Database Management Systems

Part III: Database Design

Lecture 10

Normalization: 1NF, 2NF, 3NF, BCNF

Contents

- Introduction
- Normal Forms
- Nonloss Decomposition
- Functional Dependencies
- First, Second, and Third Normal Forms
- Boyce/Codd Normal Form (BCNF)
- Dependency Preservation

Introduction

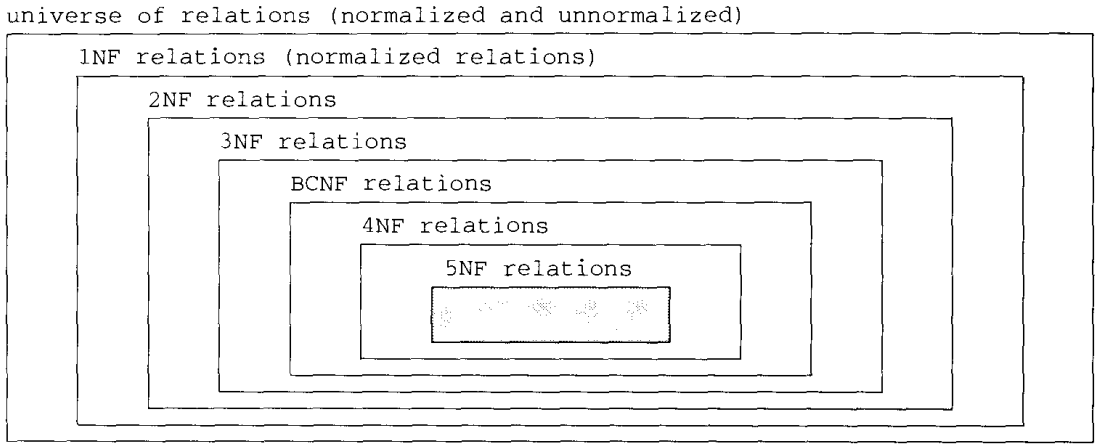
- Normalization of data is a process of analyzing the given relation based on their functional dependencies and primary keys to minimize redundancy and the insertion, deletion, and update anomalies.
- Normalization brings the database to a consistent state.
- It can be considered as a filtering or purification process to make the design have successively better quality.
- This is accomplished by designing the relations that are in an appropriate normal form.

Normal Forms

- The process of further normalization is built around the concept of normal forms.
- The normal form of a relation refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized.
- All these normal forms are based on a single analytical tool which is the functional dependencies among the attributes of a relation.

Normal Forms (Cont.)

- Numerous normal forms have been defined as shown in figure.
- Initially, Codd proposed three normal forms, which he called First, Second, and Third Normal Form.
- A stronger definition of 3NF called Boyce-Codd Normal Form (BCNF) was proposed later by Boyce and Codd.



Normalization Procedure

- The normalization procedure involves breaking down or decomposing a given relation into other relations.
- Decomposition operator in the normalization procedure is projection and recomposition operator is join.
- The process of normalization through decomposition must also confirm the existence of two additional properties: Nonloss Decomposition and Dependency Preservation.

Nonloss Decomposition

- Nonloss decomposition guarantees that the spurious tuple generation problem does not occur with respect to the relation schemas created after decomposition.
- Nonloss decomposition also guarantees that no information is lost in the process.
- Figure shows a sample tabulation of relation S and there are two possible decompositions corresponding to that tabulation.

S

S#	STATUS	CITY
S3	30	Paris
S5	30	Athens

Nonloss Decomposition (Cont.)

Case (a)

SST

S#	STATUS
S3	30
S5	30

SC

S#	CITY
S3	Paris
S5	Athens

- In case (a), the two relations SST and SC still tell us that supplier S3 has status 30 and city Paris, and supplier S5 has status 30 and city Athens.
- If we join relations SST and SC back together again, we get back the original relation S and no information is lost.
- In other words, this first decomposition is indeed nonloss.

Nonloss Decomposition (Cont.)

Case (b)

SST

S#	STATUS
S3	30
S5	30

STC

STATUS	CITY
30	Paris
30	Athens

- In case (b), we can still tell that both suppliers have status 30, but we cannot tell which supplier has which city.
- if we join SST and SC together again, we do not get back the original relation S, and so we have lost information.
- In other words, the second decomposition is not nonloss but lossy.

Functional Dependencies

- There is a formal methodology for evaluating whether a relational schema should be decomposed.
- This methodology is based upon the concepts of keys and functional dependencies.
- Whereas a superkey is a set of attributes that uniquely identifies an entire tuple, a functional dependency allows us to express constraints that uniquely identify the values of certain attributes.

Functional Dependencies (Cont.)

- Functional dependencies have to be satisfied in order to guarantee that joining the decomposed relations back together take back to the original relation.
- For example, observe that relation S satisfies the irreducible set of FDs.

S# → STATUS

S# → CITY

S

S#	STATUS	CITY
S3	30	Paris
S5	30	Athens

Functional Dependencies (Cont.)

Health's Theorem

- Let R{A, B, C} be a relation, where A, B, and C are sets of attributes.
- If R satisfies the FD A → B, then R is equal to the join of its projections on {A,B} and {A,C}.

Functional Dependencies (Cont.)

- Returning to the example, taking A as S3, B as STATUS, and C as CITY, the theorem confirms that relation S can be nonloss-decomposed into its projections on {S#, STATUS} and {S#, CITY}.
- At the same time, relation S cannot be nonloss-decomposed into projections on {S#, STATUS) and {STATUS, CITY}.

First Normal Form

- First normal form (1NF) is defined in the definition of relations (tables) itself.
- 1NF states that all the attributes in a relation must have atomic domains and that the value of any attribute in a tuple must be a single value from the domain of that attribute.
- A domain is atomic if elements of the domain are considered to be indivisible units.
- A relation schema R is in first normal form (1NF) if the domains of all attributes of R are atomic.

First Normal Form (Cont.)

- A set of names is an example of a nonatomic value as they can be divisible such as FIRST NAME and LAST NAME.
- Identification numbers such as CS001, EE1127 can also be divided into smaller units and so, they are nonatomic too.
- If a relation schema had an attribute whose domain consists of identification numbers encoded as above, the schema would not be in first normal form.

First Normal Form (Cont.)

- We rearrange the relation (left table) as below (right table) to convert it to First Normal Form.
- Each attribute must contain only a single value from its pre-defined domain.

Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

Second Normal Form

- Second normal form (2NF) is based on the concept of full functional dependency.
- Functional dependency FD: X → Y is a full functional dependency if any attribute A from X is removed, then the dependency does not hold any more.
- The FD: X→Y is a partial dependency if some attribute A ε X can be removed from X and the dependency still holds.

Second Normal Form (Cont.)

- A relation is in 2NF if and only if it is in 1NF and every non-key attribute is full functional or irreducibly dependent on the primary key.
- Note that a non-key attribute is any attribute that does not participate in the primary key of the relation concerned.

Second Normal Form (Cont.)

- For example, in Student_Project relation, the primary key attributes are Student Id and Project ID and non-key attributes are Student Name and Project Name.
- The relation is not in second normal form because Student Name and Project Name can be identified by Student Id and Project ID independently.

Student_Project

Student ID	Project ID	Student Name	Project Name

Second Normal Form (Cont.)

- We decomposed the original relation (Student_Project) into two relations (Student and Project) as follows.
- Now, there exists full functional dependency and they are in the second normal form.

Student

Student ID	Student Name	Project ID

Project

Project ID	Project Name

Third Normal Form

- Third normal form (3NF) is based on the concept of transitive dependency.
- A relation is in 3NF if and only if it is in 2NF and every non-key attribute is mutually independent each other and irreducibly dependent on the primary key.
- Two or more attributes are mutually independent if none of them is functionally dependent on any combination of others.
- Such independence implies that each such attribute can be updated independently of all the rest.

Third Normal Form (Cont.)

- For example, consider the relation P is in 3NF.

P

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

P# → PNAME

P# → COLOR

P# → WEIGHT

P# → CITY

- The attributes PNAME, COLOR, WEIGHT, and CITY are certainly all independent of one another.
- And, they are all irreducibly dependent on the primary key P#.
- According to the definition, the relation P is in 3NF.

Boyce/Codd Normal Form

- Boyce/Codd Normal Form (BCNF) eliminates all redundancy that can be discovered based on functional dependencies.
- Ideally, relational database design should strive to achieve BCNF or 3NF for every relation schema.
- A relation is in BCNF if and only if every nontrivial, left-irreducible FD has a candidate key as its determinant (left-hand side of an FD).
- Or, less formally, a relation is in BCNF if and only if the only determinants are candidate keys.

Boyce/Codd Normal Form (Cont.)

- For example, consider the suppliers relation S is in BCNF.

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

- The attributes S# and SNAME are both candidate keys (i.e., for all time, every supplier has a unique supplier number and also a unique supplier name).
- Therefore, the relation S is in BCNF.

Dependency Preservation

- The idea that the normalization procedure should decompose relations into projections has come to be known as dependency preservation.
- It is important to preserve the dependencies because each dependency in the relations represents a constraint on the database.
- Dependency preservation ensures that each functional dependency is represented in some individual relation resulting after decomposition.

Summary

- Every relation in (n+1) normal form is automatically in n normal form as well.
- Functional dependencies play the crucial role in the procedure.
- Heath's theorem tells that if a certain FD is satisfied, then a certain decomposition is nonloss.
- Reduction to BCNF is always possible that is any given relation can always be replaced by an equivalent set of relations in BCNF.
- The purpose of such reduction is to avoid redundancy and to avoid certain update, delete and insert anomalies.

Next Lecture

Part III: Database Design

Normalization

- Higher Normal Forms
- Other Normal Forms

Textbook and References

Textbook

- C. J. Date, “An Introduction to Database Systems”, 6th Edition, 1994.

Additional References

- Abraham Silberschatz, Henry F. Korth, S. Sudarshan, “Database System Concepts”, 6th Edition, 2011.
- Ramez Elmasri, Shamkant B. Navathe, “Fundamentals of Database Systems”, 6th Edition, 2010.
- https:// www.tutorialspoint.com