SQL Introduction
Software Development

CSc3350

Dr. William Gregory Johnson
Department of Computer Science
Georgia State University

GeorgiaState University | COLLEGE OF ARTS & SCIENCES

SQL Introduction

- SQL fundamentals
  - Background and history
  - Basic structures
  - DDL and DML
  - SQL and T-SQL
- SQL beginning statements
  - Start without anything
  - Create tables
  - Primary key
  - Foreign key
  - Data insert, update, delete
- SQL data exploration
  - Basic query
  - Filtering (WHERE clause)
  - Joining two tables for query

Background and History

- Structured Query Language initially created by IBM in early 1970's
- SEQUEL: Structured English Query Language; part of SYSTEM R, 1974
- SQL/86: ANSI & ISO standard
- SQL/89: ANSI & ISO standard
- SQL/92 or SQL2: ANSI & ISO standard
- SQL3: in the works...
- SQL2 supported by all major SQL based relational database management systems (not No-SQL)

Basic Structures

- Data Definition Language (DDL)
- Interactive Data Manipulation Language (Interactive DML)
- Embedded Data Manipulation Language (Embedded DML)
- Views
- Data Integrity

DDL and DDM

- The standard ‘data definition language’ commands are:
  - CREATE
  - ALTER
  - TRUNCATE
  - DROP
- The standard ‘data manipulation language’ commands are:
  - SELECT
  - INSERT
  - UPDATE
  - DELETE

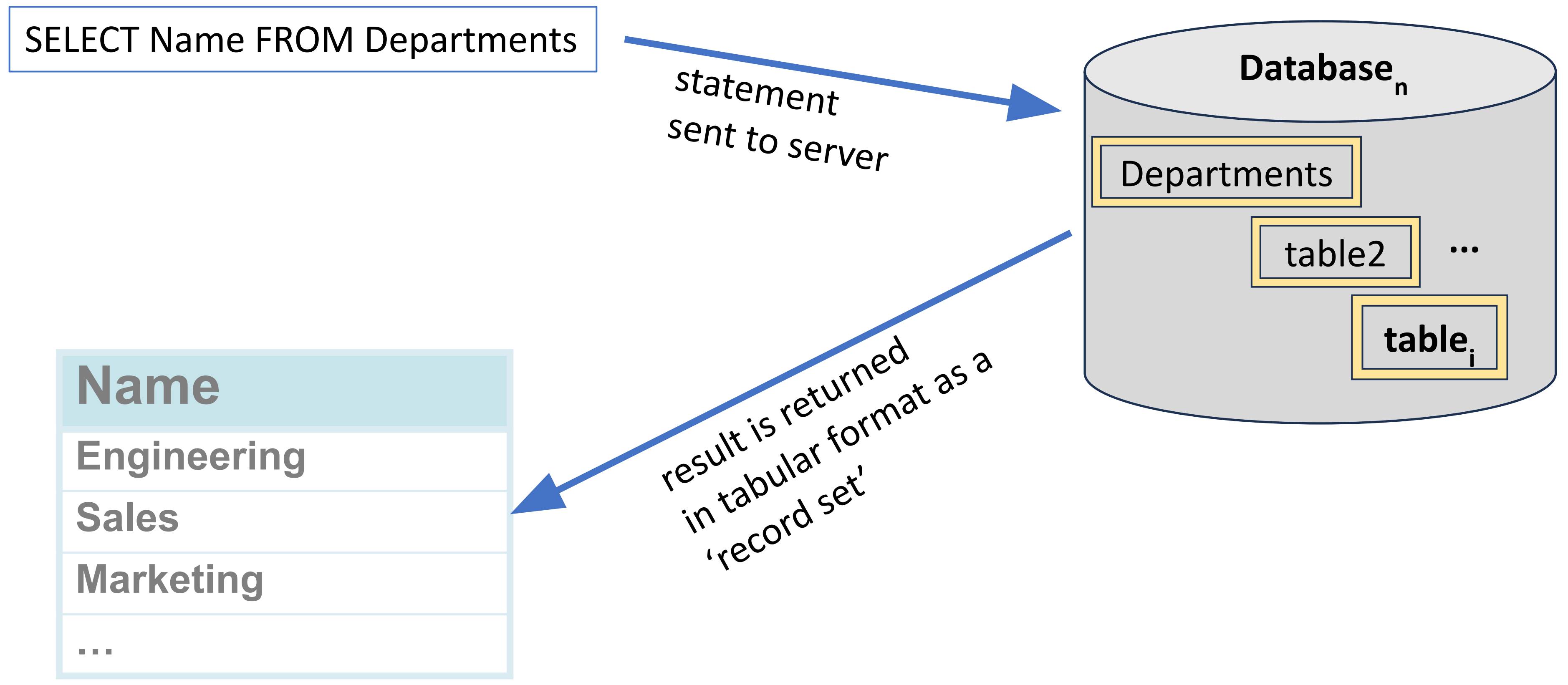
SQL and T-SQL

- The standard ‘query language’ is supported by all providers of an SQL database software system.
- Beyond the standard is ‘Transact Structured Query Language’
  - Created by Microsoft
  - Robust capability to embed programming code into a database
    - if statements, loops, exceptions
    - stored procedures, functions, triggers
  - Some providers support these extensions
  - Oracle has its own: PL/SQL

Start without anything…

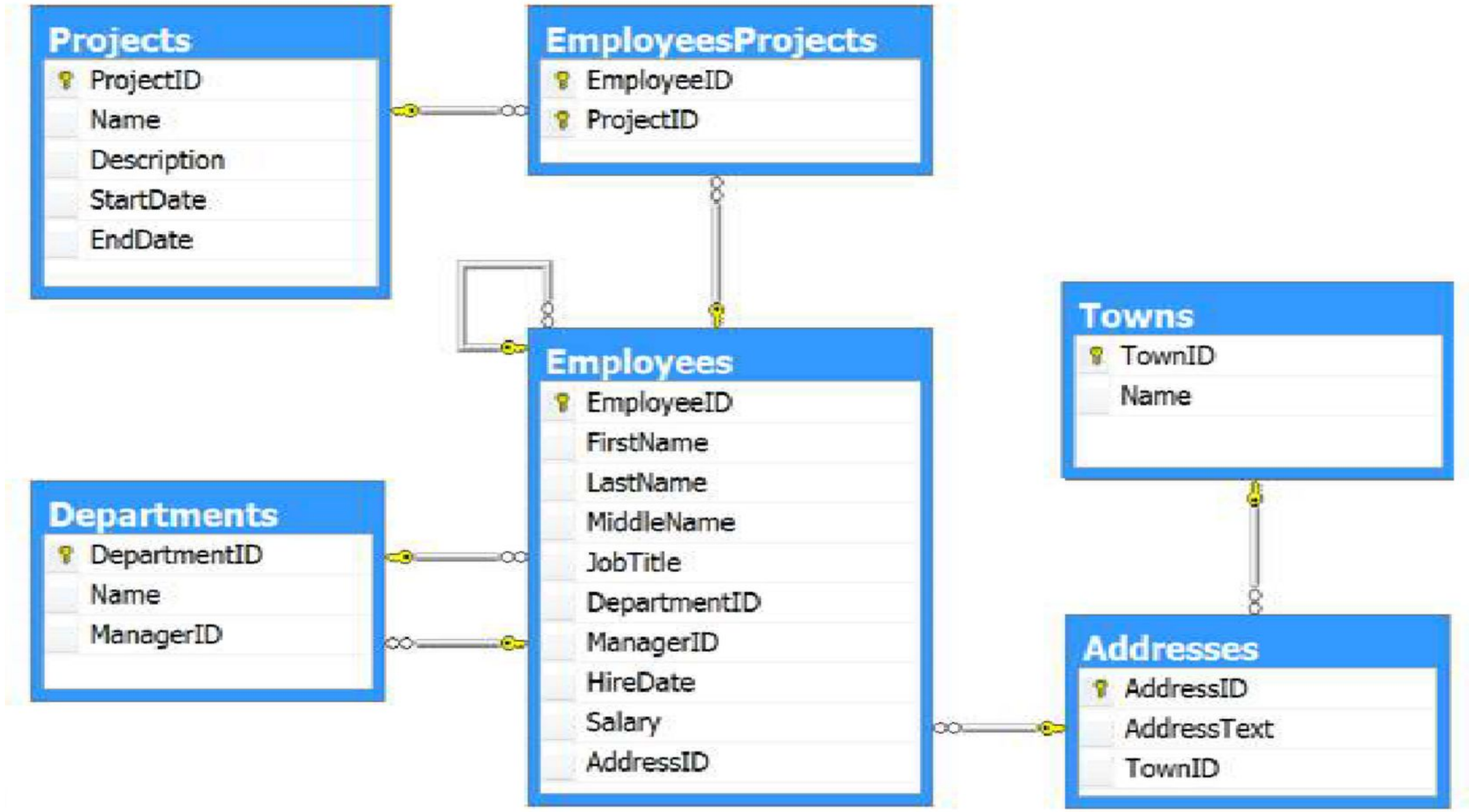
- SQL commands are executed through a database (DB) connection
- DB connection is a channel between the client and the SQL server
- DB connections take resources and should be closed when no longer used, don’t depend on OS
- Multiple clients can be connected to the SQL server at the same time

Communicating with a DB server



Name
Engineering
Sales
Marketing
…

SQL database structure (Schema)



GeorgiaState University | COLLEGE OF ARTS & SCIENCES

SQL datatypes (basic)

- String types
  - CHAR(n) – fixed-length character data, n characters long. Maximum length = 2000 bytes
  - VARCHAR(n) – variable length character data. Maximum length = 4000 bytes
  - NVARCHAR(n) – variable length UNICODE character data. Safer to use, with small amount of overhead
- Numeric types
  - NUMERIC(p,q) – general purpose numeric data type
  - INT(p) – signed integer, p digits wide
  - FLOAT(p) – floating point in scientific notation with p binary digits precision
- Date/time type
  - DATE – fixed-length date/time (dd-mm-yyyy form)

SQL Examples

```
SELECT FirstName, LastName, JobTitle FROM Employees;
```

```
SELECT *FROM Projects WHERE StartDate = ‘01/01/2023’;
```

```
INSERT INTO Projects (Name, StartDate)
VALUES (‘Introduction to SQL’, ‘01/01/2023’;
```

```
UPDATE Projects
SET EndDate = ‘08/31/2024’
WHERE StartDate = ’01/01/2023’;
```

```
DELETE FROM Projects
WHERE StartDate = ’01/01/2023’;
```

What is T-SQL?

- Transact SQL (T-SQL) is an extension created by Microsoft for the standard SQL
    - T-SQL is standard language used in Microsoft SQL Server versions.
    - Supports if statements, loops, exceptions
    - Used as a high-level procedural programming language
    - T-SQL used to write stored procedures, functions, triggers, stored in a database.

T-SQL Example

```
CREATE PROCEDURE EmpDump AS
    DECLARE @EmpID, INT, @EmpFName NVARCHAR(100), @EmpLName NVARCHAR(100)
    DECLARE emps CURSOR FOR
      SELECT EmpID, Fname, Lname FROM Employees
    OPEN emps
    FETCH NEXT FROM emps INTO @EmpID, @EmpFName, @EmpLName
    WHILE (@@FETCH_STATUS = 0) BEGIN
       PRINT CAST(@EmpID AS VARCHAR(10))+’ ‘+@EmpFName+’ ‘+@EmpLName
       FETCH NEXT FROM emps INTO @EmpID, @EmpFName, @EmpLName
    END
    CLOSE emps
    DEALLOCATE emps
GO
```

Capabilities of SQL SELECT

Projection

Take some of the columns

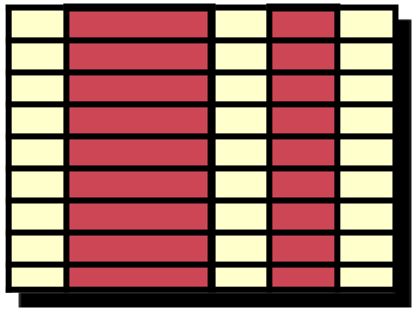


Table 1

Selection

Take some of the rows

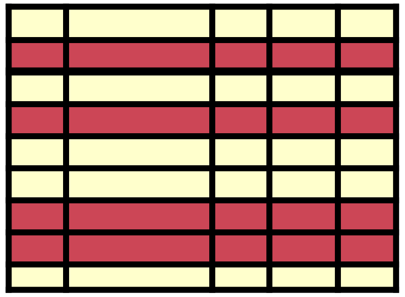


Table 2

Join

Combine tables

by some column

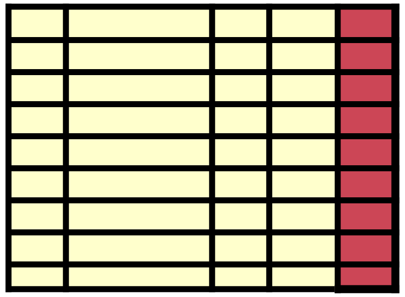


Table 1

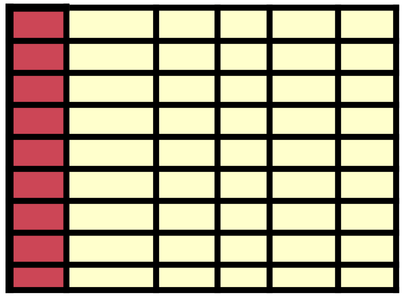


Table 2

Basic SELECT Statement

- SELECT identifies what columns (attributes)
- FROM indicates which table in current database

```
SELECT *|{[DISTINCT] column|expression [alias],. . .}
FROM table;
```

SELECT Example

- Selecting all columns from Departments table

```
SELECT *FROM Departments ORDER BY DepartmentID;
```

DepartmentID	Name	ManagerID
1	Software Dev	12
2	Engineering	4
3	Sales	190
…	…	…

- Selecting specific columns

```
SELECT DepartmentID, Name
FROM Departments;
```

DepartmentID	Name
1	Software Dev
2	Engineering
3	Sales
…	…

Arithmetic Operations

- Several operators are available: +,-,*,/,^, and others
- https://www.dataquest.io/blog/sql-operators/

```
SELECT (2+3)*4;   returns 20
```

```
SELECT Lname, Salary, Salary+300 FROM Employees;
```

Lname	Salary	(NO Column Name)
Gilbert	12500.00	12800.00
Brown	33000.00	33300.00
Patel	48000.00	48300.00

The NULL Value

- A NULL is a value that is unavailable, unassigned, unknown, or inapplicable
  - Not the same as zero or blank space
- Arithmetic expressions contain a NULL value are evaluate to NULL

```
SELECT Lname, ManagerID FROM Employees;
```

Lname	ManagerID
Sanchez	NULL
Brown	190
Johnson	4
…	…

NULL is displayed as empty space or as NULL

Georgia State University | COLLEGE OF ARTS & SCIENCES

Column Aliases

- Aliases rename a column heading and ‘name’ it to something
- Useful in calculations
- Useful in programmatic data extraction
- Immediately follows the column name
- Some SQL (not standard) are ok without the AS keyword
- Double quotation marks if alias has any spaces

```
SELECT Fname, Lname, Salary, Salary*0.2 AS “Yearly Bonus” FROM Employees;
```

Fname	Lname	Salary	Yearly Bonus
Guy	Gilbert	12500.00	2500.00
Kevin	Brown	33000.00	6600.00
Anisha	Patel	48000.00	9600.00

Concatenation Operations

- Joins columns or character strings to other columns
- Utilizes the plus sign ‘+’
- Creates a resultant column as characters in the return dataset

```
SELECT Fname+’ ‘+Lname AS “Full Name”, EmployeeID AS “No.”, Salary
FROM Employees;
```

Full Name	No.	Salary
Guy Gilbert	1	12500.00
Kevin Brown	2	33000.00
Anisha Patel	3	48000.00

Literal Character Strings

- A literal is a character, a number, or a dat3 included in the SELECT list
- Date and Character literal values must be enclosed within single quotation marks
- Each character string is output once for each row returned

```
SELECT Fname+```s last name is `+Lname AS “Our Employees”FROM Employees;
```

Our Employees
Guy’s last name is Gilbert
Kevin’s last name is Brown
Anisha’s last name is Patel
…

GeorgiaState University | COLLEGE OF ARTS & SCIENCES

Limiting the Rows Selected

- Restrict rows returned by using the WHERE clause:

```
SELECT LastName, DepartmentID
FROM Employees WHERE DepartmentID=1;
```

LastName	DepartmentID
Duffy	1
Abbas	1
Sullivan	1

```
SELECT FirstName, LastName, DepartmentID
FROM Employees WHERE LastName='Sullivan';
```

```
SELECT FirstName, LastName, Salary
FROM Employees WHERE Salary <=20000.00;
```

Other Comparison Conditions

- Use BETWEEN operator to specify a range:

```
SELECT LastName, Salary FROM Employees
WHERE Salary BETWEEN 20000 AND 22000
```

- Use IN / NOT IN to specify a set of values:

```
SELECT FirstName, LastName, ManagerID FROM
Employees WHERE ManagerID IN (109, 3, 16)
```

- Use LIKE / NOT LIKE to specify a pattern:

```
SELECT FirstName FROM Employees
WHERE FirstName LIKE 'S%'
```

Pattern Matching Example

- Special Characters:

_ Used to match any single character.

% Used to match an arbitrary number of characters.

- To find names beginning with ‘b’:

```
SELECT * FROM pet WHERE name LIKE "b%";
+--------+--------+---------+------+------------+------------+
| name   | owner | species | sex| birth | death |
+--------+--------+---------+------+------------+------------+
| Buffy| Harold | dog        | f     | 1989-05-13 | NULL        |
| Bowser | Diane    | dog      | m     | 1989-08-31 | 1995-07-29 |
+--------+--------+---------+------+------------+------------+
```

Pattern Matching Example

- To find names ending with `fy':

```
SELECT * FROM pet WHERE name LIKE "%fy";
+--------+--------+---------+------+------------+-------+
| name  | owner | species | sex| birth | death |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | cat     | f    | 1993-02-04 | NULL  |
| Buffy| Harold | dog       | f    | 1989-05-13 | NULL  |
+--------+--------+---------+------+------------+-------+
```

Pattern Matching Example

- To find names containing a ‘w’:

```
SELECT * FROM pet WHERE name LIKE "%w%";
+----------+-------+---------+------+------------+------------+
| name  | owner | species | sex   | birth | death |
+----------+-------+---------+------+------------+------------+
| Claws    | Gwen  | cat     | m    | 1994-03-17 | NULL       |
| Bowser   | Diane | dog     | m    | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen  | bird    | NULL | 1997-12-09 | NULL       |
+----------+-------+---------+------+------------+------------+
```

Pattern Matching Example

- To find names containing exactly five characters, use the _ pattern character:

```
SELECT * FROM pet WHERE name LIKE "_____";
+-------+--------+---------+------+-----------+-------+
| name  | owner | species | sex| birth | death |
+-------+--------+---------+------+-----------+-------+
| Claws | Gwen | cat        | m    | 1994-03-17 | NULL
| Buffy | Harold | dog      | f       |
+-------+--------+---------+------+-1989-05-13-+-NULL--+
                                        |
```

Regular Expression Matching

- The other type of pattern matching by using extended regular expressions.
- When you test for a match for this type of pattern, use the REGEXP and NOT REGEXP operators (or RLIKE and NOT RLIKE, which are synonyms).

Regular Expression Matching

- To find names beginning with b, use ^ to match the beginning of the name:

```
SELECT * FROM pet WHERE name REGEXP "^b";
+--------+--------+---------+------+------------+------------+
| name   | owner | species  | sex| birth | death |
+--------+--------+---------+------+------------+------------+
| Buffy | Harold | dog        | f | 1989-05-13 | NULL     |
| Bowser | Diane   |           | m | 1989-08-31 | 1995-07-29 |
dog------+--------+---------+------+------------+------------+
```

Regular Expression Matching

- To find names ending with `fy', use `$' to match the end of the name:

```
SELECT * FROM pet WHERE name REGEXP "fy$";
+--------+--------+---------+------+------------+-------+
| name   | owner | species  | sex| birth | death  |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | cat      | f     | 1993-02-04 | NULL
| Buffy | Harold | dog       | f        |
+--------+--------+---------+------+-1989-05-13-+-NULL--+
                                          |
```

Counting Rows

- Databases are often used to answer the question, "How often does a certain type of data occur in a table?"
- For example, you might want to know how many pets you have, or how many pets each owner has.
- Counting the total number of animals you have is the same question as “How many rows are in the pet table?” because there is one record per pet.
- The COUNT() function counts the number of non- NULL results.

```
SELECT COUNT(*) from pet;
```

COUNT(*)
4

Logical Operators and Brackets

- Use NOT, OR, AND operators:

```
SELECT FirstName, LastName FROM Employees
WHERE Salary>=20000.00 ANDLastName LIKE ‘s%’;
```

```
SELECT LastName FROM Employees WHERE ManagerID IS NOT NULL
AND LastName LIKE ‘%s_’;
```

```
SELECT LastName FROM Employees WHERE
NOT(ManagerID=3 OR ManagerID=4);
```

```
SELECT FirstName, LastName FROM Employees
WHERE (ManagerID=3 OR ManagerID=4)
AND (Salary >= 20000.00 OR ManagerID IS NULL);
```

Sorting with ORDER BY

- Sort rows with the ORDER BY clause:
  - ASC: ascending order, default
  - DESC: descending order

```
SELECT LastName, HireDate
FROM Employees ORDER BY HireDate;
```

LastName	HireDate
Duffy	01-02-2023
Abbas	08-15-2023
Sullivan	09-03-2023

Data from Multiple Tables

- Sometimes you need data from more than one table:

Lname	DepartmentID
Duffy	1
Abbas	2
Galvin	3

DepartmentID	DeptName
1	Engineering
2	ToolDesign
3	Sales

Lname	DeptName
Duffy	Engineering
Abbas	ToolDesign
Galvin	Sales

Cartesian Product

- This will produce Cartesian product of rows:

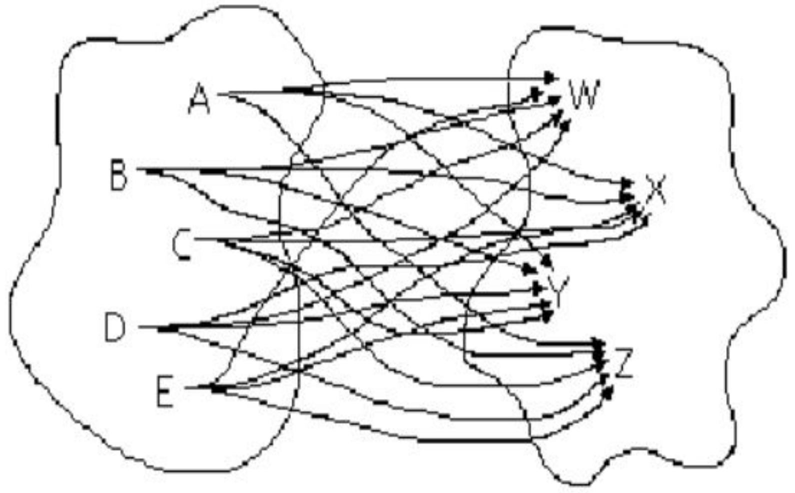
```
SELECT LastName, DeptName AS ”Department Name”
FROM Employees, Departments;
```

- The resulting dataset:

LastName	Department Name
Duffy	Document Control
Wang	Document Control
Sullivan	Document Control
Duffy	Engineering
Wang	Engineering
Sullivan	Engineering

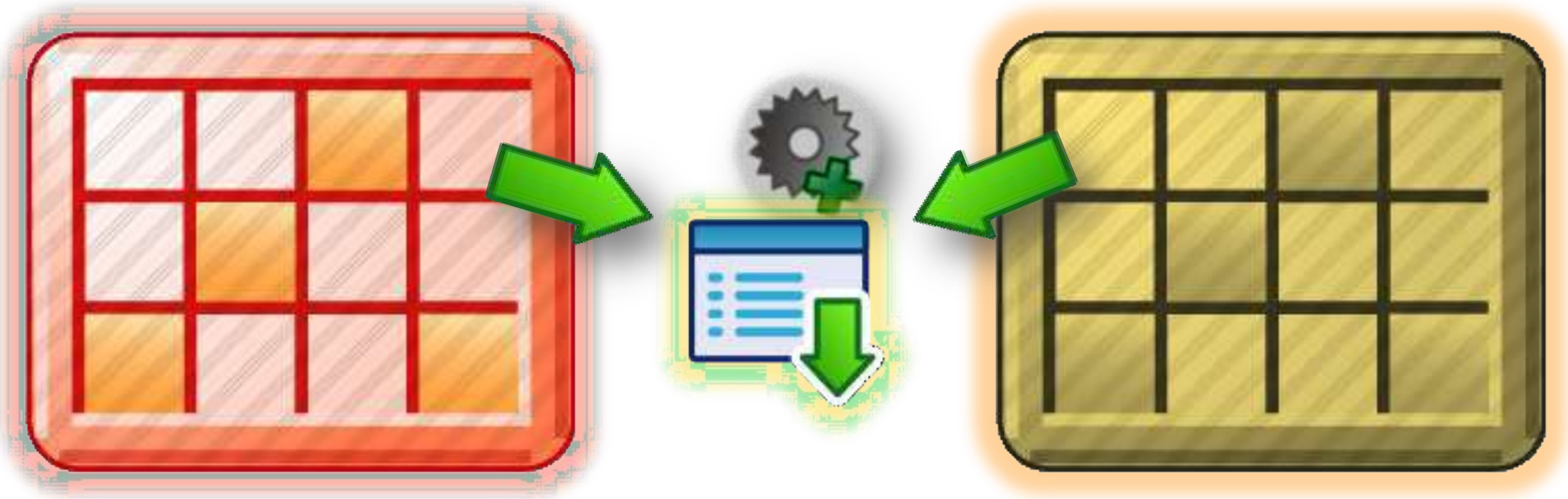
Cartesian Product (2)

- A Cartesian product is formed when:
  - A join condition is omitted
  - A join condition is invalid
  - All rows in the first table are joined to all rows in the second table
- To avoid, always include a valid join condition



Types of Joins

- Inner joins
- Left, right, and full outer joins
- Cross joins



Georgia State University | COLLEGE OF ARTS & SCIENCES

INNER JOIN with ON Clause

- To specify arbitrary conditions or specify columns to join, the ON clause is used
- Such a JOIN is also called INNER JOIN

```
SELECT e.EmployeeID, e.LastName, e.DepartmentID, d.DepartmentID, d.Name
AS “Department Name”
FROM Employees e INNER JOIN Departments d
  ON e.DepartmentID=d.DepartmentID;
```

EmployeeID	LastName	DepartmentID	DepartmentID	Department Name
1	Gilbert	7	7	Software Engineering
2	Brown	4	4	Marketing
3	Sullivan	8	8	Sales

INNER vs. OUTER Joins

- Inner join
  - A join of two tables returning only rows matching the join condition
- Left (or right) outer join
  - Returns the results of the inner join as well as unmatched rows from the left (or right) table
- Full outer join
  - Returns the results of an inner join as well as the results of a left and right join

INNER JOIN

```
SELECT e.LastName AS EmpLastName, m.EmployeeID AS MgrID,
m.LastName AS MgrLastName
FROM Employees e INNER JOIN Employees m
  ON e.ManagerID=m.EmployeeID;
```

EmpLastName	MgrID	MgrLastName
Erickson	3	Patel
Goldberg	3	Patel
Duffy	190	Sanchez
Johnson	185	Hill
Higa	185	Hill
Ford	185	Hill
Maxwell	21	Krebs
…	…	…

Attributions

1. Svetlin Nakov, http://nakov.com
2. Teleric Software Academy, http://academy.Telerik.com
3. Someshwar M. Moholkar, https://www.slideshare.net/SomeshwarMoholkar/basic-sql-and-history

Resource:

- Free tutorial for SQL fundamentals
  - https://www.dataquest.io/path/sql-skills/

GeorgiaState University | COLLEGE OF ARTS & SCIENCES

End.

CSc3350

Dr. William Gregory Johnson
Department of Computer Science
Georgia State University