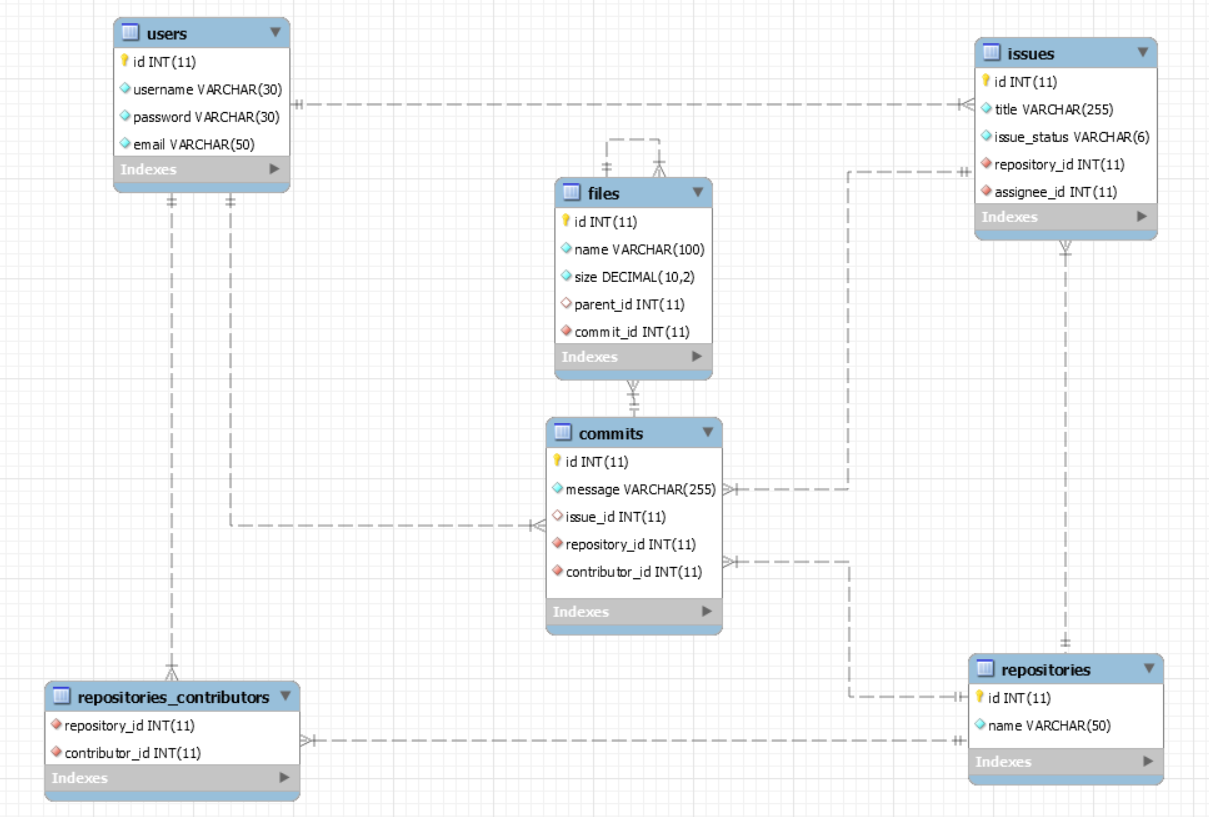
# Database Basics (MySQL) Exam Buhtig Source Control

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 3: Querying – 100 pts

And now we need to do some data extraction. **Note** that the **example results** from **this section** use a **fresh database**. It is **highly recommended** that you **clear** the **database** that has been **manipulated** by the **previous problems** from the **DML** **section** and **insert again** the **dataset** you’ve been given, to ensure **maximum consistency** with the **examples** given in this section.

### Users

Extract from the database, all of the users.

**ORDER** the results **ascending** by user id.

#### Required Columns

* id (users)
* username

#### Example

|  |  |
| --- | --- |
| **id** | **username** |
| 1 | UnderSinduxrein |
| ... | ... |

### Lucky Numbers

When a contributor has the same id as the repository he contributes to, it’s a lucky number.

**Extract** from the database, all of the repositories, which have the **same id** as their contributor.

**ORDER** the results **ascending** by repository id.

#### Required Columns

* repository\_id
* contributor\_id

#### Example

|  |  |
| --- | --- |
| **repository\_id** | **contributor\_id** |
| 1 | 1 |
| 3 | 3 |
| ... | ... |

### Heavy HTML

There are some pretty big HTML files in the Buhtig database… Unnaturally big. Extract from the database all of the files, which have size, **GREATER** than **1000**, and their name contains “html”.

**ORDER** the results **descending** by size.

#### Required Columns

* id (files)
* name
* size

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **name** | **size** |
| 49 | compile.html | 27402.59 |
| ... | ... | ... |

### Issues and Users

Extract from the database, all of the issues, and the users that are **assigned** to them, so that they end up in the following format:

{username} : {issueTitle}

**ORDER** the results **descending** by issue id.

#### Required Columns

* id (issues)
* issue\_assignee

#### Example

|  |  |
| --- | --- |
| **id** | **issue\_assignee** |
| 75 | TheDivineBel : Critical bug in Controller.php ruins application when executed |
| ... | ... |

### Non-Directory Files

Some of the files are **Directories**, because they are a **parent** to **some file**. Try to find those, which aren’t.

Extract from the database all of the files, which are **NOT** a parent to **any other file**.

Extract the size of the file and add “KB” to the **end** of it.

**ORDER** the results **ascending** by file id.

#### Required Columns

* id (files)
* name
* size

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **Name** | **size** |
| 6 | Controller.json | 14034.87KB |
| ... | ... | ... |

### Active Repositories

Extract from the database, the **top 5** repositories, in terms of count of issues on them.

**ORDER** the results **descending** by issues (count of issues), and **ascending** by repository id.

#### Required Columns

* id (repositories)
* name (repositories)
* issues (count of issues)

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **name** | **issues** |
| 11 | KartinaJS | 5 |
| ... | ... | ... |

### Most Contributed Repository

Extract from the database, the **top 1 repository** in terms of count of contributors.

If there are 2 repositories have the **same** count of contributors, **order** them **ascending**, by id.

#### Required Columns

* id (repositories)
* name (repositories)
* commits (count of commits)
* contributors (count of contributors)

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **commits** | **contributors** |
| 22 | Maxima | 1 | 6 |

### Fixing My Own Problems

Extract from the database, for every user – the count of commits he has on issues that were **assigned** to **him**.

**ORDER** the results **descending** by commits (count of commits), and **ascending** by user id.

#### Required Columns

* id (users)
* username
* commits (count of commits)

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **username** | **commits** |
| 1 | UnderSinduxrein | 1 |
| ... | ... | ... |

### Recursive Commits

**Extract** from the **database** all files which **are a** parent to **their** parent.

In other words, file “a” is a parent to file “b” and file “b” is a parent to file “a”.

Extract the file name (but only the name, without the extension). If its “index.html” you have to extract “index”, as “file”.

Extract the count of commits which **hold** the full file name (with extension) in their messages as “recursive\_count”.

**ORDER** the results **ascending** by file (file name).

#### Required Columns

* file (fileName)
* recursive\_count

#### Example

|  |  |
| --- | --- |
| **file** | **recursive\_count** |
| Find | 2 |
| ... | ... |

### Repositories and Commits

Extract from the database, for every repository – the count of users that have committed to it.

**NOTE**: **1** user may have **more** than **1** commit on the repository.

**ORDER** the results **descending** by users (count of users), and **ascending** by repository id.

#### Required Columns

* id (repositories)
* name
* users (count of users)

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **name** | **users** |
| 1 | WorkWork | 4 |
| ... | ... | ... |

## 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:

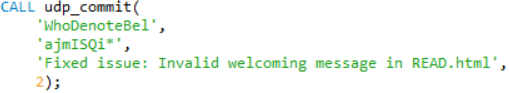
Throw an exception with error code ‘45000’ and message ‘Password is incorrect!’.

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 |
| ... | ... |  |