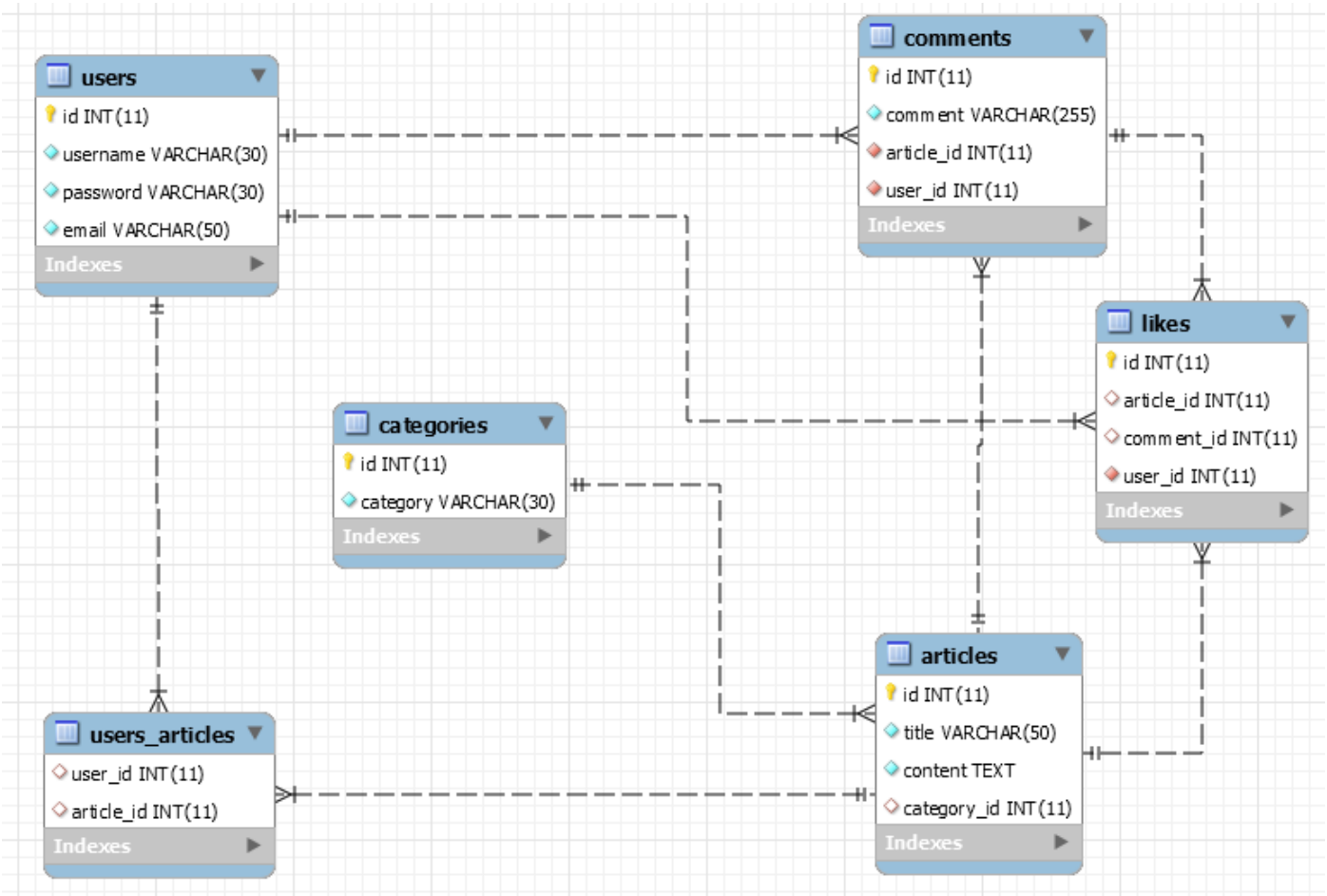
MySQL Retake Exam

Colonial Blog Database

After the successful Colonial Journey to the SoftUnia Galaxy and the success of the management system the Council has started a new Colonial Blog and your task is to create the Colonial Blog Database.

1. Section: Database Overview

You have given and Entity / Relationship Diagram of the Colonial Blog Database:



The Colonial Blog Database holds information about users, their articles, information about the article categories, likes and comments. Your task is to create a database called **colonial_blog_db**. Then you will have to create several **tables**.

- **users** – contains information about **users**.
- **categories** – contains information about **categories**.
- **articles –** contains information about **articles**.
- **users_articles –** mapping table between **users** and **articles**.
- **comments –** contains information about **comments**.
- **likes –** contains information about **likes**.

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

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

2. Section: Data Definition Language (DDL) – 40pts

1. 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.

categories

Column Name	Data Type	Constraints
`id`	**Integer**, from **1** to **2,147,483,647.**	**Primary Key** **AUTO_INCREMENT**
`category`	A **string** containing a maximum of **30 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted.

articles

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 **50 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted.
`content`	A **string** containing more than **255 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted.
`category_id`	**Integer**, from **1** to **2,147,483,647.**	Relationship with table **categories**.

users_articles

Column Name	Data Type	Constraints
`user_id`	**Integer**, from **1** to **2,147,483,647.**	Relationship with table **users**.
`article_id`	**Integer**, from **1** to **2,147,483,647.**	Relationship with table **articles**.

comments

Column Name	Data Type	Constraints
`id`	**Integer**, from **1** to **2,147,483,647.**	**Primary Key** **AUTO_INCREMENT**
`comment`	**A string** containing a maximum of **255 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted.
`article_id`	**Integer**, from **1** to **2,147,483,647.**	Relationship with table **articles.** **NULL** is **NOT** permitted.
`user_id`	**Integer**, from **1** to **2,147,483,647.**	Relationship with table **users.** **NULL** is **NOT** permitted.

likes

Column Name	Data Type	Constraints
`id`	**Integer**, from **1** to **2,147,483,647.**	**Primary Key** **AUTO_INCREMENT**
`article_id`	**Integer**, from **1** to **2,147,483,647.**	Relationship with table **articles.**
`comment_id`	**Integer**, from **1** to **2,147,483,647.**	Relationship with table **comments.**
`user_id`	**Integer**, from **1** to **2,147,483,647.**	Relationship with table **users.** **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.

3. Section: Data Manipulation Language (DML) – 30 pts

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

2. Data Insertion

You will have to **INSERT** records of data into the **`likes`** table, based on the **users** table.

For users with **id** between **16** and **20(inclusive)**, **insert data** in the **likes** table with the following values:

- For **users** with **even id**, the **like** will be on an **article**, else – **comment**.
- **Users' username length** will determine the **article_id**.
- **Users' email length** will determine the **comment_id**.

3. Data Update

UPDATE comments with id between **1** and **15(inclusive)** and meet the following conditions:

- If the comment's **id** is dividable by **2** without remainder – **'Very good article.'**.
- If the comment's **id** is dividable by **3** without remainder – **'This is interesting.'**.
- If the comment's **id** is dividable by **5** without remainder – **'I definitely will read the article again.'**.
- If the comment's **id** is dividable by **7** without remainder – **'The universe is such an amazing thing.'**.

4. Data Deletion

The Council does not like **articles** without **category**. Delete all **articles** without **category**.

4. Section: Querying – 50 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.

5. Extract 3 biggest articles

Extract from the database, the **3 biggest articles** and summarize their content. The **summary** must be **20 symbols long plus "..."** at the end. Order the results by **article id**.

Required Columns

- **title**
- **summary**

Example

title	summary
She Wants Revenge	She Wants Revenge is...
Montana gubernatorial election, 1988	The 1988 Montana gub...
Jackie Torrens	Jackie Torrens (born...

6. Golden Articles

When article has the same id as its author, it is considered Golden Article. Extract from the database **all golden articles**. Order the results ascending by article id.

Required Columns

- article_id
- title

Example

article_id	title
1	John Hyrcanus
3	Denmark in the Eurovision Song Contest 1988
...	...

7. Extract categories

Extract from the database, **all categories** with their **articles, and likes. Order** them by count of **likes descending,** then by **article's count descending** and lastly by **category's id ascending.**

Required Columns

- category
- articles (count of articles for the given category)
- likes (total likes for the given category)

Example

category	articles	likes
Animals	5	7
Nature	7	5
…	…	…

8. Extract the most commented Social article

Extract from the database, the **most commented social article with the number of comments**.

Required Columns

- title
- comments (total articles comments)

Example

title	comments

Metropolitan Police Clubs and Vice Unit	4

9. Extract the less liked comments

Extract from the database those **comments** that are **not** liked by anyone and **summarize** them and order the results by **comment id in descending order. The summary must be 20 symbols long plus "..." at the end.**

Required Columns

- **summary**

Example

summary
tincidunt eu felis f...
id ornare imperdiet ...
...

5. Section: Programmability – 30 pts

10. Get user's articles count

Create a **user defined function** with the name **`udf_users_articles_count(username VARCHAR(30))`** that receives a **username** and returns the number of articles this user has written.

Example

Query

```
SELECT u.username, udf_users_articles_count('UnderSinduxrein')
AS count
FROM articles AS a
JOIN users_articles ua
ON a.id = ua.article_id
JOIN users u
ON ua.user_id = u.id
WHERE u.username = 'UnderSinduxrein'
GROUP BY u.id;
```

name	count
UnderSinduxrein	13

11. Like article

Create a **user defined stored procedure** with the name **`udp_like_article(username VARCHAR(30), title VARCHAR(30))`** that receives a **username** and **article title** and likes the article **only if** the given username and title **exist**. If the modifying is not successful **rollback** any changes and throw an **exception** with **error code** **'45000'** and **message**: "Non-existent user." or "Non-existent article.".

Example

Query

```
CALL udp_like_article('Pesho123', 'Donnybrook, Victoria');
```

Response

Non-existent user.

Query

```
CALL udp_like_article('BlaAntigadsa', 'Na Pesho statiqta');
```

Response

Non-existent article.

Query

```
CALL udp_like_article('BlaAntigadsa', 'Donnybrook, Victoria');
SELECT a.title, u.username
FROM articles a
JOIN likes l
ON a.id = l.article_id
JOIN users u
ON l.user_id = u.id
WHERE u.username = 'BlaAntigadsa' AND a.title = 'Donnybrook, Victoria';
```

title	username
Donnybrook, Victoria	BlaAntigadsa