CSIT115 Data Management and Security

Database Design Quality

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Database Design Quality

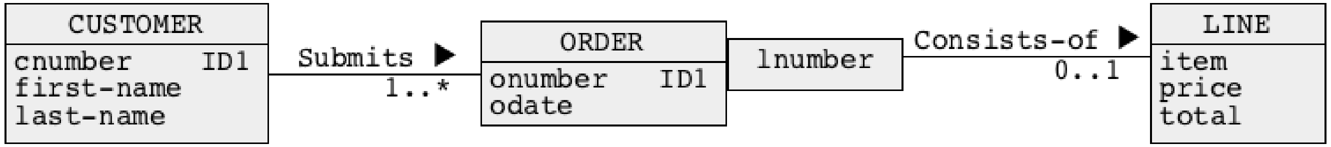
Outline

- Why not ONE BIG TABLE !?
- Where is a problem ?
- Insertion test
- Join test
- Deletion test
- Update test
- Good design guidelines

Why not ONE BIG TABLE !?

Let us consider the following database domain:

- A customer is described by a unique customer number, first, and last name
- Customers submit orders. An order is described by a unique order number and order date
- Orders consist of lines. A line contains information about a name of ordered item, price per single item, and total number of ordered items



Why not ONE BIG TABLE !?

Logical design provides the following relational schemas:

Relational schemas

```
CUSTOMER(cnumber, first-name, last-name)
PRIMARY KEY = (cnumber)

ORDERS(onumber, odate, cnumber) PRIMARY KEY = (onumber)
FOREIGN KEY = (cnaumber) REFERENCES CUSTOMER(cnumber)

LINE(onumber, lnumber, item, price total)
PRIMARY KEY = (onumber, lnumber)
FOREIGN KEY = (onumber) REFERENCES ORDERS(onumber)
```

Why not one relational schema ?

Big relational schema

```
CUSTOMER(cnumber, first-name, last-name, onumber, odate, cnumber,
         onumber, lnumber, item, price total)
PRIMARY KEY = (cnumber, onumber, lnumber)
```

Why not ONE BIG TABLE !?

Insertion of information about one customer who submitted 2 orders such that each order consists several lines reveals a problem !

Big relational table

```
+---------+-------+-------+---------+-------------+---------+-------+-------+-------+
| cnumber | fname | lname | onumber | odate       | lnumber | item  | price | total |
+---------+-------+-------+---------+-------------+---------+-------+-------+-------+
|    +----------------------+   +---------------+ |         |       |       |       |
|    |  7 | James | Bond  | |   | 7 | 2017-01-01| |       1 | bolt  | 23.04 |     5 |
|    |  7 | James | Bond  | |   | 7 | 2017-01-01| |       2 | screw | 29.01 |     3 |
|    |  7 | James | Bond  | |   | 7 | 2017-01-01| |       3 | nut   |  4.55 |     2 |
|    |    |       |       | |   +---------------+ |         |       |       |       |
|    |  7 | James | Bond  | |   | 8 | 2018-01-01| |       1 | bolt  | 23.04 |     1 |
|    |  7 | James | Bond  | |   | 8 | 2018-01-01| |       2 | screw | 23.04 |     1 |
|    |  7 | James | Bond  | |   | 8 | 2018-01-01| |       3 | nut   | 23.04 |     2 |
|    |  7 | James | Bond  | |   | 8 | 2018-01-01| |       4 | lock  | 23.04 |     1 |
|    +----------------------+   +---------------+ |         |       |       |       |
+---------+-------+-------+---------+-------------+---------+-------+-------+-------+
```

A number, first name, and last name of a customer is repeated as many times as the total number of different items purchased in all orders and order number is repeated together with order date as many times as the total number of different items purchased in an order

Why not ONE BIG TABLE !?

A multitable design does not have such a problem:

CUSTOMER schema

```
CUSTOMER(cnumber, first-name, last-name)
PRIMARY KEY = (cnumber)
```

CUSTOMER table

```
+---------+-------+-------+
| cnumber | fname | lname |
+---------+-------+-------+
|       7 | James | Bond  |
+---------+-------+-------+
```

ORDERS schema

```
ORDERS(onumber, odate, cnumber) PRIMARY KEY = (onumber)
FOREIGN KEY = (cnaumber) REFERENCES CUSTOMER(cnumber)
```

ORDERS table

```
+---------+------------+---------+
| onumber | odate      | cnumber |
+---------+------------+---------+
|       7 | 2017-01-01 |       7 |
|       8 | 2018-01-01 |       7 |
+---------+------------+---------+
```

Why not ONE BIG TABLE !?

A multitable design does not have such a problem:

LINE schema

```
LINE(onumber, lnumber, item, price total)
PRIMARY KEY = (onumber, lnumber)
FOREIGN KEY = (onumber) REFERENCES ORDERS(onumber)
```

LINE table

```
+---------+---------+-------+-------+-------+
| onumber | lnumber | item  | price | total |
+---------+---------+-------+-------+-------+
|       7 |       1 | bolt  | 23.04 |     5 |
|       7 |       2 | screw | 29.01 |     3 |
|       7 |       3 | nut   |  4.55 |     2 |
|       8 |       1 | bolt  | 23.04 |     1 |
|       8 |       2 | screw | 23.04 |     1 |
|       8 |       3 | nut   | 23.04 |     2 |
|       8 |       4 | lock  | 23.04 |     1 |
+---------+---------+-------+-------+-------+
```

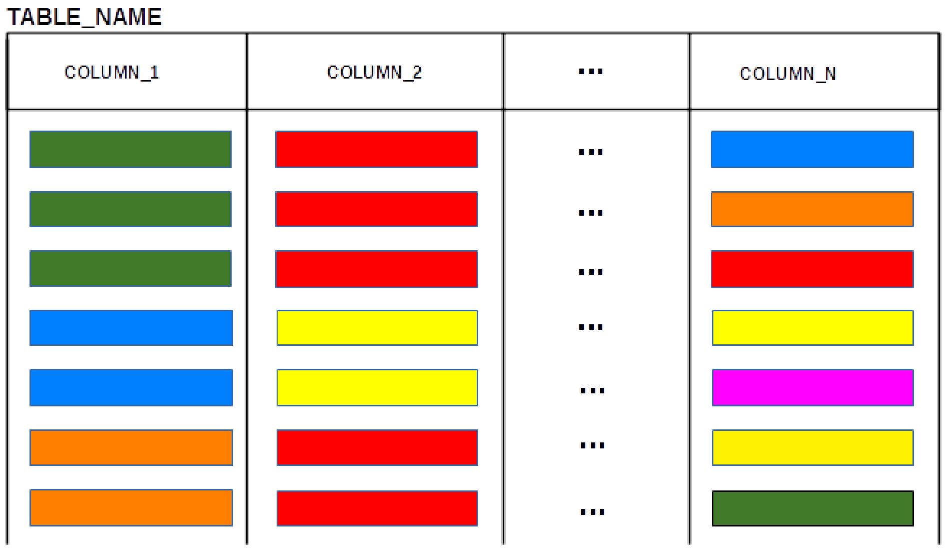
Database Design Quality

Outline

- Why not ONE BIG TABLE !?
- Where is a problem ?
- Insertion test
- Join test
- Deletion test
- Update test
- Good design guidelines

Where is a problem ?

Why do we get redundancies in an incorrectly designed relational table ?



TABLE_NAME

COLUMN_1	COLUMN_2	...	COLUMN_N
🟩	🟥	...	🟦
🟩	🟥	...	🟧
🟩	🟥	...	🟥
🟦	🟨	...	🟨
🟦	🟨	...	🟪
🟧	🟥	...	🟨
🟧	🟥	...	🟩

Data dependencies:

- If `COLUMN_1` is green then `COLUMN_2` is red
- If `COLUMN_1` is blue then `COLUMN_2` is yellow
- If `COLUMN_1` is orange then `COLUMN_2` is red
- For any colour x if `COLUMN_1` is x then `COLUMN_2 is` y

Where is a problem ?

Data dependencies can be represented as a separate relational table ...

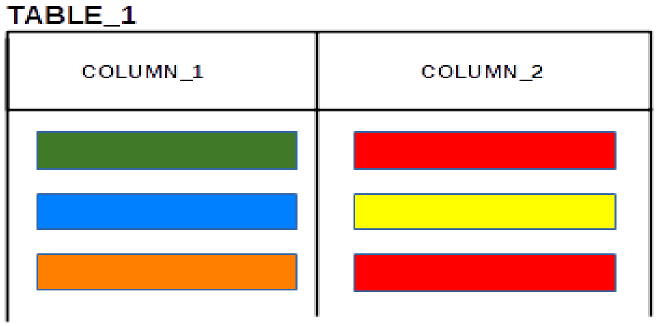


TABLE_1

COLUMN_1	COLUMN_2
🟩	🟥
🟦	🟨
🟧	🟥

... and `COLUMN_2` can be removed from the original table

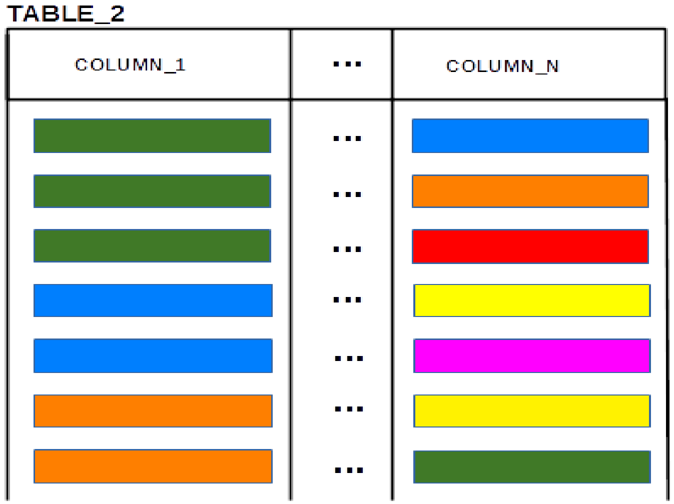


TABLE_2

COLUMN_1	...	COLUMN_N
🟩	...	🟦
🟩	...	🟧
🟩	...	🟥
🟦	...	🟨
🟦	...	🟪
🟧	...	🟨
🟧	...	🟩

Where is a problem ?

Do data dependencies exist in BIG TABLE ?

Big relational table

```
+---------+-------+-------+---------+------------+---------+-------+-------+-------+
| cnumber | fname | lname | onumber | odate      | lnumber | item  | price | total |
+---------+-------+-------+---------+------------+---------+-------+-------+-------+
|    +----------------------+    +---------------+ |        |       |       |       |
|    |  7 | James | Bond  | |    | 7 | 2017-01-01| |      1 | bolt  | 23.04 |     5 |
|    |  7 | James | Bond  | |    | 7 | 2017-01-01| |      2 | screw | 29.01 |     3 |
|    |  7 | James | Bond  | |    | 7 | 2017-01-01| |      3 | nut   |  4.55 |     2 |
|    |    |       |       | |    +---------------+ |        |       |       |       |
|    |  7 | James | Bond  | |    | 8 | 2018-01-01| |      1 | bolt  | 23.04 |     1 |
|    |  7 | James | Bond  | |    | 8 | 2018-01-01| |      2 | screw | 23.04 |     1 |
|    |  7 | James | Bond  | |    | 8 | 2018-01-01| |      3 | nut   | 23.04 |     2 |
|    |  7 | James | Bond  | |    | 8 | 2018-01-01| |      4 | lock  | 23.04 |     1 |
|    +----------------------+    +---------------+ |        |       |       |       |
+---------+-------+-------+---------+------------+---------+-------+-------+-------+
```

Data dependencies:

- If `cnumber` = 7 then `fname` = James
- If `cnumber` = 7 then `lname` = Bond
- For any customer number x if `cnumber` = x then `fname` = y and `lname` = z

Where is a problem ?

Do data dependencies exist in BIG TABLE ?

Big relational table

```
+---------+-------+-------+---------+-------------+---------+-------+-------+-------+
| cnumber | fname | lname | onumber | odate       | lnumber | item  | price | total |
+---------+-------+-------+---------+-------------+---------+-------+-------+-------+
|    +----------------------+   +---------------+ |         |       |       |       |
|    |  7 | James | Bond  | |   | 7 | 2017-01-01| |       1 | bolt  | 23.04 |     5 |
|    |  7 | James | Bond  | |   | 7 | 2017-01-01| |       2 | screw | 29.01 |     3 |
|    |  7 | James | Bond  | |   | 7 | 2017-01-01| |       3 | nut   |  4.55 |     2 |
|    |    |       |       | |   +---------------+ |         |       |       |       |
|    |  7 | James | Bond  | |   | 8 | 2018-01-01| |       1 | bolt  | 23.04 |     1 |
|    |  7 | James | Bond  | |   | 8 | 2018-01-01| |       2 | screw | 23.04 |     1 |
|    |  7 | James | Bond  | |   | 8 | 2018-01-01| |       3 | nut   | 23.04 |     2 |
|    |  7 | James | Bond  | |   | 8 | 2018-01-01| |       4 | lock  | 23.04 |     1 |
|    +----------------------+   +---------------+ |         |       |       |       |
+---------+-------+-------+---------+-------------+---------+-------+-------+-------+
```

More data dependencies:

- If `onumber` = 7 then `odate` = 2017-01-01
- If `onumber` = 8 then `odate` = 2018-01-01
- For any order number x if `onumber` = x then `odate` = y

Database Design Quality

Outline

- Why not ONE BIG TABLE !?
- Where is a problem ?
- Insertion test
- Join test
- Deletion test
- Update test
- Good design guidelines

Insertion test

How to verify if a relational schema is designed in a correct way ?

We try to insert few rows such that it is possible to create redundancies

For example, we insert few rows into a relational table ROOM

ROOM schema

```
ROOM(bldgnum, name, roomnum, area)
PRIMARY KEY = (bldgnum, roomnum)
```

ROOM table

```
+---------+----------+---------+------+
| bldgnum | bldgname | roomnum | area |
+---------+----------+---------+------+
|       3 | SCIT     |     210 |   20 |
+---------+----------+---------+------+
```

ROOM table

```
+---------+----------+---------+------+
| bldgnum | bldgname | roomnum | area |
+---------+----------+---------+------+
|       3 | SCIT     |     210 |   20 |
|       3 | SCIT     |     211 |   22 |
+---------+----------+---------+------+
```

ROOM table

```
+---------+----------+---------+------+
| bldgnum | bldgname | roomnum | area |
+---------+----------+---------+------+
|       3 | SCIT     |     210 |   20 |
|       3 | SCIT     |     211 |   22 |
|       3 | SCIT     |     213 |   20 |
+---------+----------+---------+------+
```

Insertion test

ROOM table

```
+---------+----------+---------+------+
| bldgnum | bldgname | roomnum | area |
+---------+----------+---------+------+
|       3 | SCIT     |     210 |   20 |
|       3 | SCIT     |     211 |   22 |
|       3 | SCIT     |     213 |   20 |
+---------+----------+---------+------+
```

Problems:

- It clearly visible that building name is repeated as many times as many rooms are included in a building

Insertion test

In another example, we insert few rows into a relational table WAREHOUSE

WAREHOUSE schema

```
WAREHOUSE(name, address, part, quantity)
PRIMARY KEY = (name, part)
```

WAREHOUSE table

```
+--------------+-----------------+-------+----------+
| name         | address         | part  | quantity |
+--------------+-----------------+-------+----------+
| Golden Bolts | Northfields Ave | bolt  |      210 |
+--------------+-----------------+-------+----------+
```

WAREHOUSE table

```
+--------------+-----------------+-------+----------+
| name         | address         | part  | quantity |
+--------------+-----------------+-------+----------+
| Golden Bolts | Northfields Ave | bolt  |      210 |
| Golden Bolts | Northfields Ave | lock  |       20 |
+--------------+-----------------+-------+----------+
```

WAREHOUSE table

```
+--------------+-----------------+-------+----------+
| name         | address         | part  | quantity |
+--------------+-----------------+-------+----------+
| Golden Bolts | Northfields Ave | bolt  |      210 |
| Golden Bolts | Northfields Ave | lock  |       20 |
| Golden Bolts | Northfileds Ave | screw |      211 |
+--------------+-----------------+-------+----------+
```

Insertion test

WAREHOUSE table

```
+--------------+----------------+-------+----------+
| name         | address        | part  | quantity |
+--------------+----------------+-------+----------+
| Golden Bolts | Northfields Ave | bolt  |      210 |
| Golden Bolts | Northfields Ave | lock  |       20 |
| Golden Bolts | Northfileds Ave | screw |      211 |
+--------------+----------------+-------+----------+
```

Problems:

- Address of a warehouse is repeated as many time as many different parts are stored in the warehouse
- If at some point in time there are no parts stored in a warehouse then there may be no rows to keep a warehouse address or the values of certain attributes must be set to NULL

Insertion test

In yet another example, we insert few rows into a relational table EMPLOYEE

EMPLOYEE schema

```
EMPLOYEE(enum, skill, hobby)
PRIMARY KEY = (enum, skill, hobby)
```

EMPLOYEE table

```
+------+------------+----------+
| enum | skill      | hobby    |
+------+------------+----------+
|    7 | cooking    | hiking   |
+------+------------+----------+
```

EMPLOYEE table

```
+------+------------+----------+
| enum | skill      | hobby    |
+------+------------+----------+
|    7 | cooking    | hiking   |
|    7 | cooking    | swimming |
+------+------------+----------+
```

EMPLOYEE table

```
+------+-------------+----------+
| enum | skill       | hobby    |
+------+-------------+----------+
|    7 | cooking     | hiking   |
|    7 | cooking     | swimming |
|    7 | programming | hiking   |
|    7 | programming | swimming |
+------+-------------+----------+
```

Insertion test

EMPLOYEE table

```
+------+-------------+----------+
| enum | skill       | hobby    |
+------+-------------+----------+
|    7 | cooking     | hiking   |
|    7 | cooking     | swimming |
|    7 | programming | hiking   |
|    7 | programming | swimming |
+------+-------------+----------+
```

Problems:

- Skill name must be repeated with each hobby name
- Hobby name must be repeated with each skill name
- If at some point in time an employee has no hobbies (or skills) then a value of attribute `hobby` (or `skill`) must be set to `NULL`, however, it is impossible due to `PRIMARY KEY = (enum, skill, hobby)` constraint

Database Design Quality

Outline

- Why not ONE BIG TABLE !?
- Where is a problem ?
- Insertion test
- Join test
- Deletion test
- Update test
- Good design guidelines

Join test

We consider the relational tables with the following schemas:

SUPPLIER scheme

```
SUPPLIER(sname, city)
PRIMARY KEY = (name, city)
```

COMPANY schema

```
COMPANY(cname, city)
PRIMARY KEY = (cname, city)
```

SUPPLIER table

```
+-------+-------+
| sname | city  |
+-------+-------+
| Harry | Paris |
| James | Paris |
| Robin | Rome  |
+-------+-------+
```

COMPANY table

```
+--------------+-------+
| cname        | city  |
+--------------+-------+
| Golden Bolts | Paris |
| Golden Bolts | Rome  |
| Lazy Lobster | Rome  |
+--------------+-------+
```

Join test

The result of join of SUPPLIER and COMPANY tables over a column city:

JOIN of SUPLIER and COMPANY tables

```
+-------+-------+--------------+-------+
| sname | city  | cname        | city  |
+-------+-------+--------------+-------+
| Harry | Paris | Golden Bolts | Paris |
| James | Paris | Golden Bolts | Paris |
| Robin | Rome  | Golden Bolts | Rome  |
| Robin | Rome  | Lazy Lobster | Rome  |
+-------+-------+--------------+-------+
```

Problems:

- Join of relational tables SUPPLIER and COMPANY creates the spurious row [Robin | Rome | Lazy Lobster | Rome] that represent wrong information

Database Design Quality

Outline

- Why not ONE BIG TABLE !?
- Where is a problem ?
- Insertion test
- Join test
- Deletion test
- Update test
- Good design guidelines

Deletion test

We consider a relational table with the following schema:

SHIPMENT schema

```
SHIPMENT(sname, product, part)
PRIMARY KEY = (sname, product, part)
```

- The table contains the following rows:

SHIPMENT table

```
+-------+--------------+-----------+
| sname | product      | part      |
+-------+--------------+-----------+
| James | audio system | amplifier |
| James | audio system | speakers  |
| James | computer     | hard disk |
| James | computer     | mainboard |
| James | computer     | processor |
+-------+--------------+-----------+
```

- Deletion of a product `computer` requires deletion of additional two rows
- Then, deletion of a product `audio system` causes accidental deletion of information about supplier `James`

Problems:

- Deletion of a row triggers deletion of the other rows in the same table
- Deletion of a row accidentally deletes other information

Database Design Quality

Outline

- Why not ONE BIG TABLE !?
- Where is a problem ?
- Insertion test
- Join test
- Deletion test
- Update test
- Good design guidelines

Update test

We consider a relational table with the following schema:

WAREHOUSE schema

```
WAREHOUSE(name, address, part, quantity)
PRIMARY KEY = (name, part)
```

- The table contains the following rows:

WAREHOUSE table

```
+--------------+-----------------+-------+----------+
| name         | address         | part  | quantity |
+--------------+-----------------+-------+----------+
| Golden Bolts | Northfields Ave | bolt  |      210 |
| Golden Bolts | Northfields Ave | lock  |       20 |
| Golden Bolts | Northfileds Ave | screw |      211 |
+--------------+-----------------+-------+----------+
```

- Modification of an address `Northfields Ave` requires replication of a modification in two other rows

Problem:

- Modification of a row triggers modifications of the other rows in the same table

Database Design Quality

Outline

- Why not ONE BIG TABLE !?
- Where is a problem ?
- Insertion test
- Join test
- Deletion test
- Update test
- Good design guidelines

Good design guidelines

Design a relational table such that it is easy to explain its meaning

Do not combine unrelated attributes into the same table

Design a relational table such that insertion, deletion and update tests do not cause problems

Minimize the number of attributes whose values can be missing (`NULL`)

Design the relational tables such that they can be joined with equality conditions on attributes that are either primary or foreign keys in a way that no spurious rows are generated

And the first of all ALWAYS START FROM CONCEPTUAL MODELING !

References

T. Connoly, C. Begg, Database Systems, A Practical Approach to Design, Implementation, and Management, Chapters 14.1 - 14.3 Introduction to normalization, Pearson Education Ltd, 2015