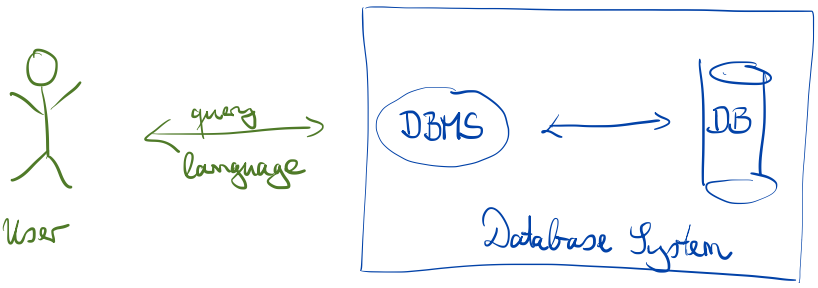
Relational Algebra and Relational Databases

Goal: Provide a logical structure for a dataset to

- prevent inconsistencies
- allow structured queries and manipulation of data
- separate query/manipulation logic from backend implementation



– queries follow logical concepts (relational algebra, SQL, ...)
– Database Management System (DBMS) translates into optimised low-level operations for the Database backend (DB)

Basic definitions:

Defined are the relational algebra expressions, SQL uses different names, DBMS like Microsoft Access or Libreoffice Base yet others.

- a **domain** $\mathbb{D}$ is a set of atomic values ("data type", "field data type")
- an **attribute** $A:\mathbb{D}$ is a name with associated domain $\mathbb{D}$ ("column", "field")
- a **(relation) schema** is a named tuple of attributes:

$$R(A_1:\mathbb{D}_1, \dots, A_n:\mathbb{D}_n)$$

E.g. Student (StudID: integer, StudName: string, Major: string)

- a **relation** $r(R)$ **over a schema** R ("table", "data sheet")

is a set of **tuples** $(d_1, \dots, d_n) \in \mathbb{D}_1 \times \dots \times \mathbb{D}_n$ ("row", "record")

Note:
– relations are sets, so no duplicate tuples allowed (assumption is broken in SQL!)
– order of tuples inside a relation does not matter, order of attributes does not matter
– values of attributes are atomic

Keys:

- a key is a minimal (!) set of attributes in a schema that uniquely identifies each tuple in a relation
- every key is a candidate key
- one key is chosen as the primary key in a relation schema
- a foreign key is a set of attributes in one relation schema that refers to the primary key in another relation schema
- a database schema is a set of relation schemas together with a set of foreign-to-primary key references (in practice, there might be more components, but this is the essence)

Convention: – underline primary keys
– dashed-underline foreign keys

E.g.: simplified university schema (omitting domains where obvious):

Student (StudID, StudName, MajorID)

Major (MajorID, MajorName, Degree: {Bachelor, Master, PhD})

Matriculate (StudID, MajorID)

Constraints that can be enforced by the DBMS:

- foreign key constraint: A foreign key must have a matching primary key in the referenced relation (or be NULL)
- tuple values are restricted to the specified domain
- In practice: other constraints (checksums, uniqueness, non-NULL, ...) may be specified and enforced

For this to work, the database schema should be designed to store any piece of information only once!

Relational Algebra: (and some simple corresponding SQL statements)

① **Union**: If $r(R)$ and $s(R)$, $(r \cup s)(R)$ with

$$r \cup s = \{t : t \in r \text{ or } t \in s\}$$

② **Difference**: If $r(R)$ and $s(R)$, $(r - s)(R)$ with

$$r - s = \{t : t \in r \text{ and } t \notin s\}$$

Note: using union and difference, we can define the intersection

$$r \cap s = r - (r - s)$$

③ **Cartesian product**: If $r(R)$ and $s(S)$ with $R \cap S = \emptyset$ ("no common attributes"), $(r \times s)(R \cup S)$ (regular set union of attributes) and

$$r \times s = \{tq : t \in r \text{ and } q \in s\}$$

(concatenation)

Note: If $R \cap S \neq \emptyset$, use implied renaming of common attributes!

E.g. student (Student):

StudID	StudName	MajorID
1	Liz	A
2	Peter	A
3	Alice	B

class (Class):

CourseNo	Module
101	Analysis
305	Optimization

student × Major:

StudID	StudName	MajorID	CourseNo	Module
1	Liz	A	101	Analysis
1	Liz	A	305	Optimization
2	Peter	A	101	Analysis
2	Peter	A	305	Optimization
3	Alice	B	101	Analysis
3	Alice	B	305	Optimization

④ **Selection**: If $r(R)$, $\left(\sigma_{cond}(r)\right)(R)$ with

$$\sigma_{cond}(r) = \{t : t \in r \text{ and } cond(t) \text{ holds true}\}$$

E.g. $\sigma_{MajorID = 'A'}(student)$:

StudID	StudName	MajorID
1	Liz	A
2	Peter	A

⑤ **Projection**: If $r(R)$, $S \subset R$ a schema, $\left(\pi_S(r)\right)(S)$ with

$$\pi_S(r) = \{s \in S : \text{there is a } t \in r \text{ whose values for attributes in } S \text{ coincide with those in } s\}$$

"delete the columns that are not in S"

E.g. $\pi_{MajorID}(student)$:

MajorID
A
B

⑥ **Rename**: If $r(R)$, $\left(\rho_x(r)\right)(R)$ is the relation named x that contains the same tuples as r.

Used to refer to results of relational algebra expressions

Composed queries:

⑦ **Conditional Join**: If $r(R)$, $s(S)$:

$$r \bowtie_{cond} s = \sigma_{cond}(r \times s)$$

⑧ **Natural Join**: If $r(R)$, $s(S)$:

$$r \bowtie s = \{t \in domain(R \cup S) : \pi_R t \in r \text{ and } \pi_S t \in s\}$$

"Select the tuples that agree on the common attributes of R and S"

Examples: (refers to Libreoffice example database "Media_without_Macros.odb")

① Select media published in 1972

$\sigma_{Pub\_Year = 1972}(Media)$

```
SELECT * FROM Media WHERE Pub_Year = 1972
```

② List all years of publication contained in the DB:

$\Pi_{Pub\_Year}(Media)$

```
SELECT DISTINCT Pub_Year FROM Media
```

③ Example of raw Cartesian product:

Postcode × Town

```
SELECT * FROM Postcode, Town
```

④ List all towns with their postcodes that are in the database

Postcode ⋈ Town

```
SELECT ID, Postcode, Town, Town_ID FROM Postcode, Town
    WHERE Postcode.Town_ID = Town.ID
```

⑤ List all media with author names

Media ⋈ rel_Media_Author ⋈ Author

```
SELECT * FROM Author a, rel_Media_Author r, Media m
    WHERE a.ID = r.Author_ID AND r.Media_ID = m.ID
```

or `SELECT First Name, Last Name, Title, Pub Year FROM ...`