

Principles of Database Systems With Internet and Java Applications

Today's Topic

Chapter 5: Improving the Quality of Database Designs

Instructor's name and information goes here
Please see the notes pages for more information.

1

Functional Dependencies and Normalization

- Begin by discussing good and bad relation schemas
- Informal measures of the quality of relation schema design
 - Semantics of the attributes
 - Reducing the redundant values in tuples
 - Reducing the null values in tuples
 - Disallowing spurious tuples
- Define Normal Forms as formal measures of the quality of schemas
 - restrictions on the form of relation schemas

2



Semantics of the Relation Attributes

- How to interpret the attribute values stored in a tuple?
- Guideline 1: Design a schema so that it is easy to explain its meaning.
- Keep attributes from different entities and relationships distinct.
- Example of mixing:
 - OwnerCar: (Oname, DLNum, CarId, Make, Manuf)
 - Oname is attribute of owner, Make is attribute of car!

3



Redundant Information in Tuples

- Previous example of OwnerCar
 - OwnerCar: (Oname, DLNum, CarId, Make, Manuf)
- Consider a table of OwnerCar :
 - (Joe, 123456789, 106, Plymouth, Chrysler)
 - (Moe, 223456789, 107, Plymouth, Chrysler)
- The Manuf attribute is redundant!
- This leads to difficulty in updates also called **Update Anomalies**
 - E.g. changing the Manuf for Joe requires also changing for Moe.

4



Update Anomalies

- Insertion Anomalies
 - When inserting a new owner, we must correctly insert the Manuf field, or will create inconsistencies
 - Cannot create a car without an owner
 - Cannot create a make without a car and an owner
- Deletion Anomalies
 - Deletion of owner of a car also deletes make and manufacturer of car
 - Deletion of owner of the last Plymouth deletes relationship between Plymouth and Chrysler
- Modification Anomalies
 - Changing the make of a car requires consistency check
 - Cannot change so that a Plymouth is made by Ford
- Guideline 2: no insertion, deletion, or modification anomalies allowed!

5



Null Values in Tuples

- May have many attributes (fat relation) which do not apply to many tuples
 - Hence, many null values in many tuples
 - Takes lots of space
 - Not sure how to treat these in Sum, Count
- Nulls can have many interpretations
 - Attribute does not apply
 - Attribute value is unknown
 - Value is known but absent
- Guideline 3: Avoid placing attributes whose values may be null in a base relation.

6



First Normal Form (1NF)

- 3 normal forms proposed by Codd in 1972
- All attribute values are **atomic** (or **indivisible**).
- This rule is now part of the definition of relation.
- Hence, the translation from ERD to relational schema requires that multi-valued attributes be transformed into tables. See Step 6, p. 174.

7



Normal Forms based on Primary Keys

- Normalization includes testing and modifying a schema until it satisfies a set of rules
- Hope to ensure that update anomalies do not occur.
- Unsatisfactory schemes are decomposed by breaking up attributes in smaller relations.
- For each rule, if a particular relation violates the rule, that relation must be broken into smaller relations

8



Some definitions

- **superkey**: a set of attributes of a relation whose values are unique within the relation.
- **key**, a superkey in which removal of any attribute makes it not a superkey. If there is more than one key, they are called **candidate keys**.
- **primary key**, arbitrarily designated candidate key, all other candidate keys are **secondary keys**.
- **prime attribute**, one which is a member of any key.
- **nonprime attribute**, one which is not prime.

9



Definition of Functional Dependency

- A functional dependency is a constraint between 2 sets of attributes from the database
 - For each value of the first set there is a unique value of the second set
- $X \twoheadrightarrow Y$ restricts the tuples that can be instances of R
- if t_1 and t_2 are instances of R
 - $t_1(X) = t_2(X)$ then $t_1(Y) = t_2(Y)$
- For example,
 - $\{DLNum\} \twoheadrightarrow \{Oname\}$
 - $\{CarId\} \twoheadrightarrow \{Make, Manuf\}$
 - $\{Make\} \twoheadrightarrow \{Manuf\}$
- Candidate keys are left hand sides of functional dependencies

10

Second Normal Form (2NF)

- $X \twoheadrightarrow Y$ is a **full functional dependency** if the removal of any attribute A from X removes the dependency
 - not $X - \{A\} \twoheadrightarrow Y$
- $X \twoheadrightarrow Y$ is a **partial dependency** if some attribute A may be removed without removing the dependency
 - $X - \{A\} \twoheadrightarrow Y$
- A relation schema R is in **2NF** if every nonprime attribute is *fully functionally dependent* on the primary key of R

11

Consider the Car Registration Document

- Fig. 5.9 Sample car registration form

State of Florida Department of Motor Vehicles
Vehicle Registration Certificate

DECAL NUMBER		SEX	BIRTHDATE MO. DAY YR.	EXPIRES MO. DAY YR.	TAG ISSUED	TAG NUMBER	
10583256	YR. 9					M	3 11 47
						COLOR	CPR
TITLE NUMBER	VEHICLE IDENTIFICATION NO.	YR. MAKE	WT/LENGTH	CLASS	MAKE	TYPE	
38543630	KLA4KA4667MCO19042	90	3179	02	ACUR	4D	
OWNER NAME AND ADDRESS				1ST OWNER DL. NO.	2ND OWNER DL. NO.		
GREGORY, MATILDA A OR ALAN M 2006 W BRANCOCH CIR TALLAHASSEE FL 32301				G26438156171	G26351353447		
				INSURANCE PIP	LIABILITY	CREDITS	REFUNDS
				C		0.00	0.00
				DATE ISSUED MO. DAY YR.		12 10 99	
TAG MONEY	MOS.	TAX \$	B.T. MOS.	BACK TAX \$	SVC. CHG. \$	OTHER \$	TAG TOTAL \$
		1.60		0.00	2.50	0.00	4.10
TITLE FEE \$	LATE FEE \$	LIEN \$	SVC. CHG. \$	TITLE TOTAL \$	SALES TAX \$	GRAND TOTAL \$	
25.00	0.00	0.00	4.25	29.25	0.00	33.25	

12



Example of Car Registration Schema

- This is a different car registration example from Fig. 5.9
- Relation owner
 - DLNum, Name, Address, City, State, Zip
- Relation Car
 - CarId, DLNum, Make, Model, Manuf, Year, Color, Owner, PurchDate, TagNum, RegisDate
- R is set of all attributes of schema
- F is set of all functional dependencies
 - {DLNum} --> {Name, Address, City, State, Zip}
 - {CarId} --> {Make, Model, Manuf, Year, Color}
 - {TagNum} --> {RegisDate}
 - {CarId, DLNum} --> {PurchDate, TagNum, ...}
 - and more!

13



Putting the CarReg Schema into 2NF

- Consider the **Owner** relation schema
 - {DLNum} is the primary key
 - Hence **Owner** is in **2NF**
- Consider the **Car** relation schema
 - {CarId, DLNum} is primary key (multiple owners)
 - {CarId} --> {Make, Model, ...}
 - Hence **Car** is not **2NF**
- Create new relations
 - **CarOwner** = {CarId, Owner, PurchDate, TagNum, RegisDate}
 - **Car** = {CarId, Make, Model, Manuf, Year, Color}
- Is it **2NF**?

14



Rules for Functional Dependencies

- Given a particular set of functional dependencies, we can find others using inference rules
 - Splitting/combining rules
 - $A \rightarrow B_1 B_2 \Leftrightarrow A \rightarrow B_1 \text{ and } A \rightarrow B_2$
 - Trivial rules
 - $A B \rightarrow B$, for all A, B
 - Transitive rule
 - $A \rightarrow B \text{ and } B \rightarrow C \Rightarrow A B \rightarrow C$
- We are interested in the closure of the set of functional dependencies under these (and other) rules

15



Inference Rules for Functional Dependency

- There are **semantically obvious** functional dependencies, usually specified by schema designer
- Other functional dependencies can be **inferred** from those
- Inference rules
 - Reflexive, X includes Y , $X \twoheadrightarrow Y$
 - Augmentation, $X \twoheadrightarrow Y$ then $XZ \twoheadrightarrow YZ$
 - Transitive, $X \twoheadrightarrow Y \twoheadrightarrow Z$ then $X \twoheadrightarrow Z$
 - Decomposition, $X \twoheadrightarrow YZ$ then $X \twoheadrightarrow Y$
 - Union, $X \twoheadrightarrow Y$ and $X \twoheadrightarrow Z$ then $X \twoheadrightarrow YZ$
 - Pseudotransitive, $X \twoheadrightarrow Y$ and $WY \twoheadrightarrow Z$ then $WX \twoheadrightarrow Z$

16



Definition of Key

- A set of one or more attributes $\{A_1, \dots, A_k\}$ is a *key* for a relation R
 - Those attributes functionally determine all other attributes of R
 - no 2 distinct tuples can agree on the key
 - no proper subset of $\{A_1, \dots, A_k\}$ is a key of R
 - a key must be *minimal*
- There can be more than one key in a relation
 - Department (DeptName, DeptNo,...)
 - since both are unique, both are keys
- A *superkey* (superset of a key) is a set of attributes that functionally determine all other attributes of the relation.

17



Third Normal Form (3NF)

- Based on **transitive dependency**, or non-key dependency
- A functional dependency $X \twoheadrightarrow Y$ is a transitive dependency if there is a set Z which is not a subset of any key, and for which $X \twoheadrightarrow Z$ and $Z \twoheadrightarrow Y$
- A relation schema is in 3NF if there is no nonprime attribute which is functionally dependent on a non-key set of attributes.
- Example of $\{\text{make}\} \twoheadrightarrow \{\text{manuf}\}$ violates 3NF since make is not a key.

18

Transforming Car into 3NF

- **Car** = {CarId, Make, Model, Manuf, Year, Color}
- {CarId}-->{Make, Model, Manuf, Year, Color}
{Make} --> {Manuf}
Not 3NF
- Car = {CarId, Make, Model, Year, Color}
MakeManuf = {Make, Manuf}
- What about {Model}-->{Make}?

19

Boyce Codd Normal Form (BCNF)

- A relation R is BCNF iff for each non-trivial dependency {A1,...Ak} -> B for R,
 - A1...Ak is a superkey
- Alternatively, collect all similar violations
 - if A1...Ak -> B1...Bn then {A1,...Ak} is a superkey
- A 3NF relation is not BCNF only if there is
 - X -> A such that
 - X is not a superkey and
 - A is a prime attribute
- Any 2-attribute relation is BCNF: e.g. R(a,b)
 - either a->b but not b->a, {a} is key but not {b}
 - a->b and b->a, both {a} and {b} are keys
 - neither a->b nor b->a, {a,b} is key

20



Why BCNF?

- BCNF schemas do not exhibit anomalies
 - only redundancy is foreign key
 - each non-key attribute appears only once
 - only update and delete problems are
 - update of key attribute must be propagated to foreign keys
 - deletion of tuple must be propagated to foreign keys, either null or delete
- All functional dependencies are key dependencies
 - Functional dependency constraints have been turned into key constraints
 - Database system can enforce key constraints

21



Conversion of DB Schema into BCNF

- Consider a single relation schema
 - Identify a BCNF violation
 - Decompose the relation to remove the violation
 - Repeat until no violations occur
- Repeat for every relation in the DB schema, including the new relations created by decomposition

22



Decomposition into BCNF

- Suppose R has a BCNF violation
 - $A_1 \dots A_n \rightarrow B_1 \dots B_m$ and $\{A_1, \dots, A_n\}$ is not superkey
 - Bs include all attributes that are dependent
 - let $\{C_1, \dots, C_k\}$ be all other attributes (not As or Bs)
- Create 2 new relations
 - $R_1(A_1, \dots, A_n, B_1, \dots, B_m)$ and
 $R_2(A_1, \dots, A_n, C_1, \dots, C_k)$
 - keys must be determined by considering resulting functional dependencies
- Consider other examples in class