# 1. MySQL – MySQLD – MySQLAdmin

\_ MySQLD: is the server executable (one of them), đây là một server dùng để manage database

\_ MySQL: is the command line client.

\_ MySQLAdmin: is a maintainance or administrative utility.

# 2. Storage Engine

**Address:** http://ktmt.github.io/blog/2013/05/07/storageenginemysql/

Storage Engine thực chất là cách MySQL lưu trữ dữ liệu trên đĩa cứng.

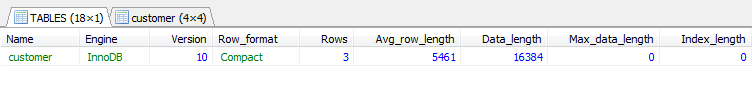
**InnoDB engine:** Là engine phức tạp nhất trong các engine của MySQL

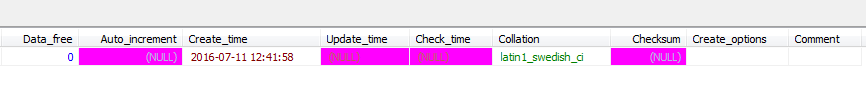
InnoDB engine có hỗ trợ transaction;

Use this query to show information table:

**show** **table** **status** **like** 'customer';

**result:**



****

Transaction define:

http://vietjack.com/sql/transaction\_trong\_sql.jsp

# 3. Select shortest and longest

Query the two cities in **STATION** with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically:

**select city, char\_length(city) city\_length from station order by city\_length, city limit 1;**

**select city, char\_length(city) city\_length from station order by city\_length desc, city limit 1;**

# 4. Select multiple conditions

Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from **STATION**. Your result cannot contain duplicates.

**select distinct city from station where city like 'a%'**

**union**

**select distinct city from station where city like 'e%'**

**union**

**select distinct city from station where city like 'i%'**

**union**

**select distinct city from station where city like 'o%'**

**union**

**select distinct city from station where city like 'u%'**

# 5. Rlike

Query the list of CITY names from **STATION** which have vowels (i.e., a, e, i, o, and u) as both their first and last characters. Your result cannot contain duplicates.

**select distinct city from station where city rlike '^[aeiouAEIOU].\*[aeiouAEIOU]$';**

# 6. RIGHT Substring

**select name from students where marks>75 order by RIGHT(name, 3),id**

# 7. Select case when

**select case**

**when A+B<=C or A+C<=B or B+C<=A then "Not A Triangle"**

**when A=B and A=C then "Equilateral"**

**when A=B or A=C or B=C then "Isosceles"**

**else "Scalene"**

**end as triangle\_type**

**from TRIANGLES;**

# 8. Group by, concat

Generate the following two result sets:

1. Query an *alphabetically ordered* list of all names in **OCCUPATIONS**, immediately followed by the first letter of each profession as a parenthetical (i.e.: enclosed in parentheses). For example: AnActorName(A), ADoctorName(D), AProfessorName(P), and ASingerName(S).
2. Query the number of ocurrences of each occupation in **OCCUPATIONS**. Sort the occurrences in *ascending order*, and output them in the following format:
3. There are a total of [occupation\_count] [occupation]s.

where [occupation\_count] is the number of occurrences of an occupation in **OCCUPATIONS** and [occupation] is the *lowercase* occupation name. If more than one *Occupation* has the same [occupation\_count], they should be ordered alphabetically.

**Note:** There will be at least two entries in the table for each type of occupation.

**select concat(name,"(",substr(occupation,1,1),")") from occupations**

**order by name asc;**

**select concat('There are a total of ',count(occupation),' ',lower(occupation),'s.') as n from occupations**

**group by occupation**

**order by n asc;**

# 9. Đếm giá trị

**set @r1=0, @r2=0, @r3=0, @r4=0;**

**select min(Doctor), min(Professor), min(Singer), min(Actor)**

**from(**

**select case when Occupation='Doctor' then (@r1:=@r1+1)**

**when Occupation='Professor' then (@r2:=@r2+1)**

**when Occupation='Singer' then (@r3:=@r3+1)**

**when Occupation='Actor' then (@r4:=@r4+1) end as RowNumber,**

**case when Occupation='Doctor' then Name end as Doctor,**

**case when Occupation='Professor' then Name end as Professor,**

**case when Occupation='Singer' then Name end as Singer,**

**case when Occupation='Actor' then Name end as Actor**

**from OCCUPATIONS**

**order by Name**

**) Temp**

**group by RowNumber;**

# 10. Binary search

## Tự làm:

**select NN,title from**

**(select**

**case when P is not NULL then P end as PP,**

**case**

**when P is NULL then N**

**when P is not NULL then N**

**end as NN,**

**case**

**when P is NULL then 'Root'**

**when P is not NULL and (select count(P) from BST where P=NN)>0 then 'Inner'**

**else 'Leaf'**

**end as title from BST order by N) TEMP;**

## Có tham khảo:

**select TEMP.n,**

**(**

**CASE WHEN TEMP.p IS NULL THEN 'Root'**

**WHEN (select count(\*) from BST TEMP2 where TEMP2.P = TEMP.N) > 0 THEN "Inner"**

**ELSE "Leaf" END**

**) from BST TEMP**

**order by TEMP.n**

# 11. Ví dụ điển hình join các table

**select c.company\_code,c.founder ,count(distinct e.lead\_manager\_code),count(distinct e.senior\_manager\_code),count(distinct e.manager\_code),count(distinct e.employee\_code)**

**from company c join**

**employee e**

**on**

**c.company\_code=e.company\_code group by c.company\_code,c.founder order by c.company\_code**

# 12. Revising aggregation

## Tự làm

set @total = 0;

select distinct @total from (select case when District='California' then (@total:=@total+population) end from CITY) TEMP;

## Tham khảo hàm sum:

|  |
| --- |
| SELECT SUM(POPULATION) |
|  |

|  |
| --- |
| FROM CITY |
|  |

WHERE DISTRICT = "California";

# 13. Rounded down - FLOOR

select FLOOR(population) from city;

# 14. Round nex integer – CEIL

SELECT CEIL(AVG(Salary)-AVG(REPLACE(Salary,'0',''))) FROM EMPLOYEES;

# 15. TOP Earning

**set @maxSalary = 0;**

**set @maxSalary := (select maxSalary from (select max(salary\*months) as maxSalary from Employee) TEMP);**

**select @maxSalary,count(\*) from Employee where salary\*months = @maxSalary;**

# 16. Round 2 decimal place

select ROUND(sum(LAT\_N),2),ROUND(sum(LONG\_W),2) from STATION;

# 17. select from min

select round(long\_w,4) from station where lat\_n=(select min(lat\_n) from station where lat\_n>38.7780)

# 18. Manhattan distance

set @a = (select min(lat\_n) from station);

set @b = (select min(long\_w) from station);

set @c = (select max(lat\_n) from station);

set @d = (select max(long\_w) from station);

set @distance = abs(@a-@c)+abs(@b-@d);

select round(@distance,4);

# 19. Square Root

set @a = (select min(lat\_n) from station);

set @b = (select min(long\_w) from station);

set @c = (select max(lat\_n) from station);

set @d = (select max(long\_w) from station);

set @distance = SQRT ((@a-@c)\*(@a-@c)+(@b-@d)\*(@b-@d));

select round(@distance,4);

# 20. Sign

Tự làm: ở đây 249 là lấy cái length

SELECT round(lat\_n,4) FROM station order by lat\_n LIMIT 1 OFFSET 249;

Tham khảo: ở đây có lẽ tốn perfomance

SELECT round( x.LAT\_N, 4 ) from STATION x, STATION y GROUP BY x.LAT\_N HAVING SUM( SIGN( 1-SIGN( y.LAT\_N -x.LAT\_N ) ) )/COUNT(\*) > .5 LIMIT 1;

ở đây y GROUP BY x.LAT\_N HAVING SUM( SIGN( 1-SIGN( y.LAT\_N -x.LAT\_N ) ) )/COUNT(\*) > .5 sẽ lấy các giá trị lớn hơn từ phân nửa

# 21. join same as inner join

select sum(c1.population) from city c1 join country c2 on c1.countrycode=c2.code and c2.continent = 'asia';

# \*22. Round down to nearest integer

## Tự làm:Sau khi query ra được các distinct continent

select c2.continent,floor(avg(c1.Population)) from city c1 join country c2 on c1.countrycode=c2.code and c2.continent='Asia';

select c2.continent,floor(avg(c1.Population)) from city c1 join country c2 on c1.countrycode=c2.code and c2.continent='Oceania';

select c2.continent,floor(avg(c1.Population)) from city c1 join country c2 on c1.countrycode=c2.code and c2.continent='Europe';

select c2.continent,floor(avg(c1.Population)) from city c1 join country c2 on c1.countrycode=c2.code and c2.continent='South America';

select c2.continent,floor(avg(c1.Population)) from city c1 join country c2 on c1.countrycode=c2.code and c2.continent='Africa';

## Tham khảo:

**select c2.continent,floor(avg(c1.Population)) from city c1 inner join country c2 on c1.countrycode=c2.code GROUP BY c2.CONTINENT;**

# \*23. join between

**select if(g.Grade < 8,NULL,s.Name),g.Grade,s.Marks**

**from Students s**

**inner join Grades g**

**on s.Marks >= g.Min\_Mark and s.Marks <= g.Max\_Mark**

**order by g.Grade desc,s.Name,s.Marks;**

# \*24. Having

## Có tham khảo 1:

**SELECT h.hacker\_id, h.name**

**FROM submissions s**

**JOIN challenges c**

**ON s.challenge\_id = c.challenge\_id**

**JOIN difficulty d**

**ON c.difficulty\_level = d.difficulty\_level**

**JOIN hackers h**

**ON s.hacker\_id = h.hacker\_id**

**WHERE s.score = d.score**

**AND c.difficulty\_level = d.difficulty\_level**

**GROUP BY h.hacker\_id**

**HAVING COUNT(s.hacker\_id) > 1**

**ORDER BY COUNT(s.hacker\_id) DESC, s.hacker\_id ASC**

## Có tham khảo 2:

**SELECT h.hacker\_id, h.name**

**FROM submissions s**

**inner JOIN challenges c**

**inner JOIN difficulty d**

**inner JOIN hackers h**

**ON s.challenge\_id = c.challenge\_id**

**and c.difficulty\_level = d.difficulty\_level**

**and s.hacker\_id = h.hacker\_id**

**WHERE s.score = d.score**

**AND c.difficulty\_level = d.difficulty\_level**

**GROUP BY h.hacker\_id, h.name**

**HAVING COUNT(s.hacker\_id) > 1**

**ORDER BY COUNT(s.hacker\_id) DESC, s.hacker\_id ASC**

# \*25. Olivander’s wand – select được chỗ min

**select w.id,p.age,w.coins\_needed,w.power from Wands w**

**inner join Wands\_Property p on w.code=p.code**

**where p.is\_evil=0 and**

**w.coins\_needed = (select min(w1.coins\_needed) from Wands w1 join Wands\_Property p1 on w1.code=p1.code where**

**w.power = w1.power and p.age=p1.age)**

**order by w.power desc, p.age desc**

# \*26. Select được chỗ max với 2 điều kiện

**select c.hacker\_id,h.name,count(c.hacker\_id) as c\_count from Challenges c**

**inner join Hackers h**

**on c.hacker\_id=h.hacker\_id**

**group by c.hacker\_id,h.name**

**having**

**/\* output anyone with a count that is equal to... \*/**

**c\_count =**

**/\* the max count that anyone has \*/**

**(SELECT MAX(temp1.cnt)**

**from (SELECT COUNT(hacker\_id) as cnt**

**from Challenges**

**group by hacker\_id**

**order by hacker\_id) temp1)**

**/\* or anyone who's count is in... \*/**

**or c\_count in**

**/\* the set of counts... \*/**

**(select t.cnt**

**from (select count(\*) as cnt**

**from challenges**

**group by hacker\_id) t**

**/\* who's group of counts... \*/**

**group by t.cnt**

**/\* has only one element \*/**

**having count(t.cnt) = 1)**

**/\* finally, the order the rows should be output \*/**

**order by c\_count DESC, c.hacker\_id**