IE301

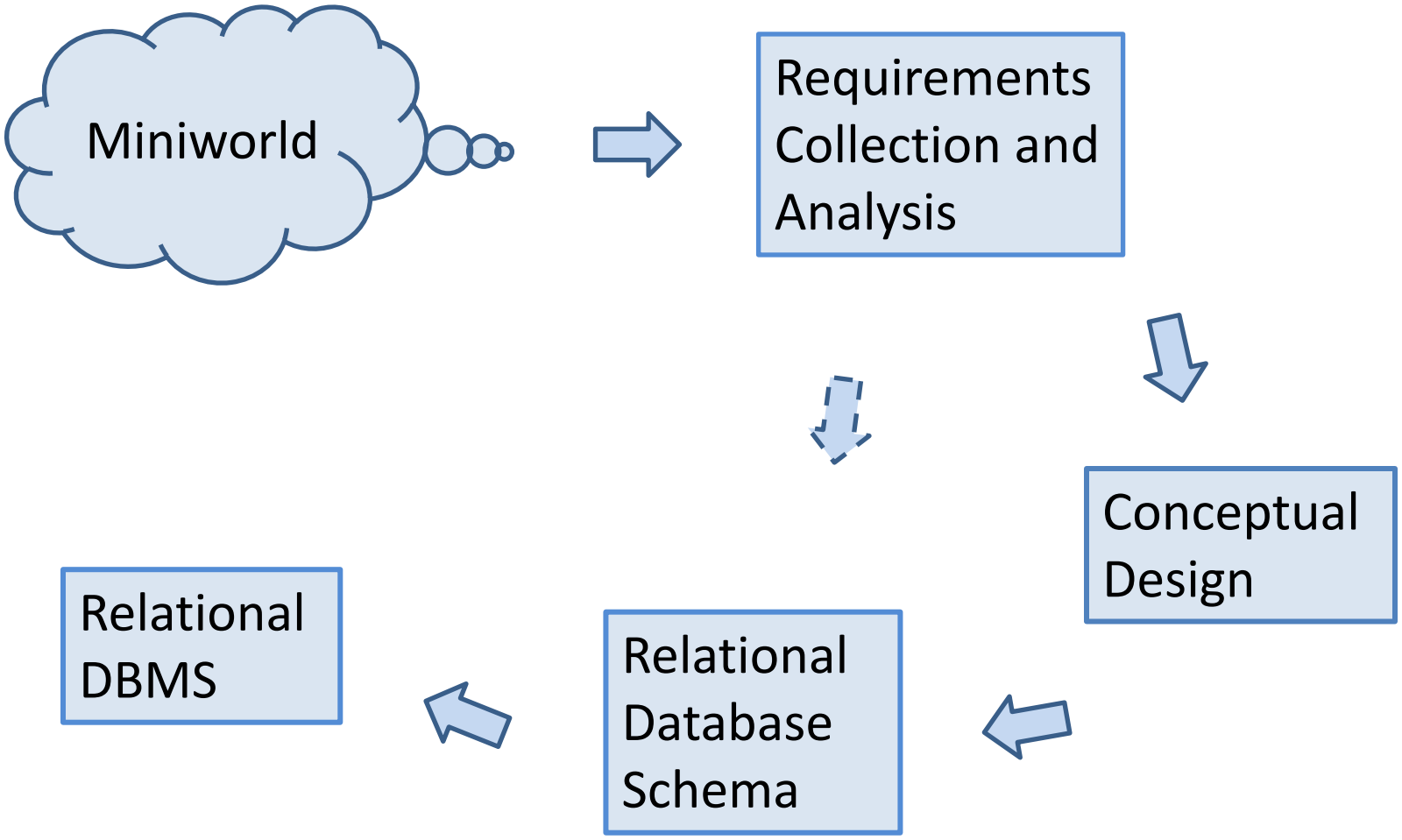
Analysis and Design of Data Systems

Lecture 14

Introduction to Relational Database Design

Aram Keryan

Phases of Database Design



Phases of Database Design

After “Requirements Collection and Analysis” phase a database designer can follow one of two scenarios:

- Start to design EER Model by identifying entity types, relationships and their respective attributes; and then map the conceptual model into relational database schema
- Or, directly start grouping attributes into relations by using common sense

Whichever approach the designer chooses his work will result in having a set of relations forming a relational database schema.

Until now we haven’t established any criteria for *goodness* of design. In other words, we couldn’t evaluate whether one grouping of attributes in relation schemas is better or worse then the other one.

Levels of Goodness of Design

At this point we will discuss the *goodness* of relation schemas at ***logical level*** – how understandable and clear the relation schemas are for the users. Important for correct formulation of queries.

The implicit goals of the design activity are *information preservation* and *minimum redundancy*.

- ✔ ***Information preservation*** implies that during the process of mapping of the conceptual design into relational database schema all the concepts, like entity types, relationships, specializations and other, be preserved.
- ✔ ***Minimum redundancy*** implies minimizing redundant storage of the same information.

Informal Design Guidelines

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 (is not covered during this lecture)

Imparting Clear Semantics to Attributes in Relations

It is assumed that attributes belonging to one relation have certain real-world meaning and a proper interpretation associated with them.

The **semantics** of a relation refers to its meaning resulting from the interpretation of attribute values in a tuple.

If the conceptual design is done carefully and the mapping procedure is followed systematically the relational schema design should have a clear meaning

Example

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

The ease with which the meaning of a relation's attributes can be explained is an *informal measure* of how well the relation is designed.

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

Examples of Violating Guideline 1

EMP_DEPT

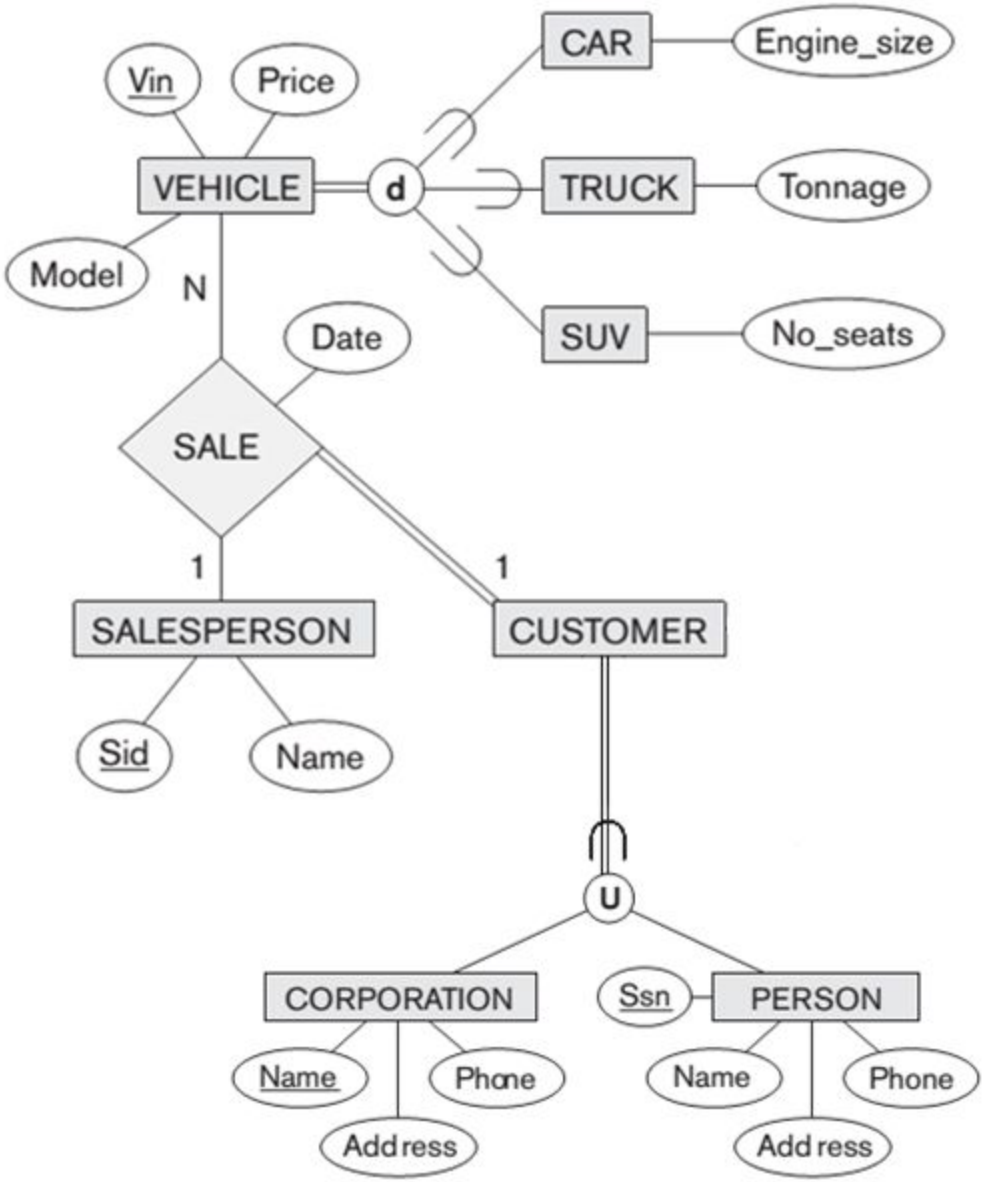
Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

Mixes attributes of employees and departments

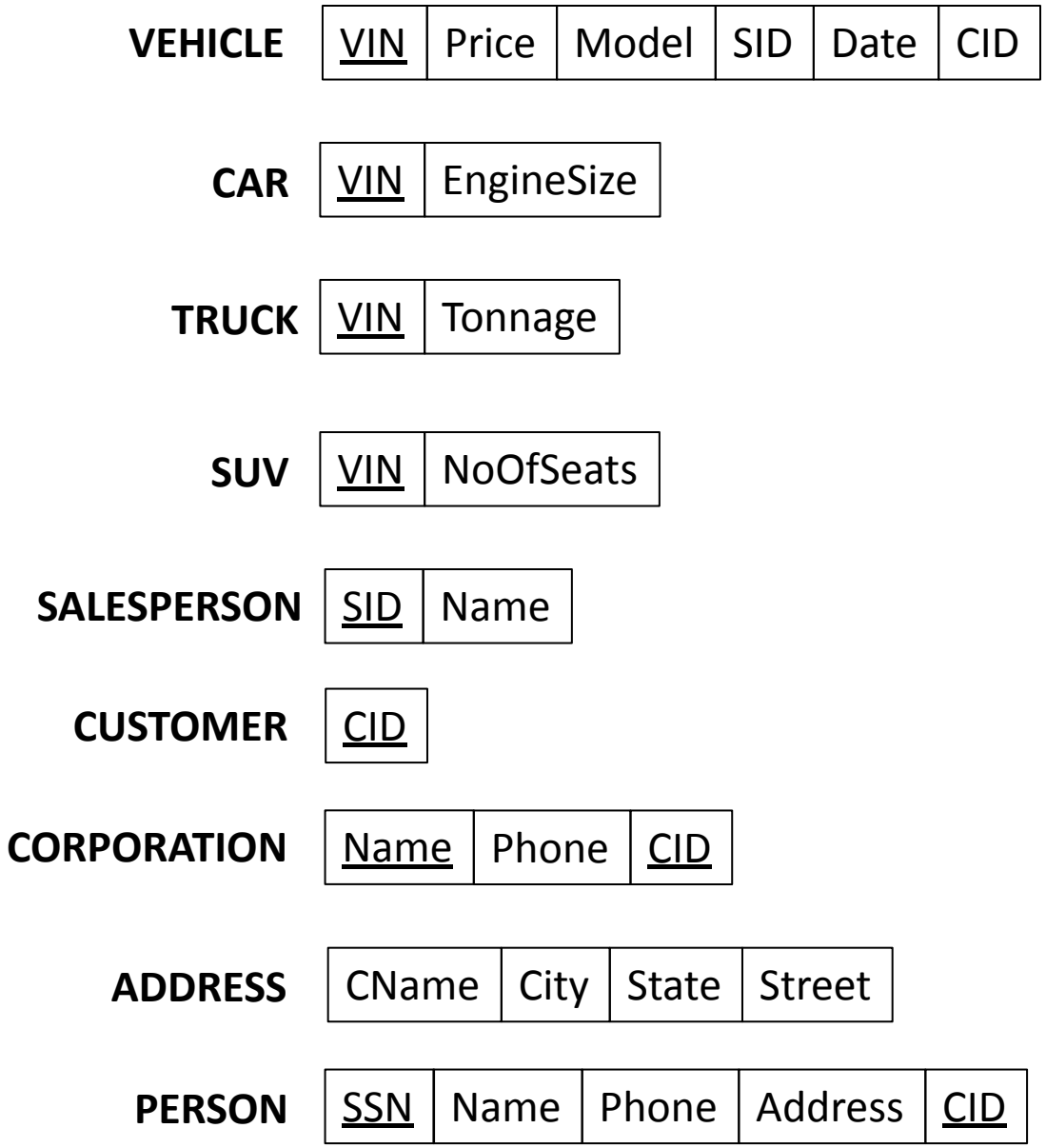
EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation

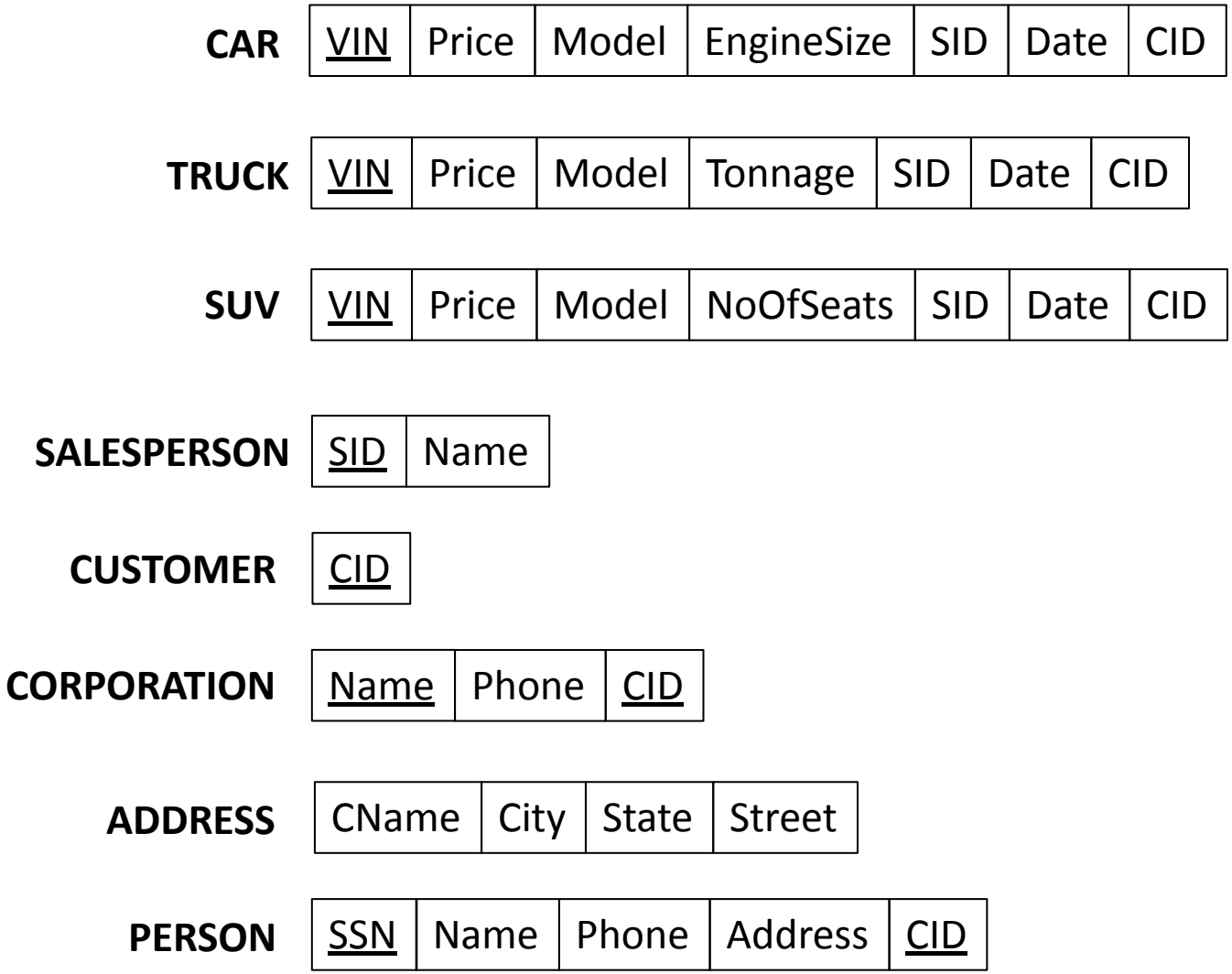
Mixes attributes of employees and projects and the WORKS_ON relationship


CAR
Engine_size
Vin
Price
VEHICLE
d
TRUCK
Tonnage
Model
N
SUV
No_seats
Date
SALE
1
1
SALESPERSON
CUSTOMER
Sid
Name
U
CORPORATION
Ssn
PERSON
Name
Phone
Address
Name
Phone
Address

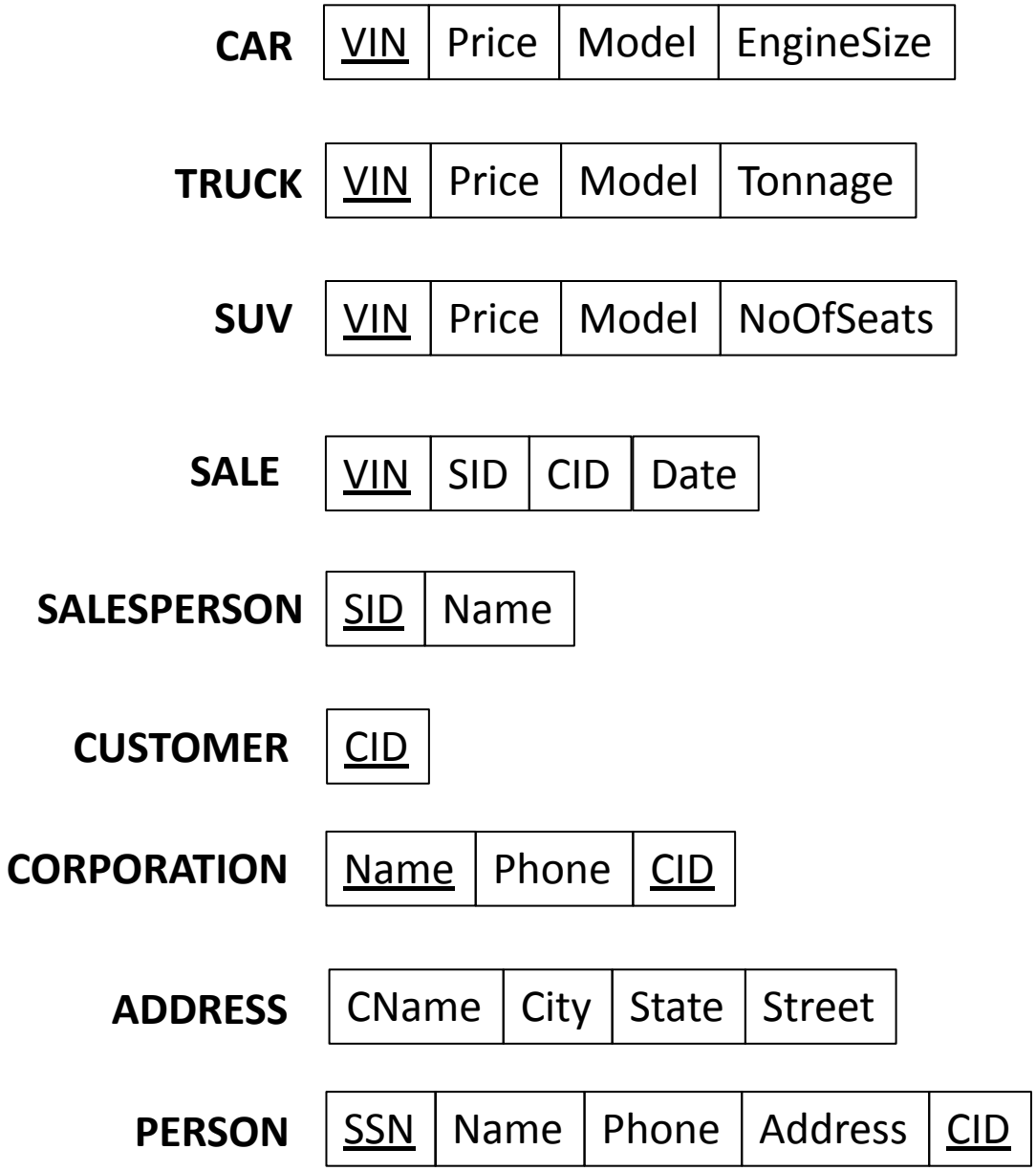
Version 1


VEHICLE
VIN
Price
Model
SID
Date
CID
CAR
VIN
EngineSize
TRUCK
VIN
Tonnage
SUV
VIN
NoOfSeats
SALESPERSON
SID
Name
CUSTOMER
CID
CORPORATION
Name
Phone
CID
ADDRESS
CName
City
State
Street
PERSON
SSN
Name
Phone
Address
CID

Version 2


CAR
VIN
Price
Model
EngineSize
SID
Date
CID
TRUCK
VIN
Price
Model
Tonnage
SID
Date
CID
SUV
VIN
Price
Model
NoOfSeats
SID
Date
CID
SALESPERSON
SID
Name
CUSTOMER
CID
CORPORATION
Name
Phone
CID
ADDRESS
CName
City
State
Street
PERSON
SSN
Name
Phone
Address
CID

Version 3


CAR
VIN
Price
Model
EngineSize
TRUCK
VIN
Price
Model
Tonnage
SUV
VIN
Price
Model
NoOfSeats
SALE
VIN
SID
CID
Date
SALESPERSON
SID
Name
CUSTOMER
CID
CORPORATION
Name
Phone
CID
ADDRESS
CName
City
State
Street
PERSON
SSN
Name
Phone
Address
CID

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.

Redundancy

EMP_DEPT

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

Anomalies

Insertion Anomalies

- To insert a new tuple for an employee who works in department number 5, we must enter all the attribute values of department 5 correctly so that they are *consistent* with the corresponding values for department 5 in other tuples
- It is difficult to insert a new department that has no employees yet

Deletion Anomalies

- If we delete from EMP_DEPT an employee tuple that happens to represent the last employee working for a particular department, the information concerning that department is lost from the database

Modification Anomalies

- if we change the value of one of the attributes of a particular department—say, the manager of department 5—we must update the tuples of *all* employees who work in that department; otherwise, the database will become inconsistent

Guideline 2

Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations

NULL Values in Tuples

Many NULLs waste space at the storage level and may also lead to problems with understanding the meaning of the attributes

Guideline 3

As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL

Example: if only 15 percent of employees have individual offices, there is little justification for including an attribute Office_number in the EMPLOYEE relation; rather, a relation EMP_OFFICES (Essn, Office_number) can be created to include tuples for only the employees with individual offices

Functional Dependency

Let's think of the whole database as being described by a single **universal** relation schema $R = \{A_1, A_2, \ldots, A_n\}$.

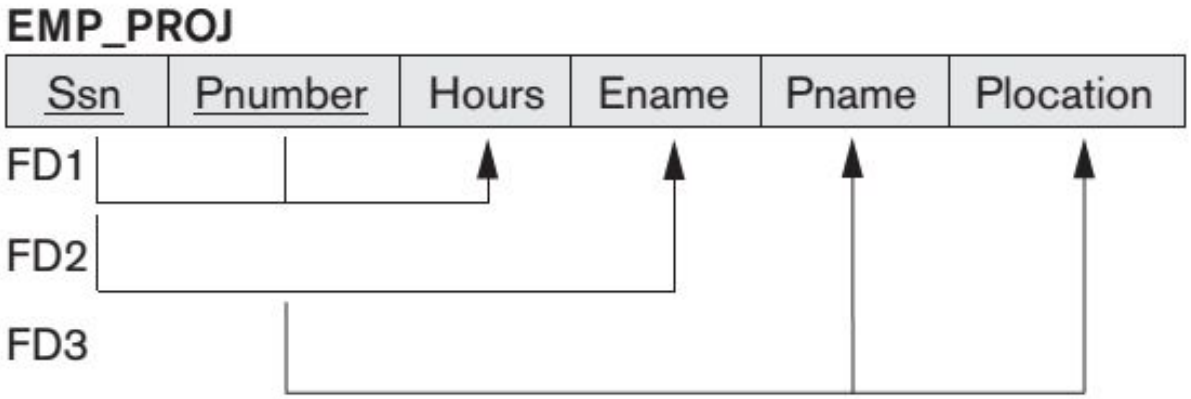
Let's X and Y are subsets of R.

Definition: A functional dependency, denoted $X \rightarrow Y$, on a relation R, specifies a constraint of the form "If two tuples of R agree on all of the attributes of X, then they also agree on all of the attributes of Y".

If $X \rightarrow Y$, then we say that Y is functionally dependent on X, or X functionally determine Y; We also may say that there is a functional dependency from X to Y.

✔ The abbreviation for functional dependency is **FD** or **f.d.**

Example



a. Ssn → Ename

b. Pnumber →{Pname, Plocation}

c. {Ssn, Pnumber} → Hours

Functional Dependency (meaning)

- A functional dependency is a property of the **semantics** or **meaning of the attributes**.

- The database designers will use their understanding of the semantics (meaning) of the attributes of *R*—that is, how they relate to one another—to specify the functional dependencies that should hold on *all* relation states of *R*.

The main use of functional dependencies is to describe further a relation schema *R* by specifying constraints on its attributes that must hold *at all times.*

FD is a property of a Relation

- A functional dependency is a *property of the relation schema R*, not of a particular instance of *R*. Therefore, an FD must be defined explicitly by someone who knows the semantics of the attributes of *R*

- One cannot determine which FDs hold and which do not unless the meaning of and the relationships among the attributes are clearly known and understood

TEACH

Teacher	Course	Text
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

FD is a property of a Relation

- A functional dependency is a *property of the relation schema R*, not of a particular instance of *R*. Therefore, an FD must be defined explicitly by someone who knows the semantics of the attributes of *R*

- One cannot determine which FDs hold and which do not unless the meaning of and the relationships among the attributes are clearly known and understood

TEACH

Teacher	Course	Text
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

Other Properties of FDs

Let's

- $R = \{A_1, A_2, \dots, A_n\}$ is a relation schema
- Let's X and Y are subsets of R
- Let's $X \rightarrow Y$

We can write $X \rightarrow Y$ as $\{X_1, X_2, \dots, X_n\} \rightarrow \{Y_1, Y_2, \dots, Y_m\}$ that is equivalent to a set of FD's:

$$\begin{cases} \{X_1, X_2, \dots, X_n\} \rightarrow Y_1 \\ \{X_1, X_2, \dots, X_n\} \rightarrow Y_2 \\ \cdots \\ \{X_1, X_2, \dots, X_n\} \rightarrow Y_m \end{cases}$$

Normal Forms based of PK's

Motivation

Normalization can be considered a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of :

(1) minimizing redundancy and
(2) minimizing the insertion, deletion, and update anomalies

In other words **Normalization** is a process to make the design have successively better quality.

If relations doesn't meet certain conditions (normal form tests) they are decomposed into 'smaller' relations schemas that meet the tests hence possess desirable properties.