SQL - Revision

Assoc. Prof. David Taniar

SQL - Revision

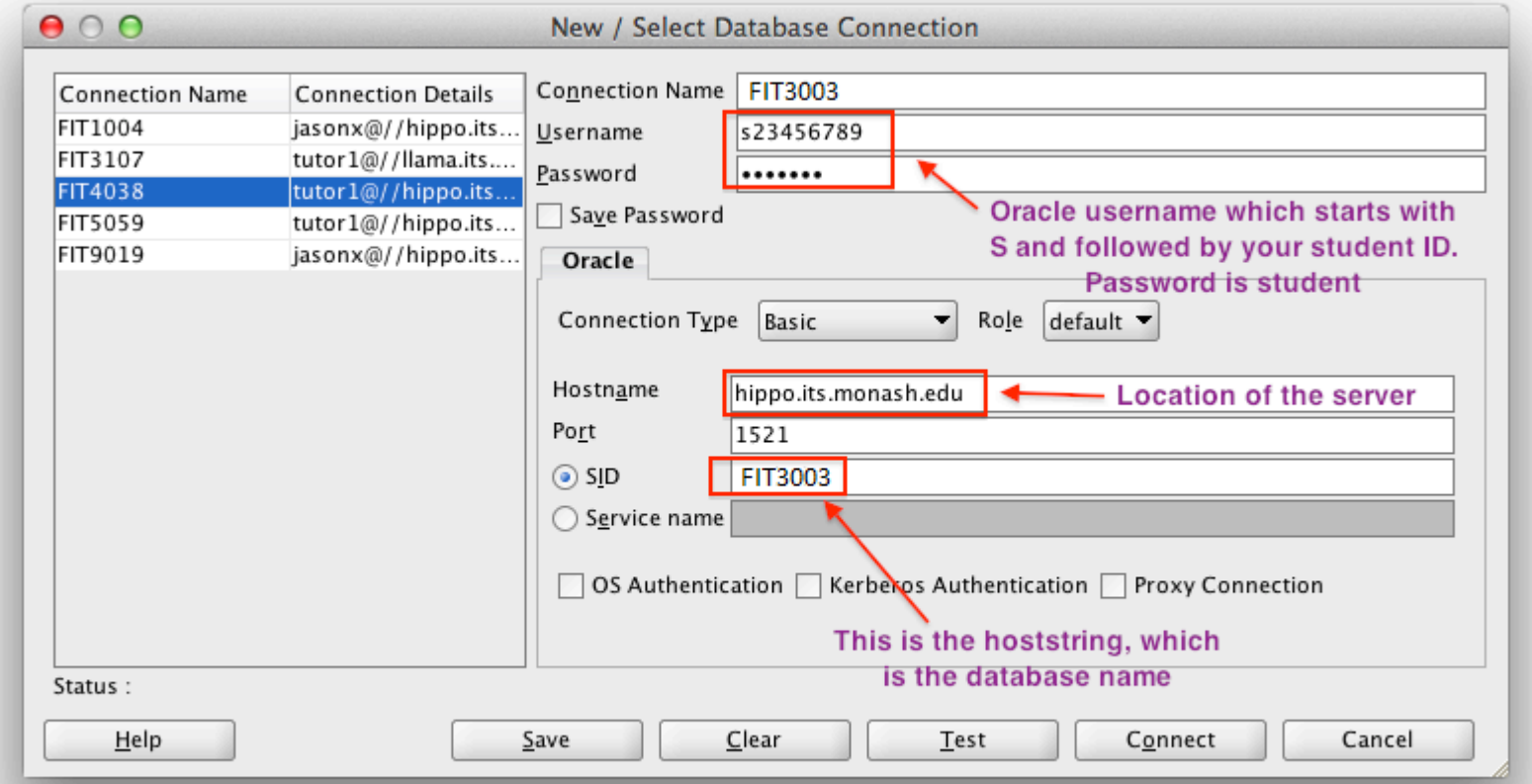
Outline:

A. Introduction to SQL client environment
B. Create tables
C. Insert into
D. Simple query retrieval
E. Updating and deleting records
F. Commit
G. Joining multiple tables
H. Aggregate functions and group by
I. Alter tables

A. Introduction to SQL client

- **There are several SQL client software:**
  - SQL Developer
  - SQL*Plus (Windows, Mac, Unix)
- **Login details:**
  - Username: S12345678 (12345678 is your student id)
  - Password: student
  - Host string: the unit code

Introduction to SQL Developer



B. Create Tables

- General Syntax:

```
CREATE TABLE <table_name>
    (attribute1        data_type    <NOT NULL>,
     attribute2        data_type,
     …
    PRIMARY KEY (attribute1),
    FOREIGN KEY (attribute1)
            REFERENCES <references table_name> (key_attribute));
```

Create Tables – Data Types

- Data type denotes a kind of data of an attribute value
  - Character data types: VARCHAR2 and CHAR

    student_name VARCHAR2(30)

    s_gender CHAR(1)
  - Number data types: NUMBER

    s_age NUMBER(2)

    item_price NUMBER(5,2)
  - Date data type: DATE

Create Tables – Data Types

- Date Data Type
  - DATE stores dates from 1/1/4712 BC to 12/31/4712 AD
  - Default date format: DD-MON-YYYY
    - example: 05-JUN-2001
  - Example declaration: s_dob DATE
  - DATE data type also stores time values

Create Tables – Data Types

- Default **time** format: HH:MI:SS A.M.
  - If no time value is given when a date is inserted, default value is 12:00:00 A.M.
  - If no date value is given when a time is inserted, default date is first day of current month
  - Example s_dob field: 07-OCT-1967 12:00:00 A.M.

Create Tables – Constraints

- Integrity Constraints
  - Primary Key attribute
  - NOT NULL constraints
    - Specifies that a field cannot be NULL
    - Sample Declaration: *Field_name data_type NOT NULL*
  - Foreign Key attribute in a table refers to another record in another table

Create Tables – Example 1

- **Example:** *create DOCTOR table, PATIENT_DETAILS table and PATIENT table*
  - The PATIENT table has foreign key references to the DOCTOR table and PATIENT_DETAILS table.
  - Commands:
    - DOCTOR table

```
CREATE TABLE Doctor
  (DName VARCHAR2(20) NOT NULL,
  DPhone NUMBER(10),
  DSpecialty VARCHAR2(25),
  DRoom VARCHAR2(10),
  PRIMARY KEY (DName));
```

Create Tables – Example 2

- PATIENT_DETAILS table

```
CREATE TABLE Patient_Details
    (PTitle VARCHAR2(5),
    PName VARCHAR2(20) NOT NULL,
    PAdd VARCHAR2(30),
    PDob DATE,
    PGender VARCHAR2(10),
    PStatus VARCHAR2(10),
    POccupation VARCHAR2(25),
    PInsurance CHAR(1),
    PRIMARY KEY(PName));
```

Create Tables – Example 3

- PATIENT table

CREATE TABLE *Patient*

(DName VARCHAR2(20) ***NOT NULL****,*

PName VARCHAR2(20) ***NOT NULL****,*

VisitDate DATE ***NOT NULL****,*

Procedure VARCHAR2(35),

PRIMARY KEY*(DName, PName, VisitDate),*

FOREIGN KEY *(DName)* ***REFERENCES*** *doctor(DName),*

FOREIGN KEY *(PName)* ***REFERENCES*** *patient_details(PName));*

Create Tables – by Copying

- You can create a new table by copying from an existing table:

```
CREATE TABLE <table_name>
    AS SELECT *
        FROM …    ;
```

- For example:

```
CREATE TABLE student
    AS SELECT *
        FROM dtaniar.student;
```

- Notes:
  - It creates and copies the records from the existing table
  - However, it does not copy the PK and FK
  - In the above example, it copies the table from dtaniar account.

Create Tables – View Tables

- **Viewing Information about Tables**
  - To view all tables in the database, the general syntax is:

    SELECT * FROM TAB;

    - For example:

      SQL> select * from tab;

```
TNAME                          TABTYPE  CLUSTERID
------------------------------ -------  ----------
DOCTOR                         TABLE
PATIENT_DETAILS                TABLE
PATIENT                        TABLE
```

Create Tables – Describe Tables

- To view the table structure, the general syntax is:

  DESCRIBE <table_name>

  - For example:

    SQL> desc doctor;

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------
DNAME                                     NOT NULL VARCHAR2(20)
DPHONE                                             NUMBER(10)
DSPECIALTY                                         VARCHAR2(25)
DROOM                                              VARCHAR2(10)
```

Create Tables – Drop Tables

- To drop an unwanted table, the general syntax is:

  DROP TABLE <table_name>;

  - For example:

    SQL> drop table patient_details;

    - If the table that you want to delete (e.g. table Doctor) is being used as a FK by another table (e.g. table Patient), then you cannot delete table Doctor.
    - In this case, you need to delete table Patient first, before deleting table Doctor.

C. Insert Into

- **General Syntax**
  - To insert values for every single attribute in a record:

    INSERT INTO <table_name>

    VALUES (attribute1_value, attribute2_value,….);

    - **Example:**

      INSERT INTO *DOCTOR*

      VALUES*('Allan', 21111, 'General Practice', 'A110');*

  - Strings are enclosed in single quotes (‘) and are case-sensitive (e.g. ‘General Practice’ is different from ‘general practice’)

Insert Into

- To insert a value of selected attributes:

  INSERT INTO <table_name> (attribute1,attribute2,…)

  VALUES (attribute1_value, attribute2_value,….)

  - **Example:**

    INSERT INTO doctor (*DName, DPhone****)***

    VALUES (*'Charles', 11437****);***

Insert Into

- The TO_DATE function:
  - **TO_DATE** (‘date_value’, ’format mask’);
  - **Example:**

INSERT INTO *PATIENT_DETAILS*

VALUES*(‘Mr’,’Albert’,’12 Burke Road’,*

TO_DATE*(‘04/Feb/1967’, ’DD/MON/YYYY‘), 'M', 'Married', 'Consulting', 'N');*

Insert Into

- The common DATE format masks

Format Mask	Formatted Data
DD-MON-YYYY	05-FEB-2007
MM/DD/YYYY	02/05/2007
HH:MI AM	02:30 PM
MONTH DAY, YYYY	FEB 5, 2007
MM/DD/YYYY HH:MI AM	02/05/2007 02:30 PM

Sample DATE format masks

Insert Into

- Insert multiple records **one-by-one**:

```
INSERT INTO DOCTOR VALUES ('Allan', 21111, 'General Practice', 'A110');
INSERT INTO DOCTOR VALUES ('Ally', 23214, 'Neurology', 'B301');
INSERT INTO DOCTOR VALUES ('Ben', 21162, 'Ophthalmology', 'C115');
```

- Insert multiple records at **once**:

```
INSERT ALL
INTO DOCTOR VALUES ('Allan', 21111, 'General Practice', 'A110')
INTO DOCTOR VALUES ('Ally', 23214, 'Neurology', 'B301')
INTO DOCTOR VALUES ('Ben', 21162, 'Ophthalmology', 'C115')
SELECT * FROM DUAL;
```

D. Simple Query Retrieval

- **Simple Retrieval**
  - **Retrieve all Records**
    - **General Syntax :**

      SELECT *

      FROM <table_name>;

- Output:

```
DNAME          DPHONE DSPECIALTY       DROOM
-------------- ------ ---------------- -------------
Charles        11437
Allan          21111 General Practice  A110
Ally           23214 Neurology         B301
Ben            21162 Ophthalmology     C115
Kate           21907 Ophthalmology     C125
Larry          32234 Paediatric        B213
Leonard        20987 General Practice  A111
Menson         27242 General Practice  A108
Precilla       25551 Cardiology        B551
Rex            24113 Psychiatry        B424
Benny          12345 Radiologist       NULL
11 rows selected.
```

- **Example:** retrieve everything from the DOCTOR table:

  *SQL> select * from doctor;*

Simple Query Retrieval

- **Retrieve Specific Fields**
  - **General Syntax:**

    SELECT <attribute1, attribute2…>

    FROM <table_name>;

  - **Example:** *select only the DName from the DOCTOR table*

    SQL> select DName from doctor;

- Output:

```
DNAME
--------------------
Allan
Ally
Ben
Benny
Charles
Kate
Larry
Leonard
Menson
Precilla
Rex
11 rows selected.
```

Simple Query Retrieval

- **Eliminating Duplicated Records *(DISTINCT qualifier)***
  - **General Syntax:**

    SELECT DISTINCT <attribute1,attribute2,…>

    FROM <table_name>;

  - **Example:** eliminating duplicates for the DSpecialty values:

    SQL> select distinct (DSpecialty) from doctor;

Simple Query Retrieval

- Original Output (without DISTINCT):

```
DSPECIALTY
----------------

General Practice
Neurology
Ophthalmology
Ophthalmology
Paediatric
General Practice
General Practice
Cardiology
Psychiatry
Radiologist
```

- Output (with DISTINCT):

```
DSPECIALTY

-------------------------

Cardiology
General Practice
Neurology
Ophthalmology
Paediatric
Psychiatry
Radiologist
7 rows selected.
```

Simple Query Retrieval

- **Conditional Retrieval**
  - Search Conditions specified for more complex data retrieval
  - **The WHERE Clause**
    - Operators:
      - equal (=)
      - greater than (>)
      - less than (<)
      - greater than or equal to (>=)
      - less than or equal to (<=)
      - Not equal (<>)

Simple Query Retrieval

- **General Syntax:**

  SELECT <attribute1,attribute2,…>

  FROM <ownername.table_name1>

  WHERE <search condition>;

- **Example:**

  SQL> select dname, dspecialty from doctor

  where dspecialty = 'General Practice';

```
DNAME                DSPECIALTY
-------------------- -------------------------
Allan                General Practice
Leonard              General Practice
Menson               General Practice
```

Simple Query Retrieval

- **“AND” or “OR”**
  - AND: both conditions must be true
    - **Example:**

```
SQL> select dname, dspecialty from doctor
2  where dname = 'Ben'
3  AND dspecialty = 'General Practice';
no rows selected
```

Simple Query Retrieval

- OR: either one of the condition is true
  - **Example:**

```
SQL> select dname, dspecialty from doctor
2  where dname = 'Ben'
3  OR dspecialty = 'General Practice';
```

```
DNAME                DSPECIALTY
-------------------- -------------------------
Allan                General Practice
Ben                  Ophthalmology
Leonard              General Practice
Menson               General Practice
```

Simple Query Retrieval

- **Other Conditions**
  - **LIKE/NOT LIKE**
    - **Example:** displaying all DName that has their name first character as ‘B’

```
SQL> SELECT dname FROM doctor
 2  WHERE dname LIKE 'B%';

DNAME
--------------------
Ben
Benny
```

Simple Query Retrieval

- **IN/NOT**
  - suitable to perform a set member search
  - **Example:**
    - IN - *displaying all DName that has their specialty either 'General Practice' or 'Cardiology' category*

```
SQL> SELECT dname, dspecialty FROM doctor
 2  WHERE dspecialty IN ('General Practice','Cardiology');

DNAME          DSPECIALTY
-------------- -------------------------
Allan          General Practice
Leonard        General Practice
Menson         General Practice
Precilla       Cardiology
```

Simple Query Retrieval

- NOT (return all records that do not match the search condition) - *retrieving the DName and DSpecialty where their specialty is not 'General Practice'*

```
SQL> SELECT dname, dspecialty FROM doctor
  2  WHERE NOT (dspecialty='General Practice');

DNAME                DSPECIALTY
-------------------- -------------------------
Ally                 Neurology
Ben                  Ophthalmology
Kate                 Ophthalmology
Larry                Paediatric
Precilla             Cardiology
Rex                  Psychiatry
Benny                Radiologist
7 rows selected.
```

Simple Query Retrieval

- **NULL/NOT NULL**
  - **Example:**
    - NULL Operator:

```
SQL> SELECT dname FROM doctor
 2  WHERE dspecialty IS NULL;


DNAME
--------------------
Charles
```

Simple Query Retrieval

- NOT NULL Operator:

SQL> SELECT dname FROM doctor
2 WHERE dspecialty IS NOT NULL;

```
DNAME
--------------------
Allan
Ally
Ben
Kate
Larry
Leonard
Menson
Precilla
Rex
Benny
10 rows selected.
```

Simple Query Retrieval

- **Sorting**
  - specify to sort the output by using ORDER BY
  - **General Syntax:**

```
SELECT <attribute1,attribtue2,..>
FROM <table_name>
ORDER BY <attribute_name> [DESC];
```

Simple Query Retrieval

- **Example:** *retrieving all DName and DSpecialty in a descending order of the DName*

```
SQL> SELECT dname, dspecialty FROM doctor
  2  ORDER BY dname DESC;

DNAME                DSPECIALTY
-------------------- -------------------------
Rex                  Psychiatry
Precilla             Cardiology
Menson               General Practice
Leonard              General Practice
Larry                Paediatric
Kate                 Ophthalmology
Charles
Benny                Radiologist
Ben                  Ophthalmology
Ally                 Neurology
Allan                General Practice
11 rows selected.
```

E. Updating and Deleting Records

- In the created tables
  - **UPDATE** command – updating
  - **DELETE** command – deletion

Updating Records

- **Update**
  - **General Syntax:**

```
UPDATE <table_name>
SET <attribute_name> = <new_value>
WHERE <expression> <operator> <expression>;
```

    - records can be updated in only one table at a time
    - update multiple fields that are within the same table
    - WHERE clause - make the command updates specific records only

Updating Records

- **Example:**
  - *updating DSpecialty of Dname 'Jerry' from 'Radiologist' to 'Neurology'*

```
UPDATE doctor
SET dspecialty = 'Neurology'
WHERE dname = 'Jerry';
```

Deleting Records

- **Delete**
  - **General Syntax:**

    DELETE FROM <table_name>

    WHERE <search_condition>;

    - remove specific records from a database table
    - use WHERE clause to specify multiple records to delete multiple records at one time
    - If the search condition is omitted, all records in the table are deleted.

Deleting Records

- **Example:**
  - deleting a single record from the DOCTOR table

    DELETE FROM *doctor*

    WHERE *DName= 'Benny';*
  - deleting multiple records from the DOCTOR table that contain DName starting with 'A'

    DELETE FROM *doctor*

    WHERE *DName LIKE 'A%';*
  - deleting all records from the DOCTOR table

    DELETE FROM *doctor;*

F. Commit

- **Commit**
  - When inserted data by issuing the INSERT command
    - the changes are only saved in the local database buffer
    - are not saved in the database
    - until you COMMIT the transaction
  - *it is important to remember to COMMIT whenever you have finished inserting values or make changes to the database values*

Commit

- **General Syntax:**
  - Sample of inserting a new record with commit

```
SQL> select * from doctor;

DNAME                DPHONE DSPECIALTY                DROOM
-------------------- ---------- ------------------------- --------
Charles               11437
Allan                 21111 General Practice          A110
Ally                  23214 Neurology                 B301
Ben                   21162 Ophthalmology             C115
Kate                  21907 Ophthalmology             C125
Larry                 32234 Paediatric                B213
Leonard               20987 General Practice          A111
Menson                27242 General Practice          A108
Precilla              25551 Cardiology                B551
Rex                   24113 Psychiatry                B424
Benny                 12345 Radiologist               NULL

11 rows selected.

SQL> INSERT INTO doctor
  2  VALUES ('Jerry','17885','Radiologist','B345');

1 row created.

SQL> commit;

Commit complete.
```

All original records currently in DOCTOR table

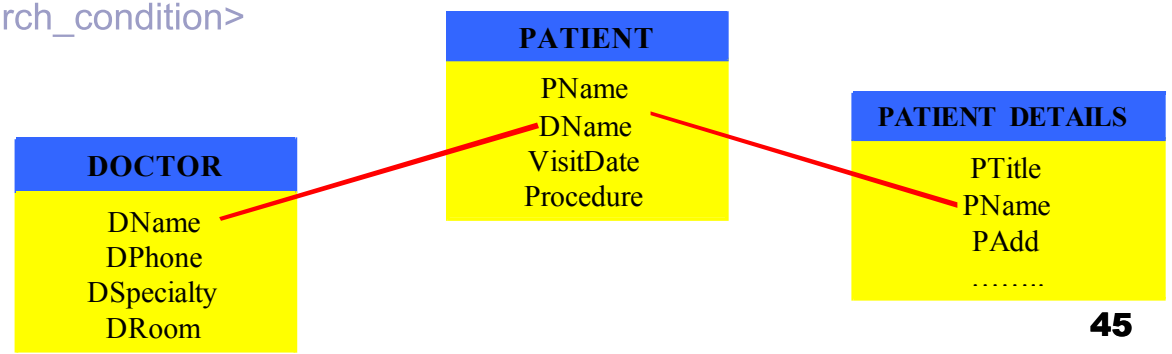
Insert a new record 'Jerry' with commit

Effect of Inserting New Record with Commit

G. Joining Multiple Tables

- **Join**
  - database query to join multiple database tables together
    - the data needed or the conditions specified come from more than one table.
  - **Syntax:**

```
SELECT <attribute1,attribute2,… >
FROM <table_name1,table_name2,…>
WHERE <table_name1.join_attribute> = <table_name2.join_attribut>
AND <search_condition>
```



Joining Multiple Tables

- joining the DOCTOR and PATIENT tables:

```
SELECT doctor.DName, DSpecialty, Patient.PName, VIsitDate, PTitle
FROM patient, doctor, patient_details
WHERE patient.DName = doctor.DName
AND patient.PName = patient_details.PName;
```

- using prefix when joining tables:
  - when more than one table is involved, a prefix for each attribute is recommended to avoid ambiguity

```
SELECT d.DName, d.DSpecialty, p.PName, p.VisitDate, pd.PTitle
FROM patient p, doctor d, patient_details pd
WHERE p.DName = d.DName
AND p.PName = pd.PName;
```

- The output of both example:

DNAME	DSPECIALTY	PNAME	VISITDATE	PTITL
Leonard	General Practice	Kristine	15-FEB-04	Miss
Leonard	General Practice	Albert	12-JAN-04	Mr
Precilla	Cardiology	Cathy	07-MAR-04	Miss
Rex	Psychiatry	Eileen	05-JAN-05	Miss
Rex	Psychiatry	Danny	01-JAN-05	Mr

H. Aggregate Functions and Group By

- **Aggregate Functions**
  - summarize the input table
  - often used include:
    - COUNT - count number of records in the input table
    - SUM - calculate the sum of a numerical attribute
    - MIN and MAX - find the smallest and the largest value of a certain attribute

Aggregate Functions

- **Example:** *returning the number of records that is available from the PATIENT table*

```
SQL> SELECT count(*) FROM patient;

  COUNT(*)
----------
         7
```

Aggregate Functions and Group By

- **Group By**
  - group an input table into a number of groups based on one or more nominated attributes
  - often used in conjunction with aggregate functions
    - *PATIENT table: group by DName*

```
SQL> SELECT dname, count(*)
     FROM patient
     GROUP BY dname;

DNAME        COUNT(*)
-----------  ----------------
Larry        1
Leonard      2
Precilla     1
Rex          3
```

Aggregate Functions and Group By

- *PATIENT table: grouping by DName with combination of getting only the record groups that contain the count value greater than 1 is as follows*

```
SQL> SELECT dname, count(DISTINCT pname)
  2  FROM patient
  3  GROUP BY dname
  4  HAVING count(DISTINCT pname) > 1;

DNAME                COUNT(DISTINCT PNAME)
---------------      ---------------------
Leonard              2
Rex                  3
```

Aggregate Functions and Group By

- *multiple tables: selecting number of patients for each specialty of all doctors*

```
SQL> SELECT d.dspecialty, count(DISTINCT pname)
  2  FROM doctor d, patient p
  3  WHERE d.dname = p.dname
  4  GROUP BY d.dspecialty;

DSPECIALTY           COUNT(DISTINCT PNAME)
-------------------  ---------------------
Cardiology           1
General Practice     2
Paediatric           1
Psychiatry           3
```

Aggregate Functions

- **Count vs. Sum**
  - COUNT - count number of records in the input table
  - SUM - calculate the sum of a numerical attribute

```
SELECT  <attribute1, attribute2>, COUNT(*)
FROM <table_name>
GROUP BY <attribute1, attribute2>;
```

```
SELECT  <attribute1, attribute2>, SUM (attribute3)
FROM <table_name>
GROUP BY <attribute1, attribute2>;
```

I. Alter Tables – Add New Fields

- **Syntax:**

  ALTER TABLE <table_name>

  ADD (attribute_name data_type_declare constraints_declare);

  - ***attribute name***: referring to the new attribute that you want to add into the existing table
  - ***data type***: defines the data type and the size of the new attribute
  - ***constraint***: defines the constraints that the new attribute might be enforced by certain constraints

- **Example:** *add a state attribute to the PATIENT_DETAILS table*

  ALTER TABLE *patient_details*
  ADD *(PState VARCHAR2(5));*

Alter Tables – Modify Fields

- **Syntax:**

  ALTER TABLE <table_name>

  MODIFY (attribute_name new_data_type);

  - ***attribute name:*** refers to the attribute that you want to modify
  - ***new data:*** defines the new data type that you want to use replacing the old one

- **Example:** *change the data type of the PState to CHAR with a size of 30*

  ALTER TABLE *patient_details*

  MODIFY *(PState CHAR(30));*

Alter Tables – Drop Columns

- **Syntax:**

  ALTER TABLE <table_name>

  DROP COLUMN attribute_name;

- **Example:** *delete the attribute PState from the PATIENT table*

  ALTER TABLE *patient_details*

  DROP COLUMN *PState;*