DBMS LAB HANDOUT 1 – B.TECH (CSE), V Sem

– by G.PUVIARASI / SL / SCS / VIT

Data

- A collection of known facts that can be recorded and have implicit meaning.

Database

- A collection of related data.

Database Management Systems

- is a software used for defining , storing , manipulating & maintaining a database

Database System

- The combination of DBMS software and database.

Instances and Schemes

- Databases change over time.
- The information in a database at a particular point in time is called an **instance** of the database.
- The overall design of the database is called the database **scheme**.
- Analogy with programming languages:
  - Data type definition - scheme
  - Value of a variable - instance

DBMS Languages

Data Definition Language (DDL)

- Used to specify a database scheme as a set of definitions expressed in a DDL
- DDL statements are compiled, resulting in a set of tables stored in a special file called a **data dictionary** or **data directory**.
- The data directory contains **metadata** (data about data)
- The storage structure and access methods used by the database system are specified by a set of definitions in a special type of DDL called a **data storage and definition** language
- **basic idea:** hide implementation details of the database schemes from the users

Data Manipulation Language (DML)

- **Data Manipulation** is:
  - **retrieval** of information from the database
  - **insertion** of new information into the database
  - **deletion** of information in the database
  - **modification** of information in the database
- A DML is a language which enables users to access and manipulate data.

- The goal is to provide efficient human interaction with the system.
- There are two types of DML:
  - **procedural**: the user specifies *what* data is needed and *how* to get it
  - **nonprocedural**: the user only specifies *what* data is needed
    - Easier for user
    - May not generate code as efficient as that produced by procedural languages

SQL - Structured Query Language

- Is the defacto standard RDBMS language
- It is based on IBM's Sequel.
- The current version of the standards by ANSI/ISO is SQL-92.
- SQL supports all the DBMS Languages.

More on SQL

- SQL has become *the* standard relational database language. It has several parts:
  - Data definition language (DDL) - provides commands to
    - Define relation schemes.
    - Delete relations.
    - Create indices.
    - Modify schemes.

    Examples **create table , create view** , **create index** , **drop table** , **create function**,etc,.

  - Interactive data manipulation language (DML) - a query language based commands to retrieve , insert, delete and modify tuples.

    Examples **select** , **insert** , **delete** , **update.**

  - Embedded data manipulation language - for use within programming languages like C, Cobol, Pascal, etc.
  - View Definition - commands for defining views

    Example **create view.**

  - Authorization - specifying access rights to relations and views.

    Examples **grant** , **grant select** , **revoke** , **revoke all.**

  - Integrity - a limited form of integrity checking.

    Examples **null**, **not null**, **unique**

  - Transaction control - specifying beginning and end of transactions.

    Examples **commit** , **rollback.**

SQL*Plus

Introduction

- SQL*Plus is the interactive (low-level) user interface to the Oracle database management system.
- SQL*Plus is used to issue queries and to view the query result on the screen.

Guidelines to use SQL

- SQL> is the prompt you get when you are connected to the Oracle database system.
- In SQL*Plus you can divide a statement into separate lines, each continuing line is indicated by a prompt such 2>, 3> etc.
- An SQL statement must always be terminated by a semicolon (;).
- Upper and lower case letters are only important for string comparisons.
- An SQL query can always be interrupted by using <Control>C.
- To exit SQL*Plus you can either type exit or quit.

Editor Commands

The most recently issued SQL statement is stored in the SQL buffer, independent of whether the statement has a correct syntax or not.The buffer can be edited using following commands.

- ✓ l[ist] lists all lines in the SQL buffer and sets the current line (marked with an "_") to the last line in the buffer.
- ✓ l<number> sets the actual line to <number>1
- ✓ c[hange]/<old string>/<new string> replaces the first occurrence of <old string> by <new string> (for the actual line)
- ✓ a[ppend]<string> appends <string> to the current line
- ✓ del deletes the current line
- ✓ r[un] executes the current buffer contents
- ✓ get<file> reads the data from the file <file> into the buffer
- ✓ save<file> writes the current buffer into the file <file>
- ✓ edit invokes an editor and loads the current buffer into the editor.

SQL*Plus Help System and Other Useful Commands

- ✓ To get the online help in SQL*Plus just type help <command>, or just help to get information about how to use the help command.

- ✓ The command passw <user> prompts the user for the old/new password.

- ✓ The command desc[ribe] <table> lists all columns of the given table together with their data types and information about whether null values are allowed or not.

- ✓ You can log your SQL*Plus session and thus queries and query results by using the command spool <file>. All information displayed on screen is then stored in <file> which automatically gets the extension .lst. The command spool off turns spooling off.

- ✓ The command copy can be used to copy a complete table. For example, the command copy from scott/tiger create EMPL using select _ from EMP;
  copies the table EMP of the user scott with password tiger into the relation EMPL. The relation EMP is automatically created and its structure is derived based on the attributes listed in the select clause.

- ✓ SQL commands saved in a file <name>.sql can be loaded into SQL*Plus and executed using the command @<name>.

- ✓ Comments are introduced by the clause rem[ark] (only allowed between SQL statements),

- ✓ or - - (allowed within SQL statements).

Rules followed for naming in oracle.

- Table names can be up to 30 characters long and cannot begin with a number.
- The table name cannot conflict with another object created in the same user account.
- It cannot be the name of an Oracle reserved word.
- There can be up to 254 columns for a single table.

Data types (Domain types) in Oracle

Datatypes (only the commonly used ones):

- **number** is used to declare both real and whole numbers. The data can be up to 38 characters long. (*p,d* are used for precision and number of decimals after the decimal point.
- **date** can store the date and time (day month, year, century, hours, minutes, and seconds). Example: '12/30/1996 23:11:12'
- **varchar2** has a maximum size of 2000 characters. Stores any kind of characters. Oracle only uses the number of bytes required. This is preferred over the **varchar**.
- **char**(*n*) is a fixed length of n, where n is <= 255. Short entries are padded with spaces.
- **long** is like varchar2, except that it holds up to 2GB of characters.
- **raw** holds a maximum length of 255 bytes of binary data.
- **long raw** is used to hold large amounts of binary data.

- ANSI standard types are allowed in the DDL, but get converted to the preferred Oracle equivalent.

Data Definition in SQL

Creating Tables

The SQL command for creating an empty table has the following form:

create table <table> (
<column 1> <data type> [not null] [unique] [<column constraint>],
. . . . . .
<column n> <data type> [not null] [unique] [<column constraint>],
[<table constraint(s)>]
);

For each column, a name and a data type must be specified and the column name must be unique within the table definition. Column definitions are separated by comma. There is no difference between names in lower case letters and names in upper case letters. In fact, the only place where upper and lower case letters matter are strings comparisons.

Constraints

The definition of a table may include the specification of integrity constraints. Basically two types of constraints are provided column constraints are associated with a single column whereas table constraints are typically associated with more than one column. However, any column constraint can also be formulated as a table constraint. The specification of a (simple) constraint has the following form:

[**constraint** <name>] **primary key** | **unique** | **not null**

A constraint can be named. It is advisable to name a constraint in order to get more meaningful information when this constraint is violated due to, e.g., an insertion of a tuple that violates the constraint. If no name is specified for the constraint, Oracle automatically generates a name of the pattern SYS C<number>.

The two most simple types of constraints have already been discussed: not null and unique.Probably the most important type of integrity constraints in a database are primary key constraints.

not null constraint
The constraint requires defined attribute values for that column, different from null.

unique constraint
The constraint specifies that no two tuples can have the same attribute value for this column.

- Unless the condition not null is also specified for a column, the attribute value null is allowed and two tuples having the attribute value null for a column do not violate the constraint.

check constraint
specifies a condition that must be met by all rows in the table.

primary key constraint

- A primary key constraint enables a unique identification of each tuple in a table.
- Based on a primary key, the database system ensures that no duplicates appear in a table.
- The primary key specification is optional, it is generally a good idea to specify one.

Example: The create table statement for our EMP table has the form

```
create table EMP (
        EMPNO char(3) not null,
        ENAME varchar2(30) not null,
        JOB varchar2(10),
        HIREDATE date,
        SAL number(7,2),
        DEPTNO number(2), constraint epk primary key(EMPNO)
);
```

ALTER

- To make alteration to a schema the schema, use the alter table command:

ALTER TABLE <table_name>**Add column** < colname > **datatype [constraint]**

ALTER TABLE <table_name> **Rename column** <old_col_name > **to** < new_col_name >

ALTER TABLE <table_name>**Modify < columnname> datatype:**

ALTER TABLE <table_name> **Drop column <col_name>**

ALTER TABLE <table_name>**Add constraint < constraint _name> check(cond)**

ALTER TABLE <table_name>**Add constraint < constraint_name >unique(attrib_name)**

ALTER TABLE <table_name>**Add constraint < constraint_name >primary key(attr_name)**

ALTER TABLE <table_name> **Disable|Enable|Drop constraint < constraint_name>**

ALTER TABLE <table_name>**Modify constraint < constraint name>**

DROP

- To totally get rid of a schema (structure from catalog) and data

Syntax

Drop Index < index_name>

Drop Synonym < syn_name>

Drop View < view_name>

Drop Sequence < seq_name>

Drop Trigger < trig_name>

Drop Table <tab_name>

Drop Table < tab_name> **cascade|restrict**

- This will cause errors if other tables are referencing the primary key. And the table will not be dropped. To make this happen: drop table *tablename* cascade constraints; Oracle will then drop all referential integrity constraints that refer to primary keys and then drop the table.

DATA MANIPULATION LANGUAGE

Retrival

The contents of a database(table) can be retrived using select command.

select Statement

select [**distinct**] *expression,..., expression*
from *table* [*alias*],...,*table* [*alias*]
[**where** *search-condition*]
[**group by** *column-name...,column-name*]
[**having** *condition*]

distinct removes all duplicates, **all** retains all duplication.
like operator allows for wildcards: **%** matches zero or more characters, **_** makes exactly one character.It is better to use **is null** and **is not null** instead of the equal and unequal operators.Search can be done on ranges, ***between*** *50* ***and*** *100*.

The select Clause

- SQL allows duplicates in relations as well as the results of SQL expressions. In those cases where we want to force the elimination of duplicates, we insert the keyword **distinct** after **select**. The default is to retain duplicates. This can be explicitly required with the keyword **all**.
- The asterisk symbol "*" can be used in place of listing all the attributes.
- A dot notation is used when explicitly identifying the table that the attribute comes from:

The from Clause

The **from** clause defines a Cartesian product of the tables in the clause.

The where Clause

SQL uses **and**, **or** and **not** (not symbols) and the comparison operators <, <=, >, >=, = , and <>. Also available is **between**:

where *amount* **between** 90000 **and** 100000

Additional, **not between** can be used.

Sub-selects

A sub-select is when the where clause contains a **select** statement.

Aggreate Functions

Oracle supports five aggregate functions, **count, sum, avg, max** and **min**.

group by Clause

The **group by** is used to form groups based on the value of certain column values. The aggregate functions then apply to the groups and not the tables.

having Clause

The **having** clause further limits what is in the result when groups do not meet the condition of the clause.

String, Number, and Date Functions

Conversion Functions

- ✓ **to_char** converts a **number** or **date** to a character string
  to_char(1234) becomes '1234'
- ✓ **to_number** converts a character string to a **number**
  to_number('1234') becomes 1234
- ✓ **to_date** converts a character string to a **date**
  to_date('12-DEC-1997 12:31:15')

String Functions

- ✓ **concatentation (||)** Concatenation two or more strings
  fname || ', ' || lname
- ✓ **lpad(string,length,['characters'])** pads a string on the left side until the string is *length* long.
- ✓ **rpad(string,length,['characters'])** pads a string on the right side until the string is *length* long.
- ✓ **ltrim(string, ['characters'])** trims from the left side
- ✓ **rtrim(string, ['characters'])** trims from the right side
- ✓ **lower(string)** converts all the characters to lower case
- ✓ **upper(string)** converts all the characters to upper case
- ✓ **initcap(string)** Converts the first character to upper case
- ✓ **length(string)** returns the length of the string
- ✓ **substr(string,start,[n])** returns the substring starting at *start*
- ✓ **instr(string, 'chars'[,start [,n])** searches *string* for the starting position, starting at *start* and find the *n* occurrence

Numeric Functions

- ✓ **+**
- ✓ **-**
- ✓ *****
- ✓ **/**
- ✓ **abs**
- ✓ **ceil**
  ceil(qoh/olevel)
- ✓ **floor**
  floor(qoh/olevel)
- ✓ **mod**
  mod(qoh, olevel)

- ✓ **power**
  power(qoh, 3)
- ✓ **sqrt**
  sqrt(total)

Date Functions

- ✓ **+, -**
- ✓ **sysdate** current date from the system
- ✓ **next_day(d, day)** returns the first day of the week after d
- ✓ **add_months(d, count)**
- ✓ **last_day(d)** returns the last day of the month for date d
- ✓ **months_between(d2, d1)**
- ✓ **least(d1, d2,...dn)** returns the earliest date
- ✓ **greatest(d1, d2,...dn)** returns the earliest date
- ✓ **trunc(d)** returns the day, with the time as 12:00am midnight
- ✓ **round(d)**
- ✓ **to_char(d, format)**
- ✓ **to_date(s, format)**

Insertion
The database can be populated with data using the insert statement.

Syntax

insert into <table> [(<column i, . . . , column j>)] **values** (<value i, . . . , value j>);

For each of the listed columns, a corresponding (matching) value must be specified. Thus an insertion does not necessarily have to follow the order of the attributes as specified in the create table statement. If a column is omitted, the value null is inserted instead. If no column list is given, however, for each column as defined in the create table statement a value must be given.

Examples:

- ✓ insert into EMP(EMPNO, ENAME, JOB, HIREDATE, SAL,DEPTNO)
  values(‘SHA’, ’HEMA’, ‘LECTURER’, ’03-FEB-01’150000.42,1);

- ✓ insert into EMP
  values(‘CRA’, ’RANI’, ‘LECTURER’, ’11-MAY-99’, 18000,1);

Entering new values each time .

- ✓ insert into EMP values (‘&ecode’,&ename’,’&ej’,’&edate’,’&esal’,’&ed’);

Syntax

If there are already some data in other tables, these data can be used for insertions into a new table. For this, we write a query whose result is a set of tuples to be inserted. Such an insert statement has the form

insert into <table> [(<column i, . . . , column j>)] <query>

Example: Suppose we have defined the following table:
create table OLDEMP (
ENO number(4) not null,
HDATE date);

Now the EMP table can be used to insert tuples into this new table OLDEMP:

- ✓ insert into OLDEMP (ENO, HDATE)
  select EMPNO, HIREDATE from EMP
  where HIREDATE < '31-DEC-60';

Updates

- Update statements are used to update records with/without conditions.
- Update statements can be committed to the database by using explicit "commit" command or it can be rolled back by using "rollback" command.
- For modifying attribute values of (some) tuples in a table, we use the update statement:

Syntax
update <table name> **set**
<column i> = <expression i>, . . . , <column j> = <expression j>
[**where** <condition>];

- An expression consists of either a constant (new value), an arithmetic or string operation, or an SQL query. Note that the new value to assign to <column i> must a the matching datatype.
- An update statement without a where clause results in changing respective attributes of all tuples in the specified table.
- Typically, however, only a (small) portion of the table requires an update.

Examples:

- ✓ The employee rani is transfered to the department 3 her salary is increased by 1000:

  update EMP set
  DEPTNO = 3, SAL = SAL +1000
  where ENAME = 'rani';

- ✓ All employees working in the departments 1 and 2 get a 15% salary increase.
  update EMP set
  SAL = SAL _ 1.15 where DEPTNO in (1,2);

- ✓ Other tables can be used to retrieve data that are used as new values. In such a case we have a <query> instead of an <expression>.

Example:

update EMP set
SAL = (select min(SAL) from EMP
where JOB = 'slect')
where JOB = 'lecturer' and DEPTNO = 1;
Explanation: The query retrieves the minimum salary of all slect. This value then is assigned to all lecturers working in department 1.

- It is also possible to specify a query that retrieves more than only one value (but still only one tuple!). In this case the set clause has the form set(<column i, . . . , column j>) = <query>.It is important that the order of data types and values of the selected row exactly correspond to the list of columns in the set clause.

Deletions
All or selected tuples can be deleted from a table using the delete command:
Syntax

delete from <table> [**where** <condition>];

If the where clause is omitted, all tuples are deleted from the table.
An alternative command for deleting all tuples from a table is the truncate table <table> command. However, in this case, the deletions cannot be undone

Example

- ✓ Delete from EMP;
- ✓ Delete from EMP where deptno = 1

Views

Creating views has the form:

create view *view-name* **as** *select-statement*

Deleting views as the form:

drop view *view-name*

commit and rollback

Changes to the database can be reversed if necessary. **commit** makes the changes permanent and **rollback** removes all changes since the last **commit**. **commit**s are automatic when exiting a SQL session.

Oracle Data Dictionary

The Oracle data dictionary stores the metadata of all objects in the Oracle system. This includes:

- tables
- constraints
- indices
- views
- synonyms
- sequences
- triggers
- procedures
- functions
- packages