Chapter 04: Relational Algebra

1. Introduction

The first language covered in this document is ***relational algebra****. It consists of a set of operations used to manipulate relations, which are considered as sets of tuples.*

In relational algebra, all operations manipulate one or two relations, and the final result is also a relation.

Generally, there are two types of operations:

✍ ***Set operations****:*

- *Union*
- *Intersection*
- *Difference*

✍ ***Specific operations****:*

- *Projection*
- *Selection*
- *Join*
- *Division*

Set Operations

2.1 Union

The union of two relations R1 and R2 with the same schema is a relation R3 with an identical schema that contains the tuples present in R1, R2, or both.

Conditions for Union:

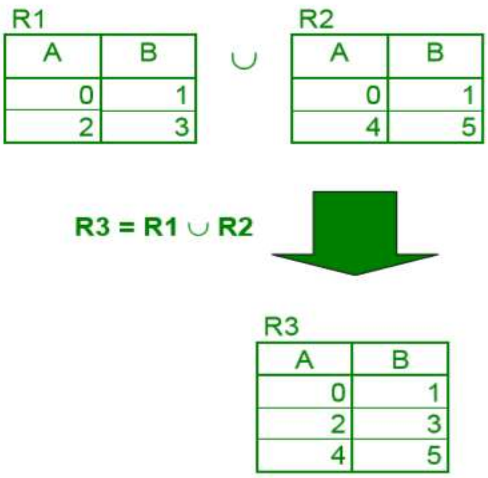
✓ ***Same schema****:*

- *The attributes (columns) in **R1** and **R2** must have the same number, order, and domain (data type).*

✓ ***No duplicates**:*

- *If a tuple exists in both **R1** and **R2**, it appears only once in the result.*

***Example 1**:*



Syntax : *Let R1 ,R2 two relation with the same schema, the union is noted:*

$$R1 \cup R2$$

Example 2:

Relations:

***R1** (Students in Class A):*

Name	***Age***
Alice	*20*
Bob	*22*

***R2** (Students in Class B):*

Name	***Age***
Charlie	*21*

Name	*Age*
Alice	*20*

Union

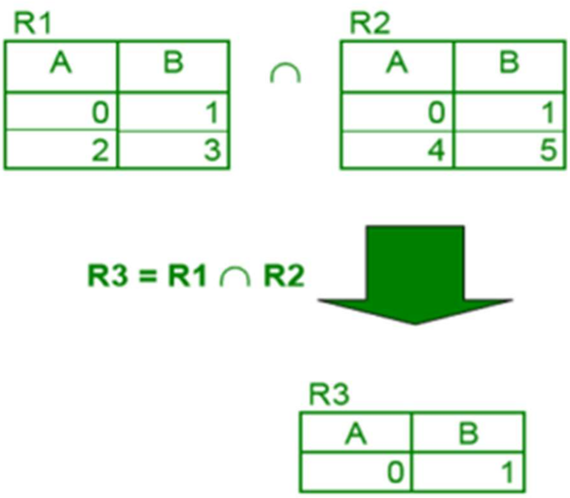
R1 ∪ R2

***Result**:*

Name	*Age*
Alice	*20*
Bob	*22*
Charlie	*21*

2.2 Intersection:

The intersection of two relations R1 and R2 with the same schema is a relation R3 with an identical schema, containing the tuples that are common to both R1 and R2.



Syntax : *Let R1 ,R2 two relation with the same schema, the intersection is noted:*

$$R1 \cap R2$$

Example 04:

Relations:

***R1** (Students enrolled in Math):*

Name	*Age*
Alice	*20*
Bob	*22*

R2 *(Students enrolled in Physics):*

Name	*Age*
Alice	*20*
Charlie	*21*

Intersection

R1 ∩ R2

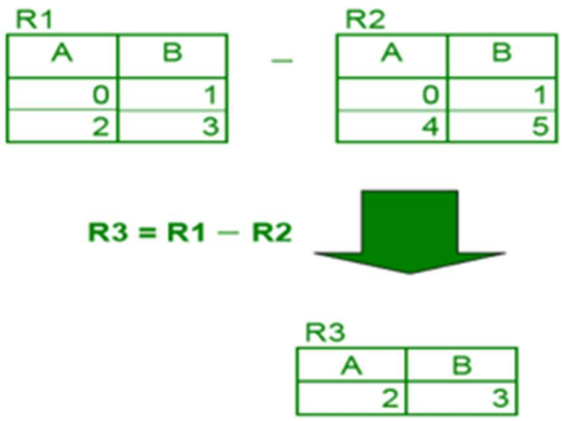
Result:

Name	*Age*
Alice	*20*

2.3 Difference:

The difference between two relations R1 and R2 with the same schema is a relation R3 with an identical schema, containing the tuples from R1 that do not belong to R2.

Example 05:



Syntax: *Let R1 ,R2 two relation with the same schema, the difference is noted:*

$$R1 - R2$$

Example 06

Relations:

R1 *(Students enrolled in Math):*

Name	***Age***
Alice	*20*
Bob	*22*
Charlie	*21*

R2 *(Students enrolled in Physics):*

Name	***Age***
Alice	*20*
Charlie	*21*

Difference operation
R1 − R2

Result:

The result is a relation R3, containing the tuples from R1 that are ***not*** *in R2.*

R3 *(Students in Math but not in Physics):*

Name	***Age***
Bob	*22*

- ✓ ***Specific Operations:***

3.1 Projection:

✍ *The projection of a relation R1 is the relation R2 obtained by removing the attributes of R1 that are not mentioned and then eliminating any duplicate tuples if necessary.*

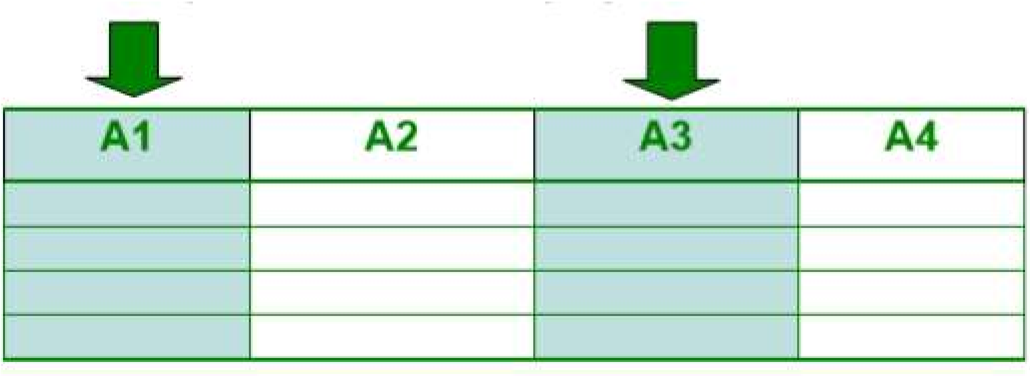
syntax : *πA1, A2, ..., An (R1)*

✍ *The projection of a relation R1 is performed on the attributes A1, A2, ..., An.*

✍ *Projection corresponds to a* ***vertical slicing****.*

Example 07:

π A1, A3 (R)



Example 08:

Let Product Table Example:

N°Product	Name	Price	QStock
01	car hyundai eon	6000 euro	10
02	Motocycle MBK	10000 euro	20
03	bicycle Detwin	300 euro	30

For each product give the quantity in stock:

π Name, Qstock (Product)

Result:

Name	QStock
car hyundai eon	10
Motocycle MBK	20
bicycle Detwin	30

3.2 Restriction:

✍ *The restriction of a relation R1 is a relation R2 with the same schema, containing only the tuples from R1 that satisfy the specified condition.*

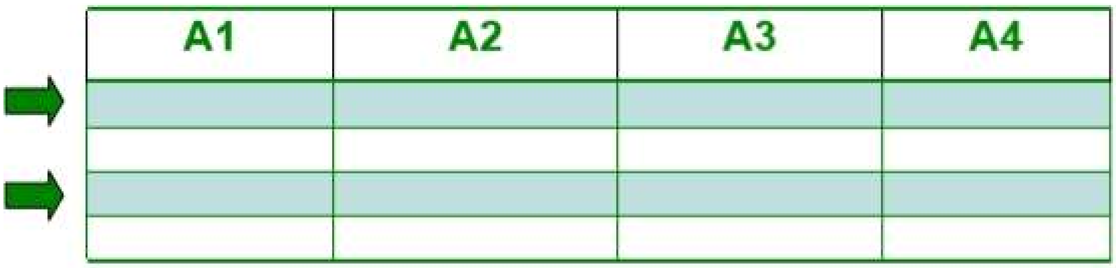
syntax: *σcondition(R1).*

✍ *The restriction of a relation R1 based on the "condition" criterion, where "condition" is a comparison relation between two attributes or between an attribute and a value.*

✍ *The restriction operation allows the extraction of tuples that satisfy a condition.*

✍ *The restriction corresponds to a* ***horizontal slicing*** *of the table that describes the relation.*

Example 09:



Example 10: *give all product where the quantity in stock superior 10.*

σ Qstock>10(Product).

3.3 Join:

- ✓ *The join of two relations R1 and R2 is a relation R3, where the tuples are obtained by concatenating the tuples from R1 with those from R2, keeping only those that satisfy the join condition.*

  syntax: *R1⋈R2(condition).*
- ✓ ***Steps of join***

1. ***Cartesian Product****: First, perform the Cartesian product between* ***R1*** *and* ***R2****. This generates all possible combinations of tuples between* ***R1*** *and* ***R2****.*

2. ***Restriction***: *Then, apply the join condition to keep only the tuples that satisfy the condition.*

- ✓ *The tuples of R1 ⋈R2(condition) are all pairs (u1,u2)(u1, u2), where u1is a tuple from R1 and u2 is a tuple from R2 that satisfy the condition.*
- ✓ *The join of two relations R1 and R2 is the* ***Cartesian product*** *of the two relations followed by a restriction based on the condition.*

The join condition must be of the form: ::
Where: attribute1 ∈ 1st relation and attribute2 ∈ 2nd relation, and "::" is a comparison operator.

Remark:

- *The join allows combining two relations using a join condition.*
- *Normalization (putting into canonical normal form) leads to decomposing the relation.*

Example 11:

Let the following two relation:

Product(Prod-ID, Name, Uprice)

Sale (ID-client, Prod-ID, date, QTE-sold)

Give for each sale the product reference, its description, price, ID client, date, and quantity sold.

The table product:

Prod-ID	Name	UPrice
01	car hyundai eon	6000 euro
02	Motocycle MBK	10000 euro
03	bicycle Detwin	300 euro

The table sale:

ID-client	Prod-ID	Date	QTE-sold
01	01	01/01/1998	1
02	02	02/01/1998	3
03	01	03/01/1998	1

Product⋈Sale(Product.Prod-ID=Sale.Prod-ID)

Prod-ID	Name	UPrice	ID-client	Date	QTE-sold
01	car hyundai eon	6000 euro	01	01/01/1998	1
02	Motocycle MBK	10000 euro	02	02/01/1998	3
01	car hyundai eon	6000 euro	03	03/01/1998	1

3.4 Division:

Let there be two relations R1(A1,A2,…,An,B1,B2,…,Bm), R2 (B1, B2, … , Bm) if the schema of R2 is a sub-schema of R1, the division of R1 by R2 is a relation R3 where:

- *The schema of R3 is the complementary sub-schema of R2 with respect to R1.*
- *A tuple (a1,a2,…,an) belongs to R3 if (a1,a2,…,an,b1,b2,…,bm) belongs to R1 for all (b1,b2,…,bm)∈R2.*

Syntax: *R1÷R2*

Example 12:

Here are the following tables; which students are enrolled in all sports?

Here the table Enrollment

Student	sport
julie	Danse
jean	judo
julie	Hand-ball
julie	judo

here the table Sport

sport
Danse
judo
Hand-ball

Student ÷ Sport

Student
julie

3.5 Aggregation:

*In **Relational Algebra**, **γ (gamma)** represents the **aggregation (grouping) operator**. It is used to perform **grouping** and **aggregate functions** like:*

- ***SUM** (Total)*
- ***AVG** (Average)*
- ***COUNT** (Number of occurrences)*
- ***MIN** (Minimum)*
- ***MAX** (Maximum)*

Syntax:

γ grouping attributes, aggregate function(attribute)(relation)

Example 13:

New Database Schema: E-Commerce System

***customers** (customer_id,name, email, city)*

***orders** (order_id , customer_id , order_date,total_amount)*

***products** (product_id , product_name , price)*

order_Items *(order_item_id, order_id , product_id ,quantity)*

Example Queries using γ (Gamma) in Relational Algebra:

1. ***Find the total number of orders per customer***

γ customer_id, COUNT(order_id) (orders)

Explanation:

- *Groups the orders table by customer_id.*
- *Counts how many orders each customer has placed.*

2. ***Find the total revenue (sum of all orders) per customer***

γcustomer_id,SUM(total_amount)(orders)

Explanation:

- *Groups the orders table by customer_id.*
- *Computes the* ***total spending*** *of each customer.*

3. ***Find the most expensive product in each order***

γ order_id, MAX(price) (Order_items ⋈ products (order_items.product_id= products.product_id))

Explanation:

- *Joins order_items and products to get the price of each product in an order.*
- *Groups by order_id and finds the* ***highest price*** *product in each order.*

4. ***Find the average quantity of items per order***

γ order_id, AVG(quantity)(order_items)

Explanation:

- *Groups the order_items table by order_id.*
- *Computes the* ***average quantity*** *of items purchased per order.*

5. ***Find the total quantity of each product sold***

γ product_id, SUM(quantity)(order_items)

Explanation:

- *Groups order_items by product_id.*

- *Computes the **total quantity** sold for each product.*

4. Some examples in Algebric Language:

The algebraic language allows formulating a question through a series of relational algebra operations. This question is called a query.

Exemple 14:

Queries on the CLIENT, PRODUCT, SALE schema of Computer hardware sales office
CLIENT (IdCli, name, city)
PRODUCT (IdPro, description, brand, price)
SALE (IdCli, IdPro, date, quantity)

Question :

1. *Give the client live in canstantine*

σ *(city='constantine') (CLIENT)*

2. *Give the product numbers of Apple brand and with a price < 50000 DA.*

π *IdPro (*σ *(brand='Apple'* $\wedge$ *price< 50000) (PRODUCT))*

3. *Give the brand and the price of the product sold 01/01/2020 with the quantity>3*

π *brand, price(*σ*(date=01/01/2020 AND quantity>3)(PRODUCT* $\bowtie$*SALE(PRODUCT.IdPro= SALE. IdPro)))*

Example 15:

Let the following schema of database of the application Diabetes.AI

user(User-ID, Lastname-user, Firstname-user, email, pass-word, DOB, diabetes_type , Date-creation)

blood-Sugar-measurement (Measurement-ID, User-ID, value, measurement-date, time-of-day)

meal_logs(Meal-ID, User-ID, photo-url, description, glycemic-index, recorded-at)

chatbot-interactions (Intercation-ID, User-ID,question,reponse,date-interaction)

Questions:

Q1) Retrieve all blood sugar readings for a specific user (user_id = 1).

σ User-ID=1(blood_sugar_measurement)

Q2) Find meals with a high glycemic index (>70).

σ glycemic_index>70(meal_logs)

Q3) Get all chatbot interactions for a specific user (user_id = 1).

σ User-ID=1(chatbot_interactions)

Q4) Retrieve all users who have Type 2 diabetes.

σ diabetes_type='Type2' (users)

Q5) Get the names of users along with their latest blood sugar measurement.

π first_name, last_name, value, measurement_date(users ⋈ blood_sugar_measurement (user. User-ID=blood_sugar_measurement. User-ID))

Q6) List meals eaten by a user along with their blood sugar levels after eating.

π meal_logs.description, blood_sugar_measurement.value,

blood_sugar_measurement.measurement_date((meal_logs ⋈blood_sugar_measurement

(meal_logs.User-ID=blood_sugar_measurement.User-ID))

Q7) Retrieve all meals recorded by a specific user (user_id = 2).

σ User-ID=2 (meal_logs)

Q8) Find users who have recorded at least one blood sugar reading.

π users.User-ID, users.first_name, users.last_name(users ⋈ blood_sugar_measurment (users.User-ID =blood_sugar_measurement.User-ID)

Q9) Retrieve all chatbot interactions where the response contains the word "insulin".

σ response LIKE %'insulin'% (chatbot_interactions)

Q10) Find the average blood sugar level of a specific user (User_ID = 3).

γ user_id,AVG(value)(σ User-ID=3(blood_sugar_measurement))

Q11) Get the max blood sugar reading for specific user (User_ID = 3).

γ user_id, MAX(value)(σ User-ID=3(blood_sugar_measurement))

Q12) Retrieve all users who have never interacted with the chatbot.

π users.User-ID, users.first_name, users.last_name(users) -

π users.User-ID, users.first_name, users.last_name(users ⋈ chatbot_interactions(users.User-ID= chatbot_interactions.User-ID))

Q13) Retrieve meals with a high glycemic index (>70) eaten by users with Type 2 diabetes.

π meal_logs.description, meal_logs.glycemic_index, users.first_name, users.last_name(σ meal_logs.glycemic_index>70 (meal_logs ⋈ users)(meal_logs.User-ID =users.User-ID ∧users.diabetes_type='Type2')))

Q14) Find the total number of blood sugar readings recorded per user.

γ User-ID, COUNT(User-ID)(blood_sugar_measurement)

Q15) Retrieve the users who have logged both a meal and a blood sugar reading on the same day.

π users.User-ID, users.first_name, users.last_name((users ⋈ meal_logs ⋈ blood_sugar_measurement)((meal_logs.User-ID =users.User-ID)∧ (meal_logs.User-ID= blood_sugar_measurement.User-ID∧ meal_logs.recorded-at= blood_sugar_measurement.measurement-date)))

Q16*) Find the highest glycemic index recorded per user.*

γ User-ID, MAX(glycemic_index)(meal_logs)

Q17) Find the total number of chatbot interactions per user

γ User-ID, COUNT(Intercation-ID) chatbot_interactions