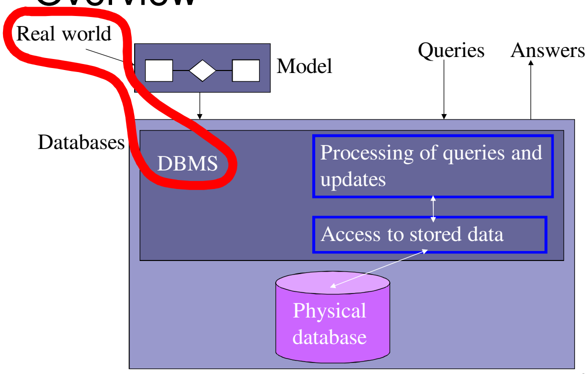
Functional Dependencies and Normalization

Jose M. Peña
jose.m.pena@liu.se

Overview



Good Design

- Can we be sure that a translation from EER-diagram to relational tables results in good database design?
- Confronted with a deployed database, how can we be sure that it is well-designed?
- What ***is*** good database design?
  - Four informal measures
  - Formal measure: normalization

Informal design guideline

- Easy to explain semantics of the relation schema
- Reducing redundant information in tuples

  Redundancy causes waste of space and update anomalies:
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

EMP(	<u>EMPID</u>,	EMPNAME,	DEPTNAME,	DEPTMGR)
	123	Smith	Research	999
	333	Wong	Research	999
	888	Borg	Administration	null

Informal design guideline

- Sometimes, it may be desirable to have redundancy to gain in runtime, i.e. trade space for time.
- In that case and to avoid update anomalies
  - either, use triggers or stored procedures to update the base tables
  - or, keep the base tables free of redundancy and use views (assuming that the views are materialized to avoid too many joins).

Informal design guideline

- Reducing 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
  - Figures 10.5 and 10.6: cartesian product results in incorrect tuples
  - <u>Only join on foreign key/primary key-attributes</u>
  - Lossless join property: guarantees that the spurious tuple generation problem does not occur

Remarks

- Relational schema: The header of the table.

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

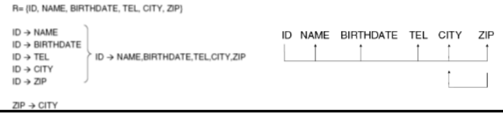
- Relation: The data in the table.
- Relation is a set, i.e. no duplicates exist.

Functional dependencies (FD)

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

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

- Despite the mathematical definition an FD cannot be determined automatically. It is a property of the semantics of attributes.



Inference rules

1. If X ⊇ Y then X→Y, or X → X (reflexive rule)
2. X→Y |= XZ → YZ (augmentation rule)
3. X → Y, Y → Z |= X → Z (transitive rule)
4. X → YZ |= X → Y (decomposition rule)
5. X → Y, X → Z |= X → YZ (union or additive rule)
6. X → Y, WY → Z |= WX → Z (pseudotransitive rule)

Inference rules

- Textbook, page 341:
  "… X → A, and Y → B does *not* imply that XY → AB."
  Prove that this statement is wrong.
- Prove inference rules 4, 5 and 6 by using **only** inference rules 1, 2 and 3.

Definitions

For any relation extension or state

- **Superkey:** a set of attributes uniquely (but not necessarily minimally!) identifying a tuple of a relation.
- **Key:** A ***set of attributes*** that uniquely and minimally identifies a tuple of a relation.
- **Candidate key:** If there is more than one ***key*** in a relation, the keys are called candidate keys.
- **Primary key:** One ***candidate key*** is chosen to be the primary key.
- **Prime attribute:** An attribute ***A*** that is part of a ***candidate*** key ***X*** (vs. nonprime attribute)

Normal Forms

- 1NF, 2NF, 3NF, BCNF (4NF, 5NF)
- ***Minimize redundancy***
- ***Minimize update anomalies***
- Normal form ↑ = redundancy and update anomalies ↓ and relations become smaller.
- Join operation to recover original relations.

1NF

- 1NF: The relation should have no non-atomic values.

R_{non1NF}

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

What about multi-valued attributes ?

Normalization

$R1_{1NF}$

ID	Name
100	Pettersson
101	Andersson
102	Svensson

$R2_{1NF}$

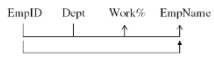
ID	LivesIn
100	Stockholm
100	Linköping
101	Linköping
102	Ystad
102	Hjo
102	Berlin

2NF

- 2NF: no **nonprime** attribute should be functionally dependent on a **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

2NF

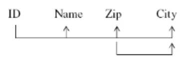
- No 2NF: A part of a candidate key can have repeated values in the relation and, thus, so can have the nonprime attribute, i.e. redundancy + insertion and modification anomalies.
- An FD X→Y is a **full functional dependency (FFD)** if removal of any attribute A_i from X means that the dependency does not hold any more.
- 2NF: Every **nonprime** attribute is fully functionally dependent on every candidate key.

3NF

- 3NF: **2NF +** no **nonprime** attribute should be functionally dependent on a set of attributes that is not a candidate key

R_{non3NF}

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



Normalization

$R1_{3NF}$

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

$R2_{3NF}$

Zip	City
58214	Linköping
10223	Stockholm

3NF

- No 3NF (but 2NF): A set of attributes that is not a candidate key can have repeated values in the relation and, thus, so can have the nonprime attribute, i.e. redundancy + insertion and modification anomalies.
- An FD X→Y is a **transitive dependency** if there is a set of attributes Z that is not a candidate key and such that both X→Z and Z→Y hold.
- 3NF: 2NF + no **nonprime** attribute is transitively dependent on any candidate key.

Little summary

- X → A
- 2NF and 3NF do nothing if A is prime.
- Assume A is nonprime.
- 2NF = decompose if X is part of a candidate key.
- 3NF = decompose if X is neither a candidate key nor part of a candidate key.
- 3NF = X is a candidate key or A is prime.

- If X is not a candidate key, then it can have repeated values in the relation and, thus, so can have A. Should this be ignored because A is prime ?

Boyce-Codd Normal Form

- BCNF: **Every determinant is a candidate key**
- BCNF = decompose if X → A is such that X is not a candidate key and A is a prime attribute.
- Example: Given R(<u>A,B</u>,C,D) and AB→CD, C→B. Then R is in 3NF but not in BCNF
  - C is a determinant but not a candidate key.
  - Decompose into R1(<u>A,C</u>,D) with AC → D and R2(<u>C</u>,B) with C → B.

BCNF: Example

At a gym, an instructor is leading an activity in a certain room at a certain time.

$R_{nonBCNF}$

<u>Time</u>	<u>Room</u>	Instructor	Activity
Mon 17.00	Gym	Tina	IronWoman
Mon 17.00	Mirrors	Anna	Aerobics
Tue 17.00	Gym	Tina	Intro
Tue 17.00	Mirrors	Anna	Aerobics
Wed 18.00	Gym	Anna	IronWoman

Time, room → instructor, activity
Time, activity → instructor , room
Time, instructor → activity, room
Activity → room

Decompose into R1(<u>Time,Activity</u>,Instructor) and R2(<u>Activity</u>,Room)

Properties of decomposition

- Keep all attributes from the universal relation R.
- Preserve the identified functional dependencies.
- Lossless join
  - It must be possible to join the smaller tables to arrive at composite information without spurious tuples.

Normalization: Example

Given universal relation

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

- Functional dependencies?
- Keys?

Normalization: Example

PID → PersonName
PID, Country → NumberVisitsCountry
Country → Continent
Continent → ContinentArea

- Based on FDs, what are keys for R?
- Use inference rules

Normalization: Example

Country → Continent, Continent → ContinentArea,
then
Country → Continent, ContinentArea (transitive + aditive rules)

PID, Country → Continent, ContinentArea (augmentation + decomposition rules),
PID, Country → PersonName (augmentation + decomposition rules),
PID, Country → NumberVisitsCountry,
then
PID, Country → Continent, ContinentArea, PersonName, NumberVisitsCountry (additive rule)

<u>PID, Country</u> is the key for R.

Normalization: Example

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

No, *PersonName* depends on a part of the candidate key (*PID*), then
R1(PID, PersonName)
R2(PID, Country, Continent, ContinentArea, NumberVisitsCountry)

Is R2 in 2NF?
No, *Continent* and *ContinentArea* depend on a part of the candidate key (*Country*), then
R1(PID, PersonName)
R21(Country, Continent, ContinentArea)
R22(PID, Country, NumberVisitsCountry)
→ R1, R21, R22 are in 2NF

2NF: *no nonprime attribute* should be functionally dependent on a ***part*** of a candidate key.

Are R1, R21, R22 in 3NF?

3NF: 2NF + no nonprime attribute should be functionally dependent on a set of attributes that is not a candidate key

R22(PID, Country, NumberVisitsCountry),
R1(PID, PersonName):
Yes, a single nonprime attribute, no transitive dependencies.

R21(Country, Continent, ContinentArea):
No, Continent defines ContinentArea, then
R211(Country, Continent)
R212(Continent, ContinentArea)

→ R1, R22, R211, R212 are in 3NF

Are R1, R22, R211, R212 in BCNF?

BCNF: Every determinant is a candidate key

R22(PID, Country, NumberVisitsCountry),
R1(PID, PersonName):
R211(Country, Continent)
R212(Continent, ContinentArea)

→ Yes

Can the universal relation R be reproduced from R1, R22, R211 and R212 without spurious tuples?

Summary and open issues

- Good design: informal and formal properties of relations
- Functional dependencies, and thus normal forms, are about attribute *semantics* (= real-world knowledge), normalization can only be automated if FDs are given.
- Are high normal forms good design when it comes to performance?
  - No, denormalization may be required.