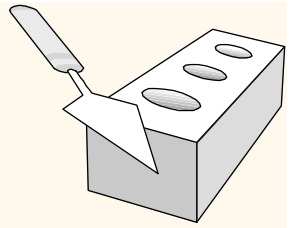
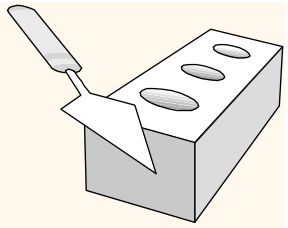


Database Application Development

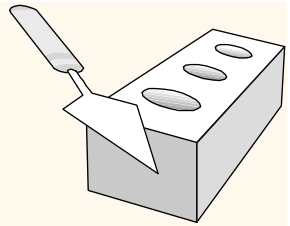
Chapter 6



Overview

Concepts covered in this lecture:

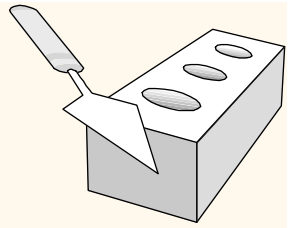
- SQL in application code
- Embedded SQL
- Cursors
- Dynamic SQL
- Stored procedures



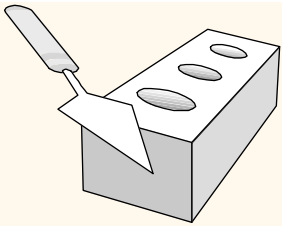
Introduction

- So far:
  - interactive SQL interface,
  - pure "SQL programs".
- In practice often:
  - queries are not ad-hoc, but programmed once and executed repeatedly,
  - need the greater flexibility of a general-purpose programming language, especially for complex calculations (e.g. recursive functions) and graphic user interfaces.
    - SQL statements part of a larger software system

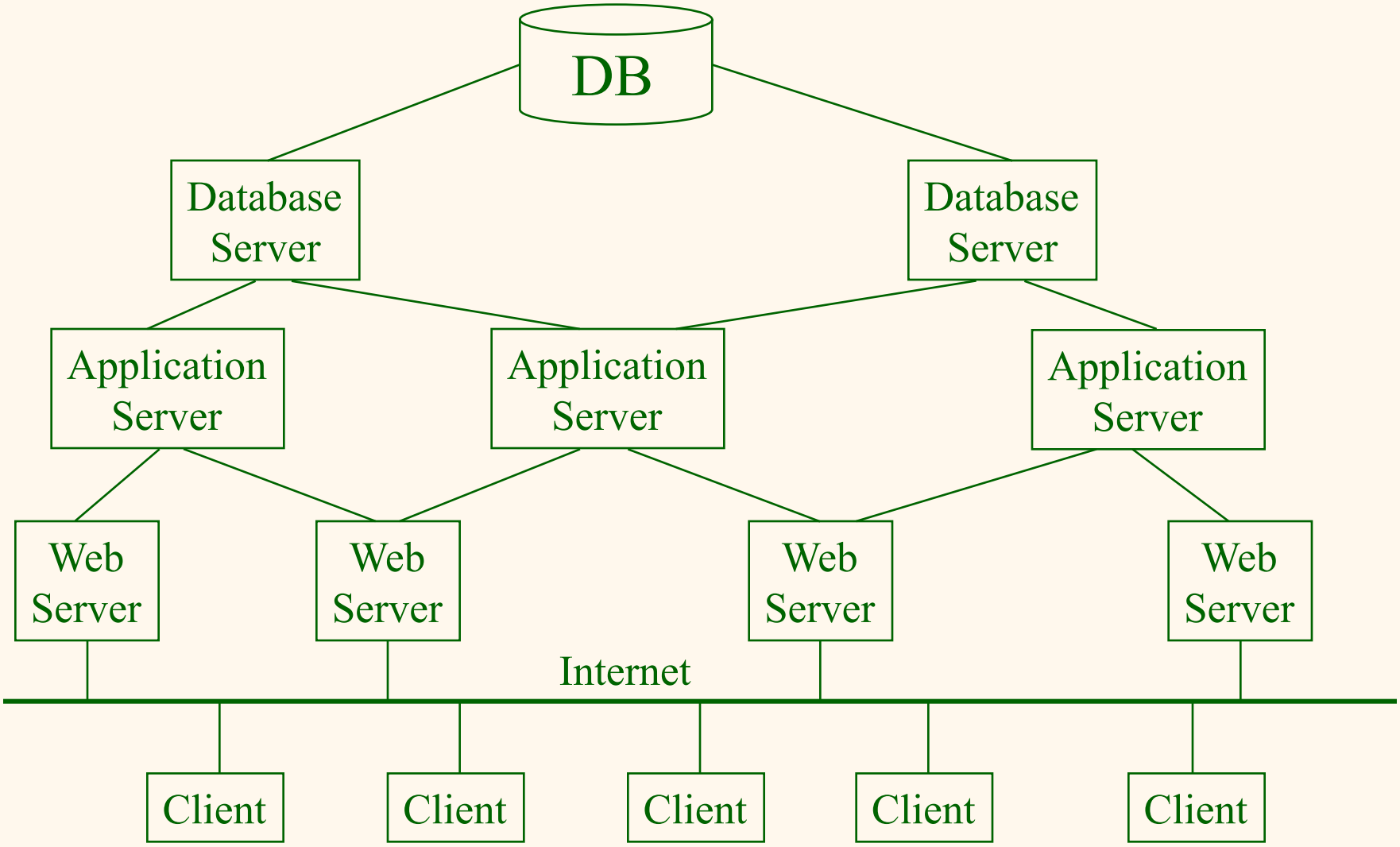
The Three-Tier Architecture



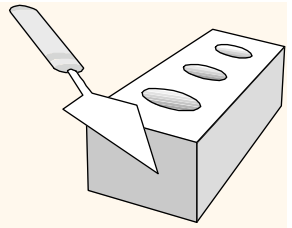
- The following *three-tier architecture* is common for database installations:
  - *Web servers* connect clients to the DBS, typically over the Internet (*web-server tier*).
  - *Applications servers* perform the "business logic" requested by the webserves, supported by the database servers (*application tier*).
  - *Database servers* execute queries and modifications of the database for the application servers (*database tier*).



The Three-Tier Architecture

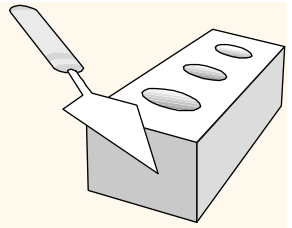


Key Questions

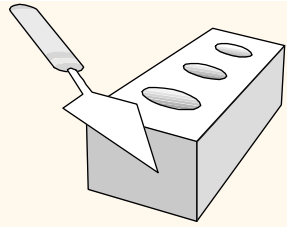


- How do we send SQL commands to a database management system from within an application program?
- How do we get the answer back in a way that can be processed by the application program?
- Rather than extending a programming language with SQL capability, how about extending SQL with programming capabilities?

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, Visual Studio).

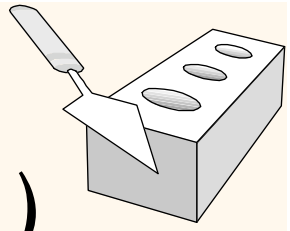


Overview

	Static Queries: Query form known at compile time	Dynamic Queries
Execution in Application Space	Embedded SQL SQLJ	API: Dynamic SQL ODBC, JDBC
Server Execution	Stored Procedure SQL/PSM	

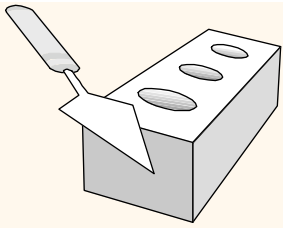
Could also have dynamic stored procedures but we won't discuss it.

SQL in Application Code (Contd.)



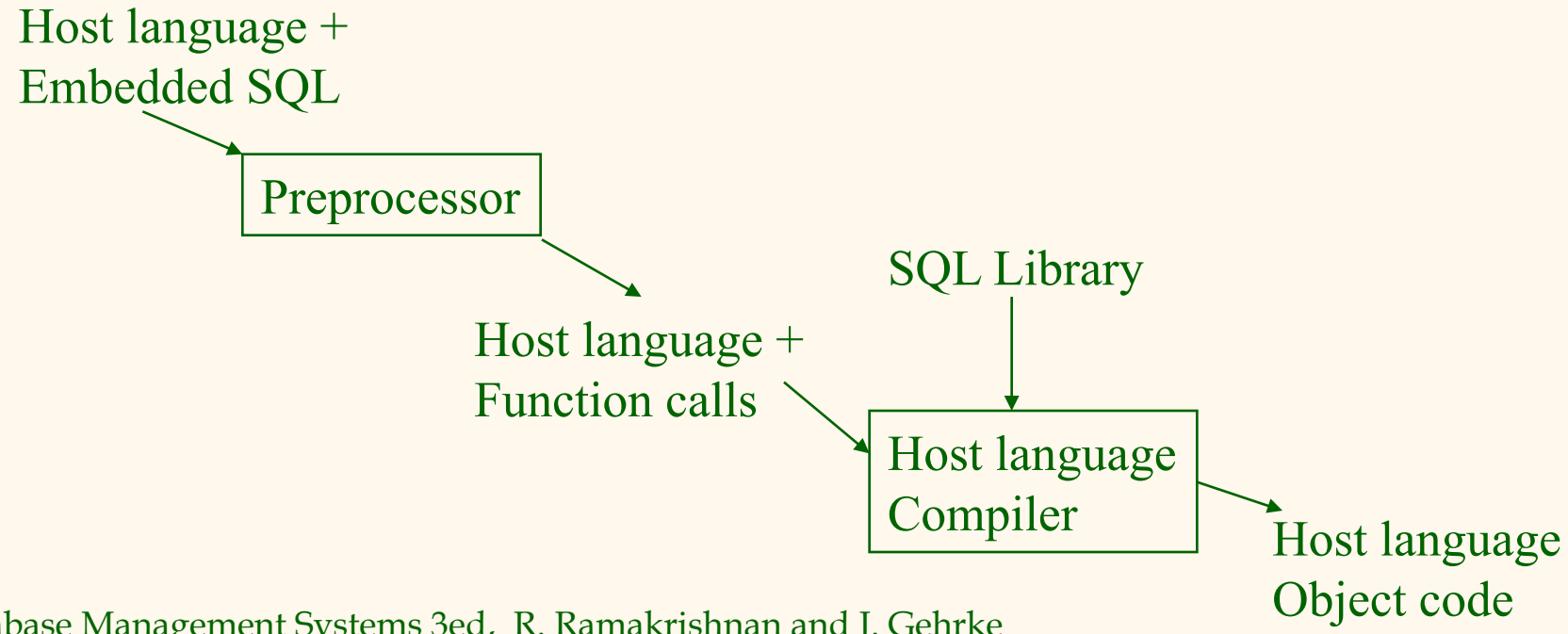
Impedance mismatch:

- SQL relations are (multi-) sets of records, with no *a priori* bound on the number of records. No such data structure exist traditionally in procedural programming languages such as C++.
  - SQL supports a mechanism called a *cursor* to handle this.

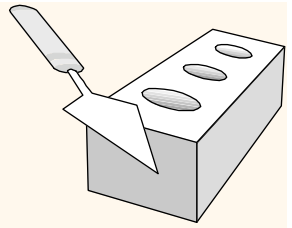


Embedded SQL

- Approach: Embed SQL in the host language.
  - A *preprocessor* converts the SQL statements into special API calls for a database system.
  - Then a regular compiler is used to compile the code.

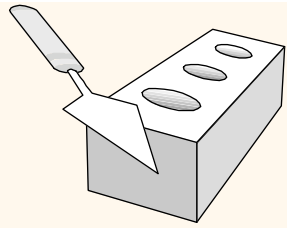


Embedded SQL



- Embedded SQL constructs:
  - Connecting to a database:
    EXEC SQL CONNECT
  - Declaring shared variables:
    EXEC SQL BEGIN (END) DECLARE SECTION
  - SQL Statements:
    EXEC SQL Statement;
    all statements except queries can be directly embedded
  - Declaring and manipulating cursors
    for embedding SQL queries

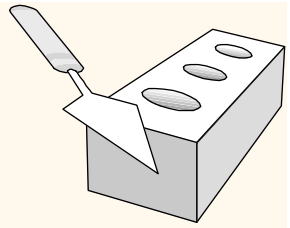
Embedded SQL: Variables



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

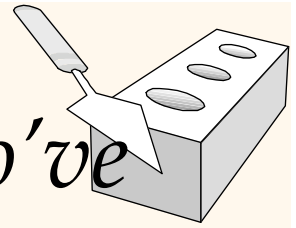
- Two special "error" variables:
  - SQLCODE (long, is negative if an error has occurred)
  - SQLSTATE (char[6], predefined codes for common errors)

Cursors



- Can declare a cursor on a relation or query statement (which generates a relation).
- Can *open* a cursor, and repeatedly *fetch* a tuple then *move* the cursor, until all tuples have been retrieved.
  - 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.
- Can also modify/delete tuple pointed to by a cursor.

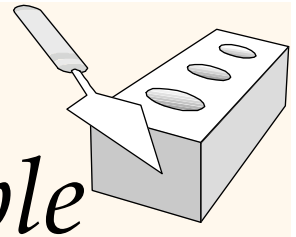
Cursor that gets names of sailors who've reserved a red boat, in alphabetical order



```
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
```

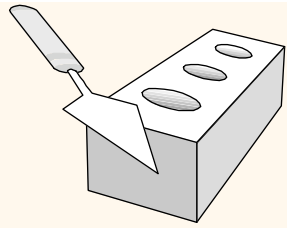
- Note that it is illegal to replace *S.sname* by, say, *S.sid* in the ORDER BY clause! (Why?)
- Can we add *S.sid* to the SELECT clause and replace *S.sname* by *S.sid* in the ORDER BY clause?

Embedding SQL in C: An Example



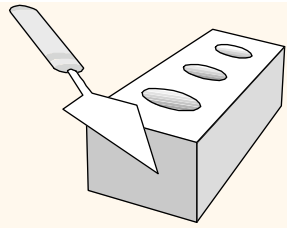
```
char SQLSTATE[6];
EXEC SQL BEGIN DECLARE SECTION
char c_sname[20]; short c_minrating; float c_age;
EXEC SQL END DECLARE SECTION
c_minrating = random();
EXEC SQL DECLARE sinfo CURSOR FOR
    SELECT S.sname, S.age      FROM Sailors S
    WHERE S.rating > :c_minrating
    ORDER BY S.sname;
EXEC SQL 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;
```

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.
  - Source code **and** executable is independent of DBMS.

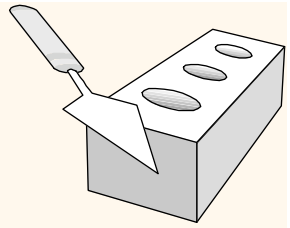


Dynamic SQL

- Often, the concrete SQL statement is known not at compile time, but only at runtime.
  - Example 1: a program prompts user for parameters of SQL query, reads the parameters and executes query.
  - Example 2: a program prompts user for an SQL query, reads and executes it.

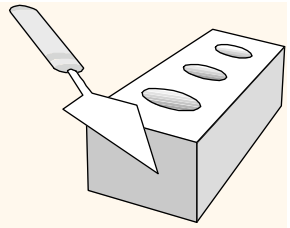
- Construction of SQL statements on-the-fly:
  PREPARE: parse and compile SQL command.
  EXECUTE: execute command.

Dynamic SQL: Example



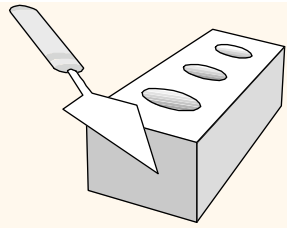
```
char c_sqlstring[]=
    {"DELETE FROM Sailors WHERE rating > 5"};
EXEC SQL PREPARE readytogo FROM :c_sqlstring;
EXEC SQL EXECUTE readytogo;
```

JDBC: Architecture



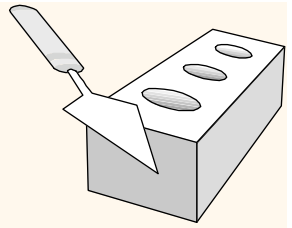
- Four architectural components:
  - Application (initiates and terminates connections, submits SQL statements)
  - Driver manager (load JDBC driver)
  - Driver (connects to data source, transmits requests and returns/translates results and error codes)
  - Data source (processes SQL statements)

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

Connections in JDBC

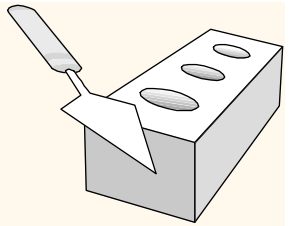


We interact with a data source through sessions. Each connection identifies a logical session.

- JDBC URL:
  jdbc:<subprotocol>:<otherParameters>

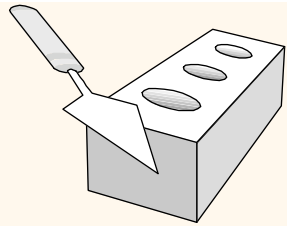
Example:

```
String url="jdbc:oracle:www.bookstore.com:3083";
Connection con;
try{
    con = DriverManager.getConnection(url,usedId,password);
} catch SQLException excpt { …}
```



Connection Class Interface

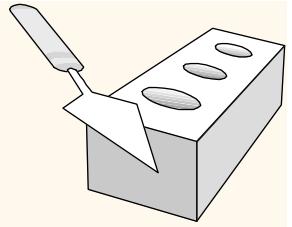
- public boolean getReadOnly() and
  void setReadOnly(boolean b)
  Specifies whether transactions in this connection are read-only
- public boolean getAutoCommit() and
  void setAutoCommit(boolean b)
  If autocommit is set, then each SQL statement is considered its own transaction. Otherwise, a transaction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



Connection Class Interface

- 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.

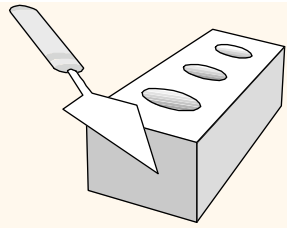
ResultSets



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()
- RecordSet, DataReader in Visual Basic

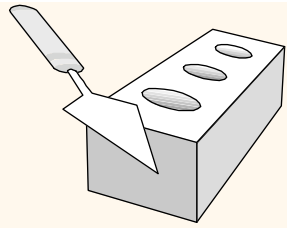
Call ResultSets



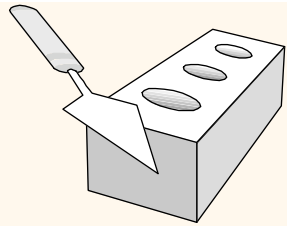
- PreparedStatement.executeUpdate only returns the number of affected records
- PreparedStatement.executeQuery returns data, encapsulated in a ResultSet object (a cursor)

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

A (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 ());
}
```

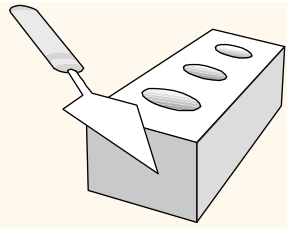


Visual Studio Example

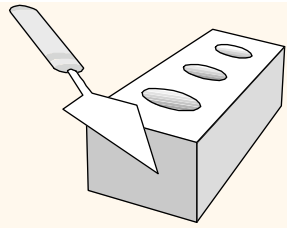
Visual Studio Connection Example

see course website.

Stored Procedures

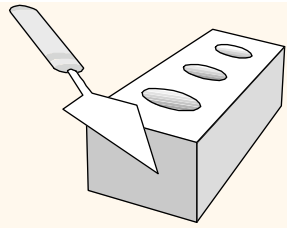


- What is a stored procedure:
  - Program executed through a single SQL statement
  - Executed in the process space of the server
- Advantages:
  - Can encapsulate application logic while staying "close" to the data
  - Reuse of application logic by different users
  - Avoid tuple-at-a-time return of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



Stored Procedures

- A *stored procedure* is a function / procedure written in a general-purpose programming language that is executed within the DBS.
- Allows to perform computations that cannot be expressed in SQL.
- Procedure executed through a single SQL statement.
- Executed in the process space of the DB server.
- SQL standard: *PSM* (Persistent Stored Modules). Extends SQL by basic concepts of a general-purpose programming language.



Stored Procedures: Examples

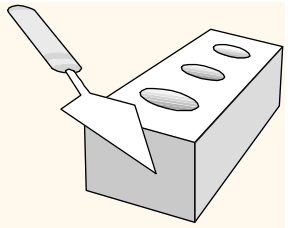
```
CREATE PROCEDURE ShowNumReservations
    SELECT S.sid, S.sname, COUNT(*)
    FROM Sailors S, Reserves R
    WHERE S.sid = R.sid
    GROUP BY S.sid, S.sname
```

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
```

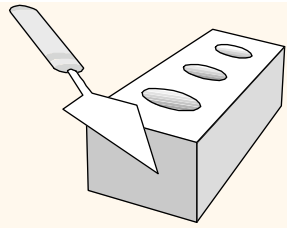
Stored Procedures: Examples (Contd.)



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”
```

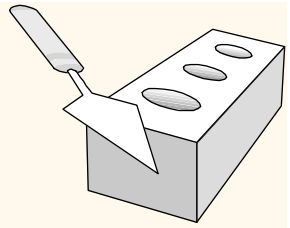
Calling Stored Procedures



```
EXEC SQL BEGIN DECLARE SECTION
Int sid;
Int rating;
EXEC SQL END DECLARE SECTION

// now increase the rating of this sailor
EXEC SQL CALL IncreaseRating(:sid,:rating);
```

SQL/PSM



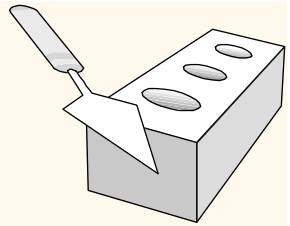
Most DBMSs allow users to write stored procedures in a simple, general-purpose language (close to SQL) → SQL/PSM standard is a representative

Declare a stored procedure:

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

Declare a function:

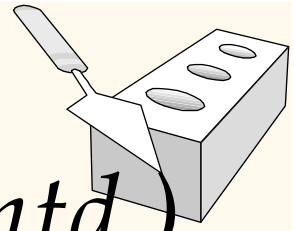
CREATE FUNCTION name (p1, …, pn) RETURNS sqlDataType
    local variable declarations
    function code;



Main SQL/PSM Constructs

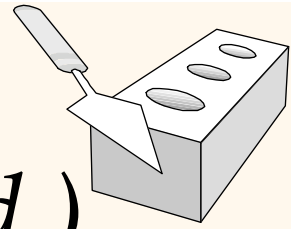
```
CREATE FUNCTION rate Sailor
        (IN sailorId INTEGER)
        RETURNS INTEGER
DECLARE rating INTEGER
DECLARE numRes INTEGER
SET numRes = (SELECT COUNT(*)
                      FROM Reserves R
                       WHERE R.sid = sailorId)
IF (numRes > 10) THEN rating =1;
ELSE rating = 0;
END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

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 without "EXEC SQL"

Calling Stored Procedures (Contd.)

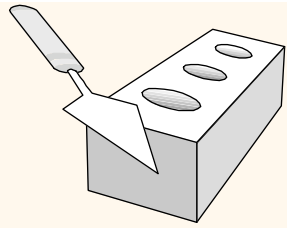


JDBC:

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

SQLJ:

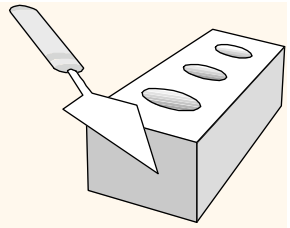
```
#sql iterator ShowSailors
    (…);
ShowSailors showsailors;
#sql showsailors={CALL
    ShowSailors};
while (showsailors.next()) {
    …
}
```



Summary

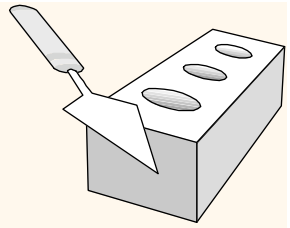
- Embedded SQL allows execution of parametrized static queries within a host language
- Dynamic SQL allows execution of completely ad-hoc queries within a host language
- Cursor mechanism allows retrieval of one record at a time and bridges **impedance mismatch** between host language and SQL
- APIs such as JDBC introduce a layer of abstraction between application and DBMS

Summary (Contd.)



- Stored procedures execute application logic directly at the server
- SQL/PSM standard for writing stored procedures

Midterm News



- Answer Key will be emailed today.
- Grades probably released today too.
- Grades are out of **70** points total.
- You can visit your midterm in office hours on Monday.