SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20031

Computing Technology Design Project

Week 06

Physical Implementation in MySQL

Database Development Lifecycle

1. Planning
2. Requirement gathering
3. Conceptual design
4. Logical design
5. **Physical design**
6. Construction
7. Implementation and rollout
8. Ongoing support

(A)DBMS (database engine) and client application

DBMS Options

- **MySQL on Mercury**:
  - with phpMyAdmin as client (easiest option)
  - with locally installed MySQL workbench
- **Local Installation of MySQL**:
  - with locally installed MySQL workbench

Install **Xampp**: contains Mysql and Apache web server
Youtube video (instructions):
https://www.youtube.com/watch?v=-f8N4FEQWyY

(B) Introduction to MySQL, SQL

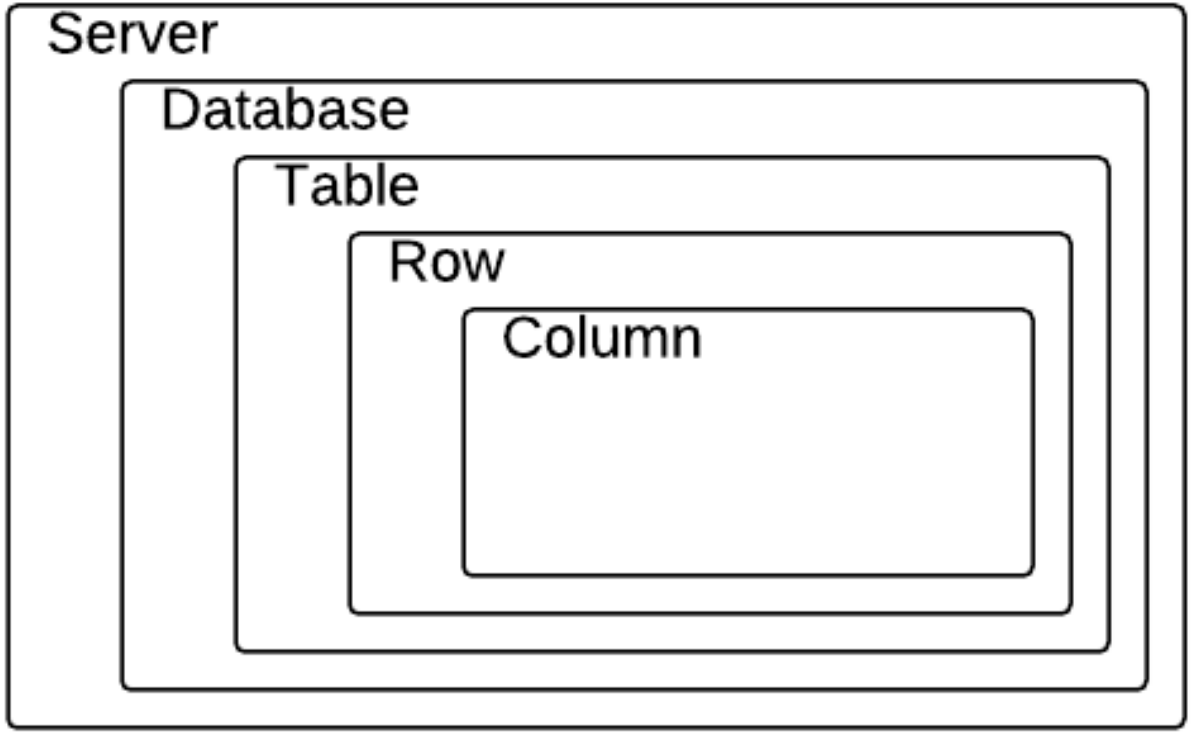
What is MySQL?

- A Relational Database Management System (RDBMS)
- Free, Open-Source Software from Oracle
- The most popular database on the planet: Powers Facebook, Wikipedia, Wordpress, many more
- Ubiquitous - it runs on everything from mobile devices up to large database server clusters

What is SQL?

- SQL is the most popular language for talking to RDBMS's like MySQL; other query languages have not gained wide acceptance.
- Stands for Structured Query Language
- The first version was SEQUEL for Structured English QUEry Language, and SQL is still often pronounced "Sequel"

How information is organized



(B) Creating the Physical Model

MySQL documentation:
https://dev.mysql.com/doc/refman/8.0/en/

MySQL fundamental types

MySQL supports a number of SQL standard types in various categories.

- Numeric
  - Integer
  - Float
- String
  - Fixed length string
  - Variable length string
  - Binary string
  - Large objects
- Date and Time
  - Date
  - Date time
  - Timestamp
- Specialty

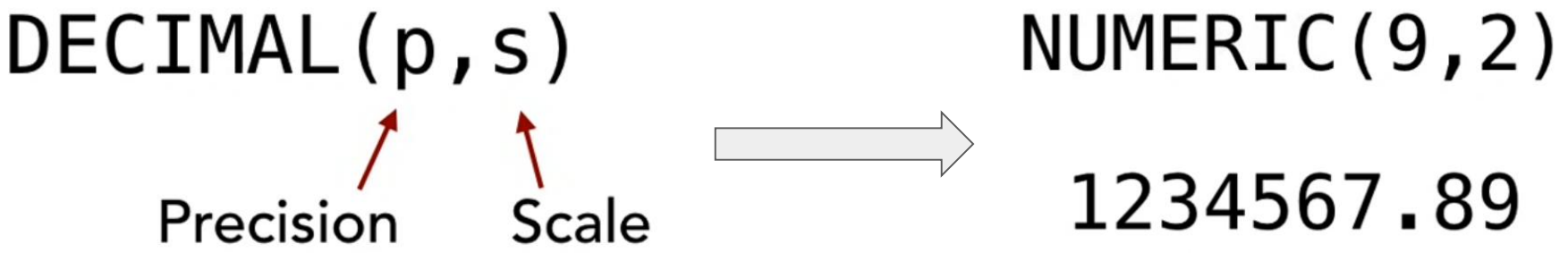
https://dev.mysql.com/doc/refman/8.0/en/data-types.html

For SQL examples, refer to file **sql-chap02.sql** of the LinkedIn course resource.

Integer types

Type	Storage *Bytes*	Minimum Value	Maximum Value
TINYINT	1	-128	127
Unsigned		0	255
SMALLINT	2	-32,768	32,767
Unsigned		0	65,535
MEDIUMINT	3	-8,388,608	8,388,607
Unsigned		0	16,777,215
INTEGER	4	-2,147,483,648	2,147,483,647
Unsigned		0	4,294,967,295
BIGINT	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
Unsigned		0	18,446,744,073,709,551,615

Decimal type

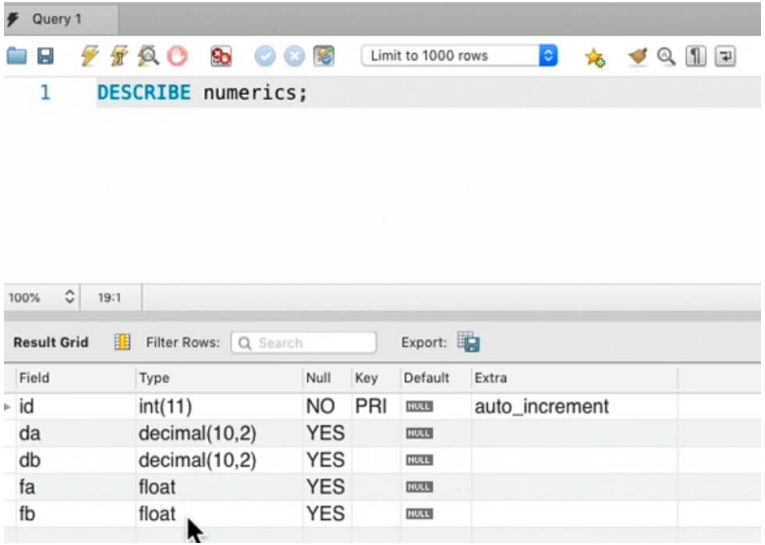


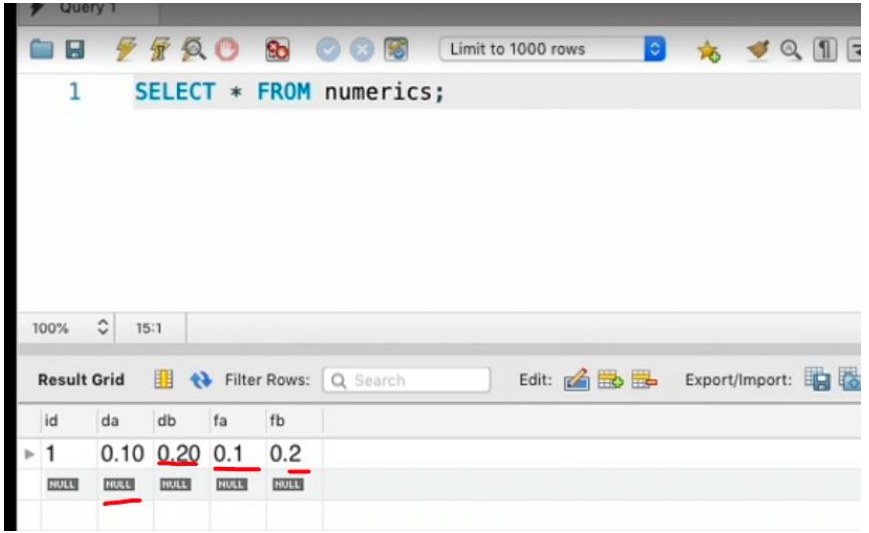
Floating point types

Type	Precision (bits)	Precision (digits)	Numeric Range
FLOAT	24	~7	±3.4e38
DOUBLE	53	~16	±1.79e308

* REAL *is* FLOAT

Floating point types





Data and Time types

Types	Description	Display Format	Range
DATETIME	Use when you need values containing both date and time information.	YYYY-MM-DD HH:MM:SS	'1000-01-01 00:00: 00' to '9999-12-31 23:59:59'.
DATE	Use when you need only date information.	YYYY-MM-DD	'1000-01-01' to '9999-12-31'.
TIMESTAMP	Values are converted from the current timezone to UTC while storing, and converted back from UTC to the current time zone when retrieved.	YYYY-MMDD HH:MM:SS	'1970-01-01 00:00: 01' UTC to '2038-01-19 03:14:07' UTC

Character string

Types	Description	Display Format	Range in characters
CHAR	Contains non-binary strings. Length is fixed as you declare while creating a table. When stored, they are rightpadded with spaces to the specified length.	Trailing spaces are removed.	The length can be any value from 0 to 255.
VARCHAR	Contains non-binary strings. Columns are variable-length strings.	As stored.	A value from 0 to 255 before MySQL 5.0.3, and 0 to 65,535 in 5.0.3 and later versions.

`CHAR(5)` *Always uses 5 bytes*

`VARCHAR(25)` *Uses up to 25 + 1 bytes*

Binary string

Types	Description	Range in bytes
BINARY	Contains binary strings.	0 to 255
VARBINARY	Contains binary strings.	A value from 0 to 255 before MySQL 5.0.3, and 0 to 65,535 in 5.0.3 and later versions.

Large object storage

Data Type	Storage Required
TINYBLOB, TINYTEXT	up to 256 + 1 bytes
BLOB, TEXT	up to 65,536 + 2 bytes
MEDIUMBLOB, MEDIUMTEXT	up to 16777216 + 3 bytes
LONGBLOB, LONGTEXT	up to 4294967296 + 4 bytes

Creating MySQL Databases

- For the session we have a database pre-created for our use
- Normally you would use this SQL statement:

```
CREATE DATABASE `foo`;
```
(Creates a database called foo)

- Then to create a user with permission to use our new database:

```
GRANT ALL ON `foo`.* TO 'someuser'@'somehost'
IDENTIFIED BY 'somepass';
```

Creating MySQL Tables

- Databases store tables, so now that we have a database we need to define a table to store data in it
- Here's an SQL statement to create a table:

```
CREATE TABLE `vm_foo` (
  `id` BIGINT UNSIGNED NOT NULL
    AUTO_INCREMENT UNIQUE,
  `name` VARCHAR(100));
```

- I've prefixed the table name with my initials (vm) and an underscore so that we can share the same database in the session. Please use your initials.

Querying Our Data

We can ask to see our data with a SELECT statement:

```
SELECT * FROM `vm_foo`;
```

Returns:

```
+----+------+
| id | name |
+----+------+
|  1 | baz  |
|  2 | bar  |
+----+------+
2 rows in set (0.00 sec)
```

Querying Certain Rows

We can add a WHERE clause to our SELECT statements to have the server only return certain rows:

```
SELECT * FROM `vm_foo` WHERE `id`=2;
```

Returns:

```
+----+------+
| id | name |
+----+------+
|  2 | bar  |
+----+------+
1 row in set (0.00 sec)
```

Getting Ordered Results

You can get the results in order by adding an ORDER BY clause to your select statement:

```
SELECT * FROM `vm_foo` ORDER BY `name`;
```

Returns:

```
+----+------+
| id | name |
+----+------+
|  2 | bar  |
|  1 | baz  |
+----+------+
2 rows in set (0.00 sec)
```

Getting Only the Columns you Want

So far we've always used SELECT * which returns all columns. We can list the columns we want to see instead of the star:

```
SELECT `name` FROM `vm_foo`;
```

Returns:

```
+------+
| name |
+------+
| baz  |
| bar  |
+------+
2 rows in set (0.00 sec)
```

Aggregate Functions

Aggregate functions summarize query results. They can be used for things like adding together column values, or counting the number of returned rows.

Try these:

```
SELECT COUNT(*) FROM `vm_foo`;

SELECT SUM(`id`) FROM `vm_foo`;
```

Foreign key relationship

- A `FOREIGN KEY` 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.
- The `FOREIGN KEY` constraint is used to prevent actions that would destroy links between tables.

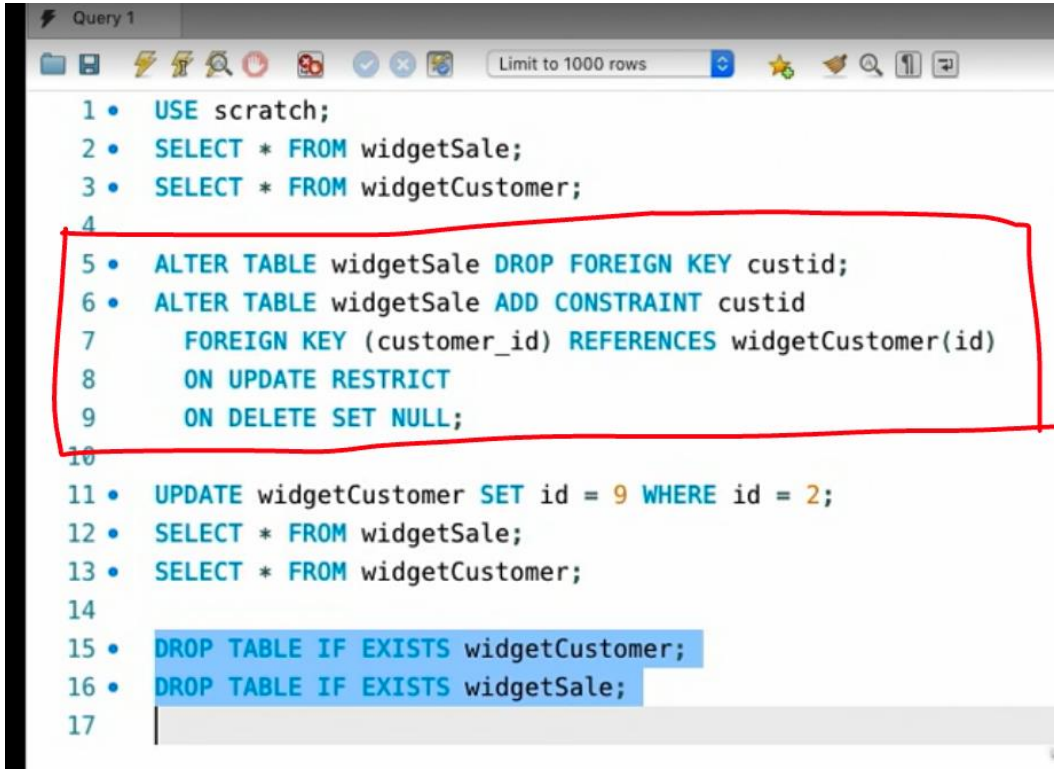
Foreign key constraint

```
CREATE TABLE widgetSale (
    id INTEGER AUTO_INCREMENT PRIMARY KEY,
    item_id INTEGER,
    customer_id INTEGER,
    quan INTEGER,
    price INTEGER,

    INDEX custid(customer_id),
    CONSTRAINT custid FOREIGN KEY custid(customer_id)
        REFERENCES widgetCustomer(id)
        ON UPDATE RESTRICT
        ON DELETE SET NULL
);
```

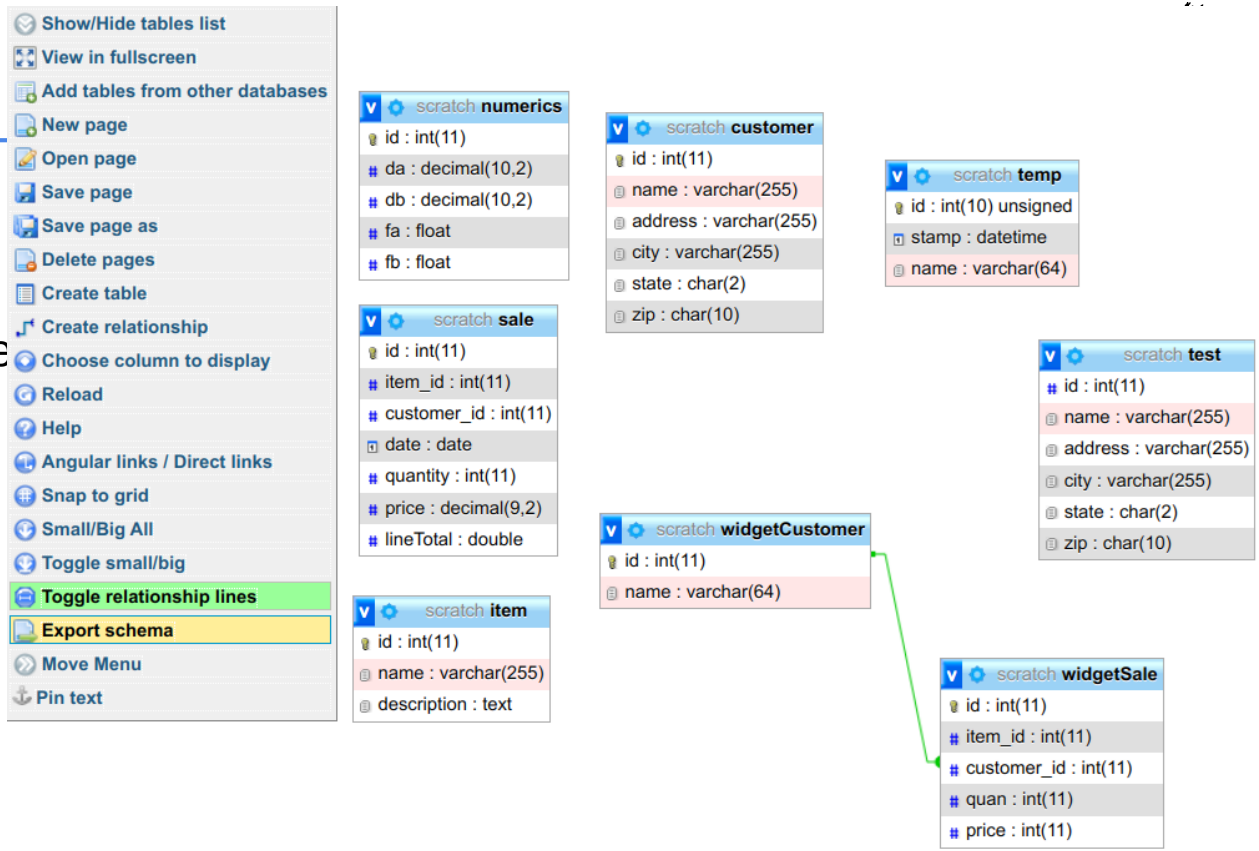
Refer to file **chap06.sql** of the LinkedIn ***advanced*** course resource.

Deleting and changing foreign keys



MySQL Designer

- Menu: **phpMyAdmin/Designer**
- Visually displays database model
- Export database design specification as PDF



Database model export: TOC

Table of contents

Database model export: Customer

1 customer

Creation: Oct 08, 2023 at 08:24 PM
Last update: Oct 08, 2023 at 08:24 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
id	int(11)		No		auto_increment			
name	varchar(255)		Yes	NULL				
address	varchar(255)		Yes	NULL				
city	varchar(255)		Yes	NULL				
state	char(2)		Yes	NULL				
zip	char(10)		Yes	NULL				

Project Update

- Write the CREATE TABLE statements for each table in your logical model. This is your physical model.
- Adjust the Jira backlog to reflect the work to be done.

Tutorial & Workshop

See Canvas.