Database Operations

Chapter 16

Operations on Tables

- A database is a collection of tables
- Operations on tables produce tables
  - The questions we ask of a database are answered with a whole table
- Users specify what they want to know and the database software finds it
- Operations specified using SQL (Structured Query Language)
  - SQL is a language for querying and modifying data and managing databases

Example Database

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Capital</th>
<th>Lat</th>
<th>Lon</th>
<th>Interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ireland</td>
<td>IE</td>
<td>Dublin</td>
<td>52</td>
<td>W</td>
<td>History</td>
</tr>
<tr>
<td>Israel</td>
<td>IL</td>
<td>Jerusalem</td>
<td>31</td>
<td>N</td>
<td>History</td>
</tr>
<tr>
<td>Italy</td>
<td>IT</td>
<td>Rome</td>
<td>42</td>
<td>W</td>
<td>Art</td>
</tr>
<tr>
<td>Jamaica</td>
<td>JM</td>
<td>Kingston</td>
<td>14</td>
<td>W</td>
<td>Beach</td>
</tr>
<tr>
<td>Japan</td>
<td>JP</td>
<td>Tokyo</td>
<td>35</td>
<td>N</td>
<td>Kabuki</td>
</tr>
</tbody>
</table>

Select

- Takes rows from one table to create a new table
- Specify the table from which rows are to be taken, and the test for selection
- Test is applied to each row of the table to determine if it should be included in result table
- Test uses attribute names, constants, and relational operators
- If the test is true for a given row, the row is included in the result table

**Example:**

```
SELECT * 
FROM Nations 
WHERE Interest = 'Beach'
```

Figure 16.7 Part of the table created by selecting countries with a Test for Interest equal to Beach.
**Project**

- Builds a new table from the columns of an existing table.
- Specify name of existing table and the columns (field names) to be included in the new table.
- The new table will have the same number of rows as the original table, unless ...
  - ... the new table eliminates a key field. Duplicate rows in the new table are eliminated.

**Syntax:**

```
SELECT <field list>
FROM <table>
```

**Example:**

```
SELECT Name, Domain, Interest
FROM Nations
```

**Select And Project**

- Can use Select and Project operations together to “trim” base tables to keep only some of the rows and some of the columns.

**Example:**

```
SELECT Name, Domain, Latitude
FROM Nations
WHERE Latitude >= 60 AND NS = 'N'
```

**Select And Project Results**

```
SELECT Name, Domain, Latitude
FROM Nations
WHERE Latitude >= 60 AND NS = 'N'
```

**Exercise**

- What is the capital of countries whose "interest" is "history" or "beach"?

**Solution:**

```
SELECT Capital
FROM Nations
WHERE Interest = 'History'
OR Interest = 'Beach'
```
Union

- Combines two tables (that have the same set of attributes)

- Syntax:
  ```
  <table1>
  UNION
  <table2>
  ```

Union Results

```sql
SELECT *
FROM Nations
WHERE Lat >= 60 AND NS = 'N'
UNION
SELECT *
FROM Nations
WHERE Lat >= 45 AND NS = 'S'
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Dom</th>
<th>Capital</th>
<th>Lat</th>
<th>Lon</th>
<th>SW</th>
<th>Interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Falkland 1</td>
<td>FE</td>
<td>Stanley</td>
<td>51</td>
<td>58</td>
<td>W</td>
<td>Nature</td>
</tr>
<tr>
<td>Finland</td>
<td>FJ</td>
<td>Helsinki</td>
<td>41</td>
<td>24</td>
<td>F</td>
<td>Nature</td>
</tr>
<tr>
<td>Greenland</td>
<td>GL</td>
<td>Nuuk</td>
<td>72</td>
<td>40</td>
<td>G</td>
<td>Nature</td>
</tr>
<tr>
<td>Iceland</td>
<td>IS</td>
<td>Reykjavik</td>
<td>65</td>
<td>10</td>
<td>V</td>
<td>Iceland</td>
</tr>
<tr>
<td>Norway</td>
<td>NO</td>
<td>Oslo</td>
<td>69</td>
<td>10</td>
<td>R</td>
<td>Viking</td>
</tr>
</tbody>
</table>

Product

- Creates a super table with all fields from both tables
- Puts the rows together
  - Each row of Table 2 is appended to each row of Table 1

- General syntax:
  ```
  SELECT *
  FROM <table1>, <table2>
  ```

Product Results

```sql
SELECT *
FROM Nations, Travelers
```

Another Table

![Figure 16.11](image)

Join

- Combines two tables, like the Product Operation, but doesn't necessarily produce all pairings

- Join operation:
  - Table1 Table2 On Match

- Match is a comparison test involving fields from each table (Table.Field)

- A match for a row from each table produces a result row that is their concatenation
Join

- General syntax:
  
  ```sql
  SELECT *
  FROM <table1> INNER JOIN <table2>
  ON <table1>.<field> = <table2>.<field>
  ```

- Can be written with product operation:
  
  ```sql
  SELECT *
  FROM <table1>, <table2>
  WHERE <table1>.<field> = <table2>.<field>
  ```

Join Applied

- For each row in one table, locate a row (or rows) in the other table with the same value in the common field
  - If found, combine the two.
  - If not found, look up the next row.

- Possible to join using any relational operator, not just = (equality) to compare fields

Exercise

- Suppose you have the following tables: `performers`, `events`, and `venues`

- Write a query to find what dates the Paramount is booked.

Solution

```sql
SELECT PerformanceDate
FROM events INNER JOIN venues
ON events.VenueID = venues.ID
WHERE Venue = 'Paramount Theater'
```

Exercise

- Write a query to find which performers are playing at the Paramount and when.
Solution

SELECT Performer, PerformanceDate
FROM (events INNER JOIN venues
    ON events.VenueID = venues.ID)
INNER JOIN performers
ON events PerformerID = performers.ID
WHERE Venue = 'Paramount Theater'