Normalization in the relational model.

An anomaly is an inconsistent, incomplete, or contradictory state of the database

- **Insertion anomaly** – user is unable to insert a new record when it should be possible to do so
- **Deletion anomaly** – when a record is deleted, other information that is tied to it is also deleted
- **Update anomaly** –a record is updated, but other appearances of the same items are not updated

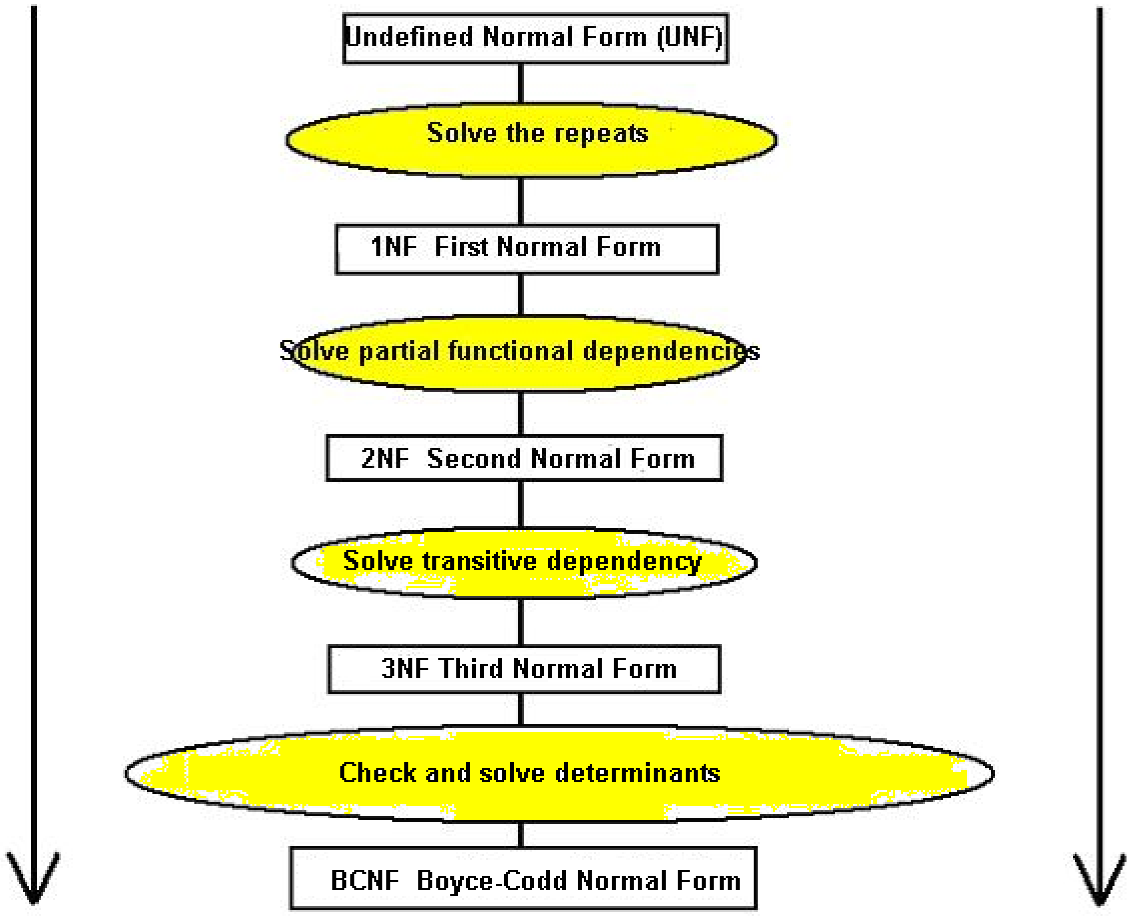
The purpose of the normalization in a database environment is:

- to develop a good description of the data, its relationships and constraints
- to produce a stable set of tables(relations) that
  - is a faithful model of the enterprise
  - is highly flexible
  - reduces redundancy-saves space and reduces inconsistency in data.
  - is free of update, insertion and deletion anomalies

Example 1

CourseNo	studID	studLastName	fID	schedule	room	grade
ART103A	S1001	Smith	F101	MWF9	H221	A
ART103A	S1010	Burns	F101	MWF9	H221	
ART103A	S1006	Lee	F101	MWF9	H221	B
CSC201A	S1003	Jones	F105	TUTHF10	M110	A
CSC201A	S1006	Lee	F105	TUTHF10	M110	C
HST205A	S1001	Smith	F202	MWF11	H221	

- **Update anomaly**: If schedule of ART103A is updated in first record, and not in second and third – inconsistent data
- **Deletion anomaly**: If record of student S1001 is deleted, information about HST205A class is lost also
- **Insertion anomaly**: It is not possible to add a new class, for MATH101A , even if its teacher, schedule, and room are known, unless there is a student registered for it, because the key contains **studID**



A table (relation) is in an UNF (undefined normal form) if it contains two or more data in a cell.

studID	**studLastName**	**major**	**credits**	**status**	**socSecNbr**
S1001	Smith	History	90	Senior	100222500
S1003	Jones	Math	95	Senior	222333444
S1006	Lee	**CSC** **Math**	15	Freshman	088111222
S1010	Burns	Art	63	Junior	099111222
S1060	Jones	CSC	25	Freshman	064624123

The “**major”** attribute is not single-valued for each tuple.

1NF First Normal Form

1. A relation is in **1NF** if every attribute is single-valued for each tuple
2. Each cell of the table has only one value in it
3. Domains of attributes are **atomic**: no sets, lists, repeating fields or groups allowed in domains

UNF to 1NF

Solve the repeats

1. For each multi-valued attribute, create a new table, in which you place the **key of the original table and the multi-valued attribute**. Keep the original table, with its key.

studID	studLastName	credits	status	socSecNbr
S1001	Smith	90	Senior	100222500
S1003	Jones	95	Senior	222333444
S1006	Lee	15	Freshman	088111222
S1010	Burns	63	Junior	099111222
S1060	Jones	25	Freshman	064624123

studID	major
S1001	History
S1003	Math
S1006	Math
S1006	CSC
S1010	Art
S1060	CSC

2. If the number of repeats is limited, make additional columns for multiple values

studID	studLastName	major1	major2	credits	status	socSecNbr
S1001	Smith	History		90	Senior	100222500
S1003	Jones	Math		95	Senior	222333444
S1006	Lee	CSC	Math	15	Freshman	088111222
S1010	Burns	Art		63	Junior	099111222
S1060	Jones	CSC		25	Freshman	064624123

3. "Flatten" the original table by making the multi-valued attribute part of the key

studID	studLastName	major	credits	status	socSecNbr
S1001	Smith	History	90	Senior	100222500
S1003	Jones	Math	95	Senior	222333444
S1006	Lee	CSC	15	Freshman	088111222
S1006	Lee	Math	15	Freshman	088111222
S1010	Burns	Art	63	Junior	099111222
S1060	Jones	CSC	25	Freshman	064624123

2NF Second Normal Form

The 2NF is based on the concept of functional dependency.

A **functional dependency** (FD) is a type of relationship between attributes

If **α** and **β** are sets of attributes of relation R, we say that **β** is functionally dependent on **α** if each **α** value in R has associated with it exactly one value of **β** in R.

Alternatively, if two tuples have the same **α** values, they must also have the same **β** values

Write **α → β**, read α functionally determines β, or **β** is functionally dependent on **α.**

FD is actually a many-to-one relationship between A and B

$(A_1, B_1, C_1, D_1, E_1, F_1) \rightarrow (I_1, J_1, K_1, L_1, M_1)$

Example 2

Let R be the table Student (studID, studLastName, credits, status, socSecNbr)

studID	**studLastName**	**credits**	**status**	**socSecNbr**
S1001	Smith	90	Senior	100222500
S1003	Jones	95	Senior	222333444
S1006	Lee	15	Freshman	088111222
S1010	Burns	63	Junior	099111222
S1060	Jones	25	Freshman	064624123

FDs in R include

{studID}→{studLastName}, but not the reverse

{studID} →{studLastName, credits, status, socSecNbr, stuId}

{socSecNbr}→{studID, studLastName, credits, status, socSecNbr}

{credits}→{status}, but not {status}→{credits}

We say that the functional dependency X→Y is trivial if set {Y} is a subset of set {X}

***Example 3*: If A and B are attributes of R,**

- **{A}→{A}**
- **{A,B} →{A}**
- **{A,B} →{B}**
- **{A,B} →{A,B}**

are all trivial FDs

Keys

- **Superkey** – functionally determines all attributes in a relation
- **Candidate key** - superkey that is a minimal identifier (no extraneous attributes)

- **Primary key** - candidate key actually used
- Primary key has **no-null** constraint and **uniqueness** constraint
- Should also enforce uniqueness and no-null rule for candidate keys

In a relation R, the set of attributes β is **fully functionally dependent** on set of attributes α of R

- if β is functionally dependent on α
- but not functionally dependent on any proper subset of α

α is the smallest determinant for β. This means every attribute in α is needed to functionally determine β

Example 3.

In table Student1(courseNo, studID, studLastName, facID, schedule, room, grade) some functional dependency are:

{courseNo, studID} → {studLastName}

{courseNo, studID} →{facID}

{courseNo, studID} →{schedule}

{courseNo, studID} →{room}

{courseNo, studID} →{grade}

courseNo → facID **partial FD

courseNo → schedule **partial FD

courseNo →room ** partial FD

studID → studLastName ** partial FD

A relation is in 2NF if it is in 1NF and all the non-key attributes are fully functionally dependent on the key.

- No non-key attribute is FD on just part of the key
- If key has only one attribute, and R is 1NF, R is automatically 2NF

1NF to 2NF

Solve partial functional dependency

- Identify each partial FD
- Remove the attributes that depend on each of the determinants so identified
- Place these determinants in separate relations along with their dependent attributes

- In original relation keep the composite key and any attributes that are fully functionally dependent on all of it
- Even if the composite key has no dependent attributes, keep that relation to connect logically the others

Example 4.

Student1 (courseNo, studID, studLastName, facID, schedule, room, grade)

FDs grouped by determinant:

{courseNo} → {courseNo, facID, schedule, room}

{stuId} → {studID, studLastName}

{courseNo, studID} → {courseNo, studID, facID, schedule, room, studLastName, grade}

Replace Student1 by tables grouped by determinants:

Course (courseNo, facID, schedule, room)

Student2(studID, studLastName)

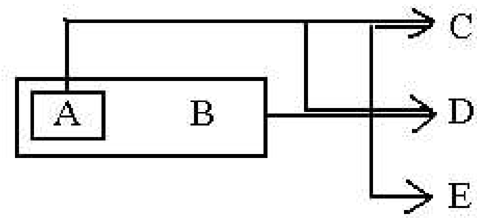
Student3(courseNo, studID, grade)

Example 5.

Given the table R (<u>A, B</u>, C, D, E).

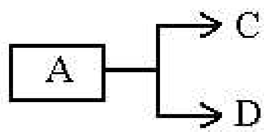
R (A, B, C, D, E)　　　　key A, B

$A \longrightarrow C$
$A \longrightarrow D$
$A, B \longrightarrow C$
$A, B \longrightarrow D$
$A, B \longrightarrow E$



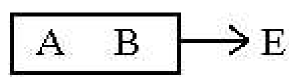
We replace R by the tables R1 and R2

R1 (<u>A</u>, C, D)　key A



and

R2 (<u>A, B</u>, E) key (A, B)



Example 6.

Given the table R (<u>A, B</u>, C. D, E)

R (<u>A, B</u>, C, D, E)　key A, B

$A, B \rightarrow C$
$A, B \rightarrow D$
$A, B \rightarrow E$
$A \rightarrow C$
$A \rightarrow D$
$B \rightarrow E$

Replace R by R1 and R2

R1 (<u>A</u>, C, D)　key A

R2 (<u>B</u>, E) key B

Example 7.

Given R (<u>Code-Supplier, Code-Product</u>, Quantity, Name-Supplier)

Code-Supplier	Code-Product	Quantity	Name-Supplier
F1	P1	10	TEK
F1	P2	20	TEK
F1	P3	30	TEK
F2	P2	12	SONY
F2	P4	10	SONY
F3	P3	12	BLII

FDs

(Code-Supplier, Code-Product)→Quantity

(Code-Supplier, Code-Product)→Name-Supplier

Code-Supplier→Name-Supplier

Replace R by tables R1 and R2.

R1

Code-Supplier	Code -Product	Quantity
F1	P1	10
F1	P2	20
F1	P3	30
F2	P2	12
F2	P4	10
F3	P3	12

R2

Code- Supplier	Name- Supplier
F1	TEK
F2	SONY
F3	BLII

How is better, R or R1 and R2?

Add

We like to add a new supplier without knowing its products

- **Can't add in R**
- We can add it in R_2.

Delete

Suppose the supplier F3 stops to supply the product P3.

- **Delete tuple for F3, P3 from R. We loose all information concerning**

supplier F3.

- Delete F3, P3 in R1. No lost of information about F3.

Update

Change the name of supplier F1 to XTER.

- **Scan all tuples in R to find and update F1.**
- Change one single tuple in R2 (key Code-Supplier).

Transitive dependency

If A, B, and C are attributes of table R, such that A →B, and B → C, then C is transitively dependent on A

Example 8

Student (studID, studLastName, major, credits, status)

FD:

credits → status

studID → credits

by transitivity it implies studID→status

Transitive dependencies cause update, insertion, deletion anomalies.

3NF Third Normal Form

A relation is in **third normal form** (3NF) if whenever a non-trivial functional dependency X→A exists, then either X is a superkey or A is a member of some candidate key

- **To be 3NF, relation must be 2NF and have no transitive dependencies**
- **No non-key attribute determines another non-key attribute. Here key includes "candidate key"**

2NF to 3NF

Solve transitive dependency

- Remove the dependent attribute from the relation
- Create a new table with the dependent attribute and its determinant
- Keep the determinant in the original table

Example: Given the relation R (A, B, C).

R(A, B, C)   key A

A → B
A → C
B → C
B not → A

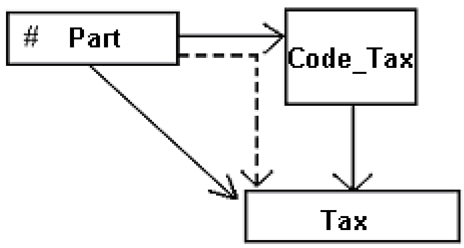
Replace R by

R1(<u>A</u>, B) key A

R2(<u>B</u>, C) key B

Example 9 : Given the relation R (<u># Part</u>, Code_Tax, Tax).

# Part	Code_Tax	Tax
P1	1	18.6%
P2	1	18.6%
P3	2	33%
P4	1	18.6%
P5	1	18.6%



Replace R by R_1(# Part, Code_Tax) and R_2(Code_Tax, Tax)

R_1

# Part	Code_Tax
P1	1
P2	1
P3	2
P4	1
P5	1

R_2

Code_Tax	Tax
1	18.6%
2	33%

How is better, R or R1 and R2?

Add

We like to add a new 8.6% tax with code 3

- No way to add it in R. The key is not defined for only tax code (?, 3, 8.6%).
- New tuple in R2 with Code_Tax 3.

Delete

Delete part P_3

- Delete P3 in R. BUT we loose the information concerning the tax (33%) for Code_Tax =2.
- Delete P3 in R1. No lost of information.

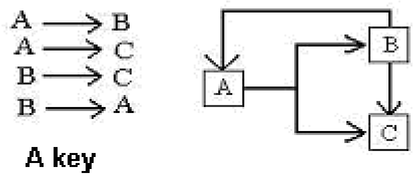
Update

Change Tax at 17% for Code_Tax =1.

- Scan all tuples of R to find tuples having Code_Tax=1 to make the update.
- One tuple update in R_2.

Observation:

The relation R (A, B, C) has to candidate keys A and B, A is the primary key.



This table is in 3NF.

Example 10

Student (<u>studID</u>, studLastName, major, credits, status)

with FD credits→status

- Remove the dependent attribute, *status*, from the relation
- Create a new table with the dependent attribute and its determinant, *credits*
- Keep the determinant in the original table

Student2 (<u>studID</u>, studLastName, major, credits)

Stats (<u>credits</u>, status)

BCNF Boyce-Codd Normal Form

A relation is in Boyce/Codd Normal Form (BCNF) if whenever a non-trivial functional dependency X→A exists, then X is a superkey

If there is just one single candidate key, the forms are equivalent

3NF to BCNF

Check and solve determinants

Look in a 3NF table for:

- two or more composites candidates keys
- two candidates keys having common attributes

1. **identify all candidate keys in R.**
2. **identify all FD in R R.**
3. **if there exists determinants which are not candidate keys , replace R by R_1 and R_2 and put in R_2 the determinants and the corresponding tuples**

Example 11

Given the table R (<u>M, N</u>, O, P)

.

(M, N) → O

(M, N) → P

P → M

Replace R by

$R_1(\underline{M, N}, O)$

$R_2(\underline{P}, M)$

$R_3(\underline{N, P})$

4. If there are common attributes between candidate keys replace R by R1 and R2 such as the common attribute is in only one table.

Example 12.

The suppliers' names are unique. There are two candidate keys.

(SupplierId, ProductId) and (SupplierName, ProductId)

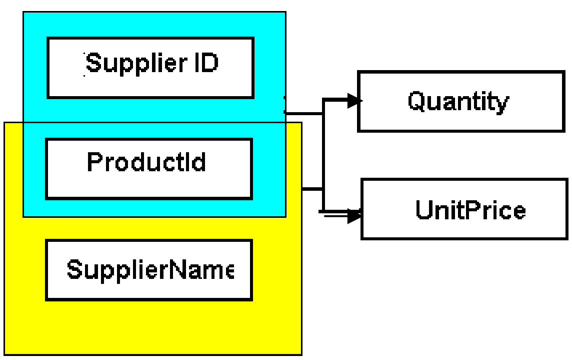
and there are functional dependencies between parts of the candidate keys.

SuppilerID → SupplierName

SupplierName → SupplierID

Non_BoyceCode : Table

SupplierId	SupplierName	ProductId	Quantity	UnitPrice
5	Cooperativa de Quesos "Las Cabras"	11	54	14
6	Mayumi's	42	10	234
70	Fromage Fortini	72	5	43.65
70	Fromage Fortini	11	15	14.25
0		0	0	0



BoyceCode_2 : Table

	SupplierId	ProductId	Quantity	UnitPrice
	5	11	54	14
	6	42	10	234
	70	72	5	43.65
	70	11	15	14.25
*	0	0	0	0

Record: 1 of 4

BoyceCode_1 : Table

	SupplierId	SupplierName
	5	Cooperativa de Quesos "Las Cabras"
	6	Mayumi's
	70	Fromage Fortini
*	0	

Record: 1 of 3