Chapter 1

RELATIONAL DATA MODEL

The relational model of data presents a logical view of a database in which the user perceives the data to be organized in tabular form. This is a very simple and intuitive view of data which hides all the complex details of how the data is actually stored and accessed within a computer. Moreover, over the years, very sophisticated and efficient structures and algorithms have been developed to implement database systems based on the relational model. As a result, relational databases are being used overwhelmingly in the industry and anywhere else there is a need to manage large amounts of data.

In this chapter, the basic concepts and foundations of the relational data model are presented. In addition, two sample relational databases are described which will be used throughout this book.

1.1 Relational Database

The relational data model has a strong mathematical foundation based on set theory. A very brief mathematical definition of a relational database follows.

A relation scheme is a finite sequence of unique attribute names. For example,

\[ \text{EMPLOYEES} = (\text{EMPID}, \text{ENAME}, \text{ADDRESS}, \text{SALARY}) \]
is a relation scheme with four attribute names.

A domain is a set of values. With each attribute name, A, a domain, \( \text{dom}(A) \), is associated. This domain includes a special value called null. For example, \( \text{dom}(\text{EMPID}) \) could be the set of all possible integers between 1000 and 9999 and the special null value.

Given a relation scheme \( R = A_1, \ldots, A_n \), a relation \( r \) on the scheme \( R \) is defined as any finite subset of the Cartesian product

\[
\text{dom}(A_1) \times \ldots \times \text{dom}(A_n).
\]

Assuming appropriate domains for the EMPLOYEES relation scheme, a sample relation under this scheme could be

\[
\{(1111, 'Jones', '111 Ash St.', 20000),
(2222, 'Smith', '123 Elm St.', 25000),
(3333, 'Brown', '234 Oak St.', 30000)\}
\]

Each of the elements of a relation is also referred to as a tuple.

A relational database scheme, \( D \), is a finite set of relation schemes,

\[
\{ R_1, \ldots, R_m \}.
\]

A relational database on scheme \( D \) is a set of relations

\[
\{ r_1, \ldots, r_m \}
\]

where each \( r_i \) is a relation on the corresponding scheme \( R_i \).

### 1.2 Integrity Constraints

In addition to the data content, a relational database consists of a set of conditions, referred to as integrity constraints, that must be met or satisfied by the data content at all times. The relational database is referred to as a valid database if its data content satisfies all the integrity constraints specified in its definition. Individual relations are referred to valid if they satisfy all the constraints imposed on them. Three very basic and important types of constraints are discussed here.
1.2. INTEGRITY CONSTRAINTS

Primary Key: A key for a relation scheme R is any subset, K of R that satisfies the property that in every valid relation under the scheme R, it is not possible to have two different tuples with the same values under K. A candidate key for R is any key for R such that none of its proper subsets is also a key. It is the case that every relation scheme has at least one candidate key. The primary key for a relation scheme R is one of the candidate keys chosen by the designer of the database. For the EMPLOYEES relation scheme, the EMPID attribute by itself is the primary key. It is not always the case that the primary key is a singleton. In many situations, the primary key consists of more than one attribute. The primary key attributes are required to satisfy the not null constraint, i.e., no tuple can have a null value under the primary key attributes. This property of primary keys is often referred to as the entity integrity rule.

Referential Integrity/Foreign Key: The referential integrity constraint is a condition that is specified across two relations. During the design of a relational database, the designer may create a relation scheme R which includes the primary key attributes of another relation scheme, say S. In such a situation, the referential integrity constraint specifies the condition that the values that appear under the primary key attributes in any valid relation under scheme R must also appear in the relation under scheme S. The attributes in the scheme R that correspond to the primary key attributes of scheme S collectively are referred to as a foreign key in scheme R. Unlike the primary key attributes, the foreign key attributes do not have to satisfy the not null constraint.

As an example, consider the EMPLOYEES relation scheme and the two additional relation schemes

\[
\text{PROJECTS} = (\text{PROJID}, \text{PROJNAME}, \text{LOCATION})
\]

\[
\text{WORKSIN} = (\text{EID}, \text{PROJID}, \text{HOURS}).
\]

The PROJECTS relation scheme represents information about different projects and the WORKSIN relation scheme represents information about which employee works for which projects and
how many hours do they work. It is clear that the primary key for PROJECTS is the lone attribute PROJID. The primary key for the WORKSIN relation scheme, which represents a relationship between EMPLOYEES and PROJECTS, is the combination of EMPID and PROJID attributes. It is assumed that a single employee can work for multiple projects and that a project certainly can have many employees. The relation scheme WORKSIN includes primary key attributes from the EMPLOYEES relation scheme and the PROJECTS relation scheme. The referential integrity constraint in this situation dictates that if an EMPID value is present in a valid relation under the WORKSIN scheme then the same value must also be present in the relation under the EMPLOYEES scheme. In a similar manner, a PROJID value in a valid relation under the WORKSIN scheme must also be present in the relation under the PROJECTS scheme.

**Not Null:** This constraint specifies the condition that tuple values under certain attributes (specified to be not null) cannot be null. This condition is usually always imposed on the primary key attributes\(^1\). Other attributes may also be constrained to be not null if the need arises. In the EMPLOYEES relation scheme, the attributes EMPID and ENAME are likely candidates on which the not null constraint should be imposed.

### 1.3 Tabular View of a Relation

Informally, a relation as defined earlier, can also be viewed as a table made up of rows and columns. The columns are labeled with the attribute names of the relation scheme and the rows correspond to individual tuples of the relation. For example the sample relation under the EMPLOYEES relation scheme can be viewed as the following table:

\(^1\)In Oracle, primary key attributes are automatically constrained to be not null.
with four columns labeled with the attribute names in the relation scheme and three rows corresponding to the three tuples in the relation. Since the relation scheme and the tuples are defined to be sequences it is important to keep the order of the components within a row in correspondence with the column names.

1.4 Sample Databases

Two databases are described in this section. The first one represents information that is usually kept in the grade books of instructors of courses in a university. The second database represents information that is usually maintained by a mail order company which sells products by mail.

Grade book database

The grade book database consists of six relations defined on the six schemes shown in Figure 1.1.

<table>
<thead>
<tr>
<th>EMPID</th>
<th>ENAME</th>
<th>ADDRESS</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Jones</td>
<td>111 Ash St.</td>
<td>20000</td>
</tr>
<tr>
<td>2222</td>
<td>Smith</td>
<td>123 Elm St.</td>
<td>25000</td>
</tr>
<tr>
<td>3333</td>
<td>Brown</td>
<td>234 Oak St.</td>
<td>30000</td>
</tr>
</tbody>
</table>

CATALOG(CNO, CTITLE)
STUDENTS(SID, FNAME, LNAME, MINIT)
COURSES(TERM, LINO, CNO, A, B, C, D)
COMPONENTS(TERM, LINO, COMPCODE, MAXPOINTS, WEIGHT)
ENROLLS(SID, TERM, LINO)
SCORES(SID, TERM, LINO, COMPCODE, POINTS)

Figure 1.1: Grade Book Database Scheme

- The CATALOG relation keeps information about course numbers and course titles of courses taught by a particular instructor. CNO is the primary key for the CATALOG relation.
• The **STUDENTS** relation keeps information about the students of a particular instructor. The **SID** attribute is the primary key for the **STUDENTS** relation.

• The **COURSES** relation keeps information about the various courses that have been taught by a particular instructor. The **TERM** attribute corresponds to the term (such as Fall97 or Spring98) in which the course was taught, the **LINENO** is a unique section number assigned by the registrar of the university within a term. The combination of **TERM** and **LINENO** is the primary key for this relation. The attributes **A**, **B**, **C** and **D** are numeric attributes that hold as values the scores at which the corresponding grades are assigned (for example **A** = 90, **B** = 80, **C** = 70, **D** = 60). The **CNO** attribute is a foreign key in this relation as it appears as a primary key in **CATALOG**.

• The **COMPONENTS** relation keeps information about the various grading components (such as homework, quizzes, exams etc.) for a particular course taught by the instructor. For each course taught, identified by the attributes **TERM** and **LINENO**, this relation records information about the grading component, the maximum points assigned to this component and the weight of this component relative to the other components. Since each course may have multiple components, the combination of the attributes **TERM**, **LINENO** and **COMPNAME** forms the primary key. Since the combination of **TERM** and **LINENO** appearing in this relation is a primary key for the **COURSES** relation, it is classified as a foreign key.

• The **ENROLLS** relation records information about which student was enrolled in which course taught by the instructor. The combination of all three attributes (**SID**, **TERM** and **LINENO**) forms the primary key. There are two foreign keys in this relation: **SID** referring to the **STUDENTS** relation and the combination of **TERM** and **LINENO** referring to the **COURSES** relation.

• The **SCORES** relation records the grading component scores (or points) for each student enrolled in a course. The combination
of the attributes S ID, T ERM, L INENO and C OMPNAME forms the primary key for this relation. There are two foreign keys in this relation: the combination of S ID, T ERM and L INENO referring to the ENROLLS relation and the combination of attributes T ERM, L INENO and C OMPNAME referring to the C OMPONENTS relation.

A sample instance of the grade book database is shown in Figure 1.2.

\begin{center}
catalog
\begin{tabular}{|l|l|}
\hline
CNO & CTITLE \\
\hline
csc226 & Introduction to Programming I \\
csc227 & Introduction to Programming II \\
csc343 & Assembly Programming \\
csc481 & Automata and Formal Languages \\
csc498 & Introduction to Database Systems \\
csc880 & Deductive Databases and Logic Programming \\
\hline
\end{tabular}
\end{center}

\begin{center}
students
\begin{tabular}{|l|l|l|l|}
\hline
SID & F NAME & L NAME & M INIT \\
\hline
1111 & Nandita & Rajshekhar & K \\
2222 & Sydney & Corn & A \\
3333 & Susan & Williams & B \\
4444 & Naveen & Rajshekhar & B \\
5555 & Elad & Yam & G \\
6666 & Lincoln & Herring & F \\
\hline
\end{tabular}
\end{center}

\begin{center}
courses
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline
TERM & LINENO & CNO & A & B & C & D \\
\hline
f96 & 1031 & csc226 & 90 & 80 & 65 & 50 \\
f96 & 1032 & csc226 & 90 & 80 & 65 & 50 \\
sp97 & 1031 & csc227 & 90 & 80 & 65 & 50 \\
\hline
\end{tabular}
\end{center}

Figure 1.2: Grade Book Database Instance
### components

<table>
<thead>
<tr>
<th>TERM</th>
<th>LINENO</th>
<th>COMPNAME</th>
<th>MAXPOINTS</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>f96</td>
<td>1031</td>
<td>exam1</td>
<td>100</td>
<td>30</td>
</tr>
<tr>
<td>f96</td>
<td>1031</td>
<td>quizzes</td>
<td>80</td>
<td>20</td>
</tr>
<tr>
<td>f96</td>
<td>1031</td>
<td>final</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>f96</td>
<td>1032</td>
<td>programs</td>
<td>400</td>
<td>40</td>
</tr>
<tr>
<td>f96</td>
<td>1032</td>
<td>midterm</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>f96</td>
<td>1032</td>
<td>final</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>sp97</td>
<td>1031</td>
<td>paper</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>sp97</td>
<td>1031</td>
<td>project</td>
<td>100</td>
<td>50</td>
</tr>
</tbody>
</table>

### enrolls

<table>
<thead>
<tr>
<th>SID</th>
<th>TERM</th>
<th>LINENO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>f96</td>
<td>1031</td>
</tr>
<tr>
<td>2222</td>
<td>f96</td>
<td>1031</td>
</tr>
<tr>
<td>4444</td>
<td>f96</td>
<td>1031</td>
</tr>
<tr>
<td>1111</td>
<td>f96</td>
<td>1032</td>
</tr>
<tr>
<td>2222</td>
<td>f96</td>
<td>1032</td>
</tr>
<tr>
<td>3333</td>
<td>f96</td>
<td>1032</td>
</tr>
<tr>
<td>5555</td>
<td>sp97</td>
<td>1031</td>
</tr>
<tr>
<td>6666</td>
<td>sp97</td>
<td>1031</td>
</tr>
</tbody>
</table>

Figure 1.2: Grade Book Database Instance (Continued)
## 1.4. SAMPLE DATABASES

<table>
<thead>
<tr>
<th>SID</th>
<th>TERM</th>
<th>LINENO</th>
<th>COMPNAME</th>
<th>POINTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>f96</td>
<td>1031</td>
<td>exam1</td>
<td>90</td>
</tr>
<tr>
<td>1111</td>
<td>f96</td>
<td>1031</td>
<td>quizzes</td>
<td>75</td>
</tr>
<tr>
<td>1111</td>
<td>f96</td>
<td>1031</td>
<td>final</td>
<td>95</td>
</tr>
<tr>
<td>2222</td>
<td>f96</td>
<td>1031</td>
<td>exam1</td>
<td>70</td>
</tr>
<tr>
<td>2222</td>
<td>f96</td>
<td>1031</td>
<td>quizzes</td>
<td>40</td>
</tr>
<tr>
<td>2222</td>
<td>f96</td>
<td>1031</td>
<td>final</td>
<td>82</td>
</tr>
<tr>
<td>4444</td>
<td>f96</td>
<td>1031</td>
<td>exam1</td>
<td>83</td>
</tr>
<tr>
<td>4444</td>
<td>f96</td>
<td>1031</td>
<td>quizzes</td>
<td>71</td>
</tr>
<tr>
<td>4444</td>
<td>f96</td>
<td>1031</td>
<td>final</td>
<td>74</td>
</tr>
<tr>
<td>1111</td>
<td>f96</td>
<td>1032</td>
<td>programs</td>
<td>400</td>
</tr>
<tr>
<td>1111</td>
<td>f96</td>
<td>1032</td>
<td>midterm</td>
<td>95</td>
</tr>
<tr>
<td>1111</td>
<td>f96</td>
<td>1032</td>
<td>final</td>
<td>99</td>
</tr>
<tr>
<td>2222</td>
<td>f96</td>
<td>1032</td>
<td>programs</td>
<td>340</td>
</tr>
<tr>
<td>2222</td>
<td>f96</td>
<td>1032</td>
<td>midterm</td>
<td>65</td>
</tr>
<tr>
<td>2222</td>
<td>f96</td>
<td>1032</td>
<td>final</td>
<td>95</td>
</tr>
<tr>
<td>3333</td>
<td>f96</td>
<td>1032</td>
<td>programs</td>
<td>380</td>
</tr>
<tr>
<td>3333</td>
<td>f96</td>
<td>1032</td>
<td>midterm</td>
<td>75</td>
</tr>
<tr>
<td>3333</td>
<td>f96</td>
<td>1032</td>
<td>final</td>
<td>88</td>
</tr>
<tr>
<td>5555</td>
<td>sp97</td>
<td>1031</td>
<td>paper</td>
<td>80</td>
</tr>
<tr>
<td>5555</td>
<td>sp97</td>
<td>1031</td>
<td>project</td>
<td>90</td>
</tr>
<tr>
<td>6666</td>
<td>sp97</td>
<td>1031</td>
<td>paper</td>
<td>80</td>
</tr>
<tr>
<td>6666</td>
<td>sp97</td>
<td>1031</td>
<td>project</td>
<td>85</td>
</tr>
</tbody>
</table>

Figure 1.2: Grade Book Database Instance (Continued)
CHAPTER 1. RELATIONAL DATA MODEL

Mail order database

The mail order database consists of six relations defined on the six schemes shown in Figure 1.3:

- **EMPLOYEES** (ENO, ENAME, ZIP, HDATE)
- **PARTS** (PNO, PNAME, QOH, PRICE, LEVEL)
- **CUSTOMERS** (CNO, CNAME, STREET, ZIP, PHONE)
- **ORDERS** (ONO, CNO, ENO, RECEIVED, SHIPPED)
- **ODETAILS** (ONO, PNO, QTY)
- **ZIPCODES** (ZIP, CITY)

Figure 1.3: Mail Order Database Scheme

- The **EMPLOYEES** relation keeps information about the employees of the mail order company. The ENO attribute is the primary key. The ZIP attribute is a foreign key referring to the ZIPCODES table.

- The **PARTS** relation keeps a record of the inventory of the company. For each part, besides its number and name, the quantity on hand, unit price and the reorder level are recorded. PNO is the primary key for this relation.

- The **CUSTOMERS** relation keeps information about the customers of the mail order company. Each customer is assigned a customer number, CNO, which serves as the primary key. The ZIP attribute is a foreign key referring the ZIPCODES relation.

- The **ORDERS** relation contains information about the orders placed by customers, the employee who took the order, the order receive and ship dates. ONO is the primary key. The CNO attribute is a foreign key referring the CUSTOMERS relation and the ENO attribute is a foreign key referring the EMPLOYEES table.

- The **ODETAILS** relation contains information about the various parts ordered by the customer within a particular order. The combination of ONO and PNO attributes forms the primary key.
The ONO attribute is a foreign key referring the ORDERS relation and the PNO attribute is a foreign key referring the PARTS relation.

- The ZIPCODES relation maintains information about the zip codes for various cities. ZIP is the primary key.

A sample instance of the mail order database is shown in Figure 1.4.

<table>
<thead>
<tr>
<th>employees</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ENO</td>
<td>ENAME</td>
<td>ZIP</td>
<td>HDATE</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>Jones</td>
<td>67226</td>
<td>12-DEC-95</td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>Smith</td>
<td>60606</td>
<td>01-JAN-92</td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>Brown</td>
<td>50302</td>
<td>01-SEP-94</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>parts</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PNO</td>
<td>PNAME</td>
<td>QOH</td>
<td>PRICE</td>
<td>LEVEL</td>
</tr>
<tr>
<td>10506</td>
<td>Land Before Time I</td>
<td>200</td>
<td>19.99</td>
<td>20</td>
</tr>
<tr>
<td>10507</td>
<td>Land Before Time II</td>
<td>156</td>
<td>19.99</td>
<td>20</td>
</tr>
<tr>
<td>10508</td>
<td>Land Before Time III</td>
<td>190</td>
<td>19.99</td>
<td>20</td>
</tr>
<tr>
<td>10509</td>
<td>Land Before Time IV</td>
<td>60</td>
<td>19.99</td>
<td>20</td>
</tr>
<tr>
<td>10601</td>
<td>Sleeping Beauty</td>
<td>300</td>
<td>24.99</td>
<td>20</td>
</tr>
<tr>
<td>10701</td>
<td>When Harry Met Sally</td>
<td>120</td>
<td>19.99</td>
<td>30</td>
</tr>
<tr>
<td>10800</td>
<td>Dirty Harry</td>
<td>140</td>
<td>14.99</td>
<td>30</td>
</tr>
<tr>
<td>10900</td>
<td>Dr. Zhivago</td>
<td>100</td>
<td>24.99</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>customers</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CNO</td>
<td>CNAME</td>
<td>STREET</td>
<td>ZIP</td>
<td>PHONE</td>
</tr>
<tr>
<td>1111</td>
<td>Charles</td>
<td>123 Main St.</td>
<td>67226</td>
<td>316-636-5555</td>
</tr>
<tr>
<td>2222</td>
<td>Bertram</td>
<td>237 Ash Avenue</td>
<td>67226</td>
<td>316-689-5555</td>
</tr>
<tr>
<td>3333</td>
<td>Barbara</td>
<td>111 Inwood St.</td>
<td>60606</td>
<td>316-111-1234</td>
</tr>
</tbody>
</table>

Figure 1.4: Mail Order Database Instance
### orders

<table>
<thead>
<tr>
<th>ONO</th>
<th>CNO</th>
<th>ENO</th>
<th>RECEIVED</th>
<th>SHIPPED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1020</td>
<td>1111</td>
<td>1000</td>
<td>10-DEC-94</td>
<td>12-DEC-94</td>
</tr>
<tr>
<td>1021</td>
<td>1111</td>
<td>1000</td>
<td>12-JAN-95</td>
<td>15-JAN-95</td>
</tr>
<tr>
<td>1022</td>
<td>2222</td>
<td>1001</td>
<td>13-FEB-95</td>
<td>20-FEB-95</td>
</tr>
<tr>
<td>1023</td>
<td>3333</td>
<td>1000</td>
<td>20-JUN-97</td>
<td>null</td>
</tr>
</tbody>
</table>

### odetails

<table>
<thead>
<tr>
<th>ONO</th>
<th>PNO</th>
<th>QTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1020</td>
<td>10506</td>
<td>1</td>
</tr>
<tr>
<td>1020</td>
<td>10507</td>
<td>1</td>
</tr>
<tr>
<td>1020</td>
<td>10508</td>
<td>2</td>
</tr>
<tr>
<td>1020</td>
<td>10509</td>
<td>3</td>
</tr>
<tr>
<td>1021</td>
<td>10601</td>
<td>4</td>
</tr>
<tr>
<td>1022</td>
<td>10601</td>
<td>1</td>
</tr>
<tr>
<td>1022</td>
<td>10701</td>
<td>1</td>
</tr>
<tr>
<td>1023</td>
<td>10800</td>
<td>1</td>
</tr>
<tr>
<td>1023</td>
<td>10900</td>
<td>1</td>
</tr>
</tbody>
</table>

### zipcodes

<table>
<thead>
<tr>
<th>ZIP</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>67226</td>
<td>Wichita</td>
</tr>
<tr>
<td>60606</td>
<td>Fort Dodge</td>
</tr>
<tr>
<td>50302</td>
<td>Kansas City</td>
</tr>
<tr>
<td>54444</td>
<td>Columbia</td>
</tr>
<tr>
<td>66002</td>
<td>Liberal</td>
</tr>
<tr>
<td>61111</td>
<td>Fort Hays</td>
</tr>
</tbody>
</table>

Figure 1.4: Mail Order Database Instance (Continued)
1.5 Relational Algebra

The relational algebra is a set of algebraic operations that take as input one (for unary operators) or two (for binary operators) relations and return as output a relation. Using these operations, one can answer arbitrary ad-hoc queries against the database content. A good understanding of the relational algebra makes the task of phrasing complex queries in SQL much easier. The relational algebraic operators are briefly introduced next and then some queries against the sample databases are answered using these operations.

The relational operators are usually classified into two categories: set-theoretic operations and relation-theoretic operations.

1.5.1 Set-theoretic operations

The set-theoretic operations include union, difference, Cartesian product, and intersection. These operations are borrowed from mathematical set theory and are applicable in the relational model because relations are nothing but sets of tuples.

The union, difference and intersection operators are binary operators that operate on two union-compatible relations. Two relations are union-compatible if they have the same number of attributes and the domains of the corresponding attributes in the two relations are the same. Consider two relations $r$ and $s$ that are union-compatible. The set-theoretic operations are defined as follows:

**Union:** $r \cup s = \{t | t \in r \text{ or } t \in s \}$

In other words, the union of two union-compatible relations contains all the tuples from each of the relations.

**Difference:** $r - s = \{t | t \in r \text{ and } t \notin s \}$

The difference of two union-compatible relations contains all those tuples in the first relation that are not present in the second relation.

**Intersection:** $r \cap s = \{t | t \in r \text{ and } t \in s \}$

The intersection of two union-compatible relations contains all the tuples that are contained in both the relations.
**Cartesian Product:** The Cartesian product is a binary operator that takes as input two relations (r and s on any schemes) and produces a relation on the scheme that is the concatenation of the relation schemes of the input relations. The tuples in the Cartesian product are constructed by concatenating each tuple in the first input relation with each tuple in the second input relation. Formally,

\[ r \times s = \{t1.t2|t1 \in r \text{ and } t2 \in s\}, \]

where, \( t1.t2 \) is the concatenation of tuples \( t1 \) and \( t2 \) to form a larger tuple.

Examples of the set-theoretic operations are shown in Figure 1.5. Since the attribute names in a relation scheme must be unique, the scheme of the Cartesian product of relations \( r \) and \( s \) in the example contains attribute names prefixed by \( r \) and \( s \).

### 1.5.2 Relation-theoretic operations

The relation-theoretic operations include *rename*, *select*, *project*, *natural join*, and *division* among others.

**Rename:** The rename operator takes as input a relation and returns the same relation as output, but under a different name. This operation is useful and necessary for queries which need to refer to the same relation more than once. The symbolic notation for the rename operator is \( \rho_s(r) \), where \( r \) is the input relation and \( s \) is the new name.

**Select:** The select operator acts as a horizontal filter for relations. Given a selection condition, the select operator produces an output relation which consists of only those tuples from the input relation that satisfy the selection condition. Symbolically, the select operator is written as \( \sigma_F(r) \), where \( F \) is the selection criterion and \( r \) is the input relation. Formally, the select operator is defined as follows:

\[ \sigma_F(r) = \{t|t \in r \text{ and } t \text{ satisfies } F\}. \]
Figure 1.5: Set-theoretic Operators
**Project:** The project operator acts as a vertical filter for relations. Given a sub-list of attribute names of a relation, the project operator keeps only those values that correspond to the sub-list of attribute names and discards other values in tuples. Symbolically, the project operator is written as $\pi_A(r)$, where $A$ is a sub-list of the attributes of $r$. Formally, the project operator is defined as follows:

$$\pi_A(r) = \{t[A] | t \in r\}$$

where $t[A]$ is a tuple constructed from $t$ by keeping the values that correspond to the attributes in $A$ and discarding other values.

**Natural Join:** The natural join operator takes as input two relations and produces as output a relation whose scheme is the concatenation of the two schemes of the input relations with any duplicate attribute names discarded. A tuple in the first input relation is said to match a tuple in the second input relation if they have the same values under the common attributes. The tuples in the natural join are constructed by concatenating each tuple in the first input relation with each matching tuple in the second input relation and discarding the values under the common attributes of the second relation. Symbolically, the natural join is written as $r \bowtie s$, where $r$ is a relation on scheme $R$ and $s$ is a relation on scheme $S$. Formally, the natural join operation is defined as follows:

$$r \bowtie s = \{t | (∃u \in r)(∃v \in s)(t[R] = u \text{ and } t[S] = v)\}$$

**Division:** The division operator takes as input two relations, called the dividend relation ($r$ on scheme $R$) and the divisor relation ($s$ on scheme $S$) such that all the attributes in $S$ also appear in $R$ and $s$ is not empty. The output of the division operation is a relation on the scheme $R$ with all the attributes common with $S$ discarded. A tuple $t$ is put in the output of the operation if for all tuples $u$ in $s$, the tuple $tu$ is in $r$, where $tu$ is a tuple constructed from $t$ and $u$ by combining the individual values in these tuples in the proper order to form a tuple in $r$. Symbolically, the division operation is written as $r \div s$ and is defined as follows:
1.5. RELATIONAL ALGEBRA

\[ r \div s = \{ t | (\forall u \in s)(tu \in r) \} \]

Examples of the relation-theoretic operations are shown in Figure 1.6.

\[
\begin{array}{|c|c|c|}
\hline
A & C & D \\
\hline a & c & d \\
a & e & f \\
a & g & h \\
b & c & d \\
b & g & h \\
c & c & d \\
c & e & f \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
C & D \\
\hline c & d \\
e & f \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|}
\hline
B & C & D \\
\hline b & c & d \\
b & e & f \\
\hline
\end{array}
\]

Figure 1.6: Relation-theoretic Operations

Among the relational operators presented so far, there are six basic operators: union, difference, Cartesian product, rename, select, and project. This basic set of operations has the property that none of them can be expressed in terms of the others. The remaining operators presented, namely intersection, natural join, and division can be expressed in terms of the basic operators as follows:

**Intersection:** \[ r \cap s = r - (r - s) \]
**Natural Join:** \( r \bowtie s = \pi_{R \cap S}(\sigma_F(r \times s)) \)

where \( F \) is a selection condition which indicates that the tuple values under the common attributes of \( r \) and \( s \) are equal.

**Division:** \( r \div s = \pi_{R-S}(r) - \pi_{R-S}((\pi_{R-S}(r) \times s) - r) \)

Even though relation schemes are defined as sequences, they are treated as sets in these equalities for simplicity.

An explanation for the equality for division is in order! First, all candidate tuples for the result are calculated by the expression

\[ \pi_{R-S}(r) \]

Next, these candidate tuples are combined with all tuples of \( s \) in the following expression

\[ \pi_{R-S}(r) \times s \]

to give a relation containing all combinations of candidate tuples with all tuples of \( s \). Since we are looking for tuples under the scheme \( R - S \) which combine with all tuples of \( s \) and are also present in \( r \), if we subtract \( r \) from the previous expression, we will get all the combinations of tuples that are “missing” in \( r \). By projecting these tuples on \( R - S \), we get all those tuples that should not go to the result in the following expression.

\[ \pi_{R-S}((\pi_{R-S}(r) \times s) - r) \]

Finally, we subtract this set from the set of all candidate tuples and obtain the output relation of the division operator.

### 1.5.3 Querying using relational algebra

The following is a list of queries against the two sample databases and the corresponding relational algebraic expressions that return the answers to the queries. The relational algebraic expressions are broken up into smaller parts and are assigned to temporary variables. One could easily write one whole expression from these individual parts and thereby not requiring the assignment primitive.
1.5. RELATIONAL ALGEBRA

Grade book database queries

Q1 Get the names of students enrolled in the Assembly Programming class in the f96 term.

\[ t_1 := \sigma_{\text{TITLE='Assembly Programming}}(\text{catalog}) \]
\[ t_2 := \sigma_{\text{TERM='f96}}(\text{courses}) \]
\[ t_3 := t_1 \bowtie t_2 \bowtie \text{enrolls} \bowtie \text{students} \]
\[ \text{result} := \pi_{\text{FNAME,LNAME,MINIT}}(t_3) \]

Q2 Get the SID values of students who did not enroll in any class during the f96 term.

\[ \pi_{\text{SID}}(\text{students}) - \pi_{\text{SID}}(\sigma_{\text{TERM='f96}}(\text{enrolls})) \]

Q3 Get the SID values of students who have enrolled in csc226 and csc227,

\[ t_1 := \pi_{\text{SID}}(\text{enrolls} \bowtie \sigma_{\text{CNO='csc226}}(\text{courses})) \]
\[ t_2 := \pi_{\text{SID}}(\text{enrolls} \bowtie \sigma_{\text{CNO='csc227}}(\text{courses})) \]
\[ \text{result} := t_1 \cap t_2 \]

Q4 Get the SID values of students who have enrolled in csc226 or csc227,

\[ t_1 := \pi_{\text{SID}}(\text{enrolls} \bowtie \sigma_{\text{CNO='csc226}}(\text{courses})) \]
\[ t_2 := \pi_{\text{SID}}(\text{enrolls} \bowtie \sigma_{\text{CNO='csc227}}(\text{courses})) \]
\[ \text{result} := t_1 \cup t_2 \]

Q5 Get the SID values of students who have enrolled in all the courses in the catalog.

\[ \pi_{\text{SID,CNO}}(\text{courses} \bowtie \text{enrolls}) \div \pi_{\text{CNO}}(\text{catalog}) \]
Mail order database queries

Q6 Get part names of parts that cost less than 20.00.

\[
\pi_{\text{PNAME}}(\sigma_{\text{PRICE}<20.00}(\text{parts}))
\]

Q7 Get pairs of CNO values of customers who have the the same zip-code.

\[
t_1 := \rho_{c_1} (\text{customers}) \times \rho_{c_2} (\text{customers})
\]

\[
t_2 := \sigma_{c_1.ZIP=c_2.ZIP \text{ and } c_1.CNO<c_2.CNO} (t_1)
\]

\[
\text{result} := \pi_{c_1.CNO,c_2.CNO}(t_2)
\]

Q8 Get the names of customers who have ordered parts from employees living in Wichita.

\[
t_1 := \pi_{\text{ENO}} (\text{employees} \bowtie \sigma_{\text{CITY}=\text{Wichita}} (\text{zipcodes}))
\]

\[
\text{result} := \pi_{\text{CNAME}} (\text{customers} \bowtie \text{orders} \bowtie t_1)
\]

Q9 Get CNO values of customers who have ordered parts only from employees living in Wichita.

\[
t_1 := \pi_{\text{ENO}} (\text{employees} \bowtie \sigma_{\text{CITY} \neq \text{Wichita}} (\text{zipcodes}))
\]

\[
\text{result} := \pi_{\text{CNO}} (\text{orders}) - \pi_{\text{CNO}} (\text{orders} \bowtie t_1)
\]

Q10 Get CNO values of customers who have ordered parts from all employees living in Wichita.

\[
t_1 := \pi_{\text{ENO}} (\text{employees} \bowtie \sigma_{\text{CITY}=\text{Wichita}} (\text{zipcodes}))
\]

\[
\text{result} := \pi_{\text{CNO,ENO}} (\text{orders}) \div t_1
\]