Lecture 14: Database Normalization

BADM/ACCY 352

Spring 2017

Instructor: Yi Yang, PhD

Previously

- How to create a database with constraints

This lecture

- We rethink data modeling and learn database normalization.

A note for Lab4

- If you use a Mac computer
- AND your MySQL Workbench is 6.3.9
- AND your MacOS is 10.12, aka, Sierra
- you may not be able to use “forward engineering”
- Send me an email.

A letter from previous student

"I worked as a data analyst at an investment bank, and my job is to analyze the risk and return of our trading team…. I was surprised that many tables in my team database system do not even have a primary key. There are many data duplicates….Things get even worse if I want to join tables, then I will get even more data duplicates…I find normalization super useful…"

Motivation

- What are the advantages of using relational database for data management?

- How to improve a bad database design?
- How to go from Excel to Relational database?

Motivation

SalesStaff						
EmployeeID	**SalesPerson**	**SalesOffice**	**OfficeNumber**	**Customer1**	**Customer2**	**Customer3**
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
1004	John Hunt	New York	212-555-1212	Dell	HP	Apple
1005	Martin Hap	Chicago	312-555-1212	Boeing		

Motivation

A good database design minimizes data redundancy and data duplication, avoids data anomalies, and makes data retrieval easy.

SalesStaff						
EmployeeID	**SalesPerson**	**SalesOffice**	**OfficeNumber**	**Customer1**	**Customer2**	**Customer3**
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
1004	John Hunt	New York	212-555-1212	Dell	HP	Apple
1005	Martin Hap	Chicago	312-555-1212	Boeing		

Red flag! too many information in one table.

Insert anomaly

We cannot record a new sales office until we also know the sales person. (Why?)

EmployeeID	SalesPerson	SalesOffice	OfficeNumber	Customer1	Customer2	Customer3
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
1004	John Hunt	New York	212-555-1212	Dell	HP	Apple
1005	Martin Hap	Chicago	312-555-1212	Boeing		
???	???	Atlanta	312-555-1212			

Update anomaly

if the office number changes, then there are multiple updates that need to be made.

EmployeeID	SalesPerson	SalesOffice	OfficeNumber	Customer1	Customer2	Customer3
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
1004	John Hunt	New York	212-555-1212	Dell	HP	Apple
1005	Martin Hap	Chicago	312-555-1212	Boeing		

Delete anomaly

if John Hunt retires, then deleting that row causes user to lose information about the New York office.

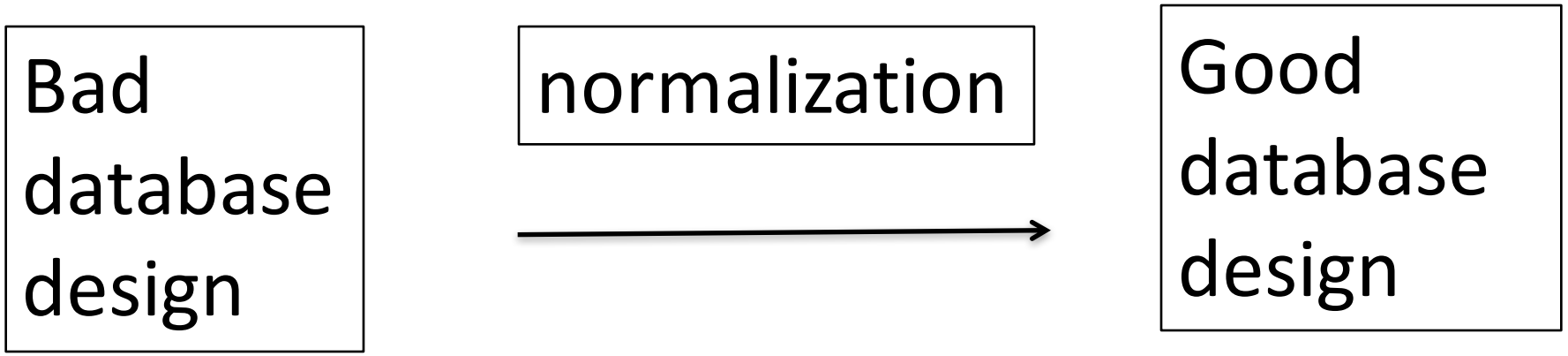
EmployeeID	SalesPerson	SalesOffice	OfficeNumber	Customer1	Customer2	Customer3
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
~~1004~~	~~John Hunt~~	~~New York~~	~~212-555-1212~~	~~Dell~~	~~HP~~	~~Apple~~
1005	Martin Hap	Chicago	312-555-1212	Boeing		

Data retrieval issue

If you want to search for a specific customer such as Ford, you would have to write a query like

```
SELECT SalesOffice
FROM SalesStaff
WHERE Customer1 = 'Ford' OR
      Customer2 = 'Ford' OR
      Customer3 = 'Ford'
```

How about ranking the table by customer?


Bad database design
normalization
Good database design

Database Normalization

- Normalization is a process for evaluating and correcting table schema to minimize data redundancies and data duplications, thereby reducing the likelihood of data anomalies.
- Different stage you can achieve
  - first normal form (1NF)
  - second normal form (2NF)
  - third normal form (3NF)

Dependency revisit

- Dependency (between two attributes): if each value of A determines one and only one value of B, then attribute A functionally determines attribute B, i.e., A → B

SalesStaff						
EmployeeID	**SalesPerson**	**SalesOffice**	**OfficeNumber**	**Customer1**	**Customer2**	**Customer3**
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
1004	John Hunt	New York	212-555-1212	Dell	HP	Apple
1005	Martin Hap	Chicago	312-555-1212	Boeing		

EmployeeID (PK) -> all the rest attributes;
SalesOffice -> OfficeNumber;

```
SELECT P_CODE, P_DESCRIPT, P_PRICE, V_CODE FROM PRODUCT;
```

P_CODE	P_DESCRIPT	P_PRICE	V_CODE
13-Q2/P2	7.25-in. pwr. saw blade	14.99	21344
14-Q1/L3	9.00-in. pwr. saw blade	17.49	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	39.95	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	43.99	23119
2232/QTY	B&D jigsaw, 12-in. blade	109.92	24288
2232/QWE	B&D jigsaw, 8-in. blade	99.87	24288
2238/QPD	B&D cordless drill, 1/2-in.	38.95	25595
23109-HB	Claw hammer	9.95	21225
23114-AA	Sledge hammer, 12 lb.	14.40	NULL
54778-2T	Rat-tail file, 1/8-in. fine	4.99	21344
89-WRF-Q	Hicut chain saw, 16 in	256.99	24288

Does P_DESCRIPT determine P_PRICE?

```
SELECT EMP_NUM, EMP_FNAME, EMP_LNAME, EMP_DOB FROM EMP;
```

EMP_NUM	EMP_FNAME	EMP_LNAME	EMP_DOB
100	George	Kolmycz	1945-06-15 00:00:00
102	Rhett	VanDam	1958-11-14 00:00:00
103	Anne	Jones	1974-10-16 00:00:00
104	John	Lange	1971-11-08 00:00:00
105	Robert	Williams	1975-03-14 00:00:00
106	Jeanine	Smith	1968-02-12 00:00:00
107	Jorge	Diante	1974-08-21 00:00:00
108	Paul	Wiesenbach	1966-02-14 00:00:00
109	George	Smith	1961-06-18 00:00:00
110	Leighla	Genkazi	1970-05-19 00:00:00

Does EMP_FNAME determine EMP_DOB?

Partial Dependency

- Attribute C is only determined by part of the primary key.
  - If (A, B) → (C, D) & B → C, then B → C is a partial dependency
  - a table of students may have student id, student name, course id, course name and grades

STU_ID	STU_NAME	COURSE_ID	COURSE_NAME	GRADES

Transitive Dependency

- if A → B, B → C, and A is the PK

  then A → C is a transitive dependency
- a table of cities may have city id, city name, state name, state capital

City_ID	State_Name	State_Captial

- How to identify?
  - A dependency exists among non-prime key attributes.

1NF

- A table is in 1NF when:
  - PK is defined
  - Each table cell contains only one value, not set of values

- How:
  - Divide multi-value attributes to separate rows.
  - Identify PK.

1NF

Un-normalized table

Student	Course	Grade
Alice	Database, Accounting	87, 90
Bob	Finance	84
Charlie	Database	86

after 1NF

sID	Student	cID	Course	Grade
s1	Alice	c1	Database	87
s1	Alice	c2	Accounting	90
s2	Bob	c3	Finance	84
s3	Charlie	c1	Database	86

Does this satisfy 1NF?

Student	Course	Grade
Alice	Database	87
Alice	Accounting	90
Bob	Finance	84
Charlie	Database	86

or

ID	Student	Course	Grade
1	Alice	Database	87
2	Alice	Accounting	90
3	Bob	Finance	84
4	Charlie	Database	86

2NF

- A table is in 2NF when:
  - It's in 1NF
  - It contains no partial dependencies.

- How:
  - Make new tables to eliminate partial dependencies
  - Each component of PK will become the PK in a new table

2NF

1NF table

sID	Student	cID	Course	Grade
s1	Alice	c1	Database	87
s1	Alice	c2	Accounting	90
s2	Bob	c3	Finance	84
s3	Charlie	c2	Database	86

PK: (sID, cID)

sID -> Student
cID -> Course
(sID, cID) -> Grade

Partial dependencies!

2NF table

sID	Student
s1	Alice
s2	Bob
s3	Charlie

cID	Course
c1	Database
c2	Accounting
c3	Finance

sID	cID	Grade
s1	c1	87
s1	c2	90
s2	c3	84
s3	c2	86

Question

If 1NF table does not have a composite primary key, then the table is already in 2NF.

a) True

b) False

3NF

- A table is in 3NF when
  - It's in 2NF
  - It contains no transitive dependency.

- How:
  - Make new tables to eliminate transitive dependencies

- Optimal relational DB requires that all tables be at least in 3NF.

3NF example

2NF table

TABLE_BOOK_DETAIL

Book ID	Genre ID	Genre Type	Price
1	1	Gardening	25.99
2	2	Sports	14.99
3	1	Gardening	10.00
4	3	Travel	12.99
5	2	Sports	17.99

PK: (BookID)

BookID -> Genre ID

BookID -> Price

GenreID -> Genre Type

Transitive dependencies!

3NF table

TABLE_BOOK

Book ID	Genre ID	Price
1	1	25.99
2	2	14.99
3	1	10.00
4	3	12.99
5	2	17.99

TABLE_GENRE

Genre ID	Genre Type
1	Gardening
2	Sports
3	Travel

3NF example

1NF table,
also in 2NF

ID	Student	Course	Grade
1	Alice	Database	87
2	Alice	Accounting	90
3	Bob	Finance	84
4	Charlie	Database	86

PK: (ID)

ID -> Student
ID -> Course
ID -> (Student, Course)
(Student, Course) -> Grade

Partial dependencies!

ID	Student	Course
1	Alice	Database
2	Alice	Accounting
3	Bob	Finance
4	Charlie	Database

Student	Course	Grade
Alice	Database	87
Alice	Accounting	90
Bob	Finance	84
Charlie	Database	86

Normalization Summary

To convert an un-normalized table into 3NF:

- 1NF
- 2NF
- 3NF
- In 2NF and 3NF, list PK and all dependencies, identify partial dependency (2NF) and transitive dependency (3NF)
- Check the tables to make sure they meet the requirements.

Practice

Which normalization form of this table is?

a) 1NF

b) 2NF

c) 3NF

TABLE_PURCHASE_DETAIL

Customer ID	Store ID	Purchase Location
1	1	Los Angeles
1	3	San Francisco
2	1	Los Angeles
3	2	New York
4	3	San Francisco

How do you normalize this table?

Practice

TABLE_PURCHASE_DETAIL

Customer ID	Store ID	Purchase Location
1	1	Los Angeles
1	3	San Francisco
2	1	Los Angeles
3	2	New York
4	3	San Francisco

PK: (Customer ID, Store ID)

StoreID -> Purchase_Location

Partial dependencies!

TABLE_PURCHASE

Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3

TABLE_STORE

Store ID	Purchase Location
1	Los Angeles
2	New York
3	San Francisco

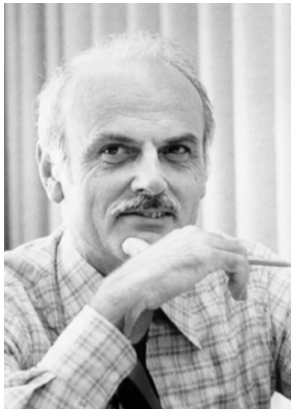
2NF

"The key, the whole key, and nothing but the key, so help me Codd."

1NF

3NF

Dr. Edgar Codd



When to use normalization

- After initial relational modeling design
  - Use normalization to analyze the relationships among the attributes within each entity, and determine if the structure can be improved through normalization.
- Redesign a data model
  - Modify existing data that can be in text files, spreadsheets, or older database model.

The goals of normalization (1)

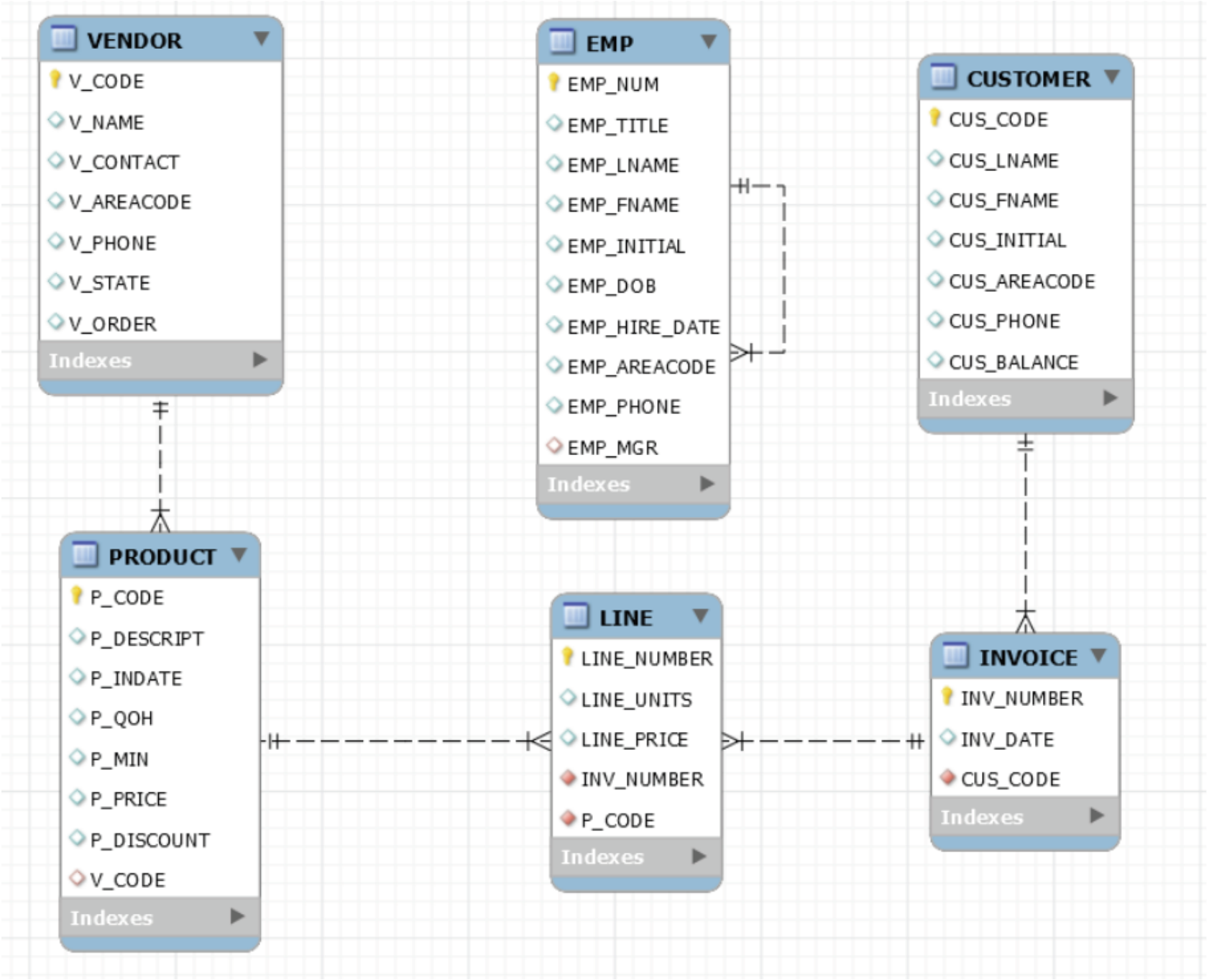
- Each table presents a single subject.
  - a COURSE table only contains course data;
  - a STUDENT table only contains student data.

- No data will be *unnecessarily* stored in more than one table.
  - minimum redundancy, so that data are updated in only one place

The goals of normalization (2)

- All non-prime attributes in a table are dependent on the primary key
  - data are uniquely identifiable by a PK value

- Each table is void of insertion/update/deletion anomalies
  - ensure the data integrity and consistency

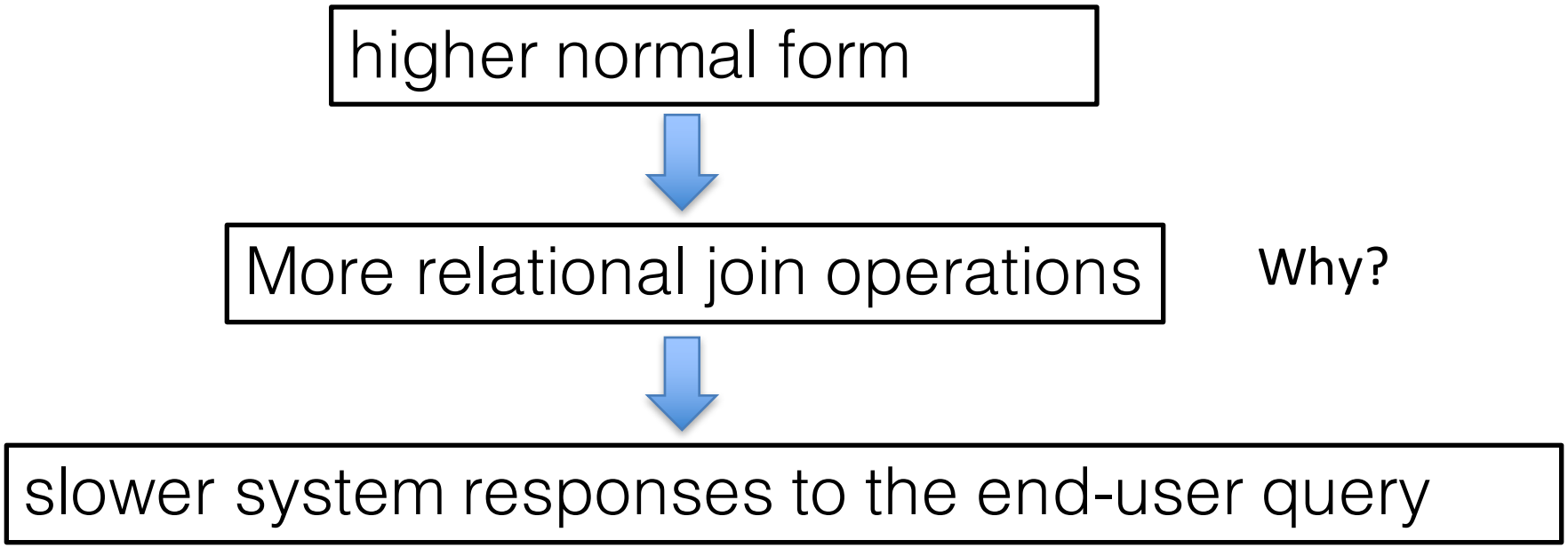

VENDOR
V_CODE
V_NAME
V_CONTACT
V_AREACODE
V_PHONE
V_STATE
V_ORDER
Indexes
EMP
EMP_NUM
EMP_TITLE
EMP_LNAME
EMP_FNAME
EMP_INITIAL
EMP_DOB
EMP_HIRE_DATE
EMP_AREACODE
EMP_PHONE
EMP_MGR
Indexes
CUSTOMER
CUS_CODE
CUS_LNAME
CUS_FNAME
CUS_INITIAL
CUS_AREACODE
CUS_PHONE
CUS_BALANCE
Indexes
PRODUCT
P_CODE
P_DESCRIPT
P_INDATE
P_QOH
P_MIN
P_PRICE
P_DISCOUNT
V_CODE
Indexes
LINE
LINE_NUMBER
LINE_UNITS
LINE_PRICE
INV_NUMBER
P_CODE
Indexes
INVOICE
INV_NUMBER
INV_DATE
CUS_CODE
Indexes

Pros of normalization

- No redundant data, storage saving.
- No duplication data, better data integrity and less risks of mistakes.
- No duplication data, less chance of data inconsistence.
- Data change can instantly be cascaded across any related records.

Cons of normalization

- End up with more tables than un-normalized table. Thus, some queries (join operation) will be “slower”. (However, it’s arguable!)



Note

- Breaking down M:N relationship is different than normalization.
- Breaking down M:N relationship is what you should consider for two entities in relational modeling.
- Normalization is applied on one table.
- **Normalization + ERD = Good database design**

Note

- There are higher level of normalization, 4NF, 5NF, BCNF.
- 3NF will perform suitably in business database.
- Highest level of normalization is not always the most desirable.
- In some cases, de-normalization is applied to the tables.
- Data model in NoSQL databases are usually de-normalized.

Summary

- We learn database normalization.

- Learning objective:
  - Why normalization
  - How to convert an un-normalized table into 3NF