Lecture Notes on Database Normalization

Chengkai Li
Department of Computer Science and Engineering
The University of Texas at Arlington

April 15, 2012

I decided to write this document, because many students do not have the textbook, the slides themselves won't be sufficient, and we did not have enough time in lecture to elaborate it in further details. My goal is to summarize the concepts we learned and explain various points about normalization through examples. These examples can help you solve similar problems in homework and exam. To thoroughly understand these topics, you should read the textbook.

1 Keys of a Relation and Transitive Closure

1.1 Concepts and Procedures

Concept 1 (superkeys, candidate keys (keys), primary key, secondary keys) Here we summarize various concepts related to keys.

Superkeys: A set of attributes in a relation is a superkey if it can determine the rest of the attributes in the relation. In other words, given a tuple, if we know its values on all the attributes in a superkey, then we can determine its values on all remaining attributes. Under set semantics, it means we can uniquely identify a tuple by its superkey value; under bag semantics, it means we can identify the identical tuples by their superkey value.

Candidate keys (also simply called *keys*): A candidate key is a superkey and none of its proper subset is a superkey. In other words, a candidate key is a minimal superkey. It cannot be made smaller. Removing any attribute from it will disqualify it as a superkey. (Hence any proper superset of a candidate key is not a candidate key, and any proper subset of a candidate key is not a candidate key either.)

Primary key: If a relation has multiple candidate keys, then one of the candidate keys is designated as the primary key.

Secondary keys: If a relation has multiple candidate keys, except the primary key, every other candidate key is a secondary key.

Concept 2 (prime attribute, nonprime (nonkey) attribute) An attribute is a *prime attribute* if it belongs to any candidate key. (Note that we cannot replace "candidate key" by "superkey" in this concept, because the set of all attributes of a relation is also a superkey, making every attribute belong to at least one superkey.)

An attribute is a *nonprime (nonkey) attribute* if it is not a prime attribute, i.e., it is not part of any candidate key.

Note: The Table 15.1 in the textbook (and in the slides of Chapter 15) is not fully rigorous. For 3NF, it says "Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes)." It should be "Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of attributes that is not a superkey)." The original statement does not cover the case in which a nonkey attribute

is functionally determined by a set of attributes containing both nonkey and prime attributes but is not a superkey.

Concept 3 (Functional Dependency) The concept of *functional dependency* (FD) was repeatedly explained in lectures and you can find its definition and explanation in slides and textbook. It is very important for understanding all other concepts.

A note to make is that given any superkey (and thus candidate key) X in a relation schema and any set of attributes Y in the schema, there exists an FD $\mathsf{X} \to \mathsf{Y}$. (The FD is nontrivial if Y does not overlap with X.)

Concept 4 (Transitive Closure) Given a set of attributes X, the *transitive closure* of X, denoted by X^+, is the set of attributes that can be derived directly or transitively from X according to functional dependencies (FDs). To simplify the notation, we also use a^+ (instead of $\{\mathsf{a}\}^+$) to denote the transitive closure of a single attribute a.

Procedure 1 (How to compute transitive closure?) Given a set of attributes X, to compute X^+, we start with X^+=X. If there exits a functional dependency (FD) $\mathsf{A} \to \mathsf{B}$ such that $\mathsf{A} \subseteq \mathsf{X}^+$, we will make X^+=$\mathsf{X}^+ \cup \mathsf{B}$. We keep doing this until we reach a *fixed point*, i.e., X^+ does not change anymore.

A less formal and more intuitive way of describing the procedure is as follows. At the beginning, the transitive closure of X includes the attributes in X itself. We find functional dependencies (FDs) whose left-hand sides are included in the transitive closure and expand the transitive closure to include the right-hand sides. We keep doing this, until the transitive closure cannot be further expanded.

Procedure 2 (How to find candidate keys by transitive closure?) Given a set of attributes X in a relation schema, if its transitive closure X^+ contains all attributes in the schema, then X is a superkey. Furthermore, if none of X's proper subsets can be a superkey, then X is a candidate key.

1.2 Examples

Problem 1 Consider the following FDs for relation schema $\mathbf{R}(\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e})$: $\mathsf{a} \to \mathsf{bc}$; $\mathsf{cd} \to \mathsf{e}$; $\mathsf{b} \to \mathsf{d}$; $\mathsf{e} \to \mathsf{a}$. List all candidate keys for $\mathbf{R}$.

Solution 1 To find candidate keys, we can compute the transitive closure of every possible subset of the attributes, i.e., every possible left-hand side of an FD.

Compute a^+: Begin with a^+=$\{\mathsf{a}\}$; expand a^+ to $\{\mathsf{abc}\}$ by FD $\mathsf{a} \to \mathsf{bc}$; further expand it to $\{\mathsf{abcd}\}$ by FD $\mathsf{b} \to \mathsf{d}$; further expand it to $\{\mathsf{abcde}\}$ by FD $\mathsf{cd} \to \mathsf{e}$. Hence a^+=$\{\mathsf{abcde}\}$.

Compute b^+: b^+=$\{\mathsf{b}\}$; then b^+=$\{\mathsf{bd}\}$ by FD $\mathsf{b} \to \mathsf{d}$.

Compute c^+: c^+=$\{\mathsf{c}\}$. It cannot be expanded. As we can see, none of the FDs has c only in the left-hand side.

Compute d^+: d^+=$\{\mathsf{d}\}$.

Compute e^+: e^+=$\{\mathsf{e}\}$; then e^+=$\{\mathsf{ae}\}$ by FD $\mathsf{e} \to \mathsf{a}$; we already know a^+=$\{\mathsf{abcde}\}$, therefore e^+=$\{\mathsf{abcde}\}$.

Since a^+=$\{\mathsf{abcde}\}$ and e^+=$\{\mathsf{abcde}\}$, we know that both a and e are candidate keys.

To find remaining candidate keys, we can continue the enumeration of all possible combinations of attributes and compute their transitive closures. However, we can avoid the enumeration of several cases, by making the following observation.

Remember we showed in Concept 1 that any proper superset of a candidate key cannot be a candidate key. Instead, such a proper superset is only a superkey. Hence, we can conclude that any proper superset of a or e is not a candidate key. Based on this observation, any multi-attribute candidate key must not contain a or e.

For finding possible two-attribute candidate keys, we only need to look at bc, bd, and cd.

$\{bc\}^+$: First $\{bc\}^+=\{bc\}$; then $\{bc\}^+=\{bcd\}$ by $b \to d$; furthermore $\{bc\}^+=\{bcde\}$ by $cd \to e$; finally $\{bc\}^+=\{abcde\}$ by $e \to a$. Therefore bc is a candidate key.

$\{bd\}^+$: First $\{bd\}^+=\{bd\}$. It cannot be further expanded. (Another way of reasoning about it: We have FD $b \to d$. Hence $\{bd\}^+$ is equal to b^+, which we have already computed.)

$\{cd\}^+$: First $\{cd\}^+=\{cd\}$; then $\{cd\}^+=\{cde\}$ by $cd \to e$. Based on the previous steps, we already know e is a candidate key. Hence cd is also a candidate key.

For finding possible three-attribute candidate keys, we only need to consider bcd. However, it cannot be a candidate key, given that bc (cd) is a candidate key.

In summary, a, e, bc, cd are all the candidate keys.

Problem 2 Consider the following FDs for relation schema $\mathbf{R}(a, b, c, d, e)$: $ab \to c$; $cd \to e$; $de \to b$. List all candidate keys for $\mathbf{R}$.

Solution 2 Again, we can enumerate all subsets of the schema including itself and compute the transitive closures of these subsets. However, we can save time and avoid such exhaustive examination, by following some key observations.

Note that none of the FDs has a or d in its right-hand side. In other words, a and d cannot be derived from any other attributes. Hence any candidate key must contain both a and d.

Consider possible 2-attribute candidate keys, i.e., ad. $ad^+=\{ad\}$. It is not a candidate key.

Consider possible 3-attribute candidate keys, i.e., abd, acd, ade. $abd^+=\{abcde\}$, $acd^+=\{abcde\}$, $ade^+=\{abcde\}$. All of them are candidate keys.

It is easy to see that there cannot be any 4-attribute or 5-attribute candidate keys. As mentioned above, a candidate key must contain both a and d. Thus a 4-attribute or 5-attribute candidate key must be a superset of abd, or acd, or ade, which are all candidate keys. That would disqualify it from being considered a candidate key.

2 Normal Forms

In defining normal forms, we only consider nontrivial FDs. An FD $X \to Y$ is trivial if X subsumes Y, i.e, $X \supseteq Y$.

We also only consider simple FD in which the right-hand side is a single attribute. Thus we will represent it by $X \to y$, where the lower case y indicates it is a single attribute. An FD with multiple attributes in the right-hand side can be split into individual simple FDs.

2NF: The following statements are all equivalent. They all define what 2NF is about.

- *A relation is in 2NF if every nonprime attribute is fully functional dependent on every candidate key.*
- *A relation is in 2NF if there does not exist an FD* $X \to y$ *such that* y *is a nonprime attribute and* X *is a proper subset of a candidate key.*
- *A relation is in 2NF if there does not exist a nonprime attribute* y *that is partially dependent on any candidate key.*

3NF: The following statements are all equivalent. They all define what 3NF is about.

• *A relation is in 3NF if it is in 2NF* ***and*** *there does not exist a nonprime attribute* y *that is transitively dependent on any candidate key.*

• *A relation is in 3NF if there does not exist a nonprime attribute* y *that is dependent on any set of attributes that is not a superkey.*

• *A relation is in 3NF if given any FD* $\mathsf{X} \rightarrow \mathsf{y}$*, either* y *is a prime attribute or* X *is a superkey.*

BCNF: The BCNF condition is really simple.

• *A relation is in BCNF if given any FD* $\mathsf{X} \rightarrow \mathsf{y}$*,* X *is a superkey.*

In practice, we often want our relations in BCNF or 3NF. Higher normal forms 4NF and 5NF are not pursued in practice. The lower one 2NF is often insufficient.

3 Decomposition

3.1 Decomposition Algorithms

Procedure 3 (How to determine if a relation is in 2NF, 3NF, or BCNF?) Find all candidate keys based on given FDs. Then check if any FD violates the corresponding normal form, according to the definitions of 2NF, 3NF, and BCNF.

Proposition 1 *Any relation schema with two attributes is in BCNF.*

Proof 1 We have proved this in class. It is also a question in HW3. We will provide the proof in HW3 solution.

Procedure 4 (How to decompose a relation R into a set of relations in BCNF?)

1. *To start with, pick an FD* $\mathsf{X} \rightarrow \mathsf{y}$ *that violates BCNF in* $\boldsymbol{R}$*.*
2. *Compute* X^+*, the transitive closure of* X*.*
3. *Decompose* $\boldsymbol{R}$ *into* $\boldsymbol{R1}$ *and* $\boldsymbol{R2}$*.* $\boldsymbol{R1}$ *contains all attributes in* X^+*,* $\boldsymbol{R2}$ *contains the set of attributes* $(\boldsymbol{R} - \mathsf{X}^+) \cup \mathsf{X}$*, which is also equivalent to the set of attributes* $\boldsymbol{R} - (\mathsf{X}^+ - \mathsf{X})$*. In other words,* $\boldsymbol{R2}$ *contains* X *itself and all attributes outside of the transitive closure of* X*.*
4. *Find which original FDs in* $\boldsymbol{R}$ *are still applicable (i.e., preserved) in* $\boldsymbol{R1}$ *and* $\boldsymbol{R2}$*. Find all candidate keys of* $\boldsymbol{R1}$ *and* $\boldsymbol{R2}$ *based on the preserved FDs in these relations, and find if they are in BCNF.*
5. *If* $\boldsymbol{R1}$ *or* $\boldsymbol{R2}$ *is not in BCNF, recursively decompose the relation, until all relations are in BCNF. (It is guaranteed that we will eventually have a set of relations that are all in BCNF. See Proposition 1.)*

3.2 Lossless Join Property and Perseverance of FDs

There are two consequences of decomposition:

(1) The original schema is decomposed into multiple relations. Thus the attributes are not all together anymore. Some of the original FDs may be reserved in one or more of the new relations, i.e., they can be checked in these relations. Some FDs may be lost. **It is not always possible to preserve all FDs.**

(2) The tuples in the original relation are projected into the new relations. We may wonder– *Can we recover the original relation?* In other words, do we get exactly the same relation if we perform natural joins of the new relations? After all, we will join these relations for various queries that are supposed to be executed on the original single relation.

Proposition 2 If we follow the above procedure to decompose a relation into BCNF relations, it is guaranteed that the original relation can be recovered *losslessly*, i.e., we can obtain the original relation exactly by natural joins of the new relations.

In HW3, through one question, you will see that if we do not follow the above procedure (e.g., if we decompose a relation in a fashion that is not based on a violating FD), then the *lossless* property is not guaranteed. We may not get back the exact original relation.

Proposition 3 By following an algorithm of 3NF decomposition, a relation can be decomposed into 3NF relations such that (1) the lossless join property is guaranteed and (2) all FDs are preserved. We will not discuss the details of this algorithm. Instead, I encourage you to apply the BCNF decomposition algorithm, but use an FD that violates 3NF to start with. Although it is not the same as the 3NF decomposition algorithm, you get some exercise and better understanding by checking if join is lossless and if FDs are preserved.

3.3 Examples

Problem 3 Consider the following FDs for relation schema $\mathbf{R}(\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d})$: $\mathsf{ab}\rightarrow\mathsf{c}$; $\mathsf{c}\rightarrow\mathsf{d}$; $\mathsf{d}\rightarrow\mathsf{a}$.
(1) Find all candidate keys.
(2) Find all BCNF violations.
(3) Decompose R into relations in BCNF.
(4) What FDs are not preserved by BCNF.

Solution 3
(1) b is not in the right-hand side of any of the given FDs. Hence every candidate key must contain b. b^+=$\{\mathsf{b}\}$. Hence b itself is not a candidate key. We need to consider its supersets. ab^+=$\{\mathsf{abcd}\}$, bc^+=$\{\mathsf{abcd}\}$, bd^+=$\{\mathsf{abcd}\}$. All of them are candidate keys. And thus there cannot be any candidate key with more than 2 attributes.

(2) ab, bc, and bd are candidate keys. Let's check each of the given FDs:
$\mathsf{ab}\rightarrow\mathsf{c}$: left-hand side is a candidate key, and thus a superkey. Hence it does not violate BCNF.
$\mathsf{c}\rightarrow\mathsf{d}$: left-hand side is not a superkey. Hence it violates BCNF.
$\mathsf{d}\rightarrow\mathsf{a}$: left-hand side is not a superkey. Hence it violates BCNF.

(3) We can decompose by starting with one of the violating FDs.

(3.1) Start with $\mathsf{c}\rightarrow\mathsf{d}$: The transitive closure of c is c^+=$\{\mathsf{a},\mathsf{c},\mathsf{d}\}$. Hence $\mathbf{R}$ can be decomposed into $\mathbf{R1}(\mathsf{b},\mathsf{c})$ and $\mathbf{R2}(\mathsf{a},\mathsf{c},\mathsf{d})$. For $\mathbf{R1}(\mathsf{b},\mathsf{c})$, we know a 2-attribute relation schema must be in BCNF. In $\mathbf{R2}(\mathsf{a},\mathsf{c},\mathsf{d})$, the original FDs $\mathsf{c}\rightarrow\mathsf{d}$ and $\mathsf{d}\rightarrow\mathsf{a}$ are preserved. The only candidate key is c. Hence there is a transitive dependency: $\mathsf{c}\rightarrow\mathsf{d}$ and then $\mathsf{d}\rightarrow\mathsf{a}$. a is a nonprime attribute in $\mathbf{R2}$. Thus the FD $\mathsf{d}\rightarrow\mathsf{a}$ in $\mathbf{R2}$ violates 3NF (and thus BCNF too). By this violating FD, we will further decompose $\mathbf{R2}$ into $\mathbf{R3}(\mathsf{c},\mathsf{d})$ and $\mathbf{R4}(\mathsf{a},\mathsf{d})$. Both $\mathbf{R3}$ and $\mathbf{R4}$ have only 2 attributes. Therefore they are in BCNF. Hence the original $\mathbf{R}$ is decomposed into $\mathbf{R1}$, $\mathbf{R3}$, $\mathbf{R4}$ so that all of them are in BCNF.

(3.2) Start with $\mathsf{d} \rightarrow \mathsf{a}$: The transitive closure of d is $\mathsf{d}^+ = \{\mathsf{a}, \mathsf{d}\}$. Hence $\mathbf{R}$ can be decomposed into $\mathbf{R5}(\mathsf{b}, \mathsf{c}, \mathsf{d})$ and $\mathbf{R6}(\mathsf{a}, \mathsf{d})$. **R6** must be in BCNF. In $\mathbf{R5}(\mathsf{b}, \mathsf{c}, \mathsf{d})$, $\mathsf{c} \rightarrow \mathsf{d}$ is preserved. Hence b, c is the only candidate key in **R5**. $\mathsf{c} \rightarrow \mathsf{d}$ violates 2NF (and thus 3NF and BCNF as well) in **R5** since c is only part of a candidate key and d is a nonprime attribute. We further decompose **R5** by this violating FD into $\mathbf{R7}(\mathsf{b}, \mathsf{c})$ and $\mathbf{R8}(\mathsf{c}, \mathsf{d})$. Both are in BCNF. The original $\mathbf{R}$ is decomposed into **R6**, **R7**, **R8** so that all of them are in BCNF.

(4) If we follow the decomposition in (3.1) or (3.2), $\mathsf{ab} \rightarrow \mathsf{c}$ is not preserved in any of the resulting relations. As we can see, it is not always possible to preserve all original FDs during decomposition.