INFO20003 Database Systems

Xiuge Chen

Tutorial 5

2021.03.29

Agenda

1. **Relational algebra (RA) review - 15 min**
2. **Relational algebra and SQL statements - 35 min**

Notice

1. Assignment 1 has released - LMS Assessments
2. due date: **10:00 am Saturday 03 April**
3. Tips:
    - Try modeling practice first - LMS Practice on your own
    - Read case study multiple times before designing
    - Derive from case study not real world examples
    - Subjective process, many possible solutions
    - Every time make a choice, list assumptions (400 words)
    - Carefully follow the rules about transforming models

Relational algebra (RA) review

Relation algebra

- procedural query language for relational model
- provide theoretical foundation for RD and SQL
- consists of a collection of **operators** (unary/binary)
- **operand**: instance(s) of a relation, returns a relation instance.
- **Five basic operators** of Relational Algebra that can form other compound operators

Relational algebra (RA) review

Fundamental operations

Removal operators: Selection (σ) and Projection (π)

- **Projection**:
  - $\pi_{A1, A2, \ldots, An}$ (R) where R is relation and A are attributes that 'projected'
  - Create new relation with a **subset of attributes**
  - All tuples are included, but only **chosen attributes** are kept
  - Projection operator has to *eliminate duplicates*
- **Selection**:
  - σC (R) where R is relation and C is condition used to **filter rows**
  - Create new relation consisting of those **rows** for which C is true

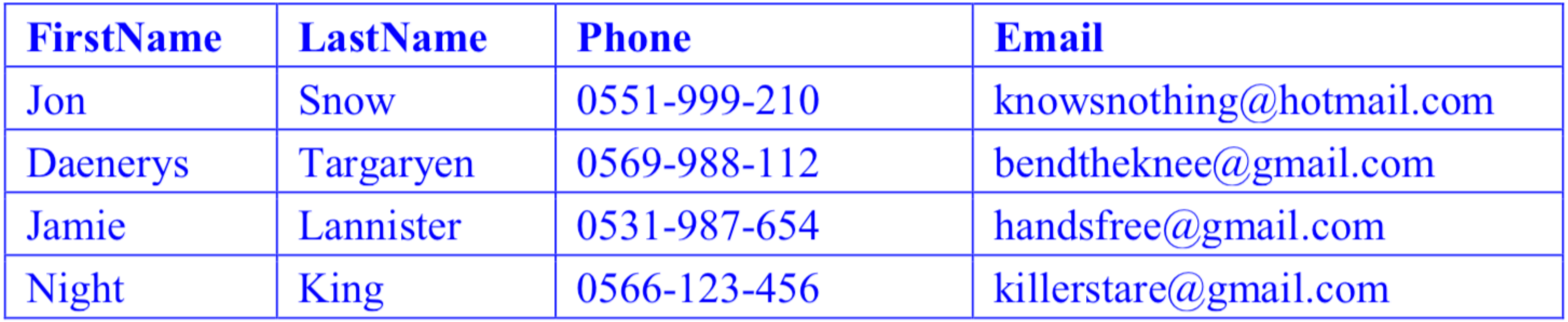
Projection Example

FirstName	LastName	Phone	Email
Jon	Snow	0551-999-210	knowsnothing@hotmail.com
Daenerys	Targaryen	0569-988-112	bendtheknee@gmail.com
Jamie	Lannister	0531-987-654	handsfree@gmail.com
Night	King	0566-123-456	killerstare@gmail.com

The expression $\pi_{\text{FirstName, LastName}}$ (Person) will result in:

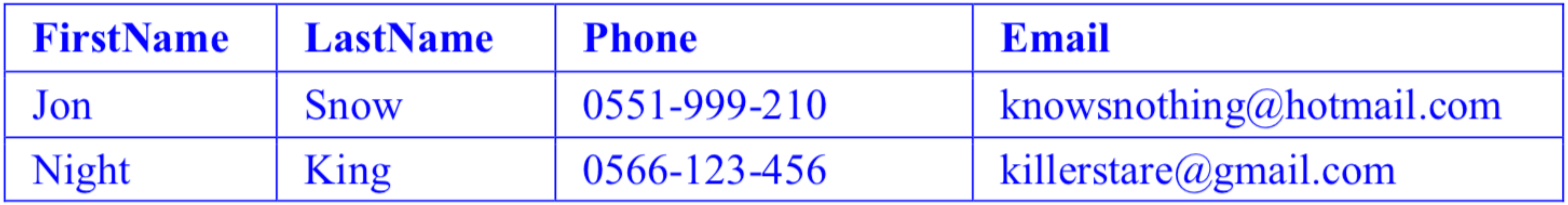
FirstName	LastName
Jon	Snow
Daenerys	Targaryen
Jamie	Lannister
Night	King

Selection Example



FirstName	LastName	Phone	Email
Jon	Snow	0551-999-210	knowsnothing@hotmail.com
Daenerys	Targaryen	0569-988-112	bendtheknee@gmail.com
Jamie	Lannister	0531-987-654	handsfree@gmail.com
Night	King	0566-123-456	killerstare@gmail.com

$$\sigma_{\text{FirstName} = \text{'Jon'} \vee \text{LastName} = \text{'King'}}(\text{Person})$$



FirstName	LastName	Phone	Email
Jon	Snow	0551-999-210	knowsnothing@hotmail.com
Night	King	0566-123-456	killerstare@gmail.com

Selection Projection Combination

FirstName	LastName	Phone	Email
Jon	Snow	0551-999-210	knowsnothing@hotmail.com
Daenerys	Targaryen	0569-988-112	bendtheknee@gmail.com
Jamie	Lannister	0531-987-654	handsfree@gmail.com
Night	King	0566-123-456	killerstare@gmail.com

$$\pi_{\text{FirstName, LastName}}(\sigma_{\text{FirstName} = \text{'Jon'} \lor \text{LastName} = \text{'King'}}(\text{Person}))$$

FirstName	LastName
Jon	Snow
Night	King

Relational algebra (RA) review

Fundamental operations

Set operators: Set-difference (–) and Union (U)

- **constraint:** both relations must have the same attributes with the same domains. the ordering of attributes should be kept consistent.
- **Set-difference**:
  - **R – S.** result will be every row which is in R but not in S
- **Union**:
  - **R ∪ S.** result will be every row which is either in R or S
  - Duplicates are removed

Union Example

GoodGuys

FirstName	LastName
Jon	Snow
Daenerys	Targaryen

BadGuys

FirstName	LastName
Cersei	Lannister
Night	King

GoodGuys ∪ BadGuys will result in:

FirstName	LastName
Jon	Snow
Daenerys	Targaryen
Cersei	Lannister
Night	King

Difference Example

RandomCombo1

FirstName	LastName
Jon	Snow
Daenerys	Targaryen
Jamie	Lannister
Night	King

RandomCombo2

FirstName	LastName
Night	King
Arya	Stark
Cersei	Lannister
Daenerys	Targaryen

RandomCombo1 – RandomCombo2 will result in:

FirstName	LastName
Jon	Snow
Jamie	Lannister

Relational algebra (RA) review

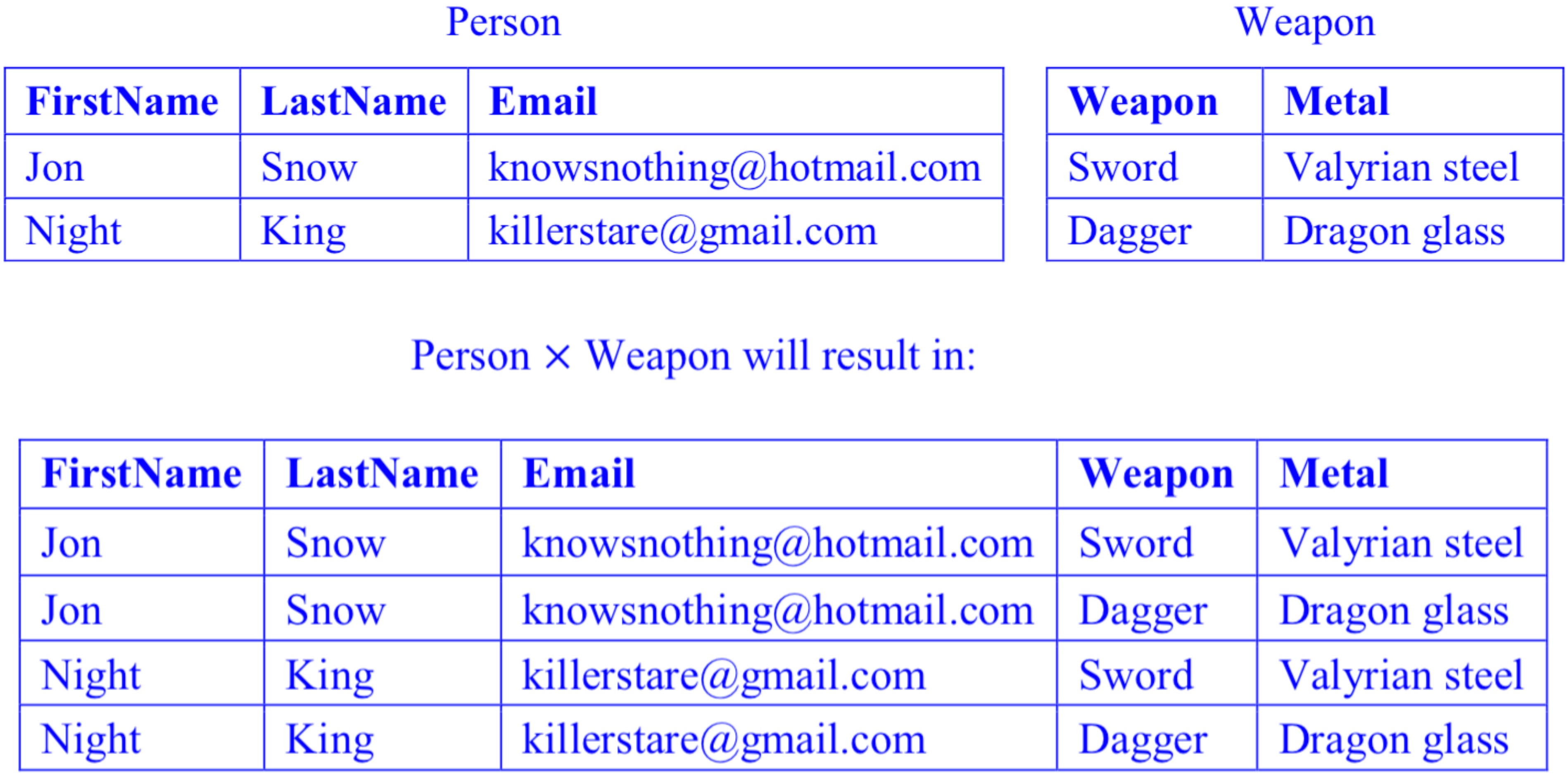
Fundamental operations

Set operators: Cross Product (×)

- **Cross Product** (×):
  - Each row of R pairs with each row of S. The resulting schema has all the attributes from both relations. If some attributes have same name, rename them by using renaming operator.

$$\rho\,(\underset{\text{Result relation name}}{C}(1 \rightarrow sid1, 5 \rightarrow sid2),\ S1 \times R1)$$

Cross Product Example



Person

FirstName	LastName	Email
Jon	Snow	knowsnothing@hotmail.com
Night	King	killerstare@gmail.com

Weapon

Weapon	Metal
Sword	Valyrian steel
Dagger	Dragon glass

Person × Weapon will result in:

FirstName	LastName	Email	Weapon	Metal
Jon	Snow	knowsnothing@hotmail.com	Sword	Valyrian steel
Jon	Snow	knowsnothing@hotmail.com	Dagger	Dragon glass
Night	King	killerstare@gmail.com	Sword	Valyrian steel
Night	King	killerstare@gmail.com	Dagger	Dragon glass

Relational algebra (RA) review

Compound operations

- operators are not adding any computational power to the language but are useful shorthand.
- All these operators can be expressed using the basic operators

Relational algebra (RA) review

Compound operations

set operator: Intersection(∩)

- **Intersection** (∩):
  - union compatible
  - result is a relation containing all the tuples which are present in both relations

$$R \cap S = R - (R - S)$$

Relational algebra (RA) review

Intersection Example

RandomCombo1

FirstName	LastName
Jon	Snow
Daenerys	Targaryen
Jamie	Lannister
Night	King

RandomCombo2

FirstName	LastName
Night	King
Arya	Stark
Cersei	Lannister
Daenerys	Targaryen

RandomCombo1 $\cap$ RandomCombo2 will result in:

FirstName	LastName
Daenerys	Targaryen
Night	King

Relational algebra (RA) review

Compound operations

set operator: Natural Join(⋈)

- **Natural Join(⋈)**:
  - identifies attributes common to each relation
  - pairing each tuple from R and S where the common attributes are equal
  - In general are compound operators involving cross product, selection and occasionally projection
  - omit duplicate attributes

Relational algebra (RA) review

Compound operations

set operator: Natural Join(⋈)

- **Natural Join(⋈) steps**:
  - Compute R × S
  - Select rows where attributes that appear in both relations have equal values.
  - Project all unique attributes and one copy of each of the common ones.

Relational algebra (RA) review

Natural Join Example

Person

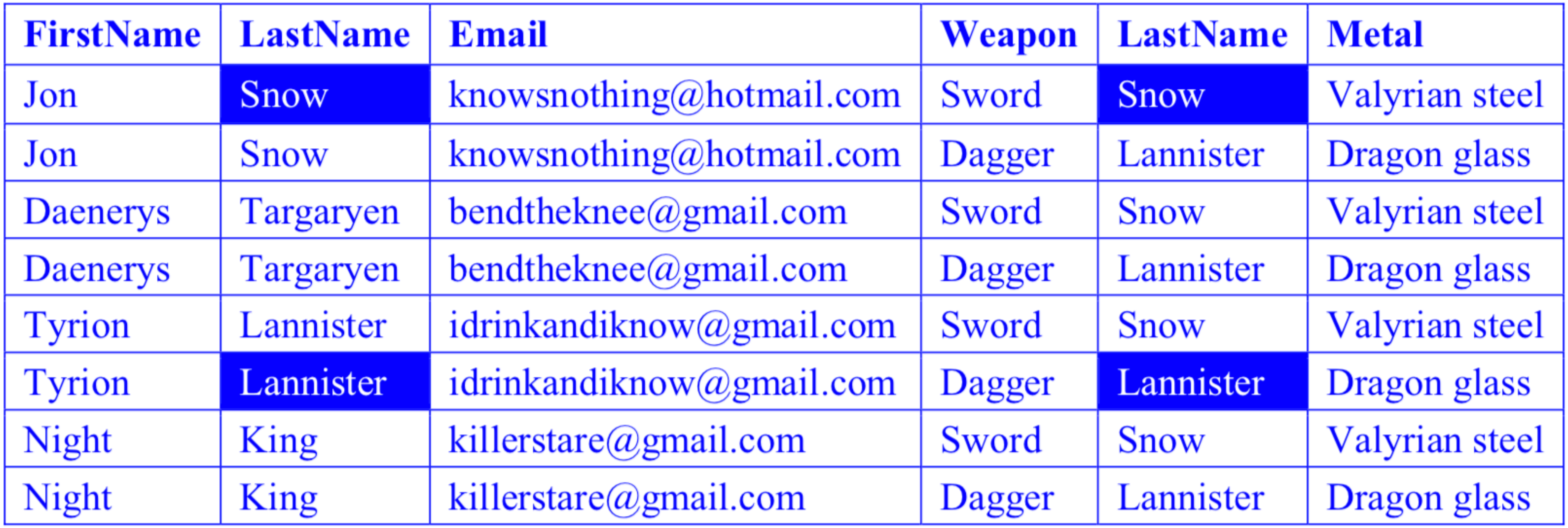
FirstName	LastName	Email
Jon	Snow	knowsnothing@hotmail.com
Daenerys	Targaryen	bendtheknee@gmail.com
Tyrion	Lannister	idrinkandiknow@gmail.com
Night	King	killerstare@gmail.com

WeaponOwner

Weapon	LastName	Metal
Sword	Snow	Valyrian steel
Dagger	Lannister	Dragon glass

Person × Weapon (intermediate result):

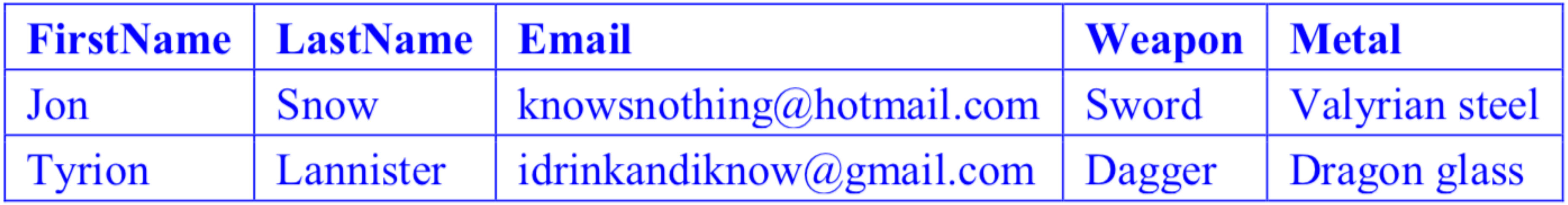
Natural Join Example



FirstName	LastName	Email	Weapon	LastName	Metal
Jon	Snow	knowsnothing@hotmail.com	Sword	Snow	Valyrian steel
Jon	Snow	knowsnothing@hotmail.com	Dagger	Lannister	Dragon glass
Daenerys	Targaryen	bendtheknee@gmail.com	Sword	Snow	Valyrian steel
Daenerys	Targaryen	bendtheknee@gmail.com	Dagger	Lannister	Dragon glass
Tyrion	Lannister	idrinkandiknow@gmail.com	Sword	Snow	Valyrian steel
Tyrion	Lannister	idrinkandiknow@gmail.com	Dagger	Lannister	Dragon glass
Night	King	killerstare@gmail.com	Sword	Snow	Valyrian steel
Night	King	killerstare@gmail.com	Dagger	Lannister	Dragon glass

Natural Join Example

Person ⋈ **Weapon** will result in:



FirstName	**LastName**	**Email**	**Weapon**	**Metal**
Jon	Snow	knowsnothing@hotmail.com	Sword	Valyrian steel
Tyrion	Lannister	idrinkandiknow@gmail.com	Dagger	Dragon glass

Compound operations

set operator: Condition Join (Theta/Inner Join)

- **Condition Join(⋈$_c$) steps**:
  - R ⋈$_c$ S joins rows from relation R and S such that the Boolean condition C is true
  - commonly C is of the type A = B, making an “equi-join”.

$$R \bowtie_C S = \sigma_C(R \times S)$$

Condition Join Example

Person

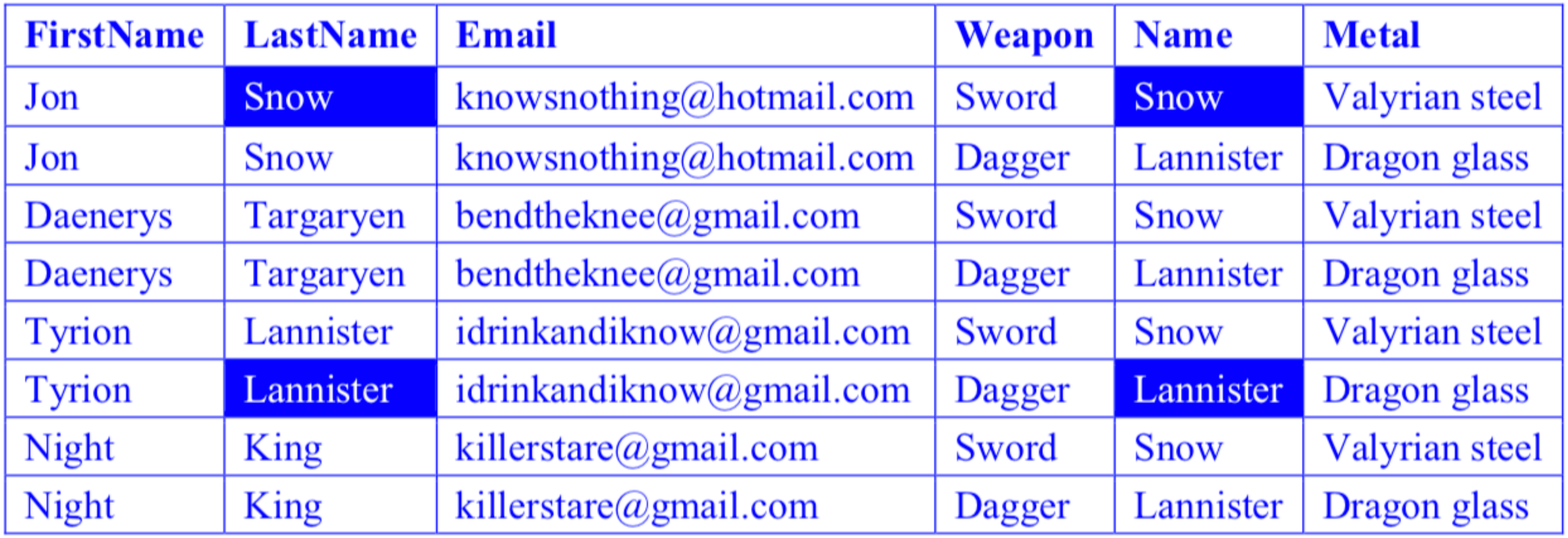
FirstName	LastName	Email
Jon	Snow	knowsnothing@hotmail.com
Daenerys	Targaryen	bendtheknee@gmail.com
Tyrion	Lannister	idrinkandiknow@gmail.com
Night	King	killerstare@gmail.com

WeaponOwner

Weapon	Name	Metal
Sword	Snow	Valyrian steel
Dagger	Lannister	Dragon glass

Person × Weapon (intermediate result):

Condition Join Example



FirstName	LastName	Email	Weapon	Name	Metal
Jon	Snow	knowsnothing@hotmail.com	Sword	Snow	Valyrian steel
Jon	Snow	knowsnothing@hotmail.com	Dagger	Lannister	Dragon glass
Daenerys	Targaryen	bendtheknee@gmail.com	Sword	Snow	Valyrian steel
Daenerys	Targaryen	bendtheknee@gmail.com	Dagger	Lannister	Dragon glass
Tyrion	Lannister	idrinkandiknow@gmail.com	Sword	Snow	Valyrian steel
Tyrion	Lannister	idrinkandiknow@gmail.com	Dagger	Lannister	Dragon glass
Night	King	killerstare@gmail.com	Sword	Snow	Valyrian steel
Night	King	killerstare@gmail.com	Dagger	Lannister	Dragon glass

Relational algebra (RA) review

Condition Join Example

Person $\bowtie_{\text{LastName = Name}}$ **Weapon** will result in:

FirstName	**LastName**	**Email**	**Weapon**	**Name**	**Metal**
Jon	Snow	knowsnothing@hotmail.com	Sword	Snow	Valyrian steel
Tyrion	Lannister	idrinkandiknow@gmail.com	Dagger	Lannister	Dragon glass

Relational algebra (RA) review

Any questions?

Structured Query Language(SQL)

- Domain-specific language
- Language for data manipulation in RD
- Allow to create/delete tables, add/update/remove data, etc

Structured Query Language(SQL)

- **Data Definition Language (DDL)**: To define and setup the database CREATE, ALTER, DROP
- **Data Manipulation Language (DML)**: To maintain and use the database, SELECT, INSERT, DELETE, UPDATE
- **Data Control Language (DCL)**: To control access to the database, GRANT, REVOKE
- **Other Commands**: Administer the database, Transaction Control

Structured Query Language(SQL)

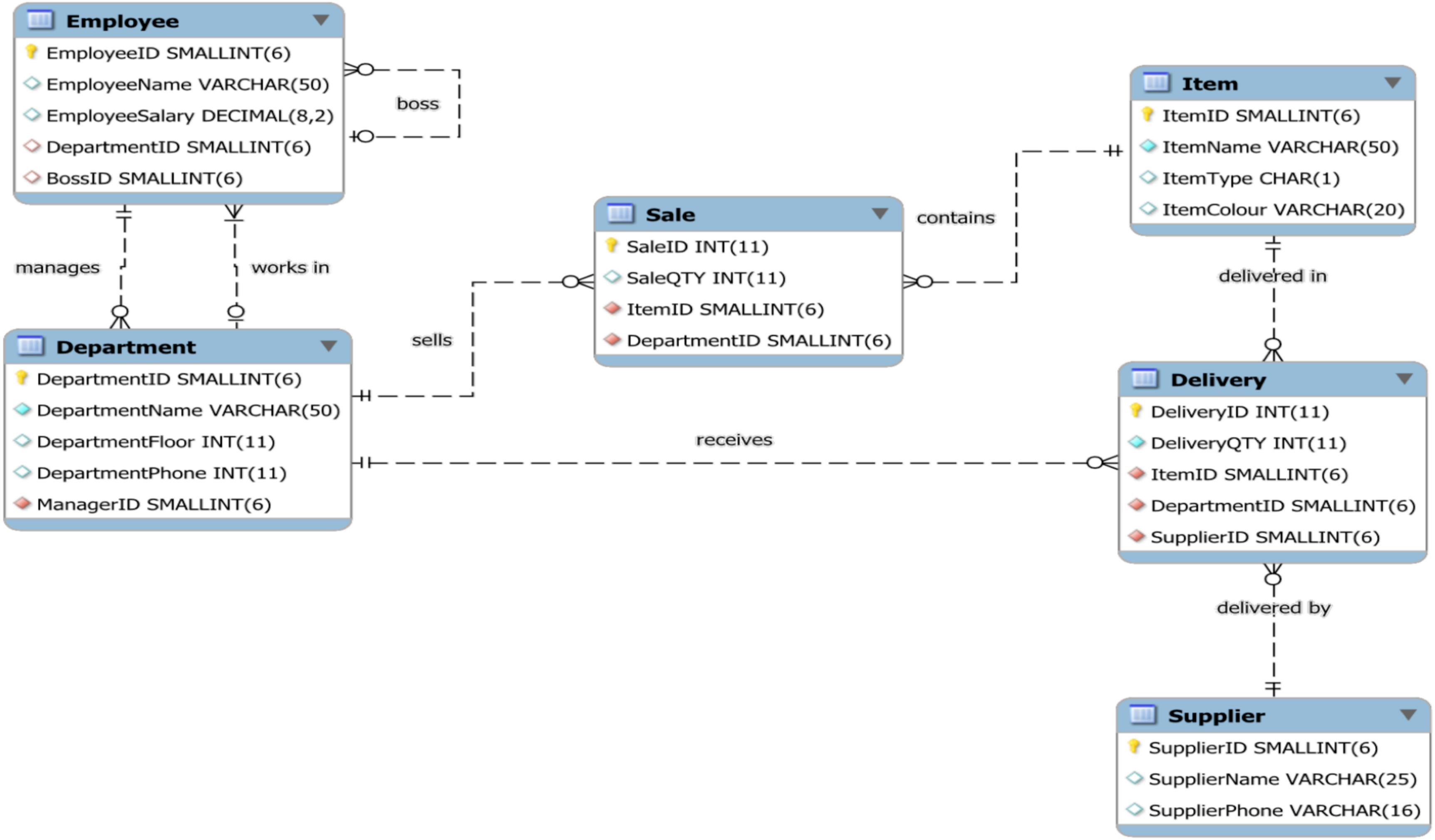
- SELECT [ALL | DISTINCT] *select_expr* [, *select_expr* ...]
  - List the columns (and expressions) that are returned from the query
- [FROM *table_references*]
  - Indicate the table(s) or view(s) from where the data is obtained
- [WHERE *where_condition*]
  - Indicate the conditions on whether a particular row will be in the result
- [GROUP BY {*col_name* | *expr* } [ASC | DESC], ...]
  - Indicate categorisation of results
- [HAVING *where_condition*]
  - Indicate the conditions under which a particular category (group) is included in the result
- [ORDER BY {*col_name* | *expr* | *position*} [ASC | DESC], ...]
  - Sort the result based on the criteria
- [LIMIT {[*offset*,] *row_count* | *row_count* OFFSET *offset*}]
  - Limit which rows are returned by their return order (ie 5 rows, 5 rows from row 2)

Structured Query Language(SQL) review

Any questions?

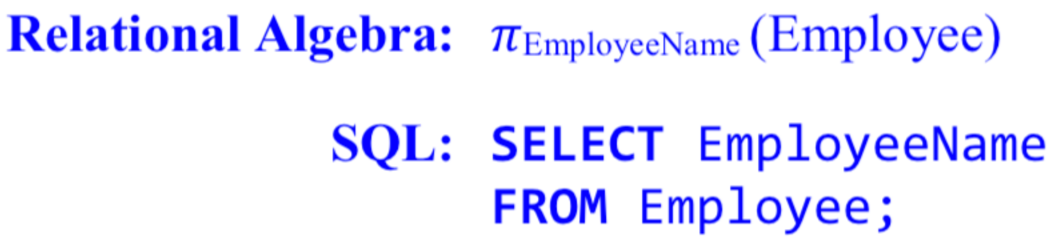
Relational algebra and SQL statements

Consider the following schema:



Relational algebra and SQL statements

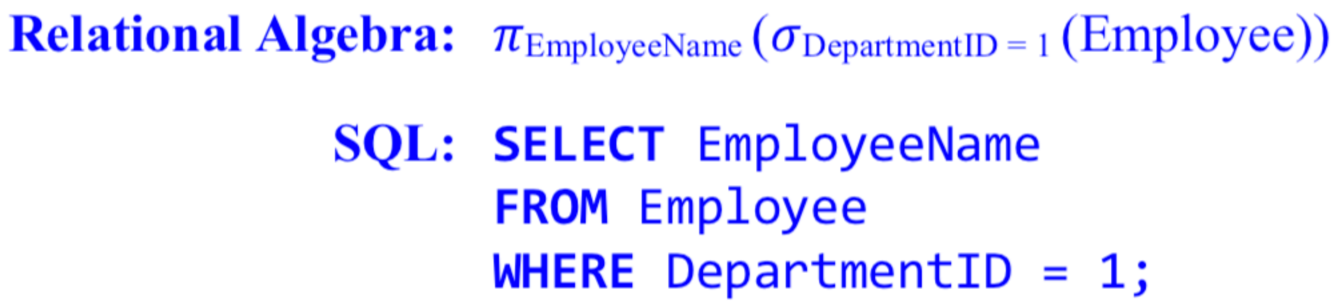
a. Find the names of all employees.



Relational Algebra: $\pi_{\text{EmployeeName}}(\text{Employee})$

SQL:
```
SELECT EmployeeName
FROM Employee;
```

b. Find the names of all employees in department number 1.

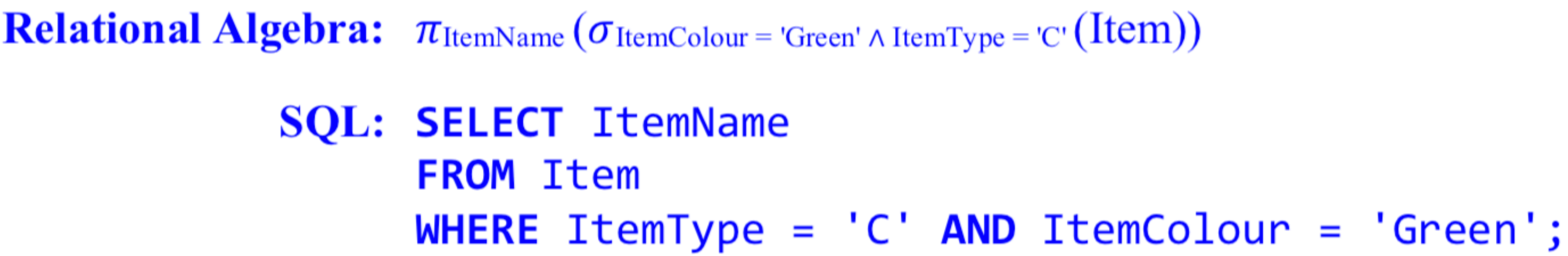


Relational Algebra: $\pi_{\text{EmployeeName}}(\sigma_{\text{DepartmentID} = 1}(\text{Employee}))$

SQL:
```
SELECT EmployeeName
FROM Employee
WHERE DepartmentID = 1;
```

Relational algebra and SQL statements

c. List the names of green items of type C.

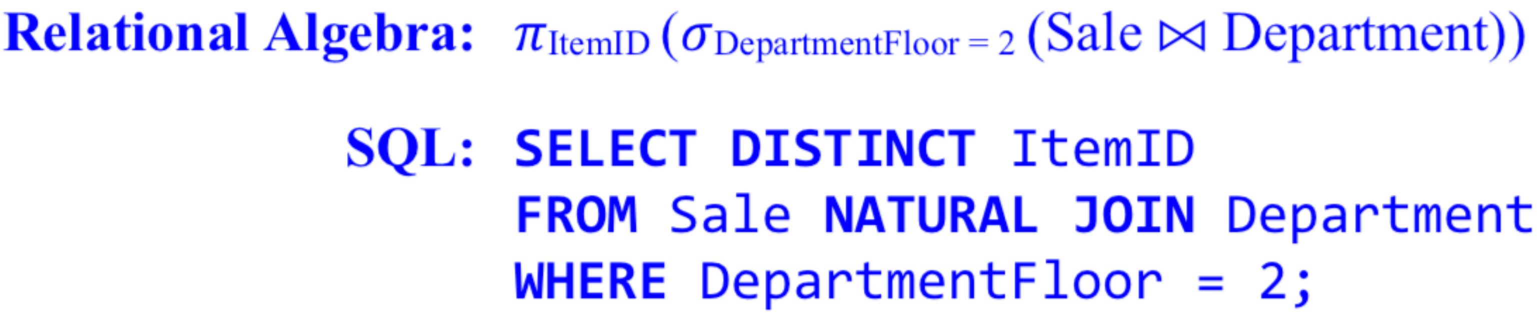


Relational Algebra: $\pi_{\text{ItemName}}(\sigma_{\text{ItemColour} = \text{'Green'} \wedge \text{ItemType} = \text{'C'}}(\text{Item}))$

SQL:

```
SELECT ItemName
FROM Item
WHERE ItemType = 'C' AND ItemColour = 'Green';
```

d. Find the items sold by the departments on the second floor (only show ItemID).



Relational Algebra: $\pi_{\text{ItemID}}(\sigma_{\text{DepartmentFloor} = 2}(\text{Sale} \bowtie \text{Department}))$

SQL:

```
SELECT DISTINCT ItemID
FROM Sale NATURAL JOIN Department
WHERE DepartmentFloor = 2;
```

THE UNIVERSITY OF MELBOURNE

Relational algebra and SQL statements

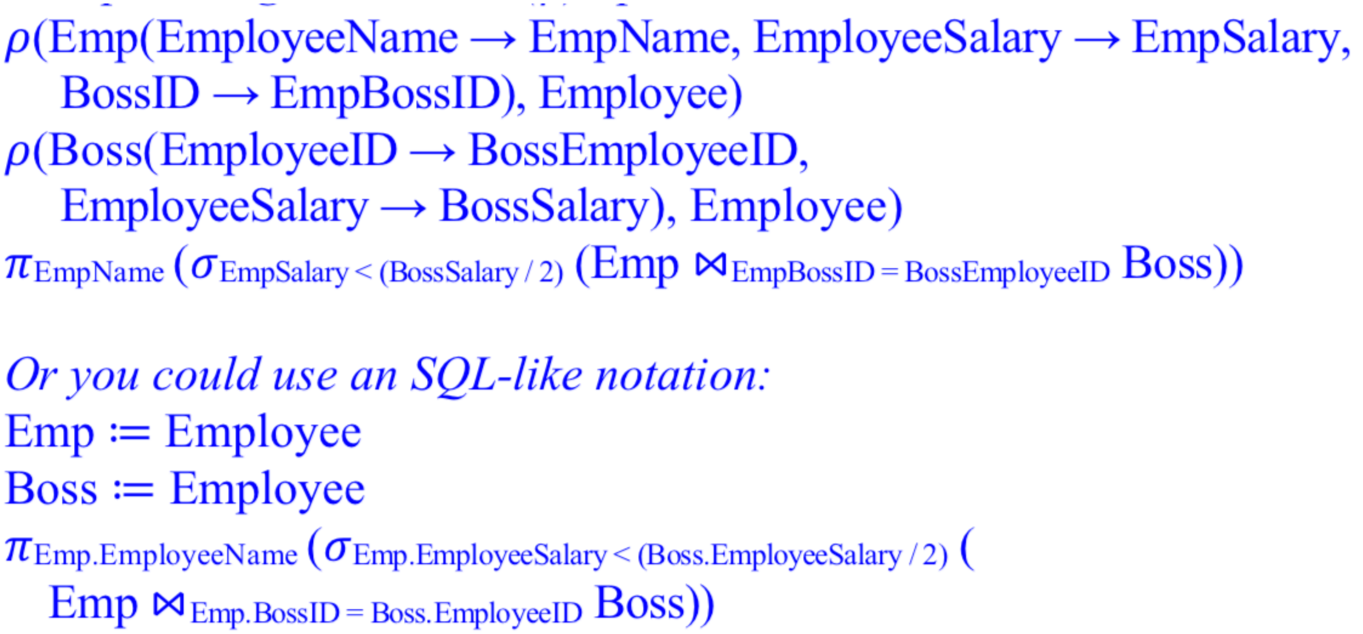
e. Find the names of brown items sold by the Recreation department.

Relational Algebra: $\pi_{\text{ItemName}}(\sigma_{\text{DepartmentName}=\text{'Recreation'} \wedge \text{ItemColour}=\text{'Brown'}}(\text{Item} \bowtie \text{Sale} \bowtie \text{Department}))$

SQL:
```
SELECT ItemName
FROM Item NATURAL JOIN Sale NATURAL JOIN Department
WHERE DepartmentName = 'Recreation'
  AND ItemColour = 'Brown';
```

Relational algebra and SQL statements

f. Find the employees whose salary is less than half that of their managers.



$\rho(\text{Emp}(\text{EmployeeName} \rightarrow \text{EmpName}, \text{EmployeeSalary} \rightarrow \text{EmpSalary},$
$\quad \text{BossID} \rightarrow \text{EmpBossID}), \text{Employee})$

$\rho(\text{Boss}(\text{EmployeeID} \rightarrow \text{BossEmployeeID},$
$\quad \text{EmployeeSalary} \rightarrow \text{BossSalary}), \text{Employee})$

$\pi_{\text{EmpName}}(\sigma_{\text{EmpSalary} < (\text{BossSalary}/2)}(\text{Emp} \bowtie_{\text{EmpBossID} = \text{BossEmployeeID}} \text{Boss}))$

Or you could use an SQL-like notation:

$\text{Emp} := \text{Employee}$

$\text{Boss} := \text{Employee}$

$\pi_{\text{Emp.EmployeeName}}(\sigma_{\text{Emp.EmployeeSalary} < (\text{Boss.EmployeeSalary}/2)}($
$\quad \text{Emp} \bowtie_{\text{Emp.BossID} = \text{Boss.EmployeeID}} \text{Boss}))$

SQL:

```
SELECT Emp.EmployeeName
FROM Employee AS Emp
  INNER JOIN Employee AS Boss
  ON Emp.BossID = Boss.EmployeeID
WHERE Emp.EmployeeSalary < (Boss.EmployeeSalary / 2);
```

Relational algebra and SQL statements

Any questions?