

Key Points – Normal Forms and Normalization

We want our relational schema to satisfy these design principles.

Careful design at the ER stage and careful application of the ER $\rightarrow$ relational model transformation rules can produce a good relational design.

- and problems detected can often be fixed up by hand

Normalization provides a formal method for removing redundancies.

- work through a series of *normal forms*, identifying *functional dependencies* that violate the definition of the normal form and splitting the relation to fix those problems
 - each normal form prohibits a certain kind of redundancy

Functional Dependencies

A *functional dependency* $X \rightarrow Y$ occurs when the value of a set of attributes X completely determines the value of another set of attributes Y .

- “there can be only one” – there can be only one instance of a particular set of values for Y for a particular set of values for X
 - does not necessarily mean that it is possible to compute the values for Y from the values of X
- existence of functional dependencies depends on the semantics of the attributes

Example.

STUDENT(sid, name, classyear, dean)

- classyear $\rightarrow$ dean [deans are assigned by class year]

Key Points – Normal Forms

- First Normal Form (1NF)
 - eliminates things that aren't valid relational schemas
- Second Normal Form (2NF)
 - eliminates partial dependencies – non-key attributes depend on a whole key
- Third Normal Form (3NF)
 - eliminates transitive dependencies – non-key attributes depend only on a key
- Boyce-Codd Normal Form (BCNF)
 - eliminates all redundancy due to functional dependencies
- Fourth Normal Form (4NF)
 - eliminates independent multi-valued dependencies
- Fifth Normal Form (5NF)
 - eliminates any remaining join dependencies

Questions

Is there an ideal normal form every database design should achieve?

- achieving 3NF or BCNF is typically good enough
- higher normal forms are not as useful in practice
 - violations are relatively rare and can be hard to detect
- may choose not to normalize to the highest possible normal form
 - balance removal of redundancy (more relations) with performance (fewer relations reduces need for joins)

Key Points – Normal Forms Definitions

- **First Normal Form (1NF)** eliminates things that aren't valid relational schemas
 - all values are atomic – no composite attributes
 - each tuple has a single value for each attribute (can be NULL) – no multivalued attributes
 - every relation has a primary key – no duplicate rows
- **Second Normal Form (2NF)** eliminates partial dependencies – non-key attributes only depend on a whole key
 - satisfies 1NF
 - for all FDs $X \rightarrow A$, either A has no non-key attributes or X is not a proper subset of a key
- **Third Normal Form (3NF)** eliminates transitive dependencies – non-key attributes depend only on a key
 - satisfies 2NF
 - for all non-trivial FDs $X \rightarrow A$, either X is a superkey or everything in $A-X$ is part of a key
- **Boyce-Codd Normal Form (BCNF)** eliminates all redundancy due to functional dependencies
 - for all non-trivial FDs $X \rightarrow A$, X is a superkey
 - in a *trivial* FD $X \rightarrow A$, X is a superset of A
 - a *superkey* is a set of attributes which uniquely identify tuples in the relation

4

Questions

What is a (proper) subset of a key?

- a *key* is a set of attributes which together uniquely identify tuples in the relation
- a *subset* of a set is made up of zero or more elements of the set
- a *proper subset* must have fewer elements than the set
 - a set S is a subset of itself
 - a set S is not a proper subset of itself

CPSC 343: Database Theory and Practice • Fall 2026

115

Determining Candidate Keys

- a relation may have more than one key
 - each is a *candidate key*
- normal form definitions refer to “a key” – any key, not necessarily just the primary key
- a *key* is a set of attributes K which together uniquely identify tuples in the relation
 - K is a key if its *closure* K^+ contains all of the attributes in the relation
- computing S^+
 - start with a set of attributes S
 - repeatedly
 - find an FD $X \rightarrow A$ where $X \subseteq S$ (X is entirely contained in the current S)
 - add A to S : $S = S \cup A$until nothing new gets added

CPSC 343: Database Theory and Practice • Fall 2026

116

Determining Candidate Keys

- observations
 - attributes only on the left side of an FD or not in any FD must be in every candidate key
 - attributes only on the right side of an FD will not be in any candidate key
 - attributes on both sides of an FD might be in a candidate key
- process
 - start with the “must have” core
 - add “both sides” attributes until the closure covers all of the attributes → candidate key!
 - repeat with different combinations of “both sides” attributes, skipping supersets of found keys (want only minimal keys)
- shortcut if you already have a primary key –
 - for any FD $X \rightarrow A$ where A is part of a known key, replace A with X and trim to a minimal key
 - $(K-A) \cup X$ is also a superkey
 - repeat with each new key until nothing else is found

7

1NF

- First Normal Form (1NF) eliminates things that aren't valid relational schemas
 - all values are atomic – no composite attributes
 - each tuple has a single value for each attribute (can be NULL) – no multivalued attributes
 - every relation has a primary key – no duplicate rows

OrderLine(OrderID, ProductID, Quantity, ProductName)

- ProductID → ProductName
- OrderID, ProductID → Quantity candidate key is (OrderID, ProductID)

Assignment(EmpID, SSN, Project, Hours)

- EmpID → SSN
- SSN → EmpID
- EmpID, Project → Hours candidate keys are (EmpID, Project), (SSN, Project)

Schedule(Room, Building, Time, Instructor, Course)

- Room → Building
- Room, Time → Course
- Instructor, Time → Room
- Course → Instructor candidate keys are (Time, Room), (Time, Instructor), (Time, Course)

9

2NF

- Second Normal Form (2NF) eliminates partial dependencies – non-key attributes only depend on a whole key
 - satisfies 1NF
 - for all FDs $X \rightarrow A$, either A has no non-key attributes or X is not a proper subset of a key

OrderLine(OrderID, ProductID, Quantity, ProductName)

⊘ ProductID → ProductName

OrderID	ProductID	Quantity	ProductName
1001	P7	2	Widget
1002	P7	5	Widget
1002	P9	1	Gadget

same product name repeats on every row with a given product ID
a product ID can be repeated because it is only part of a key

120

3NF

- Third Normal Form (3NF) eliminates transitive dependencies – non-key attributes depend only on a key
 - satisfies 2NF
 - for all non-trivial FDs $X \rightarrow A$, either X is a superkey or everything in A-X is part of a key

Employee(EmpID, Name, DeptID, DeptName)

- ⊘ DeptID → DeptName
- EmpID → DeptID [employee is in only one department]

EmpID	Name	DeptID	DeptName
1	Ana	D1	Sales
2	Ben	D1	Sales
3	Cal	D2	Support

same department name repeats on every row with a given department ID
a department ID can be repeated because it is not a key

121

BCNF

- Boyce-Codd Normal Form (BCNF) eliminates all redundancy due to functional dependencies
 - for all non-trivial FDs $X \rightarrow A$, X is a superkey
 - in a trivial FD $X \rightarrow A$, X is a superset of A
 - a superkey is a set of attributes which uniquely identify tuples in the relation

Assignment(EmpID, SSN, Project, Hours)

- ⊘ EmpID → SSN
- ⊘ SSN → EmpID [each employee has exactly one SSN and vice versa]
- EmpID, Project → Hours

EmpID	SSN	Project	Hours
E1	111	Alpha	10
E1	111	Beta	5
E2	222	Alpha	20

the SSN is repeated for every project an employee works on

122

Questions

Is it always the case that a higher normal form implies a lower normal form?

- e.g. if a schema is in 2NF, will it always also be 1NF?

- normalization is meant to be progressive
 - move from lower to higher normal forms

- in many cases, satisfying a lower normal form is explicitly included in the definition of a higher normal form

• Second Normal Form (2NF) eliminates dependencies on partial keys

- satisfies 1NF
- for all FDs $X \rightarrow A$, either A has no non-key attributes or X is not a proper subset of a key

• Third Normal Form (3NF) eliminates dependencies on non-key attributes

- satisfies 2NF
- for all non-trivial FDs $X \rightarrow A$, either X is a superkey or everything in A-X is part of a key

• Boyce-Codd Normal Form (BCNF) eliminates all redundancy due to functional dependencies

- for all non-trivial FDs $X \rightarrow A$, X is a superkey
- in a trivial FD $X \rightarrow A$, X is a superset of A
- a superkey is a set of attributes which uniquely identify tuples in the relation

Consider the following relation:

REGISTERED(student, course, instructor, major)

The functional dependencies are:

- student $\rightarrow$ major
- course $\rightarrow$ instructor

What normal form(s) is this schema in? Choose all that apply.

- ★ 1NF
- 2NF
- 3NF
- BCNF
- none of these

Consider the following relations:

REGISTERED(student, course)
STUDENT(student, major)
COURSE(course, title)

The functional dependencies are:

- student $\rightarrow$ major
- course $\rightarrow$ title

What normal form(s) is this schema in? Choose all that apply.

- ★ 1NF
- ★ 2NF
- ★ 3NF
- ★ BCNF
- none of these

Consider the following relations:

LOTS(propertyid, county, lotnum, area, price, taxrate)

The functional dependencies are:

- county, lotnum $\rightarrow$ propertyid
- county $\rightarrow$ taxrate
- area $\rightarrow$ price

What normal form(s) is this schema in? Choose all that apply.

- ★ 1NF
- ★ 2NF
- 3NF
- BCNF
- none of these