S.Q.L.

SQL means Structured Query Language. SQL is a database computer language designed for managing data in relational database management systems (RDBMS).

RDBMS technology is based on the concept of Relational Tables in which data is displayed in Rows and Columns. Following are different types of SQL statements:

SELECT	Data retrieval

INSERT
UPDATE
DELETE
MERGE
	Data manipulation language (DML)

CREATE ALTER DROP RENAME TRUNCATE	Data definition language (DDL)
COMMIT ROLLBACK	Transaction control Language (TCL)
GRANT REVOKE	Data control language (DCL)

BASIC DATATYPES

CHAR	For fixed length character data
VARCHAR2	For variable length character data
NUMBER	For numeric data
DATE	For date/time data
LONG	Variable-length character data up to 2 GB
CLOB	Character data up to 4 GB
RAW BLOB BFILE	Raw binary data
BLOB	Binary data up to 4 GB
BFILE	Binary data stored in an external file

SQL STATEMENTS

CREATE	Create is used to create database objects i.e. Table, View, Procedure, Function, Type, Trigger etc. e.g. create table student (roll number(10), name char(100)); create table games(game varchar2(100),roll number(10));

INSERT	Insert is used to insert Data into a table
	e.g.
	insert into student(roll,name) values(1,'RAJU');
	insert into student(roll,name) values(2,'RAMU');
	insert into games(game,roll) values('BADMINTON',1);

UPDATE	Update is used to modify existing data of a table e.g. update student set name='RITU' where roll=2;

<table>
<tr><td>DELETE</td><td>Delete is used to Delete data from a table
e.g. delete from student where roll=2;</td></tr>
<tr><td>DROP</td><td>Drop is used to Remove database objects i.e. Table, View, Procedure, Function, Type, Trigger etc.
e.g. Drop table student;</td></tr>
<tr><td>ALTER</td><td>Alter is used to modify structure of a database object.
e.g. alter table student add (address varchar2(1000));</td></tr>
<tr><td>COMMIT</td><td>Commit statement is used to Save changes performed in tables.
e.g. commit;</td></tr>
<tr><td>ROLLBACK</td><td>Rollback statement is used to discard all pending data changes i.e Undo changes
e.g. rollback;</td></tr>
<tr><td>GRANT</td><td>Grant is used to give permissions to a User
e.g. Grant select on student to scott;</td></tr>
<tr><td>REVOKE</td><td>Revoke is used to remove permissions from a user.
e.g. Revoke select on student from scott;</td></tr>
<tr><td>SELECT</td><td>Select is used to retrieve data from one or more tables

SELECT * | columns FROM table | view
WHERE condition(s)
ORDER BY columns ASC/DESC;
select * from student;
select roll from student;
select name from student where roll=1;
select distinct name from student order by name desc;
select first_name||’ ‘||last_name as name from employees;

select a.roll, a.name, b.game -- equi join
 from student a, games b
 where a.roll=b.roll;

select a.roll, a.name, b.game --left outer join
 from student a, games b
 where a.roll=b.roll(+);

select a.roll, a.name, b.game --right outer join
 from student a, games b
 where a.roll(+)=b.roll;

select a.roll, a.name, b.game -- full outer foin
 from student a full outer join games b
 on (a.roll=b.roll);

select a.first_name,b.first_name manager --self join
from employees a, employees b
where b.employee_id=a.manager_id

select a.first_name, b.grade --non equi join
from employees a, grades b
where a.salary between b.max_sal and b.min_sal;

select * from students,games; --- cross join</td></tr>
</table>

	select first_name from employees -- to club data from two different tables UNION Select department_name from departments;
COMPARISON CONDITIONS	= Equal to > Greater than >= Greater than or equal to < Less than <= Less than or equal to <> Not equal to BETWEEN ...AND... Between two values (inclusive) IN(set) Match any of a list of values LIKE Match a character pattern IS NULL Is a null value NULL means blank, its not 0 and any operation with it results only NULL

	SELECT * FROM STUDENT WHERE ROLL=5;
	SELECT * FROM STUDENT WHERE ROLL>=5;
	SELECT * FROM STUDENT WHERE ROLL<5;
	SELECT * FROM STUDENT WHERE ROLL<>5;
	SELECT * FROM STUDENT WHERE ROLL BETWEEN 5 AND 10;
	SELECT * FROM STUDENT WHERE ROLL IN (2,5);
	SELECT * FROM STUDENT WHERE NAME LIKE ‘R%’;
	SELECT * FROM STUDENT WHERE NAME IS NULL;

CONVERSION	NVL(column,value)
	NVL is used to replace NULL with a specified value
	DECODE(column,value1,v1,value2,v2,defaultvalue)
	Decode is used to changed the value of a column with another value.
	SELECT NVL(commission_pct,0), first_name from employees;
	SELECT DECODE(commission_pct,null,’nil’,0.5,’half’,’good’) FROM EMP;

UTILTIY FUNCTIONS	SUBSTR: To make a substring of a String SELECT SUBSTR(‘Manash Deb’,7,3) FROM DUAL; --- Deb LENGTH: To know length of a String SELECT LENGTH(‘Manash Deb’) FROM DUAL; --- 10 ROUND: To Round off a Value upto a specified limit SELECT ROUND(15.21545454,2) FROM DUAL; --- 15.22 TRUNC: To Truncate a Value upto a specified limit SELECT TRUNC(15.21545454,2) FROM DUAL; --- 15.21 MOD: To know the reminder of a Division SELECT MOD(5,2) FROM DUAL; --- 1 TO_CHAR(date,format): To convert a date into any format we like SELECT TO_CHAR(sysdate,’DD MON YY DAY HH24:MI:SS’) from dual; TO_DATE(string,format): To convert a String into a date SELECT TO_DATE(‘01012011’,’DDMMYYYY’) from dual; (Here SYSDATE returns present date and time)
LOGICAL CONDITIONS	AND Select * from student where roll>5 and name =’RAMU’; OR Select * from student where roll>5 OR name =’RAMU’; NOT Select * from student where roll>5 AND NAME IS NOT NULL;

SUBQUERY	A subquery is a SELECT statement inside a
	clause of another SQL statement.
	SELECT *columns* FROM *table* WHERE *expr operator*
	 (SELECT *columns* FROM *table*);
	SELECT last_name FROM employees WHERE salary >

	(SELECT salary FROM employees WHERE employee_id = 149) ;

CORRELATED SUBQUERY	Correlated subqueries are used for row-by-row processing. Each subquery is executed once for every row of the outer query.
	Using Correlated Subqueries Each time a row from the outer query is processed, the inner query is evaluated.
	Find all employees who earn more than the average salary in their department.
	SELECT * from employees outer
	WHERE SALARY>
	(SELECT AVG(salary)
	FROM employees
	WHERE department_id =outer.department_id)

GROUP FUNCTIONS	Group functions operate on sets of rows to give one result per group. Few of them are: COUNT, MIN, MAX, AVG, SUM SELECT COUNT(*) FROM employees WHERE department_id=50; Note: Group functions ignore null values

GROUP BY	GROUP BY clause is used to group data as per specific column or columns using group functions.
	SELECT department_id, COUNT(*) FROM employees
	GROUP BY department_id;
	Note: Any column, mentioned in select clause must be there in GROUP BY clause.

HAVING CLAUSE	Having is used to put conditions on the grouped data got by group functions.
	SELECT department_id,COUNT(*) FROM employees
	WHERE salary>3000
	GROUP BY department_id
	HAVING count(*)>2;

DATABASE OBJECTS

TABLE	Table is the basic unit of storage. It is composed of rows and columns.
VIEW	Logical representation of data from one or more tables. It is a virtual table, which does not actually store data. It can be said as a stored select statement. e.g. Create view gamesview as select a.roll, a.name, b.game from student a, games b where a.roll=b.roll;
INDEX	Used to improve performance of queries. Create index student_index on student(roll);
CONSTRAINT	Constraints enforce rules at table level. alter table student add constraint studentrollpk primary key(roll); There are 5 types of Constraints in Oracle: NOT NULL (Data must be there) UNIQUE (No duplicate data) PRIMARY KEY (NOT NULL + UNIQUE) FOREIGN KEY (Data should exist in corresponding Table) CHECK (Defines a logical condition that the values must satisfy)

PL/SQL

PL/SQL is the procedural extension to SQL with design features of programming languages

ANONYMOUS BLOCK	It is used to write a PL/SQL block that need not be stored permanently in the database.
	
	DECLARE (optional)
	  Variable_name datatype/cursor defination;
	BEGIN
	  SQL statements;
	   PL/SQL Statements;
	EXCEPTION (optional)
	   Action to perform in case of errors
	END;
	
	DECLARE
	 v_variable VARCHAR2(5);
	BEGIN
	SELECT column_name INTO v_variable FROM table_name;
	END;

Assignment Operatior	:=
	Variable_name := value;
	V_variable := 6;

Print Output	DBMS_OUTPUT.PUT_LINE

Control Structure	IF Condition THEN statement
	ELSIF Condition THEN statement
	ELSE statement
	END IF;

LOOP	LOOP
	 Statement(s);
	EXIT WHEN condition;
	END LOOP;
	
	DECLARE --- print values from 1 to 100
	 J number(3):=1;
	BEGIN
	 LOOP
	  DBMS_OUTPUT.PUT_LINE(J);
	  J:=j+1;
	  IF J>100 THEN
	    EXIT;
	  END IF;
	   ----EXIT WHEN J>100;
	 END LOOP;
	END;
	/
	BEGIN --- print values from 1 to 100
	 FOR J IN 1..10 LOOP
	  DBMS_OUTPUT.PUT_LINE(J);
	 END LOOP;

	END;

CURSOR	The Cursor can be defined as a pointer to the current tuple. Whenever a query results in a number of tuples, we can use cursor. It consists of Open, fetch and close operation.
	DECLARE
	CURSOR C1 IS SELECT NAME FROM STUDENT;
	VNAME VARCHAR2(100);
	BEGIN
	LOOP
	FETCH C1 INTO VNAME;
	EXIT WHEN C1%NOTFOUND;
	DBMS_OUTPUT.PUT_LINE(VNAME);
	END LOOP;
	END;
	
	DECLARE
	CURSOR C1 IS SELECT NAME FROM STUDENT;
	BEGIN
	FOR I IN C1 LOOP
	DBMS_OUTPUT.PUT_LINE(I.NAME);
	END LOOP;
	END;

PROCEDURE	It's a program to perform an action.
	e.g.
	CREATE OR REPLACE PROCEDURE ABC AS
	BEGIN
	DBMS_OUTPUT.PUT_LINE('HELLO');
	END;
	/
	EXEC ABC;

FUNCTION	It's a program that returns a value
	CREATE OR REPLACE FUNCTION ABCD return CHAR is
	BEGIN
	return 'HELLO';
	END;
	/
	BEGIN
	DBMS_OUTPUT.PUT_LINE(ABCD);
	END;
	/

TRIGGER	It's a PL/SQL block or a PL/SQL procedure associated with a table or view or database. A trigger is activated on the occurrence of a particular event like insert, update, delete.
	CREATE OR REPLACE TRIGGER STUDTRIG
	AFTER INSERT OR UPDATE OR DELETE ON STUDENT
	FOR EACH ROW
	BEGIN
	DBMS_OUTPUT.PUT_LINE('INSERT_UPDATE_DELETE');
	END;
	/
	insert into student(roll,name,address) values
	(2,'RINKU','SAGARPUR');

SUPERKEY

A superkey is a combination of attributes that can be used to uniquely identify a database record. A table might have many superkeys.

CANDIDATE KEY

A candidate key is a special subset of superkeys that do not have any extraneous information in them.

PRIMARY KEY

A primary key is a candidate key which the database designer has chosen to identify a record uniquely.

NON-PRIME ATTRIBUTE

A non-prime attribute is an attribute that does not occur in any candidate key.

ALTERNATE KEY

In the context of relational databases, an alternate key (or secondary key) is any candidate key which is not selected to be the primary key.

Examples:

Student (Name, Age, Roll, Email, Address)

This table has many possible superkeys. Three of these are (Roll, Name), (Roll,Name,Email), (Name,Email), (Roll), (Email) etc. Of these, only (Roll) and (Email) are candidate key, as the others contain information not necessary to uniquely identify records. Name,Age,Address in the above table is a non-prime attribute. Roll in the above table is choosen as primary key. So, Email is the alternate key.

ER DIAGRAM

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes.

THE NORMAL FORMS (NF):

The normal forms (abbrev. NF) of relational database theory provide criteria for determining a table's degree of vulnerability to logical inconsistencies and anomalies.

The main normal forms are summarized below. (*MCS-43 syllabus starts from 4 NF*)

Normal form	Brief definition
First normal form (1NF)	There are no duplicated rows in the table. All underlying domains contain atomic values only
Second normal form (2NF)	There should not be any Partial Dependencies in the Relations
Third normal form (3NF)	There should not be any Transitive Dependencies in the Relations
Boyce–Codd normal form (BCNF)	All Determinants must be a Key. All Keys should be well-defined.
Fourth normal form (4NF)	Every non-trivial multivalued dependency in the table is a dependency on a superkey
Fifth normal form (5NF)	Every non-trivial join dependency in the table is implied by the superkeys of the table
Domain/key normal form (DKNF)	Every constraint on the table is a logical consequence of the table's domain constraints and key constraints

FUNCTIONAL DEPENDENCY:

A functional dependency is denoted by X → Y, between two sets of attributes X and Y. X→Y means that the value of Y component is uniquely determined by the value of X component. This is functional dependency from X to Y (but not Y to X)

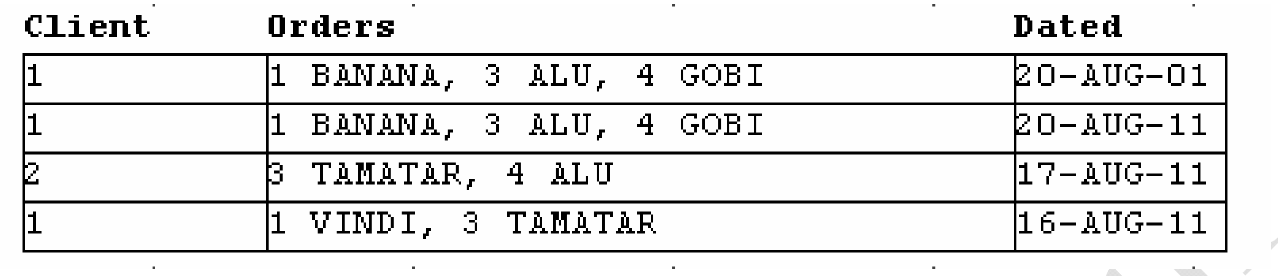
X → Y implies that for every unique value of X, value of Y is same. Also, all non keys are functionally dependent on candidate keys.

Example:
STUDENT (enrolno, sname, cname, classlocation, hours)

In the above relation, the following F.D. exists:

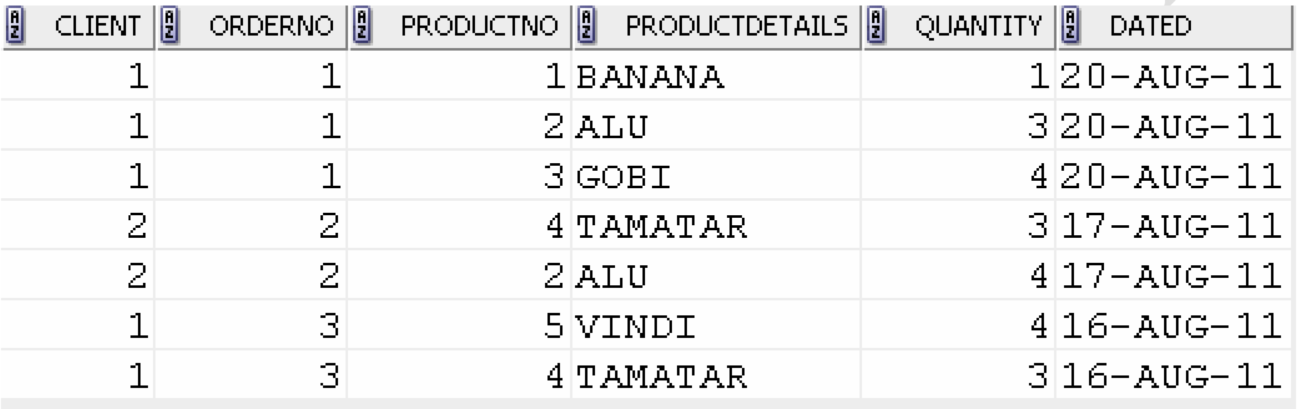
enrolno →sname (the enrolment number of a student uniquely determines the student names, so sname is functionally dependent on enrolment number).

Unnormalized Table: The relation below in not normalized as the values are not atomic.



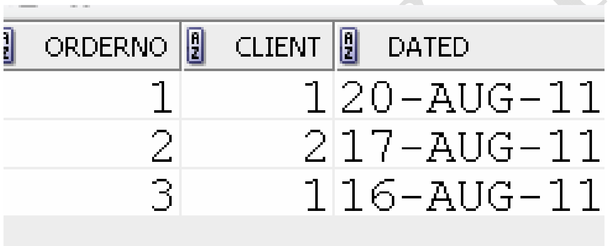
Client	Orders	Dated
1	1 BANANA, 3 ALU, 4 GOBI	20-AUG-01
1	1 BANANA, 3 ALU, 4 GOBI	20-AUG-11
2	3 TAMATAR, 4 ALU	17-AUG-11
1	1 VINDI, 3 TAMATAR	16-AUG-11

1NF: The table below is in 1NF, as there are no duplicated rows and columns are atomic.

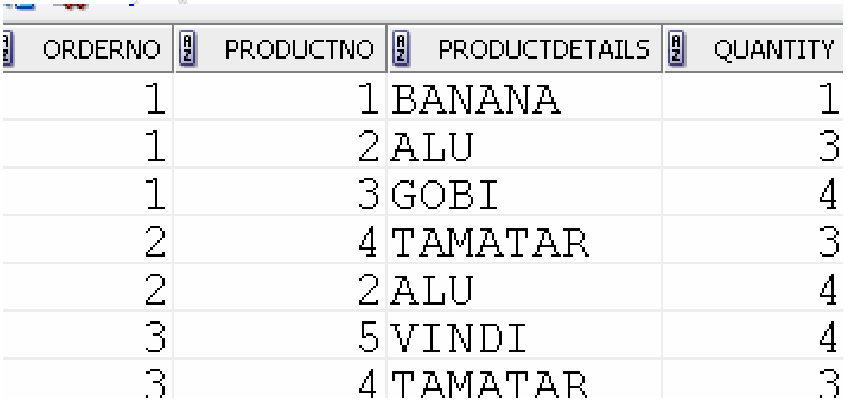


CLIENT	ORDERNO	PRODUCTNO	PRODUCTDETAILS	QUANTITY	DATED
1	1	1	BANANA	1	20-AUG-11
1	1	2	ALU	3	20-AUG-11
1	1	3	GOBI	4	20-AUG-11
2	2	4	TAMATAR	3	17-AUG-11
2	2	2	ALU	4	17-AUG-11
1	3	5	VINDI	4	16-AUG-11
1	3	4	TAMATAR	3	16-AUG-11

2NF: Partial dependency means dependency of certain attributes to a subset of candidate key. In the above Structure, the candidate key is (client, orderno, productno). But there is a dependency (orderno →client). As client is not fully dependent on the candidate key, we decompose it to remove this partial dependency. Again there is another partial dependency (orderno →dated). So we also remove dated table and put it in another table.

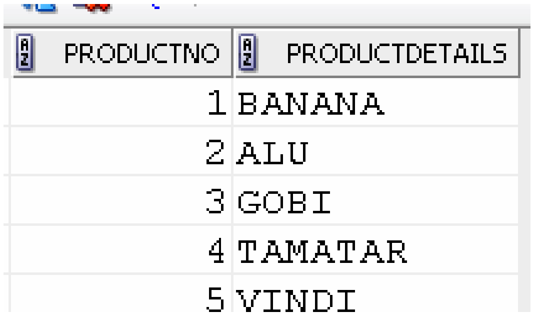


ORDERNO	CLIENT	DATED
1	1	20-AUG-11
2	2	17-AUG-11
3	1	16-AUG-11

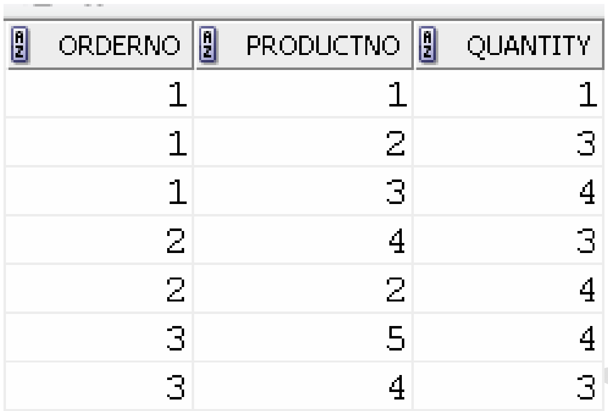


ORDERNO	PRODUCTNO	PRODUCTDETAILS	QUANTITY
1	1	BANANA	1
1	2	ALU	3
1	3	GOBI	4
2	4	TAMATAR	3
2	2	ALU	4
3	5	VINDI	4
3	4	TAMATAR	3

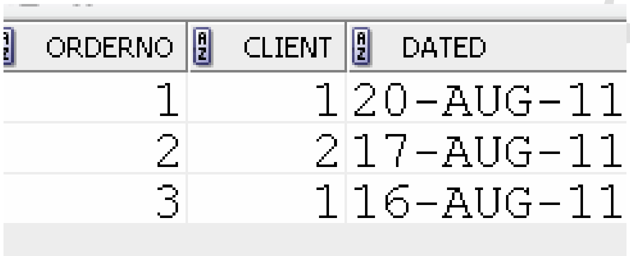
3NF: In the above table,candidate key is (orderno,productno). But there exists an F.D. (productno →productdetails). Productno is not the candidate key of the table. So the column productdetails is partially dependent on candidate key. To remove this partial dependency, we decompose the structure as follows:



PRODUCTNO	PRODUCTDETAILS
1	BANANA
2	ALU
3	GOBI
4	TAMATAR
5	VINDI



ORDERNO	PRODUCTNO	QUANTITY
1	1	1
1	2	3
1	3	4
2	4	3
2	2	4
3	5	4
3	4	3



ORDERNO	CLIENT	DATED
1	1	20-AUG-11
2	2	17-AUG-11
3	1	16-AUG-11

BCNF: Mostly all tables that are in 3NF are also in BCNF. For BCNF, All determinants should be candidate key. Determinant means, the key which determines an attribute i.e. left hand side of a functional dependency. In above relations, we have already made all determinants candidate key.

To make a relation in BCNF, we should define all primary key and foreign keys of the table properly.

MULTI VALUED DEPENDENCY

Employee	Project	Language
Ajay	A	C
Ajay	A	Java
Ajay	A	C++
Ajay	B	C
Ajay	B	Java
Ajay	B	C++
Vijay	A	C
Vijay	A	Java
Vijay	B	C
Vijay	B	Java
Sujay	A	C

The multi-valued dependency X → → Y is said to hold for a relation R(X, Y, Z) if, for a given set of value for attribute X, there is a set of associated values for the set of attributes Y and the Y values depend only on X values and have no dependence on the set of attributes Z. Whenever X → → Y holds, so does X → → Z.
In the above example, employee → → project and employee → → language holds.

Functinal Dependency (FD) is a special case of MVD, where every X determines exactly one Y, never more than one. So, all FDs are MVDs, but not all MVDs are FDs.

Trivial MVD: An MVD X → → Y is called trivial, if one of the following is true:

(i) Y is a subset of X

(ii) XUY are all attributes.

Non-Trivial MVD: An MVD X → → Y is called trivial, if Y is not a subset of X and XUY are not all attributes.

4NF DEFINITION:
Every non-trivial multi-valued dependency in the table is a dependency on a super-key.

For a relation R(X,Y,Z) having non trivial MVDs X→→Y and X→→Z, We decompose the relation into two trivial MVDs, R_1 (X,Y) and R_2 (X,Z) to get it into 4NF.

To get the above example into 4NF, we decompose the relation into:
R1(Employee,Project) and R2(Employee,Language).

Employee	Project
Ajay	A
Ajay	B
Vijay	A
Vijay	B
Sujay	A

Employee	Language
Ajay	C
Ajay	Java
Ajay	C++
Vijay	C
Vijay	Java
Sujay	C

JOIN DEPENDENCY:

Employee	Project	Language
Ajay	A	C
Ajay	B	Java
Ajay	B	C++
Vijay	A	C
Vijay	B	Java
Sujay	A	C

In the above Relation, there are three attributes Employee, Project and Language.
(i) Ajay can do Project A,B; Vijay can do Project A,B and Sujay can do Project A.
(ii) Ajay knows C, Java, C++; Vijay knows C, Java and Sujay knows C language.
(iii) Project A can be done only in C language and Project B only in Java and C++.

If we decompose the above relation into only two parts, it will not be able to represent the complete information. So we need a third relation to represent the same:

Employee	Project
Ajay	A
Ajay	B
Vijay	A
Vijay	B
Sujay	A

Employee	Language
Ajay	C
Ajay	Java
Ajay	C++
Vijay	C
Vijay	Java
Sujay	C

Project	Language
A	C
B	Java
B	C++

A Join dependency is generalization of Multi-valued dependency. A relation R satisfies join dependency $*(R_1, R_2, ..., R_n)$ if and only if R is equal to the join of $R_1, R_2, ..., R_n$ where R_i are subsets of the set of attributes of R. A table *T* is subject to a join dependency if *T* can always be recreated by joining multiple tables each having a subset of the attributes of *T*.

A join dependency is called trivial if one of R_i is R. A join dependency $*(R_1, R_2)$ is equivalent to MVD $R_1 \cap R_2 \rightarrow\rightarrow R_2$. So every JD is also in MVD.

The join dependency in the above example would be:
*((employee, project), (employee, language), (project, language))

FIFTH NORMAL FORM (PROJECT JOIN NORMAL FORM)

A relation R is in 5NF or PJNF if for all join dependencies at least one of the following holds:

(a) $*(R_1, R_2, ..., R_n)$ is a trivial join-dependency
(b) Every R_i is a candidate key for R.

Database Catalogue and Data Dictionary:

A Database Catalogue provides the information mainly accessed by the various software modules of the DBMS, such as DDL and DML compilers, the query optimiser, the transaction processor, report generators, and the constraint enforcer.

A Data Dictionary is a data structure that stores meta-data, i.e., data about data. It is mainly used by the designers, users, and administrator for information on system hardware and software configurations, documentation and other information relevant to system administration.

Data dictionaries may be divided into three categories: **USER, ALL, and DBA.**

e.g. to view all objects of the user,

SELECT object_name, object_type FROM USER_OBJECTS;

To view all objects of the database to which the user has access,

SELECT object_name, object_type FROM ALL_OBJECTS;

If the user is an administrator (DBA), he can check all objects of all users by,

SELECT object_name, object_type FROM SYS.DBA_OBJECTS;

TRANSACTION

A transaction may be defined as a collection of operations on the database that performs a single logical function in a database application or it should be an inseparable list of database operations. Transaction has certain characteristics. These characteristics are known as the ACID properties. i.e. ATOMICITY, CONSISTENCY, ISOLATION and DURABILITY.

ATOMICITY: Atomicity means perform all steps or no steps.

CONSISTENCY: Consistency means if the transaction violates the databases consistency rules, then the entire transaction will be rolled back.

ISOLATION: Isolation means that separate transactions running at the same time on same data should be handled properly.

DURABILITY: Durability means that the changes made to the system are permanent.

On-line Transaction Processing (OLTP)

(Computerized processing in which each transaction is processed immediately and the affected records are updated)

Software Model: Waterfall Model

Steps of Waterfall Model:
1) Requirement Analysis
2) Design
3) Coding
4) Testing
5) Maintenance

Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system.

Context Free Diagram:

Zero Level DFD is known as Context Free Diagram. The whole system is denoted by a Oval Box in it and all input/outputs of the system are shown.

Testing

Testing is a process of executing a program with the intent of finding an error.

Testing Types:

a) Unit Testing: To test each unit i.e. modules of the Software
b) Integration Testing: To test integration between various modules.
c) System Testing: To test the overall system

White Box Testing

White box testing is performed to reveal problems with the internal structure of a program.

Black Box Testing

Black box tests are performed to assess how well a program meets its requirements, looking for missing or incorrect functionally

Software Security

Security of a system refers to protection of the system as well as security of data that is maintained through application stored within it.

Cost Calculation through COCOMO:

COCOMO model stands for Constructive Cost Model. It is the best known and most thoroughly documented of all the software cost estimation models. It also provides three levels of the models: Basic, Intermediate and Detailed.

In COCOMO model, the development effort assumes the following form-

E=aSbm

Where

E= Effort

S=Value of Source in LOC

M=Multiplier that is determined from a set of 15 Cost driver's attributes

The following are examples of the above cost drivers

- Size of the applicatio0n database
- Complexity of the project
- Reliability requirements for the software
- Performance constraints in run time
- Capability of the software
- Scheduling constraints