
Lecture 7: Database Application Development

Dragan Mirkovic
Department of Computer Science
University of Houston

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Announcements

- Today:
 - Database application development
 - Chapter 6 in Ramakrishnan & Gehrke
- Thursday 10/07/04
 - Quiz #4 E/R diagrams and E/R to relational database design
 - Ch. 2 in Garcia-Molina and 3.5 in Ramakrishnan.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Introduction

- Relational DBMS supports an interactive SQL interface
 - Works fine for simple applications
- Many applications require more flexibility
 - General purpose programming language + SQL data manipulation
- Solution
 - Application Programming Interface (API) for SQL
 - Embedded SQL

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Overview

Concepts covered in this lecture:

- SQL in application code
 - Embedded SQL
 - Data access with static SQL queries in application code
 - Cursors
 - Bridge the gap between SQL output and the host language
 - Dynamic SQL
 - Create queries at runtime
 - JDBC, SQLJ
 - Programming interfaces for java
 - Stored procedures

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

SQL in Application Code

- SQL commands can be called from within a host language (e.g., C++ or Java) program.
 - SQL statements can refer to **host variables** (including special variables used to return status).
 - Must include a statement to **connect** to the right database.
- Two main integration approaches:
 - Embed SQL in the host language (Embedded SQL, SQLJ)
 - Create special API to call SQL commands (JDBC)
- Impedance mismatch:
 - SQL relations are (multi-) sets of records with no fixed bound on the number of records.
 - No such data structure in languages such as C++.
 - SQL supports a mechanism called a *cursor* to handle this.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Embedded SQL

- Approach: Embed SQL in the host language.
 - A preprocessor converts the SQL statements into special API calls.
 - Then a regular compiler is used to compile the code.
- Language constructs:
 - Connecting to a database:
EXEC SQL CONNECT
 - Declaring variables:
EXEC SQL BEGIN (END) DECLARE SECTION
 - Statements:
EXEC SQL Statement;

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Embedded SQL: Variables

- Example:

```
EXEC SQL BEGIN DECLARE SECTION
char c_sname[20];
long c_sid;
short c_rating;
float c_age;
EXEC SQL END DECLARE SECTION
```

 - SQL standard defines the corresponding types
- Two special “error” variables:
 - SQLCODE (long, is negative if an error has occurred)
 - SQLSTATE (char[6], predefined codes for common errors)

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Cursors

- Mechanism for the set-of-records abstraction
- We can declare a cursor on a relation or query statement (which generates a relation).
- We can *open* a cursor, and repeatedly *fetch* a tuple then *move* the cursor, until all tuples have been retrieved.
 - We can use a special clause, called ORDER BY, in queries that are accessed through a cursor, to control the order in which tuples are returned.
 - Fields in ORDER BY clause must also appear in SELECT clause.
 - The ORDER BY clause, which orders answer tuples, is only allowed in the context of a cursor.
- Can also modify/delete tuple pointed to by a cursor.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Example

- Cursor that gets names of sailors who've reserved a red boat, in alphabetical order

```
EXEC SQL DECLARE sinfo CURSOR FOR
  SELECT S.sname
  FROM Sailors S, Boats B, Reserves R
  WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'
  ORDER BY S.sname
```

- Note that it is illegal to replace S.sname by, say, S.sid in the ORDER BY clause! (Why?)
- Can we add S.sid to the SELECT clause and replace S.sname by S.sid in the ORDER BY clause?

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Embedding SQL in C

- Example:

```
char SQLSTATE[6];
EXEC SQL BEGIN DECLARE SECTION
char c_sname[20]; short c_minrating; float c_age;
EXEC SQL END DECLARE SECTION
c_minrating = random();
EXEC SQL DECLARE sinfo CURSOR FOR
  SELECT S.sname, S.age FROM Sailors S
  WHERE S.rating > :c_minrating
  ORDER BY S.sname;
EXEC SQL OPEN sinfo;
do {
  EXEC SQL FETCH sinfo INTO :c_sname, :c_age;
  printf("%s is %d years old\n", c_sname, c_age);
} while (SQLSTATE != '02000');
EXEC SQL CLOSE sinfo;
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Properties of Cursors

- The general form of a cursor declaration
DECLARE *cursorname* [INSENSITIVE] [SCROLL] CURSOR
 [WITH HOLD]
 FOR *some query*
 [ORDER BY *order-item-list*]
 [FOR READ ONLY | FOR UPDATE]
 - INSENSITIVE makes the cursor insensitive to changes after the cursor has been opened. (otherwise implementation dependent)
 - SCROLL makes the cursor scrollable.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Dynamic SQL

- SQL query strings are now always known at compile time (e.g., spreadsheet, graphical DBMS frontend): Allow construction of SQL statements on-the-fly
- Example:

```
char c_sqlstring[]=
    {"DELETE FROM Sailors WHERE rating>5"};
EXEC SQL PREPARE readytogo FROM :c_sqlstring;
EXEC SQL EXECUTE readytogo;
```
- Requires output from the SQL interpreter at run-time and can generate a considerable overhead.
- Embedded commands are prepared once at compile-time and can be executed many times

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Database APIs

- Alternative to embedding
- Rather than modify compiler, add library with database calls (API)
- Special standardized interface: procedures/objects
- Pass SQL strings from language, presents result sets in a language-friendly way
- Sun's JDBC: Java API
- Supposedly DBMS-neutral
 - a “driver” traps the calls and translates them into DBMS-specific code
 - database can be across a network

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

JDBC: Architecture

- Four architectural components:
 - Application (initiates and terminates connections, submits SQL statements)
 - Driver manager (load JDBC driver, passes JDBC function calls between application and drivers)
 - Driver (connects to data source, transmits requests and returns/translates results and error codes)
 - Data source (processes SQL statements and returns results)

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

JDBC Architecture (Contd.)

- Four types of drivers:
- Bridge:
 - Translates SQL commands into non-native API.
Example: JDBC-ODBC bridge. Code for ODBC and JDBC driver needs to be available on each client.
- Direct translation to native API, non-Java driver:
 - Translates SQL commands to native API of data source. Need OS-specific binary on each client.
- Network bridge:
 - Send commands over the network to a middleware server that talks to the data source. Needs only small JDBC driver at each client.
- Direction translation to native API via Java driver:
 - Converts JDBC calls directly to network protocol used by DBMS. Needs DBMS-specific Java driver at each client.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

JDBC Classes and Interfaces

- JDBC = collection of Java classes and interfaces
 - `java.sql` package
- Steps to submit a database query:
 - Load the JDBC driver
 - Connect to the data source
 - Execute SQL statements

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

JDBC Driver Management

- All drivers are managed by the DriverManager class
- Loading a JDBC driver:
 - In the Java code:
`Class.forName("oracle/jdbc.driver.OracleDriver");`
 - When starting the Java application:
`-Djdbc.drivers=oracle/jdbc.driver`

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Connections in JDBC

- We interact with a data source through sessions. Each connection identifies a logical session.
- JDBC URL: specifies the connection
`jdbc:<subprotocol>:<otherParameters>`
- Example:

```
String url="jdbc:oracle:www.bookstore.com:3083";
Connection con;
try{
    con = DriverManager.getConnection(url,userId,password);
} catch SQLException excpt {
    System.out.println(excpt.getMessage());
    return;
}
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Connection Class Interface

- `public int getTransactionIsolation()` and `void setTransactionIsolation(int level)`
Sets isolation level for the current connection.
- `public boolean getReadOnly()` and `void setReadOnly(boolean b)`
Specifies whether transactions in this connection are read-only
- `public boolean getAutoCommit()` and `void setAutoCommit(boolean b)`
If autocommit is set, then each SQL statement is considered its own transaction. Otherwise, a transaction is committed using `commit()`, or aborted using `rollback()`.
- `public boolean isClosed()`
Checks whether connection is still open.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Executing SQL Statements

- Three different ways of executing SQL statements:
 - `Statement` (base class, both static and dynamic SQL statements)
 - `PreparedStatement` (semi-static SQL statements)
 - `CallableStatement` (stored procedures)
- `PreparedStatement` class:
Precompiled, parametrized SQL statements:
 - Structure is fixed
 - Values of parameters are determined at run-time (uses '?' for values of the parameters)
 - Different ways of execution:
 - `executeUpdate()` - no records returned
 - `executeQuery()` - returns a `ResultSet` object (a cursor)

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Executing SQL Statements (Contd.)

```
String sql="INSERT INTO Sailors VALUES(?,?,?,?)";
PreparedStatement pstmt=con.prepareStatement(sql);
pstmt.clearParameters();
pstmt.setInt(1,sid);
pstmt.setString(2,sname);
pstmt.setInt(3, rating);
pstmt.setFloat(4,age);
// we know that no rows are returned, thus we use
executeUpdate()
int numRows = pstmt.executeUpdate();
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

ResultSets

- PreparedStatement.executeUpdate only returns the number of affected records
- PreparedStatement.executeQuery returns data, encapsulated in a ResultSet object (a cursor)

```
ResultSet rs=pstmt.executeQuery(sql);
// rs is now a cursor
While (rs.next()) {
    // process the data
}
```

A ResultSet is a very powerful cursor:

- previous(): moves one row back
- absolute(int num): moves to the row with the specified number
- relative (int num): moves forward or backward
- first() and last()

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Matching Java and SQL Data Types

SQL Type	Java class	ResultSet get method
BIT	Boolean	getBoolean()
CHAR	String	getString()
VARCHAR	String	getString()
DOUBLE	Double	getDouble()
FLOAT	Double	getDouble()
INTEGER	Integer	getInt()
REAL	Double	getFloat()
DATE	java.sql.Date	getDate()
TIME	java.sql.Time	getTime()
TIMESTAMP	java.sql.TimeStamp	getTimeStamp()

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

JDBC: Exceptions and Warnings

- Most of java.sql can throw and SQLException if an error occurs.
- SQLWarning is a subclass of SQLException; not as severe (they are not thrown and their existence has to be explicitly tested)

```
try {
    stmt=con.createStatement();
    warning=con.getWarnings();
    while(warning != null) {
        // handle SQLWarnings;
        warning =
        warning.getNextWarning();
    }
    con.clearWarnings();
    stmt.executeUpdate(queryString);
    warning = con.getWarnings();
    ...
} //end try
catch( SQLException SQLe) {
    // handle the exception
}
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Examining Database Metadata

- DatabaseMetaData object gives information about the database system and the catalog.
- Example name of the driver and version:

```
DatabaseMetaData md = con.getMetaData();
// print information about the driver:
System.out.println(
    "Name:" + md.getDriverName() +
    "version: " + md.getDriverVersion());
```
- Many different methods supported:
 - getCatalogs() – returns a ResultSet with info about the catalog relations

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Database Metadata (Contd.)

```
DatabaseMetaData md=con.getMetaData();
ResultSet trs=md.getTables(null,null,null,null);
String tableName;
While(trs.next()) {
    tableName = trs.getString("TABLE_NAME");
    System.out.println("Table: " + tableName);
    //print all attributes
    ResultSet crs = md.getColumns(null,null,tableName, null);
    while (crs.next()) {
        System.out.println(crs.getString("COLUMN_NAME" + ", ");
    }
}
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

A (Semi-)Complete Example

```
Connection con = // connect
    DriverManager.getConnection(url, "login", "pass");
Statement stmt = con.createStatement(); // set up stmt
String query = "SELECT name, rating FROM Sailors";
ResultSet rs = stmt.executeQuery(query);
try { // handle exceptions
    // loop through result tuples
    while (rs.next()) {
        String s = rs.getString("name");
        Int n = rs.getFloat("rating");
        System.out.println(s + " " + n);
    }
} catch (SQLException ex) {
    System.out.println(ex.getMessage ()
        + ex.getSQLState () + ex.getErrorCode ());
}
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

SQLJ

- Complements JDBC with a (semi-)static query model:
 - Compiler can perform syntax checks, strong type checks, consistency of the query with the schema
 - All arguments always bound to the same variable:

```
#sql = {
    SELECT name, rating INTO :name, :rating
    FROM Books WHERE sid = :sid;
```
 - Compare to JDBC:

```
sid=rs.getInt(1);
if (sid==1) {sname=rs.getString(2);}
else { sname2=rs.getString(2);}
```
- SQLJ (part of the SQL standard) versus embedded SQL (vendor-specific)
- Query results are stored in an **iterator** object

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

SQLJ Code

- Example: 5 steps in usage of an iterator

```
Int sid; String name; Int rating;
// 1. Declare Iterator Class
#sql iterator Sailors(Int sid, String name, Int rating);
// 2. Instantiate the Iterator
Sailors sailors;
// 3. Initialize the Iterator Using a SQL Statement
#sailors = {
    SELECT sid, sname INTO :sid, :name
    FROM Sailors WHERE rating = :rating
};
// 4. Iteratively Read the Rows from the Iterator Objects
while (sailors.next()) {
    System.out.println(sailors.sid + " " + sailors.sname);
}
// 5. Close the Iterator Object
sailors.close();
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

SQLJ Iterators

- Two types of iterators (“cursors”):
- Named iterator
 - Needs both variable type and name and allows retrieval of columns by name (See example on previous slide)
- Positional iterator
 - Need only variable type, and then uses FETCH .. INTO construct:

```
#sql iterator Sailors(Int, String, Int);
Sailors sailors;
#sailors = ...
while (true) {
    #sql {FETCH :sailors INTO :sid, :name};
    if (sailors.endFetch()) { break; }
    // process the sailor
}
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Stored Procedures

- What is a stored procedure:
 - Program executed through a single SQL statement
 - Executed in the process space of the server
- Advantages:
 - Can encapsulate application logic while staying “close” to the data
 - Reuse of application logic by different users
 - Avoid tuple-at-a-time return of records through cursors

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Stored Procedures: Examples

```
CREATE PROCEDURE ShowNumReservations
  SELECT S.sid, S.sname, COUNT(*)
  FROM Sailors S, Reserves R
  WHERE S.sid = R.sid
  GROUP BY S.sid, S.sname
```

Stored procedures can have [parameters](#):

- Three different modes: IN, OUT, INOUT

```
CREATE PROCEDURE IncreaseRating(
  IN sailor_sid INTEGER, IN increase INTEGER)
UPDATE Sailors
  SET rating = rating + increase
  WHERE sid = sailor_sid
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Examples (Contd.)

Stored procedure do not have to be written in SQL:

```
CREATE PROCEDURE TopSailors(  
    IN num INTEGER)  
LANGUAGE JAVA  
EXTERNAL NAME  
    "file:///c:/storedProcs/rank.jar"
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Calling Stored Procedures

```
EXEC SQL BEGIN DECLARE SECTION  
Int sid;  
Int rating;  
EXEC SQL END DECLARE SECTION  
  
// now increase the rating of this sailor  
EXEC CALL IncreaseRating(:sid,:rating);
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Calling Stored Procedures (Contd.)

JDBC:

```
CallableStatement cstmt=  
    con.prepareCall("{call  
        ShowSailors}");  
ResultSet rs =  
    cstmt.executeQuery();  
while (rs.next()) {  
    ...  
}
```

SQLJ:

```
#sql iterator  
    ShowSailors(...);  
ShowSailors showsailors;  
#sql showsailors={CALL  
    ShowSailors};  
while (showsailors.next()) {  
    ...  
}
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

SQL/PSM

Most DBMSs allow users to write stored procedures in a simple, general-purpose language (close to SQL) → SQL/PSM standard is a representative

Declare a stored procedure:

```
CREATE PROCEDURE name(p1, p2, ..., pn)  
    local variable declarations  
    procedure code;  
;
```

Declare a function:

```
CREATE FUNCTION name (p1, ..., pn) RETURNS  
    sqlDataType  
    local variable declarations  
    function code;  
;
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Main SQL/PSM Constructs

```
CREATE FUNCTION rate Sailor
  (IN sailorId INTEGER)
  RETURNS INTEGER
DECLARE rating INTEGER
DECLARE numRes INTEGER
SET numRes = (SELECT COUNT(*)
              FROM Reserves R
              WHERE R.sid = sailorId)
IF (numRes > 10) THEN rating =1;
ELSE rating = 0;
END IF;
RETURN rating;
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Main SQL/PSM Constructs (Contd.)

- Local variables (DECLARE)
- RETURN values for FUNCTION
- Assign variables with SET
- Branches and loops:
 - IF (condition) THEN statements;
 ELSEIF (condition) statements;
 ... ELSE statements; END IF;
 - LOOP statements; END LOOP
- Queries can be parts of expressions
- Can use cursors naturally without “EXEC SQL”

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

SQL Extensions: PL/SQL

- All major DBMS provide ways for writing stored procedures
 - Procedural extensions
 - Vendor specific:
 - Example:
 - PL/SQL is Oracle's procedural extension to SQL.
 - Server-side, stored procedural language.
 - Needs no explicit installation or licensing.
 - It is an implicit part of the Oracle Database.
 - References:
 - http://otn.oracle.com/tech/pl_sql/index.html

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Introduction to PL/SQL

- PL/SQL stands for Procedural Language/SQL.
- PL/SQL extends SQL by adding constructs found in procedural languages, resulting in a structural language that is more powerful than SQL.
- The basic unit in PL/SQL is a *block*.
 - All PL/SQL programs are made up of blocks
 - Blocks can be nested within each other.
- Typically, each block performs a logical action in the program.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

PL/SQL Block Structure

- A PL/SQL block has the following structure:

DECLARE

/* Declarative section:

variables, types, and local subprograms. */

BEGIN

/* Executable section:

- procedural and SQL statements go here. */

/* This is the only section of the block that is required. */

EXCEPTION

/* Exception handling section: error handling statements go here. */

END;

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

PL/SQL Structure

- The only SQL statements allowed in a PL/SQL program are
 - SELECT,
 - INSERT,
 - UPDATE,
 - DELETE
 - and several other data manipulation statements plus some transaction control.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

PL/SQL Execution

- To execute a PL/SQL program, we must follow the program text itself by
 - A line with a single dot ("."), and then
 - A line with run;
- As with Oracle SQL programs, we can invoke a PL/SQL program either by
 - typing it in sqlplus
 - putting the code in a file and invoking the file

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

PL/SQL: Variables and Types

- Information is transmitted between a PL/SQL program and the database through *variables*.
- Every variable has a specific type associated with it. That type can be
 - One of the types used by SQL for database columns
 - A generic type used in PL/SQL such as NUMBER declared to be the same as the type of some database column
 - The most commonly used generic type is NUMBER. Variables of type NUMBER can hold either an integer or a real number.
 - The most commonly used character string type is VARCHAR(*n*), where *n* is the maximum length of the string in bytes.
 - This length is required, and there is no default. For example, we might declare:

```
DECLARE
  price NUMBER;
  myBeer VARCHAR(20);
```
 - Note that PL/SQL allows BOOLEAN variables, even though Oracle does not support BOOLEAN as a type for database columns.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

PL/SQL Example

- **SQL:**

```
CREATE TABLE T1(  
  e INTEGER,  
  f INTEGER );  
  
DELETE FROM T1;  
  
INSERT INTO T1 VALUES(1, 3);  
  
INSERT INTO T1 VALUES(2, 4);
```
- **PL/SQL:**

```
DECLARE  
  a NUMBER;  
  b NUMBER;  
  
BEGIN  
  SELECT e,f INTO a,b  
  FROM T1  
  WHERE e>1;  
  INSERT INTO T1 VALUES(b,a);  
END;  
  
run;
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Summary

- Embedded SQL allows execution of parametrized static queries within a host language
- Dynamic SQL allows execution of completely ad-hoc queries within a host language
- Cursor mechanism allows retrieval of one record at a time and bridges impedance mismatch between host language and SQL
- APIs such as JDBC introduce a layer of abstraction between application and DBMS
- SQLJ: Allows embedded SQL in Java. Static model, queries checked at compile-time.
- Stored procedures execute application logic directly at the server and can be called from SQL
- SQL/PSM standard for writing stored procedures

D. Mirkovic, COSC 3480: Design of File and Database Systems, Fall 2004

Homework

- Read Ch 6. In Ramakrishnan & Gehrke