

# Relational Model and Relational Algebra

# Learning Outcomes

- Be able to explain the terminology of relational model.
- Understand how tables are used to represent data.
- Be able to explain properties of database relations.
- Be able to identify candidate, primary, alternate, and foreign keys.
- Understand and be able to explain entity integrity and referential integrity.
- Be able to formulate queries in relational algebra.

# Relational Model

# Relational model

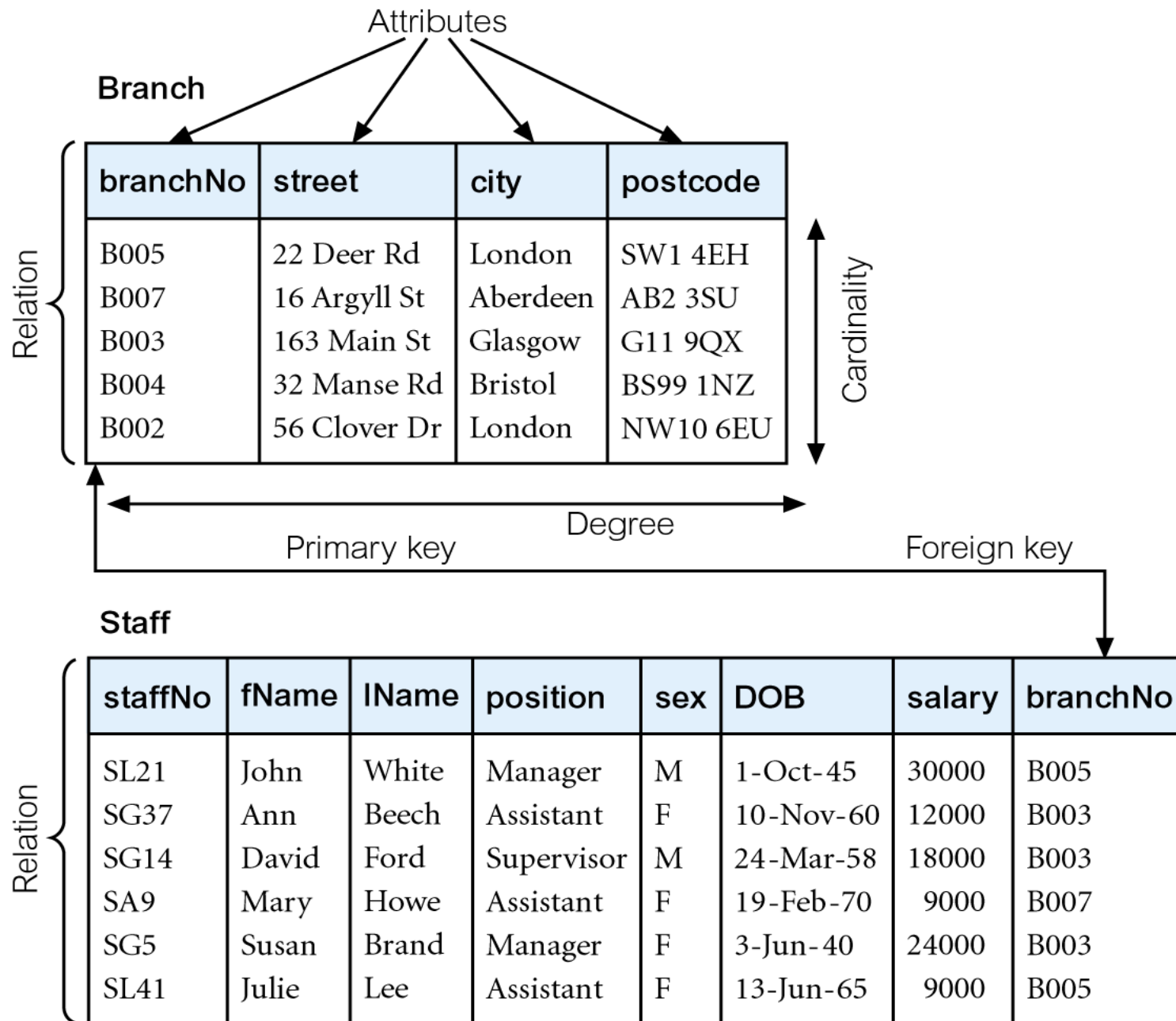
- E. F. Codd proposed relational data model in 1970.
- In the relational model, all data is logically structured within **relations** (tables).
- Each **relation** is made up of **attributes** (columns) of data.
- Each **tuple** (row) contains one value per attribute.

# Terminology

- Relation: A relation is a table with columns and rows.
- Attribute: An attribute is a named column of a relation.
- Domain: the set of allowable values for one or more attributes.

# Terminology

- Tuple: A tuple is row of a relation.
- Degree: the number of attributes in a relation.
- Cardinality: the number of tuples in a relation.
- Relational database: A collection of normalized relations with distinct relation names.



# Examples of Attribute Domains

Attribute	Domain Name	Meaning	Domain Definition
branchNo	BranchNumbers	The set of all possible branch numbers	character: size 4, range B001–B999
street	StreetNames	The set of all street names in Britain	character: size 25
city	CityNames	The set of all city names in Britain	character: size 15
postcode	Postcodes	The set of all postcodes in Britain	character: size 8
sex	Sex	The sex of a person	character: size 1, value M or F
DOB	DatesOfBirth	Possible values of staff birth dates	date, range from 1-Jan-20, format dd-mmm-yy
salary	Salaries	Possible values of staff salaries	monetary: 7 digits, range 6000.00–40000.00



# Properties of relations

- Relation name is distinct from all other relation names in relational schema.
- Each cell of relation contains exactly one atomic (single) value. (*First normal form*)
- Each attribute has a distinct name.
- Values of an attribute are all from the same domain.
- Each tuple is distinct; there are no duplicate tuples.
- Order of attributes has no significance.
- Order of tuples has no significance, theoretically.

# Relational keys

- Candidate Key
  - A set of attributes that uniquely identifies a tuple within a relation.
  - Uniqueness : In each tuple, candidate key uniquely identify that tuple.
  - Irreducibility: No proper subset of the candidate key has the uniqueness property.
- Primary Key
  - Candidate key selected to identify tuples uniquely within relation.
- Foreign Key
  - Attribute, or set of attributes, within one relation that matches candidate key of some other (possibly same) relation.

# Relational Schema (representing a relation)

- A relation can be represented in this format below:  
Relation name(attribute 1, attribute 2, ... attribute n)

Examples:

- Branch(branchNo, street, city, postcode)
- Staff(staffNo, fName, lName, position, sex, DOB, salary, branchNo)

## Client

clientNo	fName	lName	telNo	prefType	maxRent
CR76	John	Kay	0207-774-5632	Flat	425
CR56	Aline	Stewart	0141-848-1825	Flat	350
CR74	Mike	Ritchie	01475-392178	House	750
CR62	Mary	Tregear	01224-196720	Flat	600

- Exercise 1: write down the relational schema for Client relation in *DreamHome* database.

Client(clientNo, fName, lName, telNo, prefType, maxRent)

Extra exercise: write down the relational schema for all the relations in *DreamHome* Database.

# Integrity Constraints

- Null
  - Represents value for an attribute that is currently unknown or not applicable for tuple.
  - Deals with incomplete or exceptional data.
  - Represents the absence of a value and is not the same as *zero* or *spaces*, which are values.

# Integrity Constraints

- Entity Integrity

In a base relation, no attribute of a primary key can be null.

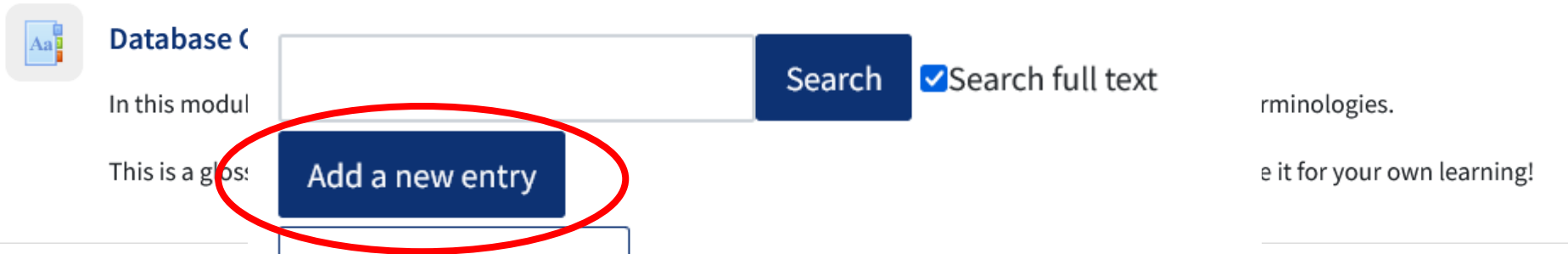
- Referential Integrity

If foreign key exists in a relation, either foreign key value must match a candidate key value of some tuple in its home relation or foreign key value must be null.

- General Constraints: specific constraints on values (e.g. Height (centimetres) must be  $> 0$  and  $< 250$  ).

# Glossary

- A “crowd funded” glossary – every student can add entry and comment on an entry.



The screenshot shows a web interface for a glossary. On the left, there is a sidebar with a document icon and the text 'Database C', 'In this modul', and 'This is a glos:'. The main area contains a search bar with a 'Search' button and a checkbox labeled 'Search full text'. Below the search bar, there is a blue button labeled 'Add a new entry' which is circled in red. To the right of the search bar, there is text that reads 'rminologies.' and 'e it for your own learning!'.

Making good contributions will help yourself and your classmates.

Students with good contributions will get 1 bonus mark for coursework.

# What have we learned?

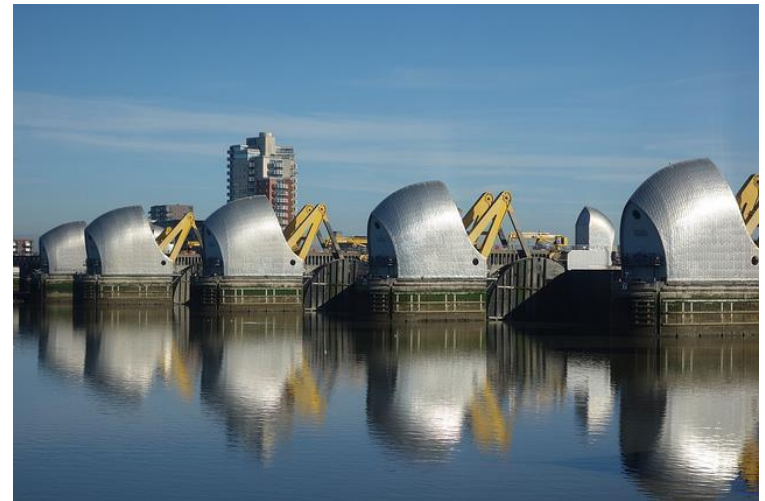
- Aspects of relational model
  - Relation (table), Attribute (column), Tuple (row)
  - Degree (no. columns), Cardinality (no. rows)
  - Domain (set of allowable values for attribute)
- Relational keys
  - Candidate, Primary, Foreign
- Integrity
  - Entity (primary key not null)
  - Referential (foreign key matches a candidate key value in its home relation or is null)
  - General (requirements on values)



# Relational Algebra

# Relational Algebra

- Relational algebra is formal language associated with the relational model.
- Relational algebra operations work on one or more relations to define another relation without changing the original relations.
- Allows expressions to be nested, just as in arithmetic.  
This property is called closure.



“Thames Barrier Closure” by Chris Wheal,  
flickr

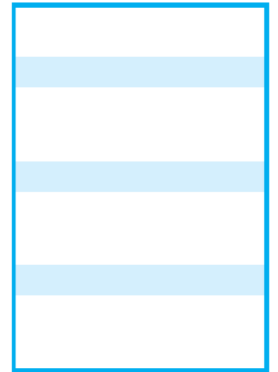
# Relational Algebra

- Basic operations:
  - Selection
  - Projection
  - Cartesian product
  - Join
- Set operations:
  - Union
  - Set difference
  - Intersection

# Selection

- $\sigma_{\text{condition}}(R)$

Works on a single relation R and defines a relation that contains only those tuples (rows) of R that satisfy the specified condition (predicate).



(a) Selection

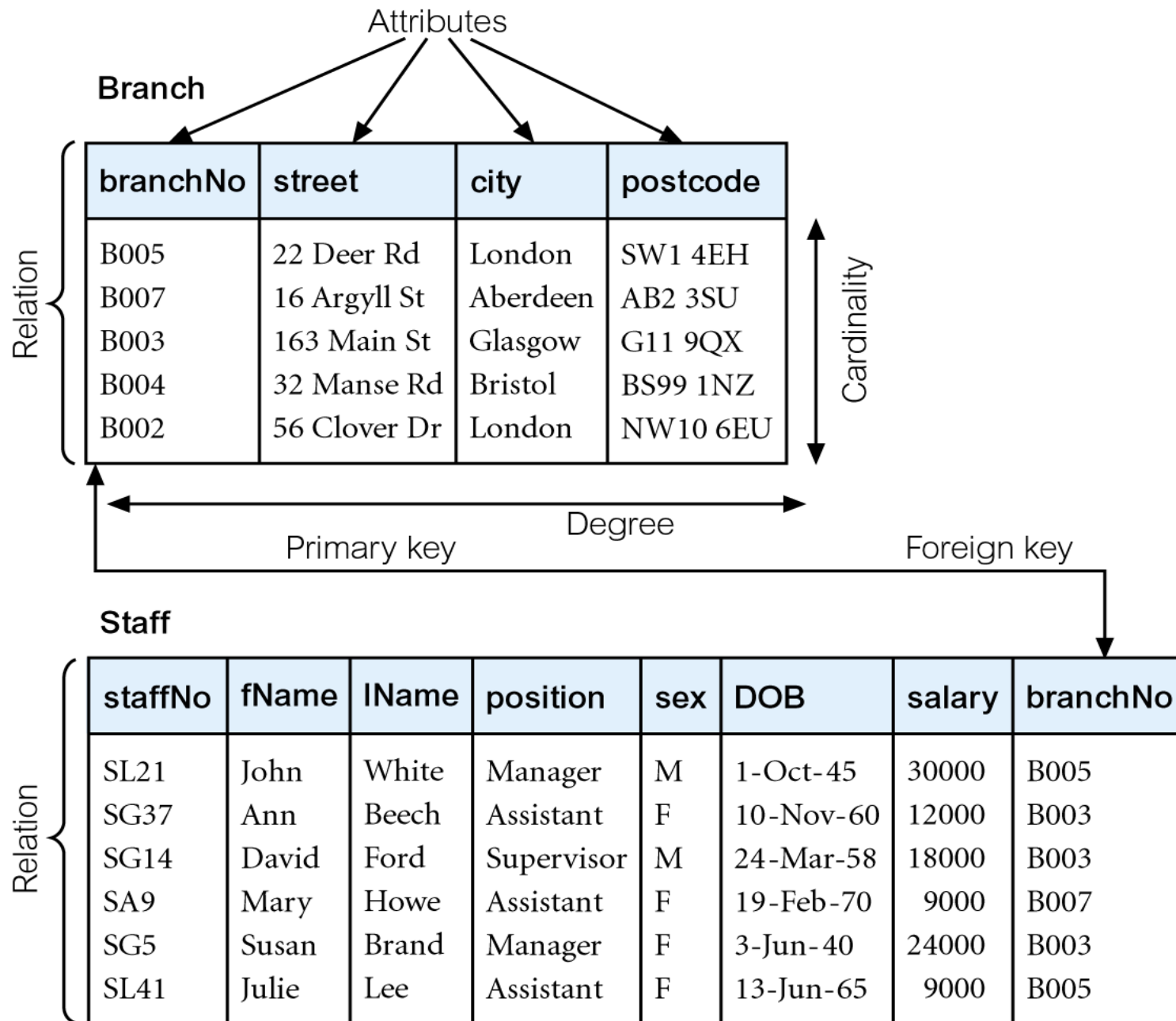
# Selection example

Relation:  
**Students**

Name	ID	Course	Year
Yang Wang	20184123	IoT	2018
Yixin Xi	20184113	Telecomm	2019
Zhou Li	20164888	Telecomm	2019
Hengda Yin	20161333	e-Comm	2019

$$\sigma_{\text{year} = 2019}(\text{Students})$$

Name	ID	Course	Year
Yixin Xi	20184113	Telecomm	2019
Zhou Li	20164888	Telecomm	2019
Hengda Yin	20161333	e-Comm	2019



# Selection

## Example

- List all staff with a salary greater than £10,000.

$\sigma_{\text{salary} > 10000} (\text{Staff})$

- List all female staff with a salary greater than £10,000.

$\sigma_{\text{salary} > 10000 \wedge \text{sex} = \text{"F"}} (\text{Staff})$

## Exercise 2:

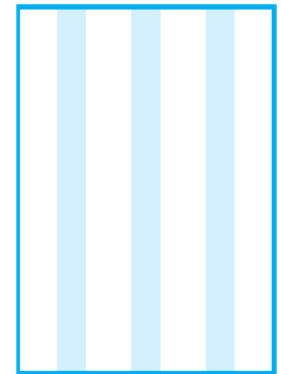
- List all branches in London.

$\sigma_{\text{city} = \text{"London"}} (\text{Branch})$

# Projection

- $\Pi_{\text{col1}, \dots, \text{coln}}(R)$

Works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.



(b) Projection



# Projection example

Relation:  
**Students**

Name	ID	Course	Year
Yang Wang	20184123	IoT	2018
Yixin Xi	20184113	Telecomm	2019
Zhou Li	20164888	Telecomm	2019
Hengda Yin	20161333	e-Comm	2019

$\Pi_{\text{course, year}}(\text{Students})$

Course	Year
IoT	2018
Telecomm	2019
e-Comm	2019

# Projection

## Example

- Produce a list of salaries for all staff, showing only staffNo, fName, lName, and salary details.

$\Pi_{\text{staffNo, fName, lName, salary}}(\text{Staff})$

## Exercise 3

- In DreamHome database, produce a list of all the cities that have branches.

$\Pi_{\text{city}}(\text{Branch})$

# Combine selection and projection

## Example

- List staff number, position and salary of any staff whose salary is greater than £20,000.

*Answer 1:*  $\sigma_{\text{salary} > 20000}(\Pi_{\text{staffNo}, \text{position}, \text{salary}}(\text{Staff}))$

*Answer 2:*  $\Pi_{\text{staffNo}, \text{position}, \text{salary}}(\sigma_{\text{salary} > 20000}(\text{Staff}))$

## Exercise 4

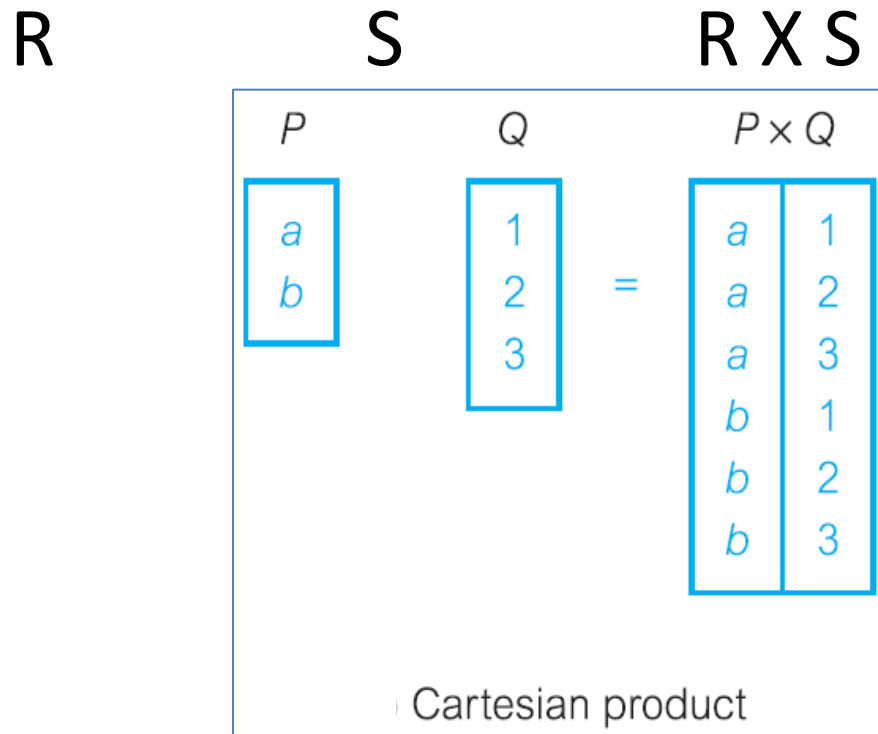
- List the branchNo of all branches in London.  
(Hint: order of operation is important!)

$\Pi_{\text{branchNo}}(\sigma_{\text{city} = \text{"London"}}(\text{Branch}))$

# Cartesian product

- $R \times S$

Defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.



# Cartesian product example

Relation:  
**Students**

Name	ID
Yang Wang	20184123
Yixin Xi	20184113
Zhou Li	20164888

Relation:  
**Courses**

Course	Year
IoT	2018
Telecomm	2019

**Exercise 5:** What is the degree & cardinality of the cartesian product? Students x Courses

Degree 4  
Cardinality 6

Name	ID	Course	Year
Yang Wang	20184123	IoT	2018
Yixin Xi	20184113	IoT	2018
Zhou Li	20164888	IoT	2018
Yang Wang	20184123	Telecomm	2019
Yixin Xi	20184113	Telecomm	2019
Zhou Li	20164888	Telecomm	2019

# Join

- Derivative of Cartesian product, equivalent to performing a Selection operation, using the join condition as the selection formula, over the Cartesian product of the two operand relations.
- Natural Join
- Theta join

# Natural Join

- $R \bowtie S$

- The result of the natural join is the set of all combinations of tuples in  $R$  and  $S$  that are equal on their common attribute names.

$T$		$U$		$T \bowtie U$		
$A$	$B$	$B$	$C$	$A$	$B$	$C$
$a$	1	1	$x$	$a$	1	$x$
$b$	2	1	$y$	$a$	1	$y$
		3	$z$			

Natural join

# Natural join example

Relation:  
**Students**

Name	ID	Course	Year
Yang Wang	20184123	IoT	2018
Yixin Xi	20184113	Telecomm	2019
Zhou Li	20164888	Telecomm	2019
Hengda Yin	20161333	e-Comm	2019

Relation:  
**Years**

Year	Animal
2018	Dog
2019	Pig
2020	Rat
2021	Ox

Students ⋈ Years

Name	ID	Course	Year	Animal
Yang Wang	20184123	IoT	2018	Dog
Yixin Xi	20184113	Telecomm	2019	Pig
Zhou Li	20164888	Telecomm	2019	Pig
Hengda Yin	20161333	e-Comm	2019	Pig



# Natural Join

## Example

- List the client names and comments of all clients who have viewed a property for rent.

$\Pi_{fName, lName, propertyNo, comment} (Client \bowtie Viewing)$

## Exercise 6

- List the staffNo of the staff who works in a branch in London.

$\Pi_{staffNo}(\sigma_{city = \text{"London"}}(Staff \bowtie Branch))$

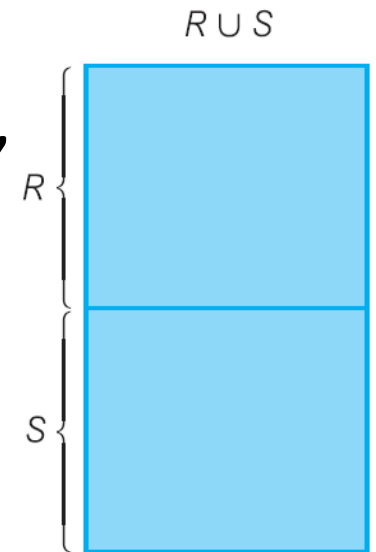
# Theta join

- $R \bowtie_F S$ 
  - Defines a relation that contains tuples satisfying the condition  $F$  from the Cartesian product of  $R$  and  $S$ .
- Can rewrite Theta join using basic Selection and Cartesian product operations.

$$R \bowtie_F S = \sigma_F (R \times S)$$

# Union

- $R \cup S$ 
  - Union of two relations  $R$  and  $S$  defines a relation that contains all the tuples of  $R$ , or  $S$ , or both  $R$  and  $S$ , duplicate tuples being eliminated.
  - $R$  and  $S$  must be union-compatible.
- Union compatible: Same number of attributes and corresponding attributes have the same domain.
- If  $R$  and  $S$  have  $I$  and  $J$  tuples, respectively, union is obtained by concatenating them into one relation with a maximum of  $(I + J)$  tuples.



# Union example

Relation:  
**Students1**

Name	ID
Yang Wang	20184123
Yixin Xi	20184113

Relation:  
**Students2**

Name	ID
Zhou Li	20164888
Hengda Yin	20161333

**Students1  $\cup$  Students2**

Name	ID
Yang Wang	20184123
Yixin Xi	20184113
Zhou Li	20164888
Hengda Yin	20161333

# Union

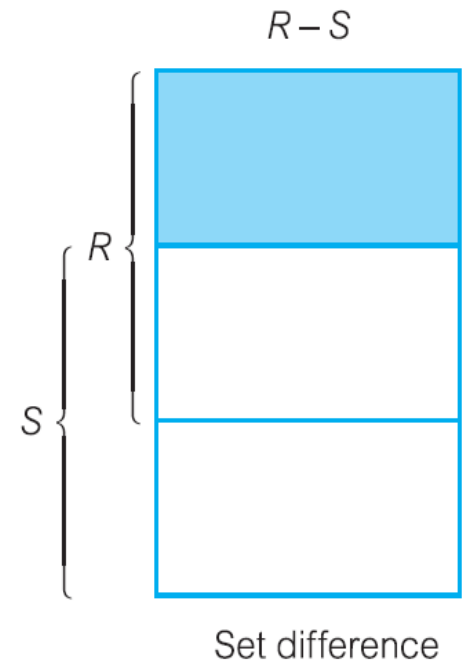
## Exercise 7

- List all cities where there is either a branch office or a property for rent.

$$\Pi_{\text{city}}(\text{Branch}) \cup \Pi_{\text{city}}(\text{PropertyForRent})$$

# Set difference

- $R - S$ 
  - Defines a relation consisting of the tuples that are in relation  $R$ , but not in  $S$ .
  - $R$  and  $S$  must be union-compatible.



# Set Difference example

Relation:

**Students1**

Name	ID
Yang Wang	20184123
Yixin Xi	20184113
Zhou Li	20164888
Hengda Yin	20161333

Relation:

**Students2**

Name	ID
Zhou Li	20164888
Hengda Yin	20161333

Students1 – Students2

Name	ID
Yang Wang	20184123
Yixin Xi	20184113

# Set Difference

## Exercise 8

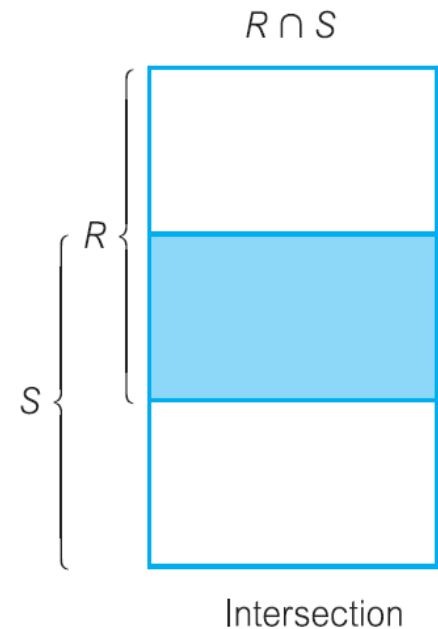
- List all cities where there is a branch office but no properties for rent.

$$\Pi_{\text{city}}(\text{Branch}) - \Pi_{\text{city}}(\text{PropertyForRent})$$



# Intersection

- $R \cap S$ 
  - Defines a relation consisting of the set of all tuples that are in both R and S.
  - R and S must be union-compatible.
- Expressed using basic operations:
$$R \cap S = R - (R - S)$$



# Intersection example

Relation:  
**Students1**

Name	ID
Yang Wang	20184123
Yixin Xi	20184113
Zhou Li	20164888

Relation:  
**Students2**

Name	ID
Hengda Yin	20161333
Yang Wang	20184123
Yixin Xi	20184113

**Students1  $\cap$  Students2**

Name	ID
Yang Wang	20184123
Yixin Xi	20184113

# Intersection

## Exercise 9

- List all cities where there is both a branch office and at least one property for rent.

$$\Pi_{\text{city}}(\text{Branch}) \cap \Pi_{\text{city}}(\text{PropertyForRent})$$

# Relational Algebra Exercise

Given the following relational schema:

Hotel(hotelNo, hotelName, city)

Room(roomNo, hotelNo, type, price)

Booking(hotelNo, guestNo, dateFrom, dateTo, roomNo)

Guest(guestNo, guestName, guestAddress)

Formulate the following queries in relational algebra:

1. List all hotels in Beijing.
2. Give the hotel number of those hotels with a room price greater than £50.
3. List the price and types of all rooms in Happy Hotel.
4. Produce a relation containing all rooms at all hotels.
5. Give all hotel names with a room price above £50.
6. List the guest number of all guests currently staying at the Happy Hotel. (Hint: use `current_date()` for today's date)
7. List guest number of all guests who have ever booked a hotel in Beijing.
8. List guest number and name of those guests who have stayed in executive rooms.

# Sample solution

1.  $\sigma_{\text{city} = \text{"Beijing"}}(\text{Hotel})$
2.  $\Pi_{\text{hotelNo}}(\sigma_{\text{price} > 50}(\text{Room}))$
3.  $\Pi_{\text{price}, \text{type}}(\sigma_{\text{hotelName} = \text{"Happy"}}(\text{Hotel} \bowtie \text{Room}))$
4.  $\text{Hotel} \bowtie \text{Room}$
5.  $\Pi_{\text{hotelName}}(\sigma_{\text{price} > 50}(\text{Hotel} \bowtie \text{Room}))$
6.  $R1 = \sigma_{\text{hotelName} = \text{"Happy"} \wedge \text{dateFrom} < \text{current\_Date}() \wedge \text{dateTo} > \text{current\_Date}}(\text{Hotel} \bowtie \text{Room})$   
Final result:  $\Pi_{\text{guestNo}}(R1)$
7.  $\Pi_{\text{guestNo}}(\sigma_{\text{city} = \text{"Beijing"}}(\text{Booking} \bowtie \text{Hotel}))$
8.  $\Pi_{\text{guestNo}, \text{guestName}}(\sigma_{\text{type} = \text{"Executive"}}(\text{Guest} \bowtie \text{Booking} \bowtie \text{Room}))$

