```
SELECT TABLE_NAME FROM USER_TABLES;

--INSERT?
--INSERT INTO <T/N>(C/N)
--VALUES(VALUES)
SELECT * FROM ENROL;
CREATE TABLE A_ENROL
AS
SELECT *
FROM ENROL
WHERE STU_NO <'20150000';

DESC A_ENROL;
SELECT * FROM A_ENROL;

DROP TABLE A_ENROL;

INSERT INTO A_ENROL(SUB_NO,STU_NO,ENR_GRADE)
VALUES(108,20151062,92);

INSERT INTO A_ENROL
VALUES(109,20151088,85);

SELECT * FROM A_ENROL;

INSERT INTO A_ENROL(SUB_NO,STU_NO)
VALUES(110,20152088);

INSERT INTO A_ENROL
VALUES(111,20153075,NULL);

SELECT * FROM A_ENROL;
```

```
--복수행 삽입?
SELECT * FROM ENROL;
SELECT * FROM ENROL
WHERE STU_NO LIKE '2015%';

INSERT INTO A_ENROL
SELECT * FROM ENROL
WHERE STU_NO LIKE '2015%';

SELECT * FROM A_ENROL;

--UPDATE?
--UPDATE <T/N>
--SET CO--NAME=??
--WHERE ??
SELECT * FROM A_ENROL;

UPDATE A_ENROL
SET ENR_GRADE=ENR_GRADE+5;

UPDATE A_ENROL
SET ENR_GRADE=ENR_GRADE+5
WHERE SUB_NO=101;

--과목이름이 시스템분석설계인 과목만 점수를 10점 업데이트하라.
UPDATE A_ENROL
SET ENR_GRADE=ENR_GRADE+10
WHERE SUB_NO=(SELECT SUB_NO
FROM SUBJECT
WHERE SUB_NAME='시스템분석설계');
```

```
SELECT * FROM A_ENROL;

--DELETE?
--DELETE FROM <T/N>
--WHERE ???
DELETE FROM A_ENROL
WHERE STU_NO = 20131001;

--A_ENROL테이블에서 과목이름이 기계요소설계인 과목번호를 가진 내용을 삭제하시오.
DELETE FROM A_ENROL
WHERE SUB_NO=(SELECT SUB_NO
FROM SUBJECT
WHERE SUB_NAME='기계요소설계');

SELECT SUB_NO
FROM SUBJECT
WHERE SUB_NAME='기계요소설계';

--다중튜플삭제?
DELETE FROM A_ENROL;

--TCL?
SELECT * FROM B_STUDENT;
DELETE FROM B_STUDENT;
SELECT * FROM B_STUDENT;
ROLLBACK;
SELECT * FROM B_STUDENT;

DELETE FROM B_STUDENT;
SELECT * FROM B_STUDENT;

--DDL의 경우 자동으로 커밋이 되기 때문에 롤백이 되지 않음
CREATE TABLE C_STUDENT(STU_NO NUMBER,
```

```
STU_NAME CHAR(10));
ROLLBACK;
SELECT * FROM B_STUDENT;

SELECT * FROM A_STUDENT;
DELETE FROM A_STUDENT;
SELECT * FROM A_STUDENT;
ROLLBACK;

--병행처리?
SELECT *
FROM A_STUDENT;
INSERT INTO A_STUDENT(STU_NO,STU_NAME)
VALUES(10,'홍');
SELECT * FROM A_STUDENT;
COMMIT;

--DML 실전문제
--배경환경구축테이블 생성
CREATE TABLE EMP1
AS
SELECT * FROM EMP
WHERE DEPTNO IN(20,30);

CREATE TABLE DEPT1
AS
SELECT * FROM DEPT;

CREATE TABLE SALGRADE1
AS
SELECT * FROM SALGRADE;

SELECT * FROM EMP1;
```

```
SELECT * FROM DEPT1;
SELECT * FROM SALGRADE1;

--사원번호 7703,사원이름 JOSH,사원직무 SALESMAN,상급자사원번호가
--7566,급여1400,커미션0,부서번호 20인 사원이 오늘 입사하였다.
INSERT INTO EMP1
VALUES(7703,'JOSH','SALESMAN',7566,SYSDATE,1400,0,20);

SELECT * FROM EMP1;

--2.사원번호 7401,사원이름 HOMER,급여 1300,부서번호10인 사원이
--입사하였다.
INSERT INTO EMP1(EMPNO,ENAME,SAL,DEPTNO)
VALUES(7401,'HOMER',1300,10);

SELECT *FROM EMP1;

--3.사원번호 7323,사원이름 'BRENDA'부서번호 30,사원번호 7499와
--동일한 급여를 받는 사원이 입사하였다.
INSERT INTO EMP1(EMPNO,ENAME,SAL,DEPTNO)
VALUES(7323,'BRENDA',(SELECT SAL FROM EMP1 WHERE EMPNO=7499),30);

SELECT * FROM EMP1;

--4.사원(EMP)테이블에서 부서번호가 10인 데이터를 EMP1테이블에 삽입하라?
INSERT INTO EMP1
SELECT * FROM EMP WHERE DEPTNO = 10;
SELECT * FROM EMP1;

--5.사원번호 7369의 사원직무를 ANALYST로 바꾸어라
UPDATE EMP1
SET JOB='ANALYST'
WHERE EMPNO=7369;
SELECT * FROM EMP1;
```

--6.부서번호 20인 직원들의 급여를 10%감하라

```
UPDATE EMP1
SET SAL=SAL-SAL*0.1
WHERE DEPTNO=20;
```

--7.모든 사원의 급여를 10%증가시켜라

```
UPDATE EMP1
SET SAL=SAL+100;
SELECT * FROM EMP1;
```

--8.사원번호 7902 상급자사원번호를 7654,부서번호를 30으로 바꾸라

```
UPDATE EMP1
SET MGR=7654,DEPTNO=30
WHERE EMPNO=7902;

SELECT * FROM EMP1;
```

--9.지역이 DALLAS인 사원들의 급여를 10감하라.

```
UPDATE EMP1
SET SAL=SAL-10
WHERE DEPTNO=(SELECT DEPTNO FROM DEPT1 WHERE LOC='DALLAS');
SELECT * FROM EMP1;
```

--10.급여등급이 2인 사원들의 급여를 20감하라

```
UPDATE EMP1
SET SAL=SAL-20
WHERE EMPNO IN(SELECT EMPNO
FROM EMP1,SALGRADE
WHERE SAL BETWEEN LOSAL AND HISAL
AND GRADE=2);

SELECT EMPNO
FROM EMP1,SALGRADE
```

```
WHERE SAL BETWEEN LOSAL AND HISAL
AND GRADE=2;

SELECT * FROM EMP1;

--11.사원번호7499가 퇴사하였다.
DELETE FROM EMP1
WHERE EMPNO=7499;

SELECT * FROM EMP1;

--12.부서번호 50,부서이름 'PLANNING',지역'MIAMI'가 추가되었다.
INSERT INTO DEPT1
VALUES('50','PLANNING','MIAMI');
SELECT * FROM DEPT1;

--13.부서번호가 40인 부서가 60으로 변경되었다.
UPDATE DEPT1
SET DEPTNO=60
WHERE DEPTNO=40;

SELECT * FROM DEPT1;

--14.부서번호가 30인 부서가 폐지되었다.
DELETE FROM DEPT1
WHERE DEPTNO=30;
SELECT * FROM DEPT1;

--15.DEPT1테이블에 없는 부서번호들을 갖고 있는 사원들의 부서번호를
--99로 변경하라
UPDATE EMP1
SET DEPTNO=99
WHERE DEPTNO NOT IN (SELECT DEPTNO
```

```
FROM DEPT1);
SELECT * FROM EMP1;

--16.EMP1에서 99번 번호를 삭제하라
DELETE FROM EMP1
WHERE DEPTNO=99;

SELECT * FROM EMP1;
--17.상급자사원번호가 없는 사원의 급여를 100올렸다.
UPDATE EMP1
SET SAL=SAL+100
WHERE MGR IS NULL;
SELECT * FROM EMP1;
--18.JONES,JOSH,CLARK가 30번 부서로 옮겼다.
UPDATE EMP1
SET DEPTNO=30
WHERE ENAME IN('JONES','JOSH','CLARK');

SELECT * FROM EMP1;

--19.커미션이 NULL인 데이터를 0으로 바꾸라
--NA,NULL,0
UPDATE EMP1
SET COMM=0
WHERE COMM IS NULL;
SELECT * FROM EMP1;

--20.EMP1의 전체 테이블을 삭제하라.
--EMP1의 전체튜플을 제거하라.
--EMP1의 전체레코드를 제거하라
DELETE FROM EMP1;
SELECT * FROM EMP1;
```

```
--DDL?
--CREATE/DROP/TRUNCATE/ALTER
CREATE TABLE TEST1
(U_ID VARCHAR2(20),
U_DATE DATE);
DESC TEST1;
SELECT * FROM TEST1;
--기존의 테이블을 이용하여 새로운 테이블을 만드는 방법
CREATE TABLE T_STUDENT
AS
SELECT * FROM STUDENT
WHERE STU_DEPT='기계';
DESC T_STUDENT;
SELECT * FROM T_STUDENT;
--열내용을 추가하는 방법
ALTER TABLE T_STUDENT
ADD (ARMY CHAR(1));

DESC T_STUDENT;
SELECT * FROM T_STUDENT;
--열의 데이터타입을 변경하는 방법
ALTER TABLE T_STUDENT
MODIFY(ARMY NUMBER);

DESC T_STUDENT;
--열의 내용을 삭제하는 방법
ALTER TABLE T_STUDENT
DROP(ARMY);

DESC T_STUDENT;
SELECT * FROM T_STUDENT;
```

```
--열의 이름을 바꾸는 방법
ALTER TABLE T_STUDENT
RENAME COLUMN STU_NAME TO NAME;
--테이블 이름을 변경하는 방법
RENAME T_STUDENT TO TEST_STUDENT;
DESC T_STUDENT;
DESC TEST_STUDENT;

--테이블의 데이터 삭제하는 방법 (완전삭제)
TRUNCATE TABLE TEST_STUDENT;
DESC TEST_STUDENT;
SELECT * FROM TEST_STUDENT;
ROLLBACK;

--테이블삭제?
DROP TABLE TEST_STUDENT;
DESC TEST_STUDENT;

--CONSTRAINT(제약조건)_데이터의 신뢰성을 갖추기 위해서
--1.NOT NULL (NN)
--2.UNIQUE KEY (UK)
--3.PRIMARY KEY (PK)
--4.FOREIGN KEY (FK)
--5.CHECK (NN)
--NOT NULL CONSTRAINT CASE ?
CREATE TABLE T_STUDENT(
STU_NO CHAR(9),
STU_NAME VARCHAR2(12),
STU_DEPT VARCHAR2(20)
CONSTRAINT N_STU_DEPT NOT NULL,
STU_GRADE NUMBER(1),
STU_CLASS CHAR(1),
STU_GENDER CHAR(1),
```

```
STU_HEIGHT NUMBER(5,2),
STU_WEIGHT NUMBER(5,2));

--제약조건을 확인하는 방법
SELECT * FROM USER_CONSTRAINTS
WHERE TABLE_NAME='T_STUDENT';

DROP T_STUDENT;

CREATE TABLE T_STUDENT(
STU_NO CHAR(9),
STU_NAME VARCHAR2(12),
CONSTRAINT U_STU_NAME UNIQUE,
STU_DEPT VARCHAR2(20)
CONSTRAINT N_STU_DEPT NOT NULL,
STU_GRADE NUMBER(1),
STU_CLASS CHAR(1),
STU_GENDER CHAR(1),
STU_HEIGHT NUMBER(5,2),
STU_WEIGHT NUMBER(5,2);
SELECT * FROM USER_CONSTRAINTS
WHERE TABLE_NAME='T_STUDENT';

DROP TABLE T_STUDENT;

CREATE TABLE T_STUDENT(
STU_NO CHAR(9),
STU_NAME VARCHAR2(12)
CONSTRAINT U_STU_NAME UNIQUE,
STU_DEPT VARCHAR2(20)
CONSTRAINT N_STU_DEPT NOT NULL,
STU_GRADE NUMBER(1),
```

```
STU_CLASS CHAR(1),
STU_GENDER CHAR(1),
STU_HEIGHT NUMBER(5,2),
STU_WEIGHT NUMBER(5,2),
CONSTRAINT P_STU_NO PRIMARY KEY(STU_NO)
);

--PRIMARY KEY 를 동시에 두개를 할당하는 경우의 CASE임
CREATE TABLE T_ENROL(
SUB_NO CHAR(3),
STU_NO CHAR(9),
ENR_GRADE NUMBER(3),
CONSTRAINT P_ENROL PRIMARY KEY(SUB_NO,STU_NO)
);
SELECT *
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='T_ENROL';

CREATE TABLE T_SUBJECT
AS SELECT * FROM SUBJECT;

DROP TABLE T_ENROL;

CREATE TABLE T_ENROL(
SUB_NO NUMBER(3),
STU_NO CHAR(9),
ENR_GRADE NUMBER(3),
CONSTRAINT ENR_SUB_NO_FK1 FOREIGN KEY(SUB_NO) REFERENCES
T_SUBJECT(SUB_NO),
CONSTRAINT ENR_STU_NO_FK1 FOREIGN KEY(STU_NO) REFERENCES
T_STUDENT(STU_NO),
CONSTRAINT ENR_PK1 PRIMARY KEY(SUB_NO,STU_NO));
```

```
SELECT *
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='SUBJECT';
SELECT *
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='STUDENT';

DROP TABLE T_STUDENT;
--CHECK?
CREATE TABLE T_STUDENT(
STU_NO CHAR(9),
STU_NAME VARCHAR2(12)
CONSTRAINT U_STU_NAME UNIQUE,
STU_DEPT VARCHAR2(20)
CONSTRAINT N_STU_DEPT NOT NULL,
STU_GRADE NUMBER(1),
STU_GENDER CHAR(1)
CONSTRAINT C_STU_GENDER CHECK (STU_GENDER IN('M','F')),
STU_HEIGHT NUMBER(5,2),
STU_WEIGHT NUMBER(5,2),
CONSTRAINT P_STU_NO PRIMARY KEY(STU_NO));

SELECT * FROM USER_CONSTRAINTS
WHERE TABLE_NAME='SUBJECT';

--제약조건의 삭제방법?
--CASCADE[종속]: 두 테이블을 연결해서 pk를 가지고 있는 쪽의 값을 삭제하면
--FK로 연결된 값이 동시에 삭제되게 하는 옵션
ALTER TABLE T_ENROL
DROP CONSTRAINT ENR_SUB_NO_FK1 CASCADE;
```

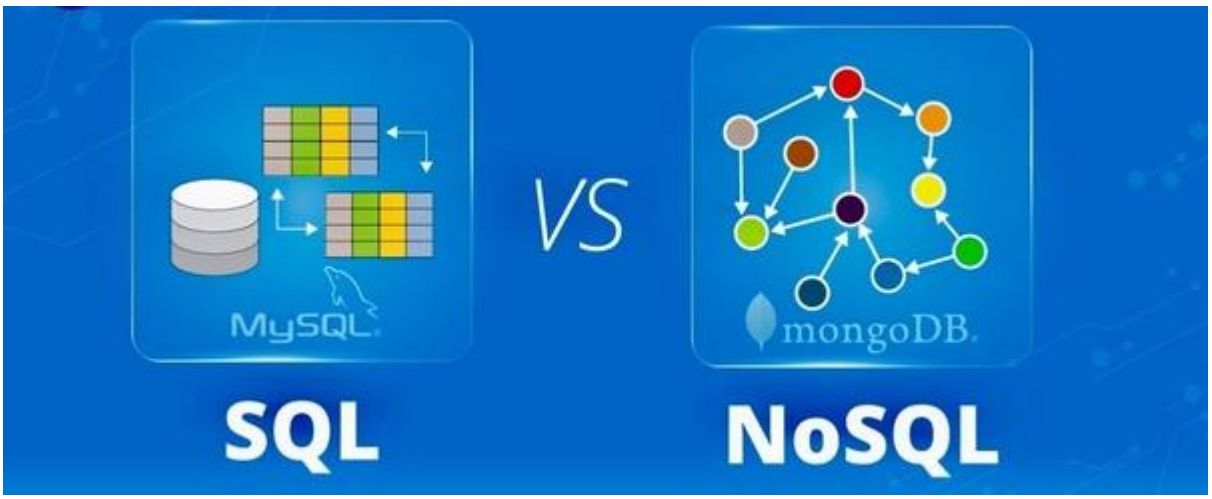
```
--제약조건의 비활성화/활성화?
ALTER TABLE T_STUDENT
DISABLE CONSTRAINT N_STU_DEPT;

ALTER TABLE T_STUDENT
ENABLE CONSTRAINT N_STU_DEPT;

CREATE TABLE T_SUBJECT
AS SELECT * FROM SUBJECT;

SELECT * FROM T_SUBJECT;

ALTER TABLE T_SUBJECT
ADD FOREIGN KEY REFERENCES SUB_NO;

--VIEW?
--단순뷰
CREATE OR REPLACE VIEW V_STUDENT1
AS
SELECT * FROM STUDENT
WHERE STU_DEPT='컴퓨터정보';
--조인뷰?
CREATE OR REPLACE VIEW V_ENROL1
AS
SELECT SUB_NAME,SUB_NO,STU_NO,ENR_GRADE
FROM ENROL NATURAL JOIN SUBJECT;
SELECT * FROM V_ENROL1;

--학과별 평균신장보다 큰 학생들의 학번,이름,신장을 검색하라
SELECT STU_NO,STU_NAME,A.STU_DEPT,STU_HEIGHT
FROM STUDENT A, (SELECT STU_DEPT,AVG(STU_HEIGHT) AS AVG_HEIGHT
FROM STUDENT GROUP BY STU_DEPT) B
WHERE A.STU_DEPT=B.STU_DEPT
```

AND A.STU_HEIGHT> B.AVG_HEIGHT;

DROP TABLE T_SUBJECT;

[Database] RDBMS와 NoSQL의 차이점

출처 :https://khj93.tistory.com/entry/Database-RDBMS%EC%99%80-NOSQL-%EC%B0%A8%EC%9D%B4%EC%A0%90



SQL vs NoSQL

이번 포스팅에서는 RDBMS와 NoSQL의 차이점을 알아보려고 합니다. 그전에 RDBMS는 무엇이고 왜 사용하며 NoSQL은 무엇이고 왜 사용을 할까요? 그리고 그 두 DB의 차이점은 무엇이며 서로에 대한 장단점은 무엇이 있을까요?

해당 설명을 하기 앞서 기본 용어들에 대해 정리를 하고 가겠습니다.

Database와 DBMS 그리고 SQL

Databse란 일반적으로 컴퓨터 시스템에 전자 방식으로 저장된 구조화된 정보 또는 데이터의 체계적인 집합을 의미합니다.

DBMS란(DataBase Management System) 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해 주고 데이터베이스를 관리해 주는 소프트웨어입니다.

SQL이란(Strucured Query Language) 관계형 데이터베이스 관리 시스템의 데이터를 관리하기 위해 설계된 특수 목적의 프로그래밍 언어이며 관계형 데이터베이스 관리 시스템에서 자료의 검색과 관리, 데이터베이스 스키마 생성과 수정, 데이터베이스 객체 접근 조정 관리를 위해 고안되었습니다.

RDBMS

위에서 DBMS는 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해주고 데이터베이스를 관리해 주는 소프트웨어라고 설명을 했습니다. 또한 기존의 RDBMS에서의 저장 방식은 SQL에 의해 저장되고 있으며 정해진 스키마에 따라 데이터를 저장하여야 합니다. RDBMS에는 DBMS앞에 R이 붙어 있습니다. 이 R은(Relational)의 약자로 RDBMS는 관계형 데이터베이스 관리 시스템을 의미합니다. 이름과 같이 RDBMS는 RDB를 관리하는 시스템이며 RDB는 관계형 데이터 모델을 기초로 두고 모든 데이터를 2차원 테이블 형태로 표현하는 데이터베이스입니다.

관계형 데이터베이스(RDMBS)는 아래와와 같이 구성된 테이블이 다른 테이블들과 관계를 맺고 모여있는 집합체로 이해할 수 있습니다.

관계형 데이터베이스(RDMBS)에서는 이러한 관계를 나타내기 위해 외래 키(foreign key)라는 것을 사용합니다.

이러한 테이블간의 관계에서 외래 키를 이용한 테이블 간 Join이 가능하다는 게 RDBMS의 가장 큰 특징입니다.

[회원 Table]

회원 번호 (Primary Key)	회원 이름	휴대폰 번호
1111111	김희진	010-xxxx-xxxx
2222222	김또깡	010-yyyy-yyyy

[주문 Table]

주문 번호 (Primary Key)	주문 회원 번호 (foreign key)	주문 상품
20200207xxxxxxxx	1111111	컴퓨터
20200207yyyyyyy	2222222	키보드
20200207zzzzzzz	2222222	마우스

NoSQL

NoSQL이란(Not Only SQL)의 약자로 말 그대도 위에서 설명한 RDB 형태의 관계형 데이터베이스가 아닌 다른 형태의 데이터 저장 기술을 의미하고 있습니다. 또한 NoSQL에서는 RDBMS와는 달리 테이블 간 관계를 정의하지 않습니다. 데이터 테이블은 그냥 하나의 테이블이며 테이블 간의 관계를 정의하지 않아 일반적으로 테이블 간 Join도 불가능합니다.

NoSQL은 점점 빅데이터의 등장으로 인해 데이터와 트래픽이 기하급수적으로 증가함에 따라 RDBMS에 단점인 성능을 향상시키기 위해서는 장비가 좋아야 하는 Scale-Up의 특징이 비용을 기하급수적으로 증가시키기 때문에 데이터 일관성은 포기하되 비용을 고려하여 여러 대의 데이터에 분산하여 저장하는 Scale-Out을 목표로 등장하였습니다.

NoSQL을 하면 가장 유명한 Document 기반의 MongoDB를 많이 떠올리지만 MongoDB는 NoSQL한 종류로 NoSQL은 하기와 같이 다양한 형태의 저장 기술을 지원하고 있습니다.

이 다양한 형태의 저장기술은 RDBMS 스키마에 맞추어 데이터를 관리해야 된다는 한계를 극복하고 수평적 확장성(Scale-out)을 쉽게 할 수 있다는 장점을 가지고 있습니다.

1. Key-Value Database

- Key-Value Database는 데이터가 Key와 Value의 쌍으로 저장된다. Key는 Value에 접근하기 위한 용도로 사용되며, 값은 어떠한 형태의 데이터라도 담을 수 있다. 심지어는 이미지나 비디오도 가능하다. 또한 간단한 API를 제공하는 만큼 질의의 속도가 굉장히 빠른 편이다.
- 대표적인 NoSQL Key-Value Model로는 Redis, Riak, Amazon Dynamo DB 등이 있다.

2. Document Database

- Documnet Database 데이터는 Key와Document의 형태로 저장된다. Key-Value 모델과 다른 점이라면 Value가 계층적인 형태인 도큐먼트로 저장된다는 것이다. 객체지향에서의 객체와 유사하며, 이들은 하나의 단위로 취급되어 저장된다. 다시 말해 하나의 객체를 여러 개의 테이블에 나눠 저장할 필요가 없어진다는 뜻이다.
- 주요한 특징으로는 객체-관계 매핑이 필요하지 않다. 객체를 Document의 형태로 바로 저장 가능하기 때문이다. 또한 검색에 최적화되어 있는데, 이는 Ket-Value 모델의 특징과 동일하다. 단점이라면 사용이 번거롭고 쿼리가 SQL과는 다르다는 점이다. 도큐먼트 모델에서는 질의의 결과가 JSON이나

xml 형태로 출력되기 때문에 그 사용 방법이 RDBMS에서의 질의 결과를 사용하는 방법과 다르다.

- 대표적인 NoSQL Document Model로는 MongoDB, CouthDB 등이 있다.

3. Wide Column Database

- Column-family Model 기반의 Database이며 이전의 모델들이 Key-Value 값을 이용해 필드를 결정했다면, 특이하게도 이 모델은 키에서 필드를 결정한다. 키는 Row(키 값)와 Column-family, Column-name을 가진다. 연관된 데이터들은 같은 Column-family 안에 속해 있으며, 각자의 Column-name을 가진다. 관계형 모델로 설명하자면 어트리뷰트가 계층적인 구조를 가지고 있는 셈이다. 이렇게 저장된 데이터는 하나의 커다란 테이블로 표현이 가능하며, 질의는 Row, Column-family, Column-name을 통해 수행된다.
- 대표적인 NoSQL Column-family Model로는 HBase, Hypertable 등이 있다.

4. Graph Database

- Graph Model Model에서는 데이터를 Node와 Edge, Property와 함께 그래프 구조를 사용하여 데이터를 표현하고 저장하는 Database입니다. 개체와 관계를 그래프 형태로 표현한 것이므로 관계형 모델이라고 할 수 있으며, 데이터 간의 관계가 탐색의 키일 경우에 적합하다. 페이스북이나 트위터 같은 소셜 네트워크에서(내 친구의 친구를 찾는 질의 등) 적합하고, 연관된

데이터를 추천해주는 추천 엔진이나 패턴 인식 등의 데이터베이스로도 적합하다.

- 대표적인 NoSQL Graph Model로는 Neo4J가 있다.

RDBMS와 NoSQL의 장단점

RDBMS

장점

- RDBMS는 위에서 설명을 하였듯이 정해진 스키마에 따라 데이터를 저장하여야 하므로 명확한 데이터 구조를 보장하고 있습니다.
- 또한 관계는 각 데이터를 중복없이 한 번만 저장할 수 있습니다.

단점

- 테이블간테이블 간 관계를 맺고 있어 시스템이 커질 경우 JOIN문이 많은 복잡한 쿼리가 만들어질 수 있습니다.
- 성능 향상을 위해서는 서버의 성능을 향상 시켜야하는 Scale-up만을 지원합니다. 이로 인해 비용이 기하급수적으로 늘어날 수 있습니다.
- 스키마로 인해 데이터가 유연하지 못합니다. 나중에 스키마가 변경 될 경우 번거롭고 어렵습니다.

NoSQL

장점

- NoSQL에서는 스키마가 없기 때문에 유연하며 자유로운 데이터 구조를 가질 수 있습니다. 언제든 저장된 데이터를 조정하고 새로운 필드를 추가할 수 있습니다.
- 데이터 분산이 용이하며 성능 향상을 위한 Saclue-up 뿐만이 아닌 Scale-out 또한 가능합니다.

단점

- 데이터 중복이 발생할 수 있으며 중복된 데이터가 변경 될 경우 수정을 모든 컬렉션에서 수행을 해야 합니다.
- 스키마가 존재하지 않기에 명확한 데이터 구조를 보장하지 않으며 데이터 구조 결정가 어려울 수 있습니다.

RDBMS, NoSQL 언제 사용해야 될까요?

RDBMS는 데이터 구조가 명확하며 변경 될 여지가 없으며 명확한 스키마가 중요한 경우 사용하는 것이 좋습니다. 또한 중복된 데이터가 없어(데이터 무결성) 변경이 용이하기 때문에 관계를 맺고 있는 데이터가 자주 변경이 이루어지는 시스템에 적합합니다.

NoSQL은 정확한 데이터 구조를 알 수 없고 데이터가 변경/확장이 될 수 있는 경우에 사용하는 것이 좋습니다. 또한 단점에서도 명확하듯이 데이터 중복이 발생할 수 있으며 중복된 데이터가 변경될 시에는 모든 컬렉션에서 수정을 해야 합니다. 이러한 특징들을 기반으로 Update가 많이 이루어지지 않는 시스템이 좋으며 또한 Scale-out이 가능하다는 장점을 활용해 막대한 데이터를 저장해야 해서 Database를 Scale-Out를 해야 되는 시스템에 적합합니다.

1. DML(Data Manipulation Language)

- RDBMS 내 테이블의 데이터를 저장,수정,삭제하는 명령어

>>INSERT,UPDATE,DELETE[WHERE]

2.DDL(Data Definition Language)

- RDBMS 내 데이터 관리를 위해 테이블을 포함한 여러 객체를 생성,수정,삭제하는 명령어

>>CREATE,ALTER,RENAME,TRUNCATE,DROP

3.TCL(Tracsaction Control Language)

- 트랜잭션 데이터의 영구 저장,취소 등과 관련된 명령어

>>COMMIT,ROLLBACK

4.DCL(Data Control Language)

- 데이터 사용 권한과 관련된 명령어

>>GRANT TO, ALTER USER , DROP USER , DROP USER CASCADE[종속]

5.DQL(Data Query Language)

- RDBMS에 저장한 데이터를 원하는 방식으로 조회하는 명령어

>> SELECT ,WHERE , GROUP BY [HAVING],JOIN,서브쿼리

6. DML _

6-1. INSERT?

--INSERT INTO T/N(C/N) *INSERT 구조*

--VALUES(VALUES)

--ENROL 테이블 조회

SELECT * FROM ENROL;

-구조확인

DESC ENROL;

-ENROL 테이블을 조회하여 20150000 보다 작은 수를 가져와서 A_ENROL 테이블을 생성

CREATE TABLE A_ENROL

```
AS

SELECT *

FROM ENROL

WHERE STU_NO <20150000;


DESC A_ENROL;

SELECT * FROM A_ENROL;


--데이터 생성

INSERT INTO A_ENROL(SUB_NO,STU_NO,ENR_GRADE)

VALUES(108,20151062,92);


INSERT INTO A_ENROL

VALUES(109,20152088,85);


SELECT * FROM A_ENROL;


INSERT INTO A_ENROL(SUB_NO,STU_NO)

VALUES(110,20152088);
```

```
SELECT * FROM A_ENROL;


INSERT INTO A_ENROL

VALUES(111,20153075,NULL);


SELECT * FROM A_ENROL;
```

6-2. 복수행 삽입?

```
--ENROL 테이블 조회

SELECT * FROM ENROL;

-- 학번이 2015로 시작하는 ENROL 테이블 조회

SELECT * FROM ENROL

WHERE STU_NO LIKE '2015%';

-- 학번이 2015로 시작하는 ENROL 테이블 조회하여 A_ENROL에 INSERT 함

INSERT INTO A_ENROL

SELECT * FROM ENROL

WHERE STU_NO LIKE '2015%';
```

```
SELECT * FROM A_ENROL;
```

6-3. UPDATE?

```
--UPDATE <T/N>

--SET CO--NAME=??

--WHERE ??

SELECT * FROM A_ENROL;


UPDATE A_ENROL

SET ENR_GRADE=ENR_GRADE+5;


UPDATE A_ENROL

SET ENR_GRADE=ENR_GRADE+5

WHERE SUB_NO=101;


--과목이름이 '시스템분석설계'인 그 과목만 점수를 10점 업데이트하라

UPDATE A_ENROL

SET ENR_GRADE=ENR_GRADE+10

WHERE SUB_NO=(SELECT SUB_NO
```

```
FROM SUBJECT

WHERE SUB_NAME='시스템분석설계');



SELECT SUB_NO

FROM SUBJECT

WHERE SUB_NAME='시스템분석설계';

SELECT * FROM A_ENROL;
```

6-4. DELETE?

```
--DELETE FROM <T/N>

--WHERE ???

DELETE FROM A_ENROL

WHERE STU_NO=20131001;



SELECT * FROM A_ENROL;
```

--A_ENROL테이블에서 과목이름이 기계요소설계인 과목번호를 가진 내용을

삭제하라

```
DELETE FROM A_ENROL
```

```
WHERE SUB_NO=(SELECT SUB_NO

FROM SUBJECT

WHERE SUB_NAME='기계요소설계');


SELECT SUB_NO

FROM SUBJECT

WHERE SUB_NAME='기계요소설계';
```

6-5.다중튜플삭제?

```
DELETE FROM A_ENROL;
```

7. TCL?

```
SELECT * FROM B_STUDENT;

DELETE FROM B_STUDENT;

SELECT * FROM B_STUDENT;


ROLLBACK;
```

```
SELECT * FROM B_STUDENT;

DELETE FROM B_STUDENT;

SELECT * FROM B_STUDENT;

CREATE TABLE C_STUDENT(STU_NO NUMBER,

STU_NAME CHAR(10));


ROLLBACK;


SELECT * FROM B_STUDENT;


SELECT * FROM A_STUDENT;

DELETE FROM A_STUDENT;

SELECT * FROM A_STUDENT;


ROLLBACK;
```

7-1. 병행처리?

```
SELECT *

FROM A_STUDENT;
```

```
INSERT INTO A_STUDENT(STU_NO,STU_NAME)

VALUES(10,'홍');

SELECT * FROM A_STUDENT;


COMMIT;
```

8. DDL?

--CREATE/DROP/TRUNCATE/ALTER

8-1. CREATE

```
CREATE TABLE TEST1

(U_ID VARCHAR2(20),

U_DATE DATE);


DESC TEST1;


SELECT * FROM TEST1;
```

8-2.기존의테이블을 이용하여 새로운 테이블을 만드는 방법

```
CREATE TABLE T_STUDENT

AS

SELECT * FROM STUDENT

WHERE STU_DEPT='기계';


DESC T_STUDENT;


SELECT * FROM T_STUDENT;
```

8-3.열내용을 추가하는 방법

```
ALTER TABLE T_STUDENT

ADD (ARMY CHAR(1));

DESC T_STUDENT;

SELECT * FROM T_STUDENT;
```

8-4.열의 데이터타입을 변경하는 방법

```
ALTER TABLE T_STUDENT

MODIFY(ARMY NUMBER);

DESC T_STUDENT;
```

8-5.열의 내용을 삭제하는 방법

ALTER TABLE T_STUDENT

DROP(ARMY);

DESC T_STUDENT;

SELECT * FROM T_STUDENT;

8-6.열의 이름을 바꾸는 방법

ALTER TABLE T_STUDENT

RENAME COLUMN STU_NAME TO NAME;

8-7.테이블 이름을 변경하는 방법

RENAME T_STUDENT **TO** TEST_STUDENT;

DESC T_STUDENT;

DESC TEST_STUDENT;

8-8.테이블의 데이터삭제하는방법(완전삭제)

TRUNCATE TABLE TEST_STUDENT;

DESC TEST_STUDENT;

SELECT * FROM TEST_STUDENT;

ROLLBACK;

8-9.테이블삭제?

DROP TABLE TEST_STUDENT;

DESC TEST_STUDENT;

8-10. CONSTRAINT(제약조건)

--1.NOT NULL (C)

--2.UNIQUE KEY (UK)

--3.PRIMARY KEY (PK)

--4.FOREIGN KEY (FK)

--5.CHECK (C)

8-10-1.NOT NULL CONSTRAINT CASE?

--T_STUDENT 테이블 생성 제약조건 STU_DEPT 에 걸기

```
CREATE TABLE T_STUDENT(
STU_NO CHAR(9),
STU_NAME VARCHAR2(12),
STU_DEPT VARCHAR2(20)
CONSTRAINT N_STU_DEPT NOT NULL,
STU_GRADE NUMBER(1),
STU_CLASS CHAR(1),
STU_GENDER CHAR(1),
STU_HEIGHT NUMBER(5,2),
STU_WEIGHT NUMBER(5,2));
```

8-10-2. 제약조건을 확인하는 방법

```
SELECT *
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='T_STUDENT';
```

```
DROP TABLE T_STUDENT;
--T_STUDENT 테이블을 UNIQUE ,NOT NULL 제약조건을 걸어준다.
```

--NOT NULL : 특정 열에 데이터의 중복 여부와는 상관없이 NULL의 저장을

허용하지 않는 제약 조건

--(반드시 열에 값이 존재해야만 하는 경우)

--UNIQUE : 열에 저장할 데이터의 중복을 허용하지 않고자 할 때 사용

```
CREATE TABLE T_STUDENT(
STU_NO CHAR(9),
STU_NAME VARCHAR2(12)
CONSTRAINT U_STU_NAME UNIQUE,
STU_DEPT VARCHAR2(20)
CONSTRAINT N_STU_DEPT NOT NULL,
STU_GRADE NUMBER(1),
STU_CLASS CHAR(1),
STU_GENDER CHAR(1),
STU_HEIGHT NUMBER(5,2),
STU_WEIGHT NUMBER(5,2));


SELECT *
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='T_STUDENT';
```

```
DROP TABLE T_STUDENT;


CREATE TABLE T_STUDENT(
STU_NO CHAR(9),
STU_NAME VARCHAR2(12)
CONSTRAINT U_STU_NAME UNIQUE,
STU_DEPT VARCHAR2(20)
CONSTRAINT N_STU_DEPT NOT NULL,
STU_GRADE NUMBER(1),
STU_CLASS CHAR(1),
STU_GENDER CHAR(1),
STU_HEIGHT NUMBER(5,2),
STU_WEIGHT NUMBER(5,2),
CONSTRAINT P_STU_NO PRIMARY KEY(STU_NO) );
```

8-10-3.PRIMARY KEY를 동시에 두개를 할당하는 경우의 CASE임.

```
CREATE TABLE T_ENROL(
SUB_NO CHAR(3),
```

```
STU_NO CHAR(9),

ENR_GRADE NUMBER(3),

CONSTRAINT P_ENROL PRIMARY KEY(SUB_NO,STU_NO));


SELECT *

FROM USER_CONSTRAINTS

WHERE TABLE_NAME='T_ENROL';


SELECT *

FROM USER_CONSTRAINTS

WHERE TABLE_NAME='SUBJECT';

SELECT *

FROM USER_CONSTRAINTS

WHERE TABLE_NAME='STUDENT';


DROP TABLE T_ENROL;

DESC STUDENT;

DESC SUBJECT;
```

```
CREATE TABLE T_SUBJECT

AS

SELECT *

FROM SUBJECT;



SELECT *

FROM USER_CONSTRAINTS

WHERE TABLE_NAME='SUBJECT';



SELECT * FROM T_SUBJECT;

SELECT * FROM SUBJECT;

DROP TABLE T_SUBJECT;



CREATE TABLE T_SUBJECT(

SUB_NO NUMBER(5),

SUB_NAME VARCHAR2(20),

SUB_PROF CHAR(10),

SUB_GRADE CHAR(5),

SUB_DEPT VARCHAR2(10),
```

CONSTRAINT P_SUB_NO **PRIMARY KEY**(SUB_NO));

```
CREATE TABLE T_ENROL(
SUB_NO NUMBER(5),
STU_NO VARCHAR2(9),
ENR_GRADE NUMBER(3),
CONSTRAINT ENR_SUB_NO_FK1 FOREIGN KEY(SUB_NO) REFERENCES
T_SUBJECT(SUB_NO),
--CONSTRAINT ENR_STU_NO_FK2 FOREIGN KEY(STU_NO) REFERENCES
STUDENT(STU_NO),
CONSTRAINT ENR_PK1 PRIMARY KEY(SUB_NO,STU_NO));
DROP TABLE T_ENROL;

SELECT *
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='T_ENROL';

--CHECK?
DROP TABLE T_STUDENT;
```

--CHECK: 열에 저장할 수 있는 값의 범위 또는 패턴을 정의할 때 사용

```
CREATE TABLE T_STUDENT(
STU_NO CHAR(9),
STU_NAME VARCHAR2(12)
CONSTRAINT U_STU_NAME UNIQUE,
STU_DEPT VARCHAR2(20)
CONSTRAINT N_STU_DEPT NOT NULL,
STU_GRADE NUMBER(1),
STU_GENDER CHAR(1)
CONSTRAINT C_STU_GENDER CHECK (STU_GENDER IN('M','F')),
STU_HEIGHT NUMBER(5,2),
STU_WEIGHT NUMBER(5,2),
CONSTRAINT P_STU_NO PRIMARY KEY(STU_NO));
SELECT * FROM USER_CONSTRAINTS
WHERE TABLE_NAME='T_STUDENT';
```

--제약조건의 비활성화/활성화?

```
ALTER TABLE T_STUDENT
DISABLE CONSTRAINT N_STU_DEPT;
```

```
ALTER TABLE T_STUDENT

ENABLE CONSTRAINT N_STU_DEPT;


--제약조건의 삭제방법?

ALTER TABLE T_ENROL

DROP CONSTRAINT ENR_SUB_NO_FK1 CASCADE;

SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME='T_ENROL';


--VIEW?

--단순뷰

CREATE OR REPLACE VIEW V_STUDENT1

AS

SELECT * FROM STUDENT

WHERE STU_DEPT='컴퓨터정보';

SELECT * FROM V_STUDENT1;

--조인뷰?

CREATE OR REPLACE VIEW V_ENROL1

AS

SELECT SUB_NAME,SUB_NO,STU_NO,ENR_GRADE
```

```
FROM ENROL NATURAL JOIN SUBJECT;

SELECT * FROM V_ENROL1;

₩

--학과별 평균신장보다 큰 학생들의 학번, 이름, 신장을 검색하라

--IN LINE VIEW

SELECT STU_NO,STU_NAME,A.STU_DEPT,STU_HEIGHT

FROM STUDENT A,(SELECT STU_DEPT,AVG(STU_HEIGHT) AS AVG_HEIGHT

FROM STUDENT GROUP BY STU_DEPT) B

WHERE A.STU_DEPT=B.STU_DEPT

AND A.STU_HEIGHT>B.AVG_HEIGHT;
```