http://www.rlvision.com/blog/method-for-determining-candidate-keys-and-highest-normal-form-of-a-relation-based-on-functional-dependencies/

Database normalization

First normal form (1NF)

Relational schema is in 1NF if all occurrences of rows in the respective relational table contain the same number of fields and include the atomic values only, (there is no repeating fields and groups)

e#	name	car used
950001	Peter	Toyota, PKR234, Ford, WER545
932345	Paul	Honda, RTQ456
960020	Joan	Holden, KLR197, Holden, KLR567

Keys

Superkey
Superkey is a nonempty subset X of relational schema $R = (A_1, \ldots, A_n)$ such that for any two rows t_1, t_2 in a relational table defined over R $t_1[X] \neq t_2[X]$
If X is a superkey in R than $X \rightarrow (A_1, \ldots, A_n)$

Minimal key
Minimal key is a superkey K with an additional property such that removal of any attribute from K will cause K not to be a superkey

Keys

Primary key
Primary key is an arbitrarily selected minimal key

Candidate key
Candidate key is any other minimal key which is not primary

important key def here

Functional dependencies and keys

Let $R = (A_1, \ldots, A_n)$ be a relational schema (a header of relational table) and let X, Y be nonempty subsets of R such that $X \cup Y = R$

(1) If $X \rightarrow Y$ is valid in R then X is a key (!!!)

(2) If X is a key then $X \rightarrow Y$ is valid in R

Attributes

Prime attribute
Prime attribute is an attribute from relational schema R which is a member of at least one candidate key in R

Nonprime attribute
Nonprime attribute is an attribute which is a not prime

Full and partial functional dependencies

Full functional dependency

Full functional dependency is a functional dependency $X \rightarrow Y$ such that removal of any attribute A from X causes that $(X-A) \not\rightarrow Y$

Partial functional dependency

Partial functional dependency is a functional dependency which is not full

Second normal form (2NF)

Relational schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on a primary key of schema R

Second normal form (2NF)

Inventory

part	quantity	warehouse	warehouse-address

warehouse → warehouse-address

part, warehouse → quantity

part, warehouse → warehouse-address

part, warehouse → warehouse-address, quantity

Minimal key = (part, warehouse)

Second normal form (2NF)

Inventory

part	quantity	warehouse	warehouse-address

warehouse → warehouse-address

part, warehouse → quantity

Minimal key = (part, warehouse)

Schema Inventory is not in 2NF because nonprime attribute warehouse-address depends on a part (warehouse) of a key (part, warehouse)

Functional dependency which "violates 2NF" is warehouse → warehouse-address

Second normal form (2NF)

Inventory

part	quantity	warehouse	warehouse-address

warehouse → warehouse-address

part, warehouse → quantity

Minimal key = (part, warehouse)

If all minimal keys in a schema consist of only one attribute (single attribute keys) then such schema is always in 2NF

This is because any nonprime attribute in the schema does not depend on a part of a key (because each key consists of one attribute only !)

Second normal form (2NF)

Inventory

part	quantity	warehouse	warehouse-address

Schema Inventory should be decomposed into the following schemas

Store

part	quantity	warehouse

Location

warehouse	warehouse-address

Second normal form (2NF)

Store		
part	quantity	warehouse

part, warehouse → quantity

Minimal key = (part, warehouse)

Schema Store is in 2NF because no nonprime attributes (quantity) depends on a part of a key (part, warehouse)

Second normal form (2NF)

Location	
warehouse	warehouse-address

warehouse → warehouse-address

Minimal key = (warehouse)

Schema Location is in 2NF because no nonprime attributes (warehouse-address) depends on a part of a key (warehouse)

Second normal form (2NF)

Location	
warehouse	warehouse-address

warehouse → warehouse-address

Minimal key = (warehouse)

Every relational schema, which consists of at most 2 attributes is always in 2NF

Transitive functional dependency

Functional dependency X → Y valid in a relational schema R is a transitive functional dependency if there exists a nonempty subset Z of R, that is not a subset of any key in R and such that the functional dependencies X → Z and Z → Y are valid in R

We say that Y is transitively dependent on X in schema R if X → Y is valid in R and X → Y is a transitive functional dependency

Third normal form (3NF)

Relational schema R is in 3NF if it is in 2NF and no nonprime attribute of R is transitively dependent on the primary key

Third normal form (3NF)

Supplier			
s#	sname	company-name	city

s# → sname
s# → company-name
s# → city
company-name → city

Minimal key = (s#)

Schema Supplier is in 2NF because there are no nonprime attributes which are functionally dependent on a part of minimal key s#

Third normal form (3NF)

Supplier

s#	sname	company-name	city

s# → sname

s# → company-name

s# → city

company-name → city

Minimal key = (s#)

Schema Supplier is not in 3NF because attribute city is transitively dependent on s#

s# → company-name & company-name → city

Third normal form (3NF)

Supplier

s#	sname	company-name	city

Schema Supplier should be decomposed into the following schemas

Supplier-new

s#	sname	company-name

Company

company-name	city

Third normal form (3NF)

Supplier-new

s#	sname	company-name

s# → sname

s# → company-name

Minimal key = (s#)

Schema Supplier-new is in 3NF because no attribute is transitively dependent on s#

Third normal form (3NF)

Company

company-name	city

company-name → city

Minimal key = (company-name)

Schema Company is in 3NF because no attribute is transitively dependent on company-name

Third normal form (3NF)

Company

company-name	city

company-name → city

Every relational schema, which consists of at most 2 attributes is always in 3NF

Third normal form (3NF)

Alternative definition

A relational schema R is in 3NF if whenever a functional dependency X → A is valid in R then either:

(1) X is a superkey in R, or

(2) A is a prime attribute in R

Third normal form (3NF)

Supplier

s#	sname	company-name	city

s# → sname
s# → company-name
s# → city
company-name → city

Minimal key = (s#)

Schema Supplier is not in 3NF because
(1) attribute company_name is not a superkey and
(2) attribute city is not prime

Third normal form (3NF)

Location

city	street	zip-code

city, street → zip-code
zip-code → city

Minimal key$_1$ = (city, street)

Minimal key$_2$ = (zip-code, street)

Schema Location is in 3NF because attribute city used on the right hand side of functional dependency
zip-code → city is prime

Third normal form (3NF)

Location

city	street	zip-code
NY	55	484
NY	56	484
LA	55	473
LA	56	473
LA	57	474

Repetition of [LA ... 473] and [NY ... 484] is forced by functional dependency
zip-code → city

Boyce-Codd normal form (BCNF)

A relational schema R is in BCNF if whenever functional dependency X → A holds in R then:
(1) X is a superkey in R

~~(2) or A is a prime attribute in R~~ (3NF only !!!)

Boyce-Codd normal form (BCNF)

Location

city	street	zip-code

city, street → zip-code
zip-code → city

Minimal key$_1$ = (city, street)

Minimal key$_2$ = (zip-code, street)

Schema Location is not in BCNF because zip-code is not a superkey
zipcode → city & zip_code ↛ street

Boyce-Codd normal form (BCNF)

Location

city	street	zip-code

Schema Location should be decomposed into the following schemas

CS

street	zip-code

CZ

city	zip-code

Boyce-Codd normal form (BCNF)

CS

street	zip-code

Minimal key$_2$ = (zip-code, street)

Schema CS is in BCNF because does not exist a functional dependency whose left hand side is not a superkey

Boyce-Codd normal form (BCNF)

CZ

city	zip-code

zip-code → city

Minimal key = (zip-code)

Schema CZ is in BCNF because zip-code is a superkey

Boyce-Codd normal form (BCNF)

CS

street	zip-code

Minimal key = (zip-code, street)

CZ

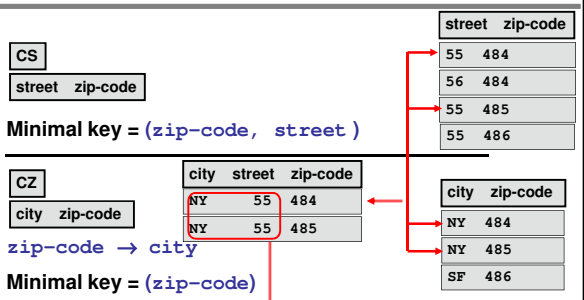
city	zip-code

zip-code → city

Minimal key = (zip-code)

Every relational schema, which consists of at most 2 attributes is always in BCNF

Boyce-Codd normal form (BCNF)



Normalization to BCNF costs a functional dependency

city, street → zip-code

Normalization of relational schemas

Given a relational schema R = { A_1, ..., A_n}

Identify all functional dependencies

Use functional dependencies to derive all minimal keys

Use functional dependencies and minimal keys to identify the highest normal form satisfied by R

Decompose R into the schemas in BCNF (3NF)

Normalization of relational schemas

Shipment

s#	city	status	p#	quantity

s# → city
s# → status
city → status
s#,p# → quantity
s#,p# → city
s#,p# → status

Schema Shipment is in 1NF

Minimal key = (s#,p#)

Normalization of relational schemas

Shipment

s# city status p# quantity

s# → city
s# → status
city → status
s#,p# → quantity
s#,p# → city
s#,p# → status

Schema Shipment is not in 2NF

Nonprime attribute city depends on a part of a key (s#,p#)

Minimal key = (s#,p#)

Normalization of relational schemas

Shipment

s# city status p# quantity

Schema Shipment should be decomposed into the following schemas

SP

s# p# quantity

Schema SP is in BCNF

s#,p# → quantity

Supplier

s# city status

Schema Supplier is not in 3NF(& BCNF)

s# → city
city → status
s# → status

Normalization of relational schemas

Supplier

s# city status

Schema Supplier should be decomposed into the following schemas

Supplier'

s# city

Schema Supplier' is in BCNF

s# → city

Location

city status

Schema Location is in BCNF

city → status

Normalization of relational schemas

Supplier

s# city status

Schema Supplier may be alternatively decomposed into the following schemas

Supplier'

s# city

Schema Supplier' is in BCNF

s# → city

Supplier''

s# status

Schema Supplier'' is in BCNF

s# → status

References

Elmasri R., Navathe S. B., *Fundamentals of Database Systems*, chapters 10.3,10.4,10.5

R. Ramakrishnan, J. Gehrke *Database Management Systems*, chapters 19.2, 19.3