GALGOTIAS UNIVERSITY

School of Computing Science and Engineering

Program: BCA

Course Code:BCAC2102

Course Name: Database Management System

Lecture-17

Topic- Normal Forms

Faculty:-Dr. Satyajee Srivastava

Lecture-16(RECAP)

Topic- Functional Dependencies

Objective :

Functional Dependencies and their properties

Lecture-16

Consider Two table:- 1.

STATE table:

State Abbrev	StateName	Union Order	StateBird	State Population
CT	Connecticut	5	American robin	3,287,116
MI	Michigan	26	robin	9,295,297
SD	South Dakota	40	pheasant	696,004
TN	Tennessee	16	mocking bird	4,877,185
TX	Texas	28	mocking bird	16,986,510

Lecture-16

Consider Two table:- 2.

CITY table:

State Abbrev	CityName	City Population
CT	Hartford	139,739
CT	Madison	14,031
CT	Portland	8,418
MI	Lansing	127,321
SD	Madison	6,257
SD	Pierre	12,906
TN	Nashville	488,374
TX	Austin	465,622
TX	Portland	12,224

Lecture-16

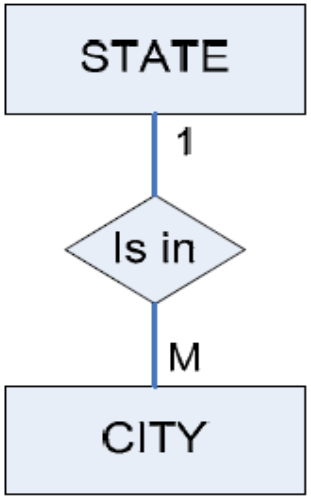
Outline Notation

STATE(StateAbbrev, StateName, UnionOrder,
StateBird, StatePopulation)

CITY(StateAbbrev, CityName, CityPopulation)
StateAbbrev foreign key to STATE

- Underline all parts of each primary key
- Note foreign keys with "*attribute* foreign key to *TABLE*"

Lecture-16



- one-to-many relationships: to determine the direction, always start with "one"
  - "*one* city is in *one* state"
  - "*one* state contains *many* cities"
- the foreign key is always in "the many" – otherwise it could not be atomic (it would have to be a list)

Lecture-16

Functional Dependency

- attribute B is functionally dependent on attribute A if given a value of attribute A, there is only one possible corresponding value of attribute B
  - that is, any two rows with the same value of A must have the same value for B
- attribute A is the determinant of attribute B if attribute B is functionally dependent on attribute A
  - in the STATE relation above, StateAbbrev is a determinant of all other attributes
  - in the STATE relation, the attribute StateName is also a determinant of all other attributes
  - so, StateAbbrev and StateName are both candidate keys for STATE

Lecture-16

- in the CITY relation above, the attributes (StateAbbrev, CityName) together are a determinant of the attribute CityPopulation
- in the CITY relation, the attribute CityName is not a determinant of the attribute CityPopulation because multiple cities in the table may have the same name

Lecture-16

- in the CITY relation above, the attributes (StateAbbrev, CityName) together are a determinant of the attribute CityPopulation
- in the CITY relation, the attribute CityName is not a determinant of the attribute CityPopulation because multiple cities in the table may have the same name

Lecture-16

- Functional dependencies

Functional dependencies are a constraint on the set of legal relations.

Let, $\alpha \subseteq R$ and $\beta \supseteq R$. The functional dependency $\alpha \rightarrow \beta$ holds on R if in any legal relation r(R). For all pairs of tuples t_1 and t_2 in r such that $t_1(\alpha) = t_2(\alpha)$

Functional dependencies provide a means for defining additional constraints on a relational schema. In simple words, a tuple value in one attribute uniquely determines the tuple's value in another attribute.

Example:

WORKER–ID uniquely determines NAME and WORKER–ID uniquely determines SKILL–TYPE, therefore functional dependencies as

FD : WORKER–ID → NAME

FD : WORKER–ID → SKILL–TYPE

The notation "→" is read "functionally determines".

Lecture-16

- Functional dependencies

Thus, in these examples, WORKER-ID functionally determines NAME, WORKER-ID functionally determines SKILL-TYPE.

The attribute on the left hand side of an FD is called a determinant because its value determines the value of the attribute on right-hand side. A relation's key is a determinant, since its value uniquely determines the value of every attribute in a tuple.

(1) Functional dependency of two attributes :

Consider Branch relation

Branch (Branch-name, branch-city, assets) on Branch-schema.

Branch-name → branch-city

Branch-name → assets

Lecture-16

- Functional dependencies

(2) Example for No functional dependencies :

Consider Depositor–schema relation

Depositor (customer–name, Account–number)

Here, customer–name and Account–number together form primary key.

Customer–name and Account–number are foreign keys.

Therefore no functional dependencies.

Lecture-16

- Functional dependencies

(3) Example of a relation that has a functional dependency in which the determinant has two or more attributes.
Consider STUDENT_COURSE_INFO relation
xSTUDENT_COURSE_INFO (Name, Course, Grade, Phone–no., Major, Course–Dept)

Name → Phone
Course →Course–Dept
Name course → Grade

Name course is a candidate key
Name & Course are prime attributes.
Grade is fully functionally dependent on the candidate key.
Phone–no, Course–Dept and major are partially dependent on the candidate key.

Lecture-16

- Functional dependencies

(4) Transitive and Trivial Functional dependencies :

Transitive Functional Dependencies :

It occurs when a non key attribute is functionally dependent on one or more other non key attributes.

A functional dependency $X \rightarrow Y$ in a relation scheme & is a transitive dependency if there is a set of attributes Z that is neither a candidate key nor a subset if any key of R and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold.

Trivial Functional dependencies :

Let R be a relation on the relation schema R, then R satisfies the functional dependency $X \rightarrow Y$ if a given set of values for each of the values of the attribute in X uniquely determines each of the values of the attributes in Y. Y is said to be functionally dependent on X. The functional dependency is denoted as $X \rightarrow Y$, where X is the left hand side or the determinant of the R and Y is the right hand side of the FD.

A functional dependency $X \rightarrow Y$ is said to be trivial if $Y \subseteq X$ or $Y \subseteq X$, $X \rightarrow Y$.

A functional dependency $X \rightarrow Y$ is said to be trivial functional dependency if $Y \subseteq X$.

Lecture-16

- Functional dependencies

  - A functional dependency is **trivial** if it is satisfied by all instances of a relation
    - Example:
      - *ID, name* $\rightarrow$ *ID*
      - *name* $\rightarrow$ *name*
    - In general, $\alpha \rightarrow \beta$ is trivial if $\beta \subseteq \alpha$

Lecture-16

Closure of a Set of Functional Dependencies

- Given a set F of functional dependencies, there are certain other functional dependencies that are logically implied by F.
  - For example: If $A \rightarrow B$ and $B \rightarrow C$, then we can infer that $A \rightarrow C$
- The set of **all** functional dependencies logically implied by F is the **closure** of F.
- We denote the *closure* of F by $\mathbf{F^+}$.
- F^+ is a superset of F.

Lecture-16

The closure of F, denoted by F^+, is the set of all functional dependencies logically implied by F.

The closure of F can be found by using a collection of rules called **Armstrong axioms.**

Reflexivity rule: If A *is a set of attributes and B is subset or equal to A, then* A→B holds.

Augmentation rule: If A→B holds and C is a set of attributes, then CA→CB holds

Transitivity rule: If A→B holds and B→C holds, then A→C holds.

Union rule: If A→B holds and A→C then A→BC holds

Decomposition rule: If A→BC holds, then A→B holds and A→C holds.

Pseudo transitivity rule: If A→B holds and BC→D holds, then AC→D holds.

Lecture-17

Topic- Normal Forms

Objective :

To acquire knowledge about Normal Forms

Lecture-17

Topic- Normal Forms

Objective :

To acquire knowledge about Normal Forms

Lecture-17

Topic- Normal Forms

Objective :

To acquire knowledge about Normal Forms

Lecture-17

Topic- Normal Forms

Objective :

To acquire knowledge about Normal Forms

Lecture-17

Topic- Normal Forms

Objective :

To acquire knowledge about Normal Forms

Lecture-17

Topic- Normal Forms

Objective :

To acquire knowledge about Normal Forms

Lecture-17

Topic- Normal Forms

Objective :

To acquire knowledge about Normal Forms

Lecture-17

Topic- Normal Forms

Objective :

To acquire knowledge about Normal Forms

Lecture-17

Topic- Normal Forms

Objective :

To acquire knowledge about Normal Forms

Lecture-17

Normalization

Normalization: is the process of "fixing" relational schemata so that they avoid three closely related kinds of problems.

Storage redundancy: The same information is repeated many times.

Unnecessary information dependency: Information about some x cannot be represented without having at least corresponding instance of y.

Update anomalies: The way in which data is represented complicates the support of certain kinds of updates.

Lecture-17

Illustration of Problems of an Unnormalized Schema

Firm

SSN	Name	Dept	Bldg
000112222	Alice	3	8
000113333	Bruce	3	8
000114444	Carol	3	8
000115555	David	5	7
000116666	Alice	4	7

$SSN \rightarrow \{Name, Dept\}$

$Dept \rightarrow Bldg$

- The FD Dept $\rightarrow$ Bldg does not define a key and leads to problems.

Lecture-17

Storage redundancy: The information about Department 3 is repeated three times.

Update anomaly: If the building of Department 3 is to be changed, three updates are necessary.

Unnecessary information dependency:

- Information about an employee who does not have a department requires null values.
- Information about a department cannot be represented unless at least one employee works in it.

Lecture-17

Approaches to Normalization

Approaches to normalization: There are two principle approaches to normalization, and each will be considered in these slides.

Decomposition: Break larger relations into smaller ones.

Synthesis: Begin with a set of dependencies (usually FDs), and construct a corresponding relational schema.

The changes forced by normalization: Generally speaking, by forcing FDs to define (super)key dependencies, the problems identified above are minimized or disappear completely... but the devil is in the details.

Lecture-17

Normal Forms

Normal forms: In early research on the relational model, a number of so called *normal forms* were developed.

- The principal ones which are based upon FDs were developed in the following order:

$$1NF \rightarrow 2NF \rightarrow 3NF \rightarrow BCNF$$

- There are some others which are based upon other types of dependencies: 4NF, 5NF, DKNF.

Lecture-17

- For pedagogical reasons, they will considered in the reverse order of development:

$$\text{BCNF} \rightarrow \text{3NF} \rightarrow \text{2NF} \rightarrow \text{1NF}$$

- The main focus will be upon BCNF and 3NF, as 2NF is largely of historical interest and 1NF is just a constraint on domains.

Lecture-17(Normal Forms)

Design "Anomalies"

This Apply relation exhibits three types of anomalies:

1.Redundancy

2.Update Anomaly

3.Deletion Anomaly

Normalization

Design "Anomalies"

This Apply relation exhibits three types of anomalies:

1.Redundancy

2.Update Anomaly

3.Deletion Anomaly

Lecture-17

(Class-Assignment)

Explain Normalization with design issues.

Thank You