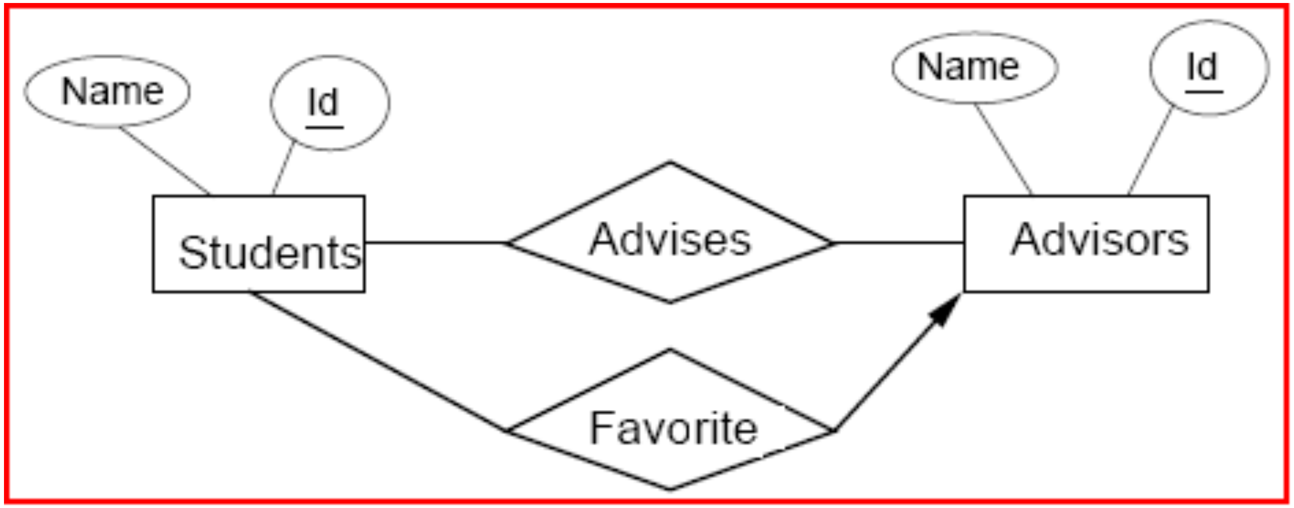
Functional Dependencies

Dr. Bing Zhou

Functional Dependencies:

1. A well developed design theory for relational database (what makes a good relational database schema)
2. are building blocks that enable the analysis of data redundancies, and the elimination of anomalies caused by them (through the process of normalization)
3. A generalization of the idea of a key for a relation

Example



- Convert to relations:
  - Students(Id, Name)
  - Advisors(Id, Name)
  - Advises(StudentId, AdvisorId)
  - Favorite(StudentId, AdvisorId)

- We perversely decide to convert Students, Advisors, and Favorite into one relation.
  - Students(Id, Name, AdvisorId, AdvisorName, FavoriteAdvisorId)

Example of a Bad Relation

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

- If you know a student's Id, can you determine the values of any other attributes?
  - Name and FavoriteAdvisorId.

$\boxed{\begin{array}{l} \texttt{Id} \longrightarrow \texttt{Name} \\ \texttt{Id} \longrightarrow \texttt{FavoriteAdvisorId} \end{array}}$ $\boxed{\texttt{AdvisorId} \longrightarrow \texttt{AdvisorName}}$

- Can we say Id → AdvisorId?
  - NO! Id is not a key.
- What is the key for the Students?
  - {Id, AdvisorId}
- Why is this relation "bad"?
  - Parts of the key determine other attributes.

Motivation for Functional Dependencies

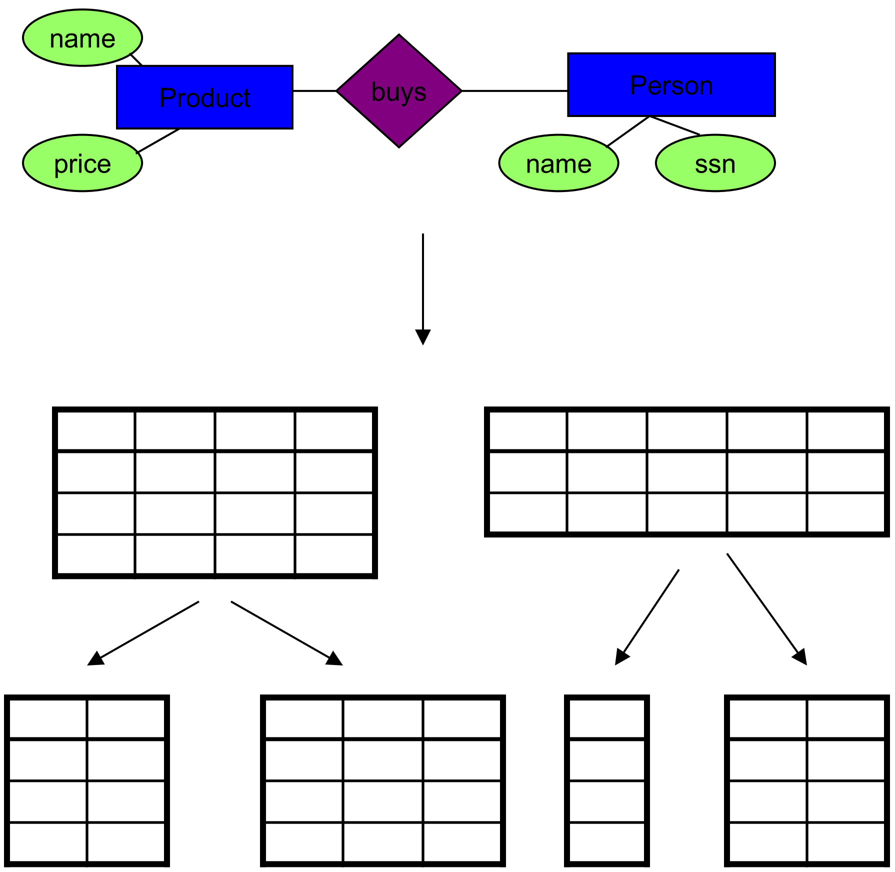
- Reason about constraints on attributes in relational designs.
- Procedurally determine the keys of a relation.
- Detect when a relation has redundant information.
- Improve database designs systematically using normalization.

Relational Schema Design

Conceptual Model:

Relational Model:
plus FD's

Normalization:
Eliminates anomalies



Definition of Functional Dependency

- 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*

$$\boxed{\text{if } t_{\texttt{AdvisorId}} = u_{\texttt{AdvisorId}}, \text{ then } t_{\texttt{AdvisorName}} = u_{\texttt{AdvisorName}}}$$

Definition of Functional Dependency

- *X* → *A* is an assertion about a relation *R* that whenever two tuples of *R* agree on all the attributes of *X*, then they must also agree on the attribute *A*.
  - Say "*X* → *A* holds in *R*."
- A *functional dependency* (FD) on a relation R is a statement
  - If two tuples in R agree on attributes $A_1, A_2, \ldots, A_n$ then they agree on attribute *B*.
  - Notation: $A_1 A_2 \ldots A_n \rightarrow B$

- FD says that for every pair of tuples t and u in any instance of R, if $t_{A_1} = u_{A_1}$ and $t_{A_2} = u_{A_2}$ and $\ldots t_{A_n} = u_{A_n}$, then $t_B = u_B$.
- The set of attributes $A_1, A_2, \ldots A_n$ *functionally determine* B.
- An FD is a constraint on a single relation schema. It must hold on every instance of the relation.
- You cannot deduce an FD from a relation instance.

Functional Dependency ?

- A **functional dependency** is a constraint between two sets of attributes in a relation

- An attribute or set of attributes X is said to **functionally determine** another attribute Y (written X → Y) if and only if each X value is associated with at most one Y value. Customarily we call X determinant set and Y a dependent set.

- So if we are given the value of X we can determine the value of Y.

Examples of FDs

- What FDs can we assert for the relation

`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 $\rightarrow$ CourseName

Number DeptName $\rightarrow$ Classroom

Number DeptName $\rightarrow$ Enrollment

Number DeptName $\rightarrow$ CourseName Classroom Enrollment

- Is *Number → Enrollment* an FD?

Example

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

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

Reasonable FD's to assert:

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

Example

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

name -> addr

name -> favBeer

beersLiked -> manf

FDs With Multiple Attributes

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

Use of Functional Dependencies

- We use functional dependencies to:
  - test relations to see if they are legal under a given set of functional dependencies.
    - If a relation *R* is legal under a set *F* of functional dependencies, we say that *R* satisfies *F.*
  - specify constraints on the set of legal relations
    - We say that *F* holds on *R* if all legal relations on *R* satisfy the set of functional dependencies *F.*

Keys of Relations

- FDs allow us to formally define keys.
- A set of attributes $\{A_1, A_2, \ldots A_n\}$ is a *key* for a relation R if

Uniqueness $\{A_1, A_2, \ldots A_n\}$ functionally determine all the other attributes of R and

Minimality no proper subset of $\{A_1, A_2, \ldots A_n\}$ functionally determines all the other attributes of R.

- A *superkey* is a set of attributes that has the uniqueness property but is not necessarily minimal.

Example

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

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

- {name, beersLiked} is a key because together these attributes determine all the other attributes.
  - name -> addr favBeer
  - beersLiked -> manf
- In this example, there are no other keys, but lots of superkeys.
  - Any superset of {name, beersLiked}.

Example of Keys

- What is the key for
  - Courses(Number, DeptName, CourseName, Classroom, Enrollment)?
- The key is {Number, DeptName}.
  - These attributes functionally determine every other attribute.
  - No proper subset of {Number, DeptName} has this property.

- What is the key for
  - Teach(Number, DepartmentName, ProfessorName, Classroom)?
- The key is {Number, DepartmentName}.
  - Why?

Keys in the Conversion from E/R to Relational Designs

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

Keys in the Conversion from E/R to Relational Designs

- 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.

Keys in the Conversion from E/R to Relational Designs

- If the relationship R is multi-way, we need to reason about the FDs that R satisfies.
  - There is no simple rule.
  - If R has an arrow towards entity set E, at least one key for the relation for R excludes the key for E.

FD’s From “Physics”

- While most FD’s come from E/R keyness and many-one relationships, some are really physical laws.

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

Example

- Branch

branchname	loan	customer	amount
Mall St	*17*	*Jones*	*1000*
Logan	*23*	*Smith*	*2000*
Queen	*15*	*Hayes*	*1500*
Mall St	*14*	*Jackson*	*1500*
King George	*93*	*Curry*	*500*
Queen	*25*	*Glenn*	*2500*
Andrew	*10*	*Brooks*	*2500*
Logan	*30*	*Johnson*	*750*

- Is Loan → Customer a valid FD ?
  - Loan→Amount?
  - Loan→Branchname?
  - Loan→Customer Branchname Amount?
  - Loan Branchname →Amount?

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c2
a2	b1	c3

- A → B
- C → B

Student(SSN, sName, address,
HScode, HSname, HScity, GPA, priority)

$\underline{123}$
$\underline{123}$

SSN → sName

SSN → address ←

HScode → HSname, HScity

HSname, HScity → HScode

SSN → GPA

GPA → priority

SSN → priority

- Consider relation obtained from Hourly_Emps:
  - Hourly_Emps (*ssn*, *name*, *lot*, *rating*, *hrly_wages*, *hrs_worked*)
- We will denote this relation schema by listing the attributes: SNLRWH
  - This is really the *set* of attributes {S,N,L,R,W,H}.
- Some FDs on Hourly_Emps:
  - *ssn* is the key: $S \rightarrow SNLRWH$
  - *rating* determines *hrly_wages*: $R \rightarrow W$

Rules for Manipulating FDs

- Learn how to reason about FDs.
- Define rules for deriving new FDs from a given set of FDs.
- Next class: use these rules to remove “anomalies” from relational designs.
- Example: a relation R with attributes A, B, and C, satisfies the FDs $A \rightarrow B$ and $B \rightarrow C$. What other FDs does it satisfy?

$$A \rightarrow C$$

- What is the key for R ?
  - A, because $A \rightarrow B$ and $A \rightarrow C$

Equivalence of FDs

- An FD F follows from a set of FDs T if every relation instance that satisfies all the FDs in T also satisfies F.
- $A \rightarrow C$ follows from $T = \{A \rightarrow B, B \rightarrow C\}$

- Two sets of FDs S and T are equivalent if each FD in S follows from T and each FD in T follows from S.
- $S = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ and $T = \{A \rightarrow B, B \rightarrow C\}$ are equivalent.

- These notions are useful in deriving new FDs from a given set of FDs.

Inference Rules for FDs

$$A_1, A_2, \ldots A_n \longrightarrow B_1, B_2, \ldots B_m$$

Is equivalent to

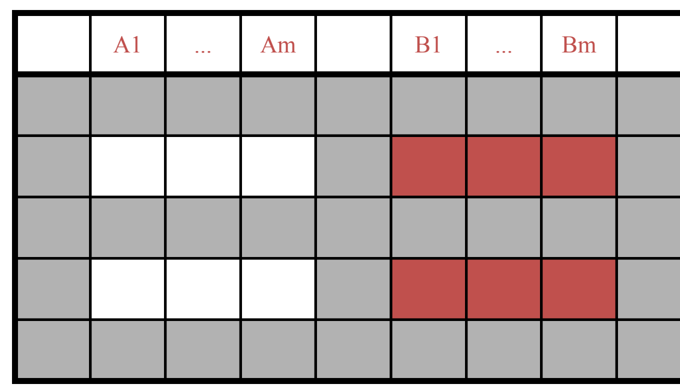
$$A_1, A_2, \ldots A_n \longrightarrow B_1$$

$$A_1, A_2, \ldots A_n \longrightarrow B_2$$

$$\ldots$$

$$A_1, A_2, \ldots A_n \longrightarrow B_m$$

Splitting rule and Combing rule



Splitting and Combining FDs

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

> ▶ For the relation Courses is the FD
> $\texttt{Number DeptName} \longrightarrow \texttt{CourseName}$
> equivalent to the set of FDs
> $\{\texttt{Number} \longrightarrow \texttt{CourseName}, \texttt{DeptName} \longrightarrow \texttt{CourseName}\}$?

  – **No !**

Triviality of FDs

An FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is

- *trivial* if the B's are a subset of the A's,
  $\{B_1, B_2, \ldots B_n\} \subseteq \{A_1, A_2, \ldots A_n\}$
- *non-trivial* if at least one B is not among the A's,
  $\{B_1, B_2, \ldots B_n\} - \{A_1, A_2, \ldots A_n\} \neq \emptyset$
- *completely non-trivial* if none of the B's are among the A's, i.e.,
  $\{B_1, B_2, \ldots B_n\} \cap \{A_1, A_2, \ldots A_n\} = \emptyset$.

- What good are trivial and non-trivial dependencies?
  - Trivial dependencies are always true.
  - They help simplify reasoning about FDs.