University of Basrah
College of CS & IT
Assist Lec. Mohammed Dheyaa Badr

Objectives

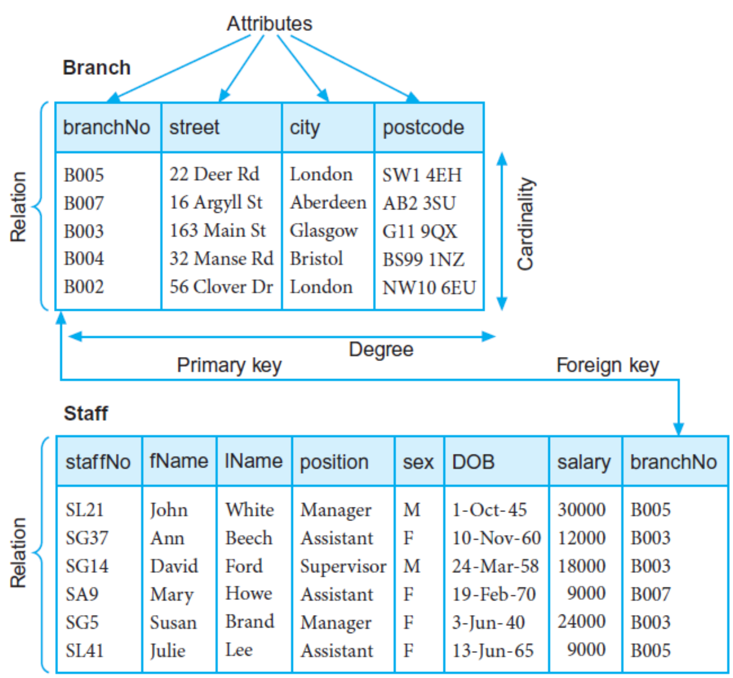
In this Lecture you will learn:

- The basic terminology.
- How to form queries in the relational algebra.
- The categories of relational Data Manipulation Languages (DMLs).

1. Terminology

1.1 Relational Data Structure

Relation: A relation is a table with columns and rows.
Attribute: An attribute is a named column of a relation.
Domain: A domain is the set of allowable values for one or more attributes.
Tuple: A tuple is a row of a relation.
Degree: The degree of a relation is the number of attributes it contains.
Cardinality: The cardinality of a relation is the number of tuples it contains.
Relational database: A collection of normalized relations with distinct relation names.
Null: Represents a value for an attribute that is currently unknown or is not applicable for this tuple.



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

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
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Fig. 1: Basic Structure of Rational Model.

Superkey: An attribute, or set of attributes, that uniquely identifies a tuple within a relation.

A superkey uniquely identifies each tuple within a relation. However, a superkey may contain additional attributes that are not necessary for unique identification, and we are interested in identifying superkeys that contain only the minimum number of attributes necessary for unique identification.

Candidate key

A candidate key *K* for a relation *R* has two properties:

- *Uniqueness.* In each tuple of *R*, the values of *K* uniquely identify that tuple.
- *Irreducibility.* No proper subset of *K* has the uniqueness property.

Primary key: The candidate key that is selected to identify tuples uniquely within the relation.

Because a relation has no duplicate tuples, it is always possible to identify each row uniquely. This means that a relation always has a primary key. In the worst case, the entire set of attributes could serve as the primary key, but usually some smaller subset is sufficient to distinguish the tuples.

The candidate keys that are not selected to be the primary key are called **alternate keys**. For the Branch relation, if we choose branchNo as the primary key, postcode would then be an alternate key. For the Viewing relation, there is only one candidate key, comprising clientNo and propertyNo, so these attributes would automatically form the primary key.

Foreign key: An attribute, or set of attributes, within one relation that matches the candidate key of some (possibly the same) relation.

When an attribute appears in more than one relation, its appearance usually rep- resents a relationship between tuples of the two relations. For example, the inclusion of branchNo in both the Branch and Staff relations is quite deliberate and links each branch to the details of staff working at that branch. In the Branch relation, branchNo is the primary key. However, in the Staff relation, the branchNo attribute exists to match staff to the branch office they work in. In the Staff relation, branchNo is a foreign key. We say that the attribute branchNo in the Staff relation **targets** the primary key attribute branchNo in the **home relation,** Branch. These common attributes play an important data manipulation, as we see in the next chapter.

1.2 Representing Relational Database Schemas

A relational database consists of any number of normalized relations. The relational schema for part of the ***DreamHome*** case study in APENDEX is:

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)

The common convention for representing a relation schema is to give the name of the relation followed by the attribute names in parentheses. Normally, the primary key is underlined.

1.3 Integrity Constraints

There are constraints (called **domain constraints**) that form restrictions on the set of values allowed for the attributes of relations. In addition, there are two important **integrity rules**, which are constraints or restrictions that apply to all instances of the database. The two principal rules for the relational model are known as entity integrity and **referential integrity**.

1.3.1 Entity Integrity

In a base relation, no attribute of a primary key can be null.

1.3.2 Referential Integrity

If a foreign key exists in a relation, either the foreign key value must match a candidate key value of some tuple in its home relation or the foreign key value must be wholly null.

1.3.3 General Integrity

Additional rules specified by the users or database administrators of a database that define or constrain some aspect of the enterprise.

2. The Relational Algebra

The relational algebra is a theoretical language with operations that work on one or more relations to define another relation without changing the original relation(s). Thus both the operands and the results are relations, and so the output from one operation can become the input to another operation.

2.1 Unary Operations

We start the discussion of the relational algebra by examining the two unary operations: **Selection** and **Projection.**

2.1.1 Selection

The Selection (or Restriction) operation works on a single relation ***R*** and defines a relation that contains only those tuples of ***R*** that satisfy the specified **condition** (or ***predicate***).

$$\sigma_{predicate}\ (R)$$

Listing 1 : List all ***staff*** with a salary greater than £10000.

Algebraic form :

$$\sigma_{Salary > 1000}\ (Staff)$$

MySQL form:

Analysis: Here, the input relation is Staff and the predicate is salary > 10000. The Selection operation defines a relation containing only those Staff tuples with a salary greater than £10000

Input Relations

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
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Result Relations

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

2.1.2 Projection

The Projection operation 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.

$$\Pi_{a_1, \ldots, a_n}(\mathbf{R})$$

Listing 2: Produce a list of salaries for all staff, showing only the *staffNo, fName, lName,* and *salary* details.

Algebraic form:

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

Analysis: The Projection operation defines a relation that contains only the designated Staff attributes *staffNo, fName, lName*, and *salary* in the specified order.

Input Relations

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
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Result Relations

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

2.2 Set operation

The Selection and Projection operations extract information from only one relation. There are obviously cases where we would like to combine information from several relations.

2.2.1 Union

The 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.

$$\mathbf{R} \cup \mathbf{S}$$

Listing 3: List all ***cities*** where there is either a ***branch*** office or a ***property for rent***.

Algebraic form:

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

Analysis:

To produce union-compatible relations, we first use the Projection operation to project the *Branch* and *PropertyForRent* relations over the attribute city, eliminating duplicates where necessary.

Input Relations

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

Result Relations

city
London
Aberdeen
Glasgow
Bristol

2.2.2 Set Difference

The Set difference operation defines a relation consisting of the tuples that are in relation R, but not in S. R and S must be union-compatible.

R – S

Listing 4: List all cities where there is a branch office but no properties for rent.

Algebraic form:

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

Analysis:

As in the previous example, we produce union-compatible relations by projecting the *Branch* and *PropertyForRent* relations over the attribute city. We then use the Set difference operation to combine these new relations to produce the result shown in Result's figure.

Input Relations

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

Result Relations

city
Bristol

2.2.3 Intersection

The Intersection operation defines a relation consisting of the set of all tuples that are in both R and S. R and S must be union-compatible.

R ∩ S

Listing 5: List all cities where there is both a branch office and at least one property for rent.

Algebraic form:

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

Analysis:

As in the previous example, we produce union-compatible relations by projecting the Branch and PropertyForRent relations over the attribute city. We then use the Intersection operation to combine these new relations to produce the result shown in result's figure.

Input Relations

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

Result Relations

city
Aberdeen
London
Glasgow

2.2.4. Cartesian product

The Cartesian product operation defines a relation that is the concatenation of every tuple of relation *R* with every tuple of relation *S*.

- The Cartesian product operation multiplies two relations to define another relation consisting of all possible pairs of tuples from the two relations. Therefore, if one relation has *I* tuples and *N* attributes and the other has *J* tuples and *M* attributes, the Cartesian product relation will contain ($I * J$) tuples with ($N + M$) attributes.
- It is possible that the two relations may have attributes with the same name. In this case, the attribute names are prefixed with the relation name to maintain the uniqueness of attribute names within a relation.

Listing 6: List the *names* and *comments* of all ***clients*** who have ***viewed*** a property for rent.

Algebraic form:

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

Analysis:

In this above form, this relation contains more information than we require as shown in Result 1. For example, the first tuple of this relation contains different *clientNo* values. To obtain the required list, we need to carry out a Selection operation on this relation to extract those tuples where *Client.clientNo* = *Viewing.clientNo*. thus it becomes:

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

Input Relations

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	

Result Relations 1:

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	

Result Relations 2:

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

2.3 Join Operations

- Typically, we want only combinations of the Cartesian product that satisfy certain conditions and so we would normally use a **Join operation** instead of the Cartesian product operation.
- The Join operation, which combines two relations to form a new relation.
- Join is a derivative of Cartesian product.

- Join is equivalent to performing a Selection operation over the Cartesian product of the two operand relations.
- Join is one of the most difficult operations to implement efficiently in an RDBMS and is one of the reasons why relational systems have intrinsic performance problems.

- There are various forms of the Join operation, each with subtle differences, some more useful than others:
  - Theta join
  - Equijoin (a particular type of Theta join)
  - Natural join
  - Outer join
  - Semijoin

2.3.1 Inner Join

An Inner Join returns only the rows in both tables that match the join condition.

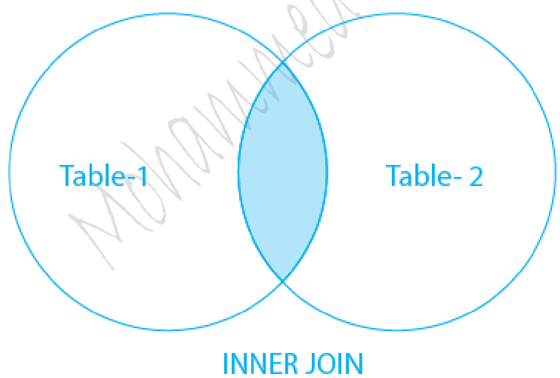


Fig. 2 : Inner Join.

2.3.1.1 Theta Join (Ø Join)

The Theta join operation defines a relation that contains tuples satisfying the predicate (or Condition) F from the Cartesian product of R and S.

$$\mathbf{R} \bowtie_F \mathbf{S}$$

We can rewrite the Theta join in terms of the basic Selection and Cartesian product operations:

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

Listing 6: List all employee names and addresses whom age is greater than the salary

Algebraic form:

$$\Pi_{Name}(Table1) \bowtie_{Table1.age > Table2.Salary} \Pi_{Address}(Table2)$$

Algebraic form:

```
SELECT Table1.Name, Table2.Address
FROM Table1
INNER JOIN Table2
ON Table1.Age > Table2.Salary;
```

Analysis:

To perform a Theta Join, we can join the two tables on the condition that the age in Table1 is greater than the salary in Table2

Input Relations

Table1:

ID	Name	Age
1	Alice	23
2	Bob	28
3	Charlie	32

Table2:

ID	Address	Salary
2	New York	50000
3	Boston	65000
4	San Diego	75000

Result Relations

Name	Address

2.3.1.2 EquiJoin

In the case where the predicate F contains only equality in theta join, the term Equijoin is used instead.

Listing 7: List the names and comments of all clients who have viewed a property for rent.

Algebraic form:

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

SQL form:

```sql
SELECT Client.clientNo, Client.fName, Client.lName, Viewing.propertyNo,
Viewing.comment  FROM Client
INNER JOIN Viewing
ON Client.clientNo = Viewing.clientNo;
```

Analysis:
we used the Cartesian product and Selection operations to obtain this list. However, the same result is obtained using the Equijoin operation.

Input Relations

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	

Result Relations

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

2.3.1.3 Natural join

- The Natural join is 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.
- A Natural Join is a type of Join that matches columns with the same name in both tables.

Listing 8: List the names and comments of all clients who have viewed a property for rent.

Algebraic form:

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

SQL form:

```
SELECT Client.clientNo, Client.fName, Client.lName, Viewing.propertyNo,
Viewing.comment  FROM Client
NATURAL JOIN Viewing
```

Analysis:

We use Equijoin to produce the same result relation, but the resulting relation had two occurrences of the join attribute clientNo. We can use the Natural join to remove one occurrence of the clientNo attribute.

Input Relations

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	

Result Relations

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

Listing 9: List all information of all employee who have use their off day.

Algebraic form:

$$\Pi_{\text{ID,Name,Age}}(\text{Table1}) \bowtie \Pi_{\text{ID,Address,Salary}}(\text{Table2})$$

SQL form:

```
SELECT Table1.ID, Table1.Name, Table1.Age, Table2.Address, Table2.Salary
FROM Table1
NATURAL JOIN Table2;
```

Analysis:

To perform a Natural Join, we can simply use the above query.

Input Relations

Table1:

ID	Name	Age
1	Alice	23
2	Bob	28
3	Charlie	32

Table2:

ID	Address	Salary
2	New York	50000
3	Boston	65000
4	San Diego	75000

Result Relations

ID	Name	Age	Address	Salary
2	Bob	28	New York	50000
3	Charlie	32	Boston	65000

2.3.2 Outer join

Often in joining two relations, a tuple in one relation does not have a matching tuple in the other relation; in other words, there is no matching value in the join attributes.

We may want tuples from one of the relations to appear in the result even when there are no matching values in the other relation. This may be accomplished using the Outer join.

The (left) Outer join is a join in which tuples from R that do not have matching values in the common attributes of S are also included in the result relation.

Missing values in the second relation are set to **null**.

> An Outer Join in DBMS returns all the rows from one table and the matching rows from the other table. If there is no match, NULL values are returned for the missing rows.

2.3.2.1 Left Outer join

The (left) Outer join is a join in which tuples from R that do not have matching values in the common attributes of S are also included in the result relation. Missing values in the second relation are set to null

$$\mathbf{R} ⟕ \mathbf{S}$$

> A Left Outer Join in DBMS returns all the rows from the left table and the matching rows from the right table. If there is no match, NULL values are returned for the missing rows.

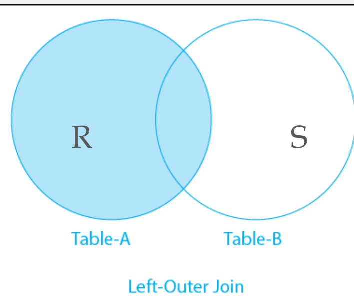


Fig. 3: Left Outer Join

Listing 10: Produce a status report (Location) on property viewings.

Algebraic form:

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

SQL form:

```sql
SELECT PropertyForRent.PropertyNo, PropertyForRent.street, PropertyForRent.city,
Viewing.*
FROM PropertyForRent
LEFT JOIN Viewing;
```

Analysis:

Note that properties PL94, PG21, and PG16 have no viewings, but these tuples are still contained in the result with nulls for the attributes from the Viewing relation.

Input Relations

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

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	

Result Relations

propertyNo	street	city	clientNo	viewDate	comment
PA14	16 Holhead	Aberdeen	CR56	24-May-13	too small
PA14	16 Holhead	Aberdeen	CR62	14-May-13	no dining room
PL94	6 Argyll St	London	**null**	**null**	**null**
PG4	6 Lawrence St	Glasgow	CR76	20-Apr- 13	too remote
PG4	6 Lawrence St	Glasgow	CR56	26-May-13	
PG36	2 Manor Rd	Glasgow	CR56	28-Apr- 13	
PG21	18 Dale Rd	Glasgow	**null**	**null**	**null**
PG16	5 Novar Dr	Glasgow	**null**	**null**	**null**

2.3.2.2 Right Outer join

A Right Outer Join returns all the rows from the right table and the matching rows from the left table. If there is no match, NULL values are returned for the missing rows.

The (Right) Outer join is a join in which tuples from S that do not have matching values in the common attributes of R are also included in the result relation.

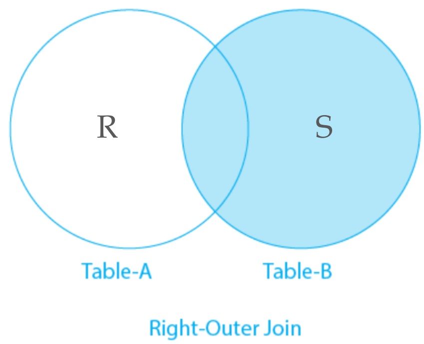


Fig. 4 : Right Outer Join.

Listing 11: List all off days addresses with employees names.

Algebraic form:

$$\Pi_{\text{Name}}\,(\text{Table1}) ⟖ \Pi_{\text{Address}}\,(\text{Table2})$$

SQL form:

```
SELECT  Table1.Name, Table2.Address
FROM Table1
RIGHT JOIN Table2
ON Table1.ID = Table2.ID;
```

Analysis:

To perform a Right Outer Join, we can join the two tables on the ID column.

Input Relations

Table1:

ID	Name	Age
1	Alice	23
2	Bob	28
3	Charlie	32

Table2:

ID	Address	Salary
2	New York	50000
3	Boston	65000
4	San Diego	75000

Result Relations

Name	Address
Bob	New York
Charlie	Boston
NULL	San Diego

2.3.2.2 Full Outer join

A Full Outer Join returns all the rows from both tables and NULL values for the missing rows.

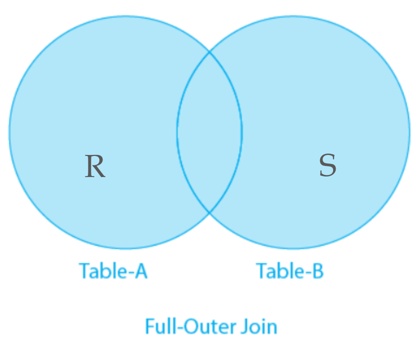


Fig. 4 : Right Outer Join.

Listing 12: List all off days addresses with employees names.

SQL form:

```
SELECT Table1.Name, Table2.Address
FROM Table1
FULL OUTER JOIN Table2
ON Table1.ID = Table2.ID;
```

Analysis:

To perform a Full Outer Join, we can join the two tables on the ID column.

Input Relations

Table1:

ID	Name	Age
1	Alice	23
2	Bob	28
3	Charlie	32

Table2:

ID	Address	Salary
2	New York	50000
3	Boston	65000
4	San Diego	75000

Result Relations

Name	Address
Alice	NULL
Bob	New York
Charlie	Boston
NULL	San Diego

2.3.3 Semi join

The Semijoin operation defines a relation that contains the tuples of R that participate in the join of R with S satisfying the predicate F.

The Semijoin operation performs a join of the two relations and then projects over the attributes of the first operand. One advantage of a Semijoin is that it decreases the number of tuples that need to be handled to form the join.

$$R \triangleright_F S = \Pi_A(R \bowtie_F S) \qquad \text{A is the set of all attributes for R}$$

Listing 13: List complete details of all staff who work at the branch in Glasgow.

Algebraic form:

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

SQL form:

```
SELECT PropertyForRent.PropertyNo, PropertyForRent.street, PropertyForRent.city,
Viewing.*
FROM PropertyForRent
LEFT JOIN Viewing;
```

Analysis:

If we are interested in seeing only the attributes of the Staff relation, we can use the fol lowing Semijoin operation, producing the relation shown in the result.

Input Relations

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

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
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Result Relations

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

A Quick Revision of Joins in DBMS

- To assist you in a good revision, a quick revision on Joins in DBMS is given below:
- Joins in DBMS is used to combine tables.
- There are three types of joins: inner joins, natural joins, and outer joins.
- Inner joins are classified into two types: Theta Join(for relational operators) and Equi Join(for Equality).
- There are three types of outer joins in DBMS: left outer join, right outer join, and full outer join.

- Natural join is only performed when at least one matching attribute exists in both tables.
- No matter the Join condition, a left outer join always returns every row from the left table.
- Regardless of the Join condition, Right Outer Join always returns all rows from the right table.
- Regardless of the join condition, Complete Outer Join always returns all rows from both tables.

Table 5.1 Operations in the relational algebra.

OPERATION	NOTATION	FUNCTION
Selection	$\sigma_{predicate}(R)$	Produces a relation that contains only those tuples of R that satisfy the specified *predicate*.
Projection	$\Pi_{a_1, \ldots, a_n}(R)$	Produces a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.
Union	$R \cup S$	Produces 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.
Set difference	$R - S$	Produces a relation that contains all the tuples in R that are not in S. R and S must be union-compatible.
Intersection	$R \cap S$	Produces a relation that contains all the tuples in both R and S. R and S must be union-compatible.
Cartesian product	$R \times S$	Produces a relation that is the concatenation of every tuple of relation R with every tuple of relation S.
Theta join	$R \bowtie_F S$	Produces a relation that contains tuples satisfying the predicate *F* from the Cartesian product of R and S.
Equijoin	$R \bowtie_F S$	Produces a relation that contains tuples satisfying the predicate *F* (which contains only equality comparisons) from the Cartesian product of R and S.
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.
(Left) Outer join	$R ⟕ S$	A join in which tuples from R that do not have matching values in the common attributes of S are also included in the result relation.
Semijoin	$R \triangleright_F S$	Produces a relation that contains the tuples of R that participate in the join of R with S satisfying the predicate *F*.

Checkpoints:

1- Discuss the differences between the five Join operations: Theta join, Equijoin, Natural join, Outer join, and Semijoin. Give examples to illustrate your answer.

2- Using relational algebra, produce a report of all employees from the IT and planning departments who are born after 1990.

The following tables form part of a database held in an RDBMS:

Employee (<u>empNo</u>, fName, lName, address, DOB, sex, position, deptNo)

Department (<u>deptNo</u>, deptName, mgrEmpNo)

Project (<u>projNo</u>, projName, deptNo)

WorksOn (<u>empNo</u>, <u>projNo</u>, <u>dateWorked</u>, hoursWorked)

where

Employee contains employee details and empNo is the key.

Department contains department details and deptNo is the key. mgrEmpNo identifies the employee who is the manager of the department. There is only one manager for each department.

Project contains details of the projects in each department and the key is projNo (no two departments can run the same project).

WorksOn contains details of the hours worked by employees on each project, and empNo/ projNo/dateWorked form the key.

Formulate the following queries in relational algebra, tuple relational calculus, and domain relational calculus.

A. List all employees.
B. List all the details of employees who are female and born after 1990.
C. List all employees who are not managers and are paid more than $1500.
D. Produce a list of the names and addresses of all employees who work for the IT department.
E. Produce a list of the names of all employees who work on the SCCS project.
F. Produce a complete list of all managers who are due to retire this year, in alphabetical order of surname.

APENDEX:

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

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
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

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

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk

PrivateOwner

ownerNo	fName	lName	address	telNo	eMail	password
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212	jkeogh@lhh.com	********
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419	cfarrel@gmail.com	********
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728	tinam@hotmail.com	********
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025	tony.shaw@ark.com	********

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	

Registration

clientNo	branchNo	staffNo	dateJoined
CR76	B005	SL41	2-Jan-13
CR56	B003	SG37	11-Apr-12
CR74	B003	SG37	16-Nov-11
CR62	B007	SA9	7-Mar-12