

COMP67321

Normalisation in relational modelling

Goran Nenadic

School of Computer Science
University of Manchester

1

Aims

- Understand potential anomalies in RDBM
- Understand normalisation of relations
 - functional dependencies
 - normal forms and transformations

2

Plan

- Pitfalls in relational database design
- Database design guidelines
- Normalisation
 - functional dependencies
 - normal forms
 - multi-valued dependencies

3

Pitfalls in RDB design

- Relational database design requires that we find a “good” collection of relation schemas.
- A bad design may lead to
 - repetition of information
 - inability to represent certain information
 - update anomalies
 - spurious tuples

4

‘Suitable Set of Relations’

- Characteristics of a suitable set of relations include:
 - the *minimal* number of attributes necessary to support the data requirements of the enterprise;
 - attributes with a close logical relationship are found in the same relation;
 - *minimal* redundancy with each attribute represented only once with the important exception of attributes that form all or part of foreign keys.

5

continued

Pitfalls in RDB design

- Design goals
 - avoid redundant data
 - ensure that relationships among attributes are represented
 - facilitate the checking of updates for violation of database integrity constraints.

6

Data redundancy and update anomalies

- The major aim of relational DB design is to group attributes into relations to minimise data redundancy
- Potential benefits
 - updates to the data stored in the database are achieved with a minimal number of operations thus reducing the opportunities for data inconsistencies.
 - reduction in the file storage space required by the base relations thus minimizing costs.

7

Redundancy

<i>branch-name</i>	<i>branch-city</i>	<i>assets</i>	<i>customer-name</i>	<i>loan-number</i>	<i>amount</i>
Downtown	Brooklyn	9000000	Jones	L-17	1000
Redwood	Palo Alto	2100000	Smith	L-23	2000
Perryridge	Horseneck	1700000	Hayes	L-15	1500
Downtown	Brooklyn	9000000	Jackson	L-14	1500

- Data for *branch-name*, *branch-city*, *assets* are repeated for each loan that a branch makes
- Wastes space
- Complicates updating, introducing possibility of inconsistency of *assets* value

8

Inability to represent data

<i>branch-name</i>	<i>branch-city</i>	<i>assets</i>	<i>customer-name</i>	<i>loan-number</i>	<i>amount</i>
Downtown	Brooklyn	9000000	Jones	L-17	1000
Redwood	Palo Alto	2100000	Smith	L-23	2000
Perryridge	Horseneck	1700000	Hayes	L-15	1500
Downtown	Brooklyn	9000000	Jackson	L-14	1500

- Cannot store information about a branch if no loans exist
- Can use null values, but they are difficult to handle.

9

Guideline 1: semantics of the relation and attributes

- Informally, each tuple in a relation should represent one entity or relationship instance
- Attributes of different entities (EMPLOYEEs, DEPARTMENTS, PROJECTs) should not be mixed in the same relation (if possible)
- Only foreign keys should be used to refer to other entities

10

continued

Guideline 1: semantics of the relation and attributes

- Entity and relationship attributes should be kept apart as much as possible.
- Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.

11

Guideline 2: redundant information

Design a schema that does not suffer from the insertion, deletion and update anomalies.

If there are any anomalies present (due to performance reasons, e.g., in Data Warehouses), then note them so that applications can take them into account

12

Guideline 3: null values

- Reasons for nulls:
 - attribute not applicable or invalid
 - attribute value unknown (may exist)
 - value known to exist, but unavailable
- Relations should be designed such that their tuples will have as few NULL values as possible
- Attributes that are NULL frequently could be placed in separate relations

13

Spurious tuples

- Bad design may result in erroneous results for certain JOIN operations
- Spurious tuples are the result of bad decomposition and composition of relations (JOIN) and represent spurious or faulty information.
- Spurious tuples are created when two tables are joined on attributes that are neither primary keys nor foreign keys.

14

example

Spurious tuples

```
EMP_LOCS(ENAME, PLOCATION)
EMP_PROJ1(SSN, PNUMBER, HOURS, PNAME,
PLOCATION)
```

What would happen if we join EMP_LOCS and EMP__PROJ on PLOCATION?

15

Guideline 4: spurious tuples

The relations should be designed to satisfy the “lossless” join condition. No spurious tuples should be generated by doing a natural join of any relations.

Schemas should be designed so as to allow JOINS with equality conditions only on attributes that are keys

19

Normalisation

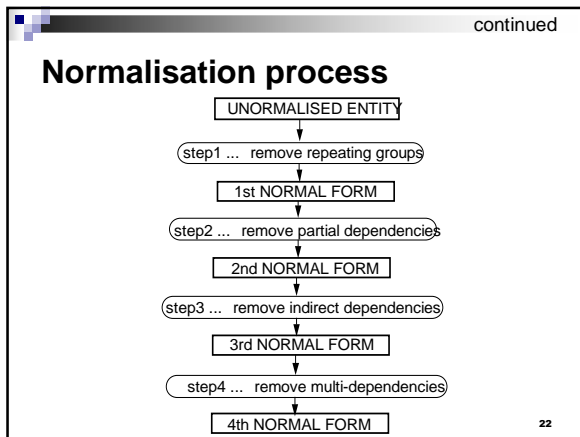
- Simplify the design of a DB by deriving relation structures that avoid update anomalies (Insertion, Deletion, Modification, Access)
 - produce a ‘suitable set of database relations’
- Based on the notion of **normal forms** which are a linear progression of rules used to achieve the optimum relational structure
- Benefits
 - easier for the user to access and maintain the data
 - minimise storage space

20

Normalisation process

- Based on the notions of *functional dependencies, primary key, candidate keys, multivalued dependencies and join dependencies*
 - First Normal Form (1NF)
 - Second Normal Form (2NF)
 - Third Normal Form (3NF)
 - Boyce Codd Normal Form (BCNF)
 - Fourth Normal Form (4NF)
 - Fifth Normal Form (5NF)

21



First Normal Form (1NF)

- A relation is said to be in 1NF if all attribute values are atomic (or indivisible)
- 1NF is part of the formal definition of a relation and it has been defined to disallow multi-valued attributes and composite attributes.

23

example

Example 1NF

STUDENT NUMBER		S0843215	
STUDENT NAME		P. Smith	
STUDENT ADDRESS		1, Upton Drive, Sale	
COURSE NO	TITLE	TUTOR NAME	TUTOR NO
PM951	Computing	T. Long	037428
S212	Biology	S. Short	096524

REG_FORM(Student_No, Student_Name, Student_Address, (Course_No, Title, Tutor_Name, Tutor_No))

24

example

1NF Example

- Remove non-atomic attributes by decomposition and generation of new relations

STUDENT(Student_No, Student_Name, Student_Address)

ENROLMENT(Student_No, Course_No, Title, Tutor_Name, Tutor_No)

25

continued

Normalisation process

- At each of the following steps, we consider relationships between an entity's attributes
 - these relationships are known as **functional dependencies**
- A functional dependency ($X \rightarrow Y$) is a constraint between two sets of attributes X and Y in a relation

26

Functional dependencies

- $X \rightarrow Y$ means that **X functionally determines Y** if and only if whenever two tuples agree on their X-value, they must agree on their Y-value.
- Example
 - $X \rightarrow Y$ where
 - X = (Street Number, Street Name, County)
 - Y = (Post-Code)
- Not symmetric: if $X \rightarrow Y$, then $Y \rightarrow X$ may or may not be true

27

Functional dependencies

- All attributes of a relation R are functionally dependent on the primary key of the relation as well as any candidate keys of R
- A functional dependency is a property of the relation schema (intension) and not of its particular valid relation state (extension)

Inference rules

- Used to derive new dependencies from a given set of dependencies F
- The notation $F \models X \rightarrow Y$ denotes that the functional dependency $X \rightarrow Y$ is inferred from the set of functional dependencies F.
- The **closure** F^+ of F is the set of all functional dependencies that can be inferred from F.
- Two sets of functional dependencies E and F are equivalent if $E^+ = F^+$

Inference rules

- if $X \supseteq Y$ then $X \rightarrow Y$ (Reflexive rule)
- $\{X \rightarrow Y\} \models XZ \rightarrow YZ$ (Augmentation rule)
- $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$ (Transitive rule)
- $\{X \rightarrow YZ\} \models X \rightarrow Y, X \rightarrow Z$ (Decomposition or projective rule)
- $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$ (Union or additive rule)
- $\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$ (Pseudotransitive)

example

Functional dependencies

- Suppose that we specify the following set F of functional dependencies on a relation schema:

$$F = \{SSN \rightarrow \{ENAME, BDATE, ADDRESS, DNUMBER\}, \\ DNUMBER \rightarrow \{DNAME, DMGRSSN\}\}$$
- Give examples of additional functional dependencies that we can *infer* from F
- Is this true: $SSN \rightarrow DMGRSSN$

31

example

Functional dependencies

A	B	C	D	E
A1	B1	C1	D1	E1
A1	B2	C2	D2	E1
A2	B1	C3	D3	E1
A2	B1	C4	D3	E1
A3	B2	C5	D1	E1

Which of the following functional dependencies are satisfied?

- $A \rightarrow B$
- $\{A,B\} \rightarrow D$
- $C \rightarrow E$

32

example

Exercise

Would the following dependencies be full functional dependencies?

- $\{Surname, PassportNo\} \rightarrow First\ Name$
- $\{First\ Name, Surname, Date\ of\ Birth\} \rightarrow Maiden_Name$

33

Functional dependencies

- A set of functional dependencies F is **minimal** if it satisfies the following:
 1. Every dependency $X \rightarrow Y$ in F has a single attribute in the right-hand side Y
 2. We cannot remove any dependency from F and still have an equivalent set of FDs
 3. We cannot replace any dependency $X \rightarrow A$ in F with a dependency $Y \rightarrow A$ where Y is a proper subset of X and still have an equivalent set of FDs

Full functional dependency

- A functional dependency $X \rightarrow Y$ is a **full functional dependency** if removal of any attribute A from X means that the dependency does not hold
- Example
if $\{SSN, PNUMBER\} \rightarrow \{HOURS\}$ holds, then
 $SSN \rightarrow HOURS$ and/or
 $PNUMBER \rightarrow HOURS$ **do not** hold
- Otherwise it is **partial functional dependency**

Second Normal Form (2NF)

- A relation is said to be in 2NF if it is in 1NF and every non-key attribute is fully dependent on the (entire) primary key
- To transform a relation from 1NF to 2NF
 - identify all functional dependencies
 - rewrite relations so that each non-key attribute is functionally dependent on the whole of the primary key

example

2NF Example

STUDENT(Student_No, Student_Name, Student_Address)
 ENROLMENT(Student_No, Course_No, Title, Tutor_Name, Tutor_No)

Course_No -> Title
 Course_No -> Tutor_Name
 Course_No -> Tutor_No

- Rewrite relations for 2NF

STUDENT(Student_No, Student_Name, Student_Address)
 ENROLMENT(Student_No, Course_No)
 COURSE(Course_No, Title, Tutor_Name, Tutor_No)

37

example

2NF Example

PURCHASE ORDER			
ORDER NUMBER	1023		
SUPPLIER NUMBER	500028		
ORDER DATE	09/05/88		
DELIVERY DATE	25/07/88		
PART NO.	PART-DESC	QTY-ORD	PRICE
0463	Hook	150	15.00
1492	Bolt	1000	10.00
3164	Spanner	10	5.00
TOTAL			30.00

PURCHASE_ORDER (Order_No, Supplier_No, Order_Date, Delivery_Date, Total_Price)
 PURCHASE_ITEM (Order_No, Part_No, Part_Desc, Quantity_Ord, Price)

38

example

2NF Example

- Rewrite relations so as non-key attributes are functionally dependent on the primary key

PURCHASE_ORDER (Order_No, Supplier_No, Order_Date, Delivery_Date, Total_Price)
 PURCHASE_ITEM (Order_No, Part_No, Quantity_Ord, Price)
 PART(Part_No, Part_Desc)

39


Third Normal Form (3NF)

- A relation is said to be in 3NF if it is in 2NF and if all non-key attributes are **mutually independent**
- To transform a relation from 2NF to 3NF
 - identify functional dependencies between non-key attributes (called **transitive dependencies**)
 - rewrite relations so that each non-key attributes are functionally independent from each other

40

example

3NF Example

- Relations in 2NF
STUDENT(Student_No, Student_Name, Student_Address)
ENROLMENT(Student_No, Course_No)
COURSE(Course_No, Title, Tutor_No, Tutor_Name)

- Identify transitive dependencies and rewrite relations for 3NF
STUDENT(Student_No, Student_Name, Student_Address)
ENROLMENT(Student_No, Course_No)
COURSE(Course_No, Title, Tutor_No)
TUTOR(Tutor_Name, Tutor_No)

41

example

3NF Example

- Relations in 2NF
PURCHASE_ORDER (Order_No, Supplier_No, Order_Date, Delivery_Date, Total_Price)
PURCHASE_ITEM (Order_No, Part_No, Quantity_Ord, Price)
PART(Part_No, Part_Desc)
 - Identify transitive dependencies and rewrite relations for 3NF
PURCHASE_ORDER (Order_No, Supplier_No, Order_Date, Delivery_Date, Total_Price)
PURCHASE_ITEM (Order_No, Part_No, Quantity_Ord, Price)
PART(Part_No, Part_Desc)
- (No transitive dependencies)

42

continued

Third Normal Form (3NF)

- An alternative definition for the 3NF: A relation R is said to be in 3NF if it is in 2NF and whenever a functional dependency $X \rightarrow A$ holds, either
 - (a) X is a superkey of R or
 - (b) A is a prime attribute of R
- A set of attributes X is called **superkey** of R if it is true that no two tuples for X have the same values
- An attribute of a relational schema R is called **prime attribute** if it is a member of any key of R

43

Boyce-Codd normal form

- A relation is said to be in Boyce-Codd Normal Form (BCNF) if it is in 2NF and whenever a functional dependency $X \rightarrow A$ holds in R then X is a superkey of R
- BCNF is slightly more restrictive than 3NF and in practice most 3NF relations will also be in BCNF
- Ideally, database schemas should comply to BCNF

44

continued

Boyce-Codd normal form

- Difference between 3NF and BCNF is that for a functional dependency $A \rightarrow B$, 3NF allows this dependency in a relation if B is a primary-key attribute and A is not a candidate key, whereas, BCNF insists that for this dependency to remain in a relation, A must be a candidate key.
- Every relation in BCNF is also in 3NF. However, a relation in 3NF is not necessarily in BCNF.

45

Boyce-Codd normal form

- Violation of BCNF is quite rare.
 - In practice, mostly ignored
- The potential to violate BCNF may occur in a relation that:
 - contains two (or more) composite candidate keys;
 - the candidate keys overlap, that is have at least one attribute in common.

46

Multi-valued dependency

- Y is multi-valued dependent on X ($X \twoheadrightarrow Y$), if a particular value of X determines a set of values of Y and the set Y does not depend on any other attribute of the relation.

- For our example,
ENAME \twoheadrightarrow PNAME
ENAME \twoheadrightarrow DNAME

EMPLOYEE

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

- $X \twoheadrightarrow Y$ is called a **trivial** MVD if
(a) Y is a subset of X or (b) $X \cup Y = R$

47

Fourth Normal Form (4NF)

- A relation schema R is in 4NF with respect to a set of dependencies F if, for every nontrivial multivalued dependency $X \twoheadrightarrow Y$ in F^+ , X is a superkey for R
- To transform a relation from 3NF to 4NF
 - detect any multi-valued dependencies
 - decompose relation accordingly

48

Fifth Normal Form (5NF)

- A relation decomposed into two relations must have the lossless-join property, which ensures that no spurious tuples are generated when relations are reunited through a natural join
- However, there are requirements to decompose a relation into more than two relations. Although very rare, these cases are managed by join dependency and fifth normal form (5NF).
- 5NF: Defined as a relation that has no join dependency; typically ignored

49

exercise

Exercise

BOOK (Book_title, Authormame, Book_type, Listprice, Author_affil, Publisher)

- Assumptions:
 - Author_affil refers to the affiliation of author
 - Dependences:

Book_title → {*Publisher*, *Book_type*}
Book_type → *Listprice*
Authormame → *Author_affil*

- What normal form is the relation in? Explain your answer.

50

ER Model and Normalisation

- When an ER diagram is carefully designed, identifying all entities correctly, the tables generated from the ER diagram should not need further normalisation.
- However, in a real (imperfect) design there can be functional dependencies from non-key attributes of an entity to other attributes of the entity

51

ER Model and Normalisation

- E.g. *employee* entity with attributes *department-number* and *department-address*, and an FD *department-number* → *department-address*
 - a good design would have made department an entity
- FDs from non-key attributes of a relationship set are possible, but rare - most relationships are binary

De-normalisation for performance

- May want to use non-normalized schema for performance
 - E.g. many data warehouses are typically not in high normalised forms
- Example
 - displaying *customer-name* along with *account-number* and *balance* would require a join of *account* with *depositor*

De-normalisation for performance

- Alternative 1: use denormalised relation containing attributes of *account* as well as *depositor* with all above attributes
 - Faster lookup (+)
 - Extra space and extra execution time for updates (-)
 - Extra coding to deal with updates and possibility of error in extra code (-)
- Alternative 2: use a materialised view

Summary

- A database design can suffer from redundancy, inability to represent information and update anomalies
- Normalisation
 - supports the design by deriving relation structures that avoid these pitfalls
 - based on the notion of normal forms which are a linear progression of rules used to achieve the optimum relational structure
- Non-optimal relational structures are used in applications (e.g., lower query execution time)

55

Reading for this lecture

56



57
