UNIT 4

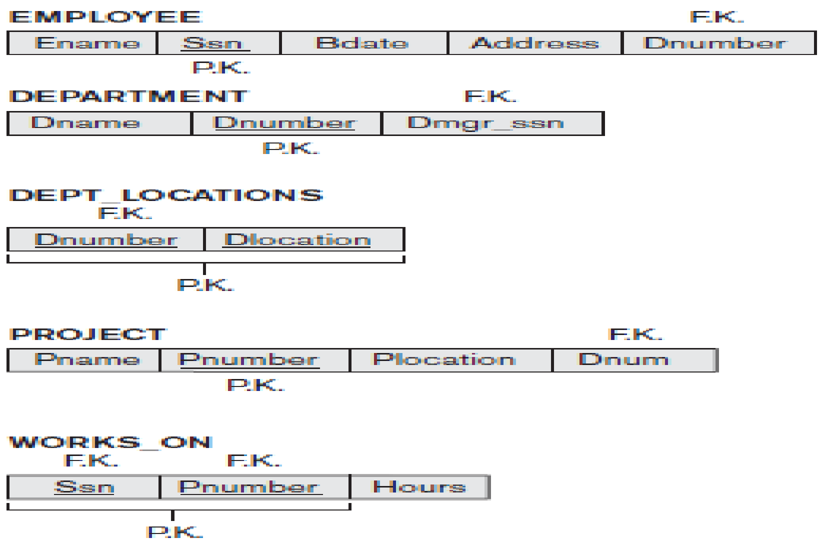
Informal Design Guidelines for Relation Schemas

Before discussing the formal theory of relational database design, we discuss four *informal guidelines* that may be used as *measures to determine the quality* of relation schema design:

- Making sure that the semantics of the attributes is clear in the schema
- Reducing the redundant information in tuples
- Reducing the NULL values in tuples
- Disallowing the possibility of generating spurious tuples

Semantics to Attributes in Relations

The **semantics** of a relation refers to its meaning resulting from the interpretation of attribute values in a tuple. systematically, the relational schema design should have a clear **meaning**.



Guideline 1: Design a relation schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity types and relationship types into a single relation.

if a relation schema corresponds to one entity type or one relationship type, it is straightforward to interpret and to explain its meaning. Otherwise, if the relation corresponds to a mixture of multiple entities and relationships, semantic ambiguities will result and the relation cannot be easily explained.

Redundant Information in Tuples and Update Anomalies

One goal of schema design is to minimize the storage space used by the base relations . Grouping attributes into relation schemas has a significant effect on storage space.
For example, compare the space used by the two base relations EMPLOYEE and DEPARTMENT in Figure with that for an EMP_DEPT base relation in

Figure , which is the result of applying the NATURAL JOIN operation to EMPLOYEE and DEPARTMENT.

In EMP_DEPT, the attribute values pertaining to a particular department (Dnumber, Dname, Dmgr_ssn) are repeated for *every employee who works for that department.* In contrast, each department's information appears only once in the DEPARTMENT relation in Figure

EMPLOYEE

Ename	Ssn	Bdate	Address	Dnumber
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Storing natural joins of base relations leads to an additional problem referred to as **update anomalies**.
These can be classified into insertion anomalies, deletion anomalies, and modification anomalies Anomalies that cause redundant work to be done during insertion into and modification of a relation, and that may cause accidental loss of information during a deletion from a relation

Guideline 2
Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations.
Reducing redundant values in tuples. Save storage space and avoid update anomalies

NULL Values in Tuples
In some schema designs we may group many attributes together into a —fat‖ relation. If many of the attributes do not apply to all tuples in the relation, we end up with many NULLs in those tuples. This can waste space at the storage

level and may also lead to problems with understanding the meaning of the attributes and with specifying JOIN operations at the logical level.

Guideline 3

As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL. If NULLs are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation. Waste of storage space due to NULLs and the difficulty of performing elections, aggregation operations, and joins due to NULL values

Generation of Spurious Tuples

Generation of invalid and spurious data during joins on base relations with matched attributes that may not represent a proper (foreign key, primary key) relationship

Guideline 4

Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantees that no spurious tuples are generated. Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious tuples

Functional Dependencies

A functional dependency is a constraint between two sets of attributes from the database. Suppose that our relational database schema has *n* attributes *A*1, *A*2, ..., *An*;

Definition. A **functional dependency**, denoted by $X \rightarrow Y$, between two sets of attributes *X* and *Y* that are subsets of *R* specifies a *constraint* on the possible tuples that can form a relation state *r* of *R*.

The constraint is that, for any two tuples *t*1 and *t*2 in *r* that have $t1[X] = t2[X]$, they must also have $t1[Y] = t2[Y]$.

This means that the values of the *Y* component of a tuple in *r* depend on, or are *determined by,* the values of the *X* component; alternatively, the values of the *X* component of a tuple uniquely (or **functionally**) *determine* the values of the *Y* component.

There is a functional dependency from *X* to *Y*, or that *Y* is **functionally dependent** on *X*. The abbreviation for functional dependency is **FD** or **f.d.** The set of attributes *X* is called the **left-hand side** of the FD, and *Y* is called the **right-hand side**.

a. Ssn→Ename
b. Pnumber →{Pname, Plocation}
c. {Ssn, Pnumber}→Hours

Inference Rules for Functional Dependencies

An FD $X \rightarrow Y$ is **inferred from** a set of dependencies F specified on R if $X \rightarrow Y$ holds in *every* legal relation state r of R; that is, whenever r satisfies all the dependencies in F, $X \rightarrow Y$ also holds in r. The closure $F+$ of F is the set of all functional dependencies that can be inferred from F.

The following six rules IR1 through IR6 are well-known inference rules for functional dependencies:

IR1 (reflexive rule)1: If X Y, then $X \rightarrow Y$.

IR2 (augmentation rule)2: $\{X \rightarrow Y\} \mid= XZ \rightarrow YZ$.

IR3 (transitive rule): $\{X \rightarrow Y, Y \rightarrow Z\} \mid= X \rightarrow Z$.

IR4 (decomposition, or projective, rule): $\{X \rightarrow YZ\} \mid= X \rightarrow Y$.

IR5 (union, or additive, rule): $\{X \rightarrow Y, X \rightarrow Z\} \mid= X \rightarrow YZ$.

IR6 (pseudotransitive rule): $\{X \rightarrow Y, WY \rightarrow Z\} \mid= WX \rightarrow Z$.

Proof of IR1. Suppose that $X \supseteq Y$ and that two tuples $t1$ and $t2$ exist in some relation
instance r of R such that $t1\ [X] = t2\ [X]$. Then $t1[Y] = t2[Y]$ because $X \supseteq Y$; hence, $X \rightarrow$Y must hold in r.
Proof of IR2 (by contradiction). Assume that $X \rightarrow Y$ holds in a relation instance
r of R but that $XZ \rightarrow YZ$ does not hold. Then there must exist two tuples $t1$ and $t2$ in r such that (1) $t1\ [X] = t2\ [X]$, (2) $t1\ [Y] = t2\ [Y]$, (3) $t1\ [XZ] = t2\ [XZ]$, and (4) $t1\ [YZ] \neq t2\ [YZ]$. This is not possible because from (1) and (3) we deduce (5) $t1\ [Z] = t2\ [Z]$, and from (2) and (5) we deduce (6) $t1\ [YZ] = t2\ [YZ]$, contradicting
(4).
Proof of IR3. Assume that (1) $X \rightarrow Y$ and (2) $Y \rightarrow Z$ both hold in a relation r. Then for any two tuples $t1$ and $t2$ in r such that $t1\ [X] = t2\ [X]$, we must have (3) $t1\ [Y] = t2\ [Y]$, from assumption (1); hence we must also have (4) $t1\ [Z] = t2\ [Z]$ from (3) and assumption (2); thus $X \rightarrow Z$ must hold in r.

Proof of IR4 (Using IR1 through IR3).
1. $X \rightarrow YZ$ (given).
2. $YZ \rightarrow Y$ (using IR1 and knowing that $YZ \supseteq Y$).
3. $X \rightarrow Y$ (using IR3 on 1 and 2).

Proof of IR5 (using IR1 through IR3).
1. $X \rightarrow Y$ (given).
2. $X \rightarrow Z$ (given).
3. $X \rightarrow XY$ (using IR2 on 1 by augmenting with X; notice that $XX = X$).

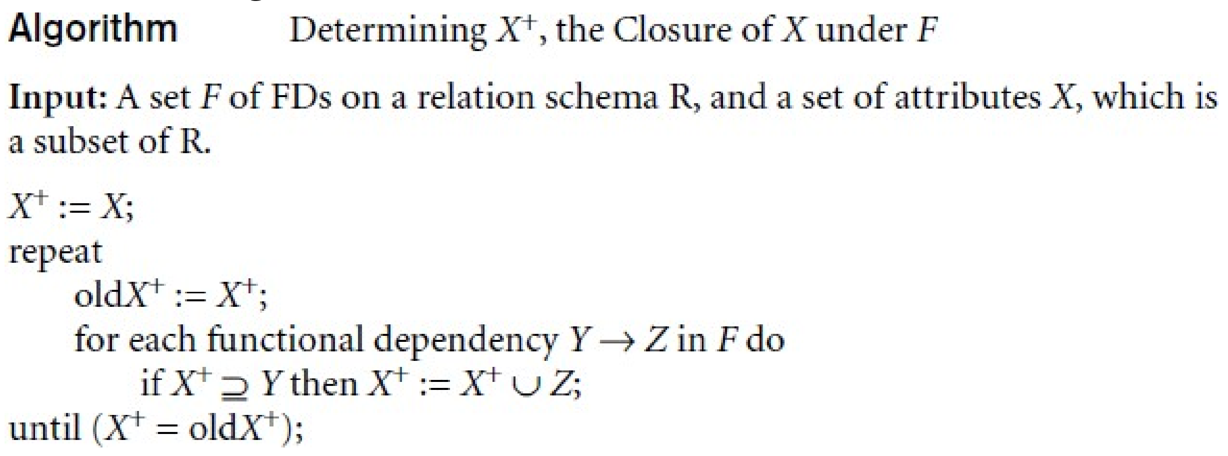
4. $XY \rightarrow YZ$ (using IR2 on 2 by augmenting with Y).
5. $X \rightarrow YZ$ (using IR3 on 3 and 4).

Proof of IR6 (using IR1 through IR3).

1. $X \rightarrow Y$ (given).
2. $WY \rightarrow Z$ (given).
3. $WX \rightarrow WY$ (using IR2 on 1 by augmenting with W).
4. $WX \rightarrow Z$ (using IR3 on 3 and 2).

Algorithm to find Closure

Definition. For each such set of attributes *X*, we determine the set *X*+ of attributes that are functionally determined by *X* based on *F*; *X*+ is called the **closure of X under F**. Algorithm 16.1 can be used to calculate *X*+.



Algorithm Determining X^+, the Closure of X under F

Input: A set F of FDs on a relation schema R, and a set of attributes X, which is a subset of R.

```
$X^+ := X$;
repeat
    old$X^+ := X^+$;
    for each functional dependency $Y \rightarrow Z$ in $F$ do
        if $X^+ \supseteq Y$ then $X^+ := X^+ \cup Z$;
until ($X^+ =$ old$X^+$);
```

Minimal Sets of Functional Dependencies

a **minimal cover** of a set of functional dependencies *E* is a set of functional dependencies *F* that satisfies the property that every dependency in *E* is in the closure *F*+ of *F*.

In addition, this property is lost if any dependency from the set *F* is removed; *F* must have no redundancies in it, and the dependencies in *F* are in a standard form. To satisfy these properties, we can formally define a set of functional dependencies *F* to be **minimal** if it satisfies the following conditions:

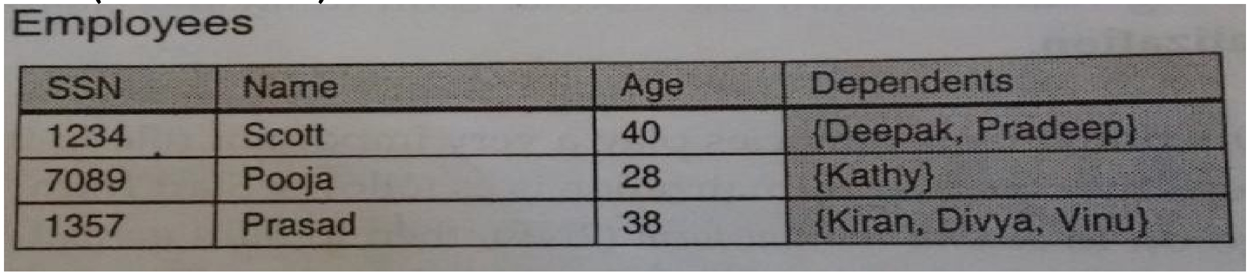
1. Every dependency in *F* has a single attribute for its right-hand side.
2. We cannot replace any dependency $X \rightarrow A$ in *F* with a dependency $Y \rightarrow A$, where *Y* is a proper subset of *X*, and still have a set of dependencies that is equivalent to *F*.
3. We cannot remove any dependency from *F* and still have a set of dependencies that is equivalent to *F*.

Finding a Minimal Cover *F* for a Set of Functional Dependencies *E*
Input: A set of functional dependencies E.

1. Set $F := E$.
2. Replace each functional dependency $X \rightarrow \{A1, A2, \ldots, An\}$ in *F* by the *n* functional dependencies $X \rightarrow A1, X \rightarrow A2, \ldots, X \rightarrow An$.
3. For each functional dependency $X \rightarrow A$ in *F* ,for each attribute *B* that is an element of *X*
if $\{ \{F - \{X \rightarrow A\} \} \cup \{ (X - \{B\}) \rightarrow A\} \}$ is equivalent to *F*, then replace $X \rightarrow A$ with $(X - \{B\}) \rightarrow A$ in *F*.
4. For each remaining functional dependency $X \rightarrow A$ in *F*
if $\{F - \{X \rightarrow A\} \}$ is equivalent to *F*, then remove $X \rightarrow A$ from *F*.

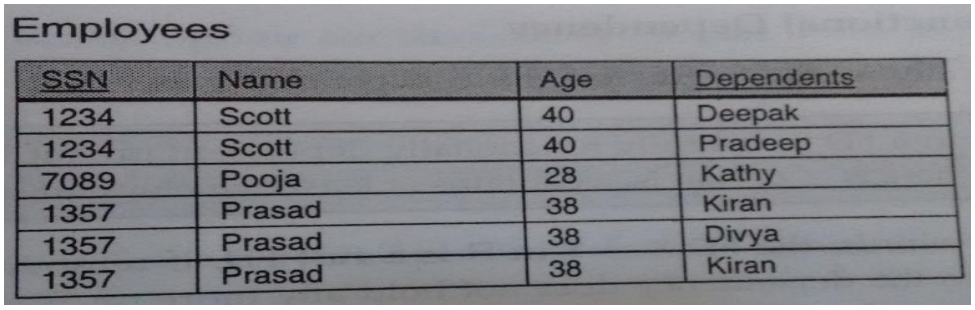
Normalization of Relations
The normalization process, as first proposed by Codd (1972a), takes a relation schema through a series of tests to *certify* whether it satisfies a certain **normal form**.
The process, which proceeds in a top-down fashion by evaluating each relation against the criteria for normal forms and decomposing relations as necessary, can thus be considered as *relational design by analysis.* 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)

First Normal Form
A relation schema R is 1NF if every attribute of R takes only a single value(atomic value)


Employees

SSN	Name	Age	Dependents
1234	Scott	40	{Deepak, Pradeep}
7089	Pooja	28	{Kathy}
1357	Prasad	38	{Kiran, Divya, Vinu}

Employees Table with Multivalue Attribute and not in 1NF

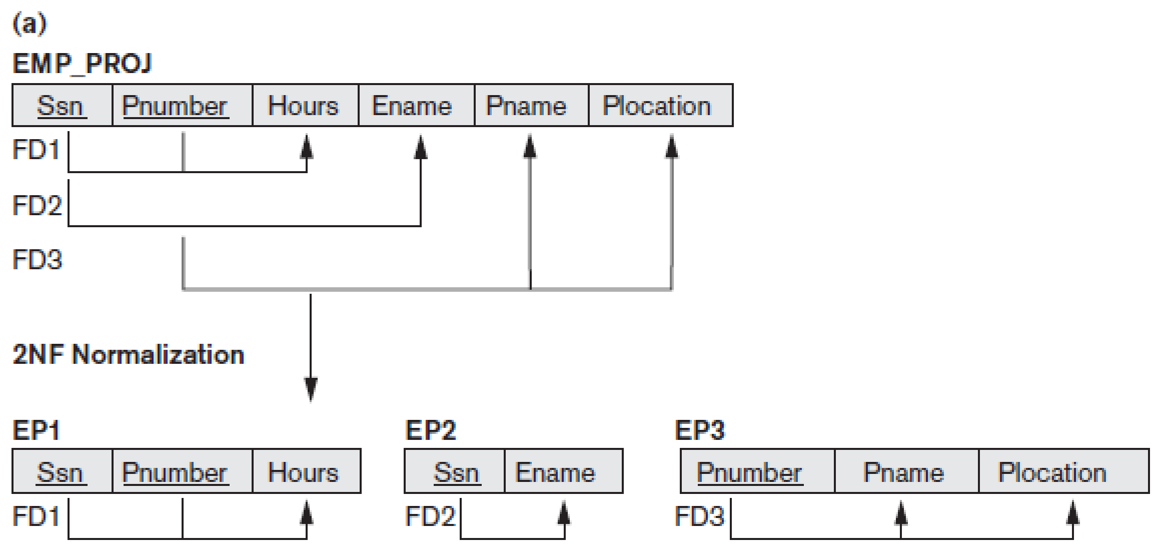

Employees

SSN	Name	Age	Dependents
1234	Scott	40	Deepak
1234	Scott	40	Pradeep
7089	Pooja	28	Kathy
1357	Prasad	38	Kiran
1357	Prasad	38	Divya
1357	Prasad	38	Kiran

Fig: Employees Table in 1NF

Second Normal Form

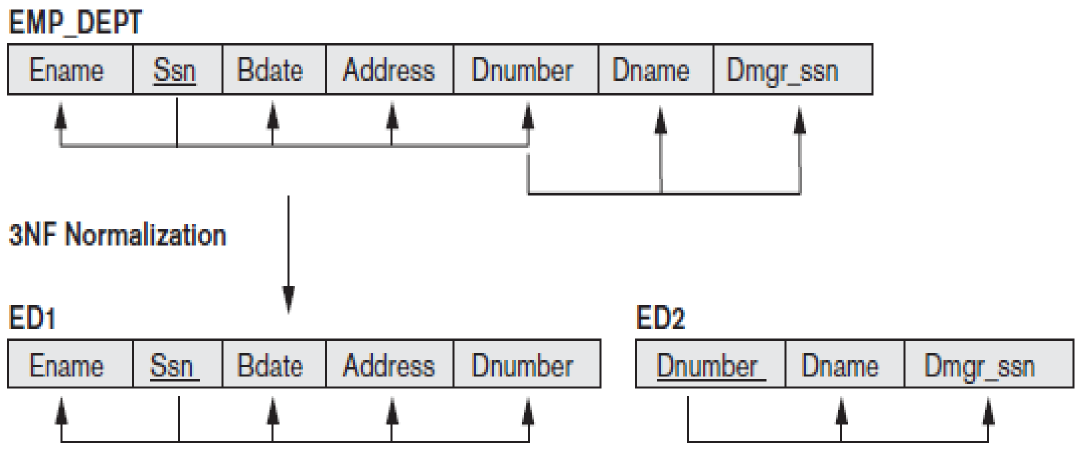
A relation schema *R* is in 2NF if every nonprime attribute *A* in *R* is *fully functionally dependent* on the primary key of *R*.



If a relation schema is not in 2NF, it can be *second normalized* or *2NF normalized* into a number of 2NF relations in which nonprime attributes are associated only with the part of the primary key on which they are fully functionally dependent.
Therefore, the functional dependencies FD1, FD2, and FD3 in Figure (a) lead to the decomposition of EMP_PROJ into the three relation schemas EP1, EP2, and EP3 shown in Figure (b)- each of which is in 2NF.

Third Normal Form

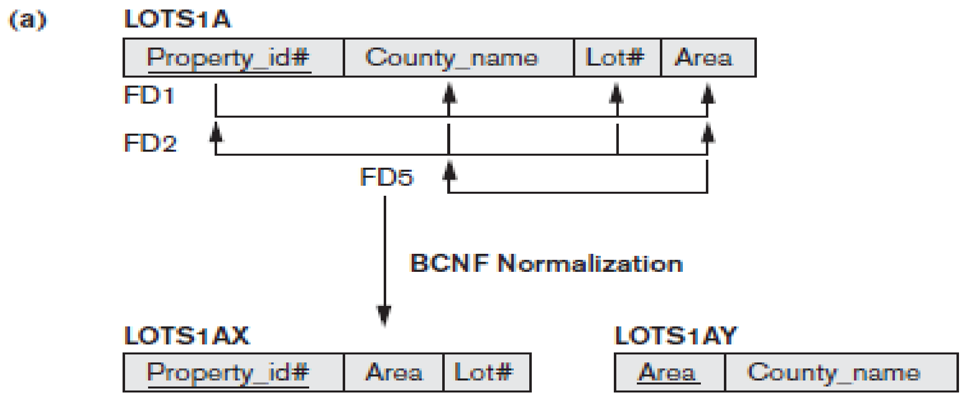
a relation schema *R* is in **3NF** if it satisfies 2NF *and* no nonprime attribute of *R* is transitively dependent on the primary key.

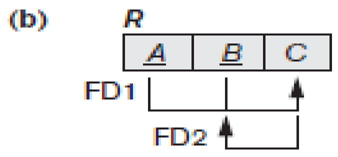


The relation schema EMP_DEPT in Figure (a) is in 2NF, since no partial dependencies on a key exist. However, EMP_DEPT is not in 3NF because of the transitive dependency of Dmgr_ssn (and also Dname) on Ssn via Dnumber.

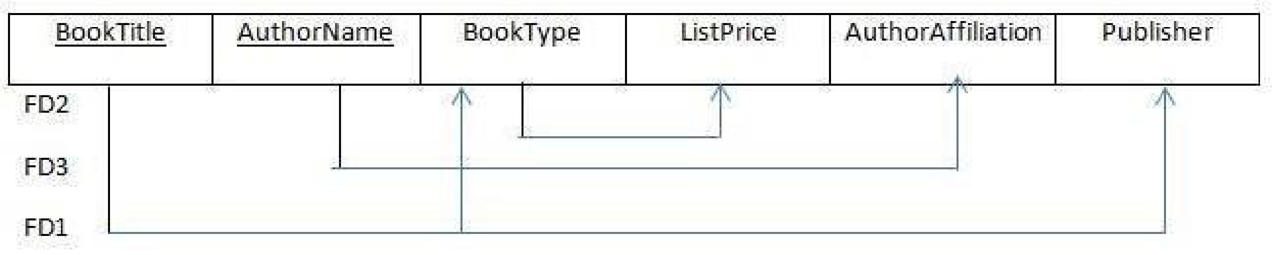
Boyce-Codd Normal Form

Boyce-Codd normal form (BCNF) was proposed as a simpler form of 3NF, but it was found to be stricter than 3NF. That is, every relation in BCNF is also in 3NF; however, a relation in 3NF is *not necessarily* in BCNF.
Definition. A relation schema *R* is in **BCNF** if whenever a *nontrivial* functional dependency *X*→*A* holds in *R*, then *X* is a superkey of *R*.





Problem 1
Consider the following relation for published books:
BOOK(BookTitle, AuthorName, BookType, ListPrice, AuthorAffiliation, Publisher)
Suppose the following dependencies exist:
BookTitle → BookType, Publisher BookType → ListPrice AuthorName → AuthorAffiliation
What normal form is the relation in? explain your answer. Apply normalization until you cannot decompose the relations further. State the reasons behind each decomposition.
Solution:
The relation is in 1NF and not in 2NF as no attributes are fully functionally dependent on the key (BookTitle and AuthorName). It is also not in 3NF.



IT IS NOT IN 2NF because the partial Dependencies exist

{BookTitle,AuthorName} → {Publisher, BookType}

{BookdTitle,AuthorName} → AuthorAffiliation

• Thus, these attributes are not fully functionally dependent on the primary key

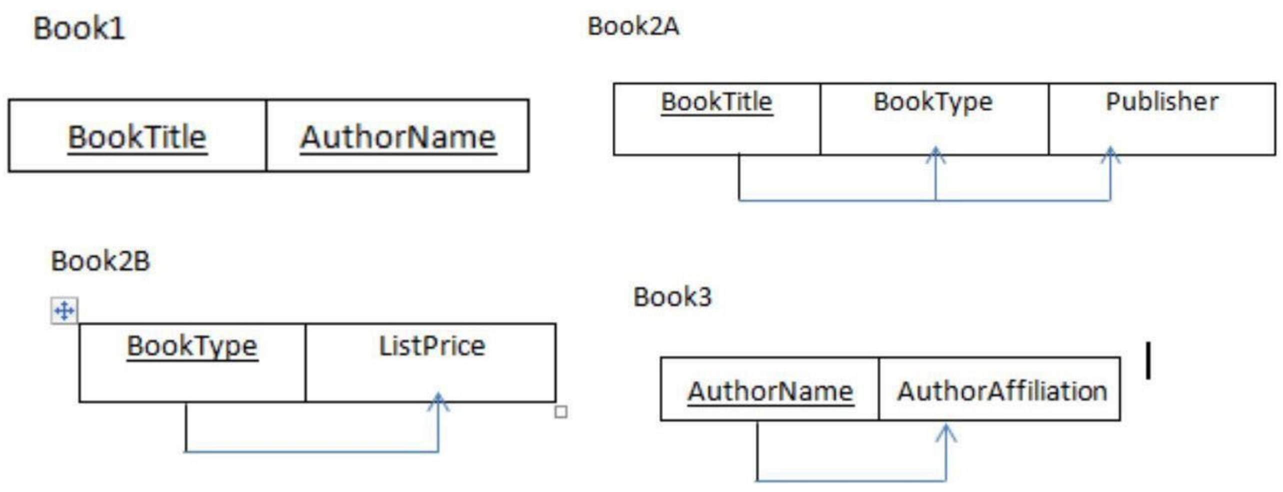
The 2NF decomposition will eliminate the partial dependencies.

2NF decomposition:

Book1(BookTitle, AuthorName)

Book2(BookTitle, BookType, ListPrice, Publisher)

Book3(AuthorName, AuthorAffiliation)



Problem 2

Consider the following relation:

CAR_SALE(Car#, DateSold, Salesman#, Commission%, DiscountAmount)

Assume that a car may be sold by multiple salesmen, and hence

{Car#, Salesman#} is the primary key.

Additional dependencies are:

Car# → DateSold

Car# → DiscountAmount

DateSold → DiscountAmount

Salesman# → Commission%

Based on the given primary key, is the relation in 1NF, 2NF, 3NF?

Why or why not?

How would you successively normalize it completely?

Solution:

The relation is in 1NF because all attribute values are single atomic values.
The relation is not in 2NF because:
Car# → DateSold
Car# → DiscountAmount
Salesman# → Commission%
Thus, these attributes are not fully functionally dependent on the primary key.
2NF decomposition:
CAR_SALE1(Car#, DateSold, DiscountAmount)
CAR_SALE2(Car#, Salesman#)
CAR_SALE3(Salesman#, Commission%)
The relations are not in 3NF because:
Car# → DateSold → DiscountAmount
DateSold is neither a key itself nor a subset of a key and DiscountAmount is not a prime attribute.
3NF decomposition:

- CAR_SALES1A(Car#, DateSold)
- CAR_SALES1B(DateSold, DiscountAmount)
- CAR_SALE2(Car#, Salesman#)
- CAR_SALE3(Salesman#, Commission%)