# Relational Algebra Part Two

Database Management - CIS 386 01 FA17

By: Dr. Aos Mulahuwaish

## **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
  - Natural join
  - Outer join

### 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.

# **Example - Natural join**

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

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

clientNo	fName	IName	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.

# **Example - Left Outer join**

Produce a status report on property viewings.

Π<sub>propertyNo, street, city</sub>(PropertyForRent) ⋈ Viewing

propertyNo s	street	city	clientNo	viewDate	comment
PA14 10 PL94 6 PG4 6 PG4 6 PG36 2 PG21 18	16 Holhead 16 Holhead 6 Argyll St 6 Lawrence St 6 Lawrence St 2 Manor Rd 18 Dale Rd 5 Novar Dr	Aberdeen Aberdeen London Glasgow Glasgow Glasgow Glasgow Glasgow	CR56 CR62 null CR76 CR56 CR56 null null	24-May-01 14-May-01 null 20-Apr-01 26-May-01 28-Apr-01 null	too small no dining room null too remote null null

### **Division**

- R ÷ 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:

$$T_{1} \leftarrow \Pi_{C}(R)$$

$$T_{2} \leftarrow \Pi_{C}((S \times T_{1}) - R)$$

$$T \leftarrow T_{1} - T_{2}$$

# **Example - Division**

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

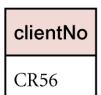
```
(\Pi_{\text{clientNo, propertyNo}}(\text{Viewing})) \div (\Pi_{\text{propertyNo}}(\sigma_{\text{rooms = 3}}(\text{PropertyForRent})))
```

$\Pi_{\text{clientNo,propertyNo}}$	(	V	i/	ew	/in	g	)
------------------------------------	---	---	----	----	-----	---	---

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

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

propertyNo	
PG4 PG36	



RESULT

# **Aggregate Operations**

- 3 AL(R)
  - Applies aggregate function list, AL, to R to define a relation over the aggregate list.
  - AL contains one or more (<aggregate\_function>,<attribute>) pairs .
- Main aggregate functions are: COUNT, SUM, AVG, MIN, and MAX.

# Example – Aggregate Operations

 How many properties cost more than £350 per month to rent?

```
\rho_R(myCount) \mathfrak{I}_{COUNT \text{ propertyNo}} (\sigma_{rent > 350} (PropertyForRent))
```

