Capitolo 20

I Sistemi Informativi

Il Sistema Informativo

- È il sistema che gestisce le informazioni che, attualmente, vengono codificate e rappresentate sotto forma di dati, memorizzati su appositi supporti, a cui si può accedere e che possono essere interpretati per fornire informazioni
  - informazione: elemento di conoscenza di fatti e situazioni
  - dato: codifica di elementi di informazione sotto forma di simboli, che devono essere elaborati per acquisire un significato

Base di Dati

- È una collezione di dati usata per rappresentare informazioni interessanti per un sistema informativo
- I dati hanno caratteristiche più stabili rispetto alle procedure con cui si accede ad essi e con cui vengono manipolati
- Esempio: dati bancari
  - i dati hanno sempre la stessa struttura, mentre variano spesso le procedure usate per reperirli e modificarli
  - le nuove procedure ereditano i vecchi dati, con apposite trasformazioni

Sistemi di Gestione di Basi di Dati (i)

- Il software dedicato alla gestione dei dati ha iniziato ad essere prodotto dai primi anni ‘70 e in modo più organizzato dai primi anni ‘80
- In assenza di software specifico, per la gestione dei dati sono stati e sono usati linguaggi di programmazione tradizionali (come Java) o dedicati (come il Cobol)

Sistemi di Gestione di Basi di Dati (ii)

- Il primo approccio alla gestione dei dati è quello dell'uso di file, per mantenere i dati memorizzati in modo persistente in memoria di massa
  - memorizzando i dati su file si possono utilizzare solo semplici meccanismi di accesso e di condivisione
  - le procedure scritte in un linguaggio di programmazione utilizzano uno o più file privati
  - i dati che sono utili a diverse applicazioni sono replicati tante volte quante sono le applicazioni che li utilizzano, con problemi di ridondanza e di incoerenza
- Una base di dati è una collezione di dati gestita da un Sistema di Gestione di Basi di Dati
- Anch'essa è realizzata su file (come i file system), ma tali file hanno organizzazioni dei dati più sofisticate

Data Base Management Systems (DBMS)

- Sono sistemi software che sono in grado di gestire:
  - grandi collezioni di dati
  - in modo condiviso
  - e persistente
  - garantendo affidabilità
  - e privatezza
  - con efficienza
  - ed efficacia

DBMS (i)

- grandi collezioni di dati
  - i dati possono avere dimensioni enormi, e comunque molto maggiori della memoria centrale disponibile, quindi vanno gestiti in memoria secondaria
- in modo condiviso
  - più applicazioni e più utenti devono poter accedere a dati comuni, riducendo così la ridondanza dei dati e la possibilità di inconsistenza
  - per permettere l'accesso contemporaneo (condiviso) ai dati, il DBMS è dotato del meccanismo di controllo della concorrenza

DBMS (ii)

- garantendo affidabilità
  - il DBMS garantisce il contenuto della base di dati anche in caso di malfunzionamento hardware e software
  - possiede moduli di salvataggio e ripristino (backup e recovery)
- e privatezza
  - ciascun utente è riconosciuto e viene abilitato a svolgere solo le attività che gli competono, con meccanismi di autorizzazione

DBMS (iii)

- con efficienza
  - il DBMS deve svolgere le operazioni utilizzando risorse in modo accettabile dagli utenti (tempo e spazio)
  - questo fattore dipende dalle tecniche usate per implementare il DBMS e per progettare e realizzare la base di dati
  - tutte le funzionalità richiedono un sistema dimensionato in modo sufficientemente ampio
- con efficacia
  - il DBMS deve avere una efficacia complessiva rispetto agli utenti che ne fanno uso, deve cioè essere produttivo

Modelli di Dati

- Un modello di dati è un modello di organizzazione dei dati
- Fornisce un modo per descrivere la struttura dei dati e permettere l'accesso ai dati da parte dell'elaboratore
- Ogni modello di dati mette a disposizione dei meccanismi di strutturazione che permettono di definire nuovi tipi
- Esempio:
  - Java è dotato di meccanismi che permettono di definire record (class) e array([])

Modelli Logici

- Il modello logico indica che il modello dei dati astrae dai dettagli implementativi, ma rispecchia particolari strutture di organizzazione dei dati (es., ad alberi, a grafi, a tabelle)
- Esistono quattro modelli logici di dati:
  - il modello gerarchico
  - il modello reticolare
  - il modello relazionale
  - il modello a oggetti

I 4 Modelli Logici

- modello gerarchico
  - basato su gerarchie ad albero, è stato il modello usato nei primi DBMS (fine anni sessanta)
- modello reticolare
  - detto anche codasyl, dal comitato di standardizzazione, basato su grafi (inizio anni settanta)
- modello relazionale
  - permette di definire tipi col costruttore relazione, che consente di organizzare i dati in insiemi di record di struttura fissa (fine anni settanta)
- modello a oggetti
  - sviluppato come alternativa al modello relazionale, basato sul paradigma della programmazione ad oggetti (anni ottanta)

Il Modello Relazionale

- È il modello di maggiore successo
- Permette di organizzare i dati secondo relazioni (tabelle) la cui teoria si basa sugli insiemi
- Le corrispondenze fra i dati di relazioni diverse sono rappresentate per mezzo dei dati stessi
- Fornisce indipendenza dei dati a livello fisico da quelli a livello logico
  - gli utenti e i programmatori che sviluppano le applicazioni fanno riferimento esclusivamente al livello logico
  - I dati descritti a livello logico sono realizzati mediante strutture fisiche opportune, che non è necessario conoscere per accedere ai dati
  - Ha sostituito i modelli gerarchico e reticolare, che si basavano sulla sottostante struttura realizzativa con dettagli di implementazione che richiedevano l'uso di puntatori e di ordinamento dei dati

Relazioni e Tabelle (i)

- Dati due insiemi D_1 e D_2 si dice prodotto cartesiano di D_1 e D_2 e si indica con D_1xD_2) l'insieme di tutte le coppie (v_1,v_2) tali che v_1 appartiene a D_1 e v_2 appartiene a D_2
- Una relazione matematica sui due insiemi D_1 e D_2 è un sottoinsieme del prodotto cartesiano D_1xD_2
- D_1 e D_2 si dicono domini della relazione
- I domini possono essere interi, stringhe o altro

Relazioni e Tabelle (ii)

- Graficamente le relazioni si rappresentano sotto forma di tabelle
- Generalizzando rispetto al numero di insiemi il prodotto cartesiano di D_1, D_2 , …, D_n si indica con D_1xD_2x…xD_n e una relazione matematica su tali domini è un sottoinsieme del prodotto cartesiano
- Il numero di domini si dice grado del prodotto cartesiano e della relazione
- Il numero di *n*-ple della relazione è chiamato cardinalità

Schemi e istanze

- Lo schema della base di dati, che descrive le caratteristiche dei dati, può essere considerato invariante
- L'istanza invece rappresenta i dati effettivamente presenti nella base di dati
- Esempio: se creiamo la relazione Docenza**:**

DOCENZA (Corso, NomeDocente)

  - Lo **schema** della relazione **(componente intensionale)** è costituito dalla sua intestazione, ovvero il nome della relazione, e dai suoi attributi
  - L'**istanza** è l'insieme delle righe che compongono la relazione **(componente estensionale)** e può variare nel tempo, quando i dati vengono modificati

Livelli di Astrazione degli Schemi

- La standardizzazione dell'architettura dei DBMS comprende tre livelli:
  - schema logico
    - è una descrizione della base di dati usando il modello logico adottato dal DBMS
  - schema interno
    - rappresentazione dello schema logico su strutture fisiche di memorizzazione (file sequenziale, file hash, file sequenziale con indici)
  - schema esterno
    - è la descrizione di una porzione di interesse della base di dati, come vista di uno o più utenti
    - più schemi esterni possono essere associati ad uno stesso schema logico
    - è possibile definire delle viste, ovvero relazioni derivate da quelle presenti nel DB

Indipendenza dei Dati

- L'architettura a tre livelli garantisce nei DBMS l'indipendenza dei dati, cioè permette agli utenti di interagire con il sistema senza doversi occupare degli aspetti realizzativi usati nella costruzione della base di dati
  - indipendenza fisica
    - si possono modificare le strutture fisiche come l'organizzazione dei file o la loro allocazione fisica sui dispositivi di memorizzazione, senza che ciò richieda la modifica della descrizione dei dati ad alto livello o la modifica dei programmi applicativi
  - indipendenza logica
    - si possono aggiungere schemi esterni senza dover modificare lo schema logico e la sottostante organizzazione fisica dei dati

Linguaggi

- **Data Definition Language** (DDL)
  - è il linguaggio di definizione dei dati che serve per definire gli schemi esterni, gli schemi logici e quelli fisici oltre alle autorizzazioni per l'accesso
- **Data Manipulation Language** (DML)
  - linguaggio di manipolazione dei dati che serve per scrivere le interrogazioni e gli aggiornamenti delle istanze della base di dati
- Alcuni linguaggi, come **SQL**, includono sia DDL che DML

Accesso ai Dati

- L’accesso alla base di dati può avvenire:
  - tramite linguaggi testuali interattivi come SQL
  - tramite comandi interattivi immersi in programmi scritti in linguaggi di interrogazione tradizionali, come Java (linguaggi ospite)
  - con comandi di tipo interattivo all’interno di linguaggi di programmazione ad hoc
  - tramite interfacce grafiche che permettono di formulare interrogazioni senza usare un linguaggio testuale

Utenti

- L’amministratore della base di dati
  - il DataBase Administrator (DBA) è il responsabile della progettazione, controllo e amministrazione della base di dati e garantisce un controllo centralizzato dei dati, tra cui la gestione delle autorizzazioni
- I progettisti e i programmatori di applicazioni
  - realizzano i programmi di accesso alla base di dati con il DML
- Gli utenti
  - utilizzano la base di dati per le proprie attività:
    - utenti **finali** che utilizzano transazioni predefinite
    - utenti **casuali** in grado di formulare nuove interrogazioni e nuovi aggiornamenti alla base di dati

Transazione

- Unità elementare di operazioni con caratteristiche di **correttezza**, **robustezza** (ai guasti) e **isolamento**
- La transazione deve essere incapsulata all'interno di due comandi:
  - **begin transaction** (bot)
  - **end transaction** (eot)
- All'interno della transazione possono comparire due istruzioni:
  - **commit** (fa andare a buon fine la transazione)
  - **rollback** (annulla gli effetti del lavoro già svolto)

Esempio di Transazione

- Trasferimento da un conto bancario ad un altro

  begin transaction

  x=x-10;

  y=y+10;

  commit

  end transaction

  - La transazione si dice ben formata quando inizia con **begin transaction**, al suo interno viene eseguita una sola tra le istruzioni di **commit** e di **rollback**, non vengono eseguite operazioni di accesso o di modifica della base di dati dopo l'esecuzione di **commit** o di **rollback** e termina con **end transaction**
  - Alcune volte **end transaction** viene forzato direttamente dal sistema dopo l'esecuzione di un **commit** o di un **rollback**

Proprietà ACID delle transazioni

- **A**tomicity (atomicità)
- **C**onsistency (consistenza)
- **I**solation (isolamento)
- **D**urability (persistenza)

Atomicità

- Una transazione è un'**unità atomica indivisibile** e non può lasciare la base di dati in uno stato intermedio
  - o vengono portate a compimento con successo tutte le operazioni tra **begin transaction** ed **end transaction**
  - oppure, se si verifica un errore, il sistema deve essere in grado di disfare tutte le operazioni compiute dalla transazione fino a quell'istante
    - quando viene eseguito un **rollback** il sistema deve riportare alla situazione precedente l'inizio della transazione con un **undo**
    - quando la transazione ha eseguito il **commit**, il sistema deve garantire che le operazioni compiute dalla transazione siano effettivamente eseguite, e ciò può richiedere di rifare le operazioni svolte con un **redo**

Consistenza

- Una transazione non deve violare vincoli di integrità della base di dati
  - ad esempio: non possono esistere due studenti con lo stesso numero di matricola
- I vincoli possono essere controllati subito o al momento del **commit**
- Particolare attenzione va fatta con operazioni che possono (temporaneamente) violare i vincoli di integrità
  - ad esempio: trasferimento di denaro tra due conti di una stessa banca

Isolamento

- L’esecuzione di una transazione deve essere indipendente dalla contemporanea esecuzione di altre transazioni
- Inoltre, il risultato concorrente di più transazioni deve essere lo stesso di quello che si otterrebbe eseguendole ciascuna da sola
- Ogni transazione quindi deve essere indipendente dalle altre
  - altrimenti potrebbe succedere che l’effetto di un **rollback** di una transazione provochi il **rollback** a catena di altre transazioni

Persistenza

- Richiede che non venga perso l'effetto di una transazione che ha effettuato il commit
- Quindi tutte le operazioni effettuate devono essere rese permanenti con accessi a disco, giungendo in questo modo al vero e proprio **end of transaction**
- Il termine delle transazione, in realtà, può risultare differito a seconda della politica di gestione della memoria centrale (al fine di minimizzare gli accessi al disco ed aumentare l'efficienza)