Database Management Systems

*Project 4*

|  |  |
| --- | --- |
| COURSE | Database Management Systems |
| SECTION | 001 |
| SEMESTER | Fall 2015 |
| INSTRUCTOR | Dr. Le Gruenwald |
| GROUP PROJECT NUMBER | 4 |
| GROUP MEMBERS | David Findley, Coleman Jackson, Swati Nemade, Stephen Smart |
| SCORE |  |

**Problem 1:**

1. Our only 2 potential minimal candidate keys are [manager] and [classid, id, gender]. we attempt closure under manager these.

(manager)+

result := manager

result := manager, gender, age, classid, id (given FD)

result := manager, gender, age, classid, id, name (id -> name)

result := manager, gender, age, classid, id, name, salary (gender, id, classid -> salary)

Manager is a minimal super key or a candidate key on EMPLOYEE

(classid, id, gender)+

result := classid, id, gender

result := classid, id, gender, salary, manager (given FD)

result := classid, id, gender, salary, manager, name (id -> name)

result := classid, id, gender, salary, manager, name, age (manager -> age)

(classid, id, gender) is a minimal super key or candidate key.

1. 1NF: employee is already in first normal form because all attributes are atomic, no other requirements.

2NF: We are not in second normal form because [id] a prime attribute implies [name] a non prime attribute, while [name] is implied by candidate key [classid, id, gender]

3NF: employee is not in third normal form because there are 3 FD’s that violate the rules of 3NF (1) name -> age (2) name -> id (3) id -> name, all three right attributes are not prime, all three are not trivial, and manager is our only superkey.

BCNF: employee is not 3NF therefore it is not BCNF, all three examples given are not in BCNF for the same reasons.

1. The lowest normal form not met by employee is 2NF, therefore we will attempt to rewrite employee as a 2NF relation.

id -> name does not hold under 2NF, therefore

R = {employee - {name}, {id, name}}

EMPLOYEE(id, age, classid, gender, manager, salary) CandKey {(manager), (CIG)}

FD{(classid, id, gender) -> (salary, manager),

(manager) -> (gender, age, classid, id)}

EMPLOYEE\_NAME(id, name) CandKey {(id), (name)}

FD{(id) -> (name)}

FD{(name) -> (age, id)} is lossy in this decomposition.

All non prime attributes in our new tables are totally dependent on the candidate keys.

1. Our new tables share the attribute id which is unique, therefore, there will be no spurious tuples in our join.
2. Our decomposition is not dependency preserving because the dependency (name) -> (age, id) requires a join to check in our new tables.

**Problem 2:**

* Recipies have a unique id, a title, author, instructions, category, and rating. The category of the recipe depends on the author.
* Chefs have a unique id, name, speciality, age, rank, country. Their speciality is determined by their country of origin.
* Chefs cook recipes and the date, cook\_time, and restaurant is recorded. The location depends on the date, because each day of the week the Chefs work at a certain restaurant

Recipe FD { (id) -> (title, author, instructions, category, rating), (author) -> (category)}

Chef FD { (id) -> (name, speciality, age, rank, country), (country) -> (speciality)}

Cooks FD {(chef\_id, recipe\_id, date) -> (cook\_time, restaurant), (date) -> (restaurant)}

3NF Relational Schema

--------------------------------------------------------------

chef(id, name, age, rank, country)

country\_speciality(country, speciality)

recipe(id, title, author, instructions, rating)

author\_category(author, category)

cooks(chef\_id, recipe\_id, date, cook\_time)

restaurant\_date(date, restaurant)

Schema Choice Explanation:

----------------------------------------------------------------

All attributes are assumed atomic to meet 1NF.

In order to satisfy 2NF we moved restaurant out of the cooks relation because restaurant is determined by date alone, meaning that the attribute is not fully dependent on the primary key.

In order to satisfy 3NF requirements the FD (country) -> (speciality) had to be decomposed because 1) It is not trivial, 2) country is not a superkey, and 3) speciality is not a prime attribute.

Along the same line, the FD (author) -> (category) was not 3NF because author is not a superkey, and category is not a prime attribute.

This table conforms to 3NF, therefore to 1NF and 2NF.

**Problem 3:**

1. Factors:
   * Accurate and reliable information
   * legal right to use and store information
   * whether we should differentiate between information retrieved from the web and information provided by OEM.

1) Users:

Factor 1 impacts users because false information will mislead users.

Factor 2 impacts users because if they are using our information that we are not allowed use, then they will believe that they will have approval to use such information.

2) OEM:

Factor 1 impacts OEM because it will reflect negatively if they are providing inaccurate

information to users.

Factor 2 impacts OEM because they will be legally culpable for using information they

were not allowed to use.

Factor 3 impacts OEM because the data generated by OEM is the fruit of their hard

labor. Failure to represent their work explicitly on their own website will do a

disservice to them and represent all work as equally valid, which may not be true.

3) Society:

Factor 2 impacts society because we must maintain a high standard of integrity

because if we are a weak link in data management then we are contributing to a

societal problem of data misuse.

b. For our example we chose to represent tornado data including date, f scale, cost in

millions of dollars, and location.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tornadoes (TABLE T)** | | | |
| **Date** | **F\_Scale** | **Cost\_in\_Millions** | **Location** |
| **May 05, 2015** | **4** | **100** | **Lawton, OK** |
| **April 10, 2010** | **2** | **20** | **Ardmore, OK** |
| **July 02, 2012** | **4** | **.00002** | **Broken Bow, OK** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tornadoes (TABLE T1)** | | |  | **Tornado\_Damage (TABLE T2)** | |
| **Date** | **F\_Scale** | **Location** |  | **F\_Scale** | **Cost\_in\_Millions** |
| **May 05, 2015** | **4** | **Lawton, OK** |  | **4** | **100** |
| **April 10, 2010** | **2** | **Ardmore, OK** |  | **2** | **20** |
| **July 02, 2012** | **4** | **Broken Bow, OK** |  | **4** | **.00002** |

|  |  |  |  |
| --- | --- | --- | --- |
| **INNER JOIN (T1 to T2) on F\_Scale** | | | |
| **Date** | **F\_Scale** | **Cost\_in\_Millions** | **Location** |
| **May 05, 2015** | **4** | **100** | **Lawton, OK** |
| **May 05, 2015** | **4** | **.00002** | **Lawton, OK** |
| **July 02, 2012** | **4** | **.00002** | **Broken Bow, OK** |
| **July 02, 2012** | **4** | **100** | **Broken Bow, OK** |
| **April 10, 2010** | **2** | **20** | **Ardmore, OK** |

1) Users: This impacts users because we are now generating insignificant data. This data has no purpose even in aggregation or as a representation of a researcher's findings, this data can’t be used for *anything* and we are responsible for creating it. Users who look through this data are being completely misled.

2) OEM: Databases often represent the only permanent storage for important and expensive data, especially in applications such as this. Storing this data with a lossy join will *ruin* the data that belongs to OEM and without a backup it can never be retrieved. This makes us responsible to OEM for destroying the information they gave us to manage.

3) Society: Data like this that is stored publically was likely funded by taxpayer dollars. The public users of our database are owners of the data more so than even OEM so our failure to keep their data represents waste of taxpayer money and time. In keeping a lossy join we become responsible for valuable information being irretrievably ruined.