CSCI317 Database Performance Tuning

Relational Algebra

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Relational Algebra

Outline

Relational algebra ? What is it ?

Operations

`SELECT` statement versus relational algebra

Relational algebra ? What is it ?

Relational algebra is one two formal query languages associated with the relational database model

Relational algebra is a collection operations whose arguments are relational tables, sets of attributes, and formulas of prepositional calculus

An operation of relational algebra accepts one or two relational tables as the arguments and it returns a relational table as the result

The other arguments of relational algebra operations are set of attributes (columns) and formulas of prepositional calculus

What about the other formal language?

The other formal language is called as a relational calculus and it is based on the language of First Order Logic

Relational Algebra

Outline

- Relational algebra ? What is it ?
- Operations
- `SELECT` statement versus relational algebra

Operations

Query

- A query expressed in the language of relational algebra is an expression built over the operations of relational algebra and the arguments like relational tables, sets of attributes and fromulas of prepositional calculus

Operations

- Selection: $\sigma_{\varphi}(r)$
- Projection: $\pi_{X}(r)$
- Cross-product: $r \times s$
- Join: $r \bowtie_{\varphi} s$
- Antijoin: $r \sim_{\varphi} t$
- Set operations: $r \cup t$ (union), $r \cap t$ (intersection) $r - t$ (difference)

Operations

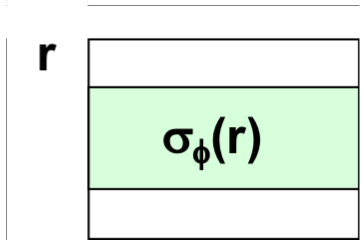
Selection

Let r be a relational table built over a schema (a set of attributes) Y

Selection operation σ applied to a relational table r is denoted by $\sigma_{\varphi}(r)$ and it returns a set of all rows $v \in r$ such that evaluation of $\varphi(v)$ returns TRUE

For example $\sigma_{salary > 80K}(EMPLOYEE)$ returns all rows from EMPLOYEE table that have a value of attribute salary greater than 80K

Selection operation horizontally restricts a relational table



Operations

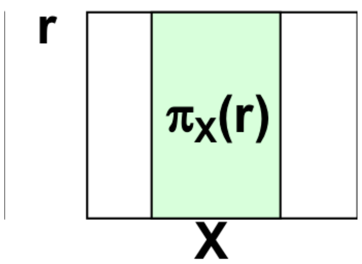
Projection

Let r be a relational table built over a schema (a set of attributes) Y, let $X \subseteq Y$

Projection operation π applied to a relational table r is denoted by $\pi_X(r)$ and it returns a set of all rows v such that there exists a row $w \in r$ and $v = w[X]$

For example $\pi_{\{e\#,salary\}}$ (EMPLOYEE) returns a table built over a schema {e#,salary} (two columns) such that each one of its rows in "included" in a relational table EMPLOYEE

Projection operation vertically restricts a relational table



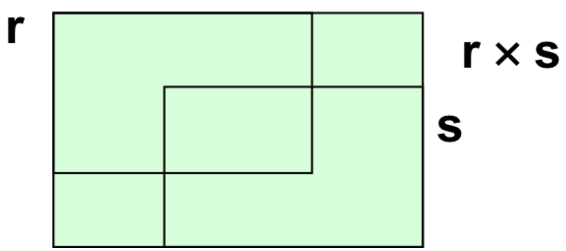
Operations

Cross-product

Let r be a relational table built over a schema (a set of attributes) Y, and s be a relational table built over a schema X

Cross-product operator × applied to the relational tables r and s is denoted by r × s and it returns all rows v such that v = concat(u,w), u ∈ r and w ∈ s

Cross product operation returns all possible concatenations of the rows from r and s



Operations

Cross-product

For example, an expression
EMPLOYEE(e#, name, salary, dname) × DEPARTMENT(dname, budget)

returns a relational table
ED(e#, name, salary, E.dname, D.dname, budget)
that consists of the concatenations of all rows from the relational tables EMPLOYEE and DEPARTMENT

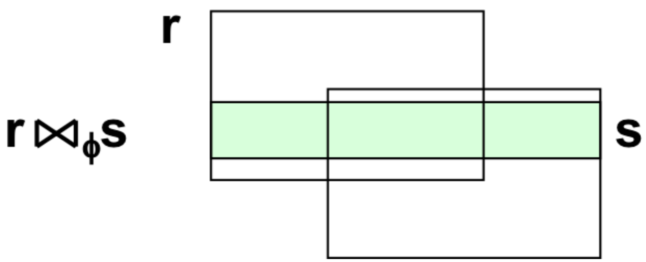
Operations

Join

Let r be a relational table built over a schema (a set of attributes) Y, and s be a relational table built over a schema X, let φ be a formula of propositional logic over $X \cup Y$

Join operation $\bowtie_\varphi$ applied to the relational tables r and s is denoted by $r \bowtie_\varphi s$ and it returns a value of relational algebra expression $\sigma_\varphi(r \times s)$

Join operation concatenates all pairs of rows(v,w), where $v \in r$ and $w \in s$ that satisfy a given condition φ(v,w)



Operations

Join

For example, an expression

$\text{EMP1(e\#, salary, dname)} \bowtie_{\text{EMP1.salary > EMP2.salary}} \text{EMP2(e\#, salary, dname)}$

returns a relational table

EMP12(E1.e#,E1.salary,E1.dname,E2.e#,E2.salary,E2.dname)

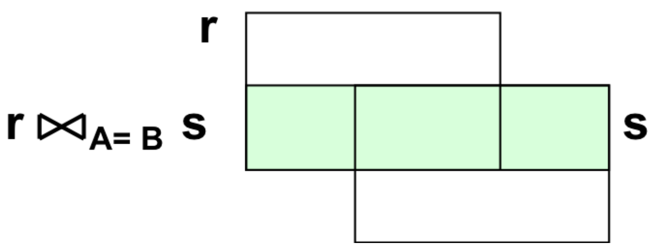
that contains the concatenations of all rows from EMP1 and EMP2 and such that EMP1.salary > EMP2.salary

Operations

Equijoin

Let r be a relational table built over a schema (a set of attributes) Y, and s be a relational table built over a schema X, let $A \in Y$ and $B \in X$

Equijoin operation $\bowtie_{A=B}$ applied to the relational table r and s is denoted by
$r \bowtie_{A=B} s$ and it returns a value of relational algebra expression
$\sigma_{r.A=s.B}(r \times s)$



Operations

Equijoin

For example

EMP(e#, name, salary, dname) $\bowtie_{\text{EMP.dname, DEPT.dname}}$ DEPT(dname, budget)

returns a relational table

ED(e#, name, salary E.dname, D.dname, budget)

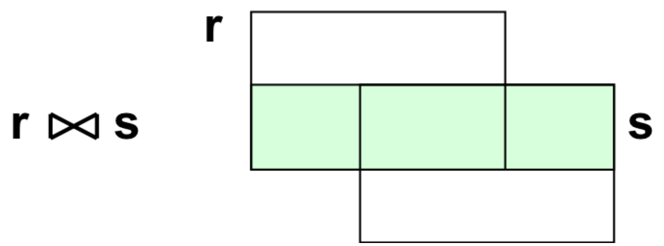
that consists of the concatenations of all rows the relational tables EMP and DEPT such that EMP.dname = DEPT.dname

Operations

Natural join

Let r be a relational table built over a schema (a set of attributes) Y, and s be a relational table built over a schema X, let $A = Y \cap X$

Natural join operation $\bowtie$ applied to the relational tables r and s is denoted by $r \bowtie s$ and it returns a value of relational algebra expression $\pi_{\text{schema(r)} \cup \text{schema(s)}} (\sigma_{r.A=s.A} (r \times s))$



Operations

Natural join

For example, an expression
EMP(e#, name, salary, dname) ⋈ DEPT(dname, budget)

returns a relational table ED(e#,name, dname, budget)

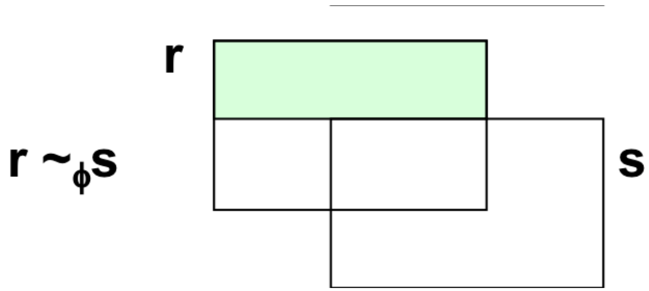
that consists of the concatenations of all rows from the relational tables EMP and DEPT and EMP.dname = DEPT.dname = ED.dname

Operations

Antijoin

Let r be a relational table built over a schema (a set of attributes) Y, and s be a relational table built over a schema X, let φ be a propositional formula over $X \cup Y$

Antijoin operation $\sim_{\varphi}$ applied to the relational tables r and s is denoted by
$r \sim_{\varphi} s$ and it returns a value of relational algebra expression
$r - \pi_Y(\sigma_{\varphi}(r \times s))$



Operations

Antijoin

For example, an expression
$EMP1(e\#, salary, dname) \sim_{EMP1.salary\ =\ EMP2.salary} EMP2(e\#, salary, dname)$

returns a relational table EMP12(E1.e#,E1.salary,E1.dname)

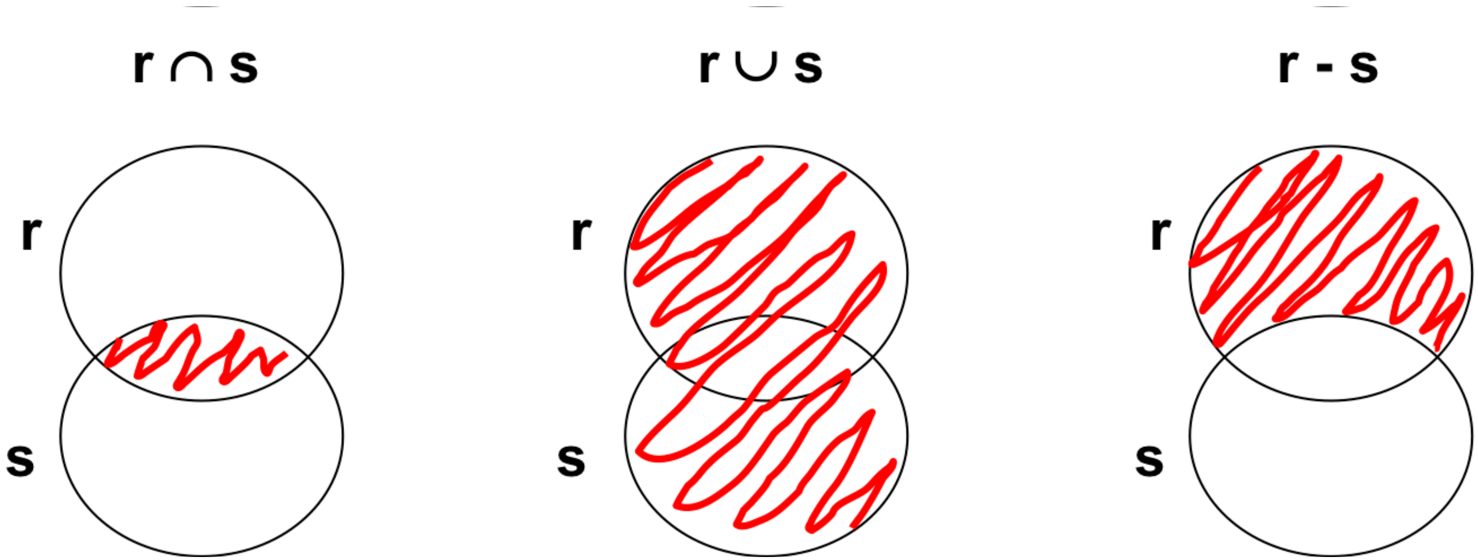
that contains all rows from a relational table EMP1 such that for all values in EMP1.salary does not exist a value EMP2.salary where EMP1.salary = EMP2.salary

Operations

Set operators

Let r be a relational table built over a schema (a set of attributes) Y, and s be a relational table built over a schema Y

Visualizations of set operations are the following



Relational Algebra

Outline

- Relational algebra ? What is it ?
- Operations
- `SELECT` statement versus relational algebra

SELECT statement versus relational algebra

Any SELECT statement that does not contain GROUP BY clause and group functions can be translated into a relational algebra expression

SELECT statement with a selection and projection

```
SELECT S_NAME, S_ADDRESS
FROM SUPPLIER
WHERE S_ACCTBAL > 100;
```

$\pi_{S\_NAME,S\_ADDRESS}(\sigma_{S\_ACCTBAL > 100}(SUPPLIER))$

SELECT statement with a selection, projection, and INTERSECT operation

```
SELECT P_NAME
FROM PART
WHERE P_SIZE = 6
 INTERSECT
SELECT P_NAME
FROM PART
WHERE P_TYPE = 'XYZ';
```

$\pi_{P\_NAME}(\sigma_{P\_SIZE=6}(PART)) \cap \pi_{P\_NAME}(\sigma_{P\_TYPE='XYZ'}(PART))$

SELECT statement versus relational algebra

SELECT statement a selection, projection, and join operation

```
SELECT L_SHIPDATE
FROM LINEITEM, ORDERS
WHERE LINEITEM.L_ORDERKEY = ORDERS.O_ORDERKEY AND
      ORDERS.O_TOTALPRICE > 100;
```

$$\pi_{L\_SHIPDATE}(\sigma_{O\_TOTALPRICE>100}(ORDERS \bowtie_{L\_ORDERKEY = O\_ORDERKEY} LINEITEM))$$

SELECT statement with a selection, projection, and two join operations

```
SELECT C_NAME, N_NAME
FROM CUSTOMER JOIN ORDERS
              ON CUSTOMER.C_CUSTKEY = ORDERS.O_CUSTKEY
              JOIN NATION
              ON CUSTOMER.C_NATIONKEY = NATION.N_NATIONKEY
WHERE C_ACCTBAL = 0;
```

$$\pi_{C\_NAME,\ N\_NAME}(\sigma_{C\_ACCTBAL=0}(CUSTOMER) \bowtie NATION) \bowtie ORDERS))$$

SELECT statement versus relational algebra

SELECT statement with a projection and left outer join operation

```
SELECT C_CUSTKEY, O_ORDERKEY
FROM CUSTOMER LEFT OUTER JOIN ORDERS
              ON C_CUSTKEY = O_CUSTKEY;
```

$((\pi_{C\_CUSTKEY} (CUSTOMER) - \pi_{O\_CUSTKEY} (ORDERS)) \times \{NULL\}) \cup$
$\pi_{C\_CUSTKEY,O\_ORDERKEY} (CUSTOMER \bowtie_{C\_CUSTKEY=O\_CUSTKEY} ORDERS)$

Nested SELECT statement

```
SELECT O_TOTALPRICE
FROM ORDERS
WHERE O_CUSTKEY IN (SELECT C_CUSTKEY
                    FROM CUSTOMER
                    WHERE C_ACCTBAL = 0);
```

$\pi_{O\_TOTALPRICE} (ORDERS \bowtie_{O\_CUSTKEY=C\_CUSTKEY} (\sigma_{C\_ACCTBAL=0} (CUSTOMER)))$

SELECT statement versus relational algebra

Nested SELECT statement

```
SELECT O_TOTALPRICE
FROM ORDERS
WHERE O_CUSTKEY NOT IN (SELECT C_CUSTKEY
                        FROM CUSTOMER
                        WHERE C_ACCTBAL = 0);
```

$\pi_{O\_TOTALPRICE}(ORDERS \sim_{O\_CUSTKEY=C\_CUSTKEY} (\sigma_{C\_ACCTBAL=0}(CUSTOMER)))$

Nested SELECT statement with existential quantifier

```
SELECT O_TOTALPRICE
FROM ORDERS
WHERE EXISTS (SELECT C_CUSTKEY
              FROM CUSTOMER
              WHERE ORDERS.O_CUSTKEY = CUSTOMER.C_CUSTKEY AND C_ACCTBAL = 0);
```

$\pi_{O\_TOTALPRICE}(ORDERS \bowtie_{O\_CUSTKEY=C\_CUSTKEY} (\sigma_{C\_ACCTBAL=0}(CUSTOMER)))$

SELECT statement versus relational algebra

Nested SELECT statement with negated existential quantifier

```
SELECT O_TOTALPRICE
FROM ORDERS
WHERE NOT EXISTS (SELECT C_CUSTKEY
                  FROM CUSTOMER
                  WHERE ORDERS.O_CUSTKEY = CUSTOMER.C_CUSTKEY AND C_ACCTBAL = 0);
```

$\pi_{O\_TOTALPRICE}(ORDERS \sim_{O\_CUSTKEY=C\_CUSTKEY} (\sigma_{C\_ACCTBAL=0}(CUSTOMER)))$

References

R. Ramakrishnan and J. Gherke Database Management Systems, 3rd ed. Mc Graw-Hill, 2003,chapter 4.2