



# TABA\_DB/SQL실습1

# Tibero Install

- 실습환경 AWS
- root 계정으로 설치

root 계정 접속이 처음이면

`sudo passwd` #root 계정 패스워드 설정

`su`

```
[ec2-user@ip-172-31-15-113 ~]$ sudo passwd
Changing password for user root.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-15-113 ~]$ su
Password:
[root@ip-172-31-15-113 ec2-user]#
```

# JRE 설치

- `yum update && yum install -y jre`

```
[root@localhost Downloads]# yum install jre
Last metadata expiration check: 0:45:30 ago on Sun 09 Jul 2023 09:44:39 PM KST.
Dependencies resolved.
=====
Package                Arch  Version                Repo                Size
=====
Installing:
  java-11-openjdk        x86_64 1:11.0.18.0.10-3.el9   appstream           440 k
Installing dependencies:
  copy-jdk-configs       noarch 4.0-3.el9              appstream            28 k
  java-11-openjdk-headless x86_64 1:11.0.18.0.10-3.el9   appstream            40 M
  javapackages-filesystem noarch 6.0.0-4.el9            appstream            13 k
  lksctp-tools           x86_64 1.0.19-2.el9           baseos                94 k
  lua                    x86_64 5.4.4-4.el9            appstream            188 k
  lua-posix              x86_64 35.0-8.el9             appstream            151 k
  mkfontscale            x86_64 1.2.1-3.el9            appstream            32 k
  ttmkfdir               x86_64 3.0.9-65.el9           appstream            53 k
  tzdata-java            noarch 2023c-1.el9            appstream            230 k
  xorg-x11-fonts-Type1   noarch 7.5-33.el9             appstream            505 k

Transaction Summary
=====
```

# JRE 설치

- 설치 확인
- `java -version`

```
[root@ip-172-31-15-113 ec2-user]# java -version
openjdk version "22" 2024-03-19
OpenJDK Runtime Environment Corretto-22.0.0.37.1 (build 22+37-FR)
OpenJDK 64-Bit Server VM Corretto-22.0.0.37.1 (build 22+37-FR, mixed mode, sharing)
[root@ip-172-31-15-113 ec2-user]#
```

# Tibero6 다운로드

- 테크넷 접속 후 회원가입 <https://technet.tmaxsoft.com/>
- 다운로드 - 데이터베이스 - Tibero - Tibero6 다운로드

**TechNet**

제품정보 | 기술지식 | **다운로드** | 고객지원

미들웨어  
인터페이스 프레임워크  
비즈니스 프레임워크  
성능관리/리호스팅  
**데이터베이스**  
Tibero  
ProSync  
ZetaData  
SysMasterDB  
운영체제  
클라우드  
데이터 사이언스 플랫폼  
데모라이선스 신청

**Tibero**

☒ All ☐ Hot Fix ☐ Security Fix ☐ Fix ☐ Service Pack ☐ 일반패치 ☐ 권고패치

**Tibero 7**

MAIN Tibero 7 다운로드 | 설치안내 | 매뉴얼 다운로드 | 온라인매뉴얼 | 변경사항

**Tibero 6**

MAIN Tibero 6 다운로드 | 설치안내 | 매뉴얼 다운로드 | 온라인매뉴얼 | 변경사항

# Tibero6 다운로드

- OS버전에 맞는 Tibero6 다운로드
- Linux(x86) 64-bit

The screenshot shows the Tibero6 download interface. At the top, there are tabs for different operating systems: MS, Linux, IBM, HP, and Solaris. The 'Linux' tab is selected. Below the tabs, a list of Linux distributions is shown: Linux (Itanium) 64-bit, Linux (x86) 32-bit, Linux (x86) 64-bit, Linux (POWER) 64-bit, and Linux (x86) 64-bit (3.10). The 'Linux (x86) 64-bit' option is selected, and the breadcrumb path 'Linux > Linux (x86) 64-bit' is displayed. Below this, there is a section titled '첨부파일' (Attachments) with a red box highlighting the download links. The links are: 'linux64(tibero6-bin-FS07-CS-2005-linux64-186930-opt-tested.tar.gz) 469.21 MB' and 'linux64(tibero6-bin-FS07-CS-2005-linux64-186930-opt-tested.md5) 85.00 bytes'.

MS Linux IBM HP Solaris

- Linux (Itanium) 64-bit
- Linux (x86) 32-bit
- Linux (x86) 64-bit
- Linux (POWER) 64-bit
- Linux (x86) 64-bit (3.10)

Linux > Linux (x86) 64-bit

첨부파일

linux64(tibero6-bin-FS07-CS-2005-linux64-186930-opt-tested.tar.gz) 469.21 MB

linux64(tibero6-bin-FS07-CS-2005-linux64-186930-opt-tested.md5) 85.00 bytes

# Tibero6 데모라이센스 신청

- 메인 페이지에서 데모라이선스 신청 클릭
- 리눅스 서버에서 **hostname** 명령어 실행으로 본인의 **hostname**을 적어줘야함

```
[root@ip-172-31-15-113 ec2-user]# hostname
ip-172-31-15-113.ap-northeast-2.compute.internal
[root@ip-172-31-15-113 ec2-user]#
```

데모라이선스 발급 신청 - Chrome

technet.tmaxsoft.com/ko/front/common/demoPopup.do

데모라이선스 발급 신청

제품명	Tibero	Tibero 6
Version	6.0	
발급유형	Demo	
Host Name		hostname 확인
Edition	Standard	
E-mail		
성명		
연락처		
회사명		
사용목적	*사용 목적을 반드시 기재 바랍니다.	

데모 라이선스는 제품 구입 전 테스트 및 검토를 위하여 제한된 기간 동안 발급받아 사용 가능한 라이선스입니다. 아래 명시된 유효기간 내에서만 사용가능하며, 테스트 및 검토 용도가 아닌 실제 운영시스템에 사용하는 것은

로그아웃 | 마이페이지 | Korean

검색

로그인하셨습니다. | LOGOUT

마이페이지

데모라이선스 신청

데모라이선스는 제품구입 전 테스트 및 검토를 위하여 제한된 기간 동안 발급받아 사용 가능한 라이선스입니다.

공지사항 more

[WebtoB : 중요공지] 보안 취약점

# WinSCP 다운

- windows에서 linux서버로 파일 전송 가능하게 해주는 SW
- winscp 설치(<https://winscp.net/>)





# WinSCP 접속

- AWS - 인스턴스 - 연결 - 하단에 있는 계정@퍼블릭IPv4 DNS 복사

인스턴스에 연결 정보

다음 옵션 중 하나를 사용하여 인스턴스 i-0ee0d00abeafaa9e (aws\_tibero)에 연결

EC2 인스턴스 연결   Session Manager   **SSH 클라이언트**   EC2 직렬 콘솔

인스턴스 ID  
i-0ee0d00abeafaa9e (aws\_tibero)

1. SSH 클라이언트를 엽니다.
2. 프라이빗 키 파일을 찾습니다. 이 인스턴스를 시작하는 데 사용되는 키는 Tibero.pem입니다.
3. 필요한 경우 이 명령을 실행하여 키를 공개적으로 볼 수 없도록 합니다.  
`chmod 400 "Tibero.pem"`
4. 퍼블릭 DNS을(를) 사용하여 인스턴스에 연결:  
`ec2-3-35-47-228.ap-northeast-2.compute.amazonaws.com`

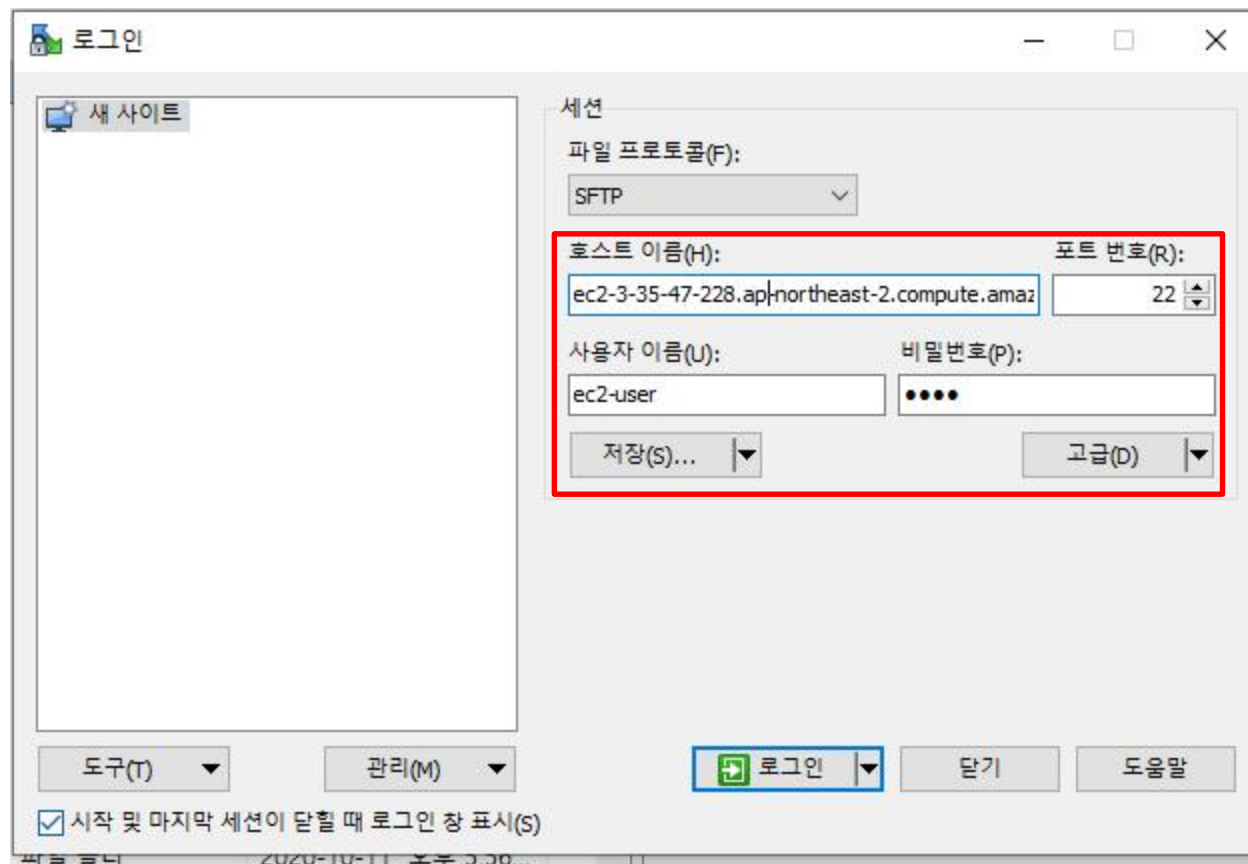
예:  
`ssh -i "Tibero.pem" ec2-user@ec2-3-35-47-228.ap-northeast-2.compute.amazonaws.com`

참고: 대부분의 경우 추정된 사용자 이름은 정확합니다. 하지만 AMI 사용 지침을 읽고 AMI 소유자가 기본 AMI 사용자 이름을 변경했는지 확인하세요.

취소

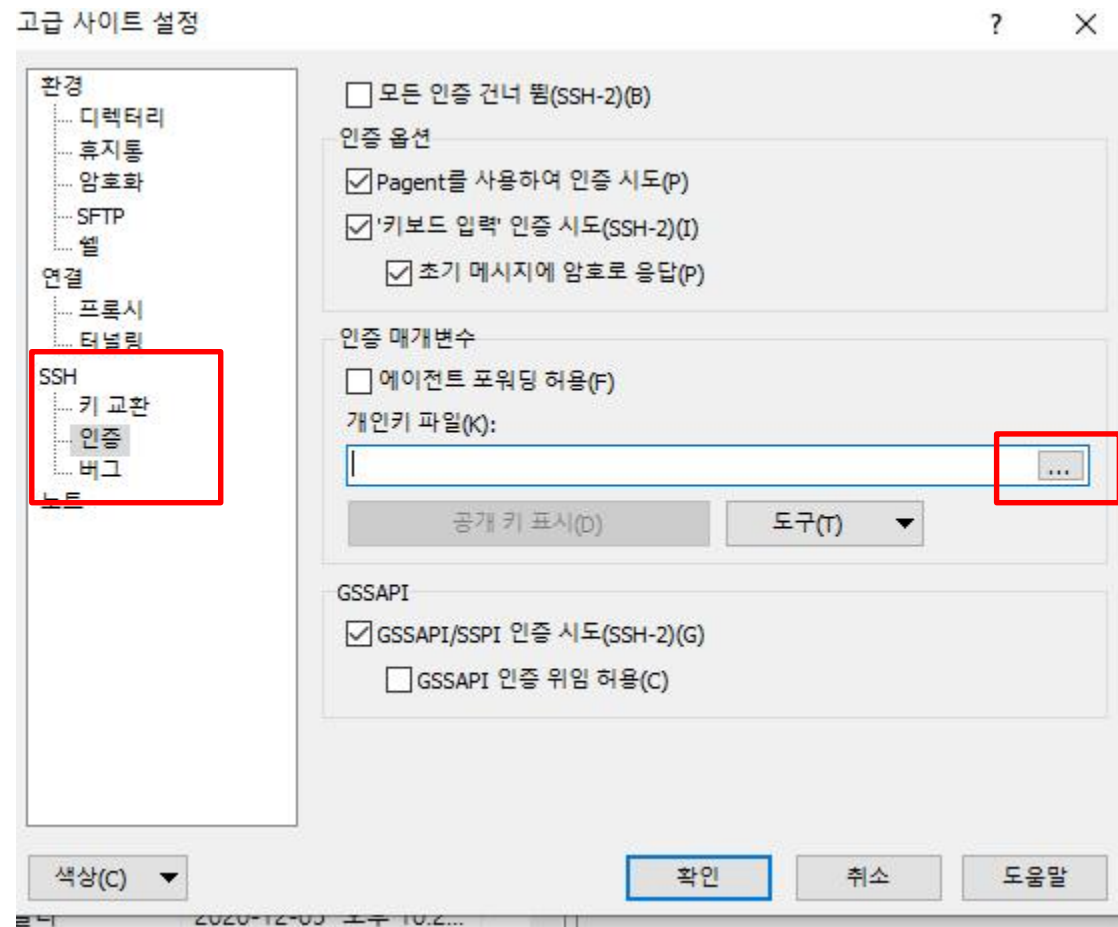
# winscp를 이용해 파일 전송

- 호스트 이름, 사용자 이름(ec2-user), 패스워드 입력 후
- 고급 클릭



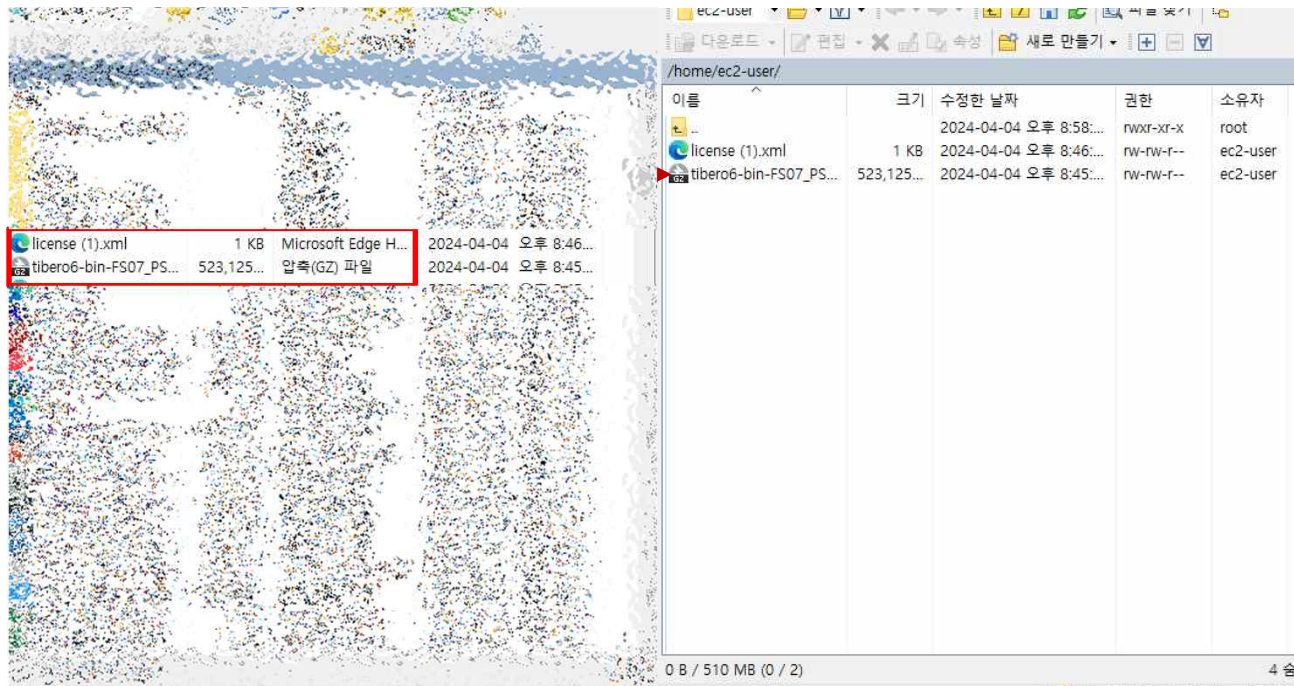
# Tibero Install

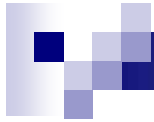
- 좌측 SSH-인증 탭 선택 후 개인키 파일 등록



# Tibero Install

- license.xml 과 tibero설치 파일을 드래그&드랍으로 리눅스에 옮기기
- /home/ec2-user 경로에 저장





# Tibero Install

- `mkdir /home/tibero` #tibero 폴더 생성
- `mv tibero6-bin-FS07_PS01-linux64_3.10-269987-20240327170532.tar.gz` ₩  
/home/tibero #파일 이동
- `cd /home/tibero` # 생성한 tibero 폴더로 이동
- `tar -xzvf tibero6-bin-FS07_PS01-linux64_3.10-269987-20240327170532.tar.gz`
- `mv /home/ec2-user/license.xml /home/tibero/tibero6/license` #라이선스를 tibero -  
license 폴더로 이동

# Tibero 설치를 위한 설정 변경

- vi /etc/sysctl.conf

kernel.shmmni = 4096

kernel.shmall = 2097152

kernel.shmmax = 2147483648

kernel.sem = 10000 32000 10000 10000

fs.file-max = 6815744

net.ipv4.ip\_local\_port\_range = 1024 65500

```
# For more information, see sysctl.conf(5) and sysctl(8)
#
kernel.shmmni = 4096
kernel.shmall = 2097152
kernel.shmmax = 2147483648
kernel.sem = 10000 32000 10000 10000
fs.file-max = 6815744
net.ipv4.ip_local_port_range = 1024 65500
```

# Tibero 설치를 위한 설정 변경

- vi /etc/security/limits.conf

tibero	soft	nproc	2047
tibero	hard	nproc	16384
tibero	soft	nofile	1024
tibero	hard	nofile	65536

```
tibero          soft    nproc    2047
tibero          hard    nproc    16384
tibero          soft    nofile   1024
tibero          hard    nofile   65536
```

## Tibero 환경 변수 설정

- 자바 환경변수 설정

readlink -f \$(which /usr/bin/java)

```
[root@ip-172-31-15-113 tibero6]# readlink -f $(which /usr/bin/java)  
/usr/lib/jvm/java-22-amazon-corretto.x86_64/bin/java
```

/usr/lib/jvm/java-22-amazon-corretto.x86\_64/  
bin 앞까지가 JAVA\_HOME 경로



# Tibero 환경 변수 설정

- 환경 변수 설정
- vi ~/.bashrc

```
##JAVA_HOME##  
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.362.b09-4.el9.x86_64
```

```
##Tibero ##  
export TB_HOME=/home/tibero/tibero6  
export TB_SID=tibero  
export TB_PROF_DIR=$TB_HOME/bin/prof  
export LD_LIBRARY_PATH=$TB_HOME/lib:$TB_HOME/client/lib  
export SHLIB_PATH=$LD_LIBRARY_PATH:$SHLIB_PATH  
export LIBPATH=$LD_LIBRARY_PATH:$LIBPATH
```

```
##PATH##  
export PATH=$PATH:$JAVA_HOME/bin:$TB_HOME/bin:$TB_HOME/client/bin
```

```
:wq #저장  
source ~/.bashrc #환경변수 설정 적용
```

# Tibero Install

- Tibero에 필요한 파일 설치
- `yum install -y libnsl && yum install -y ncurses* libaio`
- `ln -s /usr/lib64/libncurses.so.6.2 /usr/lib64/libncurses.so.5`
- `ln -s /usr/lib64/libtinfo.so.6 /usr/lib64/libtinfo.so.5`
- `ln -s /usr/lib64/libform.so.6.2 /usr/lib64/libform.so.5`
- `$TB_HOME/config/gen_tip.sh`
- `tbboot nomount`
- `tbsql sys/tibero`

```
[root@ip-172-31-15-113 config]# ./gen_tip.sh
Using TB_SID "tibero"
/home/tibero/tibero6/config/tibero.tip generated
/home/tibero/tibero6/config/psm_commands generated
/home/tibero/tibero6/client/config/tbdsn.tbr generated.
Running client/config/gen_esql_cfg.sh
Done.
[root@ip-172-31-15-113 config]# tbboot nomount
Change core dump dir to /home/tibero/tibero6/bin/prof.
*****
* Failed to access pstack path.
* Path: /usr/bin/pstack
* Pstack is required to dump the callstack.
*****
Listener port = 8629

Tibero 6

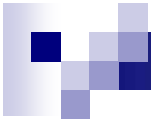
TmaxTibero Corporation Copyright (c) 2008-. All rights reserved.
Tibero instance started up (NOMOUNT mode).
[root@ip-172-31-15-113 config]# tbsql sys/tibero

tbSQL 6

TmaxTibero Corporation Copyright (c) 2008-. All rights reserved.

Connected to Tibero.

SQL> █
```



# Tibero Install

```
create database "tibero"
user sys identified by tibero
maxinstances 8
maxdatafiles 100
character set MSWIN949
national character set UTF16
logfile
  group 1 'log001.log' size 100M,
  group 2 'log002.log' size 100M,
  group 3 'log003.log' size 100M
maxloggroups 255
maxlogmembers 8
noarchivelog
datafile 'system001.dtf' size 100M autoextend on next 100M maxsize unlimited
default temporary tablespace TEMP
  tempfile 'temp001.dtf' size 100M autoextend on next 100M maxsize unlimited
  extent management local autoallocate
undo tablespace UNDO
  datafile 'undo001.dtf' size 100M autoextend on next 100M maxsize unlimited
  extent management local autoallocate;

quit
```



Database created 문구가 나오면 됨

# Tibero Install

- tbboot
- \$TB\_HOME/scripts/system.sh
- sys와 syscat 암호 입력
- sys - tibero
- syscat - syscat
- 나머지 y누르면 됨.

```
[root@ip-172-31-15-113 scripts]# ./system.sh
Enter SYS password:
Enter SYSCAT password:
Creating additional system index...
Dropping agent table...
Creating client policy table ...
Creating text packages table ...
```



# DDL : Data Definition Language

## 데이터 정의어(Data Definition Language)

데이터 정의어(이하 DDL)는 데이터 간에 관계를 정의하여 데이터베이스 구조를 설정하는 SQL 문장

구분	명령어	설명
데이터베이스	CREATE DATABASE	데이터베이스를 생성한다.
	ALTER DATABASE	데이터베이스를 변경한다.
테이블	CREATE TABLE	테이블을 생성한다.
	ALTER TABLE	테이블을 변경한다.
	DROP TABLE	테이블을 제거한다.
테이블 스페이스	CREATE TABLESPACE	테이블 스페이스를 생성한다.
	ALTER TABLESPACE	테이블 스페이스를 변경한다.
	DROP TABLESPACE	테이블 스페이스를 제거한다.

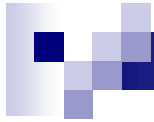
## 데이터 정의어(Data Definition Language)

구분	명령어	설명
인덱스	CREATE INDEX	인덱스를 생성한다.
	ALTER INDEX	인덱스를 변경한다.
	DROP INDEX	인덱스를 제거한다.
뷰	CREATE VIEW	뷰를 생성한다.
	ALTER VIEW	뷰를 변경한다.
	DROP VIEW	뷰를 제거한다.
동의어	CREATE SYNONYM	동의어를 생성한다.
	DROP SYNONYM	동의어를 제거한다.
사용자	CREATE USER	사용자를 생성한다.
	ALTER USER	사용자를 변경한다.
	DROP USER	사용자를 제거한다.

## 데이터 정의어(Data Definition Language)

구분	명령어	설명
함수	CREATE FUNCTION	함수를 생성한다.
	ALTER FUNCTION	함수를 변경한다.
	DROP FUNCTION	함수를 제거한다.
프러시저	CREATE PROCEDURE	프러시저를 생성한다.
	ALTER PROCEDURE	프러시저를 변경한다.
	DROP PROCEDURE	프러시저를 제거한다.
타입	CREATE TYPE	타입을 생성한다.
	ALTER TYPE	타입을 변경한다.
	DROP TYPE	타입을 제거한다.





## 데이터 정의어(Data Definition Language)

구분	명령어	설명
권한	GRANT	사용자에게 특권을 부여한다.
	REVOKE	사용자에게 특권을 회수한다.
역할	CREATE ROLE	역할을 생성한다.
	ALTER ROLE	역할을 변경한다.
	DROP ROLE	역할을 제거한다.
객체	RENAME	테이블, 뷰, 동의어, 시퀀스 등의 스키마 객체의 이름을 변경한다.

## Create Tables to Store Data

테이블 인스턴스 –테이블의 구조와 칼럼의 특성을 요약

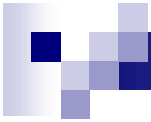
Symbols	Explanation
<b>PK</b>	기본 키 열
<b>FK</b>	외래 키 열
<b>FK1, FK2</b>	동일한 테이블에 있는 두 개의 외부 키
<b>NN</b>	NOT NULL 열
<b>U</b>	고유 열

## Create Tables to Store Data

### Table Instance Chart

#### Table Name: E\_EMP

Column Name	ID	LAST_NAME	FIRST_NAME	START_DATE	SALARY	MANAGER_ID	DEPT_ID
Key Type	PK					FK1	FK2
Nulls / Unique	NN, U					NN	
FK Ref Table						S_EMP	S_DEPT
FK Ref Column						ID	ID
Data Type	NUMBER	CHAR	CHAR	DATE	NUMBER	NUMBER	NUMBER
Maximum Length	7	25	25		11	7	7
Sample Data	1	Alexander	Lee	2022-09-01	1500		10
	2	Jonathan	Hong	2022-10-05	1000	1	20



## Create Tables to Store Data

### Syntax

```
CREATE TABLE [user.] table_name
  ({column_name datatype | table_constraint
  ,column_name datatype | table_constraint});
```

where	<i>table_name</i>	테이블 이름
	<i>column_name</i>	열 이름
	<i>datatype</i>	데이터 타입
	<i>table_constraint</i>	제약 조건

## Create Tables to Store Data

스키마 객체 이름

- 이름 부여 시 따옴표(“ ”) 없는 식별자는 모두 대문자로 간주
- 따옴표 있는 식별자는 대소문자를 구분

모두 다른 식별자	모두 같은 식별자
<b>department</b> <b>“department”</b> <b>“Department”</b>	<b>department</b> <b>DEPARTMENT</b> <b>“DEPARTMENT”</b>

식별자를 기술 할 때 규칙

- 길이가 30bytes를 넘으면 안된다.
- 따옴표 없는 식별자는 알파벳, 한글, 숫자, 언더바(\_), \$, #만 사용. 단, 숫자, '\$', '#'는 첫 글자로 올 수 없다.
- 따옴표 있는 식별자는 공백을 포함한 어떤 문자 사용 가능. 다만, 큰따옴표(“ ”)는 사용 불가능.
- 하나의 네임스페이스 안에 서로 다른 두 객체가 동일한 이름을 사용할 수 없다.



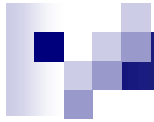
## Create Tables to Store Data

### Tibero에서 제공하는 데이터 타입

구분	데이터 타입
문자형	CHAR, VARCHAR, VARCHAR2, NCHAR, NVARCHAR, NVARCHAR2, RAW, LONG, LONG RAW
숫자형	NUMBER, INTEGER, FLOAT
날짜형	DATE, TIME, TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE
간격형	INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND
대용량 객체형	CLOB, BLOB, XMLTYPE
내재형	ROWID

## Create Tables to Store Data

Datatype	Description
CHAR(s)	고정된 문자열 길이, 최대 2,000자까지 선언, 문자열의 길이가 0인 값은 NULL로 인식
	CHAR(size[BYTE   CHAR]) -> EXAM CHAR(10)
VARCHAR2(s)	가변 문자열 길이, 최대 4,000자까지 선언, 문자열의 길이가 0인 값은 NULL로 인식
	VARCHAR2(size[BYTE   CHAR]) -> EXAM VARCHAR2(10)
VARCHAR(s)	VARCHAR2 타입과 동일
LONG	VARCHAR2와 비슷하지만, 최대 2GB까지 선언
DATE	연도는 BC 9,999 ~ AD 9,999까지 표현,
NUMBER(p,s)	정수 또는 실수를 저장, 음양으로 절댓값이 $1.0 \times 10^{-130}$ 보다 크거나 같고, $1.0 \times 10^{126}$ 보다 작은 38자리의 수를 표현할 수 있으며 0과 ±무한대를 포함한다.

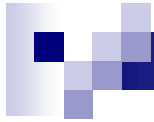


## Create Tables to Store Data

### 데이터 무결성 제약 조건

Constraint	Description
NOT NULL	NULL값을 허용하지 않고, 반드시 데이터를 입력. 제약조건이 NULL이면 해당 컬럼은 NULL값을 허용.
UNIQUE	해당 칼럼이 중복되는 데이터가 존재할 수 없는 유일성을 보장하는 제약조건
PRIMARY KEY	NOT NULL, UNIQUE 제약 조건의 결합과 같다. 테이블 또는 뷰는 단 한 개의 PRIMARY KEY 제약조건을 가질 수 있다.
FOREIGN KEY	같은 테이블 또는 서로 다른 두 개 테이블의 키 컬럼 사이의 관계
CHECK	expr로 표현한 조건이 항상 참이 되도록 유지. 특정 조건을 평가 후 만족하지 못하면 에러 발생





# Create Tables to Store Data

## EXAMPLE

Table Name: S\_REGION

Column Name	ID	NAME
Key Type	PK	
Nulls/Unique	NN, U	NN, U
Sample data	1	North America
	2	South America
	3	Africa/Middle East
	4	Asia
	5	Europe

```
SQL> CREATE TABLE s_region
2   ( id                NUMBER(7),
3     name              VARCHAR2(50)
4       CONSTRAINT s_region_name_nn NOT NULL,
5       CONSTRAINT s_region_id_pk PRIMARY KEY (id),
6       CONSTRAINT s_region_name_uk UNIQUE (name));
Table 'S_REGION' created.
```

# Create Tables to Store Data

## EXAMPLE

Table Name: S\_DEPT

Column Name	deptno	dname	loc
Key Type	PK		
Nulls/Unique			
Datatype	NUMBER	VARCHAR2	VARCHAR2
Maximum Length	2	14	13
Sample data	10	ACCOUNTING	NEW YORK
	20	RESEARCH	DALLAS
	30	SALES	CHICAGO
	40	OPERATIONS	BOSTON

```
SQL> CREATE TABLE s_dept
2  ( deptno          NUMBER(2),
3    dname           VARCHAR2(14),
4    loc             VARCHAR2(13),
5    CONSTRAINT s_dept_pk PRIMARY KEY(deptno));
Table 'S_DEPT' created.
```

## Create Tables to Store Data

### EXAMPLE

Table Name: S\_DEPT

```
INSERT INTO S_DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK');  
INSERT INTO S_DEPT VALUES (20, 'RESEARCH', 'DALLAS');  
INSERT INTO S_DEPT VALUES (30, 'SALES', 'CHICAGO');  
INSERT INTO S_DEPT VALUES (40, 'OPERATIONS', 'BOSTON');
```

# Create Tables to Store Data

## EXAMPLE

Table Name: S\_EMP

Column Name	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
Key Type	PK			FK1				FK2
Nulls/Unique	NN, U	NN						
FK Ref Table				S_EMP				S_DEPT
FK Ref Column				EMPNO				DEPTNO
Datatype	NUMBER	VARCHAR2	VARCHAR2	NUMBER	DATE	NUMBER	NUMBER	NUMBER
MaxLength	4	10	9	4		7	7	2
Sample data	7839	KING	PRESIDENT	null	1981-11-17	5000	null	10
	7566	JONES	MANAGER	7839	1981-02-04	2975	null	20
	7902	FORD	ANALYST	7566	1981-03-12	3000	null	20

## Create Tables to Store Data

```
SQL> create table s_emp  
( empno NUMBER (7) CONSTRAINT s_emp_empno_nn NOT NULL,  
  ename VARCHAR2 (10) CONSTRAINT s_emp_ename_nn NOT NULL,  
  job VARCHAR2 (9),  
  mgr NUMBER (4),  
  hiredate DATE,  
  sal NUMBER (7),  
  comm NUMBER (7),  
  deptno NUMBER (2),  
  constraint s_emp_id_pk PRIMARY KEY (empno),  
  constraint s_emp_mgr_fk FOREIGN KEY(mgr) REFERENCES s_emp(empno),  
  constraint s_emp_deptno_fk FOREIGN KEY(deptno) REFERENCES s_dept(deptno));
```

## Create Tables to Store Data

### EXAMPLE

Table Name: S\_EMP

```
INSERT INTO S_EMP VALUES (7839,'KING','PRESIDENT',NULL,'81-11-17',5000,NULL,10);
INSERT INTO S_EMP VALUES (7698,'BLAKE','MANAGER',7839,'81-05-01',2850,NULL,30);
INSERT INTO S_EMP VALUES (7782,'CLARK','MANAGER',7839,'81-05-09',2450,NULL,10);
INSERT INTO S_EMP VALUES (7566,'JONES','MANAGER',7839,'81-04-01',2975,NULL,20);
INSERT INTO S_EMP VALUES (7654,'MARTIN','SALESMAN',7698,'81-09-10',1250,1400,30);
INSERT INTO S_EMP VALUES (7499,'ALLEN','SALESMAN',7698,'81-02-11',1600,300,30);
INSERT INTO S_EMP VALUES (7844,'TURNER','SALESMAN',7698,'81-08-21',1500,0,30);
INSERT INTO S_EMP VALUES (7900,'JAMES','CLERK',7698,'81-12-11',950,NULL,30);
INSERT INTO S_EMP VALUES (7521,'WARD','SALESMAN',7698,'81-02-23',1250,500,30);
INSERT INTO S_EMP VALUES (7902,'FORD','ANALYST',7566,'81-12-11',3000,NULL,20);
INSERT INTO S_EMP VALUES (7369,'SMITH','CLERK',7902,'80-12-09',800,NULL,20);
INSERT INTO S_EMP VALUES (7788,'SCOTT','ANALYST',7566,'82-12-22',3000,NULL,20);
INSERT INTO S_EMP VALUES (7876,'ADAMS','CLERK',7788,'83-01-15',1100,NULL,20);
INSERT INTO S_EMP VALUES (7934,'MILLER','CLERK',7782,'82-01-11',1300,NULL,10);
```

# CONFIRM THE STRUCTURE OF A TABLE

테이블 구조 확인

## Syntax

```
DESCRIBE table_name
```

## Example

```
SQL> DESCRIBE s_emp;
```

COLUMN_NAME	TYPE	CONSTRAINT
EMPNO	NUMBER(7)	PRIMARY KEY NOT NULL
ENAME	VARCHAR(10)	NOT NULL
JOB	VARCHAR(9)	
MGR	NUMBER(4)	REFERENTIAL
HIREDATE	DATE	
SAL	NUMBER(7)	
COMM	NUMBER(7)	
DEPTNO	NUMBER(2)	REFERENTIAL
INDEX_NAME	TYPE	COLUMN_NAME
S_EMP_ID_PK	NORMAL	EMPNO

## ADD A COLUMN TO A TABLE

테이블에 칼럼 추가하기

### Syntax

```
ALTER TABLE table_name  
ADD ( column_name datatype  
[, column_name datatype]...)
```

### Example

```
SQL> ALTER TABLE s_region  
2 ADD (comments VARCHAR2 (255));  
Table 'S_REGION' altered.
```



## MODIFY A COLUMN

테이블에 존재하는 칼럼 속성 변경  
데이터 타입, 기본값, 제약조건 변경

### Syntax

```
ALTER TABLE table_name  
MODIFY ( column_name datatype  
[, column_name datatype]...)
```

### Example

S\_EMP 테이블의 ENAME 칼럼의 길이 20으로 변경 및 SAL 값은 NULL이 될 수 없음

```
SQL> ALTER TABLE s_emp  
2  MODIFY (ename VARCHAR2 (20));  
Table 'S_EMP' altered.
```

```
SQL> ALTER TABLE s_emp  
2  MODIFY (SAL NOT NULL);  
Table 'S_EMP' altered.
```

## ADD AND REMOVE DATA CONSTRAINTS

### Example

S\_EMP 테이블의 외래키 삭제 및 추가

```
SQL> ALTER TABLE s_emp  
2 DROP CONSTRAINT s_emp_mgr_fk;  
Table 'S_EMP' altered.
```

```
SQL> ALTER TABLE s_emp  
2 ADD CONSTRAINT s_emp_mgr_fk  
3 FOREIGN KEY (mgr) REFERENCES s_emp(empno);  
Table 'S_EMP' altered.
```

## DROP A TABLE

DROP TABLE 명령어를 사용해 데이터베이스에서 테이블 제거.  
DROP TABLE 명령어는 한번 실행되면 취소할 수 없다.

### Syntax

```
DROP TABLE table_name
```

### Example

S\_REGION 테이블 삭제

```
SQL> DROP TABLE s_region;
```

Table 'S\_REGION' dropped.

## Simplify Data Access with Views

- 테이블 뷰를 만들어 논리적 하위 집합 또는 데이터 조합을 표시한다.
- 뷰는 실제 데이터가 포함되지 않는다.
- 뷰 이름은 테이블과 같은 네임스페이스를 사용하므로 스키마 내 다른 이름과 중복되면 안된다.
  
- 장점
  - 접근 제어로 보안 제공
  - 데이터 관리가 편리
  
- 단점
  - 삽입, 삭제, 갱신, 연산에 제약이 있다.

# CREATE VIEWS

## Syntax

```
CREATE VIEW view_name [ (alias, [alias]...)  
AS query  
WHERE  
WITH CHECK OPTION
```

where	<i>view_name</i>	뷰 이름
	<i>alias</i>	별칭
	<i>query</i>	SELECT 문
	<i>WITH CHECK OPTION</i>	해당 옵션을 사용하면 INSERT/UPDATE 가능
	<i>WITH READ ONLY</i>	읽기 전용 뷰
	<i>constraint</i>	CHECK OPTION에 할당 된 제약조건

# CREATE VIEWS

## Example

부서 번호가 10인 직원 번호, 이름, 직무가 포함된 뷰를 생성

```
SQL> CREATE VIEW empvu10
  2 AS SELECT empno, ename, job
  3 FROM s_emp
  4 WHERE deptno = 10;
```

View 'EMPVU10' created.

```
SQL> SELECT *
  2 FROM empvu10;
```

EMPNO	ENAME	JOB
7782	CLARK	MANAGER
7839	KING	PRESIDENT
7934	MILLER	CLERK

3 rows selected.

# CREATE VIEWS

## Example

부서 번호가 20인 뷰를 생성. 직원 번호는 ID , 이름은 EMPLOYEE, 직무는 TITLE로 표현.

```
SQL> CREATE VIEW empvu20 (id, employee, title)
  2 AS SELECT empno, ename, job
  3 FROM s_emp
  4 WHERE deptno = 20;
```

View 'EMPVU20' created.

```
SQL> SELECT *
  2 FROM empvu20;
```

ID	EMPLOYEE	TITLE
7369	SMITH	CLERK
7566	JONES	MANAGER
7788	SCOTT	ANALYST
7876	ADAMS	CLERK
7902	FORD	ANALYST

5 rows selected.

# CREATE VIEWS

## Example

월급이 1500 이상인 뷰를 생성. 직원 번호는 ID , 이름은 NAME, 월급은 MONTHLY\_SALARY로 표현.

```
SQL> CREATE VIEW salvu1500
  AS SELECT empno ID, ename NAME, sal MONTHLY_SALARY
  FROM s_emp
  WHERE sal >= 1500;
```

View 'SALVU1500' created.

```
SQL> SELECT *
  2 FROM salvu1500;
```

ID	NAME	MONTHLY_SALARY
7499	ALLEN	1600
7566	JONES	2975
7698	BLAKE	2850
7782	CLARK	2450
7788	SCOTT	3000
7839	KING	5000
7844	TURNER	1500
7902	FORD	3000

8 rows selected.



# CREATE VIEWS

## Example

S\_EMP 테이블에서 부서 번호가 30인 뷰를 'WITH CHECK OPTION'을 이용하여 생성

```
SQL> CREATE VIEW empvu30
2 AS SELECT *
3 FROM s_emp
4 WHERE deptno = 30
5 WITH CHECK OPTION;
```

부서 번호를 20으로 업데이트

```
UPDATE empvu30
SET deptno=20
WHERE deptno=30;
TBR-10010: Statement does not satisfy the WHERE clause of the view.
```

부서 번호가 30인 새로운 사원 정보 입력

```
SQL> INSERT INTO empvu30
VALUES (9999, 'TABU', 'STUDENT', NULL, '22-10-05', 1500, NULL, 30);

1 row inserted.
```

## CONFIRM VIEW NAMES AND STRUCTURES

**USER\_VIEWS**에서 현재 사용자에게 속한 뷰의 정보를 조회할 수 있다.

### Example

```
SQL> DESCRIBE user_views;
```

COLUMN_NAME	TYPE	CONSTRAINT
VIEW_NAME	VARCHAR(128)	
TEXT	LONG	

### Example

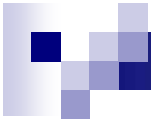
```
SQL> SELECT * FROM user_views;
```

```
VIEW_NAME
```

```
TEXT
```

```
SALVU1500
```

```
SELECT empno ID, ename NAME, sal MONTHLY_SALARY
FROM s_emp
WHERE sal >=
```



## DROP A VIEW

### Syntax

```
DROP VIEW view_name
```

### Example

EMPVU10 뷰 삭제

```
SQL> DROP VIEW empvu10;
```

View 'EMPVU10' dropped.

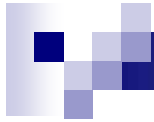
## Generate Primary Key Value

- 유일한 연속적 값을 생성할 수 있는 스키마 객체
- 기본 키 또는 유일 키에 값을 넣을 때 사용

### Syntax

```
CREATE SEQUENCE sequence_name
  [ INCREMENT by n ]
  [ START WITH n ]
  [ { MAXVALUE n | NOMAXVALUE } ]
  [ { CYCLE | NOCYCLE } ]
  [ { CACHE n | NOCACHE } ]
  [ { ORDER | NOORDER } ]
```

where	<i>table_name</i>	테이블 이름
-------	-------------------	--------



## Generate Primary Key Value

where	<i>table_name</i>	테이블 이름
	INCREMENT BY	시퀀스 간격(default : 1)
	START WITH	시퀀스 시작 값
	MAXVALUE	시퀀스 최댓값
	NOMAXVALUE	최댓값 지정 x
	CYCLE	최댓값 도달 시 재시작
	CACHE	캐시를 사용해서 미리 할당 (default : 20)
	ORDER	시퀀스 값 순서 유지

## Generate Primary Key Value

### Example

50부터 시작하고 10씩 증가하는 시퀀스 생성

```
SQL> CREATE SEQUENCE s_dept_id  
2 MINVALUE 1  
3 MAXVALUE 99999  
4 INCREMENT BY 10  
5 START WITH 50  
6 NOCACHE  
7 NOORDER  
8 NOCYCLE;
```

# CONFIRM SEQUENCES

## USER\_SEQUENCES Columns

Column	Description
SEQUENCE_NAME	시퀀스 이름
MIN_VALUE	최솟값
MAX_VALUE	최댓값
INCREMENT_BY	증가 값
CYCLE_FLAG	반복 유무
ORDER_FLAG	순서 유무
CACHE_SIZE	캐시할 시퀀스 번호 수
LAST_NUMBER	디스크에 기록된 마지막 시퀀스 번호

# CONFIRM SEQUENCES

## Example

```
SQL> DESC user_sequences;
```

COLUMN_NAME	TYPE	CONSTRAINT
SEQUENCE_NAME	VARCHAR(128)	
MIN_VALUE	NUMBER	
MAX_VALUE	NUMBER	
INCREMENT_BY	NUMBER	
CYCLE_FLAG	VARCHAR(1)	
ORDER_FLAG	VARCHAR(1)	
IF_Avail	VARCHAR(1)	
CACHE_SIZE	NUMBER	
LAST_NUMBER	NUMBER	



# CONFIRM SEQUENCES

## Example

```
SQL> SELECT sequence_name, min_value, max_value, increment_by, cycle_flag  
2 FROM user_sequences;
```

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG
S_EMP_ID	1	99999	10	N

1 row selected.

# REFERENCE PRIMARY KEY VALUES

**INSERT** 명령에서 시퀀스를 참조하여 값을 자동으로 생성

Expression	Description
sequence_name.NEXTVAL	시퀀스의 다음 값을 반환
sequence_name.CURRVAL	시퀀스의 마지막 값은 반환

## Example

새로운 부서 생성

```
SQL> INSERT INTO s_dept
  2 VALUES (s_dept_id.NEXTVAL, 'HR', 'SEOUL');

SQL> INSERT INTO s_dept
  2 VALUES (s_dept_id.NEXTVAL, 'FINANCE', 'MILPITAS');

SQL> SELECT * FROM s_dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	HR	SEOUL
60	FINANCE	MILPITAS

# DROP A SEQUENCE

## Syntax

```
DROP SEQUENCE sequence_name
```

## Example

S\_DEPT\_ID 시퀀스 삭제

```
SQL> DROP SEQUENCE s_dept_id;
```

Sequence 'S\_DEPT\_ID' dropped.

## Improve Query Performance

- 데이터베이스 테이블에 하나 이상의 인덱스를 작성하여 일부 쿼리 성능을 향상 시킬 수 있다.
- 자주 사용되는 **WHERE** 조건이나 **JOIN**
- 많은 양의 데이터 값을 가진 열
- 테이블 전체 데이터 중 **10-15%** 데이터를 처리하는 경우 효과적
- 테이블이 작거나 자주 업데이트 되는 경우 인덱스는 비효율적.

데이터베이스에서 테이블 데이터가 액세스 되는 방법

Access Method	Description
by ROWID	데이터의 정확한 위치를 나타내는 행 주소를 사용한 방법
FULL-TABLE SCAN	테이블의 모든 행을 순차적으로 검색하는 방법
by INDEX	열 값의 정렬된 트리 구조를 사용한 이진 검색

# Improve Query Performance

## Example

**ROWID로 테이블 데이터 액세스**

```
SQL> SELECT ename  
2 FROM s_emp  
3 WHERE rowid = 'AAArBAACAAAABFAAK';
```

**FULL-TABLE SCAN 으로 테이블 데이터 액세스**

```
SQL> SELECT ename  
2 FROM s_emp;
```



# CREATE INDEXS

UNIQUE 인덱스를 만들어 칼럼에 중복될 수 없는 유일 값 보장

## Syntax

```
CREATE INDEX index_name  
ON table_name (column_name [, column_name] ...)
```

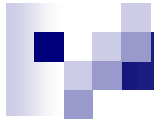
where	<i>index_name</i>	인덱스 이름
	<i>table_name</i>	테이블 이름
	<i>column_name</i>	열 이름

## Example

직원 이름 열에 인덱스 생성

```
SQL> CREATE INDEX s_emp_ename_i  
2 ON s_emp(ename);
```

Index 'S\_EMP\_ENAME\_I' created.



## CONFIRM THE EXISTENCE OF INDEXES

**USER\_INDEX**에서 인덱스 정보 확인

**USER\_INDEX** 테이블

Column	Description
INDEX_NAME	인덱스 이름
TABLE_OWNER	인덱스 소유자
TABLE_NAME	인덱싱된 개체 이름
TABLE_TYPE	인덱싱된 개체 유형
UNIQUENESS	인덱스의 고유성: UNIQUE or NONUNIQUE

# CONFIRM THE EXISTENCE OF INDEXES

**USER\_INDEX**에서 인덱스 정보 확인

```
SQL> DESCRIBE user_indexes;
```

COLUMN_NAME	TYPE	CONSTRAINT
INDEX_NAME	VARCHAR(128)	
INDEX_TYPE	VARCHAR(26)	
TABLE_OWNER	VARCHAR(128)	
TABLE_NAME	VARCHAR(128)	
TABLE_TYPE	VARCHAR(9)	
UNIQUENESS	VARCHAR(9)	



# CONFIRM THE EXISTENCE OF INDEXES

## Example

생성한 인덱스 표시

```
SQL> SELECT index_name, uniqueness
2 FROM user_indexes
3 WHERE table_name = 'S_EMP';
```

INDEX_NAME	UNIQUENESS
S_EMP_ID_PK	UNIQUE
S_EMP_ENAME_I	NONUNIQUE

	차이점	공통점
PK(Primary Key)	NOT NULL	중복될 수 없는 유일값
	하나의 PK	
	OBJECT - CONSTRAINT	
UNIQUE INDEX	NULL	
	여러 개 생성 가능	
	OBJECT - INDEX	

## DROP A INDEX

### Syntax

```
DROP INDEX index_name
```

### Example

S\_DEPT\_ID 시퀀스 삭제

```
SQL> DROP INDEX s_emp_ename_i;
```

Index 'S\_EMP\_ENAME\_I' dropped.

## Control User Access: Overview

- 데이터베이스 관리자는 사용자에게 **SQL** 보안 명령을 사용해 테이블에 대한 액세스 권한을 제공
- Control User Access
  - 데이터베이스에 대한 권한 제공
  - 테이블 및 시퀀스와 같은 사용자 개체에 대한 액세스를 제공하고 제거
  - 데이터 사전에서 주어진 권한 및 받은 권한 확인
  - 데이터베이스 개체에 대한 동의어 또는 대체 이름 작성

## SYSTEM PRIVILEGES: OVERVIEW

- 데이터베이스 관리자는 사용자에게 시스템 권한을 부여하여 사용자는 특정 작업을 수행할 수 있다.
- 시스템 권한은 명령을 실행할 수 있는 권한이다.

### Type of System Privileges

System Privilege	Description
In One's Own Schema	자신의 스키마에 테이블 및 시퀀스를 생성할 수 있는 권한
On all Objects of a Specified Type	모든 스키마에서 테이블 생성 및 테이블 또는 뷰를 업데이트할 수 있는 권한
On the System or a User	사용자를 생성할 수 있는 권한

## SYSTEM PRIVILEGES: OVERVIEW

- 60개가 넘는 고유한 시스템 권한이 있다.
- 각 시스템 권한을 통해 사용자는 특정 작업을 수행할 수 있다.

### Type of System Privileges

Class	System Privilege	Operations Permitted
SESSION	CREATE SESSION	데이터베이스 연결 허용
TABLE	CREATE TABLE	테이블 및 인덱스 생성
		CONNECT, DML, DROP, ALTER, TRUNCATE 가능
TABLE	SELECT ANY TABLE	모든 스키마에 모든 테이블, 뷰 쿼리 사용 가능

## GRANT SYSTEM PRIVILEGES

- 데이터베이스 관리자는 GRANT SQL 명령을 사용하여 사용자 및 역할 권한을 부여하고 취소할 수 있다

### Syntax

```
GRANT system_priv TO [user, role, PUBLIC] [WITH ADMIN OPTION]
```

where	<i>system_priv</i>	부여되는 시스템 권한
	<i>TO</i>	권한을 부여 받는 대상
	<i>user</i>	일반 사용자
	<i>role</i>	권한 받을 역할
	<i>PUBLIC</i>	권한 받을 공유 사용자
	<i>WITH ADMIN OPTION</i>	사용할 수 있는 특권, 남에게 부여할 수 있는 관리 권한

# GRANT SYSTEM PRIVILEGES

## Example

실습을 위한 새로운 사용자 생성

```
SQL> CREATE USER scott  
2 IDENTIFIED by tiberio;
```

GRANT SQL 명령을 사용해 **scott** 사용자의 스키마에 테이블을 생성할 수 있는 권한 부여

```
SQL> GRANT CREATE SESSION, CREATE TABLE TO scott;
```

Granted.

**scott** 사용자에게 스키마의 테이블을 변경할 수 있는 권한 부여

```
SQL> GRANT ALTER ANY TABLE TO scott;
```

Granted.

# CONFIRM SYSTEM PRIVILEGES GRANTED

**DBA\_SYS\_PRIVS에서 시스템 권한 확인**

```
SQL> DESCRIBE dba_sys_privs;
```

COLUMN_NAME	TYPE	CONSTRAINT
GRANTEE	VARCHAR(128)	
PRIVILEGE	VARCHAR(40)	
ADMIN_OPTION	VARCHAR(3)	

**scott 사용자에게 부여된 권한 조회**

```
SQL> SELECT grantee, privilege FROM dba_sys_privs
2 WHERE grantee='SCOTT';
```

GRANTEE	PRIVILEGE
SCOTT	CREATE SESSION
SCOTT	CREATE TABLE
SCOTT	ALTER ANY TABLE

3 rows selected.





## OBJECT PRIVILEGES

데이터베이스 관리자가 사용자에게 부여할 수 있는 객체 권한으로는 테이블, 뷰, 시퀀스, 프로시저가 있다.

스키마 객체 특권	테이블	뷰	시퀀스	PSM 프로그램 (프 러시저, 함수 등)	디렉터리
SELECT	O	O	O		
INSERT	O	O			
ALTER	O		O		
UPDATE	O	O			
DELETE	O	O			
TRUNCATE	O				
EXECUTE				O	
INDEX	O				
REFERENCES	O	O			
READ					O
WRITE					O

## GRANT OBJECT PRIVILEGES

- 데이터베이스 관리자는 GRANT 명령을 사용하여 사용자 및 역할 권한을 부여하고 취소할 수 있다.

### Syntax

GRANT *object\_priv* ON [*column*] OBJECT TO [*user, role, PUBLIC*] [*WITH GRANT OPTION*]

where	<i>object_priv</i>	부여되는 객체 권한
	<i>column</i>	특정 객체 일부 칼럼
	<i>(username.)OBJECT</i>	스키마 객체 권한 대상이 되는 객체
	<i>TO</i>	객체 권한을 부여 받는 사용자
	<i>WITH GRANT OPTION</i>	부여 받은 권한을 다른 사용자에게 부여할 수 있는 권한

# GRANT OBJECT PRIVILEGES

## Example

scott에게 S\_EMP 테이블을 조회 할 수 있는 권한 부여

```
SQL> GRANT SELECT ON s_emp TO scott;  
Granted.
```

scott에게 S\_EMP 테이블에 직원 번호, 직원 이름, 부서 번호를 삽입할 수 있는 권한과 월급을 수정할 수 있는 권한 부여

```
SQL> GRANT INSERT(empno, ename, deptno),  
      2 UPDATE(sal)  
      3 ON s_emp TO scott;  
  
Granted.
```

## GRANT OBJECT PRIVILEGES

### Example

scott에게 S\_DEPT 테이블을 조회 할 수 있는 권한과 다른 사람에게 동일한 권한을 부여할 수 있는 권한 부여

```
GRANT SELECT, INSERT
ON s_dept
TO scott
WITH GRANT OPTION;
```

scott이 시스템의 모든 사용자에게 sys의 S\_DEPT 테이블을 조회할 수 있는 권한 부여

```
SQL> conn scott
Enter Password:

SQL> GRANT SELECT
  2 ON sys.s_dept
  3 TO PUBLIC;

Granted.
```



## CONFIRM OBJECT PRIVILEGES GRANTED

### USER\_TAB\_PRIVS\_MADE Columns

Column	Description
GRANTEE	권한이 부여된 사용자 이름
TABLE_NAME	객체 이름
GRANTOR	권한을 부여한 사용자 이름
PRIVILEGE	부여된 권한
GRANTABLE	WITH GRANT OPTION 부여 유무

SQL> DESCRIBE user\_tab\_privs\_made

COLUMN_NAME	TYPE	CONSTRAINT
GRANTEE	VARCHAR(128)	
TABLE_NAME	VARCHAR(128)	
GRANTOR	VARCHAR(128)	
PRIVILEGE	VARCHAR(40)	
GRANTABLE	VARCHAR(3)	

# CONFIRM OBJECT PRIVILEGES GRANTED

## Example

현재 사용자가 **scott**에게 부여된 객체 조회

```
SQL> SELECT *
```

```
2 FROM user_tab_privs_made
```

```
3 WHERE GRANTEE = 'SCOTT';
```

GRANTEE	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE
SCOTT	S_DEPT	SYS	INSERT	YES
SCOTT	S_DEPT	SYS	SELECT	YES
SCOTT	S_EMP	SYS	SELECT	NO

3 rows selected.

# CONFIRM OBJECT PRIVILEGES GRANTED

## Example

사용자 **scott**이 자신에 부여된 객체 조회

```
SQL> SELECT *
2 FROM user_tab_privs_recd;
```

OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE
SYS	S_EMP	SYS	SELECT	NO
SYS	S_DEPT	SYS	SELECT	YES
SYS	S_DEPT	SYS	INSERT	YES

3 rows selected.

# REMOVE OBJECT PRIVILEGES

SQL 명령어 REVOKE를 사용해 다른 사용자에게 부여된 권한을 제거

## Syntax

```
REVOKE privilege, privilege...
ON object_name
FROM [user1_name, user2_name ... | PUBLIC | role]
[CASCADE CONSTRAINTS]
```

where	CASCADE CONSTRAINTS	REFERENCE 스키마 객체 권한을 회수하는 경우
		참조 무결성 제약조건을 지운 뒤 권한 회수 (CASCADE CONSTRAINT 을 사용하지 않고, 회수하면 에러 발생)

## Example

```
SQL> REVOKE SELECT, INSERT
2 ON s_dept
3 FROM scott;
```

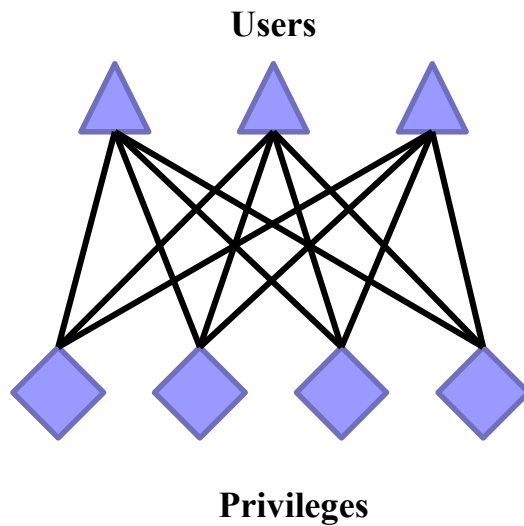
Revoked.



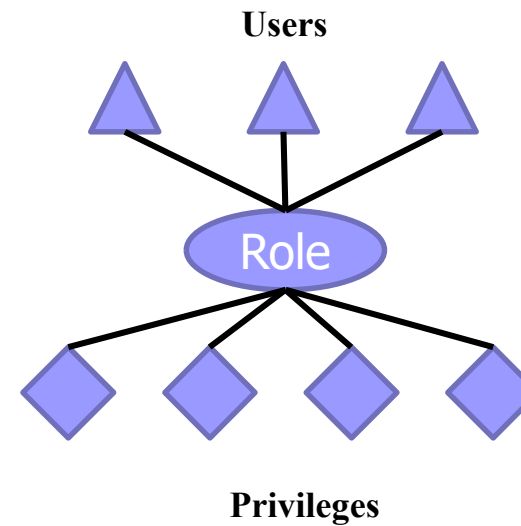
# PRIVILEGES GROUPED BY ROLE

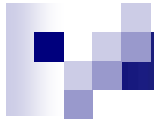
- 역할 사용을 통해 권한 관리를 단순화 시킴
- 시스템 권한과 객체 권한으로 구성 가능
- 역할은 사용자가 소유하지 않고, 스키마에도 존재하지 않음.

**Grant Privileges Without Roles**



**Grant Privileges With Roles**





## CREATE A ROLE

### Syntax

```
CREATE ROLE role_name [NOT IDENTIFIED] [IDENTIFIED BY password]
```

where	<i>role_name</i>	역할 이름
	<i>NOT IDENTIFIED</i>	패스워드 사용하지 않는다.(Default 값)
	<i>IDENTIFIED BY</i>	역할에 패스워드 설정
	<i>password</i>	패스워드

# CREATE A ROLE

## Example

역할 이름을 **acct\_rec**으로 생성

```
SQL> CREATE ROLE acct_rec;  
Role 'ACCT_REC' created.
```

패스워드가 **bicentennial**이고, 역할 이름을 **acct\_pay**으로 생성

```
SQL> CREATE ROLE acct_pay  
2 IDENTIFIED BY bicentennial;
```

동의어와 테이블 생성 권한이 있는 **manager** 역할 생성

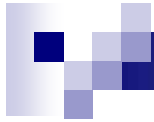
```
SQL> GRANT CREATE TABLE, CREATE SYNONYM  
2 TO manager;  
Granted.
```

신규 사용자 **kevin**에게 **manager** 권한 부여

```
SQL> GRANT manager TO kevin;  
  
Granted.
```



# DML : Data Manipulation Language



## 데이터 조작용어(Data Manipulation Language)

데이터 조작용어(이하 DML)는 데이터베이스에 저장된 데이터에 대한 질의, 삽입, 갱신, 삭제를 수행하기 위한 SQL 문장

명령어	설명
SELECT	데이터를 조회한다.
INSERT	데이터를 삽입한다.
UPDATE	데이터를 변경한다.
DELETE	데이터를 삭제한다.

# DISPLAY TABLE STRUCTURE

칼럼의 이름과 데이터 타입을 포함한 테이블 구조 확인

## Syntax

```
DESC[RIBE] tablename
```

## Example

```
SQL> DESCRIBE s_emp;
```

COLUMN_NAME	TYPE	CONSTRAINT
EMPNO	NUMBER(4)	PRIMARY KEY
ENAME	VARCHAR(10)	
JOB	VARCHAR(9)	
MGR	NUMBER(4)	
HIREDATE	DATE	
SAL	NUMBER(7,2)	
COMM	NUMBER(7,2)	
DEPTNO	NUMBER(2)	REFERENTIAL

INDEX_NAME	TYPE	COLUMN_NAME
PK_EMP	NORMAL	EMPNO

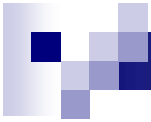
## Display Data with the SELECT Statement

SQL SELECT 문을 사용하여 데이터베이스 테이블의 데이터를 조회할 수 있다.

### Syntax

```
SELECT column_informantion
FROM table(s)
WHERE condition
ORDER BY expression or keyword
```

where	SELECT	검색할 열, 식 또는 상수를 지정
	FROM	데이터를 가지고 올 테이블 지정
	WHERE	특정 행을 검색할 기준(선택사항)
	ORDER BY	조회된 데이터를 정렬



# DISPLAY ALL DATA IN A TABLE

테이블의 모든 칼럼을 출력하려면 **SELECT** 키워드에 '\*'를 입력

## Syntax

```
SELECT *  
FROM table_name
```

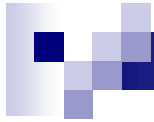
## Example

```
SQL> SELECT *  
2 FROM s_dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4 rows selected.





# DISPLAY ALL DATA IN A TABLE

조회하고 싶은 칼럼 이름과 해당 칼럼이 정의된 테이블을 입력하여 조회

## Syntax

```
SELECT [DISTINCT] column_name [, column_name]  
FROM table_name
```

## Example

```
SQL> SELECT dname  
2    FROM s_dept;
```

```
DNAME  
-----  
ACCOUNTING  
RESEARCH  
SALES  
OPERATIONS
```

4 rows selected.

# DISPLAY ALL DATA IN A TABLE

S\_EMP 테이블의 칼럼과 구조를 조회하고, 사원 이름, 급여를 출력

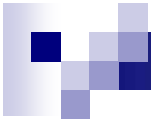
## Example

```
SQL> SELECT dname
2 FROM s_dept;
```

```
SQL> DESC s_emp;
```

```
SQL> SELECT ename, sal
2 FROM s_emp;
```

ENAME	SAL
SMITH	800
ALLEN	1600
WARD	1250
JONES	2975
MARTIN	1250
BLAKE	2850
CLARK	2450
SCOTT	3000
KING	5000
TURNER	1500
ADAMS	1100
JAMES	950
FORD	3000



# DISPLAY UNIQUE COLUMNS OF DATA

**DISTINCT** 절을 사용하여 고유한 데이터 행을 조회한다. **DISTINCT** 절은 **SELECT** 문 결과가 반환되기 전 중복된 행을 제거한다.

## Syntax

```
SELECT [DISTINCT] column_name [, column_name]  
FROM table_name
```

## Example

```
SQL> SELECT job  
2 FROM s_emp;  
JOB  
-----  
CLERK  
SALESMAN  
SALESMAN  
MANAGER  
SALESMAN  
MANAGER  
MANAGER  
ANALYST  
PRESIDENT  
SALESMAN  
CLERK  
CLERK  
ANALYST  
STUDENT
```

```
SQL> SELECT DISTINCT job  
2 FROM s_emp;  
  
JOB  
-----  
CLERK  
ANALYST  
STUDENT  
PRESIDENT  
SALESMAN  
MANAGER
```

## DISPLAY SPECIFIC COLUMN NAMES

별칭 - **SELECT** 문에서 칼럼 이름을 대체로 정의할 수 있다.

별칭에 공백, 특수문자가 포함되어 있거나, 대소문자 구분이 필요하면 별칭을 큰 따옴표로 묶는다.

### Example

```
SQL> SELECT DISTINCT job "Title"
2 FROM s_emp;
```

Title

-----

CLERK  
ANALYST  
STUDENT  
PRESIDENT  
SALESMAN  
MANAGER

```
SQL> SELECT ename EMPLOYEES
2 FROM s_emp;
```

EMPLOYEES

-----

SMITH  
ALLEN  
WARD  
JONES  
MARTIN  
BLAKE  
CLARK  
SCOTT  
KING  
TURNER  
ADAMS  
JAMES  
FORD

# DISPLAY SPECIFIC ROWS OF DATA

WHERE절을 사용해 조건에 맞는 행을 조회한다.

## Syntax

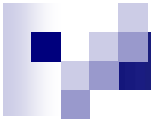
```
SELECT { * | column_name [, column_name...] }  
FROM table_name  
WHERE condition
```

## Example

```
SQL> SELECT ename  
2   FROM s_emp  
3   WHERE sal > 2000;
```

```
ENAME  
-----  
KING  
BLAKE  
CLARK  
JONES  
FORD  
SCOTT
```

6 rows selected.



# DISPLAY SPECIFIC ROWS OF DATA

## Comparison Operators Overview

=	Equal to
<>	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

# DISPLAY SPECIFIC ROWS OF DATA

## Comparison Operators Overview

<b>BETWEEN...AND...</b>	between two values
<b>NOT BETWEEN... AND...</b>	Not between two values
<b>IN (list)</b>	Equal to any member of the following list
<b>NOT IN (list)</b>	Not Equal to any member of the following list
<b>LIKE</b>	Match a character pattern using wildcard characters
<b>IS NULL</b>	Is a null
<b>IS NOT NULL</b>	Is not a null

## DISPLAY SPECIFIC COLUMN NAMES

=

부서 번호가 20인 직원의 부서 번호, 사원 이름, 직무를 조회

### Example

```
SQL> SELECT deptno, ename, job
2   FROM s_emp
3   WHERE deptno = 20;
```

DEPTNO	ENAME	JOB
20	JONES	MANAGER
20	FORD	ANALYST
20	SMITH	CLERK
20	SCOTT	ANALYST
20	ADAMS	CLERK



# DISPLAY SPECIFIC COLUMN NAMES

=

BLAKE의 사원 번호, 이름, 직무, 급여를 조회

## Example

```
SQL> SELECT empno, ename, job, sal
2 FROM s_emp
3 WHERE ename = 'Blake';
```

0 row selected.

```
SQL> SELECT empno, ename, job, sal
2 FROM s_emp
3 WHERE ename = 'BLAKE';
```

EMPNO	ENAME	JOB	SAL
7698	BLAKE	MANAGER	2850

1 row selected.

## DISPLAY SPECIFIC COLUMN NAMES

=

입사 날짜가 82/12/22일인 사원의 이름, 부서 번호, 입사 날짜 조회

### Example

```
SQL> SELECT ename, deptno, hiredate
2  FROM s_emp
3  WHERE hiredate = '82/12/22;
```

ENAME	DEPTNO	HIREDATE
SCOTT	20	0082/12/22

# DISPLAY SPECIFIC COLUMN NAMES

&lt; &gt;

직무가 **MANAGER**가 아닌 사원의 이름, 직무, 급여 조회

## Example

```
SQL> SELECT ename, job, sal
2   FROM s_emp
3   WHERE job <> 'MANAGER';
```

ENAME	JOB	SAL
KING	PRESIDENT	5000
MARTIN	SALESMAN	1250
ALLEN	SALESMAN	1600
TURNER	SALESMAN	1500
JAMES	CLERK	950
WARD	SALESMAN	1250
FORD	ANALYST	3000
SMITH	CLERK	800
SCOTT	ANALYST	3000
ADAMS	CLERK	1100
MILLER	CLERK	1300

11 rows selected.



## DISPLAY SPECIFIC COLUMN NAMES

&gt;

입사 날짜가 82/11/01보다 큰 사원의 이름, 입사 날짜 조회

### Example

```
SQL> SELECT ename, hiredate
2   FROM s_emp
3   WHERE hiredate > '82/11/01';
ENAME    HIREDATE
-----
SCOTT    82/12/22
ADAMS    83/01/15
```

급여가 2,000이상인 사원의 이름, 급여 조회

```
SQL> SELECT ename, sal
2   FROM s_emp
3   WHERE sal > 2000;
ENAME    SAL
-----
KING      5000
BLAKE     2850
CLARK     2450
JONES     2975
FORD      3000
SCOTT     3000
```

# DISPLAY SPECIFIC COLUMN NAMES

>=

부서 번호가 30이상인 사원의 부서 번호, 이름 조회

## Example

```
SQL> SELECT deptno, ename
2 FROM s_emp
3 WHERE deptno >= 30;
```

DEPTNO ENAME

```
-----
30 BLAKE
30 MARTIN
30 ALLEN
30 TURNER
30 JAMES
30 WARD
```

이름이 SCOTT 이상인 사원의 이름과 직무 조회

```
SQL> SELECT ename, job
2 FROM s_emp
3 WHERE ename >= 'SCOTT';
```

ENAME JOB

```
-----
TURNER SALESMAN
WARD SALESMAN
SMITH CLERK
SCOTT ANALYST
```

## DISPLAY SPECIFIC COLUMN NAMES

&lt;

입사 날짜가 81/05/01 보다 작은 사원의 이름, 부서 번호, 급여, 입사 날짜 조회

### Example

```
SQL> SELECT deptno, ename, sal, hiredate
2 FROM s_emp
3 WHERE hiredate < '81/05/01';
```

DEPTNO	ENAME	SAL	HIREDATE
20	JONES	2975	81/04/01
30	ALLEN	1600	81/02/11
30	WARD	1250	81/02/23
20	SMITH	800	80/12/09

4 rows selected.

## DISPLAY SPECIFIC COLUMN NAMES

&lt;=

부서 번호가 10이하인 사원의 이름, 부서 번호, 직무 조회

### Example

```
SQL> SELECT deptno, ename, job  
2 FROM s_emp  
3 WHERE deptno <= 10;
```

DEPTNO	ENAME	JOB
10	KING	PRESIDENT
10	CLARK	MANAGER
10	MILLER	CLERK

3 rows selected.

## DISPLAY SPECIFIC COLUMN NAMES

**BETWEEN**

입사 날짜가 80/12/01과 81/03/01 사이에 있는 사원의 이름과 입사 날짜 조회

### Example

```
SQL> SELECT ename, hiredate
  2 FROM s_emp
  3 WHERE hiredate BETWEEN '80/12/01' AND '81/03/01';
```

ENAME	HIREDATE
ALLEN	81/02/11
WARD	81/02/23
SMITH	80/12/09

급여가 1,500과 2,000 사이인 사원의 이름과 급여 조회

```
SQL> SELECT ename, sal
  2 FROM s_emp
  3 WHERE sal BETWEEN 1500 AND 2000;
```

ENAME	SAL
ALLEN	1600
TURNER	1500

2 rows selected.



## DISPLAY SPECIFIC COLUMN NAMES

**NOT  
BETWEEN**

부서 번호가 10과 20 사이에 있지 않은 부서 번호 조회

### Example

```
SQL> SELECT *
  2 FROM s_dept
  3 WHERE deptno NOT BETWEEN 10 AND 20;
```

DEPTNO	DNAME	LOC
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

사원 이름이 ALLEN과 SMITH 사이에 있지 않은 사원 이름과 부서 번호 조회

```
SQL> SELECT ename, deptno
  2 FROM s_emp
  3 WHERE ename NOT BETWEEN 'ALLEN' AND 'SMITH';
```

ENAME	DEPTNO
TURNER	30
WARD	30
ADAMS	20

# DISPLAY SPECIFIC COLUMN NAMES

**IN**

직무가 **MANAGER**, **SALESMAN**인 사원 이름, 직무, 부서 번호 조회

## Example

```
SQL> SELECT ename, job, deptno
2 FROM s_emp
3 WHERE job IN ('MANAGER', 'SALESMAN');
```

ENAME	JOB	DEPTNO
BLAKE	MANAGER	30
CLARK	MANAGER	10
JONES	MANAGER	20
MARTIN	SALESMAN	30
ALLEN	SALESMAN	30
TURNER	SALESMAN	30
WARD	SALESMAN	30

7 rows selected.

## DISPLAY SPECIFIC COLUMN NAMES

NOT IN

부서 번호가 10, 20이 아닌 사원의 사원 번호, 이름, 부서 번호 조회

### Example

```
SQL> SELECT empno, ename, deptno  
2 FROM s_emp  
3 WHERE deptno NOT IN (10,20);
```

EMPNO	ENAME	DEPTNO
7698	BLAKE	30
7654	MARTIN	30
7499	ALLEN	30
7844	TURNER	30
7900	JAMES	30
7521	WARD	30

6 rows selected.

## DISPLAY SPECIFIC COLUMN NAMES

**LIKE**

사원 이름이 'M'으로 시작하는 사원 이름 조회

### Example

```
SQL> SELECT ename
2 FROM s_emp
3 WHERE ename LIKE 'M%';
```

```
ENAME
-----
MARTIN
MILLER
```

입사 날짜가 '01'로 끝나는 사원 이름, 입사 날짜 조회

```
SQL> SELECT ename, hiredate
2 FROM s_emp
3 WHERE hiredate LIKE '%01';
```

```
ENAME    HIREDATE
-----
BLAKE     81/05/01
JONES     81/04/01
```

## DISPLAY SPECIFIC COLUMN NAMES

**LIKE**

사원 이름에 'A'를 포함하는 사원 조회

### Example

```
SQL> SELECT ename  
2 FROM s_emp  
3 WHERE ename LIKE '%A%';
```

ENAME

-----

BLAKE

CLARK

MARTIN

ALLEN

JAMES

WARD

ADAMS

7 rows selected.

## DISPLAY SPECIFIC COLUMN NAMES

**LIKE**

사원 이름이 'B'로 시작해서 'E' 끝나는 다섯 글자의 사원 이름 조회

### Example

```
SQL> SELECT ename
  2 FROM s_emp
  3 WHERE ename LIKE 'B__E';
ENAME
-----
BLAKE
```

두 번째 문자가 'L'인 사원의 이름 조회

```
SQL> SELECT ename
  2 FROM s_emp
  3 WHERE ename LIKE '_L%';

ENAME
-----
BLAKE
CLARK
ALLEN

3 rows selected.
```

# DISPLAY SPECIFIC COLUMN NAMES

IS NULL

커미션이 NULL 값인 사원의 이름, 직무, 급여 조회

## Example

```
SQL> SELECT ename, job, sal
2 FROM s_emp
3 WHERE comm IS NULL;
```

ENAME	JOB	SAL
KING	PRESIDENT	5000
BLAKE	MANAGER	2850
CLARK	MANAGER	2450
JONES	MANAGER	2975
JAMES	CLERK	950
FORD	ANALYST	3000
SMITH	CLERK	800
SCOTT	ANALYST	3000
ADAMS	CLERK	1100
MILLER	CLERK	1300

10 rows selected.

## DISPLAY SPECIFIC COLUMN NAMES

**IS NOT  
NULL**

커미션이 NULL 값인 아닌 사원의 이름, 직무, 커미션 조회

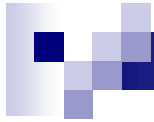
### Example

```
SQL> SELECT ename, job, comm
2 FROM s_emp
3 WHERE comm IS NOT NULL;
```

ENAME	JOB	COMM
MARTIN	SALESMAN	1400
ALLEN	SALESMAN	300
TURNER	SALESMAN	0
WARD	SALESMAN	500

4 rows selected.





## DISPLAY ROWS WITH COMPLEX CONDITIONS

AND, OR을 이용해 WHERE절에서 여러 조건을 결합하여 테이블 조회

### Syntax

```
SELECT { * | column_name [, column_name...] }  
FROM table_name  
WHERE condition {AND | OR} condition
```

# DISPLAY ROWS WITH COMPLEX CONDITIONS

**AND**

부서 번호가 20이고, 급여가 2,000 보다 큰 사원의 이름, 급여, 부서 번호 조회

## Example

```
SQL> SELECT ename, sal, deptno
2 FROM s_emp
3 WHERE deptno = 20
4 AND sal > 2000;
```

ENAME	SAL	DEPTNO
JONES	2975	20
FORD	3000	20
SCOTT	3000	20

부서 번호가 10이고, 직무가 MANAGER인 사원의 이름, 급여, 부서 번호, 직무 조회

```
SQL> SELECT ename, sal, deptno, job
2 FROM s_emp
3 WHERE deptno = 10
4 AND job = 'MANAGER';
```

ENAME	SAL	DEPTNO	JOB
CLARK	2450	10	MANAGER

# DISPLAY ROWS WITH COMPLEX CONDITIONS

OR

부서 번호가 20이거나, 직무가 SALESMAN인 사원의 이름, 급여, 부서 번호, 직무 조회

## Example

```
SQL> SELECT ename, sal, deptno, job
2 FROM s_emp
3 WHERE deptno = 20
4 OR job = 'SALESMAN';
```

ENAME	SAL	DEPTNO	JOB
JONES	2975	20	MANAGER
MARTIN	1250	30	SALESMAN
ALLEN	1600	30	SALESMAN
TURNER	1500	30	SALESMAN
WARD	1250	30	SALESMAN
FORD	3000	20	ANALYST
SMITH	800	20	CLERK
SCOTT	3000	20	ANALYST
ADAMS	1100	20	CLERK

9 rows selected.



## DISPLAY ROWS WITH COMPLEX CONDITIONS

우선 순위는 모든 비교 연산자, AND 그리고 OR 순서다.

### Rules of Precedence

ORDER Evaluated	Operator
1	All Comparison Operators (=, <>, >, >=, <, <=, IN, LIKE, IN NULL, BETWEEN...AND)
2	AND
3	OR

## DISPLAY ROWS WITH COMPLEX CONDITIONS

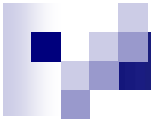
급여가 1,000 이상이고, 직무 번호가 10 또는 20인 사원의 이름, 급여, 직무 번호 조회

### Example

```
SQL> SELECT ename, sal, deptno
FROM s_emp
WHERE sal >= 1000 AND deptno = 10 OR deptno = 20;
```

ENAME	SAL	DEPTNO
SMITH	800	20
JONES	2975	20
CLARK	2450	10
SCOTT	3000	20
KING	5000	10
ADAMS	1100	20
FORD	3000	20

7 rows selected.



## ORDER THE ROWS DISPLAYED

특정 칼럼을 기준으로 정렬이 가능하다.

### Syntax

```
SELECT { * | column_name [, column_name...] }  
FROM table_name  
WHERE condition  
ORDER BY column_name {ASC | DESC} [, column_name [ASC | DESC] ...]
```

where	column_name	칼럼 이름
	table_name	테이블 이름
	ASC	행을 오름차순으로 정렬(Default)
	DESC	행을 내림차순으로 정렬

## ORDER THE ROWS DISPLAYED

급여를 기준으로 오름차순 정렬

### Example

```
SQL> SELECT ename, sal  
2 FROM s_emp  
3 WHERE deptno = 30  
4 ORDER BY sal;
```

ENAME	SAL
JAMES	950
MARTIN	1250
WARD	1250
TURNER	1500
ALLEN	1600
BLAKE	2850

6 rows selected.

## ORDER THE ROWS DISPLAYED

급여를 기준으로 내림차순 정렬

### Example

```
SQL> SELECT ename, sal  
2 FROM s_emp  
3 WHERE deptno = 30  
4 ORDER BY sal DESC;
```

ENAME	SAL
BLAKE	2850
ALLEN	1600
TURNER	1500
MARTIN	1250
WARD	1250
JAMES	950

6 rows selected.



## ORDER THE ROWS DISPLAYED

사원 번호, 사원 이름, 부서 번호를 사원 번호 및 부서 번호로 오름차순 정렬

### Example

```
SQL> SELECT empno, ename, deptno
2 FROM s_emp
3 ORDER BY empno, deptno;
```

EMPNO	ENAME	DEPTNO
7369	SMITH	20
7499	ALLEN	30
7521	WARD	30
7566	JONES	20
7654	MARTIN	30
7698	BLAKE	30
7782	CLARK	10
7788	SCOTT	20
7839	KING	10
7844	TURNER	30
7876	ADAMS	20
7900	JAMES	30
7902	FORD	20
7934	MILLER	10

14 rows selected.

## ORDER THE ROWS DISPLAYED

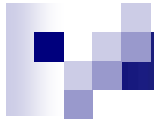
급여가 1,500이상인 사원의 이름, 부서 번호, 급여를 조회하고, 부서 번호로 오름차순, 급여로 내림차순하여 정렬

### Example

```
SQL> SELECT ename, deptno, sal
2 FROM s_emp
3 WHERE sal >= 1500
4 ORDER BY deptno, sal DESC;
```

ENAME	DEPTNO	SAL
KING	10	5000
CLARK	10	2450
FORD	20	3000
SCOTT	20	3000
JONES	20	2975
BLAKE	30	2850
ALLEN	30	1600
TURNER	30	1500

8 rows selected.



# INSERT DATA

INSERT 문을 이용해 테이블에 새로운 행을 추가할 수 있다.

## Syntax

INSERT INTO *table* [*column*] VALUES (*value*, *value*...)

where	<i>table</i>	테이블 이름
	<i>column</i>	칼럼 이름
	<i>value</i>	열에 해당하는 값

# INSERT DATA

INSERT 절에 칼럼을 나열하지 않고 테이블의 모든 칼럼에 대한 값을 삽입

## Example

```
SQL> DESC s_dept;
```

COLUMN_NAME	TYPE	CONSTRAINT
DEPTNO	NUMBER(2)	PRIMARY KEY
DNAME	VARCHAR(14)	
LOC	VARCHAR(13)	
INDEX_NAME	TYPE	COLUMN_NAME
PK_DEPT	NORMAL	DEPTNO

S\_DEPT 테이블에 'HR' 부서 정보 입력

```
SQL> INSERT INTO s_dept
2 VALUES (50, 'HR', 'SEOUL');
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	HR	SEOUL

# INSERT DATA

INSERT 절에 칼럼을 나열하고 테이블의 모든 칼럼에 대한 값을 삽입

## Example

```
SQL> DESCRIBE s_emp;
```

COLUMN_NAME	TYPE	CONSTRAINT
EMPNO	NUMBER(7)	PRIMARY KEY NOT NULL
ENAME	VARCHAR(10)	NOT NULL
JOB	VARCHAR(9)	
MGR	NUMBER(4)	REFERENTIAL
HIREDATE	DATE	
SAL	NUMBER(7)	
COMM	NUMBER(7)	
DEPTNO	NUMBER(2)	REFERENTIAL
INDEX_NAME	TYPE	COLUMN_NAME
S_EMP_ID_PK	NORMAL	EMPNO

# INSERT DATA

S\_EMP 테이블에 새로운 직원 정보 입력

## Example

```
SQL> INSERT INTO s_emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
VALUES (1234,'ALEX', 'DESIGNER', 7839, SYSDATE, 3000, NULL, 20);
```

1 row inserted.

```
SQL> SELECT *
2 FROM s_emp
3 WHERE ename = 'ALEX';
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1234	ALEX	DESIGNER	7839	2022/10/07	3000		20

1 row selected.

# INSERT SPECIAL VALUES

새로운 사원 CHRIS에 대한 정보를 입력하되, 직무, 담당 매니저, 급여 정보를 제공하지 않는다. 빈 문자열로 NULL을 명시적으로 삽입

## Example

```
SQL> INSERT INTO emp
2 VALUES (7633, 'CHRIS', '', '', 2500, '', 50);
```

1 row inserted.

새로운 사원 HENRY에 대한 정보를 입력하되, 직무, 담당 매니저, 급여 정보를 제공하지 않는다. NULL을 암시적으로 삽입

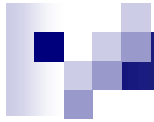
```
SQL> INSERT INTO s_emp(empno, ename, sal, deptno)
2 VALUES (7634, 'HENRY', 1300, 50);
```

1 row inserted.

```
SQL> SELECT *
2 FROM s_emp
3 WHERE empno IN(7633, 7634);
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7634	HENRY				1300		50
7633	CHRIS				2500		50

2 rows selected.



# INSERT SPECIAL VALUES

데이터베이스 사용자의 이름으로 S\_EMP 테이블에 정보 입력하기

## Example

```
SQL> INSERT INTO s_emp (empno, ename, hiredate, sal, deptno)
2 VALUES (7636, USER, SYSDATE, 2000, 10);
```

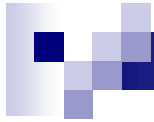
1 row inserted.

```
SQL> SELECT *
2 FROM s_emp
3 WHERE empno = 7636;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7636	SYS			2022/10/09	2000	10	

1 row selected.





# UPDATE DATA

UPDATE 문으로 기존 행 수정

## Syntax

```
UPDATE table
SET VALUES (column = value)
[WHERE condition]
```

where	<i>table</i>	업데이트 할 테이블 이름
	<i>column</i>	업데이트 할 칼럼 이름
	<i>value</i>	새로운 값
	<i>condition</i>	조건에 맞는 행을 업데이트

# UPDATE DATA

사원번호 7636의 부서 번호와 ALEX의 부서 번호와 급여 변경

## Example

```
SQL> UPDATE s_emp
2 SET deptno = 20
3 WHERE empno = 7636;
```

1 row updated.

```
SQL> UPDATE s_emp
2 SET deptno = 20, sal = 4000
3 WHERE ename = 'ALEX';
```

1 row updated.

```
SQL> SELECT empno, ename, sal, deptno
2 FROM s_emp
3 WHERE empno = 7636 OR ename = 'ALEX';
```

EMPNO	ENAME	SAL	DEPTNO
1234	ALEX	4000	20
7636	SYS	2000	20

2 rows selected.

## UPDATE DATA

회사에서 부서 번호 10인 직원들에게 커미션을 1,000씩 지급한다.

### Example

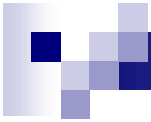
```
SQL> UPDATE s_emp  
2 SET comm = 1000  
3 WHERE deptno = 10;
```

3 rows updated.

```
SQL> SELECT ename, comm, deptno  
2 FROM s_emp  
3 WHERE deptno= 10;
```

ENAME	COMM	DEPTNO
CHRIS	1000	10
CLARK	1000	10
KING	1000	10

3 rows selected.



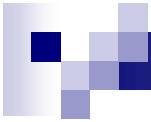
## DELETE DATA

DELETE 문으로 기존 행 삭제

### Syntax

```
DELETE FROM table  
[WHERE condition]
```

where	<i>table</i>	테이블 이름
	<i>condition</i>	조건에 맞는 행을 삭제



# DELETE DATA

S\_EMP 테이블에서 ALEX 사원 삭제

## Example

```
SQL> DELETE FROM s_emp  
2 WHERE ename = 'ALEX';
```

1 row deleted.

```
SQL> SELECT ename  
2 FROM s_emp  
3 WHERE ename = 'ALEX';
```

0 row selected.

# DELETE DATA

**S\_EMP** 테이블에서 부서 번호 **50**인 사원들 삭제

## Example

```
SQL> DELETE FROM s_emp  
2 WHERE deptno = 50;
```

2 row deleted.

```
SQL> SELECT ename  
2 FROM s_emp  
3 WHERE deptno = 50;
```

0 row selected.

조건이 없이 테이블 이름만 입력한 경우, 테이블 내 모든 데이터가 삭제된다. 전체 테이블을 삭제하는 경우가 아니면 **WHERE** 절을 생략하면 안된다.

# CONTROL TRANSACTIONS

- 트랜잭션(TRANSACTION) - INSERT, UPDATE, DELETE
- 데이터 조작 작업은 데이터베이스 버퍼에 영향을 준다.
- 현재 사용자는 SELECT 문으로 데이터 조작 작업의 결과를 검토할 수 있다.
- 다른 사용자는 현재 사용자에게 대한 데이터 조작 작업의 결과를 볼 수 없다.
- 영향을 받은 행은 LOCK이 걸리게 되고, 다른 사용자는 행을 변경할 수 없다.

## Control Transaction Logic

Statement	Description
COMMIT	현재 트랜잭션을 종료하고 트랜잭션의 갱신된 내용을 데이터베이스에 반영
ROLLBACK	현재 트랜잭션을 종료하고 트랜잭션에서 갱신된 내용 모두를 취소

# CONTROL TRANSACTIONS

## State of the Data After COMMIT

- COMMIT 문을 사용하여 보류중인 모든 변경 내용(INSERT, UPDATE, DELETE)을 영구적으로 만든다.
- COMMIT 후 데이터 변경 사항이 데이터베이스 파일에 기록된다.
- 영향을 받은 행은 LOCK이 해제되고, 다른 사용자가 행을 변경할 수 있다.

S\_DEPT 테이블에 새로운 부서 추가하고, COMMIT하기

## Example

```
SQL> INSERT INTO dept (deptno, dname, loc)
2 VALUES (60, 'LAW', 'LA');
```

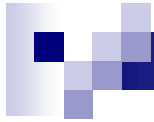
```
SQL> SELECT *
2 FROM dept
3 WHERE dname = 'LAW';
```

DEPTNO	DNAME	LOC
60	LAW	LA

```
SQL> COMMIT;
```

Commit completed.





# CONTROL TRANSACTIONS

## State of the Data After ROLLBACK

- 데이터의 변경이 취소되고, 데이터의 이전 상태가 복원
- 영향을 받은 행은 LOCK이 해제되고, 다른 사용자가 행을 변경할 수 있다.

# CONTROL TRANSACTIONS

S\_EMP 테이블에 부서 번호 20인 직원들의 행을 실수로 삭제했다. ROLLBACK을 이용해 복원한다.

## Example

```
SQL> DELETE FROM s_emp
2 WHERE deptno = 20;
```

5 rows deleted.

```
SQL> SELECT *
2 FROM s_emp
3 WHERE deptno = 20;
```

0 row selected.

```
SQL> ROLLBACK;
```

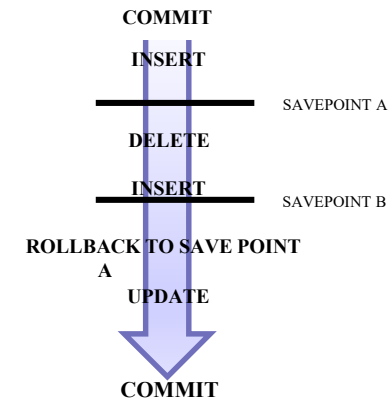
Rollback completed.

```
SQL> SELECT *
2 FROM s_emp
3 WHERE deptno = 20;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	0080/12/09	800	20	
7566	JONES	MANAGER	7839	0081/04/01	2975	20	
7788	SCOTT	ANALYST	7566	0082/12/22	3000	20	
7876	ADAMS	CLERK	7788	0083/01/15	1100	20	
7902	FORD	ANALYST	7566	0081/12/11	3000	20	

## CONTROL TRANSACTIONS

- **SAVEPOINT** 문을 이용하여 현재 트랜잭션에 저장 지점을 만든다.
- 부분 롤백을 수행하기 위해선 저장 지점을 미리 만들어야 한다.
- 동일한 이름의 저장 지점을 설정하면 이전의 저장 지점은 삭제된다.



### Alter Transaction Logic

Statement	Description
SAVEPOINT	현재 트랜잭션 내에서 저장 지점 표시
ROLLBACK TO SAVEPOINT	저장 지점이 표시된 후 보류 중인 변경 내용 삭제

# CONTROL TRANSACTIONS

**S\_EMP** 테이블에 새로운 사원의 정보를 추가하고 **SAVEPOINT a, b** 만들기

## Example

```
SQL> INSERT INTO s_emp(empno, ename, hiredate, sal)
  2 VALUES (3790, 'GOODMAN', SYSDATE, 2000);
```

1 row inserted.

```
SQL> SAVEPOINT a;
```

Savepoint created.

```
SQL> INSERT INTO s_emp(empno, ename, hiredate, sal)
  2 VALUES (3791, 'BADMAN', SYSDATE, 1000);
```

1 row inserted.

```
SQL> SAVEPOINT b;
```

Savepoint created.

```
SQL> INSERT INTO s_emp(empno, ename, hiredate, sal)
  2 VALUES (3792, 'YESMAN', SYSDATE, 3000);
1 row inserted.
```

```
SQL> SELECT empno, ename
  2 FROM s_emp;
```

EMPNO ENAME

```
-----
3790 GOODMAN
3791 BADMAN
3792 YESMAN
7369 SMITH
7499 ALLEN
7521 WARD
7566 JONES
7654 MARTIN
7698 BLAKE
7782 CLARK
7788 SCOTT
7839 KING
7844 TURNER
7876 ADAMS
7900 JAMES
7902 FORD
```

# CONTROL TRANSACTIONS

마지막 두 사원을 실행 취소하되, 첫 번째 사원은 그대로 둔다. **SAVEPOINT**를 이용해 영구적으로 변경한다.

## Example

```
SQL> ROLLBACK TO SAVEPOINT a;
```

Rollback completed.

```
SQL> COMMIT;
```

Commit completed.

```
SQL> SELECT empno, ename
2 FROM s_emp;
```

```
EMPNO ENAME
```

```
-----
3790 GOODMAN
7369 SMITH
7499 ALLEN
7521 WARD
7566 JONES
7654 MARTIN
7698 BLAKE
7782 CLARK
7788 SCOTT
7839 KING
7844 TURNER
7876 ADAMS
7900 JAMES
7902 FORD
```



# CONTROL TRANSACTIONS

- COMMIT 또는 ROLLBACK을 암묵적으로 수행하는 상황에 주의해야 한다.

## Implicit Transaction Processing

Circumstance	Result
CREATE TABLE과 같은 DDL 명령어 실행	자동 COMMIT
명시적 COMMIT 또는 ROLLBACK을 하지 않고 데이터베이스 정상 종료	자동 COMMIT
비정상적인 종료 또는 시스템 오류	자동 ROLLBACK



PERFORM COMPUTATIONS WITH DATA

## PERFORM COMPUTATIONS WITH DATA

산술 및 SQL 함수를 사용하여 다양한 방법으로 데이터를 수정하고 조회한다.

- 숫자 및 날짜 값을 사용하여 계산 수행
- NULL 값을 포함하는 계산 처리
- 숫자, 날짜 및 문자 값을 수정
- 날짜 값을 다양한 방식으로 표시
- 행 그룹에 대한 계산 조회 및 수행

Arithmetic Operators	
+	더하기
-	빼기
*	곱하기
/	나누기



## PERFORM COMPUTATIONS WITH NUMBERS

직무가 'MANAGER'인 사원에 대해 \$500의 급여 인상을 계산 후 사원 이름, 급여, 인상된 급여(NEW SALARY)를 출력하라

### Example

```
SQL> SELECT ename, sal, sal + 500 "NEW SALARY"
      FROM s_emp
      WHERE job = 'MANAGER'
      ORDER BY sal + 500;
```

ENAME	SAL	NEW SALARY
CLARK	2450	2950
BLAKE	2850	3350
JONES	2975	3475

3 rows selected.

## PERFORM COMPUTATIONS WITH NUMBERS

부서 번호가 20인 사원에 대해 급여의 10%를 보너스를 지급 후, 사원 이름, 급여, 보너스(BONUS)를 출력하라.

### Example

```
SQL> SELECT ename, sal, sal * 0.1 BONUS
2 FROM s_emp
3 WHERE deptno = 20
4 ORDER BY sal * 0.1;
```

ENAME	SAL	BONUS
SMITH	800	80
ADAMS	1100	110
JONES	2975	297.5
SCOTT	3000	300
FORD	3000	300

5 rows selected.

# PERFORM COMPUTATIONS WITH NUMBERS

## Rules of Precedence

산술 연산자 우선 순위는 다음과 같다.

1. 곱셈과 나누기 (\*, /)
2. 덧셈과 빼기 (+, -)

## Example

부서 번호 10인 사원에 대해 연간 보상을 지급한다. 이름, 급여, 연간 보상(ANNUAL COMPENSATION)을 출력하라. 연간 보상은 급여에 \$100를 상여 후 12를 곱한다.

```
SQL> SELECT ename, sal, (sal + 100) * 12 "ANNUAL COMPENSATION"
2 FROM s_emp
3 WHERE deptno = 10;
```

ENAME	SAL	ANNUAL COMPENSATION
CLARK	2450	30600
KING	5000	61200

2 rows selected.

# PERFORM COMPUTATIONS WITH NUMBERS

## ROUND

**ROUND(num1, num2)** – num1을 소수점 아래 num2 위치에서 반올림한 값을 반환

### Example

직무가 CLERK인 사원에게 급여를 근무 일수로 나눈 성과급을 지급한다. 이름, 급여, 성과급(PERFORMANCE PAY)을 출력하라. 근무 일수는 22일, 성과급은 소수점 둘째 자리에서 반올림 한다.

```
SQL> SELECT ename, sal, ROUND(sal/22, 2)
2 FROM s_emp
3 WHERE job = 'CLERK';
```

ENAME	SAL ROUND(SAL/22,2)	
-----		
SMITH	800	36.36
ADAMS	1100	50
JAMES	950	43.18

# PERFORM COMPUTATIONS WITH NUMBERS

## TRUNC

**TRUNC(num1, num2)** – num1을 소수점 아래 num2 위치에서 버림한 값을 반환

### Example

부서 번호가 20인 사원에게 급여를 근무 일수로 나눈 성과급을 지급한다. 이름, 급여, 성과급(PERFORMANCE PAY)을 출력하라. 근무 일수는 30일, 성과급은 소수점 둘째 자리에서 버림 한다.

```
SQL> SELECT ename, sal, TRUNC(sal /30, 2)
2 FROM s_emp
3 WHERE deptno = 20;
```

ENAME	SAL	TRUNC(SAL/30,2)
SMITH	800	26.66
JONES	2975	99.16
SCOTT	3000	100
ADAMS	1100	36.66
FORD	3000	100

# PERFORM COMPUTATIONS WITH NUMBERS

## MOD

**MOD(num1, num2)** – num1을 num2로 나눈 나머지를 반환하는 함수

### Example

사원 번호가 짝수인 사원의 정보를 모두 출력하라.

```
SQL> SELECT *
2 FROM s_emp
3 WHERE MOD(empno, 2) = 0;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
3790	GOODMAN			2022/10/09	2000		
7566	JONES	MANAGER	7839	1981/04/01	2975	20	
7654	MARTIN	SALESMAN	7698	1981/09/10	1250	1400	30
7698	BLAKE	MANAGER	7839	1981/05/01	2850	30	
7782	CLARK	MANAGER	7839	1981/05/09	2450	10	
7788	SCOTT	ANALYST	7566	1982/12/22	3000	20	
7844	TURNER	SALESMAN	7698	1981/08/21	1500	0	30
7876	ADAMS	CLERK	7788	1983/01/15	1100	20	
7900	JAMES	CLERK	7698	1981/12/11	950	30	
7902	FORD	ANALYST	7566	1981/12/11	3000	20	

10 rows selected.

# PERFORM COMPUTATIONS WITH NUMBERS

NVL

**NVL(*expr1*, *expr2*)** – *expr1*이 NULL이 아니면 *expr1*을 반환하고, *expr1*이 NULL이면 *expr2*를 반환하는 함수

## Example

부서 번호가 30인 사원에게 급여와 커미션을 더한 금액을 지급한다. 사원 이름, 급여, 커미션, 추가 금액을 출력하라. 단, 커미션이 없으면 0으로 출력하라

```
SQL> SELECT ename, sal, comm, NVL(sal + comm, 0)
2 FROM s_emp
3 WHERE deptno = 30;
```

ENAME	SAL	COMM	NVL(SAL+COMM,0)
ALLEN	1600	300	1900
WARD	1250	500	1750
MARTIN	1250	1400	2650
BLAKE	2850		0
TURNER	1500	0	1500
JAMES	950		0

## PERFORM COMPUTATIONS WITH DATES

산술 연산자를 이용해 날짜를 계산할 수 있다.

Date Expression
<i>date + number</i>
<i>date - number</i>
<i>date - date</i>

### Example

급여가 2,000 이상인 사원의 견습 기간 종료일을 확인한다. 이름, 입사 날짜, 견습 기간 종료일을 출력하라. 견습 기간은 입사 후 90일까지이다.

```
SQL> SELECT ename, hiredate, hiredate + 90
2 FROM s_emp
3 WHERE sal >= 2000;
```

ENAME	HIREDATE	HIREDATE+90
JONES	1981/04/01	1981/06/30
BLAKE	1981/05/01	1981/07/30
CLARK	1981/05/09	1981/08/07
SCOTT	1982/12/22	1983/03/22
KING	1981/11/17	1982/02/15
FORD	1981/12/11	1982/03/11



# PERFORM COMPUTATIONS WITH DATES

**SYSDAT  
E**

SYSDATE 함수를 이용해 오늘 날짜를 출력할 수 있다.

## Example

부서 번호 20인 사원의 이름과 근무 기간을 일, 월로 출력한다. 단, 월은 30일로 계산한다

```
SQL> SELECT ename, SYSDATE - hiredate DAYS, (SYSDATE - hiredate) / 30 MONTHS
2 FROM s_emp
3 WHERE deptno = 20;
```

ENAME	DAYS	MONTHS
SMITH	15280.3261	509.344204
JONES	15167.3261	505.577537
SCOTT	14537.3261	484.577537
ADAMS	14513.3261	483.777537
FORD	14913.3261	497.110871

5 rows selected.

# PERFORM COMPUTATIONS WITH DATES

## ADD\_MONTHS

**ADD\_MONTHS(date, integer)** – date에 integer만큼의 달을 더한 결과를 구하는 함수

### Example

직무가 **MANAGER**인 사원의 이름과 입사 날짜에 6개월을 더한 값을 출력한다.

```
SQL> SELECT ename, ADD_MONTHS(hiredate, 6)
2 FROM s_emp
3 WHERE job = 'MANAGER';
```

ENAME	ADD_MONTHS(HIREDATE,6)
JONES	1981/10/01
BLAKE	1981/11/01
CLARK	1981/11/09

3 rows selected.

## PERFORM COMPUTATIONS WITH DATES

### LAST\_DAY

**LAST\_DAY(*date*)** - 특정 일자에 해당하는 월의 마지막 일자를 표시하는 함수

### Example

현재 달의 마지막 일자를 출력하라

```
SQL> SELECT LAST_DAY(SYSDATE)
2 FROM dual;
```

```
LAST_DAY(SYSDATE)
-----
2022/10/31
```

```
1 row selected.
```

### NEXT\_DAY

**NEXT\_DAY(*date*, *str*)** - *date*와 가장 가까운 다음 요일 *str*의 날짜를 반환하는 함수

# PERFORM COMPUTATIONS WITH DATES

## MONTHS\_BETWEEN

**MONTHS\_BETWEEN(date1, date2)** – date1과 date2의 개월 차를 구하는 함수. 두 날짜 사이의 일수를 31로 나눈 값을 반환

### Example

직무가 SALESMAN인 사원의 이름, 입사일, 근무 개월을 출력하라

```
SQL> SELECT ename, hiredate, MONTHS_BETWEEN(SYSDATE, HIREDATE)
FROM s_emp
WHERE job = 'MANAGER';
```

ENAME	HIREDATE	MONTHS_BETWEEN(SYSDATE, HIREDATE)
ALLEN	1981/02/11	499.979055
WARD	1981/02/23	499.591958
MARTIN	1981/09/10	493
TURNER	1981/08/21	493.656474

4 rows selected.

## REFORMAT DATES

### 날짜 형식

Element	Description	Element	Description
DD	1개월 중 몇 번째 날인지 출력( 1- 31)	YYYY	연도를 네 자리로 출력 (2022)
DY	축약 표기한 요일을 출력 (ex : THU)	HH:MI:SS	시간, 분, 초 (10:00:00)
DAY	요일을 출력 ( ex : MONDAY)	Q	분기를 출력 (1-4)
MM	달을 출력 ( 1-12)	HH24	시간을 출력 (0 – 23)
MON	축약된 달 이름을 출력 ( ex : DEC)	RM	달을 로마 숫자로 출력 (I-XII)
MONTH	달을 출력한다 (ex : NOVEMBER)	AM or PM	오전 또는 오후
YY	연도를 두 자리로 출력 (22)	TZH or THM	시간대에서 시간 or 분 출력

# REFORMAT DATES

## TO\_CHAR

**TO\_CHAR(date\_value, format\_mask)** – date\_value를 format\_mask 형식에 따라 문자열로 변환 하는 함수

### Example

직무가 **MANAGER**인 사원의 이름, 입사일을 **YY년 MM월 DD일** 형식으로 출력하라

```
SQL> SELECT ename, TO_CHAR(HIREDATE,'YY"년"MM"월"DD"일")
FROM s_emp
WHERE job = 'MANAGER';
```

ENAME	HIREDATE	TO_CHAR(HIREDATE,'YY"년"MM"월"DD"일")
JONES	1981/04/01	81 년 04 월 01
BLAKE	1981/05/01	81 년 05 월 01
CLARK	1981/05/09	81 년 05 월 09

3 rows selected.

# REFORMAT DATES

TO\_DATE

**TO\_CHAR(character\_value, format\_mask)** – character\_value를 format\_mask 형식에 따라 Date로 변환 하는 함수

## Example

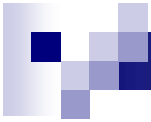
S\_EMP 테이블에 신입사원을 추가한다. 입사일은 900708이다

```
SQL> INSERT INTO s_emp(empno, ename, hiredate, deptno)
  2 VALUES (8371, 'MARU', TO_DATE('19900708', 'YYYYMMDD'), 40);
```

1 row inserted.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
8371	MARU			1990/07/08			40

1 row selected.



# MANIPULATE CHARACTER STRINGS

||

접합 연산자로 문자열을 결합 시킨다.

## Example

부서 번호 20인 사원의 사원 번호와 이름을 합쳐 출력한다.

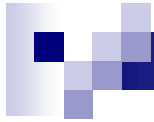
```
SQL> SELECT empno || ' ' || ename "ID AND EMPLOYEE"  
2 FROM s_emp  
3 WHERE deptno = 20;
```

ID AND EMPLOYEE

-----  
7369 SMITH  
7566 JONES  
7788 SCOTT  
7876 ADAMS  
7902 FORD

5 rows selected.





# MANIPULATE CHARACTER STRINGS

문자 함수

**NAME : Delhi Sports**

Function	EXAMPLE	RESULT
INITCAP	INITCAP (NAME)	Delhi Sports
UPPER	UPPER (NAME)	DELHI SPORTS
LOWER	LOWER (NAME)	delhi sports
SUBSTR	SUBSTR (NAME, 1, 4)	Delh
LENGTH	LENGTH (NAME)	12

# MANIPULATE CHARACTER STRINGS

## INITCAP

INITCAP – 첫 글자만 대문자로 변환한다.

### Example

직무가 CLERK인 사원의 이름, 직무를 출력한다. 이름은 첫 글자는 대문자, 나머지는 소문자로 출력한다.

```
SQL> SELECT INITCAP(ename) NAME, job
2 FROM s_emp
3 WHERE job = 'CLERK';
```

NAME	JOB
Smith	CLERK
Adams	CLERK
James	CLERK

3 rows selected.

# MANIPULATE CHARACTER STRINGS

## UPPER

**UPPER** - 모든 문자를 대문자로 변환한다.

### Example

INITCAP 함수를 이용해 출력한 직원 이름을 UPPER 함수를 이용해 대문자로 변환한다.

```
SQL> SELECT UPPER(INITCAP(ename)) NAME, job
2 FROM s_emp
3 WHERE job = 'CLERK';
```

NAME	JOB
SMITH	CLERK
ADAMS	CLERK
JAMES	CLERK

3 rows selected.

# MANIPULATE CHARACTER STRINGS

## LOWER

**LOWER** - 모든 문자를 소문자로 변환한다.

### Example

부서 번호 30인 직원 이름, 직무를 출력한다. **LOWER** 함수를 이용해 모두 소문자로 변환한다.

```
SQL> SELECT LOWER(ename), LOWER(job)
2 FROM s_emp
3 WHERE deptno = 30;
```

```
LOWER(ENAME) LOWER(JOB)
```

```
-----
allen      salesman
ward       salesman
martin     salesman
blake      manager
turner     salesman
james      clerk
```

```
6 rows selected.
```

# MANIPULATE CHARACTER STRINGS

## SUBSTR

**SUBSTR(char, m [, n])** – char 내 m 번째 위치로부터 n 길이의 문자열을 추출하는 함수. n이 지정되지 않으면 마지막까지 추출

### Example

부서 번호 10인 직원 이름을 뒤에서 첫 번째에서 2개까지 출력한다.

```
SQL> SELECT ename, SUBSTR(ename, 1, 2)
2 FROM s_emp
3 WHERE deptno = 10;
```

```
ENAME      SUBSTR(ENAME,1,2)
-----
CLARK      CL
KING       KI
```

2 rows selected.

# MANIPULATE CHARACTER STRINGS

## LENGTH

**LENGTH** – 문자열의 길이를 반환하는 함수

### Example

부서 번호 30인 직원 이름과 이름의 길이를 출력하라.

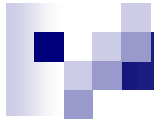
```
SQL> SELECT ename, LENGTH(ename)
2 FROM s_emp
3 WHERE deptno = 30;
```

ENAME	LENGTH(ENAME)
ALLEN	5
WARD	4
MARTIN	6
BLAKE	5
TURNER	6
JAMES	5

이름 길이가 6이상인 직원의 이름을 출력하라

```
SQL> SELECT ename
2 FROM s_emp
3 WHERE LENGTH(ename) >= 6;
```

ENAME
MARTIN
TURNER



## PERFORM SUMMARY COMPUTATIONS

숫자 함수

Function	Description
AVG	평균값
MAX	최댓값
MIN	최솟값
SUM	합
COUNT	로우의 개수를 세는 함수

# MANIPULATE CHARACTER STRINGS

AVG SUM
MIN MAX

## Example

직무가 SALESMAN 인 사원들의 급여 평균(AVERAGE), 최댓값(MAXIMUM), 최솟값(MINIMUM), 합(SUM)을 출력하라

```
SQL> COLUMN average FORMAT $99,999.99
SQL> COLUMN maximum FORMAT $99,999.99
SQL> COLUMN minimum FORMAT $99,999.99
SQL> COLUMN sum FORMAT $99,999.99
SQL> SELECT AVG(sal) average, MAX(sal) maximum, MIN(sal) minimum, SUM(sal) sum
2 FROM s_emp
3 WHERE job = 'SALESMAN';
```

AVERAGE	MAXIMUM	MINIMUM	SUM
\$1,400.00	\$1,600.00	\$1,250.00	\$5,600.00

1 row selected.



# MANIPULATE CHARACTER STRINGS

COUNT

## Example

S\_EMP 테이블에 있는 총 사원의 수를 구하라

```
SQL> SELECT COUNT(*)
      2 FROM s_emp;
```

```
COUNT(*)
```

```
-----
      14
```

1 row selected.

커미션을 받는 사원의 수를 구하라

```
SQL> SELECT COUNT(comm) "Employees with Comm"
      2 FROM s_emp;
```

```
Employees with Comm
```

```
-----
       4
```

1 row selected.

## GROUP ROWS TOGETHER

부서 번호가 30인 사원은 6명으로 6번 표시가 된다. **GROUP BY** 절을 사용하면 각 부서에 대해 한 줄로 출력할 수 있다.

```
SQL> SELECT empno, ename, deptno
2 FROM s_emp
3 WHERE deptno = 30;
```

EMPNO	ENAME	DEPTNO
7499	ALLEN	30
7521	WARD	30
7654	MARTIN	30
7698	BLAKE	30
7844	TURNER	30
7900	JAMES	30

6 rows selected.

```
SQL> SELECT deptno, COUNT(*) NUBMER
FROM s_emp
WHERE deptno = 30
GROUP BY deptno;
```

DEPTNO	NUBMER
30	6

1 row selected.

## GROUP ROWS TOGETHER

GROUP BY 및 HAVING 절이 있는 행 그룹에 대해 요약 결과를 출력한다.

### Syntax

```
SELECT column_name
FROM table_name
WHERE condition
GROUP BY group_by_expression
```

**Example** where

*group\_by\_expression*  
*n*

그룹화가 되는 기준의 열을 지정

S\_EMP 테이블에 있는 직무를 출력한다.

```
SQL> SELECT job, COUNT(*) "Number"
2 FROM s_emp
3 GROUP BY job;
```

JOB	Number
CLERK	3
ANALYST	2
PRESIDENT	1
SALESMAN	4
MANAGER	3

# GROUP ROWS TOGETHER

## Example

부서 번호에 따른 직원 수 출력

```
SQL> SELECT deptno, COUNT(*) "Head Count"
2 FROM s_emp
3 GROUP BY deptno;
```

DEPTNO Head Count	
10	2
40	1
20	5
30	6

**GROUP BY** 절 없이 정규 열과 그룹 함수를 함께 사용하면 안된다.

```
SQL> SELECT deptno, COUNT(*) "Employees Within Titles"
2 FROM s_emp;
TBR-8038: Expression is not in a GROUP BY clause.
```

## GROUP ROWS TOGETHER

두 이상의 **GROUP BY** 칼럼을 나열하여 그룹 및 하위 그룹에 대한 결과를 출력한다.

### Example

```
SQL> SELECT deptno, job, COUNT(*) "Employees Within Titles"
2 FROM s_emp
3 WHERE deptno = 30
4 GROUP BY deptno, job;
```

DEPTNO	JOB	Employees Within Titles
30	CLERK	1
30	SALESMAN	4
30	MANAGER	1

```
SQL> SELECT job, deptno, COUNT(*) "Employees Within Titles"
2 FROM s_emp
3 WHERE deptno = 30
4 GROUP BY job, deptno ;
```

JOB	DEPTNO	Employees Within Titles
CLERK	30	1
SALESMAN	30	4
MANAGER	30	1

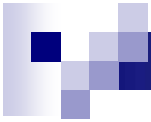
## DISPLAY SPECIFIC GROUPS

특정 행 또는 특정 그룹 출력

### Syntax

```
SELECT column_name [,column_name]  
FROM table_name  
WHERE condition  
GROUP BY group_by_expression  
HAVING condition
```

where	<i>condition</i>	지정된 조건이 참(TRUE)인 그룹만 반환
-------	------------------	-------------------------



## DISPLAY SPECIFIC GROUPS

직원이 세 명 이상인 부서의 급여 평균과 직원 수를 출력하라

### Example

```
SQL> SELECT deptno, AVG(sal) average, COUNT(*) "Number of Employees"  
2 FROM s_emp  
3 GROUP BY deptno  
4 HAVING COUNT(*) >= 3;
```

DEPTNO	AVERAGE	Number of Employees
--------	---------	---------------------

20	\$2,175.00	5
30	\$1,566.67	6

2 rows selected.

## DISPLAY SPECIFIC GROUPS

각 직무 별 급여의 합이 5,000보다 큰 직무와 급여 합을 출력하라. 급여의 합은 내림차순으로 정렬

### Example

```
SQL> SELECT job, SUM(sal) sum
2 FROM s_emp
3 GROUP BY job
4 HAVING SUM(sal) > 5000
5 ORDER BY SUM(sal) DESC;
```

JOB	SUM
MANAGER	\$8,275.00
ANALYST	\$6,000.00
SALESMAN	\$5,600.00

3 rows selected.





DISPLAY DATA FROM MULTIPLE TABLES

# DISPLAY DATA FROM MULTIPLE TABLES

- 조인을 작성하여 단일 SELECT 문 안에 있는 관련 테이블의 데이터를 표시한다.
- 두 개 이상의 테이블에 유사한 정보가 들어 있는 경우 동일한 테이블에 모두 저장된 것처럼 정보를 제공한다.

Desired Display
직원 이름과 해당 부서
부서명과 해당 지역

ID	LAST_NAME	FIRST_NAME	USERID	START_DATE	COMMENTS	MANAGER_ID	TITLE	DEPT_ID	SALARY	COMMISSION_PCT
1	Velasquez	Carmen	cvelasqu	1990-03-03			President	50	2500	
2	Ngao	LaDoris	Ingao	1990-03-08		1	VP, Operations	41	1450	
3	Nagayama	Midori	mnagayam	1991-06-17		1	VP, Sales	31	1400	
4	Quick-To-See	Mark	mquickto	1990-04-07		1	VP, Finance	10	1450	
5	Ropeburn	Audry	aropebur	1990-03-04		1	VP, Administration	50	1550	
6	Urguhart	Molly	murguhar	1991-01-18		2	Warehouse Manager	41	1200	
7	Menchu	Roberta	rmenchu	1990-05-14		2	Warehouse Manager	42	1250	
8	Biri	Ben	bbiri	1990-04-07		2	Warehouse Manager	43	1100	
9	Catchpole	Antoinette	acatchpo	1992-02-09		2	Warehouse Manager	44	1300	
10	Havel	Marta	mihavel	1991-02-27		2	Warehouse Manager	45	1307	
11	Magee	Colin	cmagee	1990-05-14		3	Sales Representative	31	1400	10
12	Giljum	Henry	hgijum	1992-01-18		3	Sales Representative	32	1490	12.5
13	Sedeghi	Yasmin	ysedeghi	1991-02-18		3	Sales Representative	33	1515	10
14	Nguyen	Mai	mnguyen	1992-01-22		3	Sales Representative	34	1525	15
15	Dumas	Andre	adumas	1991-10-09		3	Sales Representative	35	1450	17.5
16	Maduro	Elena	emaduro	1992-02-07		6	Stock Clerk	41	1400	
17	Smith	George	gsmith	1990-03-08		6	Stock Clerk	41	940	
18	Nozaki	Akira	anozaki	1991-02-09		7	Stock Clerk	42	1200	
19	Patel	Vikram	vpatel	1991-08-06		7	Stock Clerk	42	795	
20	Newman	Chad	cnewman	1991-07-21		8	Stock Clerk	43	750	
21	Markarian	Alexander	amarkari	1991-05-26		8	Stock Clerk	43	850	
22	Chang	Eddie	echang	1990-11-30		9	Stock Clerk	44	800	
23	Patel	Radha	rpatel	1990-10-17		9	Stock Clerk	34	795	
24	Dancs	Bela	bdancs	1991-03-17		10	Stock Clerk	45	860	
25	Schwartz	Sylvie	sschwart	1991-05-09		10	Stock Clerk	45	1100	

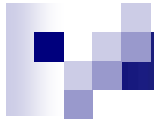
ID	NAME
1	North America
2	South America
3	Africa / Middle East
4	Asia
5	Europe

REGION

ID	NAME	REGION_ID
10	Finance	1
31	Sales	1
32	Sales	2
33	Sales	3
34	Sales	4
35	Sales	5
41	Operations	1
42	Operations	2
43	Operations	3
44	Operations	4
45	Operations	5
50	Administration	1

DEPT

EMP



## DISPLAY DATA FROM MULTIPLE TABLES

- View를 생성해서 여러 테이블을 단일 테이블인 것처럼 참조할 수 있다.
- WHERE 절에 간단한 조인 조건을 작성하여 둘 이상의 관련 테이블의 행을 조회한다.

### Syntax

```
SELECT table.column, table.column...  
FROM table1, table2...  
WHERE table1.column = table2.column
```

where	<i>table.column</i>	데이터를 검색하는 테이블 및 칼럼
	<i>table1.column = table2.column</i>	테이블을 결합 하는 조건, 칼럼 이름 앞에 테이블 명을 표시

## DISPLAY DATA FROM RELATED TABLES

모든 직원의 성, 부서 ID 그리고 부서 이름을 출력하라

### Example

```
SQL> SELECT e_emp.last_name, e_emp.dept_id, e_dept.name
2 FROM e_emp, e_dept
3 WHERE e_emp.dept_id = e_dept.id;
```

LAST_NAME	DEPT_ID	NAME
Velasquez	50	Administration
Ngao	41	Operations
Nagayama	31	Sales
Quick-To-See	10	Finance
Ropeburn	50	Administration
Urguhart	41	Operations
...		

25 rows selected.



## DISPLAY DATA FROM RELATED TABLES

모든 부서의 부서 ID, 지역 ID 및 지역 이름을 출력하라

### Example

```
SQL> SELECT e_dept.id "Department ID",  
2 e_region.id "Region ID",  
3 e_region.name "Region Name"  
4 FROM e_dept, e_region  
5 WHERE e_dept.region_id = e_region.id;
```

Department ID	Region ID	Region Name
---------------	-----------	-------------

50	1	North America
10	1	North America
41	1	North America
42	2	South America

...

12 rows selected.



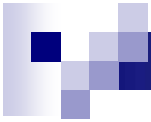
## DISPLAY DATA FROM RELATED TABLES

북미(North America) 지역의 모든 부서의 지역 ID, 지역 이름, 부서 ID 및 부서 이름을 출력하라

### Example

```
SQL> SELECT e_region.id "Region ID",  
           e_region.name "Region Name",  
           e_dept.id "Dept ID",  
           e_dept.name "Dept Name"  
FROM e_dept, e_region  
WHERE e_region.id = e_dept.region_id  
AND e_region.name = 'North America';
```

Region ID	Region Name	Dept ID	Dept Name
1	North America	50	Administration
1	North America	10	Finance
1	North America	41	Operations
1	North America	31	Sales



## DISPLAY DATA WITHOUT A DIRECT MATCH

- 외부 조인이 있는 다른 테이블의 행과 직접 일치하지 않는 행을 한 테이블에서 출력할 수 있다.

### Syntax

```
SELECT table.column, table.column...  
FROM table1, table2...  
WHERE table1.column = table2.column (+)
```

```
SELECT table.column, table.column...  
FROM table1, table2...  
WHERE table1.column(+) = table2.column
```

where	<i>table1.column = table2.column</i>	테이블을 함께 결합하는 조건.
	<i>(+)</i>	외부 결합 기호. <b>WHERE</b> 절 조건 아무쪽에 배치할 수 있지만, 동시에는 안된다.

## DISPLAY DATA WITHOUT A DIRECT MATCH

모든 고객의 영업 담당자 이름과 ID 및 고객 이름을 출력한다. 고객에게 영업 담당자가 할당되지 않는 경우에도 고객 이름을 포함한다.

### Example

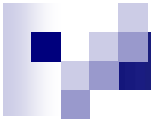
```
SQL> COLUMN last_name HEADING 'Sales Rep Name' FORMAT A15
SQL> COLUMN id HEADING 'Sales Rep ID'
SQL> COLUMN name HEADING 'Customer Name' FORMAT A30
SQL> SELECT e_emp.last_name, e_emp.id, e_customer.name
  2 FROM e_emp, e_customer
  3 WHERE e_emp.id(+) = e_customer.sales_rep_id
  4 ORDER BY e_emp.id;
```





## DISPLAY DATA WITHOUT A DIRECT MATCH

Sales Rep Name	Sales Rep ID	Customer Name
-----		
Magee	11	Womansport
Magee	11	Beisbol Si!
Magee	11	Big John's Sports Emporium
Magee	11	Ojibway Retail
Giljum	12	Unisports
Giljum	12	Futbol Sonora
Sedeghi	13	Hamada Sport
Nguyen	14	OJ Atheletics
Nguyen	14	Delhi Sports
Dumas	15	Kam's Sporting Goods
Dumas	15	Sportique
Dumas	15	Muench Sports
Dumas	15	Kuhn's Sports
Dumas	15	Sporta Russia
		Sweet Rock Sports
15 rows selected.		



## DISPLAY DATA WITHOUT A DIRECT MATCH

모든 고객과 해당 주문의 고객 ID, 고객 이름 및 주문 ID를 출력하라. 주문하지 않은 고객 ID와 이름도 출력하라.

### Example

```
SQL> SELECT e_customer.id "Customer ID",  
2 e_customer.name "Customer Name",  
3 e_ord.id "Order ID"  
4 FROM e_customer, e_ord  
5 WHERE e_customer.id = e_ord.customer_id(+);
```



## DISPLAY DATA WITHOUT A DIRECT MATCH

Customer ID	Customer Name	Order ID
201	Unisports	97
202	OJ Athletics	98
203	Delhi Sports	99
204	Womansport	100
205	Kam's Sporting Goods	101
206	Sportique	102
208	Muench Sports	103
208	Muench Sports	104
209	Beisbol Si!	105
210	Futbol Sonora	106
211	Kuhn's Sports	107
212	Hamada Sport	108
213	Big John's Sports Emporium	109
214	Ojibway Retail	110
204	Womansport	111
210	Futbol Sonora	112
207	Sweet Rock Sports	
215	Sporta Russia	

18 rows selected.

## CREATE A VIEW OF MULTIPLE TABLES

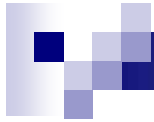
**CREATE VIEW** 문에 여러 테이블 쿼리를 포함시켜 여러 테이블의 **VIEW**를 생성한다.

모든 직원의 성, 부서 ID 및 부서 이름을 포함하는 **VIEW**를 생성하라

### Example

```
SQL> CREATE VIEW empdeptvu AS
  2 SELECT e_emp.last_name,
  3        e_emp.dept_id,
  4 e_dept.name
  5 FROM e_emp, e_dept
  6 WHERE e_emp.dept_id = e_dept.id;
```

View 'EMPDEPTVU' created.



# CREATE A VIEW OF MULTIPLE TABLES

## Example

Sales Rep Name	DEPT_ID	Customer Name
Velasquez	50	Administration
Ngao	41	Operations
Nagayama	31	Sales
Quick-To-See	10	Finance
Ropeburn	50	Administration
Urguhart	41	Operations
Menchu	42	Operations
Biri	43	Operations
Catchpole	44	Operations
Havel	45	Operations
...		

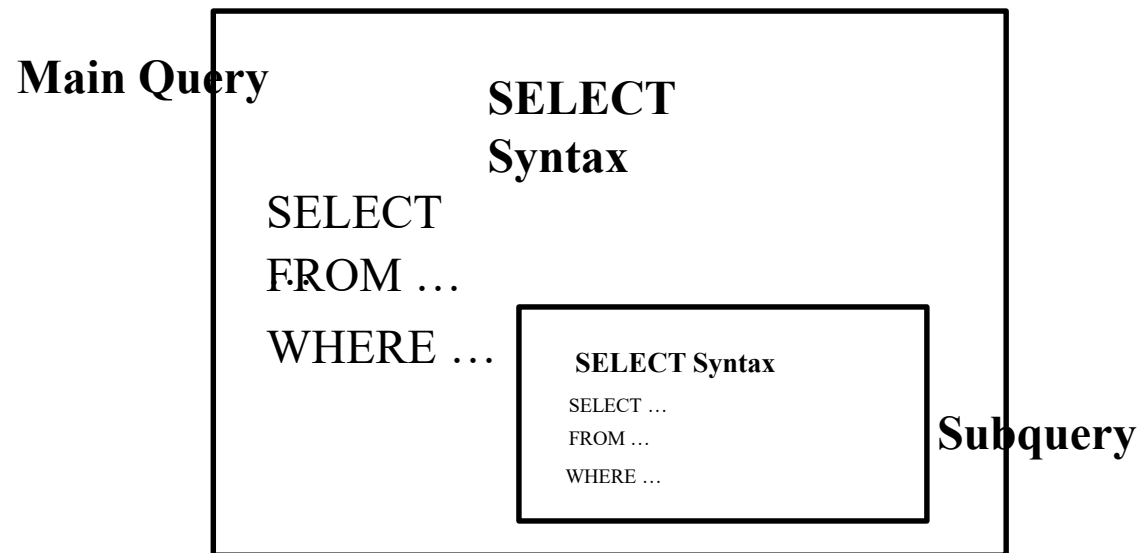
25 rows selected.



PASS VALUES BETWEEN QUERIES

## PASS VALUES BETWEEN QUERIES

- 하위 쿼리를 작성하여 다른 쿼리의 결과에 따라 달라지는 쿼리와 함께 데이터를 출력한다.



## NEST QUERIES THE RETURN ONE ROW

- SELECT 문을 중첩하여 하위 쿼리에서 반환된 단일 행에 따라 기본 쿼리로 표시된 행을 제한한다.
- 예를 들어 Smith와 동일한 직무를 가진 직원을 Smith가 어떤 직무를 가지고 있는지 모른채로 출력한다.

### Syntax

```
SELECT column_name, [, column_name]
FROM table_name
WHERE expression operator
      (SELECT column_name, [, column_name]
      FROM table_name)
```

where	<i>operator</i>	>, =, <, IN 과 같은 비교 연산자
-------	-----------------	-------------------------



## NEST QUERIES THE RETURN ONE ROW

Smith와 동일한 직무를 가진 직원의 성을 출력하라.

### Example

```
SQL> SELECT last_name, title
      FROM e_emp
      WHERE title =
        (SELECT title
         FROM e_emp
         WHERE last_name = 'Smith');
```

Sales Rep Name	TITLE
Maduro	Stock Clerk
Smith	Stock Clerk
Nozaki	Stock Clerk
Patel	Stock Clerk
Newman	Stock Clerk
Makarian	Stock Clerk
Chang	Stock Clerk
Patel	Stock Clerk
Dancs	Stock Clerk
Schwartz	Stock Clerk

## NEST QUERIES THE RETURN ONE ROW

Biri와 같은 부서에 있는 모든 직원의 성, 직무, 부서 ID를 출력하라.

### Example

```
SQL> SELECT last_name, title, dept_id
2 FROM e_emp
3 WHERE dept_id = (SELECT dept_id
4                 FROM e_emp
5                 WHERE UPPER(last_name) = 'BIRI');
```

Sales Rep Name	TITLE	DEPT_ID
Biri	Warehouse Manager	43
Newman	Stock Clerk	43
Makarian	Stock Clerk	43

3 rows selected.

## NEST QUERIES THE RETURN MULTIPLE ROW

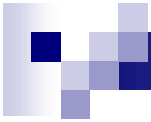
지역 2에 있는 부서에서 일하는 직원의 직원 ID, 성, 급여, 부서 ID를 출력하라

### Example

```
SQL> SELECT id, last_name salary, dept_id
2 FROM e_emp
3 WHERE dept_id IN
4     (SELECT id
5      FROM e_dept
6      WHERE region_id = 2);
```

Sales Rep ID SALARY	DEPT_ID
7 Menchu	42
12 Giljum	32
18 Nozaki	42
19 Patel	42

4 rows selected.



## NEST QUERIES THE RETURN MULTIPLE ROW

다중 행 하위 쿼리는 IN과 같은 다중 행 연산자를 사용해야 한다.

Patel이라는 직원과 함께 부서에서 일하는 모든 직원의 직원 ID, 성, 부서 ID를 출력하라

### Example

```
SQL> SELECT id, last_name, dept_id
2 FROM e_emp
3 WHERE dept_id = (SELECT dept_id
4                  FROM e_emp
5                  WHERE UPPER(last_name) = 'PATEL');
```

TBR-11002: Subquery returned multiple rows where a scalar value was expected.

## NEST MULTIPLE QUERIES

영업부(Sales) 또는 지역 2에 있는 모든 직원의 직원 ID, 성, 급여 및 부서 ID 를 출력하라

### Example

```
SQL> SELECT id, last_name, dept_id
2 FROM e_emp
3 WHERE dept_id IN (SELECT id
4                   FROM e_dept
5                   WHERE name = 'Sales')
6 OR dept_id IN (SELECT id
7               FROM e_dept
8               WHERE region_id = 2);
```

Sales Rep ID	Sales Rep Name	DEPT_ID
3	Nagayama	31
7	Menchu	42
11	Magee	31
12	Giljum	32
13	Sedeghi	33
14	Nguyen	34
15	Dumas	35
18	Nozaki	42
19	Patel	42
23	Patel	34

10 rows selected.



# TABA\_DB/SQL실습2



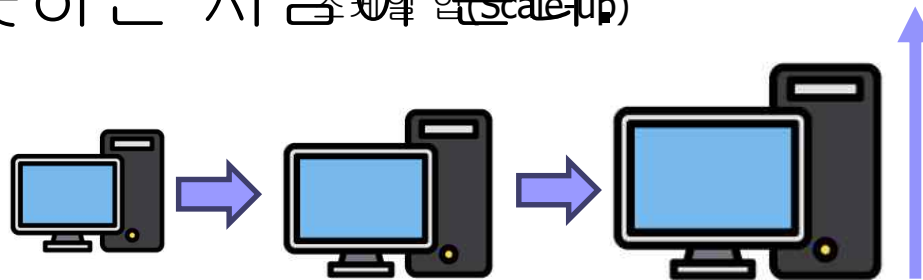
# 전통적 데이터 처리 시스템

- 전통적인 데이터 처리 시스템
  - 과거에는 데이터 처리를 위한 시스템 확장이 어려웠음
  - 전통적인 방식에서는 한 대의 컴퓨터의 처리 능력에 제한

# 전통적 데이터 처리 시스템

## ■ 스케일 업

- 시스템을 확장할 때 구조를 바꿀 필요가 없다.
- 소프트웨어를 고사양 서버로 단순히 이관하는 방식으로 확장성을 높임 (생각보다 어려움)
- 언젠가는 스케일 업 방식으로 더 이상 확장하지 못하는 시점이 온다

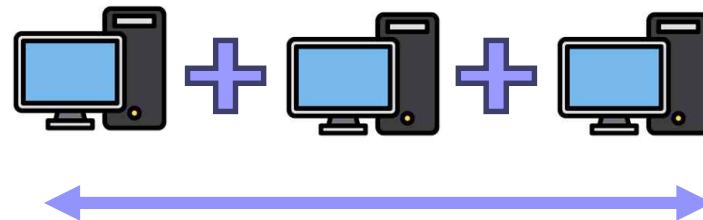




# 전통적 데이터 처리 시스템

## ■ 스케일 아웃

- 여러 대의 장비에 처리를 분산 시키는 방식
- 스케일 업 방식에 비해 비용이 저렴
- 서버들에 데이터 처리 방법을 개발해야 했음.
- 스케줄링, 장애처리 고려 등  
스케일 아웃(Scale-out)





# 전통적 데이터 처리 시스템

## ■ 한계점

- 큰 기업, 정부 기관, 학계를 제외한 곳에서 전통적인 스케일 업 및 스케일 아웃 방식을 그다지 사용하지 않았음.
- 스케일 업 : 비용이 비쌌음.
- 스케일 아웃 : 시스템 개발 및 관리가 어려움
- 여러 대의 호스트나 여러 개의 CPU 성능을 효과적으로 활용하기 어려움.
- 원하는 만큼 작업량을 효율적 처리하기 위한 전략은 엄청난 노력을 요함.



# 전통적 데이터 처리 시스템

## ■ 스케일 업의 한계

- 데이터양을 늘어나지만 하드웨어는 한계가 있음
- 고사양 서버를 한 대가 아닌 두 대, 세 대 ?
- 하이브리드 아키텍처는 하드웨어 구입 비용 및 클러스터 관리를 위한 로직 개발 두 가지 모두 필요.
- 빅데이터 처리 업계에서는 스케일 아웃 방식이 사실상 표준



# 전통적 데이터 처리 시스템

## ■ 스케일 업의 한계

- 많은 작업을 병렬 처리하는 하드웨어를 유연하게 사용하기 위해서는 소프트웨어를 똑똑하고, 하드웨어는 단순하게
- 하드웨어는 리소스 셋으로만 이용
- 소프트웨어가 처리 작업들에 하드웨어를 할당하는 역할



# 솔루션

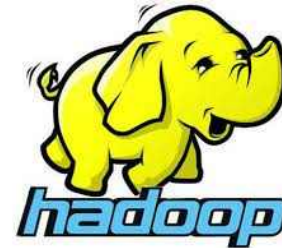
## ■ 장애 예측

- 서버가 늘어나도 각 서버의 장애나 문제점은 영향을 주지 않게 해야함.
- 각각의 장비는 주기적으로 장애가 발생할 수 있다는 점을 고려

# 하둡(Hadoop)

## ■ 하둡

- 2003년, 2004년 구글 내부의 기술을 설명하는
- 구글 파일시스템(GFS)과 맵리듀스(MapReduce)
- 더그 커팅은 구글의 GFS와 맵리듀스 논문에서 영감을 받아 시스템을 구현
- 하둡은 아파치 오픈소스 재단의 최상위 레벨 프로젝트



출처 : <https://post.naver.com/viewer/postView.nhn?volumeNo=28925185&memberNo=50533718>



# 하둡(Hadoop)

## ■ 하둡 구성요소

### □ 하둡 분산 파일 시스템(HDFS, Hadoop Distributed File System)

- 클러스터에 스케일 아웃 방식으로 대용량 데이터 셋을 저장하는 파일시스템
- 지연율보다 처리율에 최적화 / 이중화가 아닌 데이터 복제를 통해 고가용성을 얻음

### □ 맵리듀스(MapReduce)

- 대용량 데이터를 병렬 처리 하기 위해 개발된 프로그래밍 모델
- 입력 데이터를 분산 처리하는 맵(Map) 함수 단계와 다시 하나의 결과물로 합치는 리듀스(Reduce) 함수 단계로 나눈다.



# 하둡(Hadoop)

## ■ 공통 구성 요소

□ HDFS와 맵리듀스는 아래와 같은 원칙을 지킨다.

- 저가 서버들로 구성된 클러스터에서 구동하도록 설계
- 서버를 추가함으로써 용량 및 성능을 확장하는 방식
- 장애 탐지 및 대응 메커니즘
- 사용자는 해결할 문제 자체만 집중하도록 시스템의 많은 부분을 드러내지 않는다.
- 물리적 시스템을 제어하는 소프트웨어 클러스터를 구성하는 아키텍처





# 하둡(Hadoop)

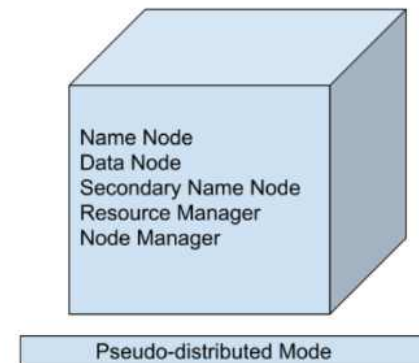
- 하둡의 세 가지 구동방식
  - 로컬 자립형 방식(Standalone Mode)
    - Hadoop이 실행되는 기본 모드
    - HDFS를 사용하지 않음
    - mapred-site.xml, core-site.xml, hdfs-site.xml 등 구성 파일을 수정할 필요 없음
    - 디버깅 목적으로 사용

# 하둡(Hadoop)

- 하둡의 세 가지 구동방식

- 가분산 방식(Pseudo-distributed Mode)

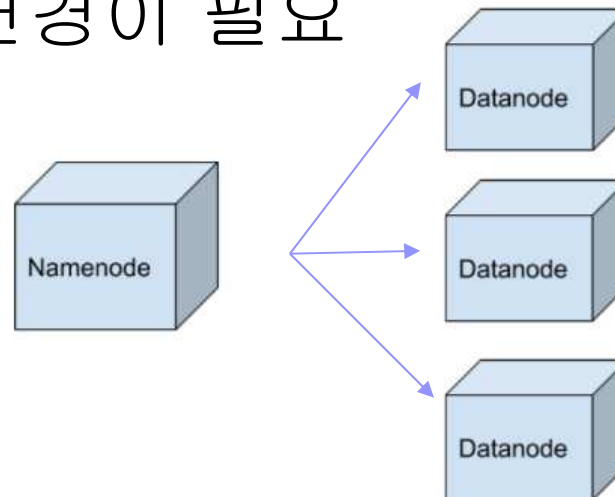
- Namenode와 Datanode가 모두 동일한 시스템에 있음
    - 최적화된 미니 클러스터를 효과적으로 생성
    - 구성 파일 변경이 필요



# 하둡(Hadoop)

## ■ 하둡의 세 가지 구동방식

- masternode와 datanode가 별도로 되어 있음
- 데이터가 여러 노드에 걸쳐 사용되고 분산
- 구성 파일 변경이 필요





# 하둡 설치

- 실습환경 VirtualBox Ubuntu
- (CPU 3, Memory 6144(6G), 50GB이상)
- AWS에서 실습 X
  
- JAVA 및 ssh설치
  - `sudo apt install -y openjdk-8-jdk openssh-server openssh-client`

# 하둡 설치

- 실습에 사용 할 하둡 계정 추가
  - `sudo adduser hdoop`
  - `su hdoop #hdoop` 계정으로 접속
  - hdoop 계정으로 접속 시 `pwd` 명령어를 통해 현재 디렉토리 위치를 확인 후 hdoop 계정이 작업할 수 있는 폴더로 이동

```
hdoop@dblab-VirtualBox:/home/dblab$ pwd
/home/dblab
```

hdoop 계정이 현재 있는 위치는 /home/dblab

```
hdoop@dblab-VirtualBox:/home/dblab$ cd /home/hdoop/
hdoop@dblab-VirtualBox:~$ pwd
/home/hdoop
```

작업할 수 있는 폴더로 이동해야함  
`cd /home/hdoop`



# 하둡(Hadoop)

- 하둡 다운로드

`wget https://downloads.apache.org/hadoop/common/hadoop-3.2.4/hadoop-3.2.4.tar.gz --no-check-certificate`

- 하둡 압축 풀기

`tar -xzvf hadoop-3.2.4.tar.gz`

# 하둡(Hadoop) cont.

- 환경변수 설정(~/.bashrc 파일 설정)

vi ~/.bashrc (제일 하단에 작성)

- ##JAVA\_HOME 위치 찾는 법
- readlink -f /usr/bin/javac

```
hadoop@ip-172-31-2-77:~/hadoop-3.2.4$ readlink -f /usr/bin/javac  
/usr/lib/jvm/java-8-openjdk-amd64/bin/javac
```

자바의 위치는 실습자료와 다를 수 있으니 꼭 확인하고 설정

```
##JAVA_HOME##
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

# 하둡(Hadoop)

```
## HADOOP_HOME ##  
export HADOOP_HOME=/home/hdoop/hadoop-3.2.4  
export HADOOP_INSTALL=$HADOOP_HOME  
export HADOOP_MAPRED_HOME=$HADOOP_HOME  
export HADOOP_COMMON_HOME=$HADOOP_HOME  
export HADOOP_HDFS_HOME=$HADOOP_HOME  
export YARN_HOME=$HADOOP_HOME  
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native  
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin:$JAVA_HOME/bin  
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

:wq로 저장 후

source ~/.bashrc #환경변수 적용

echo \$HADOOP\_HOME #적용 확인

```
##JAVA_HOME##  
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/  
  
##HADOOP_HOME##  
export HADOOP_HOME=/home/hdoop/hadoop-3.2.4  
export HADOOP_INSTALL=$HADOOP_HOME  
export HADOOP_MAPRED_HOME=$HADOOP_HOME  
export HADOOP_COMMON_HOME=$HADOOP_HOME  
export HADOOP_HDFS_HOME=$HADOOP_HOME  
export YARN_HOME=$HADOOP_HOME  
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native  
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin:$JAVA_HOME/bin
```



# 하둡(Hadoop)

## ■ hadoop-env.sh 파일 설정

- vi \$HADOOP\_HOME/etc/hadoop/hadoop-env.sh
- #export JAVA\_HOME 주석 제거 후 JAVA\_HOME 입력

```
# Technically, the only required environment variable is JAVA_HOME.
# All others are optional. However, the defaults are probably not
# preferred. Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

# Location of Hadoop. By default, Hadoop will attempt to determine
# this location based upon its execution path.
```

# 하둡(Hadoop)

## ■ core-site.xml 설정

- HDFS와 하둡 핵심 property 정의

- vi \$HADOOP\_HOME/etc/hadoop/core-site.xml

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/hadoop/tmpdata</value>
  </property>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/hadoop/tmpdata</value>
  </property>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://127.0.0.1:9000</value>
  </property>
</configuration>
```

- mkdir /home/hadoop/tmpdata #디렉토리 생성

# 하둡(Hadoop)

## ■ hdfs-site.xml 설정

### □ namenode와 datanode 저장소 디렉토리 설정

```
<configuration>
  <property>
    <name>dfs.data.dir</name>
    <value>/home/hadoop/dfsdata/namenode</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>/home/hadoop/dfsdata/datanode</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

```
<configuration>
  <property>
    <name>dfs.data.dir</name>
    <value>/home/hadoop/dfsdata/namenode</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>/home/hadoop/dfsdata/datanode</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

# 하둡(Hadoop)

## ■ mapred-site.xml 설정

### □ 맵리듀스 파일 값을 정의

```
<configuration>  
  <property>  
    <name>mapreduce.framework.name</name>  
    <value>yarn</value>  
  </property>  
</configuration>
```

```
<configuration>  
  <property>  
    <name>mapreduce.framework.name</name>  
    <value>yarn</value>  
  </property>  
</configuration>
```

# 하둡(Hadoop)

## ■ yarn-site.xml 설정

### □ YARN에 관련된 세팅을 정의하는 파일

```
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>0.0.0.0</value>
</property>
<property>
  <name>yarn.resourcemanager.address</name>
  <value>0.0.0.0:8032</value>
</property>
```



# 하둡(Hadoop)

## ■ yarn-site.xml 설정

### □ YARN에 관련된 세팅을 정의하는 파일

```
<property>
  <name>yarn.web-proxy.address</name>
  <value>0.0.0.0:8089</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-whitelist</name>
  <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,
  HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
</property>
```

# 하둡(Hadoop)

## ■ yarn-site.xml 설정

```
<configuration>
<!-- Site specific YARN configuration properties -->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>0.0.0.0</value>
  </property>
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>0.0.0.0:8032</value>
  </property>
  <property>
    <name>yarn.web-proxy.address</name>
    <value>0.0.0.0:8089</value>
  </property>
  <property>
    <name>yarn.acl.enable</name>
    <value>0</value>
  </property>
  <property>
    <name>yarn.nodemanager.env-whitelist</name>
    <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
  </property>
</configuration>
```



# 하둡(Hadoop)

- SSH-KEYGEN

- 하둡은 각 노드들이 통신을 해야하기 때문에 패스워드 없이 접속하도록 ssh-keygen 설정

```
ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa  
cat ~/.ssh/id_rsa.pub >>  
~/.ssh/authorized_keys  
chmod 0600 ~/.ssh/authorized_keys
```



# 하둡(Hadoop)

## ■ 네임노드 포맷

- hadoop을 가동하기 전 HDFS 파일시스템을 포맷해야함

hdfs namenode -format

```
hadoop@ip-172-31-2-77:~/hadoop-3.2.4$ hdfs namenode -format
WARNING: /home/hadoop/hadoop-3.2.4/logs does not exist. Creating.
2024-03-19 02:58:58,941 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = ip-172-31-2-77/172.31.2.77
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.2.4
STARTUP_MSG:   classpath = /home/hadoop/hadoop-3.2.4/etc/hadoop:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/commons-collections-3.2.2.jar:/home/
/hadoop/hadoop-3.2.4/share/hadoop/common/lib/kerb-admin-1.0.1.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jsr311-api-1.1.1.jar:/home/hadoop/h
hadoop/hadoop-3.2.4/share/hadoop/common/lib/commons-math3-3.1.1.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/paranamer-2.3.jar:/home/hadoop/h
hadoop-3.2.4/share/hadoop/common/lib/gson-2.9.0.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jsr305-3.0.2.jar:/home/hadoop/hadoop-3.2.4/share
2.4/share/hadoop/common/lib/kerb-common-1.0.1.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jetty-server-9.4.43.v20210629.jar:/home/hadoop/had
/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jetty-io-9.4.43.v20210629.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/httpcore-4.4.13.jar:/home
ome/hadoop/hadoop-3.2.4/share/hadoop/common/lib/animal-sniffer-annotations-1.17.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jersey-server-1
.5.2.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/kerb-core-1.0.1.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jackson-databind-2.14
a-2.5.0.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jaxb-api-2.2.11.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jaxb-impl-2.2.3-1
```

# 하둡(Hadoop)

## ■ 하둡 실행하기

- start-dfs.sh
- start-yarn.sh
- jps

```
hadoop@ip-172-31-2-77:~/hadoop-3.2.4$ jps
5968 ResourceManager
5763 SecondaryNameNode
5459 NameNode
6262 WebAppProxyServer
6088 NodeManager
5581 DataNode
6509 Jps
```

# 하둡 예제 (Wordcount)

## ■ 하둡 명령어

hdfs dfs [GENERIC\_OPTIONS]  
[COMMAND\_OPTIONS]

- cat -파일 내용 출력

hdfs dfs -cat

- cp -hdfs 내부에서 파일을 복사

hdfs dfs -cp

- mkdir -특정 path에 폴더 생성

hdfs dfs -mkdir

- mv -hdfs 내부에서 파일 옮기기

hdfs dfs -mv

- put -local에서 파일을 hdfs에 저장

hdfs dfs -input

- copyToLocal -hdfs에 있는 파일을 local에 다운  
hdfs dfs -copyToLocal

- du -hdfs 내부 특정 file이나 디렉토리의 사이즈  
를 보여줌


hdfs dfs -du

- ls -특정 디렉토리의 파일 혹은 디렉토리 출력

hdfs dfs -ls

- rm - hdfs에서 폴더 혹은 파일 삭제

hdfs dfs -rm



# 하둡 예제 (Wordcount)

- Wordcount에 사용될 예제 파일 다운

- wget <http://www.gutenberg.org/cache/epub/1661/pg1661.txt>

- hdfs에 폴더 생성하기

- hdfs dfs -mkdir -p /sample/input

- hdfs에 예제파일 이동하기

- hdfs dfs -put pg1661.txt /sample/input

# 하둡 예제 (Wordcount)

## ■ 워드카운트 실행하기

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.4.jar wordcount /sample/input /sample/output
```

## ■ 결과물 확인

```
hdfs dfs -cat /sample/output/part*
```

```
sundial 1
sundial, 1
sundial,' 1
sundial."" 1
sundial?' 1
sundials 1
sunk 8
sunk, 1
sunlight; 1
sunset, 1
sunshine. 1
superb 1
superior 1
superior, 1
superscribed 1
superscription 1
supper 4
supper, 2
supper." 1
supplementing 1
```

# 하둡 예제 (Wordcount)

## ■ Teragen / terasort

- 하둡 성능을 측정하기 위해 사용
- 일정 데이터를 생성하고 데이터를 정렬할 때 속도를 측정

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.4.jar  
teragen W
```

```
-Ddfs.replication=1 W
```

```
-Dmapred.map.tasks=36 W
```

```
-Dmapred.reduce.tasks=16 W
```

```
5000 /teragen/teragen.data
```

복사본 1개 맵 테스트 36 / 리듀스 테스트 16 / 생성 용량 5,000/ 생성된 데이터 저장할 곳

# 하둡 예제 (Wordcount)

## ■ Teragen

```
2024-03-24 18:07:40,871 INFO mapred.Task: Final Counters for attempt_local1133262668_0001_m_000000_0: Counters: 22
File System Counters
  FILE: Number of bytes read=316597
  FILE: Number of bytes written=866543
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=0
  HDFS: Number of bytes written=10737418200
  HDFS: Number of read operations=5
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=3
  HDFS: Number of bytes read erasure-coded=0
Map-Reduce Framework
  Map input records=107374182
  Map output records=107374182
  Input split bytes=83
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=173
  Total committed heap usage (bytes)=297271296
org.apache.hadoop.examples.terasort.TeraGen$Counters
  CHECKSUM=230593859918397906
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=10737418200
2024-03-24 18:07:40,871 INFO mapred.LocalJobRunner: Finishing task: attempt_local1133262668_0001_m_000000_0
2024-03-24 18:07:40,871 INFO mapred.LocalJobRunner: map task executor complete.
2024-03-24 18:07:40,935 INFO mapreduce.Job: map 100% reduce 0%
2024-03-24 18:07:40,935 INFO mapreduce.Job: Job job_local1133262668_0001 completed successfully
2024-03-24 18:07:40,940 INFO mapreduce.Job: Counters: 22
File System Counters
  FILE: Number of bytes read=316597
  FILE: Number of bytes written=866543
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=0
  HDFS: Number of bytes written=10737418200
  HDFS: Number of read operations=5
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=3
  HDFS: Number of bytes read erasure-coded=0
Map-Reduce Framework
  Map input records=107374182
  Map output records=107374182
  Input split bytes=83
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=173
  Total committed heap usage (bytes)=297271296
org.apache.hadoop.examples.terasort.TeraGen$Counters
  CHECKSUM=230593859918397906
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=10737418200
```

## ■ 생성된 파일 확인

```
oem@ubuntu-server:~$ hdfs dfs -ls /teragen
Found 2 items
-rw-r--r-- 1 oem supergroup          0 2024-03-24 18:07 /teragen/_SUCCESS
-rw-r--r-- 1 oem supergroup 10737418200 2024-03-24 18:07 /teragen/part-m-000000
```



# 하둡 예제 (Wordcount)

## ■ terasort

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.4.jar  
terasort /teragen/teragen.data /teragen/terasort.data
```



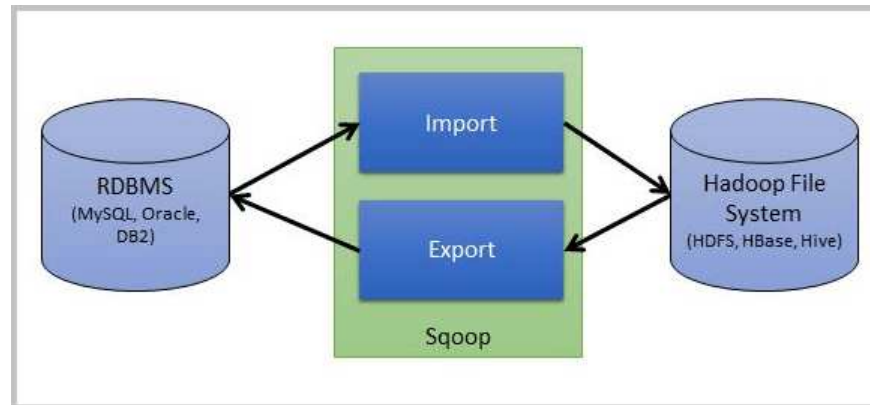
# 하둡 예제 (Wordcount)

```
2024-03-24 18:26:34,215 INFO mapreduce.Job: Counters: 36
  File System Counters
    FILE: Number of bytes read=485799799745
    FILE: Number of bytes written=938092152454
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=446412866700
    HDFS: Number of bytes written=10737418200
    HDFS: Number of read operations=8670
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=164
    HDFS: Number of bytes read erasure-coded=0
  Map-Reduce Framework
    Map input records=107374182
    Map output records=107374182
    Map output bytes=10952166564
    Map output materialized bytes=11166915408
    Input split bytes=9600
    Combine input records=0
    Combine output records=0
    Reduce input groups=107374182
    Reduce shuffle bytes=11166915408
    Reduce input records=107374182
    Reduce output records=107374182
    Spilled Records=322122546
    Shuffled Maps =80
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=2993
    Total committed heap usage (bytes)=147702939648
  Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
  File Input Format Counters
    Bytes Read=10737418200
  File Output Format Counters
    Bytes Written=10737418200
2024-03-24 18:26:34,215 INFO terasort.TeraSort: done
real    5m30.607s
user    5m15.090s
sys     0m44.482s
```

# Apache Sqoop

## ■ Apache Sqoop(SQL to Hadoop)

- Hadoop과 관계형 데이터베이스 간 데이터를 전송할 수 있도록 설계된 오픈소스 소프트웨어
- Oracle, MySQL등 RDBMS 특정 테이블, 데이터를 HDFS로 옮길 수 있음
- 반대로 HDFS에 저장된 데이터를 RDBMS로 옮길 수 있음
- Sqoop은 2009년 처음 버전이 나왔고, 현재 프로젝트는 종료됨



# Apache Sqoop

## ■ 스쿱 설치

cd /home/hdoop

wget [https://archive.apache.org/dist/sqoop/1.4.7/sqoop-1.4.7.bin\\_\\_hadoop-2.6.0.tar.gz](https://archive.apache.org/dist/sqoop/1.4.7/sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz)

tar -xzvf sqoop-1.4.7.bin\_\_hadoop-2.6.0.tar.gz #압축해제

mv sqoop-1.4.7.bin\_\_hadoop-2.6.0 sqoop #폴더명 변경

rm sqoop-1.4.7.bin\_\_hadoop-2.6.0.tar.gz #tar파일 삭제

vi ~/.bashrc #환경변수 추가

export SQOOP\_HOME=/home/hdoop/sqoop

```
export HADOOP_HOME=/home/hdoop/hadoop-3.0.0
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
export SQOOP_HOME=/home/hdoop/sqoop
```

# Apache Sqoop

## ■ sqoop 환경설정

cd /home/hadoop

wget [https://archive.apache.org/dist/sqoop/1.4.7/sqoop-1.4.7.bin\\_\\_hadoop-2.6.0.tar.gz](https://archive.apache.org/dist/sqoop/1.4.7/sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz)

tar -xzf sqoop-1.4.7.bin\_\_hadoop-2.6.0.tar.gz #압축해제

mv mv sqoop-1.4.7.bin\_\_hadoop-2.6.0 sqoop #폴더명 변경

rm sqoop-1.4.7.bin\_\_hadoop-2.6.0.tar.gz #tar파일 삭제

vi ~/.bashrc #환경변수 추가

export SQOOP\_HOME=/home/hadoop/sqoop

PATH에 Sqoop도 추가

export PATH=\$PATH:\$HADOOP\_HOME/sbin:\$HADOOP\_HOME/bin:\$SQOOP\_HOME/bin

source ~/.bashrc

```
## SPOOP ##
export SPOOP_HOME=/home/hadoop/sqoop

## HADOOP ##
export HADOOP_HOME=/home/hadoop/hadoop-3.1.4
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin:$SQOOP_HOME/bin
```

# Apache Sqoop

## ■ sqoop 환경설정

```
cp sqoop/conf/sqoop-env-template.sh sqoop/conf/sqoop-env.sh
```

```
vi sqoop/conf/sqoop-env.sh
```

```
export HADOOP_COMMON_HOME=$HADOOP_HOME
```

```
export HADOOP_MAPRED_HOME=$HADOOP_HOME
```

```
#Set path to where bin/hadoop is available
export HADOOP_COMMON_HOME=$HADOOP_HOME

#Set path to where hadoop-*.core.jar is available
export HADOOP_MAPRED_HOME=$HADOOP_HOME

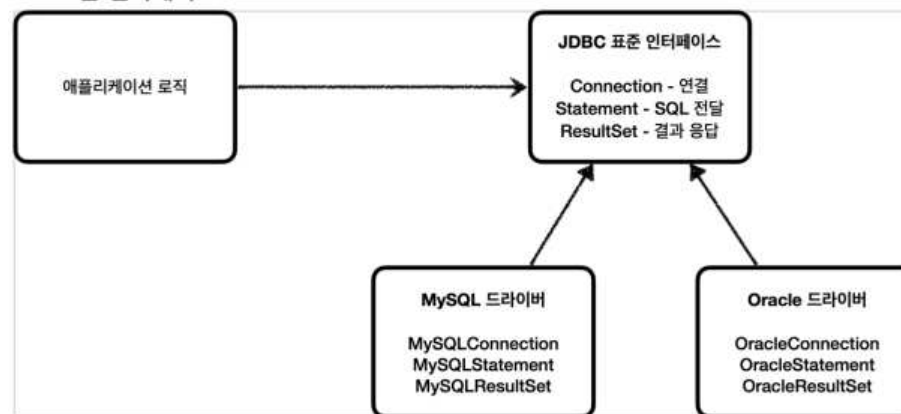
#set the path to where bin/hbase is available
#export HBASE_HOME=

#Set the path to where bin/hive is available
#export HIVE_HOME=

#Set the path for where zookeeper config dir is
#export ZOOKEEPER=
```

# Sqoop 예제

- MySQL JDBC 다운
- JDBC(Java Database Connectivity) 는 Java기반 애플리케이션의 애플리케이션의 데이터를 데이터베이스에서 저장 및 업데이트하거나, 데이터베이스에 저장된 데이터를 Java에서 사용할 수 있도록 하는 자바 API





# Sqoop 예제

- MySQL JDBC 다운 및 Sqoop lib 디렉토리에 옮기기

wget <https://downloads.mysql.com/archives/get/p/3/file/mysql-connector-j-8.0.33.tar.gz>

tar xzvf mysql-connector-j-8-0.33.tar.gz

cp mysql-connector-j-8.0.33/mysql-connector-j-8.0.33.jar \$SQOOP\_HOME/lib/

- StringUtils 관련 오류 예방으로 commons 라이브러리 설치

wget <https://dlcdn.apache.org/commons/lang/binaries/commons-lang-2.6-bin.tar.gz>

tar zxvf commons-lang-2.6-bin.tar.gz

cp commons-lang-2.6/commons-lang-2.6.jar \$SQOOP\_HOME/lib

# Sqoop 예제

hadoop 계정에 sudo 권한주기

1. 기본 계정으로 접속
2. sudo passwd
3. su

```
ubuntu@ubuntu-VirtualBox:~$ sudo passwd
[sudo] password for ubuntu:
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
ubuntu@ubuntu-VirtualBox:~$ su
Password:
root@ubuntu-VirtualBox:/home/ubuntu# exit
```



# Sqoop 예제

hadoop 계정에 sudo 권한주기

sudoers 파일은 readonly 파일이라 일반적인 파일과 수정 방법이 다름.

vi /etc/sudoers

root 계정 아래 내용 추가

hadoop ALL=(ALL:ALL) ALL

w!

q!

exit 로 root 빠져나온 뒤 su hadoop 계정 접속

```
# Cnmd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
hadoop  ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

# Sqoop 예제

## ■ MySQL 설치

`sudo apt install -y mysql-server`

`sudo systemctl status mysql.service`

```
hdoop@ubuntu-VirtualBox:/home/ubuntu$ sudo systemctl status mysql.service
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2024-03-26 02:38:00 KST; 33s ago
     Process: 29465 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
    Main PID: 29473 (mysqld)
      Status: "Server is operational"
        Tasks: 38 (limit: 2261)
       Memory: 352.5M
          CPU: 1.571s
      CGroup: /system.slice/mysql.service
              └─29473 /usr/sbin/mysqld
```

# Sqoop 예제

- MySQL 패스워드 설정
- 초기 mysql 접속 시 패스워드 없이 접속 가능

`mysql -u root -p`

패스워드 설정 (대소문자, 숫자, 특수기호, 8자리 이상 들어가야함.)

`ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Dankook1!';`

```
hdoop@ubuntu-VirtualBox:/home/ubuntu$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Dankook1!';
Query OK, 0 rows affected (0.02 sec)

mysql>
```

# Sqoop 예제

- MySQL Test 데이터베이스 다운 및 압축풀기

wget [https://github.com/datacharmer/test\\_db/releases/download/v1.0.7/test\\_db-1.0.7.tar.gz](https://github.com/datacharmer/test_db/releases/download/v1.0.7/test_db-1.0.7.tar.gz)

tar -xvzf test\_db-1.0.7.tar.gz

- MySQL 에 데이터 로드

cd test\_db/

mysql -u root -p < employees.sql

```
hadoop@ubuntu-VirtualBox:~/test_db$ mysql -u root -p < employees.sql
Enter password:
INFO
CREATING DATABASE STRUCTURE
INFO
storage engine: InnoDB
INFO
LOADING departments
INFO
LOADING employees
INFO
LOADING dept_emp
INFO
LOADING dept_manager
INFO
LOADING titles
INFO
LOADING salaries
data_load_time_diff
00:01:18
```

# Sqoop 예제

- 테스트 데이터베이스 조회하기

```
mysql -u root -p  
show databases;  
use employees;  
show tables;
```

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| employees |  
| information_schema |  
| mysql |  
| performance_schema |  
| sys |  
+-----+  
5 rows in set (0.05 sec)
```

```
mysql> show tables;  
+-----+  
| Tables_in_employees |  
+-----+  
| current_dept_emp |  
| departments |  
| dept_emp |  
| dept_emp_latest_date |  
| dept_manager |  
| employees |  
| salaries |  
| titles |  
+-----+  
8 rows in set (0.01 sec)
```



# Sqoop 예제

- sqoop으로 MySQL 특정 데이터베이스의 테이블 조회하기  
sqoop list-tables --connect jdbc:mysql://localhost/employees ₩  
--username root ₩  
--password Dankook1!
- sqoop으로 MySQL -> HDFS 데이터 보내기  
sqoop import --connect jdbc:mysql://localhost/employees ₩  
--username root --password Dankook1! --table employees -m 1

# Sqoop 예제

```
2024-03-26 03:41:51,677 INFO mapreduce.Job: Job job_1711383692141_0001 running in uber mode : false
2024-03-26 03:41:51,689 INFO mapreduce.Job: map 0% reduce 0%
2024-03-26 03:42:12,855 INFO mapreduce.Job: map 100% reduce 0%
2024-03-26 03:42:17,590 INFO mapreduce.Job: Job job_1711383692141_0001 completed successfully
2024-03-26 03:42:18,984 INFO mapreduce.Job: Counters: 33
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=246278
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=87
    HDFS: Number of bytes written=13821993
    HDFS: Number of read operations=6
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
    HDFS: Number of bytes read erasure-coded=0
  Job Counters
    Launched map tasks=1
    Other local map tasks=1
    Total time spent by all maps in occupied slots (ms)=18891
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=18891
    Total vcore-milliseconds taken by all map tasks=18891
    Total megabyte-milliseconds taken by all map tasks=19344384
  Map-Reduce Framework
    Map input records=300024
    Map output records=300024
    Input split bytes=87
    Spilled Records=0
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=242
    CPU time spent (ms)=5220
    Physical memory (bytes) snapshot=139124736
    Virtual memory (bytes) snapshot=2494951424
    Total committed heap usage (bytes)=32571392
    Peak Map Physical memory (bytes)=139124736
    Peak Map Virtual memory (bytes)=2494951424
  File Input Format Counters
    Bytes Read=0
  File Output Format Counters
    Bytes Written=13821993
2024-03-26 03:42:19,059 INFO mapreduce.ImportJobBase: Transferred 13.1817 MB in 155.7499 seconds (86.6648 KB/sec)
2024-03-26 03:42:19,072 INFO mapreduce.ImportJobBase: Retrieved 300024 records.
```

# Sqoop 예제

- localhost:8088



▼ Cluster

- [About](#)
- [Nodes](#)
- [Node Labels](#)
- [Applications](#)
  - NEW
  - NEW SAVING
  - SUBMITTED
  - ACCEPTED
  - RUNNING
  - FINISHED
  - FAILED
  - KILLED
- [Scheduler](#)

► Tools

## Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	...
1	0	0	1	0

## Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	...
1	0	0

## Scheduler Metrics

Scheduler Type	Scheduling Resource Type
Capacity Scheduler	[memory-mb (unit=M), vcores]

Show 20 ▼ entries

ID	User	Name	Application Type	Queue	Application Priority
<a href="#">application_1711383692141_0001</a>	hadoop	employees.jar	MAPREDUCE	default	0

Showing 1 to 1 of 1 entries





# Sqoop 예제

- 명령어로 하둡 데이터 조회하기
- `hdfs dfs -ls`

```
hdoop@ubuntu-VirtualBox:~$ hdfs dfs -ls
Found 1 items
drwxr-xr-x  - hdoop supergroup          0 2024-03-26 03:42 employees
```



# TABA\_DB/SQL실습3

# 쿠버네티스란?



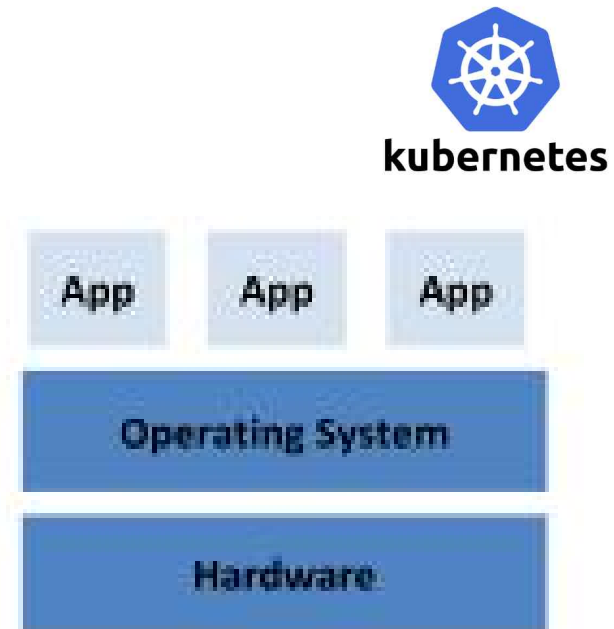
kubernetes

- 컨테이너를 쉽고 빠르게 배포, 확장 및 관리를 자동화해주는 오픈소스 플랫폼
- 명칭은 키잡이(helmsman)나 파일럿을 뜻하는 그리스어에서 유래
- K8s라는 표기는 "K"와 "s" 사이 8글자를 나타내는 약식 표기
- 구글이 2014년 쿠버네티스 프로젝트를 오픈소스화 함
- 관리자가 서버를 배포할 때 원하는 상태를 선언하는 방식 사용(Desired State)

# 쿠버네티스란?

## ■ 전통적인 배포 시대(Traditional Deployment)

- 초기 조직은 애플리케이션을 물리 서버에서 실행
- 리소스 할당 문제 발생
- 여러 물리 서버에서 각 애플리케이션을 실행
- 리소스가 충분히 활용되지 않음.
- 물리 서버를 유지하는 데에 높은 비용



Traditional Deployment



kubernetes

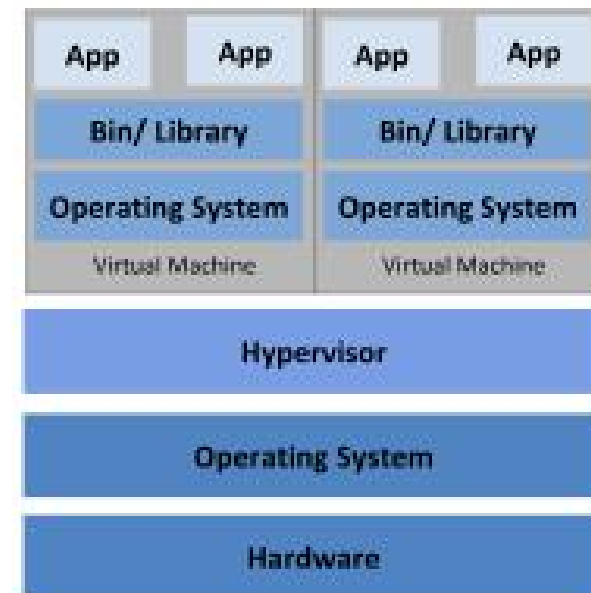
# 쿠버네티스란?



kubernetes

## ■ 가상화된 배포 시대(Virtualized Deployment)

- 전통적인 배포 시대의 해결책으로 가상화 도입
- 단일 물리서버의 **CPU**에서 여러 가상 머신 실행
- 가상화를 사용하면 **VM**간에 격리로 보안 제공
- 물리서버에서 리소스를 효율적으로 사용

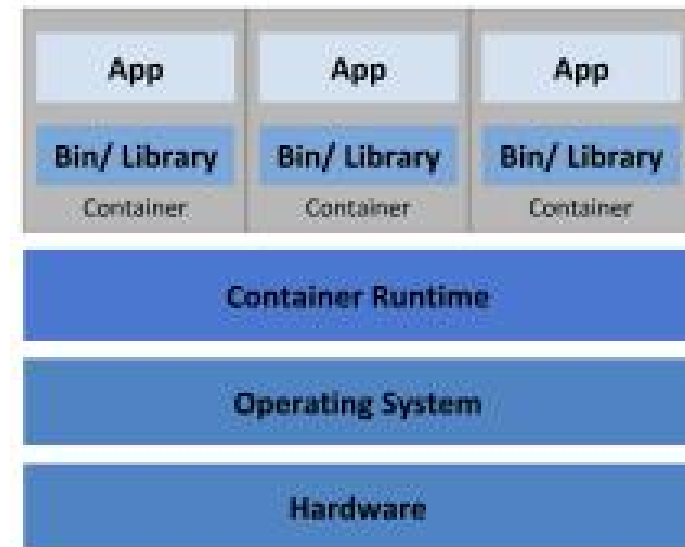


Virtualized Deployment

# 쿠버네티스란?



- 컨테이너 개발 시대(Container Deployment)
  - VM과 유사하지만 격리 속성 완화
  - 애플리케이션 간 OS 공유
  - VM이미지를 사용하는 것보다 컨테이너 이미지 생성이 쉽고 효율적
  - 클라우드 및 OS간 이식성(Ubuntu, RHEL, 퍼블릭 클라우드)



Container Deployment

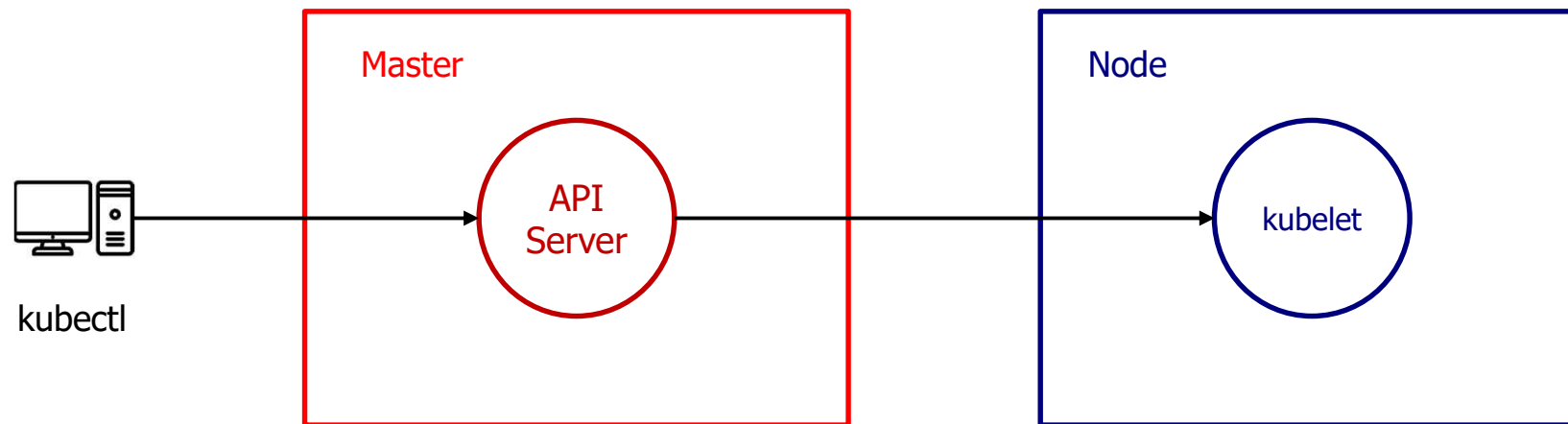


# 쿠버네티스란?

- 컨테이너 오케스트레이션
  - 컨테이너 기반 애플리케이션 배포 관리, 제어 및 모니터링, 스케일링, 네트워킹 관리 도구
- 컨테이너 오케스트레이션 종류
  - 도커 스웜(Docker Swarm), 아파치 메소스(Apache Mesos), 노마드(Nomad)
  - 쿠버네티스가 컨테이너 기반 인프라 시장에서 사실상 표준

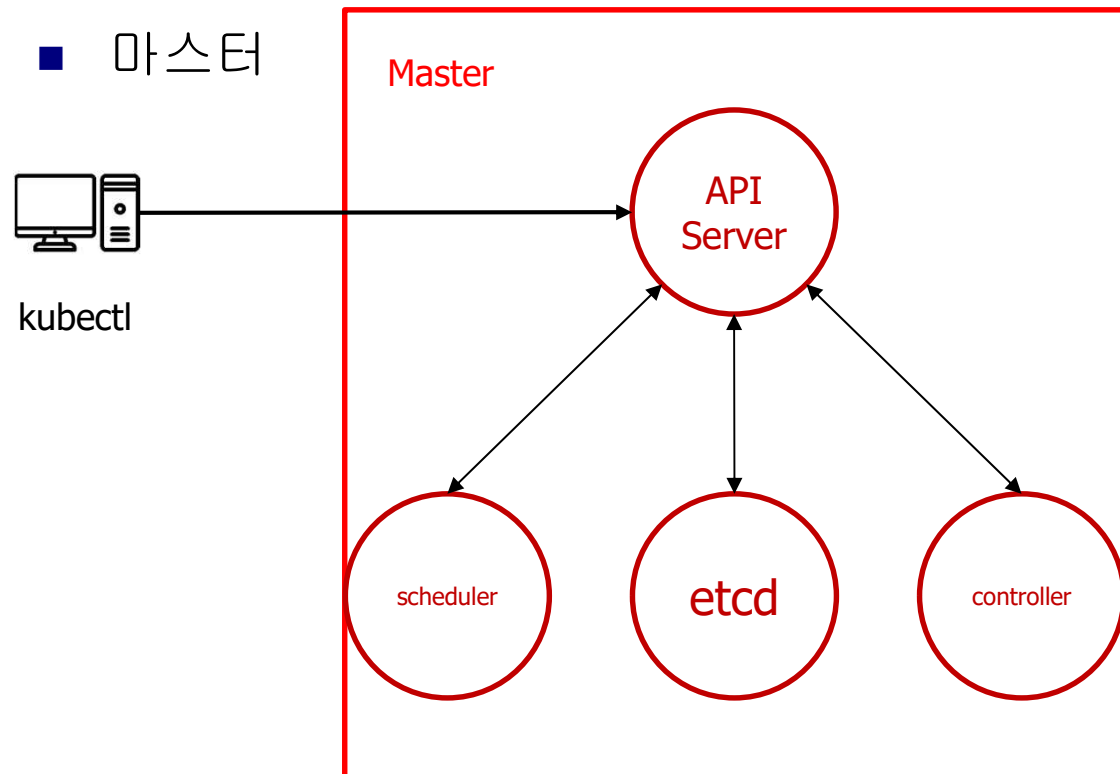
# 쿠버네티스 구조

## ■ 마스터 - 노드 구조



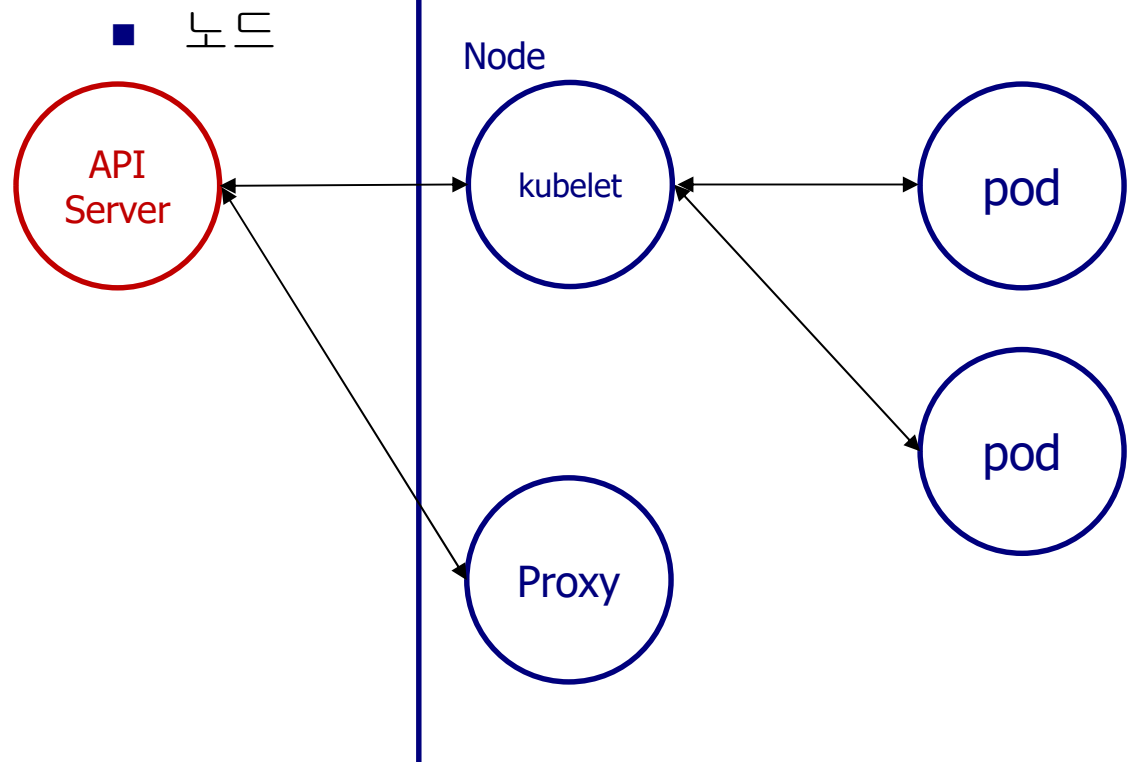


# 쿠버네티스 구조



- 핵심 컴포넌트
- Kubectl
  - 마스터의 API Server 통신
- API Server
  - 관리자 요청 및 내부 모듈과 통신
  - etcd와 유일하게 통신
- etcd
  - 모든 상태와 데이터를 저장
  - Key-Value 형태
- Scheduler
  - 새로 생성된 pod을 감지
- controller
  - 다양한 컨트롤러 존재
  - 복제, 노드, 엔드포인트 등

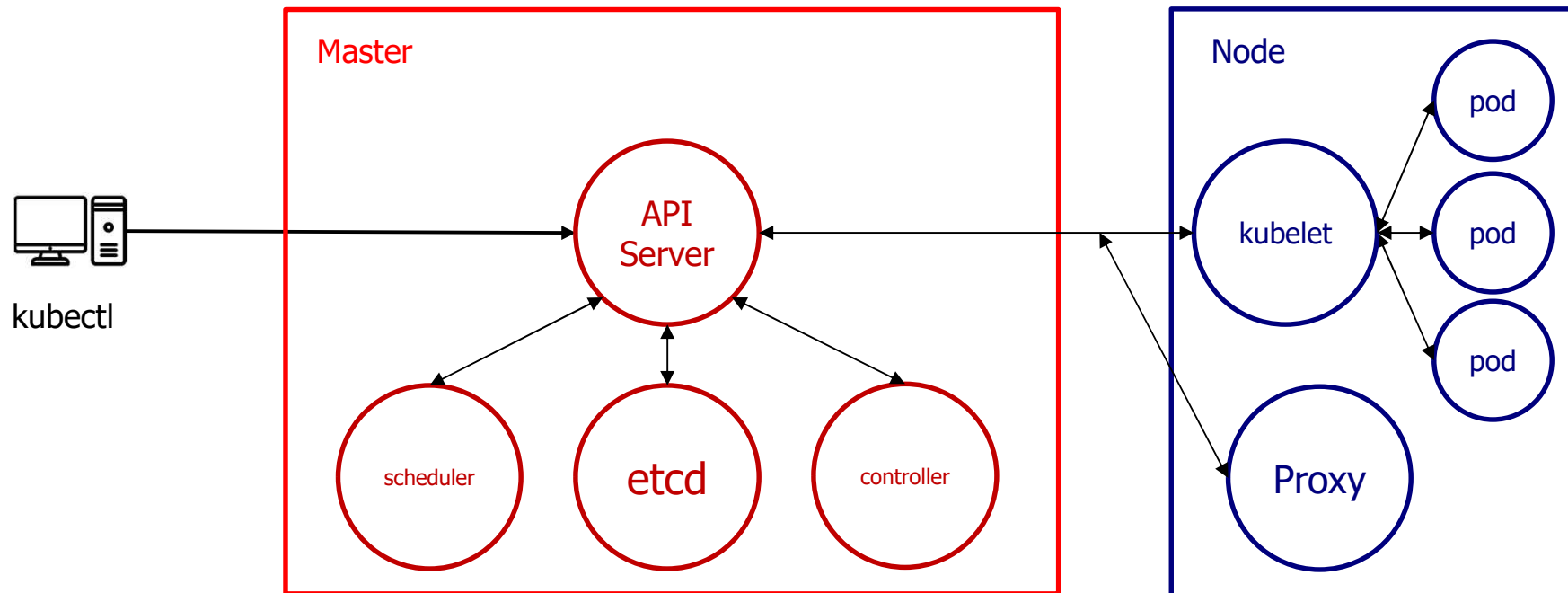
# 쿠버네티스 구조



- 핵심 컴포넌트
- kubelet
  - 각 노드에서 실행
  - pod 실행, 중지 및 상태 체크
  - CRI(Container Runtime Interface)
    - Docker, Containerd, CRI-O...
- proxy
  - 네트워크 프록시와 부하 분산 역할
  - 내/외부 통신 설정 등

# 쿠버네티스 구조

## ■ 쿠버네티스 프로세스





# minikube Install

## ■ minikube

- 쿠버네티스 클러스터를 설치하기 위해선 3대의 마스터 서버와  $n$ 개의 노드 서버가 필요.
- mac, Linux, Windows에서 K8s 클러스터를 빠르게 설정해주는 도구
- minikube를 이용하면 로컬에서 K8s 클러스터를 만들 수 있다.



# minikube Install

## ■ 업데이트 및 유틸리티 설치

```
sudo yum update -y && sudo yum install -y utils
```

## ■ Docker-ce 레포 추가

```
sudo yum config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

## ■ Docker 설치

```
sudo yum install -y docker-ce docker-ce-cli containerd.io
```



# minikube Install

- Docker 실행 및 확인

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

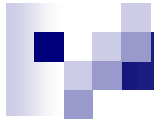
```
sudo systemctl status docker
```

- 도커 그룹에 유저 추가

```
sudo groupadd docker #(docker group 생성)
```

```
sudo usermod -aG docker $USER #(docker group 해당유저 추가)
```

```
newgrp docker #(적용하기)
```



# minikube Install

## ■ minikube

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64  
sudo install minikube-linux-amd64 /usr/local/bin/minikube  
sudo rm minikube-linux-amd64
```



# minikube Install

## ■ minikube

minikube start

```
spark@ubuntu-VirtualBox:~$ minikube start
* minikube v1.33.0 on Ubuntu 22.04 (vbox/amd64)
* Automatically selected the docker driver. Other choices: ssh, none
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.43 ...
* Downloading Kubernetes v1.30.0 preload ...
```

```
spark@ubuntu-VirtualBox:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```



# minikube Install

## ■ kubectl

`curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl`

```
spark@ubuntu-VirtualBox:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100  138    100  138    0     0   523      0 --:--:-- --:--:-- --:--:--   524
100 49.0M    100 49.0M    0     0 10.3M      0 0:00:04 0:00:04 --:--:-- 11.2M
```

`curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"`  
`echo "$(cat kubectl.sha256) kubectl" | sha256sum --check`

```
spark@ubuntu-VirtualBox:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100  138    100  138    0     0   560      0 --:--:-- --:--:-- --:--:--   560
100   64    100   64    0     0   202      0 --:--:-- --:--:-- --:--:--   202
spark@ubuntu-VirtualBox:~$ echo "$(cat kubectl.sha256) kubectl" | sha256sum --check
kubectl: OK
```



# minikube Install

## ■ kubectl

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

```
chmod +x kubectl
```

```
mkdir -p ~/.local/bin
```

```
mv ./kubectl ~/.local/bin/kubectl
```

```
kubectl version --client
```

```
spark@ubuntu-VirtualBox:~$ kubectl version --client  
Client Version: v1.30.0  
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```



# minikube Install

## ■ minikube 및 kubectl 명령어

minikube start : cluster 실행  
minikube delete : cluster 실행  
minikube stop / pause : 정지 / 일시정지  
minikube ip : 노드의 ip 확인  
minikube ssh : node 접속

kubectl cluster-info : cluster 설정 확인  
kubectl delete pod pod명 --grace-period=0 --force : pod 강제 삭제  
kubectl get events : event 모니터링  
kubectl describe pods/{pod명} or nodes/{node명} : 상세 보기



# 쿠버네티스 오브젝트

## ■ 파드(Pod)

- 쿠버네티스에서 생성하고 관리할 수 있는 배포 가능한 가장 작은 컴퓨팅 단위
- 하나 이상의 컨테이너 그룹
- **Pod**는 스토리지 및 네트워크를 공유하고, 구동하는 방식에 대한 명세를 갖는다.
- 컨테이너와 비슷한 개념이지만 완전히 같은 개념은 아님

# 쿠버네티스 오브젝트

- 파드 생성
- vim 편집기로 yaml작성

vi simple-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

```
kubectl run
- CLI로 pod를 생성
kubectl create
- yaml으로 pod 생성
- 동일한 pod가 있을 경우 에러 발생
kubectl apply
- yaml으로 pod 생성
- 동일한 pod가 없으면 새로운 pod 생성
- 동일한 pod가 있으면 기존 config와 비교해서 수정된 부분 업데이트
```

kubectl apply -f simple-pod.yaml #파드 생성

kubectl get po #생성된 파드 확인

kubectl delete -f simple-pod.yaml

```
spark@ubuntu-VirtualBox:~$ kubectl apply -f simple-pod.yaml
pod/nginx created
spark@ubuntu-VirtualBox:~$ kubectl get po nginx
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0           14s
spark@ubuntu-VirtualBox:~$ kubectl delete -f simple-pod.yaml
pod "nginx" deleted
```



# 쿠버네티스 오브젝트

## ■ 네임스페이스(Namespace)

- 쿠버네티스에서 사용되는 리소스들을 구분해 관리하는 그룹
- **Pod, Service** 등은 네임스페이스별로 생성 및 관리 가능, 접근 권한도 다르게 설정 가능
- 네임스페이스는 여러 개의 팀이나 프로젝트에 걸쳐 많은 사용자가 있는 환경에서 사용하도록 만들어짐
- 쿠버네티스는 **default, kube-node-lease, kube-public, kube-system**이라는 네 개의 네임스페이스를 갖고 있다.

# 쿠버네티스 오브젝트

- 네임스페이스 조회

kubectl get namespace or ns

```
spark@ubuntu-VirtualBox:~$ kubectl get ns
NAME                STATUS   AGE
default             Active   15h
kube-node-lease     Active   15h
kube-public         Active   15h
kube-system         Active   15h
```

- 새 네임스페이스 생성1

vi my-namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: news1
```

- kubectl create -f my-namespace.yaml

- 아래 명령어로 네임스페이스 생성 가능  
kubectl create namespace | ns news2

kubectl get ns

```
spark@ubuntu-VirtualBox:~$ kubectl create -f my-namespace.yaml
namespace/news1 created
spark@ubuntu-VirtualBox:~$ kubectl create namespace news2
namespace/news2 created
spark@ubuntu-VirtualBox:~$ kubectl get ns
NAME                STATUS   AGE
default             Active   15h
kube-node-lease     Active   15h
kube-public         Active   15h
kube-system         Active   15h
news1               Active   18s
news2               Active    5s
```

kubectl delete ns news2 #네임스페이스 삭제



# 쿠버네티스 오브젝트

## ■ 서비스(Service)

- 파드는 언제든지 다른 노드로 옮겨지거나 삭제될 수 있음.
- 생성될 때마다 새로운 IP를 받게 되는데, 내/외부 통신을 유지하기 어려움
- 쿠버네티스에서 실행되고 있는 파드를 네트워크에 노출시키는 가상의 컴포넌트
- 내/외부의 애플리케이션과 연결 혹은 사용자와 연결될 수 있도록 고정 IP를 갖는 서비스를 이용해 통신 가능





# 쿠버네티스 오브젝트

## ■ 서비스 타입

### □ Cluster IP

- 가장 기본이 되는 **Service** 타입
- 클러스터 내부 통신만 가능, 외부 트래픽 불가능

### □ NodePort

- 클러스터 내/외부 통신이 가능
- 외부 트래픽을 전달받을 수 있음
- 노드의 포트를 사용

### □ LoadBlancer

- 기본적으로 외부에 존재하며, 클라우드 프로바이더와 함께 사용되어 외부 트래픽을 받음

### □ ExternalName

- 위 3가지와 전혀 다른 서비스 타입
- 외부로 나가는 트래픽을 변환하기 위한 용도
- 도메인 이름을 변환하여 연결해주는 역할



# 쿠버네티스 오브젝트

## ■ 볼륨(Volume)

- 쿠버네티스에서 파드를 생성하면 디렉토리를 임시로 사용함
- 컨테이너가 삭제되면 파일 손실되는 문제 발생
- 파드가 사라지더라도 사용할 수 있는 디렉터리는 볼륨 오브젝트를 이용해 생성
- **Docker**의 볼륨과 비슷한 개념
- 쿠버네티스 버전에 따라 사용 가능한 볼륨이 있음
- 볼륨 종류
  - **emptyDir** : 일시적 데이터 저장, 파드가 삭제되면 스토리지도 삭제
  - **localPath**: 노드의 파일시스템을 파드로 마운트
  - **nfs**
  - **awsEBS, gcePersistentDisk, azureDisk** : 클라우드 전용 스토리지
  - **PV(Persistent Volumes)** : 영구볼륨, 포드가 종료되어도 데이터는 삭제되지 않음.
  - **PVC(Persistent Volumes Claim)** : 생성된 **PV**를 사용

# 쿠버네티스 오브젝트

## ■ PV yaml 작성

demoPV.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: demo-pv
spec:
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/pv/log"
  persistentVolumeReclaimPolicy: Retain
```

## ■ capacity : 볼륨크기

## ■ accessModes

- ReadWriteOnce : 하나의 노드에서 RW 가능
- ReadOnlyMany : 여러노드에서 R 가능
- ReadWriteMany : 여러 노드에서 RW가능

## ■ persistentVolumeReclaimPolicy

- PV 종료 시 볼륨에 저장된 데이터 삭제 옵션
- Retain : PVC가 삭제되어도 PV의 데이터 보존
- Delete : PVC가 삭제되면 데이터 및 PV도 삭제
- Recycle : PVC가 삭제되면 데이터만 삭제

# 쿠버네티스 오브젝트

- PVC yaml 작성

demoPVC.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: demo-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Mi
```

- capacity : 볼륨크기

- accessModes

- 사용하는 PV와 동일한 옵션을 사용해야함

- requests

- 사용을 원하는 스토리지 요구 명시
- storage : 사용하고자 하는 최소한의 크기

# 쿠버네티스 오브젝트

- PV, PVC 생성 및 조회

kubectl create -f demoPV.yaml

kubectl create -f dempPVC.yaml

kubectl get persistentvolume #pv

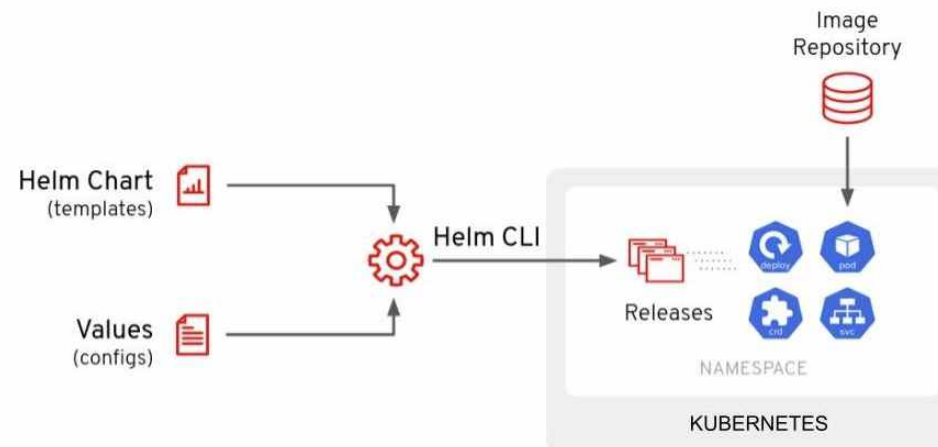
```
spark@ubuntu-VirtualBox:~$ kubectl create -f dempPV.yaml
persistentvolume/demo-pv created
spark@ubuntu-VirtualBox:~$ kubectl create -f dempPVC.yaml
persistentvolumeclaim/demo-pvc created
spark@ubuntu-VirtualBox:~$ kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM
demo-pv	100Mi	RWX	Retain	Available	
pvc-32e18ed2-a636-49e8-8a99-6c930e2a8257	2Gi	RWO	Delete	Bound	default/storage-prometheus-alertmanager-0
pvc-8cdb1b2e-13e0-4367-9f2d-03824f77c5eb	50Mi	RWX	Delete	Bound	default/demo-pvc

# Helm

## ■ Helm

- K8s 애플리케이션을 위한 오픈소스 패키지 매니저
- Helm 구성 요소
  - Chart : 애플리케이션을 배포하는데 사용되는 관련 Kubernetes YAML파일
  - Repository : 차트를 저장, 공유, 배포할 수 있는 곳
  - Release : Kubernetes 클러스터에 배포된 차트 특정 인스턴스





# Helm

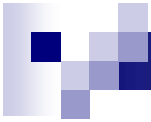
## ■ Helm 설치

```
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3  
chmod 700 get_helm.sh  
./get_helm.sh
```

## ■ 설치 확인

```
helm version
```

```
spark@ubuntu-VirtualBox:~$ helm version  
version.BuildInfo{Version:"v3.14.4", GitCommit:"81c902a123462fd4052bc5e9aa9c513c4c8fcl42", GitTreeState:"clean", GoVersion:"go1.21.9"}
```



# Helm

## ■ Helm 명령어

- `helm search hub` : 저장소에 있는 `helm`차트를 `helm hub`에서 검색
- `helm install chart명` : 특정 `chart`패키지 설치
- `helm show values chart명` : `chart` 옵션 설정가능한 `values` 조회
- `helm uninstall release명` : 릴리즈 삭제
- `helm repo list` : 저장된 저장소 목록 조회
- `helm repo add {name} {url}` : 저장소 저장
- `helm repo remove {name}` : 저장소 삭제





# Helm을 이용한 모니터링 시스템 만들기

## ■ 프로메테우스(Prometheus)

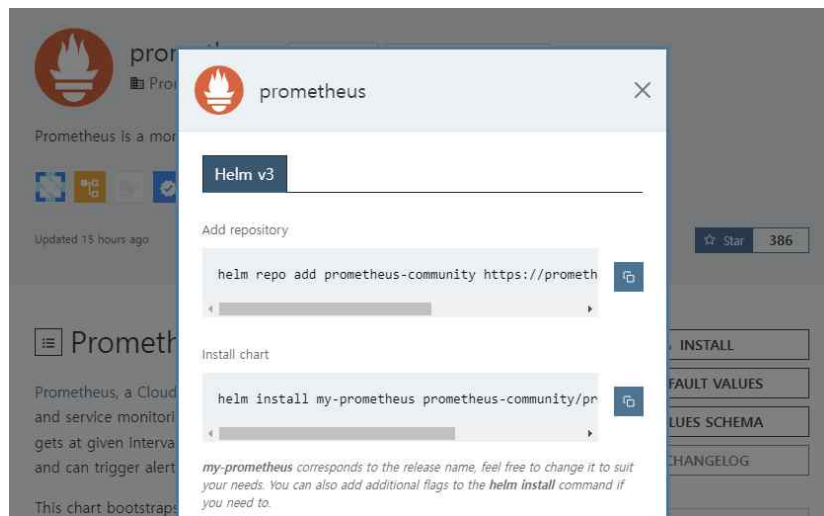
- SoundCloud에서 만든 오픈소스 모니터링 툴
- Kubernetes환경에서 모니터링하기 원하는 리소스로부터 **metric**을 수집하고 해당 **metric**을 이용해 모니터링

## ■ 그라파나(Grafana)

- 오픈소스 시각화 분석 도구
- 여러 데이터 소스에 대한 대시보드 템플릿 제공
- 프로메테우스도 **UI**를 제공하지만, 기능이 빈약해 그라파나와 연동 사용

# Helm을 이용한 모니터링 시스템 만들기

- 프로메테우스(Prometheus)
- <https://artifacthub.io/> 에서 Prometheus 검색 - Install - Add repo  
helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>  
helm repo update



# Helm을 이용한 모니터링 시스템 만들기

- 프로메테우스(Prometheus)
- helm install prometheus prometheus-community/prometheus
- kubectl get all

```
spark@ubuntu-VirtualBox:~/cache/helm/repository$ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/prometheus-alertmanager-0       1/1     Running   0           5m54s
pod/prometheus-kube-state-metrics-76fc9c6f55-znbn5  1/1     Running   0           5m54s
pod/prometheus-prometheus-node-exporter-xmmwj       1/1     Running   0           5m54s
pod/prometheus-prometheus-pushgateway-7c758897fd-cttlb  1/1     Running   0           5m54s
pod/prometheus-server-55768b86b9-5rjqj             2/2     Running   0           5m54s

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE
service/kubernetes                  ClusterIP     10.96.0.1     <none>       443/TCP          83m
service/prometheus-alertmanager     ClusterIP     10.104.30.167 <none>       9093/TCP         5m54s
service/prometheus-alertmanager-headless ClusterIP     None          <none>       9093/TCP         5m54s
service/prometheus-kube-state-metrics ClusterIP     10.96.185.187 <none>       8080/TCP         5m54s
service/prometheus-prometheus-node-exporter ClusterIP     10.106.132.146 <none>       9100/TCP         5m54s
service/prometheus-prometheus-pushgateway ClusterIP     10.103.200.90 <none>       9091/TCP         5m54s
service/prometheus-server            ClusterIP     10.109.183.118 <none>       80/TCP           5m54s
service/prometheus-server-ext        NodePort      10.111.104.25 <none>       80:32753/TCP    101s

NAME                                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-prometheus-node-exporter 1          1         1         1             1           kubernetes.io/os=linux 5m54s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-kube-state-metrics 1/1     1             1           5m54s
deployment.apps/prometheus-prometheus-pushgateway 1/1     1             1           5m54s
deployment.apps/prometheus-server            1/1     1             1           5m54s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-kube-state-metrics-76fc9c6f55 1          1         1           5m54s
replicaset.apps/prometheus-prometheus-pushgateway-7c758897fd 1          1         1           5m54s
replicaset.apps/prometheus-server-55768b86b9 1          1         1           5m54s
```

# Helm을 이용한 모니터링 시스템 만들기

- 프로메테우스(Prometheus)
- 외부에서 접속(VM에서 접속)

```
kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext  
minikube service prometheus-server-ext
```

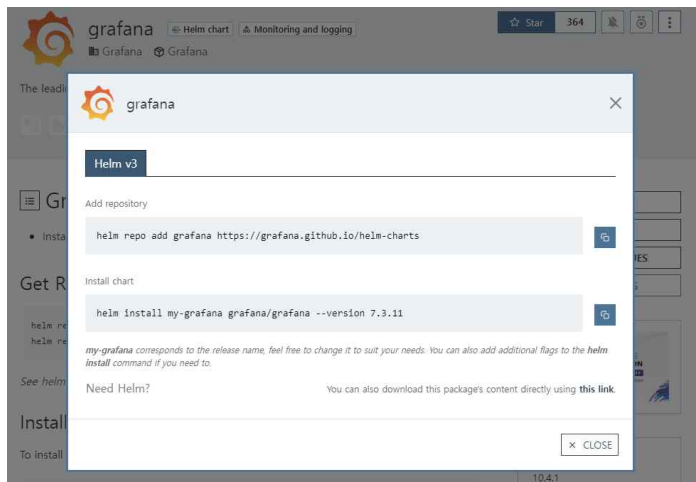
The image shows a terminal window and a web browser. The terminal window displays the command `minikube service prometheus-server-ext` and its output, which is a table showing the service details.

NAMESPACE	NAME	TARGET PORT	URL
default	prometheus-server-ext	80	http://192.168.58.2:32753

The web browser shows the Prometheus UI at the address `192.168.58.2:32753/graph?g0.expr=&g0.tab=1&g0.display_mode=lines&g0.show_exemplars=0`. The UI includes a search bar, a query input field, and a graph area.

# Helm을 이용한 모니터링 시스템 만들기

- 그라파나(Grafana)
- <https://artifacthub.io/> 에서 Grafana 검색 - Install - Add repo  
helm repo add grafana <https://grafana.github.io/helm-charts>  
helm repo update



# Helm을 이용한 모니터링 시스템 만들기

## ■ 그라파나(Grafana)

helm install grafana grafana/grafana

kubectl get all

```
spark@ubuntu-VirtualBox:~/cache/helm/repository$ kubectl get all
NAME                                     READY   STATUS    RESTARTS   AGE
pod/grafana-5ccd97668d-cx19x           1/1     Running   0           3m14s
pod/prometheus-alertmanager-0          1/1     Running   0           17m
pod/prometheus-kube-state-metrics-76fc9c6f55-znbn5  1/1     Running   0           17m
pod/prometheus-prometheus-node-exporter-xmmwj  1/1     Running   0           17m
pod/prometheus-prometheus-pushgateway-7c758897fd-cttlb  1/1     Running   0           17m
pod/prometheus-server-55768b86b9-5rjqj  2/2     Running   0           17m

NAME                                     TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service/grafana                         ClusterIP     10.106.237.143 <none>        80/TCP           3m14s
service/grafana-ext                     NodePort     10.98.8.196   <none>        80:31012/TCP     3m3s
service/kubernetes                      ClusterIP     10.96.0.1     <none>        443/TCP          94m
service/prometheus-alertmanager         ClusterIP     10.104.30.167 <none>        9093/TCP         17m
service/prometheus-alertmanager-headless ClusterIP     None         <none>        9093/TCP         17m
service/prometheus-kube-state-metrics   ClusterIP     10.96.185.187 <none>        8080/TCP         17m
service/prometheus-prometheus-node-exporter ClusterIP     10.106.132.146 <none>        9100/TCP         17m
service/prometheus-prometheus-pushgateway ClusterIP     10.103.200.90  <none>        9091/TCP         17m
service/prometheus-server               ClusterIP     10.109.183.118 <none>        80/TCP           17m
service/prometheus-server-ext           NodePort     10.111.104.25 <none>        80:32753/TCP     12m

NAME                                     DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-prometheus-node-exporter  1         1         1       1             1           kubernetes.io/os=linux  17m

NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana                  1/1     1             1           3m14s
deployment.apps/prometheus-kube-state-metrics  1/1     1             1           17m
deployment.apps/prometheus-prometheus-pushgateway  1/1     1             1           17m
deployment.apps/prometheus-server          1/1     1             1           17m
```

# Helm을 이용한 모니터링 시스템 만들기

## ■ 그라파나(Grafana)

kubectl get po

```
spark@ubuntu-VirtualBox:~/cache/helm/repository$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
grafana-5ccd97668d-cx19x           1/1     Running   0           23m
prometheus-alertmanager-0          1/1     Running   0           37m
prometheus-kube-state-metrics-76fc9c6f55-znbn5  1/1     Running   0           37m
prometheus-prometheus-node-exporter-xmmwj       1/1     Running   0           37m
```

pod명 확인 후 패스워드 변경

kubectl exec --namespace default -it pod명 -- /bin/bash #pod 접속

grafana-cli admin reset-admin-password Tiber01! #Tiber01!는 패스워드

# Helm을 이용한 모니터링 시스템 만들기

## ■ 그라파나(Grafana)

kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-ext

minikube service grafana-ext

```
spark@ubuntu-VirtualBox: ~/.cache/helm/repository$ minikube service grafana-ext
```

NAMESPACE	NAME	TARGET PORT	URL
default	grafana-ext	80	http://192.168.58.2:31012



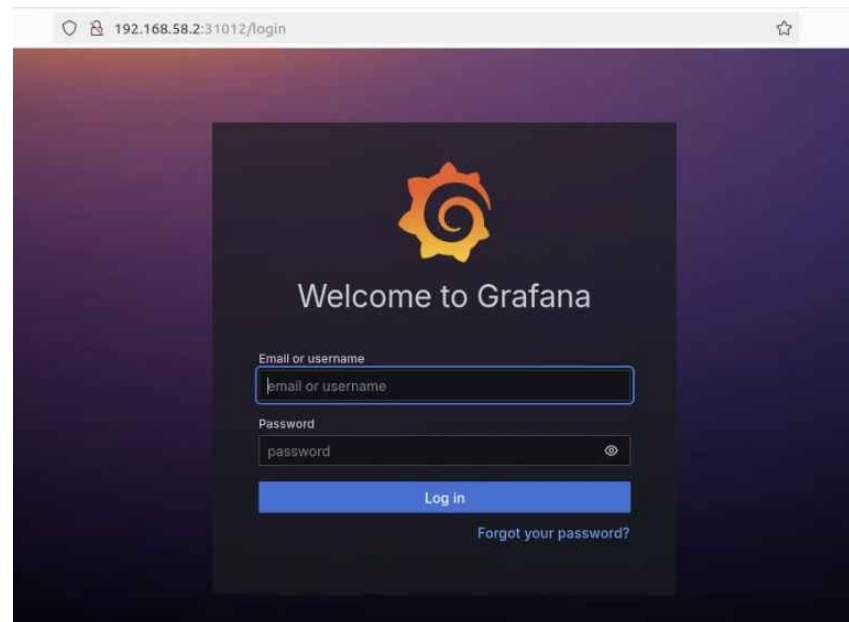
# Helm을 이용한 모니터링 시스템 만들기

- 그라파나(Grafana)

192.168.58.2:31012

ID : admin

PW : 변경한 패스워드



# Wordpress 만들기

- helm chart 구조
- <chart name> /  
Chart.yaml : 차트의 메타 데이터  
values.yaml : 패키지 사용자화  
templates/ : YAML 오브젝트 파일
- artifacthub.io 에서 wordpress 검색 후 repo 추가  
helm repo add bitnami https://charts.bitnami.com/bitnami
- 내부 저장소에서 검색  
helm search repo wordpress

```
ubuntu@ubuntu-VirtualBox:~$ helm search repo wordpress
NAME                CHART VERSION  APP VERSION  DESCRIPTION
bitnami/wordpress  22.2.7         6.5.3        WordPress is the world's most popular blogging ...
bitnami/wordpress-intel 2.1.31        6.1.1        DEPRECATED WordPress for Intel is the most popu...
```



- ```
ubuntu@ubuntu-VirtualBox:~$ he inspect values bitnami/wordpress
# Copyright Broadcom, Inc. All Rights Reserved.
# SPDX-License-Identifier: APACHE-2.0

## @section Global parameters
## Global Docker image parameters
## Please, note that this will override the image parameters, including dependencies, configured to use the global value
## Current available global Docker image parameters: imageRegistry, imagePullSecrets and storageClass

## @param global.imageRegistry Global Docker image registry
## @param global.imagePullSecrets Global Docker registry secret names as an array
## @param global.storageClass Global StorageClass for Persistent Volume(s)
##
global:
  imageRegistry: ""
  ## E.g.
  ## imagePullSecrets:
  ##   - myRegistryKeySecretName
  ##
  imagePullSecrets: []
  storageClass: ""
  ## Compatibility adaptations for Kubernetes platforms
  ##
  compatibility:
    ## Compatibility adaptations for Openshift
    ##
    openshift:
      ## @param global.compatibility.openshift.adaptSecurityContext Adapt the securityContext sections of the deployment to
      ## runAsUser, runAsGroup and fsGroup and let the platform use their allowed default IDs. Possible values: auto (apply if th
      ## he adaptation always), disabled (do not perform adaptation)
      ##
      adaptSecurityContext: auto
```

# Wordpress 만들기

- wordpress 설치

helm install my-wordpress bitnami/wordpress

kubectl get all

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
my-wordpress-cf9758478-wtvqf        1/1     Running   0           3m44s
my-wordpress-mariadb-0              1/1     Running   0           3m44s
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
my-wordpress-cf9758478-wtvqf        1/1     Running   0           3m46s
my-wordpress-mariadb-0              1/1     Running   0           3m46s
ubuntu@ubuntu-VirtualBox:~$ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/my-wordpress-cf9758478-wtvqf    1/1     Running   0           3m49s
pod/my-wordpress-mariadb-0          1/1     Running   0           3m49s

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)
service/kubernetes                  ClusterIP     10.96.0.1     <none>        443/TCP
service/my-wordpress                LoadBalancer 10.106.216.205 <pending>    80:30589/TCP,443:3273
service/my-wordpress-mariadb        ClusterIP     10.96.207.31  <none>        3306/TCP

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-wordpress        1/1     1             1           3m49s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/my-wordpress-cf9758478 1         1         1       3m49s

NAME                                READY   AGE
statefulset.apps/my-wordpress-mariadb 1/1     3m49s
```

# Wordpress 만들기

- Pod 정보 확인

- `kubectl describe po my-wordpress-cf9758478-wtvqf` #개인마다 Pod의 이름은 다름

```
Environment:
  BITNAMI_DEBUG: false
  ALLOW_EMPTY_PASSWORD: yes
  WORDPRESS_SKIP_BOOTSTRAP: no
  MARIADB_HOST: my-wordpress-mariadb
  MARIADB_PORT_NUMBER: 3306
  WORDPRESS_DATABASE_NAME: bitnami_wordpress
  WORDPRESS_DATABASE_USER: bn_wordpress
  WORDPRESS_DATABASE_PASSWORD: <set to the key 'mariadb-password' in secret 'my-wordpres
  WORDPRESS_USERNAME: user
  WORDPRESS_PASSWORD: <set to the key 'wordpress-password' in secret 'my-wordpr
  WORDPRESS_EMAIL: user@example.com
  WORDPRESS_FIRST_NAME: FirstName
  WORDPRESS_LAST_NAME: LastName
  WORDPRESS_HTACCESS_OVERRIDE_NONE: no
  WORDPRESS_ENABLE_HTACCESS_PERSISTENCE: no
  WORDPRESS_BLOG_NAME: User's Blog!
  WORDPRESS_TABLE_PREFIX: wp_
  WORDPRESS_SCHEME: http
  WORDPRESS_EXTRA_LB_CONFIG_CONTENT:
```

# Wordpress 만들기

- wordpress 삭제

helm uninstall my-wordpress bitnami/wordpress

kubectl get all

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
my-wordpress-cf9758478-wtvqf        1/1     Running   0           3m44s
my-wordpress-mariadb-0              1/1     Running   0           3m44s
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
my-wordpress-cf9758478-wtvqf        1/1     Running   0           3m46s
my-wordpress-mariadb-0              1/1     Running   0           3m46s
ubuntu@ubuntu-VirtualBox:~$ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/my-wordpress-cf9758478-wtvqf    1/1     Running   0           3m49s
pod/my-wordpress-mariadb-0          1/1     Running   0           3m49s

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)
service/kubernetes                  ClusterIP      10.96.0.1     <none>        443/TCP
service/my-wordpress                LoadBalancer  10.106.216.205 <pending>    80:30589/TCP,443:3273
service/my-wordpress-mariadb        ClusterIP      10.96.207.31  <none>        3306/TCP

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-wordpress        1/1     1             1           3m49s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/my-wordpress-cf9758478 1         1         1       3m49s

NAME                                READY   AGE
statefulset.apps/my-wordpress-mariadb 1/1     3m49s
```

# Wordpress 만들기

- wordpress 차트 커스터마이징

helm install my-wordpress bitnami/wordpress --set mariadb.auth.username=hello\_wordpress

kubectl get po

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
custmz-wordpress-7c675b5474-kwz9r  1/1     Running   0           92s
custmz-wordpress-mariadb-0         1/1     Running   0           92s
```

kubectl describe po custmz-wordpress-7c675b5474-kwz9r

```
Environment:
  BITNAMI_DEBUG:      false
  ALLOW_EMPTY_PASSWORD:  yes
  WORDPRESS_SKIP_BOOTSTRAP: no
  MARIADB_HOST:        custmz-wordpress-mariadb
  MARIADB_PORT_NUMBER: 3306
  WORDPRESS_DATABASE_NAME: bitnami_wordpress
  WORDPRESS_DATABASE_USER: hello_wordpress
  WORDPRESS_DATABASE_PASSWORD: <set to the key 'mariadb-password' in secret 'custmz-wordpress-mariadb'> Optional: fa
  WORDPRESS_USERNAME:   user
  WORDPRESS_PASSWORD:   <set to the key 'wordpress-password' in secret 'custmz-wordpress'> Optional: fa
  WORDPRESS_EMAIL:      user@example.com
  WORDPRESS_FIRST_NAME: FirstName
  WORDPRESS_LAST_NAME:  LastName
  WORDPRESS_HTACCESS_OVERRIDE_NONE: no
  WORDPRESS_ENABLE_HTACCESS_PERSISTENCE: no
```

# Wordpress 만들기

- wordpress 차트 커스터마이징

helm install custmz-wordpress bitnami/wordpress -f valuesF.yaml

kubectl describe po custmz-wordpress-6fc7c49494-bbv56

```
BITNAMI_DEBUG: false
ALLOW_EMPTY_PASSWORD: yes
WORDPRESS_SKIP_BOOTSTRAP: no
MARIADB_HOST: custmz-wordpress-mariadb
MARIADB_PORT_NUMBER: 3306
WORDPRESS_DATABASE_NAME: hello_wordpress
WORDPRESS_DATABASE_USER: bn_wordpress
WORDPRESS_DATABASE_PASSWORD: <set to the key 'mariadb-p
```

vi valuesF.yaml

```
mariadb:
  auth:
    database: hello_wordpress
```





# TABA\_DB/SQL실습4

# Apache Kafka


## ■ Apache Kafka


- 비즈니스 소셜 네트워크 링크드인(LinkedIn)에서 개발
- 데이터 파이프라인, 스트리밍 분석, 데이터 통합에 사용하는 오픈소스 분산 이벤트 스트리밍 플랫폼
- Publish - Subscribe 시스템으로 데이터가 이동하는 대량의 파이프라인을 구성
- Fortune 100개 기업 중 80% 이상이 Kafka를 사용


### APACHE KAFKA


More than **80% of all Fortune 100 companies** trust, and use Kafka.

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

  
**10 OUT OF 10**  
MANUFACTURING

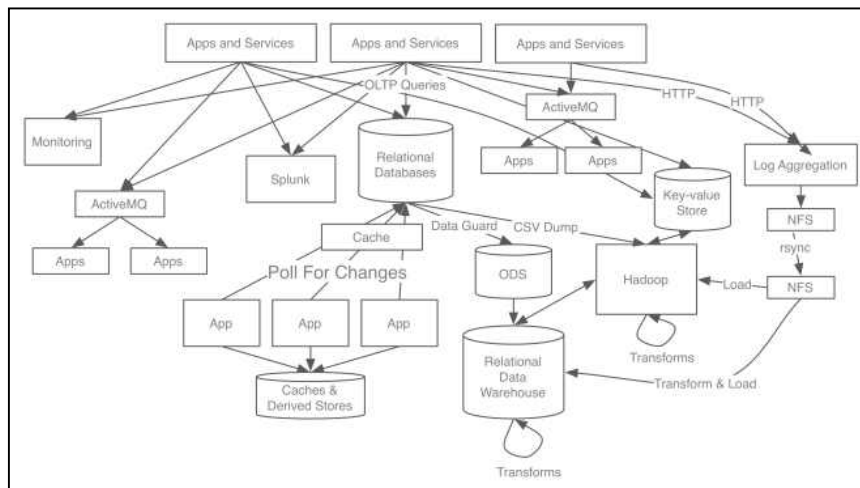
  
**7 OUT OF 10**  
BANKS

  
**10 OUT OF 10**  
INSURANCE

  
**8 OUT OF 10**  
TELECOM

# Apache Kafka

## ■ Before Kafka



## ■ 시스템 복잡도 증가

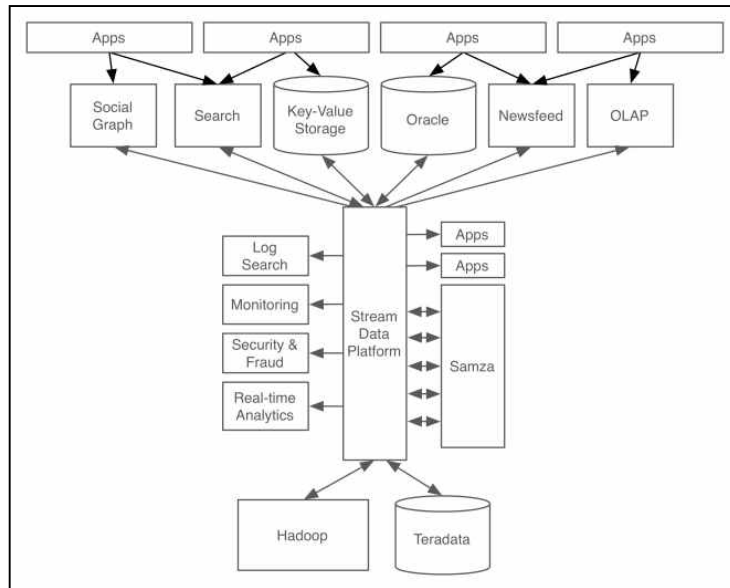
- 데이터 흐름 파악 및 시스템 관리 어려움
- 특정 부분 장애 발생 시 조치 시간 증가(연결된 앱을 체크해야함)
- HW 및 SW 업그레이드 시 관리 포인트 및 작업시간 증가

## ■ 데이터 파이프라인 관리 어려움

- 각 앱마다 시스템 간 별도의 파이프라인 존재
- 파이프라인마다 데이터 포맷 및 처리 방식이 다름
- 새로운 파이프라인 확장이 어려움
- 데이터 불일치 가능성

# Apache Kafka

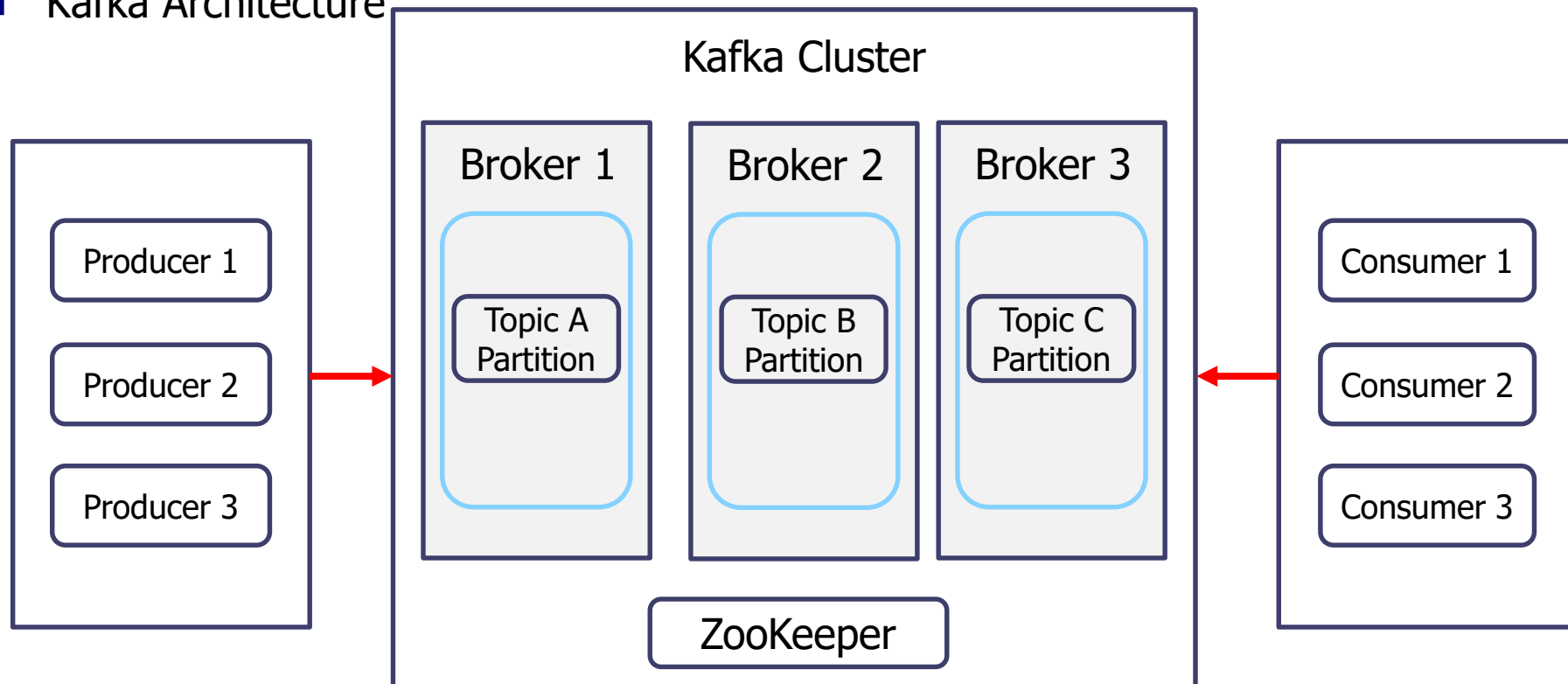
## ■ After Kafka



- 모든 이벤트/ 데이터 흐름을 중앙에서 관리
- 새로운 서비스 및 시스템이 추가되어도 **Kafka**에서 제공하는 표준 포맷으로 연결
- 개발자는 각 서비스 연결이 아닌, 서비스들의 비즈니스 로직에 집중 가능

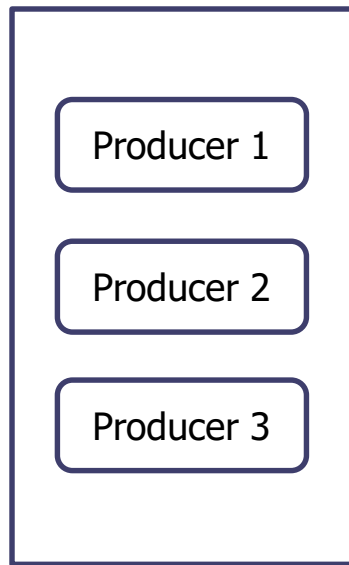
# Apache Kafka

## ■ Kafka Architecture





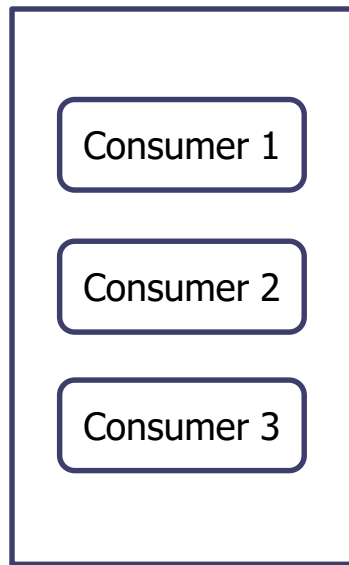
# Apache Kafka



## ■ Producer

- 메시지를 만들어서 카프카 클러스터에 전송
- 메시지 전송 시 **Batch** 처리 가능
- **Key**값을 지정해 특정 파티션으로만 전송 가능

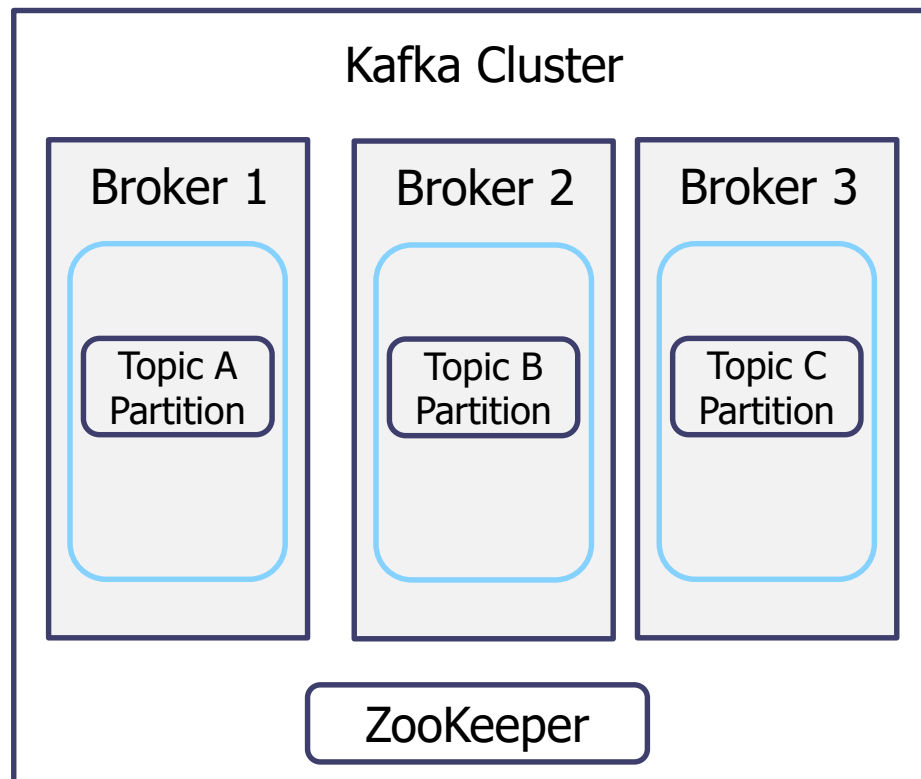
# Apache Kafka



## ■ Consumer

- 카프카 클러스터에서 메시지를 읽고 처리
- 한 개의 컨슈머는 여러 개의 토픽을 처리할 수 있음
- 한 번 저장된 메시지를 여러 번 소비 가능
- 컨슈머는 컨슈머 그룹에 속함
  - 어떤 컨슈머가 다운된다면, 다른 컨슈머가 해당 파티션의 메시지를 다시 구독한다.
  - **offset** 정보를 그룹간 공유하기 때문에 다운되기 전 마지막으로 읽었던 메시지 위치로부터 시작함

# Apache Kafka



## ■ Topic

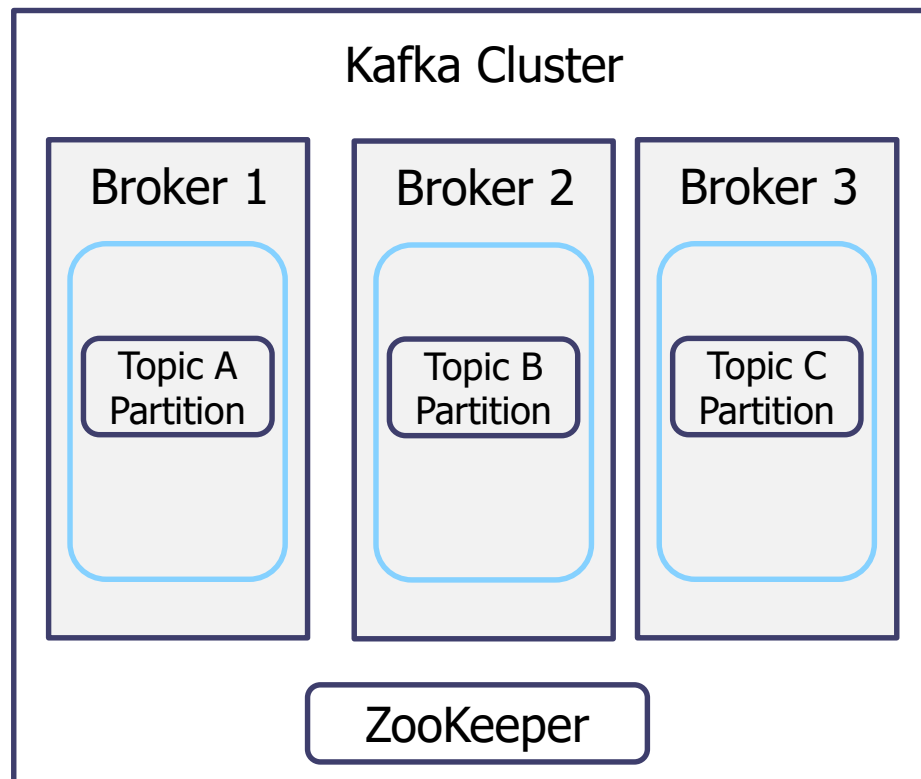
- 각각의 메시지를 목적에 맞게 구분
- 메시지를 전송 및 소비할 때 토픽을 반드시 입력
- 컨슈머는 자신이 담당하는 토픽의 메시지를 처리
- 한 개의 토픽은 한 개 이상의 파티션으로 구성

## ■ Partition

- 분산 처리를 위해 사용
- 토픽 생성 시 파티션 개수 지정 가능
- 파티션 내부에 메시지는 **offset**으로 구분
- 파티션이 많으면 처리량이 좋지만 장애 복구 시간 증가



# Apache Kafka



## ■ Offset

- 컨슈머에서 메시지를 어디까지 읽었는지 저장하는 값
- 컨슈머 그룹의 컨슈머들은 각각의 파티션에 자신이 가져간 메시지의 위치 정보(**offset**)을 기록
- 컨슈머 장애 발생 후 복구되면 마지막으로 읽었던 위치에서 다시 읽음

## ■ Broker

- 실행된 카프카 서버
- 프로시저와 컨슈머는 별도의 앱으로 구성
- 브로커는 **Kafka Cluster** 내부 존재
- 저장하고 관리하는 역할 수행

## ■ ZooKeeper

- 분산 앱 관리를 위한 코디네이션 시스템



# Install Kafka on minikube

## ■ Strimz Operator

- Strimz 는 쿠버네티스 환경에서 kafka를 운영하기 위해 만들어진 operator
- 간단하게 카프카 클러스터, 구성요소 배포 및 관리, 설정, 브로커 관리, 토픽, 유저 생성 등 가능
- <https://strimzi.io/>
- Cluster Operator : Apache Kafka cluster, Kafka Connect 등 관리
- Entity Operator : 토픽 및 사용자로 구성



# Install Kafka on minikube

- Install Kafka

권장하는 메모리 사양

```
minikube start --memory=4096
```

- kafka 네임스페이스 생성

```
kubectl create namespace kafka
```

- [strimzi.io](https://strimzi.io)에서 다운받는 ClusterRoles 및 ClusterRoleBindings용 YAML에는 myproject라는 네임스페이스가 기본

- 네임스페이스를 kafka로 사용하도록 변경.

```
kubectl create -f 'https://strimzi.io/install/latest?namespace=kafka' -n kafka
```

# Install Kafka on minikube

- 생성된 pod 및 CRD 확인
- CRD(Custom Resources) :Kafka, 토픽에 사용할 사용자 정의 리소스

kubectl get crd -n kafka

kubectl get all -n kafka

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get all -n kafka
NAME                                     READY   STATUS    RESTARTS   AGE
pod/strimzi-cluster-operator-b9c59999f-2bdlr   1/1     Running   0          59s

NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/strimzi-cluster-operator   1/1     1            1          59s

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/strimzi-cluster-operator-b9c59999f   1         1         1       59s
```

# Install Kafka on minikube

## ■ Apache Kafka cluster 생성

kubectl apply -f <https://strimzi.io/examples/latest/kafka/kraft/kafka-single-node.yaml> -n kafka

위 YAML을 확인하려면 아래 링크로 접속하면 됨

<https://github.com/strimzi/strimzi-kafka-operator/blob/main/examples/kafka/kraft/kafka-single-node.yaml>

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaNodePool
metadata:
  name: dual-role
  labels:
    strimzi.io/cluster: my-cluster
spec:
  replicas: 1
  roles:
    - controller
    - broker
  storage:
    type: jbod
    volumes:
      - id: 0
        type: persistent-claim
        size: 100Gi
        deleteClaim: false
        kraftMetadata: shared
---
```

controller와 broker가 생성되는  
것을 확인

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
  annotations:
    strimzi.io/node-pools: enabled
    strimzi.io/kraft: enabled
spec:
  kafka:
    version: 3.7.0
    metadataVersion: 3.7-IV4
    listeners:
      - name: plain
        port: 9092
        type: internal
        tls: false
      - name: tls
        port: 9093
        type: internal
        tls: true
    config:
      offsets.topic.replication.factor: 1
      transaction.state.log.replication.factor: 1
      transaction.state.log.min.isr: 1
      default.replication.factor: 1
      min.insync.replicas: 1
```

listener의 정보 확인

plain(암호화되지 않고 전송 또는 저장된 데이터를 의미)

9092로 접속하면 통신이 가능

# Install Kafka on minikube

## ■ 생성된 pod 확인

kubectl get all -n kafka

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get all -n kafka
NAME                                READY   STATUS    RESTARTS   AGE
pod/my-cluster-dual-role-0          1/1     Running   0           39m
pod/my-cluster-entity-operator-d6c5c645-tkksn  2/2     Running   0           38m
pod/strimzi-cluster-operator-b9c59999f-2bd1r  1/1     Running   0           43m

NAME                                TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)                                     AGE
service/my-cluster-kafka-bootstrap  ClusterIP     10.97.28.196 <none>        9091/TCP,9092/TCP,9093/TCP                39m
service/my-cluster-kafka-brokers    ClusterIP     None         <none>        9090/TCP,9091/TCP,8443/TCP,9092/TCP,9093/TCP 39m

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-cluster-entity-operator  1/1     1             1           38m
deployment.apps/strimzi-cluster-operator    1/1     1             1           43m

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/my-cluster-entity-operator-d6c5c645  1         1         1       38m
replicaset.apps/strimzi-cluster-operator-b9c59999f  1         1         1       43m
```



# Send and receive msg

- 두 개의 putty 실행
- 1번 putty에는 Producer , 2번 putty에는 Consumer
- Producer 생성

```
kubectl -n kafka run kafka-producer -ti --image=quay.io/strimzi/kafka:0.41.0-kafka-3.7.0 --rm=true ₩  
--restart=Never -- bin/kafka-console-producer.sh --bootstrap-server my-cluster-kafka-bootstrap:9092 ₩  
--topic my-topic
```

- Consumer 생성

```
kubectl -n kafka run kafka-consumer -ti --image=quay.io/strimzi/kafka:0.41.0-kafka-3.7.0 --rm=true ₩  
--restart=Never -- bin/kafka-console-consumer.sh --bootstrap-server my-cluster-kafka-bootstrap:9092 ₩  
--topic my-topic --from-beginning
```

# Send and receive msg

- 프로시저에서 메시지를 보내면 컨슈머에서 메시지 확인 가능
- 단, topic 및 port가 동일해야함

```
ubuntu@ubuntu-VirtualBox:~$ kubectl -n kafka run kafka-producer -ti --image=quay.io/strimzi/kafka:0.41.0-kafka-3.7.0 --rm=true --restart=Never -- bin/kafka-console-producer.sh --bootstrap-server=cluster-kafka-bootstrap:9092 --topic my-topic
If you don't see a command prompt, try pressing enter.
>dankook
>
```

```
ubuntu@ubuntu-VirtualBox:~$ kubectl -n kafka run kafka-consumer -ti --image=quay.io/strimzi/kafka:0.41.0-kafka-3.7.0 --rm=true --restart=Never -- bin/kafka-console-consumer.sh --bootstrap-server=cluster-kafka-bootstrap:9092 --topic my-topic --from-beginning
If you don't see a command prompt, try pressing enter.
hello
hello
dankook
```





# Kafka Connect

- Kafka Connect(커넥트)

- Kafka의 컴포넌트 중 하나
- 커넥터(Connector)를 동작하도록 실행해주는 프로세스
- 커넥터를 사용하기 위해서는 커넥트가 실행되어야 함.
- 커넥트를 이용해 **kafka**로부터 데이터를 보내거나, 데이터를 가지고 올 수 있음.
- 이미 만들어진 기존 커넥트를 활용할 수 있고, 운영 환경에서 변경 가능



# Kafka Connect

## ■ Kafka Connect

- 단일 실행 모드 커넥트
  - 간단한 데이터 파이프라인을 구성하거나 개발용으로 사용
- 분산 모드 커넥트
  - 실질적으로 상용화에 활용하려면 분산 모드 커넥트로 구성해서 운영
  - 여러 개의 커넥트를 묶어서 운영하는 방식
  - 클러스터로 묶은 커넥트는 일부 커넥트에 장애가 발생해도 파이프라인을 자연스럽게 장애 조치해서 실행중인 커넥트를 처리할 수 있게 함.



# Kafka Connect

## ■ Kafka Connector(커넥터)

- 커넥터는 실질적으로 데이터를 처리하는 코드가 담긴 **JAR** 패키지
- 따라서 커넥터 안에는 파이프라인에 필요한 메서드, 설정 등 코드가 있음
- 싱크 커넥터(Sink Connector)
  - 데이터를 싱크한다는 뜻으로, 특정 토픽에 있는 데이터를 오라클, MySQL 등 특정 DB로 보내는 역할을 하는 커넥터 (컨슈머와 같은 역할)
- 소스 커넥터(Source Connector)
  - DB로부터 데이터를 가져와서 토픽에 넣는 역할 (프로듀서와 같은 역할)

# Kafka Connect 예제

- 아래와 같은 환경 구축
- MySQL - Kafka - PostgreSQL을 커넥터로 연결
- MySQL에서 데이터를 생성하면 데이터 PostgreSQL에서 받기



# Kafka Connect 예제

## ■ MySQL on Kubernetes

mysql.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  namespace: kafka
spec:
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - image: mysql:5.6
          name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: Dankook1!
            - name: LC_ALL
              value: C.UTF-8
          ports:
            - containerPort: 3306
              name: mysql
          volumeMounts:
            - name: mysql-initdb
              mountPath: /docker-entrypoint-initdb.d
          volumes:
            - name: mysql-initdb
              configMap:
                name: mysql-config
                items:
                  - key: initdb.sql
                    path: initdb.sql
```

mysql-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: mysql
  namespace: kafka
spec:
  ports:
    - port: 3306
  selector:
    app: mysql
```

MySQL 환경변수로 비밀번호 설정  
현재는 Dankook1!로 설정되어있음.

# Kafka Connect 예제

## ■ MySQL on Kubernetes

mysql-config.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-config
  namespace: kafka
data:
  initdb.sql: |
    SET GLOBAL binlog_format = 'ROW';
    SET GLOBAL server_id = 184054;
    CREATE DATABASE test;
    use test;
    CREATE TABLE example (id INT(16) NOT NULL, name
    VARCHAR(32));
    INSERT INTO example VALUES (111, 'connecttest');
```

initdb.sql은 MySQL이 처음 실행될 때 수행되는 명령어

1. test 데이터베이스 생성
2. test 데이터베이스 사용
3. example 테이블 생성 및 데이터 생성

# Kafka Connect 예제

## ■ PostgreSQL on Kubernetes

postgresql.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres
  namespace: kafka
spec:
  selector:
    matchLabels:
      app: postgres
  replicas: 1
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
        - name: postgres
          image: postgres:9.6.2-alpine
          ports:
            - containerPort: 5432
          env:
            - name: TZ
              value: 'Asia/Seoul'
            - name: POSTGRES_USER
              value: postgres
            - name: POSTGRES_PASSWORD
              value: Dankook1!
```

postgresql-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: postgres-service
  namespace: kafka
  labels:
    app: postgres
spec:
  type: ClusterIP
  ports:
    - port: 5432
  selector:
    app: postgres
```

PostgreSQL의 환경 변수 설정  
패스워드는 Dankook1!

# Kafka Connect 예제

- 생성된 데이터베이스 확인

kubectl get po -n kafka

```
oem@ubuntu-server:~/strimzi-kafka-example$ kubectl get po -n kafka
NAME                                READY   STATUS    RESTARTS   AGE
my-cluster-dual-role-0              1/1     Running   0           89m
my-cluster-entity-operator-d6c5c645-lc8xx  2/2     Running   0           89m
mysql-5f87695797-dqdcn             1/1     Running   0           19m
postgres-f9559496c-6pgzt           1/1     Running   0           18m
strimzi-cluster-operator-b9c59999f-667ng  1/1     Running   0           90m
```





# Kafka Connect 예제

- 스키마 레지스트리(Schema Registry)
  - 카프카 클라이언트 사이에서 메시지의 스키마를 저장, 관리하는 웹 어플리케이션
  - 데이터의 호환성 유지 및 관리를 위해 개발
  - **Producer, Consumer**가 많을 수 있는 분산 시스템 환경에서 데이터에 대한 호환성을 유지
  - **strimzi**에서는 제공하지 않기 때문에 **confluent**에서 제공하는 스키마 레지스트리 사용

# Kafka Connect 예제

## ■ Schema\_registry

### schema-registry.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: schema-registry
  namespace: kafka
spec:
  selector:
    matchLabels:
      app: schema-registry
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: schema-registry
    spec:
      containers:
        - name: my-cluster-schema-registry
          image: confluentinc/cp-schema-registry:7.0.1
          ports:
            - containerPort: 8081
          env:
            - name: SCHEMA_REGISTRY_KAFKASTORE_BOOTSTRAP_SERVERS
              value: my-cluster-kafka-bootstrap:9092
            - name: SCHEMA_REGISTRY_HOST_NAME
              valueFrom:
                fieldRef:
                  fieldPath: status.podIP
            - name: SCHEMA_REGISTRY_LISTENERS
              value: http://0.0.0.0:8081
            - name: SCHEMA_REGISTRY_KAFKASTORE_SECURITY_PROTOCOL
              value: PLAINTEXT
```

### schema-registry-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: schema-registry
  namespace: kafka
spec:
  ports:
    - port: 8081
  clusterIP: None
  selector:
    app: schema-registry
```

# Kafka Connect 예제

## ■ Kafka connect

kafka-connect.yaml

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
  name: my-connect-cluster3
  namespace: kafka
  annotations:
    # use-connector-resources configures this KafkaConnect
    # to use KafkaConnector resources to avoid
    # needing to call the Connect REST API directly
    strimzi.io/use-connector-resources: "true"
spec:
  image: aldfkaks/kafka-connect-jdbc:v1.0
  replicas: 1
  bootstrapServers: my-cluster-kafka-bootstrap:9092
  config:
    config.storage.replication.factor: 1
    offset.storage.replication.factor: 1
    status.storage.replication.factor: 1
    config.providers: file
    config.providers.file.class: org.apache.kafka.common.config.provider.FileConfigProvider
    key.converter: io.confluent.connect.avro.AvroConverter
    key.converter.schema.registry.url: http://schema-registry.kafka.svc.cluster.local:8081
    value.converter: io.confluent.connect.avro.AvroConverter
    value.converter.schema.registry.url: http://schema-registry.kafka.svc.cluster.local:8081
```

spec.image : 사용할 컨넥트 이미지

spec.bootstrapServers : 카프카 클러스터 주소

# Kafka Connect 예제

## ■ MySQL Source connector

mysql-source-connector.yaml

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnector
metadata:
  name: jdbc-mysql-source-connector
  namespace: kafka
  labels:
    strimzi.io/cluster: my-connect-cluster3
spec:
  class: io.confluent.connect.jdbc.JdbcSourceConnector
  tasksMax: 1
  config:
    mode: "incrementing"
    incrementing.column.name: "id"
    connection.url: "jdbc:mysql://mysql.kafka.svc.cluster.local:3306/test"
    connection.user: "root"
    connection.password: "Dankook1!"
    table.whitelist: "example"
```

config.connection.url : 연결할 데이터베이스(test  
데이터베이스를 연결)

config.connection.user : root 유저

config.connection.password : 비밀번호

config.table.whitelist: 어떤 테이블에 대해 읽을지 필터링

# Kafka Connect 예제

## ■ MySQL Source connector

postgres-sink-connector.yaml

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnector
metadata:
  name: postgres-connector
  namespace: kafka
  labels:
    strimzi.io/cluster: my-connect-cluster3
spec:
  class: io.confluent.connect.jdbc.JdbcSinkConnector
  tasksMax: 1
  config:
    topics: "example"
    connection.url: "jdbc:postgresql://postgres-service.kafka.svc.cluster.local:5432/postgres"
    connection.user: "postgres"
    connection.password: "Dankook1!"
    auto.create: "true"
```

config.topics : 구독할 테이블  
config.connection.url: 저장 될 PostgreSQL의  
데이터베이스  
config.connection.user : root 유저  
config.connection.password : 비밀번호  
config.table.whitelist: 어떤 테이블에 대해 읽을지 필터링

# Kafka Connect 예제

- 각 데이터 베이스 접속하기

kubectl get po -n kafka

```
oem@ubuntu-server:~/strimzi-kafka-example$ kubectl get po -n kafka
NAME                                READY   STATUS    RESTARTS   AGE
my-cluster-dual-role-0              1/1     Running   0           113m
my-cluster-entity-operator-d6c5c645-1c8xx  2/2     Running   0           112m
my-connect-cluster3-connect-0       1/1     Running   0           3m39s
mysql-6c46664b98-wccqv              1/1     Running   0           3m21s
postgres-f9559496c-tdqnb            1/1     Running   0           3m18s
strimzi-cluster-operator-b9c59999f-667ng  1/1     Running   0           113m
```

# Kafka Connect 예제

## ■ 각 데이터 베이스 접속하기

### MySQL

- `kubectl exec -it mysql-6c46664b98-wccqv -n kafka -- mysql -u root -p`
- 각 개인마다 pod명은 다름

```
dem@ubuntu-server:~/strimzi-kafka-example$ kubectl exec -it mysql-6c46664b98-wccqv -n kafka -- mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.6.51 MySQL Community Server (GPL)

Copyright (c) 2000, 2021, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

# Kafka Connect 예제

## ■ 각 데이터 베이스 접속하기

생성된 데이터베이스, 테이블 및 데이터 조회

show databases;

use test;

select \* from example;

exit; #데이터베이스 나가기

```
oem@ubuntu-server:~$ kubectl exec -it mysql-6987bc7f6b-jbf4p -n kafka -- mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.6.51 MySQL Community Server (GPL)

Copyright (c) 2000, 2021, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from example;
+-----+
| id | name      |
+-----+
| 111 | connecttest |
+-----+
1 row in set (0.00 sec)
```



# Kafka Connect 예제

## ■ 각 데이터 베이스 접속하기

### PostgreSQL

- `kubectl exec -it postgres-f9559496c -n kafka -- psql -U postgres`
- `\dt`
- `\q` #데이터베이스 종료

```
oem@ubuntu-server:~$ kubectl exec -it postgres-75fcff5cc5-x2861 -n kafka -- psql -U postgres
psql (9.6.2)
Type "help" for help.

postgres=# \dt
               List of relations
 Schema | Name   | Type  | Owner
-----+-----+-----+-----
 public | example | table | postgres
(1 row)
```

# 크롤링 데이터 DB에 저장하기

- Python에서 크롤링 데이터 MySQL에 저장하기

- Python Pod 생성

python.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: python
  namespace: kafka
  labels:
    app: python
spec:
  containers:
  - name: python
    image: python
    command: ["sleep", "infinity"]
```

# 크롤링 데이터 DB에 저장하기

- MySQL에 새로운 DB 및 유저 생성

- `kubect exec -it mysql-6c46664b98-wccqv -n kafka -- mysql -u root -p`

`CREATE DATABASE crawl_db DEFAULT CHARACTER SET utf8; #데이터베이스 생성`

`CREATE USER crawl_user IDENTIFIED BY 'Dankook1!'; #유저명 :crawl_user, 패스워드 Dankook1!`

`GRANT ALL ON crawl_db.* TO crawl_user; crawl_db 데이터베이스의 권한을 crawl_crawl_user에게 주기`

```
mysql> CREATE DATABASE crawl_db DEFAULT CHARACTER SET utf8;
Query OK, 1 row affected (0.00 sec)

mysql> CREATE USER crawl_user IDENTIFIED BY 'Dankook1!';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL ON crawl_db.* TO crawl_user;
Query OK, 0 rows affected (0.00 sec)
```

# 크롤링 데이터 DB에 저장하기

- Python Pod로 접속 후 파이썬 패키지 설치

- `kubectl get po -n kafka`

```
oem@ubuntu-server:~$ kubectl get po -n kafka
NAME                                READY   STATUS    RESTARTS   AGE
my-cluster-dual-role-0             1/1     Running   0           26h
my-cluster-entity-operator-d6c5c645-1c8xx  2/2     Running   0           26h
my-connect-cluster3-connect-0      1/1     Running   0           24h
mysql-6987bc7f6b-jbf4p             1/1     Running   0           15h
postgres-75fcff5cc5-x286l          1/1     Running   0           14h
python                             1/1     Running   0           6h43m
schema-registry-745c585464-qqr8     1/1     Running   0           14h
strimzi-cluster-operator-b9c59999f-667ng 1/1     Running   0           26h
```

`kubectl exec -it python -n kafka -- /bin/bash` #생성한 Python pod 접속

`apt update && apt install vim` #pod 접속 후 업데이트 및 vim 편집기 설치

`pip install mysqlclient` #pip을 이용해 mysqlclient 패키지 설치

# 크롤링 데이터 DB에 저장하기

## ■ Python - MySQL 연결 확인 테이블 생성

vi mysqltest.py

```
import MySQLdb
```

```
conn = MySQLdb.connect(  
    user="crawl_user",  
    passwd="Dankook1!",  
    host="10.96.61.228",  
    db="crawl_db")
```

```
print(type(conn))  
# <class 'MySQLdb.connections.Connection'>  
cursor = conn.cursor()  
print(type(cursor))  
# <class 'MySQLdb.cursors.Cursor'>  
cursor.execute("CREATE TABLE test (id int)")  
cursor.execute("insert into test values(123)")
```

```
conn.commit()
```

host 확인 하는 방법  
kubectl get svc -n kafka

```
pem@ubuntu-server:~$ kubectl get svc -n kafka
```

| NAME                            | TYPE      | CLUSTER-IP     | EXTERNAL-IP | PORT(S)            |
|---------------------------------|-----------|----------------|-------------|--------------------|
| my-cluster-kafka-bootstrap      | ClusterIP | 10.107.39.6    | <none>      | 9091/TCP, 9092/TCP |
| my-cluster-kafka-brokers        | ClusterIP | None           | <none>      | 9090/TCP, 9092/TCP |
| my-connect-cluster3-connect     | ClusterIP | None           | <none>      | 8083/TCP           |
| my-connect-cluster3-connect-api | ClusterIP | 10.107.230.207 | <none>      | 8083/TCP           |
| mysql                           | ClusterIP | 10.100.111.92  | <none>      | 3306/TCP           |
| postgres-service                | ClusterIP | 10.109.191.109 | <none>      | 5432/TCP           |
| schema-registry                 | ClusterIP | None           | <none>      | 8081/TCP           |

```
mysql> desc test;
```

| Field | Type    | Null | Key | Default | Extra |
|-------|---------|------|-----|---------|-------|
| id    | int(11) | YES  |     | NULL    |       |

```
1 row in set (0.00 sec)
```

# 크롤링 데이터 DB에 저장하기

## ■ vi melon.py # (1)

pip install requests bs4 # 패키지 설치

```
import MySQLdb
import requests
from bs4 import BeautifulSoup

if __name__ == "__main__":
    RANK = 100 # 멜론 차트 순위가 1 ~ 100위까지 있음

    header = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko'
    }
    req = requests.get('https://www.melon.com/chart/week/index.htm',
                       headers=header) # 주간 차트를 크롤링 할 것임
    html = req.text
    parse = BeautifulSoup(html, 'html.parser')

    titles = parse.find_all("div", {"class": "ellipsis rank01"})
    singers = parse.find_all("div", {"class": "ellipsis rank02"})

    title = []
    singer = []

    for t in titles:
        title.append(t.find('a').text)

    for s in singers:
        singer.append(s.find('span', {"class": "checkEllipsis"}).text)
    items = [item for item in zip(title, singer)]
```



# 크롤링 데이터 DB에 저장하기

## ■ vi melon.py # (2)

```
conn = MySQLdb.connect(
    user="crawl_user",
    passwd="Dankook1!",
    host="10.100.111.92",
    db="crawl_db"
    # charset="utf-8"
)
cursor = conn.cursor()
cursor.execute("DROP TABLE IF EXISTS melon")

cursor.execute("CREATE TABLE melon (`rank` int, title text, singer text)")

i = 1

for item in items:
    cursor.execute(
        "INSERT INTO melon (rank, title, singer) VALUES (%s, %s, %s)", (i, item[0], item[1]))
    i += 1

conn.commit()
```

# 크롤링 데이터 확인하기

## ■ MySQL Source Connector

mysql-source-connector.yaml

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnector
metadata:
  name: jdbc-mysql-source-connector
  namespace: kafka
  labels:
    strimzi.io/cluster: my-connect-cluster3
spec:
  class: io.confluent.connect.jdbc.JdbcSourceConnector
  tasksMax: 1
  config:
    mode: "bulk"
    poll.interval.ms: "86400000"
    connection.url: "jdbc:mysql://mysql.kafka.svc.cluster.local:3306/crawl_db"
    connection.user: "crawl_user"
    connection.password: "Dankook1!"
    table.whitelist: "melon"
```

### Connector mode

1. **incrementing** : 특정 칼럼의 증가분만 감지, 기존 행의 수정 및 삭제는 감지되지 않음. 따라서 특정 칼럼을 작성해줘야함.  
mode: "incrementing"  
incrementing.column.name: "id"
2. **bulk** : 데이터를 폴링할 때 마다 전체 테이블 복사. 폴링 시간을 적어줘야함.  
mode: "bulk"  
poll.interval.ms: "86400000"
3. **timestamp** : timestamp형 칼럼일 경우 새 행과 수정된 행을 감지  
mode: "timestamp"  
timestamp.column.name: "date"



# 크롤링 데이터 확인하기

## ■ MySQL Source Connector

postgres-sink-connector.yaml

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnector
metadata:
  name: postgres-connector
  namespace: kafka
  labels:
    strimzi.io/cluster: my-connect-cluster3
spec:
  class: io.confluent.connect.jdbc.JdbcSinkConnector
  tasksMax: 1
  config:
    topics: "melon"
    connection.url: "jdbc:postgresql://postgres-service.kafka.svc.cluster.local:5432/postgres"
    connection.user: "postgres"
    connection.password: "Dankook1!"
    auto.create: "true"
```

# 크롤링 데이터 확인하기

## ■ 커넥터 배포하기

```
kubectl apply -f mysql-source-connector.yaml  
kubectl apply -f postgres-sink-connector.yaml
```

```
kubectl get KafkaConnector -n kafka
```

```
oem@ubuntu-server:~/strimzi-kafka-example$ kubectl get KafkaConnector -n kafka  
NAME                                CLUSTER           CONNECTOR CLASS      MAX TASKS  READY  
jdbc-mysql-source-connector         my-connect-cluster3  io.confluent.connect.jdbc.JdbcSourceConnector    1          True  
postgres-connector                 my-connect-cluster3  io.confluent.connect.jdbc.JdbcSinkConnector      1          True
```

# 크롤링 데이터 확인하기

## ■ PostgreSQL 접속 및 데이터 확인

kubectrl exec -it postgres-75fcff5cc5-whcxx -n kafka -- psql -U postgres

\dt #스키마 조회

select \* from melon limit 10; #melon 테이블 상위 10개 출력

```
postgres=# \dt
          List of relations
 Schema | Name  | Type  | Owner
-----+-----+-----+-----
 public | melon | table | postgres
(1 row)

postgres=# select * from melon limit 10;
 rank |          title          | singer
-----+-----+-----
  1 | Supernova               | aespa
  2 | SPOT! (feat. JENNIE)   | 지코 (ZICO)
  3 | 헤아 (HEYA)            | IVE (아이브)
  4 | Magnetic               | 아일릿 (ILLIT)
  5 | 고민종류              | QWER
  6 | 나는 아픈 건 말 질색이니까 | (여자)아이들
  7 | 소나기                 | 이클립스 (ECLIPSE)
  8 | 미안해 미워해 사랑해  | Crush
  9 | 첫 만남은 계획대로 되지 않아 | TWS (투어스)
 10 | 전상연                 | 이창현
(10 rows)
```