

## Chapter 18: Databases

CS 80: Internet Programming

Instructor: Mark Edmonds

### What is a database?

- A more-structured, multi-sheet excel spreadsheet
- We call an excel sheet a **table**, which has rows and columns of data

### What is a database?

- Each table contains a column with **primary keys**. Each row has a unique value for the primary key's column
  - This guarantees that each row has at least one unique value
  - Means we can identify and access a row uniquely
  - Examples of primary keys: ID numbers, social security numbers, etc

### What is a database?

- We access data in a database through a **query**, which is just a request of information from a table
- In this class, we will use SQL queries

**Employee Table**

|       | Number      | Name     | Department | Salary | Location    |
|-------|-------------|----------|------------|--------|-------------|
| Row { | 23603       | Jones    | 413        | 1100   | New Jersey  |
|       | 24568       | Kerwin   | 413        | 2000   | New Jersey  |
|       | 34589       | Larson   | 642        | 1800   | Los Angeles |
|       | 35761       | Myers    | 611        | 1400   | Orlando     |
|       | 47132       | Neumann  | 413        | 9000   | New Jersey  |
|       | 78321       | Stephens | 611        | 8500   | Orlando     |
|       | Primary key |          | Column     |        |             |

**Figure 1:** Employee Table**Employee Table**

- Rows are not guaranteed to be stored in any particular order
- Any column value *except* the primary key may exist in multiple rows
- Each column represents a different data attribute
- Each row is typically unique
- Typically we won't be interested in all the rows, and typically we won't be interested in all the columns of the rows we are interested in
- We can have multiple tables in a database
  - When a primary key is used as a column in a different table, it is called a **foreign key**. This is important for linking tables together; without this mechanism we wouldn't have a way to guarantee unique links between tables

**Employee Table**

- An example query result: show where each department is located in increasing order by department number

| Department | Location    |
|------------|-------------|
| 413        | New Jersey  |
| 611        | Orlando     |
| 642        | Los Angeles |

**Figure 2:** Query result

## The Rule of Referential Integrity

- A guarantee that every foreign key is used as a primary key in a different table
- We can determine if a particular row in a table is valid; e.g. it's foreign key exists in a different table as a primary key
- This also enables **joining**, where we bring data from multiple tables together; the foreign key acts as the link between the tables

## A Books database

- Consists of three tables: *Authors*, *AuthorISBN*, and *Titles*.

### Authors Table

- Contains information about authors
- AuthorID is the primary key

### Authors Table

| Column    | Description   |
|-----------|---|
| AuthorID  | Author's ID number in the database. In the books database, this integer column is defined as <b>autoincremented</b> —for each row inserted in this table, the AuthorID value is increased by 1 automatically to ensure that each row has a unique AuthorID. This column represents the table's primary key. |
| FirstName | Author's first name (a string).   |
| LastName  | Author's last name (a string).  |

**Figure 3:** Authors table

**Authors Table**

| AuthorID | FirstName | LastName |
|----------|-----------|----------|
| 1        | Paul      | Deitel   |
| 2        | Harvey    | Deitel   |
| 3        | Abbey     | Deitel   |
| 4        | Michael   | Morgano  |
| 5        | Eric      | Kern     |

**Figure 4:** Authors Table**AuthorISBN Table**

- Maintains AuthorID's and ISBNs
- ISBN is a foreign key - it is the primary key in the Title table
- AuthorID is a foreign key - it is the primary key in the Authors table
- Primary key is the combination of the two foreign keys - guaranteed to be unique

**AuthorsISBN Table**

| Column   | Description   |
|----------|---|
| AuthorID | The author's ID number, a foreign key to the Authors table. |
| ISBN     | The ISBN for a book, a foreign key to the Titles table.     |

**Figure 5:** AuthorsISBN Table Description

**AuthorsISBN Table**

| AuthorID | ISBN       | AuthorID | ISBN       |
|----------|------------|----------|------------|
| 1        | 0132152134 | 2        | 0132575663 |
| 2        | 0132152134 | 1        | 0132662361 |
| 1        | 0132151421 | 2        | 0132662361 |
| 2        | 0132151421 | 1        | 0132404168 |
| 1        | 0132575663 | 2        | 0132404168 |
| 1        | 013705842X | 1        | 0132121360 |
| 2        | 013705842X | 2        | 0132121360 |
| 3        | 013705842X | 3        | 0132121360 |
| 4        | 013705842X | 4        | 0132121360 |
| 5        | 013705842X |          |            |

**Figure 6:** AuthorsISBN Table**Titles Table**

- Maintains information about ISBNs and Titles
- ISBN is the primary key
- Title is a string

**Titles Table**

| Column | Description  |
|--------|--|
| ISBN   | ISBN of the book (a string). The table's primary key. ISBN is an abbreviation for "International Standard Book Number"—a numbering scheme that publishers use to give every book a unique identification number. |
| Title  | Title of the book (a string).  |

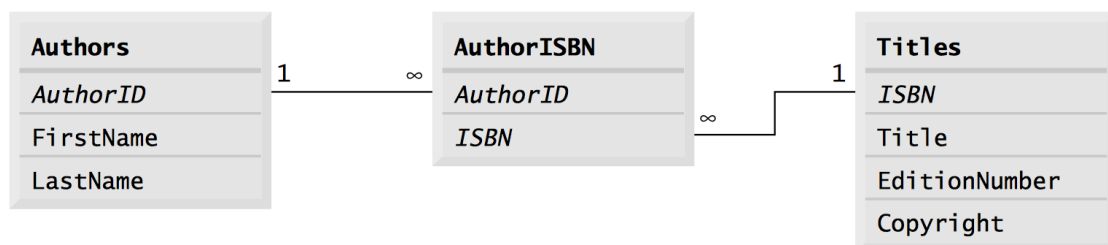
**Figure 7:** Titles Table Description 1

**Titles Table**

| Column        | Description                              |
|---------------|--|
| EditionNumber | Edition number of the book (an integer). |
| Copyright     | Copyright year of the book (a string).   |

**Figure 8:** Titles Table Description 2**Titles Table**

| ISBN       | Title   | EditionNumber | Copyright |
|------------|---|---------------|-----------|
| 0132152134 | Visual Basic 2010 How to Program                | 5             | 2011      |
| 0132151421 | Visual C# 2010 How to Program                   | 4             | 2011      |
| 0132575663 | Java How to Program                             | 9             | 2012      |
| 0132662361 | C++ How to Program                              | 8             | 2012      |
| 0132404168 | C How to Program                                | 6             | 2010      |
| 013705842X | iPhone for Programmers: An App-Driven Approach  | 1             | 2010      |
| 0132121360 | Android for Programmers: An App-Driven Approach | 1             | 2012      |

**Figure 9:** Titles Table**Table Relationships****Figure 10:** Table Relationships

- Italics indicate a primary key
- Lines between tables indicate how tables are connected
- Numbers indicate the degree of the connection
  - E.g AuthorID is unique in the Authors table, but may be used multiple times in the AuthorISBN table
- AuthorISBN provides a *many-to-many relationship* between Authors and Titles
  - An author can write many books and a book can have many authors

## SQL

- SQL is a means to query, or access, information in a database
- SQL is a bit of a language in itself, people make careers out of being SQL wizards

## SQL Keywords

| SQL keyword | Description   |
|-------------|---|
| SELECT      | Retrieves data from one or more tables.   |
| FROM        | Tables involved in the query. Required in every SELECT.   |
| WHERE       | Criteria for selection that determine the rows to be retrieved, deleted or updated. Optional in a SQL query or a SQL statement. |
| GROUP BY    | Criteria for grouping rows. Optional in a SELECT query.   |
| ORDER BY    | Criteria for ordering rows. Optional in a SELECT query.   |
| INNER JOIN  | Merge rows from multiple tables.  |
| INSERT      | Insert rows into a specified table.   |
| UPDATE      | Update rows in a specified table.   |
| DELETE      | Delete rows from a specified table.   |

**Figure 11:** SQL Keywords

## SQL Queries

- Query Structure:
  - Specify the columns we want with **SELECT**
  - Specify the tables to get the data from with **FROM**

- Specify additional criteria with **WHERE**

## SQL Queries

- Example for the Books database:

```
1  /* query AuthorID and LastName columns from Authors table */
2  SELECT AuthorID, LastName
3      FROM Authors
```

- Gets the AuthorID and LastName from the Authors table

| AuthorID | LastName |
|----------|----------|
| 1        | Deitel   |
| 2        | Deitel   |
| 3        | Deitel   |
| 4        | Morgano  |
| 5        | Kern     |

**Figure 12:** SQL Author ID and Last name query

## SQL Queries

- A more complicated example:

```
1  /* query with constraint that copyright is greater than 2010 */
2  SELECT Title, EditionNumber, Copyright
3      FROM Titles
4      WHERE Copyright > '2010'
```



| Title   | EditionNumber | Copyright |
|---|---------------|-----------|
| Visual Basic 2010 How to Program                | 5             | 2011      |
| Visual C# 2010 How to Program                   | 4             | 2011      |
| Java How to Program                             | 9             | 2012      |
| C++ How to Program                              | 8             | 2012      |
| Android for Programmers: An App-Driven Approach | 1             | 2012      |

**Figure 13:** SQL title, edition, copyright query

## SQL Queries

- Note: SQL strings are always single quoted ('), not double quoted (")
  - Note: A wildcard (\*) may be used to select all columns, e.g. `SELECT * FROM Titles` selects all columns from the Titles table

## Pattern Matching

- The `WHERE Copyright > '2010'` line above uses an operator >
  - We also have the <, >, <=, >=, =, != and `LIKE`
- `LIKE` indicates a pattern matching qualification on a query
- The percent sign (%) indicates a search for zero or more characters at the percent character's position (a wildcard)
- The underscore (\_) indicates a single wildcard at the position of the underscore

## Pattern Matching Examples

- Select all authors whose last name begins with a 'D'

```
1 /* query with last names that start with 'D' */
2 SELECT AuthorID, FirstName, LastName
3     FROM Authors
4     WHERE LastName LIKE 'D%'
```

- Select all authors whose last name starts with any character, followed by an 'o', followed by any number of additional characters

```
1  /* query with last names that have a second character of 'o' */
2  SELECT AuthorID, FirstName, LastName
3      FROM Authors
4      WHERE LastName LIKE '_o%'
```

## Ordering

- We can add a qualifier to sort the query's result through **ORDER BY**
- Can sort by ascending (ASC), descending (DESC)
  - Default sorting order is by ascending (so if you don't specify, ascending order will be used)

## Ordering

- Can order by multiple columns - sorts by outer sort first, then each additional sorting specified (e.g. last name then by first name)

```
1  /* query and order by ascending order */
2  SELECT columnName1, columnName2, ...
3      FROM tableName
4      ORDER BY column ASC
5  /* query and order by descending order */
6  SELECT columnName1, columnName2, ...
7      FROM tableName
8      ORDER BY column DESC
```

## Exercise

- Write a query to get the AuthorID, FirstName, LastName, and order the results by descending last name

## Exercise

- Solution:

```
1  /* query and order by descending order */
2  SELECT AuthorID, FirstName, LastName
3      FROM Authors
4      ORDER BY LastName DESC
```

| AuthorID | FirstName | LastName |
|----------|-----------|----------|
| 4        | Michael   | Morgano  |
| 5        | Eric      | Kern     |
| 1        | Paul      | Deitel   |
| 2        | Harvey    | Deitel   |
| 3        | Abbey     | Deitel   |

**Figure 14:** Solution

### Exercise

- Write a query to get the ISBN, Title, EditionNumber, and Copyright from the Titles table that end with the phrase 'How to Program' and order them in ascending order by title

### Exercise

- Solution:

```
1  /* query that ends with 'How to Program' and sort by ascending
   order */
2  SELECT ISBN, Title, EditionNumber, Copyright
3      FROM Titles
4      WHERE Title LIKE '%How to Program'
5      ORDER BY Title ASC
```

| ISBN       | Title                            | Edition-Number | Copyright |
|------------|----------------------------------|----------------|-----------|
| 0132404168 | C How to Program                 | 6              | 2010      |
| 0132662361 | C++ How to Program               | 8              | 2012      |
| 0132575663 | Java How to Program              | 9              | 2012      |
| 0132152134 | Visual Basic 2005 How to Program | 5              | 2011      |
| 0132151421 | Visual C# 2005 How to Program    | 4              | 2011      |

**Figure 15:** Solution

### Merging Data from Multiple Tables

- Achieved through a **INNER JOIN** which merges rows from two tables by matching values in columns that are common to both tables
  - This is a common use of primary and foreign keys!

### Merging Data from Multiple Tables

```
1 /* query that merges data from multiple tables */
2 SELECT columnName1, columnName2, ...
3 FROM table1
4 INNER JOIN table2
5 ON table1.columnName = table2.columnName
```

### Exercise

- Write a query that results in the first name, last name, and ISBN for every author and order them by last name, then first name, in ascending order

### Exercise

- Solution:

```
1  /* query that merges data from multiple tables */
2  SELECT FirstName, LastName, ISBN
3      FROM Authors
4      INNER JOIN AuthorISBN
5      ON Authors.AuthorID = AuthorISBN.AuthorID
6      ORDER BY LastName, FirstName
```

- Note: the use of `Authors.AuthorID` and `AuthorISBN.AuthorID` is called a **qualified name**. When tables have matching column names, we must be more specific by stating which table we want to use with each column (think about this: why we didn't have to qualify the `ISBN` in the `SELECT` part of the query?)

## Exercise

| FirstName | LastName | ISBN       | FirstName | LastName | ISBN       |
|-----------|----------|------------|-----------|----------|------------|
| Abbey     | Deitel   | 013705842X | Paul      | Deitel   | 0132151421 |
| Abbey     | Deitel   | 0132121360 | Paul      | Deitel   | 0132575663 |
| Harvey    | Deitel   | 0132152134 | Paul      | Deitel   | 0132662361 |
| Harvey    | Deitel   | 0132151421 | Paul      | Deitel   | 0132404168 |
| Harvey    | Deitel   | 0132575663 | Paul      | Deitel   | 013705842X |
| Harvey    | Deitel   | 0132662361 | Paul      | Deitel   | 0132121360 |
| Harvey    | Deitel   | 0132404168 | Eric      | Kern     | 013705842X |
| Harvey    | Deitel   | 013705842X | Michael   | Morgano  | 013705842X |
| Harvey    | Deitel   | 0132121360 | Michael   | Morgano  | 0132121360 |
| Paul      | Deitel   | 0132152134 |           |          |            |

**Figure 16:** SQL qualified name query

## SQL Insert

- Insert inserts a row into a table
- Syntax:

```
1  /* insertion syntax */
2  INSERT INTO tableName ( columnName1, columnName2, ..., columnNameN
3                          )
```

```
3      VALUES ( value1, value2, ..., valueN )
```

## Inserting, Updating, and Deleting Information from a table

- Example:

```
1  /* insertion example */
2  INSERT INTO Authors ( FirstName, LastName )
3      VALUES ( 'Sue', 'Red' )
```

| AuthorID | FirstName | LastName |
|----------|-----------|----------|
| 1        | Paul      | Deitel   |
| 2        | Harvey    | Deitel   |
| 3        | Abbey     | Deitel   |
| 4        | Michael   | Morgano  |
| 5        | Eric      | Kern     |
| 6        | Sue       | Red      |

**Figure 17:** Query Result

## SQL Update

- Update modifies existing data in a table
- Syntax:

```
1  /* update syntax */
2  UPDATE tableName
3      SET columnName1 = value1, columnName2 = value2, ...,
        columnNameN = valueN
4      WHERE criteria
```

## SQL Update

- Example:

```
1  /* update example */
2  UPDATE Authors
3      SET LastName = 'Black'
4      WHERE LastName = 'Red' AND FirstName = 'Sue'
```

| AuthorID | FirstName | LastName |
|----------|-----------|----------|
| 1        | Paul      | Deitel   |
| 2        | Harvey    | Deitel   |
| 3        | Abbey     | Deitel   |
| 4        | Michael   | Morgano  |
| 5        | Eric      | Kern     |
| 6        | Sue       | Black    |

**Figure 18:** Query Result

## SQL Update

- Question: What else could we have used for the **WHERE** portion of the previous example?

## SQL Update

- Question: What else could we have used for the **WHERE** portion of the previous example?

```
1  /* update example */
2  UPDATE Authors
3      SET LastName = 'Black'
4      WHERE AuthorID = 6
```

## SQL Delete

- Removes a row from a table
- Syntax:

```
1  /* delete syntax */
2  DELETE FROM tableName
3      WHERE criteria
```

- Example:

```
1  /* delete syntax */
2  DELETE FROM Authors
3      WHERE LastName = 'Black' AND FirstName = 'Sue'
```

## MySQL

- Open source, multiuser, multithreaded relational database that uses SQL to interact and manipulate data

## MySQL

- Reasons why MySQL is good:
  1. Scalability. You can embed it in an application or use it in massive data warehousing environments.
  2. Performance. You can optimize performance based on the purpose of the database in your application.
  3. Support for many programming languages. Later chapters demonstrate how to access a MySQL database from PHP (Chapter 19).
  4. Implementations of MySQL for Windows, Mac OS X, Linux and UNIX.
  5. Handling large databases (e.g., tens of thousands of tables with millions of rows).

## MySQL

- You can start MySQL from XAMPP (Chapter 17)