COSC 304
Introduction to Database Systems

Normalization

Dr. Ramon Lawrence
University of British Columbia Okanagan
ramon.lawrence@ubc.ca

Normalization

Normalization is a technique for producing relations with desirable properties.

Normalization decomposes relations into smaller relations that contain less redundancy. This decomposition requires that no information is lost and reconstruction of the original relations from the smaller relations must be possible.

Normalization is a bottom-up design technique for producing relations. It pre-dates ER modeling and was developed by Codd in 1972 and extended by others over the years.

- Normalization can be used after ER modeling or independently.
- Normalization may be especially useful for databases that have already been designed without using formal techniques.

Normalization Motivation

The purpose of normalization is to develop good relational schemas that minimize redundancies and update anomalies.

Redundancy occurs when the same data value is stored more than once in a relation.

- Redundancy wastes space and reduces performance.

Update anomalies are problems that arise when trying to insert, delete, or update tuples and are often caused by redundancy.

The goal of normalization is to produce a set of relational schemas $R_1, R_2, \ldots, R_m$ from a set of attributes $A_1, A_2, \ldots, A_n$.

- Imagine that the attributes are originally all in one big relation R= $\{A_1, A_2, .., A_n\}$ which we will call the ***Universal Relation***.
- Normalization divides this relation into $R_1, R_2, \ldots, R_m$.

The Power of Normalization Example Relations

Emp Relation

eno	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	Budget	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

Dept Relation

dno	dname	mgreno
D1	Management	E8
D2	Consulting	E7
D3	Accounting	E5
D4	Development	null

The Power of Normalization Universal Relation

A Universal Relation with all attributes:

eno	pno	resp	hours	ename	bdate	title	salary	supereno	dno	dname	mgreno	pname	budget
E1	P1	Manager	12	J. Doe	01-05-75	EE	30000	E2				Instruments	150000
E2	P1	Analyst	24	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5	Instruments	150000
E2	P2	Analyst	6	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5	DB Develop	135000
E3	P3	Consultant	10	A. Lee	07-05-66	ME	40000	E6	D2	Consulting	E7	Budget	250000
E3	P4	Engineer	48	A. Lee	07-05-66	ME	40000	E6	D2	Consulting	E7	Maintenance	310000
E4	P2	Programmer	18	J. Miller	09-01-50	PR	20000	E6	D3	Accounting	E5	DB Develop	135000
E5	P2	Manager	24	B. Casey	12-25-71	SA	50000	E8	D3	Accounting	E5	DB Develop	135000
E6	P4	Manager	48	L. Chu	11-30-65	EE	30000	E7	D2	Consulting	E7	Maintenance	310000
E7	P3	Engineer	36	J. Jones	10-11-72	SA	50000		D1	Management	E8	Budget	250000

Universal(eno, pno, resp, hours, ename, bdate, title, salary, supereno, dno, dname, mgreno, pname, budget)

What are some of the problems with the Universal Relation?

Normalization will allow us to get back to the starting relations.

Update Anomalies

There are three major types of update anomalies:

- ***Insertion Anomalies*** - Insertion of a tuple into the relation either requires insertion of redundant information or cannot be performed without setting key values to `NULL`.
- ***Deletion Anomalies*** - Deletion of a tuple may lose information that is still required to be stored.
- ***Modification Anomalies*** - Changing an attribute of a tuple may require changing multiple attribute values in other tuples.

Example Relation Instances

Emp Relation

eno	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

Dept Relation

dno	dname	mgreno
D1	Management	E8
D2	Consulting	E7
D3	Accounting	E5
D4	Development	null

EmpDept Relation

eno	ename	bdate	title	salary	supereno	dno	dname	mgreno
E1	J. Doe	01-05-75	EE	30000	E2	null	null	null
E2	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5
E3	A. Lee	07-05-66	ME	40000	E7	D2	Consulting	E7
E4	J. Miller	09-01-50	PR	20000	E6	D3	Accounting	E5
E5	B. Casey	12-25-71	SA	50000	E8	D3	Accounting	E5
E6	L. Chu	11-30-65	EE	30000	E7	D2	Consulting	E7
E7	R. Davis	09-08-77	ME	40000	E8	D1	Management	E8
E8	J. Jones	10-11-72	SA	50000	null	D1	Management	E8

Insertion Anomaly Example

eno	ename	bdate	title	salary	supereno	dno	dname	mgreno
E1	J. Doe	01-05-75	EE	30000	E2	null	null	null
E2	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5
E3	A. Lee	07-05-66	ME	40000	E7	D2	Consulting	E7
E4	J. Miller	09-01-50	PR	20000	E6	D3	Accounting	E5
E5	B. Casey	12-25-71	SA	50000	E8	D3	Accounting	E5
E6	L. Chu	11-30-65	EE	30000	E7	D2	Consulting	E7
E7	R. Davis	09-08-77	ME	40000	E8	D1	Management	E8
E8	J. Jones	10-11-72	SA	50000	null	D1	Management	E8

Consider these two types of insertion anomalies:

- 1) Insert a new employee E9 working in department D2.
  - ⇨ You have to redundantly insert the department name and manager when adding this record.
- 2) Insert a department D4 that has no current employees.
  - ⇨ This insertion is not possible without creating a dummy employee id and record because `eno` is the primary key of the relation.

Deletion Anomaly Example

eno	ename	bdate	title	salary	supereno	dno	dname	mgreno
E1	J. Doe	01-05-75	EE	30000	E2	null	null	null
E2	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5
E3	A. Lee	07-05-66	ME	40000	E7	D2	Consulting	E7
E4	J. Miller	09-01-50	PR	20000	E6	D3	Accounting	E5
E5	B. Casey	12-25-71	SA	50000	E8	D3	Accounting	E5
E6	L. Chu	11-30-65	EE	30000	E7	D2	Consulting	E7
E7	R. Davis	09-08-77	ME	40000	E8	D1	Management	E8
E8	J. Jones	10-11-72	SA	50000	null	D1	Management	E8

Consider this deletion anomaly:

- Delete employees E3 and E6 from the database.
- Deleting those two employees removes them from the database, and we now have lost information about department D2!

Modification Anomaly Example

eno	ename	bdate	title	salary	supereno	dno	dname	mgreno
E1	J. Doe	01-05-75	EE	30000	E2	null	null	null
E2	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5
E3	A. Lee	07-05-66	ME	40000	E7	D2	Consulting	E7
E4	J. Miller	09-01-50	PR	20000	E6	D3	Accounting	E5
E5	B. Casey	12-25-71	SA	50000	E8	D3	Accounting	E5
E6	L. Chu	11-30-65	EE	30000	E7	D2	Consulting	E7
E7	R. Davis	09-08-77	ME	40000	E8	D1	Management	E8
E8	J. Jones	10-11-72	SA	50000	null	D1	Management	E8

Consider these modification anomalies:

- 1) Change the name of department D3 to Embezzling.
  - ⇨ You must update the department name in 3 different records.
- 2) Change the manager of D1 to E4.
  - ⇨ You must update the `mgreno` field in 2 different records.

Desirable Relational Schema Properties

Relational schemas that are well-designed have several important properties:

- 1) The most basic property is that relations consists of attributes that are logically related.
  - ⇨ The attributes in a relation should belong to only one entity or relationship.
- 2) ***Lossless-join property*** ensures that the information decomposed across many relations can be reconstructed using natural joins.
- 3) ***Dependency preservation property*** ensures that constraints on the original relation can be maintained by enforcing constraints on the normalized relations.

Normalization Question

Question: How many of the following statements are ***true***?

1) Normalization is a bottom up process.

2) Anomalies with updates often are indicators of a bad design.

3) The lossless-join property means that it is possible to reconstruct the original relation after normalization using joins.

4) The dependency preservation property means that constraints should be preserved before and after normalization.

A) 0 **B)** 1 **C)** 2 **D)** 3 **E)** 4

Functional Dependencies

Functional dependencies represent constraints on the values of attributes in a relation and are used in normalization.

A ***functional dependency*** (abbreviated ***FD***) is a statement about the relationship between attributes in a relation. We say a set of attributes *X* functionally determines an attribute *Y* if given the values of *X* we always know the only possible value of *Y*.

- Notation: $X \rightarrow Y$
- *X* functionally determines *Y*
- *Y* is functionally dependent on *X*

Example:

- eno → ename
- eno, pno → hours

Notation for Functional Dependencies

A functional dependency has a left-side called the ***determinant*** which is a set of attributes, and one attribute on the right-side.

$$\underbrace{\text{eno, pno}}_{\text{determinant}} \rightarrow \underbrace{\text{hours}}_{\text{determined attribute}}$$

Strictly speaking, there is always only one attribute on the RHS, but we can combine several functional dependencies into one:

eno, pno → hours
eno, pno → resp
eno, pno → hours, resp

Remember that this is really short-hand for two functional dependencies.

The Semantics of Functional Dependencies

Functional dependencies are a property of the ***domain*** being modeled **NOT** of the data instances currently in the database.

- This means that similar to keys you cannot tell if one attribute is functionally dependent on another by looking at the data.

Example: Emp Relation

eno	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

- List the functional dependencies of the attributes in this relation.

The Semantics of Functional Dependencies (2)

Functional dependencies are directional.

eno → ename does not mean that ename → eno

Example: Emp Relation

eno	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

- Given an employee name there may be multiple values for `eno` if we have employees in the database with the same name.
- Thus knowing `ename`, does not uniquely tell us the value of `eno`.

Trivial Functional Dependencies

A functional dependency is ***trivial*** if the attributes on its left-hand side are a superset of the attributes on its right-hand side.

Examples:

eno → eno
eno, ename → eno
eno, pno, hours → eno, hours

Trivial functional dependencies are not interesting because they do not tell us anything.

- Trivial FDs basically say "If you know the values of these attributes, then you uniquely know the values of any subset of those attributes.

We are only interested in ***nontrivial*** FDs.

Identifying Functional Dependencies

Identify all non-trivial functional dependencies in the Universal Relation:

eno	pno	resp	hours	ename	bdate	title	salary	supereno	dno	dname	mgreno	pname	budget
E1	P1	Manager	12	J. Doe	01-05-75	EE	30000	E2				Instruments	150000
E2	P1	Analyst	24	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5	Instruments	150000
E2	P2	Analyst	6	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5	DB Develop	135000
E3	P3	Consultant	10	A. Lee	07-05-66	ME	40000	E6	D2	Consulting	E7	Budget	250000
E3	P4	Engineer	48	A. Lee	07-05-66	ME	40000	E6	D2	Consulting	E7	Maintenance	310000
E4	P2	Programmer	18	J. Miller	09-01-50	PR	20000	E6	D3	Accounting	E5	DB Develop	135000
E5	P2	Manager	24	B. Casey	12-25-71	SA	50000	E8	D3	Accounting	E5	DB Develop	135000
E6	P4	Manager	48	L. Chu	11-30-65	EE	30000	E7	D2	Consulting	E7	Maintenance	310000
E7	P3	Engineer	36	J. Jones	10-11-72	SA	50000		D1	Management	E8	Budget	250000

Why the Name "Functional" Dependencies?

Functional dependencies get their name because you could imagine the existence of some function that takes in the parameters of the left-hand side and computes the value on the right-hand side of the dependency.

Example: eno, pno $\rightarrow$ hours

f(eno, pno) $\rightarrow$ hours

```
int f (String eno, String pno)
{       // Do some lookup...
        return hours;
}
```

Remember that no such function exists, but it may be useful to think of FDs this way.

Functional Dependencies and Keys

Functional dependencies can be used to determine the candidate and primary keys of a relation.

- For example, if an attribute functionally determines all other attributes in the relation, that attribute can be a key:

eno $\rightarrow$ eno, ename, bdate, title, supereno, dno

- `eno` is a candidate key for the `Employee` relation.

Alternate definition of keys:

- A set of attributes *K* is a ***superkey*** for a relation *R* if the set of attributes *K* functionally determines all attributes in *R*.
- A set of attributes *K* is a ***candidate key*** for a relation *R* if *K* is a *minimal* superkey of *R*.

Functional Dependencies and Keys (2)

EmpDept Relation

eno	ename	bdate	title	salary	supereno	dno	dname	mgreno
E1	J. Doe	01-05-75	EE	30000	E2	null	null	null
E2	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5
E3	A. Lee	07-05-66	ME	40000	E7	D2	Consulting	E7
E4	J. Miller	09-01-50	PR	20000	E6	D3	Accounting	E5
E5	B. Casey	12-25-71	SA	50000	E8	D3	Accounting	E5
E6	L. Chu	11-30-65	EE	30000	E7	D2	Consulting	E7
E7	R. Davis	09-08-77	ME	40000	E8	D1	Management	E8
E8	J. Jones	10-11-72	SA	50000	null	D1	Management	E8

eno $\rightarrow$ eno, ename, bdate, title, salary, supereno, dno

dno $\rightarrow$ dname, mgreno

Question: List the candidate key(s) of this relation.

Functional Dependency Rules

Functional dependencies follow rules (Armstrong's Axioms 1974) with the most important being ***transitivity***.

Transitive Rule: If $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$

Functional Dependency Question

Question: How many of the following statements are ***true***?

1) Functional dependencies are not directional.

2) A trivial functional dependency has its right side as a subset of (or the same as) its left side.

3) A key is a set of attributes that functionally determines all attributes in a relation.

4) Functional dependencies can be determined by examining the data in a relation.

A) 0 **B)** 1 **C)** 2 **D)** 3 **E)** 4

Computing the Closure of FDs

The transitivity rule of FDs can be used for three purposes:

- 1) To determine if a given FD $X \rightarrow Y$ follows from a set of FDs F.
- 2) To determine if a set of attributes *X* is a superkey of *R*.
- 3) To determine the set of all FDs (called the ***closure*** F^+) that can be inferred from a set of initial functional dependencies *F*.

The basic idea is that given any set of attributes *X,* we can compute the set of all attributes X^+ that can be functionally determined using *F*. This is called the ***closure of X under F***.

- For purpose #1, we know that $X \rightarrow Y$ holds if Y is in X^+.
- For purpose #2, *X* is a superkey of *R* if X^+ is all attributes of R.
- For purpose #3, we can compute X^+ for all possible subsets X of R to derive all FDs (the ***closure*** F^+) .

Computing the Attribute Closure

The algorithm is as follows:

- Given a set of attributes X.
- Let $X^+ = X$
- Repeat
  - Find a FD in F whose left side is a subset of X^+.
  - Add the right side of F to X^+.
- Until (X^+ does not change)

After the algorithm completes you have a set of attributes X^+ that can be functionally determined from X. This allows you to produce FDs of the form:

- $X \rightarrow A$ where A is in X^+

Computing Attribute Closure Example

R (A,B,C,D,E,G)

$F = \{A \rightarrow B,C \;\; , \;\; C \rightarrow D \;\; , \;\; D \rightarrow G\}$

Compute $\{A\}^+$:

- $\{A\}^+ = \{A\}$ (initial step)
- $\{A\}^+ = \{A,B,C\}$ (by using FD $A \rightarrow B,C$)
- $\{A\}^+ = \{A,B,C,D\}$ (by using FD $C \rightarrow D$)
- $\{A\}^+ = \{A,B,C,D,G\}$ (by using FD $D \rightarrow G$)

Similarly we can compute $\{C\}^+$ and $\{E,G\}^+$:

- $\{C\}^+ = \{C,D,G\}$
- $\{E,G\}^+ = \{E,G\}$

Using Attribute Closure to Detect a FD

The attribute closure algorithm can be used to determine if a particular FD $X \rightarrow Y$ holds based on the set of given FDs F.

- Compute X^+ and see if Y is in X^+.

Example:

- $R = (A,B,C,D,E,G)$
- $F = \{A \rightarrow B,C \;\; , \;\; C \rightarrow D \;\; , \;\; D \rightarrow G\}$
- Does the FD $C,D \rightarrow G$ hold?
  - Compute $\{C,D\}^+ = \{C,D,G\}$. Yes, the FD $C,D \rightarrow G$ holds.
- Does the FD $B,C \rightarrow E$ hold?
  - Compute $\{B,C\}^+ = \{B,C,D,G\}$. No, the FD $B,C \rightarrow E$ does not hold.

Function Dependency Closure Question

Question: Given the following, does `{A,B → D}` hold?

`R = (A,B,C,D)`

`F = { A,B → C , C → D , D → A }`

A) yes

B) no

FD Closure Question 2

Question: Given the following, does `{B,D → A,C}` hold?

`R = (A,B,C,D)`

`F = {A,B → C , B,C → D , C,D → A ,`

`A,D → B}`

A) yes

B) no

Function Dependency Key Question

Question: Given the following FDs, how many keys of R?

`R = (A,B,C,D)`

`F = { A,B → C , C → D , D → A }`

A) 0

B) 1

C) 2

D) 3

E) 4

Function Dependency Key Question 2

Question: Given the following FDs, how many keys of R?

$R = (A,B,C,D)$

$F = \{A,B \rightarrow C \;,\; B,C \rightarrow D \;,\; C,D \rightarrow A \;,\; A,D \rightarrow B\}$

A) 0
B) 1
C) 2
D) 3
E) 4

Minimal Set of FDs

A set of FDs *F* is ***minimal*** if it satisfies these conditions:

- Every FD in *F* has a single attribute on its right-hand side.
- We cannot replace any FD $X \rightarrow A$ in *F* with $Y \rightarrow A$ where $Y \subset X$ and still have a set of dependencies that is equivalent to *F*.
- We cannot remove any FD and still have a set that is equivalent to *F*.

A minimal set of FDs is like a canonical form of FDs.

Note that there may be many *minimal covers* for a given set of functional dependencies *F*.

Full Functional Dependencies

A functional dependency $A \rightarrow B$ is a ***full functional dependency*** if removal of any attribute from *A* results in the dependency not existing any more.

- We say that *B* is *fully functionally dependent* on *A*.
- If remove an attribute from *A* and the functional dependency still exists, we say that *B* is partially dependent on *A*.

Example:

eno → ename	(full FD)
eno, ename → salary, title	(partial FD - can remove ename)
eno, pno → hours, resp	(full FD)

Full Functional Dependency Question

Question: **True or False:** You can determine if certain, valid functional dependencies are full functional dependencies without having any domain knowledge.

A) true

B) false

Normal Forms

A relation is in a particular ***normal form*** if it satisfies certain normalization properties.

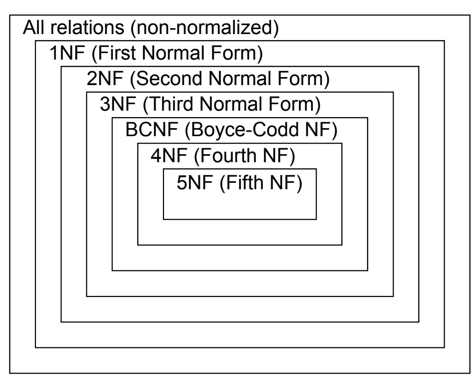
There are several normal forms defined:

- 1NF - First Normal Form
- 2NF - Second Normal Form
- 3NF - Third Normal Form
- BCNF - Boyce-Codd Normal Form
- 4NF - Fourth Normal Form
- 5NF - Fifth Normal Form

Each of these normal forms are stricter than the next.

- For example, 3NF is better than 2NF because it removes more redundancy/anomalies from the schema than 2NF.

Normal Forms



First Normal Form (1NF)

A relation is in ***first normal form*** (***1NF***) if all its attribute values are atomic.

That is, a 1NF relation cannot have an attribute value that is:

- a set of values (multi-valued attribute)
- a set of tuples (nested relation)

1NF is a standard assumption in relational DBMSs.

- However, object-oriented DBMSs and nested relational DBMSs relax this constraint.

A relation that is not in 1NF is an ***unnormalized*** relation.

A non-1NF Relation

eno	ename	pno	resp	hours
E1	J. Doe	P1	Manager	12
E2	M. Smith	P1	Analyst	24
		P2	Analyst	6
E3	A. Lee	P3	Consultant	10
		P4	Engineer	48
E4	J. Miller	P2	Programmer	18
E5	B. Casey	P2	Manager	24
E6	L. Chu	P4	Manager	48
E7	J. Jones	P3	Engineer	36

Two equivalent representations

eno	ename	pno	resp	hours
E1	J. Doe	P1	Manager	12
E2	M. Smith	{P1,P2}	{Analyst,Analyst}	{24,6}
E3	A. Lee	{P3,P4}	{Consultant,Engineer}	{10,48}
E4	J. Miller	P2	Programmer	18
E5	B. Casey	P2	Manager	24
E6	L. Chu	P4	Manager	48
E7	J. Jones	P3	Engineer	36

Two ways to convert a non-1NF relation to a 1NF relation:

- 1) ***Splitting Method*** - Divide the existing relation into two relations: non-repeating attributes and repeating attributes.
  - ⇨ Make a relation consisting of the primary key of the original relation and the repeating attributes. Determine a primary key for this new relation.
  - ⇨ Remove the repeating attributes from the original relation.
- 2) ***Flattening Method*** - Create new tuples for the repeating data combined with the data that does not repeat.
  - ⇨ Introduces redundancy that will be later removed by normalization.
  - ⇨ Determine primary key for this flattened relation.

Converting a non-1NF Relation to 1NF Using Splitting

eno	ename	pno	resp	hours
E1	J. Doe	P1	Manager	12
E2	M. Smith	P1	Analyst	24
		P2	Analyst	6
E3	A. Lee	P3	Consultant	10
		P4	Engineer	48
E4	J. Miller	P2	Programmer	18
E5	B. Casey	P2	Manager	24
E6	L. Chu	P4	Manager	48
E7	J. Jones	P3	Engineer	36

Also need original primary key: eno

Repeating group: (pno, resp, hours)

eno	ename
E1	J. Doe
E2	M. Smith
E3	A. Lee
E4	J. Miller
E5	B. Casey
E6	L. Chu
E7	J. Jones

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36

Converting a non-1NF Relation to 1NF Using Flattening

eno	ename	pno	resp	hours
E1	J. Doe	P1	Manager	12
E2	M. Smith	P1	Analyst	24
		P2	Analyst	6
E3	A. Lee	P3	Consultant	10
		P4	Engineer	48
E4	J. Miller	P2	Programmer	18
E5	B. Casey	P2	Manager	24
E6	L. Chu	P4	Manager	48
E7	J. Jones	P3	Engineer	36

eno	ename	pno	resp	hours
E1	J. Doe	P1	Manager	12
E2	M. Smith	P1	Analyst	24
E2	M. Smith	P2	Analyst	6
E3	A. Lee	P3	Consultant	10
E3	A. Lee	P4	Engineer	48
E4	J. Miller	P2	Programmer	18
E5	B. Casey	P2	Manager	24
E6	L. Chu	P4	Manager	48
E7	J. Jones	P3	Engineer	36

Second Normal Form (2NF)

A relation is in ***second normal form*** (***2NF***) if it is in 1NF and every non-prime attribute is fully functionally dependent on a candidate key.

- A ***prime attribute*** is an attribute in any candidate key.

If there is a FD $X \rightarrow Y$ that violates 2NF:

- Compute X^+.
- Replace R by relations: $R_1 = X^+$ and $R_2 = (R - X^+) \cup X$

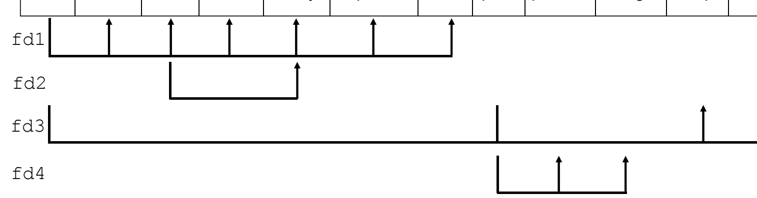
Note:

- By definition, any relation with a single key attribute is in 2NF.

Second Normal Form (2NF) Example

EmpProj relation:

eno	ename	title	bdate	salary	supereno	dno	pno	pname	budget	resp	hours



fd1 and fd4 are partial functional dependencies. Normalize to:

- Emp (eno, ename, title, bdate, salary, supereno, dno)
- WorksOn (eno, pno, resp, hours)
- Proj (pno, pname, budget)

Second Normal Form (2NF) Example (2)

Emp relation:

eno	ename	title	bdate	salary	supereno	dno

fd1

fd2

WorksOn relation:

eno	pno	resp	hours

fd3

Proj relation:

pno	pname	budget

fd4

Third Normal Form (3NF)

A relation is in ***third normal form*** (***3NF***) if it is in 2NF and there is no non-prime attribute that is transitively dependent on the primary key.

That is, for all functional dependencies $X \rightarrow Y$ of R, one of the following holds:

- Y is a prime attribute of R
- X is a superkey of R

Third Normal Form (3NF) Example

Emp relation:

eno	ename	title	bdate	salary	supereno	dno

fd1

fd2

fd2 results in a transitive dependency eno $\rightarrow$ salary. Remove it.

Emp

eno	ename	title	bdate	supereno	dno

fd1

Pay

title	salary

fd2

Normalization Question

Consider the universal relation R(A,B,C,D,E,F,G,H,I,J) and the set of functional dependencies:

- F= { $A,B \rightarrow C$; $A \rightarrow D,E$; $B \rightarrow F$; $F \rightarrow G,H$; $D \rightarrow I,J$ }

List the keys for *R*.

Decompose *R* into 2NF and then 3NF relations.

Boyce-Codd Normal Form (BCNF)

A relation is in ***Boyce-Codd normal form*** (***BCNF***) if and only if every determinant of a non-trivial FD is a superkey.

To test if a relation is in BCNF, we take the determinant of each non-trivial FD in the relation and determine if it is a superkey.

The difference between 3NF and BCNF is that 3NF allows a FD $X \rightarrow Y$ to remain in the relation if X is a superkey **or** Y is a prime attribute. BCNF only allows this FD if X is a superkey.

- Thus, BCNF is more restrictive than 3NF. However, in practice most relations in 3NF are also in BCNF.

Boyce-Codd Normal Form (BCNF) Example

Consider the `WorksOn` relation where we have the added constraint that given the hours worked, we know exactly the employee who performed the work. (i.e. each employee is FD from the hours that they work on projects). Then:

WorksOn

eno	pno	resp	hours

fd1

fd2

Relation is in 3NF but not BCNF.

eno	hours

fd2

pno	hours	resp

Note that we lose the FD eno,pno $\rightarrow$ resp, hours.

BCNF versus 3NF

We can decompose to BCNF but sometimes we do not want to if we lose a FD.

The decision to use 3NF or BCNF depends on the amount of redundancy we are willing to accept and the willingness to lose a functional dependency.

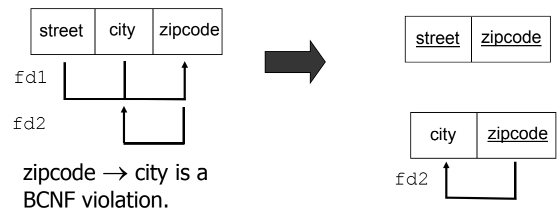
Note that we can always preserve the lossless-join property (recovery) with a BCNF decomposition, but we do not always get dependency preservation.

In contrast, we get both recovery and dependency preservation with a 3NF decomposition.

BCNF versus 3NF Example

An example of not having dependency preservation with BCNF:

- street,city → zipcode and zipcode → city
- Two keys: {street,city} and {street, zipcode}



BCNF versus 3NF Example (2)

Consider an example instance:

	street	zip
invalid	545 Tech Sq.	02138
MIT	545 Tech Sq.	02139

city	zip	
Cambridge	02138	Harvard
Cambridge	02139	MIT

Join tuples with equal zipcodes:

street	city	zip	
545 Tech Sq.	Cambridge	02138	invalid
545 Tech Sq.	Cambridge	02139	MIT

Note that the decomposition did not allow us to enforce the constraint that street,city → zipcode even though no FDs were violated in the decomposed relations.

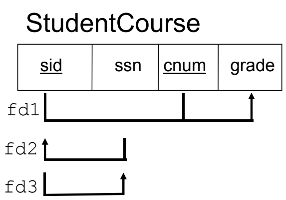
☆ Conversion to BCNF

Algorithm for converting to BCNF given relation *R* with FDs *F*:

- Look among the given FDs for a BCNF violation $X \rightarrow Y$.
- Compute X^+.
  - ⇨ Note that X^+ cannot be the set of all attributes or else *X* is a superkey.
- Replace R by relations with schemas:
  - ⇨ $R_1 = X^+$
  - ⇨ $R_2 = (R - X^+) \cup X$
- Project the FDs *F* onto the two new relations by:
  - ⇨ Computing the closure of *F* and then using only the FDs whose attributes are all in R_1 or R_2.

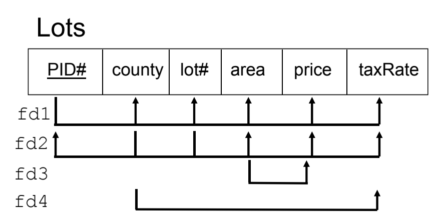
Normalization to BCNF Question

Given this schema normalize into BCNF directly.



Normalization Question

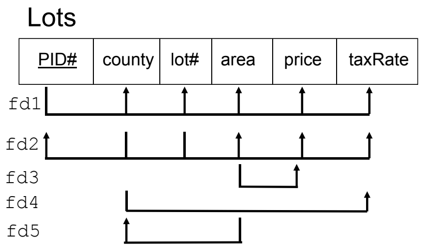
Given this database schema normalize into BCNF.



- Description:
  - ⇨ Lots (parcels of land) are identified within each county.
  - ⇨ Each lot can be uniquely identified either by the Property ID# or by the County name and the Lot#.
    - property id's are unique but lot#'s are unique only with the county
  - ⇨ The tax rate is constant within each county.
  - ⇨ The price is based on the area of the land.

Normalization Question 2

Given this database schema normalize into BCNF.



- ◆New FD5 says that the size of the parcel of land determines what county it is in.

Normalization to BCNF Question

Given this schema normalize into BCNF:

- ◆R (courseNum, secNum, offeringDept, creditHours, courseLevel, instructorSSN, semester, year, daysHours, roomNum, numStudents)
- ◆courseNum → offeringDept,creditHours, courseLevel
- ◆courseNum, secNum, semester, year → daysHours, roomNum, numStudents, instructorSSN
- ◆roomNum, daysHours, semester, year → instructorSSN, courseNum, secNum

Fourth Normal Form (4NF) and Fifth Normal Form (5NF)

Fourth normal form (4NF) and fifth normal formal (5NF) are rarely used in practice.

A relation is in ***fourth normal form*** (***4NF***) if it is in BCNF and contains no non-trivial multi-valued dependencies.

- ◆A multi-valued dependency (MVD) occurs when two independent, multi-valued attributes are present in the schema.

A relation is in ***fifth normal form*** (***5NF***) if the relation has no join dependency.

- ◆A join dependency implies that spurious tuples are generated when the relations are natural joined.

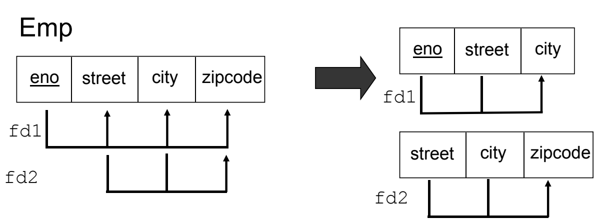
Normal Forms in Practice

Normal forms are used to prevent anomalies and redundancy. However, just because successive normal forms are better in reducing redundancy that does not mean they always have to be used.

For example, query execution time may increase because of normalization as more joins become necessary to answer queries.

Normal Forms in Practice Example

For example, `street` and `city` uniquely determine a `zipcode`.



In this case, reducing redundancy is not as important as the fact that a join is necessary every time the `zipcode` is needed.

- ◆When a `zipcode` does change, it is easy to scan the entire `Emp` relation and update it accordingly.

Normal Forms and ER Modeling

Normalization and ER modeling are two independent concepts.

You can use ER modeling to produce an initial relational schema and then use normalization to remove any remaining redundancies.

- ◆If you are a good ER modeler, it is rare that much normalization will be required.

In theory, you can use normalization by itself. This would involve identifying all attributes, giving them unique names, discovering all FDs and MVDs, then applying the normalization algorithms.

- ◆Since this is a lot harder than ER modeling, most people do not do it.

Normal Forms in Practice Summary

In summary, normalization is typically used in two ways:

- To improve on relational schemas after ER design.
- To improve on an existing relational schemas that are poorly designed and contain redundancy and potential for anomalies.

In practice, most designers make sure their schemas are in at least 3NF or BCNF because this removes most anomalies and redundancies. If multi-valued dependencies exist, they should definitely be removed to go to 4NF.

There is always a tradeoff in normalization: Normalization reduces redundancy but fragments the relations which makes it more expensive to query. Some redundancy may help performance.

- This is the classic time versus space tradeoff!

Conclusion

Normalization is produces relations with desirable properties and reduces redundancy and update anomalies.

Normal forms indicate when relations satisfy certain properties.

- 1NF - All attributes are atomic.
- 2NF - All attributes are fully functionally dependent on a key.
- 3NF - There are no transitive dependencies in the relation.
- 4NF - There are no multi-valued dependencies in the relation.
- 5NF - The relation has no join dependency.

In practice, normalization is used to improve schemas produced after ER design and existing relational schemas.

- Full normalization is not always beneficial as it may increase query time.

Objectives

- Explain process of normalization and why it is beneficial.
- Describe the concept of the Universal Relation.
- What are the motivations for normalization?
- What are the 3 types of update anomalies? Be able to give examples.
- Describe the lossless-join property and dependency preservation property.
- Define and explain the concept of a functional dependency.
- What is a trivial FD?
- Describe the relationship between FDs and keys.
- Be able to compute the closure of a set of attributes.

Objectives (2)

- Define normal forms, be able to list the normal forms, and draw a diagram showing their relationship.
- Define 1NF. Know how to convert non-1NF relation to 1NF.
- Define 2NF. Convert 1NF relation to 2NF.
- Define 3NF. Convert 2NF relation to 3NF.
- Define BCNF. Convert 3NF relation to BCNF. Compare BCNF and 3NF. Be able to convert directly to BCNF using algorithm.
- Explain normalization in terms of the time versus space tradeoff.
- Explain the relationship between normalization and ER modeling.