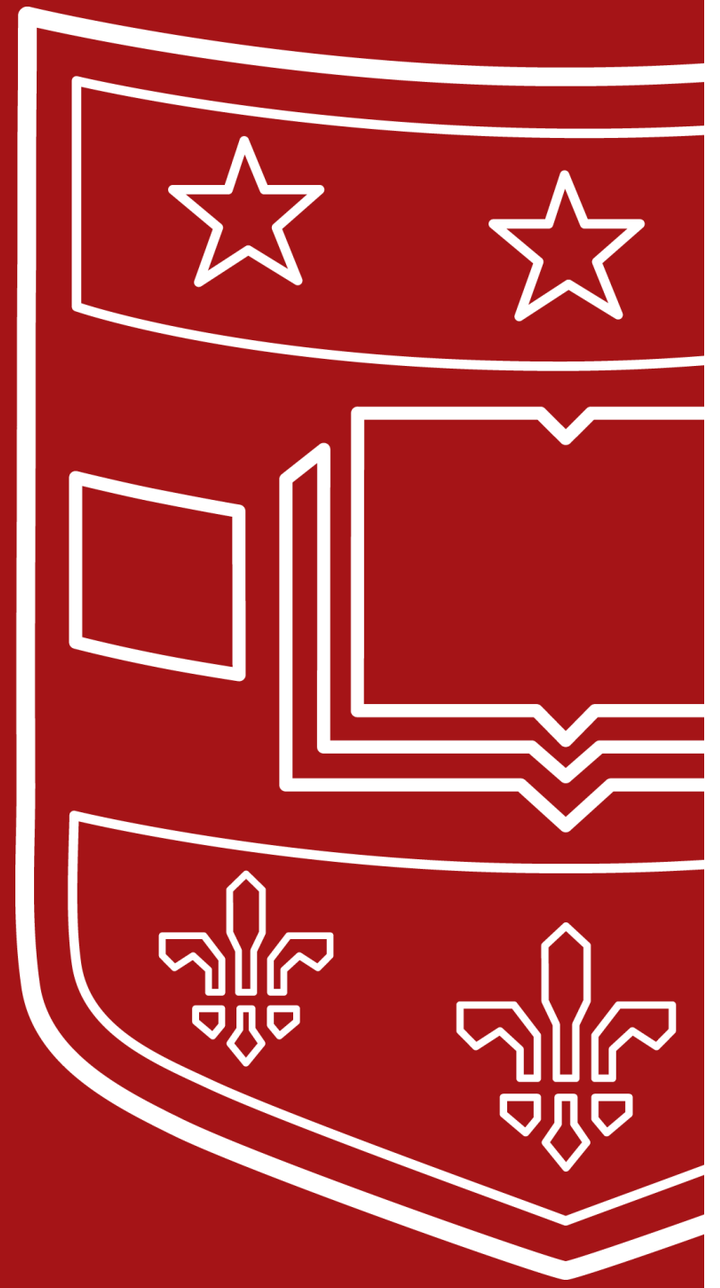
# Database Management Systems

- SQL Language Processing



Washington University in St. Louis

# Processing Language

- Known as parsing

- Multiple Techniques
  - Top Down
  - Bottom Up

- The result of parsing is what is known as a parse tree
  - Not the same as a query tree

# Grammars

- The rules that dictate how a language is parsed is called a grammar
  - What tokens do we expect to see?
  - In what order do we expect the tokens to appear?

# Simplified Grammar for Select

```
Query ::= SELECT SelectItem
          FROM FromItem
          WHERE Expression
          GROUP BY Column
SelectItem ::= Column |
               Column, SelectItem
FromItem ::= Table |
             Table JoinItem
JoinItem ::= JOIN Table ON Expression |
             JOIN Table ON Expression JoinItem
Expression ::= Column Operator Column
```

# Example Parse Tree

- SELECT FirstName, LastName
  FROM Employees

# Execution Plans

- If we can construct a valid parse tree we know the query is valid
  - But how to execute?

- We must transform this parse tree into something more useful
  - Like what?
  - How do we do this?

# Visitor Pattern

- Idea: We wish to categorize the nodes of our parse tree
  - Inheritance, interfaces, etc.

- We can then use these categories to locate nodes of interest to us
  - Example: Aggregates

# Creating A Visitor

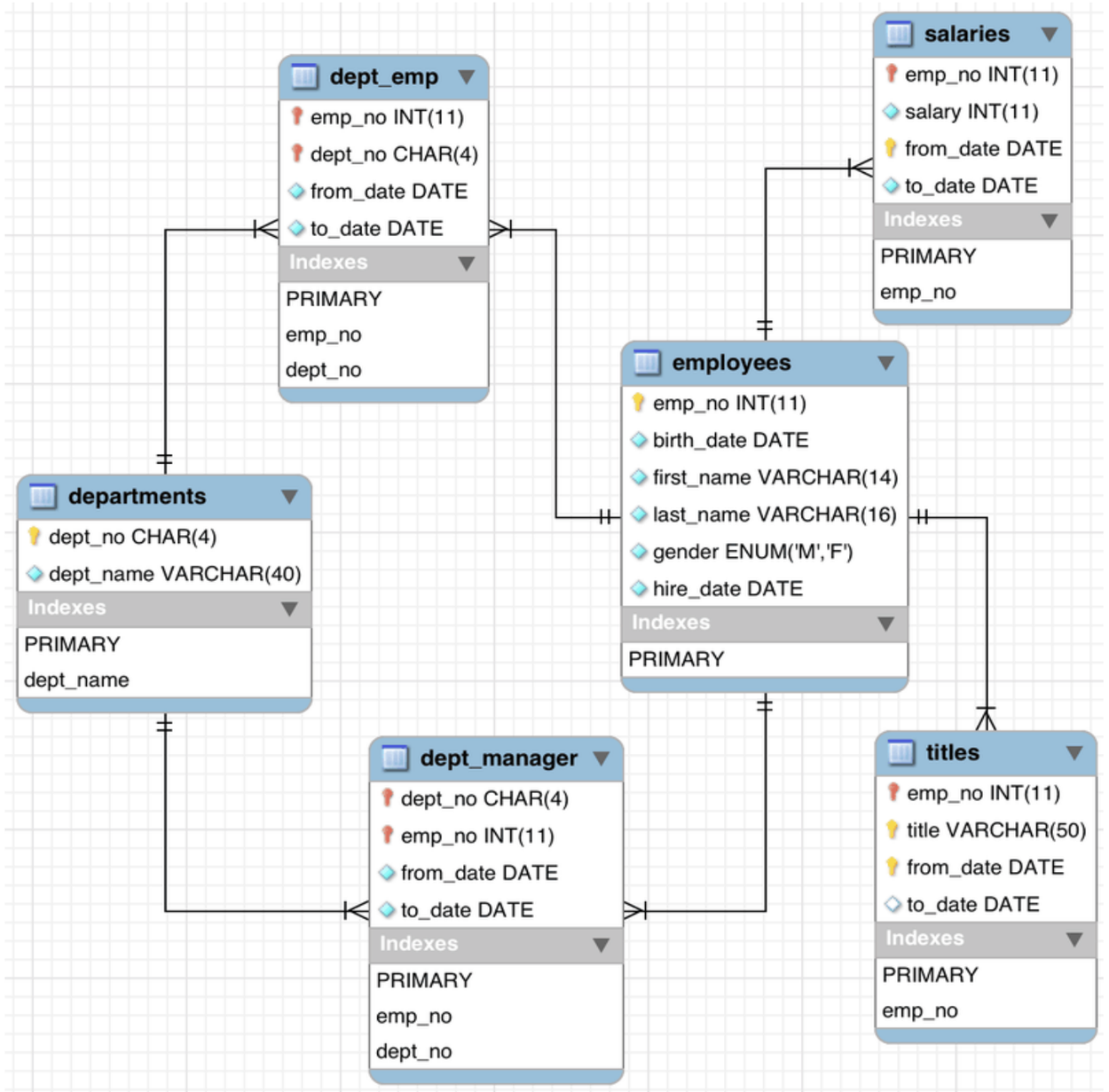
- Class contains visit methods
  - Parameter is node type of interest
- As we traverse the tree, we search for nodes that we care about
  - Start with most specific type, work towards more general
- Code in visit methods performs the desired task

# Execution Plan

- Once we have finished processing the parse tree, we now have a query tree that can be used to execute the query
  - Called an execution plan

- But are we done?

# ER Diagram


dept_emp
emp_no INT(11)
dept_no CHAR(4)
from_date DATE
to_date DATE
Indexes
PRIMARY
emp_no
dept_no
salaries
emp_no INT(11)
salary INT(11)
from_date DATE
to_date DATE
Indexes
PRIMARY
emp_no
employees
emp_no INT(11)
birth_date DATE
first_name VARCHAR(14)
last_name VARCHAR(16)
gender ENUM('M','F')
hire_date DATE
Indexes
PRIMARY
departments
dept_no CHAR(4)
dept_name VARCHAR(40)
Indexes
PRIMARY
dept_name
dept_manager
dept_no CHAR(4)
emp_no INT(11)
from_date DATE
to_date DATE
Indexes
PRIMARY
emp_no
dept_no
titles
emp_no INT(11)
title VARCHAR(50)
from_date DATE
to_date DATE
Indexes
PRIMARY
emp_no

# Practice Problem

- Write a query that will return a list of all employees who do not have a title

# Practice Problem

- Constructe a parse tree and a query tree for the following query:

```
SELECT Title, Developer, Platform
FROM VideoGame JOIN Developer ON
     VideoGame.DevID = Developer.ID
WHERE Year = 1989
```

# Practice Problem

- You are given the following relational algebra query. Construct the original SQL query.

$$\pi_{sid}(\bowtie_{enrollments.sid=students.sid}(\bowtie_{course.courseno=enrollments.courseno}(\bowtie_{department.depno=course.depno}(\sigma_{departmentname=computerscience}(Departments), Courses), Enrollments), Students))$$

# Practice Problem

- Create an ER diagram for an application that allows users to create playlists of songs. A playlist contains one or more songs. A song can have one or more artists associated with it.