



DMIF, Università di Udine

Tecnologie Digitali per il Cibo e la Ristorazione

Basi di Dati Relazionali - Fondamenti

Andrea Brunello

andrea.brunello@uniud.it

A.A. 2021-2022



Database Management System (DBMS)

- Un DBMS è un sistema contenente informazioni riguardanti uno specifico dominio:
 - Collezione di dati correlati (database, base di dati)
 - Insieme di programmi per accedere ai dati
 - Definizione delle strutture per memorizzare l'informazione
 - Manipolazione dell'informazione memorizzata
- Applicazioni dei DBMS:
 - Ambito bancario: trasferimento di fondi, gestione di conti correnti, azioni, ...
 - Compagnie aeree: prenotazione di voli, gestione degli orari
 - Università: registrazione degli studenti, gestione degli esami, gestione del corpo docente
 - Vendite online: tracciamento ordini, suggerimenti
- Le basi di dati possono essere molto grandi
- Toccano ogni aspetto della nostra vita, anche se le interfacce utente spesso ne nascondono l'esistenza



Esempio di una base di dati universitaria

- Applicazioni per le quali possono essere impiegati i DBMS (es. in ambito universitario):
 - Aggiunta/rimozione di studenti/professori/corsi
 - Registrazione degli studenti ai corsi
 - Generazione degli elenchi degli studenti
 - Registrazione degli esiti degli esami
 - Calcolo di statistiche (media dei voti)
 - Generazione del libretto
- In origine, tutte queste funzioni erano implementate come programmi e file ad-hoc salvati su disco
- Primi database relazionali impiegati commercialmente negli anni '80, dopo 10 anni di sviluppo



- Ridondanza ed inconsistenza dei dati
 - Formati multipli, replicazione dell'informazione
- Difficoltà nell'accesso ai dati
 - Nuovo programma per ogni nuova operazione
 - Programma per registrare un nuovo studente, programma per calcolare il numero di studenti iscritti, ...
- Problemi di integrità:
 - Vincoli di dominio “seppelliti” nel codice dei programmi, invece di essere chiaramente descritti (e.g., la lode può essere assegnata solo se il punteggio è 30)
 - Difficile elencare, modificare, aggiungere, rimuovere vincoli



Svantaggi di salvare l'informazione su disco

- Atomicità degli aggiornamenti
 - Fallimenti del sistema potrebbero lasciare delle operazioni incomplete
 - Ad esempio, il trasferimento di fondi fra due c/c dovrebbe compiersi nella sua interezza o non avvenire affatto
- Accesso concorrente da parte di più utenti
 - Requisito fondamentale per ottenere delle prestazioni accettabili (es., prenotazione biglietti in una compagnia aerea)
 - Accesso concorrente non correttamente gestito può generare inconsistenze
- Problemi di sicurezza
 - Difficile consentire ad utenti diversi l'accesso a diversi insiemi di dati, e la possibilità di eseguire solo certi tipi di operazioni

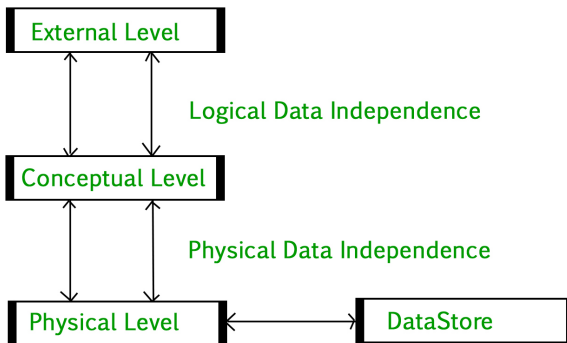


Scenari che giustificano l'uso di un DBMS

- Un DBMS consente di risolvere i problemi elencati
- Esso risulta quindi utile nei casi in cui si ha abbiano una o più delle casistiche:
 - **Grandi quantità di dati:** non possono essere memorizzate interamente nella memoria principale
 - **Dati globali:** i dati sono d'interesse per un ampio insieme di utenti e applicazioni. Non sono legati ad uno specifico utente o programma
 - **Dati persistenti:** La base di dati e l'informazione contenuta esistono indipendentemente dall'utente o dall'applicazione, a differenza es. dei dati manipolati all'interno di un programma, che terminano la loro esistenza al termine dell'esecuzione del programma



- Un concetto fondamentale nell'informatica e nelle basi di dati è quello di **astrazione**
- L'astrazione consente di tenere sotto controllo la complessità, permettendo all'utente di focalizzarsi di volta in volta su ciò che è importante, tralasciando dettagli irrilevanti che genererebbero confusione
- Nella pratica, viene realizzata “stratificando” il DBMS
 - Livello più basso (**fisico**) che gestisce la memorizzazione dell'informazione su disco
 - Livello logico che descrive come i dati sono organizzati nel database
 - Livello più alto (**delle viste**) che gestisce la presentazione dell'informazione agli utenti e ai programmi



- Indipendenza fisica dei dati: possibilità di apportare modifiche al livello fisico senza dover modificare il livello logico (conceptual level)



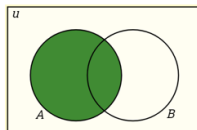
- In matematica, una collezione di elementi rappresenta un insieme se esiste un criterio oggettivo che permette di decidere univocamente se un qualunque elemento fa parte o no del raggruppamento
- Ad esempio: l'insieme dei numeri naturali, l'insieme dei caratteri ASCII, l'insieme dei nomi delle capitali
- Gli oggetti che compongono un insieme vengono detti elementi di tale insieme
- Notazione: A insieme, a elemento dell'insieme A .
Indichiamo l'appartenenza con $a \in A$
 - Esempio: $A = \{7, 1, 3, 2, 5\}$, allora $1 \in A$



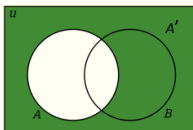
Il concetto di insieme è caratterizzato dalle seguenti proprietà:

- un elemento può appartenere o non appartenere ad un determinato insieme, non ci sono vie di mezzo
- un elemento non può comparire più di una volta in un insieme
- gli elementi di un insieme non hanno un ordine in cui essi compaiono
- gli elementi di un insieme lo caratterizzano univocamente: due insiemi coincidono se e solo se hanno gli stessi elementi

Set Operations and Venn Diagrams



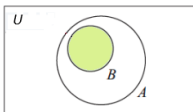
Set A



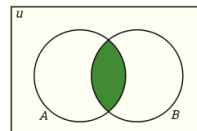
A' the complement of A



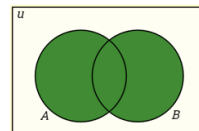
A and B are disjoint sets



B is proper subset of A
 $B \subset A$



Both A and B
A intersect B
 $A \cap B$



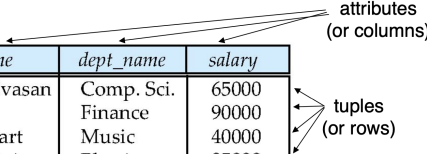
Either A or B
A union B
 $A \cup B$



- **Multinsieme:** generalizzazione del concetto di insieme in cui sono ammessi gli elementi ripetuti
 - Ad esempio: $A = \{1, 1, 3, 3, 2, 3\}$
- **Prodotto cartesiano:** dati due insiemi A e B , il prodotto cartesiano di A e B , indicato con $A \times B$, è l'insieme delle coppie ordinate (a, b) dove $a \in A$ e $b \in B$
 - Ad esempio: $A = \{1, 2, 3\}$, $B = \{red, blue\}$, allora
 $A \times B = \{(1, red), (1, blue), (2, red), (2, blue), (3, red), (3, blue)\}$

- Un modello dei dati è una collezione di strumenti che consentono di descrivere
 - I dati
 - Le relazioni fra i dati
 - La semantica dei dati
 - I vincoli che sussistono sui dati
- Ci focalizzeremo sul **modello relazionale** (Codd, 1969), in quanto è ad oggi quello più diffuso grazie alla sua efficacia e semplicità
- Particolarmente adatto al trattamento di dati strutturati

- Nel modello relazionale, i dati vengono rappresentati mediante tabelle (relazioni) identificate da un nome
- Tutte le righe (tuple) appartenenti ad una stessa tabella sono caratterizzate dagli stessi campi (record-based model)
- Layout “rigido”, dati strutturati



ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

- Attraverso gli attributi è possibile specificare dei legami fra le tabelle, relazioni fra i dati

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table



- L'insieme dei valori ammissibili per un dato attributo è detto **dominio** dell'attributo
- Ad esempio: stringa, booleano, numero intero, numero decimale, data, numero fra 1 e 30
- Un DBMS mette a disposizione un insieme più o meno vasto di tipi; è generalmente possibile estendere tale insieme con tipi personalizzati
- I tipi devono essere atomici (indivisibili): ad esempio, non si può avere una stringa *indirizzo* che a sua volta si compone delle sottostringhe *via*, *numero civico*, *città*

- Il valore nullo (**null**) è un particolare valore facente parte di ogni dominio
- Null rappresenta un valore che può essere:
 - *sconosciuto*: numero di telefono mancante relativo ad un determinato cliente
 - *non ancora noto*: un professore è stato appena assunto, e deve ancora essere assegnato ad un dipartimento
 - *non applicabile*: consumo di carburante in Km/l di un'auto elettrica
- Come vedremo descrivendo il linguaggio SQL, il trattamento dei valori nulli richiede particolare accortezza

- Siano A_1, A_2, \dots, A_n attributi
- $R(A_1, A_2, \dots, A_n)$ è uno **schema di relazione**
 - Lo “scheletro” di una tabella
 - Esempio: *instructor*(*ID*, *name*, *dept_name*, *salary*)
- Formalmente, dati D_1, D_2, \dots, D_n i domini associati agli attributi A_1, A_2, \dots, A_n , una (istanza di) **relazione** r è un sottoinsieme di $D_1 \times D_2 \times \dots \times D_n$
- Nel nostro caso, una relazione è un insieme finito di tuple (a_1, a_2, \dots, a_n) , dove ciascun $a_i \in D_i$



- Al fine di garantire la bontà dell'informazione memorizzata nella base di dati, è possibile (e auspicabile) definire dei vincoli di integrità, che limitano i dati che è possibile memorizzare nelle tabelle
- Vincoli di integrità intra-relazionali:
 - Vincolo not-null
 - Vincolo di univocità (unique)
 - Chiave primaria (primary key)
- Vincoli di integrità inter-relazionali:
 - Chiave esterna (foreign key)

- Posto su una colonna, impedisce la presenza di valori nulli
- Nella seguente tabella, sono consentiti i valori nulli su *state*
- Se si desidera che ad ogni cliente sia sempre associato un valore di *state*, è possibile imporre un vincolo not-null
- Non sarà più possibile poi inserire un cliente per cui il valore di *state* non è noto

	customername	state	country
▶	Australian Collectors, Co.	Victoria	Australia
	Anna's Decorations, Ltd	NSW	Australia
	Souvenirs And Things Co.	NSW	Australia
	Australian Gift Network, Co	Queensland	Australia
	Australian Collectables, Ltd	Victoria	Australia
	Salzburg Collectables	NULL	Austria
	Mini Auto Werke	NULL	Austria
	Petit Auto	NULL	Belgium

- Viene specificato su una o più colonne
- Impone che i valori memorizzati in una colonna o in un gruppo di colonne siano univoci tra le righe di una tabella
- Ad esempio, in una tabella che memorizza alberghi, non vogliamo avere più alberghi con lo stesso nome in una stessa città
- Poniamo allora un vincolo unique definito sulle colonne *Nome albergo* e *Città* della seguente relazione

Partiva IVA	Nome albergo	Città	Stelle	Numero camere
1053362646	Ritz	Roma	5	205
1053362600	Ritz	Milano	4	215
1233362688	Da Mimmo	Napoli	3	80
1053369902	Friuli	Udine	3	50
1325239931	Friuli	Udine	2	45

Figure: Relazione *Alberghi*



- Alcuni attributi giocano un ruolo fondamentale
- Conoscendo il loro valore, è possibile identificare univocamente una tupla all'interno della tabella
- Ad esempio:
 - Il codice fiscale in una tabella che tiene traccia dei dipendenti di un'azienda
 - Il nome dell'albergo e la città nell'esempio della slide precedente
 - Sempre nella slide precedente, la Partita IVA



- Sia K un sottoinsieme degli attributi di $R(A_1, \dots, A_n)$
- K è una **superchiave** di R se i valori di K sono sufficienti per identificare una tupla in ogni possibile istanza di relazione di R
 - *(Nome albergo, Città, Stelle)* nella relazione *Alberghi*
- Una superchiave K è detta **chiave candidata** se K è minimale
 - *(Nome albergo, Città)* nella relazione *Alberghi*, o *(Partita IVA)* nella medesima relazione
- Una delle chiavi candidate viene scelta come **chiave primaria**
 - Quale? Tipicamente quella con meno attributi



- La chiave primaria gioca un ruolo fondamentale nelle relazioni, in quanto se ben definita impedisce la presenza di dati duplicati e potenzialmente inconsistenti
- Il vincolo di chiave primaria presuppone in modo naturale la presenza di due ulteriori vincoli:
 - **Integrità dell'entità:** gli attributi facenti parte della chiave primaria non possono essere nulli
 - **Unique:** in quanto non si possono avere due righe i cui valori coincidono sugli attributi della chiave primaria



- Quali sono le chiavi candidate più naturali della relazione *Libro*(ISBN, autore, anno, editore, genere, numero pagine)?
 - E assumendo che un autore possa scrivere solamente un libro in un dato anno?
 - E assumendo che un autore possa scrivere solamente un libro in un dato anno per un certo editore?
- Quali sono le chiavi candidate più naturali della relazione *Studente*(codice fiscale, matricola, nome, cognome, indirizzo)?

- La chiave esterna consente di legare i valori che compaiono in due diverse tabelle
- I valori che appaiono in una tabella (**referenziante**) devono comparire anche in un'altra tabella (**referenziata**)

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table



- La chiave esterna può essere definita su uno o più attributi
- Sugli attributi della tabella referenziata deve sussistere un vincolo di univocità
- Cosa fare quando uno o più valori referenziati vengono cancellati da una tabella?
 - **Restrict:** impedisce la cancellazione
 - **Cascade:** cancella a cascata tutte le tuple che facevano riferimento a tali valori
 - Altro, ad esempio imposta un valore di default alle tuple che facevano riferimento a tali valori