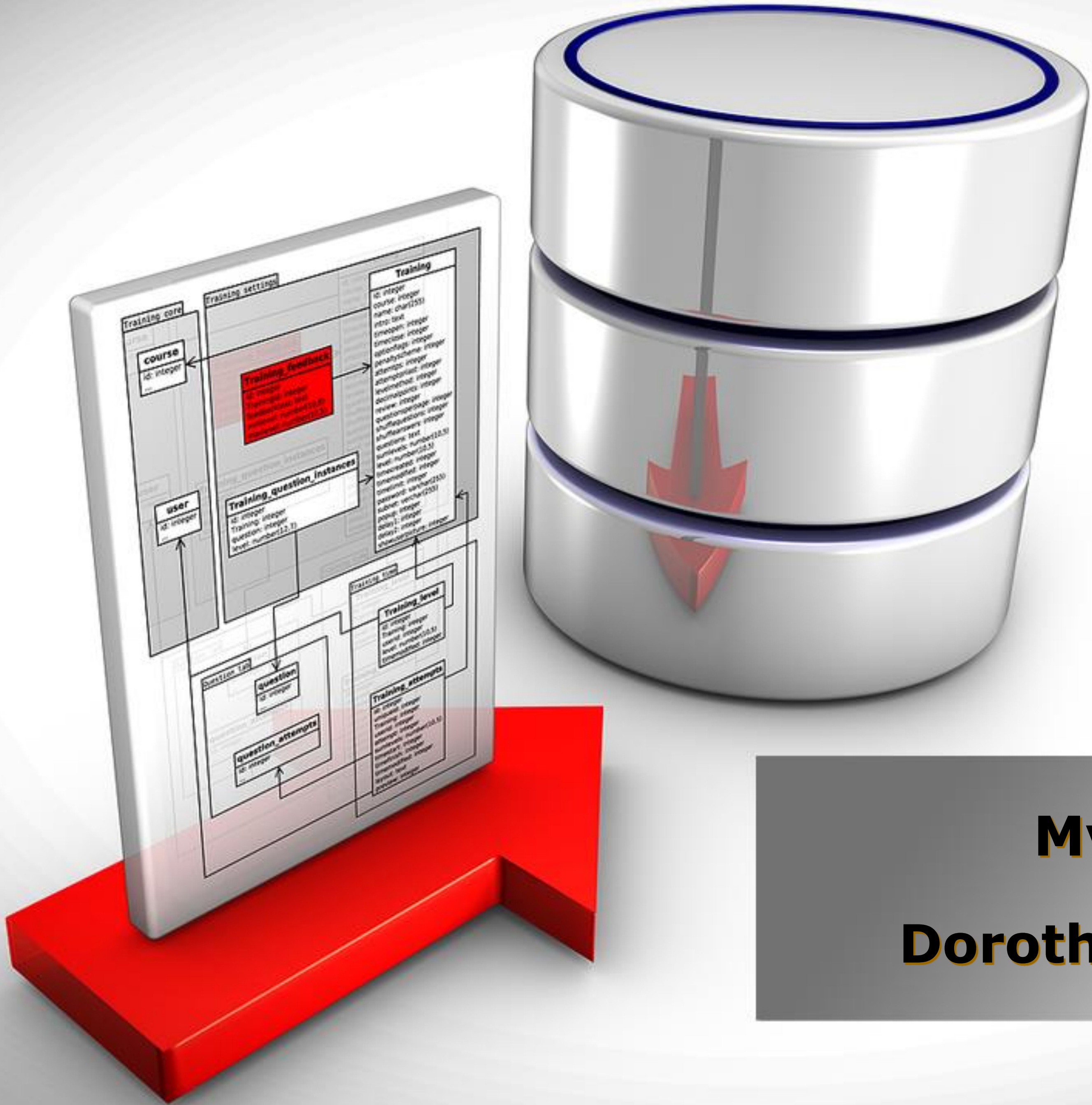

MySQL

Dorothy Guirgues

What is MySQL?

- MySQL is a relational database management system
- MySQL is open-source
- MySQL is free
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, scalable, and easy to use
- MySQL is cross-platform

What is MySQL? (Cont.)

- MySQL is compliant with the ANSI SQL standard
- MySQL was first released in 1995
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: My

Who Uses MySQL?

- Huge websites like Facebook, Twitter, Airbnb, Booking.com, Uber, GitHub, YouTube, etc.
- Content Management Systems like WordPress, Drupal, Joomla!, Contao, etc.
- A very large number of web developers around the world.

Show Data On Website

To build a web site that shows data from a database, you will need:

- An RDBMS database program (like MySQL)
- A server-side scripting language, like PHP
- To use SQL to get the data you want
- To use HTML / CSS to style the page

What is RDBMS?

- RDBMS stands for Relational Database Management System.
- RDBMS is a program used to maintain a relational database.
- RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access.
- RDBMS uses SQL queries to access the data in the database.

What is a Database Table?

- A table is a collection of related data entries, and it consists of columns and rows.
- A column holds specific information about every record in the table.
- A record (or row) is each individual entry that exists in a table.

What is SQL?

- SQL is the standard language for dealing with Relational Databases.
- SQL is used to insert, search, update, and delete database records.

How to Use SQL ?

- Example

```
SELECT * FROM Customers;
```

- Note: SQL keywords are NOT case sensitive: select is the same as SELECT

How to Use SQL ? (Cont.)

- Some database systems require a semicolon at the end of each SQL statement.
- Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

CREATE DATABASE

- ➢ Used to create a new SQL database.

❑ Syntax

CREATE DATABASE *databasename*;

❑ Example

CREATE DATABASE testDB;

DROP DATABASE

- Used to drop an existing SQL database.

❑ Syntax

```
DROP DATABASE databasename;
```

❑ Example

```
DROP DATABASE testDB;
```

CREATE TABLE

- Used to create a new table in a database.

- Syntax

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

CREATE TABLE (Cont.)

- Example

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

CREATE TABLE From Another Table

- A copy of an existing table can also be created using CREATE TABLE.

- The new table gets the same column definitions. All columns or specific columns can be selected.

- If you create a new table using an existing table, the new table will be filled with the existing values from the old table.

CREATE TABLE From Another Table (Cont.)

- Syntax

```
CREATE TABLE new_table_name AS
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
```

- Example

```
CREATE TABLE TestTable AS
SELECT customername,
contactname
FROM customers;
```

DROP TABLE

- Used to drop an existing table in a database.

- Syntax

```
DROP TABLE table_name;
```

- Example

```
DROP TABLE Shippers;
```

TRUNCATE TABLE

- Used to delete the data inside a table, but not the table itself.

- Syntax

```
TRUNCATE TABLE table_name;
```

- Example

```
TRUNCATE TABLE Persons;
```

ALTER TABLE

- Used to add, delete, or modify columns in an existing table.
- Also used to add and drop various constraints on an existing table.

ALTER TABLE - ADD Column

- To add a column in a table

- Syntax

ALTER TABLE *table_name*
ADD *column_name datatype*;

- Example

```
ALTER TABLE Customers
ADD Email varchar(255);
```

ALTER TABLE - DROP COLUMN

- To delete a column in a table.

- Syntax

ALTER TABLE *table_name*
DROP COLUMN *column_name*;

- Example

```
ALTER TABLE Customers
DROP COLUMN Email;
```

ALTER TABLE - MODIFY COLUMN

- To change the data type of a column in a table.

- Syntax

ALTER TABLE *table_name*
MODIFY COLUMN *column_name*
datatype;

- Example

```
ALTER TABLE Persons
MODIFY COLUMN DateOfBirth year;
```

MySQL Constraints

- Used to specify rules for data in a table.
- Used to limit the type of data that can go into a table
- Can be column level or table level
- Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement

MySQL Constraints (Cont.)

- Syntax

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

MySQL Constraints (Cont.)

Following constraints are commonly used:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT
- CREATE INDEX

NOT NULL Constraint

- By default, a column can hold NULL values.
- The NOT NULL constraint enforces a column to NOT accept NULL values.
- This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field

NOT NULL Constraint (Cont.)

- NOT NULL on CREATE TABLE.

- Example

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

NOT NULL Constraint (Cont.)

- NOT NULL on ALTER TABLE.

- Example

```
ALTER TABLE Persons
MODIFY Age int NOT NULL;
```

UNIQUE Constraint

- The UNIQUE constraint ensures that all values in a column are different.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table

UNIQUE Constraint (Cont.)

- UNIQUE on CREATE TABLE.

- Example

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    UNIQUE (ID)
);
```

UNIQUE Constraint (Cont.)

- To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns.

- Example

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT UC_Person UNIQUE
(ID, LastName));
```

UNIQUE Constraint (Cont.)

- UNIQUE on ALTER TABLE.

- Example

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

UNIQUE Constraint (Cont.)

- To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns.

❑ Example

```
ALTER TABLE Persons
ADD CONSTRAINT UC_Person UNIQUE
 (ID,LastName);
```

UNIQUE Constraint (Cont.)

- To drop a UNIQUE constraint.

- Example

```
ALTER TABLE Persons
DROP INDEX UC_Person;
```

PRIMARY KEY Constraint

- Uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields)

PRIMARY KEY Constraint (Cont.)

- PRIMARY KEY on CREATE TABLE.

- Example

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID));
```

PRIMARY KEY Constraint (Cont.)

- To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns.

❑ Example

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID
,LastName));
```

PRIMARY KEY Constraint (Cont.)

- PRIMARY KEY on ALTER TABLE.

- Example

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

PRIMARY KEY Constraint (Cont.)

- To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns.

- Example

```
ALTER TABLE Persons
ADD CONSTRAINT PK_Person PRIMAR
Y KEY (ID,LastName);
```

PRIMARY KEY Constraint (Cont.)

- To drop a PRIMARY KEY constraint.

- Example

```
ALTER TABLE Persons
DROP PRIMARY KEY;
```

FOREIGN KEY Constraint

- Used to prevent actions that would destroy links between tables.
- Is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
- The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

FOREIGN KEY Constraint (Cont.)

- FOREIGN KEY on CREATE TABLE.

- Example

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFEREN
CES Persons(PersonID));
```

FOREIGN KEY Constraint (Cont.)

- To name a FOREIGN KEY constraint, and to define a FOREIGN KEY constraint on multiple columns.

- Example

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    CONSTRAINT FK_PersonOrder FOREIGN K
EY (PersonID)
REFERENCES Persons(PersonID));
```

FOREIGN KEY Constraint (Cont.)

- FOREIGN KEY on ALTER TABLE.

- Example

```
ALTER TABLE Orders
ADD FOREIGN KEY (PersonID) REFERE
NCES Persons(PersonID);
```

FOREIGN KEY Constraint (Cont.)

- To name a FOREIGN KEY constraint, and to define a FOREIGN KEY constraint on multiple columns.

- Example

```
ALTER TABLE Orders
ADD CONSTRAINT FK_PersonOrder
FOREIGN KEY (PersonID) REFERENCES
 Persons(PersonID);
```

FOREIGN KEY Constraint (Cont.)

- To drop a FOREIGN KEY constraint.

- Example

```
ALTER TABLE Orders
DROP FOREIGN KEY FK_PersonOrder;
```

CHECK Constraint

- The CHECK constraint is used to limit the value range that can be placed in a column

CHECK Constraint (Cont.)

- CHECK on CREATE TABLE.

- Example

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18));
```

CHECK Constraint (Cont.)

- To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns.

- Example

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=
18 AND City='Sandnes'));
```

CHECK Constraint (Cont.)

- CHECK on ALTER TABLE.

- Example

```
ALTER TABLE Persons
ADD CHECK (Age>=18);
```

CHECK Constraint (Cont.)

- To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns.

- Example

```
ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CH
ECK (Age>=18 AND City='Sandnes');
```

CHECK Constraint (Cont.)

- To drop a CHECK constraint.

- Example

```
ALTER TABLE Persons
DROP CHECK CHK_PersonAge;
```

DEFAULT Constraint

- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified

DEFAULT Constraint (Cont.)

- DEFAULT on CREATE TABLE.

- Example

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City
varchar(255) DEFAULT 'Sandnes');
```

DEFAULT Constraint (Cont.)

- The DEFAULT constraint can also be used to insert system values, by using functions like CURRENT_DATE().

❑Example

```
CREATE TABLE Orders (
    ID int NOT NULL,
    OrderNumber int NOT NULL,
    OrderDate
date DEFAULT CURRENT_DATE());
```

DEFAULT Constraint (Cont.)

- DEFAULT on ALTER TABLE.

- Example

```
ALTER TABLE Persons
ALTER City SET DEFAULT 'Sandnes';
```

DEFAULT Constraint (Cont.)

- To drop a DEFAULT constraint.

- Example

```
ALTER TABLE Persons
ALTER City DROP DEFAULT;
```

CREATE INDEX

- The CREATE INDEX statement is used to create indexes in tables.

- Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries

CREATE INDEX (Cont.)

- Creates an index on a table. Duplicate values are allowed.

- Syntax

CREATE INDEX *index_name*
ON *table_name* (*column1*, *column2*, …);

CREATE INDEX (Cont.)

- Creates a unique index on a table. Duplicate values are not allowed.

- Syntax

CREATE UNIQUE INDEX *index_name* ON *table_name* (*column1*, *column2*, …);

CREATE INDEX (Cont.)

- Example

```
CREATE INDEX idx_lastname
ON Persons (LastName);
```

- Index on combination of columns

- Example

```
CREATE INDEX idx_pname
ON Persons (LastName, FirstName);
```

DROP INDEX

- Used to delete an index in a table.

- Syntax

```
ALTER TABLE table_name
DROP INDEX index_name;
```

AUTO INCREMENT Field

- Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

- Often this is the primary key field that we would like to be created automatically every time a new record is inserted

AUTO INCREMENT Field (Cont.)

- MySQL uses the AUTO_INCREMENT keyword to perform an auto-increment feature.

- By default, the starting value for AUTO_INCREMENT is 1, and it will increment by 1 for each new record.

AUTO INCREMENT Field (Cont.)

- Example

```
CREATE TABLE Persons (
    Personid
int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (Personid));
```

AUTO INCREMENT Field (Cont.)

- To let the AUTO_INCREMENT sequence start with another value

❑ Example

```
ALTER TABLE Persons
AUTO_INCREMENT=100;
```

Data Types

- Each column in a database table is required to have a name and a data type.

- In MySQL there are three main data types: string, numeric, and date and time.

(ex. CHAR, VARCHAR, BOOLEAN, INT, FLOAT, DOUBLE, DATE, DATETIME, YEAR)

Any Questions?

Thank you