Recupero -----

Viste

Significato ed uso di una vista:
una vista è rappresentata da una query (SELECT), il cui risultato può essere utilizzato come se fosse una tabella

create view *NomeVista* [(*ListaAttributi*)] as <*SelectSQL* >
[with [local | cascaded] check option]

Operazioni effettuabili su una vista:

- select
- update (con forti limitazioni – non sempre implementato)

Importanza progettuale delle viste:

- modello dei dati esterni
- indipendenza (parziale) dalle modifiche
- semplificazione concettuale delle interrogazioni

Definite anche le viste ricorsive in SQL 1999 ma non sono state implementate quasi da nessun DBMS

Viste materializzate

Alcuni DBMS supportano le viste materializzate (materialized views).
Si tratta di viste i cui risultati vengono scritti all'interno del DB, dati che vengono aggiornati automaticamente a intervalli regolari.
Permettono di fare interrogazioni più veloci su dati che però rischiano di non essere "up to date".

Viste aggiornabili

I dati di una vista possono essere manipolati e soprattutto AGGIORNATI.
Non tutti i DBMS supportano questa possibilità.
I dati scritti sulla vista vengono inseriti nelle tabelle su cui è definita la vista stessa.
Per fare ciò sono necessari controlli molto complessi.

Non tutte le viste possono essere scrivibili.
Non sono scrivibili le viste che utilizzano funzioni di aggregazione (con una clausola GROUP BY), perché i dati risultanti non sono in una tabella, ma rielaborati durante la query.

Controllo dell'accesso

CHI – COME – COSA

Il sistema del controllo si basa su permessi o "privilegi".

grant < *Privileges* | all privileges > on *Resource* to *Users* [with grant option]

dove i *Privileges* possono essere:

- insert: inserire nuovi record
- update: modificare il contenuto
- delete: eliminare record
- select: leggere i dati della risorsa
- references: definizione di vincoli di integrità referenziale verso questa risorsa
- usage: utilizzo in una definizione

Utilizzo di viste e permessi sulla stessa per filtrare i dati di una relazione e renderli solo parzialmente visibili ad un utente.

Nelle implementazioni più avanzate dell'SQL (non mySQL) esiste anche il concetto di ROLE (ruolo), a cui associare dei permessi.

Gli utenti vengono poi associati ai ruoli in modo da mantenere una coerenza di comportamento su un insieme di utenti.

Confinamento (o isolamento):

nei sistemi le politiche di sicurezza impongono un confinamento, cioè gli utenti possono operare solo su certe risorse e non su altre

AAA significa "authentication, authorization and accounting"

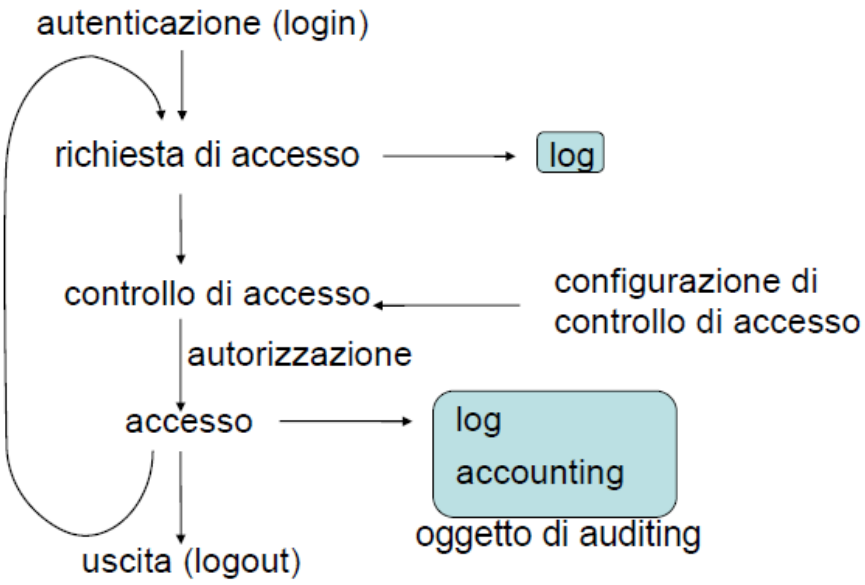
- autenticazione (authentication)
- controllo degli accessi (authorization)
- tracciamento del consumo delle risorse da parte degli utenti (accounting).

Definito in : "RFC 2903 : Generic AAA Architecture"

Cos'è un RFC (Request for Comments) ?

documenti e standard definiti dall'Internet Engineering Task Force (IETF)

Vedi RFC-1 ...



Tecniche di sicurezza

Le tecniche di protezione sono principalmente le seguenti:

- partizionamento in basi di dati monolivello (con relativi problemi di ridondanza e incongruenza dati);
- tecniche crittografiche per dati sensibili;
- integrità : utilizzo dei checksum (somme di controllo) che, calcolate in funzione del valore del dato, vengono memorizzate e ricalcolate a ogni accesso al dato. Se la somma risulta cambiata, vuol dire che il dato è stato alterato in maniera impropria (controllo di integrità);
- front end di sicurezza tra utente e DBMS;
- *views*, ovvero definizione da parte dell'amministratore del database (DBA) di sottoschemi (viste) della base di dati che mostrano solo i dati accessibili.

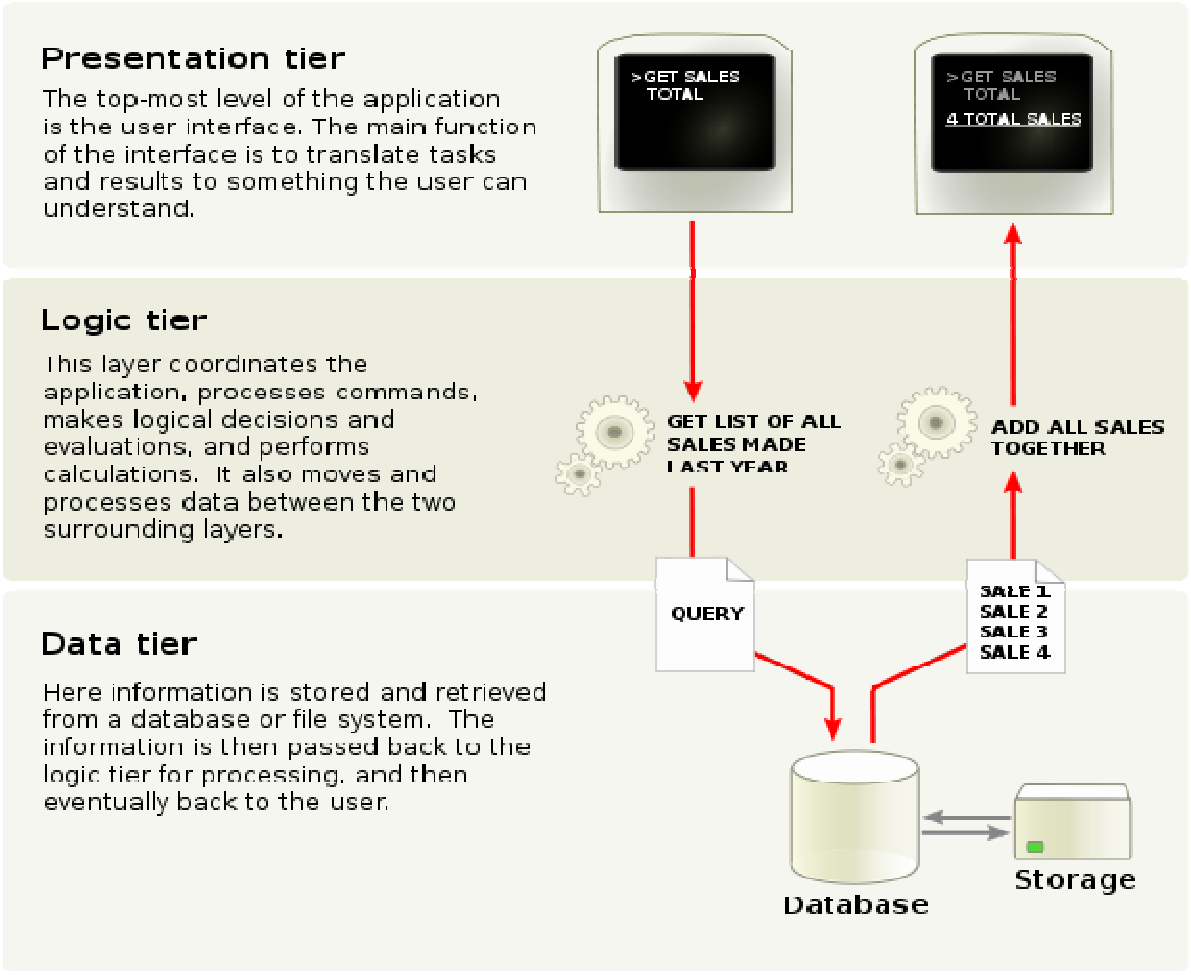
Elementi base per il controllo degli accessi ai DBMS

- Privileges
- Views
- Stored Procedures
- Roles
- Virtual Private Database

Sicurezza architetturale della basi di dati

Esistono più possibili soluzioni architetturali per la protezione dei dati di un DBMS, ad esempio per applicazioni WEB:

- architettura *two tier*, il server Web accede direttamente ai dati del DBMS;
- architettura *three tier*, in cui un application server fa da intermediario fra il server Web ed i dati del DBMS.



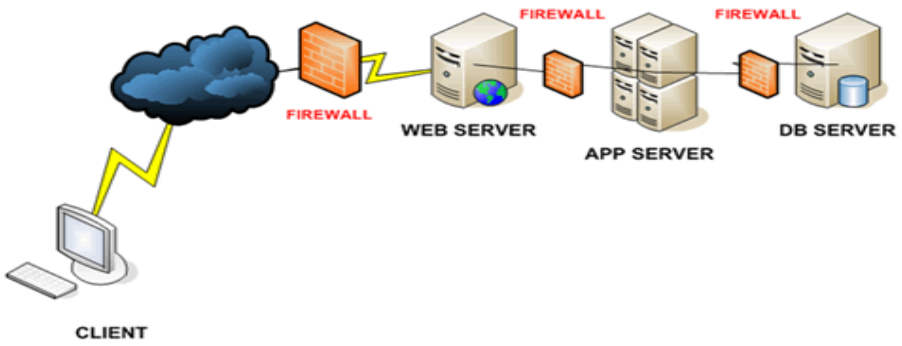
Strumenti per la sicurezza Architetturale

- Firewall
- DMZ (DeMilitarizedZone)

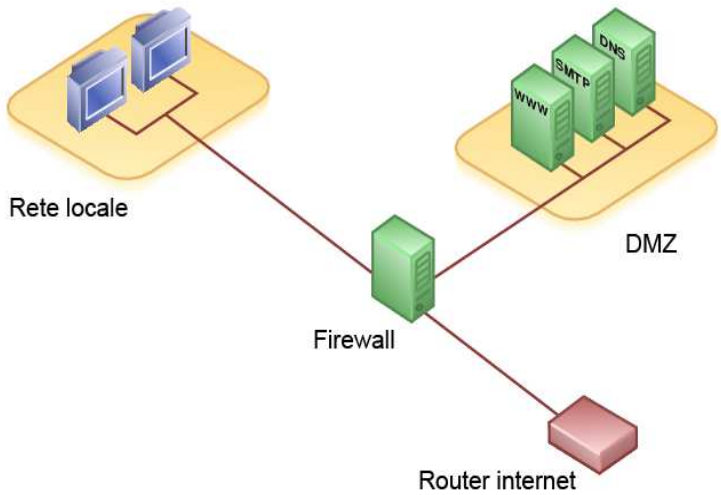
Firewall:

*Apparato di rete hardware o software che **filtra** tutti i flussi entranti ed uscenti, da e verso una rete o un computer, applicando regole che contribuiscono alla sicurezza della stessa.*

La stessa rete SICURIZZATA



Una **DMZ** (**demilitarized zone**) è un segmento isolato di LAN (una "sottorete") raggiungibile sia da reti interne che esterne che permette, però, connessioni esclusivamente *verso* l'esterno: gli host nella DMZ non possono connettersi alla rete aziendale interna.



Transazioni

Una transazione è un insieme di operazioni che devono essere eseguite in modo indivisibile.

Le proprietà dell'INDIVISIBILITA' sono (A.C.I.D.):

- Atomicità
- Consistenza
- Isolamento
- Persistenza (Durabilità)

Costrutti transazionali:

- start transaction (implicita)
- commit
- rollback
- autocommit

Transazioni distribuite: Two Phase Commit (2PC)

Il protocollo di aggiornamento a due fasi (2PC) è un algoritmo distribuito che prevede il coordinamento tra tutti i nodi in un sistema distribuito per convalidare una transazione.

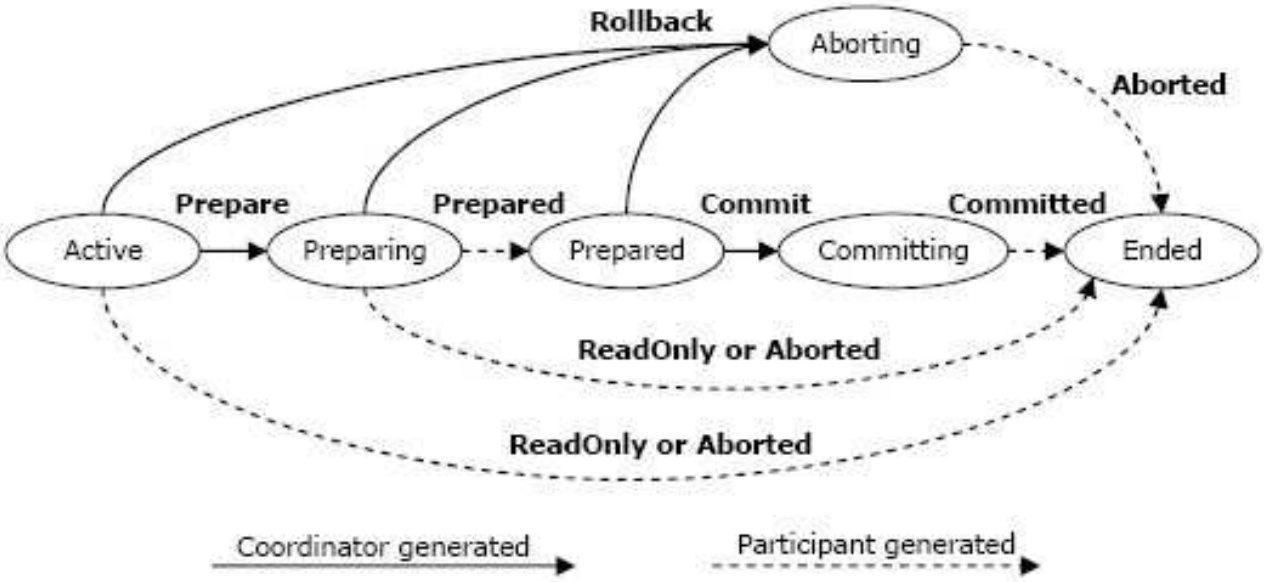
Le due fasi sono :
- **fase di richiesta di validazione**, nella quale il coordinatore centrale (dopo aver aperto ed eseguito la transazione su tutti i nodi) richiedere una convalida da ognuno di essi
- **fase di validazione**, il coordinatore, avendo ricevuto risposta positiva da ognuno dei nodi, completa la transazione confermandola a tutti i nodi coinvolti

Fase 1 (votazione):

1. Il coordinatore invia una richiesta canCommit? ad ogni partecipante alla transazione.
2. Quando un partecipante riceve una richiesta canCommit? risponde con il suo voto (Yes o No) al coordinatore. Prima di votare Yes, si prepara al commit salvando gli oggetti in memoria permanente. Se vota No il partecipante esegue la abort immediatamente.

Fase 2 (completamento dell'accordo per il voto):

1. Il coordinatore raccoglie i voti (incluso il proprio).
   (a) In assenza di guasti e se tutti i voti sono Yes il coordinatore decide di eseguire commit della transazione e invia una richiesta di doCommit ad ogni partecipante.
   (b) Altrimenti il coordinatore decide una abort della transazione e invia una richiesta di doAbort a tutti i partecipanti che hanno votato Yes.
2. I partecipanti che hanno votato Yes aspettano una richiesta di doCommit o doAbort dal coordinatore. Quando un partecipante riceve uno di questi messaggi esegue quanto richiesto e nel caso della commit, esegue una chiamata haveCommitted di conferma al coordinatore.


Rollback
Aborting
Aborted
Prepare
Prepared
Commit
Committed
Active
Preparing
Prepared
Committing
Ended
ReadOnly or Aborted
ReadOnly or Aborted
Coordinator generated
Participant generated

Stored procedure

Estendono l'SQL con un linguaggio computazionalmente completo.

Le procedure vengono memorizzate all'interno del DBMS.

Resta il problema dell'interazione con l'utente.

Definizione

Una stored procedure è un segmento di codice composto da istruzioni SQL dichiarative e procedurali, che viene memorizzato all'interno della base di dati stessa come parte dello schema, e che può essere invocato da un programma, da una regola attiva (in inglese “trigger”) o da un'altra stored procedure.

Dopo che la stored procedure è stata definita è quindi possibile utilizzarla come se rientrasse nell'insieme dei comandi SQL predefiniti, richiamandola e passandole eventualmente dei parametri.

Più precisamente la stored procedure viene memorizzata nel catalog del database, cioè nell'insieme delle tabelle in cui sono descritti tutti gli oggetti del database.

Pregi e difetti di una stored procedure

Il progettista di una base di dati può sfruttare i vantaggi delle stored procedure.

In modo particolare i pregi di una stored procedure sono legati:

- all'aumento delle prestazioni, poiché una stored procedure compilata e memorizzata nel catalog della base di dati viene eseguita più facilmente e velocemente rispetto a delle normali istruzioni SQL che devono essere interpretate di volta in volta.
- alla possibilità di creare regole di business legate ai dati e controlli d'integrità, grazie all'interazione coi trigger, che controllano la violazione dei vincoli, e le stored procedure, che permettono di mettere in atto un'adeguata azione. Il vantaggio dell'utilizzo di una stored procedure, in questi casi, consiste nel fatto che la procedura è memorizzata nel catalog, e può essere richiamata ogni qualvolta si riscontra la violazione per cui è stata progettata.
- alla riduzione del traffico tra l'applicazione che si appoggia al database e il server del database stesso, in quanto con una stored procedure è sufficiente mandare al server il nome della procedura e attendere il risultato. Al contrario se si utilizzano delle istruzioni SQL compilate direttamente dal client, il traffico di dati necessario per ottenere il risultato è maggiore, poiché i risultati delle singole istruzioni vengono tutti scambiati tra il server ed il client.
- al possibile riutilizzo di una stored procedure, in quanto è uno strumento che può essere richiamato da ogni applicazione che utilizza la base di dati. Infatti le stored procedure estendono l'interfaccia della base di dati per tutte le possibili applicazioni, permettendo allo sviluppatore del database di non dover programmare le funzioni che sono supportate dalla stored procedure in ogni possibile applicazione che dovrà interagire con la base di dati.
- alla sicurezza del database, poiché l'amministratore della base di dati può garantire il diritto di accedere alla stored procedure senza bisogno di concedere alcun privilegio sulle tabelle del database sottostante. Quindi le stored procedure si possono considerare come degli oggetti del database separati, con i loro privilegi.

Le stored procedure, pur essendo degli strumenti che offrono grandi potenzialità,

presentano dei difetti, che sono legati:

- al loro contenuto, in quanto una stored procedure contiene in prevalenza delle istruzioni di SQL dichiarative, che rendono di difficile scrittura una stored procedure con regole complesse, che al contrario sono più facilmente implementabili con altri linguaggi di programmazione, come per esempio Java e C. Esiste però la tendenza ad ampliare i linguaggi procedurali tipici di ogni DBMS con estensioni legate ai linguaggi di programmazione, in particolar modo a Java. Ciò avviene, per esempio, nel caso dei DBMS Oracle, in cui è possibile scrivere delle stored procedure estendendo il linguaggio PL/SQL con il linguaggio Java.
- alla mancata possibilità di effettuare un efficace debug all'interno del DBMS, cosa invece possibile con gli altri linguaggi di programmazione esterni al DBMS.
- al mantenimento, perché scrivere e mantenere efficiente nel tempo una stored procedure richiede delle conoscenze specifiche, che non tutti gli sviluppatori possiedono.

Per lo sviluppo di una stored procedure ci si avvale di un linguaggio procedurale, che varia da DBMS a DBMS.

L'American National Standards Institute (ANSI) nel 2003 ha definito un nuovo standard per il linguaggio SQL, denominato SQL:2003. Tra le funzionalità introdotte dal nuovo standard, vi rientra la definizione di un modulo per il linguaggio procedurale, denominata **SQL/PSM** (Persistent Stored Module).

Considerando che lo standard è nato nel 2003, solamente una parte dei DBMS sviluppati recentemente o che hanno introdotto la possibilità di sviluppare stored procedure negli ultimi anni, hanno implementato un linguaggio procedurale basato su SQL/PSM.

In pratica quindi ogni DBMS implementa un linguaggio procedurale differente

Triggers

Realizzano il paradigma (o Pattern) Evento-Condizione-Azione (ECA) [*Event-Condition-Action*].

Trigger quindi come azione da svolgere a fronte di un evento, se è valida una condizione.

Gli eventi possono essere:

- insert
- update (record o campo)
- delete

1 Trigger è associato (sensibile) ad 1 solo Evento.

L'azione può essere l'invocazione di una Stored Procedure !

Al trigger sono associati due elementi speciali: OLD e NEW, i record prima e dopo la modifica richiesta (nessun OLD per l'Insert, nessun NEW per il Delete).

I trigger possono permettere l'implementazione di vincoli d'integrità più sofisticati di quelli visti in precedenza.

Si possono realizzare catene di trigger, dove l'azione di uno causa lo scatenarsi di un altro trigger, e cosi' via, fino a generare situazioni estremamente complesse e col rischio di cicli infiniti (e conseguente blocco del sistema).

Un trigger si può definire come una parte di codice formata da istruzioni procedurali e dichiarative memorizzate nel catalog della base di dati, come avviene per le stored procedure, ma che non viene attivato tramite chiamata, bensì in seguito al verificarsi di un evento nella base di dati, quali ad esempio l'inserimento o la modifica di un record.

I trigger seguono il paradigma Evento-Condizione-Azione (ECA), secondo il quale il trigger viene attivato quando si verifica un determinato evento all'interno del database.

Se è soddisfatta una determinata condizione, allora il trigger esegue un'azione prestabilita. Un trigger è sensibile soltanto a un evento.

Solitamente l'obiettivo di un trigger è la determinazione delle reazioni che devono essere messe in atto al verificarsi di determinate eventi.

In relazione al paradigma ECA, l'evento può essere rappresentato da operazioni SQL di aggiornamento dello stato del database. Al contrario la condizione può essere un qualsiasi predicato SQL, e l'azione che deve essere svolta dal trigger è rappresentata da un comando SQL o dalla chiamata di una stored procedure.

Embedded SQL

Meccanismo

Inserire del codice SQL all'interno di un programma scritto in un linguaggio normale.

Il programma verrà PRECOMPILATO da un apposito sistema che traduce le istruzioni SQL in chiamate a librerie particolari che permettono l'interazione tra il programma stesso e il DBMS specifico.

Vi è quindi la necessità di definire delle sezioni e delle parole chiave che permettono al precompilatore di capire cosa deve fare.

Problemi:

- specificità rispetto al linguaggio, al DBMS, all'ambiente (S.O., networking, etc.)
- “conflitto d'impedenza”: programmi ragionano per record, SQL per insiemi

Statico

Le istruzioni SQL sono definite una volta per tutte, con al limite dei valori passati come parametri dal programma alle istruzioni SQL.

E' una soluzione che può avere ovviamente dei problemi di flessibilità, ma è molto performante perché il precompilatore può effettuare molte ottimizzazioni, ed inoltre molti errori possono essere scoperti in fase appunto di precompilazione.

Dinamico

Le istruzioni SQL sono costruite a run-time, ad esempio creando la stringa con pezzi diversi a seconda delle esigenze del momento.

Il precompilatore non può effettuare tutti i controlli e le ottimizzazioni del caso statico.

Se il programma non lavora bene possono verificarsi improvvisi errori dovuti a particolari situazioni di esecuzione.

Cursori

I cursori sono una soluzione per risolvere il problema del conflitto d'impedenza: permettono di scorrere (appunto cursore) lungo il set di dati ottenuti da una query.

Un cursore è associato quindi ad una select, e può essere impostato per essere mono o bidirezionale, per permettere o no aggiornamenti dei dati letti.

Fasi di vita di un cursore:

- DECLARE
- OPEN
- FETCH
- CLOSE

Se il cursore è mono-direzionale permetterà di leggere un record alla volta dal primo all'ultimo, attraverso il comando NEXT.

Invece se il cursore è bidirezionale permette di tornare indietro ma anche di saltare tra i vari record:

NEXT, PRIOR, FIRST, LAST, ABSOLUTE, RELATIVA

Call Level Interface (CLI)

Meccanismo

Definizione di una libreria (INTERFACE) che permette di effettuare delle chiamate (CAL) al DBMS tramite comandi SQL.

Questa libreria può essere più o meno generica, astratta rispetto a:

DBMS

Sistema Operativo

Networking

Se viene definito uno Standard, allora più sviluppatori, più società possono fare dei prodotti che sono tra loro INTERCAMBIABILI, cioè hanno la stessa INTERFACCIA.

Se il linguaggio ha poi un meccanismo per collegare dinamicamente diverse librerie che presentano la stessa interfaccia a seconda di opportuni parametri di configurazione allora i programmi avranno una portabilità elevatissima.

Esempio: JDBC

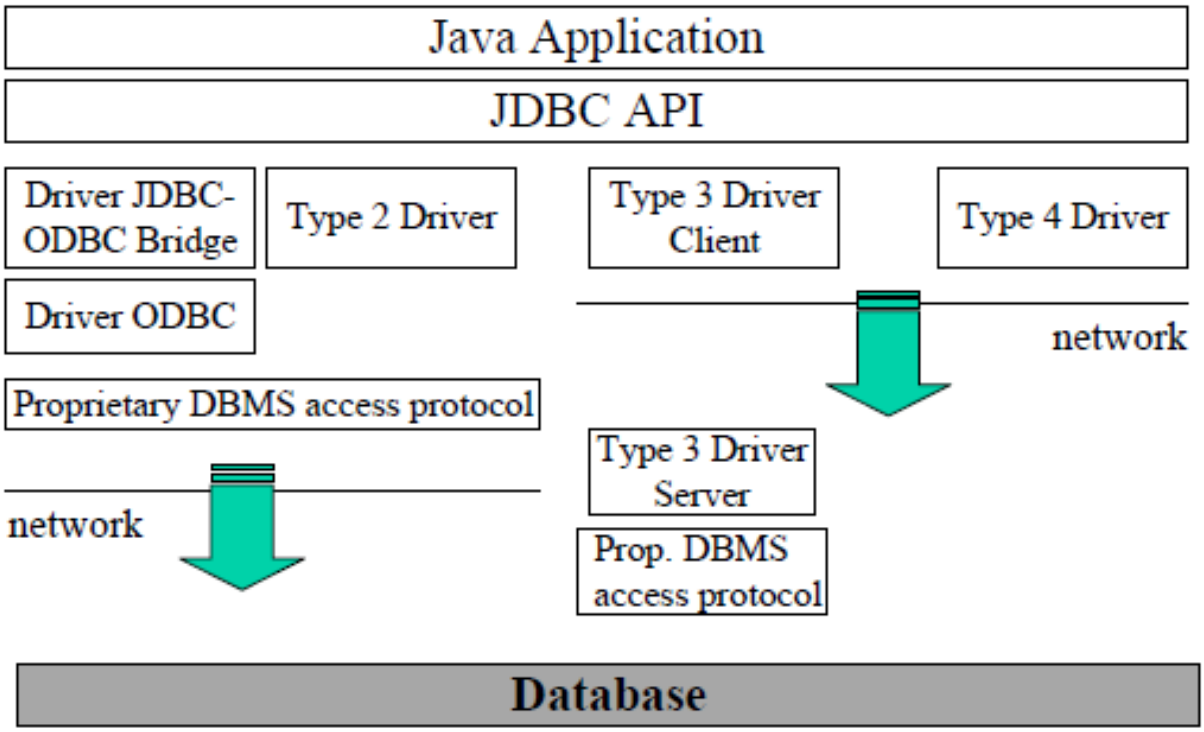
Elementi di un sistema che utilizza JDBC:

- applicazione Java
- JDBC Driver Manager
- Driver / Bridge / Middleware
- eventualmente Native Driver oppure ODBC
- DBMS

Struttura tipica di un programma che utilizza JDBC

- caricamento del driver (eventualmente sulla base di opportuni parametri di configurazione)
- apertura della connessione (con eventuali userId e password)
- creazione del contesto esecutivo (statement)
- esecuzione tramite lo statement delle istruzioni SQL (statiche o dinamiche)
- eventuale lettura dei dati attraverso oggetti di tipo ResultSet
- chiusura dello statement e della connessione

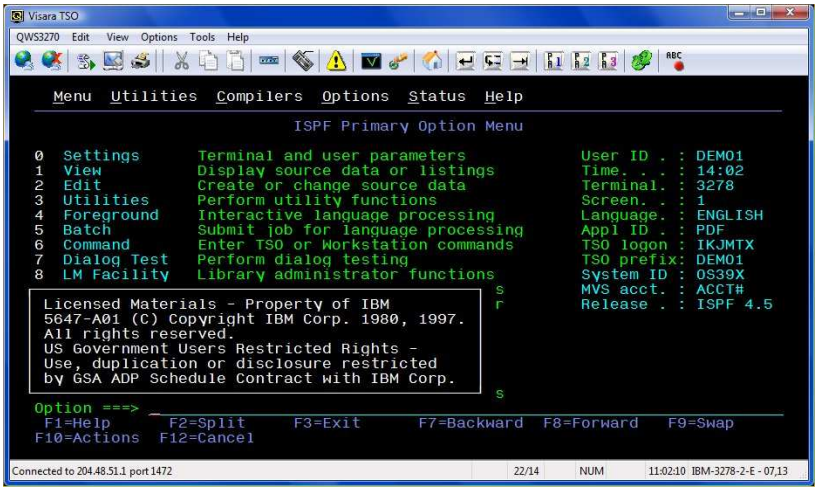
Architettura dei tipi di driver JDBC



Altri argomenti:

Architetture applicative

- MainFrame Batch
- MainFrame CICS (3270)
- Distribuito puro (PC non integrati)
- Client – Server
- Multi-Tiers
- SOA



Web Services

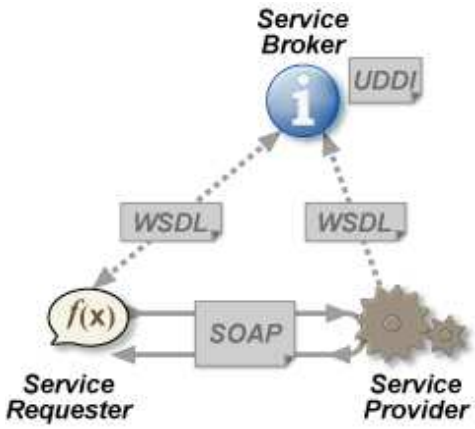
Definiti da World Wide Web Consortium (W3C)

Un Web Service (servizio web) è un sistema software progettato per realizzare l'interoperabilità tra diversi sistemi collegati da una medesima rete (sia locale che “Wide”).

Elemento fondamentale di un Web Service è quella di offrire un'interfaccia software (descritta in un formato standard (WSDL)) utilizzando la quale altri sistemi possono interagire con il Web Service stesso, tramite le funzioni descritte nell'interfaccia stessa, utilizzando "messaggi" racchiusi in una "busta".

Tali messaggi sono trasportati tramite il protocollo HTTP e scritti in formato XML.

- Web Service
- Interfaccia
- Messaggi
- SOAP
- HTTP
- XML

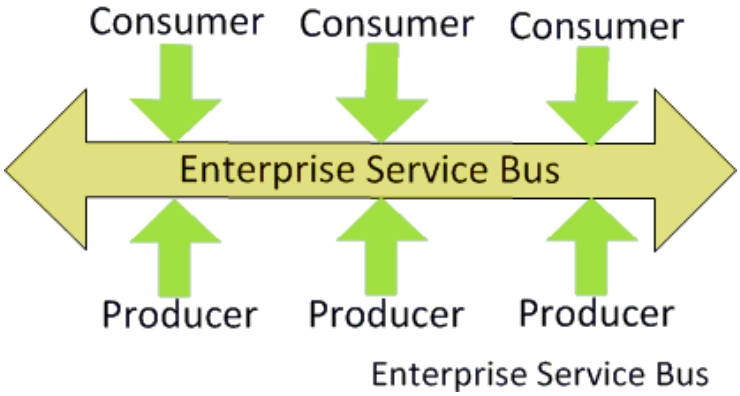


SOA

Service-Oriented Architecture (SOA) indica generalmente un'architettura software adatta a supportare l'uso di servizi Web per garantire l'interoperabilità tra diversi sistemi così da consentire l'utilizzo delle singole applicazioni come componenti del processo di business e soddisfare le richieste degli utenti in modo integrato e trasparente.

ESB

Un Enterprise Service Bus (ESB) è un'infrastruttura software che fornisce servizi di supporto ad architetture SOA complesse. Un ESB si basa su sistemi disparati, interconnessi con tecnologie eterogenee, e fornisce in maniera consistente servizi di orchestration, sicurezza, messaggistica, routing intelligente e trasformazioni, agendo come una dorsale attraverso la quale viaggiano servizi software e componenti applicativi.



Enterprise Service Bus

Clustering

Un cluster (dall'inglese grappolo), è un insieme di computer connessi tramite una rete telematica.

Lo scopo di un cluster è quello di distribuire una elaborazione molto complessa o molto pesante tra i vari computer componenti il cluster.
L'importanza dei cluster è che l'intero cluster può essere visto dalle applicazioni client come un unico computer, non come la somma di tanti tra loro distinti.

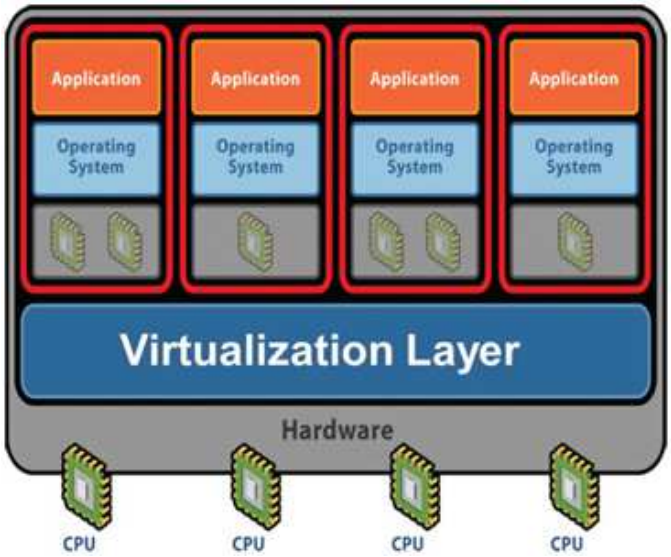
Esistono tre tipi di cluster:

- Fail-over
- Load balancing
- High Performance Computing

Virtualizzazione

Con il termine "virtualizzazione" si indica la possibilità di eseguire contemporaneamente più sistemi operativi virtuali su una singola macchina fisica, mantenendoli però, dal punto di vista logico, distinti.
Il sistema operativo "ospitante" (l'host) crea di fatto una sorta di hardware virtuale all'interno del quale vengono eseguiti i diversi sistemi operativi "ospiti" (i guest).

Di fatto la parte inferiore dello stack software è un sistema operativo ordinario che è installato direttamente sul server.
Sopra di questo, un layer di virtualizzazione gestisce il reindirizzamento e l'emulazione delle risorse fisiche rimappandole verso quelle virtuali.
Nei server virtuali sono installati dei sistemi operativi, ovvero i guest, i quali penseranno di avere tutta una macchina per sé, ignorando il fatto di essere in una macchina virtuale e quindi anche l'esistenza degli altri guest eseguiti sullo stesso hardware.

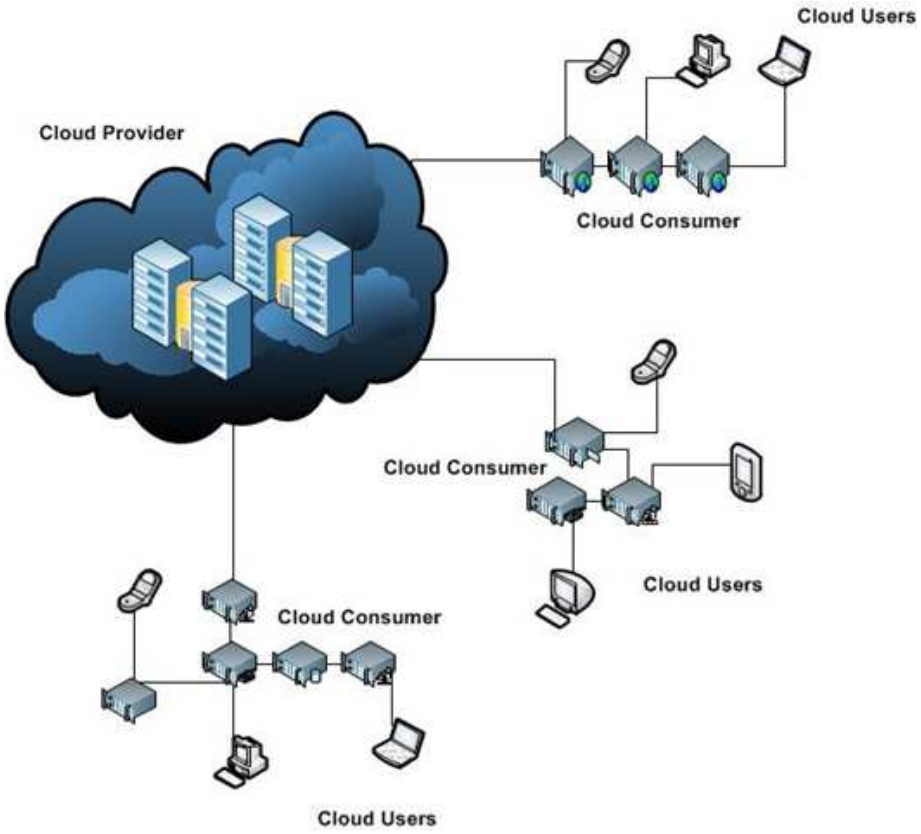


Cloud computing

Cloud computing has been defined as “a model for enabling convenient, **on-demand** network access to a **shared pool** of configurable **computing resources** (e.g., networks, servers, storage, applications, and services) that can be **rapidly provisioned and released** with minimal management effort or service provider interaction.”

The following player categories are involved in a cloud computing ecosystem:

- **Cloud Providers.**
  They provide the hosting platform and cloud infrastructure services.
- **Cloud Consumers.**
  They utilize the cloud platform and create applications and services for the users.
- **Cloud Users.**
  They use the applications and services provided by the cloud consumers.



Cloud Characteristics

On demand-self-service is one of the cloud characteristics. It enables a consumer to provision computing resources as needed automatically, without intervention on the part of the cloud solution provider. From now on we will use the term cloud to mean cloud computing.

Also a **broad network access** allows for computing resources to be available through standard mechanisms such as HTTP and SOAP protocols. This is to enable access by a variety of clients such as laptops, mobile phones and other devices.

Another important characteristic is **resource pooling**. The cloud resources are pooled to serve multiple consumers using a multi-tenant model where resources are assigned based on consumer demand.