Introducing basic query flows

Database is a collection of interrelated data. Users have to work to insert database, it will take some time, but the major benefit of using database is ease of retrieving data. Database Management System has its own language known as Structure Query Language (SQL) which enables users to create database, relations (tables), manipulate tables, insert new records, update existing records, remove records and retrieve records.

The major advantage of database is ease in retrieving data. SQL select statement is created to retrieve data. SQL Select statement consists of

- Relation/s from which data is to be retrieved
- condition on the dataset
- list of required attributes from the data set
- ordering of the retrieved dataset

In relational database, work is performed in relations and result is also relation. It is always interesting to talk about the execution of the query (basically select statement). But there is not any specific way how the query flows, it is up to the query optimizer to decide how to execute query.

Here, we can make some assumptions about the basic query flow or execution. It is only assumption:

- first dataset is formed from the relation/s specified in the from clause
- second if there is condition, then specified condition is applied to the formed data set
- if there is group by and having clauses then these are worked
- then ordering of the data is done if specified
- Then specified attributes in the select clause are returned as new dataset.

These steps seem realistic but it is query optimizer and algorithm used which decide how to execute the query. Query optimizer will create plan to execute query. If the query is simple like “select name from member”, in this case optimizer will not spend too much time to create plan. If a query requires join then different plans to execute same query will be created and cost-based evaluation of each created plan is done. Cost here means amount of memory, CPU, time etc.

Actually, query flow or execution is a very complicated task. So here, we will focus on procedural query language, it works on one or more relations as input and produces new relation as output. This procedural language will help to understand how SQL statement works. At the same time also try to understand how equivalent SQL select statement can be written for relational algebra statements.

Relational Algebra (R. A.)

Relational algebra is a procedural query language. In procedural query language user instructs a sequence of operations to be performed on the database to compute required result. It consists of six fundamental operators and from those six operators other operators are formed. The six fundamental operators are:

- **Select (σ)**: It is unary operator that works on single relation and returns rows from relation which satisfies given conditions
- **Project (π)**: This is also unary operator that works on single relation and returns specified columns from the relation.
- **Cartesian Product (X)**: This is binary operator and works on two relations and return the cartesian product of the given relations.
- **Rename (ρ)**: This is unary operator used to give new name for relations.
- **Set difference (-)**: This is binary operator and works on two relations and returns difference between two relations
- **Set Union: (U)**: This is binary operator and works on two relations are returns union of given relations.

SQL SELECT Statement

SQL Select statement is used to retrieve records from relations (tables). The basic syntax of SQL select statement is as follows:

SELECT attributes list FROM relations list WHERE condition [GROUP BY attribute/s ORDER BY ..]

Here expression given in [] is optional. The select part of the SELECT statement takes list of attribute/s of which values are required; FROM part takes list of relation/s, that is the datasets from where data are to be retrieved, if more than one relation is mentioned then those relations are merged to form single dataset; WHERE part takes condition which records must satisfy, the mentioned condition is checked on the relation or merged relations mentioned in the FROM part.

We can see equivalence between the SQL Select statement and operators of the relational algebra. Now in coming section we will see how SQL select statements equivalent relational algebra query can be written.

Equivalence between SQL select statement and Relational Algebra (R. A.) queries.

To explain RA and equivalence with SQL, we will consider following relations:

- Member (memberid, name, contactno, gender)
- Account (accountno, type, createddate, memberid) Foreign Key memberid references Member

SELECT (σ):

This select operator of the R. A. is used to specify conditions which the records must satisfy. Let's consider following queries:

a> Retrieve all the members who are male

R.A. → $\sigma_{gender='m'}$(member)

SQL → Select * from member where gender = 'm'

Here condition is applied as subscript of select operator in R.A. and condition is applied in where clause in SQL select.

b> Retrieve member whose is female and contact number is '9843631516'

R.A. → $\sigma_{gender='f'\wedge contactno = '9843631516'}(member)$

SQL → Select * from member where gender = 'f' and contactno= '9843631516'

Here, two conditions are specified, one is for gender and other for contactno. In R. A. symbol **^** specifies and; and in SQL **and** is used to specify and. If and operator is used then both conditions must be true to return records.

PROJECT (η)

This operator is used to retrieve required columns from given relations.

a> Retrieve name of all members.

R.A. → $\eta_{name}(member)$

SQL → Select name from member
Here, attribute is written as subscript of project operator in R. A. and in SQL select, attribute name is written after select word. Both with return relation with only one column, that is name column.

b> Retrieve name and gender of all members.

R.A. → $\eta_{name, gender}(member)$

SQL → Select name, gender from member
Here attributes are separated by comma in both type of queries

Operators can be linked

Relational algebra operators can be linked. That is select and project operators can be written in a single query.

a> Retrieve name of members whose is female and contact number is '9843631516'

R.A. → $\eta_{name}(\sigma_{gender='f'\wedge contactno = '9843631516'}(member))$

SQL → Select name from member where gender = 'f' and contactno= '9843631516'

Here, in R. A., work of select operator is performed first then project operator is applied on the result provided by select. In SQL, Select part does the work of R. A. project operator and Where part does the work of R. A. Select operator.

Cartesian Product (X)

This operator merges two mentioned relations. It is similar to the cartesian product with the sets. To illustrate R. A. Cartesian product let's consider following two relations with few records:

Employee Post

Name	Age	Gender
Rabin	22	M
Keeran	24	F
Suresh	25	M

Post	Salary
Manager	30000
Sen. Assistant	24000
Jun. Assistant	20000

The Cartesian product between given two relations employee and post will be as follows:

Name	**Age**	**Gender**	**Post**	**Salary**
Rabin	22	M	Manager	30000
Rabin	22	M	Sen. Assistant	24000
Rabin	22	M	Jun. Assistant	20000
Keeran	24	F	Manager	30000
Keeran	24	F	Sen. Assistant	24000
Keeran	24	F	Jun. Assistant	20000
Suresh	25	M	Manager	30000
Suresh	25	M	Sen. Assistant	24000
Suresh	25	M	Jun. Assistant	20000

The first record of employee is merged with all the three records of post, then second record of employee is merged with all the three records of post and third record of employee is merged with all the three records of post. In this way there are 9 records (3 X 3) and 5 columns (3 + 2).

So, when cartesian product between two relations is performed then

- ➔ Number of columns in the resulting relation will be sum of columns of given two relations. There are three columns in employee and two three columns in post. So, 3 + 2 = 5 is the number of columns in the resulting relation.
- ➔ Number of records in the resulting relation is product of number of records of the given two relations. There are three records in employee and three records in post, so, 3 X 3 = 9 is the number of records in the product relation.

In SQL, if select two or more relations are specified in the FROM part then cartesian product of the mentioned relations is formed.

R. A. → employee X post

SQL → Select * from employee, post;

Rename (ρ)

This operator is used to given new name to the existing relation

ρ_r (member)

here, member relation will be known by r also. This operator will returns the result of expression under given name also.

$\rho_r(\sigma_{gender='f' \wedge contactno = '9843631516'}(member))$

SQL → (Select * from member where gender ='f' and contactno = '9843631516') as r

here, the result returned by select operator will be known as r in both R. A. and SQL select.

Set difference (-)

This operator returns the difference between given relations. It helps to get records of one relation which are not in another relation. The mentioned relations must be compatible with each other. That is the number of attributes (columns) in both relations must match and data type of i^{th} column of first relation must be same as the data type of i^{th} column of the second relation. This is known as **union compatibility rule**. Let's take following table of customers

Customer relation

Name	Age	Gender
Dev	34	M
Anju	34	F
Suresh	25	M

If we take set difference of Employee table mentioned above and this customer table:

R. A. →Employee – customer

SQL → select * from employee except select * from customer

the resulting table will be

Name	Age	Gender
Rabin	22	M
Keeran	24	F

Union Operator (U)

Union operator returns records from two tables, if there is duplicate records only one record is returned without repetition of the same record. To take union of two relations they must be union compatible. If we took union of Employee relation and customer relation following will be the resulting relation.

R. A. → Employee U Customer

SQL → Select * from employee union Select * from customer

Name	Age	Gender
Rabin	22	M
Keeran	24	F
Dev	34	M
Anju	34	F
Suresh	25	M

Here record of Suresh appeared in both employee and customer so only one record is considered in case of R. A. but in SQL both records of Suresh will be displayed.

Additional relation Algebra Operators.

Natural Join (⋈):

Natural Join operator combines cartesian product and select operators of R. A. Select operator is applied to the cartesian product of given relations, and select operator will match the values of common attributes in the product table. Let's reconsider the following relations:

- Member (memberid, name, contactno, gender)
- Account (accountno, type, createddate, memberid) Foreign Key memberid references Member

a> Retrieve name of all members who have account.

R. A. → η_{name} ($\sigma_{member.memberid = account.memberid}$(Member X Account))
OR
R. A. → η_{name} (Member ⋈ Account)
SQL → Select name from member, account where member.memberid = account.memberid

Creating queries

Let's consider following scenario. We will see this scenario in different chapters also.

There are eight teams participating in a tournament. Each team has eleven players, and one of the players of the team is a caption of that team. Each team has a coach. A game is played between two teams and one of the team wins the game. Player playing faults are penalized. Teams playing unfair also have to pay penalty. The total penalty a team has to pay is the sum of penalty it has to pay and sum of penalties its players have to pay.
From above given Scenario, we can find following entities and relationships between the entities.

- Team (teamname, estd, address) [*Note estd = established date*]
- Player (playerno, name, address)
- Coach (coachno, coachname, address)
- Teampenality (gameno, amount)
- Playerpenalty (teamname, gameno, amount)
- Plays (playerno, teamname)
- Coachedby (teamname, coachno)
- Caption (teamname, playerno, address)
- Game (gameno, teamname1, teamname2, date, winner)

Here in given scenario, we can see that player and team are related and type of relationship is one-to-many from player to team. So, it is not necessary to create table for relationship, as making primary key of team as foreign key in player is sufficient to show who is playing from which team. Same applies between coach and team as relationship is one-to-many from coach to team.

Introducing basic query flows
Dinesh

If intermediate table for relationship is created then there may require joining three tables for certain queries. But if intermediate table is not created only join between two tables. The design of the database may also affect the way query is written.
Let's consider following query list the name of all player playing for team 'Sankata'.

SQL → select name from player where teamname = 'Sankata';

R. A. → $\eta_{name}(\sigma_{teamname= \text{'Sankata'}}(player))$

Let's consider another example:

- Name of caption and coach of the winning team of game with gameno = 3;

  R4 ← Select coachname from Coach where teamname = (select winner from R1);
  R3 ← Select name from player where playerno = (Select plaerno from R2);
  R2 ← Select playerno from caption where teamname = (select winner from R1);
  R1 ← Select winner from game where gameno = 3;

  Here to answer given query is extracted using simple queries and sub queries, Relations R3 and R4 will give answer to mentioned query.

SQL33→Select c.coachname, p.playername from player as p, coach as c, game as g, caption as cp where player.teamname = g.winner and c.teamname = g.winner and cp.playerno = p.playerno and cp.teamname = cp.teamname and g.gameno = 3.

This is bit complicated query. The execution of this SQL33 and simple queries (from R1 to R4) will be quite different.

1. Show correspondence between SQL select statement and RA operator.
2. How is join performed? Explain with example.
3. Write SQL statement to perform difference between two relations.
4. How will new records inserted in relations using RA operator?
5. How update of records can be performed using RA operator?
6. Explain a query which can be written in different order.