Table of Contents

The Role of a Database in a Web Application ................................................................................. 2
Review of Database Terms and Concepts ....................................................................................... 3
Entity Relationship Diagrams ......................................................................................................... 5
  Types of Table Relationships ....................................................................................................... 5
SQL Commands Typically Used From a Web Application ............................................................... 6
  SQL INSERT Statement ................................................................................................................. 6
  SQL UPDATE Statement .............................................................................................................. 6
  SQL DELETE Statement .............................................................................................................. 6
  SQL SELECT Statement .............................................................................................................. 6
MySQL and MySQL Workbench ........................................................................................................ 7
SUMMARY ..................................................................................................................................... 8
The Role of a Database in a Web Application

- Web Pages often invoke Web APIs to run server side programs that can access a database.
- This JavaScript technique (of calling Web APIs) is called AJAX.
Review of Database Terms and Concepts

- A **database table** contains data about objects of a particular type or entity. Examples of entities are Customers, Products.
  - The **rows** in a table are called **records**.
  - The **columns** in a table are called **fields** or **attributes**.
  - Columns may have **constraints**. For example, when you define a column as being of data type “Date” (and it is a required field), this means that the database has a constraint that says, no record may be added unless it has a valid date in that field. If you specify another column as being 20 characters, the database will not allow a record to be added if it has more than 20 characters in that field.
  - Each table should have a **primary key**, which is one or more fields that uniquely identifies a record within the table. When you specify a primary key, you are setting up a constraint that says this table “shall not contain more than one record with the same primary key value as another record (in the same table)”.
  - A table may have zero or more **foreign keys** which “point to” another record (in the same or different table). When you specify a foreign key (with “integrity enforced”), you are setting up a constraint that says “no foreign key value can point to a non-existing record”.

- A **database** is a collection of related tables. When we design a database, we determine which tables need to be included for a given application, what fields or attributes are needed for each table, and how those tables should be related.

- A **database management system (DBMS)** is system software that encapsulates the database. There is no way to add, edit, delete, or even view the data unless you go through the database management system. You can think of the DBMS as the “bouncer” at a night club. Nobody unauthorized gets in. No actions (e.g., updates) are allowed if they would break the rules (constraints).
  - Small to mediums sized web applications tend to use MySQL as their DBMS.
  - Larger web apps might use Microsoft SQL Server or Oracle.
  - MSAccess is a “personal” DBMS that can also be shared and used for small applications.

As we specify the field types and specific relationships between tables, this creates **constraints** to be enforced by the Database management system. The DBMS ensures that
- only valid data goes in (e.g., valid dates in date fields, valid numbers in number fields),
- there are no duplicate primary key values in any table, and
- there are no invalid foreign key values (one record pointing to another record that does not exist).
The example below shows

- a database with two tables: an Employee table and a Department table.
- The Employee table contains 7 rows (also called records or objects), each representing a particular Employee.
- There are 8 columns in the Employee table (columns are also called fields or attributes) – so, that means we know 8 pieces of data (first name, last name, address, etc.) about each Employee in our table.
- In our example, Emp_ID has been specified as the primary key. Once a certain field (or fields) is specified as primary key, the DBMS will never allow two records with the same primary key value to be stored in a table.
- In our example, the Employee table has one foreign key field, Dept_ID that points to a record in the Department table. Once we tell the DBMS that a foreign key of one table points to the primary key of another table, the DBMS will ensure that there is never a pointer to nowhere.

<table>
<thead>
<tr>
<th>Employee Table</th>
<th>Department Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emp_ID</td>
<td>Last_name</td>
</tr>
<tr>
<td>1</td>
<td>Paquette</td>
</tr>
<tr>
<td>2</td>
<td>Johnson</td>
</tr>
<tr>
<td>4</td>
<td>Bradish</td>
</tr>
<tr>
<td>5</td>
<td>Braak</td>
</tr>
<tr>
<td>6</td>
<td>Shuster</td>
</tr>
<tr>
<td>7</td>
<td>Tate</td>
</tr>
</tbody>
</table>

Given the above data (with primary keys and foreign keys specified), the DBMS will prevent the following actions.

- Cannot add this record into the Employee table (Duplicate Primary Key). Emp_Id = 7 is already there.

<table>
<thead>
<tr>
<th>Emp_ID</th>
<th>Last_name</th>
<th>First_name</th>
<th>Employee_Adrs</th>
<th>Dept_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Smith</td>
<td>Mary</td>
<td>123 Any St</td>
<td>5</td>
</tr>
</tbody>
</table>

- Can’t add this record into the Employee table (Invalid Foreign Key). There is no Department with Dept_Id = 888

<table>
<thead>
<tr>
<th>Emp_ID</th>
<th>Last_name</th>
<th>First_name</th>
<th>Employee_Adrs</th>
<th>Dept_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Smith</td>
<td>Mary</td>
<td>123 Any St</td>
<td>888</td>
</tr>
</tbody>
</table>

- Cannot delete (would create invalid foreign key for several Employee Records that have Dept_ID = 5)

<table>
<thead>
<tr>
<th>Dept_ID</th>
<th>Dept_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Service</td>
</tr>
</tbody>
</table>

Using a Database concept called “JOIN”, you can extract data that LOOKS redundant data (see below), but there is no actual redundancy within the database (no copies of data stored).

```
SELECT Last_name, First_Name, Employee_Adrs, Employee.Dept_Id, Dept_Name
FROM Employee, Department WHERE Employee.Dept_ID = Department.Dept_ID ORDER BY Emp_ID;
```

<table>
<thead>
<tr>
<th>Last_name</th>
<th>First_name</th>
<th>Employee_Adrs</th>
<th>Dept_ID</th>
<th>Dept_Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paquette</td>
<td>Joe</td>
<td>2224 Web Blvd</td>
<td>6</td>
<td>Manufacturing</td>
</tr>
<tr>
<td>Johnson</td>
<td>Fred</td>
<td>410 Deer Ave</td>
<td>1</td>
<td>Marketing</td>
</tr>
<tr>
<td>Bradish</td>
<td>Emily</td>
<td>512 Plaza Place</td>
<td>5</td>
<td>Seattle</td>
</tr>
<tr>
<td>Braak</td>
<td>John</td>
<td>112 N Lincoln</td>
<td>6</td>
<td>Manufacturing</td>
</tr>
<tr>
<td>Shuster</td>
<td>George</td>
<td>5003 Country Ln</td>
<td>1</td>
<td>Marketing</td>
</tr>
<tr>
<td>Tate</td>
<td>Frank</td>
<td>603 Pinecrest Rd</td>
<td>5</td>
<td>Chehalis</td>
</tr>
</tbody>
</table>
Entity Relationship Diagrams

Most DMBS tools (e.g., oracle, MS SQL Server) provide a data modeling tool. MySQL has a data modeling tool, but we will not have time to use it. But I mention it because when you get a job, you’ll likely be exposed to data modeling diagrams.

- Each table is shown as a box. Each table name is capitalized.
- The table’s primary key field (PK) is underlined, with other fields shown below.
- Foreign Key (FK) fields show relationships to other tables (in this diagram, they are denoted with a star). For example, Order.Customer_ID points to Customer.Customer_ID, telling which customer placed the order. FKs are typically listed last.
- In some DB modeling tools, FKs are not even listed since you know there is an FK by virtue of the relationships shown (e.g., the line between Customer and Order).

Types of Table Relationships

We have so many Web Development topics to cover that we must have a very simple database. We do not have any Many To Many relationships, but here are the two main types of database relationships.

- One to Many, like Order to Customer
  - Each Order is made by exactly one Customer. A Customer can make more than one Order.
- Many to Many, like Product and Order
  - One Order can have many Products.
  - One Product can exist on many Orders.

To implement a many to many relationship, you need an associative table, such as the Order_Line table (above).
SQL Commands Typically Used From a Web Application

SQL INSERT Statement

```
INSERT INTO Authors (firstname, lastname) VALUES ('Sally', 'Kyvernitis');
```

- The INSERT will fail if any database constraint is violated, e.g., invalid data type, required field not supplied, record already exists with same primary key, or foreign key value does not exist in the table it references (if it references any other tables, which this example does not).
- This example assumes that the primary key is auto-increment and so is not supplied in INSERT.

SQL UPDATE Statement

```
UPDATE Authors SET lastname = 'Smith', firstname='Susan' WHERE authorid = 123;
```

- In the user interface of a web app, we typically update a particular record by its primary key, but SQL also allows for bulk updates, e.g.,
  ```
  UPDATE Stock SET price=price*1.05 WHERE dept='MENS';
  ```
- So, remember to add the WHERE clause or you’ll be updating all the records in the table.
- Like the INSERT statement, the UPDATE statement will fail if any database constraint is violated, e.g., invalid data type, required field not supplied, record already exists with same primary key, or foreign key value does not exist in the table it references (if it references any other tables, which this example does not).

SQL DELETE Statement

```
DELETE FROM Authors WHERE authorid = '123';
```

- In web app user interfaces, this is typically the only kind of delete we do – find a particular record by its primary key, have the user confirm that this is record they REALLY want to delete, then delete it.
- Don’t forget the Where clause or ALL records from the Authors table will be deleted.
- BUT the DBMS will only allow the delete only if no other records are pointing to the record that is about to be deleted. If the intent is to truly delete this record even though other records are pointing to this record, then the programmer should first delete all those other records, THEN delete the user’s record.

SQL SELECT Statement

A SELECT statement was explained above (employee joined with department). Here are some more examples:

```
SELECT * FROM Titles;
SELECT Title, ISBN FROM Titles;
SELECT * FROM Titles WHERE ISBN = '12345';
SELECT * FROM Titles WHERE Title like 'Cat%' order by Title ASC;
SELECT * FROM Titles, AuthorISBN, Author WHERE
   Title.ISBN = AuthorISBN.ISBN   AND
   AuthorISBN.authorID = Authors.AuthorID   AND
   Author.lastName = 'Feehan';  (called “inner join”)
```
MySQL and MySQL Workbench

- MySQL is an open source Database Management System (DBMS) that is typically offered by low cost web hosts. MySQL is the DBMS of choice for most small to medium sized web applications.
- MySQL Workbench is open source software that provides developers with a GUI so that they do not have to type raw SQL into MySQL.

Without MySQL Workbench, the Developer would have to use telnet to remote login to the database server and type in raw SQL commands.

With MySQL Workbench, the Developer can point/click and generated SQL is sent to the DBMS.
SUMMARY

- You learned about the role of a database in a web application – what happens when a Web API is invoked by the browser.
- You learned basic Database terminology, including Primary Key, Foreign Key, and Table relationships.
- You learned that a Database Management System (DBMS) is SW that provides the ONLY access to a database.
  - It enforces constraints, e.g., data type, PKs are unique, FKs point to valid PKs
  - It controls access only to authorized users
- You saw an example of related database tables (employee, department) and how we want to design the database with no redundancy. We can get the “redundancy” back by joining data from two related tables, but we try never to store anything that can be calculated (to avoid ambiguity).
- You saw an example of a Entity Relationship Diagram and learned about different types of relationships between database tables.
- You saw examples of basic SQL statements (SELECT, INSERT, UPDATE, DELETE)
- You learned about MySQL (DBMS) and MySQL Workbench (developer’s GUI for MySQL).