Database Design Theory

- Overview of Database Design Process
- Functional Dependencies and Normalization
  - Functional dependencies (FDs)
  - redundancy and update anomalies
  - third normal form (3NF) and Boyce-Codd normal form (BCNF)
  - design algorithms for 3NF and BCNF
- Multivalued Dependencies
  - MVDs
  - fourth normal form (4NF)
  - design algorithms
- Desirable Properties of Relational Database Schemes

Overview of database design process

- Database management system
  - the key to the successful management of business and institutions
    - → computerized functions
      - ☞ mission critical systems
    - → consolidation of information resources
    - → decision support system and executive information system
- Typical phases of the database design process
  - requirement analyses
  - conceptual design
  - physical design
  - system implementation

- Requirement Analysis
  - Identify the major application areas and user groups
  - analyze the existing documents, policies,
  - study the current operating environments, information flow, and data transactions
  - collect the written response from potential users
- Conceptual Design
  - complete understanding of database structure, semantics, interrelationships and constrains
  - based on high-level data models, such as E-R model

Logic Database Design

- System independent phase
  - obtain a desirable database scheme in the database model of the chosen database management system
- System dependent phase
  - adjust the database scheme obtained in the previous phase to conform to the chose database management system
  - DDL statements

Physical Database Design

- Purpose
  - to specify the appropriate file structures and indexes
- Criteria
  - efficiency
- Approach
  - analyzing the database queries and transactions, including expected frequency
  - specifying the general user requirements
- Guideline
  - speeding natural join operations
  - separate read-only and update transactions
  - index files for search and hashing for random access
  - focus on attributes used most frequently

Implementation

- Coding
  - DDL for database scheme
  - SDL for physical scheme
  - develop application programs
- Testing
- Operation and Maintenance

Bad Database Design

- Pitfalls in Relational Database Design
  - Repetition of information
  - Inability to represent certain information
  - Loss of information

Consider the following relation schemes:

Branch = (branch-name, assets, branch-city)
Borrow = (branch-name, loan-number, customer-name, amount)
Deposit = (branch-name, account-number, customer-name, amount)

Repetition of information

Consider an alternative design with the single scheme below

Lending = (branch-name, assets, branch-city, loan-number, customer-name, amount)

branch-name	assets	branch-city	loan-number	customer-name	amount
Downtown	9000	Edmonton	17	Jones	1000
Downtown	9000	Edmonton	93	Smith	2000
Downtown	9000	Edmonton	93	Hays	2900
Redwood	21000	Edmonton	23	Jackson	1200
Redwood	21000	Edmonton	23	Smith	2000
SUB	17000	Edmonton	19	Hays	2900
SUB	17000	Edmonton	19	Turner	500
SUB	17000	Edmonton	19	Brooks	2200

Representation of Information

Consider another scheme below:

BD-scheme =
(branch-name, customer-name loan-number, amount, account-number, balance)

What if a customer wishes to open an account but not a loan ?

Solution: (1) Sorry, unless you are going to borrow money from us, we cannot let you open an account.

or

(2) Use null values.

Loss of information

Consider yet another alternative design in which Borrow is decomposed into two schemes, as follows:

Amt-scheme = (customer-name, amount)
Loan-scheme = (branch-name, loan-number, amount)

Let Amt = $\Pi_{customer-name, amount}(borrow)$

Loan = $\Pi_{branch-name, loan-number, amount}(borrow)$

? Amt $\bowtie$ Loan = borrow

Another example

R(A, B, C, D)

A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d1
a3	b1	c1	d2
a4	b2	c2	d3

R1(B, C)

B	C
b1	c1
b2	c2

R2 = (A, D)

A	D
a1	d1
a2	d1
a3	d2
a4	d3

R1 ⋈ R2

A	B	C	D
a1	b1	c1	d1
a2	b1	c2	d1
a1	b2	c2	d1
a2	b2	c2	d1
a3	b1	c1	d2
a4	b2	c2	d3

Informal Design Guidelines

- Avoid information repetition
  - redundancy
  - update anomalies
- Avoid null values as much as possible
  - difficulties with interpretation
    - → don’t know, don’t care, known but unavailable, does not apply
- Preserve information
  - avoid spurious joins

Normalization Using Functional Dependencies

Integrity constraints:

Let R be a relation scheme and IC be a set of rules, called integrity constraints. A relation instance **r** is said to be legal under IC if **r** satisfies all rules in IC.

For examples:

(1) any customer has only one address
(2) account number shall be the key of Account.
(3) the grades of students in C391 should be independent on the salary level of instructors.

Motivation Example

StudentCourse = (Student_name, Address, Course, Grade)

Student = (Student_name, Address)
Course = (Student_name, Course, Grade)

? StudentCourse = Student ⋈ Course

Student_name	Address	Course	Grade
Tom	101 Hub	391	A
Sarah	202 Hub	291	B
Mike	303 Hub	304	D
Chris	404 Hub	391	A
Chris	**505 Hub**	**291**	**B**

Functional Dependencies

- Functional dependencies (FDs)

  Let R be a relation scheme, and $X \subseteq R$ and $Y \subseteq R$ be sets of attributes.Then the functional dependency

  $$X \rightarrow Y$$

  holds on R if in any legal relation r, for all pairs of tuples t1 and t2 in r

  $$t_1[X] = t_2[X] \quad \Rightarrow \quad t_1[Y] = t_2[Y].$$

Example: Student# $\rightarrow$ StudentName
BranchName $\rightarrow$ Assets
Key $\rightarrow$ AllAttributes

Example

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d2
a2	b2	c2	d2
a2	b3	c2	d3
a3	b3	c2	d4

?

$AB \rightarrow C$
$AB \rightarrow D$
$AB \rightarrow CD$
$CD \rightarrow AB$
$A \rightarrow C$
$C \rightarrow A$

$A \rightarrow D$
$BD \rightarrow A$
$BD \rightarrow C$
$ABC \rightarrow B$

- We shall use FDs to test relations to see if they are legal under a given set of functional dependencies.
  - We say **r** satisfies a set F of FDs if **r** is legal under F
- We shall use FDs to specify constraints on the set of legal relations.
  - thus, we concern ourselves only with relations that satisfy a given set of FDs.

Functional dependencies allow us to express certain constraints that cannot be expressed using key constraints

Functional dependencies represent value-relationships between attributes.

- Assume F is a set of FDs. We say
  - $F \models X \rightarrow Y$

  if for any relation **r** that satisfies F, **r** satisfies $X \rightarrow Y$.

For examples, $\Phi \models AB \rightarrow A$

$\{ X \rightarrow Y \} \models XA \rightarrow Y$

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

The **closure of F**, denoted by F^+, is defined as

$\{ X \rightarrow Y \mid F \models X \rightarrow Y \}$

The **closure of X** with respect to F, denoted as X^+, is defined as

$\{A \mid F \models X \rightarrow A \}.$

Armstrong's Axioms

- Reflexivity rule
  - If $Y \subseteq X$ then $X \rightarrow Y$.
- Augmentation rule
  - If $X \rightarrow Y$ then $XW \rightarrow Y$.
- Transitivity rule
  - If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$.

The Armstrong's axioms are sound and complete wrt FDs in that

(1) if $X \rightarrow Y$ can be derived from F by the above three inference rules then $F \models X \rightarrow Y$; and

(2) if $F \models X \rightarrow Y$ then $X \rightarrow Y$ can be derived using the above three inference rules.

three derived rules

- Union rule
  - If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$
- Decomposition rule
  - If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$
- Pseudotransitivity rule
  - If $X \rightarrow Y$ and $WY \rightarrow Z$ then $XW \rightarrow YZ$.

Example: $F = \{A \rightarrow B,\ A \rightarrow C,\ CG \rightarrow H,\ CG \rightarrow I,\ B \rightarrow H\}$

We list some members of F^+ below:

$A \rightarrow H, \quad CG \rightarrow HI, \quad AG \rightarrow I$

How to determine if F |== $X \rightarrow W$

Algorithm

Input: a set F of FDs
$X \rightarrow W$

Output: Yes if F |== $X \rightarrow W$, and
No if otherwise

Method: (compute X^+ with respect to F)

1. Result:= X;
2. **while** (changes to Result) **do**
   for each FD $Y \rightarrow Z$ in F **do**
   if $Y \subseteq$ Result **then** Result := Result $\cup$ Z;
3. **if** $W \subseteq$ Result **then** return Yes
   else return No;

Example: Compute AG^+ with respect to F in the previous slide.

Motivation Examples for Minimal Covers

Consider the following sets of FDs

F1 = { A $\rightarrow$ B, A $\rightarrow$ C, A $\rightarrow$ BC }, F2 = { A $\rightarrow$ B , A $\rightarrow$ C }

F3 = { A $\rightarrow$ B, B $\rightarrow$ C, A $\rightarrow$ C }, F4 = { A $\rightarrow$ B , B $\rightarrow$ C }

F5 = { AB $\rightarrow$ C , A $\rightarrow$ B }, F6 = { A $\rightarrow$ C, A $\rightarrow$ B }

F7 = { A $\rightarrow$ BC }, F8 = { A $\rightarrow$ B, A $\rightarrow$ C }

Obviously, F1 <=> F2, F3 <=>F4, F5 <=> F6, and F7 <=> F8.

Which one is a better set to characterize the FDs ?

- Cover
  - F is said to be a cover of E if $F \models E$ and $E \models F$
- Minimal
  - F is said to be minimal if
    - → every FD in F is of the form $X \rightarrow A$
    - → $F' \not\models F$ for any $F' \subset F$,
    - → for any $X \rightarrow A$ in F and any $X' \subset X$,
      - ☞ $(F \setminus \{X \rightarrow A\}) \cup \{X' \rightarrow A\}) \not\models F$
- F is said be a minimal cover of E if
  - F is a cover of E and
  - F is minimal

Algorithm for computing the minimal cover

Input: F: set of FDs
Output: G: a minimal cover of F
Method

1. FOR each $X \rightarrow A_1, \ldots A_n$ in F DO {* right reducing *}
   Add $X \rightarrow A_i$ into G;
2. FOR each $X \rightarrow A$ in G DO
   FOR each B in X DO
   IF $G \models (X \setminus B) \rightarrow A$ Then
   replace $X \rightarrow A$ with $(X \setminus B) \rightarrow A$; {* left reducing *}
3. FOR each $X \rightarrow A$ in G DO
   Delete $X \rightarrow A$ from G if $G \setminus \{X \rightarrow A\} \models G$;
   {* eliminate redundancy*}

(We should do left-reducing before the eliminating.)

Consider R = ABCDEGHI and F contains the following FDs

$A \rightarrow BD$

$C \rightarrow D$

$AD \rightarrow CE$

$DC \rightarrow H$

Find a minimal cover of F

FM = { $A \rightarrow B$, $C \rightarrow D$, $A \rightarrow C$, $A \rightarrow E$, $C \rightarrow H$ }

Desirable Properties of Decomposition

- Join loss-less decomposition
  - a necessary requirement
- Dependence preservation
- Minimizing redundancy
  - Boyce-Codd normal form
  - third normal form
- .

- Join loss-less decomposition

  Let U be a set of attributes, IC a set of constraints, and
  D = { R1, R2, ..., Rn} a set of relation schemes such that
  R1 ∪ R2 ∪ ... ∪ Rn = U.

  Then D is said to be a join loss-less decomposition of U wrt IC
  if for any legal relation **u** over U under IC, we have

$$u = \Pi_{R_1}(u) \bowtie \Pi_{R_2}(u) \bowtie \cdots \bowtie \Pi_{R_n}(u)$$

  Join loss-less is a necessary requirement for the decomposition.

Theorem 1

Let R be a relation scheme and F a set of FDs. Then

$\{R_1, R_2\}$ is a join loss - less decomposition of R wrt. F

if either $R_1 \cap R_2 \rightarrow R_1$

or $R_1 \cap R_2 \rightarrow R_2$.

Theorem 2

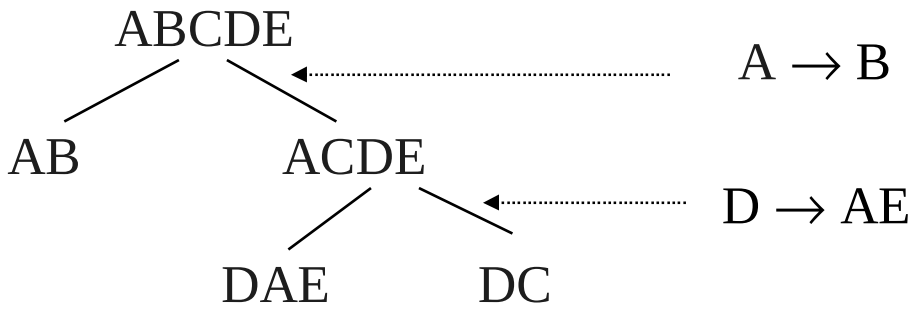
If D = { R1, ..., Ri, ..., Rn} is join loss-less wrt F and
{Q1,..., Qm} is a joint loss-less decomposition of Ri. Then

{R1, ..., Q1,..., Qm, ..., Rn} is also join loss-less wrt F.

Example 1 Consider U = ABCD and F = { A → B, C → A }.
D = { AB, CA, CD } is a join loss-less decomposition of U.

Example 2 Consider U = ABCDE and F = { A → B, D → AE }. Then

D = { AB, DAE, DC} is a joint loss-less decomposition, because



- Dependency Preservation
  - Let U be a set of attributes, F a set of FDs, and D = { R1, R2, ..., Rn} a decomposition of U. Then D is said to be a dependency preserving decomposition of U wrt F if there exists a set G of FDs such that
    - → F ≡ G, and
    - → for each X → W in G, there exists Ri in D such that XW ⊆ Ri.

Dependency preservation is desirable because we wish the constraints can be easily checked.

An alternative definition for dependency preserving decomposition

1. A relation scheme R **preserves** an FD $X \rightarrow W$ if $XW \subseteq R$.

2. A database scheme $D = \{ R_1, ..., R_i, ..., R_n \}$ **preserves** an FD $X \rightarrow W$ if there exits R_i in D such that R_i preserves $X \rightarrow W$.

3. A database scheme $D = \{ R_1, ..., R_i, ..., R_n \}$ **preserves** a set F of FDs if D preserves every FD in F.

4. A database scheme $D = \{ R_1, ..., R_i, ..., R_n \}$ is said to be a **dependency preserving decomposition** of $\bigcup_{i=1}^{n} R_i$ wrt a set F of FDs if

there exists a set E of FDs such that
   (a) E is a cover of F, and
   (b) D preserves E.

- Minimizing redundancy
  - redundancy
  - update anomalies

- Let R be a relation scheme and $X \rightarrow A$ holds in R. Then $X \rightarrow A$ is said to be
  - a partial dependency if X is a proper subset of a candidate key of R
  - a transitive dependency if X is not a subset of any candidate key of R. That is, there exists a key K of R such that $K \rightarrow X$ and $X \rightarrow A$.

(Sometimes, we say $K \rightarrow A$ is a transitive dependency.)

Boyce-Codd Normal Form

- A relation scheme R is said to be in Boyce-Codd normal form (BCNF) if for any non-trivial FD $X \rightarrow A$ which holds in R, X is a key of R, that is, $X \rightarrow A$ holds in R.
  - no partial redundancy
  - no transitive redundancy
- Let U be a set of attributes, F be a set of FDs, and D = {R1, ..., Rn} be a decomposition of U. Then D is said to be a BCNF decomposition of U with respect to F if
  - R is a join loss-less decomposition of U wrt F, and
  - every relation scheme Ri in D is in BCNF wrt F.

- Example

  Regist (Course#, Student#, Grade, Address, Phone)

  is not in BCNF since

  Student# $\rightarrow$ Address holds but Student # is not a key

  Course (Course#, Prof, Office, Phone)

  not in BCNF because Prof $\rightarrow$ Office holds but Prof is not a key

But

{ (Course#, Student#, Grade), (Student#, Adress, Phone)} is a BCNF decomposition of Regist.

{ (Course#, Porf), (Prof, Office, Phone)} is a BCNF decomposition of Course.

- Let U be a set of attributes and F a set of FDs.
  - Is it possible to find a BCNF decomposition of U ?
  - Is is possible to find a dependency preserving and BCNF decomposition of U ?
  - How ?

Third Normal Form

- A relation scheme R is said to be in third normal form (3NF) with respect to a set F of FDs if for any nontrivial FD $X \rightarrow A$ which holds in R either
  - X is a key of R, i.e., $X \rightarrow R$ holds, or
  - A is a prime attribute, i.e., an attribute contained in a candidate key for R.
- A relation scheme in 3NF may still contain partial and transitive redundancy. However, whenever $X \rightarrow A$ is a partial/transitive dependency A is a prime attribute.
  - Prime attributes vs. redundancy

    For example, consider $F = \{ A \rightarrow BCD, CD \rightarrow AB, B \rightarrow C \}$ and $R = ABCD$.

    Then R is not in BCNF since $B \rightarrow C$ is a transitive dependency.
    R, however, is in 3NF with respect to F because C is a prime attribute.
    Note that CD is a candidate key of R.

- Let U be a set of attributes, F a set of FDs, and D = {R1, ..., Rn} a decomposition of U. Then D is said to be a 3NF decomposition of U with respect to F if
  - D is a join loss-less decomposition of U wrt F, and
  - every Ri in D is in 3NF wrt F.

Comparison of BCNF and 4NF

- Best goal
  - → join loss-less
  - → dependency preserving
  - → BCNF
- Better goal
  - → join loss-less
  - → dependency preserving
  - → 3NF
- Alternative
  - → join loss-less
  - → BCNF

Algorithm for BCNF decomposition

Input U: a set of attributes
F: a set of FDs
Output D: a BCNF decomposition of U wrt F
Method

(1) D = {U};
(2) **while** there exists a relation scheme Q in D that is not in BCNF **do**
begin
find a nontrivial FD $X \rightarrow W$ that violates BCNF, i.e.,
$X \rightarrow W$ in F^+ and $XW \subseteq Q$ and $X \not\rightarrow Q$;
$X^* := \{ A \mid A \text{ is in } (Q - X) \text{ and } F \models X \rightarrow A \}$;
replace Q in D by two schemes $(X \cup X^*)$ and $X(Q - X^*)$
end;

Note that it is NP-complete to determine whether a relation scheme is in BCNF wrt F.

Example Let us consider the relation scheme CTHRSG, where C = Course, T= Teacher, H=Hour, R=Room, S=Student, and G = Grade. The functional dependencies F are

$C \rightarrow T$

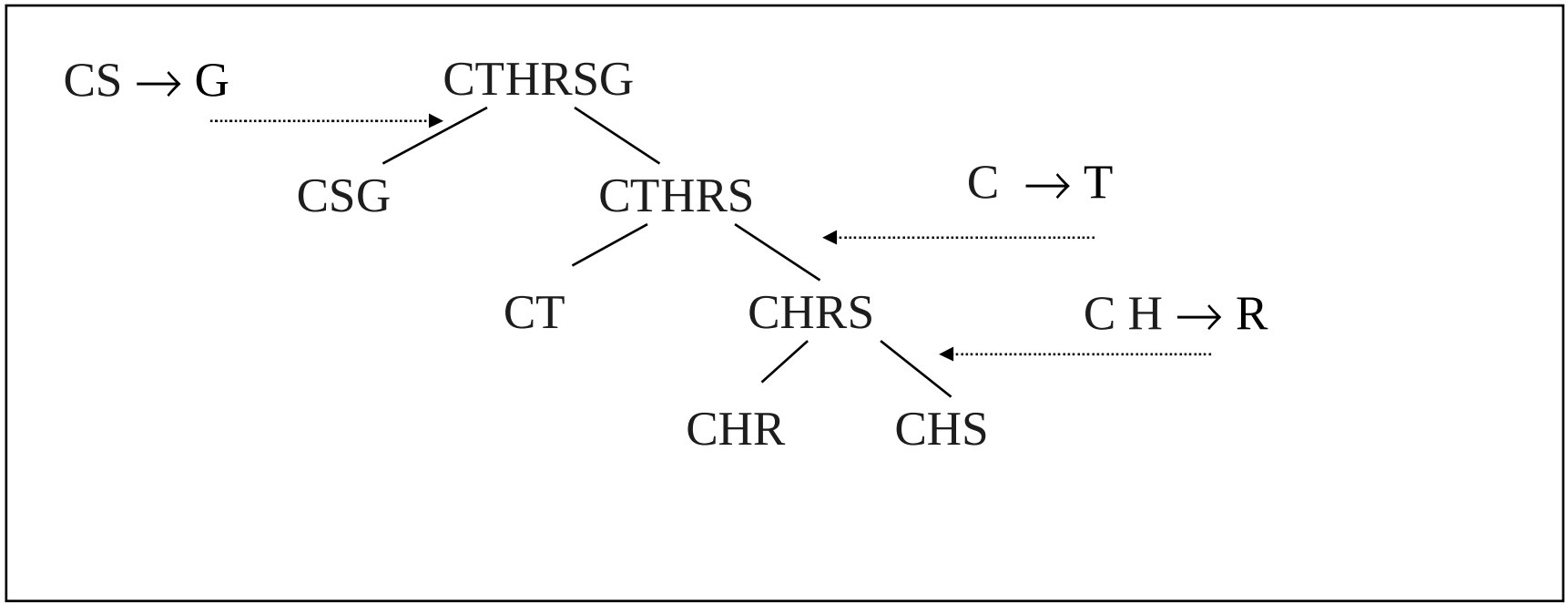
$HR \rightarrow C$

$HT \rightarrow R$

$CS \rightarrow G$

$HS \rightarrow R.$

The only key for CTHRSG is HS.



D_1 = { CSG, CT, CHR, CHS }.

Not a bad design:

CSG: course registration with grades

CT: the teacher for each course

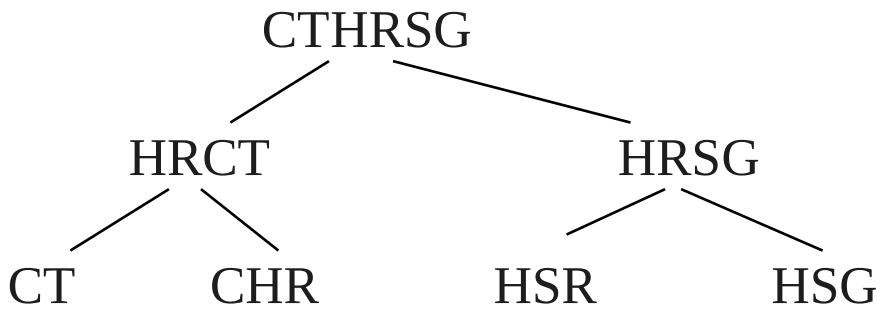
CHR: the hours at which each course meets and the room for each hour

CHS: the schedule of courses and hours for each student

Is D_1 dependency preserving ?

Is CHS really needed ?

How about the decomposition below ?



Algorithm for 3NF and Dependency Preserving Decomposition

Input U: a set of attributes
F: a set of FDs
Output D: a dependency preserving and 3NF decomposition of U wrt F
Method

1. find a minimal cover G of F;
2. **for** each left-hand side X that appears in G
   create a relation scheme $\{X, A_1, ..., A_n\}$
   where $X \rightarrow A_i$ are all the FDs in G with X as left hand side ;
3. **if** none of the relation scheme created in step 2 contains a key of U
   create one more relation scheme that contains only a candidate key of U.

Example Let us consider the relation scheme CTHRSG, where C = Course, T= Teacher, H=Hour, R=Room, S=Student, and G = Grade. The functional dependencies F are

$C \rightarrow T$

$HR \rightarrow C$

$HT \rightarrow R$

$CS \rightarrow G$

$HS \rightarrow R$.

The only key for CTHRSG is HS, and F is minimal.

Therefore, we have D = { CT, CHR, HRT, CSG, HRS }.

Normalization Using MVDs

- Multivalued dependencies (MVDs)
  - specify constraints among independent data items
  - can be used for further eliminating redundancy
- Inference rules for both FDs and MVDs
- Fourth Normal Form
  - relation scheme without non-trivial MVDs
  - design algorithm

Example Let us consider the relation scheme CTHRSG, where C = Course, T= Teacher, H=Hour, R=Room, S=Student, and G = Grade. The functional dependencies F are

$C \rightarrow T$

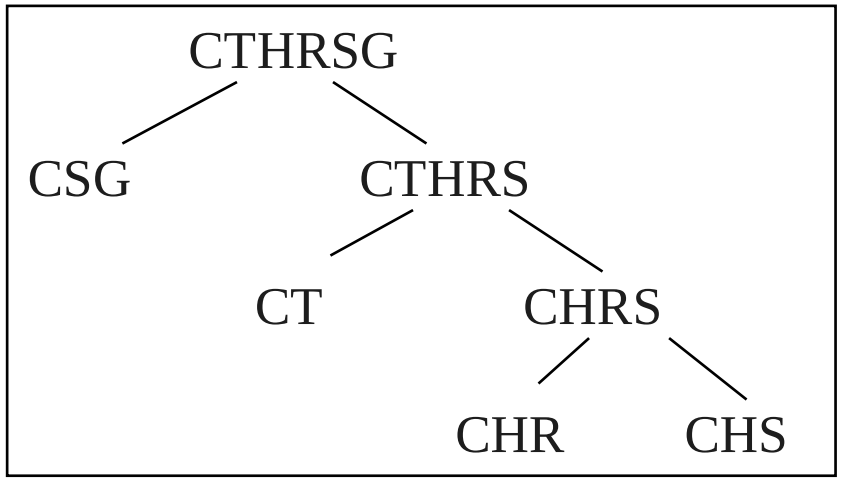
$HR \rightarrow C$

$HT \rightarrow R$

$CS \rightarrow G$

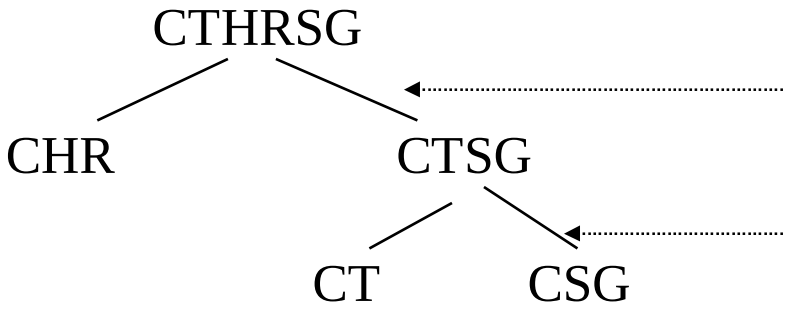
$HS \rightarrow R.$

The only key for CTHRSG is HS.



Is CHS really needed ?

What if we can decompose CTHRSG in the following way ?



not a lossless decomposition wrt F

A loss-less decomposition wrt F

We know that CHR and CTSG are relatively independent and therefore, the decomposition seems join lossless, though it cannot be proved under given F.

A sample relation

C	T	H	R	S	G
c391	Yuan	M10	V112	Peter	8
c391	Yuan	W10	V111	Peter	8
c391	Yuan	F10	V112	Peter	8
c391	Yuan	M10	V112	Sharon	9
c391	Yuan	W10	V111	Sharon	9
c391	Yuan	F10	v112	Sharon	9

Example Consider Faculty (Prof, Course, GraduatesStudent).

We assume a prof may teach upto 8 courses and each course may be taught by more than one instructor. Further, a prof may supervise any number of graduate students, and co-supervisors are common.

Therefore, there are no non-trivial FDs which hold in Faculty.
Consequently, Faculty is in BCNF.

We know it is much better to have two relations for this example:

Teach (Prof. Course)

Supervise (Prof. GraduateStudent)

Multivalued Dependencies (MVDs)

Let R be a relation scheme and $X \subseteq R$ and $Y \subseteq R$ be sets of attributes. Then the Multivalued dependency

$\mathbf{X \rightarrow\rightarrow Y}$ holds on R if in any legal relation **r** of R,

for all pairs of tuples t_1 and t_2 in r such that $t_1[X] = t_2[X]$, there exists tuples t_3 and t_4 in **r** such that:

1. $t_1[X] = t_2[X] = t_3[X] = t_4[X]$
2. $t_3[Y] = t_1[Y]$ and $t_3[R–Y] = t_2[R–Y]$
3. $t_4[Y] = t_2[Y]$ and $t_4[R–Y] = t_1[R–Y]$

Example

1. For the example of CTHRSG, we have
   $C \rightarrow\rightarrow HR$; $C \rightarrow\rightarrow T$; $C \rightarrow\rightarrow SG$

2. For Faculty = { Prof, Course, GraduateStudent }
   Prof $\rightarrow\rightarrow$ Course; Prof $\rightarrow\rightarrow$ GraduateStudent

3. Consider Bank = { Customer, Account, Balance, Loan, Amount}
   Customer $\rightarrow\rightarrow$ Account, Balance
   Customer $\rightarrow\rightarrow$ Loan, Amount

4. Consider Employee (Name, Project, Dependent)
   Name $\rightarrow\rightarrow$ Project; Name $\rightarrow\rightarrow$ Dependent

- Understanding MVDs
  - $X \rightarrow\rightarrow Y$ means the values of Y is independent on the values of all attributes other than XY
  - $X \rightarrow\rightarrow Y$ can be viewed as $X \rightarrow \{Y\}$, where {Y} denoted the set value of all Y's.
    - → Note that the set value is not allowed in first normal form.

Inference rules for MVDs and FDs

- Reflexivity for FDs
  - If $Y \subseteq X$ then $X \rightarrow Y$.
- Augmentation rule for FDs
  - If $X \rightarrow Y$ then $XW \rightarrow Y$.
- Transitivity rule for FDs
  - If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$.
- Complementation rule for MVDs
  - If $X \twoheadrightarrow Y$ then $X \twoheadrightarrow (R - XY)$
- Augmentation for MVDs
  - If $X \twoheadrightarrow Y$ and $V \subseteq W$ then $WX \twoheadrightarrow VY$.
- Transitivity rule for MVDs
  - If $X \twoheadrightarrow Y$ and $Y \twoheadrightarrow Z$ then $X \twoheadrightarrow (Z - Y)$.
- rules for both FDs and MVDs
  - If $X \rightarrow Y$ then $X \twoheadrightarrow Y$.
  - If $X \twoheadrightarrow Y$ and there exits W such that $W \cap Y = \emptyset$ and $W \rightarrow Z$, then $X \rightarrow Z$.

Fourth Normal Form (4NF)

- A relation scheme R is in Fourth Normal Form (4NF) with respect to a set M of FDs and MVDs if for every non-trivial MVD $X \rightarrow\rightarrow W$ in M^+ that holds in R, X is a key of R.
- Let U be a set of attributes and M be a set of FDs and MVDs. Then D = {R1, ..., Rn} is a 4NF decomposition of U if
  - D is a join loss-less decomposition of U, and
  - every relation scheme Ri in D is in 4NF wrt M.

Example

1. For the example of CTHRSG, we have
   C $\rightarrow\!\!\rightarrow$ HR | T | SG
   Thus, {CHR, CT, CSG} is a 4NF decomposition of CTHRSG.

2. For Faculty = { Prof, Course, GraduateStudent }
   Prof $\rightarrow\!\!\rightarrow$ Course | GraduateStudent
   Thus, { (Prof, Course); (Prof, GraduateStudent)} is a 4NF decomposition of Faculty.

3. Consider Bank = { Customer, Account, Balance, Loan, Amount}
   Customer $\rightarrow\!\!\rightarrow$ Account, Balance | Loan, Amount
   Thus, {(Customer, Loan, Amount); (Customer, Account, Balance)} is a 4NF decomposition of Bank.

4. Consider Employee (Name, Project, Dependent)
   Name $\rightarrow\!\!\rightarrow$ Project | Dependent
   Thus, {(Name, Project); (Name, Dependent)} is a 4NF decomposition of Employee.

Join loss-less decomposition

{R1, R2} is a join loss-less decomposition of R with respect to a set of FDs and MVDs if

$$R1 \cap R2 \rightarrow\rightarrow R1.$$

Algorithm for 4NF decomposition

Input U: a set of Attributes
M: a set of MVDs on U
Output D: a 4NF decomposition of U
Method

1. D = {U};
2. **while** there exists a relation scheme Q in D that is not in 4NF **do**
   begin
   find a non-trivial MVD $X \rightarrow\rightarrow W$ in Q that violates 4NF;
   replace Q in D by two schemes $(XW \cap Q)$ and $(Q - W)X$;
   end

Problems with the algorithm

- How to determine if a relation scheme obtained from the decomposition is in 4NF
  - embeded MVDs
- How to choose an appropriate MVD that violates 4NF for the decomposition

Problems with 4NF decomposition

- The set of MVDs considered must be a 4NF cover
  - it is very difficult to compute a 4NF cover.
- Roles of FDs during the decomposition