CSE 106

Lecture 13 – Relational Databases and SQL

Acknowledgement:

https://en.wikipedia.org/wiki/Relational_database

https://database.guide/what-is-acid-in-databases

https://www.w3schools.com/sql/sql_intro.asp

Relational database

- A model for organizing data in a database
- Organizes data into tables with rows and columns
- A unique key identifies each row (primary key)
- SQL used by many relational databases to access and manipulate data

ACID

- Four crucial properties define relational database transactions (ACID):
  - **Atomicity** - guarantees that all the transaction succeeds or none of it does
  - **Consistency** - ensures that a transaction can only bring the database from one valid state to another (can’t be corrupted)
  - **Isolation** - No transaction will be affected by any other transaction
  - **Durability** - ensures that data changes become permanent once the transaction is committed

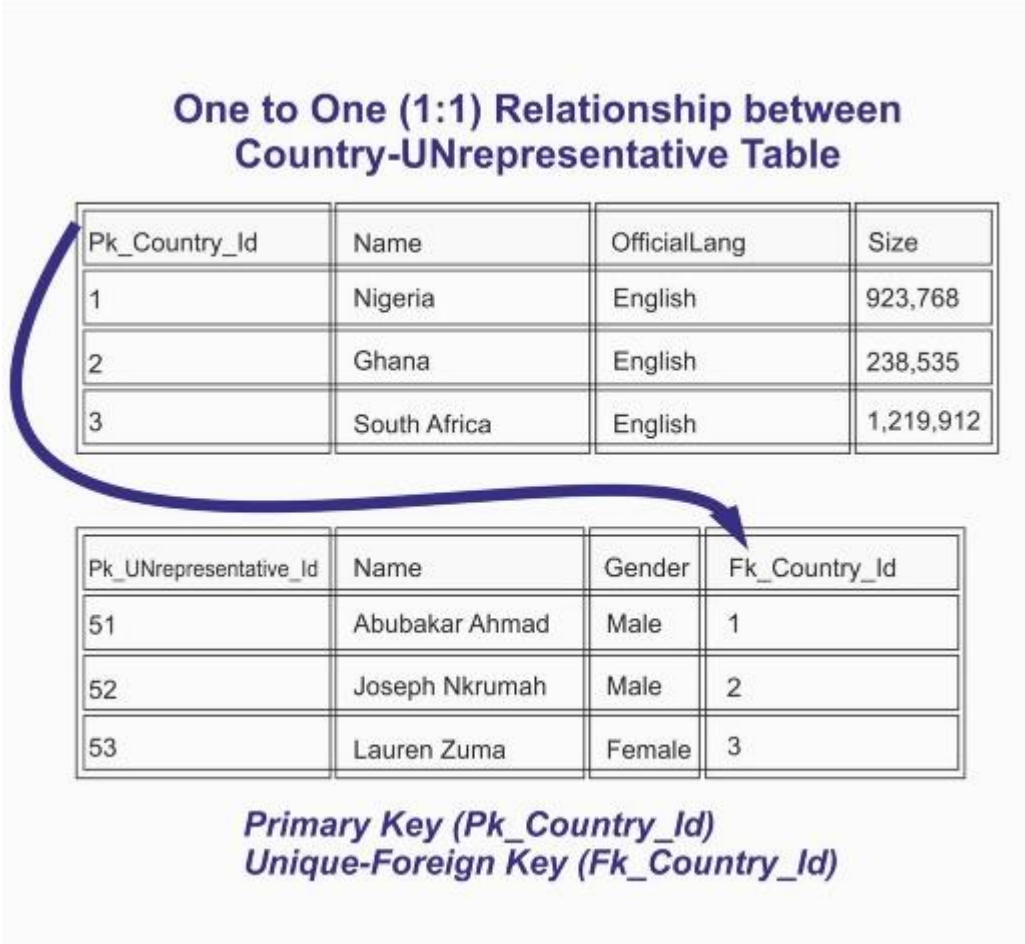
Database Tables

- A collection of related data entries consisting of columns and rows
- A record is a row of data with a unique ID or primary key
- Every column has data of the same type (also called field or attribute)

ID	Name	AGE	Salary
1	James Down	32	112,000
2	Dudley Jones	19	21,000
3	Jesus Gonzalez	61	89,000
4	Sam Smith	53	159,000
5	Ritesh Peresh	47	143,000
6	Susan Miller	26	67,000

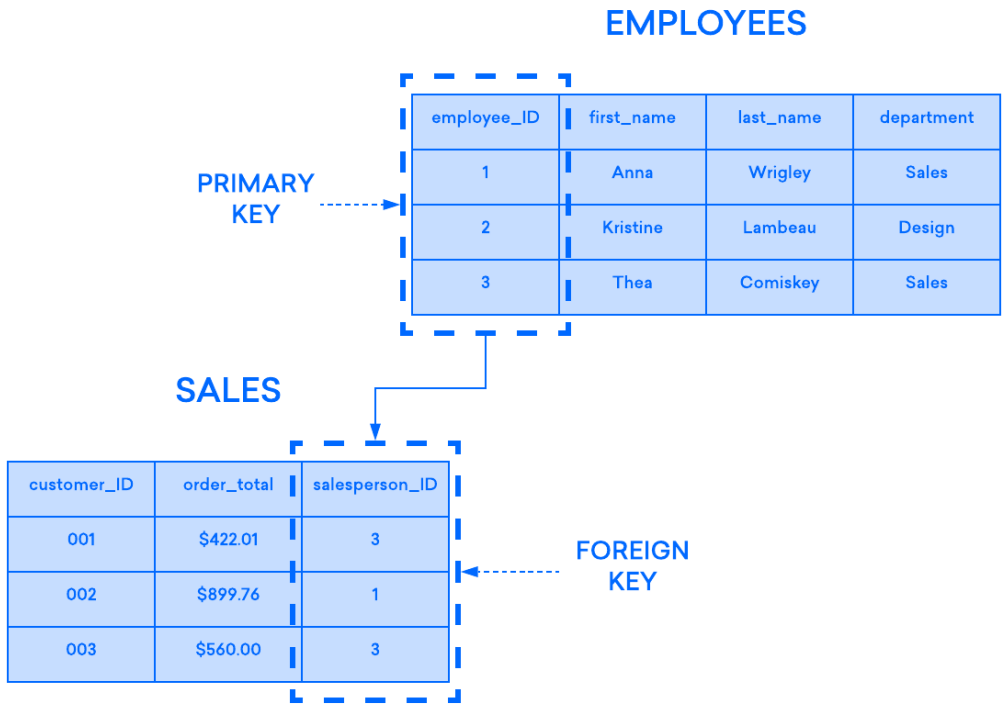
One-to-One Relationship with Foreign Key

- You can add one-to-one relationships between tables with a foreign key
- The foreign key is just the primary key of the table that is referencing it
- Example:
  - The primary key of Country table is used as the foreign key of UN Rep table
  - One Country has One UN Rep and vice versa



One-to-Many Relationship with Foreign Key

- You can add one-to-many relationships between tables with a foreign key
- The foreign key is just the primary key of the table that is referencing it
- Example:
  - The primary key of EMPLOYEES table is used as the foreign key of SALES table
  - One employee has many sales



One-to-Many Relationship with Foreign Key

- This example shows one author to many books
- Notice the table with the foreign key (Books) has the “many”

AuthorID	*AuthorName*
2	Dennis Lehane
3	Agatha Christie
4	J K Rowling

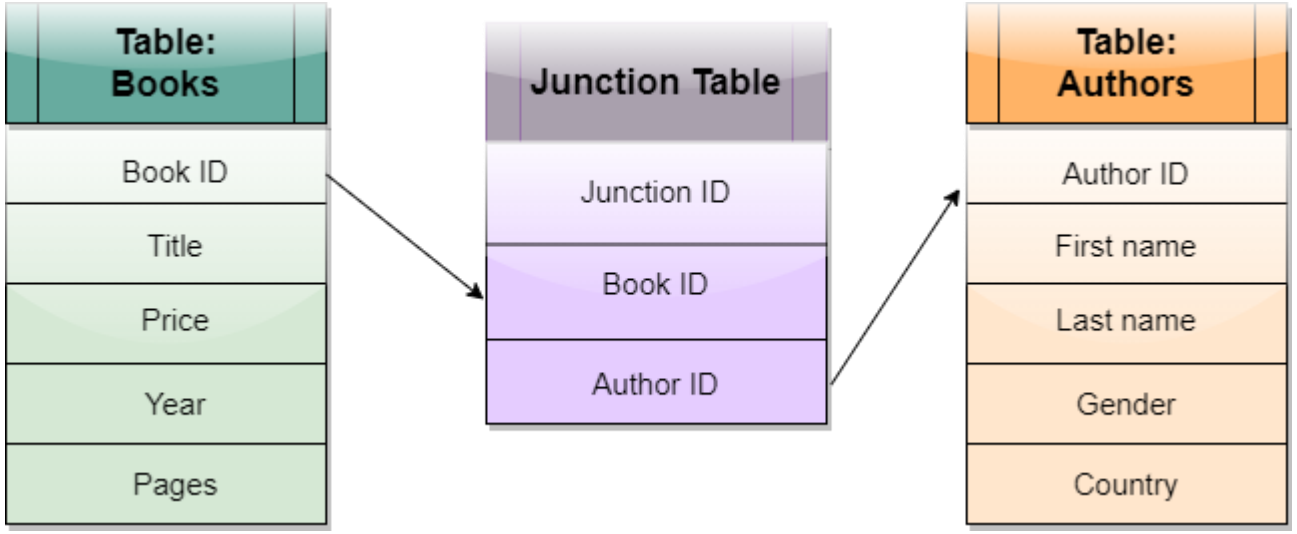
1

n

BookID	*BookName*	*AuthorID*
100	Harry Potter and the Goblet of Fire	4
101	Harry Potter and the Deathly Hallows	4
102	Murder on the Orient Express	3
103	Prayers for Rain	2
104	Death on the Nile	3
105	Harry Potter and the Chamber of Secrets	4

Many-to-Many Relationships

- Multiple records in one table are related to multiple records in another table
- Introduces a third table to define the relationship

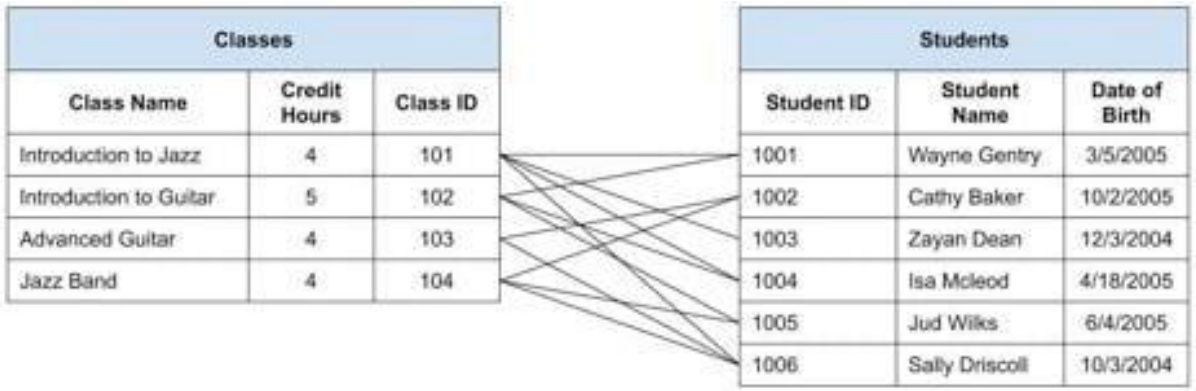


Many-to-Many Relationships

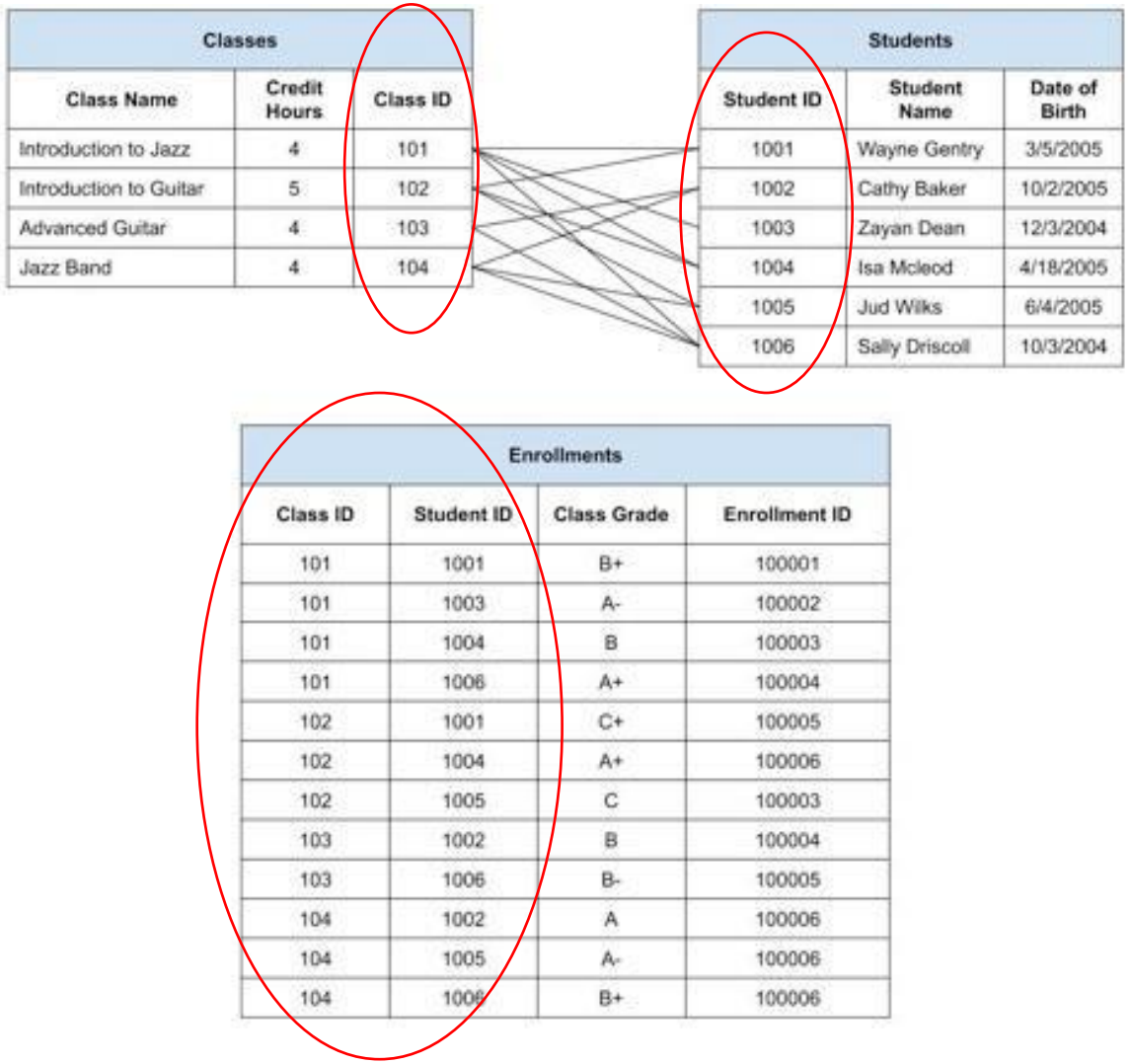
- Example: Many classes to many students

Classes		
Class Name	**Credit Hours**	**Class ID**
Introduction to Jazz	4	101
Introduction to Guitar	5	102
Advanced Guitar	4	103
Jazz Band	4	104

Students		
Student ID	**Student Name**	**Date of Birth**
1001	Wayne Gentry	3/5/2005
1002	Cathy Baker	10/2/2005
1003	Zayan Dean	12/3/2004
1004	Isa Mcleod	4/18/2005
1005	Jud Wilks	6/4/2005
1006	Sally Driscoll	10/3/2004



Many-to-Many Relationships



Classes

Class Name	Credit Hours	Class ID
Introduction to Jazz	4	101
Introduction to Guitar	5	102
Advanced Guitar	4	103
Jazz Band	4	104

Students

Student ID	Student Name	Date of Birth
1001	Wayne Gentry	3/5/2005
1002	Cathy Baker	10/2/2005
1003	Zayan Dean	12/3/2004
1004	Isa Mcleod	4/18/2005
1005	Jud Wilks	6/4/2005
1006	Sally Driscoll	10/3/2004

Enrollments

Class ID	Student ID	Class Grade	Enrollment ID
101	1001	B+	100001
101	1003	A-	100002
101	1004	B	100003
101	1006	A+	100004
102	1001	C+	100005
102	1004	A+	100006
102	1005	C	100003
103	1002	B	100004
103	1006	B-	100005
104	1002	A	100006
104	1005	A-	100006
104	1006	B+	100006

RDBMS

- Stands for **R**elational **D**atabase **M**anagement **S**ystem
- The basis for SQL, and for all modern database systems such as:
  - MS SQL Server
  - Oracle
  - MySQL
  - PostgreSQL
  - SQLite
- The data in RDBMS is stored in database objects called tables

SQL

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- There are different versions of SQL
- All of them support the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner

Wake-up

- https://youtu.be/wBbApbVjoto

Create Database and Tables

- CREATE DATABASE databasename;
- CREATE TABLE table_name (column1 datatype, column2 datatype, ...);
- https://www.w3schools.com/sql/sql_create_table.asp

```
CREATE DATABASE testDB;
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

Delete/Alter Databases and Tables

- **Delete database**
  - DROP DATABASE databasename;
- **Delete table**
  - DROP TABLE tablename;
- **Add column**
  - ALTER TABLE tablename ADD column_name datatype;
- **Delete column**
  - ALTER TABLE tablename DROP COLUMN columnname;

Query Data (SELECT)

- Query certain columns in a table
  - SELECT column1, column2, ... FROM table_name;
- Query all columns in a table
  - SELECT * FROM table_name;
- Query only unique (distinct) values
  - SELECT DISTINCT column FROM table_name;
- https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

Query Data (WHERE)

- The WHERE clause is used to filter records that fulfill a specified condition
- SELECT column1, column2 ... FROM table_name WHERE condition;
- Conditions include: =, >, <, BETWEEN, LIKE, IN, and more

```
SELECT * FROM Customers
WHERE Country='Mexico';

SELECT * FROM Customers
WHERE CustomerID=1;
```

AND, OR and NOT Operators

- The WHERE clause can be combined with AND, OR, and NOT operators to filter records based on more than one condition
- SELECT column1, column2, ... FROM table_name

WHERE condition1 AND condition2 AND condition3 ...;

- SELECT column1, column2, ... FROM table_name

WHERE condition1 OR condition2 OR condition3 ...;

- SELECT column1, column2, ... FROM table_name

WHERE NOT condition;

Adding new data

- The INSERT INTO statement is used to insert new records in a table
- Inserting data into specified columns
  - INSERT INTO table_name (column1, column2, column3, ...)
    VALUES (value1, value2, value3, ...);
- Inserting data into all columns
  - INSERT INTO table_name VALUES (value1, value2, value3, ...);

Updating data

- The UPDATE statement is used to modify existing records in a table
- UPDATE table_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

- Example:

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

Delete data

- The DELETE statement is used to delete existing records in a table.
- DELETE FROM table_name WHERE condition;
- Example:

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

JOIN

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

Orders

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Customers

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

JOIN

- The example produces the following result

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

Different Types of SQL JOINs

- (INNER) JOIN:
  - Returns records that have matching values in both tables
- LEFT (OUTER) JOIN:
  - Returns all records from left table, and matched records from the right table
- RIGHT (OUTER) JOIN:
  - Returns all records from right table, and matched records from the left table
- FULL (OUTER) JOIN:
  - Returns all matching records from both tables whether the other table matches or not