Intro to DB

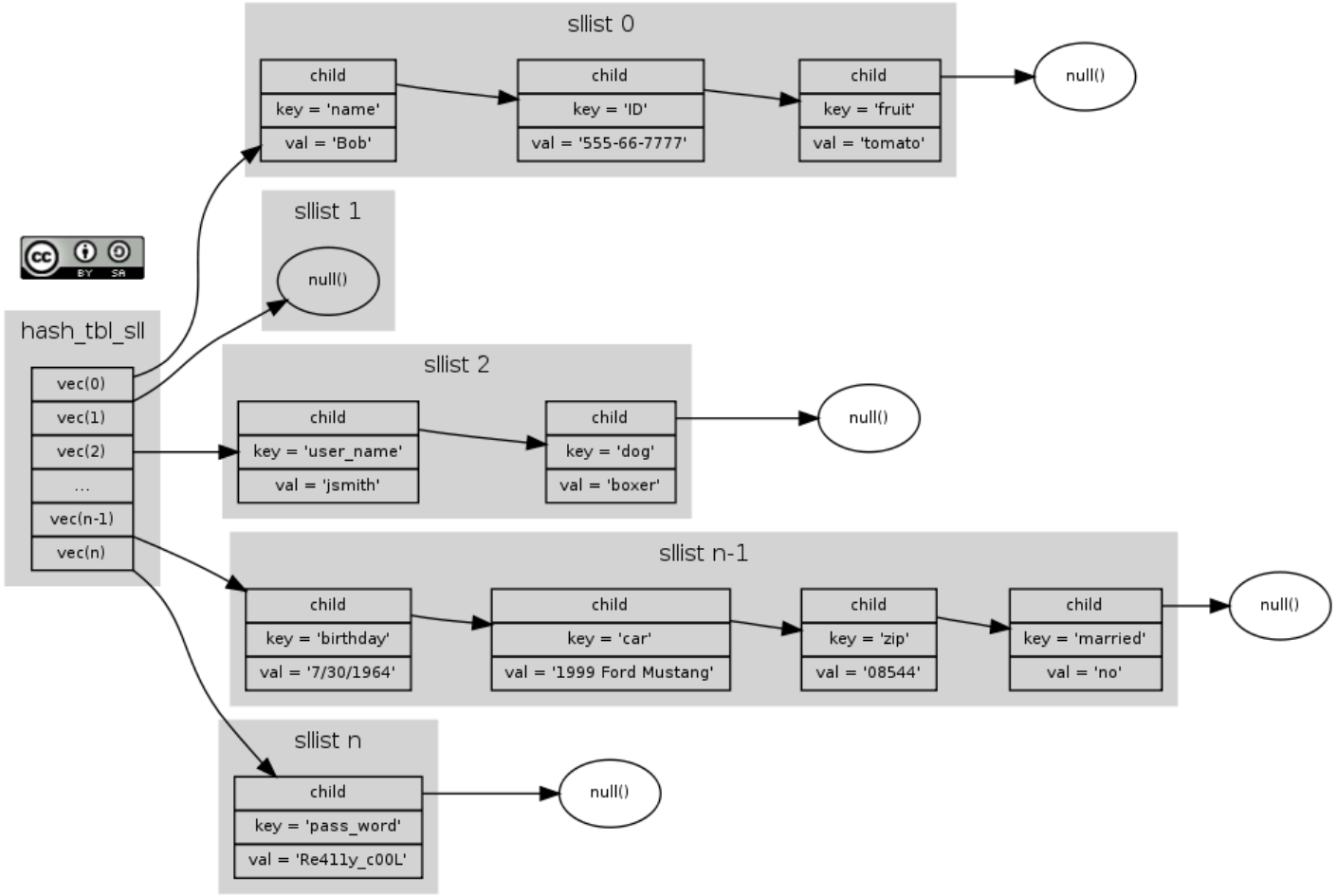
CHAPTER 2
RELATIONAL MODEL

Chapter 2: Relational Model

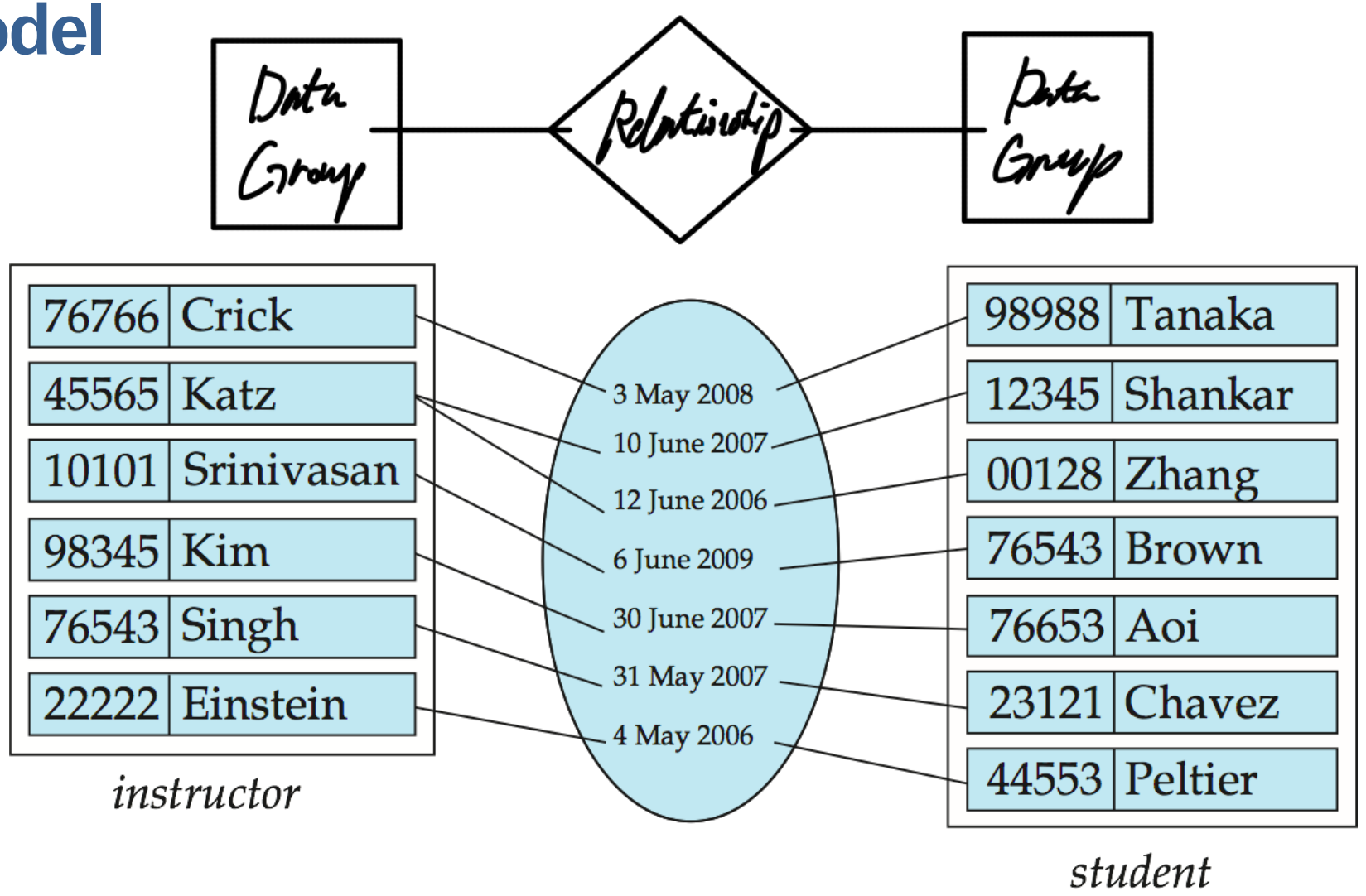
- Structure of Relational Databases
- Database Schema
- Keys
- Schema Diagrams
- Relational Query Languages
- Relational Algebra

Data Model

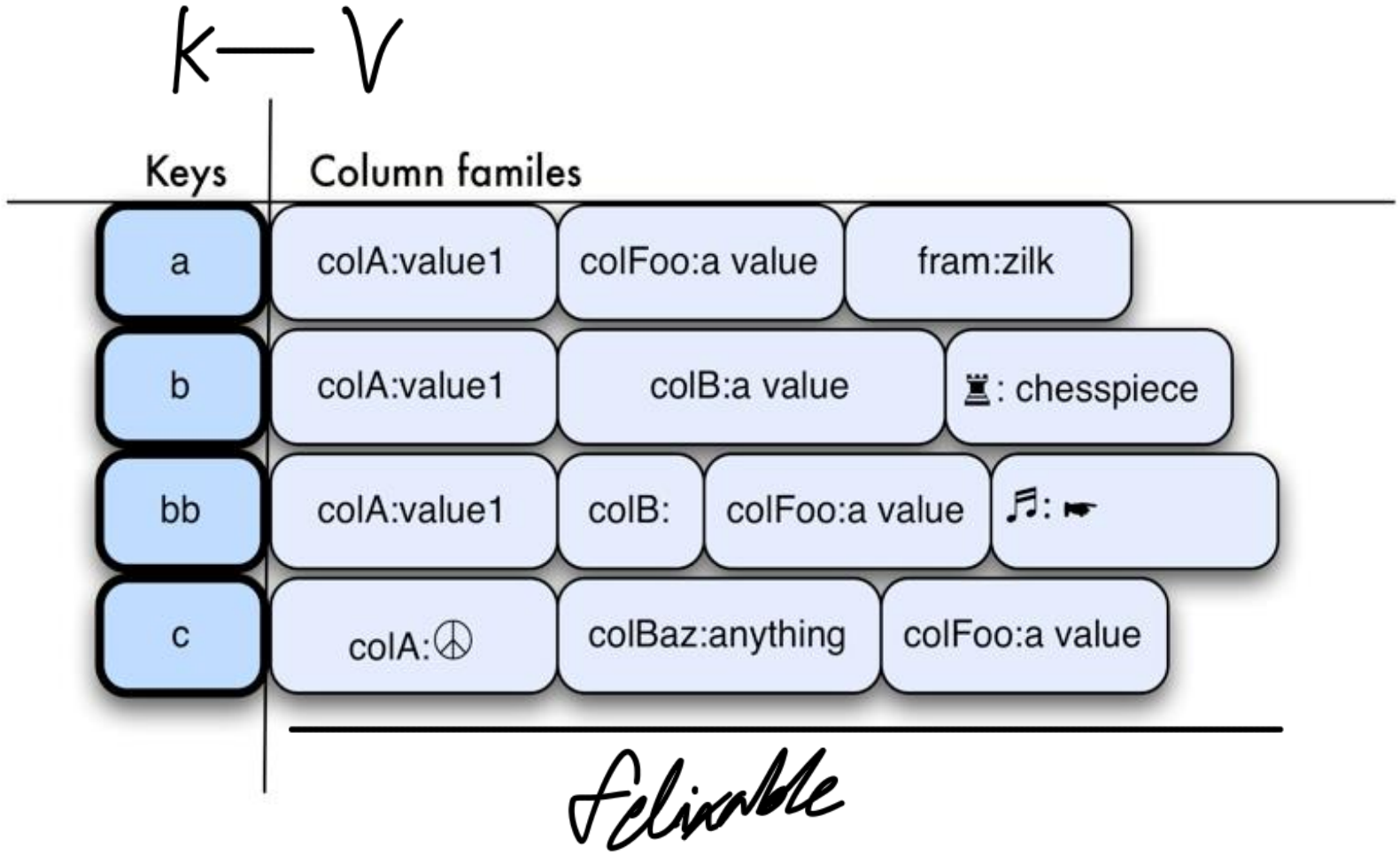
- The framework/formalism for representing data and their relationships



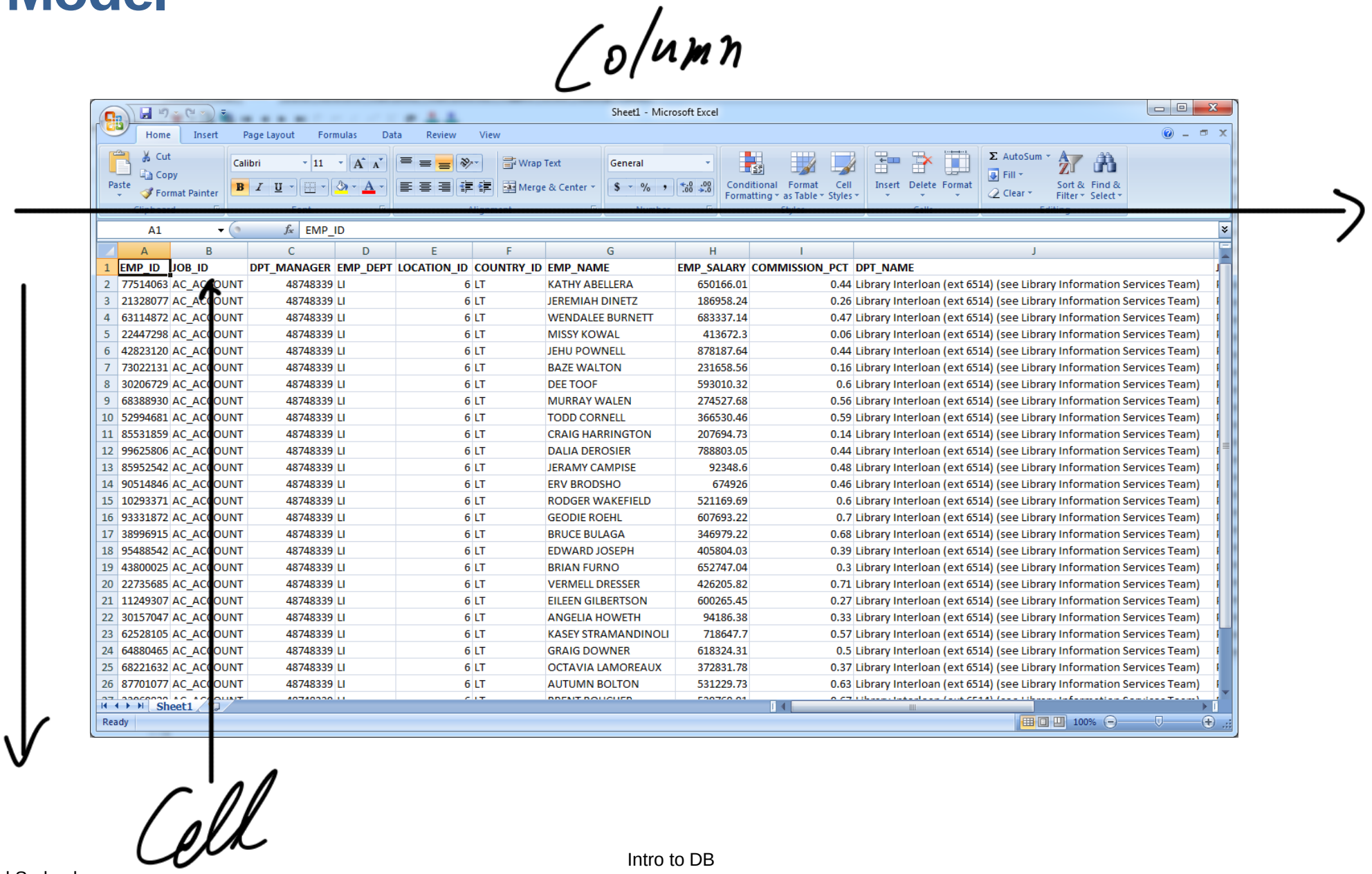
Data Model


Data Group
Relationship
Data Group
76766 Crick
45565 Katz
10101 Srinivasan
98345 Kim
76543 Singh
22222 Einstein
instructor
3 May 2008
10 June 2007
12 June 2006
6 June 2009
30 June 2007
31 May 2007
4 May 2006
98988 Tanaka
12345 Shankar
00128 Zhang
76543 Brown
76653 Aoi
23121 Chavez
44553 Peltier
student

Data Model

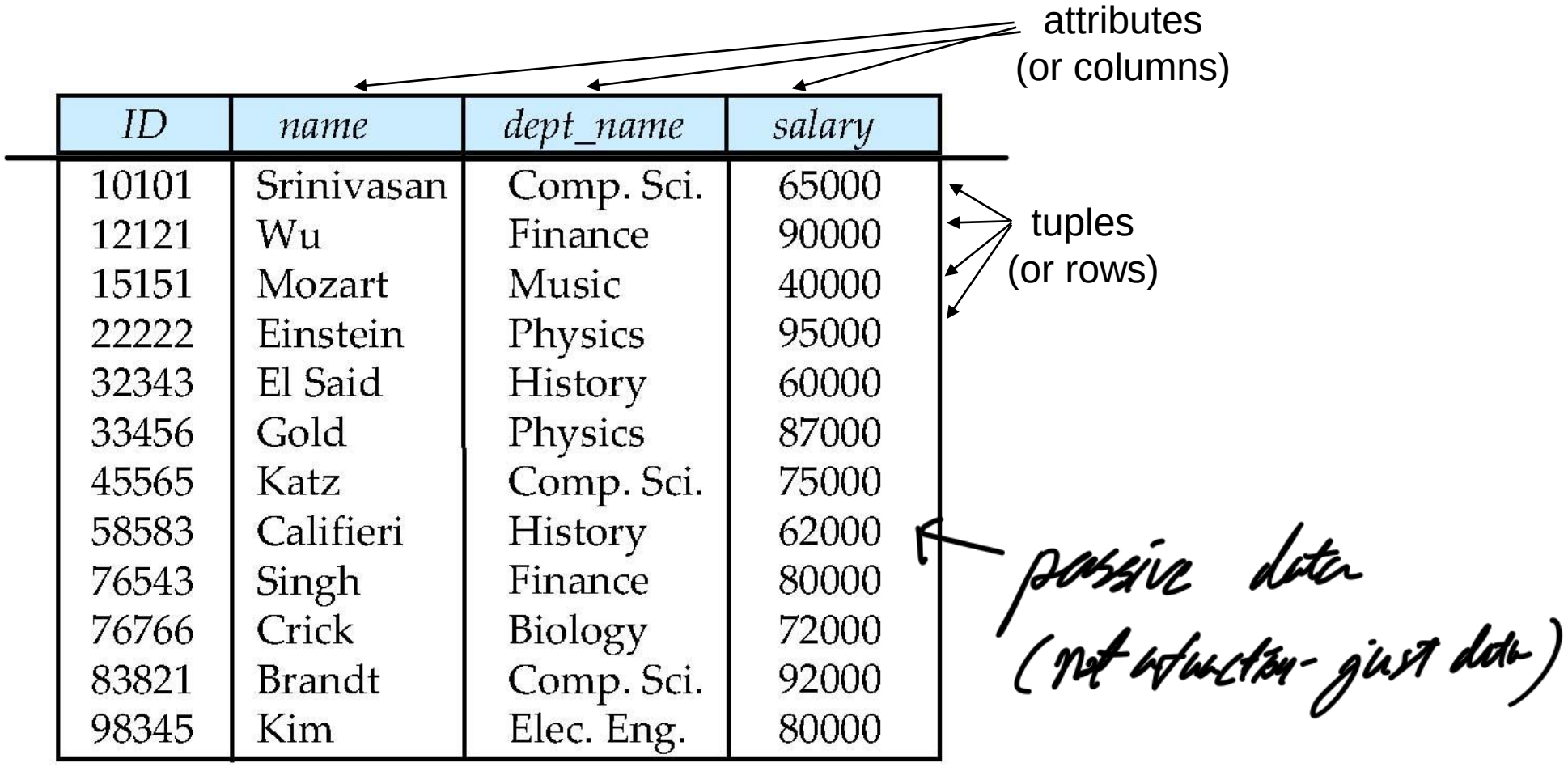

k — V
Keys
Column familes
a
colA:value1
colFoo:a value
fram:zilk
b
colA:value1
colB:a value
: chesspiece
bb
colA:value1
colB:
colFoo:a value
: ☛
c
colA:☮
colBaz:anything
colFoo:a value
felicable

Data Model



EMP_ID	JOB_ID	DPT_MANAGER	EMP_DEPT	LOCATION_ID	COUNTRY_ID	EMP_NAME	EMP_SALARY	COMMISSION_PCT	DPT_NAME
77514063	AC_ACCOUNT	48748339	LI	6	LT	KATHY ABELLERA	650166.01	0.44	Library Interloan (ext 6514) (see Library Information Services Team)
21328077	AC_ACCOUNT	48748339	LI	6	LT	JEREMIAH DINETZ	186958.24	0.26	Library Interloan (ext 6514) (see Library Information Services Team)
63114872	AC_ACCOUNT	48748339	LI	6	LT	WENDALEE BURNETT	683337.14	0.47	Library Interloan (ext 6514) (see Library Information Services Team)
22447298	AC_ACCOUNT	48748339	LI	6	LT	MISSY KOWAL	413672.3	0.06	Library Interloan (ext 6514) (see Library Information Services Team)
42823120	AC_ACCOUNT	48748339	LI	6	LT	JEHU POWNELL	878187.64	0.44	Library Interloan (ext 6514) (see Library Information Services Team)
73022131	AC_ACCOUNT	48748339	LI	6	LT	BAZE WALTON	231658.56	0.16	Library Interloan (ext 6514) (see Library Information Services Team)
30206729	AC_ACCOUNT	48748339	LI	6	LT	DEE TOOF	593010.32	0.6	Library Interloan (ext 6514) (see Library Information Services Team)
68388930	AC_ACCOUNT	48748339	LI	6	LT	MURRAY WALEN	274527.68	0.56	Library Interloan (ext 6514) (see Library Information Services Team)
52994681	AC_ACCOUNT	48748339	LI	6	LT	TODD CORNELL	366530.46	0.59	Library Interloan (ext 6514) (see Library Information Services Team)
85531859	AC_ACCOUNT	48748339	LI	6	LT	CRAIG HARRINGTON	207694.73	0.14	Library Interloan (ext 6514) (see Library Information Services Team)
99625806	AC_ACCOUNT	48748339	LI	6	LT	DALIA DEROSIER	788803.05	0.44	Library Interloan (ext 6514) (see Library Information Services Team)
85952542	AC_ACCOUNT	48748339	LI	6	LT	JERAMY CAMPISE	92348.6	0.48	Library Interloan (ext 6514) (see Library Information Services Team)
90514846	AC_ACCOUNT	48748339	LI	6	LT	ERV BRODSHO	674926	0.46	Library Interloan (ext 6514) (see Library Information Services Team)
10293371	AC_ACCOUNT	48748339	LI	6	LT	RODGER WAKEFIELD	521169.69	0.6	Library Interloan (ext 6514) (see Library Information Services Team)
93331872	AC_ACCOUNT	48748339	LI	6	LT	GEODIE ROEHL	607693.22	0.7	Library Interloan (ext 6514) (see Library Information Services Team)
38996915	AC_ACCOUNT	48748339	LI	6	LT	BRUCE BULAGA	346979.22	0.68	Library Interloan (ext 6514) (see Library Information Services Team)
95488542	AC_ACCOUNT	48748339	LI	6	LT	EDWARD JOSEPH	405804.03	0.39	Library Interloan (ext 6514) (see Library Information Services Team)
43800025	AC_ACCOUNT	48748339	LI	6	LT	BRIAN FURNO	652747.04	0.3	Library Interloan (ext 6514) (see Library Information Services Team)
22735685	AC_ACCOUNT	48748339	LI	6	LT	VERMELL DRESSER	426205.82	0.71	Library Interloan (ext 6514) (see Library Information Services Team)
11249307	AC_ACCOUNT	48748339	LI	6	LT	EILEEN GILBERTSON	600265.45	0.27	Library Interloan (ext 6514) (see Library Information Services Team)
30157047	AC_ACCOUNT	48748339	LI	6	LT	ANGELIA HOWETH	94186.38	0.33	Library Interloan (ext 6514) (see Library Information Services Team)
62528105	AC_ACCOUNT	48748339	LI	6	LT	KASEY STRAMANDINOLI	718647.7	0.57	Library Interloan (ext 6514) (see Library Information Services Team)
64880465	AC_ACCOUNT	48748339	LI	6	LT	GRAIG DOWNER	618324.31	0.5	Library Interloan (ext 6514) (see Library Information Services Team)
68221632	AC_ACCOUNT	48748339	LI	6	LT	OCTAVIA LAMOREAUX	372831.78	0.37	Library Interloan (ext 6514) (see Library Information Services Team)
87701077	AC_ACCOUNT	48748339	LI	6	LT	AUTUMN BOLTON	531229.73	0.63	Library Interloan (ext 6514) (see Library Information Services Team)

Data Model - Relational



ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Relation (in Set Theory)

- **Binary relation** *R* on a set *A*
  is a collection of ordered pairs of elements of *A*

$A = \{1, 2, 3\}$

- $R \subseteq A \times A$

$< = \{\langle 1,2\rangle, \langle 1,3\rangle, \langle 2,3\rangle\}$

$\leq = \{\langle 1,1\rangle, \langle 1,2\rangle, \langle 1,3\rangle, \langle 2,2\rangle, \langle 2,3\rangle, \langle 3,3\rangle\}$

- *(Bipartite) Relation R2* between elements of set *B* and set *C*

ex) B = Set of Country / C = Set of City

Capital City = $\{\langle B_i, C_j\rangle \cdots\}$

Basic Structure of a Relation

- Formally,

  given sets D_1, D_2, …. D_n , a *relation* r is a subset of D_1 x D_2 x … x D_n

  $r \subseteq D_1$ X ….. X D_n (n: degree of r)

  or

  $r = \{ <d_1, ..., d_n> \mid d_1 \in D_1, ..., d_n \in D_n \}$ (set of tuples)

- Example: *customer-name* = {Jones, Smith, Curry, Lindsay}
  customer-street = {Main, North, Park}
  customer-city = {Harrison, Rye, Pittsfield}

  Then,

  r = { (Jones, Main, Harrison), (Smith, North, Rye), (Curry, North, Rye), (Lindsay, Park, Pittsfield)}

  is a relation over *customer-name x customer-street x customer-city*

Attributes

actual data of customer's address

Attribute Types

- Each attribute of a relation has a name
- Domain of an attribute: the set of allowed values allowed for each attribute
- Attribute values are (normally) required to be atomic, that is, indivisible
  - E.g. multivalued attribute values are not atomic
  - E.g. composite attribute values are not atomic

- The special value *null* is a member of every domain
  - Represents *unknown or missing value*

Relation Schema

- $A_1, A_2, \ldots, A_n$ are *attributes*

- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

  e.g.

  Customer-schema = (*customer-name, customer-street, customer-city*)

- $r(R)$ is a *relation* (*variable*) on the *relation schema R*

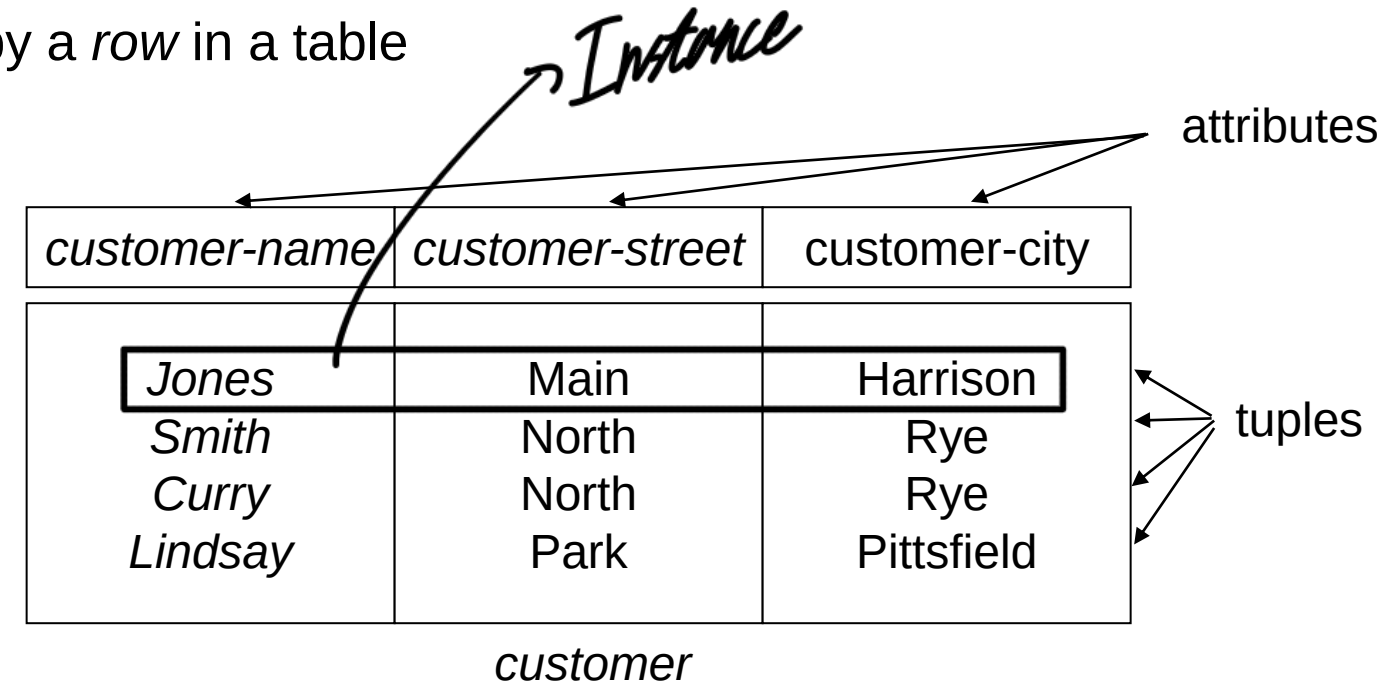
  e.g.

  customer(*Customer-schema*)

Relation Instance

- The current values of a relation
  - Specified by a table

- An element *t* of *r* is a *tuple*,
  - represented by a *row* in a table



Relations are Unordered

- Order of tuples is irrelevant

  (tuples may be stored in an arbitrary order)

- E.g. *instructor* relation with unordered tuples

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Relational Database : Set of Multiple table

- A database consists of multiple relations
- Information about an enterprise is broken up into parts, with each relation storing one part of the information

  E.g.: *instructor*: information about teachers in the university

  student: information about students

  advisor: information about which instructor advises which students

- Storing all information as a single relation is not a good idea

  univ (*instructor -ID, name, dept_name, salary, student_Id*, ..)

- Database design (Chapter 6 & 7) deals with how to decide on the relational schemas

A relational database for a university

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565			
98345			
76766			
10101			
58583			
83821			
15151			
33456			
76543			

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Desig		
CS-315	Robotics		
CS-319	Image Proce		
CS-347	Database Sy		
EE-181	Intro. to Dig		
FIN-201	Investment		
HIS-351	World Histo		
MU-199	Music Video		
PHY-101	Physical Pri		

dept_name	building	budget
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	
CS-190	2	Spring	2009	Taylor	3128	
CS-315	1	Spring	2010	Watson	120	
CS-319	1	Spring	2010	Watson	100	
CS-319	2	Spring	2010	Taylor	3128	
CS-347	1	Fall	2009	Taylor	3128	
EE-181	1	Spring	2009	Taylor	3128	
FIN-201	1	Spring	2010	Packard	101	
HIS-351	1	Spring	2010	Painter	514	
MU-199	1	Spring	2010	Packard	101	
PHY-101	1	Fall	2009	Watson	100	

course_id	prereq_id
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

Keys

- Let $K \subseteq R$
- *K* is a *superkey* of *R*

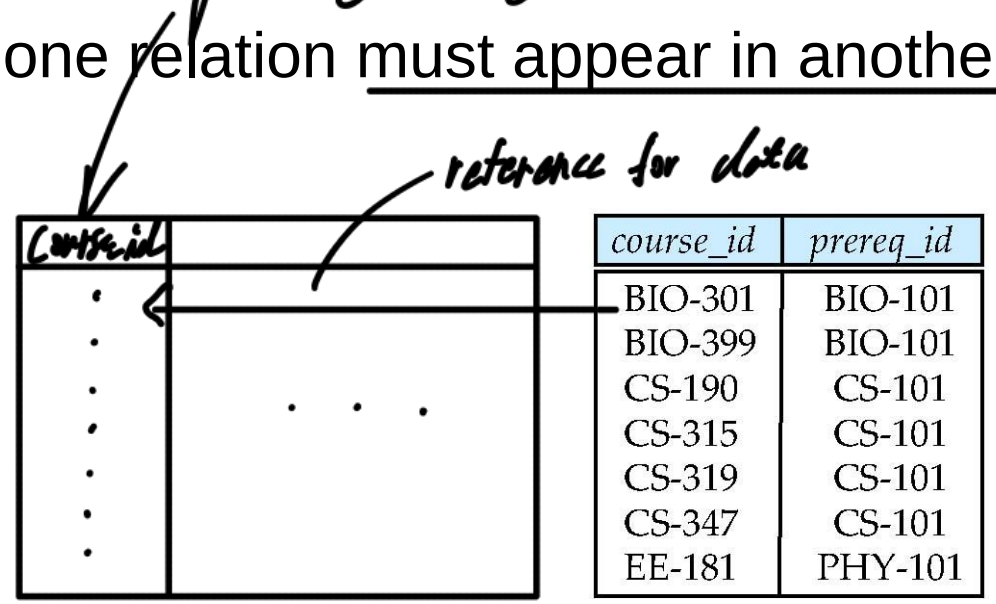
  if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R).*
- By "possible *r*" we mean a relation *r* that could exist in the enterprise we are modeling.
  - Example: {*customer-name, customer-street*} and {*customer-name*}

    are both superkeys of *Customer*, if no two customers can possibly have the same name.
- *K* is a *candidate key* if *K* is minimal
  - Example: {*customer-name*} is a candidate key for *Customer*,

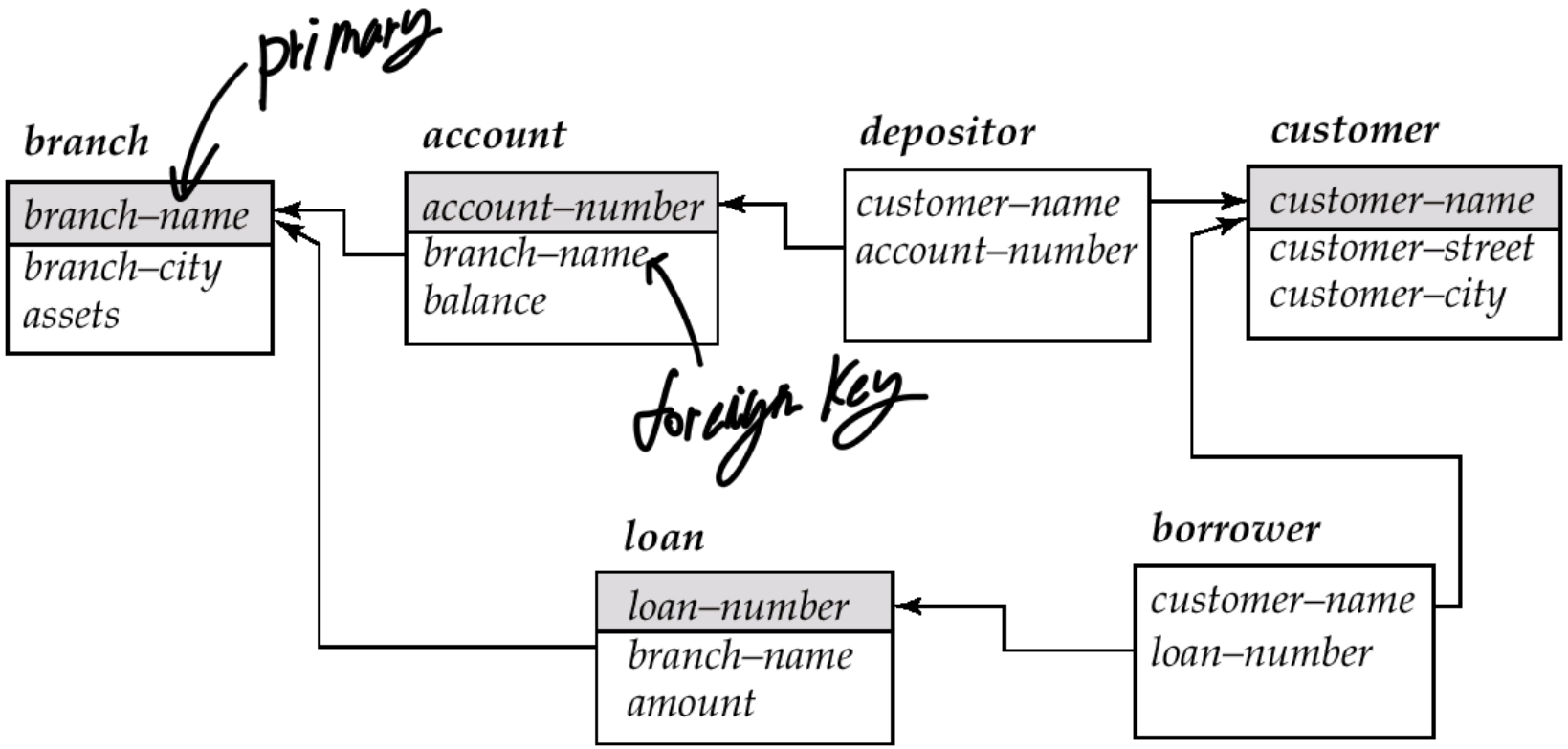
    since it is a superkey, and no subset of it is a superkey.

Keys

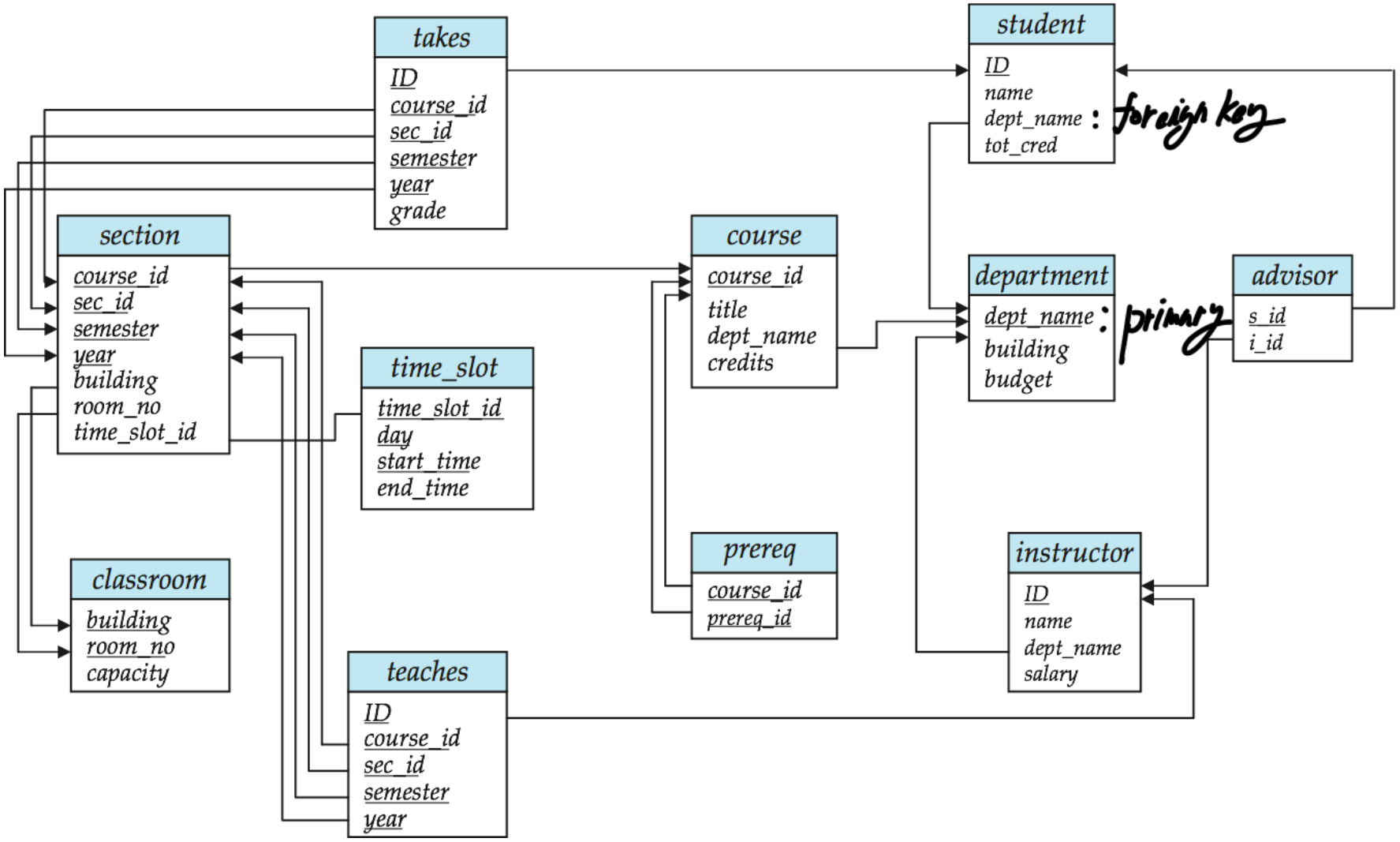
- **Primary key**: one of the candidate keys is selected to be the primary key of the table
  - which one? : representative candidate key of Data
- **Foreign key**: Value in one relation must appear in another : kind of pointer
  - **Referencing** relation
  - **Referenced** relation



Schema Diagram for a Banking Enterprise



Schema Diagram for the University



Query Languages

- Language in which user requests information from the database.
  - procedural : Sequencial
    - specify what data are needed
    - and "how" to get those data
  - nonprocedural: Conditional (SQL)
    - specify only "what" data are needed
    - declarative
- "Pure" languages:
  - Relational Algebra
  - Tuple Relational Calculus
  - Domain Relational Calculus
- Pure languages form underlying basis of query languages that people use.

Relational Algebra : Procedural

- Algebra : operators and operands 대수
  - Relational algebra
    - operands : relations
    - operators : basic operators (+ additional operations)
      - take two or more relations as inputs and give a new relation as a result.
- Procedural language
- Operators
  - select
  - project
  - union
  - set difference
  - Cartesian product
  - rename

  join
  division
  assignment
  …

Examples of Relational Operators

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{\text{salary}>=85000}(instructor)$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID,\ salary}(instructor)$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
$\bowtie$ (Natural Join)	$instructor \bowtie department$
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
$\times$ (Cartesian Product)	$instructor \times department$
	Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
$\cup$ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the two input relations.

Select Operation – selection of tuples

Relation r :

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

: record

$\sigma_{A=B \wedge D>5}(r)$:

$(A = B) \wedge (D > 5)$

and

A	B	C	D
α	α	1	7
β	β	23	10

Select Operation

- Notation: $\sigma_p(r)$
- p is called the selection predicate
- Defined as: $\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$

  - Where p is a formula in propositional calculus consisting of terms connected by : $\wedge$ (*and*), $\vee$ (*or*), $\neg$ (*not*)
  - Each term is one of:

    <attribute> *op* <attribute> or <constant>

    where *op* is one of: $=, \neq, >, \geq, <, \leq$

- Example: $\sigma_{\textit{branch-name}=\text{“Perryridge”}}(\textit{account})$

Project Operation – selection of columns

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

get A,C column

$\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Project Operation

- Notation:

$$\Pi_{A1, A2, \ldots, Ak}\ (r)$$

  where A_1, A_2 are attribute names and r is a relation name.
- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result

  – since relations are sets

- E.g.: To eliminate the *branch-name* attribute of *account*

$$\Pi_{account\text{-}number,\ balance}\ (account)$$

↳ Indicate Attribute to get

Union Operation – merging two relations

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cup s$:

A	B
α	1
α	2
β	1
β	3

Union Operation

- Notation: $r \cup s$
- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid (Union compatible)
  1. *r, s* must have the *same arity* (same number of attributes)
  2. The attribute domains must be *compatible*

  (e.g., 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

  same Schema

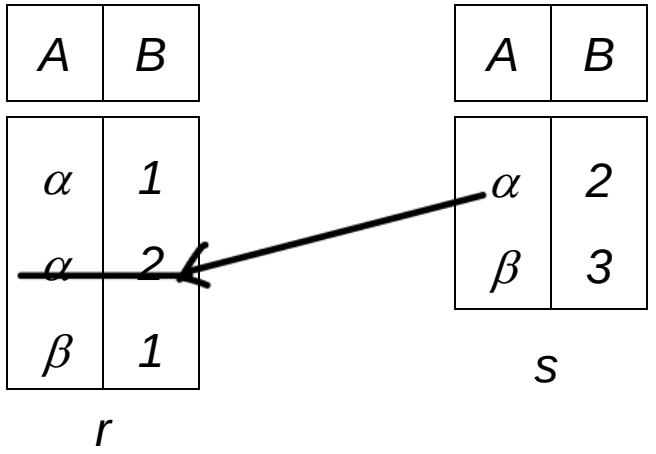
  e.g.: to find all customers with either an account or a loan

  $\Pi_{customer\text{-}name}(depositor) \cup \Pi_{customer\text{-}name}(borrower)$

but $\Pi_{customer\text{-}name}(depositor \cup borrower)$ is invalid

Set Difference Operation : 차집합

- Relations *r, s:*



A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- *r – s:*

A	B
α	1
β	1

Set Difference Operation

- Notation $r - s$
- Defined as:

$$r - s = \{t \mid t \in r \text{ and } t \notin s\}$$

- Set differences must be taken between *compatible* relations.
  - r and s must have the *same arity*
  - attribute domains of r and s must be compatible

Cartesian-Product Operation – Example

- Relations *r, s:*

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s:*

A	B	C	D	E
α	1	α	10	a
α	1	β	19	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

∴ ∀ combination set of r, s

Cartesian-Product Operation

- Notation: $r \times s$
- Defined as:

$$r \times s = \{ t\, q \mid t \in r \text{ and } q \in s \}$$

  - Assume that attributes of $r(R)$ and $s(S)$ are disjoint; ⟶ no same attr
    i.e., $R \cap S = \varnothing$.
  - If not, renaming of attributes is needed.

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$
- $r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	19	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

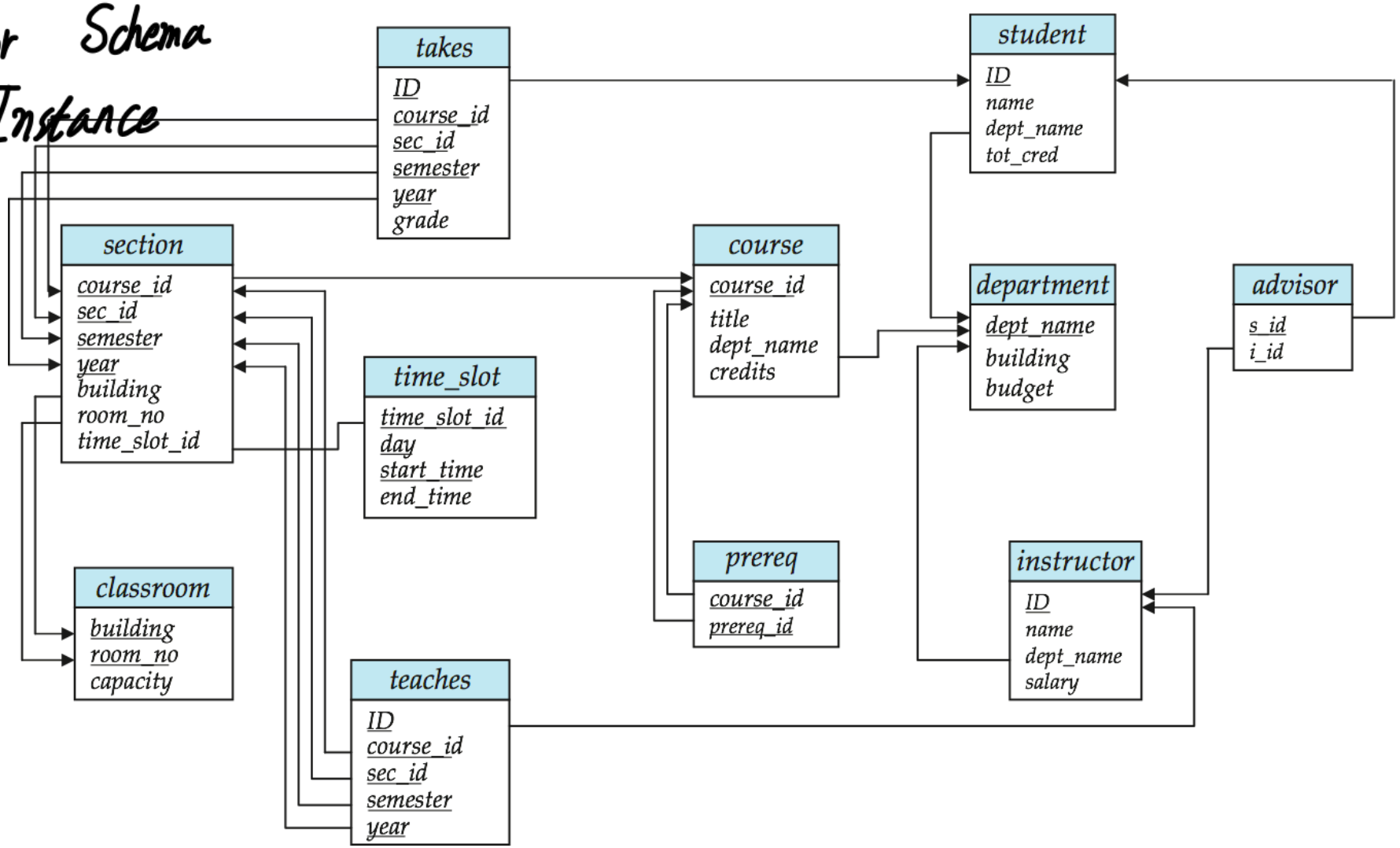
- $\sigma_{A=C}(r \times s)$

same

A	B	C	D	E
α	1	α	10	a
β	2	β	20	a
β	2	β	20	b

Schema Diagram for University Database

Query for Schema
not for Instance



Example Queries

- Find all students in the CS department

$$\sigma_{dept\_name = \text{"CS"}}(student)$$

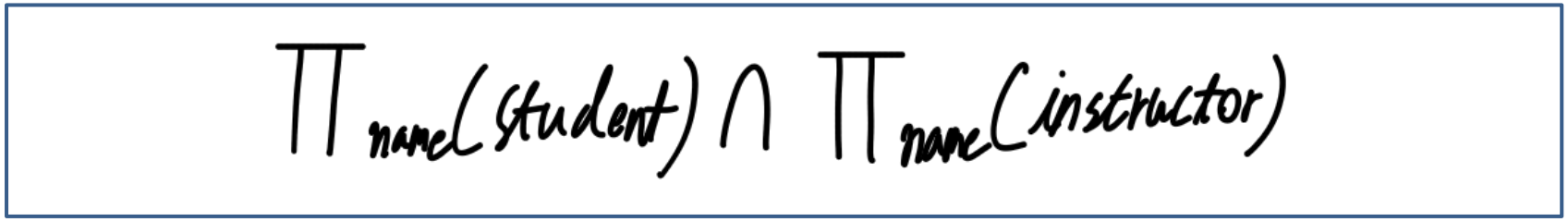
- Find the name of each student in the CS department

$$\Pi_{name}(\sigma_{dept\_name = \text{'CS'}}(Student))$$

Example Queries

- Find the names of all persons who are either an instructor or a student

- Find the names of all persons who are both an instructor and a student
  (assuming names are unique)



Example Queries

- Find the names of all students who takes/took course CS-101.

Query 1

$$\Pi_{name}(\sigma_{course\_id='CS\text{-}101'}(\sigma_{takes.ID=Student.ID}(takes \times Student)))$$

↳ referring

Query 2

$$\Pi_{name}(\sigma_{takes.ID=student.ID}(Student \times \sigma_{course\_ID='CS\text{-}101'}(takes)))$$

Rename Operation In Cartesian Product

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.

- $\rho_N (E)$ returns the expression E under the name N

- If a relational-algebra expression E has arity n, then

$$\rho_{N\ (A1, A2, \ldots, An)} (E)$$

→ new Attr Names

returns the result of expression E under the name N, and with the attributes renamed to $A_1, A2, \ldots, An$.

Formal Definition

- A *basic expression in the relational algebra* consists of either one of the following:
  - A relation in the database
  - A constant relation

- Let E_1 and E_2 be relational-algebra expressions; the following are all *relational-algebra expressions*:
  - $E_1 \cup E_2$ Union
  - $E_1 - E_2$ difference
  - $E_1 \times E_2$ Cartesian product
  - $\sigma_p(E_1)$, P is a predicate on attributes in E_1 select
  - $\prod_s(E_1)$, S is a list consisting of some of the attributes in E_1 project
  - $\rho_N(E_1)$, N is the new name for the result of E_1 rename

$\cap$: join? $\bar{E}_1 - (\bar{E}_1 - \bar{E}_2) = E_1 \cap E_2$

Additional Operations

We define additional operations that do not add any power to the relational algebra, but that simplify common queries.

- Set intersection
- Natural join
- Assignment
- Outer Join
- Generalized Projection
- Aggregation

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:

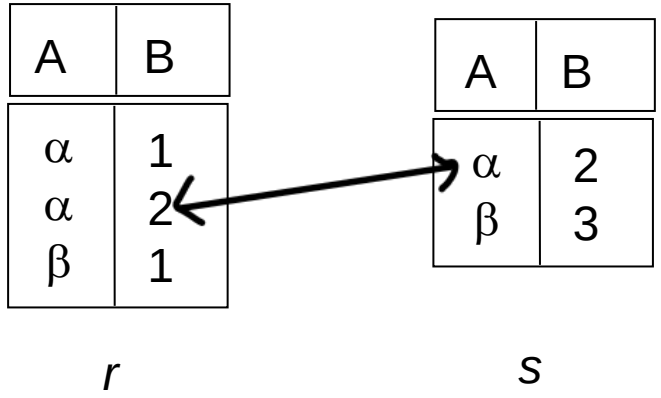
$$r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$$

- Assume union compatibility:
  - r, s have the *same arity*
  - attributes of r and s are compatible

- Note: $r \cap s = r - (r - s)$

Set-Intersection Operation – Example

- Relation *r*, *s*:



- $r \cap s$

A	B
α	2

Natural-Join Operation

join table with primary-foreign key-set
combine if match attr

- Let $r(R)$ and $s(S)$
- Notation: $r \bowtie s$
- The result is a relation on schema $R \cup S$ which is obtained by considering each pair of tuples t_r from r and t_s from s.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$,
    a tuple t is added to the result, where
    t has the same value as t_r on R, and t has the same value as t_s on S.
    (i.e., $t[R]=t_r$ *and* $t[S]=t_s$)

- Example:

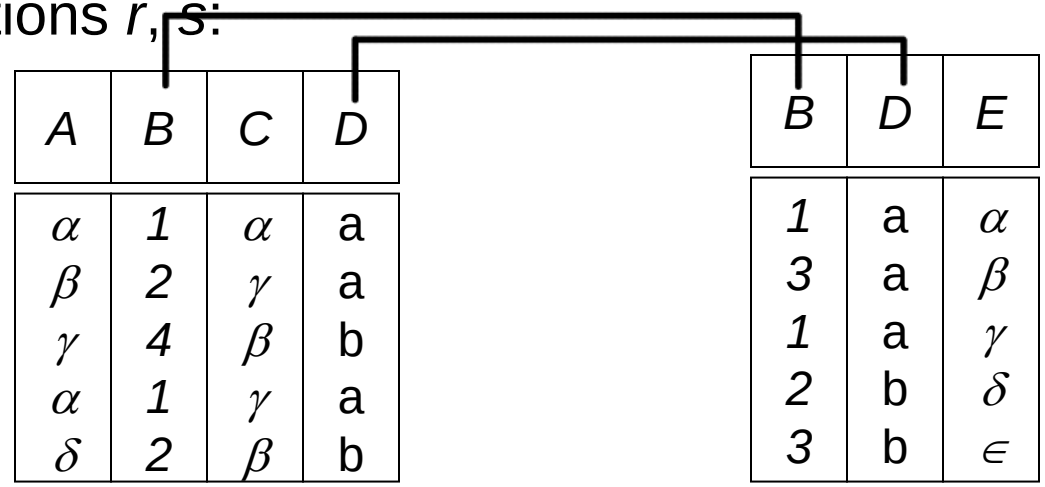
  $R = (A, B, C, D)$ & $S = (E, B, D)$

  Result schema = (A, B, C, D, E)

  $$r \bowtie s = \prod_{r.A,\ r.B,\ r.C,\ r.D,\ s.E} (\sigma_{r.B=s.B \wedge r.D=s.D} (r \times s))$$

Natural-Join Operation – Example

- Relations r, s:



A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	$\in$

s

- $r \bowtie s$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Properties

- $(r \bowtie s) \bowtie t = r \bowtie (s \bowtie t)$: exchange
- If $R \cap S = \varnothing$ then $r \bowtie s = r \times s$
- If $R = S$ then $r \bowtie s = r \cap s$

- Theta Join
  - combine selection with Cartesian product

select

$$r \bowtie_\theta s = \sigma_\theta(r \times s)$$

where θ is a predicate on $R \times S$

Assignment Operation

- The assignment operation (←)
  - provides a convenient way to express complex queries
  - write query as a sequential program consisting of a series of assignments
- Assignment must always be made to a temporary relation variable.
  - The result to the right hand side is assigned to the relation variable on the left hand side.
  - May use the variable in subsequent expressions.
- Example: $r \cap s = r - (r - s)$

  $temp \leftarrow r - s$
  $result \leftarrow r - temp$

) such as variable

Example Queries

- Find the names of all students who takes/took both courses CS-101 and CS-190.
  - Query 1

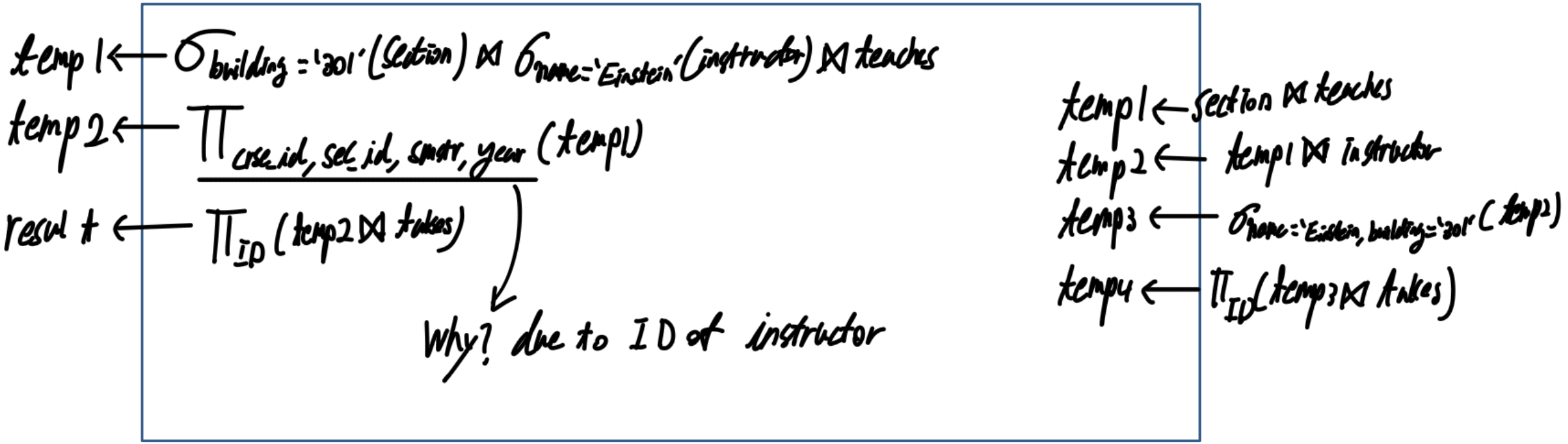
$$\Pi_{name}(\sigma_{course\_id='CS-101'}(student \bowtie takes) \cap \sigma_{course\_id='CS-190'}(student \bowtie takes))$$

  - Query 2

$$\Pi_{name}((\sigma_{course\_id='CS-101'}(takes) \cap \sigma_{course\_id='CS-190'}(takes)) \bowtie student)$$

Example Queries

- Find the IDs of all students who were taught by an instructor named Einstein in building 301.



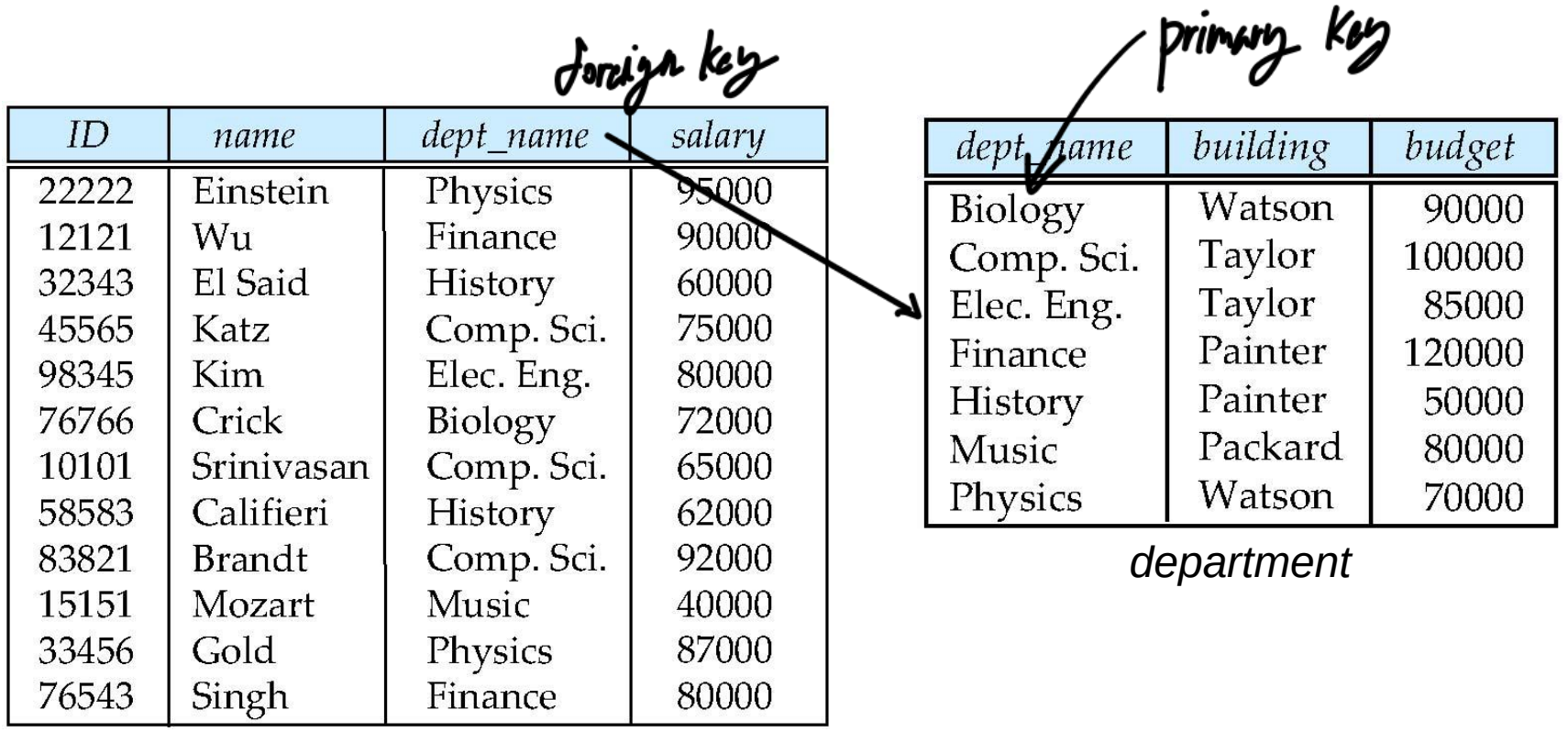
DISCUSSIONS – CHAPTER 2

Discussion 2-1

A. Define *super key, candidate key*, and *primary key*.
B. Why do you think “key”, without any qualifier, is meant to be candidate key instead of super key?

Discussion 2-2

- Consider the foreign key constraint from the *dept_name* attr. of *instructor* to the *department* relation. Give examples of inserts and deletes to these relations, which can cause a violation of the foreign key constraint.



instructor

department

Discussion 2-3

- In the instance of *instructor* shown below, no two instructors have the same name. From this, can we conclude that *name* can be used as a super key (or primary key) of *instructor*?

yes. but not primary key

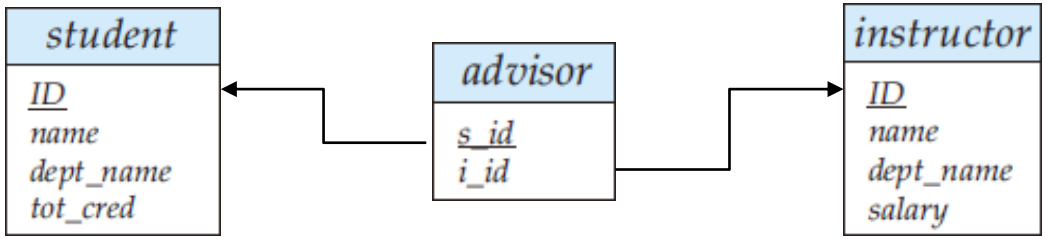
instructor

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Discussion 2-4

- Consider the *advisor* relation shown below, with *s_id* as the primary key of *advisor*. Suppose a student can have more than one advisor. Then, would *s_id* still be a primary key of the *advisor* relation? If not, what should the primary key of *advisor* be?

no, tuple of s_id & i_id



Discussion 2-5

person (*pname*, *street*, *city*)
works (*pname*, *cname*, *salary*)
company (*cname*, *city*) /* keys are underlined

- Using the above database schema, represent the following queries in *relational algebra*.
  - A. *Find all names of persons.* $\Pi_{pname}(person)$
  - B. *Find the names of persons who live in "Seoul"* $\Pi_{pname}(\sigma_{city = "Seoul"}(persons))$
  - C. *Find the names of persons who work in "SNU"* $\Pi_{pname}(\sigma_{cname = "SNU"}(works))$

Discussion 2-6

person (*pname*, *street*, *city*)
works (*pname*, *cname*, *salary*)
company (*cname*, *city*) /* keys are underlined

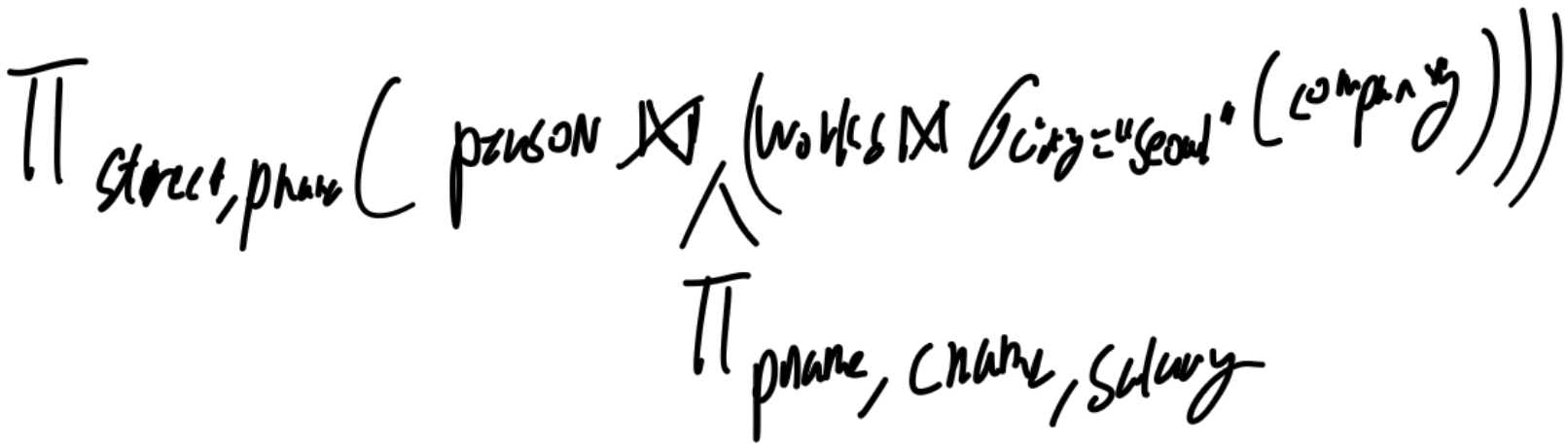
- Using the above database schema, represent the following queries in *relational algebra*.
  - A. *Find all cities in the database* $\Pi_{city}(person) \cup \Pi_{city}(company)$
  - B. *Find the names of people who do not work* $\Pi_{pname}(person) - \Pi_{pname}(works)$
  - C. *Find names of people who work in "SNU" and earn more than 1,000,000*

$$\Pi_{pname}(\sigma_{cname = \text{"SNU"} \wedge salary > 1{,}000{,}000}(works))$$

Discussion 2-7

person (*pname*, *street*, *city*)
works (*pname*, *cname*, *salary*)
company (*cname*, *city*) /* keys are underlined

- Using the above database schema, represent the following queries in *relational algebra*.
  - A. *Find names and addresses of persons who work for "SNU".*
  - B. *Find company names located in the city where "SNU" is located.*
  - C. *Find names and addresses of persons who work for companies located in "Seoul".*



Discussion 2-8

person (*pname*, *street*, *city*)
works (*pname*, *cname*, *salary*)
company (*cname*, *city*) /* keys are underlined

- Using the above database schema, represent the following queries in *relational algebra*.
  A. *Find pairs of person names who live in the same city.*
  B. *Find pairs of person names who live in the same city, without duplicate pairs.*

Discussion 2-9

Discuss why the following property holds for any two relations *r(R)* and *s(S).*

A. $r \bowtie s = s \bowtie r$

B. If $R=S$ then $r \bowtie s = r \cap s$.

Discussion 2-10

- *Use parenthesis to indicate the proper orders of operations in the following relational algebra expression.*

$$\sigma_p\, E_1 \cup E_2 \times E_3 \bowtie \rho_N\, E_3 \cap E_4 - E_5$$

Discussion 2-11

- *Fill the following precedence table for relational algebra operators:* σ, $\cup$, x, $\bowtie$, ρ, $\cap$, $-$, Π

Prec.	Operators	Notes
0	( )	highest
1		
2		
3		
4		

Discussions 2-12

person (*pname*, *street*, *city*)
works (*pname*, *cname*, *salary*)
company (*cname*, *city*)

- Using the above database schema, represent the following queries in *relational algebra*.
  - *Find the name of person with the largest salary.*

END OF CHAPTER 2