CITS1402 Relational Database Management Systems

Week 3—Relational Algebra

Sem1/2024

Dr Mehwish Nasim/ CSSE

cits1402-sem1-pmc@uwa.edu.au

CITS1402

Week3

Contents

Relational Algebra

Meaning of the term relational completeness.

How to form queries in relational algebra.

Objectives

Meaning of the term relational completeness.

How to form queries in relational algebra.

How to form queries in tuple relational calculus.

How to form queries in domain relational calculus.

Categories of relational DML.

Introduction

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.

Chapter 5 - Objectives

Meaning of the term relational completeness.

How to form queries in relational algebra.

How to form queries in tuple relational calculus.

How to form queries in domain relational calculus.

Categories of relational DML.

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.

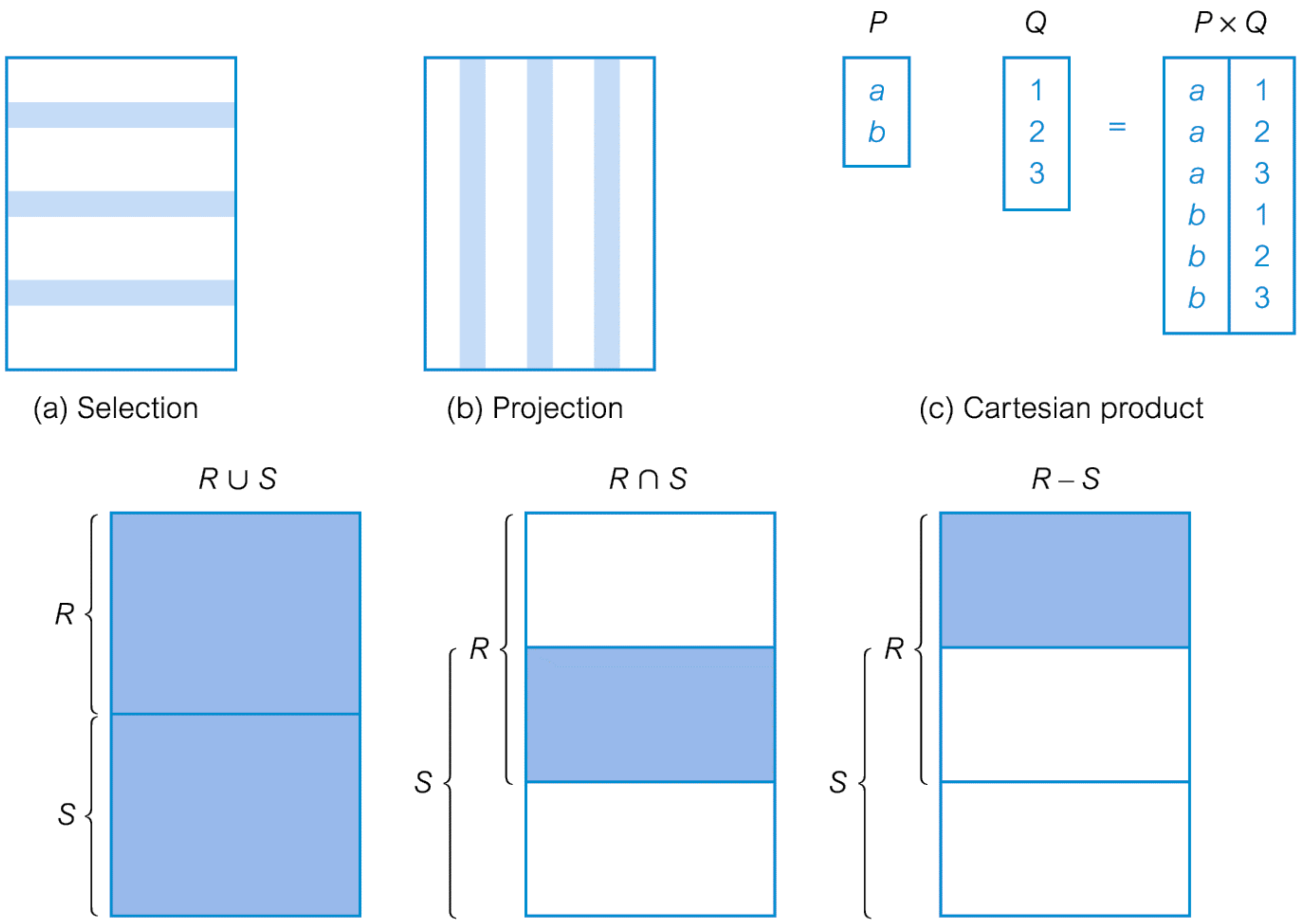
Relational Algebra

Selection,	σ
Projection,	Π
Cartesian product,	X
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.

Relational Algebra Operations



(a) Selection

(b) Projection

(c) Cartesian product

(d) Union

(e) Intersection

(f) Set difference

Relational Algebra Operations

T

A	B
a	1
b	2

U

B	C
1	x
1	y
3	z

$T \bowtie U$

A	B	C
a	1	x
a	1	y

(g) Natural join

$T \triangleright_B U$

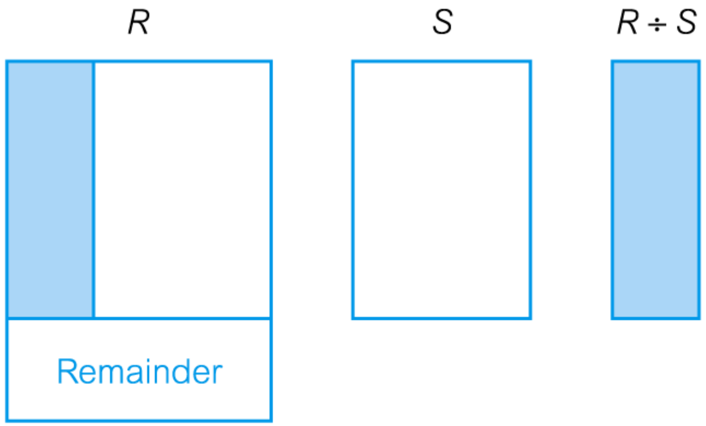
A	B
a	1

(h) Semijoin

$T ⟕_C U$

A	B	C
a	1	x
a	1	y
b	2	

(i) Left Outer join



(j) Divis on (shaded area)

V

A	B
a	1
a	2
b	1
b	2
c	1

W

B
1
2

$V \div W$

A
a
b

Example of division

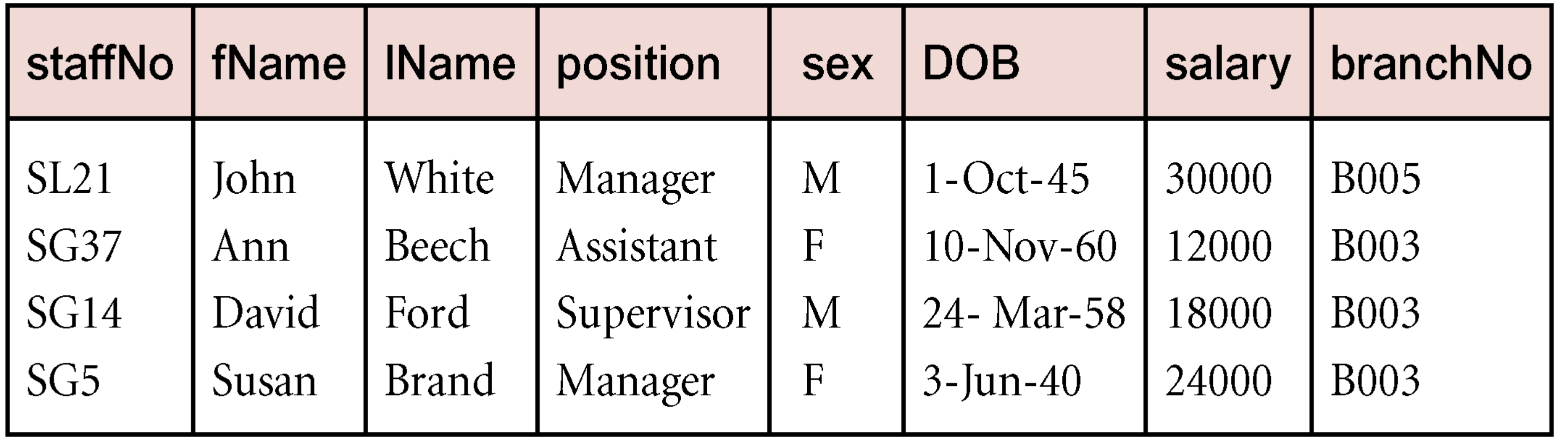
Selection (or Restriction)

$\sigma_{predicate}$ (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*).

Example - Selection (or Restriction)

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



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

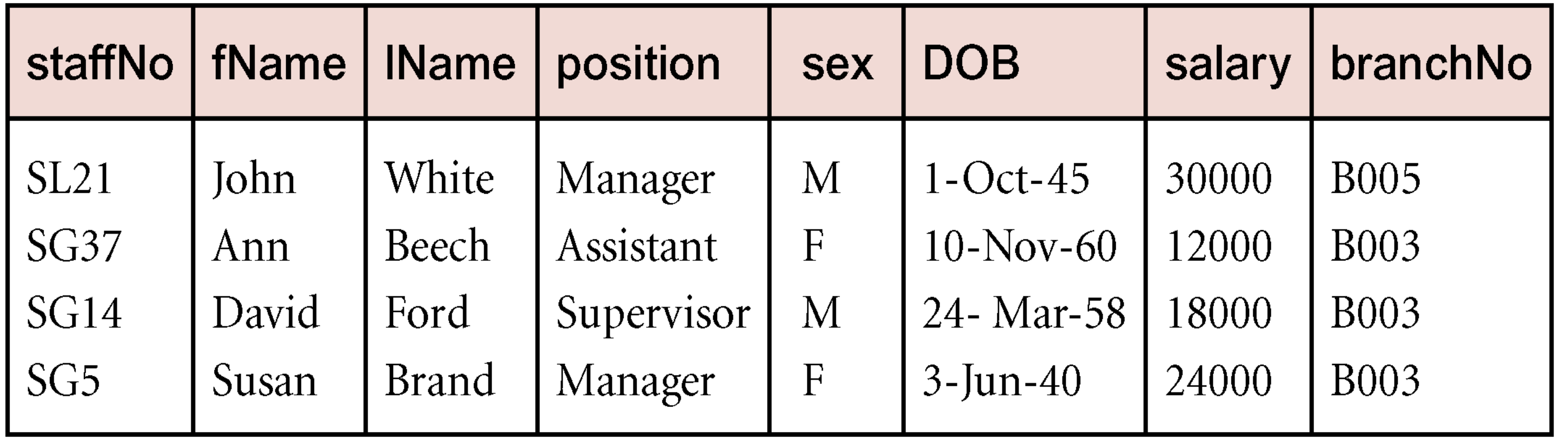
DreamHome Database

Branch	(branchNo, street, city, postcode)
Staff	(staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent	(propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)
Client	(clientNo, fName, lName, telNo, prefType, maxRent, email)
PrivateOwner	(ownerNo, fName, lName, address, telNo, email, password)
Viewing	(clientNo, propertyNo, viewDate, comment)
Registration	(clientNo, branchNo, staffNo, dateJoined)

Example - Selection (or Restriction)

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

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



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

$\Pi_{\text{col1}, \ldots, \text{coln}}(R)$

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.

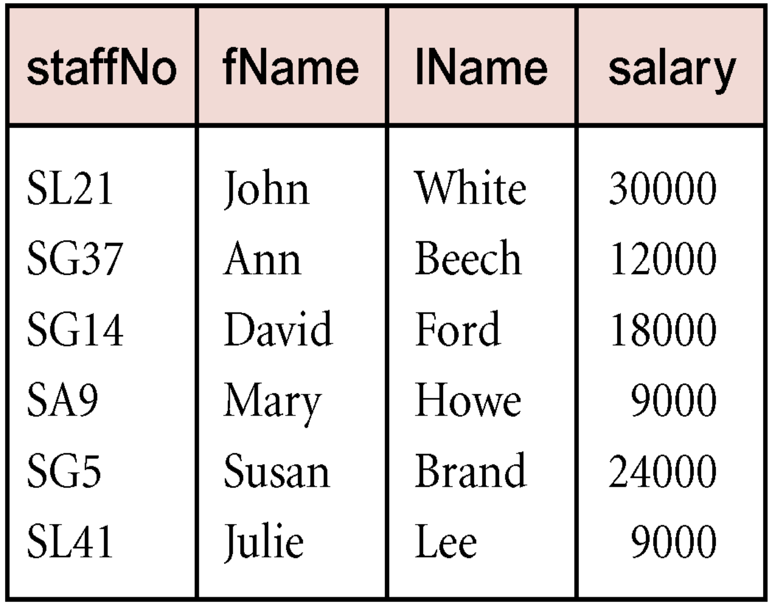
Example - Projection

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

Example - Projection

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

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



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.

Example - Union

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

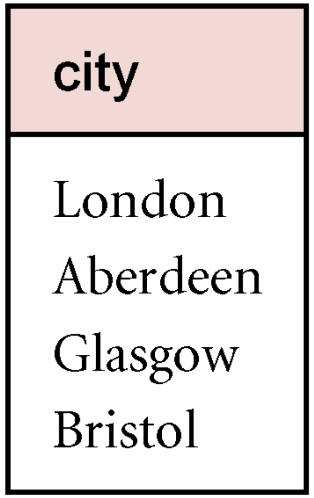
DreamHome Database

Branch	(branchNo, street, city, postcode)
Staff	(staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent	(propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)
Client	(clientNo, fName, lName, telNo, prefType, maxRent, email)
PrivateOwner	(ownerNo, fName, lName, address, telNo, email, password)
Viewing	(clientNo, propertyNo, viewDate, comment)
Registration	(clientNo, branchNo, staffNo, dateJoined)

Example - Union

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

$$\Pi_{city}(\text{Branch}) \cup \Pi_{city}(\text{PropertyForRent})$$



city
London
Aberdeen
Glasgow
Bristol

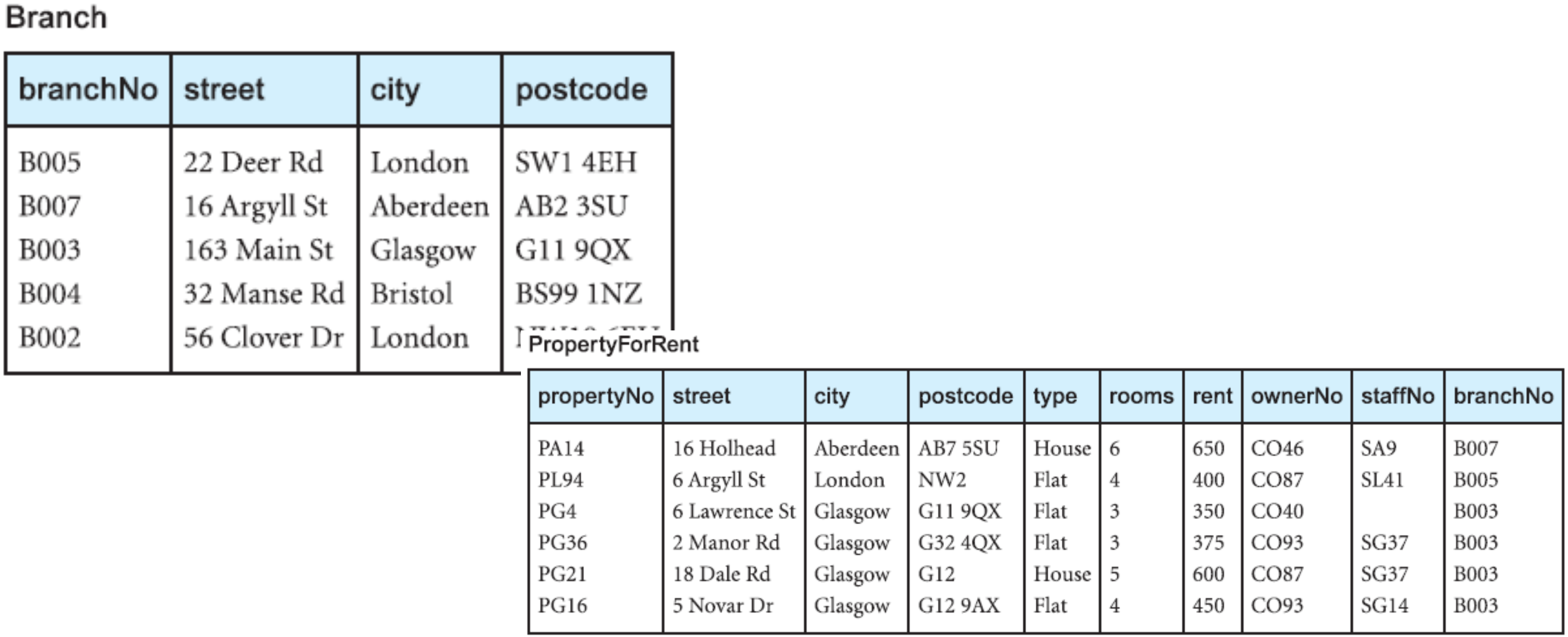
Set Difference

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.**

Example - Set Difference

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



Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

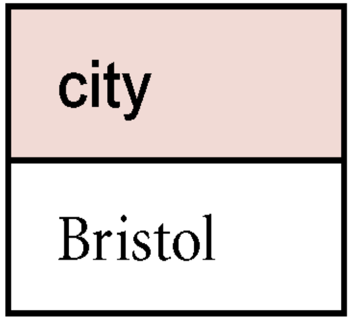
PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Example - Set Difference

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

Π_{city}**(Branch) – Π_{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.

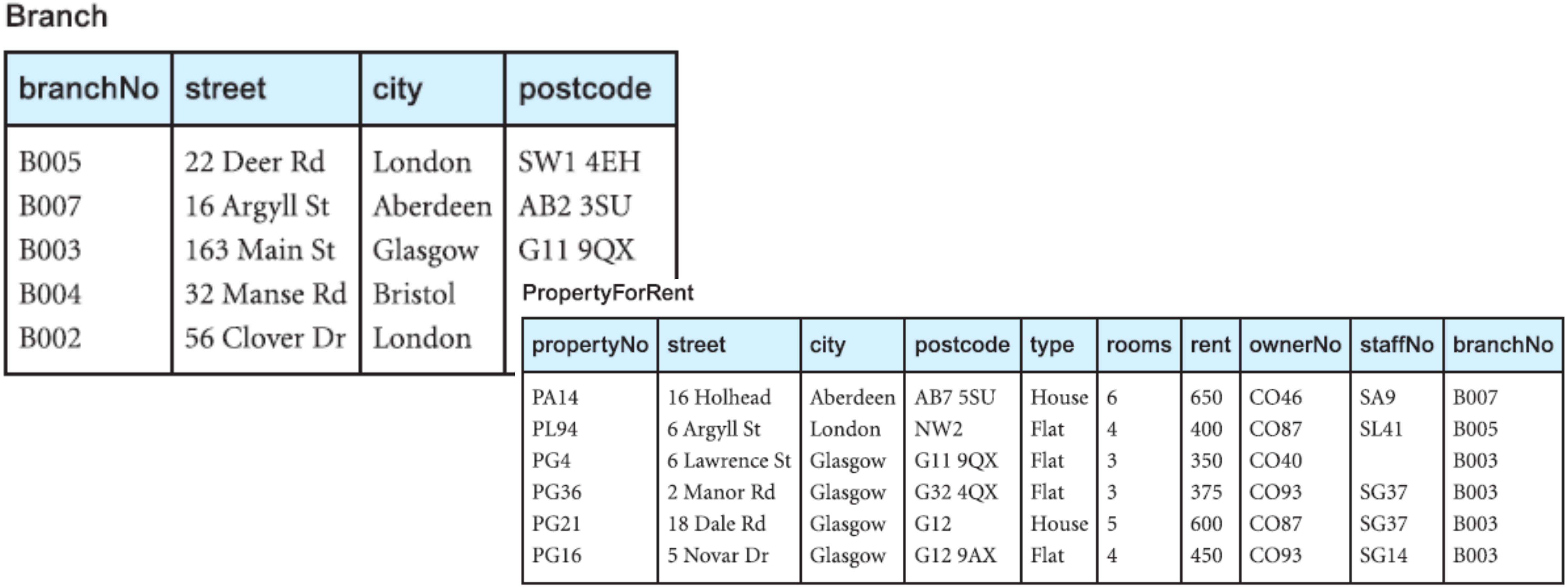
R and S must be union-compatible.

Expressed using basic operations:

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

Example - Intersection

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



Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	
B002	56 Clover Dr	London	

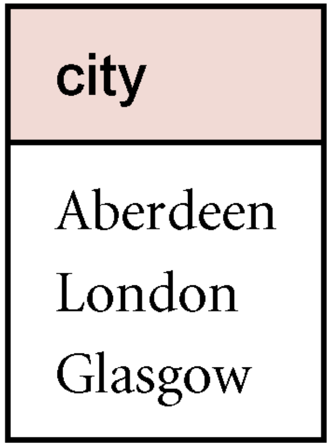
PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Example - Intersection

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

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



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.

Example - Cartesian product

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

DreamHome Database

Branch	(branchNo, street, city, postcode)
Staff	(staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent	(propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)
Client	(clientNo, fName, lName, telNo, prefType, maxRent, email)
PrivateOwner	(ownerNo, fName, lName, address, telNo, email, password)
Viewing	(clientNo, propertyNo, viewDate, comment)
Registration	(clientNo, branchNo, staffNo, dateJoined)

Example - Cartesian product

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

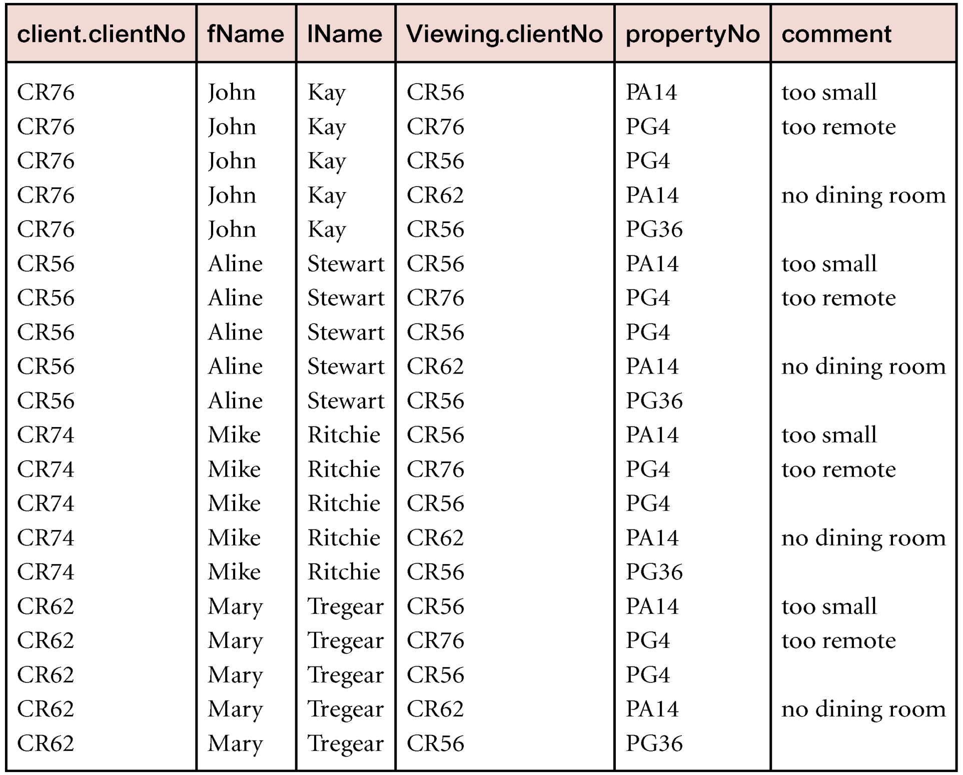
client.clientNo	fName	lName	Viewing.clientNo	propertyNo	comment
CR76	John	Kay	CR56	PA14	too small
CR76	John	Kay	CR76	PG4	too remote
CR76	John	Kay	CR56	PG4	
CR76	John	Kay	CR62	PA14	no dining room
CR76	John	Kay	CR56	PG36	
CR56	Aline	Stewart	CR56	PA14	too small
CR56	Aline	Stewart	CR76	PG4	too remote
CR56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

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	

Example - Cartesian product

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

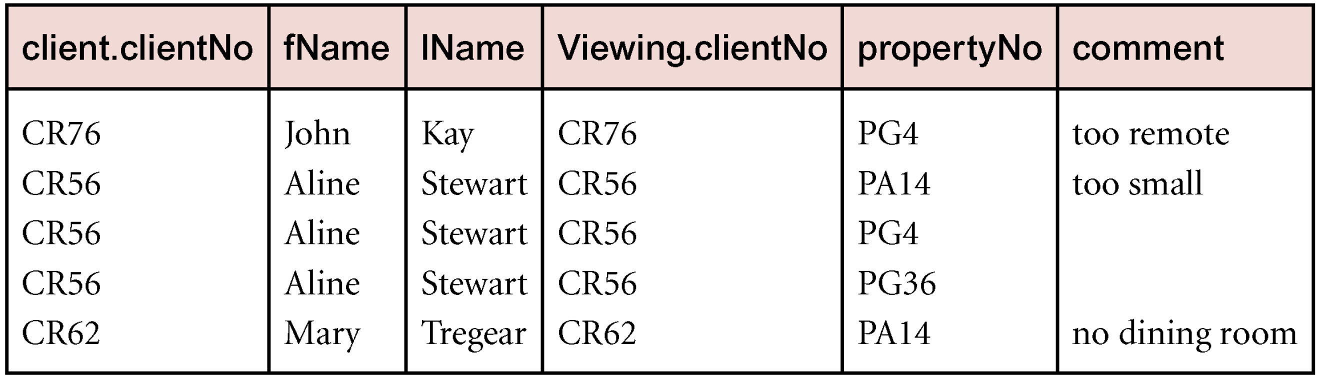
$$(\Pi_{\text{clientNo, fName, lName}}(\text{Client})) \text{ X } (\Pi_{\text{clientNo, propertyNo, comment}}(\text{Viewing}))$$



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

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

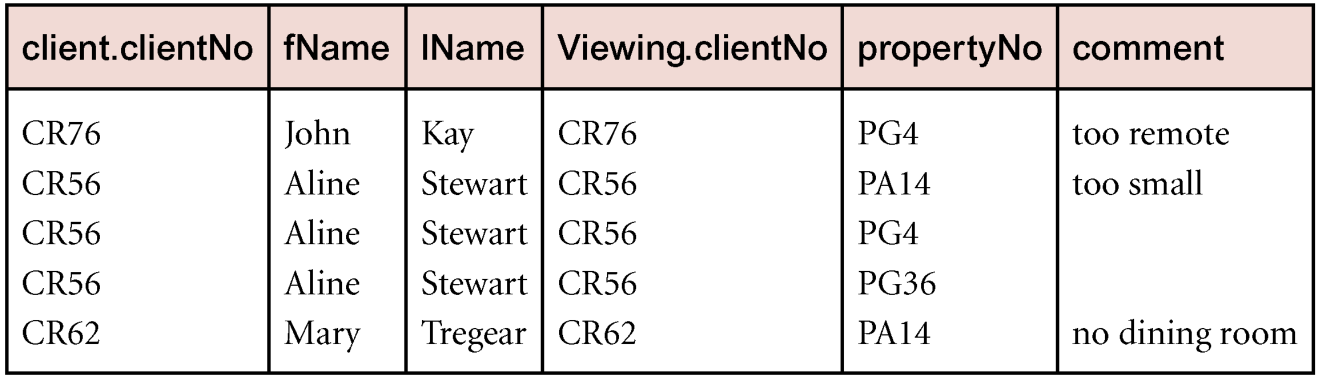


client.clientNo	fName	lName	Viewing.clientNo	propertyNo	comment
CR76	John	Kay	CR76	PG4	too remote
CR56	Aline	Stewart	CR56	PA14	too small
CR56	Aline	Stewart	CR56	PG4	
CR56	Aline	Stewart	CR56	PG36	
CR62	Mary	Tregear	CR62	PA14	no dining room

Cartesian product and Selection

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

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



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.

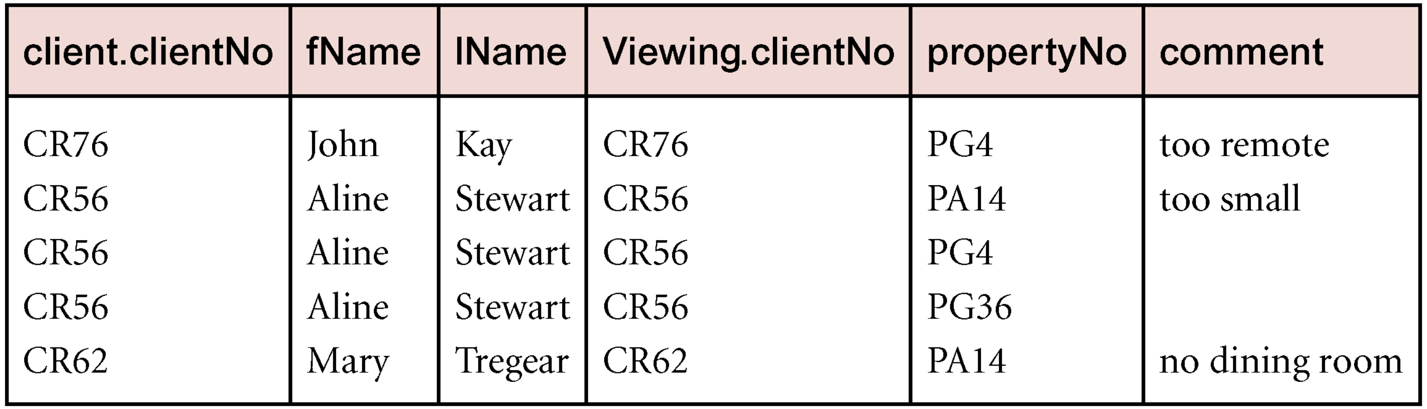
$$R \bowtie_F S = \sigma_F(R \times S)$$

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.

Example - Equijoin

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



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

Example - Equijoin

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

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

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

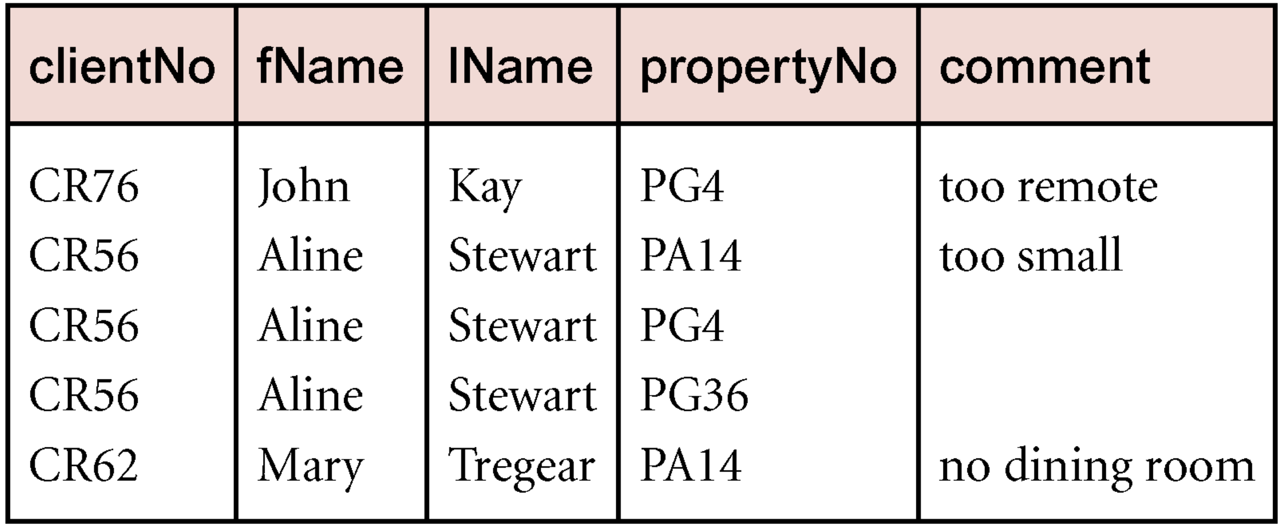
$R \bowtie S$

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.

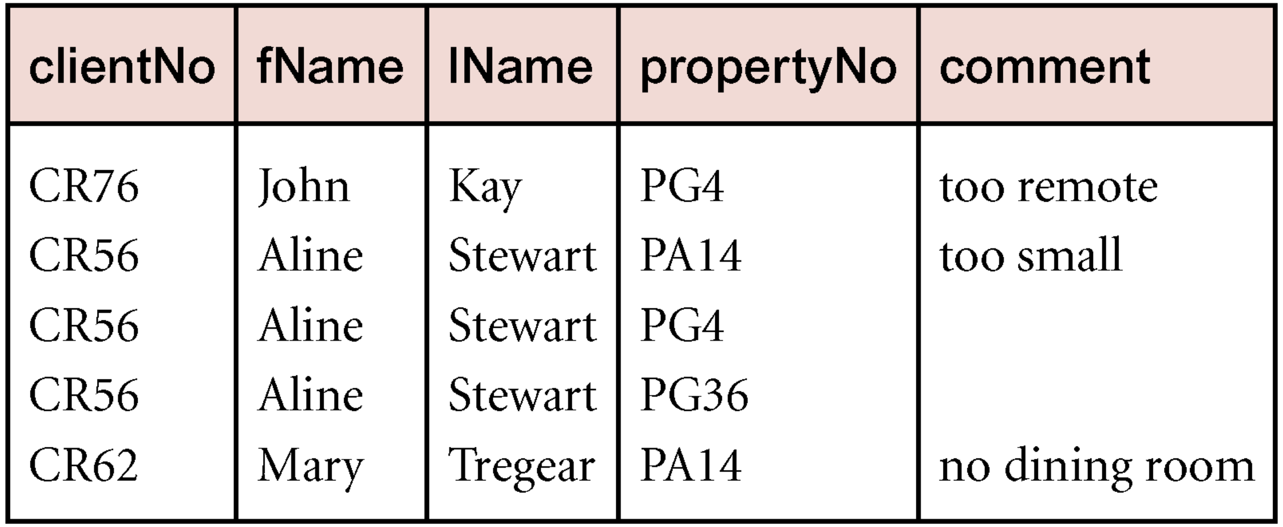


clientNo	fName	lName	propertyNo	comment
CR76	John	Kay	PG4	too remote
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR56	Aline	Stewart	PG36	
CR62	Mary	Tregear	PA14	no dining room

Example - Natural join

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

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



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

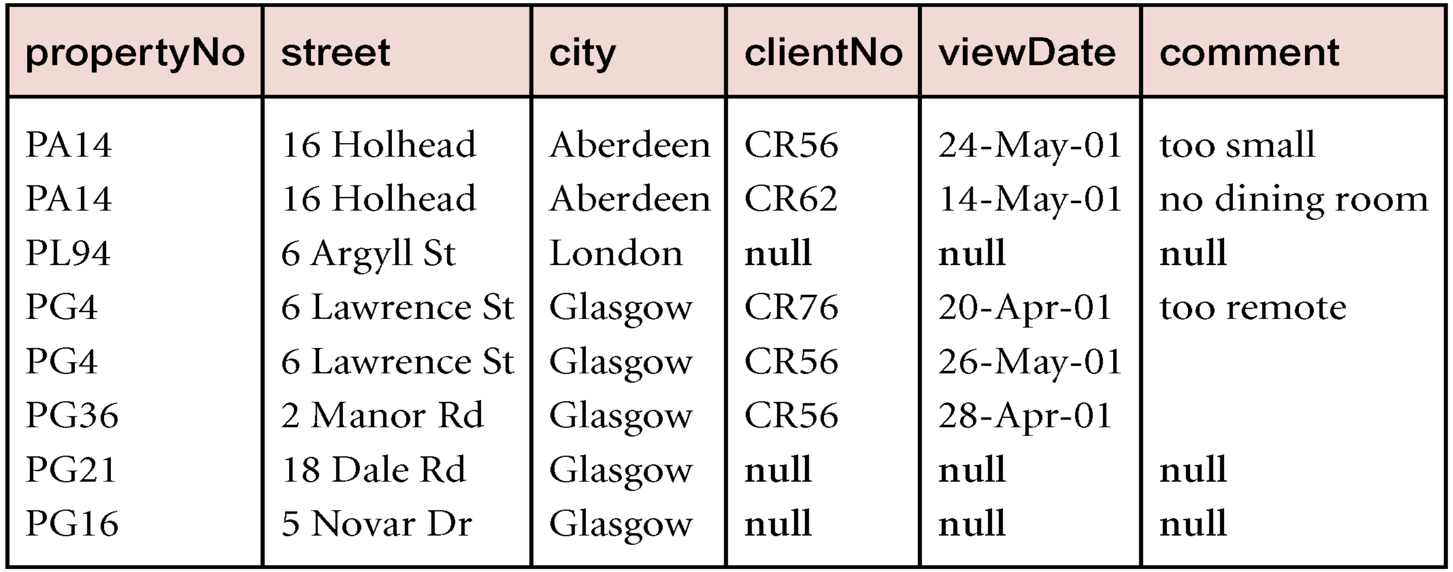
To display rows in the result that do not have matching values in the join column, use Outer join.

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.

Example - Left Outer join

Produce a status report on property viewings.

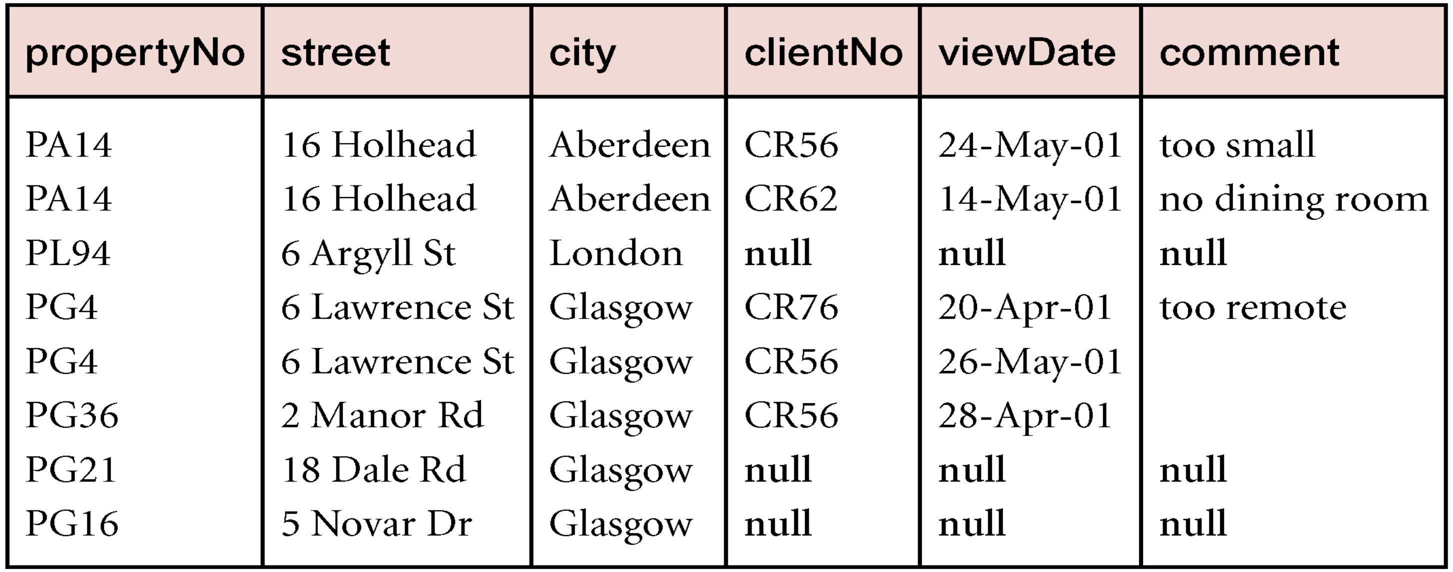


propertyNo	street	city	clientNo	viewDate	comment
PA14	16 Holhead	Aberdeen	CR56	24-May-01	too small
PA14	16 Holhead	Aberdeen	CR62	14-May-01	no dining room
PL94	6 Argyll St	London	**null**	**null**	**null**
PG4	6 Lawrence St	Glasgow	CR76	20-Apr-01	too remote
PG4	6 Lawrence St	Glasgow	CR56	26-May-01	
PG36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

Example - Left Outer join

Produce a status report on property viewings.

$\Pi_{\text{propertyNo, street, city}}$(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

$R \triangleright_F S$

Defines a relation that contains the tuples of R that participate in the join of R with S.

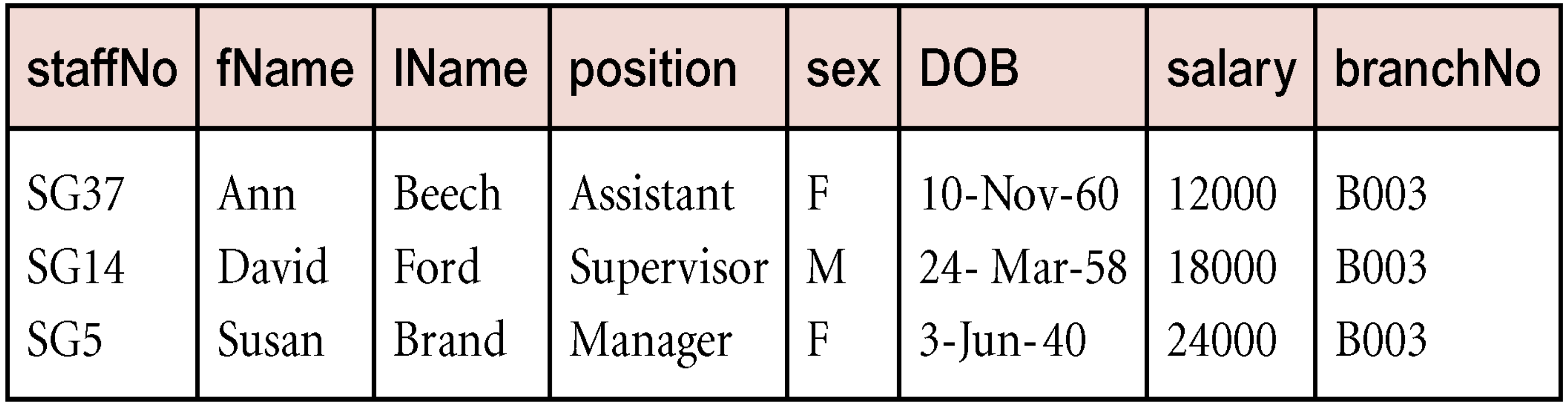
Can rewrite Semijoin using Projection and Join:

$$R \triangleright_F S = \Pi_A(R \bowtie_F S)$$

where “A” are only attributes from R

Example - Semijoin

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

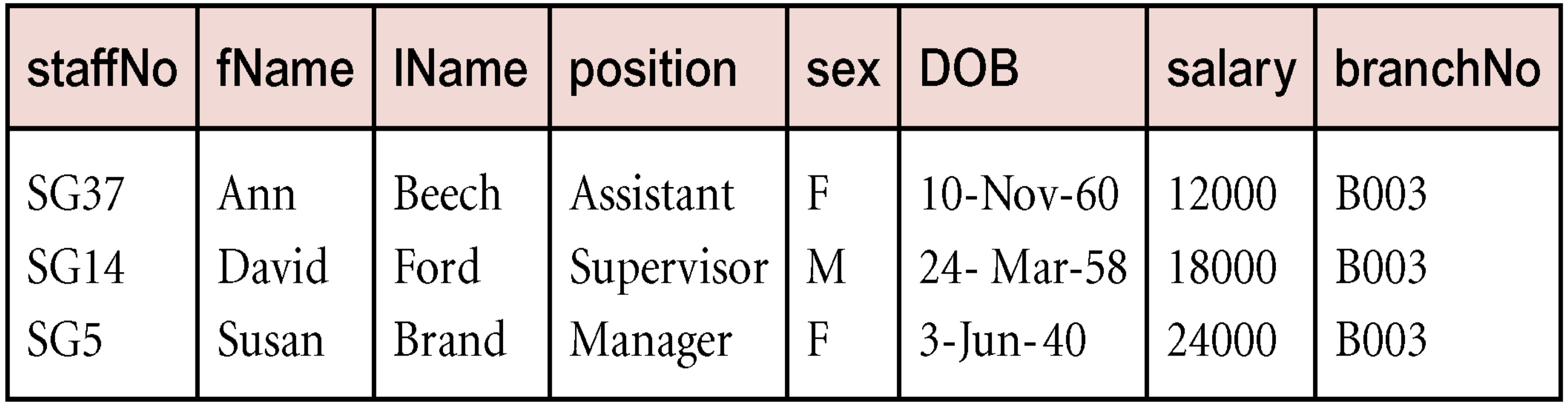


staffNo	fName	lName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24- Mar-58	18000	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003

Example - Semijoin

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

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



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.

C = A – B, where C is the set attributes of R that are not attributes of S

Good for “all” type queries

Expressed using basic operations:

$T_1 \leftarrow \Pi_C(R)$

$T_2 \leftarrow \Pi_C((S \text{ X } T_1) - R)$

$T \leftarrow T_1 - T_2$

Example - Division

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

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

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

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

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

propertyNo
PG4
PG36

RESULT

clientNo
CR56

Aggregate Operations

$\mathfrak{I}_{AL}(R)$

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.

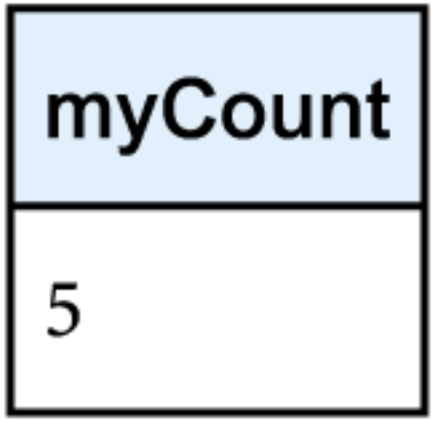
Example – Aggregate Operations

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

Example – Aggregate Operations

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

$$\rho_{R}(\text{myCount})\ \mathfrak{I}_{\text{COUNT propertyNo}}\ (\sigma_{\text{rent} > 350}\ (\text{PropertyForRent}))$$



myCount
5

(a)

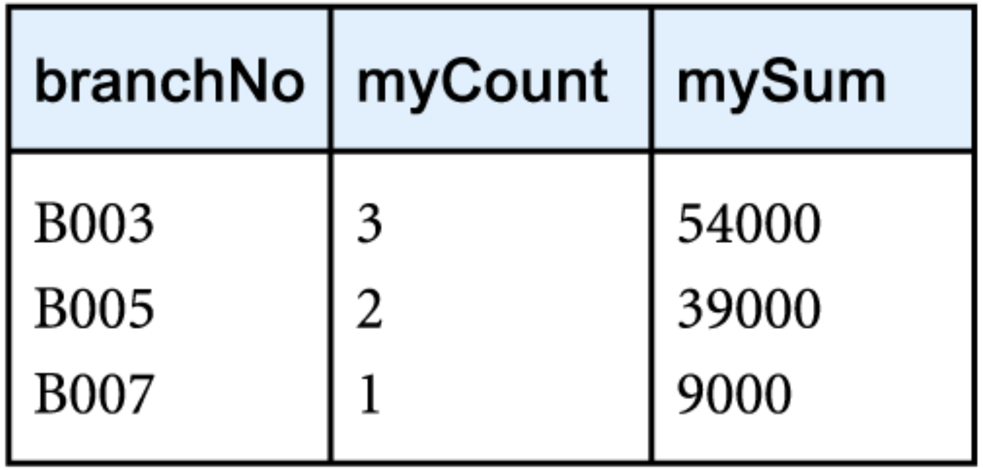
Grouping Operation

$_{GA}\mathfrak{I}_{AL}(R)$

- Groups tuples of R by grouping attributes, GA, and then applies aggregate function list, AL, to define a new relation.
- AL contains one or more (<aggregate_function>, <attribute>) pairs.
- Resulting relation contains the grouping attributes, GA, along with results of each of the aggregate functions.

Example – Grouping Operation

Find the number of staff working in each branch and the sum of their salaries.

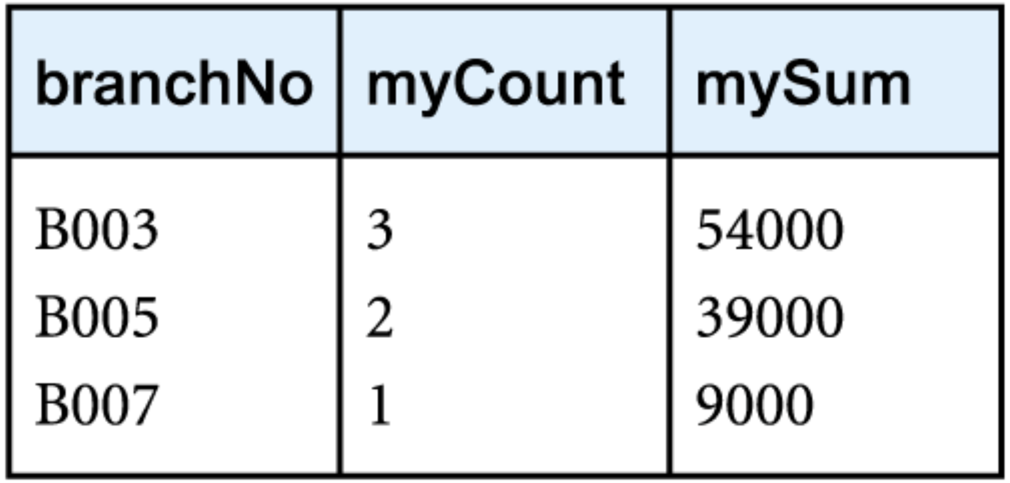


branchNo	myCount	mySum
B003	3	54000
B005	2	39000
B007	1	9000

Example – Grouping Operation

Find the number of staff working in each branch and the sum of their salaries.

$$\rho_{R}(\text{branchNo, myCount, mySum})\ {}_{\text{branchNo}}\Im_{\text{COUNT staffNo, SUM salary}}\ (\text{Staff})$$



branchNo	myCount	mySum
B003	3	54000
B005	2	39000
B007	1	9000

Chapter 5 - Objectives

Meaning of the term relational completeness.

How to form queries in relational algebra.

How to form queries in tuple relational calculus.

How to form queries in domain relational calculus.

Categories of relational DML.