Normalization

Dr. Rafal A. Angryk

Outline

- Normal Forms Based on Primary Keys
  - Normalization of Relations
  - Practical Use of Normal Forms
  - Definitions of Keys and Attributes Participating in Keys
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
- General Normal Form Definitions (For Multiple Keys)
- Intro to BCNF (Boyce-Codd Normal Form)

Normalization of Relations(1)

- Normalization: The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations
- Normal form: Condition using keys and Functional Dependencies of a relation to certify whether a relation schema is in a particular normal form

Normalization of Relations(2)

- 2NF, 3NF, BCNF are based on keys and Functional Dependencies of a relation schema
- 4NF based on keys and multi-valued dependencies
- 5NF based on keys and join dependencies (Chapter 11)
- Additional properties may be needed to ensure a good relational design (lossless join, dependency preservation; Chapter 11)

Practical Use of Normal Forms

- Normalization is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The practical utility of these normal forms becomes questionable when the constraints on which they are based are hard to understand or to detect
- The database designers *need not* normalize to the highest possible normal form. (usually we go up to 3NF, BCNF or 4NF)
- **Denormalization**: the process of storing the join of higher normal form relations as a base relation—which is in a lower normal form

Definitions of Keys and Attributes Participating in Keys (1)

- A **superkey** of a relation schema $R = \{A_1, A_2, ..., A_n\}$ is a set of attributes S (i.e. ***subset-of R***) with the property that no two tuples t_1 and t_2 in any legal relation state r of R will have $t_1[S] = t_2[S]$
  Superkey = any set of attributes that make a tuple unique
- A **key** K is a superkey with the *additional property* that removal of any attribute from K will cause K not to be a superkey any more.
  Key = a minimum superkey

Definitions of Keys and Attributes Participating in Keys (2)

- **Candidate key** - if a relation has more than one key, each is a candidate key
- **Primary key** - an arbitrary candidate key
- **Secondary key** - a candidate key that is not the PK

Definitions of Keys and Attributes Participating in Keys (3)

- If a relation schema has more than one key, each is called a candidate key. One of the candidate keys is *arbitrarily* designated to be the primary key, and the others are called ***secondary keys***.
- A prime attribute must be a *member of some candidate key*
- A **nonprime** (or **nonkey**) **attribute** is not a prime attribute - that is, it is *not a member of any candidate key*.

First Normal Form (1)

- Disallows composite attributes, multivalued attributes, and nested relations; attributes whose values *for an individual tuple* are non-atomic
- 1NF is considered now to be part of the formal definition of a relation

First Normal Form (2)

- 1NF states that:
  - the domains of attributes must include ONLY atomic values
  - the value of any attribute in a tuple must be a single value from the domain of that attribute
  - all attributes are functionally dependent on the PK
- Look at this schema for departments:

DNAME	DNUMBER	DMGRSSN	DLOCATIONS

Normalizing the relation into 1NF

DNAME	DNUMBER	DMGRSSN	DLOCATIONS

- The problem with this table is that department dlocation is **multivalued**.
- What are the possible solutions?
  - Have the primary key be composite {dnumber, dlocation}
    - Problems with this?
  - If you know the maximum possible number of locations (e.g. 3), make separate attributes for all of them
    - Problem with this?
  - Make a new table with dname and dlocation
    - This is the preferred solution

Example: Normalization into 1NF

Figure 10.8 Normalization into 1NF. (a) Relation schema that is not in 1NF. (b) Example relation instance. (c) 1NF relation with redundancy.

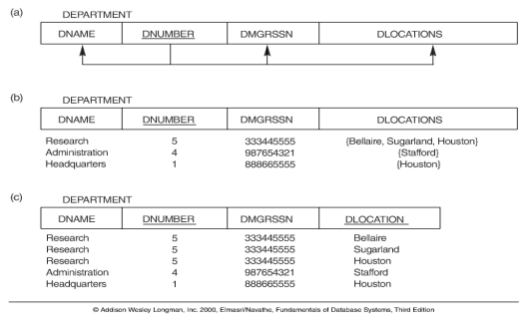
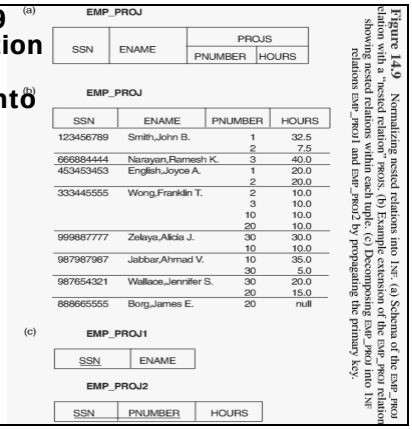


Figure 10.9 Normalization of nested relations into 1NF



(a) EMP_PROJ

SSN	ENAME	PROJS	
		PNUMBER	HOURS

(b) EMP_PROJ

SSN	ENAME	PNUMBER	HOURS
123456789	Smith,John B.	1	32.5
		2	7.5
666884444	Narayan,Ramesh K.	3	40.0
453453453	English,Joyce A.	1	20.0
		2	20.0
333445555	Wong,Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya,Alicia J.	30	30.0
		10	10.0
987987987	Jabbar,Ahmad V.	10	35.0
		30	5.0
987654321	Wallace,Jennifer S.	30	20.0
		20	15.0
888665555	Borg,James E.	20	null

(c) EMP_PROJ1

SSN	ENAME

EMP_PROJ2

SSN	PNUMBER	HOURS

Figure 14.9 Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a "nested relation" PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposing EMP_PROJ into 1NF relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

Second Normal Form (1)

- Uses the concepts of Functional Dependencies and Primary Key

  Reminder:

- Full functional dependency - a FD Y -> Z where removal of any attribute from Y means the FD does not hold any more

  Examples:

  - {SSN, PNUMBER} -> HOURS is a **full FD** since neither SSN -> HOURS nor PNUMBER -> HOURS hold

  - {SSN, PNUMBER} -> ENAME is *not* a full FD (it is called a ***partial FD***) since SSN -> ENAME also holds

Second Normal Form (2)

- A relation schema R is in second normal form (2NF) if every non-prime attribute A in R is fully functionally dependent on the primary key

- R can be decomposed into 2NF relations via the process of 2NF normalization

Second Normal Form (3)

- A relation is in 2NF if: every nonkey attribute is fully functionally dependent on the primary key.
- If a relation is not in 2NF, it can be normalized (decomposed) into a number of 2NF relations
- **Problem**:

  Normalize the EMP_PROJ table

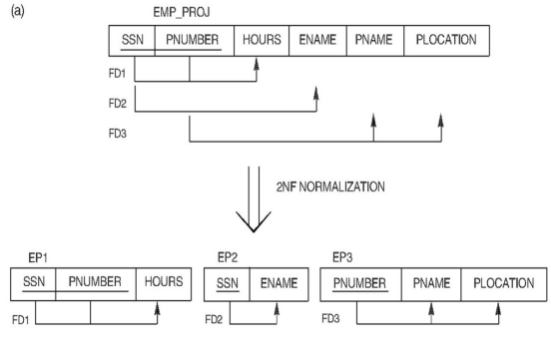
EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION

Second Normal Form (4)

- Procedure:
  - Figure out the dependencies, noting the determinates
    - A determinate is any attribute(s) on which some other attribute(s) are dependent
  - Put each determinate in a table by itself, and
  - Include in each table the attributes that are dependent on that determinate

Second Normal Form (5)



Third Normal Form (1)

- Look at this relation

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN

- Is it in 1NF? In 2NF?
- Is there still a problem? What is the problem?

The problem with the EMP_DEPT table

- DNAME depends on DNUM, not the primary key.
- This is called a transitive dependency.
- A functional dependency X-> Y is a transitive dependency if
  - there is a set of attributes Z of the relation and both X -> Z and Z ->Y hold
- E.g. SSN -> DNUM, DNUM->DNAME

Third Normal Form (2)

- A relation is in 3NF if every nonkey attribute is:
  - Fully functionally dependent on the primary key (i.e. in 2NF).
  - Nontransitively dependent on the primary key.

Example: We can normalize EMP_DEPT by decomposing it into two 3NF relations.

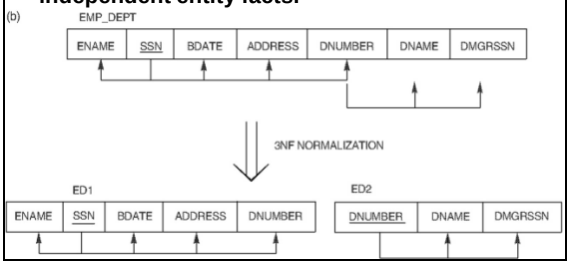
Intuitively, we see that the two results represent independent entity facts.

In such case a natural join will recover the original relation.

Third Normal Form (2)

We can normalize EMP_DEPT by decomposing it into two 3NF relations.

Intuitively, we see that the two results represent independent entity facts.



Summary of Normal forms

- A relation is in 1NF the domains of attributes must include only atomic values.
- A relation is in 2NF if every nonkey attribute is fully functionally dependent on a candidate key.
- A relation is in 3NF if every nonkey attribute is in 2NF and nontransitively dependent on the primary key.

p. 321 in your textbook

Perform Normalization of Universal Table

- AKA Unnormalized Table

Student#	Advisor	Adv-Room	Classes
1022	Jones	412	101-07, 143-01, 159-02
4123	Smith	216	201-01, 211-02, 214-01

Go ahead and have some PRACTICE!

1NF: No Non-atomic Values

- Field Class (i.e. Class1, Class2, and Class3) in the above records are indications of design trouble. Since one student may have several classes, these classes should be listed as atomic values.
- Another way to look at this problem is with a one-to-many relationship. Do not put the one side and the many side in the same table. Instead, create another table in first normal form by eliminating the repeating group (Class#)

1NF: No Non-atomic Values

Student#	Advisor	Adv-Room	Class#
1022	Jones	412	101-07
1022	Jones	412	143-01
1022	Jones	412	159-02
4123	Smith	216	201-01
4123	Smith	216	211-02
4123	Smith	216	214-01

- This is 1NF with redundancy (notice the PK)
- What are our other options?

2NF: Eliminate Redundant Information by analysis of FDs

- Note the multiple Class# values for each Student# value in the above table. Class# is not fully functionally dependent on Student#, so this relationship is not in second normal form.

2NF: Eliminate Redundant Information by analysis of FDs

Students

Student#	Advisor	Adv-Room
1022	Jones	412
4123	Smith	216

Registration

Student#	Class#
1022	101-07
1022	143-01
1022	159-02
4123	201-01
4123	211-02
4123	214-01

3NF: Eliminate Data which is only TRANSITIVELY Dependent On Key

- Adv-Room (the advisor's office number) is functionally dependent on the Advisor attribute. Advisor attribute is functionally dependent on the Student# attribute.
- We have TRANSITIVE DEPENDENCY here!!!
- The solution is to move that attribute from the Students table to the Faculty table.

3NF: Eliminate Data which is only TRANSITIVELY Dependent On Key

Students

Student#	Advisor
1022	Jones
4123	Smith

Faculty

Name	Room
Jones	412
Smith	216

Registration

Student#	Class#
1022	101-07
1022	143-01
1022	159-02
4123	201-01
4123	211-02
4123	214-01

General Normal Form Definitions (For Multiple Keys) (1)

- **The above definitions consider the primary key only**
- **The following more general definitions take into account relations with multiple candidate keys**
- **A relation schema R is in the second normal form (2NF) if every non-prime attribute A in R is fully functionally dependent on *every key* of R**

General Normal Form Definitions (2)

- **Superkey of relation schema R - a set of attributes S of R that contains a key of R**
- **A relation schema R is in third normal form (3NF) if whenever a FD X -> A holds in R, then either:**
  - **(a) X is a superkey of R, or**
  - **(b) A is a prime attribute of R**

NOTE:

Boyce-Codd normal form disallows condition (b) above

BCNF (Boyce-Codd Normal Form)

- **A relation schema R is in Boyce-Codd Normal Form (BCNF) if whenever an FD X -> A holds in R, then X is a superkey of R**
- **Each normal form is strictly stronger than the previous one**
  - **Every 2NF relation is in 1NF**
  - **Every 3NF relation is in 2NF**
  - **Every BCNF relation is in 3NF**
- **There exist relations that are in 3NF but not in BCNF**
- **BCNF is a simpler, yet stricter form of 3NF.**
- **In general, our goal is to have each relation in BCNF (or in 3NF)**

Remarks

- **In general, it is best to have a relational schema in BCNF**
- **If that is not possible, 3NF will do**
- **2NF and 1NF are not considered good relation schema designs.**
  - **They allow too much data redundancy which leads to update anomalies**

Summary of Normal forms

- A relation is in 1NF the domains of attributes must include only atomic values
- A relation is in 2NF if every nonkey attribute is fully functionally dependent on a candidate key.
- A relation is in 3NF if every nonkey attribute is in 2NF and nontransitively dependent on a candidate key
- A relation is in BCNF if and only if every determinate is a candidate key

Two approaches to DB design – Intro to Ch. 11

Top-Down

- Top-down – **designing a conceptual schema in a high-level data model like an ER**
  - **The E-R diagram must then be mapped to a relational model**
  - **Then each relation is analyzed for FD and PK and normalized if necessary**

Bottom-Up

- Bottom-up – **the db schema starts out as a universal relation with all the attributes for the entire DB**
  - **The designer then specifies al the dependencies**
  - **Then a normalization algorithm is applied to make the relational schema by decomposing the universal relation**

Decomposition

- **When a relation is broken down into other relations, we must be sure not to lose any information.**
- **Notice that the decomposition operator is a** projection
- **The decomposition must be reversible**
- **Reversible means that the original relation is equal to the** join **of its projection**

Properties of decompositions

- Attribute preservation
  - **The union of all the decomposed tables must equal the universal relation**
- Dependency preservation
  - **We need to preserve the dependencies because each dependency represents a constraint on the DB**
- Lossless (nonadditive) joins
  - **This assures that no spurious tuples are generated when a natural join is applied to the relations in the decomposition**
  - **We looked at one really bad decomposition where many spurious tuples were generated**

Normalization algorithms

- **Algorithms have been written that assure that all the preceding properties are adhered to**
- **There is a branch of DB theory that studies and proves that these algorithms result in a relational schema where none of the semantics is lost**
- One of the problems **with these algorithms is that ALL of the FD must be specified**
- Another problem **is that these algorithms are not deterministic in general**
- In conclusion: **even though these algorithms make designing a DB schema rote, they cannot eliminate the art required by design**


Questions?