Normalization of Relations

- **Normalization**:
  - The process of analysing the given relation schemas based on their FDs and PKs to achieve the desirable properties of (i) minimizing redundancy and (ii) minimizing the insertion, deletion and update anomalies
  - Unsatisfactory relation schemas that do not meet certain conditions (normal form tests) are decomposed into smaller relation schemas that meet the tests
  - Normalization procedure provides the DB designers with
    - A formal framework for analysing relation schemas based on their keys and on the FDs among their attributes
    - A series of normal form tests that can be carried out on individual relation schemas so that the relational database can be normalized to any desired degree

Normalization of Relations

- **Normal form**: Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form
  - Normal form of a relation refers to the highest normal form condition that it meets
  - Indicates the degree to which it has been normalized
- Process of normalization thru' decomposition must also confirm the existence of additional properties that the relational schemas, taken together, must possess
  - Lossless Join property
  - Dependency preservation property

Normalization of Relations

- 2NF, 3NF, BCNF based on keys and FDs of a relation schema
- 4NF based on keys, multi-valued dependencies : MVDs; 5NF based on keys, join dependencies : JDs
- Additional properties may be needed to ensure a good relational design (lossless join, dependency preservation)

3.2 Practical Use of Normal Forms

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The database designers ***need not*** normalize to the highest possible normal form. (usually up to 3NF, BCNF or 4NF)

Definitions of Keys and Attributes Participating in Keys

- A **superkey** of a relation schema $R = \{A_1, A_2, ..., A_n\}$ is a set of attributes S *subset-of* R with the property that no two tuples t_1 and t_2 in any legal relation state r of R will have $t_1[S] = t_2[S]$
- A **key** K is a superkey with the *additional property* that removal of any attribute from K will cause K not to be a superkey any more.

Definitions of Keys and Attributes Participating in Keys

- If a relation schema has more than one key, each is called a **candidate key.** One of the candidate keys is *arbitrarily* designated to be the **primary key,** and the others are called *secondary keys*.
- A **Prime attribute** must be a member of *some candidate key*
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

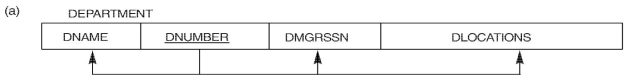
First Normal Form

- Disallows composite attributes, multivalued attributes and nested relations
  - The domain of an attribute must include only atomic values and the value of any attribute in a tuple must be a single value from the domain of that attribute

Considered to be part of the definition of relation

Normalization into 1NF

Figure 14.8 Normalization into 1NF. (a) Relation schema that is not in 1NF. (b) Example relation instance. (c) 1NF relation with redundancy.



(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

Second Normal Form

- Uses the concepts of **FD**s, **primary key**

Definitions:

- **Prime attribute** - attribute that is member of the primary key K
- **Full functional dependency** - a FD Y -> Z where removal of any attribute from Y means the FD does not hold any more

Examples: - {SSN, PNUMBER} -> HOURS is a full FD since neither SSN -> HOURS nor PNUMBER -> HOURS hold

- {SSN, PNUMBER} -> ENAME is *not* a full FD (it is called a *partial dependency*) since SSN -> ENAME also holds

Second Normal Form

- A relation schema R is in **second normal form** (**2NF**) if every non-prime attribute A in R is fully functionally dependent on the primary key
- R can be decomposed into 2NF relations via the process of 2NF normalization

Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Example state of relation DEPARTMENT. (c) 1NF version of same relation with redundancy.

(a)

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS

(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

Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a "nested relation" attribute PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

(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

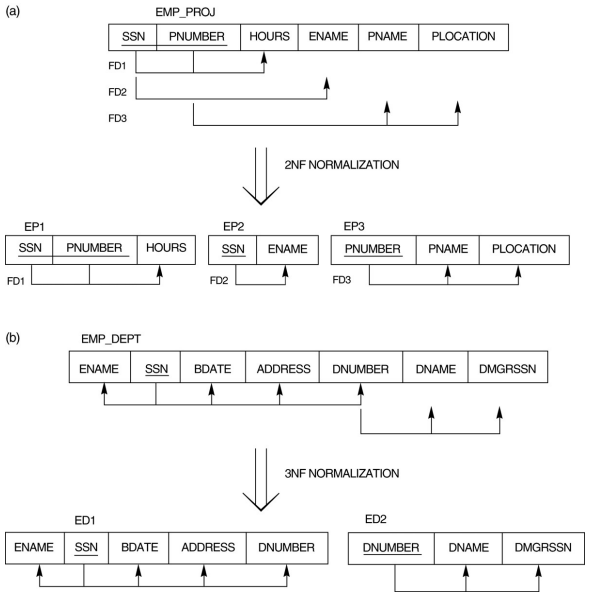
(c) **EMP_PROJ1**

SSN	ENAME

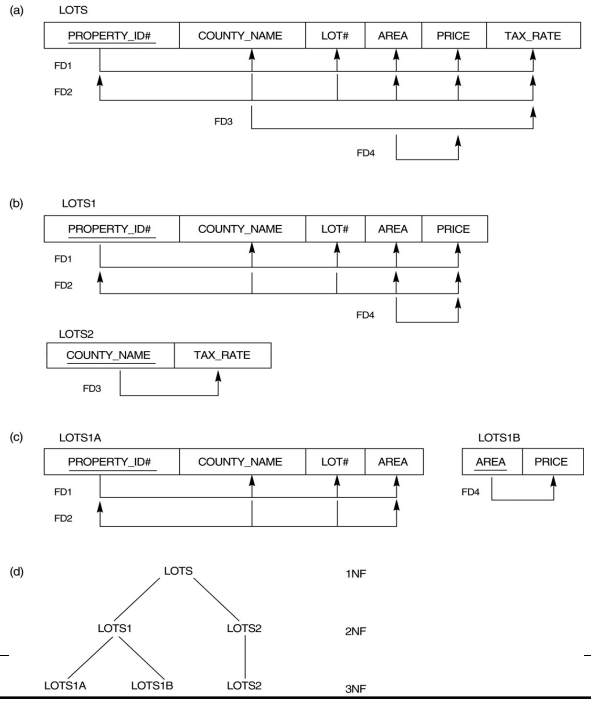
EMP_PROJ2

SSN	PNUMBER	HOURS

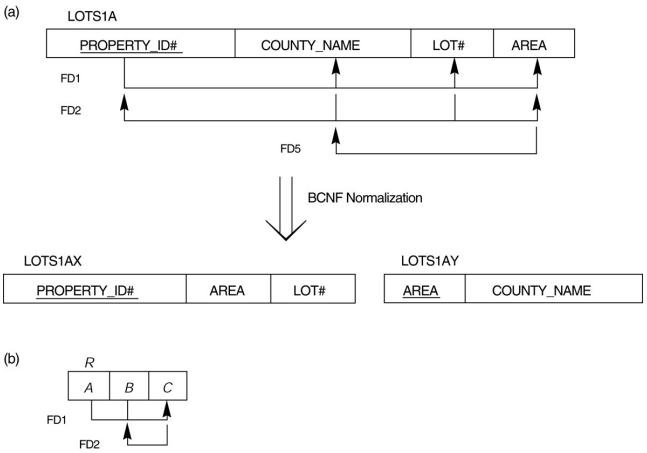
Normalizing into 2NF and 3NF.
(a) Normalizing EMP_PROJ into 2NF relations
(b) Normalizing EMP_DEPT into 3NF relations.



Normalization into 2NF and 3NF. (a) the LOTS relation with its functional dependencies FD1 though FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2.
(c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of the progressive normalization of LOTS.



Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.



A relation TEACH that is in 3NF but not BCNF.

TEACH

STUDENT	COURSE	INSTRUCTOR
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe

Third Normal Form

Definition:

- **Transitive functional dependency** - a FD X -> Z that can be derived from two FDs X -> Y and Y -> Z
  - A FD X→Y is a transitive dependency if there is a set of attributes Z that is neither a candidate key nor a subset of any key of R, and both X→ Z and Z→ Y hold

Examples:

- SSN -> DMGRSSN is a *transitive* FD since
SSN -> DNUMBER and DNUMBER -> DMGRSSN hold
- SSN -> ENAME is *non-transitive* since there is no set of attributes X where SSN -> X and X -> ENAME

Third Normal Form

- A relation schema R is in **third normal form** (**3NF**) if it is in 2NF *and* no non-prime attribute A in R is transitively dependent on the primary key
- R can be decomposed into 3NF relations via the process of 3NF normalization

NOTE:

In X -> Y and Y -> Z, with X as the primary key, we consider this a problem only if Y is not a candidate key. When Y is a candidate key, there is no problem with the transitive dependency .

E.g., Consider EMP (SSN, Emp#, Salary).

Here, SSN -> Emp# -> Salary and Emp# is a candidate key.

General Normal Form Definitions (For Multiple Keys)

- The above definitions consider the primary key only
- The following more general definitions take into account relations with multiple candidate keys
- A relation schema R is in **second normal form** (**2NF**) if every non-prime attribute A in R is fully functionally dependent on *every key* of R

General Normal Form Definitions

Definition:

- **Superkey** of relation schema R - a set of attributes S of R that contains a key of R
- A relation schema R is in **third normal form** (**3NF**) if whenever a FD X **->** A holds in R, then either:
  - (a) X is a superkey of R, or
  - (b) A is a prime attribute of R

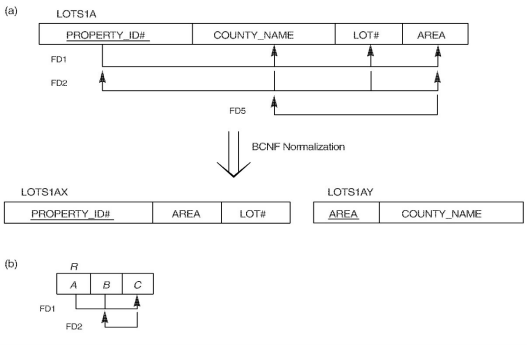
NOTE: Boyce-Codd normal form disallows condition (b) above

BCNF (Boyce-Codd Normal Form)

- A relation schema R is in **Boyce-Codd Normal Form** (**BCNF**) if whenever an FD X -> A holds in R, then X is a superkey of R
- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- There exist relations that are in 3NF but not in BCNF
- The goal is to have each relation in BCNF (or 3NF)

Boyce-Codd normal form

Figure 14.12 Boyce-Codd normal form. (a) BCNF normalization with the dependency of FD2 being "lost" in the decomposition. (b) A relation *R* in 3NF but not in BCNF.



A relation TEACH that is in 3NF but not in BCNF

Figure 14.13 A relation TEACH that is in 3NF but not in BCNF.

TEACH

STUDENT	COURSE	INSTRUCTOR
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe

Achieving the BCNF by Decomposition (1)

- Two FDs exist in the relation TEACH:
  fd1: { student, course} -> instructor
  fd2: instructor -> course
- {student, course} is a candidate key for this relation and that the dependencies shown follow the pattern in Figure 10.12 (b). So this relation is in 3NF but not in BCNF
- A relation **NOT** in BCNF should be decomposed so as to meet this property, while possibly forgoing the preservation of all functional dependencies in the decomposed relations. (See Algorithm 11.3)

Achieving the BCNF by Decomposition (2)

- Three possible decompositions for relation TEACH
  1. {student, instructor} and {student, course}
  2. {course, instructor } and {course, student}
  3. {instructor, course } and {instructor, student}
- All three decompositions will lose fd1. We have to settle for sacrificing the functional dependency preservation. But we cannot sacrifice the non-additivity property after decomposition.
- Out of the above three, only the 3rd decomposition will not generate spurious tuples after join.(and hence has the non-additivity property).
- A test to determine whether a binary decomposition (decomposition into two relations) is nonadditive (lossless) is discussed in section 11.1.4 under Property LJ1. Verify that the third decomposition above meets the property.

Consider the following relation for published books:
BOOK (Book_title, Authorname, Book_type, Listprice, Author_affil, Publisher)
Author_affil referes to the affiliation of the author. Suppose the following dependencies exist:
Book_title -> Publisher, Book_type
Book_type -> Listprice
Author_name -> Author-affil
(a) What normal form is the relation in? Explain your answer.
(b) Apply normalization until you cannot decompose the relations further. State the reasons behind each decomposition.

Answer:
Given the relation
Book(Book_title, Authorname, Book_type, Listprice, Author_affil, Publisher)
and the FDs
Book_title → Publisher, Book_type
Book_type → Listprice
Authorname → Author_affil
(a)The key for this relation is Book_title,Authorname. This relation is in 1NF and not in 2NF as no attributes are FFD on the key. It is also not in 3NF.
(b) 2NF decomposition:
Book0(Book_title, Authorname)
Book1(Book_title, Publisher, Book_type, Listprice)
Book2(Authorname, Author_affil)
This decomposition eliminates the partial dependencies.
3NF decomposition:
Book0(Book_title, Authorname)
Book1-1(Book_title, Publisher, Book_type)
Book1-2(Book_type, Listprice)
Book2(Authorname, Author_affil)
This decomposition eliminates the transitive dependency of Listprice

Consider the universal relation R = {A, B, C, D, E, F, G, H, I} and the set of functional dependencies F = { {A, B} -> {C}, {A} -> {D, E}, {B} -> {F}, {F} -> {G, H}, {D} -> {I, J} }. What is the key for R? Decompose R into 2NF, then 3NF relations.

Answer:

A minimal set of attributes whose closure includes all the attributes in R is a key. (One can also apply algorithm 15.4a (see chapter 15 in the textbook)). Since the closure of {A, B}, {A, B}+ = R, one key of R is {A, B} (in this case, it is the only key).

To normalize R intuitively into 2NF then 3NF, we take the following steps (alternatively, we can apply the algorithms discussed in Chapter 15):

First, identify partial dependencies that violate 2NF. These are attributes that are functionally dependent on either parts of the key, {A} or {B}, alone. We can calculate the closures {A}+ and {B}+ to determine partially dependent attributes:

{A}+ = {A, D, E, I, J}. Hence {A} -> {D, E, I, J} ({A} -> {A} is a trivial dependency)

{B}+ = {B, F, G, H}, hence {A} -> {F, G, H} ({B} -> {B} is a trivial dependency)

To normalize into 2NF, we remove the attributes that are functionally dependent on part of the key (A or B) from R and place them in separate relations R1 and R2, along with the part of the key they depend on (A or B), which are copied into each of these relations but also remains in the original relation, which we call R3 below:

R1 = {A, D, E, I, J}, R2 = {B, F, G, H}, R3 = {A, B, C}

The new keys for R1, R2, R3 are underlined. Next, we look for transitive dependencies in R1, R2, R3. The relation R1 has the transitive dependency {A} -> {D} -> {I, J}, so we remove the transitively dependent attributes {I, J} from R1 into a relation R11 and copy the attribute D they are dependent on into R11. The remaining attributes are kept in a relation R12. Hence, R1 is decomposed into R11 and R12 as follows:

R11 = {D, I, J}, R12 = {A, D, E}

The relation R2 is similarly decomposed into R21 and R22 based on the transitive dependency {B} -> {F} -> {G, H}:

R2 = {F, G, H}, R2 = {B, F}

The final set of relations in 3NF are {R11, R12, R21, R22, R3}

Acknowledgement

Reference for this lecture is

- Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems,* Pearson Education.

The authors and the publishers are gratefully acknowledged.