POWERING POTENTIAL

MySQL Part 1

Module 18

MYSQL

Contents

What are Databases?

A collection of objects and the relationships that link them together

- → They can store a wide range of information
- → Mostly text / numbers
- → Images
- → Sound
- → Video

> “A database is a collection of data, typically describing the activities of one or more related organizations”
>
> – Ramakrishnan & Gehrike, Database Management Systems, 3rd Ed, 2003

Databases come in different forms

- → Flat-files store all the data in a single file
- → Networks of linked objects and their relationships
- → Hierarchical databases are a tree structure

A Brief History of Databases

As more information was being stored there was a need to organise it somehow

- → Need some way of searching for information
- → Need to answer questions

They have existed in some form since the early 1960s

- → Started with a network model
- → Relational model proposed in the 1970s representing both objects and the relationships between them

SQL – Structured Query Language

- → Used to communicate with the database
- → Formalised in the late 1980s
- → Various revisions (1986, 1989, 1992, 1999, 2003, 2006, 2008, 2011)

What are Databases Used For?

Storing information in a way that we can use
→ Query information to answer questions

Databases are used in most programs, webpages and applications
→ Track current stock levels
→ Login details
→ High score table in a game
→ Music collection
→ Financial reports

A good database is able to
→ Store information efficiently
→ Provide fast and easy access
→ Keep data correct and safe (integrity and security)

Relational Databases

Relational databases store information in **tables**

→The links between the tables are known as relations
→Tables are formed of columns and rows, sometimes known as fields and records
→Each field or column has a name and a data type
→The types of data you can use depend upon the database server you are using

Within each table one or more composite columns are designated as the **primary key**

→The data in this column(s) must be unique within the table

A table may contain **foreign keys** which cross-reference the record to a primary key in another table

The **schema** of a database describes its tables including their column names and data types

NoSQL Databases ("Not only SQL")

Does not use table based data

- Used for storing (persisting) objects
- Objects can be anything

Different methods of storing and accessing information

- Column – Uses timestamps to differentiate data (Cassandra)
- Document – Uses XML / JSON to store information (MongoDB)
- Key/Value – Associative arrays (CouchDB)
- Graph – Based on graph theory, many links between objects
- Multi-modal – Mixtures of the above and relational model

Support query languages

- Often similar to SQL

Database Management Systems

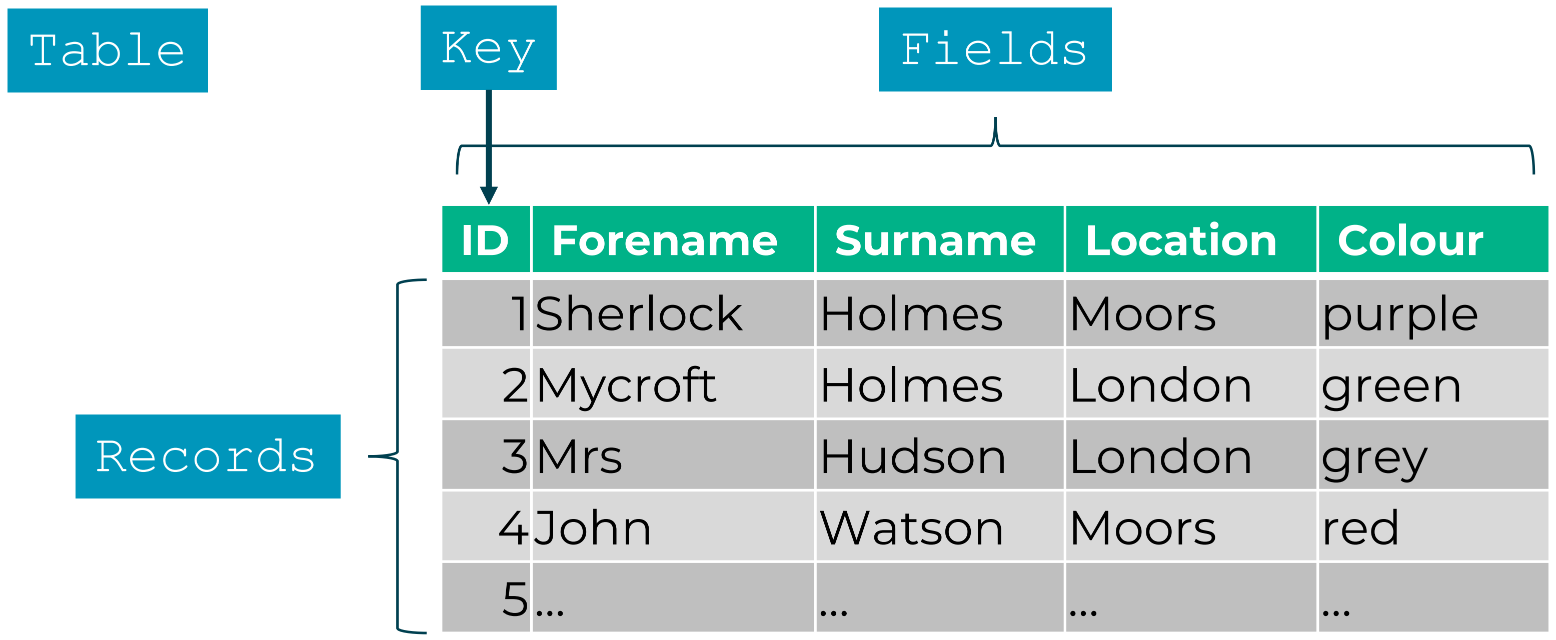
The DBMS is the engine that controls the database
→ You do not interact directly with data, but instead pass queries through the DBMS

Many different DBMS available
→ Oracle
→ Sybase (Part of SAP)
→ DB2
→ Microsoft SQL Server
→ PostgreSQL
→ MySQL
→ MariaDB

Intro to Relational Databases

Information is stored in databases in tables

- Each table should be about one ‘thing’
- Each row in a table is known as a record
- Each column has a heading known as the field
- Each row must have a unique identifier known as the key



ID	Forename	Surname	Location	Colour
1	Sherlock	Holmes	Moors	purple
2	Mycroft	Holmes	London	green
3	Mrs	Hudson	London	grey
4	John	Watson	Moors	red
5	…	…	…	…

Relationships between data

One to One

- For each ID in the first table, there is exactly one record in the second table
- Often used when a table has many different fields and we want to segment data

One to Many

- A one way relationship, there is one record in the first table connecting to many in the second
- For example:

→ *One trainer will have one home location but…*

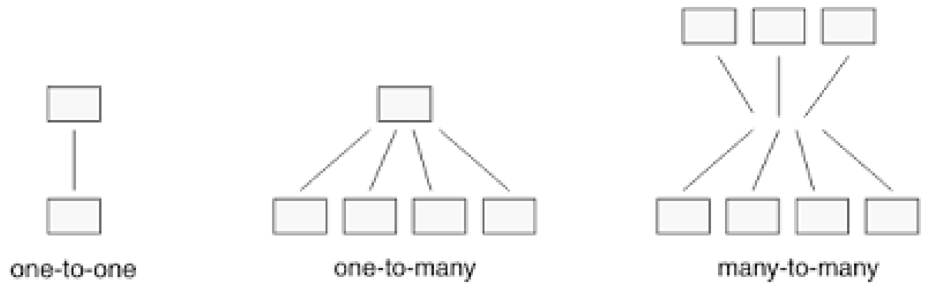
→ *One home location will have many trainers*

Many to Many

- For each record in the first table there are many records in the second table and vice versa
- For example:

→ *A delegate can attend many events*

→ *An event can have many delegates*



Terminology

DBMS – Database Management System

→ The engine that we interact with that stores the tables, queries and procedures. Processes SQL statements and responds with the requested information

Table

→ Information stored about a specific ‘thing’ or entity

Record / Row

→ A row in the table, a single instance of that ‘thing’

Field / Column

→ The column headings, specific attributes about each record

Key

→ Primary, Foreign and Compound

→ Unique identifiers for tables and ways to relate data together

SQL

→ Structured Query Language

Introducing MySQL

MySQL a popular open source relational database management system

→ Currently owned by Oracle

→ Available for Linux, AIX, BSD, Mac OS X, Windows, Solaris, others …

→ Free version released under GPL2 licence

Enterprise versions offer additional features

→ Backup

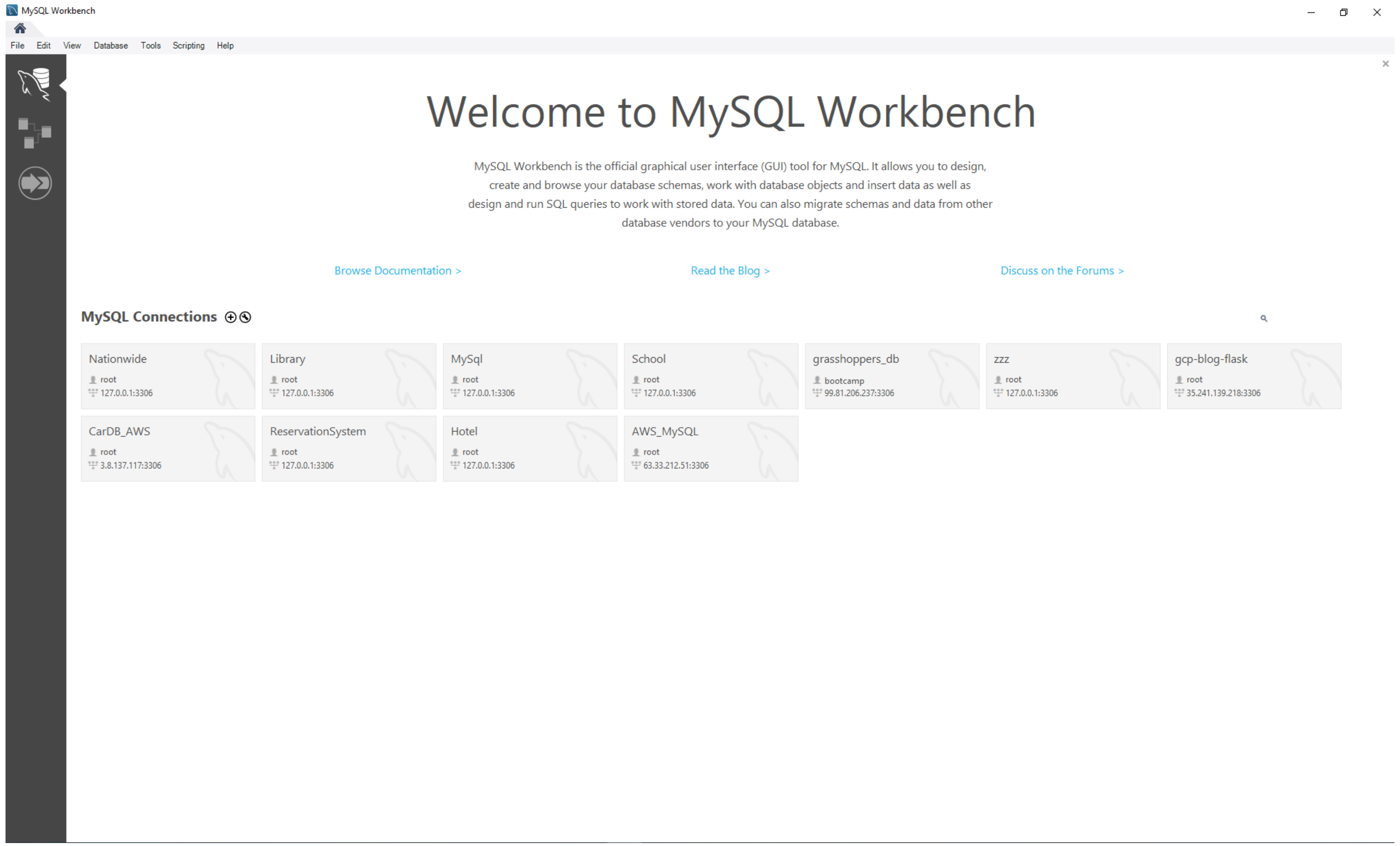
→ Replication and High Availability

→ Scalability

→ Auditing

→ Technical Support

MySQL Workbench



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Browse Documentation >

Read the Blog >

Discuss on the Forums >

MySQL Connections

Connection	User	Host
Nationwide	root	127.0.0.1:3306
Library	root	127.0.0.1:3306
MySql	root	127.0.0.1:3306
School	root	127.0.0.1:3306
grasshoppers_db	bootcamp	99.81.206.237:3306
zzz	root	127.0.0.1:3306
gcp-blog-flask	root	35.241.139.218:3306
CarDB_AWS	root	3.8.137.117:3306
ReservationSystem	root	127.0.0.1:3306
Hotel	root	127.0.0.1:3306
AWS_MySQL	root	63.33.212.51:3306

What is SQL?

Structured Query Language

→ It allows us to communicate with a database

→ Statements are interpreted by the DBMS and changes made to the underlying tables and structure

SQL can be used for many tasks

→ Creating a database

→ Updating and inserting new records

→ Querying a database

Database backup is done by translating the tables and contents into a set of SQL statements that can be executed to recreate the system. Use DATA EXPORT and DATA IMPORT options of the SERVER menu in MySQL Workbench.

SQL Categories

Data Query Language (DQL)
→ Use the SELECT statement to display data

Data Definition Language (DDL)
→ Define objects within the database with CREATE
→ Modify the structure of database objects with ALTER
→ Permanently remove a database object with DROP

Data Manipulation Language (DML)
→ Add records to a table with INSERT
→ Modify table data with UPDATE
→ Remove table data with DELETE

Data Control Language (DCL)
→ Control database permissions with GRANT and REVOKE

Switching Database

```
mysql> use mysql;
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

Showing Tables

```
mysql> show tables;
+----------------------------+
| Tables_in_mysql            |
+----------------------------+
| columns_priv               |
| db                         |
| event                      |
| func                       |

…

| user                       |
+----------------------------+
34 rows in set (0.00 sec)

mysql>
```

Importing a database

Step 1: Create an SQL file or create a dump of an existing database using DATA EXPORT

```
-- create a database
CREATE DATABASE QATRAINING;
use QATraining;

-- TABLES
CREATE TABLE clients(
      ClientID int not null primary key AUTO_INCREMENT,
      CompanyName varchar(50) not null
);
…etc
```

Step 2: Import the database via the DATA IMPORT option of the Server menu

Data Definition Language (DDL)

Creating and Deleting Databases Using MySQL

Create a database

```
Without this, attempting to create
a database that already exists
would cause an error
```

```
create database if not exists store;
```

Delete a database

→ Be careful!

```
Without this, attempting to delete
a database that does not exist
would cause an error
```

```
drop database if exists store;
```

Creating and Deleting Tables

Creating tables

```
use cart;

create table user
(   id int not null primary key auto_increment,

    username varchar(100) not null,
    password varchar(100) not null
);
```

Create the table in this database

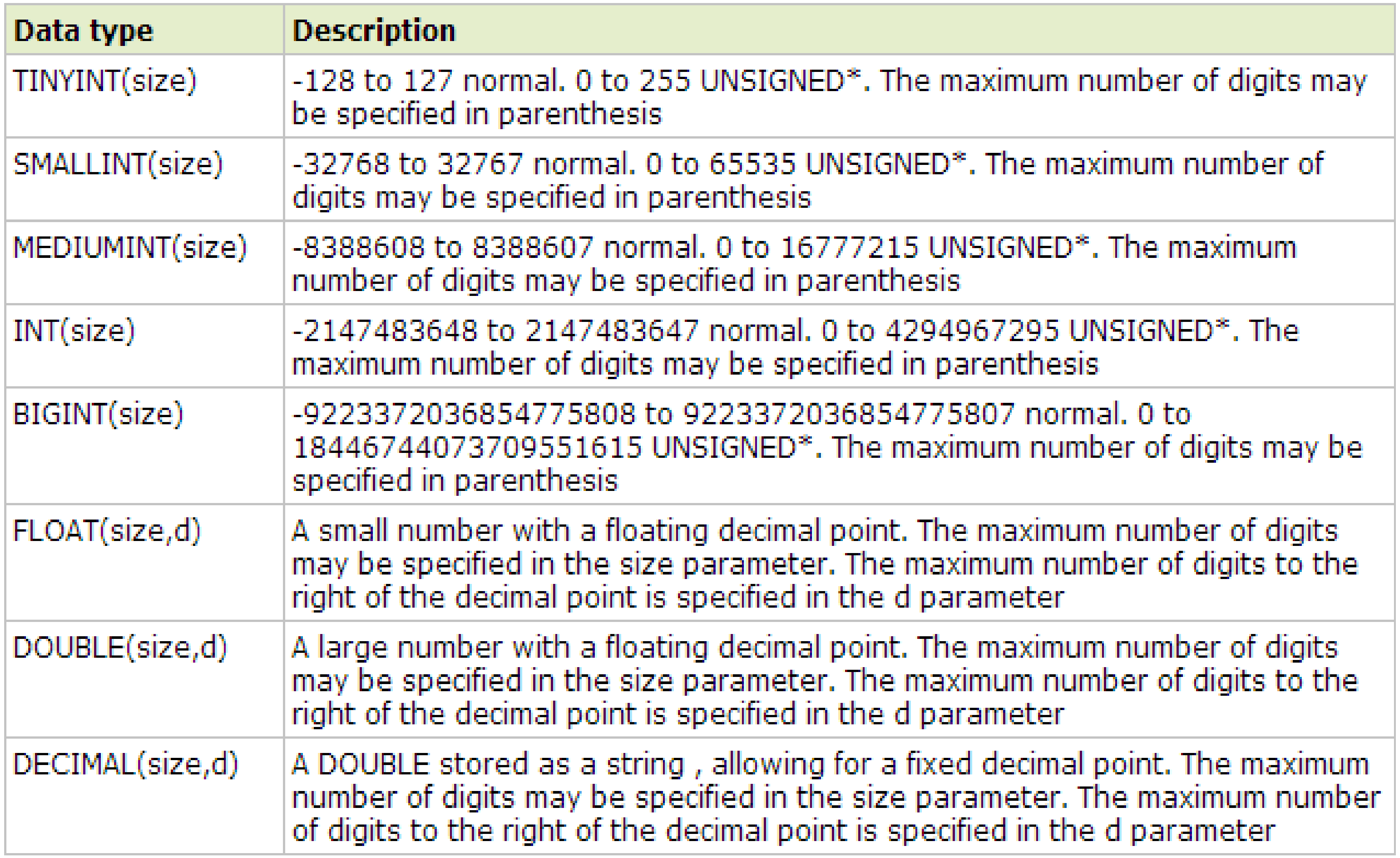
Column type

Column name

Deleting tables

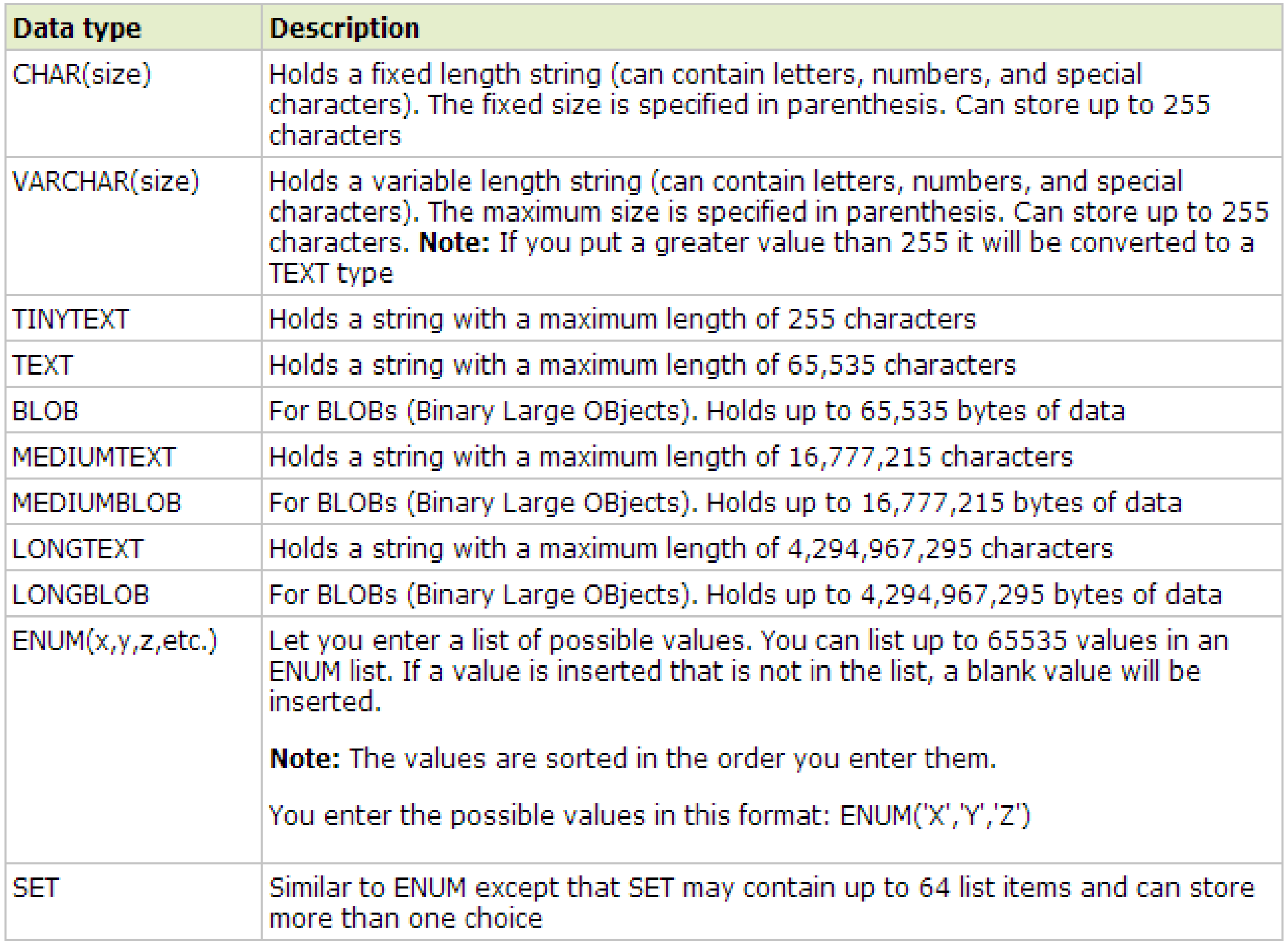
```
drop table user;
```

Numeric data types in MySQL



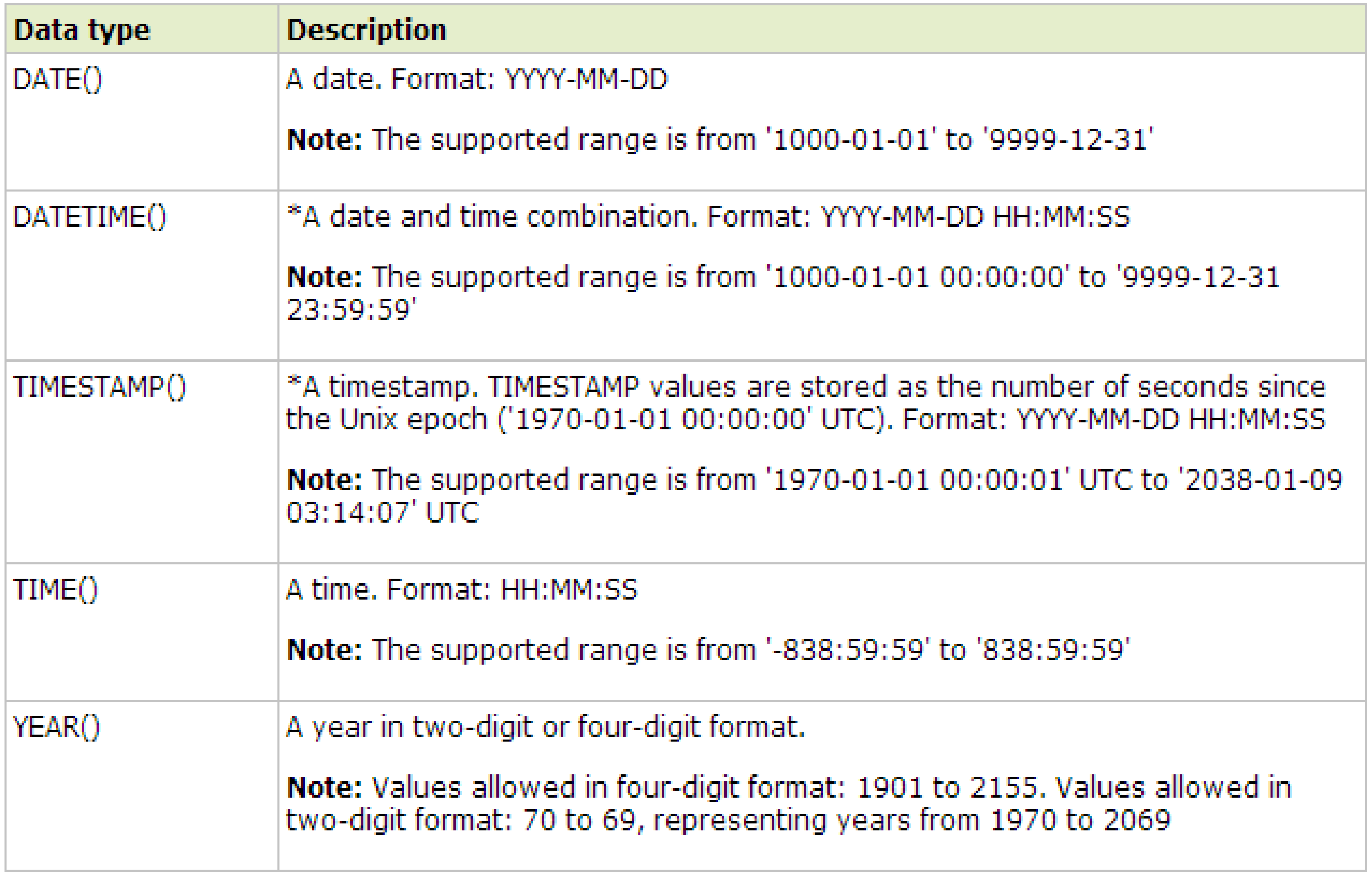
Data type	Description
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

Text Data Types in MySQL



Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. **Note:** The values are sorted in the order you enter them. You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

Time and Date Data Types in MySQL



Data type	Description
DATE()	A date. Format: YYYY-MM-DD **Note:** The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MM:SS **Note:** The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MM:SS **Note:** The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MM:SS **Note:** The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. **Note:** Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

Defining a Primary Key

- To identify a field as a primary key add the **primary key** attribute after the field type

```
id int primary key
```

- This will enforce the fact that each entry must be unique
- To create a composite (multi-column key) define a key after the column list

```
mysql>

create table MyTable (
           id1 int,
           id2 int,
           primary key (id1, id2)
);
```

Defining a Foreign Key

Foreign key constraints can be defined when the table is created

```
create table cart
(
       user_id int,
       item_id int,

       foreign key (item_id) references items(id),
   foreign key (user_id) references users(id)
) engine = innodb;
```

Parent table

Must specify innodb. The default storage back-end ("myisam") does not understand foreign keys

Primary key in parent table

Storage back-ends

MySQL supports several "back-ends" for storage, including

Backend	Description
MyISAM	The default storage engine. Fast and lightweight but does not support transactions or foreign keys
InnoDB	A modern storage engine that supports transactions and foreign keys
IBMDB2I	Intended for interoperability with native IBM DB2 databases
MEMORY	In-memory storage – very fast but will not survive a restart of the server
CSV	Uses simple comma separated value files
BLACKHOLE	The mysql equivalent of `/dev/null`

The back-end type is specified when the table is created:

```
create table foobar
( ...
) engine = innodb;
```

auto_increment, not null and other constraints

Fields can be constrained so only certain items can be stored:

auto_increment

→ Allows for the field to be automatically calculated based on the previous record. This is frequently used for primary keys

not null

→ When adding a record to the table this field must be provided

default

→ The default value to be stored in the table if no value is provided

check [salary < 50000]

→ Allows us to place a constraint directly on the data being stored

Referential Integrity

Referential integrity means that every foreign key in a table has a matching primary key in the table it refers to

```
mysql> delete from cart where item_id = 102;
```

The InnoDB storage back-end is able to enforce referential integrity:

```
mysql> delete from cart where item_id = 102;
ERROR 1451 (23000): Cannot delete or update a parent row: a
foreign key constraint fails
```

Inserting Data

The Insert SQL statement adds new information into the database

```
insert into [table] values ([value1], [value2], … );
insert into item values(1, "Camera", 1700);
```

You can specify which fields to insert

→Any column not named will be filled with nulls

```
insert into [table](column1, column2)
values ([value1], [value2], … );

insert into user(username, email) values ('sholmes',
'sholmes@example.com');
```

Inserting more than one row at once

You can add more than one row at a time

→ Keep adding the values as comma separated tuples

```
insert into items(name, price)
values ('Sony A6000', 500),
       ('Sony A7S', 1500),
       ('Sony A7RII', 2000);
```

→ This is often the format generated by the backup / data export program

This format typically does not work with code libraries

- In your application code you will need to issue multiple independent INSERTs

Common Errors

ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY’

- → You have tried to add a record with the same key as another record in the table already. Primary keys need to be unique

ERROR 1364 (HY000): Field 'id' doesn't have a default value

- → You have tried to add a record but not specified all the columns and the record can not be created as there is no default value for the non-null column stated.
- → You can also get this if you use default for a field that does not have a default value available

ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails

- → The foreign key used in the record does not exist in the other table. The database is trying to preserve referential integrity
- → This will only work in InnoDB tables!

Summary: Creating a Database and Tables

If you omit the **id** field a value will be automatically generated (if auto_increment is used)

SQL commands may be upper or lower case

```
CREATE DATABASE dbname;

USE dbname;

CREATE TABLE tblname (
      id int primary key auto_increment,
      fieldTwo text
);

INSERT INTO tblname(fieldTwo) VALUES("sometext");
```

Activity:
Instructor Walkthrough

Your instructor will walk you through creating a database with tables and data.

Any questions?

Activity:
Exercise 18