**Learning outcomes:**

* To understand the of major RDBMS concepts and their function.
* To understand the importance of the constraint and they can use it efficiently.
* To use the inbuilt functions proficiently.

Unit Contents Summary

3.1 Introduction

3.2 Series and Data Frame

3.3 Index

3.4 Rank

3.5 Sort

**3.1 Introduction:**

A relational database is a collection of information that organizes data in predefined

relationships where data is stored in one or more tables (or "relations") of columns and rows,

making it easy to see and understand how different data structures relate to each other. SQL

(Standard Query Language) is developed specifically for querying and manipulating data.

There are many RDBMS Software are available in the market. The popular languages are “SQL

server” developed by Microsoft, “Oracle” developed by Oracle Corporation, “MySQL” is

developed by Oracle but it is open-source.

There may be slight changes in the syntax of these languages. But most of the Statements will

run smoothly.

**ACID Properties of RDBMS**

**1. Atomicity:**

This property is very useful. Let's discuss an example first. You have booked a ticket

for travel and the fare has been deducted from your account but no seat has been

assigned. This shouldn't happen. Either situation should be done, or none at all. That is,

either the entire transaction occurs at once or none at all.

**2. Consistency:**

This means that integrity constraints must be satisfied in order for the database to

remain consistent before and after a transaction. This means that even if you book a

ticket, the website must display all information accurately

**3. Isolation:**

In an RDBMS each transaction should be separate and should not depend on other

transactions. This is made possible by the isolation property. Isolation hides the effects

of a transaction until it is committed. This reduces the risk of confusion. For example,

when booking a ticket on a travel booking website. Seat assignments are made only

after the fare has been debited from your account. Until then, it can be used by another

User.

**4. Durability:**

Durability means that data can be recovered after a failed transaction. Let's continue

with the same example. If you want to cancel your ticket, it should be possible to do

so on your website. And that space must be available to another user.

**Features of RDBMS**

**1. Data Redundancy:**

An RDBMS we should make sure to completely avoids storing duplicate data. Storing

the same data in multiple files is not just a waste of time, money and disk space.

**2. Data Inconsistency:**

Data inconsistency means that different copies of the same data do not match. For

example, if you have multiple copies of the same data and have changed the data.

Suppose the phone numbers are stored different in one file and different in another.

Therefore, different copies of the same data will not match. The same underlying data

existing in different files with different meanings is data inconsistency.

**3. Data Isolation:**

Data segregation means that the data is spread across different files and files of

different formats. Writing a new application program to retrieve the data is difficult.

Since all files have different formats, getting information from these files is very

difficult and nothing more than data separation.

**4. Data Integrity:**

Data integrity means that data values may need to satisfy integrity constraints. For

example, if you are managing a bank database and account balance is an attribute,

assume that some constraints are satisfied. Every customer should have at least 1000/-

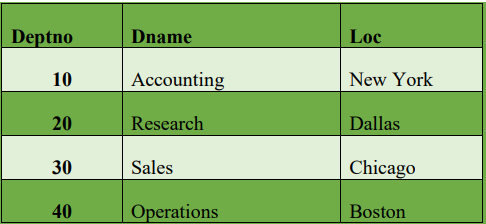
Rs Balance. This is called integrity constraints.

**Reference Table for this chapter**

In order to understand all the concept in Relational Database Management System. Students

need to know the syntax as well as practical based example. And for that we need to use the

tables. The tables which we are going to use in this chapter is given below:



create table dept(

deptno number(2,0),

dname varchar2(14),

loc varchar2(13),

constraint pk\_dept primary key (deptno)

)

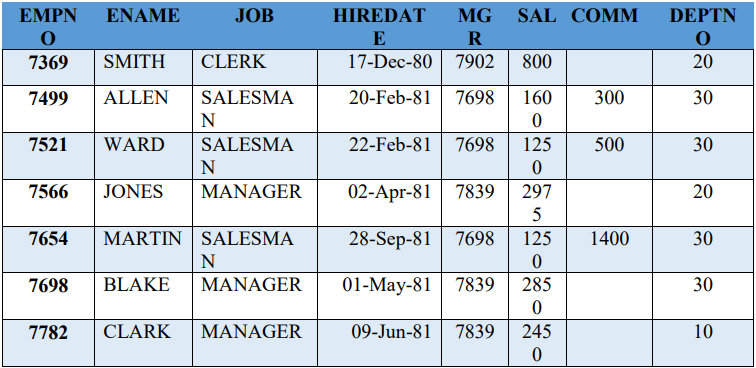
insert into DEPT (DEPTNO, DNAME, LOC) values(10, 'ACCOUNTING', 'NEW

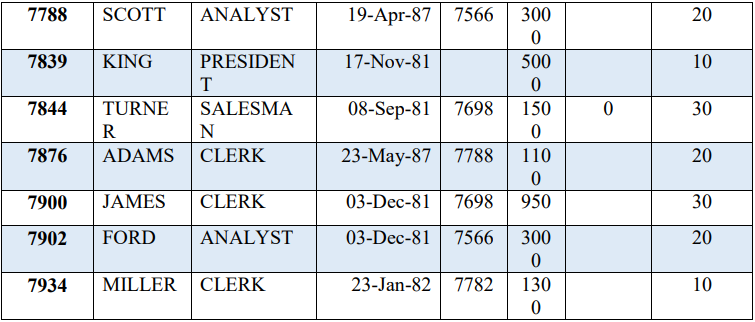
YORK')

insert into dept values(20, 'RESEARCH', 'DALLAS')

insert into dept values(30, 'SALES', 'CHICAGO')

insert into dept values(40, 'OPERATIONS', 'BOSTON')





create table emp(

empno number(4,0),

ename varchar2(10),

job varchar2(9),

mgr number(4,0),

hiredate date,

sal number(7,2),

comm number(7,2),

deptno number(2,0),

constraint pk\_emp primary key (empno),

constraint fk\_deptno foreign key (deptno) references dept (deptno)

)

insert into emp values(7839, 'KING', 'PRESIDENT', null,to\_date('17-11-1981','dd-mmyyyy'),5000, null, 10)

insert into emp values(7698, 'BLAKE', 'MANAGER', 7839,to\_date('1-5-1981','dd-mmyyyy'), 2850, null, 30)

insert into emp values(7782, 'CLARK', 'MANAGER', 7839,to\_date('9-6-1981','dd-mmyyyy'), 2450, null, 10)

insert into emp values(7566, 'JONES', 'MANAGER', 7839,to\_date('2-4-1981','dd-mmyyyy'),2975, null, 20)

insert into emp values(7788, 'SCOTT', 'ANALYST', 7566,to\_date('13-JUL-87','dd-mmrr') - 85,3000, null, 20)

insert into emp values(7902, 'FORD', 'ANALYST', 7566,to\_date('3-12-1981','dd-mmyyyy'), 3000, null, 20)

insert into emp values(7369, 'SMITH', 'CLERK', 7902,to\_date('17-12-1980','dd-mmyyyy'), 800, null, 20)

insert into emp values(7499, 'ALLEN', 'SALESMAN', 7698,to\_date('20-2-1981','dd-mmyyyy'),1600, 300, 30)

44

insert into emp values(7521, 'WARD', 'SALESMAN', 7698,to\_date('22-2-1981','dd-mmyyyy'),1250, 500, 30)

insert into emp values(7654, 'MARTIN', 'SALESMAN', 7698,to\_date('28-9-1981','ddmm-yyyy'), 1250, 1400, 30)

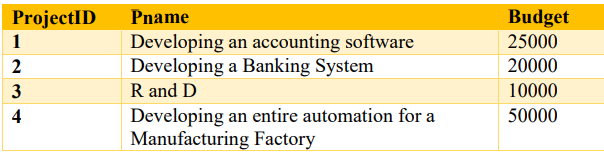
insert into emp values(7844, 'TURNER', 'SALESMAN', 7698,to\_date('8-9-1981','dd-mmyyyy'), 1500, 0, 30)

insert into emp values(7876, 'ADAMS', 'CLERK', 7788,to\_date('13-JUL-87', 'dd-mm-rr')

- 51,1100, null, 20)

insert into emp values(7900, 'JAMES', 'CLERK', 7698,to\_date('3-12-1981','dd-mmyyyy'),950, null, 30)

insert into emp values(7934, 'MILLER', 'CLERK', 7782,to\_date('23-1-1982','dd-mmyyyy'),1300, null, 10)



create table Project(

ProjectID number(2,0),

Pname varchar2(25),

Budget number(7,2),

constraint pk\_Proj primary key (ProjectID)

)

insert into Project values(1, “Developing a accounting software”, 25000)

insert into Project values(2, “Developing a Banking System”, 20000)

insert into Project values(3, “R and D”, 10000)

insert into Project values(1, “Developing an entire automation for a Manufacturing Factory”,

50000)

**Constraints**

1. NOT NULL constraints

2. Unique constraints

3. Primary key constraints

4. (Table) Check constraints

5. Foreign key (referential) constraints

**1. NOT NULL constraints**

NOT NULL constraints prevent database values from becoming null. Column cannot

be empty. This means that the data must be stored in this column.

**2. Unique constraints**

A unique constraint ensures that each value of a given column must be unique. In other words, a unique constraint prevents multiple rows from having the same value in the same column or combination of columns, but allows some values to be null. A table can have any number of unique constraints.

**3. Primary key constraints**

A primary key is a column that contains values that uniquely identify each row in a table. A database table must have a primary key. A primary key is a column or combination of columns that has the same properties as a unique constraint. This means that the primary key constraint combines the NOT NULL and unique constraints in one declaration. That is, it prevents multiple rows from having the same value in the same column or combination of columns, and prevents values from being null. You can use primary key and foreign key constraints to define relationships between tables. A table can have only one primary key.

**4. (Table) Check constraints**

Check conditions require values in the database to meet certain conditions. These

conditions are specified by user. So, the example we discussed that the in the bank

account the balance must be at least 1000/- Rs. This type of constraints can be applied

using check constraints.

**5. Foreign key (referential) constraints**

Foreign key constraints allow you to define desired relationships between and within

tables. Referential integrity is enforced by adding foreign key (or referential)

constraints to the table and column definitions and creating an index on all foreign key

columns. Once index and foreign key constraints are defined, changes to table and

column data are checked against the defined constraints. Completion of the requested

action depends on the results of constraint checks. A foreign key constraint requires

values in one table to match values in another table.

**Note 1:** These constraints are created when the table is created or after the table is created with alter table, so see the create table statement to see the full syntax, including examples. Either way, you have to create the table.

**Note 2:**A constraint can be one of the following:

**A column-level constraint**

You can see all the syntax and examples by looking at the create table statement which we have seen in the beginning at “03.05 reference tables for this chapter”. Column-level

constraints refer to a single column within a table and do not specify a column name (except check constraints).

**A table-level constraint**

Table-level constraints are the constraints which are applied to one or more columns within a table. Table-level constraints specify the name of the column to which they apply.

**Relation between two Table**

As the name suggest in RDBMS. The most important is how you can relate tables.

**Condition for relating two tables:**

if you want to connect the two tables then both tables must obey some condition. Let's look at this condition:

* Both tables must have common fields. This common field must have the same size, data type, and format.
* At least one field in each data table must be defined as a primary key.
* A database cannot contain multiple tables with the same name.

As you can see, the tables “Emp” and “Dept” satisfy all these conditions. Both tables have a common column "deptno". The size, data type, and format are the same for both tables.

There are three types of relation in RDBMS.

* One to One
* One to Many
* Many to Many

**One to One:**

o Both tables can only have one record on each side of the relationship.

o Each primary key value references only zero or his one record in the associated

table.

o Most one-to-one relationships are enforced by business rules and not naturally

followed by data. Even without such a rule, you can usually combine both tables

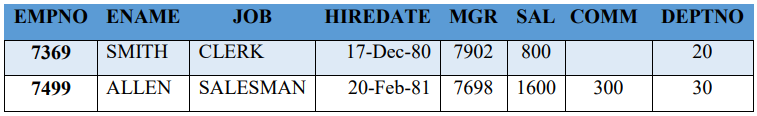
without violating the normalization rules.

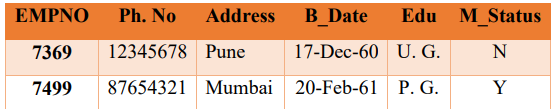
o For example, to store personal information (phone number, address, date of birth,

education, marital status, etc.) for the employees specified in the "employees" table

above. This new table will have a one-to-one relationship with the "emp" table. The

"empDetail" table has only one record for each employee.





**One to Many:**

o A primary key table contains only one record related to zero, one, or many records

in the related table.

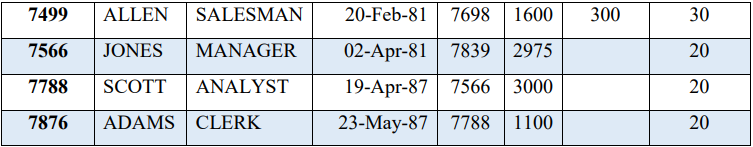
o For example, each department may have multiple employees. Each employee

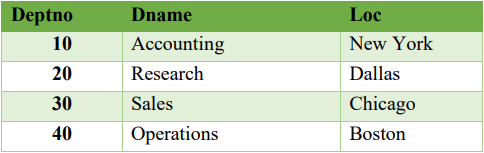
belongs to only one department. Here you can see that the “SMITH” is working in

dept 20. And there are many employees who are working in the same department.

But in “dept“ table there is only one record for ddeptno 20.







**Many to Many:**

o Each record in both tables can reference zero or any number of records in the other

table. These relationships always require third table, called the related or linked

table. This is because relational systems cannot record relationships directly.

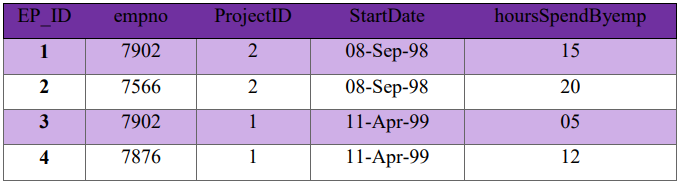
o For example, you want to assign an employee to a project. A project can have

multiple collaborators working on it. Also, an employee can be assigned multiple

projects. In this situation, you should create a third table like this: the employee

whose empno is 7902 are working two different projects. And the project is having

multiple employees working in that project from different department.



**Data types:**

There are many built in data types are available. We are going to discuss commonly used data types here.

| Sr. No. | Type | Description |
| --- | --- | --- |
| 1 | VARCHAR2 (size [BYTE | CHAR]) | Variable-length character string having maximum length size bytes or characters. Maximum size is 4000 bytes or characters, and minimum is 1 byte or 1 character. You must specify size for VARCHAR2. |
| 2 | NUMBER (p, s) | Number having precision p and scale s. The precision p can range from 1 to 38. The scale s can range from -84 to 127. |
| 3 | DATE | Valid date ranges from January 1, 4712 BC to December 31, 9999 AD. |

**Types of Statement:**

There is difference of opinion on the number of types of statement. So, we discuss the

popular types. So, the major types are as follows:

**1. Data Definition Language (DDL) Statements**

Data definition language (DDL) statements are used to perform create, drop, alter, truncate.

**2. Data Manipulation Language (DML) Statements**

Data Manipulation Language (DML) statements are used to perform insert, update and delete.

**3. Transaction Control Statements (TCL)**

Transaction Control Language (TCL) Statements are used to perform commit, savepoint and

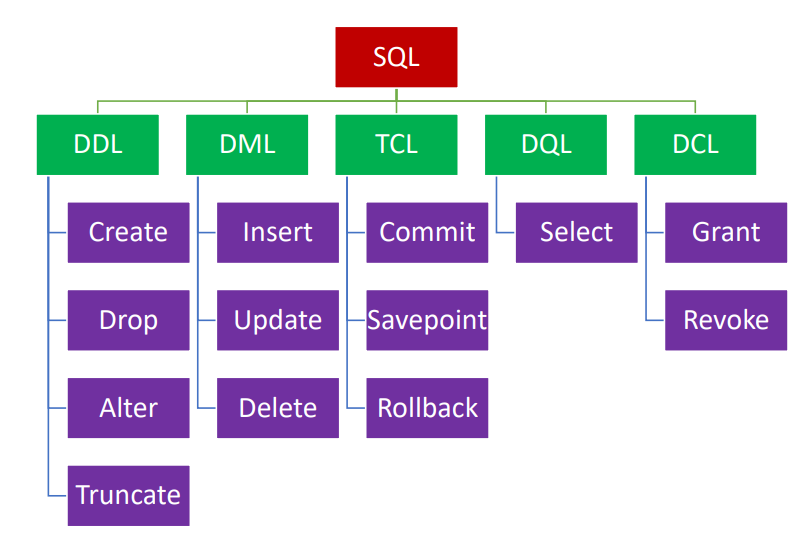
Rollback.

**4. Data Query Language (DQL)**

Data Query language (DQL) statements are used to perform “select”.

**5. Data Control Statements (DCL)**

Data Control Language (DCL) Statements are used to perform Grant and Revoke.

****

**SQL Statement**

**1. Create table:**

Table is the basic structure to store user data / observations.

create table tableName (column datatype [Default expr] [,...])

create table dept(deptno number(2,0), dname varchar2(14), loc varchar2(13),

constraint pk\_dept primary key (deptno))

As you can see that the primary key constraint is also given. There are 3 columns here

named deptno, dname, loc and their respective datatypes are as number, varchar2,

Varchar2.

**NOTE:** The name of the constraint is optional here it is pk\_dept. But if you do not

provide the name then it will be auto generated.

We already discussed the concept of constraint. Let us see the constraints example now.

**Not NULL:**

dname varchar2(14) constraint dnmNN NOT NULL.

The constraint dnmNN ensures that no dname in the table has a null dname. So, every

department must have a name. remember that blank spaces are different than NULL

**Primary Key:**

dept(deptno number(2,0) constraint pk\_dept primary key

you can also write column level constraint when you defining the column or at

the end of create statement. Normally, if there is a single column primary key

you can write at the time of defining the column.

Otherwise, if the primary key is a combination of two column, then you must

give at the end of the statement. As follows:

create table dept(deptno number(2,0), dname varchar2(14), loc varchar2(13),

constraint pk\_dept primary key (deptno, dname) )

This is table level constraint. You can see that we are written this at the end of

the statement.

**Unique constraint**

dname varchar2(14) constraint dnmU unique

The constraint dnmU identifies the dname column as a unique key. This constraint

ensures that no two-department name in the table have the same.

**Check Constraint**

create table dept(

deptno number (2,0) constraint check\_dno Check (div\_no BETWEEN 1 AND 30) ,

dname varchar2(14) Constraint check\_dname Check (dname = UPPER(dname)),

loc varchar2(13) Constraint check\_loc

Check (loc IN ('Dallas','Boston', ‘Chicago’, ‘New York’)),

constraint pk\_dept primary key (deptno)

)

o check\_dno ensures that no depatment numbers are less than 1 or greater than

30.

o check\_dname ensures that all department names are in uppercase.

o check\_loc restricts locations to 'Dallas','Boston', ‘Chicago’, ‘New York’.

**Foreign Key Constraints**

create table emp(

empno number(4,0),

ename varchar2(10),

job varchar2(9),

mgr number(4,0),

hiredate date,

sal number(7,2),

comm number(7,2),

deptno number(2,0),

constraint pk\_emp primary key (empno),

constraint fk\_deptno foreign key (deptno) references dept (deptno)

)

The constraint fk\_deptno ensures that all departments given for employees in the

dept table are present in the departments table. However, employees can have null

department numbers, meaning they are not assigned to any department. To ensure

that all employees are assigned to a department, you could create a NOT NULL

constraint on the deptno column in the dept table in addition to the REFERENCES

constraint.

**2. Alter table:**

* ALTER TABLE table\_name ADD column\_name column-definition
* ALTER TABLE table\_name MODIFY column\_name column\_type
* ALTER TABLE table\_name DROP COLUMN column\_name

With help of Alter table statement, you can

* Add a column
* Modify the column
* Drop the column
* alter table dept add column phone\_no varchar2(8)
* alter table dept modify column phone\_no varchar2(10)
* alter table dept drop column phone\_no

Here the column phone\_no is added in to the dept table. Then the same columns width is increased and in the last statement the same column is dropped

**2. Drop table:**

Drop table statement is used to move a table to the recycle bin.

The example of drop statement is given below. Here, Product table is deleted.

drop table product

**3. Truncate table:**

Instead of deleting the table all together if you want to delete the all the records only then you can use truncate table statement. The example of truncate statement is given below. Here, all the rows / records of Product table are deleted.

truncate table product

**4. Insert:**

To add a record in the existing table, insert command is used.

insert into DEPT (DEPTNO, DNAME, LOC) values (10, 'ACCOUNTING', 'NEW YORK')

OR

insert into DEPT values (10, 'ACCOUNTING', 'NEW YORK')

here we are adding one record into the dept table. Writing the column name is optional but

them you must provide values for all the columns in the table and the sequence of the column

must be the same as in the table. So, the following statement is wrong.

insert into DEPT values ('ACCOUNTING', 'NEW YORK', 10)

**5. Update:**

The update statement updates the value of one or more columns for all rows / records of the table for which the WHERE clause evaluates to TRUE. If "where" clause is not provided then it will change for all the records of the table.

Update table-Name [[AS] correlation-Name] Set column-Name = Value

update dept set dname = 'Marketing' where depto = 30

Here the department name is changed to "Marketing" for the department whose deptno is 30.

we will discuss 'where' clause in detail when we discuss 'select'

**6. Delete:**

The delete statement removes entire rows of data from a specified table for which the where

clause evaluates to TRUE.

delete from dept where deptno = 30

the delete statement without where clause is equivalent to truncate

**7. Commit:**

Use the COMMIT statement to end your current transaction and make permanent all changes

performed in the transaction. A transaction is a sequence of SQL statements that Oracle

Database treats as a single unit. This statement also erases all savepoints in the transaction

and releases transaction locks.

Until you commit a transaction:

* You can see any changes you have made during the transaction by querying the
* modified tables, but other users cannot see the changes. After you commit the
* transaction, the changes are visible to other users' statements that execute after the
* commit.
* You can roll back (undo) any changes made during the transaction with the ROLLBACK statement.

**8. Rollback:**

Use the ROLLBACK statement to undo work done in the current transaction or to manually

undo the work done by an in-doubt distributed transaction.

A simple rollback or commit erases all savepoints. When you roll back to a savepoint, any

savepoints marked after that savepoint are erased. The savepoint to which you roll back

remains.

**9. Savepoint:**

The SAVEPOINT statement names and marks the current point in the processing of a

transaction. With the ROLLBACK TO statement, savepoints undo parts of a transaction

instead of the whole transaction.

**10. Grant:**

Use the GRANT statement is used to give privileges to a specific user or role, or to all users, to perform actions on database objects.

GRANT privilege-type ON [TABLE] { table-Name} TO grantees

Grant select On table emp to user1

Grant update On table emp to user1

**11. Revoke**

Use the REVOKE statement to remove privileges from a specific user, or from all users, to

perform actions on database objects.

REVOKE privilege-type ON [ TABLE ] { table-Name | view-Name } FROM grantees

revoke update on table emp from user1

**12 Select**

The SELECT statement is used to select data from a database. The data returned is stored in a

result table, called the result-set.

**To retrieve all the columns of all rows**

Select \* from dept

**To retrieve selected columns of all rows**

Select dname, loc from dept

**Using arithmetic operators**

Select ename, sal, sal+2000 from emp

**Defining a column alias**

Select empno, ename as “Employee Name”, sal as Salary from emp

**Concatenating**

Select ename || ‘ working as a ‘ || Job from emp

**Eliminating duplicate rows**

Select distinct deptno from emp

**To see the structure of table**

Describe emp

**Limiting number of rows using a selection clause “where”**

Select \* from emp where sal >= 2000

**NoSQL:**

There are many opinions about the name NoSQL, it is from "not only SQL" it means, NoSQL

can store variety type of information. Or the term NoSQL is from the word "Non SQL". Any

way the main point is NoSQL stores data in way different from SQL. as it's a non-tabular

Databases.

“A NoSQL database provides a mechanism for storage and retrieval of data that is modelled in

means other than the tabular relations used in relational databases.”14

In recent advanced technology generates / collects data in many different ways. now the data

on which we analyse is structure, semi structured and polymorphic.

let us see the example of semi structured data

* email
* whatsapp data
* emoji
* social media -- twitter, facebook, Linedln
* web site data -- instagram YouTube
* mobile and its communication data
* scientific data -- financial report of any company, survey report
* Digital surveillance -- video or audio files
* Satellite image

as you can see that it is not possible to store this type of data in any RBMS. Hence, the solution

is NoSQL. in NoSQL you can store all type of data.

**Features of NoSQL**

**Flexible schemas:** In RBMS, you already know which type of data you are going to store hence

you can create the table according to the data. But in NoSQL, you don't know the type of data

/ information in advance which you want to store. In NoSQL the schema is flexible. In NoSQL

one single document can hold.

If you want to make any changes in the document (like Add a new field, make change in the

data, or delete the field, modify the data type) then instead of making the change you need to

create a new document altogether.

**Horizontal scaling:** in RBMS the scaling is vertical and in NoSQL scaling is horizontal.

**Fast queries due to the data model:** since all it is using single document concept the queries

are fast.

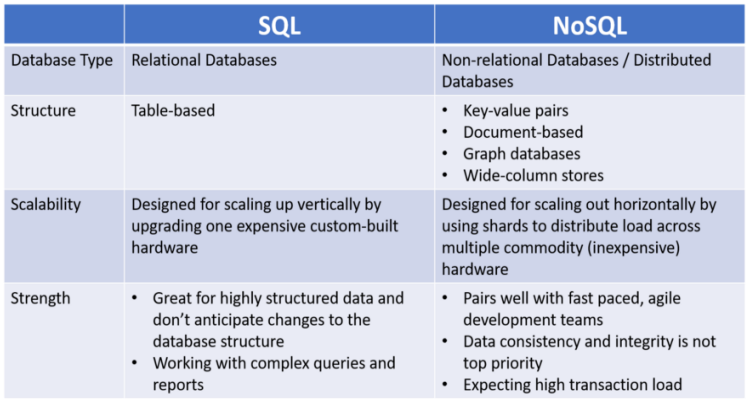
**Types of NoSQL databases**

* Document databases store: in this type document is stored similar to JSON (JavaScript Object Notation). Each document contains multiple fields and values.
* Key-value databases: are a simpler type of database where each item contains keys and

values.

* Wide-column stores: store data in tables, rows, and dynamic columns.
* Graph databases store: data is stored in the form of nodes and edges. Nodes typically store information about people, places, and things, while edges store information about the relationships between the nodes.

**Difference between SQL and NoSQL**

****

**Exercise:**

1. Explain the Relational Database management system in Detail.

2. Explain the NOSQL databases in detail.

3. Explain the Difference between SQL and NoSQL in detail.

4. Write a short note on following points.

a. Create Table

b. Alter Table

c. Drop Table

d. Truncate Table

5. list out and explain the different Constraints from Relational Database Management

system.