



UNIVERSITÀ DEGLI STUDI DI  
PARMA  
Facoltà di SCIENZE  
MATEMATICHE, FISICHE e NATURALI  
Corso di Laurea in Informatica

**SINCRONIZZAZIONE TRA BASI DI  
DATI CON TECNOLOGIA ORACLE  
10g**

Tesi di Laurea Triennale

Candidato:  
**Luca Faggiani**

Relatore:  
**Prof. Ing. Giulio Destri**

Correlatore:  
**Ing. Alberto Picca**

Anno Accademico 2006/2007

*Ai miei genitori,  
Sante e Marzia.*

## Ringraziamenti

Mi sembra doveroso, giunto a questo significativo momento della mia carriera scolastica, dedicare qualche parola alle persone che hanno contribuito più o meno direttamente al raggiungimento della laurea.

Il primo ringraziamento va quindi ai miei genitori, Sante e Marzia, e mia sorella Annalisa, che in tutti questi anni hanno saputo sostenermi, permettendomi di portare a termine il corso di studi senza mai farmi mancare nulla.

Un sentito ringraziamento va anche al relatore, il Prof. Ing. Giulio Destri, che nonostante i tanti impegni quotidiani è riuscito comunque a seguirmi durante il periodo di tirocinio e la stesura della tesi con grande disponibilità.

Ringrazio anche il correlatore, l'Ing. Alberto Picca, per il supporto prestatomi durante il periodo di tirocinio presso l'azienda Area Solution Providers.

Mi sembra opportuno citare anche i compagni e amici di università, Chiara, Rino, Alessandro e tutti gli altri che mi hanno permesso di ambientarmi in questo ateneo nel migliore dei modi.

Naturalmente ringrazio anche gli amici e i compagni di casa che in questi anni mi hanno accompagnato nelle tante esperienze fatte, Daniel, Jacopo, Luca, Marco, Matteo, Giovanni e tutti gli altri.

# Indice

Capitolo 1 – Introduzione . . . . .	6
1.1 – Applicazioni e Database. . . . .	6
1.2 – Comunicazione tra Client e Server. . . . .	9
1.3 – I database Server. . . . .	11
1.4 – L’obiettivo della tesi: sincronizzazione tra database . . . . .	12
Capitolo 2 – I database relazionali ed il loro ruolo . . . . .	14
2.1 – Il database relazionale . . . . .	15
2.2 – I database server presenti sul mercato . . . . .	18
2.3 – Il database server entro i sistemi informatici e la struttura database- centrica. . . . .	19
2.4 – Le situazioni pratiche di comunicazione tra DB Server. . . . .	22
2.5 – Gli strumenti di comunicazione. . . . .	23
Capitolo 3 – Il database server Oracle 10g . . . . .	25
3.1 – Struttura di Oracle 10g. . . . .	25
3.2 – Versioni di Oracle 10g presenti e loro caratteristiche . . . . .	37
3.3 – Oracle server ed i componenti ausiliari . . . . .	39
3.4 – Gli strumenti di sincronizzazione di Oracle 10g . . . . .	41
Capitolo 4 – Il problema proposto: sincronizzazione tra Database centrale e database remoto. . . . .	48
4.1 – Il contesto del problema. . . . .	48
4.2 – Applicazioni pratiche del problema . . . . .	49
4.3 – Lo schema di esempio: le operazioni da compiere. . . . .	49
Capitolo 5 – Uso del DB Link. . . . .	57
5.1 – Caratteristiche del DB Link . . . . .	57
5.2 – Applicazione al problema. . . . .	59
5.3 – Esiti dell’azione . . . . .	60
Capitolo 6 – Uso degli Oracle Stream. . . . .	62
6.1 – Caratteristiche degli Oracle Stream . . . . .	62
6.2 – Applicazione al problema. . . . .	64
6.3 – Esiti dell’azione: limiti di applicabilità . . . . .	74

Capitolo 7 – Uso di Oracle Grid . . . . .	75
7.1 – Caratteristiche di Oracle Grid. . . . .	75
7.2 – Applicazione al problema ed esiti dell’azione . . . . .	78
Capitolo 8 – Uso di Oracle SQL Developer . . . . .	80
8.1 – Applicazione al problema. . . . .	80
8.2 – Esiti e limiti emersi. . . . .	84
Capitolo 9 – Conclusioni. . . . .	85
9.1 – Analisi del lavoro svolto e confronto tra i metodi . . . . .	85
9.2 – Possibilità di estensioni future . . . . .	86
Bibliografia . . . . .	87

# Capitolo1

## Introduzione

La presente tesi di Laurea Triennale è stata sviluppata dal candidato dopo un periodo di tirocinio svolto presso l'azienda informatica Area Solutions Providers S.r.l., nella sede operativa Lombardia est ed Emilia, sotto il coordinamento dell'Ing. Alberto Picca, e con la supervisione del Prof. Ing. Giulio Destri.

Durante tale periodo di tirocinio ho acquisito i fondamentali della tecnologia Oracle, frequentando anche un corso di due giorni presso la Oracle Corporation Italia a Milano, durante i quali sono state illustrate le principali caratteristiche dei database Oracle 10g e 11g.

Una volta acquisite i concetti principali sulle architetture, ho affrontato la questione relativa alla sincronizzazione tra basi di dati, argomento principale della tesi, individuando ed analizzando le varie possibilità implementative che la tecnologia Oracle mette a disposizione nei propri database.

Attraverso tale analisi ho potuto, in una seconda fase, realizzare le soluzioni più convenienti in termini di praticità, sottolineando per ognuna di queste gli aspetti principali, passando così dal modello teorico del problema ad un modello reale.

Gli strumenti utilizzati per l'implementazione pratica sono stati i database Oracle 10g nelle versioni *Standard Edition* ed *Express Edition*.

Nei primi capitoli di questa tesi vengono introdotti i concetti più teorici, legati all'introduzione delle nozioni base delle basi di dati e dell'architettura Oracle, mentre nei capitoli seguenti vengono descritti gli esperimenti pratici da me eseguiti e mostrati i risultati sperimentali ottenuti durante i test sulle soluzioni individuate.

## 1. Applicazioni e Database

Nel contesto aziendale di oggi, il **sistema informativo** riveste un ruolo di fondamentale importanza, perché una buona implementazione di questo può rappresentare un potenziale vantaggio rispetto alla concorrenza.

Il sistema informativo può essere definito come "l'insieme di persone, apparecchiature, procedure aziendali il cui compito è quello di produrre e conservare le informazioni che servono per operare nell'impresa e gestirla"(M. De Marco in [De Marco 2000]).

Il sistema informativo quindi interagisce con tutti gli altri processi aziendali, ed ha come oggetto di trattazione l'**informazione**.

L'informazione è la principale risorsa scambiata, selezionata ed elaborata nelle attività gestionali di coordinamento e controllo (C. Ciborra si veda [Ciborra 2002]).

Essa è una risorsa immateriale e non tangibile, ma per esistere nel mondo fisico, deve essere rappresentata in modo fisico.

La rappresentazione dell'informazione in termini fisici è costituita dai **dati** [Destri 2007].

I Dati, che possono essere di natura digitale ma anche di diversa natura, come ad esempio quella cartacea, vengono intesi come i “fatti grezzi” a partire dai quali si possono dedurre informazioni utili alle significative decisioni aziendali.

Per essere utili i dati devono essere organizzati, archiviati e classificati in modo da essere facilmente reperibili.

Lo scopo principale di una base di dati (in inglese *database*) è proprio questo.

Un database è definito come un insieme strutturato di dati che, in un momento, fornisce una rappresentazione semplificata di una realtà in evoluzione.

Un database consente la gestione dei dati stessi (l’inserimento, la ricerca, la cancellazione ed il loro aggiornamento) da parte di applicazioni software.

Una base di dati, oltre ai dati veri e propri, deve contenere anche le informazioni sulle loro rappresentazioni e sulle relazioni che li legano.

Spesso la parola “database” è, impropriamente, usata come abbreviazione dell’espressione **Database Management System (DBMS)**, ma come vedremo i due concetti sono differenti [ACPT 2002] e [ACFPT 2003].

In un sistema informatico, la base di dati può essere manipolata direttamente da programmi applicativi (applicazioni), interfacciandosi direttamente con il sistema operativo, strategia adottata universalmente fino agli anni settanta, e tuttora impiegata quando i dati hanno una struttura molto semplice, o quando sono elaborati da un solo programma applicativo, oppure attraverso appositi sistemi software, detti **Sistemi per la gestione di basi di dati** (i DBMS appunto).

L’introduzione dei Database Management System, a partire dalla fine degli anni Sessanta, fu motivata dal fatto che la gestione di basi di dati complesse condivise da più applicazioni richiedeva l’utilizzo di un unico sistema software in grado di accedere in maniera efficace a grandi quantità di dati, e di gestire l’accesso garantendo l’integrità.

L’utilizzo dei database e la relativa gestione attraverso i DBMS rispetto alle applicazioni software possiede vantaggi sotto molti punti di vista;

Sotto il punto di vista della facilità di progettazione, l’utilizzo dei database permette:

- di creare diverse “visibilità” dei dati adatte a ciascuna applicazione;
- di gestire in maniera automatica e controllata i dati, come ad esempio in caso di cancellazione di un record da cui ne dipendono logicamente degli altri, porta anche la cancellazione di quelli dipendenti;
- di estendere i linguaggi esistenti o linguaggi ad alto livello, orientati a facilitare la manipolazione dei dati;

Sotto il punto di vista della facilità di gestione, il database permette:

- di modificare e arricchire le strutture di dati nel tempo senza la modifica delle applicazioni esistenti;
- di aggiungere nuovi campi all’interno di tracciati record già esistenti senza modificare le applicazioni che già li utilizzavano;
- di rendere esplicite le relazioni logiche tra dati, che altrimenti restano nascoste all’interno delle elaborazioni delle varie applicazioni;

Sotto il punto di vista dell’efficacia delle implementazioni il database permette:

- di ridurre la ridondanza dei dati, a differenza dell’utilizzo delle applicazioni in cui uno stesso dato può essere memorizzato su più di una di esse, e quindi una conseguente riduzione dell’occupazione fisica di memoria;
- di garantire una congruenza dei dati;

Sotto il punto di vista dell’utente il database garantisce:

- la disponibilità di un linguaggio per definire le caratteristiche dei dati e le strutture logiche che li legano;

- la disponibilità di estensioni dei linguaggi di programmazione che semplificano le operazioni sui dati;
- la disponibilità di accedere direttamente ai dati;

A causa di questi vantaggi, oggi, l'utilizzo dei DBMS per la gestione delle basi di dati, è molto più diffuso rispetto all'utilizzo di applicazioni specifiche, soprattutto in molti campi come la contabilità, le risorse umane, la finanza, la gestione di rete o la telefonia.

La differenza più rilevante tra DBMS e applicazioni è che il DBMS è progettato per sistemi multi-utente. Per fare questo si appoggia a un kernel che supporta nativamente il multitasking e il collegamento in rete.

Una tipica applicazione per la gestione dei database non includerebbe, infatti, tali funzionalità, ma si appoggerebbe al sistema operativo per consentire all'utente di fruirne dei vantaggi.

La quasi totalità dei database presenti sul mercato, oggi, utilizzano il Database Management System come strumento per l'organizzazione, la memorizzazione e il reperimento dei dati.

Un DBMS garantisce anche la sicurezza e l'integrità del database.

Ogni transazione, infatti, ossia ogni sequenza di operazioni che hanno un effetto globale sul database gode delle cosiddette proprietà **ACID**:

- "A" sta per Atomicity (Atomicità), ossia se tutte le operazioni della sequenza terminano con successo si ha il *commit* e la transazione è conclusa con successo, ma se anche solo una di queste operazioni fallisce, l'intera transazione è abortita;
- "C" sta per Consistency (Consistenza), ossia ogni transazione, una volta terminata, deve lasciare il database in uno stato consistente;
- "I" sta per Isolation (Isolamento), ossia nel caso di esecuzione di transazioni concorrenti, non ci deve essere influenzamento dell'una sulle altre;
- "D" sta per Durability (Durabilità), ossia gli effetti sul database prodotti da una transazione terminata con successo sono permanenti, quindi non possono essere compromessi da eventuali malfunzionamenti;

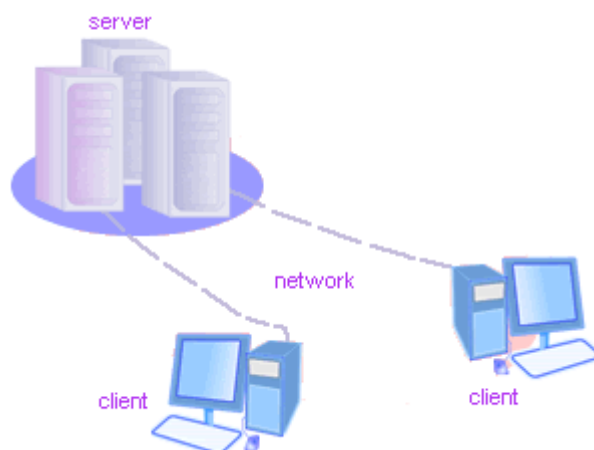
Quindi un database è una struttura che si evolve nel tempo per mezzo di transazioni, che partono sempre da uno stato consistente e terminano comunque sempre lasciando il database in uno stato consistente.

Nel capitolo 2 vedremo più in dettaglio le varie strutture che i DBMS possono avere, ed in particolare approfondiremo il concetto di Relational DBMS e del modello relazionale, un modello strutturale di dati molto diffuso nei database odierni.

## 2. Comunicazione fra client e server

Prima di trattare gli argomenti principali di questa tesi è utile dare una definizione formale del **modello client server** [Destri 2007] e [OHE 1999] in modo che risulti più chiaro il concetto di **database server**, di cui parleremo approfonditamente.

Il modello client server, oggi il modello di riferimento per il networking e le applicazioni di database, si contrappone come filosofia e come struttura al Mainframe. Nel modello di computing del Mainframe vi è un server centrale, detto appunto Mainframe, che esegue tutte le operazioni sui dati e usa terminali “stupidi” (senza capacità propria di elaborazione) come unità di visualizzazione e di inserimento dei dati.



*Fig.1.1*

Il sistema client server, invece, è un tipo di modello implementato da applicazioni di rete in cui una macchina client, con capacità di elaborazione propria, istanzia l'interfaccia utente (e non solo, come vedremo) dell'applicazione e si connette con una macchina server o ad un sistema di database per richiedere risorse in esso contenute.

La macchina server permette ad un certo numero di client di condividere le risorse, lasciando che sia il server a gestire gli accessi alle risorse per evitare conflitti tipici dei primi sistemi informatici.

Ad esempio il browser che richiede una determinata pagina web attraverso internet ad un server dove tale pagina è memorizzata, rappresenta un client che richiede una risorsa (la pagina web) ad un server.

Per introdurci nell'analisi del modello client server è utile dare una classificazione delle funzioni fondamentali che un generico applicativo svolge.

Queste possono essere suddivise in tre blocchi principali:

- *Presentazione*, ossia la visualizzazione tramite interfaccia grafica ed interazione con l'utente;
- *Applicazione*, ossia elaborazione dell'informazione;
- *Data Management*, ossia acquisizione e mantenimento dell'informazione;

Secondo il tipo di distribuzione di queste funzioni si possono individuare diversi livelli:

- Il primo livello che si può individuare è quello in cui tutta la parte computazionale è spostata sul server, mentre il client rappresenta solamente un display. Siamo praticamente in una situazione analoga a quella del Mainframe,

dove l'impiego di risorse cade tutto sull'elaboratore principale, mentre il client è semplicemente un terminale.

- Il secondo livello individuato è molto simile al primo, ma in questa situazione il server non implementa in alcun modo la funzione di presentazione; un esempio può essere un browser che interpreta il codice HTML presente sul server: è implicata quindi l'esistenza di una rete tra client e server.
- Il terzo livello individuabile, prevede lo spostamento di una parte di applicazione anche sul client, che quindi non potrà più avere un hardware troppo "thin", ma dovrà essere dotato di adeguate capacità di elaborazione e di caching dell'informazione. Un esempio per questa struttura è un browser dotato di linguaggio Java: la caratteristica di tale linguaggio è quella di fornire un applicativo (Applet java) in rete al client, il quale lo interpreterà e lo eseguirà usando la Java virtual machine che sta in memoria.
- Il quarto livello rappresenta il sistema client server "completo"; La particolarità di questa configurazione, venendo lasciata sul server solo la funzione di Data Management, è di produrre il completo spostamento dell'informazione da elaborare sul client. Una soluzione di questo tipo è stata resa possibile grazie ai progressi fatti a livello di Data Management (ODBC e driver nativi di acceso al database, linguaggio SQL, Visual Basic).
- Il quinto ed ultimo livello che si può individuare è il cosiddetto "fat client" che rappresenta un vero e proprio database distribuito; Parte dei dati, infatti, risiedono sul client che è capace di gestirli, modificarli e memorizzarli sia localmente che in remoto.

Secondo il tipo di applicazione di rete che si vuole sviluppare è opportuno scegliere il livello più appropriato.

Quando un client si connette direttamente ad un server o ad un database, si dice **2-tier architecture** (architettura a 2 livelli).

Recentemente però sono state sviluppate nuove architetture in cui il client, che implementa solamente la funzione di presentazione (thin client), si connette ad un server, che implementa la funzione di applicazione, il quale comunica transitivamente (ossia successivamente) con un database server, che implementa la funzione di Data Management, memorizzando i dati utilizzati dall'applicazione. Tale architettura è chiamata **3-tier architecture**.

Il collegamento tra client e server è implementato per mezzo di una rete di comunicazione utilizzando un determinato protocollo. Tale protocollo può essere in chiaro, e quindi i dati in transito possono essere intercettati, oppure in certi casi crittografato, proteggendo i dati in transito dalle intercettazioni.

Solitamente due macchine possono comunicare tra loro rispettando norme che sono determinate da protocolli di rete.

L'aderenza ai protocolli di rete garantisce che le due macchine possano comunicare correttamente, anche se sono state realizzate indipendentemente.

Si possono individuare due tipi di comunicazione tra client e server:

- **comunicazione orientata alla connessione**
- **comunicazione senza connessione**

Nella comunicazione orientata alla connessione il client e il server attuano il cosiddetto *handshaking*, ossia una serie di procedure in cui le due macchine si scambiano pacchetti di controllo prima di spedire i dati per prepararsi alla comunicazione. La comunicazione orientata alla connessione è utilizzata, tra gli altri, dal protocollo di rete **TCP** (Transmission Control Protocol).

La comunicazione senza connessione, invece, non prevede nessun handshaking, e i dati vengono direttamente inviati sulla rete. Questo tipo di comunicazione è utilizzato, tra gli altri, dal protocollo di rete **UDP** (User Datagram Protocol).

La comunicazione orientata alla connessione è più lenta rispetto a quella senza connessione, a causa dei pacchetti di handshaking, ma risulta più affidabile in caso di perdita nella rete di dati o errori.

In definitiva si può dire che la quasi totalità delle applicazioni distribuite oggi è implementata seguendo il modello client server.

### 3. I database server

I *database server* sono macchine ottimizzate per ospitare i programmi che costituiscono il database reale e sulle quali girano solo il DBMS e il software ad esso correlato (nelle situazioni reali spesso queste macchine svolgono anche funzioni non correlate con la gestione del database) [ACFPT 2003].

I database server si occupano di fornire i servizi di utilizzo del database ad altri programmi ed ad altre macchine secondo la modalità client server.

Il server ha il compito di memorizzare i dati, ricevere le richieste dei client ed elaborare le risposte appropriate.

Essi sono concepiti, oltre che per la memorizzazione dei dati anche per fornire un accesso rapido ed efficace ad una pluralità di utenti contemporaneamente e garantire protezione sia dai guasti che dagli accessi indebiti.

Solitamente si tratta di macchine multiprocessore e con dischi fissi configurati in modalità RAID per la memorizzazione stabile ed affidabile dei dati che garantisca la continuità del servizio anche in caso di guasto ad un componente (sistemi *fault tolerant*).

In ambienti dove vengono processate transizioni con moli di dati particolarmente elevate vengono utilizzati anche componenti hardware che hanno la funzione specifica di acceleratori di database e che sono collegati ad uno o più server attraverso canali preferenziali ad alta velocità.

In prima analisi si possono individuare diversi sottosistemi che devono essere contenuti in un database server per il corretto funzionamento; tali sottosistemi possono essere classificati in:

- sistema di I/O dei dati;
- sistema di Sicurezza, per la gestione degli utenti e dei diritti di accesso ai dati;
- sistema di gestione della memoria dove memorizzare i dati;
- sistema per la gestione dei Lock, che amministra le situazioni in caso di dati condivisi da più utenti;
- sistema di controllo sulle transazioni, che controlla che le transazioni eseguite siano andate a buon fine;
- sistema di gestione del Logging degli utenti;
- sistema per il Recovery dei dati in caso di bisogno;
- sistema di controllo dei processi;
- sistema di elaborazione delle query che vengono sottoposte;
- sistema di Storage Management, che permette un salvataggio dei dati più intelligente;

Alla luce di tale analisi si può osservare che i database server non sono affatto semplici, e sono formati da vari moduli che lavorano anche in maniera autonoma le une dalle altre.

In generale database server e client operano in rete utilizzando un protocollo di comunicazione (spesso TCP/IP).

In ogni caso il client, per usufruire dei servizi, deve contattare il *listener* del server, utilizzando il relativo indirizzo e numero di porta (che si presuppone conosciuto a priori) il quale riserverà un canale di comunicazione apposito per la comunicazione tra il server e quel client.

Per permettere la creazione di nuove applicazioni client, la maggior parte dei database server supportano le API (*Application Program Interface*) dell'*Open Database Connectivity (ODBC)* o *Java Database Connectivity (JDBC)*, che forniscono ai programmatori strumenti per l'accesso ai database.

I database server rappresentano la fetta più ampia di mercato, accanto ai database cosiddetti "*personal*", ossia a basi di dati destinati all'uso da parte di un solo utente con mole di dati ristretta.

La quasi totalità dei **Data warehouse** aziendali è implementata tramite un database server.

Come vedremo i database con tecnologia Oracle sono database server con struttura client server [DBJW 2004].

## 4. L'obiettivo della tesi: sincronizzazione tra database

Oggigiorno una delle sfide degli ambienti di business distribuiti è la condivisione di informazioni tra un insieme eterogeneo di applicazioni e di database [Chappell 2004]. Nel momento in cui il numero di applicazioni cresce, si finisce con l'utilizzare molti database, magari con tecnologie differenti. La condivisione dell'informazione in ambienti così complessi ed eterogenei può essere dispendiosa. Per questo motivo è sempre più utile trovare metodi che facilitano questa condivisione. L'obiettivo della presente tesina è proprio quello di analizzare i principali metodi messi a disposizione dalla tecnologia Oracle per la **sincronizzazione tra database** [Alapati 2005] e [Oracle10gMan].

La parola "sincronizzazione" in questo ambito è da intendersi non solo come replica di oggetti memorizzati fisicamente in un database in un altro, ma anche come riferimento a dati che non sono fisicamente memorizzati all'interno di un database, ma che sono registrati in un altro ad esso collegato; in altre parole il concetto di sincronizzazione è da intendersi come condivisione di dati tra più database.

Nella realtà, infatti, capita spesso che un insieme di dati memorizzato in un database debba essere utilizzato da più persone, e quindi debba essere replicato in più basi di dati.

Un esempio può essere fornito dal database centrale di un'azienda in cui gli impiegati registrano gli ordini che l'azienda riceve giornalmente.

In questa situazione può essere utile replicare la banca dati degli ordini, in maniera non sincrona (ad esempio la replica degli ordini del giorno può essere effettuata la notte dello stesso giorno), su un altro database, sul quale i manager dell'azienda possono effettuare analisi di vendite, per valutare l'andamento dell'impresa.

Quando si parla di sincronizzazione tra database, si può individuare sempre un **database master**, che è il database di partenza, quello che contiene i dati da replicare, ed un **database slave**, che è il database di destinazione in cui i dati vengono replicati. Nell'esempio precedente il database master è rappresentato dalla banca dati centrale in cui vengono registrati gli ordini giornalieri, mentre il database slave è rappresentato dal database utilizzato dai manager in cui i dati, in un certo momento, vengono replicati.

Al momento della replica dei dati si dice che i due database sono sincronizzati in quando contengono gli stessi oggetti.

La replica degli oggetti può avvenire in due modalità:

- **Replica sincrona**, che prevede la copia dei dati dal database master al database slave nel momento stesso in cui avviene una modifica;
- **Replica asincrona**, che prevede la copia dei dati dal database master al database slave solamente in un momento deciso dall'utente, copiando tutti i dati modificati dall'ultima sincronizzazione.

Nella realtà spesso è utile creare dei meccanismi di sincronizzazione che permettano il passaggio dei dati in entrambe le direzioni, ossia che rendano possibile le repliche sia, dal database master a quello slave, sia dal database slave a quello master;

L'esempio tipico ci viene fornito dalla situazione del **“commesso viaggiatore”**:

un commesso viaggiatore ha la necessità di raccogliere i dati di interesse in un database periferico di modeste dimensioni, e riversare questi nel database centrale dove vengono archiviati tutti i dati. In tale situazione è utile che il canale di trasmissione tra database centrale (master) e database periferico (slave) sia percorribile in entrambe le direzioni.

La situazione del commesso viaggiatore, appena descritta, rappresenta il modello teorico seguito nell'ambito di questa tesi per l'individuazione e l'analisi delle soluzioni di sincronizzazione tra database con tecnologia Oracle;

E' stato utilizzato, infatti, come database periferico il **Database Oracle 10g Express Edition**, database gratuito adatto a modeste entità di dati che supporta la tecnologia Oracle, e come database centrale il **Database Oracle 10g Standard Edition**, il database Oracle che rappresenta una buona fetta di mercato.

Partendo da questa situazione sono stati individuati i metodi attraverso i quali i due database possono condividere dati in essi memorizzati, giungendo, poi, in seconda analisi, a stabilire quale fosse quello preferibile.

# Capitolo2

## I database relazionali ed il loro ruolo

Come già detto nel capitolo precedente i database vengono gestiti da sistemi software detti Database Management System(DBMS) [ACFPT 2003].

Un gruppo di studio conosciuto come **ANSI/X3/SPARC** ha proposto un modello strutturale per i DBMS cui la maggior parte dei produttori aderiscono.

Tale modello si basa su tre livelli o gestori, a ciascuno dei quali corrisponde un particolare livello di astrazione nel quale viene descritto il modello dai dati.

Tali livelli sono:

- **Livello fisico:** che si occupa delle operazioni di livello più vicino all'hardware, fornendo una descrizione a basso livello di tutto il database; Attraverso la sua definizione viene specificata la struttura di memorizzazione del database, ovvero come sono organizzati i dati nei file e quali strutture ausiliarie sono presenti per facilitarne l'accesso.
- **Livello logico:** che si occupa della gestione dell'archivio di dati, e fornisce una visione logica dell'organizzazione completa del database; in questo livello possono essere definiti i tipi di dati contenuti, le loro proprietà, le associazioni e i vincoli di integrità.
- **Livello esterno:** è il livello cui l'utente vede il database, e consente l'accesso ai dati offrendo la possibilità di compiere operazioni su di essi; è la componente che supporta l'interfacciamento verso programmi applicativi, scritti in linguaggi di programmazione convenzionali, che effettuano chiamate al DBMS per accedere ai dati del database.

Questa struttura a livelli consente di avere indipendenza di dati tra un livello e l'altro; E' possibile, quindi, modificare, ad esempio, il livello logico senza modificare il livello esterno, e, quindi, senza dover riscrivere i programmi applicativi [Zaffanella 2002].

All'interno del livello logico viene implementato il "*modello dei dati*"; per modello dei dati si intende l'insieme dei concetti utilizzati per organizzare i dati di interesse e descrivere la struttura in modo comprensibile ad un calcolatore [ACPT 2002].

Esistono diversi "modelli di dati" che un database può supportare:

- **Modello gerarchico;**
- **Modello reticolare;**
- **Modello relazionale;**
- **Modello ad oggetti;**

Il modello **gerarchico** fu definito negli anni sessanta, durante la prima fase di sviluppo dei DBMS ed è tuttora diffuso soprattutto in ambito Mainframe; Esso è basato sull'uso di strutture ad albero (e quindi gerarchie, da cui il nome), in cui, a partire da un dato padre, si può accedere a dati figli che da esso dipendono (e a loro volta questi possono essere padri di altri figli). Il modello gerarchico è adatto alla rappresentazione di informazioni la cui struttura è gerarchica essa stessa, ma non sono adatti in situazioni in cui si devono creare relazioni, perché in tali situazioni è necessaria l'introduzione di dati duplicati.

Il modello **reticolare** (detto anche *CODASYL* dal nome del comitato di standardizzazione che lo definì con precisione) fu introdotto negli anni sessanta, come

evoluzione del modello gerarchico, permettendo di superare la rigidità della struttura ad albero di quest'ultimo.

Il modello reticolare è basato sull'uso dei grafi: le informazioni contenute in un record sono messe in relazione con altre memorizzando queste ultime in una lista circolare, che inizia e finisce nel record stesso; tale lista è costituita da record di tipo speciale detti *connettori*. I connettori hanno una struttura tale da permettere l'appartenenza a due liste e di fatto mettono in correlazione le informazioni contenute in record distinti. Nel modello relazionale, introdotto negli anni settanta, i dati vengono organizzati in insiemi di record chiamati e rappresentati come tabelle; le relazioni tra le informazioni derivano dalla corrispondenza di alcuni campi di record appartenenti a tabelle diverse; Il modello **relazionale** è il modello utilizzato nella maggior parte dei database presenti oggi sul mercato e per questo motivo sarà opportunamente approfondito nel resto del capitolo.

Infine, il modello **ad oggetti**, fu introdotto negli anni ottanta come evoluzione del modello relazionale, estendendo alle basi di dati il paradigma di programmazione ad oggetti (*Object Oriented*).

Nei successivi paragrafi introduciamo il concetto di database relazionale, ossia un database che utilizza come "modello di dati" il modello relazionale, e ne vediamo le principali caratteristiche; affronteremo, poi, il discorso relativo alla struttura database-centrica ed individueremo i principali metodi di sincronizzazione nei database relazionali.

## 1. Il database relazionale

Un database si dice relazionale quando implementa a livello logico dei dati il modello relazionale (un database è relazionale se aderisce alle *12 regole di Codd*).

Il modello relazionale è un modello implementato nel livello logico dei Database Management System (che quindi vengono detti anche **Relational Database Management system** o **RDBMS**).

Esso fu proposto da **Edgard F. Codd** nel 1970 per semplificare la scrittura di interrogazioni sui database e per favorire l'indipendenza dei dati; venne reso disponibile come modello logico in DBMS reali nel 1981.

Codd individuò una serie di regole, 13 per la precisione, per definire i requisiti che un DBMS deve soddisfare per essere considerato relazionale; Codd scrisse questi criteri nell'ambito di un'iniziativa che serviva ad evitare che la sua definizione di database relazionale fosse resa meno restrittiva quando, nei primi anni ottanta, i venditori di database si affannarono per rimpiazzare i vecchi prodotti con un prodotto pseudo-relazionale.

La 12 regole individuate da Codd sono:

- **Regola 0:** *il sistema deve potersi definire come relazionale, base di dati, e sistema di gestione.* Affinché un sistema possa definirsi sistema relazionale per la gestione di basi di dati (RDBMS), tale sistema deve usare le proprie funzionalità relazionali (e solo quelle) per la gestione della base di dati.
- **Regola 1:** *l'informazione deve essere rappresentata sotto forma di tabelle;* Le informazioni nel database devono essere rappresentate in maniera univoca, e precisamente attraverso valori in colonne che costituiscano, nel loro insieme, righe di tabelle.

- **Regola 2:** *la regola dell'accesso garantito:* tutti i dati devono essere accessibili senza ambiguità (questa regola è in sostanza una riformulazione del requisito per le chiavi primarie). Ogni singolo valore scalare nel database deve essere logicamente indirizzabile specificando il nome della tabella che lo contiene, il nome della colonna in cui si trova e il valore della chiave primaria della riga in cui si trova.
- **Regola 3:** *trattamento sistematico del valore NULL;* il DBMS deve consentire all'utente di lasciare un campo vuoto, o con valore NULL. In particolare, deve gestire la rappresentazione di informazioni mancanti e quello di informazioni inadatte in maniera predeterminata, distinta da ogni valore consentito (per esempio, "diverso da zero o qualunque altro numero" per valori numerici), e indipendente dal tipo di dato. E' chiaro inoltre che queste rappresentazioni devono essere gestite dal DBMS sempre nella stessa maniera.
- **Regola 4:** la descrizione del database deve avvenire ad alto livello logico tramite i metadati.
- **Regola 5:** deve esistere un linguaggio che permetta la gestione dei dati (come SQL).
- **Regola 6:** si possono creare delle viste per vedere una parte dei dati. Queste viste devono essere aggiornabili.
- **Regola 7:** le operazioni che avvengono sul database devono avvenire anche sulle tabelle.
- **Regola 8:** i dati memorizzati nel database devono essere indipendenti dalle strutture di memorizzazione fisiche.
- **Regola 9:** i dati devono essere indipendenti dalla struttura logica del database per garantire la crescita naturale e la manutenzione del database.
- **Regola 10:** le restrizioni sui dati devono essere memorizzate nel database.
- **Regola 11:** l'accesso ai dati è indipendente dal tipo di supporto per la lettura o memorizzazione degli stessi.
- **Regola 12:** l'accesso ai dati non deve annullare le restrizioni o i vincoli di integrità del linguaggio principale.

Queste regole, pubblicate da Codd nell'articolo "*Is Your DBMS Really Relational?*" nel 1985, sono, infatti, così rigide che anche i sistemi la cui unica interfaccia sia il linguaggio SQL non ne soddisfano alcune.

Per questo motivo molti considerano come relazionali anche dei DBMS che non soddisfano alcune di queste regole. (molti dei database presenti sul mercato e venduti come "relazionali" non soddisfano tutte le regole).

Il modello relazionale si basa sull'algebra relazionale (un linguaggio di interrogazione *procedurale* definito da operatori di base, la cui applicazione a relazioni produce sempre come risultato una relazione) e sulla teoria degli insiemi ed ha come concetto fondamentale quello di **relazione** (che come vedremo sarà rappresentabile come una **tabella**).

L'assunto fondamentale del modello relazionale è che tutti i dati sono rappresentati come **relazioni** e sono manipolati con gli operatori dell'algebra relazionale.

Il concetto di relazione proviene dalla matematica ed ha la seguente definizione:

Presi  $D_1, \dots, D_n$  ( $n$  insiemi anche non distinti) si può definire un'operazione detta *prodotto cartesiano*, indicata con  $D_1 \times \dots \times D_n$ , che definisce un insieme di  $n$ -uple  $(d_1, \dots, d_n)$  tali che  $d_1 \in D_1, \dots, d_n \in D_n$ . Si dice **relazione** di  $n$  elementi un qualunque sottoinsieme di  $D_1 \times \dots \times D_n$  e si dicono **domini della relazione** gli insiemi  $D_1, \dots, D_n$  [Zaffanella 2002].

Il **dominio** dei dati, rappresenta quindi l'insieme dei valori che può assumere un determinato campo di una relazione.

Ad ogni dominio viene dato un nome, detto **attributo**, che descrive il ruolo che tale dominio ha nella relazione.

Un attributo è una coppia ordinata di "nome di attributo" e "nome di tipo", mentre un valore di attributo è un valore specifico valido per quel tipo di dato. Viene inoltre detta **tupla** un insieme non ordinato di valori degli attributi.

Una relazione consiste di una **testata** e di un **corpo**, dove la testata è un insieme di attributi e il corpo è un insieme di  $n$  tuple. La testata di una relazione è anche la testata di ciascuna delle sue tuple.

Nei database relazionali le relazioni vengono rappresentate graficamente tramite le tabelle, in particolare la definizione della tabella è la testata e i dati che vi appaiono sono il corpo.

Ogni colonna di tale tabella costituisce un attributo. La tabella è quindi rappresentabile come una lista di colonne, ciascuna delle quali ha un nome univoco (l'attributo appunto) e un dominio. Il valore di un attributo è il valore di una cella identificata da una specifica coppia riga-colonna.

Le tuple nei database relazionali sono rappresentate dalle righe delle tabelle.(fig.2.1).



Fig.2.1

In altre parole, quindi, un database relazionale è un insieme di tabelle, collegate tramite determinati attributi, che hanno la caratteristica di avere diversi valori associati per ogni tupla della tabella e detti **chiave primaria**. Quando questi vengono utilizzati per relazionare due tabelle vengono dette **chiavi esterne**.

Il modello relazionale consente al progettista di database di creare una rappresentazione consistente e logica dell'informazione. La consistenza viene ottenuta inserendo nel progetto del database appropriati vincoli detti **vincoli di integrità**.

Per integrità si intende:

- **Integrità semantica**, ossia la modifica dei dati deve avvenire all'interno del dominio previsto per l'attributo;
- **Integrità di contesa**, ossia il valore dei dati va protetto da situazioni conflittuali di modifica contemporanea (lock);
- **Integrità di entità**, ossia nessun attributo che fa parte di una chiave primaria può essere nullo;
- **Integrità referenziale**, ossia per ogni chiave esterna non nulla deve esistere la chiave primaria a cui fare riferimento;

- **Integrità definita dall'utente**, ossia controlli di dominio per garantire la correttezza formale dei dati.

Un vincolo di integrità quindi è definito come una proprietà che deve essere soddisfatta dalle tuple delle tabelle.

Il collegamento tra le varie tabelle attraverso l'utilizzo di attributi chiave, permette di mettere in relazione qualsiasi tabella della base di dati con qualsiasi altra, senza troppa difficoltà; Questo concetto è il punto di forza dei database relazionali ed è il motivo per il quale questi sono diventati i più diffusi sul mercato.

Il ruolo che i database relazionali, oggi, rivestono nelle aziende e nei sistemi informativi di queste è di primaria importanza.

Molte imprese, infatti, utilizzano ancora fogli di calcolo e file per la gestione aziendale, ma questo sistema ha gravi limitazioni. Per prima cosa, tentare di gestire e ottimizzare "isole di informazioni" disperse nell'intera impresa presenta grossi problemi di accesso ai dati. Le prestazioni, specialmente durante i picchi di domanda degli utenti, possono risultare del tutto insoddisfacenti. Inoltre, quando i dati sono salvati in vari formati su differenti macchine, le attività di sicurezza e privacy possono risultare particolarmente impegnative, per non parlare delle operazioni di backup e recovery.

Un database relazionale risolve tutti questi problemi e offre una potenza straordinaria, specialmente per quanto riguarda la capacità di analisi. I database relazionali permettono di manipolare i dati in modi anche complessi. Permettono inoltre di recuperare tutti i record abbinati a criteri specifici, collegare tabelle attraverso chiavi e aggiornare i record in serie.

## 2. I database server presenti sul mercato

Il mercato dei database odierno è rappresentato da un grande numero di soluzioni software che implementano sistemi di gestione di basi di dati con modello relazionale (RDBMS) e architettura client server [Destri 2007].

Una classificazione dei database può essere fatta esaminando il tipo di licenza software posseduta: esistono, infatti, database **liberi** o **open source** e database **commerciali**.

La differenza principale, come è facile intuire, è che i database open source hanno una licenza libera, e sono utilizzabili liberamente da tutti senza nessun corrispettivo in denaro, mentre i database commerciali hanno una licenza proprietaria, che bisogna acquistare per poter utilizzare.

I principali database liberi o open source sono:

- **Firebird SQL**
- **My SQL**
- **PostgreSQL**
- **SQLite**

Tra questi i più diffusi sono sicuramente My SQL, utilizzato molto in applicazioni web, perché garantisce un buon interfacciamento a linguaggi di scripting come il php, e PostgreSQL, utilizzato molto in ambiti accademici.

I principali database di natura commerciale sono invece:

- **DB2 di IBM**
- **Microsoft SQL Server**
- **Oracle Database**

Secondo una ricerca di **Idc** sul mercato mondiale dei database il giro d'affari complessivo dei database è dell'ordine dei 20 miliardi di dollari (2004); Il 75% del mercato è dominato da **Oracle**, **IBM**, e **Microsoft** con quote rispettivamente del 39,8%, 31,3% e 12,1%.

Negli ultimi anni, molti vendor commerciali hanno sviluppato, accanto alle versioni a pagamento, versioni gratuite per attirare l'attenzione verso la propria tecnologia degli utenti open source, auspicando un passaggio in futuro alle versioni a pagamento, più complete e personalizzabili.

E' il caso ad esempio di Microsoft che ha rilasciato Microsoft SQL Server 2005 Express Edition, oppure di Oracle che ha rilasciato una versione Express (gratuita) dei propri database.

### **3. Il database server entro i sistemi informatici e la struttura database-centrica**

Come già precedentemente detto, i sistemi informativi sono oggi il cuore di qualsiasi azienda. La porzione di sistema informativo che gestisce le informazioni utilizzando la tecnologia informatica viene detto **sistema informatico** [De Marco 2000].

Il sistema informatico viene spesso indicato con l'acronimo "IT", da **Information Technology** o anche da "ICT" da **Information and Communication Technology**.

Esso, quindi, tratta, elabora e memorizza l'informazione nei formati digitali utilizzando tecnologie informatiche come calcolatori, periferiche, mezzi di comunicazione, programmi ecc [BFM 2001].

Storicamente le aziende che adottano una struttura di organizzazione "*a funzioni*", ossia una divisione rigida delle attività aziendali comuni in dipartimenti autonomi, sono le meno adatte ad utilizzare le tecnologie informatiche come supporto alle attività produttive; Questo perché nella pratica succede molto spesso che in tali aziende ogni reparto procede alla "meccanizzazione", ossia all'inserimento semplice delle tecnologie IT entro funzioni del lavoro, per proprio conto e le applicazioni specialistiche non sempre prevedono una facile integrazione tra di loro, creando il cosiddetto problema delle "isole informatiche", cioè sistemi informatici diversi in ogni reparto, che usano formati di rappresentazione dei dati diversi, costringendo i responsabili IT alla creazione di apposite interfacce di comunicazione, che effettuino la traduzione dei dati tra i vari formati [Chappell 2004].

Nelle moderne aziende strutturate "*a processi*", in cui le attività sono suddivise in base allo scopo della loro realizzazione, e quindi sono riunite anche attività che non appartengono allo stesso dipartimento aziendale, i flussi informatici coinvolgono diverse divisioni, per cui i sistemi informatici collegati ai processi devono essere flessibili e riprogrammabili rapidamente seguendo l'evoluzione dei processi business stessi.

In tale situazioni è essenziale disporre di una base di dati unica, centralizzata e condivisa dalle varie aree funzionali cosicché i dati siano resi disponibili a tutti (**struttura database-centrica**).

Queste è il ruolo che i database server rivestono all'interno del contesto aziendale.

Le informazioni che normalmente vengono trattate all'interno di un'azienda sono di vario genere ed entità a seconda del tipo di attività delle quali scaturiscono.

Parlare di database, nel contesto delle grandi aziende, come unico strumento informatico per la gestione delle informazioni è un po' riduttivo, in quanto esistono diversi sistemi applicativi adatti alla gestione di ogni tipo di informazione [Destri 2007]. Tali sistemi si possono classificare come:

- Sistemi di **livello operativo**, che costituiscono di solito il componente quantitativamente più presente del sistema informatico. Supportano la registrazione delle attività elementari e delle transazioni che si svolgono nell'azienda. Come, ad esempio, vendite, incassi, depositi contante, paghe. Il loro scopo principale è quindi supportare le attività routinarie e registrare il flusso delle transazioni entro l'azienda, al livello operativo. Il loro componente fondamentale sono i **Transaction Processing System (TPS)** detti anche **On-line Transaction Processing (OLTP)**, che svolgono e registrano le transazioni di routine necessarie per le attività quotidiane.
- Sistemi di **gestione della conoscenza** che possono essere i sistemi per ufficio, come il pacchetto MS Office o OpenOffice, che aumentano la produttività dei lavoratori sui documenti e i dati, oppure i sistemi di gestione della conoscenza veri e propri, meglio noti come **Knowledge Working System (KWS)**, che supportano i lavoratori della conoscenza nella creazione di nuova conoscenza, e permettono di consultare le informazioni.
- Sistemi di **supporto dell'attività manageriale**, che favoriscono le attività di controllo e monitoraggio e le attività decisionali ed amministrative dei middle manager. Tali sistemi sono composti dai **Management Information System (MIS)**, che servono principalmente le funzioni di pianificazione e controllo, supportando le decisioni a livello manageriale traendo solitamente i dati dalle applicazioni dei livelli sottostanti.
- Sistemi di **supporto delle attività strategiche**, che aiutano i senior manager ad affrontare i problemi strategici e a valutare le tendenze a lungo termine. Sono formati dagli **Executive-Support System (ESS)**.

Alla base di ognuna di queste macrocategorie di sistemi software vi è un database server.

A seconda del tipo di sistema però la struttura interna del database cambia; Le basi di dati per l'attività quotidiana di OLTP normalmente sono caratterizzate da:

- Normalizzazione completa delle tabelle (per normalizzazione si intende in questo contesto il processo volto all'eliminazione della ridondanza e del rischio di inconsistenza del database);
- Alto numero di tabelle e di associazioni;
- Dati memorizzati al minimo livello di granularità;
- Frequente uso di interrogazioni che richiedono join (operatore dell'algebra relazionale che unisce due tuple in relazione tra loro) di molte tabelle;
- La struttura dei dati non varia di frequente;
- Ottimizzazione per l'inserimento dei dati e lettura di un piccolo numero di record alla volta;

Passando da un sistema transazionale ad un sistema di analisi, cambiano le caratteristiche di:

- Normalizzazione;
- Prestazioni su query e modifica dei dati;
- Profondità storica;
- Complessità delle query;
- Dettaglio degli eventi rilevanti.

La fase estrazione dei dati dalla base dati di produzione, di trasformazione dei dati nella rappresentazione più adatta all'analisi da effettuare e di caricamento dei dati nel programma di analisi viene detta Extract Transform and Load (**ETL**), ed è effettuata ogni qual volta si debbano eseguire analisi strategiche.

Viene detto **On-Line Analytical Processing (OLAP)** il processo di analisi completa dei dati.

Esso viene eseguito utilizzando un database che ha le seguenti caratteristiche:

- Le entità sono denormalizzate;
- Lo schema del database è semplice (con meno tabelle e meno relazioni) per una comprensione più facile da parte dell'utente.
- I dati memorizzati possono essere aggregati o riassuntivi;
- Le interrogazioni richiedono pochi join;
- E' ottimizzato per la consultazione di grandi moli di dati e solitamente è in sola lettura.

Esistono dei modelli di database generici pensati per queste esigenze come lo *Star Schema* e il *Snowflake Schema* [ACPT 2002].

Un database OLAP può essere realizzato sfruttando un generico database relazionale, ma esistono anche soluzioni specifiche diverse (*OLAP DB Server*) per quanto poco usate.

Come abbiamo già detto per database server intendiamo un database relazionale che implementa il modello client server per permettere l'accesso ai dati.

Nel contesto aziendale, in cui vi è un unico database centrale dove vengono memorizzate le informazioni, i client, che condividono i dati, di solito inviano delle richieste che sono per lo più messaggi in formato SQL, al server. Il server elabora questi messaggi e produce come risultato un set di dati che spedisce come risposta ai client, oppure, nei casi di istruzioni di modifica dei dati (inserimento, aggiornamento, cancellazione) il server effettua la modifica dei dati, notificando poi al client l'azione svolta.

Il database centrale definisce le diverse entità in maniera univoca e omogenea per tutti i client al quale fanno riferimento. Tale database garantisce la mancanza di ridondanza tra i dati e l'integrità del sistema.

Nelle grandi imprese il database server (naturalmente relazionale) unico è strettamente correlato al concetto di **data warehouse**.

Un data warehouse rappresenta il "magazzino di dati" a livello di impresa, ossia un insieme di strumenti per convertire un vasto insieme di dati in informazioni utilizzabili dall'utente, con la possibilità di accedere a tutti i dati dell'impresa, centralizzati in un solo database che garantisce coerenza e consolidamento dei dati, velocità di accesso alle informazioni e supporto all'analisi dei dati.

Il data warehouse a volte può essere segmentato per questioni di praticità o per dividere i dati in base ai dipartimenti aziendali; si parla allora di **data mart** (l'insieme di data mart di tutti i dipartimenti aziendali forma il data warehouse).

La figura seguente (*Figura 2.2*) illustra la struttura *database-centrica* che oggi la maggior parte delle aziende adotta.

In essa si vede bene come i dati giornalieri e di routine vengano registrati dai sistemi OLTP nel database centrale (data warehouse) e da lì, attraverso sistemi OLAP vengano utilizzati per la realizzazione di analisi strategiche. Si consiglia [Destri2007] per approfondimenti.

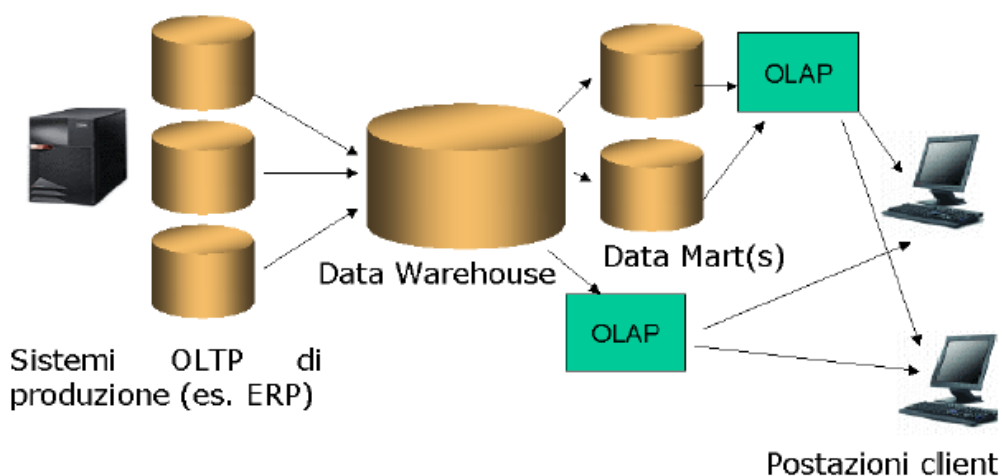


Fig.2.2

## 4. Le situazioni pratiche di comunicazione fra db server

I dati che trovano collocazione all'interno di un database, hanno la caratteristica di essere dati importanti per gli utenti di quest'ultimo.

Questa importanza ha spinto, nel corso degli anni, alla creazione di database con politiche di sicurezza sempre più affidabili e accurate.

Il ruolo fondamentale che i dati rivestono, porta spesso, nella realtà, alla necessità di replica di questi in più basi di dati, sia per l'esecuzione di analisi in parallelo, sia per operazioni di backup dei dati.

In tali situazioni quindi è utile che i database implementino funzioni di comunicazione con altri database.

Nell'attuale mercato, la quasi totalità di database server dispone di diversi metodi di sincronizzazione da e verso il database.

Prima di analizzare in linea generale questi metodi, diamo alcuni esempi di situazioni pratiche in cui la comunicazione tra database server è importante:

*Esempio 1:* Supponiamo che un'azienda che opera in un determinato settore utilizzi un database centrale dove registra tutte le proprie transazioni. All'inizio della propria vita, l'impresa acquista un database di modeste dimensioni per contenere le spese. Dopo qualche esercizio, a seguito dell'ampliamento del proprio giro d'affari, l'azienda ha la necessità di passare ad un database di dimensioni più ampie per registrare il numero di transazioni crescente.

Per non perdere i dati registrati fino a quel momento, e mantenere la centralità di un unico database, è necessaria la replica dei dati del "vecchio" database server nel "nuovo" database server.

*Esempio 2:* Una società leader di un gruppo di società, periodicamente vuole redigere un'analisi sull'andamento del gruppo; per fare ciò essa ha bisogno che i dati che sono memorizzati nei database di ogni società appartenente al gruppo vengano replicati tutti in un database centrale su cui fare l'analisi.

Per fare ciò c'è bisogno di un meccanismo di comunicazione tra i database delle società e il database centrale.

*Esempio 3:* Un terzo è ultimo esempio, ci può essere fornito dalla situazione in cui un rappresentante registra i contratti conclusi durante la settimana in un database su un dispositivo portatile come ad esempio un laptop.

La società per cui lavora richiede che i contratti conclusi da tutti i rappresentanti siano registrati su di un database centrale alla fine di ogni settimana.

Per fare ciò i dati memorizzati sul database del laptop devono essere riversati sul database della società.

Questa situazione rispecchia il modello teorico già discusso precedentemente del commesso viaggiatore.

E' chiaro che anche in questa situazione è utile un meccanismo di comunicazione tra i due database.

Come detto quindi in tutti i tre gli esempi è opportuno utilizzare un metodo di comunicazione ottimale tra i vari database.

I database, che devono essere collegati mediante un canale di comunicazione, devono disporre di metodi per la sincronizzazione automatica in modo da facilitare le cose.

Nel prossimo paragrafo vediamo le soluzioni che, in linea generale, possono essere implementate dai database attualmente disponibili.

## 5. Gli strumenti di comunicazione

Gli strumenti di comunicazioni individuabili nei database in commercio sono molteplici, e sono diversi per ognuno dei vendor che li ha sviluppati.

Durante lo svolgimento del tirocinio, ho potuto approfondire le soluzioni specifiche che la tecnologia Oracle mette a disposizione.

La trattazione di tale soluzioni, ristretta però al contesto del problema da me affrontato, è descritta nella seconda parte nel presente documento (capitolo 4).

L'obiettivo di questo paragrafo è invece quello di individuare le soluzioni di comunicazione tra database applicabili in situazioni generali, in cui non è specificato il tipo di database.

Quanto descritto in questo paragrafo deve, quindi, essere interpretato come modello teorico da seguire per individuare le tecniche ottimali che sono applicabili nelle situazioni descritte nel paragrafo precedente.

In generale quindi le strade percorribili quando si deve fare comunicare due database sono le seguenti:

*Sviluppo di applicazioni ad hoc:* La tecnica forse più immediata e personalizzabile per la comunicazioni tra database è lo sviluppo di applicativi ad hoc che realizzino le operazioni richieste.

Grazie infatti alle API **ODBC (Open Database Connectivity)** e **JDBC (Java Database Connectivity)**, che presentano i maggiori database presenti sul mercato, è

possibile sviluppare programmi, con i più diffusi linguaggi di programmazione che interagiscono con i DBMS.

Tali programmi possono, quindi, prelevare dati da un database e inserirli in un altro, con molta efficienza e praticità, senza scomodare funzioni specifiche del database.

Nella pratica, anche grazie alle numerose funzioni già incorporate nei DBMS, questa soluzione è applicata solo in situazioni particolari, situazioni, cioè che non rispecchiano lo standard.

*Import-export dei dati:* La maggior parte dei DBMS in commercio prevede la possibilità di esportare ed importare dati in formati specifici.

Tale possibilità permette di rendere disponibili “all'esterno” del database le banche dati.

I meccanismi di import-export sono utilizzati anche per passare dati tra database con tecnologie differenti, in quanto, nella maggior parte dei casi, è possibile esportare i dati anche in formati diversi da quelli utilizzati dal database stesso. Ad esempio in Oracle è possibile esportare ed importare i dati in formato compatibile con Microsoft Access.

*Database-link:* I principali database presenti sul mercato permettono di creare link, ossia in pratica canali di comunicazione diretti, tra database. Tali link vengono detti *Database-link* o *DBlink*. Un DBlink creato da un database A and un database B, permette di leggere i dati memorizzati fisicamente in B nel database A.

E' importante sottolineare che i DBlink non replicano i dati, ma creano indici ai dati stessi.

Approfondiremo il concetto di DBlink utilizzando la tecnologia Oracle nel proseguo della tesi.

*Funzioni specifiche dei DBMS:* Le più importanti tecnologie di database, oggi, mette a disposizione funzioni di sincronizzazioni e di replica dei dati tra database.

Oracle, ad esempio, incorpora nei propri database gli **Oracle Streams**. Tale funzione, come vedremo meglio, permette di creare dei flussi di dati tra database Oracle in maniera automatica e pulita, senza dover scomodare altre funzioni.

In generale tutte le tecnologie di database presenti sul mercato hanno soluzioni specifiche per la comunicazione tra database.

# Capitolo 3

## Il Database server Oracle 10g

Oracle Corporation è tra le più grandi società al mondo di software per le imprese, la prima ad aver commercializzato il database relazionale per la gestione dei dati (1977). Tra i clienti Oracle ci sono altre 2000 istituzioni ed enti pubblici in tutto il mondo, il suo successo è legato all'alta affidabilità, alla sicurezza dei dati e alla potenza dell'architettura su cui si basa.

L'ultimo DB server della casa californiana già in uso nel mercato è il **DB server Oracle 10g** (la versione 11g è da poco stata rilasciata e non è ancora in uso in sistemi di produzione).

Tale DB server sostituisce il precedente DB Server Oracle 9i che ha avuto molto successo e che ancora attualmente è molto diffuso.

Come la stessa Oracle Corporation ha annunciato al momento dell'introduzione del Database 10 g sul mercato, la caratteristica innovativa del DB è la possibilità di interfacciarsi al Grid Computing (da qui il la "g" del nome) [Oracle10gMan].

Questa caratteristica, che tratteremo più approfonditamente più avanti, permette di governare un numero anche elevato di diversi DB server Oracle (e non solo) come un'unica risorsa, in modo da renderli immediatamente utilizzabili per ogni evenienza di elaborazione aziendale.

Nei successivi paragrafi analizzeremo dapprima la struttura generale dei DB server 10g e le varie versioni presenti sul mercato, dopodichè elencheremo i principali componenti aggiuntivi dei DB Oracle 10g e il loro scopo, per poi trattare, infine, i metodi tradizionali che Oracle mette a disposizione per la sincronizzazione tra Database. Per ulteriori informazioni su Oracle si vedano [AreaSPOracle 2007], da cui sono tratte alcune delle figure del presente capitolo, [Oracle10gMan], [Alapati 2005], [DBJW 2004].

### 1. Struttura di Oracle 10g

Come detto più volte, in informatica, per database si intende uno strumento per raccogliere e conservare i dati in forma strutturata.

Abbiamo, inoltre, già introdotto la distinzione tra database server e database personal, distinguendo gli uni dagli altri a seconda del tipo di architettura e dal tipo di utenti.

Focalizziamo ora la nostra attenzione sulla specifica struttura dei DB server Oracle 10g.

Il Database server Oracle 10g ha una struttura rigidamente client/server di tipo multiutente (tranne che nella versione personal), in cui il server gestisce solamente i dati e non ha alcuna interfaccia utente (è un demone in Unix ed un servizio in Windows), mentre il client solitamente è rappresentato da un programma che permette di eseguire interrogazioni sul server, come un terminale SQL, (SQL\*Plus, SQLDeveloper, SQL Worksheet) oppure da un programma di controllo come OEM

(Oracle Enterprise Manager, un'applicazione Java che permette di gestire il server via attraverso una interfaccia grafica).

In quanto database server multiutente, Oracle server deve gestire l'accesso, anche simultaneo, ai medesimi dati da parte di più utenti, oltre a gestire anche i profili degli stessi e i loro diritti di accesso ai singoli dati.

Molti dei file di configurazione di Oracle DB server sono in formato testo o addirittura SQL, e quindi modificabili anche attraverso un editor (facendo estrema attenzione a sintassi e formattazione). E' comunque meglio eseguire le configurazioni del server attraverso gli appositi programmi di configurazione contenuti nella dotazione dell'installazione client di Oracle oppure, in alcuni casi, intervenendo direttamente sulle strutture di metadati rappresentati come oggetti del database attraverso un terminale SQL.

La dotazione del client contiene comunque il terminale SQL, ambienti di programmazione SQL o ancora programmi di generazione dinamica di SQL, Client ODBC, .NET, Java, che si connettono al server tramite API.

Il programma client che lavora con Oracle può dialogare con il server attraverso la rete (fig.3.1) oppure essere fisicamente memorizzato sulla stessa macchina del server (fig.3.2);

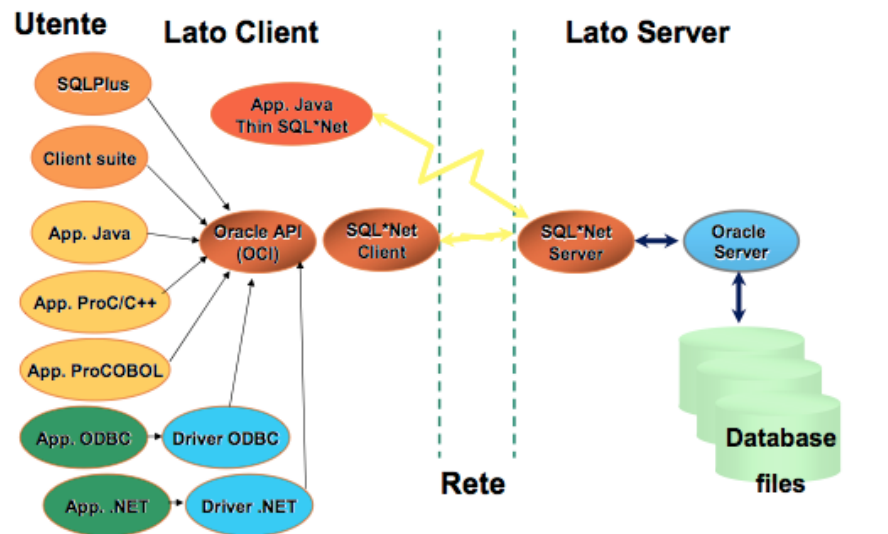


fig.3.1

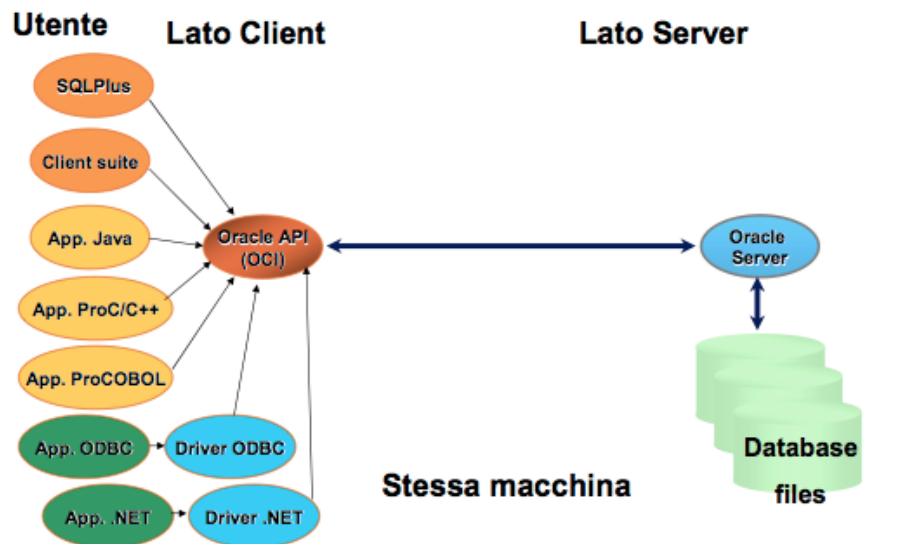


fig.3.2

Oltre ad utilizzare un client specifico, la programmazione del server, intendendo con tale termine l'interazione con il DBMS di Oracle attraverso istruzioni SQL, può avvenire utilizzando script *SQL*, ossia file contenenti una sequenza di istruzioni SQL che vengono eseguite utilizzando un terminale, oppure *Stored Procedure* in PL/SQL, ossia programmi memorizzati all'interno del database che sono scritti in PL/SQL (Procedural Language /Structured Query Language), un linguaggio tipico dei sistemi Oracle, oppure in Java.

In un DB server Oracle esistono diversi utenti, ognuno dei quali ha un determinato ruolo, in base al quale può svolgere diverse azioni; di default, esistono due utenti (*system* e *sys*) che hanno il ruolo di DBA, ossia di database administrator; questi possono amministrare e configurare il DB, agire sulle tabelle di ogni altro utente e accedere alla OEM Console.

Ogni utente ha associato uno *schema*, ossia una collezione di oggetti (che possono essere tabelle, ma anche viste ed altro) da lui posseduta ed identificata con il proprio nome.

Tutto ciò che un utente crea viene memorizzato all'interno del proprio *schema*, su quale può compiere ogni azione.

Per capire bene l'architettura del Database Oracle Server 10g, è importante fare distinzione tra *database* ed *istanza*: con *database* in Oracle si intende l'insieme dei file dei dati e dei file di controllo, mentre con *istanza* si intende l'insieme dei processi che gestiscono e fanno funzionare il database.

In altre parole l'*istanza* è il vero motore de DB Oracle, che gestisce i vari dati da memorizzare.

La figura seguente (fig.3.3) chiarisce meglio questo concetto, mostrando le entità coinvolte nei DB server Oracle;

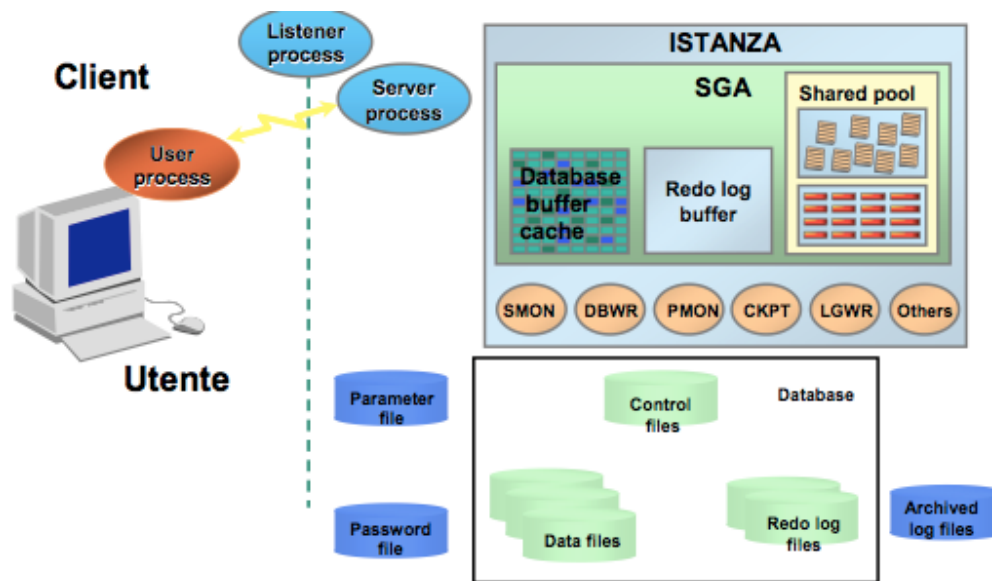


fig.3.3

Da qui si vede bene come con *database* si intenda il rettangolo inferiore della figura, formato da file, mentre con *istanza* si intende il rettangolo nella parte alta, formato da processi e aree di memoria.

Ogni database deve obbligatoriamente fare riferimento almeno ad un'istanza, è anche possibile avere più di un'istanza per database.

Nei successivi due sottoparagrafi analizziamo più in dettaglio l'istanza e il database di Oracle.

### Istanza

Come detto per *istanza* si definisce quell'insieme di processi ed aree di memoria dedicate che "copre" (gestisce) uno ed un solo DB.

Quando il client deve effettuare una richiesta al Db server, avviene ciò che è rappresentato in figura 3.4, ossia nella macchina client viene attivato un *user process* che contatta il *listener process*, il processo che governa le connessioni via rete sulla macchina server, il quale provvede a dedicare a tale *user process* un proprio *server process*, ossia un processo lato server che permette di elaborare le richieste fatte dal client. Nella configurazione dedicated server, tipicamente usata nelle installazioni medio-piccole, per ogni *user process* vi è un *server process* dedicato che utilizza una propria area di memoria, non condivisa con gli altri processi, che contiene anche le informazioni sulla sessione di collegamento; tale area di memoria è detta **Program Global Area (PGA)**. Nella configurazione shared server, usata quasi esclusivamente nelle installazioni medio-grandi, invece troviamo, accanto al listener, uno o più processi dispatcher che smistano le richieste provenienti da più client verso i server disponibili per esaudirle.

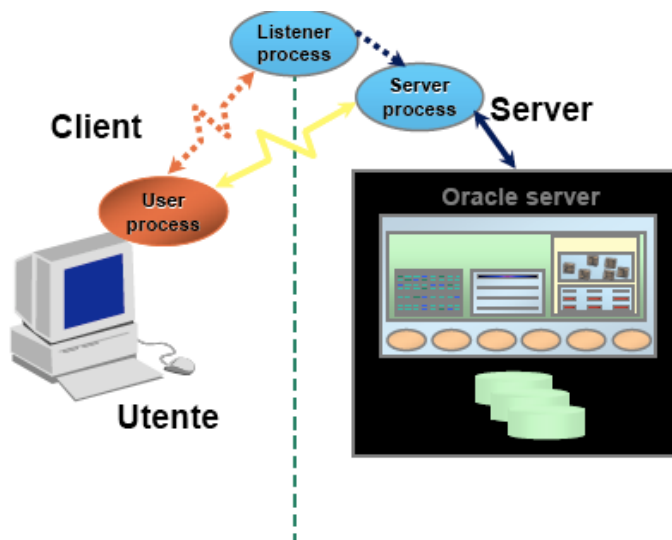


fig.3.4

Il *server process* è colui che interagisce in prima persona con l'istanza del DB server Oracle.

All'interno dell'istanza si possono distinguere due importanti elementi.

- Un insieme di aree di memoria;
- Dei processi di background;

L'insieme delle aree di memoria è detta **Sistem Global Area (SGA)**, e contiene tre elementi molto importanti:

- **Shared Pool**

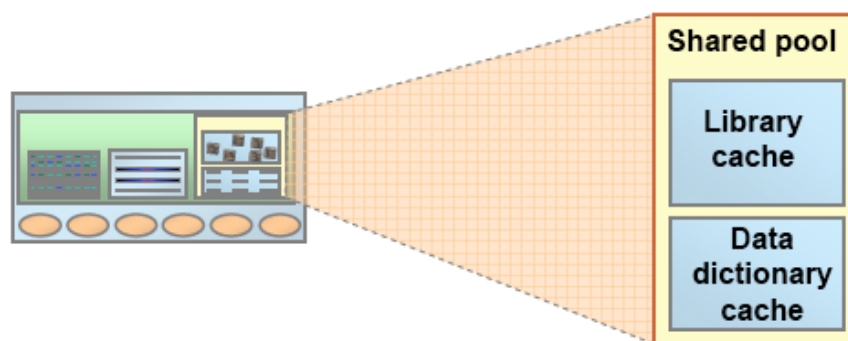


fig.3.5

Questa area di memoria dalle dimensioni definite dal parametro `SHARED_POOL_SIZE`, contiene la **Library cache** (o SQL Area), che mantiene i sorgenti degli statement richiesti, i comandi già parserizzati e l'ordine di esecuzione; essa è di fondamentale importanza per le prestazioni perché Oracle effettua il parse di ogni statement una sola volta e lo pone poi nella library cache, che viene condivisa con tutti gli utenti del DB; quindi quando più utenti richiamano lo stesso statement SQL, ne viene utilizzata l'immagine presente nella library cache.

Nella shared pool trova spazio inoltre la **Data dictionary cache**, che mantiene le definizioni di tabelle e colonne e i rispettivi privilegi di accesso.

## - Database Buffer Cache

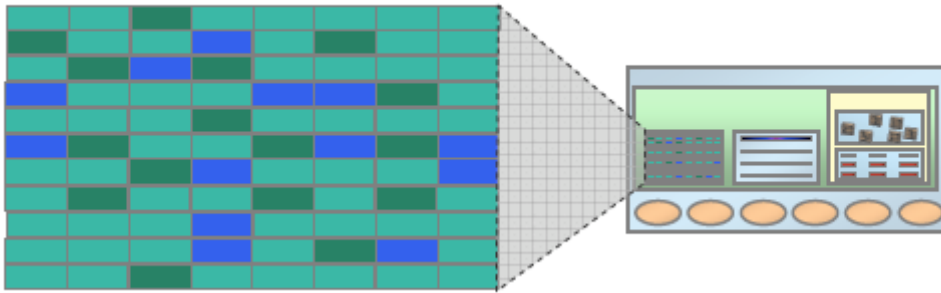


fig.3.6

Oracle, come altri sistemi software (ad esempio sistemi operativi), utilizza una buffer cache per ottimizzare la gestione dei blocchi dei dati in memoria. Infatti, questa cache tiene una copia dei blocchi letti dal file dei dati presente nel database sottostante all'istanza, di modo che, alla futura richiesta del medesimo dato, non si debba accedere al file dei dati nel DB, operazione che costa molto in termini di tempo, ma si possa leggere direttamente il dato all'interno della database buffer cache.

Tutti i processi di background dell'istanza, possono accedere a tale cache.

Il vantaggio principale nell'utilizzo di una buffer cache è che vi è una riduzione sostanziale dell'I/O fisico sui file per i blocchi più frequentemente utilizzati; inoltre l'accesso dei blocchi che sono nella cache è più veloce, e la modifica di questi garantisce il controllo della concorrenza e la consistenza del blocco.

Per organizzare i buffer nella cache di Oracle vengono utilizzate 3 diverse liste di puntatori ai singoli blocchi:

- *la Dirty list*, detta anche *write list* o LRUW, che contiene i blocchi modificati e non ancora scritti sul disco;
- *la Least recently used list*, detta anche *replacement list* o LRU, che contiene blocchi liberi, ossia non modificati e disponibili per essere riutilizzati, blocchi "pinned", ossia blocchi che al momento sono acceduti e non possono essere sostituiti, blocchi "dirty" che sono stati modificati ma non ancora spostati nella dirty list;
- *la hashed chain list*, che contiene gli stessi blocchi delle precedenti liste, ma tali blocchi sono ordinati in base al loro *data block address (DBAs)*.

Questa lista è di norma utilizzata per trovare i blocchi nella cache, infatti, quando un processo deve accedere ad un blocco, prima esamina questa lista per vedere se il blocco è già in memoria, se trovato può essere immediatamente utilizzato, altrimenti deve essere letto dal datafile nella buffer cache.

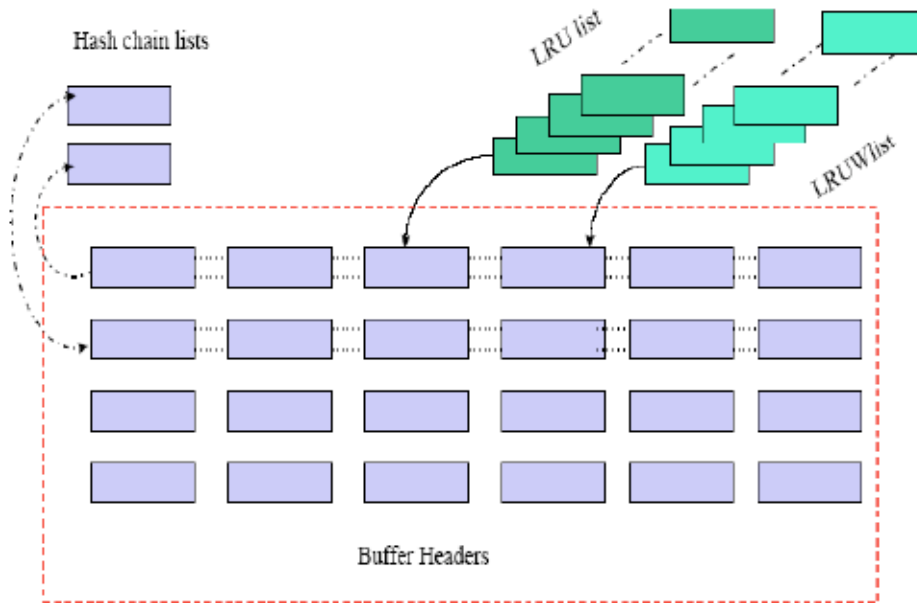


fig.3.7

### - Redo log Buffer

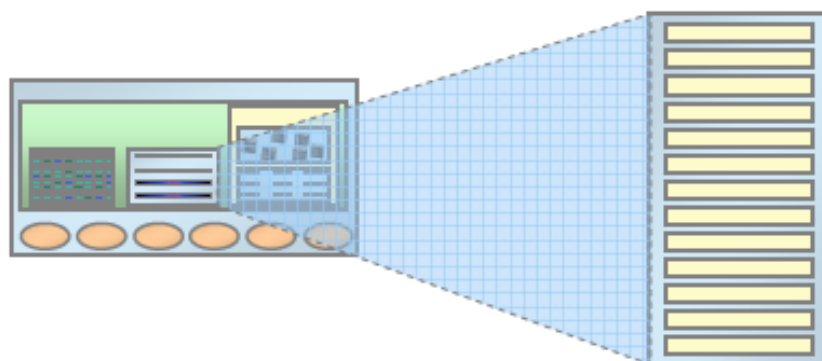


fig.3.8

Il Redo log buffer, è un buffer circolare che registra i cambiamenti fatti attraverso l'istanza, attraverso la memorizzazione dei cosiddetti change vector, contenenti l'immagine del blocco modificato prima e dopo la modifica, oltre che altri importanti parametri tra cui l'identificativo univoco di transazione, chiamato Single Change Number (SCN).

Il fatto di essere circolare comporta la sovrascrittura dei dati più vecchi, perché l'uso di tale buffer è sequenziale.

I dati che vengono messi all'interno del Redo log Buffer, vengono poi scritti all'interno dei Redo log file nel database grazie al processo dell'istanza LGWR.

La dimensione del buffer è definita dal parametro LOG\_BUFFER.

Accanto alle precedenti aree di memoria, come detto trovano spazio i processi di background:

- **DBWR** (DataBaseWRiter): è un processo di background che scrive i blocchi

che vengono modificati nella SGA nei tablespace; solitamente vengono scritti prima i blocchi usati meno di recente; quando si esegue un COMMIT (comando SQL per rendere effettive le modifiche su database) i cambiamenti sono memorizzati entro i file del redo log e quindi resi persistenti, ma non necessariamente vengono subito scritti anche nei tablespace;

- **LGWR** (LoG WRiter): è un processo di background che scrive i cambiamenti dal redo log buffer della SGA entro i file del redo log; durante un COMMIT i cambiamenti sono memorizzati entro i file del redo log; solo quando tutti i cambiamenti sono stati salvati nel file, viene segnalato il successo del COMMIT;
- **SMON** (System MONitor): è un processo di background che gestisce l'istanza; tale processo ha il compito di risolvere e segnalare le condizioni di deadlock fra le transazioni, provvedere al rilascio dei segmenti temporanei non più in uso, provvedere al ricompattamento dei tablespace, riunendo insieme le aree libere frammentate;
- **PMON** (Process MONitor): è un processo di background che gestisce i processi utente; in caso di fallimento di un processo durante una transazione provvede al rollback (meccanismo di ripristino dei vecchi dati), ripristinando per i dati coinvolti nella transazione la situazione preesistente all'inizio della transazione stessa;
- **ARCH** (ARCHiver): è un processo di background che gestisce la copia automatica dei file del Redo Log in un'apposita area di salvataggio;
- **CKPT** (ChecKPoint Process): è un processo di background che controlla i checkpoint;
- **RECO** (RECOOverer): è un processo di background che risolve i fallimenti di transazioni distribuite;
- **LCK** (Lock) compie l'azione del locking nelle transazioni intra-istanza;
- **MMON** (Memory MONitor) controlla lo status della SGA;
- **MMAN** (Memory MANager) gestisce dinamicamente la SGA ridistribuendo la memoria tra le sue componenti quando necessario;

Analizzati ora tutti gli elementi che formano un'istanza guardiamo cosa accade all'interno di questa durante il normale svolgimento delle operazioni tra client e DB server;

Quando si esegue una query sul database vengono eseguiti tre operazioni fondamentali:

- *parse*: in cui viene cercato se un comando SQL identico è già stato eseguito e il suo piano di esecuzione si trova quindi ancora in memoria, viene controllata la sintassi SQL, dei nomi degli oggetti e i relativi privilegi di accesso, viene eseguito il lock degli oggetti usati durante il parsing, e viene creato e salvato un piano di esecuzione dell'istruzione;
- *execute*: vengono identificate le righe selezionate nella query;
- *fetch*: viene restituito allo *user process* le righe selezionate nella query;

La figura 3.9 mostra l'istanza nel normale funzionamento; da qui si può vedere che la interazione con il database da parte del client inizia con una richiesta dell'*user process* al *server process* il quale interagisce direttamente con l'istanza passandogli la query; l'istanza esegue l'elaborazione della query e restituisce il risultato al *server process* il quale provvederà a restituirla all'*user process*; se è stata eseguita una istruzione DML, all'esecuzione della COMMIT il LGWR scrive all'interno del redo log file i dati memorizzati nel redo log buffer; in un tempo successivo (in modo

asincrono) il DBWR scrive all'interno dei data file i blocchi modificati nella buffer cache.

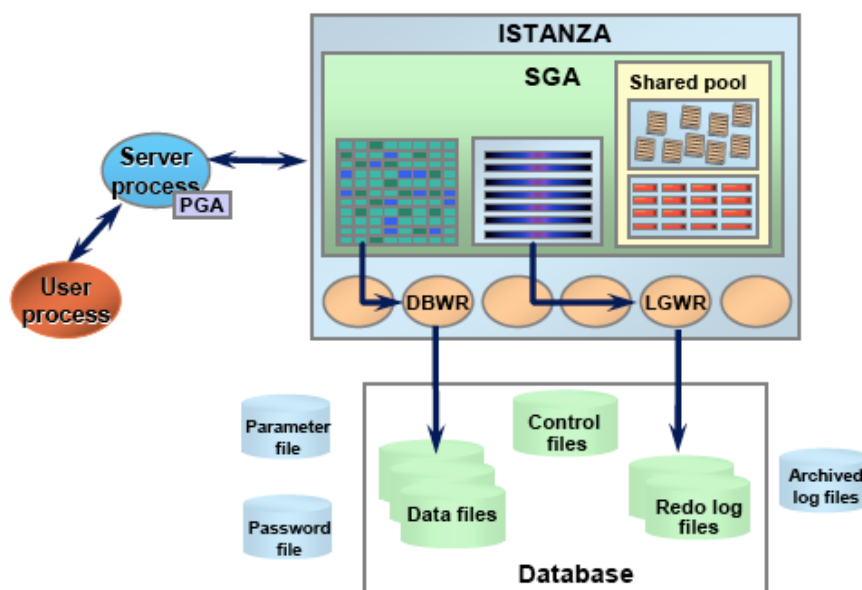


fig.3.9

## Database

Il database (abbr. DB) consiste di strutture logiche di memorizzazione, per immagazzinare e gestire i dati (tabelle, indici, ecc.), e di strutture fisiche di memorizzazione che contengono le strutture logiche;

I servizi offerti dalle strutture logiche del server sono indipendenti dalle strutture fisiche che le contengono; questo perché le strutture logiche possano essere progettate nello stesso modo indipendentemente dall'hardware e dal sistema operativo impiegati. Quindi sia che abbiamo un'installazione di server Oracle su sistema operativo Microsoft, Linux o Solaris non troveremo alcuna differenza nella progettazione delle strutture logiche di memorizzazione.

Le strutture logiche di memorizzazione sono formate da:

### Tablespace

Al vertice della scala gerarchica delle strutture logiche di memorizzazione troviamo i Tablespace, strutture che raggruppano, a loro volta, quelle di livello più basso.

Solitamente è ammissibile suddividere i dati in differenti tablespace in base ad uno specifico significato logico dettato dalla nostra applicazione che interagirà con il database. Per esempio si può creare un tablespace per ciascun utente che andrà a memorizzare i propri dati nel database.

Questa suddivisione logica ha l'enorme vantaggio di consentire l'amministrazione di una porzione limitata del database, ovvero di un tablespace, senza intaccare le funzionalità delle parti rimanenti.

All'interno di un tablespace sono memorizzati quindi uno o più file fisici (data file).

Ogni database deve avere uno o più tablespace; di default i DB Oracle hanno un tablespace chiamato SYSTEM, che contiene gli oggetti principali del database (in particolare le tabelle di sistema).

## **Blocchi**

Un blocco dati è la più piccola unità di memorizzazione in Oracle, pertanto indivisibile, e corrisponde ad un numero di byte scelto dal DBA durante la creazione del database. E' sempre multiplo della capacità del blocco usato dal sistema operativo su cui si opera.

## **Extent (estensione)**

E' composto di un numero specifico di blocchi di dati contigui.

## **Segment (segmento)**

E' formato da un insieme di extent. Ne sono un esempio le tabelle o gli indici.

Ogni qualvolta si crea un segment, Oracle alloca al suo interno almeno un extent che, a sua volta, contiene almeno un blocco. Tutto questo è sempre deciso dal DBA.

Un segment può essere associato esclusivamente ad un tablespace.

Le strutture fisiche di memorizzazione sono formate da:

## **Data File**

File che contengono tutti i dati del DB. Ogni database Oracle deve avere uno o più data file.

Ciascun data file è associato esclusivamente ad un tablespace, ma un tablespace può essere formato anche da più di un data file. Ne esiste sicuramente almeno uno per il tablespace SYSTEM.

## **Redo log file**

All'interno dei redo log vengono registrate, sotto forma di insiemi di change vector, tutte le modifiche occorse ai dati.

In particolare entro un change vector vengono memorizzate le seguenti informazioni:

- timestamp dell'inizio della transazione
- Nome identificativo della transazione
- Nome dell'oggetto coinvolto
- Situazione prima della transazione (*before image*)
- Situazione dopo la transazione (*after image*)
- Indicatori del COMMIT come lo stato della transazione e il timestamp di fine transazione.

Ogni DB possiede almeno due file di redo log, perché Oracle scrive in questi file in maniera circolare: quando un file di redo log è pieno allora Oracle scrive in quello successivo, quando l'ultimo file di redo log è pieno allora Oracle ricomincia a scrivere dal primo, assicurandosi però di memorizzare i blocchi cambiati nei data file prima di sovrascrivere i change vector associati ai cambiamenti da essi subiti.

Se una qualsiasi anomalia non permettesse di scrivere le modifiche occorse al database nei rispettivi data file, allora si può ottenere tali modifiche dai redo log file.

Le modifiche ai dati, pertanto, non sono mai perse, almeno finché i redo log file non subiscono alterazioni.

I redo log file rappresentano la cronologia delle modifiche ai dati e sono di vitale importanza per l'integrità del DB. Oracle permette di avere più di una copia per ciascun redo log file: questa importante caratteristica è denominata **multiplexing dei redo log file**.

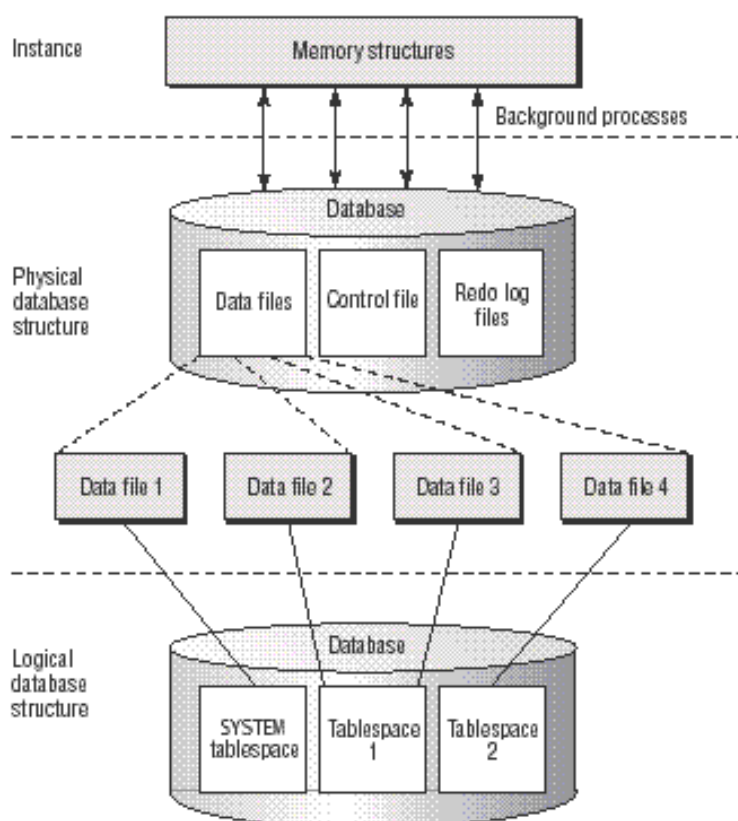
## Control file

Il control file non è altro che un piccolo file binario che contiene informazioni sulla struttura del DB. Paragonando Oracle al sistema operativo Windows, si può pensare al control file come il registry di Oracle.

In particolare il contenuto del DB file è il seguente:

- Database name e suo identificatore
- Data di creazione del database
- Locazione dei data file e dei redo log file
- Nomi dei tablespace
- Storia del log
- Informazioni per il backup
- Informazioni del Redo
- Numero del corrente log
- Informazioni del checkpoint

Ogni database ha almeno un control file. Tale file è di vitale importanza e, per questo, è consigliato configurarne più di una copia (**multiplexing di control file**).



*fig.3.10*

## *Utenti del database*

Come detto Oracle 10g è un database multiutente, il che significa che più di un utente opera sul database.

Per questo motivo l'accesso al database richiede l'invio di username e password, che identificano univocamente un utente.

Ogni utente ha il possesso di un insieme di oggetti che viene detto **schema**. Solitamente in un database è prassi comune di creare diversi utenti, ognuno dei quali ha uno specifico scopo.

Per motivi prestazionali sarebbe opportuno creare un tablespace distinto per ogni utente.

Ogni utente ha associato un tablespace di default (USERS se non impostato diversamente) per la memorizzazione delle sue tabelle, oltre che un tablespace temporaneo, dove vengono allocati spazi temporanei per operazioni lunghe e complesse.

Per default nei DB Oracle esistono due utenti che sono creati automaticamente e hanno diritto all'accesso a tutto il database:

- SYS, utente SuperDBA (DataBaseAdministrator) che è proprietario del database data dictionary; SYS può compiere ogni operazione, compresa la cancellazione totale del database;
- SYSTEM, utente DBA che è proprietario delle tabelle di sistema usate internamente.

Un utente ha associato un profilo (**profile**) ed uno o più ruoli (**role**).

Nel profilo sono contenute le regole per l'accesso dell'utente; con il profilo inoltre possono essere impostati parametri come il tempo massimo di connessione, il numero massimo di tentativi di accesso e il tempo di validità della password.

Il ruolo identifica l'insieme di azioni che il singolo utente può compiere all'interno del database.

Ogni ruolo, infatti, abilita ad un insieme di operazioni elementari che l'utente può svolgere (*system privileges*).

I ruoli possono essere creati o modificati dal DBA; per default ogni utente creato riceve il ruolo di CONNECT che lo abilita a connettersi al DB e creare propri oggetti. In seguito ad un utente possono essere attribuiti altri ruoli, tipicamente RESOURCE che abilita, tra l'altro, a compiere alcune azioni sui dati.

## 2. Versioni di Oracle 10g presenti e loro caratteristiche

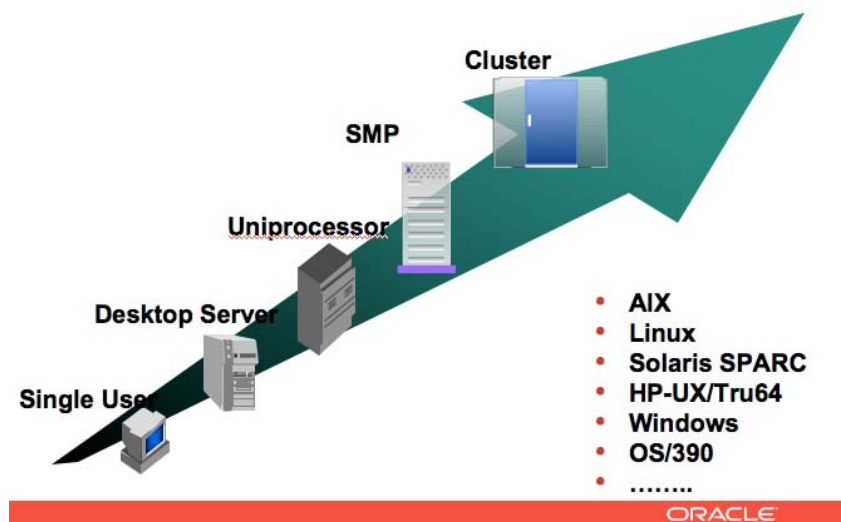


Fig.3.11

La filosofia con cui la Oracle Corporation crea le soluzioni software rivolte ai propri clienti è quella di “pensare in grande e partire in piccolo”.

Questo motto sta ad indicare che i software Oracle sono creati per un insieme molto ampio di aziende di dimensioni più svariate.

Secondo tale filosofia, infatti, le aziende di più modesta dimensione possono utilizzare sistemi più piccoli Oracle, e, man mano che le aziende aumentano la propria grandezza, adeguare tali sistemi, dato che tra tutti i sistemi Oracle c'è massima compatibilità di versione e di piattaforma.

Tale compatibilità garantisce che il passaggio da un sistema Oracle più piccolo ad un sistema più grande, e più adeguato all'accresciuto giro d'affari, sia senza dolore.

Questo concetto è ben espresso nella figura 3.11.

Per questo motivo le versioni presenti sul mercato di Oracle DB server 10g sono molteplici come illustra la figura 3.12.

Nei prossimi sottoparagrafi vediamo le caratteristiche delle varie versioni.

## La Famiglia Oracle10g

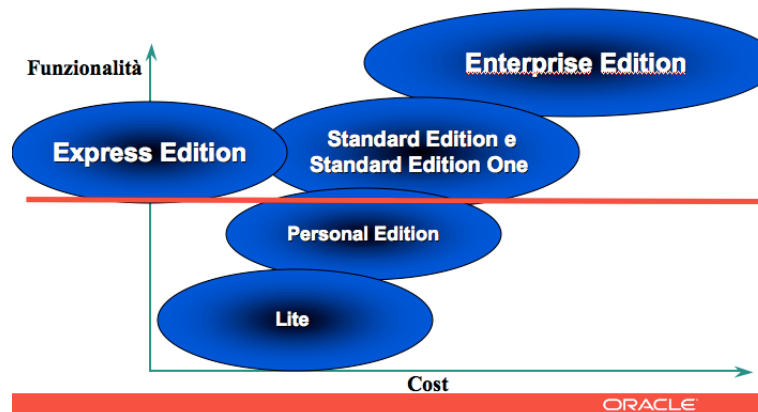


Fig.3.12

### Personal Edition

La versione personal edition è stata introdotta sul mercato da Oracle negli anni 90, come alternativa al database della Microsoft Access;

Sostanzialmente è un DB a singolo utente, che presenta il kernel dell'Enterprise Edition.

La versione è presente solo su piattaforma Windows ed è a pagamento.

### Standard Edition

Questa versione nasce dalla necessità di avere un DB che potesse avere un insieme di funzioni il più standard possibile.

La standard edition, infatti, è sviluppata a partire dal kernel dell'Enterprise Edition, ma ha la differenza che i pacchetti che si possono installare sono quelli decisi dalla Oracle e non c'è possibilità di "customizzazione".

Esiste anche la versione Standard Edition One, identica alla Standard Edition come funzionalità, che si può installare solamente su macchine aventi un solo processore.

La Standard Edition è installabile sulle maggiori piattaforme ed è a pagamento.

### Enterprise Edition

È la versione più completa e destinata alle grandi imprese. L'Enterprise Edition permette di essere personalizzata in maniera totale, poiché si ha la possibilità di attivare tutte le componenti accessorie possibili a seconda delle necessità.

È presente sul mercato nella maggior parte delle piattaforme ed è a pagamento; le licenze Oracle possono essere vendute a seconda degli utenti che utilizzeranno il software oppure a seconda del numero di processori delle macchine sulle quali andranno ad operare.

### Express Edition

L'Express Edition è una versione multiutente come la Standard e l'Enterprise, ma ha la caratteristica di essere gratuita e liberamente scaricabile.

Essa è stata introdotta da Oracle per attirare l'attenzione degli utenti dei numerosi DBMS gratuiti, come per esempio MySQL, di modo da farli avvicinare al mondo Oracle.

L'Express Edition ha però dei limiti rispetto alle altre versioni a pagamento:

- l'installazione è standard e non personalizzabile;
- utilizza solo una CPU, anche se installata su macchine con più core;
- non può avere più di 4 GB di database (inteso come spazio dei tablespaces di tutti gli utenti);
- non vengono rilasciate patch per l'eliminazione di eventuali difetti software;
- non prevede supporto tecnico Oracle, anche se vi sono forum dedicati.

### **Lite Edition**

È una versione embedded, cioè molto leggera che è destinata ad essere installata su piccoli apparecchi;

La Lite Edition non è personalizzabile né ha tutte le funzioni dell'Enterprise Edition, ma è spesso utilizzata nei PDA o Palmari per la raccolta di dati che vengono poi copiati su DB veri e propri.

## **3. Oracle server ed i componenti ausiliari**

Accanto ad Oracle DB Server 10g, esistono una serie di componenti aggiuntive o attivabili che migliorano le prestazioni del database; in questo paragrafo trattiamo brevemente quali sono quelle principali.

### **Times Ten**

Oracle Times Ten è un sistema di caching di primo livello che è affiancato ai database Oracle per migliorarne le prestazioni in termini di tempo di risposta;

Utilizzando Times Ten, quando il database si avvia, viene interamente caricato in una memoria volatile di Times Ten e lì completamente gestito; Il fatto che la memoria sia volatile implica che sia anche molto più veloce rispetto alla memoria del database, e proprio per questo le risposte alle interrogazioni sono in tempo reale.

Times Ten dà la possibilità di fare caching sia in uscita, cioè risponde in tempo reale alle query, sia in entrata, cioè riceve e memorizza temporaneamente i dati modificati dagli utenti, che poi saranno copiati nei file del database per renderli permanenti.

Per fare in modo che Times Ten funzioni da cache del DB occorre che sia installata sulla stessa macchina dove è installato il DB.

Times Ten è utilizzata da grandi aziende e nelle situazioni in cui il tempo reale di risposta è essenziale.

### **Automatic Storage Management (ASM)**

L'ASM è una feature di Oracle 10g che permette di gestire automaticamente lo Storage dei dati (dischi, file dei dati ecc.).

L'obiettivo dell'ASM è quello di gestire e ottimizzare in maniera trasparente agli utenti la memorizzazione dei dati nei dischi.

I dischi vengono raggruppati in unità logiche chiamate *Disk Group*.

All'interno di tali *Disk Group* possono essere presenti più dischi che vengono gestiti in automatico dall'ASM.

Un punto di forza dell'Automatic Storage Management è il *Dynamic Rebalancing*;

Tale meccanismo permette di inserire un nuovo disco all'interno di un *Disk Group*, in caso ad esempio in cui gli altri dischi dello stesso *Disk Group* siano quasi saturi, e di

bilanciare il carico dei dati su tutti i dischi del disk group, compreso quello nuovo, tutto questo a database funzionante.

L'ASM permette inoltre di gestire i cosiddetti *hot-spot*, ossia dati che vengono richiesti molto frequentemente; se infatti in un disco vengono memorizzati dati chiesti molto frequentemente, questo è sottoposto ad una grossa mole di lavoro, che, nel tempo, può portare a rotture dello stesso. ASM distribuisce i blocchi più richiesti su tutti i dischi del disk group, di modo che venga distribuito anche il carico di lavoro.

### **Oracle Data Guard**

Oracle Data Guard è una componente chiave della strategia di *Business Continuity* che la Oracle Corporation si prefigge di offrire ai suoi clienti, che consente ai sistemi basati su Oracle di operare ininterrottamente 24 ore su 24.

Questo servizio prevede un ampio insieme di funzionalità di protezione dei dati e di *Disaster Recovery* che consentono alle aziende di sopravvivere alle calamità, agli errori umani e ai danneggiamenti che possono avere ripercussioni negative sul Database Oracle.

Oracle Data Guard prevede la presenza di un'istanza di lavoro (detta **Primary**) ed una o più istanze dormienti (dette **Standby**) che vengono continuamente aggiornate e mantenute allineate rispetto all'istanza Primary per poterne prendere il posto in caso di fail.

Nella normale modalità di lavoro l'istanza *Primary*, attraverso il processo DBWR, scrive sui datafile (che contengono le tabelle e gli indici della base dei dati) ed, attraverso il processo LGWR, riporta sui log i blocchi modificati. Quando si è in modalità *archive log* inoltre il processo ARCH si occupa di salvare i Redo Log su una directory contenente gli Archived Redo Log.

Se viene configurato il Data Guard, il processo ARCH si occupa anche di trasferire, mediante i servizi NetX, gli archived Redo Log sulle istanze di *Standby*.

Tali istanze sono aperte in modalità di recover ed applicano quindi il contenuto dei Redo Log mantenendo sincronizzati i dati.

Le istanze possono essere ospitate sullo stesso sistema oppure distribuite in rete geografica.

Le transazioni occorse sul Primary vengono ribaltate mediante la trasmissione e l'applicazione dei Redo Log. L'applicazione dei Redo log può avvenire in due diversi modi:

- **Redo Apply:** i Redo Log vengono applicati sull'istanza in standby che è identica alla Primary; in questo caso si parla di **Physical Standby Database**.
- **SQL Apply:** i Redo Log vengono convertiti in SQL ed applicati sull'istanza di Standby; si parla in questo caso di **Logical Standby Database**

La protezione offerta da Data Guard può essere configurata in modo da massimizzare uno dei seguenti obiettivi:

- **Sicurezza:** il COMMIT si conclude quando il file di redolog è stato inviato ad almeno un'istanza di Standby. Se non vi è alcuna istanza di Standby in grado di raccogliere le transazioni, l'istanza Primary si blocca. In questo modo ogni transazione è replicata in almeno due basi di dati distinte.
- **Disponibilità:** molto simile al caso precedente ma nel caso di blocco delle istanze di Standby, o del servizio di rete che trasferisce i Redo Log, le attività sul Primary database proseguono ugualmente. In questo modo si rende massima la disponibilità dei dati.
- **Prestazioni:** i Redo Log vengono trasferiti al momento dell'archiviazione privilegiando così le prestazioni del sistema. In questa modalità in caso di

perdita dell'istanza principale il recovery avverrà fino all'ultimo Redo Log archiviato.

Nelle prime due configurazioni è il processo LGWR ad effettuare l'invio dei Redo Log all'istanza di Standby. Nell'ultima configurazione (che è quella di default) è generalmente il processo ARCH.

Il passaggio di un'istanza dal ruolo di Standby al ruolo di Primary avviene a causa di una delle seguenti due condizioni:

- **Switchover**: quando è il DBA che richiede il passaggio;
- **Failover**: in caso l'istanza Primary non sia disponibile;

## 4. Gli strumenti di sincronizzazione di Oracle 10g:

### i. DataBase Link

Tra i vari metodi di sincronizzazione tra database Oracle, i DB link rappresentano sicuramente il metodo più veloce e semplice per eseguire query su dati condivisi.

Un DB link è un puntatore che definisce un percorso di comunicazione unidirezionale da un DB Server Oracle ad un altro DB server.

Tale puntatore del link viene definito come elemento di una tabella del dizionario dei dati del DB di partenza, e punta ad una tabella del DB di destinazione.

Per accedere al link, è necessario aver eseguito la connessione al DB locale contenente l'elemento del dizionario dei dati.

Il DB link è unidirezionale, ossia è possibile leggere i dati condivisi da un DB verso l'altro, ma non il contrario; questo poiché un client connesso al DB locale A può utilizzare un DB link memorizzato nel DB A per accedere ai dati nel DB remoto B, mentre gli utenti del DB B non possono utilizzare lo stesso link per accedere al DB A (al limite devono anche loro creare un DB link da B ad A).

Per stabilire questo tipo di connessione bisogna che a ciascun DB del sistema distribuito sia assegnato un nome di DB globale univoco. Gli utenti possono accedere agli oggetti di un altro utente in un DB remoto vincolati al set di privilegi del proprietario dell'oggetto, ossia un utente locale può accedere ad un DB remoto senza essere un utente del DB remoto. Le informazioni sui link a cui l'utente può accedere sono nella vista del dizionario dati `USER_DB_LINKS`.

Il vantaggio principale dell'utilizzo dei DB link è senza dubbio quello di poter disporre di dati che non sono memorizzati nel DB locale, ma sono mantenuti nel DB remoto, senza avere la necessità di trasferirli fisicamente facendo un *EXPORT* nel remoto e un *IMPORT* nel locale.

Utilizzando i DB link è possibile inoltre creare delle modifiche transazionali su entrambi i DB: quando si esegue una `COMMIT`, che sancisce la fine di una transazione con il DB, le modifiche appaiono contemporaneamente su entrambi i database; Ciò è utile in una situazione in cui, ad esempio, si hanno il DB degli ordini e il DB della banca che registra la situazione economica; quando si esegue un nuovo ordine lo si registra nell'opportuno DB e automaticamente si può fare in modo che venga scalato dal conto l'importo di tale ordine.

I DB link e la loro sintassi sono descritti più in dettaglio nel capitolo 5.

## ii. Oracle Streams

Oracle Streams rappresenta un'infrastruttura flessibile che può essere usata per semplificare la condivisione di informazioni tra due Database Oracle.

Oracle Streams consente di condividere modifiche al database e altre informazioni in un flusso, per la propagazione di eventi all'interno di un database o da un database a un altro.

Esso, infatti, cattura le modifiche del database, le mette in un'area per poterle applicare su più istanze, le propaga ad uno o più database di destinazione e le applica.

Utilizzando Oracle Streams, sistemi Enterprise possono catturare, propagare ed applicare le informazioni come segue:

- All'interno di un database Oracle
- Tra due database Oracle
- Tra più database Oracle
- Tra un db Oracle ed un database non Oracle

Oracle Streams inizia con il catturare i cambiamenti. Tali cambiamenti (dati, tabelle, etc) che hanno luogo in un database, sono registrati nei Redo Log files.

Il processo di "*Streams capture*" estrae questi cambiamenti dai Redo Log e li formatta in un "**logical change record**" (LCR). Gli LCR vengono storicizzati in una coda (*staged*).

Successivamente, Oracle Streams propaga gli LCR da una coda (la "*producer queue*") all'altra (la "*consumer queue*") e applica (o "*consume*") gli LCR dalla "*consumer queue*" al database di destinazione.

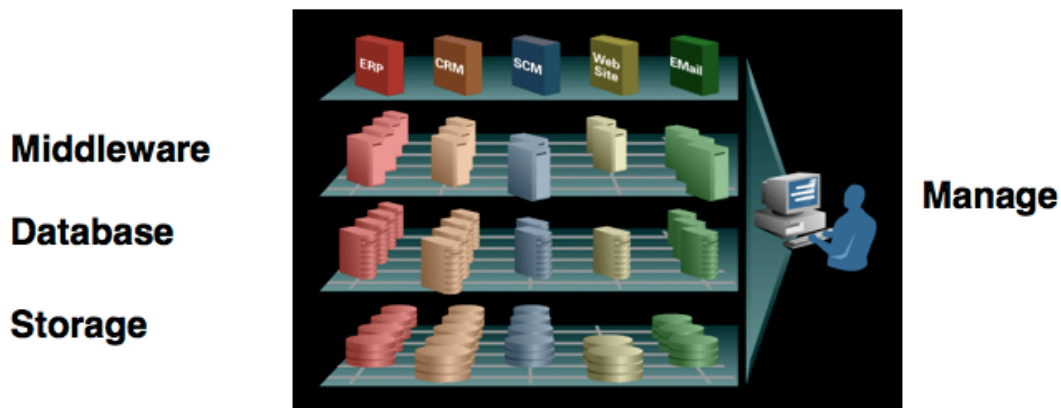
Ne risulta una funzione che offre funzionalità e flessibilità maggiori rispetto alle soluzioni tradizionali per il recupero e la gestione delle informazioni e per la condivisione delle informazioni con altri database e applicazioni.

Gli Streams possono essere utilizzati per eseguire compiti come:

- Data replication
- Estrazione e caricamento di Data warehouse
- Notifica degli eventi
- Message queuing
- Migrazione di piattaforma del database
- Upgrade di database ed applicazioni

Esamineremo in dettaglio l'utilizzo degli Oracle Streams per il Data replication nel capitolo 6.

### iii. Oracle Grid functions



## Oracle Enterprise Grid Computing

ORACLE

Fig.3.13

Prima di descrivere in che modo Oracle implementa il paradigma di Grid computing, vediamo cosa esso significa [Chappell 2004] e [GRID 2005].

Il Grid computing è stato in origine concepito ed introdotto per la comunità accademica e a fini di ricerca. Analogamente ad Internet, scaturito, in origine, dalle esigenze di comunicazione dei ricercatori scientifici dislocati sul territorio, il Grid computing nasce dalla necessità della comunità scientifica di creare un ambiente di elaborazione dinamico per la condivisione di risorse e risultati, in modo scalabile con il tempo, mantenendo contenuti i costi.

Tale concetto di Grid Computing è stato poi ripreso ed adattato alla situazione aziendale, dove occorre un'infrastruttura che potesse rendere flessibili le risorse IT, dislocate nelle varie isole di elaborazione.

Negli odierni ambienti aziendali, infatti, i sistemi informativi sono divisi in varie aree funzionali (ambienti diversi dedicati a singole applicazioni) e fanno un uso inefficiente delle risorse, sono lenti nel cambiare e richiedono alti costi di gestione.

Il Grid computing aziendale risolve questi problemi fornendo un'infrastruttura software adattabile, che fa un uso efficiente di server e basso costo e delle soluzioni di storage modulare, in grado di bilanciare i carichi di lavoro in maniera efficiente e di garantire una vera *capacità on demand* (capacità di elaborazione nel momento del bisogno).

L'idea centrale del Grid è l'elaborazione intesa come un servizio, in cui l'effettiva ubicazione fisica dei dati, degli applicativi o del computer utilizzato per elaborare una richiesta, non dovrebbe condizionare in alcun modo il risultato delle operazioni di elaborazione. L'utente deve essere comunque e sempre in grado di richiedere informazioni o calcoli e di ricevere i risultati di queste operazioni.

Tale servizio deve essere simile, ad esempio, all'erogazione di energia elettrica: l'utente non ha idea di dove si trovi il generatore o di come sia stata realizzata la rete elettrica, ma, al contrario, si limita a richiedere elettricità e a ottenerla.

L'obiettivo del Grid, quindi, è proprio quello di rendere le risorse ubiquitarie; da qui la sua denominazione di "grid" o griglia.

Il Grid computing aziendale coordina l'uso di cluster di macchine per creare una singola entità logica, come un database, ma anche un application server.

Poiché viene implementata una singola entità logica su più macchine, le società possono aumentare o ridurre la capacità in piccoli incrementi mirati, direttamente online.

Grazie alla possibilità di aggiungere capacità *on demand* a una particolare funzione, le aziende possono beneficiare di una maggiore flessibilità per adattarsi ai picchi di lavoro, conseguendo così un migliore utilizzo dei componenti hardware e tempi di risposta ottimali per il business.

Perché un'infrastruttura si possa definire di Grid computing occorre che abbia i seguenti attributi:

- **Virtualizzazione** a ogni livello dello stack di elaborazione; questo è inteso come la trasposizione di ogni entità fisica e logica di un Grid in un servizio; in questo modo si evitano i legami fissi tra le risorse e questo fa in modo che queste possano reagire più rapidamente al variare delle circostanze e di adattarsi senza compromettere le prestazioni del sistema.
- **Allocazione dinamica** del lavoro; questo significa semplicemente distribuire gli elementi da fornire dove essi sono necessari. Nel contesto del Grid, questo è implementato da un gestore di servizi che conosce i requisiti in termini di risorse di un elemento del Grid e la disponibilità di risorse di un altro elemento, quindi provvede a collegare i due in modo automatico e dinamico per consentire un uso efficiente delle risorse. In altre parole questi procede a regolare le associazioni tra elementi al variare delle circostanze.
- **Pooling delle risorse**; questi è necessario perché i Grid giungano a un migliore utilizzo delle risorse, fattore chiave per contribuire alla riduzione dei costi. Grazie alla riunione dei singoli dischi in batterie di storage, ad esempio, oppure di server individuali in blade farm, i processi runtime del Grid, che accoppiano dinamicamente i consumatori e i fornitori di servizi, possiedono una maggiore flessibilità nell'ottimizzare le associazioni.
- **Software autoadattabile**, con ottimizzazione e correzione automatica; questi è necessario perché non è possibile far funzionare l'infrastruttura Grid se ogni suo singolo nodo richiede costanti ottimizzazioni e interventi manuali; per questo motivo è necessario un software in grado di automatizzare gran parte delle numerose operazioni di manutenzione e messa a punto tradizionalmente eseguite dal personale IT.
- **Unificazione della gestione** e di allocazione delle risorse; l'intervento del personale di amministrazione sul sistema Grid risulterà semplificato se eseguito da un tool unificato in grado di eseguire l'allocazione, il monitoraggio e l'amministrazione di tutti gli elementi del Grid.

Con i nuovi applicativi 10g, Oracle permette alle aziende di iniziare agevolmente l'evoluzione verso il modello del Grid computing. Queste nuove tecnologie, soddisfano i requisiti del Grid computing, precedentemente elencati, per storage, database, application server e applicativi.

In particolare Oracle database 10g presenta parecchie funzionalità specifiche per gli ambienti Grid dette appunto **Grid functions**.

Di seguito vengono elencate le principali Grid Functions che sono implementate in Oracle Database 10g:

### **Real Application Clusters (RAC)**

Oracle Real Application Clusters (RAC) è un'infrastruttura, già presente nella versione Oracle 9i, che permette di eseguire un singolo database su un cluster di più macchine.

In sostanza il RAC non è altro che un insieme di istanze Oracle che operano su macchine diverse, il cluster appunto, ma che condividono lo stesso database.

In questo modo le istanze condividono gli stessi dati pur girando in macchine separate collegate tra loro.

Quando un utente esterno si vuole connettere al database vede questo insieme di istanze come un'unica istanza, con la potenza di elaborazione pari alla somma delle potenze di ogni macchina appartenete al cluster.

La caratteristica più importante del RAC è che garantisce la possibilità al database di aggiungere e bilanciare il carico di lavoro su un nuovo nodo con nuova capacità di elaborazione, oppure rinunciare a una macchina quando non è più necessaria, durante la normale esecuzione. (questo è il concetto di capacità elaborativa *on demand*).

### **Automatic Storage Management (ASM)**

Come detto nel precedente paragrafo, l'ASM semplifica la gestione dello storage per i database Oracle perchè ne permette la virtualizzazione. Astraendo i dettagli di gestione dello storage, viene migliorato l'accesso ai dati tramite una ripartizione molto sofisticata, semplificando, così, il lavoro del DBA.

Invece di gestire molti file di database, il DBA devono intervenire solo su un numero ridotto di gruppi di dischi.

Automatic Storage Management permette inoltre di bilanciare l'I/O di un database su tutti i nodi del cluster.

### **Database autogestito**

Il primo passo verso la gestione semplificata dell'ambiente Grid consiste nel rendere ciascun singolo sistema più indipendente dagli interventi umani.

Oracle 10g, grazie al suo innovativo database autogestito, riduce il numero di operazioni di manutenzione e ottimizzazione degli amministratori.

Oracle Database 10g include inoltre un'infrastruttura intelligente che invia "istantanee" dei dati statistici e sul carico di lavoro agli amministratori per la loro analisi.

Il database consente di diagnosticare automaticamente una serie di problemi, tra cui carenze di gestione dei collegamenti, contesa dei blocchi e prestazioni SQL scadenti.

Oracle Database 10g è in grado di risolvere autonomamente numerosi dei problemi diagnosticati e, negli altri casi, di suggerire ai DBA le giuste misure correttive.

#### iv. Uso dello strumento Oracle SQLDeveloper

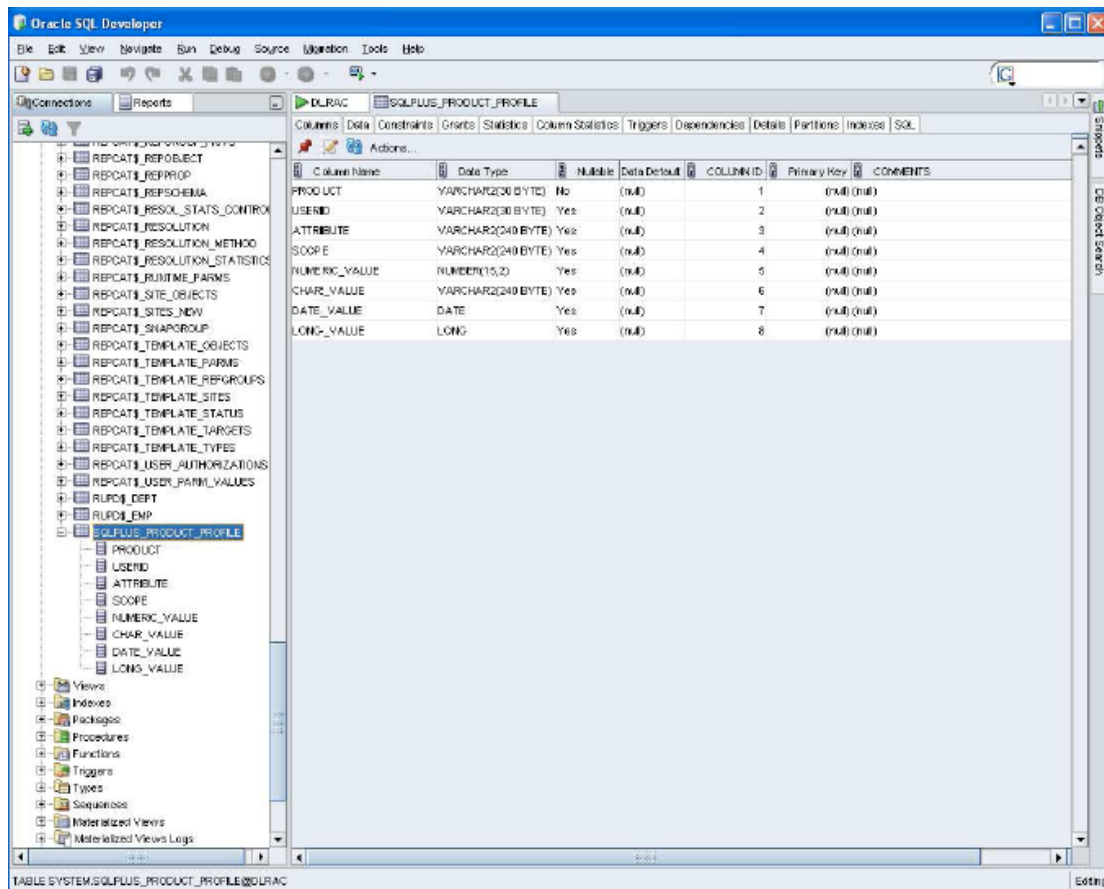


Fig.3.14

Oracle SQL Developer è un tool grafico completamente gratuito che la Oracle Corporation mette a disposizione di DBA e programmatori per l'iterazione con i DB Oracle 9i o superiori.

Questa applicazione permette di creare più connessioni sia con più database Oracle, sia anche con DBMS di altri vendor.

Inserendo il nome utente con il quale si ci vuole connettere, la relativa password e il SID del database relativo SQL Developer permette di creare una connessione al database e di memorizzarla per l'accesso futuro.

Una volta connessi al database vengono mostrati tutti gli oggetti che appartengono allo schema dell'utente con il quale si ci è connessi.

SQL Developer permette di fare molte operazioni sugli oggetti di un database, come, ad esempio, la creazione automatica degli script SQL necessari alla creazione di qualsiasi oggetto presente nel DB.

Tra le varie operazioni fattibili vi è anche quella che permette la sincronizzazione tra gli oggetti di due schemi di database diversi.

Tale funzione è la funzione **Schema Diff**, che permette di comparare due schemi appartenenti a database diversi (ma anche allo stesso database).

In pratica, tale funzione, non fa altro che paragonare gli oggetti che si è scelto, appartenenti ad uno schema di un utente (*schema sorgente*), con gli oggetti di un altro utente (*schema destinazione*), che può appartenere allo stesso database, ma anche a database diversi.

Il risultato di tale comparazione è una serie di comandi SQL, salvabili anche sotto forma di script SQL, la cui esecuzione porta alla creazione nello schema di destinazione degli oggetti che appartengono allo schema sorgente, ma che non appartengono allo schema di destinazione.

In questo modo una volta eseguiti i comandi SQL creati dalla funzione Schema Diff, lo schema dell'utente indicato come schema di destinazione avrà oltre ai propri oggetti anche quelli che appartenevano allo schema di destinazione e che prima non aveva.

Per fare un paragone matematico la funzione Schema Diff, come dice il nome stesso, esegue la differenza tra lo schema sorgente e lo schema destinazione.

Anche se questa non è la soluzione più elegante è sicuramente la più pratica e veloce per la replica di moli di dati di piccole dimensioni.

Oltre alle tabelle è possibile replicare anche altri tipi di oggetti come funzioni, viste, package e procedure).

Per un maggior dettaglio dell'uso dell'applicativo SQL Developer come strumento per la sincronizzazione vedere il capitolo 8.

## Capitolo 4

# Il problema proposto: sincronizzazione tra database centrale e database remoto

Nei primi tre capitoli, finora trattati, sono stati introdotti i concetti fondamentali per lo studio e la comprensione del problema principale affrontato in questa tesi: **la sincronizzazione tra un database centrale ed uno remoto, sfruttando il più possibile strumenti ausiliari del database automatici, per minimizzare la necessità di scrivere procedure SQL e/o applicativi esterni ad hoc.**

In questa seconda parte (capitolo 4 e successivi) viene descritto dettagliatamente l'esperimento pratico realizzato durante il periodo di tirocinio per l'analisi del problema proposto.

## 1. Il contesto del problema

La prima cosa da fare, quando si vuole studiare un determinato problema, è descrivere il contesto di partenza, ossia l'insieme di elementi che lo identificano.

Come già detto, la situazione "ideale" da prendere in considerazione come *modello teorico* del problema proposto è quella del **commesso viaggiatore**.

In essa sono presenti due entità fondamentali: il **database centrale**, ed il **database remoto**.

Il database centrale è il database sul quale, l'azienda per cui il commesso viaggiatore lavora, entro cui sono registrati tutti i dati di interesse.

Il database remoto è il database sul quale il commesso viaggiatore memorizza i dati che raccoglie nei suoi vari spostamenti e dove sono presenti anche i sottoinsiemi di dati di interesse per il commesso estratti dal database centrale (ad esempio prezzi di singoli prodotti).

Il commesso viaggiatore alla fine di ogni suo "viaggio" deve riversare i dati raccolti nel database centrale dell'azienda, aggiornando nel contempo i dati in proprio possesso (come ad esempio i listini prezzi) e deve potere compiere tale operazione sia dall'interno della rete aziendale, sia da remoto.

Si devono studiare i vari metodi di sincronizzazione tra il database centrale e quello remoto, tenendo presente che la comunicazione tra i due database deve avvenire in entrambe le direzioni, perché i dati devono poter essere passati, sia dal centrale al remoto, sia viceversa, in base alle necessità specifiche.

## 2. Applicazioni pratiche del problema

Quanto appena descritto nel paragrafo precedente rappresenta il *modello teorico* del problema che si vuole risolvere.

Per poter analizzare i vari metodi di sincronizzazione tra database occorre passare dal modello teorico al relativo *modello pratico* che ne descrive le proprietà in maniera tangibile.

Tale modello pratico rappresenta la proiezione nella realtà del problema, che per sua natura è puramente teorico.

Per rappresentare la situazione del commesso viaggiatore (modello teorico), quindi, si possono utilizzare due database installati su altrettanti host differenti e collegati in rete (implementazione pratica).

Tali database fungeranno uno da db centrale, e l'altro da db remoto. Per rendere più veritiera l'implementazione pratica del problema è opportuno che il db remoto sia di dimensioni più piccole rispetto a quello centrale, e inoltre abbia minori funzionalità, ovvero appartenga ad una serie meno costosa o addirittura gratuita come Oracle Express.

Lo scopo del tirocinio da me svolto è stato proprio quello di realizzare un modello pratico che rappresentasse al meglio la situazione del commesso viaggiatore e di studiare i principali metodi di sincronizzazione tra i due database coinvolti.

Nel prossimo paragrafo vengono descritte le varie scelte implementative e le operazioni eseguite durante l'esperimento descritto in questa tesi.

## 3. Lo schema di esempio: operazioni da compiere

L'esperimento sviluppato durante il periodo di tirocinio, per passare dal *modello teorico* del problema al *modello pratico*, può essere riassunto descrivendo lo schema logico che si è seguito per realizzarlo, che consta delle seguenti fasi principali:

- *Realizzazione dell'ambiente operativo;*
- *Realizzazione dello schema su cui eseguire i test;*
- *Individuazione delle principali soluzioni di sincronizzazione tra database con tecnologia Oracle;*
- *Implementazione delle soluzioni individuate;*
- *Test operativo di ognuna delle soluzioni e annotazione dei risultati;*

Ognuna di queste fasi rappresenta un insieme di operazioni che sono state eseguite per la realizzazione dell'esperimento. Di seguito vengono riportate dettagliatamente tutte queste operazioni.

### *Realizzazione dell'ambiente operativo*

Questa fase costituisce le "fondamenta" del progetto. E' qui infatti che si è creato l'ambiente su cui poi si è andato ad operare, e si sono definiti gli strumenti utilizzabili.

In questa fase sono stati scelti ed installati i software necessari e sono stati realizzati i canali di comunicazione tra i database.

Nell'esperimento si è scelto di utilizzare la tecnologia Oracle per i database; La motivazione di questa scelta è da ricercare nel fatto che i database Oracle incorporano un grosso numero di funzioni per la condivisione di dati.

Come database centrale è, infatti, stato utilizzato il DB **Oracle Database 10g** nella versione **Standard Edition** installato su piattaforma *Windows XP Home Edition Sp.2*, mentre come database remoto è stato utilizzato il DB **Oracle Database 10g** nella versione **Express Edition** su sistema *Windows XP Home Edition Sp.2*.

I due database sono stati installati su altrettanti host collegati da una rete cablata LAN (Local Area Network) Ethernet, per consentire il passaggio dei dati tra un DB e l'altro. La presenza da tempo sul mercato di numerosi software per reti private virtuali (VPN) [Destri 1998] rende possibile trasferire facilmente un modello di questo tipo sviluppato su rete locale ad una rete geografica, praticamente senza cambiamenti diretti sui database coinvolti e sugli strumenti utilizzati.

Per la messa in opera del sistema è stato necessario eseguire le seguenti due operazioni:

- Assegnazione di indirizzi IP fissi per ognuno dei due host della rete; nell'esperimento sono stati assegnati gli indirizzi 192.168.0.1 all'host "centrale" (per host "centrale" si intende l'host su cui è installato il database centrale e i client SQL) e 192.168.0.2 all'host "remoto"(quello sul quale è stato installato il database remoto);
- Apertura della porta 1521 (la porta utilizzata dal *listener* di Oracle) nel firewall di Windows XP di entrambi gli host, per garantire lo scambio di pacchetti tra i database.

L'installazione dei database Oracle è molto intuitiva grazie all'utilizzo dell'**Oracle Universal Installer**, un programma Java che gestisce l'installazione dei sistemi Oracle e che ha lo stesso look-and-feel su tutte le piattaforme.

In particolare l'installazione del database Standard Edition permette di installare sull'host in questione, sia il DB server che il client (SQLPlus) anche separatamente.

L'installazione del DB Express Edition, invece, installa solamente il database server senza installare altri programmi che funzionino da client.

Come client di database server possono essere usati programmi indipendenti dal database, come *SQLPlus*, *SQL Worksheet*, *SQL Developer* [Oracle10gMan], [Alapati 2005], che permettono il collegamento a qualsiasi DB server Oracle, oppure la console Web del database, una applicazione Java installata insieme ai database Oracle 10g a cui è possibile accedere da qualsiasi browser. All'interno di tale console è possibile gestire l'istanza, personalizzare il database ed eseguire comandi SQL.

Solitamente, nella pratica, l'interazione con il database Express avviene per mezzo della sua specifica console web, grandemente semplificata rispetto a quella dello Standard, mentre l'interazione con lo Standard avviene più frequentemente per mezzo di programmi client veri e propri; non c'è, tuttavia, un modo oggettivamente più conveniente per operare con i database; tutto dipende dalla familiarità che si ha con i vari strumenti.

Da notare che i programmi client SQL, utilizzati solitamente per l'esecuzione di comandi sul database, possono essere installati solamente su di uno dei due host (nell'esperimento quello centrale). Questo perché i client permettono di connettersi sia ai DB server che stanno sia sulla stessa macchina (`localhost= 127.0.0.1`), sia su macchine collegate in rete.

E' quindi possibile eseguire query SQL utilizzando, ad esempio, SQLPlus su entrambi i database server, fornendo semplicemente l'indirizzo IP, il numero di porta e il nome univoco del database sul quale si vuole eseguire.

Tra i parametri richiesti durante l'installazione (per una documentazione più accurata sull'istallazione si veda [manuali Oracle]) vi è il nome dell'istanza o **SID**: questo è il nome univoco che verrà utilizzato ogni volta che ci si vorrà connettere ad un'istanza; nell'esperimento è stato dato il SID `oracle` all'istanza del database Standard Edition ed il SID `xe` all'istanza dell'Express Edition.

Se tutto è stato fatto correttamente alla fine delle precedenti operazioni si ha un ambiente perfettamente funzionante in cui si hanno due database collegati in rete capaci di memorizzare i dati e comunicare con i client e tra di loro.

Questa situazione rappresenta il punto di partenza dell'esperimento che ha come scopo quello di individuare, realizzare e verificare i metodi di comunicazione tra questi due database.

E' importante sottolineare che il fatto di utilizzare due database con la stessa tecnologia costruttiva del motore ma con funzionalità diverse, come sono lo Standard e l'Express Edition di Oracle 10g, restringa molto il campo delle soluzioni di comunicazione applicabili.

Infatti se lo stesso esperimento che stiamo affrontando fosse stato eseguito utilizzando, ad esempio, due database Oracle Enterprise Edition, si avrebbe avuto un numero decisamente maggiore di metodi di comunicazione, derivante dal numero maggiore di funzionalità che sono implementate in tali database.

#### *Realizzazione dello schema su cui eseguire i test*

In questa fase i due database Oracle vengono popolati di oggetti che serviranno come campione per eseguire i test.

Come prima cosa è stato creato un utente in entrambi i database, che verrà utilizzato per tutto l'arco dell'esperimento:

Attraverso l'Oracle Enterprise Manager Console, un client per l'amministrazione dei DB Oracle, è stato creato in entrambi i database un nuovo utente, il cui schema servirà per l'esecuzione dei test. A tale utente è stato dato il nome AREA.

In entrambi i database, quindi, è stato creato uno schema relativo al nuovo utente con il nome AREA.

Una volta fatto questo, si è proceduto a creare oggetti all'interno dello schema AREA, di ciascun database, attraverso l'esecuzione di comandi SQL.

In particolare sono stati creati:

- Tre *tabelle*: `CR_MAIN_DATA`, `CR_ADDRESS`, `CR_E_ADDRESS` che conterranno i record da sincronizzare;
- Tre *sequenze*: `S_CR_MAIN_DATA`, `S_CR_ADDRESS`, `S_CR_E_ADDRESS`, che permettono di incrementare di un certo valore un contatore;
- Tre *trigger*: `TR_S_MAIN_DATA`, `TR_S_CR_ADDRESS`, `TR_S_CR_E_ADDRESS`, che permettono di associare ad un evento un determinato vincolo;

In particolare le sequenze e i trigger sono stati costruiti per fare in modo che il campo "ID" di ciascuna delle tre tabelle incrementi di un valore ogni volta che si aggiunge un record automaticamente.

Vediamo i comandi SQL utilizzati:

- **Per la creazione delle tre tabelle:**

*Tabella CR\_E\_ADDRESS:*

```
CREATE TABLE "CR_E_ADDRESS"  
  ("ID" NUMBER(11,0),  
   "ID_CR_MAIN_DATA" NUMBER(11,0) DEFAULT 0,  
   "ID_ALL_TYPE_E_ADDRESS" NUMBER(11,0) DEFAULT 0,  
   "VALUE" VARCHAR2(50),  
   "NOT_USING" NUMBER(1,0) DEFAULT 0,  
   "MAIN" NUMBER(1,0) DEFAULT 0,  
   "SORTING" VARCHAR2(50)) ;  
  
ALTER TABLE "CR_E_ADDRESS" ADD CONSTRAINT  
"CR_E_ADDRESS_PK" PRIMARY KEY ("ID") ENABLE;
```

*Tabella CR\_ADDRESS:*

```
CREATE TABLE "CR_ADDRESS"  
  ("ID" NUMBER(11,0),  
   "ID_CR_MAIN_DATA" NUMBER(11,0) DEFAULT 0,  
   "DESCRIPTION" VARCHAR2(255),  
   "ADDRESS" VARCHAR2(100),  
   "ZIPCODE" VARCHAR2(10),  
   "CITY" VARCHAR2(100),  
   "PROVINCE" VARCHAR2(10),  
   "STATE" VARCHAR2(50),  
   "GEO_COORD" VARCHAR2(255),  
   "NOT_USING" NUMBER(1,0) DEFAULT 0,  
   "IS_MAIN" NUMBER(1,0) DEFAULT 0,  
   "SORTING" VARCHAR2(50)) ;  
  
ALTER TABLE "CR_ADDRESS" ADD CONSTRAINT  
"CR_ADDRESS_PK" PRIMARY KEY ("ID") ENABLE;
```

*Tabella CR\_MAIN\_DATA:*

```
CREATE TABLE "CR_MAIN_DATA"  
  ("ID" NUMBER(11,0),  
   "ID_ALL_TYPE_TITLE" NUMBER(11,0) DEFAULT 0,  
   "COMPANY_NAME" VARCHAR2(60),  
   "NAME" VARCHAR2(40),  
   "SURNAME" VARCHAR2(40),  
   "NICKNAME" VARCHAR2(40),  
   "VAT_NUMBER" VARCHAR2(20),  
   "FISCAL_CODE" VARCHAR2(20),  
   "IS_PERSON" NUMBER(1,0) DEFAULT 0,  
   "BIRTH_DATE" DATE,
```



```

BEFORE INSERT ON CR_E_ADDRESS
FOR EACH ROW
BEGIN
SELECT S_CR_E_ADDRESS.nextval
INTO :new.ID
FROM dual;
END;
/
ALTER TRIGGER "TR_S_CR_E_ADDRESS" ENABLE;

```

*Trigger TR\_S\_CR\_MAIN\_DATA:*

```

CREATE OR REPLACE TRIGGER "TR_S_CR_MAIN_DATA"
BEFORE INSERT ON CR_MAIN_DATA
FOR EACH ROW
BEGIN
SELECT S_CR_MAIN_DATA.nextval
INTO :new.ID
FROM dual;
END;
/
ALTER TRIGGER "TR_S_CR_MAIN_DATA" ENABLE;

```

Questi comandi devono essere digitati all'interno di un client SQL. Utilizzando, ad esempio, SQLPlus, che è il client che viene installato insieme al database server Oracle Standard Edition, è necessario prima connettersi al database fornendo il nome utente, la relativa password e il SID del database; se vogliamo eseguire i comandi sul database Standard Edition digitiamo da riga di comando:

```
sqlplus area@oracle_localhost
```

il sistema chiederà la password associata all'utente *area*; una volta immessa si è connessi al database ed è possibile eseguire i comandi SQL.

Allo stesso modo ci si può connettere con il database Express Edition, digitando da riga di comando:

```
sqlplus area@xe_192.168.0.2
```

E' possibile inoltre salvare tutti i comandi SQL in una *script* ed eseguirli contemporaneamente; ad esempio se chiamiamo *area.sql* la script composta dai comandi di creazione degli oggetti appena visti possiamo eseguirla, una volta connessi al database, digitando:

```
SQL>@area.sql
```

In questo modo vengono eseguiti automaticamente tutti i comandi dello script.

E' opportuno ricordare che per rendere permanenti le modifiche fatte al database Oracle è necessario digitare il comando `commit`;

A questo punto la situazione che ci troviamo davanti è quella di due database che nello schema *area* hanno gli stessi oggetti (*Fig.4.1*);

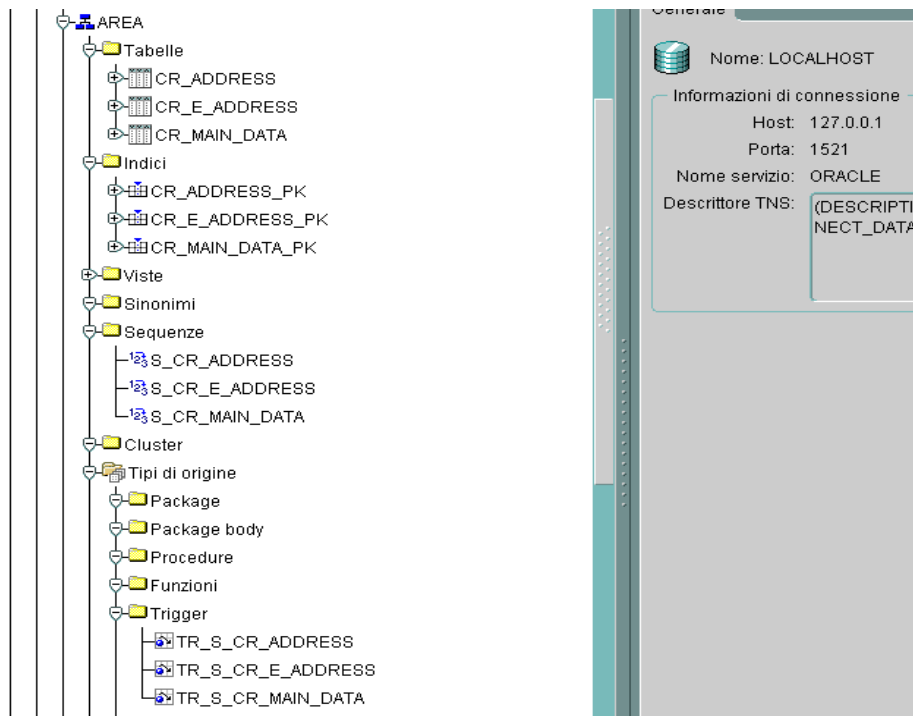


Fig.4.1

A questo punto, per poter completare questa fase, è opportuno “riempire” le tabelle di un numero di record significativo.

In entrambi i database sono stati inseriti 300 record per ognuna delle 3 tabelle utilizzando il comando SQL “insert into”. Per eseguire i test è opportuno identificare per ogni record il database sul quale è stato inserito originariamente, di modo che, quando si esegue la sincronizzazione tra database, si possa riconoscere facilmente a che database appartiene (nell’esperimento si è scelto di aggiungere ad ogni record memorizzato nelle tabelle dell’Express Edition la desinenza \_XE, nei campi il cui dominio era la stringa).

#### *Individuazione delle principali soluzioni di sincronizzazione tra database con tecnologia Oracle*

In questa fase sono state analizzate le possibili metodologie di sincronizzazione che la tecnologia Oracle mette a disposizione nei propri database.

Partendo dalla documentazione ufficiale di Oracle [Oracle10gMan] e da altri documenti reperiti in internet, si sono studiati i vari metodi di sincronizzazione tra database Oracle. Per ogni metodo individuato se ne sono analizzate le caratteristiche principali e si è valutato se tale metodo fosse applicabile nel modello pratico realizzato.

Occorre ricordare che l’obiettivo è anche quello di individuare, tra i metodi analizzati, quello che richiede la minore necessità di scrivere istruzioni nel database e/o programmi esterni, per compiere materialmente l’operazione di sincronizzazione nel minor tempo possibile e nel modo più “automatico” possibile.

Come già detto, il fatto di utilizzare lo Standard e l’Express Edition di Oracle 10g come specifici database server restringe molto il cerchio delle soluzioni applicabili perché tali database non posseggono tutte le funzionalità che Oracle implementa nell’Enterprise Edition (ma sono anche molto meno costosi di tale versione, o addirittura gratuiti come nel caso dell’Express Edition).

Si sono così individuati quattro metodi di sincronizzazione:

- *DB link*;
- *Oracle Stream*;
- *Oracle Grid Computing*;
- *Uso di SQL Developer*;

Tali soluzioni sono state analizzate approfonditamente e realizzate nella successiva fase.

#### *Implementazione delle soluzioni individuate*

Le soluzioni individuate e prese in considerazione nella fase precedente, sono state implementate in questa fase, seguendo il manuale ufficiale Oracle [manuali Oracle], e la documentazione reperibile in internet.

Per l'implementazione di ognuna delle soluzioni è stato opportuno eseguire delle operazioni che sono descritte approfonditamente nei capitoli successivi.

#### *Test operativo di ognuna delle soluzioni e annotazione dei risultati*

Questa è la fase operativa più "interessante". Ognuna delle quattro soluzioni implementate nella precedente fase è stata analizzata e testata nella pratica. Durante i test sono stati annotati i parametri più importanti e sono emersi, per ognuna delle soluzioni, gli aspetti positivi e negativi.

Tali aspetti sono descritti approfonditamente nei capitoli successivi.

Alla fine dell'esperimento, quindi, si è potuto fare un paragone tra i vari metodi di sincronizzazione individuando, per ogni punto di vista, quale fosse quello migliore. I risultati di tale analisi comparativa saranno presentati nel Capitolo 9.

# Capitolo 5

## Uso del DB Link

### 1. Caratteristiche del DB Link

Come dice la parola stessa, **Oracle Database Link** è un collegamento (*link*) tra due database Oracle [Oracle10gMan], [DBJW 2004], [Alapati 2005].

I DB Link sono veri e propri oggetti di un database Oracle, e permettono di accedere, dall'interno di un database locale (dove viene definito il DB Link), a dati "remoti", cioè memorizzati in un database remoto, collegato al locale.

Formalmente un DB Link si può definire come un puntatore che definisce un percorso di comunicazione unidirezionale da un Db Server Oracle ad un altro DB server.

Tale puntatore del link viene definito come elemento di una tabella del dizionario dei dati del DB di partenza, e punta ad una tabella del Db di destinazione.

Si dice che i DB Link sono "unidirezionali" perché essi rappresentano un canale di comunicazione non invertibile; quindi uno stesso DB Link permette di leggere i dati da un database server Oracle in un altro, ma non il contrario.

Per questo motivo per creare un canale bidirezionale tra due database occorre utilizzare due DB Link: se A e B sono i database da collegare bidirezionalmente, si deve costruire un Db Link da A a B nel database A, per la comunicazione A-B, e uno da B ad A, nel database B per la comunicazione B-A.

In questo modo sarà possibile sia accedere ai dati memorizzati nel database B direttamente da A, sia accedere ai dati memorizzati nel database A direttamente da B.

Gli utenti del database locale (su cui è definito il DB Link) possono accedere agli oggetti di un altro utente in un DB remoto vincolati al set di privilegi del proprietario dell'oggetto, ossia un utente locale può accedere ad un DB remoto senza essere un utente del DB remoto.

In base agli utenti che possono utilizzare i DB link si possono individuare tre tipi di link:

- *Private DB Link*
- *Public DB Link*
- *Global DB Link*

I *Private DB Link*, sono link creati su uno specifico schema, e solo l'utente proprietario del DB Link (cioè colui che lo ha creato) può utilizzare tale Db Link per accedere ai dati e agli oggetti memorizzati in corrispondenza del database remoto.

Nei *Public DB Link* tutti gli utenti del database (e, quindi, non solo chi lo ha creato) possono usare il DB Link per referenziare i dati e gli oggetti remoti.

In una rete di database Oracle, spesso, vengono creati i *Global DB Link*, ossia DB Link che possono essere utilizzati da tutti i database appartenenti alla rete.

La sintassi dei comandi SQL per creare un Database Link è la seguente:

```
CREATE [SHARED] [PUBLIC] DATABASE LINK nome_DBLink
[CONNECT TO [CURRENT_USER] [nome_user IDENTIFIED BY pwd]]
```

[AUTHENTICATED BY *nome\_user* IDENTIFIED BY *pwd*]  
 USING *connect\_string*.

Dove le opzioni (le “parole chiave” che stanno tra []) e i parametri ( le parole scritte in corsivo) stanno ad indicare:

- SHARED: opzione che rende un unico DBLink condiviso tra più applicazioni, e consente di limitare il numero di connessioni necessarie tra il database locale e quello remoto.
- PUBLIC: opzione che rende il DB Link pubblico, ossia utilizzabile da tutti gli utenti del database. Se l’opzione PUBLIC non è presente, viene creato un *Private DB Link*.
- *nome\_DBLink*: è il nome che viene dato al DB Link. Questo può essere un nome qualsiasi, meglio se significativo. Esso sarà usato in seguito alla creazione per utilizzare tale DB Link .
- CONNECT TO: opzione che permette di specificare con quale utente desidero connettermi al database remoto. In particolare si può far seguire CURRENT USER, che specifica che l’utente attuale è anche l’utente con il quale si vuole accedere al database remoto (naturalmente tale utente deve avere un account valido sul database remoto), oppure da *nome\_user* IDENTIFIED BY *pwd*, con il quale si specifica un utente (*nome\_user*) e la relativa password (*pwd*) per la connessione al database remoto (e si ci connette al database remoto con tale utente).
- AUTHENTICATED BY *nome\_user* IDENTIFIED BY *pwd*: questa opzione permette di specificare l’username (*nome\_user*) e la relativa password (*pwd*) per l’autenticazione dell’utente nell’istanza remota; rappresenta una misura di sicurezza, in quanto tale utente è utilizzato solamente per l’autenticazione nell’istanza remota, ma nessuna altra operazione gli è consentita.
- *connect\_string*: specifica il nome con cui viene identificato, nel file *tnsnames.ora* del server, il SID o il SERVICE NAME del db a cui connettersi.

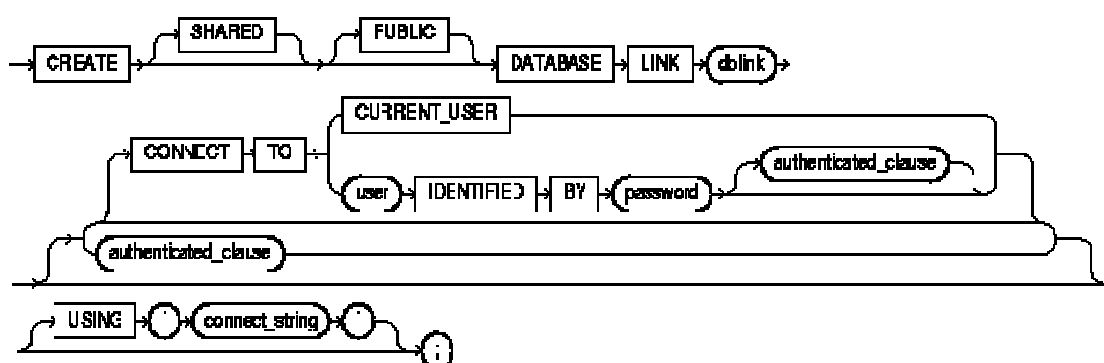


Fig. 5.1

Una volta creato, un DB Link, è utilizzato per fare riferimento ad oggetti remoti; la sintassi per referenziare ad esempio tabelle memorizzate in un database remoto è la seguente:

*nome\_user.table@nome\_dblink*

dove *nome\_user* è il nome dell'utente con cui ci si connette al database remoto, *table* è il nome di una tabella presente nello schema di *nome\_user* a cui si fa riferimento, *nome\_dblink* è il nome del DB Link datogli al momento della creazione.

Nel prossimo paragrafo vediamo come il meccanismo dei DB Link può essere utilizzato per la risoluzione del problema del “commesso viaggiatore”.

## 2. Applicazione al problema

La prima soluzione presa in considerazione per la sincronizzazione tra Database Oracle è stata quella degli Oracle DB Link.

In questo caso, per sincronizzazione, non si intende la replica da un database all'altro dei dati di interesse, ma la semplice disponibilità, questo, perché, come abbiamo visto, i DB Link non provvedono da soli alla duplicazione dei dati.

Come detto nel capitolo precedente il canale di comunicazione di cui abbiamo bisogno nel nostro modello pratico deve essere bidirezionale; occorrono quindi due DB Link per realizzarlo, uno definito nello Standard Edition e l'altro nell'Express Edition.

Nello Standard Edition si crea un DB Link, che permette di accedere ai dati memorizzati nell'Express Edition con il seguente comando:

```
CREATE DATABASE LINK DBL_ST_XE
CONNECT TO AREA IDENTIFIED BY password_user_area
USING 192.168.0.2:1521/XE
```

Questi comandi creano un Db Link di nome DBL\_ST\_XE che si connette al database Express, il cui SERVICE NAME è rappresentato dalla terna *host : porta / SID*, ed è appunto 192.168.0.2:1521/XE, utilizzando solamente lo schema dell'utente AREA.

Allo stesso modo nell'Express Edition si può creare il seguente DB Link:

```
CREATE DATABASE LINK DBL_XE_ST
CONNECT TO AREA IDENTIFIED BY password_user_area
USING 192.168.0.1:1521/ORACLE
```

Dove, rispetto al precedente, cambiano solamente il nome del DB Link, che è DBL\_XE\_ST e il SERVICE NAME che è 192.168.0.1:1521/ORACLE cioè la terna *host : porta / SID* relativa allo Standar Edition.

Una volta creati questi DB Link è possibile eseguire query in un database su tabelle che non sono memorizzate in tale database, ma nell'altro.

Vediamone un esempio:

Da riga di comando mi connetto, utilizzando SQL\*Plus, al database Standard Edition con utente AREA digitando:

```
sqlplus area@localhost
```

Inserisco la password relativa all'utente AREA .

Una volta connesso posso eseguire una qualsiasi query specificando la tabella del database remoto e il nome del DB Link.

Ad esempio posso visualizzare tutte le città inserite nella tabella CR\_ADDRESS memorizzata nell'Express in questo modo:

```
SELECT city FROM area.CR_ADDRESS@DBL_ST_XE
```

Allo stesso modo posso visualizzare tutte le città inserite nella tabella CR\_ADDRESS e memorizzata nello Standard, direttamente dall'Express Edition;

Mi connetto con SQL\*Plus all'Express Edition con :

```
sqlplus area@xe_192.168.0.2
```

Poi eseguo la query:

```
SELECT city FROM area.CR_ADDRESS@ DBL_XE_ST
```

In questo modo visualizzo tutte le città inserite in CR\_ADDRESS.

Oltre che con istruzioni SQL DML (*Data Manipulation Language*) i DB Link possono essere utilizzati anche con istruzioni SQL DDL (*Data Definition Language*). Questo sta a significare che oltre a query su tabelle remote, si possono anche eseguire operazioni di creazione, eliminazione ed inserimento di tabelle e dati remoti.

Sarà quindi possibile inserire record in una tabella (INSERT TO) memorizzata nell'Express direttamente dallo Standard e viceversa.

### 3. Esiti dell'azione

Come visto, quindi, i DB Link permettono di condividere dati tra più database server, sia Oracle, ma anche di altri Vendor.

Questa caratteristica li rende facilmente utilizzabili nell'ambito dei database distribuiti.

I Db Link rappresentano un metodo veloce e facilmente implementabile per sincronizzare due database.

Nell'esperimento svolto durante il periodo di tirocinio, i DB Link sono stati utilizzati per creare un canale di comunicazione tra i due database Oracle. Tale canale ha permesso di condividere i dati tra i due database, in entrambi i sensi.

Dai test fatti su questa soluzione emerge che, nelle situazioni in cui è importante avere a disposizione dati remoti, i DB Link rappresentano uno strumento molto efficace e conveniente, in quanto evitano la replica dei dati remoti nel database locale, ma ne garantiscono la disponibilità.

I Db Link sono molto efficaci anche quando ci troviamo di fronte ad una rete di database; In questa situazione è possibile, connettendosi ad un solo database server, eseguire operazioni su tutti gli altri database server della rete.

Il punto di forza che i DB Link hanno è sicuramente quello di essere facilmente realizzabili attraverso pochi comandi SQL.

Tuttavia, in caso in cui si debbano replicare i dati tra i database, non è possibile l'utilizzo dei DB Link.

In definitiva quindi i DB Link sono un semplice, ma potente, metodo per creare collegamenti tra database, che permettono la condivisione di oggetti, ma non direttamente la replica, che, al bisogno, dovrà comunque essere realizzata da apposite istruzioni inserite nel database e/o in programmi esterni.

# Capitolo 6

## Uso degli Oracle Stream

### 1. Caratteristiche degli Oracle Stream

Gli Oracle Streams rappresentano un metodo molto efficace ed efficiente, che consente la sincronizzazione tra DB, incorporato nei database Oracle a partire dalla versione 10g [Oracle10gMan], [DBJW 2004] e [Alapati 2005].

Il concetto di sincronizzazione espresso in questo ambito è quello di replica dei dati di interesse da un database ad un altro.

Gli Oracle Streams consentono di condividere modifiche al database e altre informazioni in un flusso (*Stream*), per la propagazione di eventi all'interno del database o da un database a un altro.

Le informazioni specificate, cioè quelle che devono essere replicate, vengono instradate alle destinazioni indicate.

Questo metodo offre quindi funzionalità e flessibilità maggiori rispetto alle soluzioni tradizionali per il recupero e la gestione delle informazioni e per la condivisione dei dati con altri database e applicazioni.

Il funzionamento degli Oracle Stream è molto semplice:

Catturano le modifiche di un database, le memorizzano in un'area temporanea per poterle applicare su più istanze, le propagano ad uno o più database di destinazione e le applicano su tali database.

Tale propagazione può avvenire:

- All'interno di un database Oracle;
- Tra due database Oracle;
- Tra più database Oracle;
- Tra un database Oracle ed un database non Oracle;

I cambiamenti del database, ossia le modifiche DML e DDL apportate agli oggetti di database, sono catturati da un processo di background Oracle e sono registrati nei redo log files.

Il processo di “*Streams capture*” estrae questi cambiamenti dai redo log e li formatta in un “*logical change record*” (**LCR**).

Ogni LCR viene storicizzato in una coda (*stanged*) dal processo “*Streams Messaging*”, denominato anche “*Oracle Streams Advanced Queuing*”.

Successivamente, Oracle Stream propaga gli LCR da una coda (la *producer queue*) all'altra (la *consumer queue*) ed applica gli LCR dalla “*consumer queue*” al database di destinazione.

Il database di destinazione quindi risulta sincronizzato con quello di partenza grazie ai flussi di modifiche che gli vengono inviate e applicate dall'Oracle Stream, man mano che queste vengono eseguite sul database di partenza.

A causa della sua semplicità e della sua versatilità, nella realtà, questo metodo viene solitamente utilizzato per eseguire compiti come:

- Data replication;
- Estrazione e caricamento di Data Warehouse;
- Notifica degli eventi;

- Message queuing;
- Migrazione di piattaforma del database;
- Upgrade di database e applicazioni;

Obiettivo di questo capitolo è illustrare ed analizzare il metodo dell'Oracle Stream impiegato per il "Data replication".

Con questo termine si intende la replica delle modifiche eseguite in un database sorgente in uno di destinazione.

I passi generali che si devono seguire per costruire un ambiente di "Stream replication" sono:

1. Impostare il db in ARCHIVELOG mode, in quanto per assicurare che l'informazione contenuta nei log files sia disponibile per il processo "Stream Capture", occorre che il processo di background ARCH copi i redo log files prima che questi siano sovrascritti, per evitare perdite di modifiche.
2. Creare lo "Stream administrator", ossia un utente che gestisce tutto l'ambiente Stream (*Stream Environment*). Questo utente deve possedere alcuni specifici privilegi e deve creare alcune tabelle per storicizzare informazioni. Per ogni database partecipante allo Stream, è necessario creare un utente e designare questo come Stream administrator.
3. Impostare i parametri di inizializzazione; è infatti indispensabile, in ogni database partecipante, impostare i parametri di inizializzazione nel modo corretto; Ad esempio si deve impostare il parametro GLOBAL\_NAMES a TRUE in entrambi i db perché questo influenza la creazione dei DB Link. Gli altri parametri da impostare in questo passo sono riassunti nella seguente tabella:

Parameter	Source (TEST10G1)	Destination (TEST10G2)
GLOBAL_NAMES	TRUE	TRUE
COMPATIBLE	10.1.0	10.1.0
JOB_QUEUE_PROCESSES	2 (or higher)	N/A
STREAMS_POOL_SIZE	200MB (minimum recommended)	200MB (minimum recommended)

4. Creare un database link, per il passaggio delle informazioni dal database sorgente a quello destinazione;
5. Costruire la coda sorgente e quella destinazione, per consentire che i dati si spostino dal database sorgente a quello destinazione attraverso le code.
6. Costruire il logging supplementare nel database sorgente, in quanto prima di iniziare a catturare i cambiamenti nel database sorgente, è necessario aggiungere un logging supplementare sulle tabelle che verranno modificate. Il logging supplementare mette informazioni aggiuntive nei redo logs files utili durante il processo di "apply".
7. Configurare il processo di "cattura" nel database sorgente, per indicare quali modifiche DML e DDL siano catturate.
8. Configurare il processo di propagazione, che consiste nel collegare le due code create.

9. Creare la tabella di destinazione, perché prima di iniziare a replicare le modifiche da una tabella sorgente a una di destinazione, la tabella deve esistere nel database di destinazione.
10. Fornire i privilegi allo Stream Administrator di destinazione, in modo che egli possa applicare i cambiamenti catturati dal database sorgente.
11. Istanziamento del *system change number* (SCN), ossia un numero associato alla tabella del database sorgente che si vuole replicare, che permette di ignorare i cambiamenti catturati prima di istanziare lo SCN.
12. Configurazione dei processi di “apply” al database di destinazione, ossia occorre associare al processo “apply” la coda di destinazione da cui prende le modifiche.
13. Avvio dei processi di cattura e di apply, cioè i processi configurati precedentemente che permettono di catturare i cambiamenti nella tabella sorgente e replicarli nella tabella di destinazione.

Questa serie di passi permette di creare uno Stream per la replica delle modifiche su una tabella da un database sorgente ad uno destinazione. Ma come vediamo nel prossimo paragrafo, questo non è l'unico metodo; la web console incorporata nei database server Oracle permette di facilitare ulteriormente la creazione degli streams tra database.

## 2. Applicazione al problema

Il secondo metodo analizzato ed implementato nell'esperimento svolto sono gli **Oracle Stream**.

Come detto, questi permettono di catturare le modifiche che avvengono in un database e di replicarle in un altro database.

Per questo motivo gli Oracle Stream si adattano bene a risolvere il problema del “commesso viaggiatore”.

Nell'ambito dell'esperimento svolto Oracle Stream è stato utilizzato come metodo per il “data replication”, ossia come metodo di replica dei dati.

Nel paragrafo precedente sono stati definiti i passi generali necessari per creare uno Stream tra due database.

Nella pratica, però, ci sono metodi più semplici ed immediati per la creazione degli Streams; Il più immediato è sicuramente quello che viene messo a disposizione dalla **Oracle DB Console**, la console web dei database Oracle.

Come detto precedentemente, ogni database Oracle incorpora una console di amministrazione web cui è possibile accedere da qualsiasi browser.

Attraverso tale console è possibile impostare e gestire in maniera grafica, e alle volte intuitiva, i parametri dell'istanza e personalizzare il database.

Attraverso l'Oracle DB Console è possibile, seguendo una serie di passi, impostare uno streams tra due database.

Di seguito viene riportata la procedura eseguita per creare uno stream tra il database Standard Edition e l'Express Edition utilizzando la DB Console dello Standard.

Come prima cosa, dopo aver avviato entrambi i database, è opportuno accedere alla DB Console dello Standard Edition; per fare questo è opportuno digitare su un qualsiasi browser l'indirizzo:

*http://nomehost:1158/em*

Questo è l'url per accedere alla DB Console, dove al posto di *nomehost* deve essere messo l'indirizzo ip dell'host su cui il database Standard Edition gira.

Nel caso si acceda da un browser che risiede sulla stessa macchina del database si può mettere *localhost*.

La pagina iniziale che ci si trova di fronte appena ci si collega alla console web è quella del login; in essa vanno inseriti un nome utente e la relativa password il cui account sia valido sul database.

Solitamente la DB Console è utilizzata per amministrazione del database da parte di figure professionali specifiche come i DBA (DataBase Administrator). Per questo motivo, la maggior parte delle operazioni che si possono eseguire dalla DB Console necessitano dei privilegi di amministratore. La creazione di stream è una di queste. E' quindi opportuno accedere alla console con l'username SYS e la relativa password decisa al momento dell'installazione del database.

Se tutto è inserito correttamente si accede alla pagina iniziale, in cui sono riportati, oltre il menù di navigazione tra le varie schede, anche alcuni dati sulle prestazioni del database. La figura 6.1 rappresenta la pagina iniziale della DB console del database Oracle Standard Edition 10g.

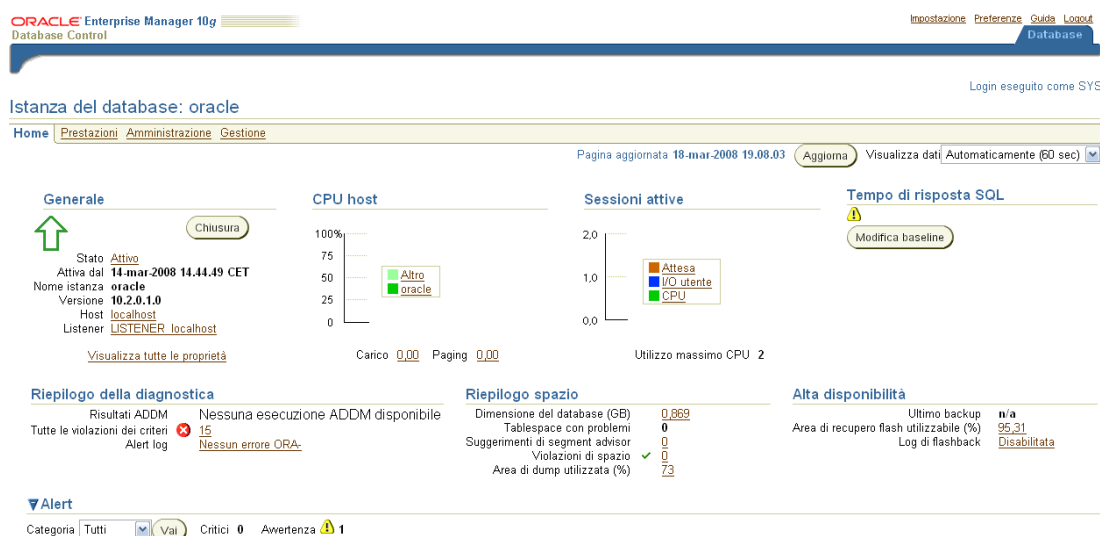


Fig. 6.1

Attraverso il menù di navigazione si accede alla scheda denominata **Gestione** (l'ultima da sinistra verso destra).

In tale scheda vengono visualizzati tutti i collegamenti che forniscono le funzioni di controllo del flusso di dati tra database Oracle e l'esterno.

Nella sezione centrale denominata **Spostamento dei dati** trova spazio la sotto-sezione **Flussi**.

In essa sono presenti due collegamenti: Impostazione e Gestione.

Impostazione permette di creare ed impostare nuovi Streams mentre Gestione permette di gestire quelli esistenti.

Selezionando **Impostazione** si accede alla pagina illustrata nella figura 6.2;

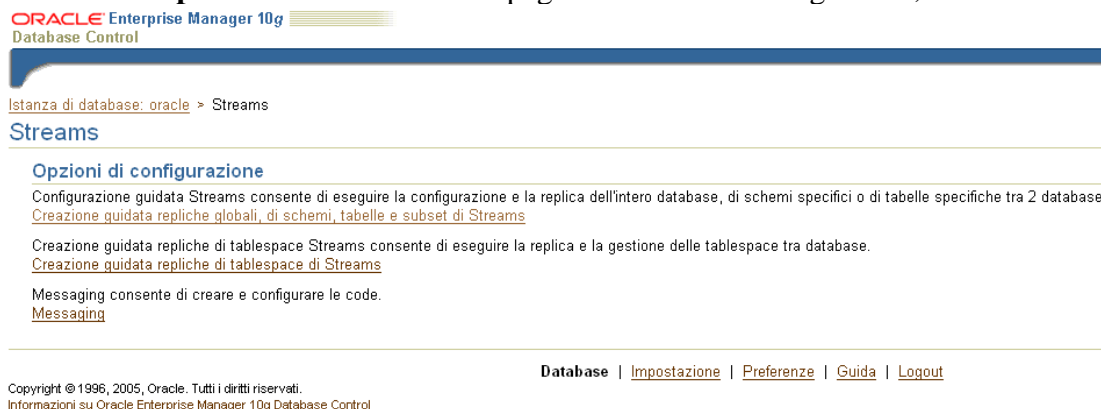


Fig. 6.2

In questa pagina è possibile configurare tre tipi di Flussi:

- *Creazione guidata repliche globali, di schemi, tabelle e subset di Streams*, che permette di eseguire la configurazione e la replica dell'intero database, di schemi o tabelle specifiche tra due database.
- *Creazione guidata repliche di tablespace di Streams*, che consente di eseguire la replica e la gestione delle tablespace tra database.
- *Messaging*, che consente di creare e configurare le code utilizzate dagli Oracle Stream.

Prima di selezionare una delle precedenti voci, occorre che il database sia avviato in ARCHIVELOG mode, perché, come abbiamo già detto, è necessario che i redo log files siano archiviati.

Per default un database Oracle funziona in modalità NOARCHIVELOG, ma è possibile cambiare lo stato in qualunque momento in ARCHIVELOG eseguendo i seguenti comandi SQL:

```
SHUTDOWN;  
STARTUP MOUNT EXCLUSIVE;  
ALTER DATABASE ARCHIVELOG;  
ALTER DATABASE OPEN;
```

Una volta cambiato lo stato del database è possibile selezionare una delle precedenti creazioni guidate.

Per realizzare quello che ci siamo proposti, selezioniamo la prima voce; a questo punto parte la creazione guidata dello stream che consta di 5 step:

1. Il primo passo consiste nell'indicazione di un utente o nella creazione di un nuovo utente nel database di origine, che avrà la funzione di Streams Administrator. Sono richieste, quindi, o il nome di un utente esistente e la relativa password o il nome di nuovo utente e la relativa password. (Fig. 6.3). Supponiamo di creare un nuovo utente selezionando "Crea amministratore Streams" e inseriamo come username `stradmin` e come password `stradmin`.



Fig.6.3

- Il secondo passo è la consiste nella configurazione del DB di destinazione, in particolare bisogna immettere: *Nome host*, *Porta*, *SID o Nome servizio*, *Amministratore streams* e *Password*. Anche nel database di destinazione quindi bisogna indicare un utente che copre il ruolo di Streams Administrator. Anche qui è possibile indicare come amministratore un utente esistente oppure crearne uno nuovo appositamente. (Fig.6.4)

Nel nostro esempio riempiamo i campi così:

Nome host: 192.168.0.2

Porta: 1521

SID: XE

Amministratore Streams: stradmin

Password: stradmin



Fig. 6.4

3. Il terzo passo permette di configurare il metodo di replica degli oggetti; come prima cosa viene chiesto di scegliere il metodo di replica più opportuno, ossia un metodo per selezionare gli oggetti da replicare; E' possibile scegliere uno di questi metodi:
  - *Regola globale*, che consente di specificare gli schemi e le tabelle da escludere;
  - *Regola dello schema*, che consente di specificare gli schemi da includere e le tabelle da escludere;
  - *Regola della tabella*, che consente di specificare le tabelle da includere o una clausola WHERE per includere un subset di una tabella.

E' nel passo successivo poi che si andrà a specificare schemi e/o tabelle e/o clausole WHERE.

Nel nostro esempio selezioniamo *Regola dello schema* per replicare tutte le modifiche apportate ad uno schema.

Vengono inoltre chiesti i nomi da attribuire ai processi di recupero (*Capture*), di propagazione (*Propagation*) e di applicazione (*Apply*).

Come detto nel precedente paragrafo il processo "*Capture*" è il processo che cattura le modifiche nel database di partenza, il processo "*Propagation*" propaga le modifiche tra le varie code fino al database di destinazione e il processo "*Apply*" applica le modifiche al database di destinazione.

Di default i nomi inseriti sono rispettivamente STREAMS\_CAPTURE, STREAMS\_PROPAGATION e STREAMS\_APPLY.

Ma si possono creare nuovi processi con nuovi nomi cliccando sui pulsanti a fianco di ogni form.

Più in basso viene chiesto di specificare una directory nel database di origine per la memorizzazione del file di *dump* di esportazione di *DataPump* e di specificare anche una directory nel database di destinazione per la memorizzazione dei file di log e del file *dump* di importazione di *DataPump* generati.

Selezionando **Crea oggetto directory** è possibile creare "a mano" la cartella ed indicarne il percorso che ha sul file system; le cartelle che si devono creare sono due; una sul database di origine e una sul database di destinazione; nella creazione dello Streams tra Standard e Express si creano la cartella `streams` nello Standard Edition con path:

```
C:\oracle\product\10.2.0\admin\oracle\streams
```

e la cartella `streams` nell'Express Edition con path:

```
C:\oraclexe\app\oracle\admin\XE\streams
```

Si deve inoltre indicare nelle due form il nome da attribuire all'oggetto directory dei due database; in entrambi possiamo scrivere `streamdir`.

Per rendere "manipolabili" tali cartelle occorre che i database dispongano dei privilegi di lettura e scrittura su queste.

Per assegnare tali privilegi occorre, prima di utilizzare lo Stream, eseguire il comando SQL GRANT che permette di attribuire privilegi ad oggetti del database; Bisogna quindi digitare, una volta connessi ad ognuno dei due database, il seguente comando:

```
GRANT read , write ON SYS.streamdir TO PUBLIC;
```

Tale comando deve essere eseguito su entrambi i database, e ha il significato di rendere pubblici (cioè utilizzabili da tutti gli utenti) i privilegi di scrittura e lettura sull'oggetto del database *streamdir* (cioè la directory streams creata) appartenente allo schema *SYS* (cioè l'utente con il quale ci si è connessi alla DB console). E' possibile inoltre definire altre opzioni di replica nella parte bassa dell'interfaccia.(Fig. 6.5).

The screenshot shows the 'Configura Streams: Configura replica' window. At the top, there are navigation tabs: 'Database di origine', 'Database di destinazione', 'Configura replica' (active), 'Selezione oggetti', and 'Revisione'. Below the title bar, there are buttons for 'Annulla', 'Indietro', 'Passo 3 di 5', and 'Avanti'. The main content is divided into sections: 'Replca' (with radio buttons for 'Regola globale', 'Regola dello schema' (selected), and 'Regola della tabella'), 'Processi' (with input fields for 'Nome recupero' (STREAMS\_CAPTURE), 'Nome propagazione' (STREAMS\_PROPAGATION), and 'Nome applicazione' (STREAMS\_APPLY)), 'Oggetti directory' (with input fields for 'Database di origine' and 'Database di destinazione', each with a 'Crea oggetto directory' button), and 'Opzioni' (with a checked checkbox for 'Recupera, propaga e applica modifiche DML (Data Manipulation Language)').

Fig. 6.5

4. Il quarto passo prevede di selezionare il nome utente e la password relativa all'utente del quale se ne vuole replicare le modifiche allo schema. Nel nostro caso si seleziona lo schema AREA.
5. Il quinto ed ultimo passo è un riepilogo dei parametri inseriti in questa creazione guidata. Eseguendo le operazioni pratiche descritte fino ad ora si ha il seguente riepilogo:

<p>Database di origine: oracle Database di destinazione: xe</p> <p>1. Operazioni di impostazione</p> <ul style="list-style-type: none"> <li>- Creare un amministratore Streams "stradmin" nel database di origine "ORACLE.REGRESS.RDBMS.DEV.US.ORACLE.COM".</li> <li>- Creare un amministratore Streams "stradmin" nel database di destinazione "xe".</li> <li>- Configurare la replica "Schema".</li> <li>-- Includere i seguenti schemi "AREA".</li> <li>- Recuperare, propagare e applicare modifiche DML (Data Manipulation Language)</li> <li>- Copiare i dati esistenti nel database di destinazione come parte della configurazione</li> </ul> <p>2. Operazioni di esportazione/importazione</p> <ul style="list-style-type: none"> <li>- Esportare tutti gli oggetti selezionati dal database di origine.</li> <li>- Importarli nel database di destinazione.</li> </ul> <p>3. Operazioni di avvio</p> <ul style="list-style-type: none"> <li>- Avviare prima il processo di applicazione nel database di destinazione.</li> <li>- Avviare il processo Recupera nel database di origine.</li> </ul>
---

Come si vede la procedura è suddivisa in tre sezioni:

- La prima riguarda l'impostazione dello Stream;
- La seconda riguarda l'importazione e l'esportazione dei dati da replicare ;
- La terza riguarda l'avvio del meccanismo Oracle Streams;

Quanto visto fino ad ora permette di creare uno Stream che replica gli oggetti nello schema AREA dello Standard Edition nello schema AREA dell'Express Edition.

Il metodo utilizzato per la creazione di tale Stream, attraverso la DB Console, non è l'unico possibile; lo Stream appena creato è implementabile anche attraverso l'esecuzione da riga di comando di script SQL; In particolare tale Stream è il risultato dell'esecuzione delle seguenti tre script SQL (tali script rappresentano le tre sezioni in cui era suddivisa la creazione guidata della DB console):

### Script SQL di setup dello stream:

```
set echo on;
ACCEPT dba_pwd_src PROMPT 'Enter Password of user "SYSTEM" to create
Streams Admin at Source : ' HIDE
ACCEPT strm_pwd_src PROMPT 'Enter Password of Streams Admin
"stradmin" to be created at Source : ' HIDE
ACCEPT dba_pwd_dest PROMPT 'Enter Password of user "system" to
create Streams Admin at Destination : ' HIDE
ACCEPT strm_pwd_dest PROMPT 'Enter Password of Streams Admin
"stradmin" to be created at Destination : ' HIDE
connect "SYSTEM"/&dba_pwd_src;
create user "STRADMIN" identified by &strm_pwd_src;
grant DBA, IMP_FULL_DATABASE, EXP_FULL_DATABASE to "STRADMIN";
BEGIN
  DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE(
    grantee => '"STRADMIN"',
    grant_privileges => true);
END;
/
COMMIT;
connect "STRADMIN"/&strm_pwd_src;
CREATE DATABASE LINK XE connect to "STRADMIN" identified by
&strm_pwd_dest using
'(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=192.168.0.2)
(PORT=1521))))(CONNECT_DATA=(SID=x)(server=DEDICATED))';
BEGIN
  DBMS_STREAMS_ADM.SET_UP_QUEUE(
    queue_table => '"STREAMS_CAPTURE_QT"',
    queue_name => '"STREAMS_CAPTURE_Q"',
    queue_user => '"STRADMIN"');
END;
/
COMMIT;
BEGIN
  DBMS_STREAMS_ADM.ADD_SCHEMA_RULES(
    schema_name => '"AREA"',
    streams_type => 'capture',
    streams_name => '"STREAMS_CAPTURE"',
    queue_name => '"STRADMIN"."STREAMS_CAPTURE_Q"',
    include_dml => true,
    include_ddl => false,
```

```

        include_tagged_lcr => false,
        inclusion_rule      => true);
END;
/
BEGIN
  DBMS_STREAMS_ADM.ADD_SCHEMA_PROPAGATION_RULES(
    schema_name           => ' "AREA" ',
    streams_name          => ' "STREAMS_PROPAGATION" ',
    source_queue_name     => ' "STRADMIN"."STREAMS_CAPTURE_Q" ',
    destination_queue_name => ' "STRADMIN"."STREAMS_APPLY_Q"@XE ',
    include_dml           => true,
    include_ddl           => false,
    source_database       =>
'ORACLE.REGRESS.RDBMS.DEV.US.ORACLE.COM',
    inclusion_rule        => true );
END;
/
COMMIT;
connect
"SYSTEM"/&dba_pwd_dest@"(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL
=TCP)(HOST=192.168.0.2)(PORT=1521)))(CONNECT_DATA=(SID=XE)(SERVER=DED
ICATED)))";
create user "STRADMIN" identified by &strm_pwd_dest;
grant DBA, IMP_FULL_DATABASE, EXP_FULL_DATABASE to "STRADMIN";

BEGIN
  DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE(
    grantee => ' "STRADMIN" ',
    grant_privileges => true);
END;
/
COMMIT;
connect
"STRADMIN"/&strm_pwd_dest@"(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTO
COL=TCP)(HOST=192.168.0.2)(PORT=1521)))(CONNECT_DATA=(SID=XE)(SERVER=
DEDICATED)))";

CREATE DATABASE LINK ORACLE.REGRESS.RDBMS.DEV.US.ORACLE.COM connect
to "STRADMIN" identified by &strm_pwd_src using
'(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(P
ORT=1521)))(CONNECT_DATA=(SID=oracle)(server=DEDICATED)))';
BEGIN
  DBMS_STREAMS_ADM.SET_UP_QUEUE(
    queue_table => ' "STREAMS_APPLY_QT" ',
    queue_name  => ' "STREAMS_APPLY_Q" ',
    queue_user  => ' "STRADMIN" ');
END;
/
BEGIN
  DBMS_STREAMS_ADM.ADD_SCHEMA_RULES(
    schema_name           => ' "AREA" ',
    streams_type          => ' apply ',
    streams_name          => ' "STREAMS_APPLY" ',
    queue_name            => ' "STRADMIN"."STREAMS_APPLY_Q" ',
    include_dml           => true,
    include_ddl           => false,
    include_tagged_lcr   => false,
    inclusion_rule        => true);
END;
\

```

## Script SQL di import/export dei dati dello stream:

```
ACCEPT strm_pwd_src PROMPT 'Enter Password of Streams Admin
"stradmin" at Source : ' HIDE
ACCEPT strm_pwd_dest PROMPT 'Enter Password of Streams Admin
"stradmin" at Destination : ' HIDE
connect
"STRADMIN"/&strm_pwd_dest@"(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTO
COL=TCP)(HOST=192.168.0.2)(PORT=1521)))(CONNECT_DATA=(SID=XE)(SERVER=
DEDICATED)))";
set serverout on;
DECLARE
  handle1 number;
  ind number;
  percent_done number;
  job_state VARCHAR2(30);
  le ku$_LogEntry;
  js ku$_JobStatus;
  jd ku$_JobDesc;
  sts ku$_Status;
BEGIN
  handle1 := DBMS_DATAPUMP.OPEN('IMPORT','SCHEMA',
'ORACLE.REGRESS.RDBMS.DEV.US.ORACLE.COM');
  DBMS_DATAPUMP.ADD_FILE(handle1, 'StreamImport_1197913640671.log',
'streamdir', '', DBMS_DATAPUMP.KU$_FILE_TYPE_LOG_FILE);
  DBMS_DATAPUMP.SET_PARAMETER(handle1, 'FLASHBACK_SCN', 1073613);
  DBMS_DATAPUMP.METADATA_FILTER(handle1, 'SCHEMA_EXPR', 'IN
(''AREA'')');
  DBMS_DATAPUMP.SET_PARAMETER(handle1, 'INCLUDE_METADATA', 1);
  DBMS_DATAPUMP.START_JOB(handle1);
  percent_done :=0;
  job_state := 'UNDEFINED';
  while (job_state != 'COMPLETED') and (job_state != 'STOPPED') loop
    dbms_datapump.get_status(handle1,
dbms_datapump.ku$_status_job_error +
dbms_datapump.ku$_status_job_status + dbms_datapump.ku$_status_wip,-
1,job_state,sts);
    js := sts.job_status;
    if js.percent_done != percent_done
    then
      dbms_output.put_line('*** Job percent done = ' ||
to_char(js.percent_done));
      percent_done := js.percent_done;
    end if;
    if(bitand(sts.mask, dbms_datapump.ku$_status_wip) != 0)
    then
      le := sts.wip;
    else
      if(bitand(sts.mask,dbms_datapump.ku$_status_job_error) != 0)
      then
        le := sts.error;
      else
        le := null;
      end if;
    end if;
    if le is not null
    then
      ind := le.FIRST;
      while ind is not null loop
```

```

        dbms_output.put_line(le(ind).LogText);
        ind := le.NEXT(ind);
    end loop;
end if;
end loop;
dbms_output.put_line('Job has completed');
dbms_output.put_line('Final job state = ' || job_state);
dbms_datapump.detach(handle1);
END;
/

```

### Script SQL di startup dello stream:

```

ACCEPT strm_pwd_src PROMPT 'Enter Password of Streams Admin
"stradmin" at Source : ' HIDE
ACCEPT strm_pwd_dest PROMPT 'Enter Password of Streams Admin
"stradmin" at Destination : ' HIDE
connect
"STRADMIN"/&strm_pwd_dest@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTO
COL=TCP)(HOST=192.168.0.2)(PORT=1521)))(CONNECT_DATA=(SID=XE)(SERVER=
DEDICATED)))";
set serverout on;
BEGIN
DBMS_APPLY_ADM.SET_SCHEMA_INSTANTIATION_SCN(
    source_schema_name    => '"AREA"',
    source_database_name  => 'ORACLE.REGRESS.RDBMS.DEV.US.ORACLE.COM',
    instantiation_scn     => 1073617,
    recursive              => true);
END;
/
DECLARE
    v_started number;
BEGIN
SELECT DECODE(status, 'ENABLED', 1, 0) INTO v_started
FROM DBA_APPLY where apply_name = 'STREAMS_APPLY';
if (v_started = 0) then
    DBMS_APPLY_ADM.START_APPLY(apply_name => '"STREAMS_APPLY"');
end if;
END;
/
connect "STRADMIN"/&strm_pwd_src;

set serverout on;
DECLARE
    v_started number;
BEGIN
SELECT DECODE(status, 'ENABLED', 1, 0) INTO v_started
FROM DBA_CAPTURE where CAPTURE_NAME = 'STREAMS_CAPTURE';
if (v_started = 0) then
    DBMS_CAPTURE_ADM.START_CAPTURE(capture_name =>
'"STREAMS_CAPTURE"');
end if;
END;
/
BEGIN
DBMS_OUTPUT.PUT_LINE('*** Progress Message ==> Started the capture
process STREAMS_CAPTURE at source database oracle and the apply
process STREAMS_APPLY at the destination database successfully.
***');
END;
/

```

### 3. Esiti dell'azione: limiti di applicabilità

Come detto, gli Oracle Stream rappresentano un metodo estremamente flessibile e funzionale per la sincronizzazione tra database.

Le loro caratteristiche e la loro possibilità di personalizzazione, permettono di poter utilizzare gli Stream in varie situazioni pratiche.

Oracle Stream può catturare, propagare ed applicare i cambiamenti in un database, automaticamente, includendo sia modifiche DML che DDL. Applicazioni che richiedono replicazione, data warehouses, database migrations, e database upgrade possono beneficiare di Oracle Stream.

Se analizziamo come vengono creati gli Streams e come questi funzionano ci accorgiamo che questi utilizzano due funzionalità principali incorporate nei database Oracle: la *Datapump* e i *DB Link*.

La *Datapump* è un utility, molto simile all'import/export, che è stata introdotta nei database Oracle 10g. Essa permette di importare dati nel database ed esportarli utilizzando un formato binario particolare. Attraverso l'impostazione di alcuni parametri la *Datapump* è molto più personalizzabile rispetto all'import/export dei dati. Nell'Oracle Stream la *Datapump* è utilizzata per spostare i dati tra le varie code fino al database di destinazione.

I *DB Link*, di cui abbiamo ampiamente parlato nel precedente capitolo, vengono utilizzati per collegare i due database.

In altre parole, quindi, Oracle Stream non è altro che una *Datapump* eseguita su di un *DB Link*.

Come detto precedentemente, il meccanismo dei *DB Link* permetteva la disponibilità dei dati remoti, ma non la replica. L'introduzione degli Oracle Stream ha permesso di superare tale limite, consentendo cioè, di replicare sul database anche i dati remoti.

Per quanto riguarda il nostro problema, gli Oracle Stream rappresentano proprio ciò di cui avevamo bisogno.

Utilizzando questi, infatti, è possibile tenere sincronizzati i due schemi del database master (il database centrale) e del database slave (il database del commesso viaggiatore): si può fare in modo quindi che tutte le modifiche DML e DDL applicate al database slave siano replicate sul database master, ma anche il contrario, costruendo uno stream su entrambi i database.

Tuttavia, questo, non è possibile nel nostro modello pratico costruito per rappresentare la situazione del commesso viaggiatore.

Questo perché il database slave, l'Oracle Express Edition, non ha incorporata la funzionalità dell'Oracle Stream, e quindi, esso, può ricoprire solamente il ruolo di database di destinazione in uno Stream. Questo è dovuto al fatto che l'Express Edition è un database gratuito, che è utilizzato soprattutto in ambienti locali, non distribuiti, quindi non presenta molte funzionalità per ambienti distribuiti, come ad esempio la *Datapump*.

Tale limite di applicabilità è superabile, se, al posto del database Express Edition, si fosse utilizzato un altro database Standard Edition, o, meglio ancora, se si fossero utilizzati due database Enterprise Edition.

In tale situazione risultava immediato la creazione di uno Stream per la sincronizzazione tra il database master e lo slave, e uno Stream per la sincronizzazione tra il database slave e il master.

In definitiva, si può concludere dicendo che il meccanismo degli Stream è molto efficace in situazioni di database distribuiti e in situazioni di Data Replication.

# Capitolo 7

## Uso di Oracle Grid

### 1. Caratteristiche di Oracle Grid

Come detto più volte in questo documento, la predisposizione al **Grid Computing**, è la principale innovazione incorporata nei software Oracle 10g [Oracle10gMan] e [Alapati 2005].

Tali software, infatti, permettono alle aziende di iniziare agevolmente l'evoluzione verso il modello del Grid Computing. Assieme, Oracle Database 10g, Oracle Application Server 10g e Oracle Enterprise Manager 10g forniscono il primo software di infrastruttura completo per il Grid Computing.

Per Grid Computing, si intende un'infrastruttura software a "griglia" adattabile, che fa uso efficiente di server e di soluzioni di storage modulari, in grado di bilanciare i carichi di lavoro in modo efficace e di garantire una vera capacity on demand, ossia garantire capacità di elaborazione nel momento del bisogno.

Il punto di forza dei sistemi di Grid Computing è sicuramente il fatto di rendere le risorse molto flessibili e capaci di adattarsi alle esigenze.

Attraverso l'implementazione del modello di Grid Computing è possibile creare un'infrastruttura software di importanza critica, eseguibile su un vasto numero di piccoli computer collegati in rete [Chappell 2004] e [GRID 2005], che permette:

- **Di costruire un'entità a partire da un insieme di componenti.** Il grid computing coordina l'uso di cluster di macchine per creare una singola entità logica, come un database. Questo permette di aumentare o ridurre la capacità in piccoli incrementi mirati, direttamente online e a macchine funzionanti. Grazie a questa possibilità di aggiungere capacità on demand ad una particolare funzione, le aziende possono beneficiare di una maggiore flessibilità per adattarsi ai picchi di lavoro, conseguendo così un migliore utilizzo dei componenti hardware e tempi di risposta ottimali per il business.
- **Di gestire un'insieme di componenti come fosse una sola entità.** Il grid computing permette di gestire e amministrare gruppi di istanze di database, trattandoli come un'unica entità logica, semplificandone così la gestione.

Per capire meglio come funziona l'Oracle Grid, facciamo un esempio pratico in cui l'ausilio di tale funzione apporta dei benefici:

Supponiamo che un'azienda che produce giocattoli utilizzi due database server separati, uno per registrare gli ordini ricevuti ed uno per registrare le fatture e gli altri atti finanziari.

Accade che, nel mese di dicembre, quando si avvicina il Natale, l'azienda riceve un grosso numero di ordini di giocattoli, che provoca una grossa mole di lavoro per il database degli ordini. Tale database server rischia di non essere sufficiente per il lavoro da svolgere, e, quindi, di dovere essere supportato da un altro database server; Nello stesso periodo, invece, il database server della finanza è poco sfruttato (Fig. 7.1).

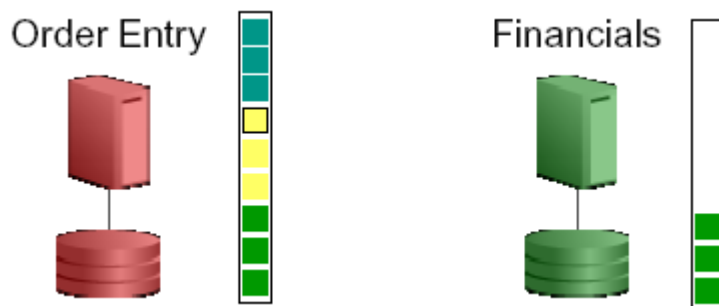


Fig. 7.1

Quando arriva gennaio, gli ordini di giocattoli calano, mentre aumentano le fatture e i fatti finanziari da registrare, derivanti dal grosso numero di ordini ricevuti nel mese di dicembre. In tale situazione quindi è il database della finanza ad avere bisogno di un altro database a fianco per gestire il grosso carico di lavoro, mentre il database degli ordini è poco sfruttato (Fig. 7.2).

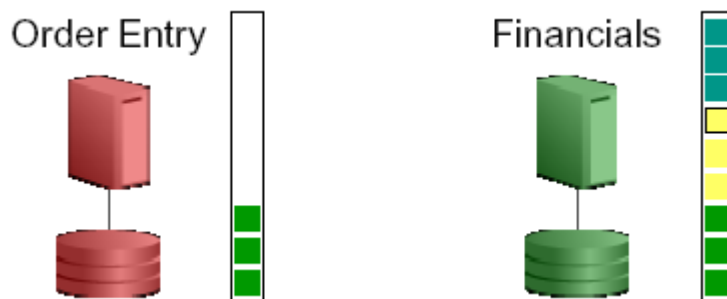


Fig. 7.2

Quindi per aumentare le prestazioni del sistema appena descritto, sarebbero necessari quattro database, due per gestire gli ordini, e due per gestire la finanza.

Oracle Grid permette di creare un sistema equivalente utilizzando solamente tre database, risparmiando quindi l'acquisto di un database.

Con lei, infatti, è possibile mettere tre database in "griglia" e fare in modo che, secondo le esigenze, uno di questi database funzioni, sia come database degli ordini, che come database della finanza.

Questa situazione è ben rappresentata dalla figura 7.3 in cui sono inserite in "griglia" tre istanze che "girano" su due database, quello degli ordini e quello della finanza. Grazie all'Oracle Grid, l'istanza di mezzo può supportare le due vicine secondo la necessità: a dicembre, può dedicarsi alla registrazione degli ordini, mentre a gennaio, può occuparsi della finanza.



Fig. 7.3

Ne risulta così un carico di lavoro più bilanciato su tutte e tre le istanze. Questo appena descritto è il concetto di capacità on demand su cui si fonda il Grid Computing: le risorse sono così flessibili che sono impiegabili in ogni momento là dove ce ne è bisogno.

Come già parzialmente illustrato nel Capitolo 3, i database server Oracle 10g hanno numerose funzionalità specifiche per gli ambienti Grid. Attualmente sul mercato nessun altro vendor mette a disposizione un vero Grid database. Oracle Grid si basa sui **Real Application Cluster (RAC)**, introdotti già in Oracle 9i.

Di seguito vengono riportate le caratteristiche principali che rendono il database Oracle 10g un database per gli ambienti di Grid Computing.

### **Real Application Cluster**

Oracle Real Application Clusters consente di eseguire un singolo database su più nodi in cluster di un Grid, con il pooling delle risorse di elaborazione di diverse macchine standard.

Oracle offre caratteristiche di flessibilità esclusive grazie alla sua capacità di fornire il carico di lavoro su più macchine, in quanto si tratta dell'unica tecnologia di database a non richiedere il partizionamento e la distribuzione dei dati assieme al lavoro.

In Oracle 10g, il database può iniziare immediatamente il bilanciamento del carico su un nuovo nodo con una nuova capacità di elaborazione, mentre viene riassegnato da un database a un altro, e può rinunciare a una macchina quando non è più necessaria. Altri database non possono ampliarsi e contrarsi durante l'esecuzione e, pertanto, non possono utilizzare l'hardware con altrettanta efficienza.

Il nuovo clusterware (software in grado di costruire il cluster di server dal punto di vista del sistema operativo) integrato in Oracle 10g semplifica il clustering eliminando la necessità di acquistare, installare, configurare e supportare clusterware di terze parti. In questo modo sarà possibile aggiungere e rimuovere facilmente i singoli server dal cluster Oracle, senza alcun downtime.

Oracle dispone dell'unica tecnologia di database che include il clusterware per tutti i sistemi operativi ed è in grado di ridurre al minimo le possibilità di errore in un ambiente in cluster.

### **Automatic Storage Management**

Automatic Storage Management semplifica la gestione dello storage per i database Oracle.

Astraendo i dettagli di gestione dello storage, Oracle migliora le prestazioni di accesso ai dati tramite una ripartizione molto sofisticata, senza richiedere ulteriore lavoro dei DBA.

Invece di gestire molti file di database, i DBA Oracle dovranno intervenire solo su un numero ridotto di gruppi di dischi (insiemi di dispositivi disco che Oracle gestisce come un'unica unità logica). L'amministratore può definire un particolare gruppo di dischi come predefinito per il database e Oracle procederà automaticamente all'allocazione dello storage e a creare o eliminare i file associati all'oggetto database. Automatic Storage Management offre inoltre i vantaggi delle tecnologie di storage, quali RAID o Logical Volume Manager (LVM).

Oracle consente di bilanciare l'I/O di più database su tutti i dispositivi in un gruppo di dischi e implementa lo striping e il mirroring per migliorare le prestazioni di I/O e l'affidabilità dei dati.

In più, Oracle può riassegnare i dischi da nodo a nodo e da cluster a cluster, riconfigurando automaticamente il gruppo. Poiché Automatic Storage Management è

concepito per lavorare esclusivamente con Oracle, garantisce prestazioni ottimali rispetto alle soluzioni generiche di virtualizzazione dello storage.

### **Allocazione delle informazioni**

Oltre alla distribuzione del lavoro su più nodi e dei dati su più dischi, in Oracle Database 10g ha luogo un ulteriore tipo di allocazione: quella delle informazioni stesse.

Secondo il volume delle informazioni e della frequenza di accesso, può essere necessario spostare i dati da dove risiedono attualmente oppure dividerli su più database.

Oracle 10g include diverse funzionalità per consentire l'accesso alle informazioni dove e quando sono necessarie, per collegare chi fornisce le informazioni a coloro che le richiedono.

La funzionalità più spinta ed in tempo reale in tal senso è Oracle Stream, di cui abbiamo parlato nel capitolo precedente, che permette di migrare i dati da un database a un altro mentre sono entrambi online.

I trasferimenti di grandi quantità di dati, maggiormente idonei in talune circostanze, sono gestiti da alcune funzionalità come la *Data Pump* e *Transportable Tablespaces*.

## **2. Applicazione al problema ed esiti dell'azione**

Tornando a quello che era il nostro problema di partenza, cioè lo studio dei vari metodi per la sincronizzazione tra database, possiamo osservare come teoricamente l'infrastruttura di Grid Computing potrebbe garantire una soluzione per la condivisione dei dati.

Un cluster di database è formato da un insieme di database che condividono dati.

Per la sincronizzazione tra due database, quindi, è possibile creare un cluster formato dai due database.

Tuttavia la condivisione dei dati non è il motivo principale che spinge le aziende a creare cluster di database; essi sono creati soprattutto per gestire una grossa mole di dati, per bilanciare il lavoro su più macchine e per evitare che guasti influenzino la produzione.

L'utilizzo del Grid Computing per la sola replica dei dati tra database, è un concetto piuttosto forzato, in quanto l'infrastruttura Grid è stata introdotta per altri scopi.

Utilizzare il paradigma di Grid Computing solo per poter condividere dati tra due database, può essere paragonato ad andare a fare la spesa con un'auto da Formula 1: possiamo sicuramente arrivare al supermercato, ma l'auto è stata concepita per fare gare su piste, ed utilizzarla per andare a fare la spesa, rappresenterebbe uno spreco.

Allo stesso modo come esistono auto più adatte per andare a fare la spesa rispetto ad un'auto da Formula 1, esistono metodi migliori e più adatti per la semplice sincronizzazione tra database, come quelli visti e discussi fino a qui.

Durante lo svolgimento dell'esperimento preso in considerazione in questa tesi sono emersi alcuni limiti di applicabilità dell'Oracle Grid.

Come detto Oracle Grid si basa sul RAC Real Application Cluster. Tale funzionalità permette di creare cluster di database; un limite però del RAC è rappresentato dal fatto che tutti i database del cluster devono essere Oracle, e, più precisamente,

occorre che siano della stessa versione; E' quindi possibile creare cluster di database Oracle 10g Enterprise Edition, ma non cluster di database in cui parte dei database è Standard Edition e parte è Enterprise Edition.

Inoltre il meccanismo dei RAC non è presente su tutte le versioni, ma solo nei database Oracle 10g Enterprise Edition e Standard Edition.

Questo fatto non permette di includere in sistemi Grid database Oracle più piccoli come ad esempio l'Express Edition.

Questa è la motivazione principale per cui, nel nostro modello pratico non è stato possibile creare un'infrastruttura che utilizzasse l'Oracle Grid come strumento di sincronizzazione tra il database Standard Edition e quello Express Edition.

Per concludere si può dire che il Grid Computing è uno standard emergente che sta prendendo piede soprattutto negli ambiti aziendali, che garantisce vantaggi in termini di flessibilità dei sistemi, disponibilità delle risorse, possibilità di scalabilità e contenimento di costi.

# Capitolo 8

## Uso di Oracle SQL Developer

### 1. Applicazione al problema

Oracle SQL Developer è un'applicazione client di database server, sviluppato da Oracle Corporation per l'interazione con i database, dotata di tutte le caratteristiche di un IDE per la programmazione SQL.

Lo scopo principale di Oracle SQL Developer è quello di facilitare la creazione, la modifica e la gestione degli oggetti di un database relazionale e di renderla comprensibile anche a chi non conosce il linguaggio SQL.

Questa applicazione funge quindi anche da terminale client che, anziché essere testuale come SQL\*Plus, presenta un'interfaccia grafica che permette, attraverso l'utilizzo di opportuni bottoni, di compiere tutte le operazioni che si possono eseguire con il linguaggio SQL.

SQL Developer permette di interfacciarsi con tutti i database Oracle anche di precedenti versioni rispetto alla 10g, ma anche con database di altri vendor.

Per tutti gli esempi e gli screenshot illustrati in questo capitolo ci riferiremo a Oracle SQL Developer versione 1.2.0.




All'interno di SQL Developer è incorporato un tool, che permette di mettere in evidenza le differenze, in termini di oggetti di database, tra due schemi appartenenti allo stesso database, ma anche a database diversi. Tale tool prende il nome di **Schema Diff**.

Proprio grazie a tale tool, SQL Developer rappresenta un modo alternativo ai precedenti, per la sincronizzazione tra database.

Vediamo ora le operazioni da compiere per sincronizzare due schemi appartenenti a database diversi (per schemi dello stesso database la procedura è simile).

Come prima cosa dopo aver avviato SQL Developer occorre creare le connessioni ai due database.

Per fare questo è opportuno selezionare il bottone  in alto a sinistra nella scheda Connections.

A questo punto si apre la finestra relativa alle connessioni (Fig. 8.2);

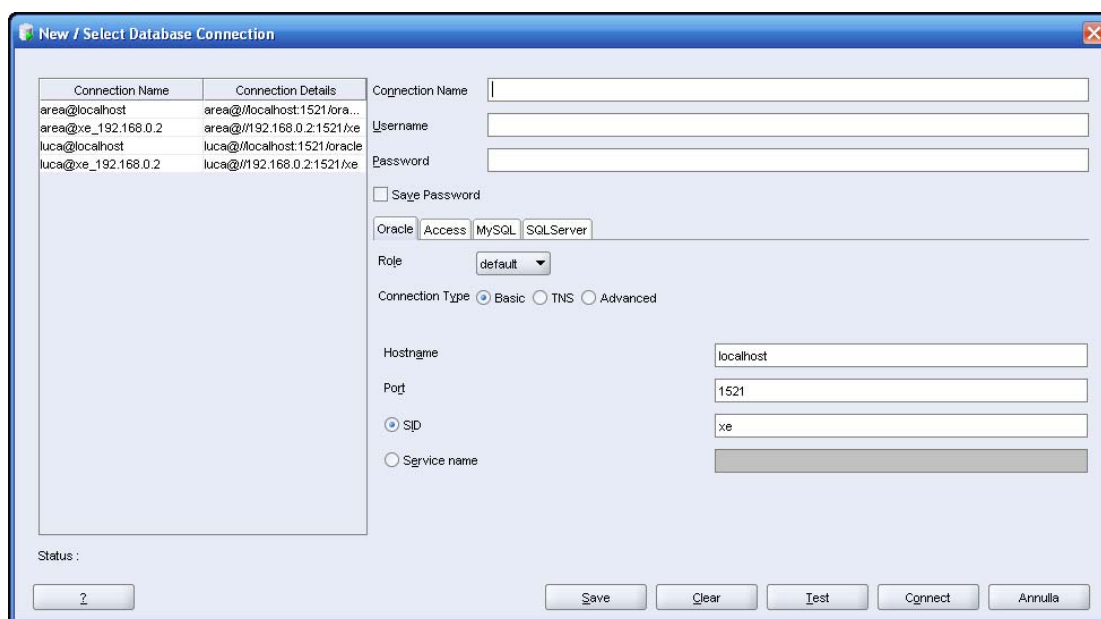


Fig. 8.2

In essa sono richiesti: il nome da attribuire alla connessione che si sta creando, il nome utente e la relativa password con il quale si ci vuole connettere al database (è opportuno specificare che una connessione permette di visualizzare solo gli oggetti appartenenti allo schema dell'utente con cui si è creata la connessione al DB) e le informazioni sul database a cui si ci connette. Le informazioni sul database cambiano a seconda del vendor del database: con SQL Developer è infatti possibile connettersi a diversi database, come ad esempio i database Oracle, Access, MySQL e SQL Server. Nel nostro caso specifico si seleziona la scheda identificata con Oracle e si inserisce: il tipo di connessione, l'host name su cui è memorizzato il database, la porta, e il SID. Nel caso dello Standard Edition si inserisce:

*Connection type:* BASIC

*Hostname:* LOCALHOST

*Port:* 1521

*SID:* ORACLE

Nel caso dell'Express Edition invece si inserisce:

*Connection type:* BASIC

*Hostname:* 192.168.0.2

*Port:* 1521

*SID:* XE

Una volta inseriti i dati relativi alle connessioni, queste si salvano per evitare che tali dati siano reinseriti tutte le volte che si accede al database.

Quando si è ultimata la creazione di entrambe le connessioni è possibile accedere al database relativo, semplicemente cliccando sul nome della connessione, facendo aprire, così, un menù a tendina che riporta i vari oggetti presenti nello schema dell'utente con cui si è creata la connessione.

A questo punto è possibile utilizzare il tool Schema Diff.

Per lanciare Schema Diff occorre selezionare nel menù in alto la voce **Tools** e nella finestra a tendina che si apre la voce **Schema Diff** (Fig. 8.3).

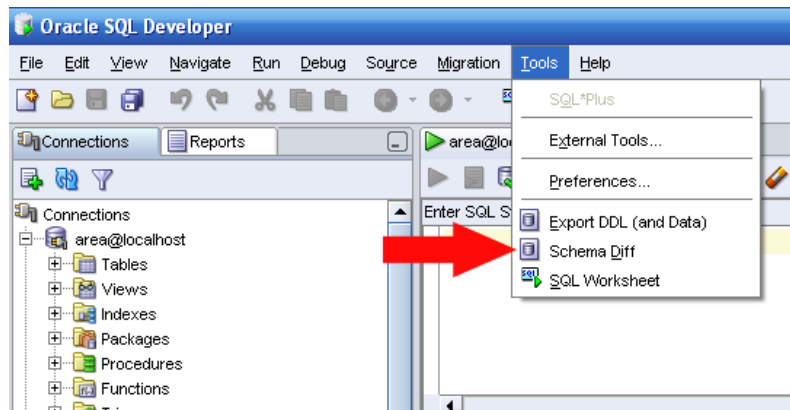


Fig. 8.3

A questo punto si apre la finestra relativa al tool Schema Diff; tale finestra presenta tre schede:

- La scheda **Source**: (Fig. 8.4) in essa è possibile scegliere le opzioni per lo schema di partenza, ossia lo schema da comparare allo schema di arrivo: in particolare si sceglie lo schema di partenza, e quali oggetti di questo schema paragonare (FUNCTION, MATERIALIZED VIEW, PACKAGE, PROCEDURE, TABLE, VIEW).

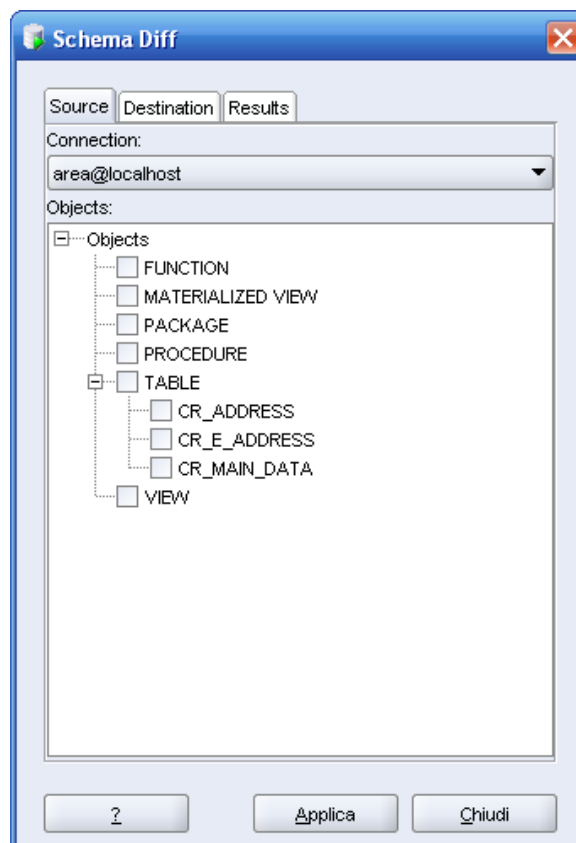
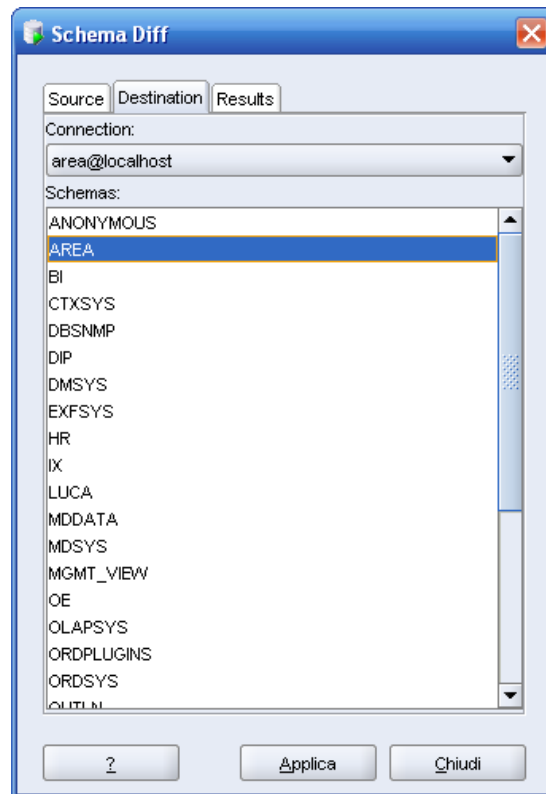


Fig. 8.4

- La scheda **Destination**: (Fig. 8.5) in essa è possibile impostare lo schema di arrivo, ossia lo schema che verrà comparato con quello di partenza.



*Fig. 8.5*

- La scheda **Results:** (Fig. 8.6) in essa vengono scritti i comandi SQL che permettono la sincronizzazione tra lo schema di arrivo e lo schema di partenza; una volta scelti i due schemi nelle due precedenti schede e selezionato **Applica** compare in questa scheda le istruzioni SQL, la cui esecuzione porta alla creazione e alla modifica degli oggetti che sono nello schema di partenza e che non sono nello schema di arrivo. E' anche possibile salvare tali istruzioni in uno script da eseguire in un secondo momento.

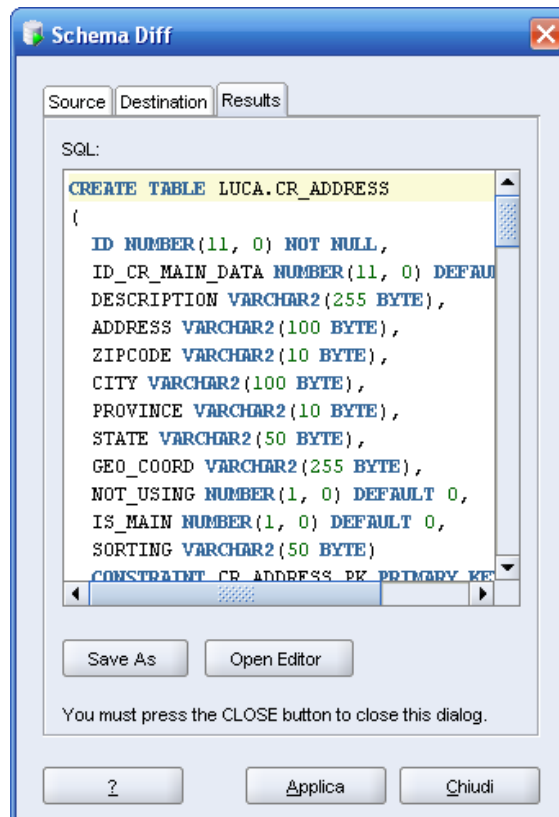


Fig. 8.6

Come dice il nome di questa funzione, Schema Diff non fa altro che evidenziare le differenze tra gli oggetti dello schema sorgente e dello schema di destinazione e inoltre illustra quali comandi SQL bisogna eseguire sul database di destinazione per sincronizzare gli schemi.

Naturalmente per una sincronizzazione totale tra gli schemi, può essere necessario scambiare lo schema sorgente e lo schema di destinazione, perché ci possono essere oggetti presenti nello schema di destinazione che non sono presenti nello schema sorgente.

## 2. Esiti e limiti emersi

L'utilizzo della funzione *Schema Diff* come metodo di sincronizzazioni tra schemi è molto efficace ed efficiente. Esso permette di paragonare tutti gli schemi di un database e di database diversi, e di creare i comandi SQL capaci di sincronizzare gli schemi. In tal modo, in maniera automatica, vengono determinate le differenze tra gli schemi e indicate le correzioni da attuare.

Purtroppo però la sincronizzazione così ottenibile non prende in considerazione i dati veri e propri, intesi come ad esempio i record di una tabella, ma prende in considerazione solo gli oggetti del database, come viste, tabelle etc.

Questo fatto fa sì che non si possano replicare i dati di un database in un altro, ma solamente gli oggetti in questo presenti.

La funzione *Schema Diff* rappresenta comunque uno strumento utile ed immediato per la replica degli oggetti di uno schema.

# Capitolo 9

## Conclusioni

### 1. Analisi del lavoro svolto e confronto tra i metodi

Alla luce delle analisi svolte sulle varie soluzioni per la sincronizzazione tra database fatte fino a questo momento, è possibile, ora, redigere un bilancio che paragoni le varie soluzioni e permetta di indicare quale, tra le tante, sia la migliore in termini di praticità, per essere applicata nella situazione, da noi presa come modello, del “commesso viaggiatore”.

Prima di fare paragoni, però, è utile riepilogare i risultati emersi per ogni soluzione di sincronizzazione vista:

- *DB Link*: come abbiamo visto nei database Oracle è possibile creare oggetti particolari che permettono di collegare due database; tali oggetti sono detti DB Link. Essi, una volta opportunamente configurati, danno la possibilità di accedere in lettura a dati remoti, memorizzati, cioè, su database remoti, collegati in rete. Essi permettono di creare un canale di comunicazione unidirezionale, tra un database sorgente e uno destinazione.
- *Oracle Stream*: gli Oracle Stream catturano le modifiche che avvengono in un database, le propagano attraverso una serie di code, e poi le applicano ad un database di arrivo; Essi quindi sono formati da tre processi principali, che possono essere personalizzati, detti *capture*, *propagation* e *apply*. Da test eseguiti sugli stream è emerso che questi sono realizzati per mezzo di una *Datapump*, ossia una funzione Oracle che permette di fare l'import/export dei dati, eseguita su di un *DB Link*. Anche in questo caso il canale di comunicazione è unidirezionale.
- *Oracle Grid*: per Oracle Grid, si intendono una serie di funzioni incorporate in alcuni database Oracle 10g, che permettono di creare una “griglia” di risorse informatiche, in primo luogo i database server, che possono essere utilizzate nel momento del bisogno indipendentemente dalla loro ubicazione. Per Grid Computing, abbiamo visto che, si intende uno standard, relativamente nuovo, che si basa sul concetto di *capacity on demand*: si deve creare un sistema in cui le risorse sono molto flessibili e si adattano ad essere utilizzate là dove ce ne è più bisogno. In tale sistema quindi è possibile incrementare la capacità elaborativa in qualsiasi momento. Tale standard è implementato in alcune versioni dei database Oracle per mezzo di funzioni specifiche per gli ambienti distribuiti: RAC, ASM, Database Autogestito, Allocazione delle informazioni.
- *SQL Developer*: Attraverso l'uso dell'applicativo SQL Developer e più precisamente attraverso l'uso del tool Schema Diff, in esso incorporato, è

possibile confrontare schemi di database differenti e, attraverso l'esecuzione di comandi SQL che schema Diff crea, sincronizzare i due schemi.

In questa soluzione è possibile selezionare quali oggetti dello schema si vogliono comparare; Schema Diff quindi permette di evidenziare le differenze in termini di oggetti tra due schemi e ci suggerisce i comandi SQL per renderli uguali. Questa soluzione non copia i dati presenti negli oggetti, ma permette solamente di sincronizzare gli oggetti; Se ad esempio scegliamo di sincronizzare due schemi in cui quello di partenza ha una tabella in cui sono stati inseriti dei record, e quello di destinazione non la ha, schema Diff permette, attraverso l'esecuzione sul database di destinazione dei comandi da lui creati, di replicare la tabella, ma non i record in essa presenti.

Si vuole ora indicare quale tra questi metodi, si adatta meglio al nostro problema di partenza. Per "adatta meglio" si intende quale soluzione richiede la minore necessità di scrivere istruzioni nel database e/o programmi esterni, per compiere materialmente l'operazione di sincronizzazione nel minor tempo possibile e nel modo più "automatico" possibile.

Sicuramente il metodo dei *DB Link* rappresenta uno strumento che si adatta bene a svolgere compiti di condivisione dei dati; esso è facilmente implementabile e non necessita di gestioni complicate. Inoltre una volta creato un *DB Link* esso è utilizzabile in qualsiasi momento, semplicemente referenziandolo.

Il limite però che i *DB Link* presentano è quello di non fornire "in automatico" la replica delle informazioni, ma solamente la condivisione dell'accesso alle stesse.

Tale limite è superato con l'introduzione degli Oracle Stream; essi garantiscono le stesse prestazioni e la stessa semplicità di gestione dei *DB Link*, ma in più permettono di replicare i dati da un database all'altro; rispetto ai *DB Link*, gli Stream hanno una procedura di creazione indubbiamente più complessa, ma grazie alla creazione guidata messa a disposizione dalla *DB Console Oracle* il loro uso è più facile.

In definitiva, quindi, tra i vari metodi di sincronizzazione che la tecnologia Oracle mette a disposizione nei propri database (e quindi si escludono gli applicativi ad hoc creati "artigianalmente" e tutti gli altri applicativi che si possono reperire atti a tale scopo) gli Oracle Stream rappresentano la soluzione che meglio si adatta al problema del "commesso viaggiatore".

## **2. Possibilità di estensioni future**

L'esperimento trattato nella presente tesi di laurea, prende in considerazione solamente le soluzioni di sincronizzazione tra gli specifici database Oracle 10g Standard Edition e Oracle 10g Express Edition. Il fatto di utilizzare tali strumenti restringe il campo dei metodi applicabili. Un'estensione possibile a questo esperimento potrebbe essere quella di analizzare le soluzioni di sincronizzazione con database Oracle che incorporano un più ampio numero di funzionalità come l'Oracle 10g Enterprise Edition.

# Bibliografia

[ACPT 2002] P. Atzeni, S. Ceri, S. Paraboschi, R. Torlone - Basi di Dati: Modelli e Linguaggi di Interrogazione - McGraw-Hill Italia, Milano, 2002

[ACFPT 2003] P. Atzeni, S. Ceri, P. Fraternali, S. Paraboschi, R. Torlone - Basi di Dati: Architetture e Linee di Evoluzione - McGraw-Hill Italia, Milano, 2003

[Alapati 2005] S.R. Alapati - Expert Oracle Database 10g Administration (Expert's Voice) – Ed. Apress, 2005

[AreaSPOracle 2007] - "Materiale Didattico Corso di Oracle 10g DBA e SQL" di AreaSP.

[BFM 2001] G. Bracchi, C. Francalanci, G. Motta - Sistemi Informativi e aziende in rete - McGraw-Hill Italia, Milano, 2001

[Chappell 2004] D.A. Chappell – Enterprise Service Bus – O'Reilly, 2004

[Ciborra 2002] C. Ciborra - The Labyrinths of Information: Challenging the Wisdom of Systems - Oxford University Press, Oxford. 2002

[DBJW 2004] C. Dawes, B. Bryla, J.C. Johnson, M. Weishan - Oracle 10g Administration I Study Guide (1Z0-042) - Ed. Sybex 2004

[De Marco 2000] M. De Marco - Sistemi Informativi Aziendali - Franco Angeli Edizioni, Milano 2000

[Destri 1998] G. Destri. Una rete privata virtuale con F-Secure SSH, LOGIN n. 10, Edizioni Infomedia, Maggio/Giugno 1998.

[Destri 2007] G. Destri - "Introduzione ai sistemi informativi aziendali" - Monte Università Parma Editore, 2007

[GRID 2005] sito web principale sul Grid Computing, <http://www.gridcomputing.com/>

[OHE 1999] R. Orfali, D. Harkey, J. Edwards - Client/Server Survival Guide, 3rd Edition – Ed. Wiley, 1999

[Oracle10gMan] Documentazione ufficiale di Oracle Corporation sul DB Server 10g, scaricabile da <http://www.oracle.com/technology/documentation/database10gr2.html>

[Zaffanella 2002] E. Zaffanella – Lucidi del corso di Basi di Dati 2002