Agenda

- Moving Window
- CTE
- JDBC

```
-- display statement for accid=1
INSERT INTO transactions VALUES(1, '2000-01-08', 1000);
INSERT INTO transactions VALUES(1, '2000-01-08', 2000);
```

COMMON TABLE EXPRESSION (CTE)

- Derived table is a virtual table returned from a sub-query in FROM clause of outer query.
- This is also referred as "Inline view".
- The derived table must have an alias.
- CTE is a virtual table returned from a select query
- Used to simplify the queries and make it readable
- CTE are of 2 types

1. Non Recursive
2. Recursive

1. Non Recursive CTE

2. Recursive CTE

```
void seq(int s, int e) {
    if(s <= e) {
    printf("%d",s);
    seq(s+1, e);
    }
}
```

```
WITH RECURSIVE seq(n) AS(
(SELECT 1)  -- anchor (s)
UNION
(SELECT n+1 FROM seq -- recursive member
WHERE n<4) -- base condition (e)
)
 SELECT * FROM seq;

-- OR

WITH RECURSIVE seq AS(
```

```
(SELECT 1 AS n)  -- anchor (s)
UNION
(SELECT n+1 FROM seq -- recursive member
WHERE s<4) -- base condition (e)
)
SELECT * FROM seq;
```

JDBC

- RDBMS understand SQL language only.
- JDBC driver converts Java requests in database understandable form and database response in Java understandable form.
- JDBC drivers are of 4 types

1. Type I - Jdbc Odbc Bridge driver
    - ODBC is standard of connecting to RDBMS (by Microsoft).
    - Needs to create a DSN (data source name) from the control panel.
    - From Java application JDBC Type I driver can communicate with that ODBC driver (DSN).
    - The driver class: sun.jdbc.odbc.JdbcOdbcDriver -- built-in in Java.
    - database url: jdbc:odbc:dsn
    - Advantages:
    - Can be easily connected to any database.
    - Disadvantages:
    - Slower execution (Multiple layers).
    - The ODBC driver needs to be installed on the client machine.
2. Type II - Partial Java/Native driver
    - Partially implemented in Java and partially in C/C++. Java code calls C/C++ methods via JNI.
    - Different driver for different RDBMS. Example: Oracle OCI driver.
    - Advantages:
    - Faster execution
    - Disadvantages:
        - Partially in Java (not truely portable)
        - Different driver for Different RDBMS
3. Type III - Middleware/Network driver
    - Driver communicate with a middleware that in turn talks to RDBMS.
    - Example: WebLogic RMI Driver
    - Advantages:
        - Client coding is easier (most task done by middleware)
    - Disadvantages:
        - Maintaining middleware is costlier
        - Middleware specific to database
4. Type IV

- Database specific driver written completely in Java.
- Fully portable.
- Most commonly used.
- Example: Oracle thin driver, MySQL Connector/J, ...

MySQL Programming Steps

- step 0: Add JDBC driver into project/classpath. In Eclipse, project -> right click -> properties -> java build path -> libraries -> Add external jars -> select mysql driver jar.
- step 1: Create JDBC connection using helper class DriverManager.

```
Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/classwork", "root",
"manager");
```

- step 3: Create the statement.

```
Statement stmt = con.createStatement();
```

- step 4: Execute the SQL query using the statement and process the result.

```
String sql = "non-select query";
int count = stmt.executeUpdate(sql); // returns number of rows affected
OR
String sql = "select query";
ResultSet rs = stmt.executeQuery(sql);
while(rs.next()) // fetch next row from db(return false when all rows completed)
{
  x = rs.getInt("col1");
  // get first column from the current row
  y = rs.getString("col2");
  // get second column from the current row
  z = rs.getDouble("col3");
  // get third column from the current row
  // process/print the result
}
rs.close();
```

- step 5: Close statement and connection.

```
con.close();
stmt.close();
```

SQL Injection

- Building queries by string concatenation is inefficient as well as insecure.
- Example:

```
dno = sc.nextLine();
sql = "SELECT * FROM emp WHERE deptno="+dno;
```

- If user input "10", then effective SQL will be "SELECT _ FROM emp WHERE deptno=10". This will select all emps of deptno 10 from the RDBMS.
- If user input "10 OR 1", then effective SQL will be "SELECT _ FROM emp WHERE deptno=10 OR 1". Here "1" represent true condition and it will select all rows from the RDBMS.
- In Java, it is recommeded NOT to use "Statement" and building SQL by string concatenation. Instead use PreparedStatement.

PreparedStatement

- PreparedStatement represents parameterized queries.

```
String sql = "SELECT * FROM students WHERE name=?";
PreparedStatement stmt = con.prepareStatement(sql);

System.out.print("Enter name to find: ");
String name = sc.next();

stmt.setString(1, name);
ResultSet rs = stmt.executeQuery();

while(rs.next()) {
  int roll = rs.getInt("roll");
  String name = rs.getString("name");
  double marks = rs.getDouble("marks");
  System.out.printf("%d, %s, %.2f\n", roll, name, marks);
}
```

- The same PreparedStatement can be used for executing multiple queries. There is no syntax checking repeated. This improves the performance.