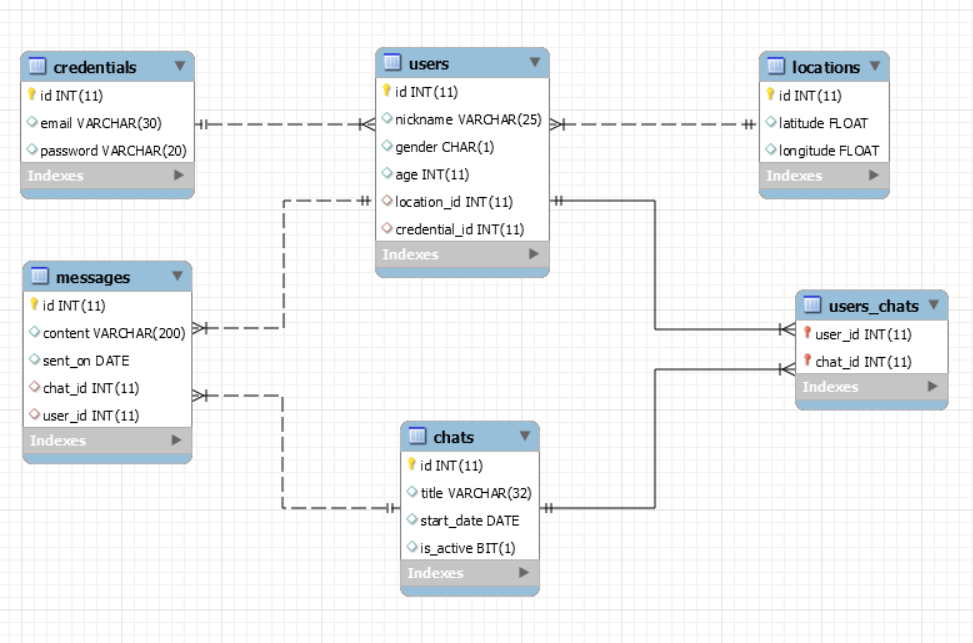
# Database Basics (MySQL) Exam Preparation I The Nerd Herd

That is the name of the newest super-hot awesome chat application! You are part of the developing team and your job is to manage the database underneath – creating the database, manipulating data, adding programmability functionality etc. You can see more details below.

## Section I: Data Definition Language (DDL) – 30 pts

### Database Overview

You have been given the E/R diagram for the database of the application:



Your task is to create a database called the\_nerd\_herd. Then you will have to create **6 tables**.

* users – contains information about the users, registered in the application.
  + Each user has a **location**.
  + Each user has **exactly one** **credential**.
* credentials - contains information about the credentials of a user (password, email etc.)
* location – contains geographical information about the location of each person.
* chats – contains information about the chats.
* users\_chats – a mapping table between the users and the chats.
  + Chats can have **many users** participating in them.
  + Users can participate in **many chats**.
* messages – contains information about the messages sent from the users in the chats.

### 01. Table Design

You have been given the following requirements for the design of the tables. Implement each table exactly as described, with the given names, data types, constraints etc. Make sure you do it, exactly as described.

**users**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **0** to **2,147,483,647** | Unique table identificator, Identity |
| nickname | **String**, up to **25** symbols |  |
| gender | **Character**, up to **1** symbols |  |
| age | **Integer**, from **0** to **2,147,483,647** |  |
| location\_id | **Integer**, from **0** to **2,147,483,647** | Relationship with table locations |
| credential\_id | **Integer**, from **0** to **2,147,483,647** | Relationship with table credentials,  Unique values |

**locations**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **0** to **2,147,483,647** | Unique table identificator |
| latitude | **Floating-point** number |  |
| longitude | **Floating-point** number |  |

**credentials**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **0** to **2,147,483,647** | Unique table identificator |
| email | **String**, up to **30** symbols |  |
| password | **String**, up to **20** symbols |  |

**chats**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **0** to **2,147,483,647** | Unique table identificator |
| title | **String**, up to **32** symbols |  |
| start\_date | **Date**, **WITHOUT** time |  |
| is\_active | **Bit** |  |

**messages**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **0** to **2,147,483,647** | Unique table identificator |
| content | **String**, up to **200** symbols |  |
| sent\_on | **Date**, **WITHOUT** time |  |
| chat\_id | **Integer**, from **0** to **2,147,483,647** | Relationship with table chats |
| user\_id | **Integer**, from **0** to **2,147,483,647** | Relationship with table users |

**users\_chats**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| user\_id | **Integer**, from **0** to **2,147,483,647** | Unique table identificator, Relationship with table users |
| chat\_id | **Integer**, from **0** to **2,147,483,647** | Unique table identificator, Relationship with table chats |

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

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

In this section you will have to perform some data manipulations on your newly created database. Obviously, it is currently empty, so you will have to fill the data before you can manipulate it. The employers have provided a data.sql file which contains the data you must import in your database in order, to be able to work on this task.

### 02. Insert

Do you remember **ASL**? It stands for **Age**, **Sex**(gender), **Location**. You have been tasked to **insert** a couple of **messages** based on the userstable.

In other words, for **every record** in the users table, in a certain **range**, you must **insert** a **record** in the messages table. Each message should be formed by the **data** of the **corresponding** user.

The messages should be formed in the following way:

* content – **concatenation** of the age, gender, latitude, longitude, split by a **dash** (‘-’).
* sent\_on – should be the date ‘2016-12-15’.
* chat\_id – should be the **FINAL result** of the **calculations**, depending on several conditions:
  + If the corresponding **user** is a **Female** (user’s gender = ‘F’), you should **multiply** his age by **2** and take the **square root** of the **resulting value**.
  + If the corresponding **user** is a **Male** (user’s gender = ‘M’), you should **divide** his age by **18** and get the **resulting value** to the **power** of **3**. (**value ^ 3**).
* user\_id – should be **equal** to the **user’s** id.

NOTE: Messages, should be inserted, for **all users** whose id is between **10** and **20**, **inclusively**.

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **content** | **sent\_on** | **chat\_id** | **user\_id** |
| 56-F-10.1857-123.734 | 2016-12-15 | 11 | 12 |

### 03. Update

The back-end developers have slightly messed up. Due to their mistake, some chats have messages which were **sent on** a **date**, **EARLIER** than the start\_date of their chat. Guess what? You’ll have to fix that.

For all chats, which have messages before their start\_date.

UPDATE the start\_date’s value, so that it **equals** the sent\_on date of the **EARLIEST** message in that chat.

|  |  |  |  |
| --- | --- | --- | --- |
| **chat\_id** | **start\_date** | **message\_id** | **sent\_on** |
| 4 | 2013-10-28 | 53 | 2011-06-22 |

|  |  |  |  |
| --- | --- | --- | --- |
| **chat\_id** | **start\_date** | **message\_id** | **sent\_on** |
| 4 | 2011-06-22 | 53 | 2011-06-22 |

### 04. Delete

For this task, you should DELETE some **non-needed** data from the database.  
Remove all locations which **do NOT** have a **user** located there.

## Section 3: Querying – 100 pts

It is recommended that you use a clear database for this task. Drop all the data and insert it again in order to ensure maximum consistency with the examples in the problem descriptions. Use the data.sql file again to fill your database.

The Seniors want to test your ability to extract valuable data from the database, so they have given you a few tasks. You will have to extract records from tables, merge table results, filter table results, and extract specified columns.

### 05. Age Range

Extract all users that are **aged** between **22** and **37**, **inclusively**.

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

#### Required columns

* nickname
* gender
* age

#### Example

|  |  |  |
| --- | --- | --- |
| **nickname** | **gender** | **age** |
| sbell0 | F | 23 |
| ... | ... | ... |

### 06. Messages

Extract all messages that are sent **AFTER 12.05.2014** and contain the word “just”.

**ORDER** the results by the message id in **descending** order.

#### Required columns

* content
* sent\_on

#### Example

|  |  |
| --- | --- |
| **content** | **sent\_on** |
| odio cras mi pede malesuada in imperdiet et commodo vulputate justo in | 2014-07-30 |
| ... | ... |

### 07. Chats

Extract all chats that are **NOT** **ACTIVE** and their **title length** is **less than** **5**, **OR** the **3rd** and **4th** **letters** are equal to “tl”.

**ORDER** the results by title in **descending** order.

#### Required columns

* title
* is\_active

#### Example

|  |  |
| --- | --- |
| **title** | **is\_active** |
| Viva | 0 |
| ... | ... |

### 08. Chat Messages

Extract all chats with messages sent **BEFORE 26.03.2012** and chat title with **last letter** equal to “x”.

**ORDER** the results by chat id and message id in **ascending** order.

#### Required columns

* id (chats)
* tittle
* id (messages)

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **tittle** | **id** |
| 5 | Trippledex | 72 |
| ... | ... | ... |

### 09. Message Count

Extract all chats and the **amount** of messages they have. Some messages may **NOT** have a chat.

**FILTER** messages with id **less** than **90**.

**SELECT** only the **first 5** results.

**ORDER** the results by total\_messages in **descending** order and chat id in **ascending** order.

#### Required columns

* id (chats)
* total\_messages

#### Example

|  |  |
| --- | --- |
| **id** | **total\_messages** |
| 37 | 4 |
| ... | ... |

### 10. Credentials

Extract all users with emails **ENDING** with “co.uk”.

**ORDER** the results by email in **ascending** order.

#### Required columns

* nickname
* email
* password

#### Example

|  |  |  |
| --- | --- | --- |
| **nickname** | **email** | **password** |
| vburkek | mperkinst@amazon.co.uk | lCFO0hSeRt |
| ... | ... | ... |

### 11. Locations

Extract all users who do **NOT** have a location.

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

#### Required columns

* id (users)
* nickname
* age

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **nickname** | **age** |
| 11 | ahunta | 63 |
| ... | ... | ... |

### 12. Left Users

Extract all messages sent from users who have **LEFT** the chat (they are **NOT** in the **corresponding** to the **message** – **CHAT** anymore).

**Filter** the results only for chat with id – 17.

**ORDER** the results by message id in **descending** order.

#### Required columns

* id (messages)
* chat\_id
* user\_id

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **chat\_id** | **user\_id** |
| 65 | 17 | 24 |
| ... | ... | ... |

### 13. Users in Bulgaria

Extract all users that are located in Bulgaria, and extract the chat they are in.

The latitude of Bulgaria is in range **[41.139999; 44.129999]**.   
Thelongitude of Bulgaria in range **[22.209999; 28.359999]**.

**ORDER** the results by chat title in **ascending** order.

#### Required columns

* nickname
* title (chats)
* latitude
* longitude

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **nickname** | **title** | **latitude** | **longitude** |
| slittle1 | Lotlux | 42.09028 | 25.03239 |
| ... | ... | ... | ... |

### 14. Last Chat

Extract the **LAST sent** messages (**IF** there is **ANY**) of the **LAST** chat.

The criteria for the messages is the sent\_on date, and the criteria for the chats is the start\_date date.

**ORDER** the results by message id in **ascending** order.

#### Required columns

* title (chats)
* content

#### Example

|  |  |
| --- | --- |
| **title** | **content** |
| Bigtax | NULL |
| ... | ... |

## Section 4: Programmability – 40 pts

You have shown exceptional skills in working with databases. The Senior developers have decided to let you write some automated logic on the application. You’ll need to define several functions, procedures and triggers.

### 15. Radians

Create a **user defined FUNCTION** that transforms **degrees** to **radians**.

The formula should **multiply** the **given degrees** by Pi and then **divide** the result by **180**.

The **return type** must be FLOAT.

The function should be called – udf\_get\_radians**.**

#### Parameters

* degrees

#### Example



|  |
| --- |
| **radians** |
| 0.38606685400009155 |

### 16. Change Password

Create a **user defined PROCEDURE** that receives an email and a password.

The procedure should **CHANGE** the password with the **newly** provided one.

If the given email does **NOT** exist, throw an **exception** with SQLSTATE = 45000 and message “The email does't exist!”.

The procedure should be called – udp\_change\_password**.**

#### Parameters

* email
* new\_password

#### Example

|  |  |
| --- | --- |
| **email** | **password** |
| abarnes0@sogou.com | LOL77s |



|  |  |
| --- | --- |
| **email** | **password** |
| abarnes0@sogou.com | new\_pass |

### 17. Send Message

Create a **user defined PROCEDURE** that receives a user\_id, a chat\_id and the content of a message.

The procedure should then **form** a message with the **given parameters** and the date ‘2016-12-15’ as sent\_on.

If there is **NO** chat with that user throw an exception with SQLSTATE = 45000 and message –   
“There is no chat with that user!”.

The procedure should be called – udp\_send\_message.

#### Parameters

* user\_id
* chat\_id
* content

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **content** | **sent\_on** | **chat\_id** | **user\_id** |
| Awesome | 2016-12-15 | 17 | 19 |

### 18. Log Messages

Create a **TRIGGER** that **logs** any **DELETED** message from table messages.

The **log table** should be called messages\_log and should have exactly the same **structure** as table messages.

The name of the trigger is not important.

**NOTE: Submit ONLY your CREATE TRIGGER statement.**

#### Example



**messages\_log**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **content** | **sent\_on** | **chat\_id** | **user\_id** |
| 1 | in faucibus orci luctus et ultrices posuere cubilia curae mauris | 2016-11-18 | 49 | 20 |

## Section 5: Bonus – 10 pts

Looks like the Senior developers have a really high impression of you. They’ve given you a bonus task. If you manage to complete it, you’ll be awarded.

### 19. Delete Users

Create a **TRIGGER** that will help you to **DELETE** a user.

The name of the trigger is not important.

**NOTE:** **Submit ONLY your CREATE TRIGGER statement.**

#### Example

