Today

- Normalization (cont)

Assignments

- HW7 out
- Quiz on Tues
Types of Anomalies

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>Eric</td>
<td>10</td>
<td>HR</td>
</tr>
<tr>
<td>A13</td>
<td>Eric</td>
<td>12</td>
<td>CS</td>
</tr>
<tr>
<td>A03</td>
<td>Anne</td>
<td>12</td>
<td>CS</td>
</tr>
</tbody>
</table>

“Update Anomaly”

• If the CS dept. changes its name, we must change multiple rows

“Insertion Anomaly”

• If a department has no employees, where do we store its id and name?

“Deletion Anomaly”

• If A12 quits, the information about the HR department will be lost

Anomalies are in addition to wasted space

• e.g., the dept. name is stored multiple times
Using NULL values

<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>NULL</td>
<td>NULL</td>
<td>10</td>
<td>HR</td>
</tr>
<tr>
<td>A13</td>
<td>Eric</td>
<td>12</td>
<td>CS</td>
</tr>
<tr>
<td>A03</td>
<td>Anne</td>
<td>12</td>
<td>CS</td>
</tr>
</tbody>
</table>

NULL values can help insertion and deletion anomalies

• But NULL values have their own issues
  – make aggregate operators harder to use
  – can be unclear what NULL means
  – may need to use outer joins
  – in this case, eid is a primary key (so can’t contain a null)

• NULL values also don’t address update anomalies or redundancy issues
Decomposition

Normalization involves decomposing the table into separate tables

• Also referred to as “partitioning”

• After decomposing, we check to see if redundancy remains (... repeat)

Functional Dependencies

• Key to understanding when and how to decompose schemas

• Generalize the notion of keys
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>Eric</td>
<td>10</td>
<td>HR</td>
</tr>
<tr>
<td>A13</td>
<td>Eric</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 \( \text{eid} \) is a key (... a different take on a key constraint)

- If we know the \( \text{eid} \) value, all other values are known
- If 2 rows had same \( \text{eid} \) value, they have same values for every other attribute
- Thus, given an \( \text{eid} \) value, all other values are “determined”

A key is like a (mathematical) “function”

- a function always returns the same value for a given input
- \( f: \text{eid} \rightarrow \text{name} \times \text{dept} \times \text{dept\_name} \) ... cartesian product of domains
- e.g.: \( f(\text{A01}) = (\text{Alice},12,\text{CS}) \)

We say that \( \text{eid} \) “functionally determines” all other attribute values

- This relationship is called a “functional dependency” (FD)
- And write FDs as:
  - \( \text{eid} \rightarrow \text{name}, \text{dept}, \text{dept\_name} \)
  - which implies: \( \text{eid} \rightarrow \text{name}, \text{eid} \rightarrow \text{dept}, \text{and} \text{eid} \rightarrow \text{dept\_name} \)
**Functional Dependencies**

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

Q: Which of these could be functional dependencies?

- name $\rightarrow$ dept
- name $\rightarrow$ dept_name
- dept $\rightarrow$ dept_name
  ... YES!
- dept_name $\rightarrow$ dept
  ... Maybe (if dept. names are unique)

For sets $A$ and $B$, $A \rightarrow B$ is a functional dependency ...

- if whenever two rows agree on $A$ they also agree on $B$
- if so, we say $A$ functionally determines $B$

There are three special kinds of FDs ... $A, B, X$ are sets of attributes

- **Key FDs** of the form $X \rightarrow A$ where $X$ contains a key
  - $X$ is called a superkey
- **Trivial FDs** of the form $A \rightarrow B$ such that $B \subseteq A$
  - e.g: name, dept $\rightarrow$ dept
  - these are boring, but become important later ...
- **Non-Key, Non-Trivial** FDs
  - The rest: the non-key FDs and those that aren't trivial

Like keys, FDs are based on the application semantics
Enforcing functional dependencies

For our table

Emp(eid, name, dept, deptname)
  - with key eid
  - and FD dept \rightarrow deptname

Q: Although eid is the key for this table ... is it still possible for there to be 2 names for the same department?

• YES!

• The DBMS can enforce keys, but not non-key FDs

What are possible non-key, non-trivial FDs in these examples?

Customer(cid, address, city, state, zip)
  - zip \rightarrow state
  - address, city, state \rightarrow zip

Enrollment(student_id, class_id, instructor_id, student_name, instructor_name)
  - instructor_id \rightarrow instructor_name
  - student_id \rightarrow student_name