Università degli Studi di Modena e Reggio Emilia

Facoltà di Ingegneria – Sede di Modena

Corso di Laurea in Ingegneria Informatica – *Nuovo Ordinamento*

Confronto tra Microsoft SQL Server 2000 e MySQL 5.0

Relatore:
Chiar.mo Prof. Sonia Bergamaschi

Candidato:
Entela Kazazi

Correlatore:
Ing. Antonio Sala

Anno Accademico 2005-2006

Indice

1. Introduzione

1.1 Obiettivo della tesi …… 1

1.2 Database , DBMS, RDBMS …… 2

1.3 SQL …… 3

2. Caratteristiche Generali dei DBMS

2.1 Caratteristiche di MySQL ……..... 4

2.1.1 Descrizione di MySQL 5.0 …… 4
2.1.2 Portabilità di MySQL 5.0 e compatibilità con SQL92 ……... 6
2.1.3 Punti di forza di MySQL 5.0 …… 6

2.2 Caratteristiche di Microsoft SQL Server 2000 ……... 7

2.2.1 Descrizione di Microsoft SQL Server 2000……... 7
2.2.2 Portabilità di SQL Server 2000 e compatibilità con SQL92…… 8
2.2.3 Punti di forza di SQL Server 2000 ……..…….. 9

2.3 Installazione ……... 10

2.3.1 Installazione di MySQL 5.0 e configurazione …… 10
2.3.2 Installazione di SQL Server 2000 e configurazione …… 11

2.4 Costi …… 13

2.4.1 Costi di MySQL 5.0 ....…... 13
2.4.2 Costi di SQL Server 2000 ……... 14

2.5 Sicurezza …… ……... 15

2.5.1 Sicurezza di MySQL 5.0 …… 16
2.5.2 Sicurezza di SQL Server 2000 ……... 21

3. Prestazioni

3.1 Database "genes22" …… 28

3.1.1 Select ……... 29
3.1.2 Join …… 30
3.1.3 Self Join ……... 32
3.1.4 Left Join/Right Join/Full Join …… 33
3.1.5 Join tra 3 tabelle ……..... 35
3.1.6 Join tra 4 tabelle …… 38
3.1.7 Join tra 5 tabelle …… 40
3.1.8 Join tra 6 tabelle …… 41
3.1.9 Conclusioni ……... 42

3.2 Database "marker19"…… 43

3.2.1 Select ……... 44
3.2.2 Join …… 45
3.2.3 Self Join ……... 46
3.2.4 Left Join/Right Join/Full Join …… 47
3.2.5 Join tra 3 tabelle ……..... 49
3.2.6 Join tra 4 tabelle …… 49
3.2.7 Query Innestate …… 50
3.2.8 Conclusioni ……... 52

4. Conclusioni ……... 53

5. Bibliografia ……... 54

Indice delle figure

Figura 1 - Fase di autenticazione in MySQL 5.0 …….. 17

Figura 2 - Scelta della modalità di autenticazione in SQL Server 2000 … 22

Figura 3 - Database "genes22"…… 28

Figura 4 - Database "marker19"…….. 43

Ringraziamenti

Desidero ringraziare anzitutto la Professoressa Sonia Bergamaschi per l'aiuto fornitomi durante lo svolgimento della mia tesi e per la disponibilità dimostrata.

Un ringraziamento particolare va all' Ing Antonio Sala per la sua disponibilità e per il suo prezioso aiuto.

Esprimo tutta la mia gratitudine ai miei genitori per il loro costante supporto nell'intero arco dei miei studi, per la fiducia e per tutti i sacrifici che hanno fatto per me.

Grazie anche al mio fidanzato e a mio fratello per il supporto e per l'affetto che mi hanno sempre dimostrato.

Un ringraziamento va a tutti i miei amici per il sostegno e per la forza che mi hanno dato durante questi anni.

Grazie di cuore a tutti…

Entela Kazazi

1. Introduzione

1.1 Obbiettivo della tesi

Le informazioni sono uno dei beni più preziosi della nostra società. La gestione delle informazioni con strumenti informatici avviene normalmente tramite una base di dati, che è una collezione di dati che rappresentano le informazioni di interesse per un organizzazione. Durante l'elaborato di questa tesi si cercherà di mettere a confronto due DBMS : l'open source MySQL 5.0 e Microsoft SQL Server 2000 che come dedotto dal nome è un DBMS sviluppato da Microsoft.
Il confronto si concentrerà sui seguenti aspetti:

1. Portabilità
2. Compatibilità con lo standard SQL92
3. Punti di forza
4. Installazione e richieste hardware
5. Costi e licenze
6. Sicurezza

L'obbiettivo del secondo capitolo sarà quello di vedere le prestazioni dei due DBMS. Verranno effettuate delle query su due database: "genes22" e "marker19" in cui sono contenuti dati relativi ai geni e ai marker dei cereali e si confronteranno i tempi di risposta. I due database differiscono per il numero di record in essi memorizzati :"genes22" contiene 58557 record e "marker19" contiene 5,832,461 record.

1.2 Database, DBMS e RDBMS

Il termine **database**, tradotto in italiano con **banca dati** o **base di dati** indica un insieme di dati riguardanti uno stesso argomento, o più argomenti correlati tra loro, strutturata in modo tale da consentire l'uso dei dati stessi (e il loro aggiornamento) da parte di applicazioni software. La base di dati, oltre ai dati veri e propri, deve contenere anche le informazioni sulle loro rappresentazioni e sulle relazioni che li legano.

Un **DBMS (Database Management System)** è un sistema software in grado di gestire una collezione di dati che sono condivisi da più applicazioni e utenti. Un DBMS deve:

- gestire una grande quantità di dati.
- gestire la condivisione dei dati tra diversi utenti ed applicazioni.
- gestire le transazioni.
- garantire l' affidabilità dei dati, cioè la capacità di ripristino a fronte di malfunzionamenti (resilienza), meccanismi di salvataggio (backup) e ripristino (recovery).
- offrire una "visione strutturata" dei dati in base al modello logico usato.
- garantire la privatezza dei dati in base a dei meccanismi di autorizzazione.

Un **RDBMS** (**Relational Database Management System)** è un sistema relazionale per la gestione di basi di dati ed indica un DBMS basato sul modello relazionale ed è stato introdotto da Edgar F.Codd. I requisiti minimi per cui un DBMS può essere chiamato RDBMS sono:

- deve presentare i dati all'utente sotto forma di relazioni (una presentazione a tabelle può soddisfare questa proprietà) .
- deve fornire operatori relazionali per manipolare i dati in forma tabellare.

1.3 SQL

SQL (**Structured Query Language**) è un linguaggio creato per l'accesso a informazioni memorizzate nei database. L'SQL nasce nel 1974 ad opera di Donald Chamberlin, nei laboratori dell'IBM. Nel 1983 IBM rilasciò DB2, il suo DBMS

relazionale più diffuso ancora oggi. SQL divenne subito lo standard industriale per i software che utilizzano il modello relazionale. L'ANSI lo adottò come standard fin dal 1986, senza apportare modifiche sostanziali alla versione inizialmente sviluppata da IBM. Nel 1987 la ISO fece lo stesso. Questa prima versione standard è denominata SQL/86. Negli anni successivi si realizzarono altre versioni, che furono SQL/89, SQL/92 e SQL/2003.

Struttura

Essendo un linguaggio dichiarativo, SQL non richiede la stesura di sequenze di operazioni (come ad es. i linguaggi imperativi), piuttosto di specificare le proprietà logiche delle informazioni ricercate. Esso si divide in tre sottoinsiemi:

- DDL (Data Definition Language) - DDL serve a creare, modificare o eliminare gli oggetti in un database. Sono i comandi DDL a definire la struttura del database e quindi dei dati ivi contenuti. Ma non fornisce gli strumenti per modificare i dati stessi: per tale scopo di usa il DML. L'utente deve avere i permessi necessari per agire sulla struttura del database e questi permessi vengono assegnati tramite il DCL (Data Control Language).
- DML (Data Manipulation Language) -DML fornisce i comandi per inserire, modificare, eliminare o leggere i dati all'interno delle tabelle di un database. La struttura di questi dati deve già essere stata definita tramite il DDL. Esempi di comandi sono SELECT,INSERT, UPDATE, DELETE etc.
- DCL (Data Control Language) - DCL serve a fornire o revocare agli utenti i permessi necessari per poter utilizzare i comandi DML e DDL, oltre agli stessi comandi DCL (che gli servono per poter a sua volta modificare i permessi su alcuni oggetti).

Operatori

Gli operatori, messi a disposizione da SQL standard si dividono in quattro categorie:

- Operatori di confronto
- Operatori aritmetici
- Operatori condizionali
- Operatori logici

2. Caratteristiche Generali dei DBMS

2.1 Caratteristiche di MySQL

2.1.1 Descrizione di MySQL 5.0

MySQL è un RDBMS open source composto da un client con interfaccia a caratteri e un server. Il codice di MySQL viene sviluppato fin dal 1979 dalla ditta TcX ataconsult, adesso MySQL AB , ma è solo dal 1996 che viene distribuita una versione che supporta SQL. MySQL svolge il compito di DBMS nella piattaforma LAMP, una delle più usate e installate su Internet per lo sviluppo di siti e applicazioni web dinamiche. Per esempio il software MediaWiki , che gestisce i siti del progetto Wikipedia , è basato su database MySQL. Esistono diversi tipi di MySQL Manager , ovvero di strumenti per l'amministrazione di MySQL. Uno dei programmi più popolari per amministrare i database MySQL è phpMyAdmin che richiede un server web come Apache_HTTP_Server ed il supporto del linguaggio PHP. Si può utilizzare facilmente tramite un qualsiasi browser web. Un'alternativa è rappresentata da MySQL-Front. In alternativa la stessa MySQL AB offre programmi quali MySQLcc (MySQL control center), MySQL Administrator (amministrazione del database, degli utenti, operazioni pianificate, carico del server, ...) e MySQL Query Browser (per l'esecuzione di svariati tipi di query). Possiede delle interfacce per diversi linguaggi di programmazione: C ,C++,C#, Perl ,PHP, Python, compreso un driver ODBC (Open Database Connectivity), due driver Java e un driver per Mono e .NET.

Il 22 dicembre 2003 viene rilasciata la prima versione della serie 5.0, che è entrata in produzione il 19 ottobre 2005. Le funzionalità più significative aggiunte a questa versione rispetto alle precedenti sono:

- **le viste** : tabelle virtuali ricavate da una query SQL.
- **le stored procedures** : come nei linguaggi di programmazione esistono vari metodi per scrivere una sola volta il codice da eseguire in diversi punti, così nel linguaggio SQL è possibile stabilire a priori delle funzioni o procedure personalizzate. Questo passaggio permette di:

  1. Rendere disponibile agli script che si interfacciano con il nostro database un set di API (application program interface) in modo da essere sicuri del funzionamento di una query su tutti i sistemi.

  2. Semplificare notevolmente le query attraverso l'uso di chiamate a funzioni personalizzate.

  Esattamente come le funzioni dei linguaggi di programmazione, le stored procedures possono essere composte da valori in input, valori in output, tipi di valori (è necessaria la dichiarazione del tipo di variabile utilizzata), costrutti di controllo del flusso (if/ then/ else, case, cicli), passaggio di variabile per indice e chiamate ad altre funzioni personalizzate. In MySQL le stored procedures devono essere chiamate sempre tramite una istruzione di tipo CREATE FUNCTION.

- **i trigger** : istruzioni SQL che vengono lanciate automaticamente prima o dopo l'esecuzione di determinate operazioni su determinate tabelle .

- **i cursori** : sono dei puntatori ai recordset di specifiche query utili soprattutto nell'ambito delle procedure personalizzate. Permettono di accedere a questi recordset e tramite il comando FETCH assegnare i valori dei vari campi a specifiche variabili.
- MySQL introduce anche il concetto di **transazione**, anche se limitatamente alle sole tabelle di tipo **InnoDB**. Una transazione permette di portare a termine una serie di query solo se tutte quante vanno a buon fine. Nel caso in cui anche una sola delle query generi un errore è possibile gestirlo per evitare inconsistenze nel database.
- **INFORMATION_SCHEMA** : un database virtuale che descrive la struttura di tutti gli altri database; inoltre i comandi SHOW, che anch'essi restituiscono informazioni sulla struttura dei database, sono stati potenziati .
- **indici mobili** .
- il tipo di **dati BIT** .
- gestione appropriata del fuso orario (timezone) .
- i tipi di tabella **Archive** e **Federated** (per salvare i dati in un server remoto).
- un'**API (**application program interface**)** ben strutturata per sviluppare nuovi tipi di tabelle .

Tutte queste funzionalità sono state aggiunte allo scopo di avvicinare le funzionalità offerte dal prodotto con quelle dei DBMS di classe enterprise sviluppati da grandi aziende come Oracle, IBM e Microsoft. Il celebre database open source include poi nuove caratteristiche che semplificano la migrazione dei dati dalle piattaforme proprietarie: tra queste vi è il Migration Toolkit in grado di importare, per mezzo di un front-end grafico, tutti i dati e gli oggetti archiviati nei database di Oracle, SQL Server e MS Access.
Le modalità di programmazione utilizzabili con MySQL sono molto ampie e complete. Per esempio si può utilizzare il linguaggio Java accedendo al DB con un driver JDBC Connector.
In MySQL una tabella può essere di diversi tipi (o storage engine). Ogni tipo di tabella presenta proprietà e caratteristiche differenti (transazionale o meno, migliori prestazioni, diverse strategie di locking, features particolari, ecc). Esiste poi un'API che si può utilizzare per creare in modo relativamente facile un nuovo tipo di tabella, che poi si può installare senza dover ricompilare o riavviare il server.

Gli storage engine più usati sono:

- **MYISAM**: Engine di default molto veloce e compatto.
- **INNODB**: Engine che consente la gestione completa delle transazioni.
- **MEMORY**: Engine che mantiene in memoria i dati. Velocissimo ed ovviamente adatto per le elaborazioni temporanee.
- **NDB**: Engine con funzionalita' di cluster.

2.1.2 Portabilità di MySQL 5.0 e compatibilità con SQL92

MySQL, essendo scritto in linguaggio C e C++ è disponibile su molti differenti

sistemi operativi tra cui AIX, AmigaOS, BSDi, Digital Unix, FreeBSD, HP-UX, GNU/Linux, Mac OS X, NetBSD, Novell NetWare, OpenBSD, OS/2 Warp, SGI IRIX, Solaris, SunOS, SCO OpenServer, SCO UnixWare, SGI Irix, Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows 2003 ,Windows XP. Dal 1996 supporta la maggior parte della sintassi SQL ma MySQL non è completamente compatibile allo standard ANSI SQL. Per esempio in MySQL 5.0 non è implementato il full outer join che è previsto invece dallo standard SQL92.

2.1.3 Punti di forza di MySQL 5.0

MySQL e' il DBMS relazionale Open Source piu' diffuso al mondo (Questo è lo slogan ufficiale: **MySQL :The world's most popular open source database**). L'aumento esponenziale della diffusione di MySQL è spiegabile con l'enorme successo del movimento open source.
I principali punti di forza sono:

- Gratis per l'utilizzo come open source. Il prodotto è open source, quindi viene fornito oltre al prodotto anche il codice sorgente e questo consente all' utente di vedere come funzionano i programmi e di modificarli a seconda delle proprie esigenze.
- Eccezionale diffusione, soprattutto per le applicazioni web.
- Disponibile anche con una licenza commerciale e supporto tecnico.
- Leggero e di poco impatto sui server su cui viene installato .
- Semplice nell'utilizzo, nella configurazione e nell'amministrazione.
- Elevate prestazioni.
- Consente l'utilizzo di differenti storage engine tra cui scegliere.
- Disponibile per una grande varietà di piattaforme.

2.2 Caratteristiche di Microsoft SQL Server

2.2.1 Descrizione di Microsoft SQL Server 2000

Microsoft SQL Server è un RDBMS prodotto da Microsoft. Nelle prime versioni era utilizzato per basi di dati medio-piccole, ma negli ultimi cinque anni (a partire dalla versione 2000) è stato utilizzato anche per la gestione di basi di dati di grandi dimensioni. L'ingresso di Microsoft nel mondo dei database di fascia "enterprise" (DBMS che gestisce una grande quantità di dati) risale intorno al 1989 quando cominciò la competizione con Oracle, IBM e Sybase che erano i dominatori del mercato. La prima versione fu SQL Server per OS/2. SQL Server 2000 possiede delle interfacce per diversi linguaggi di programmazione: C, C++,C# , Perl ,PHP,Visual Basic e Python.Comprende anche i seguenti driver:ODBC (Open Database Connectivity), JDBC (Java Database Connectivity) , OLEDB e un driver per .NET . Microsoft SQL Server comunica sulla rete utilizzando un protocollo a livello di applicazione chiamato "Tabular Data Stream" (TDS).

SQL Server 2000 presenta rispetto alle versioni precedenti altre funzionalità migliorate, quali:

- vincoli di integrità referenziali a catena
- trigger INSTEAD OF e AFTER
- indici per le colonne calcolate
- nuovi tipi di dati (quali ad esempio i BIGINT)
- regole di confronto a livello di colonna.

Un database SQL server

- E' proprietà di un singolo utente ,ma può contenere oggetti di proprietà di altri utenti.
- Dispone di tabelle di sistema per catalogare le definizioni del database.
- Gestisce un proprio insieme di profili utente e assegnazione di diritti di accesso.
- Dispone di un log di registrazione delle operazioni e gestisce le transazioni.
- Può espandersi su più unita disco e su più macchine.

Presenta vari database di sistema:

- **Database master**

Tabelle di sistema con i risultati del processo di installazione del sistema e di tutti i database creati di seguito.

- **Database model**

Database modello ,da cui si creano tutti i nuovi database per copia.

- **Database tempdb**

Area di lavoro che viene creata ad ogni avvio.

- **Database msdb**

Database di supporto per le operazioni periodiche.

- **Database pubs**

Database esempio usato come riferimento nella documentazione.

- **Database Northwind**

Database esempio.

Un database SQL server consiste di tre tipi di file :

- **MDF :** file di dati primario ,che elenca tutti gli elementi del database ed i dati memorizzati.
- **NDF :** file di dati secondari (può non essere presente).
- **LDF :** file di log delle operazioni.

E' possibile creare più file MDF / NDF per distribuire il contenuto del database su più dispositivi fisici e per controllare le dimensione degli stessi file di dati.

I file di log registrano i dati relativi alle transazioni che interessano il database. Supportano inoltre:

- Ripristino (recovery) di transazioni per richiesta di ROLLBACK.
- Recupero di transazioni incomplete all'avvio del DBMS.
- Ri-esecuzione di transazioni dopo il ripristino del DBMS in seguito ad un problema di sistema.

La gestione del log adotta una politica di WRITE – AHEAD : nessun dato viene scritto nel database se prima non è stata completata la scrittura delle informazioni di transazione nel file di log. Il file di log viene visto dal sistema come un'unica entità logica anche se è distribuito su più file fisici. Il file di log è riempito come un file sequenziale ,come wrap –around dei contenuti dalla fine a riprendere all' inizio.

2.2.2 Portabilità di Microsoft SQL Server 2000 e compatibilità con SQL92

Microsoft SQL Server può essere utilizzato solo con un sistema operativo Windows (NT/2000/XP) e usa una variante del linguaggio SQL92 standard chiamata T-SQL (Transact-SQL) che è stato arricchito da funzioni proprietarie per gestire le transazioni ,le stored procedures e per facilitare certe operazioni. Esempio la keyword TOP restringe il risultato di una query sui primi **x** elementi ritornati.

2.2.3 Punti di forza di Microsoft SQL Server 2000

Microsoft SQL Server rappresenta un RDBMS ad alte prestazioni progettato per gestire altissimi volumi di operazioni transazionali in ambiente multiutente. La gestione di un database passa principalmente attraverso funzioni di:

- Amministrazione del database: creazione di tabelle, operazioni di manutenzione dello stesso e backup.

- Controllo degli accessi, ossia gestione di utenti e di permessi sia in termini di operazioni (possibilità di creare oggetti o di eseguire procedure) sia in termini di sicurezza dei dati (possibilità di accedere o meno a dati riservati).

- Controllo delle operazioni sui dati attraverso la creazione di procedure utilizzando il T-SQL.

Queste operazioni sono facilmente gestibili all'interno di Enterprise Manager ,che è lo strumento grafico che consente l'amministrazione di Microsoft SQL Server 2000.
Microsoft SQL server 2000 offre un interfaccia grafica molto intuitiva anche per gli utenti meno esperti. Esistono numerosi strumenti quali : query analyzer per

l’esecuzione delle query, importazione ed esportazione dati e servizi di trasformazione dei dati .
Inoltre SQL Server 2000 supporta la clausola FOR XML che restituisce i risultati di istruzioni SQL sotto forma di documento XML(**eXtensible Markup Language** è lo standard Internet emergente per i dati. I documenti XML possono essere elaborati rapidamente in HTML (Hypertext Markup Language), il più importante linguaggio per la visualizzazione delle pagine Web). SQL Server 2000 supporta inoltre le query XPath di applicazioni per Internet e Intranet. I documenti XML possono essere aggiunti ai database di SQL Server e la clausola OPENXML consente di esporre dati di un documento XML sotto forma di set di risultati relazionali.
Tramite SQL Server 2000 è quindi possibile interagire completamente con i dati XML, questo rende SQL Server molto flessibile e facilita molto il lavoro di chi si occupa di database orientati ad applicazioni Web. Con SQL Server 2000 è possibile effettuare analisi dei dati attraverso lo strumento OLAP(On Line Analitycal Processing) che è un insieme di tecniche software per analizzare velocemente grandi quantità di dati, anche in modo complesso.

2.3 Installazione

2.3.1 Installazione di MySQL 5.0 e configurazione

MySQL 5.0 si può scaricare dal sito ufficiale www.mysql.com. Una volta scelta la versione ci troviamo davanti ad una lista di file scaricabili che si differenziano in base alle piattaforme supportate e al tipo di file che ci interessa (file binari eseguibili o i sorgenti da compilare). MySQL può girare su qualsiasi sistema operativo Windows[1] a 32 bit (quindi da Windows 95 in poi); per poterlo installare sulla nostra macchina ci basta il supporto per TCP/IP e circa **200 MB** di spazio disponibile (può essere eseguita anche con 32 MB RAM). La situazione ideale tuttavia è quella di un sistema da Windows 2000 in poi (oppure Windows NT), che ci permette di eseguire MySQL come un servizio.
La scelta del pacchetto da installare è piuttosto semplice:

- **Windows Essentials**: come suggerisce il nome, contiene i componenti essenziali per MySQL ed è fornito con l'installatore Windows.
- **Windows**: è il pacchetto completo, comprensivo di componenti opzionali quali l'embedded server e la benchmark suite.
- **Without installer**: contiene gli stessi file del pacchetto completo, ma non ha l'installatore Windows.

Effettuato il download, per eseguire l'installazione basta lanciare il file .msi, che chiama in causa l'installatore di Windows.Verrà quindi avviato l'**Installation**

Wizard di MySQL, che ci guiderà attraverso i seguenti passi:

- Per prima cosa ci viene chiesto che tipo di installazione vogliamo eseguire. La scelta è tra: tipycal, complete e costum.
- Una volta effettuata la scelta viene effettuata l'installazione vera e propria.
- L'installazione è già terminata, ma l'ultima schermata dell'Installation Wizard ci chiede se vogliamo lanciare il Configuration Wizard. Lasciamo selezionata la checkbox per configurare immediatamente il nostro server MySQL.
- Abbiamo ora la possibilità di scegliere tra una configurazione standard o dettagliata.
- Ci viene chiesto se vogliamo eseguire MySQL come servizio, se vogliamo lanciarlo automaticamente e se vogliamo includere la directory 'bin' nel path di Windows.
- L'ultima schermata ci consente di impostare la password di root attraverso la quale ci sarà possibile di amministrare il nostro server.

Quindi l'installazione è molto semplice.

[1] Per praticità abbiamo visto l'installazione su una macchina con sistema operativo Windows.

2.3.2 Installazione di Microsoft SQL Server 2000 e configurazione

I requisiti di sistema per l'installazione di **Microsoft SQL Server 2000** sono:
• Processore Intel o compatibile;
• Pentium 166 o superiore;
• 32 MB RAM per la versioni desktop (64 MB raccomandati);
• 64 MB RAM per le restanti versioni (128 MB o più sono consigliabili);

Lo spazio su disco necessario:
• 270 MB installazione completa
• 250 MB installazione tipica
• 95 MB installazione minima
• 44 MB installazione desktop

Nel caso di componenti opzionali come Analysis service e English Query sono richiesti rispettivamente da 50 a 130 MB (tipica e completa) e 80 MB.
Per concludere, è richiesta la presenza di Internet Explorer 5.0 per l'esecuzione della management console e per visualizzare i manuali in linea prodotti nel formato HTML.

Il prodotto è disponibile in tre differenti versioni:

• Standard Edition: è la soluzione per aziende di piccole-medie dimensioni che non necessitano delle funzionalità di analisi avanzate di SQL Server 2000 Enterprise Edition.

• Workgroup Edition: include un insieme completo di strumenti per la gestione

della maggior parte delle funzionalità dell'edizione Standard, ma è ottimizzata per l'uso di utenti singoli.

• Enterprise Edition: è la versione completa di SQL Server 2000. Offre caratteristiche di scalabilità ed affidabilità oltre alle funzionalità di analisi avanzate (OLAP: On Line Analitycal Processing).

Per tutte e tre le versioni la procedura di installazione di Microsoft SQL Server 2000 consiste nell'esecuzione di un wizard , unica avvertenza: è necessario utilizzare un account con privilegi amministrativi e l'installazione può essere effettuata solo su macchine in cui è installato un sistema operativo Windows (Windows NT/2000/XP). Scegliere dal menu principale Componenti di SQL Server 2000 e Installa Server di Database per avviare l'installazione guidata. Il programma di setup consente anche l'installazione su computer remoti selezionando l'apposita opzione.
Per iniziare la fase di installazione vera e propria selezionare Crea una nuova istanza di SQL Server o installa gli strumenti client, dopo aver inserito il cd-key continuare scegliendo Strumenti client e server. Durante questa fase selezionare Istanza predefinita per la prima installazione: infatti da questa versione è supportata la creazione di istanze multiple che si comportano come database server separati fisicamente, quindi con impostazioni e amministrazioni distinte, fondamentali in ambienti di hosting. L'installazione continua dopo aver selezionato il Tipo di installazione (Tipica o Minima) chiedendo di inserire l'account di servizio ossia l'utente che può avere accesso ai due servizi installati (SQL Server: il motore del database vero e proprio e Agent SQL Server responsabile delle operazioni schedulate all'interno del database) e la modalità di autenticazione ovvero in che modo SQL Server autentica gli utenti per avere accesso ai dati.
In SQL Server è possibile attivare due modalità diverse di autenticazione:

• Modalità di autenticazione di Windows: consente di connettersi utilizzando il proprio account utente. In questo caso SQL Server sfrutta il meccanismo di protezione di Windows e questo consente di avere gli stessi username e password sia per l'ambiente di rete che per il database server.

• Modalità mista (autenticazione di Windows e di SQL Server): se un utente si connette con username non verificato da un sistema Windows l'autenticazione viene eseguita da SQL Server che verifica l'esistenza di tale account nei propri utenti e ne verifica la password. Nel caso che l'utente non abbia le credenziali, l'accesso è negato.

L'installazione termina scegliendo la modalità di licenza. Le opzioni disponibili sono due:

• Processor License: richiede una singola licenza per ciascun processore installato sulla macchina su cui è eseguito SQL Server .

• Server/Per-Seat Client Access License (CAL): richiede una licenza per la macchina su cui è installato il prodotto e una licenza per ciascun client che vi accede (CAL).

L'installazione presenta media difficoltà.

2.4 Costi

2.4.1 Costi di MySQL 5.0

MySQL è un DBMS relazionale open source (scaricabile da internet). Il codice di MySQL è di proprietà della omonima società MySQL AB, viene distribuito con la licenza GNU GPL (GNU General Public License, che è una licenza per software libero) oltre che con una licenza commerciale. Una buona parte del codice del client è licenziato con la GNU LGPL e può dunque essere utilizzato per applicazioni commerciali. Con la licenza GPL, MySQL è gratuito. Gli utenti possono scaricare il software ,modificarlo e distribuirlo. Comunque in questo caso gli utenti devono rendere il codice modificato "open source". MySQL AB offre una licenza commerciale per organizzazioni che non vogliono rendere le loro applicazioni sviluppate con MySQL "open source". MySQL non è un software open source tipico poiché tutto il progetto (codice relativo) è di proprietà della società MySQL AB. Un recente articolo di Computer World ("MySQL Breaks Into the Data Center") ha spiegato come MySQL sia diventato il database open source più famoso a livello mondiale, e perché le grandi aziende intenzionate a ridurre i costi di gestione lo stiano utilizzando per ottimizzare ulteriormente la propria infrastruttura IT.

MySQL riduce il costo totale di gestione[2] (Total Cost of Ownership - TCO) del software per database attraverso:

- La riduzione dei costi per le licenze database di oltre il 90%
- La riduzione dei tempi di fermo del sistema del 60%
- La riduzione delle spese per l'hardware del 70%
- La riduzione dei costi di amministrazione, ingegnerizzazione e supporto fino al 50%

I costi delle licenze sono:

- Ricerche senza scopo di lucro: Gratuito
- Uso senza modifiche della sorgente: Gratuito
- Uso con modifica della sorgente ma con la restrizione di rendere open source l' applicazione sviluppata : Gratuito
- Con modifiche della sorgente : i prezzi variano in base all' edition dell' Enterprise (Basic, Silver, Gold, Platinum) da 595$ a 4995$ /server/anno. Non ci sono restrizioni sul numero di processori e sul numero di connessioni.

[2] I dati sono stati rilevati dal sito ufficiale di MySQL: www.mysql.com

2.4.2 Costi di Microsoft SQL Server 2000

Microsoft SQL Server è un RDBMS commerciale prodotto da Microsoft.
Ci sono vari tipi di licenze **:**

- Licenza server: una licenza per ogni server su cui viene installato il prodotto server ed ogni utente o dispositivo che accede ai dati deve essere provvisto di una CAL (client access license).

- Licenza processore: una licenza per ogni processore fisico accessibile dal sistema operativo su cui viene installato SQL Server.

I prezzi[3] variano a seconda del tipo di licenza e della edition (workgroup, standard, enterprise) da 730$ a 24000$ /server e da 140$ a 160$ per una CAL (licenza per cliente).Con la licenza server per ciascuna edition c'è un numero limite di CPU per server. Per esempio con l' enterprise edition il limte è di 32 CPU,per la standard edition è 4 CPU e per la workgroup edition è di 2 CPU. Invece i prezzi delle licenze processore variano da 60$ a 810$ /CPU. Una versione ridotta di Microsoft SQL Server 2000 chiamata MSDE (Microsoft SQL Server Desktop Engine) viene distribuita con prodotti come Visual Studio, Microsoft Access ed altri. MSDE presenta alcune restrizioni: supporta solo database con dimensioni massime di 2GB, non ha strumenti per essere amministrato ed è programmato per ridurre le prestazioni quando si superano gli 8 accessi concorrenti.

[3] I dati sono stati rilevati dal sito ufficiale di Microsoft :www.microsoft.com

2.5 Sicurezza

Le informazioni sono uno dei beni più preziosi della nostra società. La sicurezza è un aspetto molto importante da valutare in un DBMS. Tutti i più moderni RDBMS adottano una architettura di sicurezza basata su tre differenti livelli di protezione:

- **autenticazione**: questa è la fase di verifica dell'identità dell'utente attraverso una password.
- **autorizzazione**: è la fase che segue l'autenticazione nella quale il sistema deve determinare a quali risorse l'utente può avere accesso e con quali modalità operative.
- **auditing**: questa fase si contraddistingue per l'adozione di mezzi idonei ad identificare e riconoscere possibili abusi oltre che ad assicurare l'integrità delle informazioni.

Quando si parla di sicurezza dei database si intende principalmente fare riferimento alla necessità di garantire l'accessibilità dei dati. Qualunque attività diretta ad implementare e/o rafforzare la sicurezza di questi sistemi presuppone il rispetto dei seguenti principi:

- la definizione di pratiche e procedure di sicurezza chiare e di facile attuazione.
- la predisposizione di livelli di sicurezza multipli .
- l'applicazione costante del principio "dei privilegi minori".

Senza la presenza di "strati" di sicurezza multipli diventa estremamente facile connettersi ad un database sfruttando la presenza di account noti e con password di default oppure bypassando la sicurezza del sistema operativo e dei vari dispositivi che compongono l'infrastruttura di rete (firewall, router, ecc...). Analogamente il principio dei privilegi minori, in virtù del quale ogni utente deve avere assegnati soltanto i privilegi strettamente indispensabili al compimento delle sue attività e questo evita i danni che possono derivare da un uso improprio oppure da azioni compiute in modo accidentale. Aspetti molto importanti sono:

- **segretezza e confidenzialità**: i dati devono poter essere consultati e/o modificati soltanto da parte di chi sia debitamente autorizzato.

- **integrità ed autenticità**: i dati non devono poter essere manipolati .
- **accessibilità:** i dati devono essere sempre disponibili eventualmente anche attraverso il loro immediato ripristino.

2.5.1 Sicurezza di MySQL 5.0

MySQL è un robusto database relazionale multipiattaforma .

La fase di autenticazione

MySQL adotta un modello di sicurezza articolato secondo due livelli:

- server
- database

Questa diversificazione degli strati di sicurezza si traduce in un continuo processo di controllo degli accessi, che di regola viene esplicato in due fasi distinte:

1. Verifica da parte del processo server dei privilegi necessari all'utente per la connessione.
2. Verifica, per ogni richiesta utente, del possesso dei privilegi necessari a soddisfarla.

Il controllo degli accessi viene eseguito tramite le informazioni memorizzate all'interno di alcune tabelle di sistema dette anche **grant tables**:

Tabella	Informazioni
user	Determina a quali utenti è consentita l'accesso al server
db	Memorizza i database ai quali un utente può accedere con l'indicazione dell'host da cui è possibile l'accesso e le operazioni che egli può eseguire
host	E' utilizzata per specificare a quali database e da quali host un utente può connettersi ed in tal senso rappresenta una estensione della tabella db
tables_priv	Indica i privilegi di ogni utente a livello di tabella
columns_priv	Indica i privilegi di ogni utente a livello di colonne

Nelle tabelle **user (**per rendere più sicuro il sistema bisogna rimuovere l'anonimous user dalla tabella user**)**, **db** ed **host** i campi dei privilegi possono assumere i valori

'Y' (yes) e 'N' (no) mentre per le tabelle **table_priv** e **column_priv** questi campi possono assumere i valori mostrati nella seguente tabella:

Tabella	Colonna	Valori
tables_priv	Table_priv	'Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter'
tables_priv	Column_priv	'Select', 'Insert', 'Update', 'References'
columns_priv	Column_priv	'Select', 'Insert', 'Update', 'References'

Lo scopo della fase di autenticazione degli utenti consiste nella verifica della correttezza delle seguenti informazioni:

- l'host dal quale viene inoltrata la richiesta di connessione.
- il nome e la password dell'utente MySQL.

figura 1: Fase di autenticazione in MySQL 5.0

L'utilizzo del nome dell'host come credenziale di autenticazione insieme con la coppia nome utente e password trova una sua giustificazione nell'esigenza avvertita di distinguere ulteriormente un utente a seconda della postazione dalla quale egli si connette.
Come già detto il processo di autenticazione sfrutta le tabelle host,db e user, in particolare i valori contenuti nei campi user ed host della tabella user sono molto importanti per la sicurezza del database.

Nella seguente tabella riepilogativa viene mostrato a titolo semplificativo come i valori assegnati a queste due colonne siano determinanti ai fini del completamento del processo di connessione a MySQL.

Host	User	Risultato
Myhost.loc.it	Eni	L'utente "Eni" può connettersi al server myhost.loc.it
Myhost.loc.it		Tutti gli utenti possono connettersi al server myhost.loc.it
%	Eni	Eni può connettersi da qualunque host
%		Tutti gli utenti possono connettersi da qualunque host
%.loc.it	Eni	Eni può connettersi da qualunque host del dominio loc.it

Una volta completata con successo la fase di autenticazione il sistema verifica per ogni ulteriore operazione richiesta il possesso dei privilegi necessari a soddisfarla. Mentre per le operazioni di tipo amministrativo (es: SHUTDOWN, RELOAD ecc...) la verifica coinvolge la sola tabella user, per tutte le altre richieste relative alle operazioni sui dati (ad. esempio INSERT, UPDATE ecc..) il processo server verifica innanzitutto la presenza di privilegi amministrativi; se questi ultimi sono trovati allora l'accesso è consentito mentre, in caso contrario, l'esistenza del privilegio specifico viene accertata secondo le modalità di seguito indicate:

1. il server estrae le informazioni dalla tabella db controllando la correttezza delle informazioni contenute nei campi host, user e db.
2. in caso di esito negativo l'accesso è negato mentre in caso di esito positivo, se la colonna host non è nulla, questo record definisce un privilegio specifico del database.

In particolare quando viene trovata una entry nella tabella specificata, i privilegi sono calcolati come l'intersezione dei privilegi impostati a 'Y' presenti nelle tabelle host e db. In questo modo diventa possibile definire dei privilegi nella tabella db e restringerli per host usando la tabella **host**. Riassumendo l'intero processo si può dire che l'accesso è concesso se sussiste una delle seguenti condizioni:

1. l'utente possiede dei privilegi globali.
2. l'utente possiede dei privilegi specifici determinati per mezzo delle tabelle host e db.
3. l'utente possiede privilegi relativi a tabelle.
4. l'utente possiede privilegi relativi a colonne.

Privilegi

I privilegi sono speciali diritti concessi ad uno o più utenti che implicano la facoltà di eseguire determinati comandi o di compiere determinate azioni su specifici oggetti del database.

MySQL mette a disposizione quattro tipologie di privilegi che si differenziano a seconda del contesto di validità che assumono:

1. **privilegi a livello globale**: riguardano tutti i database del server e vengono memorizzati nella tabella **user.**
2. **privilegi a livello di database**: riguardano tutti gli oggetti di un database e sono memorizzati nella tabella **db**.
3. **privilegi a livello di tabella**: sono relativi ad una tabella di un database e vengono memorizzati nella tabella **tables_priv**.
4. **privilegi a livello di colonna**: riguardano una singola colonna di una tabella e sono memorizzati nella tabella **columns_priv**.

I privilegi possono essere concessi e revocati mediante i comandi GRANT e REVOKE. Inoltre l'assegnazione o la revoca possono anche avvenire inserendo direttamente nelle relative tabelle un record oppure cancellando un record già esistente. La differenza fondamentale che intercorre tra i due tipi di comandi è che nel primo caso l'operazione produce degli effetti immediati laddove, nel secondo caso, la produzione degli effetti è subordinata all'esecuzione di un ulteriore comando quale **mysqladmin flush-privileges** (oppure **mysqladmin reload**).

Inoltre in MySQL esistono, per comodità di amministrazione, degli appositi privilegi che racchiudono in se altri privilegi come ad esempio ALL_PRIVILEGES la cui assegnazione implica il riconoscimento all'utente dei diritti di SELECT, INSERT, UPDATE, INDEX, ALTER, CREATE, DROP, DELETE e GRANT. Un altra interessante caratteristica di MySQL è che attraverso il sistema dei privilegi consente di assegnare agli utenti i seguenti profili di utilizzo:

- numero di interrogazioni sul database per ora
- numero di aggiornamenti sulle tabelle per ora
- numero di connessioni per ora

Sfruttando tale meccanismo risulta possibile circoscrivere i possibili tentativi di abuso delle risorse del database server che possono anche essere sintomo di attacchi tendenti a saturare le capacità di gestione dei dati da parte del sistema .

Inoltre MySQL 5.0:

- fornisce un supporto per OpenSSL (Secure Sockets Layer protocollo che utilizza la crittografia per fornire sicurezza nelle comunicazioni su internet) per realizzare comunicazioni cifrate.
- consente di eseguire periodicamente le opportune procedure di backup dei dati.
- consente l’utilizzo del Lock a livello di riga (consente l’accesso esclusivo di un utente in lettura/scrittura su una riga).
- consente l’utilizzo delle transazioni ACID (Atomicità, Consistenza, Isolamento, Durabilità) anche se limitatamente alle InnoDB.

atomicità: la transazione non è indivisibile nella sua esecuzione .

consistenza: dopo l'esecuzione di una query il database si deve trovare in uno stato consistente ,ovvero non deve violare eventuali vincoli di integrità.

isolamento: ogni transazione deve essere eseguita in modo isolato e indipendentemente dalle altre transazioni, l'eventuale fallimento di una transazione non deve interferire con le altre transazioni in esecuzione.

durabilità: detta anche **persistenza**, si riferisce al fatto che una volta che una transazione ha richiesto un COMMIT (comando per confermare la transazione), i cambiamenti apportati non dovranno essere più persi. Per evitare che nel lasso di tempo fra il momento in cui la base di dati si impegna a scrivere le modifiche e quello in cui li scrive effettivamente si verifichino perdite di dati dovuti a malfunzionamenti, vengono tenuti dei registri di log, dove sono annotate tutte le operazioni sul DB.

- può essere utilizzato in configurazione di cluster con cui i dati vengono partizionati e distribuiti su più server permettendo di disporre continuamente delle informazione anche in caso di guasto di un server (in questo caso vengono usati i storage engine NDB cluster).

Ci sono alcune utili opzioni che possono essere impiegate nell'esecuzione del servizio MySQL e che ne influenzano le caratteristiche:

-safe-show-database: con questa opzione il comando SHOW DATABASES mostra soltanto i database in relazione ai quali l'utente possiede almeno un privilegio.
-safe-user-create: questa opzione impedisce la possibilità di creare un nuovo utente tramite l'istruzione GRANT nell'ipotesi in cui non sussista il privilegio di INSERT per la tabella user.
-skip-name-resolve: con questa opzione viene forzato l'inserimento di indirizzi IP oppure del localhost in tutte le colonne host delle tabelle dei grant.
-skip-networking: impedisce l'instaurazione di connessioni TCP/IP.
-skip-show-database: impedisce l'esecuzione del comando SHOW DATABASES.

2.5.2 Sicurezza di Microsoft SQL Server 2000

Come la maggior parte dei moderni RDBMS anche Microsoft SQL Server 2000 adotta un modello di sicurezza contraddistinto dalla presenza di tre differenti livelli di protezione: autenticazione, autorizzazione ed auditing. SQL Server 2000 è stato esaminato dal Governo degli Stati Uniti, l'applicazione è risultata conforme ai requisiti per la certificazione di sicurezza C2 (praticamente ha un livello di sicurezza adatto per le applicazioni del governo).

La fase di Autenticazione

In SQL Server 2000, prima di poter accedere alle risorse , è necessario passare attraverso una fase di autenticazione che coinvolge due livelli differenti: il server ed il database.
Il passaggio attraverso queste due fasi distinte viene reso possibile tramite il ricorso ai login ed agli account utente. Un login può essere sia esterno, cioè proprio del sistema operativo Windows, sia interno vale a dire memorizzato direttamente nel database.
Gli account utente invece sono necessari per consentire l'accesso alle risorse del database. Quando viene creato un account utente all'interno di un database esso viene associato ad un login SQL Server oppure ad un utente/gruppo del sistema operativo. Fondamentalmente SQL Server 2000 prevede due modalità distinte di autenticazione:

- **basata sul sistema operativo (Windows Authentication Mode)**: è la modalità di default; quando viene utilizzata non occorre fornire un nome di login ed una password per connettersi al server SQL poiché l'autenticazione viene mediata da Windows attraverso un account e/o gruppo di account validi all'interno del sistema stesso. In questi casi l'amministratore del database deve semplicemente specificare nella fase di configurazione gli utenti/gruppi che hanno il permesso di autenticarsi ed, in tal modo, garantisce indirettamente agli stessi l'accesso al sistema nel quale risiede il processo del server di database.
- **modalità mista (Mixed Mode)**: questo tipo di autenticazione viene così chiamata poiché essa può essere basata sul sistema operativo oppure sul server di database. In quest' ultima ipotesi l'amministratore del database deve prima creare degli account di login che non hanno nulla a che vedere con gli utenti/gruppi del sistema operativo e le cui password vengono memorizzate all'interno del database master (nella tabella **sysxlogins**). Tra le due citate modalità quella da preferire è naturalmente quella basata su Windows poiché essa oltre ad essere strettamente integrata nell'ambiente operativo impedisce il transito sulla rete delle coppie login/password necessarie affinché l'autenticazione abbia luogo.

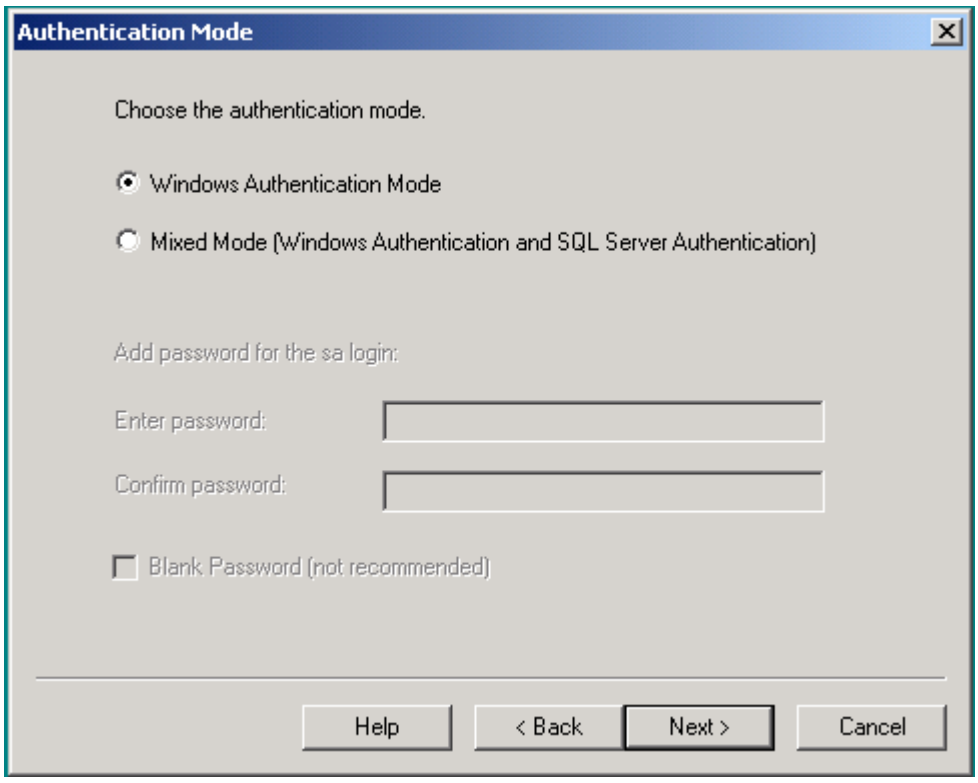

figura 2: Scelta modalità di autenticazione in Microsoft SQL Server 2000

Quando si installa SQL Server in un computer Windows NT 4.0 o Windows 2000 che utilizza il file system NTFS, le directory di destinazione dell'installazione vengono protette in modo da limitare l'accesso ai soli account impostati per i servizi di SQL Server e al gruppo di amministratori predefinito. Per impostazione predefinita, l'applicazione viene installata nella directory C:\Programmi\Microsoft SQL Server\MSSql. Durante l'installazione di SQL Server vengono protette in questo modo anche le chiavi di registro di SQL Server, HKLM\Software\Microsoft\MSSQLServer
o HKLM\Software\Microsoft\Microsoft SQL Server\MSSQL$*NomeIstanza*, per le istanze denominate, e le relative sottochiavi. Per rendere più sicuro il sistema si consiglia di disabilitare l'account guest a livello di sistema operativo ed eliminarlo da tutti i database di produzione mediante l'Enterprise Manager oppure con la procedura registrata **sp_dropuser.**

Accesso alle risorse

Il completamento della fase di autenticazione da parte di un utente non comporta automaticamente l'accesso a tutti i database di SQL Server 2000 e soprattutto, non implica che egli possa avere accesso a tutti gli oggetti anche all'interno di uno stesso database.
L'uso degli oggetti del database da parte dei vari utenti è regolamentato attraverso le autorizzazioni che generalmente vengono espresse e concesse sotto forma di ruoli e permessi.

I ruoli

Il concetto di ruolo è molto simile, per non dire identico, a quello di gruppo di utenti del sistema operativo. In tale contesto un ruolo non è altro che un astrazione concettuale utilizzata per raggruppare un insieme di utenti all'interno di una stessa unità logica alla quale possono essere poi conferiti o negati determinati privilegi.

SQL Server 2000 prevede una serie di ruoli differenti:

- **public role**: questo ruolo possiede i permessi di default per gli utenti di un database, esiste in ogni database di SQL Server compresi quelli di sistema (master, msdb, tempdb e model) e non può essere eliminato.
- **fixed server roles**: sono ruoli predefiniti che hanno una valenza globale e pertanto esistono al di fuori del contesto dei singoli database. Di regola essi vengono associati ai login in modo da garantire a questi ultimi i permessi amministrativi che essi implicano. Questi ruoli non possono essere concessi singolarmente agli account utente, non sono modificabili e non è possibile crearne altri dello stesso tipo.

La tabella seguente riporta una lista di questi ruoli insieme con l'indicazione delle loro caratteristiche:

Ruolo	Descrizione
sysadmin	E' in grado di compiere qualsiasi attività.
serveradmin	Amministra le opzioni di configurazione del server e può eseguire lo shutdown dello stesso.
setupadmin	Gestisce ed amministra i server collegati e le procedure di startup.
securityadmin	Gestisce le impostazioni di sicurezza a livello di server, inclusi quelli collegati, e possiede il permesso CREATE DATABASE. Inoltre è in grado di eseguire il reset delle password per i login di autenticazione.
processadmin	E' in grado di terminare i processi di SQL Server.
dbcreator	Può creare, modificare e ripristinare qualsiasi database di SQL Server.
diskadmin	Gestisce i file sul disco.
bulkadmin	Permette agli utenti diversi dagli amministratori di sistema di eseguire le istruzioni **bulkadmin.** Gli utenti di questo gruppo possono caricare dati da qualsiasi file disponibile in rete e da qualsiasi computer che esegue il server ed è accessibile mediante l’account del servizio di SQL Server.

- **fixed database roles**: anche questi sono ruoli predefiniti esistenti tuttavia nel contesto dei singoli database. Generalmente essi vengono concessi agli account utente ma i relativi permessi non possono essere modificati in alcun modo.

La tabella seguente riporta una lista di questi ruoli insieme con l'indicazione delle loro caratteristiche:

Ruolo	Descrizione
db_owner	Esegue tutte le attività di manutenzione e

	configurazione del database.
db_accessadmin	Aggiunge o rimuove le autorizzazioni di accesso per gli utenti ed i gruppi del sistema operativo e per i login di SQL Server.
db_datareader	E' in grado di leggere i dati da tutte le tabelle degli utenti.
db_datawriter	E' in grado di aggiungere, cancellare e modificare i dati in tutte le tabelle degli utenti
db_ddladmin	Può eseguire qualsiasi comando DDL (Data Definition Language) in un database.
db_securityadmin	Modifica l'appartenenza ai ruoli e gestisce i permessi.
db_backupoperator	Esegue il backup del database.
db_denydatareader	E' un ruolo di tipo restrittivo nel senso che esso non può leggere i dati di qualsiasi tabella degli utenti del database.
db_denydatawriter	Anche questo è un ruolo di tipo restrittivo nel senso che non può aggiungere, modificare o cancellare dati in nessuna delle tabelle degli utenti del database.

- **user defined database roles**: si tratta di ruoli che possono essere creati a livello di singolo database per raggruppare una serie di permessi che caratterizzano l'attività di un gruppo di utenti.
- **application roles**: consentono di restringere l'accesso ai dati sulla base dell'applicazione che ogni singolo utente sta usando in un determinato momento. In sostanza tali ruoli permettono di delegare all'applicazione la responsabilità dell'autenticazione dell'utente. Infatti tutto ciò che l'amministratore deve fare è creare il ruolo applicativo ed assegnare allo stesso i singoli permessi che lo caratterizzano mentre sarà l'applicazione utilizzata sul lato client a dover attivare a run-time il ruolo attraverso l'invocazione di una procedura registrata (**sp_setapprole**) così da ottenere tutti i permessi precedentemente associati al ruolo in fase di creazione.

I ruoli applicativi possiedono le seguenti caratteristiche:

- a causa del meccanismo dinamico con il quale vengono attivati questi ruoli non possono essere attribuiti ai singoli utenti.
- essi prevalgono sui permessi standard nel senso che, per effetto della loro attivazione, l'applicazione perde tutti gli altri permessi normalmente associati con il login/utente adoperato durante la connessione al server.
- essi esistono soltanto nel contesto di un singolo database per cui se l'applicazione deve iniziare una transazione che interessa un altro database occorre che in quest' ultimo sia attivato un account utente di tipo guest e che esso abbia le necessarie autorizzazioni per la lettura/scrittura di dati.
- gli utenti che utilizzano i ruolo applicativi sono soggetti ad attività di auditing all'interno del database.

I permessi

I permessi possono essere concessi ad utenti e ruoli del database oppure ad utenti/gruppi del sistema operativo ma mai direttamente a login di SQL Server. Attraverso i permessi vengono specificate le istruzioni ed i comandi SQL che un utente può eseguire nonché le modalità con le quali egli può avere accesso agli oggetti del database.
A seconda del contesto i permessi possono essere di tipo **GRANT**, **REVOKE** e **DENY**: un permesso di tipo GRANT assume il valore di concessione positiva mentre REVOKE e DENY hanno valore di restrizioni.
È comunque importante sottolineare la differenza fondamentale che intercorre tra REVOKE e DENY: l'effetto di quest' ultimo è quello di annullare l'esecuzione di un istruzioni o l'accesso ad oggetti anche quando esiste una GRANT**,** dal momento che DENY ha la precedenza su tutte le altre. Al contrario l'istruzione REVOKE produce come effetto l'inefficacia di una precedente GRANT o DENY. Quando un utente si connette a SQL Server viene istanziata e mantenuta in memoria per tutta la durata della sessione una particolare struttura detta PSS (Process Status Structure) contenente il SID (Security Identifier) dell'utente e del gruppo di appartenenza nonché una serie di informazioni di sicurezza e di stato.
Durante la fase iniziale di accesso al database il server effettua una verifica leggendo la tabella di sistema **sysusers** per stabilire se all'utente sia stato negato l'accesso al database direttamente oppure per via della sua condizione di appartenenza ad un gruppo.
Una volta accertato che l'utente abbia il diritto di accesso, il sistema esamina la tabella **sysmembers** per consentire l'attivazione di tutti i ruoli concessi.
Terminata questa prima fase, per ogni successiva richiesta di accesso a tabelle, viste ed altri oggetti, viene sempre valutato il possesso dei necessari permessi tramite una lettura della tabella di sistema **syspermissions**.
Se a seguito di tale controllo il sistema rinviene l'esistenza di una **deny** allora all'utente viene impedito l'accesso mentre, in caso contrario, l'accesso può aver luogo normalmente purché nella tabella indicata esista un record che lo garantisca esplicitamente.
In ogni caso le informazioni relative al possesso dei permessi vengono collocate in una memoria cache così da velocizzare tutte le verifiche derivanti da eventuali richieste successive relative ai medesimi oggetti.
Inoltre, per evitare situazioni di inconsistenza dei dati causate dalla sopravvenuta concessione di nuovi permessi, queste stesse informazioni sono aggiornate mediante un meccanismo basato sull'incremento e la verifica di contatori interni.
SQL Server 2000 consente di utilizzare il log eventi di Windows per controllare l’accesso al server. Per configurare il livello di controllo è possibile utilizzare SQL Server Enterprise Manager o la stored procedure estesa **xp_loginconfig.** Le impostazioni di controllo disponibili sono:

- **None**: non vengono registrate informazioni di controllo.
- **Success**: vengono registrati solo gli accessi con esito positivo .
- **Failure**: vengono registrati solo gli accessi con esito negativo.
- **All**: vengono registrati tutti gli accessi, indipendentemente dall’esito.

E’ anche disponibile uno strumento estremamente potente, SQL Profiler che di fatto consente di analizzare molti eventi ,inclusi i seguenti:

- le attività degli utenti finali come ad esempio l'esecuzione di comandi SQL oppure l'attivazione di ruoli applicativi.
- le attività dei database administrator quali l'esecuzione di istruzioni SQL di creazione delle strutture e la configurazione dei database o del server.
- gli eventi di sicurezza come ad esempio quelli concernenti l'assegnazione di GRANT, REVOKE e DENY oppure l'assegnazione, la rimozione e la configurazione di ruoli.
- gli eventi derivanti dall'uso delle utilità di backup/restore oppure dall'esecuzione di comandi di bulk insert.
- gli eventi propri del server come startup/shutdown.
- gli eventi di controllo: aggiunta, modifica e interruzione del controllo.

Il monitoraggio di questi eventi avviene in real-time con visualizzazione immediata sullo schermo oppure con la possibilità di salvare l'output in un file di testo oppure all'interno di una tabella di database.

Inoltre SQL Server 2000 supporta:

- la crittografia dei dati e del traffico di rete tra i sistemi client e server di una rete attraverso il protocollo SSL/TSL (Transport Security Layer) .
- la restrizione dell'accesso diretto degli utenti alle tabelle facendo in modo che i dati vengano letti e scritti mediante l'utilizzo di viste e procedure registrate.
- la protezione, dove necessario del codice delle procedure registrate, dei triggers, delle viste e delle funzioni definite dall'utente adottando nelle apposite istruzioni SQL la clausola **WITH ENCRYPTION**.
- la possibilità di criptare i dati a livello di tabelle facendo uso di librerie di terze parti oppure sviluppandone di proprie tramite le CryptoAPI messe a disposizione dal sistema operativo nel caso di database contenenti informazioni sensibili.
- la criptazione del filesystem EFS (Encrypted File System) fornite dal sistema operativo Windows 2000 per proteggere le cartelle dove risiedono i file di dati e di log della istanza di SQL Server.
- l’esecuzione periodica di opportune procedure di backup dei dati sfruttando la possibilità di fornire una password per il backup.
- le transazioni ACID.
- lock a livello di riga.

Con SQL Server 2000, gli amministratori di database dispongono di tutte le funzionalità necessarie per configurare e gestire server di database sicuri e perfettamente integrati con il sistema di protezione di Windows.

3. Prestazioni

3.1 Database “genes22”

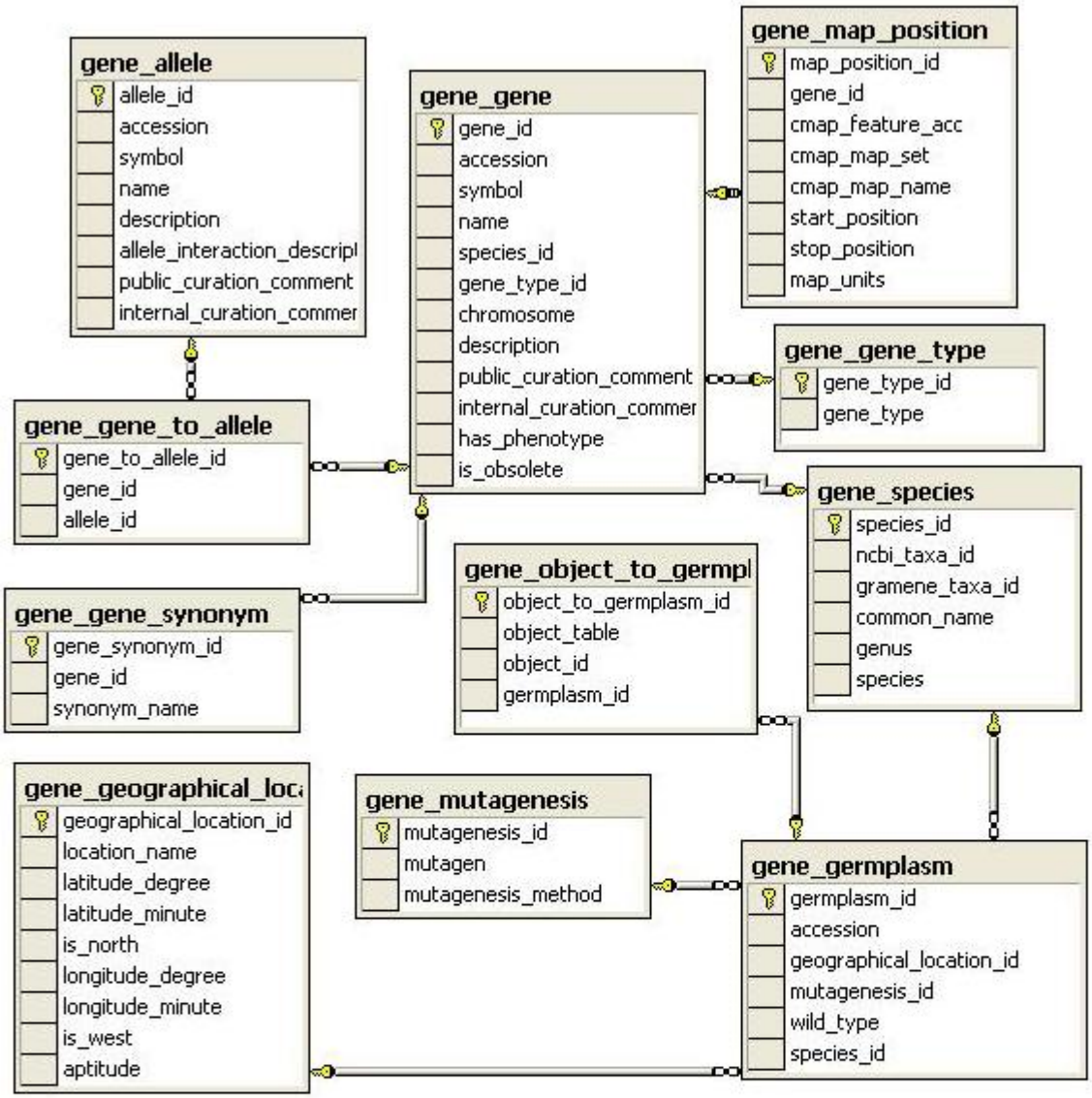

figura 3: Database "Genes22"

Nella figura 3 viene presentato attraverso un diagramma, il database "genes22" sul quale ,usando i due DBMS saranno effettuate le query e si confronteranno i tempi di risposta . L'intero database è costituito da 20 tabelle e 58557 record. Nel diagramma vengono rappresentate le tabelle sulle quali sono state effettuate le query.

Nota In MySQL per amministrare il database è stato usato MySQL Administrator e MySQL Query Browser per effettuare le query e il database è stato creato usando lo storage engine, MyISAM.

3.1.1 Select

Con il comando SELECT abbiamo la possibilità di estrarre i dati, in modo mirato, dal database.

Schema relazionale

gene_allele(allele_id, accession, symbol, name, description, allele_interaction_description, public_curation_comment, internal_curation_comment)

Seleziona l' allele_id, nome e simbolo di tutti gli alleli dei geni con allele_id compreso tra 1 e 500 e con nome che inizia con m. Visualizza l'output ordinato in senso discendente rispetto all' allele_id.

```
SELECT allele_id, name, symbol
FROM gene_allele
WHERE allele_id BETWEEN 1 AND 500
AND name LIKE 'm%'
ORDER BY 1 DESC
```

DBMS	*Tempo di risposta*
MySQL 5.0	0 ***s***
SQL Server 2000	1 ***s***

Nota Sia su Query Analyzer (GUI di Microsoft SQL Server 2000 per eseguire le query) e anche su MySQL Query Browser (GUI di MySQL 5.0 per eseguire le query) ci viene restituito il tempo di esecuzione delle query.

3.1.2 Join

Il risultato di un join tra due schemi relazionali R(X) e S(Y) comprende sole le tuple r di R(X) che possono essere messe in corrispondenza ,in base al predicato di join con una tupla s di S(y).

Query a) - Schema relazionale

gene_gene (gene_id, accession, symbol, name, species_id, gene_type_id, chromosome, description, public_curation_comment, internal_curation_comment, has_phenotype, is_obsolete)

FK : species_id **REFERENCES** gene_species
FK : gene_type_id **REFERENCES** gene_gene_type

gene_gene_synonym (gene_synonym_id, gene_id, synonym_name)

FK : gene_id **REFERENCES** gene_gene

Seleziona per ogni gene il suo nome ,il suo sinonimo , il gene_id e il gene_synonym_id.

```
SELECT gene_gene.name, gene_gene_synonym.synonym_name,
gene_gene.gene_id, gene_gene_synonym.gene_synonym_id
FROM gene_gene , gene_gene_synonym
WHERE gene_gene.gene_id = gene_gene_synonym.gene_id
```

Che è equivalente a :

```
SELECT gene_gene.name, gene_gene_synonym.synonym_name,
gene_gene.gene_id, gene_gene_synonym.gene_synonym_id
FROM gene_gene JOIN gene_gene_synonym
ON (gene_gene.gene_id = gene_gene_synonym.gene_id)
```

DBMS	***Tempo di risposta***
MySQL 5.0	1 ***s***
SQL Server 2000	2 ***s***

Query b) - Schema relazionale

gene_gene (gene_id, accession, symbol, name, species_id, gene_type_id, chromosome, description, public_curation_comment, internal_curation_comment, has_phenotype, is_obsolete)

FK : species_id **REFERENCES** gene_species
FK : gene_type_id **REFERENCES** gene_gene_type

gene_species (species_id, ncbi_taxa_id, gramene_taxa_id, common_name, genus, species)

b.1) Seleziona per ogni gene il suo nome ,la specie a cui appartiene , il gene_id e la species_id.

```
SELECT gene_gene.name,gene_gene.gene_id,
gene_species.species , gene_species.species_id
FROM gene_gene , gene_species
WHERE gene_species.species_id = gene_gene.species_id
```

b.2) Seleziona i nomi dei geni e i rispettivi gene_id che appartengono alla specie ‘sativa’.

```
SELECT gene_gene.name,gene_gene.gene_id
FROM gene_gene , gene_species
WHERE gene_species.species_id = gene_gene.species_id
AND species = 'sativa'
```

Query b.1)

DBMS	***Tempo di risposta***
MySQL 5.0	0 ***s***
SQL Server 2000	0 ***s***

Query b.2)

DBMS	***Tempo di risposta***
MySQL 5.0	0 ***s***
SQL Server 2000	0 ***s***

Query c) - Schema relazionale

gene_gene (gene_id, accession, symbol, name, species_id, gene_type_id, chromosome, description, public_curation_comment, internal_curation_comment, has_phenotype, is_obsolete)

FK : species_id **REFERENCES** gene_species
FK : gene_type_id **REFERENCES** gene_gene_type

gene_map_position (map_position_id, gene_id, cmap_feature_acc, cmap_map_set, cmap_map_name, start_position, stop_position, map_units)

FK : gene_id **REFERENCES** gene_gene

Seleziona per ogni gene il suo nome , il gene_id ,il nome della mappa ,la posizione iniziale e la sua posizione finale.

```
SELECT gene_gene.name,gene_gene.gene_id,
gene_map_position.cmap_map_name ,
gene_map_position.start_position,
gene_map_position.stop_position
FROM gene_gene , gene_map_position
WHERE gene_gene.gene_id= gene_map_position.gene_id
```

DBMS	*Tempo di risposta*
MySQL 5.0	0 ***s***
SQL Server 2000	0 ***s***

3.1.3 SELF JOIN

IL self join è un join tra una tabella e se stessa.

Schema relazionale

gene_gene (gene_id, accession, symbol, name, species_id, gene_type_id, chromosome, description, public_curation_comment, internal_curation_comment, has_phenotype, is_obsolete)

FK : species_id **REFERENCES** gene_species
FK : gene_type_id **REFERENCES** gene_gene_type

gene_species (species_id, ncbi_taxa_id, gramene_taxa_id, common_name, genus, species)

Seleziona i gene_id , i simboli e il common_ name della specie dei geni che hanno un numero di cromosomi uguale al numero di cromosomi del gene ‘silky1’.

```
SELECT  g1.gene_id, g1.symbol, gene_species.common_name
FROM gene_gene g1, gene_gene g2,gene_species
WHERE g2.species_id = gene_species.species_id
AND g1.chromosome = g2.chromosome
AND g1.name = 'silky1'
```

DBMS	*Tempo di risposta*

MySQL 5.0	0 ***s***
SQL Server 2000	0 ***s***

3.1.4 LEFT JOIN /RIGHT JOIN /FULL JOIN

Una tupla r di R(X) (s di S(Y)) che non contribuisce al join è detta tupla dangling.

a) LEFT/RIGHT OUTER JOIN

Il left/right outer join tra due schemi relazionali R(X) e S(Y) comprende nel risultato di join anche le tuple dangling di R(X) / S(Y).

Schema relazionale

gene_species (species_id, ncbi_taxa_id, gramene_taxa_id, common_name, genus, species)
gene_germplasm (germplasm_id, accession, geographical_location_id, mutagenesis_id, wild_type, species_id)

FK : geographical_location_id **REFERENCES** gene_geographical_location
FK : mutagenesis_id **REFERENCES** gene_mutagenesis
FK : species_id **REFERENCES** gene_species

1)LEFT JOIN
Seleziona i germplasmi che appartengono ad una specie includendo quelli che non appartengono a nessuna specie.

2) RIGHT JOIN
Seleziona le specie e i germplasmi corrispondenti, includendo le specie per cui non c'è una corrispondenza.

```
SELECT gene_germplasm.*, gene_species.*
FROM gene_germplasm LEFT/RIGHT JOIN  gene_species
ON(gene_species.species_id = gene_germplasm.species_id)
```

1)LEFT JOIN

DBMS	***Tempo di risposta***
MySQL 5.0	0 ***s***
SQL Server 2000	0 ***s***

2)RIGHT JOIN

DBMS	*Tempo di risposta*
MySQL 5.0	0 ***s***
SQL Server 2000	0 **s**

b) FULL OUTER JOIN

Schema relazionale

gene_species (species_id, ncbi_taxa_id, gramene_taxa_id, common_name, genus, species)
gene_germplasm (germplasm_id, accession, geographical_location_id, mutagenesis_id, wild_type, species_id)

FK : geographical_location_id **REFERENCES** gene_geographical_location
FK : mutagenesis_id **REFERENCES** gene_mutagenesis
FK : species_id **REFERENCES** gene_species

Seleziona i germplasmi e le specie corrispondenti includendo le tuple per cui no ci sono corrispondenze.

```
SELECT gene_germplasm.*, gene_species.*
FROM gene_germplasm FULL JOIN  gene_species
ON(gene_species.species_id = gene_germplasm.species_id)
```

Questa query scritta seguendo lo standard SQL92 funziona solo in Microsoft SQL Server 2000 e non in MySQL 5.0 perché quest'ultimo non supporta il full join.Quindi per fare un full join in MySQL 5.0 bisogna fare un unione tra un left e un right outer join.

```
SELECT *
FROM gene_germplasm LEFT JOIN gene_species ON
(gene_germplasm.species_id = gene_species.species_id)
UNION
SELECT *
FROM gene_germplasm
RIGHT JOIN gene_species ON (gene_germplasm.species_id =
gene_species.species_id)
```

FULL JOIN

DBMS	*Tempo di risposta*
MySQL 5.0	0 ***s***
SQL Server 2000	1 ***s***

3.1.5 JOIN tra 3 tabelle

Query a) - Schema relazionale

gene_gene (gene_id, accession, symbol, name, species_id, gene_type_id, chromosome, description, public_curation_comment, internal_curation_comment, has_phenotype, is_obsolete)

FK : species_id **REFERENCES** gene_species
FK : gene_type_id **REFERENCES** gene_gene_type

gene_gene_to_allele (gene_to_allele_id, gene_id, allele_id)

FK : gene_id **REFERENCES** gene_gene
FK : allele_id **REFERENCES** gene_allele

gene_allele(allele_id, accession, symbol, name, description, allele_interaction_description, public_curation_comment, internal_curation_comment)

Seleziona per ciascun gene con gene_id > 100 il nome ,il gene_id , il gene_to_allele_id ,il nome dell'allele ed il allele_id.

```
SELECT gene_gene.name,gene_gene.gene_id,
gene_gene_to_allele.gene_to_allele_id,
gene_allele.name, gene_allele.allele_id
FROM gene_gene ,gene_gene_to_allele ,gene_allele
WHERE gene_gene.gene_id = gene_gene_to_allele.gene_id
AND gene_gene_to_allele.allele_id =
gene_allele.allele_id
AND gene_gene.gene_id > 100
```

Che è equivalente alla seguente query in base allo standard SQL92:

```
SELECT gene_gene.name,gene_gene.gene_id,
gene_gene_to_allele.gene_to_allele_id,
gene_allele.name, gene_allele.allele_id
```

```
FROM (gene_gene JOIN gene_gene_to_allele ON
(gene_gene.gene_id = gene_gene_to_allele.gene_id))
JOIN gene_allele ON(gene_gene_to_allele.allele_id =
gene_allele.allele_id)
WHERE gene_gene.gene_id > 100
```

Questa query funziona solo in Microsoft SQL Server 2000 e non funziona in MySQL 5.0. Per funzionare in MySQL 5.0 deve essere riscritta e la query equivalente è la seguente:

```
SELECT gene_gene.name,gene_gene.gene_id,
gene_gene_to_allele.gene_to_allele_id,
gene_allele.name, gene_allele.allele_id
FROM gene_gene JOIN ( gene_gene_to_allele
JOIN gene_allele ON gene_allele.allele_id =
gene_gene_to_allele.allele_id)
ON gene_gene.gene_id = gene_gene_to_allele.gene_id
WHERE gene_gene.gene_id > 100
```

DBMS	***Tempo di risposta***
MySQL 5.0	0 ***s***
SQL Server 2000	0 ***s***

Query b) - Schema relazionale

gene_map_position (map_position_id, gene_id, cmap_feature_acc, cmap_map_set, cmap_map_name, start_position, stop_position, map_units)

FK : gene_id **REFERENCES** gene_gene

gene_gene (gene_id, accession, symbol, name, species_id, gene_type_id, chromosome, description, public_curation_comment, internal_curation_comment, has_phenotype, is_obsolete)

FK : species_id **REFERENCES** gene_species
FK : gene_type_id **REFERENCES** gene_gene_type

gene_gene_type (gene_type_id, gene_type)

Seleziona il map_position_id ,le feature della mappa ed il nome di tutti i geni con tipo ‘CDS (Protein coding)’

```
SELECT gene_map_position.map_position_id,
gene_map_position.cmap_feature_acc, gene_gene.name
FROM gene_map_position, gene_gene, gene_gene_type
```

```
WHERE gene_map_position.gene_id = gene_gene.gene_id
AND gene_gene.gene_type_id =
gene_gene_type.gene_type_id
AND gene_gene_type.gene_type  = 'CDS (Protein coding)'
```

DBMS	*Tempo di risposta*
MySQL 5.0	0 ***s***
SQL Server 2000	0 ***s***

Query c) - Schema relazionale

gene_gene (gene_id, accession, symbol, name, species_id, gene_type_id, chromosome, description, public_curation_comment, internal_curation_comment, has_phenotype, is_obsolete)

FK : species_id **REFERENCES** gene_species
FK : gene_type_id **REFERENCES** gene_gene_type

gene_gene_type (gene_type_id, gene_type)
gene_species (species_id, ncbi_taxa_id, gramene_taxa_id, common_name, genus, species)

Seleziona il nome , il numero dei cromosomi e il nome comune della specie dei geni che appartengono alla specie ‘sativa’ e il tipo è ‘CDS (Protein coding)’

```
SELECT gene_gene.name,
gene_gene.chromosome,gene_species.common_name
FROM  gene_gene, gene_gene_type ,gene_species
WHERE gene_gene.species_id = gene_species.species_id
AND gene_gene.gene_type_id =
gene_gene_type.gene_type_id
AND gene_species.species = 'sativa'
AND gene_gene_type.gene_type  = 'CDS (Protein coding)'
```

DBMS	*Tempo di risposta*
MySQL 5.0	0 ***s***
SQL Server 2000	0 ***s***

3.1.6 JOIN tra 4 tabelle

Query a) - Schema relazionale

gene_gene (gene_id, accession, symbol, name, species_id, gene_type_id, chromosome, description, public_curation_comment, internal_curation_comment, has_phenotype, is_obsolete)

FK : species_id **REFERENCES** gene_species
FK : gene_type_id **REFERENCES** gene_gene_type

gene_gene_synonym (gene_synonym_id, gene_id, synonym_name)

FK : gene_id **REFERENCES** gene_gene

gene_gene_type (gene_type_id, gene_type)
gene_species (species_id, ncbi_taxa_id, gramene_taxa_id, common_name, genus, species)

Seleziona il nome, il simbolo, la descrizione ,la specie di appartenenza e il sinonimo dei geni appartenenti al tipo 'Not sequenced ' e con gene_synonym_id > 200.

```
SELECT gene_gene.name, gene_gene.symbol
,gene_gene.description, gene_species.species,
gene_gene_synonym.synonym_name
FROM gene_gene,gene_species
,gene_gene_type,gene_gene_synonym
WHERE gene_gene.gene_id = gene_gene_synonym.gene_id
AND gene_species.species_id = gene_gene.species_id
AND gene_gene_type.gene_type_id =
gene_gene.gene_type_id
AND gene_gene_type.gene_type = 'Not sequenced'
AND gene_gene_synonym.gene_synonym_id  > 200
```

DBMS	***Tempo di risposta***
MySQL 5.0	0 ***s***
SQL Server 2000	0 ***s***

Query b) - Schema relazionale

gene_gene (gene_id, accession, symbol, name, species_id, gene_type_id, chromosome, description, public_curation_comment, internal_curation_comment, has_phenotype, is_obsolete)

FK : species_id **REFERENCES** gene_species
FK : gene_type_id **REFERENCES** gene_gene_type

gene_species (species_id, ncbi_taxa_id, gramene_taxa_id, common_name, genus, species)
gene_allele(allele_id, accession, symbol, name, description, allele_interaction_description, public_curation_comment, internal_curation_comment)
gene_gene_to_allele (gene_to_allele_id, gene_id, allele_id)

FK : gene_id **REFERENCES** gene_gene
FK : allele_id **REFERENCES** gene_allele

Seleziona gli alleli e i rispettivi simboli dei geni la cui 'genus' della specie è 'Oryza' e che presentano fenotipo.

```
SELECT gene_allele.allele_id, gene_allele.symbol
FROM gene_gene,gene_allele ,gene_species ,
gene_gene_to_allele
WHERE gene_gene.gene_id = gene_gene_to_allele.gene_id
AND gene_species.species_id = gene_gene.species_id
AND gene_gene_to_allele.allele_id =
gene_allele.allele_id
AND gene_species.genus = 'Oryza'
AND gene_gene.has_phenotype = 'yes'
```

DBMS	*Tempo di risposta*
MySQL 5.0	0 ***s***
SQL Server 2000	0 ***s***

3.1.7 JOIN tra 5 tabelle

Schema relazionale

gene_gene (gene_id, accession, symbol, name, species_id, gene_type_id, chromosome, description, public_curation_comment, internal_curation_comment, has_phenotype, is_obsolete)

FK : species_id **REFERENCES** gene_species
FK : gene_type_id **REFERENCES** gene_gene_type

gene_gene_type (gene_type_id, gene_type)
gene_species (species_id, ncbi_taxa_id, gramene_taxa_id, common_name, genus, species)
gene_allele(allele_id, accession, symbol, name, description, allele_interaction_description, public_curation_comment, internal_curation_comment)
gene_gene_to_allele (gene_to_allele_id, gene_id, allele_id)

FK : gene_id **REFERENCES** gene_gene
FK : allele_id **REFERENCES** gene_allele

Seleziona per ciascun gene che appartiene alla specie con nome comune 'Rice' e tipo 'Not sequenced' , il nome ,il nome dell'allele ,la descrizione del gene e il numero dei cromosomi.

```
SELECT gene_gene.name, gene_allele.name
,gene_gene.description,gene_gene.chromosome
FROM gene_gene,gene_allele ,gene_species ,
gene_gene_to_allele ,gene_gene_type
WHERE gene_gene.gene_id = gene_gene_to_allele.gene_id
AND gene_species.species_id = gene_gene.species_id
AND gene_gene_to_allele.allele_id =
gene_allele.allele_id
AND gene_gene_type.gene_type_id =
gene_gene.gene_type_id
AND gene_species.common_name = 'Rice'
AND gene_gene_type.gene_type = 'Not sequenced'
```

DBMS	***Tempo di risposta***
MySQL 5.0	0 ***s***
SQL Server 2000	0 ***s***

3.1.8 JOIN tra 6 tabelle

Schema relazionale

gene_gene (gene_id, accession, symbol, name, species_id, gene_type_id, chromosome, description, public_curation_comment, internal_curation_comment, has_phenotype, is_obsolete)

FK : species_id **REFERENCES** gene_species
FK : gene_type_id **REFERENCES** gene_gene_type

gene_gene_synonym (gene_synonym_id, gene_id, synonym_name)
gene_gene_type (gene_type_id, gene_type)
gene_species (species_id, ncbi_taxa_id, gramene_taxa_id, common_name, genus, species)
gene_allele(allele_id, accession, symbol, name, description, allele_interaction_description, public_curation_comment, internal_curation_comment)
gene_gene_to_allele (gene_to_allele_id, gene_id, allele_id)
FK : gene_id **REFERENCES** gene_gene
FK : allele_id **REFERENCES** gene_allele

Seleziona per ciascun gene che appartiene alla specie ‘sativa’ e tipo ‘CDS (Protein coding)’, il nome ,il nome dell’allele ,e il sinonimo.

```
SELECT gene_gene.name, gene_allele.name
,gene_gene_synonym.synonym_name
FROM gene_gene,gene_allele ,gene_species ,
gene_gene_to_allele ,gene_gene_type, gene_gene_synonym
WHERE gene_gene.gene_id = gene_gene_to_allele.gene_id
AND gene_gene.gene_id= gene_gene_synonym.gene_id
AND gene_species.species_id = gene_gene.species_id
AND gene_gene_to_allele.allele_id =
gene_allele.allele_id
AND gene_gene_type.gene_type_id =
gene_gene.gene_type_id
AND gene_species.species = 'sativa'
AND gene_gene_type.gene_type = 'CDS (Protein coding)'
```

DBMS	***Tempo di risposta***
MySQL 5.0	0 ***s***
SQL Server 2000	1 ***s***

3.1.9 Conclusioni

Sul database sono stati effettuati dei join tra varie tabelle perché sono le query che "costano" di più in termini di tempi di risposta. Come si può vedere anche dai tempi di risposta delle varie query le prestazioni dei due DBMS sono sostanzialmente simili (tempi di risposta "istantanei") per il 75% delle query e per il 25% SQL Server 2000 ha dei tempi di risposta superiori .

3.2 Database "marker19"

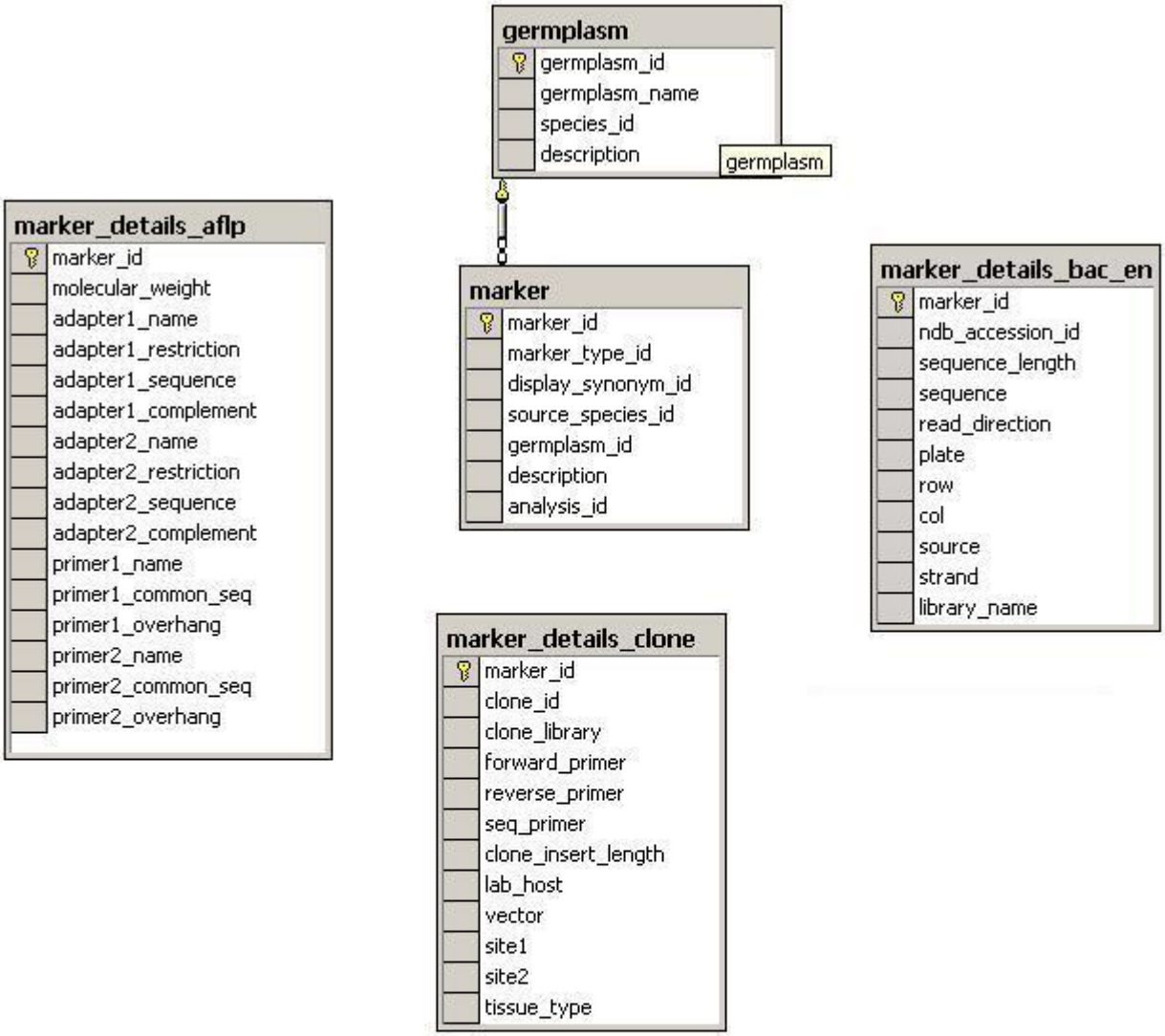

Figura 4 – Database "marker19"

Nella figura 4 viene presentato attraverso un diagramma, il database "marker19" che è un altro database sul quale saranno effettuate delle query ,usando i due DBMS e si confronteranno i tempi di risposta . L'intero database è costituito da 5 tabelle e 5,832,461 record.

Nota In MySQL per amministrare il database è stato usato MySQL Administrator e MySQL Query Browser per effettuare le query e il database è stato creato usando lo storage engine, MyISAM.

3.2.1 Select

Schema relazionale

marker(marker_id ,marker_type_id, display_synonym_id, source_species_id, germplasm_id, description,anlysis_id **)**

FK germplasm_id **REFERENCES** germplasm

germplasm(germplasm_id, gemplasm_name, species_id, description**)**
marker_details_clone(marker_id, clone_id, clone_library, forward_primer, reverse_primer, seq_primer, clone_insert_length, lab_host, vector, site1, site2, tissue_type **)**
marker_details_aflp(marker_id, molecular_weight, adapter1_name, adapter1_restriction, adapter1_sequence, adapter1_complement, adapter2_name, adapter2_restriction, adapter2_sequence, adapter2_complement, primer1_name, primer1_common_seq, primer1_overhang, primer2_name, primer2_common_seq, primer2_overhang**)**
marker_details_bac_end_sequence(marker_id**,** ndb_accession_id, sequence_length, sequence, read_direction, plate, row, col ,source ,strand ,library_name**)**

Seleziona marker_id, sequence_length e sequence di tutti i marker con marker_id incluso tra 30000 e 300000 e con sequenza di DNA che inizia con G. Visualizza l'output ordinato in senso decrescente rispetto all' marker_id.

```
SELECT marker_id , sequence_length, sequence
FROM marker_details_bac_end_sequence
WHERE marker_id between 30000 and 300000
AND sequence LIKE 'G%'
ORDER BY 1 DESC
```

DBMS	*Tempo di risposta*
MySQL 5.0	37 ***s***
SQL Server 2000	44 ***s***

3.2.2 Join

Seleziona per ogni marker il marker_id e il clone_id.

```
SELECT marker_details_bac_end_sequence.marker_id,
marker_details_clone.clone_id
FROM marker_details_bac_end_sequence JOIN
marker_details_clone
ON (marker_details_bac_end_sequence.marker_id  =
marker_details_clone.marker_id)
```

DBMS	***Tempo di risposta***
MySQL 5.0	12 ***s***
SQL Server 2000	34 ***s***

Seleziona per ogni marker con marker_id > 600000,il marker_id , il clone_id e il clone_library .

```
SELECT marker.marker_id, marker_details_clone.clone_id,
marker_details_clone.clone_library
FROM marker JOIN marker_details_clone
ON (marker.marker_id = marker_details_clone.marker_id)
AND marker.marker_id > 600000
```

DBMS	***Tempo di risposta***
MySQL 5.0	28 ***s***
SQL Server 2000	50 ***s***

Per ogni germplasm con specie_id=2 visualizza la descrizione del marker , il germplasm_id e il germplasm_name.

```
SELECT marker.description, germplasm.germplasm_id,
germplasm.germplasm_name
FROM marker JOIN germplasm
ON (marker.germplasm_id = germplasm.germplasm_id)
AND germplasm.species_id = 2
```

DBMS	***Tempo di risposta***

MySQL 5.0	28 ***s***
SQL Server 2000	50 ***s***

Per ogni marker con marker_id > 350000 e sequenza di DNA che inizia con G visualizza il marker_id, il germplasm_id, la sequenza di DNA e la lunghezza della sequenza.

```
SELECT  marker.marker_id, marker.germplasm_id,
marker_details_bac_end_sequence.sequence,
marker_details_bac_end_sequence.sequence_length
FROM marker_details_bac_end_sequence JOIN  marker
ON (marker_details_bac_end_sequence.marker_id =
marker.marker_id)
AND marker.marker_id > 350000
AND marker_details_bac_end_sequence.sequence LIKE 'G%'
```

DBMS	***Tempo di risposta***
MySQL 5.0	1min 42 ***s***
SQL Server 2000	2min 47 ***s***

3.2.3 SELF JOIN

Seleziona la sequenza ,il source e il gemplasm_id dei marker che hanno la stessa lunghezza di DNA del marker con marker_id = 32000

```
SELECT  m1.sequence, m2.source, marker.germplasm_id
FROM  marker_details_bac_end_sequence m1 ,
marker_details_bac_end_sequence m2, marker
WHERE m1.marker_id = marker.marker_id
AND m1.sequence_length = m2.sequence_length
AND m1.marker_id = 32000
```

DBMS	***Tempo di risposta***
MySQL 5.0	26 ***s***
SQL Server 2000	28 ***s***

3.2.4 LEFT JOIN /RIGHT JOIN /FULL JOIN

a) LEFT/RIGHT OUTER JOIN

1)LEFT JOIN
Seleziona tutti i germplasm_id con il marker_id < 100000 corrispondente includendo quelli che non hanno un marker_id corrispondente.

```
SELECT germplasm.germplasm_id, marker.marker_id
FROM  germplasm LEFT JOIN marker
ON ( marker.germplasm_id = germplasm.germplasm_id)
AND marker.marker_id < 100000
```

2) RIGHT JOIN
Seleziona tutti i clone_id con il marker_id > 200000 corrispondente includendo quelli che non hanno un marker_id corrispondente.

```
SELECT marker.marker_id, marker_details_clone.clone_id
FROM marker RIGHT JOIN marker_details_clone
ON ( marker.marker_id = marker_details_clone.marker_id)
AND marker.marker_id > 200000
```

1)LEFT JOIN

DBMS	***Tempo di risposta***
MySQL 5.0	46 ***s***
SQL Server 2000	2 ***s***

2)RIGHT JOIN

DBMS	***Tempo di risposta***
MySQL 5.0	1min 16 ***s***
SQL Server 2000	38 ***s***

b) FULL OUTER JOIN

Seleziona tutti i marker_id < 60000 con il clone_id corrispondente ,includendo i marker_id (clone_id) per cui non esiste una corrispondenza con il clone_id (marker_id).

```
SELECT marker_details_bac_end_sequence.marker_id,
marker_details_clone.clone_id
```

```
FROM marker_details_bac_end_sequence FULL JOIN
marker_details_clone
ON (marker_details_bac_end_sequence.marker_id =
marker_details_clone.marker_id)
AND marker_details_bac_end_sequence.marker_id < 60000
```

Questa query scritta seguendo lo standard SQL92 funziona solo in Microsoft SQL Server 2000 e non in MySQL 5.0 perché quest'ultimo non supporta il full join.Quindi per fare un full join in MySQL 5.0 bisogna fare un unione tra un left e un right outer join.

```
SELECT marker_details_bac_end_sequence.marker_id,
marker_details_clone.clone_id
FROM marker_details_bac_end_sequence LEFT JOIN
marker_details_clone
ON (marker_details_bac_end_sequence.marker_id =
marker_details_clone.marker_id)
AND marker_details_bac_end_sequence.marker_id < 60000

UNION
SELECT marker_details_bac_end_sequence.marker_id,
marker_details_clone.clone_id
FROM marker_details_bac_end_sequence RIGHT JOIN
marker_details_clone
ON (marker_details_bac_end_sequence.marker_id =
marker_details_clone.marker_id)
AND marker_details_bac_end_sequence.marker_id < 60000
```

FULL JOIN

DBMS	***Tempo di risposta***
MySQL 5.0	2min 52 ***s***
SQL Server 2000	50 ***s***

3.2.5 JOIN tra 3 tabelle

Seleziona per ogni marker con marker_id > 35000 e species_id del germplasm corrispondente uguale a 1,il marker_id, la source_species_id, il nome del germplasm e la sequenza di DNA.

```
SELECT marker.marker_id, marker.source_species_id,
germplasm.germplasm_name,
marker_details_bac_end_sequence.sequence
```

```
FROM marker_details_bac_end_sequence , marker,
germplasm
WHERE marker_details_bac_end_sequence.marker_id =
marker.marker_id
AND marker.germplasm_id = germplasm.germplasm_id
AND marker.marker_id > 35000
AND germplasm.species_id=1
```

DBMS	*Tempo di risposta*
MySQL 5.0	4 ***s***
SQL Server 2000	36 ***s***

3.2.6 JOIN tra 4 tabelle

Seleziona per ogni marker con species_id del germplasm corrispondente uguale a 1, il marker_id, il germplasm_id ,la sequenza di DNA e il clone_library.

```
SELECT marker.marker_id, germplasm.germplasm_id,
marker_details_bac_end_sequence.sequence,
marker_details_clone.clone_library
FROM marker, marker_details_clone,  germplasm,
marker_details_bac_end_sequence
WHERE marker.marker_id = marker_details_clone.marker_id
AND marker.germplasm_id = germplasm.germplasm_id
AND marker.marker_id =
marker_details_bac_end_sequence.marker_id
AND germplasm.species_id = 1
```

DBMS	*Tempo di risposta*
MySQL 5.0	1 ***s***
SQL Server 2000	34 ***s***

3.2.7 Query innestate

Una interrogazione viene detta innestata o nidificata se la sua condizione e' formulata usando il risultato di un'altra interrogazione, chiamata subquery.

Seleziona il marker_type_id dei germplasmi con species_id = 1.

```
SELECT marker.marker_type_id
FROM marker
WHERE germplasm_id in (SELECT germplasm_id
                       FROM germplasm
                       WHERE species_id = 1)
```

DBMS	*Tempo di risposta*
MySQL 5.0	40 ***s***
SQL Server 2000	34 ***s***

Seleziona la descrizione dei marker con marker_id < 60000 i cui germplasmi non appartengono alla specie con species_id = 1.

```
SELECT marker.description
FROM marker
WHERE marker_id < 60000
AND NOT EXISTS
     (SELECT *
     FROM germplasm
     WHERE germplasm.germplasm_id = marker.germplasm_id
     AND species_id = 1)
```

DBMS	*Tempo di risposta*
MySQL 5.0	7 ***s***
SQL Server 2000	5 ***s***

Seleziona il nome dei germplasmi il cui clone_id = OR_BBa0088A07.

```
SELECT germplasm.germplasm_name
FROM germplasm
WHERE germplasm_id in
            (SELECT germplasm_id
             FROM marker
             WHERE marker_id in
                      (SELECT marker_id
                      FROM marker_details_clone
                      WHERE clone_id= 'OR_BBa0088A07'))
```

DBMS	*Tempo di risposta*

MySQL 5.0	1min 33 ***s***
SQL Server 2000	9 ***s***

Seleziona la species_id dei germplasmi dei marker la cui sequenza di DNA inizia con CAGGA.

```
SELECT germplasm.species_id
FROM germplasm
WHERE germplasm_id in
            (SELECT germplasm_id
             FROM marker
             WHERE marker_id in
                (SELECT marker_id
                  FROM marker_details_bac_end_sequence
                  WHERE sequence like 'CAGGA%'))
```

DBMS	***Tempo di risposta***
MySQL 5.0	1min 16 ***s***
SQL Server 2000	41 ***s***

3.2.8 Conclusioni

Sul database sono stati effettuati dei join e delle query innestate tra varie tabelle. Come si può vedere anche dai tempi di risposta delle varie query le prestazioni dei due DBMS sono molto diversi.

	Tempi di risposta dei DBMS	
Query	Tempo di risposta superiore	Tempo di risposta inferiore
INNER JOIN	SQL Server 2000	MySQL 5.0
LEFT/RIGHT/FULL JOIN	MySQL 5.0	SQL Server 2000
Query Innestate	MySQL 5.0	SQL Server 2000

Se nella query abbiamo un left/right o full join i tempi di risposta di MySQL 5.0 sono superiori più del doppio rispetto ai tempi di risposta di SQL Server 2000 e anche in caso di query innestate i tempi di risposta di MySQL 5.0 sono superiori. Invece in caso di inner join tra 2 o più tabelle i tempi di riposta di MySQL 5.0 sono inferiori.

4. Conclusioni

Il progetto che intendevo realizzare è stato portato a termine e può dirsi concluso: è stato approfondito l'uso di SQL Server 2000 ed è stato esaminato un alternativa a questo RDBMS data da MySQL 5.0 .Tutte le funzionalità che sono state aggiunte a MySQL 5.0 lo hanno avvicinato di più ai DBMS di classe enterprise .

Scegliere quale DBMS usare dipende dalle esigenze del progetto specifico. Microsoft SQL Server 2000 rappresenta un RDBMS ad alte prestazioni progettato per gestire altissimi volumi di operazioni transazionali in ambiente multiutente. SQL Server 2000 presenta molte funzionalità per manipolare e gestire i dati e gli amministratori di database dispongono di tutte le funzionalità necessarie per configurare e gestire server di database sicuri e perfettamente integrati con il sistema di protezione di Windows (unico sistema operativo con cui si può utilizzare SQL Server 2000).

MySQL 5.0 presenta 3 punti di forza molto importanti:

- Open Source
- Multipiattaforma
- Costi contenuti anche con licenza commerciale

Concludendo, il progetto svolto è stato molto interessante e particolarmente stimolante in quanto mi ha offerto la possibilità di usare strumenti ed approfondire argomenti del tutto nuovi.

5. Bibliografia

[1] D.Beneventano , S.Bergamaschi , M.Vincini ,***Progetto di Basi di Dati Relazionali***, ed. Pitagora Editrice Bologna

[2] R.M.Riordan **, *Programmare Microsoft SQL Server 2000 Passo per Passo,*** ed. Mondadori Informatica

[3] O.Reilly ,***MySQL in a Nutshell***

[4] DataBase Group Dipartimento di Ingegneria dell'Informazione Facoltà di Ingegneria di Modena:
http://www.dbgroup.unimo.it/

[5] Microsoft Home Page:
http://www.microsoft.com

[6] Documentazione in linea di SQL Server 2000 (SQL Book On Line)

[7] Sicurezza di Microsoft SQL Server 2000
http://www.microsoft.com/italy/sql/previous/2000/securitywp.mspx

[8] MySQL Home Page
http://www.mysql.com

[9] Manuale di riferimento di MySQL 5.0
www.mysql.com/documentation/refman-5.0

[10] Informazioni sui database
http://sicurezza.html.it/guide
http://database.html.it/guide
(guide, esempi,articoli…)

[11] Informazioni sui database:"genes22" e "marker19"
www.gramene.org

[12] http://www.wikipedia.org