Relational Calculus

Tuple and domain calculi are collectively referred to as relational calculus. In relational calculus, a query is expressed as a formula consisting of a number of variables and an expression involving these variables. The formula describes the properties of the result relation to be obtained. In relational tuple calculus variables represent the tuples from specified relation. In relational domain calculus the variables represent values drawn from specified domains.

Relational calculus, which in fact means calculating with relations, is based on predicate calculus, which is calculating with predicates.

Propositions that specifies a properties consist of an expression that names an individual object and an expression, called the predicate that stands for the property that the individual object possesses.

A proposition is a declarative (or assertive) sentence, e.g. “It is raining”.

A convenient method of writing the statement is to place the predicate first and follows it with the object enclosed in parentheses. Therefore the statement “ABC is a company” can be written as “is a company (ABC)”.

Now we can drop the “is a” part and write the first statement as “company (ABC)”.

Finally if we use symbols for both the predicate and the object we can use symbols for both the predicate and the object we can rewrite the statement as P(x). The lower case letter from end of the alphabet (……x,y,z) denotes variables the beginning letters (a,b,c……)

denotes constraints, and upper case letters denotes predicate.

In P(x) where x is the argument is a one-place or monadic predicate DBMS(x) and COMPANY(y) are examples of monadic predicates the variable x and y are replaceable by constants or name of the individual objects such as DMBS(ISS).

We have a predicate of degree n, where the predicate takes n arguments. For example bigger_than(WXY,BCD).

A predicate followed by its arguments is called an atomic formula. Example DBMS(x), COMPANY(y) etc.

Predicate calculus is a formal language which consists of symbols. We have already seen the symbols i.e. variable, constants and predicates. WE can also specify logical connectors such as “not” or “negation” denoted by ¬, “or” denoted by (∨), “and” denoted by (∧), “implication” denoted by (→)

Other interesting formulas are formed with the use of quantifiers: universal or “for all” denoted by ∀ and existential or “for some” denoted by ∃.

The notions expressed by the quantifiers asserts that “everything has a certain property” and that “something has certain property”. Therefore, (∀x) P(x)and (∃x) p(x) are used to specify that “for all x, x is P” (or simply that “everything is P”) and “for some x, x is P” (or simply that “something is P”).

Ex:- (∃x) DBMS(x) is a formula that states that there is something that is a DBMS. We can also say that there exist something that possesses the property of a DBMS.

Structured Query Language:

SQL is an acronym for Structured Query Language. It is available in a number of database management packages based on the relational model of data, for example in DB2 or the IBM and UNIFY of the UNIFY corporation.

SQL is originally defined by D.D. Chamberlain in 1974. SQL underwent a number of modifications over the years, Today SQL has become an official ANSI standard.

SQL commands can be roughly classified into three major categories with regard to their functionality.

1. DDL(Data definition Language)→ Those commands that create and maintain the database are grouped into the class called Data Definition Language(DDL).

Data definition in SQL is via the create statement. The statement can be used to create a table, index or view. To create a table the create statement specifies the name of the table and the names and data types of each column of the table. Its format is:

Create table <Table_name> (<attribute list>)

Where the attribute list is specified as

<attribute list> :: <attribute name> (<data type>)
[not null][,<attribute list>][,……]

The data type supported by SQL depends on the particular implementation. However following data type are generally included : integer, decimal, real(i.e.

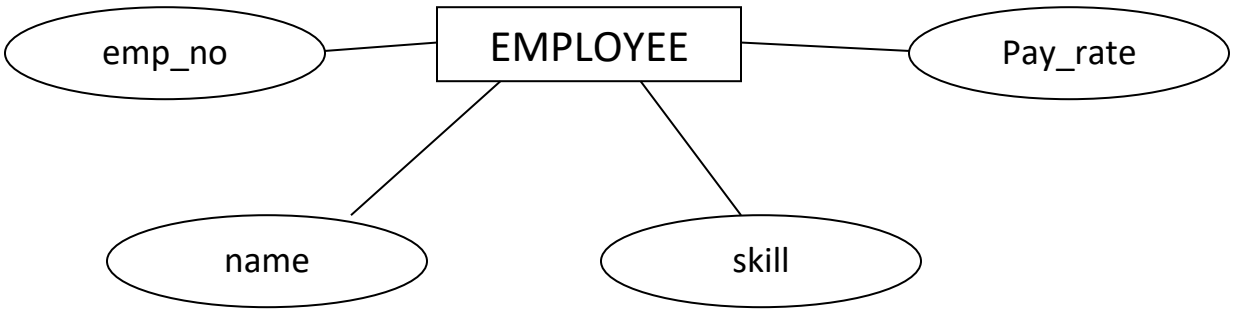
floating point values) and character string, both of fixed and varying length.

```
    <datatype>::=  <integer>|<smallint>|
                   <char(n)>|<varchar(n)>|
                   <float>|<decimal(p[q])>
```

Ex:-

```
        Create table EMPLOYEE
        (
            emp_no integer not null,
            name    char(25),
            skill   char(20),
            pay_rate    decimal(10,2)
        );
```

E-R Diagram of above table



Oracle has provided the following data type:

Data Type	Description
char(size)	char data Size is specified in number For example char(10)
varchar2(size)	Same as above (recommended for less storage)
date	Used for specifying date
long	Used to specify numeric data

	NUMBER(SIZE)
	NUMBER(SIZE,DEC) Number with digits equal to specified size.
DEC	Indicates decimal digits
FLOAT	Same as number
INTEGER	Same as number but without decimal
SMALLINT	Same as Integer

The definition of existing relation can be altered using the 'alter' statement. This statement allows a new column to be added to an existing relation the existing tuples of the altered relation are logically considered to be assigned the NULL value for the added column. The syntax of the 'alter' statement and an example showing the attribute phone_number added to the EMPLOYEE relation is as follows:

::➔

(1) **Adding**

Syntax:

```
alter table existing_table_name add column_name data_type[,……..];
```

Ex:-

```
alter table EMPLOYEE add phone_number decimal(10);
```

(2) **Modifying table in a column**

Ex:- **alter table EMPLOYEE modify name char(30);**

(3) **Dropping column in a table**

Ex:- **alter table EMPLOYEE drop column skill;**

(4) **Rename column in a table**

Ex:- alter table EMPLOYEE rename name to empname;

Data Manipulation Language(DML):- Data manipulation language is used for modifying the data in a table. Modification can be made by using insertion, updating or deletion operations.

Inserting data in Table:- A row is inserted into a table using 'insert' statement.

Syntax:-

(1) INSERT INTO TABLE VALUES (value)

Ex:-

INSERT INTO EMPLOYEE VALUES (101, 'John', 50000.00, 9824563211)

Note:- In this we cannot change the order in which data is inserted, so we have to be careful while writing the above insert statement.

(2) INSERT INTO EMPLOYEE (Emp_no, emp_name, phone_number) values (102, 'Burner', 3215629321);

Note:- In this method values are inserted according to the sequence in which they are given.

If a user want to enter more records then he will need to write the insert statement again and again. We can use a substitution variable to insert a large number records.

Ex:-

```
insert into EMPLOYEE values (&empl_no, &empname, &pay_rate, &phone_number);
```

Now we can insert another records in the same table just by typing 'run' at the SQL prompt.

Ex:-

SQL>run<Enter>
SQL>/<Enter>

The run command is used for executing previous commands.