FUNCTIONAL DEPENDENCIES

Attributes are grouped together to form relations (tables)

Tables are put together to form a relational database schema

The above process is specified by the database designer or by mapping an ER/EER diagram to a relational schema

We need to be able to measure how "**good**" a particular grouping is, w.r.t. other possible groupings

The quality of a relational database schema is evaluated at two levels:

- the ***logical*** (conceptual) level (how clear is the meaning of the attributes ➔ easier to formulate queries)
- the ***implementation*** (storage) level (how the tuples are stored and updated ➔ more efficient query execution)

Database design can be done using 2 methodologies:

top-down (start with some relations and refine the decomposition until all requirements are met)

bottom-up (start with basic relationships between attributes and build up relations)

GENERAL DESIGN GUIDELINES

- A relation should correspond to a single entity type or a single relationship type
- A database should be designed so as to avoid update (insert, delete, modify) anomalies
- Limit the number of null values in tuples (problems with JOIN, COUNT, SUM etc)
- Design relations schemas that can be JOINed with equality conditions on either primary or foreign keys (spurious tuples)

Concept of FD

- ***Functional Dependency:*** a constraint between two sets of attributes from the database
- ***Universal relation schema:*** assemble all the attributes in one big relation $\mathbf{R=\{A_1, \ldots ,A_n\}}$ (theoretical concept)
- Used to define ***1**^st^**, 2**^nd^**, 3**^rd^ **, BCNF***

Definition of FD

- $X \rightarrow Y$ (X, Y subsets of **R**) specifies a limitation on the possible tuples in a potential relation state **r** of **R**
- For any tuples t_1, t_2 with $\mathbf{t_1}[X] = \mathbf{t_2}[X]$ we must also have $\mathbf{t_1}[Y] = \mathbf{t_2}[Y]$
- The values of the Y component of a tuple in **r are determined** by the values of the X component. The values of the X component **uniquely (functionally) determine** the values of the Y component

FD Terminology

- There is an FD from X to Y
- Y is functionally dependent on X
- X is the ***<u>left-hand side</u>*** of the FD
- Y is the ***<u>right-hand side</u>*** of the FD
- X **functionally determines** Y **in R** iff every two tuples that agree on their X-values, agree on their Y-values

Remarks on FDs

- If X is a candidate key for **R**, then $X \rightarrow Y$ for any subset Y of **R**
- $X \rightarrow Y$ in R, does not mean $Y \rightarrow X$ in **R**
- An FD is a semantic property of the attributes
- ***Legal relation:*** is a relation state that satisfies the specified FDs
- FDs specify constraints that must always hold

Examples of FDs

- *{Province, #DriverLicence}* → *SIN*
- In the **EMPLOYEEPROJECT** relation schema
  - *SIN* → *ENAME*
  - *PNUMBER* → *{PNAME, PLOCATION}*
  - *{SIN, PNUMBER}* → *HOURS*
- ***Diagrammatic Notation*** for FDs:

  FDs (horizontal lines) **lhs** (vertical lines) **rhs** (pointing arrows)

A subtle point about FDs

- An FD is a property of the relation schema, **not** of a particular relation state
- As a consequence, an FD cannot be inferred from a given relation state **r**
- However, it is sufficient to exhibit a counterexample to show that a certain FD cannot hold
- *TEXT* → *COURSE* **?** *COURSE* → *TEXT*

Inference Rules for FDs

- **F** denotes the set of all FDs in **R**
- Other FDs may be ***deduced*** from **F**
- $\mathbf{F}^+$ (the ***closure*** of **F**) is the set of all possible FDs deduced from **F**
- ***Example:*** F={*SIN*→{*ENAME, ADDRESS, BDATE, DNUMBER*}, *DNUMBER*→{*DNAME,DMGRSIN*} }
- We can ***infer*** the following $\mathbf{F}^+$ elements:
  SIN→{*DNAME,DMGRSIN*},*SIN*→*SIN, DNUMBER*→*DNAME*

- An FD $X \rightarrow Y$ is ***inferred from*** a set of FDs **F** on **R** if $X \rightarrow Y$ holds in every legal relation state **r** of **R** (if **r** satisfies all the FDs in **F**, then **r** satisfies $X \rightarrow Y$)
- A systematic way to establish FDs is provided by a set of ***inference rules*** that can be used to infer new FDs from given ones
- ***NOTATIONS:*** **F** $\models X \rightarrow Y$ & $\{X,Y\} \rightarrow Z$ $XY \rightarrow Z$

A SET OF INFERENCE RULES

- IR1 (reflexivity) Y subset of X, $X \rightarrow Y$
- IR2 (augmentation) $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
- IR3 (transitivity) $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$
- IR4 (decomposition) $\{X \rightarrow YZ\} \models X \rightarrow Y$
- IR5 (union) $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$
- IR6 (pseudo-transitivity)
  $\{X \rightarrow Y, WY \rightarrow Z\} \models \mathrm{W}X \rightarrow Z$

Comments on the Inference Rules

- Applying IR4 repeatedly we can decompose an FD $X \rightarrow \{A_1, A_2, \ldots, An\}$ into a set of FDs: $\{X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow An\}$
- Applying IR5 repeatedly we do the opposite
- IR1, …, IR6 can be proved from the definition **or** by contradiction **or** by using previously proved rules

Armstrong's Axioms

- {IR1, IR2, IR3} Armstrong's Axioms **AA**
- **AA** are ***sound*** (any FD inferred from **F** using AA will hold in any relation state **r** of **R**, that satisfies a given set of FDs **F**)
- **AA** are ***complete*** (we can compute the closure $\mathbf{F}^{+}$ of a given set of FDs **F**, using exclusively AA)

Usage of AA in database design

- Specify a set **F** of semantically obvious FDs on the attributes of **R**
- Use **AA** to infer additional FDs (2 steps)
  - Determine each set X of attributes that appear as lhs of FDs in **F**
  - Determine the set X^+ (closure of X under **F**) of all attributes functionally determined by X based on **F**

Algorithm to compute X^+

INPUT: a set of FDs $\mathbf{F}$, a set X of lhs of elements in $\mathbf{F}$

OUTPUT: the set of attributes X^+ (closure of X under $\mathbf{F}$)

STEP 1. Assign $X^+ := X$ (***justification: IR1)***

STEP 2. Repeat

old$X^+ := X^+$ **(for loop justification: IR3, IR4)**

for each FD $Y \rightarrow Z$ in $\mathbf{F}$ do

if Y subset of X^+ then $X^+ := X^+ \cup Z$

Until $X^+ =$ oldX^+

Example of application of the closure computation algorithm

- In the **EMPLOYEEPROJECT** relation schema
- **F** = $\{SIN \rightarrow ENAME, PNUMBER \rightarrow \{PNAME, PLOCATION\}, \{SIN, PNUMBER\} \rightarrow HOURS\}$
  - $\{SIN\}^+ = \{SIN, ENAME\}$
  - $\{PNUMBER\}^+ = \{PNUMBER, PNAME, PLOCATION\}$
  - $\{SIN, PNUMBER\}^+ = \{SIN, PNUMBER, ENAME, PNAME, PLOCATION, HOURS\}$

In general X^+ **U** Y^+ different than $(X$ **U** $Y)^+$

Equivalent sets of FDs

- Consider **E, F** two sets of FDs
  - **E** ***is covered by*** **F** (**F** ***covers*** **E**), **E** subset of $\mathbf{F}^+$
  - **E, F** ***equivalent***, $\mathbf{E}^+ = \mathbf{F}^+$ *in words:*
  - *Every FD in F can be deduced from E and vice-versa*
- Determine whether **F** covers **E**: ***a)*** compute X^+ wrt **F** for every FD $X \rightarrow Y$ in **E** ***b)*** check Y subset of X^+ ***c)*** if ***b)*** is true for every FD in **E**, then **F** covers **E**.
- Determine whether **E, F** are equivalent: **E** covers **F** and **F** covers **E**.

Minimal Sets of FDs

A set of FDs **F** is ***minimal*** if:

- The rhs of every FD in **F** is a single attribute
- If we replace a FD $X \rightarrow A$ with a FD $Y \rightarrow A$ where Y is a proper subset of X, we will obtain a set of FDs ***not equivalent*** to **F**
- If we remove a FD from **F**, we will obtain a set of FDs ***not equivalent*** to **F**

Minimal Set ➔ canonical form & no redundancies

A ***minimal cover*** of a set of FDs **F**, is a minimal set of FDs **G**, that is also equivalent to **F**

Algorithm to compute a minimal cover

INPUT: a set of FDs $\mathbf{F}$

OUTPUT: a minimal cover $\mathbf{G}$ of $\mathbf{F}$

STEP 1. Assign $\mathbf{G} := \mathbf{F}$

STEP 2. Replace each FD $X \rightarrow \{A_1, A_2, \ldots, An\}$ in $\mathbf{G}$

by n FDs $\{X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow An\}$

STEP 3. For each FD $X \rightarrow A$ in $\mathbf{G}$,

For each attribute **B** of X,

if ($\mathbf{G}$-$\{X \rightarrow A\}$) **U** {(X-{**B**})→A} equiv. to $\mathbf{G}$

then replace $X \rightarrow A$ by X-{**B**}$\rightarrow A$ in $\mathbf{G}$

STEP 4. For each remaining FD $X \rightarrow A$ in $\mathbf{G}$,

if $\mathbf{G}$-$\{X \rightarrow A\}$ is equiv. to $\mathbf{G}$

then remove $X \rightarrow A$ from $\mathbf{G}$

Normalization

- Use FDs to describe the semantics of relations schemas
- Assume that a set of FDs and a **primary key** are given for each relation
- Define 1st, 2nd, 3rd NF, BCNF (and higher NF)
- **Evaluate** each relation against each NF and **decompose** it (top-down design) in order to obtain relations that satisfy NF

<u>Normalization Process</u>

- Framework to analyze relations schemas based on their keys and FDs among attributes
- Normal Form tests carried out on relation schemas to <u>***normalize***</u> them to any desired degree

<u>***Normal Form of a relation:***</u>

the <u>***highest***</u> normal form condition satisfied by the relation

Other properties that a good relational schema must have: <u>***nonadditive join***</u> (spurious tuples), <u>***dependency preservation***</u>

Surprise Quiz !

- ***<u>Superkey</u>***
- ***<u>Key</u>***
- ***<u>Candidate Key</u>***
- ***<u>Primary Key</u>***
- ***<u>Foreign Key</u>***

<u>Prime attribute</u>

<u>Nonprime attribute</u>

Some of these definitions are useful in defining the 1^{st} , 2^{nd} , 3^{rd} NF introduced by Codd in 1972

member of some candidate key of the relation

First Normal Form 1NF

- **1NF** Atomicity of attribute domains
- Attribute values allowed by **1NF** are single indivisible atomic values from the attribute domain
- In particular, **1NF** forbids

(a) multi-valued attributes

(b) nested relations

(c) relations as values of tuples

Normalization into 1NF (I)

- When a relation is not in **1NF** there are 3 main techniques to ***normalize*** it using the attribute **A** that violates **1NF** (case **A** is a ***composite attribute***)
  - Remove **A** and place it in a new relation together with the primary key
  - Expand the key, so that there will be a separate tuple for each atomic value of **A** (pb: introduces redundancy)
  - If a max number of values **n**, is known for **A**, replace **A**, with **n** atomic attributes (pb: introduces null values)

Normalization into 1NF (II)

- When a relation is not in **1NF** there is one other technique to ***normalize*** it using the attribute **A** that violates **1NF** (case **A** is a ***multi-valued attribute***)
  - Remove **A** into a new relation and ***propagate*** the primary key into this new relation
  - The primary key of the new relation will combine the partial key of the nested relation and the primary key of the original relation
  - This process can be applied recursively to denest relations

Second Normal Form 2NF

- **2NF** FULL FD OF ALL NONPRIME ATTRIBUTES ON PRIMARY KEY
- ***Full FD:*** $X \rightarrow Y$ is a Full FD if removal of any attribute of X destroys the FD
- ***Partial FD:*** $X \rightarrow Y$ is a Partial FD if removal of some attribute of X does not destroy the FD
- If the primary key contains only one attribute then the relation satisfies the **2NF** criterion
- 2NF is concerned with FDs whose lhs attributes are parts of the primary key

Normalization into 2NF

- If a relation schema is not in 2NF, it can be ***normalized to 2NF***: decomposed into a number of relations in which nonprime attributes are fully functionally dependent on the primary key.
- 2NF normalization recipe:
  - Set up a new relation for each partial key with its dependent attribute(s)
  - Keep a relation with the original primary key and its fully functionally dependent attribute(s)

Third Normal Form 3NF

- **3NF** NO NONPRIME ATTRIBUTE IS TRANSITIVELY DEPENDENT ON THE PRIMARY KEY OF R
- ***<u>Transitive FD:</u>*** $X \rightarrow Y$ is a Transitive FD in R if there is a set of attributes Z (that is neither a candidate key nor a subset of any key of R) such that both $X \rightarrow Z$ and $Z \rightarrow Y$ are valid FDs.
- ***<u>Example:</u>*** *SIN*→*DMGRSIN* is a transitive FD (*Z* = *DNUMBER* not a key, neither a subset of the key)

Normalization into 3NF

- If a relation schema is not in 3NF, it can be ***normalized to 3NF***: decomposed into a number of relations in which no nonprime attributes are transitively dependent on the primary key.
- 3NF normalization recipe:
  - Set up a new relation for each nonprime attribute that is transitively dependent on the primary key

General Definitions of 2NF, 3NF

- We want to design relation schemas that do not contain neither ***partial*** nor ***transitive*** dependencies
- 2NF normalization disallows partial dependencies
- 3NF normalization disallows transitive dependencies
- These definitions of 2NF & 3NF take into account only the ***primary key*** and not the candidate keys
- The more general definitions of 2NF & 3NF take into account ***all*** candidate keys of a relation
- ***Prime attribute:*** part of any candidate key

General Definition of 2NF

- **R is in 2NF if every nonprime attribute is fully functionally dependent on every key of R**
- *Example:* LOTS relation with 2 candidate keys PROPERTY_ID# and {COUNTY_NAME, LOT#}
- LOTS violates 2NF because of FD3
- To normalize LOTS in 2NF we decompose it into 2 relations by removing the problematic nonprime attribute
- Both new relations are in 2NF

General Definition of 3NF

- **R is in 3NF if for every FD $X \rightarrow A$ either (a) X is a superkey of R or (b) A is a prime attribute of R**
- *Example:* LOTS2 is in 3NF, LOTS1 is not in 3NF because of FD4 (which gives rise to a transitive FD)
- To normalize LOTS1 in 3NF we decompose it into 2 relations by removing the problematic nonprime attribute together with the lhs of FD4

Boyce-Codd Normal Form BCNF

- Stronger requirement than 3NF
- Every relation in BCNF is also in 3NF, but not vice-versa
- ***Motivating Example:*** LOTS relation with FD1…FD4
  - Suppose there are 1000s of lots but from only two counties
  - (Suppose that lot sizes from county1 are 0.5, 0.6, 0.7 and lot sizes from county2 are 1.2, 1.5, 1.8, 2.1) → we have an additional FD FD5: *AREA→ COUNTY_NAME* however, 3NF is not violated
  - Since there are only 7 possible area values, FD5 could be represented in a separate relation **R(AREA,COUNTY_NAME)** to avoid repeating the same information the 1000s of tuples

BCNF

- **R is in BCNF if for every FD $X \rightarrow A$, X is a superkey of R**
- FD5 violates BCNF (AREA is not a superkey)
- To normalize LOTS1A in BCNF we decompose it onto two relations by removing the problematic FD