Technical University of Denmark

DTU Compute
Department of Applied Mathematics and Computer Science

Database project specification
Presentation of the mandatory group project

©Giovanni Meroni

These slides have been prepared by Giovanni Meroni, partly reusing/modifying slides by Anne Haxthausen and Thorbjørn Konstantinovitz.

Practical Information

- Mandatory
  - It is mandatory to hand in a mandatory group project and get it approved in order to participate in the final written examination!
  - The final mark for this course is the mark from the written examination.
- Groups
  - You **must work in the registered groups on DTU Learn.**
  - The members in a group must contribute equally to the project. Only persons who have contributed are allowed to appear as report authors.
  - During exercise sessions, a TA and a room will be assigned to each group, according to the following scheme:
    (1) Group 1-6 – Esben – Room 306-97
    (2) Group 7-13 – Victor – Room 306-108a
    (3) Group 14-19 – Ali – Room 306-99
    (4) Group 20-26 – Berfin – Room 306-108b
  - Remember to state your group number when interacting with TAs

Practical Information

- Hand-In: What, Where and When
  - Each group should upload the following files to the Assignment on DTU Learn, not later than **Friday April 5th at 17:00**:
    1. The **group project report** in a .pdf file named *id*_02327GroupReport2024.pdf, where *id* is your group number. It must have the sections explained on one of the following pages.
    2. An **SQL script** in a .sql file named *id*_02327DatabaseScript1_2024.sql , where *id* is your group number. It must contain
       (1) the statements used to create the database, its tables and views (as used in section 3 of the report)
       (2) the statements used to populate the tables (as used in section 4)
    3. An **SQL script** in a .sql file named *id*_02327DatabaseScript2_2024.sql , where *id* is your group number. It must contain
       (1) the delete/update statements used to change the tables (as in section 5), and
       (2) the queries made (as in section 6), and
       (3) the statements used to create and apply functions, procedures, triggers, and events (as in section 7)
    4. A **ZIP archive** in a .zip file named *id*_02327Application_2024.zip , where *id* is your group number. It must contain
       (1) The Java source code of your application.
       (2) The Excel spreadsheet 'Effort.xlsx'

    It is a requirement that there are no run-time errors when running the scripts under MariaDB.
- The Results of the Group Project Evaluation
  - Will be communicated via DTU Learn.

Task, Objectives, and Scope

- Task:
  - is to develop and document a database of your own choice.
- Objective:
  - is to get practical experience with data modelling and database design.
- Scope:
  - SQL programming of the database is requested, minimal application logic and no user interface. MariaDB and Java/JDBC must be used for the implementation.

Task, Objectives, and Scope

- In the database project, all names, data and case information are fiction, any resemblance to reality is not intentional.
- The goal is hands-on experience with SQL database design and Java database access
- Only SQL and Java database access, minimal application logic and no user interface
- It is not allowed to use the autoincrement function in MariaDB to form primary keys, unless you can argue that there are no other possible candidate keys.

Introduction and background

Introduction

You have been asked by the DKAvisen publisher to create a database to keep track of all the news articles being published in its newspapers, the journalists who wrote them, and the photos being shot.

In addition, DKAvisen wants to use this database to determine the performance of its team. In particular, DKAvisen wants to know how many readers newspapers, issues and articles were able to attract.

Background

DKAvisen is a well-established publisher of newspapers, which is facing a loss of revenues due to the decline of the newspaper industry.

To address this issue, DKAvisen aims at better monitoring its performance. In particular, DKAvisen needs to know:

- For each topic, the most read news article.
- The top 10 journalists whose articles, overall, attracted the highest number of reads.
- Which are the reporters whose photos were never used more than once.
- Which topics, overall, attracted less reads that the average.
- Which journalists were both writers and reporters, having shot at least a photo that was used for a news article they wrote.

Requirements specification

DKAvisen has described a number of requirements for the database:

- First, it must be possible to register newspapers. Each newspaper is characterized by the title, the founding date, and the periodicity (i.e., after how many days a new edition is published).

- For each newspaper, multiple editions exist. An edition is characterized by the date when it was published. It also contains a reference to the journalist who was the editor.

- Each edition contains one or more news articles. An article is characterized by the title, the text, a topic, how many times it was read, and a reference to the journalists who wrote it. For each journalist, the role they had when writing the article (e.g., leader, advisor, etc.) must be present as well.

- News articles can also contain one or more photos. A photo is characterized by a title, the date when it was shot, and a reference to the journalist who was the reporter. The same photo can be used for multiple news articles.

- Journalists are characterized by their CPR number, first name, last name, address (consisting of street name, civic number, city, ZIP code and country), telephone numbers and email addresses.

Mandatory Report Requirements

Mandatory Sections	Tasks and Contents of Mandatory Sections
Title Page	Course Name & No, Group No, Project Title, Student Names and Study Numbers, Date.
1. Conceptual Design	Show an Entity-Relationship Diagram for the domain of your database using the Textbook Adapted UML Notation. Explain. Discuss choices made.
2. Logical Design	Convert your conceptual design into a logical design (relation schemas) and discuss any choices made. Show a database schema diagram in the Textbook notation.
3. Implementation	Create a MariaDB database with tables and views implementing the logical design.
4. Database Instance	Populate the tables with data, and list data for all tables and views.
5. SQL Table Modifications	Give examples of typical SQL table update and delete statements. Show the results of the statements.
6. SQL Data Queries	Write SQL table queries to answer the 5 questions introduced in the Background section. For each query explain informally what it asks about. Show also the output of the queries.
7. SQL Programming	Give examples of functions, procedures, and triggers, and explain what they do. Show illustrative usage examples.
8. Java Database Access	Describe how you implemented a Java program to load data into the database from file.

Title Page and Report Format

- Title Page
  - Make the Title Page inviting and interesting
    - It gives the reader a first good impression
  - Include
    - Course name & number
    - Project title
    - Group number
    - Student names, study numbers
    - Date

- Report Format
  - Include page numbers.
  - Include a table of contents.
  - Include pictures and drawings to clarify text.
  - Use readable fonts for various text elements and picture captions.
  - Include an appendix for additional material, if needed.
  - Include a bibliography, if needed.

University DB as an example

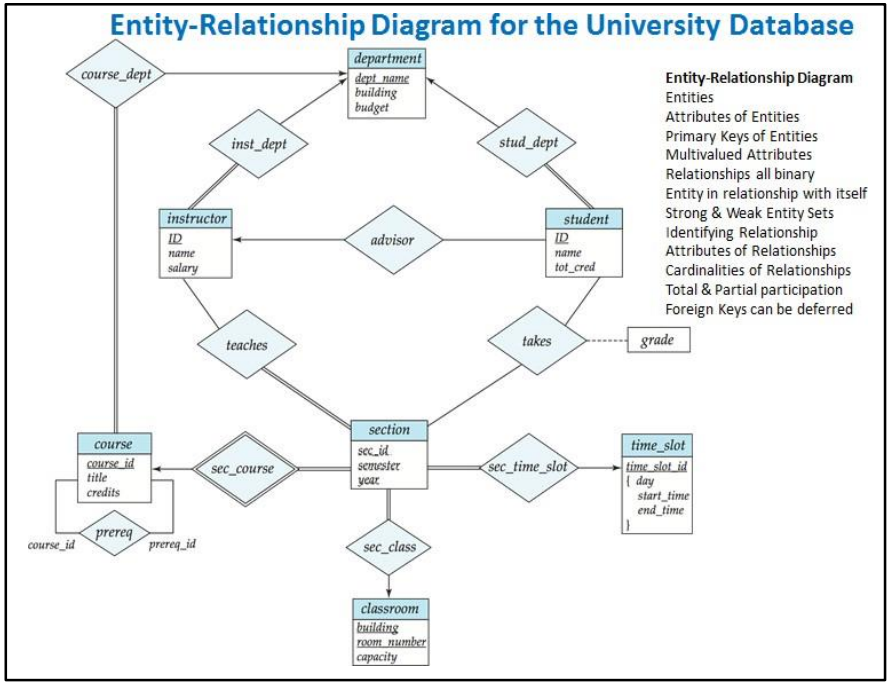
As an example of a database project, the following is demonstrated how the textbook's University database could be developed.

It is for inspiration and with the use of Danish database terminology.

Remember that development is an iterative process in which the result of previous phases is often modified when later phases have been completed and have provided greater insight!

1. Conceptual Design

- Show an Entity-Relationship Diagram for the domain of your database using the Textbook Adapted UML Notation.
  - Use the Textbook Adapted UML Notation and follow the strict rules to show (1) strong & weak entity sets with their names, attributes and primary keys, and (2) relationship sets with their names, attributes, primary keys, cardinalities, and total/partial participation.
  - Example:



- Explain. Explain the meaning of entities, relationships, and their attributes.
- Discuss any choices made. E.g. why the cardinalities and participation constraints are chosen as they are etc.

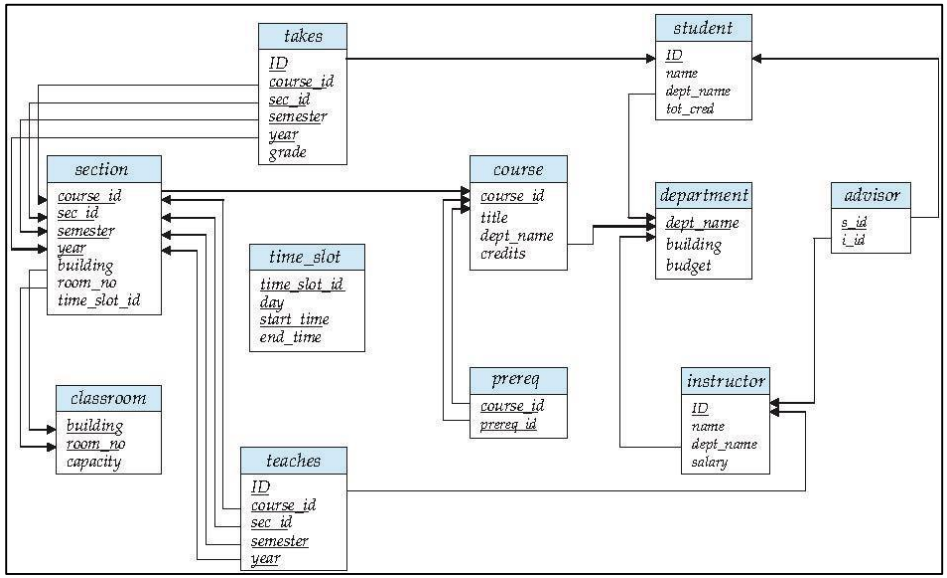
2. Logical Design

- Convert your conceptual design into a logical design (relation schemas inclusive specification of foreign keys) and discuss any choices made. You must follow the method described in the course slides and book (Chapter 7).
  - Example: Conversion of the diagram shown on previous slides gives

    Instructor(InstID, InstName, DeptName, Salary) **foreign key** (DeptName) **references** Department(DeptName)
    Department(DeptName, Building, Budget)
    ….

- Show a database schema diagram for the relation schemas of the logical design in the Textbook notation. Example of a database schema diagram for the University DB:

Textbook notation:

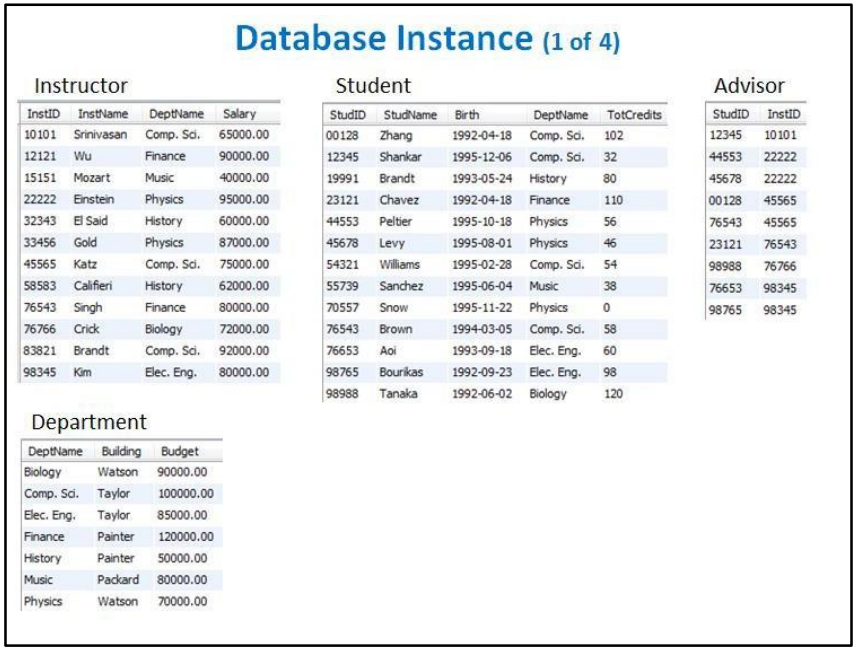


3. Implementation

- Create a MariaDB database with tables and views implementing the logical design (as achieved after possible revisions in step 4):
  - Use SQL statements CREATE DATABASE, CREATE TABLE and CREATE VIEW.
  - Show the statements in the report.

4. Database Instance

- Populate the tables with data, and list data for all tables and views.
  1. Use SQL **INSERT** to populate the tables.
  2. Use SQL **SELECT * FROM** *table* to list instances of all tables and views.
  3. Show the output of step 2 in the report.
- Example of output, from one of the course slides:



Database Instance (1 of 4)

Instructor

InstID	InstName	DeptName	Salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

Student

StudID	StudName	Birth	DeptName	TotCredits
00128	Zhang	1992-04-18	Comp. Sci.	102
12345	Shankar	1995-12-06	Comp. Sci.	32
19991	Brandt	1993-05-24	History	80
23121	Chavez	1992-04-18	Finance	110
44553	Peltier	1995-10-18	Physics	56
45678	Levy	1995-08-01	Physics	46
54321	Williams	1995-02-28	Comp. Sci.	54
55739	Sanchez	1995-06-04	Music	38
70557	Snow	1995-11-22	Physics	0
76543	Brown	1994-03-05	Comp. Sci.	58
76653	Aoi	1993-09-18	Elec. Eng.	60
98765	Bourikas	1992-09-23	Elec. Eng.	98
98988	Tanaka	1992-06-02	Biology	120

Advisor

StudID	InstID
12345	10101
44553	22222
45678	22222
00128	45565
76543	45565
23121	76543
98988	76766
76653	98345
98765	98345

Department

DeptName	Building	Budget
Biology	Watson	90000.00
Comp. Sci.	Taylor	100000.00
Elec. Eng.	Taylor	85000.00
Finance	Painter	120000.00
History	Painter	50000.00
Music	Packard	80000.00
Physics	Watson	70000.00

5. SQL Table Modifications

- Give examples of a SQL table update statement and a delete statement.
  - Show with illustrative examples how you do table modifications using the SQL commands INSERT, UPDATE and DELETE.

INSERT

- Add a new row to Course

```
INSERT Course VALUES ('CS-437', 'Database Systems', 'Comp. Sci.', 4);
```

- Or equivalently

```
INSERT COURSE (CourseID, Title, DeptName, Credits)
VALUES ('CS-437', 'Database Systems', 'Comp. Sci.', 4);
```

- Add a new row in Student with TotCredits set to NULL

```
INSERT Student VALUES ('3003', 'Green','1993-04-16', 'Finance', NULL);
```

- Adding multiple rows

```
INSERT Course Values
('CS-437', 'Database Systems', 'Comp. Sci.', 4),
('CS-528', 'Big Data Systems', 'Comp. Sci.', 5),
('CS-530', 'Data Warehouse', 'Comp. Sci.', 4);
```

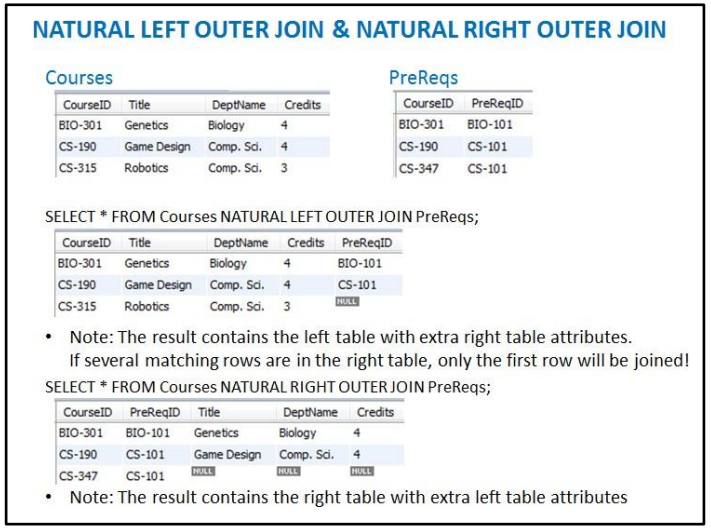
Example from the Introductory SQL 1 lesson

6. SQL Data Queries

Answer the following questions by writing appropriate SQL table queries:

- For each topic, show the most read news article.
- Show the top 10 journalists whose articles, overall, attracted the highest number of reads.
- Show reporters whose photos were never used more than once.
- Identify which topics, overall, attracted less reads that the average.
- Identify which journalists were both writers and reporters, having shot at least a photo that was used for a news article they wrote.

Example from the Intermediate SQL lesson



NATURAL LEFT OUTER JOIN & NATURAL RIGHT OUTER JOIN

Courses

CourseID	Title	DeptName	Credits
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

PreReqs

CourseID	PreReqID
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

SELECT * FROM Courses NATURAL LEFT OUTER JOIN PreReqs;

CourseID	Title	DeptName	Credits	PreReqID
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	NULL

- Note: The result contains the left table with extra right table attributes. If several matching rows are in the right table, only the first row will be joined!

SELECT * FROM Courses NATURAL RIGHT OUTER JOIN PreReqs;

CourseID	PreReqID	Title	DeptName	Credits
BIO-301	BIO-101	Genetics	Biology	4
CS-190	CS-101	Game Design	Comp. Sci.	4
CS-347	CS-101	NULL	NULL	NULL

- Note: The result contains the right table with extra left table attributes

7. SQL Programming

- Give examples of functions, procedures, and triggers, and explain what they do. Give one example of each.
  - Remember also to show illustrative usage examples of how they work.
- Example:

Functions - Example

- Create a function and test it
  - Given a department name, the function returns the number of instructors

```
DELIMITER //
CREATE FUNCTION DeptInstCount (vDeptName VARCHAR(20)) RETURNS INT
BEGIN
  DECLARE vDeptInstCount INT;
  SELECT COUNT(*) INTO vDeptInstCount FROM Instructor
  WHERE DeptName = vDeptName;
  RETURN vDeptInstCount;
END; //
DELIMITER ;
```

DeptName	Instructors
Biology	1
Comp. Sci.	3
Elec. Eng.	1
Finance	2
History	2
Music	1
Physics	2

The SQL DELIMITER is temporarily changed from ";" to "//" in order to allow ";" in the function body!

  - `SELECT DeptName, DeptInstCount(DeptName) AS Instructors FROM Department;`
  - Find department name and budget of all departments with two or more instructors.

```
SELECT DeptName, Budget FROM Department
WHERE DeptInstCount(DeptName) >= 2;
```

DeptName	Budget
Comp. Sci.	100000.00
Finance	120000.00
History	50000.00
Physics	70000.00

5. Advanced SQL — Functions — 12

8. Java Database Access

(change) The purpose of developing a java program for data management is to form a foundation for later programming of workflows and user interfaces in DKAvisen content management system.

In the assignment, you must show that from a java program you can load data into your database that comes from a data file received daily from DKAvisen media server.

With this task comes an example of a data file (uploads.csv). The data file contains information about photos, as well as the reporters who shot them. You must develop a program that registers information about new photos in the database, as well as new reporters.

The data file contains several lines. Each line contains information on a photo and the reporter who shot it. A line contains several attributes. Attribute values are separated by a special character (;). Such a file format is called a comma separated file (a .csv file). Along with the assignment, an example of a data file (uploads.csv) is provided. You don't need to implement the parsing of this file format from scratch. There are java classes that do this. The Java classes are presented on the next slide. The attributes of a line in the data file are:

Title, Date, CPR, First Name; Last Name; Street Name; Civic Number; ZIP code; Country

An example of a data file can be seen below:

```
1  Royal Wedding;20040514;10103040;Mark;Miller;Nullvej;132;2800;Kgs. Lyngby
2  COVID-19 Restrictions;20200311;30302125;Olga;Owens;Nybrovej;28;2840;Holte
3  Fatal Car Accident;20211009;30302125;Olga;Owens;Nybrovej;28;2840;Holte
4  Pottery Exhibit;20211110;10203344;Lukas;Laas;Østergaardsvej;12;2820;Gentofte
5  Superliga Final Match;20220529;20208981;Pia;Pabst;Hovedgade;9;2850;Nærum
6  SAS Strike;20220718;30302125;Olga;Owens;Nybrovej;28;2840;Holte
7  Election Day;20221101;10204410;Nick;Nassar;Vestergaardsvej;13;2830;Virum
8  Snowstorm;20230306;10103040;Mark;Miller;Nullvej;132;2800;Kgs. Lyngby
```

You can construct test data for your program by adding/modifying the data file or by making a completely new one. It is just important that the format described above is observed (i.e., the structure of the CSV file should not be changed).

8. Java Database Access

Use the java classes provided with the task to load the data file. The java classes provided have the following purpose:

File name	Purpose
`PhotoAndReporter.java`	A class that contains information about a photo and the reporter who shot it
`PhotosAndReportersLoader.java`	A class that handles the loading of data from the CSV, interprets the data and transforms it into a list of java objects (PhotoAndReporter)
`LoaderExample.java`	An example of using PhotosAndReportersLoader.java
`Reporter.java`	A class that contains attributes for a Reporter
`Photo.java`	A class that contains attributes for a Photo

Below is an example of the LoaderExample:

1. Instantiate class PhotosAndReportersLoader

2. Load all footages and reporters in the CSV file whose name is passed as the first argument of LoaderExample by calling the method loadPhotosAndReporters. The method returns a list of PhotoAndReporter objects

3. Loop through the list of PhotoAndReporter objects and print the contents

```java
public class LoaderExample {

    no usages
    public static void main(String[] args) {
        PhotosAndReportersLoader loader = new PhotosAndReportersLoader();
        try {
            System.out.println("loading from "+args[0]);
            List<PhotoAndReporter> photosAndReporters = loader.loadPhotosAndReporters(args[0]);
            for(PhotoAndReporter photoAndReporter : photosAndReporters) {
                System.out.print("\tPhoto: " + photoAndReporter.getPhoto());
                System.out.println("\tReporter: " + photoAndReporter.getReporter());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Participants' efforts in group work

- Document the participants' efforts in the group work by filling in the attached spreadsheet 'Effort.xslx'
  - The basis for the award of individual grades is a combination of effort and results
  - Therefore, document who has prepared which sections
- Sample participant table

- If everyone has worked equally on all sections, then everyone in the group will get the same personal evaluation.
- If everyone has worked equally but on different sections, then the quality of the different sections will impact on the personal evaluation.
- PS. Remember that the global effort of all participants in a section must be equal to 100%

Group number	Member	1. Conceptual Design	2. Logical Design	3. Implementation	4. Database Instance	5. SQL Table Modifications	6. SQL Data Queries	7. SQL Programming	8. Java Database Access	AVERAGE
X	Alice Andersen	25%	25%	20%	20%	15%	20%	20%	15%	20,0%
	Bob Baron	20%	20%	20%	20%	20%	20%	20%	20%	20,0%
	Charlie Chambers	20%	20%	25%	15%	20%	20%	20%	20%	20,0%
	Diana Davidson	15%	15%	15%	25%	20%	25%	20%	25%	20,0%
	Edgar Ewans	20%	20%	20%	20%	25%	15%	20%	20%	20,0%
	TOTAL	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%