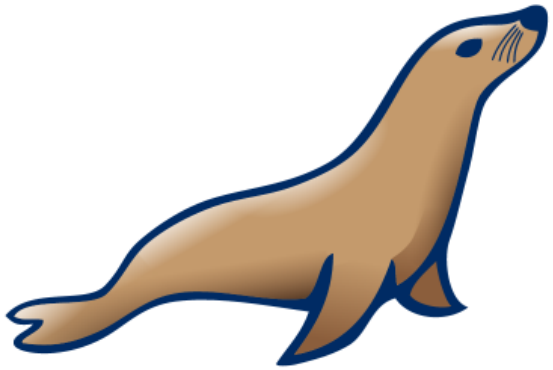


MariaDB

DBMS	제작사	운영체제	기타
MariaDB	MariaDB	Unix, Linux, Windows	오픈 소스(무료)
MySQL	Oracle	Unix, Linux, Windows, Mac	오픈 소스(무료), 상용
PostgreSQL	PostgreSQL	Unix, Linux, Windows, Mac	오픈 소스(무료)
Oracle	Oracle	Unix, Linux, Windows	상용 시장 점유율 1위
SQL Server	Microsoft	Windows	
DB2	IBM	Unix, Linux, Windows	메인프레임 시장 점유율 1위
Access	Microsoft	Windows	PC용
SQLite	SQLite	Android, iOS	모바일 전용, 오픈 소스(무료)

PYK

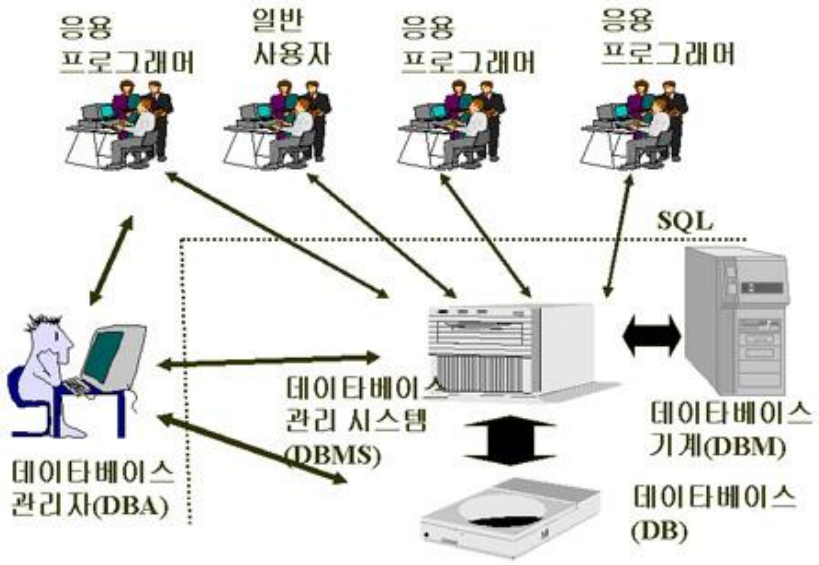
데이터(Data)란 ?

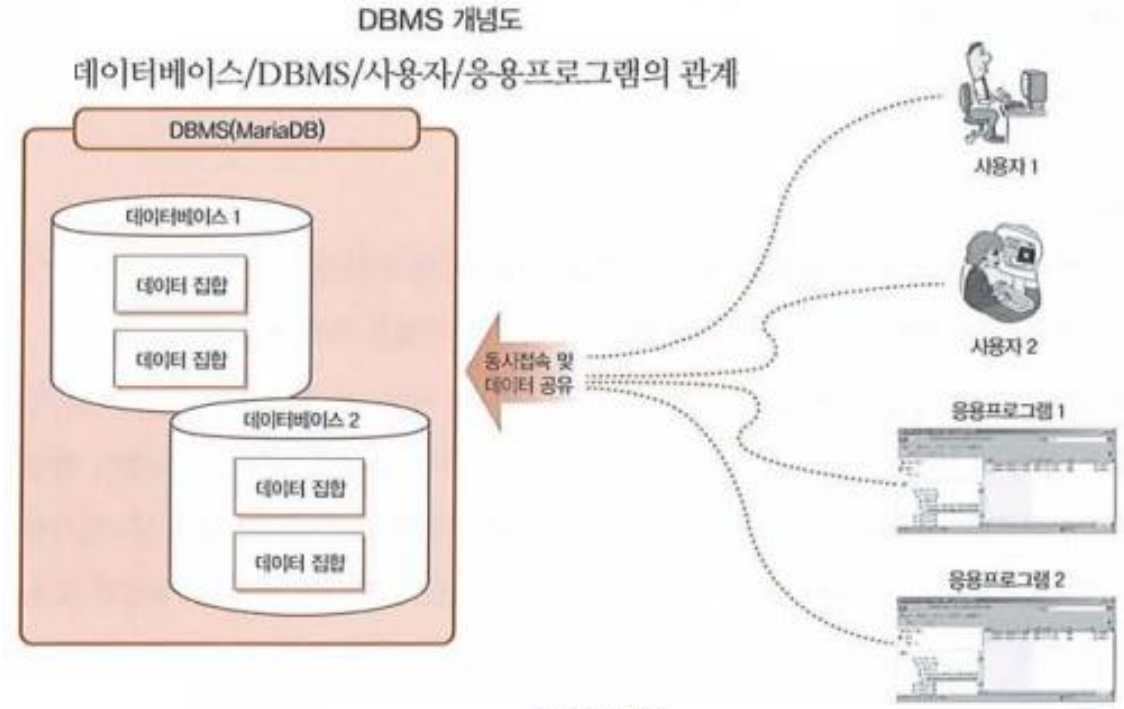
사람들에 의해 수집된 사실이나 값이 정형화되고 기록될 만한 가치가 있다고 판단되는 현실 세계의 어떤 현상이나 사건에 대한 구체적인 묘사

정보 처리 시스템이란?

조직이나 개인이 필요로 하는 정보를 제공하는 시스템으로서, 데이타베이스와 DBMS, 이들을 이용하는 응용 프로그램 및 지원 소프트웨어로 이루어진 시스템

데이타베이스 시스템 : 구조

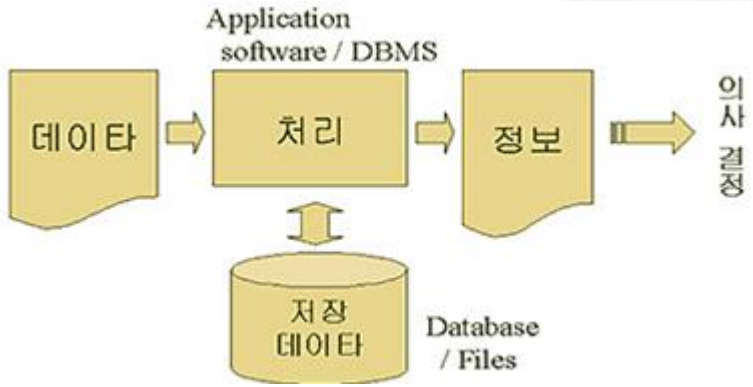




화일 vs. 데이터베이스

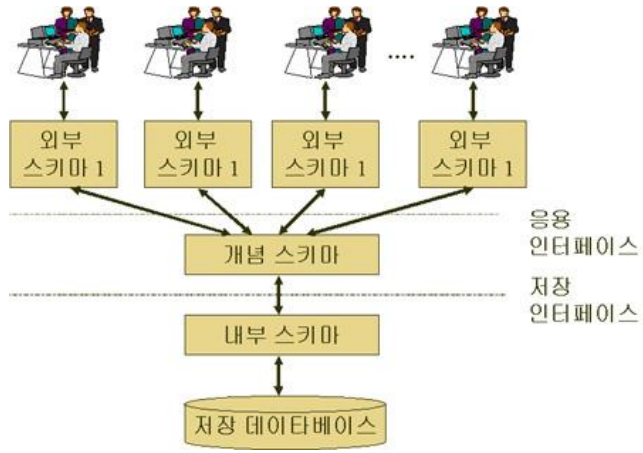
파일(File) : 물리적 비트들의 연속 (사용자 측면)
서로 다른 언어에서 각각의 화일 연산자를 이용하여 조작 가능
문제점 : 데이터 종속, 데이터 중복

데이터베이스 : 논리적 구조의 데이터의 집합 (사용자 측면)
표준 언어를 이용하여 데이터 조작 가능



DBS : Database - 정의

어느 한 조직의 여러 응용 시스템들이 공용할 수 있도록 통합, 저장된 운영 데이타 집합
(통합된 데이터, 저장된 데이터, 운영 데이터, 공용 데이터)



DBS : Database - 특징

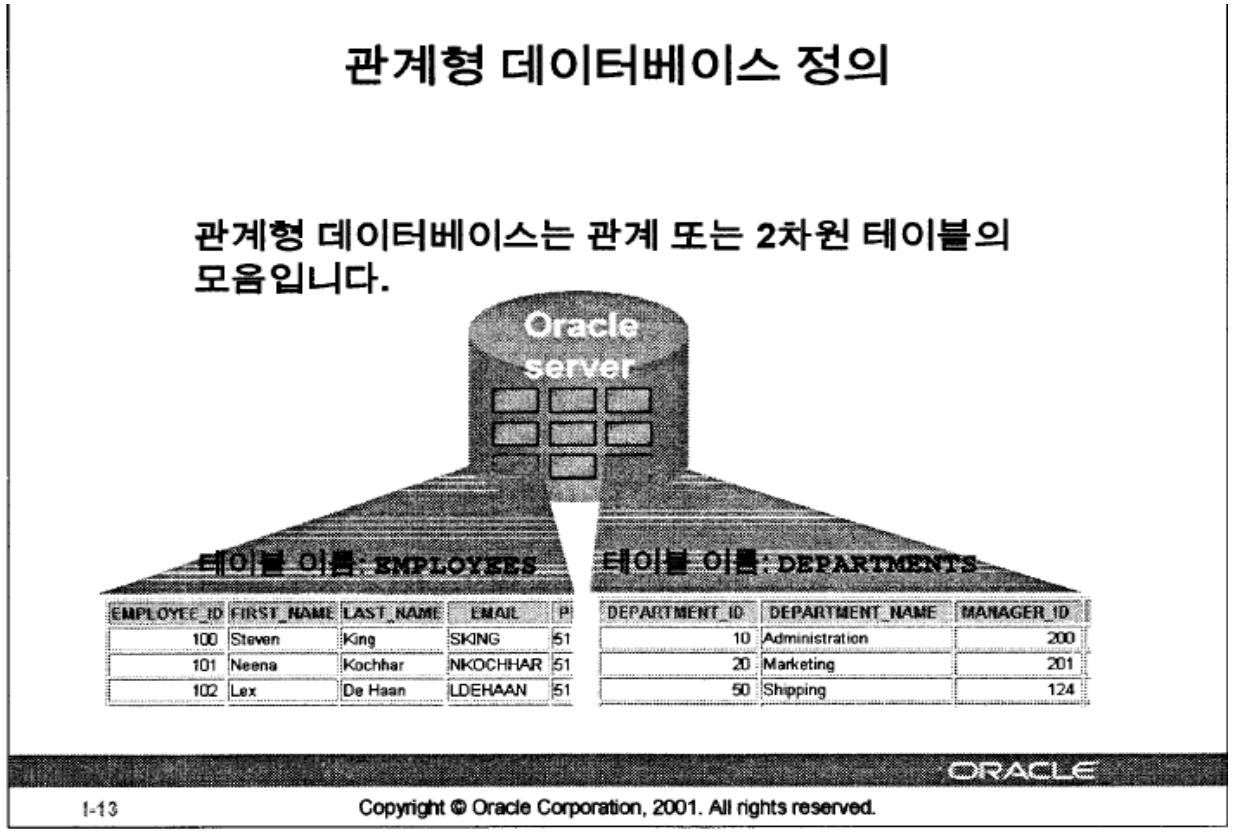
- 실시간 접근성(Real-time accessibility) : 사용자의 요구에 대한 즉각적인 응답(Response)
- 계속적인 변화(Continuous evolution) : 삽입,삭제,갱신 작업이 수시로 발생
- 동시 공용(Concurrent sharing) : 여러 사용자가 동시에 자기가 원하는 데이터에 접근 가능
- 내용에 의한 참조(Content reference) : 데이터에 대한 조건으로 원하는 결과를 검출

스키마 : 데이타베이스의 논리적 정의

- 외부 스키마(External schema)
  각 사용자의 입장에서 본 데이타베이스 구조, 사용자마다 서로 다른 데이타베이스 스키마를 가짐, 개념 스키마에 대한 서브스키마(sub schema)
- 개념 스키마(Conceptual schema)
  조직 전체의 입장에서 본 데이타베이스 구조, 한 개의 스키마만 존재하며, 서로 다른 사용자가 공유, 데이터 객체(개체,관계), 제약조건에 대한 명세를 유지
- 내부 스키마(Internal schema)
  저장 장치의 입장에서 본 데이타베이스 구조, 각 데이터 객체의 저장 구조를 표현함. 내부 레코드의 형식, 인덱스의 유무, 저장 데이터 항목의 표현 방법

DBS : DBMS - 정의

사용자와 데이터베이스 사이에 위치하여 사용자의 요구에 따라 데이터베이스를 조작하고 제어하는 기능을 제공하는 소프트웨어
=> Data + Hardware + Software + User



관계형 데이터베이스 정의

관계형 데이터베이스는 관계 또는 2차원 테이블을 사용하여 정보를 저장합니다.

예를 들어, 회사 내 모든 사원에 대한 정보를 저장해야 할 경우, 관계형 데이터베이스에서는 사원 테이블, 부서 테이블, 급여 테이블 등 여러 테이블을 생성하여 사원에 대한 각 정보를 저장합니다.

관계형 데이터 모델이란?

E.F.Codd : "a Relational Model of Data for Large Shared Data Banks." Communications of the ACM

테이블(Relation:관계)의 집합

'테이블, 행' => '릴레이션, 튜플'

궁극적으로 테이블, 컬럼 들과 같은 DB객체에 관계를 정의해 DB의 무결성과 성능을 높이자는 의도다.

관계 데이터베이스와 파일 시스템의 비교

관계 데이터베이스	파일 시스템
릴레이션	파일
튜플	레코드
애트리뷰트	필드

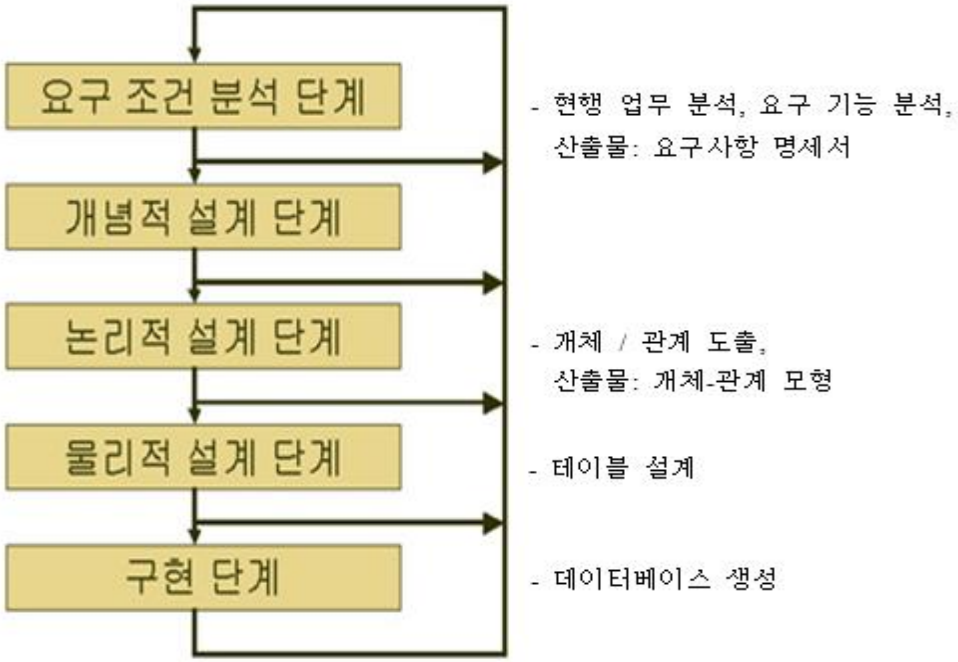
dept_no	dept_name	dept_loc	dept_tel
10	총무부	서울	02-504-1234
20	영업부	서울	02-504-1235
30	전산부	서울	02-631-1921
40	관리부	인천	032-553-4980

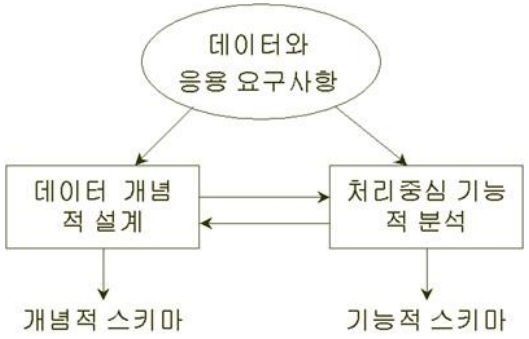
** 데이터 모델링 **

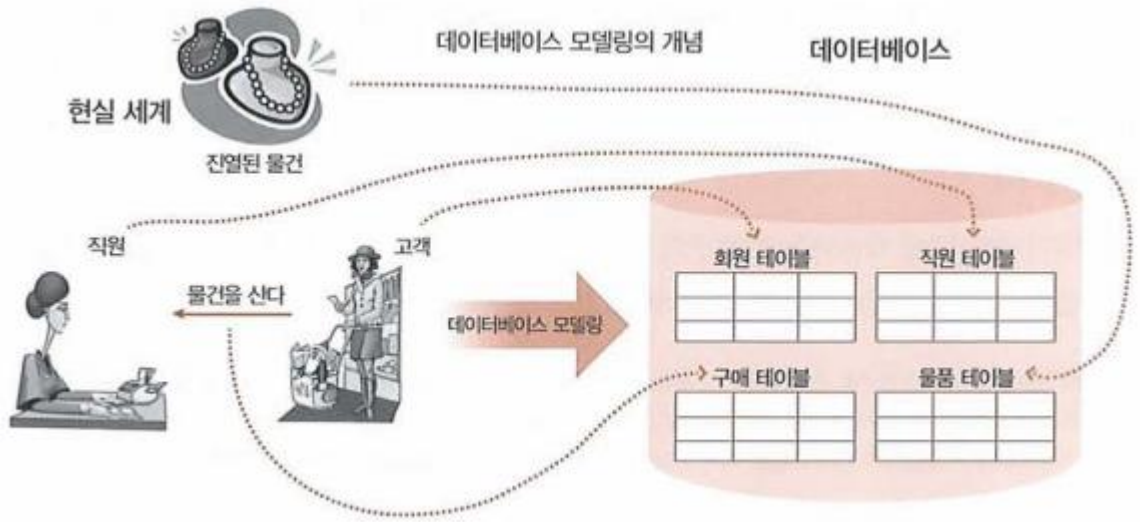
데이터베이스를 새롭게 구축하기 위해서 사용자의 요구사항을 잘 분석해 이를 추상화하여 문서화 하는 과정을 데이터 모델링이라 한다. 현실세계를 데이터베이스로 표현하기 위해 개념적인 구조와 논리적인 구조를 거쳐 실제 데이터를 저장할 수 있는 물리적인 구조로 변환되어야 하는데 이런 과정을 데이터베이스 설계라 하고, 이 과정에서 중요 핵심은 모델이며, 구성요소에 따라 논리적모델과 물리적 모델로 구분된다.

* 데이터 모델링을 하는 주요 이유

업무 정보를 구성하는데 기초가 되는 정보를 일정한 표기법으로 표현함으로써 정보시스템 구축 대상이 되는 업무내용을 정확하게 분석한다.







* 논리적 데이터베이스 설계.

■ 테이블 및 Entity(개체) : 레코드와 같은 의미
■ 각 테이블의 컬럼명(Attribute) : 필드
■ 유일값 요구, null 허용여부, 컬럼에 저장될 자료형 등의 컬럼 특성
■ 기본키
■ 관계 : 개체간의 연관성을 의미한다. 즉, 어떤 테이블의 행은 다른 테이블의 하나 이상의 행에 종속적이다.
이러한 테이블 사이의 종속성을 관계라 한다.
관계를 정의하기 위해서는 다른 테이블의 기본키를 참조하는 컬럼을 참조키라 한다.

* 물리적 데이터베이스 설계

: 개발에 사용할 DB를 선정하여 특정 DB로 구현될 수 있도록 구체적으로 설계하는 과정이며, 이의 산출물은 테이블 명세다.

- 일대일 관계
  기본 테이블의 각 행은 참조 테이블의 하나의 행과 관계된다.
  일대일 관계는 참조키가 유일하게 정의됨으로써 구현된다. (중복 불가)

- 일대다 관계
  기본 테이블의 각 행은 참조 테이블의 하나 또는 그 이상의 행들과 관계된다.
  예를 들어 1명의 고객은 많은 주문을 할 수 있다. 그러나 어떤 하나의 주문은 많은 고객에 의해 정해 질 수 없다.

- 다대다 관계
  한 테이블의 많은 행들은 다른 테이블의 많은 행들과 관계되어 진다.
  예를 들어 '홍길동'이란 저자는 여러 종류의 책을 쓸 수 있고, '오라클'이란 제목의 책은 여러 저자들에 의해 쓰여질 수 있다. 두 테이블 간의 다대다 관계는 다른 세번째 테이블의 생성 및 각각의 초기화 테이블로부터 연결된 일대다 관계 생성에 의해 구현된다.

■ 정규화 (Nomalization)

여러 테이블 간의 관계를 설계할 때 일반적으로 어떤 논리적 비일관성들이 발생한다.
정규화 과정은 이런 비일관성이 일어나지 않도록 한다.
정규화는 일관성이 있는 데이터베이스 설계를 생성하기 위해 테이블, 키, 컬럼, 관계 등을 정제하는 과정이다.
정규화는 테이블에 많은 테스트 적용을 통해 수행된다.
정규화는 5레벨까지 있을 수 있는데 보통은 제1정규화에서 제3정규화까지 적용한다.

[데이터베이스 정규화 개념 및 방법]

고객 방문 기록

고객 이름	출생연도	주소	연락처	구매한 물건	단가(천 원)	수량
이승기	1987	서울	011-111-1111			
김범수	1979	경남	011-222-2222	운동화	30	2
김범수	1979	경남	011-222-2222	노트북	1000	1
김경호	1971	전남	019-333-3333			

고객 이름	출생연도	주소	연락처	구매한 물건	단가(천 원)	수량
이승기	1987	서울	011-111-1111			
김경호	1971	전남	019-333-3333			
윤종신	1969	경남	안 남김			
김범수	1979	경남	011-222-2222	운동화	30	2
김범수	1979	경남	011-222-2222	노트북	1000	1
조용필	1950	경기	011-444-4444	모니터	200	1

정규화 시 지켜야 할 사항

1) 레코드는 고유하게 구별되어야 한다.
2) 필드는 고유하고 최소한의 정보만 가지도록 해야 한다.
3) 한 테이블에는 하나의 주제를 가지며 기능적 단위로 구성한다.
4) 필드는 서로 독립적이어야 한다.
5) 다른 필드의 정보를 가지고 계산하여 알 수 있는 필드는 없어야 한다.

사용 용도에 따른 SQL(사용자와 DB 간 연결을 가능하게 하는 표준 검색어) 분류

- DDL(데이터 정의어) : DB의 논리적 구조를 정의 — create, alter, drop, rename, truncate…
- DML(데이터 조작어) : 데이터 조작에 사용 — select(DQL), insert, update, delete…
- DCL(데이터 제어) : DB에 접근 권한, 시스템의 트랜젝션 등을 관리 — grant, revoke, commit, rollback, savepoint…

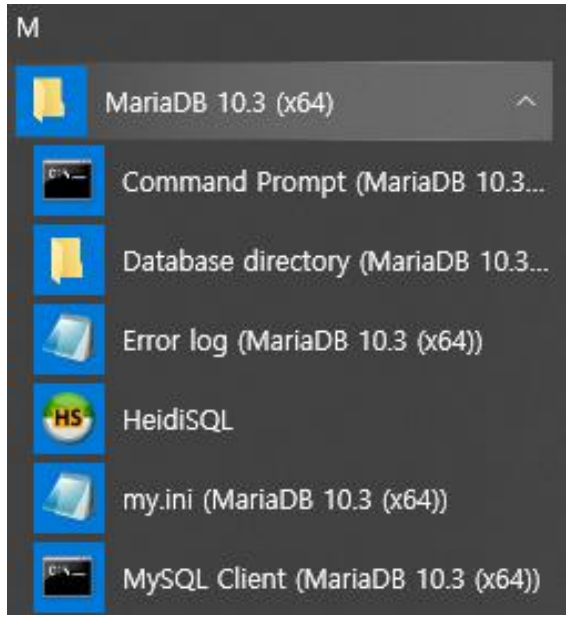
* MariaDB 설치

https://downloads.mariadb.org/mariadb

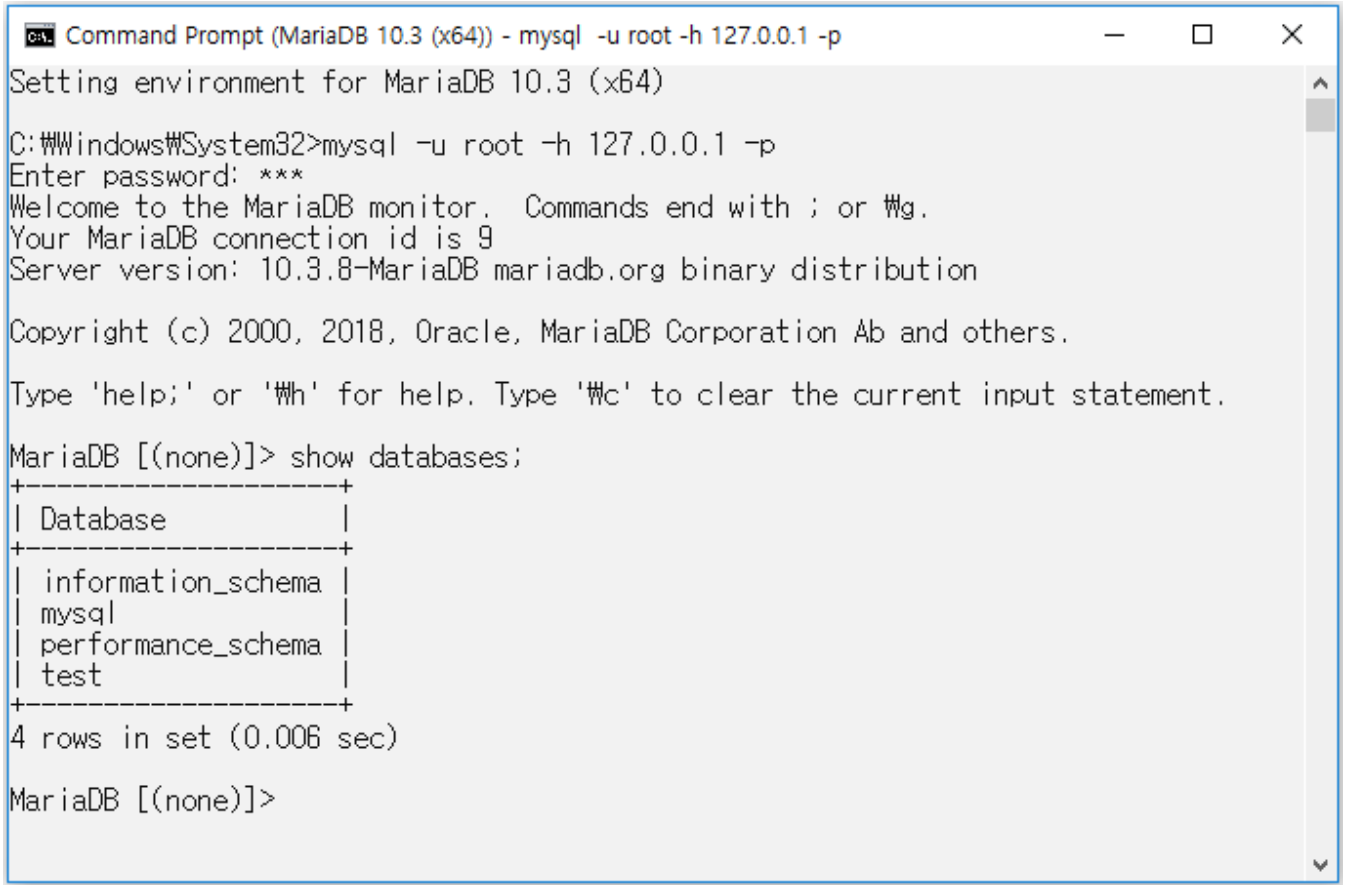
32/64비트 환경에 맞게 msi 설치파일을 다운로드 한다.

mariadb-10.2.14-winx64.zip	ZIP file	Windows x86_64	194.8 MB	Checksum Instructions
mariadb-10.2.14-winx64.msi	MSI Package	Windows x86_64	55.0 MB	Checksum Instructions

** sql 실행을 위한 창 실행 **



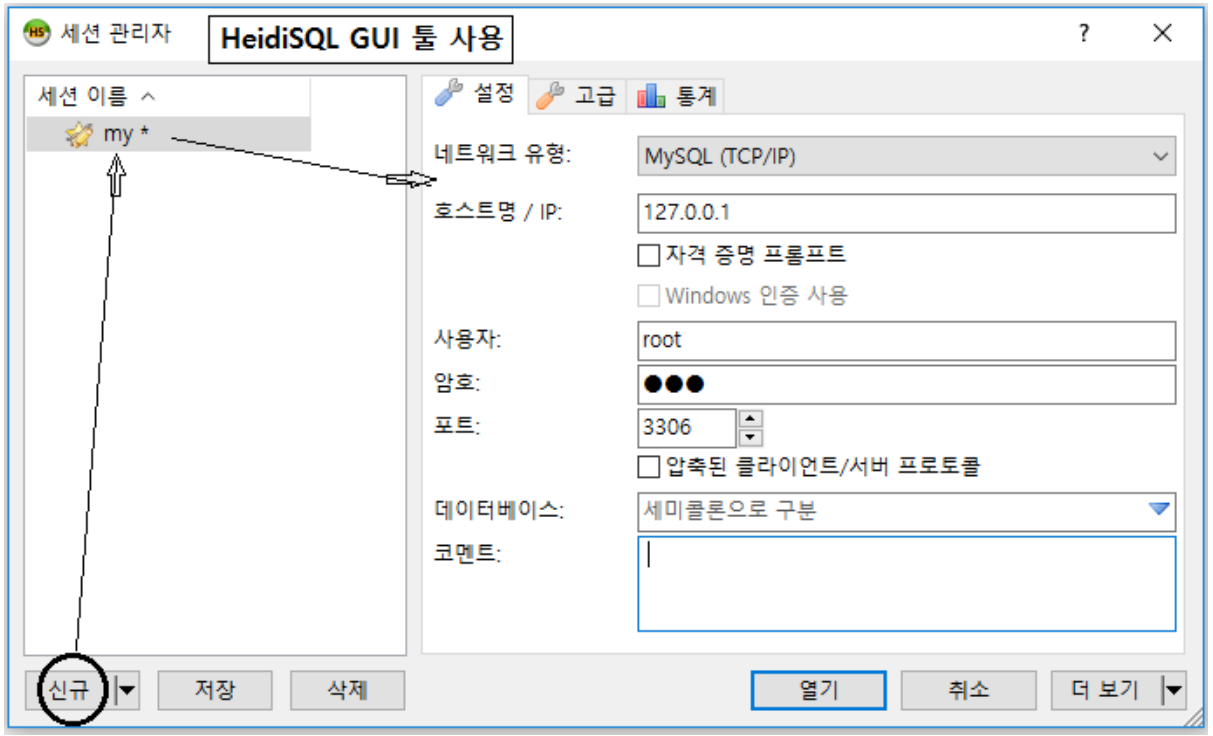
방법1) DOS 창에서 수행



기본 데이터 베이스

MariaDB [mysql]> show databases;
information_schema <-- 메타 데이터가 저장되는 데이터베이스
mysql <-- 사용자 인증 정보, stored program, 이벤트 정보 등이 저장되는 기본 데이터베이스
performance_schema <-- 각종 이벤트, 잠금, 잠금 대기 등의 정보를 기록하는 테이블 구조 저장,
서버 기동될 때 마다 초기화 되며, 데이터는 메모리에 저장
test <-- 테스트용 데이터베이스(삭제하는 것이 보안상 좋음)

방법2) HeidiSQL tool을 사용해서 SQL 수행 화면 실행



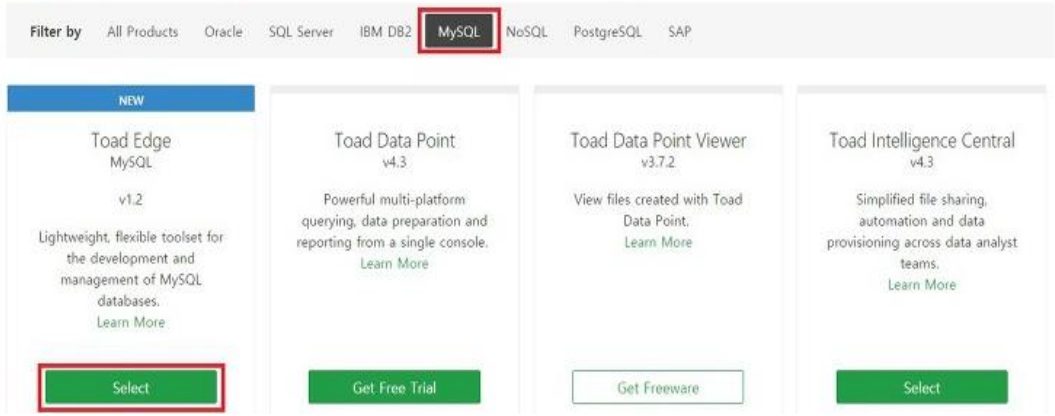
기타 툴

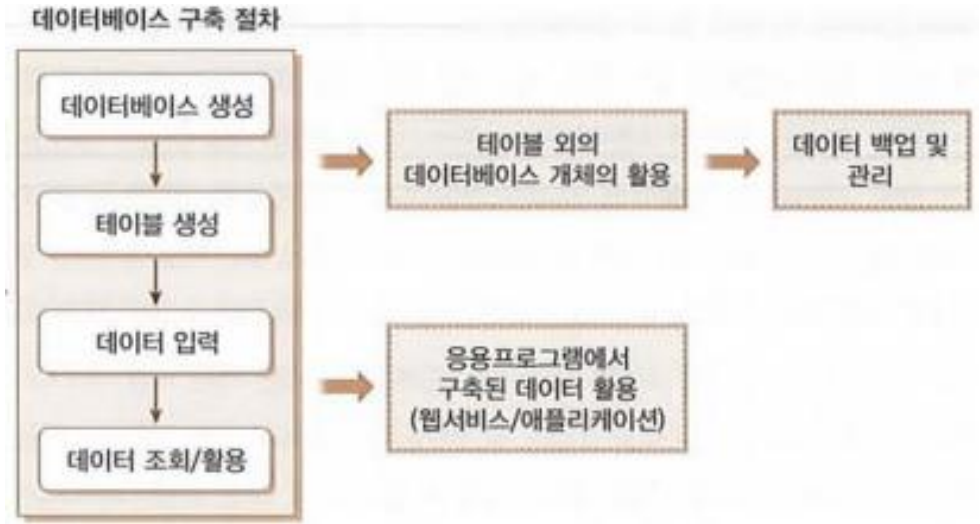
Toad For MySQL 설치

Workbench는 무겁기 때문에 가벼운 Toad를 설치한다.

https://community.toadworld.com/p/downloads 접속

MySQL 선택 > Toad Edge 선택





* 데이터베이스 생성 및 삭제

```
MariaDB [(none)]> show databases;
MariaDB [(none)]> create database demo;   -- demo 데이터베이스 생성
MariaDB [(none)]> drop database demo;     -- demo 데이터베이스 삭제
```

* 레코드 간단히 다루기 * 자료형의 종류

[숫자형 데이터 타입]

데이터 타입	의미	크기	설명
TINYINT	매우 작은 정수	1 byte	-128 ~ 127 (부호없이 0 ~ 255)
SMALLINT	작은 정수	2 byte	-32768 ~ 32767
MEDIUMINT	중간 크기의 정수	3 byte	-(-8388608) ~ -1(8388607)
INT	표준 정수	4 byte	~ -1
BIGINT	큰 정수	8 byte	- ~ -1
FLOAT	단정도 부동 소수	4 byte	-3.40E+45 ~ 3.40E+45 (no unsigned)
DOUBLE	배정도 부동 소수	8 byte	-1.7976E+320 ~ 1.7976E+320 (no unsigned)
DECIMAL(m,n)	고정 소수	m과 n에 따라 다르다	숫자 데이터지만 내부적으로 String형태로 저장됨. 최대 65자.
BIT(n)	비트 필드	m에 따라 다르다	

[날짜형 데이터 타입].

데이터 타입	의미	크기	설명
DATE	CCYY-MM-DD	3 byte	1000-01-01 ~ 9999-12-31
DATETIME	CCYY-MM-DD hh:mm:ss	8 byte	1000-01-01 00:00:00 ~ 9999-12-31
TIMESTAMP	CCYY-MM-DD hh:mm:ss	4 byte	1970-01-01 00:00:00 ~ 2037
TIME	hh:mm:ss	3 byte	-839:59:59 ~ 839:59:59
YEAR	CCYY 또는 YY	1 byte	1901 ~ 2155

[문자형 데이터 타입]

데이터 타입	의 미	크 기	설 명
CHAR(n)	고정 길이 문자열	n byte	
VARCHAR(n)	가변 길이 문자열	Length + 1 byte	
BINARY(n)	고정 길이 이진 문자열	n byte	
VARBINARY(n)	가변 길이 이진 문자열	Length + 1 byte or 2 byte	
TINYBLOB	매우 작은 BLOB(Binary Large Object)	Length + 1 byte	
BLOB	작은 BLOB	Length + 2 byte	최대크기 64KB
MEDIUMBLOB,	중간 크기 BLOB	Length + 3 byte	최대크기 16MB
LONGBLOB	큰 BLOB	Length + 4 byte	최대크기 4GB
TINYTEXT	매우 작은 문자열	Length + 1 byte	
TEXT	작은 문자열	Length + 2 byte	최대크기 64KB
MEDIUMTEXT	중간 크기 문자열	Length + 3 byte	최대크기 16MB
LONGTEXT	큰 문자열	Length + 4 byte	최대크기 4GB

- 테이블 생성

형식) create table 테이블명(칼럼명 자료형... constraint...)

```
MariaDB [(none)]> use test
Database changed
MariaDB [test]> create table test(no int, name varchar(10));   -- 테이블 생성, 한글 입력 불가.
MariaDB [test]> show tables;
MariaDB [test]> drop table test;    -- 테이블 제거
```

HeidiSQL 툴 사용하기

```
create table test(no int primary key, name varchar(10) not null,
                  tel varchar(15), inwon int, addr text) charset=utf8;
describe test;   -- 테이블의 구조 확인
DESC test;
```

COLUMNS (6×5)

Field	Type	Null	Key	Default
no	int(11)	NO	PRI	(NULL)
name	varchar(10)	NO		(NULL)
tel	varchar(15)	YES		(NULL)
inwon	int(11)	YES		(NULL)
addr	text	YES		(NULL)

- 자료 추가

형식) insert into 테이블명(칼럼명…) values(입력자료…)

```
insert into test(no,name,tel,inwon,addr) values(1,'인사과','123-1111',5,'삼성동');
```

- 자료 수정

형식) update 테이블명 set 칼럼명 = 수정할값,… where 조건

```
update test set inwon=7 where no=1;
```

- 자료 삭제

형식1) delete from 테이블명 (칼럼명…) where 조건 -- 부분적으로 레코드 삭제

형식2) truncate table 테이블명 -- where 조건을 주지 않음. 전체 레코드 삭제

* 무결성 제약조건 : 잘못된 자료의 입력을 막고자 다양한 입력 제한 조건을 줄 수 있다.

- Domain 제약조건 : 레코드 생성 시 각 칼럼의 이름과 성격, 크기 null 값 허용 여부
- 기본키 제약조건(Primary key)
- 사용자 정의 제약조건 : check, unique, foreign key...

기본 키 제약 조건 : 중복 레코드 입력 방지, PK칼럼을 기준으로 오름차순 정렬, NULL 허용 x

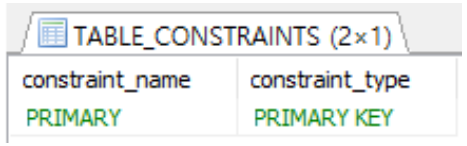
방법1) 컬럼 레벨 - 각 컬럼 선언 시 제약조건 부여 (PK의 이름은 컴이 부여)

```
create table aa(bun int(5) primary key,irum char(10));
-- 제약조건 확인
SELECT constraint_name, constraint_type FROM information_schema.table_constraints WHERE table_name = 'aa';
```

방법2) 테이블 레벨-- 칼럼을 모두 정의한 후 제약조건을 부여

```
create table aa(bun int,irum char(10), constraint aa_bun_pk primary key(bun));
drop table aa;
```

- 제약조건 추가 : alter table 테이블명 add constraint 제약조건명 제약조건 (대상 칼럼명)
- 제약조건 제거 : alter table 테이블명 drop constraint 제약조건명
- 칼럼의 비활성화 및 활성화 : alter table 테이블명 disable{enable} constraint 제약조건명


TABLE_CONSTRAINTS (2×1)

constraint_name	constraint_type
PRIMARY	PRIMARY KEY

Check 제약조건 : 입력되는 자료의 특정 컬럼 값 검사.

```
create table aa(bun int,irum char(10),nai int(2) check(nai >= 20));
```

Unique 제약조건 : 특정 컬럼의 동일한 값 입력 불허.

```
create table aa(bun int, irum char(10) unique);
```

외부키(참조키, Foreign key) 제약조건 : 다른 테이블의 컬럼 값을 참조 (fk의 대상은 pk로 한다)

- on delete cascade : 부모 테이블의 행이 삭제되는 경우 자식 테이블의 종속 행을 삭제.

Default : 특정 컬럼에 초기값 부여. 열 값이 없는 행을 입력할 경우 열에 NULL 값이 입력되는 것을 방지할 수 있다.

```
create table aa(bun int AUTO_INCREMENT,irum char(10),juso varchar(20) default '여의도동', primary key(bun))charset=utf8;
```

INDEX (색인) : 검색 속도를 증진시키기 위해 컬럼에 색인 부여.

PK 컬럼은 자동으로 색인이 부여된다.

index를 사용해야 하는 경우
- 레코드 수가 많을 경우
- 조건 절인 where에 자주 사용되는 컬럼
- join에 자주 사용되는 경우
- NULL을 많이 포함하고 있는 컬럼

index를 자제해야 하는 경우
- 입력, 수정, 삭제 등의 작업이 빈번한 테이블

1) 테이블의 인덱스 확인하기
   SHOW INDEX FROM tablename;
2) 테이블의 인덱스 추가하기 : 컬럼은 1개도 가능, 2개 이상도 가능
   ALTER TABLE tablename ADD INDEX indexname (column1, column2);
3) 테이블의 유니크 인덱스 추가하기능
   ALTER TABLE tablename ADD UNIQUE INDEX indexname (column1, column2);
4) 테이블의 인덱스 삭제하기
   ALTER TABLE tablename DROP INDEX indexname;

```
SHOW INDEX FROM aa;   -- 테이블의 인덱스 확인하기

create index ind_irum on aa(irum);

ALTER TABLE aa ADD INDEX ind_juso (juso);

ALTER TABLE aa DROP INDEX ind_juso;

ALTER TABLE aa ADD UNIQUE INDEX ind_bun (bun);   -- unique 칼럼에 대한 인덱스
```

테이블 관련 명령

1. 테이블 형식 변경
Engine : ALTER TABLE [테이블명] ENGINE = [형식];
Type : ALTER TABLE [테이블명] TYPE = [형식];

2. 테이블 이름 변경
ALTER TABLE [테이블명] RENAME [바꿀이름];
RENAME TABLE [테이블명] TO [바꿀이름];

3. 컬럼 추가
마지막에 추가 : ALTER TABLE [테이블명] ADD COLUMN [컬럼이름] [컬럼타입];
지정 컬럼 뒤에 추가 : ALTER TABLE [테이블명] ADD COLUMN [컬럼이름] [컬럼타입] AFTER [컬럼이름];
제일 앞에 추가 : ALTER TABLE [테이블명] ADD COLUMN [컬럼이름] [컬럼타입] FIRST;

4. 컬럼 삭제
ALTER TABLE [테이블명] DROP COLUMN [컬럼이름];

5. 컬럼 변경
ALTER TABLE [테이블명] MODIFY [컬럼이름] [새컬럼타입];
ALTER TABLE [테이블명] CHANGE [컬럼이름] [새컬럼이름] [새컬럼타입];

6. 인덱스에 새항목 추가
ALTER TABLE [테이블명] ADD INDEX([컬럼이름])

7. 인덱스 삭제
ALTER TABLE [테이블명] DROP INDEX [컬럼이름]
DROP INDEX [인덱스이름] ON [테이블명]

8. 기본키(Primary Key or PK) 지정하기
ALTER TABLE [테이블명] ADD PRIMARY KEY([컬럼이름]);

9. 기본키(Primary Key or PK) 삭제하기
ALTER TABLE [테이블명] DROP PRIMARY KEY;

10. 자동 증가 값 초기화 하기
ALTER TABLE [테이블명] AUTO_INCREMENT=[a];

Query (질의) 연습 - SELECT

: db 서버로부터 자료를 클라이언트로 읽어 오는 명령

select 형식

select [distinct] db명.소유자명.테이블명.칼럼명 [as 별명]... [into 테이블명] from 테이블명...
where 조건... order by 기준키 asc[desc]...

- 정렬 : order by 칼럼명 [asc | desc]

- 연산자 사용 : 우선순위 () > 산술 > 관계 > 비교 > is null, like, in > between, not > and > or

- 비교 연산 : = , >, <, >=, <=, <> -- 논리 연산 : and, or, not, between

SQL SELECT 문의 기능

SELECT 문은 데이터베이스에서 정보를 검색합니다. SELECT 문을 사용하여 다음을 수행할 수 있습니다.

- 프로젝션: SQL의 프로젝션 기능을 사용하면 테이블에서 질의 결과로 반환될 열을 사용자가 필요한 만큼 선택할 수 있습니다.
- 선택: SQL의 선택 기능을 사용하면 테이블에서 질의 결과로 반환될 행(row)을 선택할 수 있으며 다양한 조건을 사용하여 표시할 행을 제한할 수 있습니다.
- 조인: SQL의 조인 기능을 사용하면 서로 다른 테이블 간에 링크를 생성하여 각 테이블에 저장된 데이터를 함께 가져올 수 있습니다. 조인에 대한 내용은 다른 단원에서 자세히 설명합니다.

우선순위 규칙

계산 순서	연산자
1	산술 연산자
2	연결 연산자 ||
3	비교 조건
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT 논리 조건
7	AND 논리 조건
8	OR 논리 조건

괄호를 사용하여 우선순위 규칙을 무시하고 우선순위를 변경할 수 있습니다.

내장 함수 : 데이터 아이템 조작의 효율성 증진

- 단일 행 함수 : 각 행에 대해 작업한다. 행 단위 처리.

* 문자 함수

① 대소문자 조작 함수 - LOWER, UPPER
② 문자조작함수 - CONCAT (문자열1, 문자열2) , SUBSTR(대상문자열, 인덱스)
LENGTH (문자의 갯수), INSTR(대상문자열, 검색문자:검색문자의 위치값 검색)
LPAD, RPAD, TRIM, REPLACE (대상문자열,old,new)

* 숫자 함수

1) ROUND (대상 숫자, 지정 자릿수 : 반올림)
2) TRUNCATE (대상 숫자, 지정 자릿수 : 숫자 절삭)
3) MOD (대상 숫자, 나눌 숫자 : 나머지 값)
4) 기타

* 날짜 함수

1) DB 서버의 현재 날짜와 시간을 반환
curdate() current_date() now() sysdate() current_timestamp()

2) 날짜 더하고 빼기
```
SELECT ADDDATE('2020-08-01', 3);    -- 3일 더하기
SELECT SUBDATE('2020-08-01', 3);    -- 3일 빼기
```

3) 단순히 일 차이를 가져올 때 DATEDIFF 함수, 차이를 연, 분기, 월, 주, 일, 시, 분, 초를 지정하여 가져올 때 TIMESTAMPDIFF 함수
```
SELECT DATEDIFF(NOW(), '2017-03-01');    # 1051 일
```

4) LAST_DAY, DAYOFYEAR ...
```
SELECT SYSDATE(), LAST_DAY(SYSDATE()), DAYOFYEAR(SYSDATE()), DAYOFWEEK(SYSDATE());
```

* 형 변환 함수

- 자동변환(implicit conversion) :

```
select jikwon_pay * '0.5' from jikwon
```

- 강제변환(explicit conversion)

DATE_FORMAT 날짜를 문자로

- DATE_FORMAT : **날짜형 자료 서식을 이용해 문자열로 출력**

참고 : 숫자 Format 서식

```
SELECT FORMAT(1234.567, 2);  -- Result: 1,234.57
SELECT FORMAT(1234.567, 0);  -- Result: 1,235
SELECT FORMAT(1234.567, -2); -- Result: 1,235
SELECT LPAD(56,5,0);   -- 00056
SELECT RPAD(56,5,0);   -- 56000
```

참고 : 날짜 및 시간 Format 서식

%M	월(Janeary, December, ...)
%W	요일(Sunday, Monday, ...)
%D	월(1st, 2dn, 3rd, ...)
%Y	연도(1987, 2000, 2013)
%y	연도(87, 00, 13)
%X	연도(1987, 2000) %V와 같이 쓰임.
%x	연도(1987, 2000) %v와 같이 쓰임.
%a	요일(Sun, Tue, ...)
%d	일(00, 01, 02, ...)
%e	일(0, 1, 2, ...)
%c	월(1, 2, ..., 12)
%b	월(Jan, Dec, ...)
%j	몇번째 일(120, 365)
%H	시(00, 01, 02, 13, 24)
%h	시(01, 02, 12)
%I(대문자 아이)	시(01, 02, 12)
%l(소문자 엘)	시(1, 2, 12)
%i	분(00, 01, 30)
%r	"hh:mm:ss AM|PM"
%T	"hh:mm:ss"
%S	초
%s	초
%p	AM, PM
%w	요일(0, 1, 2) 0:일요일
%U	주(시작:일요일)
%u	주(시작:월요일)
%V	주(시작:일요일)
%v	주(시작:월요일)

* 기타 함수

- RANK() : 순위를 결정

- NVL (value1,value2) : value1이 null이 아니면 value2를 쓴다. value1과 value2의 자료형은 일치

- NVL2 (value1,value2,value3) : value1이 null인지 평가 : null이면 value3을 null이 아니면 vlaue2
  자료형이 일치x

- NULLIF (value1, value2) : 두개의 값이 일치하면 NULL을 두개의 값이 일치하지 않으면 value1을 취함.

* Conditional Expressions (조건 표현식) : 비교 지원

① case 표현식 when 비교값1 then 결과값1
  when 비교값2 then 결과값2
  when 비교값3 then 결과값3
  [else 결과값n]
end as 컬럼별명

② if(조건) 참값, 거짓값 as 컬럼별명

복수 행 함수

: 전체 자료를 그룹별로 구분하여 통계 결과를 얻기 위한 함수
NULL값 무시 (count(*)는 Null값 허용

그룹 함수 - group by 절 : 소계 출력

형식: select 그룹칼럼명, 계산함수()... from 테이블명
  where 조건
  group by 그룹칼럼명 having 출력결과 조건

- 그룹칼럼에 order by 할 수 없다. 단, 출력결과는 order by 가능

JOIN : 하나 이상의 테이블에서 데이터 추출

```
-- Cross Join
  select jikwon_name, buser_name from jikwon, buser;                -- 120 행 출력
  select jikwon_name, buser_name from jikwon cross join buser;  -- 120 행 출력

-- EQUI Join
  select jikwon_name, buser_name from jikwon, buser where buser_num=buser_no;
  select jikwon_name, buser_name from jikwon inner join buser on  buser_num=buser_no;

-- NON-EQUI Join

-- Joins ： inner, outer joins
```

상품

상품코드	상품명
A	가
B	나
C	다
D	라

제조사

취급회사	취급상품코드
갑	A
을	B
병	
정	D
정	A

Inner join

상품명	취급회사
가	갑
나	을
라	정
가	정

Union : 구조가 일치하는 두 개 이상의 테이블 자료 합쳐 보기

Merge : 기존에 존재하는 행이 있다면 갱신되고, 존재하지 않는다면 insert 된다.

Left outer join

상품명	취급회사
가	갑
나	을
다	
라	정
가	정

Right outer join

상품명	취급회사
가	갑
나	을
	병
라	정
가	정

* Subquery : 질의문 안에 질의문이 있는 형태 (안쪽 질의 결과를 바깥쪽에서 참조)

```
--'박치기' 직원과 직급이 같은 직원 출력
    SELECT jikwon_jik FROM jikwon WHERE jikwon_name = '박치기';  --문1
    SELECT * FROM jikwon WHERE jikwon_jik = '사원';              --문2
--문1 +  문2
    SELECT * FROM jikwon WHERE jikwon_jik = (SELECT jikwon_jik FROM jikwon
     WHERE jikwon_name = '박치기');
```

** any, all 연산자 : null 인 자료는 제외하고 작업 함.

<any : subquery의 반환값 중 최대값 보다 작은 ~

>any : subquery의 반환값 중 최소값 보다 큰 ~

<all : subquery의 반환값 중 최소값 보다 작은 ~

>all : subquery의 반환값 중 최대값 보다 큰 ~

-상관 서브 쿼리

: 안쪽 질의에서 바깥쪽 질의를 참조하고, 다시 안쪽의 결과를 바깥쪽 질의에서 참조하는 형태.

- Sub Query를 이용한 Table 생성 및 insert 수행

update나 delete 에서도 서브쿼리를 사용할 수 있다.

Database Transactions

: 단위 별 데이터 처리를 의미한다. 한 사용자에 의해 한 개 이상의 SQL 구문을 포함하는 가장 작은 논리적인 작업을 의미한다.
클라이언트 상에 읽혀진 자료에 변화가 있으면 수정된 자료에 대해 commit이나 rollback으로 원본 DB에 저장 또는 취소할 수 있다.
이 때, 여러 개의 데이터에 대한 수정, 삭제, 추가 작업을 하나의 단위로 묶어 놓고, commit 이나 rollback을 수동으로 처리 한다.
이로 인해 클라이언트 상에서의 잘못된 작업에 대한 선별 취소가 가능하다. 데이터의 일관성 보장

- Transaction의 구성

① DML(Data Manipulation Language) : select(DQL), insert, update, delete
② DDL(Data Definition Language) : create, drop, alter, truncate, rename, comment,
③ DCL(Data Control Language) : grante, revoke
④ TCL(Transaction Control Language) : commit, rollback, savepoint

- Transaction의 종료

① rollback, commit, DDL, DCL 수행 후.
② SQLPLUS 종료 (비정상종료 -> rollback, 정상종료-> commit)
③ 서버종료 -> rollback
④ 타임아웃 -> rollback

* autocommit 설정을 변경하는 방법

Session 레벨에서 autocommit 설정을 변경할 때는, 아래의 구문을 사용한다.

• autocommit 설정값 확인
```
show variables like 'autocommit%';
```

• autocommit 설정 또는 해제
```
SET AUTOCOMMIT = TRUE;    -- 설정        SET AUTOCOMMIT = FALSE;    -- 해제
```

• 트랜잭션
COMMIT : 저장되지 않은 모든 자료를 데이터베이스에 저장하고, 현재의 트랜잭션을 종료한다. 원본 데이터가 갱신된다.
ROLLBACK : 저장되지 않은 모든 자료의 변경사항을 취소하고, 현재의 트랜잭션을 종료한다.
최종 commit이 발생했던 시점 이후의 DML작업 무시.

VIEW 파일

: 물리적인 테이블을 근거로 SELECT 문의 조건을 파일로 작성하여, 가상의 테이블로 사용한다.

물리적으로 데이터를 포함하지 않으므로 별도의 메모리 소모가 없다.

■ view를 사용하는 이유

1. 복잡하고 긴 쿼리문을 view로 단순화 할 수 있다.
2. 보안을 강화할 수 있다.
3. 동일한 데이터로부터 다양한 가상의 테이블을 얻을 수 있다.
4. 자료의 독립성 확보

형식) CREATE [or replace] view 뷰파일명 as SELECT 문
DROP view ~

SELECT jikwon_no,jikwon_name,jikwon_pay from jikwon where jikwon_ibsail < '2010-12-31';
➔ 위의 결과를 갖고 작업을 할 때가 많을 경우, 조건을 view 파일로 작성하여 수행하면 효과적.

```
CREATE or REPLACE view v_a as
SELECT jikwon_no,jikwon_name,jikwon_pay from jikwon where jikwon_ibsail < '2010-12-31';

SHOW TABLES;
SHOW FULL TABLES IN test WHERE TABLE_TYPE LIKE 'VIEW';      -- view 파일 목록 보기
```

계정(사용자) 생성 및 권한, 보안

DB 서버를 사용하는 클라이언트들에 대해 별도의 계정을 주고, 자신의 계정으로만 DB 자료를 공유 한다. 이 때 각 클라이언트들에 대해 select문을 포함해서 여러 sql 문의 사용권한을 제한함으로 해서, 클라이언트의 자격에 맞는 권한을 행사할 수 있도록 하여, DB의 자료가 함부로 남용 또는 손상되지 않도록 보호해야 한다.

권한

권한범위	권한	설명
전역	CREATE USER	사용자 생성
전역	FILE	LOAD DATA INFILE 같은 디스크의 파일 접근 시 필요한 권한
전역	PROCESS	MariaDB내의 프로세스 조회 권한
전역	RELOAD	FLUSH를 실행할 수 있는 권한
전역	SHUTDOWN	서버를 종료할 수 있는 권한
전역	SUPER	특정한 상황에서 제한을 넘어서 뭔가를 할 수 있는 권한 예) read_only 설정 시 데이터 변경 가능
데이터베이스 레벨	CREATE	새로운 데이터 베이스 생성
데이터베이스 레벨	DROP	데이터베이스 삭제
데이터베이스 레벨	GRANT OPTION	데이터베이스의 권한을 다른 사용자에게 부여할 수 있는 권한
데이터베이스 레벨	EVENT	이벤트 생성 및 삭제
데이터베이스 레벨	LOCK TABLE	명시적으로 테이블을 잠그는 권한
테이블 레벨	ALTER	테이블 구조 변경
테이블 레벨	CREATE	테이블 생성
테이블 레벨	DELETE	레코드 삭제
테이블 레벨	DROP	테이블 삭제
테이블 레벨	GRANT OPTION	테이블에 대한 권한을 다른 사용자에게 부여할 수 있는 권한
테이블 레벨	INSERT	레코드 입력
테이블 레벨	SELECT	레코드 조회
테이블 레벨	UPDATE	레코드 수정
STORED PROGRAM	ALTER ROUTIN	프로시저/함수 등 변경
STORED PROGRAM	EXECUTE	프로시저/함수 등 실행
STORED PROGRAM	GRANT OPTION	프로시저/함수 등의 권한을 다른 사용자에게 부여할 수 있는 권한

*** 계정 생성 및 권한 부여**

```
-- 비밀번호가 1111인 tester 계정을 생성.
MariaDB [(none)]> create user 'tester'@'localhost' identified by '1111';
 -- tester 계정에 demo 데이터베이스의 모든 사용권한을 부여한다.
 -- '%' 대신 localhost를 사용할 경우 외부에서 접속 불가
MariaDB [(none)]> grant all privileges on *.* to tester@localhost identified by '1111';
MariaDB [(none)]> show grants for tester@localhost;       -- 권한 조회
MariaDB [(none)]> flush privileges;       -- 권한설정을 새로 반영.
```

MySQL을 설치하면 기본적으로 로컬(localhost)에서만 접속이 가능하도록 되어 있다.

혼자 개발하는 거면 상관없지만,하나의 계정으로 여러 곳에서 사용할 경우 사용권한 부여시

'localhost' 대신 '%' 을 사용하여 외부에서 해당 계정에 대해서 외부에서 접근이 가능하도록 한다.

*** 계정 생성 확인**

```
C:₩Windows₩system32> mysql -u tester -p      ← 생성한 tester 계정으로 접속
Enter password: ****
Welcome to the MariaDB monitor.  Commands end with ; or ₩g.
Your MariaDB connection id is 9
MariaDB [(none)]>
```

사용자 권한 부여

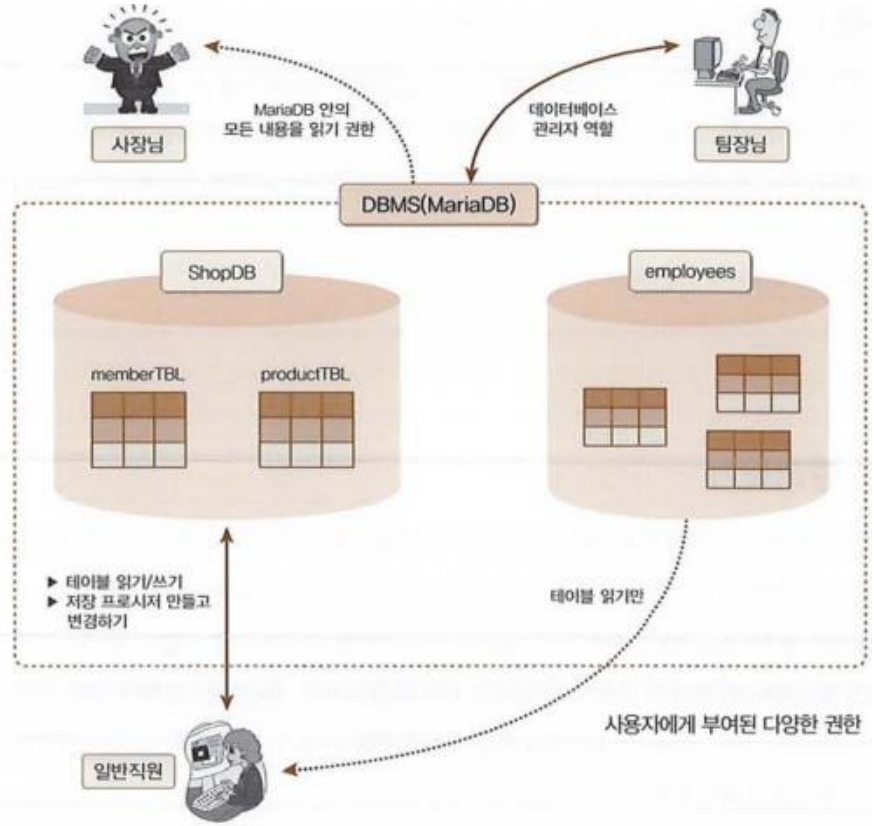
```
# 권한만 부여
MariaDB [(none)]> grant 권한목록 on db.table to 'user'@'host';

# 계정을 생성하고 권한까지 부여
MariaDB [(none)]> grant 권한목록 on db.table to 'user'@'host'
                  identified by 'password' with grant option;

# 글로벌(원격 접속 가능) 권한 부여
MariaDB [(none)]> grant super on *.* to 'jigi'@'%';

# 데이터베이스 권한 부여
MariaDB [(none)]> grant event on *.* to 'jigi'@'%';
MariaDB [(none)]> grant event on test.* to 'jigi'@'%';

# 테이블 권한 부여
MariaDB [(none)]> grant select,insert,update,delete on *.* to 'jigi'@'%';
MariaDB [(none)]> grant select,insert,update,delete on test.* to 'jigi'@'%';
MariaDB [(none)]> grant select,insert,update,delete on test.test_table to 'jigi'@'%';
MariaDB [(none)]> grant select,insert,update(uid) on test.test_table to 'jigi'@'%';    -- update는 uid 컬럼만 가능하다.
```



권한 그룹

```
# 권한 그룹(role) 생성
MariaDB [(none)]> create role dba;    -- 해당 롤을 만든 사용자만 롤 권한 부여 가능
MariaDB [(none)]> create role developer with admin jigi;  -- 해당 롤을 만든 사용자와 jigi 계정만 롤 권한 부여 가능

MariaDB [(none)]> grant all privileges on *.* to dba;     -- 모든 권한을 dba 롤에 부여
MariaDB [(none)]> grant select,insert,update,delete on test.* to developer;    -- test 데이터베이스의 select, DML 권한만 developer 롤에 부여
```

*** 간단한 백업과 복구

-- mysql DB 백업하기
> # mysqldump -uroot -ppass DBname > backup.sql -- 리눅스
C:₩WINDOWS₩System32> mysqldump -uroot -p123 mydb > c:₩work₩backup.sql -- 윈도우

1.root 에 db계정명을 적는다.
2.pass 부분에 mysql 비밀번호를 적는다. 비밀번호를 적지 않고 넘어가면 mysqldump 명령어 수행 시 비밀번호를 물어봄.
3.dbname 부분에 db명을 적는다.
4.backup 부분에 백업할 파일명을 적는다.

-- mysql 전체 DB 백업
> # mysqldump -uroot -p -A > all_backup.sql
C:₩WINDOWS₩System32>mysqldump -uroot -p123 -A > c:₩work₩all_backup.sql
혹은
mysqldump -uroot -p --all-databases > all_backup.sql
C:₩WINDOWS₩System32>mysqldump -uroot -p123 --all-databases > c:₩work₩all_backup.sql

-- mysql 특정 db의 특정 테이블만 백업
* dump 데이터베이스의 test 테이블만 백업한다.
# mysqldump -uroot -p dump mydb > my_backup.sql
C:₩WINDOWS₩System32>mysqldump -uroot -p123 mydb > c:₩work₩my_backup.sql

-- mysql schema 정보만 백업
> # mysqldump -uroot -p --no-data schemainfo > schemainfo.sql

-- 모든 db 복원
> # mysql -uroot -p < backup.sql
C:₩WINDOWS₩System32>mysql -uroot -p123 mydb < c:₩work₩backup.sql

저장 프로시저(Stored Procedure)

SQL 언어에 절차적인 프로그래밍 언어를 포함시켜 만든 것을 PL / SQL이라고 한다. 변수, 조건문, 반복문 등과 같은 프로그래밍 기법을 구사할 수 있어 SQL 처리를 좀 더 빠르고도 효과적으로 할 수 있다.

기본 구조는 DECLARE(선언부) ~ BEGIN(실행부) ~ EXCEPTION(예외처리부) ~ END로 이루어진다. 이러한 PL / SQL을 하나의 단위로 묶어 데이터베이스에 저장 한 후 실행할 수 있는데 이를 저장 프로시저라고 한다.

-- 저장 프로시저의 의미와 작성 방법 https://recoveryman.tistory.com/186?category=595002

BEGIN에서 END까지가 저장 프로시저의 본체
- 구분 문자(;) 변경하기
- 명령문이 완성되지 않은 상태에서 실행되면 곤란.
- 저장 프로시저에서 END를 입력하고 나서 CREATE PROCEDURE 명령이 실행되도록 환경을 변경.
- 그러려면 저장 프로시저를 작성하기 전에 구분 문자를(;)이 아닌 다른 문자로 변경해 둠. 일반적으로 //을 사용. $$도 가능.
- 구분 문자를 //으로 변경할 때에는 DELIMITER 명령을 사용.

EX)

```
DELIMITER //
create or REPLACE procedure sp_1 (a int, b INT)
begin
  SELECT a + b;
END;;
//
DELIMITER ;   -- 구분 문자를 원래대로 되돌려 놓는 명령

CALL sp_1(1, 3);
SHOW PROCEDURE STATUS;          -- PROCEDURE 목록 보기
SHOW CREATE PROCEDURE sp_1;   -- 내용 확인
DROP PROCEDURE sp_1;
```

사용자 정의 함수 만들기

```
CREATE or REPLACE FUNCTION 함수명(인수 INT ,,,) RETURNS 반환type
BEGIN
    …
   RETURN 반환값;
END
```

Ex1)
 -'BMI(body Mass Index) = 22'가 표준 체중이라고 했을 때, 다음과 같은 식이 성립.
 - 표준 체중 = 신장(cm 단위) X 신장(cm 단위) X 22 / 10000

```
DELIMITER //
CREATE or REPLACE FUNCTION fu1(height INT) RETURNS DOUBLE
BEGIN
   RETURN height * height * 22 / 10000;
END
//
DELIMITER ;


SELECT fu1(174);
```