CS601: Relational Databases. Introduction to SQL.

Olga Karpenko

This presentation is based on the materials of Prof. Engle

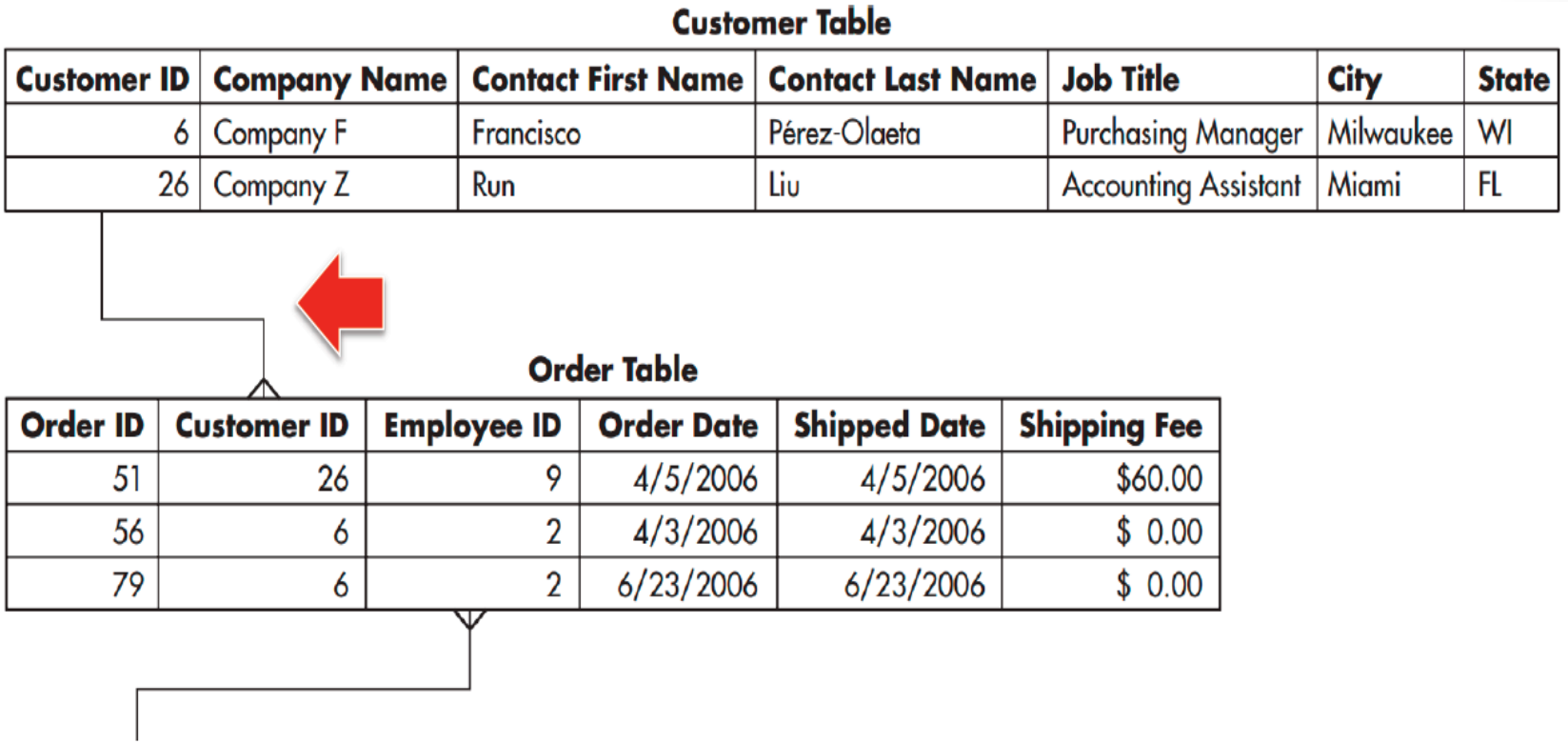
Announcements

- Lab 6 due date changed to Sunday night
- Check your MySql user account:

Database

- Container for storing structured data
- Persistent Storage
  - Information is stored on disk(s)
  - Slower access to information than when it's in memory
  - Remains even when system is powered off

Relational Tables



Customer Table

Customer ID	Company Name	Contact First Name	Contact Last Name	Job Title	City	State
6	Company F	Francisco	Pérez-Olaeta	Purchasing Manager	Milwaukee	WI
26	Company Z	Run	Liu	Accounting Assistant	Miami	FL

Order Table

Order ID	Customer ID	Employee ID	Order Date	Shipped Date	Shipping Fee
51	26	9	4/5/2006	4/5/2006	$60.00
56	6	2	4/3/2006	4/3/2006	$ 0.00
79	6	2	6/23/2006	6/23/2006	$ 0.00

Primary Key

- A column (or column(s)) whose values uniquely identify a row
- Used to refer to a specific row
- Ex: customer ID, student ID
- Rules:
  - No two rows can have the same primary key
  - Every row must have a primary key value

DataBase Management System

- Handles data storage, maintenance, and retrieval
- Manages concurrent data access
- Transactions are atomic (all-or-nothing)
- Often separate physical, logical layers
  - How data is stored may not be how it is viewed

Relational DBMS

- Supports relationships between tables
- Able to join data across tables together using queries
- Stable and mature, with many providers
  - *MySQL, PostreSQL, SQL Server, Oracle...*

http://docs.oracle.com/javase/tutorial/jdbc/overview/database.htm

How to Access MySQL

- Login to a CS lab computer (say, g1212)

  Example:

  ssh yourUsfUsername@stargate.cs.usfca.edu

  ssh hrn23526

- Login to mySQL server

  mysql –h sql.cs.usfca.edu –u user## –p

  where user## is mysql name assigned to you

CS MySQL Accounts

- Change password

```
SET PASSWORD = PASSWORD('newpass');
```

- Choose database to use

```
USE user##;

SHOW TABLES;
```

- Start creating tables and queries

SQL
Structured Query Language

SQL

- Structured Query Language (SQL)
  - Often pronounced "sequel" or S-Q-L
  - Standard language for communicating with relational databases
- Many different SQL servers
  - MySQL, Oracle, Postgres, SQL Server, etc.

SQL Statements

- Keywords
  - Traditionally UPPERCASE (not required)
  - Includes operators (e.g. =, >, etc.)
- Identifiers
  - Traditionally lowercase
- Constants

SQL Statements

```
SELECT name
FROM teams
WHERE id = 9;
```

SQL Statements: Keywords

```
SELECT name
FROM teams
WHERE id = 9;
```

SQL Statements: Identifiers

```
SELECT name
FROM teams
WHERE id = 9;
```

Data Definition Language (DDL)

- Create/manipulate structure
  - Tables, Databases, Schemas, etc.
- Major statements:

  CREATE: Adds new database object

  ALTER: Manipulates existing database object

  DROP: Removes database object

DDL

- Format Example

  CREATE TABLE tabname (colname TYPE CONSTRAINTS, …);

- Example:

  CREATE TABLE students (

  id INTEGER NOT NULL PRIMARY KEY,

  name VARCHAR(50) NOT NULL,

  program VARCHAR(50));

Checking the Layout

- DESCRIBE tableName;

Column Data Type

- Numeric Data Types
  - INTEGER, FLOAT, etc.
- Character Data Types
  - CHAR, VARCHAR, etc.
- Temporal Data Types
  - DATE, TIME, etc.

Numeric Data Types

- TINYINT
- SMALLINT -32,768 to 32,767
- MEDIUMINT
- INTEGER -2,147,483,648 to 2,147,483,647
- BIGINT

Numeric Data Types

- REAL, FLOAT,..
  - Differences implementation dependent
  - Compatible with scientific notation

Character Data Types

- CHAR(width)
  - Fixed width
  - Anything < width has trailing spaces
- VARCHAR(maxwidth)
  - Width may vary up to maximum
  - More overhead (small) than CHAR()
- NCHAR(), NVCHAR()
  - 2 bytes per character
  - Supports larger character sets like UTF-8

Data Types

- CLOB: Character Large Object
  - For storing large text values

    e.g. article source text
  - Can't be indexed or sorted
- BLOB: Binary Large Object
  - For storing large binary values

    e.g. images, files

Temporal Data Types

- DATE, TIME, TIMESTAMP (Date & Time)
  - Input Format
    - Depends on database system
    - All support YYYY-MM-DD date input format
  - Storage Format
  - Display Format
    - Format returned in a SELECT query

      e.g. MM/DD/YYYY

Other Column Keywords

- PRIMARY KEY
  - Indicates column values are unique
  - Allows each row to be uniquely identified
- AUTO_INCREMENT
  - Defined in mySQL
  - Automatically increments value for each row

Inserting data into the table

- Example Statement

```
INSERT INTO students ( name, program )
VALUES ( ’Carolyn', 'Computer Science' ),
            (‘Xin’, ‘Economics’);
```

Data Manipulation Language

- Create/manipulate data
  - Operates on rows
- Major statements:
  - INSERT: Creates new row(s) in table
  - SELECT: Retrieve data from table
  - UPDATE: Update value in col in row(s) in table
  - DELETE: Removes rows from table

Data Manipulation Language

- Format Example

  INSERT INTO tabname (column order)

  VALUES (column values);

- Example Statement

  INSERT INTO students (name, program)

  VALUES ('Yasmin', 'Computer Science');

SQL SELECT Statements

- Retrieves information from database
- Common clauses
  - SELECT (always first)
  - FROM
  - WHERE
  - HAVING
  - ORDER BY (always last)

SELECT Statement: Example

```
SELECT *
FROM students
WHERE id > 3;
```

Sorting results

```
SELECT *
FROM students
ORDER BY name DESC;
```

Aggregate Functions

- AVG
- SUM
- COUNT
- MAX/MIN
- …

Aggregate Functions

```
SELECT AVG(GPA) AS avgGPA
FROM students;

SELECT COUNT(*) AS numStudents
FROM students;
```

Concatenating fields

- Joining values together (by appending) to form a single long value

```
SELECT  CONCAT (name, ': ', instructor) AS courseInfo
FROM courses
ORDER BY name;
```

Grouping

- Divide data into logical sets so you can perform aggregate calculations on each group
- Ex: Number of students registered for each course:

```
SELECT courseId, COUNT(*) AS numStudents
FROM enrollment
GROUP BY courseId
```

Filtering groups

- Ex: List of courseIds for which there are 3 or more registered students
- Use WHERE?
  - No, **WHERE filters rows, not groups**

Filtering groups: HAVING

```
SELECT courseId, COUNT(*) AS numStudents
FROM enrollment
GROUP BY courseId
HAVING  (COUNT(*))>=3;
```

Example: Product Catalog

- Customers, vendors, products, orders, orderitems, productnotes
- Multiple products by the same vendor
  - Where would you store vendor info (name, address, phone etc..)?

Example: Product Catalog

- Customers, vendors, products, orders, orderitems, productnotes
- Multiple products by the same vendor
  - Where would you store vendor info (name, address, phone etc..)?
  - Not with products!
    - Would be repeating info
    - If it changes -> need to update everywhere

References

- Ben Forta, MySQL: Crash course