CS157A: Introduction to Database Management Systems

Chapter 3. Design Theory For Relational Databases

Design Anomalies

- Anomalies: problems caused by too much information is crammed into a single relation
  - Redundancy - capturing info multiple times
  - Update Anomalies – forget to update in a tuple
  - Deletion Anomalies – deleting a tuple causes a loss of other information as a side effect

Solution: Normalization

The goal of normalization is to decompose a relation into several in a way that the decomposition will have

- Elimination of Anomalies
- Recoverability of Information
- Preservation of Dependencies

Normalization

Relation + its functional dependencies → normalization → Relations in BCNF

Relation + its functional dependencies + its multivalued dependencies → normalization → Relations in 4th Normal Form

Idea of Normalization

- Define BCNF in terms of FD and key.
- From a given mega relation, discover all true FD's: closure algorithm
- Identify BCNF violations and decompose relations until no BCNF violation exists

Functional Dependencies

- Definition
  - In a relation *R*, a set of attributes A is said to **functionally determine** another set of attributes B ($A \rightarrow B$), if two tuples of R agree on A then they also agree on B.
  - R satisfies a FD if the FD is true for every instance of R
- Implication
  - Each A value is associated with precisely one *Y* value.
  - If the A value is known, then the B value corresponding to *A* can be determined by looking up in any tuple of R containing the A value.

Functional Dependencies

Suppose a relational schema is (<u>A</u>, <u>B</u>, <u>C</u>) and <u>A</u> → <u>B</u>

<u>A</u>	**<u>B</u>**	**<u>C</u>**
a	b	c1
a	b	c2

Example: Functional Dependencies

Based on the knowledge of the real world:

`Movies1(title, year, length, genre, studioName, starName)`

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

```
title year → length genre studioName (o)
title year → starName (x)
```

Keys

- In a relation R with no duplicates, if $\underline{A} \rightarrow$ all other attributes, then $\underline{A}$ is a key of R.
- Minimal key: no proper subset of $\underline{A}$ functionally determines all other attributes
- Super key: a set of attributes that contains a key.

Example: Keys

[Q] Is {title, year, starName} a key for Movie1 ?

[A] Yes. {title, year, starName} functionally determines all other attributes of Movie1.

[Q] Is the key minimal?

[A] Yes. No proper subset of the key can functionally determine all other attributes.

{title} {year} {starName} {title, year}

{title, starName}{year, starName}

Functional Dependency Rules

With a given set of FDs, we can deduce other functional dependencies using following rules.

- Splitting rule
- Combining rule
- Trivial dependency rules (two of them)
- Transitive rule

Functional Dependency Rule

Example: If we are told that a relation R (A, B, C) satisfies the FD's A→ B and B→ C, we can deduce that R also satisfies the FD A→ C.

- Let (a, b_1, c_1) and (a, b_2, c_2) be two tuples that agree on attribute A.
- Since R satisfies $A \rightarrow B$ it follows that $b_1=b_2$ so the tuples are: (a, b, c_1) and (a, b, c_2)
- Similarly, since R satisfies $B \rightarrow C$ and the tuples agree on B they will agree also on C. So, $c_1=c_2$.

The Splitting/Combining Rule involving the right side of FDs

$$A_1A_2 \ldots A_n \rightarrow B_1$$
$$A_1A_2 \ldots A_n \rightarrow B_2$$
$$\ldots$$
$$A_1A_2 \ldots A_n \rightarrow B_m$$

Combining Rule

$$\xrightarrow{\text{Combining Rule}} \quad A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m. \quad \xleftarrow{\text{Splitting Rule}}$$

Splitting Rule

Example: The Splitting/Combining Rule involving the right side of FDs

title year → length title year → genre title year → studioName	Combining Rule $\longrightarrow$ $\longleftarrow$ Splitting Rule	title year → length genre studioName

Can we split the left side ?

[Q]

From, title year $\rightarrow$ length, can we deduce

title$\rightarrow$ length (false FD)

year $\rightarrow$ length (false FD) ?

[A] No

Trivial Functional Dependencies

- A functional dependency $A_1A_2 \ldots A_n \rightarrow B$ is **trivial** if B is a subset of A

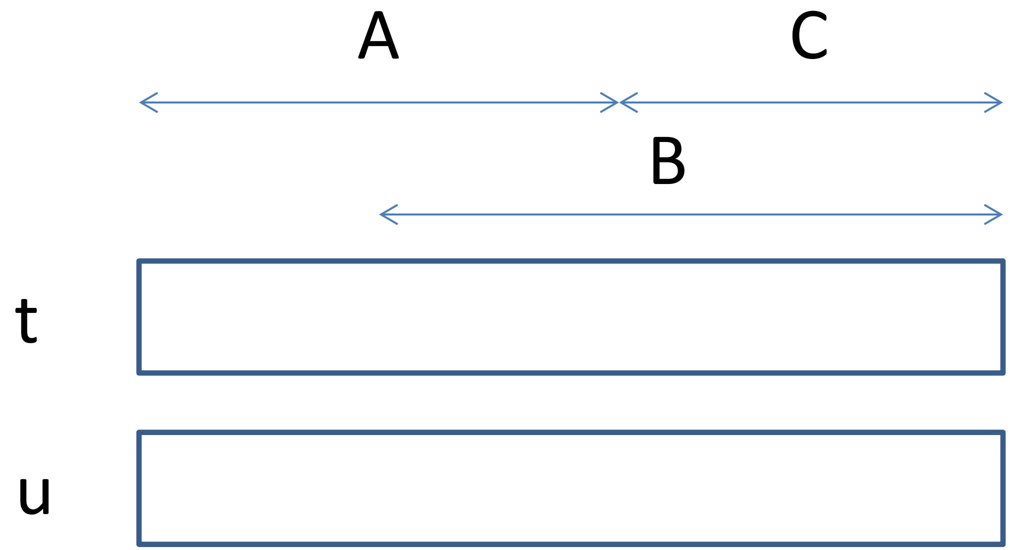
  Example: title year $\rightarrow$title

- A functional dependency $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$ is:
  - **Nontrivial** if at least one of the B's is not among the A's.
  - **Completely nontrivial** if A and B do not have any overlap.

  Example: title year $\rightarrow$ year length is nontrivial but not completely nontrivial.

Trivial Dependency Rule

- We can always remove from the right side of a FD those attributes that appear on the left.
- Suppose A → B. Then, A→ C



Computing the Closure of Attributes

- The closure of a set of attributes $\{A_1, A_2, \ldots, A_n\}$: $\{A_1, A_2, \ldots, A_n\}^+$
- Suppose $\{A_1, A_2, \ldots, A_n\}$ is a set of attributes and S is a set of FD's. $\{A_1, A_2, \ldots, A_n\}^+$ under the dependencies in S is the set of attributes B, which are functionally determined by $A_1, A_2, \ldots, A_n$
- That is, it finds $A_1A_2\ldots A_n \rightarrow B$ that follows from S
- Since we allow trivial dependencies, $A_1, A_2, \ldots, A_n$ are in $\{A_1, A_2, \ldots, A_n\}^+$.

Algorithm 3.7: Closure of a Set of Attributes

Input: A set of attributes $\{A_1, A_2, \ldots, A_n\}$ and a set of FD's S
Output: The closure $\{A_1, A_2, \ldots, A_n\}^+$

1 Let X be a set of attributes that eventually will become the closure. Initialize X to be $\{A_1, A_2, \ldots, A_n\}$.
2 Now, repeatedly search for some FD in S:

$$B_1B_2\ldots B_m \rightarrow C$$

such that all of B's are in the set X, but C is not. We then add C to X.
3 Repeat step 2 as many times as necessary until no more attributes can be added to X. (Since X can only grow, and the number of attributes is finite, eventually nothing more can be added to X.)
4 The set X after no more attributes can be added to it is the: $\{A_1, A_2, \ldots, A_n\}^+$.

Example

- Let's consider a relation with attributes A, B, C, D, E and F. Suppose that this relation satisfies the FD's:

  $AB \rightarrow C$, $BC \rightarrow AD$, $D \rightarrow E$, $CF \rightarrow B$.

  What is $\{A,B\}^+$?

$X = \{A,B\}$ Use: $AB \rightarrow C$

$X = \{A,B,C\}$ Use: $BC \rightarrow AD$

$X = \{A,B,C,D\}$ Use: $D \rightarrow E$

$X = \{A,B,C,D,E\}$ No more changes to X are possible so $X = \{A,B\}^+$.

- The FD: $CF \rightarrow B$ cannot be used because its left side is never contained in X.

Use of the closure of attributes

We can test whether any given functional dependency $A_1A_2\ldots A_n \rightarrow B$ **follows** from a set of dependencies **S**.

- Compute $\{A_1, A_2, \ldots, A_n\}^+$ using the set of dependencies S.
- If $B \in \{A_1, A_2, \ldots, A_n\}^+$ then the FD: $A_1A_2\ldots A_n \rightarrow B$ **does** follow from **S**.
- If $B \notin \{A_1, A_2, \ldots, A_n\}^+$ then the FD: $A_1A_2\ldots A_n \rightarrow B$ **doesn't** follow from **S**.

Example: Use of the closure of attributes

[Q] Consider the previous example. Test whether $AB \rightarrow D$ follows from the set of the dependencies.

[A] Yes since $D \in \{A,B\}^+ = \{A,B,C,D,E\}$.

[Q] Consider the FD: $D \rightarrow A$.

[A] No since $A \notin \{D\}^+ = \{D,E\}$. We say, $D \rightarrow A$ does not follow from the given set of dependencies.

X= {D] Use D→ E

X= {D, E} we have reached the closure.

The Transitive Rule

- If $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$ and $B_1B_2 \ldots B_m \rightarrow C_1C_2 \ldots C_m$, then $A_1A_2 \ldots A_n \rightarrow C_1C_2 \ldots C_k$
- Prove this rule using the closure of attributes
  1. With $\{A_1A_2 \ldots A_n\}$ and two FDs $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$ and $B_1B_2 \ldots B_m \rightarrow C_1C_2 \ldots C_m$
  2. $\{A_1, A_2, \ldots , A_n\}^+ = \{A_1A_2 \ldots A_{n,}\ B_1B_2 \ldots B_{m,}\ C_1C_2 \ldots C_m\}$
  3. Therefore, $A_1A_2 \ldots A_n \rightarrow C_1C_2 \ldots C_k$ follows from the given FDs.

Example: The Transitive Rule

title	year	length	genre	studioName	studioAddr
Star Wars	1977	124	sciFi	Fox	Hollywood
Eight Below	2005	120	drama	Disney	Buena Vista
Wayne's World	1992	95	comedy	Paramount	Hollywood

title year → studioName
studioName → studioAddr

Then, we can deduce a new FD based on the transitive rule.

title year → studioAddr

Closures and Keys

- $\{A_1, A_2, \ldots, A_n\}^+$ is the set of **all** attributes of a relation if and only if $\{A_1, A_2, \ldots, A_n\}$ is a **superkey** for the relation.
- Only then does $A_1, A_2, \ldots, A_n$ functionally determines all the attributes.
- We can test if $A_1, A_2, \ldots, A_n$ is a key for a relation by checking:
  - **first** that $\{A_1, A_2, \ldots, A_n\}^+$ contains all attributes,
  - **and** if any of attribute is removed from $\{A_1, A_2, \ldots, A_n\}$, then $\{\text{reduced set of attributes}\}^+$ will not contain all the attributes (minimal key).

Design of Relational Database Schemas

- The principal problem that we encounter is redundancy, where a fact is repeated in more than one tuple.

Example: relation with redundancy

Movie1 Relation

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

Anomalies

- **Redundancy**.
  - capturing info multiple times unnecessarily. E.g. length and genre.
- **Update anomalies**.
  - forget to update in a tuple
  - E.g. we could change the length of *Star Wars* to 125, in the first tuple, and forget to do the same in the second and third tuple.
- **Deletion anomalies**.
  - deleting a tuple causes a loss of other information as a side effect
  - E.g. if we delete Vivien Leigh. we will lose all the information about Gone With the Wind.

Decomposing Relations - Example

Movie2 Relation

title	year	length	genre	studioNam
Star Wars	1977	124	sciFi	Fox
Gone With the Wind	1939	231	drama	Disney
Wayne’s World	1992	95	comedy	Paramount

title	year	starName
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone With the Wind	1939	Vivien Leigh
Wayne’s World	1992	Dana Carvey
Wayne’s World	1992	Mike Meyers

Movie3 Relation

Decomposing Relations - Example

- No true redundancy!
- The update anomaly disappeared. If we change the length of a movie, it is done only once.
- The deletion anomaly disappeared. If we delete all the stars from $Movie_2$ we still will have the other info for a movie.

Boyce-Codd Normal Form (BCNF)

- The goal of decomposition is to replace a relation by several that do not exhibit anomalies.
- There is a simple condition called Boyce-Codd Normal Form under which the anomalies can be guaranteed not to exist.
- A relation is in BCNF if and only if: whenever there is a nontrivial dependency $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$ for R, it must be the case that $\{A_1, A_2, \ldots, A_n\}$ is a superkey for R.
- That is, the left side of every nontrivial functional dependency must contain a key.

Example: BCNF

Relation Movie1 is not in BCNF.

- {title, year, starName} is a key of the relation.
- Consider the FD: title year$\rightarrow$length filmType studioName
- The left side of the above dependency is not a superkey. In particular we know that the title and the year does not functionally determine starName.

Movie2 is in BCNF.

- The only key is {title, year} and
- title year $\rightarrow$ length filmType studioName holds in the relation

Example: BCNF

A relation with two attributes is always in BCNF.

(a) If there is no nonTrivial FDs, then it is in BCNF

(b) A→ B, but not B→A: A is the only key and the left side of non-trivial FD A→ B contains A.

(c)B→ A, but not A→ B: Symmetric to (b)

(d) A→ B and B→ both A and B are keys. A→ B contains a key (A) and B→ A contains a key(B) in their left sides, respectively.

Decomposition into BCNF

- Decomposition Strategy
  - Find a non-trivial FD $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$ that violates BCNF, i.e. $A_1A_2 \ldots A_n$ is not a superkey.
  - Decompose the relation schema into two overlapping relation schemas:
  - One is all the attributes involved in the violating FD and
  - The other is the left side of the FD and all the other attributes not involved in the FD.
  - By repeatedly, choosing suitable decompositions, we can break any relation schema into a collection of smaller schemas in BCNF.
- The original relation should be able to be reconstructed from the decomposed relations.

Projecting Functional Dependencies

- It will be used to find FDs for the decomposed relations so that we can eventually check that the decomposed relations are in BCNF.
- Suppose a relation R with a set of FD's S and R1 is a projection of R.
  - What FDs hold for R1 ?
  - The algorithm will find all FDs that follow from S and involve only attributes of R1

Algorithm 3.12: Projecting Functional Dependencies

Input: R1 and a set of FD's on R
Output: a set of FD's T that hold in R1
Method:

- Consider each subset X of attributes of R1.
- Compute X^+ using the FD on R.
- At the end throw out the attributes of R, which aren't in R1.
- Then, add to T all nontrivial FD's $X \rightarrow A$ such that A is in X^+ and A is an attribute of R1
- Construct a minimal basis of T.

Example: Projecting Functional Dependencies

- Consider $R(A, B, C, D, E)$ decomposed into $R1(A, C, D)$ and another relation. Let FDs of R be: $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$.
- $\{A\}^+ = \{A,B,C,D\}$ $T = \{A \rightarrow C, A \rightarrow D\}$
- $\{C\}^+ = \{C,D\}$ $T = \{A \rightarrow C, A \rightarrow D, C \rightarrow D\}$
- $\{D\}^+ = \{D\}$ $T = \{A \rightarrow C, A \rightarrow D, C \rightarrow D\}$

Since $\{A\}^+$ includes all attributes, we don't need to consider any superset of $\{A\}$.

- $\{C,D\}^+ = \{C,D\}$ $CD \rightarrow C$ or $CD \rightarrow D$ are trivial. Therefore, T won't be changed.
- Based on the transitive rule, $A \rightarrow D$ follows from $A \rightarrow C$ and $C \rightarrow D$.
- $T = \{A \rightarrow C, C \rightarrow D\}$ which is a minimal basis.

Some simplifications

- Don't need to compute the closure of the empty set or of the set of all attributes because they never yield a non-trivial FD.
- If we find X^+ = all attributes, don't bother computing the closure of any supersets of X.

Algorithm 3.20: BCNF Decomposition Algorithm

Input: A Relation R with a set of functional dependencies S
Output: Decomposed relations in BCNF

The following steps can be applied recursively to any relation R and a set of FD's S.

- Check if R is in BCNF. If so, return R as it is.
- If there are BCNF violations, let one be $X \rightarrow Y$.
- Use Algorithm 3.7 to compute X^+. The relation will be decomposed into R1 = X+ and R2 = X and the attributes that are not in X^+.
- Use Algorithm 3.12 to project FD's for R1 and R2. Let these be S1 and S2, respectively.
- Recursively decompose R1 and R2 using this algorithm.

Return the union of the results of these decompositions.

Example: BCNF (continued)

- Consider a schema:
  `(title, year, studioName, president, presAddr)`
  and functional dependencies:
  `title year → studioName`
  `studioName → president`
  `president → presAddr`
- To find BCNF violating FDs, you need to find keys of this relation. Compute {title, year}+, {studioName}+, {president}+ and see if you get all the attributes of the relation. If not, you got a BCNF violation, and need to break relation. You will find that {title, year} is the only key.
- Last two violate BCNF.

Example: BCNF (continued)

- Decomposition can start with any of these violating FDs. Let's starting with studioName → president
- Add to the right-hand side any other attributes in the closure of studioName (optional but often reduces the amount of work)

1. X={studioName} studioName→president
2. X={studioName, president} president→presAddr
3. X={studioName}$^+$={studioName, president, presAddr}

Example: BCNF (continued)

The choice of FD is now studioName → president presAddr
Therefore, the original relation is decomposed into

Movies1(title, year, studioName)
Movies2(studioName, president, presAddr):

[Q1] Is Movies1 in BCNF ?
[A] Yes

1. Using Algorithm 3.12, find a minimal basis of FDs that hold in Movies1.
2. You will find that Movies1 has a basis title year → studioName.
3. See if {title, year} is a key by finding its closure and see if the closure includes all attributes of Movie1.
   $\{title, year\}^+ = \{title, year, studioName\}$

Example: BCNF (continued)

Movies2(studioName, president, presAddr)

[Q2] Is Movies2 in BCNF ?

[A] No

1. Using Algorithm 3.12, find a minimal basis of FDs that hold in Movies2.

2. You will find that Movies2 has bases

studioName→ president & president → presAddr

3. See if {studioName} and {president} are keys.

{studioName}$^+$ = {studioName, present, presAddr}

{president}$^+$ = {president, presAddr} and thus it is not a key.
We conclude president→ presAddr is a BCNF voilation.

Example: BCNF (continued)

- We have to decompose Movie2 into

Movie2-1 (president, presidentAddr)

Movie2-2 (president, studioName)

Both of them are relations with 2 attributes and thus in BCNF.

- Final result

Movies1(title, year, studioName)

Movie2-1 (president, presidentAddr)

Movie2-2 (president, studioName)