Querying Relational Databases

Databases

Pierre Senellart

PSL UNIVERSITÉ PARIS

23 February 2026

Outline

The relational algebra

- Algebraic language to express queries
- A relational algebra expression produces a new relation from the database relations
- Each operator takes 0, 1, or 2 subexpressions
- Main operators:

Op.	Arity	Description	Condition
R	0	Relation name	$R \in \mathcal{L}$
$\rho_{A \to B}$	1	Renaming	$A, B \in \mathcal{L}$
$\Pi_{A_1 \dots A_n}$	1	Projection	$A_1 \dots A_n \in \mathcal{L}$
σ_φ	1	Selection	φ formula
$\times$	2	Cross product	
$\cup$	2	Union	schema-compatible
$\setminus$	2	Difference	schema-compatible
$\bowtie_\varphi$	2	Join	φ formula

Relation name

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Expression: `Guest`

Result:

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Renaming

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Expression: $\rho_{\texttt{id}\to\texttt{guest}}(\texttt{Guest})$

Result:

guest	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Projection

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Expression: $\Pi_{\texttt{email},\texttt{id}}(\texttt{Guest})$

Result:

email	id
john.smith@gmail.com	1
alice@black.name	2
john.smith@ens.fr	3

Selection

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Expression: $\sigma_{\texttt{arrival}>\texttt{2026-01-12}\wedge\texttt{guest}=\texttt{2}}(\texttt{Reservation})$

Result:

id	guest	room	arrival	nights
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

The formula used in the selection can be any Boolean combination of comparisons of attributes to attributes or constants.

Cross product

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Expression: $\Pi_{\texttt{id}}(\texttt{Guest}) \times \Pi_{\texttt{name}}(\texttt{Guest})$

Result:

id	name
1	Alice Black
2	Alice Black
3	Alice Black
1	John Smith
2	John Smith
3	John Smith

Cross product

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Expression: $\Pi_{\texttt{id}}(\texttt{Guest}) \times \Pi_{\texttt{name}}(\texttt{Guest})$

Result:

id	name
1	Alice Black
2	Alice Black
3	Alice Black
1	John Smith
2	John Smith
3	John Smith

Cross product is rarely useful on its own, most often as part of a join.

Union

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Expression: $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2}(\texttt{Reservation})) \cup \Pi_{\texttt{room}}(\sigma_{\texttt{arrival}=\text{2026-01-15}}(\texttt{Reservation}))$

Result:

room
107
302
504

Union

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Expression: $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2}(\texttt{Reservation})) \cup \Pi_{\texttt{room}}(\sigma_{\texttt{arrival}=\text{2026-01-15}}(\texttt{Reservation}))$

Result:

room
107
302
504

This simple union could have been written $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2 \vee \texttt{arrival}=\text{2026-01-15}}(\texttt{Reservation}))$. Not always possible.

Difference

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Expression: $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2}(\texttt{Reservation})) \setminus \Pi_{\texttt{room}}(\sigma_{\texttt{arrival}=\text{2026-01-15}}(\texttt{Reservation}))$

Result:

room
107

Difference

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Expression: $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2}(\texttt{Reservation})) \setminus \Pi_{\texttt{room}}(\sigma_{\texttt{arrival}=\text{2026-01-15}}(\texttt{Reservation}))$

Result:

room
107

This simple difference could have been written $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2 \wedge \texttt{arrival} \neq \text{2026-01-15}}(\texttt{Reservation}))$. Not always possible.

Difference

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Expression: $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2}(\texttt{Reservation})) \setminus \Pi_{\texttt{room}}(\sigma_{\texttt{arrival}=\text{2026-01-15}}(\texttt{Reservation}))$

Result:

room
107

This simple difference could have been written $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2 \wedge \texttt{arrival} \neq \text{2026-01-15}}(\texttt{Reservation}))$. Not always possible.

Note: `EXCEPT` has been added fairly recently (2022) to MySQL.

Join

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Expression: $\texttt{Reservation} \bowtie_{\texttt{guest}=\texttt{id}} \texttt{Guest}$

Result:

id	guest	room	arrival	nights	name	email
1	1	504	2026-01-01	5	John Smith	john.smith@gmail.com
2	2	107	2026-01-10	3	Alice Black	alice@black.name
3	3	302	2026-01-15	6	John Smith	john.smith@ens.fr
4	2	504	2026-01-15	2	Alice Black	alice@black.name
5	2	107	2026-01-30	1	Alice Black	alice@black.name

The formula used in the join can be any Boolean combination of comparisons of attributes of the table on the left to attributes of the table on the right.

Note on the join

- The join is not an elementary operator of the relational algebra (but it is very useful)
- It can be seen as a combination of renaming, cross product, selection, projection
- Thus:

$$\begin{aligned} & \texttt{Reservation} \bowtie_{\texttt{guest}=\texttt{id}} \texttt{Guest} \\ \equiv\ & \Pi_{\texttt{id},\texttt{guest},\texttt{room},\texttt{arrival},\texttt{nights},\texttt{name},\texttt{email}}(\\ & \quad \sigma_{\texttt{guest}=\texttt{temp}}(\texttt{Reservation} \times \rho_{\texttt{id}\to\texttt{temp}}(\texttt{Guest}))) \end{aligned}$$

- If R and S have for attributes $\mathcal{A}$ and $\mathcal{B}$, we note $R \bowtie S$ the natural join of R and S, where the join formula is $\bigwedge_{A \in \mathcal{A} \cap \mathcal{B}} R.A = S.A$.

Illegal operations

- All expressions of the relational algebra are not valid
- The validity of an expression generally depends on the database schema
- For example:
  - No reference to the name of a relation that doesn't exist in the database schema
  - One cannot reference (within a renaming, projection, selection, join) an attribute that does not exist in the result of a sub-expression
  - One cannot compute the union or difference of two relations with different schemas
  - One cannot produce (cross product, join, renaming) a table with two attributes with the same name
- Systems implementing the relational algebra may do a static or dynamic verification of these rules, or sometimes ignore them

End-to-end example (relational algebra)

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Question: Which emails correspond to guests who have a reservation for room 504?

Expression: $\Pi_{\texttt{email}}(\sigma_{\texttt{room}=504}(\texttt{Reservation} \bowtie_{\texttt{guest}=\texttt{id}} \texttt{Guest}))$

Result:

email
john.smith@gmail.com
alice@black.name

This is: join $\rightarrow$ select $\rightarrow$ project

Bag semantics

In bag semantics (what is actually used by RDBMS):

- All operations return multisets
- In particular, projection and union can introduce multisets even when initial relations are sets

Outline

SQL

- Structured Query Language, standard language (ISO/IEC 9075, several versions [ISO, 1987, 1999]) to interact with an RDBMS
- Unfortunately, implementation of the standard very variable from one RDBMS to the next
- Many little things (e.g., available types) vary between RDBMSs instead of following the standard
- Differences more syntactical than major
- Where it makes a difference, we use the MySQL version
- Two main parts: DDL (Data Definition Language) to define the schema and DML (Data Manipulation Language) to query and update data
- Declarative language: express what you mean, the system will take care of translating this into an efficient execution plan

Syntax of SQL

- Quite verbose, designed to be almost readable as English words [Chamberlin and Boyce, 1974]
- Keywords are case-insensitive, traditionally written in all uppercase
- Identifiers are either case-sensitive or case-insensitive, depending on the RDBMS and its configuration
- I will try using the following conventions:
  - keywords: upper case
  - table names: capitalized
  - attribute names: lower case
- Comments introduced with --
- SQL statements end with a ";" in some contexts (e.g., command line client) but the ";" is not properly part of the statement

NULL

- In SQL, NULL is a special value that an attribute can take within a tuple
- Denotes absence of value
- Different from 0, from an empty string, etc.
- Weird tri-valued logic: True, False, NULL
- A normal comparison (equality, inequality, etc.) with NULL always returns NULL
- `IS NULL`, `IS NOT NULL` can be used to test whether a value is NULL
- When evaluating a selection predicate, values evaluating to `FALSE` or `NULL` are both rejected

Data Definition Language

SQL

```
CREATE TABLE Guest(id INTEGER, name TEXT, email VARCHAR(256));
CREATE TABLE Reservation(id INTEGER, guest INTEGER,
  room INTEGER, arrival DATE, nights INTEGER);
```

But also:

- `DROP TABLE Guest;` to destroy a table
- `RENAME TABLE Guest TO Guest2;` to rename a table
- `ALTER TABLE Guest MODIFY COLUMN id TEXT;`
  to change the type of a column

Constraints

Specified at the creation of a table, or added later on (with `ALTER TABLE`)

`PRIMARY KEY`	for the primary key; only one per table, it is the key that will be used for physical organization of data; implies `NOT NULL`
`UNIQUE`	for other keys
`REFERENCES`	for foreign keys
`CHECK`	for Check constraints
`NOT NULL`	to indicate that an attribute cannot take the NULL value

Constraints

SQL

```
CREATE TABLE Guest(
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  email VARCHAR(256) UNIQUE CHECK (email LIKE '%@%')
);

CREATE TABLE Reservation(
  id INTEGER PRIMARY KEY,
  guest INTEGER NOT NULL REFERENCES Guest(id),
  room INTEGER NOT NULL CHECK (room>0
    AND room<651),
  arrival DATE NOT NULL,
  nights INTEGER NOT NULL CHECK (nights>0),
  UNIQUE(room, arrival),
  UNIQUE(guest, arrival)
);
```

Updates

- Insertions:

```
INSERT INTO Guest(id,name) VALUES (5,'John');
```

- Deletions:

```
DELETE FROM Reservation WHERE id>4;
```

- Modifications:

```
UPDATE Reservation SET room=205 WHERE room=204;
```

Updates

SQL

```sql
INSERT INTO Guest VALUES
  (1,'John Smith','john.smith@gmail.com'),
  (2,'Alice Black','alice@black.name'),
  (3,'John Smith','john.smith@ens.fr');

INSERT INTO Reservation VALUES
  (1,1,504,'2026-01-01',5),
  (2,2,107,'2026-01-10',3),
  (3,3,302,'2026-01-15',6),
  (4,2,504,'2026-01-15',2),
  (5,2,107,'2026-01-30',1);
```

Queries

General following form:

SQL

```
SELECT ... FROM ... WHERE ...
GROUP BY ... HAVING ...
UNION SELECT ... FROM ...
```

`SELECT`	projection, renaming, aggregation
`FROM`	cross product
`WHERE`	selection (optional)
`GROUP BY`	grouping (optional)
`HAVING`	selection on the grouping (optional)
`UNION`	union (optional)

Other keywords: `ORDER BY` to reorder, `LIMIT` to limit to first k results, `DISTINCT` to impose set semantics, `EXCEPT` for set difference, etc.

Renaming

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

$$\rho_{\texttt{id}\to\texttt{guest}}(\texttt{Guest})$$

SQL

```
SELECT id AS guest, name, email
FROM Guest;
```

Projection

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

$$\Pi_{\texttt{email,id}}(\texttt{Guest})$$

SQL

```sql
SELECT DISTINCT email, id
FROM Guest;
```

Selection

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

$$\sigma_{\texttt{arrival}>\texttt{2026-01-12}\wedge\texttt{guest}=2}(\texttt{Reservation})$$

SQL

```sql
SELECT *
FROM Reservation
WHERE arrival>'2026-01-12' AND guest=2;
```

Cross product

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

$$\Pi_{\texttt{id}}(\texttt{Guest}) \times \Pi_{\texttt{name}}(\texttt{Guest})$$

SQL

```sql
SELECT *
FROM
  (SELECT DISTINCT id FROM Guest) AS temp1,
  (SELECT DISTINCT name FROM Guest) AS temp2;
```

Union

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

$$\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2}(\texttt{Reservation}))$$
$$\cup\, \Pi_{\texttt{room}}(\sigma_{\texttt{arrival}=\texttt{2026-01-15}}(\texttt{Reservation}))$$

SQL

```
  SELECT room
  FROM Reservation
  WHERE guest=2
UNION
  SELECT room
  FROM Reservation
  WHERE arrival='2026-01-15';
```

Difference

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

$$\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2}(\texttt{Reservation}))$$
$$\setminus\ \Pi_{\texttt{room}}(\sigma_{\texttt{arrival}=\texttt{2026-01-15}}(\texttt{Reservation}))$$

SQL

```
  SELECT room
  FROM Reservation
  WHERE guest=2
EXCEPT
  SELECT room
  FROM Reservation
  WHERE arrival='2026-01-15';
```

Join

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

$$\texttt{Reservation} \bowtie_{\texttt{guest}=\texttt{id}} \texttt{Guest}$$

SQL

```sql
SELECT Reservation.*, name, email
FROM Reservation JOIN Guest ON guest=Guest.id;

SELECT Reservation.*, name, email
FROM Reservation, Guest
WHERE guest=Guest.id;
```

End-to-end example (SQL)

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2026-01-01	5
2	2	107	2026-01-10	3
3	3	302	2026-01-15	6
4	2	504	2026-01-15	2
5	2	107	2026-01-30	1

Question: Which emails correspond to guests who have a reservation for room 504?

SQL

```
SELECT DISTINCT email
FROM Reservation JOIN Guest ON guest=Guest.id
WHERE room=504;
```

Outline

References

- Classic textbook on database systems [Garcia-Molina et al., 2008]
- More recent one, also very complete [Silberschatz et al., 2019]
- Classic textbook on database theory [Abiteboul et al., 1995] (advanced material, but the first chapters are readable)
- Details of SQL: standards are not public documents (and not useful for a final user); use the RDBMS documentation instead
- For example, MySQL has a comprehensive documentation at `https://dev.mysql.com/doc/refman/9.6/en/sql-statements.html` (and using \h in the command line client)

Licensing

Bibliography I

Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. http://webdam.inria.fr/Alice/.

Donald D. Chamberlin and Raymond F. Boyce. SEQUEL: A structured English query language. In *Proc. SIGFIDET/SIGMOD Workshop*, volume 1, 1974.

Hector Garcia-Molina, Jeffrey D. Ulman, and Jennifer Widom. *Database Systems: The Complete Book*. Pearson, 2nd edition, 2008.

ISO. *ISO 9075:1987: SQL*. International Standards Organization, 1987.

ISO. *ISO 9075:1999: SQL*. International Standards Organization, 1999.

Avi Silberschatz, Henry F. Korth, and S. Sudarshan. Database system concepts, 2019.