# CSC 443 – Data Base Management Systems

Lecture 9 – Introduction to Relational Algebra

## What are Relational Algebra and Relational Calculus?

- Relational algebra and relational calculus are formal languages associated with the relational model.
  - Informally, relational algebra is a (high-level) procedural language and relational calculus a non-procedural language.
  - However, formally both are equivalent to one another.
- A language that produces a relation that can be derived using relational calculus is relationally complete.

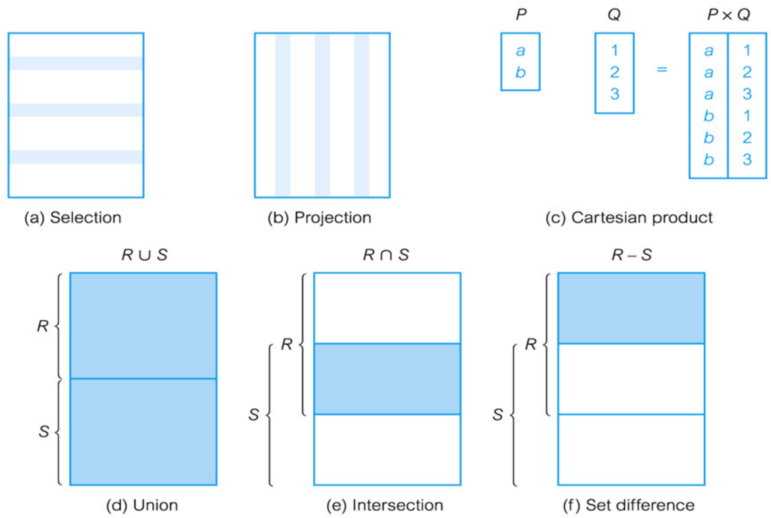
## What is Relational Algebra?

- Relational algebra operations work on one or more relations to define another relation without changing the original relations.
- Both operands and results are relations, so output from one operation can become input to another operation.
- Allows expressions to be nested, just as in arithmetic. This property is called ***closure***.

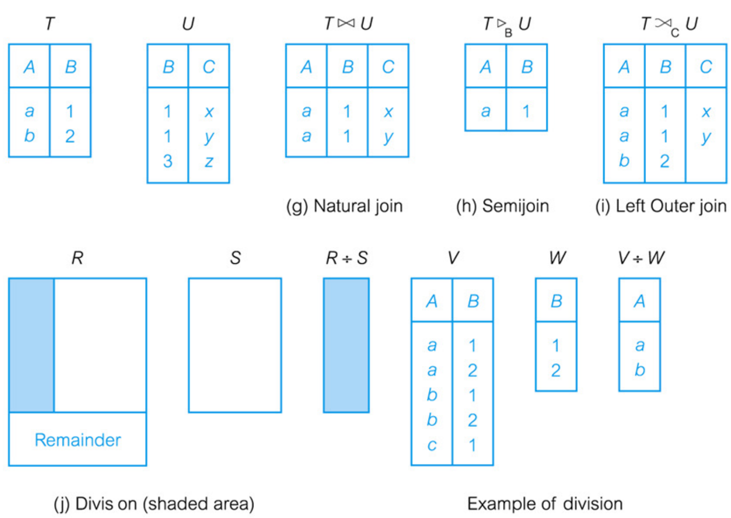
## Operations of Relational Algebra

- Five basic operations in relational algebra: Selection, Projection, Cartesian product, Union, and Set Difference.
- These perform most of the data retrieval operations needed.
- Also have Join, Intersection, and Division operations, which can be expressed in terms of 5 basic operations.

# Operations of Relational Algebra




# Operations of Relational Algebra



# Selection (or Restriction)

- $\sigma_{predicate}$ (R)
  - Works on a single relation R and defines a relation that contains only those tuples (rows) of R that satisfy the specified condition (***predicate***).

# Selection – An Example

- List all staff with a salary greater than £10,000.

  $\sigma_{salary > 10000}$ (Staff)

| staffNo | fName | lName | position | sex | DOB | salary | branchNo |
|---|---|---|---|---|---|---|---|
| SL21 | John | White | Manager | M | 1-Oct-45 | 30000 | B005 |
| SG37 | Ann | Beech | Assistant | F | 10-Nov-60 | 12000 | B003 |
| SG14 | David | Ford | Supervisor | M | 24- Mar-58 | 18000 | B003 |
| SG5 | Susan | Brand | Manager | F | 3-Jun-40 | 24000 | B003 |

# Projection

- $\Pi_{col1, \ldots, coln}(R)$
  - Works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.

# Projection – An Example

- Produce a list of salaries for all staff, showing only staffNo, fName, lName, and salary details.

$\Pi_{staffNo,\ fName,\ lName,\ salary}(Staff)$

| staffNo | fName | lName | salary |
|---|---|---|---|
| SL21 | John | White | 30000 |
| SG37 | Ann | Beech | 12000 |
| SG14 | David | Ford | 18000 |
| SA9 | Mary | Howe | 9000 |
| SG5 | Susan | Brand | 24000 |
| SL41 | Julie | Lee | 9000 |

# Union

- R ∪ S
  - Union of two relations R and S defines a relation that contains all the tuples of R, or S, or both R and S, duplicate tuples being eliminated.
  - R and S must be union-compatible.
- If R and S have *I* and *J* tuples, respectively, union is obtained by concatenating them into one relation with a maximum of (*I* + *J*) tuples.

# Union – An Example

- **List all cities where there is either a branch office or a property for rent.**

  $\Pi_{city}$**(Branch)** ∪ $\Pi_{city}$**(PropertyForRent)**

| city |
|---|
| London |
| Aberdeen |
| Glasgow |
| Bristol |

## Set Difference

- R – S
  - Defines a relation consisting of the tuples that are in relation R, but not in S.
  - R and S must be union-compatible.

## Set Difference – An Example

- **List all cities where there is a branch office but no properties for rent.**

  $\Pi_{city}$**(Branch) –** $\Pi_{city}$**(PropertyForRent)**

| city |
| --- |
| Bristol |

# Intersection

- **$R \cap S$**
  - **Defines a relation consisting of the set of all tuples that are in both R and S.**
  - **R and S must be union-compatible.**
- **Expressed using basic operations:**

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

# Intersection – An Example

- List all cities where there is both a branch office and at least one property for rent.

  $\Pi_{city}(Branch) \cap \Pi_{city}(PropertyForRent)$

| city |
| --- |
| Aberdeen |
| London |
| Glasgow |

# Cartesian Product

- R X S
  - Defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.

# Cartesian Product – An Example

- **List the names and comments of all clients who have viewed a property for rent.**

**$(\Pi_{clientNo,\ fName,\ lName}(Client))$ X $(\Pi_{clientNo,\ propertyNo,\ comment}(Viewing))$**

| client.clientNo | fName | lName | Viewing.clientNo | propertyNo | comment |
|---|---|---|---|---|---|
| **CR76** | **John** | **Kay** | **CR56** | **PA14** | **too small** |
| **CR76** | **John** | **Kay** | **CR76** | **PG4** | **too remote** |
| CR76 | John | Kay | CR56 | PG4 | |
| CR76 | John | Kay | CR62 | PA14 | no dining room |
| CR76 | John | Kay | CR56 | PG36 | |
| CR56 | Aline | Stewart | CR56 | PA14 | too small |
| CR56 | Aline | Stewart | CR76 | PG4 | too remote |
| CR56 | Aline | Stewart | CR56 | PG4 | |
| CR56 | Aline | Stewart | CR62 | PA14 | no dining room |
| CR56 | Aline | Stewart | CR56 | PG36 | |
| CR74 | Mike | Ritchie | CR56 | PA14 | too small |
| CR74 | Mike | Ritchie | CR76 | PG4 | too remote |
| CR74 | Mike | Ritchie | CR56 | PG4 | |
| CR74 | Mike | Ritchie | CR62 | PA14 | no dining room |
| CR74 | Mike | Ritchie | CR56 | PG36 | |
| CR62 | Mary | Tregear | CR56 | PA14 | too small |
| CR62 | Mary | Tregear | CR76 | PG4 | too remote |
| CR62 | Mary | Tregear | CR56 | PG4 | |
| CR62 | Mary | Tregear | CR62 | PA14 | no dining room |
| CR62 | Mary | Tregear | CR56 | PG36 | |

# Cartesian Product and Selection – An Example

- Use selection operation to extract those tuples where Client.clientNo = Viewing.clientNo.

$\sigma_{Client.clientNo = Viewing.clientNo}((\prod_{clientNo,\ fName,\ lName}(Client))$ X
$(\prod_{clientNo,\ propertyNo,\ comment}(Viewing)))$

| client.clientNo | fName | lName | Viewing.clientNo | propertyNo | comment |
|---|---|---|---|---|---|
| CR76 | John | Kay | CR76 | PG4 | too remote |
| CR56 | Aline | Stewart | CR56 | PA14 | too small |
| CR56 | Aline | Stewart | CR56 | PG4 | |
| CR56 | Aline | Stewart | CR56 | PG36 | |
| CR62 | Mary | Tregear | CR62 | PA14 | no dining room |

- Cartesian product and Selection can be reduced to a single operation called a ***Join***.

# Join Operations

- Join is a derivative of Cartesian product.
- Equivalent to performing a Selection, using join predicate as selection formula, over Cartesian product of the two operand relations.
- One of the most difficult operations to implement efficiently in an RDBMS and one reason why RDBMSs have intrinsic performance problems.

# Join Operations

- Various forms of join operation
  - Theta join
  - Equijoin (a particular type of Theta join)
  - Natural join
  - Outer join
  - Semijoin

# Theta join (θ-join)

- $R \bowtie_F S$
  - Defines a relation that contains tuples satisfying the predicate F from the Cartesian product of R and S.
  - The predicate F is of the form $R.a_i \ \theta \ S.b_i$ where θ may be one of the comparison operators ($<, \leq, >, \geq, =, \neq$).

# Theta join (θ-join)

- Can rewrite Theta join using basic Selection and Cartesian product operations.

  $R \bowtie_F S = \sigma_F(R \times S)$

- Degree of a Theta join is sum of degrees of the operand relations R and S. If predicate F contains only equality (=), the term Equijoin is used.

# Equijoin – An Example

- List the names and comments of all clients who have viewed a property for rent.

  $(\Pi_{clientNo,\ fName,\ lName}(Client)) \bowtie_{Client.clientNo = Viewing.clientNo} (\Pi_{clientNo,\ propertyNo,\ comment}(Viewing))$

| client.clientNo | fName | lName | Viewing.clientNo | propertyNo | comment |
|---|---|---|---|---|---|
| CR76 | John | Kay | CR76 | PG4 | too remote |
| CR56 | Aline | Stewart | CR56 | PA14 | too small |
| CR56 | Aline | Stewart | CR56 | PG4 | |
| CR56 | Aline | Stewart | CR56 | PG36 | |
| CR62 | Mary | Tregear | CR62 | PA14 | no dining room |

# Natural Join

- R ⋈ S
  – An Equijoin of the two relations R and S over all common attributes *x*. One occurrence of each common attribute is eliminated from the result.

# Natural Join – An Example

- List the names and comments of all clients who have viewed a property for rent.

$(\Pi_{clientNo,\ fName,\ lName}(Client)) \bowtie (\Pi_{clientNo,\ propertyNo,\ comment}(Viewing))$

| clientNo | fName | lName | propertyNo | comment |
|---|---|---|---|---|
| CR76 | John | Kay | PG4 | too remote |
| CR56 | Aline | Stewart | PA14 | too small |
| CR56 | Aline | Stewart | PG4 | |
| CR56 | Aline | Stewart | PG36 | |
| CR62 | Mary | Tregear | PA14 | no dining room |

# Outer Join

- To display rows in the result that do not have matching values in the join column, use Outer join.
- R ⟕ S
  - (Left) outer join is join in which tuples from R that do not have matching values in common columns of S are also included in result relation.

# Outer Join – An Example

- Produce a status report on property viewings.

$\Pi_{propertyNo,\ street,\ city}$(PropertyForRent) ⟕ Viewing

| propertyNo | street | city | clientNo | viewDate | comment |
|---|---|---|---|---|---|
| PA14 | 16 Holhead | Aberdeen | CR56 | 24-May-01 | too small |
| PA14 | 16 Holhead | Aberdeen | CR62 | 14-May-01 | no dining room |
| PL94 | 6 Argyll St | London | null | null | null |
| PG4 | 6 Lawrence St | Glasgow | CR76 | 20-Apr-01 | too remote |
| PG4 | 6 Lawrence St | Glasgow | CR56 | 26-May-01 | |
| PG36 | 2 Manor Rd | Glasgow | CR56 | 28-Apr-01 | |
| PG21 | 18 Dale Rd | Glasgow | null | null | null |
| PG16 | 5 Novar Dr | Glasgow | null | null | null |

# Semijoin

- $R \rhd_F S$
  - Defines a relation that contains the tuples of R that participate in the join of R with S.
- Can rewrite Semijoin using Projection and Join:
  - $R \rhd_F S = \Pi_A(R \bowtie_F S)$

# Semijoin – An Example

- List complete details of all staff who work at the branch in Glasgow.

$Staff \rhd_{Staff.branchNo=Branch.branchNo}(\sigma_{city='Glasgow'}(Branch))$

| staffNo | fName | lName | position | sex | DOB | salary | branchNo |
|---|---|---|---|---|---|---|---|
| SG37 | Ann | Beech | Assistant | F | 10-Nov-60 | 12000 | B003 |
| SG14 | David | Ford | Supervisor | M | 24- Mar-58 | 18000 | B003 |
| SG5 | Susan | Brand | Manager | F | 3-Jun-40 | 24000 | B003 |

# Division

- $R \div S$
  - Defines a relation over the attributes C that consists of set of tuples from R that match combination of *every* tuple in S.

- Expressed using basic operations:

  $T_1 \leftarrow \Pi_C(R)$

  $T_2 \leftarrow \Pi_C((S \times T_1) - R)$

  $T \leftarrow T_1 - T_2$

# Division – An Example

- Identify all clients who have viewed all properties with three rooms.

  $(\Pi_{clientNo,\ propertyNo}(Viewing)) \div$
  $(\Pi_{propertyNo}(\sigma_{rooms\ =\ 3}\ (PropertyForRent)))$

$\Pi_{clientNo,propertyNo}(Viewing)$

| clientNo | propertyNo |
|---|---|
| CR56 | PA14 |
| CR76 | PG4 |
| CR56 | PG4 |
| CR62 | PA14 |
| CR56 | PG36 |

$\Pi_{propertyNo}(\sigma_{rooms=3}(PropertyForRent))$

| propertyNo |
|---|
| PG4 |
| PG36 |

RESULT

| clientNo |
|---|
| CR56 |