Introduction to Relational Databases

Hugh Darwen

had6@open.ac.uk
web.onetel.com/~hughdarwen/M358
www.thethirdmanifesto.com

Adapted from Warwick University course material used in
CS233: “Introduction to Relational Databases”
Section 1: Introduction

(Explanatory notes are available for off-line study.)

Some Preliminaries

The theory taught in this part of the course was originally devised by Edgar F. Codd in 1969. His seminal paper (1970) was entitled *A Relational Model of Data for Large Shared Data Banks*.

We will use a language called **Tutorial D** for examples and exercises. (Not in M358!)

We will use **Rel**, an implementation of **Tutorial D,** for our on-line work. (Not in M358!)

What Is a Database?

An *organised*, machine-readable collection of *symbols*, to be *interpreted* as a *true* account of some *enterprise*.

Machine-*updatable*, too …

… so a database is also a collection of *variables*.

A database is typically available to a community of *users*, with possibly varying requirements.

“Organised Collection of Symbols”

For example:

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

The symbols are organised into rows and columns, thus forming a table. One of the rows is different in kind from the others.

“To Be Interpreted as a True Account”

For example (from the table just shown):

StudentId	Name	CourseId
S1	Anne	C1

Perhaps those green symbols, organised as they are with respect to the blue ones, are to be understood to mean:

> “Student S1, named Anne, is enrolled on course C1.”

“Collection of Variables”

ENROLMENT

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

ENROLMENT is a *variable*. Perhaps the table we saw earlier was once its *value*. If so, it (the variable) has been *updated* since then (the row for S4 has been added).

What Is a Relational Database?

A database whose symbols are organised into a collection of *relations*. Here is a relation, shown in tabular form:

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

Might be the value currently assigned to ENROLMENT, a *relation variable* ("relvar").

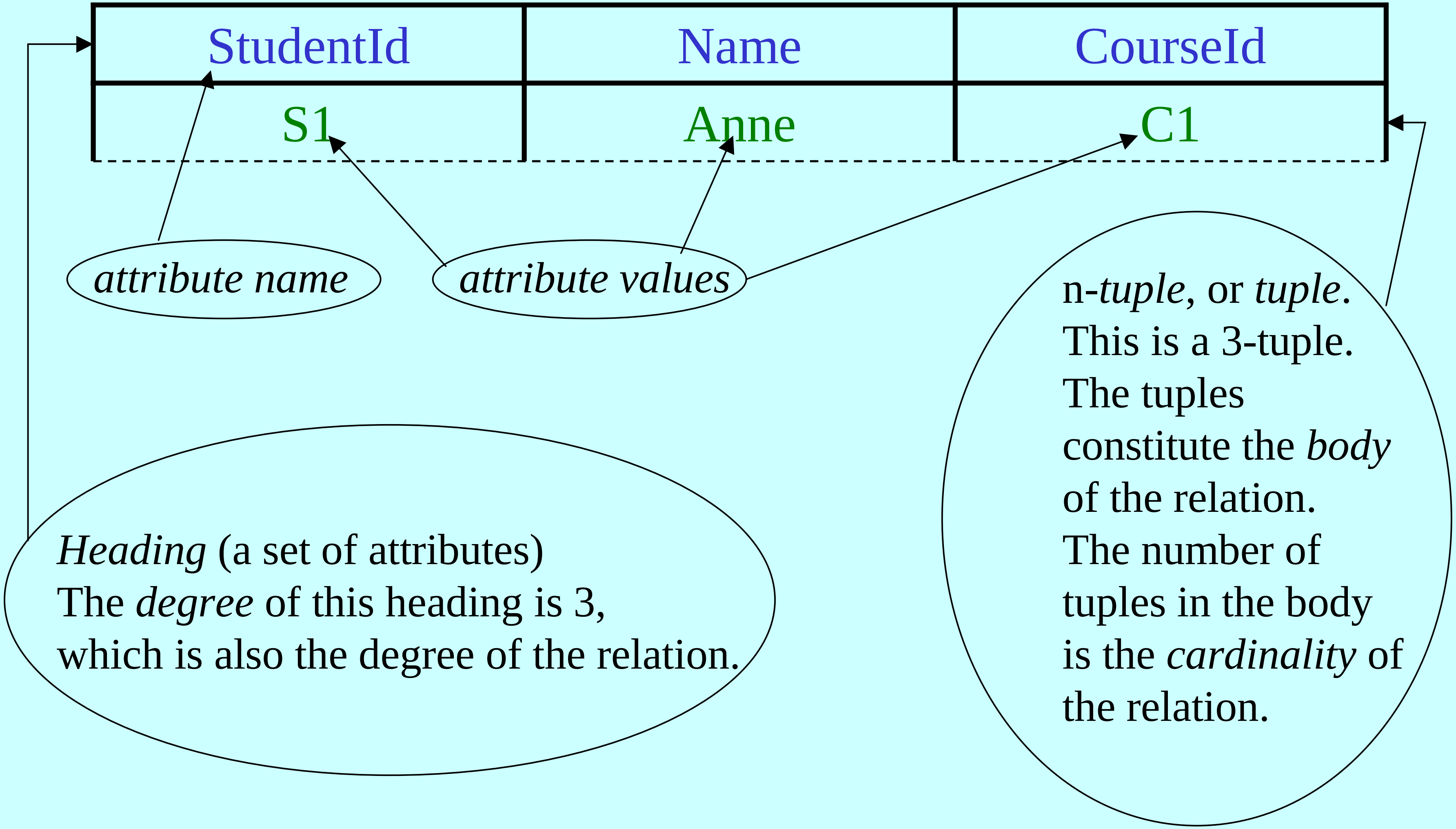
Relation ≠ Table

This table is different from the one we have just seen, but it represents the same relation:

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

In other words, the relation represented does not depend on the order in which we place the rows or the columns in the table.

Anatomy of a Relation



StudentId	Name	CourseId
S1	Anne	C1

attribute name

attribute values

n-*tuple*, or *tuple*.
This is a 3-tuple.
The tuples constitute the *body* of the relation.
The number of tuples in the body is the *cardinality* of the relation.

Heading (a set of attributes)
The *degree* of this heading is 3, which is also the degree of the relation.

What Is a DBMS?

A piece of software for managing databases and providing access to them.

A DBMS responds to *imperatives* (“statements”) given by *application programs*, custom-written or general-purpose, executing on behalf of users.

Imperatives are written in the *database language* of the DBMS (e.g., SQL).

Responses include completion codes, messages and results of *queries*.

What Does a DBMS Do?

In response to requests given by application programs:

- creates and destroys variables
- takes note of integrity rules (*constraints*)
- takes note of *authorisations* (who is allowed to do what, to what)
- updates variables (honouring constraints and authorisations)
- provides results of *queries*
- and more

Now, how does a *relational* DBMS do these things? …

Create and Destroy Variables

Creation (in **Tutorial D**):

```
VAR ENROLMENT BASE RELATION
        { StudentId    SID ,
          Name         CHAR,
          CourseId     CID }
      KEY { StudentId, CourseId } ;
```

Destruction:

```
DROP VAR ENROLMENT ;
```

Take Note of Integrity Rules

E.g., can’t have more than 20,000 enrolments altogether. In **Tutorial D**:

```
CONSTRAINT MAX_ENROLMENTS
    COUNT ( ENROLMENT ) ⩽ 20000  ;
```

And if a constraint ceases to be applicable:

```
DROP CONSTRAINT MAX_ENROLMENTS ;
```

Take Note of Authorisations

E.g. (perhaps – but not in **Tutorial D**):

```
PERMISSION U9_ENROLMENT FOR User9
TO READ ENROLMENT  ;

PERMISSION U8_ENROLMENT FOR User8
TO UPDATE ENROLMENT ;
```

Permissions sometimes need to be withdrawn:

```
DROP PERMISSION U9_ENROLMENT ;
```

Updates Variables

E.g.:

```
DELETE ENROLMENT WHERE StudentId = ‘S4’ ;

UPDATE ENROLMENT WHERE StudentId = ‘S1’
SET ( Name := ‘Ann’ ) ;

INSERT ENROLMENT
   RELATION {
      TUPLE { StudentId ‘S4’ ,
              Name ‘Devinder’ ,
              CourseId  ‘C1’ } }  ;
```

Provides Results of Queries

E.g.: How many students are enrolled on each course?

```
SUMMARIZE ENROLMENT BY { CourseId }
  ADD COUNT ( ) AS No_of_students
```

The result is another relation! In tabular form:

CourseId	No_of_students
C1	3
C2	1
C3	1

EXERCISE

Consider this table:

A	B	A
1	2	3
4		5
6	7	8
9	9	?
1	2	3

Give three reasons why it cannot be representing a relation.