ALGEBRA RELACIONAL

A algebra relacional é uma linguagem téorica com operações que trabalham numa ou mais relaçoes para definir outra relação sem mudar a original.

Selection (or Restriction)

$\sigma_{predicate}(R)$	The Selection operation works on a single relation R and defines a relation that contains only those tuples of R that satisfy the specified condition (*predicate*).

Example 4.1 Selection operation

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

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

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 £10,000. The result of this operation is shown in Figure 4.2. More complex predicates can be generated using the logical operators $\wedge$ (AND), $\vee$ (OR) and ~ (NOT).

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

Figure 4.2 Selecting salary > 10000 from the Staff relation.

Projection

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

Example 4.2 Projection operation

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

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

In this example, the Projection operation defines a relation that contains only the designated Staff attributes staffNo, fName, lName, and salary, in the specified order. The result of this operation is shown in Figure 4.3.

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

Figure 4.3
Projecting the Staff relation over the staffNo, fName, lName, and salary attributes.

Union

$R \cup S$ 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.

Example 4.3 Union operation

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

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

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. We then use the Union operation to combine these new relations to produce the result shown in Figure 4.4.

city
London
Aberdeen
Glasgow
Bristol

Figure 4.4
Union based on the city attribute from the Branch and PropertyForRent relations.

Set difference

R – S 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.

Example 4.4 Set difference operation

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

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

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 Figure 4.5.

city
Bristol

Figure 4.5 Set difference based on the city attribute from the Branch and PropertyForRent relations.

Intersection

R ∩ S 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.

Example 4.5 Intersection operation

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

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

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 Figure 4.6.

city
Aberdeen
London
Glasgow

Figure 4.6 Intersection based on city attribute from the Branch and PropertyForRent relations.

Note that we can express the Intersection operation in terms of the Set difference operation:

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

Cartesian product

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

Example 4.6 Cartesian product operation

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

The names of clients are held in the Client relation and the details of viewings are held in the Viewing relation. To obtain the list of clients and the comments on properties they have viewed, we need to combine these two relations:

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

This result of this operation is shown in Figure 4.7. In its present form, this relation contains more information than we require. 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. The complete operation is thus:

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

The result of this operation is shown in Figure 4.8.

Figure 4.7 Cartesian product of reduced Client and Viewing relations.

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

Theta join (θ-join)

$R \bowtie_F S$ The Theta join operation 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 ($<, \leq, >, \geq, =, \neq$).

Example 4.7 Equijoin operation

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

In Example 4.6 we used the Cartesian product and Selection operations to obtain this list. However, the same result is obtained using the Equijoin operation:

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

or

$$\text{Result} \leftarrow \text{TempClient} \bowtie_{\text{TempClient.clientNo = TempViewing.clientNo}} \text{TempViewing}$$

The result of these operations was shown in Figure 4.8.

Natural join

R ⋈ S 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.

Example 4.8 Natural join operation

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

In Example 4.7 we used the Equijoin to produce this list, 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:

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

or

$$\text{Result} \leftarrow \text{TempClient} \bowtie \text{TempViewing}$$

The result of this operation is shown in Figure 4.9.

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

Figure 4.9 Natural join of restricted Client and Viewing relations.

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.

R ⟕ S 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.

Example 4.9 Left Outer join operation

Produce a status report on property viewings.

In this case, we want to produce a relation consisting of the properties that have been viewed with comments and those that have not been viewed. This can be achieved using the following Outer join:

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

The resulting relation is shown in Figure 4.10. 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.

propertyNo	street	city	clientNo	viewDate	comment
PA14	16 Holhead	Aberdeen	CR56	24-May-04	too small
PA14	16 Holhead	Aberdeen	CR62	14-May-04	no dining room
PL94	6 Argyll St	London	null	null	null
PG4	6 Lawrence St	Glasgow	CR76	20-Apr-04	too remote
PG4	6 Lawrence St	Glasgow	CR56	26-May-04	
PG36	2 Manor Rd	Glasgow	CR56	28-Apr-04	
PG21	18 Dale Rd	Glasgow	null	null	null
PG16	5 Novar Dr	Glasgow	null	null	null

Semijoin

$R \triangleright_F S$ The Semijoin operation defines a relation that contains the tuples of R that participate in the join of R with S.

Example 4.10 Semijoin operation

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

If we are interested in seeing only the attributes of the Staff relation, we can use the following Semijoin operation, producing the relation shown in Figure 4.11.

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

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

Figure 4.11 Semijoin of Staff and Branch relations.

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

Example 4.11 Division operation

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

We can use the Selection operation to find all properties with three rooms followed by the Projection operation to produce a relation containing only these property numbers. We can then use the following Division operation to obtain the new relation shown in Figure 4.12.

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

Figure 4.12 Result of the Division operation on the Viewing and PropertyForRent relations.

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

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

clientNo
CR56

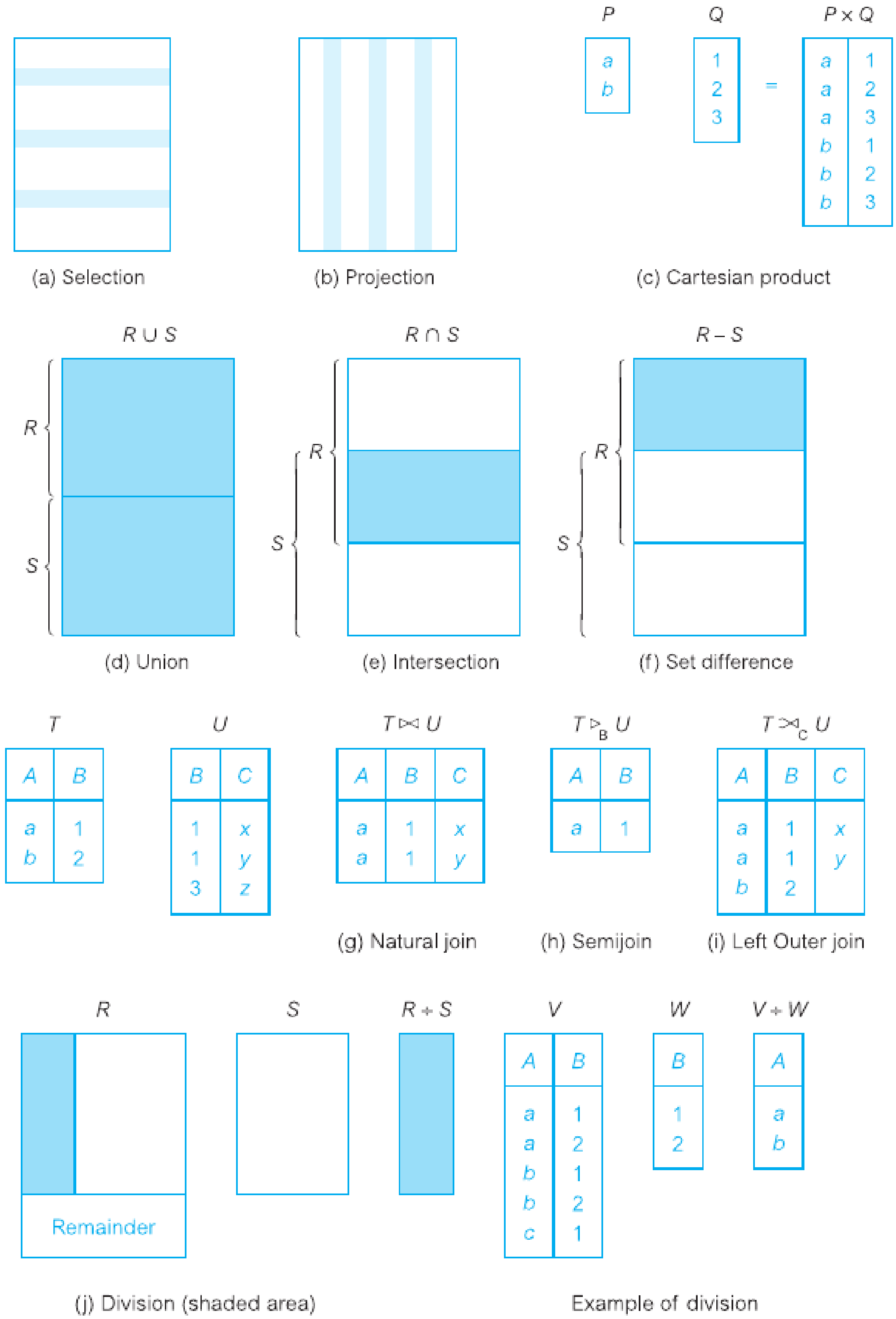
Aggregate operations

$\mathbf{\Im_{AL}(R)}$	Applies the aggregate function list, AL, to the relation R to define a relation over the aggregate list. AL contains one or more (<aggregate_function>, <attribute>) pairs.

The main aggregate functions are:

- COUNT – returns the number of values in the associated attribute.
- SUM – returns the sum of the values in the associated attribute.
- AVG – returns the average of the values in the associated attribute.
- MIN – returns the smallest value in the associated attribute.
- MAX – returns the largest value in the associated attribute.

OVERVIEW



P
Q
P × Q
a
b
1
2
3
=
a 1
a 2
a 3
b 1
b 2
b 3
(a) Selection
(b) Projection
(c) Cartesian product
R ∪ S
R ∩ S
R – S
R
S
(d) Union
(e) Intersection
(f) Set difference
T
U
T ⋈ U
T ⊳B U
T ⟕C U
A B
a 1
b 2
B C
1 x
1 y
3 z
A B C
a 1 x
a 1 y
A B
a 1
A B C
a 1 x
a 1 y
b 2
(g) Natural join
(h) Semijoin
(i) Left Outer join
R
S
R ÷ S
V
W
V ÷ W
Remainder
A B
a 1
a 2
b 1
b 2
c 1
B
1
2
A
a
b
(j) Division (shaded area)
Example of division