SQL Cursor Processing

Chapter 11

The Impedance Mismatch

- *Result set:* The collection of rows returned by the execution of a database query
- Most programming languages (e.g. C, Python, Ruby) handle one record at a time, but SQL queries return entire sets of records with one command
  - Many call this an *impedance mismatch*.

The Cursor

- The *cursor*, available in most RDBMSs overcomes this problem by providing a *pointer* to the result set, thus allowing "one record at a time" processing by advancing the cursor
  - Parallels reading a traditional flat file in a program written in a language, such as C
  - Cursor must be defined and opened before it may be used

EMPLOYEE ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT
145	John	Russell	14000	0.4
146	Karen	Partners	13500	0.3
147	Alberto	Errazuriz	12000	0.3
148	Gerard	Cambrault	11000	0.3
149	Eleni	Alotkey	10500	0.2
150	Peter	Tucker	10000	0.3
151	David	Bernstein	9500	0.25
152	Peter	Hall	9000	0.25
153	Christopher	Olsen	8000	0.2

Cursor Implementation

- In Oracle, cursor support is part of the PL/SQL language extension to SQL
- Similarly, in Microsoft SQL Server and Sybase, cursors are supported by the Transact SQL language extension to SQL
- Some syntax variances across vendors

Cursor Processing Sequence

- Declare the Cursor
  - Must occur before any other reference to the cursor
  - Generally not an executable statement
  - *Host variables* (host language variables referenced by the the SQL statements) must also be declared.

```
DECLARE CURSOR ny_customers AS
   SELECT customer_number, name,
          address, city, zip_code
     FROM customer
    WHERE state = 'NY';
```

Cursor Processing Sequence

- Open the Cursor
  - Must be opened before any data can be retrieved
  - The DBMS may or may not fetch rows at this point
    - If query requires a sort, it is likely the table will be sorted when the cursor is opened

```
OPEN Emp;
```

Cursor Processing Sequence

- Fetch from the CURSOR
  - The FETCH command returns rows
  - Status code indicates when no more rows exist
  - Rows are returned in result set sequence, unless cursor scrolling is used (covered a bit later)
  - Host variables are updated with each fetch

```
FETCH Emp
 INTO :EmpID, :FirstName, :LastName,
      :Salary, :CommissionPCT;
```

Cursor Processing Sequence

- Close the Cursor
  - Frees up resources
  - Can usually be re-opened
  - When host variable used in WHERE clause, must close and re-open for a change in value to take effect
    - For example `WHERE SALARY > :SearchSal`

```
CLOSE Emp;
```

Cursor Sensitivity (Optional)

- Cursor Sensitivity:
  - Controls whether cursor is can return data that was changed by statements outside the cursor
  - Possible Settings:
    - SENSITIVE: Significant changes made by statements outside the cursor immediately affect the results within the cursor
    - INSENSITIVE: Significant changes made by statements outside the cursor do not affect the results returned by the cursor.
    - ASENSITIVE: Cursor sensitivity is defined by the DBMS implementation. Significant changes may or may not be visible within the cursor
  - Vendor variations: Oracle uses CURSOR_SHARING initialization parameter

Cursor Sensitivity Example

```
DECLARE Emp CURSOR
  SENSITIVE
  FOR
  SELECT EMPLOYEE_ID, FIRST_NAME,
         LAST_NAME, SALARY,
         COMMISSION_PCT
    FROM EMPLOYEE
   ORDER BY EMPLOYEE_ID;
```

Cursor Scrollability (Option)

- Cursor Scrollability
  - Controls the options that the FETCH statement can use when retrieving data.
  - If the SCROLL option is specified, the FETCH statement can use a number of advanced options that allow it to move through the result set and retrieve rows in a non-sequential manner.
  - If NO SCROLL (the default) is specified, rows can only be returned in result set sequence

Example: Sensitive and Scroll

```
DECLARE Emp CURSOR
  SENSITIVE SCROLL
  FOR
  SELECT EMPLOYEE_ID, FIRST_NAME,
         LAST_NAME, SALARY,
         COMMISSION_PCT
    FROM EMPLOYEE
   ORDER BY EMPLOYEE_ID;
```

Cursor Scrolling

- FETCH Scrolling Options:
  - NEXT - Retrieves the next row
  - PRIOR - Retrieves the previous row
  - FIRST - Retrieves the first row in the result set
  - LAST - Retrieves the last row in the result set
  - ABSOLUTE <value> - Retrieves the row specified by the <value> placeholder
  - RELATIVE <value> - Retrieves the row specified by the <value> placeholder relative to current cursor position

Cursor Scrolling Examples

```
FETCH NEXT Emp
 INTO :EmpID, :FirstName, :LastName,
      :Salary, :CommissionPCT;
```

EMPLOYEE ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT
145	John	Russell	14000	0.4
146	Karen	Partners	13500	0.3
147	Alberto	Errazuriz	12000	0.3
148	Gerard	Cambrault	11000	0.3
149	Eleni	Alotkey	10500	0.2
150	Peter	Tucker	10000	0.3
151	David	Bernstein	9500	0.25
152	Peter	Hall	9000	0.25
153	Christopher	Olsen	8000	0.2

Cursor Holdability

- Cursor Holdability:
  - Refers to whether a cursor is automatically closed when a transaction is committed
  - WITH HOLD - specifies that the cursor remains open on commit
  - WITHOUT HOLD – specifies that the cursor is automatically closed on commit

```
DECLARE Emp CURSOR
  SENSITIVE SCROLL WITH HOLD
  FOR …
```

Cursor Updatability

- Cursor Updatability
  - Controls whether rows fetched by the cursor can be updated
  - READ ONLY - specifies that rows cannot be updated
  - FOR UPDATE OF <column list> - specifies that rows can be updated or deleted
  - WHERE CURRENT OF <cursor-name> clause is used to reference the last row fetched from the cursor