Intro. to SQL

DBC340

Mike Pangburn

Learning Objectives

- ✓ Understand the data-representation terminology underlying relational databases
- ✓ Understand core SQL concepts that stem from relational algebra
  - ✓ cover all the fundamental operators other than divide
- ✓ Gain familiarity with SQL syntax

Relational Databases and Tables

- A Relational Database is a database management system (DBMS) that holds a set of tables, like Excel worksheets, each filled with data
- A table = a set of rows or "records"
  - Like a list…
  - …but rows have no assumed order
    - Even columns have no assumed order

A sample database table

Product (Table name)

Attribute names

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Tuples or rows

DB Terminology overview

Term	Definition
Relation (also, *Entity*)	A table.
Tuple (also, *record* or "*entity instance*")	A row.
Attribute (also, *field*)	A column.
Attribute value	The value in a particular table cell (i.e., a row/column intersection).
Primary key	Specified column(s) providing a unique value for every row.
Table schema	The set of attributes (not their values) defining the structure of a table.
Database schema	The full set of named tables and their schemas, in the full database (which could contain many tables).

SQL – what is it?

- All SQL query take some rectangular input table(s) and, using that data as you nicely ask it to, generates your desirable output table
  - You must ask nicely, i.e., properly
- A simple language
  - Not a true programming language
  - Much easier to learn
    - Still very hard to master
- Why was SQL developed?
  - As a user-friendly means to define and manipulating data in relational databases
  - Incredibly valuable for addressing managerial ad hoc query questions

SQL Query

Basic form:

```
SELECT  attributes
FROM    table          (or multiple tables, often "joined")
WHERE   conditions     ("row restrictions")
```

1-table query with row filtering

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  *
FROM    Product
WHERE   category='Gadgets'
```

"row restriction"

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

A query projecting specific columns

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  PName, Price, Manufacturer
FROM    Product
WHERE   Price > 100
```

"row restriction" with certain columns "projected"

PName	Price	Manufacturer
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

Selections

What goes in the WHERE clause:

- x = y, x < y, x <= y, etc
  - For numbers, they have the usual meanings
  - For characters: lexicographic ordering
  - For dates and times, earlier is less than later
- Pattern matching on strings: s LIKE p (next slide)
- You can combine selections using AND / OR type logic
  - E.g., WHERE x < y AND (x < z OR y > z)

The **LIKE** operator

- s **LIKE** p: pattern matching on strings
- p may contain two special "wildcard" symbols:
  - % = any sequence of characters
  - _ = any single character
  - Note: these two special characters are system dependent… "Google" the LIKE wildcard symbols for any system you use

Product(Name, Price, Category, Manufacturer)
Find all products whose name mentions 'gizmo':

```
SELECT  *
FROM    Products
WHERE   PName LIKE '%gizmo%'
```

Hoe to eliminate identical output rows?

```
SELECT  category
FROM    Product
```

Category
Gadgets
Gadgets
Photography
Household

Compare to:

```
SELECT  DISTINCT category
FROM    Product
```

Category
Gadgets
Photography
Household

Ordering the Results

```
SELECT   pname, price, manufacturer
FROM     Product
WHERE    category='Gadgets' AND price > 50
ORDER BY price, pname
```

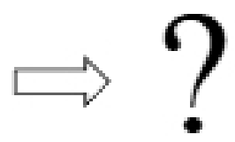
Ordering is ascending, unless you specify ORDER BY DESC
(DESC means you want descending order)

Ties are broken by the second attribute on the ORDER BY list, etc.

Ordering the Results

```
SELECT   Category
FROM     Product
ORDER BY PName
```

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi



"Joining tables" in SQL

- Connect two or more tables:

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

What is the connection? Do you think the DBMS sees that?

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Joins using SQL

Product (pname, price, category, manufacturer)
Company (cname, stockPrice, country)

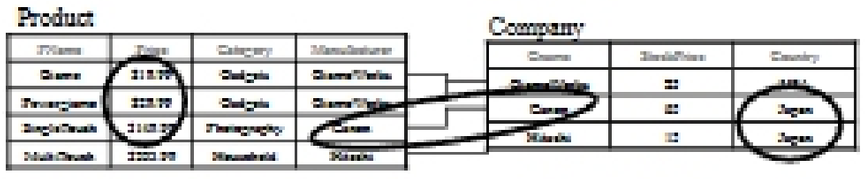
Find all products under $200 manufactured in Japan; return their names and prices.

Join between Product and Company

```
SELECT   PName, Price
FROM     Product, Company
WHERE    Manufacturer=CName AND Country='Japan'
         AND Price <= 200
```

Joins using SQL

Product



Company


```
SELECT   PName, Price
FROM     Product, Company
WHERE    Manufacturer=CName AND Country='Japan'
         AND Price <= 200
```

PName	Price
SingleTouch	$149.99

Joins and unexpected "duplicated" results

Example: Find all countries that manufacture some product in the 'Gadgets' category

Product (pname, price, category, manufacturer)
Company (cname, stockPrice, country)

```
SELECT   Country
FROM     Product, Company
WHERE    Manufacturer=CName AND
         Category='Gadgets'
```