Relational Data Model

Operators

Relational Database Principles
Ritchie, Ch. 2

Modern Database Management
McFadden, Ch. 9

Database Processing
Kroenke, Ch. 6, 9 & 10

Database Management Systems
Ramakrishnan, Ch. 3 & 4, Sec. 5.1 & 5.2

Overview

- Relational Algebra
  - Definition
  - Closure
  - SQL

- Major Operators
  - Projection
  - Restriction
  - Join

- Other Operators
  - Union
  - Intersection
**Definition**

A relational system is a system in which:

1. The data is perceived by the user as tables (and nothing but tables); and
2. The operators at the user’s disposal (e.g. for data retrieval) are operators that generate new tables from old.

C. J. Date

---

- All data in the relational data model (RDM) is stored in one or more tables, called relations. This is important because having all data stored in tables means that all the data is accessed in exactly the same way.
- The RDM provides a set of operators that can be used to manipulate and retrieve data. The operators take as input a set of tables, change them in some way and produce another set of tables as the result.
  - This is important because each operator can take as input the output of another operator.
  - This is similar to the use of operators, such as addition or multiplication, in mathematics. For example, the results of two additions can be added or multiplied together.

---

**Relational Algebra**

- A language for manipulating relations in a database.

- **Definition**
  “a collection of operators that take relations as their operands and return relations as their results”

C. J. Date

- When the RDM was defined a language for manipulating the relations was also defined, called relational algebra.
- Relational algebra is a set of operators that can be used to manipulate a set of relations. For example, the relational algebra operator restriction copies a set of tuples from a relation and places them in a new relation.
- The set of operators defined in relational algebra is the minimum set of operators that should be available in any relational database management system (RDBMS).
- Most RDBMSs implement a user-friendly version of relational algebra called SQL.
Closure

- A relational algebra operation takes as input one or more relations and outputs one or more relations

\[ \text{Table 1} + \text{Table 2} = \text{New Table} \]

- Therefore, the output of one operation can be used as the input to another.

SQL

- Database language
  - A set of commands for querying and manipulating databases.
  - Equivalent to relational algebra.
- Supports
  - Creating database structures
  - Changing data in the database
  - Retrieving data from the database

- Most RDBMSs do not use relational algebra directly. Instead they use an equivalent database language called SQL (Structured Query Language).

- SQL is a standard language that provides commands for:
  - Creating and changing database structures. For example, adding a new relation or deleting and existing relation.
  - Changing data in the database. For example, inserting a new tuple into a relation.
  - Retrieving data from the database. For example, retrieving all tuples that match a certain value.
- McFadden et al proposes the following advantages for using standard SQL:
  - Reduced training costs.
  - Increased productivity
  - Application portability
  - Application longevity
  - Reduced dependence on a single vendor
  - Cross-system communication
Overview

- Relational Algebra
  - Definition
  - Closure
  - SQL
- Major Operators
  - Projection
  - Restriction
  - Join
- Other Operators
  - Union
  - Intersection

Projection

- Extract a subset of the attributes (columns) in a relation to create a third relation.
- Relational Algebra syntax
  \[ R_1 = \text{PROJECT} (R) (a_1, a_2, a_3) \]
- SQL syntax
  ```
  SELECT a_1, a_2, a_3
  FROM R
  ```
**Projection Example**

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>DEPTNO</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Smith</td>
<td>D1</td>
<td>10000</td>
</tr>
<tr>
<td>E2</td>
<td>Jones</td>
<td>D1</td>
<td>14000</td>
</tr>
<tr>
<td>E3</td>
<td>Brown</td>
<td>D2</td>
<td>8000</td>
</tr>
<tr>
<td>E4</td>
<td>White</td>
<td>D2</td>
<td>9000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ENAME</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>10000</td>
</tr>
<tr>
<td>Jones</td>
<td>14000</td>
</tr>
<tr>
<td>Brown</td>
<td>8000</td>
</tr>
<tr>
<td>White</td>
<td>9000</td>
</tr>
</tbody>
</table>

**Restriction**

- Extracting some of the tuples (rows) in a relation to create a third relation.
- Relational Algebra Syntax
  \[ R_1 = \text{RESTRICT} (R) \ (X \ # \ Y) \]
- SQL Syntax
  
  ```sql
  SELECT *
  FROM R
  WHERE X \ # \ Y
  ```

  # - comparison operator, e.g. equal to, less than, etc.
Restriction Example

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>DEPTNO</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Smith</td>
<td>D1</td>
<td>10000</td>
</tr>
<tr>
<td>E2</td>
<td>Jones</td>
<td>D1</td>
<td>14000</td>
</tr>
<tr>
<td>E3</td>
<td>Brown</td>
<td>D2</td>
<td>8000</td>
</tr>
<tr>
<td>E4</td>
<td>White</td>
<td>D2</td>
<td>9000</td>
</tr>
</tbody>
</table>

Relational Algebra

\[ R_1 = \text{Restrict}(\text{EMP})(\text{DEPTNO} = 'D2') \]

SQL

\[ \text{SELECT * FROM EMP WHERE DEPTNO = 'D2'} \]

Join

- Combining two relations to create a third relation.
- The relations are joined by matching a common pair of attributes in both relations.
  - Common attributes share the same domain.
- Relational Algebra Syntax

\[ R = \text{JOIN}(A)(B)(a_1 = b_1) \]

- SQL Syntax

\[
\begin{align*}
\text{SELECT *} \\
\text{FROM } A, B \\
\text{WHERE } A.a_1 = B.b_1
\end{align*}
\]
Join Example

<table>
<thead>
<tr>
<th>DEPT</th>
<th>EMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>E1: Smith</td>
</tr>
<tr>
<td>D2</td>
<td>E2: Jones</td>
</tr>
<tr>
<td>D3</td>
<td>E3: Brown</td>
</tr>
<tr>
<td></td>
<td>E4: White</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEPTNO</th>
<th>DNAME</th>
<th>BUDGET</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Marketing</td>
<td>10M</td>
</tr>
<tr>
<td>D2</td>
<td>Development</td>
<td>12M</td>
</tr>
<tr>
<td>D3</td>
<td>Research</td>
<td>5M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>DEPTNO1</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Smith</td>
<td>D1</td>
<td>10000</td>
</tr>
<tr>
<td>E2</td>
<td>Jones</td>
<td>D1</td>
<td>14000</td>
</tr>
<tr>
<td>E3</td>
<td>Brown</td>
<td>D2</td>
<td>8000</td>
</tr>
<tr>
<td>E4</td>
<td>White</td>
<td>D2</td>
<td>9000</td>
</tr>
</tbody>
</table>

Relational Algebra

\[ R = \text{JOIN}(\text{EMP}) \ (\text{DEPT}) \ (\text{DEPTNO}=\text{DEPTNO2}) \]

SQL

\[ \text{SELECT } * \text{ FROM EMP, DEPT WHERE EMP.DEPTNO=DEPT.DEPTNO} \]

Overview

- Relational Algebra
  - Definition
  - Closure
  - SQL
- Major Operators
  - Projection
  - Restriction
  - Join
- Other Operators
  - Union
  - Intersection
**Union**

- Adding together the tuples (rows) in two relations to create a third relation.
  - The structure (attributes) of the two relations must be the same.
- Relational Algebra Syntax
  \[ R = \text{UNION} (A) (B) \]
- SQL Syntax
  
  ```sql
  SELECT * FROM A
  UNION
  SELECT * FROM B
  ```

---

**Union Example**

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>DEPTNO</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Smith</td>
<td>D1</td>
<td>10000</td>
</tr>
<tr>
<td>E2</td>
<td>Jones</td>
<td>D1</td>
<td>14000</td>
</tr>
<tr>
<td>E3</td>
<td>Brown</td>
<td>D2</td>
<td>8000</td>
</tr>
<tr>
<td>E4</td>
<td>White</td>
<td>D2</td>
<td>9000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>DEPTNO</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E8</td>
<td>Peters</td>
<td>D3</td>
<td>15000</td>
</tr>
<tr>
<td>E9</td>
<td>Orange</td>
<td>D1</td>
<td>34000</td>
</tr>
<tr>
<td>E10</td>
<td>Pale</td>
<td>D6</td>
<td>12000</td>
</tr>
<tr>
<td>E11</td>
<td>Green</td>
<td>D5</td>
<td>7500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>DEPTNO</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Smith</td>
<td>D1</td>
<td>10000</td>
</tr>
<tr>
<td>E2</td>
<td>Jones</td>
<td>D1</td>
<td>14000</td>
</tr>
<tr>
<td>E3</td>
<td>Brown</td>
<td>D2</td>
<td>8000</td>
</tr>
<tr>
<td>E4</td>
<td>White</td>
<td>D2</td>
<td>9000</td>
</tr>
<tr>
<td>E8</td>
<td>Peters</td>
<td>D3</td>
<td>15000</td>
</tr>
<tr>
<td>E9</td>
<td>Orange</td>
<td>D1</td>
<td>34000</td>
</tr>
<tr>
<td>E10</td>
<td>Pale</td>
<td>D6</td>
<td>12000</td>
</tr>
<tr>
<td>E11</td>
<td>Green</td>
<td>D5</td>
<td>7500</td>
</tr>
</tbody>
</table>

**Relational Algebra**

\[ R = \text{UNION} (\text{EMP}) (\text{RETIRED-EMP}) \]

**SQL**

```sql
SELECT * FROM EMP
UNION
SELECT * FROM RETIRED-EMP
```
Intersection

- Extracting the tuples that are common in two relations to create a third relation.
  - The structure (attributes) of the two relations must be the same.
- Formal Syntax
  \[ R = \text{INTERSECT} (A) \ (B) \]
- SQL Syntax
  
  ```sql
  SELECT * FROM A
  INTERSECT
  SELECT * FROM B
  ```

Intersection Example

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>DEPTNO</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Smith</td>
<td>D1</td>
<td>10000</td>
</tr>
<tr>
<td>E2</td>
<td>Jones</td>
<td>D1</td>
<td>14000</td>
</tr>
<tr>
<td>E3</td>
<td>Brown</td>
<td>D2</td>
<td>8000</td>
</tr>
<tr>
<td>E4</td>
<td>White</td>
<td>D2</td>
<td>9000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MANAGEMENT-EMP</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>DEPTNO</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Smith</td>
<td>D1</td>
<td>10000</td>
</tr>
<tr>
<td>E3</td>
<td>Brown</td>
<td>D2</td>
<td>8000</td>
</tr>
<tr>
<td>E10</td>
<td>Pale</td>
<td>D6</td>
<td>12000</td>
</tr>
<tr>
<td>E11</td>
<td>Green</td>
<td>D2</td>
<td>7500</td>
</tr>
</tbody>
</table>

Relational Algebra

\[ R = \text{INTERSECT} (\text{EMP}) \ (\text{MANAGEMENT-EMP}) \]

SQL

```sql
SELECT * FROM EMP
INTERSECT
SELECT * FROM MANAGEMENT-EMP
```