CS352 Lecture - Database Application Development

Last revised February 28, 2023

Objectives:

1. To discuss client-server application that uses a database
2. To discuss use of embedded SQL
3. To discuss SQL injection in the context of database application development

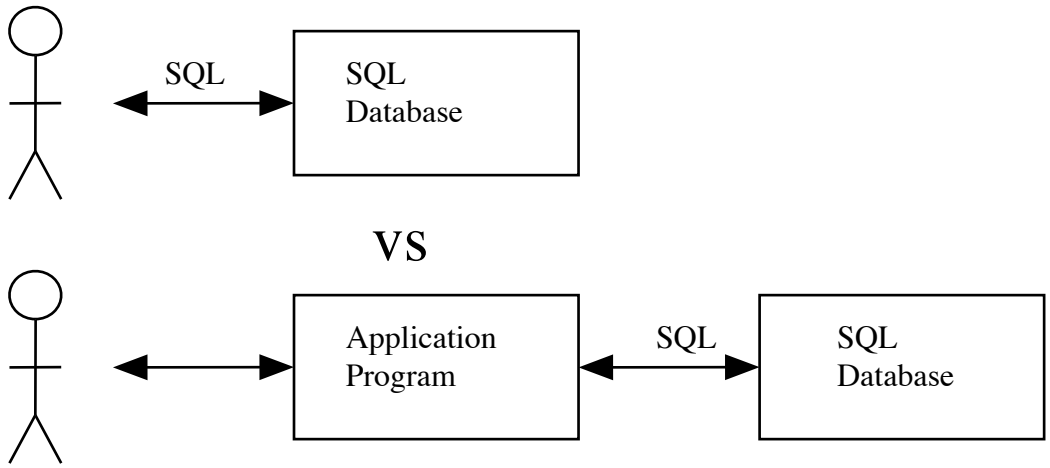
Materials:

1. Projectable of database access from an application
2. Projectable of three tier architecture
3. Projectable of JDBC architecture
4. Projectable of Example of JDBC Code
5. Projectable of JDBC Code revised to use prepared statement
6. Projectable of SQLJ example and of Java code after separation
7. Projectable of SQL Injection attack examples

I. Introduction

A. Thus far, we have used SQL as the means of actually accessing/modifying the database.

B. Of course, the majority of people accessing information stored in a database don't do so directly using SQL. Instead, they run an application program. While the application program may store its data in various kinds of application-specific files (the file processing approach), frequently it stores its information in SQL database.



PROJECT

Examples of the latter?

ASK

Numerous - many web-based ecommerce systems use a database to actually store the data; also bank tellers, insurance agents ...

C. When a user interacts with an application program that stores its data in a generic database, there are three kinds of tasks that are performed:

1. Database tasks - tasks related to accessing/modifying information in the database (e.g. tasks corresponding to SQL `select, insert, update, delete` or the equivalent in some other DML).

2. "Business logic" tasks - tasks related to the actual logic of the application (which vary widely from application to application, of course.) Quite a few things might fall into this category, including

   a) Carrying out the task(s) that the software is designed to do: looking up information, recording purchases/reservations/..., etc.

   b) Ensuring that the appropriate "business rules" are adhered too - e.g., for example, if a system is registering students for courses one important "rule" that needs to be enforced is that a student cannot be signed up for two different courses meeting at the same time (at all, or perhaps without some sort of special permission)

c) Ensuring that users are properly authenticated if sensitive information is being made available or data is being modified.

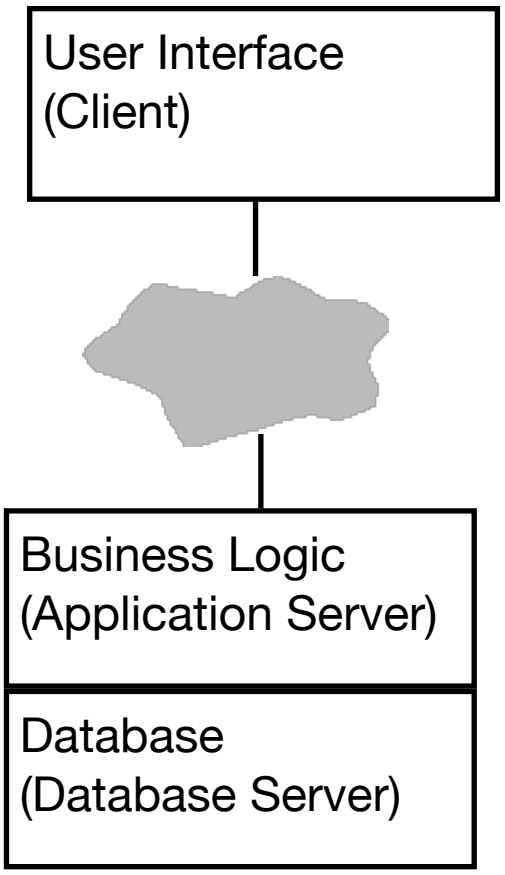
3. User interface tasks - tasks concerned with presenting information to the user and accepting commands from the user. This could be via a command line interface or some kind of dedicated hardware (e.g an ATM), but is often done through a GUI or the web.

D. It is certainly possible for all three kinds of tasks to be done by the same program.

Example: your library project for CPS122 (sort of - but could have been made more this way)

However, most applications make use of a client-server model, where the user interface resides on the user's computer or mobile device which communicates with the application using the internet - typically via http.

The most common architecture is one called the three-tier architecture.



PROJECT: three-tier architecture

1. In many cases, the middle layer also includes a web server that responds to http requests from a web browser (serving as the user interface) or an app running on a mobile device.

2. It is possible for the application and database servers to either be the same system or to be on two separate systems communicating vial a network. Either way, though, they are conceptually distinct and communicate with each other - often through SQL

E. The book included some discussion of communication between the user interface and the application server via http, but we won't deal with this today. (At Gordon, this is dealt with in our Internet Programming and Mobile Systems courses.)

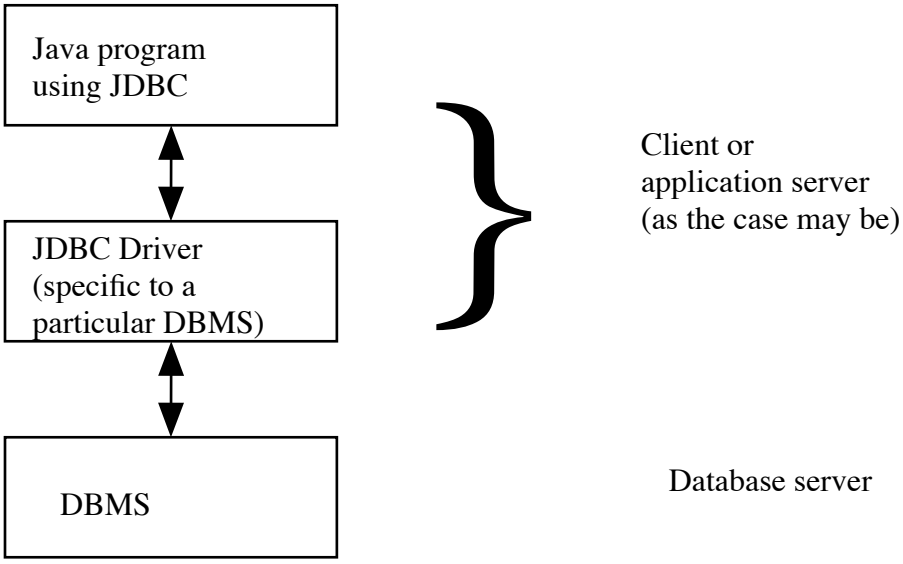
Our focus in this lecture will be on the interaction between the application and database layers.

II. **Accessing the Database from Within a Program**

A. One of two approaches is typically used to allow the application code to access the database

1. One approach - called <u>dynamic SQL</u> - involves the application program generating SQL statements as character strings. Such a string is then passed to the database, where it is interpreted. In the case of Java, the technology used is called JDBC - Java Database Connectivity. (You have used this for a lab in CS221.)

   a) A system that uses JDBC has the following architecture:



PROJECT - JDBC ARCHITECTURE

(1)The Java program itself will work on any platform that supports Java and with any DBMS that supports JDBC.

(2)The JDBC driver - a collection of classes typically furnished by the database vendor - serve to translate operations in the Java program into network messages to the database server in a form that is appropriate to the specific DBMS being used. (Thus, each brand of DBMS needs its own JDBC driver; but since the JDBC driver is often written in Java, it could be platform-independent. However, some drivers do use platform-specific native code, in which case they are platform-specific as well.).

(3)The QBE program you used on a homework set is structured this way

(a) The QBE program ran as a Java application on the your host system.

(b) It used the JDBC driver you installed when setting up your Db2 container. Though the driver was installed in your container, it was available on your host because of the shared directory. (This is why it was critical that QBE be placed in the directory shared between your host and container.)

(c)It actually communicated with the DBMS running under docker in your container.

(4)Just to review, here is an example of JDBC code (from CPS221 lab).

PROJECT JDBC Code Example

(a)A SQL statement is constructed as a character string, using string concatenation as with any Java string.

(b)A SQL statement is executed by passing it as a parameter to a method of class `java.sql.Statement`, which in turn passes it to the JDBC driver, which in turn passes it on to the database - with the result traveling back to the program via the reverse of the same path.

(5)The process of parsing the SQL statement is computationally expensive. For some statements (e.g. those involving joins), some effort may also be expended by the DBMS on planning a good strategy for executing the query (we will see later in the course how much of a difference this can make). This can also be computationally expensive.

Because of the computation potentially involved in parsing and strategy planning, JDBC incorporates the notion of a "prepared statement", which allows this work to be performed once - when the statement is prepared - rather than each time it needs to be executed. A prepared statement can have parameters, which are values to be supplied when the prepared statement is actually executed.

PROJECT - revision of JDBC to use a prepared statement

(a)Note the use of ? as a placeholder in original statement

(b)Note the use of an appropriate method (in this case `setString()`, though there are many others) to set the parameters.

(c)Note the use of `executeUpdate()` to actually execute the prepared statement with the specified parameters., (There is also an `executeQuery()` that returns a `ResultSet`, which is used with `select` statements.)

b) JDBC is actually based on an earlier technology (originated by Microsoft) called ODBC - Open Database Connectivity - which supports database access by application programs written in a variety of different programming languages through a library. (The architecture is similar to JDBC, except for the presence of an ODBC library between the program and the driver; for this reason, ODBC drivers are also platform-specific).

2. The other approach - called static SQL - involves the application program being a mixture of SQL and some other programming language.

a) The language in which the SQL is embedded is called the host language.

(1)The syntax details for any particular language are common to that language - regardless of what brand of DBMS is being used.

(2)Of course, SQL itself is also (relatively standard).

(3)However, the actual implementation of embedded SQL for a specific language/DBMS is typically furnished by the DBMS vendor.

b) Embedded SQL implementations exist for a variety of host languages. For example, db2 comes with support for embedding SQL in Java, C/C++, COBOL, FORTRAN, Perl, and REXX.

c) We will be using this approach for the programming project in the course. We will be using an approach to embedding sql in java known as SQLJ.

(1) The "style" of SQLJ is a bit different from SQL embeddings in other languages. It was developed jointly by 7 major software vendors to tale advantage of some distinctive characteristics of the Java language, such as built-in support for threading.

(2)Despite being an ANSI standard, SQLJ never really "caught on" - in part due to lack of support being included in IDEs such as NetBeans, Eclipse, etc.

(3)Nonetheless, we will use it for the second project in this course because its style is very similar to SQL embeddings in other host languages, and GUI development is much easier in Java than in other languages typically used with embedded SQL.

d) Here is an example of embedded SQL code, using SQLJ.

PROJECT EXAMPLE OF SQLJ

(1)The overall program is a standard Java program.

(2)The program contains embedded SQL statements. In the case of Java, the embedded SQL is bracketed by `#sql { ... }`. (Other languages typically bracket the SQL by a construct like `exec SQL ... ;`).

(3)Information flows between the two languages via host variables. A host variable is a variable in the syntax of the host language, and can be used as such by the host language code. In the SQL code, it is a marked as being a host variable by being preceded by a colon - e.g. `categoryName` is a Java parameter that is used in the SQL code under the name `:categoryName`.

(a)A host variable may serve to pass information from the Java program into the SQL code (e.g. `:categoryName`).

(b)A host variable may serve to pass information from the SQL code back to the Java program. (e.g. `:checkoutPeriod`).

(c)Not illustrated here - but also possible - is a host variable that serves both purposes.

(4)A SQL statement is executed when it is encountered in the normal flow of execution of the host language code - i.e. it is executed just as if it were a host language statement. In fact, embedded SQL statements can appear in host language control structures, in which case they may be executed conditionally or repeatedly.

e) When a program uses embedded SQL, a more complex process is needed to prepare it for execution. For example, here is the process used by db2 for SQL code embedded in Java - referred to as SQLJ. (The process details will vary from language to language and DBMS to DBMS).

(1) The process begins with a source file that contains a mixture of Java and SQL code (as in the example just projected). Such a file typically has the file type .sqlj.

(2) The SQLJ file is processed by the sqlj compiler (furnished by IBM), which splits it into two files:

(a) A “pure java” file that contains the Java code plus calls to SQL procedures.

(e) A SQL module file.

For example, the following is an excerpt from the results of invoking sqlj on the file projected earlier. (This excerpts correspond to the routine we just looked at.)

PROJECT - Java code for example procedure;

(In the case of the db2 implementation of sqlj, the SQL module code.is not created in a human-readable form, so I can't project that)

(3) The "pure java" file is then translated by the standard Java compiler into one or more class files.

(4) The SQL code is bound to the database, using a program furnished by IBM. In effect, what happens is that stored procedures are created in the database that correspond to the SQL code in the original program.

As part of the translation process, the textual form of the SQL statements is replaced by a compiled form. Thus, the task of parsing the SQL and constructing an optimized strategy for processing a query is done once at build time, rather than each time a given SQL statement is executed (as is the case with dynamic SQL).

f) When the program runs, it establishes a connection to the DBMS, and then executes the SQL statements embedded in the Java code as they are encountered during execution. An embedded SQL statement is executed by sending a message to the DBMS requesting it to execute the appropriate stored procedure that was created by translating the SQL.

III.SQL Injection

A. The book discusses quite a number of security issues related to web applications.

B. One that is particularly relevant to the development of database applications is SQL injection which was discussed in the assigned reading.

1. How does SQL Injection work?

ASK

a. Step through projected first example

b. Discuss second as well - obviously the inserted statement could be any.

2. How might a hacker introduce a SQL injection?

    1. Values entered into a field on a web form.

    2. Query strings (?= in URL)

3. What strategies can be used to prevent a program from being attacked this way?

    ASK

    a. Much less than ideal: check all user provided input for the presence of close quote characters and then escape them (e.g. with '\').

        Problem: too easy to forget to do this in one place. Therefore, it is best to NEVER create SQL statements by concatenating strings when working with user-supplied data,

    b. Much better: use of parameterized statements, since user entered data is always treated as the value of a database column rather than as SQL code to be executed.

        e.g. in Java, the following are safe alternatives (and similar mechanisms exist for other languages.)

        1. Use of parameterized prepared statements with JDBC.

        2. Use of embedded SQL with SQLJ.