

HOMEWORK 2

Task 1,1

SINGLE COLUMN INDEX :

Single index is a lookup tables created based on one table column as the key that allows to speed up data retrieval

Consider this example: we are doing a review of the beta players applications of our new adventure MMORPG game. 6000 beta players and we are adding records to the table as they come in.

Result: at this stage we have an unsorted gamers list that simply stores the first names, last names, class,position etc.

Task: we need to find a participant called Teemo which could require looking at every single entry in the list, It will take time to find the player,

SELECT * FROM PLAYER WHERE FNAME = 'TEEMO';

A query like the one above reads all first, name records (aka a full table scan) to find all matches which is a slow process,

The screenshot shows two panes in MySQL Workbench. The left pane displays the SQL editor with the following code:

```
CREATE INDEX first_name_index ON users (first_name) USING BTREE;
```

The right pane shows the results of a SELECT query on the users table, which contains 23 rows of data. The columns are ID, FNAME, LNAME, CLASS, POSITION, and SSN. The data includes various player names like Leet, Sait, Lox, Crownguard, Sarah, Fortune, etc.

The query took 0,018sec to complete

WITH INDEX

Adding a single column index would store underlying data in some logical order for example in our case for firstname

CREATE INDEX index_FNAME on PLAYER(FNAME) ;

```

CREATE INDEX index_FNAME_PNAME ON players(FNAME,PNAME);

```

Action Output:

#	Time	Action	Message	Duration / Fetch
12	21:19:19	USE NAME	using innodb storage engine	0.0000 sec
14	21:19:19	CREATE TABLE PLAYER(ID VARCHAR(10), FNAME VARCHAR(10), CLASS VARCHAR(10), POSITION VARCHAR(10), SSN VARCHAR(10))		0.0000 sec
15	21:19:20	INSERT INTO PLAYER(ID,FNAME,CLASS,POSITION,SSN) VALUES ('1','TEEMO','SHROOMER','SPECIALIST','TOP','2345'), ('2','ZED','FIREFLY','FIGHTER','TOP','5678'), ('3','CECIL','HEIDI','SPECIALIST','MD','4561'), ('4','ANNIE','MASTUR','PAGE','MD','8784'), ('5','FLORA','LAURENT','SLAYER','TOP','7897'), ('6','JELLINE','FIREFLY','FIGHTER','TOP','5678'), ('7','MALCOLM','GRAVES','SPECIALIST','ADC','4578'), ('8','IMELIA','LETO','FIGHTER','TOP','5689'), ('9','HINATRA','RABOT','SLAYER','TOP','6668')	Records: 9 Duplicates: 0 Warnings: 0	1.030 sec
16	21:19:20	SELECT * FROM PLAYER WHERE FNAME = 'TEEMO' LIMIT ...	1 row(s) returned	0.018 sec / 0.0001 sec
17	21:19:20	USE NAME	using innodb storage engine	0.0000 sec
18	21:19:20	CREATE INDEX index_FNAME_PNAME ON PLAYERS(FNAME,PNAME)		0.0003 sec
19	21:19:34	USE GAME	using innodb storage engine	0.00018 sec
20	21:19:34	CREATE INDEX index_FNAME_PNAME ON PLAYERS(FNAME)	Records: 0 Duplicates: 0 Warnings: 0	1.373 sec

And after creating the index, you can send the same query

SELECT * FROM applicants WHERE first.name = 'Teemo';

```

SELECT * FROM applicants WHERE first.name = 'Teemo';

```

Action Output:

#	Time	Action	Message	Duration / Fetch
15	21:19:20	USE NAME	using innodb storage engine	0.0000 sec
16	21:19:20	SELECT * FROM PLAYER WHERE FNAME = 'TEEMO' LIMIT ...	Records: 8 Duplicates: 0 Warnings: 0	0.0001 sec / 0.0001 sec
17	21:19:20	USE GAME	using innodb storage engine	0.0000 sec
18	21:19:20	CREATE TABLE PLAYER(ID VARCHAR(10), FNAME VARCHAR(10), CLASS VARCHAR(10), POSITION VARCHAR(10), SSN VARCHAR(10))		0.0000 sec
19	21:19:34	USE NAME	using innodb storage engine	0.0000 sec
20	21:19:34	CREATE INDEX index_FNAME_PNAME ON PLAYERS(FNAME)	Records: 0 Duplicates: 0 Warnings: 0	0.0003 sec
21	21:19:34	USE GAME	using innodb storage engine	0.00018 sec
22	21:19:34	SELECT * FROM PLAYER WHERE FNAME = 'TEEMO' LIMIT ...	1 row(s) returned	0.00023 sec

The query took 0,00031sec to complete this time.

COMPOSITES/ MULTI COLUMN INDEX :

Composites and multi column indexes is a lookup tables created based on multiples table columns that allows to speed up data retrieval

Consider the same example than before : there is probably several person who plays as 'SPECIALIST' class so let's search for people playing SPECIALIST class and TOP position

SELECT * FROM PLAYERS WHERE class = 'SPECIALIST' and position='TOP';

The screenshot shows the MySQL Workbench interface. On the left, the 'SQL' tab displays the SQL command:

```

CREATE INDEX class_pos_index ON PLAYER (SPECIALIST, POSITION);

```

On the right, the 'Results' tab shows the execution log with the message:

The query took 0,00055sec to complete

`CREATE INDEX class_pos_index ON PLAYER (SPECIALIST, POSITION);`

`SELECT * FROM PLAYER
WHERE
CLASS= 'SPECIALIST'
AND
POSITION = 'TOP';`

After creating the index, you can send the query below again

`SELECT * FROM PLAYERS WHERE class = 'SPECIALIST' and position='TOP';`

The screenshot shows the MySQL Workbench interface. On the left, the 'SQL' tab displays the same SQL command as before:

```

CREATE INDEX class_pos_index ON PLAYER (SPECIALIST, POSITION);

```

On the right, the 'Results' tab shows the execution log with the message:

The query took 0,00030sec to complete

Cluster INDEX :

A cluster index is the table itself, which enforces the order of the rows in the table.

Task 1,2

I decided to have an single index on the price of Sweet because when I go to the Bakery , I am usually looking for a cheap snack before going home therefore, I am not searching for a particular type of Sweet but at a Sweet which will cost me less than 0,50£ for example,

USE Bakery;

```
SELECT distinct item_name from Sweet where price <= 0.50;
```

A screenshot of the MySQL Workbench application. The main window shows a database browser with several databases listed: 'parts_database', 'homework2', and 'bakery_database'. Below the database list is a large text area containing a series of SQL queries related to the 'bakery_database'. The sidebar on the right contains four numbered sections labeled 1 through 4, each showing a diagram of a table structure with various columns and relationships. At the bottom of the screen, there is a 'Sweet' tab in a browser-like interface showing a list of items: 'item name', 'doughnut', and 'cinnamon bun'. A status bar at the very bottom indicates 'Query Completed'.

Without the index, the query took 0,030sec,

```
CREATE INDEX index_price ON Sweet(price);
```

```
SELECT distinct item_name from Sweet where price <= 0.50;
```

With the index, the query took 0,00045sec

Task 1.3

As a banker, I would like to keep an eye on my client who have their balance under 0 to check if they have a overdraft allowed or not,

```
SELECT account_holder_name from accounts where balance<=0 AND  
overdraft_allowed=1;
```

```

17 mars 22:49 • MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
rootconnection
parts_database homework2 transaction_bank_account_example*
transaction_bank_account_example*
Limit to 1000 rows ▾
CREATE INDEX index_price ON Sweet(price);
SELECT distinct item_name from Sweet where price <= 0.50;
use bank;
select * from accounts;
SELECT account_holder_name from accounts where balance<=0 AND overdraft_allowed=1;
# account_holder.name
1 Jack
accounts 16
Action Output ▾
Time Action Message Duration / Fetch
44 22:49:42 SELECT account_holder_name from accounts where balance<=0 AND overdraft_allowed=1; 1 row(s) returned 0,00039 sec / 0,000...
45 22:49:42 use bank 0 row(s) affected 0,00019 sec
46 22:49:49 select * from accounts LIMIT 0, 1000 4 row(s) returned 0,00025 sec / 0,000...
47 22:49:50 use bank 0 row(s) affected 0,00022 sec
48 22:49:51 SELECT account_holder_name from accounts where balance<=0 AND overdraft_allowed=1; 1 row(s) returned 0,00034 sec / 0,000...
Query Completed

```

The query took 0,00039sec to complete.

With index **CREATE INDEX index_overdraf_balance ON accounts(balance,overdraft_allowed);** and then sending the same query again

SELECT account_holder_name from accounts where balance<=0 AND overdraft_allowed=1;

```

17 mars 22:56 • MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
rootconnection
parts_database homework2 transaction_bank_account_example*
transaction_bank_account_example*
Limit to 1000 rows ▾
CREATE INDEX index_overdraf_balance ON accounts(balance,overdraft_allowed);
SELECT account_holder_name from accounts where balance<=0 AND overdraft_allowed=1;
# account_holder.name
1 Jack
accounts 16
Action Output ▾
Time Action Message Duration / Fetch
51 22:54:54 SELECT account_holder_name from accounts where balance<=0 AND overdraft_allowed=1; 1 row(s) returned 0,00023 sec
52 22:54:54 use bank 0 row(s) affected 0,0001 sec
53 22:54:54 SELECT account_holder_name from accounts where balance<=0 AND overdraft_allowed=1; 1 row(s) returned 0,00094 sec / 0,000...
54 22:55:55 use bank 0 row(s) affected 0,00017 sec
55 22:55:55 SELECT account_holder_name from accounts where balance<=0 AND overdraft_allowed=1; 1 row(s) returned 0,00033 sec / 0,000...
Query Completed

```

The query took 0,00033sec to complete.

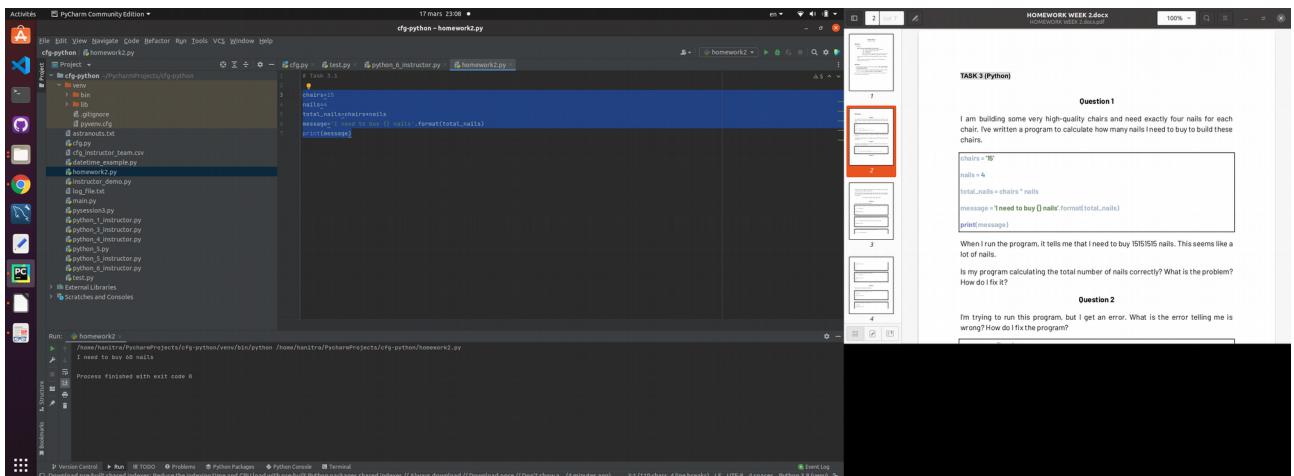
Creating indexes are sometimes storage consuming and have some effeciency improvement.

Task 3Q1

The problem is 15 which is not consider as a number but as a word

The fix is to remove the ' before and after 15 as below

```
chairs=15
nails=4
total_nails=chairs*nails
message='I need to buy {} nails'.format(total_nails)
print(message)
```



Task 3.Q2

The problem is Penelope which should be between " as Penelope is a word. If Penelope is not between " , the program considers as a variable which we didn't define. See the fixed code below

```
my_name='Penelope'
my_age=29
message='My name is {} and I am {} years old'.format(my_name,my_age)
print(message)
```

Task 3.Q3

```
box=input("how many boxes of eggs do you have ? ")
eggs=6
box_eggs = float(box) * float(eggs)
omelettes= float(box_eggs)/4
message='you can make {} omelettes with {} boxes of eggs'.format(omelettes,box)
print(message)
```

Task 3.Q4.1

```
my_str="I love coding."
my_str=my_str.replace(".", "!")
print(my_str)
```

Task 3.Q4.2

```
my_str_1="EVERY Exercise Brings Me CLoser to Completing my GOALS"
```

```
my_str_1=my_str_1.lower()
print(my_str_1)
```

Task 3.Q4.3

```
my_str_2="We enjoy travelling"
ans_1=my_str_2.startswith('A')
print(ans_1)
```

Task 3.Q4.4

```
my_str_3="1.458.001"
ans_2=len(my_str_3)
print(ans_2)
```

The lenght of the string is 9

Task 3.Q5.1

```
wrd="Python"
ans_1=wrd[2:6]
print(ans_1)
```

Task 3.Q5.2

```
wrd="Python"
ans_1=wrd[0:4]
print(ans_1)
```

Task 3.Q5.3

```
wrd="Python"
ans_1=wrd[2:4]
print(ans_1)
```

Task 3.Q5.4

```
wrd="Python"
ans_1=wrd[2:5:2]
print(ans_1)
```

Task 3.6.

it is taking each number between 0 to 100 and multiply to 'o'. As you can't multiply a string with a number without type/format conversion, it wrote 100 times 'o'

Task 3.Q7

The function doesn't return a value,

the fix is as below

```
def calculate_vat(amount):
    q=amount * 1.2
    return q
#outside function
total=calculate_vat(100)
print(total)
```

Task 3 .Q8

```
item_1_name=input("what is the first article ? ")
item_1_price = input("what is the price of first article ? ")
item_2_name=input("what is the first article ? ")
item_2_price = input("what is the price of second article ? ")
item_3_name=input("what is the third article ? ")
item_3_price = input("what is the price of third article ? ")

total = float(item_1_price)+float(item_2_price)+float(item_3_price)
message1 ='{}.....{}'.format(item_1_name,item_1_price)
message2 ='{}.....{}'.format(item_2_name,item_2_price)
message3 ='{}.....{}'.format(item_3_name,item_3_price)
message4 ='Total.....{}'.format(total)
print(message1)
print(message2)
print(message3)
print(message4)
```