

SQL Databases Using MySQL and PostgreSQL

CTEC173 I/2007F

Client/Server Architecture

- Clients and Servers
- Clients deal with user interface
- SQL server executes SQL and deals with database access
- Connected by network

SQL Servers

- SQL server manages the data and memory
- Manages multiple databases and multiple users
- Keeps track of actual location of data on disks
- Maintains mapping of logical data description to physical data storage

SQL Servers (2)

- SQL server understands SQL
- Compiles and executes SQL statements
- Returns results to client programs
- Multi-threaded architecture
- SQL server itself manages multiple users
- Does not rely on host operating system to perform multi-tasking

SQL Databases

- Definition of a database:

Collection of tables holding related information

- Multiple databases are common
- Data dictionary is stored in each database

SQL

- SQL stands for “Structured Query Language” and is an ANSI Standard
- It includes constructs for:

Data Definition

Data Manipulation

Data Control

Creating a Database

- You create a database with the **create database** command
- You must have *create database* permission, or be the *database administrator* (DBA)
- Syntax:

```
create database dbname
```

Creating Databases (2)

- In MySQL, you can create a database through the operating system command line:

```
mysqladmin create dbname
```

- In PostgreSQL, you can do this using:

```
createdb dbname
```

Displaying Information About Databases - MySQL

- In MySQL, from the `mysql` prompt, enter:

show databases;

```
mysql> show databases ;
+-----+
| Database                |
+-----+
| information_schema      |
| test                    |
| world                   |
+-----+
3 rows in set (0.05 sec)

mysql>
```

Displaying Information About Databases - PostgreSQL

- In PostgreSQL, from the `psql` prompt, enter:

`\l` (*lowercase L*) or `\l+` (*for more info*)

```
postgres=# \l
          List of databases
  Name      | Owner   | Encoding
-----+-----+-----
 phppgadmin | postgres | SQL_ASCII
 postgres   | postgres | SQL_ASCII
 template0  | postgres | SQL_ASCII
 template1  | postgres | SQL_ASCII
 test       | postgres | SQL_ASCII
(5 rows)

postgres=#
```

Finding out where you are

- When you are connected to the server, you are automatically accessing your "default" database.
- To find out what database you are currently accessing, use:

In MySQL: `select database() ;` or `\s`

In PostgreSQL, the current database is shown in the prompt (e.g., `test=#`)

- To access another database, run

use *database_name* (*in MySQL*)

\c *database_name* (*in PostgreSQL*)

- The database context remains set until you change it or exit your application.

What Goes In a Database?

- Databases contain data arranged in tables.
- Tables contain columns which can have datatypes, rules, and defaults.
- Tables can also have indexes, and the database may contain views, procedures, and triggers based on the tables.
-

Steps In Defining A Table

- Decide on what kinds of data (entities) are to be tracked.
- Divide the data into tables.
- Name the columns in each table, and decide which datatype (and length, if appropriate) each should be.
- Decide which columns should permit **null** values and which should not.

- Decide whether or not a default and/or rule is needed for each column. Consider the relationship between null/not null status of a column and the default or rule.
- Create a user-defined datatype for a group of columns having identical characteristics (datatype, null status, default, and rule).
- Create the tables.
- Define the defaults and rules.

- Bind the defaults and rules to the appropriate columns or user-defined datatypes.
- Load the data.
- Define and create the indexes.

Columns

- In order to define a table, we have to define the datatype of each column.
- Each column is of a single datatype: system-defined or user-defined.
- Example:
number integer,
name character(20)

Standard SQL Datatypes

- SQL defines distinct data types named by the following keywords:

**CHARACTER, CHARACTER VARYING,
CHARACTER LARGE OBJECT,
BINARY LARGE OBJECT, BIT, BIT VARYING,
NUMERIC, DECIMAL, INTEGER, SMALLINT,
ENUMERATED, FLOAT, REAL, DOUBLE
PRECISION, BOOLEAN, DATE, TIME,
TIMESTAMP, and INTERVAL.**

Creating Tables

- Tables are defined using the `create table` statement.
- The command does four things:
 1. defines a table name
 2. defines the number and names of the columns
 3. defines the kind of data that goes into each column
 4. defines whether nulls are allowed into each column

- For example, to create the `id` table:

```
create table id  
(  
    number integer not null primary key,  
    name character varying (20) not null  
) ;
```
- There can be a large number of (for example, up to 250) columns in a table, but such a large number is probably a symptom of a poor database design.

Examining Table Structure

- On MySQL, you can examine your table using the `describe tablename` command; or `show tables` to list all tables.
- On PostgreSQL, you can examine your table using the `\d tablename` command; just `\d` to list tables.

Defaults & Constraints

- Defaults specify the data to be entered into a column when a user does not specify it on input.
- Constraints are used to restrict values that can be entered into a column.
- The way both defaults and constraints are implemented varies on MySQL and PostgreSQL (and other database systems.)

Altering Tables

- Columns can be added to tables even after data is in the table.
- When columns are added, the columns added must allow nulls.
- This is because when you add a column, the server does not go back through the rows adding data to the new columns.

- Syntax:

```
alter table tablename  
    add columnname datatype NULL  
    [ , columnname datatype NULL ... ]
```

- Example:

```
alter table id  
    add issue_date date null,  
    add expiry_date date null ;
```

- You can rename or delete (**drop**) columns using `alter table` as well.

Indexes

- Indexes are used to improve performance.
- If no index is defined, the entire table must be searched to satisfy a where condition.
- Indexes also provide a mechanism for enforcing uniqueness.
- By default, an index is created for the primary key columns when a table is created.

Creating an index

- Syntax:

```
create [ unique ] index name  
  on table ( column, column, ... )
```

- Example:

```
create index issued_idx  
  on id ( issue_date ) ;
```

Enforcing Uniqueness

- In a unique index, no two rows may have the same index value.
- An attempt to create a duplicate key (by inserting or updating) will fail if there is a unique index on the key.
- The check for duplicates is performed:
 - When the index is created
 - On insert
 - On update

Primary, Foreign and Common Keys

- Relational database theory includes the concepts of:
 - **Primary Key**
One or more columns that uniquely identify a row. Primary keys must be unique and not null.
 - **Foreign Key**
One or more columns that refer to the primary key of another table.
 - **Common Key**
One or more columns that are frequently used to join tables.

- You can document which keys are primary, foreign, and common so that application programs have access to this information.
- You are thereby making explicit a logical relationship that is implicit in your database design.
- Keys are not the same as indexes.
- Keys are defined at logical design time while indexes are defined at physical design time for performance and other reasons.

Referential Integrity

- A column (or group of columns) which is used to uniquely identify rows in a table is called the **primary key** of the table.
- A column which contains the primary key to another table is called a **foreign key**.
- Maintaining the relationships between primary keys and foreign keys is known as maintaining the **referential integrity** of the database.

Ensuring Referential Integrity

- When the primary key of a table is changed or deleted and it is used as a foreign key in another table, the other table should be changed also.
- When a foreign key is changed or created, it should be validated against the primary key to which it refers.

SQL Statements: Select

- SQL is a programming language.
- SQL creates, maintains, and queries all these objects.
- The SELECT statement retrieves zero or more rows from one or more tables.
- You specify the columns you want, or * for all columns.
- You specify the table(s) in the WHERE clause.
- You can use a WHERE clause to restrict the rows that are retrieved.

Select Examples

```
select * from id
```

Retrieves all columns and all rows from the id table.

```
select name from id
```

Retrieves all names from the id table.

```
select name from id where number < 100
```

Retrieves all names from id table whose numbers are less than 100.

Using Like and Wildcards

- The ANSI Standard like predicate accepts the following wildcards:

% matches any number of characters (including none)

_ matches any single character

- Example:

```
select number, name, expiry_date  
from id where name like 'J%'
```

Functions and Operators

- Both MySQL and PostgreSQL have a rich set of operators and functions for processing and manipulating data
- For details, refer to the **MySQL 5.0 Reference Manual**, Chapter 11, and the **PostgreSQL 8.2.1 Documentation**, Chapter 9.

Datetime Processing

- You can find the current date and time from server's machine
- You can directly compare two datetime expressions
- You can convert a date to a string in a number of styles
- You can access the component parts of a date
- You can add to or subtract from a given date or time
- You can calculate the difference between two dates or times

Aggregate Functions

- You can use built-in aggregate functions to perform operations on the values of a column in all the returned rows:

count(*)	Number of selected rows
count(<i>column_name</i>)	Number of non-null values in the column
sum(<i>column_name</i>)	Value total
avg(<i>column_name</i>)	Average value
max(<i>column_name</i>)	Maximum value
min(<i>column_name</i>)	Minimum value

- You can mix aggregates and non-aggregates in the same select statement, but there are certain things to watch out for.
- Often, you will include the non-aggregates as part of a **GROUP BY** clause. For example:

```
SELECT student_name, MIN(test_score),  
       MAX(test_score)  
FROM student  
GROUP BY student_name;
```

Sorting Results

- An **ORDER BY** clause sorts the query results into ascending (**ASC**) or descending (**DESC**) order.
- You can sort by more than one column.
- Example:

```
select name, number  
from id  
order by name;
```

Select into

- You can create a new table from one or more existing tables.
- PostgreSQL (standard SQL):
`select column [, column ...]
into new-table
from existing-table [, table ...]
[where ...]`
- MySQL (unique to MySQL):
`insert into new-table (new-column [, column ...])
select existing-column [, column ...]
from existing-table`

Inserting New Data Into Existing Tables

- The insert statement creates a single new row in an existing table.
- There are two methods of specifying the data:
 - I. You can specify the columns to receive the data:

```
insert into existing-table ( col1, col2, ... )  
values ( 'val1', 'val2', ... )
```

- insert ... continued ...

2. Or you can provide the data in column order:

```
insert into existing-table  
values ( 'val1', 'val2', ... )
```

- If you use the second method, data must be provided for all columns in the correct order.

- An `insert` statement containing a `select` instead of values gets data from another table. In this way, a single insert statement can create more than one row.
- Example:

```
insert into ____  
select * from ____  
where _____
```

Update

- The update statement changes all or part of an existing row in a single table.
- A where clause will limit the update to selected rows.

Delete

- The delete statement is used to remove rows from a single table.
- An unqualified delete statement will delete ALL rows from a table.
- A where clause will limit the rows deleted.

Deleting vs. Dropping tables

- To delete an entire table quickly, without logging all the rows, use:
`truncate table table_name`
- This will get rid of the rows in the table, but now the table itself.
- To get rid of the data and the table, use:
`drop table table_name`

Access to the Server

- In order for a user to access the server and its databases, that user must have a server account.
- Only the Database Administrator (DBA) can grant server access.

Access to Databases

- Access to the server does not grant access to any database
- Only the Database Owner (DBO) or Database Administrator (DBA) can grant access to a database.

- Adding a user to a database:
- To remove a user from a database:

Finding out who you are

- To find out what your server user name is, run:
- Within each database you have access to, you have a username. To find out what that is, use

- Only the DBA can perform server-wide commands.
- Only the DBO can:
 - Add, drop, change users in the database
 - Grant/revoke permission to use certain commands

Controlling access to commands

- { grant | revoke } { create database ... } { to | from } user [, user ...]

Controlling access to objects

- { grant | revoke } { select | insert | delete | update | execute | all } on { db.table } { to | from } user [, user ...]
- Chronology determines resulting permission:
 - To exclude a small number of users, grant to public and the revoke from some users.
 - To exclude a small number of columns, grant the entire table, and then revoke the specific columns.

Notes On Permissions

- The object's owner has exclusive right to modify the object's schema.
- The owner controls the access of other users to the data.
- Until granted permission, no one but the owner has any access to an object -- not even DBO.

To keep things simple:

- Make the DBO the owner of all objects in a database.
- If you grant access to tables, make sure you deny access to columns you don't want others to see.

Views

- A view is an imaginary table created from existing tables.
- A view may contain data from more than one table.
- A view may conceal data in a table.
- A view may be used to:
 - Enhance security
 - Simplify queries
 - Isolate users from changes in the structure of the underlying tables

Creating A View

- Views are created with the `create view` statement
- Syntax:
`create view view_name`
`[(column_name [, column_name ...])]`
`as select_statement`
- If the column names are omitted, the columns in the view have the same names as the columns in the base tables.

- When you create a view, only the definition is stored.
- The system displays the data when the view is called.
- If the table upon which a view depends no longer exists, you will get an error when you call the view.

Backup and Restore

- Backups are needed to restore the database after a media failure or major mistake.
- After a media failure or major mistake, you can recover a database using the database dumps.
- Only the DBA or the DBO can dump and load; you can grant permission to do either.

Dumping the Database

- MySQL and PostgreSQL have different command line utilities (with manual pages) to dump a database, and different dump formats.
- MySQL: `mysqldump`
- PostgreSQL: `pg_dump`

Restoring The Database

- See the MySQL Reference Manual, Chapter 5, Section 5.9 for more information.
- PostgreSQL has a `pg_restore` utility to match `pg_dump`; refer to its manual page and to the PostgreSQL Documentation, Chapter 23, for more information.

Recipe For Creating a System

- Establish which databases are to be created
- Establish available disk resources
- Determine allocation of resources to databases and dumps
- Create databases
- Add accounts and users

- Transfer ownership of databases to new owners
- Backup the master database
- Set up operational procedures for recovery and backup
- Write defaults
- Create tables

- Load data into databases
- Backup the databases
- Write views, rules, stored procedures and triggers
- Backup the databases
- Test the system
- Backup the databases

For more information...

- See the links on the Lab #2 page:

[http://technology.niagarac.on.ca/courses/
ctec1731/lab2/](http://technology.niagarac.on.ca/courses/ctec1731/lab2/)