Database Design Issues, Part I

Hugh Darwen

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

CS252.HACD: Fundamentals of Relational Databases
Section 8: Database Design Issues, Part I

1

A "Reducible" Relation

WIFE_OF_HENRY_VIII

Wife#	FirstName	LastName	Fate
1	Catherine	of Aragon	divorced
2	Anne	Boleyn	beheaded
3	Jane	Seymour	died
4	Anne	of Cleves	divorced
5	Catherine	Howard	beheaded
6	Catherine	Parr	survived

(Note the underscoring of the primary key attribute.)

2

"Decomposing" H8's Wives

W_FN

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

W_LN_F

Wife#	LastName	Fate
1	of Aragon	divorced
2	Boleyn	beheaded
3	Seymour	died
4	of Cleves	divorced
5	Howard	beheaded
6	Parr	survived

3

Join Dependency

The *join dependency* (JD) that holds in WIFE_OF_HENRY_VIII and allows us to decompose in W_FN and W_LN_F is written like this:

* { { Wife#, FirstName }, { Wife#, LastName, Fate } }

• The star indicates a JD.
• The operands of a JD are written inside braces.
• Each operand is a set of attributes, hence the inner braces.

If a given JD holds in relvar *r*, then at all times *r* = the join of the projections indicated by the operands of the JD.

4

A Join Dependency That Does Not Hold

Although W_FN is irreducible, we can of course take several projections of it, the following two in particular:

W_FN { Wife# }

Wife#
1
2
3
4
5
6

W_FN { FirstName }

FirstName
Catherine
Anne
Jane

But the JOIN of these two does not yield W_FN, so the JD *{ { Wife# }, { FirstName} } does not hold in W_FN.

5

Decomposition of W_LN_F

W_LN_F can be further decomposed:

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

6

3-Way Join Dependency

So the following JD holds in W_LN_F:

* { { Wife#, LastName }, { Wife#, Fate } }

and we can conclude that the following 3-way JD holds in WIFE_OF_HENRY_VIII:

* { { Wife#, FirstName }, { Wife#, LastName }, { Wife#, Fate } }

i.e., WIFE_OF_HENRY_VIII =
WIFE_OF_HENRY_VIII { Wife#, FirstName } JOIN
WIFE_OF_HENRY_VIII { Wife#, LastName } JOIN
WIFE_OF_HENRY_VIII { Wife#, Fate }

7

Design Comparison

Does the decomposed design have any advantages?

The single relvar design carries an implicit constraint to the effect that every wife has a wife number, a first name, a last name and a fate.

This constraint is not implicit in the decomposed design.

In fact, to enforce it, each of W_FN, W_LN and W_F needs a foreign key referencing each of the other two.

But then the first attempt to insert a tuple into any of them must fail (unless multiple assignment is available).

8

Conclusion

In the example at hand, the single relvar design is preferred, *so long as the constraint implied by it truly reflects the real world.*

(For example, if it turns out that in fact not every wife has a last name, then we should separate out W_LN.)

But the example at hand is rather special:
• Each operand of the 3-way JD that holds in WIFE_OF_HENRY_VIII includes a key of that relvar
• and it is the same key in each case, viz. {Wife#}

We need to look at some not-so-special examples.

9

A Not-So-Special JD

ENROLMENT

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

Recall that we decided to "split" (i.e., decompose) this one, as follows …

10

Splitting ENROLMENT

IS_CALLED

StudentId	Name
S1	Anne
S2	Boris
S3	Cindy
S4	Devinder

IS_ENROLLED_ON

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

Notice the JD: *{ { StudentId, Name }, { StudentId, CourseId } } that holds in ENROLMENT.

11

Functional Dependency

Because {StudentId} is a key of the projection ENROLMENT{StudentId, Name}, we say that the following *functional dependency* (FD) holds in ENROLMENT:

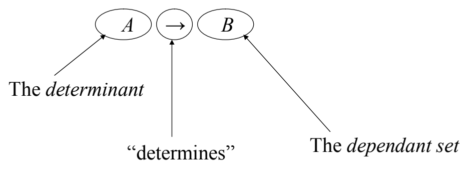
{ StudentId } → { Name }

• The arrow, pronounced "determines", indicates an FD.
• Each operand is a set of attributes (hence the braces).

Name is a function of StudentId.
For each StudentId there is exactly one Name.

12

Anatomy of an FD



Reminder: The determinant is a *set* of attributes, and so is the dependant set.

P.S. “dependant” is not a misspelling! It’s the noun, not the adjective.

13

FDs That Do Not Hold in ENROLMENT

{ Name } → { StudentId }
{ Name } → { CourseId }
{ CourseId } → { StudentId }
{ CourseId, Name } → { StudentId }
{ StudentId } → { CourseId }
{ StudentId, Name } → { CourseId }

also: { x } → { StudentId }, because x is not an attribute of ENROLMENT.

14

Theorems About FDs

Assume $A \rightarrow B$ holds in *r*. Then:

Left-Augmentation: If A' is a superset of A, then $A' \rightarrow B$ holds in *r*.

Right-reduction: If B' is a subset of B, then $A \rightarrow B'$ holds in *r*.

Transitivity: If $A \rightarrow B$ and $B \rightarrow C$ hold in *r*, then $A \rightarrow C$ holds in *r*.

In general: If $A \rightarrow B$ and $C \rightarrow D$ hold in *r*, then:

$$A \cup (C - B) \rightarrow B \cup D$$

holds in *r*.

15

Left-Irreducibility

If $A \rightarrow B$ holds in *r* and there is no proper subset A' of A such that $A' \rightarrow B$ holds in *r,* then $A \rightarrow B$ is a *left-irreducible* FD (in *r*). In this case, *B* is sometimes said to be *fully* dependent on *A*.

Conversely, if there *is* such a subset A', then $A \rightarrow B$ is a *left-reducible* FD (in *r*), and *B* is therefore *not fully* dependent on *A*.

16

FDs and Keys

If $A \rightarrow B$ is an FD in *r* and $A \cup B$ constitutes the entire heading of *r*, then A is a *superkey* of *r*.

If $A \rightarrow B$ is a left-irreducible FD in *r* and $A \cup B$ constitutes the entire heading of *r*, then *A* is a *key* of *r*.

(The longer term *candidate key* is often used instead of *key,* for historical reasons.)

17

Normal Forms

Arising from the study of JDs in general and FDs in particular, various “normal forms” have been defined:
- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- **Boyce/Codd Normal Form (BCNF)**
- Fourth Normal Form (4NF)
- **Fifth Normal Form (5NF)**
- **Sixth Normal Form (6NF)**

Each of these is in a sense stricter than its immediate predecessor. The ones shown in bold are particularly important. The others were early attempts that eventually proved inadequate.

18

Normalisation

Normalisation is the act of decomposing a relvar that fails to satisfy a given normal form (e.g., BCNF) such that the result is an equivalent set of two or more "smaller" relvars that do satisfy that normal form.

We decompose by taking the projections specified in a given join dependency (JD).

In the case of ENROLMENT, the given JD is
*{ { StudentId, Name }, {StudentId, CourseId } }
determined by the FD { StudentId } → { Name }

19

Purposes of Normalisation

A database all of whose relvars satisfy 6NF has the following possibly desirable properties:

- No redundancy (i.e., no recording of the same information more than once)
- No "update anomalies" (to be explained later)
- Orthogonality (independent recording of the simplest facts)

But 5NF is usually sufficient (and 6NF is sometimes problematical with existing technology), as we shall see …

20