Module 3.4 – SQL: Functions, Set Operators, Views, DDL, TCL, DCL

3. SQL: Structured Query Language

- 3.1 Basic SQL Statement
  - Overview
- 3.2 SQL Select Clauses and Aggregate Functions
  - Clauses and aggregate functions
- 3.3 Subqueries
  - WHERE, IN, HAVING, Attribute List (Inline Subquery), Correlated
- 3.4 Functions, Set Operators, Views, DDL, TCL, DCL

DALHOUSIE UNIVERSITY
Inspiring Minds

Module 3.4 – SQL: Functions, Set Operators, Views, DDL, TCL, DCL

Learning Outcomes

- Understand and explain SQL subqueries and functions
- Create and use SQL statements with subqueries and functions
- Create and use stored procedures and triggers

• Textbook Readings

- SQL - Chap 7-8

• Testing*

*Main (but not the only ones) sections of the textbook used for testing are identified in parentheses

- SQL Subqueries and Functions(Chap 7.8 – 7.11)

SET OPERATORS

- SET OPERATORS
  - UNION
  - INTERSECT
  - EXCEPT (MINUS)
  - 3 out of 8 operators covered
    UNION, INTERSECT, DIFFERENCE, PRODUCT, and DIVIDE, SELECT, PROJECT, JOIN

SET OPERATORS - UNION

UNION … Union of tuples in both operands

with duplicate tuples removed (unless ALL is included)

```
SELECT    CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE
FROM      CUSTOMER
UNION
SELECT    CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE
FROM      CUSTOMER_2;
```

- Unique set (duplicate rows are removed
- UNION ALL
  - To KEEP DUPLICATES
- Operands Union Compatible?

Database name: Ch07_SaleCo

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_BALANCE
10010	Ramas	Alfred	A	615	844-2573	0.00
10011	Dunne	Leona	K	713	894-1238	0.00
10012	Smith	Kathy	W	615	894-2285	345.86
10013	Olowski	Paul	F	615	894-2180	536.75
10014	Orlando	Myron		615	222-1672	0.00
10015	O'Brian	Amy	B	713	442-3381	0.00
10016	Brown	James	G	615	297-1228	221.19
10017	Williams	George		615	290-2556	768.93
10018	Farriss	Anne	G	713	382-7185	216.55
10019	Smith	Olette	K	615	297-3809	0.00

Table name: CUSTOMER_2

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
345	Terrell	Justine	H	615	322-9870
347	Olowski	Paul	F	615	894-2180
351	Hernandez	Carlos	J	723	123-7654
352	McDowell	George		723	123-7768
365	Tirpin	Khaleed	G	723	123-9876
368	Lewis	Marie	J	734	332-1789
369	Dunne	Leona	K	713	894-1238

Query name: qryUNION-of-CUSTOMER-and-CUSTOMER_2

CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
Brown	James	G	615	297-1228
Dunne	Leona	K	713	894-1238
Farriss	Anne	G	713	382-7185
Hernandez	Carlos	J	723	123-7654
Lewis	Marie	J	734	332-1789
McDowell	George		723	123-7768
O'Brian	Amy	B	713	442-3381
Olowski	Paul	F	615	894-2180
Orlando	Myron		615	222-1672
Ramas	Alfred	A	615	844-2573
Smith	Kathy	W	615	894-2285
Smith	Olette	K	615	297-3809
Terrell	Justine	H	615	322-9870
Tirpin	Khaleed	G	723	123-9876
Williams	George		615	290-2556

SET OPERATORS - INTERSECT

INTERSECT … Result is intersection of sets (tuples that appear in both operands)

```
SELECT      CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE
FROM        CUSTOMER
INTERSECT
SELECT      CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE
FROM        CUSTOMER_2;
```

INTERSECT …. continued

Example:

Find customer codes for all customers who are in area code 615 and who have made purchases. (If a customer has made a purchase, there must be an invoice record for that customer.)

```
SELECT      CUS_CODE
FROM        CUSTOMER
WHERE       CUS_AREACODE = '615'
INTERSECT
SELECT       DISTINCT CUS_CODE
FROM        INVOICE ;
```

CUS_CODE
10012
10014

INTERSECT …. continued

Example:

Find customer codes for all customers who are in area code 615 and who have made purchases. (If a customer has made a purchase, there must be an invoice record for that customer.)

```
SELECT      CUS_CODE
FROM        CUSTOMER
WHERE       CUS_AREACODE = '615'
INTERSECT
SELECT       DISTINCT CUS_CODE
FROM        INVOICE ;
```

CUS_CODE
10012
10014

EXCEPT (INUS) … [EXCEPT | MINUS]

EXCEPT … returns only the rows that appear in the first set but not in the second operand

Example: Find which customers in the CUSTOMER table are not found in the CUSTOMER_2 table:

```
SELECT    CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE
FROM      CUSTOMER
MINUS
SELECT    CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE
FROM      CUSTOMER_2;
```

CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
Ramas	Alfred	A	615	844-2573
Smith	Kathy	W	615	894-2285
Orlando	Myron		615	222-1672
O'Brian	Amy	B	713	442-3381
Brown	James	G	615	297-1228
Williams	George		615	290-2556
Farriss	Anne	G	713	382-7185
Smith	Olette	K	615	297-3809

SET OPERATORS – EQUIVALENT STATEMENTS

INTERSECT

```
SELECT   CUS_AREACODE FROM CUSTOMER
INTERSECT
 SELECT  V_AREACODE FROM VENDOR;    Area codes for customers with a vendor having the same a-code
```

Equivalent statement (returns equivalent/same result):

```
SELECT   DISTINCT CUS_AREACODE
FROM CUSTOMER JOIN VENDOR ON CUS_AREACODE = V_AREACODE;
```

EXCEPT

```
SELECT   CUS_CODE FROM CUSTOMER WHERE CUS_AREACODE = '615'
EXCEPT
SELECT   DISTINCT CUS_CODE FROM INVOICE;   … customers who do not have purhases
```

Equivalent statement (returns equivalent/same result):

```
SELECT   CUS_CODE FROM CUSTOMER
WHERE   CUS_AREACODE = '615' AND CUS_CODE NOT IN (SELECT DISTINCT CUS_CODE FROM INVOICE);
```

3.3.2 SQL FUNCTIONS

- May be DBMS vendor dependent
- Date and time functions … lots … see section 7.9 of the textbook
- Numeric
  - ABS, CEILING, FLOOR, ROUND
- STRING
  - CONCAT, UPPER, LOWER, SUBSTRING, …
- CONVERSION
  - CAST, CONVERT, …

3.2.3 DDL: Create Table

- CREATE TABLE command

```
CREATE TABLE tablename (
column1         data type       [constraint] [,
column2         data type       [constraint] ] [,
PRIMARY KEY     (column1        [, column2]) ] [,
FOREIGN KEY     (column1        [, column2]) REFERENCES tablename] [,
CONSTRAINT      constraint ] );
```

- SQL constraints
  - FOREIGN KEY
  - NOT NULL
  - UNIQUE
  - DEFAULT
  - CHECK

3.2.3 DDL: Create Table … continued

- Create a table with a SELECT statement
  - Rapidly creates a new table based on selected columns and rows of an existing table using a subquery
  - Automatically copies all of the data rows returned
- SQL indexes
  - CREATE INDEX improves the efficiency of searches and avoids duplicate column values
  - DROP Index deletes an index

Create Table … continued

```
CREATE TABLE PRODUCT (
P_CODE          VARCHAR(10)     NOT NULL        UNIQUE,
P_DESCRIPT      VARCHAR(35)     NOT NULL,
P_INDATE        DATE            NOT NULL,
P_QOH           SMALLINT        NOT NULL,
P_MIN           SMALLINT        NOT NULL,
P_PRICE         NUMBER(8,2)     NOT NULL,
P_DISCOUNT      NUMBER(5,2)     NOT NULL,
V_CODE          INTEGER,
PRIMARY KEY (P_CODE),
FOREIGN KEY (V_CODE) REFERENCES VENDOR ON UPDATE CASCADE);
```

DDL: SQL Constraint

- PRIMARY KEY (V_CODE)
- FOREIGN KEY (V_CODE) REFERENCES VENDOR ON UPDATE CASCADE
  - Preservation of the foreign key constraint using on *update cascade*
  - ON DELETE CASCADE / ON UPDATE CASCADE
- NOT NULL
- UNIQUE
- DEFAULT
- CHECK

```
CREATE TABLE CUSTOMER (
CUS_CODE        NUMBER       PRIMARY KEY,
CUS_LNAME       VARCHAR(15)  NOT NULL,
CUS_FNAME       VARCHAR(15)  NOT NULL,
CUS_INITIAL     CHAR(1),
CUS_AREACODE    CHAR(3)      DEFAULT '615'     NOT NULL
                             CHECK(CUS_AREACODE IN ('615','713','931')),
CUS_PHONE       CHAR(8)      NOT NULL,
CUS_BALANCE     NUMBER(9,2)  DEFAULT 0.00,
CONSTRAINT CUS_UI1 UNIQUE (CUS_LNAME, CUS_FNAME));
```

- When you create the column definition (you can specify a column constraint)
- When you use the CONSTRAINT keyword - table constraint

Create Tables Examples

```
CREATE TABLE CUSTOMER (
CUS_CODE        NUMBER       PRIMARY KEY,
CUS_LNAME       VARCHAR(15)  NOT NULL,
CUS_FNAME       VARCHAR(15)  NOT NULL,
CUS_INITIAL     CHAR(1),
CUS_AREACODE    CHAR(3)      DEFAULT '615'     NOT NULL
                             CHECK(CUS_AREACODE IN ('615','713','931')),
CUS_PHONE       CHAR(8)      NOT NULL,
CUS_BALANCE     NUMBER(9,2)  DEFAULT 0.00,
CONSTRAINT CUS_UI1 UNIQUE (CUS_LNAME, CUS_FNAME));
```

```
CREATE TABLE INVOICE (
INV_NUMBER   NUMBER   PRIMARY KEY,
CUS_CODE     NUMBER   NOT NULL REFERENCES CUSTOMER(CUS_CODE),
INV_DATE     DATE     DEFAULT SYSDATE NOT NULL,
CONSTRAINT INV_CK1 CHECK (INV_DATE > '01-JAN-2018'));
```

Note, CUS_CODE is a foreign key

Examples … continued

```
CREATE TABLE LINE (
INV_NUMBER        NUMBER          NOT NULL,
LINE_NUMBER       NUMBER(2,0)     NOT NULL,
P_CODE            VARCHAR(10)     NOT NULL,
LINE_UNITS        NUMBER(9,2)     DEFAULT 0.00    NOT NULL,
LINE_PRICE        NUMBER(9,2)     DEFAULT 0.00    NOT NULL,
PRIMARY KEY (INV_NUMBER, LINE_NUMBER),
FOREIGN KEY (INV_NUMBER) REFERENCES INVOICE ON DELETE CASCADE,
FOREIGN KEY (P_CODE) REFERENCES PRODUCT(P_CODE),
CONSTRAINT LINE_UI1 UNIQUE(INV_NUMBER, P_CODE));
```

Create Table with Select

```
CREATE TABLE    PART AS
SELECT          P_CODE AS PART_CODE, P_DESCRIPT AS PART_DESCRIPT,
                P_PRICE AS PART_PRICE, V_CODE
FROM PRODUCT;
```

Table structure and data copied, but not constraints
(i.e., no primary key specified and no foreign key specified)

MS ACCESS Version only

```
SELECT      P_CODE AS PART_CODE, P_DESCRIPT AS PART_DESCRIPT,
            P_PRICE AS PART_PRICE    INTO PART
FROM        PRODUCT;
```

SQL Indices

- CREATE [UNIQUE] INDEX indexname ON tablename(column1 [, column2])

```
CREATE INDEX P_INDATEX ON PRODUCT(P_INDATE);

CREATE UNIQUE INDEX P_CODEX ON PRODUCT(P_CODE);

CREATE UNIQUE INDEX EMP_TESTDEX ON TEST(EMP_NUM, TEST_CODE, TEST_DATE);
```

- DROP INDEX indexname

```
DROP INDEX PROD_PRICEX ;
```

Altering table Structures

- BE VERY VERY VERY CAREFUL

- Add / modify columns
  - ALTER TABLE tablename {ADD | MODIFY}
    (columnname datatype [{ADD | MODIFY} columnname datatype]);
- Add / modify constraints
  - ALTER TABLE tablename ADD constraint [ADD constraint]
- Remove column of constraint
  - ALTER TABLE tablename DROP {PRIMARY KEY | COLUMN columnname | CONSTRAINT constraintname };

Changing a Column's Data type

- If the column to be changed already contains data, you can make changes in the column's characteristics if those changes do not alter the data type.
  - ALTER TABLE PRODUCT MODIFY (V_CODE CHAR(5));
  - ALTER TABLE PRODUCT MODIFY (P_PRICE DECIMAL(9,2));
- Adding a column
  - ALTER TABLE PRODUCT ADD (P_SALECODE CHAR(1));

Adding Primary Key, Foreign Key, and Check Constraints

- ALTER TABLE PART
  ADD PRIMARY KEY (PART_CODE);

- ALTER TABLE PART
  ADD FOREIGN KEY (V_CODE) REFERENCES VENDOR;

- ALTER TABLE PART ADD
  CHECK (PART_PRICE >= 0);

- ALTER TABLE PART
  ADD PRIMARY KEY (PART_CODE)
  ADD FOREIGN KEY (V_CODE) REFERENCES VENDOR CHECK
  ADD CHECK (PART_PRICE >= 0);

- ALTER TABLE LINE
  ADD PRIMARY KEY (INV_NUMBER, LINE_NUMBER)
  ADD FOREIGN KEY (INV_NUMBER) REFERENCES INVOICE
  ADD FOREIGN KEY (P_CODE) REFERENCES PRODUCT;

Dropping a Column, Dropping a Table

- Dropping a Column

  ALTER TABLE VENDOR DROP COLUMN V_ORDER;

- Dropping a Table

  DROP TABLE PART;

Inserting Rows

- INSERT INTO tablename VALUES (value1, value2, ..., valuen)
  - INSERT INTO VENDOR VALUES (21225,’Bryson, Inc.','Smithson','615','223-3234','TN','Y');
  - INSERT INTO VENDOR VALUES (21226,’Superloo, Inc.','Flushing','904','215-8995','FL','N’);

- Inserting rows with optional attributes
  - Optional attributes will not be NULL
  - INSERT INTO PRODUCT(P_CODE, P_DESCRIPT) VALUES ('BRT-345','Titanium drill bit');

Inserting table Rows with a SELECT Subquery

```
INSERT INTO     target_tablename[(target_columnlist)]
SELECT          source_columnlist
FROM            source_tablename;
```

Created a table to insert into:

```
CREATE TABLE PART(
        PART_CODE           CHAR(8),
        PART_DESCRIPT       CHAR(35),
        PART_PRICE          DECIMAL(8,2),
        V_CODE              INTEGER,
        PRIMARY KEY         (PART_CODE)
)
```

Inserting table Rows with a SELECT Subquery … continued

```
INSERT INTO PART (PART_CODE, PART_DESCRIPT, PART_PRICE, V_CODE)
SELECT P_CODE, P_DESCRIPT, P_PRICE, V_CODE
FROM PRODUCT;
```

Updating table Rows

```
UPDATE      tablename
SET         columnname = expression [, columnname = expression]
[WHERE      conditionlist] ;
```

```
UPDATE      PRODUCT
SET         P_INDATE = '18-JAN-2018'
WHERE       P_CODE = '13-Q2/P2';
```

```
UPDATE      PRODUCT
SET         P_INDATE = '18-JAN-2018' ,  P_PRICE = 17.9,  P_MIN =-10
WHERE       P_CODE = '13-Q2/P2';
```

Deleting Table Rows

- DELETE FROM tablename
  [WHERE conditionlilst] ;

```
DELETE FROM   PRODUCT
WHERE         P_CODE = 'BRT-345' ;

DELETE FROM   PRODUCT
WHERE         P_MIN = 5 ;
```

MySQL – Safe mode that prevents updates without including the primary key in conditionlist:

```
SET SQL_SAFE_UPDATES = 0;
DELETE FROM PRODUCT WHERE P_MIN = 5;
SET SQL_SAFE_UPDATES = 1;
```

DCL: Commit and Rollback

- COMMIT … Permanently store changes in the DB
- ROLLBACK … Undo changes made since the last commit
- In programs, BEGIN TRASNACTON and END TRANSACTION defines the scope of a transaction. In command line … transaction is one SQL statement (?)

Virtual Tables

CREATE VIEW viewname AS SELECT query;

```
CREATE VIEW     PROD_STATS AS
SELECT          V_CODE, SUM(P_QOH*P_PRICE) AS TOTCOST, MAX(P_QOH) AS MAXQTY,
                MIN(P_QOH) AS MINQTY,     AVG(P_QOH) AS AVGQTY
FROM            PRODUCT
GROUP BY        V_CODE;
```

Updatable View

- A view (virtual table) may consist of selecting from a number of underlying (base) tables.
- A view is updatable if its primary key consists of primary key of the underlying tables (tables on which the view is defined).
- A view may be updatable if it does not satisfy the above condition … if encountered after the course, you need to investigate …

Stored Procedures and Triggers

Stored procedures and Triggers … will be covered as parts of an assignment.

Questions and Answers (Q/A)