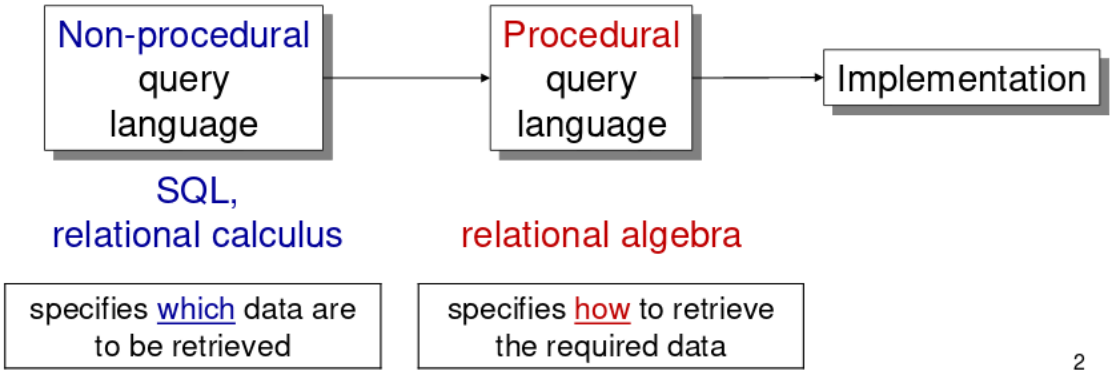
Relational Calculus and Relational Algebra

1 Relational Calculus and Algebra

Relational Calculus: Formal definition of a new relation from existing relations in the DB
Relational Algebra: How to build a new relation from existing relations in the DB

Their place in the big picture



- Relational calculus
  - Relations are considered as **sets of elements** (attribute values)
  - The new relation is defined from the old one(s) using a set theoretic expression
- Relational algebra
  - Based on mathematical relations (tables)
  - A theoretical language with operations on one (or more) input relations (tables) to define one new (output) relation
  - The input relation(s) remain unchanged
- Relational Algebra and Calculus
  - Based on logic; equivalent languages (same expressive power)
  - For every algebra expression ↔ an equivalent calculus expression
- Both are formal and non user friendly languages
  - Used as the basis for higher level DML such as SQL
- A query language is **relationally complete** if it can be used to produce any relation that can be derived using relational algebra/calculus expressions
- Most modern query languages (like SQL)
  - Are relationally complete
  - Have additional operations (summing, grouping, ordering)
  - More expressive power than relational algebra
  - But still not expected to be ”Turing complete”

2 Relational Calculus

In first order logic (or 'predicate calculus'):

- Predicate: Truth valued function (true/false) with arguments
- Proposition: The expression obtained when we substitute values to the arguments of a predicate (can be true/false)
- Let $P(x)$ and $Q(x)$ be two predicates with argument x. Then:
  - The 'set of all x such that both $P(x)$ and $Q(x)$ are true' is:

$$\{x|P(x) \wedge Q(x)\}$$

  - The 'set of all x such that $P(x)$ or $Q(x)$ is true' is:

$$\{x|P(x) \vee Q(x)\}$$

  - The 'set of all x such that $P(x)$ is not true' is:

$$\{x| \sim P(x)\}$$

3 Tuple Relational Calculus

- Variables $\leftrightarrow$ Tuples of a relation
- Aim: To find tuples for which some predicate(s) is true
- To specify that a tuple S belongs to the relation `Staff` we use the predicate `Staff(S)`
- Examples
  - All tuples S of the relation `Staff` that have salary > 10000

$$\{S|Staff(S) \wedge S.salary > 10000\}$$

  - The salaries of all members of staff which earn > 10000

$$\{S.salary|Staff(S) \wedge S.salary > 10000\}$$

Domain relational calculus (another type of calculus):

- Variables take values from domains of attributes of a relation (instead of tuples)
- Use of quantifiers while building predicates:
  - Existential quantifier $\exists$ (there exists)
  - Universal quantifier $\forall$ (for all)
- Tuple variables that are:
  - Quantified by $\exists$ or $\forall \rightarrow$ 'bound variables'
  - Not quantified by $\exists$ or $\forall \rightarrow$ 'Free variables'

Example:
The names of all staff members who work in a branch in london:

$$\{S.name|Staff(S) \wedge (\exists B)(Branch(B) \wedge (B.branchNo = S.branchNo) \wedge (B.city =' London'))\}$$

Here S.name is a free variable and B is a bound variable

4 Relational Algebra

- Both input and output are relations
  - The output can become input to another relation
  - Expressions can be nested
  - This property is called "closure"
- Relations are **closed** under Relational Algebra in the same sense as "numbers are closed under arithmetic expressions"
- In Relational Algebra all involved tuples from the input relation(s) are manipulated in one statement (with no loops)
- Six basic operations in two categories:
  - Unary operations
    - Selection (σ)
    - Projection (π)
    - Rename (ϱ)
  - Binary operations
    - Union ($\cup$)
    - Set Difference ($-$)
    - Cartesian Product ($\times$)
- Several derived operations (that can be expressed using the basic operations)
  - Intersection ($\cap$)
  - Division ($\div$)
  - Join (Natural Join, Equi-Join, Theta Join, Outer Join, Semi-Join)

5 Selection

$\sigma_{predicate}(R)$

- Unary operation i.e. it works on a single relation R
- Outputs a subset of the relation R that contains only the tuples (rows) that satisfy the specified condition (predicate)
- i.e. it returns a "Horizontal Slice" of R

6 Projection

$\Pi_{\text{col-1,...,col-n}}(R)$

- Unary operation
- Outputs a subset of the relation R that contains only the specified attributes (columns) with names $col-1, ..., col-n$ and also eliminates duplicates
- i.e. it returns a "vertical slice" of R (by removing non-matching attributes)
- It can be combined with a selection

7 Union

$R \cup S$

- Binary operation
- Outputs a new relation having all tuples of R, or S, or both R and S, and also eliminates duplicate tuples

That is:

- It combines the rows from both tables, removing any redundant (common) row in the process
- If R has I tuples and S has J tuples, the output relation will have at most $I + J$ tuples

8 Set Difference

$R - S$

- Binary operation
- Outputs a new relation having all tuples that exist in R but not in S

That is:

- It removes from R any common rows that appear in both tables R and S
- If R has I tuples and S has J tuples, the output relation will have at least $I - J$ tuples
- Similarly, $S - R$ removes from S its common rows with R

9 Union Compatibility

To compute $R \cup S$ and $R - S$

- The schemas of the relations R and S must match, i.e.:
  - R and S must have the same number of attributes
  - Every pair of corresponding attributes must have the same domain
- The R and S are called union-compatible

If one of the relations has extra attributes the usual trick is to use projection to create union-compatible relations

10 Intersection

$R \cap S$

- Binary operation
- The output relation has all tuples existing in both R and S

That is:

- It removes from R any rows that appear only in R
- Equivalently: It removes from S any rows appearing only in S
- If R has I tuples and S has J tuples, the output relation will have at most $min\{I, J\}$ tuples

To compute $R \cap S$ the relations R and S must be union-compatible

Intersection is a derived operation

$$R \cap S = R - (R - S)$$

11 Cartesian Product

$R \times S$

- Binary operation
- Outputs a new relation that is a concatenation of every tuple from R with every tuple from S
- No further "compatibility" assumptions on the relations
- Recall: the ordering of the tuples does not matter

That is:

- It "multiplies" the relations R and S
- If R has I tuples, N attributes and S has J tuples, M attributes then $R \times S$ has $(I * J)$ tuples and $(N + M)$ attributes

If R and S have attributes with the same name:

- The attribute names are prefixed with the relation name
- e.g. *R.name* and *S.name*

12 Rename

Relational Algebra operations can be very complex:

- We decompose it into a series of smaller operations
- We give names to the intermediate operations (to reuse them)
- For this we iteratively use the assignment operation " $\leftarrow$ "

A simple (but very useful alternative)

- The rename operation $\varrho_X(E)$ returns E renames as X
- Moreover: $\varrho_{X(A_1,A_2,...,A_n)}(E)$ returns E renamed as X, where the attributes of X are defined as $A_1, A_2, ..., A_n$

Especially useful when, for example:

- We want to compute a join of a relation with itself (i.e. we can create a copy of the relation with a different name)

13 Division

$R \div S$

- Binary operation, i.e. it works on two relations R and S
- A particular type of query that appears often in applications

Notation:

- Let R have the set of attributes A
- Let S have the set of attributes B, where $B \subseteq A$
- Define $C = A - b$ (i.e. the attributes of R that are not in S)

The division operator $R \div S$ outputs a relation over the attributes C that consists of the tuples from R that match every tuple in S

Division $R \div S$ is a derived operation

- Compute all C-tuples of R that are not "disqualified" by a tuple in S
- A C-tuple of R is "disqualified", if by attaching it to a tuple of S, we obtain a tuple that is not in R
- Disqualified C-tuples of R: $\Pi_C((\Pi_C(R) \times S) - R)$
- Tuples of $R \div S$: $\Pi_C(R)$- disqualified tuples

14 Join

- The combination of Cartesian product and Selection can be reduced to a single operation, called a Join
- A Join is equivalent to:
  - Build the Cartesian product of the two operand relations
  - Perform a Selection (using the join predicate F)
- Notation
  - $R \bowtie_F S = \sigma_F(R \times S)$ where F is a predicate
  - If F contains only "=", then this is an Equijoin

15 Natural Join

- $A_1, A_2, ..., A_k$: common attributes of relations R and S
- $R \bowtie S = \Pi_{C_1,...,C_x}(\sigma_{R.A_1=S.A_1,...,R.A_k=S.A_k}(R \times S))$
  Where $C_1, ..., C_x$ are the attributes of $R \times S$ without the duplicates
- It is an equijoin of the relations R and S over all their attributes that have the same name, without duplications

Steps:

- $R \times S$
- For each attribute with the same name A in both R and S, select the tuples where $R.A = S.A$ (from $R \times S$)
- Remove one of the duplicate columns corresponding to the above pairs of attributes

16 Summary