**1.데이터 베이스 접속하기**mysql -u 사용자명 -p 데이터베이스이름  
mysql -u root (계정과 비번을 만들지 않았을때)  
mysql -u root -p(루트 비번이 있을때)  
  
**2.데이터베이스 추가**mysqladmin -u root -p create 데이터베이스명  
  
**3.루트 계정비밀번호 바꾸기**mysqladmin -u root -p password 새비밀번호  
mysqladmin -u root password 비밀번호(비밀번호를 처음 설정시)  
  
**4.사용자추가 및 권한 설정**사용자계정:abc , 사용자 패스워드:1234  
  
접속 : mysql -u root -p

mysql> USE mysql;

mysql> INSERT INTO user (Host, User, Password) VALUES ('localhost', '계정아이디', password('비밀번호'));  
mysql> INSERT INTO user (Host, User, Password) VALUES ('%', '계정아이디', password('비밀번호'));  
mysql> FLUSH privileges;

계정이 생성되었다면, 그 계정이 접근할 수 있는 데이터베이스를 생성하고 권한을 부여해야 합니다.

mysql> CREATE DATABASE 데이터베이스명;  
mysql> GRANT ALL privileges ON 데이터베이스명.\* TO [계정아이디@locahost](mailto:%ACC4%C815%C544%C774%B514@locahost) IDENTIFIED BY '비밀번호';  
mysql> GRANT ALL privileges ON 데이터베이스명.\* TO [계정아이디@'%'](mailto:%ACC4%C815%C544%C774%B514@'%25') IDENTIFIED BY '비밀번호';  
  
**5.테이블 생성**create table test;  
  
**6.테이블 삭제**drop table test;  
  
**7.현재상태보기**status;  
  
**8.테이블목록보기**show tables;  
  
**9.데이터베이스 목록보기**show databases;  
  
**10.테이블구조보기**explain tablesname;  
  
**11.데이터베이스 삭제**drop database jsp

\*체크\*  
  
우선, 기본적으로 mysql 은 **tcp 3306** 번 포트를 통해 서비스를 제공합니다.

따라서, 외부에서 DB에 접근할 수 있도록 하기 위해서는

서버의 **tcp 3306** 번 포트가 방화벽에 막혀 있지는 않은지 확인해야 하며

만약 막혀 있을 경우 열어주어야 합니다.

**iptables -L** 명령으로 차단/허용 리스트 조회가 가능합니다. 만약 포트가 막혀 있을 경우,

**#** iptables -I INPUT -p tcp --dport 3306 -j ACCEPT

명령으로 열어주시면 됩니다.

출처: [http://vkein.tistory.com/entry/MySQL-접속-명령어](http://vkein.tistory.com/entry/MySQL-%EC%A0%91%EC%86%8D-%EB%AA%85%EB%A0%B9%EC%96%B4) [Vkein Programer]

\*where문에 조건을 넣고싶을 때!

-AND와 OR를 잘 활용하라!

Ex) 특정 검색조건에따라 Select하고 싶을 때(검색조건이 특정id로 넘어오면 해당 id에 따른 데이터만 보여줄것. 하지만 all이면 전체가 다 나오도록)

Set @user\_id = ‘ALL’ // select @user\_id:=’ALL’ from dual //사용자 변수 user\_id에 값 삽입

Select

\*

from

table1

where

(@user\_id=’ALL’ AND 1=1)

OR (user\_id=@user\_id) //else if의 개

념은 AND를 또 넣어주면됨

위의 내용은 아래와 같음

if(@user\_id==”ALL”){

1=1

}else{

user\_id=@user\_id

}

-

**UNION과 UNION ALL 의 차이 및 주의 사항**

ANSI SQL에서 제안하는 집합 연산 "UNION", "INTERSECT", "MINUS" 중에서  
**MySQL에서는 UNION 집합 연산만 제공**하고 있다.   
(하지만 MySQL에서 INTERSECT나 MINUS를 다른 형태의 쿼리로 풀어서 사용할 수 있다.)  
  
이 글에서는 UNION 에 대해서 좀 더 자세히 알아 보고자 한다.  
UNION 집합 연산은 다시 아래와 같이 두가지 종류로 나누어진다.  
**- UNION ALL  
  - UNION DISTINCT**  
  
우리가 일반적으로 사용하는 방식인 **아무런 추가 키워드 없이 UNION 만 사용하는 것은  
UNION DISTINCT 를 줄여서 사용하고 있는 것이다**.  
**UNION ALL과 UNION DISTINCT를 레코드가 많은 결과에 대해서 적용해본 사람은   
아마도 둘의 처리 방식에 대해서 의구심을 가져본 적이 있을 것이다.**  
레코드 건수가 많아지면 많아질수록 그 성능 차이는 엄청난 차이를 보여줄 것이다.  
우선, 아래와 같이 2개씩 동일한 레코드 데이터를 가지고 있는 tab1과 tab2라는 테이블이 있다.  
  
mysql>*SELECT fdpk, fddata FROM tab1;*  
+------+--------+  
| fdpk | fddata |  
+------+--------+  
|    1 | data1  |  
|    2 | data2  |  
+------+--------+  
2 rows in set (0.00 sec)  
  
mysql>*SELECT fdpk, fddata FROM tab2;*  
+------+--------+  
| fdpk | fddata |  
+------+--------+  
|    1 | data1  |  
|    2 | data2  |  
+------+--------+  
2 rows in set (0.01 sec)  
  
그러면, 이 두개 테이블에 대해서 각각 UNION과 UNION ALL을 사용하는 쿼리를 실행해보자.  
  
mysql>*SELECT fdpk, fddata*  
    -> *FROM (*  
    ->*SELECT fdpk, fddata FROM tab1*  
    -> ***UNION ALL***  
    -> *SELECT fdpk, fddata FROM tab2*  
    -> *) x;*  
+------+--------+  
| fdpk | fddata |  
+------+--------+  
|    1 | data1  |  
|    2 | data2  |  
|    1 | data1  |  
|    2 | data2  |  
+------+--------+  
4 rows in set (0.00 sec)  
  
mysql>*SELECT fdpk, fddata*  
    -> *FROM (*  
    -> *SELECT fdpk, fddata FROM tab1*  
    -> ***UNION***  
    -> *SELECT fdpk, fddata FROM tab2*  
    -> *) x;*  
+------+--------+  
| fdpk | fddata |  
+------+--------+  
|    1 | data1  |  
|    2 | data2  |  
+------+--------+  
2 rows in set (0.00 sec)  
  
두개의 퀴리 실행 결과 UNION은 레코드가 반으로 줄었다.  
이미 다들 알고 있다시피 **UNION은 UNION DISTINCT와 동일한 작업을 하기 때문에 중복되는 레코드를 제거**했음을 알 수 있다.  
하지만, **UNION ALL의 경우에는 별도의 중복 제거 과정을 거치지 않고 그냥 결과를 내려준다**.  
아주 중요한 내용이지만, 사실 이 내용을 다들 별로 신경쓰지 않고 모두들 UNION을 즐겨 사용한다.  
  
안타깝게도, MySQL의 실행계획에서는 둘의 차이를 전혀 느낄 수 없다.  
+----+--------------+------------+------+..+------+..+------+------+-------+  
| id | select\_type  | table      | type |..| key  |..| ref  | rows | Extra |  
+----+--------------+------------+------+..+------+..+------+------+-------+  
|  1 | PRIMARY      | <derived2> | ALL  |..| NULL |..| NULL |    4 |       |  
|  2 | DERIVED      | tab1       | ALL  |..| NULL |..| NULL |    2 |       |  
|  3 | UNION        | tab2       | ALL  |..| NULL |..| NULL |    2 |       |  
|NULL| UNION RESULT | <union2,3> | ALL  |..| NULL |..| NULL | NULL |       |  
+----+--------------+------------+------+..+------+..+------+------+-------+  
  
**하지만 중복 제거는 그냥 얻을 수 있는 결과가 아니다.**그러면, MySQL이 내부적으로 어떻게 중복을 제거하는 것일까 ?  
  
내부적인 처리를 알아보기 전에, 레코드의 중복이라는 표현을 했는데 이 중복의 기준이 무었일까 ?  
    1. 각 테이블의 Primary key ?  
    2. 전체 테이블의 모든 필드 ?  
    3. 각 서브 쿼리에서 SELECT된 튜플(레코드)의 모든 필드 ?  
  
그렇다. 이미 SELECT된 결과를 가지고 UNION하기 때문에 SELECT되기 전의 테이블이나 레코드에 대한 정보는 알 수 없다.  
그래서, 중복 여부의 판단은 SELECT된 튜플들에 속해있는**모든 컬럼의 값들 자체가 중복 체크의 기준**이 되는 것이다.  
  
자~, 그러면 이제 MySQL이 내부적으로 UNION ALL과 UNION을 처리하는 과정을 알아보자.  
1. 최종 UNION [ALL | DISTINCT] 결과에 적합한 임시 테이블(Temporary table)을 메모리 테이블로 생성  
2. UNION 또는 UNION DISTINCT 의 경우, Temporary 테이블의 모든 컬럼으로 Unique Hash 인덱스 생성3. 서브쿼리1 실행 후 결과를 Temporary 테이블에 복사  
4. 서브쿼리2 실행 후 결과를 Temporary 테이블에 복사  
5. 만약 3,4번 과정에서 Temporary 테이블이 특정 사이즈 이상으로 커지면   
    Temporary 테이블을 Disk Temporary 테이블로 변경   
    (이때 Unique Hash 인덱스는 Unique B-Tree 인덱스로 변경됨)  
6. Temporary 테이블을 읽어서 Client에 결과 전송  
7. Temporary 테이블 삭제  
  
UNION 두 가지의 차이는 2번 과정 딱 하나이다. 중복 제거를 위해서 Temporary 테이블에 인덱스를 생성하느냐 ?. 그렇지 않느냐 ?.별로 중요하지 않은 것 같지만, 이 인덱스로 인해서 3,4번 과정의 작업이 작지 않은 성능 차이가 만들어 내게 된다.  
실제 UNION을 실행하는 데이터의 건수에 따라서 다르겠지만, 1.5 ~ 4배 가량의 성능 차이로 UNION ALL이 빠르게 처리된다.  
만약 처리중 데이터의 량이 작아서 5번 과정을 거치지 않는다면 메모리 Temporary 테이블에 Hash 인덱스를 사용하기 때문에   
속도 차이가 아주 미세할 것이다.   
하지만 데이터량이 커져서 5번 과정을 거치게 되면 Disk Temporary 테이블에 B-Tree 인덱스를 사용하기 때문에 큰 성능 차이를 보이게 될 것이다.  
이 성능 차이는 UNION 하는 두 집합에 중복되는 레코드가 있든 없든 관계 없이 발생할 것이다.  
  
위에서 잠깐 알아보았던, "중복의 기준"을 생각하면, UNION 하는 컬럼들의 수가 많아지고 레코드의 사이즈가 커질수록 두 작업 모두에게 불리하겠지만, UNION ALL보다는 UNION에 더 악영향이 클 것이다.  
  
**결론은,**  
0. UNION 이든지 UNION ALL이든지 사실 그리 좋은 SQL 작성은 아니다.   
    UNION이 필요하다는 것은 사실 두 엔터티(테이블)가 하나의 엔터티(테이블)로 통합이 되었어야   
    할 엔터티들이었는데, 알 수 없는 이유로 분리 운영되는 경우가 상당히 많다.   
    즉 모델링 차원에서 엔터티를 적절히 통합하여 UNION의 요건을 모두 제거하자.  
1. 두 집합에 절대 중복된 튜플(레코드)가 발생할 수 없다는 보장이 있다면 UNION ALL을 꼭 사용하자.  
    두 집합에서 모두 각각의 PK를 조회하는데, 그 두 집합의 PK가 절대 중복되지 않는 형태  
2. 중복이 있다 하더라도 그리 문제되지 않는다면 UNION 보다는 UNION ALL을 사용하자.  
3. 만약 UNION이나 UNION ALL을 사용해야 한다면, 최소 필요 컬럼만 SELECT 하자

출처 : <http://intomysql.blogspot.com/2011/01/union-union-all.html>

**하나의 컬럼에 delimiter로 구분하여 여러 데이터 들어가있는것들 잘라서 하나의 컬럼과 여러행으로 표현해주는방법**

**fruit테이블**

**fruitName컬럼**

**데이터 -> 사과,배,참외,복숭아,수박**

**이것을 행으로 만드는방법!**

**결과값**

**fn**

**사과**

**배**

**참외**

**복숭아**

**수박**

**select**

numbers.n,

**SUBSTRING\_INDEX** (**SUBSTRING\_INDEX**(fruitName,',',numbers.n),',',-1) fn

**from**

(**select** 1 n **union** **all** **select** 2

**union** **all** **select** 3 **union** **all** **select** 4

**union** **all** **select** 5 **union** **all** **select** 6

**union** **all** **select** 7 **union** **all** **select** 8

**union** **all** **select** 9 **union** **all** **select** 10) numbers **INNER** **JOIN** fruit

**on** **CHAR\_LENGTH** (fruitName)

- **CHAR\_LENGTH** ( **REPLACE** ( fruitName , ',' , '' ))>= numbers.n-1

*/\*1,2,3,4,5 는이해감.. \*/*

Substring\_index(컬럼명,’delim’,숫자)

Delim에 따라 쪼개진것들 숫자에 맞춰 가져옴(양수이면 왼쪽부터, 음수는 오른쪽부터)

행을 열로 바꾸는 방법,,

Ex)

Fruit table

fruitName | fruitPrice

사과 1000

배 500

딸기 2000

참외 5000

수박 7000

망고 10000

이것을

사과 | 배 | 딸기 | 참외 | 수박 | 망고

1000 500 2000 5000 7000 10000

이렇게 바꾸려면

**select**

**case** **when** e.fruitName='배' **then** e.fruitProce **end** **as** '배',

**case** **when** e.fruitName='딸기' **then** e.fruitProce **end** **as** '딸기',

**case** **when** e.fruitName='참외' **then** e.fruitProce **end** **as** '참외',

**case** **when** e.fruitName='사과' **then** e.fruitProce **end** **as** '사과',

**case** **when** e.fruitName='수박' **then** e.fruitProce **end** **as** '수박',

**case** **when** e.fruitName='망고' **then** e.fruitProce **end** **as** '망고'

**from** fruit e

이렇게 시도하면



이렇게 나온다..

이를 해결하기위해서는 null을 없애고 한줄로 쭉 만들어줘야하는데,

이때 사용하는 것이 group\_concat()이다.

**select**

**group\_concat**(**case** **when** e.fruitName='배' **then** e.fruitProce **end**)**as** '배',

**group\_concat**(**case** **when** e.fruitName='딸기' **then** e.fruitProce **end**) **as** '딸기',

**group\_concat**(**case** **when** e.fruitName='참외' **then** e.fruitProce **end**) **as** '참외',

**group\_concat**(**case** **when** e.fruitName='사과' **then** e.fruitProce **end**) **as** '사과',

**group\_concat**(**case** **when** e.fruitName='수박' **then** e.fruitProce **end**) **as** '수박',

**group\_concat**(**case** **when** e.fruitName='망고' **then** e.fruitProce **end**) **as** '망고'

**from** fruit e;



깔금하게나온다.

그러나 만약 fruit table에 동일한 fruitName이 하나 더있으면 구분자 ,(comma 이게 디폴트) 로 추가된값이 나오게된다.



이를 해결하기위해서 from에서 서브쿼리를 사용하여 group by로 묶어주면 해결!

**select**

**group\_concat**(**case** **when** e.fruitName='배' **then** e.fruitProce **end**)**as** '배',

**group\_concat**(**case** **when** e.fruitName='딸기' **then** e.fruitProce **end**) **as** '딸기',

**group\_concat**(**case** **when** e.fruitName='참외' **then** e.fruitProce **end**) **as** '참외',

**group\_concat**(**case** **when** e.fruitName='사과' **then** e.fruitProce **end**) **as** '사과',

**group\_concat**(**case** **when** e.fruitName='수박' **then** e.fruitProce **end**) **as** '수박',

**group\_concat**(**case** **when** e.fruitName='망고' **then** e.fruitProce **end**) **as** '망고'

**from** (**select** **sum**(fruitProce) **as** fruitProce,fruitName **from** fruit **group** **by** fruitName) e



굿.

\*\*추가로

pivot테이블 생성

-http://stratosprovatopoulos.com/web-development/mysql/pivot-a-table-in-mysql/

전체적인설명

-https://ntalbs.github.io/2015/sql-transpose/

Group\_concat

설명 내용 굿 : <http://blackbull.tistory.com/3>

group으로 묶어서 나오는 결과를 concat( concatenate – 결부시키다) 요청한 구분자에따라(디폴트는 , ) 연결지어줌

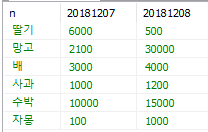
열을 행으로 바꾸는 방법

예를들면

fruit테이블이 아래와같을 때(bdate에 따라 과일값들을 행으로!)



이를 아래와 같이 변경!



1.첫번째 열 n을 만들어주어야함

**select** '사과' n

**union** **all**

**select** '배'

**union** **all**

**select** '딸기'

**union** **all**

**select** '자몽'

**union** **all**

**select** '망고'

**union** **all**

**select** '수박'

**결과화면**



2. 데이터들을 행으로 다 뽑기

**select**

t.n,f.bdate,

**case** t.n

**when** '사과' **then** f.사과

**when** '배' **then** f.배

**when** '딸기' **then** f.딸기

**when** '자몽' **then** f.자몽

**when** '망고' **then** f.망고

**when** '수박' **then** f.수박

**end** **as** price

**from**

(**select** \* **from** fruit)f,

(**select** '사과' n

**union** **all**

**select** '배'

**union** **all**

**select** '딸기'

**union** **all**

**select** '자몽'

**union** **all**

**select** '망고'

**union** **all**

**select** '수박')t

결과화면



중복되서 한줄로 나옴..(날짜에 따라 따로 분류해서 행으로 나눠야함)

3. case when을 통해서 column을 만들어주고 group by 로 묶을것!

**select**

a.n,

**case** **when** a.bdate='2018-12-07' **then** a.price **end** **as** '20181207' ,

**case** **when** a.bdate='2018-12-08' **then** a.price **end** **as** '20181208'

**from**

(**select**

t.n,f.bdate,

**case** t.n

**when** '사과' **then** f.사과

**when** '배' **then** f.배

**when** '딸기' **then** f.딸기

**when** '자몽' **then** f.자몽

**when** '망고' **then** f.망고

**when** '수박' **then** f.수박

**end** **as** price

**from**

(**select** \* **from** fruit)f,

(**select** '사과' n

**union** **all**

**select** '배'

**union** **all**

**select** '딸기'

**union** **all**

**select** '자몽'

**union** **all**

**select** '망고'

**union** **all**

**select** '수박')t

)a



Group by 로 묶으지않으면 이렇게나옴..

최종으로 group by로 묶으면

**select**

a.n,

**min**(**case** **when** a.bdate='2018-12-07' **then** a.price **end**) **as** '20181207' ,

**min**(**case** **when** a.bdate='2018-12-08' **then** a.price **end**) **as** '20181208'

**from**

(**select**

t.n,f.bdate,

**case** t.n

**when** '사과' **then** f.사과

**when** '배' **then** f.배

**when** '딸기' **then** f.딸기

**when** '자몽' **then** f.자몽

**when** '망고' **then** f.망고

**when** '수박' **then** f.수박

**end** **as** price

**from**

(**select** \* **from** fruit)f,

(**select** '사과' n

**union** **all**

**select** '배'

**union** **all**

**select** '딸기'

**union** **all**

**select** '자몽'

**union** **all**

**select** '망고'

**union** **all**

**select** '수박')t

)a

**group** **by** a.n



굳

\*\*참고로 아래와같이 한줄만 행으로 바꾸는 것은 2번까지만하면 됨.



**회사 데이터 예제**

**select**

t.n,

**case**

**when** t.n='OK' **then** s.totalOK

**when** t.n='NG' **then** s.totalNG

**end** **as** val

**from**

(**select**

**sum**(OK) **as** totalOK,

**sum**(NG) **as** totalNG

**from**

edst\_complete\_code c

**where**

c.BDate **between** '2018-01-01' **and** '2018-12-31'

**and** user\_id='admin'

**group** **by** user\_id)s ,

(**select** 'OK' n

**union** **all**

**select** 'NG')t

 => 

여기를 참고함 : https://idchowto.com/?p=12772