CSC 742
Database Management Systems

Topic #10: SQL

Part A: Data Definition Language (DDL)

Schema and Catalog

- Schema
  - A collection of relations (tables)
  - Identified by a schema name
  - Include tables, constraints, views, domains, and others.

CREATE SCHEMA COMPANY AUTHORIZATION Bob;

- Catalog
  - A named collection of schemas.
  - Integrity constraints can be defined between relations only if they exist in the same catalog.

CREAT TABLE

- Define: a new relation
  - Specify name and attributes
  - Specify constraints
    - Attribute constraints
    - Relation constraints
      - Primary key
      - Other keys
      - Referential integrity constraint


```
CREAT TABLE EMPLOYEE
( FNAME VARCHAR(15) NOT NULL,
  LNAME VARCHAR(15) NOT NULL,
  SSN CHAR(9) NOT NULL,
  …,
  SUPERSSN CHAR(9),
  DNO INT NOT NULL,
PRIMARY KEY (SSN),
FOREIGN KEY (SUPERSSN) REFERENCE EMPLOYEE(SSN),
FOREIGN KEY (DNO) REFERENCE DEPARTMENT (DNUMBER));
```

```
CREAT TABLE DEPARTMENT
( DNAME VARCHAR(15) NOT NULL,
  DNUMBER INT NOT NULL,
  MGRSSN CHAR(9) NOT NULL,
  MGRSTARTDATE DATE,
PRIMARY KEY(DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN));
```


Referential Triggered Actions

- Two events
  - ON DELETE
  - ON UPDATE
- Three options
  - SET NULL
  - SET DEFAULT
  - CASCADE

```
CREAT TABLE DEPARTMENT
( DNAME VARCHAR(15) NOT NULL,
  DNUMBER INT NOT NULL,
  MGRSSN CHAR(9) NOT NULL,
  MGRSTARTDATE DATE,
PRIMARY KEY(DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN)
   ON DELETE SET DEFAULT  ON UPDATE CASCADE);
```

Self-Study

- DROP SCHEMA
- DROP TABLE
- ALTER TABLE
  - Alter attributes
  - Alter constraints

Part B: Data Manipulation Language (DML)

DML

- Our focus is how to formulate queries.
- Self-study:
  - INSERT
  - DELETE
  - UPDATE

SELECT

- Used for retrieval
- Used to specify subqueries for retrieval and for the other operations
- Not quite the σ or select operator of the relational algebra
- SELECT
  - tuple queries
  - aggregate queries

SELECT (Cont'd)

- Basic paradigm

  SELECT $column_1$, …, $column_n$

  FROM $table_1$, …, $table_m$

  WHERE condition
  - The WHERE condition can be a boolean combination of other conditions involving the tables $table_1$ through $table_m$

Employee

SSN	Lname	Fname	Salary
111-22-3333	Smith	John	30000
121-23-3333	Wong	Frank	45000
153-32-1342	Wallace	Jennifer	43000
154-33-3333	Borg	James	56000
555-44-5555	English	Joyce	53000

List the names of the employees whose salary is more than 50,000.

```
SELECT Lname, Fname
FROM Employee
WHERE Salary > 50000;
```

Relational Algebra:
Select + Project

Employee

Fname	Lnane	SSN
Alice	Zelaya	999-88-7777
Jennifer	Wallace	111-22-3333
Joyce	White	222-33-4444

Dependent

Fname	Lnane	ESSN
Eric	Zelaya	999-88-7777
Alex	Wallace	111-22-3333

List the names of the dependents of Alice Zelaya.

```
SELECT Dependent.Fname, Dependent.Lname
FROM Employee, Dependent
WHERE Employee.Fname = 'Alice' AND
      Employee.Lname = 'Zelaya' AND
      Employee.SSN = Dependent.ESSN;
```

Relational Algebra:
Select + Project + Join

Exercise 1.

- Find names of employees in the research dept

Employee(Fname, Lname, SSN, Bdate, Address, Sex, Salary, SuperSSN, Dno)

Department(Dname, Dnumber, MgrSSN, MgrStartDate)

```
SELECT _____, _____
FROM _____, _____
WHERE _____ AND
      _____;
```


Exercise 2

- For every project in 'Stafford' list the controlling dept number and the dept manager's last name

Employee(Fname, Lname, SSN, Bdate, Address, Sex, Salary, SuperSSN, Dno)

Department(Dname, Dnumber, MgrSSN, MgrStartDate)

Project(Pname, Pnumber, Plocation, Dnum)

```
SELECT _____, _____
FROM _____, _____, _____
WHERE _____ AND
   _____ AND _____;
```


SQL Variants: *

Employee

SSN	Lname	Fname	Salary
111-22-3333	Smith	John	30000
121-23-3333	Wong	Frank	45000
153-32-1342	Wallace	Jennifer	43000
154-33-3333	Borg	James	56000
555-44-5555	English	Joyce	53000

```
SELECT *
FROM Employee
WHERE Salary > 50000;
```


SQL Variants: ALL and DISTINCT

Employee

SSN	Lname	Fname	Salary
111-22-3333	Smith	John	30000
121-23-3333	Wong	Frank	30000
153-32-1342	Wallace	Jennifer	43000
154-33-3333	Borg	James	53000
555-44-5555	English	Joyce	53000

```
SELECT ALL Salary
FROM Employee;
```

```
SELECT DISTINCT Salary
FROM Employee;
```


SQL Variants: Renaming

Employee(Fname, Lname, SSN, Bdate, Address, Sex, Salary, SuperSSN, Dno)

Retrieve the employee's name and the names of their immediate supervisor.

```
SELECT E.Fname, E.Lname, S.Fname, S.Lname
FROM Employee AS E, Employee AS S
WHERE E.SuperSSN = S.SSN;
```

SQL Variants: IS [NOT] NULL

Employee

SSN	Lname	Fname	Salary
111-22-3333	Smith	John	30000
121-23-3333	Wong	Frank	30000
153-32-1342	Wallace	Jennifer	43000
154-33-3333	Borg	James	NULL
555-44-5555	English	Joyce	53000

```
SELECT Lname, Fname
FROM Employee
WHERE Salary IS NULL;
```

```
SELECT Lname, Fname
FROM Employee
WHERE Salary IS NOT NULL;
```


SQL Variants: ORDER BY

Employee

SSN	Lname	Fname	Salary
111-22-3333	Smith	John	30000
121-23-3333	Wong	Frank	30000
153-32-1342	Wallace	Jennifer	43000
154-33-3333	Borg	James	NULL
555-44-5555	English	Joyce	53000

```
SELECT Lname, Fname
FROM Employee
WHERE Salary IS NOT NULL
ORDER BY Salary DESC;
```


Nested Queries

- A nested query (subquery): a query in the WHERE clause of another query.
- Some paradigms of subqueries are
  - <column> [NOT] IN <subquery>
  - <column> <op> <subquery>
  - <column> <op> ANY|ALL <subquery>
  - [NOT] EXISTS<subquery>
  - <subquery> CONTAINS <subquery>

Nested Queries Example: 1

- Find all the dept 6 projects located where a dept 5 project is located

Project(Pname, Pnumber, Plocation, Dnum)

```
SELECT Project.Pnumber
FROM Project
WHERE _________________ AND
      Project.Plocation IN
      (SELECT Project.Plocation
       FROM Project
       WHERE _________________);
```


Nested Queries Example: 2

- List the name of the employee who has the highest salary.

Employee

SSN	Lname	Fname	Salary
111-22-3333	Smith	John	30000
121-23-3333	Wong	Frank	30000
153-32-1342	Wallace	Jennifer	43000
154-33-3333	Borg	James	56000
555-44-5555	English	Joyce	53000

```
SELECT E.Lname, E.Fname
FROM Employee
WHERE _____________
      (SELECT Salary FROM Employee);
```


Subqueries

- Subqueries can't modify tables
- For <column> <op> <subquery>, <subquery> can return exactly one value from one column
- *Correlated subquery*: When the subquery includes a field that is supplied from the outer query

Correlated Subqueries: 1

Employee(SSN, Lname, Fname, Dno, Address, …)
Department(Dno, Dname, Location, …)

- Find names of employees who live where their dept is located
- Taking a slight liberty with the address column

```
SELECT E.Lname, E.Fname
FROM Employee AS E
WHERE _________
        (SELECT D.Dno
         FROM Department AS D
         WHERE  D.Location = _______);
```


Correlated Subqueries: 2

- These are the most interesting kinds of subqueries
- Intuitively, the subquery is evaluated once for each tuple in the main query
- Often the same table will show up in both the main query and the subquery
- Here, the use of aliases is a necessity, not merely a convenience

CONTAINS Example

- Find SSNs of employees who work on all the projects controlled by dept 5.

Employee(Fname, Lname, SSN, Bdate, Address, Sex, Salary, SuperSSN, Dno)

Works_on(ESSN, Pno, Hours)

Project(Pname, Pnumber, Plocation, Dnum)


```
SELECT Fname, Lname
FROM Employee AS E
WHERE (  (SELECT Pno
          FROM Works_on AS W
          WHERE _____)
         CONTAINS
          (SELECT Pnumber
           FROM Project AS P
           WHERE _____) );
```


Achtung!

ANY means some

- Thus, "salary > ANY (...)" could hold for the second lowest salary from the subquery evaluated within the ANY

EXISTS

- Check whether the result of a subquery is empty.
- List the names of managers who have at least one dependent.

Department(Dno, Dname, MgrSSN, …)

Employee(Fname, Lname, SSN, …)

Dependent(Fname, Lnane, ESSN)

```
SELECT Fname, Lname
FROM Employee AS E
WHERE EXISTS  (SELECT *
        FROM Dependent AS D WHERE _____)
        AND EXISTS (SELECT *
        FROM Department AS D WHERE _____));
```

Aggregate Operators

- SQL includes operators that combine data from a single column of several tuples
  - SUM (only numeric)
  - AVG (only numeric)
  - COUNT
  - MAX and MIN
- All eliminate NULLs except COUNT(*)
- All include duplicates unless DISTINCT
- COUNT(*) includes NULLs and duplicates

List the sum of the salaries of all employees, the highest, the lowest, and the average salary,

Employee

SSN	Lname	Fname	Salary
111-22-3333	Smith	John	30000
121-23-3333	Wong	Frank	30000
153-32-1342	Wallace	Jennifer	43000
154-33-3333	Borg	James	56000
555-44-5555	English	Joyce	53000

```
SELECT SUM(Salary), MAX(Salary), MIN(Salary), AVG (Salary)
FROM Employee;
```


GROUP BY

SELECT ...
FROM …
WHERE …
GROUP BY columns

- The GROUP BY columns must appear in the SELECT list as well.

List the highest salary, the lowest, and the average salary of each department.

Employee

SSN	Lname	Fname	DNo	Salary
111-22-3333	Smith	John	1	30000
121-23-3333	Wong	Frank	1	36000
153-32-1342	Wallace	Jennifer	2	43000
154-33-3333	Borg	James	2	56000
555-44-5555	English	Joyce	3	53000

```
SELECT Dno, MAX(Salary), MIN(Salary), AVG(Salary)
FROM Employee
GROUP BY Dno;
```


HAVING

- A selection condition that applies to different groups resulting from a GROUP BY.

List the highest salary, the lowest, and the average salary of the departments whose numbers are less than 3.

Employee

SSN	Lname	Fname	DNo	Salary
111-22-3333	Smith	John	1	30000
121-23-3333	Wong	Frank	1	36000
153-32-1342	Wallace	Jennifer	2	43000
154-33-3333	Borg	James	2	56000
555-44-5555	English	Joyce	3	53000

```
SELECT Dno, MAX(Salary), MIN(Salary), AVG(Salary)
FROM Employee
GROUP BY Dno
HAVING Dno < 3;
```

Calculus to SQL: 1

A methodology for producing sound SQL queries

- Write calculus expressions
- Systematically map them to SQL queries
  - "normalize" the queries
    - Map ∀ to ∃.
  - map free variables
  - map quantifiers

- Find SSNs of employees who work on all the projects controlled by dept 5.

Employee(Fname, Lname, SSN, Bdate, Address, Sex, Salary, SuperSSN, Dno)

Works_on(ESSN, Pno, Hours)

Project(Pname, Pnumber, Plocation, Dnum)

{e.SSN|Employee(e) ∧(∀p: Project(p) ∧(p.Dnum≠5 ∨ (∃w: Works_on(w) ∧ w.Pno=p.Pnumber ∧ w.ESSN=e.SSN))}

(∀p: COND(p)) ↔ (¬∃p: ¬COND(p))
¬(A∧B) ↔(¬A ∨ ¬B)
¬(A∨B) ↔(¬A ∧ ¬B)

Calculus to SQL: 2

- With enhancements, the calculus can also be used to formulate aggregate queries in SQL
  - aggregate operators
  - bound and free variables revisited
- Generating aggregate queries logically
- Potential ambiguities in distributive and collective readings

Calculus to SQL: 3

We can use the calculus even within the modification statements of SQL

- Modification operators involve
  - a simple command, which is
  - applied to tuples selected via a subquery
- Subqueries can be formulated via the calculus just as for queries

SQL Views

- An SQL *view* is a table derived from other tables
  - base (physical) tables—ultimately depend on these defining tables
  - other views
- Views are
  - usually virtual
  - sometimes materialized

View Specification

- Views are specified with the paradigm

  CREATE VIEW view

  AS SELECT …
- The SELECT … part is the *defining query*
- In a view definition, we can
  - define column names of view
  - use aggregation (GROUP BY)

View Resolution

- Views are
  - computed by a sort of macro expansion when needed—*view resolution*
  - *query modification*: compute fresh
  - *view materialization*: store
  - always up to date: modifications to the defining tables are automatically reflected in the view

General Constraints

- SQL allows declarative constraints such as

  CREATE ASSERTION assertion-name

  CHECK (enhanced subselect query)
- The DBMS checks if any ASSERTION is ever violated