# ITC 5104 DATABASE DESIGN AND SQL

Lecture 5

Chapter 5 Oracle 12c: SQL

Data Manipulation

# Objectives

- Add a record to an existing table
- Add a record containing a NULL value to an existing table
- Use a subquery to copy records from an existing table
- Modify existing rows within a table
- Use substitution variables with an UPDATE command
- Issue transaction control statements COMMIT, ROLLBACK and SAVEPOINT
- Differentiate among DDL, DML and transaction control commands
- Delete records

# Overview of Commands

| Command | Description |
|---|---|
| INSERT | Adds new row(s) to a table. The user can include a subquery to copy row(s) from an existing table. |
| UPDATE | Adds data to, or modifies data within, an existing row |
| COMMIT | Permanently saves changed data in a table |
| ROLLBACK | Allows the user to "undo" uncommitted changes to data |
| DELETE | Removes row(s) from a table |

# ACCTMANAGER Table

Worksheet | Query Builder

```
CREATE TABLE acctmanager
(
amid          CHAR(4) CONSTRAINT acctmanager_amid_pk PRIMARY KEY,
amfirst       VARCHAR2(12) CONSTRAINT acctmanager_amfirst_nn NOT NULL,
amlast        VARCHAR2(12) CONSTRAINT acctmanager_last_nn NOT NULL,
amedate       DATE  DEFAULT SYSDATE,
amsal         NUMBER(8,2),
amcomm        NUMBER(7,2) DEFAULT 0,
region        CHAR(2) CONSTRAINT acctmnanager_region_ck
                      CHECK (region IN ('N', 'NW', 'NE', 'S', 'SE', 'SW', 'W', 'E'))
);
```

Script Output

Task completed in 0.093 seconds

```
table ACCTMANAGER created.
```

This is the table we will insert new data records to

# Data to be Inserted

| ID | Name | Employment date | Salary | Commission | Region |
|---|---|---|---|---|---|
| T500 | Nick Taylor | September 5 ,2009 | $42,000.00 | $3,500.00 | NE |
| L500 | Mandy Lopez | October 1, 2009 | $47,000.00 | $1,500.00 | |
| J500 | Sammie Jones | Today | $39,500.00 | $2,000.00 | NW |

In the following slides, the data above will be added to the ACCTMANAGER table. Blank spaces indicate that the data that has not been provided yet, there are additional columns that will be added that are not shown here

# Insert Command

```
INSERT INTO tablename [(columnname, …)]
VALUES (datavalue, …);
```

# Insert Command

- The keywords INSERT INTO are followed by the name of the table into which the rows will be entered
- The table name is followed by the column names that will contain data
- The VALUES clause identifies the data to be entered
- If a column is omitted in the INSERT INTO clause, no data for that column is specified in the VALUES clause
- The data specified in the VALUES clause must be in the same sequence as the columns in the INSERT INTO clause

# Insert Command

- If no columns are specified in the INSERT INTO clause, a value is required for each column in the table, in the same order as the columns appear in the table
- If more than one column is listed, column names must be *separated by commas*
- If more than one data value is entered, they must be *separated by commas*
- As with the SELECT statement and the WHERE clause, *non-numeric data is enclosed in single quotation marks* and *numeric data requires no quotation marks*

# Insert Command

Worksheet | Query Builder

```
1  INSERT INTO acctmanager
2  VALUES ('T500', 'NICK', 'TAYLOR', '05-SEP-09', 42000, 3500, 'NE');
3
4  SELECT *
5  FROM acctmanager;
```

Script Output

Task completed in 1.061 seconds

```
1 rows inserted.
AMID AMFIRST      AMLAST       AMEDATE        AMSAL     AMCOMM REGION
---- ------------ ------------ --------- ---------- ---------- ------
T500 NICK         TAYLOR       05-SEP-09       42000       3500 NE
```

# Insert Command

- The INSERT statement issued on the previous slide requires no list of columns in the INSERT INTO clause since data is provided for all the required columns of the table
- The character data in the VALUES clause is entered in all upper case characters, so this is how it will be stored in the table
- *It is important to be consistent with respect to the case of the letters when entering character data values*
- Oracle will respond with a “1 row created” message confirming the insertion of the row of data
- A SELECT statement confirms that the record is inserted

# Insert Command

Worksheet | Query Builder

```
1  INSERT INTO acctmanager
2  VALUES (B100, 'BILL', 'WRIGHT', '05-SEP-92', 92000, 7500, 'NE');
3
```

Script Output

Task completed in 0.171 seconds

```
Error starting at line : 1 in command -
INSERT INTO acctmanager
VALUES (B100, 'BILL', 'WRIGHT', '05-SEP-92', 92000, 7500, 'NE')
Error at Command Line : 2 Column : 9
Error report -
SQL Error: ORA-00984: column not allowed here
00984. 00000 -  "column not allowed here"
*Cause:
*Action:
```

The above INSERT statements show an error caused by omitting the single quotations around a character field

# Insert Command

Worksheet | Query Builder

```
1  INSERT INTO acctmanager
2  VALUES ('B100', 'BILL, 'WRIGHT', '05-SEP-92', 92000, 7500, 'NE');
3
```

Script Output

Task completed in 0.141 seconds

```
Error starting at line : 1 in command -
INSERT INTO acctmanager
VALUES ('B100', 'BILL, 'WRIGHT', '05-SEP-92', 92000, 7500, 'NE');
Error at Command Line : 2 Column : 25
Error report -
SQL Error: ORA-00917: missing comma
00917. 00000 -  "missing comma"
*Cause:
*Action:
```

In this case the single quotation was omitted from the one side of BILL. Results in the above error, Could also display a ORA_01756 error, quoted string not properly terminated error

# Insert Command

- The record to be entered into the ACCTMANAGER table on the next slide contains data for Mandy Lopez. As our initial data indicates, there is no value for REGION, we want it to be NULL
- There are several ways to enter a NULL value
- All three methods shown on the next few slides are valid methods for entering NULL values
- But, no matter which method is used they all work, the record will be deleted each time and reentered using the three different methods

# Insert Command

Worksheet | Query Builder

```
INSERT INTO acctmanager (amid, amfirst, amlast, amedate, amsal, amcomm)
VALUES ('L500', 'MANDY', 'LOPEZ', '01-OCT-09', 47000, 1500);
```

Script Output

Task completed in 0.046 seconds

```
1 rows inserted.
```

In this case a column list is provided specifying the columns to be inserted to the table, notice the REGION is omitted, and so is it value

The result will be a NULL value being inserted for the REGION column

# Insert Command

Worksheet | Query Builder

```
1  INSERT INTO acctmanager
2  VALUES ('L500', 'MANDY', 'LOPEZ', '01-OCT-09', 47000, 1500, '');
3
```

Script Output x

Task completed in 0.047 seconds

```
1 rows inserted.
```

In the second case, there is no column list so the INSERT is expecting a value for each column

For the REGION column only two single quotes are specified, ‘’ there is no space between the single quotes, this indicates a NULL value is to be inserted for the column

# Insert Command

Worksheet | Query Builder

```
1  INSERT INTO acctmanager
2  VALUES ('L500', 'MANDY', 'LOPEZ', '01-OCT-09', 47000, 1500, NULL);
3
```

Script Output

Task completed in 0.046 seconds

```
1 rows inserted.
```

In the third case there is no column list so a value is needed for each column

In this case the REGION vlaue is specified as a NULL, the keyword NULL can be used to indicate that a NULL value is being entered for the REGION

# Insert Command

- The next record to be inserted to the ACCTMANAGER table will be done by providing a list of the columns that require values in the INSERT INTO clause
- When the row is inserted in this fashion, the actual values in the VALUES clause must be in entered in the same sequence they are provided in the INSERT INTO clause
- The value for the AMEDATE is omitted, when the table was created we gave it a DEFAULT value of SYSDATE

# Insert Command

Worksheet | Query Builder

```
1  INSERT INTO acctmanager (amid, amfirst, amlast, amsal, amcomm, region)
2  VALUES ('J500', 'Sammie', 'Jones', 39500, 2000, 'NW');
3
4  SELECT *
5  FROM acctmanager;
6
```

Script Output

Task completed in 0.094 seconds

```
1 rows inserted.
AMID AMFIRST      AMLAST       AMEDATE        AMSAL     AMCOMM REGION
---- ------------ ------------ --------- ---------- ---------- ------
T500 NICK         TAYLOR       05-SEP-09      42000       3500 NE
L500 MANDY        LOPEZ        01-OCT-09      47000       1500
J500 Sammie       Jones        22-FEB-15      39500       2000 NW
```

The output indicates a problem with the data in the way it was entered. Also, notice the value for date, where did it come from, it was omitted! Answer: the DEFAULT of SYSDATE was inserted for the omitted column

# Insert Command

Worksheet | Query Builder

```
1  INSERT INTO acctmanager (amid, amfirst, amlast, amedate, amsal, amcomm, region)
2  VALUES ('J500', 'Sammie', 'Jones', DEFAULT, 39500, 2000, 'NW');
3
4  SELECT *
5  FROM acctmanager;
6
```

Script Output x

Task completed in 0.094 seconds

```
1 rows inserted.
AMID AMFIRST      AMLAST       AMEDATE        AMSAL     AMCOMM REGION
---- ------------ ------------ --------- ---------- ---------- ------
T500 NICK         TAYLOR       05-SEP-09      42000       3500 NE
L500 MANDY        LOPEZ        01-OCT-09      47000       1500
J500 Sammie       Jones        22-FEB-15      39500       2000 NW
```

In this case all columns are specified and the DEFAULT keyword is used for the AMEDATE column, specifying to use the DEFAULT value of SYSDATE

# Insert Command

Worksheet | Query Builder

```sql
INSERT INTO acctmanager (amid, amfirst, amlast, amedate, amsal, amcomm, region)
VALUES ('J500', 'Sammie', 'Jones', SYSDATE, 39500, 2000, 'NW');

SELECT *
FROM acctmanager;
```

Script Output

Task completed in 0.093 seconds

```
1 rows inserted.
AMID AMFIRST      AMLAST       AMEDATE        AMSAL     AMCOMM REGION
---- ------------ ------------ --------- ---------- ---------- ------
T500 NICK         TAYLOR       05-SEP-09      42000       3500 NE
L500 MANDY        LOPEZ        01-OCT-09      47000       1500
J500 Sammie       Jones        22-FEB-15      39500       2000 NW
```

You can actually specify that the SYSDATE value be inserted to the column directly, this is a keyword that will provide the current system date from the server

# Insert Command

- The data inserted for Sammie Jones is in mixed case. This would cause problems using the SELECT statement, since the data entered should be consistent. *Remember that stored data is case sensitive.* The inconsistent case for AMFIRST and AMLAST will have to be fixed, this will be done later in this lecture
- A date also appears in the record for Sammie Jones. This is because there was a default value for the AMEDATE column. Since DEFAULT says if it was left as a NULL, SYSDATE is to be inserted in its place

# Insert with Virtual Columns

- A virtual column is a column that the value is generated based on the values of other columns
- In other words the database system generates the value for the column automatically based on the manipulation or a calculation defined for the column
- If a virtual column is included in a table in will affect how the INSERT command can be used
- The virtual column must be ignored in an INSERT command

# Insert with Virtual Columns

- A new column is going to be added to our table called AMEARN
- It is going to contain the total earnings for the account manager, this is the SALARY plus the COMMISSION
- A query will be used against the table to see how the new virtual column behaves

# Insert with Virtual Columns

Worksheet | Query Builder

```
1  ALTER TABLE acctmanager
2  ADD amearn AS (amsal + amcomm);
```

Script Output

Task completed in 0.062 seconds

```
table ACCTMANAGER altered.
```

New virtual column AMEARN added to the table as a calculation

# Insert with Virtual Columns

Worksheet | Query Builder

```
1 SELECT *
2 FROM acctmanager;
```

Script Output | Query Result

SQL | All Rows Fetched: 3 in 0.047 seconds

| | AMID | AMFIRST | AMLAST | AMEDATE | AMSAL | AMCOMM | REGION | AMEARN |
|---|---|---|---|---|---|---|---|---|
| 1 | T500 | NICK | TAYLOR | 05-SEP-09 | 42000 | 3500 | NE | 45500 |
| 2 | L500 | MANDY | LOPEZ | 01-OCT-09 | 47000 | 1500 | (null) | 48500 |
| 3 | J500 | Sammie | Jones | 22-FEB-15 | 39500 | 2000 | NW | 41500 |

When used in a SELECT the calculation is performed that defined the virtual column and the result of the calculation is displayed

# Insert with Virtual Columns

Worksheet | Query Builder

```
1  INSERT INTO acctmanager (amid, amfirst, amlast, amsal, amcomm, region, amearn)
2  VALUES ('D500', 'SCOTT', 'DAVIS', 53000, 6000, 'NW', 59000);
3
```

Script Output

Task completed in 0.141 seconds

```
Error starting at line : 1 in command -
INSERT INTO acctmanager (amid, amfirst, amlast, amsal, amcomm, region, amearn)
VALUES ('D500', 'SCOTT', 'DAVIS', 53000, 6000, 'NW', 59000)
Error at Command Line : 1 Column : 72
Error report -
SQL Error: ORA-54013: INSERT operation disallowed on virtual columns
54013. 0000 -  "INSERT operation disallowed on virtual columns"
*Cause:    Attempted to insert values into a virtual column
*Action:   Re-issue the statment without providing values for a virtual column
```

With our newly defined virtual column an attempt to insert a value to the virtual column was unsuccessful, an error was presented as a result

# Insert with Virtual Columns

Worksheet | Query Builder

```
INSERT INTO acctmanager (amid, amfirst, amlast, amsal, amcomm, region)
VALUES ('D500', 'SCOTT', 'DAVIS', 53000, 6000, 'NW');

SELECT *
FROM acctmanager;
```

Script Output

Task completed in 0.109 seconds

```
1 rows inserted.
AMID AMFIRST      AMLAST       AMEDATE        AMSAL     AMCOMM REGION     AMEARN
---- ------------ ------------ --------- ---------- ---------- ------ ----------
T500 NICK         TAYLOR       05-SEP-09      42000       3500 NE          45500
L500 MANDY        LOPEZ        01-OCT-09      47000       1500             48500
J500 Sammie       Jones        22-FEB-15      39500       2000 NW          41500
D500 SCOTT        DAVIS        22-FEB-15      53000       6000 NW          59000
```

A value is never inserted to a virtual column, as you see the SELECT shows the result of the virtual column

# Handling Single Quotes in Insert

- Inserting values that contain single quotes raises an error because they are confused with the single quotations used to enclose character data and date values
- How would be insert the name Peg O’Hara?
- Let’s try

# Handling Single Quotes in Insert

Worksheet | Query Builder

```
1  INSERT INTO acctmanager (amid, amfirst, amlast, amsal, amcomm, region)
2  VALUES ('M500', 'Peg', 'O'Hara', 46000, 2000, 'SW');
3
```

Script Output

Task completed in 0.14 seconds

```
Error starting at line : 1 in command -
INSERT INTO acctmanager (amid, amfirst, amlast, amsal, amcomm, region)
VALUES ('M500', 'Peg', 'O'Hara', 46000, 2000, 'SW');
Error at Command Line : 2 Column : 27
Error report -
SQL Error: ORA-00917: missing comma
00917. 00000 -  "missing comma"
*Cause:
*Action:
```

When a single quote is inserted it produces a ORA-00917 error

# Handling Single Quotes in Insert

- The error states that the issue is a missing comma
- The single quote is treated as the closing quote for the string ‘O’ for the last name
- To instruct Oracle to treat a single quote as part of the string value enter two single quotes together in the value, do not use the double quote!!!
- The result will appear as O’’Hara

# Handling Single Quotes in Insert

Worksheet | Query Builder

```
1  INSERT INTO acctmanager (amid, amfirst, amlast, amsal, amcomm, region)
2  VALUES ('M500', 'Peg', 'O''Hara', 46000, 2000, 'SW');
3
```

Script Output

Task completed in 0.031 seconds

```
1 rows inserted.
```

With this modification provided the row is successfully inserted

# Handling Single Quotes in Insert

Worksheet | Query Builder

```
1 SELECT *
2 FROM acctmanager;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.047 seconds

| | AMID | AMFIRST | AMLAST | AMEDATE | AMSAL | AMCOMM | REGION | AMEARN |
|---|---|---|---|---|---|---|---|---|
| 1 | T500 | NICK | TAYLOR | 05-SEP-09 | 42000 | 3500 | NE | 45500 |
| 2 | L500 | MANDY | LOPEZ | 01-OCT-09 | 47000 | 1500 | (null) | 48500 |
| 3 | J500 | Sammie | Jones | 22-FEB-15 | 39500 | 2000 | NW | 41500 |
| 4 | D500 | SCOTT | DAVIS | 22-FEB-15 | 53000 | 6000 | NW | 59000 |
| 5 | M500 | Peg | O'Hara | 22-FEB-15 | 46000 | 2000 | SW | 48000 |

Table data shown with our new row inserted with a single quote in he last name

# Inserting Data from an Existing Table

- Previously, we used the CREATE TABLE command to create a new table with a subquery that creates and populates the table based on the structure and content of an existing table
- If the table already exists and all you need to do is records to the table, you can use the *INSERT INTO command with a subquery*

# Inserting Data from an Existing Table

```
INSERT INTO tablename [(columnname, …)]
subquery;
```

# Inserting Data from an Existing Table

- The main difference between the INSERT INTO command with actual data values and a subquery is that the VALUES *clause is not included when the command is used with a subquery*
- The data are derived from the results of the subquery and the VALUES clause would actually *indicate that there are actual data values to enter into the table*
- Unlike other commands, the INSERT INTO command *does not require the subquery to be enclosed in parentheses*

# Inserting Data from an Existing Table

Worksheet | Query Builder

```
1  CREATE TABLE acctbonus
2  (
3  amid          CHAR(4),
4  amsal         NUMBER(8,2),
5  region        CHAR(2)
6  );
```

Script Output

Task completed in 0.063 seconds

A new table is created called ACCTBONUS, only has the three columns but no data

# Inserting Data from an Existing Table

Worksheet | Query Builder

```
1  INSERT INTO acctbonus (amid, amsal, region)
2         SELECT amid, amsal, region
3           FROM acctmanager;
```

Script Output x | Query Result x

Task completed in 0.063 seconds

```
5 rows inserted.
```

# Inserting Data from an Existing Table

- Having this data in a separate table allows the IT department to work on the data in the new table without interfering with the actual data records in the original ACCTMANAGER table
- Testing can be done on the new table without interfering with the records personnel needs in ACCTMANAGER and changes can be made to the original ACCTMANAGER table without interfering with the data in the ACCTBONUS table

# Inserting Data from an Existing Table

- The SELECT clause of the subquery lists the columns to be copied from the original table (ACCTMANAGER) which is identified in the FROM clause
- There is no column list provided with the INSERT INTO clause since the columns returned by the subquery are in the same order as the columns in the new table definition (ACCTBONUS)
- This new table can now be queried as shown

# Inserting Data from an Existing Table

Worksheet | Query Builder

```
1  SELECT *
2  FROM acctbonus;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.047 seconds

| | AMID | AMSAL | REGION |
|---|---|---|---|
| 1 | T500 | 42000 | NE |
| 2 | L500 | 47000 | (null) |
| 3 | J500 | 39500 | NW |
| 4 | D500 | 53000 | NW |
| 5 | M500 | 46000 | SW |

Our new table with the 5 rows of data added to it

# Modifying Existing Rows

- There will be times when record data needs to be changed
- People move, prices change, etc.
- INSERT INTO is only used to add records to a table, it does not allow modification to existing data
- The UPDATE command is used to *modify data in existing rows*
- We will look at *how to do updates*, and *how to create interactive scripts to be used for performing updates through substitution variables*

# Update Command

```
UPDATE tablename
SET columnname = new_datavalue, …
[WHERE condition];
```

# UPDATE Command

- The UPDATE clause identifies the table containing the record(s) to be changed
- The SET clause is used to identify the column to be changed and the new value to be assigned to it
- More than one column and value can be specified, separated by commas
- The optional WHERE clause identifies the exact row to be changed by the UPDATE command
- If the WHERE clause is omitted, then the column specified in the SET clause will update all records in the specified table

# UPDATE Command

Worksheet | Query Builder

```
1  UPDATE acctmanager
2  SET amedate = '01-AUG-09'
3  WHERE amid = 'J500';
```

Script Output | Query Result

Task completed in 0.047 seconds

```
1 rows updated.
```

# UPDATE Command

- The SET clause in the previous slide changes the date to be modified
- When the SELECT statement is used to view the data, the date appears as the new date value
- The WHERE clause is used to identify exactly which record should be altered, the primary key field is used to update only the specified record

# UPDATE Command

Worksheet | Query Builder

```
1  UPDATE acctmanager
2  SET amfirst = UPPER(amfirst),
3      amlast = UPPER(amlast);
```

Script Output | Query Result

Task completed in 0.046 seconds

```
5 rows updated.
```

The above UPDATE command updates all records in the ACCTMANAGER table, since no WHERE clause is specified as a condition, more on UPPER Function will be shown in future

# UPDATE Command

Worksheet | Query Builder

```
SELECT *
FROM acctmanager;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.046 seconds

| | AMID | AMFIRST | AMLAST | AMEDATE | AMSAL | AMCOMM | REGION | AMEARN |
|---|---|---|---|---|---|---|---|---|
| 1 | T500 | NICK | TAYLOR | 05-SEP-09 | 42000 | 3500 | NE | 45500 |
| 2 | L500 | MANDY | LOPEZ | 01-OCT-09 | 47000 | 1500 | (null) | 48500 |
| 3 | J500 | SAMMIE | JONES | 01-AUG-09 | 39500 | 2000 | NW | 41500 |
| 4 | D500 | SCOTT | DAVIS | 22-FEB-15 | 53000 | 6000 | NW | 59000 |
| 5 | M500 | PEG | O'HARA | 22-FEB-15 | 46000 | 2000 | SW | 48000 |

Modifications made to our data, can you see the differences?

# UPDATE Command

Worksheet | Query Builder

```
1  UPDATE acctmanager
2  SET region = 'W'
3  WHERE region IN ('NE', 'NW');
```

Script Output | Query Result

Task completed in 0.062 seconds

```
3 rows updated.
```

The above UPDATE command used to update REGION to W, the WHERE clause specifies which rows are to be updated

# UPDATE Command

Worksheet | Query Builder

```
1  SELECT *
2  FROM acctmanager;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.031 seconds

| | AMID | AMFIRST | AMLAST | AMEDATE | AMSAL | AMCOMM | REGION | AMEARN |
|---|---|---|---|---|---|---|---|---|
| 1 | T500 | NICK | TAYLOR | 05-SEP-09 | 42000 | 3500 | W | 45500 |
| 2 | L500 | MANDY | LOPEZ | 01-OCT-09 | 47000 | 1500 | (null) | 48500 |
| 3 | J500 | SAMMIE | JONES | 01-AUG-09 | 39500 | 2000 | W | 41500 |
| 4 | D500 | SCOTT | DAVIS | 22-FEB-15 | 53000 | 6000 | W | 59000 |
| 5 | M500 | PEG | O'HARA | 22-FEB-15 | 46000 | 2000 | SW | 48000 |

Table contents with changes shown

# Substitution Variables

- In some cases, it may seem like a great deal of effort to add a record to a table or even update a record
- This can be very true if you need to modify many different records
- For example, say a table called CUSTOMER table now needs to contain data that identifies the marketing region for each customer
- After the Region column is added to the table using the ALTER TABLE command, every *customer record will need to be updated with the value for the new column*

# Substitution Variables

Worksheet | Query Builder

```
SELECT *
FROM customer;
```

Script Output | Query Result

SQL | All Rows Fetched: 24 in 0.049 seconds

| | CUSTOMER_ID | CUSTOMER_LAST_NAME | CUSTOMER_FIRST_NAME | CUSTOMER_ADDRESS | CUSTOMER_CITY | CUSTOMER_STATE | CUSTOMER_ZIP | CUSTOMER_PHONE | REGION |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Anders | Maria | 345 Winchell Pl | Anderson | IN | 46014 | (765) 555-7878 | (null) |
| 2 | 2 | Trujillo | Ana | 1298 E Smathers St | Benton | AR | 72018 | (501) 555-7733 | (null) |
| 3 | 3 | Moreno | Antonio | 6925 N Parkland Ave | Puyallup | WA | 98373 | (253) 555-8332 | (null) |
| 4 | 4 | Hardy | Thomas | 83 d'Urberville Ln | Casterbridge | GA | 31209 | (478) 555-1139 | (null) |
| 5 | 5 | Berglund | Christina | 22717 E 73rd Ave | Dubuque | IA | 52004 | (319) 555-1139 | (null) |
| 6 | 6 | Moos | Hanna | 1778 N Bovine Ave | Peoria | IL | 61638 | (309) 555-8755 | (null) |
| 7 | 7 | Citeaux | Fred | 1234 Main St | Normal | IL | 61761 | (309) 555-1914 | (null) |
| 8 | 8 | Summer | Martin | 1877 Ete Ct | Frogtown | LA | 70563 | (337) 555-9441 | (null) |
| 9 | 9 | Lebihan | Laurence | 717 E Michigan Ave | Chicago | IL | 60611 | (312) 555-9441 | (null) |
| 10 | 10 | Lincoln | Elizabeth | 4562 Rt 78 E | Vancouver | WA | 98684 | (360) 555-2680 | (null) |
| 11 | 11 | Snyder | Howard | 2732 Baker Blvd. | Eugene | OR | 97403 | (503) 555-7555 | (null) |
| 12 | 12 | Latimer | Yoshi | City Center Plaza 516 Main St. | Elgin | OR | 97827 | (503) 555-6874 | (null) |
| 13 | 13 | Steel | John | 12 Orchestra Terrace | Walla Walla | WA | 99362 | (509) 555-7969 | (null) |
| 14 | 14 | Yorres | Jaime | 87 Polk St. Suite 5 | San Francisco | CA | 94117 | (415) 555-5938 | (null) |
| 15 | 15 | Wilson | Fran | 89 Chiaroscuro Rd. | Portland | OR | 97219 | (503) 555-9573 | (null) |
| 16 | 16 | Phillips | Rene | 2743 Bering St. | Anchorage | AK | 99508 | (907) 555-7584 | (null) |
| 17 | 17 | Wilson | Paula | 2817 Milton Dr. | Albuquerque | NM | 87110 | (505) 555-5939 | (null) |
| 18 | 18 | Pavarotti | Jose | 187 Suffolk Ln. | Boise | ID | 83720 | (208) 555-8097 | (null) |
| 19 | 19 | Braunschweiger | Art | P.O. Box 555 | Lander | WY | 82520 | (307) 555-4680 | (null) |
| 20 | 20 | Nixon | Liz | 89 Jefferson Way Suite 2 | Providence | RI | 02909 | (401) 555-3612 | (null) |
| 21 | 21 | Wong | Liu | 55 Grizzly Peak Rd. | Butte | MT | 59801 | (406) 555-5834 | (null) |
| 22 | 22 | Nagy | Helvetius | 722 DaVinci Blvd. | Concord | MA | 01742 | (351) 555-1219 | (null) |
| 23 | 23 | Jablonski | Karl | 305 - 14th Ave. S. Suite 3B | Seattle | WA | 98128 | (206) 555-4112 | (null) |
| 24 | 24 | Chelan | Donna | 2299 E Baylor Dr | Dallas | TX | 75224 | (469) 555-8828 | (null) |

The REGION column has been added, it contains NULLs

# Substitution Variables

Worksheet | Query Builder

```
1 UPDATE customer
2 SET region = 'NE'
3 WHERE customer_state = 'MA';
```

Script Output | Query Result

Task completed in 0.052 seconds

```
1 rows updated.
```

UPDATE can be used to set the Region value for each column. However, this is tedious since the command must be reentered for each state

# Substitution Variables

- Rather than type the same command again and again to modify the column with the necessary values, we can use substitution variables
- A substitution variable in a SQL command instructs Oracle to use a substituted value in place of the variable when command is executed
- To include a substitution variable in a SQL command, use the ampersand (&) followed by the name to be used for the variable
- *This value will be substituted with actual data when the command is executed*

# Substitution Variables

Worksheet | Query Builder

```
1 UPDATE customer
2 SET region = '&region'
3 WHERE customer_state = '&state';
```

Script Output | Query Result

ScriptRunner Task

Enter Substitution Variable

REGION:

NE

OK Cancel

When the UPDATE is executed, the user is prompted to provide values at execution time.

A dialog box appears to prompt for the REGION, a value of NE is entered, click OK

# Substitution Variables

Worksheet | Query Builder

```
1 UPDATE customer
2 SET region = '&region'
3 WHERE customer_state = '&state';
```

Script Output | Query Result

ScriptRunner Task

Enter Substitution Variable

STATE:

RI

OK Cancel

Since we had two variables, you are now prompted to enter the value for the STATE, in this case RI

# Substitution Variables

Worksheet | Query Builder

```
1  UPDATE customer
2  SET region = '&region'
3  WHERE customer_state = '&state';
```

Script Output | Query Result

Task completed in 0.045 seconds

```
old:UPDATE customer
SET region = '&region'
WHERE customer_state = '&state'
new:UPDATE customer
SET region = 'NE'
WHERE customer_state = 'RI'
1 rows updated.
```

You are then presented with details of the substitution that will be used in your UPDATE command

You only need to execute the command again and enter new values

This could be saved as a script then reloaded to use it over

# Substitution Variables

Worksheet | Query Builder

```
1 UPDATE customer
2 SET region = '&region'
3 WHERE customer_state = '&state';
```

Script Output | Query Result

Task completed in 0.046 seconds

```
old:UPDATE customer
SET region = '&region'
WHERE customer_state = '&state'
new:UPDATE customer
SET region = 'SE'
WHERE customer_state = 'GA'
1 rows updated.
```

The command was executed a second time, this time the SE region was entered and GA for the state of Georgia

# Substitution Variables

- If you did not want to or have the time to update all regions at the same time, you could create a script file of the command as we discussed earlier
- Once created, the script can be loaded and executed whenever you need it

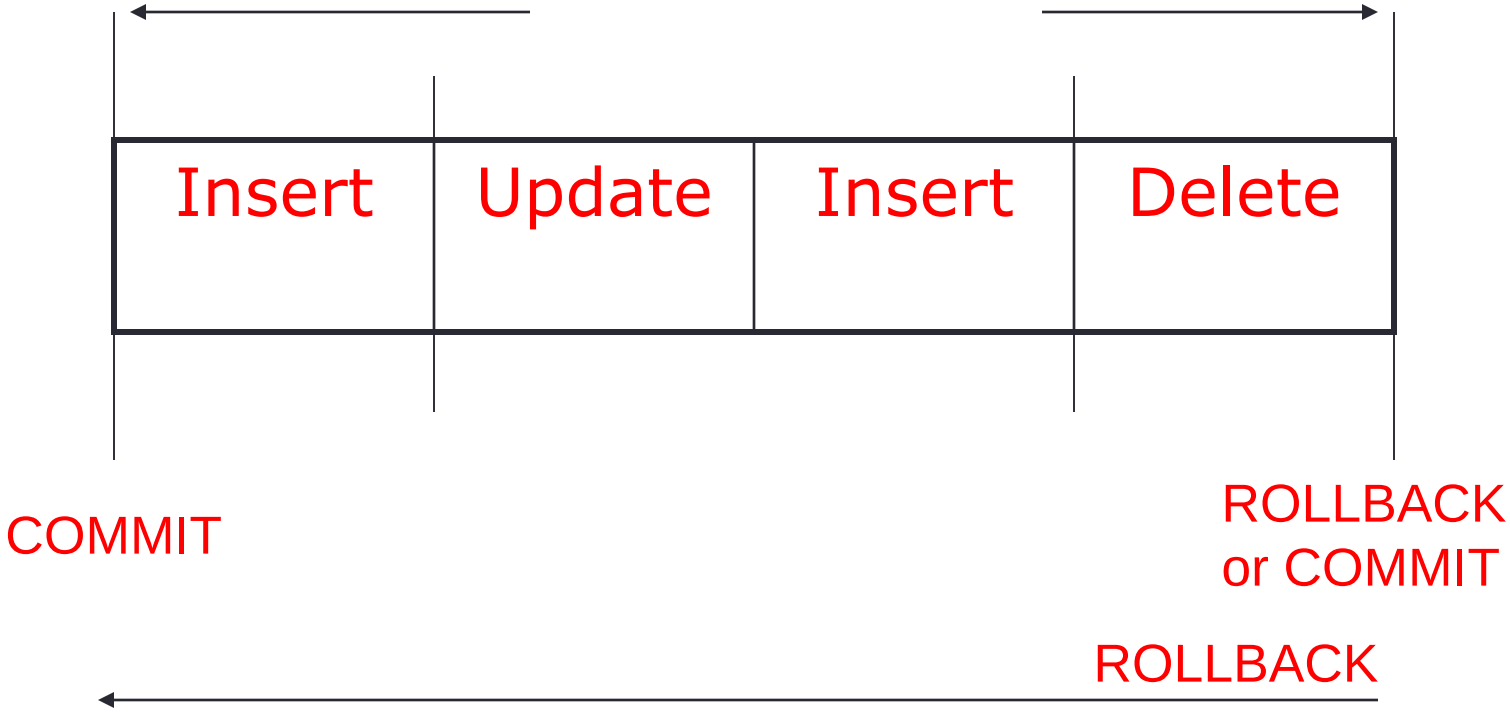
# Transaction Control Statements

- In Chapters 3 and 4, we issued DDL commands to create, alter and drop database objects
- In this chapter, so far we have performed operations with the INSERT INTO and UPDATE commands
- These commands are called DML (*Data Manipulation Language*) commands
- Changes made to data by DML commands are *not permanently saved to the table when the SQL statement is executed*
- This allows the user the flexibility of issuing a transaction control statement either to save the modified data or to undo the changes made

# Creating Transactions and Committing New Data

- Transaction
  - Logical unit of work consisting of one or more SQL DML commands
    - INSERT, UPDATE, DELETE
  - All transaction commands must succeed or none can succeed
- Transaction results are not visible to other users until they are "committed" to the database
- Until a transaction is committed, it can easily be "rolled back" (undone)

# Controlling Transactions




A transaction consists of any number of DML commands. A transaction must either be committed or rolled back. COMMIT saves transactions, ROLLBACK reverses the DML commands

# Transaction

- A transaction starts when you type one or more DML commands
- A transaction ends when you issue either the COMMIT or ROLLBACK command
- Either all commands in the transaction COMMIT or they all ROLLBACK when either of the following commands are used

```
COMMIT;
ROLLBACK;
```

# Transactions

- When a transaction is started, each subsequent DML command is executed and the database is updated
- Information is also recorded to allow the database to return to its state as of the last time it was committed
- If a COMMIT is issued, all DML statements issued in the transaction will be saved and the changes become permanent
- If a ROLLBACK is issued, all DML statements issued in the transaction will be undone, the changes discarded, and the data is returned to its last committed state

# COMMIT and ROLLBACK Commands

Worksheet | Query Builder

```
UPDATE customer
SET region = 'NW'
WHERE customer_state = 'OR';

SELECT *
FROM customer
WHERE customer_state = 'OR';

ROLLBACK;

SELECT *
FROM customer
WHERE customer_state = 'OR';
```

Script Output | Query Result

Task completed in 0.204 seconds

```
3 rows updated.
CUSTOMER_ID CUSTOMER_LAST_NAME          CUSTOMER_FIRST_NAME         CUSTOMER_ADDRESS                                        CUSTOMER_CITY   CUSTOMER_STATE  CUSTOMER_ZIP CUSTOMER_PHONE         REGION
----------- ----------------------------- ----------------------------- ------------------------------------------------------- --------------- --------------- ------------ ------------------------ ------
         11 Snyder                      Howard                      2732 Baker Blvd.                                        Eugene          OR              97403        (503) 555-7555          NW
         12 Latimer                     Yoshi                       City Center Plaza 516 Main St.                          Elgin           OR              97827        (503) 555-6874          NW
         15 Wilson                      Fran                        89 Chiaroscuro Rd.                                      Portland        OR              97219        (503) 555-9573          NW

rollback complete.
CUSTOMER_ID CUSTOMER_LAST_NAME          CUSTOMER_FIRST_NAME         CUSTOMER_ADDRESS                                        CUSTOMER_CITY   CUSTOMER_STATE  CUSTOMER_ZIP CUSTOMER_PHONE         REGION
----------- ----------------------------- ----------------------------- ------------------------------------------------------- --------------- --------------- ------------ ------------------------ ------
         11 Snyder                      Howard                      2732 Baker Blvd.                                        Eugene          OR              97403        (503) 555-7555
         12 Latimer                     Yoshi                       City Center Plaza 516 Main St.                          Elgin           OR              97827        (503) 555-6874
         15 Wilson                      Fran                        89 Chiaroscuro Rd.                                      Portland        OR              97219        (503) 555-9573
```

An UPDATE was done to the CUSTOMER table, then the changes were undone with the ROLLBACK command

# COMMIT and ROLLBACK Commands

Worksheet | Query Builder

```
UPDATE customer
SET region = 'NW'
WHERE customer_state = 'WA';

SELECT *
FROM customer
WHERE customer_state = 'WA';

COMMIT;

SELECT *
FROM customer
WHERE customer_state = 'WA';
```

Script Output | Query Result

Task completed in 0.182 seconds

```
4 rows updated.
CUSTOMER_ID CUSTOMER_LAST_NAME          CUSTOMER_FIRST_NAME         CUSTOMER_ADDRESS                                        CUSTOMER_CITY   CUSTOMER_STATE CUSTOMER_ZIP CUSTOMER_PHONE         REGION
----------- ----------------------------- ----------------------------- ------------------------------------------------------- --------------- -------------- ------------ ---------------------- ------
          3 Moreno                      Antonio                     6925 N Parkland Ave                                     Puyallup        WA             98373        (253) 555-8332         NW
         10 Lincoln                     Elizabeth                   4562 Rt 78 E                                            Vancouver       WA             98684        (360) 555-2680         NW
         13 Steel                       John                        12 Orchestra Terrace                                    Walla Walla     WA             99362        (509) 555-7969         NW
         23 Jablonski                   Karl                        305 - 14th Ave. S. Suite 3B                             Seattle         WA             98128        (206) 555-4112         NW

committed.
CUSTOMER_ID CUSTOMER_LAST_NAME          CUSTOMER_FIRST_NAME         CUSTOMER_ADDRESS                                        CUSTOMER_CITY   CUSTOMER_STATE CUSTOMER_ZIP CUSTOMER_PHONE         REGION
----------- ----------------------------- ----------------------------- ------------------------------------------------------- --------------- -------------- ------------ ---------------------- ------
          3 Moreno                      Antonio                     6925 N Parkland Ave                                     Puyallup        WA             98373        (253) 555-8332         NW
         10 Lincoln                     Elizabeth                   4562 Rt 78 E                                            Vancouver       WA             98684        (360) 555-2680         NW
         13 Steel                       John                        12 Orchestra Terrace                                    Walla Walla     WA             99362        (509) 555-7969         NW
         23 Jablonski                   Karl                        305 - 14th Ave. S. Suite 3B                             Seattle         WA             98128        (206) 555-4112         NW
```

An UPDATE was done to the CUSTOMER table, then the changes were made permanent with the COMMIT command

# Transaction Processing

- Transaction processing enables every user to see a consistent view of the database
- To achieve this, a user cannot view or update data values that are involved in another user's uncommitted transactions because the pending transactions might be rolled back

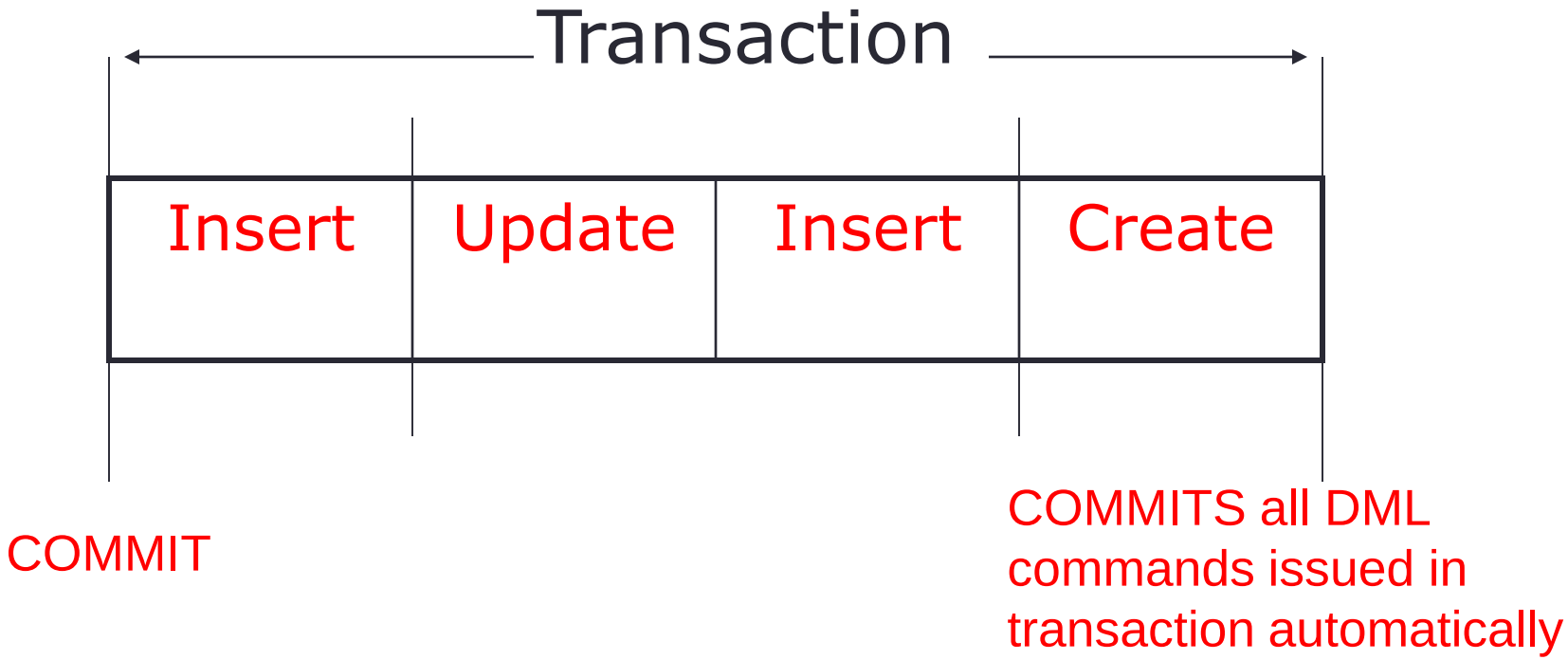
# Transaction Processing

- Transaction processing is implemented by locking data records that are involved in uncommitted update or delete operations. This prohibits other users from viewing or modifying them
- When the transaction is committed or rolled back, the locks on the data record are released and the changed data values are available to other users

# Implicit Transaction Processing

- An automatic commit occurs under the following circumstances:
  - DDL statement is issued
  - DCL statement is issued (Data Control Language - these are related to user permissions)
  - Normal exit from SQL DEVLOPER, you will be prompted to COMMIT or ROLLBACK any changes, without explicitly issuing COMMIT or ROLLBACK
- An automatic rollback occurs under the following circumstances:
  - An abnormal termination of SQL Developer
  - A system failure

# Implicit Transaction Processing



# SAVEPOINT Command

- You can use rollbacks with savepoints that mark the beginning of individual sections of a transaction
- By using savepoints, you can roll back part of a transaction

# SAVEPOINT Command

Worksheet | Query Builder

```
INSERT INTO dept
VALUES (10, 'IT', '416 675-3111');

SAVEPOINT A;

INSERT INTO dept
VALUES (20, 'DEVELOPMENT', '416 675-6622');

SAVEPOINT B;

INSERT INTO dept
VALUES (30, 'HELP DESK', '416 675-8888');

INSERT INTO dept
VALUES (40, 'FOOD SERVICES', '416 675-3663');
```

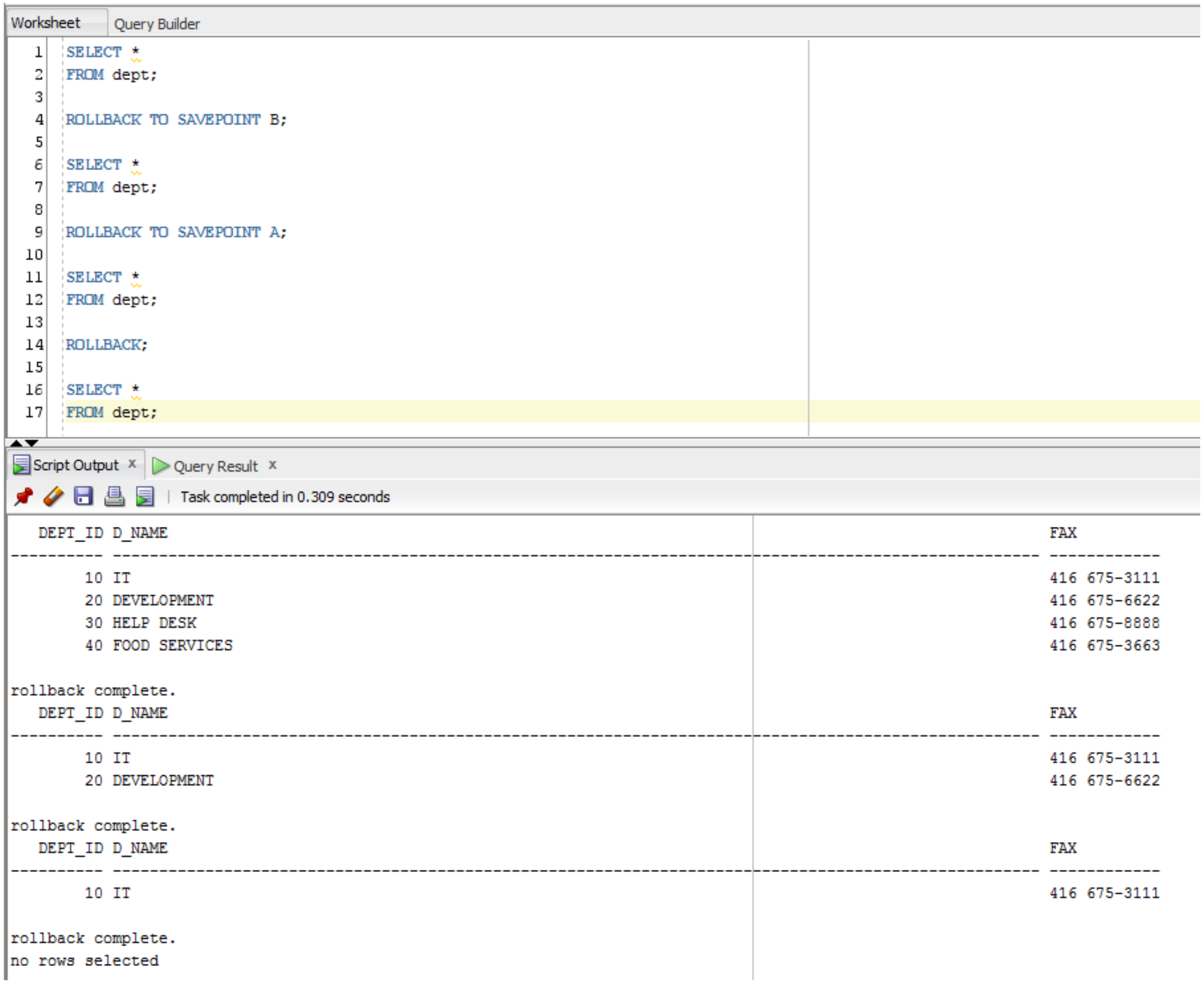
Script Output | Query Result

Task completed in 0.289 seconds

```
1 rows inserted.
SAVEPOINT A
1 rows inserted.
SAVEPOINT B
1 rows inserted.
1 rows inserted.
```

Two SAVEPOINTS were created, they were created within the transaction set of records

# SAVEPOINT Command



```sql
SELECT *
FROM dept;

ROLLBACK TO SAVEPOINT B;

SELECT *
FROM dept;

ROLLBACK TO SAVEPOINT A;

SELECT *
FROM dept;

ROLLBACK;

SELECT *
FROM dept;
```

Task completed in 0.309 seconds

```
   DEPT_ID D_NAME                                                                                               FAX
---------- ---------------------------------------------------------------------------------------------------- ------------
        10 IT                                                                                                   416 675-3111
        20 DEVELOPMENT                                                                                          416 675-6622
        30 HELP DESK                                                                                            416 675-8888
        40 FOOD SERVICES                                                                                        416 675-3663

rollback complete.
   DEPT_ID D_NAME                                                                                               FAX
---------- ---------------------------------------------------------------------------------------------------- ------------
        10 IT                                                                                                   416 675-3111
        20 DEVELOPMENT                                                                                          416 675-6622

rollback complete.
   DEPT_ID D_NAME                                                                                               FAX
---------- ---------------------------------------------------------------------------------------------------- ------------
        10 IT                                                                                                   416 675-3111

rollback complete.
no rows selected
```

All the rows are displayed, there are 4 of them

ROLLBACK to SAVEPOINT B

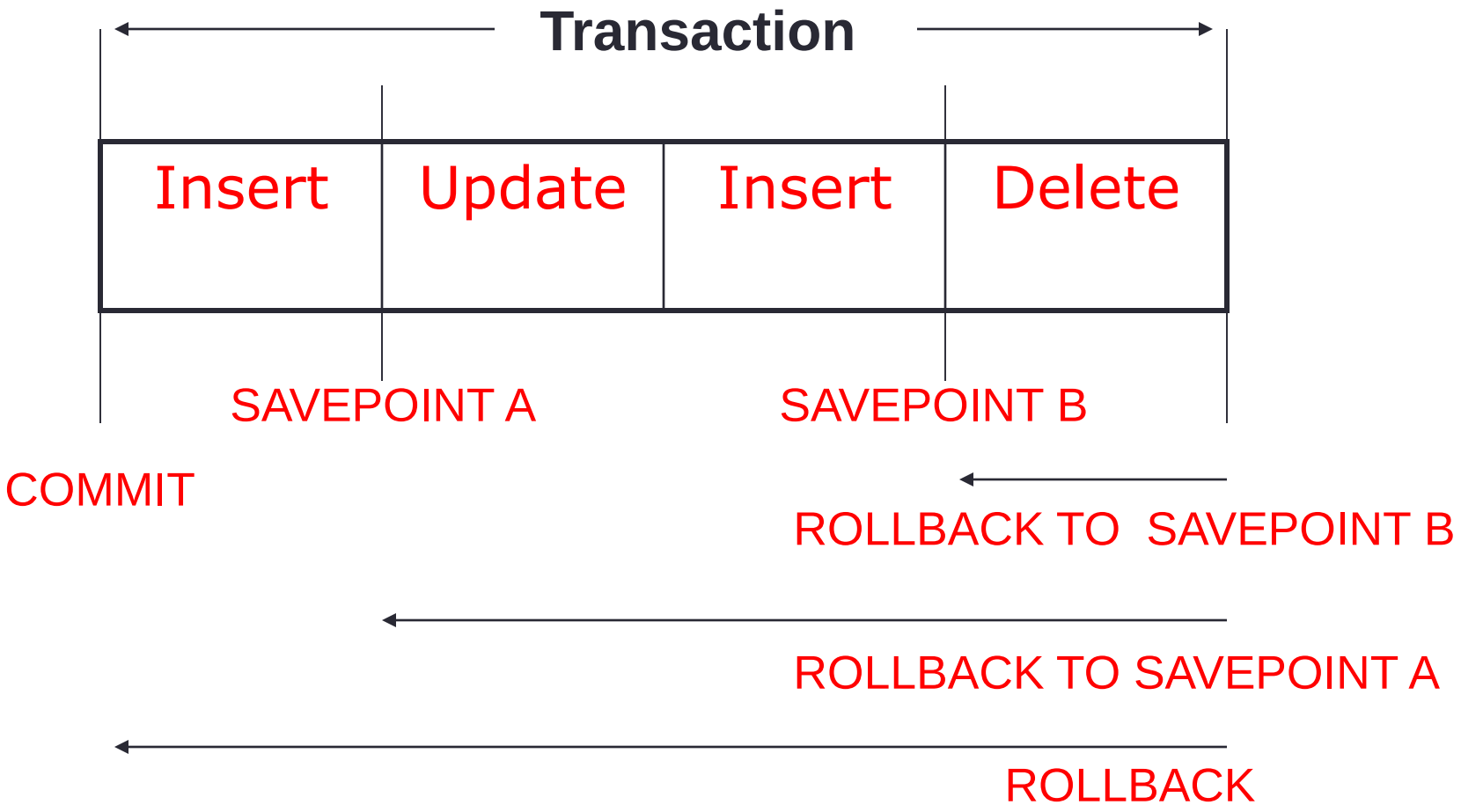
Rows are displayed again, now there are 2 of them

ROLLBACK to SAVEPOINT A

Display the rows, now there is only one row

ROLLBACK now; no rows displayed

# Controlling Transactions



# Deleting Rows

- There are times when rows need to be removed from the database tables
- This command is one of the easiest to use
- In fact, it may be a little too easy!

# Deleting Rows

```
DELETE FROM tablename
[WHERE condition];
```

# Deleting Rows

- The user does not have an opportunity to specify any columns in the DELETE command
- DELETE is used to remove an entire row
- The WHERE clause is optional as well and is used to identify the row or rows in the table
- If the WHERE clause is omitted, you will remove all the rows in the table

# Deleting Rows

0.21799999 seconds

Worksheet | Query Builder

```
1  DELETE FROM acctmanager
2  WHERE amid = 'D500';
3
4  SELECT *
5  FROM acctmanager;
6
7  ROLLBACK;
8
9  SELECT *
10 FROM acctmanager;
```

Script Output | Query Result

Task completed in 0.218 seconds

```
1 rows deleted.
AMID AMFIRST      AMLAST       AMEDATE        AMSAL     AMCOMM REGION     AMEARN
---- ------------ ------------ --------- ---------- ---------- ------ ----------
T500 NICK         TAYLOR       05-SEP-09      42000       3500 W           45500
L500 MANDY        LOPEZ        01-OCT-09      47000       1500             48500
J500 SAMMIE       JONES        01-AUG-09      39500       2000 W           41500
M500 PEG          O'HARA       22-FEB-15      46000       2000 SW          48000

rollback complete.
AMID AMFIRST      AMLAST       AMEDATE        AMSAL     AMCOMM REGION     AMEARN
---- ------------ ------------ --------- ---------- ---------- ------ ----------
T500 NICK         TAYLOR       05-SEP-09      42000       3500 W           45500
L500 MANDY        LOPEZ        01-OCT-09      47000       1500             48500
J500 SAMMIE       JONES        01-AUG-09      39500       2000 W           41500
D500 SCOTT        DAVIS        22-FEB-15      53000       6000 W           59000
M500 PEG          O'HARA       22-FEB-15      46000       2000 SW          48000
```

The row for AMID of D500 was removed or deleted

The ROLLBACK was used to undo t he deletion so the row is still in the table

This was verified by using the SELECT

# Deleting Rows

Worksheet | Query Builder

```
1  DELETE FROM acctmanager;
2
3  SELECT *
4  FROM acctmanager;
5
6  ROLLBACK;
7
8  SELECT *
9  FROM acctmanager;
```

Script Output | Query Result

Task completed in 0.193 seconds

```
5 rows deleted.
no rows selected


rollback complete.
AMID AMFIRST      AMLAST       AMEDATE        AMSAL     AMCOMM REGION     AMEARN
---- ------------ ------------ --------- ---------- ---------- ------ ----------
T500 NICK         TAYLOR       05-SEP-09      42000       3500 W           45500
L500 MANDY        LOPEZ        01-OCT-09      47000       1500             48500
J500 SAMMIE       JONES        01-AUG-09      39500       2000 W           41500
D500 SCOTT        DAVIS        22-FEB-15      53000       6000 W           59000
M500 PEG          O'HARA       22-FEB-15      46000       2000 SW          48000
```

The WHERE clause is optional

If a WHERE clause is not used the entire set of data is removed from the table

A ROLLBACK will recover the lost data

Just make sure you do not issue a COMMIT prior to attempting the ROLLBACK, it will not work after the COMMIT

# Quote to Remember

- "*Databases are unavoidable*"
  - David Gallardo
  - JAVAPro Magazine September 2003, Vol. 7 No. 9