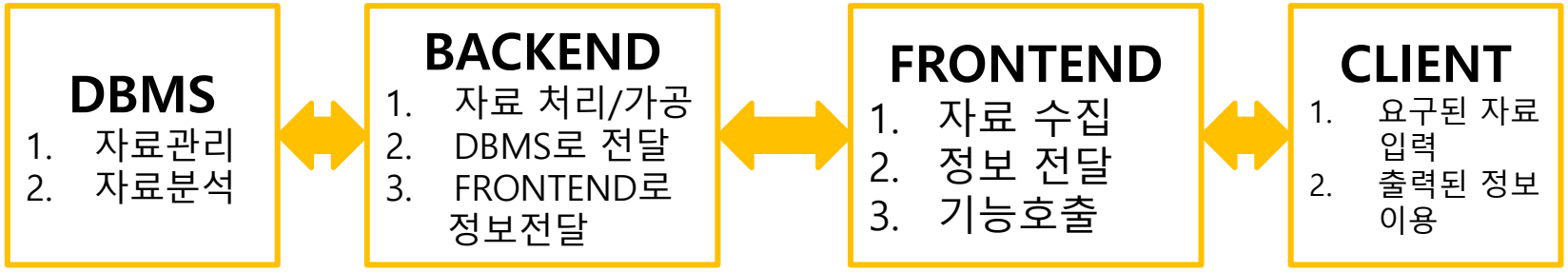

K G 아 이 티 뱅 크
JAVA
J A V A

PL / SQL

PL / SQL

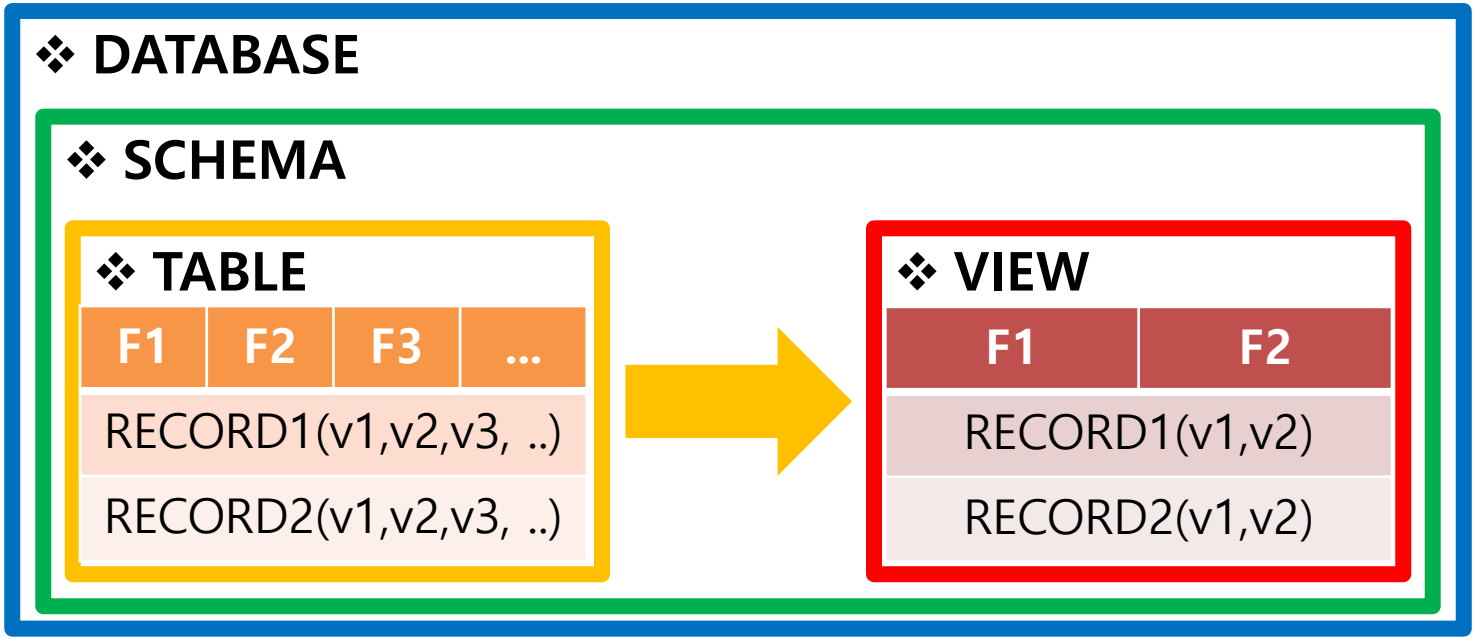
- **Oracle DBMS를 효율적으로 이용하기 위해 준비된 SQL**
  - 국제표준규격 SQL을 기반으로 하여 전용기능을 추가함
  - **DBMS별로 큰 틀은 비슷하니 세부사용방식은 다름**
- **DBMS는 대량의 자료(데이터)를 "관리"하는 전용 시스템**
  - 저장/검색/삭제/갱신/분석/무결성유지/보안/백업 등등...
  - 개발자가 자료를 다루는 부담을 줄여주는 것이 주 목적
- **DBMS는 자료를 적절하게 "처리/가공"하는 시스템이 아님**
  - 개발자는 DBMS에게 적절하게 변환된 자료를 제공해야 함



❖ **DBMS에는 수많은 개념이 존재하지만 크게 3가지로 구분됨**

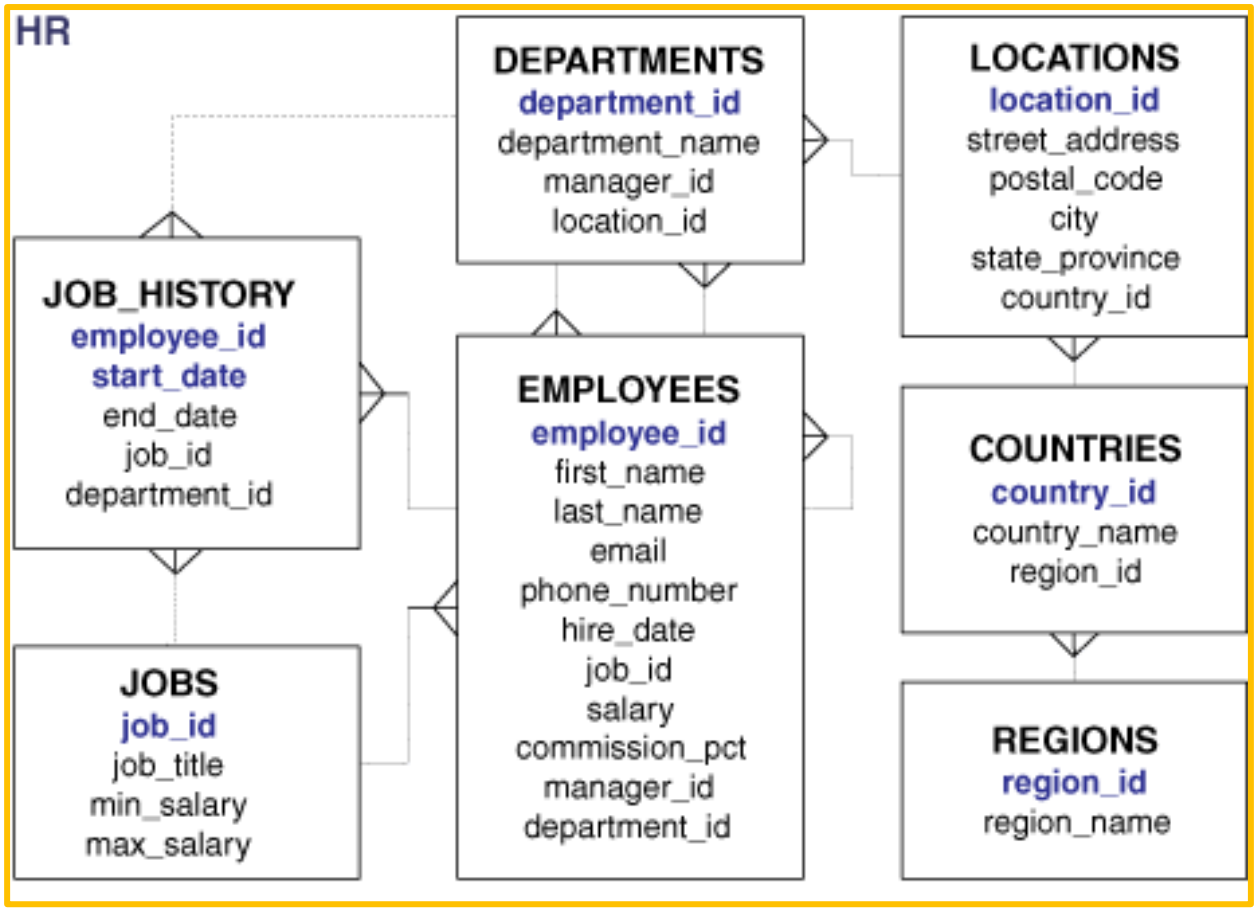
1. **스키마(schema)**
   - ➢ 데이터베이스의 전반적인 구조/설계를 정의하는 개념
2. **테이블(Table)**
   - ➢ 행과 열이 존재하며 레코드(값)와 필드(변수명)로 대응됨
3. **뷰(View)**
   - ➢ 데이터를 선택적으로 조회/검색한 가상의 테이블

❖ **데이터베이스를 잘 다루기 위해선 큰 그림을 파악해야만 함**



❖ 참고사항 : Sample DB인 Human Resource DB의 다이어그램

- ➢ 각 테이블에는 기준/기본이 되는 **필드**가 존재함
- ➢ 이를 **Candidate Key / Primary Key**라고 함



❖ **DBMS와 상호작용의 기초이자 핵심은 CRUD이고 구현해야 함**

- **C**reate : 데이터베이스에 새로운 데이터를 생성하는 과정
- **R**ead : 데이터베이스로부터 필요한 데이터를 읽는 과정
- **U**pdate : 데이터베이스의 데이터를 수정/갱신하는 과정
- **D**elete : 데이터베이스의 데이터를 삭제하는 과정

❖ **SQL로 작성된 코드는 쿼리문(Query)라고 하며 다르게 취급함**

- 기본적으로 "개발용" 언어가 아닌 "질의/요청용" 언어이니 주의
- 기본적인 키워드와 이에 대응되는 조합으로 구성되어 있음

```
-- Create : INSERT 문에 의하여 구현됨
INSERT INTO T_NAME(col1, col2, ...) values(val1, val2);
-- Read   : SELECT 문에 의하여 구현됨
SELECT * FROM T_NAME;
-- Update : UPDATE 문에 의하여 구현됨
UPDATE T_NAME SET col2 = value2 WHERE col1 = value1;
-- Delete : DELETE 문에 의하여 구현됨
DELETE FROM T_NAME WHERE col1 = value1;
```

핵심 쿼리문

❖ SELECT

- FROM절과 WHERE절을 조합해 요청한 정보를 반환하는 쿼리문
- 얼마만큼의 정보가 필요한지 필드를 지정하여 요청함
- FROM절은 어디(TABLE/VIEW)로부터 가져올 것인지를 지정함
- WHERE절은 조건과 일치하는 레코드를 가져오기 위해 사용함

```
-- 모든 레코드의 모든 필드를 table1로부터 가져올 때
SELECT * FROM table1;

-- 모든 레코드의 name,value필드만 table1로부터 가져올 때
SELECT name,value FROM table1;

-- name필드가 '고길동'인 레코드만 table2로부터 가져올 때
SELECT * FROM table2 WHERE name='고길동';

-- name이 'A'인 레코드의 val필드를 table3로부터 가져올 때
SELECT val FROM table3 WHERE name='A';
```

❖ INSERT

- INTO절과 VALUES절을 조합해 레코드의 저장요청하는 쿼리문
- 테이블을 지정시 필드명을 모두 적어주는 것이 기본원칙임
- VALUES는 지정한 필드명 순서로 값을 배치하여 전달함
- 오로지 추가용이며, 절대로 수정할 수 없으니 주의해야 함

```
-- table1에 '고길동', 34를 저장할 때
INSERT table1(name, age) VALUES ('고길동', 34);

-- 원하는 순서로 조정하여 맞출 수 있음
INSERT table1(age, name) VALUES (34, '고길동');

-- NULLABLE필드가 존재할 경우 순서를 반드시 조정해야 함
INSERT table1(name) VALUES ('고길동');

-- 필요에 따라 COMMIT 쿼리문을 이용해 즉시 반영함
INSERT table1(name) VALUES ('마이콜'); COMMIT;
```

❖ UPDATE

- SET절과 WHERE절로 일치하는 레코드를 전부 갱신하는 쿼리문
- SET을 통해 지정한 필드에 지정한 값으로 갱신처리를 수행함
- WHERE절은 AND/OR/NOT을 조합하여 정교하게 지정이 가능
- 일치하는 모든 레코드가 갱신되니 WHERE절이 굉장히 중요함

```
-- table1의 name이 '고길동'인 필드의 age를 36으로 변경
UPDATE table1 SET age=36 WHERE name='고길동';

-- table1의 age가 35인 모든 레코드의 val필드를 'C'로 변경
UPDATE table1 SET val='C' WHERE age=35;

-- table2의 v1이 1이고 v2가 2인 레코드의 v3를 4로 변경
UPDATE table2 SET v3=4 WHERE v1=1 AND v2=2;

-- table3의 v1이 1이거나 v2가 2인 레코드의 v3를 4로 변경
UPDATE table3 SET v3=4 WHERE v1=1 OR v2=2;
```

❖ DELETE

- FROM절과 WHERE절로 일치하는 레코드를 제거하는 쿼리문
- WHERE절은 안붙여도 되긴 하나, **절대 그런 짓을 하면 안됨**
- **WHERE절이 없으면 모든 레코드를 무차별 삭제하게 됨**
- 일치하는 모든 레코드가 삭제되니 WHERE절이 굉장히 중요함

```
-- 기업으로부터 억대단위 손해배상을 청구받는 명령어
DELETE FROM table1;

-- table3의 v1이 1인 모든 레코드를 제거함
DELETE FROM table3 WHERE v1=1;

-- 세미콜론으로 쿼리문이 끝나는 구조이니 여러줄로 작성가능
DELETE FROM table1
  WHERE name='고길동' AND age=35 AND value='A';
-- 굉장히 위험하기에 COMMIT여부를 신중하게 판단해야 함
COMMIT;
```

자주 사용되는 쿼리문

❖ CREATE

- VIEW, TABLE, 계정 생성등 다양하게 사용되는 쿼리문
- JOIN 쿼리문과 결합되어 적절한 VIEW 생성에 자주 사용됨
- VIEW는 TABLE에 직접 반영되지 않는 **"관측창"**같은 것
- 사용시에는 SELECT문과 조합하여 생성하게 됨

```
-- table1에서 모든 레코드 중 name만 보는  view1을 생성함
CREATE VIEW view1 AS SELECT name FROM table1;

-- table1에서 age가 9이상인 레코드만 보는 view2를 생성함
CREATE VIEW view2 AS SELECT * FROM table1 WHERE age>=9;

-- 세미콜론으로 쿼리문이 끝나는 구조이니 여러줄로 작성가능
CREATE VIEW view3 AS
  SELECT * from table3
  WHERE age>=30 AND salary=60000;
```

❖ JOIN

- ➢ 다른 테이블과 조합된 결과를 만드는데 사용되는 쿼리문
- ➢ 조합형태에 따라 INNER, LEFT, RIGHT, FULL OUTER 4종이 있음
  - ① **INNER JOIN** : 기본값 / 일치하는 것만 보여줌
  - ② **LEFT JOIN** : 없으면 붙여주는 테이블의 필드를 NULL처리
  - ③ **RIGHT JOIN** : 없을 경우 기준 테이블의 필드를 NULL처리
  - ④ **FULL OUTER JOIN** : 없으면 모든 테이블 필드를 NULL처리

```
SELECT t1.v1, t2.v2 FROM table1 t1

-- 하나의 테이블에는 하나만 가능하며, table1이 기준임
INNER JOIN table2 t2 ON t1.v3 = t2.v3;
LEFT JOIN table2 t2 ON t1.v3 = t2.v3;
RIGHT JOIN table2 t2 ON t1.v3 = t2.v3;
FULL OUTER JOIN table2 t2 ON t1.v3 = t2.v3;
```