Relational Algebra, Principles and Part I

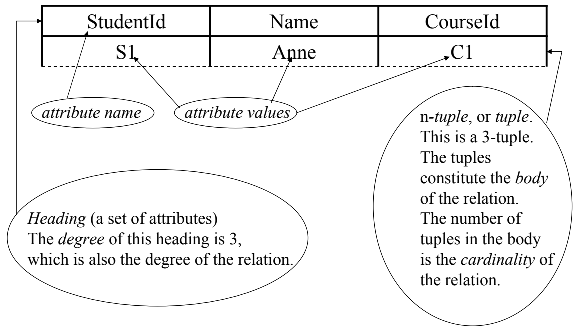
Hugh Darwen

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

CS252.HACD: Fundamentals of Relational Databases
Section 4: Relational Algebra, Principles and Part I

1

Anatomy of a Relation



2

ENROLMENT Example

ENROLMENT (a relation variable, or *relvar*)

StudentId	Name	CourseId
S1	Anne	C1
S1	Anne	C2
S2	Boris	C1
S3	Cindy	C3
S4	Devinder	C1

Predicate: StudentId is called Name and is enrolled on CourseId
Note *redundancy*: S1 is *always* called Anne!

3

Splitting ENROLMENT

IS_CALLED

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

Student StudentId is called Name

IS_ENROLLED_ON

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

Student StudentId is enrolled on course CourseId

4

Relations and Predicates (1)

Consider the predicate: StudentId is called Name

… is called --- is the *intension* (meaning) of the predicate.

The parameter names are arbitrary. "S is called N" means the same thing (has the same intension).

The *extension* of the predicate is the set of *true* propositions that are *instantiations* of it:
{ S1 is called Anne, S2 is called Boris, S3 is called Cindy, S4 is called Devinder, S5 is called Boris }

Each tuple in the body (extension) of the relation provides the values to substitute for the parameters in one such instantiation.

5

Relations and Predicates (2)

Moreover, each proposition in the extension has exactly one corresponding tuple in the relation.

This 1:1 correspondence reflects the *Closed-World Assumption*:

A tuple representing a true instantiation is in the relation.
A tuple representing a false one is out.

The Closed-World Assumption underpins the operators we are about to meet.

6

Relational Algebra

Operators that operate on relations and return relations.

In other words, operators that are *closed over* relations. Just as arithmetic operators are closed over numbers.

Closure means that every invocation can be an operand, allowing expressions of arbitrary complexity to be written. Just as, in arithmetic, e.g., the invocation b-c is an operand of a+(b-c).

The operators of the relational algebra are relational counterparts of *logical* operators: AND, OR, NOT, EXISTS. Each, when invoked, yields a relation, which can be interpreted as the extension of some predicate.

7

Logical Operators

Because relations are used to represent predicates, it makes sense for relational operators to be counterparts of operators on predicates. We will meet examples such as these:

Student StudentId is called Name **AND** StudentId is enrolled on course CourseId.

Student StudentId is enrolled **on some course**.

Student StudentId is enrolled on course CourseId AND StudentId is **NOT** called Devinder.

Student StudentId is NOT enrolled on any course **OR** StudentId is called Boris.

8

Meet The Operators

Logic	Relational counterpart
AND	JOIN restriction (WHERE) extension SUMMARIZE *and some more*
EXISTS	projection
OR	UNION
(AND) NOT	(semi)difference
	RENAME

9

JOIN (= AND)

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

IS_CALLED **JOIN** IS_ENROLLED_ON

IS_CALLED

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

IS_ENROLLED_ON

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

10

IS_CALLED JOIN IS_ENROLLED_ON

StudentId	Name	CourseId
S1	Anne	C1
S1	Anne	C2
S2	Boris	C1
S3	Cindy	C3
S4	Devinder	C1

Seen this before? Yes, this is our original ENROLMENT. The JOIN has reversed the split. (And has "lost" the second Boris.)

11

Definition of JOIN

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

The heading *Hs* of *s* is the union of the headings of *r1* and *r2*.

The body of *s* consists of those tuples having heading *Hs* that can be formed by taking the union of *t1* and *t2,* where *t1* is a tuple of *r1* and *t2* is a tuple of *r2*.

If *c* is a common attribute, then it must have the same declared type in both *r1* and *r2*. (I.e., if it doesn't, then *r1* JOIN *r2* is undefined.)

Note: JOIN, like AND, is both commutative and associative.

12

RENAME

Sid1 is called Name

IS_CALLED **RENAME** (StudentId AS Sid1)

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

Sid1	Name
S1	Anne
S2	Boris
S3	Cindy
S4	Devinder
S5	Boris

13

Definition of RENAME

Let *s* = *r* **RENAME** (*A1* AS *B1*, ... *An* AS *Bn*)

The heading of *s* is the heading of *r* except that attribute *A1* is renamed to *B1* and so on.

The body of *s* consists of the tuples of *r* except that in each tuple attribute *A1* is renamed to *B1* and so on.

14

RENAME and JOIN

Sid1 is called Name AND so is Sid2

IS_CALLED RENAME (StudentId AS Sid1) JOIN
IS_CALLED RENAME (StudentId AS Sid2)

Sid1	Name	Sid2
S1	Anne	S1
S2	Boris	S2
S2	Boris	S5
S5	Boris	S2
S3	Cindy	S3
S4	Devinder	S4
S5	Boris	S5

15

Special Cases of JOIN

What is the result of R JOIN R?

R

What if all attributes are common to both operands?

It is called "intersection".

What if no attributes are common to both operands?

It is called "Cartesian product"

16

Interesting Properties of JOIN

It is *commutative*: *r1* JOIN *r2* ≡ *r2* JOIN *r1*

It is *associative*: (*r1* JOIN *r2*) JOIN *r3* ≡ *r1* JOIN (*r2* JOIN *r3*)
So **Tutorial D** allows JOIN{*r1*, *r2*, …} (note the braces)

We note in passing that these properties are important for *optimisation* (in particular, of query evaluation).

Of course it is no coincidence that logical AND is also both commutative and associative.

17

Projection (= EXISTS)

Student StudentId is enrolled **on some course**.

IS_ENROLLED_ON { StudentId }
= IS_ENROLLED_ON { ALL BUT CourseId }

Given:

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

To obtain:

StudentId
S1
S2
S3
S4

18

Definition of Projection

Let *s* = *r* { *A1, … An* }
(= *r* { ALL BUT *B1, … Bm* })

The heading of *s* is the subset of the heading of *r* given by { *A1, … An* }.

The body of *s* consists of each tuple that can be formed from a tuple of *r* by removing from it the attributes named *B1, … Bm*.

Note that the cardinality of *s* can be less than that of *r* but cannot be more than that of *r*.

19

How ENROLMENT Was Split

```
VAR IS_CALLED BASE
   SAME_TYPE_AS (ENROLMENT { StudentId, Name })
    KEY { StudentId };
IS_CALLED := ENROLMENT { StudentId, Name };

VAR IS_ENROLLED_ON BASE
  SAME_TYPE_AS (ENROLMENT { ALL BUT Name })
   KEY { StudentId, CourseId } ;
IS_ENROLLED_ON := ENROLMENT { ALL BUT Name };
```

Can be done even more economically—see the Notes!

20

Special Cases of Projection

What is the result of R { ALL BUT }?

R

What is the result of R { }?

A relation with no attributes at all, of course!

There are two such relations, of cardinality 1 and 0. The pet names TABLE_DEE and TABLE_DUM have been advanced for these two, respectively.

21

Another Special Case of JOIN

What is the result of R JOIN TABLE_DEE ?

R

So TABLE_DEE is the *identity* under JOIN (cf. 0 under addition and 1 under multiplication.)

22