



CUHKSZ-Overflow

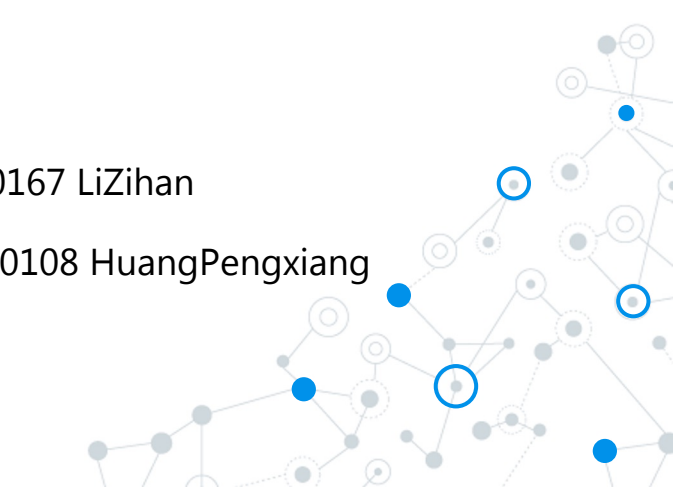
-an online forum database design

119010010 ChenBoyi

119010167 LiZihan

119010185 LinghuHan

119010108 HuangPengxiang



Contents

1

Introduction

2

Database Design

3

SQL Queries

4

Speed up queries
with Index

5

Web Demo

6

Data Mining

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by circles of varying sizes, some with concentric rings, and the lines are thin and grey. The diagram is partially cut off by the top and left edges of the slide.

1.

Introduction

A forum database designed for CUHKSZ programmers as a problem-solving tool



MAIN FUNCTION

Link the question to its correct answer

User

Login, Register, Upload profile...

Blog

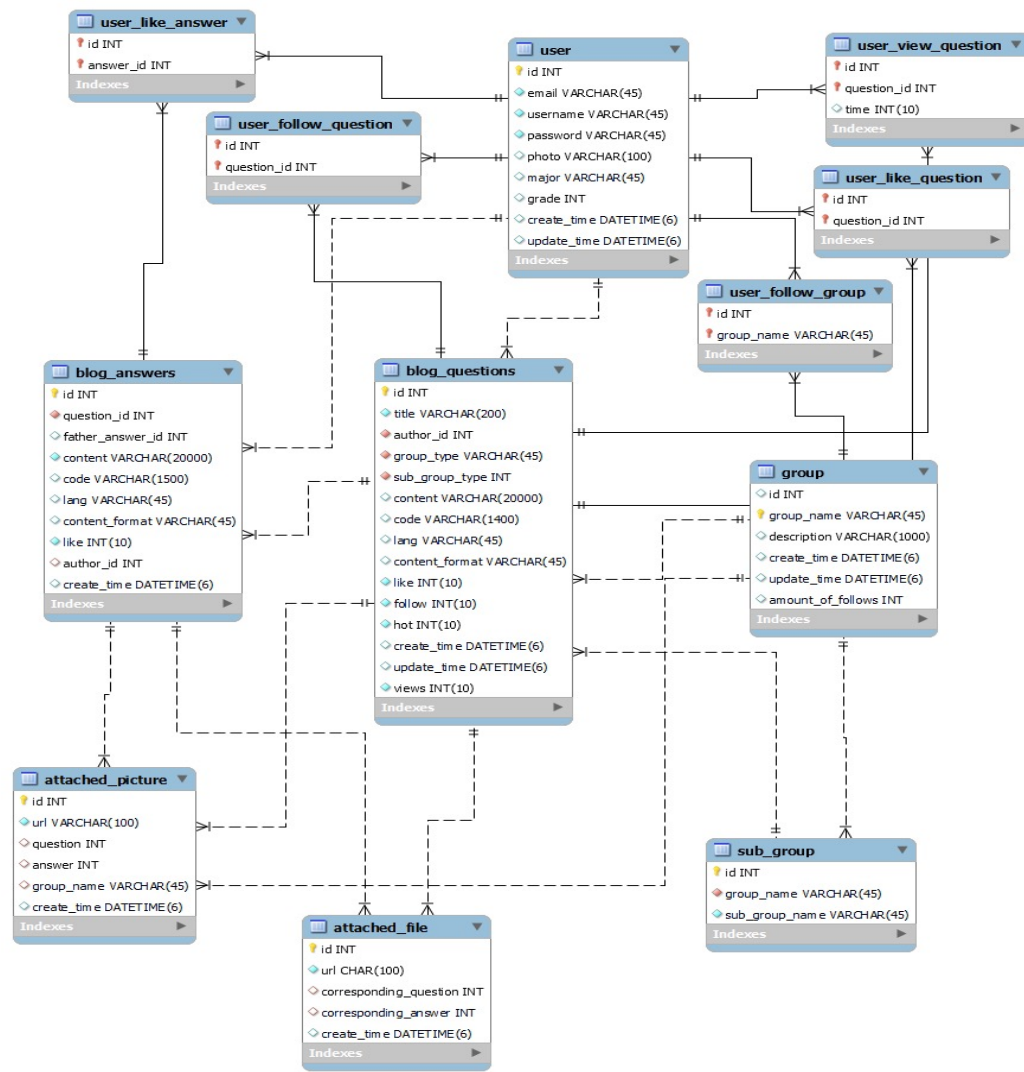
Post, Like, Follow, Answer...

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric circles, suggesting a hierarchical or central structure. The lines are thin and gray, connecting the nodes in a non-linear fashion.

2.

Database Design

E-R diagram



Schema

user(<u>id</u> , email, username, password, photo, major, grade, create_time, update_time)
group(<u>id</u> , group_name, description, create_time, update_time, amount_of_follows)
sub_group(<u>id</u> , group_name, sub_group_name)
blog_questions(<u>id</u> , title, author_id, group_type, sub_group_type, content, code, lang, content_format, like, follow, hot, create_time, update_time, views)
blog_answers(<u>id</u> , question_id, father_answer_id, content, code, lang, content_format, like, author_id, create_time)
file(<u>id</u> , url, corresponding_question, corresponding_answer, create_time)
picture(<u>id</u> , url, question, answer, group_name, create_time)
...

Normalization

group(id, group_name, sub_group_name, description, create_time, update_time, amount_of_follows)

id	group_name	sub_group_name	description	create_time	update_time	amount_of_follows
1	CSC3170	{Assignment1, Assignment2, ...}	Database System	2022-04-30	null	100



group(id, group_name, description, create_time, update_time, amount_of_follows)

id	group_name	description	create_time	update_time	Amount_of_follows
1	CSC3170	Database System	2020-04-30	Null	100

sub_group(id, group_name, sub_group_name)

id	group_name	sub_group_name
1	CSC3170	Assignment1
2	CSC3170	Assignment2

1NF

Normalization

Unique id



Nonprime attributes are fully functionally dependent on the primary key (id)

No nonprime attributes are transitively dependent on the primary key (id)

2NF & 3NF



3.

Sample Queries

DDL: Definition with integrity constraints

```
-- Table structure for table `Our_project_blog_questions`
DROP TABLE IF EXISTS `Our_project_blog_questions`;
CREATE TABLE `Our_project_blog_questions` (
  `id` int NOT NULL AUTO_INCREMENT,
  `title` varchar(200) NOT NULL,
  `author_id` int NOT NULL,
  `group_type` varchar(45) NOT NULL,
  `sub_group_type` int NOT NULL,
  `content` varchar(20000) DEFAULT NULL,
  `code` varchar(1400) DEFAULT NULL,
  `lang` varchar(45) DEFAULT NULL,
  `content_format` varchar(45) DEFAULT NULL,
  `like` int(10) unsigned zerofill NOT NULL,
  `follow` int(10) unsigned zerofill NOT NULL,
  `hot` int(10) unsigned zerofill NOT NULL,
  `create_time` datetime(6) DEFAULT NULL,
  `update_time` datetime(6) DEFAULT NULL,
  `views` int(10) unsigned zerofill NOT NULL,
  PRIMARY KEY (`id`),
  KEY `sub_group_name_idx` (`sub_group_type`),
  KEY `email_idx` (`author_id`),
  KEY `group_type_idx` (`group_type`),
  CONSTRAINT `author_id` FOREIGN KEY (`author_id`) REFERENCES `Our_project_user` (`id`),
  CONSTRAINT `group_type` FOREIGN KEY (`group_type`) REFERENCES `Our_project_group` (`group_name`),
  CONSTRAINT `sub_group_type` FOREIGN KEY (`sub_group_type`) REFERENCES `Our_project_sub_group` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb3;
```

Query 1: directly interacts with database

```
LOCK TABLES `Our_project_user_like_answer` WRITE;  
INSERT INTO `Our_project_user_like_answer` VALUES (4,1),(5,1),(2,2),(5,2),(3,3),(1,4),(3,4),(4,4);  
UNLOCK TABLES;
```

Concurrency control

Query 2: Query within python: SELECT

```
conn = pymysql.connect(host="175.140.1.84", port=3306, user="root", passwd="123456", db="test", charset="utf8")
cursor = conn.cursor()
Answer_T = []
Answer_C = []
blockNumberTotalList = []
lenList = len(question)
searchIndex = 1
for item in question:
    if item[0].isalpha:
        sql = 'select * from Our_project_{} where WORDS = "{}";'.format(item[0].upper(), item)
        cursor.execute(sql)
        a=cursor.fetchall()
        # print(a)
        # print(len(a))
    else:
        sql = 'select * from Our_project_{} where WORDS = "{}";'.format('OTHERS', item)
        cursor.execute(sql)
        a=cursor.fetchall()

    for i in range(len(a)):
        if a[i][1] not in blockNumberTotalList: blockNumberTotalList.append(a[i][1])
        Ttemp1 = [0]*(lenList+1)
        Ctemp1 = [0]*(lenList+1)
        Ttemp1[0] = a[i][1]
        Ttemp1[searchIndex] = a[i][2]
        Ctemp1[0] = a[i][1]
        Ctemp1[searchIndex] = a[i][3]
        Answer_T.append(Ttemp1)
        Answer_C.append(Ctemp1)
    searchIndex += 1
searchIndex -= 1
conn.close()
```

Query 3: Query within python: UPDATE

```
import pymysql
#delete demo with host language.

#1. connect to the DB
conn = pymysql.connect(host="██████", port=3306, user="██████", passwd="██████", db="CSC3170", charset="utf8")
cursor = conn.cursor()

#2. command
newUsername = 'hands'
newpassword = '123123'
sql = "update our_project_user set username=%s and password=%s"%(newUsername, newpassword)
cursor.execute(sql)

#3. commit
cursor.commit()

#4. close conection
cursor.close()
conn.close()
```

Query 3: Query within python: DELETE

```
import pymysql
#delete demo with host language.

#1. connect to the DB
conn = pymysql.connect(host="██████████", port=3306, user="██████", passwd="██████████", db="CSC3170", charset="utf8")
cursor = conn.cursor()

#2. command
deleteId = 3
sql = "Delete from our_project_user where id=%s"
cursor.execute(sql, deleteId)
#3. commit
cursor.commit()

#4. close conection
cursor.close()
conn.close()
```

Other Ways

Alternative Method: Query set

```
titleDB = Blog_Questions.objects.values('title')
for title in titleDB:
    DBlist.append(title['title'].upper())
print(DBlist)
```

It is less comprehensive than SQL, but still a way to interact with the database.



4.

Speed up queries with index

Hash Index

- ⦿ **Advantage:** faster when searching for a specific row (e.g. user id)
- ⦿ **Disadvantage:** Use “memory” engine, volatile, not secure

B-Tree Index

Advantage: can improve speed on searching for a range of values

sub_group_name

username

question_id

...

username

before

```
3 • EXPLAIN SELECT id FROM csc3170.our_project_user WHERE username = 'hehfei';
```

Result Grid											
Filter Rows: <input type="text"/> Export: Wrap Cell Content: <input type="checkbox"/>											
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	our_project_user	HULL	ALL	HULL	HULL	HULL	HULL	51	10.00	Using where

after

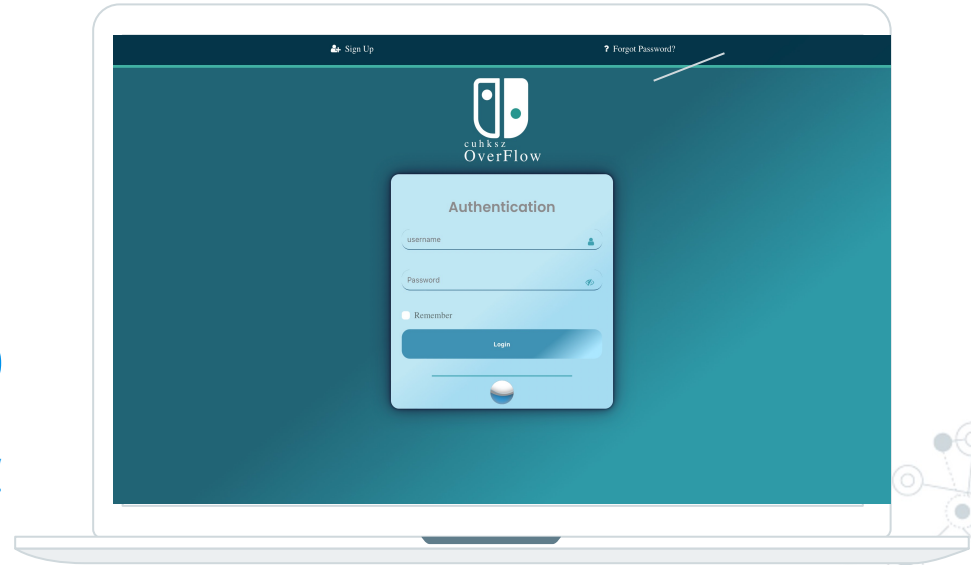
```
1 • ALTER TABLE csc3170.our_project_user ADD INDEX Index_user_name (username);  
2 -- DROP INDEX Index_user_name on csc3170.our_project_user;  
3 • EXPLAIN SELECT id FROM csc3170.our_project_user WHERE username = 'hehfei';
```

Result Grid											
Filter Rows: <input type="text"/> Export: Wrap Cell Content: <input type="checkbox"/>											
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	our_project_user	HULL	ref	Index_user_name	Index_user_name	137	const	1	100.00	Using index

```
username = request.POST['username']  
userid = User.objects.filter(username = username).values()[0]['id']
```

5. Web Demo

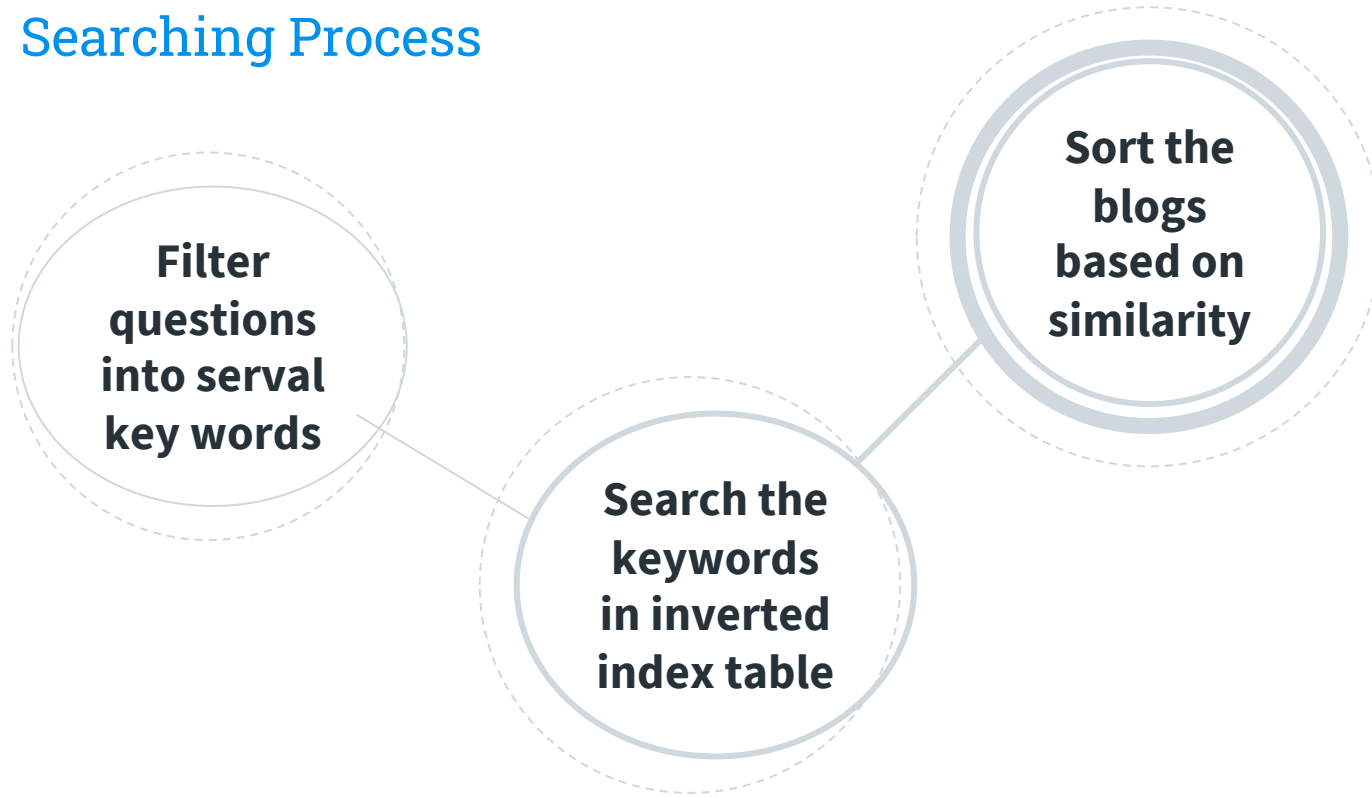
<http://www.cuhksz-stackoverflow.cn/>





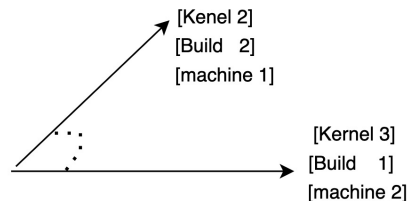
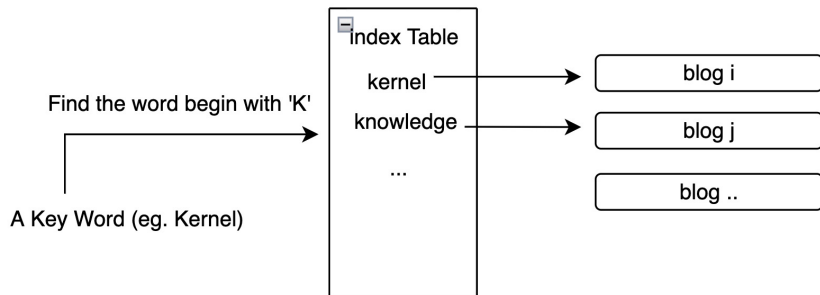
6. **Data Mining**

Searching Process



Search Engine

- ◎ **Key word tokenized:** Natural Language processing (filter the stop words, tense identify, upper letter)
- ◎ **Searching Speed:** inverted index table
- ◎ **Similarity Comparision:** TF-IDF, key words vector projection



Thanks!

If you have any questions, feel free to raise up!