# Database Design Issues, Part II

Hugh Darwen

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

CS252.HACD: Fundamentals of Relational Databases
Section 9: Database Design Issues, Part II

1

## First Normal Form (1NF)

On CS252 we shall assume that every relation, and therefore every relvar, is in 1NF.

The term (due to E.F. Codd) is not clearly defined, partly because it depends on an ill-defined concept of "atomicity" (of attribute values).

Some authorities take it that a relation is in 1NF iff none of its attributes is relation-valued or tuple-valued. It is certainly recommended to avoid use of such attributes (especially RVAs) in database relvars.

2

## 2NF and 3NF

These normal forms, originally defined by E.F. Codd, were really "mistakes". You will find definitions in the textbooks but there is no need to learn them.

The faults with Codd's original definition of 3NF were reported to him by Raymond Boyce. Together they worked on an improved, simpler normal form, which became known as Boyce-Codd Normal Form (BCNF).

3

## Boyce/Codd Normal Form (BCNF)

BCNF is defined thus:

> Relvar *R* is in BCNF if and only if for every nontrivial FD $A \rightarrow B$ satisfied by *R*, *A* is a superkey of *R*.

More loosely, "every nontrivial determinant is a [candidate] key".

BCNF addresses redundancy arising from JDs that are consequences of FDs.

(Not all JDs are consequences of FDs. We will look at the others later.)

4

## Splitting ENROLMENT (bis)

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 |

The attributes involved in the "rogue" FD have been separated into IS_CALLED, *and now we can add student S5*!

5

## Advantages of BCNF

With reference to our ENROLMENT example, decomposed into the BCNF relvars IS_CALLED and IS_ENROLLED_ON:

- Anne's name is recorded twice in ENROLMENT, but only once in IS_CALLED. In ENROLMENT it might appear under different spellings (Anne, Ann), unless the FD { StudentId } → { Name} is declared as a constraint. **Redundancy** is the problem here.
- With ENROLMENT, a student's name cannot be recorded unless that student is enrolled on some course, and an anonymous student cannot be enrolled on any course. Lack of **orthogonality** is the problem here.

6

## Another Kind of Rogue FD

TUTORS_ON

| StudentId | TutorId | TutorName | CourseId |
|---|---|---|---|
| S1 | T1 | Hugh | C1 |
| S1 | T2 | Mary | C2 |
| S2 | T3 | Lisa | C1 |
| S3 | T4 | Fred | C3 |
| S4 | T1 | Hugh | C1 |

Assume the FD { TutorId } → { TutorName } holds.

7

## Splitting TUTORS_ON

TUTOR_NAME

| TutorId | TutorName |
|---|---|
| T1 | Hugh |
| T2 | Mary |
| T3 | Lisa |
| T4 | Fred |
| T5 | Zack |

TUTORS_ON_BCNF

| StudentId | TutorId | CourseId |
|---|---|---|
| S1 | T1 | C1 |
| S1 | T2 | C2 |
| S2 | T3 | C1 |
| S3 | T4 | C3 |
| S4 | T1 | C1 |

Now we can put Zack, who isn't assigned to anybody yet, into the database. Note the FK required for TUTORS_ON_BCNF.

8

## Dependency Preservation

SCOR

| StudentId | CourseId | Organiser | Room |
|---|---|---|---|
| S1 | C1 | Owen | 13 |
| S1 | C2 | Olga | 24 |
| S2 | C1 | Owen | 13 |

Assume FDs: { CourseId } → { Organiser }
{ Organiser } → { Room }
{ Room } → { Organiser }

Which one do we address first?

9

## Try 1: {CourseId}→{Organiser}

SCR

| StudentId | CourseId | Room |
|---|---|---|
| S1 | C1 | 13 |
| S1 | C2 | 24 |
| S2 | C1 | 13 |

CO

| CourseId | Organiser |
|---|---|
| C1 | Owen |
| C2 | Olga |

"Loses" { Room } → { Organiser }
and {Organiser } → { Room }

10

## Try 2: {Room}→{Organiser}

SCR

| StudentId | CourseId | Room |
|---|---|---|
| S1 | C1 | 13 |
| S1 | C2 | 24 |
| S2 | C1 | 13 |

OR

| Organiser | Room |
|---|---|
| Owen | 13 |
| Olga | 24 |

"Loses" { CourseId } → { Organiser }

11

## Try 3: {Organiser}→{Room}

SCO

| StudentId | CourseId | Organiser |
|---|---|---|
| S1 | C1 | Owen |
| S1 | C2 | Olga |
| S2 | C1 | Owen |

OR

| Organiser | Room |
|---|---|
| Owen | 13 |
| Olga | 24 |

Preserves all three FDs!

(But we must still decompose SCO, of course)

12

## An FD That Cannot Be Preserved

TUTOR_FOR

| StudentId | TutorId | CourseId |
|---|---|---|
| S1 | T1 | C1 |
| S1 | T2 | C2 |
| S2 | T3 | C1 |
| S3 | T4 | C3 |
| S4 | T1 | C1 |

Now assume the FD { TutorId } → { CourseId } holds.

This is a third kind of rogue FD.

13

## Splitting TUTOR_FOR

TUTORS

| StudentId | TutorId |
|---|---|
| S1 | T1 |
| S1 | T2 |
| S2 | T3 |
| S3 | T4 |
| S4 | T1 |

TEACHES

| TutorId | CourseId |
|---|---|
| T1 | C1 |
| T2 | C2 |
| T3 | C1 |
| T4 | C3 |

Note the keys.

Have we "lost" the FD { StudentId, CourseId } → { TutorId } ?

And the FK referencing IS_ENROLLED_ON?

14

## Reinstating The Lost FD

Need to add the following constraint:

```
CONSTRAINT KEY_OF_TUTORS_JOIN_TEACHES
  IS_EMPTY ( ( TUTORS JOIN TEACHES )
  GROUP { ALL BUT StudentId, CourseId } AS G
  WHERE COUNT ( G ) > 1 ) ;
```

or equivalently:

```
CONSTRAINT KEY_OF_TUTORS_JOIN_TEACHES
   WITH TUTORS JOIN TEACHES AS TJT :
     COUNT ( TJT ) = COUNT ( TJT { StudentId, CourseId } ) ;
```

15

## And The Lost Foreign Key

The "lost" foreign key is easier:

```
CONSTRAINT FK_FOR_TUTORS_JOIN_TEACHES
  IS_EMPTY ( ( TUTORS JOIN TEACHES )
                NOT MATCHING
                IS_ENROLLED_ON ) ;
```

16

## In BCNF But Still Problematical

TBC1

| Teacher | Book | CourseId |
|---|---|---|
| T1 | Database Systems | C1 |
| T1 | Database in Depth | C1 |
| T1 | Database Systems | C2 |
| T1 | Database in Depth | C2 |
| T2 | Database in Depth | C2 |

Assume the JD *{ { Teacher, Book }, { Book, CourseId } } holds.

17

## Normalising TBC1

TB

| Teacher | Book |
|---|---|
| T1 | Database Systems |
| T1 | Database in Depth |
| T2 | Database in Depth |

BC

| Book | CourseId |
|---|---|
| Database Systems | C1 |
| Database in Depth | C1 |
| Database Systems | C2 |
| Database in Depth | C2 |

We have lost the constraint implied by the JD, but does a teacher really have to teach a course just because he or she uses a book that is used on that course?

18

## Fifth Normal Form (5NF)

5NF caters for *all* harmful JDs.

> **Relvar *R* is in 5NF iff every nontrivial JD that holds in *R* is implied by the keys of *R*.** (Fagin's definition, 1979)
>
> Apart from a few weird exceptions, a JD is "implied by the keys" if every projection is a superkey. (Date's definition – but see the Notes for this slide)

To explain "nontrivial": A JD is trivial if and only if one of its operands is the entire heading of *R* (because every such JD is clearly satisfied by *R*).

19

## A JD of Degree > 2

TBC2

| Teacher | Book | CourseId |
|---|---|---|
| T1 | Database Systems | C1 |
| T1 | Database in Depth | C1 |
| T1 | Database Systems | C2 |
| T1 | Database in Depth | C2 |
| T2 | Database in Depth | C2 |

Now assume the JD *{ { Teacher, Book }, { Book, CourseId }, { Teacher, CourseId } } holds.

20

## Normalising TBC2

TB

| Teacher | Book |
|---|---|
| T1 | Database Systems |
| T1 | Database in Depth |
| T2 | Database in Depth |

BC

| Book | CourseId |
|---|---|
| Database Systems | C1 |
| Database in Depth | C1 |
| Database Systems | C2 |
| Database in Depth | C2 |

TC

| Teacher | CourseId |
|---|---|
| T1 | C1 |
| T1 | C2 |
| T2 | C2 |

(and we've "lost" the constraint again)

21

## Sixth Normal Form (6NF)

6NF subsumes 5NF and is the strictest NF:

> Relvar *R* is in 6NF if and only if every JD that holds in *R* is trivial.

6NF provides maximal orthogonality, as already noted, but is not normally advised. It addresses additional anomalies that can arise with temporal data (beyond the scope of this course—and, what's more, the definition of join dependency has to be revised).

22

## Wives of Henry VIII in 6NF

W_FN

| Wife# | FirstName |
|---|---|
| 1 | Catherine |
| 2 | Anne |
| 3 | Jane |
| 4 | Anne |
| 5 | Catherine |
| 6 | Catherine |

W_LN

| Wife# | LastName |
|---|---|
| 1 | of Aragon |
| 2 | Boleyn |
| 3 | Seymour |
| 4 | of Cleves |
| 5 | Howard |
| 6 | Parr |

W_F

| Wife# | Fate |
|---|---|
| 1 | divorced |
| 2 | beheaded |
| 3 | died |
| 4 | divorced |
| 5 | beheaded |
| 6 | survived |

*Not* a good idea!

23

## EXERCISE (see Notes)

24