5B – SQL, THE SEQUEL

CS 1655 Introduction to Data Science

Alexandros Labrinidis – http://labrinidis.cs.pitt.edu
University of Pittsburgh

JOINS

Or, how to combine information from different tables

Cartesian Product

```
select *
from relationA, relationB
```

- Columns of result:
  - All columns of relationA plus all columns of relationB
- Rows of results:
  - Combination of all tuples from relationA with all tuples of relationB

- Cardinality of result = cardinality(A) * cardinality(B)
- Arity of result = arity(A) + arity(B)

Cartesian Product Example

People

Name	Number
Alice	4231
Bob	56
Chris	363

Offices

Name	Building	ROOM
Alice	SENSQ	5432
Bob	CL	32123
Chris	BENDM	105

```
select * from people, offices
```

Cartesian Product Example - Results

P.name	P.number	O.name	O.building	O.ROOM
Alice	4231	Alice	SENSQ	5432
Alice	4231	Bob	CL	32123
Alice	4231	Chris	BENDM	105
Bob	56	Alice	SENSQ	5432
Bob	56	Bob	CL	32123
Bob	56	Chris	BENDM	105
Chris	363	Alice	SENSQ	5432
Chris	363	Bob	CL	32123
Chris	363	Chris	BENDM	105

select * from people, offices

However…

- It makes more sense to only combine RELATED tuples

- **In other words**:
  - for tables that share a common attribute
  - for cases where the values for the common attribute are the same

Cartesian Product No More

P.name	P.number	O.name	O.building	O.ROOM
Alice	4231	Alice	SENSQ	5432
Bob	56	Bob	CL	32123
Chris	363	Chris	BENDM	105

```
select *
from people, offices
where people.name = offices.name
```

This is called a JOIN

Equijoin

```
select *
from people, offices
where people.name = offices.name
```

In general:

- Predicate must have equality condition
- Check condition before combine tuples
- All attributes are retained, from both relations

Aliasing

```
select *
from people as p, offices as o
where p.name = o.name
```

In general:

- Aliasing can happen for relation names OR for attribute names

```
select  people.name as pn
from people, offices as o
where people.name = o.name
```

Natural Join

- Similar to equijoin, but:
  - Equality predicate is “forced” on all common attributes between the two relations
  - Common attributes are only included once in the results
- Example:

```
select *
from people NATURAL JOIN offices
```

People

Name	Number
Alice	4231
Bob	56
Chris	363

Offices

Name	Building	ROOM
Alice	SENSQ	5432
Bob	CL	32123
Chris	BENDM	105

Natural Join – Results

P.name	P.number	O.Building	O.ROOM
Alice	4231	SENSQ	5432
Bob	56	CL	32123
Chris	363	BENDM	105

```
select *
from people NATURAL JOIN offices
```

Outer Join

- Outer Join has tuples that do not match the join condition
- Given two tables:
  - Table A
  - Table B
- There are three flavors of outer join:
  - **LEFT OUTER JOIN**
    - = all of A plus matches from B
  - **RIGHT OUTER JOIN**
    - = all of B plus matches from A
  - **FULL OUTER JOIN**
    - = union of left and right outer join

Outer Join Examples (1)

Name	Number
Alice	4231
Bob	56
Chris	363
David	1234

Name	Building	ROOM
Alice	SENSQ	5432
Bob	CL	32123
Chris	BENDM	105
Eve	CL	100

P.name	P.number	O.Building	O.ROOM
Alice	4231	SENSQ	5432
Bob	56	CL	32123
Chris	363	BENDM	105
David	1234	NULL	NULL
Eve	NULL	CL	100

- Blue = natural join
- Blue + Green = left outer join
- Blue + Green + Red = full outer join

Outer Join Examples (2)

Name	Number
Alice	4231
Bob	56
Chris	363
David	1234

Name	Building	ROOM
Alice	SENSQ	5432
Bob	CL	32123
Chris	BENDM	105
Eve	CL	100

P.name	P.number	O.Building	O.ROOM
Alice	4231	SENSQ	5432
Bob	56	CL	32123
Chris	363	BENDM	105
Eve	NULL	CL	100

- Blue = natural join
- Blue + Red = right outer join

AGGREGATE QUERIES

Aggregation functions

- SQL provides five standard aggregation functions
  - SUM
  - MAX
  - MIN
  - AVG
  - COUNT

- Example:

```
select sum (salary) as total_salary from employees

select count (name) as total_employees from employees
where hire_year > 2010
```

Aggregation with grouping

- Suppose you want to report the average grade **per** student year (e.g., for freshmen, sophomores, etc)

- Solution #1:
  - select avg(grade) from students where year="freshman"
  - select avg(grade) from students where year="sophomore"
  - select avg(grade) from students where year="junior"
  - select avg(grade) from students where year="senior"

- Solution #2:
  - select year, avg(grade) from students group by year

General case of aggregation

- **SELECT** *A-list, F(B)*
- **FROM** *Table1*
- **WHERE** *selection-condition*
- **GROUP BY** *A-list*
- **HAVING** ***predicate***

1. Apply where selection condition (INPUT FILTER)
2. Partition according to grouping attributes (e.g., A-list)
3. Compute aggregate functions (e.g., F(B))
4. Apply having predicate (OUTPUT FILTER)

FANCIER QUERIES

Top-K queries

- Need to limit results
- This varies from vendor to vendor
- MySQL:
  - **`SELECT`** * **`FROM`** Table **`LIMIT`** 10 **`OFFSET`** 20;
  - (mirrors functionality of going to web site and getting results page by page)

Using "IN"

- We can use "**in**" to utilize subexpressions in where clause
- **Example**:
  - select *
    from employees
    where employee_id in (select id from employee_of_the_month)
- Almost equivalent to above:
  - select *
    from employees as e, employee_of_the_month as m
    where e.employee_id = m.id
- Differences:
  - (a) number of attributes returned, and
  - (b) handling of same person being employee of the month multiple times

MODIFICATION QUERIES

Insert

- STUDENT(SID,Name,Major);

- Two forms: **Implicit** (list) and **Explicit** (set)
- **Implicit**:

```
INSERT INTO STUDENT
VALUES (165, 'Susan', 'CS');
```

- **Explicit**:

```
INSERT INTO STUDENT (SID, Name)
VALUES (165, 'Susan Jones');

INSERT INTO STUDENT (Name, SID)
VALUES ('Susan Jones', 165);
```

Delete

- Delete removes all selected tuples by the condition in the WHERE-clause

- **Examples**:

```
DELETE FROM STUDENT
WHERE SID = 165;
```

```
DELETE FROM STUDENT
WHERE Name = 'John';
```

```
DELETE FROM STUDENT;
```

```
DELETE FROM STUDENT
WHERE DNO IN
    ( SELECT DNUM
      FROM DEPT
      WHERE Dname = 'CS' );
```

Update

- Update can apply to a single relation
- Updates all the selected tuples by the condition in the WHERE-clause
- Examples:

```
UPDATE STUDENT
SET Name = 'Kathy Jones'
WHERE SID = 165;
```

```
UPDATE STUDENT
SET Major = 'CS'
WHERE  DNO IN
      ( SELECT DNUM
        FROM DEPT
        WHERE Dname = 'CS' );
```

REAL EXAMPLES