08. Relational Database Design

[PM Jat, DAIICT, Gandhinagar]

Contents

Database Design as a Process

Database design is a process of determining "good" database schema.

The measure of goodness depends on the data model. In general, however, minimizing "data redundancies" is the striving objective of database design. Data redundancy primarily means a fact/value is occurring at more than one place in the database.

There is no formal way of measuring goodness for ER Model. ER model should capture complete and correct representation of all data and its semantics.

Relational model defines a formal method of measuring goodness of a relation, called Normal Forms. Normal Forms are 1NF, 2NF, 3NF, BCNF, 4NF, and 5NF; higher the Normal Form, lesser the redundancies are, and the "better" a relation is.

Relational Database Design as process

Input:

- Database Requirement Specifications
- Data items and constraints

Final Output:

- (Good) Relational schema
- Measure of Goodness: Minimum data redundancy

Before getting into details of these formal measures, let us first understand what the "data redundancy" is, and what are undesirable consequences of redundancy.

Data Redundancies and update Anomalies

Consider the following relations of the XIT database and its relation schema that we have been using in previous chapters, are shown below. If you follow the ER route where you convert ERD into relational schema using ER to relations mapping rules that we have studied in the previous chapter.

studid character	name character	progid character	cpi numeric
101	Rahul	BCS	8.70
102	Vikash	BEC	6.80
103	Shally	BEE	7.40
104	Alka	BEC	7.90
105	Ravi	BCS	9.30

pid character	pname character vary	intake smallint	did character
BCS	BTech(CS)	30	CS
BEC	BTech(ECE)	40	EE
BEE	BTech(EE)	40	EE
BME	BTech(ME)	40	ME

did character	dname character varying(30)
CS	Computer Engineering
EE	Electrical Engineering
ME	Mechanical Engineering

Now, suppose, we want to explore if above relations are good enough?

To answer this question, let us consider some of its alternatives. Compare and see if this is better over other options. One possibility is, let us say, we merge PROGRAM and DEPARTMENT and have a single relation PD (is basically program relation with an additional attribute DNAME). This is basically JOIN of PROGRAM and DEPARTMENT.

pid characte	pname character vary	intake smallint	did chara	dname character varying(30)
BCS	BTech(CS)	30	CS	Computer Engineering
BEC	BTech(ECE)	40	EE	Electrical Engineering
BEE	BTech(EE)	40	EE	Electrical Engineering
BME	BTech(ME)	40	ME	Mechanical Engineering

Does the above represent a correct set of facts? Try understanding this -

- What will be the interpretation of a tuple in combine relation PD?
- What will be the key in PD?

A tuple of PD contains all information of as program only, value of attribute DNAME is interpreted as name of department offering the program. Key of relation is PID. Do you see "data redundancy" here?

If not very obvious, let us also consider another case. Here we have combined relation of Student and Program relations (and let us call it SP), as shown below-

studid charact	name character	cpi numeric	progid charact	pname character vary	intake smallin	did chara
101	Rahul	8.70	BCS	BTech(CS)	30	CS
102	Vikash	6.80	BEC	BTech(ECE)	40	EE
103	Shally	7.40	BEE	BTech(EE)	40	EE
104	Alka	7.90	BEC	BTech(ECE)	40	EE
105	Ravi	9.30	BCS	BTech(CS)	30	CS

What are the undesirable consequences of redundancies?

Considering cost of modern storages, storage should not be of a concern, and even if it is, it probably gets compensated by reduced requirement performing joins wile answering queries (note most queries we need to join).

More serious problem associated with redundancies "database update anomalies".

One anomaly, you hopefully must have already noted, in combined PD, we cannot have a department details unless there is some program offered in that department (in PD). Or we cannot have a program detail unless there is some student attached with it (in SP)? This may be seriously undesirable.

1. Insertion Anomaly
2. Modify Anomaly
3. Deletion Anomaly

Insertion Anomaly

- Attempt inserting a new program in PD?
- Attempt inserting a new Department in PD?

While inserting, a new program, we need to supply all Department details as well. In this case we only have attribute DNAME; but could be more in other cases – for example in combined SP, attempt adding a new student. This is not only discomfort but we may also end up having inconsistent program details (in SP); inconsistent, since it is to be repeated with every program it offers.

Inserting a new Department in PD, without a program is not possible? We can choose to have null values for the program portion of data but since PID is the key; it cannot be NULL.

These are the two examples of Insertion anomalies caused due to data redundancy.

Modification Anomaly

- Attempt changing program for a student in SP? Not only it requires we supply, all information of the new program for the student in question, it may also lead database having inconsistent details of the program in database; there is a possibility of two tuples having same program information do not have the same program information.

Deletion Anomaly

- If we delete the program, and that program happens to be only program, the department also gets deleted; that is we lose the department details as well!
- How do we delete a department? Deletion basically becomes changing department attributes value of a program to either NULL or some other department data. This is weird.

So, avoiding redundancy and hence avoiding update anomalies is the objective of relational schema design. As already said, normal forms are the measure of goodness of a relation schema. Normal forms are defined in terms of "functional dependencies".

Next, we attempt to understand functional dependencies.

Function Dependencies

Function Dependency is a type of dependency among attributes of a database.

Let us say we put all attributes of the XIT database in a single relation, and let us call it RXIT. Below is a depiction of it, and is a join of STUDENT, PROGRAM, and DEPARTMENT relation of the XIT database.

studid charact	name character	cpi numeri	progid charact	pname character varyi	intake smallint	did chara	dname character varying(30)
101	Rahul	8.70	BCS	BTech(CS)	30	CS	Computer Engineering
102	Vikash	6.80	BEC	BTech(ECE)	40	EE	Electrical Engineering
103	Shally	7.40	BEE	BTech(EE)	40	EE	Electrical Engineering
104	Alka	7.90	BEC	BTech(ECE)	40	EE	Electrical Engineering
105	Ravi	9.30	BCS	BTech(CS)	30	CS	Computer Engineering

Do you observe the following in relation RXIT? When a value in the **DID** column repeats, then the value in the **DNAME** column also repeats. Are you able to explain why it is happening?

Same way, when a value in the **PROGID** column repeats, then the value in the **PNAME** column also repeats. Then, actually, values in other columns INTAKE, DID, and DNAME also repeats. Do you find any reason why is it happening?

These are, actually, observations of function dependency. Let it be stated as follows:

Suppose there are two tuples t1 and t2 from a relation r, if we have t1[X] = t2[X] and then we also have t1[Y] = t2[Y], then we say that Y is functionally dependent on X.

A Function Dependency also referred to as FD in short, is expressed as $X \rightarrow Y$.

This is also read as X functionally determines Y.

Functional Dependencies are used to formally detect "redundancies in a relation".

Note that FDs come from the meaning of attributes, not by looking at instances (in an instance repetition may be merely coincidental).

Let there be another informal way of understanding and detecting function dependencies.

Let us begin with $f: X \rightarrow Y$; where X and Y are attributes in a database. Note that X and Y can be attribute sets as well. Let this mapping be understood as follows: "**For given the value x for attribute X, the function maps to a single value of attribute Y. If such a mapping exists, then we say; *X functionally determines Y* or Y is functionally dependent on X**".

For example, consider searching DNAME for a given PROG-ID, say BCS, you find that it gives you a single value "Computer Engineering", even if it occurs at multiple places it gives us a single value. Therefore we say that PROG-ID functionally determines DNAME, i.e. $PROGID \rightarrow DNAME$.

On the contrary, on searching for STUDENT-ID for a given PROG-ID, you do not get a single value; and it is as expected that there are multiple students for a given prog-id. Therefore, we say that PROG-ID does not functionally determine STUDENT-ID.

It is important to note while studying functional dependencies, that, we see attributes as independent data items and remain oblivious to relations. As in the XIT database, we look at all attributes: STUDID, NAME, STUD_PROGID, CPI, PROG_NAME, INTAKE, PROG_DID, DNAME. And study FDs independent of any relation.

Hope you also find following functional dependencies that hold true on XIT database.

studid→ name	**studid→ dname**
studid→ cpi	**progid→ pname**
studid→ progid	**progid→ did**
studid→ pname	**progid→ dname**
studid→ intake	**progid→ intake**
studid→ did	**did → dname**

Another perspective to understand function dependency

Let X and Y be some subset of attribute from R (may take RXIT mentioned above), and then we run following query on R-

```
SELECT distinct Y FROM R WHERE X=xval;
```

If the query returns a single value for Y, then we can say that there is a FD $X \rightarrow Y$. Try experimenting this with following relation, and verify above set of FDs.

What will be returned from following queries when executed on RXIT?

```
SELECT distinct dname FROM RXIT WHERE progid= 'BCS';
SELECT distinct intake FROM RXIT WHERE progid= 'BCS';
SELECT distinct studentidFROM RXIT WHERE progid= 'BCS';
```

And, we can infer that

$progid \rightarrow dname$

$progid \rightarrow intake$

$progid \not\rightarrow studentid$

Why do we study Functional Dependencies?
1. FDs are formal means of detecting redundancies 2. Used for evaluation of a relational design 3. Used for improving a relational design

Expression of Functional Dependencies

Remember in FD, $X \rightarrow Y$; both are sets. Suppose X is set of attribute A and B, and Y has attributes C and D, then X and Y, respectively represented as {A,B} and {C,D}, and function dependency is expressed as-

$$\{A, B\} \rightarrow \{C, D\}$$

However for short, often you find it expressed as AB → CD

Exercise #1: Identify FDs in Company Database

Considering universal schema for company database,
RCOMP(eno, name, salary, super_eno, bdate, gender, emp_dno, dname, mgr_eno, mgrstartdate, dlocation, pno, pname, plocation, hours, dep_name, dep_gender, dep_bdate, dep_relationship)

eno	ename	salary	super_eno	dno	dname	mgr_eno	pno	pname	hours	pdno	pdname	pdmgreno
105	Jayesh	55000		1	Headquater	104	1	ProductX	32.5	5	Research	102
102	Sanna	35000	105	5	Research	102	1	ProductX	32.5	5	Research	102
106	Vijay	53000	105	4	Administration	106	1	ProductX	32.5	5	Research	102
108	Priya	45000	106	4	Administration	106	1	ProductX	32.5	5	Research	102
104	Ramesh	38000	106	5	Research	102	1	ProductX	32.5	5	Research	102
103	Kalpesh	25000	102	5	Research	102	1	ProductX	32.5	5	Research	102
107	Ahmad	25000	106	4	Administration	106	1	ProductX	32.5	5	Research	102
111	Kirit	40000	102	5	Research	102	1	ProductX	32.5	5	Research	102
105	Jayesh	55000		1	Headquater	104	2	ProductY	7.5	5	Research	102
102	Sanna	35000	105	5	Research	102	2	ProductY	7.5	5	Research	102
106	Vijay	53000	105	4	Administration	106	2	ProductY	7.5	5	Research	102
108	Priya	45000	106	4	Administration	106	2	ProductY	7.5	5	Research	102
104	Ramesh	38000	106	5	Research	102	2	ProductY	7.5	5	Research	102
103	Kalpesh	25000	102	5	Research	102	2	ProductY	7.5	5	Research	102
107	Ahmad	25000	106	4	Administration	106	2	ProductY	7.5	5	Research	102
111	Kirit	40000	102	5	Research	102	2	ProductY	7.5	5	Research	102
105	Jayesh	55000		1	Headquater	104	3	ProductZ	40	5	Research	102
102	Sanna	35000	105	5	Research	102	3	ProductZ	40	5	Research	102
106	Vijay	53000	105	4	Administration	106	3	ProductZ	40	5	Research	102
108	Priya	45000	106	4	Administration	106	3	ProductZ	40	5	Research	102

Does following expressions sound you correct in company schema? What check do we apply?

eno→dname
eno→ hours
eno→ pname
super_eno→ eno
pno→ dname
(eno, pno) → hours

Following is complete set of function dependencies in company database-

eno→ ename	eno→ mgrstartdate	pno→ dname
eno→ salary	dno→ dname	pno→ mgr_eno
eno→ super_eno	dno→ mgr_eno	pno→ mgrstartdate
eno→ bdate	dno→ mgrstartdate	{eno, pno} →hours
eno→ gender	pno→ pname	{eno, dep_name} →dep_gender
eno→ dno	pno→ dno	{eno, dep_name} → dep_bdate
eno→ dname	pno→ plocation	{eno, dep_name} → relationship
eno→ mgr_eno		

It is important to note that functional dependencies are drawn from problem domain and essentially capture database constraints.

For example -

- When we say dno→ dname, it means is that there is only one name for a given dno.
- Given FD eno→ super_eno implies that there is only one supervisor for an employee.
- Being No FD super_eno → eno, implies that there can be multiple eno for a given super_eno; similary, Not being FD dno→dlocation, mean that there can be multiple locations for a department.
- Being FD {Course_No, AcadYr, Semester} → FacultyID implies that there can be only one instructor for a course offering.

Exercise #2: Function Dependencies for DA-Acad.

Following are all attributes; having understood of database requirements and various constraints. Attempt figuring out functional dependencies -

RACAD(StudetID, StdName, ProgID, Batch, CPI, Semester_SPI, Semester_CPI, CourseNo, CourseName, Credit, course_grade, FacultyID, FacultyName, AcadYear, Semester)

StudetID→ StdName	FacultyID→ FacultyName
StudetID→ ProgID	{StudetID,AcadYear,Semester} → Semester_SPI
StudetID→ Batch	{StudetID,AcadYear,Semester} → Semester_CPI
StudetID→ CPI	{StudetID,CourseNo,AcadYear,Semester} → Course_Grade
CourseNo→ CourseName	{CourseNo,AcadYear,Semester} → FacultyID
CourseNo→ Credit	

Key defined in terms of functional dependencies

Consider this: if there are two tuples in the "Student" table in the XIT database having the same value of the attribute "Student-ID" then both tuple has to be identical. Right? **And, why**?

If an attribute or a set of attributes repeats in a relation, and that leads to duplication of the tuple then that set of attribute would be a super-key. We do not want tuples to repeat. Why? A basic characteristic of a relation – "tuples are distinct by values".

From here, we can draw a definition of Key in terms of FD:

Consider a relation R(X,Y), where X and Y are disjoint attribute Sets (and note X U Y is R). If we have FD X → Y, then X cannot repeat? Because that will make Y to repeat and that will make tuple to repeat.

- Therefore, X is super-key.
- If X is minimal, then it is Key.

In other words: If a set of attributes X that functionally determines all other attributes of a relation R, and X is minimal, then X is Key [candidate key].

Trivial Function Dependency

X → Y is trivial if Y is a subset of X. This is trivial because the subset will always be functionally determined by the superset. For example: {ENO, ENAME} → ENAME. Proof?

Non-trivial FDs: FD X → Y is "non-trivial" when Y is not a subset of X; but there can be some overlap of attributes; for example: {ENO} → {ENO, ENAME}.

An FD is said to be completely non-trivial if X and Y are disjoint in an FD, X →Y

Trivial FDs are simply discarded, as these are always true, and make no distinction.

Functional Dependency inference rules

A database scenario typically has a finite set of FDs and is often referred to as F. There is always a "base" set of functional dependencies in the given database scenario. F may have some "inferred" or "implied" FDs from the base set. Implied FDs are redundant FDs. A base set is a minimal set of FDs with no "inferred" FDs and is typically referred as F_{min}.

Often we require computing a "minimal FD set" from an identified set of FDs. Certain functional dependency inference rules are used for detecting implied or inferred functional dependencies and finally, such FDs can be dropped for computing a minimal set.

Following are three basic inference rules, known as "Armstrong's axioms"-

- Reflexive Rule: if Y ⊆ X then X→Y; basically trivial FD rule.
  - Example: {ENO, PNO} → {PNO}
- Augmentation Rule: {X→Y} ⊨ XZ → YZ
  - Example: {ENO} → {ENAME} ⊨{ENO, PPNO} → {ENAME, PNO}
- Transitive Rule: {X→Y, Y→Z} ⊨ X → Z
  - Example: {ENO} → {DNO} and {DNO} → {DNAME} ⊨ {ENO } → {DNAME}

We can prove these rules by going back to the basic definition of function dependency!

There are certain derived rules from these basic axioms that become handy in identifying implied FDs; are following -

- Union Rule: {X→Y, X→Z} ⊨ X → YZ
  - Example: {ENO} → {ENAME} and {ENO} → {SALARY} ⊨ {ENO} → {ENAME, SALARY}
  - Proof:
    X→Y ⊨X→XY (3)
    X→Z ⊨XY→YZ (4)
    Transitively from (3) and (4) ⊨ X→YZ
- Decomposition Rule:{X→YZ} ⊨ X → Y, and X → Z
  - Example: {ENO} → {ENAME, SALARY} ⊨ {ENO} → {ENAME} and {ENO} → {SALARY}
  - Proof:
    X→YZ and YZ → Y ⊨X→Y
    X→YZ and YZ → Z ⊨X→Z
- Pseudo-transitivity Rule: If X → Y and WY →Z, then WX → Z
  - Proof:
    X → Y (FD1-given)
    WY → Z(FD2-given)
    WX → WY(FD3- using Augmentation rule to FD1)
    WX → Z(using transitive rule on FD2 & FD3)

Exercise #3:

Given FD set AB → C, BC → AD, D → E, CF → B Can we infer FD AB→D?	*Solution:* AB → C (given) ⊨AB → BC, and BC → AD (given) ⊨BC → D Applying transiting AB → BC and BC → D, we can infer that AB → D

Clouse of FD set

There is a notion of closure set of a FD set F, denoted as F^+.

It is a set of FDs that include F and all inferred FDs from F.

For example if you have FD set F = { A → B, A → C, B → D }, F^+ will include all possible FDs like A → {B,C}, A → {B,C,D}, A→ D, or so.

You can see that closure is just opposite of base set (minimal set).

By iteratively applying Armstrong's axioms on a FD set, its closure can be computed. Closure set normally is not computed. The concept is however used in various discussions. For instance, it can be used for checking if a FD holds true on a database or not. If the FD in discourse be proved to be in F^+, it is inferred from F.

Closure of Attributes

Closure of Attribute (or set of attributes) is set of all attributes that are functionally determined by the attribute(s). Closure is represented by X^+.

Here is algorithm for computing closure. It goes as following-

Begin with setting X to X^+, and scan all FDs, and wherever Left side of a FD is subset of X^+, add Right side attributes to X^+. This is repeated till we find X^+ is unchanged in an iteration.

```
Input: X, F
Output: X+

X+ := X;
repeat
   oldX+ := X+
   for each fd Y→Z in F do
      if X+ is superset of Y then
       //this means X+ → Y ⊨ X+ → Z ⊨ X → Z
       //Therefore Z can be added X+
          X+ := X+U Z;
until (X+= oldX+);
```

Here is why algorithm will work correctly?

Consider a FD Y→Z, when we find left side of FD, i.e. Y, is subset of X^+, then we can say that X^+→Y (Reflexive Rule, trivial FD); we have following FDs, as premise:

$X \rightarrow X^+$; $X^+ \rightarrow Y$; and $Y \rightarrow Z$

From here, we can infer that $X \rightarrow Z$, therefore Z can be appended to X^+.

Exercise #4: Compute the following attribute closure using this algorithm

1. XIT database: $\{StudID\}^+$
2. Company Database: $\{ENO, PNO\}^+$
3. F = {AB → C, BC → AD, D → E, CF → B}, compute $\{AB\}^+$

Where can we use attribute closure?

- First, closure can be used to check if a functional dependency is implied from a given set F; for example, does FD X → Y implied from F? It can be answered by checking - if **Y is subset of X^+** then answer is YES, otherwise NO.
- Second, closure can be used for determining the key of a relation. If the closure of any set of attributes X contains all attributes of a relation, then we say that X is super-key of relation. If X is minimal then it is key. (will see little later)

Exercise #5: Given R(A,B,C,D,E,F), and F = {AB → C, BC → AD, D → E, CF → B},
Does FD {A,B}→{D}inferred from F?

Apply attribute closure algorithm and, we get $\{A,B\}^+ = \{A,B,C,D,E\}$;

This implies that AB determines D.

FD Set Equivalence

Suppose there are two FD sets F and G.

If all FDs of G can be inferred (or implied) from F, then we say that F covers G.

If F covers G and G covers F then we say that FD sets F and G are equivalent.

The next exercise is to check if the given FD sets F and G are equivalent!

Exercise #6: Are the following two sets equivalent (question sourced from Elmasri/navathe)?

F = {A → C, AC → D, E → AD, E → H} and G = {A → CD, E → AH}.

Two sets F, and G are equivalent only if G can be inferred by F and F can be inferred from G.

F	G
A → C (F1) AC → D (F2) E → AD (F3) E → H (F4)	A → CD (G1) E → AH (G2)

Let us see if FDs of G are inferred from F?

FD, G1: A→CD?

We can show that C is redundant in FD AC → D (F1)?

A → C (F1), A → AC (by augmentation rule)

A → AC and AC → D (F2), we can say that A → D (let us say F5)

By applying UNION Rule on F1 and F5, we get **A → CD**, that is G1

FD, G2: E→AH?

By applying Decomposition rule on F3, we get E → A (let us say F6)

By applying UNION Rule on F(6) and F(4), we get **E→AH**

So, all FD of G is covered by F.

Now see if FDs of F are inferred from G.

FD, F1: A → C? We get this by applying the decomposition rule on A → CD (G1).

FD, F2: AC → D?

By applying augmentation rule G1, we get AC → CD, and then

by applying the decomposition rule to it we get **AC → D**

FD, F3: E → AD?

We get E → A (Let us say G3) by applying decomposition rule on (G2)

We get E → D (Let us say G4) by applying the transitive rule on G3 and A → D (get by applying decomposition rule on G1)

By applying the union rule on G3 and G4 we get **E → AD**

FD, F4: E → H? We get this by applying the decomposition rule on E → AH (G2)

Minimal FD set

A FD set, that does not have any implied FD called as Minimal FD set.

This has the advantage of having to work with less number of Functional Dependencies (still covers all FDs). For example, consider FDs in XIT database; following FDs are inferred and can be dropped. By dropping inferred FDs, we do not lose any FD/constraint.

studid→ pname

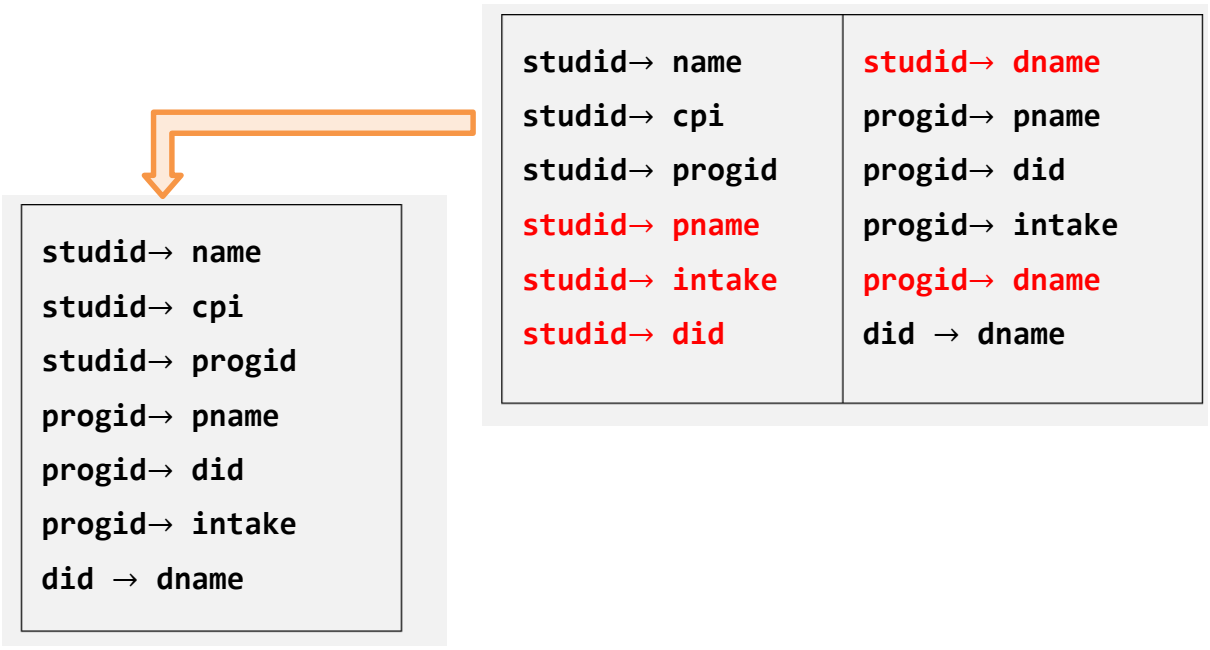
studid→ intake

studid→ did

studid→ dname

progid→ dname

The figure below shows the minimal FD set for XIT database-



Algorithm for Determining Minimal Set

- Drop all trivial FDs
- Write the FDs in following (canonical) form-
  - Have only one attribute in right hand side and
  - Make left side "irreducible"
- Remove redundant FDs if any (i.e. No inferred FDs)
  - Should be easy to notice duplicate and trivial FD when written in canonical form
  - See if there are any transitively inferred FDs
  - See still there are some inferred FDs; remove them.

Korth's book calls minimal FD set "**Canonical Cover**" for a given FD set.

Exercise #7: Compute Minimal Set of F = {A → C, AC → D, E → AD, E → H}

Make single attribute at right side

= {A → C, AC → D, E → A, E → D, E → H}

Make left side irreducible?

Except 2nd FD, other FD already have single attribute; in 2nd FD see if we can drop either of attribute?

FD1: A → C(given)

FD2: A →AC (Augmentation Rule)

FD3: AC → D (given)

FD4: A → D (transitively inferred from FD2 and FD3)

Thus we have A → D

Therefore we can conclude that C is redundant in FD AC → D and FD can be replace with A → D;Also, FDs E → A and A → D transitively infer E → D, and can be dropped; thus minimal FD set-

= {A → C, A → D, E → A, E → H}

Exercise #8: Compute Minimal Set of F = {AB → CD, A → E, E →C } {source: Korth book]

Make single attribute at right side

= {AB → C, AB → D, A → E, E → C}

Make left side irreducible?

There are two FDs AB → C and AB → D, where we may have redundant attributes.Since D is not appearing in any other FDs; dependency AB → D is already irreducible. For other FD AB → C, let us see if we can drop A or B. Since we have FDs A → E and E → C, transitively these FD infer A → C. Having A → C indicates that B is redundant in AB → C, and we have FD A → C replacing AB → C. Also since A → C is transitively inferred, it can be dropped. Thus we have following set as minimal

= {AB → D, A → E, E → C}

On Computing A^+, you should find C in it; and this is a confirmation that B is redundant in FD AB → C.

Alternate form for expressing minimal FD set-

- For all FDs, where X is determinant, combine all right hand side attributes using Union Rule; for example, we have X → A1, X → A2, X → A3; we replace all such FDs with a single FD, X → {A1, A2, A3}.

Computation of Key

For a relation R, and given set of FDs F, you can compute key for R as following-

- Pick one possible minimum set of attributes, X, and compute closure, if closure includes all attributes of R, then X is key.
- Ensure X is minimal, if so, it is Key; X is minimal if closure of none of its subset contains all attributes of relation R.
- A relation might have multiple key, you should also see if there is some other attribute(s) Y, is a key.

Exercise #9: Compute Keys

1. R(ABCD), F = {AB → C, AC → D}
2. R(ABCDE), F = {AB →C, CD → E}
3. R(ABCDE), F = {A → B, C → D, AC → E}
4. R(A,B,C,D,E,F), F = {AB → C, BC → AD, D → E, CF → B}
5. R(CourseNo, Sem, AcadYear, InstructorID, StudentID, Grade).
   Identify FDs yourself.

Solution for (5) above: Since close of AB does not include F; ABF^+ will be ABCDEF, and we say that ABF is the key; you should be able establish that closure of none of its subset will include all attributes. However there is another attribute set CF, and CF+ is ABCDEF

Therefore, the relation has two keys: ABF, and CF (and both are minimal)

Answersfor exercise 09:

1. AB
2. ABD
3. AC
4. ABF and CF
5. {Course-No, Semester, Acad-Year, Student-ID}

Exercises

1. Find out function dependency in attributes from TGMC scenario?
   a. MemberID
   b. MemberEmail
   c. TeamID
   d. TeamUserName
   e. TeamUserPassword
   f. MentorID
   g. MentorName
   h. InstituteID
   i. InstituteName

2. Do following FDs hold true? Attributes have been drawn from already seen situations.

   a. Does ISBN determine book title?
   b. Does book title determine ISBN?
   c. Does Accession No determine ISBN?
   d. Does ISBN determine Accession No?

e. Does Author ID determine book title?
   Assuming that an author can write multiple books and a book can have multiple authors.

3. Compute minimal set for following FD sets

   a. $\{A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C\}$
   b. $\{B \rightarrow A, D \rightarrow A, AB \rightarrow D\}$
   c. $\{AB \rightarrow C, B \rightarrow D, D \rightarrow A\}$
   d. $\{AB \rightarrow CD, B \rightarrow C, C \rightarrow D\}$
   e. $\{A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C\}$
   f. $\{ABC \rightarrow D, A \rightarrow B\}$

4. Compute Keys for given relation R and FD set F-

   a. R (A, B, C, D, E, F), and following FD set
      $A \rightarrow BDE$
      $F \rightarrow A$
   b. R (A, B, C, D, E) and FD set
      $AB \rightarrow C$
      $CD \rightarrow E$
      $DE \rightarrow B$
   c. R (A, B, C, D, E, F, G) and FD set
      $A \rightarrow B, AC \rightarrow ED, B \rightarrow F, A \rightarrow G$

5. Given RMED(Trade-Name, Generic-Name, Manufacturer, Batch-No, Stock, MRP, Tax-Rate), and following FDs, Compute Key
   a. Trade-Name → Generic-Name
   b. Trade-Name → Manufacturer
   c. Batch-No → Trade-Name
   d. Batch-No → Stock
   e. Batch-No → MRP
   f. Generic-Name → Tax-Rate