

COMP 3311

DATABASE MANAGEMENT

SYSTEMS

TUTORIAL 3

RELATIONAL ALGEBRA (RA) AND

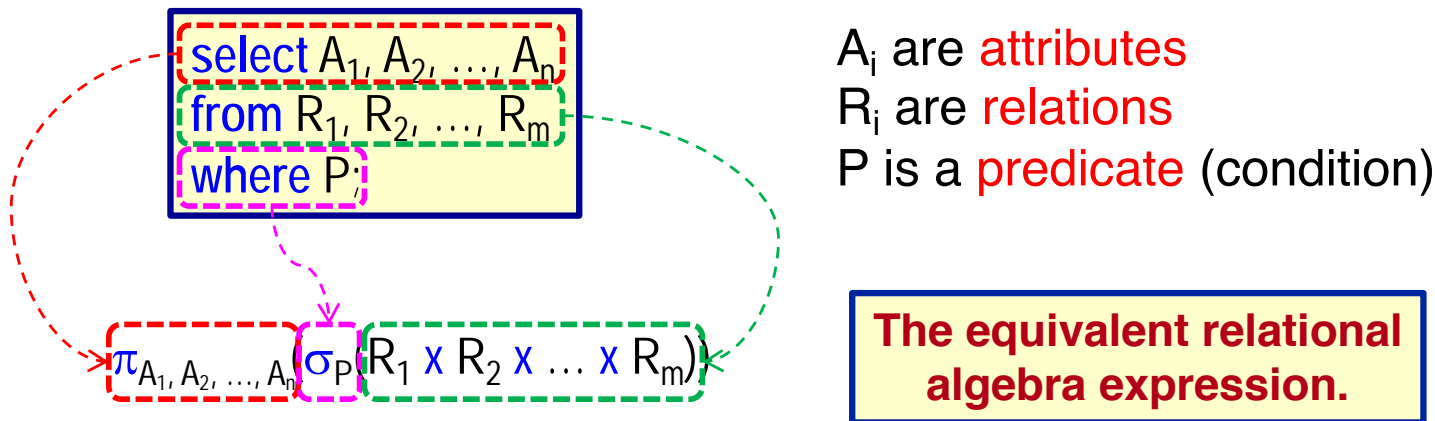
STRUCTURED QUERY LANGUAGE (SQL)

REVIEW: RELATIONAL ALGEBRA OPERATIONS

| Operation | Symbol | Action |
|-------------------|--------------|---|
| Selection | σ | Selects rows in a table that satisfy a predicate |
| Projection | π | Removes unwanted columns from a table |
| Union | \cup | Finds rows that belong to either table 1 or table 2 |
| Intersection | \cap | Finds tuples that appear in both table 1 and in table 2 |
| Set difference | $-$ | Finds rows that are in table 1, but are not in table 2 |
| Cartesian product | \times | Allows the rows in two tables to be combined |
| Join | \bowtie | Cartesian product followed by a selection |
| Assignment | \leftarrow | Assigns a result to a temporary variable |
| Rename | ρ | Allows a table and/or its columns to be renamed |

REVIEW: SQL BASIC STRUCTURE

- SQL is used in all commercial relational DBMS.
- It is based on **set** and **relational algebra** operations with certain **modifications** and **enhancements**.
- An SQL query has the basic form:



- An SQL query result is a relation (**but it may contain duplicates**).
- ✎ **SQL queries can be nested (composed).**

REVIEW: RELATIONAL ALGEBRA TO SQL

Let $R(a, b, c)$ and $S(d, e, f)$ be two union-compatible relation schemas.

Convert the following algebra expressions to SQL (for simplicity, you can omit **distinct**).

1. $\pi_a R$

```
select a
from R
```

3. $\pi_{a, f}(R \text{ JOIN}_{c=d} S)$

```
select a, f
from R, S
where c=d;
```

2. $\sigma_{c=12} R$

```
select *
from R
where c=12;
```

4. $\pi_a R - \pi_d S$

```
select a
from R
minus
select d
from S
```

EXAMPLE RELATIONAL SCHEMA

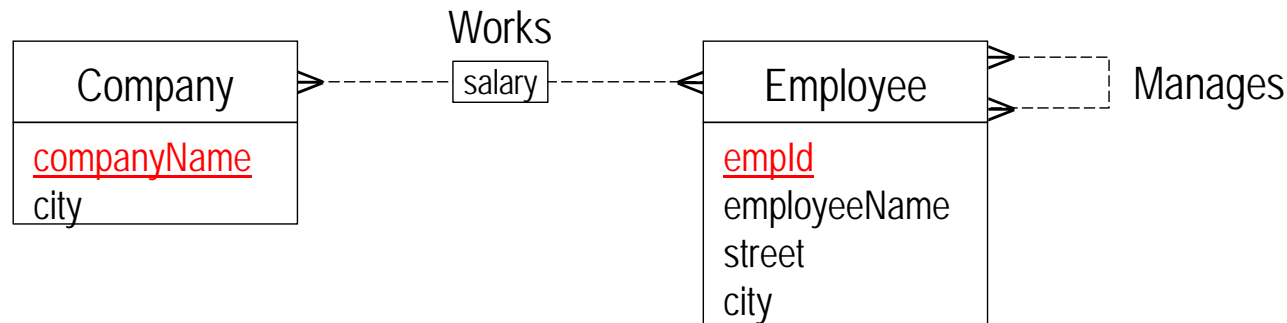
Employee(empld, employeeName, street, city)

Works(empld, companyName, salary)

Company(companyName, city)

Manages(employeeEmpld, managerEmpld)

What is the E-R schema for this relational schema?



What should be the cardinality constraints for Works? \Rightarrow N:M

What should be the participation constraints for Works? \Rightarrow unknown (ask client)

What should be the cardinality constraints for Manages? \Rightarrow N:M

What should be the participation constraints for Manages? \Rightarrow unknown (ask client)

EXAMPLE RELATIONAL SCHEMA

Attribute names in
italics are foreign
key attributes.

Employee(empld, employeeName, street, city)

Works(empld, companyName, salary)

Company(companyName, city)

Manages(employeeEmpld, managerEmpld)

Answer the first three exercises using both

RA - Relational Algebra

SQL - Structured Query Language

EXERCISE 1


Find the names of employees who earn more than \$10,000 and live in Hong Kong.

RA $\pi_{\text{employeeName}}(\sigma_{\text{salary} > 10000 \wedge \text{city} = \text{'Hong Kong'}}(\text{Employee} \bowtie \text{Works}))$


SQL

```
select Employee.employeeName
from Employee natural join Works
where salary > 10000
and city = 'Hong Kong';
```

Is it necessary to specify “Employee.employeeName” in the select clause?

 **No** The attribute `employeeName` is unique in the join result.

Is it OK to specify “Employee.empld” in the select clause?

 **No** You cannot qualify on the join attribute in a natural join.

Employee(empld, employeeName, street, city)

Works(empld, companyName, salary)

Company(companyName, city)

Manages(employeeEmpld, managerEmpld)



EXERCISE 2

Find the names of the employees who are not managers.

RA

$\pi_{\text{employeeName}}$
 $((\pi_{\text{empId}, \text{employeeName}}(\text{Employee})) -$
 $(\pi_{\text{empId}, \text{employeeName}}(\text{Employee} \bowtie_{\text{Employee.empId}=\text{Manages.managerEmpId}} \text{Manages})))$

all employees

employees who are managers

SQL

```

select employeeName
from ((select empId, employeeName
      from Employee)
minus
(select empId, employeeName
 from Employee, Manages
 where Employee.empId=Manages.managerEmpId));
    
```

Is it necessary to include “empId” in the inner select clauses?

Yes because employeeName values may not be unique.

Note that projecting on employeeName in RA will remove duplicates and so the result may not be correct. SQL retains duplicates so the result will always be correct.

Employee(empId, employeeName, street, city)

Company(companyName, city)

Works(empId, companyName, salary)

Manages(employeeEmpId, managerEmpId)



EXERCISE 3

Find the names of all persons who work for First Bank Corporation and live in the city where the company is located.

RA $\pi_{\text{employeeName}}((\text{Employee JOIN Works}) \text{ JOIN } (\sigma_{\text{companyName}=\text{'First Bank Corporation'}}(\text{Company})))$

SQL

```
select employeeName
from Employee E, Works W, Company C
where E.empld=W.empld
      and E.city=C.city
      and W.companyName=C.companyName
      and C.companyName='First Bank Corporation';
```

join

selection

Is this a solution?

👉 Yes!

```
select employeeName
from Employee natural join Works natural join Company
where companyName='First Bank Corporation';
```

Employee(empld, employeeName, street, city)

Company(companyName, city)

Works(empld, companyName, salary)

Manages(employeeEmpld, managerEmpld)



EXERCISE 4

Find all cities where employees live or where companies are located.

```
select city
from Employee
union
select city
from Company;
```

Employee(empld, employeeName, street, city)

Works(empld, companyName, salary)

Company(companyName, city)

Manages(employeeEmpld, managerEmpld)



EXERCISE 5

Find the names of all employees who work (in at least one company) and the city of the company in ascending order of employee names.

Is it OK to use natural join for this query?

```
select employeeName, C.city
from Employee E, Works W, Company C
where C.companyName=W.companyName
and E.empld=W.empld
order by employeeName asc;
```

asc is optional since it is the default ordering.

How about this query?

👉 No!

Also uses city to join Employee and Company.

```
select employeeName, city
from Employee natural join Works natural join Company
order by employeeName asc;
```

How about this query?

👉 Yes!

```
select employeeName, Company.city
from Employee natural join Works join Company using (companyName)
order by employeeName asc;
```

Employee(empld, employeeName, street, city)

Company(companyName, city)

Works(empld, companyName, salary)

Manages(employeeEmpld, managerEmpld)

EXERCISE 6

Find the names and cities of employees who work for exactly one company.

```
select employeeName, city
from Employee E
where unique (select *
              from Works
              where empld=E.empld);
```

Correct SQL but will not execute in Oracle as `unique` is not implemented.

```
select employeeName, city
from Employee E
where 1 = (select count(*)
          from Works
          where empld=E.empld);
```

Alternate query that obtains the desired result and that will execute in Oracle.

Employee(empld, employeeName, street, city)

Works(empld, companyName, salary)

Company(companyName, city)

Manages(employeeEmpld, managerEmpld)

