RELATIONAL ALGEBRA

Relational Query Languages

- *Query languages*: Allow manipulation and retrieval of data from a database.
- Relational model supports simple, powerful QLs:
  - Formal foundation based on logic.
  - Allows for optimization.
- Query Languages != programming languages!
  - QLs not intended to be used for complex calculations.
  - QLs support easy, efficient access to large data sets.

Credit: R. Ramakrishnan, J. Gehrke

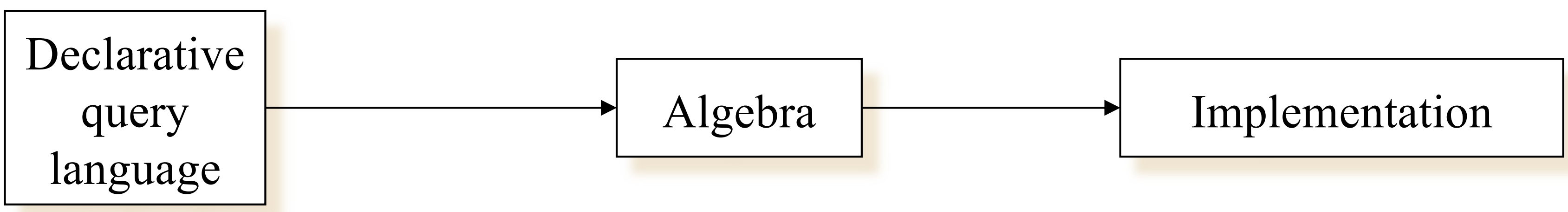
DBMS Architecture

How does a SQL engine work ?

- SQL query $\rightarrow$ relational algebra plan
- Relational algebra plan $\rightarrow$ Optimized plan
- Execute each operator of the plan

Relational Algebra

- Formalism for creating new relations from existing ones
- Its place in the big picture:



SQL,
relational calculus

Relational algebra
Relational bag algebra

Formal Relational Query Languages

Two mathematical Query Languages form the basis for SQL, and for implementation:

- *Relational Algebra*: More operational, very useful for representing execution plans.
- *Relational Calculus*: Lets users describe what they want, rather than how to compute it. (Non-operational, *declarative*.)

What is an "Algebra"

- Mathematical system consisting of:
  - *Operands* --- variables or values from which new values can be constructed.
  - *Operators* --- symbols denoting procedures that construct new values from given values.

What is Relational Algebra?

- An algebra whose operands are relations or variables that represent relations.
- Operators are designed to do the most common things that we need to do with relations in a database.
  - The result is an algebra that can be used as a *query language* for relations.

Core Relational Algebra

- Union, intersection, and difference.
  - Usual operations, but *both operands must have the same relation schema*.
- Selection: picking certain rows.
- Projection: picking certain columns.
- Products and joins: compositions of relations.
- Renaming of relations and attributes.

Since each operation returns a relation, operations can be *composed*

Selection

- R1 := σ_C (R2)
  - C is a condition (as in "if" statements) that refers to attributes of R2.
  - R1 is all those tuples of R2 that satisfy C.

Example: Selection

Relation Sells:

bar	beer	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

JoeMenu := $\sigma_{bar=\text{“Joe's”}}$(Sells):

bar	beer	price
Joe's	Bud	2.50
Joe's	Miller	2.75

- Selects rows that satisfy *selection condition*.
- *Schema* of result identical to schema of (only) input relation.
- *Result* relation can be the *input* for another relational algebra operation. (*Operator composition*.)

Projection

- R1 := π_L (R2)
  - *L* is a list of attributes from the schema of R2.
  - R1 is constructed by looking at each tuple of R2, extracting the attributes on list *L*, in the order specified, and creating from those components a tuple for R1.

Example: Projection

Relation Sells:

bar	beer	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

Prices := $\pi_{beer,price}$(Sells):

beer	price
Bud	2.50
Bud	2.50
Miller	2.75
Miller	3.00

Schema of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation.

Example

Relation Sells:

bar	beer	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

JoePrices := $\pi_{beer,price}(\sigma_{bar=\text{“Joe's”}}(Sells))$

Beer	Price
Bud	2.50
Miller	2.75

Extended Projection

- Using the same π_L operator, we allow the list L to contain arbitrary expressions involving attributes:
  - Arithmetic on attributes, e.g., $A+B \rightarrow C$.

Example: Extended Projection

R = (A B)

A	B
1	2
3	4

$\pi_{A+B \rightarrow C, A \rightarrow A1, A \rightarrow A2}$ (R) =

C	A1	A2
3	1	1
7	3	3

Product

- R3 := R1 X R2
  - Pair each tuple t1 of R1 with each tuple t2 of R2.
  - Concatenation t1t2 is a tuple of R3.
  - Schema of R3 is the attributes of R1 and then R2, in order.
  - But beware attribute *A* of the same name in R1 and R2:
    - In relational algebra use renaming to distinguish
    - in SQL use R1.*A* and R2.*A*.

Example: R3 := R1 X R2

R1(

A	B
1	2
3	4

)

R2(

B	C
5	6
7	8
9	10

)

R3(

A,	R1.B,	R2.B,	C
1	2	5	6
1	2	7	8
1	2	9	10
3	4	5	6
3	4	7	8
3	4	9	10

)

Theta-Join

- R3 := R1 $\bowtie_C$ R2
  - Take the product R1 X R2.
  - Then apply σ_C to the result.
- As for σ_C can be any boolean-valued condition.
  - A θ B, where θ is =, <, etc.; hence the name "theta-join."

Example: Theta Join

Sells(

bar,	beer,	price)
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

Bars(

name,	addr
Joe's	Maple St.
Sue's	River Rd.

)

$$\text{BarInfo} := \text{Sells} \bowtie_{\text{Sells.bar} = \text{Bars.name}} \text{Bars}$$

BarInfo(

bar,	beer,	price,	name,	addr
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

)

Natural Join

- A useful join variant (*natural* join) connects two relations by:
  - Equating attributes of the same name, and
  - Projecting out one copy of each pair of equated attributes.
- Denoted R3 := R1 $\bowtie$ R2.

Example: Natural Join

Sells(

bar,	beer,	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

)

Bars(

bar,	addr
Joe's	Maple St.
Sue's	River Rd.

)

BarInfo := Sells $\bowtie$ Bars

Note: Bars.name has become Bars.bar to make the natural join non-trivial

BarInfo(

bar,	beer,	price,	addr
Joe's	Bud	2.50	Maple St.
Joe's	Milller	2.75	Maple St.
Sue's	Bud	2.50	River Rd.
Sue's	Coors	3.00	River Rd.

)

Renaming

- The ρ operator gives a new schema to a relation.
- R1 := $\rho_{R1(A1,\ldots,An)}$(R2) makes R1 be a relation with attributes A1,…,A*n* and the same tuples as R2.
- Simplified notation: R1(A1,…,A*n*) := R2.

Example: Renaming

Bars(

name,	addr
Joe's Sue's	Maple St. River Rd.

)

R(bar, addr) := Bars

R(

bar,	addr
Joe's Sue's	Maple St. River Rd.

)

Set Operators

- Union, Intersection and Difference are defined only for **union compatible** relations.
- Two relations are union compatible if they have the same set of attributes and the types (domains) of the attributes are the same.
- E.g., two relations that are not union compatible:
  - **Student (sNumber, sName)**
  - **Course (cNumber, cName)**

Relational Algebra on Bags

- A *bag* (or *multiset*) is like a set, but an element may appear more than once.
- Example: {1,2,1,3} is a bag.
- Example: {1,2,3} is also a bag that happens to be a set.

Union: ∪

- Consider two bags R_1 and R_2 that are union-compatible. Suppose a tuple t appears in R_1 m times, and in R_2 n times. Then in the union, t appears $m + n$ times.

R_1

A	B
1	2
3	4
1	2

R_2

A	B
1	2
3	4
5	6

$R_1 \cup R_2$

A	B
1	2
1	2
1	2
3	4
3	4
5	6

Intersection: $\cap$

- Consider two bags R_1 and R_2 that are union-compatible. Suppose a tuple t appears in R_1 m times, and in R_2 n times. Then in the intersection, t appears min (m, n) times.

R_1

A	B
1	2
3	4
1	2

R_2

A	B
1	2
3	4
5	6

$R_1 \cap R_2$

A	B
1	2
3	4

Difference: -

- Consider two bags R_1 and R_2 that are union-compatible. Suppose a tuple t appears in R_1 m times, and in R_2 n times. Then in $R_1 - R_2$, t appears max (0, m - n) times.

R_1

A	B
1	2
3	4
1	2

R_2

A	B
1	2
3	4
5	6

$R_1 - R_2$

A	B
1	2

Building Complex Expressions

- Combine operators with parentheses and precedence rules.
- Three notations, just as in arithmetic:
  - Sequences of assignment statements.
  - Expressions with several operators.
  - Expression trees.

Sequences of Assignments

- Create temporary relation names.
- Renaming can be implied by giving relations a list of attributes.
  - $\pi_{A+B\text{->}C,A\text{->}A1,A\text{->}A2}$ (R)
- Example: R3 := R1 $\bowtie_C$ R2 can be written:

R4 := R1 X R2

R3 := σ_C (R4)

Expressions in a Single Assignment

Example: the theta-join R3 := R1 $\bowtie_C$ R2 can be written as

- R3 := σ_C (R1 X R2)

- Precedence of relational operators: (parentheses supercedes)
  - [σ, π, ρ] (highest).
  - [X, $\bowtie$].
  - $\cap$.
  - [$\cup$, —]

Expression Trees

- Leaves are operands --- either variables standing for relations or particular, constant relations.
- Interior nodes are operators, applied to their child or children.

Example: Tree for a Query

- Using the relations Bars(name, addr) and Sells(bar, beer, price), find the names of all the bars that are either on Maple St. or sell Bud for less than $3.

As a Tree:

Using the relations Bars(name, addr) and Sells(bar, beer, price), find the names of all the bars that are either on Maple St. or sell Bud for less than $3.