

3. 사용자 및 친한

MySQL의 사용자 계정

⇒ 사용자의 아이디 포함 + 어느 IP에서 접속하는지 확인.

MySQL 8.0

⇒ 친한 데어서 관리하는 '역할 개념' 도입

3.1 사용자 식별

MySQL 사용자 => 사용자 계정 + 사용자 접속지점 : '계정의 일부'

클라이언트가 실행된 [호스트명 or 도메인
IP 주소]

MySQL 계정 인증 => 아이디 + 호스트 명시

ID와 IP 주소를 겹칠 때 ' or '

if) localhost에서 svc_id라는 아이디 등록

⇒ 다른 컴퓨터에서는 svc_id라는 아이디로 접속X.

ex)

'svc_id' @ '127.0.0.1'

if) 모든 곳에 접속 가능한 사용자 계정 등록하고 싶다면?

⇒ 호스트 부분을 (%) 문자로 대체.

모든 IP or 모든 호스트명 의미

주의사항 : 서로 동일 아이디 있을 때 MySQL 서버가 해당 사용자 인증 위해 어떤 계정 선택?

ex)

'svc_id' @ '192.168.0.10' (pw: 123)

'svc_id' @ '%' (pw: abc)

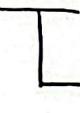
친한이나 계정 정보에 대해
MySQL이 범위가 가장 작은 것
항상 먼저 선택

범위가 좁은 것
⇒ % 문자 포함 X 인 것.

⇒ 결국 중첩된 계정 생성X

3.2 사용자 계정 관리

3.2.1 시스템계정 . 일반계정

SYSTEM_USER 권한  O : 시스템 계정 (System Account)
X : 일반 계정 (Regular Account)

1) 시스템 계정

: DB Server 관리자 권한 계정

: 일반계정) 관리 ... 생성. 삭제. 수정
시스템 계정

: DB Server 관리자 **중요작업** 담당

- 계정관리 (계정 생성. 삭제, 권한부여. 제거)
- 다른 세션 or 그 세션에서 실행 중인 쿼리 강제 종료
- 스토어드 프로그램 생성 시 DEFINER를 타 사용자로 설정

- 스토어드 프로그램 (Stored Program)

: 프로시저, 함수, 트리거, 이벤트 같은 걸 말한다.

: DB 안에서 프로그램처럼 저장되어서
명령어 하나로 복잡한 작업을 자동으로 실행 가능.

- DEFINER

: 이 스토어드 프로그램 실행할 때, 누가 실행가인 것처럼
권한 적용할 거 정하는 것.

: DEFINER = 'user_name' @ 'host' => '설정 방법'

↳ 이 프로그램을 실행하는 사용자는 누구 권한으로
실행할 것인가?

ex)

시스템 관리자 (root)가 스토어드 프로시저 하나 만든다.

이걸 일반 사용자 Steve 권한으로 실행되게 하고 싶다.

```
CREATE DEFINER='steve' @'localhost' PROCEDURE my-procedure()
BEGIN
    : —, 작업 내용,
END;
```

시스템 계정과 일반 계정의 개념 도입

- └ DBA ... SYSTEM_USER 권한.
└ 일반 사용자 ... SYSTEM_USER 권한 부여 X
) => 이렇게 만들기 위해.

(참고)

사용자 : MySQL 서버 사용하는 주체
계정 : MySQL 서버에 로그인하기 위한 신별자

- MySQL 내장 계정.

1) 'mysql.sys' @ 'localhost'

: MySQL 8.0부터 기본으로 내장된 sys 스키마의 객체의 DEFINER로 사용되는 계정.

2) 'mysql.session' @ 'localhost'

: MySQL 플러그인이 서비스로 접속할 때 사용되는 계정

3) 'mysql.infoschema' @ 'localhost'

: information_schema 이 정의된 뷰의 DEFINER로 사용되는 계정

- └ 이 3개의 계정은 차후부터 관계 있다.
 의도적으로 풀지 않는 한 노출 걱정 X.

3. 2. 2 계정 생성

~ MySQL 5.7

└ GRANT 명령 → 친한 부여 + 계정 생성 가능.

MySQL 8.0 ~

└ CREATE USER 명령 → GRANT 명령. ... 주문해서 실행.

- 계정 생성 옵션

└ 계정의 인증 방식, 비밀번호

└ 비밀번호 관련 옵션 → 유효기간, 이력 개수, 재사용 불가 기간

└ 기본역할 (Role)

└ SSL 옵션

└ 계정 잠금 여부

CREATE USER 'user'@'%'

IDENTIFIED WITH 'mysql_native_password' BY 'password' : 사용자 인증방식 + pw 설정
REQUIRE NONE : 암호화된 SSL/TLS 캐싱 사용여부

PASSWORD EXPIRE INTERVAL 30 DAY : 유효기간 설정 옵션

ACCOUNT UNLOCK : 계정 잠금 여부

PASSWORD HISTORY DEFAULT : 한번 사용한 pw 재사용 못하게 설정하는 옵션

PASSWORD REUSE INTERVAL DEFAULT : 한번 사용한 pw 재사용 금지 기간 설정

PASSWORD REQUIRE CURRENT DEFAULT : 새로운 비밀번호 변경시 현재 비밀번호 필요로 할지 말지 결정하는 옵션.

~ MySQL 5.7

: Native Authentication 이 기본 인증.

MySQL 8.0 ~

: Caching SHA-2 Authentication 이 기본 인증.

:

SSL/TLS or RSA 키 페어 필요

=> MySQL 5.7 연결 방식과 다름.

보안수준 낮아지지만 기본 버전과의 호환성 고려 시

Caching SHA-2 보단 Native 인증 방식으로 설정 가능.

L MySQL 8.0에서도 Native Authentication 기본 인증 방식으로 설정 가능.

my.cnf 설정 파일에 추가.

: SET GLOBAL default_authentication_plugin = "mysql_native_password"

- IDENTIFIED WITH

- : 사용자 인증 방식 + PW 설정.
- : IDENTIFIED WITH 키가 반드시 인증방식 명시
- : 기본 인증 방식이면 IDENTIFIED BY 'password' 형식.

인증방식 \Rightarrow Native Pluggable Authentication, Caching SHA-2 Pluggable Authentication
PAM Pluggable Authentication, LDAP Pluggable Authentication

① Native

- : MySQL 5.7 까지 사용된 방식
- : 비밀번호에 대한 SHA-1 알고리즘 값 저장하고 해시 값 일치하는지 비교 인증.

② Caching SHA-2

- : MySQL 8.0 버전에서 보관된 인증방식
- : SHA-2 (256비트) 알고리즘 사용.
- : 저장된 해시 값이 보안에 더 중점을 두었다.
- : 내부적으로 Salt 키가 필요하며 수천번의 해시 계산 수행해서 결과 생성.
∴ 동일한 키 값에 대해서도 다른 결과 나옴.

L, But, 시간 소모적 ↑ + 성능 저하.

↓ ... 보관

해시 결과 값을 메모리에 캐시해서 사용.

- : 이 인증방식 사용하려면 SSL/TLS or RSA 키페어 반드시 사용.

L Client에서 접속 시, SSL 옵션 활성화.

③ PAM

- : 유닉스·리눅스·파스워드 or LDAP 같은 일부 인증 사용할 수 있게 해주는 인증 방식

L 엔터프라이즈 애플리케이션만 가능

④ LDAP Pluggable Authentication

- : LDAP 이용한 일부 인증 사용할 수 있게 해주는 방식

L 엔터프라이즈 애플리케이션 가능

~ MySQL 5.7

: Native Authentication 이 기본 인증 방식

MySQL 8.0 ~

: Caching SHA-2 Authentication 이 기본인증.

↳ SSL/TLS 또는 RSA 키페어 필요.

∴ 기존 5.7 까지의 연결 방식과는 다른 방식으로 접속.

=> 보안수준 낮아지지만 기존 버전과 호환성 고려한다면?

: Native Authentication 인증 방식.

ex)

Native Authentication을 기본 인증 방식으로 설정

: SET GLOBAL default_authentication_plugin = "mysql_native_password"

CREATE USER or ALTER USER 명령

=> MySQL 서버 계정을 생성 or 변경할 때 연결방식과
비밀번호 풀션, 주권 사용과 관련된 여러 풀션 설정 가능.

- REQUIRE

- : MySQL 서버에 접속할 때 암호화된 SSL/TLS 채널 사용 여부 설정.
설정하지 않으면 비암호화 채널로 연결

if) Caching SHA-2 Authentication 인증 방식 사용

=> '암호화' 된 채널만 가능.

- PASSWORD EXPIRE

- : 비밀번호 유효기간 설정하는 옵션.

별도 명시 X => default_password_lifetime 시스템 변수에 저장된 기간으로 유효기간이 설정.

개발자 or DBA or PW는 유효기간 설정 => 보안상 안전

응용 프로그램 접속용 계정이 유효기간 설정 => 위험.

:

왜?

- L 유효기간 만료 시, 응용 프로그램 더 이상 DB에 접속 X
- L 웹 사이트나 서비스가 장애 상태되어, 사용자에게 서비스 불가 오류 발생.
ex) 사이트가 잘 돌아가다가 DB 접속 오류로 다운되는 현상.
- L DB 계정 유효기간 적용 정신 X
L 관리자가 직접 연장해야하는데, 이를 깜빡하면 장애 발생 가능성 ↑

PASSWORD EXPIRE 옵션

- PASSWORD EXPIRE : 계정 생성 동시에 비밀번호의 만료 처리

- PASSWORD EXPIRE NEVER : 만료기간 X

- PASSWORD EXPIRE DEFAULT : default_password_lifetime 시스템 변수에 저장된 기간으로 비밀번호 유효기간 설정

- PASSWORD EXPIRE INTERVAL n DAY : 유효기간 오늘부터 n일자로 설정

- PASSWORD HISTORY

: 한번 사용한 비밀번호 재사용 X.

옵션 └ PASSWORD HISTORY DEFAULT

: password-history 시스템 변수에 저장된 개수만큼 비밀번호

이력 저장하며 저장된 이력이 남아 있는 비밀번호 재사용 X

└ PASSWORD HISTORY n

: 이력을 최근 n개까지만 저장.

: 저장된 이력에 남아 있는 비밀번호 재사용 X.

한 번 사용한 비밀번호 사용 못하게 하려면 이전에 사용한 비밀번호를 MySQL 서비스가 기억해야 한다.

이를 위해 MySQL 서비스는 mysql DB = password-history 테이블 사용.

- PASSWORD REUSE INTERVAL

: 한번 사용했던 비밀번호 재사용 금지기간 설정하는 옵션.

: 명시 X → password-reuse-interval 시스템 변수에 저장된 기간으로 설정.

옵션 └ PASSWORD REUSE INTERVAL DEFAULT

: password-reuse-interval 변수에 저장된 기간으로 설정.

└ PASSWORD REUSE INTERVAL n DAY

: n일자 이후에 비밀번호 재사용 할 수 있게 설정

- PASSWORD REQUIRE

- 비밀번호 만료되어 새로운 비밀번호로 변경할 때
현재 비밀번호를 필요로 할지 말지를 결정.

명시 X \Rightarrow password_require_current 시스템 변수의 값으로 설정.

옵션 CURRENT

: 비밀번호 변경할 때, 현재 비밀번호 먼저 입력하도록 설정.

OPTIONAL

: 변경 시, 현재 비밀번호 입력하지 않아도 되도록 설정

DEFAULT

: password_require_current 시스템 변수의 값으로 설정.

- ACCOUNT LOCK / UNLOCK

- 계정 생성 시, or ALTER USER 명령 사용해 계정 정보
변경할 때 계정 사용 못하게 잠금지 여부

옵션 LOCK

: 계정 사용 못하게 잠금

UNLOCK

: 잠긴 계정 다시 사용 가능 상태로 잠금해제.

3.3 비밀번호 관리

3.3.1 고수준 비밀번호

MySQL 서비스 비밀번호 → 기호기간 or 이력관리 통한 개사용 금지
+
글자조합 강제 and 금칙어 설정) 가능

pw 기호성 체크 → validate_password 컴포넌트 이용.

```
INSTALL COMPONENT 'file:///component-validate_password';
```

비밀번호 정책 —— LOW

: 비밀번호 길이만 검증

MEDIUM

: 길이 검증 + 숫자·대소문자·특수문자·비합 검증.

) 선택

STRONG

: MEDIUM 검증 + 금칙어 검증

비밀번호 길이 : validate_password.length 시스템 변수에 설정된 길이 이상 사용해야함.

숫자·대소문자·특수문자 : validate_password.mixed-case-count

validate_password.number-count

validate_password.special-char-count) 변수에 설정된 글자 수 이상 포함했는지 검사.

금칙어 : validate_password.dictionary-file 시스템 변수에 설정된

사전 파일이 명시된 단어 포함하고 있는지 검증.

'금칙어 파일' : 텍스트 파일로 작성하면 된다.

그리고 MySQL 서버에 등록.

=> 단, STRONG인 경우에 작동.

```
SET GLOBAL validate_password.dictionary-file = 'prohibitive-word.data';
```

```
SET GLOBAL validate_password.policy = 'STRONG';
```

MySQL 5.7 : 플러그인 형태

MySQL 8.0 : 컴포넌트 형태.

- 플러그인

: 기존 MySQL 서버에 특정 기능을 플러그인처럼 추가/제거하는 구조.

=> INSTALL PLUGIN / UNINSTALL PLUGIN 사용.

L 설치 위치: plugin 디렉토리에 .so, .dll 파일 존재.

(장) : 비교적 간단 설치/제거
 : 안정적

(단) : 특별성↓
 : 내부 API 접근 제한적
 : 보안 확장성면에서 부족.

- 컴포넌트

: MySQL 8.0부터 새롭게 도입.

: 모듈화된 확장구조

: 서비 기능을 더 가연하게 확장하고 통합하기 위한 구조

=> INSTALL COMPONENT 'file://component-validate-password';

L 설치 위치: component 디렉토리 .so 형태

(장) : 기능 간 의존성 관리와
 인터페이스 명확하여
 모듈화 험수

(단) : 구조 복잡
 설정 관리 학습 곡선 존재.

3. 2. 2 야종 비밀번호

많은 헉프로그램 서비스들이 공동으로 DB 서버 사용.

↳ 계정 정보 공동 사용 경우 ↑.

∴ DB 서버 계정 정보 변경 쉽지 않음.

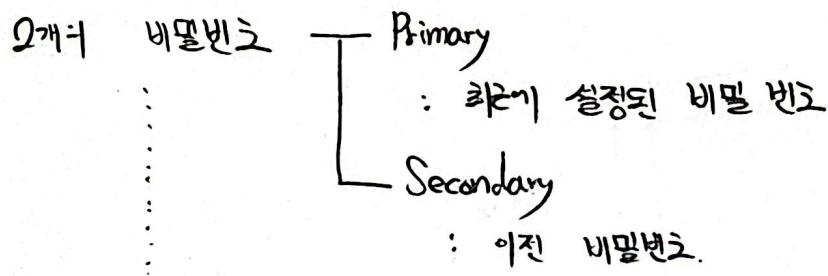
계정의 비밀번호는 서비스가 실행 중인 상태에서 변경 불가.

∴ 몇 년동안 사용하는 경우 ↑

이러한 문제 해결 위해 MySQL 8.0 버전부터

비밀번호 2개의 값을 동시에 사용할 수 있는 기능 추가.

⇒ Dual password. (2개의 비밀번호 중 하나만 열차하면 통과)



RETAIN CURRENT PASSWORD 옵션 추가.

ex)

1. pw를 1234로 설정.

2. pw를 4321로 변경.

↳ 이때 옵션으로 RETAIN CURRENT PASSWORD.

Primary pw는 1234로 변경. Secondary pw는 빈 상태.

↓
4321이 Primary pw로 바뀜. 1234가 Secondary 3 된다.

: 아무거나 입력해도 로그인 가능.

추후. 노안기해 삭제하는 것이 좋음.

↳ ALTER USER 'root'@'localhost' DISCARD OLD PASSWORD;

3.4 권한 (Privilege)

~ MySQL 5.7 → 글로벌 권한과 객체 단위의 권한으로 구분.

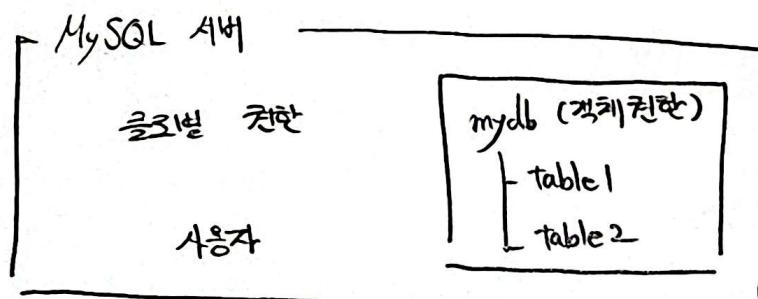
글로벌 권한

- : DB나 Table 이외의 객체에 적용되는 권한. ... GRANT 명령 시 특정 객체 명시X

객체 권한

- : DB나 Table 제외하는데 필요한 권한. ... GRANT 명령 시 특정 객체 명시O

글로벌 권한 => 모든 객체 권한
ALL 권한 {
 객체 권한 => 모든 글로벌 권한.



MySQL 8.0 ~

: 동적 권한 추가.

L MySQL 서비스가 시작되면서 동적으로 생성하는 권한.

정적 권한 : MySQL 서비스 소스코드에 고정적으로 명시되어 있는 권한.

MySQL 5.7 까지는 SUPER 권한이 DB 관리 외에 꼭 필요한 권한.

But, 8.0부터는 SUPER이 꼬개져 동적권한으로 분산.

L 백업 관리자 / 복제 관리자 개별로 꼭 필요한 권한만 부여.

권한 부여 => GRANT.

첫 권한의 특성이 따라 GRANT 명령의 ON 절에 명시되는 오브젝트의 내용 바껴야 한다.
DB or Table

=> GRANT privilege-list ON db.table TO 'user'@'localhost';
GRANT 권한리스트 ON 대상 TO 사용자.

8.0부터는 존재하지 않는 사용자에 대해 GRANT 명령 실행 시
'에러' 발생.

∴ 사용자 먼저 생성 후 GRANT 명령으로 권한 부여.

GRANT OPTION 권한은 따로 GRANT OPTION이라고 명시X.

L> GRANT SELECT,INSERT ON mydb.* TO 'user'@'localhost' WITH GRANT OPTION;
붙여야 한다.

GRANT OPTION

: 이 사용자가 자신이 부여 받은 권한을 다른 사용자에게도 부여할 수 있는 권한.

비교.

GRANT SELECT ON mydb.* TO
(
 ① 'user'@'localhost';
 ② 'user'@'localhost' WITH GRANT OPTION;

① : user는 SELECT 할 수 있지만 다른 사용자에게 SELECT 권한 줄 수 X

② : user는 SELECT 할 수 있고, 다른 사용자에게 SELECT 권한 줄 수 있음.

privilege-list 뒤에는 구분자 (,) 사용하여 여러개 동시에 명시 가능.

TO 카운트 뒤에는 '권한 부여할 대상 사용자' 명시

ON 카운트 뒤에는 어떤 DBか 어떤 오브젝트에 권한 부여할 지 결정.

1) 글로벌 권한.

=> GRANT SUPER ON *.* TO 'user'@'localhost';

→ MySQL 서버 전체.

글로벌 권한은 특정 DB나 테이블에 부여될 수 X.

권한 부여 시 ON 절에 항상 *.* 사용.

ex)

CREATE USER, CREATE ROLE 같은 글로벌 권한은

DB 단위나 오브젝트 단위로 부여할 수 있는 권한 X.

∴ 항상 *.* 로 모든 대상을 사용할 수 있음.

2) DB 권한.

=> GRANT EVENT ON *.* TO 'user'@'localhost';

=> GRANT EVENT ON employees.* TO 'user'@'localhost';

'DB권한' => 특정 DB에 대해서만 권한 부여

or

서버에 존재하는 모든 DB에 대해 권한 부여

∴ ON 절에 *.*이나 employees.* 모두 사용 가능.

DB권한만 부여하는 경우 (DB권한은 테이블에 대해 부여X)

employees.department과 같이 테이블까지 명시 X.

DB권한은 서버의 모든 DB에 적용할 수 있으므로 대상에 *.* 사용.

3.5 역할 (Role)

MySQL 8.0부터 전한 쪽에서 역할 사용 가능.

실제 MySQL 서버 내부적으로 역할은 계정과 같은 모듈

CREATE ROLE 명령 이용해 역할 정의 가능.

=> CREATE ROLE

role_emp_read,
role_emp_write;

) 빈 접두어만 있는 역할 정의.
:: GRANT로 권한 부여.

=> GRANT SELECT ON employees.* TO role_emp_read;

GRANT INSERT, UPDATE, DELETE ON employees.* TO role_emp_write;

L, role_emp_read 역할에 SELECT 권한

role_emp_write 역할에 INSERT, UPDATE, DELETE 권한

기본적으로 역할은 그 자체로 사용x.

계정이 부여해야 한다.

=> GRANT role_emp_read TO reader@'127.0.0.1';

GRANT role_emp_read, role_emp_writer TO writer@'127.0.0.1';

L, reader 계정: role_emp_read 역할

writer 계정: " + role_emp_writer 역할.

이 상태에서 reader, writer 계정으로 로그인하고 employees DB 데이터

조회·변경 시 전한 없다는 알림 뜬다.

실제 역할은 부여되어있지만 활성화해야 함 => SET ROLE 명령

3) 테이블 권한.

- ① GRANT SELECT, INSERT, UPDATE, DELETE ON *.* TO 'user'@'localhost';
- ② " ON employees.* TO 'user' @'localhost' ;
- ③ " ON employees.department TO 'user' @'localhost' ;

서버의 모든 DB에 권한 부여 가능

특정 DB-1 고노젝트 대해서만 권한 부여 가능.

특정 DB-2 특정 테이블만 권한 부여 가능.

i.) 테이블의 특정 칼럼에 대해서만 권한 부여.

칼럼에 부여할 수 있는 권한 => INSERT, UPDATE, SELECT.

각 권한 뒤 칼럼 명시 해야함.

GRANT SELECT, INSERT(dept-name) ON employees.department ~

View도 하나의 테이블로 인식.

View 만들어두면 View 칼럼에 대해 권한 체크하지 않고

View 자체에 대한 권한만 체크.

개정 or 권한이 부여된 권한이 역할 확인

=> SHOW GRANT or mysql DB-1 권한 관련 테이블 참조.