COP 4710: Database Systems
Spring 2007

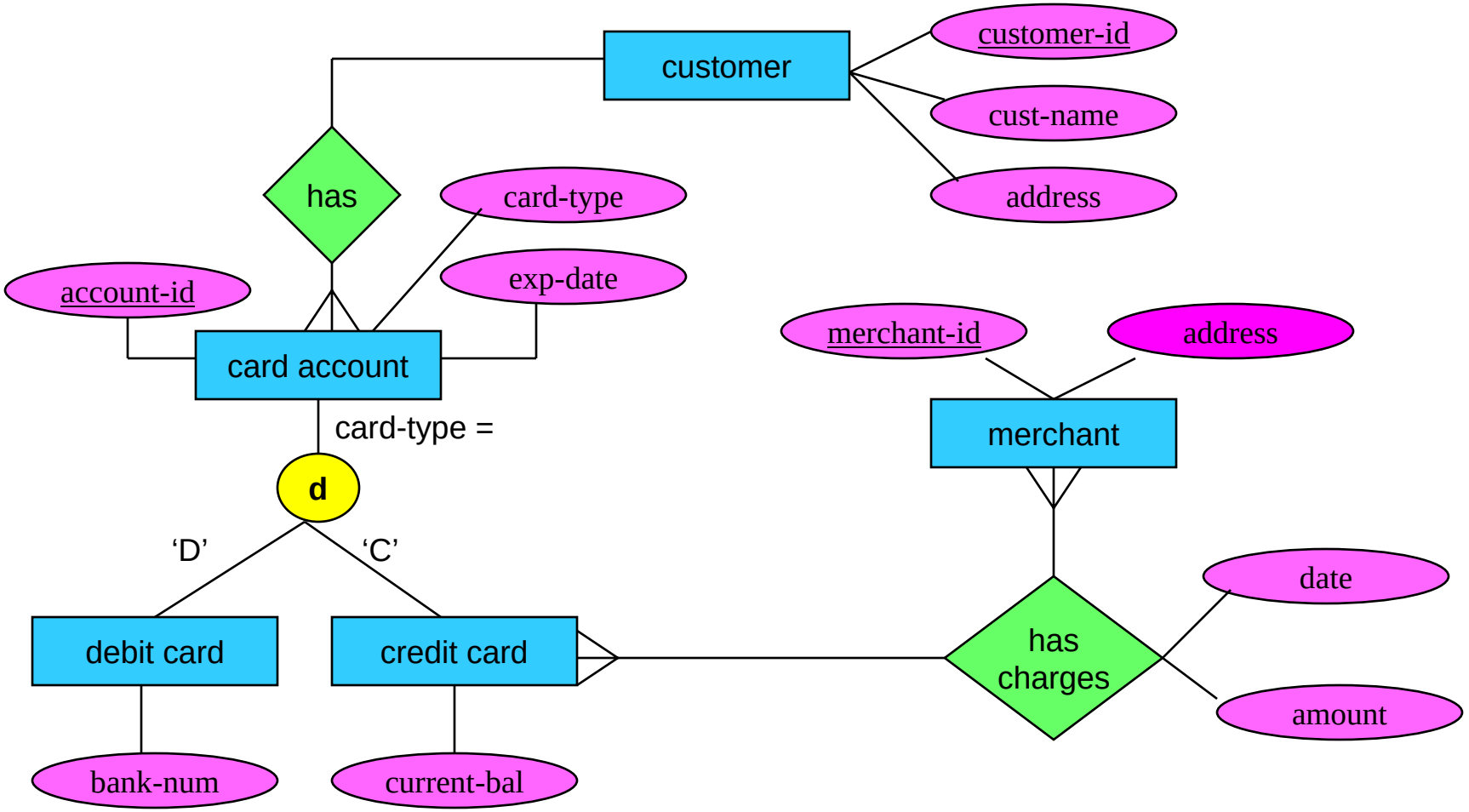
Chapter 4 – Relational Query Languages – Part 1

Instructor : Mark Llewellyn
markl@cs.ucf.edu
ENG3 236, 823-2790
http://www.cs.ucf.edu/courses/cop4710/spr2007

School of Electrical Engineering and Computer Science
University of Central Florida

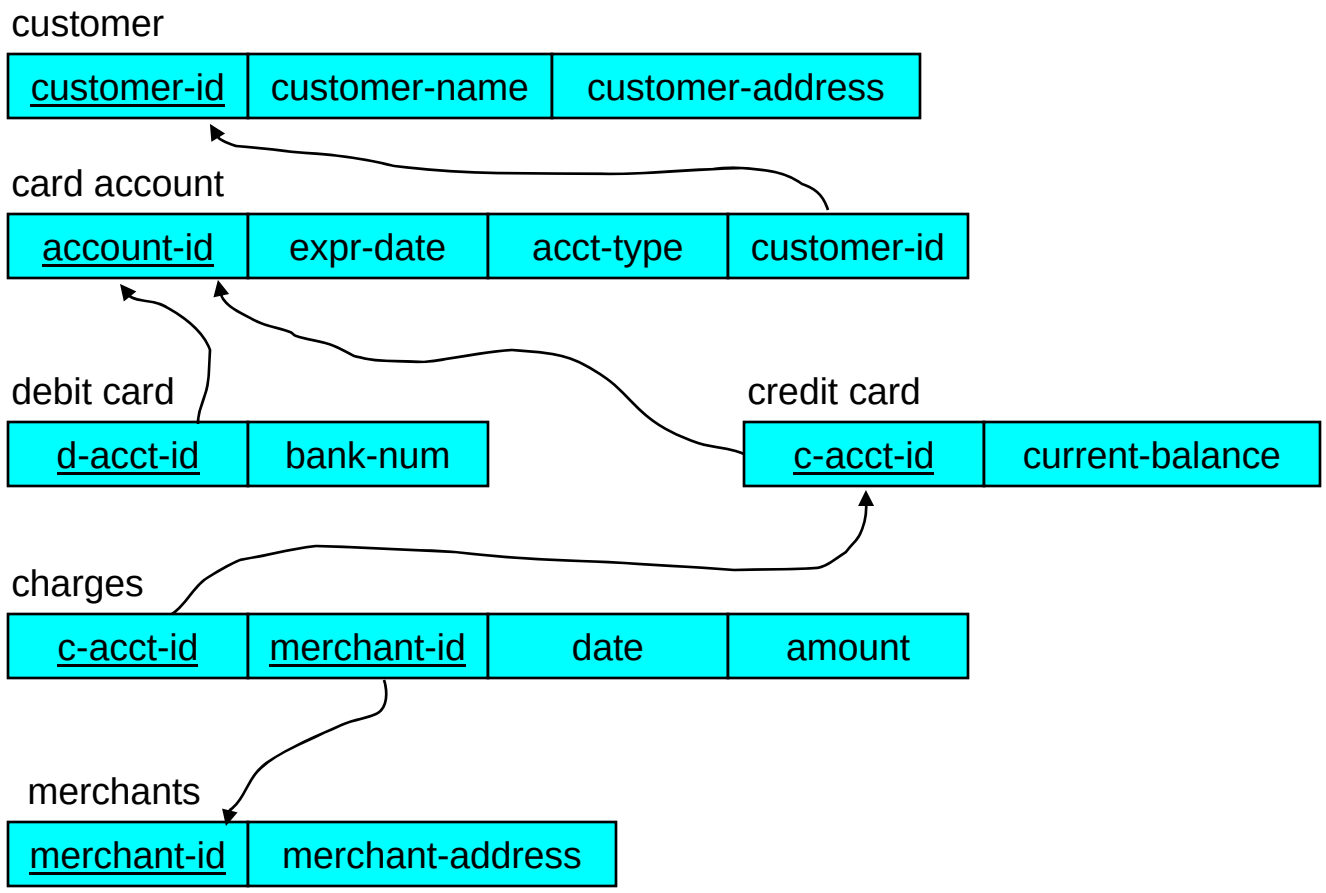
Mapping E-R Diagrams to Relational Schemas

PRACTICE problem from previous notes



Mapping E-R Diagrams to Relational Schemas

SOLUTION to practice problem



Query Languages

- A query language is a language in which a database user requests information from the database.
  - Most query languages are on a higher-level than standard programming languages like C and Java. Query languages fall into a category of languages known as 4GL.
- Query languages can be broadly categorized into two groups: procedural languages and nonprocedural languages.
  - A procedural query language requires the user to specify a sequence of operations on the db to compute the desired result. (User specifies how and what.)
  - A nonprocedural query language requires the user to describe the desired result without needing to specify the sequence of operations required to obtain the result. (User specifies only what.)

Query Languages (cont.)

- Most commercially available relational database systems offer a query language which is categorized as a hybrid query language.
- Hybrid query languages include elements of both the procedural and nonprocedural approaches to query languages.
- For the time being we are going to examine "pure" relational query languages. These languages are terse and formal and lack many of the syntactic elements available in commercial languages, but they illustrate the fundamental techniques utilized by all query languages for extracting data from the database.

Query Languages (cont.)

- As we examine these pure languages, bear in mind that, although the pure languages do not contain such features, a complete query language has facilities for inserting and deleting tuples from relations as well as for modifying existing tuples.
- Procedural language:
  1. relational algebra
- Nonprocedural languages:
  1. relational tuple calculus
  2. relational domain calculus

Relational Algebra

- The *relational algebra* is a procedural query language. It consists of set operations which are either unary or binary, meaning that either one or two relations are operands to the set operations.
- Each of the set operations produces a relation as its output.
- There are five fundamental operations in the relational algebra and several additional operations which are defined in terms of the five fundamental operations.
- There is also a *rename* operation which is sometimes referred to as a fundamental operation, we'll save this one for a little while.
- The five fundamental operations are: *select, project, union, set difference,* and *Cartesian product*. We will examine each operation individually before combining operations into more powerful expressions.

Relational Algebra (cont.)

- The five fundamental operations are: *select, project, union, set difference*, and *Cartesian product*.
- There are several additional (redundant) operations that have been defined in the relational algebra. The most common of these include: intersection, natural join, division, semi-join, and outer join.
- We will examine each operation individually before combining operations into more powerful expressions.

Selection Operator

Type: unary
Symbol: Greek letter sigma, σ
General form: $\sigma_{(predicate)}$(relation instance)
Schema of result relation: same as operand relation
Size of result relation (tuples): $\leq$ |operand relation|
Examples:
$\sigma_{(major = \text{“CS”})}$(students)
$\sigma_{(major = \text{“CS” and hair-color = “brown”})}$(students)
$\sigma_{(hours\text{-}attempted > hours\text{-}earned)}$(students)

- The select operation selects tuples from a relation instance which satisfy a specified predicate.
- In general, a predicate, may contain any of the logical comparative operators, which are =, $\neq$, <, $\leq$, >, $\geq$. Furthermore, several predicates may be combined using the connectives *and* ($\wedge$), *or* ($\vee$), and *not* ($\neg$).
- The select operation may be thought of as providing a horizontal cross-section of the operand relation.

Selection Operator Examples

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6
a	c	no	7
b	b	no	69
c	a	yes	24
d	d	yes	47
h	d	yes	34
e	c	no	26
a	a	yes	5

$r = \sigma_{(A = 'a')}(R)$

A	B	C	D
a	a	yes	1
a	d	no	6
a	c	no	7
a	a	yes	5

$r = \sigma_{(A = 'a' \wedge C = \text{"yes"})}(R)$

A	B	C	D
a	a	yes	1
a	a	yes	5

$r = \sigma_{(B = 'm')}(R)$

A	B	C	D

an empty relation

Projection Operator

Type: unary
Symbol: Greek letter pi, π
General form: $\pi_{(\text{attribute-list})}(\text{relation instance})$
Schema of result relation: specified by <attribute-list>
Size of result relation (tuples): $\leq |$ operand relation $|$
Examples:
$\pi_{(\text{student-id, name, major})}(\text{students})$
$\pi_{(\text{name, advisor})}(\text{students})$
$\pi_{(\text{name, gpa, hours-attempted})}(\text{students})$

- The project operation can be viewed as producing a vertical cross-section of the operand relation.
- If the operation produces duplicate tuples, these are typically removed from the result relation in keeping with its set-like characteristics.

Projection Operator Examples

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6
a	c	no	7
b	b	no	69
c	a	yes	24
d	d	yes	47
h	d	yes	34
e	c	no	26
a	a	yes	5

$r = \pi_{(A, C)}(R)$

A	C
a	yes
b	no
c	yes
a	no
d	yes
h	yes
e	no

$r = \pi_{(A, D)}(R)$

A	D
a	1
b	7
c	34
a	6
a	7
b	69
c	24
d	47
h	34
e	26
a	5

$r = \pi_{(C)}(R)$

C
yes
no

Union Operator

Type: binary
Symbol: union symbol, $\cup$
General form: $r \cup s$, where *r* and *s* are union compatible
Schema of result relation: schema of operand relations
Size of result relation (tuples): $\leq \max\{ |r| + |s| \}$
Examples:

$r \cup s$ $\qquad\qquad$ $\pi_{(a, b)}(r) \cup \pi_{(a, b)}(s)$

- The union operation provides a means for extracting information which resides in two operand relations which must be *union compatible*. Union compatibility requires that two conditions hold:

1. Relations *r(R)* and *s(S)* in the expression $r \cup s$ must be of the same degree (*arity)*. That is, they must have the same number of attributes.

2. The domains of the *i*th attribute of *r(R)* and the *i*th attributes of *s(S)* must be the same, for all *i*.

Union Operator Examples

R

A	B	D
a	a	1
b	d	7
c	f	34
a	d	6
a	c	7

T

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

r = R ∪ T

not valid – R and T are not union compatible

r = R ∪ S

E	F	G
a	a	1
b	d	7
c	f	34
a	d	6
a	c	7
a	m	4
b	c	22
a	d	16

S

X	Y	Z
a	m	4
b	c	22
a	d	16
a	c	7

X

A	B
a	a
b	d
a	c

r = T ∪ X

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

Set Difference Operator

Type: binary
Symbol: –
General form: r – s, where *r* and *s* are union compatible
Schema of result relation: schema of operand relation
Size of result relation (tuples): ≤ | relation r |
Examples: r – s

- The set difference operation allows for the extraction of information contained in one relation that is not contained in a second relation.
- As with the union operation, the set difference operation requires that the two operand relations be union compatible.

Set Difference Operator Examples

R

A	B	D
a	a	1
b	d	7
c	f	34
a	d	6
a	c	7

T

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

r = R – T

not valid – R and T are not union compatible

r = R – S

E	F	G
a	a	1
b	d	7
c	f	34
a	d	6

r = T – X

A	B
c	f
a	d

S

X	Y	Z
a	m	4
b	c	22
a	d	16
a	c	7

X

A	B
a	a
b	d
a	c

r = X – T

A	B

empty relation

r = S – R

E	F	G
a	m	4
b	c	22
a	d	16

Cartesian Product Operator

Type: binary
Symbol: $\times$
General form: $r \times s$ (no restrictions on *r* and *s*)
Schema of result relation: schema *r* $\times$ schema *s* with renaming
Size of result relation (tuples): > | relation *r* | and > | relation *s* |
Examples:

$r \times s$

- The Cartesian product operation allows for the combining of any two relations into a single relation.
- Recall that a relation is by definition a subset of a Cartesian product of a set of domains, so this gives you some idea of the behavior of the Cartesian product operation.

Cartesian Product Operator Examples

T

A	B
a	a
b	d

X

A	B
a	a
b	d
a	c
c	a

r = T × X

T.A	T.B	X.A	X.B
a	a	a	a
a	a	b	d
a	a	a	c
a	a	c	a
b	d	a	a
b	d	b	d
b	d	a	c
b	d	c	a

Cartesian Product Operator Examples

R

A	B	C	D
a	a	1	yes
b	d	7	yes
c	f	34	no

S

X	Y	Z
a	m	4
b	c	22
a	d	16
a	c	7

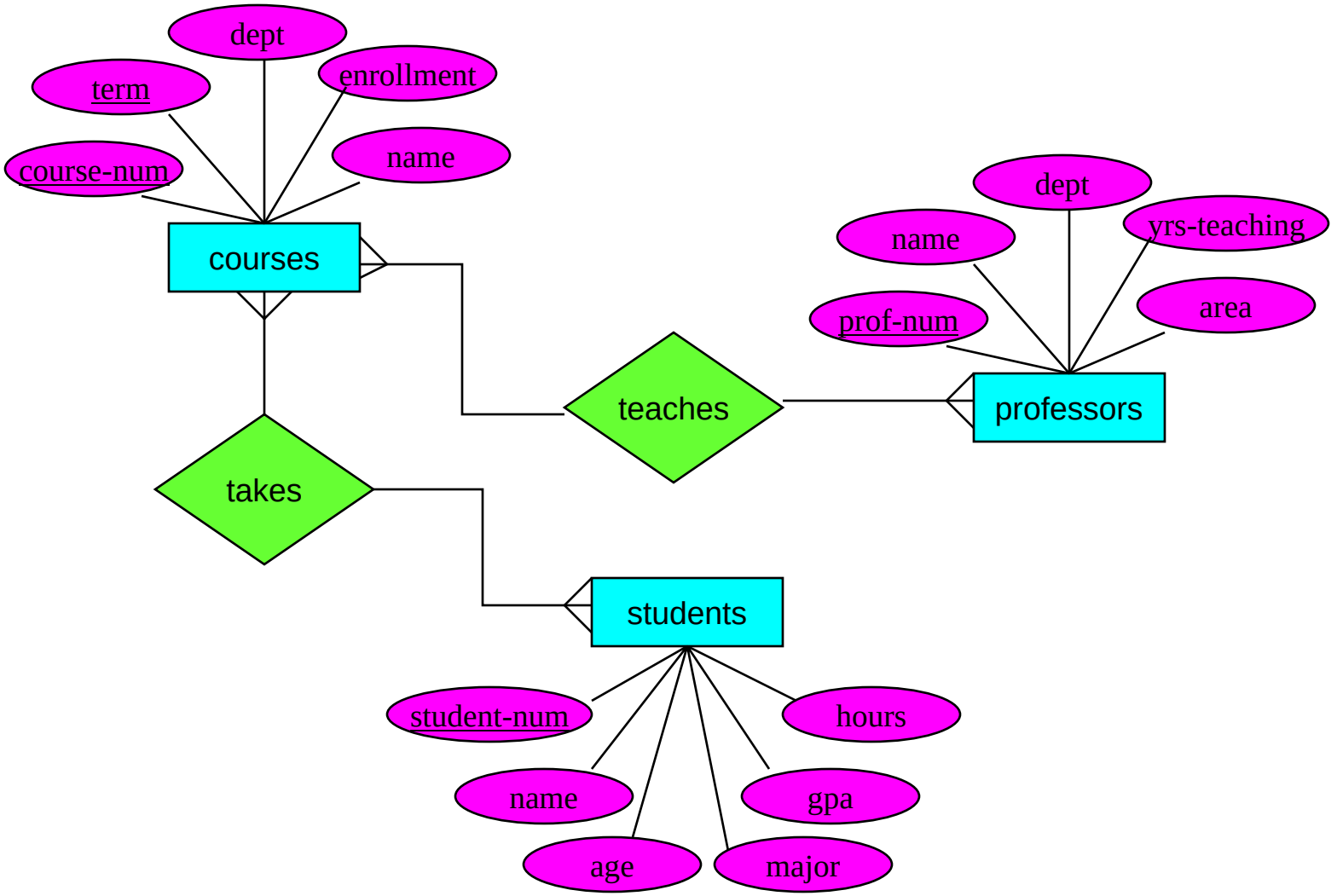
r = R × S

A	B	C	D	X	Y	Z
a	a	1	yes	a	m	4
a	a	1	yes	b	c	22
a	a	1	yes	a	d	16
a	a	1	yes	a	c	7
b	d	7	yes	a	m	4
b	d	7	yes	b	c	22
b	d	7	yes	a	d	16
b	d	7	yes	a	c	7
c	f	34	no	a	m	4
c	f	34	no	b	c	22
c	f	34	no	a	d	16
c	f	34	no	a	c	7

Relational Algebra Expressions

- While each of the five fundamental relational algebra operators can be used individually to form a query, their expressive power is tremendously enhanced when they are combined together to form query expressions.
- Before we introduce the redundant operations in relational algebra we'll look at forming more complicated combinations of the five fundamental operations. [This will also make you appreciate the redundant operations all the more.]
- To form meaningful queries we need to be able to pose them against a database. For all of the examples that follow, we'll use the following database:

Relational Algebra Expressions (cont.)



Relational Algebra Expressions (cont.)

- Using the techniques for converting an ERD into a set of relational schemas we have the following resulting schemas:

  S = STUDENTS(s#, name, age, major, gpa, hours_completed)

  C = COURSES(c#, term, name, dept, enrollment)

  P = PROFESSORS(p#, name, dept, yrs_teaching, area)

  TA = TAKES(s#, c#, term, grade)

  TE = TEACH(p#, c#, term)

- When you first begin to write queries in a new query language, it is sometimes helpful to actually visualize the data that might be in one of the operand (argument) relations upon which you are operating. To this end, the last two pages of this set of notes provides an instance of each of the relations above so that you can perform this visualization, however, this is something that you will need to move away from as you get more advanced in your query composition, because you do not want to influence the design of your query by visualizing a relation instance that may not contain all possible tuples that your query will encounter during execution.

Relational Algebra Expressions (cont.)

Example Query 1:

Find the names of all the students who are Computer Science majors.

Approach:

- First select all of the students who are CS majors.

$$r = \sigma_{(major = \text{“Computer Science”})}(S)$$

- Next project only the name attribute from the previous result.

$$result = \pi_{(name)}(r)$$

Complete Query Expression:

$$result = \pi_{(name)}(\sigma_{(major = \text{“Computer Science”})}(S))$$

Relational Algebra Expressions (cont.)

Example Query 2:

Find the student-num (s#) and name of all the students who have completed more than 90 hours.

Approach:

- First select all of the students who have completed more than 90 hours.

$$r = \sigma_{(hours\_completed > 90)}(S)$$

- Next project the student-num and name attributes from the previous result.

$$result = \pi_{(s\#, name)}(r)$$

Complete Query Expression:

$$result = \pi_{(s\#, name)}(\sigma_{(hours\_completed > 90)}(S))$$

Relational Algebra Expressions (cont.)

Example Query 3:

Find the names of all those students who are less than 20 years old who have completed more than 80 hours.

Approach:

- First select all of the students who have completed more than 80 hours and are less than 20 years old.

$$r = \sigma_{((hours\_completed > 80)\ AND\ (age < 20))}(S)$$

- Next project the name attribute from the previous result.

$$result = \pi_{(name)}(r)$$

Complete Query Expression:

$$result = \pi_{(name)}(\sigma_{((hours\_completed > 80)\ AND\ (age < 20))}(S))$$

Relational Algebra Expressions (cont.)

Example Query 4:

Find the names of all the courses that are offered by either Computer Science or Physics.

Approach:

- First select all of the courses that are offered by either CS or Physics.

$$r = \sigma_{((dept = Computer\ Science)\ or\ (dept = Physics))}(C))$$

- Next project the name attribute from the previous result.

$$result = \pi_{(name)}(r)$$

Complete Query Expression:

$$result = \pi_{(name)}(\sigma_{((dept = Computer\ Science)\ or\ (dept = Physics))}(C))$$

Relational Algebra Expressions (cont.)

Example Query 5:

Find the name of every professor who taught a course in the Fall 2006 term.

Approach:

- First put the professor information together with the course information.
- Next, select only related professors and courses from previous result.
- Finally, select only the students name from the previous result.

Complete Query Expression:

$$result = \pi_{(P.name)}(\sigma_{((TE.term = Fall\ 2006)\ AND\ (P.p\# = TE.p\#))}(P \times TE))$$

Relational Algebra Expressions (cont.)

Example Query 6:

Find the names of all the students who took a course in the Fall 2006 term that was taught by a professor who had more than 20 years of teaching experience.

Approach:

- First put the professor information together with the course information together with the teaches information together with the takes information.
- Next, select only related students, professors and courses from previous result.
- Finally, select only the students name from the previous result.

Complete Query Expression:

$$result = \pi_{(S.name)}(\sigma_{((TA.term = Fall\ 2006)\ AND\ (P.yrs\_teaching > 20)\ AND\ (S.s\# = TA.s\#)\ AND\ (P.p\# = TE.p\#)\ AND\ (TA.c\# = TE.c\#)\ AND\ (TA.term = TE.term))}(S \times P \times TA \times TE))$$

Relational Algebra Expressions (cont.)

Example Query 7:

Find the names of all the professors who are either in the Computer Science department or have more than 20 years of teaching experience.

Complete Query Expression:

$$result = [\pi_{(name)}(\sigma_{(dept = Computer\ Science)}(P))] \cup [\pi_{(name)}(\sigma_{(yrs\_teaching > 20)}(P))]$$

or:

$$result = \pi_{(name)}(\sigma_{((dept = Computer\ Science)\ OR\ (yrs\_teaching > 20))}(P))$$

Relational Algebra Expressions (cont.)

Example Query 8:

Find the student numbers for those students who were enrolled only in the Spring 2007 term.

Complete Query Expression:

$$result = [\pi_{(TA.s\#)}(\sigma_{(TA.term = Spring\ 2007)}(TA))] - [\pi_{(TA.s\#)}(\sigma_{(TA.term \neq Spring\ 2007))}(TA))]$$

Note: The following query expression is not correct for this query!!! Why?

$$result = \pi_{(TA.s\#)}(\sigma_{(TA.term = Spring\ 2007))}(TA))$$

Sample Relation Instances: S relation

s#	name	age	major	gpa	hrs_completed
S1	Michael Schumacher	19	Computer Science	4.00	45
S5	Jean Alesi	20	Physics	3.46	78
S3	Rubens Barrichello	21	Math	3.82	33
S2	Giancarlo Fisichella	18	Math	2.73	23
S4	Jarno Trulli	18	Computer Science	1.48	99
S7	Bernd Schneider	19	Computer Science	2.29	45
S6	Mika Hakkinen	20	English	2.37	33

Sample Relation Instances: C relation

c#	term	name	dept	enrollment
C1	Fall 2002	CS1	CS	120
C1	Spring 2002	CS1	CS	100
C4	Fall 2002	Architecture	CS	97
C3	Fall 2002	Database	CS	86
C5	Spring 2001	Physics I	Physics	135
C5	Fall 2002	Physics I	Physics	125
C6	Summer 2002	Calculus III	Math	67

Sample Relation Instances: P relation

p#	name	dept	yrs_teaching
P1	Wilson	CS	5
P2	Davis	Math	32
P3	deMoser	CS	17
P4	Roberts	Physics	14

Sample Relation Instances: TA relation

s#	c#	term	grade
S1	C3	Fall 03	A
S3	C4	Fall 02	B
S4	C6	Summer 03	C
S5	C5	Spring 01	D
S5	C1	Fall 02	A
S5	C3	Fall 03	C
S5	C6	Summer 02	C
S5	C4	Fall 03	A
S3	C5	Spring 01	C
S3	C1	Fall 03	A
S2	C4	Fall 03	D

Sample Relation Instances: TE relation

p#	c#	term
P1	C3	Fall 2002
P3	C4	Fall 2002
P4	C6	Summer 2002
P2	C5	Spring 2001
P2	C1	Spring 2002
P1	C4	Fall 2002
P3	C1	Fall 2002