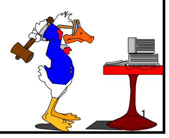
Normalisation



Introduction

- The normalization process, as first proposed by Codd,
- takes a relation schema through a series of tests to "certify" whether it satisfies a certain normal form.

2

Introduction

- The process, which proceeds in a top-down fashion by evaluating each relation against the criteria for normal forms and decomposing relations as necessary,
- can thus be considered as *relational design by analysis*

3

Introduction

- 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.

4

Introduction

- All these normal forms are based on the functional dependencies among the attributes of a relation.
- Later, a fourth normal form (4NF) and a fifth normal form (5NF) were proposed,
- based on the concepts of multivalued dependencies and join dependencies, respectively;

5

Introduction

- Normalizing a logical database design involves using formal methods to separate the data into multiple related tables.
- The characteristics of normalised database are a large number of tables with few columns.
- An Unnormalised table suffers from insertion, deletion and update anomalies.

6

The purpose of Normalization

- The reduction in columns of a normalised table means fewer indexes are required, this in turn improves the performance of database querying.
- The opportunity for database inconsistency is reduced.

The purpose of Normalization cont:

- There will be fewer null values for data that is either not required or not known.
- Normalization aims to avoid redundant duplication.
- A normalised relation include faster sorting

Normal Forms

- A FD X → Y is a *full functional dependency* if removal of any attribute from X means that the dependency does not hold any more;
- otherwise, it is a *partial functional dependency*.

Full & Partial Dependency

Supp-Part (S#, Sname, P#, PackSize)

Full Key Dependency:

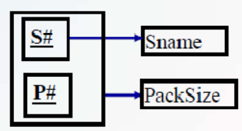
{S#,P#}➔ {Packsize}

Partial Key Dependency:

S# ➔ Sname

Supp-Part

S#	SNAME	P#	PACKSIZE
S2	Jones	P1	10
S7	Smith	P6	25
S2	Jones	P4	40
S5	Jones	P1	20



1st Normal Form

- An attribute is *prime* if it is a member of *any* key (Primary or candidate).
- A relation R is in *first normal form* if domains of attributes include only atomic values.

1st Normal Form

- This implies that we should disallow composite attributes, multi-valued attributes, and nested relations
- In other words, forbid all attributes whose values for an individual tuple are non-atomic

The steps of transformation from UNF into 1NF

1. Nominate an attribute or group of attributes to act as the key for the unnormalized table.
2. Identify the repeating group(s) in the unnormalized table, which repeats for the key attribute(s).

13

The steps of transformation from UNF into 1NF

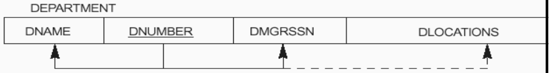
3 a. Remove the repeating group by entering appropriate data into the empty columns of tuples containing the repeating data ('flattening' the table),

3b. or by placing the repeating data along with a copy of the original key attribute (s) into a separate relation.

14

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

(a) DEPARTMENT



(b) DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c) DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

15

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.

(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

16

(c) Decomposing EMP_PROJ into 1NF relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

(c) EMP_PROJ1

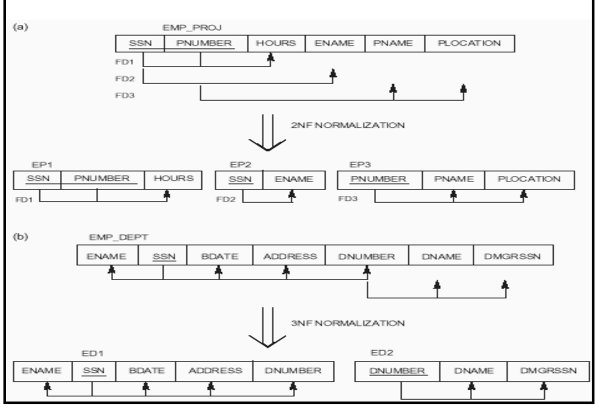
SSN	ENAME

EMP_PROJ2

SSN	PNUMBER	HOURS

17

Figure 14.10 The normalization process. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.



2nd Normal Form

- A relation R is in *second normal form* if every non-prime attribute A in R is not partially dependent on any key of R.
- Alternatively, R is in 2NF if every non-prime attribute A in R is fully dependent on every key of R.

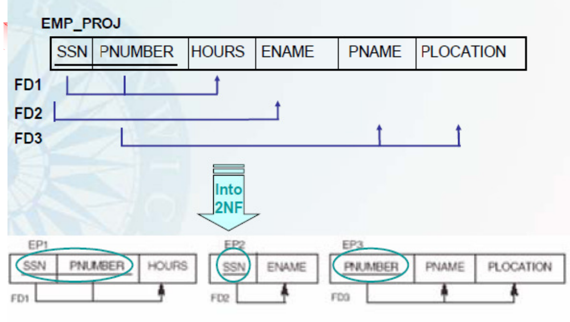
19

steps of transformation from 1NF into 2NF

1. Identify the primary key for the 1NF relation.
2. Identify the functional dependencies in the relation.
3. If partial dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their determinant.

20

Examples of transformation into 2NF



21

3rd Normal Form

- A relation R is in *third normal form* if for every FD X → A that holds on R, either
  - X is a superkey of R, or
  - A is a prime attribute of R.
- Alternative Def .
  - No transitive dependencies – If there is a set of attributes Z that is neither a candidate key nor a subset of any key (primary or candidate) of R , X → Z and Z → Y holds.

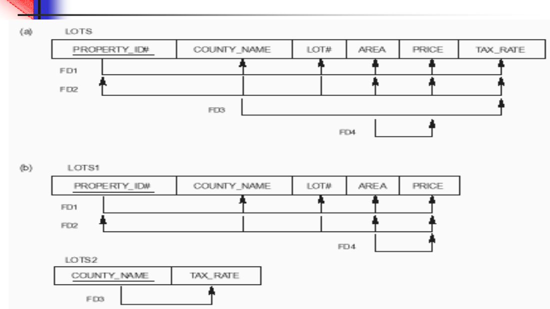
22

steps of transformation from 2NF into 3NF

1. Identify the primary key in the 2NF relation.
2. Identify functional dependencies in the relation.
3. If transitive dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their determinant (dominant).

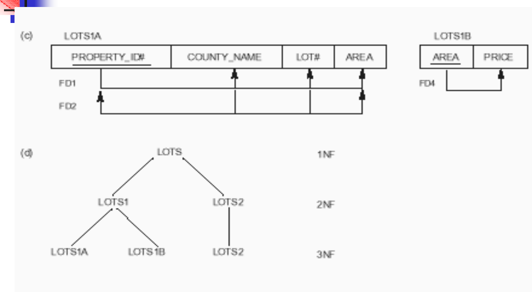
23

Normalization to 2NF and 3NF. (a) The lots relation schema and its functional dependencies FD1 through FD4. (b) Decomposing lots into the 2NF relations LOTS1 and LOTS2.



24

(c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of normalization of lots.



25

Boyce-Codd normal

- A relation R is in *Boyce-Codd normal form* if for every FD X → A that holds on R, X is a superkey of R.
- A relation is in BCNF, if and only if every determinant is a candidate key.

26

steps of transformation from 3NF into BCNF

1. Identify all candidate keys in the relation.
2. Identify all functional dependencies in the relation.
3. If functional dependencies exist in the relation where their determinants are not candidate keys for the relation, remove the functional dependencies by placing them in a new relation along with a copy of their determinant.

27

BCNF / Example /1

– Consider this scenario:
  – The DSD company provides end user software training in Database, Network & Spreadsheets
  – DSD employs several trainers in each of the three subject.
  – Each trainer teaches only one subject, that is a Database trainer teaches Database only.
  – Corporate customers may elect to purchase training contracts for one or more subjects.

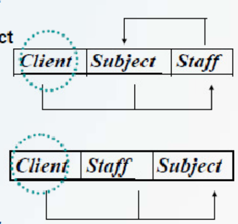
Client	Subject	Staff
1001	Database	Ala
1001	Network	Sati
1002	Database	Ala
1003	Spreadsheet	Phil
1004	Database	Alun

28

BCNF / Example /2

- 2 composite candidate keys:
  - FD1:{Client, Subject} → Staff
  - FD2:{Client & Staff} → Subject
- These candidate keys are overlapping on Client.

Staff **is a determinate but not a candidate key**



29

- What is the Problem?
  - Anomalies
- Delete client 1004 will also delete Tony teaches Database. So is for client 1001 on Network.
- Hence, we need to decompose table into two to get rid of redundancies.

Client	Subject	Staff
1001	Database	Ala
1001	Network	Sati
1002	Database	Ala
1003	Spreadsheet	Phil
1004	Database	Alun

30

Client	Staff

Client	Staff
1001	Ala
1001	Sati
1003	Phil
1004	Alun
1002	Ala

Staff	Subject

Staff	Subject
Ala	Database
Sati	Network
Phil	Spreadsheet
Alun	Database

31

Decomposition.

- A more purist way –
- Normalization: a process in which unsatisfactory relational schemas are decomposed into smaller relation schemas that possess desirable properties.
- Starting with a single *universal relation schema* R = A_1, A_2,…. A_n that includes *all* the attributes of the database.

32

Decomposition

- Decompose R into a set of relation schemas D ={R_1, R_2,… R_m} using the FDs specified by the database designers.
- D is called a *decomposition* of R.

33

Properties of Decompositions

- There are three important properties of a decomposition:
  - Attribute preservation
  - Lossless Join
  - Dependency Preservation

34

Attribute preservation property

- : Each attribute in R will appear in at least one relation schema R_i in the decomposition so that no attributes are "lost".
- Another goal of decomposition is to have each individual relation R_i in the decomposition D be in BCNF or 3NF.

35

Dependency Preservation Property

Definition:

- Given a set of dependencies *F* on *R*, the **projection** of *F* on R_i, denoted by $\pi_{R_i}(F)$ where R_i is a subset of *R*, is the set of dependencies $X \rightarrow Y$ in F^+ such that the attributes in $X \cup Y$ are all contained in R_i.

36

Dependency Preservation Property

- Hence, the projection of F on each relation schema R_i in the decomposition D is the set of functional dependencies in F^+, the closure of F, such that all their left- and right-hand-side attributes are in R_i.

37

Dependency Preservation Property:

- a decomposition $D = \{R_1, R_2, ..., R_m\}$ of R is **dependency-preserving** with respect to F if the union of the projections of F on each R_i in D is equivalent to F; that is,
  $((\pi_{R1}(F)) \cup ... \cup (\pi_{Rm}(F)))^+ = F^+$
- **Claim 1:** It is always possible to find a dependency-preserving decomposition D with respect to F such that each relation R_i in D is in 3NF.

38

Lossless join property

Definition:

- a decomposition $D = \{R_1, R_2, ..., R_m\}$ of R has the **lossless (nonadditive) join property** with respect to the set of dependencies F on R if, for *every* relation state r of R that satisfies F, the following holds, where * is the natural join of all the relations in D:
  $* (\pi_{R1}(r), ..., \pi_{Rm}(r)) = r$

39

Testing for a Lossless Join

- If we project R onto $R_1, R_2, ..., R_k$, can we recover R by rejoining?
- Any tuple in R can be recovered from its projected fragments.
- So the only question is: when we rejoin, do we ever get back something we didn't have originally?

40

Example of Lossy-Join Decomposition

- Lossy-join decompositions result in information loss.
- Example: Decomposition of $R = (A, B)$
  $R_1 = (A)$ $R_2 = (B)$

A	B
α	1
α	2
β	1

r

A
α
β

$\Pi_A(r)$

B
1
2

$\Pi_B(r)$

$\Pi_A(r) \bowtie \Pi_B(r)$

A	B
α	1
α	2
β	1
β	2

41

3NF Synthesis Algorithm

- Given a relation schema R and a set of FDs F,
- The following steps produce a 3NF decomposition of R that satisfies the lossless join condition and is dependency preserving:
- Find a minimal cover for F, say G.

42

3NF Synthesis Algorithm

- For each FD X -> A in G, use XA as the schema of one of the relations in the decomposition.
- If none of the schemas from Step 2 includes a superkey for R, add another relation schema that is a key for R.
- 4. Delete any of the schemas from Step 2 that is contained in another.

43

minimal cover for the FD's:

1. Right sides are single attributes.
2. No FD can be removed.
3. No attribute can be removed from a left side.

44

Constructing a Minimal Basis

1. Split right sides.
2. Repeatedly try to remove an FD and see if the remaining FD's are equivalent to the original.
3. Repeatedly try to remove an attribute from a left side and see if the resulting FD's are equivalent to the original.

45

Example 1: 3NF Synthesis

- Address (Street, City, Postcode)
- F= {SC -> P, P -> C}.
- Step 1 of the algorithm finds that F3 is a minimal cover.
- Step 2 of the algorithm would produce {P,C} and {S,C,P}.
- Step 3 finds that SC is a superkey.
- Step 4 deletes {P,C} to leave just {S,C,P}.

46

Example 2

- schema(S) = {ENAME, CNAME, SAL}
- F = {Ename -> Salary}.
- Step 1 of the algorithm finds that F2 is a minimal cover.
- Step 2 of the algorithm would produce {E,S}.
- Step 3 finds no superkey, so adds relation schema {E,C}.
- Step 4 finds nothing to delete.

47

Example 3

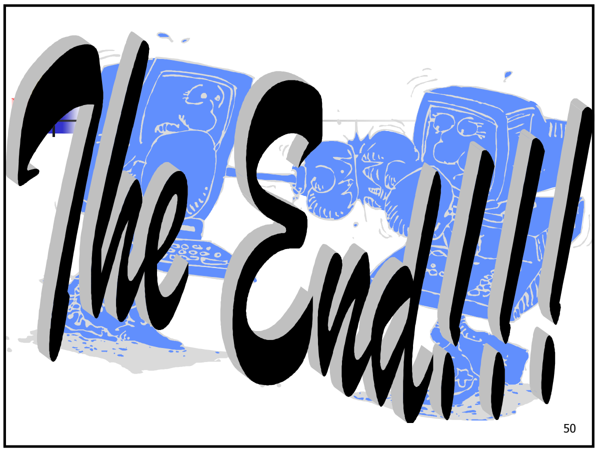
- F = {AB->CD, C -> AD, D -> A }.
- Step 1 of the algorithm finds that F is **not a minimal cover.**
- First we form a canonical set of FDs:
- {AB -> C, AB -> D, C -> A, C -> D, D -> A }.
- Then we find that AB -> D and C -> A are redundant.
- So we are left with minimal cover
- G = {AB -> C, C -> D, D -> A }.
- Try the rest.

48

Exercise

- Consider the set of attributes { Drinker, Address, Pub, Location, Beer, Cost }, along with the following set of FDs:
  - Drinker -> Address
  - Pub -> Location
  - Pub, Beer -> Cost, Location
- Produce a set of 3NF relation schemas for the above.

49



50