I) Overview of schema objects in Oracle

Associated with each database user is a *schema*. A schema is a collection of schema objects. Examples of schema objects include tables, views, indexes, clusters, etc. Schema objects are logical data storage structures.

1. **Tables**

*Tables* are the basic unit of data storage in an Oracle database. Data is stored in *rows* and *columns*. You define a table with a *table name* (such as EMP) and set of columns. You give each column a *column name* (such as EMPNO, ENAME, and JOB), a *datatype* (such as VARCHAR2, DATE, or NUMBER), and a *width* (the width might be predetermined by the datatype, as in DATE) or *precision* and *scale* (for columns of the NUMBER datatype only). A row is a collection of column information corresponding to a single record.

You can optionally specify rules for each column of a table. These rules are called *integrity constraints*.

Once you create a table, you insert rows of data using SQL statements. Table data can then be queried, deleted, or updated using SQL.

2. **Oracle Datatypes**

Each column value and constant in a SQL statement has a *datatype*, which is associated with a specific storage format, constraints, and a valid range of values. When you create a table, you must specify a datatype for each of its columns.

Among the Oracle built-in datatypes are:

- **The character datatypes**: store character (alphanumeric) data in strings. We distinguish:
  - *The CHAR datatype*: stores **fixed**-length character strings. When you create a table with a CHAR column, you must specify a string length. (The default is 1.) Oracle then guarantees that:
    - When you insert or update a row in the table, the value for the CHAR column has the fixed length.
    - If you give a shorter value, the value is blank-padded to the fixed length.
    - If a value is too large, Oracle returns an error.
  - *VARCHAR2 and VARCHAR Datatypes*: The VARCHAR2 datatype stores variable-length character strings. When you create a table with a VARCHAR2 column, you specify a maximum string length. For each row, Oracle stores each value in the column as a variable-length field (unless a value exceeds the column's maximum length, in which case Oracle returns an error).
    - For example: assume you declare a column VARCHAR2 with a maximum size of 50 characters. If only 10 characters are given for the VARCHAR2 column value in a particular row, the column in the row's row piece stores only the 10 characters not 50. The VARCHAR datatype is currently synonymous with the VARCHAR2 datatype.
• **The NUMBER datatype**: stores fixed and floating-point numbers. The positive, negative and zero numbers can be stored in a NUMBER column. For numeric columns you can specify the column as:
  - column_name NUMBER
  - column_name NUMBER (precision, scale)
    Optionally, you can also specify a precision (total number of digits) and scale (number of digits to the right of the decimal point).
  If a precision is not specified, the column stores values as given. If no scale is specified, the scale is zero.
• **The DATE datatype**: stores point-in-time values (dates and times) in a table. The DATE datatype stores the year (including the century), the month, the day, the hours, the minutes, and the seconds (after midnight).

3. **Data integrity**

Oracle uses integrity constraints to prevent invalid data entry into the base tables of the database. You can define integrity constraints to enforce the business rules you want to associate with the information in a database. If any of the results of a DML statement execution violate an integrity constraint, Oracle rolls back the statement and returns an error.

Oracle data integrity constraints are:

**NOT NULL Integrity Constraints**

By default, all columns in a table allow nulls (the absence of a value). A NOT NULL constraint requires a column of a table contain no null values. For example, you can define a NOT NULL constraint to require that a value be input in the ENAME column for every row of the EMP table.

**UNIQUE Key Integrity Constraints**

A UNIQUE key integrity constraint requires that every value in a column or set of columns (key) be unique—that is, no two rows of a table have duplicate values in a specified column or set of columns. The column (or set of columns) included in the definition of the UNIQUE key constraint is called the *unique key*.

**PRIMARY KEY Integrity Constraints**

Each table in the database can have at most one PRIMARY KEY constraint. The values in the group of one or more columns subject to this constraint constitute the unique identifier of the row. In effect, each row is named by its primary key values.

The Oracle implementation of the PRIMARY KEY integrity constraint guarantees that both of the following are true:
  - No two rows of a table have duplicate values in the specified column or set of columns.
  - The primary key columns do not allow nulls (that is, a value must exist for the primary key columns in each row).

**FOREIGN KEY (Referential) Integrity Constraints**

A FOREIGN KEY constraint is used whenever two tables are related by a common column (or set of columns). For example, to guarantee that every value of the column DNO of the table EMP must match a value of the column DNUMBER of the table DEP, we have to define a PRIMARY or UNIQUE key constraint on the column DNUMBER in the parent table (i.e. DEP) and a FOREIGN KEY constraint in the column DNO in the child table (i.e. EMP) in order to maintain the relationship between the two table.
Foreign keys can consist of multiple columns. However, a composite foreign key must reference a composite primary or unique key with the same number of columns and the same datatypes.

Referential integrity constraints can specify particular actions to be performed on the dependent rows in a child table if a referenced parent key value is modified. The referential actions supported by the FOREIGN KEY integrity constraints of Oracle:

- **UPDATE and DELETE No Action**: For example, if a primary key value is referenced by a value in the foreign key, the referenced primary key value cannot be deleted because of the dependent data.
- **DELETE CASCADE**: For example, if a row in a parent table is deleted, and this row's primary key value is referenced by one or more foreign key values in a child table, the rows in the child table that reference the primary key value are also deleted from the child table.
- **Delete Set Null**: For example, if we delete a department, then all the rows for all employees working in that department will have their DNO value set to null.

<table>
<thead>
<tr>
<th>DML Statement</th>
<th>Issued Against Parent Table</th>
<th>Issued Against Child Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT</td>
<td>Always OK if the parent key value is unique.</td>
<td>OK only if the foreign key value exists in the parent key or is partially or all null.</td>
</tr>
<tr>
<td>UPDATE No Action</td>
<td>Allowed if the statement does not leave any rows in the child table without a referenced parent key value.</td>
<td>Allowed if the new foreign key value still references a referenced key value.</td>
</tr>
<tr>
<td>DELETE No Action</td>
<td>Allowed if no rows in the child table reference the parent key value.</td>
<td>Always OK.</td>
</tr>
<tr>
<td>DELETE Cascade</td>
<td>Always OK.</td>
<td>Always OK.</td>
</tr>
<tr>
<td>DELETE Set Null</td>
<td>Always OK.</td>
<td>Always OK.</td>
</tr>
</tbody>
</table>

**CHECK Integrity Constraints**

A CHECK integrity constraint on a column or set of columns requires that a specified condition be true or unknown for every row of the table. If a DML statement results in the condition of the CHECK constraint evaluating to false, the statement is rolled back.

CHECK constraints enable you to enforce very specific or sophisticated integrity rules by specifying a check condition. The condition of a CHECK constraint has some limitations:

- It must be a Boolean expression evaluated using the values in the row being inserted or updated, and
- It cannot contain subqueries, sequences, the SQL functions SYSDATE, UID, USER, or USERENV, or the pseudocolumns LEVEL or ROWNUM.

4. **Views**

A view is a tailored presentation of the data contained in one or more tables (or other views). A view takes the output of a query and treats it as a table; therefore, a view can be thought of as a "stored query" or a "virtual table". You can use views in most places where a table can be used.
Since views are derived from tables, they have many similarities. For example, you can query views, and with some restrictions you can update, insert into, and delete from views. All operations performed on a view actually affect data in some base table of the view and are subject to the integrity constraints and triggers of the base tables.

Unlike a table, a view is not allocated any storage space, nor does a view actually contain data; rather, a view is defined by a query that extracts or derives data from the tables the view references. These tables are called base tables. Base tables can in turn be actual tables or can be views themselves. Because a view is based on other objects, a view requires no storage other than storage for the definition of the view (the stored query) in the data dictionary.

Views provide a means to present a different representation of the data that resides within the base tables. Views are very powerful because they allow you to tailor the presentation of data to different types of users. Views are often used:

- To provide an additional level of table security by restricting access to a predetermined set of rows and/or columns of a table. For example, we can create the STAFF view that does not show the SALARY column of the base table EMP.
- To hide data complexity. For example, a single view might be defined with a join, which is a collection of related columns or rows in multiple tables. However, the view hides the fact that this information actually originates from several tables.
- To simplify commands for the user. For example, views allow users to select information from multiple tables without actually knowing how to perform a join.
- To present the data in a different perspective from that of the base table. For example, the columns of a view can be renamed without affecting the tables on which the view is based.
- To isolate applications from changes in definitions of base tables. For example, if a view's defining query references three columns of a four column table and a fifth column is added to the table, the view's definition is not affected and all applications using the view are not affected.
- To express a query that cannot be expressed without using a view. For example, a view can be defined that joins a GROUP BY view with a table, or a view can be defined that joins a UNION view with a table.
- To save complex queries. For example, a query could perform extensive calculations with table information. By saving this query as a view, the calculations can be performed each time the view is queried.

Oracle stores a view's definition in the data dictionary as the text of the query that defines the view. When you reference a view in a SQL statement, Oracle merges the statement that references the view with the query that defines the view.

A join view is defined as a view that has more than one table or view in its FROM clause (a join) and that does not use any of these clauses: DISTINCT, AGGREGATION, GROUP BY, START WITH, CONNECT BY, ROWNUM, and set operations (UNION ALL, INTERSECT, and so on).

An updatable join view is a join view, which involves two or more base tables or views, where UPDATE, INSERT, and DELETE operations are permitted. The data dictionary views ALL_UPDATABLE_COLUMNS, DBA_UPDATABLE_COLUMNS, and USER_UPDATABLE_COLUMNS contain information that indicates which of the view columns are updatable.
II) Questions

1. Create into your account a table “Emp” containing 4 columns (i.e. fields):
   a. Empno : employee number on three digits. This column is the primary key of the table Emp.
   b. Ename: the employee name, which is a variable string that couldn’t exceed 20 characters. This field could not be empty.
   c. Salary: the employee salary, which is a number.
   d. Entry: the employee joining date, which is a date.

2. Retrieve all the objects created in you account

3. Give the structure of the table Emp.

4. Add at the end of the table Emp a new column named Age represented as a number on 3 digits. After this modification, list the structure of the table again.

5. Change the size of the datatype of the column Ename to allow strings up to 30 characters. List the structure of the table again.

6. Add a default value equals to 100 for the column Salary.

7. Drop the column age from the table Emp. List the structure of the table again.
8. Create into your account another table “Dep” containing 2 columns:
   a. Dnumber: the department number on one digit. This column is the primary key of the table Emp.
   b. Dname: the department name, which is a fixed string of 15 characters.

9. Add a new column to the table Emp, named Dno. On that column, create a foreign key constraint, named FK1, which allows referencing the key attribute of the table Dep.

10. Add a constraint, named CH1, on the column salary that checks that the values entered are always positives.

11. Remove the foreign key constraint FK1.

12. re-create the foreign key constraint FK1 on the field DNO of Emp, which allows referencing the key attribute of the table Dep. Also, this constraint will allow deleting all the employees of a department if this department is deleted

13. Insert theses tuples into the table Dep. Display the table Dep.

<table>
<thead>
<tr>
<th>Dnumber</th>
<th>Dname</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Personal</td>
</tr>
<tr>
<td>2</td>
<td>Store</td>
</tr>
<tr>
<td>3</td>
<td>Planning</td>
</tr>
<tr>
<td>4</td>
<td>Accounting</td>
</tr>
<tr>
<td>4</td>
<td>External</td>
</tr>
</tbody>
</table>

14. Insert these tuples to the table Emp. Display the table Emp

<table>
<thead>
<tr>
<th>Empno</th>
<th>Ename</th>
<th>Salary</th>
<th>Entry</th>
<th>Dno</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>Ali</td>
<td>200</td>
<td>01-jan-2001</td>
<td>1</td>
</tr>
</tbody>
</table>
15. Insert a new employee having a number equals to 110, a name Emad and omitting all the other information. Display the table Emp.

16. Create a table Temp that is a copy of the table Emp but only with the following columns {Empno, Ename, entry, dno}. Display the table Temp.

17. Retrieve all the employees. Retrieve all the departments.

18. Give the name of employees that joined their department before the 1st January 2001.

19. Give the employee number, the employee name, and the department name for all the employees.

20. List the employee number and the employee name (in ascendant order of the employees name) the employees having a name containing the letter m or beginning with the letter a.
21. Update the salary of the employee having empno=114 with the minimum salary of the department having as number 1. Then display the table Emp after.

22. Update the information of the employee having empno=113 like this: the entry value to “02-jan-2001” and the salary to 550. Display the table Emp.

23. Delete the information of the department having the number 4. Display the tables Dep and the table Emp.

24. Create a view, named Dep_Info, giving for each department having more than an employee the following information: department number, number of employees and their average salary. Then, Display the objects contained in your account.

25. List the information of the department number equals to 1.
III) Questions

Use the Mesri’s database, found in my account, to answer the following questions:

26. Retrieve the name of male employees who were born before the year 1938

27. Retrieve the supervisor name for employees living at ‘Houston’ and working at a department located at the same town.

28. List the different dependent relationships.

29. Retrieve the SSN of employees that do not work for any project controlled by department number 4.

30. Retrieve the name of each employee who works on all projects controlled by department 5 (explain how the SQL will proceed to execute this statement)
31. For each employee, retrieve the name of his or her immediate supervisor.

32. For each department which it is assigned more than two employees, list the department number, the department name, the number of employees assigned and the average salary.

33. Create a view called project_summary, which lists for each project the project number, the project name, the number of employees, and the minimum and maximum salary.