</talentlabs>

Express Lecture 10

Work with a MySQL Database

</talentlabs>

Agenda

- Data Model Design
- Database Introduction
- Database on the cloud
- MySQL demo

</talentlabs>

Data Model Design with Entity Relationship Diagram (ERD)

</talentlabs>

</talentlabs>

Database Modeling Review

This section only covers the basics of designing a SQL Database, for advance design please revisit the Database Module.

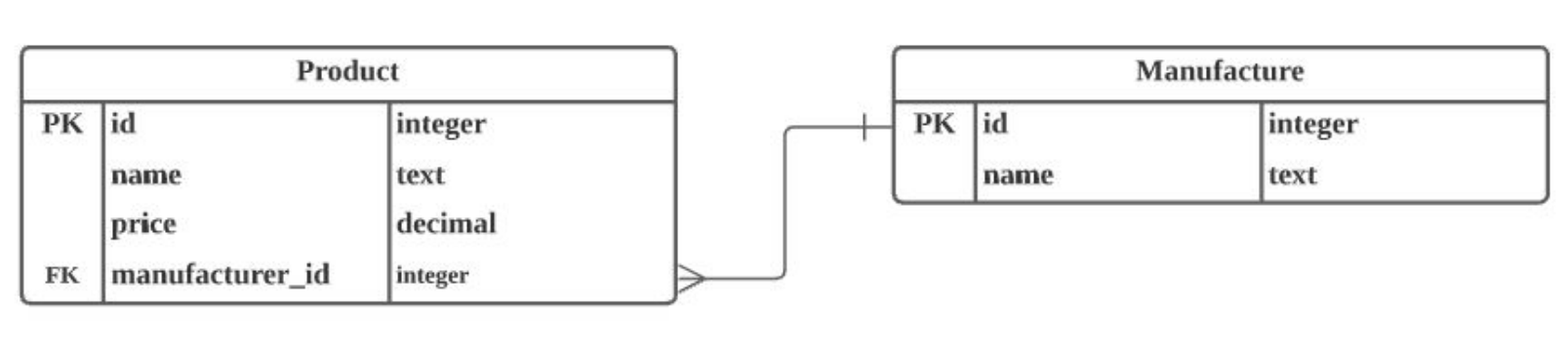
A **table represents an entity**. A **line represents the relationship between 2 entities**. Therefore we call these diagrams **Entity Relationship Diagram (ERD).**

Each **entity contains different columns**, and each **column has a data type.**

Important concepts in SQL database:

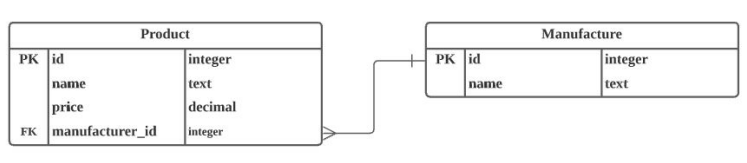
- Table
- Column
- Column Type
- Primary Key
- Foreign Key

ERD Example



Here is the ERD for the Product Listing application. You can see we assume a **Manufacturer** can have many **Products**.
Two tables can be **joined by the column manufacturer_id**.

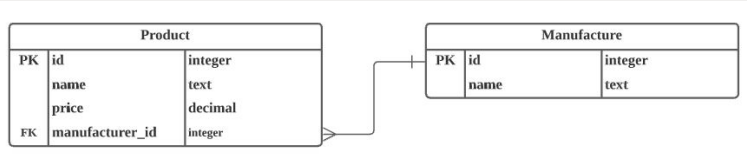
Example Product data



Here is the ERD for the Product Listing application. You can see we assume a **Manufacturer** can have many **Products**. Two tables can be **joined by the column manufacturer_id**.

id	name	price	manufacture_id
1	Lego City 2824: Advent Calendar 2010	3.42	1
2	LEGO Friends 41016: Advent Calendar	24.95	1
3	LEGO Star Wars 75018: Jek-14's Stealth Starfighter	68.87	1
4	Disney Phineas and Ferb 8 Ferb Plush, soft, cuddle doll toy	19.99	2
5	DESPICABLE ME 2 - Minion cuddly Soft Toy - Plush Figures Banana 28-33 cm, Minion Typ:Bob	19.99	2

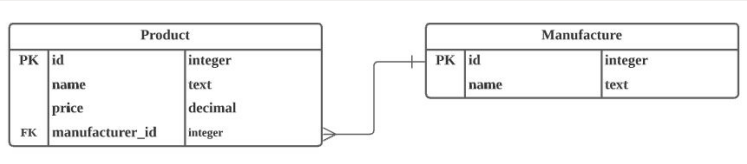
Example Manufacturer data



id	name
1	Lego
2	Disney

Here is the ERD for the Product Listing application. You can see we assume a **Manufacturer** can have many **Products**. Two tables can be **joined by the column manufacturer_id**.

Example data



Example **Manufacturer** table data:

id	name
1	Lego
2	Disney

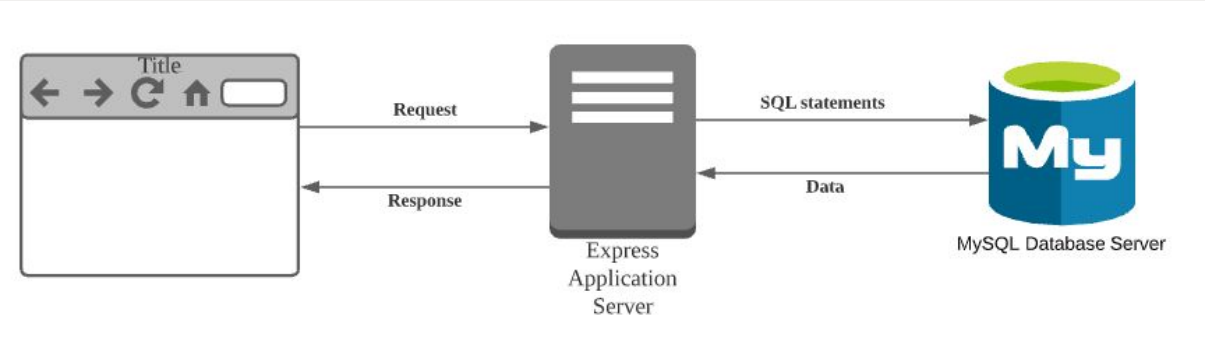
Example **Product** table data:

id	name	price	manufacture_id
1	Lego City 2824: Advent Calendar 2010	3.42	1
2	LEGO Friends 41016: Advent Calendar	24.95	1
3	LEGO Star Wars 75018: Jek-14's Stealth Starfighter	68.87	1
4	Disney Phineas and Ferb 8 Ferb Plush, soft, cuddle doll toy	19.99	2
5	DESPICABLE ME 2 - Minion cuddly Soft Toy - Plush Figures Banana 28-33 cm, Minion Typ:Bob	19.99	2

Adding Database to the System

</talentlabs>

Database is just another server

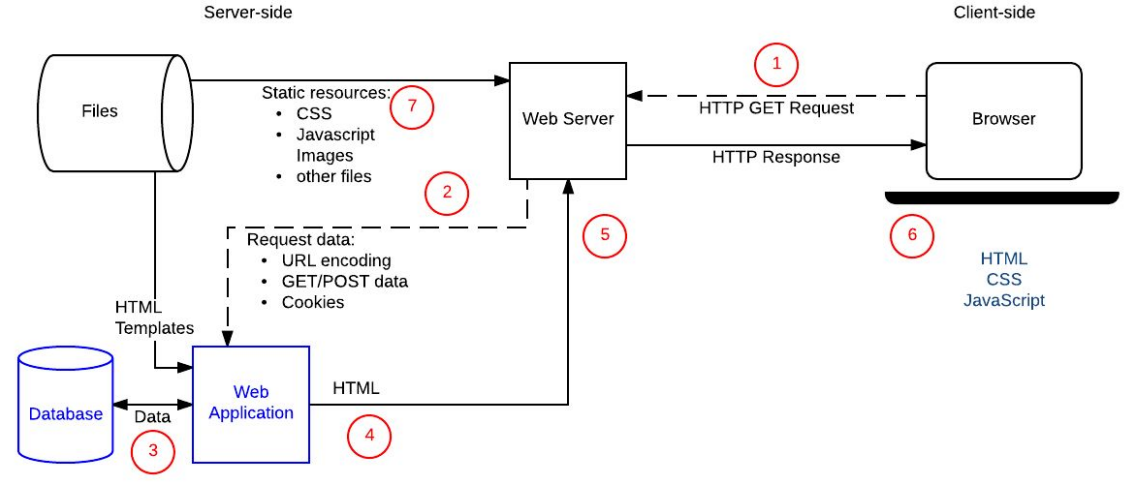


Most of the time, we will persist data into a database. It is because:

1. Database is designed to **query data efficiently**.
2. Database is designed to have some level of **fault tolerance and data integrity.**

We can treat the **database server just as another computer/server.** When the Express application needs data to fulfil a request, it will ask for the database.

Let’s review the Server-side rendering diagram



To understand the full picture, look at the steps 2 ~ 4 in the above diagram. We can see in order to render a HTML template, the Express application might need to **ask for data from the database**.

Previously, we have been doing the steps 2 ~ 4 with hard-coded data.

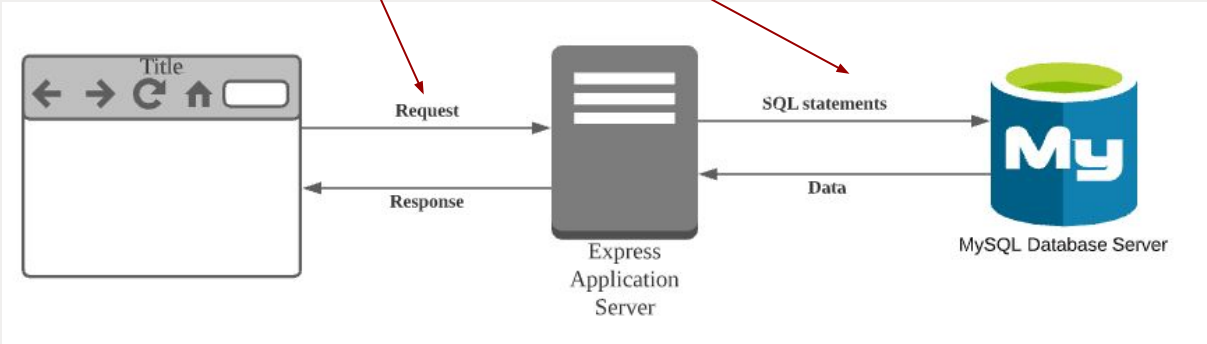
High Level Idea

Let's say if we need to support **a route to get a product with its id** like this:

```
router.get("/products/:id", function (req, res, next) {
  ....
});
```

The **underlying SQL statement** is:

```
select id, name, price from product where id = 1;
```



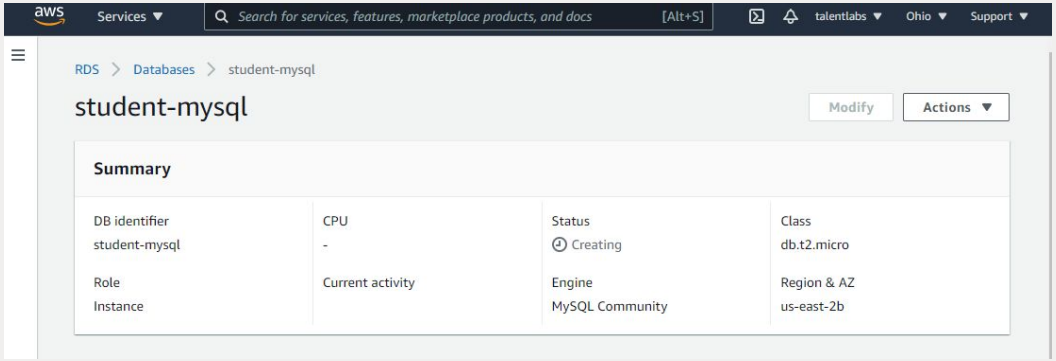
MySQL Database on the Cloud

</talentlabs>

Readings and References:
https://www.mysql.com/

Unlike SQLite, MySQL is a production ready database system. According to this website, it is the 2nd most popular database now.

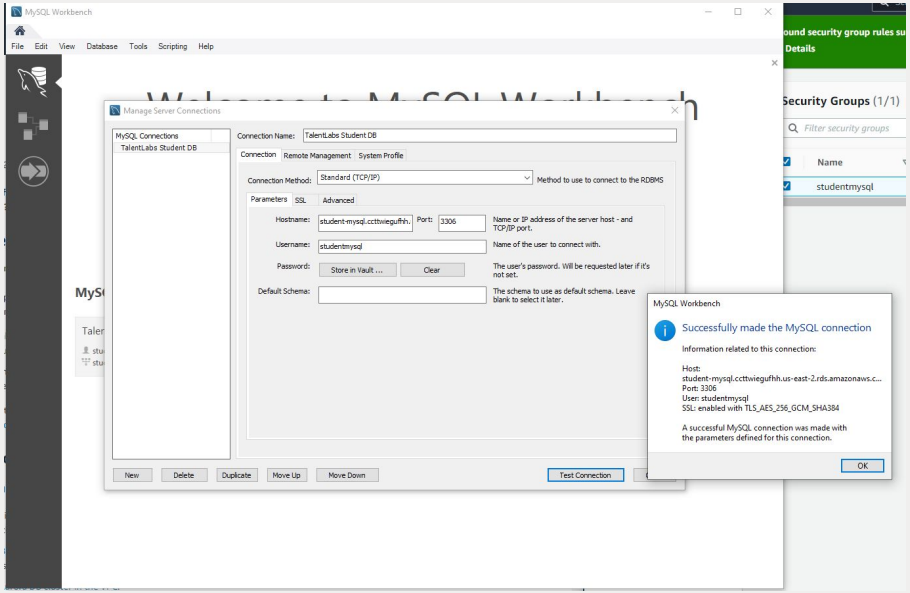
In this class we have set up a MySQL database on the cloud for you. Nowadays, most people will use a managed database from a cloud provider, because a database is a critical system. Cloud provider promises High availability of the database system.



MySQL Workbench

Download: https://www.mysql.com/products/workbench/

Connection details:
Host: student-mysql.ccttwiegufhh.us-east-2.rds.amazonaws.com
Port: 3306
Username: studentmysql
Password: studentmysql

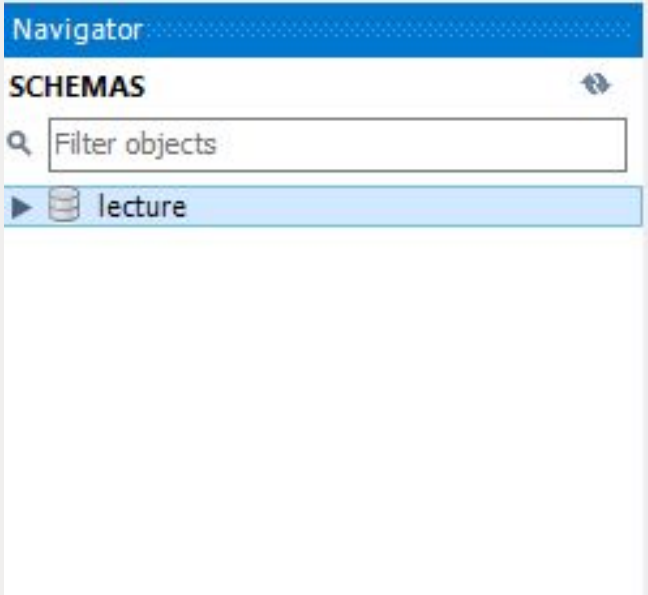


MySQL Workbench

Each student will work on their own schema, **don’t edit / delete other people's schema**.

If it is your first time connecting to this database, you can create a new schema for your project.

Right click your own schema and **set your own schema as default**.



Basic SQL Demo

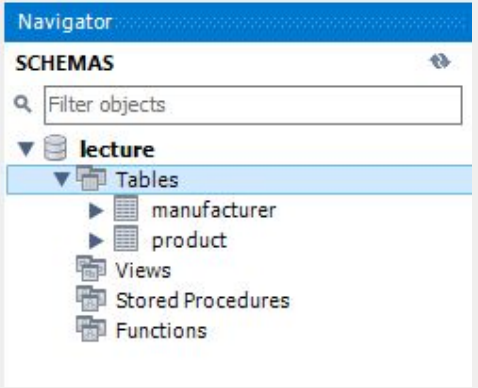
</talentlabs>

Create Table

Create a new SQL tab in the MySQL workbench and execute the following statements:

```
create table if not exists manufacturer (
      id int auto_increment primary key,
      name text
);


create table if not exists product (
    id int auto_increment primary key,
    name text,
    price decimal(19, 4),
    manufacturer_id int,
    foreign key (manufacturer_id) references manufacturer(id)
);
```



Insert Data

Create a new SQL tab in the MySQL workbench and execute the following statements:

```
insert into manufacturer (id, name)
values (1, "Lego"), (2, "Disney");

insert into product (id, name, price, manufacturer_id)
values (1, "Product 1", 99.9, 1), (2, "Product 2", 90.2, 2);
```

Query Data

Create a new SQL tab in the MySQL workbench and execute the following statements:

```
select * from manufacturer;

select * from product;
```

Filtering

Get the **product with id = 1:**

```
select * from product where id = 1;
```

Get the **a list products with manufacturer_id = 1:**

```
select * from product where manufacturer_id = 1;
```

Join Tables

```
select * from product
left join manufacturer
 on product.manufacturer_id = manufacturer.id;
```

Get the **product with id = 1 + the details of the manufacturer:**

```
select
 product.id as id,
 product.name as name,
 product.price as price,
 product.manufacturer_id as manufacturer_id,
 manufacturer.name as manufacturer_name
from product
left join manufacturer
 on product.manufacturer_id = manufacturer.id
where product.id = 1;
```

Get the **product with manufacturer_id = 1 + the details of the manufacturer:**

```
select
 product.id as id,
 product.name as name,
 product.price as price,
 product.manufacturer_id as manufacturer_id,
 manufacturer.name as manufacturer_name
from product
left join manufacturer
 on product.manufacturer_id = manufacturer.id
where product.manufacturer_id = 1;
```

Basic SQL Constraints

</talentlabs>

Primary Key

```
create table if not exists product (
    id int auto_increment primary key,
    name text,
    price decimal(19, 4),
    manufacturer_id int,
    foreign key (manufacturer_id) references manufacturer(id)
);
```

In the relational model of databases, a primary key is a specific choice of a minimal set of attributes (columns) that **uniquely specify a tuple (row) in a relation (table).**

If we try to insert records with duplicate id:
Error Code: 1062. **Duplicate entry '1' for key 'product.PRIMARY'**

```
insert into product (id, name, price, manufacturer_id)
values (1, "Product 1", 99.9, 1), (1, "Product 2", 90.2, 2);
```

Foreign Key Constraint

```
create table if not exists manufacturer (
     id int auto_increment primary key,
     name text
);

create table if not exists product (
    id int auto_increment primary key,
    name text,
    price decimal(19, 4),
    manufacturer_id int,
    foreign key (manufacturer_id) references manufacturer(id)
);
```

Foreign key is used to ensure the **data integrity (Referential integrity)**. A foreign key is a set of attributes in a table that **refers to the primary key of another table. The foreign key links these two tables.**

Because the database management system enforces referential constraints, it must **ensure data integrity if rows in a referenced table are to be deleted (or updated)**. If dependent rows in referencing tables still exist, those references have to be considered.

Foreign Key Constraint

Foreign key is used to ensure the **data integrity (Referential integrity)**. A foreign key is a set of attributes in a table that **refers to the primary key of another table. The foreign key links these two tables.**

Because the database management system enforces referential constraints, it must **ensure data integrity if rows in a referenced table are to be deleted (or updated)**. If dependent rows in referencing tables still exist, those references have to be considered.

Case 1: On insertion, make sure the related record exists.
Now, we have 2 **Manufacturers** in the databases, how about if we try to insert a product with manufacturer_id equal to a non existing manufacturer?

```
insert into product (id, name, price, manufacturer_id)
values (3, "Product 1", 99.9, 3);
```

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`lecture`.`product`, CONSTRAINT `product_ibfk_1` FOREIGN KEY (`manufacturer_id`) REFERENCES `manufacturer` (`id`))

Foreign Key Constraint

Foreign key is used to ensure the **data integrity (Referential integrity)**. A foreign key is a set of attributes in a table that **refers to the primary key of another table. The foreign key links these two tables.**

Because the database management system enforces referential constraints, it must **ensure data integrity if rows in a referenced table are to be deleted (or updated)**. If dependent rows in referencing tables still exist, those references have to be considered.

Case 2: On deletion, make sure the deletion won't break any relationships.

```
delete from manufacturer where id = 1;
```

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`lecture`.`product`, CONSTRAINT `product_ibfk_1` FOREIGN KEY (`manufacturer_id`) REFERENCES `manufacturer` (`id`))

Foreign Key Constraint

Foreign key is used to ensure the **data integrity (Referential integrity)**. A foreign key is a set of attributes in a table that **refers to the primary key of another table. The foreign key links these two tables.**

Because the database management system enforces referential constraints, it must **ensure data integrity if rows in a referenced table are to be deleted (or updated)**. If dependent rows in referencing tables still exist, those references have to be considered.

This is ok because we delete all relationships first:

```
delete from product where manufacturer_id = 1;
delete from manufacturer where id = 1;
```