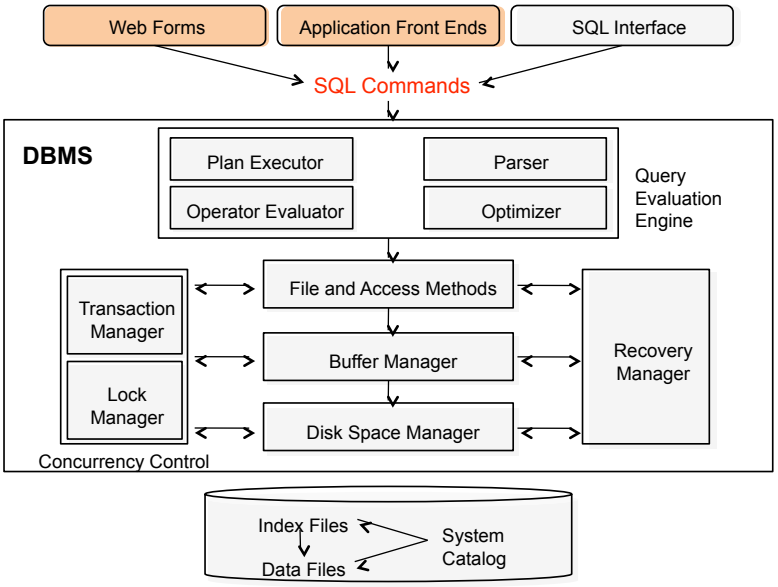
CPSC 421
Database Management Systems

Lecture 10:
Embedded SQL

* Some material adapted from R. Ramakrishnan, L. Delcambre, and B. Ludaescher

Basic Database Architecture



Embedded SQL

The majority of SQL is generated from software applications through "*Embedded SQL*"…

- Embedded SQL allows data from a DBMS to be accessed from within a regular software program
- Embedded SQL programs can
  - Convert/modify data before it is presented to users
  - Control what data is visible to users
  - Generate SQL dynamically based on user preferences

Embedded SQL

- SQL commands can be called from within a host language such as C/C++, .NET, PHP, Java, …
  - Typically provided through a set of libraries or modules
  - Includes some type of statements to connect to the right database
  - SQL statements can refer to *host variables* … the regular variables in your program (to pass parameters to queries)
- Because SQL (and the relational model) are often very different than the host language
  - The use of SQL in the language may not be "natural"
  - This means that it can be difficult to integrate records/relations with the data structures and constructs provided by the language
  - This is often referred to as the "*impedance mismatch*" problem

Embedded SQL

- SQL query results are often large (multi-) sets of records
- Most languages cannot efficiently (or practically) represent or hold typical query results
- To address this, SQL supports a mechanism called a "*cursor*" for accessing query results
  - Cursors are somewhat similar to standard IO stream APIs
  - As well as "iterators" …

Cursors

- Cursors can be declared on a relation or query statement (which generates a relation)
- Like an IO stream (or iterator) … we can
  - open a cursor
  - (repeatedly) fetch records from a cursor
  - move the cursor to a new location
  - determine if all tuples have been retrieved
  - modify or delete a tuple pointed to by a cursor

Cursors

- Capabilities vary from one DBMS to another
  - Move forward only
  - Move forward or backward one row at a time
  - Move to arbitrary locations
  - What are some advantages/disadvantages?
- Cursor placement
  - A cursor is placed BEFORE the first row of the result
  - Special "end of result" value (like EOF) is used to denote when past the last row

Embedded SQL implementations

- We will look at two different approaches for Embedded SQL
  - Using SQL in a scripting language (PHP)
  - An API-based approach for C (for mysql

Example Schema

- Suppose we want to track products and categories for a retailer

Products(ProductID : int, CategoryID : int, ProductName : string, UnitPrice : currency)

Categories(CategoryID : int, CategoryName : string)

Products.CategoryID REFERENCES Categories.CategoryID

- BTW, are these tables "normalized", i.e., in BCNF?

Embedded SQL in PHP

- Another (common) approach is to use a library (i.e., predefined set of function calls)
- PHP uses this approach
  - along with many other languages (including Java) …
  - Each DBMS provides a different library (e.g., we will look at the MySQL one; PostgreSQL has its own, etc.)

A Quick Note on PHP

- PHP is a general-purpose scripting language …
  - it is similar to other (procedural) scripting languages like Perl, tcsh, bash, python, etc.
  - it is primarily used as a scripting language for web development since it can be easily embedded with HTML
  - … and provides useful functions for building web applications
  - this makes it easy to create web applications that require access to back-end databases
  - there are lots of resources, tutorials, and examples for PHP on the web

Embedded SQL in PHP

- First we have to connect to a database

```
<?php
  $host = 'localhost';
  $user = 'bowers';
  $pw = 'secret';
  $db = 'products';
  mysql_connect($host, $user, $pw)
    or die(mysql_error());
  mysql_select_db($db);
?>
```

- All PHP scripts are enclosed in <?php … ?>

Embedded SQL in PHP

- Now that we are connected, we can query the DB

```
… connect to db …
<html>
  <body>
    <h2>Product names and prices</h2>
    <?php
      $sql_stmt = "SELECT P.ProductName, P.UnitPrice
                   FROM Products P, Categories C
                   WHERE P.CategoryName = 'Beverages' AND
                         P.CategoryID = C.CategoryID
                   ORDER BY P.UnitPrice";
      $results = mysql_query($sql_stmt)
                   or die('Invalid query: ' . mysql_error());
```

The $results variable is a handle to a cursor

Embedded SQL in PHP

- And retrieve and display the results of the query

```
        while($row = mysql_fetch_row($results)) {
          list($product_name, $unit_price) = $row;
          print "$product_name, $unit_price <br/>";
        }
        mysql_close();
      ?>
    </body>
  </html>
```

The $results variable is a handle to a cursor

Embedded SQL in PHP

A number of helper functions are provided …

```
# get the number of rows of a result
$num_rows = mysql_num_rows($result);
print "The query returned " . $num_rows . " results\n";
#  list the databases (for the current connection)
$databases = mysql_list_dbs();
$dbs = mysql_num_rows($databases);
for($i = 0; $i < $dbs; $i++)
  print mysql_db_name($databases, $i) . "\n";
# list the tables
$tables = mysql_list_tables($db);
$num_tables = mysql_num_rows($tables);
for($i = 0; $i < $num_tables; $i++)
  print mysql_tablename($tables, $i) . "\n";
# establish multiple connections
$connect1 = mysql_connect($hostname1, $user1, $pw1);
$connect2 = mysql_connect($hostname2, $user2, $pw2);
```

Embedded SQL in PHP

- For more info on PHP …
  - http://www.php.net
  - http://www.w3schools.com/PHP/
  - http://www.php.net/mysql/
  - Many other language bindings besides MySQL

  - Warning: I haven't tried using PHP w/ MySQL on ada!
  - Both are installed though …

Embedded MySQL in C

- MySQL provides a C API …
  - A different approach than using special preprocessing macros as before
  - Warning: I haven't tried this on ada

- Connecting to MySQL

```
#include "/usr/include/mysql/mysql.h"
int main() {
   MYSQL * mysql;
   MYSQL_RES * result;
   MYSQL_ROW row;
   mysql_init(mysql);
   mysql_real_connect(mysql, host, user,  password, db, 0, NULL, 0);
```

Port, Socket, Client Flag

Embedded MySQL in C

- Querying a MySQL database

```
    mysql_query(mysql, "SELECT * FROM …");
    result = mysql_store_result(mysql); // create cursor
    while(row = mysql_fetch_row(result)) {
       … print results as row[0], row[1], …
    }
    mysql_free_result(result);
    mysql_close(mysql);
    return 0;
}
```


Embedded MySQL in C

- Compiling …

```
gcc prog.c –I/usr/include/mysql –L/usr/lib/mysql
   -lmysqlclient -1m -1z
```

- The basic idea is to compile with the specific mysql libraries … as opposed to using the precompiler

Other Embedded SQL Solutions

- ODBC – Open Database Connectivity
  - Older standard, proposed by Microsoft but driven by the database community
  - A number of vendors (Oracle, etc.) make ODBC drivers available
- JDBC – Java Database Connectivity
  - Similar to ODBC but for Java
  - Also, many vendors provide drivers
  - Provides a single Java API for accessing any database that supports JDBC
  - This differs than, e.g., PHP, where each vendor has a different set of API calls (pg_connect, etc.)

Database Language Commands

- DDL
  - "Data Definition Language"
  - Schema-level commands
- DML
  - "Data Manipulation Language"
  - Row-level commands

DDL – Data Definition Language

- Create, edit, or delete database objects
  - Tables
  - Stored Procedures
  - Data Types
  - NOT ROWS!

- Drop table:

  DROP TABLE Patient;

- Create table:

  CREATE TABLE Patient (…);

DDL – Data Definition Language

- Alter table:

  ALTER TABLE Patient

  * Plus any of the following:

  [ADD COLUMN]

  [ALTER COLUMN]

  [DROP COLUMN]

  [ADD CONSTRAINT]

  [DROP CONSTRAINT]

DML – Data Manipulation Language

- Inserting, updating, or deleting rows
- Deleting rows:

```
DELETE FROM Patient
WHERE FirstName LIKE 'B%';
```

- Note: This will potentially delete multiple patients!

DML – Data Manipulation Language

- Inserting rows:

```
INSERT INTO Patient (ID, FirstName, LastName)
VALUES (4, 'Sue', 'Smith'), (6, 'John', 'Jones');

INSERT INTO Patient (ID, FirstName, DateOfBirth)
SELECT ID, Fname, DOB
FROM Transfers
WHERE Status = 1;
```

  - This will insert a patient for each row returned from the query

DML – Data Manipulation Language

- Updating rows:

```
UPDATE TABLE Patient
SET FirstName = 'Bob',
      DateOfBirth = AddDays(DateOfBirth, 1)
WHERE ID = 555;
```

  - This will change the FirstName and DateOfBirth for the patient with ID 555.