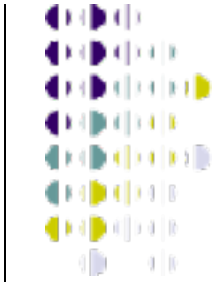
Normalization

Edited by: Nada Alhirabi

Normalization:Why do we need to normalize?



1. **To avoid redundancy (less storage space needed, and data is consistent)**

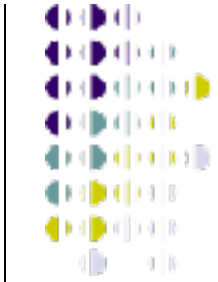
Ssn	c-id	Grade	Name	Address
123	cs331	A	smith	Main
123	cs351	B	smith	Main

2. **To avoid update/delete anomalies**

Ssn	c-id	Grade	Name	Address
123	cs33	1A	smith	Main
…	…	…	…	…
234	null	null	jones	Forbes

Insertion anomaly: Cannot make a record Jones' address because he is not taking any classes

Normal Forms



- First Normal Form – 1NF
- Second Normal Form – 2NF
- Third Normal Form – 3NF:
  - In practice, "normalized" means in BCNF or 3NF
- Fourth Normal Form – 4NF
- Fifth Normal Form – 5NF
- Boyce-Codd Normal Form – BCNF

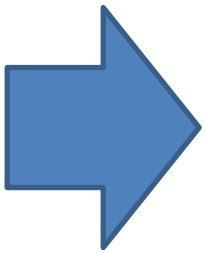
Only those are covered

First Normal Form (1NF)

- **1NF: all attributes are atomic ("no repeating groups")**

Last Name	First Name
Smith	Peter Mary John
Greg	Anne Michael

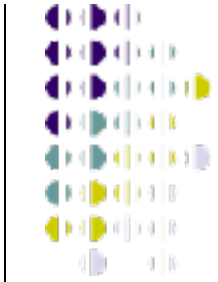
Not in 1NF



Normalized to 1NF

Last Name	First Name
Smith	Peter
Smith	Mary
Smith	John
Greg	Anne
Greg	Michael

Second Normal Form (2NF)

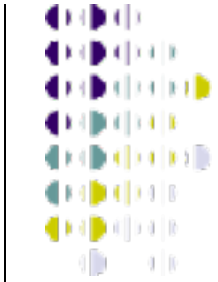


- **2NF:**
  - **1NF and**
  - **all non-key attributes are fully dependent on the PK ("no partial dependencies")**

Student	Course_ID	Grade	Address
Erik	CIS331	A	80 Ericsson Av.
Sven	CIS331	B	12 Olafson St.
Inge	CIS331	C	192 Odin Blvd.
Hildur	CIS362	A	212 Reykjavik St.

Not in 2NF

Second Normal Form (2NF)

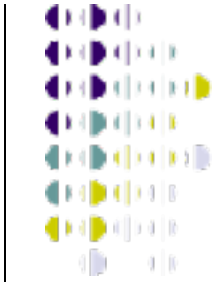


Student	Address
Erik	80 Ericsson Av.
Sven	12 Olafson St.
Inge	192 Freya Blvd.
Hildur	212 Reykjavik St.

Student	Course_ID	Grade
Erik	CIS331	A
Sven	CIS331	B
Inge	CIS331	C
Hildur	CIS362	A

Normalized to 2NF

Third Normal Form (3NF)



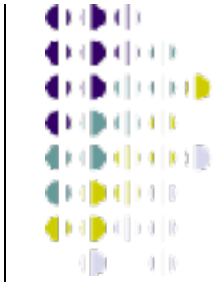
3NF:

- **2NF and**
- **no transitive dependencies**
  - Transitivity: If A → B and B → C, then A → C

Student	Course_ID	Grade	Grade_value
Erik	CIS331	A	4.00
Sven	CIS331	B	3.00
Inge	CIS331	C	2.00
Hildur	CIS362	A	4.00

Not in 3NF

Third Normal Form (3NF)

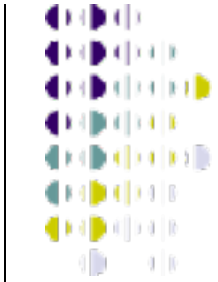


Student	Course_ID	Grade
Erik	CIS331	A
Sven	CIS331	B
Inge	CIS331	C
Hildur	CIS362	A

Grade	Grade_value
A	4.00
B	3.00
C	2.00

Normalized to 3NF

Extra examples



Example1: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123 8123450987

Two employees (Jon & Lester) are having two mobile numbers so the company stored them in the same field as you can see in the table above.

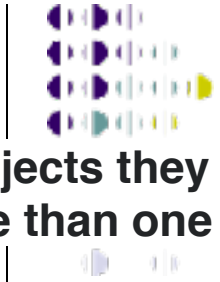
1NF: “each attribute of a table is atomic”

This table is **not in 1NF** as the rule says “each attribute of a table must have atomic (single) values”,
(the emp_mobile values for employees Jon & Lester violates that rule.)

To make the table complies with 1NF we should have the data like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123
104	Lester	Bangalore	8123450987

Second normal form (2NF)



Example2: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

teacher_id	subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate Keys: {teacher_id, subject}
Non prime attribute: teacher_age

The table is in 1 NF because each attribute has atomic values. **However**, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says "**no** non-prime attribute is dependent on the proper subset of any candidate key of the table".

To make the table complies with 2NF we can break it in two tables like this:

teacher_details table:

teacher_id	teacher_age
111	38
222	38
333	40

teacher_subject table:

teacher_id	subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Third Normal form (3NF)

Example3: Suppose a company wants to store the complete address of each employee, they create a table named employee_details that looks like this:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

Super keys: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}…so on
Candidate Keys: {emp_id}
Non-prime attributes: all attributes except emp_id are non-prime as they are not part of any candidate keys.
Here, emp_state, emp_city & emp_district dependent on emp_zip. And, emp_zip is dependent on emp_id that makes non-prime attributes (emp_state, emp_city & emp_district) transitively dependent on super key (emp_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

employee table:

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

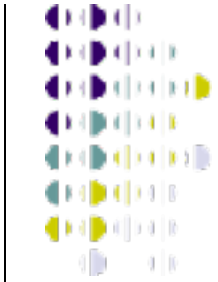
employee_zip table:

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

Normalization: Final Thoughts

- There are higher normal forms (4NF, 5NF), but we will not talk about them
- In practice, "normalized" means in BCNF or 3NF
- Luckily, in practice, ER diagrams lead to normalized tables (but do not rely on luck)

Normalization: summary



- **Why do we normalize?**
  - **To avoid redundancy (less storage space needed, and data is consistent)**
  - **To avoid update/delete anomalies**
- **A good decomposition should:**
  - **be a lossless join decomposition**
    (you can recover original tables with a join)
  - **preserve dependencies**
    (FD's should not span two tables)
- **1NF (all attributes are atomic)**
- **2NF (no partial dependencies)**
- **3NF (no transitive dependencies)**

Source for the most slides are taken from : Vladimir Vacic, Temple University and http://beginnersbook.com/2015/05/normalization-in-dbms/ website.