Database System

北京交通大学软件学院

王方石 教授

E-mail: fshwang@bjtu.edu.cn

Chapter 2-Contents

2.1 Relational Data Model

2.2 Relation Integrity

2.3 Relational Algebra

2.1 Relational Data Model

Relational Model is based on the mathematical concept of a relation, which is physically represented as a table.

- **Single Data Structure---relation**
  - **relation** – It is a 2D table with columns and rows.
  - All entities and relationship among the entities are expressed by relations.
  - All operands and results of relational operations are relations.
- **Data Manipulation---relational operations**

  Query, insert, delete, update
- **integrity rules（=correctness + validity+ consistency）**
  - Entity integrity
  - referential integrity
  - user-defined integrity(enterprise constraints)

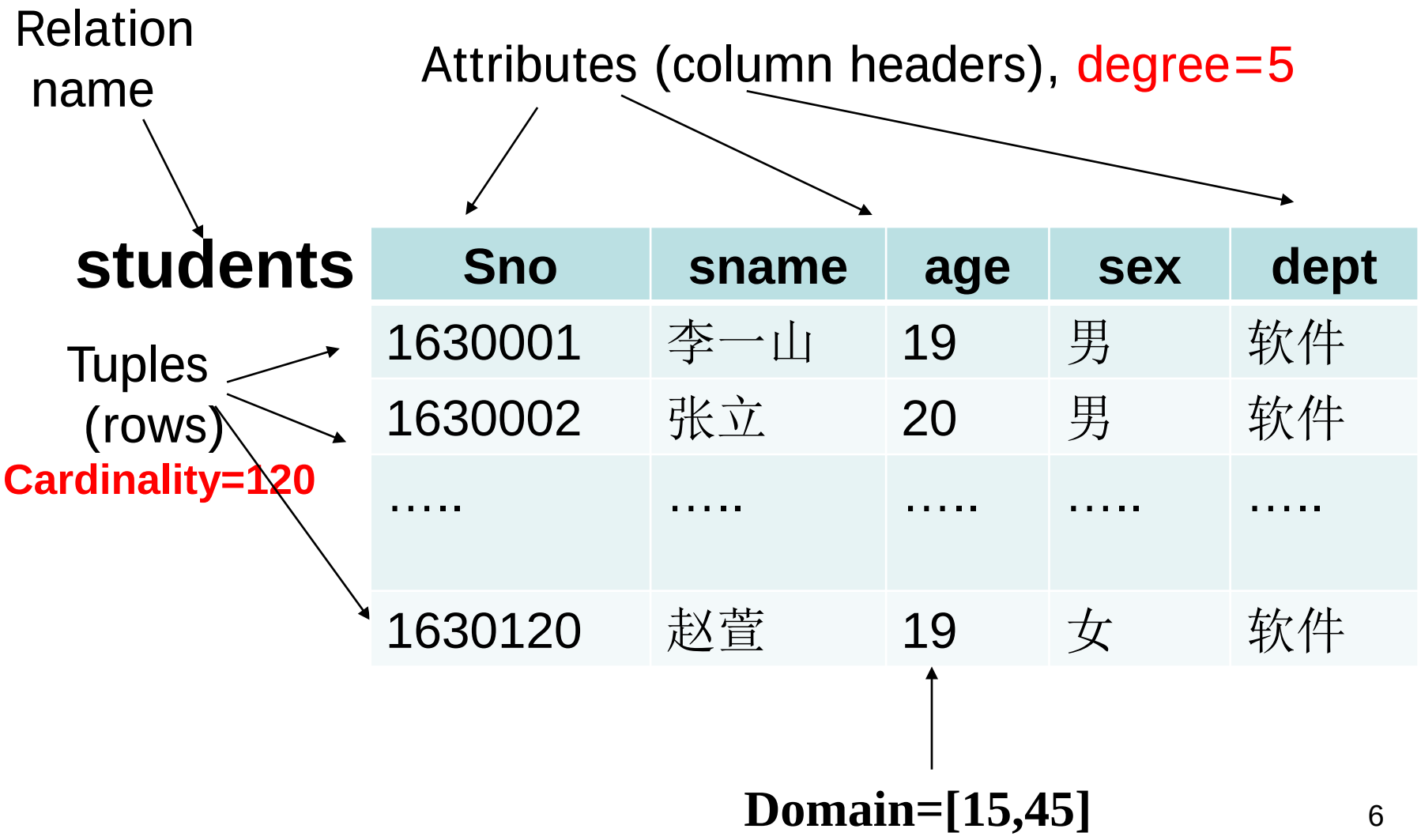
2.1.1 Terminology

- **A relation is a table with columns and rows.**
  - **Only applies to logical structure of the database,**
  - **not the physical structure， which we can implement using a variety of storage structures (such as a heap file or hash file).**
- **An attribute is a named column of a relation.**
- **A domain is the set of allowable values for one or more attributes.**

Relational Model Terminology

- A **tuple** is a row of a relation.
- The **degree** (维、度) is the number of attributes in a relation.
- The **cardinality**（基数） is the number of tuples in a relation.
- **Relational Database** is a collection of normalized **relations** with distinct relation names.

A Relation is a Table



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

Alternative Terminology for Relational Model

Table 2.1 Alternative terminology for relational model terms.

Formal terms	Alternative 1	Alternative 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

2.1.2 Database Relations

- **Relation schema（关系模式）**
  - **A named relation defined by a set of attribute and domain name pairs.**
  - **Example: students(sno, sname, DOB) or** students(sno:StudentNumbers, sname: string, DOB: DateOfBirth)

Attribute	Domain name	Meaning	Domain definition
sno	**StudentNumbers**	The set of all possible students numbers	Character: size 7,range 1900000-1900999
DOB	DateOfBirth	Possible values of student birth dates	Date,range from 1-Jan-2000, format dd-mmm-yyyy

2.1.2 Database Relations

- **Relation schema is type, relation is value.**
- **Example:**
  - **students(sno, sname, DOB) is Relation schema**
  - **The following table is relation.**

Sno	sname	DOB
1630001	李一山	12-Jan-1997
1630002	张立	20-Feb-1998
…..	…..	…..
1630120	赵萱	19-Mar-1999

2.1.2 Database Relations

- **Relational database schema**

  （关系数据库模式）

  – **A set of all relation schemas, each with a distinct name.**

- **Relational database schema is type（型）, relational database is value（值）.**

Properties of Relations

(1) The Relation has a name that is distinct from all other relation names in the relational schema.

(2) Each cell of relation contains exactly one atomic (single) value.

(3) Each attribute has a distinct name in one relation.

(4) Values of an attribute are all from the same domain.

(5) Each tuple is distinct; there are no duplicate tuples.

(6)The order of attributes has no significance.

Properties of Relations (cont.)

(7)The order of tuples has no significance, theoretically. (However, in practice, the order may affect the efficiency of accessing records. E.g. search)

S

sno	sn	age	sex
s1	John	19	M
s2	Kate	17	F
s3	Alice	18	F
s4	Mary	21	M

SC1

sno	cno	G
s1	c1	79
s1	c3	85
s2	c1	60
s2	c2	83
s2	c3	90
s3	c1	95
s3	c2	80
s4	c1	75
s4	c2	85

SC2

sno	cno	G
s1	c1	79
s3	c1	95
s2	c1	60
s4	c2	85
s2	c2	83
s3	c2	80
s2	c3	90
s4	c1	75
s1	c3	85

Estates Agency-Dream of Home

Type(型)--stable

Branch

branchNo	street	city	Postcode

Branch （ branchNo, street, city, Postcode）

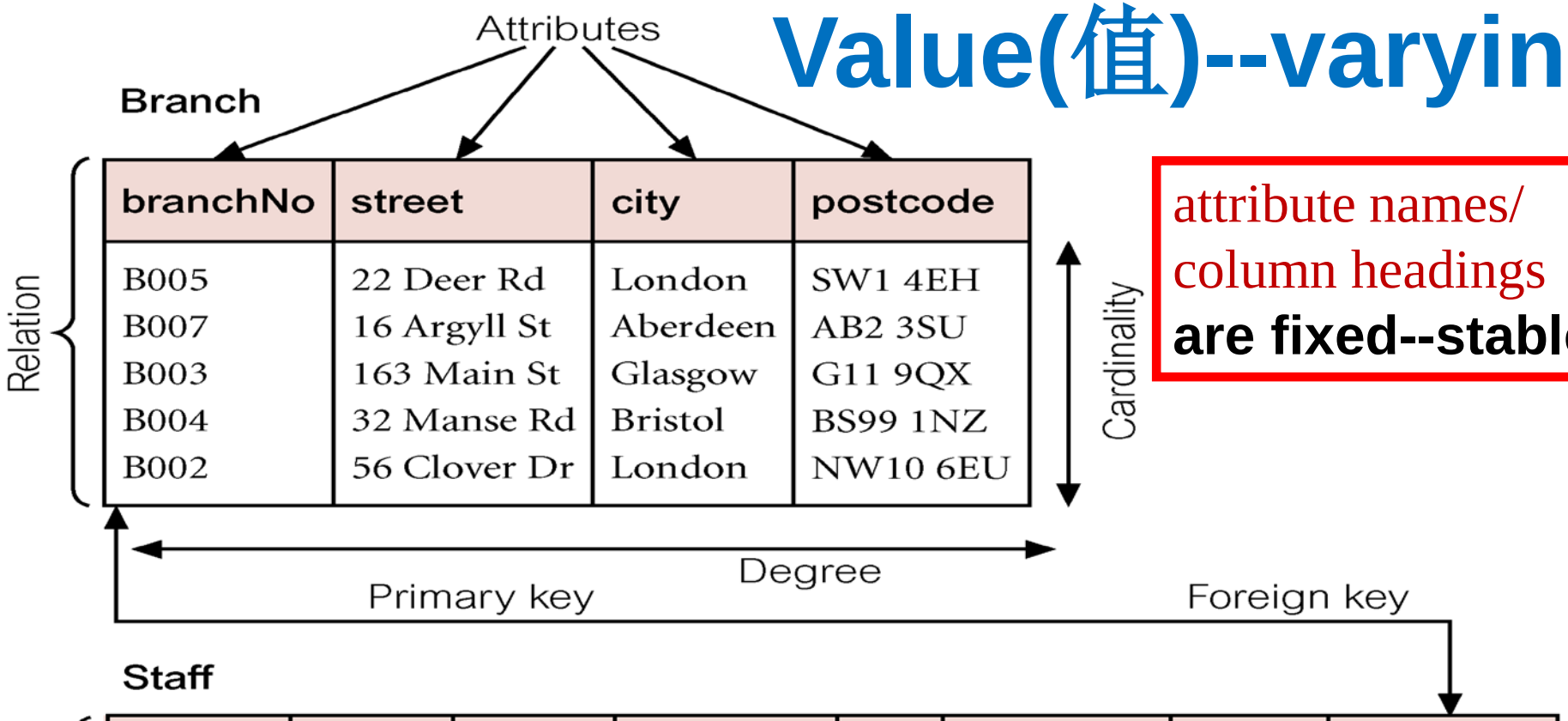
Staff

staffNo	fName	lName	position	sex	DOB	Salary	branchNo

Staff（ staffNo, fName, lName, position, sex, DOB, Salary, branchNo）

attribute names / column headings **are fixed—stable.**

Value(值)--varying



attribute names/
column headings
are fixed--stable

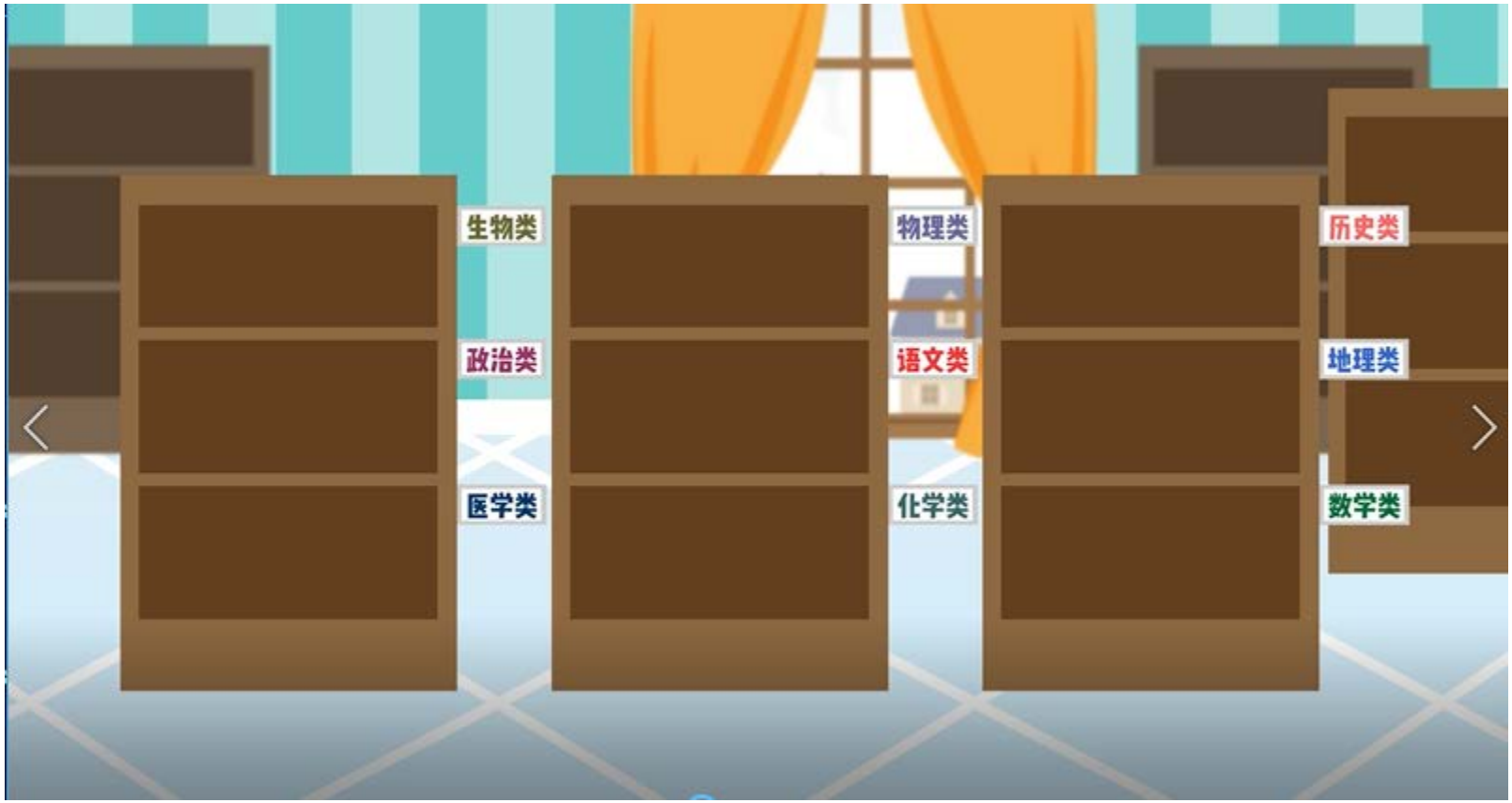
Staff

Relation

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

An empty book-shelf： a **relation schema**, label=**attribute name**

A library room with several empty book-shelves:
a **database schema**



A shelf with books： **a relation** ,

A library room with the above book-shelves：a **database**

2.1.3 Relational Keys（关键字）

- **Superkey（超键）**
  - **An attribute, or a set of attributes, that uniquely identifies a tuple within a relation.**

S(sno, sname ,age, sex)

(sno)
(sno, sname)
(sno, sname,age)
(sno, sname,sex)
(sno, age)
(sno, sex)
(sno, age,sex)
(sno, sname ,age, sex)

Branch (branchNo, street, city, Postcode)

(branchNo)
(branchNo, street)
(branchNo, street, city)
(branchNo, street, Postcode)
(branchNo , city)
(branchNo, Postcode)
(branchNo, city, Postcode)
(branchNo, sname, city, Postcode)

2.1.3 Relational Keys

◆Candidate Key（候选键）

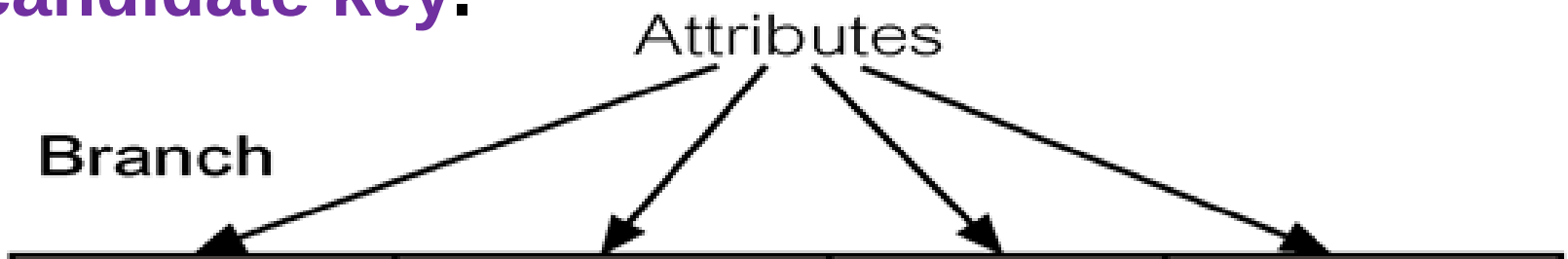
- ➢**Superkey (K) such that no proper subset is a superkey within the relation. e.g. {a, b} :{ } {a} {b}**
- ➢**Or A superkey that contains only the minimum number of columns necessary for unique identification.**

◆A candidate key, *K*, for a relation *R* has **two properties**:

- ➢ **uniqueness** – in each tuple of *R*, the values of *K* uniquely identify that tuple;
- ➢ **irreducibility** – no proper subset of *K* has the uniqueness property.

Relational Keys

Only by using the semantic information can we be certain that an attribute combination is a candidate key.



branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Relational Keys

composite key

Viewing (**clientNo, propertyNo**, viewDate, comment)

If a client may view a property more than once, then we could add **viewDate** to the composite key. However, we assume that this is not necessary.

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Example of Candidate Key

- S (sno, sname ,age, sex) CK=(sno)
- C (cno, cname, credit) CK=(cno)
- SC (sno, cno, grade) CK=(sno，cno) composite key
- SS (sno, sname, cno, grade)

CK1=(sno,cno),
CK2=(sname,cno) } Two composite keys

Note:

There may be several candidate keys for a relation.

Relational Keys (cont.)

◆Primary Key (PK，主键、主码)

- Candidate key that is selected to identify tuples (records) uniquely within the relation (table).

◆Alternate Keys (AK，备用键)

- Candidate keys that are not selected to be primary key.

e.g. Branch (branchNo, street, city, Postcode)

PK=(branchNo), AK=(street, city)

◆Primary-key attribute(主属性)

- An attribute included in any candidate key.

◆Foreign Key（外键）

- Attribute, or set of attributes, within one relation that matches candidate key of some (possibly same) relation.

e.g. Staff (staffNo, fName, lName, position, branchNo, leaderNo)

2.2 Relational Integrity

◆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.

e.g. sc (sno, cno, grade)

grade=0 and grade is null

Null

sc (sno, cno, grade)

SC

sno	cno	grade
S1	C1	78
S1	C2	86
S2	C2	0
S2	C3	76
S3	C3	
S4	C4	80

null

not spaces

Relation Integrity

(1) **Entity Integrity**

(2) **Referential Integrity**

(3) **Enterprise Constraints**

2.2.1 Entity Integrity

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

i.e. A primary-key attribute can not be a null.

e.g. s (sno, sname, age, sex)

c (cno,cname,credits)

sc (sno, cno, grade)

2.2.2 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 wholly null.

- The name of foreign key need not be the same with that of its referencing primary key.

e.g. S (sno, sname, age, sex)

sc (snum , cno, Grade) sno can not be null

Branch(branchNo,street,city,postcode)

Staff (staffNo, fName, lName, position, branchNo, leaderNo)

Either branchNo or leaderNo may be null.

B

branchNo	street	city	postcode
B003	18 Dale Rd	Glasgow	G12
B005	22 Deer Rd	London	SW14EH
B007	16 Argyll St	Aberdeen	AB23SU

S

staffNo	….	branchNo	leaderNo
SL41	….	B005	SL31
SL21	….	B025	SL41
SA9	….		SL41
SG14	….	B003	
SG37	….	B003	SG14

2.2.2 Referential Integrity

s (sno, sname, age, sex) --Referenced Relation

sc (sno, cno, grade) --Referencing Relation

s

sno	sname	age	sex
S1	LI	17	M
S2	SHI	19	F
S3	LIU	21	F
S4	CHEN	20	M

sc

sno	cno	grade
S1	C1	78
S1	C2	86
S2	C2	77
	C3	76
S3	C3	89
S5	C4	80

No !

2.2.2 Referential Integrity

e.g c (cno,cname,credits)

sc (sno, cno, grade)

C

cno	cname	credits
C1	MATHS	4
C2	PHYSICS	3
C3	DB	2
C4	OS	2

SC

Sno	Cno	G
S1	C1	78
S1	C2	86
S2		77
S2	C3	76
S3	C3	89
S4	C5	80

No!

2.2.3 Enterprise Constraints (User-defined Integrity)

– **Additional rules specified by users or database administrators of a database that define or constrain some aspect of the enterprise..**

e.g. The salary of a full professor can not be less than that of a associate professor.

2.2.3 Enterprise Constraints

Teacher

Tno	Tname	department	Title	salary
T1	Wang	SE	Prof	9000
T2	Ding	CS	Assistant Prof.	7000
T3	Zhang	EE	Associate Prof.	**9500**
T4	Tian	EE	Associate Prof.	8500

2.3 Relational Algebra

- **Relational algebra operations work on one or more relations to define another relation without changing the original relations.**

- **Both operands and results are relations, so output from one operation can become input to another operation.**

2.3 Relational Algebra (cont.)

- **Five basic operations in relational algebra: Selection, Projection, Union, Cartesian product, and Set Difference.**
- **These perform most of the data retrieval operations needed.**
- **Also have Join, Intersection, and Division operations, which can be expressed in terms of 5 basic operations.**

2.3 Relational Algebra (cont.)

operator		semantic
Set operators	∪	union
	∩	intersection
	−	difference
	×	Cartesian product
Special relation operators	σ	Selection
	Π	projection
	⋈	join
	÷	division

2.3.1 unary Operations

(1)Selection (or Restriction)

$\sigma_{predicate}(R)$ 【Sigma】，不是 δ【Delta】

The Selection operation works on a single relation R and defines a relation that contains only those tuples (rows) of R that satisfy the specified condition (*predicate*).

Example - Selection (or Restriction)

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

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

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24- Mar-58	18000	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003

Example: More complex predicates can be generated using the **logical operators** ∧ (AND), ∨ (OR) and ~ (NOT).

$$\sigma_{A2>4 \wedge A1='a'}(R)$$

R

A_1	A_2	A_3
a	7	e
a	2	f
a	5	g
b	4	h

result

A_1	A_2	A_3
a	**7**	**e**
a	**5**	**g**

unary Operations

(2) Projection

$\Pi_{col1, \ldots, coln}(R)$

The Projection operation 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.

Example - Projection

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

Π **staffNo, fName, lName, salary (Staff)**

staffNo	fName	lName	salary
SL21	John	White	30000
SG37	Ann	Beech	12000
SG14	David	Ford	18000
SA9	Mary	Howe	9000
SG5	Susan	Brand	24000
SL41	Julie	Lee	9000

e.g. t —-- a tuple

$$\pi_{3,1}(R)=\{t|t=<t_3,t_1> \wedge <t_3,t_1> \in R\}$$

Or $\pi_{A3,A1}(R)=\{t|t=<t_3,t_1> \wedge <t_3,t_1> \in R\}$

R

A_1	A_2	A_3
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_2

结果:

A_3	A_1
c_1	a_1
c_2	a_2

union compatible

Two tables must be *union compatible*.

- **The numbers of columns in two tables should be the same.**
- **The corresponding columns should get values from the same domain.**

S1 (sno: **char(4)**, sname:char (10), age: smallint)
S2 (sno: **char(5)**, sname:char (10), age: smallint)

No!

S1 (sno: char(4), sname:char (10), age: smallint, **sex: char**)
S2 (sno: char(4), sname:char (10), age: smallint)

No!

S1 (sno: char(4), sname:char (10), age: smallint)
S2 (snum: char(4), name:char (10), s_age: smallint)

Yes!

e. g.

R

A	B	C
1	2	3
4	5	6
7	8	9

S

A	B	C
7	8	9
4	5	6
5	1	12

$R \cup S$

A	B	C
1	2	3
4	5	6
7	8	9
5	1	12

union-compatible

- Note that attributes names are not used in defining union-compatibility.
- In some cases, the Projection operation may be used to make two relations union-compatible.

Branch (branchNo, street, **city**, postcode)
Property (propertyNo, street, **city**, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

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

$$\Pi_{city}(Branch) \cup \Pi_{city}(Property)$$

Binary operations

2.3.2 Set Operation

(1) Union

◆**R ∪ S**

- **The 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.**

◆ **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.**

(2) Set Difference

- **R – S**
  - **The Set difference operation defines a relation consisting of the tuples that are in relation R, but not in S.**
  - **R and S must be union-compatible.**

$$R_1 - R_2 = \{t \mid t \in R_1 \wedge t \notin R_2\}$$

e.g

R

A	B	C
1	2	3
4	5	6
7	8	9

S

A	B	C
7	8	9
4	5	6
5	1	12

R-S

A	B	C
1	2	3

Branch (branchNo, street, **city**, postcode)
Property (propertyNo, street, **city**, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

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

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

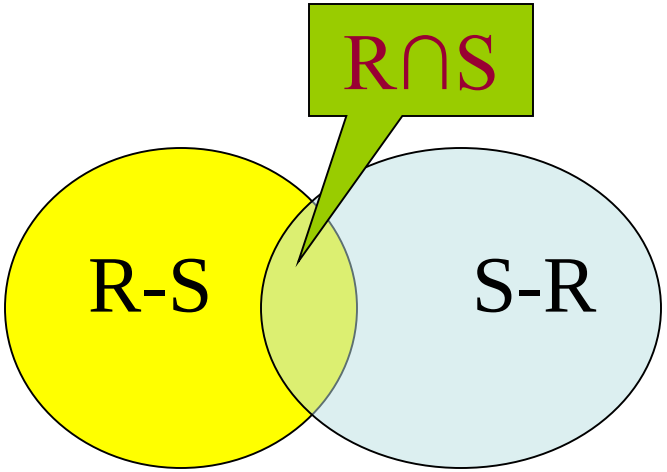
(3) Intersection

- **R ∩ S**
  - **The Intersection operation defines a relation consisting of the set of all tuples that are in both R and S.**
  - **R and S must be union-compatible.**

- **Note that we can express the Intersection operation in terms of the Set difference operation :**

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

Intersection



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

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

e. g.

R

A	B	C
1	2	3
4	5	6
7	8	9

S

A	B	C
7	8	9
4	5	6
5	1	12

R ∩ S

A	B	C
4	5	6
7	8	9

Branch (branchNo, street, **city**, postcode)
Property (propertyNo, street, **city**, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

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

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

(4) Cartesian product

◆R × S

The Cartesian product operation defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.

(4) Cartesian product

- **The Cartesian product operation multiplies two relations to define another relation consisting of all possible pairs of tuples from the two relations.**
- **Therefore, if one relation has *I* tuples and *N* attributes and the other has *J* tuples and *M* attributes, the Cartesian product relation will contain (*I* * *J*) tuples with (*N* + *M*) attributes.**
- **It is possible that the two relations may have attributes with the same name. In this case, the attribute names are prefixed with the relation name to maintain the uniqueness of attribute names within a relation.**

e. g.

$$D = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \times \begin{bmatrix} a \\ b \end{bmatrix} \Rightarrow D = \begin{bmatrix} 1 & a \\ 1 & b \\ 2 & a \\ 2 & b \\ 3 & a \\ 3 & b \end{bmatrix}$$

e.g. R×S

R

A	B	C
a	b	c
b	c	e
e	d	c

S

C	D
c	d
e	f

R×S

A	B	R.C	S.C	D
a	b	c	c	d
a	b	c	e	f
b	c	e	c	d
b	c	e	e	f
e	d	c	c	d
e	d	c	e	f

R×S结果中元组的个数（即基数, 行数Cardinality）= R和S中行数之积

属性的个数(即元数, 列数Degree)
=R和S中列数之和

2.3.3 Join Operations

- **The Join operation, which combines two relations to form a new relation, is one of the essential operations in the relational algebra.**
- **Join is a derivative of Cartesian product.**
- **It is equivalent to performing a Selection, using join predicate as selection formula, over Cartesian product of the two operand relations.**
- **Join is one of the most difficult operations to implement efficiently in an RDBMS and one reason why RDBMSs have intrinsic performance problems.**

Join Operations

- **There are various forms of join operation, each with subtle differences, some more useful than others:**
  - **Theta join (θ-join)**
  - **Equi_join (a particular type of Theta join)**
  - **Natural join**
  - **Outer join**

    (Right Outer join , Left Outer join)
  - **Semijoin**

(1) Theta join (θ-join)

- $R \bowtie_F S$
  - It defines a relation that contains tuples satisfying the predicate F from the Cartesian product of R and S.
  - The predicate F is of the form $R.a_i \ \theta \ S.b_i$ where θ may be one of the comparison operators (<, ≤, >, ≥, =, ≠).

Theta join (θ-join)

- **We can rewrite Theta join using basic Selection and Cartesian product operations.**

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

- **As with Cartesian product, the Degree of a Theta join is the sum of degrees of the operand relations R and S.**
- **In the case where the predicate F contains only equality (=), the term *Equijoin* is used instead.**

e.g.

求：$R1 \underset{2>2}{\bowtie} R2$

R_1

A_1	A_2	A_3
a	4	e
b	2	f
c	5	g
d	1	h

R_2

A_1	A_2	A_3
b	2	e
c	3	f
b	5	h

result:

$R_1.A_1$	$R_1.A_2$	$R_1.A_3$	$R_2.A_1$	$R_2.A_2$	$R_2.A_3$
a	4	e	b	2	e
a	4	e	c	3	f
c	5	g	b	2	e
c	5	g	c	3	f

θ- join (one or more condition)

R

A	B	C
1	2	3
4	5	6
7	8	9

S

D	E
3	1
6	2

$R \bowtie_{2 < 1} S$

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

$R \bowtie_{2<1 \wedge 1 \geq 2} S$

A	B	C	D	E
1	2	3	3	1
4	5	6	6	2

R

A	B	C
1	2	3
4	15	6
7	8	9

S

D	E	F
3	1	4
6	2	13
5	1	12

$R \bowtie S$

$1>2 \wedge 2<3$

A	B	C	D	E	F
7	8	9	6	2	13
7	8	9	5	1	12

Equijoin

R

A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

S

B	C	D
b	c	d
b	c	e
a	d	b

$R \bowtie_{3=2} S$

A	R.B	R.C	S.B	S.C	D
a	b	c	b	c	d
a	b	c	b	c	e
d	b	c	b	c	d
d	b	c	b	c	e
c	a	d	a	d	b

$R \bowtie_{2=1 \wedge 3=2} S$

$R \bowtie_{1=1} S$

A	R.B	R.C	S.B	S.C	D
b	b	f	b	c	d
b	b	f	b	c	e
a	b	c	a	d	b

Example of Equijoin

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

Viewing (clientNo, propertyNo, viewDate, comment)

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

$\sigma_{\text{Client.clientNo = Viewing.clientNo}}$ **degree=6**

$((\Pi_{\text{clientNo, fName, lName}} (\text{Client})) \times$

$(\Pi_{\text{clientNo, propertyNo, comment}} (\text{Viewing})))$

Use equijoin，degree=6

$(\Pi_{\text{clientNo, fName, lName}} (\text{Client})) \bowtie_{\text{Client.clientNo = Viewing.clientNo}}$

$(\Pi_{\text{clientNo, propertyNo, comment}} (\text{Viewing}))$

(2) Natural join

◆R ⋈ S

- The Natural join is an **Equijoin** of the two relations R and S over **all common attributes** ***x***.
- One occurrence of each **common attribute is eliminated** from the result.

- The Natural join operation performs an Equijoin **over all the attributes** in the two relations that **have the same name**.
- The **degree** of a Natural join is the **sum of the degrees** of the relations R and S less the number of attributes in ***x***.

等值连接与自然连接的区别

R

A	B	C
a1	b1	5
a1	b2	6
a2	b3	8
a2	b4	12

S

B	E
b1	3
b2	7
b3	10
b3	2
b5	2

Equi_join（degree=5）

A	R.B	C	S.B	E
a1	b1	5	b1	3
a1	b2	6	b2	7
a2	b3	8	b3	10
a2	b3	8	b3	2

Natural join（degree=4）

A	B	C	E
a1	b1	5	3
a1	b2	6	7
a2	b3	8	10
a2	b3	8	2

Example of Natural join

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

Viewing (clientNo, propertyNo, viewDate, comment)

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

Use equijoin，degree=6

$(\Pi_{\text{clientNo, fName, lName}}(\text{Client})) \bowtie_{\text{Client.clientNo = Viewing.clientNo}} (\Pi_{\text{clientNo, propertyNo, comment}}(\text{Viewing}))$

Use natural join，degree=5

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

Schemas for Results
(关系操作结果的关系模式)

- **Union, intersection,** and **difference**: the schemas of the two operands must be **union-compatible**, so use that schema for the result.
- **Selection**: schema of the result is the same as the schema of the operand.
- **Projection**: the list of attributes tells us the schema.

Schemas for Results --- (2)

- **Product**: schema is the attributes of both relations.
  - Use R.*A*, etc., to distinguish two attributes named *A*.
- **Theta-join**: same as product.
- **Natural join**: union of the attributes of the two relations.

(3) Outer join

- Often in joining two relations, a tuple in one relation does not have a matching tuple in the other relation.
- In other words, there is no matching value in the join attributes.
- We may want tuples from one of the relations to appear in the result even when there are no matching values in the other relation.
- This may be accomplished using the **Outer join**.
- **To display rows in the result that do not have matching values in the join column, use Outer join.**

(3) Outer join

- The Outer join is becoming more widely available in relational systems and is a specified operator in the SQL standard (see Chapter 3).
- The **advantage of an Outer join** is that information is preserved, that is, the Outer join preserves tuples that would have been lost by other types of join.

(3) Outer join

- **悬浮元组（Dangling tuple）**

  两个关系***R***和***S***在做自然连接时，关系***R***中某些元组有可能在***S***中不存在公共属性上值相等的元组，从而造成***R***中这些元组在操作时被舍弃了，这些被舍弃的元组称为**悬浮元组**。

- 如果把悬浮元组也保存在结果关系中，而在其他属性上填空值**(Null)**，就叫做**外连接。**
  - **左外连接(LEFT OUTER JOIN或LEFT JOIN)**，只保留左边关系***R***中的悬浮元组
  - **右外连接(RIGHT OUTER JOIN或RIGHT JOIN)**，只保留右边关系***S***中的悬浮元组

Left outer join

R ⟕ S

- **The Left outer join is join in which tuples from R that do not have matching values in the common columns of S are also included in result relation.**
- **Left Outer join** keeps every tuple in **the left-hand** relation in the result.
- **Missing values in the second relation are set to null.**

Example for Left (natural) outer join

R

A	B	C
a	b	c
b	b	f
c	a	d

S

B	C	D
b	c	d
b	c	e
a	d	b
e	f	g

R ⋈ S 自然连接

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b

R ⟗ S 全自然外连接

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null
null	e	f	g

R ⟕ S 左自然外连接

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null

Right outer join

$R \bowtie S$

- **The Right outer join is join in which tuples from S that do not have matching values in common columns of R are also included in result relation.**
- **Right Outer join** keeps every tuple in the **right-hand** relation in the result.
- **Missing values in the first relation are set to null.**

Example for Right (natural) outer join

R

A	B	C
a	b	c
b	b	f
c	a	d

S

B	C	D
b	c	d
b	c	e
a	d	b
e	f	g

R ⋈ S 自然连接

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b

R ⟗ S 全自然外连接

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null
null	e	f	g

R ⟖ S 右自然外连接

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
null	e	f	g

(4) Semijoin

$R \triangleright_F S$ $R \ltimes_F S$

- The **Semijoin** operation defines a relation that contains the tuples of **R** that participate in the join of R with S.
- The Semijoin operation performs a join of the two relations and then projects over the attributes of the **first operand**.

(4) Semijoin

- One advantage of a Semijoin is that it decreases the number of tuples that need to be handled to form the join.
- It is particularly useful for computing joins in distributed systems.
- We can rewrite the Semijoin using the Projection and Join operations:

$$R \triangleright_F S = \Pi_A(R \bowtie_F S)$$

where A is the set of all attributes for R.

- This is actually a Semi-Theta join (半θ连接).
- There are variants for Semi-Equijoin and Semi-Natural join.

Example for Semijoin

R

A	B	C
a	b	c
b	b	f
c	a	d

S

B	C	D
b	c	d
b	c	e
a	d	b
e	f	g

R ⋈ S 自然连接

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b

R ▷ $_{R.B=S.B \wedge R.C=S.C}$ S

A	B	C
a	b	c
c	a	d

半 θ 连接

2.3.4 Division Operation

$r \div s$

- Suited to queries that **include** the phrase "for all".
- Let *r* and *s* be relations on schemas R and S respectively where
  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

  The result of r ÷ s is a relation on schema

  $R - S = (A_1, \ldots, A_m)$

  Let X= $(A_1, \ldots, A_m)$, Y= $(B_1, \ldots, B_n)$

$$R \div S = \{t_r[X] \mid t_r \in R \wedge \pi_y(S) \subseteq Y_x\}$$

where x= $t_r[X]$,

Y_x is the image set of x in R (象集)

The steps of division: R ÷ S

(1) All attributes in R are divided into two sets X and Y, R(X, Y), where Y is the first part of S(Y,Z).

(2) Let the image set (IS) Y_x of a certain value x of X is as follows.

$$Y_x=\{t[Y] \mid t \in R \wedge t[X]=x\}$$

(3) If Y_x includes all t[Y] in table S, then output x into result set.

e. g:

R

X		Y	
A	B	C	D
a	b	c	d
a	b	e	f
a	b	d	e
b	c	e	f
e	d	c	d
e	d	e	f

S

Y	
C	D
c	d
e	f

When x=(b,c), its IS Y_x为:

C	D
e	f

When x=(a,b), its IS Y_x为:

C	D
c	d
e	f
d	e

When x=(e,d), its IS Y_x为:

C	D
c	d
e	f

S

C	D
c	d
e	f

The Result set of R÷S:

X

A	B
a	b
e	d

e.g. R÷S

R

X		Y
A	B	C
a_1	b_1	c_2
a_2	b_3	c_7
a_3	b_4	c_6
a_1	b_2	c_3
a_4	b_6	c_6
a_2	b_2	c_3
a_1	b_2	c_1

S

Y		Z
B	C	D
b_1	c_2	d_1
b_2	c_1	d_1
b_2	c_3	d_2

(1) When $x=(a_1)$, Its IS is

B	C
b_1	c_2
b_2	c_3
b_2	c_1

X's values $\{a_1, a_2, a_3, a_4\}$

(2)When x=(a_2) its IS:

B	C
b_3	c_7
b_2	c_3

(3) When x=(a_3) its IS:

B	C
b_4	c_6

(4) When x=(a_4) its IS:

B	C
b_6	c_6

S's project on Y (B,C)

B	C
b_1	c_2
b_2	c_1
b_2	c_3

R÷S:

X

A
a_1

Example of Division Operation

Relations *r, s*:

r

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
ϵ	6
ϵ	1
β	2

s

B
1
2

***r* ÷ *s*:**

A
α
β

Another Division Example

Relations *r*, *s*:

r

A	*B*	*C*	*D*	*E*
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

s

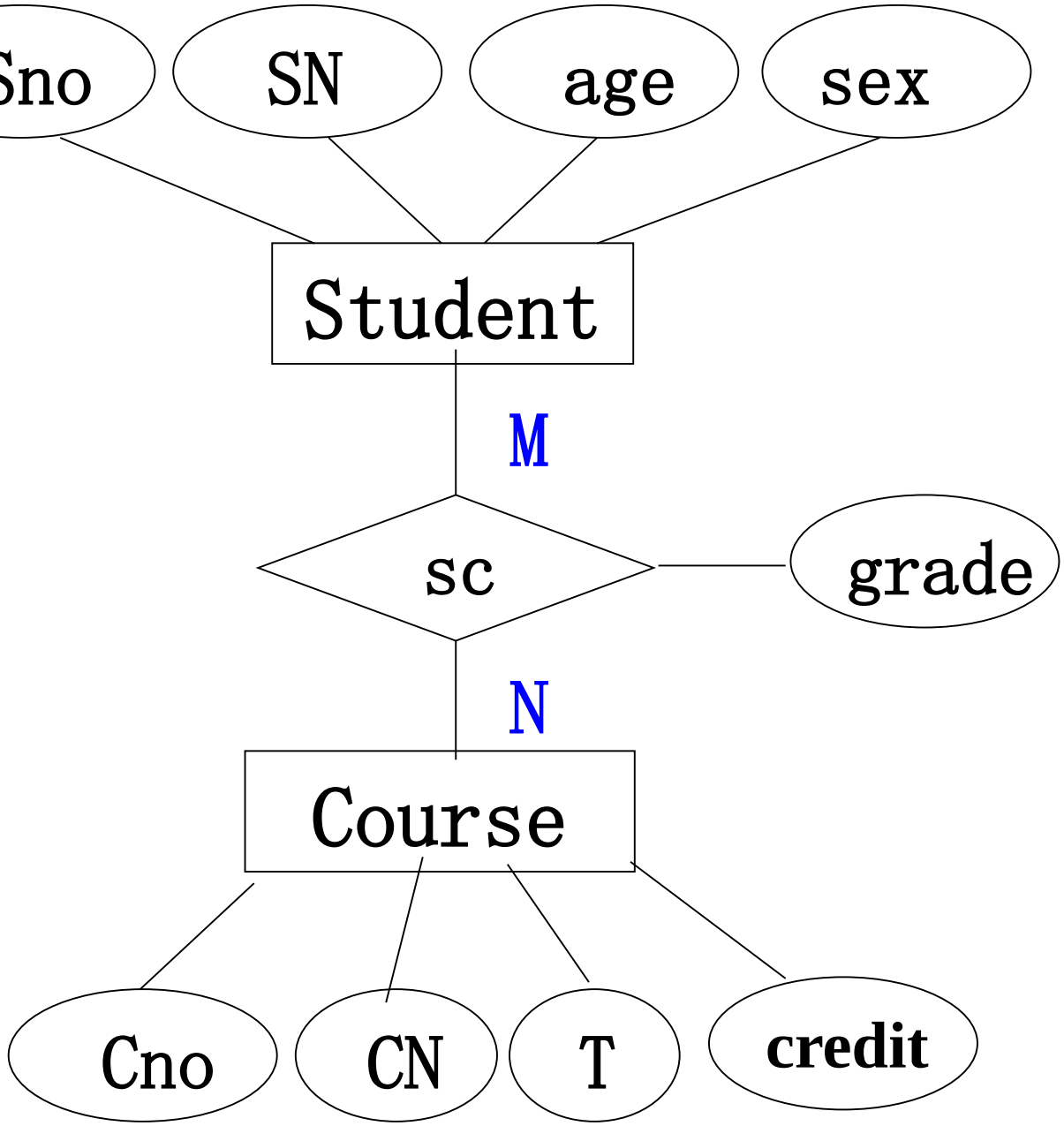
D	*E*
a	1
b	1

***r* ÷ *s*:**

A	*B*	*C*
α	a	γ
γ	a	γ

Exercises：

学号—Sno
姓名—SN
年龄—age
性别—sex
课程号—Cno
课程名—CN
教师名—T
学分-credit
成绩—G



case：

学号—Sno
姓名—SN
年龄—age
性别—sex
课程号—Cno
课程名—CN
教师名—T
学分-credit
成绩—G

Relation Schemas

Student: S(Sno,SN,age,sex)

Course: C(Cno,CN,T,credit)

SC: SC(Sno,Cno,G)

An instance of S(Sno,SN,age,sex)

Sno	SN	age	sex
S1	LI	17	M
S2	SHI	19	F
S3	LIU	21	F
S4	CHEN	20	M

An instance of C(Cno,CN,T,credit)

Cno	CN	T	credit
C1	MATHS	LIU	6
C2	PHYSICS	LI	4
C3	DB	SHI	2
C4	OS	FAN	2

An instance of SC(Sno,Cno,G)

Sno	Cno	G
S1	C1	90
S1	C2	78
S1	C3	86
S1	C4	77
S2	C1	76
S2	C2	89
S3	C3	75
S4	C4	80

1.Find all students who takes C2 as an elective course, list their student numbers and grades.

$$\pi_{Sno,\,G}(\sigma_{Cno='C2'}(SC))$$

$$\sigma_{Cno='C2'}(\pi_{Sno,\,G}(SC))$$ wrong

2.Find all students who takes C2 as an elective course, list their student numbers and names.

$$\pi_{Sno,\,SN}(\sigma_{Cno='C2'}(S \bowtie SC))$$

$$\pi_{Sno,\,SN}(S \bowtie \sigma_{Cno='C2'}(SC))$$

$$\pi_{S.Sno,\,SN}(\sigma_{Cno='C2' \wedge S.Sno=SC.Sno}(S \times SC))$$

3. Find all students who takes DB as an elective course, list their student numbers and names.

$$\pi_{Sno,\,SN}(\sigma_{CN='DB'}(S \bowtie SC \bowtie C))$$

$$\pi_{S.Sno,\,SN}(\sigma_{CN='DB' \wedge S.Sno=SC.Sno \wedge SC.Cno=C.Cno}(S \times SC \times C))$$

$$\pi_{Sno,\,SN}(S \bowtie SC \bowtie \sigma_{CN='DB'}(C))$$

4. Find all students who takes C2 or C4 as an elective course, list their student numbers.

$$\pi_{Sno}(\sigma_{Cno='C2' \vee Cno='C4'}(SC))$$

5. Find all students who at least takes C2 and C4 as elective courses, list their student numbers.

$$\pi_{1}(\sigma_{1=4 \wedge 2=\text{'C2'} \wedge 5=\text{'C4'}}(SC \times SC))$$

1	2	3	4	5	6
Sno	Cno	G	Sno	Cno	G

6. Find all students who do not take C2 as an elective course, list their names and ages.

$$\pi_{SN,\,age}(S) - \pi_{SN,\,age}(S \bowtie \sigma_{Cno=\text{'C2'}}(SC))$$

$$\pi_{SN,\,age}(S \bowtie \sigma_{Cno\neq\text{'C2'}}(SC))$$ **wrong**

7. **Find all students who takes all courses as their elective courses, list their names.**

$$\pi_{SN}(S \bowtie (\pi_{Sno,\,Cno}(SC) \div \pi_{Cno}(C)))$$

8. **Find all students who takes all course which S2 takes as their elective courses, list their student numbers.**

$$\pi_{Sno,\,Cno}(SC) \div \pi_{Cno}(\sigma_{Sno='S2'}(SC))$$