Schema Refinement, Functional Dependencies and Normal Form

EGCI 321: LECTURE 11 (WEEK 6)

Outline

Introduction

- Problems due to Poor Designs

Functional Dependencies

- Logical Implications of FDs
- Attribute Closure

Schema Decomposition

- Lossless-Join Decompositions
- Dependency Preservation

Normal Forms based on FDs

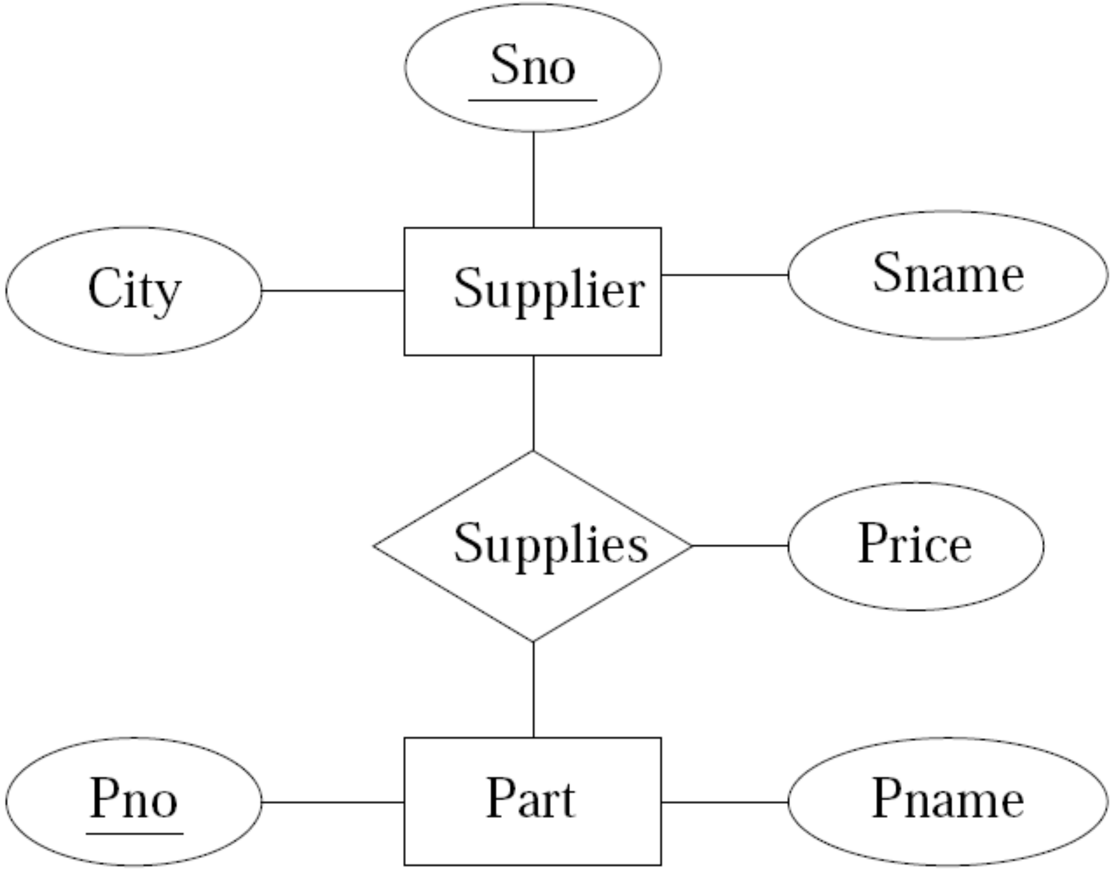
- Boyce-Codd Normal Form
- Third Normal Form

A Parts/Suppliers Database Example

Description of a parts/suppliers database

- Each type of part has a name and an identifying number and may be supplied by zero or more suppliers
  - Each supplier may offer the part at a different price
- Each supplier has an identifying number, a name, and a contact location for ordering parts

Parts/Suppliers Example (cont.)



An E-R diagram for the parts/suppliers database.

Parts/Suppliers Example (cont.)

Suppliers

Sno	Sname	City
S1	Magna	Ajax
S2	Budd	Hull

Parts

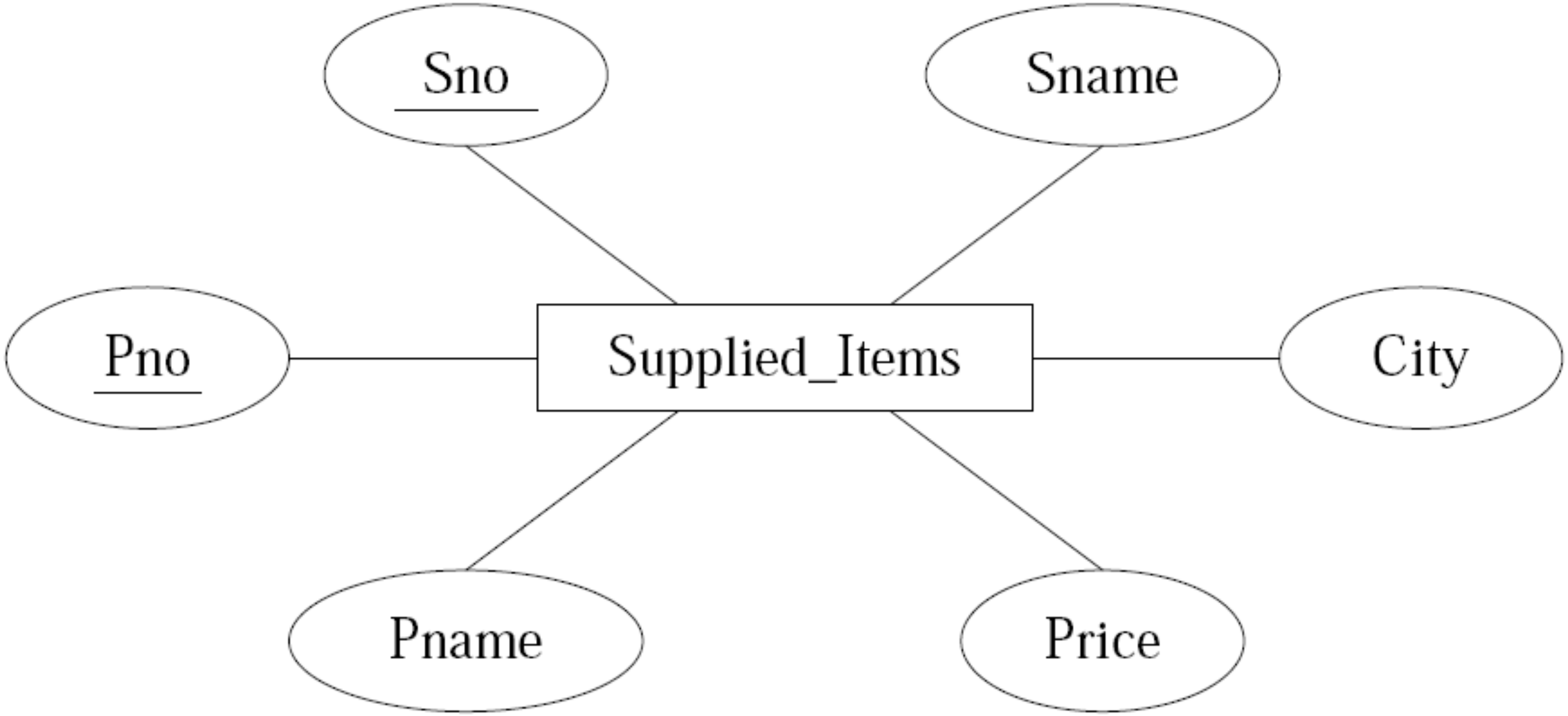
Pno	Pname
P1	Bolt
P2	Nut
P3	Screw

Supplies

Sno	Pno	Price
S1	P1	0.50
S1	P2	0.25
S1	P3	0.30
S2	P3	0.40

An instance of the parts/suppliers database.

Alternative Parts/Suppliers Database



An alternative E-R model for the parts/suppliers database.

Alternative Example (cont.)

Supplied_Items

<u>Sno</u>	Sname	City	<u>Pno</u>	Pname	Price
S1	Magna	Ajax	P1	Bolt	0.50
S1	Magna	Ajax	P2	Nut	0.25
S1	Magna	Ajax	P3	Screw	0.30
S2	Budd	Hull	P3	Screw	0.40

A database instance corresponding to the alternative E-R model.

Change Anomalies

Consider

- Is one schema better than the other?
- What does it mean for a schema to be good?

- The <u>single-table schema</u> suffers from several kinds of problems:
  - Update problems (e.g. changing name of supplier)
  - Insert problems (e.g. add a new item)
  - Delete problems (e.g. Budd no longer suppliers screws)
  - Likely increase in space requirements

- The multi-table schema does not have these problems

Another Alternative Parts/Supplier Database

Is more tables always better?

Snos

Sno
S1
S2

Snames

Sname
Magna
Budd

Cities

City
Ajax
Hull

Inums

Inum
I1
I2
I3

Inames

Iname
Bolt
Nut
Screw

Prices

Price
0.50
0.25
0.30
0.40

Information about relationships is lost!

Designing Good Databases

Goals

- A methodology for evaluating schemas (detecting anomalies)
- A methodology for transforming bad schemas into good schemas (repairing anomalies)

How do we know an anomaly exists?

- Certain types of integrity constraints reveal regularities in database instances that lead to anomalies

What should we do if an anomaly exists?

- Certain schema decompositions can avoid anomalies while retaining all information in the instances

Functional Dependencies (FDs)

Idea: Express the fact that in a relation schema (values of) a set of attributes uniquely **determine** (value of) another set of attributes.

Definition (Functional Dependency)

Let R be a relation schema, and $X, Y \subseteq R$ sets of attributes. The **functional dependency**

$$X \rightarrow Y$$

Holds on R if whenever an instance of R contains two tuples t and u such that $t[X] = u[X]$ then it is also true that $t[Y] = u[Y]$.

We say that *X functionally determines Y (in R).*

Notation: $t[A_1, \ldots A_k]$ means projection of tuple t onto the attributes $A_1, \ldots, A_k$. In other words, $(t.A_1, \ldots, t.A_k)$.

Examples of Functional Dependencies

Consider the following relation schema:

EmpProj

SIN	PNum	Hours	EName	PName	PLoc	Allowance

SIN determines employee name

$SIN \rightarrow Ename$

Project number determines project name and location

$PNum \rightarrow Pname, Ploc$

Allowances are always the same for the same number of hours at the same location

$Ploc, Hours \rightarrow Allowance$

Functional Dependencies and Keys

Keys (as defined previously):

- A **superkey** is a set of attributes such that no two tuples (in an instance) agree on their values for those attributes.
- A **candidate key** is a minimal superkey.
- A **primary key** is a candidate key chosen by the DBA

Relating keys and FDs:

- If $K \subseteq R$ is a **superkey** for relation schema R, then dependency $K \rightarrow R$ holds on R.
- If dependency $K \rightarrow R$ holds on R and we assume that R *does not contain duplicate tuples* *(i.e. relational model)* then $K \subseteq R$ is a **superkey** for relation schema R

Closure of FD Sets

How do we know what additional FDs hold in a schema?

- The closure of the set of functional dependencies F (denoted F^+) is the set of all functional dependencies that are satisfied by every relational instance that satisfies F.
- Informally, F^+ includes all of the dependencies in F, plus any dependencies they imply.

Reasoning About FDs

Logical implications can be derived by using inference rules called **Armstrong's axioms**

- (reflexivity) $X \subseteq Y \;\Rightarrow\; Y \to X^{+}$
- (augmentation) $X \to Y \Rightarrow XZ \to YZ$
- (transitivity) $X \to Y, \quad Y \to Z \;\Rightarrow X \to Z$

The axioms are

- Sound (anything derived from F is in F^{+})
- Complete (anything in F^{+} can be derived)

Additional rules can be derived

- (union) $X \to Y\,, X \to Z \Rightarrow X \to YZ$
- (decomposition) $X \to YZ \Rightarrow X \to Y$

Reasoning About FDs (example)

Example: F = { SIN, PNum $\rightarrow$ Hours

SIN $\rightarrow$ Ename

PNum $\rightarrow$ PName, Ploc

Ploc, Hours $\rightarrow$ Alowance }

A derivation of SIN, PNum $\rightarrow$ Allowance:

1) SIN, PNum $\rightarrow$ Hours ($\in$ F)
2) PNum $\rightarrow$ PName, PLoc ($\in$ F)
3) PLoc, Hours $\rightarrow$ Allowance ($\in$ F)
4) SIN, PNum $\rightarrow$ PNum(reflexivity)
5) SIN, PNum $\rightarrow$ PName, Ploc (transitvity, 4, and 2)
6) SIN, PNum $\rightarrow$ PLoc(decomposition, 5)
7) SIN, PNum $\rightarrow$ PLoc, Hours(union, 6,1)
8) SIN, PNum $\rightarrow$ Alowance (transitivity 7, 3)
9) Pnum $\rightarrow$ Alowance (transitivity 8, 4)

Computing Attribute Closures

There is a more efficient way of using Armstrong's axiom, if we only want to derive the maximal set of attributes functionally determined by some *X* (called the **attribute closure of** *X*)

function $ComputeX^+(X, F)$
begin
    $X^+ := X;$
    while true do
        if there exists $(Y \to Z) \in F$ such that
            (1) $Y \subseteq X^+$, and
            (2) $Z \not\subseteq X^+$
        then $X^+ := X^+ \cup Z$
        else exit;
    return X^+;
end

Computing Attribute Closures (cont.)

- Let R be a relational schema and F a set of functional dependencies on R. Then

Theorem: X is a superkey of R if and only if

$$Compute\ X+\ (X, F) = R$$

Theorem: $X \rightarrow Y \in F^+$ if and only if

$$Y \subseteq ComputeX^+(X, F)$$

Attribute Closure Example

Example: $F = \{$ SIN $\rightarrow$ EName

PNum $\rightarrow$ PName, PLoc

PLoc, Hours $\rightarrow$ Allowance $\}$

ComputeX^+ ({PNum, Hours}, F):

FD	X^+
Initial	PNum, Hours
PNum $\rightarrow$ Pname, Ploc	PNum, Hours, PName, PLoc
PLoc, Hours $\rightarrow$ Alowance	PNum, Hours, PName, PLoc, Allowance

Reference

1. Ramakrishnan R, Gehrke J., Database management systems, 3rd ed., New York (NY): McGraw-Hill, 2003.