Lecture 14:
• Normalization (cont)

Announcements:
• HW-4 out, due Tues, Oct 25
• PS-3 out, due Tues, Nov 8

Database Normal Forms and Decomposition

**Normal forms**: “levels” of allowed redundancy based on attribute dependencies
- First Normal Form ... atomic values
- Second & Third Normal Form ... allow some redundancy
- Boyce–Codd Normal Form (BCNF) ... no redundancy from functional deps
- Fourth & Fifth Normal Forms ... for multivalued dependencies

**Normalization**: “decomposes” a table into smaller tables
- Decomposition based on the attribute dependencies that exist
- Follows a decomposition algorithm
- We’ll give a sense for how this works (but not dive into algorithms)

*Note*: assumes a universal relation
- All relations in a database “come from” a single (large) relation
Keys Revisited

EmpDept

<table>
<thead>
<tr>
<th>eid</th>
<th>name</th>
<th>dept</th>
<th>dept_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A01</td>
<td>Alice</td>
<td>12</td>
<td>CS</td>
</tr>
<tr>
<td>A12</td>
<td>Bob</td>
<td>10</td>
<td>HR</td>
</tr>
<tr>
<td>A13</td>
<td>Bob</td>
<td>12</td>
<td>CS</td>
</tr>
<tr>
<td>A03</td>
<td>Anne</td>
<td>12</td>
<td>CS</td>
</tr>
</tbody>
</table>

Because **eid** is a key: Given an **eid** value, all other values known
- Since only one row can have the given **eid** value
- We say the values are “determined” by **eid**

A key is like a (mathematical) “function”: same input always gives same value
- \( f: \text{eid} \rightarrow \text{name} \times \text{dept} \times \text{dept\_name} \) ... cartesian product of domains
- e.g.: \( f(A01) = \langle \text{Alice}, 12, \text{CS} \rangle \)

Keys Revisited (cont)

We say that **eid** “functionally determines” all other EmpDept attribute values
- Relationship is called a “functional dependency” (FD)
- Written as: \( \text{eid} \rightarrow \text{name}, \text{dept}, \text{dept\_name} \)
- Implying: \( \text{eid} \rightarrow \text{name}, \text{eid} \rightarrow \text{dept}, \text{and} \ \text{eid} \rightarrow \text{dept\_name} \)

Not all FDs have to be on (implied by) keys

Check In: Which of these could be functional dependencies?
- \( \text{name} \rightarrow \text{dept} \)
- \( \text{name} \rightarrow \text{dept\_name} \)
- \( \text{dept} \rightarrow \text{dept\_name} \) ... YES!
- \( \text{dept\_name} \rightarrow \text{dept} \) ... Maybe (if dept. names are unique)
Types of Functional Dependencies

There are three special kinds of FDs ... where $X$, $Y$ are sets of attributes

**Key FDs** of the form $X \rightarrow Y$ where $X$ contains a key
- i.e., $X$ is a superkey
- the database can enforce these for us

**Trivial FDs** of the form $X \rightarrow Y$ such that $Y \subseteq X$
- e.g: name, dept $\rightarrow$ dept
- these are “boring”

**Non-Key, Non-Trivial FDs**
- The rest: the non-key FDs that aren’t trivial
- These are the “bad” ones

Like keys, FDs are based on the application semantics

---

Using the DBMS to Enforce Functional Dependencies

**For our table:** EmpDept($eid$, name, dept, dept_name)
- with key $eid$
- and FD dept $\rightarrow$ dept_name

Q: Is it still possible for there to be 2 names for the same department?
- YES! ... even with the FD dept $\rightarrow$ dept_name
- The DBMS can enforce (candidate) keys, but not non-key, non-trivial FDs

**Check In:** Identify possible non-key, non-trivial FDs in this example:

Enrollment($student_id$, course_id, instr_id, student_name, instr_name)
- instr_id $\rightarrow$ instr_name
- student_id $\rightarrow$ student_name
- course_id $\rightarrow$ instr_id
Second Normal Form (2NF)

A relation is in 2NF if ...

- every non-key attribute is fully dependent on each candidate key
- note that a key (i.e., “prime”) attribute is in at least one candidate key

*Alt Def.* A relation is in 2NF if for every non-trivial FD $X \rightarrow Y$, either ...

- $X$ is not a proper subset of a candidate key; or else
- $Y$ contains attributes only from candidate keys (i.e., only prime attributes)

*Note:* Sometimes referred to as not having any “partial (key) dependencies”

2NF Examples

**Check In:** Is this relation in 2NF?

Enrollment\((\text{student_id, class_id, instr_id, student_name, instr_name})\)

- No, because of the FD: $\text{student_id} \rightarrow \text{student_name}$
- And this FD can’t be enforced by the DBMS (via keys)

**Check In:** Q: Is this decomposition in 2NF?

Enrollment\((\text{student_id, class_id, instr_id, instr_name})\)

- Yes, even with the FD: $\text{instr_id} \rightarrow \text{instr_name}$
- This FD also can’t be enforced by the DBMS (via keys)
Third Normal Form (3NF)

A relation is in 3NF if for every non-trivial FD $X \rightarrow Y$, either:

- $X \rightarrow Y$ is a key FD ($X$ is a superkey); or
- $Y$ is a part of some candidate key for $R$

3NF sometimes defined as 2NF without “transitive dependencies”
- i.e., without FDs of the form $X \rightarrow Y$ where $X$ and $Y$ are non-prime

Boyce-Codd Normal Form (BCNF)

A relation is in BCNF if all of its non-trivial FDs are ...

- Key FDs (of the form $X \rightarrow Y$ for superkey $X$)

Are either of these relations in BCNF? (Why or why not...)
- EmpDept(eid, name, dept, dept_name)
- Assigned(eid, pid, emp_name, percent)

BCNF relations have no redundancy caused by FDs
- redundancy if there is an FD between attributes
- and there can be repeated entries of data for those attributes