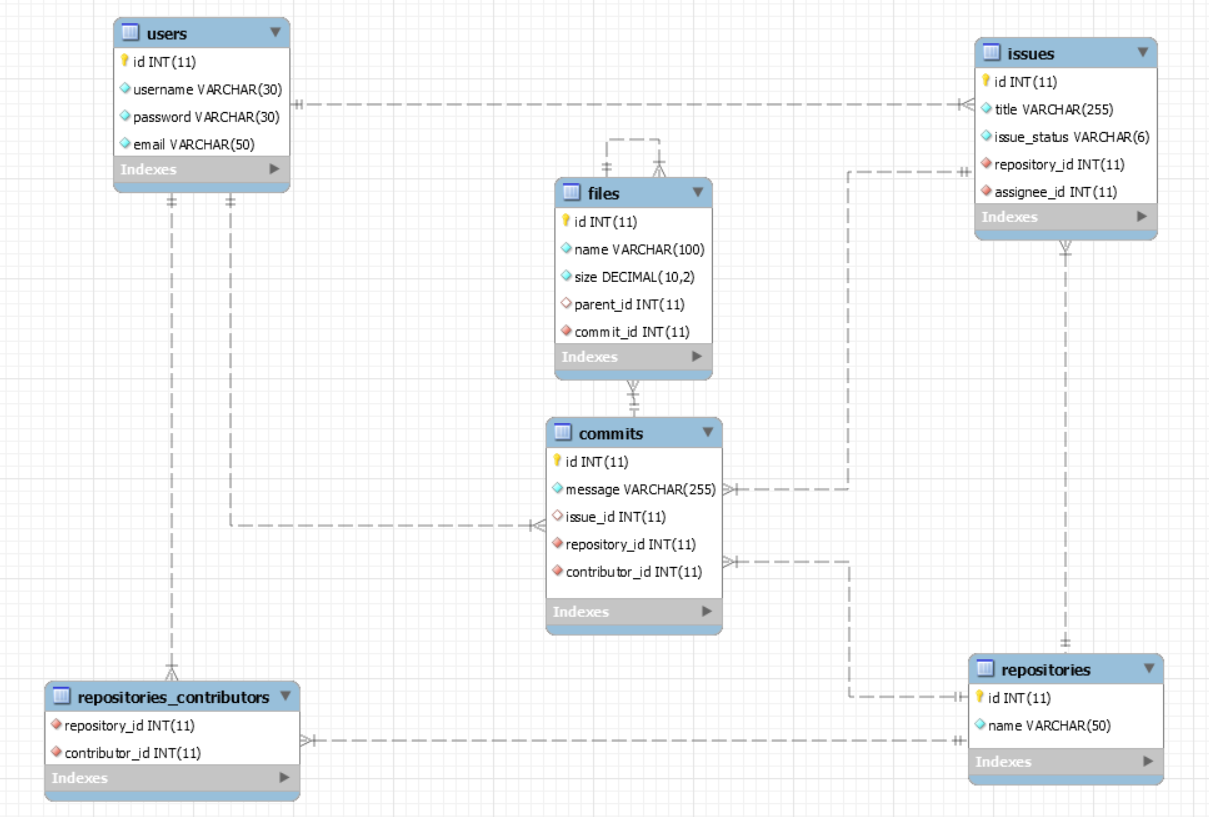
# Database Basics (MySQL) Exam Buhtig Source Control

You’ve most likely heard of Github. Well … There is a side project called “Buhtig” which is the back-up data of Github. You are one of the few selected to work in the multi-billion company, as one of the back-up database managers. You’ll need to prove your skills by designing and manipulating data in the Instagraph prototype.

## Section 0: Database Overview

You have been given an Entity / Relationship Diagram of the Buhtig Database:

****

The **Buhtig Database** needs to hold information about **users**, **repositories**, **issues**, **commits** & **files**.

Your task is to create a database called buhtig. Then you will have to create several **tables**.

* users – contains information about the **users**.
* repositories – contains information about the **repositories**.
* repositories\_contributors – a **many** to **many** **mapping** table between the **repositories** and the **users**.
* issues – contains information about the **issues**.
  + Each issue has a repository.
  + Each issue has an assignee (user).
* commits – contains information about the **commits**.
  + Each commit **MAY** have an issue.
  + Each commit has a repository.
  + Each commit has a contributor (user).
* files – contains information about the files.
  + Each file MAY have a parent (file).
  + Each file has a commit.

## Section 1: Data Definition Language (DDL) – 40 pts

Make sure you implement the whole database correctly on your local machine, so that you could work with it.

The instructions you’ll be given will be the minimal needed for you to implement the database.

### Table Design

You have been tasked to create the tables in the database by the following models:

**users**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| username | A **string** containing a maximum of **30 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |
| password | A **string** containing a maximum of **30 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| email | A string containing a maximum of **50 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |

**repositories**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| name | A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |

**repositories\_contributors**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| repository\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table repositories. |
| contributor\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table users. |

**issues**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| title | A **string** containing a maximum of **255 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted. |
| issue\_status | A **string** containing a maximum of **6 characters**.  Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| repository\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table repositories.  **NULL** is **NOT** permitted**.** |
| assignee\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table users.  **NULL** is **NOT** permitted**.** |

**commits**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| message | A **string** containing a maximum of **255 characters**.  Unicode is **NOT** needed. | **NULL** is **NOT** permitted. |
| issue\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table issues. |
| repository\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table repositories.  **NULL** is **NOT** permitted**.** |
| contributor\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table users.  **NULL** is **NOT** permitted**.** |

**files**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| name | A **string** containing a maximum of **100 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted. |
| size | **DECIMAL**, up to **10 digits**, **2** of which after the **decimal point**. | **NULL** is **NOT** permitted**.** |
| parent\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table files. |
| commit\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table commits.  **NULL** is **NOT** permitted**.** |

Submit your solutions in Judge on the first task. Submit **all** SQL table creation statements.

You will also be given a data.sql file. It will contain a **dataset** with random data which you will need to **store** in your **local database**. This data will be given to you so you will not have to think of data and lose essential time in the process. The data is in the form of **INSERT** statement queries.

## Section 2: Data Manipulation Language (DML) – 30 pts

Here we need to do several manipulations in the database, like changing data, adding data etc.

### Data Insertion

You will have to **INSERT** records of data into the issues table, based on the files table. For files with id between **46** and **50** (**inclusive**), **insert data** in the issues table with the **following values**:

* title –set it to **“**Critical Problem With {fileName}!**”.** Where the fileName is the name of the file.
* issue\_status – set it to “open”.
* repository\_id – **MULTIPLY** the id of the file by **2** and **DIVIDE** it by **3**.
  + **ROUND** the resulting value **UP**.
* assignee\_id – the file’s commit’s contributor’s id**.**

### Data Update

**UPDATE** all contributors to repositories which have the **same** id (**value**) as the repository they **contribute** to.

**SET** them as a contributor to the repository with the **lowest** id (by **value**) which has **no contributors**.

If there aren’t any repositories with no contributors do nothing.

### Data Deletion

Buhtig is all about activity, and activity is expressed in issues. Issues indicate the constant process of development. Naturally, inactive repositories are being treated as abandoned. **DELETE** all repositories which do **NOT** **have any** issues.

## Section 4: Programmability – 30 pts

The time has come for you to prove that you can be a little more dynamic on the database. So, you will have to write several procedures.

### Commit

Create a stored procedure udp\_commit which accepts the following parameters:

* username
* password
* message
* issue\_id

And checks the following things:

If the username does **NOT exist** in the users table:

Throw an exception with error code ‘45000’ and message ‘No such user!’.

If the password does **NOT** match the username in the users table:

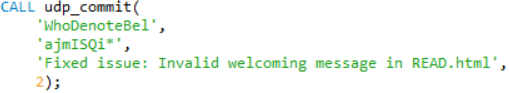
Throw an exception with error code ‘45000’ and message ‘Password is incorrect!’.

If there is no issue with the given id in the issues table:

Throw an exception with error code ‘45000’ and message ‘The issue does not exist!’.

If **all checks pass**, extract the id of the corresponding user, from the users table, then the repository\_id of the issue, from the issues table, and **INSERT** a new commit into the commits table with the **extracted data**.

The **procedure** should also **update** the issue’s status to ‘closed’.



#### Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **message** | **issue\_id** | **repository\_id** | **contributor\_id** |
| ... | ... | ... | ... | ... |
| 51 | Fixed Issue: Invalid welcoming message in READ.html | 2 | 34 | 6 |

### Filter Extensions

Create a stored procedure udp\_findbyextension which accepts the following parameters:

* extension

And extracts all files that **have** the **given** extension. (like index.html for example)

The procedure should **extract** the file’s id, name and size.

The file’s size should have “KB” attached to it as a **suffix**.

The files should be ordered **ascending** by file id.



#### Result

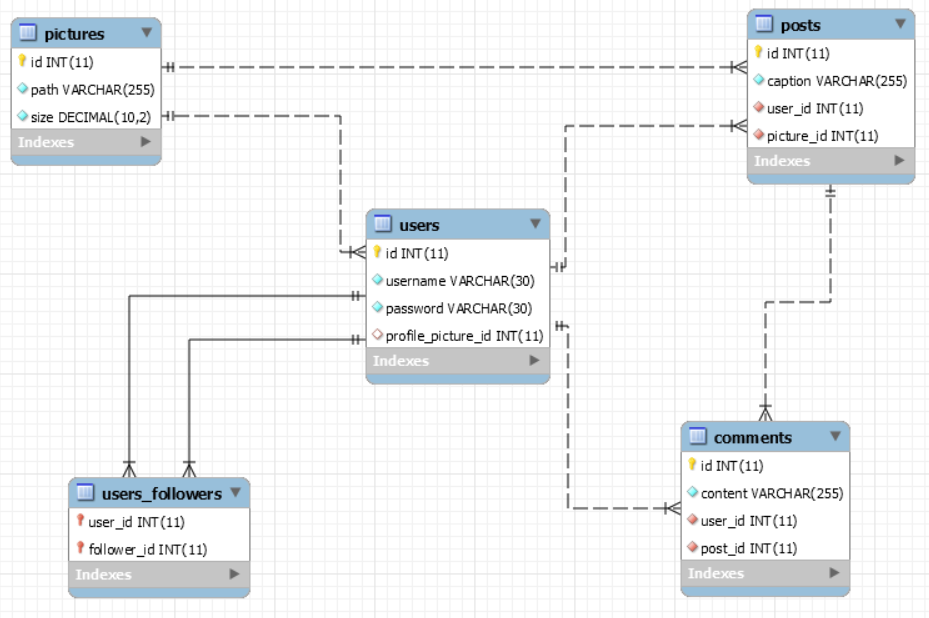
|  |  |  |
| --- | --- | --- |
| **id** | **caption** | **user** |
| 13 | Beat.html | 907.30KB |
| 17 | Login.html | 2863.23KB |
| ... | ... |  |

# Database Basics (MySQL) Retake Exam Instagraph

You’ve most likely heard of Instagram. Well … There is a side project called “Instagraph” which is the back-up data of Instagram. You are one of the few selected to work in the multi-billion company, as one of the back-up database managers. You’ll need to prove your skills by designing and manipulating data in the Instagraph prototype.

## Section 0: Database Overview

You have been given an Entity / Relationship Diagram of the Instagraph Database:

****

The Instagraph Database needs to hold information about pictures, users, posts and comments.

Your task is to create a database called instagraph\_db. Then you will have to create several **tables**.

* pictures – contains information about the **pictures**.
* users – contains information about the **users**.
  + Each user may have a profile picture.
* posts – contains information about the **posts**.
  + Each post has a user.
  + Each post has a picture.
* comments – contains information about the **comments**.
  + Each comment has a user.
  + Each comment has a post.
* users\_followers – a **many** to **many** table connected to the **users**.

## Section 1: Data Definition Language (DDL) – 40 pts

Make sure you implement the whole database correctly on your local machine, so that you could work with it.

The instructions you’ll be given will be the minimal needed for you to implement the database.

### Table Design

You have been tasked to create the tables in the database by the following models:

**pictures**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| path | A **string** containing a maximum of **255 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| size | **Decimal**, **up** to **10 digits**, **2** of which after the **decimal point**. | **NULL** is **NOT** permitted**.** |

**users**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| username | A **string** containing a maximum of **30 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |
| password | A **string** containing a maximum of **30 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| profile\_picture\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table pictures. |

**posts**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| caption | A **string** containing a maximum of **255 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted. |
| user\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table users.  **NULL** is **NOT** permitted**.** |
| picture\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table pictures.  **NULL** is **NOT** permitted**.** |

**comments**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| content | A **string** containing a maximum of **255 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted. |
| user\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table users.  **NULL** is **NOT** permitted**.** |
| post\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table posts.  **NULL** is **NOT** permitted**.** |

**users\_followers**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| user\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table users. |
| follower\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table users. |

Submit your solutions in Judge on the first task. Submit **all** SQL table creation statements.

You will also be given a data.sql file. It will contain a **dataset** with random data which you will need to **store** in your **local database**. This data will be given to you so you will not have to think of data and lose essential time in the process. The data is in the form of **INSERT** statement queries.

## Section 2: Data Manipulation Language (DML) – 30 pts

Here we need to do several manipulations in the database, like changing data, adding data etc.

### Data Insertion

You will have to **INSERT** records of data into the comments table, based on the posts table. For posts with id between **1** and **10**, insert data in the comments table with the following values:

* content –set it to **“**Omg!{name}!This is so cool!**”.** Where the name is the username of the user that **posted** the post.
* user\_id – **MULTIPLY** the id of the post by **3** and **DIVIDE** it by **2**.
  + **ROUND** the resulting value **UP**.
* post\_id – the post’s id**.**

### Data Update

**UPDATE** all users which do **NOT** have a profile picture. **Set** their profile picture id to the **count** of **followers** they have. If they have 0, set it to the user’s id.

### Data Deletion

Naturally, unpopular profiles are being treated as abandoned. **DELETE** all users which do **NOT** **follow** anyone and **no one follows** them.