Intro to SQL


DATABASE
DATA
RELATIONAL
DBMS
SERVER
MIRRORING
PERFORMANSE
AVAILABILITY
TYPE
DESIGNER
NETWORK
RELATIONSHIP
LAYOUT
OPTIMIZATION
MIGRATION
SYSTEM
QUERY
OBJECT
ARCHITECTURE
ENTITY
STORAGE
CONTENT
BUSINESS
DEVELOPER
DESIGN
TABLE
SECURITY
RECOVERY
BACKUP
SQL
XML
SOFTWARE
KEY
RESTORE
DEVELOPMENT
ADMINISTRATION
INDEX
LANGUAGE
COLLECTION
STRUCTURE
TRANSACTION
CONCEPT
REPLICATION
INFORMATION
TECHNOLOGIES

Objectives

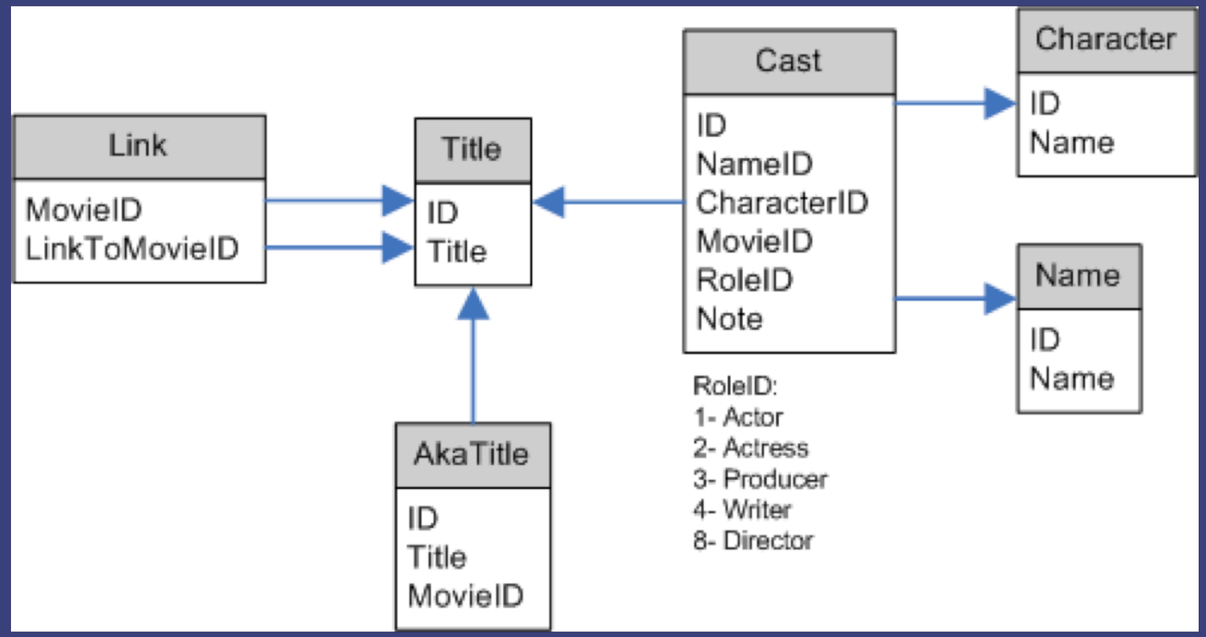
By the end of this lesson you will be able to:

- Understand what SQL is
- Explain why we use SQL
- Add, Update, Delete data in a database
- Retrieve info in a database

When and Why do we use Data?

What is a Relational Database

Name	FName	City	Age	Salary
Smith	John	3	35	$280
Doe	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250



Vocab

→ Schema

→ Entity

→ Attribute

→ Data Types

Think of your tables like a spreadsheet

Date	Store	Sales_Amount
10/27/2000	Boulder	10.46
10/28/2000	Denver	12.58

-Row = Entity or Tuples
-Column = Attribute

SQL

Open Source

-MySQL
-PostgresQL
-SQLite

Proprietary

-Microsoft Access
-Microsoft SQL Server
-Oracle

PostgreSQL

Why?

→ Fully open-sourced

→ Been in development for over 15 years

→ Fully ACID compliant (Atomicity, Consistency, Isolation, Durability)

→ Exceptional documentation

Structured Query Language

→ DDL - Data Definition Language

→ DML - Data Manipulation Language

→ DCL - Data Control Language

SQL is structured similar to the English language.

Postgresql Command Line Tools

→ \h - Get help

→ \c - Connect to a database

→ \l - List all databases you can see in Postgres

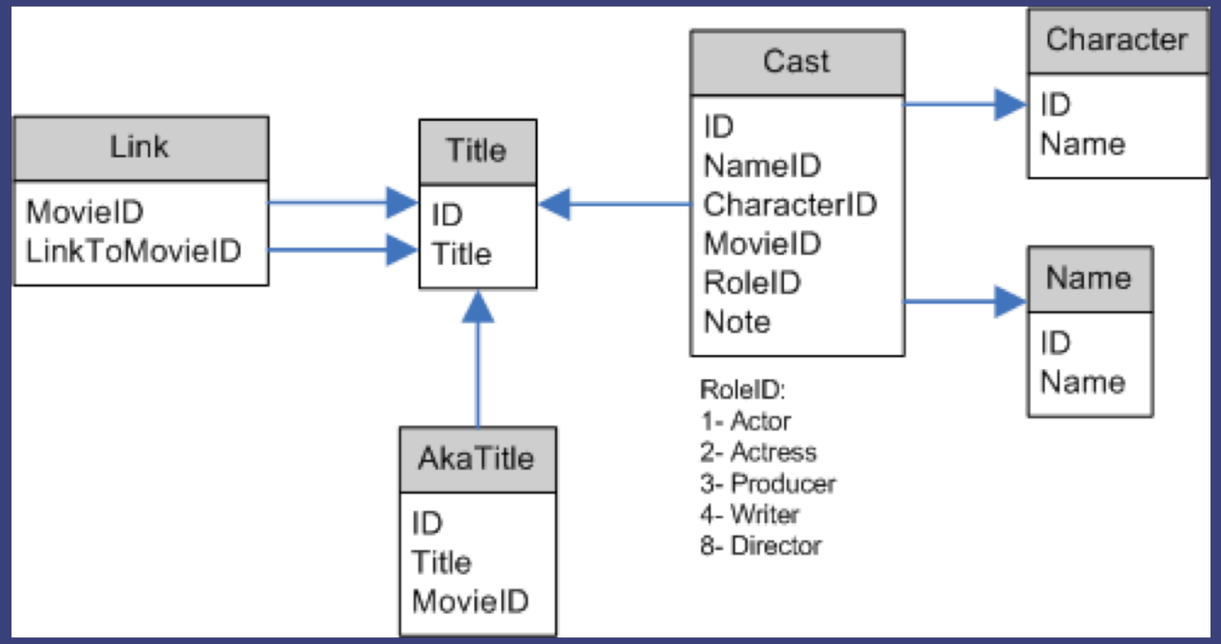
→ \dt - Describe all tables in the current database

→ \d [tablename] - Describe the columns of a table

→ \q - Quit

DDL

Data Definition Language



ID	NameID	CharacterID	MovieID	RoleID	Note
1	10068	304	1006583294	1	"Hugh Jackman playing Wolverine in X-Men"
2	16456	653	1006583294	2	"Halle Berry playing Storm in X-Men"

Data Types

→ integer

→ text

→ char

→ varchar

→ boolean

→ serial

Constraints

→ NOT NULL

→ UNIQUE

→ PRIMARY KEY

→ FOREIGN KEY

→ also many more

CREATE TABLE

```
CREATE TABLE table_name (
    column_name data_type column_constraint,
    table_constraint
);
```

Example

```
CREATE TABLE stores (
    store_id INT PRIMARY KEY,
    store_loc VARCHAR(100) NOT NULL,
    sales  INT NOT NULL
);
```

ALTER TABLE

```
ALTER TABLE table_name
    ADD new_column data_type column_constraint;

ALTER TABLE table_name
    ALTER COLUMN column_name TYPE new_data_type;

ALTER TABLE table_name
DROP column_1,
DROP column_3;
```

DROP TABLE

```
DROP TABLE [IF EXISTS] table_name;

DROP TABLE table_name1,table_name2,...;
```

DML

Data Manipulation Language

INSERT

To add Data into your table you need to INSERT that information

```
INSERT INTO table_name
VALUES(column1, column2, column3);

INSERT INTO sales
VALUES('10/27/2000, 'Boulder', 10.46);

INSERT INTO table_name (column1_name, column3_name)
VALUES(column1, column3);
```

SELECT ... FROM

→ The basic command for retrieving data from a database table is to SELECT data FROM a table. Not surprisingly, the keywords "SELECT" and "FROM" make up the core of a SQL statement.

```
SELECT “COLUMN NAME”
FROM “TABLE_NAME”;
```

SELECT ... FROM

→ You can select more than 1 row

```
SELECT “COLUMN_1”, “COLUMN_2”
FROM “TABLE_NAME”;
```

→ You can select all columns

```
SELECT *
FROM “TABLE_NAME”;
```

→ You can select unique values as well

```
SELECT DISTINCT “COLUMN_1”
```

ORDER BY

When we want to list the data in a particular order

```
SELECT *
FROM “TABLE_NAME”
WHERE “CONDITION”
ORDER BY “COLUMN_NAME” [ASC|DESC];
```

MATHEMATICAL FUNCTIONS

SQL has built-in mathematical functions to allow us to perform mathematical operations on the data. Common mathematical functions include:

→ SUM

→ AVG

→ COUNT

→ MAX

→ MIN

GROUP BY

Date	Store	Sales_Amount
10/27/2000	Boulder	10.46
10/28/2000	Denver	12.58
08/16/1998	Boulder	36.33
04/03/2015	Denver	33.87

```
SELECT MAX (Sales_Amount)
FROM Sales;
```

→ Finds the highest Sales Amount from all stores

GROUP BY

Date	Store	Sales_Amount
10/27/2000	Boulder	10.46
10/28/2000	Denver	12.58
08/16/1998	Boulder	36.33
04/03/2015	Denver	33.87

```
SELECT Store, MAX (Sales_Amount)
```

GROUP BY

```
SELECT “COLUMN_1”, FUNCTION (“COLUMN_2”)
FROM “TABLE_NAME”
WHERE “CONDITION”
GROUP BY “COLUMN_1”;
```

HAVING

-Previously we had talked about using the WHERE keyword to filter results.

-We cannot use WHERE to filter based on the result of a function, because we need to specify the filtering condition after SQL has calculated the function, and consequently any filtering condition based on the function needs to be specified after the GROUP BY phrase. So we cannot use the WHERE

HAVING

```
SELECT “COLUMN_NAME”, FUNCTION(“OTHER_COLUMN_NAME”)
FROM “TABLE_NAME”
GROUP BY “COLUMN_NAME”
HAVING (CONDITION based on FUNCTION);
```

HAVING

→ Using the SALES_HISTORY table we had earlier. If we want to sum the sales amount for each store, but only want to see results for stores with total sales amount greater than 100, we use the following SQL:

```
SELECT Store, SUM(Sales_Amount)
FROM Sales
GROUP BY Store
HAVING SUM(Sales_Amount) > 12;
```

Order of SQL Commands

→ SELECT

→ FROM

→ JOIN

→ WHERE

→ GROUP BY

→ HAVING

Update

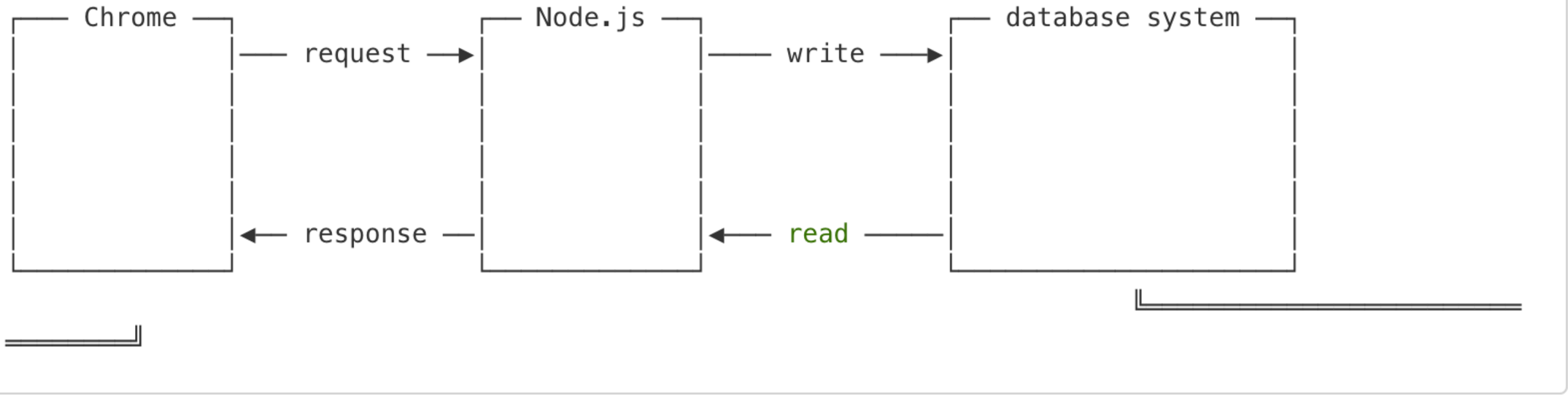
```
UPDATE table_name
SET column_name = new_value
WHERE column_name = original_value;
```

Delete

```
DELETE FROM table_name;

DELETE FROM table_name
WHERE column_name = certain_value;
```

Bringing this altogether!



Links

→ http://www.sql-tutorial.net/SQL-Cheat-Sheet.pdf

→ http://www.sqltutorial.org/

→ https://modern-sql.com/use-case/literate-sql

→ http://sqlzoo.net/

Objectives

By the end of this lesson you will be able to:

- Understand what SQL is
- Explain why we use SQL
- Add, Update, Delete data in a database
- Retrieve info in a database