DSAA 5012
Advanced Data Management for Data Science

LECTURE 10
RELATIONAL DATABASE DESIGN: NORMALIZATION

RELATIONAL DATABASE DESIGN: OUTLINE

- ✓ Functional Dependencies
  - Definition
  - Functional Dependencies and Keys
  - Inference Rules for Functional Dependencies
  - Closure of Attribute Sets
  - Canonical Cover

- ➡ **Normalization**
  - **Goals**
  - **First Normal Form (1NF)**
  - **Second Normal Form (2NF)**
  - **Third Normal Form (3NF)**
  - **Boyce-Codd Normal Form (BCNF)**

RECALL

RELATIONAL DATABASE DESIGN AND FUNCTIONAL DEPENDENCIES

- Relational database design requires that we find a "good" collection of relation schemas.

  ☞ **A bad design may lead to several problems.**

- Functional dependencies can be used to refine a relation schema reduced from an E-R schema by iteratively decomposing it (called normalization) to place it in a certain normal form.
  - The first four normal forms ⟹ use only functional dependencies.
  - Additional normal forms ⟹ use other types of dependencies

☞ **Normal forms do not guarantee a good design!**

NORMALIZATION

- Normalization decomposes unsatisfactory relation schemas into fragments (i.e., breaks them up into two or more relation schemas) that possess more desirable properties.
  ☞ **Eliminates data redundancy and update anomalies, preserves dependencies and is lossless.**

- Normalization provides a series of tests for relation schemas.
  ☞ **If a relation schema fails the test, then we need to decompose it.**

- Normalization is expressed in terms of normal forms:
  - the first four normal forms ⟹ use only FDs
  - additional normal forms ⟹ use other types of dependencies

☞ **Normal forms do not guarantee a good design!**

NORMALIZATION: GOALS

Design Guideline 1: Clear Semantics for Attributes

Design a relation schema so that it is easy to explain its meaning. Typically, this means that we should not combine attributes from multiple real-world entities in a single relation schema.

☞ **Each relation schema should have a well-defined, unambiguous meaning.**

Design Guideline 2: Minimize Use of Null Values

As far as possible, avoid placing attributes in relation schemas whose values may be null. If nulls are unavoidable, make sure that they apply in exceptional cases only.

☞ **Null values lead to problems of understanding the meaning of attributes and specifying certain operations (e.g., aggregation operations).**

NORMALIZATION: GOALS (cont'd)

Design Guideline 3: Minimize Redundancy

Design relation schemas so that no insertion, deletion or update anomalies occur in the relation instances. If any update anomalies are present, note them clearly so that update programs will operate correctly.

☞ **A relation schema has redundancy if there is an FD where the LHS is not a key.**

☞ **Redundant data in relations can cause operation anomalies.**

Operation Anomalies

- insertion (e.g., insert license fee for engine size 5)
- deletion (e.g., delete instance “BMW, 7.35i, …”)
- update (e.g., update license fee for engine size 1)

Car

make	model	engineSize	fee	origin	tax
Nissan	Sunny	1	4,000	Japan	90
Fiat	Mirafiori	1	4,000	Italy	85
Honda	Accord	1	4,000	Japan	90
Toyota	Camry	4	7,000	Canada	50
Ford	Mustang	4	7,000	Canada	50
Ford	Mustang	2	5,000	U.S.A.	75
BMW	7.35i	3	6,000	Germany	95
Toyota	Camry	1	4,000	Japan	90

NORMALIZATION: GOALS (cont'd)

Design Guideline 4: Lossless Decomposition

The normalized relation schemas should contain the same information as the original schema; otherwise, decomposition results in information loss.

- A decomposition is lossless (aka lossless join) if the original relation instance can be recovered from the schema fragments.
  - Joining all the fragments results in exactly the original relation instance.
- In general, a decomposition of R into R_1 and R_2 is lossless *if and only if* at least one of the following FDs is in F^+:

$$R_1 \cap R_2 \rightarrow R_1 \qquad R_1 \cap R_2 \rightarrow R_2$$

☞ **The common attributes of R_1 and R_2 must be a superkey for R_1 *or* R_2.**

LOSSY DECOMPOSITION EXAMPLE

Decompose R(A, B, C) into R_1(A, B) and R_2(B, C).

R

A	B	C
a	1	*m*
a	2	*n*
b	1	*p*

decompose →

R_1

A	B
a	1
a	2
b	1

R_2

B	C
1	*m*
2	*n*
1	*p*

R_1 JOIN R_2 on B

A	B	C
a	1	*m*
a	1	*p*
a	2	*n*
b	1	*m*
b	1	*p*

The *decomposition is lossy* since the join produces two extra tuples. Thus, the decomposition "loses" some information! Note that the common attribute B is not a superkey of either R_1 or R_2.

NORMALIZATION: GOALS (cont'd)

Design Guideline 5: Preserve Functional Dependencies

As far as possible, functional dependencies should be preserved within each relation schema; otherwise, checking updates for violation of functional dependencies may require computing joins, which is expensive.

- Functional dependencies represent real-world constraints.
- If a functional dependency does not appear in any relation schema (i.e., it is "lost"), the constraint may be much more difficult to enforce.
- The decomposition of a relation schema R with FDs F is a set of schema fragments R_i with FDs F_i.
  - F_i is the subset of dependencies in F^+ (the closure of F) that involves only attributes in R_i.
- The decomposition is dependency preserving *if and only if* $(\cup F_i)^+ = F^+$.
  - Every FD in F is present in some fragment R_i.

DECOMPOSITION EXAMPLE:
NON-DEPENDENCY PRESERVING

R(A, B, C) **Key:** A F = {A→B, B→C, A→C} $F=F^{+}$

For FD B→C, LHS is not a key ⟹ R can have considerable redundancy.

Solution: Break R into relations R_1(A, B), R_2(A, C) (normalization).

X Invalid! Violates B→C.

R

A	B	C
1	2	3
2	2	3
3	2	3
4	2	4

decompose →

✓ Valid! No FD violation.

R_1

A	B
1	2
2	2
3	2
4	2

R_2

A	C
1	3
2	3
3	3
4	4

✓ Valid! No FD violation.

The decomposition is lossless since the common attribute A is a key for R_1 (and R_2).

The decomposition is not dependency preserving because F_1 = {A→B}, F_2 = {A→C} and $(F_1 \cup F_2)^{+} \neq F^{+}$. **The FD B→C is lost.**

In practice, each "lost" FD is implemented as an assertion (a type of constraint), which is checked when there are updates. Thus, to find violations on B→C, R_1 and R_2 must be joined, which can be very expensive.

DECOMPOSITION EXAMPLE: DEPENDENCY PRESERVING

R(A, B, C)   **Key:** A F = {A→B, B→C, A→C} $F = F^+$

Break R into relations R_1(A, B), R_2(B, C).

R

A	B	C
1	2	3
2	2	3
3	2	3
4	2	4

X Invalid! Violates B→C.

decompose →

✓ Valid! No FD violation.

R_1

A	B
1	2
2	2
3	2
4	2

R_2

B	C
2	3
2	4

X Invalid! Violates B→C.

The decomposition is lossless since the common attribute B is a key for R_2.

The decomposition is dependency preserving because F_1 = {A→B}, F_2 = {B→C} and $(F_1 \cup F_2)^+ = F^+$ (since A→B, B→C ⊨ A→C).

Violations of the FDs can be found by inspecting the individual tables, without performing a join.

☞ **How a relation is decomposed, may determine whether functional dependencies are preserved.**

RELATIONAL DATABASE DESIGN: NORMALIZATION

EXERCISES 1, 2

EXERCISE 1

Given: R(A, B, C, D, E) F = {A→BC}

Decomposition: R_1(A, B, C) and R_2(A, D, E)

a) Is the decomposition lossless? Why? (*iff* $R_1 \cap R_2 \rightarrow R_1$ or $R_1 \cap R_2 \rightarrow R_2$)

Yes The common attribute A is a key for R_1.

b) Is the decomposition dependency preserving? Why? (*iff* $(\cup F_i)^+ = F^+$)

Yes A→BC is preserved in R_1.

c) Is the decomposition R_1(A, B, C) and R_2(C, D, E) lossless? Why?

No C is not a key for any table.

EXERCISE 2

Given: R(A, B, C, D, E) F = {A→BC, CD→E, B→D, E→A}

Decomposition: R_1(A, B, C) and R_2(A, D, E)

a) Is the decomposition lossless? (*iff* $R_1 \cap R_2 \rightarrow R_1$ or $R_1 \cap R_2 \rightarrow R_2$)

Yes The common attribute A is a key for R_1.

b) Is the decomposition dependency preserving? (*iff* $(\cup F_i)^+ = F^+$)

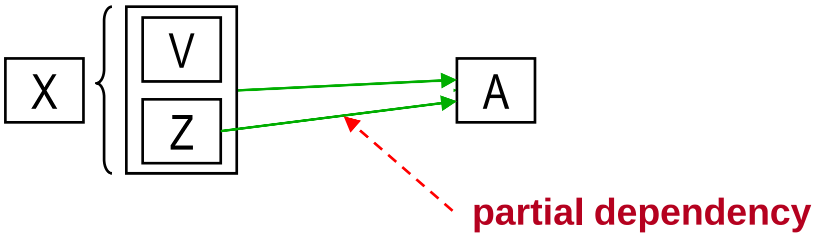
No We lose CD→E and B→D.

NORMALIZATION:
SOME DEFINITIONS AND VISUAL AIDS

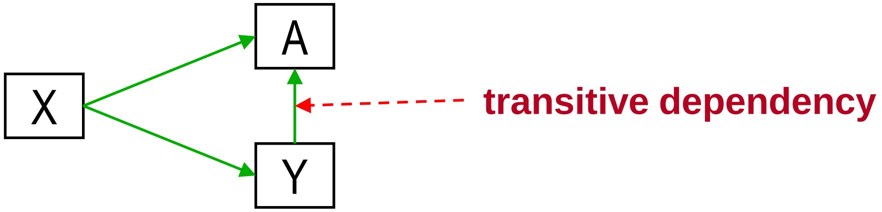
Prime and non-prime attributes

An attribute is a prime attribute if it is part of any candidate key. Otherwise, it is a non-prime attribute.

Partial dependency visualization



Transitive dependency visualization



NORMALIZATION:
EXAMPLE RELATION SCHEMA & DATABASE

Car

make	model	engineSize	fee	origin	tax
Nissan	Sunny	1	4,000	Japan	90
Fiat	Mirafiori	1	4,000	Italy	85
Honda	Accord	1	4,000	Japan	90
Toyota	Camry	4	7,000	Canada	50
Ford	Mustang	4	7,000	Canada	50
Ford	Mustang	2	5,000	U.S.A.	75
BMW	7.35i	3	6,000	Germany	95
Toyota	Camry	1	4,000	Japan	90

Functional Dependencies

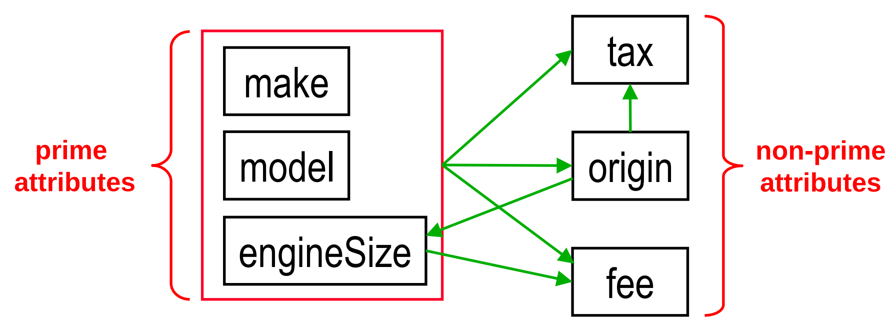
make, model, engineSize $\rightarrow$ origin
make, model, engineSize $\rightarrow$ tax
make, model, engineSize $\rightarrow$ fee

due to the primary key

origin $\rightarrow$ tax
engineSize $\rightarrow$ fee
origin $\rightarrow$ engineSize

from real-world knowledge

FD visualization



FIRST NORMAL FORM (1NF)

A relation schema is in *First Normal Form (1NF)* if all attributes are atomic (single-valued).

☞ **There are no multi-valued or composite attributes.**

- Relation schemas are always in 1NF according to the definition of the relational model and according to our strategy for reducing an E-R schema to relation schemas.

SECOND NORMAL FORM (2NF)

A relation schema is in *Second Normal Form (2NF)* if all non-prime attributes are <u>fully</u> functionally dependent on every candidate key.

- R is a relation schema, with the set F of FDs.
- R is in 2NF *if and only if*

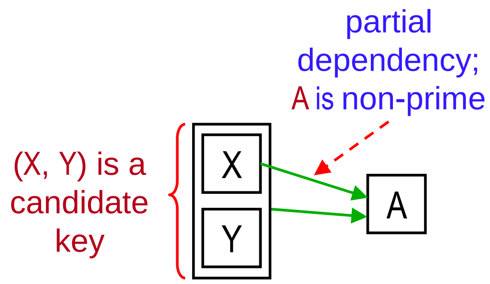
For each FD: X→A in F^+:

A ∈ X *(the FD is trivial)* ***or***

(for LHS) X is not a proper subset of a candidate key for R ***or***

(for RHS) A is a prime attribute for R.

☞ **A subset of a candidate key cannot determine a non-prime attribute.**



SECOND NORMAL FORM (2NF) EXAMPLE

– make, model, engineSize is a candidate key (it is not a proper subset).

– engineSize is a proper subset of a candidate key.

– fee is a non-prime attribute.

– Hence, the relation schema is not in 2NF due to the FD engineSize→fee.

Note redundancy.

Car

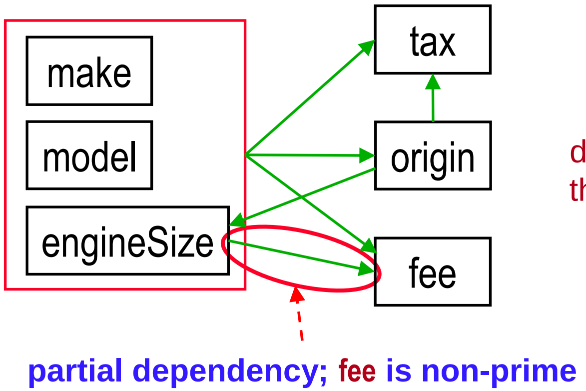
make	model	engineSize	fee	origin	tax
Nissan	Sunny	1	4,000	Japan	90
Fiat	Mirafiori	1	4,000	Italy	85
Honda	Accord	1	4,000	Japan	90
Toyota	Camry	4	7,000	Canada	50
Ford	Mustang	4	7,000	Canada	50
Ford	Mustang	2	5,000	U.S.A.	75
BMW	7.35i	3	6,000	Germany	95
Toyota	Camry	1	4,000	Japan	90

Recall Operation Anomalies

- insertion (e.g., insert license fee for engine size 5)
- deletion (e.g., delete instance "BMW, 7.35i, …")
- update (e.g., update license fee for engine size 1)

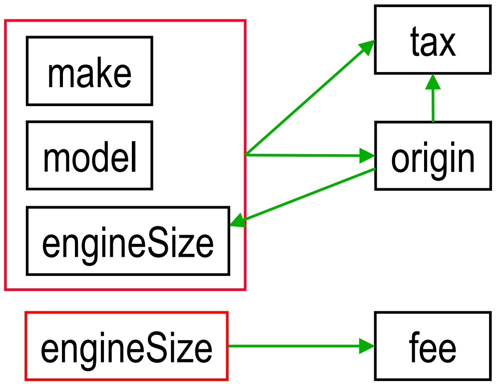
SECOND NORMAL FORM (2NF) EXAMPLE (cont'd)

FDs in original schema



decompose the schema

FDs in 2NF schemas



decompose the tables

The decomposition resolves the previous anomalies.

Car

make	model	engineSize	origin	tax
Nissan	Sunny	1	Japan	90
Fiat	Mirafiori	1	Italy	85
Honda	Accord	1	Japan	90
Toyota	Camry	4	Canada	50
Ford	Mustang	4	Canada	50
Ford	Mustang	2	U.S.A.	75
BMW	7.35i	3	Germany	95
Toyota	Camry	1	Japan	90

Licensing

engineSize	fee
1	4000
2	5000
3	6000
4	7000

RELATIONAL DATABASE DESIGN: NORMALIZATION

EXERCISE 3

EXERCISE 3

2NF
R is in 2NF *if and only if*
For each FD: X→A in F^+:
A ∈ X (*trivial FD*) ***or***
X is not a proper subset of a candidate key for R ***or***
A is a prime attribute for R.

a) Given: R(A, B, C, D) F = {AB→CD, B→C}

Is R in 2NF? Why?

Key: AB AB$^+$={A, B, C, D} B$^+$={B, C}

No For B→C, B *is* a proper subset of the key AB *and* C is non-prime.
So, R is not in 2NF.

b) Given: R(A, B, C, D) F = {AB→CD, C→D}

Is R in 2NF? Why?

Key: AB AB$^+$={A, B, C, D} C$^+$={C, D}

Yes For C→D, C *is not* a proper subset of the key, so R is in 2NF.

THIRD NORMAL FORM (3NF)

A relation schema is in *Third Normal Form (3NF)* if it is in 2NF and every non-prime attribute is non-transitively dependent on every candidate key.

- R is a relation schema, with set F of FDs.

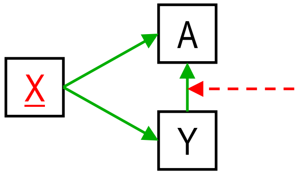
- R is in 3NF *if and only if*

  For each FD: X→A in F^+:

  A ∈ X (*trivial FD*) ***or***

  (for LHS) X is a superkey for R ***or***

  (for RHS) A is a prime attribute for R.



transitive dependency;
Y is not a superkey;
A is non-prime

☞ **For every FD that does not contain extraneous attributes either:**

- the LHS is a candidate key, ***or***
- the RHS is a prime attribute (i.e., it is part of some candidate key).

THIRD NORMAL FORM (3NF) EXAMPLE

– For the FD origin→tax, origin is not a superkey.

– tax is not a prime attribute.

– Hence, the relation schema is not in 3NF due to the FD origin→tax.

Note redundancy.

Car

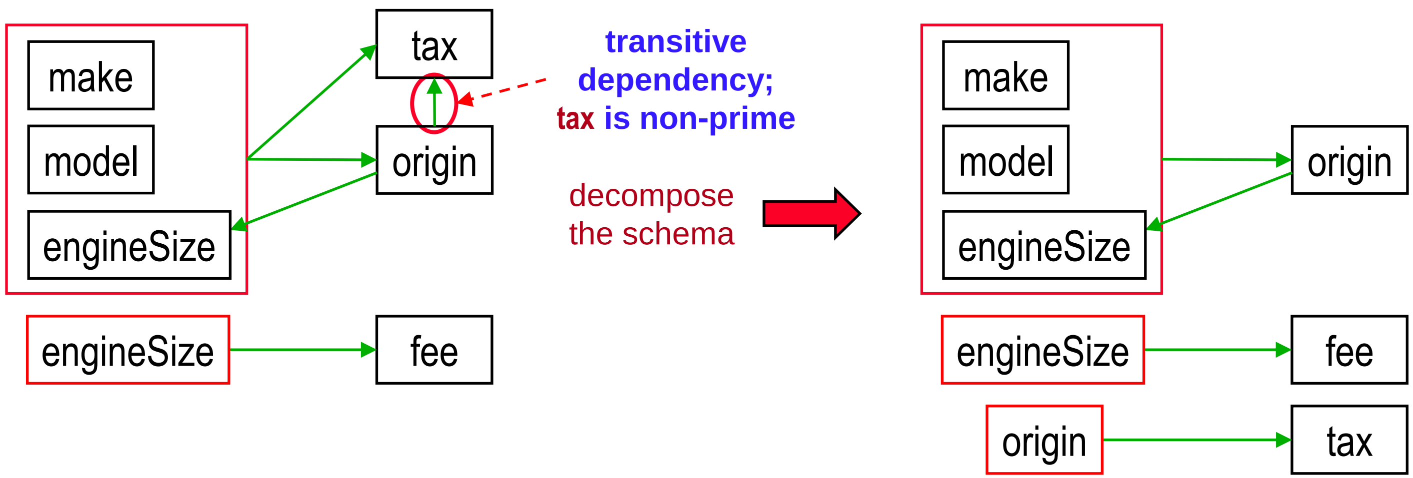
make	model	engineSize	origin	tax
Nissan	Sunny	1	Japan	90
Fiat	Mirafiori	1	Italy	85
Honda	Accord	1	Japan	90
Toyota	Camry	4	Canada	50
Ford	Mustang	4	Canada	50
Ford	Mustang	2	U.S.A.	75
BMW	7.35i	3	Germany	95
Toyota	Camry	1	Japan	90

Operation Anomalies

- insertion (e.g., insert tax rate 40 for Australia)
- deletion (e.g., delete instance "BMW, 7.35i, …")
- update (e.g., update tax rate for origin Japan)

THIRD NORMAL FORM (3NF) EXAMPLE (cont'd)

FDs in 2NF schemas | **FDs in 3NF schemas**



decompose the tables

The decomposition resolves the previous anomalies.

Car

make	model	engineSize	origin
Nissan	Sunny	1	Japan
Fiat	Mirafiori	1	Italy
Honda	Accord	1	Japan
Toyota	Camry	4	Canada
Ford	Mustang	4	Canada
Ford	Mustang	2	U.S.A.
BMW	7.35i	3	Germany
Toyota	Camry	1	Japan

ImportTax

origin	tax
Canada	50
U.S.A.	75
Italy	85
Japan	90
Germany	95

Licensing

engineSize	fee
1	4000
2	5000
3	6000
4	7000

If none of the decomposed relations contains a candidate key of the original relation, then add a relation containing one of the candidate keys.

3NF DECOMPOSITION ALGORITHM

Let R be the initial relation schema with FDs F.

Compute a canonical cover F_c of F.

S = ∅ (S is a set of relation schemas)

For each FD X→Y in the canonical cover F_c

S = S ∪ (X, Y) (for each FD create a relation schema; add it to S)

If no schema contains a candidate key for R

Choose any candidate key K

S = S ∪ K (add any candidate key as a relation schema)

☞ **The algorithm always creates a lossless-join, dependency preserving, 3NF decomposition.**

(Also called the 3NF Synthesis Algorithm.)

RELATIONAL DATABASE DESIGN: NORMALIZATION

EXERCISE 4

EXERCISE 4

Identify the candidate key(s) and the current highest normal form for each of the following relation schemas given their corresponding FDs.

a) R(A, B, C, D, E) $F = \{A \rightarrow B, C \rightarrow D\} = F^+$

$A^+ = \{A, B\}$ $C^+ = \{C, D\}$

Candidate keys: ACE

⟹ For A→B and C→D

i. A and C ***are*** proper subsets of the candidate key ACE (both FDs fail 1st 2NF test).

ii. both B and D ***are not*** prime attributes of R (both FDs fail 2nd 2NF test).

☞ Both FDs violate 2NF.

Normal form: 1NF

2NF
R is in 2NF *if and only if*
For each FD: X→A in F^+:
A ∈ X (*trivial FD*) ***or***
X is not a proper subset of a candidate key for R ***or***
A is a prime attribute for R.

3NF
R is in 3NF *if and only if*
For each FD: X→A in F^+:
A ∈ X (*trivial FD*) ***or***
X is a superkey for R ***or***
A is a prime attribute for R.

EXERCISE 4 (cont'd)

Identify the candidate key(s) and the current highest normal form for each of the following relation schemas given their corresponding FDs.

b) R(A, B, C) $F = \{AB \rightarrow C, C \rightarrow B\} = F^+$

$AB^+ = \{A, B, C\}$ $C^+ = \{C, B\}$

Candidate keys: AB, AC

⟹ For AB→C,
C *is* a prime attribute of R
(FD passes 2nd 2NF and 3NF tests).

⟹ For C→B,
B *is* a prime attribute of R
(FD passes 2nd 2NF and 3NF tests).

☞ Both FDs satisfy 3NF.

Normal form: 3NF

2NF
R is in 2NF *if and only if*
For each FD: X→A in F^+:
A ∈ X (*trivial FD*) ***or***
X is not a proper subset of a candidate key for R ***or***
A is a prime attribute for R.

3NF
R is in 3NF *if and only if*
For each FD: X→A in F^+:
A ∈ X (*trivial FD*) ***or***
X is a superkey for R ***or***
A is a prime attribute for R.

EXERCISE 4 (cont'd)

Identify the candidate key(s) and the current highest normal form for each of the following relation schemas given their corresponding FDs.

c) R(A, B, C, F) $\quad$ F = {AB→C, C→F} = F^+

AB^+={A, B, C, F} $\quad$ C^+={C, F}

Candidate keys: AB

⟹ For AB→C

i. AB *is not* a proper subset of a candidate key (FD passes 1st 2NF test);

ii. AB is a superkey for R (FD passes 1st 3NF test).

⟹ For C→F

i. C *is not* a proper subset of a candidate key (FD passes 1st 2NF test);

ii. C is not a superkey of R (FD fails 1st 3NF test);

iii. F *is not* a prime attribute (FD fails 2nd 3NF test).

☞ Both FDs satisfy 2NF.

Normal form: 2NF

2NF

R is in 2NF *if and only if*

For each FD: X→A in F^+:

- A ∈ X (*trivial FD*) ***or***
- X is not a proper subset of a candidate key for R ***or***
- A is a prime attribute for R.

3NF

R is in 3NF *if and only if*

For each FD: X→A in F^+:

- A ∈ X (*trivial FD*) ***or***
- X is a superkey for R ***or***
- A is a prime attribute for R.

BOYCE-CODD NORMAL FORM (BCNF)

A relation schema is in *Boyce-Codd Normal Form (BCNF)* if every determinant (left hand side) of its FDs is a superkey.

- R is a relation schema, with the set F of FDs.
- R is in BCNF *if and only if*

  For each FD: X→A in F^+:
  A ∈ X (*trivial FD*) ***or***
  X is a superkey for R.

☞ **For every FD that does not contain extraneous attributes, the LHS is a candidate key.**

- BCNF tables have no redundancy (that can be removed by FDs).
- If a table is in BCNF it is also in 3NF (and 2NF and 1NF).

BOYCE-CODD NORMAL FORM (BCNF): EXAMPLE

– For the FD origin→engineSize, origin is not a superkey.

– Hence, this relation schema is not in BCNF due to the FD origin→engineSize.

Note redundancy.

Car

make	model	engineSize	origin
Nissan	Sunny	1	Japan
Fiat	Mirafiori	1	Italy
Honda	Accord	1	Japan
Toyota	Camry	4	Canada
Ford	Mustang	4	Canada
Ford	Mustang	2	U.S.A.
BMW	7.35i	3	Germany
Toyota	Camry	1	Japan

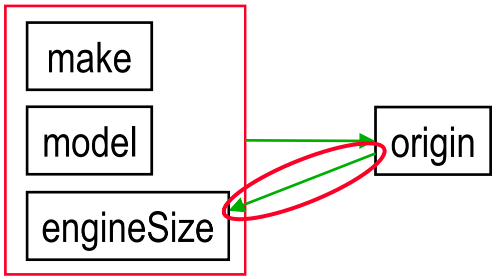
Note: Need to use null values if we want to represent an engine size and origin, but do not know the make and model.

Operation Anomalies

- insertion (e.g., insert engine size 5 from Korea)
- deletion (e.g., delete all instances "Ford, Mustang, …")
- update (e.g., update engine size for Japan)

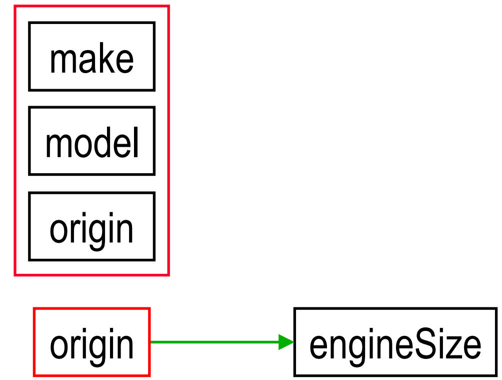
BOYCE-CODD NORMAL FORM (BCNF): EXAMPLE (cont'd)

FDs in 3NF schema



decompose the schema

FDs in BCNF schemas



decompose the tables

Car

make	model	origin
Nissan	Sunny	Japan
Fiat	Mirafiori	Italy
Honda	Accord	Japan
Toyota	Camry	Canada
Ford	Mustang	Canada
Ford	Mustang	U.S.A.
BMW	7.35i	Germany
Toyota	Camry	Japan

Country

origin	engineSize
Italy	1
Canada	4
U.S.A.	2
Germany	3
Japan	1

This decomposition avoids the 3NF problems of redundancy and null values.

BOYCE-CODD NORMAL FORM (BCNF): EXAMPLE (cont'd)

- We can generate the original relation instance by joining the two fragments, using a full outer join.

Car

make	model	origin
Nissan	Sunny	Japan
Fiat	Mirafiori	Italy
Honda	Accord	Japan
Toyota	Camry	Canada
Ford	Mustang	Canada
Ford	Mustang	U.S.A.
BMW	7.35i	Germany
Toyota	Camry	Japan

⟗

Country

origin	engineSize
Italy	1
Canada	4
U.S.A.	2
Germany	3
Japan	1

=

Car

make	model	engineSize	origin
Nissan	Sunny	1	Japan
Fiat	Mirafiori	1	Italy
Honda	Accord	1	Japan
Toyota	Camry	4	Canada
Ford	Mustang	4	Canada
Ford	Mustang	2	U.S.A.
BMW	7.35i	3	Germany
Toyota	Camry	1	Japan

Is the decomposition dependency preserving?

No. We lose the FD make, model, engineSize→origin.

A relation may not have a dependency preserving BCNF decomposition!

Can we have a dependency preserving decomposition?

No. However, the relation schema is decomposed, make, model, engineSize→origin is lost since it involves all the attributes of the original 3NF Car relation.

BCNF DECOMPOSITION ALGORITHM

Let R be the initial relation schema with set of FDs F.

Compute F^+

S = R

Until all relation schemas in S are in BCNF

  For each R in S

    For each FD X→Y that violates BCNF for R

      S = (S – R) ∪ (R – Y) ∪ (X, Y)

End until

- When a relation schema R with BCNF violation X→Y is found:
  1. Remove R from S, the set of relation schemas.
  2. Add a schema that has the same attributes as R except for Y (i.e., remove the RHS of the FD from R).
  3. Add a second schema that contains the attributes in X and Y (i.e., create a relation schema for the FD X→Y).

BCNF DECOMPOSITION EXAMPLE

Given: R(A, B, C, D, E) F = {A→BE, C→D} (Note: F=F^+)

Candidate key: AC

- Both functional dependencies violate BCNF because the LHS is not a candidate key.
- Pick A→BE
  - We can also choose C→D ⟹ different choices may lead to different decompositions.
- R(A, B, C, D, E) generates: R_1(<u>A, C</u>, D) (remove the RHS of A→BE from R)
  R_2(<u>A</u>, B, E) (based on the FD A→BE)

Do we need to decompose further? Yes Why?

BCNF DECOMPOSITION EXAMPLE (cont'd)

We have: $R_1(\underline{A, C}, D)$ and $R_2(\underline{A}, B, E)$

F = {A→BE, C→D} (Note: F=F^+)

Candidate key: AC

- We need to decompose $R_1(\underline{A, C}, D)$ because of the FD C→D.
- Thus $R_1(A, C, D)$ is replaced with $R_3(A, C)$ and $R_4(C, D)$.
- Final decomposition: $R_2(A, B, E)$, $R_3(A, C)$, $R_4(C, D)$.

BCNF DECOMPOSITION EXAMPLE (cont'd)

Final decomposition: R_2(A, B, E) R_3(A, C) R_4(C, D)

FDs: $F_2 = \{A \rightarrow BE\}$ $F_3 = \varnothing$ $F_4 = \{C \rightarrow D\}$

Is the decomposition lossless?

Yes the algorithm always creates lossless decompositions.

In step S = (S – R) ∪ (R – Y) ∪ (X, Y) we replace R with relations (R – Y) and (X, Y) that have X as the common attribute and X→Y (i.e., X is the key of (X, Y)).

Is the decomposition dependency preserving?

Yes because $(F_2 \cup F_3 \cup F_4)^+ = F^+$

☞ **But remember:** sometimes dependencies <u>cannot</u> be preserved.

RELATIONAL DATABASE DESIGN: SUMMARY

Functional Dependencies

- FDs are constraints derived from the application domain.
- FDs can be used to refine a relation schema reduced from an E-R schema.

Normalization

- When an E-R schema is not well designed, the relation schemas generated from it may have undesirable properties (update anomalies).
- Using functional dependencies, normalization remove these update anomalies by decomposing a relation schema into normal forms.
- While BCNF is the "best" normal form, it may not be dependency preserving.
- There is always a dependency preserving 3NF decomposition and, in practice, 3NF is often "good enough" for most applications.

COMP 3311: SYLLABUS

- ✓ Introduction
- ✓ Entity-Relationship (E-R) Model and Database Design
- ✓ Relational Algebra
- ✓ Structured Query Language (SQL)
- ✓ Relational Database Design
- ➡ **Storage and File Structure**
- Indexing
- Query Processing
- Query Optimization
- Transactions
- Concurrency Control
- Recovery System
- NoSQL Databases

RELATIONAL DATABASE DESIGN: NORMALIZATION

EXERCISES 5, 6, 7

Upload your completed exercise worksheet to Canvas by **11 p.m. on March 5th.**