Relational Database Design

Pno.	Projtitle	Rollno	Name	Class	mks
10	Tourism	RM2007022	Kunal	SE	25
		RM2007025	Tirth		
		RF2007023	Pooja		
30	Hospital	RM2006011	Yazad	TE	50
		RM2006014	Sahil		
20	Content	RF2005073	Pooja	BE	100
		RM2007022	Kunal	SE	
40	RTO				

Each project has a set of students

Proj no.	Projtitle	Rollno	Stud name	Class	mks
10	tourism	RM2007022	Kunal	SE	25
10	tourism	RM2007025	Tirth	SE	25
10	toursm	RF2007023	Pooja	SE	25
30	Hospital mgmt	RM2006011	Yazad	TE	50
30	Hospital mgmt	RM2006014	Sahil	TE	50
20	Content mgmt	RF200573	Pooja	BE	100
		RM2007022	Kunal	SE	
40	RTO				

Redundancy may cause spelling mistake and thus data inconsistency

Proj no.	Projtitle	Rollno	Stud name	Class	mks
10	tourism	RM2007022	Kunal	SE	25
10	tourism	RM2007025	Tirth	SE	25
10	toursm	RF2007023	Pooja	SE	25
30	Hospital mgmt	RM2006011	Yazad	TE	50
30	Hospital mgmt	RM2006014	Sahil	TE	50
		RF2005073	Pooja	BE	100
		RM200722	Kunal	SE	
25	RTO				

Redundant data creates difficulty in searching

Proj no.	Projtitle	Rollno	Stud name	Class	mks
10	tourism	RM2007022	Kunal	SE	25
10	tourism	RM2007025	Tirth	SE	25
10	tourism	RF2007023	**Pooja Manwani**	SE	25
30	Hospital mgmt	RM2006011	Yazad	TE	50
30	Hospital mgmt	RM2006014	Sahil	TE	50
		RF2005073	**Pooja Mehta**	BE	100
		RM2007022	Kunal	SE	
25	RTO				

Primary key is not defined, hence data can repeat

Proj no.	Projtitle	Rollno	Stud name	Class	mks
10	tourism	RM2007022	Kunal	SE	25
10	tourism	RM2007025	Tirth	SE	25
10	tourism	RF2007023	Pooja	SE	25
30	Hospital mgmt	RM2006011	Yazad	TE	50
30	Hospital mgmt	RM2006014	Sahil	TE	50
		RF2005073	Pooja	BE	100
		RM2007022	Kunal	SE	
25	RTO				

Primary key is not defined, hence data can be null

Projno.	Projtitle	Rollno	Stud name	Class	mks
10	tourism	RM2007022	Kunal	SE	25
10	tourism	RM2007025	Tirth	SE	25
10	tourism	RF2007023	Pooja	SE	25
30	Hospital mgmt	RM2006011	Yazad	TE	50
30	Hospital mgmt	RM2006014	Sahil	TE	50
NULL	NULL	RF2005073	Pooja	BE	100
NULL	NULL	RM2007022	Kunal	SE	Null
25	RTO	NULL	NULL	NUll	Null

Insertion anomaly

Proj no.	Projtitle	Rollno	Stud name	Class	mks
10	tourism	RM2007022	Kunal	SE	25
10	tourism	RM2007025	Tirth	SE	25
10	tourism	RF2007023	Pooja	SE	25
30	Hospital mgmt	RM2006011	Yazad	TE	50
30	Hospital mgmt	RM2006014	Sahil	TE	50
Null	Null	RF2005073	Pooja	BE	100
Null	Null	RM2007022	Kunal	SE	Null
25	RTO	Null	Null	Null	Null

Deletion anomaly

Proj no.	Projtitle	Rollno	Stud name	Class	mks
10	tourism	RM2007022	Kunal	SE	25
10	tourism	RM2007025	Tirth	SE	25
10	tourism	RF2007023	Pooja	SE	25
30	Hospital mgmt	RM2006011	Yazad	TE	50
30	Hospital mgmt	RM2006014	Sahil	TE	50
		RF2005073	Pooja	BE	100
		RM2007022	Kunal	SE	
25	RTO				

Update anomaly

Proj no.	Projtitle	Rollno	Stud name	Clas s	mks
10	tourism	RM2007022	Kunal	SE	25
10	tourism	RM2007025	Tirth	SE	25
10	tourism	RF2007023	Pooja	SE	25
30	Hospital admn	RM2006011	Yazad	TE	50
30	Hospital admn	RM2006014	Sahil	TE	50
		RF2005073	Pooja	BE	100
		RM2007022	Kunal	SE	
25	RTO				

Dependency

Proj no.	Projtitle	Rollno	Stud name	Class	mks
10	tourism	RM2007022	Kunal	SE	25
10	tourism	RM2007025	Tirth	SE	25
10	tourism	RF2007023	Pooja	SE	25
30	Hospital mgmt	RM2006011	Yazad	TE	50
30	Hospital mgmt	RM2006014	Sahil	TE	50
		RF2005073	Pooja	BE	100
		RM2007022	Kunal	SE	
25	RTO				

Normalization & Normal form

- **Normalization**

It is a tool to validate and improve a logical design, so that it satisfies certain conditions that avoid redundancy and anomalies like insert anomaly, delete anomaly, update anomaly.

It is based on the analysis of functional dependencies.

- **Normal form**

It is a state of relation that results by applying simple rules regarding functional dependencies (or relationships between attributes) to that relation.

Types of Normal forms

- First normal form
- Second normal form
- Third normal form
- Boyce-Codd normal form
- Fourth normal form
- Fifth normal form

Functional Dependency

- A kind of Unique-value constraint.
- Knowledge of these constraints vital to eliminate redundancy in database.
- An FD on a relation R is of the form X → Y

  (X functionally determines Y) where X, Y are sets of attributes of R, such that whenever two tuples in R have same values on all the attributes of *X* , they must have same values on all the attributes of Y .
- Given X → Y where Y is the set of all attributes of R then X is a key for R.

Example FDs

- Consider the following relational schema

  student_t(st_id, name, birth_date, gender, hostel_no, room_no, sem, year)

Example FDs

st_id	name	birth_date	gender	hostel_no	room_no	sem	year
2009A7PS001	arun	01/12/1991	m	BH-1	1	1	2011-12
2009A7PS001	arun	01/12/1991	m	BH-1	1	2	2011-12
2009A7PS001	arun	01/12/1991	m	BH-3	10	1	2010-11
2009A7PS001	arun	01/12/1991	m	BH-4	1	2	2010-11
2009A7PS011	arti	11/10/1991	f	CH-4	5	1	2010-11
2009A7PS011	arti	11/10/1991	f	CH-4	1	2	2010-11
2009A7PS011	arti	11/10/1991	f	CH-5	5	1	2011-12

Example FDs

Then following FDs hold on 'student_t'

st_id → name

st_id → birth_date

st_id → gender

or

st_id → name birth_date gender

But st_id → hostel_no, st_id → room_no,

st_id → sem and st_id → year

do not hold on 'student_t'

Example FDs

- Consider the following relational schema

  course_t(course_no, title, units, ltp, sem, year)

Example FDs

Course_no	title	units	ltp	sem	year
CS C210	Operating Systems	3	300	1	2010-11
CS C210	Operating Systems	3	300	2	2010-11
CS C210	Operating Systems	3	300	1	2011-12
CS C352	Database Systems	3	300	2	2010-11
CS C352	Database Systems	3	300	1	2011-12
CS C352	Database Systems	3	300	2	2011-12

Example FDs

Then following FD's hold on 'course_t'

course_no → title

course_no → units

course_no → ltp

or

course_no → title units ltp

But course_no → sem and

course_no → year

do not hold on 'course_t'

Example FDs

- Consider the following relational schema

  st_marks_t(st_id, st_name, course_no, course_title, section_no, test_no, marks)

Example FDs

Then following FD's hold on 'st_marks_t'

st_id → st_name
course_no → course_title
st_id course_no → section_no, test_no, marks

But st_id → course_title and
course_no → st_name
do not hold on 'st_marks_t'

Note: Is the above one correct? If not what is the error?

Example FDs

The following is the correct one.

The following FDs hold on ‘st_marks_t’

- st_id → st_name
- course_no → course_title
- st_id course_no → section_no
- st_id course_no , test_no → marks

Example FDs

- Consider the following relational schema

  Employee_t(emp_id, emp_name, dept_id_w, dep_name_w, dept_id_h)

Example FDs

emp_id	emp_name	dept_id_w	dept_name_w	dept_id_h
245	Bharat	cs	comp. science	cs
245	Bharat	math	mathematics	cs
300	Karan	cs	comp. science	
301	Hari	cs	comp. science	math
301	Hari	math	mathematics	math
311	Bharat	math	mathematics	

emp_id → emp_name dept_id_h

dept_id_w → dept_name_w

First normal form (1NF): Atomic values and define primary keys

Proj no.	Projtitle	Rollno	fname	Phone no	Class	mks
10	tourism	RM2007022	Kunal	111	SE	25
10	tourism	RM2007022	Kunal	222	SE	25
10	tourism	RM2007025	Tirth	333	SE	25
10	tourism	RF2007023	Pooja	444	SE	25
30	Hospital mgmt	RM2006011	Yazad	555	TE	50
30	Hospital mgmt	RM2006014	Sahil	666	TE	50
		RF2005073	Pooja	777	BE	100
		RM2007022	Kunal	111	SE	

First normal form (1NF): Atomic values and define primary keys

Rollno	**Phone no**
RM2007022	111
RM2007022	222
RM2007025	333
RF2007023	444
RM2006011	555
RM2006014	666
RF2005073	777
~~RM2007022~~	~~111~~

Second normal form (2NF): Functional dependency Project table

Projno.	Projtitle
10	tourism
30	Hospital mgmt
25	RTO

Second normal form (2NF): Functional dependency

Student table

<u>Rollno</u>	fname	Class
RM2007022	Kunal	SE
RM2007025	Tirth	SE
RF2007023	Pooja	SE
RM2006011	Yazad	TE
RM2006014	Sahil	TE
RF2005073	Pooja	BE
~~RM2007022~~	~~Kunal~~	~~SE~~

Second normal form (2NF): **Functional dependency**

Student-Project table

Projno.	**Rollno**	mks
10	RM2007022	25
10	RM2007025	25
10	RF2007023	25
30	RM2006011	50
30	RM2006014	50

Third Normal form(3NF): Transitive Dependency

<u>Rollno</u>	fname	Class	Marks
RM2007022	Kunal	SE	25
RM2007025	Tirth	SE	25
RF2007023	Pooja	SE	25
RM2006011	Yazad	TE	50
RM2006014	Sahil	TE	50
RF2005073	Pooja	BE	100

Third Normal form(3NF): Transitive Dependency

Rollno	Class
RM2007022	SE
RM2007025	SE
RF2007023	SE
RM2006011	TE
RM2006014	TE
RF2005073	BE

Class	Marks
SE	25
TE	50
BE	100

Requirements: 1NF

- Each table has a primary key: minimal set of attributes which can uniquely identify a record

- The values in each column of a table are atomic (No multi-value attributes allowed).

- All attributes are dependent on primary key

Requirements 2NF

A table is in second normal form if

- Its in first normal form
- It includes no partial dependencies (where an attribute is dependent on only a part of a primary key)

Definition 3NF

- Its in second normal form
- It contains no transitive dependencies (where a non-key attribute is dependent on another non-key attribute)
- If A→B→C then decompose the table into two relations having columns (A,B) and (B,C).

Q. Design the following schema into best normal form

Order table (

- invoice no.,
- Date,
- Customer no.
- Customer name,
- Customer address,
- Customer city,
- Customer state,
- Item id
- Item description,
- Item quantity,
- Item price,
- Item total,
- Order total price)

Answer

- Orders(<u>invoiceno</u>, customer no, order date,order total price)
- Customers(<u>customerno</u>, custname, address, custcity, custstate)
- Items(<u>itemid,</u> itemdescr, itemprice)
- orderlist(<u>invoiceno, itemid</u>)

Attribute closure

- Suppose {A1,A2,….,An} is a set of attributes and S is a set of FD's. The closure of {A1,A2,….,An} under the FD's in S is the set of all attributes B such that every relation that satisfies all the FD's in set S also satisfies A1,A2,….,An → B. i.e., A1,A2,….,An follows from the FD's of S. It is denoted as A1A2…An by {A1,A2,…,An}+.

- Starting with the given set of attributes, we repeatedly expand the set by adding the right sides of FD's as soon as we have included their left sides. Eventually, we can not expand the set any more and the resulting set is the closure.

Attribute closure (cont.)

- **The following steps explain in more detail about the above:**

  step1. Let X be a set of attributes that eventually will become the closure. First initialize X to {A1,A2,…,An}.

  step2. Search for some FD B1B2…Bm → C such that all of B1,B2,…,Bm are in the set of attributes X, but not C. Then add C to the set X.

  step3. Repeat step 2 as many times as necessary until no more attributes can be added to X. Since X can only grow and number of attributes of any relation schema are finite, eventually nothing can be added to X.

  step4. The set X, after no more attributes can be added to it is the correct value of {A1,A2,…,An}+.

Attribute closure

- Let F be a set of FD's holding on a relation R. Let X, Y be sets of attributes of R. Then Y is said to be attribute closure of X, denoted by $X^+ = Y$, if $X \rightarrow Y$ 'follows' from F
- Algorithm

  Ex: Find attribute closure of {A,B} w.r.t the FD's:

  $$AB \rightarrow C,\ BC \rightarrow AD,\ D \rightarrow E$$

  – Useful to check whether a given FD follows from given set of FD's: example $AB \rightarrow D$, $D \rightarrow A$
  – Useful to check whether a given set of attributes forms key w.r.t the FD's
  – Useful to find all FD's that hold on a relation

Closure Test

- An easier way to test is to compute the *closure* of Y, denoted Y^+.
- Basis: $Y^+ = Y$.
- Induction: Look for an FD's left side X that is a subset of the current Y^+. If the FD is $X \rightarrow A$, add A to Y^+.

Attribute closure(example)

- The closure of {A,B} is {A,B}+.
- Let X = {A,B}
- These two attributes are on the left side of FD, AB → C are in X, so add C,D to X, i.e., X= {A,B,C,D}.
- In FD BC → E, the left side attributes B C form the subset of X, so add E to X, i.e., X={A,B,C,D,E}.
- In FD BE → F, the left side attributes B E form subset of X, so add F to X, i.e., x={A,B,C,D,E,F}.
- The FD AH → J, can not be used, because the left side attributes A H do not form subset of X.
- Finally {A,B}+ = {A,B,C,D,E,F}.

Example schema

sid	name	serno	subj	cid	exp-grade
1	Sam	570103	AI	520	B
23	Nitin	550103	DB	550	A
45	Jill	505103	OS	505	A
1	Sam	505103	OS	505	C

Rules of FD's

- Assume W, X, Y, Z are sets of attributes of R.
  - If $Y \subseteq X$ then $X \rightarrow Y$ (reflexivity)

    name, sid → name
  - If $X \rightarrow Y$ then $ZX \rightarrow ZY$ (augmentation)

    serno → subj so serno, exp-grade → subj, exp-grade
  - If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$ (transitivity)

    serno → cid and cid → subj

    so serno → subj
  - If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$ (Union)
  - If $X \rightarrow Y$ and $Z \subseteq Y$

    then $X \rightarrow Y$ and $X \rightarrow Z$ (decomposition)
  - If $W \rightarrow X$ and $XY \rightarrow Z$ then $WY \rightarrow Z$ (pseudotransitivity)
  - If $XY \rightarrow ZY$ then $XY \rightarrow Z$
- First three are known as Armstrong's Axioms

Rules of FD's (cont.)

- Reflexivity:

  If {B1,B2,….,Bj} $\subseteq$ {A1,A2,….,Ai} then

  A1A2….Ai → B1B2…Bj. These are called trivial FD's.

- Augmentation:

  If A1A2….Ai → B1B2…Bj then

  A1,A2,….,AiC1C2…Ck → B1B2…BjC1C2…Ck

  for any set of attributes C1,C2,…,Ck.

- Transitivity:

  If A1A2….Ai → B1B2…Bj and B1B2…Bj → C1C2…Ck then A1A2….Ai → C1C2…Ck.

Rules of FD's (cont.)

- Union:
- If $A_1A_2....A_i \rightarrow B_1B_2...B_j$ and

  $A_1A_2....A_i \rightarrow C_1C_2...C_k$ then

  $A_1A_2....A_i \rightarrow B_1B_2...B_jC_1C_2...C_k$
- Decomposition:
- If $A_1A_2....A_i \rightarrow B_1B_2...B_jC_1C_2...C_k$ and

  $A_1A_2....A_i \rightarrow B_1B_2...B_j$ then

  $A_1A_2....A_i \rightarrow C_1C_2...C_k$

Rules of FD's (cont.)

- If W → X and XY → Z then WY → Z

  If D1D2…Dl → A1A2….Ai and

  A1A2…Ai B1B2…Bj → C1C2…Ck then

  D1D2…Dl B1B2…Bj → C1C2…Ck.

- If XY → ZY then XY → Z

  If A1A2…Ai B1B2…Bj→C1C2…CkB1B2…Bj

  then A1A2….Ai B1B2…Bj → C1C2…Ck.

Closure of a Set of FD's

Defn. Let *F* be a set of FD's.

Its *closure*, F^+, is the set of all FD's:

{X → Y | X → Y is derivable from *F* by Armstrong's Axioms}

Which of the following are in the closure of our Student-Course FD's?

name → name

cid → subj

serno → subj

cid, sid → subj

cid → sid

sid, serno → subj, exp-grade

Equivalence of FD sets

Defn. Two sets of FD's, F and G, are *equivalent* if their closures are equivalent, $F^+ = G^+$

e.g., these two sets are equivalent:

$\{XY \rightarrow Z, X \rightarrow Y\}$ and

$\{X \rightarrow Z, X \rightarrow Y\}$

- F^+ contains a huge number of FD's (exponential in the size of the schema)
- Would like to have smallest "representative" FD set

Minimal Cover (Canonical Cover)

Defn. A FD set F is *minimal* if:

1. Every FD in F is of the form $X \rightarrow A$, where A is a single attribute
2. For no $X \rightarrow A$ in F is: $F - \{X \rightarrow A\}$ equivalent to F
3. For no $X \rightarrow A$ in F and $Z \subset X$ is: $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ equivalent to F

we express each FD in simplest form

in a sense, each FD is "essential" to the cover

Defn. F is a *minimum cover* for G if F is minimal and is equivalent to G.

e.g.,

$\{X \rightarrow Z, X \rightarrow Y\}$ is a minimal cover for

$\{XY \rightarrow Z, X \rightarrow Z, X \rightarrow Y\}$

More on Closures

If F is a set of FD's and $X \rightarrow Y \notin F^+$
then for some attribute $A \in Y$, $X \rightarrow A \notin F^+$

Proof by counterexample.

Assume otherwise and let $Y = \{A_1, \ldots, A_n\}$
Since we assume $X \rightarrow A_1, \ldots, X \rightarrow A_n$ are in F^+
then $X \rightarrow A_1 \ldots A_n$ is in F^+ by union rule,
hence, $X \rightarrow Y$ is in F^+ which is a contradiction

Why Armstrong's Axioms?

Why are Armstrong's axioms (or an equivalent rule set) appropriate for FD's? They are:

- *Consistent*: any relation satisfying FD's in F will satisfy those in F^+
- *Complete*: if an FD X → Y cannot be derived by Armstrong's axioms from F, then there exists some relational instance satisfying F but not X → Y

- In other words, Armstrong's axioms derive ***all*** the FD's that should hold

Proving Consistency

We prove that the axioms' definitions must be true for any instance, e.g.:

- For augmentation (if $X \rightarrow Y$ then $XW \rightarrow YW$):

  If an instance satisfies $X \rightarrow Y$, then:

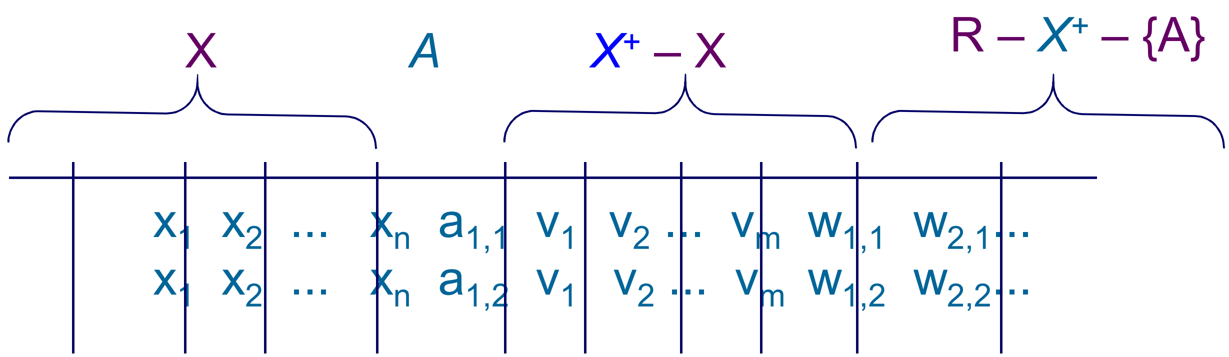
  - For any tuples $t_1, t_2 \in r$,
    if $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$ by defn.

  - If, additionally, it is given that $t_1[W] = t_2[W]$,
    then $t_1[YW] = t_2[YW]$

Proving Completeness

Suppose X → Y $\notin$ F^+ and define a relational instance *r* that satisfies F^+ but not X → Y:

- Then for some attribute A $\in$ Y, X → A $\notin$ F^+
- Let some pair of tuples in *r* agree on X^+ but disagree everywhere else:



Proof of Completeness cont'd

- Clearly this relation fails to satisfy $X \rightarrow A$ and $X \rightarrow Y$.
  We also have to check that it satisfies any FD in F^+.

- The tuples agree on only X^+.
  Thus the only FD's that might be violated are of the form $X' \rightarrow Y'$ where $X' \subseteq X^+$ and Y' contains attributes in
  $R - X^+ - \{A\}$.

- But if $X' \rightarrow Y' \in F^+$ and $X' \subseteq X^+$ then $Y' \subseteq X^+$ (reflexivity and augmentation).
  Therefore $X' \rightarrow Y'$ is satisfied.