Normalization Theory

Normalization

- **Good relation schema design**
  - Goodness of relation schema can be done by (at)

  a) Logical level

  b) Implementation level

Normalization-

Taking relations through normal forms.

Making a good schema design.

Refining the database

Informal Design Guidelines For Relation Schemas

1) **Guidelines 1:**

Design a relation schema that do not combine attributes from multiple entity types & relationship types.

Ex.Emp_dept

(a) ssn, ename, add, dno, dname,mgrssn

Ex. Emp_proj

(b) ssn, pno, hours, name, pname, plocation

2) Redundant Information in Tuples and Update anomalies

- when two attributes are mixed.

Update anomalies

i) Insertion – emp,dept

ii) deletion – last emp of dept

iii) Modification – mgr ssn all tuples.

Ex. Emp_dept

ssn	Name	age	dno	dname	mgrssn
101	Kumar	25	1	Site	Null
102	Ram	28	1	Site	101
103	John	23	2	Scse	104
104	Siva	29	2	Scse	Null
105	Nikhil	30	3	Smbs	104

Guidelines 2:

Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations. If any anomalies are present, note them clearly and make sure that the programs that update the database will operate correctly.

- Tables are put together to reduce storage space.
- create views for base relations with joins for easy querying.

3) Null values in tuples

- Waste storage
- Join operations at logical level

Guidelines 3:

Avoid placing attributes in a base relation whose values may frequently be null.

4) Generation of Spurious Tuples

- Ex. Emp_loc

Ename	Ploc
Ram	P1
Kumar	P2
Mukesh	P1

- Ex. Emp_Proj

SSN	Pno	Hrs	Pname	Ploc
11	44	10	PA	P1
12	45	15	PB	P2
13	46	14	PC	P1

SSN	Pno	Pname	Ename	Ploc
11	44	PA	Ram	P1
12	45	PB	Kumar	P2
13	46	PC	Ram	P1

Guidelines 4 :

- Design relation schemas so that they can be joined with equality conditions on attributes that are either PRIMARY KEYS or FOREIGN KEYS.
- Do not have common attributes that are either PK or FK.
- If such relations are unavoidable do not join them, because it produces spurious tuples.

Normalization

- Relational Db design by analysis
- Taking relations through normal forms.
- Normalization of data

    decomposing relations to minimize redundancy and update anomalies.

Properties of Normalization

- Loss less join/ Non additive join property

Decomposed relation doesn't give spurious tuples.

- Dependency preservation

    each functional dependency is there in some decomposed relation.

Functional Dependencies

- Functional dependency is a constraint between 2 sets of attributes from the database.
- Functional dependency is a property of the semantics of the attributes.
- Database designers specify the semantics by Functional dependency.

FD x->y

x determines y

t1[x]=t2[x] => t1[y]=t2[y]

x, y are set of attributes.

Inference Rules

Armstrong's axioms are a set of inference rules used to infer all the functional dependencies on a relational database. They were developed by William W. Armstrong.

Reflexivity: if Y is a subset of X, then X determines Y

if $Y \subseteq X$, $X \rightarrow Y$

Augmentation: also known as a partial dependency, says if X determines Y, then XZ determines YZ for any Z

if $X \rightarrow Y$, $XZ \rightarrow YZ$

Transitivity: X determines Y, and Y determines Z, then X must also determine Z

if $X \rightarrow Y$ & $Y \rightarrow Z$, $X \rightarrow Z$

Union: X determines Y and X determines Z then X must also determine Y and Z

if $X \rightarrow Y$ & $X \rightarrow Z$, $X \rightarrow YZ$

Decomposition: if X determines Y and Z, then X determines Y and X determines Z separately

Closure property and Key

Let S be set of FD
For every Si in S take the LHS attribute and find the closure
If a attribute determines all the attributes in the relation it is the PK
If none of LHS attribute determine all, check for combination of LHS attributes

Closure:
$\{X\}^+$ of $X \rightarrow Y$ is add LHS and RHS attributes, add the attributes that are determined by added attributes. Continue till no determination

•Algorithm:

Determining X^+, the Closure of X under F

$X^+ := X;$

repeat

    oldX+ := X^+;

    for each functional dependency Y -> Z in F do

    If $Y \subseteq X^+$ then $X^+ := X^+ \cup Z$;

until ($X^+ = oldX^+$),

Example

Emp_dep (Eno,Ename,Add, Dno, Dname, Dloc)

FD={ Eno → Ename ; Eno → Add; Eno → Dno;
Dno → Dname, Dloc)

Closure:

$\{Eno\}^+ =$

Eno,
Ename,
Add
Dno
Dname, Dloc

Prime or Non Prime attribute

- An attribute of relation R is Prime attribute if member of some key(ck or pk).

- An attribute is called non prime if it is not a Prime attribute.

First Normal Form (1NF)

Attributes should have atomic values.

- ORDBMS are removed.
- Eg. Department

R(dno, dname, mgrssn, dloc)

INF

D1(dno, dloc)

D2(Dno,dname,mgrssn)

Second Normal Form (2NF)

- R is in 2NF if every non-prime attribute A in R is fully functionally dependent on Primary Key.

    (i.e) there should not be partial dependency on Primary Key.

Emp_Proj(SSn,pno,ename,pname,ploc,hrs)

Ssn -> ename
Pno -> pname,ploc
Ssn, pno -> hrs

2NF

E(ssn,ename) P(pno,pname,ploc) EP(ssn,pno,hrs)

Third Normal Form (3NF)

- R is in 3NF if no non-prime attribute A in R is transitively dependent on Primary Key.

Emp_Dept(SSn,ename,dno,dname,dloc,mgrssn)

Ssn -> ename,dno
dno -> dname,dloc,mgrssn

3NF

E(ssn,ename,dno)   D(dno,dname,dloc,mgrssn)

Emp_Proj (SSn,pno,ename,pname,ploc,hrs)

Ssn -> ename
Pno -> pname,ploc
Ssn, pno -> hrs

$\{ssn\}^+$ = ssn, ename

$\{pno\}^+$ = pno, pname,ploc

$\{ssn,pno\}^+$ = ssn, pno, hrs, ename, pname, ploc

Algorithms

- Decomposition till 3NF with Dependency preserving

- Given : R base relation & set of FD

1.Find a minimal cover G for F

2. For each left-hand-side X of a functional dependency that appears in G, create a relation schema in D with attributes {X U {A1 } U {A2 }... U {Ak },
where X -> A1 , X -> A2 , ... , X -> Ak are Functional dependencies. Make X as c.k

3. Place any remaining attributes in a relation.

Algorithms

- Decomposition till 3NF with loss less join property

- Follow previous algorithm, If key of the base relation is not there in decomposed create a new relation that contains the key

BOYCE-CODD NORMAL FORM(BCNF)

Every determinant in a relation must be a candidate key.

FD X->Y X is a determinant; can be single or composite attribute

Eg. Teaching (Student, Course, Instructor)

FD1 : student, course -> Instructor
FD2 : Instructor -> Course

Boyce-codd normal form(BCNF)

Relational decomposition into BCNF with nonadditive join property

1. For the relation R that not in BCNF

    take X -> Y that violates BCNF

    create 2 relations (R – Y) & (X U Y)

student, course -> Instructor

Instructor -> Course

key ? S,C

FD that violates I → C

(R – Y) & (X U Y) (S,I) (I,C)

Algorithms

PROPERTIES OF RELATIONAL DECOMPOSITIONS

a) Attribute preservation

b) Dependency preservation only till 3NF

c) Lossless join (Nonadditive)

D = {R1 ,R2 ,….Rm} of R has lossless property with respect to F on R if every relation state r of R that satisfies F holds ΠR1 (r) * ΠRm (r) =r

Minimal set of dependencies

A set of functional dependencies F is said to be minimal:

1) Every dependency in F has a single attribute for its RHS
2) We cannot replace any dependency $X \rightarrow A$ with $Y \rightarrow A$ 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 equal to F

Algorithm for minimal cover: Given – Set of dependency E

1. Set F:= E
2. Replace X→ A1,A2,…. By X→A1, X→ A2 …
3. X→ A in F , for each attribute B element of X if { F- (X→A)} U { X-(B) → A } is equivalent to F , then replace X → A with {X-(B) → A} in F
4. For remaining FD X→ A in F , if F- { X→ A} is equal to F then remove X → A

Example 1: F ={ B→ A, D→ A, AB → D }

Step 1:

write FD with one attribute on RHS

Its already in the canonical form

B→ A, D→ A, AB → D

Step 2:

FD with more than one LHS attribute : AB → D

check whether can be replaced by A→ D or B → D

Take B→ A

BB→ A B	(by augumentation)
B → A B	
AB → D	(third FD)
B → D	(by transitivity)

B→ A, D→ A, B → D

Continuation…

Step 3:

check whether any FD can be derived from other

$B \rightarrow A$, $D \rightarrow A$, $B \rightarrow D$

$B \rightarrow D$, $D \rightarrow A \Rightarrow B \rightarrow A$

eliminate $B \rightarrow A$

Final : $D \rightarrow A$, $B \rightarrow D$

Hence minimal cover $\{B \rightarrow D , D \rightarrow A \}$

Example 2: G ={ A→ BCDE, CD → E}

Step 1:

A→ B
A →C
A → D
A → E
CD → E

Step 2:

CD→ E , Replace by { D→ E or C → E}
No replacement

Step 3:

A→ CD (by union) , CD → E => A → E (by trans)
remove A → E

Minimal F = { A→ BCD, CD → E}

Example 3: G ={ A→ C, AC → D, E → AD, E → H }

Step 1:

A→ C
AC → D
E → A
E → D
E → H

Step 2:

AC → D
Replace by A→D or C→ D
A → C
AA → AC
A→ AC
AC → D
A → D
AC→ D can be replaced by A→ D

Step 3:

$A \rightarrow C$

$A \rightarrow D$

$E \rightarrow A$

$E \rightarrow D$

$E \rightarrow H$

No further elimination can be done.

$A \rightarrow CD$, $E \rightarrow A,D,H$ (by union)

minimal

Consider the following relations for an order-processing application database at ABC,Inc.

ORDER (<u>O#,</u> Odate, Cust#, Total_amount)

O# → Odate, Cust#, Total_amount

{ o#} + = o#, odate, cust#, total_amount

satisfies INF
satisfies 2NF
Satisfies 3NF

Order R is in 3NF

Consider the following relations for an order-processing application database at ABC,Inc.

ORDER_ITEM(O#, I#, Qty_ordered, Total_price, Discount%)

O#,I# → total_price

I# → Discount%, Qty_ordered

{ O#,I#} + = O#,I#, total_price, discount%,qty_ordered

{I#} + = I#, discount%,qty_ordered

O#,I# is the key

Satisfies 1NF

Not 2NF (partial dependency on Key)

The relation is in INF

D1(O#,I#,total_price) ↓ D2 (I#,discount%, qty_ordered)

Satisfies 2NF

Satisfies 3NF

Example

Book (book_title, author_name, book_type , listprice, author_affiliation ,publisher)

book_title → publisher, book_type

book_type → listprice

author_name → author_affiliation

- Key ? book_title, author_name
- Till 3NF

  D1(book_title,publisher,book_type)

  D2(book_type,listprice)

  D3(author_name,author_affiliation)

  D4(book_title,author_name)

Algorithms

Checking lossless join decomposition

a) Create a matrix , row= no of decomposed, col=no of attributes in base

b) Fill all the rows with b

c) Taking rach row for a decomposed relation fill with a if it is present in the decomposed

d) For a FD x→ y if there is a ‘a’ in the x column make the y column as a

e) If one row contains all ‘a’ then the decomposition is lossless.

Example from (2NF)

Emp_Proj(<u>SSn,pno</u>,ename,pname,ploc,hrs)

Ssn -> ename
Pno -> pname,ploc
Ssn, pno -> hrs

E(ssn,ename)
P(pno,pname,ploc)
EP(ssn,pno,hrs)

Emp_Proj(<u>SSn,pno</u>,ename,pname,ploc,hrs)

SSN	PNO	Ename	Pname	Ploc	Hrs

Emp_Proj(<u>SSn,pno</u>,ename,pname,ploc,hrs)

SSN	PNO	Ename	Pname	Ploc	Hrs
B	B	B	B	B	B
B	B	B	B	B	B
B	B	B	B	B	B

E(ssn,ename)
P(pno,pname,ploc)
EP(ssn,pno,hrs)

SSN	PNO	Ename	Pname	Ploc	Hrs
A	B	A	B	B	B
B	B	B	B	B	B
B	B	B	B	B	B

E(ssn,ename)
P(pno,pname,ploc)
EP(ssn,pno,hrs)

SSN	PNO	Ename	Pname	Ploc	Hrs
A	B	A	B	B	B
B	**A**	B	**A**	**A**	B
B	B	B	B	B	B

E(ssn,ename)
P(pno,pname,ploc)
EP(ssn,pno,hrs)

SSN	PNO	Ename	Pname	Ploc	Hrs
A	B	A	B	B	B
B	A	B	A	A	B
A	**A**	B	B	B	**A**

SSN	PNO	Ename	Pname	Ploc	Hrs
A	B	A	B	B	B
B	A	B	A	A	B
A	A	A	B	B	A

SSN	PNO	Ename	Pname	Ploc	Hrs
A	B	A	B	B	B
B	A	B	A	A	B
A	A	A	A	A	A

SSN	PNO	Ename	Pname	Ploc	Hrs
A	B	A	B	B	B
B	A	B	A	A	B
A	A	A	A	A	A

SSN	PNO	Ename	Pname	Ploc	Hrs
A	B	A	B	B	B
B	A	B	A	A	B
A	A	A	A	A	A

Loss less join property is present

4NF

A relation is in 4NF , if every non trival dependency $X \rightarrow\rightarrow Y$ has X as super key

Trivial MVD: $X \rightarrow\rightarrow Y$, Y subset of X, or X union Y is the Relation

EMP(ename, pname, dname)

Ename $\rightarrow\rightarrow$ pname   ename $\rightarrow\rightarrow$ dname

Non trivial, X is not superkey

4NF

D1(ename, pname)   D2(ename, dname)

5NF (Join Dependency)

A relation schema is in 5NF or PJNF (Project Join Normal Form) with respect to set of dependencies , if for every non trivial join dependency JD(R1,R2,R3…) in F+ , every Ri is the super key of R.

Supply(supplier, project, part)
Supplier→→ project,part

X U Y => R

5NF

S	P	T
S1	P1	T1
S1	P1	T2
S1	P2	T1
S1	P3	T1

S	P
S1	P1
S1	P2
S1	P3

S	T
S1	T1
S1	T2

JOIN

S	P	T
S1	P1	T1
S1	P1	T2
S1	P2	T1
S1	P2	T2
S1	P3	T1
S1	P3	T2

S	P
S1	P1
S1	P2
S1	P3

JOIN

P	T
P1	T1
P1	T2
P2	T1
P3	T1

JOIN

S	T
S1	T1
S1	T2

S	P	T
S1	P1	T1
S1	P1	T2
S1	P2	T1
S1	P3	T1

Summary - Normalization

- Definition , properties
- Informal design guidelines
- Normal forms – INF,2NF,3NF,BCNF, 4NF,5NF
- Algorithms – minimal cover, till 3NF, BCNF, check lossless join property
- HOT: Given relations a) normalize till 3NF or BCNF
  b) check lossless join property
  c) Find minimal cover
  d) check update anomalies