovvie}\}$$
 è BCNF

$$F \equiv F_1 \cup F_{21} \cup F_{22}$$

proiezione di F su questo insieme di attributi, potrò includere solo il nome e l'indirizzo, il progetto rimane escluso. Questa seconda tabella R_2 deve essere a sua volta decomposta, quindi come calcolo la chiusura di codice che mi dà il codice, nome e l'indirizzo e trovo questa tabella definita su questo insieme di T_{21} , che è BCNF, e poi facendo la differenza ritrovo proprio l'insieme codice progetto, a cui associo la proiezione che in questo caso è l'insieme vuoto, quindi trovo anche in questo caso la stessa decomposizione e quindi è un

Focalizzare sulle differenze degli algoritmi: il primo step da fare è determinare la chiave della relazione e la chiave della relazione è codice, perché tramite codice determino tutti gli attributi del mio insieme. Questo mi permette di dire in maniera precisa che lo schema quindi non è BCNF perché ho una dipendenza da progetto che non è chiave ma non è neanche 3NF, perché a destra di questa dipendenza non ho altri attributi primi. L'applicazione dell'algoritmo 3NF: le dipendenze sono già state unite, quindi le dipendenze che avevano a sinistra la stessa parte, perciò si tratta di andare a trasformare la prima dipendenza in una relazione, quella su codice, nome, indirizzo e progetto e la seconda dipendenza nella tabella progetto, data finale e budget quindi questo è lo step tre dell'algoritmo, le due relazioni che ho ottenuto non sono una contenuta nell'altra, quindi lo step quattro lo salto. La chiave nella tabella precedente era codice e codice è contenuto in R_1 , quindi non devo fare niente. Questa è la decomposizione che l'algoritmo restituisce. Queste due tabelle in particolare sono in BCNF perché sia in R_1 che in R_2 codice è chiave di R_1 e il progetto è chiave di R_2 . Con la BCNF ottengo la stessa decomposizione ma ci arrivo in maniera diversa: parto dalla dipendenza che a sinistra non ha una super chiave, quindi calcolo la chiusura di progetto che mi dà progetto, data finale e budget e la proiezione corrisponde proprio a questa dipendenza e poi facendo la differenza troviamo codice, nome, indirizzo e progetto e quindi in questo caso ritroviamo la stessa decomposizione di prima ottenuta però facendo un procedimento diverso.

Esempio: la relazione è sempre la stessa ma cambiano le dipendenze. La chiave è costituita dalla coppia codice-progetto perché ho bisogno di tutti e due che gli attributi per determinare l'insieme totale di attributi e quindi di nuovo lo schema non è BCNF né 3NF. Per la forma 3NF: si esegue il passo 3, alla dipendenza codice, nome e indirizzo associo lo schema R_1 con attributi codice, nome e indirizzo, alla dipendenza progetto determina data finale e budget associo lo schema progetto, data finale e budget e dopodiché con questa decomposizione avrei due tabelle in cui nessuna delle due contiene la chiave iniziale della mia relazione, quindi applico il passo 5 dell'algoritmo aggiungendo una relazione R_3 costituita dalla coppia codice e progetto, aggiungo soltanto le dipendenze ovvie. Per la BCNF: in questo caso si parte o da progetto o da codice, quindi io sono partita da progetto e ho trovato la chiusura di progetto rispetto ad F e la proiezione di F su questo insieme T_1 e questo mi determina la prima tabella che è BCNF perché il progetto è chiave, poi devo fare la differenza tra tutti gli attributi meno quelli che ho appena considerato ma rimettendoci progetto e quindi determino codice, nome, indirizzo e progetto. In questo caso quando vado a fare la

esempio in cui di nuovo la decomposizione BCNF mantiene anche le dipendenze (l'unione degli insiemi F corrisponde all'insieme iniziale).

$\text{FREQ}(\text{cod_stud}, \text{name}, \text{indirizzo}, \text{cod_corso}, \text{name_corso}, \text{tipo}, \text{periodo})$

$F = \{\text{cod_stud} \rightarrow \text{name indirizzo}, \text{cod_corso} \rightarrow \text{name_corso tipo periodo}\}$

La chiave è $K = \{\text{cod_stud}, \text{cod_corso}\}$ e quindi lo schema non è né BCNF né 3NF.

Per 3NF si ha:

$R_1(\text{cod_stud}, \text{name}, \text{indirizzo}), F_1 = \{\text{cod_stud} \rightarrow \text{name indirizzo}\}$

$R_2(\text{cod_corso}, \text{name_corso}, \text{tipo}, \text{periodo}), F_2 = \{\text{cod_corso} \rightarrow \text{name_corso tipo periodo}\}$

$R_3(\text{cod_stud}, \text{cod_corso}), F_3 = \{\text{dipendenze ovvie}\}$

Per BCNF si ha:

$T_1 = (\text{cod_stud})_F^+ = \{\text{cod_stud}, \text{name}, \text{indirizzo}\}$

$R_1(T_1), F_1 = \{\text{cod_stud} \rightarrow \text{name indirizzo}\}$ è BCNF

$T_2 = T - T_1 \cup \{\text{cod_stud}\} =$

$\{\text{cod_stud}, \text{cod_corso}, \text{name_corso}, \text{tipo}, \text{periodo}\}$

$R_2(T_2), F_2 = \{\text{cod_corso} \rightarrow \text{name_corso tipo periodo}\}$ non è BCNF

$T_{21} = (\text{cod_corso})_F^+ =$

$\{\text{cod_corso}, \text{name_corso}, \text{tipo}, \text{periodo}\}, F_{21} = F_2$ è BCNF

$T_{22} = T_2 - T_{21} \cup \{\text{cod_corso}\} =$

$\{\text{cod_stud}, \text{cod_corso}\}, F_{22} = \{\text{dipendenze ovvie}\}$ è BCNF

$F \equiv F_1 \cup F_{21} \cup F_{22}$

vado a fare la proiezione mi rimane soltanto la dipendenza codice corso determina nome corso tipo e periodo. Questa seconda tabella che ho ottenuto non è BCNF perché il codice corso non è chiave della relazione, perché determina soltanto questi attributi, ho bisogno anche del codice dello studente, quindi applico di nuovo l'algoritmo sulla tabella R2, trovo la chiusura di codice corso e poi per differenza mi rimane quindi la coppia codice studente il codice corso con associate le dipendenze ovvie quindi di nuovo un'applicazione di BCNF che mantiene.

Si consideri una scuola media inferiore e lo schema

$\text{ORARIO}(\text{prof}, \text{materia}, \text{classe}, \text{giorno}, \text{ora})$

$\text{dom}(\text{classe}) = \{\text{IA, IIA, IIIA, IB, } \dots\}$

$\text{dom}(\text{giorno}) = \{\text{lunedì, martedì, } \dots\}$

$\text{dom}(\text{ora}) = \{8.30, 9.30, 10.30, 11.30, 12.30\}$

$F = \{\text{classe materia} \rightarrow \text{prof}, \text{classe giorno ora} \rightarrow \text{materia}, \text{prof giorno ora} \rightarrow \text{classe}\}$

Le possibili chiavi sono $K_1 = \{\text{classe, giorno, ora}\}$ e

$K_2 = \{\text{prof, giorno, ora}\}$

Lo schema è in 3NF dato che le dipendenze

$\text{classe giorno ora} \rightarrow \text{materia}$ e $\text{prof giorno ora} \rightarrow \text{classe}$ hanno a sinistra una chiave e la dipendenza $\text{classe materia} \rightarrow \text{prof}$ ha a destra un attributo primo.

Lo schema non è in BCNF.

$\text{ORARIO}(\text{prof}, \text{materia}, \text{classe}, \text{giorno}, \text{ora})$

$F = \{\text{classe materia} \rightarrow \text{prof}, \text{classe giorno ora} \rightarrow \text{materia}, \text{prof giorno ora} \rightarrow \text{classe}\}$

Per BCNF si ha:

$T_1 = (\text{classe, materia})_F^+ = \{\text{classe, materia, prof}\}$

$R_1(T_1), F_1 = \{\text{classe materia} \rightarrow \text{prof}\}$ è BCNF

$T_2 = T - T_1 \cup \{\text{classe, materia}\} = \{\text{classe, materia, giorno, ora}\}$

$R_2(T_2), F_2 = \{\text{classe giorno ora} \rightarrow \text{materia}\}$ è BCNF

Ma $F \neq F_1 \cup F_2$

ad applicare l'algoritmo di analisi, quindi parto dalla coppia classe-materia, calcolo la chiusura di questo insieme di attributi che mi darà classe materia e professore, quindi questa chiusura mi determina la prima la prima tabella R1 a cui associo la dipendenza stessa e questa è chiaramente BCNF e vado a fare poi la differenza per trovare la seconda tabella R2 e trovo l'insieme classe materia giorno e ora a cui associo la seconda dipendenza che è costituita proprio da questi attributi e anche questa tabella è BCNF, però in questo caso ho perso una dipendenza perché l'ultima dipendenza professore giorno determina classe non l'ho inclusa in nessuna delle tabelle che ho trovato.

Esempio. La chiave di questa relazione è costituita dalla coppia codice studente e codice corso, ho bisogno di tutti e due questi attributi per poter determinare gli altri e lo schema non è né BCNF né 3NF. L'applicazione di 3NF: alla prima dipendenza la relazione è costruita su un codice studente, nome e indirizzo, associo alla seconda la relazione definita sul codice corso, nome corso, tipo e periodo, anche in questo caso non ho trovato una relazione contenuta nell'altra, non ho la chiave in nessuna delle due relazioni, quindi dovrò aggiungere una terza tabella costruita sulla coppia di attributi cod studente e cod corso. Per quanto riguarda invece l'algoritmo di analisi: tutte e due le dipendenze non soddisfano la condizione, quindi potrei partire da una o dall'altra, io sono partito dalla prima, ho trovato quindi la chiusura di codice studente, quindi la parte sinistra della dipendenza, questo mi dà codice studente nome e indirizzo, la proiezione è costituita dalla stessa dipendenza. In questo caso perché ci sono poi le altre dipendenze che coinvolgono attributi che non determino sicuramente con gli attributi che ho a disposizione, quindi questa prima tabella che ho trovato è BCNF, perché il codice studente è la chiave della tabella R1. Per quanto riguarda R2 devo andare a fare la differenza tolgo da T gli attributi T1 e rimetto però il codice studente. In questo caso quando poi

Esempio: in questo caso lo schema è 3NF, quindi ho bisogno di comporlo in BCNF. volendo fare le cose in maniera rigorosa dovremmo accertarci che questo insieme di dipendenze sia in forma ridotta: abbiamo dipendenze che hanno a sinistra più di un attributo quindi bisogna capire se questi attributi che abbiamo a sinistra sono tutti necessari oppure se qualcuno di questi lo potrei eliminare, però si vede abbastanza facilmente che questo non è possibile perché se elimino un attributo dalle dipendenze che ho poi non riesco sicuramente a rideterminare questi vincoli, questo insieme di dipendenze è già in forma ridotta. le possibili chiavi di questa relazione sono due perché una chiave è costituita da classe, giorno e ora, perché tramite classe giorno e ora determino la materia e poi quindi tramite la prima dipendenza determino anche il professore, d'altra parte se considero professore giorno e ora anche questa formula è chiave perché tramite questa terna di attributi determino la classe e poi tramite la seconda dipendenza determino la materia, quindi ho due chiavi e lo schema questa volta è in terza forma normale perché ho la seconda e la terza dipendenza che hanno a sinistra una super chiave. lo schema non è in BCNF per colpa della prima dipendenza, quindi è da quella che partiro

- Una agenzia bancaria vuole mantenere alcuni dati relativi ai clienti e ai loro conti correnti. Un cliente può avere più conti correnti e un conto corrente può essere cointestato.
- $CLIENTE(\text{codice}, \text{name}, \text{cognome}, \text{ind}, \text{tel}, \text{num_conto}, \text{saldo})$
 $F = \{\text{codice} \rightarrow \text{name cognome ind tel}, \text{num_conto} \rightarrow \text{saldo}\}$
- La chiave è $K = \{\text{codice}, \text{num_conto}\}$ e quindi lo schema non è né BCNF né 3NF.

Per 3NF:

$$R_1(\text{codice}, \text{name}, \text{cognome}, \text{ind}, \text{tel}), F_1 = \{\text{codice} \rightarrow \text{name cognome ind tel}\}$$

$$R_2(\text{num_conto}, \text{saldo}), F_2 = \{\text{num_conto} \rightarrow \text{saldo}\}$$

$$R_3(\text{codice}, \text{num_conto}), F_3 = \{\text{dipendenze ovvie}\}$$

Per BCNF si ha:

$$T_1 = (\text{codice})_F^+ = \{\text{codice}, \text{name}, \text{cognome}, \text{ind}, \text{tel}\}$$

$$R_1(T_1), F_1 = \{\text{codice} \rightarrow \text{name cognome ind tel}\} \text{ è BCNF}$$

$$T_2 = T - T_1 \cup \{\text{codice}\} = \{\text{codice}, \text{num_conto}, \text{saldo}\}$$

$$R_2(T_2), F_2 = \{\text{num_conto} \rightarrow \text{saldo}\} \text{ non è BCNF}$$

$$T_{21} = (\text{num_conto})_F^+ = \{\text{num_conto}, \text{saldo}\}, F_{21} = F_2 \text{ è BCNF}$$

$$T_{22} = T_2 - T_{21} \cup \{\text{num_conto}\} = \{\text{num_conto}, \text{codice}\}, F_{22} = \{\text{dipendenze ovvie}\} \text{ è BCNF}$$

$$F \equiv F_1 \cup F_{21} \cup F_{22}$$

fare la conversione 3NF: abbiamo R1 e R2 che corrispondono alle due dipendenze, siccome però la chiave è costituita da codice e numero conto, ho bisogno di aggiungere una terza tabella costituita su questa coppia di attributi. Per BCNF: si comincia da una delle due dipendenze, quindi di nuovo sono partite dalla prima ho trovato la chiusura di codice e la proiezione di F su T1 e questa prima decomposizione è BCNF perché il codice è chiave di R1. In R2 mi rimangono gli attributi codice numero conto e saldo con dipendenza associata numero conto determina saldo quindi questa non è BCNF, perché il numero del conto non è chiave di questa relazione, quindi applico di nuovo l'algoritmo, calcolo la chiusura dell'attributo numero e quindi determino questa relazione costruita sull'insieme T21, fatto soltanto dal numero conto e dal saldo e poi come ultimo passo devo calcolare la differenza che mi restituisce la coppia numero conto e il codice.

Una agenzia bancaria vuole mantenere alcuni dati relativi ai clienti e ai loro conti correnti. Un cliente può avere più conti correnti ma un conto ha un solo intestatario.

$$CLIENTE(\text{codice}, \text{name}, \text{cognome}, \text{ind}, \text{tel}, \text{num_conto}, \text{saldo})$$

$$F = \{\text{codice} \rightarrow \text{name cognome ind tel}, \text{num_conto} \rightarrow \text{saldo}, \text{num_conto} \rightarrow \text{codice}\}$$

La chiave è $K = \{\text{num_conto}\}$ e quindi lo schema non è né BCNF né 3NF.

Per 3NF:

$$R_1(\text{num_conto}, \text{saldo}, \text{codice}), F_1 = \{\text{num_conto} \rightarrow \text{saldo codice}\}$$

$$R_2(\text{codice}, \text{name}, \text{cognome}, \text{ind}, \text{tel}), F_2 = \{\text{codice} \rightarrow \text{name cognome ind tel}\}$$

Per BCNF:

$$T_1 = (\text{codice})_F^+ = \{\text{codice}, \text{name}, \text{cognome}, \text{ind}, \text{tel}\}$$

$$R_1(T_1), F_1 = \{\text{codice} \rightarrow \text{name cognome ind tel}\} \text{ è BCNF}$$

$$T_2 = T - T_1 \cup \{\text{codice}\} = \{\text{codice}, \text{num_conto}, \text{saldo}\}$$

$$R_2(T_2), F_2 = \{\text{num_conto} \rightarrow \text{saldo}, \text{codice}\} \text{ è BCNF}$$

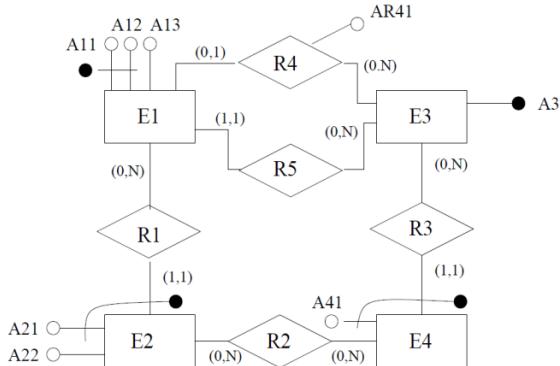
$$F = F_1 \cup F_2$$

quindi trovo la stessa decomposizione in questo caso BCNF al primo passo.

Esercizio: una descrizione di una relazione e si tratta poi di capire quali sono le dipendenze associate. Queste informazioni ora sono abbastanza imprecise, però danno un'idea di quella che potrebbe essere una relazione che mi permette di contenere queste informazioni, quindi ho bisogno di una tabella cliente che contiene il codice del cliente, il nome, il cognome, l'indirizzo, telefono, numero di conto e il saldo. Avrei bisogno di gestire le informazioni che sono in cliente. Il codice del cliente determina le caratteristiche del cliente, quindi il nome e cognome, l'indirizzo e il telefono. Dopodiché un cliente può avere più conti correnti e un conto corrente può essere cointestato, quindi se un cliente può avere più conti correnti intanto il codice non può determinare il numero del conto, perché altrimenti ce ne avrei uno solo. Un conto corrente può essere cointestato e quello che posso dire è che in corrispondenza di un certo numero di conto sicuramente determino il saldo, quello sarà sempre lo stesso, ma potrebbero esserci più clienti che hanno quel conto corrente. Da una descrizione si ricavano formalmente le dipendenze e in questo caso la chiave con queste dipendenze è costituita dalla coppia codice e numero. Lo schema quindi non è né di BCNF né 3NF, quindi andiamo a

esercizio: abbiamo un'informazione diversa ovvero un cliente può avere più conti correnti la differenza è che un conto ha un solo intestatario. nello schema cambia perché questa volta il numero del conto determina anche il codice del cliente, quindi ci sono praticamente le stesse dipendenze che avevo prima ma ho anche la dipendenza numero conto determina codice, che è quella che mi dice che è un cliente e che un conto corrente non può essere cointestato, quindi la chiave è il numero del conto perché tramite il numero del conto determino il codice del cliente e anche il suo saldo. non è né BCNF e né 3NF perché ho dipendenze, che a parte le ultime due, non hanno a sinistra una super chiave. la decomposizione 3NF: a differenza di prima in questo caso non ho la necessità di aggiungere una terza tabella perché con la decomposizione è cambiata la chiave della relazione iniziale e questa volta il numero conto è contenuto in una delle due relazioni, quindi non ho necessità di aggiungerne una terza. l'applicazione dell'algoritmo di analisi: con il calcolo della chiusura dell'attributo codice e poi con il calcolo della differenza

Tradurre lo schema ER in figura nel corrispondente schema relazionale, evidenziando le chiavi, gli eventuali attributi che possono assumere valori nulli e i vincoli di integrità:



Si ha:

E1(A11, A12, A13, A3R5, A3R4*, AR41*)

E2(A21, A11, A12, A22)

E3(A3)

E4(A41, A3)

R2(A21, A11, A12, A41, A3)

con vincoli tra E1 ed E3, E2 ed E1, E4 ed E3.

Esercizio: ci sono quattro entità e svariate associazioni che legano tra loro queste entità e ogni associazione ovviamente è descritta dalla coppia di cardinalità e per ogni entità è indicato l'identificatore e eventualmente esterno. L'obiettivo dell'esercizio è quello di tradurre questo schema nel modello relazionale utilizzando le regole che abbiamo visto. Una prima cosa che si può fare quando si deve fare questo tipo di esercizi è valutare quante saranno le tabelle che ci dobbiamo aspettare, abbiamo quattro entità quindi sappiamo che ogni entità dovrà essere tradotta in una tabella, per quanto riguarda le associazioni ce ne sono diverse di tipo uno a molti e poi ce n'è una che invece è un'associazione di tipo molti a molti. Le associazioni molti a molti di sicuro devono essere tradotte in tabelle. Questo schema deve avere almeno 5 tabelle. La prima entità diventerà una tabella che avrà come attributi A11, A12 e A13 e ovviamente la chiave della tabella sarà costituita da questa coppia A11-A12. Questa entità partecipa a due associazioni uno a molti partecipandovi con cardinalità massima 1, quindi quello che si può fare nella traduzione è portare queste associazioni dentro la tabella che corrisponde con 1, quindi l'associazione si traduce portando dentro alla tabella con l'identificatore che identifica l'entità E3, però siccome l'entità E1 partecipa con cardinalità minima zero questi due attributi li potremo indicare con l'asterisco ad indicare che possono assumere valori nulli, cioè possono esserci degli oggetti che non partecipano a questa associazione. Con uno partecipa ancora ad un'altra associazione sempre con la stessa entità però è diversa quindi anche questa dobbiamo tradurla in maniera separata, questa è un'associazione invece che è obbligatoria. L'entità E2 ha due attributi, però attenzione è identificata esternamente da E1, quindi questo che cosa vuol dire che dovremo portare dentro alla tabella E2 anche gli attributi che identificano E1, in questo modo trattiamo sia l'identificatore esterno che l'associazione R1, in questi casi si fa tutto nello stesso momento. L'associazione R2 poi la trattiamo a parte. E3 partecipa a tre associazioni con cardinalità massima N, per cui le associazioni non devono essere incluse in E3, quindi è una tabella costituita da un solo attributo, chiave di E3. In E4 dovremmo mettere l'attributo A41, ma siccome abbiamo un'identificazione esterna tramite l'associazione R3 dovremmo aggiungere anche A3 che è l'attributo che identifica E3. Manca solo da tradurre

associazioni dentro la tabella che corrisponde con 1, quindi l'associazione si traduce portando dentro alla tabella con l'identificatore che identifica l'entità E3, però siccome l'entità E1 partecipa con cardinalità minima zero questi due attributi li potremo indicare con l'asterisco ad indicare che possono assumere valori nulli, cioè possono esserci degli oggetti che non partecipano a questa associazione. Con uno partecipa ancora ad un'altra associazione sempre con la stessa entità però è diversa quindi anche questa dobbiamo tradurla in maniera separata, questa è un'associazione invece che è obbligatoria. L'entità E2 ha due attributi, però attenzione è identificata esternamente da E1, quindi questo che cosa vuol dire che dovremo portare dentro alla tabella E2 anche gli attributi che identificano E1, in questo modo trattiamo sia l'identificatore esterno che l'associazione R1, in questi casi si fa tutto nello stesso momento. L'associazione R2 poi la trattiamo a parte. E3 partecipa a tre associazioni con cardinalità massima N, per cui le associazioni non devono essere incluse in E3, quindi è una tabella costituita da un solo attributo, chiave di E3. In E4 dovremmo mettere l'attributo A41, ma siccome abbiamo un'identificazione esterna tramite l'associazione R3 dovremmo aggiungere anche A3 che è l'attributo che identifica E3. Manca solo da tradurre

Si consideri uno schema di base di dati relazionale contenente le seguenti relazioni:

INSEGNAMENTI(Codice, Denominazione)

STUDENTI(Matricola, Cognome, Nome)

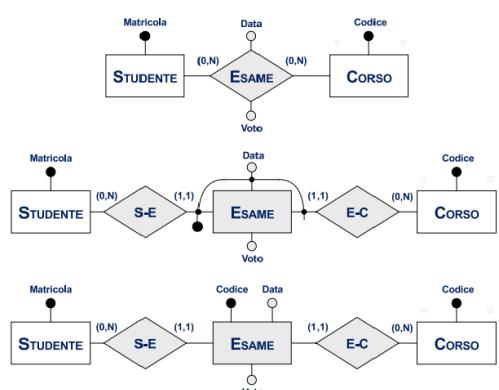
ESAMI(Studente, Corso, Data, Voto)

con vincoli di integrità referenziale

- tra l'attributo Studente della relazione ESAMI e la relazione STUDENTI
- tra l'attributo Corso della relazione ESAMI e la relazione INSEGNAMENTI.

Ricostruire una rappresentazione concettuale della basi di dati con il modello Entità-Relazione.

Si discuta inoltre come dovrebbe cambiare lo schema concettuale e quindi lo schema logico se si volesse tenere traccia anche delle bocciature (uno studente può sostenere più volte lo stesso esame)



l'associazione R2 che è di tipo molti a molti, si deve creare una tabella in cui andiamo a mettere gli identificatori delle entità che sono coinvolte ma in questo caso dobbiamo fare attenzione perché sia E2 che E4 sono identificate esternamente, è chiaro che ci dobbiamo portare dietro tutti gli attributi che costituiscono le chiavi di E2 ed E4, quindi in R2 avremo 5 attributi che corrispondono alla chiave di E2 e di E4. Devono essere inseriti dei vincoli che in questo caso sono esplicativi perché ho usato dei nomi uguali per identificare gli attributi che si corrispondono. Tutte le entità che partecipano con cardinalità massimo uno dovranno contenere dei vincoli di integrità per fare riferimento agli attributi che si corrispondono.

Esercizio: abbiamo queste tre tabelle nel primo caso abbiamo insegnamenti con attributi codice e denominazione codice rappresenta la chiave poi abbiamo studenti con attributi matricola cognome e nome e poi esami con attributi studente corso, data e voto. L'attributo studente della relazione esami corrisponde alla matricola della tabella studenti e il corso sempre nella tabella degli esami corrisponde al codice del corso nella tabella degli insegnamenti, quindi dobbiamo partire da questa descrizione e immaginare quale possa essere il modello entità relazione. Il fatto che sugli attributi di studente e corso ci siano delle chiavi, dei vincoli di integrità referenziale ci deve far venire il sospetto che la tabella esami possa derivare proprio dalla

traduzione di un'associazione che esiste tra insegnamenti e studenti e infatti la prima cosa che ci viene in mente è proprio questo schema iniziale. Abbiamo lo studente il corso e l'associazione in questo caso molti a molti con i due attributi data e voto. L'esercizio però poi chiedeva anche un'altra cosa si discute inoltre come dovrebbe cambiare lo schema concettuale e quindi lo schema logico se si volesse tenere conto anche delle bocciature quindi uno studente può sostenere più volte e lo stesso esame quindi con questo schema ER che abbiamo appena considerato ovviamente possiamo rappresentare soltanto una coppia studente-corso alla volta. Devo promuovere l'associazione in entità e quindi introduco una nuova entità esame che associo a studente e corso tramite due associazioni di tipo uno a molti e però appunto siccome voglio rappresentare un qualcosa di diverso quello che devo fare è aggiungere l'identificatore dell'entità esame, anche la data, in questo modo se esame è rappresentato esternamente da studenti e corso e dalla data. Se pensiamo alla traduzione: io andrò a tradurre le entità esame questa diventerà una tabella in cui dovrò mettere gli identificatori gli studenti e corso più la data, cioè la chiave di esame sarà costituita da questi tre attributi e non soltanto da due, questo mi permette di rappresentare tante volte la coppia studente corso al variare della data.

Sulla base di dati dell'esercizio precedente,

INSEGNAMENTI(Codice, Denominazione)

STUDENTI(Matricola, Cognome, Nome)

ESAMI(Studente, Corso, Data, Voto)

specificare in algebra relazionale le seguenti interrogazioni:

- Trovare denominazione dell'insegnamento, data e voto degli esami sostenuti dallo studente con matricola 123456789.
- Trovare la matricola degli studenti che hanno superato almeno due esami dopo il 1/1/2015.
- Trovare nome e cognome degli studenti che hanno preso il voto maggiore nell'esame relativo all'insegnamento con codice BDSI, immaginando che 30 e 1ode sia codificato con 31.
- Trovare denominazione dell'insegnamento, data e voto degli esami sostenuti dallo studente con matricola 123456789.

$\pi_{denominazione, data, voto}(\sigma_{studente=123456789}(ESAMI) \bowtie_{corso=codice} INSEGNAMENTI)$

Dato lo schema

$R(premio, titolo, giuria, vincitore)$

con

$$F = \{ \begin{aligned} & premio \rightarrow titolo \quad giuria, \quad titolo \rightarrow titolo \quad giuria, \\ & giuria \rightarrow titolo \quad vincitore, \quad vincitore \rightarrow premio \quad titolo, \\ & \quad titolo \quad giuria \rightarrow premio \quad vincitore \end{aligned}$$

si calcoli una copertura ridotta di F evidenziando i passi dell'algoritmo.

- (1) $premio \rightarrow titolo$
- (2) $premio \rightarrow giuria$
- (3) $titolo \rightarrow titolo$ è una dipendenza ovvia
- (4) $titolo \rightarrow giuria$
- (5) $giuria \rightarrow titolo$
- (6) $giuria \rightarrow vincitore$
- (7) $vincitore \rightarrow premio$
- (8) $vincitore \rightarrow titolo$
- (9) $titolo \quad giuria \rightarrow premio$
- (10) $titolo \quad giuria \rightarrow vincitore$

Trovare la matricola degli studenti che hanno superato almeno due esami dopo il 1/1/2015.

$ESAMI1 ::= \sigma_{data > 1/1/2015}(ESAMI)$

$ESAMI1(studente, corso, data, voto)$

$ESAMI2 ::= \rho_{corso2, data2, voto2 \leftarrow corso, data, voto}(ESAMI1)$

$ESAMI2(studente, corso2, data2, voto2)$

$\pi_{studente}(\sigma_{corso \neq corso2}(ESAMI1 \bowtie ESAMI2))$

Trovare nome e cognome degli studenti che hanno preso il voto maggiore nell'esame relativo all'insegnamento con codice BDSI, immaginando che 30 e 1ode sia codificato con 31.

$VOTIBDSI1 ::= \pi_{voto, studente}(\sigma_{corso=BDSI}(ESAMI))$

$VOTIBDSI1(voto, studente)$

$VOTIBDSI2 ::= \rho_{voto2, studente2 \leftarrow voto, studente}(ESAMI)$

$VOTIBDSI2(voto2, studente2)$

$MAXVOTO ::= \pi_{voto}(VOTIBDSI1)$

$- \pi_{voto}(\sigma_{voto < voto2}(VOTIBDSI1 \bowtie VOTIBDSI2))$

$\pi_{matricola, cognome}(STUDENTI \bowtie matricola=studente (VOTIBDSI1 \bowtie MAXVOTO))$

Devono essere esaminate le dipendenza (9) e (10):

(9) $titolo \quad giuria \rightarrow premio$ diventa $giuria \rightarrow premio$

(10) $titolo \quad giuria \rightarrow vincitore$ diventa $giuria \rightarrow vincitore$

Si ha $(giuria)_F^+ = \{giuria, titolo, vincitore, premio\}$ quindi $titolo$ è ridondante sia in (9) che in (10).

(1) $premio \rightarrow titolo$ si elimina per (2) e (5)

(2) $premio \rightarrow giuria$

(4) $titolo \rightarrow giuria$

(5) $giuria \rightarrow titolo$ si elimina per (6) e (8)

(6) $giuria \rightarrow vincitore$

(7) $vincitore \rightarrow premio$ si elimina per (8), (4) e (9)

(8) $vincitore \rightarrow titolo$

(9) $giuria \rightarrow premio$

(10) $giuria \rightarrow vincitore$ si elimina perché uguale a (6)

Esercizio: dobbiamo trovare una copertura ridotta di questo insieme di dipendenza utilizzando l'algoritmo che abbiamo visto allora in questo caso il consiglio è quello di cercare di essere metodici nel fare l'esercizio, quindi primo passo vi ricordo è quello di convertire tutte le dipendenze in modo tale da avere soltanto un attributo a destra, quindi applichiamo la regola di decomposizione e conviene anche numerare le dipendenze per poter poi far riferimento al numero. Facendo questo primo passo mi accorgo che c'è la dipendenza titolo determina titolo che è una dipendenza ovvia, quindi posso eliminarla. Per quanto riguarda le altre dipendenze devono essere semplificate le dipendenze che hanno a sinistra un numero di attributi maggiore di uno, allora in questo caso ci sono solo due dipendenze che hanno questa caratteristica, la 9 e la 10, quindi quello che devo fare è chiedermi se il 9 e 10 questi due attributi titolo e giuria sono tutti e due necessario oppure se uno di questi lo posso eliminare. In tutti e due casi posso eliminare l'attributo titolo sia nella dipendenza 9 sia nella dipendenza 10, questo perché calcolando la chiusura di giuria rispetto all'insieme F determino titolo, vincitore e premio, quindi vuol dire che l'attributo titolo lo posso togliere perché io riesco a determinare sia il premio che il vincitore direttamente da giuria senza bisogno di passare da titolo, quindi posso semplificare queste due dipendenze 9 e 10. L'ultimo step dell'algoritmo è quello invece in cui si chiede se qualche dipendenza può essere eliminata perché implicata dalle altre. Allora nel caso di premio determina titolo la posso eliminare perché ci sono altre due dipendenze premio determina giuria e giuria determina titolo, quindi per transitività. Giuria determina titolo la posso eliminare perché giuria determina il vincitore e il vincitore determina il titolo. Vincitore determina premio la posso eliminare perché ho una dipendenza vincitore determina titolo e titolo determina giuria e poi giuria determina premio. Infine ho questa dipendenza giuria determina vincitore che è identica alla dipendenza numero 6.

Il concetto di transazione

Transazione: parte di programma caratterizzata da un inizio, una fine e al cui interno deve essere eseguito una e una sola volta uno dei seguenti comandi:

- commit work: per terminare correttamente;
- rollback work: per abortire la transazione.

Una transazione con varie decisioni

Un sistema transazionale è in grado di definire ed eseguire transazioni per conto di un certo numero di applicazioni concorrenti.

Specifiche di transazioni in MySQL

In MySQL il supporto alle transazioni è opzionale e si ottiene definendo una tabella di tipo InnoDB. Una transazione in MySQL è una sequenza di istruzioni che parte con una istruzione START TRANSACTION; e termina con una istruzione COMMIT o ROLLBACK.

- Se si termina una istruzione con COMMIT; tutte le modifiche fatte durante la transazione vengono accettate.
- Se si termina una istruzione con ROLLBACK; tutte le modifiche vengono annullate.

Normalmente MySQL lavora in modo di autocommit e ciascuna interrogazione in esecuzione è isolata in una transazione (è come se MySQL aggiungesse START TRANSACTION e COMMIT ad ogni interrogazione). Questo significa che se si esegue un'istruzione che aggiorna o modifica una tabella, MySQL memorizza gli aggiornamenti su disco.

L'autocommit si disabilita con il comando SET AUTOCOMMIT=0; e si riporta MySQL in autocommit con SET AUTOCOMMIT=1; (la variabile AUTOCOMMIT è locale alla sessione e se ne vede il valore tramite l'istruzione SELECT @@AUTOCOMMIT).

Se si usano tabelle che non supportano le transazioni, tutte le modifiche vengono rilasciate immediatamente, indipendentemente dalla configurazione dell'autocommit.

Immaginiamo di avere un'applicazione che fa riferimento a conti correnti bancari, quindi ci saranno da gestire diversi conti correnti di più persone e quello che si vuol fare è una transazione. Conto corrente sarà una tabella e metto saldo uguale a

```
start transaction;
update ContoCorrente
set Saldo = Saldo + 10 where NumConto = 12202;
update ContoCorrente
set Saldo = Saldo-10 where NumConto = 42177;
select Saldo into A
from ContoCorrente
where NumConto = 42177;
if (A>=0) then commit work
else rollback work;
```

Consideriamo la seguente definizione di tabella:

```
CREATE TABLE t(
  id int primary key,
  val char(10)
) ENGINE =InnoDB;
```

Dopo aver fatto partire la transazione, inseriamo un valore nella tabella.

```
start transaction;
insert into t values (1,'xx');
```

Possiamo fare una SELECT dentro la transazione e vedere che il nuovo valore è stato correttamente inserito.

```
select * from t;
```

Se poi chiudiamo la transazione con COMMIT, una nuova SELECT ci assicura che il valore nuovo è ancora lì.

```
commit;
select * from t;
```

Tuttavia se chiudiamo la transazione con ROLLBACK, tutto cambia.

```
start transaction;
insert into t values (2,'yy');
select * from t;
rollback;
```

Questo si può vedere con una nuova SELECT:

```
select * from t;
```

Mentre il nuovo valore inserito è visibile all'interno della transazione, con il comando ROLLBACK tutte le modifiche all'interno della transazione vanno perse.

saldo +10 dove conto è uguale a un certo numero, contemporaneamente faccio un'altra update sempre della stessa tabella in cui vado a decrementare il valore di saldo di 10 da un altro conto corrente quindi aggiungo 10 al conto 12.202 e tolgo 10 all'altro. Quello che si fa è una selezione del saldo che è rimasto in quel conto corrente dopo questa operazione, lo metto nella variabile a e poi c'è questo controllo se a maggiore o uguale di zero, e posso completare la transazione con un commit, altrimenti faccio il rollback e annullo l'operazione perché non è possibile fare questo passaggio di denaro.

Proprietà delle transazioni

Le transazioni godono delle cosiddette proprietà acide (dall'inglese acid):

- Atomicità
- Consistenza
- Isolamento
- Durabilità (persistenza)

Atomicità

Una transazione è una unità atomica di elaborazione. Non può lasciare la base di dati in uno stato intermedio: 1 un guasto o un errore prima del commit debbono causare l'annullamento (UNDO) delle operazioni svolte; 2 un guasto o errore dopo il commit non deve avere conseguenze; se necessario vanno ripetute (REDO) le operazioni. Esito di una transazione: 1 Commit = caso normale e più frequente 2 Abort (o rollback) - richiesto dall'applicazione = suicidio - richiesto dal sistema (violazione dei vincoli, concorrenza, incertezza in caso di fallimento) = omicidio

Se si verifica un guasto o un errore prima che sia avvenuto il commit di una transazione allora tutte le operazioni che sono state fatte fino a quel momento devono essere annullate e viceversa, se si verifica un guasto dopo un commit e magari non tutte le modifiche erano state riportate su disco allora potrebbe essere necessario rifare certe operazioni per renderle di nuovo permanenti e quindi l'utilizzo delle transazioni ci garantisce che la base dei dati non rimanga mai nello stato intermedio, quindi in caso di guasti il sistema è in grado di rimettere la situazione come com'era prima che il guasto si verificasse.

Consistenza

La transazione rispetta i vincoli di integrità. Conseguenza: - se lo stato iniziale è corretto - anche lo stato finale è corretto.

Si riferisce ai vincoli di integrità che esistono nella base di dati quindi se ci sono dei vincoli e si parte da uno stato iniziale in cui questi vincoli sono rispettati la transazione ovviamente non deve fare delle operazioni che vanno a violare i vincoli quindi la consistenza fa riferimento proprio al fatto che la transazione deve rispettare i vincoli che sono stati definiti sulla base di dati.

Isolamento

La transazione non risente degli effetti delle altre transazioni concorrenti: - l'esecuzione concorrente di una collezione di transazioni deve produrre un risultato che si potrebbe ottenere con una esecuzione sequenziale. Conseguenza: una transazione non espone i suoi stati intermedi - Si evita l'effetto domino.

al fatto che più utenti, quindi più transazioni in parallelo possono far riferimento alla stessa base di dati e ogni transazione non risenta degli effetti delle altre transazioni, cioè che l'esecuzione per chi usa la base di dati deve produrre lo stesso risultato che avrei se le richieste venissero fatte in sequenze e non in modo concorrente.

Durabilità (Persistenza)

Gli effetti di una transazione andata in commit non vanno perduti, anche in presenza di guasti: - Commit significa impegno. Possibili malfunzionamenti: - i dati sul disco possono rovinarsi; - può cadere la tensione dopo che è stato effettuato un commit e prima che il sistema abbia fatto a tempo a riportare gli aggiornamenti su disco.

Fanno riferimento al fatto che gli effetti di una transazione che è andata in commit non devono essere persi qualsiasi cosa succeda e le cause per cui questo può succedere possono essere due essenzialmente il disco in cui la base di dati è memorizzato si rovina oppure ci possono essere problemi di tensione per cui durante l'esecuzione di una certa operazione perdo il lavoro fatto in questi ultimi istanti di tempo.

Transazioni e moduli di DBMS

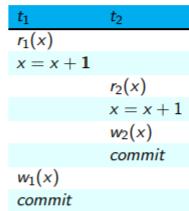
- Atomicità e durabilità - Gestore dell'affidabilità
- Isolamento - Gestore della concorrenza Consistenza - Gestore dell'integrità a tempo di esecuzione (con il supporto del compilatore del DDL).

Perdita di aggiornamento

- Due transazioni identiche:

$t_1 : r(x), x = x + 1, w(x)$
 $t_2 : r(x), x = x + 1, w(x)$

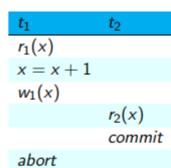
- Inizialmente $x = 2$; dopo un'esecuzione seriale $x = 4$



- Un aggiornamento viene perso: $x = 3$.

Lettura sporca

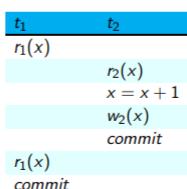
- Aspetto critico: t_2 legge uno stato intermedio sporco e lo può comunicare all'esterno.



- Il valore finale di x è 2 ma t_2 ha letto 3.

Letture inconsistenti

- La transazione t_1 legge due volte.



- La transazione t_1 legge due valori diversi per x !

l'operazione che t_1 ha fatto viene annullata, per cui t_2 ha letto il valore 3, ma in realtà il valore finale di x è 2, perché la transazione t_1 è andata in abort.

Letture inconsistenti: abbiamo due transazioni, la seconda transazione legge, modifica e scrive un certo oggetto x , poi va in commit, la prima transazione legge due volte lo stesso oggetto, una volta prima che t_2 inizi a fare le sue operazioni e poi una seconda volta dopo che t_2 ha scritto ed è andato in commit, quindi la stessa transazione leggi due volte l'oggetto x leggendo due valori diversi per quell'oggetto.

Aggiornamento fantasma: immaginiamo di avere tre oggetti x , y e z e che tra questi oggetti esista il seguente vincolo, cioè che la loro somma debba essere uguale a 1000. abbiamo due transazioni, inizia t_1 leggendo il valore di x , poi t_2 che legge il valore di y , di nuovo t_1 che legge y , quindi dopo questa lettura sia t_1 che t_2 hanno lo stesso valore di y . t_2 decrementa di 100 il valore di y e fa un'operazione analoga anche per Z , quindi aumenta di 100 il valore di Z . t_1 legge Z ma Z a questo punto è stato modificato, quindi non contiene più il valore di prima ma conterrà un valore più 100 rispetto al precedente, quindi il vincolo non sarà più 1000 ma sarà 1100, perché un oggetto è stato letto dopo che la transazione l'aveva modificata ed era andata in commit.

Inserimento fantasma: si ha quando in una nuova tupla appare improvvisamente nella base di dati. Immaginiamo di avere due transazioni, la prima transazione legge gli stipendi degli impiegati del dipartimento e calcola la media, quindi fa una media su un

Controllo di concorrenza

La concorrenza è fondamentale: decine o centinaia di transazioni al secondo, non possono essere seriali. Esempi: banche, prenotazioni aeree.

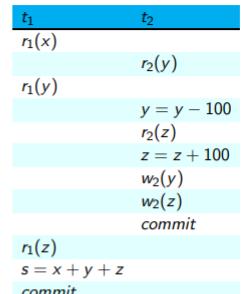
Modello di riferimento: operazioni di input-output su oggetti astratti x , y , z . Problema: anomalie causate dall'esecuzione concorrente.

Perdita di aggiornamento: immaginiamo di avere due transazioni identiche, quindi due applicazioni che fanno questa operazione. Rappresentano operazioni di lettura e scrittura nella base di dati, quindi una read r sarà una selezione semplice, una write w potrebbe essere un inserimento, un update o una cancellazione. La prima transazione legge l'oggetto x e poi lo modifica perché lo incrementa di uno, quindi poi dopo aver fatto la modifica riscrive questo dato nella base di dati. La seconda transazione fa la stessa cosa, però le due transazioni fanno queste operazioni in momenti diversi. Immaginiamo che x sia uguale a 2, se facessi queste due transazioni in modo seriale, quindi prima t_1 e poi t_2 alla fine x verrebbe 4. Immaginiamo invece che queste due transazioni siano fatte in parallelo, quindi inizia t_1 a richiedere l'oggetto x in lettura, poi incrementa i valori di x , poi prima che la transazione t_1 sia andata in commit arriva una seconda transazione che fa la stessa cosa, legge x ma x ancora vale 2, perché la prima transazione non è andata in commit, quindi incrementa x e viene scritto di nuovo il valore nella base dei dati. L'effetto di questo esempio è che viene perso un aggiornamento.

Lettura sporca: abbiamo due transazioni, t_1 che legge un certo oggetto x , lo modifica e poi scrive quell'oggetto nella base di dati. Siccome t_1 ha già scritto il valore di x , quando t_2 legge x vedrà il valore modificato. Il problema è che t_1 va in abort, quindi

Aggiornamento fantasma

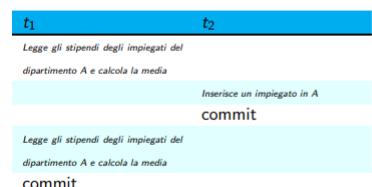
- Assumiamo che esista il vincolo $x + y + z = 1000$



- $s = 1100$: t_1 vede un aggiornamento non coerente.

Inserimento fantasma

- Una nuova tupla compare improvvisamente nella base di dati



Anomalie

- Perdita di aggiornamento: W-W
- Lettura sporca: R-W (o W-W) con abort
- Letture inconsistenti: R-W
- Aggiornamento fantasma: R-W
- Inserimento fantasma: R-W su dato nuovo

certo numero di tuple, contemporaneamente c'è una seconda transazione che inserisce un nuovo impiegato e va in commit. La prima transazione fa di nuovo la stessa operazione fatta prima, quindi legge gli stipendi e calcola la media ma è chiaro che l'inserimento del nuovo impiegato avrà come effetto che la media sarà diversa.

Livelli di isolamento in SQL

- Il livello di isolamento può essere scelto per ogni transazione:
 - **read uncommitted** permette letture sporche, letture inconsistenti, aggiornamenti fantasma e inserimenti fantasma;
 - **read committed** evita letture sporche ma permette letture inconsistenti, aggiornamenti fantasma e inserimenti fantasma;
 - **repeatable read** evita tutte le anomalie esclusi gli inserimenti fantasma: livello default per InnoDB in MySQL;
 - **serializable** evita tutte le anomalie.
- Nota: la perdita di aggiornamento è sempre evitata.
- Per vedere il livello di isolamento in MySQL:
`select @@transaction_isolation;`
 e si può modificare con set come in:
`set session transaction isolation level read committed;`

Livelli di isolamento in SQL

- La garanzia di isolamento è molto onerosa per i DBMS e per questo è prevista la possibilità di specificare un grado di protezione ridotto.
- I livelli di isolamento diversi da serializable (l'unico che garantisce i massimi requisiti di isolamento) vanno usati esclusivamente con transazioni di sola lettura.
- Quando una transazione effettua letture e scritture è opportuno usare comunque il livello di isolamento massimo.
- Per le applicazioni nei quali la correttezza dei dati è vitale, ad esempio quelle finanziarie, dovrà essere scelto il livello più elevato, mentre per applicazioni in cui la correttezza non è importante, ad esempio valutazioni statistiche in cui valori approssimati sono accettabili, potrà essere scelto un livello inferiore.

Schedule

- Sequenza di operazioni di input/output di transazioni concorrenti.
- Esempio:
 $S_1 : r_1(x)r_2(z)w_1(x)w_2(z)$
- Ipotesi semplificativa (che rinuoveremo in futuro, in quanto non accettabile in pratica):
 - consideriamo la **commit-proiezione** e ignoriamo le transazioni che vanno in abort, rimuovendo tutte le loro azioni dalla schedule.

sistemi di gestione di basi di dati permettono di ottenere dei livelli di isolamento più o meno forti a seconda delle necessità. Il problema è che mantenere questi livelli di isolamento, quindi evitare anomalie, è costoso. Capiere dal punto di vista teorico come il problema potrebbe essere risolto: immaginiamo per semplificare il problema di avere una sequenza di operazioni di input output di transazioni concorrenti ma diverse, quindi uno scheduler di operazioni (una sequenza). Facciamo inizialmente un'ipotesi, cioè quella di considerare soltanto le proiezioni che sono andate a buon fine, quindi senza considerare quelle che poi vanno eventualmente in abort e questo perché semplifica il problema. L'idea è quella di andare a individuare delle classi particolari di schedule e che sono quelli serializzabili. Dobbiamo caratterizzare gli schedule se realizzabili, allora per far questo è necessario definire i tipi di conflitti che possono verificarsi tra le transazioni. CSR sono quelli che poi nella pratica vengono utilizzati nei sistemi di gestione di basi di dati. Sono interessanti questi schedule perché innanzitutto la

In generale ho delle anomalie quando ho due transazioni che operano sullo stesso oggetto e almeno una delle due scrive, perché se tutte e due leggono non ci sono problemi. Nella perdita d'aggiornamento abbiamo due scritture, nella lettura sporca una lettura o una scrittura oppure due scritture, nelle letture inconsistenti, nell'aggiornamento fantasma e nell'inserimento fantasma abbiamo una lettura e una scrittura. È importante perché per questi tipi di conflitti possono essere definiti dei protocolli che permettono di evitarli.

Per capire come i sistemi di gestione risolvono questo problema dobbiamo ragionare in termini di sequenze di operazioni di input-output di transazioni concorrenti. Indicheremo una sequenza di azioni di scrittura e di lettura su certo numero di oggetti facendo riferimento a degli indici che ci indicano la transazione cui quell'operazione fa riferimento. L'obiettivo di una sequenza di transazioni è quello di realizzare questa sequenza in modo da avere un comportamento analogo a quello che avrei se le operazioni venissero fatte in serie.

Gestione transazioni-20220511 0702-1

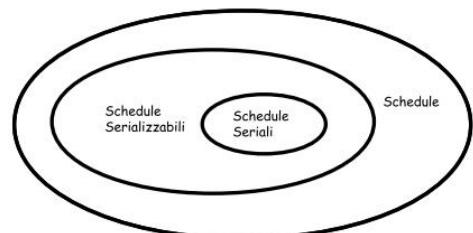
Più utenti o più applicazioni si interfacciano alla stessa base dei dati ed è necessario tenere in considerazione tutta una serie di problemi che possono verificarsi, soprattutto legati al fatto che gli utenti richiedono la stessa risorsa contemporaneamente.

Controllo di concorrenza

- Obiettivo: evitare le anomalie.
- **Scheduler:** un sistema che accetta o rifiuta (o riordina) le operazioni richieste dalle transazioni.
- **Schedule seriale:** le transazioni sono separate, una alla volta
- $S_2 : r_0(x)r_0(y)w_0(x)r_1(y)r_1(x)w_1(y)r_2(x)r_2(y)r_2(z)w_2(z)$
- **Schedule serializzabile:** produce lo stesso risultato di uno schedule seriale sulle stesse transazioni:
 - richiede una nozione di equivalenza fra schedule.

Idea base

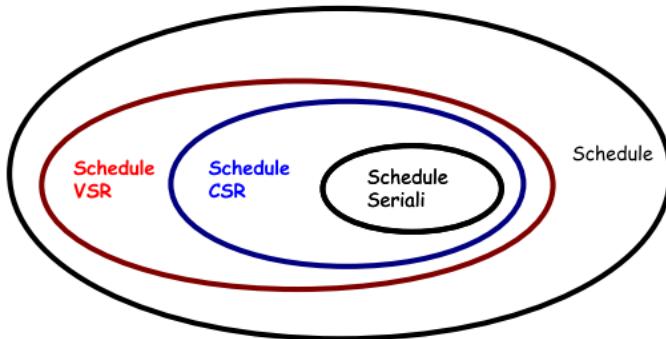
Individuare classi di schedule serializzabili che siano sottoclassi degli schedule possibili, siano serializzabili e la cui proprietà di serializzabilità sia verificabile a costo basso.



Conflict-serializzabilità

- Definizione preliminare:
 - Un'azione a_i è in **confitto** con a_j ($i \neq j$), se operano sullo stesso oggetto e almeno una di esse è una scrittura. Due casi:
 - ① conflitto **read-write** (rw o wr)
 - ② conflitto **write-write** (ww).
- **Schedule conflict-equivalenti** ($S_i \approx_C S_j$): includono le stesse operazioni e ogni coppia di operazioni in conflitto compare nello stesso ordine in entrambi.
- Uno schedule è **conflict-serializzabile** se è conflict-equivalente ad un qualche schedule seriale.
- L'insieme degli schedule conflict-serializzabili è indicato con **CSR**.

L'insieme CSR



Verifica di conflict-serializzabilità

- Per mezzo del grafo dei conflitti:

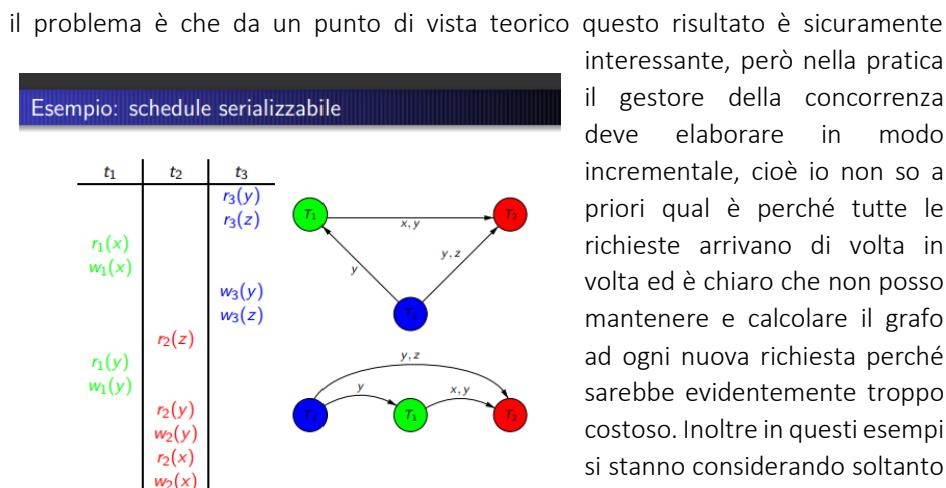
- un nodo per ogni transazione t_i ;
- un arco (orientato) da t_i a t_j se c'è almeno un conflitto fra un'azione a_i e un'azione a_j tale che a_i precede a_j .

- Teorema:** Uno schedule è in CSR se e solo se il grafo è aciclico.

CSR e aciclicità del grafo dei conflitti

- Se uno schedule S è CSR allora è \approx_C ad uno schedule seriale. Sia t_1, t_2, \dots, t_n la sequenza delle transazioni nello schedule seriale. Poiché lo schedule seriale ha tutti i conflitti nello stesso ordine dello schedule S , nel grafo di S ci possono essere solo archi (i, j) con $i < j$ e quindi il grafo non può avere cicli, perché un ciclo richiede almeno un arco (i, j) con $i > j$.
- Se il grafo di S è aciclico, allora esiste fra i nodi un ordinamento topologico (cioè una numerazione dei nodi tale che il grafo contiene solo archi (i, j) con $i < j$). Lo schedule seriale le cui transazioni sono ordinate secondo l'ordinamento topologico è equivalente a S , perché per tutti i conflitti (i, j) si ha sempre $i < j$.

nell'ordine raffigurato non ho gli stessi conflitti ma evito di avere le anomalie, perché riesco a serializzare le operazioni.



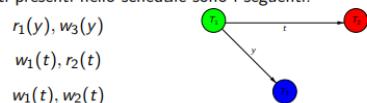
Esercizio: serializzabilità di schedule

- Sia dato lo schedule:

$r_1(x)r_1(y)r_2(z)r_3(x)w_1(t)w_2(z)r_2(x)r_2(t)r_3(y)w_2(t)$

determinare se è conflict-serializzabile.

- I conflitti presenti nello schedule sono i seguenti:



Si conclude che lo schedule è CSR.

verifica della conflict-serializzabilità può essere fatta attraverso un grafo, cioè il grafo dei conflitti.

Abbiamo tre transazioni ed è evidenziato l'ordine delle richieste di operazione indicata dalla riga in cui l'operazione si trova. Come si fa a trovare il grafo che corrisponde a questo schedule? Ogni transazione diventa un nodo nel grafo, quindi in questo caso avrò tre nodi, e poi devo disegnare un arco tra due transazioni un arco orientato se esiste un conflitto su quell'oggetto tra le due transazioni. C'è un conflitto sulla risorsa x perché t1 legge e scrive su x e poi successivamente anche t2 fa una richiesta di accesso all'oggetto x per cui disegno un arco. C'è anche un conflitto tra t2 e t1 ma in questo caso t2 prima fa la richiesta della risorsa Y e poi successivamente t1 richiede la risorsa y. Poi abbiamo un conflitto fra t2 e t3, sia su y che su z. Abbiamo un conflitto tra t3 E t1 questa volta su y e sempre per lo stesso motivo. In questo caso il grafo contiene un ciclo e quindi questo schemino non è schedule serializzabile.

Altro esempio: il grafo che si ottiene è aciclico, quindi questo schedule è serializzabile. In pratica se immaginate di avere questo grafico in cui posso muovere i nodi e gli archi e io posso trasformare questo grafo in quello sotto che è equivalente (nodi sono gli stessi e gli archi sono gli stessi), però in questo modo risulta evidente che se io eseguo le transazioni

Esempio: schedule non serializzabile

t_1	t_2	t_3
	$r_2(z)$ $r_2(y)$ $w_2(y)$	
		$r_3(y)$ $r_3(z)$
$r_1(x)$ $w_1(x)$		
	$w_3(y)$ $w_3(z)$	
	$r_2(x)$	
$r_1(y)$ $w_1(x)$		
	$w_2(x)$	

Esempio: schedule non serializzabile

t_1	t_2	t_3
	$r_2(z)$ $r_2(y)$ $w_2(y)$	
		$r_3(y)$ $r_3(z)$
$r_1(x)$ $w_1(x)$		
	$w_3(y)$ $w_3(z)$	
	$r_2(x)$	
$r_1(y)$ $w_1(x)$		
	$w_2(x)$	

Esempio: schedule serializzabile

t_1	t_2	t_3
		$r_3(y)$ $r_3(z)$
$r_1(x)$ $w_1(x)$		
	$r_2(z)$	
		$w_3(y)$ $w_3(z)$
$r_1(y)$ $w_1(y)$		
	$r_2(y)$ $w_2(y)$	
	$r_2(x)$	
		$w_2(x)$

Controllo della concorrenza in pratica

- La conflict-serializzabilità è verificabile tramite un algoritmo che con opportune strutture dati richiede tempo lineare ma è inutilizzabile in pratica.
- La tecnica sarebbe efficiente se potessimo conoscere il grafo dall'inizio, ma così non è: uno scheduler deve operare *incrementalmente*, cioè ad ogni richiesta di operazione decidere se eseguirla subito oppure fare qualcos'altro; non è praticabile mantenere il grafo, aggiornarlo e verificarne l'aciclicità ad ogni richiesta di operazione.
- Inoltre, la tecnica si basa sull'ipotesi di commit-proiezione.
- In pratica, si utilizzano tecniche che:
 - garantiscono la conflict-serializzabilità senza dover costruire il grafo;
 - non richiedono l'ipotesi della commit-proiezione.

Lock

Principio:

- tutte le letture sono precedute da **r_lock** (lock condiviso) e seguite da **unlock**;
- tutte le scritture sono precedute da **w_lock** (lock esclusivo) e seguite da **unlock**.

Quando una transazione prima legge e poi scrive un oggetto, può:

- richiedere subito un lock esclusivo;
- chiedere prima un lock condiviso e poi uno esclusivo (**lock escalation**).

Il **lock manager** riceve queste richieste dalle transazioni e le accoglie o rifiuta, sulla base della tavola dei conflitti.

Gestione dei lock

- Basata sulla tavola dei conflitti:

Richiesta	Stato risorsa		
	libero	r_loched	w_locked
r_lock	OK/r_locked	OK/r_locked	NO/w_locked
w_lock	OK/w_locked	NO/r_locked	NO/w_locked
unlock	error	OK/depends	OK/free

Un contatore tiene conto del numero di *lettori*; la risorsa è rilasciata quando il contatore scende a zero.

- Se la risorsa non è concessa, la transazione richiedente è posta in attesa (eventualmente in coda), fino a quando la risorsa non diventa disponibile.
- Il lock manager gestisce una tabella dei lock, per ricordare la situazione.

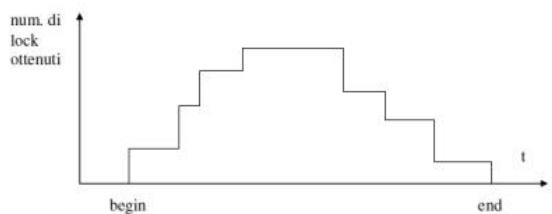
sistemi di gestione e che permette di realizzare la conflict-serializzabilità utilizzando il protocollo 2PL. Un'altra possibilità è quella di utilizzare il looking ha due fasi stretto, che prevede una condizione aggiuntiva ovvero i lock possono essere rilasciati solo dopo il commit o l'abort, quindi abbiamo questa condizione aggiuntiva rispetto alla versione precedente e quindi per costruzione con questa condizione viene superata l'ipotesi di

gestione è quello del lock. La tavola dei conflitti: se abbiamo una richiesta di lock in lettura e la risorsa è libera ovviamente la richiesta viene accettata e quell'oggetto viene emesso viene bloccato in lettura, lo stesso succede se ho una richiesta di scrittura e quell'oggetto è libero quindi la richiesta viene accettata ma ovviamente quell'oggetto viene bloccato in scrittura. Ovviamente una richiesta di unlock su un oggetto libero restituisce un errore perché non deve essere sbloccato. Che cosa succede se la richiesta è di questo tipo e invece ho su quella risorsa con un lock in lettura o in scrittura: se si richiede un lock in lettura e quell'oggetto è già bloccato in precedenza in lettura non ci sono problemi, lo schedule potrà accettare la richiesta, invece se un oggetto è stato richiesto in precedenza in scrittura e lo richiedo in lettura questo non potrà essere accettata e quindi dovrà essere messa in attesa. Un gestore deve tener conto delle richieste eventualmente metterle in coda finché la risorsa non diventa disponibile. Utilizzando la strategia appunto del lock si può ottenere la conflict-serializzabilità. In particolare utilizzando un protocollo che si chiama locking a due fasi, che si basa su due regole, allora vengono protette tutte le letture e scritture con lock, però c'è un vincolo sulle richieste e i rilasci, ovvero una transazione dopo aver rilasciato un lock non può acquisirne altri. Non è vero che un conflict-serializzabile è anche 2PL. Il fatto che 2PL implica CSR è il risultato su cui poi si basano i

Locking a due fasi: 2PL

- Usato da quasi tutti i sistemi.
- Garantisce *a priori* la conflict-serializzabilità.
- Basata su due regole:
 - proteggere tutte le letture e scritture con lock;
 - un vincolo sulle richieste e i rilasci dei lock: **una transazione, dopo aver rilasciato un lock, non può acquisirne altri**.

2PL



2PL e CSR

- Ogni schedule 2PL è anche conflict serializzabile, ma non necessariamente viceversa.
- Controesempio per la non necessità:

$$r_1(x)w_1(x)r_2(x)w_2(x)r_3(y)w_1(y)$$

- Viola 2PL: t_1 deve rilasciare il lock esclusivo su x e successivamente richiedere un lock esclusivo su y .
- Conflict-serializzabile rispetto alla sequenza t_3, t_1, t_2 .
- Sufficienza: vediamo.

2PL implica CSR

commit-proiezione. Il problema dell'affidabilità: se ci sono degli errori sia a livello software che a livello hardware, cali di tensione o il disco mi si rompe, il gestore dell'affidabilità deve garantire di poter ricostruire la base di dati e non perdere le informazioni, per poter garantire atomicità e durabilità, i sistemi utilizzano il log, ovvero un archivio permanente che registra le operazioni che sono state svolte sulla base di dati. Dobbiamo fare una distinzione tra i vari tipi di memoria che possono esistere, quindi la memoria centrale che non è persistente, la memoria di massa, che è persistente ma può danneggiarsi, e poi abbiamo bisogno anche di fare un'astrazione, cioè la memoria stabile, quindi immaginiamo di avere una memoria che non può danneggiarsi perché si tratta di un'astrazione, questo può essere perseguita attraverso ridondanza dischi replicati nastri eccetera. Il log è un file sequenziale gestito dal controllore dell'affinità del sistema ed è scritto in memoria stabile, quindi sul log possiamo fare affidamento, e in questo log andiamo a riportare tutte le operazioni che vengono fatte sulla base di dati, quindi si tratta di transazioni di una base di dati. Le possibili operazioni sono quelle che vedete elencate nel lucido, quindi il begin, quindi che dà inizio ad una transazione, e il commit che la termina, o l'abort che l'annulla, e poi le tre operazioni che sono l'inserimento, la cancellazione e l'update. Per l'inserimento ho bisogno di specificare la transazione e l'oggetto a cui faccio riferimento e AS sta per after state, cioè il valore che questo oggetto ha assunto dopo l'inserimento. Nella cancellazione ho bisogno di fare riferimento alla transazione, all'oggetto e al valore before state, quindi il valore che è stato cancellato. Se faccio un update avrò sia il valore di before state che il valore after state. Nel log possono essere memorizzate anche dei record di sistema, che sono il dump e il check point. Il checkpoint e il dump servono ad evitare che la ricostruzione in caso di guasti debba ripartire dall'inizio, cioè si usano con riferimento a tipi di guasti diversi. Il backup della base di dati viene fatto periodicamente e salva tutta la base di dati in memoria stabile. Il check point è un'operazione intermedia che serve a fare il punto della situazione fino a quel momento, in pratica ha lo scopo di evidenziare quali sono le transazioni che sono attive ancora in un certo istante e confermare che invece altre non sono iniziate oppure sono conclusive. Per poter ripristinare la situazione in caso di guasti può essere necessario disfare o rifare certe azioni. Disfare un'azione vuol dire che se si trattava di un'operazione di modifica o di cancellazione su un certo oggetto, disfarla vuol dire che io devo ricopiare sull'oggetto coinvolto il valore BS(before state) che quell'oggetto aveva prima dell'operazione. Se si trattava invece di un inserimento, disfare quell'operazione significa eliminare l'oggetto. Rifare un'azione: se si tratta di un'operazione di inserimento e di update devo copiare il valore dell'after state nell'oggetto di nuovo oppure se si trattava di un'eliminazione devo eliminare l'oggetto. Queste sono le operazioni che vengono poi fatte nel caso di guasti.

L'esito di una transazione è determinato quando viene scritto il record di commit nel file di log e in particolare se si verifica un guasto prima di questo istante allora tutte le azioni fatte fino a quel momento devono essere disfatte per ricostruire lo stato originale della base di dati, invece se si ha un guasto successivo eventualmente può essere necessario rifare certe azioni, allora un altro problema che si presenta in questo contesto è in che momento si scrive nel file di log? Ci possono essere varie strategie: si può scrivere nel log prima del database oppure si scrive il log prima del commit. Quindi quando scriviamo nella base di dati? Ci sono varie alternative, la modalità immediata: in

- S schedule 2PL.
- Consideriamo per ciascuna transazione l'istante in cui ha tutte le risorse e sta per rilasciare la prima.
- Ordiniamo le transazioni in accordo con questo valore temporale e consideriamo lo schedule seriale corrispondente.
- Vogliamo dimostrare che tale schedule è equivalente ad S: allo scopo, consideriamo un conflitto fra un'azione di t_i e un'azione di t_j con $i < j$; è possibile che compaiano in ordine invertito in S? No, perché in tal caso t_j dovrebbe aver rilasciato la risorsa in questione prima della sua acquisizione da parte di t_i .

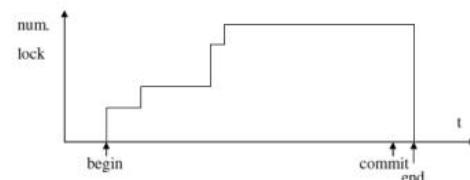
Prevenzione di un aggiornamento fantasma

t_1	t_2	x	y	z
$r_1 \text{lock}_1(x)$		free 1 : read		
$r_1(x)$			2 : write	
$r_2 \text{lock}_1(y)$				1 : wait
	$w_1 \text{lock}_2(y)$			
	$r_2(y)$			
$r_2 \text{lock}_1(z)$			$y = y - 100$	
	$w_2 \text{lock}_2(z)$			2 : write
	$r_2(z)$		$z = z + 100$	
	$w_2(z)$			
	$commit$			
	$unlock_2(y)$		1 : read	
$r_1(y)$				1 : wait
$r_1 \text{lock}_1(z)$				1 : read
	$unlock_2(z)$			
$r_1(z)$				
$s = x + y + z$				
$commit$			free	
$unlock_1(x)$				free
$unlock_1(y)$				
$unlock_1(z)$				free

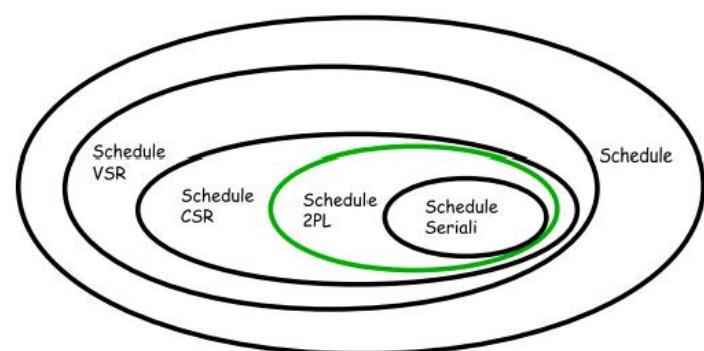
Locking a due fasi stretto

- Condizione aggiuntiva: i lock possono essere rilasciati solo dopo il commit o abort.
- Supera la necessità dell'ipotesi di commit-proiezione (ed elimina il rischio di letture sporche).

2PL stretto



CSR, VSR e 2PL



Livelli di isolamento: implementazione

- Sulle scritture si ha sempre il 2PL stretto (e quindi si evita la perdita di aggiornamento).
- **read uncommitted**: nessun lock in lettura (e non rispetta i lock altrui);
- **read committed**: lock in lettura (e rispetta quelli altrui), ma senza 2PL;
- **repeatable read**: 2PL stretto anche in lettura, con lock sulle singole tuple;
- **serializable**: 2PL stretto con lock che fanno riferimento anche a condizioni di selezione.

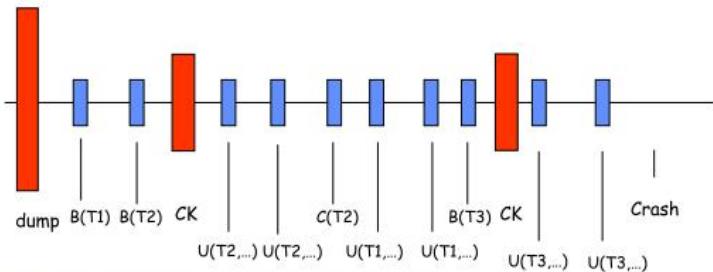
pratica in alto abbiamo la scrittura nel file di log e in basso invece abbiamo la scrittura nella base di dati e nella modalità immediata la scrittura del record log precede la scrittura sulla base di dati. In questa modalità tutte le pagine che sono modificate vengono copiate sul disco prima della scrittura del record commit sul log, il commit è stato copiato nel file di log dopo che le operazioni erano state scritte nella base di dati. Le caratteristiche di questa strategia sono che il database può contenere dei valori after state che provengono da transazioni che non sono ancora andate in commit per costruzione e quindi in questo caso è necessario fare un Undo delle transazioni che non sono andate in commit al momento del guasto e però invece non richiede il Redo perché le operazioni sono già state salvate. Un'altra strategia è la modalità differita, in pratica la scrittura del record log precede la scrittura sulla base di dati e tutte le pagine modificate vengono copiate sul disco dopo la scrittura del report commit sul log, quindi in questo caso tutte le scritture compresa quella di commit vengono fatte prima sul log e poi si va a scrivere sulla base di dati. Con questa strategia il database non contiene valori AS che provengono da transazioni che non sono conclusive e in caso di abort ovviamente non occorre fare niente e non è necessaria la procedura di Undo, però è necessaria la procedura di redo. La terza possibilità è una strategia mista in cui scritture sul database, sul disco possono avvenire in un momento qualunque rispetto alla scrittura del commit sul log. Questa modalità richiede sia le operazioni sia di undo che redo, però è possibile in questo modo ottimizzare le operazioni da parte del buffer manager.

Possono esserci due tipi di problemi, i guasti soft, quindi errori di programma, crash di sistema, cadute di tensione, quindi si perde la memoria centrale ma non si perde la memoria secondaria, in questo caso il problema può essere risolto con una ripresa a caldo. Se invece si ha un guasto hard, quindi su dispositivi di memoria secondaria, si perde la memoria secondaria ma l'importante è che non si perda la memoria stabile, quindi il file di log. Per ripristinare la situazione si può fare una ripresa a freddo. La ripresa a caldo: abbiamo un file di log in cui sono state indicate tutte le operazioni e sicuramente a un certo punto abbiamo un record di checkpoint, quindi la ripresa a caldo prevede di andare a ripercorrere il log a ritroso fino a individuare l'ultimo check point e costruire due insiemi, l'insieme undo e l'insieme redo, e si tratta poi di ripercorrere il log all'indietro fino alla più vecchia azione delle azioni che sono in questi due insiemi undo e redo, disfacendo tutte le azioni delle transazioni in undo e ripercorrendo poi in log in avanti rifacendo tutte le azioni delle transazioni in redo.

Gestore dell'affidabilità

Struttura del log

- Gestisce l'esecuzione dei comandi transazionali
 - start transaction (B, begin)
 - commit work (C)
 - rollback work (A, abort)
- e le operazioni di ripristino (recovery) dopo i guasti :
 - warm restart e cold restart.
- Assicura atomicità e durabilità
- Usa il log:
 - Un archivio permanente che registra le operazioni svolte.
 - Due metafore: il filo di Arianna e i sassolini e le briciole di Hansel e Gretel.



Persistenza delle memorie

Log, checkpoint e dump: a che cosa servono?

- Memoria centrale: non è persistente.
- Memoria di massa: è persistente ma può danneggiarsi.
- Memoria stabile: memoria che non può danneggiarsi (è una astrazione):
 - perseguita attraverso la ridondanza: dischi replicati, nastri, etc..

- Il log serve a ricostruire le operazioni.

- Checkpoint e dump servono ad evitare che la ricostruzione debba partire dall'inizio dei tempi:
 - si usano con riferimento a tipi di guasti diversi.

Il log

Undo e redo

- Il log è un file sequenziale gestito dal controllore dell'affidabilità, scritto in memoria stabile.
- Diario di bordo: riporta tutte le operazioni in ordine.
- Record nel log:
 - operazioni delle transazioni
 - begin, B(T)
 - insert, I(T,O,AS)
 - delete, D(T,O,BS)
 - update, U(T,O,BS,AS)
 - commit, C(T)
 - abort, A(T)
 - record di sistema
 - dump
 - checkpoint

- Undo di una azione su un oggetto O:

- update, delete: copiare il valore del before state (BS) nell'oggetto O
- insert: eliminare O

- Redo di una azione su un oggetto O:

- insert, update: copiare il valore dell'after state (AS) nell'oggetto O
- delete: eliminare O

- Idempotenza di undo e redo:

- undo(undo(A)) = undo(A)
- redo(redo(A)) = redo(A)

Checkpoint -1-

- Operazione che serve a *fare il punto* della situazione, semplificando le successive operazioni di ripristino: ha lo scopo di registrare quali transazioni sono attive in un certo istante (e contemporaneamente, di confermare che le altre non sono iniziate o sono finite).
- Paragone (estremo): la *chiusura dei conti* di fine anno di una amministrazione: dal 25 novembre (ad esempio) non si accettano nuove richieste di *operazioni* e si concludono tutte quelle avviate prima di accettarne di nuove.

Varie modalità, vediamo la più semplice:

- si sospende l'accettazione di richieste di ogni tipo (scrittura, inserimenti, ..., commit, abort);
- si trasferiscono in memoria di massa tutte le pagine relative a transazioni andate in commit;
- si scrive nel log un record di checkpoint che contiene gli identificatori delle transazioni in corso;
- si riprende l'accettazione delle operazioni.

Così siamo sicuri che:

- per tutte le transazioni che hanno effettuato il commit i dati sono in memoria di massa;
- le transazioni a metà strada sono elencate nel checkpoint .

Dump

- Copia completa (*di riserva, backup*) della base di dati:
 - solitamente prodotta mentre il sistema non è operativo;
 - salvata in memoria stabile;
 - un record di dump nel log indica il momento in cui il backup è stato effettuato (e dettagli pratici quali file, dispositivo, etc.).

Modalità immediata

- Schema no-redo:
 - la scrittura del record log precede la scrittura sulla base di dati;
 - tutte le pagine modificate vengono copiate su disco prima della scrittura del record commit sul log.
- Caratteristiche:
 - il DB contiene valori AS provenienti da transazioni uncommitted;
 - richiede Undo delle operazioni di transazioni uncommitted al momento del guasto;
 - non richiede Redo.

Modalità differita

- Schema no-undo:
 - la scrittura del record log precede la scrittura sulla base di dati;
 - tutte le pagine modificate vengono copiate su disco dopo la scrittura del record commit sul log.
- Caratteristiche:
 - il DB non contiene valori AS provenienti da transazioni uncommitted;
 - in caso di abort, non occorre fare niente;
 - rende superflua la procedura di Undo, richiede Redo.

Modalità mista

- Schema undo/redo:
 - le scritture sul database su disco possono avvenire in un momento qualunque rispetto alla scrittura del commit sul log.
- Caratteristiche:
 - la scrittura può avvenire in modalità sia immediata che differita;
 - consente l'ottimizzazione delle operazioni da parte del buffer manager;
 - richiede sia Undo che Redo.

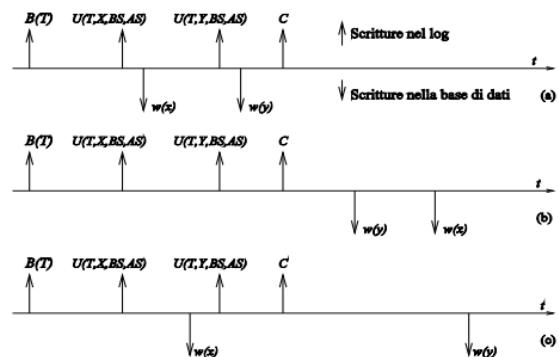
Esito di una transazione

- L'esito di una transazione è determinato irrevocabilmente quando viene scritto il record di **commit** nel log in modo sincrono:
 - una guasto prima di tale istante porta ad un **undo** di tutte le azioni, per ricostruire lo stato originario della base di dati;
 - un guasto successivo non deve avere conseguenze: lo stato finale della base di dati deve essere ricostruito, con **redo** se necessario.
- I record di **abort** possono essere scritti in modo asincrono.

Regole fondamentali per il log

- Write-Ahead-Log**: si scrive il giornale (parte before state) prima del database:
 - consente di disfare le azioni.
- Commit-Precedenza**: si scrive il giornale (parte after state) prima del commit:
 - consente di rifare le azioni.
- Quando scriviamo nella base di dati?
 - Varie alternative

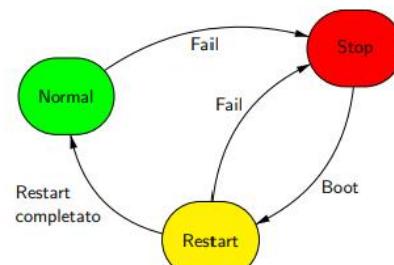
Scrittura nel log e nella base di dati



Guasti

- Guasti soft**: errori di programma, crash di sistema, caduta di tensione:
 - si perde la memoria centrale;
 - non si perde la memoria secondaria.
- Warm restart, ripresa a caldo**.
- Guasti hard**: sui dispositivi di memoria secondaria:
 - si perde anche la memoria secondaria;
 - non si perde la memoria stabile (e quindi il log).
- Cold restart, ripresa a freddo**.

Modello fail-stop



Processo di restart

- Obiettivo classificare le transazioni in:
 - completate (tutti i dati in memoria stabile);
 - in commit ma non necessariamente completate (può servire redo);
 - senza commit (vanno annullate, undo).

Ripresa a caldo

Quattro fasi:

- trovare l'ultimo checkpoint (ripercorrendo il log a ritroso);
- costruire gli insiemi UNDO (transazioni da disfare) e REDO (transazioni da rifare);
- ripercorrere il log all'indietro, fino alla più vecchia azione delle transazioni in UNDO e REDO, disfacendo tutte le azioni delle transazioni in UNDO;
- ripercorrere il log in avanti, rifacendo tutte le azioni delle transazioni in REDO.

Esempio di ripresa a caldo

Al momento del guasto il log contiene la seguente sequenza di record: si ripercorre all'indietro il log fino all'ultimo checkpoint.

B(T1)
B(T2)
U(T2, O1, B1, A1)
I(T1, O2, A2)
B(T3)
C(T1)
B(T4)
U(T3, O2, B3, A3)
U(T4, O3, B4, A4)
CK(T2, T3, T4) ⇒ UNDO={T2, T3, T4}, REDO={}
C(T4)
B(T5)
U(T3, O3, B5, A5)
U(T5, O4, B6, A6)
D(T3, O5, B7)
A(T3)
C(T5)
I(T2, O6, A8) ↪

Costruzione degli insiemi UNDO e REDO

Costruzione degli insiemi UNDO e REDO

B(T1)
B(T2)
U(T2, O1, B1, A1)
I(T1, O2, A2)
B(T3)
C(T1)
B(T4)
U(T3, O2, B3, A3)
U(T4, O3, B4, A4)
CK(T2, T3, T4)
1 C(T4)
2 B(T5)
6 U(T3, O3, B5, A5)
10 U(T5, O4, B6, A6)
5 D(T3, O5, B7)
A(T3)
3 C(T5)
4 I(T2, O6, A8)

0. UNDO={T2, T3, T4}, REDO={}
1. C(T4) ⇒ UNDO={T2, T3}, REDO={T4}
2. B(T5) ⇒ UNDO={T2, T3, T5}, REDO={T4}
3. C(T5) ⇒ UNDO={T2, T3}, REDO={T4,T5}

B(T1)
B(T2)
U(T2, O1, B1, A1)
I(T1, O2, A2)
B(T3)
C(T1)
B(T4)
U(T3, O2, B3, A3)
9 U(T4, O3, B4, A4)
CK(T2, T3, T4)
1 C(T4)
2 B(T5)
6 U(T3, O3, B5, A5)
10 U(T5, O4, B6, A6)
5 D(T3, O5, B7)
A(T3)
3 C(T5)
4 I(T2, O6, A8)

0. UNDO={T2, T3, T4}, REDO={}
1. C(T4) ⇒ UNDO={T2, T3}, REDO={T4}
2. B(T5) ⇒ UNDO={T2, T3, T5}, REDO={T4}
3. C(T5) ⇒ UNDO={T2, T3}, REDO={T4,T5}
4. D(O6)
5. O5=B7
6. O3=B5 FASE UNDO
7. O2=B3
8. O1=B1

B(T1)
B(T2)
8 U(T2, O1, B1, A1)
I(T1, O2, A2)
B(T3)
C(T1)
B(T4)
7 U(T3, O2, B3, A3)
9 U(T4, O3, B4, A4)
CK(T2, T3, T4)
1 C(T4)
2 B(T5)
6 U(T3, O3, B5, A5)
10 U(T5, O4, B6, A6)
5 D(T3, O5, B7)
A(T3)
3 C(T5)
4 I(T2, O6, A8)

B(T1)
B(T2)
8 U(T2, O1, B1, A1)
I(T1, O2, A2)
B(T3)
C(T1)
B(T4)
7 U(T3, O2, B3, A3)
9 U(T4, O3, B4, A4)
CK(T2, T3, T4)
1 C(T4)
2 B(T5)
6 U(T3, O3, B5, A5)
10 U(T5, O4, B6, A6)
5 D(T3, O5, B7)
A(T3)
3 C(T5)
4 I(T2, O6, A8)

0. UNDO={T2, T3, T4}, REDO={}
1. C(T4) ⇒ UNDO={T2, T3}, REDO={T4}
2. B(T5) ⇒ UNDO={T2, T3, T5}, REDO={T4}
3. C(T5) ⇒ UNDO={T2, T3}, REDO={T4,T5}
4. D(O6)
5. O5=B7
6. O3=B5 FASE UNDO
7. O2=B3
8. O1=B1
9. O3=A4 FASE REDO
10. O4=A6

Ripresa a freddo

- Si ripristinano i dati a partire dal backup.
- Si eseguono le operazioni registrate sul giornale fino all'istante del guasto.
- Si esegue una ripresa a caldo.

Esercizio: ripresa a caldo -1-

- Si consideri il log di un sistema di gestione di basi di dati che contiene la seguente sequenza di record.

DUMP, B(T1), I(T1,O1,A1), B(T2), I(T2,O2,A2), U(T2,O3,B3,A3), B(T3), D(T3,O4,B4), B(T4),
I(T4,O5,A5), A(T1), B(T5), I(T5,O6,A6), CK(---), C(T2), U(T5,O7,B7,A7), D(T5,O8,B8), A(T5),
U(T3,O9,B9,A9), C(T3), GUASTO

Si richiede di: 1) scrivere, in corrispondenza di ogni record di checkpoint, le transazioni attive; 2) illustrare i passi da compiere per effettuare la ripresa a caldo.

Esercizio: ripresa a caldo -2-

- ① Si individuano le transazioni attive al checkpoint, cioè quelle transazioni già iniziata e per cui non è ancora stato eseguito un commit o un abort. In questo caso, le transazioni attive al checkpoint sono T2,T3,T4,T5. Pertanto nel log si ha CK(T2,T3,T4,T5).
- ② Si determinano gli insiemi UNDO e REDO:

Record	UNDO	REDO
CK(T2,T3,T4,T5)	T2, T3, T4, T5	
C(T2)	T3, T4, T5	T2
C(T3)	T4, T5	T2, T3

Esercizio: ripresa a caldo -3-

- Si disfano le operazioni delle transazioni nell'UNDO {T4,T5}:

Record	Azione
D(T5,O8,B8)	insert O8=B8
U(T5,O7,B7,A7)	O7=B7
I(T5,O6,A6)	delete O6
I(T4,O5,A5)	delete O5

Esercizio: ripresa a caldo -4-

- Si rifanno le operazioni delle transazioni nel REDO {T2,T3}:

Record	Azione
I(T2,O2,A2)	insert O2=A2
U(T2,O3,B3,A3)	O3=A3
D(T3,O4,B4)	delete O4
U(T3,O9,B9,A9)	O9=A9