Lecture 6: Functional dependencies and normalization

Motivation

- How can we be sure that a particular database is well-designed?
- Can the data base be redesigned so that the same information and relationships are stored in a more efficient, robust way?

Definitions: Functional dependencies

- Let R be a relational schema with the attributes $A_1,...,A_n$ and let X and Y be subsets of $\{A_1,...,A_n\}$.
- Let r(R) denote a relation in the relational schema R.

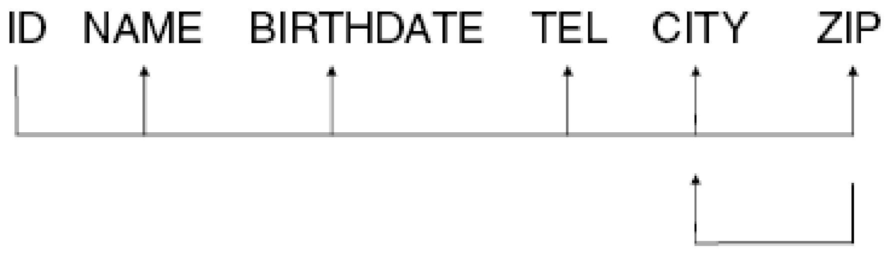
We say that X *functionally determines* Y, i.e. X → Y,
if for each pair of tuples $t_1, t_2 \in r(R)$ and for all relations r(R) we have that if $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$.

R= {ID, NAME, BIRTHDATE, TEL, CITY, ZIP}

X is called a determinant

$$\left.\begin{array}{l} ID \rightarrow NAME \\ ID \rightarrow BIRTHDATE \\ ID \rightarrow TEL \\ ID \rightarrow CITY \\ ID \rightarrow ZIP \end{array}\right\} ID \rightarrow NAME,BIRTHDATE,TEL,CITY,ZIP$$

ZIP → CITY



Inference rules

1. If $X \supseteq Y$ then $X \rightarrow Y$, or $X \rightarrow X$ (reflexive rule)
2. If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z (augmentation rule)
3. If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$ (transitive rule)
4. If $X \rightarrow YZ$, then $X \rightarrow Y$ (decomposition rule)
5. If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$ (union or additive rule)
6. If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$ (pseudotransitive rule)

We denote the union of two sets X and Y by XY ; AA=A

Inference rules

- Prove that this statement is wrong
- ”… X → A, and Y → B does *not* imply that XY → AB.”

- Prove inference rules 4, 5 and 6 by using **only** ”smaller” inference rules.

Definitions

- **Superkey:** A set of attributes that uniquely (**but not necessarily minimally**) identifies a tuple of a relation.
- **Candidate key :** A set of attributes that uniquely and **minimally** identifies a tuple of a relation.
- **Primary key:** **A particular** candidate key is chosen to be the primary key.
- **Prime attribute:** Every attribute that is part of a **candidate key** (vs. nonprime attribute).

Good design: Formal measure, normalization

<u>ID</u>	**Name**	**Zip**	**City**
<u>100</u>	Andersson	58214	Linköping
<u>101</u>	Björk	10223	Stockholm
<u>102</u>	Carlsson	58214	Linköping

- 1NF, 2NF, 3NF, BCNF (4NF, 5NF, 6NF).
- Minimize redundancy.
- Minimize update anomalies.
- The higher the normal form, the less the redundancy. Moreover, relations become more but smaller.
- Join operations are needed to recover the original relations.
- This may harm running time. So, normalization is not mandatory. In some cases, even denormalization may be preferred. In these cases, you may want to deal with redundancy via coding (e.g. procedures, triggers, views, etc.).

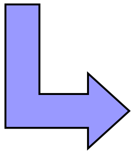
First normal form (1NF)

- The relational model has only **atomic values** and any relation **have PK**.

R$_{\text{non1NF}}$

ID	**Name**	**LivesIn**
100	Pettersson	{Stockholm, Linköping}
101	Andersson	{Linköping}
102	Svensson	{Ystad, Hjo, Berlin}

This is how we dealt with multi-valued attributes in the translation.



R1$_{\text{1NF}}$

ID	**Name**
100	Pettersson
101	Andersson
102	Svensson

R2$_{\text{1NF}}$

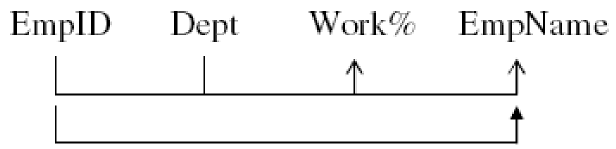
ID	***LivesIn***
100	*Stockholm*
100	*Linköping*
101	*Linköping*
102	*Ystad*
102	*Hjo*
102	*Berlin*

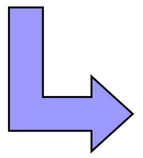
Second normal form (2NF)

- The relational model does not have any set of **non-prime** attributes that is functionally dependent on **part of a candidate key**.

R$_{non2NF}$

EmpID	***Dept***	**Work%**	**EmpName**
100	*Dev*	50	Baker
100	*Support*	50	Baker
200	*Dev*	80	Miller





Normalization

R1$_{2NF}$

EmpID	**EmpName**
100	Baker
200	Miller

R2$_{2NF}$

EmpID	***Dept***	**Work%**
100	*Dev*	50
100	*Support*	50
200	*Dev*	80

Second normal form (2NF)

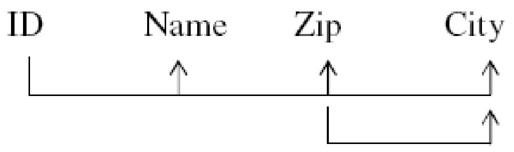
- The relational model does not have any set of **non-prime** attribute that is functionally dependent on **part of a candidate key**.
  - Why not ? Because, a part of a candidate key can have repeated values in the relation and, thus, so can have the set of non-prime attributes, which implies redundancy and thus waste of space and update anomalies.
  - Solution:
    - Assume that X $\rightarrow$ Y violates 2NF in a relational schema R.
    - Create a new relational schema R2(X,Y).
    - Remove Y from R.
    - The relational schema consists now of R and R2.

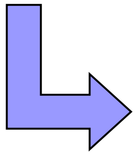
Third normal form (3NF)

- The relational model does not have any set of **non-prime** attributes that is functionally dependent on a set of attributes that **is not a candidate key**.

R$_{\text{non3NF}}$

ID	**Name**	**Zip**	**City**
100	Andersson	58214	Linköping
101	Björk	10223	Stockholm
102	Carlsson	58214	Linköping





Normalization

R1$_{\text{3NF}}$

ID	**Name**	**Zip**
100	Andersson	58214
101	Björk	10223
102	Carlsson	58214

R2$_{\text{3NF}}$

Zip	**City**
58214	Linköping
10223	Stockholm

Third normal form (3NF)

- The relational model does not have any set of **non-prime** attributes that is functionally dependent on a set of attributes that **is not a candidate key**.
  - Why not ? Because, a set of attributes that is not a candidate key can have repeated values in the relation and, thus, so can have the set of non-prime attributes, which implies redundancy and thus waste of space and update anomalies.
  - Note that 3NF implies 2NF.
  - Solution:
    - Assume that X → Y violates 3NF in a relational schema R.
    - Create a new relational schema R2(X,Y).
    - Remove Y from R.
    - The relational schema consists now of R and R2.

Little summary

- $X \rightarrow Y$
- 2NF and 3NF do nothing if Y is prime.
- Assume Y is non-prime.
- 2NF = decompose if X is part of a candidate key.
- 3NF = decompose if X is not a candidate key.
- 3NF = X is a candidate key or Y is prime.

Boyce-Codd normal form (BCNF)

- In every functional dependency in the relational model, the determinant **is a candidate key.**
- Example: Let R(<u>A,B</u>,C,D) denote a relational schema with functional dependencies AB→CD, C→B. Then R is in 3NF but not in BCNF.
  - C is a determinant but not a candidate key.
  - Decompose R into R1(<u>A,C</u>,D) with AC → D and R2(<u>C</u>,B) with C → B.
- In general:
  - Assume that X → Y violates BCNF in a relational schema R.
  - Create a new relational schema R2(X,Y).
  - Remove Y from R.
  - The relational schema consists now of R and R2.
  - You may have to find a new primary key for R.

Little summary

- $X \rightarrow Y$
- 2NF and 3NF do nothing if Y is prime.
- Assume Y is non-prime.
- 2NF = decompose if X is part of a candidate key.
- 3NF = decompose if X is not a candidate key.
- 3NF = X is a candidate key or Y is prime.
- Assume Y is prime.
- BCNF = decompose if X is not a candidate key.

Normalization: Example

- Consider the following relational schema: store number of times a person has been in a country

R(PID, PersonName, Country, Continent, ContinentArea, NumberVisitsCountry)

- Functional dependencies ?
- Candidate keys ?

Normalization: Example

- Functional dependencies:

PID → PersonName

PID, Country → NumberVisitsCountry

Country → Continent

Continent → ContinentArea

Normalization: Example

Is
R (PID,Country,Continent,ContinentArea,PersonName,NumberVisitsCountry)
in 2NF ?

No, PersonName depends on a part of a candidate key (PID), then

R1(PID, PersonName)

R2(PID, Country, Continent, ContinentArea, NumberVisitsCountry)

Is R1 in 2NF ? Yes.

Is R2 in 2NF ? No, Continent and ContinentArea depend on a part of a candidate key (Country), then

R1(PID, PersonName)

R21(Country, Continent, ContinentArea)

R22(PID, Country, NumberVisitsCountry)

Now, R1, R21, R22 are in 2NF.

2NF: No non-prime attribute should be functionally dependent on a part of a candidate key.

3NF: No nonprime attribute should be functionally dependent on a set of attributes that is not a candidate key.

Are R1, R21, R22 in 3NF?

R22(PID, Country, NumberVisitsCountry)
R1(PID, PersonName)
Yes, because they have only one non-prime attribute.

R21(Country, Continent, ContinentArea)
No, Continent determines ContinentArea, then
R211(Country, Continent)
R212(Continent, ContinentArea)

Now, R1, R22, R211, R212 are in 3NF.

Are R1, R22, R211, R212 in BCNF?

BCNF: Every determinant is a candidate key.

R22(<u>PID, Country</u>, NumberVisitsCountry)

R1(<u>PID</u>, PersonName)

R211(<u>Country</u>, Continent)

R212(<u>Continent</u>, ContinentArea)

Yes, they are in BCNF.

- **Can the original relation R be recovered by joining R1, R22, R211 and R212 without generating spurious tuples? Yes. Mind the foreign keys created during normalization !**

Desirable properties of normalization

- Keep all the attributes from the original relational model (true in our method).
- Preserve all the functional dependencies from the original relational model (false in our method).
- Lossless join (true in our method).
  - It must be possible to join the smaller relations produced by normalization and recover the original relation without generating spurious tuples.

Definition

- Minimal cover: minimal set of equivalent functional dependencies.

Motivation

- How can we be sure that a particular database is well-designed?
- What ***is*** a good database design?
  - Four informal measures.
  - One formal measure: Normalization.

Good design: Informal measures

- Easy-to-explain semantics of the relational schema.

- Minimal redundant information in tuples.
  - Why ? Redundancy causes waste of space and update anomalies:
    - Insertion anomalies.
    - Deletion anomalies.
    - Modification anomalies.

EMP(	**EMPID,**	**EMPNAME,**	**DEPTNAME,**	**DEPTMGR)**
	123	Smith	Research	999
	333	Wong	Research	999
	888	Borg	Administration	null

Good design: Informal measures

- Minimal number of NULL values in tuples.
  - Why ?
    - Efficient use of space.
    - Avoid costly outer joins.
    - Ambiguous interpretation (unknown vs. doesn’t apply).
- Disallow the possibility of generating spurious tuples.
  - How ? By joining only on foreign key/primary key attributes.

- Minimal cover