

# 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

2

## Example Database

**Nations**

Name	text	Common rather than official name
Domain	text	Internet top-level domain name
Capital	text	Nation's capital
Latitude	number	Approx. latitude of capital
N_S	Boolean	Latitude is N(orth) or S(outh)
Longitude	number	Approx. longitude of capital
E_W	Boolean	Longitude is E(ast) or W(est)
Interest	text	A short description of the country

Primary Key: Name

Name	Dom	Capital	Lat	NS	Lon	EW	Interest
Ireland	IE	Dublin	52	N	7	W	History
Israel	IR	Jerusalem	32	N	35	E	History
Italy	IT	Rome	42	N	12	E	Art
Jamaica	JM	Kingston	18	N	77	W	Beach
Japan	JP	Tokyo	35	N	143	E	Kabuki

3

## 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

4

## Select

- Syntax:

```
SELECT *
FROM <table>
WHERE <test>
```
- The asterisk (\*) means "anything"
- Example:

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

5

## Select

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

Name	Dom	Capital	Lat	NS	Lon	EW	Interest
Australia	AU	Canberra	37	S	148	E	Beach
Bahamas	BS	Nassau	25	N	78	W	Beach
Barbados	BB	Bridgetown	13	N	59	W	Beach
Belize	BZ	Belmopan	17	N	89	W	Beach
Bermuda	BM	Hamilton	32	N	64	W	Beach

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

6

## 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.

7

## Project

- Syntax:

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

- Example:

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

8

## Project

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

Name	Dom	Interest
Nauru	NR	Beach
Nepal	NP	Mountains
Netherlands	NL	Canals
New Caledonia	NC	Beach
New Zealand	NZ	Adventure

Figure 16.8 Sample entries for a Project operation on Nations.

9

## 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'
```

10

## Select And Project Results

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

Name	Dom	Lat
Finland	FI	61
Greenland	GL	72
Iceland	IS	65
Norway	NO	60

Figure 16.9 Northern, the table of countries with northern capitals.

11

## Exercise

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

- Solution:

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

12

## Union

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

```
<table1>
UNION
<table2>
```

13

## Union Results

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

Name	Dom	Capital	Lat	NS	Lon	EW	Interest
Falkland Is	FK	Stanley	51	S	58	W	Nature
Finland	FI	Helsinki	61	N	26	E	Nature
Greenland	GL	Nuuk	72	N	40	W	Nature
Iceland	IS	Reykjavik	65	N	18	W	Geysers
Norway	NO	Oslo	60	N	10	E	Vikings

14

## 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>
```

15

## Another Table

Travelers	Friend	Homeland
Friend Text A Traveling Companion	Isabela	Argentina
Homeland Text Friend's Home Country	Brian	South Africa
Primary Key: Friend	Wen	China
(a)	Clare	Canada
	(b)	

Figure 16.11 (a) The definition of the Travelers table, and (b) its values.

16

## Product Results

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

Name	Dom	Capital	Lat	NS	Log	EW	Interest	Friend	Homeland
Cyprus	CY	Nicosia	35	N	32	E	History	Clare	Canada
Czech Rep.	CZ	Prague	51	N	15	E	Pilsner	Isabella	Argentina
Czech Rep.	CZ	Prague	51	N	15	E	Pilsner	Brian	South Africa
Czech Rep.	CZ	Prague	51	N	15	E	Pilsner	Wen	China
Czech Rep.	CZ	Prague	51	N	15	E	Pilsner	Clare	Canada
Denmark	DK	Copenhagen	55	N	12	E	History	Isabella	Argentina

Figure 16.12 Some rows from the supertable that is the product of Nations and Travelers. For each row in Nations and each row in Travelers, there is a row in the product table that combines them.

17

## Join

- Combines two tables, like the Product Operation, but doesn't necessarily produce all pairings
- Join operation:
  - $Table1 \bowtie Table2$  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

18

## Join

- General syntax:

```
SELECT *
FROM <table1> INNER JOIN <table2>
ON <table1>.<field> = <table2>.<field>
```

- Can be written with product operation:

```
SELECT *
FROM <table1>, <table2>
WHERE <table1>.<field> = <table2>.<field>
```

19

## Join Applied

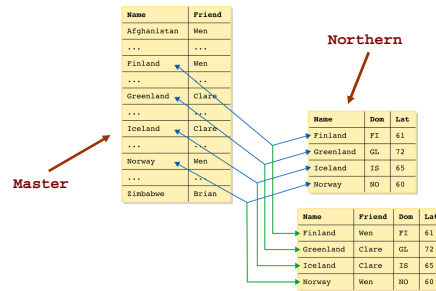


Figure 16.14 The Join operation: Master ⋈ Northern.

20

## 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

21

## Exercise

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

ID	Performer	Genre	Description
4	Aventura	R&B	According to Wikipedia: "Ave"
5	Return to Forever	Jazz	According to Wikipedia: "Ret"
6	Ladino	Electronic	According to Wikipedia: "Lad"

ID	PerformanceDate	Performance	VenueID
1	May 26, 2008	6	2
2	May 28, 2008	4	2
3	June 1, 2008	5	2

ID	Venue	Address
2	Paramount Theater	1511 Pine Street, Seattle, WA 98101
3	Shelburne Theater	1426 1st Avenue, Seattle, WA 98101

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

22

## Solution

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

23

## Exercise

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

ID	Performer	Genre	Description
4	Aventura	R&B	According to Wikipedia: "Ave"
5	Return to Forever	Jazz	According to Wikipedia: "Ret"
6	Ladino	Electronic	According to Wikipedia: "Lad"

ID	PerformanceDate	Performance	VenueID
1	May 26, 2008	6	2
2	May 28, 2008	4	2
3	June 1, 2008	5	2

ID	Venue	Address
2	Paramount Theater	1511 Pine Street, Seattle, WA 98101
3	Shelburne Theater	1426 1st Avenue, Seattle, WA 98101

24

## 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'
```