Basi di Dati

prof. Letizia Tanca

Le transazioni e il database server, cenni sui nuovi sistemi per Big Data

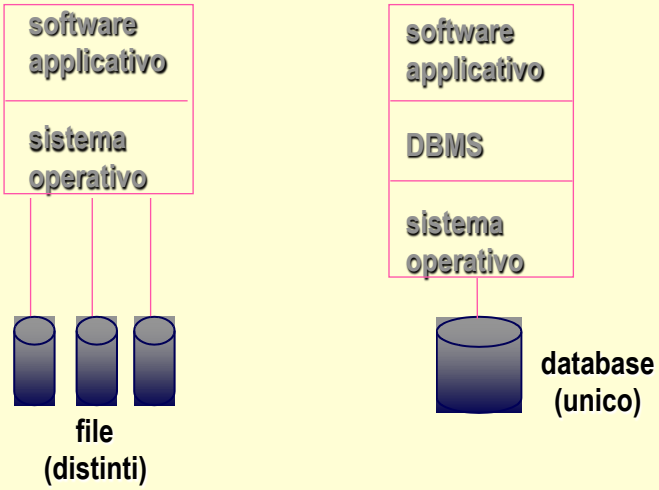
(lucidi parzialmente tratti dal libro:
Atzeni, Ceri, Paraboschi, Torlone "Introduzione alle Basi di dati", Mc Graw Hill Italia)

SGBD (DBMS) e file systems

Un **SGBD** (Data Base Management System = DBMS):

- permette di accedere in modo efficiente ai dati con una **granularità più fine che il file system**
- permette di accedere in modo diretto ai dati **basandosi sulle loro proprietà**
- effettua il controllo della concorrenza **alla granularità del singolo record**
- realizza meccanismi sofisticati per il **controllo dell'affidabilità**
- realizza meccanismi di **controllo della privatezza**
- realizza **l'atomicità delle transazioni**

File system e DBMS



Una Dase di Dati è un sistema OLTP

On Line Transaction Processing:
Elaborazione di transazioni, che realizzano i processi operativi dell'azienda-ente
- Operazioni spesso predefinite e relativamente semplici
- Ogni operazione coinvolge "pochi" dati
- Dati di dettaglio, aggiornati
- Le proprietà "acide" (Atomicità, Correttezza, Isolamento, Durabilità) delle transazioni sono essenziali

Le dimensioni delle basi di dati sono dell'ordine dei gbyte

La principale metrica di prestazione e' il *throughput delle transazioni* *(n. di transazioni al secondo)*

Sistemi OLAP

On Line Analytical Processing:

Elaborazione di operazioni per il supporto alle decisioni (*data warehouse*)

- Operazioni complesse e casuali
- Ogni operazione può coinvolgere molti dati
- Dati aggregati, storici, anche non attualissimi
- Le proprietà "acide" non sono rilevanti, perché le operazioni *sono di sola lettura*

Le dimensioni del warehouse raggiungono facilmente i terabyte

Le prestazioni considerate sono il throughput delle interrogazioni e il loro tempo di risposta

Transazione

- Una **transazione:**
  - Rappresenta la tipica unità di lavoro elementare di un Database Server
  - Identifica una unità di lavoro svolta da una applicazione

Transazione

- I DBMS classici (anche distribuiti) sono **sistemi transazionali:** mettono a disposizione un meccanismo per la definizione e l'esecuzione di transazioni
- Nell'esecuzione di una transazione devono essere garantite le proprietà ACIDe
- Sono stati proposti nuovi DBMS che **non sono sistemi transazionali**

Transazione

- *Unità elementare di lavoro effettuato da una applicazione, con proprietà fondamentali riguardanti la correttezza, il controllo di concorrenza e la robustezza.*
- Ogni transazione è incapsulata tra due comandi:
  - begin transaction (`bot`)
  - end transaction (`eot`)

Transazione

- In una transazione, necessariamente viene eseguito uno e uno solo dei due comandi seguenti:
  - `commit work` (commit)
  - `rollback work` (abort)
- Un sistema transazionale mette a disposizione delle applicazioni *meccanismi di definizione ed esecuzione delle transazioni*

Esempio astratto di transazione

```
begin transaction
x := x - 10
y := y + 10
commit work
end transaction
```

Transazione ben formata:

Una transazione che comincia con **`begin transaction`**, finisce con **`end transaction`**, nella cui esecuzione solo uno dei due comandi **`commit work`** o **`rollback work`** viene eseguito, e tale che, dopo l'esecuzione di uno dei due comandi **`commit work`** o **`rollback work`**, *nessuna altra operazione di aggiornamento sul database venga eseguita.*

Esempio più complesso (Atzeni)

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

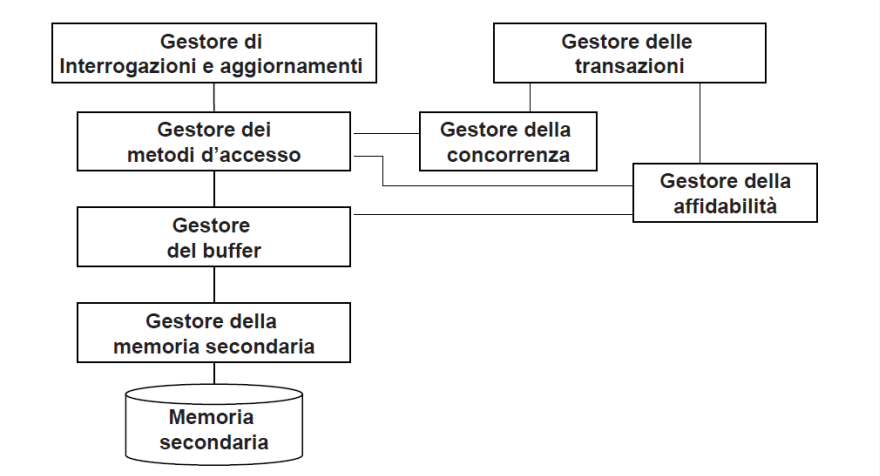

Proprietà ACIDe

- **Atomicità**: Una transazione è una unità indivisibile di esecuzione
- **Consistenza**: l'esecuzione di una transazione non deve violare i vincoli di integrità definiti sulla base di dati
- **Isolamento**: l'esecuzione di una transazione non è influenzata dalla esecuzione di altre transazioni concorrenti
- **Persistenza (Durability)**: gli effetti di una transazione andata a buon fine devono essere permanenti

Componenti di un database server transazionale

- **Ottimizzatore** - seleziona la strategia d'accesso ai dati
- **Access Methods Manager** - esegue la strategia
  - RSS (Relational Storage System)
  - OM (Object Manager)
- **Buffer Manager** - gestisce gli accessi alle pagine
- **Gestore di affidabilità** - gestisce i malfunzionamenti
- **Controllo di concorrenza** - gestisce le interferenze dovute alla concorrenza nell'accesso ai dati

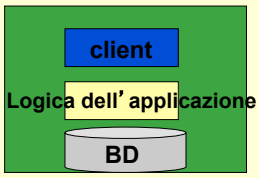
Gestore degli accessi e gestore delle transazioni (Atzeni)



Architettura e modello di esecuzione **single tier**

Mainframe + terminali intelligenti

- ***Pro:*** Facilmente gestibile da un amministratore centrale
- ***Contro:*** Interfaccia grafica richiede potenza di calcolo, sottratta alla gestione del BD



15

Architetture distribuite

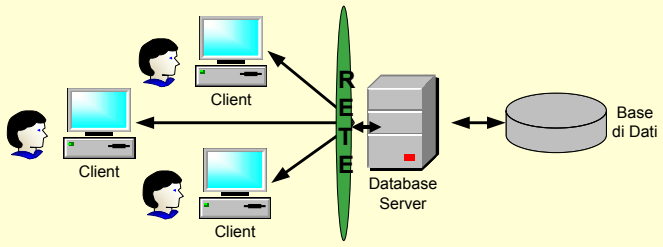
- Quasi tutte le basi di dati hanno **un' architettura distribuita** (caratterizzata dalla presenza di una rete)
- **Architettura client-server**
  - E' la più semplice e diffusa
  - **Server**: elaboratore per la gestione dei dati
  - **Client**: elaboratore per la gestione dell' applicazione e del dialogo con l' utente
  - **Rete**: collega i client con il server
- **Architettura distribuita**
  - Almeno **DUE server** che, oltre a lavorare autonomamente, in alcuni casi devono **interagire**

Architettura client-server (1)

- Più in generale, modello di interazione *tra processi software* in cui i processi si dividono in Client e Server
- **Processi Client**
  - Richiedono i servizi
  - Dedicati a interagire con l'utente finale
  - Ruolo attivo: genera richieste
- **Processi Server**
  - Offrono i servizi
  - Ruolo reattivo: si limita a rispondere alle richieste dei diversi client
- **Nota** Si parla di *processi*, generalmente i processi client e server risiedono su macchine diverse, collegate via rete
- L'interazione fra client e server richiede una interfaccia di servizi: elenco dei servizi messi a disposizione dal server

Architettura client-server (2)

- Un processo client può richiedere alcuni (pochi) servizi a vari processi server (se sono più di uno è una architettura distribuita)
- Ogni processo server risponde a (molte) richieste da parte di molti processi client gestendo in modo opportuno le relative transazioni



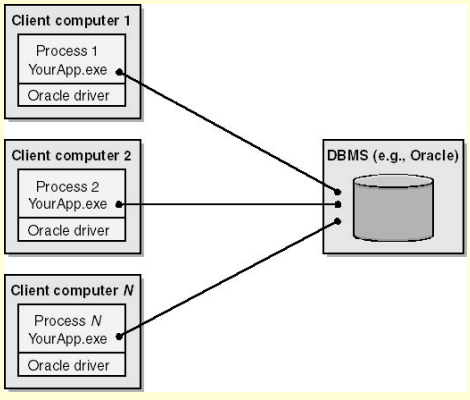
Motivi uso Client-Server

- Le funzioni di client e server sono ben definite
  - C'è una corrispondenza con gli utenti
    - Programmatore applicativo: ha la responsabilità di gestire il software relativo al client
    - Amministratore: Deve organizzare la base dati sul server per garantire prestazioni ottimali a tutti i client
- Diverse esigenze di hardware
  - Il client è un elaboratore adatto alla interazione con l'utente
    - Strumenti di produttività
    - Applicazioni "amichevoli" che accedono alla base dati
    - Grafica: non possiamo più usare terminali non intelligenti
  - Il server è dimensionato in base ai servizi che deve offrire e al carico transazionale (grande memoria centrale, grande memoria di massa, ...)

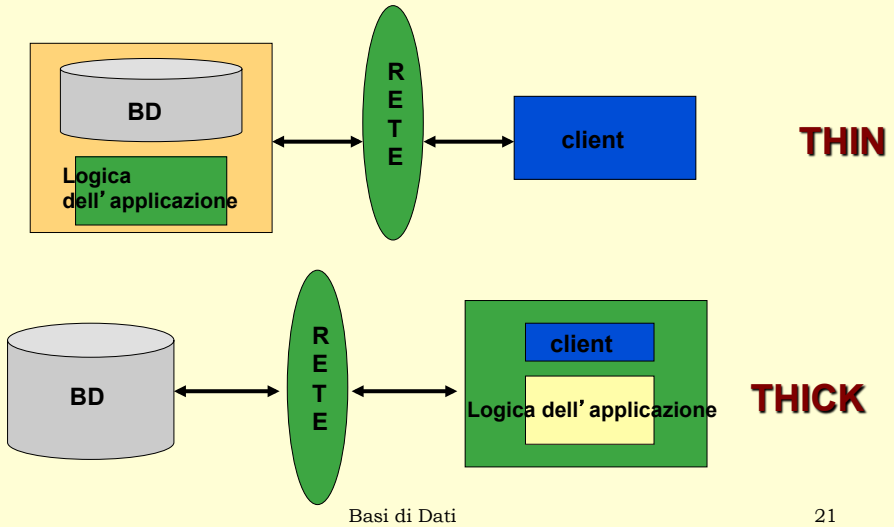
Architettura a 2 livelli

(two tier)

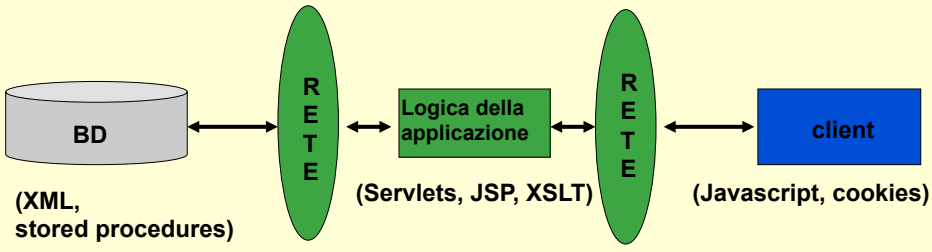
- Il client ha funzioni solo di interfaccia o anche di gestione dell'applicazione
- Il server ha funzione di gestione dei dati
- Più diffusa l'architettura *thin client*



Thin client e Thick client



Architetture three tier



NoSQL databases

- Ci si è accorti che non sempre è necessario che un sistema di gestione dei dati garantisca tutte le caratteristiche transazionali
- I DBMS che non sono transazionali vengono comunemente chiamati NoSQL
- Questo in realtà non è corretto perché il fatto che un sistema sia relazionale (e adoperi il linguaggio SQL) e il fatto che sia transazionale sono caratteristiche indipendenti

BIG DATA e Cloud

DATA CLOUDS:

- **SERVIZI di storage on demand**, affidabili, offerti su Internet con facile accesso a una quantità virtualmente infinita di risorse di memorizzazione, calcolo e rete

CLASSIFICAZIONE DEI POSSIBILI METODI DI STORAGE:

- **Centralizzati o distribuiti, transazionali, basati su un modello tradizionale (es. relazionale)** restano i più utilizzati per le applicazioni gestionali classiche (sistemi informativi aziendali etc.)
- **Federati e multi-database** per e in generale aziende e realtà che si associano e condividono i loro dati su Internet
- **Cloud databases** per supportare BIG DATA mediante tecniche di load sharing e data partitioning

BIG DATA e Cloud

I database NoSQL:

- prevedono schemi flessibili
- scalano orizzontalmente
- NON supportano tutte le proprietà acide
  - Gli aggiornamenti vengono effettuati in modo asincrono (NON VI E' supporto esplicito della concorrenza)
  - Inconsistenze potenziali nei dati vengono risolte direttamente dagli utenti

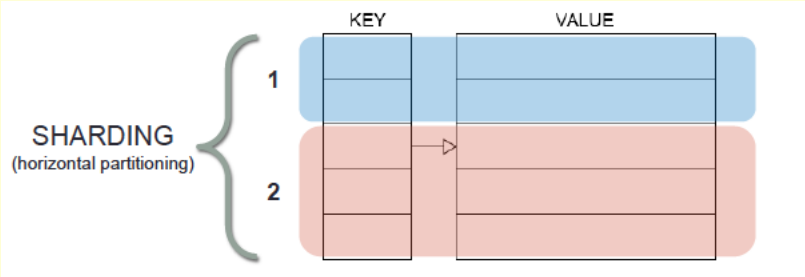
I modelli dei dati

3 categorie:

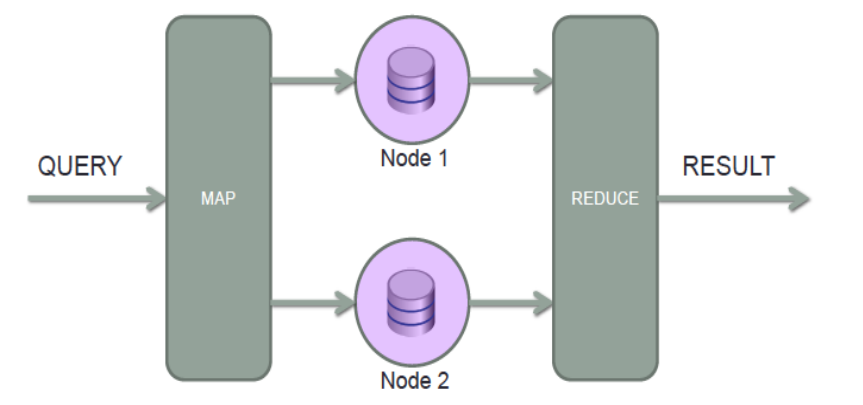
- Key –Value
- Document –based
- Column –family
- Basati su grafi

Key –Value

- Modello di riferimenti classico per I sistemi NoSQL
- Chiave: singola o composta (es.id e data)
- Valore: blob non interrogabile
- Query: ricerca per chiave
- No schema (un catalogo)
- Standard API : get / put / delete

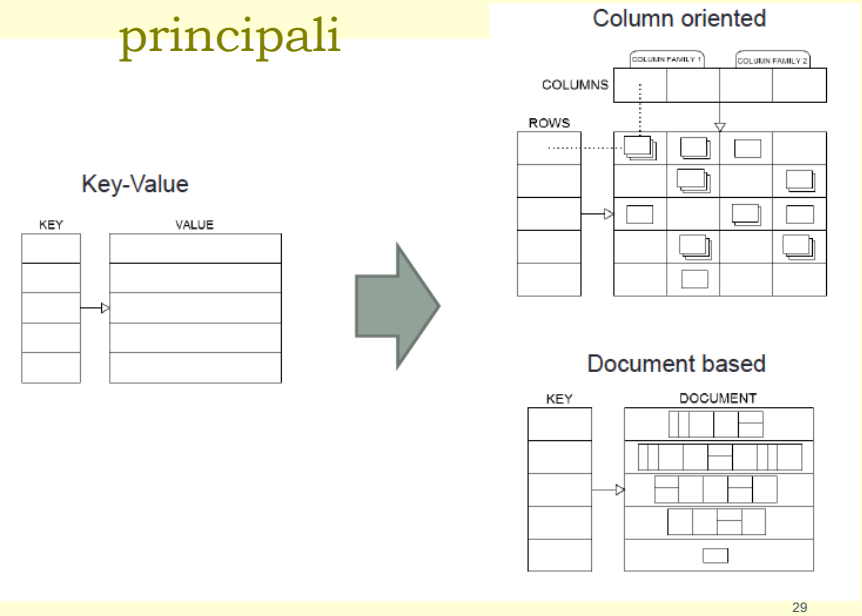


Map Reduce



28

Altri modelli principali



29

Il CAP theorem

- Un sistema di gestione di dati condiviso in rete (cloud, networked shared-data system) verifica alpiù due delle seguenti proprietà:
  - consistenza (C)
  - alta disponibilità (availability → A)
  - tolleranza alle partizioni della rete (Partitions →P)
- Da ciò si capisce perché nelle applicazioni più tradizionali, come per esempio quelle bancarie o contabili, o le prenotazioni etc. questi sistemi possano essere disastrosi
- Applicazioni interessanti:
  - Raccolta dati provenienti da sensori (append-only)
  - Datasets aggiornati poco frequentemente in generale
  - OLAP

Distributed I.S. 30

BIBLIOGRAFIA

LIBRO :

Tamer Ötzsu M., Valduriez P. – Principles of *Distributed Database Systems: 3rd ed.* - Springer, 2011

ALTRO MATERIALE DI RIFERIMENTO

•Abadi Daniel J. - *Data Management in the Cloud: Limitations and Opportunities* - IEEE Data Engineering Bulletin, Vol. 32 No. 1, March 2009
http://sites.computer.org/debull/A09mar/A09MAR-CD.pdf#page=5

•Dean J., Ghemawhat S. – *MapReduce: A Flexible Data Processing Tool* - CACM, Vol.53, n. 1, pp. 72-77, 2010

•Foster I., Yong Zhao, Raicu I., Lu S - *Cloud Computing and Grid Computing 360-Degree Compared* - Grid Computing Environments Workshop 2008, pp. 1-10, 2008
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4738445