Chapter 7: Normalization

Database System Concepts, 7th Ed.

Features of Good Relational Designs

- Suppose we combine *instructor* and *department* into *in_dep,* which represents the natural join on the relations *instructor* and *department*

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

- There is repetition of information
- Need to use null values (if we add a new department with no instructors)

Decomposition

- The only way to avoid the repetition-of-information problem in the i*n_dep* schema is to decompose it into two schemas – instructor and *department* schemas.
- Not all decompositions are good. Suppose we decompose

  employee(ID, name, street, city, salary)

  into

  employee1 (*ID*, *name*)

  employee2 (*name*, *street, city, salary*)

  The problem arises when we have two employees with the same name
- The next slide shows how we lose information -- we cannot reconstruct the original *employee* relation -- and so, this is a **lossy decomposition**.

A Lossy Decomposition

ID	*name*	*street*	*city*	*salary*
⋮				
57766	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

employee

ID	*name*
⋮	
57766	Kim
98776	Kim
⋮	

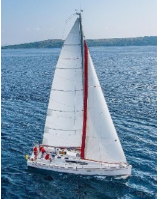
name	*street*	*city*	*salary*
⋮			
Kim	Main	Perryridge	75000
Kim	North	Hampton	67000
⋮			

natural join

ID	*name*	*street*	*city*	*salary*
⋮				
57766	Kim	Main	Perryridge	75000
57766	Kim	North	Hampton	67000
98776	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

Lossless Decomposition

- Let R be a relation schema and let R_1 and R_2 form a decomposition of R . That is R = R_1 U R_2
- We say that the decomposition is a **lossless decomposition** if there is no loss of information by replacing R with the two relation schemas R_1 U R_2



Example of Lossless Decomposition

- Decomposition of $R = (A, B, C)$

  $R_1 = (A, B)$ $R_2 = (B, C)$

r:

A	B	C
α	1	A
β	2	B

$\Pi_{A,B}(r)$:

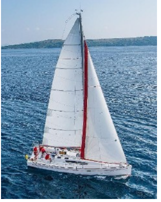
A	B
α	1
β	2

$\Pi_{B,C}(r)$:

B	C
1	A
2	B

$\Pi_{A}(r) \bowtie \Pi_{B}(r)$

A	B	C
α	1	A
β	2	B

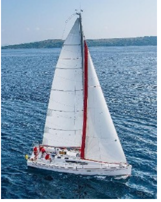


Goals of Normalization

- Let R be a relation scheme with a set F of functional dependencies.
- Decide whether a relation scheme R is in "good" form.
- In the case that a relation scheme R is not in "good" form, need to decompose it into a set of relation scheme $\{R_1, R_2, ..., R_n\}$ such that:
  - Each relation scheme is in good form
  - The decomposition is a lossless decomposition
  - Preferably, the decomposition should be dependency preserving.

Functional Dependencies

- There are usually a variety of constraints (rules) on the data in the real world.
- For example, some of the constraints that are expected to hold in a university database are:
  - Students and instructors are uniquely identified by their ID.
  - Each student and instructor has only one name.
  - Each instructor and student is (primarily) associated with only one department.
  - Each department has only one value for its budget, and only one associated building.



Functional Dependencies (Cont.)

- An instance of a relation that satisfies all such real-world constraints is called a **legal instance** of the relation;
- A legal instance of a database is one where all the relation instances are legal instances
- Constraints on the set of legal relations.
- Require that the value for a certain set of attributes determines uniquely the value for another set of attributes

Functional Dependencies Definition

- Let R be a relation schema

  $\alpha \subseteq R$ *and* $\beta \subseteq R$

- The **functional dependency**

  $\alpha \rightarrow \beta$

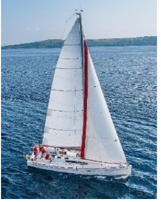
  holds on R if and only if for any legal relations r(R), whenever any two tuples t_1 and t_2 of r agree on the attributes α, they also agree on the attributes β. That is,

  $$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

- Example: Consider $r(A, B)$ with the following instance of r.

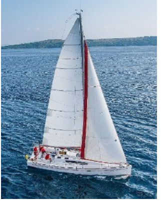
1	4
1	5
3	7

- On this instance, $B \rightarrow A$ hold; $A \rightarrow B$ does **NOT** hold,



Closure of a Set of Functional Dependencies

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



Dependency Preservation

- Testing functional dependency constraints each time the database is updated can be costly
- It is useful to design the database in a way that constraints can be tested efficiently.
- If testing a functional dependency can be done by considering just one relation, then the cost of testing this constraint is low
- When decomposing a relation it is possible that it is no longer possible to do the testing without having to perform a Cartesian Produced.
- A decomposition that makes it computationally hard to enforce functional dependency is said to be NOT **dependency preserving**.

Keys

Superkey: A superkey is a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation.

K is a **superkey** for relation schema R if and only if $K \rightarrow R$

Candidate key: A superkey may contain extraneous attributes. We are often interested in superkeys for which no proper subset is a superkey. Such minimal superkeys are called candidate keys.

K is a **candidate key** for R if and only if

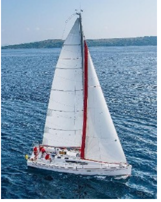
- $K \rightarrow R$, and
- for no $\alpha \subset K$, $\alpha \rightarrow R$

Primary key: We shall use the term primary key to denote a candidate key that is chosen by the database designer as the principal means of identifying tuples within a relation.

Keys

Prime key: Attributes of the relation which exist in at least one of the possible candidate keys, are called prime or key attributes.

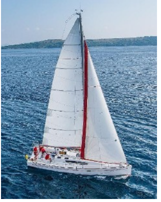
Non Prime key: Attributes of the relation which does not exist in any of the possible candidate keys of the relation, such attributes are called non prime or non key attributes.



Trivial 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 $\beta \subseteq \alpha$

Normal Forms



First Normal Form (1NF)

The table should only have single(atomic) valued attributes/columns.

Customer

Customer ID	First Name	Surname	Telephone Number
123	Pooja	Singh	555-861-2025, 192-122-1111
456	San	Zhang	(555) 403-1659 Ext. 53; 182-929-2929
789	John	Doe	555-808-9633

UNF

Customer

Customer ID	First Name	Surname	Telephone Number
123	Pooja	Singh	555-861-2025
123	Pooja	Singh	192-122-1111
456	San	Zhang	182-929-2929
456	San	Zhang	(555) 403-1659 Ext. 53
789	John	Doe	555-808-9633

1NF

Second Normal Form (2NF)

1. The table should be in the First Normal form
2. It should not have Partial Dependency (non-prime attribute cannot depend on some of the prime attributes, not all).

Electric toothbrush models

Manufacturer	Model	Model full name	Manufacturer country
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Brushmaster	SuperBrush	Brushmaster SuperBrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

1NF

Electric toothbrush models

Manufacturer	Model	Model full name
Forte	X-Prime	Forte X-Prime
Forte	Ultraclean	Forte Ultraclean
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush
Brushmaster	SuperBrush	Brushmaster SuperBrush
Kobayashi	ST-60	Kobayashi ST-60
Hoch	Toothmaster	Hoch Toothmaster
Hoch	X-Prime	Hoch X-Prime

Electric toothbrush manufacturers

Manufacturer	Manufacturer country
Forte	Italy
Dent-o-Fresh	USA
Brushmaster	USA
Kobayashi	Japan
Hoch	Germany

2NF

Third Normal Form (3NF)

1. The table should be in the Second Normal form
2. It should not have Transitive Dependency (non-prime attributes cannot depend on other non-prime attribute).

Tournament winners

Tournament	Year	Winner	Winner's date of birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

2NF

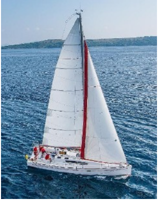
Tournament winners

Tournament	Year	Winner
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Winner's dates of birth

Winner	Date of birth
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

3NF



Boyce–Codd normal form (3.5NF)

1. The table should be in the Third Normal form
2. For any dependency A → B, A should be a super key (non-prime attribute cannot determine some of the prime attribute).

Only in rare cases does a 3NF table not meet the requirements of BCNF. A 3NF table that does not have multiple overlapping candidate keys is guaranteed to be in BCNF.

Depending on what its functional dependencies are, a 3NF table with two or more overlapping candidate keys may or may not be in BCNF.

Boyce–Codd normal form (3.5NF)

Each row in the table represents a court booking at a tennis club.
That club has one hard court (Court 1) and one grass court (Court 2)

A booking is defined by its Court and the period for which the Court is reserved. Additionally, each booking has a Rate Type associated with it.
There are four distinct rate types:
SAVER, for Court 1 bookings made by members
STANDARD, for Court 1 bookings made by non-members
PREMIUM-A, for Court 2 bookings made by members
PREMIUM-B, for Court 2 bookings made by non-members

Today's court bookings

Court	Start time	End time	Rate type
1	09:30	10:30	SAVER
1	11:00	12:00	SAVER
1	14:00	15:30	STANDARD
2	10:00	11:30	PREMIUM-B
2	11:30	13:30	PREMIUM-B
2	15:00	16:30	PREMIUM-A

3NF

Boyce–Codd normal form (3.5NF)

The table's superkeys are:

S1 = {Court, Start time}
S2 = {Court, End time}
S3 = {Rate type, Start time}
S4 = {Rate type, End time}
S5 = {Court, Start time, End time}
S6 = {Rate type, Start time, End time}
S7 = {Court, Rate type, Start time}
S8 = {Court, Rate type, End time}
ST = {Court, Rate type, Start time, End time}, the trivial superkey

Today's court bookings

Court	Start time	End time	Rate type
1	09:30	10:30	SAVER
1	11:00	12:00	SAVER
1	14:00	15:30	STANDARD
2	10:00	11:30	PREMIUM-B
2	11:30	13:30	PREMIUM-B
2	15:00	16:30	PREMIUM-A

3NF

The table does not adhere to BCNF. This is because of the dependency Rate type → Court in which the determining attribute Rate type on which Court depends – (1) is neither a candidate key nor a superset of a candidate key and (2) Court is no subset of Rate type.

Boyce–Codd normal form (3.5NF)

Rate types

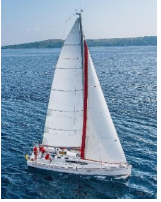
Rate type	Court	Member flag
SAVER	1	Yes
STANDARD	1	No
PREMIUM-A	2	Yes
PREMIUM-B	2	No

Today's bookings

Member flag	Court	Start time	End time
Yes	1	09:30	10:30
Yes	1	11:00	12:00
No	1	14:00	15:30
No	2	10:00	11:30
No	2	11:30	13:30
Yes	2	15:00	16:30

The candidate keys for the Rate types table are {Rate type} and {Court, Member flag}; the candidate keys for the Today's bookings table are {Court, Start time} and {Court, End time}.

Both tables are in BCNF. When {Rate type} is a key in the Rate types table, having one Rate type associated with two different Courts is impossible, so by using {Rate type} as a key in the Rate types table, the anomaly affecting the original table has been eliminated.



Comparison of BCNF and 3NF

- Advantages to 3NF over BCNF. It is always possible to obtain a 3NF design without sacrificing losslessness or dependency preservation.
- Disadvantages to 3NF.
  - We may have to use null values to represent some of the possible meaningful relationships among data items.
  - There is the problem of repetition of information.

Overall Database Design Process

We have assumed schema *R* is given

- *R* could have been generated when converting E-R diagram to a set of tables.
- *R* could have been a single relation containing *all* attributes that are of interest (called **universal relation**).
- Normalization breaks *R* into smaller relations.
- *R* could have been the result of some ad hoc design of relations, which we then test/convert to normal form.