Functional Dependencies

Source/References:
Database Systems: The Complete Book, and Elmasir/Navathe

pm jat @ daiict

What is Functional Dependency?

- Redundancies are visible here?
- Do you see reason, why we have these redundancies, or why certain values are repeated for number of times?

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

What is Functional Dependency?

- If in a tuple t1, if you have some value, let us say x1 for attribute X and y1 for attribute Y; and if there is another tuple t2 that has x1 for X, it also has y1 for Y in t2.
- Then, we say that X → Y, or X functionally determines Y; or Y is functionally determined dependent on X.
- Note that FDs comes from meaning of attribute, not by looking at instances (in instance repetition may be merely coincidental)

Function Dependencies in XIT schema

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

studid → name	studid → dname
studid → prog_id	pid → pname
studid → cpi	pid → did
studid → pid	pid → dname
studid → pname	pid → intake
studid → intake	did → dname
studid → did	

Functional Dependency – another perspective

- Inspired from function in Math; where you have $y = f(x)$.
- Let us express it as $Y \leftarrow f(X)$, and interpret it as – for given the value x for Attribute X, the function maps to a single value of attribute Y. Note that X and Y refers to attribute sets here.
- If such function exists then we say;

  X functionally determines Y, and express as $\boldsymbol{X \rightarrow Y}$

- Having said this, and consider meaning of respective attributes, do following functions exist?

  $f(studid) \rightarrow name$; or, $studid \rightarrow name$

  $f(studid) \rightarrow intake$; or, $studid \rightarrow intake$

Functional Dependency – yet another perspective

- If we interpret the "function" as "retrieve", and say, if we "retrieve" a single value of attribute(s) Y for given **xval** for attribute(s) X, then we have FD **X → Y** !

- In other words, if following SQL expression returns single value (or Null), then we have FD **X → Y** on relation R.
SELECT distinct Y FROM R WHERE X=xval;

Exercise

- Does following expressions sound you correct in company schema-

f:ssn → dname
f:ssn → proj_dept_name
f:(ssn, pno) → hours
f:ssn → hours

FD Examples

- Following FDs do not hold in respective database:

(XIT)	**dname → pname**
(XIT)	**dname → studid**
(Company)	**ssn → pname**
(Company)	**superssn → ssn**
(DA-Acad)	**studid → spi**
(DA-Acad)	**Coourse_No → instructor_id**

Exercise: do following FDs hold in Company Schema?

PNO → PLOCATION

DNO → DLOCATION

SSN → HOURS

SSN → SUPER_SSN

SSN → MGR_SSN

MGR_SSN → SUPER_SSN

DNO → SUPER_SSN

PNO → MGR_SSN

FDs in company schema

- Here are all FDs in company schema; make sure that you "agree" with all of these! (note the red-part in some names)

ssn → fname	ssn → mgrstartdate	pno → dno
ssn → minit	dno → dname	pno → plocation
ssn → lname	dno → mgrssn	pno → dname
ssn → salary	dno → mgrstartdate	pno → mgrssn
ssn → superssn	pno → pname	pno → mgrstartdate
ssn → bdate	{ssn, pno} → hours	
ssn → gender	{ssn, dep_name} → dep_gender	
ssn → dno	{ssn, dep_name} → dep_bdate	
ssn → dname	{ssn, dep_name} → relationship	
ssn → mgrssn		

FDs in company schema

- Hopefully names with some explanatory tags [red] added to them hopefully helps to understand FDs better, and some are already their in their original names.

ssn → fname ssn → minit ssn → lname ssn → salary ssn → superssn ssn → bdate ssn → gender ssn → dno ssn → dname ssn → mgrssn	ssn → mgrstartdate dno → dname dno → mgrssn dno → mgrstartdate pno → pname	pno → proj_dno pno → plocation pno → proj_dname pno → proj_mgrssn pno → proj_mgrstartdate
	{ssn, pno} → hours {ssn, dep_name} → dep_gender {ssn, dep_name} → dep_bdate {ssn, dep_name} → relationship	

Exercise: Functional dependencies

- Does ISBN determines book title?
- Does book title determines ISBN?
- Does AccessionNo determines ISBN?
- Does ISBN determines AccessionNo?
- Does AuthorID determines book title?
  Assuming that A author can write multiple books, and a book can have multiple authors.

FDs in TGMC

- List of attributes – Can you figure out FDs?
  - MemberID
  - MemberEmail
  - TeamID
  - TeamUserName
  - TeamUserPassword
  - MentorID
  - MentorName
  - InstituteID
  - InstituteName

Keys and Function Dependencies

- Consider a relation R(X,Y), where X and Y are disjoint attribute Sets (and note X U Y is R)
- If we have X → Y, then X can not repeat? <u>Why?</u>
- Therefore X is super-key.
- If X is minimal then it is Key.
- Definition of Key in terms of FD-
  - If a set of attributes X that functionally determines all other attributes of a relation R, and
  - X is minimal,

  <u>then X is Key</u>

Key defined in terms of FDs

- A set of attribute X is the key of relation R, when
  - X functionally determines all other attributes of R
  - There is no subset of X that functionally determine all other attributes of R, i.e. X is minimal.

- If there is no second condition is met, then X is Super- Key.

Trivial Functional dependencies

- X → Y is trivial if Y is subset of X.
- For example:
  - {SSN, FNAME} → FNAME; and
  - {SSN, FNAME} → SSN
- This is trivial because subset will always be functionally determined by superset.
- How do we prove this?

Trivial Functional dependencies

- Non-trivial FDs
  - FD X→Y, is "non-trivial", when Y is not subset of X; but if there some overlap of attributes; for example; {SSN} → {SSN, FNAME}
- Completely non-trivial FDs
  - FD X→Y, is "Completely non-trivial" if X and Y are disjoint.
- Trivial FDs are not of any use?

FD inference rules

- That is when you have a set of FDs given, you may be able to infer more FDs from that using “inference rules”.
- For example, consider following FD set in company schema; FDs in red are inferred from other FDs.

<table>
<tr>
<td rowspan="2">ssn → fname
ssn → minit
ssn → lname
ssn → salary
ssn → superssn
ssn → bdate
ssn → gender
ssn → dno
ssn → dname
ssn → mgrssn</td>
<td>ssn → mgrstartdate
dno → dname
dno → mgrssn
dno → mgrstartdate
pno → pname</td>
<td>pno → dno
pno → plocation
pno → dname
pno → mgrssn
pno → mgrstartdate</td>
</tr>
<tr>
<td colspan="2">{ssn, pno} → hours
{ssn, dep_name} → dep_gender
{ssn, dep_name} → dep_bdate
{ssn, dep_name} → relationship</td>
</tr>
</table>

Splitting Rules of FDs

- FD **{A,B,C} → {X,Y,Z}**, can be split to-

  {A,B,C} → X

  {A,B,C} → Y

  {A,B,C} → Z

- Example from Company Schema:
  {ssn} → {fname, salary, superssn}, can be split to

  ssn → fname

  ssn → salary

  ssn → superssn

- Note that we can not split from left hand side?

Combining Rules of FDs

- Reverse of splitting is also true, FDs

  {A,B} → X

  {A,B} → Y

  {A,B} → Z

- Above FDs can be combined to

  {A,B} → {X,Y,Z}

Transitive Rules

- If A→ B and B → C, A → C

Canonical form of writing FDs

- Right side is singleton
- Left side is minimum (or irreducible)

  pid → intake

  pid → dname

  {AcadYr, Sem, CourseNo, StudID} → grade

  {AcadYr, Sem, CourseNo} → FacultyID

- Note: Singleton attribute set is not enclosed in { }.

Canonical form of writing FDs

- However to express FD set compact, we may be combining right side for common left side.
- For example, FD set

  pid → pname

  pid → intake

  pid → did

  pid → dname

  Often may be expressed as-

  pid → {pname, intake, did, dname}

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 computed for given R.
- Method:
  - Input: a set of FDs (that hold on R), and attribute X.
  - Output: closure of X, that is represented as X^+
- X+ gives set of ALL attributed that are functionally dependent on X.

Algorithm for computing X^+

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

```
X+ := X;
repeat
    oldX+ := X+
    for each fd Y→Z in F do
        if X+ is superset of Y then
            X+ := X+ U Z;
until (X+ = oldX);
```

Example: Computing X^+

```
X+ := X;
repeat
      oldX+ := X+
      for each fd Y→Z in F do
              if X+ is superset of Y then
                      X+ := X+ U Z;
until (X+ = oldX);
```

Compute **{SSN}**$^+$ and **{SSN, PNO}**$^+$ in Company Schema using this Algorithm?

Example Computing $\{SSN\}^+$, $\{SSN,PNO\}^+$

```
X+ := X;
repeat
      oldX+ := X+
      for each fd Y→ Z in F do
            if X+ is superset of Y then
                  X+ := X+ U Z;
until (X+ == oldX);
```

FD set F:

ssn → {fname, salary, superssn, dno}

ssn → {dname, mgrssn, mgrstartdate}

dno → {dname, mgrssn, mgrstartdate}

pno → {pname, proj_dno}

{pno, ssn} → hours

{ssn, dp_name} → {dp_birthdate, relationship, dp_gender}

Computing Closure of Attributes

- Consider relation R(A,B,C,D,E,F), and
- Set of FDs (F)-
  - $AB \rightarrow C$
  - $BC \rightarrow AD$
  - $D \rightarrow E$
  - $CF \rightarrow B$

```
X+ := X;
repeat
    oldX+ := X+
    for each fd Y→ Z in F do
        if X+ is superset of Y then
            X+ := X+ U Z;
until (X+ == oldX);
```

- Compute closure of AB, i.e. $\{A,B\}^+$.

Where do we use closure of attribute?

- First, it can be used to determine key.
  - If closure of any set of attributes is all attributes, then we say it is super-key. (If it is minimal then it is key)

- Second, it can be used to check if a FD holds on a relation, for example, does FD X → Y holds on a relation R? It can be answered by checking - if **Y is subset of X^+** then answer is YES, otherwise NO.

- Exercise: does FD AB → D hold on relation R in previous example?

Does FD AB → D hold on relation R?

- Consider relation R(A,B,C,D,E,F), and
- Set of FDs (F)-
  - AB → C
  - BC → AD
  - D → E
  - CF → B

```
X+ := X;
repeat
    oldX+ := X+
    for each fd Y→ Z in F do
        if X+ is superset of Y then
            X+ := X+ U Z;
until (X+ == oldX);
```

- Answer: Yes/No?

Determination of Key

- For a relation R, and given set of FDs F, you can compute key for R.
- Steps-
  - Pick one minimum possible set of attributes, and compute closure, if closure includes all attributes, then it is key
  - In order ensure that all keys, try computing closure of all possible (minimum) combinations

Example: determine Key(s)

- R (ABCD)

- FDs
  - AB → C
  - AC → D

Example: determine Key(s)

- Relation R(ABCDE)

- Has FDs
  - AB → C
  - CD → E

- Compute closure of ??

Example: determine Key(s)

- R(ABCDE)
- FDs
  - $A \rightarrow B$
  - $C \rightarrow D$
  - $AC \rightarrow E$

Example: determine Key(s)

- Consider relation R(A,B,C,D,E,F), and
- Set of FDs (F)-
  - AB → C
  - BC → AD
  - D → E
  - CF → B

Two Keys: ABF and CF (both are minimal)

Example: determine Key(s)

- Compute the closure of assumed key and prove that it is a valid key -

  R1(CourseNo, Sem, AcadYear, InstructorID, StudentID, Grade)

  R2(MedicineName, GenericName, Batch, MRP, ExpiryDate)

Role of Functional Dependencies

- Note that FDs are constraint on Database Schema, and any valid database state should not be violating these FDs.

Inference Rules for FDs [optional]

formally discussed in Elmasri/Navathe

- IR1(Reflexive Rule) : if $X \supseteq Y$ then $X \rightarrow Y$
- IR2(Augumentation Rule): $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
- IR3(Transitive Rule): $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$
- IR4(Decomposition, or Projective Rule):
  $\{X \rightarrow YZ\} \models X \rightarrow Y$
- IR5(Union or Additive Rule):
  $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$
- IR6(Psuedotransitivity Rule):
  If X -> Y and WY -> Z, then WX -> Z
- First three rules are called as Armstrongs axioms (the person who proposed these).

Inference Rules for FDs

- IR1(Reflexive Rule) : if $X \supseteq Y$ then $X \rightarrow Y$

- Such dependencies are called **trivial** dependencies, any FD $X \rightarrow Y$ is trivial if X is superset of Y. In practice we are not interested in trivial dependencies only.

Inference Rules for FDs

- IR2(Augumentation Rule): $\{X \rightarrow Y\} \models XZ \rightarrow YZ$

- In words: adding same set of attributes at both sides given another valid FD.

- IR3(Transitive Rule): $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$

Inference Rules for FDs

- IR4(Decomposition, or Projective Rule):

  $$\{X \rightarrow YZ\} \models X \rightarrow Y \text{ and } X \rightarrow Z$$

- IR5(Union or Additive Rule):

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

  – This is reverse of IR4

Inference Rules for FDs

- IR6(Psuedotransitivity Rule):
  If X -> Y and WY -> Z, then WX -> Z

- Proof:

  X → Y (FD1-given)

  WY → Z (FD2-given)

  WX → WY (FD3- using additive rule IR2 to FD1)

  WX → Z (using transitive rule on FD2 & FD3)

Minimal set of FDs

- If we have a set of FDs that is base set and no inferred FDs are there in the set, this base set is called as minimal FD set.
- This has the advantage of having to work with less number of Functional Dependencies (still covers all FDs)
- Consider FDs in EMP-DEP relations; FDs in red are inferred from others, and can be dropped-

ssn -> fname
ssn -> salary
ssn -> superssn
ssn -> dno
ssn -> dname
ssn -> mgrssn
ssn -> mgrstartdate
dno -> dname
dno -> mgrssn
dno -> mgrstartdate

Minimal set of FDs

- Therefore by dropping inferred FDs from the set, we do not loose any FD; anyway those can be inferred from base set
- Minimal set can be determined as following-
  - Drop all trivial FDs
  - Write the FDs in canonical form, i.e.
    - Have only one attribute in right hand side and
    - make left side “irreducible”.
    - Remove redundant FDs if any (like transitive or so)
- Note: FDs, unless otherwise written, mean non-trivial FDs

Minimal set of FDs- examples

ssn -> fname
ssn -> salary
ssn -> superssn
ssn -> dno
ssn -> dname
ssn -> mgrssn
ssn -> mgrstartdate
dno -> dname
dno -> mgrssn
dno -> mgrstartdate

ssn -> fname
ssn -> salary
ssn -> superssn
ssn -> dno
dno -> dname
dno -> mgrssn
dno -> mgrstartdate

Minimal set of FDs- examples

TradeName → GenericName
TradeName → Manufacturer
BatchNo → TradeName
BatchNo → GenericName
BatchNo → Stock
BatchNo → MRP
BatchNo → TaxRate
BatchNo → Manufacturer
GenericName → TaxRate

TradeName → GenericName
TradeName → Manufacturer
BatchNo → TradeName
BatchNo → Stock
BatchNo → MRP
GenericName → TaxRate

Sources/References

- Database Systems: The Complete Book, by
  Hector G. Molina, Jeff Ullman, and Jennifer Widom
  Pearson Education, India
  Home Page:
  http://infolab.stanford.edu/~ullman/dscb.html

- Elmasir/Navathe