3NF and Dependency Preservation

Consider the Relation Contracts(Cid, Sid, Pid, dept, part, qty)

- FD's
  - *Cid -> Cid, Sid, Pid, Dept, Part, Qty.*
  - *Pid, Part -> Cid.*
  - ***Sid, Dept -> Part.***
  - ***Pid -> Sid***
- 3NF
  - ***Pid -> Sid***
  - *R2(Pid, Sid)*
    - Now its in 3NF
  - *Contracts(Cid, Pid, Dept, Part, qty)*
    - In 3NF
    - But lost dependency **Sid, Dept -> Part**

3NF decomposition:
if X -> A holds in R, then either:
a) X is a superkey of R, or
b) A is a prime attribute of R

Keys are
Cid
Pid, Part
Pid, Dept

Dependency Preserving 3NF

- Consider the Relation R(A,B,C,D) with the following FDs
  - *A-> B C D*
  - *B-> D*
  - *C-> D*
  - *3NF Decomposition*
  - *R1(B, D)*
  - *R2(A B C)*
  - *Dependencies are lost ? C->D*

3NF decomposition:
if X -> A holds in R, then either:
a) X is a superkey of R, or
b) A is a prime attribute of R

Key is **A**

Algorithm: Decomposition into 3NF

(with Dependency Preservation &Lossless Join)

Input: A universal relation *R* and a set of functional dependencies *F* on the attributes of *R*.

1. *Find a* ***minimal cover*** *G for F*

2. *For each X of a FD in G, create a relation in D with attributes* $\{X \cup \{A_1\} \cup \{A_2\} \ldots \cup \{A_k\}\}$,

3. ***If none of the relation in D contains a key of R, then create one more relation in D that contains key of R.***

4. *Eliminate Redundant relations from D*

Dependency Preserving 3NF

- Consider the Relation R(A,B,C,D) with the following FDs
  - *A-> B C D*
  - *B-> D*
  - *C-> D*

- 3NF Decomposition using synthesis Algorithm

- Minimal Cover
  - *A-> B C*
  - *B-> D*
  - *C-> D*
  - *A->D is a transitive dependencies so no need to preserve*
- Make a relation for each dependency

Example

- Consider R(ssn, pno, sal, phone, dno, pname, ploc)
- Consider following FDs
  - *FD1: ssn ->sal, phone, dno*
  - *FD2: pno-> pname, ploc*
  - *FD3: {ssn, pno }->sal, ephone, dno, pname, ploc*
- Key: {ssn, pno}
- Step1: minimal cover
  - *G: {ssn->sal,phone,dno ; pno->pname,ploc}*
- Step2:
  - *R1(ssn,sal,phone,dno)*
  - *R2(pno,pname,ploc)*
- **Step3:**
  - ***R3(ssn,pno)***

Algorithm: Decomposition into 3NF

(with Dependency Preservation &Lossless Join)

Input: A universal relation *R* and a set of functional dependencies *F* on the attributes of *R*.

1. *Find a minimal cover G for F*

2. *For each X of a FD in G, create a relation in D with attributes*

   $\{X \cup \{A_1\} \cup \{A_2\} \ldots \cup \{A_k\}\}$,

3. ***If none of the relation in D contains a key of R, then create one more relation in D that contains key of R.***

4. ***Eliminate Redundant relat***

a key
Note we do not have to preserve all keys

3NF and Dependency Preservation

Consider the Relation Contracts(Cid, Sid, Pid, dept, part, qty)

- FD's
  - *Cid -> Cid, Sid, Pid, Dept, Part, Qty.*
  - *Pid, Part -> Cid.*
  - ***Sid, Dept -> Part.***
  - ***Pid -> Sid***

Create a relation for each FD

- R1 (Cid, Pid, Dept, Qty)
- R2 (Pid, Part, Cid)
- R3(Sid, Dept, Part)
- R4(Pid , Sid)

3NF decomposition Rule

if X -> A holds in R, then either:

a) X is a superkey of R, or
b) A is a prime attribute of R

Keys are

Cid

Pid, Part

Pid, Dept

REAL WORLD EXAMPLES

Example: Designing a Student Database

1. Students take courses
2. Students typically take more than one course
3. Students can fail courses and can repeat the same course in different semesters => Students can take the same course more than once.
4. Students are assigned a grade for each course they take.

studentID(sID),
sname,
dept,
advisor,
course(ID),
credit,
semester,
grade,
course-room,
instr,
instr-office

Example: Designing a Student Database

- **Student_course(<u>sID</u>, sname, dept, advisor, <u>course</u>, credit, <u>semester</u>, grade, course-room, instr, instr-office)**
- 1NF: Flat table
- Problems:
  - *Redundancy*
  - *Insert anomalies*
  - *Delete anomalies*
  - *Update problems*

Example: Designing a Student Database

- *Define FDs*
  - **sname, dept, advisor** *are dependent only upon* ***sID***
    - sID -> sname, dept, advisor
  - ***credit*** *dependent only on* ***course*** *and is independent of which* ***semester*** *it is offered and which* ***student*** *is taking it.*
    - Course -> credit
  - **course-room, instructor and instructor-office** *only depend upon the* **course and the semester** *(are independent of which student is taking the course).*
    - Course,Semester -> course-room, instructor, instructor-office
  - *Only* ***grade*** *is dependent upon all 3 parts of the original key.*
    - Stud-Id, Course, Semester -> grade
  - ***Instructor-> instructor-office***

Example: Designing a Student Database

Student_course(sID, sname, dept, advisor, course, credit, semester, grade, course-room, instr, instr-office)

- Student (sID, sname, dept, advisor)
- Student_Reg (sID, course, semester, grade)
- Course (course, credit)
- Course_Offering (course, semester, course-room, instructor)
- Instructor (instructor, instructor-office)

  - *More organized, Less redundancy (save space??)*
  - *Performance problems?? (indirect references)*

Normalization Example -- Sales Order

Sales Order

Fiction Company
202 N. Main
Mahattan, KS 66502

CustomerNumber:	*1001*	Sales Order Number:	405
Customer Name:	*ABC Company*	Sales Order Date:	2/1/2000
Customer Address:	*100 Points*	Clerk Number:	*210*
	Manhattan, KS 66502	Clerk Name:	*Martin Lawrence*

Item Ordered	Description	Quantity	Unit Price	Total
800	*widgit small*	*40*	*60.00*	*2,400.00*
801	*tingimajigger*	*20*	*20.00*	*400.00*
805	*thingibob*	*10*	*100.00*	*1,000.00*

Order Total

Fields will be as follows:

SalesOrderNo, Date,

CustomerNo, CustomerName, CustomerAdd,

ClerkNo, ClerkName,

ItemNo, Description, Qty, UnitPrice

ItemNo -> Description
SalesOrderNo, ItemNo -> Qty, UnitPrice
SalesOrderNo -> Date, CustomerNo, CustomerName, CustomerAdd, ClerkNo, ClerkName
CustomerNo -> CustomerName, CustomerAdd
ClerkNo -> ClerkName

Normalization: First Normal Form

- Separate Repeating Groups into New Tables.
- ***Repeating Groups:*** Fields that may be repeated several times for one document/entity
- The primary key of the new table (repeating group) is always a composite key;

- Relations in 1NF:
  - *<u>SalesOrderNo</u>, <u>ItemNo</u>, Description, Qty, UnitPrice*
  - *<u>SalesOrderNo</u>, Date, CustomerNo, CustomerName, CustomerAdd, ClerkNo, ClerkName*

Normalization: Third Normal Form

A relation R is in **3NF** if X -> A holds in R, then either:
a) X is a superkey of R, or
b) A is a prime attribute of R

- **Relations in 3NF**
  - *Customers: <u>CustomerNo</u>, CustomerName, CustomerAdd*
  - *Clerks: <u>ClerkNo</u>, ClerkName*
  - *Inventory Items: ItemNo, Description*
  - *Sales Orders: <u>SalesOrderNo</u>, Date, CustomerNo, ClerkNo*
  - *SalesOrderDetail: <u>SalesOrderNo</u>, <u>ItemNo</u>, Qty, UnitPrice*

DECOMPOSITION REVISITED

Properties of Relational Decomposition

Attribute preservation condition:

- *Each attribute in R will appear in at least one relation schema R_i in the decomposition*

Dependency Preservation Property

- It is not necessary that the exact dependencies specified in *F* appear themselves in individual relations of the decomposition *D*.
- It is sufficient that the union of the dependencies that hold on the individual relations in *D* be equivalent to *F*.

Lossless (Non-additive) Join Property

- *lossless refers to loss of information, not to loss of tuples.*
- *In fact, for "loss of information" a better term is "addition of spurious information"*

Testing Binary Decompositions for Lossless Join Property

- **Binary Decomposition:** decomposition of a relation *R* into two relations.
- **Lossless join test for binary decompositions:**
  - *A decomposition $D = \{R_1, R_2\}$ of R has the lossless join property with respect to a set of functional dependencies F on R if and only if either*
    - **FD $((R_1 \cap R_2) \rightarrow (R_1 - R_2))$ is in F^+, or**
    - **FD $((R_1 \cap R_2) \rightarrow (R_2 - R_1))$ is in F^+.**

In other words,
the decomposition is lossless if the set of attributes that are used to join R_1 and R_2 i.e. $(\mathbf{R_1 \cap R_2})$ should be key either in R_1 or R_2

Example

A universal relation ***R(SSN, Ename, Pnumber, Pname, Plocation, Hours),***

with FDs

SSN-> Ename

Pnumber -> Pname, Plocation

SSN, Pnumber -> Hours

is decomposed into

R1 (Ename, Pnumber)

R2(SSN, Pnumber, Pname, Plocation, Hours)

Is it lossy or lossless ?

Activity: Decompositions Lossless or Lossy ?

A universal relation ***R(SSN, Ename, Pnumber, Pname, Plocation, Hours),***

with FDs

SSN-> Ename

Pnumber -> Pname, Plocation

SSN, Pnumber -> Hours

is decomposed into

EMP

SSN	ENAME

PROJECT

PNUMBER	PNAME	PLOCATION

WORKS_ON

SSN	PNUMBER	HOURS

Is it lossy or lossless ?

Decompositions

- Determine whether each decomposition has
  - *The dependency preservation property, and*
  - *the lossless join property, with respect to F.*
- Also determine the normal form of each relation in the decomposition

R = {A, B, C, D, E, F, G, H, I, J}

F = { {A, B}→{C},
{A}→{D, E},
{B}→{F},
{F}→{G,H},
{D}→{I, J} }.

- **D1** = {R1, R2, R3, R4, R5} ;
  - *R1 = {A, B, C} , R2 = {A, D, E} , R3 = {B, F} , R4 = {F, G, H} , R5 = {D, I, J}*
- **D2** = {R1, R2, R3} ;
  - *R1 = {A, B, C, D, E} , R2 = {B, F, G, H} , R3 = {D, I, J}*
- **D3** = {R1, R2, R3, R4, R5} ;
  - *R1 = {A, B, C, D} , R2 = {D, E} , R3 = {B, F} , R4 = {F, G, H} , R5 = {D, I, J}*