Functional Dependencies

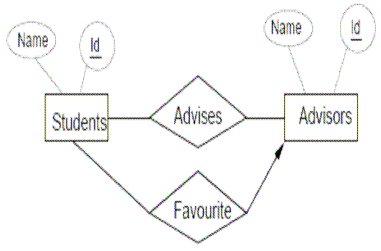
Meaning of FD's
Keys and Superkeys
Inferring FD's

1

Motivation

- Consider the relation:

Students(Id, Name,
AdvisorId,
AdvisorName,
FavouriteAdvisorId)



2

Motivation (2)

- If you know the student's ID can you determine the value of other attributes?

Name

FavouriteAdvisorId

- We say that `ID` *functionally determines* `Name` and `FavouriteAdvisorID`

- Denoted as:
  - **ID->Name**
  - **ID -> FavouriteAdvisorID**

3

Motivation (3)

- Similarly:
  - **AdvisorId -> AdvisorName**

- Can we say **Id -> AdvisorId**?
- No! Because ID is NOT a key for us for the relation described!

- This relation is bad because the "key" does not determine the other attributes.
- Decomposing relation into three relations **Students**, **Advisors** and **Advises** fixes the problem.

4

Another Example

- Consider the following relation

  Movies (title, year, length, filmtype, studioName, starName)

- FDs we can reasonable assert are:

Title, year -> length

Title, year -> filmType

Title, year -> studioName

- Intuitively, if two tuples in Movies have the same value of their title and year, then they MUST have the same length, `filmType` and `studioName`!

Title, year -\> starName

5

Formally

- A FD on R is a statement of the form – "if two tuples of R agree on attributes A1, A2,…,An then they MUST agree on another attribute B".
  - Notation: A1A2 . . .An -> B
  - The set of attributes `A1,A2, . . . An` ***functionally determine*** B.

- An FD is a constraint on a single relational schema. It must hold on every instance of the relation.

- An FD cannot be deduced from a relation instance.

6

Definition

- If `t` is a tuple in a relation `R` and `A` is an attribute of **`R`**, then t_A is the value of attribute `A` in tuple `t`.
- The FD **`AdvisorId -> AdvisorName`** holds in R if in every instance of R, for every pair of tuples t and u

if $t_{AdvisorId} = u_{AdvisorId}$, then

$t_{AdvisorName} = u_{AdvisorName}$.

`Students(Id, Name, AdvisorId, AdvisorName, FavId)`

Id	Name	AdvisorId	AdvisorName	FavId
HG	Hermione Grainger	AD	Albus Dumbledore	HtG
HG	Hermione Grainger	HtG	Hagrid	HtG
DM	Draco Malfoy	SS	Severus Snape	SS
DM	Draco Malfoy	MM	Minerva McGonagall	SS
HP	Harry Potter	AD	Albus Dumbledore	AD

7

- If:

$A_1\ A_2\ A_3 \ldots\ldots A_n \rightarrow B_1$

$A_1\ A_2\ A_3 \ldots\ldots A_n \rightarrow B_2$

$A_1\ A_2\ A_3 \ldots\ldots A_n \rightarrow B_3$

Then we can say:

$A_1\ A_2\ A_3 \ldots\ldots A_n \rightarrow B_1\ B_2 B_3$

8

Why do we need to study this?

- Formalism for reasoning about constraints on attributes in relational designs.
- Allow us to procedurally determine the keys of a relation.
- Allow us to improve database designs systematically using normalization.

9

Example

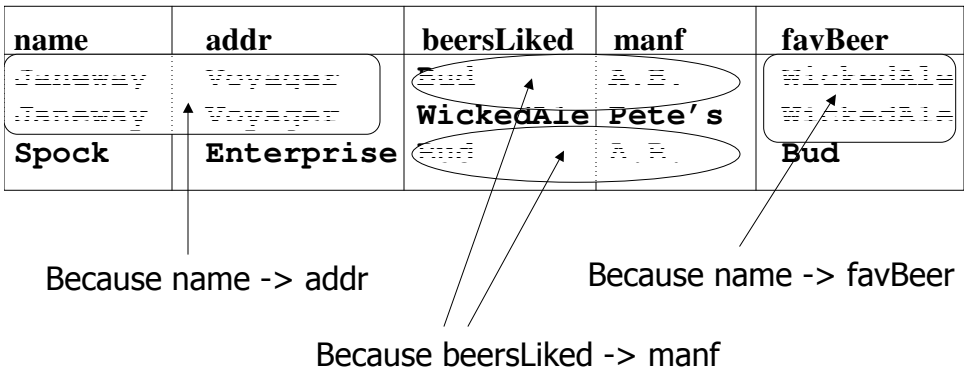
Drinkers(name, addr, beersLiked, manf, favBeer)

- Reasonable FD's to assert:

1. name -> addr
2. name -> favBeer
3. beersLiked -> manf

10

Example Data



11

Courses(Number DeptName CourseName Classroom Enrollment)

Number	DeptName	CourseName	Classroom	Enrollment
4604	CS	Databases	TORG 1020	45
4604	Dance	Tree Dancing	Drillfield	45
4604	English	The Basis of Data	Williams 44	45
2604	CS	Data Structures	MCB 114	100
2604	Physics	Dark Matter	Williams 44	100

Number DeptName -> CourseName

Number DeptName -> Classroom

Number DeptName -> Enrollment

Is Number -> Enrollment an FD?

12

FD's With Multiple Attributes

- No need for FD's with > 1 attribute on right.
  - But sometimes convenient to combine FD's as a shorthand.
- Example: name -> addr and
  name -> favBeer becomes
  name -> addr favBeer
- > 1 attribute on left may be essential.
  - Example: bar beer -> price

13

Keys of Relations

- FDs allow us to formally define keys.
- A set of attributes **{A1,A2, . . . An}** is a key for a relation R if

Uniqueness **{A1,A2, . . . An}** functionally determine all the other attributes of R **and**

Minimality no proper subset of **{A1,A2, . . . An}** functionally determines all the other attributes of R.

14

Superkeys

- A **superkey** is a set of attributes that has the uniqueness property but is not necessarily minimal.
- If a relation has multiple keys, specify one to be the primary key.
- Convention: in a relational schema, underline the attributes of the primary key.
- If a key has only one attribute `A`, we say that `A` rather than `{A}` is a key.

15

Example

- What is the key for

  Courses(Number, DeptName CourseName Classroom Enrollment)?

{Number, DeptName}.

- These attributes functionally determine every other attribute.

- No proper subset of **{Number, DeptName}** has this property.

16

Example

- What is the key for
  Teach(Number, DepartmentName ProfessorName Office)?

- The key is **{Number, DepartmentName}**
  Why?

17

Example

Drinkers(name, addr, beersLiked, manf, favBeer)

- **{name, beersLiked}** is a superkey because together these attributes determine all the other attributes.
  - name -> addr favBeer
  - beersLiked -> manf

18

Example, Cont.

- `{name, beersLiked}` is a **key** because neither `{name}` nor `{beersLiked}` can be a key.
  - `name` $\not\rightarrow$ `manf`; `beersLiked` $\not\rightarrow$ `addr`.
- There are no other keys, but lots of superkeys.
  - Any superset of `{name, beersLiked}`.

19

E/R and Relational Keys

- Keys in E/R concern entities.
- Keys in relations concern tuples.
- Usually, one tuple corresponds to one entity, so the ideas are the same.
- But --- in poor relational designs, one entity can become several tuples, so E/R keys and Relational keys are different.

20

Example Data

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	Voyager	WickedAle	Pete's	WickedAle
Spock	Enterprise	Bud	A.B.	Bud

Relational key = **{name beersLiked}**

But in E/R, **name** is a key for **Drinkers**, and beersLiked is a key for Beers.

Note: 2 tuples for Janeway entity and 2 tuples for Bud entity.

21

E/R to Relations

- If the relation comes from an entity set, the key attributes of the relation are precisely the key attributes of the entity set.

- If the relation comes from a binary relationship R between entity sets E and F:
  - **R is many-many:** key attributes of the relation are the key attributes of E and of F.
  - **R is many-one from E to F:** key attributes of the relation are the key attributes of E.
  - **R is one-one:** key attributes of the relation are the key attributes of E or of F.

22

ER to Relations

- If the relationship R is multiway, we need to reason about the FDs that R satisfies.
  - There is no simple rule.

23

Where Do Keys Come From?

1. Just assert a key *K*.
   - The only FD's are `K -> A` for all attributes *A.*

2. Assert FD's and deduce the keys by systematic exploration.
   - E/R model gives us FD's from entity-set keys and from many-one relationships.

24

More FD's From "Physics"

- Example: "no two courses can meet in the same room at the same time" tells us: hour room -> course.

25

Inferring FD's

- We are given FD's X_1 -> A_1, X_2 -> A_2,…, X_n -> A_n , and we want to know whether an FD Y -> B must hold in any relation that satisfies the given FD's.
  - Example: If A -> B and B -> C hold, surely A -> C holds, even if we don't say so.
- Important for design of good relation schemas.
- What is the Key for the schema `(A B C)` if **`A->B`** and **`B->C`**?
- `A`, since it can determine both `B` and `C`.

26

Equivalence

- An FD `F` **follows from** a set of FDs `T` if every relation instance that satisfies `F` also satisfies all the FDs in `T`
- *A –> C* follows from *T = {A –>B, B –> C}.*
- Two sets of FDs S and T are **equivalent** if S follows from T ***and*** T follows from S.
- *S = {A –>B,B –> C,A–>C}* and *T={A–> B, B –> C}* are equivalent.
- These notions are useful in deriving new FDs from a given set of FDs.

27

Splitting and Combining

- The set of FDs

```
A1 A2 . . .An –> B1
A1 A2 . . .An –> B2 . . .
A1 A2 . . .An –> Bm
```

- is equivalent to the FD

```
A1 A2 . . .An –> B1 B2 . . .Bm.
```

- This equivalence implies two rules.
- **Splitting rule:** we can split the FD T into the set of m FDs in S.
- **Combining rule:** we can combine the set of m FDs in S into the single FD T.
- These rules work because all the FDs in S and T have identical left hand sides.

28

- Can we split and combine left hand sides of FDs?

- For the relation `Courses` is the FD

`Number DeptName -> CourseName`

equivalent to the set of FDs

`{Number -> CourseName, DeptName -> CourseName}?`

- NO!

29

Triviality

- An FD `A1 A2 . . .An -> B1 B2 . . .Bm` is **trivial** if the B's are a subset of the A's.

{B1,B2, . . . Bn} ⊆ {A1,A2, . . . An}.

- It's non-trivial if at least one B is not among the A's, i.e.,

{B1,B2, ... Bn} – {A1,A2, ...An} ≠ Φ;.

- It's completely non-trivial if none of the B's are among the A's,

{B1,B2, ... Bn} ∩ {A1,A2, ...An} = Φ;.

30

- What good are trivial and non-trivial dependencies?

- Trivial dependencies are always true.
  - They help simplify reasoning about FDs.

- **Trivial dependency rule**: The FD `A1 A2 . . .An -> B1 B2 . . .Bm` is equivalent to the FD `A1 A2 . . .An -> C1 C2 . . . Ck` , where the C's are those B's that are not A's, i.e.,

{C1, C2, . ., Ck} = {B1,B2, . ,Bm} – {A1,A2, .. ,An}.

31

Closures

- Suppose a relation with attributes A,B, C,D, E, and F satisfies the FDs

AB ->C BC -> AD D -> E, CF -> B

- Given these FDs, what is the set X of attributes such that `AB -> X` is true?

X = {A,B, C,D, E}, i.e., `AB -> ABCDE.`

- What is the set Y of attributes such that BCF -> Y is true?

Y = {A,B, C,D, E, F}, i.e., `BCF -> ABCDEF`

- What is the set Z of attributes such that `AF -> Z` is true?

Z = {A, F}, i.e., `AF -> AF`.

- `X`, `Y` , and `Z` are the **closures** of **{A,B}, {B, C, F},** and **{A, F},** respectively.
- **{B, C, F}** is a superkey.

32

Formally

- Given a set of attributes **{A1,A2, . . . ,An}** and a set of FDs **S**, the **closure** of **{A1,A2, . . . ,An}** under the FDs in **S** is

- the set of attributes {B1,B2, . . . ,Bm} such that for
- 1 <= i <= m, the FD `A1 A2 . . .An -> Bi` follows from S.

- The closure is denoted by `{A1,A2, . . . ,An}+`.

33

- Which attributes must **{A1,A2, . . . ,An}+** contain at a minimum?

`{A1,A2, . . . ,An}`. Why?

Because **A1 A2 . . .An -> Ai** is a trivial FD.

34

Why Closures?

- Closures allow us to prove correctness of rules for manipulating FDs.
- **Transitive rule:** if **A1 A2 . . .An –> B1 B2 . . .Bm** and **B1 B2 . . .Bm –> C1 C2 . . . Cn** then **A1 A2 . . .An –> C1 C2 . . . Cn.**

- To prove this rule, simply check if

$\{C1, C2, \ldots, Cn\} \subseteq \{A1, A2, \ldots, An\}+$.

- Closures allow us to ***procedurally*** define keys. A set of attributes X is a key for a relation R if and only if

1. {X}+ is the set of all attributes of R and
2. for no attribute A ∈ X is {X − {A}}+ the set of all attributes of R.

35

Algorithm to determine Closures

- Given
  - a set of attributes **{A1,A2, . . . ,An}** and
  - a set of FDs S,
- compute **X = {A1,A2, . . . ,An}+.**

1. Set **X <– {A1,A2, . . . ,An}.**
2. Find an FD **B1 B2 . . .Bk –> C** in S such that **{B1,B2, . . . Bk} ⊆ X** but **C ∉ X.**
3. Add C to X.
4. Repeat the last two steps until you cannot find such an attribute C.
5. The final value of X is the desired closure.

36

Example

- Consider the "bad" relation
  - **Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId).**
- What are the FDs that hold in this relation?

Id –> Name

Id –> FavouriteAdvisorId

AdvisorId –> AdvisorName

- To compute the key for this relation,

1. Compute the closures for **all** sets of attributes.
2. Find the **minimal** set of attributes whose closure is the set of all attributes.

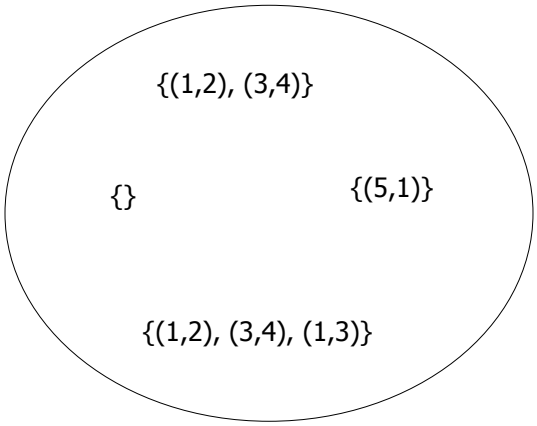
37

A Geometric View of FD's

- Imagine the set of all *instances* of a particular relation.
- That is, all finite sets of tuples that have the proper number of components.
- Each instance is a point in this space.

38

Example: R(A,B)



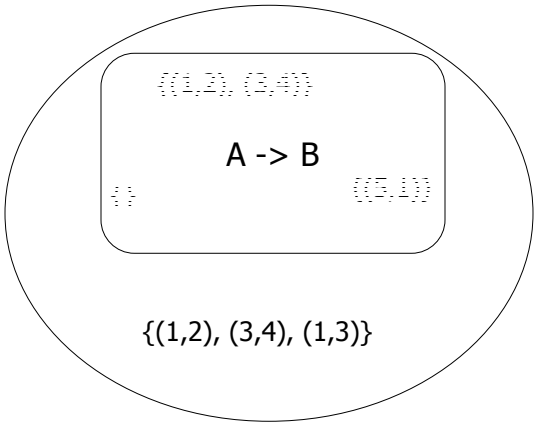
39

An FD is a Subset of Instances

- For each FD *X* -> *A* there is a subset of all instances that satisfy the FD.
- We can represent an FD by a region in the space.
- Trivial FD = an FD that is represented by the entire space.
  - Example: *A* -> *A*.

40

Example: A -> B for R(A,B)



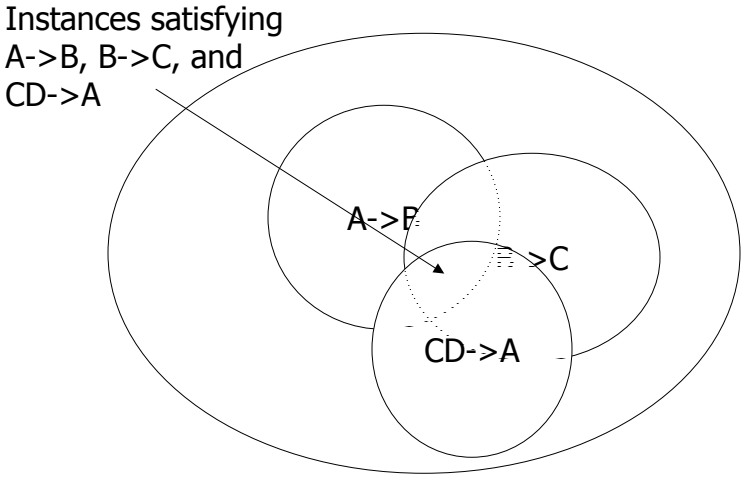
41

Representing Sets of FD's

- If each FD is a set of relation instances, then a collection of FD's corresponds to the intersection of those sets.
  - Intersection = all instances that satisfy all of the FD's.

42

Example



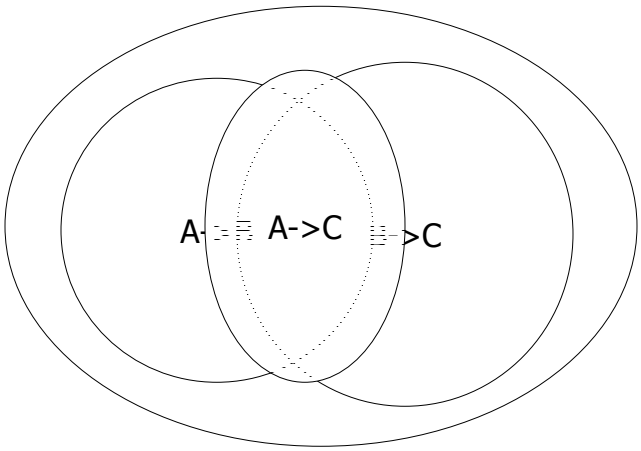
43

Implication of FD's

- If an FD Y -> B follows from FD's X_1 -> A_1,…,X_n -> A_n , then the region in the space of instances for Y -> B must include the intersection of the regions for the FD's X_i -> A_i .
  - That is, every instance satisfying all the FD's X_i -> A_i surely satisfies Y -> B.
  - But an instance could satisfy Y -> B, yet not be in this intersection.

44

Example



45