DATABASE SYSTEM PRINCIPLE

- RELATIONAL ALGEBRA & CALCULUS

李旭东 **Li-Xudong**

LEEXUDONG@NANKAI.EDU.CN

NANKAI UNIVERSITY

OBJECTIVES

- Relational Algebra
- Tuple Relational Calculus
- Domain Relational Calculus

OBJECTIVES

- Relational Algebra
- Tuple Relational Calculus
- Domain Relational Calculus

RELATIONAL ALGEBRA关系代数

- Procedural language
- Six basic operators
  - select: σ
  - project: $\prod$
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ

The operators take one or two relations as inputs and produce a new relation as a result.

SELECT OPERATION选择运算

- Notation: $\sigma_p(r)$
- *p* is called the **selection predicate**
- Defined as:

$$\sigma_p(\mathbf{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

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

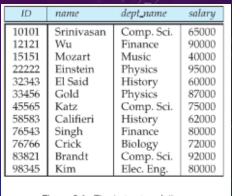
  <attribute> *op* <attribute> or <constant>

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

SELECT OPERATION选择运算

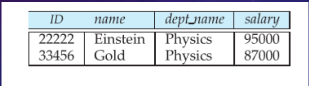
- Example of selection:

$$\sigma_{dept\_name=\text{“Physics”}}(instructor)$$



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

Figure 6.1 The *instructor* relation.



ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
33456	Gold	Physics	87000

Figure 6.2 Result of $\sigma_{dept\_name=\text{“Physics”}}(instructor)$.

PROJECT OPERATION投影运算

- Notation: $\prod_{A_1, A_2, \ldots, A_k}(r)$

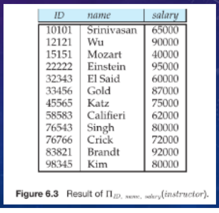
  where A_1, A_2 are attribute names and r is a relation name.
- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets

©LXD

PROJECT OPERATION投影运算

- Example: To eliminate the *dept_name* attribute of *instructor*

  $\prod_{ID,\ name,\ salary}(instructor)$



ID	*name*	*salary*
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	El Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

Figure 6.3 Result of $\Pi_{ID,\ name,\ salary}(instructor)$.

©LXD

UNION OPERATION并运算

- 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.
  1. r, s must have the *same* arity (same number of attributes)
  2. The attribute domains must be compatible (example: 2nd column of r deals with the same type of values as does the 2nd column of s)

©LXD

UNION OPERATION并运算

- Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

  $\Pi_{course\_id}(\sigma_{semester=\text{"Fall"}\ \land\ year=2009}(section)) \cup$

  $\Pi_{course\_id}(\sigma_{semester=\text{"Spring"}\ \land\ year=2010}(section))$

©LXD

SET DIFFERENCE OPERATION集合差运算

- Notation $r - s$
- Defined as:

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

©LXD

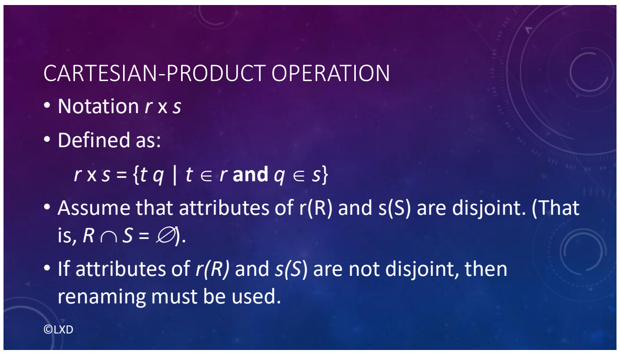
SET DIFFERENCE OPERATION集合差运算

- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

  $\prod_{course\_id}(\sigma_{semester=\text{"Fall"}\ \land\ year=2009}(section)) -$

  $\prod_{course\_id}(\sigma_{semester=\text{"Spring"}\ \land\ year=2010}(section))$

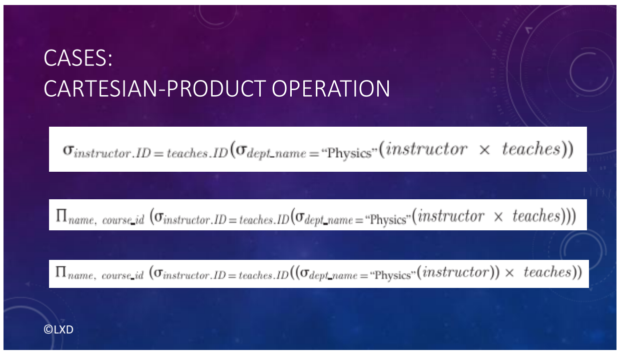
©LXD



CARTESIAN-PRODUCT OPERATION

- Notation *r* x *s*
- Defined as:
  $r \times s = \{t\,q \mid t \in r \textbf{ and } q \in s\}$
- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \varnothing$).
- If attributes of *r(R)* and *s(S)* are not disjoint, then renaming must be used.

©LXD



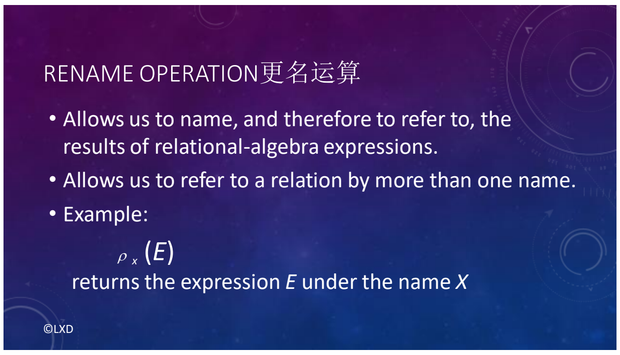
CASES: CARTESIAN-PRODUCT OPERATION

$$\sigma_{instructor.ID = teaches.ID}(\sigma_{dept\_name = \text{“Physics”}}(instructor \times teaches))$$

$$\Pi_{name,\ course\_id}\ (\sigma_{instructor.ID = teaches.ID}(\sigma_{dept\_name = \text{“Physics”}}(instructor \times teaches)))$$

$$\Pi_{name,\ course\_id}\ (\sigma_{instructor.ID = teaches.ID}((\sigma_{dept\_name = \text{“Physics”}}(instructor)) \times teaches))$$

©LXD



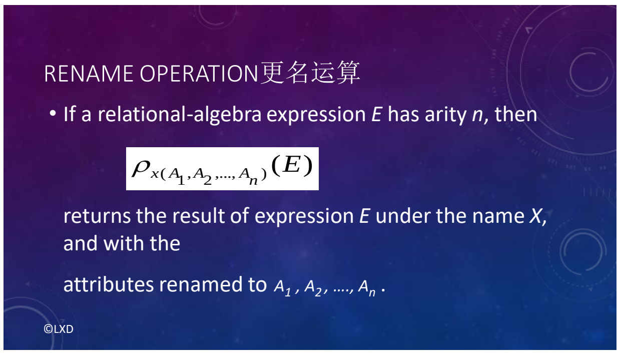
RENAME OPERATION更名运算

- 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.
- Example:

  $\rho_x\,(E)$

  returns the expression *E* under the name *X*

©LXD



RENAME OPERATION更名运算

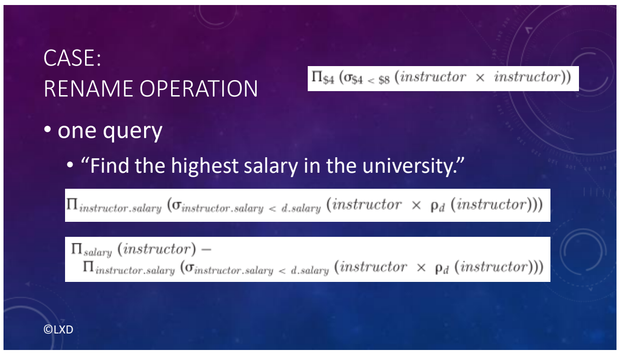
- If a relational-algebra expression *E* has arity *n*, then

  $$\rho_{x(A_1, A_2, ..., A_n)}(E)$$

  returns the result of expression *E* under the name *X*, and with the

  attributes renamed to $A_1, A_2, ..., A_n$.

©LXD



CASE: RENAME OPERATION

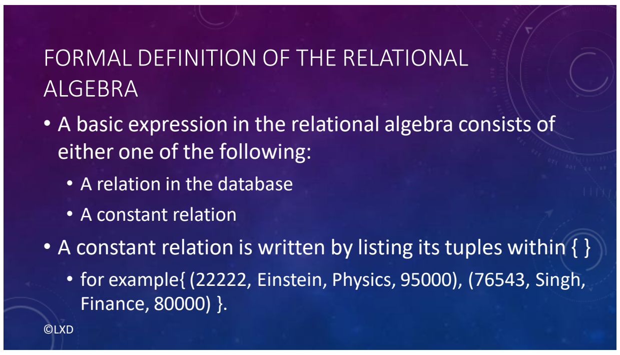
$$\Pi_{\$4}\ (\sigma_{\$4\ <\ \$8}\ (instructor\ \times\ instructor))$$

- one query
  - “Find the highest salary in the university.”

$$\Pi_{instructor.salary}\ (\sigma_{instructor.salary\ <\ d.salary}\ (instructor\ \times\ \rho_d\ (instructor)))$$

$$\begin{aligned}&\Pi_{salary}\ (instructor) - \\ &\quad \Pi_{instructor.salary}\ (\sigma_{instructor.salary\ <\ d.salary}\ (instructor\ \times\ \rho_d\ (instructor)))\end{aligned}$$

©LXD



FORMAL DEFINITION OF THE RELATIONAL ALGEBRA

- A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- A constant relation is written by listing its tuples within { }
  - for example{ (22222, Einstein, Physics, 95000), (76543, Singh, Finance, 80000) }.

©LXD

FORMAL DEFINITION OF THE RELATIONAL ALGEBRA

- A general expression in the relational algebra is constructed out of smaller subexpressions.
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_P(E_1)$, where P is a predicate on attributes in E_1
  - $\Pi_S(E_1)$, where S is a list consisting of some of the attributes in E_1
  - $\rho_x(E_1)$, where x is the new name for the result of E_1

©LXD

ADDITIONAL RELATIONAL-ALGEBRA OPERATIONS

- The Set-Intersection Operation
- The Natural-Join Operation
- The Assignment Operation
- Outer join Operations

©LXD

SET-INTERSECTION OPERATION集合交运算

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r$ **and** $t \in s \}$
- Assume:
  - *r*, *s* have the *same arity*
  - attributes of *r* and *s* are compatible
- Note: $r \cap s = r - (r - s)$

©LXD

THE NATURAL-JOIN OPERATION

$$\Pi_{name,\ course\_id}\ (instructor \bowtie teaches)$$

$$r \bowtie s = \Pi_{R \cup S}\ (\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \ldots \wedge r.A_n = s.A_n}\ (r \times s))$$

where $R \cap S = \{A_1, A_2, \ldots, A_n\}$

$R \cap S = \emptyset$, then $r \bowtie s = r \times s$

©LXD

THETA JOIN: A VARIANT OF THE NATURAL-JOIN OPERATION

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

let θ be a predicate on attributes in the schema $R \cup S$

©LXD

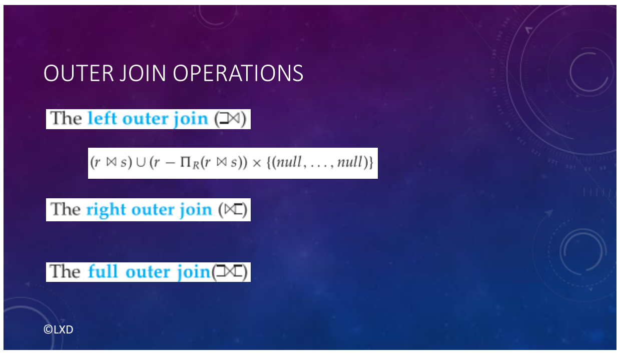
THE ASSIGNMENT OPERATION赋值运算

The **assignment** operation, denoted by $\leftarrow$

$r \bowtie s$

$$\begin{aligned} temp1 &\leftarrow R \times S \\ temp2 &\leftarrow \sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \ldots \wedge r.A_n = s.A_n}\ (temp1) \\ result &= \Pi_{R \cup S}\ (temp2) \end{aligned}$$

©LXD



OUTER JOIN OPERATIONS

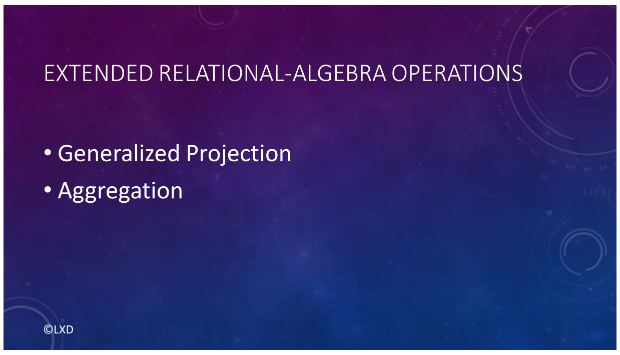
The **left outer join** ($⟕$)

$$(r \bowtie s) \cup (r - \Pi_R(r \bowtie s)) \times \{(null, \ldots, null)\}$$

The **right outer join** ($⟖$)

The **full outer join**($⟗$)

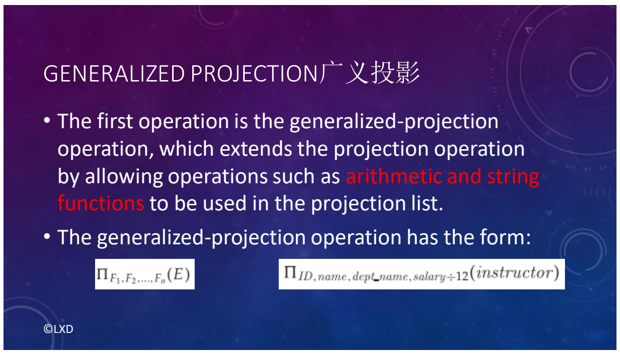
©LXD



EXTENDED RELATIONAL-ALGEBRA OPERATIONS

- Generalized Projection
- Aggregation

©LXD

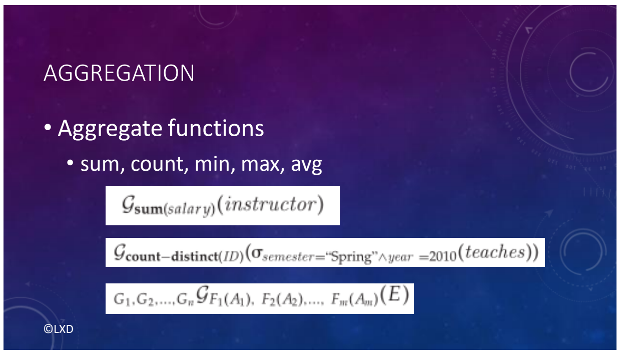


GENERALIZED PROJECTION广义投影

- The first operation is the generalized-projection operation, which extends the projection operation by allowing operations such as arithmetic and string functions to be used in the projection list.
- The generalized-projection operation has the form:

$\Pi_{F_1,F_2,\ldots,F_n}(E)$ $\Pi_{ID, name, dept\_name, salary \div 12}(instructor)$

©LXD



AGGREGATION

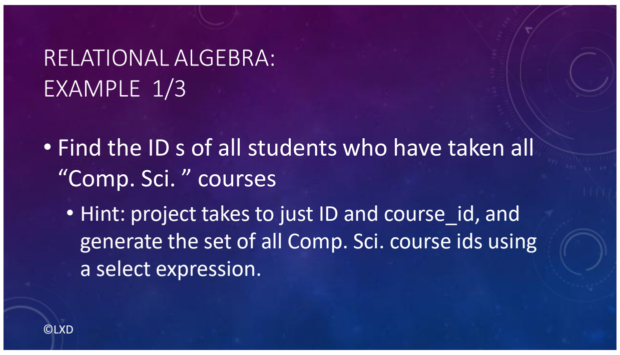
- Aggregate functions
  - sum, count, min, max, avg

$$\mathcal{G}_{\mathbf{sum}(salary)}(instructor)$$

$$\mathcal{G}_{\mathbf{count-distinct}(ID)}(\sigma_{semester=\text{“Spring”} \wedge year\ =2010}(teaches))$$

$$_{G_1,G_2,\ldots,G_n}\mathcal{G}_{F_1(A_1),\ F_2(A_2),\ldots,\ F_m(A_m)}(E)$$

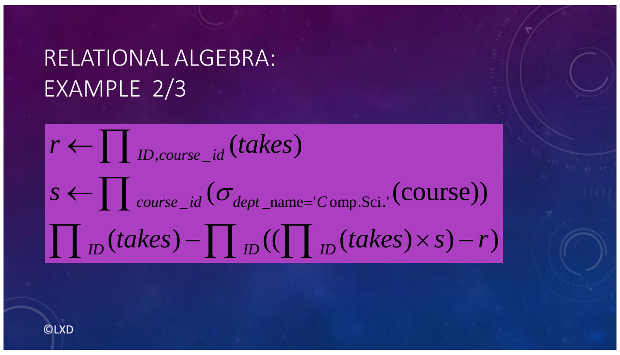
©LXD



RELATIONAL ALGEBRA: EXAMPLE 1/3

- Find the ID s of all students who have taken all “Comp. Sci. ” courses
  - Hint: project takes to just ID and course_id, and generate the set of all Comp. Sci. course ids using a select expression.

©LXD



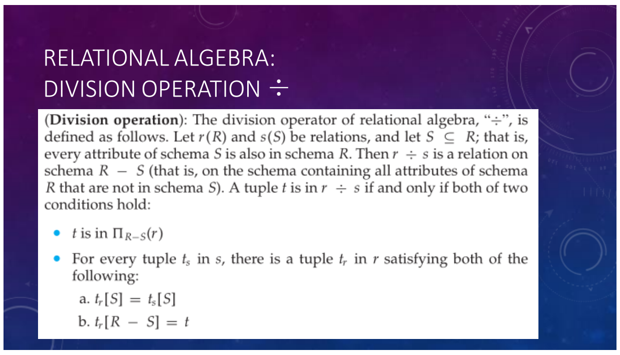
RELATIONAL ALGEBRA: EXAMPLE 2/3

$$r \leftarrow \prod_{ID, course\_id}(takes)$$

$$s \leftarrow \prod_{course\_id}(\sigma_{dept\_name='Comp.Sci.'}(\text{course}))$$

$$\prod_{ID}(takes) - \prod_{ID}((\prod_{ID}(takes) \times s) - r)$$

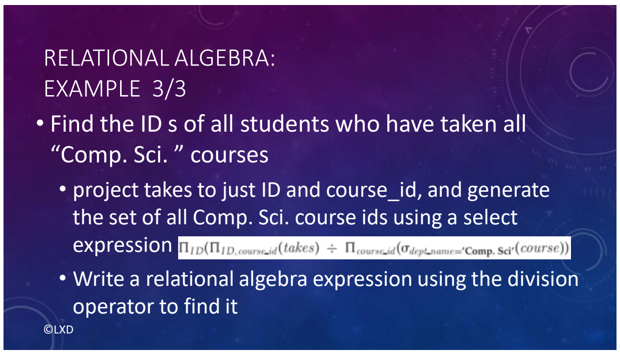
©LXD



RELATIONAL ALGEBRA: DIVISION OPERATION ÷

(**Division operation**): The division operator of relational algebra, "÷", is defined as follows. Let $r(R)$ and $s(S)$ be relations, and let $S \subseteq R$; that is, every attribute of schema S is also in schema R. Then $r \div s$ is a relation on schema $R - S$ (that is, on the schema containing all attributes of schema R that are not in schema S). A tuple t is in $r \div s$ if and only if both of two conditions hold:

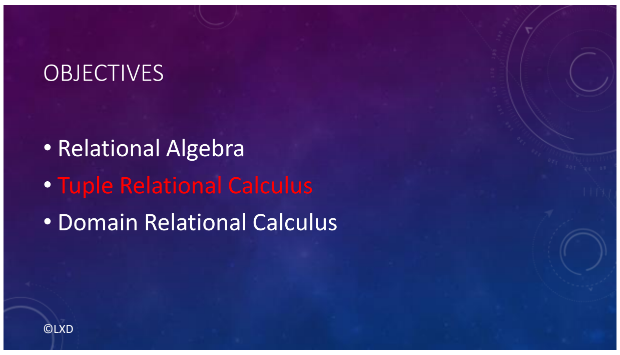
- t is in $\Pi_{R-S}(r)$
- For every tuple t_s in s, there is a tuple t_r in r satisfying both of the following:
  a. $t_r[S] = t_s[S]$
  b. $t_r[R - S] = t$



RELATIONAL ALGEBRA: EXAMPLE 3/3

- Find the ID s of all students who have taken all "Comp. Sci. " courses
  - project takes to just ID and course_id, and generate the set of all Comp. Sci. course ids using a select expression $\Pi_{ID}(\Pi_{ID, course\_id}(takes) \div \Pi_{course\_id}(\sigma_{dept\_name='Comp. Sci'}(course))$
  - Write a relational algebra expression using the division operator to find it

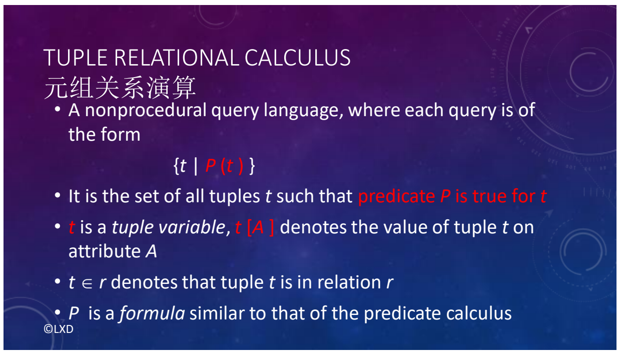
©LXD



OBJECTIVES

- Relational Algebra
- Tuple Relational Calculus
- Domain Relational Calculus

©LXD



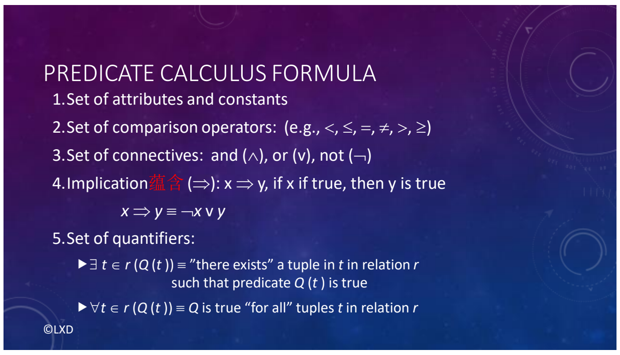
TUPLE RELATIONAL CALCULUS 元组关系演算

- A nonprocedural query language, where each query is of the form

  {*t* | *P* (*t*) }

- It is the set of all tuples *t* such that predicate *P* is true for *t*
- *t* is a *tuple variable*, *t* [*A*] denotes the value of tuple *t* on attribute *A*
- *t* ∈ *r* denotes that tuple *t* is in relation *r*
- *P* is a *formula* similar to that of the predicate calculus

©LXD



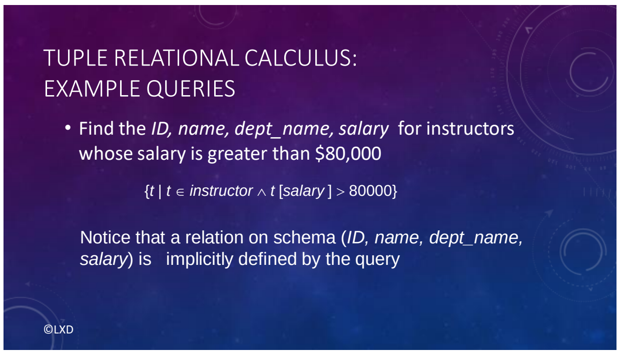
PREDICATE CALCULUS FORMULA

1. Set of attributes and constants
2. Set of comparison operators: (e.g., <, ≤, =, ≠, >, ≥)
3. Set of connectives: and (∧), or (v), not (¬)
4. Implication蕴含 (⇒): x ⇒ y, if x if true, then y is true

   x ⇒ *y* ≡ ¬*x* v *y*

5. Set of quantifiers:
   - ▶∃ *t* ∈ *r* (*Q* (*t*)) ≡ "there exists" a tuple in *t* in relation *r* such that predicate *Q* (*t*) is true
   - ▶∀*t* ∈ *r* (*Q* (*t*)) ≡ *Q* is true "for all" tuples *t* in relation *r*

©LXD



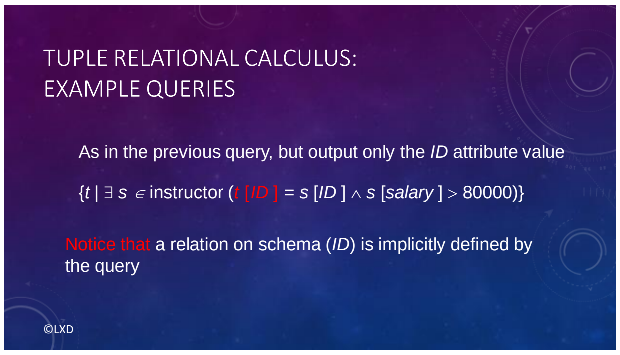
TUPLE RELATIONAL CALCULUS: EXAMPLE QUERIES

- Find the *ID, name, dept_name, salary* for instructors whose salary is greater than $80,000

  {*t* | *t* ∈ *instructor* ∧ *t* [*salary*] > 80000}

  Notice that a relation on schema (*ID, name, dept_name, salary*) is implicitly defined by the query

©LXD



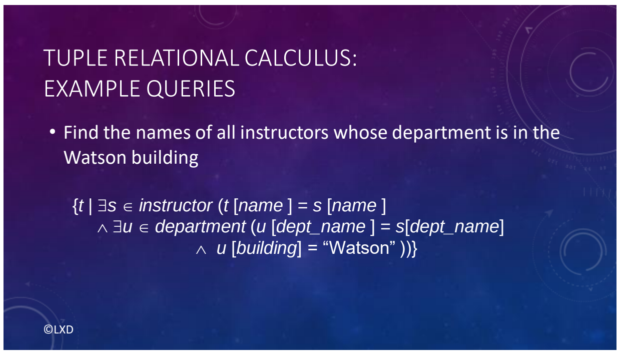
TUPLE RELATIONAL CALCULUS: EXAMPLE QUERIES

As in the previous query, but output only the *ID* attribute value

$\{t \mid \exists\, s \in \text{instructor}\ (t\,[ID\,] = s\,[ID\,] \wedge s\,[salary\,] > 80000)\}$

Notice that a relation on schema (*ID*) is implicitly defined by the query

©LXD

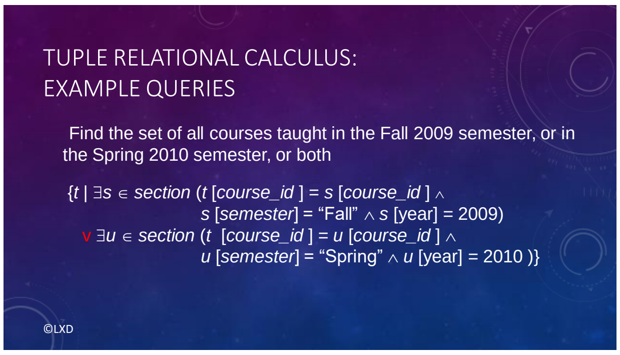


TUPLE RELATIONAL CALCULUS: EXAMPLE QUERIES

- Find the names of all instructors whose department is in the Watson building

$\{t \mid \exists s \in instructor\ (t\,[name\,] = s\,[name\,]$
$\wedge\ \exists u \in department\ (u\,[dept\_name\,] = s[dept\_name]$
$\wedge\ u\,[building] = \text{“Watson”}\ ))\}$

©LXD

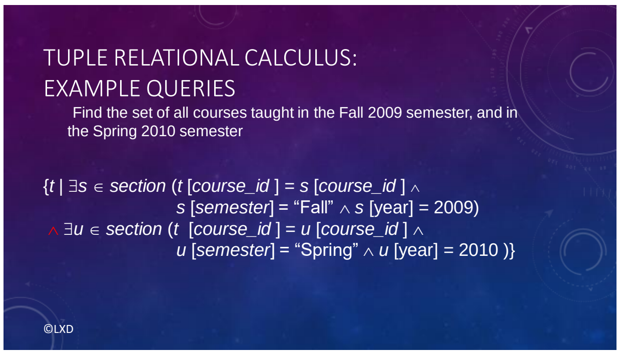


TUPLE RELATIONAL CALCULUS: EXAMPLE QUERIES

Find the set of all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

$\{t \mid \exists s \in section\ (t\,[course\_id\,] = s\,[course\_id\,] \wedge$
$s\,[semester] = \text{“Fall”} \wedge s\,[year] = 2009)$
$\vee\ \exists u \in section\ (t\ \ [course\_id\,] = u\,[course\_id\,] \wedge$
$u\,[semester] = \text{“Spring”} \wedge u\,[year] = 2010\ )\}$

©LXD

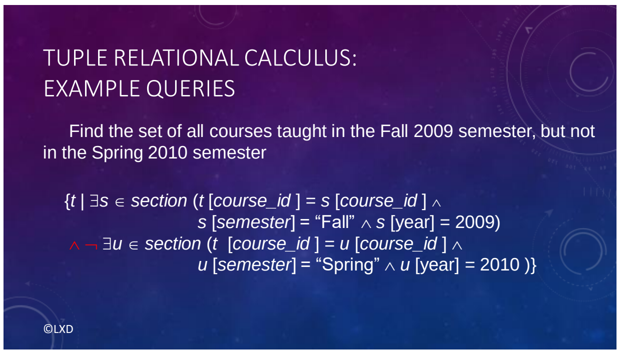


TUPLE RELATIONAL CALCULUS: EXAMPLE QUERIES

Find the set of all courses taught in the Fall 2009 semester, and in the Spring 2010 semester

$\{t \mid \exists s \in section\ (t\,[course\_id\,] = s\,[course\_id\,] \wedge$
$s\,[semester] = \text{“Fall”} \wedge s\,[year] = 2009)$
$\wedge\ \exists u \in section\ (t\ \ [course\_id\,] = u\,[course\_id\,] \wedge$
$u\,[semester] = \text{“Spring”} \wedge u\,[year] = 2010\ )\}$

©LXD

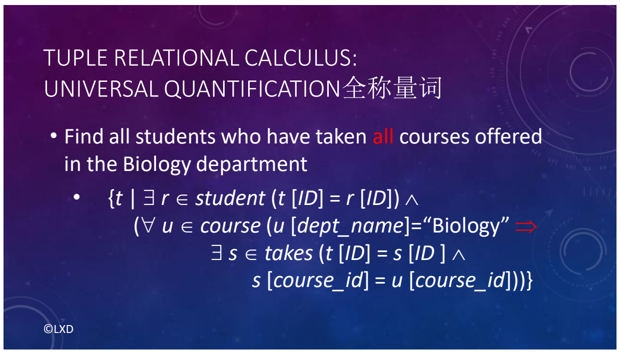


TUPLE RELATIONAL CALCULUS: EXAMPLE QUERIES

Find the set of all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$\{t \mid \exists s \in section\ (t\,[course\_id\,] = s\,[course\_id\,] \wedge$
$s\,[semester] = \text{“Fall”} \wedge s\,[year] = 2009)$
$\wedge \neg \exists u \in section\ (t\ \ [course\_id\,] = u\,[course\_id\,] \wedge$
$u\,[semester] = \text{“Spring”} \wedge u\,[year] = 2010\ )\}$

©LXD



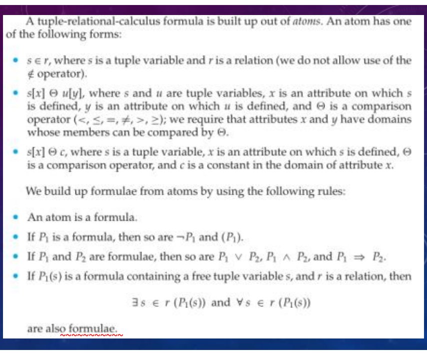
TUPLE RELATIONAL CALCULUS: UNIVERSAL QUANTIFICATION全称量词

- Find all students who have taken all courses offered in the Biology department
  - $\{t \mid \exists\ r \in student\ (t\ [ID] = r\ [ID]) \wedge$
    $(\forall\ u \in course\ (u\ [dept\_name]=\text{“Biology”} \Rightarrow$
    $\exists\ s \in takes\ (t\ [ID] = s\ [ID\ ] \wedge$
    $s\ [course\_id] = u\ [course\_id]))\}$

©LXD

FORMAL DEFINITION (形式化定义): TUPLE RELATIONAL CALCULUS

{*t* | *P* (*t*) }


A tuple-relational-calculus formula is built up out of *atoms*. An atom has one of the following forms:

- $s \in r$, where s is a tuple variable and r is a relation (we do not allow use of the $\notin$ operator).
- $s[x]\ \Theta\ u[y]$, where s and u are tuple variables, x is an attribute on which s is defined, y is an attribute on which u is defined, and Θ is a comparison operator ($<, \leq, =, \neq, >, \geq$); we require that attributes x and y have domains whose members can be compared by Θ.
- $s[x]\ \Theta\ c$, where s is a tuple variable, x is an attribute on which s is defined, Θ is a comparison operator, and c is a constant in the domain of attribute x.

We build up formulae from atoms by using the following rules:

- An atom is a formula.
- If P_1 is a formula, then so are $\neg P_1$ and (P_1).
- If P_1 and P_2 are formulae, then so are $P_1 \vee P_2$, $P_1 \wedge P_2$, and $P_1 \Rightarrow P_2$.
- If $P_1(s)$ is a formula containing a free tuple variable s, and r is a relation, then

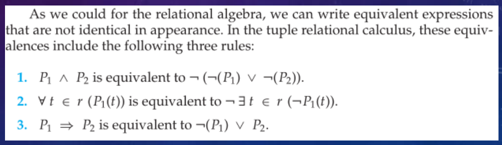
$$\exists s \in r\ (P_1(s)) \text{ and } \forall s \in r\ (P_1(s))$$

are also formulae.

©LXD

FORMAL DEFINITION: TUPLE RELATIONAL CALCULUS

{*t* | *P* (*t*) }


As we could for the relational algebra, we can write equivalent expressions that are not identical in appearance. In the tuple relational calculus, these equivalences include the following three rules:

1. $P_1 \wedge P_2$ is equivalent to $\neg(\neg(P_1) \vee \neg(P_2))$.
2. $\forall t \in r\ (P_1(t))$ is equivalent to $\neg \exists t \in r\ (\neg P_1(t))$.
3. $P_1 \Rightarrow P_2$ is equivalent to $\neg(P_1) \vee P_2$.

©LXD

SAFETY OF EXPRESSIONS表达式的安全性

- It is possible to write tuple calculus expressions that generate infinite无限多的 relations.
- For example
  - { t | ¬ *t* ∈ *r* }
  - It results in an infinite relation if the domain of any attribute of relation *r* is infinite

©LXD

SAFETY OF EXPRESSIONS表达式的安全性

- To guard against the problem, we restrict the set of allowable expressions to safe expressions.
  - Introduce the concept Domain(域) of P

©LXD

SAFETY OF EXPRESSIONS表达式的安全性

- Domain(域) of P
  - dom(P) is the set of all values referenced by P
  - They include values mentioned in P itself, as well as values that appear in a tuple of a relation mentioned in P.
- Example:
  - dom(t ∈ instructor ∧ t[salary] > 80000)
  - is the set containing 80000 as well as the set of all values appearing in any attribute of any tuple in the instructor relation.

©LXD

SAFETY OF EXPRESSIONS

- An expression {*t* | *P* (*t*)} in the tuple relational calculus is *safe* if every component of *t* appears in one of the relations, tuples, or constants that appear in dom(*P)*
  - {t |¬ (t ∈ instructor)} is not safe
  - E.g. { *t* | *t* [*A*] = 5 ∨ **true** } is not safe --- it defines an infinite set with attribute values that do not appear in any relation or tuples or constants in *P*.

©LXD

SAFETY OF EXPRESSIONS

- Safety? Consider again that query to find all students who have taken all courses offered in the Biology department
  - {*t* | ∃ *r* ∈ *student* (*t* [*ID*] = *r* [*ID*]) ∧
    (∀ *u* ∈ *course* (*u* [*dept_name*]="Biology" ⇒
    ∃ *s* ∈ *takes* (*t* [*ID*] = *s* [*ID*] ∧
    s [*course_id*] = *u* [*course_id*]))}

Without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.

©LXD

EXERCISES 1

Let the following relation schemas be given:

R = (A, B,C)

S = (D, E, F)

Let relations r(R) and s(S) be given. Give an expression in the tuple relational calculus that is equivalent to each of the following:

a) $\prod_A(r)$ **b)** $\sigma_{B=17}(r)$ **c)** $\prod_{A,F}\sigma_{C=D}(r \times s)$

©LXD

OBJECTIVES

- Relational Algebra
- Tuple Relational Calculus
- Domain Relational Calculus

©LXD

DOMAIN RELATIONAL CALCULUS 域关系演算

- A nonprocedural query language equivalent in power to the tuple relational calculus
- domain variables that take on values from an attributes domain, rather than values for an entire tuple.
- Each query is an expression of the form:

  {< x_1, x_2, ..., x_n > | *P* (x_1, x_2, ..., x_n)}

x_1, x_2, ..., x_n represent domain variables
  - *P* represents a formula similar to that of the predicate calculus

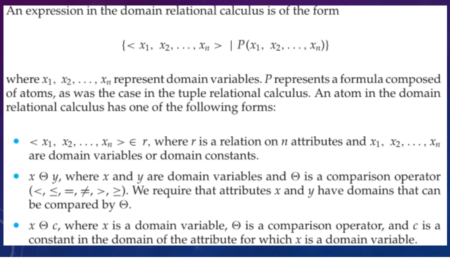
©LXD

DOMAIN RELATIONAL CALCULUS: EXAMPLE QUERIES

- Find the *ID, name, dept_name, salary* for instructors whose salary is greater than $80,000
  - {< *i, n, d, s*> | < *i, n, d, s*> ∈ *instructor* ∧ *s* > 80000}
- As in the previous query, but output only the *ID* attribute value
  - {< *i*> | < *i, n, d, s*> ∈ *instructor* ∧ *s* > 80000}

©LXD

DOMAIN RELATIONAL CALCULUS: FORMAL DEFINITION形式化定义 1/2



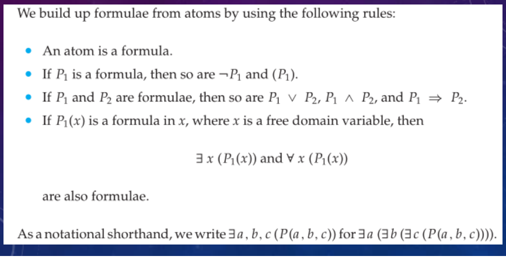
An expression in the domain relational calculus is of the form

$$\{< x_1,\ x_2, \ldots, x_n > \ \mid P(x_1,\ x_2, \ldots, x_n)\}$$

where $x_1,\ x_2, \ldots, x_n$ represent domain variables. *P* represents a formula composed of atoms, as was the case in the tuple relational calculus. An atom in the domain relational calculus has one of the following forms:

- $< x_1,\ x_2, \ldots, x_n > \in r$, where *r* is a relation on *n* attributes and $x_1,\ x_2, \ldots, x_n$ are domain variables or domain constants.
- $x\ \Theta\ y$, where *x* and *y* are domain variables and Θ is a comparison operator ($<, \leq, =, \neq, >, \geq$). We require that attributes *x* and *y* have domains that can be compared by Θ.
- $x\ \Theta\ c$, where *x* is a domain variable, Θ is a comparison operator, and *c* is a constant in the domain of the attribute for which *x* is a domain variable.

DOMAIN RELATIONAL CALCULUS: FORMAL DEFINITION形式化定义 2/2


We build up formulae from atoms by using the following rules:

- An atom is a formula.
- If P_1 is a formula, then so are $\neg P_1$ and (P_1).
- If P_1 and P_2 are formulae, then so are $P_1 \vee P_2$, $P_1 \wedge P_2$, and $P_1 \Rightarrow P_2$.
- If $P_1(x)$ is a formula in x, where x is a free domain variable, then

$$\exists\, x\ (P_1(x)) \text{ and } \forall\, x\ (P_1(x))$$

are also formulae.

As a notational shorthand, we write $\exists\, a, b, c\ (P(a, b, c))$ for $\exists\, a\ (\exists\, b\ (\exists\, c\ (P(a, b, c))))$.

DOMAIN RELATIONAL CALCULUS: MORE EXAMPLE QUERIES

Find the set of all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

$\{<c> \mid \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t > \in section \wedge$
$s = \text{“Fall”} \wedge y = 2009\ )$
$\vee \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t > \in section\ ] \wedge$
$s = \text{“Spring”} \wedge y = 2010)\}$

This case can also be written as

$\{<c> \mid \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t > \in section \wedge$
$(\ (s = \text{“Fall”} \wedge y = 2009\ )\ \vee (s = \text{“Spring”} \wedge y =$
$2010))\}$

©LXD

DOMAIN RELATIONAL CALCULUS: MORE EXAMPLE QUERIES

Find the set of all courses taught in the Fall 2009 semester, and in
the Spring 2010 semester

$\{<c> \mid \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t > \in section \wedge$
$s = \text{“Fall”} \wedge y = 2009\ )$
$\wedge \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t > \in section\ ] \wedge$
$s = \text{“Spring”} \wedge y = 2010)\}$

©LXD

DOMAIN RELATIONAL CALCULUS: SAFETY OF EXPRESSIONS

- The following expression is unsafe
  - $\{< i,n,d,s > \mid \neg(< i,n,d,s > \in instructor)\}$

©LXD

DOMAIN RELATIONAL CALCULUS: SAFETY OF EXPRESSIONS

The expression:

$\{ < x_1, x_2, \ldots, x_n > \mid P\ (x_1, x_2, \ldots, x_n)\}$

is safe if all of the following hold:

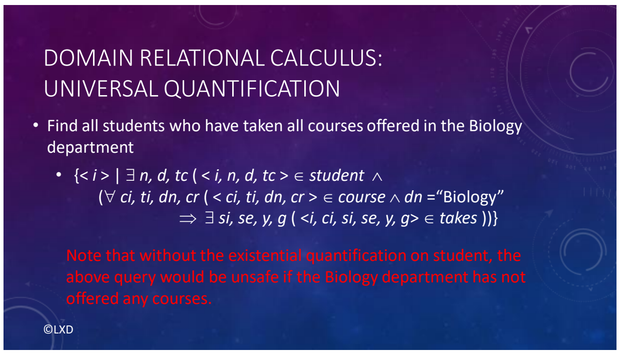
1. All values that appear in tuples of the expression are values from *dom* (P) (that is, the values appear either in P or in a tuple of a relation mentioned in P).
2. (cont.,)

©LXD

DOMAIN RELATIONAL CALCULUS: SAFETY OF EXPRESSIONS

2. For every “there exists” subformula of the form $\exists\, x\ (P_1(x))$, the subformula is true if and only if there is a value of x in *dom* (P_1) such that $P_1(x)$ is true.
3. For every “for all” subformula of the form $\forall_x\ (P_1\ (x))$, the subformula is true if and only if $P_1(x)$ is true for all values x from *dom* (P_1).

©LXD

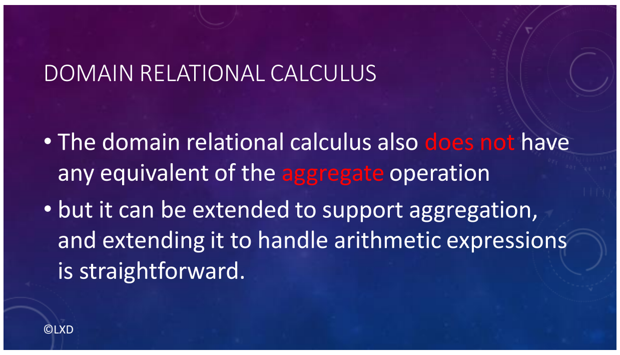


DOMAIN RELATIONAL CALCULUS: UNIVERSAL QUANTIFICATION

- Find all students who have taken all courses offered in the Biology department
  - $\{<i> \mid \exists\, n, d, tc\ (<i, n, d, tc> \in student\ \wedge$
    $(\forall\, ci, ti, dn, cr\ (<ci, ti, dn, cr> \in course \wedge dn =$ "Biology"
    $\Rightarrow \exists\, si, se, y, g\ (<i, ci, si, se, y, g> \in takes\ ))\}$

Note that without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.

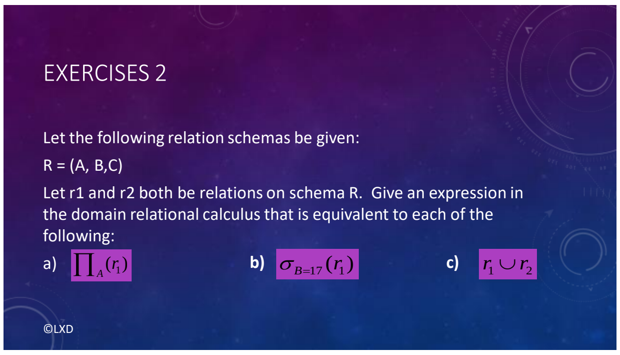
©LXD



DOMAIN RELATIONAL CALCULUS

- The domain relational calculus also does not have any equivalent of the aggregate operation
- but it can be extended to support aggregation, and extending it to handle arithmetic expressions is straightforward.

©LXD



EXERCISES 2

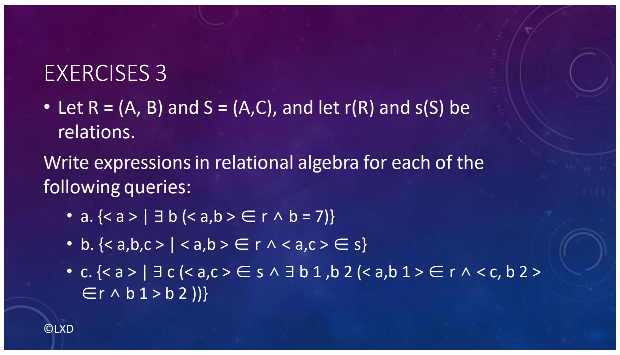
Let the following relation schemas be given:

R = (A, B,C)

Let r1 and r2 both be relations on schema R. Give an expression in the domain relational calculus that is equivalent to each of the following:

a) $\prod_A(r_1)$ **b)** $\sigma_{B=17}(r_1)$ **c)** $r_1 \cup r_2$

©LXD



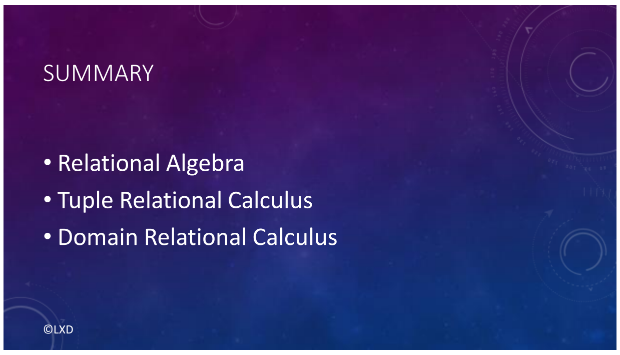
EXERCISES 3

- Let R = (A, B) and S = (A,C), and let r(R) and s(S) be relations.

Write expressions in relational algebra for each of the following queries:

- a. $\{<a> \mid \exists\, b\ (<a,b> \in r \wedge b = 7)\}$
- b. $\{<a,b,c> \mid <a,b> \in r \wedge <a,c> \in s\}$
- c. $\{<a> \mid \exists\, c\ (<a,c> \in s \wedge \exists\, b1, b2\ (<a,b1> \in r \wedge <c, b2> \in r \wedge b1 > b2\ ))\}$

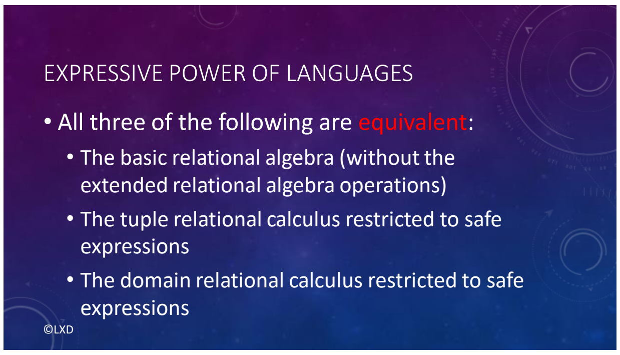
©LXD



SUMMARY

- Relational Algebra
- Tuple Relational Calculus
- Domain Relational Calculus

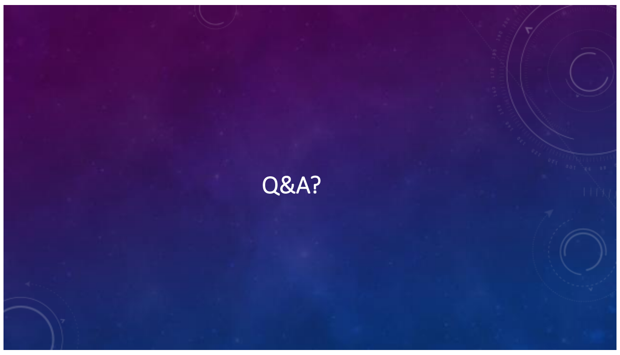
©LXD



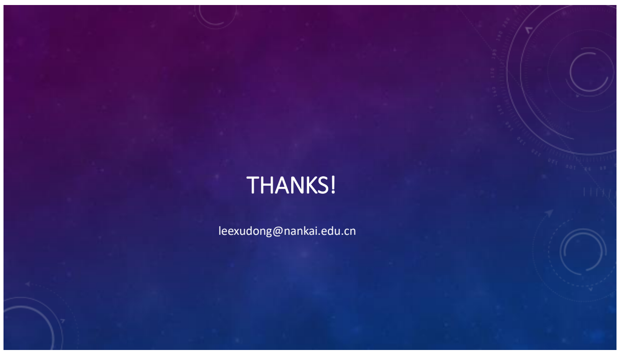
EXPRESSIVE POWER OF LANGUAGES

- All three of the following are equivalent:
  - The basic relational algebra (without the extended relational algebra operations)
  - The tuple relational calculus restricted to safe expressions
  - The domain relational calculus restricted to safe expressions

©LXD



Q&A?



THANKS!

leexudong@nankai.edu.cn