

Corso di Laurea in Ingegneria  
Informatica  
*Basi di dati*  
a.a. 2018-2019

Docente: Gigliola Vaglini  
Docente laboratorio: Francesco  
Pistolesi

1

## Principali Obiettivi del corso

- Imparare a portare a termine un buon progetto di base di dati, sia concettuale che logico.
- Imparare ad analizzare un progetto di base di dati, sia concettuale che logico, per verificarne la consistenza.
- Imparare ad impostare query per una base di dati relazionale; conoscere il modo in cui il gestore della base di dati (DBMS) esegue una query.

2

## Bibliografia e strumenti

- Lucidi lezioni ed esercitazioni del modulo sul sito del corso
  - <http://elearn.ing.unipi.it/course/view.php?id=1412>
- Atzeni, Ceri, Fraternali, Paraboschi, Torlone  
*Basi di Dati. Quinta Edizione. McGraw-Hill Italia, 2018*
- Martorini, Vaglini  
*Progettare una base di dati: dalla specifiche informali alle tabelle, Esculapio, 2011.*

3

## Comunicazione docente

- [g.vaglini@iet.unipi.it](mailto:g.vaglini@iet.unipi.it)
- Ricevimento lunedì mattina o su appuntamento
  
- [f.pistolesi@iet.unipi.it](mailto:f.pistolesi@iet.unipi.it)
- Ricevimento iscrivendosi sulla pagina  
<http://www.iet.unipi.it/f.pistolesi>

4

## Esame

- Discussione di un progetto concettuale e logico di un database relazionale con creazione della base di dati relativa e realizzazione di un set di operazioni.
- Prova pratica di scrittura di query SQL su un database già costituito
- Test scritto.

5

## Lezione 1 *Introduzione*

Queste diapositive e le successive sono state rielaborate da G. Vaglini a partire dal materiale del sito <http://www.ateneonline.it/atzeni/homeA.asp> relativo al libro “Basi di dati” - Paolo Atzeni, et al.  
Copyright © 2018 - The McGraw-Hill Companies, srl

6

## Sistema informativo

- Il sistema organizzativo è costituito da risorse e regole per lo svolgimento coordinato di attività (processi) per perseguire gli scopi propri di un'organizzazione (azienda o ente);
  - le risorse possono essere
    - persone, denaro, materiali, informazioni.
- Il sistema informativo è la componente del sistema organizzativo che acquisisce, elabora, conserva, produce le informazioni di interesse (cioè utili al perseguitamento degli scopi); inoltre esegue/gestisce i processi informativi (cioè i processi che coinvolgono informazioni)

7

## Sistemi informativi e automazione

- Il concetto di "sistema informativo" è indipendente da qualsiasi automatizzazione:
  - esistono organizzazioni la cui ragion d'essere è la gestione di informazioni (p. es. servizi anagrafici e banche) e che operano da secoli senza impiegare automatizzazioni.
- La parte del sistema informativo che gestisce informazioni con tecnologia informatica è il sistema informativo automatizzato (o sistema informatico)

8

## Le Basi di Dati

- Il cuore di un sistema informativo automatizzato è la BD, cioè un insieme organizzato di dati utilizzati per rappresentare le informazioni di interesse
- Le Basi di Dati
  - hanno dimensioni (molto) maggiori della memoria centrale dei sistemi di calcolo utilizzati
  - hanno un tempo di vita indipendente dalle singole esecuzioni dei programmi che le utilizzano (persistenza dei dati)

9

## Informazioni e...

- Un'informazione è una notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti o situazioni.
- La rappresentazione precisa (adatta ad essere elaborata) di qualsiasi forma di informazione è difficile.
- Nelle attività standardizzate dei sistemi informativi complessi, sono state introdotte col tempo forme di organizzazione e *codifica* delle informazioni
- Un'interpretazione (decodifica) permette di recuperare l'informazione originaria

10

## ..... dati

- Nei sistemi informatici, le informazioni vengono rappresentate attraverso i dati: ovvero simboli
- La rappresentazione tramite semplici simboli ha una struttura stabile nel tempo, in generale più delle procedure che vi operano; anzi nuove procedure di elaborazione ereditano i dati delle vecchie.

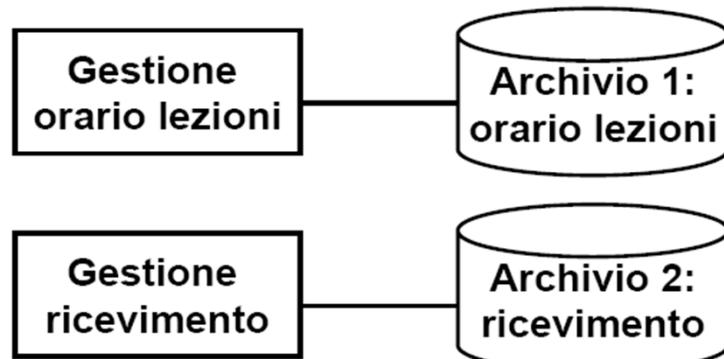
11

## Le basi di dati sono „condivise“

- Ciascun settore/attività di un’organizzazione ha un (sotto)sistema informativo
- Una base di dati è una risorsa integrata e condivisa (i vari sottosistemi informativi non sono disgiunti) fra applicazioni (al contrario dei dati privati di singoli programmi)

12

## Archivi separati



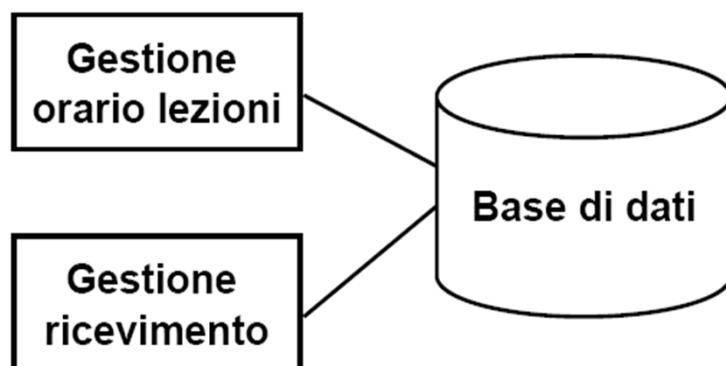
13

## Problemi

- **Informazioni ripetute**
  - Rischio di incoerenza
    - Le versioni possono non coincidere (assegnazione docenti ai corsi)
  - Nessun controllo di consistenza possibile

14

## Base di dati integrata e condivisa



15

## Vantaggi

- L'integrazione e la condivisione permettono di evitare le inconsistenze
  - Potrebbe essere anche un vantaggio per l'occupazione di memoria
- Ma chi fa i controlli??

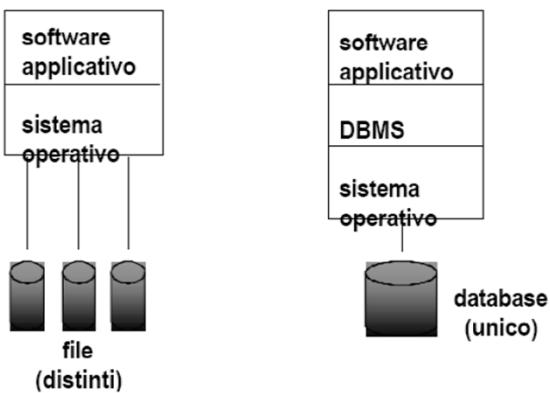
16

## Base di dati (accezione tecnologica)

- *La base di dati è un insieme organizzato di dati grandi, persistenti e condivisi gestito da un DataBase Management System (DBMS)*
- *Cosa è e dove sta?*

17

### File system e DBMS



18

## File system e DBMS

- Nei DBMS, esiste una porzione della base di dati (il catalogo o dizionario) che contiene la descrizione centralizzata (unica) dei dati, e che può essere utilizzata dai vari programmi
- I DBMS usano comunque i file per la memorizzazione dei dati, ma estendono le funzionalità dei file system, fornendo più servizi ed in maniera integrata

19

## Vantaggi e svantaggi dei DBMS

- Pro
  - gestione centralizzata con possibilità di "economia di scala"
  - disponibilità di servizi integrati
  - indipendenza dei dati (favorisce lo sviluppo e la manutenzione delle applicazioni)
- Contro
  - costo dei prodotti e della transizione verso di essi
  - non scorporabilità (spesso) delle funzionalità (con riduzione di efficienza)

20

## DataBase Management System: quali servizi?

- Il DBMS gestisce insiemi di dati (grandi, persistenti, condivisi) garantendo
  - privatezza
  - efficienza
  - efficacia
  - affidabilità

21

### I DBMS garantiscono la privatezza dei dati

- Esistono meccanismi di autorizzazione differenti per utenti differenti
  - A è autorizzato a leggere e a modificare tutti i dati
  - B è autorizzato solo a leggere il dato X mentre può leggere e/o modificare il dato Y

22

## I DBMS cercano di essere efficienti

- Cercano di utilizzare al meglio le risorse di memoria (principale e secondaria) e di ridurre il tempo di esecuzione e di risposta
- I DBMS però forniscono tante funzioni e sono quindi sempre a rischio di diventare inefficienti e per questo ci sono grandi investimenti e competizione

23

## I DBMS cercano di essere efficaci

- Forniscono agli utilizzatori funzionalità articolate, potenti e flessibili in modo da facilitare la loro produttività
- Il sistema informatico deve essere adeguatamente dimensionato e la base di dati ben progettata (e realizzata)

24

## I DBMS garantiscono l'affidabilità dei dati

- Affidabilità significa resistenza a malfunzionamenti hardware e software
- Il contenuto di una base di dati deve essere conservato intatto a lungo termine o almeno deve essere ricostruibile:
  - funzioni di backup (salvataggio) e
  - recovery (recupero)

25

## Come si garantisce l'affidabilità

- Fondamentale per garantire l' affidabilità è il concetto di transazione.
- La **transazione** è l'unità di lavoro elementare
- Tutti i DBMS (relazionali) sono sistemi transazionali, cioè mettono a disposizione un meccanismo per la definizione e l'esecuzione di transazioni

26

## Transazione: definizione

- Insieme di operazioni sulla base di dati da considerare indivisibile (atomicità), corretto anche in presenza di concorrenza (controllo della concorrenza) e con effetti definitivi (permanenza)

27

## *Atomicità*

- La sequenza di operazioni sulla base di dati viene eseguita per intero o per niente:
  - Transazione bancaria
    - trasferimento di fondi da un conto A ad un conto B: o si fanno il prelevamento da A e il versamento su B o nessuno dei due

28

## *Controllo della concorrenza*

- L'effetto dell'esecuzione di transazioni concorrenti deve essere coerente (ad esempio, "equivalente" alla loro esecuzione in sequenza: *Serializzabilità*)

29

## *Permanenza (dei risultati)*

- La conclusione positiva di una transazione corrisponde ad un impegno (in inglese *commit*) a mantenere traccia del suo risultato in modo definitivo, anche in presenza di guasti e di concorrenza

30

## Prodotti commerciali

- Prodotti software (complessi) disponibili sul mercato sono:

- Access
- DB2
- Oracle
- Informix
- Sybase
- SQLServer

31

## Descrizioni dei dati nei DBMS

- I programmi fanno riferimento ai dati, ma la loro struttura in memoria deve poter essere modificata senza dover modificare i programmi
- Viene introdotto il concetto di
  - modello logico dei dati : insieme di costrutti utilizzati per organizzare i dati di interesse e descriverne la dinamica
  - il modello dei dati fornisce ai programmi applicativi una vista astratta dei dati

32

## Modelli logici

- **Gerarchico e reticolare**

- utilizzano riferimenti esplicativi (puntatori) fra record di un file per tenere conto della strutturazione dei dati ad albero o a grafo

- **Relazionale**

- L'unico costrutto in questo modello è la relazione, che permette di definire insiemi di record omogenei a struttura fissa, tale struttura è equivalente ad una tabella
- i riferimenti fra dati in relazioni diverse sono ottenuti per mezzo dei valori stessi

33

## Schema e istanza

- Ogni tipo di dato nel modello logico ha
  - Uno schema, sostanzialmente invariante nel tempo, che ne descrive la struttura (aspetto intensionale)
    - Nel modello relazionale le intestazioni delle tabelle
  - un' istanza, i valori attuali, che possono cambiare anche molto rapidamente (aspetto estensionale)
    - Nel modello relazionale il “corpo” di ciascuna tabella

34

## *Linguaggi per basi di dati*

- La disponibilità di vari linguaggi e interfacce per la definizione di schemi e per la lettura/modifica di istanze contribuisce all'efficacia del DBMS
- Una distinzione terminologica
  - data definition language (DDL) per la definizione di schemi (logici, esterni, fisici)
  - data manipulation language (DML) per l'interrogazione e l'aggiornamento di (istanze di) basi di dati

35

## Tipi di Linguaggi per basi di dati

- linguaggi testuali interattivi (**SQL**)
- comandi (SQL) immersi in un linguaggio ospite (Pascal, Java, C ...)
- con interfacce amichevoli (senza linguaggio testuale come Access)

36

## Architettura del DBMS

- Abbiamo parlato del modello dei dati usato dai DBMS
- Parliamo adesso del modello di esecuzione, usato da un sistema in cui esiste un DBMS.

37

## Architetture distribuite

- Quasi tutte le realizzazioni di DBMS considerano un'architettura distribuita, caratterizzata dalla presenza di una rete che collega almeno due macchine che lavorano autonomamente, ma sono anche in grado di interagire
- Architettura client-server: è la più semplice e diffusa

38

## Architettura client-server

- Modello di interazione in cui i processi software si dividono in Client e Server.
  - Client
    - Richiedono i servizi
    - Dedicati a interagire con l'utente finale
    - Ruolo attivo: genera richieste
  - Server
    - Offrono i servizi
    - Ruolo reattivo: si limita a rispondere alle richieste dei diversi client
  - L'interazione fra client e server richiede una interfaccia che è l'elenco dei servizi messi a disposizione dal server

39

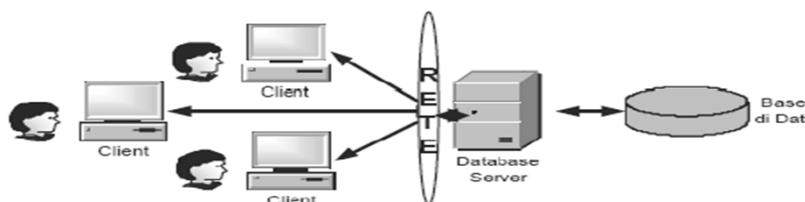
## Architettura client-server

- Processi client e server potrebbero girare sulla stessa macchina, ma
- in genere ogni processo risiede su una macchina diversa, collegata alle altre via rete
- Il server può essere unico, ma se ne esiste più di uno per realizzare una funzionalità si ha una architettura completamente distribuita

40

## Architettura client-server: esigenze HW diverse

- Un processo client può richiedere servizi a vari processi server. Il client è un elaboratore adatto alla interazione con l'utente
  - Strumenti di produttività
  - Applicazioni "amichevoli" che accedono alla base dati
- Ogni processo server risponde a (molte) richieste da parte di molti processi client gestendo in modo opportuno le relative transazioni. Il server è dimensionato in base ai servizi che deve offrire e al carico transazionale (grande memoria centrale, grande memoria di massa, ...)



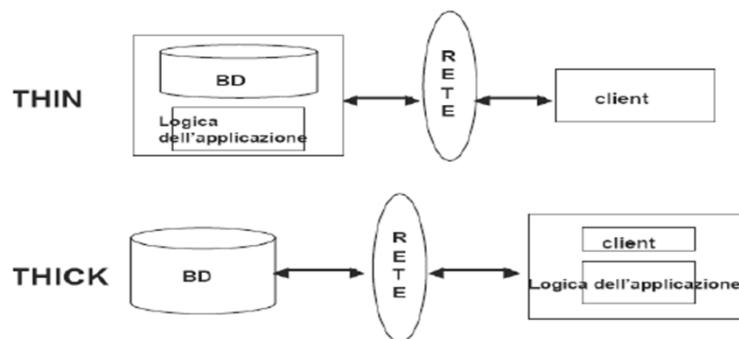
41

## Architettura two tier

- L'architettura client-server è detta anche a due livelli (*two tier*)
- Le macchine del livello Client non sono necessariamente poco complesse

42

## Struttura del client



Più diffusa l'architettura thin client

43

## Svantaggi thick client

- È necessaria fiducia tra il server e i client (i dati vengono trasmessi al client)
- Non scalabile (non più di poche centinaia di client):
  - richiede anche una altissima capacità di elaborazione da parte del server per trasmettere i dati ai client

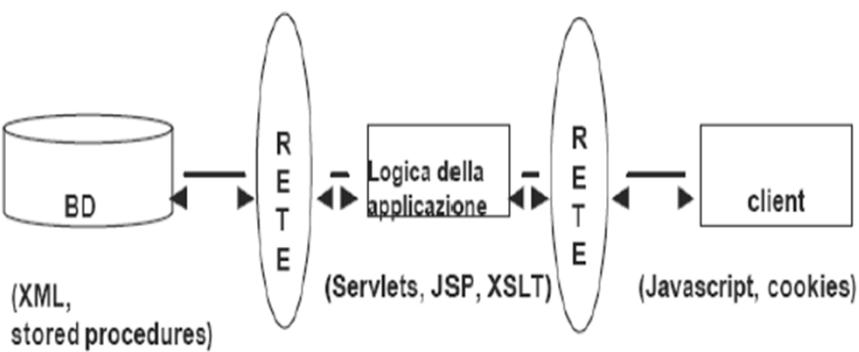
44

## Architettura a 3 livelli

- Nell'architettura three-tier è presente un secondo server, il server applicativo, responsabile di gestire la logica applicativa comune a più client
- Il client è più semplice e si occupa solo dell'interfacciamento con l'utente finale (thin client)
  - Il client invia le richieste al server applicativo
  - Il server applicativo dialoga con il server per la gestione dei dati
- Questa architettura è predisposta per Internet  
**CLIENT = BROWSER WEB**

45

## Architetture three tier



46

## Vantaggi three tier

- Si possono interconnettere sistemi eterogenei
- I client sono thin (browser web)
- Scalabilità sul numero di client: si aggiunge un server (si possono appropriatamente moltiplicare anche le macchine a livello intermedio)

47

## Big Data

- Già nel 1975 si cominciò a parlare di *Very Large Data Bases*, recentemente è stato introdotto nell'uso comune il termine *big data*.
- Da un lato la produzione di dati ha raggiunto livelli straordinari (grazie alle più svariate fonti, ad esempio: la telefonia, i sensori, le reti sociali, i dati sperimentali della fisica e della medicina), dall'altro sono stati introdotti strumenti informatici sempre più evoluti in grado di memorizzare, interrogare ed analizzare questi dati.

48

## I Big Data e le quattro V

- **Volume.** È la principale caratteristica dei *big data*. Lo stesso quantitativo di dati prodotto dall'inizio dell'umanità ad oggi sarà a breve generato in un minuto. In un minuto, ad esempio, si generano 2,3 milioni di interrogazioni a Google, 3 milioni di like e 3 milioni di share su Facebook; 2,7 milioni di video vengono scaricati da Youtube (e 139 mila nuovi video vengono caricati); 44 milioni di messaggi vengono processati, di cui 486 mila contenenti foto e 70 mila contenenti video (dati al 25/7/2017).

49

## Le quattro V (2)

- **Velocità.** Ossia la velocità con cui i dati vengono generati e scambiati.
- Tecnologie specifiche di analisi di dati in memoria e tecniche di *data streaming* consentono di analizzare i dati mentre fluiscono verso il sistema di gestione, spesso evitando di memorizzare i dati nel database dove vengono memorizzati solo i risultati dell'analisi.

50

## Le quattro V (3)

- **Varietà.** In passato l'attenzione principale era rivolta ai dati strutturati, tipicamente memorizzati in tabelle, oggi la maggior parte dei dati sono non strutturati (testi, immagini, voci, video).
- Tecnica fondamentale per gestire al meglio la varietà dei dati è la *data integration*.

51

## Le quattro V (4)

- **Veridicità.** E' possibile estrarre informazioni vere dai dati, nonostante essi contengano gravi errori, imprecisioni e incompletezze.
- Si parla di *data quality* come tecnica per gestire la (mancanza di) veridicità che caratterizza molte raccolte di dati.

52

## La scienza dei dati

- La *Data Science* è tipicamente associata ai *Big Data* ed è una materia interdisciplinare, tra la statistica e l'informatica,
- Una definizione abbastanza completa di data science include almeno i seguenti aspetti.

53

## Data science (1)

- *Data cleaning*: ha per obiettivo la costruzione di una raccolta dati che abbia un sufficiente livello di qualità.
- *Data integration*: ha per obiettivo la costruzione di una raccolta dati integrata a partire da differenti sorgenti dati. Questo è un problema che spesso richiede soluzioni ad hoc.

54

## Data science (2)

- *Data mining*, ovvero l'estrazione di informazione utile dai dati. Nel data mining sono molto utilizzate le regole di associazione, che consentono di dire quante volte, nel contesto di una specifica operazione (ad esempio, una transazione di acquisto) sono coinvolte le stesse istanze di dati (ad esempio, gli stessi oggetti).

55

## Data Science (3)

- *Metodi predittivi*. Ovvero metodi che consentono di prevedere, a partire da osservazioni nel passato, i dati caratterizzanti uno scenario futuro oppure alternativo. Tecniche statistiche consentono di selezionare i dati più utili (*feature selection*). Le predizioni vengono associate ad una probabilità.

56

## Data science (4)

- Metodi di apprendimento automatico o *machine learning*. Sono particolari metodi predittivi per classificare i dati a partire da esempi di classificazione. Alcuni dati (*training set*) sono classificati a priori (ad esempio, da un esperto) e associati ad una etichetta (*label*) che ne indica la classe.
- Il sistema di machine learning apprende come classificare gli altri dati a partire dal training set.

57

## Data science (4) cont.

- Tra i metodi di machine learning assume sempre maggior rilevanza il cosiddetto *deep learning*, che consiste in un'imitazione del comportamento del cervello umano, in quanto dotato di moduli software che simulano i neuroni e le connessioni fra di essi.

58

## Data engineering?

- Data Engineers sono coloro che preparano l'infrastruttura per analizzare i dati. Sono SE che integrano dati da varie risorse e le gestiscono; poi si assicurano siano facilmente accessibili e ottimizzano la gestione dell'intero sistema.
- Data Engineers sono concentrati sul progetto e l'architettura dei dati da analizzare, più che su le tecniche di analisi.

59

## Tecnologie per i Big data

- Alcune tecnologie associate ai big data sono state introdotte ad hoc, ad esempio i sistemi di *cloud computing* sono pensati soprattutto per l'elaborazione batch di enormi moli di dati in parallelo, e sono scarsamente utilizzabili per rispondere ad interrogazioni online.

60

## Tecnologie per i Big data (2)

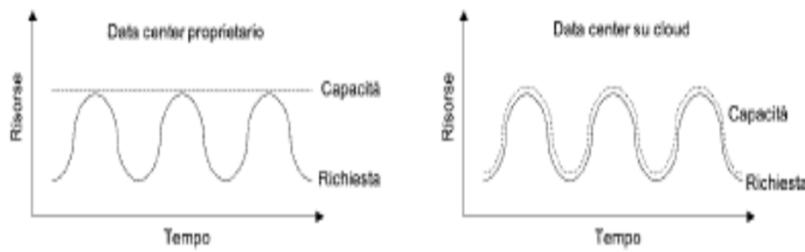
- I cosiddetti sistemi *NoSQL* rinunciano a gestire l'intera complessità del linguaggio SQL o le caratteristiche di piena transazionalità pur di gestire efficientemente alcune tipologie di dati con una semplice struttura.

61

## Le nuove applicazioni

- La maggior parte delle nuove applicazioni usa l'architettura three-tier di Internet, e le applicazioni sono eseguite sfruttando un cloud (privato o pubblico) e servono un grande numero di utenti.
- Infatti c'è grande necessità di spazio e potenza di calcolo per l'analisi dei dati ma non continuativa

62



63

## E i database?

- Poichè aumentano sia i Big Users che i Big Data, le applicazioni, i sottostanti database devono scalare; le scelte possibili sono due: scalare verticalmente (up) o orizzontalmente (out).
  - Scalare verticalmente implica un approccio centralizzato che richiede server sempre più potenti.
  - Scalare orizzontalmente implica invece un approccio distribuito che sfrutta standard server.

64

## Scalare i Database

- I database relazionali hanno un approccio centralizzato (dicevamo che il catalogo è unico) e quindi possono scalare verticalmente ma non orizzontalmente.
  - Significa che, se aumentano utenti o dati, necessitano server più potenti: più CPU, più memoria, più memoria su disco. I server molto potenti sono molto complessi e costano molto, inoltre vanno comprati così.

65

## Google, Facebook ....

- I giganti del web, che si trovano a dover gestire database di dimensioni veramente imponenti, hanno sviluppato (o contribuito allo sviluppo di) vari **NRDBMS** (database non relazionali) con approcci leggermente diversi, ma tutti con gli stessi principi di base, si raggruppano infatti sotto la dicitura di **movimento NoSQL (Not Only SQL)**.

66

## L' Approccio

- I database NoSQL sono stati sviluppati per essere distribuiti e per scalare orizzontalmente. Usano insiemi di server standard sia per i dati che le applicazioni.
- Il sistema scala facilmente aggiungendo nuovi server all'insieme precedente, i dati e le operazioni sono distribuiti sul nuovo cluster, infatti le applicazioni vedono un solo database distribuito.
- Naturalmente ci si aspetta che i server vadano in crash di tanto in tanto, quindi il database è costruito per tollerare i fallimenti ed essere in grado di recuperare i dati.

67

## Caratteristiche

- I database NoSQL presentano alcune caratteristiche comuni per quanto riguarda la scalabilità e la performance.

68

- Auto-sharding - Il database NoSQL distribuisce automaticamente i dati tra i server, senza richiedere la partecipazione delle applicazioni. Infatti i database server possono essere aggiunti o rimossi senza che le applicazioni vengano interrotte.
- Replicazione - La maggior parte dei database NoSQL effettua replicazione dei dati, memorizzandone molte copie sui vari server dello stesso data center o su diversi data center. In questo modo viene garantita un'alta affidabilità e la possibilità di sopravvivere ai "disastri". Un database NoSQL opportunamente gestito non dovrebbe mai dovere essere messo fuori linea per recuperare da fallimenti.

69

- Query distribuite efficienti - Anche nei database relazionali è possibile effettuare sharding, ma in questo caso la possibilità di effettuare query complesse potrebbe essere ridotta. Nel caso di database NoSQL invece si mantiene la completa potenza espressiva delle query anche se il database è distribuito tra centinaia di server
- Caching trasparente - I sistemi NoSQL avanzati effettuano la memorizzazione dei dati necessari per le operazioni dalla memoria su disco alla memoria centrale in modo trasparente alle applicazioni e alla realizzazione delle operazioni.

70

## Vantaggi dei sistemi NoSQL

- La **semplicità** di questi database è uno degli elementi fondamentali, è proprio questo che permette di scalare in orizzontale in maniera così efficiente
- E' possibile scegliere un database adatto alla mappatura più diretta alle object classes del proprio applicativo (Database ad oggetti). In questo modo si possono ridurre di molto i tempi dedicati allo sviluppo del metodo di scambio dati tra il database e l' applicativo stesso

71

## Svantaggi dei sistemi NoSQL

- La semplicità di questi database, però, è dovuta anche alla **mancanza dei controlli fondamentali sull'integrità dei dati**: il compito ricade totalmente sull'applicativo che dialoga col database e che, ovviamente, dovrebbe essere testato in modo molto approfondito prima di essere messo in produzione.

72

## Cont.

- La mancanza di uno standard universale (come può essere l' SQL) è un'altra delle pecche di questi database non relazionali: ogni database ha infatti le proprie API (Application programming Interface) e il suo metodo di storing e di accesso ai dati. Se lo sviluppo del database sul quale abbiamo basato il nostro applicativo venisse interrotto, il passaggio ad un altro database non sarebbe una cosa immediata, ma richiederebbe cambi più o meno profondi da apportare all'applicativo

73

## Esempi

- **MongoDB** è un sistema orientato ai documenti ed è il più diffuso tra i DBMS non relazionali; è usato ad esempio da eBay
- **Cassandra**: è forse il più famoso tra i database non relazionali di ultima generazione, è adottato per gestire gigantesche quantità di dati da compagnie come Facebook, Digg, Twitter, Rackspace e molti altri.
  - Tra le sue caratteristiche principali ci sono lo **scaling orizzontale (out)** e **un'altissima ridondanza** (potrebbe anche cadere un intero data center, con altri nodi sparsi per il globo continuerebbe a funzionare tutto).

74

## Conclusioni

- I database NoSQL non saranno i nuovi dominatori del mercato. I database relazionali saranno ancora usati in una grande varietà di applicazioni.  
Semplicemente non saranno più la scelta automatica.

75

Corso di Laurea in Ingegneria  
Informatica  
*Fondamenti di informatica II*  
Modulo “*Basi di dati*”  
a.a. 2018-2019

Docente: Gigliola Vaglini  
Docente laboratorio: Francesco  
Pistolesi

Lezione 2

Il modello relazionale

## Il modello relazionale

- Proposto da E. F. Codd nel 1970, ma usato in DBMS reali solo dal 1981
- Si basa sul concetto matematico di relazione (con una variante)

3

## Relazione matematica: struttura posizionale

$\text{Partite} \subseteq \text{string} \times \text{string} \times \text{int} \times \text{int}$

Juve	Lazio	3	1
Lazio	Milan	2	0
Juve	Roma	0	2
Roma	Milan	0	1

- Ciascuno dei domini uguali ha ruolo diverso, distinguibile attraverso la posizione
  - la struttura è posizionale
  - non c'è ordinamento fra le n-uple
  -

4

## Relazione matematica

- $D_1, \dots, D_n$  ( $n$  insiemi anche non distinti) detti domini
- prodotto cartesiano  $D_1 \times \dots \times D_n$ :
  - l'insieme di tutte le  $n$ -uple  $(d_1, \dots, d_n)$  tali che  $d_1 \in D_1, \dots, d_n \in D_n$
- **relazione** su  $D_1, \dots, D_n$ :
  - un sottoinsieme di  $D_1 \times \dots \times D_n$ .

5

## Relazione matematica: proprietà

- una relazione è un insieme di  $n$ -uple ordinate  $(d_1, \dots, d_n)$  tali che  $d_1 \in D_1, \dots, d_n \in D_n$  (cioè l'  $i$ -esimo valore proviene dall'  $i$ -esimo dominio)
- non c'è ordinamento fra le  $n$ -uple

6

## Relazione matematica: esempio

- $D_1 = \{a, b\}$
- $D_2 = \{x, y, z\}$
- prodotto cartesiano
  - $D_1 \times D_2 = \{(a, x), (a, y), (a, z), (b, x), (b, y), (b, z)\}$
- relazione
  - $r \subseteq D_1 \times D_2 = \{(a, x), (a, z), (b, y)\}$

7

## Struttura non posizionale

- A ciascun dominio si associa un nome (attributo) che ne descrive il "ruolo,"
  - L'ordinamento tra righe (o tuple) è irrilevante, e
  - anche l'ordinamento tra le colonne è irrilevante

Casa	Fuori	RetiCasa	RetiFuori
Juve	Lazio	3	1
Lazio	Milan	2	0
Juve	Roma	0	2
Roma	Milan	0	1

8

## Tabelle e relazioni

- Le relazioni hanno naturale rappresentazione per mezzo di tabelle
- Una tabella rappresenta una relazione se
  - i valori di ogni colonna sono fra loro omogenei
  - le righe sono diverse fra loro
  - le intestazioni delle colonne sono diverse tra loro

9

## Il modello relazionale è basato su valori

- i riferimenti fra dati di relazioni diverse sono ottenuti tramite valori uguali in tuple diverse

10

studenti	Matricola	Cognome	Nome	Data di nascita
	6554	Rossi	Mario	05/12/1978
	8765	Neri	Paolo	03/11/1976
	9283	Verdi	Luisa	12/11/1979
	3456	Rossi	Maria	01/02/1978
esami	Studente	Voto	Corso	
	3456	30	04	
	3456	24	02	
	9283	28	01	
	6554	26	01	
corsi	Codice	Titolo	Docente	
	01	Analisi	Mario	
	02	Chimica	Bruni	
	04	Chimica	Verdi	

11

## Alternativa

- Altri modelli (sia quelli "storici", reticolare e gerarchico, sia quello a oggetti) prevedono riferimenti esplicativi, gestiti dal sistema
- Non un valore uguale, ma un puntatore al valore

12

The diagram illustrates three tables and their relationships:

- studenti** table:

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

- esami** table:

	Studente	Voto	Corso
		30	
		24	
		28	
		26	

- corsi** table:

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

Relationships are indicated by arrows:

  - An arrow points from the "studenti" table to the "esami" table.
  - Two arrows point from the "studenti" table to the "corsi" table.
  - One arrow points from the "esami" table to the "corsi" table.
  - Three arrows point from the "corsi" table back to the "esami" table.

13

## Vantaggi

- indipendenza dalle strutture fisiche (in realtà i dati non hanno alcuna struttura)
- i dati sono portabili più facilmente da un sistema ad un altro
- l'utente finale vede gli stessi dati dei programmatore
- puntatori eventuali a livello fisico non sono visibili a livello logico

14

## Definizioni (1)

- Schema di relazione:  
un nome  $R$  con un insieme di attributi  
 $A_1, \dots, A_n$   
 $R(A_1, \dots, A_n)$
- Schema di base di dati:  
insieme di schemi di relazione  
 $R = \{R_1(X_1), \dots, R_k(X_k)\}$

15

## Definizioni (2)

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

16

## Definizioni (3)

- (Istanza di) relazione su uno schema  $R(X)$ :
  - insieme  $r$  di tuple su  $X$
- (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  $R_i$ )

17

## Relazioni su singoli attributi

studenti

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

studenti lavoratori

Matricola
6554
3456

18

## Strutture complesse

- I valori devono essere semplici (non relazioni)
- Insieme di relazioni per rappresentare strutture complesse

19

## Informazione incompleta (1)

- Il modello relazionale impone ai dati una struttura rigida
  - le informazioni sono rappresentate da tuple il cui formato deve corrispondere agli schemi di relazione
- Ma i dati disponibili possono non corrispondere al formato previsto

20

## Informazione incompleta (2)

Nome	SecondoNome	Cognome
Franklin	Delano	Roosevelt
Winston		Churchill
Charles		De Gaulle
Josip		Stalin

21

## Rappresentazione dell'informazione incompleta

- Si usa un valore distinto aggiunto a tutti i domini
  - valore nullo: denota l'assenza di un valore del dominio
  - se i valori dell'attributo  $A$  appartengono al dominio  $\text{dom}(A)$ ,  $t[A]$  è un valore del dominio oppure il valore nullo  $\text{NULL}$
- Si possono (e debbono) imporre restrizioni sulla presenza di valori nulli in una relazione

22

Troppi valori nulli rendono prive di significato le tuple relative

studenti	Matricola	Cognome	Nome	Data di nascita
	6554	Rossi	Mario	05/12/1978
	9283	Verdi	Luisa	12/11/1979
	NULL	Rossi	Maria	01/02/1978
esami	Studente	Voto	Corso	
	NULL	30	NULL	
	NULL	24	02	
	9283	28	01	
corsi	Codice	Titolo	Docente	
	01	Analisi	Mario	
	02	NULL	NULL	
	04	Chimica	Verdi	

23

Attenzione: tre tipi differenti di significato del valore nullo

- valore sconosciuto
- valore inesistente
- valore non interessante
  
- I DBMS non distinguono tra tipi diversi di valore nullo

24

## Basi di dati „scorrette“

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

25

## Una base di dati "scorretta"

Esami	Studente	Voto	Lode	Corso
276545	32			01
276545	30	e lode		02
787643	27	e lode		03
739430	24			04

Studenti	Matricola	Cognome	Nome
276545	Rossi	Mario	
787643	Neri	Piero	
787643	Bianchi	Luca	

26

## Vincoli di integrità

- Si devono associare alla base di dati delle proprietà che, se soddisfatte, esprimono la sua “correttezza” rispetto all’applicazione
- I cosiddetti “vincoli di integrità”
  - permettono una descrizione più accurata della realtà
  - danno un contributo alla “qualità dei dati”
  - sono utili nella progettazione
  - sono usati dai DBMS nella esecuzione delle interrogazioni

27

## Vincoli, schemi e istanze

- i vincoli corrispondono a proprietà del mondo reale modellato dalla base di dati e interessano tutte le istanze
- i vincoli sono associati allo schema e si considerano corrette le sue istanze che soddisfano tutti i vincoli

28

## Vincoli di integrità

- Un vincolo è un predicato che associa ad ogni istanza della base di dati il valore vero o falso
- Se il predicato vale vero la proprietà è soddisfatta
- Due tipi di vincoli
  - intrarelazionali
  - interrelazionali

29

## Vincoli intrarelazionali

- Vincoli di tupla: esprimono condizioni sui valori di ciascuna tupla, indipendentemente dalle altre
- Caso particolare
  - Vincoli di dominio: coinvolgono un solo attributo

30

Esami	Studente	Voto	Lode	Corso
	276545	32		01
	276545	30	e lode	02
	787643	27	e lode	03
	739430	24		04

Studenti	Matricola	Cognome	Nome
	276545	Rossi	Mario
	787643	Neri	Piero
	787643	Bianchi	Luca

31

## Sintassi del vincolo di dominio

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

$(Voto \geq 18) \text{ AND } (Voto \leq 30)$

32

## Vincoli su più domini

(Voto = 30) OR NOT (Lode = "e lode")

Stipendi	Impiegato	Lordo	Ritenute	Netto
	Rossi	55.000	12.500	42.500
	Neri	45.000	10.000	35.000
	Bruni	47.000	11.000	36.000

**Lordo = (Ritenute + Netto)**

33

## Identificatore di tupla

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

- non ci sono due tuple con lo stesso valore dell'attributo Matricola
- non ci sono due tuple con lo stesso valore di tutti e tre gli attributi Cognome, Nome e Nascita

34

## Chiave

- insieme di attributi che identificano univocamente le tuple di una relazione  
Formalmente:
  - un insieme  $K$  di attributi è superchiave per  $r$  se  $r$  non contiene due tuple distinte  $t_1$  e  $t_2$  con  $t_1[K] = t_2[K]$
  - $K$  è chiave per  $r$  se è una superchiave minimale per  $r$   
(cioè non contiene un'altra superchiave)

35

## Esempio (1)

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

- Matricola è una chiave:
  - è superchiave
  - contiene un solo attributo e quindi è minimale

36

## Esempio (cont.)

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

- Cognome, Nome, Nascita formano una superchiave, ma non minimale
- Cognome, Nascita non formano una chiave perche' non distinguono la tupla di riga 5 da quella di riga 1
- Nome, Nascita formano una chiave

37

## Esistenza delle chiavi

- Una relazione contiene tuple tutte distinte tra loro (è un insieme)
- Ogni relazione ha come superchiave l'insieme degli attributi su cui è definita
  - quindi ha (almeno) una chiave

38

## 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

39

## Chiavi e valori nulli

- In presenza di valori nulli delle chiavi non è possibile
  - identificare le tuple
  - realizzare i riferimenti con altre relazioni
- La presenza di valori nulli nelle chiavi deve essere limitata

40

## Chiave primaria

- Chiave su cui non sono ammessi valori nulli
- Notazione: sottolineatura

Matricola	Cognome	Nome	Corso	Nascita
86765	NULL	Mario	Ing Inf	5/12/78
78763	Rossi	Mario	Ing Civile	3/11/76
65432	Neri	Piero	Ing Mecc	10/7/79
87654	Neri	Mario	Ing Inf	NULL
43289	Neri	Mario	NULL	5/12/78

41

## Le dipendenze funzionali

- I vincoli di chiave sono particolari tipi di vincoli, parte di una categoria più vasta: le dipendenze funzionali
  - Dati due insiemi di attributi X e Y
  - Si dice che X determina Y, o che Y dipende funzionalmente da X, e si scrive  $X \rightarrow Y$  se:
    - date due tuple distinte  $t_1$  e  $t_2$ , se  $t_1[X] = t_2[X]$  allora  $t_1[Y] = t_2[Y]$
- Le dipendenze funzionali possono essere usate per garantire opportune proprietà della BD

42

## Vincoli interrelazionali

- Informazioni in relazioni diverse possono essere correlate attraverso valori comuni, in particolare, valori delle chiavi (primarie)
- Le correlazioni debbono essere "coerenti"

43

### Infrazioni

<u>Codice</u>	<u>Data</u>	<u>Vigile</u>	<u>Prov</u>	<u>Numer</u>
34321	1/2/95	3987	MI	39548K
53524	4/3/95	3295	TO	E39548
64521	5/4/96	3295	PR	839548
73321	5/2/98	9345	PR	839548

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

44

- Esiste un vincolo interrelazionale tra
  - l'attributo Vigile della relazione Infrazioni e la chiave primaria della relazione Vigili (Matricola)

45

## Vincoli di integrità referenziale

- Un vincolo di integrità referenziale fra gli attributi  $X$  (anche più di uno) di una relazione  $R_1$  e un'altra relazione  $R_2$  impone ai valori su  $X$  in  $R_1$  di comparire come valori della chiave primaria di  $R_2$

46

## Vincoli su più attributi

N.B. in questo caso

- L'ordine degli attributi tra cui è stabilito il vincolo è significativo

47

## Integrità referenziale e valori nulli

- In presenza di valori nulli i vincoli possono essere resi meno restrittivi
- Il vincolo non è fra ogni valore degli attributi  $X$  di una relazione  $R_1$  e la chiave primaria della relazione  $R_2$ , ma tra i valori di  $X$  diversi da NULL e la chiave primaria di  $R_2$

48

## Integrità referenziale e valori nulli

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

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

49

## Reazione alle violazioni dei vincoli

- Cosa succede quando si tenta di compiere un'operazione che viola un vincolo, ad esempio si cerca di inserire nella base di dati un valore non consentito per quell'attributo
- Sono possibili meccanismi per il supporto alla gestione delle violazioni ("azioni compensative")

50

## Gestione della violazione di un vincolo intrarelazionale

- Comportamento "standard":
- Rifiuto dell'operazione

51

## Operazioni interferenti con un vincolo interrelazionale

- Si producono violazioni sulla tabella interna a seguito di modifiche
  - Azione compensativa: nessuna: l'operazione viene impedita
- Azioni sulla tabella esterna ("master")
  - Si vuole eliminare una ennupla
  - Si vuol modificare l'attributo riferito

52

## Azioni compensative per violazioni derivanti da modifiche sulla tabella esterna (1)

- Cancellazione di una riga della tabella esterna o modifica dell'attributo riferito
  - Eliminazione (modifica) in cascata sulla tabella interna (cascade) delle righe corrispondenti
  - Viene posto a null il valore dell'attributo referente (set null)
  - Viene assegnato un valore di default all'attributo referente (set default)
  - La cancellazione non viene consentita (no action)

53

## Eliminazione in cascata

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

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

54

## Introduzione di valori nulli

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

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

55

## Esercizio 1a

Date le seguenti tabelle

AUTORE (Nome, Cognome, DataNascita, Nazionalità)

LIBRO (TitoloLibro, NomeAutore, CognomeAutore, Lingua)

Esiste un vincolo di integrità referenziale tra di esse?

## Esercizio 1b

- Supponendo di avere specificato una politica di cascade sulle modifiche e sulle cancellazioni, spiegare quale è l'effetto dell'esecuzione dei seguenti aggiornamenti:
  - cancella da AUTORE tutte le righe dove Cognome = 'Rossi';
  - modifica LIBRO fissando NomeAutore= 'Umberto' in tutte le righe dove CognomeAutore = 'Eco';

## Soluzione

- Il primo comando cancella dalla tabella AUTORE tutte le tuple con Cognome = 'Rossi'. A causa della politica cascade, anche tutte le tuple di LIBRO con CognomeAutore = 'Rossi' vengono eliminate.
- Il comando causa una violazione a meno che la tabella AUTORE contenga già la tupla "Umberto Eco".

- Se in Libro non c'è la tupla "Umberto Eco" ma solo "Giovanni Eco" cosa si deve fare?

## NoSQL database

- I principali metodi d' implementazione dei database NoSQL sono i seguenti:
- **Columnfamily**: i dati sono organizzati in righe e colonne, ma le righe possono avere quante colonne si vogliono e non c' è bisogno di definire le colonne come prima cosa.
- **Document store**: è l' evoluzione del metodo key/value: rispetto ai normali database relazionali, invece che immagazzinare i dati in tabella con dei campi fissi, questi vengono messi in un documento che può contenere illimitati campi di illimitata lunghezza, così se, ad esempio, di una persona conosciamo solo nome e cognome, ma magari di un' altra persona anche indirizzo, data di nascita e codice fiscale, si evita che per il primo nominativo ci siano campi inutilizzati.

## Vantaggio importante

- Sfruttando la dinamicità della dimensione delle righe, ogni elemento può contenere tutte le informazioni collegate alla chiave. Di conseguenza **non serve usare dispendiosi** (in termini di performance) **operatori di collegamento tra tabelle**, come invece avviene per i database relazionali.

61

## Svantaggio

- Controlli di consistenza:
  - Per esempio, se avessimo un database dei clienti coi relativi ordini, in un database non relazionale alla cancellazione di un cliente tutti gli ordini resterebbero comunque nel database, è quindi l'applicativo che una volta impartito il comando di cancellazione dell'utente X deve anche andare a cancellare tutti i relativi ordini, cosa che invece in un database relazionale è gestita direttamente dal DBMS.

62

**Corso di Laurea in Ingegneria  
Informatica**  
**Fondamenti di Informatica I**  
**Modulo "Basi di dati"**  
**a.a. 2018-2019**

Docente: Gigliola Vaglini  
Docente laboratorio: Francesco  
Pistolesi

1

**Lezione 3**

Significato e  
implementazione di una  
interrogazione

2

- Cosa si intende per interrogazione?
  - Operazione di lettura sul DB che può richiedere l'accesso a più di una tabella
- Cosa è necessario fare per realizzare una interrogazione?

3

## Semantica di un linguaggio di programmazione

- Operazionale
  - Si specificano le modalità di generazione del risultato
  - Nel caso di SQL si definisce questa semantica tramite la cosiddetta Algebra relazionale
- Dichiarativa
  - Si specificano le proprietà del risultato,
  - Nel caso di SQL si usa il Calcolo relazionale

4

## Due semantiche?

- Il metodo dichiarativo è l'effettiva semantica del linguaggio, infatti le interrogazioni sono espresse ad alto livello (ricordare il concetto di **indipendenza dei dati**)
  - Nessun concetto di costo (ma in realtà si può vedere il costo perché ↓)
- Il metodo operazionale è il modo che usa il DBMS per eseguire un'istruzione SQL

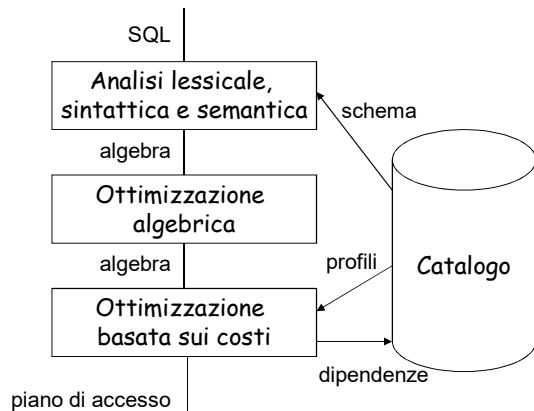
5

## Il DBMS

- Il DBMS contiene un modulo specifico detto **Query processor**, all'interno del quale è definito il processo di esecuzione delle interrogazioni
- Una parte del QP si occupa di **ottimizzare** (il QP è detto anche Ottimizzatore) la query prima dell'esecuzione (la query è scritta indipendentemente dal suo costo, importa solo il risultato). L'ottimizzatore sceglie la strategia (di solito fra diverse alternative) di esecuzione per ogni istruzione SQL.

6

# Il Query Processor



7

## "Profili" delle relazioni

- **Informazioni quantitative:**
  - cardinalità di ciascuna relazione
  - dimensioni delle tuple
  - dimensioni dei valori
  - numero di valori distinti degli attributi
  - valore minimo e massimo di ciascun attributo
- Sono memorizzate nel "catalogo" e possono essere aggiornate con comandi del tipo `update statistics`
- Utilizzate nella fase finale dell'ottimizzazione, per stimare le dimensioni dei risultati intermedi

8

## Ottimizzazione algebrica

- Il termine ottimizzazione è improprio perché il processo utilizza euristiche
- Si basa sulla nozione di equivalenza:
  - Due espressioni sono equivalenti se producono lo stesso risultato qualunque sia l'istanza attuale della base di dati

9

## Algebra relazionale

- Algebra = dati + operatori
- Algebra relazionale:
  - Dati: relazioni
  - Operatori:
    - su relazioni
    - che producono relazioni
    - e possono essere composti

10

## Operatori dell'algebra relazionale

- Operatori su insiemi
  - unione, intersezione, differenza
- Operatori su relazioni
  - ridenominazione
  - selezione
  - proiezione
  - join (join naturale, prodotto cartesiano, theta-join)

11

### Operatori su insiemi

- Le relazioni sono insiemi, quindi si possono applicare gli operatori su insiemi, cioè unione, intersezione, differenza
- I risultati debbono essere ancora relazioni
  - quindi è possibile applicare unione, intersezione, differenza solo a relazioni definite sugli stessi attributi, in modo che il risultato sia una relazione sugli stessi attributi

12

## Unione

- L'unione di due relazioni sullo stesso insieme di attributi X è una relazione su X che contiene le tuple sia dell'una che dell'altra relazione originaria

13

## Unione

Laureati triennali

Matricola	Nome	Età
7274	Rossi	32
7432	Neri	24
9824	Verdi	25

Laureati magistrali

Matricola	Nome	Età
9297	Neri	33
7432	Neri	24
9824	Verdi	25

Laureati triennali  $\cup$  Laureati magistrali

Matricola	Nome	Età
7274	Rossi	32
7432	Neri	24
9824	Verdi	25
9297	Neri	33

14

## Intersezione

- L'intersezione di due relazioni sullo stesso insieme di attributi X è una relazione su X che contiene le tuple appartenenti ad entrambe le relazioni

15

## Intersezione

Laureati triennali

Matricola	Nome	Età
7274	Rossi	32
7432	Neri	24
9824	Verdi	25

Laureati magistrali

Matricola	Nome	Età
9297	Neri	33
7432	Neri	24
9824	Verdi	25

Laureati triennali  $\cap$  Laureati magistrali

Matricola	Nome	Età
7432	Neri	24
9824	Verdi	25

16

## Differenza

- La differenza di due relazioni sullo stesso insieme di attributi X,  
–  $r_1(X) - r_2(X)$
- è una relazione su X che contiene le tuple di  $r_1$  che non appartengono anche ad  $r_2$ .

17

## Differenza

Laureati triennali

Matricola	Nome	Età
7274	Rossi	32
7432	Neri	24
9824	Verdi	25

Laureati magistrali

Matricola	Nome	Età
9297	Neri	33
7432	Neri	24
9824	Verdi	25

Laureati triennali – Laureati magistrali

Matricola	Nome	Età
7274	Rossi	32

18

È possibile l'unione delle due relazioni seguenti?

Paternità

Padre	Figlio
Adamò	Abele
Adamò	Caino
Abramo	Isacco

Maternità

Madre	Figlio
Eva	Abele
Eva	Set
Sara	Isacco

Paternità  $\cup$  Maternità

??

19

## Ridenominazione

- operatore monadico (con un argomento)
- "modifica lo schema" dell'argomento lasciando inalterata l'istanza

$\rho_{B_1..B_n \leftarrow A_1..A_n}(r)$

20

### Paternità

Padre	Figlio
Adamò	Abele
Adamò	Caino
Abramo	Isacco

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

Genitore	Figlio
Adamò	Abele
Adamò	Caino
Abramo	Isacco

21

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

Genitore	Figlio
Adamò	Abele
Adamò	Caino
Abramo	Isacco

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

### $\rho_{\text{Genitore} \leftarrow \text{Madre}}$ (Maternità)

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

### $\rho_{\text{Genitore} \leftarrow \text{Madre}}$ (Maternità)

Genitore	Figlio
Eva	Abele
Eva	Set
Sara	Isacco

22

## Selezione

- operatore monadico
- produce un risultato che
  - ha lo stesso schema dell'argomento e
  - contiene il sottoinsieme delle sue tuple che soddisfano una condizione fissata

23

## Sintassi e semantica

- data una relazione  $r(X)$ 
$$\sigma_F(r) = r'$$
  - $F$ : espressione booleana ottenuta componendo con and, or e not condizioni atomiche del tipo  $A \# B$  oppure  $A \# c$ , con  $A$  e  $B$  attributi in  $X$  con domini compatibili,  $\#$  operatore di confronto ( $<$ ,  $>$ ,  $=$ , ..) e  $c$  costante compatibile con il dominio di  $A$ .
  - $r'$  contiene il sottoinsieme delle tuple di  $r$  per cui  $F$  è' vera

24

– impiegati che guadagnano più di 50000 euro

### Impiegati

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

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

25

– impiegati che guadagnano più di 50000  
e lavorano a Milano

### Impiegati

Matricola	Cognome	Filiale	Stipendio
5998	Neri	Milano	64000

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

26

## Selezione con valori nulli

### Impiegati

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

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

- la condizione è vera solo per valori non nulli

27

per riferirsi ai valori nulli esistono forme apposite di condizioni:

IS NULL

IS NOT NULL

### Impiegati

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

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

28

## Proiezione

- operatore monadico
- produce un risultato che
  - ha parte degli attributi dell'argomento e su tali attributi contiene tutte le possibili tuple di valori esistenti nella relazione argomento

29

## Sintassi e semantica

- Sintassi,  $Y \subseteq X$   
 $\pi_Y(r(X)) = r'$
- Semantica
  - $r'$  è una relazione su  $Y$  e contiene l'insieme delle tuple di  $r$  ristrette agli attributi in  $Y$

30

– matricola e cognome di tutti gli impiegati

Matricola	Cognome
7309	Neri
5998	Neri
9553	Rossi
5698	Rossi

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

31

– cognome e filiale di tutti gli impiegati

	Cognome	Filiale
	Neri	Napoli
	Neri	Milano
	Rossi	Roma

$\pi_{\text{cognome, Filiale}} (\text{Impiegati})$

Attenzione: perché c'è differenza nella dimensione dei due risultati?

32

## Cardinalità delle proiezioni

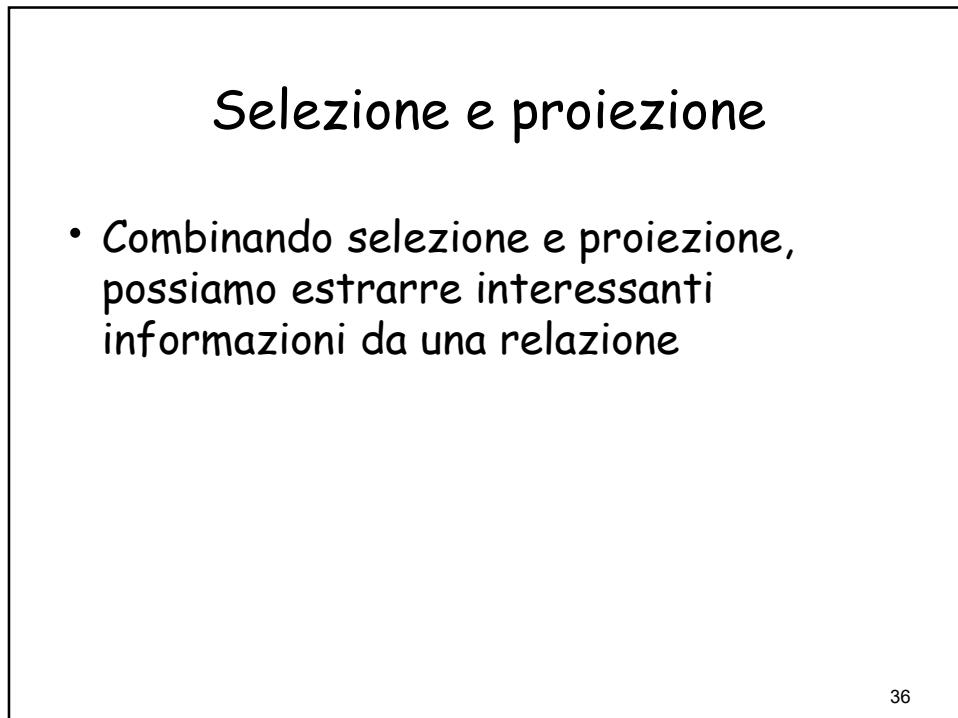
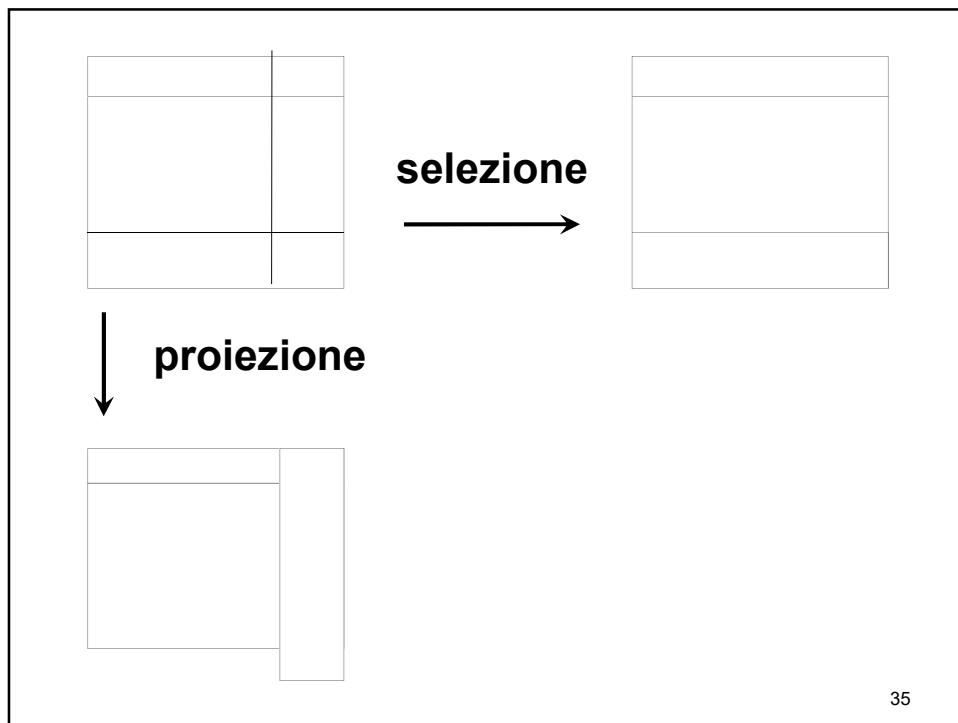
- una proiezione di  $r$ 
  - contiene al più tante tuple quante ne ha  $r$
  - può contenerne di meno
- se  $X$  è una superchiave di  $r$ , allora  $\pi_X(r)$  contiene esattamente tante tuple quante ne ha  $r$

33

## Proiezione e selezione

- Selezione  $\sigma$ 
  - decomposizione orizzontale
- Proiezione  $\pi$ 
  - decomposizione verticale

34



- matricola e cognome degli impiegati che guadagnano più di 50

Matricola	Cognome		
7309	Rossi		
5998	Neri		
5698	Neri		

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

Attenzione: ordine degli operatori

37

- Combinando selezione e proiezione, non possiamo però correlare informazioni presenti in relazioni diverse

38

## Una prima combinazione

- Il prodotto cartesiano di due relazioni,  
R e Q: operatore X

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
B	Mori
C	Bruni

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

39

## Join

- il join è l'operatore più interessante dell'algebra relazionale
- permette appunto di correlare dati in relazioni diverse

40

## Join naturale

- operatore binario (generalizzabile)
- produce come risultato una relazione tale che
  - Il suo schema ha l'unione degli attributi degli argomenti
  - L'insieme delle tuple è ottenuto componendo una tupla di ognuno degli operandi per valori uguali degli attributi comuni

41

## Sintassi e semantica

- $R_1(X_1), R_2(X_2)$
- $R_1 \bowtie R_2$  è una relazione su  $X_1 \cup X_2$  definita come
$$\{ t \mid \text{esistono } t_1 \in R_1 \text{ e } t_2 \in R_2 \text{ con } t[X_1] = t_1 \text{ e } t[X_2] = t_2 \}$$

42

## Join e proiezioni

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

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

Impiegato	Reparto	Reparto	Capo
Neri	B	B	Mori
Bianchi	B		

43

## Proiezioni e join

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

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

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

44

## In generale

- $R_1(X_1), R_2(X_2)$

$$\pi_{X_1}(R_1 \triangleright\triangleleft R_2) \subseteq R_1$$

- $R(X), X = X_1 \cup X_2$

$$(\pi_{X_1}(R)) \triangleright\triangleleft (\pi_{X_2}(R)) \supseteq R$$

45

## Relazioni senza attributi comuni

- La definizione di join funziona ugualmente
  - $R_1(X_1), R_2(X_2)$
  - $R_1 \triangleright\triangleleft R_2$  è una relazione su  $X_1 \cup X_2$  definita come
$$\{ t \mid \text{esistono } t_1 \in R_1 \text{ e } t_2 \in R_2 \text{ con } t[X_1] = t_1 \text{ e } t[X_2] = t_2 \}$$

46

## Risultato

- La relazione risultato contiene sempre un numero di tuple pari al prodotto delle cardinalità degli operandi (le tuple sono tutte combinabili )
- Equivale al prodotto cartesiano su tuple

47

Impiegati

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparti

Codice	Capo
A	Mori
B	Bruni

Impiegati  $\bowtie$  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

48

- Il **prodotto cartesiano**, può essere ridotto eseguendo una selezione

$$\sigma_F (R_1 \times R_2)$$

- L' operazione complessiva può venire eseguita tramite un operatore derivato chiamato theta-join e indicato con

$$R_1 \triangleright\triangleleft_F R_2$$

49

## Perché "theta-join"

- La condizione  $F$  è spesso una congiunzione (AND) di atomi di confronto  $A_1 \vartheta A_2$  dove  $\vartheta$  è uno degli operatori di confronto ( $=, >, <, \dots$ ) e  $A_1, A_2$  sono attributi di relazioni diverse
- se l'operatore è sempre l'uguaglianza ( $=$ ) allora si parla di equi-join

50

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

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

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

51

## Join naturale ed equi-join

Impiegati		Reparti	
Impiegato	Reparto	Reparto	Capo
<b>Impiegati <math>\bowtie</math> Reparti =</b>			
$\pi_{\text{Impiegato}, \text{Reparto}, \text{Capo}} ($			
$\rho_{I.\text{Reparto} \leftarrow \text{Reparto}} (\text{Impiegati}) \bowtie_{I.\text{Reparto} = \text{Reparto}} \text{Reparti })$			

52

## Prodotto cartesiano e join naturale

- Impiegati  $\bowtie$  Reparti =

$$\begin{array}{l} \pi_{\text{Impiegato}, \text{Reparto}, \text{Capo}} (\sigma_{I.\text{reparto} = \text{reparto}} \\ \rho_{I.\text{Reparto} \leftarrow \text{Reparto}} (\text{Impiegati}) \times \text{Reparti) } \end{array}$$

Il join non è un operatore primitivo

53

## Esempi

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

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

54

- Trovare le matricole dei capi degli impiegati che guadagnano più di 40000 euro

$$\begin{aligned} & \pi_{\text{Capo}}(\text{Supervisione} \\ & \quad \triangleright \triangleleft \text{Impiegato} = \text{Matricola} ( \\ & \quad \sigma_{\text{Stipendio} > 40000}(\text{Impiegati}))) \end{aligned}$$

55

- Trovare le matricole dei capi i cui impiegati guadagnano tutti più di 40000 euro

$$\begin{aligned} & \pi_{\text{Capo}}(\text{Supervisione}) - \\ & \quad \pi_{\text{Capo}}(\text{Supervisione} \\ & \quad \quad \triangleright \triangleleft \text{Impiegato} = \text{Matricola} \\ & \quad \quad (\sigma_{\text{Stipendio} \leq 40000}(\text{Impiegati}))) \end{aligned}$$

56

## Esempi

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

<b>Supervisione</b>	<b>Impiegato</b>	<b>Capo</b>
	<b>7309</b>	<b>5698</b>
	<b>5998</b>	<b>5698</b>
	<b>9553</b>	<b>4076</b>
	<b>5698</b>	<b>4076</b>
	<b>4076</b>	<b>8123</b>

57

- Trovare nome e stipendio dei capi degli impiegati che guadagnano più di 40000 euro

$$\begin{aligned}
 & \pi_{\text{Nome}, \text{Stipendio}} ( \\
 & \text{Impiegati} \triangleright\triangleleft \text{Matricola}=\text{Capo} \\
 & \pi_{\text{Capo}} (\text{Supervisione} \\
 & \triangleright\triangleleft \text{Impiegato}=\text{Matricola} (\sigma_{\text{Stipendio}>40000}(\text{Impiegati})))
 \end{aligned}$$

58

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

$$\begin{aligned}
 & \pi_{\text{Matr, Nome, Stip, MatrC, NomeC, StipC}} \\
 & (\sigma_{\text{Stipendio} > \text{StipC}} \\
 & \rho_{\text{MatrC, NomeC, StipC, EtàC} \leftarrow \text{Matr, Nome, Stip, Età}(\text{Impiegati})} \\
 & \quad \triangleright \triangleleft_{\text{MatrC} = \text{Capo}} \\
 & (\text{Supervisione} \triangleright \triangleleft_{\text{Impiegato} = \text{Matricola}} \text{Impiegati})) \\
 \end{aligned}$$

59

## Equivalenza di espressioni

60

## 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"

61

## Equivalenze importanti (1)

- Pushing selections down (se  $A$  è attributo di  $R_2$ )  
$$\sigma_{A=10}(R_1 \triangleright\triangleleft R_2) = R_1 \triangleright\triangleleft \sigma_{A=10}(R_2)$$
- Riduce in modo significativo la dimensione del risultato intermedio (e quindi il costo dell'operazione)

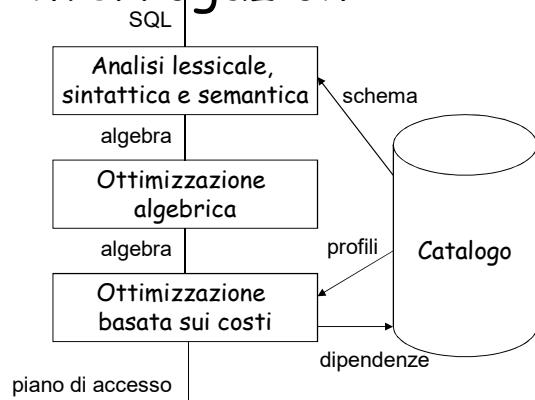
62

## Equivalenze importanti (2)

- Pushing projections down (siano dati  $R_1(X_1)$  e  $R_2(X_2)$  con  $Y_2 \subseteq X_2$ )  
$$\pi_{X_1 Y_2}(R_1 \bowtie R_2) = R_1 \bowtie \pi_{Y_2}(R_2)$$
- Riduce in modo significativo la dimensione del risultato intermedio

63

## Il processo di esecuzione delle interrogazioni



64

## Ottimizzazione algebrica

- Il termine ottimizzazione è improprio perché il processo utilizza euristiche e si basa sulla
  - nozione di equivalenza:
  - Due espressioni sono equivalenti se producono lo stesso risultato qualunque sia l'istanza attuale della base di dati
- Euristiche fondamentali:
  - selezioni e proiezioni il più presto possibile (per ridurre le dimensioni dei risultati intermedi):
    - "push selections down"
    - "push projections down"

65

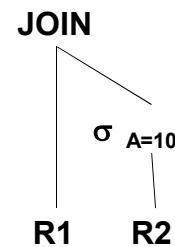
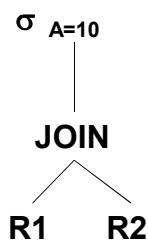
## Rappresentazione interna delle interrogazioni

- Alberi:
  - foglie: dati (relazioni, file)
  - nodi intermedi: operatori (operatori algebrici, poi effettivi operatori di accesso ai dati)

66

## Alberi per la rappresentazione di interrogazioni

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



67

## Procedura euristica dell'ottimizzatore

- Decomporre le selezioni congiuntive in successive selezioni atomiche
- Anticipare il più possibile le selezioni
- In una sequenza di selezioni, anticipare le più selettive
- Combinare prodotti cartesiani e selezioni per formare join
- Anticipare il più possibile le proiezioni (anche introducendone di nuove)

68

## Esempio

R1(ABC), R2(DEF), R3(GHI)

$$\pi_{AE} (\sigma_{B>100 \text{ AND } H=7 \text{ AND } I>2} ((R1 \text{ JOIN}_{C=D} R2) \text{ JOIN}_{F=G} R3))$$

69

## Esempio, continua

$$\pi_{AE} (\sigma_{B>100 \text{ AND } H=7 \text{ AND } I>2} ((R1 \text{ JOIN}_{C=D} R2) \text{ JOIN}_{F=G} R3))$$

- viene trasformata in

$$(\sigma_{B>100} (R1) \text{ JOIN}_{C=D} R2) \text{ JOIN}_{F=G} \sigma_{I>2} (\sigma_{H=7} (R3)))$$

- oppure

$$\begin{aligned} &\pi_{AE} ( \\ &\pi_{AEF} ((\pi_{AC} (\sigma_{B>100} (R1))) \text{ JOIN}_{C=D} R2) \\ &\quad \text{JOIN}_{F=G} \\ &\quad \pi_E (\sigma_{I>2} (\sigma_{H=7} (R3)))) \end{aligned}$$

70

## Semantica di SQL: Calcolo relazionale

71

## Calcolo relazionale

- Una famiglia di linguaggi dichiarativi, basati sul calcolo dei predicati del primo ordine
- Diverse versioni:
  - calcolo relazionale su domini
  - calcolo su ennuple con dichiarazioni di range

72

## Calcolo su domini

$\{ A_1: x_1, \dots, A_n: x_n \mid f \}$

- $A_i$  sono nomi di attributi
- $x_i$  sono nomi di variabili
- La lista di coppie  $A_i : x_i$  viene detta target list (descrive il risultato)
- $f$  è una formula
  - Formule atomiche sono  $R(A_1: x_1, \dots, A_n: x_n)$ , che è vera sui valori di  $x_1 \dots x_n$  che formano una tpla di  $R$ , e  $x_i \not\in x_j$ , che è vera sui valori di  $x_i$  e  $x_j$  che soddisfano  $\not\in$

73

## Calcolo su tuple con dichiarazione di range

$\{ x_1.Z_1, \dots, x_n.Z_n \mid x_i(R_1), \dots, x_j(R_m) \mid f \}$

- $x_1.Z_1, \dots, x_n.Z_n$  è la target list
- $x_i(R_1), \dots, x_j(R_m)$  è la range list (dice il campo di variabilità delle variabili)
- $f$  è una formula, con formule atomiche del tipo  $x_i.Z_i \not\in x_j.Z_j$ , ad esempio

74

## Base di dati per gli esempi

Impiegati(Matricola,Nome, Età,  
Stipendio)

Supervisione(Capo, Impiegato)

75

## Esempio 1a

- Trovare gli impiegati che guadagnano più di 40 milioni

{ Matricola: m, Nome: n, Età: e, Stipendio:  
s |

Impiegati (Matricola: m, Nome: n, Età: e,  
Stipendio: s)  $\wedge$  s > 40 }

76

## Esempio 1b

- Trovare gli impiegati che guadagnano più di 40 milioni

{  $i.* \mid i(\text{Impiegati}) \mid i.\text{Stipendio} > 40$  }

77

## Esempio 2a

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

{ Matricola: m, Nome: n |  
Impiegati (Matricola: m, Nome: n, Età: e,  
Stipendio: s)  $\wedge$  s > 40 }  
oppure

{ Matricola: m, Nome: n |  
 $\exists e, s (\text{Impiegati} (\text{Matricola: } m, \text{Nome: } n, \text{Età: } e,$   
Stipendio: s)  $\wedge$  s > 40) }

78

## Esempio 2b

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

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

79

## Quantificatori esistenziali e universali

- Per interrogazioni più complesse, che in algebra ad esempio richiedevano una differenza, servono altri strumenti
- $\exists, \forall$ 
  - Sono intercambiabili
  - $\exists x(f) = \neg(\forall x(\neg(f)))$
  - $\forall x(f) = \neg(\exists x(\neg(f)))$

80

## Quantificatori esistenziali e universali

- Trovare matricola e nome dei capi i cui impiegati guadagnano tutti più di 40 milioni.
- $$\{ \text{Matricola: } c, \text{Nome: } n \mid \\ \text{Impiegati}(\text{Matricola: } c, \text{Nome: } n, \text{Età: } e, \text{Stipendio: } s) \\ \wedge \\ \text{Supervisione}(\text{Capo: } c, \text{Impiegato: } m) \wedge \\ \forall m' (\forall n' (\forall e' (\forall s' ( \\ \neg (\text{Impiegati}(\text{Matricola: } m', \text{Nome: } n', \text{Età: } e', \text{Stipendio: } s') \\ \wedge \\ \text{Supervisione}(\text{Capo: } c, \text{Impiegato: } m')) \vee s' > 40))))\}$$

81

## Quantificatori esistenziali e universali

- Trovare matricola e nome dei capi i cui impiegati guadagnano tutti più di 40 milioni.
- $$\{ \text{Matricola: } c, \text{Nome: } n \mid \\ \text{Impiegati}(\text{Matricola: } c, \text{Nome: } n, \text{Età: } e, \text{Stipendio: } s) \\ \wedge \\ \text{Supervisione}(\text{Capo: } c, \text{Impiegato: } m) \wedge \\ \neg \exists m' (\exists n' (\exists e' (\exists s' ( \\ \text{Impiegati}(\text{Matricola: } m', \text{Nome: } n', \text{Età: } e', \text{Stipendio: } s') \\ \wedge \\ \text{Supervisione}(\text{Capo: } c, \text{Impiegato: } m') \wedge s' \leq 40 \wedge \\ m \neq m'))))\}$$

82

## Quantificatori esistenziali e universali

{Matricola: c, Nome: n |  
Impiegati(Matricola: c, Nome: n, Età: e, Stipendio: s)

Supervisione(Capo:c, Impiegato:m)  $\wedge$   
 $\neg \exists m' (\exists n' (\exists e' (\exists s' ($

Impiegati(Matr: m', Nome: n', Età: e', Stip: s')  $\wedge$   
Supervisione(Capo:c, Impiegato:m')  $\wedge$  s'  $\leq$  40))))}

{ i.(Matricola, Nome) | s(Supervisione), i(Impiegati) |  
i.Matricola=s.Capo  $\wedge$   $\neg (\exists i' (Impiegati)(\exists s' (Supervisione)$   
(s.Capo=s'.Capo  $\wedge$  s'.Impiegato=i'.Matricola  $\wedge$   
i'.Stipendio  $\leq$  40)))}

83

## Calcolo su domini, discussione

- Pregi:
  - dichiaratività
- Difetti:
  - "verbosità": tante variabili!
    - Le variabili del calcolo dei domini rappresentano singoli valori
    - Nel calcolo su tple rappresentano tple
  - possibilità di scrivere espressioni senza senso (dipendenti dal dominio)
    - {A:x, B:y | R(A:x)  $\wedge$  y=y}
  - nell'algebra tutte le espressioni hanno un senso (indipendenti dal dominio)

84

## Calcolo su tuple, discussione

- Il calcolo su tuple con dichiarazioni di range non permette di esprimere alcune interrogazioni importanti, in particolare le unioni:  
$$R_1(AB) \cup R_2(AB)$$
  - Ogni variabile ha un solo range nel risultato, mentre vorremmo tple sia della prima relazione che della seconda
- Nota: intersezione e differenza sono esprimibili
- Per questa ragione SQL (che è basato su questo calcolo) prevede un operatore esplicito di unione, ma non tutte le versioni prevedono intersezione e differenza

85

## Calcolo e algebra

- Calcolo e algebra sono "equivalenti"
  - per ogni espressione del calcolo relazionale che sia indipendente dal dominio esiste un'espressione dell'algebra relazionale equivalente a essa
  - per ogni espressione dell'algebra relazionale esiste un'espressione del calcolo relazionale equivalente a essa (e di conseguenza indipendente dal dominio)

86

## Calcolo e algebra: limiti

- l'insieme di interrogazioni esprimibili è significativo
- Ci sono però interrogazioni interessanti non esprimibili, ad es.
  - interrogazioni inerentemente ricorsive, come la chiusura transitiva

87

## Chiusura transitiva

Supervisione(Impiegato, Capo)

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

Impiegato	Capo
Rossi	Lupi
Neri	Bruni
Lupi	Falchi

Impiegato	Superiore
Rossi	Lupi
Neri	Bruni
Lupi	Falchi
Rossi	Falchi

88

## Chiusura transitiva

- Nell'esempio, basterebbe il join della relazione con se stessa, previa opportuna ridenominazione
- Ma aggiungiamo una nuova ennupla

Impiegato	Capo
Rossi	Lupi
Neri	Bruni
Lupi	Falchi
Falchi	Leoni

Impiegato	Superiore
Rossi	Lupi
Neri	Bruni
Lupi	Falchi
Rossi	Falchi
Lupi	Leoni
Rossi	Leoni

89

## Chiusura transitiva

- Non esiste la possibilità di esprimere l'interrogazione che calcoli la chiusura transitiva di una relazione qualunque
- In algebra relazionale l'operazione si simulerebbe con un numero di join illimitato

90

## Si consideri il seguente schema di base di dati

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

91

## Formulare in algebra relazionale le seguenti interrogazioni (1)

- nomi e cognomi dei registi che hanno diretto film che hanno incassato il primo anno di uscita meno nelle sale che per il noleggio di DVD

$$\begin{aligned} \pi_{N,C} ( & \pi_{N,C,CF} (\pi_{N,C,CR} (\text{Regista}) \triangleright\triangleleft \pi_{CF,CR} (\text{Film})) \\ & \triangleright\triangleleft \\ & \pi_{CF} (\sigma_{\text{Inc1sala} < \text{Inc1DVD}} (\pi_{\text{Inc1sala}, CF} (\text{Produzione}) \\ & \quad \triangleright\triangleleft \pi_{\text{Inc1DVD}, CF} (\text{Noleggio}) ) ) \\ & ) \end{aligned}$$

92

## calcolo dei domini

- {Nome: n, Cognome: c |  
 Regista(CodiceRegista: cr, Cognome:c,  
 Nome:n,...)  $\wedge$  Film( CodiceFilm: cf,...  
 CodiceRegista: cr,...)  $\wedge$  Produzione(..  
 CodiceFilm:cf,...Incasso1annoSala:is)  $\wedge$   
 Noleggio(CodiceFilm:cf,...  
 IncassolannoDVD:idvd)  $\wedge$  (is < idvd) }

93

## Formulare in algebra relazionale le seguenti interrogazioni (2)

- i titoli dei film i cui attori sono tutti dello stesso sesso

1.  $\pi_{\text{Titolo}}(\text{Film}) - \pi_{\text{Titolo}}(\text{Film} \triangleright \triangleleft \sigma_{\text{Sesso} \leftrightarrow s}(\pi_{CF, \text{Sesso}}(\text{Artista} \triangleright \triangleleft \text{Interpretazione}) \triangleright \triangleleft \rho_{s' \leftarrow \text{Sesso}}(\pi_{CF, \text{Sesso}}(\text{Artista} \triangleright \triangleleft \text{Interpretazione}))))$

2.  $(\pi_{\text{Titolo}}(\text{Film}) - \pi_{\text{Titolo}}(\text{Film} \triangleright \triangleleft (\pi_{CF}(\sigma_{\text{Sesso}=\text{M}^+}(\text{Artista}) \triangleright \triangleleft \text{Interpretazione})))) \cup (\pi_{\text{Titolo}}(\text{Film}) - \pi_{\text{Titolo}}(\text{Film} \triangleright \triangleleft (\pi_{CF}(\sigma_{\text{Sesso}=\text{F}^-}(\text{Artista}) \triangleright \triangleleft \text{Interpretazione}))))$

94

## calcolo dei domini

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

95

## Formulare in algebra relazionale le seguenti interrogazioni (3)

- i titoli di film con solamente attori donna che abbiano incassato in sala più del proprio costo

$$\begin{aligned} & \pi_{\text{Titolo}}(\pi_{CF, \text{Titolo}}(\text{Film}) \triangleright \triangleleft \\ & \pi_{CF}(\sigma_{\text{Inc1S}>\text{Costo}}(\text{Produzione})) \triangleright \triangleleft \\ & (\pi_{CF}(\text{Film}) - \\ & \pi_{CF}(\pi_{CA}(\sigma_{\text{Sesso}=\text{'M'}}(\pi_{CA, \text{Sesso}}(\text{Artista})))) \\ & \triangleright \triangleleft \\ & \pi_{CA, CF}(\text{Interpretazione})) \\ & ) \end{aligned}$$

96

## Estensioni dell'algebra

97

## Algebra relazionale estesa

- Il modello relazionale può essere facilmente esteso a comprendere gli operatori SQL non direttamente riconducibili agli operatori algebrici introdotti
- Questa estensione non modifica il funzionamento del modello

98

## Estensioni (1)

- Join esterno (left, right, full)
- permette di generare valori null per mezzo delle espressioni dell'algebra relazionale per modellare le informazioni mancanti
- Left( $= \triangleright \triangleleft$ ): solo tuple dell'operando sinistro sono riempite con NULL
- Right( $\triangleright \triangleleft =$ ): idem per l'operando destro
- Full( $= \triangleright \triangleleft =$ ): per entrambi gli operandi

99

## Esempi

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

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

100

## Uso del join esterno

$$\begin{aligned} & \pi_{\text{Capo}} (\sigma_{\text{Matricola}=\text{Null}} (\text{Supervisione} \\ & = \triangleright \triangleleft_{\text{Impiegato}=\text{Matricola}} \\ & \sigma_{\text{Stipendio} \leq 40000} (\text{Impiegati}))) \end{aligned}$$

101

## Estensioni (2)

- Proiezione generalizzata
  - $\pi_{F1,F2,F3}(E)$
  - F1, F2, F3 sono espressioni aritmetiche su attributi di E (che è una qualunque espressione dell'algebra) e costanti

102

Conto		
Cliente	Credito	Spese
Andrea	6000	1000
Andrea	4000	500
Maria	10000	2000
Anna	3000	1500
Filippo	3000	1000
Luigi	5000	1800
Franco	5000	2000
Maria	6000	2000
Andrea	10000	5000
Anna	5000	1000

103

## Esempio proiezione

- Si può scrivere ad esempio
  - $\pi_{\text{Cliente}, \text{Credito-Spese}}(\text{Conto})$
- ed ottenere il seguente risultato

104

	<b>Cliente</b>	<b>Credito-Spesa</b>
	<b>Andrea</b>	<b>5000</b>
	<b>Andrea</b>	<b>3500</b>

105

## Estensioni (3)

- Funzioni aggregate
  - Si possono usare nelle espressioni alcuni nomi di funzioni (operatori) che si applicano a (multi)insiemi e producono un valore come risultato

106

## Operatori aggregati

- sum, count, min, max
  - $\text{sum}_{\text{Spese}}(\text{Conto})$
  - $\text{count}_{\text{Cliente}}(\text{Conto})$
  - $\text{max}_{\text{Credito}}(\text{Conto})$
  - $\text{count-distinct}_{\text{Cliente}}(\text{Conto})$

107

## Raggruppamento

- Si possono raggruppare gli elementi di una relazione usando un'operatore apposito  
 $\text{Cliente} G_{\text{sum}(\text{Credito})}(\text{Conto})$
- Cliente è l'attributo su cui si fa il raggruppamento, sum è la funzione aggregata che si applica all'attributo Credito, Conto è la relazione su cui si applica il tutto.
- Si possono avere più attributi a sinistra e più funzioni a destra di G

108

## Un altro operatore derivato: divisione

- In generale, la divisione è utile per interrogazioni di tipo "universale"
- Vogliamo trovare i nomi dei clienti che hanno un conto corrente in tutte le filiali di banca di Pisa.
- Le relazioni sono
  - Branch(bank\_name, branch\_name, branch\_city)
  - Account(branch\_name, bank\_name, account\_number, branch\_city)
  - Depositor(account\_number, customer\_name)

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

109

## Cosa si intende

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

Equivale a

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

110

$\Pi_{CN}(\text{depositor} \triangleright\!\!\!< \text{account}) \triangleright\!\!\!< \Pi_{BN}(\sigma_{BC=Pisa'}(\text{branch}))$

Si combinano tutti i clienti presenti nel data base con le filiali di Pisa;

togliendo da questo insieme le coppie (cliente,filiale) presenti nel data base, cioè

$\Pi_{CN,BN}(\text{depositor} \triangleright\!\!\!< \text{account})$

Restano i clienti che hanno un conto in una filiale a Pisa, ma non in tutte.

Togliendo dai clienti del data base i clienti ottenuti, restano i clienti che hanno un conto in tutte le filiali di Pisa.

111

## Definizione

- Siano  $r(R)$  e  $s(S)$  relazioni con  $R \sqsubseteq S$ ,  $r \div s$  è una relazione su  $R-S$ ; una tupla  $t \in r \div s$  iff
  - $t \in \Pi_{R-S}(r)$
  - $\forall t' \in s, \exists t'' \in r$  tale che
    - $t'[S] = t''[S]$  e
    - $t'[R-S] = t$

112

## Relazioni derivate

113

## Relazioni derivate

- Relazioni di base: contenuto autonomo
- Relazioni derivate:
  - relazioni il cui contenuto è funzione del contenuto di altre relazioni (ed è definito per mezzo di interrogazioni)
  - Due tipi di relazioni derivate:
    - viste materializzate
    - viste virtuali (o viste)

114

## Viste materializzate

- relazioni derivate memorizzate nella base di dati
  - vantaggi:
    - immediatamente disponibili per le interrogazioni
  - svantaggi:
    - ridondanti
    - appesantiscono gli aggiornamenti
    - Non sempre supportate dai DBMS

115

## Viste virtuali

- Viste virtuali
  - sono supportate dai DBMS (tutti)
  - una interrogazione su una vista viene eseguita "ricalcolando" la vista (o quasi)

116

## Esempio

Afferenza

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B
Verdi	C

Direzione

Reparto	Capo
A	Mori
B	Bruni
C	Leoni

- una vista

Supervisione =

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

117

## Interrogazioni sulle viste

- Sono eseguite sostituendo alla vista la sua definizione:

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

viene eseguita come

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

118

## Viste, motivazioni

- Strumento di programmazione :
  - si può semplificare la scrittura di interrogazioni: espressioni complesse e sottoespressioni ripetute
- L'utilizzo di viste virtuali non influisce sull'efficienza delle interrogazioni

119

## Viste come strumento di programmazione

- Trovare gli impiegati che hanno lo stesso capo di Rossi
- Senza vista:

$$\begin{aligned} & \pi_{\text{Impiegato}, \text{Capo}} (\text{Afferenza} \triangleright \triangleleft \text{Direzione}) \triangleright \triangleleft \\ & \quad \pi_{\text{Capo}} ( \\ & \quad \sigma_{\text{Impiegato} = 'Rossi'} (\text{Afferenza} \triangleright \triangleleft \text{Direzione})) \end{aligned}$$

- Con la vista:

$$\begin{aligned} & \pi_{\text{Impiegato}, \text{Capo}} (\text{Supervisione}) \triangleright \triangleleft \\ & \quad \pi_{\text{Capo}} ( \\ & \quad \sigma_{\text{Impiegato} = 'Rossi'} (\text{Supervisione})) \end{aligned}$$

120

Con le viste posso evitare di usare l'operatore di ridenominazione, ad esempio, posso scrivere

**Capi := Impiegati**

Ed usare due istanze della relazione impiegati con nome diverso

121

## Estensioni dell'algebra

97

## Algebra relazionale estesa

- Il modello relazionale può essere facilmente esteso a comprendere gli operatori SQL non direttamente riconducibili agli operatori algebrici introdotti
- Questa estensione non modifica il funzionamento del modello

98

## Estensioni (1)

- Join esterno (left, right, full)
- permette di generare valori null per mezzo delle espressioni dell'algebra relazionale per modellare le informazioni mancanti
- Left( $=\triangleright\triangleleft$ ): solo tuple dell'operando sinistro sono riempite con NULL
- Right( $\triangleright\triangleleft =$ ): idem per l'operando destro
- Full( $=\triangleright\triangleleft =$ ): per entrambi gli operandi

99

## Esempi

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

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

100

## Uso del join esterno

$$\begin{aligned} & \pi_{\text{Capo}} (\sigma_{\text{Matricola}=\text{Null}} (\text{Supervisione} \\ & = \triangleright \triangleleft_{\text{Impiegato}=\text{Matricola}} \\ & \sigma_{\text{Stipendio} \leq 40000} (\text{Impiegati}))) \end{aligned}$$

101

## Estensioni (2)

- Proiezione generalizzata
  - $\pi_{F1,F2,F3}(E)$
  - F1, F2, F3 sono espressioni aritmetiche su attributi di E (che è una qualunque espressione dell'algebra) e costanti

102

Conto		
Cliente	Credito	Spese
Andrea	6000	1000
Andrea	4000	500
Maria	10000	2000
Anna	3000	1500
Filippo	3000	1000
Luigi	5000	1800
Franco	5000	2000
Maria	6000	2000
Andrea	10000	5000
Anna	5000	1000

103

## Esempio proiezione

- Si può scrivere ad esempio
  - $\pi_{\text{Cliente}, \text{Credito-Spese}}(\text{Conto})$
- ed ottenere il seguente risultato

104

	<b>Cliente</b>	<b>Credito-Spesa</b>
	<b>Andrea</b>	<b>5000</b>
	<b>Andrea</b>	<b>3500</b>

105

## Estensioni (3)

- Funzioni aggregate
  - Si possono usare nelle espressioni alcuni nomi di funzioni (operatori) che si applicano a (multi)insiemi e producono un valore come risultato

106

## Operatori aggregati

- sum, count, min, max
  - $\text{sum}_{\text{Spese}}(\text{Conto})$
  - $\text{count}_{\text{Cliente}}(\text{Conto})$
  - $\text{max}_{\text{Credito}}(\text{Conto})$
  - $\text{count-distinct}_{\text{Cliente}}(\text{Conto})$

107

## Raggruppamento

- Si possono raggruppare gli elementi di una relazione usando un'operatore apposito  
 $\text{Cliente} G_{\text{sum}(\text{Credito})}(\text{Conto})$
- Cliente è l'attributo su cui si fa il raggruppamento, sum è la funzione aggregata che si applica all'attributo Credito, Conto è la relazione su cui si applica il tutto.
- Si possono avere più attributi a sinistra e più funzioni a destra di G

108

## Un altro operatore derivato: divisione

- In generale, la divisione è utile per interrogazioni di tipo "universale"
- Vogliamo trovare i nomi dei clienti che hanno un conto corrente in tutte le filiali di banca di Pisa.
- Le relazioni sono
  - Branch(bank\_name, branch\_name, branch\_city)
  - Account(branch\_name, bank\_name, account\_number, branch\_city)
  - Depositor(account\_number, customer\_name)

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

109

## Cosa si intende

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

Equivale a

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

110

$\Pi_{CN}(\text{depositor} \triangleright\!\!\!< \text{account}) \triangleright\!\!\!< \Pi_{BN}(\sigma_{BC=Pisa'}(\text{branch}))$

Si combinano tutti i clienti presenti nel data base con le filiali di Pisa;

togliendo da questo insieme le coppie (cliente,filiale) presenti nel data base, cioè

$\Pi_{CN,BN}(\text{depositor} \triangleright\!\!\!< \text{account})$

Restano i clienti che hanno un conto in una filiale a Pisa, ma non in tutte.

Togliendo dai clienti del data base i clienti ottenuti, restano i clienti che hanno un conto in tutte le filiali di Pisa.

111

## Definizione

- Siano  $r(R)$  e  $s(S)$  relazioni con  $R \sqsubseteq S$ ,  $r \div s$  è una relazione su  $R-S$ ; una tupla  $t \in r \div s$  iff
  - $t \in \Pi_{R-S}(r)$
  - $\forall t' \in s, \exists t'' \in r$  tale che
    - $t'[S] = t''[S]$  e
    - $t'[R-S] = t$

112

## Relazioni derivate

113

## Relazioni derivate

- Relazioni di base: contenuto autonomo
- Relazioni derivate:
  - relazioni il cui contenuto è funzione del contenuto di altre relazioni (ed è definito per mezzo di interrogazioni)
  - Due tipi di relazioni derivate:
    - viste materializzate
    - viste virtuali (o viste)

114

## Viste materializzate

- relazioni derivate memorizzate nella base di dati
  - vantaggi:
    - immediatamente disponibili per le interrogazioni
  - svantaggi:
    - ridondanti
    - appesantiscono gli aggiornamenti
    - Non sempre supportate dai DBMS

115

## Viste virtuali

- Viste virtuali
  - sono supportate dai DBMS (tutti)
  - una interrogazione su una vista viene eseguita "ricalcolando" la vista (o quasi)

116

## Esempio

Afferenza

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B
Verdi	C

Direzione

Reparto	Capo
A	Mori
B	Bruni
C	Leoni

- una vista

Supervisione =

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

117

## Interrogazioni sulle viste

- Sono eseguite sostituendo alla vista la sua definizione:

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

viene eseguita come

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

118

## Viste, motivazioni

- Strumento di programmazione :
  - si può semplificare la scrittura di interrogazioni: espressioni complesse e sottoespressioni ripetute
- L'utilizzo di viste virtuali non influisce sull'efficienza delle interrogazioni

119

## Viste come strumento di programmazione

- Trovare gli impiegati che hanno lo stesso capo di Rossi
- Senza vista:

$$\begin{aligned} & \pi_{\text{Impiegato}, \text{Capo}} (\text{Afferenza} \triangleright \triangleleft \text{Direzione}) \triangleright \triangleleft \\ & \quad \pi_{\text{Capo}} ( \\ & \quad \sigma_{\text{Impiegato} = 'Rossi'} (\text{Afferenza} \triangleright \triangleleft \text{Direzione})) \end{aligned}$$

- Con la vista:

$$\begin{aligned} & \pi_{\text{Impiegato}, \text{Capo}} (\text{Supervisione}) \triangleright \triangleleft \\ & \quad \pi_{\text{Capo}} ( \\ & \quad \sigma_{\text{Impiegato} = 'Rossi'} (\text{Supervisione})) \end{aligned}$$

120

*Con le viste posso evitare di usare l'operatore di ridenominazione, ad esempio, posso scrivere*

**Capi := Impiegati**

*Ed usare due istanze della relazione impiegati con nome diverso*

121

Corso di Laurea in Ingegneria  
Informatica  
*Fondamenti di informatica II*  
Modulo "*Basi di dati*"  
a.a. 2018-2019

Docente: Gigliola Vaglini  
Docente laboratorio: Francesco  
Pistolesi

1

Lezione 4

Progettazione di basi di dati:  
metodologie e modelli

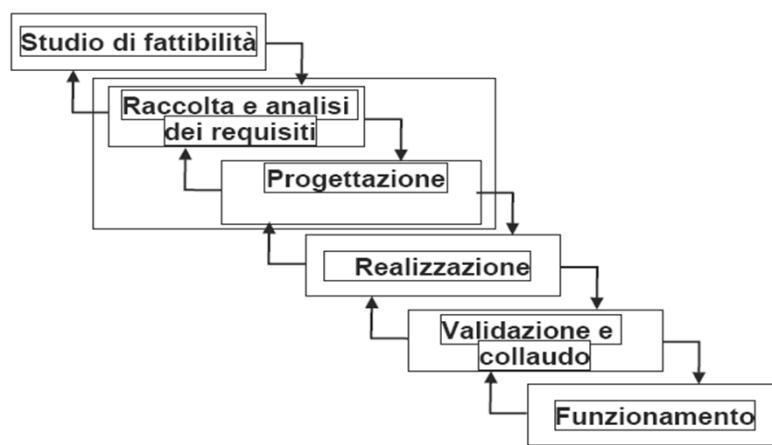
2

## Processo di sviluppo di sistemi software

- Lo sviluppo di sistemi software in generale, e di sistemi informativi in particolare, è un'attività che comprende diverse fasi.
- Ciclo di vita: sequenza di attività, anche ripetute ciclicamente, svolte da analisti, progettisti, utenti, nello sviluppo e nell'uso dei sistemi sw

3

## Primo modello del ciclo di vita Waterfall model



4

## Fasi del ciclo di vita

- Studio di fattibilità: definizione costi e priorità
- Raccolta e analisi dei requisiti: studio delle proprietà del sistema
- Progettazione: individuazione dei dati e delle funzioni
- Realizzazione
- Validazione e collaudo: sperimentazione
- Funzionamento: il sistema diventa operativo

5

## La progettazione è una fase del ciclo di vita

- Adesso ci focalizziamo sul passo 3 del ciclo di vita più alcuni aspetti di raccolta e analisi dei requisiti.
- La progettazione di un sistema software consta fondamentalmente di due aspetti
  - progettazione dei dati
  - progettazione delle applicazioni
- Nel caso di sistemi informativi il progetto dei dati ha un ruolo centrale

6

## Un buon progetto

- Per garantire prodotti di buona qualità è opportuno seguire una
  - metodologia di progetto, basata su
    - modelli per rappresentare i dati che siano facili da usare
    - decomposizione delle attività in fasi (o livelli)
    - strategie e criteri di scelta nei vari passi

7

## Progettare per livelli di astrazione

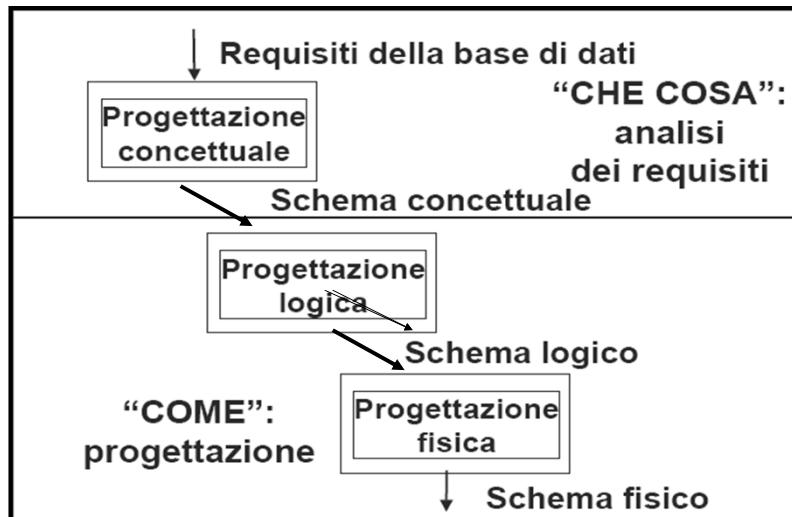
- **Livello concettuale.** Esprime i requisiti di un sistema in una descrizione adatta all'analisi dal punto di vista esterno
- **Livello logico.** Evidenzia l'organizzazione dei dati dal punto di vista del loro contenuto informativo, descrivendo la struttura di ciascun record e i collegamenti tra record diversi.
- **Livello fisico.** A questo livello la base di dati è vista come un insieme di blocchi fisici su disco. Qui viene decisa l'allocazione dei dati e le modalità di memorizzazione dei dati sul disco.

8

## Schemi di basi di dati

- I prodotti della varie fasi della progettazione sono schemi della base di dati, ognuno basato su un insieme di costrutti per organizzare i dati e descriverne la dinamica
  - Schema concettuale
  - Schema logico
  - Schema fisico

9



10

## Raccolta e analisi dei requisiti

- acquisizione dei requisiti: il reperimento dei requisiti è un'attività difficile e non standardizzabile
- analisi dei requisiti: l'attività di analisi inizia con i primi requisiti raccolti e spesso indirizza verso altre acquisizioni

11

## Come si acquisiscono i requisiti

- direttamente dagli utenti
  - interviste
  - documentazione apposita
- da documentazione esistente:
  - normative (leggi, regolamenti di settore)
  - regolamenti interni, procedure aziendali
  - realizzazioni preesistenti

12

## Interazione con gli utenti

- Problemi

- utenti diversi possono fornire informazioni diverse
- utenti a livello più alto hanno spesso una visione più ampia ma meno dettagliata
- spesso l'acquisizione dei requisiti avviene "per raffinamenti successivi"

13

## Interazione con gli utenti

- Spunti:

- effettuare spesso verifiche di comprensione e coerenza
- verificare anche per mezzo di esempi (generali e relativi a casi limite)
- richiedere definizioni e classificazioni
- far evidenziare gli aspetti essenziali rispetto a quelli marginali (ranking dei requisiti)

14

## Interazione con gli utenti tramite documentazione

- Regole generali:
  - standardizzare la struttura delle frasi
  - separare le frasi sui dati da quelle sulle funzioni
  - organizzare termini e concetti
    - costruire un glossario dei termini
      - unificare i termini (individuare i sinonimi)
      - rendere esplicito il riferimento fra termini
    - riorganizzare le frasi per concetti

15

## Un esempio

### Società di formazione (1)

**Si vuole realizzare una base di dati per una società che eroga corsi: di ogni corso vogliamo rappresentare i dati dei partecipanti 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 già frequentato (le materie sono in tutto circa 200) e il giudizio finale.**

16

### **Società di formazione (2)**

**Rappresentiamo anche i corsi attualmente attivi 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.**

17

### **Società di formazione (3)**

**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.**

18

## Glossario dei termini

Termine	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.	Materia	Docente
Società	Ente presso cui i partecipanti lavorano o hanno lavorato	Posti	Partecipante

19

Strutturazione dei requisiti  
in gruppi di frasi omogenee  
(per concetti)

20

### **Frasi di carattere generale**

**Si vuole realizzare una base di dati per una società che eroga corsi: di ogni corso vogliamo rappresentare i dati dei partecipanti e dei docenti.**

21

### **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.**

22

### **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.**

23

### **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.**

24

### **Frasi relative ai docenti**

**Per i docenti (circa 300), rappresentiamo il cognome, l'età, la città 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 società di formazione o collaboratori esterni.**

25

### **Modello concettuale**

26

## Modelli concettuali

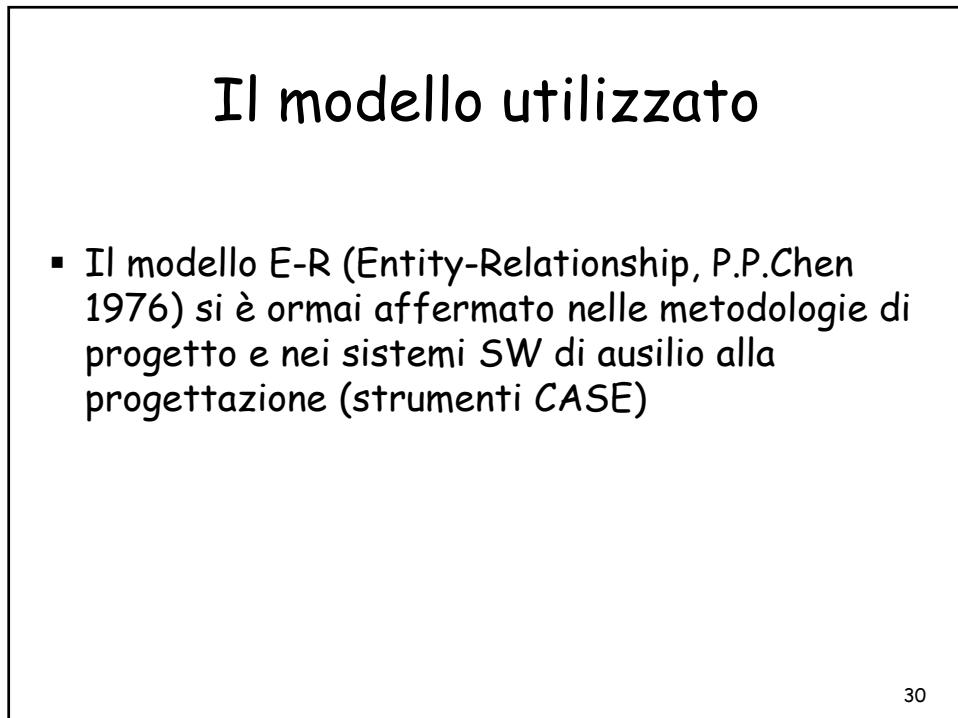
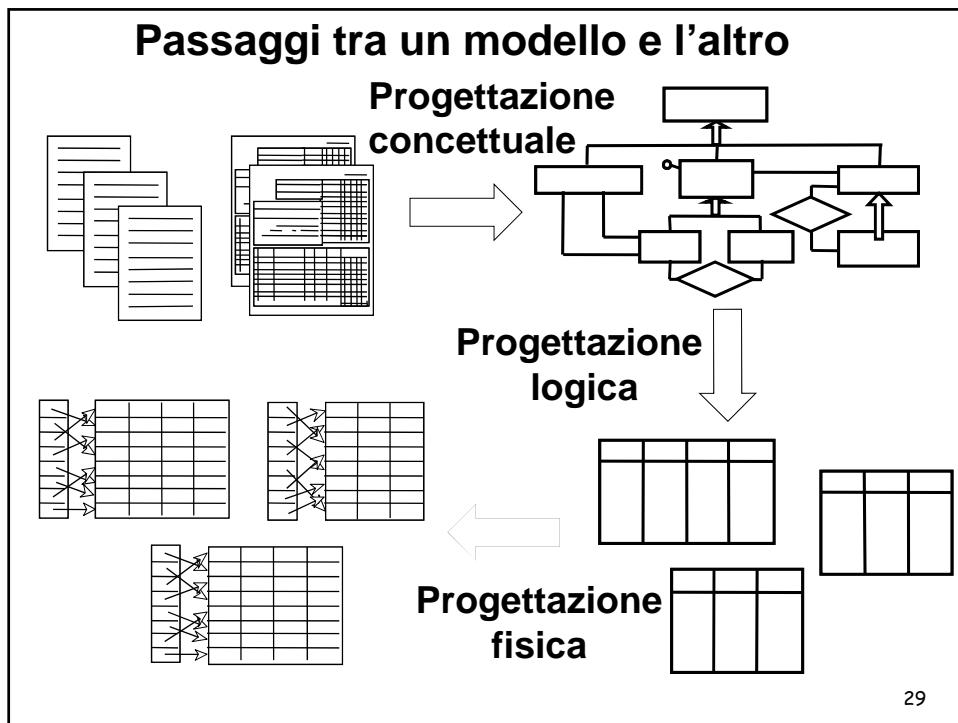
- I dati sono rappresentati in modo più vicino al modo di pensare umano
  - cercano di replicare i concetti del mondo reale
- Prevedono efficaci rappresentazioni grafiche (utili anche per la documentazione e la comunicazione)
  - i modelli successivi sono più rigidi, partendo da loro ci perderemmo nei dettagli

27

## La progettazione concettuale

- A questo livello i requisiti di un qualsiasi sistema informatico (anche dati in modo informale) sono specificati in modo:
  - **formale**: cioè in modo non ambiguo, ma adeguato a catturare le caratteristiche fondamentali del mondo da descrivere
  - **integrato**: la descrizione si riferisce alla totalità dell'ambiente

28



## Costrutti del modello E-R

- Costrutti di base
  - Entità
  - Relationship
  - Attributo
- Altri costrutti
  - Identificatore
  - Generalizzazione
  - ....

31

## Entity

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

32

## Occorrenza di Entità

- Entità
  - classe di oggetti, persone, ... "omogenei"
    - es. l'entità "impiegato"
- Occorrenza (o istanza) di entità
  - elemento della classe (l'oggetto, la persona, ..., non un valore dei dati legati all'oggetto)
    - es. un "impiegato", non so nulla di lui, ma esiste

33

## Rappresentazione grafica di entità

Impiegato

Dipartimento

Città

Vendita

34

## Entità: caratteristiche

- Ogni entità ha un nome che la identifica univocamente nello schema:
  - nomi espressivi
  - opportune convenzioni
    - singolare

35

## Relationship

- Legame logico fra due o più entità, rilevante nell'applicazione di interesse
- Esempi:
  - Residenza (fra persona e città)
  - Esame (fra studente e corso)
- Chiamata anche:
  - relazione, correlazione, associazione

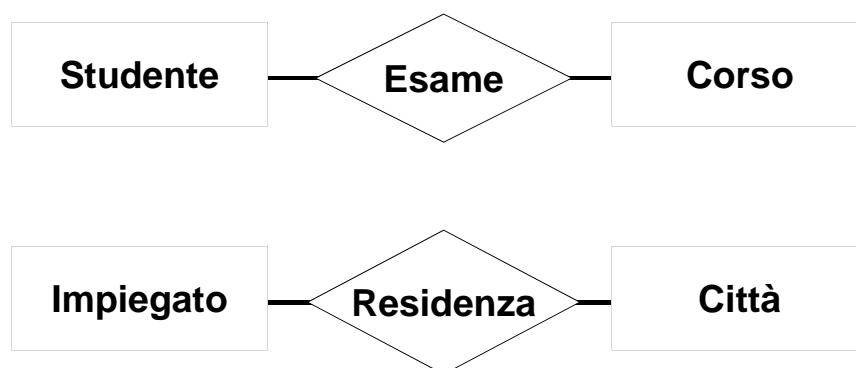
36

## Relationship: occorrenze

- Un'occorrenza di relationship binaria è una coppia di occorrenze di entità, una per ogni entità coinvolta
- Un'occorrenza di relationship n-aria è una n-upla di occorrenze di entità, una per ciascuna entità coinvolta
  - non ci possono essere occorrenze ripetute sulle stesse occorrenze di entità

37

## Rappresentazione grafica di relationship



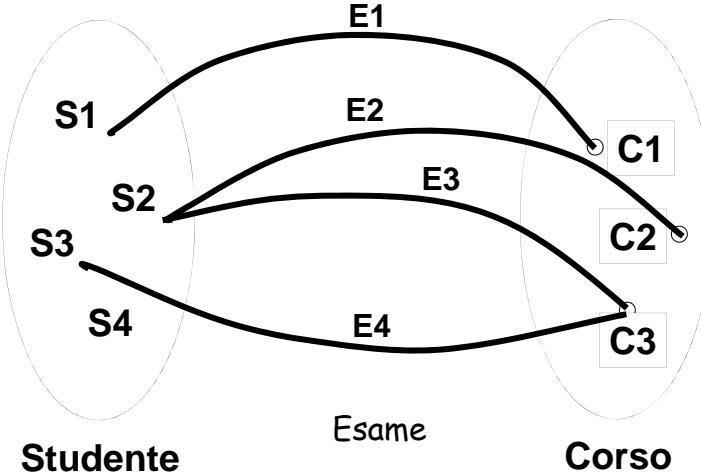
38

## Relationship: caratteristiche

- Ogni relationship ha un nome che la identifica univocamente nello schema:
  - nomi espressivi
  - opportune convenzioni
    - singolare
    - sostantivi invece che verbi (se possibile) per non dare un verso alla relationship

39

## Esempi di occorrenze (di entità e di relationship)



40

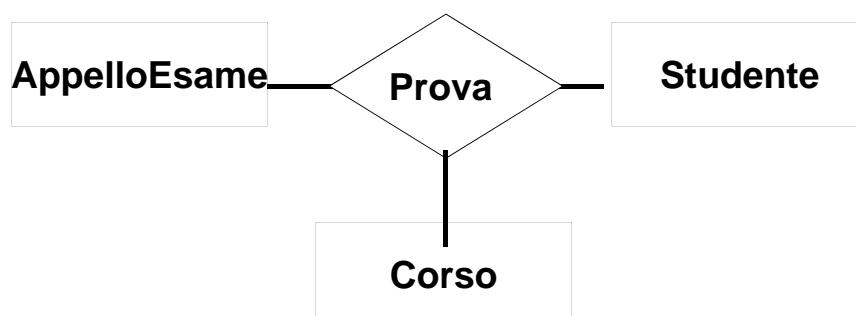
## Relationship



- ✓ Problema.
- ✓ Ma se uno studente sostiene più volte lo stesso esame prima di essere promosso, come si vede?
- ✓ Non si può vedere perché risulterebbero coppie di occorrenze di entità uguali
- ✓ Esame, studente, corso devono diventare tutte entità, messe in relazione dalla relationship Prova, ad esempio

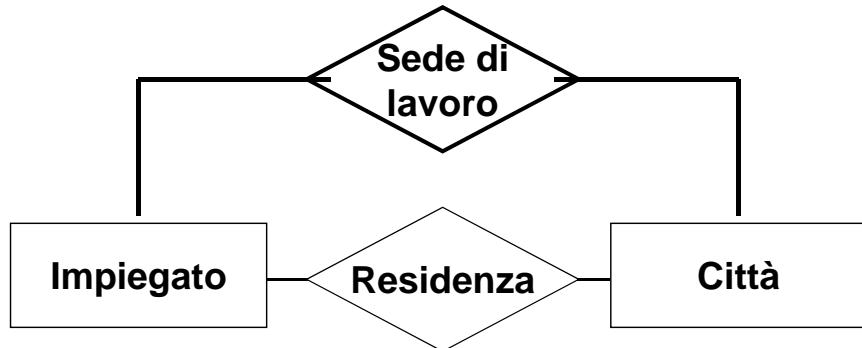
41

## Soluzione: Relationship n-aria



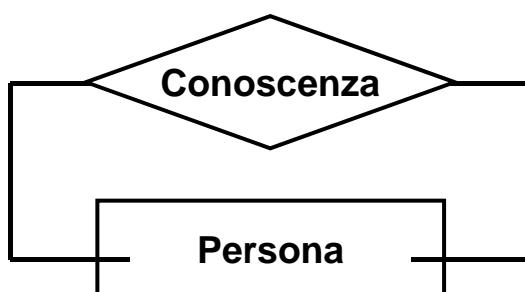
42

## Relationship diverse sulle stesse entità



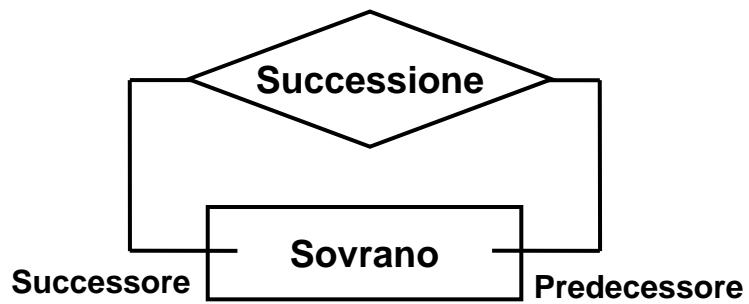
43

## Relationship ricorsiva



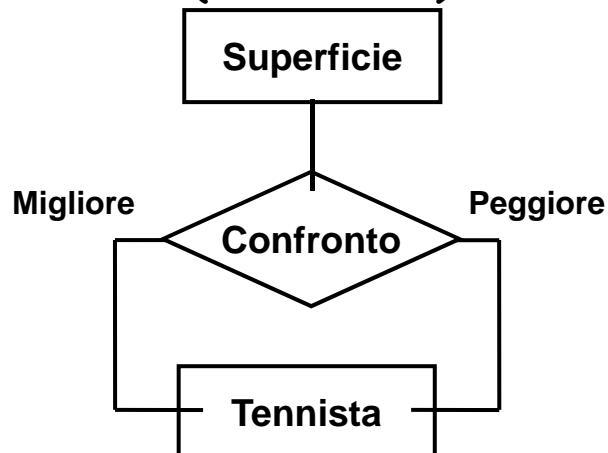
44

## Relationship ricorsiva con “ruoli”



45

## Relationship mista (ternaria)



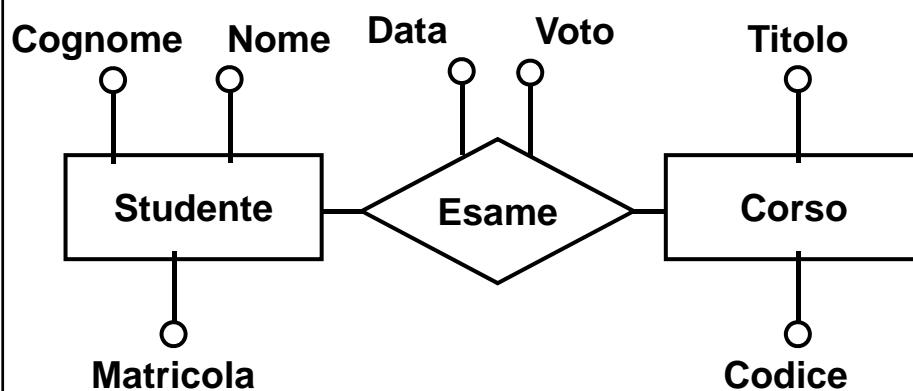
46

## Attributo

- Proprietà elementare di un'entità o di una relationship
- Associa ad ogni occorrenza di entità o relationship un valore appartenente a un insieme detto dominio dell'attributo

47

## Attributi: rappresentazione grafica



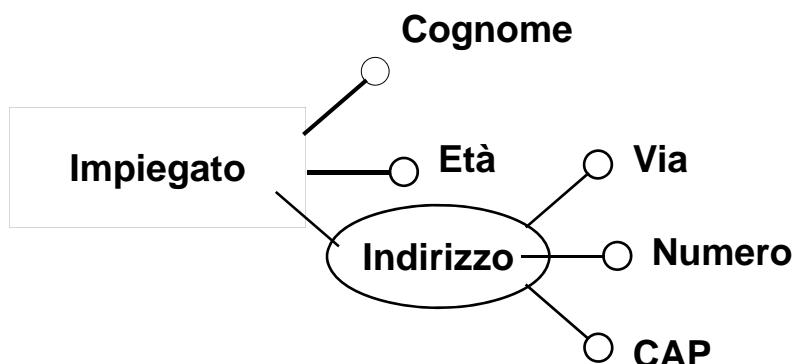
48

## Attributi composti

- Raggruppano attributi di una medesima entità o relationship che presentano affinità nel loro significato o uso
- Esempio:
  - Via, Numero civico e CAP formano un Indirizzo

49

## Rappresentazione grafica

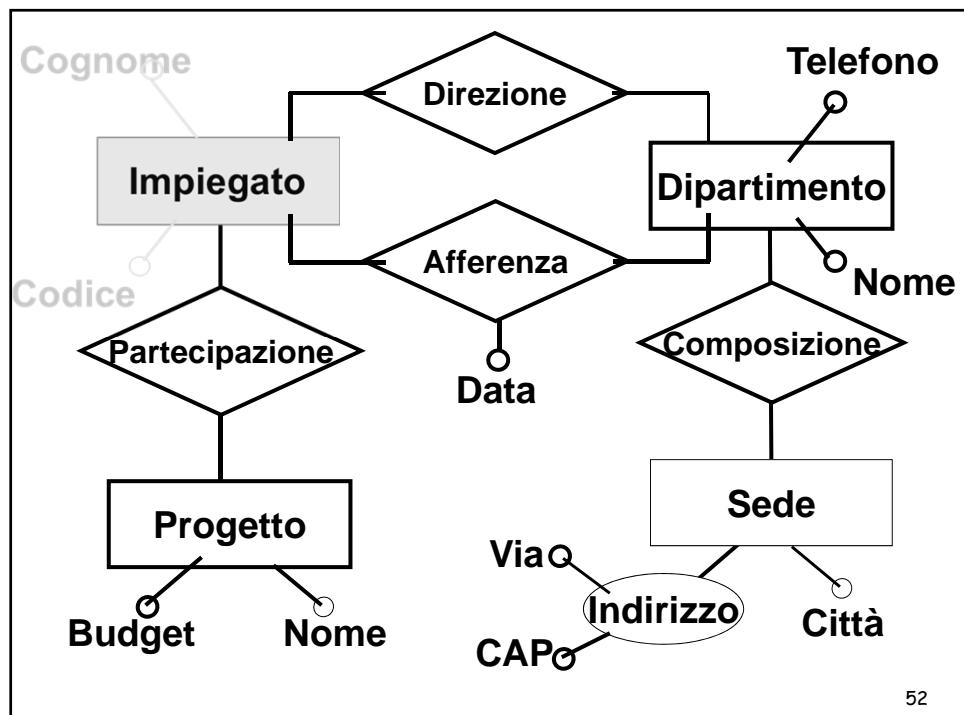


50

## Schema E-R con solo i costrutti base

- Si vuole descrivere l'organizzazione di un'azienda
  - Con sedi diverse
  - Ogni sede è composta di vari dipartimenti
  - Gli impiegati dell'azienda afferiscono ai vari dipartimenti e un'impiegato li dirige
  - Gli impiegati lavorano su progetti
  - Ogni entità o relationship può avere vari attributi.

51



52

## Concetti inesprimibili

- Un dipartimento ha un solo direttore
- Un impiegato può afferire ad solo dipartimento
- ...

53

## Altri costrutti del modello E-R

- Cardinalità
  - di relationship
  - di attributo
- Identificatore
  - interno
  - esterno
- Generalizzazione

54

## Cardinalità di relationship

- Coppia di valori associati a ogni entità che partecipa a una relationship
  - specificano il numero minimo e massimo di occorrenze della relationship cui ciascuna occorrenza di entità può partecipare

55

## Esempio di cardinalità

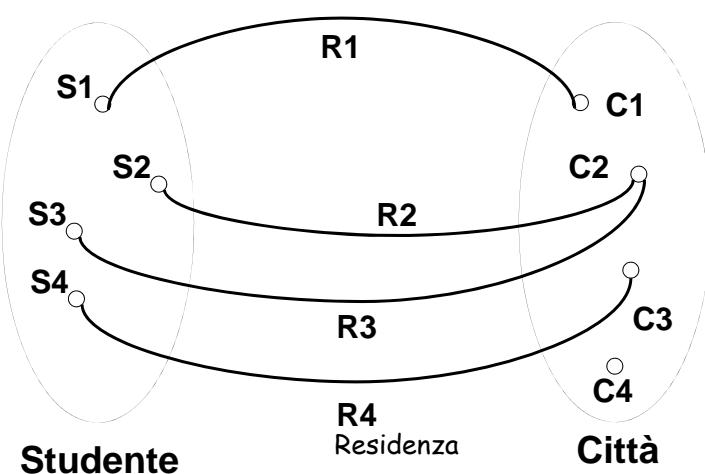


56

- 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

57

## Occorrenze di Residenza



58

## Cardinalità di Residenza



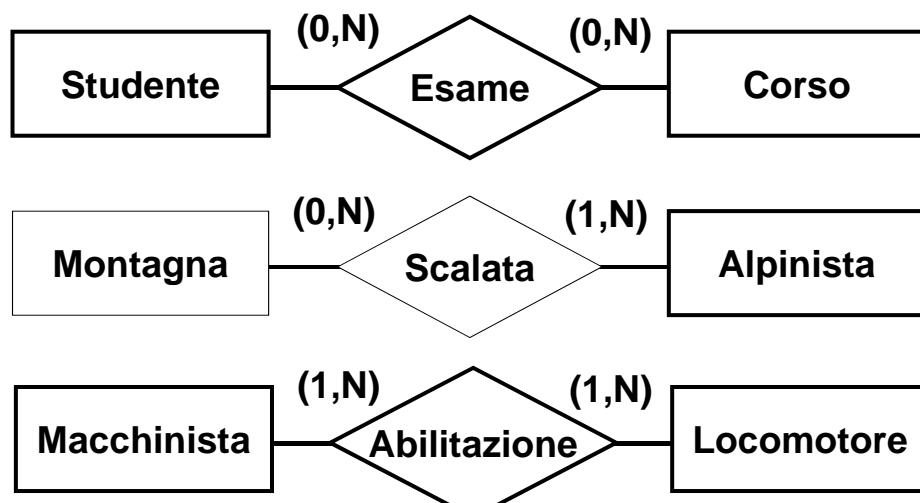
59

## Tipi di relationship

- Con riferimento alle cardinalità massime, possiamo caratterizzare una relationship come:
  - uno a uno
  - uno a molti
  - molti a molti

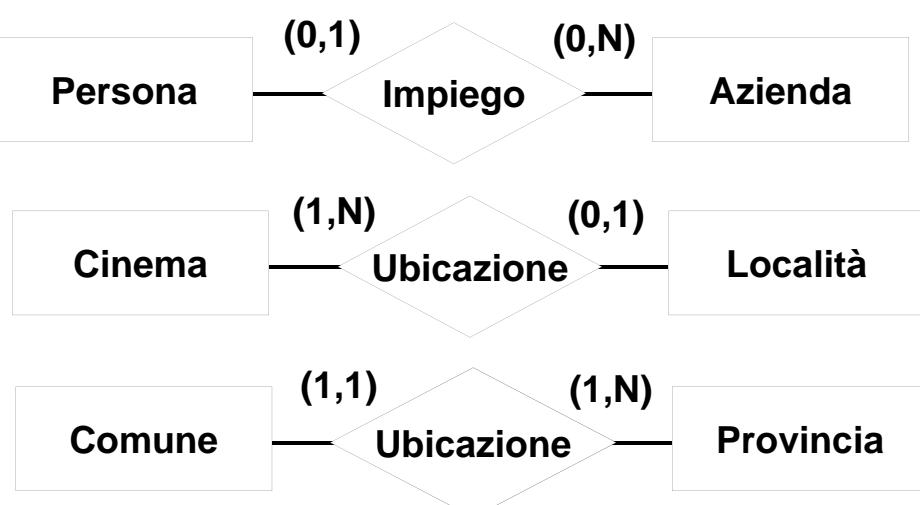
60

## Relationship "molti a molti"



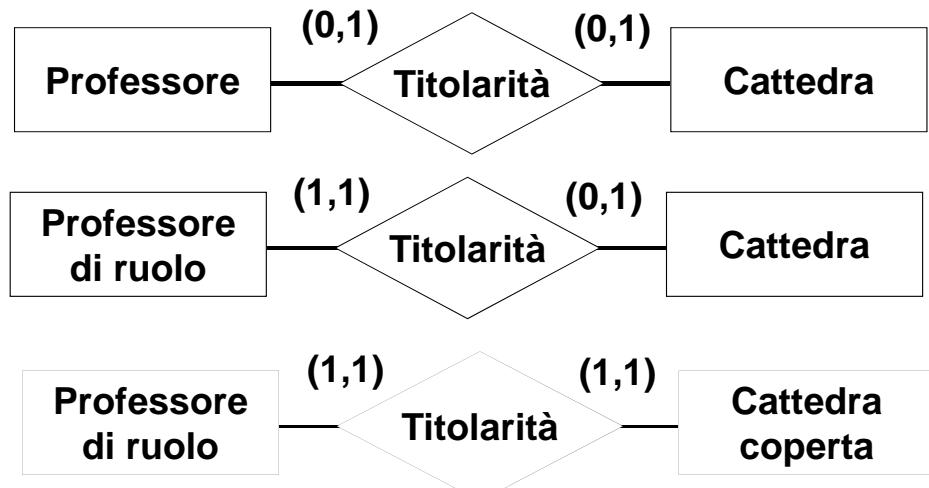
61

## Relationship "uno a molti"



62

## Relationship "uno a uno"



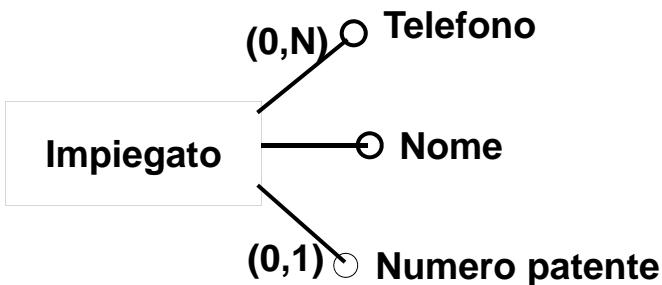
63

## Cardinalità di attributi

- È possibile associare una cardinalità anche agli attributi, con due scopi:
  - indicare opzionalità ("informazione incompleta")
  - indicare attributi multivalore

64

## Rappresentazione grafica



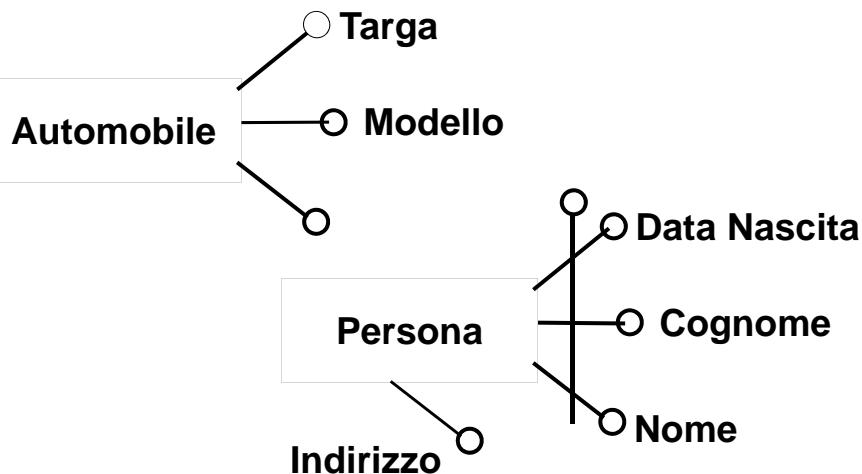
65

## Identificatore di una entità

- “strumento” per l’identificazione univoca delle occorrenze di un’entità
- costituito da:
  - attributi dell’entità
    - identificatore interno (o chiave)
  - (attributi +) la chiave di entità esterne attraverso relationship
    - identificatore esterno

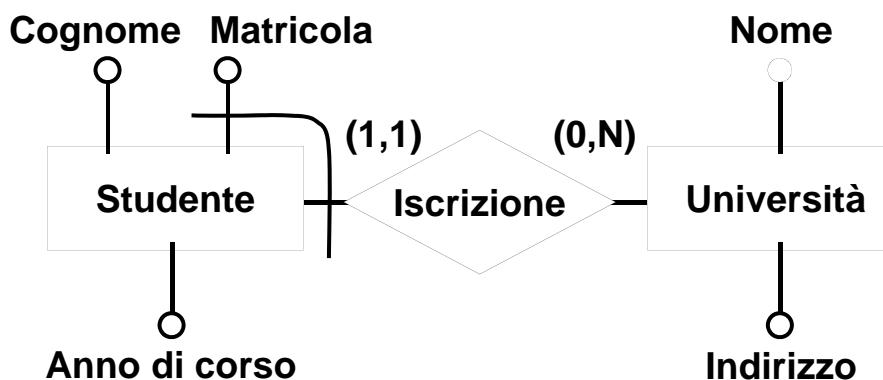
66

## Identifieri interni



67

## Identificatore esterno



68

## Identifieri: caratteristiche

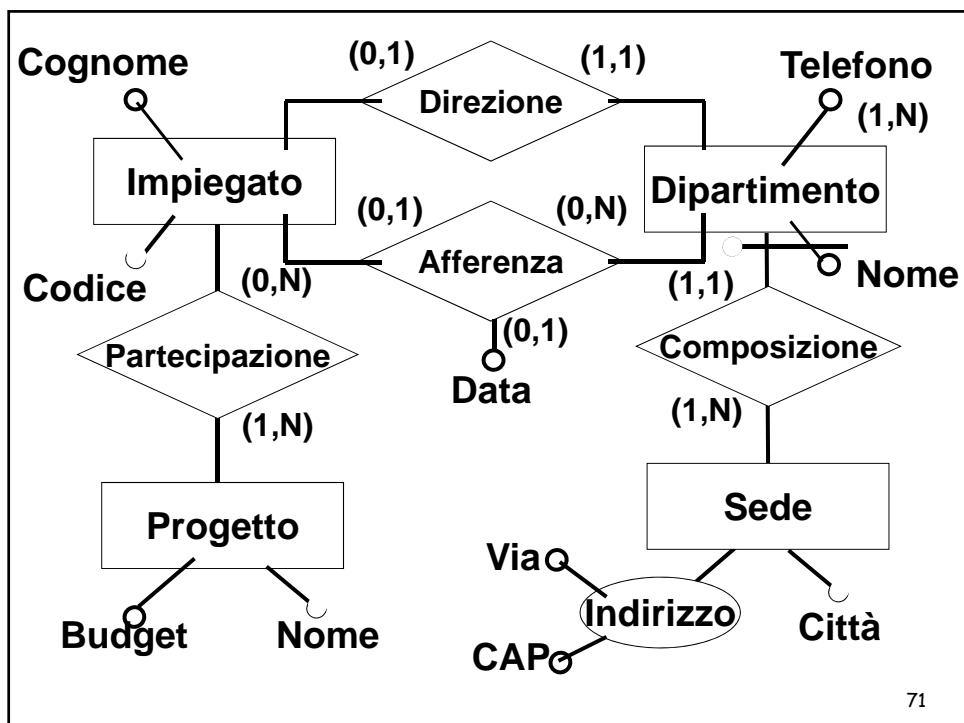
- ogni entità deve possedere almeno un identificatore, ma può averne in generale più di uno
- una identificazione esterna è possibile solo attraverso una relationship a cui l'entità da identificare partecipa con cardinalità (1,1)

69

## Schema E-R con costrutti di base, identifieri e cardinalità

- Riprendiamo l'esempio dell'azienda ed introduciamo la cardinalità per attributi e relazioni, ed individuiamo gli identifieri delle entità coinvolte.

70



71

- Il direttore di un dipartimento afferisce a quel dipartimento?
- ...

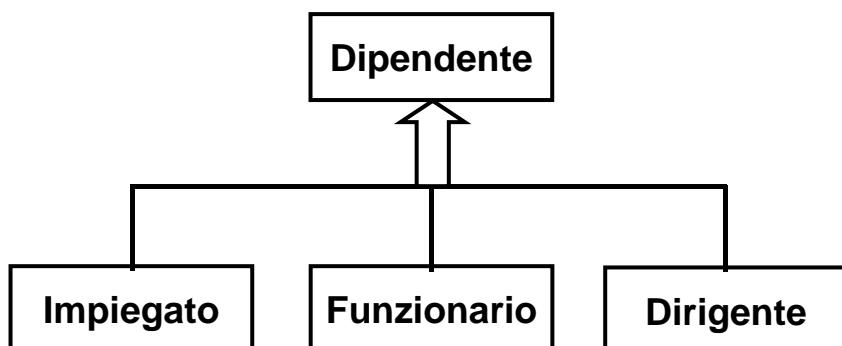
72

## Generalizzazione

- Mette 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

73

## Generalizzazione: rappresentazione grafica



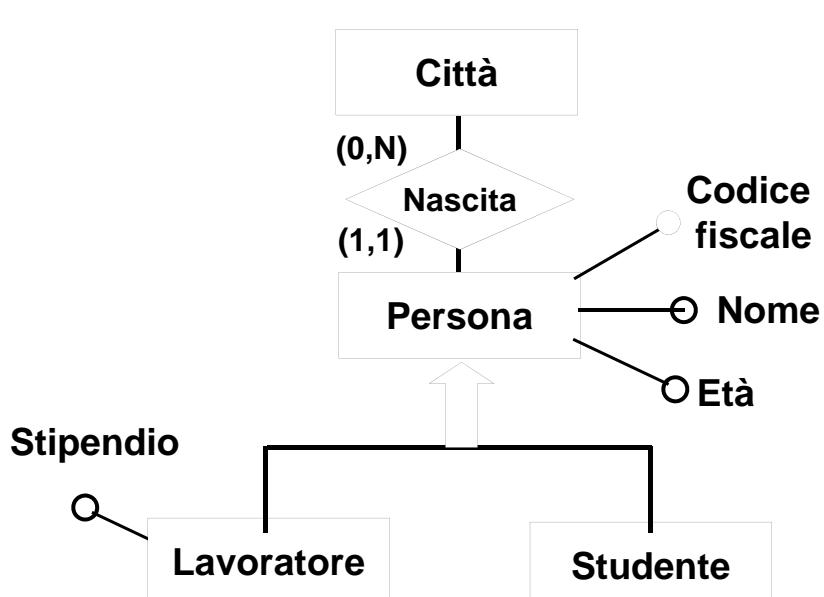
74

## Proprietà delle generalizzazioni

Se E (genitore) è generalizzazione di E<sub>1</sub>, E<sub>2</sub>, ..., E<sub>n</sub> (figlie):

- ogni proprietà di E è significativa per E<sub>1</sub>, E<sub>2</sub>, ..., E<sub>n</sub>
- ogni occorrenza di E<sub>1</sub>, E<sub>2</sub>, ..., E<sub>n</sub> è occorrenza anche di E

75



76

## Ereditarietà

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

77

## Tipi di generalizzazioni

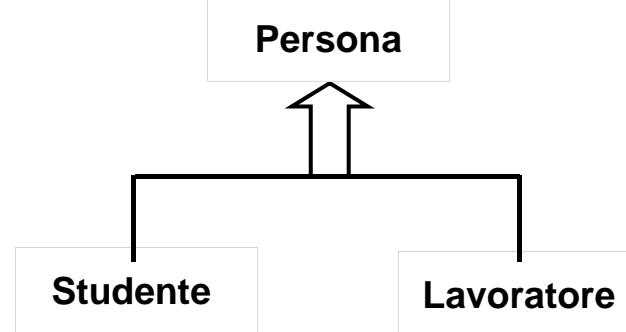
- totale se ogni occorrenza dell'entità genitore è occorrenza di almeno una delle entità figlie, altrimenti è parziale
- esclusiva se ogni occorrenza dell'entità genitore è occorrenza di al più una delle entità figlie, altrimenti è sovrapposta

78

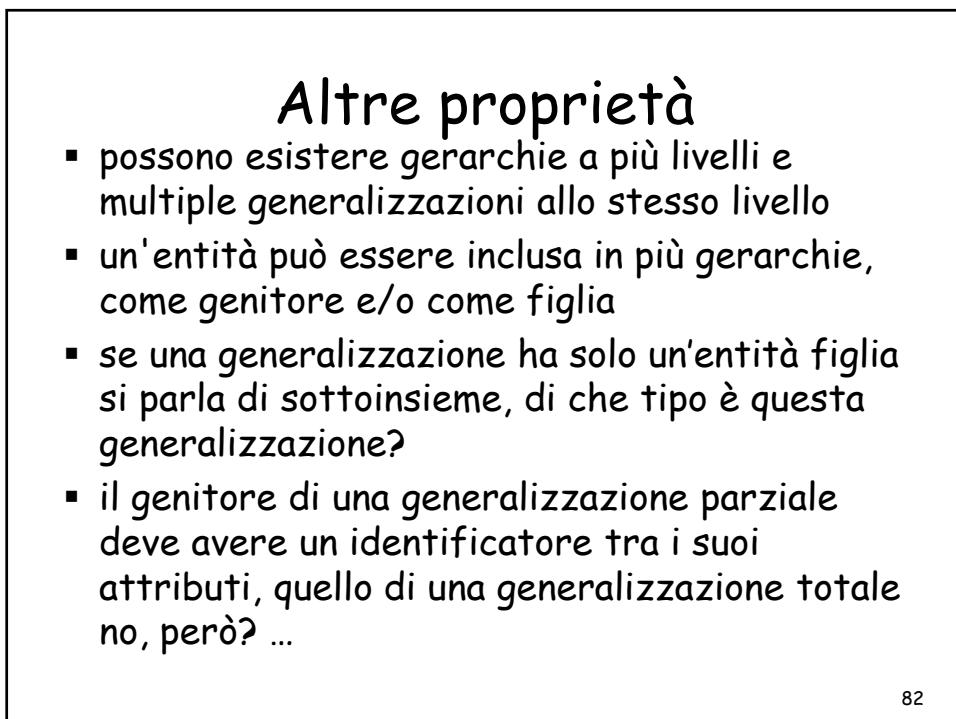
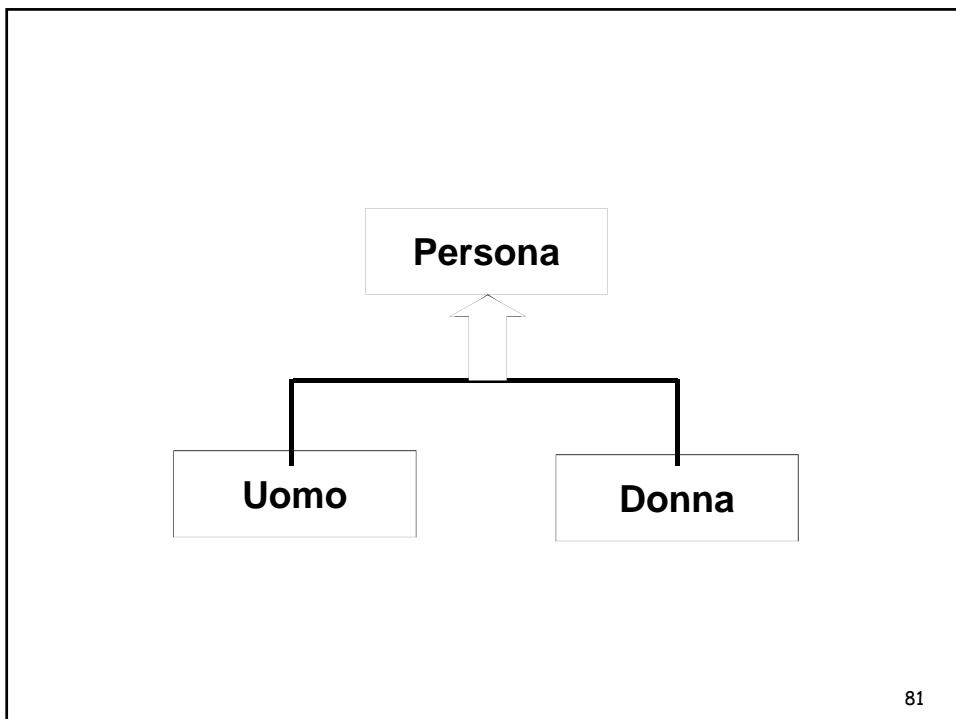
## Tipi di generalizzazioni

- Consideriamo negli schemi solo generalizzazioni esclusive (si puo' sempre trasformare una generalizzazione sovrapposta in una esclusiva) e distinguiamo fra generalizzazioni parziali e totali

79



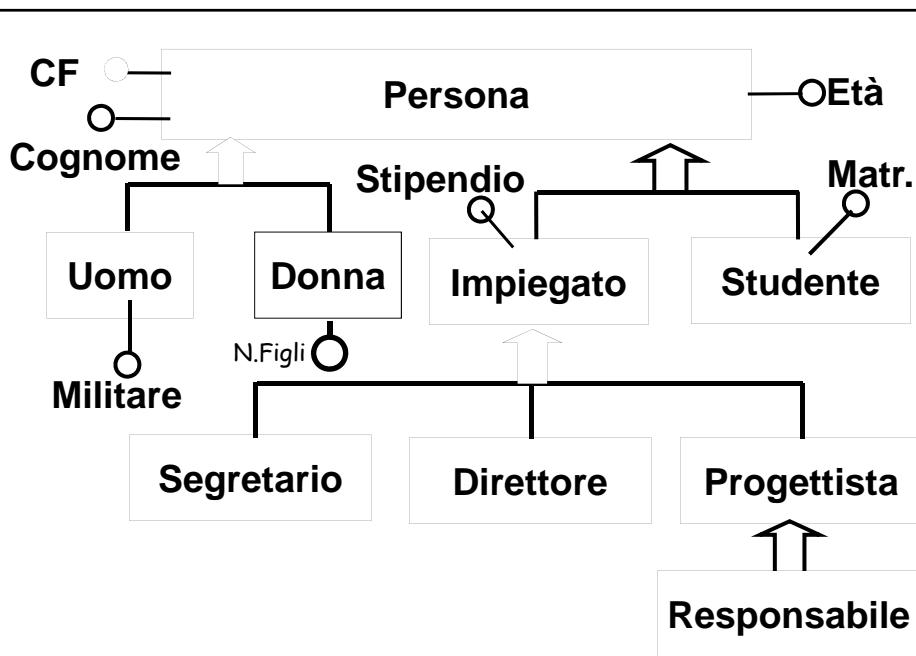
80



## Esercizio

- Le persone hanno CF, cognome ed età, gli uomini la posizione militare, le donne il numero dei figli;
- 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) hanno un numero di matricola;
- esistono persone che non sono né impiegati né studenti (ma i dettagli non ci interessano)

83



84

## Documentazione associata agli schemi concettuali

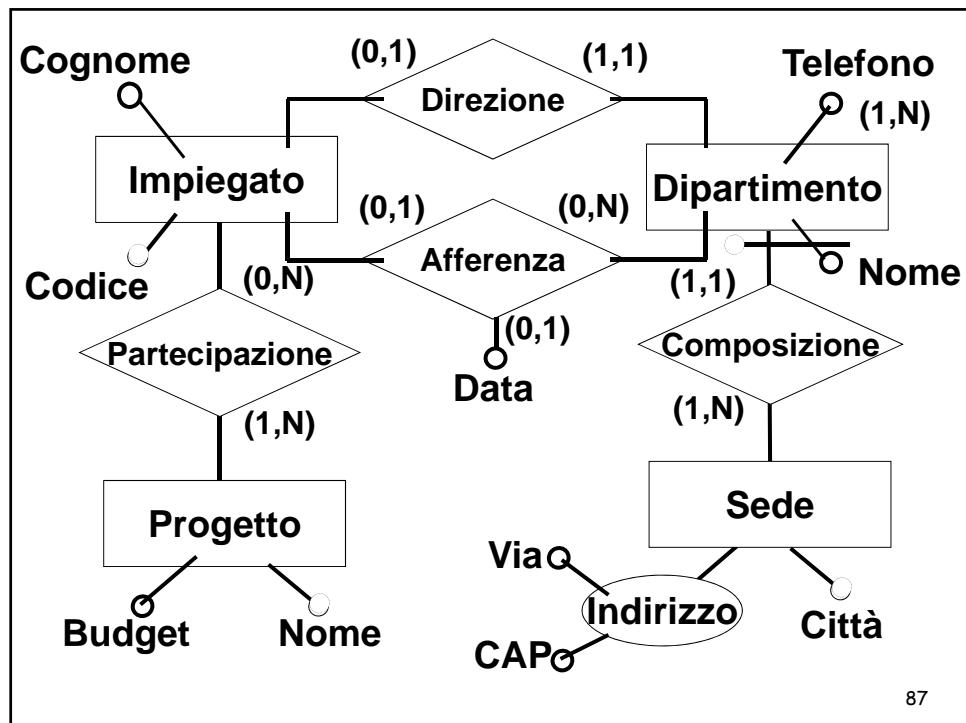
- Uno schema E-R non è quasi mai sufficiente da solo a rappresentare tutti i dettagli di un'applicazione
- Ci sono vincoli non esprimibili
- È necessario associare una documentazione di supporto

85

## Documentazione

- Dizionario dei dati
  - entità
  - relationship
- Regole aziendali
  - Vincoli di integrità
  - Possibili derivazioni

86



## Dizionario dei dati (entità)

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

88

## Dizionario dei dati (relationship)

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	

89

## Regole di vincolo

- (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 nessun progetto

90

## Regole di derivazione

- (1) Il numero di impiegati di un dipartimento si ottiene contando gli impiegati che afferiscono a tale dipartimento
- (2) Il budget di un progetto si ottiene moltiplicando per 3 la somma degli stipendi degli impiegati che vi partecipano

91

- I vincoli espressi in precedenza e le regole di derivazione riguardano i valori che vengono immessi nelle entità e associazioni in esecuzione.

92

## Affrontare il progetto

93

Quale costrutto E-R va utilizzato per rappresentare un concetto presente nelle specifiche dei requisiti?

- Bisogna basarsi sulle definizioni dei costrutti del modello E-R

94

- se ha proprietà significative e descrive oggetti con esistenza autonoma
  - entità
- se è semplice e non ha proprietà
  - attributo
- se correla due o più concetti
  - associazione
- se è caso particolare di un altro
  - generalizzazione

95

## Qualità di uno schema concettuale

- correttezza
- completezza
- leggibilità
- minimalità

96

## Strategie di progetto

- top-down
- bottom-up
- inside-out

97

## Strategia top-down

- Si parte da uno schema iniziale che viene successivamente raffinato e integrato per mezzo di primitive che lo trasformano in una serie di schemi intermedi per arrivare allo schema E-R finale

98

## Primitive di raffinamento top-down

- Da entità a associazione tra entità
- Da entità a generalizzazione
- Da associazione a insiemi di associazioni
- Da associazione a entità con associazioni
- Introduzione di attributi su entità e associazioni

99

## Strategia bottom-up

- Si parte dalle specifiche iniziali e si suddividono fino a dare specifica ad una componente minima di cui si da' lo schema E-R
- gli schemi prodotti vengono fusi e integrati fino ad ottenere lo schema finale

100

## Primitive di trasformazione Bottom-up

- Generazione di entita'
- Generazione di associazione
- Generazione di generalizzazione

101

## In pratica

- si procede di solito con una strategia ibrida (mista):
  - si individuano i concetti principali e si realizza uno schema scheletro
  - sulla base di questo si può decomporre
  - poi si raffina, si espande, si integra

102

## Definizione dello schema scheletro

- Si individuano i concetti più importanti, ad esempio perché più citati o perché indicati esplicitamente come cruciali e li si organizza in un semplice schema concettuale

103

## Archivio fotografico

Si vuole rappresentare la base di dati di un archivio fotografico distribuito in varie sedi. Le fotografie sono catalogate in base ad un catalogo di soggetti possibili, ciascun soggetto ha una propria chiave. Le foto hanno una dimensione ed uno stato di conservazione; per le foto a colori, è noto il tipo di stampa (chiaro o opaco). Le foto sono reperibili in archivi, di cui è noto il responsabile, l'indirizzo, il numero telefonico e l'orario di apertura.

Le foto possono descrivere personaggi, luoghi o oggetti. I personaggi hanno un nome ed un sesso; alcuni sono deceduti. Per i personaggi politici, si indica il partito di appartenenza e l'eventuale carica governativa ricoperta. Per gli artisti, si indica la loro attività prevalente (pittura, scultura, ...). Quando le foto descrivono opere artistiche, è noto il nome dell'opera d'arte, l'artista che l'ha realizzata, il luogo dove l'opera risiede e l'anno di realizzazione. Quando le foto descrivono luoghi o oggetti, è noto nome e descrizione.

104

## Ristrutturazione delle frasi

- Si vuole rappresentare la base di dati di un archivio fotografico distribuito in varie sedi.
- Le foto sono reperibili in archivi, di cui è noto il responsabile, l'indirizzo, il numero telefonico e l'orario di apertura.

105

- Le fotografie sono catalogate in base ai soggetti possibili.
- Le foto hanno una dimensione ed uno stato di conservazione; per le foto a colori, è noto il tipo di stampa (chiaro o opaco).

106

- I soggetti possibili di una foto sono: personaggi, luoghi o oggetti, ciascun soggetto ha una propria chiave.
- I personaggi hanno un nome ed un sesso; alcuni sono deceduti. Per i personaggi politici, si indica il partito di appartenenza e l'eventuale carica governativa ricoperta. Per gli artisti, si indica la loro attività prevalente (pittura, scultura, ...). Quando le foto descrivono opere artistiche, è noto il nome dell'opera d'arte, l'artista che l'ha realizzata, il luogo dove l'opera risiede e l'anno di realizzazione. Quando le foto descrivono luoghi o oggetti, è noto nome e descrizione del luogo o oggetto.

107

## Definizione dello scheletro

- Da una prima lettura del testo individuiamo che le entità fondamentali sono
  - Le fotografie
  - Gli archivi
  - I soggetti
- Queste entità sono in relazione fra loro ed è facilmente individuabile il seguente scheletro di base



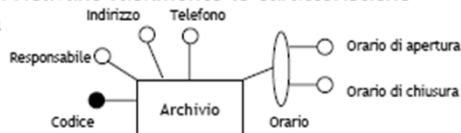
- Passiamo ora ad esaminare i singoli elementi dello scheletro...

108

# Analisi dello scheletro (1)

## □ Archivio

- ▶ Dal testo si ricavano facilmente le caratteristiche dell'entità



- ▶ Per identificatore è stato aggiunto un campo univoco "Codice"

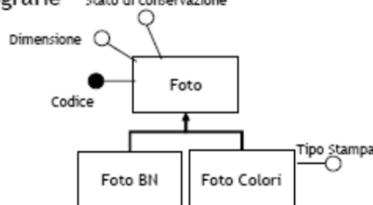
## □ Fotografie

- ▶ Dal testo si nota che esistono due tipologie di fotografie: quelle a colori e le altre (in bianco e nero). Si ha cioè una gerarchia totale ed esclusiva.

109

# Analisi dello scheletro (2)

## □ Fotografie



## □ Soggetto

- ▶ Dal testo si ricava una gerarchia abbastanza complessa
  - NOTA1: fare bene attenzione a dove si mettono gli attributi
    - Fare attenzione all'attributo nome: ricordare che i figli ereditano dai padri
  - NOTA2: mettere bene in evidenza il tipo di gerarchia (Totale, Esclusiva,...)

110

## Analisi dello scheletro (3)

### □ Soggetto

► Approfondimento: il testo non lo esplicitava, ma volendo era possibile aggiungere alcune associazioni fra le tipologie di foto

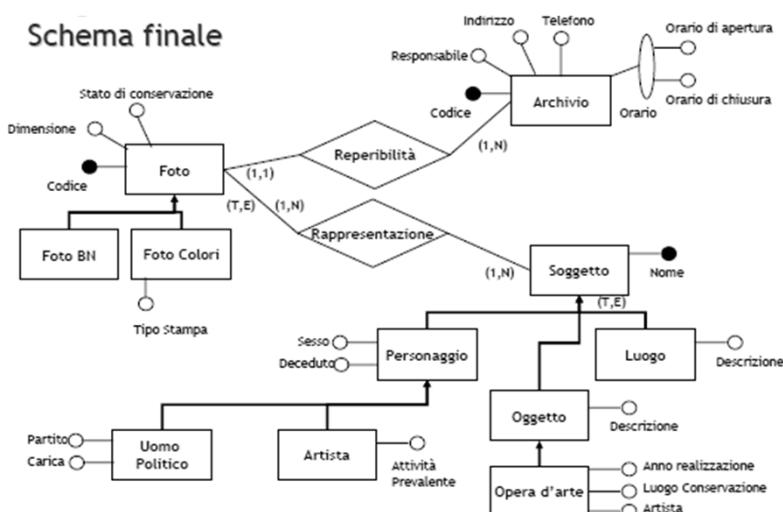
- Opera d'arte → Artista che l'ha realizzata
- Opera d'arte → Luogo di conservazione



111

## Schema finale

### Schema finale



112

## In altre parole

- Lo schema precedente non descrive un archivio in cui l'»artista» che ha realizzato un'opera d'arte sia un'occorrenza di artista. Non c'è nessun vincolo di integrità referenziale tra Artista attributo di opera d'arte e Nome, chiave di Artista.
- Se invece non metto l'attributo Artista ma inserisco un'associazione tra opera d'arte e artista...

113

Corso di Laurea in Ingegneria  
Informatica  
*Fondamenti di Informatica II*  
Modulo "*Basi di dati*"  
a.a. 2018-2019

Docente: Gigliola Vaglini  
Docente laboratorio: Francesco  
Pistolesi

Lezione 5

Dal modello concettuale al  
modello logico

## Obiettivo

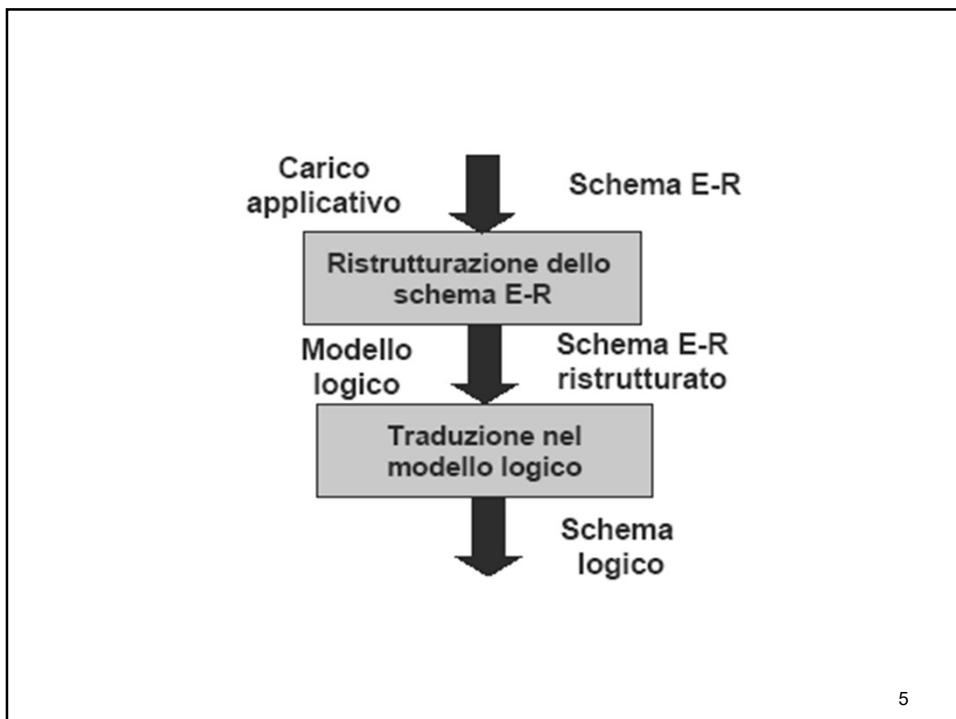
- "tradurre" in modo automatico lo schema concettuale in uno schema logico che rappresenti gli stessi dati in maniera corretta ed efficiente

3

## Dati di ingresso e uscita del tool di traduzione automatica

- Ingresso:
  - schema concettuale
  - modello logico scelto
  - informazioni sul carico applicativo (dimensione dei dati)
- Uscita:
  - schema logico

4



5

## Non è una traduzione immediata

### Motivazioni

- 1. semplificare la traduzione**
  - alcuni aspetti non sono direttamente rappresentabili
- 2. Tenere in considerazione i requisiti di prestazione**
  - "ottimizzare" le prestazioni

6

## Attività di ristrutturazione

1. Eliminazione delle generalizzazioni
2. Eliminazione degli attributi multivaleore
3. Analisi ed eventuale eliminazione delle ridondanze
4. Partizionamento/accorpamento di entità e relationship

7

## Le gerarchie nel modello relazionale

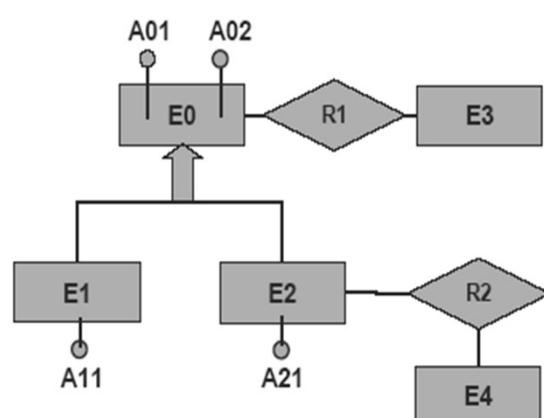
- Il modello relazionale non può rappresentare direttamente le generalizzazioni
- Le gerarchie vanno sostituite con entità e relazioni
  - Semplificare la traduzione

8

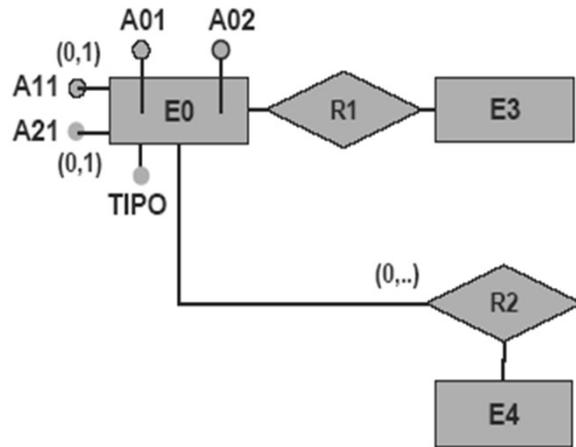
## Tre possibilità

- a. accorpamento delle figlie della generalizzazione nel genitore
- b. accorpamento del genitore della generalizzazione nelle figlie
- c. sostituzione della generalizzazione con relazioni

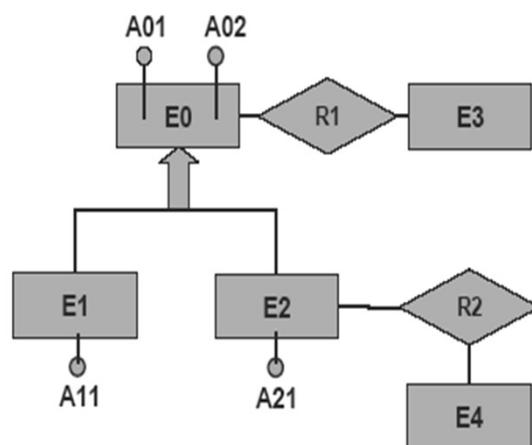
9



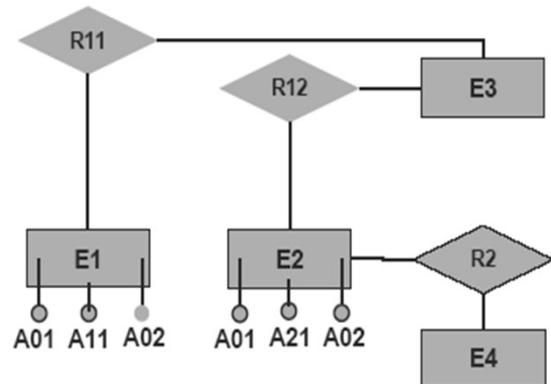
10



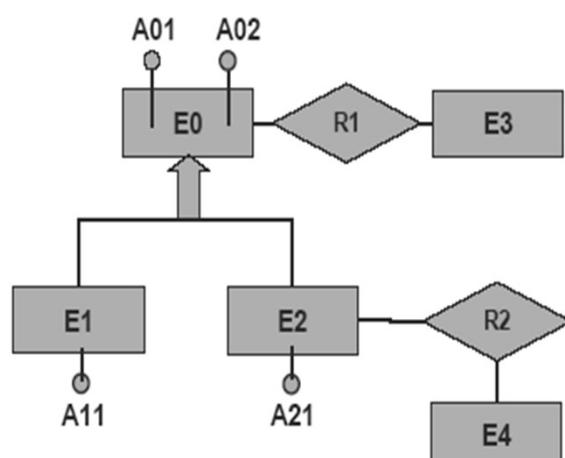
11



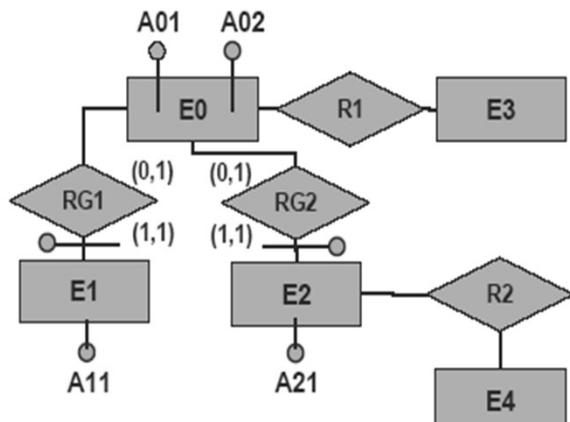
12



13



14



15

## Come scegliere

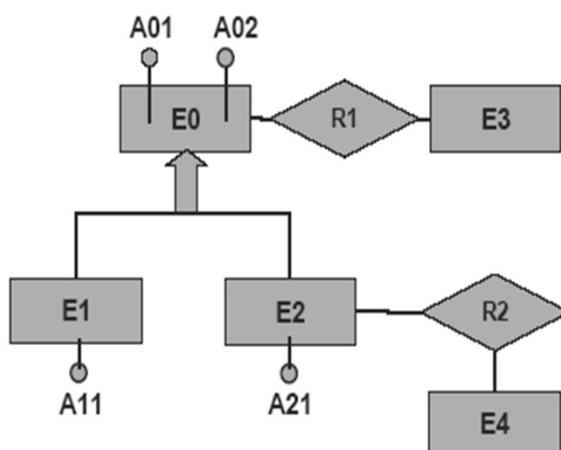
- la scelta fra le alternative si può fare basandosi sul numero e il tipo degli accessi fatti alle singole entità per eseguire le operazioni
- è possibile seguire alcune semplici regole generali

16

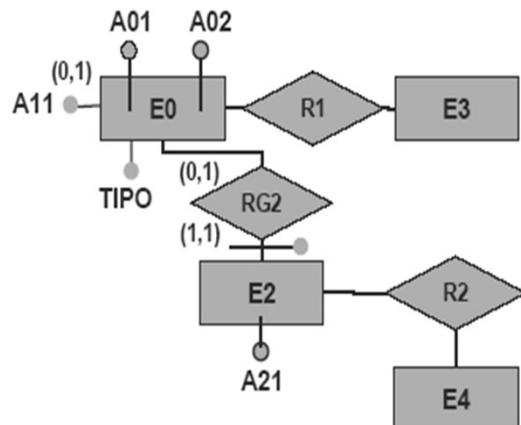
## Regole generali

- a. conviene se gli accessi al padre e alle figlie sono contestuali
- b. conviene se gli accessi sono solo alle figlie e sono distinti dall'una all'altra
- c. conviene se si effettuano accessi separati alle entità figlie e al padre
- sono anche possibili soluzioni "ibride", soprattutto in gerarchie a più livelli

17



18

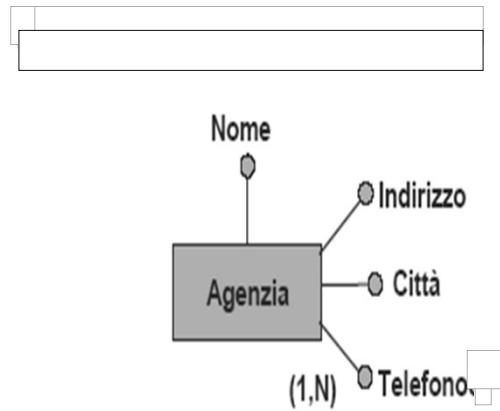


19

## Attività di ristrutturazione

1. Eliminazione delle generalizzazioni
2. Eliminazione degli attributi multivale
3. Analisi ed eventuale eliminazione delle ridondanze
4. Partizionamento/accorpamento di entità e relationship

20



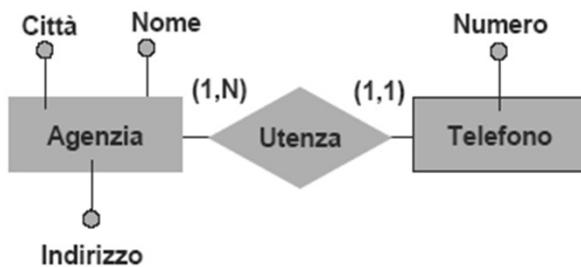
21

## Come rappresentarli

- Ripetere le tuple con ogni valore diverso dell'attributo
- Una sola tupla dimensionata al numero massimo di numeri di telefono possibili
- Spreco di memoria in entrambi i casi e possibili inconsistenze nel primo caso

22

eliminazione di attributi multivalore



23

## Ottimizzare le prestazioni

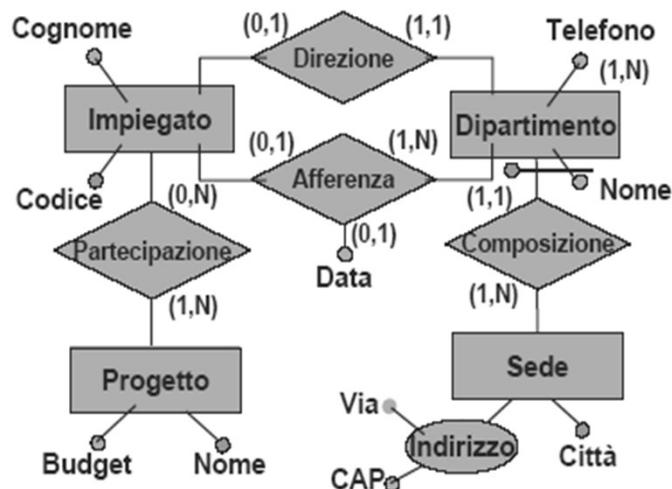
- Per ottimizzare abbiamo bisogno prima di analizzare le prestazioni
- Ma:
  - Come si possono valutare le prestazioni su uno schema concettuale?

24

## Indicatori per valutare le prestazioni

- Consideriamo degli "indicatori" dei parametri che caratterizzano le prestazioni
  - spazio: numero di occorrenze previste
  - tempo: numero di occorrenze (di entità e relationship) visitate per portare a termine un'operazione

25



26

## Tavola dei volumi

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

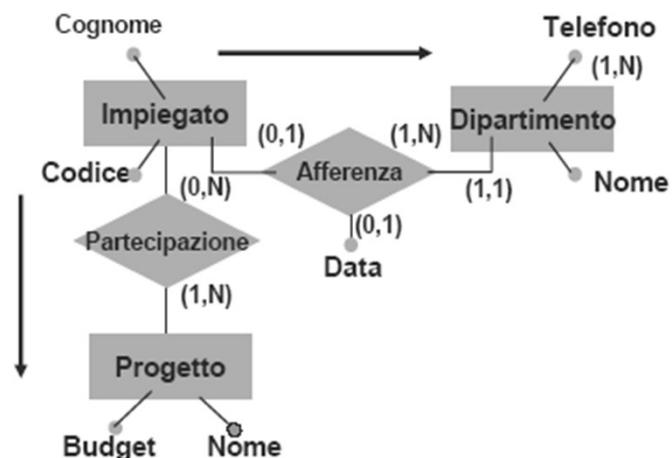
27

## Esempio di valutazione di costo

- Operazione:
  - 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

28

## Schema di navigazione



29

## Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
Impiegato	Entità	1	L
Afferenza	Relazione	1	L
Dipartimento	Entità	1	L
Partecipazione	Relazione	3	L
Progetto	Entità	3	L

30

## Attività di ristrutturazione

1. Eliminazione delle generalizzazioni
2. Eliminazione degli attributi multivaleore
3. Analisi ed eventuale eliminazione delle ridondanze
4. Partizionamento/accorpamento di entità e relationship

31

## Ridondanza

- Una ridondanza in uno schema E-R è una informazione significativa, ma derivabile da altre

32

## Motivazione

- in questa fase si decide se eliminare le ridondanze eventualmente presenti o mantenerle/inserirle in base ad una valutazione del costo delle operazioni
- **Vantaggi**
  - semplificazione delle interrogazioni
- **Svantaggi**
  - appesantimento degli aggiornamenti
  - maggiore occupazione di spazio

33

## Forme di ridondanza in uno schema E-R

- **attributi derivabili:**
  - da altri attributi della stessa entità (o associazione)
  - da attributi di altre entità (o associazioni)
- **associazioni derivabili dalla composizione di altre associazioni ( presenza di cicli)**

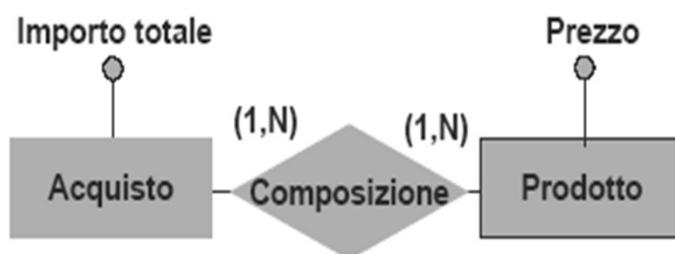
34

## Attributo derivabile

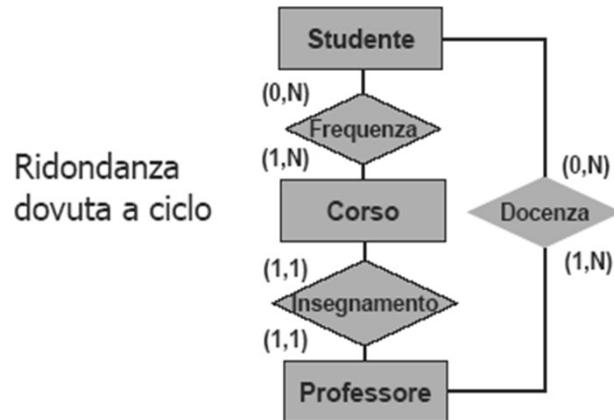


35

## Attributo derivabile da altra entità

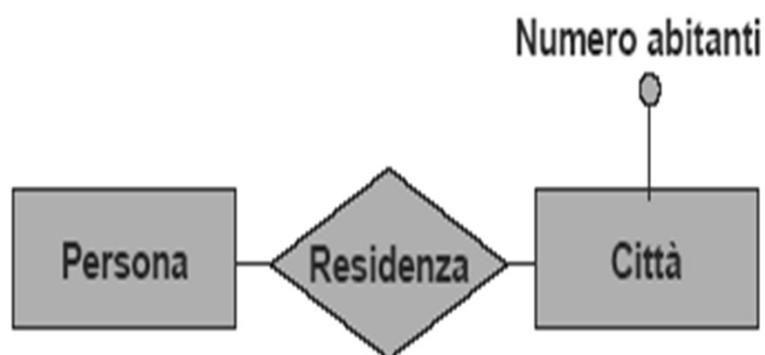


36



37

## Analisi di una ridondanza: schema E-R



38

## Tavola dei volumi e operazioni

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

- Operazione 1: memorizza una nuova persona con la relativa residenza, supponendo che la città sia già presente (500 volte al giorno)
- Operazione 2: stampa tutti i dati di una città (incluso il numero di abitanti) (2 volte al giorno)

39

## Accessi in presenza di ridondanza

### Operazione 1

Concetto	Costrutto	Accessi	Tipo
Persona	Entità	1	S
Residenza	Relazione	1	S
Città	Entità	1	L
Città	Entità	1	S

### Operazione 2

Concetto	Costrutto	Accessi	Tipo
Città	Entità	1	L

40

## Accessi in assenza di ridondanza

**Operazione 1**

Concetto	Costrutto	Accessi	Tipo
Persona	Entità	1	S
Residenza	Relazione	1	S

**Operazione 2**

Concetto	Costrutto	Accessi	Tipo
Città	Entità	1	L
Residenza	Relazione	5000	L

41

## Costi in presenza di ridondanza

- Operazione 1: 1500 accessi in scrittura e 500 accessi in lettura al giorno
- Operazione 2: trascurabile.
- Contiamo doppi gli accessi in scrittura
  - Totale di 3500 accessi al giorno

42

## Costi in assenza di ridondanza

- Operazione 1: 1000 accessi in scrittura al giorno
- Operazione 2: 10000 accessi in lettura al giorno
- Contiamo doppi gli accessi in scrittura
  - Totale di 12000 accessi al giorno

43

## Attività di ristrutturazione

1. Eliminazione delle generalizzazioni
2. Eliminazione degli attributi multivaleore
3. Analisi ed eventuale eliminazione delle ridondanze
4. Partizionamento/accorpamento di entità e relationship

44

## Motivazione

- Ristrutturazioni effettuate per rendere più efficienti le operazioni in base al principio che
  - gli accessi si riducono
    - separando attributi di un concetto che vengono acceduti separatamente
    - raggruppando attributi di concetti diversi acceduti insieme

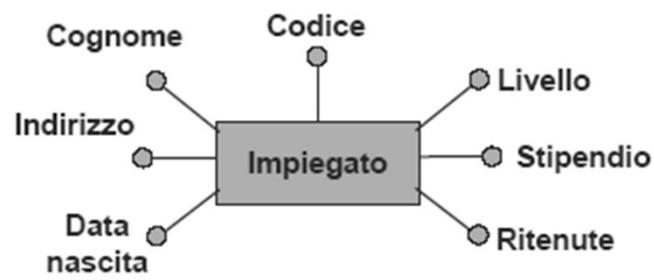
45

## Casi principali

- a. partizionamento di entità
- b. accorpamento di entità/relationship
- c. partizionamento di relationship

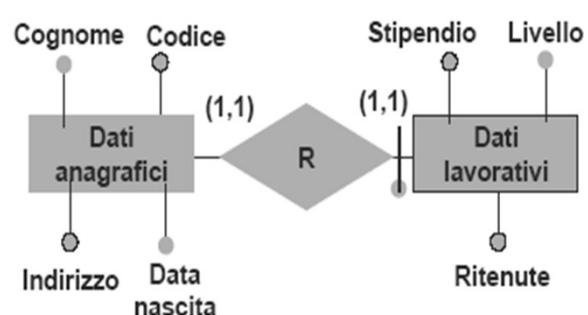
46

a. partizionamento di entità



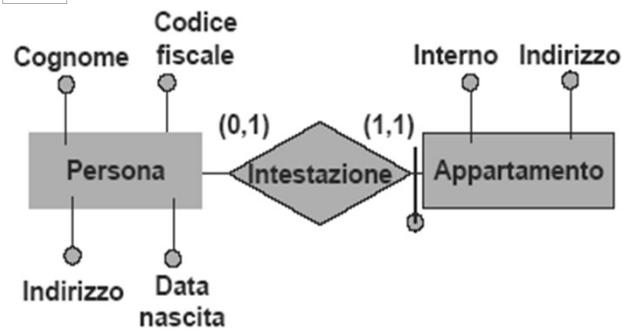
47

a. partizionamento di entità



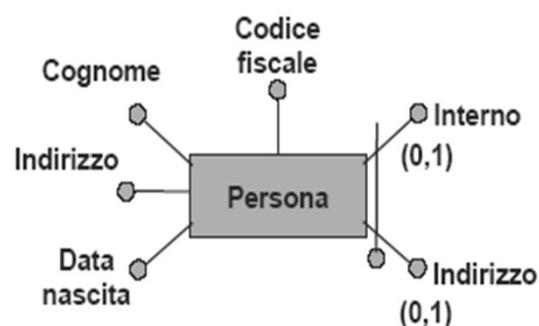
48

**b.** accorpamento di entità/ relationship



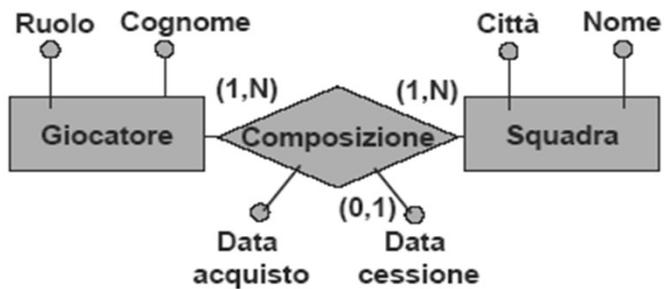
49

**b.** accorpamento di entità/ relationship



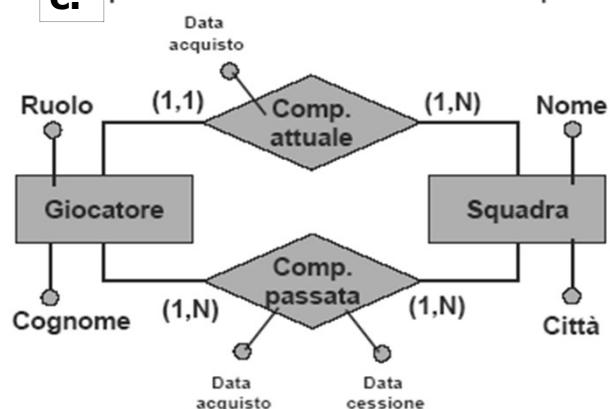
50

**C.** partizionamento di relationship



51

**C.** partizionamento di relationship



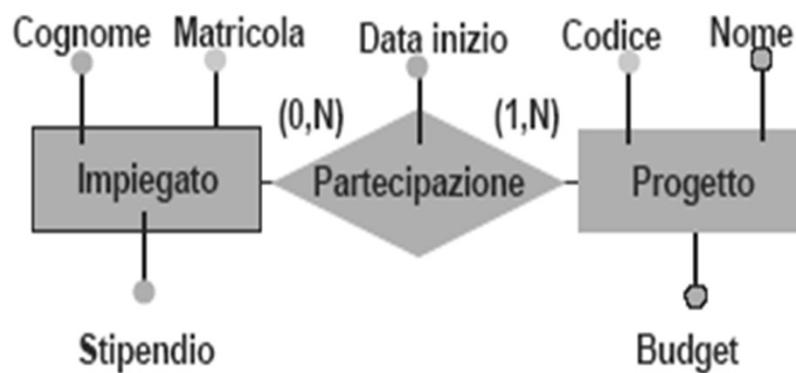
52

## Traduzione verso il modello relazionale

- idea di base:
  - le entità diventano relazioni sugli stessi attributi
  - le associazioni diventano relazioni sugli identificatori delle entità coinvolte (più gli attributi propri)

53

## Relationship multi a multi



54

## Relationship molti a molti

- Impiegato (Matricola, Cognome, Stipendio)
- Progetto (Codice, Nome, Budget)
- Partecipazione (Matricola, Codice, DataInizio)
- con vincoli di integrità referenziale fra
  - Matricola in Partecipazione e (la chiave di) Impiegato
  - Codice in Partecipazione e (la chiave di) Progetto

55

Nomi più espressivi per gli attributi  
della chiave della relazione che  
rappresenta la relationship

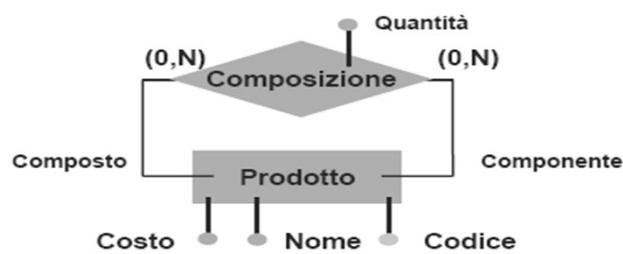
Impiegato(Matricola, Cognome, Stipendio)  
Progetto(Codice, Nome, Budget)  
Partecipazione (Matricola, Codice, DataInizio)

- diventa

Partecipazione (Impiegato, Progetto,  
DataInizio)

56

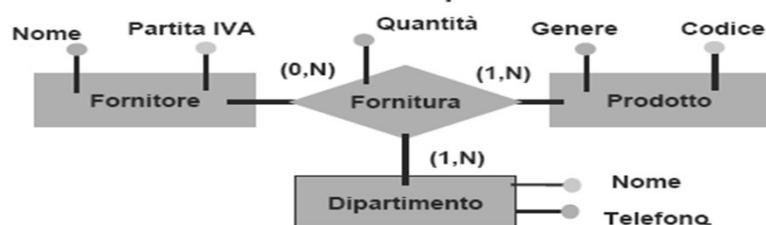
## Relationship ricorsive



Prodotto(Codice, Nome, Costo)  
Composizione (Composto, Componente, Quantità)

57

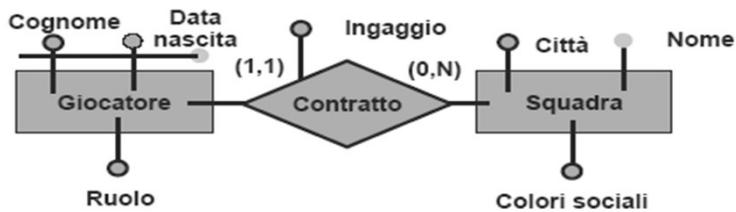
## Relationship n-arie



Fornitore(PartitaIVA, Nome)  
Prodotto(Codice, Genere)  
Dipartimento(Nome, Telefono)  
Fornitura (Fornitore, Prodotto, Dipartimento, Quantità)

58

## Relationship uno-a-molti



**Giocatore** (Cognome, DataNascita, Ruolo)  
**Contratto** (CognGiocatore, DataNascG, Squadra, Ingaggio)  
**Squadra** (Nome, Città, ColoriSociali)

- Alternative? Essendo la cardinalità di Contratto rispetto a Giocatore (1,1), la chiave delle tabelle Contratto e Giocatore.....

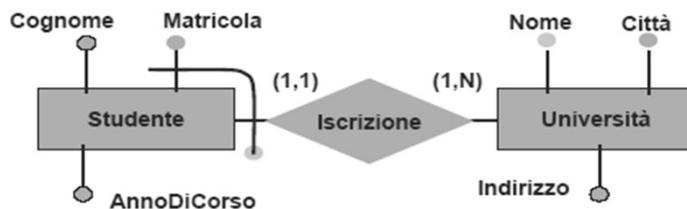
59

## Soluzione più compatta

- Giocatore( Cognome, DataNasc, Ruolo, Squadra, Ingaggio )
- Squadra ( Nome, Città, ColoriSociali )
- con vincolo di integrità referenziale fra Squadra in Giocatore e la chiave di Squadra
- se la cardinalità minima della relationship è 0 per Giocatore, allora Squadra in Giocatore deve ammettere valore nullo

60

## Entità con identificatore esterno



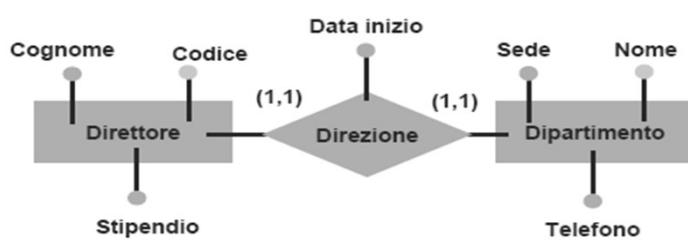
**Studente (Matricola, Università, Cognome, AnnoDiCorso)**

**Università (Nome, Città, Indirizzo)**

- con vincolo ...

61

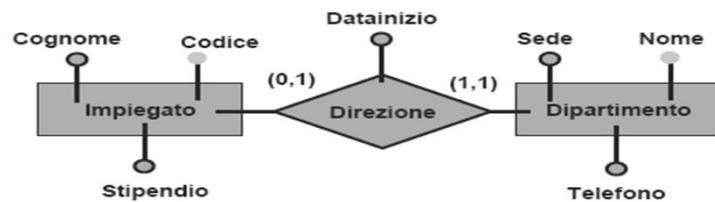
## Relationship uno-a-uno



- varie possibilità:
  - fondere da una parte o dall'altra
  - fondere tutto?

62

## Un caso fortunato



Impiegato (Codice, Cognome, Stipendio)

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

- con vincolo di integrità referenziale, e senza valori nulli

63

## Un altro caso



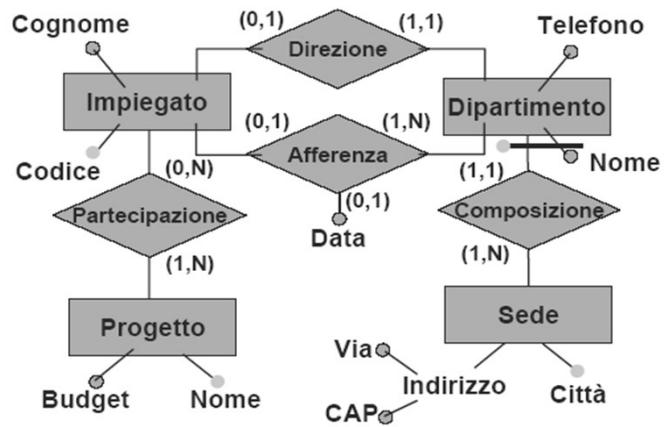
Impiegato (Codice, Cognome, Stipendio)

Dipartimento (Nome, Sede, Telefono)

Direzione (Direttore, Dipartimento, DataInizio)

- con vincoli di integrità referenziale, senza valori nulli

64



65

## Schema finale

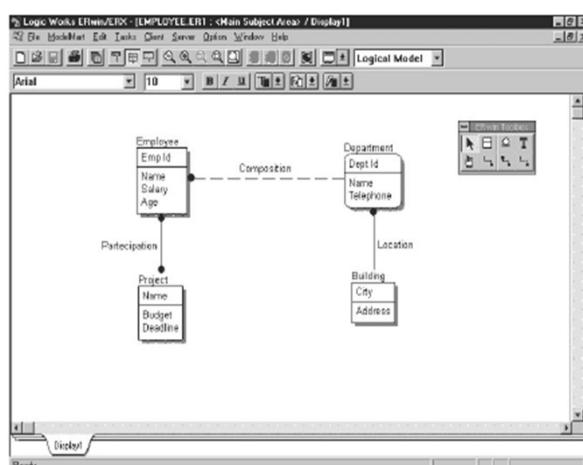
- **Impiegato(Codice, Cognome, Dipartimento, Sede, Data)**
- **Dipartimento(Nome, Città, Telefono, Direttore)**
- **Partecipazione(Impiegato, Progetto)**
- **Progetto(Nome, Budget)**
- **Sede(Città, Via, CAP)**

66

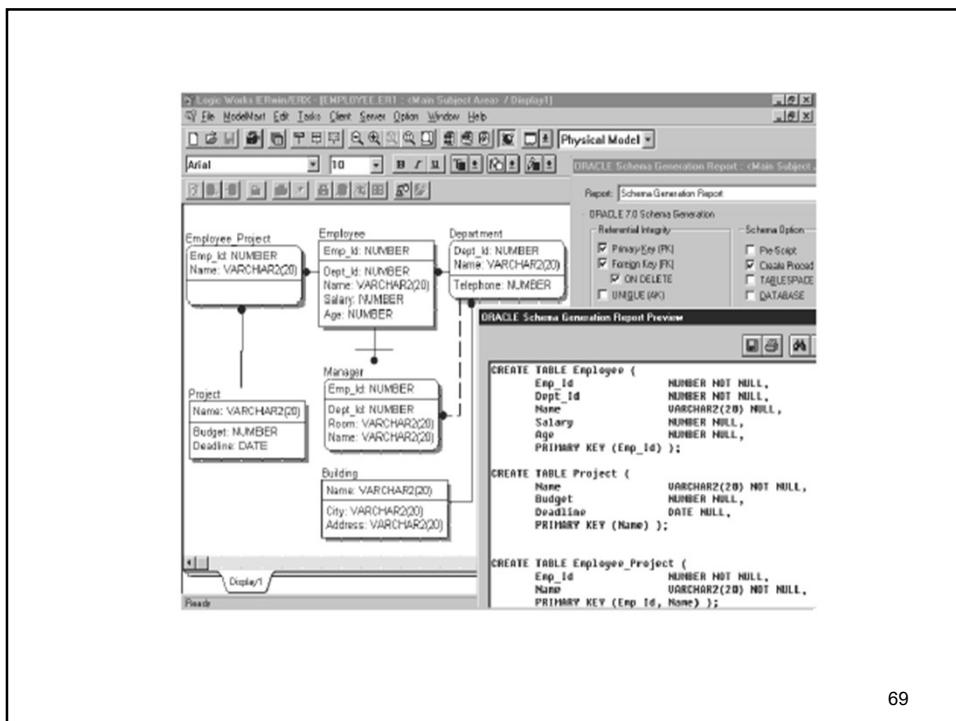
## Strumenti di supporto

- Esistono sul mercato prodotti CASE che forniscono un supporto a tutte le fasi della progettazione di basi di dati

67



68



69

Corso di Laurea in Ingegneria  
Informatica  
*Fondamenti di Informatica II*  
Modulo "Basi di dati"  
a.a. 2018-2019

Docente: Gigliola Vaglini  
Docente di laboratorio: Francesco  
Pistolesi

1

- Abbiamo messo delle ridondanze, ma cosa significa?
- Città con n.di abitanti ok
- Codicefattura - Lordo - iva - netto

2

# Lezione 6

Dipendenze funzionali  
Relazioni in forma normale

3

## Forme normali

- La forma normale è una proprietà che garantisce la "qualità" di una base di dati relazionale, cioè l'assenza di determinati difetti
  - Ad esempio, una relazione non normalizzata presenta ridondanze e anomalie

4

## Normalizzazione

- La normalizzazione è la procedura che permette di portare schemi relazionali in forma normale
- La normalizzazione è utilizzata come tecnica di verifica dei risultati della progettazione, non costituisce una metodologia di progetto

5

## Una relazione con anomalie

<u>Impiegato</u>	<u>Stipendio</u>	<u>Progetto</u>	<u>Bilancio</u>	<u>Funzione</u>
Rossi	20	Marte	2	tecnico
Verdi	35	Giove	15	progettista
Verdi	35	Venere	15	progettista
Neri	55	Venere	15	direttore
Neri	55	Giove	15	consulente
Neri	55	Marte	2	consulente
Mori	48	Marte	2	direttore
Mori	48	Venere	15	progettista
Bianchi	48	Venere	15	progettista
Bianchi	48	Giove	15	direttore

6

## Proprietà

- Ogni impiegato ha un solo stipendio (anche se partecipa a più progetti)
- Ogni progetto ha un bilancio
- Ogni impiegato in ciascun progetto ha una sola funzione (anche se può avere funzioni diverse in progetti diversi)

7

## Anomalie

- Lo stipendio di ciascun impiegato è ripetuto in tutte le ennuple relative
  - ridondanza
- Se lo stipendio di un impiegato varia, è necessario andarne a modificare il valore in diverse ennuple
  - anomalia di aggiornamento
- Se un impiegato si licenzia, dobbiamo cancellarlo in diverse ennuple
  - anomalia di cancellazione

8

## Perché questi fenomeni?

- Un'unica relazione per rappresentare informazioni eterogenee
  - gli impiegati con i relativi stipendi
  - i progetti con i relativi bilanci
  - le partecipazioni degli impiegati ai progetti con le relative funzioni

9

Per studiare in maniera sistematica questi aspetti, è necessario utilizzare il concetto di dipendenza funzionale

10

## 9.1. Le dipendenze funzionali

11

### Dipendenza funzionale

Si considerino

- la relazione  $r$  su  $R(X)$
- due sottoinsiemi non vuoti  $Y$  e  $Z$  di  $X$

esiste in  $r$  una dipendenza funzionale (FD) da  $Y$  a  $Z$  se,  
per ogni coppia di ennuple  $t_1$  e  $t_2$  di  $r$  con gli stessi  
valori su  $Y$ , risulta che  $t_1$  e  $t_2$  hanno gli stessi valori  
anche su  $Z$

- ad ogni chiave  $K$  di  $R$  corrisponde una  
dipendenza funzionale in  $R$  da  $K$  verso tutti gli  
attributi della relazione

12

## Notazione

$X \rightarrow Y$

- Esempi:

Impiegato  $\rightarrow$  Stipendio

Progetto  $\rightarrow$  Bilancio

Impiegato Progetto  $\rightarrow$  Funzione

13

## Altre FD

- Impiegato Progetto  $\rightarrow$  Progetto
- Si tratta di una FD "banale" (sempre soddisfatta)
  - $Y \rightarrow A$  è non banale se  $A$  non appartiene a  $Y$
  - $Y \rightarrow Z$  è non banale se nessun attributo in  $Z$  appartiene a  $Y$

14

<u>Impiegato</u>	<u>Stipendio</u>	<u>Progetto</u>	<u>Bilancio</u>	<u>Funzione</u>
Rossi	20	Marte	2	tecnico
Verdi	35	Giove	15	progettista
Verdi	35	Venere	15	progettista
Neri	55	Venere	15	direttore
Neri	55	Giove	15	consulente
Neri	55	Marte	2	consulente
Mori	48	Marte	2	direttore
Mori	48	Venere	15	progettista
Bianchi	48	Venere	15	progettista
Bianchi	48	Giove	15	direttore

**Impiegato → Stipendio**

**Progetto → Bilancio**

**Impiegato Progetto → Funzione**

15

Alcune FD causano anomalie

- gli impiegati hanno un unico stipendio

**Impiegato → Stipendio**

- i progetti hanno un unico bilancio

**Progetto → Bilancio**

- Ma non tutte

**Impiegato Progetto → Funzione**

Come mai?

16

## Legame tra FD e anomalie

Impiegato → Stipendio

Progetto → Bilancio

Impiegato Progetto → Funzione

- Impiegato non è una chiave
- Progetto non è una chiave
- Impiegato Progetto è chiave
- La relazione contiene alcune informazioni legate alla chiave e altre ad attributi che non formano una chiave.

17

## Ancora le FD

- *Implicazione*
- Sia  $F$  un insieme di dipendenze funzionali definite su  $R(Z)$  e sia  $X \rightarrow Y$  una delle dipendenze in  $F$ :
  - si dice che  $F$  implica  $X \rightarrow Y$  ( $F \Rightarrow X \rightarrow Y$ ) se, per ogni istanza  $r$  di  $R$  che verifica tutte le dipendenze in  $F$ , risulta verificata anche  $X \rightarrow Y$
  - si dice anche che  $X \rightarrow Y$  è implicata da  $F$

18

## FD (cont.)

- *Chiusura*
- Dato un insieme di dipendenze funzionali  $F$  definite su  $R(Z)$ , la chiusura di  $F$  è l'insieme di tutte le dipendenze funzionali implicate da  $F$
- $F^+ = \{ X \rightarrow Y \mid F \Rightarrow X \rightarrow Y \}$
- Dato un insieme di dipendenze funzionali  $F$  definite su  $R(Z)$ , un'istanza  $r$  di  $R$  che soddisfa  $F$ , soddisfa anche  $F^+$

19

## FD (cont.)

- Dato  $R(Z)$  ed un insieme  $F$  di FD, un insieme di attributi  $X$  appartenenti a  $Z$  si dice superchiave di  $R$ , se la dipendenza funzionale  $X \rightarrow Z$  è logicamente implicata da  $F$  ( $X \rightarrow Z$  è in  $F^+$ ).
- Se nessun sottoinsieme proprio di  $X$  è superchiave di  $R$ , allora  $X$  si dice chiave di  $R$

20

## Calcolo di $F^+$

- La definizione di implicazione non è direttamente utilizzabile nella pratica, essa prevede, infatti, una quantificazione universale sulle istanze della base di dati ("per ogni istanza r ...."),
- Armstrong (1974) ha fornito delle regole di inferenza che permettono di derivare effettivamente tutte le dipendenze funzionali che sono implicate da un dato insieme iniziale
- tali regole sono *corrette e complete*, cioè permettono di ottenere tutte e sole le dipendenze in  $F^+$

21

## Regole di inferenza di Armstrong

1. Riflessività: Se  $Y \subseteq X$ , allora  $X \rightarrow Y$
2. Additività (o espansione): Se  $X \rightarrow Y$ , allora  $XZ \rightarrow YZ$ , per qualunque  $Z$
3. Transitività: Se  $X \rightarrow Y$  e  $Y \rightarrow Z$ , allora  $X \rightarrow Z$

22

## Proprietà delle regole di Armstrong

- **Teorema (correttezza):** Le regole di inferenza di Armstrong sono corrette, cioè, applicandole ad un insieme F di dipendenze funzionali, si ottengono solo dipendenze logicamente implicate da F.
- **Teorema (completezza):** Le regole di inferenza di Armstrong sono complete, cioè, applicandole ad un insieme F di dipendenze funzionali, si ottengono tutte le dipendenze logicamente implicate da F.
- **Teorema (minimalità):** Le regole di inferenza di Armstrong sono minimali, cioè ignorando anche una sola di esse, l'insieme di regole che rimangono non è più completo.

23

## Esempi di prove

- **DIMOSTRARE** che,  $\forall$  istanza di ogni relazione,  
 $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$
- Supponiamo per assurdo che esista una istanza r di R in cui valga  $X \rightarrow Y$  ma non  $XZ \rightarrow YZ$ , devono perciò esistere due tuple t1 e t2 di r tali che :  
(1)  $t1[X] = t2[X]$ , (2)  $t1[Y] = t2[Y]$ ,  
(3)  $t1[XZ] \neq t2[XZ]$ , (4)  $t1[YZ] \neq t2[YZ]$   
ma ciò è assurdo, poichè da (1) e (3) si deduce:  
(5)  $t1[Z] = t2[Z]$ ,  
e da (2) e (5) si deduce :  
(6)  $t1[YZ] = t2[YZ]$ ,  
in contraddizione con la (4)

24

## (cont.)

- DIMOSTRARE che  $X \rightarrow Y$  e  $Y \rightarrow Z \Rightarrow X \rightarrow Z$
- Supponiamo per assurdo che esista una istanza  $r$  di  $R$  in cui valgano  $X \rightarrow Y$  e  $Y \rightarrow Z$ , ma non  $X \rightarrow Z$ , devono perciò esistere due tuple  $t_1$  e  $t_2$  in  $r$  tali che :
  - (1)  $t_1[X] = t_2[X]$ ,
  - (2)  $t_1[Y] = t_2[Y]$ ,
  - (3)  $t_1[Z] = t_2[Z]$ ,
  - (4)  $t_1[Z] \neq t_2[Z]$ma ciò è assurdo

25

## Regole derivate di Armstrong

### 4. Regola di unione

$$\{X \rightarrow Y, X \rightarrow Z\} \Rightarrow X \rightarrow YZ$$

### 5. Regola di pseudotransitività (o aggiunta sinistra)

$$\{X \rightarrow Y, WY \rightarrow Z\} \Rightarrow XW \rightarrow Z$$

### 6. Regola di decomposizione

$$\text{Se } Z \subseteq Y, X \rightarrow Y \Rightarrow X \rightarrow Z$$

26

## Esempi di prove

- DIMOSTRARE che  $X \rightarrow Y$  e  $X \rightarrow Z \Rightarrow X \rightarrow YZ$   
Per ipotesi valgono
  - a)  $X \rightarrow Y$
  - b)  $X \rightarrow Z$applicando la regola 2 ad (a) otteniamo
  - c)  $XZ \rightarrow YZ$applicando la stessa regola a (b) otteniamo
  - X  $X \rightarrow XZ$ ,
  - che equivale a
    - d)  $X \rightarrow XZ$per la regola 3 applicata a (d) e (c) otteniamo
  - $X \rightarrow YZ$

27

## Esempio di calcolo di $F^+$

- Prendiamo le FD dell'esempio
  - i. Impiegato  $\rightarrow$  Stipendio
  - ii. Progetto  $\rightarrow$  Bilancio
  - iii. Impiegato Progetto  $\rightarrow$  Funzione
- E usiamo la regola 2 sulle dipendenze i e ii, otteniamo
  - iv. Impiegato Progetto  $\rightarrow$  Stipendio Progetto
  - v. Progetto Impiegato  $\rightarrow$  Bilancio Impiegato
- Di conseguenza
  - Impiegato Progetto  $\rightarrow$  Stipendio Progetto Impiegato Bilancio Funzione
- e quindi Impiegato Progetto è chiave
- Ci sono altre FD in  $F^+$ ?

28

## Equivalenza

- Dato un insieme di dipendenze funzionali  $F$  è molto utile poter determinare un insieme di dipendenze funzionali  $G$  che sia equivalente ad  $F$ , ma sia anche strutturalmente più semplice
- $F$  e  $G$  sono equivalenti se  $F^+ = G^+$ , ovvero, per ogni  $X \rightarrow Y \in F$ , deve essere  $X \rightarrow Y \in G^+$  e, viceversa, per ogni  $X \rightarrow Y \in G$ , deve essere  $X \rightarrow Y \in F^+$

29

## Esempio 1

- $F = \{ A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H \}$
  - $G = \{ A \rightarrow CD, E \rightarrow AH \}$
- Verificare se  $F$  e  $G$  sono equivalenti
- Dimostro che le DF in  $F$  sono derivabili dalle DF in  $G$ , e viceversa
  - $A \rightarrow CD \Rightarrow A \rightarrow C, A \rightarrow D$
  - $A \rightarrow CD, CCD \rightarrow CD \Rightarrow AC \rightarrow CD \Rightarrow AC \rightarrow C, AC \rightarrow D$
  - $E \rightarrow AH \Rightarrow E \rightarrow A, E \rightarrow H$
  - $E \rightarrow A, A \rightarrow D \Rightarrow E \rightarrow D$
  - $E \rightarrow A, E \rightarrow D \Rightarrow E \rightarrow AD$

30

(cont.)

- $A \rightarrow C, AC \rightarrow D \Rightarrow AA \rightarrow D \Rightarrow A \rightarrow D$   
 $A \rightarrow C, A \rightarrow D \Rightarrow A \rightarrow CD$
- $E \rightarrow AD \Rightarrow E \rightarrow A, E \rightarrow D$   
 $E \rightarrow A, E \rightarrow H \Rightarrow E \rightarrow AH$

31

Ovvero

- Il calcolo di  $F^+$  è molto costoso (esponenziale nel numero di attributi dello schema nel caso peggiore),
- spesso quello che ci interessa è verificare se  $F^+$  contiene una certa dipendenza
- alternativamente si può calcolare e utilizzare la chiusura transitiva di un insieme di attributi  $X$  (meno costoso ?), infatti
  - si può dimostrare che  $X \rightarrow Y$  è in  $F^+$  sse  $Y \subseteq X^+$

32

## Algoritmo per il calcolo di $X^+$

- Denotiamo con  $X^+$  l'insieme degli attributi di  $R(Z)$  che dipendono da  $X$  (chiusura di  $X$ ) secondo  $F$ ; calcolare  $X^+$  è semplice (complessità)?
  - CalcolaChiusura( $X, F$ )=

```
{ X+ = X;
    Ripeti:
        - Fine = true;
        - Per tutte le FD in F = {Vi → Wi};
        - Se Vi ⊆ X+ e Wi ⊈ X+ allora: {X+ = X+ ∪ Wi; Fine = false}
    Fino a che Fine = true
}
```

33

## Esempio 2

- Supponiamo di avere  
 $F = \{A \rightarrow B, BC \rightarrow D, B \rightarrow E, E \rightarrow C\}$   
e calcoliamo  $A^+$ , ovvero l'insieme di attributi che dipendono da  $A$ 
  - $A^+ = A$
  - $A^+ = AB$  poiché  $A \rightarrow B$  e  $A \subseteq A^+$
  - $A^+ = ABE$  poiché  $B \rightarrow E$  e  $B \subseteq A^+$
  - $A^+ = ABEC$  poiché  $E \rightarrow C$  e  $E \subseteq A^+$
  - $A^+ = ABED$  poiché  $BC \rightarrow D$  e  $BC \subseteq A^+$
- Quindi da  $A$  dipendono tutti gli attributi dello schema, ovvero  $A$  è superchiave (e anche chiave)!

34

## Esempio 1 (cont)

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

Verificare se  $F$  e  $G$  sono equivalenti

- Invece di verificare se  $X \rightarrow Y$  in  $F$  è anche in  $G^+$ , verifico se  $Y \subseteq (X)^{+G}$  (chiusura di  $X$  rispetto a  $G'$ ), e viceversa per ogni df in  $G$
- per  $A \rightarrow C$  risulta  $(A)^{+G} = ACD$ ; o.k.  $C \subseteq (A)^{+G}$
- per  $AC \rightarrow D$  risulta  $(AC)^{+G} = ACD$ ; o.k.  $D \subseteq (AC)^{+G}$
- per  $E \rightarrow AD$  risulta  $(E)^{+G} = EADCH$ ; o.k.  $AD \subseteq (E)^{+G}$
- per  $E \rightarrow H$  risulta  $(E)^{+G} = EHADC$ ; o.k.  $H \subseteq (E)^{+G}$

E viceversa

35

quindi

- $F$  e  $G$  sono equivalenti se
  - per ogni  $X \rightarrow Y \in F, Y \in X^+$  secondo  $G$ , e,
  - per ogni  $Z \rightarrow W \in G, W \in Z^+$  secondo  $F$

36

## Importanza della chiusura di un insieme di attributi

- Dato  $R(Z)$  con le sue dipendenze  $F$ :
- La chiusura di un insieme  $X \subseteq Z$  di attributi è fondamentale per diversi scopi:
  - Si può utilizzare per verificare se una dipendenza funzionale è logicamente implicata da  $F$ 
    - $X \rightarrow Y$  è in  $F^+$  se e solo se  $Y \subseteq X^+$
  - Si può utilizzare per verificare se un insieme di attributi è superchiave o chiave
    - $X$  è superchiave di  $R$  se e solo se  $X \rightarrow Z$  è in  $F^+$ , cioè se e solo se  $Z \subseteq X^+$
    - $X$  è chiave di  $R$  se e solo se  $X \rightarrow Z$  è in  $F^+$  e non esiste alcun sottoinsieme  $Y$  di  $X$  ottenuto da  $X$  eliminando un solo elemento, tale che  $Z \subseteq Y^+$

37

## Copertura Minimale

- Alcuni attributi di una dipendenza funzionale possono essere ridondanti:
  - ridondanza a DESTRA:  $\{ A \rightarrow B, B \rightarrow C, A \rightarrow CD \}$  può essere semplificata in  $\{ A \rightarrow B, B \rightarrow C, A \rightarrow C, A \rightarrow D \}$  (FD semplici)
  - ridondanza a SINISTRA:  $\{ A \rightarrow B, B \rightarrow C, AC \rightarrow D \}$  può essere semplificata in  $\{ A \rightarrow B, B \rightarrow C, A \rightarrow D \}$  (senza attributi estranei)
- Un insieme  $F$  di dipendenze funzionali può contenere dipendenze ridondanti, ovvero ottenibili tramite le dipendenze di  $F$ 
  - Esempio:  $A \rightarrow C$  è ridondante in  $\{ A \rightarrow B, B \rightarrow C, A \rightarrow C \}$
- Intuitivamente, una copertura minimale (canonica) di  $F$  è un insieme minimale di dipendenze funzionali equivalente a  $F$  e privo di dipendenze e attributi ridondanti

38

## FD semplici

- Per minimizzare un insieme di FD è innanzitutto necessario scriverle tutte in una forma "standard" (forma canonica), in cui sulla destra c'è sempre un singolo attributo
- Supponiamo di avere
$$F = \{AB \rightarrow CD, AC \rightarrow DE\}$$
- Possiamo riscrivere F come
$$F = \{AB \rightarrow C, AB \rightarrow D, AC \rightarrow D, AC \rightarrow E\}$$

39

## Attributi "estranei"

- In alcune FD è possibile che sul lato sinistro ci siano degli attributi inutili ("estranei"): come si identificano?
- Supponiamo di avere  $F = \{AB \rightarrow C, A \rightarrow B\}$  e calcoliamo  $A^+$  e  $B^+$ 
$$A^+ = A \quad B^+ = B$$
$$A^+ = AB \text{ poiché } A \rightarrow B \text{ e } A \subseteq A^+$$
$$A^+ = ABC \text{ poiché } AB \rightarrow C \text{ e } AB \subseteq A^+$$

$C$  dipende solo da  $A$ , e in  $AB \rightarrow C$  l'attributo  $B$  è estraneo (a sua volta dipende da  $A$ ) e possiamo riscrivere l'insieme di FD più semplicemente come:  $F' = \{A \rightarrow C, A \rightarrow B\}$
- Quindi in una FD del tipo  $AX \rightarrow B$  l'attributo  $A$  è estraneo se  $X^+$  include  $B$  (ovvero  $X$  da solo determina  $B$ )

40

## FD ridondanti

- Dopo avere eliminato gli attributi estranei si deve verificare se vi sono intere FD inutili ("ridondanti"), ovvero FD che sono implicate da altre
- *Come facciamo a stabilire che una FD del tipo  $X \rightarrow A$  è ridondante?*
  - La eliminiamo da F, calcoliamo  $X^+$  e verifichiamo se include A, ovvero se con le FD che restano riusciamo ancora a dimostrare che X determina A

41

## Copertura Minimale - Definizione

- Un insieme F di FD è minimale se
  1. la parte destra di ogni FD in F è formata da un solo attributo
  2. tutti gli attributi della parte sinistra di ogni FD in F sono necessari ( se dato  $X \rightarrow A$  si toglie un attributo in X l'insieme delle FD risultante non è più equivalente ad F )
  3. tutte le DF in F sono necessarie, nessuna è ridondante ( non è possibile rimuovere una FD da F e avere un insieme equivalente a F )
- Una copertura minimale di un insieme F è un insieme minimale equivalente a F ( può essere usato al posto di F )

N.B. In generale, la copertura minimale non è unica

42

## Copertura Minimale - Algoritmo

- Calcolo di M minimale per un insieme di F :
  - $M = F$
  - ogni  $X \rightarrow \{A_1, A_2, \dots, A_n\}$  è sostituita da  $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_n$
  - ogni  $X \rightarrow A$  è sostituita da  $(X - \{B\}) \rightarrow A$   
se  $A \subseteq (X - \{B\})^+$
  - ogni rimanente  $X \rightarrow A$  in M è rimossa se  $A \subseteq X +$  anche in  $\{\{F - \{X \rightarrow A\}\}\}$

43

## Esempio

- Sia  $F = \{AB \rightarrow C, B \rightarrow A, C \rightarrow A\}$ , A è estraneo in  $AB \rightarrow C$ , quindi trasformiamo F in  $F' = \{B \rightarrow C, B \rightarrow A, C \rightarrow A\}$ , dopo possiamo eliminare  $B \rightarrow A$  trasformando  $F'$  in  $F'' = \{B \rightarrow C, C \rightarrow A\}$
- Se tentiamo di eliminare la FD ridondante prima di eliminare l'attributo estraneo non ci riusciamo
  - NB: questa operazione è bene che segua l'eliminazione degli attributi estranei

44

## 9.2 Relazioni in forma normale

45

### Forma normale di Boyce-Codd (BCNF)

- Una relazione  $r$  è in forma normale di Boyce-Codd se, per ogni dipendenza funzionale (non banale)  $X \rightarrow Y$  definita su di essa,  $X$  è superchiave di  $r$
- La forma normale richiede che i concetti in una relazione siano omogenei (tutte le proprietà sono associate alla chiave)

46

## BCNF

Se un insieme  $F$  di dipendenze per  $R$  non è in BCNF, allora in  $F$  c'è almeno una dipendenza  $X \rightarrow Y$  non banale con  $X$  non superchiave di  $R$ .

- **Teorema** Data  $R$ , se in  $F$  non c'è alcuna  $X \rightarrow Y$  non banale con  $X$  non superchiave di  $R$ , allora non ce n'è nemmeno in  $F^+$ .

47

## BCNF

- Grazie al teorema, è sufficiente analizzare una ad una le dipendenze non banali in  $F$  per verificare se ognuna ha una superchiave come membro sinistro
- occorre saper verificare se un insieme di attributi è superchiave di una relazione
  - $K$  è superchiave di  $R(Z)$  con dipendenze  $F$  se  $Z \subseteq K^+$

48

# Che facciamo se una relazione non è BCNF?

- La rimpiazziamo con altre relazioni che siano BCNF

Come?

- Decomponendo sulla base delle dipendenze funzionali, al fine di separare i concetti

49

50

È sempre così facile?

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

Impiegato → Sede  
Progetto → Sede

51

Decomponiamo sulla base  
delle dipendenze

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

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Progetto	Sede
Marte	Roma
Giove	Milano
Saturno	Milano
Venere	Milano

52

## Proviamo a ricostruire

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Progetto	Sede
Marte	Roma
Giove	Milano
Saturno	Milano
Venere	Milano

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

Diversa dalla relazione di partenza!

53

## Decomposizione senza perdita

- Una istanza  $r$  di una relazione  $R$  si decomponga senza perdita su  $X_1$  e  $X_2$  se il join naturale delle proiezioni di  $r$  su  $X_1$  e  $X_2$  è uguale a  $r$  stessa (cioè non contiene ennuple spurie)

54

## Algoritmo per la decomposizione in BCNF

- Assumiamo (senza perdita di generalità) che ogni volta che chiamiamo l'algoritmo descritto sotto, ogni dipendenza funzionale in  $F$  abbia un unico attributo come membro destro, e che  $U$  sia l'insieme di tutti gli attributi di  $R$
- $\text{Decomponi}(R, F) :=$   
{ if esiste  $X \rightarrow A$  in  $F$  con  $X$  non superchiave di  $R$   
then { sostituisci  $R$  con una relazione  $R_1$  con attributi  $U-A$ , ed una relazione  $R_2$  con attributi  $X \cup A$ ;  
 $\text{Decomponi}(R_1, F_{U-A})$ ;  
 $\text{Decomponi}(R_2, F_{X \cup A})$   
}

55

## Relazione R

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

$$F = \{\text{Impiegato} \rightarrow \text{Sede}, \text{Progetto} \rightarrow \text{Sede}\}$$

$\text{Decomponi}(R, F) := \{\text{R1(Impiegato, Progetto)}, \text{R2(Impiegato, Sede)}\}$

56

## Correttezza dell'algoritmo della decomposizione

- **Teorema** Qualunque sia l'input, l'esecuzione dell'algoritmo su tale input termina, e produce una decomposizione della relazione originaria tale che:
  - ogni relazione ottenuta è in BCNF
  - la decomposizione è senza perdita nel join

57

## Proiezione delle FD di $R(U)$ su $X \subseteq U$

- La proiezione di  $F$  su  $X$ , denotata da  $F_X$ , è l'insieme di dipendenze funzionali  $Z \rightarrow Y$  in  $F^+$  che coinvolgono solo attributi in  $X$ , cioè tali che  $Z \subseteq X$  e  $Y \subseteq X$

58

## Algoritmo

- Per calcolare  $F_X$ , cioè la proiezione di  $F$  su  $X$ , possiamo procedere per enumerazione (non si può fare meglio), evitando però di generare dipendenze funzionali "inutili"
- $\text{CalcolaProiezione}(F, X) :=$   
{ result =  $\emptyset$ ;  
per ogni sottoinsieme proprio  $S$  di  $X$ , per ogni attributo  $A$  in  $X$  tale che  $A$  non è in  $S$ , e tale che non esiste alcun sottoinsieme  $S'$  di  $S$  tale che  $S' \rightarrow A$  è in result,  
if  $A$  è in  $\text{CalcolaChiusura}(S, F)$  allora  $result = result \cup \{ S \rightarrow A \}$ ;  
}

59

## Dimensione della proiezione di $F$ su $X$

- Ci sono casi in cui la proiezione di  $F$  su  $X$  ha dimensione esponenziale rispetto alla dimensione di  $F$  e  $X$ , come mostrato dal seguente esempio
- Consideriamo  $R(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_n, C_1, C_2, \dots, C_n, D)$  e  $F = \{ A_i \rightarrow C_i, B_i \rightarrow C_i \mid 1 \leq i \leq n \} \cup \{ C_1 C_2 \dots C_n \rightarrow D \}$
- La proiezione di  $F$  su  $\{ A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_n, D \}$  è  $P = \{ X_1 X_2 \dots X_n \rightarrow D \mid X_i = A_i \text{ oppure } X_i = B_i \text{ per } 1 \leq i \leq n \}$ , la cui dimensione è ovviamente esponenziale rispetto alla dimensione dello schema  $R$  e delle dipendenze funzionali  $F$ .
- Si noti che si può dimostrare che nessun insieme equivalente a  $P$  ha cardinalità minore.

60

## Proprietà dell'algoritmo di decomposizione

- N.B. A seconda dell'ordine con cui si considerano le dipendenze funzionali, il risultato della decomposizione può cambiare

61

## Relazione R

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

$F = \{ \text{Impiegato} \rightarrow \text{Sede}, \text{Progetto} \rightarrow \text{Sede} \}$

$\text{Decomponi}(R, F) := \{ R1(\text{Impiegato}, \text{Progetto}), R2(\text{Progetto}, \text{Sede}) \}$

$\text{Decomponi}(R, F) := \{ R1(\text{Impiegato}, \text{Progetto}), R2(\text{Impiegato}, \text{Sede}) \}$

62

Consideriamo una di queste decomposizioni

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

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere

**Impiegato → Sede**

**Progetto → Sede**

63

## Osservazione

- La decomposizione è senza perdita sul join, però
  - La FD  $\text{Progetto} \rightarrow \text{Sede}$  interessa attributi che non stanno nella stessa tabella.
  - E' un problema?

64

## Problema

- Supponiamo di voler inserire una nuova ennupla che specifica la partecipazione dell'impiegato Neri, che opera a Milano, al progetto Marte

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere

Impiegato → Sede  
 Progetto → Sede

65

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere
Neri	Marte

66

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

67

## Conservazione delle dipendenze

- Una decomposizione conserva le dipendenze se ciascuna delle dipendenze funzionali dello schema originario coinvolge attributi che compaiono tutti insieme in uno degli schemi decomposti
- Progetto → Sede non è conservata

68

## Decomposizione senza perdita di dipendenze

- Sia  $R$  uno schema di relazione con dipendenze funzionali  $F$ , e sia  $X$  un sottoinsieme di attributi di  $R$
- La decomposizione di  $R$  in due relazioni con attributi  $X$  e  $Y$  è una decomposizione senza perdita di dipendenze se  $(F_X \cup F_Y)$  è equivalente a  $F$ , cioè se  $(F_X \cup F_Y)^+ = F^+$
- N.B. Non è assicurato che la decomposizione ottenuta dall'algoritmo per la decomposizione BCNF sia senza perdita di dipendenze

69

## La verifica di decomposizione senza perdita di dipendenze

- La definizione di decomposizione senza perdita di dipendenze è basata sul verificare che  $(F_X \cup F_Y)^+ = F^+$ .
- Per applicare la definizione,
  - è necessario sapere calcolare se un insieme di dipendenze funzionali è equivalente ad un altro
  - è necessario saper calcolare la proiezione di un insieme di dipendenze funzionali su un insieme di attributi

70

- Per la verifica di equivalenza si può usare un metodo polinomiale e, per ogni  $X \rightarrow Y \in F$ , calcolare  $X^+$  rispetto a  $G$  e verificare se  $Y \in X^+$ , idem per  $X \rightarrow Y \in G$  e  $X^+$  rispetto a  $F$ .
- Per calcolare la proiezione abbiamo invece un metodo esponenziale

71

### Caso interessante

- La relazione  $R(A,B,C)$ , con  $F = \{ A \rightarrow B, B \rightarrow C, A \rightarrow C \}$  non è in BCNF (qual è la chiave?).
- Se la decomponiamo in  $R1(A,B)$  e  $R2(B,C)$ , partendo da  $B \rightarrow C$ , otteniamo due relazioni in BCNF, con la proprietà che la decomposizione è senza perdita nel join. La decomposizione è anche senza perdita di dipendenze, perché la dipendenza funzionale  $A \rightarrow C$  è logicamente implicata dalle due dipendenze funzionali che valgono in  $R1$  e  $R2$ .
- N.B. Se la definizione di conservazione delle dipendenze non considerasse  $(F_X \cup F_Y)^+$  ma solo  $(F_X \cup F_Y)$ , allora la decomposizione sembrerebbe perdere la dipendenza  $A \rightarrow C$ , che non è esprimibile direttamente né in  $R1$  (cioè mediante  $F_{AB}$ ) né in  $R2$  (cioè mediante  $F_{BC}$ ).

72

## Qualità delle decomposizioni

- Una decomposizione dovrebbe sempre garantire:
  - BCNF
  - l'assenza di perdite, in modo da poter ricostruire le informazioni originarie
  - la conservazione delle dipendenze, in modo da mantenere i vincoli di integrità originari

*DB ben progettato*

73

## Relazione BCNF?

Proprietà: Ogni dirigente ha una sede; un progetto può essere diretto da più persone, ma in sedi diverse

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

74

## Verifica

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

**Progetto Sede → Dirigente**      ok  
**Dirigente → Sede**                  no

75

## Come decomporre?

- Progetto Sede → Dirigente coinvolge tutti gli attributi e quindi nessuna decomposizione può preservare tale dipendenza
- Si può trovare una BCNF, ma non potrà conservare le dipendenze

76

## Approccio differente: una nuova forma normale

- Una relazione  $r$  è in terza forma normale se, per ogni FD (non banale)  $X \rightarrow Y$  definita su  $r$ , è verificata almeno una delle seguenti condizioni:
  - $X$  è superchiave di  $r$
  - ogni attributo in  $Y$  è contenuto in almeno una chiave di  $r$

77

## Non BCNF, ma 3NF

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

Progetto Sede  $\rightarrow$  Dirigente

Dirigente  $\rightarrow$  Sede

L'attributo Sede è contenuto nella chiave

78

## Anomalie?

- C'è una ridondanza nella ripetizione della sede del dirigente per i vari progetti che dirige

79

## Confronto

- 3NF è meno restrittiva di BCNF (e ammette relazioni con alcune anomalie e ridondanze)
- il problema di verificare se una relazione è in 3NF è NP-completo (il miglior algoritmo deterministico conosciuto ha complessità esponenziale nel caso peggiore), infatti:
  - Dati R ed F e un attributo A
    - si genera non deterministicamente un sottoinsieme S degli attributi di R che contiene A,
    - si controlla se S è una chiave (non una superchiave)
- ha il vantaggio però di essere sempre "raggiungibile", cioè si può sempre ottenere una decomposizione 3NF senza perdite e che conserva le dipendenze

80

## Metodologia di decomposizione (1)

1. Data  $R$  ed  $F$  minimale, si usa

Decomponi( $R, F$ ) ottenendo gli schemi  $R_1(X_1), R_2(X_2), \dots, R_n(X_n)$  in BCFN ciascuno con dipendenze  $F_{X_i}$

2. Sia  $N$  l'insieme di dipendenze non preservate in  $R_1, R_2, \dots, R_n$ , cioè non incluse nella chiusura dell'unione dei vari  $F_{X_i}$

– Per ogni dipendenza  $X \rightarrow A$  in  $N$ , aggiungiamo lo schema relazionale  $X A$  con le dipendenze funzionali relative a  $XA$

81

## Altra metodologia (2)

- Si deriva la copertura minimale  $G$  di  $F$ .
- Si raggruppano le dipendenze in  $G$  in sottoinsiemi tali che ad ogni sottoinsieme  $G_i$  appartengono le dipendenze i cui membri sinistri hanno la stessa chiusura: i.e.  $X \rightarrow A$  e  $Y \rightarrow B$  appartengono a  $G_i$  se  $X^+ = Y^+$  secondo  $G$ .
- Si partizionano gli attributi  $U$  nei sottoinsiemi  $U_i$  individuati dai sottoinsiemi  $G_i$  del passo precedente. Se un sottoinsieme è contenuto in un altro si elimina.
- Si crea una relazione  $R_i(U_i)$  per ciascun sottoinsieme  $U_i$ , con associate le dipendenze  $G_i$ .
- Si aggiunge una relazione per gli attributi che non sono coinvolti in alcuna FD
- Se non c'è già una relazione che contenga una chiave della relazione originaria, si aggiunge

82

## esempio (metodologia 2)

Se le FD individuate su  $R(ABCDEFG)$  sono:

$AB \rightarrow CD$ ,  $AB \rightarrow E$ ,  $C \rightarrow F$ ,  $F \rightarrow G$

si generano gli schemi

$R1(ABCDE)$ ,  $R2(CF)$ ,  $R3(FG)$

83

## esempio (cont)

Se le FD su  $R(ABCD)$  sono:

$A \rightarrow BC$ ,  $B \rightarrow A$ ,  $C \rightarrow D$

si generano gli schemi

$R1(ABC)$ ,  $R2(CD)$  con  $A$  o  $B$  chiave in  $R1$

84

## esempio (cont)

Se le FD su R(ABCD) sono:

$A \rightarrow C$ ,  $B \rightarrow D$

si generano gli schemi

R1(AC), R2(BD), R3(AB)

85

## Confronto

- La prima metodologia garantisce come primo passo l'assenza di perdita sul join e poi conserva le dipendenze
- La seconda conserva le dipendenze e poi risolve l'eventuale perdita sul join

86

## In generale

- Una volta effettuata la decomposizione in 3NF con la metodologia precedente si verifica se lo schema ottenuto è anche BCNF
- Se la relazione ha una sola chiave allora le due forme normali coincidono
- N.B. nel secondo esempio questo non succede

87

## Qualità delle decomposizioni (2)

- Una decomposizione dovrebbe sempre garantire:
  - BCNF o 3NF
  - l'assenza di perdite, in modo da poter ricostruire le informazioni originarie
  - la conservazione delle dipendenze, in modo da mantenere i vincoli di integrità originari

88

- Quando una BCNF non è raggiungibile spesso è questione di cattiva progettazione

89

## Progettazione e normalizzazione

- la teoria della normalizzazione può essere usata nella progettazione logica per verificare lo schema relazionale finale
- si può usare anche durante la progettazione concettuale per verificare la qualità dello schema concettuale

90

Corso di Laurea in Ingegneria  
Informatica  
*Fondamenti di Informatica II*  
Modulo "*Basi di dati*"  
a.a. 2018-2019

Docente: Gigliola Vaglini  
Docente laboratorio: Francesco  
Pistolesi

Lezione 7

Vincoli di integrità

## CREATE TABLE (1)

```
CREATE TABLE Impiegato(  
    Matricola CHAR(6) PRIMARY KEY,  
    Nome CHAR(20) NOT NULL,  
    Cognome CHAR(20) NOT NULL,  
    Dipart CHAR(15),  
    Stipendio NUMERIC(9) DEFAULT 0,  
    FOREIGN KEY(Dipart) REFERENCES  
        Dipartimento(NomeDip),  
    UNIQUE (Cognome,Nome)  
)
```

## CREATE TABLE (2)

```
CREATE TABLE Infrazioni(  
    Codice CHAR(6) PRIMARY KEY,  
    Data DATE NOT NULL,  
    Vigile INTEGER NOT NULL  
        REFERENCES Vigili(Matricola),  
    Provincia CHAR(2),  
    Numero CHAR(6),  
    FOREIGN KEY(Provincia, Numero)  
        REFERENCES Auto(Provincia, Numero)  
        ON DELETE SET NULL  
        ON UPDATE CASCADE )
```

## Vincoli di integrità generici: check

- La clausola check permette di restringere i domini e specificare predicati che devono essere soddisfatti ogni volta che un valore viene assegnato ad una variabile in quel dominio.

5

## Vincoli di integrità generici: check

- Specifica di vincoli su attributi a livello di tabella
- check ( Condizione )
- La condizione è del tipo che può comparire in una clausola where

6

## Esempi

```
create table Impiegato
(
    Matricola character(6),
    Cognome character(20),
    Nome character(20),
    Dipart character(6),
    Ufficio character(6),
    Sesso character not null check (sesso in ('M','F'))
    Superiore character(6),
    Stipendio integer,
    check (Stipendio <= (select J.Stipendio
                           from Impiegato as J
                           where J.Matricola=Superiore)
    )
Non sempre supportata
```

7

## Sintassi, dettagli

- Sesso character not null check (sesso in ('M','F'))
  - Nella condizione è coinvolto un solo attributo
- Stipendio integer,
   
check (Stipendio <= (select J.Stipendio
 from Impiegato as J
 where J.Matricola= Superiore))
  - La condizione coinvolge più attributi

8

## CREATE DOMAIN, esempio

```
CREATE DOMAIN Voto  
AS SMALLINT DEFAULT NULL  
CHECK ( value >=18 AND value <= 30 )
```

## Vincoli di integrità generici: asserzioni

- Specifica vincoli a livello di schema della base di dati (cioè tra più tabelle)
- create assertion NomeAss  
check ( Condizione )
- Invece di mettere il vincolo in una sola delle tabelle coinvolte (poco chiaro) o in tutte (pericoloso), si mette allo stesso livello della definizione delle tabelle

## Esempi

- `create assertion AlmenoUnImpiegato  
check (1 <= ( select count(*) from  
Impiegato ))`

Anche questa non sempre supportata

11

## Asserzioni, commenti

- Con le assenze è possibile stabilire anche un qualunque vincolo predefinito
- Quando un'asserzione è stabilita, ogni variazione del database è consentita solo se non la viola
- Ad ogni assenza è associata una politica di controllo che può essere di due tipi:
  - Immediato (da verificare dopo ogni modifica)
  - Differito (da verificare dopo una sequenza di operazioni = transazione)

12

## Controllo dei vincoli

- Il costrutto per determinare (o per cambiare) il tipo di controllo associato ad un vincolo è il seguente
- `set constraints [NomeAss] (immediate | deferred)`

13

## Asserzioni, commenti

- Un vincolo immediato non soddisfatto causa l'annullamento dell'operazione di modifica che ha causato la violazione (rollback parziale)
- Un vincolo differito non soddisfatto causa l'annullamento della transazione che ha prodotto la violazione (rollback)

14

## Asserzioni, commenti

- I vincoli predefiniti sono di tipo immediato e possono essere quindi rappresentati da asserzioni con associata una politica di controllo immediato
- N.B. ad una asserzione non può però essere associata una politica di reazione alle violazioni come succede per i vincoli predefiniti

15

## Asserzioni

- Le asserzioni hanno un nome e quindi possono comparire all'interno di un'istruzione, ad es.
- drop NomeAss

16

## Basi di dati attive

- Una base di dati può contenere regole attive (chiamate *trigger*)
- Paradigma: Evento-Condizione-Azione
  - Quando un evento si verifica
  - Se la condizione è vera
  - Allora l'azione è eseguita
- Questo modello consente computazioni reattive
- Problema: è difficile realizzare applicazioni complesse

## Evento-Condizione-Azione

- **Evento**
  - Normalmente una modifica dello stato del database: insert, delete, update
  - Quando accade l'evento, il trigger è *attivato*
- **Condizione**
  - Un predicato che identifica se l'azione del trigger deve essere eseguita
  - Quando la condizione viene valutata, il trigger è *considerato*
- **Azione**
  - Una sequenza di update SQL o una procedura
  - Quando l'azione è eseguita anche il trigger è *eseguito*

## SQL:1999, Sintassi

- Lo standard SQL:1999 (SQL-3) sui trigger è stato fortemente influenzato da DB2 (IBM); gli altri sistemi non seguono lo standard (esistono dagli anni 80')
- Ogni trigger è caratterizzato da:
  - nome
  - target (tabella controllata)
  - modalità (`before` o `after`)
  - evento (`insert`, `delete` o `update`)
  - granularità (statement-level o row-level)
  - alias dei valori o tabelle di transizione
  - Azione
  - Timestamp di creazione

## SQL:1999, Sintassi

```
create trigger NomeTrigger
{before | after }
{insert | delete | update [of Column] } on
TabellaTarget
[referencing
{{old table [as] VarTuplaOld]
[new table [as] VarTuplaNew] } |
{{old [row] [as] VarTabellaOld]
[new [row] [as] VarTabellaNew] }}
[for each { row | statement }]
[when Condizione]
```

## Tipi di eventi

- **BEFORE**

- Il trigger è considerato e possibilmente eseguito prima dell'evento (i.e., la modifica del database)
- I trigger before non possono modificare lo stato del database; possono al più condizionare i valori "new" in modalità row-level (set t.new=expr)
- Normalmente questa modalità è usata quando si vuole verificare una modifica prima che essa avvenga e "modificare la modifica"

- **AFTER**

- Il trigger è considerato e eseguito dopo l'evento
- È la modalità più comune, adatta alla maggior parte delle applicazioni

## Esempio “before” e “after”

- 1. “Conditioner” (agisce prima dell’update e della verifica di integrità)

```
create trigger LimitaAumenti
before update of Salario on
    Impiegato
for each row
when (New.Salario > Old.Salario *
      1.2)
set New.Salario = Old.Salario *
      1.2
```

## Esempio “before” e “after”

- 2. “Re-installer” (agisce dopo l’update)

```
create trigger LimitaAumenti
after update of Salario on Impiegato
for each row
when (New.Salario > Old.Salario * 1.2)
set New.Salario = Old.Salario * 1.2
```

23

## Granularità degli eventi

- Modalità statement-level (di default, opzione **for each statement**)
  - Il trigger viene considerato e possibilmente eseguito solo una volta per ogni statement (comando) che lo ha attivato, indipendentemente dal numero di tuple modificate
  - In linea con SQL (set-oriented)

## Granularità degli eventi

- Modalità row-level (opzione `for each row`)
  - Il trigger viene considerato e possibilmente eseguito una volta per ogni tupla modificata
  - Scrivere trigger row-level è più semplice

25

## Clausola referencing

- Dipende dalla granularità
  - Se la modalità è row-level, ci sono due *variabili di transizione* (`old` and `new`) che rappresentano il valore precedente o successivo alla modifica di una tupla
  - Se la modalità è statement-level, ci sono due *tabelle di transizione* (`old table` and `new table`) che contengono i valori precedenti e successivi delle tuple modificate dallo statement
- `old` e `old table` non sono presenti con l'evento `insert`
- `new` e `new table` non sono presenti con l'evento `delete`

## Esempio di trigger row-level

```
create trigger AccountMonitor
  after update on Account
  for each row
  when new.Totale > old.Totale
  insert values
    (new.NumeroConto,
     new.Totale-old.Totale)
  into Pagamenti
```

## Esempio di trigger statement-level

```
create trigger ArchiviaFattureCancellate
  after delete on Fattura
  referencing old_table as VecchieFatture
  insert into FattureCancellate
  (select *
   from VecchieFatture)
```

## Esecuzione di Trigger in conflitto

- Quando vi sono più trigger associati allo stesso evento (in conflitto) vengono eseguiti come segue:
  - Per primi i **before** triggers (*statement-level* e *row-level*)
  - Poi viene eseguita la modifica e verificati i vincoli di integrità
  - Infine sono eseguiti gli **after** triggers (*row-level* e *statement level*)
- Quando vari trigger appartengono alla stessa categoria, l'ordine di esecuzione è definito in base al loro timestamp di creazione (i trigger più vecchi hanno priorità più alta)

## Controllo dell'accesso

- In SQL è possibile specificare chi (utente) e come (lettura, scrittura, ...) può utilizzare la base di dati (o parte di essa)
- Oggetto dei privilegi (diritti di accesso) sono di solito le tabelle, ma anche altri tipi di risorse, quali singoli attributi, viste o domini
- Un utente predefinito **\_system** (amministratore della base di dati) ha tutti i privilegi
- Il creatore di una risorsa ha tutti i privilegi su di essa

## Privilegi

- Un privilegio è caratterizzato da:
  - la risorsa cui si riferisce
  - l'utente che concede il privilegio
  - l'utente che riceve il privilegio
  - l'azione che viene permessa
  - la trasmissibilità del privilegio

## Tipi di privilegi offerti da SQL

- insert: permette di inserire nuovi oggetti (ennuple)
- update: permette di modificare il contenuto
- delete: permette di eliminare oggetti
- select: permette di leggere la risorsa
- references: permette la definizione di vincoli di integrità referenziale verso la risorsa (può limitare la possibilità di modificare la risorsa)
- usage: permette l'utilizzo in una definizione (per esempio, di un dominio)

## grant e revoke

- Concessione di privilegi:

grant < Privileges | all privileges > on  
Resource

to Users [ with grant option ]

– grant option specifica se il privilegio può  
essere trasmesso ad altri utenti

grant select on Department to Stefano

- Revoca di privilegi

revoke Privileges on Resource from Users  
[ restrict | cascade ]

## Autorizzazioni, commenti

- La gestione delle autorizzazioni deve "nascondere" gli elementi cui un utente non può accedere
- Come autorizzare un utente a vedere solo alcune tuple di una relazione?
  - Attraverso una vista:
    - Definiamo la vista con una condizione di selezione
    - Attribuiamo le autorizzazioni sulla vista, anziché sulla relazione di base

## Autorizzazioni, ancora

- (Estensioni di SQL:1999)
- Concetto di ruolo, cui si associano privilegi (anche articolati), poi concessi agli utenti attribuendo loro il ruolo

## Transazione

- Insieme di operazioni da considerare indivisibile ("atomico"), corretto anche in presenza di concorrenza e con effetti definitivi
- Proprietà ("acide"):
  - Atomicità
  - Consistenza
  - Isolamento
  - Durabilità (persistenza)

## Proprietà

- La sequenza di operazioni sulla base di dati viene eseguita per intero o per niente
- Al termine di una transazione, i vincoli di integrità debbono essere soddisfatti.  
"Durante" l'esecuzione ci possono essere violazioni, ma se ci sono alla terminazione allora la transazione deve essere annullata per intero ("abortita")

## Proprietà (2)

- L'effetto di transazioni concorrenti deve essere coerente (ad esempio "equivalente" alla loro esecuzione separata)
- La conclusione positiva di una transazione corrisponde ad un impegno (commit) a mantenere traccia del risultato anche in presenza di guasti e di esecuzione concorrente

## Transazioni in SQL

- Istruzioni fondamentali
  - begin transaction: specifica l'inizio della transazione (le operazioni non vengono eseguite sulla base di dati)
  - commit work: le operazioni specificate a partire dal begin transaction vengono eseguite sulla base di dati
  - rollback work: si rinuncia all'esecuzione delle operazioni specificate dopo l'ultimo begin transaction

## Procedure

- SQL:1999 (come già SQL-2) permette la definizione di procedure e funzioni (chiamate genericamente "stored procedures")
- Le stored procedures sono parte dello schema
 

```
procedure AssignCity(:Dep char(20), :City char(20))
  update Department
    set City = :City
  where Name = :Dep
```
- Lo standard prevede funzionalità limitate, le procedure sono composte da un singolo comando SQL, e non è molto recepito
- Molti sistemi offrono estensioni ricche (ad esempio Oracle PL/SQL e Sybase-Microsoft Transact SQL)

## SQL nei linguaggi di programmazione

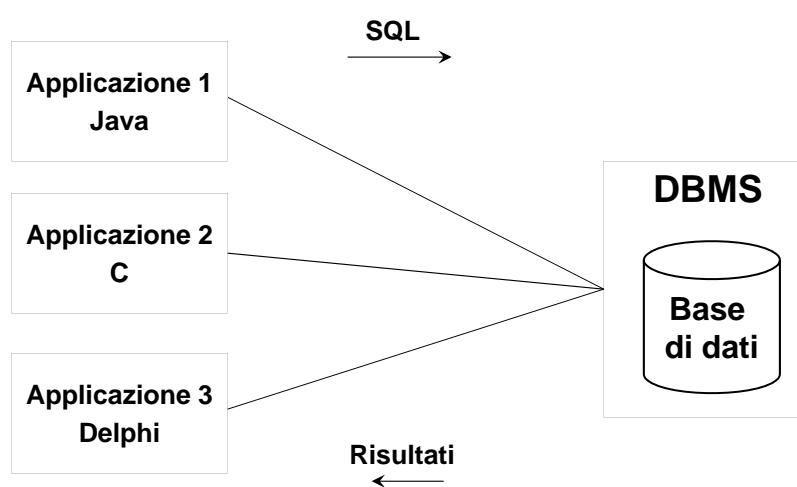
## SQL e applicazioni

- In applicazioni complesse, l'utente non vuole eseguire solo comandi SQL, ma programmi
- SQL non basta, sono necessarie altre funzionalità, per gestire:
  - input (scelte dell'utente e parametri)
  - output (con dati che non sono relazioni o se si vuole una presentazione complessa)
  - il controllo

## SQL e linguaggi di programmazione

- Le applicazioni sono scritte in
  - linguaggi di programmazione tradizionali:
    - Cobol, C, Java, Fortran
  - linguaggi "ad hoc", proprietari e non:
    - PL/SQL, Informix4GL, Delphi

## Applicazioni ed SQL: architettura



## Differenze, 1

- Accesso ai dati e correlazione:
  - linguaggio: dipende dal paradigma e dai tipi disponibili; ad esempio scansione di liste
  - SQL: join

## Differenze, 2

- tipi di base:
  - linguaggi: numeri, stringhe, booleani
  - SQL: CHAR, VARCHAR, DATE, ...
- costruttori di tipo:
  - linguaggio: dipende dal paradigma
  - SQL: relazioni e ennuple

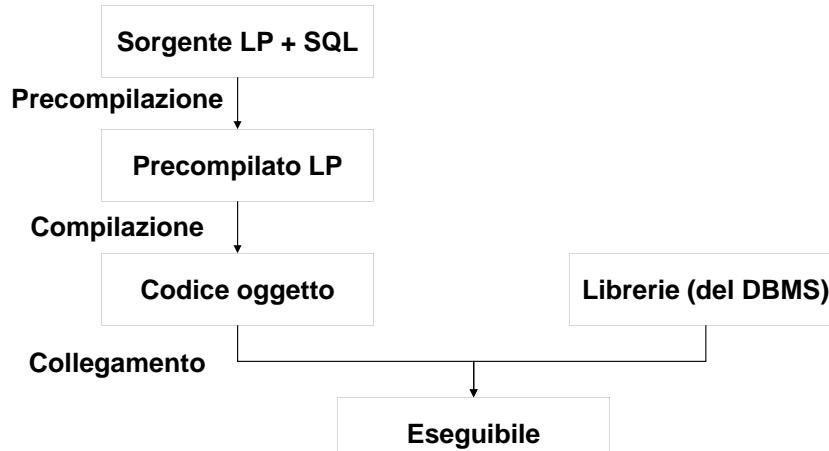
## Tecniche principali

- SQL immerso ("Embedded SQL")
  - sviluppata sin dagli anni '70
  - "SQL statico"
- SQL dinamico
- Call Level Interface (CLI)
  - più recente
  - SQL/CLI, ODBC, JDBC

## SQL immerso

- le istruzioni SQL sono "immerse" nel programma redatto nel linguaggio "ospite"
- un precompilatore (legato al DBMS) viene usato per analizzare il programma e tradurlo in un programma nel linguaggio ospite (sostituendo le istruzioni SQL con chiamate alle funzioni di una API del DBMS)

## SQL immerso, fasi



## Note

- Il precompilatore è specifico della combinazione linguaggio-DBMS-sistema operativo

## SQL immerso, un esempio

```
#include<stdlib.h>
main(){
    exec sql begin declare section;
    char *NomeDip = "Manutenzione";
    char *CittaDip = "Pisa";
    int NumeroDip = 20;
    exec sql end declare section;
    exec sql connect to utente@librobd;
    if (sqlca.sqlcode != 0) {
        printf("Connessione al DB non riuscita\n");
    } else {
        exec sql insert into Dipartimento
            values(:NomeDip,:CittaDip,:NumeroDip);
        exec sql disconnect all;
    }
}
```

## SQL immerso, commenti

- EXEC SQL denota le porzioni di interesse del precompilatore:
  - definizioni dei dati
  - istruzioni SQL
- le variabili del programma possono essere usate come “parametri” nelle istruzioni SQL (precedute da “：“) dove sintatticamente sono ammesse costanti

## SQL immerso, commenti 2

- `sqlca` è una struttura dati per la comunicazione fra programma e DBMS
- `sqlcode` è un campo di `sqlca` che mantiene il codice di errore dell'ultimo comando SQL eseguito:
  - zero: successo
  - altro valore: errore o anomalia

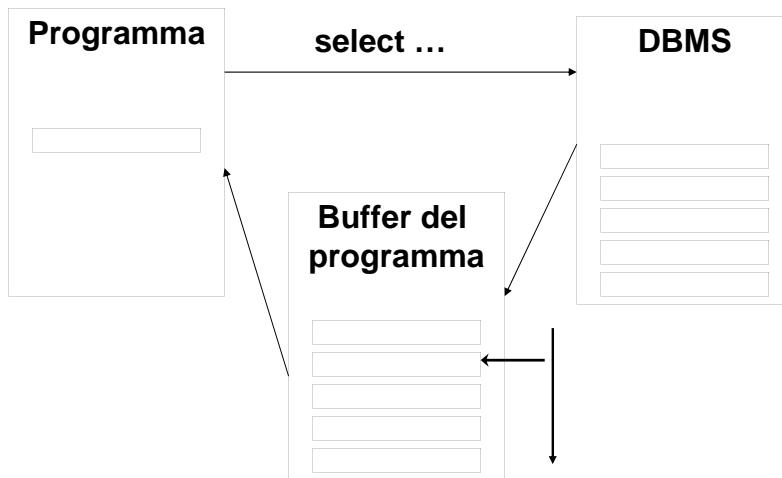
## Una difficoltà importante

- Conflitto di impedenza ("impedance mismatch")
  - Linguaggi di programmazione: operazioni su singole variabili o oggetti, si può accedere agli elementi di una tabella uno alla volta
  - SQL: operazioni su tavole che restituiscono tavole

## Interrogazioni in SQL immerso: conflitto di impedenza

- Il risultato di una select è costituito da zero o più ennuple:
  - zero o una: ok -- l'eventuale risultato può essere gestito in un record
  - più ennuple: come facciamo?
    - l'insieme (in effetti, la lista) non è gestibile facilmente in molti linguaggi
- Cursore: tecnica per trasmettere al programma una ennupla alla volta

## Cursore



## Nota

- **Il cursore**

- accede a tutte le ennuple di una interrogazione in modo globale (tutte insieme o a blocchi - è il DBMS che sceglie la strategia efficiente)
- trasmette le ennuple al programma una alla volta

## Operazioni sui cursori

Definizione del cursore

```
declare NomeCursore [ scroll ] cursor for Select ...
```

Esecuzione dell'interrogazione

```
open NomeCursore
```

Utilizzo dei risultati (una ennupla alla volta)

```
fetch NomeCursore into ListaVariabili
```

Disabilitazione del cursore

```
close cursor NomeCursore
```

Accesso alla ennupla corrente (di un cursore su singola relazione a fini di aggiornamento)

```
current of NomeCursore
```

nella clausola where

## Cursori, commenti

- Per aggiornamenti e interrogazioni "scalari" (cioè che restituiscono una sola tupla) il cursore non serve
- I cursori possono far scendere la programmazione ad un livello troppo basso, pregiudicando la capacità dei DBMS di ottimizzare le interrogazioni:
  - se "nidifichiamo" due o più cursori, rischiamo di reimplementare il join!

## SQL dinamico

- Non sempre le istruzioni SQL sono note quando si scrive il programma
- Allo scopo, è stata definita una tecnica completamente diversa, chiamata *Dynamic SQL* che permette di eseguire istruzioni SQL costruite dal programma (o addirittura ricevute dal programma attraverso parametri o da input)
- Non è banale gestire i parametri e la struttura dei risultati (non noti a priori)

## SQL dinamico

- Le operazioni SQL possono essere:
  - eseguite immediatamente  
`execute immediate SQLStatement`
  - prima "preparate":  
`prepare CommandName from SQLStatement`  
e poi eseguite (anche più volte):  
`execute CommandName [ into TargetList ] [ using ParameterList ]`

## Call Level Interface

- Indica genericamente interfacce che permettono di inviare richieste a DBMS per mezzo di parametri trasmessi a funzioni
- standard SQL/CLI ('95 e poi parte di SQL:1999)
- ODBC: implementazione proprietaria di SQL/CLI
- JDBC: una CLI per il mondo Java

## SQL immerso vs CLI

- SQL immerso permette
  - precompilazione (e quindi efficienza)
  - uso di SQL completo
- CLI
  - indipendente dal DBMS
  - permette di accedere a più basi di dati, anche eterogenee

Corso di Laurea in Ingegneria  
Informatica  
*Fondamenti di Informatica II*  
Modulo "Basi di dati"  
a.a. 2018-2019

Docente: Gigliola Vaglini  
Docente laboratorio: Francesco  
Pistolesi

1

## Lezione 8

Organizzazione fisica e  
gestione della memoria

2

## Tecnologia delle BD

- Il DBMS è una "scatola nera"
- Perché aprirla?
  - capire come funziona può essere utile per un migliore utilizzo

3

## DataBase Management System – DBMS

Sistema per gestire collezioni di dati:

- **grandi**
- **persistenti**
- **condivise**, garantendo **affidabilità e privatezza** .

- In più un DBMS deve essere **efficiente** (utilizzando al meglio le risorse di spazio e tempo del sistema) ed efficace (rendendo produttive le attività dei suoi utilizzatori).

4

## Le basi di dati sono grandi e persistenti

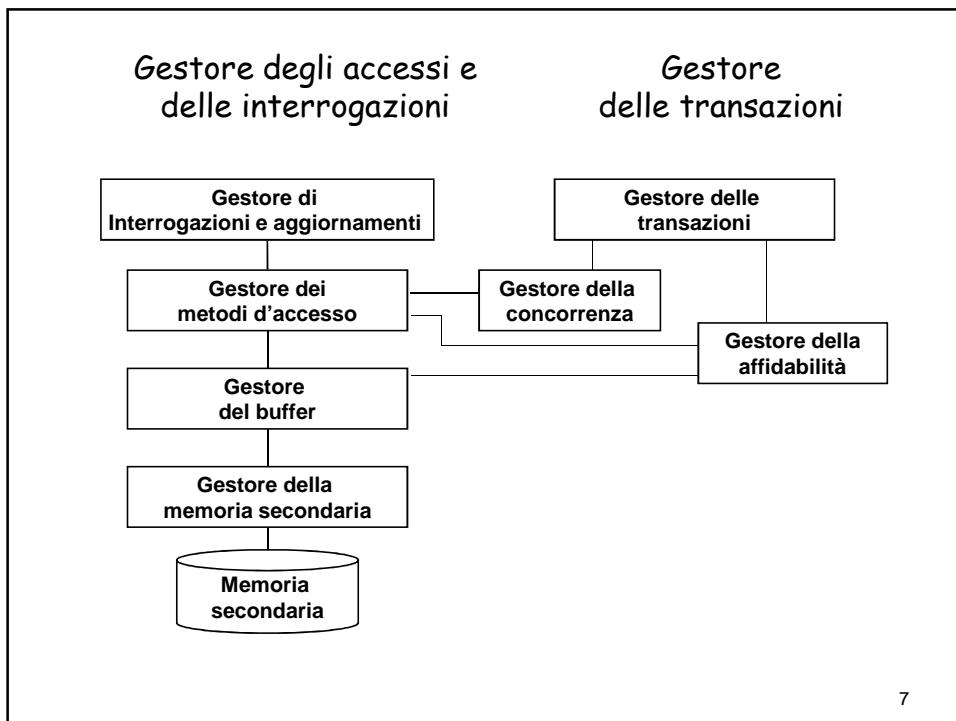
- La persistenza richiede la gestione della memoria secondaria
- La grandezza richiede che tale gestione sia sofisticata
- Gli utenti vedono il modello logico, ma le strutture logiche debbono essere gestite efficientemente in memoria secondaria:
  - servono strutture fisiche opportune

5

## Le basi di dati sono affidabili e condivise

- Le basi di dati debbono essere preservate anche in presenza di malfunzionamenti
- L'affidabilità è impegnativa per via degli aggiornamenti frequenti e della necessità di gestire il buffer
- Una base di dati è una risorsa **condivisa** fra le varie applicazioni
  - Attività diverse su dati in parte condivisi:
    - meccanismi di autorizzazione
  - Attività multi-utente su dati condivisi:
    - controllo della **concorrenza**

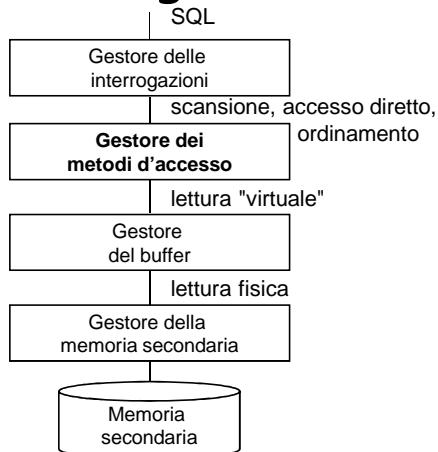
6



## Tecnologia delle basi di dati, argomenti

- Gestione della memoria secondaria e del buffer
- Organizzazione fisica dei dati
- Gestione ("ottimizzazione") delle interrogazioni
- Controllo della affidabilità
- Controllo della concorrenza

## Gestore degli accessi e delle interrogazioni



## DBMS e file system

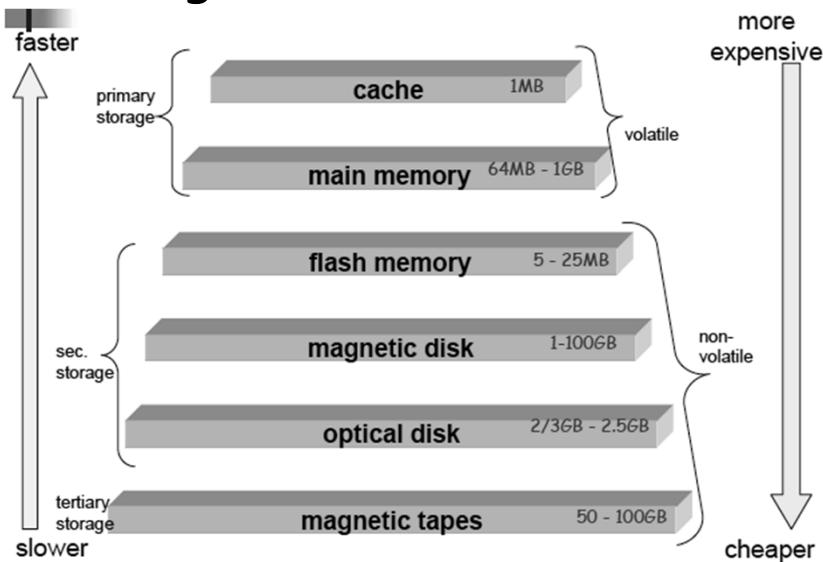
- Il file system è il componente del sistema operativo che gestisce la memoria secondaria
- I DBMS ne utilizzano le funzionalità per creare ed eliminare file e per leggere e scrivere singoli blocchi o sequenze di blocchi contigui.
- Il DBMS gestisce i file allocati come se fossero un unico grande spazio di memoria secondaria e costruisce, in tale spazio, le strutture fisiche con cui implementa le relazioni.

## Quando le basi di dati vengono interrogate ...

- I programmi possono fare riferimento solo a dati in memoria principale, quindi i dati in memoria secondaria possono essere utilizzati solo se prima trasferiti in memoria principale (questo spiega i termini "principale" e "secondaria) serve un'interazione fra memoria principale e secondaria che limiti il più possibile gli accessi alla secondaria

11

## La gerarchia di memoria



12

## Prestazioni di una memoria

- Dato un indirizzo di accesso, le prestazioni di memoria si misurano in termini della somma tra il tempo che la testina impiega per raggiungere la traccia di interesse, la latenza (tempo per accedere al primo byte del blocco di interesse) e il tempo di trasferimento (tempo necessario a muovere tutti i dati)

13

## Memoria principale e secondaria

- Accesso a memoria secondaria:
  - tempo di posizionamento della testina (10-50ms)
  - tempo di latenza (5-10ms)
  - tempo di trasferimento (1-2ms)in media non meno di 10 ms
- Il tempo di un accesso a memoria secondaria è quattro o più ordini di grandezza maggiore di quello per operazioni in memoria centrale
- Perciò, nelle applicazioni "I/O bound" (cioè con molti accessi a memoria secondaria e relativamente poche operazioni) il costo dipende esclusivamente dal numero di accessi a memoria secondaria

14

## Memoria principale e secondaria

- I dispositivi di memoria secondaria sono organizzati in **blocchi** di lunghezza (di solito) **fissa** (ordine di grandezza: alcuni KB)
- I record dei file sono memorizzati nei blocchi
- Le uniche operazioni sono la lettura e la scrittura dei dati di un blocco
- Accessi a blocchi "vicini" costano meno (contiguità): tempo di posizionamento + latenza=0
- La memoria principale è organizzata in pagine

15

## Buffer management

### • **Buffer:**

- area di memoria centrale, gestita dal DBMS (preallocata) e condivisa fra le transazioni
- organizzato in **pagine** di dimensioni pari o multiple di quelle dei blocchi di memoria secondaria (1KB-100KB)
- Se assumiamo che coincidano pagina e blocco, il caricamento di una pagina del buffer richiede una lettura in memoria secondaria, mentre salvare una pagina corrisponde ad una scrittura

16

## Dati gestiti dal buffer manager

- Il buffer
- Una directory che per ogni pagina mantiene (ad esempio)
  - il file fisico e il numero del blocco
  - due variabili di stato:
    - un contatore che indica quanti programmi utilizzano la pagina
    - un bit che indica se la pagina è "sporca", cioè se è stata modificata

17

## Il funzionamento del buffer manager

- Le politiche sono simili a quelle relative alla gestione della memoria da parte dei sistemi operativi:
  - "località dei dati": è alta la probabilità di dover riutilizzare i dati attualmente in uso
  - "legge 80-20" l'80% delle operazioni utilizza sempre lo stesso 20% dei dati

18

## Funzioni del buffer manager

- riceve richieste di lettura e scrittura (di pagine) dalle transazioni
- le esegue accedendo alla memoria secondaria solo quando indispensabile e utilizzando invece il buffer quando possibile

19

## Interfaccia

- esegue le primitive
  - **fix**: richiesta di una pagina; richiede una lettura solo se la pagina non è nel buffer (incrementa il contatore associato alla pagina offerta dal buffer manager)
  - **setDirty**: comunica che la pagina è stata modificata
  - **unfix**: indica che la transazione ha concluso l'utilizzo della pagina (decrementa il contatore associato alla pagina)
  - **force**: trasferisce in modo sincrono una pagina in memoria secondaria (su richiesta del gestore dell'affidabilità, non del gestore degli accessi)

20

## Esecuzione della fix

- Cerca la pagina nel buffer:
  - se c'è, restituisce l'indirizzo
  - altrimenti, cerca una pagina libera nel buffer (contatore a zero);
    - se la trova, vi inserisce i dati letti dalla memoria secondaria e ne restituisce l'indirizzo
    - altrimenti, due politiche alternative
      - "steal": selezione di una "vittima", pagina occupata del buffer; i dati della vittima sono scritti in memoria secondaria; vengono letti i dati di interesse dalla memoria secondaria e si restituisce l'indirizzo
      - "no-steal": l'operazione viene posta in attesa

21

## Scopo della gestione del buffer

- Ridurre il numero di accessi alla memoria secondaria
  - In caso di lettura, se la pagina è già presente nel buffer, non è necessario accedere alla memoria secondaria
  - In caso di scrittura, il gestore del buffer può decidere di differire la scrittura fisica (ammesso che ciò sia compatibile con la gestione dell'affidabilità) in modo da accorparla ad altre scritture

22

## Quindi

- Il buffer manager può far partire le scritture in due contesti diversi:
  - in modo **sincrono** quando è richiesto esplicitamente con una force
  - in modo **asincrono** quando lo ritiene opportuno (o necessario); in particolare, può decidere di anticipare o posticipare scritture per coordinarle e/o sfruttare la disponibilità dei dispositivi

23

## Tuple e blocchi

- I file sono logicamente organizzati in record
- I record sono mappati nei blocchi di memoria secondaria
- Le tuple di una relazione (record di file) stanno in blocchi contigui
- A volte in un blocco ci sono tuple di relazioni diverse ma correlate (i join sono favoriti)

24

## Blocchi e tuple

- I blocchi (componenti "fisici" di un file) e le tuple o record (componenti "logici" di una relazione) hanno dimensioni in generale diverse:
  - la dimensione del blocco dipende dal file system
  - la dimensione del record dipende dalle esigenze dell'applicazione, e può anche variare nell'ambito di un file

25

## Organizzazione delle tuple

- Ci sono varie alternative, anche legate ai metodi di accesso; in genere sono sistematiche sequenzialmente nei file, inoltre:
  - se la lunghezza delle tuple è fissa, la struttura può essere semplificata
  - alcuni sistemi possono spezzare le tuple su più blocchi (necessario per tuple grandi)

26

## Fattore di blocco

- numero di record in un blocco
  - $L_R$ : dimensione di un record (per semplicità costante nel file: "record a lunghezza fissa")
  - $L_B$ : dimensione di un blocco
  - se  $L_B > L_R$ , possiamo avere più record in un blocco:  
$$\lfloor L_B / L_R \rfloor$$
- lo spazio residuo può essere
  - utilizzato (record "spanned" o impaccati)
  - non utilizzato ("unspanned")

27

## Esercizio

- Calcolare il fattore di blocco e il numero di blocchi occupati da una relazione contenente  $T = 500000$  tuple di lunghezza fissa pari a  $L = 100$  byte in un sistema con blocchi di dimensione pari a  $B = 1$  kilobyte.

28

## Soluzione

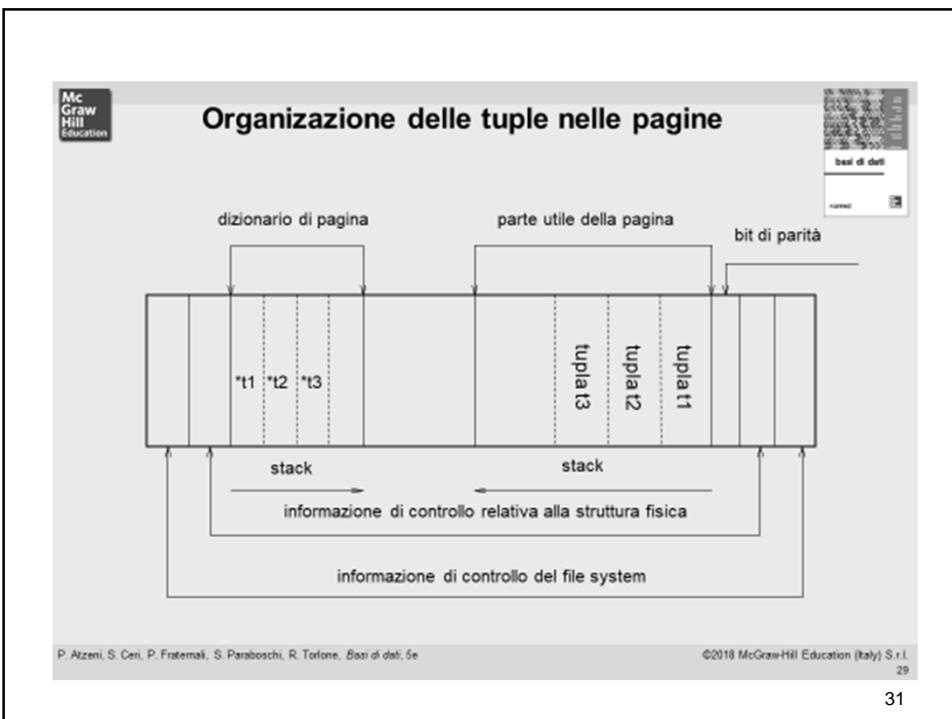
- $N_B = D_T/B \quad D_T = T^*L$
- $N_B = 50000000/1024$
- $F_B = B/L$   
 $F_B = 1024/100$

29

Considerando la dimensione della pagina uguale a quella del blocco..

- Ad esempio nel buffer
- L'organizzazione delle tuple nella pagina è simile

30



## Operazioni sulla pagina

- Inserimento/modifica di una tupla
  - Può richiedere l'allocazione di nuovo spazio
- Cancellazione
- Accesso ad una tupla o ad un campo di una tupla

## Il file system usato dal DBMS

- I DBMS tramite i sistemi operativi gestiscono il file system per memorizzare il database
- Si occupano quindi di fare l'accesso alle strutture fisiche che contengono i dati

33

## Strutture primarie e secondarie

- Le tuple organizzate all'interno dei blocchi dei file costituiscono le
  - Strutture primarie, cioè quelle che contengono propriamente i dati
- Esistono anche blocchi contenenti
  - Strutture secondarie, che sono quelle che favoriscono l'accesso ai dati senza contenerli

34

## Strutture primarie

- Possono avere una organizzazione detta **sequenziale**
- L'organizzazione sequenziale può essere
  - **seriale**: ordinamento fisico ma non logico
  - **ordinata**: con ordinamento delle tuple coerente con quello di un campo
  - **array**: posizione individuate tramite indici

35

## Organizzazione seriale

- Chiamata anche:
  - "Entry sequenced"
  - file heap
  - file disordinato
- Gli inserimenti vengono effettuati
  - in coda (con riorganizzazioni periodiche)
  - al posto di record cancellati

36

## Strutture sequenziali ordinate

- Le strutture ordinate permettono ricerche binarie
- Il problema è mantenere l'ordinamento

37

## Con accesso calcolato

- I file hash permettono un accesso diretto molto efficiente
  - La tecnica si basa su quella utilizzata per le tavole hash in memoria centrale

38

## Remind: Tavola hash

- Obiettivo: accesso diretto ad un insieme di record sulla base del valore di un campo (detto **chiave**, che per semplicità supponiamo identificante, ma non è necessario)
- Se i possibili valori della chiave sono in numero paragonabile al numero di record allora usiamo un array; ad esempio: università con 1000 studenti e numeri di matricola compresi fra 1 e 1000 o poco più e file con tutti gli studenti
- Se i possibili valori della chiave sono molti di più di quelli effettivamente utilizzati, non possiamo usare l'array (spreco); ad esempio:
  - 40 studenti e numero di matricola di 6 cifre (un milione di possibili chiavi)

39

## Tavola hash

- senza sprecare spazio
  - **funzione hash:**
    - associa ad ogni valore della chiave un "indirizzo", in uno spazio di dimensione leggermente superiore rispetto a quello strettamente necessario
    - poiché il numero di possibili chiavi è molto maggiore del numero di possibili indirizzi, la funzione non può essere iniettiva e quindi esiste la possibilità di collisioni (chiavi diverse che corrispondono allo stesso indirizzo)
    - le buone funzioni hash distribuiscono in modo causale e uniforme, riducendo le probabilità di collisione (che si riduce aumentando lo spazio ridondante)

40

## Un esempio

- 40 record
- tavola hash con 50 posizioni:
  - 1 collisione a 4
  - 2 collisioni a 3
  - 5 collisioni a 2

M	M mod 50
60600	0
66301	1
205751	1
205802	2
200902	2
116202	2
200604	4
66005	5
116455	5
200205	5
201159	9
205610	10
201260	10
102360	10
205460	10
205912	12
205762	12
200464	14
205617	17
205667	17
200268	18
205619	19
210522	22
205724	24
205977	27
205478	28
200430	30
210533	33
205887	37
200138	38
102338	38
102690	40
115541	41
206092	42
205693	43
205845	45
200296	46
205796	46
200498	48
206049	49

41

## Risoluzione delle collisioni

- Varie tecniche:
  - posizioni successive disponibili
  - tabella di overflow (gestita in forma collegata)
  - funzioni hash "alternative"
- Nota:
  - le collisioni ci sono (quasi) sempre
  - le collisioni multiple hanno probabilità che decresce al crescere della molteplicità
  - la molteplicità media delle collisioni è molto bassa

42

## Hash su file

- L'idea è la stessa, ma si sfrutta l'organizzazione in blocchi e il fatto che l'accesso è al blocco
- In questo modo si "ammortizzano" le probabilità di collisione

43

## Un esempio

- 40 record
- tavola hash con 50 posizioni:
  - 1 collisione a 4
  - 2 collisioni a 3
  - 5 collisioni a 2
- file hash con fattore di blocco 10;  
5 blocchi con 10 posizioni ciascuno, la funzione hash restituisce un indirizzo tra 0 e 4:
  - due soli overflow!

44

## Un file hash

60600	66301	205802	200268	200604
66005	205751	200902	205478	201159
116455	115541	116202	210533	200464
200205	200296	205912	200138	205619
205610	205796	205762	102338	205724
201260		205617	205693	206049
102360		205667	200498	
205460		210522		
200430		205977		
102690		205887		
205845		206092		

45

- Il record che configge si memorizza nel record successivo
- Quando si tratterà di trovarlo si dovranno fare due accessi a blocchi
- In tutti gli altri casi si dovrà fare un solo accesso.

46

## Strutture ad albero (non sequenziali)

- Dette anche indici
- Strutture basate sull'uso di puntatori ma l'accesso è in base ai valori di uno o più campi
- Si utilizzano sia come strutture primarie che secondarie

47

## Strutture ad albero

- Indice:
  - struttura per l'accesso (efficiente) ai record sulla base dei valori di un campo (o di una "concatenazione di campi") detto chiave (o, meglio, pseudochiave, perché non è necessariamente identificante);
- Un indice I di un file f è un altro file, con record a due campi: chiave e indirizzo (dei record di f o dei relativi blocchi), ordinato secondo i valori della chiave
  - Ad es., l'indice analitico di un libro: lista di coppie (termine,pagina), ordinata alfabeticamente sui termini, posta in fondo al libro e separabile da esso

48

## Tipi di indice

- **indice primario:**
  - su un campo sul cui ordinamento è basata la memorizzazione
- **indice secondario**
  - su un campo con ordinamento diverso da quello dell'ordinamento di memorizzazione
- **indice denso**
  - contiene un record per ciascun record del file
- **indice sparso**
  - contiene un numero di record inferiore rispetto a quelli del file

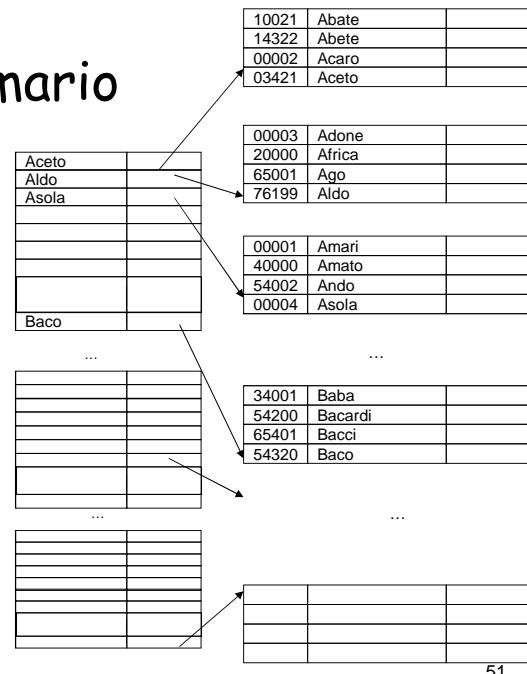
49

## Tipi di indice, commenti

- Un indice primario può essere sparso (non tutti i valori della chiave compaiono nell'indice)
  - Esempio, sempre rispetto ad un libro
    - indice generale (cap.1, sez.1)
- Gli indici secondari sono densi perché tutti i valori della chiave devono essere raggiungibili
- Ogni file può avere al più un indice primario e un numero qualunque di indici secondari (su campi diversi). Esempio:
  - una guida turistica può avere l'indice dei luoghi e quello degli artisti

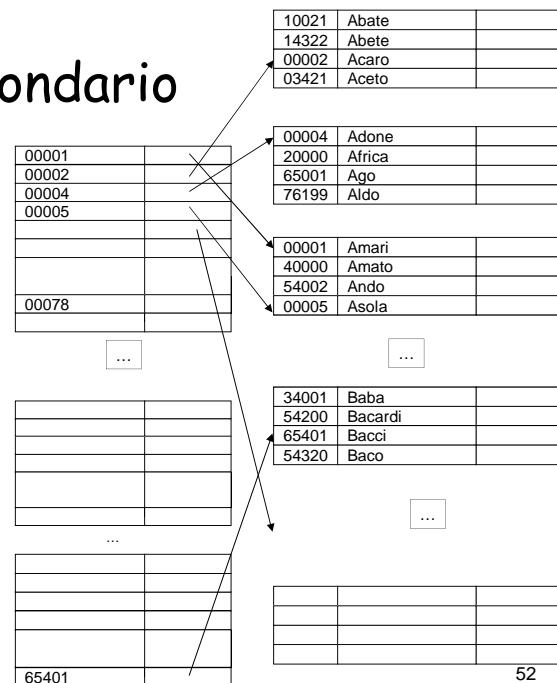
50

## Indice primario



51

## Indice secondario



52

## Caratteristiche degli indici

- Accesso diretto (sulla chiave) efficiente
- Scansione sequenziale ordinata efficiente
- Modifiche della chiave, inserimenti, eliminazioni inefficienti (come nei file ordinati)
  - tecniche per alleviare i problemi:
    - file o blocchi di overflow
    - marcatura per le eliminazioni
    - riempimento parziale
    - blocchi collegati (non contigui)
    - riorganizzazioni periodiche

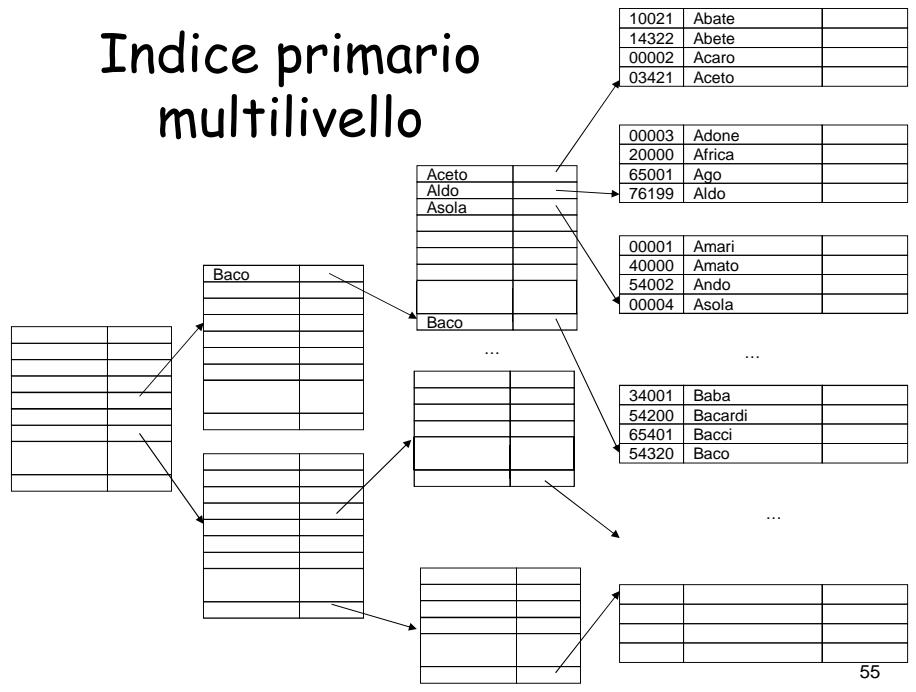
53

## Indici multilivello

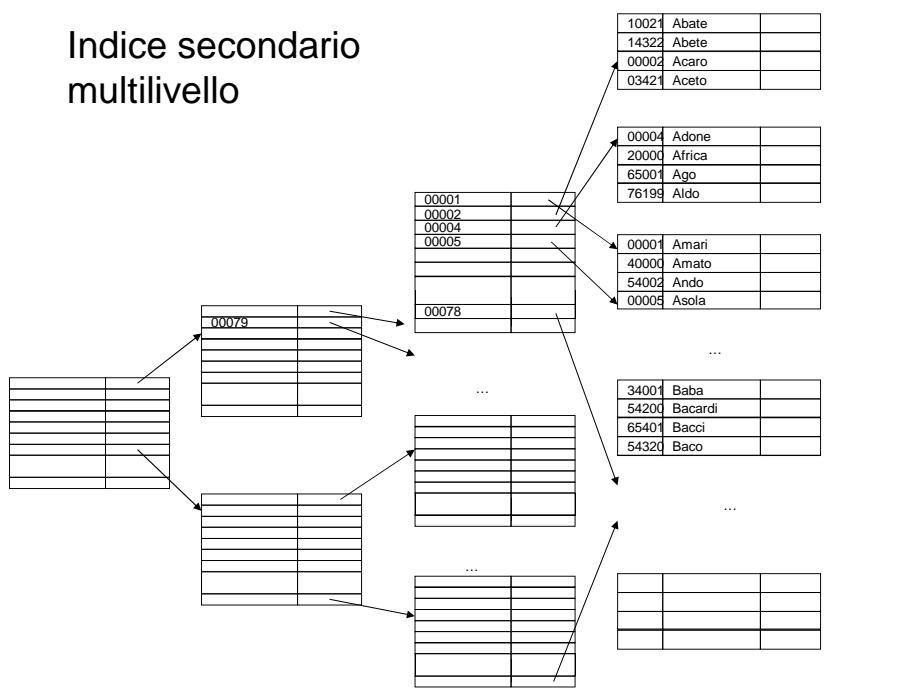
- Gli indici sono file essi stessi e quindi ha senso costruire indici sugli indici, per evitare di fare ricerche fra blocchi diversi
- Possono esistere più livelli fino ad avere il livello più alto con un solo blocco; i livelli sono di solito abbastanza pochi, perché
  - l'indice è ordinato, quindi l'indice sull'indice è sparso
  - i record dell'indice sono piccoli

54

## Indice primario multilivello



## Indice secondario multilivello



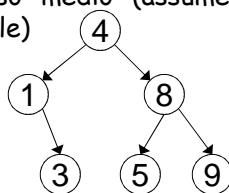
## Indici, problemi

- Le strutture di indice basate su strutture ordinate sono poco flessibili in presenza di elevata dinamicità
- Gli indici utilizzati dai DBMS sono in generale
  - indici dinamici multilivello
  - Vengono memorizzati e gestiti come B-tree (intuitivamente: alberi di ricerca bilanciati)
    - Alberi binari di ricerca
    - Alberi n-ari di ricerca
    - Alberi n-ari di ricerca bilanciati

57

## Albero binario di ricerca

- Albero binario etichettato in cui per ogni nodo il sottoalbero sinistro contiene solo etichette minori di quella del nodo e il sottoalbero destro etichette maggiori
- tempo di ricerca (e inserimento), pari alla profondità:
  - logaritmico nel caso "medio" (assumendo un ordine di inserimento casuale)



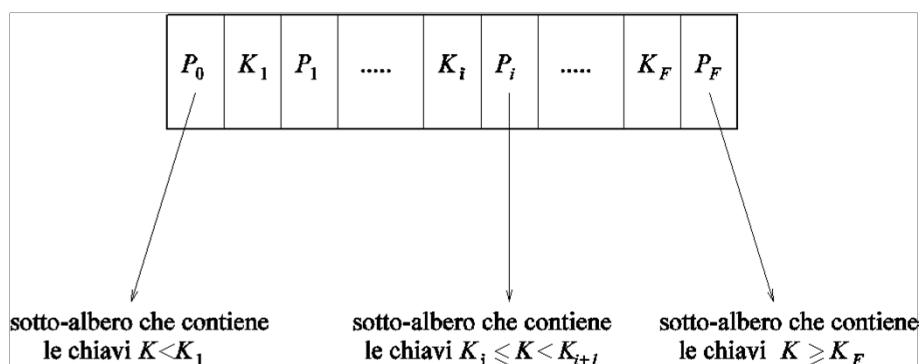
58

## Albero di ricerca di ordine P

- Ogni nodo ha (fino a) P figli e (fino a) P-1 etichette, ordinate
- Nell'i-esimo sottoalbero abbiamo tutte etichette maggiori della (i-1)- esima etichetta e minori della (i+1)-esima
- Ogni ricerca o modifica comporta la visita di un cammino radice foglia
- In strutture fisiche, un nodo può corrispondere ad un blocco
- Un B-tree è un albero di ricerca che viene mantenuto bilanciato, grazie a:
  - Riempimento parziale (mediamente 70%)
  - Riorganizzazioni (locali) in caso di sbilanciamento

59

## Organizzazione dei nodi del B-tree



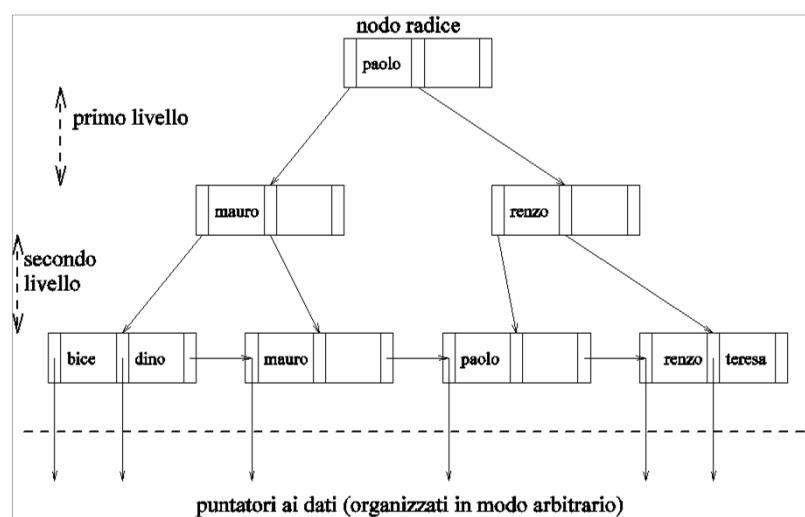
60

## B tree e B+ tree

- B+ tree:
  - le foglie sono collegate in una lista
  - molto usati nei DBMS
- B tree:
  - I nodi intermedi possono avere puntatori direttamente ai dati

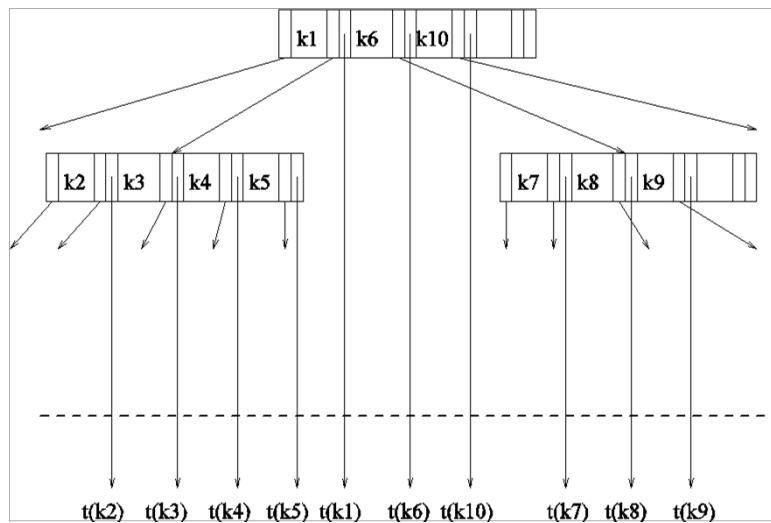
61

## Un B+ tree



62

## Un B-tree



63

## Definizione degli indici SQL

- Non è standard, ma presente in forma simile nei vari DBMS
  - `create [unique] index IndexName on TableName(AttributeList)`
  - `drop index IndexName`

64

## Strutture fisiche nei DBMS relazionali

- Struttura primaria:
  - disordinata (heap, "unclustered")
  - ordinata ("clustered")
  - hash ("clustered")
- Indici (densi/sparsi, semplici/composti):
  - Organizzazione sequenziale (statica), di solito su struttura ordinata
  - B-tree (dinamico)

65

## Strutture fisiche in alcuni DBMS

- Oracle:
  - struttura primaria
    - file heap
    - "hash cluster" (cioè struttura hash)
    - cluster (anche plurirelatazionali) anche ordinati (con B-tree denso)
  - indici secondari di vario tipo (B-tree, bitmap, funzioni)

66

## Strutture fisiche in alcuni DBMS

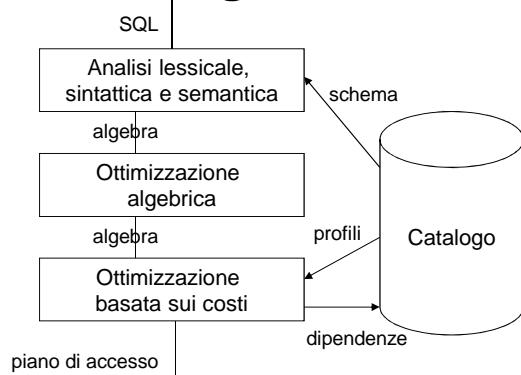
- DB2:
  - primaria: heap o ordinata con B-tree denso
  - indice sulla chiave primaria (automaticamente)
  - indici secondari B-tree densi
- SQL Server:
  - primaria: heap o ordinata con indice B-tree sparso
  - indici secondari B-tree densi

67

## Esecuzione e ottimizzazione delle interrogazioni

- **Query processor** (o **Ottimizzatore**): un modulo del DBMS
- Più importante nei sistemi attuali che in quelli "vecchi" (gerarchici e reticolari):
  - le interrogazioni sono espresse ad alto livello (ricordare il concetto di **indipendenza dei dati**):
    - insiemi di tuple
    - poca proceduralità
  - l'ottimizzatore sceglie la strategia realizzativa (di solito fra diverse alternative), a partire dall'istruzione SQL

## Il processo di esecuzione delle interrogazioni



## Ottimizzazione algebrica

- Il termine ottimizzazione è improprio (anche se efficace) perché il processo utilizza euristiche
- Si basa sulla nozione di equivalenza:
  - Due espressioni sono equivalenti se producono lo stesso risultato qualunque sia l'istanza attuale della base di dati
- I DBMS cercano di eseguire espressioni equivalenti a quelle date, ma meno "costose"
- Euristica fondamentale:
  - selezioni e proiezioni il più presto possibile (per ridurre le dimensioni dei risultati intermedi):
    - "push selections down"
    - "push projections down"

## "Profili" delle relazioni

- **Informazioni quantitative:**
  - cardinalità di ciascuna relazione
  - dimensioni delle tuple
  - dimensioni dei valori
  - numero di valori distinti degli attributi
  - valore minimo e massimo di ciascun attributo
- Utilizzate nella fase finale dell'ottimizzazione, per stimare le dimensioni dei risultati intermedi
- l'ordine in cui si fanno i join va valutato:  
dopo l'ottimizzazione algebrica (si ottengono tutte le espressioni minimali equivalenti) va fatta quella in base ai costi

71

## Esecuzione delle operazioni

- I DBMS implementano gli operatori dell'algebra relazionale (o meglio, loro combinazioni) per mezzo di operazioni di livello abbastanza basso, che però possono implementare vari operatori "in un colpo solo"
- Operatori fondamentali:
  - accesso diretto
  - scansione
- A livello più alto:
  - ordinamento
- Ancora più alto
  - Join, l'operazione più costosa

72

## Ottimizzazione basata sui costi

- Un problema articolato, con scelte relative a:
  - operazioni da eseguire (es.: scansione o accesso diretto?)
  - ordine delle operazioni (es. join di tre relazioni; ordine?)
  - i dettagli del metodo (es.: quale metodo di join)
- Architetture parallele e distribuite aprono ulteriori gradi di libertà

73

## Accesso diretto

- Può essere eseguito solo se le strutture fisiche lo permettono
  - indici
  - strutture hash

## Accesso diretto basato su indice

- Efficace per interrogazioni (sulla "chiave dell'indice")
  - "puntuali" ( $A_i = v$ )
  - su intervallo ( $v_1 \leq A_i \leq v_2$ )
- Per predici conguntivi
  - si sceglie il più selettivo per l'accesso diretto e si verifica poi sugli altri dopo la lettura (e quindi in memoria centrale)
- Per predici disgiuntivi:
  - servono indici su tutti, ma conviene usarli se molto selettivi e facendo attenzione ai duplicati

## Accesso diretto basato su hash

- Efficace per interrogazioni (sulla "chiave dell'indice")
  - "puntuali" ( $A_i = v$ )
  - NON su intervallo ( $v_1 \leq A_i \leq v_2$ )
- Per predici conguntivi e disgiuntivi, vale lo stesso discorso fatto per gli indici

## Indici e hash su più campi

- Indice su cognome e nome
  - funziona per accesso diretto su cognome?
  - funziona per accesso diretto su nome?
- Hash su cognome e nome
  - funziona per accesso diretto su cognome?
  - funziona per accesso diretto su nome?

## Ordine del join

- Join associativo
- La scelta dell'ordine delle coppie a cui si applica prima l'operatore è rilevante
- Attributo di join chiave

## Metodi di join

- Nested loop
- Merge scan
- Hash-based

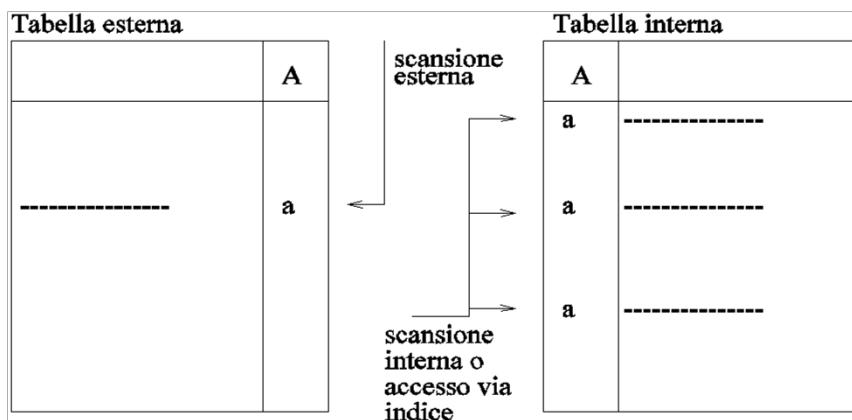
79

## nested loop

- per ogni tupla nella tabella esterna si esaminano tutte le tuple di quella interna per verificare la condizione di join.
- date R ed S, si hanno due possibilità R esterna o R interna
- il costo dipende dal numero di accessi e dal fatto che la tabella esterna sia piccola

80

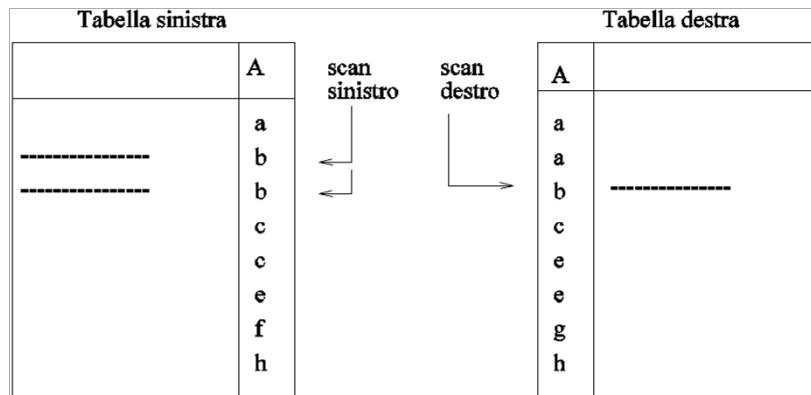
## Nested-loop



## merge scan

- si ordinano le tabelle in base agli attributi di join
- si trova l'elemento della seconda tabella da cui partire rispetto al primo elemento della prima tabella e poi si continua da lì
- il costo è l'ordinamento

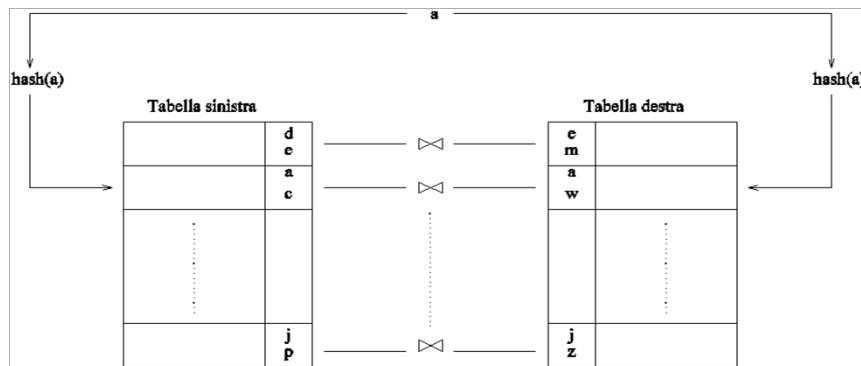
## Merge-scan



## hash

- è costruita una tabella ad accesso hash sull'attributo di join per cercare le tuple corrispondenti più velocemente
- ogni riga della prima tabella è inserita in una tabella aggiuntiva, poi si cercano le tuple della seconda con la stessa codifica hash
- serve memoria ed è migliore del nested loop nel caso di equijoin

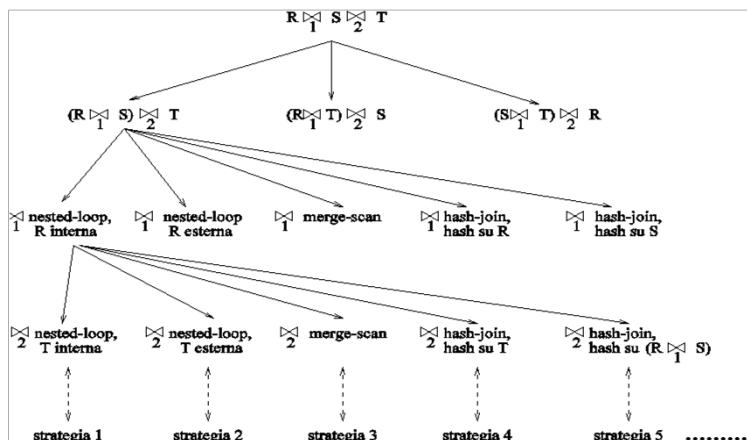
## Hash join



## Il processo di ottimizzazione

- Si costruisce un albero di decisione con le varie alternative ("**piani di esecuzione**")
- Si valuta il costo di ciascun piano
- Si sceglie il piano di costo minore
- L'ottimizzatore trova di solito una "buona" soluzione, non necessariamente l'"ottimo"

## Un albero di decisione



87

## Progettazione fisica

- La fase finale del processo di progettazione di basi di dati
- **input**
  - lo schema logico e informazioni sul carico applicativo
- **output**
  - schema fisico, costituito dalle definizioni delle relazioni con le relative strutture fisiche (e molti parametri, spesso legati allo specifico DBMS)

## Progettazione fisica nel modello relazionale

- La caratteristica comune dei DBMS relazionali è la disponibilità degli indici:
  - la progettazione logica spesso coincide con la scelta degli indici (oltre ai parametri strettamente dipendenti dal DBMS)
- Le chiavi (primarie) delle relazioni sono di solito coinvolte in selezioni e join: molti sistemi prevedono (oppure suggeriscono) di definire indici sulle chiavi primarie
- Altri indici vengono definiti con riferimento ad altre selezioni o join "importanti"
- Se le prestazioni sono insoddisfacenti, si "tara" il sistema aggiungendo o eliminando indici
- È utile verificare se e come gli indici sono utilizzati con il comando SQL `show plan`

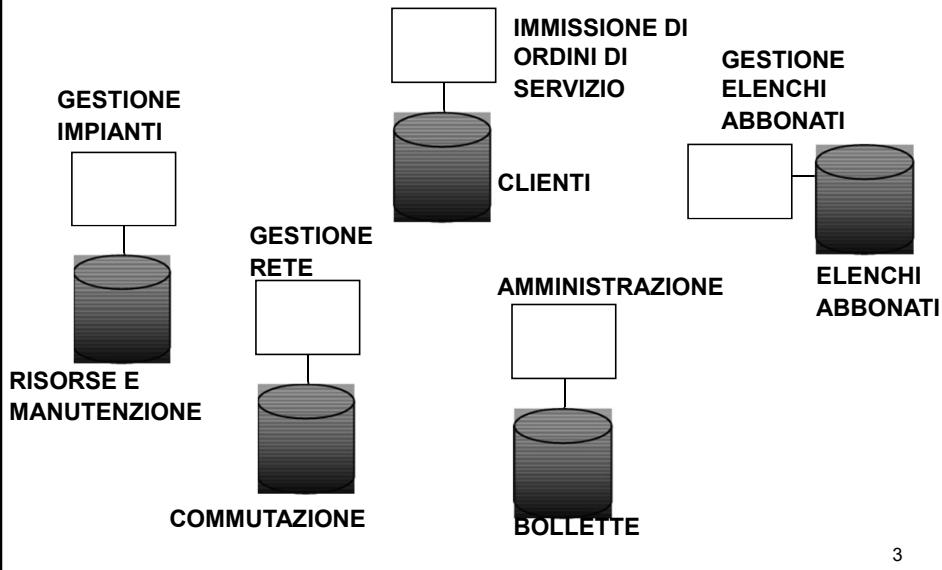
Corso di Laurea in Ingegneria  
Informatica  
*Fondamenti di Informatica II*  
Modulo "Basi di dati"  
a.a. 2018-2019

Docente: Gigliola Vaglini  
Docente laboratorio: Francesco  
Pistolesi

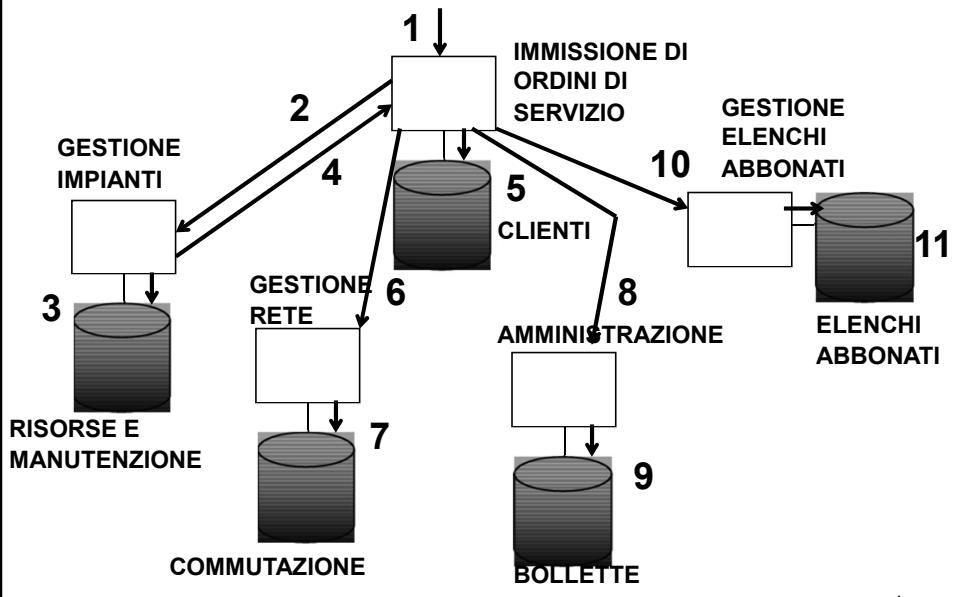
Lezione 9

Gestione delle transazioni

## Esempio di sistema informativo



## Esempio di transazione

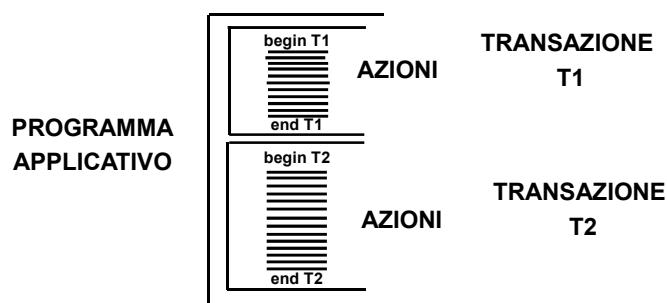


## Definizione di transazione

- Transazione: parte di programma caratterizzata da un inizio (**begin-transaction**, **start transaction** in SQL), una fine (**end-transaction**, non esplicitata in SQL) 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
- Un **sistema transazionale** è in grado di definire ed eseguire transazioni per conto di un certo numero di applicazioni concorrenti

5

## Applicazioni e transazioni



6

## Una transazione

```
start transaction;  
update ContoCorrente  
    set Saldo = Saldo + 10 where  
        NumConto = 12202;  
update ContoCorrente  
    set Saldo = Saldo - 10 where  
        NumConto = 42177;  
commit work;
```

7

## Una transazione con varie decisioni

```
start transaction;  
update ContoCorrente  
    set Saldo = Saldo + 10 where NumConto =  
        12202;  
update ContoCorrente  
    set Saldo = Saldo - 10 where NumConto =  
        42177;  
select Saldo as A  
    from ContoCorrente  
    where NumConto = 42177;  
if (A>=0)  then commit work  
            else rollback work;
```

8

## Proprietà delle transazioni

- Proprietà "ACID"
  - Atomicità
  - Consistenza
  - Isolamento
  - Durata (persistenza)

9

## Atomicità

- Una transazione non può lasciare la base di dati in uno stato intermedio
  - un guasto o un errore prima del commit debbono causare l'annullamento delle operazioni svolte
  - un guasto o errore dopo il commit non deve avere conseguenze; se necessario vanno ripetute le operazioni già fatte
- L'esito normale è il Commit : è il più frequente (99% ?)
  - L'abort (o rollback) può essere richiesto dall'applicazione = suicidio, oppure dal sistema (violazione dei vincoli, concorrenza) = omicidio

## Consistenza

- La transazione rispetta i vincoli di integrità'
- Conseguenza:
  - se lo stato iniziale e' corretto
  - anche lo stato finale e' corretto

## Isolamento

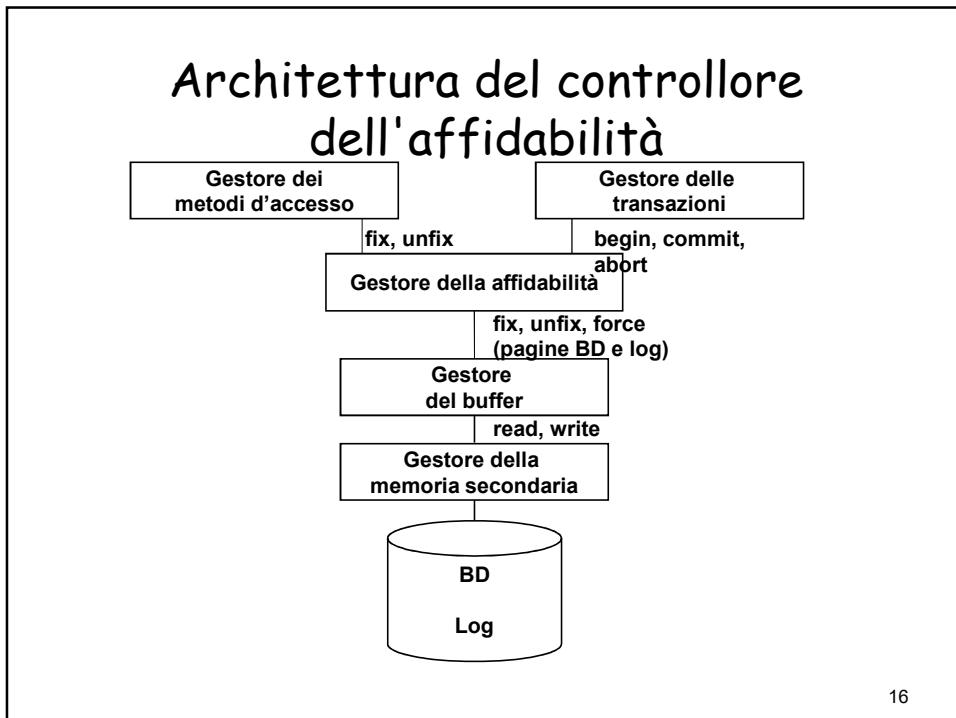
- La transazione non risente degli effetti delle altre transazioni concorrenti
  - l'esecuzione concorrente di una collezione di transazioni deve produrre un risultato che si potrebbe ottenere con una esecuzione sequenziale
- Conseguenza: una transazione non espone i suoi stati intermedi

## Durabilità (Persistenza)

- Gli effetti di una transazione andata in commit non vanno perduti ("durano per sempre"), anche in presenza di guasti

## Transazioni e moduli di DBMS

- Atomicità e durabilità
  - Gestore dell'affidabilità
- Isolamento:
  - Gestore della concorrenza
- Consistenza:
  - Gestore dell'integrità a tempo di esecuzione



## Gestore dell'affidabilità

- Gestisce l'esecuzione dei comandi transazionali
  - start transaction (*B*)
  - commit work (*C*)
  - rollback work (*A*)
- 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

17

## Persistenza delle memorie

- **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
    - ...

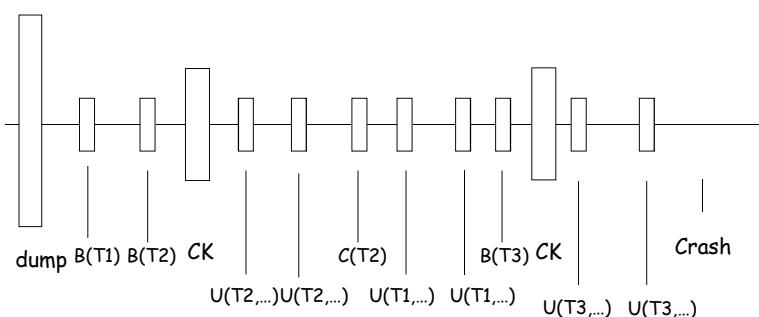
18

## Il log

- 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

19

## Struttura del log



20

## Log, checkpoint e dump: a che cosa servono?

- 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 (vedi avanti)

21

## Scritture nel log

- I record nel log sono di due tipi
  - Record di sistema: checkpoint e dump vengono scritti dal controllore dell'affidabilità
  - Record di transazione: attività svolte dalle transazioni nell'ordine in cui sono svolte (begin, commit, rollback, insert, delete, update)

22

## Checkpoint

- 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, cioè le transazioni "a metà strada"
  - e, dualmente, di confermare che le altre o non sono iniziate o sono finite; infatti per tutte le transazioni che hanno effettuato il commit i dati sono in memoria di massa

23

## Descrizione dell'operazione Ceckpoint

- Si sospende l'accettazione delle operazioni di write, commit, abort da parte delle transazioni
- Si forza (force) la scrittura in memoria di massa delle pagine del buffer modificate da transazioni che hanno fatto commit
- Si forza (force) la scrittura nel log di un record contenente gli identificatori delle transazioni attive
- Si riprendono ad accettare le operazioni da parte delle transazioni

24

## Dump

- Copia completa ("di riserva") della base di dati
  - Solitamente prodotta mentre il sistema non è operativo
  - Salvato in memoria stabile, come *backup*
  - Un record di *dump* nel log indica il momento in cui il log è stato effettuato (e dettagli pratici, file, dispositivo, ...)

25

## Record di transazione

- Begin,commit,rollback: identificativo transazione (T)
- Update: T, O, BS (before state), AS (after state)
- Insert: T, AS
- Delete: T, BS

26

## Significato delle operazioni di Undo e Redo

- 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:
  - $undo(undo(A)) = undo(A)$
  - $redo(redo(A)) = redo(A)$

27

## Esito di una transazione

- L'esito di una transazione è determinato irrevocabilmente quando viene scritto il record di **commit** nel log
  - un 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

28

**Quando il controllore dell'affidabilità può consentire la modifica del log da parte delle transazioni**

- **Regola Write-Ahead-Log:**

- si scrive la parte BS dei record del log prima di effettuare la corrispondente operazione sul database
  - consente di disfare le azioni di transazioni senza commit avendo in memoria stabile un valore corretto

- **Regola Commit-Precedenza:**

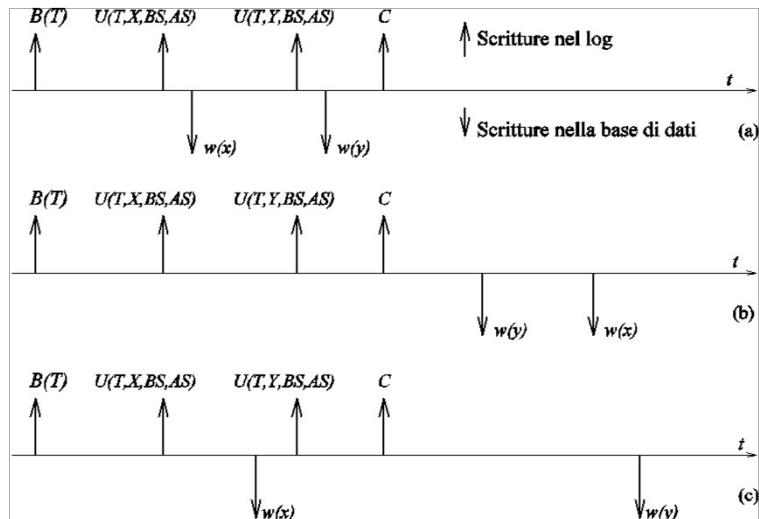
- si scrive la parte AS dei record di log prima del commit
  - consente di rifare le azioni di transazioni che hanno già fatto commit

29

**E la base di dati?**

- **Quando scriviamo nella base di dati?**
  - Varie alternative

## Scrittura nel log e nella base di dati



## Modalità immediata

- 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

- 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

- La scrittura può avvenire in modalità sia immediata che differita
- Richiede sia Undo che Redo

## In pratica

- il record di log viene scritto contemporaneamente in tutte le sue componenti
- Il *commit* si considera effettuato quando il corrispondente record di log è scritto
  - prima di questa scrittura il guasto causa l'*undo* di tutte le operazioni
  - dopo il guasto causa il *redo* di tutte le operazioni

35

## I protocolli per la scrittura nel database

- nel più usato la scrittura del log avviene prima di quella nella base di dati
- la scrittura nella base di dati può avvenire in qualunque momento, anche prima del *commit*

36

## Rollback di una transazione

- Quando una transazione deve essere cancellata, per un errore logico dell'operazione contenuta oppure per un'esigenza di sistema, un record di abort è scritto nel log e tutte le operazioni tra il begin della transazione e l'abort devono essere disfatte.

37

## Guasti

- **Guasti "soft":** errori di programma, crash di sistema, caduta di tensione
  - si perde la memoria centrale e quindi anche il buffer
  - non si perde la memoria secondaria, cioè la base di dati e il log**warm restart, ripresa a caldo**
- **Guasti "hard":** dei dispositivi di memoria secondaria
  - si perde anche la memoria secondaria, i.e. parte della base di dati
  - non si perde la memoria stabile (e quindi il log)**cold restart, ripresa a freddo**
- **La perdita del log è considerato un evento catastrofico e quindi non è definita alcuna strategia di recupero.**

38

## Processo di restart

- Obiettivo: classificare le transazioni in
  - completate (tutti i dati in memoria stabile)
  - Terminate ma con il commit non necessariamente effettuato (vanno rifatte, redo)
  - senza commit (vanno annullate, undo)

39

## Modello fail-stop

1. L'individuazione di un guasto forza l'arresto completo delle transazioni
2. Il sistema operativo viene riavviato
3. Viene avviata una procedura di restart
4. Al termine del restart il buffer è vuoto, ma le transazioni possono ripartire

40

## 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)
  - UNDO riguarda le transazioni attive ma non committed, REDO le transazioni che sono committed prima del guasto
- ripercorrere il log all'indietro, fino alla più vecchia azione delle transazioni in UNDO e REDO, disfaccendo tutte le azioni delle transazioni in UNDO
- ripercorrere il log in avanti, rifacendo tutte le azioni delle transazioni in REDO

41

## Transazioni abortite

- Le operazioni derivanti dal rollback di una transazione possono essere inserite nell'insieme UNDO e fatte al momento del recovery oppure essere fatte al momento dell'abort ed essere inserite
- Nell'insieme di REDO al momento del recovery da un guasto

42

## Ripresa a freddo

- Si ripristina la parte di dati deteriorata a partire dal backup e ci si riporta al record di dump più recente nel log
- Si eseguono le operazioni registrate sul giornale sulla parte deteriorata fino all'istante del guasto
- Si esegue una ripresa a caldo

43

## Il controllore della concorrenza

44

## Controllo di concorrenza

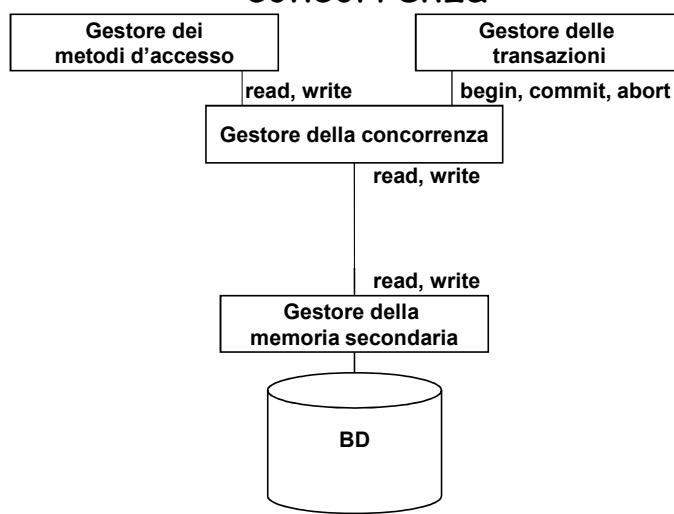
- La concorrenza è fondamentale: decine o centinaia di transazioni al secondo, non possono essere seriali

### Problema

- Anomalie causate dall'esecuzione concorrente, che quindi va governata

45

## Architettura del controllore della concorrenza



46

## 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'esecuzione concorrente:

$t_1$ bot $r_1(x)$	$t_2$ bot $r_2(x)$
$x = x + 1$	$x = x + 1$
$w_1(x)$ commit	$w_2(x)$ commit

- Un aggiornamento viene perso:  $x=3$

47

## Lettura sporca

$t_1$ bot $r_1(x)$ $x = x + 1$ $w_1(x)$	$t_2$ bot $r_2(x)$
abort	commit

- Aspetto critico:  $t_2$  ha letto uno stato intermedio ("sporco") e lo può comunicare all'esterno

48

## Letture inconsistenti

- $t_1$  legge due volte:

$t_1$	$t_2$
bot	
$r_1(x)$	
	bot
	$r_2(x)$
	$x = x + 1$
	$w_2(x)$
	commit
$r_1(x)$	
commit	

- $t_1$  legge due valori diversi per  $x$ !

49

## Aggiornamento fantasma

- Assumere ci sia un vincolo  $y + z = 1000$ ;

$t_1$	$t_2$
bot	
$r_1(y)$	
	bot
	$r_2(y)$
	$y = y - 100$
	$r_2(z)$
	$z = z + 100$
	$w_2(y)$
	$w_2(z)$
	commit
$r_1(z)$	
$s = y + z$	
commit	

- $s = 1100$ :  $t_1$  vede un aggiornamento non completo

50

## Inserimento fantasma

$t_1$

$t_2$

bot

"legge gli stipendi degli impiegati  
del dip A e calcola la media"

bot

"inserisce un  
impiegato in A"

commit

"legge gli stipendi degli impiegati  
del dip A e calcola la media"

commit

51

## 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"

52

## Schedule

- Sequenza di operazioni di input/output di transazioni concorrenti
- Esempio:

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

53

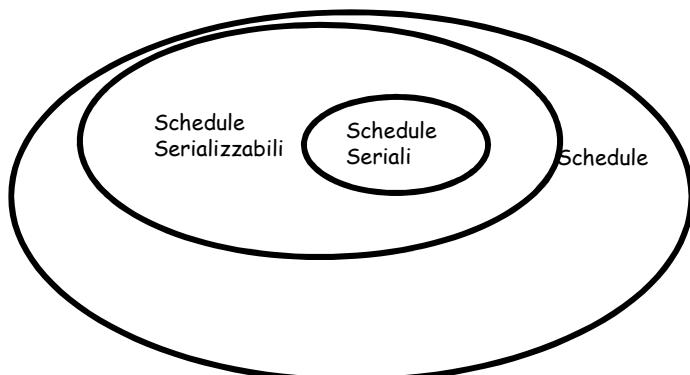
## Controllo di concorrenza

- *Obiettivo:* evitare le anomalie
- *Soluzione:* Scheduler (sistema che accetta o rifiuta, anche tramite riordino, 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 sulle stesse transazioni di uno schedule seriale
  - Richiede una nozione di equivalenza fra schedule

54

## Idea base

- Individuare classi di schedule serializzabili la cui proprietà di serializzabilità sia verificabile a costo basso



55

## View-Serializzabilità

- Schedule **view-equivalenti** ( $S_i \approx_v S_j$ ): hanno la stessa relazione legge-da e le stesse scritture finali su ogni oggetto.
- Uno schedule è **view-serializzabile** se è view-equivalente ad un qualche schedule seriale
- L'insieme degli schedule view-serializzabili è indicato con **VSR**
- Esiste la relazione legge-da tra  $r_i(x)$  e  $w_j(x)$  in  $S$  se  $w_j(x)$  precede  $r_i(x)$  in  $S$  e non c'è nessun  $w_k(x)$  ( $k \neq j$ ) tra di loro.
- $w_i(x)$  in  $S$  è **scrittura finale** se è l'ultima scrittura sull'oggetto  $x$  in  $S$

56

## View serializzabilità: esempi

- $S_1 : w_{01}(x) r_{21}(x) r_{11}(x) w_{22}(x) w_{23}(z)$   
 $S_2 : w_{01}(x) r_{11}(x) r_{21}(x) w_{22}(x) w_{23}(z)$ 
  - $S_1$  è view-equivalente allo schedule seriale  $S_2$  (e quindi è view-serializzabile)
- $S_3 : r_{11}(x) r_{21}(x) w_{12}(x) w_{22}(x)$  (perdita di aggiornamento)  
 $S_4 : r_{11}(x) r_{21}(x) w_{22}(x) r_{12}(x)$  (lettura inconsistenti)  
 $S_5 : r_{11}(x) r_{12}(y) r_{21}(z) r_{22}(y) w_{23}(y) w_{24}(z) r_{13}(z)$   
(aggiornamento fantasma)
  - $S_3, S_4, S_5$  non view-serializzabili, non view-equivalenti a nessun schedule seriale

57

## View serializzabilità: verifica

- Complessità:
  - la verifica della view-equivalenza di due schedule:
    - polinomiale
  - decidere la view-serializzabilità di uno schedule:
    - problema NP-completo
- Non è utilizzabile in pratica

58

## Conflict-serializzabilità

- Definizione preliminare:
  - Un'azione  $a_i$  è in conflitto con  $a_j$  ( $i \neq j$ ), se operano sullo stesso oggetto e almeno una di esse è una scrittura. Due casi:
    - conflitto read-write (rw o wr)
    - 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-serializable* se è conflict-equivalente ad un qualche schedule seriale
- L'insieme degli schedule conflict-serializzabili è indicato con **CSR**

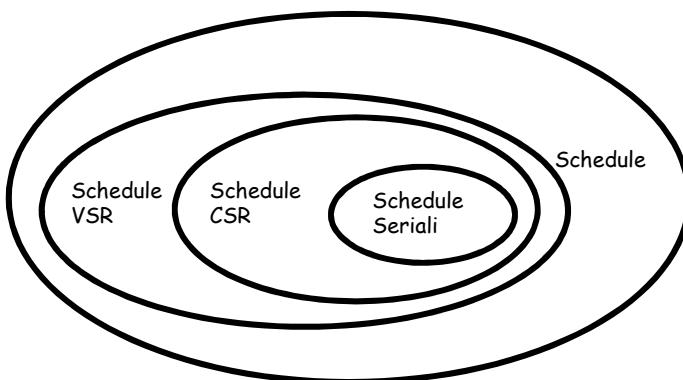
59

## VSR e CSR

- Ogni schedule conflict-serializable è anche view-serializable
  - CSR implica VSR

60

## CSR e VSR



61

## 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

62

## Grafo dei conflitti

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

63

## Conflict-serializzabilità: verifica

- La conflict-serializzabilità è più rapidamente verificabile (l'algoritmo, con opportune strutture dati richiede tempo lineare), ma necessita della costruzione del grafo dei conflitti ad ogni richiesta di scrittura
- Quindi non è praticabile mantenere il grafo, aggiornarlo e verificarne l'aciclicità ad ogni richiesta di operazione

64

## In pratica

- In pratica, si utilizzano tecniche che
  - garantiscono la conflict-serializzabilità senza dover costruire il grafo

65

## 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

66

## Transazioni ben formate rispetto al locking

- Ogni read è preceduta da un *r\_lock* e seguita da un *unlock*
- Ogni write è preceduta da un *w\_lock* e seguita da un *unlock*

67

## Gestione dei lock

- Basata sulla tavola dei conflitti (politica per la gestione dei conflitti)

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

(1) Un contatore tiene conto del numero di "lettori"; la risorsa è rilasciata solo quando il contatore scende a zero

68

## Gestione dei lock (cont)

- 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 gestore della concorrenza (o lock manager o scheduler) gestisce una tabella dei lock (lock già concessi), per ricordare la situazione

69

## Locking a due fasi

- Usato da quasi tutti i sistemi
- Garantisce "a priori" la conflict-serializzabilità
- 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 finché tutti quelli che ha acquisito non sono stati rilasciati

70

- Una transazione attraversa prima la fase di acquisizione di ciò che le serve
- Poi comincia a rilasciare

71

## Two phase locking

- |  |  |
|--|--|
| <ul style="list-style-type: none"> <li>• begin (T1)</li> <li>• w1_lock(B);</li> <li>• r1(B);</li> <li>• B:=B-50;</li> <li>• w1(B);</li> <li>• w1_lock(A);</li> <li>• r1(A);</li> <li>• A:=A+50;</li> <li>• w1(A);</li> <li>• unlock(B);</li> <li>• unlock(A);</li> <li>• commit</li> </ul> | <ul style="list-style-type: none"> <li>• begin (T1)</li> <li>• w1_lock(B);</li> <li>• r1(B);</li> <li>• B:=B-50;</li> <li>• w1(B);</li> <li>• w1_lock(A);</li> <li>• unlock(B);</li> <li>• r1(A);</li> <li>• A:=A+50;</li> <li>• w1(A);</li> <li>• unlock(A);</li> <li>• commit</li> </ul> |
|--|--|

72

## Upgrading e downgrading dei lock

- Per aumentare la concorrenza è possibile avere lock di tipo diverso, condiviso o esclusivo, usati in momenti diversi sulla stessa risorsa.
- Per una lettura si richiede un lock condiviso (contatore delle letture), quando serve scrivere si richiede il lock esclusivo.
- L'upgrade si può fare solo nella fase di acquisizione dei lock, il downgrade nella fase di rilascio.

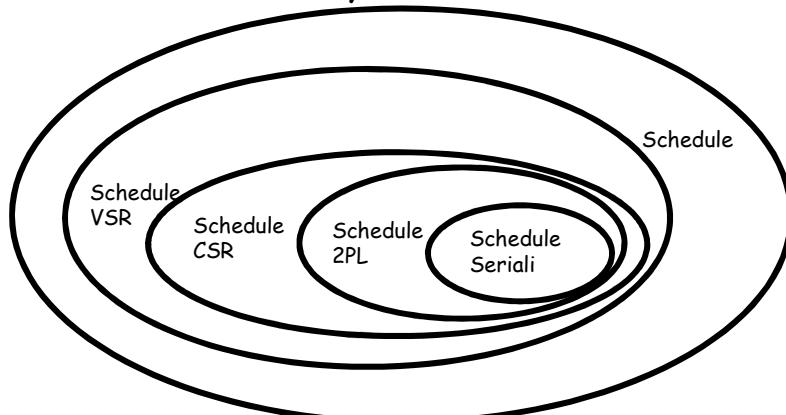
73

## 2PL e CSR

- Ogni schedule 2PL è anche conflict serializzabile, ma non è vero il viceversa
  - 2PL implica CSR

74

## CSR, VSR e 2PL



75

## 2PL presenta alcune anomalie

- Attese incrociate (o deadlock): due transazioni detengono ciascuna una risorsa e aspettano la risorsa detenuta dall'altra
- Il fallimento di una transazione che ha scritto una risorsa causa il fallimento delle transazioni che hanno letto il valore scritto

76

## Possibili anomalie: deadlock

- begin(T1)
  - w1\_lock(B);
  - r1(B);
  - B:=B-50;
  - w1(B);
- begin(T2)  
r2\_lock(A);  
r2(A);  
r2\_lock(B);  
wait T1
- w1\_lock(A);
  - wait T2
  - r1(B);
  - unlock (B)

77

## Possibili anomalie: cascading rollbacks

- begin(T1);
  - w1\_lock(A);
  - r1(A);
  - r1\_lock(B);
  - r1(B);
  - w1(A);
  - unlock(A);
  - abort
- begin(T2);  
w2\_lock(A);  
r2(A);  
w2(A);  
unlock(A);...
- begin(T3);  
r3\_lock(A);  
r3(A);...

Quando T1 fallisce il fallimento si trasmette a T2 e T3

78

## Locking a due fasi stretto

- Condizione aggiuntiva:
  - I lock possono essere rilasciati solo dopo il commit
- elimina il rischio di letture sporche e quindi di rollback in cascata

79

## Controllo di concorrenza basato su timestamp

- Tecnica alternativa al 2pL
- **Timestamp:**
  - identificatore che definisce un ordinamento totale sugli eventi di un sistema
- Ogni transazione ha un timestamp che rappresenta l'istante di inizio della transazione
- Uno schedule è accettato solo se riflette l'ordinamento seriale delle transazioni indotto dai timestamp

80

## Dettagli

- Lo scheduler ha due contatori  $RTM(x)$  e  $WTM(x)$  per ogni oggetto
- Lo scheduler riceve richieste di letture e scritture (con indicato il timestamp della transazione):
  - $read(x, ts)$ :
    - se  $ts < WTM(x)$  allora la richiesta è respinta e la transazione viene uccisa;
    - altrimenti, la richiesta viene accolta e  $RTM(x)$  è posto uguale al maggiore fra  $RTM(x)$  e  $ts$
  - $write(x, ts)$ :
    - se  $ts < WTM(x)$  o  $ts < RTM(x)$  allora la richiesta è respinta e la transazione viene uccisa,
    - altrimenti, la richiesta viene accolta e  $WTM(x)$  è posto uguale a  $ts$
- Vengono uccise molte transazioni

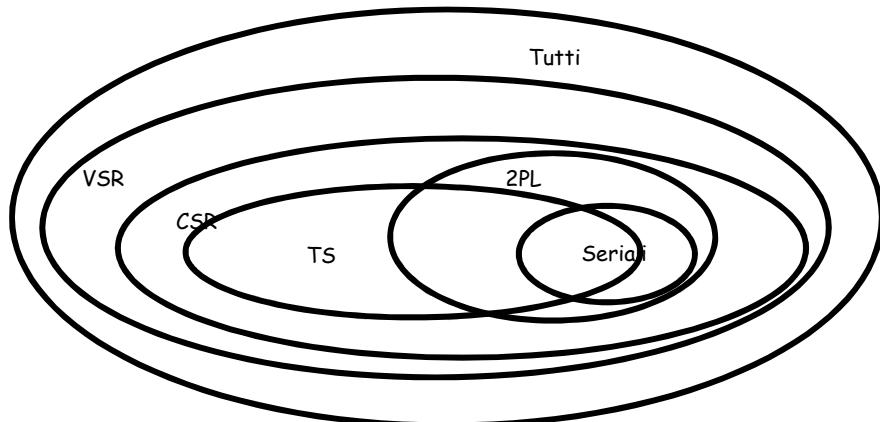
81

## 2PL vs TS

- Gli schedule TS sono automaticamente CSR: corrispondono ad un'esecuzione seriale
- Ma 2PL e TS sono incomparabili

82

## CSR, VSR, 2PL e TS



83

## Attenzione

- L'ordine seriale delle transazioni è fissato prima che le operazioni vengano richieste, tutti gli altri ordinamenti non sono accettati.
- Quando T1 comincia prima di T2, potrebbe essere abilitato uno schedule 2PL o CSR equivalente ad uno seriale T2 T1; col TS non è possibile, al limite T1 viene uccisa e poi fatta ripartire dopo T2.

84

## 2PL vs TS

- In 2PL le transazioni sono poste in attesa quando non è possibile acquisire un lock, in TS uccise e rilanciate
  - Le ripartenze sono di solito più costose delle attese:
  - conviene il 2PL
- 2PL può causare deadlock, TS no

85

## Risoluzione dello stallo

- Uno stallo corrisponde ad un ciclo nel grafo delle attese
- Tre tecniche di risoluzione
  1. Timeout (problema: scelta dell'intervallo, con trade-off)
  2. Rilevamento dello stallo
    - ricerca di cicli nel grafo delle attese
  3. Prevenzione dello stallo
    - Prevenzione: uccisione di transazioni "sospette"

86

## Livelli di isolamento in SQL:1999 (e JDBC)

- Le transazioni possono essere definite **read-only** (non possono richiedere lock esclusivi)
- 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
  - **serializable** evita tutte le anomalie
- Nota:
  - la perdita di aggiornamento è sempre evitata

87

## 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 anche in lettura
- **serializable**:
  - 2PL

88

## Esempio

- Dire se i seguenti due schedule sono view-equivalenti o conflict-equivalenti o nessuna delle due cose.

•

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

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

•