BASI DI DATI 2

APPLICAZIONI DI DESIGN & TUNING DI DATABASE

a.a 2024/2025

Prof.ssa G. Tortora

Progettazione di un db

- Abbiamo già visto in dettaglio gli aspetti teorici di progettazione di un db.
- L'attività di design di un database nel suo complesso è un processo sistematico che segue una metodologia ben definita, la metodologia di design, ed è spesso legata al tool di design fornito dalla casa produttrice:
  - (Oracle, Sybase, *ecc.*)
- Non analizzeremo una specifica metodologia, ma piuttosto vedremo come viene svolta la fase di progettazione di un db in una grande azienda.

Large Database

- In generale, sono considerati grandi database quei database con diverse decine di gigabyte di dati e 30-40 o più distinti tipi di entità.

- Tra questi figurano i sistemi di gestione delle transazioni – attivi 24 ore al giorno con grandi volumi di transazioni e centinaia o migliaia di utenti.

Il ruolo dei Sistemi Informativi in un'azienda

Il contesto organizzativo per l'utilizzo di database system

- I database sono una parte fondamentale di ogni sistema informativo aziendale.

- Il riconoscimento dell'importanza strategica di un'accurata gestione dei dati ha portato alla creazione di nuove figure professionali:
  - Un DBA o un dipartimento per l'amministrazione di database.
  - Un management per la gestione delle risorse informative.

L'importanza della gestione dei dati

- Un'accurata gestione dei dati è fondamentale perché:
  - I dati sono una risorsa dell'azienda: una corretta gestione rende più efficiente il lavoro.
  - In un'azienda sempre più funzionalità sono informatizzate, aumentando la richiesta di memorizzare grandi quantità di informazioni che siano reperibili al loro valore attuale.
  - All'aumentare dei dati e delle applicazioni, aumentano anche le relazioni tra i dati da modellare e memorizzare.

L'importanza dei database

- L'utilizzo di sistemi di basi di dati soddisfa pienamente i punti appena visti.

- Un sistema di base di dati ha anche altre due caratteristiche preziose in ambiti aziendali:
  - L'indipendenza dai dati protegge i programmi applicativi da cambiamenti sia della logica aziendale, sia della memorizzazione fisica.
  - Gli schemi esterni (o *viste*) permettono l'uso agli stessi dati da parte di diverse applicazioni.

Caratteristiche chiave

- Integrazione dei dati tra diverse applicazioni in un singolo DB.
- Facilità di sviluppo di nuove applicazioni usando linguaggi ad alto livello tipo SQL.
- Possibilità da parte dei manager di interrogare i dati ed avere risultati aggiornati.

L’evoluzione e la diffusione

- Dai primi anni ’70 fino alla metà degli anni ’80 c’era la tendenza di creare grossi repository, gestiti da un singolo DBMS centralizzato.
- Negli ultimi 15 anni la situazione si è invertita per i seguenti motivi:

1. La diffusione di Personal Database (Es. *Access, Excel, FoxPro, SQL, Anywhere*) permette a molte categorie di utenti di definire dei db personali. È possibile scaricare porzioni di DB dal server, lavorarci sopra e ri-memorizzare il tutto sul server.
2. L’avvento di DBMS distribuiti e *client/server* permette di allocare i dati su più computer per un migliore controllo ed una elaborazione locale più veloce. Gli utenti locali possono accedere ai dati remoti come client del DBMS o tramite web.

Strutturazione dei database system aziendali (2)

3. Molte organizzazioni utilizzano dizionari dei dati, cioè dei mini-DBMS per gestire metadati, quali:
   - Strutture di database:
     - Descrizione degli schemi.
     - Descrizioni del design fisico (strutture di accesso, file, taglia dei record,ecc..).
   - Informazioni sugli utenti (responsabilità, diritti di accesso).
   - Descrizioni di alto livello di transazioni ed applicativi e relazioni utente-transazioni.
   - Relazioni tra transazioni e dati.
   - Statistiche sull'utilizzo di porzioni di database.

Sistemi di gestione delle transazioni

- Sono sistemi *business critical*.
- Attivi 24 ore su 24.
- Servono centinaia di transazioni al minuto da terminali remoti e locali.
- Il tempo di risposta medio e massimo ed il numero medio di transazioni per minuto sono dei fattori critici.

- Per questo tipo di sistemi la progettazione fisica del DB è un elemento vitale.

Il ciclo di vita di un sistema informativo

Il sistema informativo

- In grosse aziende, un database system è solo una parte di un sistema informativo.
- Un sistema informativo è composto da:
  - Dati
  - DBMS
  - Hardware
  - Media di memorizzazione
  - Applicativi che interagiscono con i dati
  - Il personale che gestisce o usa il sistema
  - Gli applicativi che gestiscono l'aggiornamento dei dati
  - I programmatori che sviluppano gli applicativi

Il ciclo di vita

- Il ciclo di vita di un sistema informativo (risorse per raccolta, gestione uso e disseminazione) è detto macro ciclo di vita.
- Il ciclo di vita di un sistema di base di dati è detto micro ciclo di vita.
- Con l'aumentare della complessità e delle funzioni svolte dai DBMS, questa suddivisione diventa sempre più sfumata.

Le fasi del macro ciclo di vita

1. Analisi di fattibilità
   - Si analizzano le potenziali aree di applicazione, si effettuano degli studi di *costi/benefici*, si determina la complessità di dati e processi, e si impostano le priorità tra le applicazioni.

2. Raccolta ed analisi dei requisiti
   - Comprende una raccolta dettagliata dei requisiti con interviste ai potenziali utenti, per definire le funzionalità del sistema.

Le fasi del macro ciclo di vita (2)

3. Progettazione
   - Si divide in progettazione del database e progettazione degli applicativi che utilizzano il database.

4. Implementazione
   - Si implementa il S.I., si carica il DB e si implementano e si testano le transazioni.

Le fasi del macro ciclo di vita (3)

5. Validazione e testing
   - Si verifica che il sistema soddisfi i requisiti e le performance richieste.

6. Rilascio e manutenzione
   - La fase operativa del nuovo sistema parte quando tutte le funzionalità sono state validate.
   - Il rilascio può essere preceduto da una fase di addestramento del personale al nuovo sistema.
     Se emergono nuovi funzionalità da implementare, si ripetono i passi precedenti, per includerle nel sistema

Le fasi del micro ciclo di vita

- Le attività del ciclo di vita di un database system includono le seguenti 8 fasi:

1. Definizione del sistema
   - Si definisce l'ambito del sistema di base di dati, i suoi utenti e le funzionalità.
   - Si identificano le interfacce per le categorie di utenti, i vincoli sui tempi di risposta ed i requisiti hardware.

Le fasi del micro ciclo di vita (2)

2. Progettazione della base di dati
   - Si realizza la progettazione logica e fisica per il DBMS scelto.

3. Implementazione della base di dati
   - Si specificano le definizioni concettuali, esterne ed interne, si creano i file del db vuoti e si implementa dell'eventuale software applicativo di supporto.

Le fasi del micro ciclo di vita (3)

4. Caricamento / conversione dei dati
   - Si popola il database, o inserendo direttamente i dati o convertendo file esistenti nel nuovo formato.

5. Conversione delle applicazioni
   - Si convertono le vecchie applicazioni software al nuovo sistema.

6. Test e validazione
   - Si effettuano test e validazione del nuovo sistema.

Le fasi del micro ciclo di vita (4)

7. Operation
   - Il sistema di base di dati e le sue applicazioni diventano operativi. In genere, per un certo tempo vengono utilizzati in parallelo il vecchio ed il nuovo sistema.

8. Controllo e manutenzione
   - Il sistema è sottoposto a constante monitoraggio. Eventualmente si possono gestire aggiunte nei dati o nelle funzionalità presenti.

Il processo di progettazione di un database

La progettazione di un db

- La fase più interessante nel micro ciclo di vita è quella della progettazione.

- Il problema della progettazione può essere riformulato come:
  - Progettare la struttura logica e fisica di uno o più database, per soddisfare i requisiti degli utenti di un'organizzazione su un determinato insieme di operazioni.

La progettazione di un db (2)

- Gli scopi della fase di progettazione sono:
  1. Soddisfare i requisiti sui dati che interessano gli utenti e a cui accedono le applicazioni.
  2. Fornire una strutturazione delle informazioni naturale e facile da comprendere.
  3. Soddisfare i requisiti di elaborazione e di prestazioni (*tempo di risposta, spazio di memorizzazione, ecc.*).
- È difficile raggiungere tutti gli scopi, poiché alcuni sono in contrasto tra loro.
  - Un modello più comprensibile può comportare un costo in termini di prestazioni.

Attività della progettazione

- Il processo di progettazione è costituito da due attività parallele:
  - Progettazione di strutture e contenuti dei dati (*progettisti di basi di dati*).
  - Progettazione delle applicazioni che usano la base di dati (*ingegneri del software*).
- Le metodologie di progettazione di database si sono focalizzate sulla prima attività (approccio data-driven *vs.* progettazione process-driven).
- E' ormai riconosciuto, però, che progettisti di db e ingegneri del software debbano collaborare quanto più possibile, servendosi di tools di design per combinarle.

Fasi della progettazione di un database

- La progettazione di un database può essere vista come composta da sei fasi principali:
  1. Raccolta ed analisi dei requisiti.
  2. Progettazione dello schema concettuale.
  3. Scelta del DBMS.
  4. Mapping del data model (*design logico*).
  5. Progettazione dello schema fisico.
  6. Implementazione e tuning del database system.

Sequenzialità delle fasi

- Le sei fasi non sono eseguite in sequenza: spesso modifiche ad un livello devono essere propagate a quello superiore, creando dei cicli di feedback.

Fasi della progettazione di un database

1. Raccolta ed analisi dei requisiti.
2. Progettazione dello schema concettuale.
3. Scelta del DBMS.
4. Mapping del data model (*design logico*).
5. Progettazione dello schema fisico.
6. Implementazione e tuning del database system.

Fase 1:
Raccolta ed analisi dei requisiti

- Per progettare un db è necessario conoscere ed analizzare le aspettative degli utenti nel modo più dettagliato possibile.
- Questo processo é detto raccolta ed analisi dei requisiti.
- Per specificare i requisiti, é necessario individuare tutte le componenti che interagiranno col db.
  Tipicamente queste sono:
  - Gli utenti.
  - Le applicazioni.

Attività della Fase 1

- La Fase 1 comprende 4 attività:

1. Identificare le principali aree di applicazione, gli utenti che useranno il db e quelli il cui lavoro sarà influenzato dal db. Individuare in ogni gruppo di persone un rappresentante per portare avanti la raccolta delle specifiche.

2. Analizzare la documentazione già esistente riguardante le applicazioni.
   Esaminare anche altri tipi di documentazione, (*form, report, grafici aziendali, ecc.*) che in qualche modo possono influenzare i requisiti.

Attività della Fase 1 (2)

3. Esaminare il contesto operativo e l'utilizzo pianificato delle informazioni.
   Questa attività include l'analisi delle transazioni, dei flussi di informazioni e la specifica dei dati di input e output per ogni transazione

4. Intervistare gli utenti finali per determinare priorità ed importanza previste per le varie applicazioni.

Sviluppo dei requisiti

- Spesso all'inizio i requisiti sono informali, incompleti, inconsistenti e parzialmente incorretti.
- È quindi necessario molto lavoro per trasformarli in specifiche da dare a programmatori e tester.
- Poiché i requisiti si riferiscono ad un sistema non ancora esistente, inevitabilmente vengono spesso modificati.

Tecniche di specifica dei requisiti

- Per trasformare i requisiti in una forma strutturata, si utilizzano delle tecniche di specifica dei requisiti.

- Le principali tecniche comprendono:
  - Analisi orientata agli oggetti (OOA).
  - Diagramma di flusso dei dati (DFD).
  - Raffinamento degli obiettivi dell'applicazione.

- Esistono altre tecniche che producono una specifica formale dei requisiti che consentono verifiche matematiche di consistenza (*analisi simbolica "what-if*).
  - Sebbene più difficili da usare, queste tecniche sono fondamentali per applicazioni *mission-critical*.

Upper CASE Tools

- È possibile utilizzare dei tool CASE (*Computer Aided Software Engineering*) per controllare la completezza e la consistenza delle specifiche detti Upper CASE tool.

- Altri tool permettono di evidenziare i collegamenti tra requisiti ed altre entità di progetto, quali moduli di codice, casi di test, ecc. (basi di dati di tracciabilità).

Importanza delle specifiche

- La fase di raccolta ed analisi dei requisiti richiede un grande sforzo in termini di tempo, ma è cruciale per il successo del sistema informativo.

- Un errore dovuto a requisiti errati può essere estremamente costoso poiché può comportare la re-implementazione di buona parte del lavoro.
  - Il sistema potrebbe non rispondere alle richieste del cliente e non essere usato affatto.

Fasi della progettazione di un database

1. Raccolta ed analisi dei requisiti.
2. Progettazione dello schema concettuale.
3. Scelta del DBMS.
4. Mapping del data model (*design logico*).
5. Progettazione dello schema fisico.
6. Implementazione e tuning del database system.

Fase 2:
Design del Database Concettuale

- La seconda fase del progetto di un db consta di due attività parallele:
  - Progettazione dello schema concettuale.
    Si esaminano i requisiti per produrre lo schema concettuale del db.
  - Progettazione di transazioni ed applicazioni.
    Si esaminano le applicazioni del db per produrre specifiche di alto livello delle applicazioni.

Progettazione dello schema concettuale

- Lo schema concettuale deve essere indipendente dal DBMS per i seguenti motivi:
  1. Lo scopo dello schema concettuale é fornire una comprensione completa di struttura, semantica, relazioni e vincoli del db. Legarsi ad un DBMS porterebbe a delle restrizioni che influenzerebbe lo schema.

  2. Lo schema concettuale é una descrizione stabile dei contenuti del db. La scelta del DBMS e le decisioni di progettazione successive possono cambiare senza che questo debba essere modificato.

Progettazione dello schema concettuale (2)

3. L'utilizzo di un data model di alto livello é più espressivo e generale dei modelli di dati utilizzati dai singoli DBMS, ed è quindi più comprensibile per gli utenti.

4. La descrizione diagrammatica dello schema concettuale può essere usata molto efficacemente come mezzo di comunicazione tra gli utenti del database, i progettisti ed gli analisti. I modelli di dati dei DBMS, essendo di livello più basso, spesso mancano di un tale livello di espressività.

Caratteristiche di un modello concettuale dei dati di alto livello

- Un data model di alto livello deve godere delle seguenti proprietà:
  1. Espressività.
     Il data model deve permettere una facile distinzione tra tipi di dati, relazioni e vincoli.

  2. Semplicità e comprensibilità.
     Il modello deve essere semplice, per consentire a utenti non esperti di comprendere ed utilizzare i suoi concetti.

Caratteristiche di un modello concettuale dei dati di alto livello (2)

3. Minimalità.
   Il modello dovrebbe avere pochi concetti di base, non sovrapponibili.

4. Rappresentazione diagrammatica.
   Il modello dovrebbe avere una notazione diagrammatica per rappresentare schemi concettuali di facile comprensione.

5. Formalità.
   Il modello deve fornire dei formalismi per specificare in modo non ambiguo i dati.

Approcci alla progettazione di uno schema concettuale

- Per progettare uno schema concettuale, é necessario individuare le componenti di base di uno schema. Queste sono:
  - Le entità.
  - Le relazioni.
  - Gli attributi.
  - I vincoli di cardinalità e partecipazione.
  - Le chiavi.
  - Le gerarchie di specializzazione/generalizzazione.
  - Le entità deboli.

Approcci alla progettazione di uno schema concettuale (2)

- Esistono due approcci alla progettazione di uno schema concettuale:
  - Progetto di schema centralizzato (o one-shot).
    Tutti i requisiti di applicazioni differenti e diversi gruppi di utenti sono fusi in un singolo insieme di requisiti prima di iniziare la progettazione. Al DBA spetta il compito di decidere come unire i requisiti e di progettare l'intero schema.
    Una volta progettato l'intero schema concettuale si progettano gli schemi esterni.
  - Integrazione di viste.
    I requisiti non vengono fusi:
    - inizialmente si progetta uno schema (o vista) per ogni gruppo di utenti.
    - Successivamente, in una fase di integrazione, le viste sono fuse in un unico schema concettuale globale.

Strategie per la progettazione di schemi

- Esistono differenti strategie per creare uno schema concettuale partendo dai requisiti:
  - Strategia Top-Down.
    Si parte da uno schema con astrazioni di alto livello, che vengono successivamente raffinate:
    - Specificare entità ad alto livello.
    - Quando si vanno a specificare gli attributi dette entità si spezzano in entità di livello inferiore e si introducono relazioni.

    Es:specializzazione di un tipo di entità in sottoclassi.
  - Strategia Bottom-Up.
    Si parte da uno schema contenente astrazioni di base, che vengono via via combinate e messe in relazione tra loro.

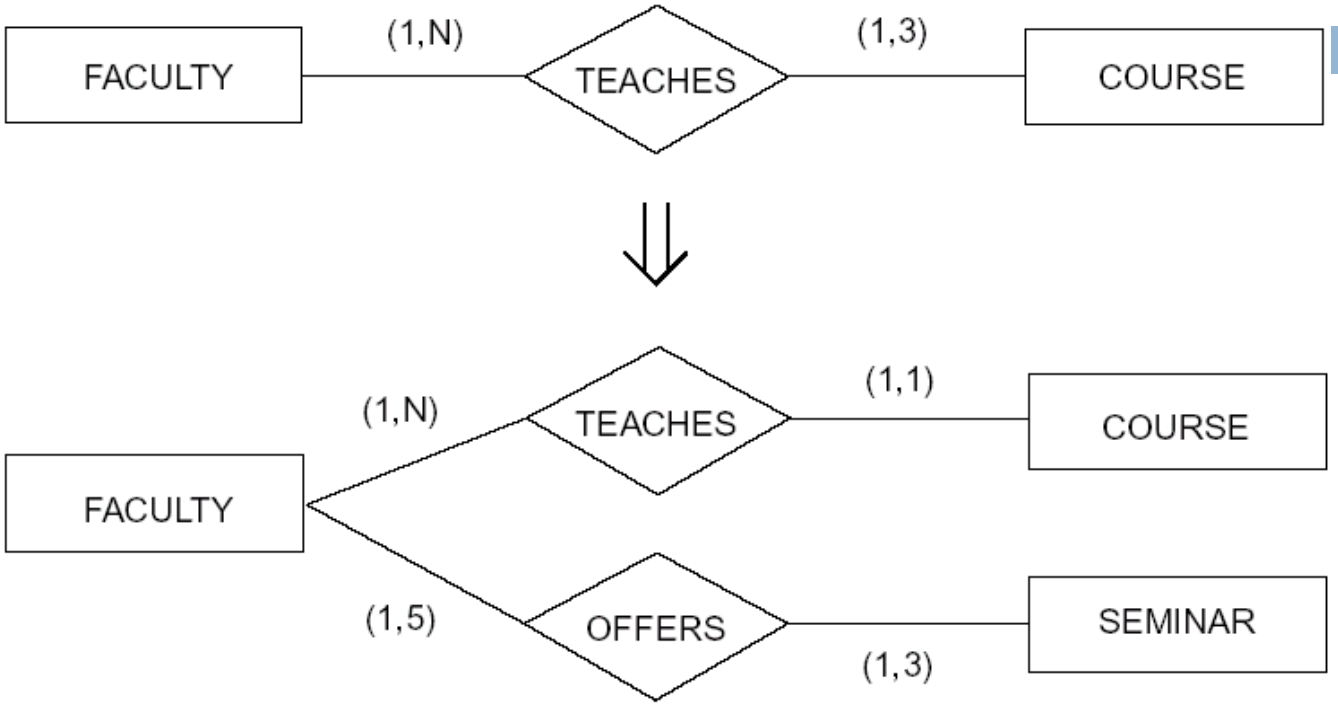
    Es: generalizzare tipi di entità in superclassi di alto livello.

Strategie per la progettazione di schemi (2)

- Strategia Inside-Out.
  È una specializzazione del bottom-up, in cui l'attenzione è focalizzata su un nucleo centrale di operazioni. La modellazione, quindi, si allarga verso l'esterno, inglobando nuovi concetti collegati a quelli già considerati.

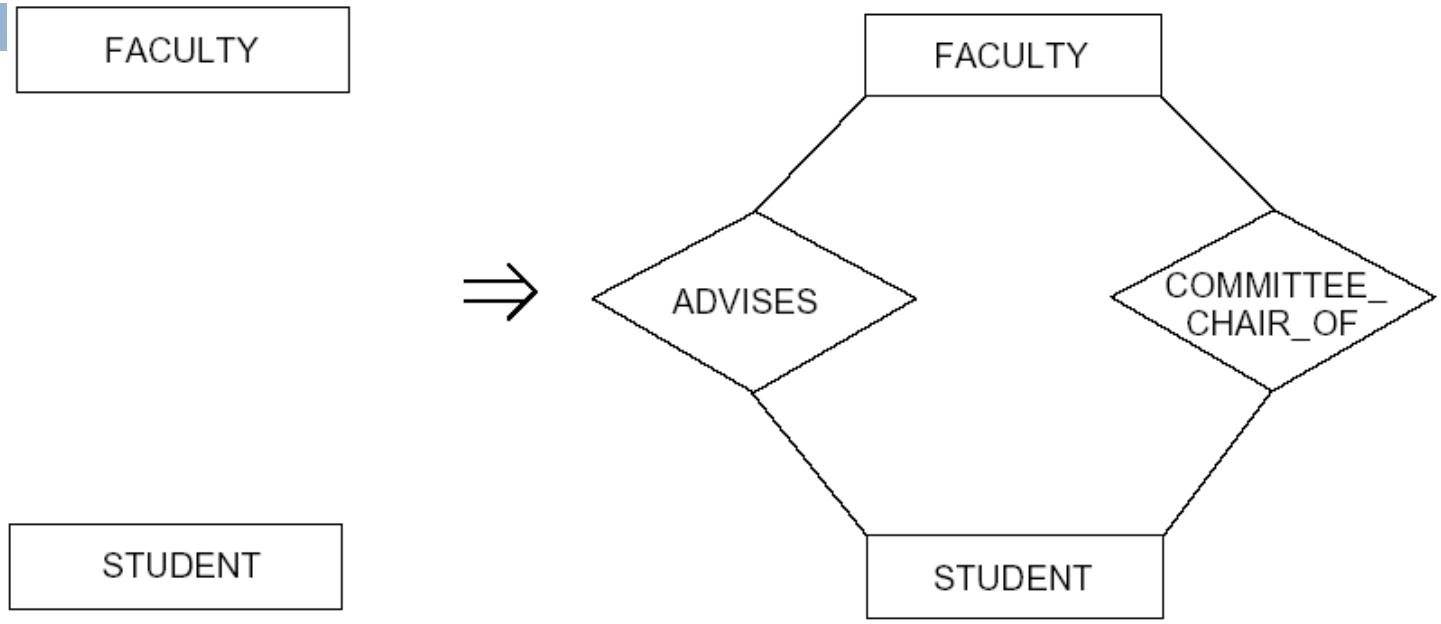
- Strategia Mixed.
  Non prevede una strategia uniforme per tutto il progetto, ma dà la possibilità di usare metodi diversi per le varie componenti, che vengono combinate successivamente.

Strategia Top-Down: *Esempio*



- Un raffinamento Top-Down:
  - L'entità Course è raffinata in Course e Seminar e la relazione TEACHES è spezzata in TEACHES e OFFERS.

Strategia Bottom-Up: *Esempio*



- Un raffinamento Bottom-Up:
  - Dopo la definizione delle entità, si ha un raffinamento, definendo le relazioni esistenti.

Integrazione di schemi

- Analizziamo una metodologia per l'integrazione di schemi in uno schema globale di database (necessaria per grossi database).

- L'integrazione di schemi può essere divisa in quattro sottocompiti:
  1. Identificazione di corrispondenze e conflitti tra gli schemi.
     In questa fase è possibile individuare quattro possibili tipi di conflitti:

Conflitti tra schemi

a) Conflitti di nome.

Possono essere di due tipi:

- **Sinonimi**: due schemi utilizzano termini diversi per identificare lo stesso concetto.

  Es: in due schemi differenti esistono il tipo di entità CUSTOMER e CLIENT che descrivono lo stesso concetto.

- **Omonimi**: due schemi utilizzano lo stesso termine per identificare concetti diversi.

b) Conflitti di tipo.

Lo stesso concetto può essere espresso in schemi diversi con costrutti di modellazione diversi.

Es: un attributo in uno schema e un tipo di entità in un altro schema.

Conflitti tra schemi (2)

c) Conflitti di dominio.
Un attributo può avere domini differenti in schemi diversi.

Es: intero in uno schema e carattere in un altro.

Conflitti di unità di misura (un attributo è descritto in **m.** in uno schema e in **Km.** in un altro).

d) Conflitti tra vincoli.
Due schemi possono imporre vincoli differenti *Es. La chiave di un tipo di entità può risultare diversa in due schemi differenti.*

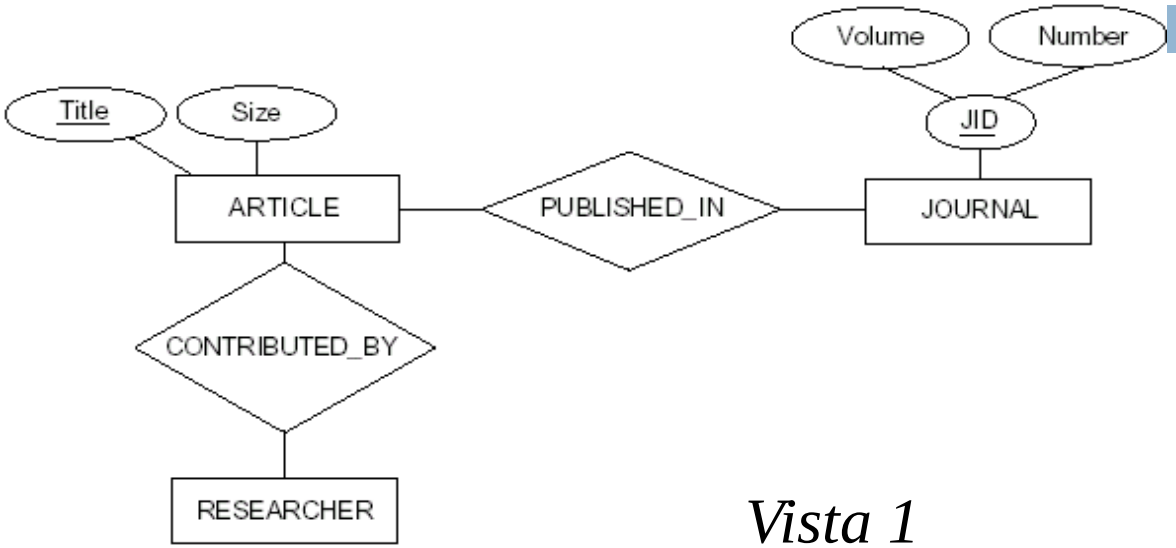
Integrazione di schemi (2)

2. Modifica delle viste per renderle conformi.

3. Fusione delle viste.
   Si crea uno schema globale fondendo tutte le viste. I concetti corrispondenti devono essere rappresentati solo una volta. Non è un compito automatizzabile, e richiede una notevole esperienza.

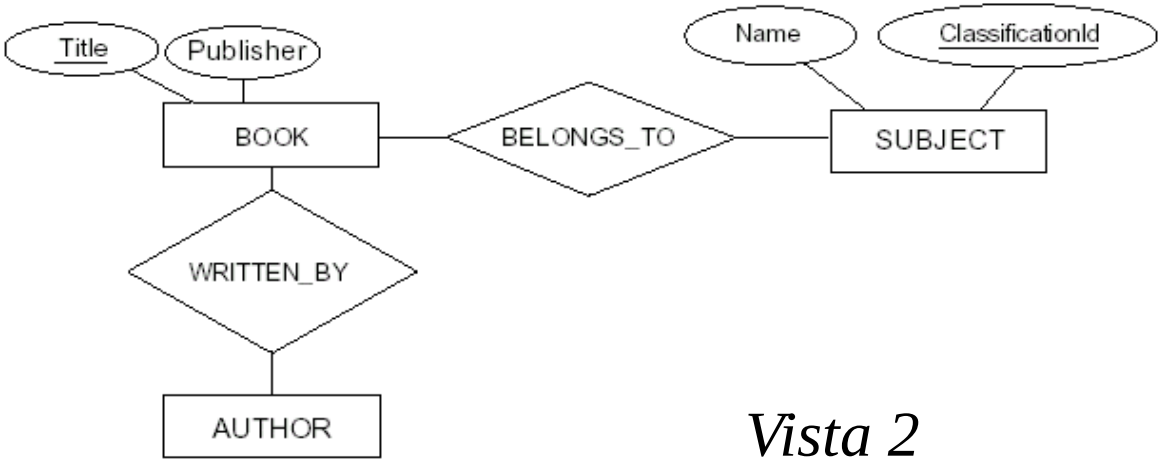
4. Ristrutturazione.
   Lo schema può essere analizzato per eliminare ridondanze.

Integrazione: *Esempio*



Vista 1

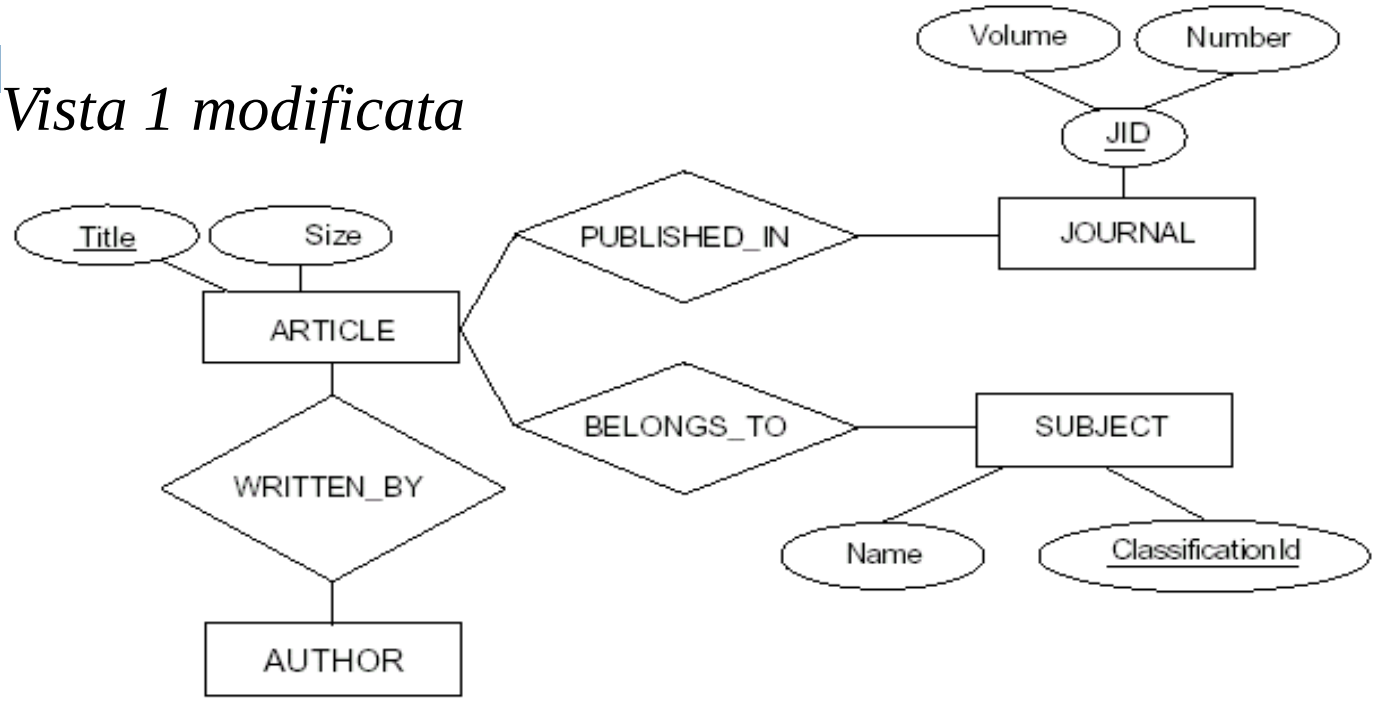
- Due viste di un database bibliografico.



Vista 2

Integrazione: *Esempio* (2)

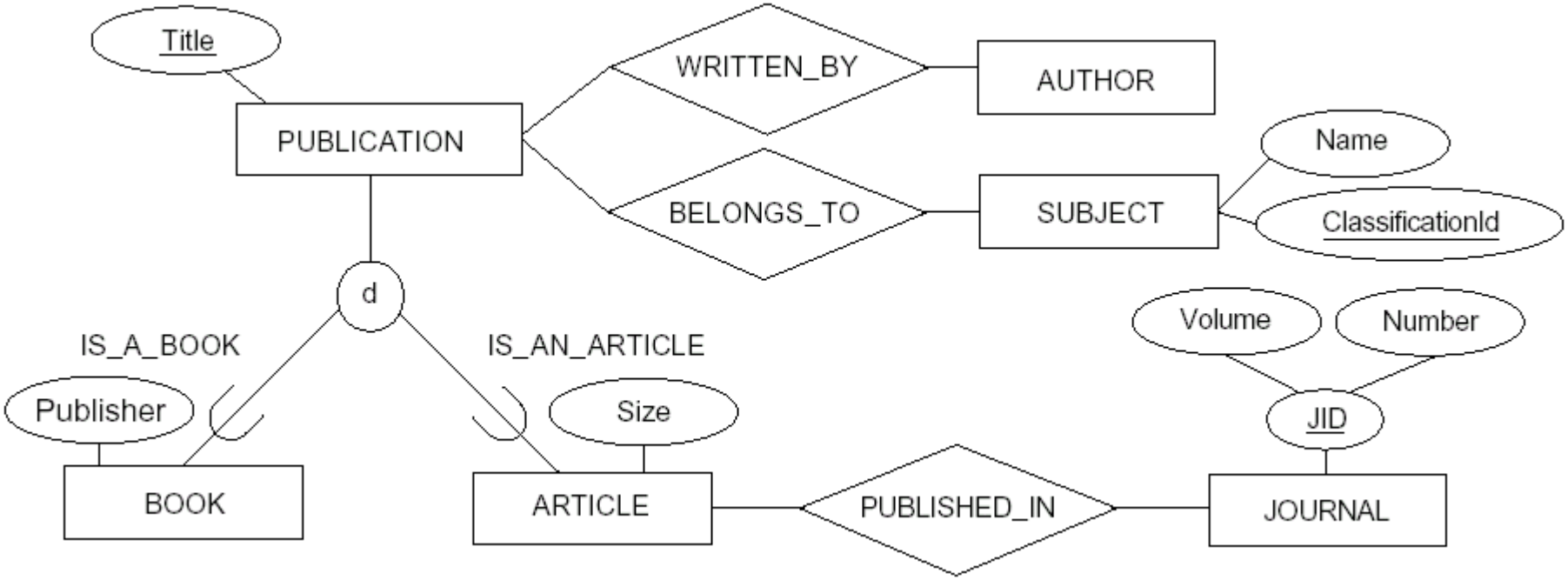
Vista 1 modificata



- La vista 1 viene modificata per rendere i contenuti conformi alla vista 2:
  - Researcher e Author
  - CONTRIBUTED_BY e WRITTEN_BY

  sono sinonimi.

Integrazione: *Esempio* (3)



- Schema finale ottenuto dall'integrazione delle due viste con la generalizzazione di **book** e **article** in **publication**.

Progettazione di transazioni

- Lo scopo di questa fase è di progettare le transazioni (o applicazioni) del database in modo indipendente dal DBMS.
- Specificare le caratteristiche funzionali delle transazioni assicura che lo schema del DB includerà le informazioni che esse richiedono.
- Difficilmente nella fase di progettazione si ha una visione completa di tutte le transazioni da implementare:
  - molte saranno identificate e realizzate al termine dell'implementazione del database.

Progettazione di transazioni (2)

- Una tecnica usata per specificare le transazioni prevede l'identificazione di:
  - Input
  - Output
  - Comportamento funzionale

- Definendo i parametri di *input/output* ed il flusso di *controllo funzionale* interno, è possibile specificare una transazione in un modo concettuale indipendente dal sistema.

Categorie di transazioni

- È possibile raggruppare le transazioni in tre categorie:
  - Transazioni di retrieval.
    Usate per recuperare dati da visualizzare o su schermo o su report.
  - Transazioni di update.
    Usate per inserire o modificare dati.
  - Transazioni miste.
    Usate per applicazioni più complesse che richiedono sia retrieval che update.

- Il design di transazioni è considerato parte dell'Ingegneria del SW.

Fasi della progettazione di un database

1. Raccolta ed analisi dei requisiti.
2. Progettazione dello schema concettuale.
3. Scelta del DBMS.
4. Mapping del data model (*design logico*).
5. Progettazione dello schema fisico.
6. Implementazione e tuning del database system.

Fase 3:
Scelta del DBMS

- La scelta del DBMS è influenzata da tre fattori:
  - Fattori tecnici.
  - Fattori economici.
  - Fattori organizzativi/aziendali.

Fattori tecnici

- Tra i fattori tecnici che possono influenzare la scelta del DBMS troviamo:
  - Data model del DBMS (*relazionale, O-O, ecc.*).
  - Strutture di memorizzazione offerte.
  - Disponibilità di interfacce per utenti e programmatori.
  - Disponibilità di tool di sviluppo.
  - Linguaggio di query supportato.
  - Possibilità di interagire con altri DBMS.

Fattori economici

- Tra i fattori economici che possono influenzare la scelta del DBMS troviamo:
  - Costi di acquisto del software (*inclusi tool per GUI, recovery/backup, design, linguaggi di programmazione, ecc.*).
  - Costi di manutenzione (*assistenza del rivenditore e costi di aggiornamento*).
  - Costi di acquisizione nuovo hardware (*memoria, terminali, driver di disco, ecc.*).
  - Costi creazione e conversione database.
  - Costi del personale.
  - Costi del training.
  - Costi operativi.

Fattori aziendali

- Tra i fattori aziendali che possono influenzare la scelta del DBMS troviamo:
  - Adozione di una "*filosofia*" in tutta l'azienda.
    Accettare un DBMS implica accettarne il data model, il paradigma di programmazione, i tool di sviluppo, ecc..
  - Familiarità del personale con il sistema.
    Scegliere un sistema già conosciuto diminuisce i costi di training.
  - Disponibilità di servizi post-vendita.
    La disponibilità di assistenza post-vendita può risultare molto importante.

Altri fattori

- Ulteriori fattori da considerare nella scelta del DBMS sono:
  - Portabilità su più piattaforme.
  - Disponibilità di tool di backup, recovery, security, ecc..
  - Integrazione con “*soluzioni complete*” per il sistema informativo.

Prodotti inclusi nei DBMS

- I DBMS si stanno evolvendo sempre più, integrando un gran numero di pacchetti software. Tra questi troviamo:
  - Browser ed editor di testo.
  - Generatori di report.
  - Software di comunicazione (*monitor TP*).
  - Soluzioni di data entry e data display, quali form, menù, ecc..
  - Tool di interrogazione via Web.
  - Tool di progettazione visuale.

Fasi della progettazione di un database

1. Raccolta ed analisi dei requisiti.
2. Progettazione dello schema concettuale.
3. Scelta del DBMS.
4. Mapping del data model (*design logico*).
5. Progettazione dello schema fisico.
6. Implementazione e tuning del database system.

Fase 4:
Data Model Mapping

- La creazione di schemi concettuali ed esterni nel data model specifico del DBMS selezionato avviene in due passi:
  1. Mapping indipendente dal sistema.
     Il mapping non considera nessuna caratteristica specifica del DBMS.
     Es. La traduzione da E-R a Relazionale.

  2. Mapping per lo specifico DBMS.
     Si modifica lo schema ottenuto al passo 1 per adeguarlo alle caratteristiche ed ai vincoli dello specifico DBMS.
- **Risultato**: istruzioni DDL per specificare gli schemi concettuali ed esterni del DB.

Fasi della progettazione di un database

1. Raccolta ed analisi dei requisiti.
2. Progettazione dello schema concettuale.
3. Scelta del DBMS.
4. Mapping del data model (*design logico*).
5. Progettazione dello schema fisico.
6. Implementazione e tuning del database system.

Fase 5: Progettazione fisica

- Comprende la definizione di strutture di memorizzazione e di access path per i file del database.
- Esistono tre parametri che guidano la progettazione fisica di un database:
  - Tempo di risposta.
    È il tempo medio trascorso dalla sottomissione di una transazione alla ricezione dei risultati.
  - Utilizzazione di spazio.
    È lo spazio totale utilizzato dal db, compresi gli indici.
  - Throughput delle transazioni.
    È il numero medio di transazioni completate al minuto (*parametro critico per sistemi transazionali quali banche, prenotazioni su linee aeree*).

Progettazione fisica (2)

- A causa dell'importanza di tali parametri, spesso nei requisiti si includono i limiti per il caso medio e per il caso pessimo.
- Le prestazioni dipendono dalla taglia dei record e dal numero di record nei file, parametri che vanno stimati.
- Spesso si utilizzano dei prototipi del sistema per valutarne le prestazioni effettive.
- Devono essere considerati gli attributi usati per accedere ai record e gli indici primari e secondari necessari.

Fasi della progettazione di un database

1. Raccolta ed analisi dei requisiti.
2. Progettazione dello schema concettuale.
3. Scelta del DBMS.
4. Mapping del data model (*design logico*).
5. Progettazione dello schema fisico.
6. Implementazione e tuning del database system.

Fase 6:

Implementazione e Tuning del database system

- L'implementazione del database è a carico del DBA e dei db designer.
- Le istruzioni DDL sono compilate ed eseguite per creare gli schemi ed i file vuoti.
- Il db viene quindi popolato con i dati:
  - Se i dati sono già esistenti in un altro formato, può essere necessario implementare delle routine di conversione.
- Infine, i programmatori implementano le transazioni utilizzando comandi DML del DBMS.

Tuning del database

- La maggior parte dei DBMS includono tool di monitoraggio.
- In base ai dati collezionati, è possibile modificare tabelle, access path, query, ecc..
- Alcune query o transazioni possono essere riscritte per migliorare le prestazioni.
- Il processo di tuning continua durante tutta l'operatività del database system.

La progettazione fisica nei database relazionali

Scopi della progettazione fisica

- La progettazione fisica di un db si propone non solo di fornire delle strutture dati appropriate, ma anche di garantire delle buone performance del database system.

- Per effettuare una buona progettazione, è necessario conoscere le query, le transazioni e gli applicativi eseguiti sul database, analizzarne la frequenza di esecuzione, e gli eventuali vincoli posti nei requisiti.

Fattori che influenzano il Design Fisico

- Analisi di query:
  - Per ogni query si deve specificare:
    1. I file a cui accede la query.
    2. Gli attributi su cui sono specificate le condizioni di selezione.
    3. Gli attributi su cui sono specificate le condizioni di join.
    4. Gli attributi di cui la query recupera i valori.

- Gli attributi identificati nei punti 2 e 3 sono candidati per la definizione di access path.

Fattori che influenzano il Design Fisico (2)

- Analisi di transazioni:
  - Per ogni transazione di update si specificano:
    1. I file aggiornati dalla transazione.
    2. Il tipo di operazioni per ogni file (*lettura, modifica, cancellazione*).
    3. Gli attributi su cui sono specificate condizioni di cancellazione o di modifica.
    4. Gli attributi i cui valori sono modificati dalla transazione.

- Gli attributi identificati nel punto 3 sono candidati per la definizione di access path.
- Gli attributi identificati nel punto 4 **non** dovrebbero essere utilizzati in access path.

Fattori che influenzano il Design fisico (3)

- Analisi sulla frequenza di esecuzione di query e transazioni:
  - Si determina la frequenza con cui le query sono eseguite.
  - In genere vale la regola “80-20”: l’80% del processing è effettuato dal 20% delle query.
- Analisi sui vincoli temporali:
  - Alcune query e transazioni possono avere dei vincoli temporali che impongono priorità nella definizione di access path.

    Es: una transazione deve terminare entro 5 s. dalla sua invocazione.

    Questo vincolo fornisce una priorità maggiore a degli attributi che dovranno essere usati per access path.

Fattori che influenzano il Design fisico (4)

- Analisi sulla frequenza di operazioni di update:
  - Il numero di access path su file aggiornati frequentemente deve essere minimizzato, in quanto produce un overhead per aggiornare anche gli access path.

- Analisi dei vincoli di univocità degli attributi:
  - Andrebbero specificati degli access path per ogni chiave candidata.

Decisioni progettuali sugli indici

- Sebbene le performance di query migliorino fortemente in presenza di indici o schemi hash, le operazioni di inserimento, modifica e cancellazione sono rallentate dagli indici.

- Le decisioni sulle indicizzazioni ricadono in una delle cinque categorie seguenti:
  1. Quando indicizzare un attributo.
     Un attributo deve essere indicizzato se è chiave o se è utilizzato in una condizione di select (*uguaglianza o range di valori*) o join da una query.

Decisioni progettuali sugli indici (2)

2. Quale attributo indicizzare.
   Un indice può essere definito su uno o più attributi. Se più attributi sono coinvolti in varie query, è necessario definire un indice multi-attributo. L'ordine degli attributi nell'indice deve corrispondere a quello nella query.

3. Quando creare un indice clustered.
   Al più un indice per tabella può essere primario o clustering. Le query su range di valori si avvantaggiano di tali indici, mentre le query di ricerca su indici, che non restituiscono dati, non hanno miglioramenti con indici clustering.

Decisioni progettuali sugli indici (3)

4. Quando usare indici hash invece di indici ad albero.
   I DB in genere usano i B^+-Tree, utilizzabili sia con condizioni di uguaglianza sia con query su range di valori. Gli indici hash, invece, funzionano solo con condizioni di uguaglianza.

5. Quando utilizzare hashing dinamico.
   Con file di dimensioni molto variabili è consigliabile utilizzare tecniche di hashing dinamico (*non offerte dai DBMS più commercializzati*).

Denormalizzare uno schema

- Lo scopo della normalizzazione è di separare attributi in relazione logica, per minimizzare la ridondanza ed evitare le anomalie di aggiornamento.
- Tali concetti a volte possono essere sacrificati per ottenere delle performance migliori su alcuni tipi di query che occorrono frequentemente.
- Questo processo è detto denormalizzazione.

Denormalizzare uno schema (2)

- Il progettista aggiunge degli attributi ad uno schema per rispondere a delle query o a dei report per ridurre gli accessi a disco, evitando operazioni di join.

  Ad esempio, si potrebbe scegliere di denormalizzare uno schema di relazione da 4NF a 2NF, una forma più debole, ma che ridurrebbe gli accessi a disco in quanto eviterebbe la necessità di effettuare una operazione di join.