CS233:HACD Introduction to Relational Databases
Notes for Section 9: Database Design Issues, Part II

1. Cover slide

Now at last we can look at *normal forms.* In so doing we will discover some issues that have still not been satisfactorily resolved.

2. First Normal Form (1NF)

No notes.

3. Second Normal Form (2NF)

2NF is sometimes defined using the term "fully dependent". A relvar is in 2NF if and only if every nonkey attribute is fully dependent on the primary key.

The assumption that the primary key is the only key is clearly unsafe in general, which is why we will not pay much attention to 2NF on this course.

4. Splitting ENROLMENT (bis)

IS_CALLED and IS_ENROLLED_ON are in 2NF. As it happens, they are also in 3NF, BCNF, 4NF, 5NF and 6NF, so we need to look at some different examples to discover cases where a lower NF holds while a higher one is violated.

It seems as if the decomposition also allows us to add enrolments for students who have no name, and that is true unless a constraint is declared to the effect that every StudentId value appearing in IS_ENROLLED_ON also appears as a StudentId value in IS_CALLED.

Exercise: Write a Tutorial D declaration for this constraint.

5. Advantages of 2NF

These problems give rise to what have been called "update anomalies":

1. Inability to record some fact (in this case, somebody's name) without at the same time recording some other fact (in this case, an enrolment).
2. Loss of a fact (in this case, somebody's name) when some other recorded fact ceases to be true (in this case, the only enrolment for the student in question).
3. The same update (e.g., correction of somebody's name) might need to be done in more than one place (so to speak).
4. When a fact is recorded (in this case, an enrolment), the accompanying fact, if already recorded (e.g. S1's name), must be copied faithfully.

Exercise: Write a **Tutorial D** constraint declaration to express the FD mentioned in the slide.

6. In 2NF But Still Problematical

Every nonkey attribute (TutorId and TutorName) is irreducibly dependent on { StudentId, CourseId }, so this is in 2NF. However, the nonkey attribute TutorName is dependent on TutorId, also a nonkey attribute.

We have the same problems as with our non-2NF relvar: T1's name is recorded twice (so we need to make sure it is always spelled the same way and not, for example, Hugh in one place and Huw in another), and we cannot record the name of a tutor who hasn't been assigned for any enrolment yet.

7. Third Normal From (3NF)

Note that *a* here is necessarily a nonkey attribute of *R*, where a nonkey attribute is one that is not a member of the primary key.

In TUTORS_ON we have an FD, { TutorId } ↦ { TutorName }, such that its determinant, its dependant set and the primary key are disjoint, thus violating 3NF. TutorName is "transitively dependent" on the primary key.

8. Splitting TUTORS_ON

Note in passing that in TUTOR_ON_3NF, { StudentId, CourseId } should be a foreign key referencing IS_ENROLLED_ON, to make sure that the enrolment to which a tutor is assigned does exist. As we have seen, that constraint can be declared in **Tutorial D** thus:

```
CONSTRAINT FK_FOR_TUTOR_ON
IS_EMPTY ( TUTOR_ON_3NF NOT MATCHING IS_ENROLLED_ON ) ;
```

9. In 3NF But Still Problematical

Under the assumed FD we have the possibly unrealistic situation, for the sake of an example, where no teacher teaches more than one course.

We have exactly the same problems of redundancy and lack of orthogonality, and yet TUTOR_FOR is in 3NF according to our definition, because TutorId is not a nonkey attribute.

Note that although each tutor teaches only one course, and exactly one tutor is assigned for a single enrolment, the same course can have several tutors (T1 and T3 both teach C1).

Note that the primary key, {StudentId, CourseId} is not the only key of TUTOR_FOR. Because {Tutor} ↦ {CourseId} holds, {Tutor, StudentId} ↦ {CourseId} also holds, under left-augmentation, therefore {Tutor, StudentId} is also a key.

But if {Tutor, StudentId} is chosen to be the primary key, then TUTOR_FOR is not in 2NF!!

So pursuit of 2NF and 3NF fails to achieve its objective in certain cases. Note that these "rogue" cases can only arise when there is more than one key and two of the keys "overlap" (have an attribute in common). However, having overlapping keys doesn't *necessarily* lead to a violation of 3NF.

10. Boyce/Codd Normal Form (BCNF)

TUTOR_FOR is not in BCNF because { TutorId } is the determinant of a nontrivial FD and yet is not a superkey.

11. Not In 3NF But Not Problematical

Under the assumed FD we have { Employee# } ↦ { NatIns# } ↦ { Name }, an apparent violation of 3NF. And yet we have no redundancy, not updating problems. We do lack orthogonality, but that is not a problem if it really is the case that every employee has both a name and a national insurance number.

The reason why EMP is not a problem in spite of apparently violating 3NF is that EMP has two keys, {Employee#} and {NatIns#}. The determinant of the FD, { NatIns# } ↦ { Name, Employee# }, that makes it violate 3NF, is a key (though not the one arbitrarily chosen to be the primary key). Because it is a key, the problems normally arising from not being in 3NF do not arise.

Next we look at a relvar that appears to be in 3NF and yet *is* problematical …

12. Splitting TUTOR_FOR

Yes, we have lost the constraint that was expressed by the key, {StudentId, CourseId} of TUTOR_FOR. If we do not somehow reinstate that constraint, it will be possible to enrol student S1 of course C1 for a second time, with tutor T3.

13. Reinstating The Lost FD

Any functional dependency, $A \mapsto B$, in relvar R can be expressed as a constraint of the form:

COUNT (R { A , B }) = COUNT (R { A })

14. And The Lost Foreign Key

Perhaps it is just as well that **Tutorial D** does not have the FOREIGN KEY shorthand. It is easy to use the same longhand here as we did for the original TUTOR_FOR relvar, just replacing the name TUTOR_FOR by (TUTORS JOIN TEACHES).

15. In BCNF But Still Problematical

Perhaps the predicate is "*Teacher* uses *Book* on course *CourseId"*. In that case, the given JD tells us that if teacher *t* uses book *b* at all and *b* is used by anybody on course *c*, then *t* uses *b* on *c*.

The two books in the example are both used on C1 and C2 and teacher T1 uses both books. The appearance of the fourth tuple shown in the diagram is implied by the fifth (showing that somebody uses "Database in Depth" on C2) and the second (showing that T1 uses "Database in Depth" on some course). The JD also tells us that T2 does not use "Database Systems" at all, for otherwise that teacher would be compelled to use it on course C2.

Now, suppose we learn that T2 is to teach course C1 and has decided to use "Database Systems" on that course. Then under the JD (assuming it is declaed as a constraint) the DBMS must reject any attempt to insert the tuple <T2, Database Systems, C1> unless the tuple <T2, Database Systems, C2> is inserted at the same time.

Once again we have redundancy: that T1 uses Database Systems is recorded twice, and that Database in Depth is used on C2 is also recorded twice. And once again we have lack of orthogonality. The simple predicates "*Teacher* uses *Book*" and "*Book* is used on course *CourseId"* are not treated independently.

16. Fourth Normal Form (4NF)

TBC1 is not in 4NF because the nontrivial JD *{ { Teacher, CourseId }, { Book, CourseId } } is not implied by the only key of TBC1, which is { Teacher, Book, CourseId }, the entire heading.

A JD is "implied by the keys of *R*" if each of its projections includes the same key of *R*.

17. Normalising TBC1

No notes.

18. In 4NF But Still Problematical

Still taking the predicate to be "*Teacher* uses *Book* on course *CourseId*", the revised JD tells us that if teacher *t* uses book *b* at all and *b* is used by anybody on course *c*, and *t* teaches course *c*, then *t* uses *b* on *c*.

We are heading for a 3-way decomposition, as we were with the decomposition on WIFE_OF_HENRY_VIII, but this time there can be no intermediate step—we cannot do it by two 2-way decompositions. Our ternary JD here is what might called *essentially* ternary.

19. Fifth Normal Form (5NF)

No notes.

20. Normalising TBC2

Verify that TB JOIN BC JOIN TC = TBC.

21. Sixth Normal Form (6NF)

No notes.

22. Wives of Henry VIII in 6NF

Use 6NF only for nonkey attributes that are "optional". If every wife has a first name but not every wife has a last name and not every wife has a fate, then the above design is a good one.

Decomposing a 5NF relvar introduces orthogonality but does not eliminate any redundancy because a 5NF relvar is free of redundancy.

End of Notes