Database Management Systems

Database Applications

Motivation

- SQL is not a Turing complete language (on purpose)
- Solution: Use SQL in conjunction with a general-purpose programming language, called host language
- Question: How can it be done?
  - Embedded SQL
  - Library calls (CLI – Call Level Interface)
  - ODBC/JDBC (still use library calls, but ...)

The Three-Tier Architecture

- Databases can run as small, standalone programs.
- However, when a large database is used , a very common architecture is called three tier/layer one.
  - Web-Server Tier: web server processes manage the interactions with users, collecting requests and presenting the responses.
  - Application Tier: invoked by web-server process to perform the business logic, deciding what kind of data to retrieve in response to the request, forming the proper query
  - Database Tier: execute the queries, return the data

The SQL Environment

- A SQL environment is the framework under which data may exist and SQL operations on data may be executed.

- It contains a collection of catalogs, schemas, etc and most importantly, two special kinds of processes: SQL clients and SQL servers.

- Client and server processes may run on the same machine.

- In order to run a database application program at a host where a SQL client exists, a connection between the client and the server must be opened.

- A connection is like a communication channel between the client and the server.

- Exactly how to establish connection depends on the DBMS and the program method.

- client — request sender (runs the application)

- server — receiver of request (runs the SQL query)

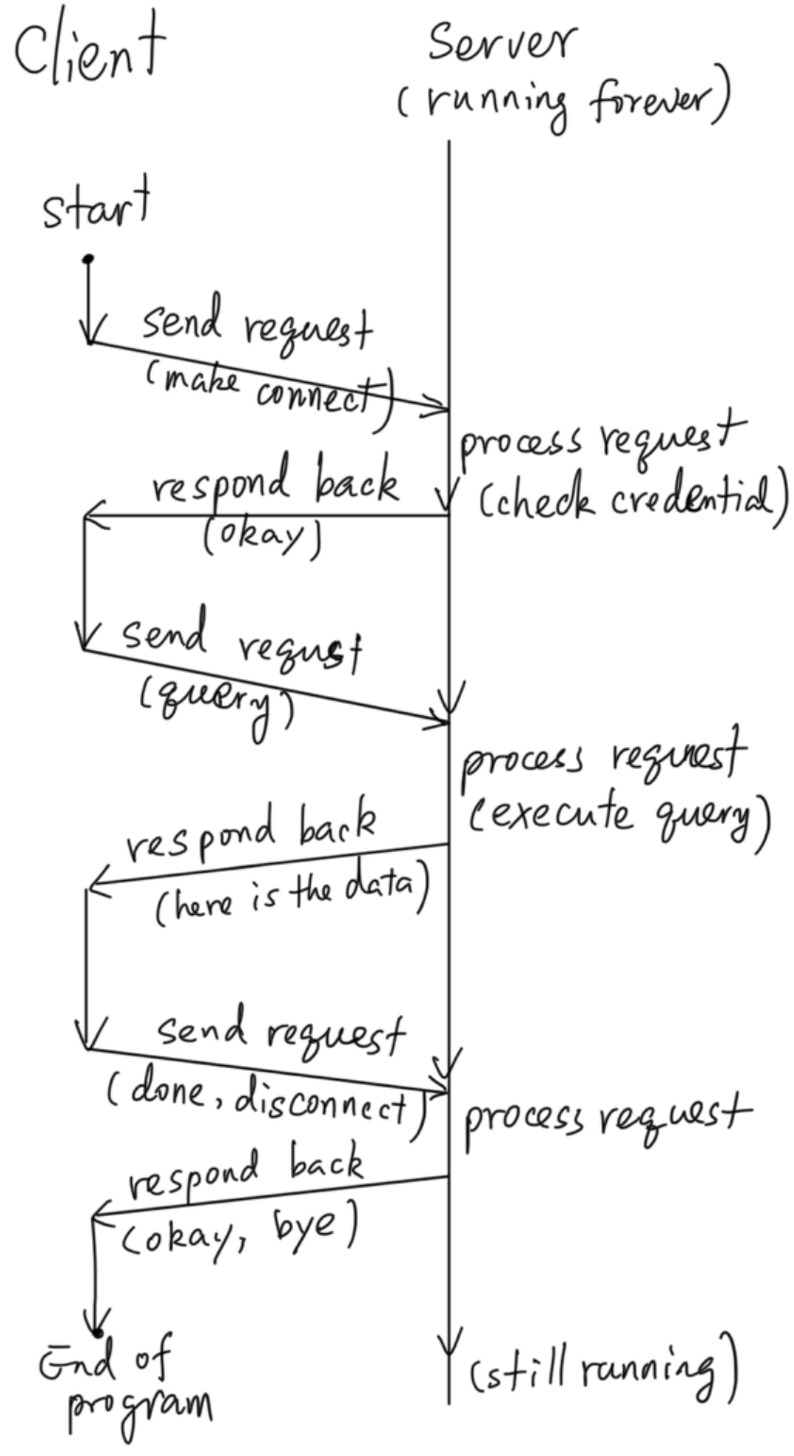
Client-side Application Structure

- various declarations
- establish connection to database server
- submit SQL queries to server according to the need
- receive results (if any) back
- do some further processing of the results (if needed) or process errors
- commit/rollback
- disconnect from database server

Server-side Process

- Start the server process
- waiting for client’s requests
- response to the client’s requests

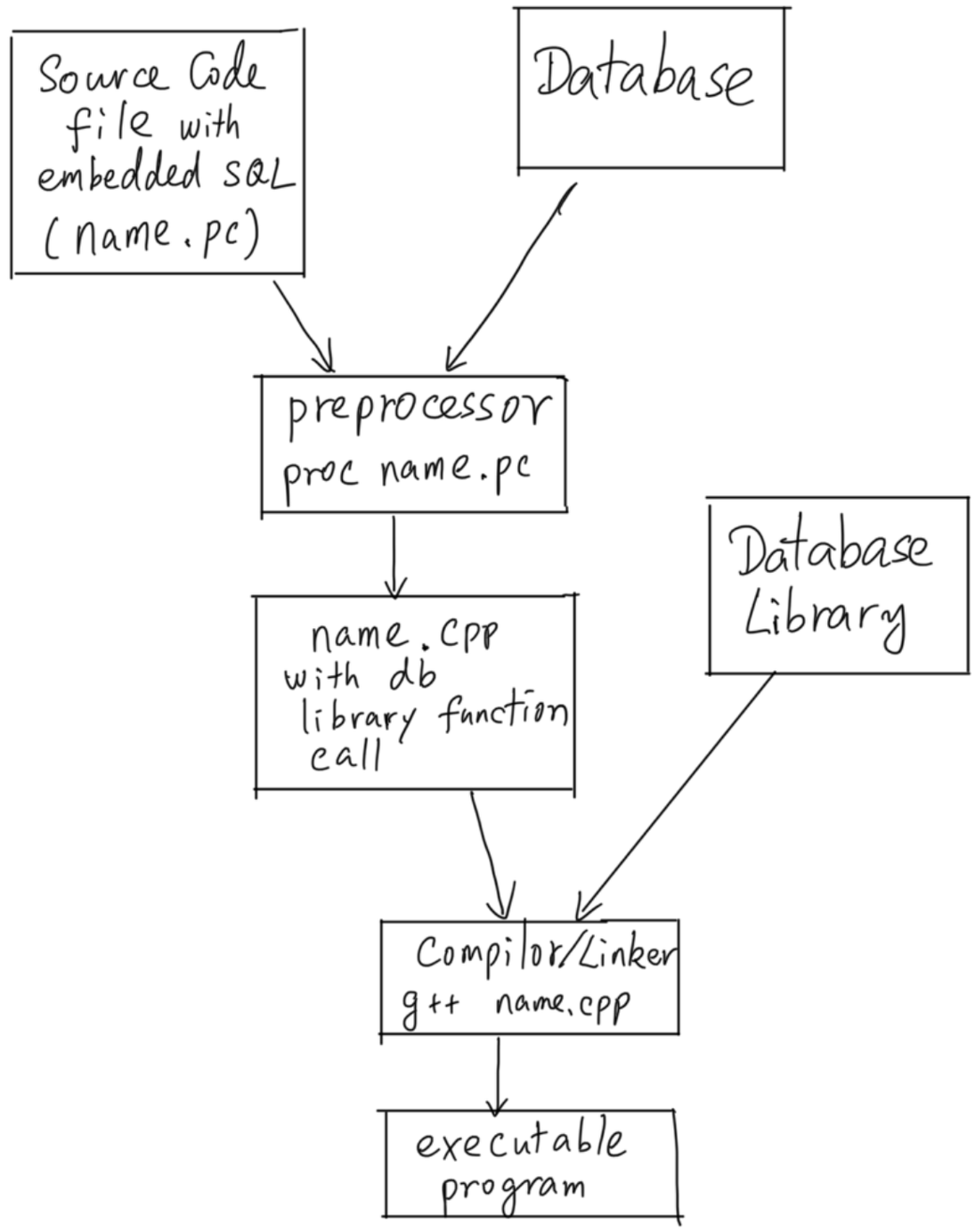
Client-Server Communication



Embedded SQL

- SQL Statements are embedded in a host language
- The application is preprocessed to pure host language program plus library calls.
- advantage:
    - preprocessing of (static) parts of queries
    - various (SQL) checks before running application
- disadvantage:
    - need pre-compiler
    - need to be bound to a database even at compile stage

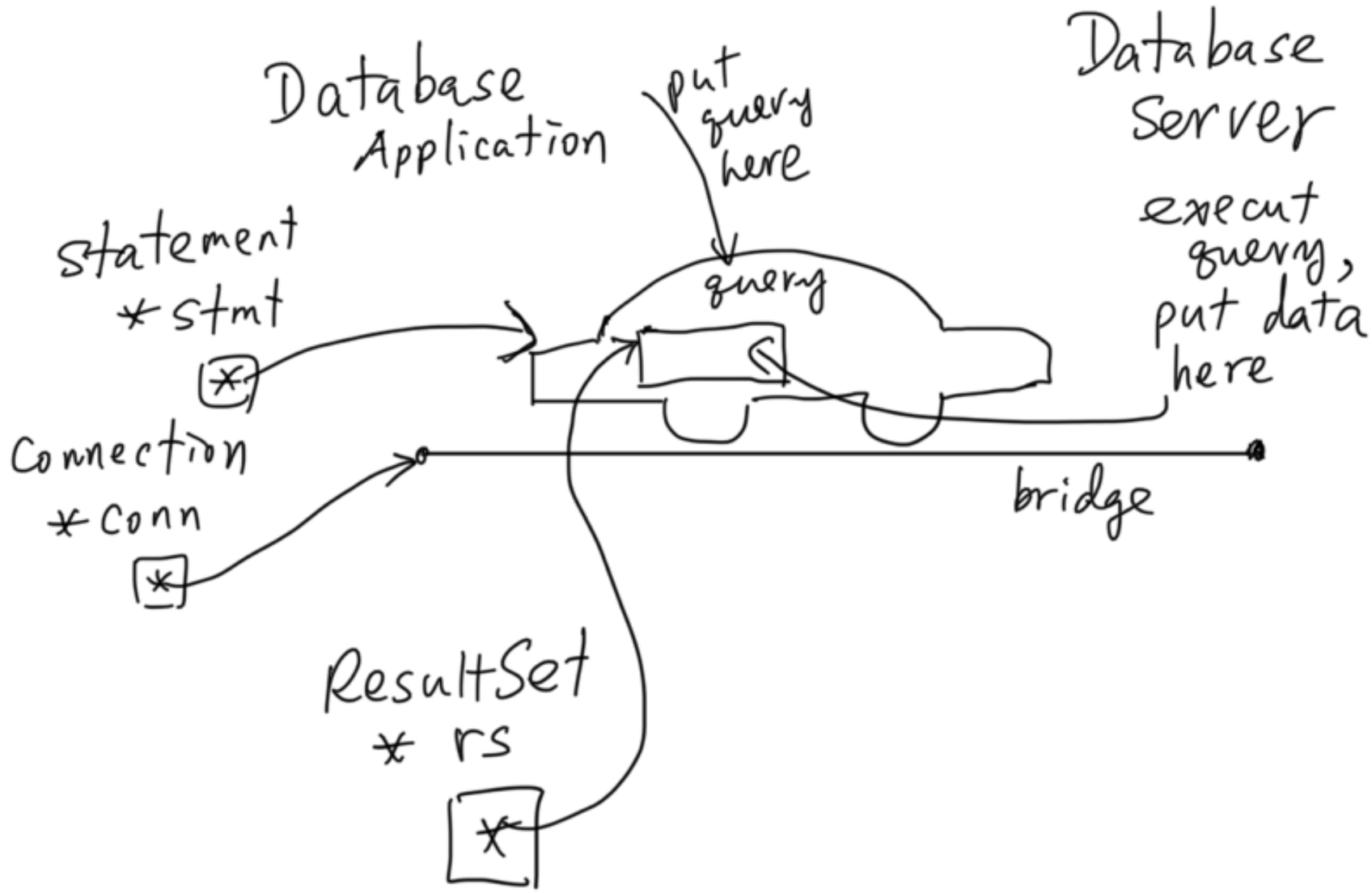
Compilation of Embedded SQL



Call Level Interface

- built on library calls
- different database product supplies different packages (libraries)
- OCCI: Oracle C++ Call Interface

DB Application



Make Connection

- Connect to db
    - All OCCI processing happens in the context of the Environment class. To create an environment and to terminate one:

      Environment *env = Environment::createEnvironment();
      Environment::terminateEnvironment(env);

    - to create and terminate a connection:

      Connection *conn =
          env->createConnection(userName, password, connectString);
      env->terminateConnection(conn);

      Note that connectString indicates where the database server runs.

Prepared vs Unprepared

- In order to submit query, you need to create a statement first.
- There are two types of statement:
    - Unprepared statement:
      Statement *stmt = conn->createStatement();
    - Prepared statement:
      Statement *stmt =
        conn->createStatement(queryStringWithParameters);
- To terminate a statement and clean up the statement space:
  conn->terminateStatement(stmt);

Execute Query

- Set parameter values:
  stmt->setInt(index, value); // if value is int type
  stmt->setString(index, value); // if value is string type

- execute update queries and DDL statements:
  int status = stmt->executeUpdate(); // or
  int status = stmt->executeUpdate(query);

- execute retrieval queries:
  ResultSet *rs = stmt->executeQuery(); // or
  ResultSet *rs = stmt->executeQuery(query);

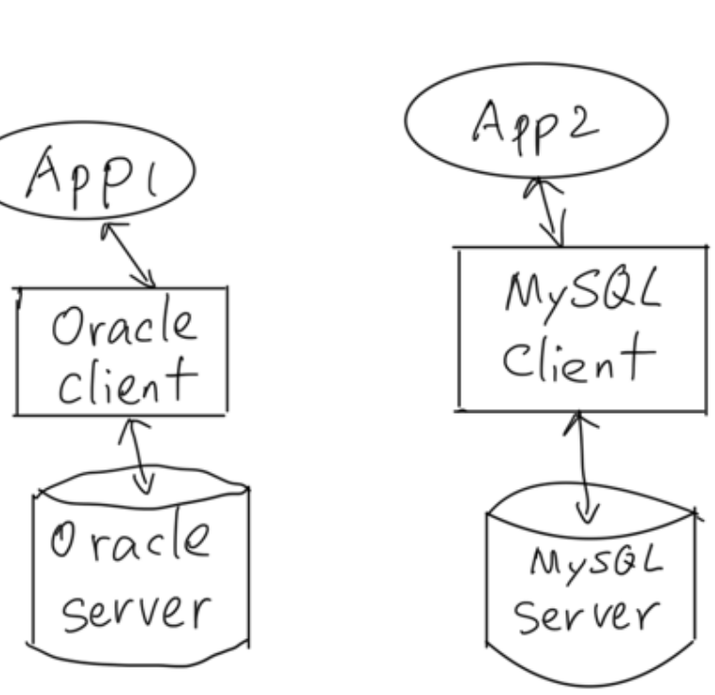
- generic execute: // not recommended
  stmt->execute();
  stmt->execute(query);

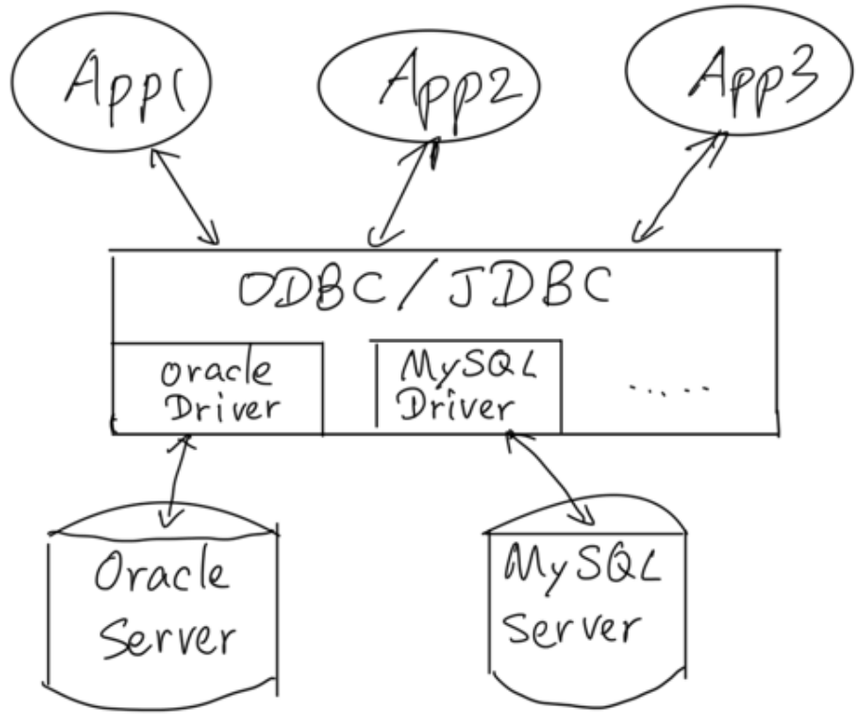
Read Result, etc

- Get result
  while (rs->next()) {
  string name = rs->getString(1);
  int age = rs->getInt(2);
  }

- Commit or Rollback
  - all queries in a program are considered as in a transaction
  - to commit or roll back updates early:
    conn->commit();
    conn->rollback();
    stmt->setAutoCommit(true/false);

Compare CLI & JDBC





JDBC

- register the driver with DriverManager:
  Class.forName(driver);

- make connection:
  Connection con = DriverManager.getConnection(connectString, userid, password);

- get a statement:
  Statement stmt = con.createStatement(); //or
  stmt = con.prepareStatement(String);

- execute query:
  int status = stmt.ExecuteUpdate(optional query string);
  ResultSet rs = stmt.ExecuteQuery(optional query string);

- retrieving answer:
  boolean rs.next()
  XXX rs.getXXX(int/String) // XXX as Int or String

- close all:
  rs.close(); stmt.close(); con.commit(); con.rollback(); con.close();

Security

- SQL Injection
- What is it?
- How to avoid it?
  - sanitize user input by escaping special characters
  - use prepared statement