SIT103/SIT772 Data and Information Management

Week 7

SQL

Data Definition Language (DDL)

Data Manipulation Language (DML)

Dr Iynkaran Natgunanathan,

email:

iynkaran.natgunanathan@deakin.edu.au,

Phone: +61 3 924 68825.

Last Week

- Relational Algebra
- Joins
- SQL Functions
- **OnTrack Task:** 6.1P SELECT with JOIN
  - `SELECT` queries from multiple tables with `JOIN`

Any Questions?

This week

- More SQL
- Data Definition Language (DDL)
  - `CREATE TABLE`
  - `ALTER TABLE`
  - `DROP TABLE`
- Data Manipulation Language (DML)
  - `INSERT`
  - `UPDATE`
  - `DELETE`
- `COMMIT` and `ROLLBACK`
- Views

Data Definition Language (DDL)

- Set of commands used to create, alter and delete relations (tables)

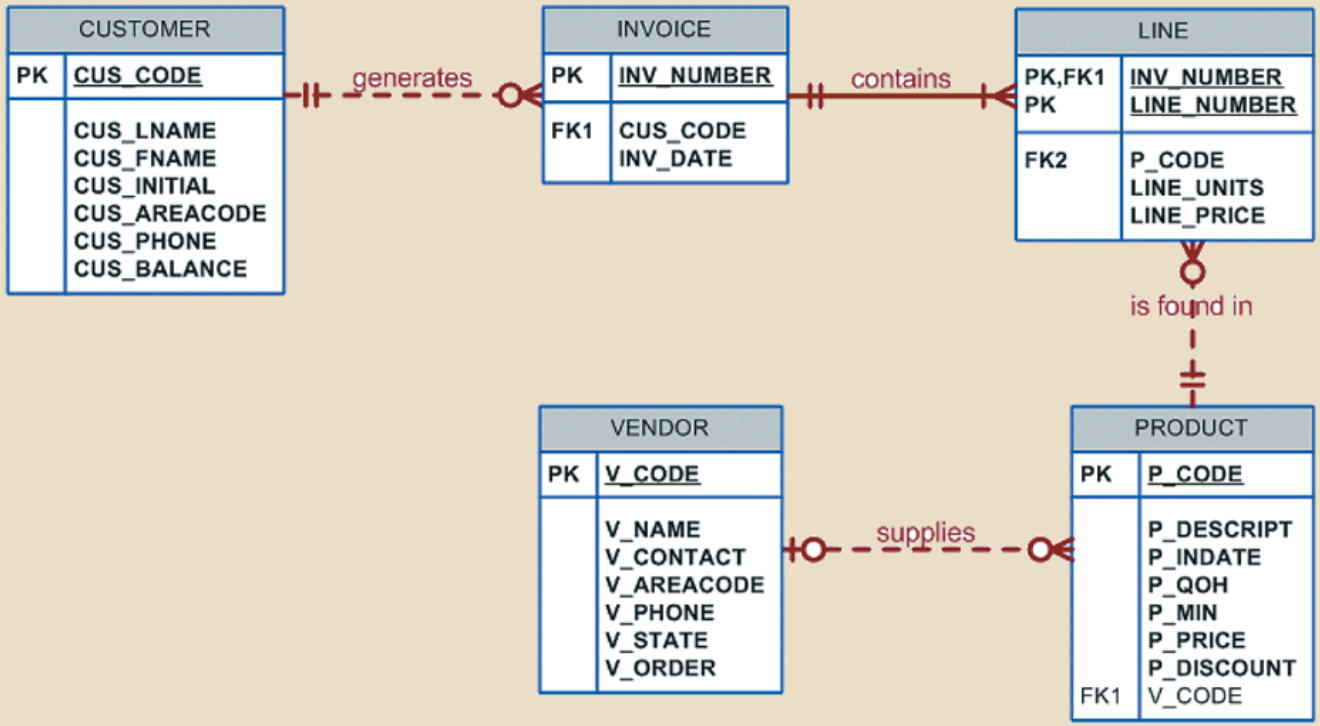
COMMAND OR OPTION	DESCRIPTION
CREATE SCHEMA AUTHORIZATION	**Creates a database schema**
CREATE TABLE	**Creates a new table in the user's database schema**
NOT NULL	Ensures that a column will not have null values
UNIQUE	Ensures that a column will not have duplicate values
PRIMARY KEY	Defines a primary key for a table
FOREIGN KEY	Defines a foreign key for a table
DEFAULT	Defines a default value for a column (when no value is given)
CHECK	Validates data in an attribute
CREATE INDEX	**Creates an index for a table**
CREATE VIEW	**Creates a dynamic subset of rows and columns from one or more tables**
ALTER TABLE	**Modifies a table's definition (adds, modifies, or deletes attributes or constraints)**
CREATE TABLE AS	**Creates a new table based on a query in the user's database schema**
DROP TABLE	**Permanently deletes a table (and its data)**
DROP INDEX	**Permanently deletes an index**
DROP VIEW	**Permanently deletes a view**

TABLE 7.2 from the textbook

Implementation from Logical ERD

- Implementing Entities in the logical ERD as relations/tables
  - Attributes data types and domain
  - Constraints

FIGURE 8.1 DATABASE MODEL



Common SQL Data types

TABLE 8.1

SOME COMMON SQL DATA TYPES

DATA TYPE	FORMAT	COMMENTS
Numeric	NUMBER(L,D) or NUMERIC(L,D)	The declaration NUMBER(7,2) or NUMERIC(7,2) indicates that numbers will be stored with two decimal places and may be up to seven digits long, including the sign and the decimal place (for example, 12.32 or –134.99).
	INTEGER	May be abbreviated as INT. Integers are (whole) counting numbers, so they cannot be used if you want to store numbers that require decimal places.
	SMALLINT	Like INTEGER but limited to integer values up to six digits. If your integer values are relatively small, use SMALLINT instead of INT.
	DECIMAL(L,D)	Like the NUMBER specification, but the storage length is a *minimum* specification. That is, greater lengths are acceptable, but smaller ones are not. DECIMAL(9,2), DECIMAL(9), and DECIMAL are all acceptable.
Character	CHAR(L)	Fixed-length character data for up to 255 characters. If you store strings that are not as long as the CHAR parameter value, the remaining spaces are left unused. Therefore, if you specify CHAR(25), strings such as *Smith* and *Katzenjammer* are each stored as 25 characters. However, a U.S. area code is always three digits long, so CHAR(3) would be appropriate if you wanted to store such codes.
	VARCHAR(L) or VARCHAR2(L)	Variable-length character data. The designation VARCHAR2(25) or VARCHAR(25) will let you store characters up to 25 characters long. However, unlike CHAR, VARCHAR will not leave unused spaces. Oracle automatically converts VARCHAR to VARCHAR2.
Date	DATE	Stores dates in the Julian date format.

CREATE TABLE

- Syntax

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

- Constraints
  - FOREIGN KEY
  - NOT NULL
  - UNIQUE
  - DEFAULT
  - CHECK

Example - VENDOR

table name

data types

```
CREATE TABLE VENDOR (
    V_CODE        INTEGER        NOT NULL,
    V_NAME        VARCHAR(35)    NOT NULL,
    V_CONTACT     VARCHAR(15)    NOT NULL,
    V_AREACODE    CHAR(4)        NOT NULL,
    V_PHONE       CHAR(8)        NOT NULL,
    V_STATE       CHAR(3)        NOT NULL,
    V_ORDER       CHAR(1)        NOT NULL,
    PRIMARY KEY (V_CODE));
```

column names

constraints

Example - PRODUCT

```
CREATE TABLE PRODUCT (
        P_CODE       VARCHAR(10)      NOT NULL,
        P_DESCRIPT   VARCHAR(35)      NOT NULL,
        P_INDATE     DATE             NOT NULL,
        P_ONHAND     INTEGER          NOT NULL,
        P_MIN        INTEGER          NOT NULL,
        P_PRICE      NUMERIC(8,2)
        P_DISCOUNT   NUMERIC(5,2)
        V_CODE       INTEGER,
        PRIMARY KEY (P_CODE),
        FOREIGN KEY (V_CODE) REFERENCES VENDOR (V_CODE));
```

This reference to the VENDOR table means that the VENDOR table must be created before this PRODUCT table.

Foreign Key Constraint

P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-17	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-17	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-17	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-18	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-18	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-17	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-17	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-18	12	5	38.95	0.05	25595
23109-HB	Claw hammer	20-Jan-18	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	02-Jan-18	8	5	14.4	0.05	
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-17	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-18	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-18	188	75	5.87	0.00	
SM-18277	1.25-in. metal screw, 25	01-Mar-18	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	24-Feb-18	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-18	18	5	119.95	0.10	25595

What happens to V_CODE of three products if you want to delete the record of vendor with V_CODE = 25595 from the VENDOR table?

V_CODE	V_NAME	V_CONTACT	V_AREACODE	V_PHONE	V_STATE	V_ORDER
21225	Bryson, Inc.	Smithson	615	223-3234	TN	Y
21226	SuperLoo, Inc.	Flushing	904	215-8995	FL	N
21231	D&E Supply	Singh	615	228-3245	TN	Y
21344	Gomez Bros.	Ortega	615	889-2546	KY	N
22567	Dome Supply	Smith	901	678-1419	GA	N
23119	Randsets Ltd.	Anderson	901	678-3998	GA	Y
24004	Brackman Bros.	Browning	615	228-1410	TN	N
24288	ORDVA, Inc.	Hakford	615	898-1234	TN	Y
25443	B&K, Inc.	Smith	904	227-0093	FL	N
25501	Damal Supplies	Smythe	615	890-3529	TN	N
25595	Rubicon Systems	Orton	904	456-0092	FL	Y

By default most DBMS systems do not allow to delete/update values used as FKs in other tables

Foreign Key Constraint (1)

- We can define what we want to do with the FK values when their corresponding in the primary table are updates or records are deleted (we want to allow delete)
- Use options when defining Foreign Key constraint

`ON UPDATE/DELETE` *action*

Action: CASCADE or SET NULL

CASCADE do the same action in the table with FK
SET NULL sets the FK values to NULL

Example:

```
FOREIGN KEY (V_CODE) REFERENCES VENDOR (V_CODE) ON DELETE SET NULL
            ON UPDATE CASCADE
```

NOTE: Oracle does not support `ON UPDATE CASCADE`

(Oracle's philosophy is PK should never change)

Other Constraints

- NOT NULL: Ensures that a column does not accept NULL
- UNIQUE: Ensures all values in a column are unique
- DEFAULT: Assigns default values when a column value is not provided
- CHECK: Ensures column values satisfy given condition

```
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      NUMERIC(9,2)  DEFAULT 0.00,
 CONSTRAINT CUS_UI1 UNIQUE(CUS_LNAME, CUS_FNAME));
```

Another way of defining PK

Name/ID of the constraint, does not real really mean anything, useful to refer to it later, e.g., to remove the constraint after the table is created

Constraints - more examples

```
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'));
```

Another way of defining FK

Column name is not specified, uses the column of the same name

```
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));
```

Deletes the record if corresponding record in invoice is deleted (useful in weak entity, e.g., delete assignments if unit it associated with is deleted)

Nothing is mentioned, so default rule (Delete is restricted)

Create table with a SELECT SQL

- Create a new table SELECTing columns and rows from an existing table
- Using sub-query (nested query) – SQL embedded within another SQL

  Note that Inner/embedded query is always executed first

- Copies attribute names, data characteristics and rows from the original table

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

- Note that constraints such as Entity Integrity (PK) and Referential Integrity (FK) are not applied automatically. Constraints must be added after the table is created.

  But how to add constraints? `ALTER TABLE`

Altering Table Structure

- All changes in existing table structure are made by using
  - `ALTER TABLE` command followed by a keyword that produces the specific change you want to make: `ADD`, `MODIFY`, and `DROP`
- Changing a column's data characteristics
  - 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
- Adding a column
  - Altering an existing table by adding one or more columns
- Adding a constraint
  - Altering an existing table to add a constraint

ALTER TABLE Syntax

- ADD/MODIFY columns

```
ALTER TABLE tablename {ADD|MODIFY} (columnname datatype
[{ADD | MODIFY} columnname datatype]);
```

- ADD constraint

```
ALTER TABLE tablename ADD constraint [ADD constraint];
```

- Remove column/constraint

```
ALTER TABLE tablename DROP {PRIMARY KEY|COLUMN
columnname|CONSTRAINT constraintname};
```

To remove a constraint, we need to refer it to by its name, which is one reason why we should name constraints while adding them (we saw earlier with create table).

ALTER TABLE Examples

- **Adding a column**

```
ALTER TABLE PRODUCT ADD (P_SALECODE CHAR(1));
```

- **Changing a column's data type**

```
ALTER TABLE PRODUCT MODIFY (V_CODE CHAR(5));
```

- **Changing a column's data characteristics**

```
ALTER TABLE PRODUCT MODIFY (P_PRICE DECIMAL(9,2));
```

- **Adding constraints**

```
ALTER TABLE PART ADD PRIMARY KEY (PART_CODE);
ALTER TABLE PART ADD FOREIGN KEY (V_CODE) REFERENCES VENDOR;
```

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

- **Dropping column/constraint**

```
ALTER TABLE VENDOR DROP COLUMN V_ORDER;
```

Some DBMSs impose restrictions on attribute deletion. You may not drop attributes that are involved in foreign key relationships, nor may you delete an attribute that has important data.

Other useful SQL Commands

- **Delete a table from the database**

  Syntax: `DROP TABLE` *tablename*;

  Example: `DROP TABLE CUSTOMER;`

- **Delete all records from a table but retain the table structure**

  Syntax: `TRUNCATE TABLE` *tablename*;

  Example: `TRUNCATE TABLE CUSTOMER;`

- **Describe the structure (column names and data types) of a table**

  Syntax: `DESCRIBE TABLE` *tablename*; OR `DESC TABLE` *tablename*;

  Example: `DESC TABLE CUSTOMER;`

- **Rename a table**

  Syntax: `RENAME TABLE` *tablename* `TO` *newtablename*; OR `ALTER TABLE` *tablename* `RENAME TO` *newtablename*;

  Example: `RENAME EMP TO EMPLOYEE;`

- **Rename a column of a table**

  Syntax: `ALTER TABLE` *tablename* `RENAME COLUMN` *columnname* `TO` *newcolumnname*;

  Example: `ALTER TABLE EMP RENAME COLUMN EMP_MGR TO EMP_MGR_CODE;`

Data Manipulation Language (DML)

- Interacting with a database
- We covered SELECT in detail in Week 4 – retrieve records
- Today, we cover three commands to change records:
  - INSERT
  - UPDATE
  - DELETE

COMMAND, OPTION, OR OPERATOR	DESCRIPTION
SELECT	**Selects attributes from rows in one or more tables or views**
FROM	Specifies the tables from which data should be retrieved
WHERE	Restricts the selection of rows based on a conditional expression
GROUP BY	Groups the selected rows based on one or more attributes
HAVING	Restricts the selection of grouped rows based on a condition
ORDER BY	Orders the selected rows based on one or more attributes
INSERT	**Inserts row(s) into a table**
UPDATE	**Modifies an attribute's values in one or more table's rows**
DELETE	**Deletes one or more rows from a table**
Comparison operators	
=, <, >, <=, >=, <>, !=	Used in conditional expressions
Logical operators	
AND/OR/NOT	Used in conditional expressions
Special operators	**Used in conditional expressions**
BETWEEN	Checks whether an attribute value is within a range
IN	Checks whether an attribute value matches any value within a value list
LIKE	Checks whether an attribute value matches a given string pattern
IS NULL	Checks whether an attribute value is null
EXISTS	Checks whether a subquery returns any rows
DISTINCT	Limits values to unique values
Aggregate functions	**Used with SELECT to return mathematical summaries on columns**
COUNT	Returns the number of rows with non-null values for a given column
MIN	Returns the minimum attribute value found in a given column
MAX	Returns the maximum attribute value found in a given column
SUM	Returns the sum of all values for a given column
AVG	Returns the average of all values for a given column

Adding rows in a table - INSERT

Column values in right order

- INSERT command is used to enter data into a table

Syntax: `INSERT INTO` *tablename* `VALUES` *(value1, value2,…, valuen)*;

Example: `INSERT INTO PRODUCT VALUES ('13-Q2/P2', '7.25-in. pwr. saw blade','13-Dec-17',32,15,14.99, 0.05, 21344);`

Column names to set values

- If some of the columns are optional and not provided

Syntax: `INSERT INTO` *tablename (columnlist)* `VALUES` *(listofvalues)*;

Example: `INSERT INTO PRODUCT (P_CODE, P_DESCRIPT) VALUES ('BRT-345','Titanium drill bit');`

Selected columns' values in the same order

- Important points:
  - Column values are enclosed within a parentheses and separated by a comma (,).
  - String/character and date values are enclosed in apostrophes and numbers without
  - NULL can be used for values not available or optional
  - FK value must exist in the primary table before inserting it in the FK table

INSERT Using SELECT subquery

- Data can be retrieved from one table can be inserted into another table using SELECT subquery

Syntax: INSERT INTO *target_tablename* [(*target_columnlist*)]
SELECT *source_columnlist* FROM *source_tablename*;

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

- Number and datatype of values returned the SELECT query must match the number and datatypes of columns in the INSERT part.

Updating Rows - UPDATE

Column values to update

- UPDATE command is used to modify data in a table

Syntax: UPDATE *tablename* SET *columnname* = *value1* [, *columnname* = *value2*]
[WHERE *conditionlist*];

Which rows to update

Example: UPDATE PRODUCT SET P_INDATE = '18-JAN-2018',
P_PRICE = 17.99, P_MIN = 10 WHERE P_CODE = '13-Q2/P2';

- If WHERE clause is not provided, values are updated for all rows
- Arithmetic operators are useful

Example: UPDATE P RODUCT SET P_PRICE = P_PRICE * 1.10
WHERE P_PRICE < 50.00;

- adds 10 percent to the price for all product whose current prices are below $50.00

UPDATE: Exercise for you

P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-17	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-17	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-17	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-18	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-18	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-17	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-17	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-18	12	5	38.95	0.05	25595
23109-HB	Claw hammer	20-Jan-18	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	02-Jan-18	8	5	14.4	0.05	
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-17	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-18	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-18	188	75	5.87	0.00	
SM-18277	1.25-in. metal screw, 25	01-Mar-18	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	24-Feb-18	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-18	18	5	119.95	0.10	25595

Increase the P_PRICE of all products supplied by vendor "Rubicon Systems" by10%

		CONTACT	V_AREACODE	V_PHONE	V_STATE	V_ORDER
		thson	615	223-3234	TN	Y
		hing	904	215-8995	FL	N
21231	D&E Supply	Singh	615	228-3245	TN	Y
21344	Gomez Bros.	Ortega	615	889-2546	KY	N
22567	Dome Supply	Smith	901	678-1419	GA	N
23119	Randsets Ltd.	Anderson	901	678-3998	GA	Y
24004	Brackman Bros.	Browning	615	228-1410	TN	N
24288	ORDVA, Inc.	Hakford	615	898-1234	TN	Y
25443	B&K, Inc.	Smith	904	227-0093	FL	N
25501	Damal Supplies	Smythe	615	890-3529	TN	N
25595	Rubicon Systems	Orton	904	456-0092	FL	Y

```
UPDATE PRODUCT SET P_PRICE = P_PRICE * 1.1 WHERE V_CODE =
(SELECT V_CODE FROM VENDOR WHERE V_NAME = 'Rubicon Systems');
```

Deleting records from a table

- DELETE FROM command allows to delete rows from a table

Which rows to delete

Syntax: `DELETE FROM` *tablename* [`WHERE` *conditionlist*];

Example: `DELETE FROM` PRODUCT `WHER` E P_CODE = 'BRT-345';

- If WHERE clause is not provided, all rows are deleted
  - equivalent to `TRUNCATE TABLE` *tablename*;

Transaction Control Commands

- **COMMIT Command**
  - Changes made to the table contents are not saved on the disk (physical storage) until the changes are committed. Changes are saved when:
    1. Database is closed (DB connection session is closed)
    2. DBMS Program is closed
    3. `COMMIT` command is used

  Syntax: `COMMIT;`

MySQL by defaults has autocommit ON
- Try: `SELECT @@autocommit;`
  `SET autocommit = 0;`
- Always commit after SQL command

- **ROLLBACK Command**
  - Allows to discard the changes made to the table contents since the last COMMIT.

  Syntax: `ROLLBACK;`

- COMMIT and ROLLBACK only work with DML commands that are used to INSERT, UPDATE, or DELETE table rows
  - changes made using DDL commands (CREATE, ALTER, DROP) are not reverted
  - ORACLE automatically COMMIT changes with a DDL command is issued.

COMMIT and ROLLBACK

- **Check your understanding**
- Assume the following sequence of actions in the current ORACLE session:
  1. CREATE a table called SALES.
  2. INSERT 10 rows in the SALES table.
  3. UPDATE two rows in the SALES table.
  4. Execute the ROLLBACK command.

Will the SALES table still exist in the database?

If yes, what will be its status? How many records?

ORACLE does auto COMMIT with DDL command to CREATE SALES table.
ROLLBACK does not revert the DDL command changes
SALES table still exists, but it will be empty as all DML changes (rows inserted and updated) are reverted

Virtual Tables: Creating a View

- A view is a **virtual table** based on a SELECT query
  - Can contain columns, computed columns, aliases, and aggregate functions from **one or more tables**
- **Base tables** are tables on which a view is based (tables used in the SELECT query)

Syntax: `CREATE VIEW` *viewname* `AS SELECT` *query*

Example:
```
CREATE VIEW PRICEGT50 AS SELECT P_DESCRIPT,
P_QOH, P_PRICE FROM PRODUCT WHERE P_PRICE > 50.00;
```

View: PRICEGT50
Base Table: PRODUCT

View example

```
SELECT * FROM PRICEGT50;
```

P_DESCRIPT	P_ONHAND	P_PRICE
Power painter, 15 psi., 3-nozzle	8	109.99
B&D jigsaw, 12-in. blade	8	109.92
B&D jigsaw, 8-in. blade	6	99.87
Hicut chain saw, 16 in.	11	256.99
Steel matting, 4'x8'x1/6", .5" mesh	18	119.95

```
SELECT * FROM PRODUCT;
```

P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-17	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-17	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-17	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-18	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-18	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-17	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-17	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-18	12	5	38.95	0.05	25595
23109-HB	Claw hammer	20-Jan-18	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	02-Jan-18	8	5	14.4	0.05	
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-17	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-18	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-18	188	75	5.87	0.00	
SM-18277	1.25-in. metal screw, 25	01-Mar-18	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	24-Feb-18	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-18	18	5	119.95	0.10	25595

Characteristics of views

- View can be used as a **table** in SQL statements
- Views can be used to maintain users access control, provides **data privacy and security**
  - can restrict users to seeing only specified columns and rows of tables
  - limit the degree of exposure of the underlying tables to the outer world
- Views take very **little space to store**; the database contains only the definition of a view, not a copy of all the data that it presents.
- Views are **dynamically updated**, i.e., re-created on demand each time it is invoked.
- Views may also be used as the **basis for reports** as they can join and aggregate information from multiple tables
  - Instead of executing complex SQL every time a particular report (e.g., weekly sales) is needed, a view can be defined with the required data
- Views can partition a table to **reduce its complexity**

Updatable Views

- View can be used to update base tables that are used in the view – **Updatable Views**
- **Not all views are updatable**
- Views are not updatable if the view defining SQL has Aggregate functions (e.g., AVG, COUNT, etc.), DISTINCT operator, GROUP BY clause, Set operators e.g., UNION, INTERSECTION, etc.) – Most restrictions are due to GROUP or JOIN operators
- Allows UPDATE the base tables using condition requiring JOIN (ORACLE does not support JOIN with the UPDATE command to update rows in a base table directly)
- Allows batch update of a base table based on data in another table

Updatable Views

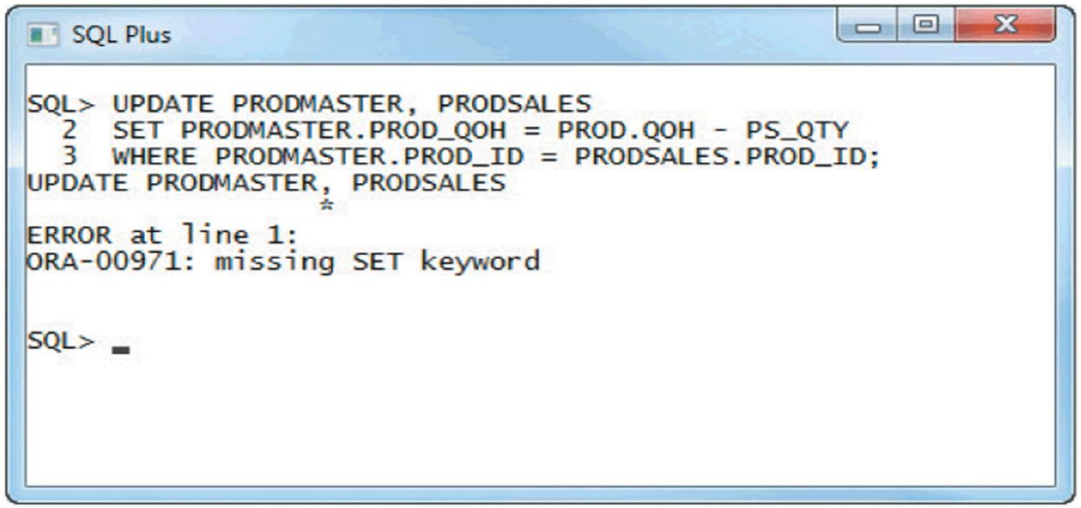
UPDATE PRODMASTER, PRODSALES SET PRODMASTER.PROD_QOH = PROD_QOH – PRODSALES.PS_QTY WHERE PRODMASTER.PROD_ID = PRODSALES.PROD_ID;

JOIN as update condition

Note that the above update statement reflects the following sequence of events:

1. Join the PRODMASTER and PRODSALES tables.
2. Update the PROD_QOH attribute (using the PS_QTY value in the PRODSALES table) for each row of the PRODMASTER table with matching PROD_ID values in the PRODSALES table.

The above query works in MySQL, but Oracle returns an error



Updatable Views

- An updatable view named PSVUPD

SQL Plus

```
SQL> CREATE VIEW PSVUPD AS (
  2      SELECT PRODMASTER.PROD_ID, PROD_QOH, PS_QTY
  3      FROM PRODMASTER, PRODSALES
  4      WHERE PRODMASTER.PROD_ID = PRODSALES.PROD_ID);

View created.

SQL> SELECT * FROM PSVUPD;

PROD_ID    PROD_QOH     PS_QTY
------- ---------- ----------
A123            67          7
BX34            37          3

SQL> _
```

Updatable Views

SQL Plus

```
SQL> SELECT * FROM PRODMASTER;

PROD_ID PROD_DESC          PROD_QOH
------- --------------- ----------
A123    SCREWS                  67
BX34    NUTS                    37
C583    BOLTS                   50

SQL> SELECT * FROM PRODSALES;

PROD_ID      PS_QTY
------- ----------
A123              7
BX34              3

SQL> UPDATE PSVUPD
  2  SET PROD_QOH = PROD_QOH - PS_QTY;

2 rows updated.

SQL> SELECT * FROM PRODMASTER;

PROD_ID PROD_DESC          PROD_QOH
------- --------------- ----------
A123    SCREWS                  60
BX34    NUTS                    34
C583    BOLTS                   50

SQL> _
```

- One easy way to determine whether a view can be used to update a base table
  - If the (primary key) columns of the base table you want to update have unique values in the view, the base table is updatable.
  - For example, if the PROD_ID column of the view returns the A123 or BX34 values more than once, The PRODMASTER table cannot be updated through the view.

PSVUPD View can be used to batch update PRODMASTER based on PRODSALES

Summary

- CREATE TABLE
- ALTER TABLE
- DROP TABLE
- INSERT
- UPDATE
- DELETE
- COMMIT
- ROLLBACK
- Views

This Week's OnTrack Task

- 7.1P DML and DDL Commands

  - SQL with Other DML commands and DDL commands

One small mistake:

- Task 1 asked you to add NOT NULL constraint to EMP_INITIAL column
- Task 4 asked you to insert a record with a NULL value, please insert the actual initial
- This is a mistake in the task sheet, thanks to Rany and Martin for picking this mistake (see "Questions for the Unit Chair" discussion forum).

- 7.2C Miniproject 2 - Part 2: Database Implementation

  - Implementing Normalised Database Designed in Task 4.2C

Next Week

- Procedural Language SQL (PL/SQL)

Thank you

See you next week

Any questions/comments?

SQL Demo

Let's see some SQL demo

Readings and References:

- Chapter 8

Database Systems : Design, Implementation, & Management 13TH EDITION, by Carlos Coronel, Steven Morris