NORMALIZACIJA

Iztok Savnik

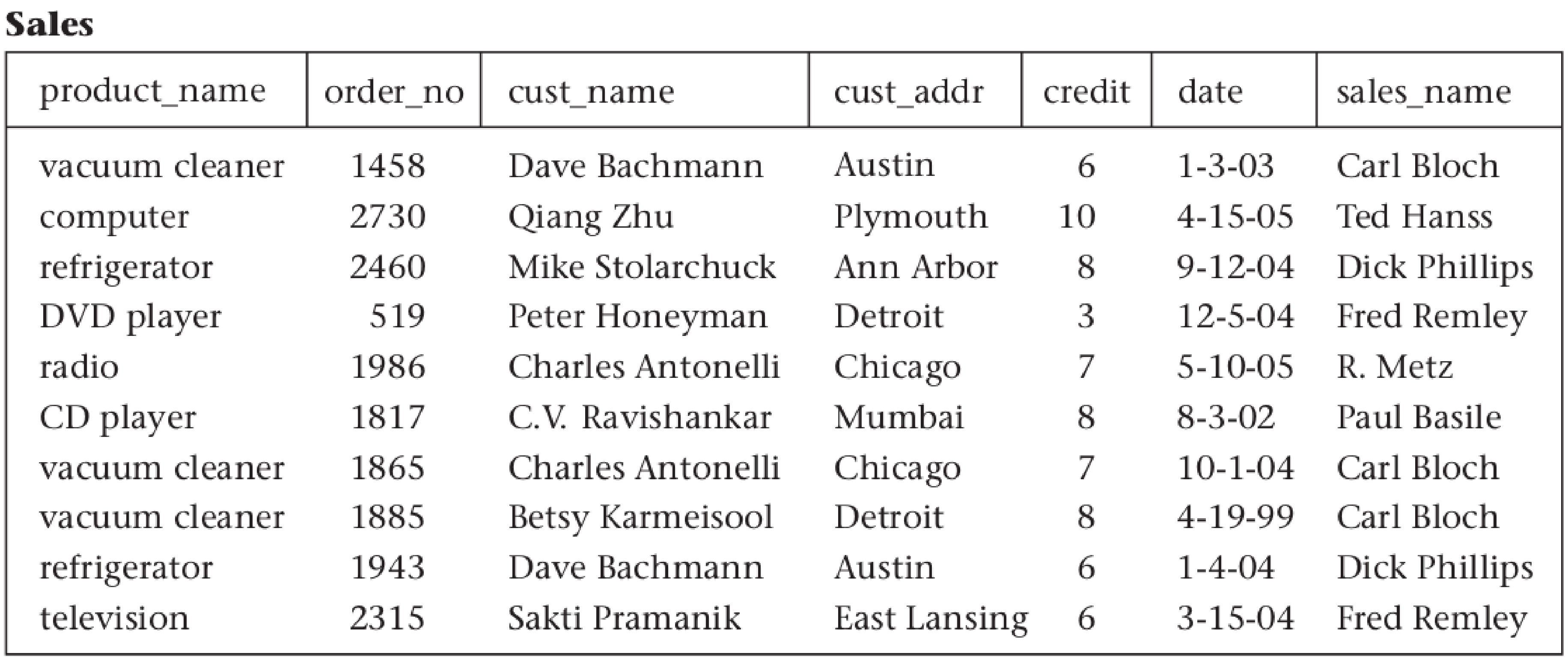
Normalization

- Fundamentals of normal forms for relational databases
- The database design step that normalizes the candidate tables
- Investigates the equivalence between the conceptual data model (e.g., the ER model) and normal forms for tables
- Good, thoughtful design of a conceptual model will result in databases that are either already normalized or can be easily normalized with minor changes

Fundamentals of normalization

- Database tables, whether they are derived from ER or UML models, sometimes suffer from some rather serious performance problems, integrity and maintainability
- Database defined as a single large table can result
  - large amount of redundant data and lengthy searches
  - long and expensive updates, and deletions
  - elimination of useful data as an unwanted side effect

Example



Sales

product_name	order_no	cust_name	cust_addr	credit	date	sales_name
vacuum cleaner	1458	Dave Bachmann	Austin	6	1-3-03	Carl Bloch
computer	2730	Qiang Zhu	Plymouth	10	4-15-05	Ted Hanss
refrigerator	2460	Mike Stolarchuck	Ann Arbor	8	9-12-04	Dick Phillips
DVD player	519	Peter Honeyman	Detroit	3	12-5-04	Fred Remley
radio	1986	Charles Antonelli	Chicago	7	5-10-05	R. Metz
CD player	1817	C.V. Ravishankar	Mumbai	8	8-3-02	Paul Basile
vacuum cleaner	1865	Charles Antonelli	Chicago	7	10-1-04	Carl Bloch
vacuum cleaner	1885	Betsy Karmeisool	Detroit	8	4-19-99	Carl Bloch
refrigerator	1943	Dave Bachmann	Austin	6	1-4-04	Dick Phillips
television	2315	Sakti Pramanik	East Lansing	6	3-15-04	Fred Remley

Problems

- Certain product and customer information is stored redundantly
- Queries, such as “Which customers ordered vacuum cleaners last month?” would require a search of the entire table
- Updates such as changing the address of the customer Dave Bachmann would require changing many rows
- Deleting an order by a customer such as Qiang Zhu, if that is his only outstanding order, deletes the only copy of his address and credit rating as a side effect

Normalization

- Classes of relational database schemes or table definitions, called normal forms
  - – analyzing the interdependencies among individual attributes associated with those tables and
  - – taking projections of larger tables to form smaller ones

First normal form

- Definition.
  - A table is in first normal form (1NF) if and only if all columns contain only atomic values, that is, each column can have only one value for each row in the table
  - the most basic level of normalized tables
- Example **Sales**
  - atributes have only atomic values

First normal form

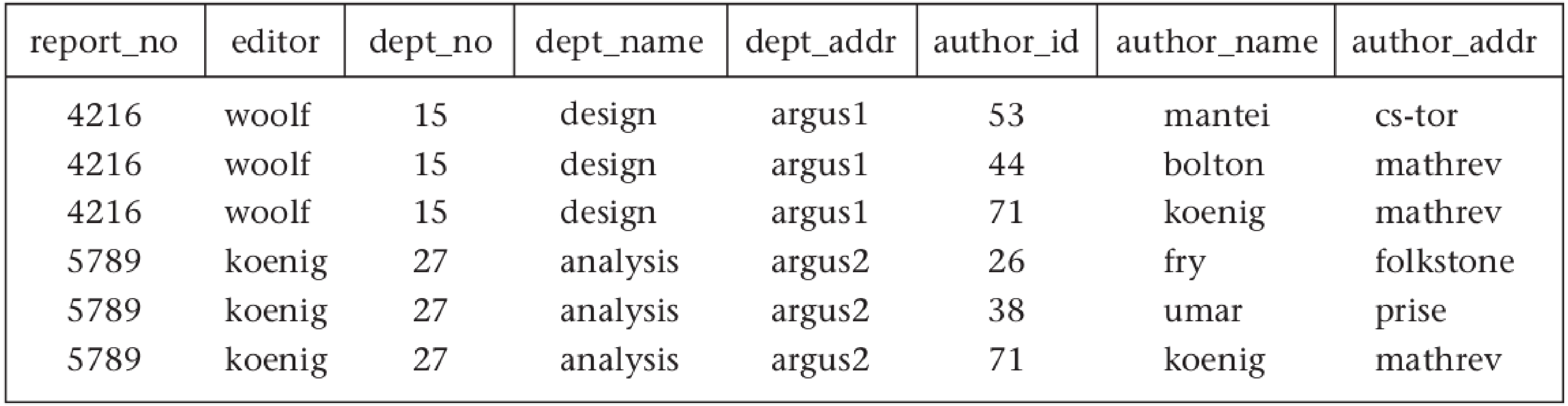
- A domain, an attribute, and a column
  - domain is the set of all possible values for a particular type of attribute
    - may be used for more than one attribute
  - column in a relational table represents a single attribute
    - more than one column may refer to different attributes from the same domain

Keys

- Superkey
  - set of one or more attributes, which, when taken collectively, allows us to identify uniquely an entity or table
- Candidate key
  - subset of the attributes of a superkey that is also a superkey, and not reducible to another superkey
- Primary key is selected arbitrarily from the set of candidate keys

Example **report**

Report



report_no	editor	dept_no	dept_name	dept_addr	author_id	author_name	author_addr
4216	woolf	15	design	argus1	53	mantei	cs-tor
4216	woolf	15	design	argus1	44	bolton	mathrev
4216	woolf	15	design	argus1	71	koenig	mathrev
5789	koenig	27	analysis	argus2	26	fry	folkstone
5789	koenig	27	analysis	argus2	38	umar	prise
5789	koenig	27	analysis	argus2	71	koenig	mathrev

Example **report**

- Composite of all the attributes of the table forms a superkey
  - duplicate rows are not allowed in relational model
- Composite (report_no, author_id) uniquely determines all the other attributes
  - neither report_no nor author_id alone can determine a row uniquely
  - composite (report_no, author_id) becomes a candidate key
  - the only candidate key therefore also the primary key
- If we had an additional column for author_ssn
  - both (report_no, author_id) and (report_no, author_ssn) would be candidate keys

Second normal form

- Functional dependency
  - one or more attributes uniquely determine the value of one or more other attributes
- Example table **report**
  - report_no -> editor, dept_no
  - dept_no -> dept_name, dept_addr
  - author_id -> author_name, author_addr

Second normal form

- **Definition**.
  - A table is in second normal form (2NF) if and only if it is in 1NF and every nonkey attribute is fully dependent on the primary key. An attribute is fully dependent on the primary key if it is on the right side of an FD for which the left side is either the primary key itself or something that can be derived from the primary key using the transitivity of FDs.

Example **report**

- An example of a transitive FD in report is the following:
  - report_no -> dept_no
  - dept_no -> dept_name
  - We can derive: report_no -> dept_name

Example **report**

- FD dept_no -> dept_name,dept_addr
  - has no component of the primary key on the left side
- FDs report_no -> editor, dept_no and author_id -> author_name, author_addr
  - contain one component of the primary key on the left side, but not both components
- **report** does not satisfy the condition for 2NF for any of the FDs

Alternative definitions

- **Definition**. (C.J.Date)
  1) Relation R is in second normal form (2NF) if and only if, for every key K of R and every nonkey attribute A of R, the FD K → {A} (which holds in R, necessarily) is irreducible.
  2) Relation R is in second normal form (2NF) if and only if, for every nontrivial FD X → Y that holds in R, at least one of the following is true: (a) X is a superkey; (b) Y is a subkey; (c) X is not a subkey.

Disadvantages of 1NF

- Update anomaly
- Insert anomaly
- Delete anomalies

Example **report**

- Transforming the 1NF table into two or more 2NF tables
- **report1**
  - report_no, editor, dept_no, dept_name, and dept_addr
- **report2**
  - author_id, author_name, and author_addr
- **report3**
  - report_no and author_id

Example **report**

- The FDs for these 2NF tables are:
  - **report1**
    - report_no -> editor, dept_no
    - dept_no -> dept_name, dept_addr
  - **report2**
    - author_id -> author_name, author_addr
  - **report3**
    - report_no, author_id
    - candidate key (no Fds)
- Anomalies eliminated

Example **report**

Report 1

report_no	editor	dept_no	dept_name	dept_addr
4216	woolf	15	design	argus 1
5789	koenig	27	analysis	argus 2

Report 2

author_id	author_name	author_addr
53	mantei	cs-tor
44	bolton	mathrev
71	koenig	mathrev
26	fry	folkstone
38	umar	prise
71	koenig	mathrev

Report 3

report_no	author_id
4216	53
4216	44
4216	71
5789	26
5789	38
5789	71

Example **report**

- Not all performance degradation is eliminated
  - report_no is still duplicated for each author,
  - deletion of a report requires updates to two tables (report1 and report3) instead of one
- Minor problems compared to those in the 1NF table report

Third normal form

- 2NF tables still suffer from
  - the same types of anomalies as the 1NF tables
  - different reasons associated with transitive dependencies
- If a transitive FD exists in a table
  - two separate facts are represented in that table
    - one fact for each FD involving a different left side

Third normal form

- Problems with transitive FD
  - if we delete a report from the database
    - this involves tables **report1** and **report3**
    - we delete the association between dept_no, dept_name, and dept_addr
  - delete anomaly

Example **report**

- Project **report1** to **report11** and **report12**
- **report11**
  - dept_no, dept_name, and dept_addr
- **report12**
  - report_no, editor, and dept_no

Third normal form

- **Definition**.
  - A table is in third normal form (3NF) if and only if for every nontrivial functional dependency X->A, where X and A are either simple or composite attributes, one of two conditions must hold.
    - Either attribute X is a superkey, or
    - attribute A is a member of a candidate key.
      - If attribute A is a member of a candidate key, A is called a prime attribute.
  - Note: a trivial FD is of the form YZ->Z

Example **report**

- After projecting report1 into report11 and report12
  - to eliminate the transitive dependency report_no -> dept_no -> dept_name, dept_addr
- We have the following 3NF tables and Fds
  - report11: report_no -> editor, dept_no
  - report12: dept_no -> dept_name, dept_addr
  - report2: author_id -> author_name, author_addr
  - report3: report_no, author_id
    - candidate keys (no FDs)

Boyce-Codd normal form

- 3NF is the most common standard for normalization in commercial databases and CASE tools
  - eliminates most of the anomalies known in databases today
  - few remaining anomalies can be eliminated by the Boyce-Codd normal form (BCNF) and higher normal forms

Boyce-Codd normal form

- **Definition**.
  - A table R is in Boyce-Codd normal form (BCNF) if for every nontrivial FD X->A, X is a superkey.
- BCNF is considered to be a strong variation of 3NF.

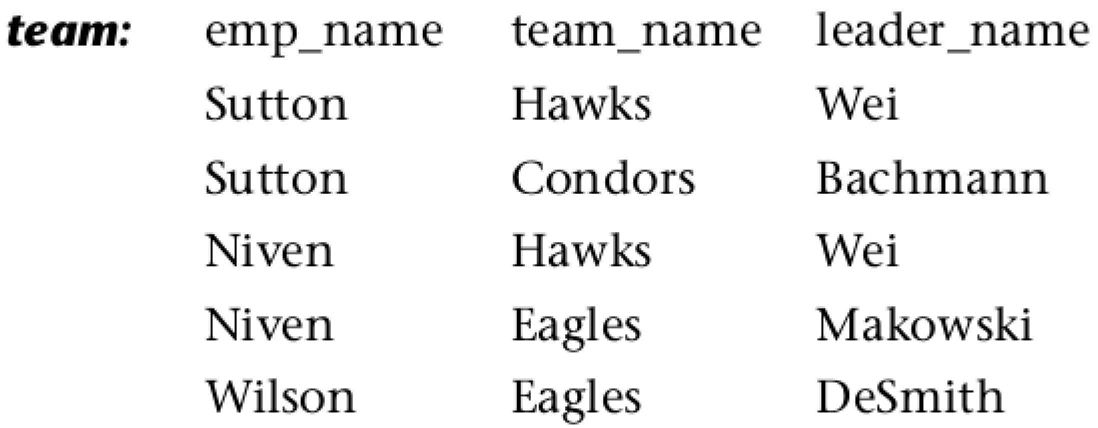
Boyce-Codd normal form

- **Definition**.
  - A table R is in Boyce-Codd normal form (BCNF) if for every nontrivial FD X->A, X is a superkey.
- BCNF is considered to be a strong variation of 3NF.

Example **team**

- 3NF table that is not BCNF
  - Assertion 1.
    - For a given team, each employee is directed by only one leader. A team may be directed by more than one leader.
    - emp_name, team_name -> leader_name
  - Assertion 2.
    - Each leader directs only one team.
    - leader_name -> team_name

Example **team**



team:

emp_name	team_name	leader_name
Sutton	Hawks	Wei
Sutton	Condors	Bachmann
Niven	Hawks	Wei
Niven	Eagles	Makowski
Wilson	Eagles	DeSmith

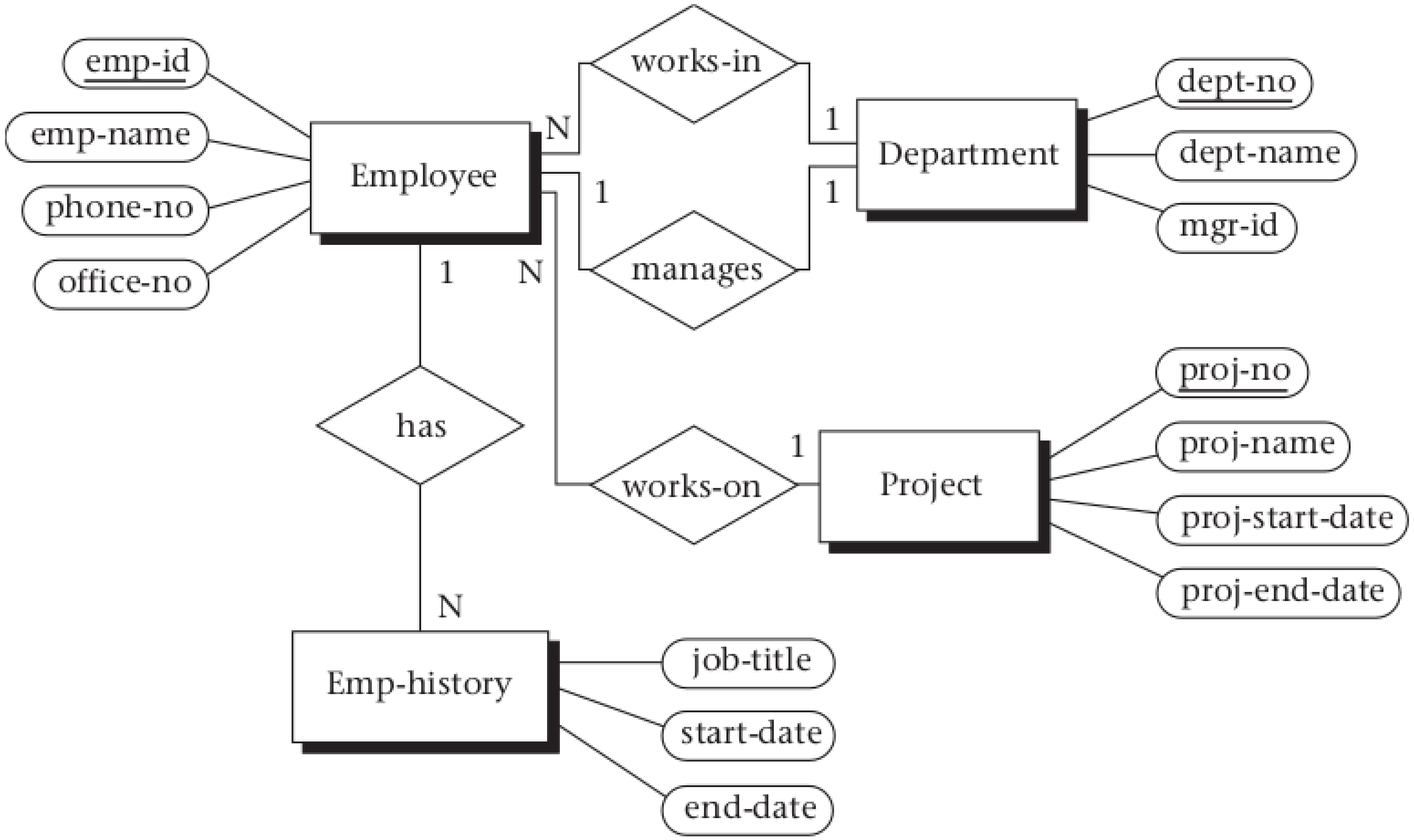
Example **team**

- Delete anomaly
  - Team table has the following delete anomaly
    - if Sutton drops out of the Condors team, then we have no record of Bachmann leading the Condors team
    - As shown by Date [1999], this type of anomaly cannot have a lossless decomposition and preserve all Fds.

Example **team**

- Delete anomaly
  - Simplest way to avoid the delete anomaly
    - create a separate table for each of the two assertions
    - these two tables are partially redundant, enough so to avoid the delete anomaly
  - This decomposition is lossless (trivially) and preserves Fds
    - it also degrades update performance due to redundancy, and necessitates additional storage space.

Example: normalized tables



Example: normalized tables

- FDs can be given explicitly, derived from the ER diagram, or derived from intuition
- Functional dependencies:
  1. emp_id, start_date -> job_title, end_date
  2. emp_id -> emp_name, phone_no, office_no, proj_no, proj_name, dept_no
  3. phone_no -> office_no
  4. proj_no -> proj_name, proj_start_date, proj_end_date
  5. dept_no -> dept_name, mgr_id
  6. mgr_id -> dept_no

Example: normalized tables

- Objective:
  - to design a relational database schema that is normalized to at least 3NF and,
  - if possible, minimize the number of tables required
- Approach:
  - to apply the definition of third normal form (3NF) to the FDs given above, and
  - create tables that satisfy the definition.

Example: normalized tables

- If we try to put FDs 1 through 6 into a single table with the composite candidate key (emp_id, start_date), we violate the 3NF definition, because FDs 2 through 6 involve left sides of FDs that are not superkeys.
  - we need to separate 1 from the rest of the FDs
- If we then try to combine 2 through 6, we have many transitivities.
  - we know that 2, 3, 4, and 5 must be separated into different tables because of transitive dependencies.
  - We then must decide whether 5 and 6 can be combined without loss of 3NF;
  - this can be done because mgr_id and dept_no are mutually dependent and both attributes are superkeys in a combined table.

Example: normalized tables

- We can define the following tables:

emp_hist:	emp_id, start_date -> job_title, end_date
employee:	emp_id -> emp_name, phone_no, proj_no, dept_no
phone:	phone_no -> office_no
project:	proj_no -> proj_name, proj_start_date, proj_end_date
department:	dept_no -> dept_name, mgr_id
	mgr_id -> dept_no

- This solution is BCNF as well as 3NF:
  - maintains all the original FDs
  - it is also a minimum set of normalized tables

Example: normalized tables

- Alternative designs may involve
  - splitting tables into partitions
    - volatile (frequently updated) and
    - passive (rarely updated) data,
  - consolidating tables
    - to get better query performance, or
    - duplicating data in different tables to get better query performance without losing integrity.

Example: normalized tables

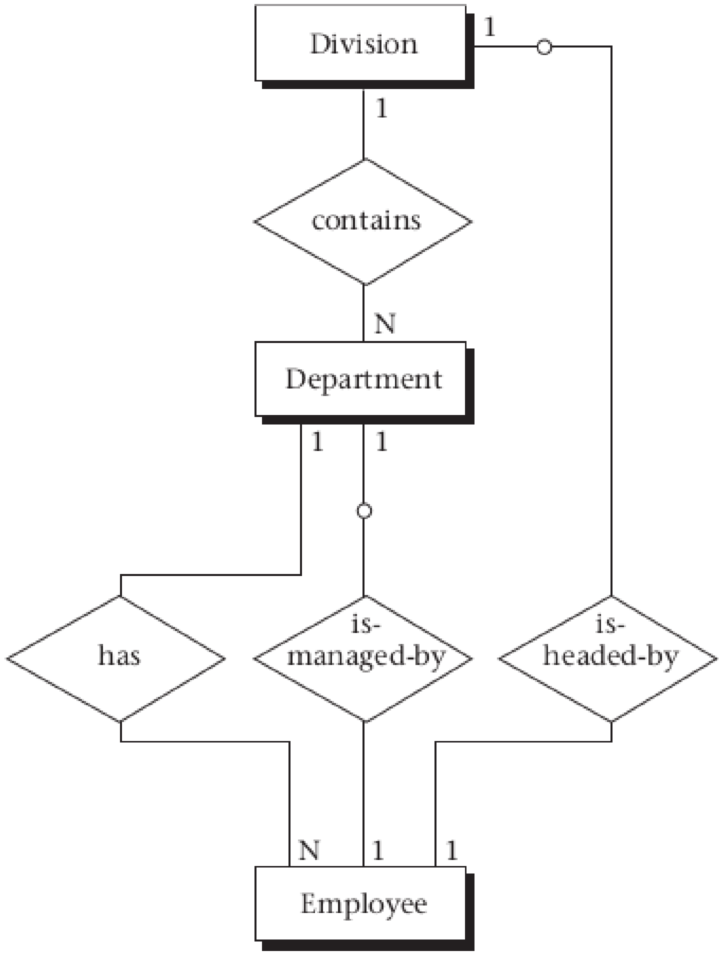
- The measures we use to assess the trade-offs in our design are:
  - Query performance (time)
  - Update performance (time)
  - Storage performance (space)
  - Integrity (avoidance of delete anomalies)

Normalization of Candidate Tables Derived from ER Diagrams

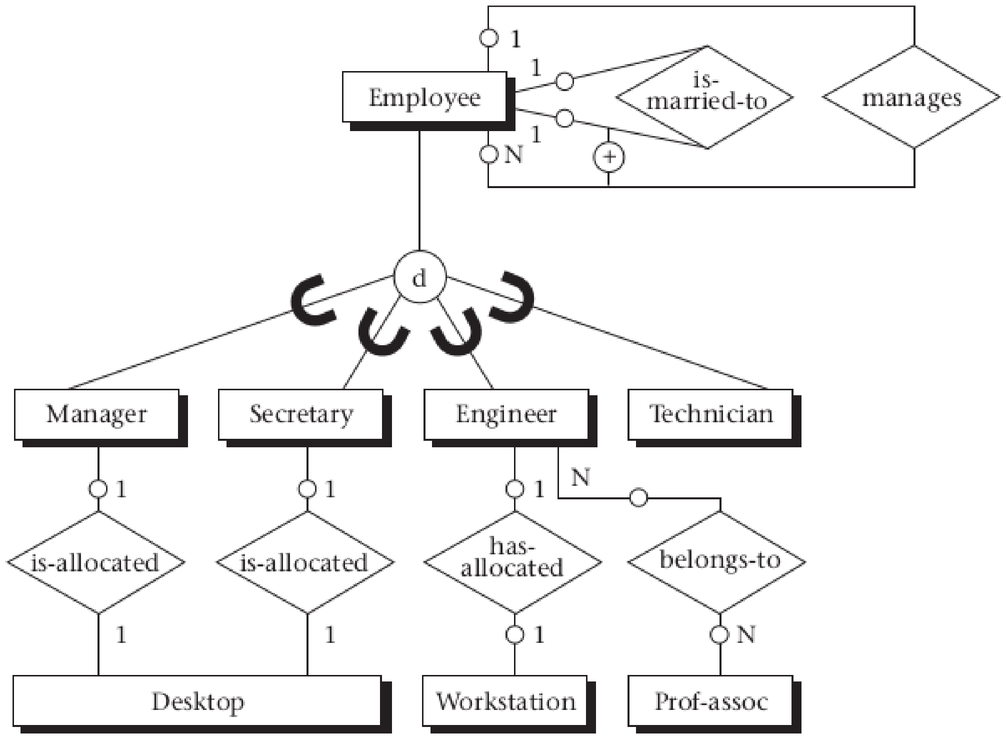
- Normalization of candidate tables => analyzing the FDs associated with those tables:
  - explicit FDs from the database requirements analysis,
  - FDs derived from the ER diagram, and
  - FDs derived from intuition.
- Primary FDs
  - Derived from the ER diagram
  - Represent the dependencies among the data elements that are keys of entities
  - Interentity dependencies.

Normalization from ER diagrams

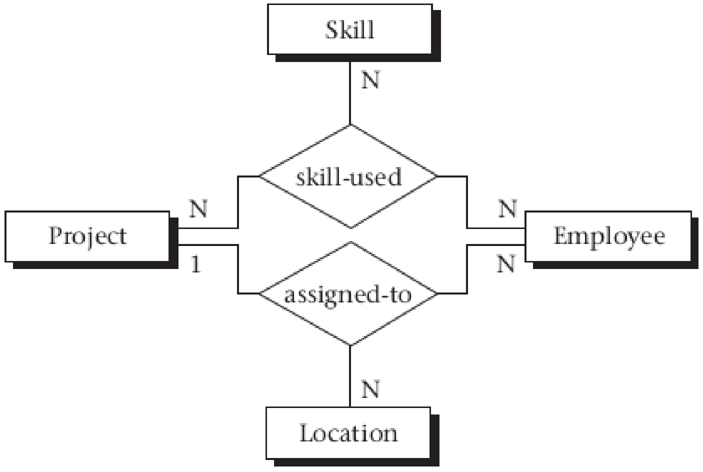
- Secondary FDs
  - Obtained explicitly from the requirements analysis.
  - Represent dependencies among data elements that comprise a single entity
  - Intraentity dependencies.
- If the ER constructs do not include nonkey attributes used in secondary FDs
  - the data requirements specification or data dictionary must be consulted.



(a) Management view



(b) Employee view



(c) Employee assignment view

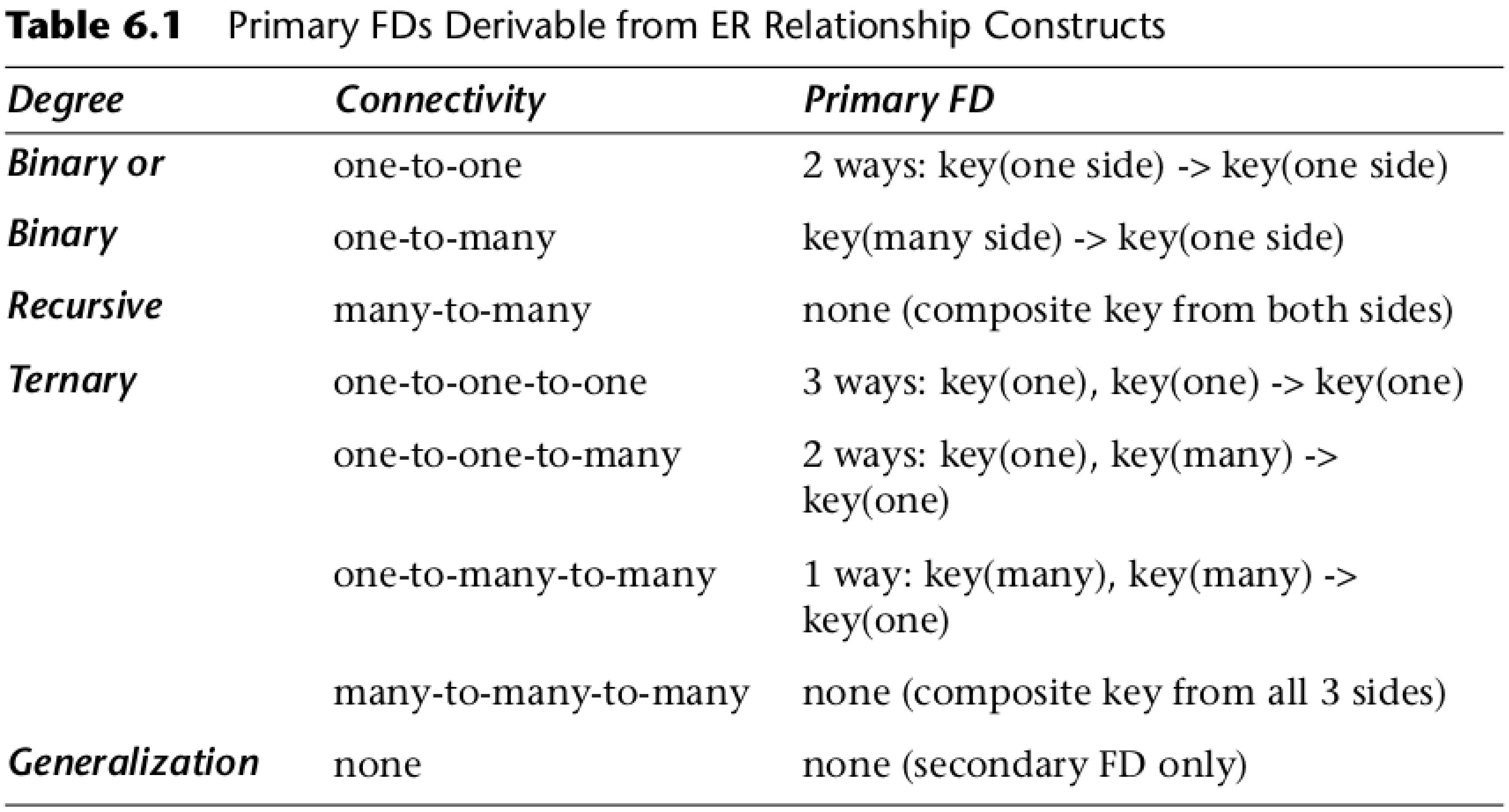


Table 6.1 Primary FDs Derivable from ER Relationship Constructs

Degree	*Connectivity*	*Primary FD*
Binary or	one-to-one	2 ways: key(one side) -> key(one side)
Binary	one-to-many	key(many side) -> key(one side)
Recursive	many-to-many	none (composite key from both sides)
Ternary	one-to-one-to-one	3 ways: key(one), key(one) -> key(one)
	one-to-one-to-many	2 ways: key(one), key(many) -> key(one)
	one-to-many-to-many	1 way: key(many), key(many) -> key(one)
	many-to-many-to-many	none (composite key from all 3 sides)
Generalization	none	none (secondary FD only)

Normalization from ER diagrams

- If the ER constructs do not include nonkey attributes used in secondary Fds
  - data requirements specification or data dictionary must be consulted
- Each candidate table will typically have several primary and secondary FDs uniquely associated
- Any table B that is subsumed by another table A can potentially be eliminated.
  - Table B is subsumed by another table A when all the attributes in B are also contained in A, and all data dependencies in B also occur in A.

Normalization from ER diagrams

- Obtain the primary FDs by applying the rules in Table 6.1
  - The results are shown in Table 6.2.
- Determine the secondary Fds
  - Table 6.3 are derived from the requirements specification and intuition
- Normalization of the candidate tables
  - Table 6.4 brings together the primary and secondary FDs

Normalization from ER diagrams

- We note that for each table except employee, all attributes are functionally dependent on the primary key (denoted by the left side of the FDs) and are thus BCNF.
- In the case of table employee, we note that spouse_id determines emp_id and emp_id is the primary key; thus spouse_id can be shown to be a superkey.
- Therefore, employee is found to be BCNF.

Table 6.2 Primary FDs Derived from the ER Diagram in Figure 4.3

dept_no -> div_no	in Department from relationship “contains”
emp_id -> dept_no	in Employee from relationship “has”
div_no -> emp_id	in Division from relationship “is-headed-by”
dept_no -> emp_id	from binary relationship “is-managed-by”
emp_id -> desktop_no	from binary relationship “has-allocated”
desktop_no -> emp_no	from binary relationship “has-allocated”
emp_id -> spouse_id	from binary recursive relationship “is-married-to”
spouse_id -> emp_id	from binary recursive relationship “is-married-to”
emp_id, loc_name -> project_name	from ternary relationship “assigned-to”

Table 6.3 Secondary FDs Derived from the Requirements Specification

div_no -> div_name, div_addr	from entity Division
dept_no -> dept_name, dept_addr, mgr_id	from entity Department
emp_id -> emp_name, emp_addr, office_no, phone_no	from entity Employee
skill_type -> skill_descrip	from entity Skill
project_name -> start_date, end_date, head_id	from entity Project
loc_name -> loc_county, loc_state, zip	from entity Location
mgr_id -> mgr_start_date, beeper_phone_no	from entity Manager
assoc_name -> assoc_addr, phone_no, start_date	from entity Prof-assoc
desktop_no -> computer_type, serial_no	from entity Desktop

Normalization from ER diagrams

- In general, we observe that candidate tables, like the ones shown in Table 6.4,
  - are fairly good indicators of the final schema and normally
  - require very little refinement to get to 3NF or BCNF. This observation is
- Important—good initial conceptual design usually results in tables that are already normalized or are very close to being normalized
- Thus, the normalization process is usually a simple task.

Table 6.4 Candidate Tables (and FDs) from ER Diagram Transformation

division	div_no -> div_name, div_addr div_no -> emp_id
department	dept_no -> dept_name, dept_addr, mgr_id dept_no -> div_no dept_no -> emp_id
employee	emp_id -> emp_name, emp_addr, office_no, phone_no emp_id -> dept_no emp_id -> spouse_id spouse_id -> emp_id
manager	mgr_id -> mgr_start_date, beeper_phone_no
secretary	none
engineer	emp_id -> desktop_no

Table 6.4 Candidate Tables (and FDs) from ER Diagram Transformation *(continued)*

technician	none
skill	skill_type -> skill_descrip
project	project_name -> start_date, end_date, head_id
location	loc_name -> loc_county, loc_state, zip
prof_assoc	assoc_name -> assoc_addr, phone_no, start_date
desktop	desktop_no -> computer_type, serial_no desktop_no -> emp_no
assigned_to	emp_id, loc_name -> project_name
skill_used	none

Determining the Minimum Set of 3NF Tables

- Synthesis algorithm developed by Bernstein, 1976
- Process usefull when you have several 100-1000 FDs
- ER modeling process automatically decomposes this problem into smaller subproblems
  - the attributes and FDs of interest are restricted to those attributes within an entity
  - any foreign keys that might be imposed upon that table

Minimum Set of 3NF Tables

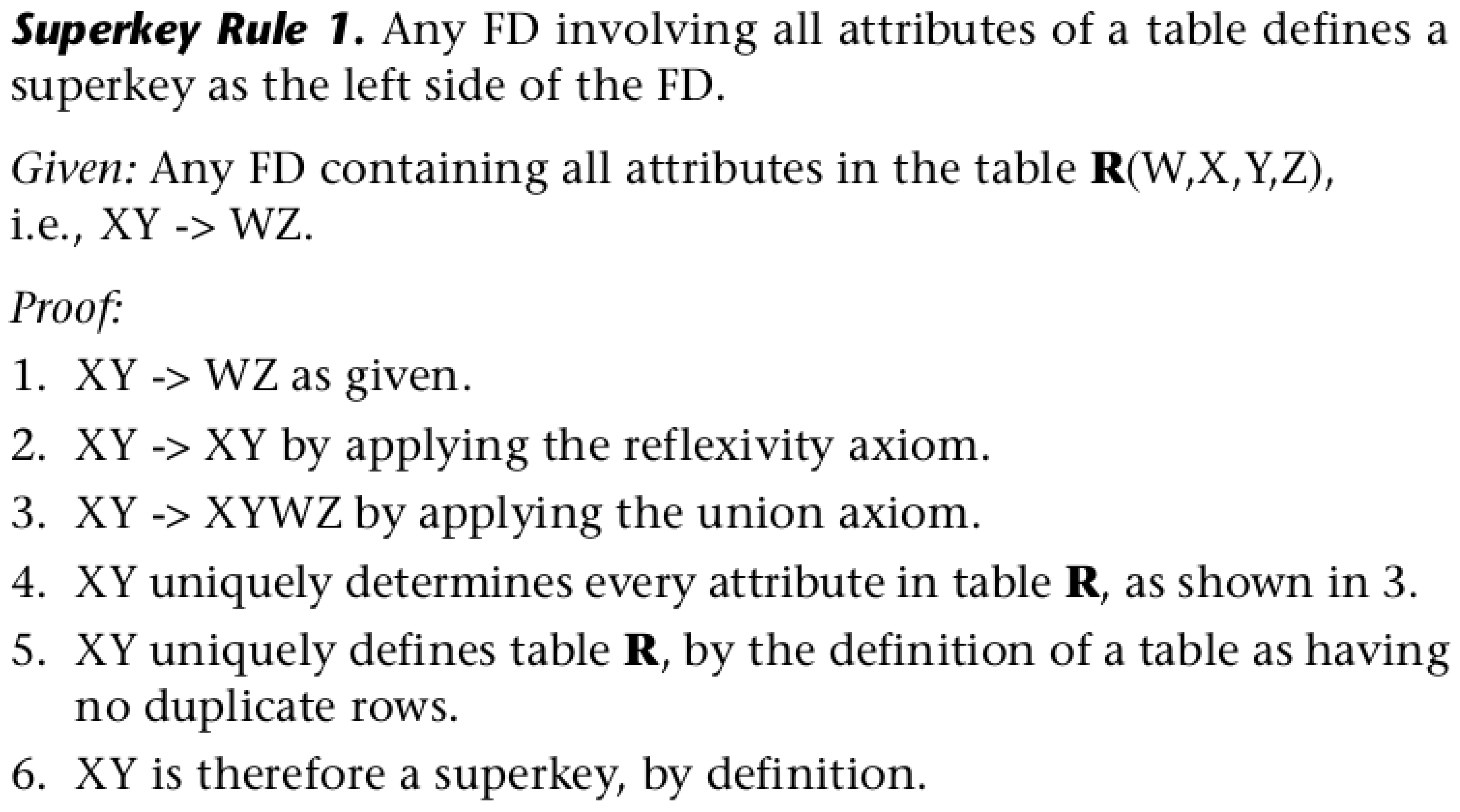
  - rarely deals with more than 10-20 attributes at a time
  - most entities are initially defined in 3NF already
  - tables that are not yet in 3NF: only minor adjustments will be needed in most cases
- Synthesis algorithm for those situations where the ER model is not useful for the decomposition
  - We make use of the well-known Armstrong axioms

Inference rules (Armstrong axioms)

Reflexivity	If Y is a subset of the attributes of X, then X -> Y (i.e., if X is ABCD and Y is ABC, then X -> Y. Trivially, X -> X)
Augmentation	If X -> Y and Z is a subset of table R (i.e., Z is any attribute in R), then XZ -> YZ.
Transitivity	If X->Y and Y->Z, then X->Z.
Pseudotransitivity	If X->Y and YW->Z, then XW->Z. (Transitivity is a special case of pseudotran.) when W = null.)
Union	If X->Y and X->Z, then X->YZ (or equivalently, X->Y,Z).
Decomposition	If X->YZ, then X->Y and X->Z.

Minimum Set of 3NF Tables

Two practical rules of thumb for deriving superkeys of tables where at least one superkey is already known



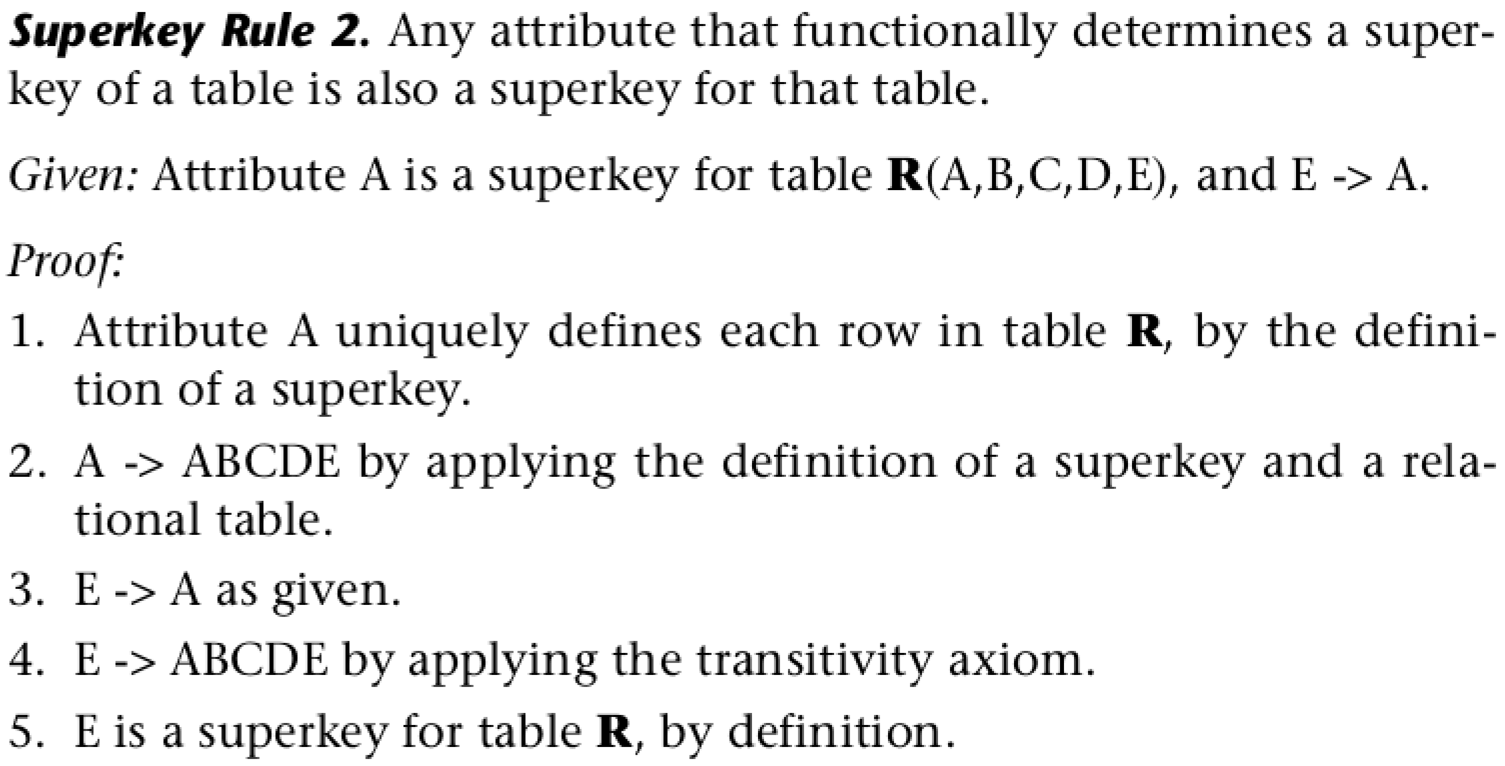
Superkey Rule 1. Any FD involving all attributes of a table defines a superkey as the left side of the FD.

Given: Any FD containing all attributes in the table **R**(W,X,Y,Z), i.e., XY -> WZ.

Proof:

1. XY -> WZ as given.
2. XY -> XY by applying the reflexivity axiom.
3. XY -> XYWZ by applying the union axiom.
4. XY uniquely determines every attribute in table **R**, as shown in 3.
5. XY uniquely defines table **R**, by the definition of a table as having no duplicate rows.
6. XY is therefore a superkey, by definition.

Minimum Set of 3NF Tables



Superkey Rule 2. Any attribute that functionally determines a superkey of a table is also a superkey for that table.

Given: Attribute A is a superkey for table **R**(A,B,C,D,E), and E -> A.

Proof:

1. Attribute A uniquely defines each row in table **R**, by the definition of a superkey.
2. A -> ABCDE by applying the definition of a superkey and a relational table.
3. E -> A as given.
4. E -> ABCDE by applying the transitivity axiom.
5. E is a superkey for table **R**, by definition.

Minimum Set of 3NF Tables

- Let H be a set of FDs that represents at least part of the known semantics of a database
- The closure of H, specified by H+, is the set of all FDs derivable from H using the Armstrong axioms
- FDs in set H:
  - A -> B, B -> C, A -> C, and C -> D
  - to derive the FDs A -> D and B -> D
  - All six FDs constitute the closure H+
- A cover of H, called H’, is any set of FDs from which H+ can be derived.

Minimum Set of 3NF Tables

- Possible covers for this example are:

1. A->B, B->C, C->D, A->C, A->D, B->D (trivial case where H’ and H+ are equal)
2. A->B, B->C, C->D, A->C, A->D
3. A->B, B->C, C->D, A->C (this is the original set H)
4. A->B, B->C, C->D

- A nonredundant cover of H is a cover of H that contains no proper subset of FDs, which is also a cover.

3NF Synthesis Algorithm

- Given a set of FDs, H, we determine a minimum set of tables in 3NF.
  1) Eliminate extraneous attributes in the left sides of the FDs
  2) Search for a nonredundant cover, G of H
  3) Partition G into groups so that all FDs with the same left side are in one group
  4) Merge equivalent keys
  5) Define the minimum set of normalized tables

3NF Synthesis Algorithm

H =

AB->C	DM->NP
A->DEFG	D->M
E->G	L->D
F->DJ	PQR->ST
G->DI	PR->S
D->KL	

Step 1. Elimination of Extraneous Attributes

The first task is to get rid of extraneous attributes in the left sides of the FDs.

The following two relationships (rules) among attributes on the left side of an FD provide the means to reduce the left side to fewer attributes.

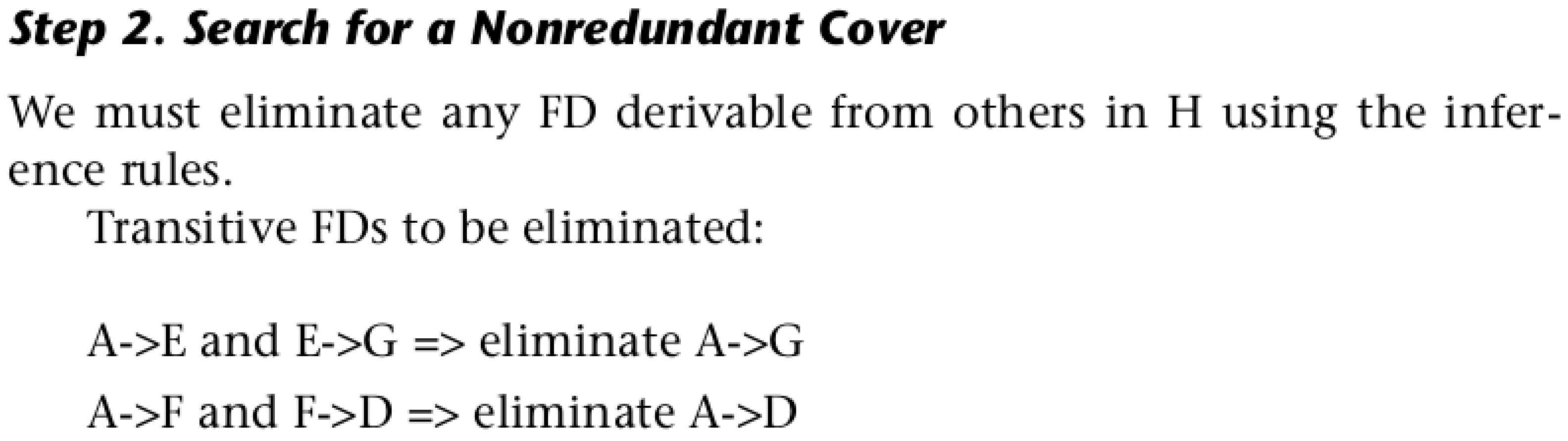
Reduction Rule 1. XY->Z and X->Z => Y is extraneous on the left side (applying the reflexivity and transitivity axioms).

Reduction Rule 2. XY->Z and X->Y => Y is extraneous; therefore X->Z (applying the pseudotransitivity axiom).

Applying these **reduction rules** to the set of FDs in H, we get:

DM->NP and D->M => D->NP

PQR->ST and PR->S => PQR->T



Step 2. Search for a Nonredundant Cover

We must eliminate any FD derivable from others in H using the inference rules.

Transitive FDs to be eliminated:

A->E and E->G => eliminate A->G

A->F and F->D => eliminate A->D

Step 3. Partitioning of the Nonredundant Cover

To partition the nonredundant cover into groups so that all FDs with the same left side are in one group, we must separate the nonfully functional dependencies and transitive dependencies into separate tables. At this point we have a feasible solution for 3NF tables, but it is not necessarily the minimum set.

These nonfully functional dependencies must be put into separate tables:

AB->C

A->EF

Groups with the same left side:

G1:	AB->C	G6:	D->KLMNP
G2:	A->EF	G7:	L->D
G3:	E->G	G8:	PQR->T
G4:	G->DI	G9:	PR->S
G5:	F->DJ		

H:	AB->C	DM->NP
	A->DEFG	D->M
	E->G	L->D
	F->DJ	PQR->ST
	G->DI	PR->S
	D->KL	

Step 4. Merge of Equivalent Keys (Merge of Tables)

In this step we merge groups with left sides that are equivalent (for example, X->Y and Y->X imply that X and Y are equivalent). This step produces a minimum set.

1. Write out the closure of all left side attributes resulting from Step 3, based on transitivities.
2. Using the closures, find tables that are subsets of other groups and try to merge them. Use Superkey Rule 1 and Superkey Rule 2 to establish whether the merge will result in FDs with superkeys on the left side. If not, try using the axioms to modify the FDs to fit the definition of superkeys.
3. After the subsets are exhausted, look for any overlaps among tables and apply Superkey Rules 1 and 2 (and the axioms) again.

In this example, note that G7 (L->D) has a subset of the attributes of G6 (D->KLMNP). Therefore, we merge to a single table, R6, with FDs D->KLMNP and L->D, because it satisfies 3NF: D is a superkey by Superkey Rule 1, and L is a superkey by Superkey Rule 2.

Step 5. Definition of the Minimum Set of Normalized Tables

The minimum set of normalized tables has now been computed. We define them below in terms of the table name, the attributes in the table, the FDs in the table, and the candidate keys for that table:

R1: ABC (AB->C with key AB)

R2: AEF (A->EF with key A)

R3: EG (E->G with key E)

R4: DGI (G->DI with key G)

R5: DFJ (F->DJ with key F)

R6: DKLMNP (D->KLMNP, L->D, with keys D, L)

R7: PQRT (PQR->T with key PQR)

R8: PRS (PR->S with key PR)

Note that this result is not only 3NF, but also BCNF, which is very frequently the case. This fact suggests a practical algorithm for a (near) minimum set of BCNF tables: Use Bernstein's algorithm to attain a minimum set of 3NF tables, then inspect each table for further decomposition (or partial replication, as shown in Section 6.1.5) to BCNF.

Fourth and Fifth Normal Forms

- Normal forms up to BCNF were defined solely on FDs
  - Enough for most practitioners
- Two more normal forms
  - Fourth NF & multivalued dependencies and
  - Fifth NF & join dependencies

Multivalued Dependencies

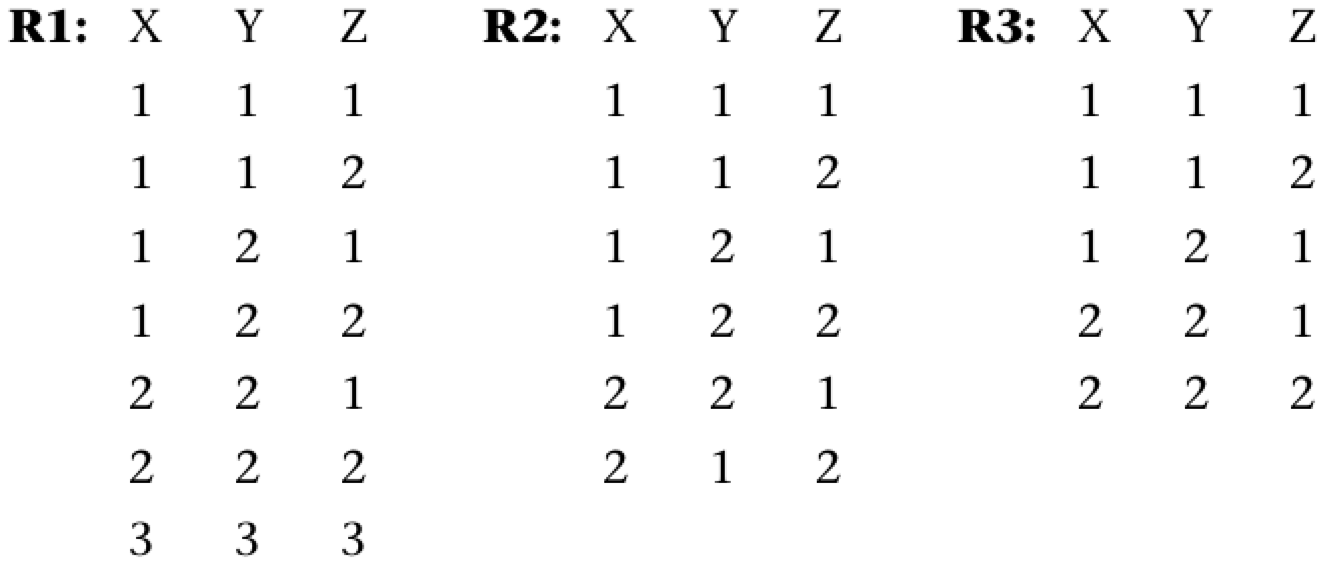
- **Definition.**

  In a multivalued dependency (MVD), X->>Y holds on table R with table scheme RS if, whenever a valid instance of table R(X,Y,Z) contains a pair of rows that contain duplicate values of X, then the instance also contains the pair of rows obtained by interchanging the Y values in the original pair. This includes situations where only pairs of rows exist. Note that X and Y may contain either single or composite attributes.

Multivalued Dependencies

- An MVD X ->> Y is trivial if Y is a subset of X, or if X union Y = RS.
- FD implies an MVD, which implies that a single row with a given value of X is also an MVD, albeit a trivial form.
- MVDs X->>Y, X->>Z appear in pairs because of the cross-product type of relationship between Y and Z=RS-X-Y

Multivalued Dependencies



R1:

X	Y	Z
1	1	1
1	1	2
1	2	1
1	2	2
2	2	1
2	2	2
3	3	3

R2:

X	Y	Z
1	1	1
1	1	2
1	2	1
1	2	2
2	2	1
2	1	2

R3:

X	Y	Z
1	1	1
1	1	2
1	2	1
2	2	1
2	2	2

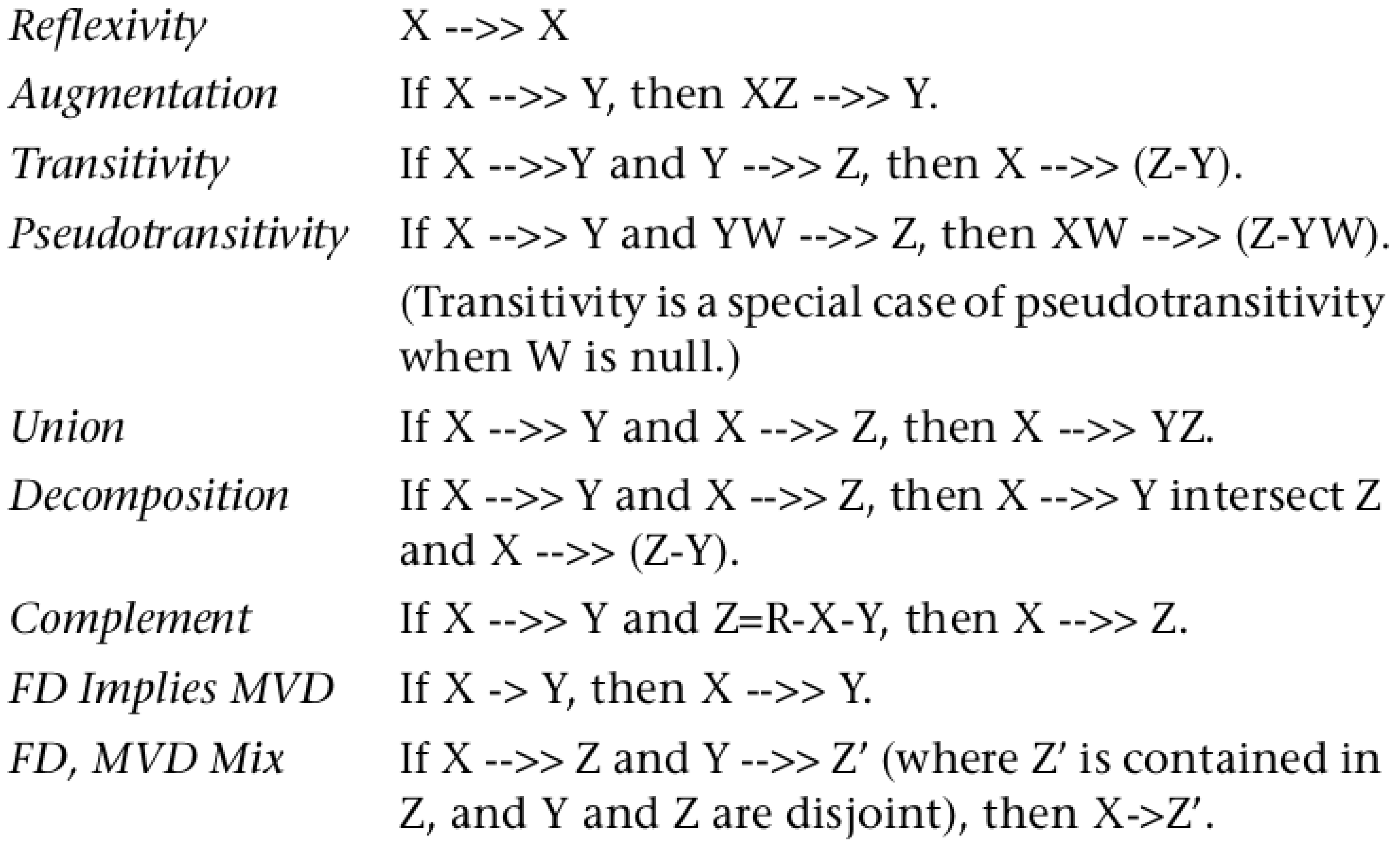
R1

- First four rows satisfy all conditions for the MVDs X->>Y and X->>Z.
  - Note that MVDs appear in pairs because of the cross-product type of relationship between Y and Z=RS-Y as the two right sides of the two MVDs.
- The fifth and sixth rows of R1 (when the X value is 2) satisfy the row interchange conditions in the above definition.
  - In both rows, the Y value is 2, so the interchanging of Y values is trivial.
- The seventh row (3,3,3) satisfies the definition trivially.

R2, R3

- In R2, R3 there is no MVD between X and Y|Z
  - Y values in the fifth and sixth rows of R2 are different (1 and 2), and interchanging the 1 and 2 values for Y results in a row (2,2,2) that does not appear in the table
  - R3 contains the first three rows that do not satisfy the criterion for an MVD, since changing Y from 1 to 2 in the second row results in a row that does not appear in the table.

Multivalued Dependency Inference Rules



Reflexivity	X -->> X
Augmentation	If X -->> Y, then XZ -->> Y.
Transitivity	If X -->>Y and Y -->> Z, then X -->> (Z-Y).
Pseudotransitivity	If X -->> Y and YW -->> Z, then XW -->> (Z-YW). (Transitivity is a special case of pseudotransitivity when W is null.)
Union	If X -->> Y and X -->> Z, then X -->> YZ.
Decomposition	If X -->> Y and X -->> Z, then X -->> Y intersect Z and X -->> (Z-Y).
Complement	If X -->> Y and Z=R-X-Y, then X -->> Z.
FD Implies MVD	If X -> Y, then X -->> Y.
FD, MVD Mix	If X -->> Z and Y -->> Z′ (where Z′ is contained in Z, and Y and Z are disjoint), then X->Z′.

Fourth Normal Form

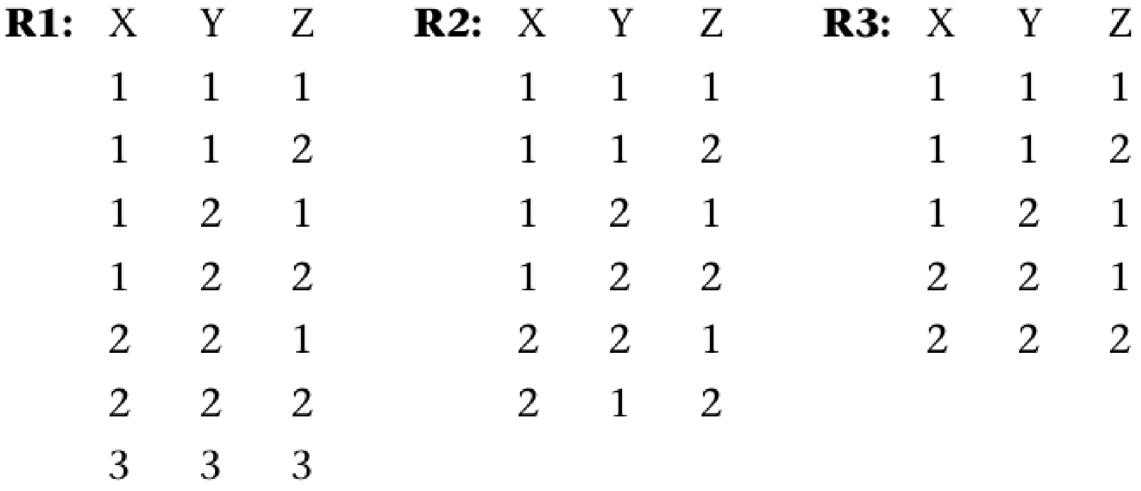
- The goal of 4NF is to eliminate nontrivial MVDs from a table by projecting them onto separate smaller tables
  - – eliminate the update anomalies associated with the MVDs
  - – easy to attain if you know where the MVDs are
  - – however, MVDs must be defined from the semantics of the database
    - can’t be determined from just looking at the data

Fourth Normal Form

- **Definition**. A table R is in fourth normal form (4NF) if and only if it is in BCNF and, whenever there exists an MVD in R (say X ->> Y), at least one of the following holds: the MVD is trivial, or X is a super-key for R.

Fourth Normal Form

- R1 is not in 4NF because at least one non-trivial MVD exists and no single column is a superkey.
- In tables R2 and R3, however, there are no MVDs. Thus these two tables are at least 4NF.



R1:

X	Y	Z
1	1	1
1	1	2
1	2	1
1	2	2
2	2	1
2	2	2
3	3	3

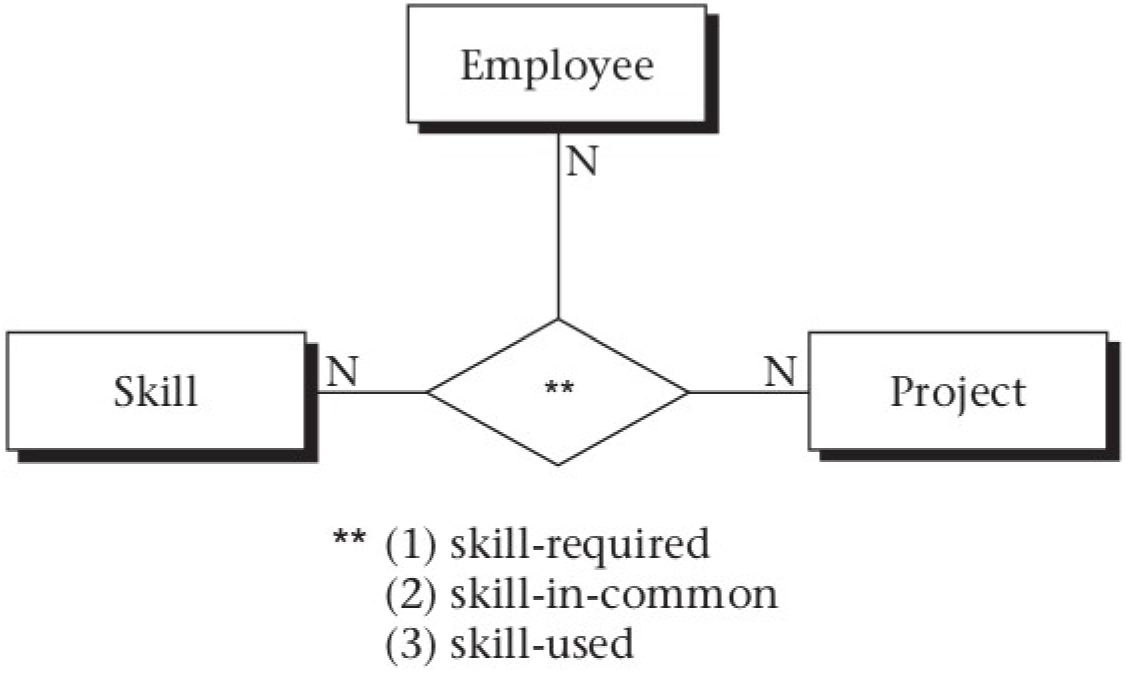
R2:

X	Y	Z
1	1	1
1	1	2
1	2	1
1	2	2
2	2	1
2	1	2

R3:

X	Y	Z
1	1	1
1	1	2
1	2	1
2	2	1
2	2	2

Ternary relationship with multiple interpretations



skill_required

emp_id	proj_no	skill_type	*MVDs(nontrivial)*
101	3	A	proj_no ->> skill_type
101	3	B	proj_no ->> emp_id
101	4	A	
101	4	C	
102	3	A	
102	3	B	
103	5	D	

skill_req1

emp_id	proj_no
101	3
101	4
102	3
103	5

skill_req2

emp_id	skill_type
101	A
101	B
101	C
102	A
102	B
103	D

skill_req3

proj_no	skill_type
3	A
3	B
4	A
4	C
5	D

Example: *skills_required*

- A two-way lossless decomposition occurs
  - skill_required is projected
    - over (emp_id, proj_no) to form skill_req1 and
    - over (proj_no, skill_type) to form skill_req3.
  - projection that is not lossless
    - over (emp_id,proj_no) to form skill_req1 and
    - over (emp_id, skill_type) to form skill_req2
- A three-way lossless decomposition occurs
  - skill_required is projected
    - over (emp_id, proj_no), (emp_id, skill_type), and (proj_no, skill_type).

Anomalies

- Tables in 4NF avoid certain update anomalies (or inefficiences).
  - Delete anomaly
    - two independent facts get tied together unnaturally so that there may be bad side effects of certain deletes
    - skill_required: last row of a skill_type may be lost if an employee is temporarily not working on any projects.

Anomalies

  - Update inefficiency
    - adding a new project in skill_required, which requires insertions for many rows to include all the required skills for that new project
    - Likewise, loss of a project requires many deletions.
- Problems are avoided after the decomposition

Decomposing Tables to 4NF

- A table is BCNF
  - it either has no FDs, or each FD is characterized by its left side being a superkey
- Select the most important MVD
  - define its complement MVD
  - decompose the table into two tables
    - containing the attributes on the left and right sides of that MVD and its complement.

Decomposing Tables to 4NF

- decomposition is lossless
  - each new table is based on the same attribute, which is the left side of both MVD
  - other MVDs may be still present
  - more decompositions by MVDs and their complements may be necessary
- process of arbitrary selection of MVDs for decomposition is continued until only trivial MVDs exist
- final tables are in 4NF.

Example: decomposion to 4NF

- R(A,B,C,D,E,F) with no FDs, and with MVDs A ->> B and CD ->> EF
  - decompose R into two tables R1(A,B) and R2(A,C,D,E,F) by using MVD A ->> B and its complement A ->> CDEF.
    - R1 is now 4NF, R2 is still only BCNF
  - decompose R2 into R21(C,D,E,F) and R22(C,D,A) by applying the MVD CD ->> EF
  - R21 and R22 are now 4NF

Example: decomposion to 4NF

- R(A,B,C,D,E,F) with no FDs, and with MVDs A ->> B and CD ->> EF
  - If we reverse the use of given MVDs, we come up with the same three 4NF tables
  - this does not occur in all cases
    - It only occurs in those tables where the MVDs have no intersecting attributes

Decomposing Tables to 4NF

- Method derives a feasible, but not necessarily a minimum, set of 4NF tables
- Side effect:
  - potentially losing some or all of FDs and MVDs

Decomposing Tables to 4NF

- Second approach
  - ignore the MVDs completely and split each BCNF table into a set of smaller tables
    - candidate key of each BCNF table being the candidate key of a new table and
    - the nonkey attributes distributed among the new tables in some semantically meaningful way
      - decomposing by superkey is lossless
    - if MVDs still exist, further decomposition must be done with the MVD/MVD-complement approach
    - decomposition by candidate keys preserves FDs, but the MVD/MVD-complement approach does not preserve either FDs or MVDs

Fifth Normal Form

- **Definition**. A table R is in fifth normal form (5NF) or project-join normal form (PJ/NF) if and only if every join dependency in R is implied by the keys of R.

Fifth Normal Form

- Lossless decomposition of a table
  - it can be decomposed by two or more projections
  - natural join of those projections (in any order) that results in the original table
- The general lossless decomposition constraint, involving any number of projections, is also known as a join dependency (JD)

Fifth Normal Form

- Example:
  - Table R with n arbitrary subsets of the set of attributes of R
  - R satisfies a join dependency over these n subsets if and only if R is equal to the natural join of its projections on them.
  - JD is trivial if one of the subsets is R itself

Fifth Normal Form

- 5NF or PJ/NF requires satisfaction of the membership algorithm [Fagin, 1979]
  - it determines whether a JD can be derived from the set of key dependencies known for this table
  - every dependency (FD, MVD, JD) is determined by the keys
- JDs are very difficult to determine in large databases with many attributes

Example: *skill-in-common*

- Table representing a ternary relationship may not have any two-way lossless decompositions
- It may have a three-way lossless decomposition
- Three binary relationships, based on the three possible projections of this table

skill_in_common

emp_id	proj_no	skill_type
101	3	A
101	3	B
101	4	A
101	4	B
102	3	A
102	3	B
103	3	A
103	4	A
103	5	A
103	5	C

skill_in_com1

emp_id	proj_no
101	3
101	4
102	3
103	3
103	4
103	5

skill_in_com2

emp_id	skill_type
101	A
101	B
102	A
102	B
103	A
103	C

skill_in_com3

proj_no	skill_type
3	A
3	B
4	A
4	B
5	A
5	C

Example: *skill-in-common*

- skill-in-common
  - “The employee must apply the intersection of his or her available skills with the skills needed to work on certain projects.”
- skill-in-common is less restrictive than skill-required
  - it allows an employee to work on a project even if he or she does not have all the skills required for that project.

Example: *skill-in-common*

- Three projections of skill_in_common result in a three-way lossless decomposition
- There are no two-way loss-less decompositions and no MVDs
  - – the table skill_in_common is in 4NF.

Example: *skill-used*

- skill-used
  - “We can selectively record different skills that each employee applies to working on individual projects.”
- Cannot be decomposed into either two or three binary tables
  - It is 5NF, has no MVDs or JDs.

skill_used	emp_id	proj_no	skill_type
	101	3	A
	101	3	B
	101	4	A
	101	4	C
	102	3	A
	102	3	B
	102	4	A
	102	4	B

Three projections on **skill_used** result in:

skill_used1		**skill_used2**		**skill_used3**	
emp_id	proj_no	proj_no	skill_type	emp_id	skill_type
101	3	3	A	101	A
101	4	3	B	101	B
102	3	4	A	101	C
102	4	4	B	102	A
		4	C	102	B

join **skill_used1** with **skill_used2** to form:

skill_used_12

emp_id	proj_no	skill_type
101	3	A
101	3	B
101	4	A
101	4	B
101	4	C
102	3	A
102	3	B
102	4	A
102	4	B
102	4	C

join **skill_used12** with **skill_used3** to form:

skill_used_123

emp_id	proj_no	skill_type
101	3	A
101	3	B
101	4	A
101	4	B (spurious)
101	4	C
102	3	A
102	3	B
102	4	A
102	4	B

Overview

- A table may have constraints that are FDs, MVDs, and JDs.
  - MVD is a special case of a JD
- To determine the level of normalization of the table
  - analyze the FDs first to determine normalization through BCNF;
  - analyze the MVDs to determine which BCNF tables are also 4NF;
  - finally, analyze the JDs to determine which 4NF tables are also 5NF.

Overview

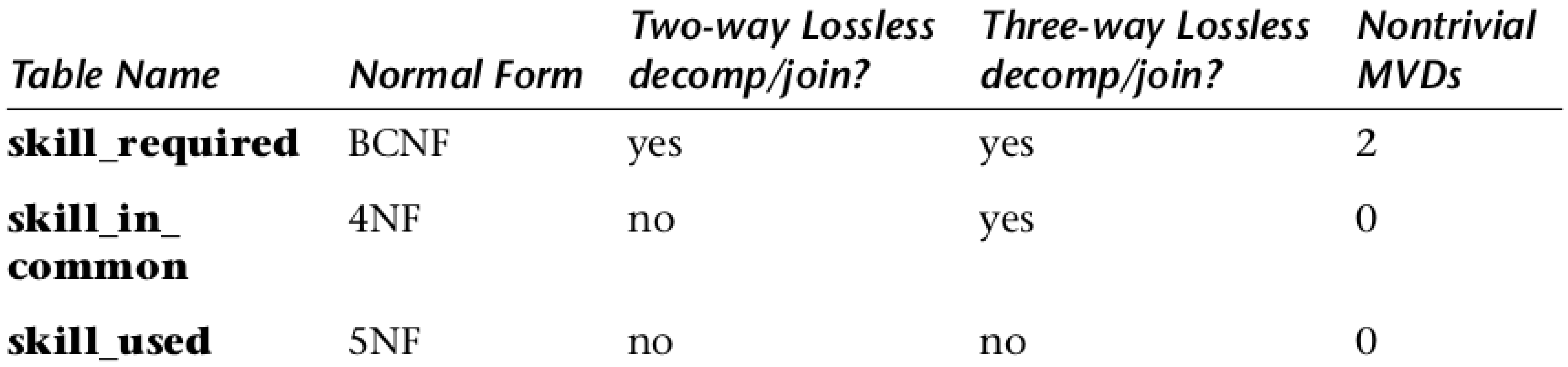


Table Name	Normal Form	Two-way Lossless decomp/join?	Three-way Lossless decomp/join?	Nontrivial MVDs
skill_required	BCNF	yes	yes	2
skill_in_ common	4NF	no	yes	0
skill_used	5NF	no	no	0

A many-to-many-to-many ternary relationship is:

- BCNF if it can be replaced by two binary relationships
- 4NF if it can only be replaced by three binary relationships
- 5NF if it cannot be replaced in any way (and thus is a true ternary relationship)