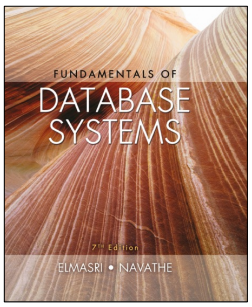
Fundamentals of Database Systems

Seventh Edition



Chapter 8

The Relational Algebra and
The Relational Calculus

2

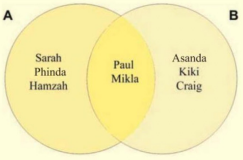
Relational Algebra

- Relational algebra and relational calculus are the mathematical basis for 'relational databases'.
- **Relational algebra…….**
  - A collection of formal operations acting on existing relations (tables) which produce new relations (tables) as a result
  - Defines theoretical way of manipulating table contents using relational operators

3

Re-visiting Set Theory

FIGURE 4.1 A simple Venn diagram



In Figure 4.1, the two circles represent the two sets A and B.The students who take both the Database and Programming units and who appear in both sets are Paul and Mikla. These will go in the overlapping sections of the two circles. Sarah, Phinda and Hamzah only take the Database unit, so these go only in the left-hand circle, whilst Asanda, Kiki and Craig only take the Programming unit and only appear in the right-hand circle.

4

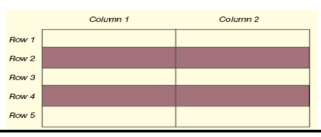
Relational Algebra Operators

- **SELECT**
- **PROJECT**
- **JOIN**
- UNION
- INTERSECT
- DIFFERENCE
- PRODUCT

5

SELECT

- Yields values for all rows found in a table
- Can be used to list either all row values or it can yield only those row values that match a specified criterion
- Yields a horizontal subset of a table



6

SELECT (continued)

- The SELECT operator denoted by σ_θ , is formally defined as:

  $\sigma_\theta (R)$

  or

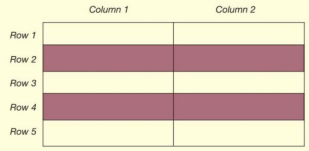
  $\sigma_{<criterion>} (RELATION)$

- where $\sigma_\theta (R)$ is the set of specified tuples of the relation *R* and θ is the predicate (or criterion) to extract the required tuples.

7

FIGURE 4.2 The SELECT operator

Figure 4.2 (a) SELECTION



Database name: Ch04_Relational_DB_Operators

Figure 4.2 (b) The PRODUCT Relation

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	€4.16
123457	Lamp	€19.87
123458	Box Fan	€8.68
213345	9 v battery	€1.52
254467	100 W bulb	€1.16
311452	Powerdrill	€27.64

Figure 4.2 (c) σ *(PRODUCT)*

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	€4.16
123457	Lamp	€19.87
123458	Box Fan	€8.68
213345	9 v battery	€1.52
254467	100 W bulb	€1.16
311452	Powerdrill	€27.64

Figure 4.2 (d) $\sigma_{price < 2.00}$ *(PRODUCT)*

P_CODE	P_DESCRIPT	PRICE
213345	9 v battery	€1.52
254467	100 W bulb	€1.16

Figure 4.2 (e) $\sigma_{p\_code = 123456}$ *(PRODUCT)*

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	€4.16

●8

SELECT (continued)

FIGURE 4.3 Selecting from the COURSE relation

Database name: Ch04_TinyUniversity

Figure 4.3 (a) the COURSE Relation

CRS_CODE	DEPT_CODE	CRS_DESCRIPTION	CRS_CREDIT
ACCT-211	ACCT	Accounting I	3
ACCT-212	ACCT	Accounting II	3
CIS-220	CIS	Introduction to Computer Science	3
CIS-420	CIS	Database Design and Implementation	4
QM-261	CIS	Intro. to Statistics	3
QM-362	CIS	Statistical Applications	4

Figure 4.3 (b) $\sigma_{dept\_code = 'CIS' AND\ crs\_credit = 4}$ *(COURSE)*

CRS_CODE	DEPT_CODE	CRS_DESCRIPTION	CRS_CREDIT
CIS-420	CIS	Database Design and Implementation	4
QM-362	CIS	Statistical Applications	4

●9

SELECT (continued)

- In SQL, the SELECT condition is typically specified in the WHERE clause of a query. The following operation:

  $\sigma_{dept\_code='CIS'\ \textbf{AND}\ course\_credit=4}$ *(COURSE)*

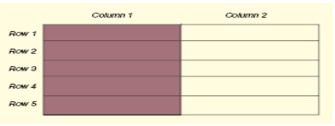
  would correspond to the following SQL query:

```
SELECT    *
FROM      COURSE
WHERE     dept_code='CIS' AND
          course_credit=4;
```

●10

PROJECT

- Yields all values for selected attributes
- Yields a vertical subset of a table



●11

PROJECT (continued)

- The PROJECT operator, denoted by ∏, is formally defined as:

  ∏ $_{a1 \ldots an}$ *(R)*

  or

  ∏ <List of attributes> (Relation)

- where the projection of the relation *R*, denoted by ∏ $_{a1 \ldots an}$ *(R)* is the set of specified attributes $_{a1 \ldots an}$ of the relation *R*.

●12

PROJECT (continued)

FIGURE 4.4 The PROJECT operator

Database name: Ch04_Relational_DB_Operators

Figure 4.4 (a) PROJECTION

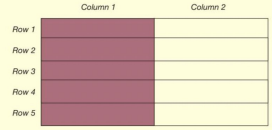


Figure 4.4 (b) The PRODUCT relation

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	€4.16
123457	Lamp	€19.87
123458	Box Fan	€8.68
213345	9 v battery	€1.52
254467	100 W bulb	€1.16
311452	Powerdrill	€27.64

Figure 4.4 (d) $\Pi_{p\_descript,price}$ (PRODUCT)

P_DESCRIPT	PRICE
Flashlight	€4.16
Lamp	€19.87
Box Fan	€8.68
9 v battery	€1.52
100 W bulb	€1.16
Powerdrill	€27.64

Figure 4.4 (c) Π_{price} (PRODUCT)

PRICE
€4.16
€19.87
€8.68
€1.52
€1.16
€27.64

Figure 4.4 (e) $\Pi_{p\_code,price}$ (PRODUCT)

P_CODE	PRICE
123456	€4.16
123457	€19.87
123458	€8.68
213345	€1.52
254467	€1.16
311452	€27.64

●13

PROJECT (continued)

- In SQL, the PROJECT attribute list is specified in the SELECT clause of a query

  $\prod_{p\_code,\ price}$ (PRODUCT)

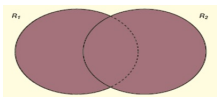
  would correspond to the following SQL query:

```
SELECT    DISTINCT p_code, price
FROM      PRODUCT
```

  (DISTINCT eliminates the dublicates. This option is not available in the formal relational algebra)

●14

Union



- Combines all rows from two tables, excluding duplicate rows
- Tables must have the same attribute characteristics
- When two or more tables share the same number of columns, i.e. have the same degree, and when they share the same (or compatible) domains, they are said to be **union-compatible**.

●15

Union (continued)

- The UNION operator, denoted by ∪, is formally defined as:

  The union of relations $R_1(a_1, a_2, \ldots, a_n)$ and $R_2(b_1, b_2, \ldots, b_n)$ denoted $R_1 \cup R_2$ with degree n, is the relation $R_3(c_1, c_2, \ldots, c_n)$ where for each i ($i = 1,2..n$), a_i and b_i must have compatible domains.

●16

Union (continued)

FIGURE 4.5 The UNION operator

Database name: Ch04_Relational_DB_Operators

Figure 4.5 (a) R_1 Union R_2

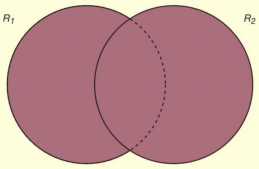


Figure 4.5 (b) The UNION_PRODUCT1 relation

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	€4.16
123457	Lamp	€19.87
123458	Box Fan	€8.68
213345	9 v battery	€1.52
254467	100 W bulb	€1.16
311452	Powerdrill	€27.64

Figure 4.5 (c) The UNION_PRODUCT2 relation

P_CODE	P_DESCRIPT	PRICE
345678	Microwave	€126.40
345679	Dishwasher	€395.00

Figure 4.5 (d) Result of UNION_PRODUCT1 ∪ UNION_PRODUCT2

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	€4.16
123457	Lamp	€19.87
123458	Box Fan	€8.68
213345	9 v battery	€1.52
254467	100 W bulb	€1.16
311452	Powerdrill	€27.64
345678	Microwave	€126.40
345679	Dishwasher	€395.00

●17

Union (continued)

FIGURE 4.6 The Union operator – COURSE ∪ COURSE2

Database name: Ch04_TinyUniversity

Figure 4.6 (a) The COURSE_RELATION

CRS_CODE	DEPT_CODE	CRS_DESCRIPTION	CRS_CREDIT
ACCT-211	ACCT	Accounting I	3
ACCT-212	ACCT	Accounting II	3
CIS-220	CIS	Introduction to Computer Science	3
CIS-420	CIS	Database Design and Implementation	4
QM-261	CIS	Intro. to Statistics	3
QM-362	CIS	Statistical Applications	4

Figure 4.6 (b) The COURSE2_RELATION

CRS_CODE	DEPT_CODE	CRS_DESCRIPTION	CRS_CREDIT
ACCT-211	ACCT	Accounting I	3
CIS-430	CIS	Advanced Databases	6

Figure 4.6 (c) Result of COURSE ∪ COURSE2

CRS_CODE	DEPT_CODE	CRS_DESCRIPTION	CRS_CREDIT
ACCT-211	ACCT	Accounting I	3
ACCT-212	ACCT	Accounting II	3
CIS-220	CIS	Introduction to Computer Science	3
CIS-420	CIS	Database Design and Implementation	4
QM-261	CIS	Intro. to Statistics	3
QM-362	CIS	Statistical Applications	4
CIS-430	CIS	Advanced Databases	6

●18

Union (continued)

FIGURE 4.7 The Union operator — not union-compatible example

Database name: Ch04_TinyUniversity

Figure 4.7 (a) the CLASS_RELATION

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	LECTURER_NUM
10012	ACCT-211	1	MWF 8:00-8:50 a.m.	BUS311	105
10013	ACCT-211	2	MWF 9:00-9:50 a.m.	BUS200	105
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10015	ACCT-212	1	MWF 10:00-10:50 a.m.	BUS311	301
10016	ACCT-212	2	Th 6:00-8:40 p.m.	BUS252	301
10017	CIS-220	1	MWF 9:00-9:50 a.m.	KLR209	228
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10019	CIS-220	3	MWF 10:00-10:50 a.m.	KLR209	228
10020	CIS-420	1	W 6:00-8:40 p.m.	KLR209	162
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
10022	QM-261	2	TTh 1:00-2:15 p.m.	KLR200	114
10023	QM-362	1	MWF 11:00-11:50 a.m.	KLR200	162
10024	QM-362	2	TTh 2:30-3:45 p.m.	KLR200	162

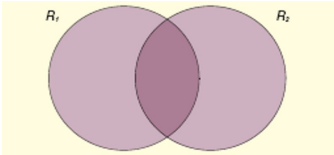
Figure 4.7 (b) Result of Π_{CRS_CODE} (COURSE) ∪ Π_{CRS_CODE} (CLASS)

CRS_CODE
ACCT-211
ACCT-212
CIS-220
CIS-420
QM-261
QM-362

●19

Intersect

- Yields only the rows that appear in both tables
- the tables must be union-compatible to give valid results.



20

Intersect (continued)

- The **INTERSECT** operator is formally defined as:

  The intersect of relations $R_1(a_1, a_2, \ldots, a_n)$ and $R_2(b_1, b_2, \ldots, b_n)$ denoted $R_1 \cap R_2$ with degree $_n$, is the relation $R_3(c_1, c_2, \ldots, c_n)$ that includes only those tuples of R_1 that also appear in R_2 where for each i ($i = 1,2..n$), a_i and b_i must have compatible domains.

21

Intersect (continued)

FIGURE 4.8 The INTERSECT operator

Database name: Ch04_Relational_DB_Operators
Figure 4.8 (a) R_1 INTERSECT R_2

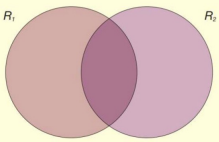


Figure 4.8 (b) The INTERSECT_RELATION_1 relation

F_NAME
George
Kuhle
Elaine
Piet
Jorge

Figure 4.8 (c) The INTERSECT_RELATION_2 relation

F_NAME
Kuhle
William
Jorge
Dennis

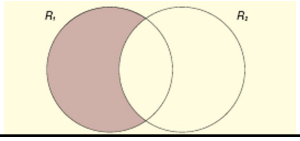
Figure 4.8 (d) Result of INTERSECT_RELATION_1 ∩ INTERSECT_RELATION_2

F_NAME
Kuhle
Jorge

22

Difference

- Yields all rows in one table not found in the other table — that is, it subtracts one table from the other
- The DIFFERENCE operator also requires that the two relations must be union-compatible.



23

Difference (continued)

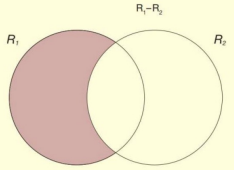
- The DIFFERENCE operator is formally defined as:

  The difference of relations $R_1(a_1, a_2, \ldots, a_m)$ and $R_2(b_1, b_2, \ldots, b_m)$ denoted $R_1 - R_2$ with degree m, is the relation $R_3(c_1, c_2, \ldots, c_m)$ that includes all tuples that are in R_1 but not in R_2 where for each i ($i = 1,2..m$), a_i and b_i must have compatible domains.

24

Difference (continued)

FIGURE 4.9 The DIFFERENCE operator



Database name: Ch04_Relational_DB_Operators

Figure 4.9 (b) The DIFF_RELATION_1 relation

F_NAME
George
Kuhle
Elaine
Piet
Jorge

Figure 4.9 (c) The DIFF_RELATION_2 relation

F_NAME
Kuhle
William
Jorge
Dennis

Figure 4.9 (d) Result of DIFF_RELATION_1 - DIFF_RELATION_2

F_NAME
George
Elaine
Piet

25

Cartesian product

- Yields all possible pairs of rows from two tables
- usually written as R_1 X R_2
- also known as the product.

26

Cartesian product (continued)

- It can be formally defined as:

  The CARTESIAN PRODUCT of two relations $R_1(a_1, a_2, \ldots, a_n)$ with cardinality i and $R_2(b_1, b_2, \ldots, b_m)$ with cardinality j is a relation R_3 with degree $k = n + m$, cardinality $i*j$ and attributes $(a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m)$.

  This can be denoted as $R_3 = R_1 \; X \; R_2$.

27

Cartesian product (continued)

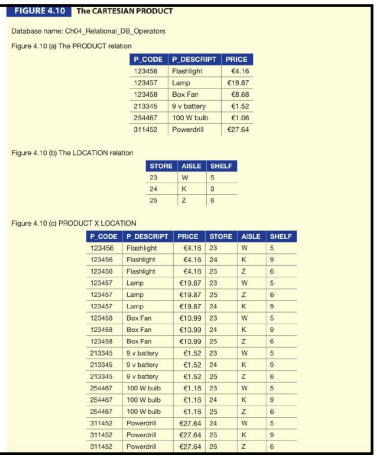


FIGURE 4.10 The CARTESIAN PRODUCT

Database name: Ch04_Relational_DB_Operators

Figure 4.10 (a) The PRODUCT relation

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	€4.16
123457	Lamp	€19.87
123458	Box Fan	€8.68
213345	9 v battery	€1.52
254467	100 W bulb	€1.06
311452	Powerdrill	€27.64

Figure 4.10 (b) The LOCATION relation

STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

Figure 4.10 (c) PRODUCT X LOCATION

P_CODE	P_DESCRIPT	PRICE	STORE	AISLE	SHELF
123456	Flashlight	€4.16	23	W	5
123456	Flashlight	€4.16	24	K	9
123456	Flashlight	€4.16	25	Z	6
123457	Lamp	€19.87	23	W	5
123457	Lamp	€19.87	25	Z	6
123457	Lamp	€19.87	24	K	9
123458	Box Fan	€10.99	23	W	5
123458	Box Fan	€10.99	24	K	9
123458	Box Fan	€10.99	25	Z	6
213345	9 v battery	€1.52	23	W	5
213345	9 v battery	€1.52	24	K	9
213345	9 v battery	€1.52	25	Z	6
254467	100 W bulb	€1.16	23	W	5
254467	100 W bulb	€1.16	24	K	9
254467	100 W bulb	€1.16	25	Z	6
311452	Powerdrill	€27.64	24	W	5
311452	Powerdrill	€27.64	25	K	9
311452	Powerdrill	€27.64	25	Z	6

28

JOIN operators

- Allows information to be combined from two or more tables
- Real power behind the relational database, allowing the use of independent tables linked by common attributes

32

Join operators defined

- A JOIN Operator may be formally defined as:

  The join of two relations $R_1(a_1, a_2, \ldots, a_n)$ and $R_2(b_1, b_2, \ldots, b_m)$ is a relation R_3 with degree $k = n + m$ and attributes $(a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m)$ that satisfy a specific join condition.

  R_1 |XI $_{\text{<join condition>}}$ R_2

33

Types of Join

- NATURAL JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN

40

Natural Join

- Links tables by selecting only rows with common values in their common attribute(s)
- The common column(s) is (are) referred to as the **join column(s)**.
- The natural join is in fact an equijoin, however, in addition, we drop the duplicate attributes, so the resulting relation contains one less column than that of the equijoin.

41

Computing the Natural Join

1. Compute R_1 X R_2. This first performs a Cartesian product to form all possible combinations of the rows of R_1 and R_2.
2. Select those tuples where $R_1 Tuple.y = R_2 Tuple.y$. Only the rows are selected where the attribute values in the join column(s) are equal.
3. Perform a PROJECT operation on either $R_1.y$ or $R_2.y$ to the result of step (2), and call it y in the final relation.

42

Natural Join Example

FIGURE 4.13 The CUSTOMER and AGENT relations

Database name: Ch04_Relational_DB_Operators

Relation: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_POSTCODE	AGENT_CODE
1132445	Strydom	4001	231
1217782	Adares	7550	125
1312243	Nokwe	678954	167
1321242	Reddy	2094	125
1542311	Smithson	1401	421
1657399	Vanloo	67543W	231

Relation: AGENT

AGENT_CODE	AGENT_PHONE
125	01812439887
167	01813426778
231	01812431124
333	01131234445

43

Natural Join Example (continued)

FIGURE 4.14 Step 1: CUSTOMER X AGENT

Database name: Ch04_Relational_DB_Operators

C.CUS_CODE	C.CUS_LNAME	C.CUS_POSTCODE	C.AGENT_CODE	A.AGENT_CODE	A.AGENT_PHONE
1132445	Strydom	4001	231	125	01812439887
1132445	Strydom	4001	231	167	01813426778
1132445	Strydom	4001	231	231	01812431124
1132445	Strydom	4001	231	333	01131234445
1217782	Adares	7550	125	125	01812439887
1217782	Adares	7550	125	167	01813426778
1217782	Adares	7550	125	231	01812431124
1217782	Adares	7550	125	333	01131234445
1312243	Nokwe	678954	167	125	01812439887
1312243	Nokwe	678954	167	167	01813426778
1312243	Nokwe	678954	167	231	01812431124
1312243	Nokwe	678954	167	333	01131234445
1321242	Reddy	2094	125	125	01812439887
1321242	Reddy	2094	125	167	01813426778
1321242	Reddy	2094	125	231	01812431124
1321242	Reddy	2094	125	333	01131234445
1542311	Smithson	1401	421	125	01812439887
1542311	Smithson	1401	421	167	01813426778
1542311	Smithson	1401	421	231	01812431124
1542311	Smithson	1401	421	333	01131234445
1657399	Vanloo	67543W	231	125	01812439887
1657399	Vanloo	67543W	231	167	01813426778
1657399	Vanloo	67543W	231	231	01812431124
1657399	Vanloo	67543W	231	333	01131234445

44

Natural Join Example (cont)

FIGURE 4.15 Step 2: Selecting rows where values in the join column match

Database name: Ch04_Relational_DB_Operators

Relation CUSTOMER X AGENT

Joining columns

C.CUS_CODE	C.CUS_LNAME	C.CUS_POSTCODE	C.AGENT_CODE	A.AGENT_CODE	A.AGENT_PHONE
1132445	Strydom	4001	231	125	01812439887
1132445	Strydom	4001	231	167	01813426778
1132445	Strydom	4001	231	231	01812431124
1132445	Strydom	4001	231	333	01131234445
1217782	Adares	7550	125	125	01812439887
1217782	Adares	7550	125	167	01813426778
1217782	Adares	7550	125	231	01812431124
1217782	Adares	7550	125	333	01131234445
1312243	Nokwe	678954	167	125	01812439887
1312243	Nokwe	678954	167	167	01813426778
1312243	Nokwe	678954	167	231	01812431124
1312243	Nokwe	678954	167	333	01131234445
1321242	Reddy	2094	125	125	01812439887
1321242	Reddy	2094	125	167	01813426778
1321242	Reddy	2094	125	231	01812431124
1321242	Reddy	2094	125	333	01131234445
1542311	Smithson	1401	421	125	01812439887
1542311	Smithson	1401	421	167	01813426778
1542311	Smithson	1401	421	231	01812431124
1542311	Smithson	1401	421	333	01131234445
1657399	Vanloo	67543W	231	125	01812439887
1657399	Vanloo	67543W	231	167	01813426778
1657399	Vanloo	67543W	231	231	01812431124
1657399	Vanloo	67543W	231	333	01131234445

The tuples shaded in blue are those where C.AGENT_CODE = A.AGENT.CODE. These are then selected to produce the results of Step 2.

45

Natural Join Example (continued)

C.CUS_CODE	C.CUS_LNAME	C.CUS_POSTCODE	C.AGENT_CODE	A.AGENT_CODE	A.AGENT_PHONE
1132445	Strydom	4001	231	231	01812431124
1217782	Adares	7550	125	125	01812439887
1312243	Nokwe	678954	167	167	01813426778
1321242	Reddy	2094	125	125	01812439887
1657399	Vanloo	67543W	231	231	01812431124

46

Natural Join Example (continued)

FIGURE 4.16 Step 3: Final relation CUSTOMER |X| AGENT

Database name: Ch04_Relational_DB_Operators

CUS_CODE	CUS_LNAME	CUS_POSTCODE	AGENT_CODE	AGENT_PHONE
1132445	Strydom	4001	231	01812431124
1217782	Adares	7550	125	01812439887
1312243	Nokwe	678954	167	01813426778
1321242	Reddy	2094	125	01812439887
1657399	Vanloo	67543W	231	01812431124

47

Relational Algebra Operators (continued)

- Natural Join:
  - Final outcome yields table that
    - Does not include unmatched pairs
    - Provides only copies of matches
  - If no match is made between the table rows
    - the new table does not include the unmatched row

48

Outer Join

- Matched pairs are retained and any unmatched values in other table are left null
- In outer join for tables CUSTOMER and AGENT, two scenarios are possible:
  - Left outer join

    Yields all rows in CUSTOMER table, including those that do not have a matching value in the AGENT table
  - Right outer join

    Yields all rows in AGENT table, including those that do not have matching values in the CUSTOMER table

49

Types of Outer Join

There are two common types of the outer join:

⟕ Left outer join – keeps data from the left-hand relation

⟖ Right outer join – keeps data from the right-hand relation

50

Computing a Left Outer Join

1. Compute R_1 X R_2. This first performs a Cartesian product to form all possible combinations of the rows of R_1 and R_2.
2. Select those tuples where $R_1Tuple.y = R_2Tuple.y$.
3. Select those tuples in R_1 that do not have matching values in R_2, so $R_1Tuple.y <> R_2Tuple.y$.
4. Perform a PROJECT operation on either $R_1.y$ or $R_2.y$ to the result of step (2), and call it simply y in the final relation. Finally, project the rest of the attributes in R_1 and R_2, except y, and drop the prefix R_1 and R_2 in the final relation.

51

FIGURE 4.13 The CUSTOMER and AGENT relations

Database name: Ch04_Relational_DB_Operators

Relation: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_POSTCODE	AGENT_CODE
1132445	Walker	M1 5RT	231
1217782	Adares	NW6 4RT	125
1312243	Rakowski	678954	167
1321242	Rodriguez	NW6 2WS	125
1542311	Smithson	N4 3YP	421
1657399	Vanloo	67543W	231

Relation: AGENT

AGENT_CODE	AGENT_PHONE
125	01812439887
167	01813426778
231	01812431124
333	01131234445

52

Left Outer Join Example

FIGURE 4.17 Left outer join : CUSTOMER ⟕ AGENT

Database name: Ch04_Relational_DB_Operators

CUS_CODE	CUS_LNAME	CUS_POSTCODE	AGENT_CODE	AGENT_PHONE
1132445	Strydom	4001	231	01812431124
1217782	Adares	7550	125	01812439887
1312243	Nokwe	678954	167	01813426778
1321242	Reddy	2094	125	01812439887
1657399	Vanloo	67543W	231	01812431124
1542311	Smithson	1401	421	NULL

53

Right Outer Join Example

FIGURE 4.18 Right outer join : CUSTOMER ⟖ AGENT

Database name: Ch04_Relational_DB_Operators

CUS_CODE	CUS_LNAME	CUS_POSTCODE	AGENT_CODE	AGENT_PHONE
1132445	Strydom	4001	231	01812431124
1217782	Adares	7550	125	01812439887
1312243	Nokwe	678954	167	01813426778
1321242	Reddy	2094	125	01812439887
1657399	Vanloo	67543W	231	01812431124
NULL	NULL	NULL	333	01131234445

54

Building Queries

In order to build a query using a **relational algebraic expression** you should follow the following steps:

1. List all the attributes we need to give the answer.
2. Select all the relations we need, based on the list of attributes.
3. Specify the relational operators and the intermediate results that are needed.

55

Case: CAR INSPECTION DATABASE

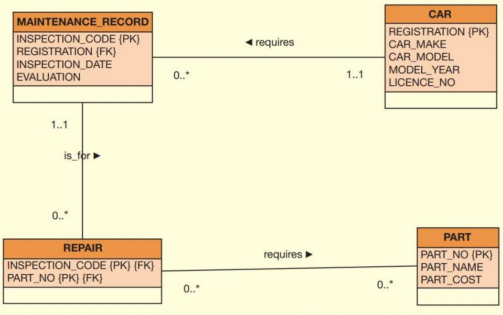
Draw an ERD for a small database that stores data about the maintenance of cars.

- Each car is required to undergo an inspection each year to test whether it is roadworthy. After each inspection, a maintenance record is created and any repairs that are needed are recorded. A repair can require new parts to be purchased and fitted. If a car needs a repair, then the evaluation is set to 'fail' until all the repairs are completed and then it is set to 'pass'.

56

Example: Car Inspection ERD

FIGURE 4.19 The car inspection ERD



57

Example: Car Inspection ERD Database

FIGURE 4.20 The car inspection database

Database name: Ch04_Car_Inspection

Table name: CAR

REGISTRATION	CAR_MAKE	CAR_MODEL	CAR_COLOUR	MODEL_YEAR	LICENCE_NO
3679MR82	Toyota	Corolla	Blue	2016	1967fr89768
E-TS865	Nissan	Micra	Red	2004	1973Smith121
PE57UVP	Peugeot	508	Blue	2017	1990bty3212
PISE567	Volkswagen	Eos	Lime	2016	DF-678-WV
ROMA482	Volkswagen	Golf GT	Black	2017	AQ-123-AV
Z-BA975	Peugeot	208	Black	2017	1980vrt7312

58

Table name: PART

PART_NO	PART_NAME	PART_COST
12390	Paint sealants	€14.95
12391	Wiper	€19.95
12392	Brake pads	€24.99
12393	Brake discs	€49.54
12395	Spark plugs	€0.99
12396	Airbag	€24.95
12397	Tyres	€25.00

Table name: MAINTENANCE_RECORD

INSPECTION_CODE	REGISTRATION	INSPECTION_DATE	EVALUATION
100036	PE57UVP	10/05/2018	FAIL
100390	ROMA482	01/09/2018	
106750	E-TS865	01/03/2016	PASS
122456	Z-BA975	03/10/2018	FAIL
145678	PISE567	30/09/2017	PASS
200450	E-TS865	21/02/2015	PASS
200456	E-TS865	01/04/2017	FAIL

Table name: REPAIR

INSPECTION_CODE	PART_NO
106750	12396
106750	12397
100036	12393
200450	12391
100036	12397
200450	12392
200456	12397

FIGURE 4.20 The car inspection database

Database name: Ch04_Car_Inspection

Table name: CAR

REGISTRATION	CAR_MAKE	CAR_MODEL	CAR_COLOUR	MODEL_YEAR	LICENCE_NO
3679MR82	Toyota	Corolla	Blue	2016	1967fr89768
E-TS865	Nissan	Micra	Red	2004	1973Smith121
PE57UVP	Peugeot	508	Blue	2017	1990bty3212
PISE567	Volkswagen	Eos	Lime	2016	DF-678-WV
ROMA482	Volkswagen	Golf GT	Black	2017	AQ-123-AV
Z-BA975	Peugeot	208	Black	2017	1980vrt7312

Example: Car Inspection Database

●59

Example 1

'List all information about cars where the model year is after 2016.'

→ $\sigma_{model\_year > 2016}(CAR)$

FIGURE 4.21 Result of $\sigma_{model\_year > 2016}(CAR)$

REGISTRATION	CAR_MAKE	CAR_MODEL	CAR_COLOUR	MODEL_YEAR	LICENSE_NO
PE57UVP	Peugeot	508	Blue	2017	1990bty3212
ROMA482	Volkswagen	Golf GT	Black	2017	AQ-123-AV
Z-BA975	Peugeot	208	Black	2017	1980vrt7312

●60

Example 1

'List all information about cars where the model year is after 2016.'

SELECT *

FROM CAR

WHERE MODEL_YEAR > 2016

FIGURE 4.21 Result of $\sigma_{model\_year > 2016}(CAR)$

●61

Example 2

'Display all the part names and their costs where the cost of the part is greater than 20.00 €.'

→ $\prod_{part\_name}(\sigma_{part\_cost > 20.00}(PART))$

FIGURE 4.22 Result of $\Pi_{part\_name}(\sigma_{part\_cost > 20.00}(PART))$

PART_NAME	PART_COST
Brake Pads	€24.99
Brake Discs	€49.54
Airbag	€24.95

●62

Example 2

'Display all the part names and their costs where the cost of the part is greater than 20.00 €.'

SELECT PART_NAME, PART_COST

FROM PART

WHERE PART_COST > 20

FIGURE 4.22 Result of $\Pi_{part\_name}(\sigma_{part\_cost > 20.00}(PART))$

●63

Example 3

- 'List the car registration and model details for all cars where the model year is 2017, where an inspection was carried out after 01/03/2018.'
  - This is a more complex query. It has to be broken down into a number of different stages, each one having a set of intermediate results.

INSPECTION_CODE	REGISTRATION	INSPECTION_DATE	EVALUATION	CAR_MODEL
100036	PE57UVP	10/05/2018	FAIL	508
100390	ROMA482	01/09/2018		Golf GT
122456	Z-BA975	03/10/2018	FAIL	208

●65

Example: Car Inspection Database

Table name: PART

PART_NO	PART_NAME	PART_COST
12390	Paint sealants	€14.95
12391	Wiper	€19.95
12392	Brake pads	€24.99
12393	Brake discs	€49.54
12395	Spark plugs	€0.99
12396	Airbag	€24.95
12397	Tyres	€25.00

Table name: MAINTENANCE_RECORD

INSPECTION_CODE	REGISTRATION	INSPECTION_DATE	EVALUATION
100036	PE57UVP	10/05/2018	FAIL
100390	ROMA482	01/09/2018	
106750	E-TS865	01/03/2016	PASS
122456	Z-BA975	03/10/2018	FAIL
145678	PISE567	30/09/2017	PASS
200450	E-TS865	21/02/2015	PASS
200456	E-TS865	01/04/2017	FAIL

Table name: REPAIR

INSPECTION_CODE	PART_NO
106750	12396
106750	12397
100036	12393
200450	12391
100036	12397
200450	12392
200456	12397

FIGURE 4.20 The car inspection database

Database name: Ch04_Car_Inspection

Table name: CAR

REGISTRATION	CAR_MAKE	CAR_MODEL	CAR_COLOUR	MODEL_YEAR	LICENCE_NO
3679MR82	Toyota	Corolla	Blue	2016	19679893768
E-TS865	Nissan	Micra	Red	2004	1973Smith12
PE57UVP	Peugeot	508	Blue	2017	1968Bo292121
PISE567	Volkswagen	Eos	Lime	2016	DF-678-WV
ROMA482	Volkswagen	Golf GT	Black	2017	AQ-123-AV
Z-BA975	Peugeot	208	Black	2017	1980vrt7312

66

Example 3 (continued)

FIGURE 4.23 Result of $\prod_{registration,\ car\_model}(\sigma_{model\_year=2017}(CAR))$

REGISTRATION	CAR_MODEL
PE57UVP	508
ROMA482	Golf GT
Z-BA975	208

Information about the CAR

FIGURE 4.24 Result of $\sigma_{inspection\_date>'01/03/2018'}(MAINTENANCE)$

INSPECTION_CODE	REGISTRATION	INSPECTION_DATE	EVALUATION
100036	PE57UVP	10/05/2018	FAIL
100390	ROMA482	01/09/2018	
122456	Z-BA975	03/10/2018	FAIL

Information about the INSPECTION

67

Example 3 (continued)

FIGURE 4.23

```
SELECT REGISTRATION, CAR_MODEL
FROM CAR
WHERE MODEL_YEAR = 2017
```

REGISTRATION	CAR_MODEL
PE57UVP	508
ROMA482	Golf GT
Z-BA975	208

Information about the CAR

FIGURE 4.24

```
SELECT *
FROM MAINTENANCE
WHERE INSPECTION_DATE > #01/03/2018#
```

INSPECTION_CODE	REGISTRATION	INSPECTION_DATE	EVALUATION
100036	PE57UVP	10/05/2018	FAIL
100390	ROMA482	01/09/2018	
122456	Z-BA975	03/10/2018	FAIL

Information about the INSPECTION

68

Example 3 (continued)

Join the rows from the first two parts of the query

(Use natural join with a common column Registration);

$TempR = \prod_{registration,\ car\_model}(\sigma_{model\_year=2017}(CAR))$

IXI

$\sigma_{inspection\_date>'01/03/2018'}(MAINTENANCE)$

69

Example 3 (continued)

Join the rows from the first two parts of the query

(Use natural join with a common column Registration);

SELECT INSPECTION, CAR.REGISTRATION, INSPECTION_DATE, EVALUATION, CAR_MODEL,

FROM CAR, MAINTENANCE

WHERE CAR.REGISRATION = MAINTENANCE.REGISTRATION

AND MODEL_YEAR = 2017

AND INSPECTION_DATE > #01/03/2018#

70

Example 3 (continued)

FIGURE 4.25 The TempR relation

Step 1: Compute the Cartesian product: MAINTENANCE_RECORD X CAR.

M.INSPECTION_CODE	M.REGISTRATION	M.INSPECTION_DATE	M.EVALUATION	C.REGISTRATION	C.CAR_MODEL
100036	PE57UVP	10/05/2018	FAIL	PE57UVP	508
100036	PE57UVP	10/05/2018	FAIL	ROMA482	Golf GT
100036	PE57UVP	10/05/2018	FAIL	Z-BA975	208
100390	ROMA482	01/09/2018		PE57UVP	508
100390	ROMA482	01/09/2018		ROMA482	Golf GT
100390	ROMA482	01/09/2018		Z-BA975	208
122456	Z-BA975	03/10/2018	FAIL	PE57UVP	508
122456	Z-BA975	03/10/2018	FAIL	ROMA482	Golf GT
122456	Z-BA975	03/10/2018	FAIL	Z-BA975	208

Step 2: SELECT only the rows for which the REGISTRATION values are equal, i.e. M. REGISTRATION = C. REGISTRATION.

Joining Columns

M.INSPECTION_CODE	M.REGISTRATION	M.INSPECTION_DATE	M.EVALUATION	C.REGISTRATION	C.CAR_MODEL
100036	PE57UVP	10/05/2018	FAIL	PE57UVP	508
100390	ROMA482	01/09/2018		ROMA482	Golf GT
122456	Z-BA975	03/10/2018	FAIL	Z-BA975	208

Step 3: Perform a PROJECT on either C.REGISTRATION or M.REGISTRATION to the result of Step 2 and drop the prefixes C and M in the final relation. The table below shows the relation TempR, which has been created as a result of Step 3.

INSPECTION_CODE	REGISTRATION	INSPECTION_DATE	EVALUATION	CAR_MODEL
100036	PE57UVP	10/05/2018	FAIL	508
100390	ROMA482	01/09/2018		Golf GT
122456	Z-BA975	03/10/2018	FAIL	208

71

Summary

- One of the key components of the relational model is the relation which allows data to be stored within the database in a structured manner.
- Relational algebra and relational calculus are the mathematical basis for 'relational data bases'.
- Relational algebra is a collection of formal operations which act on relations to produce new relations as a result.

74

Summary (continued)

- The relational model supports the relational algebra operators originally defined by Codd.
- These are known as SELECT (or RESTRICT), PROJECT, JOIN, PRODUCT, INTERSECT, UNION, and DIFFERENCE.

75

Summary (continued)

User queries can be written as relational algebraic expressions. In order to write such as an expression, the following steps should be followed:

1. List all the attributes we need to give the answer.
2. Select all the relations which we need, based on the list of attributes.
3. Specify the relational operators and the intermediate results that are needed.

76