Relational Algebra Part II

Hugh Darwen

hugh@dcs.warwick.ac.uk
www.dcs.warwick.ac.uk/~hugh

CS252.HACD: Fundamentals of Relational Databases
Section 5: Relational Algebra, Part II

1

The Running Example

IS_CALLED

StudentId	Name
S1	Anne
S2	Boris
S3	Cindy
S4	Devinder
S5	Boris

IS_ENROLLED_ON

StudentId	CourseId
S1	C1
S1	C2
S2	C1
S3	C3
S4	C1

StudentId is called Name

StudentId is enrolled on CourseId

2

Special Case of AND (1)

StudentId is called Boris

Can be done using JOIN and projection, like this:

(IS_CALLED JOIN
RELATION { TUPLE { Name NAME ('Boris') } })
{ StudentId }

but it's easier using *restriction* (and projection again):

(IS_CALLED **WHERE** Name = NAME ('Boris')) { StudentId }

result:

StudentId
S2
S5

"EXISTS Name such that StudentId is called Name AND Name is Boris" 3

A More Useful Restriction

Sid1 has the same name as Sid2 (**AND** Sid2 ≠ Sid1).

((IS_CALLED RENAME (StudentId AS Sid1))
JOIN
(IS_CALLED RENAME (StudentId AS Sid2)))
WHERE Sid1 < Sid2) { Sid1, Sid2 }

Result:

Sid1	Sid2
S2	S5

Hopelessly difficult using JOIN instead of WHERE! (Why?)

4

Definition of Restriction

Let *s* = *r* **WHERE** *c,* where *c* is a conditional expression on attributes of *r.*

The heading of *s* is the heading of *r.*

The body of *s* consists of those tuples of *r* for which the condition *c* evaluates to TRUE.

So the body of *s* is a subset of that of *r.*

5

Special Cases of Restriction

What is the result of R WHERE TRUE?

R

What is the result of R WHERE FALSE?

The empty relation with the heading of R.

6

Special Case of AND (2)

StudentId is called Name **AND** Name begins with the letter Initial.

Given:

StudentId	Name
S1	Anne
S2	Boris
S3	Cindy
S4	Devinder
S5	Boris

To obtain:

StudentId	Name	Initial
S1	Anne	A
S2	Boris	B
S3	Cindy	C
S4	Devinder	D
S5	Boris	B

Again, much too difficult with JOIN.

7

Extension

StudentId is called Name **AND** Name begins with the letter Initial.

EXTEND IS_CALLED **ADD**
(SUBSTRING (Name, 0, 1) AS Initial)

Result:

StudentId	Name	Initial
S1	Anne	A
S2	Boris	B
S3	Cindy	C
S4	Devinder	D
S5	Boris	B

8

Definition of Extension

Let *s* = **EXTEND** *r* **ADD** (*formula-1* AS *A1*, … *formula-n* AS *An*)

The heading of *s* consists of the attributes of the heading of *r* plus the attributes *A1* … *An*. The declared type of attribute *Ak* is that of *formula-k.*

The body of *s* consists of tuples formed from each tuple of *r* by **add**ing *n* additional attributes *A1* to *An*. The value of attribute *Ak* is the result of evaluating *formula-k* on the corresponding tuple of *r*.

9

Two More Relvars

COURSE

CourseId	Title
C1	Database
C2	HCI
C3	Op Systems
C4	Programming

EXAM_MARK

StudentId	CourseId	Mark
S1	C1	85
S1	C2	49
S2	C1	49
S3	C3	66
S4	C1	93

CourseId is entitled Title

StudentId scored Mark in the exam for course CourseId

10

Relations within a Relation

CourseId	Exam_Result
C1	StudentId: S1, Mark: 85; StudentId: S2, Mark: 49; StudentId: S4, Mark: 93
C2	StudentId: S1, Mark: 49
C3	StudentId: S3, Mark: 66
C4	StudentId, Mark

Call this C_ER for future reference.

The declared type of the Exam_Result attribute is RELATION { StudentId SID, Mark INTEGER }

11

To obtain C_ER from COURSE and EXAM_MARK:

```
EXTEND COURSE ADD (
        ( EXAM_MARK JOIN
          RELATION { TUPLE { CourseId CourseId } } )
          { ALL BUT CourseId }
         AS Exam_Result )
{ CourseId, Exam_Result }
```

12

An Aggregate Operator

An aggregate operator is one defined to operate on a relation and return a value obtained by aggregation over all the tuples of the operand. For example, simply to count the tuples:

COUNT (IS_ENROLLED_ON) = 5
COUNT (IS_ENROLLED_ON
WHERE CourseId = CID ('C1')) = 3

COUNT is an aggregate operator.

13

More Aggregate Operators

SUM (EXAM_MARK, Mark) = 342

AVG (EXAM_MARK, Mark) = 68.4

MAX (EXAM_MARK, Mark) = 93

MIN (EXAM_MARK, Mark) = 49

MAX (EXAM_MARK
WHERE CourseId = CID ('C2'), Mark) = 49

14

Nested Relations and Agg Ops

The top score in the exam on course CourseId was TopScore

CourseId	TopScore
C1	93
C2	49
C3	66

EXTEND C_ER WHERE COUNT (Exam_Result) > 0
ADD (MAX (Exam_Result, Mark) AS TopScore)
{ CourseId, TopScore }

15

SUMMARIZE BY

A shorthand for aggregation over nested relations. For example, those top scores in each exam can be obtained directly from EXAM_MARK by:

SUMMARIZE EXAM_MARK **BY** { CourseId }
ADD (MAX (Mark) AS TopScore)

The usual first operand of the "agg op" is now omitted because it is implied by the combination of the SUMMARIZE operand (EXAM_MARK) and the BY operand ({CourseId }).

16

SUMMARIZE PER

Takers is how many people took the exam on course CourseId

SUMMARIZE EXAM_MARK **PER** COURSE { CourseId }
ADD (COUNT() AS Takers)

result:

CourseId	Takers
C1	3
C2	1
C3	1
C4	0

Note that EXAM_MARK **BY** { CourseId } is shorthand for EXAM_MARK **PER** EXAM_MARK { CourseId }.

17

OR

StudentId is called Name **OR** StudentId is enrolled on CourseId.

StudentId	Name	CourseId
S1	Anne	C1
S1	Boris	C1
S1	Zorba	C1
S1	Anne	C4
S1	Anne	C943

and so on *ad infinitum* (almost!)

NOT SUPPORTED!

18

UNION (restricted OR)

StudentId is called Devinder **OR** StudentId is enrolled on C1.

StudentId
S1
S2
S4

(IS_CALLED WHERE Name = NAME ('Devinder')) { StudentId }
UNION
(IS_ENROLLED_ON WHERE CourseId = CID ('C1')) { StudentId }

19

Definition of UNION

Let *s* = *r1* **UNION** *r2*. Then:

r1 and *r2* must have the same heading.

The heading of *s* is the common heading of *r1* and *r2*.

The body of *s* consists of each tuple that is *either* a tuple of *r1* *or* a tuple of *r2*.

Is UNION commutative? Is it associative?

20

NOT

StudentId is **NOT** called Name

StudentId	Name
S1	Boris
S1	Quentin
S1	Zorba
S1	Cindy
S1	Hugh

and so on *ad infinitum* (almost!)

NOT SUPPORTED!

21

Restricted NOT

StudentId is called Name **AND** is **NOT** enrolled on any course.

StudentId	Name
S5	Boris

IS_CALLED **NOT MATCHING** IS_ENROLLED_ON

22

Definition of NOT MATCHING

Let *s* = *r1* **NOT MATCHING** *r2*. Then:

The heading of *s* is the heading of *r1*.

The body of *s* consists of each tuple of *r1* that matches no tuple of *r2* on their common attributes.

It follows that in the case where there are no common attributes, *s* is equal to *r1* if *r2* is empty, and otherwise is empty .

23

MINUS

Codd defined *r1* **MINUS** *r2* instead of *r1* NOT MATCHING *r2*.

MINUS is set difference and requires *r1* and *r2* to have the same heading (as in *r1* UNION *r2*).

Most textbooks follow Codd and do not even define NOT MATCHING, in spite of its greater generality.

Either can be defined in terms of the other.

Tutorial D supports both, for historical reasons only.

24