COMP 3311
DATABASE MANAGEMENT SYSTEMS

TUTORIAL 3
RELATIONAL ALGEBRA (RA) AND STRUCTURED QUERY LANGUAGE (SQL)

RELATIONAL ALGEBRA: BASIC 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
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
Rename	ρ	Allows a table and/or its columns to be renamed

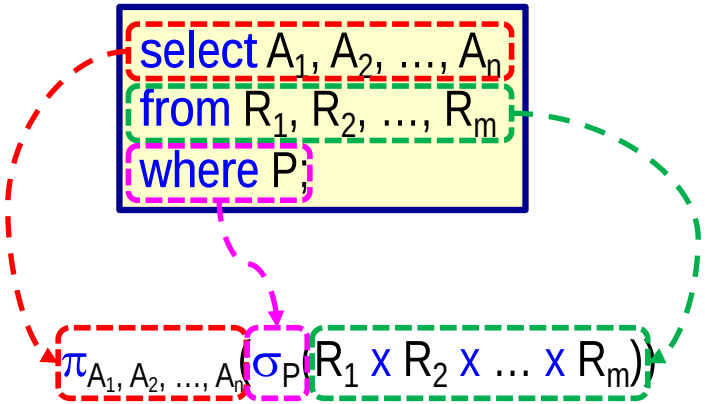
Additional operations *(not essential, but very useful)*:

Intersection	$\cap$	Finds tuples that appear in both table 1 and in table 2
Join	$\bowtie$	Cartesian product followed by a selection
Assignment	$\leftarrow$	Assigns a result to a temporary variable
Generalized Projection	π	Allows operations in the projection list
Aggregation	G	Provides aggregate functions on sets of tuples

BASIC STRUCTURE OF SQL QUERIES

- SQL is based on set and relational algebra operations with certain modifications and enhancements.

- A SQL query has the form:



A_i are attributes
R_i are relations
P is a predicate (condition)

The equivalent relational algebra expression.

- The result of an SQL query is a relation (but it may contain duplicates).

☞ **SQL queries can be nested.**

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
```

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

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

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

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

4. $\pi_a R - \pi_d S$

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

EXAMPLE RELATIONAL SCHEMA

Attribute names in italics are foreign key attributes.

Employee(empId, employeeName, street, city)

Works(*empId, companyName*, salary)

Company(companyName, city)

Manages(*employeeEmpId, managerEmpId*)

Answer the following queries using both

RA - Relational Algebra

SQL - Structured Query Language

EXAMPLE RELATIONAL SCHEMA

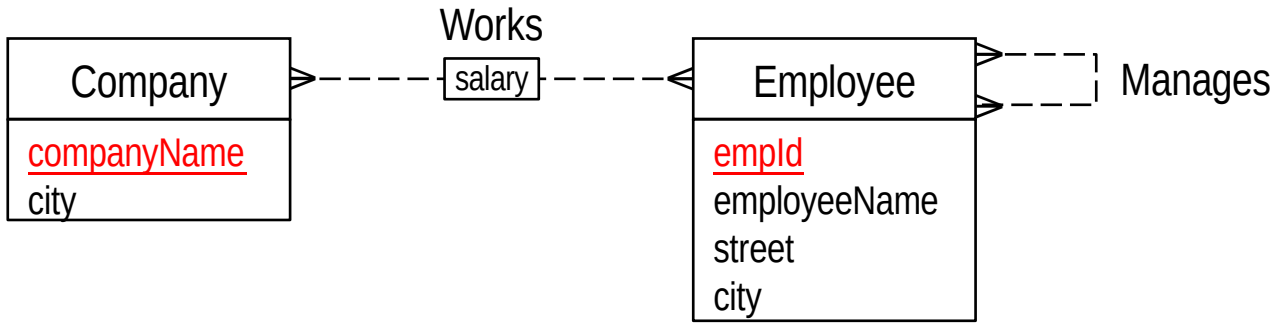
Employee(empId, employeeName, street, city)

Works(*empId, companyName*, salary)

Company(companyName, city)

Manages(*employeeEmpId, managerEmpId*)

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



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

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

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

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

EXERCISE 1

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

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

SQL

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

Is it OK to specify "Employee.employeeName" in the select clause?

☞ **No** because the attribute employeeName is unique in the join result.

Employee(empId, employeeName, street, city)

Works(*empId, companyName*, salary)

Company(companyName, city)

Manages(*employeeEmpId, managerEmpId*)

EXERCISE 2

Find the names of the employees who are not managers.

RA

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

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.

Employee(empId, employeeName, street, city)
Works(*empId, companyName*, salary)
Company(companyName, city)
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 $\pi_{employeeName}((Employee \text{ JOIN } Works) \text{ JOIN } (\sigma_{companyName='First\ Bank\ Corporation'}(Company)))$

SQL

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

join

selection

Employee(empId, employeeName, street, city)

Works(*empId, companyName*, salary)

Company(companyName, city)

Manages(*employeeEmpId, managerEmpId*)

EXERCISE 4

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

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

Employee(empId, employeeName, street, city)

Works(*empId*, *companyName*, salary)

Company(companyName, city)

Manages(*employeeEmpId*, *managerEmpId*)

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.

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

asc is optional since it is the default ordering.

Employee(empId, employeeName, street, city)

Works(*empId*, *companyName*, salary)

Company(companyName, city)

Manages(*employeeEmpId*, *managerEmpId*)

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 empId=E.empId);
```

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 empId=E.empId);
```

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

Employee(empId, employeeName, street, city)

Works(*empId, companyName*, salary)

Company(companyName, city)

Manages(*employeeEmpId, managerEmpId*)