Database programming

Elmasri/Navathe ch 12
Padron-McCarthy/Risch ch 20

DATABASE DESIGN I - IDL300 Fall 2012
Sobhan Badrozmamy
Silvia Stefanova
Department of Information Technology, Uppsala University

ODBC Demo
Database programming
Elmasri/Navathe ch 12
Padron-McCarthy/Risch ch 20
DATABASE DESIGN I - IDL300 Fall 2012
Sobhan Kadiozamany
Silvia Stefanova
Department of Information Technology, Uppsala University

ODBC Demo
Database programming

• To access a database from an application program.
• As opposed to interactive interfaces

But, a majority of database operations are made through application programs (increasingly thru web applications), like in a banking system.

generally a DBMS is a Service provided in client/server model.
banking system.

So far we have learned how to communicate to the DBMS using a direct SQL prompt.
Graphical interfaces

Build queries by picking:
- tables
- attributes
- criteria
joining is automatically taken care of.
But, a majority of database operations are made through application programs (increasingly thru web applications), like in a banking system. Generally a DBMS is a service provided in client/server model.

So far, we have learned how to communicate to the DBMS using a direct SQL prompt. Build queries by picking:
- tables
- attributes
- criteria joining is automatically taken care of.
When you click on view a message is sent to the application server.
Database program at the application Server

Works based on a specific application logic, this case banking:

- Your total balance is calculated by summing the amount of money in all of your accounts.

Typical sequence of interaction between application server and the database server:

- establish a connection
- submit queries, updates, insert
  - The queries can be dynamically built based on application logic.
- closes connection

The results are then sent back to the client, e.g. smart phones, webpages, ATM
Database programming

• To access a database from an application program.
• As opposed to interactive interfaces

But, a majority of database operations are made through application programs (increasingly thru web applications), like in a banking system.

generally a DBMS is a Service provided in client/server model.
Views

- Provides a mechanism to hide certain data from the view of certain users. To create a view we use the command:

  ```sql
  create view viewname as <query expression>
  where: <query expression> is any legal expression.
  ```

- Create a view consisting of branches and their customers:

  ```sql
  create view all_customer as
  (select branch_name, customer_name
   from depositor, account
   where depositor.account_number =
     account.account_number)
  union
  (select branch_name, customer_name
   from borrower, loan
   where borrower.loan_number = loan.loan_number)
  ```

- Find all customers of the Perryridge branch:

  ```sql
  select customer_name
  from all_customer
  where branch_name = "Perryridge"
  ```

Important: Views are not physically stored, they are calculated every time they are called.
Updating views

- Create a view of all loan data in the loan relation, hiding the amount attribute:
  
  ```
  create view branch_loan as
  select branch_name, loan_number
  from loan
  ```

- Add a new tuple to branch_loan:
  ```
  insert into branch_loan
  values ("Perryridge", "L-307")
  ```

- This insertion must be represented by inserting into the loan relation the tuple:
  ```
  ("Perryridge", "L-307", null)
  ```

- Updates on more complex views are difficult or impossible to translate, and hence are disallowed.
Database programming approaches

**Embedded**
- Database commands are embedded in a general-purpose programming language
- Disadvantages:
  - Program is dependent on the DBMS
  - Only one connection at a time is allowed

**Library of database functions**
- Available to the host language for database calls
- Known as an Application Program Interface (API)
- **JDBC**
  - Java database connectivity
  - Common SQL functions or access database
  - Strong tool for Java
  - Uses Java native code connection

**ODBC**
- OPEN database connectivity
- Common SQL functions or access database
- Uses C native code connection
- Uses Java native code connection

Better software maintenance and code reuse
Database commands are embedded in a general-purpose programming language.
general note on embedded SQL

- Most SQL statements can be embedded in a general-purpose host programming language such as COBOL, C, Java

- An embedded SQL statement is distinguished from the host language statements by enclosing it between `EXEC SQL` or `EXEC SQL BEGIN` and a matching `END-EXEC` or `EXEC SQL END` (or semicolon)
  - Syntax may vary with language
  - *Shared variables* (used in both languages) usually prefixed with a colon (:) in SQL
**embedded**

Database commands are embedded in a general-purpose programming language.

- **Connect**
  - Connection (multiple connections are possible, but only one is active)
  - CONNECT to connection-name as connection-name
  - Change from one active connection to another one using CONNECT connection-name.

- **Query**
  - syntax: (query) variables prefixed with `:`
  - This example: simple select query embedded inside a loop.

```sql
begin
  for r in (SELECT * FROM table)
  BEGIN
    for c in (SELECT * FROM another_table)
    BEGIN
      -- Do something with r and c
    END
  END
end
```

- **Disconnect**
  - syntax: DISCONNECT connection-name.
Connect

- Connection (multiple connections are possible but only one is active)
  
  CONNECT TO server-name AS connection-name
  AUTHORIZATION user-account-info;

- Change from an active connection to another one
  
  SET CONNECTION connection-name;
Variables in embedded SQL

- Variables inside `DECLARE` are shared and can appear (while prefixed by a colon) in SQL statements.
- `SQLCODE` is used to communicate errors/exceptions between the database and the program.

```sql
int loop;
EXEC SQL BEGIN DECLARE SECTION;
  varchar dname[16], fname[16], ...;
  char ssn[10], bdate[11], ...;
  int dno, dnumber, SQLCODE, ...;
EXEC SQL END DECLARE SECTION;
```
Embedded

Database commands are embedded in a general-purpose programming language

```
CONNECT connection-name.

Variables in embedded SQL:
- @var1
- @var2
- @var3

Query

insert local (query) variables prefixed with @: This example: single-valued query embedded in a loop.

Disconnect

DISCONNECT connection-name.
```
Query

uses local (host) variables prefixed with:
This example: simple retrieval query embedded inside a loop.

```sql
loop = 1;
while (loop) {
    prompt ("Enter SSN: ", ssn);
    EXEC SQL
        select FNAME, LNAME, ADDRESS, SALARY
        into :fname, :lname, :address, :salary
        from EMPLOYEE where SSN == :ssn;
    if (SQLCODE == 0) printf(fname, ...);
    else printf("SSN does not exist: ", ssn);
    prompt("More SSN? (1=yes, 0=no): ", loop);
    END-EXEC
}
```
DiSconnect

DISCONNECT connection-name;
embedded

Database commands are embedded in a general-purpose programming language

Disadvantages:
- program is dependent on the DBMS
- only one connection at a time is allowed
Database programming approaches

**embedded**
- Database commands are embedded in a general-purpose programming language
- Disadvantages:
  - Program is dependent on the DBMS
  - Only one connection at a time is allowed

**Library of database functions**
- Available to the host language for database calls
- Known as an Application Program Interface (API)
- **JDBC**
  - Java Database Connectivity
  - SQL statements can be executed from Java programs
  - Example languages: Java, C, C++, etc.

- **ODBC**
  - Open Database Connectivity
  - SQL statements can be executed from any language
  - Example languages: C, C++, Java, etc.

Better software maintenance and code reuse:
Library of database functions

Available to the host language for database calls;
- known as an Application Program Interface (API).

**JDBC**
- SQL connection function calls for Java programming
- A Java program with JDBC functions can access any relational DBMS that has a JDBC driver
- JDBC allows a program to connect to several databases (known as data sources)

**ODBC**
- ODBC API is independent of any one programming language, database system or operating system
- An ODBC connection is defined in the Operating System, the properties are:
  - Data source type: MySQL, MS SQL Server, Oracle, etc.
  - Database server address
  - Authentication info: user/pass
  - Programs written in various languages can use the same ODBC connection.
  - Any database connection is defined once for all, therefore any change in database connection is easily handled:
  - Changing an ODBC connection affects programs that use it.
SQL connection function calls for Java programming

- A Java program with JDBC functions can access any relational DBMS that has a JDBC driver
- JDBC allows a program to connect to several databases (known as data sources)

Steps in JDBC Database Access

1. Import JDBC library (java.sql)
2. Load JDBC driver
3. Create database (e.g., connect to database)
4. Create a connection object (e.g., get connection)
5. Create a statement object (e.g., PreparedStatement or CallableStatement)
6. Identify column parameters; substitute by question marks
7. Bind parameters to program variables
8. Execute SQL statement (as created by an object via JDBC's prepare query)
9. Process query results (as returned in an object of type ResultSet)

* associated to a multidimensional Java
Steps in JDBC Database Access

1. Import JDBC library \( \text{java.sql.*} \)
2. Load JDBC driver:
   \texttt{Class.forName("oracle.jdbc.driver.OracleDriver")}
3. Define appropriate variables
4. Create a connect object (via \texttt{getConnection})
5. Create a statement object from the \texttt{Statement} class:
   - \texttt{1. PreparedStatement}
   - \texttt{2. CallableStatement}
6. Identify statement parameters (designated by question marks)
7. Bound parameters to program variables
8. Execute SQL statement (referenced by an object) via JDBC’s \texttt{executeQuery}
9. Process query results (returned in an object of type \texttt{ResultSet})
   - \texttt{ResultSet} is a 2-dimentional table
ODBC API is independent of any one programming language, database system or operating system.

- An ODBC connection is defined in the Operating System, the properties are:
  - Data source type: MySQL, MS SQL Server, Oracle, etc.
  - database server address
  - Authentication info: user/pass
- programs, written in various languages can use the same ODBC connection.
- Any database connection is defined once for all, therefore any change in database connection is easily handled:
  - Changing an ODBC connection affects are programs that use it.
ODBC API is independent of any one programming language, database system or operating system.

- An ODBC connection is defined in the Operating System, the properties are:
  - Data source type: MySQL, MS SQL Server, Oracle, etc.
  - database server address
  - Authentication info: user/pass
- programs, written in various languages can use the same ODBC connection.
- Any database connection is defined once for all, therefore any change in database connection is easily handled:
  - Changing an ODBC connection affects are programs that use it.
Database programming approaches

**Embedded**
- Database commands are embedded in a general purpose programming language
- Programs are dependent on the DBMS
- Only one connection at a time is allowed

**Library of database functions**
- Available to the host language for database calls
- Known as an Application Program Interface (API)
  - ODBC API: a standard of any programming language, database, or operating system

Better software maintenance and code reuse:
In ODBC, a change in a database connection is transparently made in one place for all applications that are using it.
ODBC/JDBC architecture

both ODBC and JDBC are based on client server model.

- Database is a server
- Application program is a client.
- There could also exist middle-ware, standing in between.
ODBC API is independent of any one programming language, database system or operating system.
This layer is specific to DBMS -->

This layer is specific to DBMS -->
The independence of any one programming language, database system.

---

This layer is independent of OS and DBMS.

---

This layer is specific to Operating System.
ODBC API is independent of any one programming language, database system or operating system.

--- This layer is independent of OS and DBMS

--- This layer is specific to Operating System
JDBC API is independent of (relational) DBMS and operating system

- These sections can be merged into one, (the Middleware is not always necessary)
- This layer is specific to DBMS
These sections can be merged into one (the Middleware is not always necessary).

--- This layer is specific to DBMS

--- This layer is specific to DBMS
Independent of (relational) DBMS and operating system

These sections can be merged into one, (the Middleware is not always necessary)

--- This layer is specific to DBMS

--- This layer is specific to DBMS
JDBC API is independent of (relational) DBMS and operating system

These sections can be merged into one, (the Middleware is not always necessary)

<-- This layer is specific to DBMS

<-- This layer is specific to DBMS
ODBC/JDBC bridge

Sometimes there is no JDBC driver for a specific DBMS, so java programs have to use a "bridge".

Makes ODBC accessible from JDBC such that no special JDBC drivers are required.