Applied Databases

Lecture 6
Normal Forms

Sebastian Maneth

University of Edinburgh - February 2nd, 2017

Outline

1. Second Normal Form (2NF)
2. Third Normal Form (3NF)
3. Boyce-Codd Normal Form (BCNF)
4. Fourth Normal Form (4NF)

Functional Dependency

X = { **X1,X2, ..,Xm** }
A = { **A1,A2, ..,Ak** }

non-empty sets of attribute names

Functional dependency

X → A: for every X-tuple, there is **exactly one** A-tuple across all rows

X1	X2	X3	. . .	A1	A2
1	1	2		4	5
1	1	2		4	6

NOT allowed!

Redundancy

Relation Schema R with functional dependency $X \rightarrow A$
has **fd-redundancy** (with respect to $X \rightarrow A$) if

(1) there exists a db instance D over R that satisfies $X \rightarrow A$

(2) there exist two distinct tuples in D that have equal (X, A)-values.

X	A	Z
1	2	5
1	2	6
1	2	7

fd: $\mathbf{X} \rightarrow \mathbf{A}$

Functional dependency
$X \rightarrow A$: for every X-tuple, there is exactly one A-tuple across all rows

Redundancy

Relation Schema R with functional dependency X → A
has **fd-redundancy** (with respect to X → A) if

(1) there exists a db instance D over R that satisfies X → A

(2) there exist two distinct tuples in D that have equal (X, A)-values.

X	A	Z
1	2	5
1	2	6
1	2	7

fd: **X** → **A**

X	Z
1	5
1	6
1	7

X	A
1	2

Redundancy

Relation Schema R with functional dependency X → A
has **fd-redundancy** (with respect to X → A) if

(1) there exists a db instance D over R that satisfies X → A

(2) there exist two distinct tuples in D that have equal (X, A)-values.

X	A	Z
1	2	5
1	2	6
1	2	7

fd: **X** → **A**

X	Z
1	5
1	6
1	7

X	A
1	2

→ *No redundancy*!
→ *Smaller*!
→ *Better*!

Redundancy

Relation Schema R with functional dependency $X \rightarrow A$
has **fd-redundancy** (with respect to $X \rightarrow A$) if

(1) there exists a db instance D over R that satisfies $X \rightarrow A$

(2) there exist two distinct tuples in D that have equal (X, A)-values.

It should be clear to you that **redundancy** leads to
update anomalies!

Give examples how redundancy causes
the three kinds of update anomalies!

It should be clear that update anomalies cause inconsistency.

Warm-Up

→ what are the superkeys of this table?

X	A	Z
1	2	5
1	2	6
1	2	7

Warm-Up

→ what are the superkeys of this table?

→ what are the candidate keys of the table?

X	A	Z
1	2	5
1	2	6
1	2	7

Warm-Up

→ what are the superkeys of this table?

→ what are the candidate keys of the table?

→ what are the non-prime attributes of the table?

X	A	Z
1	2	5
1	2	6
1	2	7

Warm-Up

→ what are the superkeys of this table?

→ what are the candidate keys of the table?

→ what are the non-prime attributes of the table?

X	A	Z
1	2	5
1	2	6
1	2	7

→ how many FD's are there in total for a table with 3 columns?

Second Normal Form (2NF)

A table is in 2NF, if

[Codd,1971]

→ it is in 1NF
→ every non-prime attribute *depends* on the whole of every candidate key

---

Example (Not 2NF)

Schema(R) = { <u>City, Street, HouseNumber</u>, HouseColor, CityPopulation }

1. { City, Street, HouseNumbe r} → { HouseColor }
2. { City } → { CityPopulation }
3. CityPopulation is non-prime
4. CityPopulation depends on { City } which is NOT the whole of the (unique) candidate key { City, Street, HouseNumber }

→ there is fd-redundancy wrt { City } → { CityPopulation }

Second Normal Form (2NF)

Bring a 1NF table into 2NF
→ move an attribute depending on a strict subset of a candidate key into a new table, together with this strict subset
→ the strict subset becomes the key of the new table

Example (Convert to 2NF)

Old Schema → { City, Street, HouseNumber, HouseColor, CityPopulation }

New Schema → {City, Street, HouseNumber, HouseColor }

New Schema → { City, CityPopulation }

fd-redundacy wrt { City, CityPopulation } removed!

Second Normal Form (2NF)

A table is in 2NF, if

→ it is in 1NF
→ every non-prime attribute *depends* on the whole of every candidate key

---

→ 2NF removes some fd-redundancies! :-)

Second Normal Form (2NF)

Electric Toothbrush Models

Manufacturer	**Model**	**Model Full Name**	**Manufacturer Country**
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

primary key

→ is the table in 2NF?

Second Normal Form (2NF)

Electric Toothbrush Models

Manufacturer	**Model**	**Model Full Name**	**Manufacturer Country**
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

candidate key

→ why is this a candidate key?

→ candidate key = a minimal superkey

means:
→ cannot be made smaller.
→ there can be **many** minimal superkeys!!

Second Normal Form (2NF)

Electric Toothbrush Models

Manufacturer	**Model**	**<u>Model Full Name</u>**	**Manufacturer Country**
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

candidate key

non-prime attribute

{ Manufacturer } → { Manufacturer Country }

Electric Toothbrush Models

Manufacturer	Model	Model Full Name	Manufacturer Country
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

Electric Toothbrush Manufacturers

Manufacturer	Manufacturer Country
Forte	Italy
Dent-o-Fresh	USA
Kobayashi	Japan
Hoch	Germany

Electric Toothbrush Models

Manufacturer	Model	Model Full Name
Forte	X-Prime	Forte X-Prime
Forte	Ultraclean	Forte Ultraclean
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush
Kobayashi	ST-60	Kobayashi ST-60
Hoch	Toothmaster	Hoch Toothmaster
Hoch	X-Prime	Hoch X-Prime

Second Normal Form (2NF)

A table is in 2NF, if

→ it is in 1NF
→ every non-prime attribute *depends* on the whole of every candidate key

X	A	Z
1	2	5
1	2	6
1	2	7

→ in 2NF!
→ 2NF fails to remove all redundancies!

fd: $\mathbf{A} \rightarrow \mathbf{X}$

Third Normal Form (3NF)

A table is in 3NF, if [Codd,1972]

→ it is in 2NF
→ every non-prime attribute *is non-transitively dependent* on every candidate key

---

X	A	Z
1	2	5
1	2	6
1	2	7

transitive dependency

→ not in 3NF!

fd's: **A** → **X**
Z → **A**

Third Normal Form (3NF)

A table is in 3NF, if

[Codd,1972]

→ it is in 2NF
→ every non-prime attribute *is non-transitively dependent* on every candidate key

---

"[Every] non-key [attribute] must provide a fact about the key,
the whole key, and nothing but the key."
"so help me Codd"

If $X \rightarrow A$ is nontrivial and A is non-key, then X must be a superkey!

Third Normal Form (3NF)

A table is in 3NF, if

[Codd,1972]

→ it is in 2NF
→ every non-prime attribute *is non-transitively dependent* on every candidate key

---

BuildingID	Contractor	Fee
100	Randolph	1200
150	Ingersoll	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200

Example (Not in 3NF)

Schema → {BuildingID, Contractor, Fee}

1. {BuildingID} → {Contractor}
2. {Contractor} → {Fee}
3. {BuildingID} → {Fee}
4. Fee transitively depends on the BuildingID
5. Both Contractor and Fee depend on the entire key hence 2NF

Third Normal Form (3NF)

Bring a 2NF table into 3NF:
→ move attribute involved in transitive dependency into a new table
→ identify a primary key for the new table
→ make this primary key a **foreign key** of the original table

BuildingID	Contractor	Fee
100	Randolph	1200
150	Ingersoll	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200

BuildingID	Contractor
100	Randolph
150	Ingersoll
200	Randolph
250	Pitkin
300	Randolph

Contractor	Fee
Randolph	1200
Ingersoll	1100
Pitkin	1100

foreign key

Foreign key = set of columns that references a set of columns of another table.
The purpose of the foreign key is to ensure **referential integrity**:
only values appearing in the referenced table are permitted.

Third Normal Form (3NF)

Tournament Winners

Tournament	**Year**	**Winner**	**Winner Date of Birth**
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

→ do you see any redundancy?

Third Normal Form (3NF)

Tournament Winners

<u>Tournament</u>	**<u>Year</u>**	**Winner**	**Winner Date of Birth**
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

→ do you see any redundancy?

Third Normal Form (3NF)

Tournament Winners

Tournament	**Year**	**Winner**	**Winner Date of Birth**
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

candidate key

non-prime attribute

{ Tournament, Year } → { Winner } → { Winner Date of Birth }

Tournament Winners

Tournament	Year	Winner	Winner Date of Birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

Tournament Winners

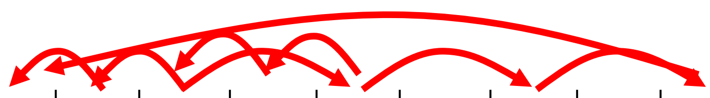
Tournament	Year	Winner
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Winner Dates of Birth

Winner	Date of Birth
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

Third Normal Form (3NF)

Bring a 2NF table into 3NF:
→ move attribute **involved in transitive dependency** into a new table
→ identify a primary key for the new table
→ make this primary key a foreign key of the original table



X	A	Z	B	C	D	E	F	G	H
1	2	5							
1	2	6							
1	2	7							

→ which one first?

→ in which order?

Third Normal Form (3NF)

"[Every] non-key [attribute] must provide a fact about the key,
the whole key, and nothing but the key."
"so help me Codd"

If $X \rightarrow A$ is nontrivial and A is non-key, then X must be a superkey!

PRO: can always find decomposition preserving dependencies

CONTRA: some redundancy may remain
$\rightarrow$ dependencies between prime attributes!

Boyce-Codd Normal Form (BCNF)

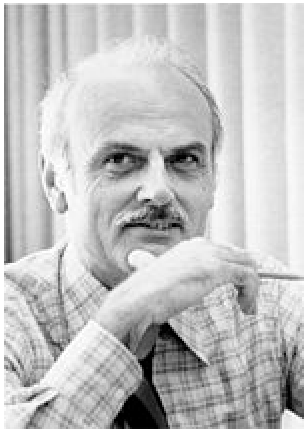
A table R is in BCNF, if for any dependency X → Y at least one of the following holds

→ (X → Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R.

(by Boyce and Codd 1974)

→ BCNF does not allow dependencies between prime attributes!

BCNF = “3NF + no dependencies between (distinct) prime attributes”

Boyce-Codd Normal Form (BCNF)

A table R is in BCNF, if for any dependency X → Y at least one of the following holds

→ (X → Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R.

(by Boyce and Codd 1974)

3NF and BCNF are (potentially) not the same, if these conditions hold:

1) The table has two or more candidate keys
2) At least two of the candidate keys are composed of more than one attribute
3) The keys are not disjoint i.e. The composite candidate keys share some attributes

Boyce-Codd Normal Form (BCNF)

A table R is in BCNF, if for any dependency X → Y at least one of the following holds

→ (X → Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R.

(by Boyce and Codd 1974)

Example (Not in BCNF)

Schema → {City, Street, ZipCode }

1. Key1 → { City, Street }
2. Key2 → { Street, ZipCode }
3. No non-key attribute hence 3NF
4. {City, Street} → {ZipCode}
5. {ZipCode} → {City}

Not a super key!

BCNF = "3NF + no dependencies between (distinct) prime attributes"

Assumes:
→ city cannot have two different streets with the same name

Boyce-Codd Normal Form (BCNF)

Bring table R into BCNF:

→ Place two candidate primary keys into separate tables
→ Place items in either of the tables, according to their dependencies on the keys

---

Example 1 (Convert to BCNF)

Old Schema → { City, Street, ZipCode }

New Schema1 → { Street, ZipCode }

New Schema2 → { City, Street }

➔ Loss of relation { ZipCode } → { City }

Alternate New Schem11 → { Street, ZipCode }

Alternate New Schema2 → { ZipCode, City }

➔ Loss of dependency { City, Street } → { ZipCode }

Boyce-Codd Normal Form (BCNF)

A table R is in BCNF, if for any dependency X → Y at least one of the following holds

→ (X → Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R.

(by Boyce and Codd 1974)

→ show how BCNF removes redundancy!

Boyce-Codd Normal Form (BCNF)

A table R is in BCNF, if for any dependency X → Y at least one of the following holds

→ (X → Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R.

(by Boyce and Codd 1974)

→ show how BCNF removes redundancy!

City	ZipCode	Street
LA	CA 90015	7th West
LA	CA 90015	8th West
LA	CA 90015	9th West

Boyce-Codd Normal Form (BCNF)

A table R is in BCNF, if for any dependency X → Y at least one of the following holds

→ (X → Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R.

(by Boyce and Codd 1974)

Good News

Lemma If R is a relation schema in BCNF

then there are **no fd-redundancies** in R

Boyce-Codd Normal Form (BCNF)

→ it can be guaranteed that no information is lost when moving to BCNF.

→ it cannot be guaranteed that some dependencies are lost (bad news)

Nearest Shops

Person	Shop Type	Nearest Shop
Davidson	Optician	Eagle Eye
Davidson	Hairdresser	Snippets
Wright	Bookshop	Merlin Books
Fuller	Bakery	Doughy's
Fuller	Hairdresser	Sweeney Todd's
Fuller	Optician	Eagle Eye

→ For each Person / Shop Type, the table tells which shop of that type is closest to the home of the person.

Candidate Keys

→ { Person, Shop Type }
→ { Person, Nearest Shop }

Not BCNF: { Nearest Shop } → { Shop Type }

→ 3NF because all attributes are prime

Nearest Shops

Person	Shop Type	Nearest Shop
Davidson	Optician	Eagle Eye
Davidson	Hairdresser	Snippets
Wright	Bookshop	Merlin Books
Fuller	Bakery	Doughy's
Fuller	Hairdresser	Sweeney Todd's
Fuller	Optician	Eagle Eye

→ bottom table is in BCNF!

→ problem: for a Person, may insert multiple Shops of the same type!

{Person, Shop Type} → {Nearest Shop} is lost!

Bad News

Shop Near Person

Person	Shop
Davidson	Eagle Eye
Davidson	Snippets
Wright	Merlin Books
Fuller	Doughy's
Fuller	Sweeney Todd's
Fuller	Eagle Eye

Shop

Shop	Shop Type
Eagle Eye	Optician
Snippets	Hairdresser
Merlin Books	Bookshop
Doughy's	Bakery
Sweeney Todd's	Hairdresser

Boyce-Codd Normal Form (BCNF)

- **Good**: no FD-redundancy
- **Bad**: FDs may be lost

$(\text{Book}[U], \Sigma)$
$U = \{\textbf{Course}, \textbf{Lecturer}, \textbf{Hour}\}$
$\Sigma = \{\mathbf{C} \to \mathbf{L}, \mathbf{LH} \to \mathbf{C}\}$

- Algorithm gives $\{(\mathbf{CL}, \{\mathbf{C} \to \mathbf{L}\}), (\mathbf{CH}, \emptyset)\}$
- $\mathbf{LH} \to \mathbf{C}$ is lost (*always*)

Fourth Normal Form (4NF)

Studio	**Director**	**Loc**
Universal	Lynch	LA
Universal	Lynch	NY
Universal	Lynch	SF
Universal	Cubrick	LA
Universal	Cubrick	NY
Universal	Cubrick	SF
Disney	Whedon	LA

Multi-valued dependency (MVD): $\mathbf{S} \twoheadrightarrow \mathbf{L}$

MVD $X \twoheadrightarrow Y$:
$I = [XY] \bowtie [X(U - Y)]$

(s, d, l) studio s employs director d and studio s has an office in l

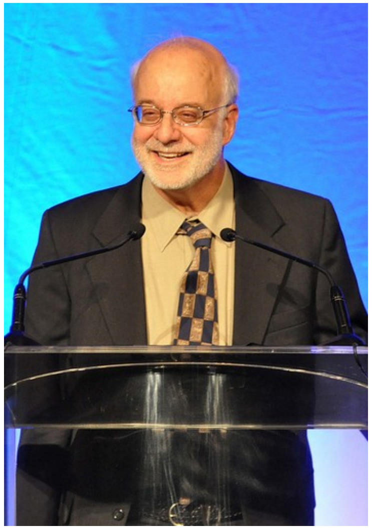
- BCNF? – Yes! (no nontrivial FDs)
- MVD-Redundancy – **Director** & **Loc** are *independent*

Fourth Normal Form (4NF)

A table R is in 4NF, if for every multi-valued dependency (mvd) X -->> Y,

→ (X -->> Y) is trivial (i.e., Y is a subset of X, or, X union Y are all attributes)
→ X is a superkey for R

[Fagin,1977]



R has multi-valued dependency (mvd) X -->> Y

If two tuples agree on all attributes in X, then their Y-values may be swapped, and the resulting two tuples must in R as well.

Note X → Y implies X -->> Y. Do you see why?

Fourth Normal Form (4NF)

A table R is in 4NF, if for every multi-valued dependency (mvd) X -->> Y,

→ (X -->> Y) is trivial (i.e., Y is a subset of X, or, X union Y are all attributes)
→ X is a superkey for R

[Fagin,1977]

Example (Not in 4NF)

Schema → {MovieName, ScreeningCity, Genre)

Primary Key: {MovieName, ScreeningCity, Genre)

1. All columns are a part of the only candidate key, hence BCNF
2. Many Movies can have the same Genre
3. Many Cities can have the same movie
4. Violates 4NF

Movie	ScreeningCity	Genre
Hard Code	Los Angles	Comedy
Hard Code	New York	Comedy
Bill Durham	Santa Cruz	Drama
Bill Durham	Durham	Drama
The Code Warrier	New York	Horror

Fourth Normal Form (4NF)

A table R is in 4NF, if for every multi-valued dependency (mvd) X -->> Y,

→ (X -->> Y) is trivial (i.e., Y is a subset of X, or, X union Y are all attributes)
→ X is a superkey for R

[Fagin,1977]

Example (Not in 4NF)

Schema → {MovieName, ScreeningCity, Genre)

Primary Key: {MovieName, ScreeningCity, Genre)

1. All columns are a part of the only candidate key, hence BCNF
2. Many Movies can have the same Genre
3. Many Cities can have the same movie
4. Violates 4NF

No!!

If Movie → Genre
then
not in BCNF!!!

Movie	ScreeningCity	Genre
Hard Code	Los Angles	Comedy
Hard Code	New York	Comedy
Bill Durham	Santa Cruz	Drama
Bill Durham	Durham	Drama
The Code Warrier	New York	Horror

Fourth Normal Form (4NF)

A table R is in 4NF, if for every multi-valued dependency (mvd) X -->> Y,

→ (X -->> Y) is trivial (i.e., Y is a subset of X, or, X union Y are all attributes)
→ X is a superkey for R

[Fagin,1977]

Example (Not in 4NF)

Schema → {MovieName, ScreeningCity, Genre)

Primary Key: {MovieName, ScreeningCity, Genre)

1. *No dependencies between prime attributes*, hence BCNF
2. Many Movies can have the same Genre
3. *A Move can have many Genres*
4. Many Cities can have the same movie
5. Violates 4NF

Movie	ScreeningCity	Genre
Hard Code	Los Angles	Comedy
Hard Code	New York	Comedy
Bill Durham	Santa Cruz	Drama
Bill Durham	Durham	Drama
The Code Warrier	New York	Horror

Fourth Normal Form (4NF)

Example 2 (Not in 4NF)

Schema → {Manager, Child, Employee}

1. Primary Key → {Manager, Child, Employee}
2. Each manager can have more than one child
3. Each manager can supervise more than one employee
4. 4NF Violated

Manager	Child	Employee
Jim	Beth	Alice
Mary	Bob	Jane
Mary	Bob	Adam

Example 3 (Not in 4NF)

Schema → {Employee, Skill, ForeignLanguage}

1. Primary Key → {Employee, Skill, Language }
2. Each employee can speak multiple languages
3. Each employee can have multiple skills
4. Thus violates 4NF

Employee	Skill	Language
1234	Cooking	French
1234	Cooking	German
1453	Carpentry	Spanish
1453	Cooking	Spanish
2345	Cooking	Spanish

Fourth Normal Form (4NF)

Bring a BCNF table into 4NF:
→ Move the two multi-valued sub-relations into separate tables
→ Identify primary keys for each new table.

Example 1 (Convert to 3NF)

Old Schema → {MovieName, ScreeningCity, Genre}

New Schema → {MovieName, ScreeningCity}

New Schema → {MovieName, Genre}

Movie	ScreeningCity	Genre
Hard Code	Los Angles	Comedy
Hard Code	New York	Comedy
Bill Durham	Santa Cruz	Drama
Bill Durham	Durham	Drama
The Code Warrier	New York	Horror

Movie	Genre
Hard Code	Comedy
Bill Durham	Drama
The Code Warrier	Horror

Movie	ScreeningCity
Hard Code	Los Angles
Hard Code	New York
Bill Durham	Santa Cruz
Bill Durham	Durham
The Code Warrier	New York

Example 2 (Convert to 4NF)

Old Schema → {Manager, Child, Employee}

New Schema → {Manager, Child}

New Schema → {Manager, Employee}

Manager	Child
Jim	Beth
Mary	Bob

Manager	Employee
Jim	Alice
Mary	Jane
Mary	Adam

Example 3 (Convert to 4NF)

Old Schema → {Employee, Skill, ForeignLanguage}

New Schema → {Employee, Skill}

New Schema → {Employee, ForeignLanguage}

Employee	Skill
1234	Cooking
1453	Carpentry
1453	Cooking
2345	Cooking

Employee	Language
1234	French
1234	German
1453	Spanish
2345	Spanish

Fourth Normal Form (4NF)

Do not underestimate importance of 4NF:

→ [Wu 1992] of real word databases, 20% were NOT in 4NF!

Redundancy

Relation Schema R with multi-valued dependency X -->> A
has **mvd-redundancy** (with respect to X -->> A) if

(1) there exists a db instance D over R that satisfies X -->> A

(2) there exist two distinct tuples in D that have equal (X, A)-values.

X	A	Z
1	3	7
1	4	9
1	4	7
1	3	9

mvd: **X** -->> **A**

Redundancy

Relation Schema R with multi-valued dependency X -->> A
has **mvd-redundancy** (with respect to X -->> A) if

(1) there exists a db instance D over R that satisfies X -->> A

(2) there exist two distinct tuples in D that have equal (X, A)-values.

X	A	Z
1	3	7
1	4	9
1	4	7
1	3	9

mvd: **X -->> A**

X	A
1	3
1	4

X	Z
1	7
1	9

→ *No redundancy*!
→ *Smaller*!
→ *Better*!

Redundancy

Relation Schema R with multi-valued dependency X -->> A
has **mvd-redundancy** (with respect to X -->> A) if

(1) there exists a db instance D over R that satisfies X -->> A

(2) there exist two distinct tuples in D that have equal (X, A)-values.

Good News

Lemma If R is a relation schema in 4NF,
then there are no mvd-redundancies in R

Challenge

Something challenging for you to think about:

Imagine a program that checks if a given relation schema is

→ in BCNF
→ in 4NF

and if not, it suggests a new schema in normal form.

Questions: → how expensive are such checks? (in terms of bigO)
→ how to makes sure no information is lost?
→ how to signal fd's that are lost?

END
Lecture 6