ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Systems@ETH Zürich

ETH Zurich
Systems Group
Prof. D. Kossmann

FS 2013
Data Modeling and Databases
Exercise Sheet 6

Normal Forms II - Solution

1 Normal Forms: Theory Questions

1. 2NF requires that non-key attributes depend on all the attributes comprising a key (and not a proper subset). What additional restriction does 3NF impose?
   Clarification: The superkey is a superset of a minimum key. In 3NF, non-key attributes must depend only on all the attributes comprising a key and nothing else. They are not allowed to depend on non-key attributes alone. So, let's say we have R(ABCD) with AB being the key. Some of them are allowed to exist without violating the 3NF rules.

   (a) $AB \rightarrow C$ ✓
   (b) $A \rightarrow D$ No
   (c) $B \rightarrow A$ ✓- A is part of a key
   (d) $B \rightarrow D$ No
   (e) $ABD \rightarrow C$ ✓

2. (a) Does a relation that complies with 3NF also comply with BCNF?
   Not necessarily. The difference is that for BCNF-compliant relations, **all** attributes (key and non-key attributes) must depend on a whole key and nothing else.

   (b) Can any schema be decomposed into BCNF? At what price?
   Every schema can be decomposed into BCNF while maintaining the lossless join property by using a distinct algorithm. However, it is not guaranteed that the functional dependencies present in the initial table will be preserved. So, valid **BCNF is not always achievable**.

   (c) A relation with attributes A, B, C and the following functional dependencies:
   $AB \rightarrow C, C \rightarrow B$ is not in BCNF.
   Why not?
   Keys of the relation are: A,B ($AB \rightarrow C$) and A,C ($C \rightarrow B, CA \rightarrow B$)
   However, the attribute B which is part of a key depends on just C and not AC. That is, it does not depend on all attributes of a key .

   (d) If we try to decompose it, what will happen?
   If (according to the decomposition algorithm) we decompose R(A,B,C) in relations (A,B) and (B,C), the functional dependency ($AB \rightarrow C$) cannot be preserved.

3. (a) How are multivalued dependencies different than functional dependencies regarding the information they provide about the schema?
   Functional dependencies provide information about what kind of tuples are **not** allowed to exist, that is, if $A \rightarrow B$ then we can't have two tuples with A=3, B=6 and A=3, B=7. There can be only one value for B for every given value of A. Multivalued dependencies, on the other hand provide the information that a set of tuples **must** exist in a relation.

(b) What conditions must the multivalued dependencies (if any) satisfy for a relation to be in 4NF?
A Table is in 4NF if and only if, for every one of its non-trivial multivalued dependencies $X \twoheadrightarrow Y$, X is a superkey, that is, X is either a candidate (minimal) key or a superset thereof.

(c) Is a decomposition to 4NF always dependency preserving and/or lossless?
If we use the algorithm presented in the lecture we can always achieve preservation of the lossless join property.However, **dependency presevation** is not always achieavable.

4. (a) Give an example of a relation that is in second normal form but not in third normal form. Draw the relational diagram and list all functional dependencies. Explain why it is in 2NF and not in 3NF.

Answer:

SUPPLIERS(supplier_no, status, city)

Functional Dependencies: $supplier\_no \rightarrow status$, $supplier\_no \rightarrow city$, $city \rightarrow status$

Comments:

Lacks mutual independence among non-key attributes.

Mutual dependence is reflected in the transitive dependencies: $supplier\_no \rightarrow city, city \rightarrow status$.

Anomalies:

INSERT: We cannot record that a particular city has a particular status until we have a supplier in that city.

DELETE: If we delete a supplier which happens to be the last row for a given city value, we lose the fact that the city has the given status.

UPDATE: The status for a given city occurs many times, therefore leading to multiple updates and possible loss of consistency.

(b) Give an example of a relation that is not in 2NF. Draw the relational diagram and list all functional dependencies. Explain why it is not in 2NF. Explain how it can be transformed into a table that is in 2NF.

Answer:

SUPPLIER(supplier_no, status, city, part_no, quantity)

Functional Dependencies: $(supplier\_no, part\_no) \rightarrow quantity$, $(supplier\_no) \rightarrow status$, $(supplier\_no) \rightarrow city$, $city \rightarrow status$ (Supplier's status is determined by location)

Comments:

Non-key attributes are not mutually independent ($city \rightarrow status$).

Non-key attributes are not fully functionally dependent on the primary key (i.e., status and city are dependent on just part of the key, namely supplier_no).

Anomalies:

INSERT: We cannot enter the fact that a given supplier is located in a given city until that supplier supplies at least one part (otherwise, we would have to enter a null value for a column participating in the primary key C a violation of the definition of a relation).

DELETE: If we delete the last (only) row for a given supplier, we lose the information that the supplier is located in a particular city.

UPDATE: The city value appears many times for the same supplier. This can lead to inconsistency or the need to change many values of city if a supplier moves.

2 2NF

Consider the following relational schema for LINEITEM:
LINEITEM (OrderNumber, ItemNumber, Description, Price, Quantity)

1. Find the functional dependencies and a key of the relation above.
   $ItemNumber \rightarrow Description, Price$
   $OrderNumber, ItemNumber \rightarrow Quantity$ This is a key.

2. What normal form is the above LINEITEM relation in ?
   The description and the price depend on parts of the key (just the ItemNumber) and not on the whole key. Therefore the relation is not in 2NF. It is in 1NF though.

3. What are some disadvantages of this choice of schema?
   The item description and price are stored unnecesarily for each instance of the particular item in the LINEITEM relation, which might lead to inconsistencies. On the other hand, a positive effect of this is that, in order to find the total price of items, no join is needed.

3 3NF

Consider the following relational schema (CONCERT) describing musical events in Switzerland:

Venue	Year	Singer	Genre
X	1999	Cher	pop
Z	1999	Cher	pop
Y	2001	Cher	pop
Y	2001	Porcupine Tree	rock

Is this in 2NF? 3NF? Determine functional dependencies to prove your point (show your working).

$Singer \rightarrow Genre$
The key is $Singer, Year, Venue$. The genre depends on the singer which is part of the key. So the relation is not even in 2NF.

Now look at this slightly different schema:

Venue	Year	Singer	Number of Attendees
X	1999	Cher	10 000
Z	1999	Cher	8 000
Y	2001	Cher	9 0000
Y	2001	Porcupine Tree	10 000

What about this? Is this in 2NF? How about 3NF?
$Venue, Year, Singer \rightarrow Numberof Attendees$. Yes. The only non-key attribute (Number of Attendees) depends on the whole key and nothing else.

The point of this exercise is to show the difference between storing superfluous information in a table (genre of music) and information that is really needed because it differs from tuple to tuple (number of attendees).

Assumptions: One singer per different concert, and singers stick to just one genre of music.
Also, singers do not visit the same venue twice in the same year.

4 Synthesis Algorithm

Consider the following relation:

$$R(A, B, C, D)$$

with the following functional dependencies (there are no further non-trivial functional dependencies).

$$A \to D$$
$$A, B \to C$$
$$A, C \to B$$

1. Specify the candidate key of this relation.

   Answer

   Closure(F, AB) = (AB), (ABD), (ABCD) ⇒ AB = Super-Key
   Closure(F, A) = (A),(AD)
   Closure(F, B) = (B)
   ⇒ AB = Candidate Key

   Closure(F, AC) = (AC),(ACD),(ABCD) ⇒ AC = Super-Key
   Closure(F, C) = (C)
   ⇒ AC = Candidate Key

2. Apply the synthesis algorithm to transform the schema into 3NF (loss and dependency preserving).

   Answer

   FC = F
   A → D ⇒ R1=A,D
   AB → C ⇒ R2 = A,B,C
   AC → B ⇒ R3 = A,C,B
   R3 ⊆ R2 ⇒ R1=A,D; R2 = A,B,C

5 Decomposition Algorithm

Consider the following relations and functional dependencies:

1. $S1(A, B, C, D)$ with functional dependencies:

   $$A, C \to D, \qquad A \to B$$

2. $R1(A, B, D, E)$; $R2(A, C, F)$ with functional dependencies:

   $$A \to B, E \qquad A \to D \qquad F \to A \qquad A, C \to F \qquad B, C \to E \qquad C \to A$$

3. $S2(A, B, C)$ with functional dependencies:

   $$A, B \to C \qquad C \to A$$

   (a) Determine all the candidate keys
   (b) In which normal form are the relations?
   (c) Transfer the relations in 3NF
   (d) Are the resulting relations in BCNF? If not, convert them to BCNF.

Answer

1. $S1(A, B, C, D)$ with functional dependencies:

$$A, C \to D, \qquad A \to B$$

The only key candidate is the pair (A, C).
$S1$ is in 1NF as there are no multi-value attributes.
$S1$ is not in 2NF as B is not entirely dependent on (A, C).

The decomposition of $S1(A, B, C, D)$ into $S11(A, B)$ and $S12(A, C, D)$ is in 2NF, 3NF and BCNF.

2. $R1(A, B, D, E)$; $R2(A, C, F)$ with functional dependencies:

$$A \to B, E \qquad A \to D \qquad F \to A \qquad A, C \to F \qquad B, C \to E \qquad C \to A$$

(a) To determine the key candidates, let's first compute the minimal basis:

Left reduction:
Reduce $AC \to F$ to $C \to F$ because $\{F\} \subseteq Closure(FD, C) = (C), (AC), (ACF)$
Reduce $BC \to E$ to $C \to E$ because $\{E\} \subseteq Closure(FD, C) = (C), (AC), (ABCE)$

Right reduction:
Reduce $C \to E$ to $C \to \oslash$ because $\{E\} \subseteq Closure(FD \backslash \{C \to E\}, C) = (C), (AC), (ABCE)$
Reduce $C \to A$ to $C \to \oslash$ because $\{E\} \subseteq Closure(FD \backslash \{C \to A\}, C) = (C), (CF), (ACF)$

After applying the union rule to FDs with the same left side, remains as minimal basis:

$$A \to BDE \qquad F \to A \qquad C \to F$$

A is the candidate key of $R1$, C is the candidate key of $R2$.

(b) Normal forms:

$R1$:
1NF yes
2NF yes - BDE is dependent on A
3NF yes - A is superkey
BCNF yes - A is superkey

$R2$:
1NF yes
2NF yes - AF is dependent on C
3NF no - $F \to A$ is not trivial. A is not part of the candidate key and F is not a superkey
BCNF no - Not in 3NF

The decomposition of $R2(A, C, F)$ into $R21(C, F)$ and $R22(A, F)$ is in 3NF and BCNF.

Note: The synthesis algorithm is not defined for two or more relations. An alternative solution would be to join the relations $R1$ and $R2$ and then apply the algorithm. The result would be identical.

3. $S2(A, B, C)$ with functional dependencies:

$$A, B \to C \qquad C \to A$$

The two key candidates for $S2$ are (A, B) and (B, C)
$S2$ is in 1NF as there are no multi-value attributes
$S2$ is in 2NF as there are no non-key attributes
$S2$ is in 3NF as A, B is a superkey, and A is part of a candidate key
$S2$ is not in BCNF as in FD $C \to A$, C is not a superkey

The decomposition of $S2(A, B, C)$ into $S21(B, C)$ and $S22(A, C)$ is in BCNF.
The decomposition is lossless (both relations can be joined), but the FD ABC is not preserved.

6 Normalization up to BCNF

Consider the following relation:

$$Shipping(ShipName, ShipType, TripId, Cargo, Port, Date)$$

and the following functional dependencies:

$$ShipName \to ShipType$$
$$TripId \to ShipName, Cargo$$
$$ShipName, Date \to TripId, Port$$

or, expanded:

$$ShipName \to ShipType$$
$$TripId \to ShipName$$
$$TripId \to Cargo$$
$$ShipName, Date \to TripId$$
$$ShipName, Date \to Port$$

and also, we can infer

$$TripId, Date \to Port$$

1. Find the candidate key(s).
   ShipName,Date AND TripId,Date.

2. Normalize to 2NF.
   This relation is not in 2NF because ShipType depends on part of a candidate key (ShipName,Date) and also Cargo depends on part of (another) candidate key (TripId,Date). Note that it's okay for ShipName to depend on TripId for 2NF, because its an attribute that belongs to a key. We split the relation into
   SHIPS(ShipName, ShipType) with $ShipName \to ShipType$, TRIPS(ShipName, TripId, Port, Date) with $ShipName, Date \to TripId, Port$, and $TripId \to ShipName$,

   and CARGO(TripId,Cargo) with $TripId \to Cargo$.

3. Normalize to 3NF.
   The TRIPS table has the functional dependencies: $TripId \to ShipName$ and $ShipName, Date \to TripId, Port$. The candidate keys are ShipName, Date and TripId,Date. There is no 3NF - violating functional dependency here, or in either of the other tables resulting from the previous decomposition.

4. Normalize to BCNF.
   There still is a problem because ShipName depends on TripId (TripId being part of a candidate key). So we split TRIPS into

   TRIPDATES (TripId, Port, Date)
   $TripId, Date \to Port$

TRIPSHIPS (TripId, ShipName)
$TripId \rightarrow ShipName$ Note that here, for this particular BCNF decomposition, the functional dependency $ShipName, Date \rightarrow TripId$ is not preserved. Neither is $ShipName, Date \rightarrow Port$