Database Management Systems

Normal Forms and Normalization

M. Emre Gürsoy

Assistant Professor
Department of Computer Engineering
www.memregursoy.com

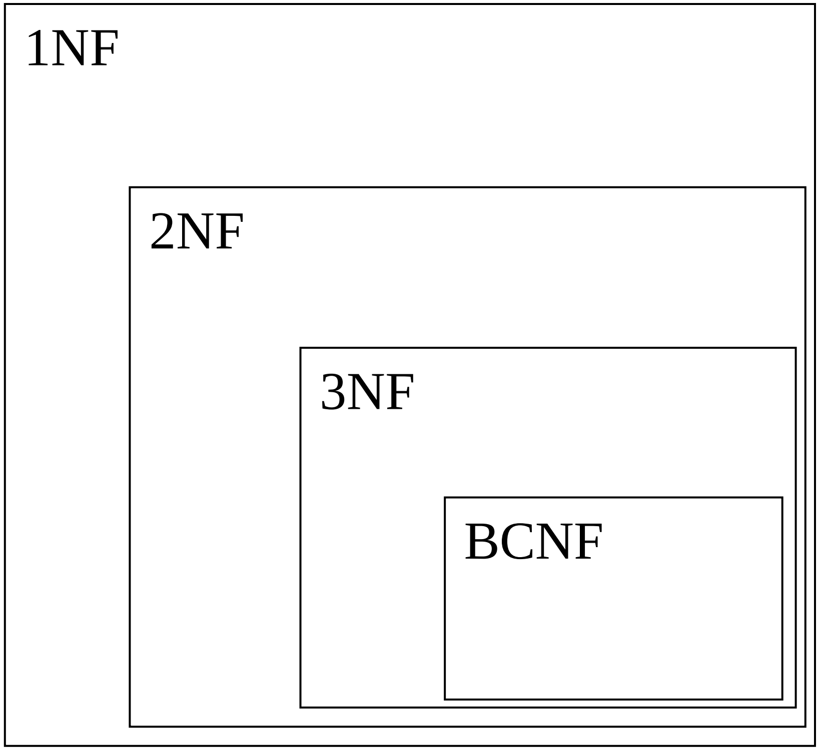
Introduction

- **Normalization** is a process that improves a database design by generating "better" relations.
- **Normal forms:** Standards for a "good" DB schema.
  - If a relation is in a certain normal form, it is known that certain kinds of problems are avoided.

- Normal forms we'll study:
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)

Normal Forms

- Requirements get stricter: 1NF -> 2NF -> 3NF -> BCNF



A relation that is in BCNF is also in 3NF

A relation that is in 3NF is also in 2NF

A relation that is in 2NF is also in 1NF

.. but not every relation that is in 1NF is also in 2NF!

First Normal Form

- We say that a relation is in 1NF if all values stored in the relation are single-valued (no set-valued attributes).
  - E.g.: In the past, there used to be the concept of "nested relations", which also aren't allowed in 1NF.

- A relation that is not in 1NF:

EmpNum	EmpPhone	EmpDegrees
123	233-9876	
333	233-1231	BA, BSc, PhD
679	233-1231	BSc, MSc

First Normal Form

- Decomposition into 1NF:
  - Move degrees to a new table, EmpNum is FK, each (EmpNum, EmpDegree) pair is a new tuple
  - An outer join between Employee and EmployeeDegree will enable you to recover the original information

EmployeeDegree

EmpNum	EmpDegree
333	BA
333	BSc
333	PhD
679	BSc
679	MSc

Employee

EmpNum	EmpPhone
123	233-9876
333	233-1231
679	233-1231

Second Normal Form

- For a relation to be in 2NF:
  - The relation must be in 1NF
  - Non-key attributes in the relation must be functionally dependent on the **whole primary key;** they are not allowed to depend on a **subset** of the primary key

- Example (not in 2NF):
  - R(<u>Title, PubId, AuId</u>, Price, AuAddress)
  - FDs:
    - Title, PubId, AuId -> Price
    - AuId -> AuAddress
  - Violation: AuAddress is a non-key attribute, yet it depends on AuId which is a **subset** of the key

Second Normal Form

- Another example (not in 2NF):
  - R(<u>Studio, Movie</u>, Budget, StudioCity)
  - FDs:
    - Studio, Movie -> Budget
    - Studio -> StudioCity
  - Violation: StudioCity is a non-key attribute, yet it depends on Studio which is a **subset** of the key

How can we normalize to achieve 2NF?

2NF Decomposition

<u>**Strategy for 2NF decomposition:**</u>

1. Find the non-key attribute that is dependent on only a part of the key (the violation).
2. Create a new table with that attribute and that part of the key.
3. If other attributes are dependent on the same part of the key, also place them in the new table.
4. Make the partial key copied from the original table to the new table the primary key of the new table.
5. Repeat the above until you achieve 2NF.

2NF Decomposition

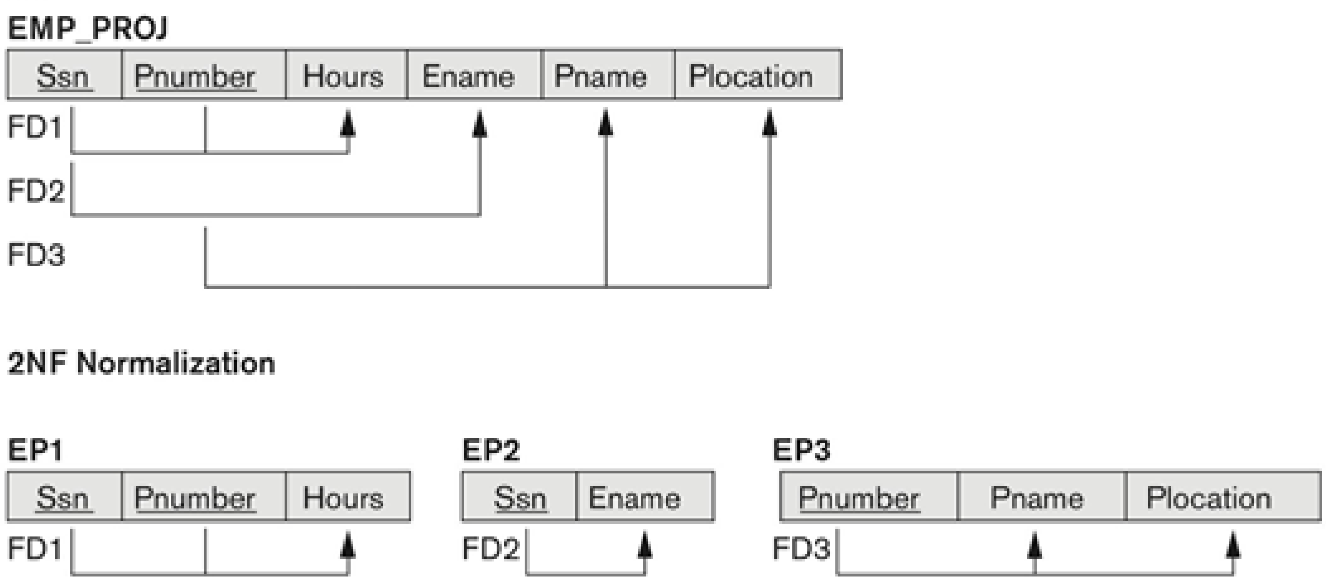
- Old: R(Title, PubId, AuId, Price, AuAddress)
  - Violation of 2NF: AuId -> AuAddress
- New, in 2NF:
  - R1(Title, PubId, AuId, Price)
  - R2(AuId, AuAddress)

****

- Old: R(Studio, Movie, Budget, StudioCity)
  - Violation of 2NF: Studio -> StudioCity
- New, in 2NF:
  - R1(Studio, Movie, Budget)
  - R2(Studio, StudioCity)

2NF Decomposition

- EMP_PROJ doesn't satisfy 2NF
- Which FDs violate 2NF?



Third Normal Form

- For a relation to be in 3NF:
  - The relation must be in 2NF
  - Non-key attributes in the relation must be functionally dependent on **only a candidate key**; they are not allowed to depend on non-key attributes

- Implications of 3NF:
  - No inter-dependencies among non-key attributes
  - No transitive dependency on primary key
    - K -> A -> B

Third Normal Form

- Example (not in 3NF):
  - R(<u>Studio</u>, StudioCity, CityTemp)
  - FDs:
    - Studio -> StudioCity
    - Studio -> CityTemp
    - StudioCity -> CityTemp
  - Is the relation in 2NF? Yes.
  - Violation of 3NF: CityTemp depends on StudioCity, which is not a candidate key

Third Normal Form

- Example (not in 3NF):
  - R(<u>Title, PubId</u>, PageCount, Price)
  - FDs:
    - Title, PubId -> PageCount
    - PageCount -> Price
    - [implicit] Title, PubId -> Price
  - Is the relation in 2NF? Yes.
  - Violation of 3NF: Price depends on PageCount, which is not a candidate key.

3NF Decomposition

Strategy for 3NF decomposition:

1. Find the violation, i.e., the non-key to non-key dependency X -> Y.
2. Create a new table; put X and Y in the new table. X is the primary key of the new table.
3. Remove Y but keep X in the original table. X in the original table and X in the new table have a foreign key relationship.
4. Repeat the above until you achieve 3NF.

3NF Decomposition

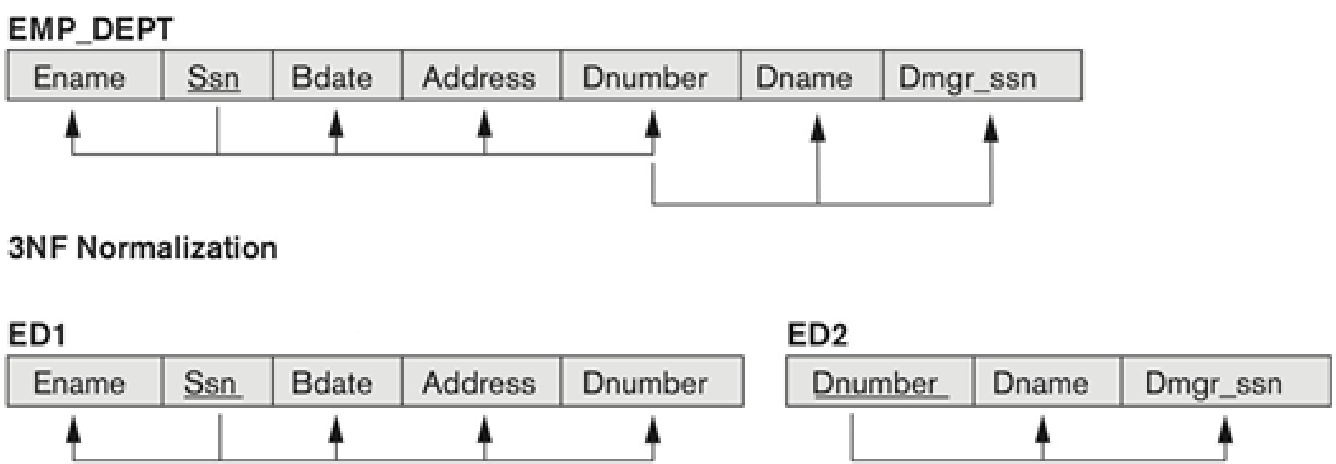
- Old: R(Studio, StudioCity, CityTemp)
  - Violation of 3NF: StudioCity -> CityTemp
- New, in 3NF:
  - R1(Studio, StudioCity)
  - R2(StudioCity, CityTemp)

****

- Old: R(Title, PubId, PageCount, Price)
  - Violation of 3NF: PageCount -> Price
- New, in 3NF:
  - R1(Title, PubId, PageCount)
  - R2(PageCount, Price)

3NF Decomposition

- Does EMP_DEPT satisfy 1NF?
- Does EMP_DEPT satisfy 2NF?
- Does EMP_DEPT satisfy 3NF? No.
  - Which FD or FDs violate 3NF?



Normal Forms - Informally

- 1NF
  - Attributes are single-valued and depend on **the key**
- 2NF
  - Non-key attributes depend on **the whole key**
- 3NF
  - Non-key attributes depend on **nothing but the key**

Boyce-Codd Normal Form

- 2NF and 3NF place constraints on what non-key attributes can depend on.
  - But how about what key attributes can depend on?
  - How about multiple candidate keys?

TEACH

Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar

FDs:
- Student, Course -> Instructor
- Instructor -> Course

{Student, Course} is cand. key

Satisfies 2NF? Yes.
Satisfies 3NF? Yes.
Satisfies BCNF? No!

Boyce-Codd Normal Form

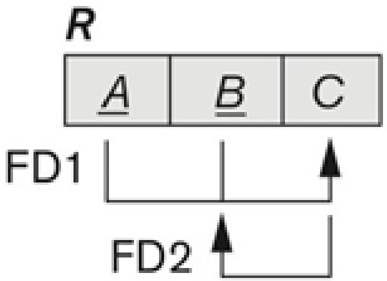
- For a relation to be in BCNF:
  - The relation must be in 3NF
  - Whenever an FD X->A holds, then X must be a key of the relation

- "Every determinant must be a candidate key."
  - A non-key attribute shouldn't be determining any other attribute (including subsets of the key!)

BCNF Violations

- Example (not in BCNF):
  - R(MovieTitle, PersonName, MovieID, Role, Payment)
  - MovieID -> MovieTitle violates BCNF

A typical kind of BCNF violation:



Properties of Decompositions

- Two properties of decompositions
  - **Lossless join (non-additive) decomposition**: does not cause information loss or spurious tuples
  - **Dependency-preserving decomposition**: does not cause any FDs to be lost
- Ideally we'd like to have both, but when achieving BCNF, typically we can't have both
  - Pick one, abandon the other
- Lossless join is a "must", dependency-preservation is "nice to have"
- Our BCNF decomposition algorithm will satisfy lossless join property, but it will not necessarily preserve dependencies

Lossy Decomposition

- Consider the following decomposition of **R** into **R1** and **R2**

R

Model Name	Price	Category
a11	100	Canon
s20	200	Nikon
a70	150	Canon

R1

Model Name	Category
a11	Canon
s20	Nikon
a70	Canon

R2

Price	Category
100	Canon
200	Nikon
150	Canon

Lossy Decomposition

- What is the result when you join **R1** and **R2**?

Model Name	**Price**	**Category**
a11	100	Canon
a11	150	Canon
s20	200	Nikon
a70	100	Canon
a70	150	Canon

- This is a lossy decomposition
  - Violates **lossless join (non-additivity)** property
- Our BCNF decomposition should not behave like this

BCNF Decomposition

Algorithm: Relational Decomposition into BCNF with lossless (non-additive) join property

Input: A universal relation *R* and a set of functional dependencies *F* on the attributes of *R*.

1. Set D := {R};
2. While there is a relation *Q* in *D* that is not in BCNF

   do {

   choose a relation *Q* in *D* that is not in BCNF;

   find a functional dependency $X \rightarrow Y$ in *Q* that violates BCNF;

   replace *Q* in *D* by two relation schemas (*Q* - *Y*) and ($X \cup Y$);

   };

BCNF Decomposition

- Old: R(Student, Course, Instructor)
  - Violation of BCNF: Instructor -> Course
- New, in BCNF:
  - R1(Student, Instructor)
  - R2(Instructor, Course)

****

- Old: R(MovieTitle, PersonName, MovieID, Role, Payment)
  - Violation of BCNF: MovieID -> MovieTitle
- New, in BCNF:
  - R1(PersonName, MovieID, Role, Payment)
  - R2(MovieID, MovieTitle)

Testing Losslessness

Algorithm: Testing for Lossless Join Property

Input: A universal relation *R*, a decomposition $D = \{R_1, R_2, ..., R_m\}$ of *R*, and a set *F* of functional dependencies.

1. Create an initial matrix *S* with one row *i* for each relation R_i in *D*, and one column *j* for each attribute A_j in *R.*
2. Set $S(i,j) := b_{ij}$ for all matrix entries.

   (* each b_{ij} is a distinct symbol associated with indices (i,j) *)
3. For each row *i* representing relation schema R_i

   {for each column j representing attribute A_j

   {if (relation R_i includes attribute A_j) then set

   $S(i,j) := a_j$;};};

(* each a_j is a distinct symbol associated with index (*j*) *)

Testing Losslessness

<u>Algorithm : Testing for Lossless Join Property</u> (cont.)

4. Repeat the following loop until a *complete loop execution* results in no changes to *S*

{for each functional dependency $X \rightarrow Y$ in *F*

{for all rows in *S which have the same symbols* in the columns corresponding to attributes in *X*

{make the symbols in each column that correspond to an attribute in *Y* be the same in all these rows using the following rules:

- <u>If any of the rows has an "*a*" symbol for the column,</u> set the other rows to that *same* "*a*" symbol in the column.
- <u>If no "*a*" symbol exists for the attribute in any of the rows,</u> choose one of the "*b*" symbols that appear in one of the rows for the attribute and set the other rows to that same "*b*" symbol in the column ;};};};

5. If a row is made up entirely of "*a*" symbols, then the decomposition has the lossless join property; otherwise it does not.

Example #1

$R = \{\text{Ssn, Ename, Pnumber, Pname, Plocation, Hours}\}$ $D = \{R_1, R_2, R_3\}$

$R_1 = \text{EMP} = \{\text{Ssn, Ename}\}$

$R_2 = \text{PROJ} = \{\text{Pnumber, Pname, Plocation}\}$

$R_3 = \text{WORKS\_ON} = \{\text{Ssn, Pnumber, Hours}\}$

$F = \{\text{Ssn} \rightarrow \text{Ename}; \text{Pnumber} \rightarrow \{\text{Pname, Plocation}\}; \{\text{Ssn, Pnumber}\} \rightarrow \text{Hours}\}$

	Ssn	Ename	Pnumber	Pname	Plocation	Hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

This is the matrix at the end of Step 3.

	Ssn	Ename	Pnumber	Pname	Plocation	Hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	$\cancel{b_{32}}\ a_2$	a_3	$\cancel{b_{34}}\ a_4$	$\cancel{b_{35}}\ a_5$	a_6

(Matrix S after applying the first two functional dependencies; last row is all "a" symbols so we stop)

Example #2

- R(A,B,C,D,E)
- ***F*** = {A->C, B->C, C->D, DE->C, CE->A}
- Decomposed into {R1, R2, R3, R4, R5}
  - R1(A,D)
  - R2(A,B)
  - R3(B,E)
  - R4(C,D,E)
  - R5(A,E)

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{23}	b_{24}	b_{25}
R_3	b_{31}	a_2	b_{33}	b_{34}	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{53}	b_{54}	a_5

- Q: Does this decomposition satisfy lossless join property?
  - Equivalent: Is this decomposition lossless?

Example #2

Apply A->C:

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	$\cancel{b_{23}}\ b_{13}$	b_{24}	b_{25}
R_3	b_{31}	a_2	b_{33}	b_{34}	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	$\cancel{b_{53}}\ b_{13}$	b_{54}	a_5

Apply B->C:

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{13}	b_{24}	b_{25}
R_3	b_{31}	a_2	$\cancel{b_{33}}\ b_{13}$	b_{34}	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{13}	b_{54}	a_5

Apply C->D:

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{13}	$\cancel{b_{24}}\ a_4$	b_{25}
R_3	b_{31}	a_2	b_{13}	$\cancel{b_{34}}\ a_4$	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{13}	$\cancel{b_{54}}\ a_4$	a_5

Example #2

Apply DE->C:

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{13}	a_4	b_{25}
R_3	b_{31}	a_2	$\cancel{b_{13}}$ a_3	a_4	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	$\cancel{b_{13}}$ a_3	a_4	a_5

Apply CE->A:

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{13}	a_4	b_{25}
R_3	$\cancel{b_{31}}$ a_1	a_2	a_3	a_4	a_5
R_4	$\cancel{b_{41}}$ a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

The third row consists entirely of *a* symbols.
Hence, the decomposition SATISFIES the lossless join property.

Example #3

- R(A,B,C)
- ***F*** = {AB->C, C->B}
- Decomposed into R1(A,B) and R2(B,C)

Doesn't satisfy lossless join property!

	A	B	C
R_1	a_1	a_2	b_{13}
R_2	b_{21}	a_2	a_3

After step 3

AB->C

	A	B	C
R_1	a_1	a_2	b_{13}
R_2	b_{21}	a_2	a_3

C->B

	A	B	C
R_1	a_1	a_2	b_{13}
R_2	b_{21}	a_2	a_3

- Instead, decomposed into R1(A,C) and R2(B,C)

	A	B	C
R_1	a_1	b_{12}	a_3
R_2	b_{21}	a_2	a_3

AB->C

	A	B	C
R_1	a_1	b_{12}	a_3
R_2	b_{21}	a_2	a_3

C->B

	A	B	C
R_1	a_1	$\cancel{b_{12}}\ a_2$	a_3
R_2	b_{21}	a_2	a_3

Satisfies lossless join property!

Relational Synthesis

- We have been studying **decomposition**
  - Given an actual relation, break it down into several relations to satisfy the next NF
  - 1NF -> find 2NF violations -> 2NF -> find 3NF violations -> 3NF -> find BCNF violations -> BCNF
- **Synthesis** is the other way around
  - Given all attributes in one potentially hypothetical relation and a set of FDs among the attributes, design an appropriate DB schema
  - Bottom-up approach
- We will learn an algorithm for 3NF synthesis
  - How would you synthesize BCNF?

Relational Synthesis

Algorithm : Relational Synthesis into 3NF with Dependency Preservation and Lossless (Non-Additive) Join Property

Input: A universal relation *R* and a set of functional dependencies *F* on the attributes of *R*.

1. Find a minimal cover *G* for *F.*
2. For each left-hand-side *X* of a functional dependency that appears in *G,* create a relation in *D* with attributes:

   $\{X \cup \{A_1\} \cup \{A_2\} \dots \cup \{A_k\}\}$, where

   $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_k$ are the only dependencies in *G* with *X* as left-hand-side (*X* is the *key* of this relation).
3. If none of the relations in *D* contains a key of *R*, then create one more relation in *D* that contains attributes that form a key of *R*.

Example

- R = {ssn, ename, bdate, address, dno, dname, dmgrssn}
- FDs:
  - ssn -> ename, bdate, address, dno
  - dno -> dname, dmgrssn

- Output of synthesis algorithm:
  - R1(<u>ssn</u>, ename, bdate, address, dno)
  - R2(<u>dno</u>, dname, dmgrssn)

Exercise

Consider the following relation and set of FDs:

- R(<u>Order, Product</u>, Quantity, UnitPrice, Customer, Address)
  - Order -> Customer
  - Customer -> Address
  - Product -> UnitPrice
  - Order -> Address

- **Which NF is this relation in?**
  - **(Can you see violations of 1NF, 2NF, 3NF, BCNF?)**
- **Decompose the relation to satisfy 2NF. Then decompose to satisfy 3NF, then BCNF.**
- **Is this decomposition lossless? Depndcy-preserving?**