Some Simple Normalization Steps

CIT 381
What Normalization Is For

Avoid redundancy

“One fact in one place”

Analyze dependencies among attributes

May need to decompose tables into subtables

Generally avoided by common sense during design phase.
Example of Redundancy

Consider a table containing student info and enrollment info (here CRN means course reference number)

<table>
<thead>
<tr>
<th>stud#</th>
<th>name</th>
<th>CRN</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>Alice</td>
<td>15</td>
</tr>
<tr>
<td>102</td>
<td>Alice</td>
<td>17</td>
</tr>
<tr>
<td>150</td>
<td>Bob</td>
<td>15</td>
</tr>
</tbody>
</table>
Clearly Redundant

The **student-enroll** table is clearly bad design. Let’s examine why.

Obviously the student information should be kept separately…

… in a **student** table, whose key would be **stud#**.
If we know the stud#, we know the name. This is written as

\[ \text{stud#} \rightarrow \text{name} \]

and is known as a functional dependency.
Partial Dependency

A *partial dependency* is a situation where an attribute in a table can depend on just part of the key, not the whole thing.

This is exhibited in the `student-enroll` table:

its key is `(stud#, CRN)`

and a partial dependency is

`stud# → name`
Second Normal Form

Partial dependencies in a table are one cause of redundancy. A schema is in *second normal form* if all partial dependencies have been eliminated.

To eliminate them, decompose the offending tables:

<table>
<thead>
<tr>
<th>student</th>
<th>enroll</th>
</tr>
</thead>
<tbody>
<tr>
<td>stud#</td>
<td>name</td>
</tr>
<tr>
<td>102</td>
<td>Alice</td>
</tr>
<tr>
<td>150</td>
<td>Bob</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hey, where is first normal form?

First normal form simply requires that all attributes are atomic, or cannot be broken down into sub-parts. Each attribute value is a single fact.

This is pretty much the definition of the relational model.
More Redundancy

Consider the CourseOfferings table below, with key CRN:

CourseOfferingsInstr

<table>
<thead>
<tr>
<th>CRN</th>
<th>title</th>
<th>instr#</th>
<th>instrName</th>
</tr>
</thead>
<tbody>
<tr>
<td>1778</td>
<td>Intro DB</td>
<td>16</td>
<td>Craig</td>
</tr>
<tr>
<td>1426</td>
<td>Art Hist</td>
<td>18</td>
<td>Yolanda</td>
</tr>
<tr>
<td>1376</td>
<td>French I</td>
<td>16</td>
<td>Craig</td>
</tr>
</tbody>
</table>
This is in 2nd Normal Form

<table>
<thead>
<tr>
<th>CRN</th>
<th>title</th>
<th>instr#</th>
<th>instrName</th>
</tr>
</thead>
<tbody>
<tr>
<td>1778</td>
<td>Intro DB</td>
<td>16</td>
<td>Craig</td>
</tr>
<tr>
<td>1426</td>
<td>Art Hist</td>
<td>18</td>
<td>Yolanda</td>
</tr>
<tr>
<td>1376</td>
<td>French I</td>
<td>16</td>
<td>Craig</td>
</tr>
</tbody>
</table>

No partial dependencies here (there can’t be)

…but there is obviously redundancy!
Transitive Dependencies

Clearly the Instructor information should have been in a separate table. Why?

Because of a transitive dependency:
from a key to a non-key to another non-key

\[ CRN \rightarrow instr\# \rightarrow instrName \]
Third Normal Form

First, put schema into 2nd normal form (no partial dependencies).

Then ensure there are no transitive dependencies. If so, decompose offending table(s).

**CourseOffering:** CRN  title  instr#

**Instructor:** instr#  instrName
Boyce Codd Normal Form

A very formal statement, but worth seeing since BCNF is the gold standard.

Let X and Y be sets of attributes. A relation R is in BCNF if for all functional dependencies $X \rightarrow Y$ that hold on R, either:

i) it’s a trivial FD (Y is contained in X), or

ii) $X$ is a superkey for R
Higher Normal Forms

Suppose we already have tables

**Instructor**, with key *instr#*

**Text**, with key *title*

**Course**, with key *courseNum*

Business Rule: A course may be taught several times, by different instructors.

Business Rule: The texts used in a course do not depend on the instructor. (That is, another instructor must use the same text(s).)
In BCNF, But Still Redundant

<table>
<thead>
<tr>
<th>courseNum</th>
<th>instr#</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>381</td>
<td>14</td>
<td>Intro DB</td>
</tr>
<tr>
<td>381</td>
<td>14</td>
<td>SQL</td>
</tr>
<tr>
<td>210</td>
<td>17</td>
<td>Intro Java</td>
</tr>
</tbody>
</table>

If we want to add instructor 17 to 381, we must add two lines:

<table>
<thead>
<tr>
<th>courseNum</th>
<th>instr#</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>381</td>
<td>17</td>
<td>Intro DB</td>
</tr>
<tr>
<td>381</td>
<td>17</td>
<td>SQL</td>
</tr>
</tbody>
</table>
Fourth Normal Form

The table

\textbf{CourseInstrText:} \textit{courseNum} \textit{instr\#} \textit{text}

has no functional dependencies, so is BCNF.

But it’s redundant due to a bad \textit{multivalued dependency}:

the \textbf{set} of texts depend on the course.

The formalization is long -- let’s skip it. Anyway, the table above is not in 4NF (nor 5NF, 6NF).
Denormalization

Sometimes you may not want to decompose a table, usually for the sake of efficiency. (Joins are expensive.)

Thus, it may be worth living with redundancy.

Two Main Points

*Do this consciously, not by accident.*

*Be able to justify your decision.*
Data Model Scorecard

1. How well do the characteristics of the model support the type of model?
2. How well does the model capture the requirements?
3. How complete is the model?
4. How structurally sound in the model?
5. How well does the model leverage generic structures?

(from Hoberman text)
scorecard, continued

6. How well does the model follow naming standards?

7. How well has the model been arranged for readability?

8. How good are the definitions?

9. How consistent is the model with the enterprise?

10. How well does the metadata match the data?