

PLEASE COMPLETE THE SOCRATIVE Pulse Survey

URL: gosocrative.com

ROOM NAME: JAVAGOLD

Please do a 'git pull upstream main' to get today's review materials

Intro to Databases and Select

Module 2: 01

Module 2 Overview

Client-Server Programming in Java

SQL

Week 1: SQL Language

Week 2: Using SQL in Java

API

Week 3: Consuming & Creating APIs in Java

Week 2: APIs continued

Module 2 API mini-capstone

Week 5 Overview

Monday

Intro to
Databases

Tuesday

Ordering,
limiting, and
grouping

Wednesday

SQL Joins

Thursday

INSERT,
UPDATE, &
DELETE

Friday

Review

Today's Objectives

- New Academic Fellow
- Capstone Recap
- Module 1 Checkpoint
- Introduction to Databases
- Tables, Rows, and Columns
- ANSI-SQL Data Types
- SQL Queries: SELECT

Intro to Databases

A **database** is an organized collection of data that can be accessed, managed, and updated.

A **relational database** is a particular type of database built upon the relational model of data

Data in a **relational database** can be accessed and reassembled in many different ways without having to reorganize the data.

- Each **entity** is stored in a table.
- Columns are called **attributes**
- **Rows** represent individual records.

Rows represent individual records and consist of many attributes organized using **columns**.

(R)DBMS

A Relational Database Management System ((R)DBMS) is a software application designed to manage a database. It has four basic functions

1. Data Definition
2. Data Storage
3. Data Retrieval
4. Administration

RDBMSs include databases like Oracle, Microsoft SQL Server, PostgreSQL, MySQL, are relational, and are commonly called **SQL Databases**.

NoSQL Databases are those that do not use a relational structure, instead they structure data specific to the problem they are designed to solve. NoSQL databases include MongoDB, Cassandra, Google BigTable, HBase, DynamoDB, and Firebase.

[DB-Engines](#) has a ranking measuring popularity of current DBMS platforms.

SQL and NoSQL Databases

Pros of Relational Databases

- Great for structured data
- Use of an existing query language (SQL)
- Great for complex queries
- Easy data navigation
- High level of data integration, due to relationships and constraints among tables
- Transactions are secure
- High reliability

Cons of Relational Databases

- Up-front schema definition
- No adaptation to changing requirements: dynamic changes to an item affect all the other items in the same table
- Data processing may be slow. High performance is possible with expensive hardware

Pros of Non-Relational Databases

- Flexible data model
- Rapid adaptation to changing requirements: dynamic changes to a item do not affect the other items
- Storage of huge amount of data with little structure
- High performance

Cons of Non-Relational Databases

- Low reliability
- Manual query language
- It is difficult to verify data integrity and consistency

Feature	Non Relational Database	Relational Database
Availability	Good	Good
Consistency	Poor	Good
Data Storage	Optimised for Huge Data	Medium - Large Data Size
Performance	High	Low
Reliability	Poor	Good
Scalability	High	High (but more expensive)

Tables, Columns, Rows

An **entity** is a set of data being stored a *table*.

A **Table** defines a set of data elements and the structure to store them. Tables are structured into *columns* and *rows*.

Columns - attributes of a table and define the name and data type. A table has a set number of defined columns.

Rows - the data being stored. A table has an unlimited number or rows.

A **Cell** is the location where a *column* and *row* intersect, and is used to refer to a specific value or row of data (*entity*).

Column

id	name	countrycode	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
3796	Houston	USA	Texas	1953631
3797	Philadelphia	USA	Pennsylvania	1517550
3798	Phoenix	USA	Arizona	1321045
3799	San Diego	USA	California	1223400
3800	Dallas	USA	Texas	1188580
3801	San Antonio	USA	Texas	1144646
3802	Detroit	USA	Michigan	951270

Column

- Tables have a set number.
- Define the data the table will hold
- Provides a label for each part of the data being stored

Columns on this Table

id, name, countrycode, district, population

Row

id	name	countrycode	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
3796	Houston	USA	Texas	1953631
3797	Philadelphia	USA	Pennsylvania	1517550
3798	Phoenix	USA	Arizona	1321045
3799	San Diego	USA	California	1223400
3800	Dallas	USA	Texas	1188580
3801	San Antonio	USA	Texas	1144646
3802	Detroit	USA	Michigan	951270

Row

- Tables have a unlimited number (0...n)
- Contain the data
- Has a value for each column

Cell

id	name	countrycode	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
3796	Houston	USA	Texas	1953631
3797	Philadelphia	USA	Pennsylvania	1517550
3798	Phoenix	USA	Arizona	1321045
3799	San Diego	USA	California	1223400
3800	Dallas	USA	Texas	1188580
3801	San Antonio	USA	Texas	1144646
3802	Detroit	USA	Michigan	951270

Cell

- The intersection of a column and row
- Used to identify a specific row of data

In this example we would identify the row we want to access by saying the ROW where the CELL in the COLUMN labelled 'name' has the value 'Chicago'

ANSI SQL

SQL - Structured Query Language : a language that lets you access and manipulate databases

ANSI-SQL - A standard that databases must follow to be considered a SQL Database.

All SQL databases support the ANSI-SQL language, however, most databases extend it with their own proprietary additions.

Character Data Types

1. **char(#)** - character. # defined the length of the data.
2. **varchar(#)** - varying character. # defined the length of the data.
3. **text** - text based data that is not limited by a predefined size

String literals in SQL use single quotes: 'Hello'

Three Strings are :

- PYTHON
- HTML
- SCHEMA.ORG

P	Y	T	H	O	N	blank	blank	blank	blank	blank	blank
H	T	M	L	blank	blank	blank	blank	blank	blank	blank	blank
S	C	H	E	M	A	.	O	R	G	blank	blank

Three Strings are :

- PYTHON
- HTML
- LINUX SERVER

P	Y	T	H	O	N						
H	T	M	L								
L	I	N	U	X	blank	S	E	R	V	E	R

Numeric Data Types

1. **int** or integer - similar to Java's int
2. **bigint** - Big Integer, similar to Java's long
3. **decimal(p, s)** - floating point numbers, similar to Java's double or BigDecimal
 - a. **p** - **precision** - the total number of digits being stored
 - b. **s** - **scale** - number of digits to the right of the decimal point

1234.567 has a *precision* of 7

1234.567 has a *scale* of 3

There are other numeric data types that specific to each RDMS. This is just a basic list of the ANSI types and not representative of every numeric data type.

Other Data Types

1. boolean - true/false

- a. Not the same in all SQL databases
 - i. MySQL - tinyint(1)
 - ii. PostgreSQL / Oracle - boolean
 - iii. MS SQL Server - bit

2. DATE

- a. yyyy-mm-dd

3. TIME

- a. hh:mm:ss

4. TIMESTAMP / DATETIME

- a. yyyy-mm-dd hh:mm:ss

Date and Time datatypes also support the usage of 24-hour vs 12-hour clocks and Time Zones.

Structured Query Language (SQL)

SQL is a declarative programming language used to manage a database and its data.

A **declarative programming language** specifies what actions should be performed rather than how to perform those actions.

SQL Consists of 3 sub-languages

1. **DDL - Data Definition Language** - defines the structure of the data
2. **DML - Data Manipulation Language** - query and modify the data
3. **DCL - Data Control Language** - used to administer the database

SELECT

The **SELECT** clause *indicates what columns* to return in the results of the query

The **FROM** clause *indicates which table(s)* to retrieve the data from.

```
SELECT name, population FROM city;
```

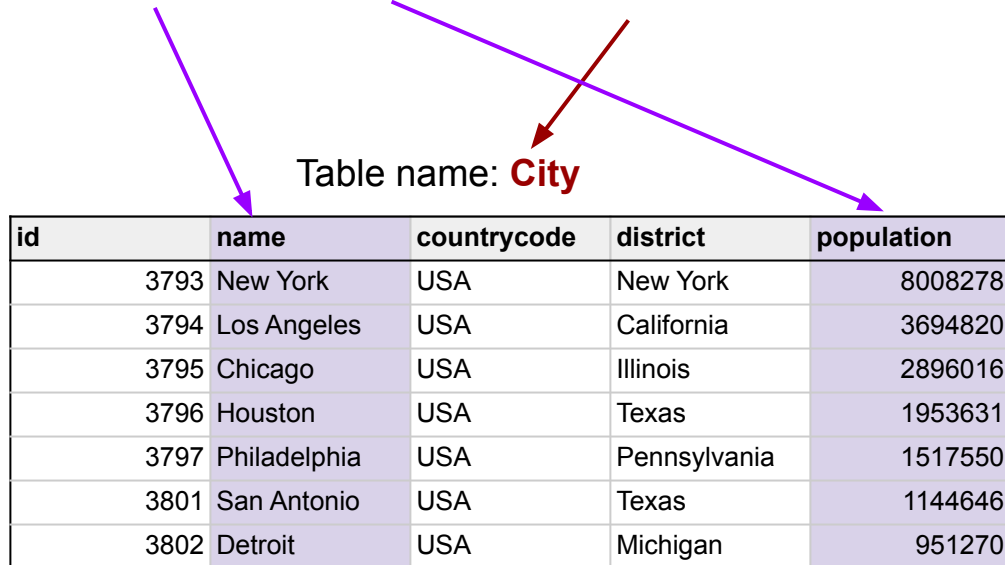


Table name: **City**

id	name	countrycode	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
3796	Houston	USA	Texas	1953631
3797	Philadelphia	USA	Pennsylvania	1517550
3801	San Antonio	USA	Texas	1144646
3802	Detroit	USA	Michigan	951270

SELECT Modifiers

***** can be used to select all columns from a table.

```
SELECT * FROM country;
```

AS can be used with a column name to give it an **Alias** (new name)

```
SELECT name AS 'CityName' FROM city;
```

Or to give a name to a combined result:

```
SELECT ( col1 + col2 ) AS 'Sum'
```

DISTINCT can be used with a column name to return only unique values from that column.

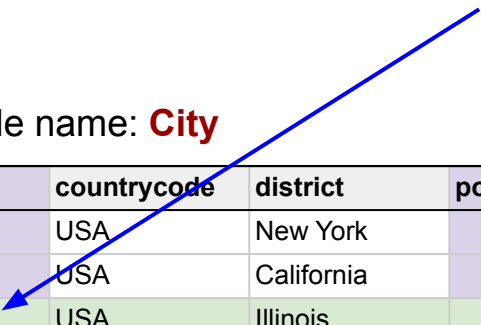
```
SELECT DISTINCT name FROM city;
```

WHERE

The **WHERE** clause is used to filter the rows returned in the results using a boolean condition. Rows that match that the expressions evaluates true for are returned by the query.

```
SELECT name, population FROM city WHERE name = 'Chicago';
```

Table name: **City**



id	name	countrycode	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
3796	Houston	USA	Texas	1953631
3797	Philadelphia	USA	Pennsylvania	1517550
3801	San Antonio	USA	Texas	1144646
3802	Detroit	USA	Michigan	951270

WHERE Clause Conditionals

=	equal to
!=, <>	Not equal to
>, <, >=, <=	Greater/Less Than
IS NULL	value is null
IS NOT NULL	value is not null
IN (val1, val2, ...)	Value is IN the list
NOT IN (val1, val2, ...)	Value is NOT IN the list
BETWEEN va1 AND val2	The value is between the 2 values Example: number BETWEEN 2 AND 10;
LIKE (%)	The value matches a pattern created with % Example: a% - the value starts with a %a - the value ends with a %a% - the value contains an a

Conditionals can be chained together using **AND** and **OR**.

Precedent can be set using **()**

Example:

WHERE num > 5 AND
(name LIKE 'A%'
OR name LIKE 'B%')

Strings in SQL use *single quotes*.

name = 'John'

Interview Preparation

- Understand what a database is
- Understand what SQL is
- Understand what a DBMS and RDBMS is
- Know the difference between SQL and NoSql Databases
- Be able to write simple queries that retrieve data