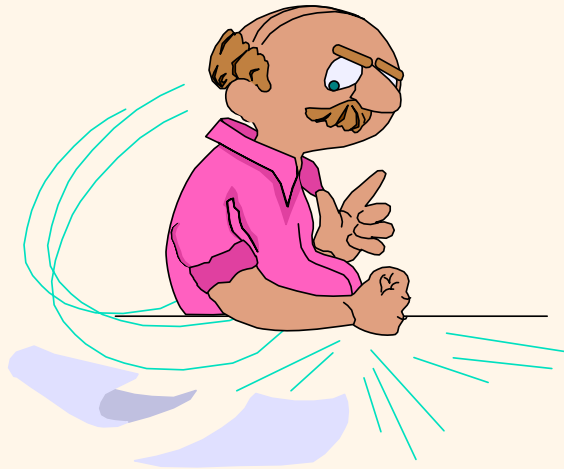
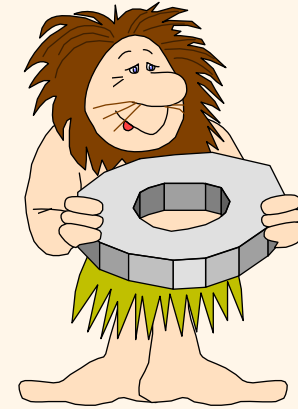


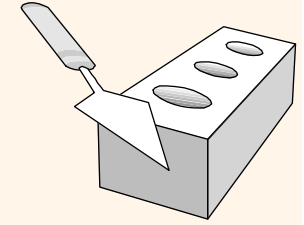
Sistemi di gestione delle basi di dati



Cos'è un DBMS?



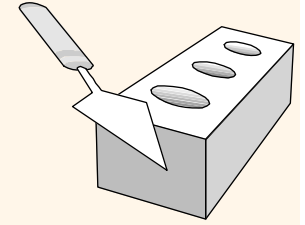
- ❖ Una collezione integrata molto grande di dati
- ❖ Modella organizzazioni del mondo reale
 - Entità (ad esempio studenti, corsi)
 - Relazioni (ad esempio, Madonna segue il corso CS564)
- ❖ Un Database Management System (DBMS) è un pacchetto software progettato per memorizzare e gestire basi di dati



File verso DBMS

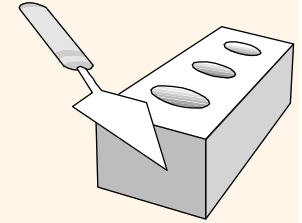
- ❖ File : collezione di dati che risiede su un dispositivo di memoria esterna
 - Strutturato in accordo ai requisiti di una applicazione
- ❖ Applicazioni costruite su **file system** basate sui metodi di accesso messi a disposizione dal sistema operativo e sulle organizzazioni di file disponibili nel linguaggio di programmazione

File verso DBMS



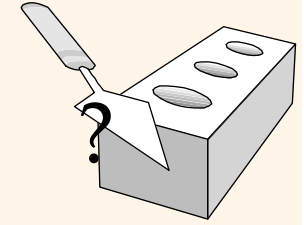
- ❖ Necessità di spostare grandi insiemi di dati tra memoria principale e memoria secondaria (ad esempio in caso di buffering, di accessi orientati alla pagina, di indirizzamento a 32 bit, etc.)
- ❖ Codifica speciale per interrogazioni diverse
- ❖ Necessità di proteggere i dati da inconsistenza dovuta a utenti multipli che accedono i dati simultaneamente in maniera concorrente
- ❖ Ripristino da *crash*
- ❖ Sicurezza e controllo degli accessi

Perché usare un DBMS?



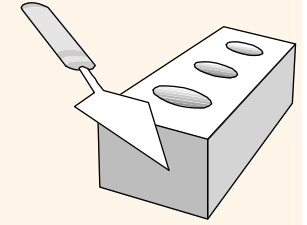
- ❖ Indipendenza dei dati e accesso efficiente
- ❖ Tempo ridotto di sviluppo dell'applicazione
- ❖ Integrità dei dati e sicurezza
- ❖ Amministrazione dei dati uniforme
- ❖ Accesso concorrente, ripristino da *crash*

Perché studiare le basi di dati??



- ❖ Spostamento dalla computazione all'informazione
 - A "livello base": gestire i dati caotici del Web
 - Ad "alto livello": applicazioni scientifiche
- ❖ Gli insiemi di dati aumentano in varietà e volume
 - Librerie digitali, video interattivi, progetto Genoma Umano, progetto EOS
- ❖ ... necessità di crescita esponenziale dei DBMS
- ❖ I DBMS coprono gran parte dell'informatica
- ❖ Sistemi operativi, linguaggi, teoria, Intelligenza Artificiale, multimedialità, logica

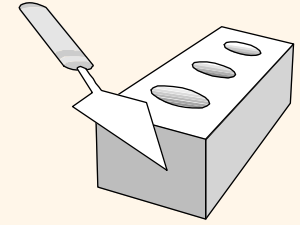




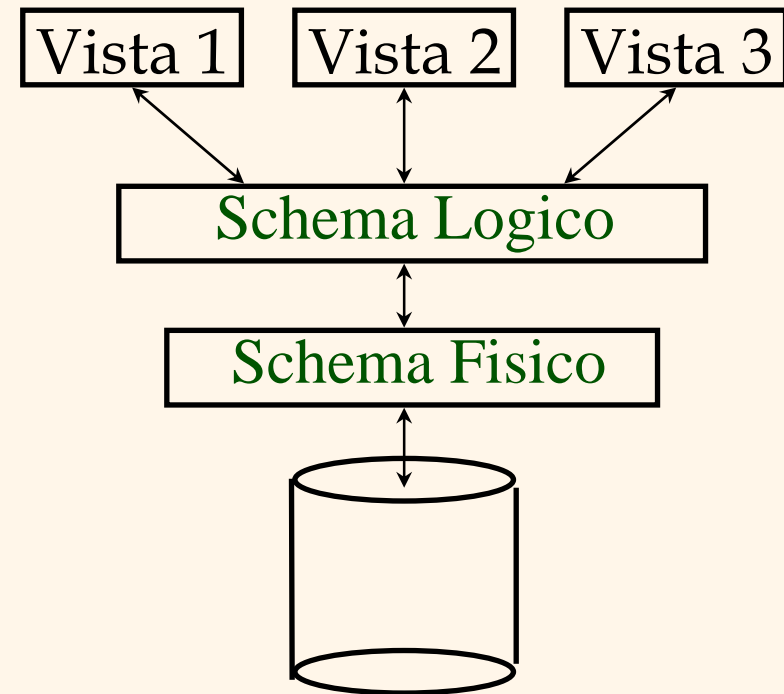
Modelli di dati

- ❖ Un modello di dati è una collezione di concetti per la descrizione dei dati
- ❖ Uno schema è una descrizione di una particolare collezione di dati, che fa uso del modello di dati fornito
- ❖ Il modello di dati relazionale è il modello oggi più usato
 - Concetto chiave: relazione, fondamentalmente una tabella con righe e colonne
 - Ogni relazione ha uno schema, che descrive le colonne, o campi

Livelli di astrazione

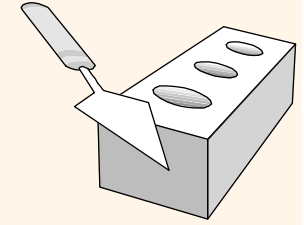


- ❖ Molte viste, un singolo schema logico (concettuale) e uno schema fisico
 - Le viste descrivono i dati come vengono visti dagli utenti
 - Lo schema logico definisce la struttura logica
 - Lo schema fisico descrive i file e gli indici usati



** Gli schemi sono definiti usando il DDL; i dati sono modificati/interrogati usando il DML .*

Esempio: La base di dati di una Università



❖ Schema logico:

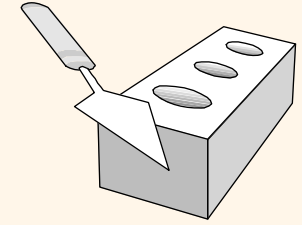
- Studenti(sid:string, nome:string, login: string, età:integer, media:real)
- Corsi(cid:string, cnome:string, crediti:integer)
- Iscritto(sid:string, cid:string, voto:string)

❖ Schema fisico:

- Le relazioni sono memorizzate come file non ordinati
- Indice sulla prima colonna di Studenti

❖ Schema esterno (vista):

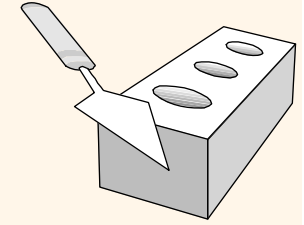
- info_corso(cid:string, iscritti:integer)



*Indipendenza dei dati **

- ❖ Le applicazioni sono separate dalla struttura e dalla memorizzazione dei dati
- ❖ Indipendenza logica dei dati: protezione dalle modifiche alla struttura logica dei dati
- ❖ Indipendenza fisica dei dati: protezione dalle modifiche alla struttura fisica dei dati

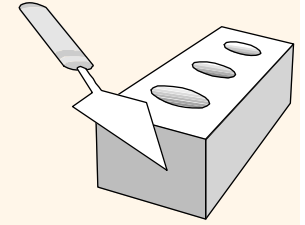
** Uno dei più importanti benefici dell'uso di un DBMS!*



Controllo di concorrenza

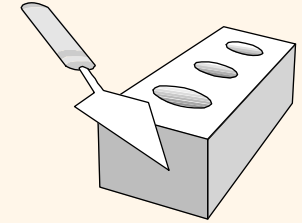
- ❖ L'esecuzione concorrente di programmi utente è essenziale per ottenere buone prestazioni dal DBMS
 - Poiché gli accessi al disco sono frequenti, e relativamente lenti, è importante tenere occupata la CPU lavorando su diversi programmi utente concorrentemente
- ❖ Le azioni interallacciate di diversi programmi utente possono portare a inconsistenza: ad esempio, un assegno viene pagato mentre viene calcolato il bilancio del conto corrente
- ❖ Il DBMS garantisce che tali problemi non si presentino: ogni utente può immaginare di essere l'unico utente del sistema

Transazione: l'esecuzione di un programma sulla base di dati



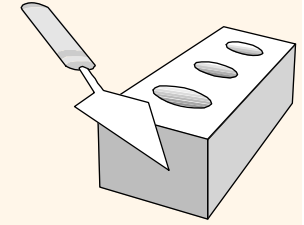
- ❖ Il concetto chiave è la transazione, che è una sequenza atomica di azioni sulla base di dati (letture/scritture)
- ❖ Ciascuna transazione, eseguita completamente, deve lasciare la base di dati in uno stato consistente se esso era consistente quando la transazione ha avuto inizio
 - Gli utenti possono specificare alcuni semplici vincoli di integrità sui dati, e il DBMS garantirà tali vincoli
 - Al di là di questo, il DBMS non capisce realmente la semantica dei dati (ad esempio non capisce come sono calcolati gli interessi su un conto bancario)
 - Quindi, garantire che una transazione (eseguita da sola) conservi la consistenza dei dati è, in ultima analisi, responsabilità dell'utente!

Scheduling di transazioni concorrenti



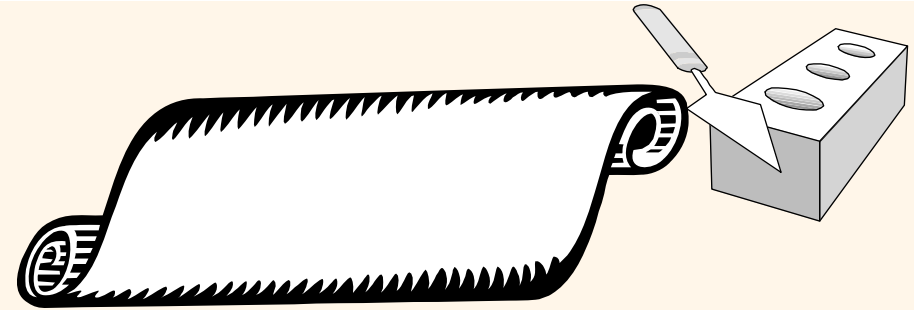
- ❖ Il DBMS garantisce che l'esecuzione di $\{T_1, \dots, T_n\}$ è equivalente a una qualche esecuzione seriale $T_1' \dots T_n'$
 - Prima di leggere/scrivere un oggetto, una transazione richiede un *lock* sull'oggetto, e aspetta finché il DBMS fornisce tale *lock*. Tutti i *lock* sono rilasciati al termine della transazione (protocollo di *locking* Strict 2PL)
 - Idea: se una azione di T_i (ad esempio, scrivere X) influenza T_j (che magari legge X), una tra T_i e T_j , diciamo T_i , otterrà il blocco su X per prima, e T_j dovrà aspettare fino al termine di T_i ; ciò in pratica dà un ordinamento alle transazioni
 - Che succede se T_j ha già un blocco su Y e T_i in seguito richiede un blocco su Y ? (Deadlock!) T_i o T_j deve essere interrotta e fatta ripartire!

Atomicity



- ❖ Il DBMS garantisce *l'atomicità* (una transazione è una unità indivisibile - proprietà tutto-o-niente) anche se il sistema va in *crash* durante una transazione
- ❖ Idea: mantenere un log (una storia) di tutte le azioni effettuate dal DBMS nell'eseguire un insieme di transazioni:
 - prima che una modifica sia fatta alla base di dati, la voce corrispondente del log viene messa al sicuro (protocollo WAL: spesso il supporto del sistema operativo in questo ambito è inadeguato)
 - dopo un crash, gli effetti delle transazioni eseguite parzialmente sono annullati usando il log (grazie al WAL, se una voce del log non era stata salvata prima del crash, le corrispondenti modifiche alla base di dati non erano state apportate!)

Il Log



- ❖ Le seguenti azioni sono registrate nel log:
 - Ti scrive un oggetto: il vecchio valore e il nuovo valore
- ❖ Il record nel log deve essere salvato su disco prima della pagina modificata
 - Ti termina/si interrompe: un record nel log registra tale azione
- ❖ I record nel log sono collegati tramite l'ID della transazione, così che sia semplice annullare una transazione specificata (ad esempio per risolvere un deadlock)
- ❖ Il log è spesso duplicato e archiviato in una memoria “stabile”
- ❖ Tutte le attività connesse al log (e di fatto tutte le attività connesse al controllo di concorrenza, come blocco/sblocco, gestione dei deadlock, ecc) sono gestite dal DBMS in maniera trasparente

Le basi di dati fanno felici queste persone...

- ❖ Utenti finali e produttori di DBMS
- ❖ Programmatori di applicazioni per DBMS
 - Ad esempio abili gestori di siti Web
- ❖ Amministratori di basi di dati (DBA)
 - Progettano gli schemi logico/fisico
 - Gestiscono la sicurezza e le autorizzazioni
 - Gestiscono disponibilità dei dati, ripristino da crash
 - Modificano lo schema della base di dati con l'evolversi delle necessità

Devono capire come funziona un DBMS!



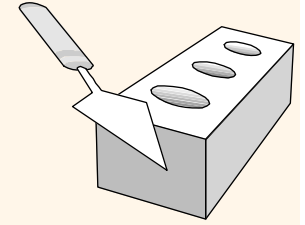
Struttura di un DBMS

Questi strati devono tenere in considerazione il controllo di concorrenza e il ripristino

- ❖ Un tipico DBMS ha una struttura stratificata
- ❖ La figura non mostra le componenti di controllo di concorrenza e di ripristino
- ❖ Questa è una tra le possibili architetture; ciascun sistema ha le proprie varianti



Sommario



- ❖ Un DBMS è usato per mantenere e interrogare grandi insiemi di dati
- ❖ Tra i benefici, il ripristino dai crash del sistema, l'accesso concorrente, il rapido sviluppo di applicazioni, l'integrità dei dati e la sicurezza
- ❖ I livelli di astrazione portano all'indipendenza dei dati
- ❖ Un DBMS tipicamente ha un'architettura stratificata
- ❖ I DBA svolgono un lavoro di responsabilità e ben pagato!
- ❖ La ricerca e sviluppo nei DBMS è una della più vaste ed eccitanti aree dell'informatica

