SQL 활용

1. 기본 SQL 작성하기

1 - 1 테이블의 데이터 사전 조회

① 데이터 사전

1. 데이터 사전 개념

데이터 사전(Data Dictionary)에는 데이터베이스의 데이터(사용자 데이터)를 제외한 모든 정보(DBMS가 관리하는 데이터)가 있다. 데이터 사전의 내용을 변경하는 권한은 시스템 사용자(데이터베이스 관리자: DBA)가 가진다. 반면 일반 사용자에게는 단순 조회만 가능한 읽기 전용 테이블 형태가 제공된다.
데이터를 제외한(데이터를 구성하는) 모든 정보라는 점은 데이터의 데이터를 말한다. 따라서 데이터 사전은 메타 데이터(Meta data)로 구성되어 있음을 의미한다

2. 데이터 사전 내용

데이터 사전 안에 포함된 메타 데이터의 유형은 다음과 같다.

- 사용자 정보(ID, 비밀번호 및 권한 등)
- 데이터베이스 객체(테이블, 인덱스, 뷰 등)
- 무결성 제약 상태
- 함수, 프로시저 및 트리거 정보 등

데이터 사전에 포함된 정보가 메타 데이터라는 것은 모든 DBMS 제품에 공통이지만 데이터 사전을 만드는 방법, 관리하는 방법 등의 차이로 메타 데이터를 구성하는 내용은 제품마다 다르다.

3. 데이터 사전 활용

사용자에게 데이터 사전은 단순 조회의 대상이지만 데이터베이스 엔진을 이루는 파서, 옵티마이저와 같은 구성 요소에 대한 데이터 사전은 작업을 수행하는 데 필요한 참조 정보이다. 뿐만 아니라 작업을 수행할 대상이기도 하다

② 테이블에 대한 기본적인 데이터 사전 조회

1. Oracle에서 테이블의 데이터 사전 조회

Oracle 사용자는 뷰(View)로 데이터 사전에 접근할 수 있다. Oracle에서 뷰로 만들어진 데이터 사전은 오브젝트에 접근할 수 있는 사용자 권한에 따라 세 가지로 구분된다

DBA_ > ALL_ > USER_

Oracle에서는 이와 같은 지시자 뒤에 오브젝트 이름을 붙이는 형식으로 뷰의 이름이 정해진다.
여기서 오브젝트는 테이블, 칼럼, 제약 조건, 인덱스 등을 의미하므로 다음과 같은 영역별 조회문 구성이 가능하다

〈표 1-1〉 오* 데이터 사전 영역

영역	검색 범위	테이블 관련 데이터 사전	용도
DBA_	데이터베이스의 모든 객체 조회 가능 (DBA_는 시스템 접근 권한)	DBA_TABLES	모든 테이블 목록 확인
		DBA_TAB_COLUMNS	모든 테이블의 구성 칼럼 목록 확인
		DBA_TAB_COMMENTS	모든 테이블의 커멘트 확인
		DBA_CONSTRAINTS	모든 제약 조건 확인
ALL_	자신의 계정으로 접근할 수 있는 객체와 다른 계정에 접근 가능한 권한을 가진 모든 객체 조회 가능	ALL_TABLES	권한 있는 테이블 목록 확인
		ALL_TAB_COLUMNS	권한 있는 테이블의 구성 칼럼 목록 확인
		ALL_TAB_COMMENTS	권한 있는 테이블의 코멘트 확인
		ALL_CONSTRAINTS	권한 있는 제약 조건 확인
USER_	현재 자신의 계정이 소유한 객체 조회 가능	USER_TABLES	자기 계정의 테이블 목록 확인
		USER_TAB_COLUMNS	자기 계정의 테이블 구성 칼럼 목록 확인
		USER_TAB_COMMENTS	자기 계정의 테이블의 코멘트 확인
		USER_CONSTRAINTS	자기 계정의 제약 조건 확인

2. MY-SQL에서 테이블의 데이터 사전 조회

데이터 사전은 테이블 형태로 구성되어 있다. 따라서 테이블의 내용을 검색하기 위해서는 해당 테이블의 위치와 이름을 정확히 알고 있어야 한다. 여기서 위치는 데이터베이스를 의미한다. M*-SQL에서 데이터 사전은 Information_schema라는 데이터베이스 안에 존재한다. 따라서 이 안의 테이블을 조회하기 위해서는 우선 해당 데이터베이스로 이동해서 테이블 목록을 요청해야 한다.

- use Information_schema; -- 이동
- show tables; -- 테이블 목록 보기

테이블 목록으로 데이터 사전을 구성하는 테이블 이름을 확인하고, SELECT문을 통해 해당 테이블의 내용을 조회할 수 있다

① 데이터 정의(DDL문)

1. DDL 개념

DDL(Data Definition Language)은 영문 그대로 '데이터를 정의하는 언어'이다. 보다 직관적으로 설명하자면 '데이터를 담을 그릇을 생성하는 언어'이다. 이렇게 생성한 그릇을 DBMS에서는 오브젝트(Object)라고 한다. 다음과 같이 DDL을 통해 생성할 수 있는 오브젝트 대상을 확인한다

〈표 1-3〉 DDL 대상

DDL 대상	설명	비고
스키마(Schema)	- DBMS의 특징과 구축 환경에 기반한 데이터 구조 - 하나의 데이터베이스라고 이해할 수 있음	DBMS마다 차이
도메인(Domain)	- 속성의 데이터 타입과 제약 조건 등을 설정한 정보 - 속성이 가지는 값의 범위로 이해할 수 있음	예를 들어, 주소를 VARCHAR(120)로 정의
테이블(Table)	- 데이터 저장 공간	본 학습 대상
뷰(View)	- 1개 이상의 물리 테이블을 통해서 만드는 가상의 논리 테이블	학습 2-2 참조
인덱스(Index)	- 빠른 검색을 위한 데이터 구조	학습 2-2 참조

2. DDL 유형

오브젝트를 생성, 변경, 그리고 제거하기 위해 다음과 같은 명령어를 사용한다

〈표 1-4〉 DDL 명령어

구분	DDL 명령어	내용
생성	CREATE	오브젝트 생성
변경	ALTER	오브젝트 변경
삭제	DROP	오브젝트 삭제
	TRUNCATE	오브젝트 내용 삭제(데이터베이스 내의 로깅 작업 건너뜀)

DDL 명령어로 분류되지는 않지만 DDL과 같이 사용되는 명령어가 있다. 비상용 제품인 MYSQL의 경우, 생성된 오브젝트의 목록을 조회하기 위해서는 SHOW 명령문을 사용하며, 내용을 조회하기 위해서는 SELECT문을 사용한다. 상용 제품인 Oracle의 경우 SELECT로 목록과 내용을 조회한다.

3. DDL 작성

데이터베이스를 구성하기 위해 스키마, 도메인, 테이블, 뷰, 인덱스 등과 같은 오브젝트에 대한 DDL 작업이 필요하지만, 본 학습에서는 테이블만을 대상으로 한다

(1) 테이블 생성

테이블 생성을 위한 DDL 사용 방법은 다음과 같이 두 가지 종류로 구분할 수 있다.

〈표 1-5〉 테이블 생성 SQL문

구분	문법
신규 생성	CREATE TABLE 테이블명 (칼럼명 데이터 타입 [DEFAULT 값] [NOT NULL] {,칼럼명 데이터 타입 [DEFAULT 값] [NOT NULL] }* [PRIMARY KEY (칼럼 리스트),] {[FOREIGN KEY (칼럼 리스트) REFERENCES 테이블 명 [(칼럼명)] [ON DELETE 옵션] [ON UPDATE 옵션]], }* [CHECK (조건식) | UNIQUE(칼럼명)]) ;
다른 테이블 정보를 활용한 테이블 생성	CREATE TABLE 테이블명 AS SELECT문;

위와 같이 다른 테이블을 이용해서 신규 테이블을 생성하는 방법은 DBMS 제품마다 차이가 있다

(2) 테이블 변경

ALTER를 이용하여 테이블 구조를 변경하는 문법은 다음과 같다.

〈표 1-6〉 ALTER 이용한 테이블 변경 SQL문

구분	문법
열 추가	ALTER TABLE 테이블명 ADD 칼럼명 데이터 타입 [DEFAULT 값]
열 데이터 타입 변경	ALTER TABLE 테이블명 MODIFY 칼럼명 데이터 타입 [DEFAULT 값]
열 삭제	ALTER TABLE 테이블명 DROP 칼럼명

(3) 테이블 삭제, 절단, 이름 변경

DROP TABLE, TRUNCATE TABLE, RENAME TABLE 명령문을 사용하여 테이블 삭제, 테이블의 데이터 삭제, 테이블 이름을 변경할 수 있다. 테이블 존재 및 테이블 내용을 삭제하기 위한 명령어의 사용 문법은 다음과 같다.

〈표 1-7〉 테이블 삭제 및 이름 변경 방법

구분	문법
테이블 삭제	DROP TABLE 테이블명
테이블 내용 삭제	TRUNCATE TABLE 테이블명
테이블명 변경	RENAME TABLE 이전_테이블명 TO 새로운_테이블명
	ALTER TABLE 이전_테이블명 RENAME 새로운_테이블명

(4) 제약 조건 적용

다음과 같은 제약 조건을 테이블 생성 과정에 적용할 수 있다. CREATE 문을 통해 테이블 생성 과정에서 제약 조건을 설정할 수 있고, ALTER 문을 통해 이미 생성된 테이블의 제약 조건을 변경할 수 있다

〈표 1-8〉 테이블 생성에 사용되는 제약 조건

제약 조건	설명
PRIMARY KEY(PK)	- 테이블의 기본 키를 정의함 - 기본적으로 NOT NULL, UNIQUE 제약 사항이 설정됨
FOREIGN KEY(FK)	- 테이블에 외래 키를 정의함 - 참조 대상을 테이블명(칼럼명) 형식으로 작성해야 함 - 참조 무결성이 위배되는 상황 발생 시, 다음 옵션으로 처리 가능 (CASCADE, NO ACTION, SET NULL, SET DEFAULT)
UNIQUE	- 테이블에서 해당하는 열값은 유일해야 함을 의미함 - 테이블에서 모든 값이 다르게 적재되어야 하는 열에 설정함
NOT NULL	- 테이블에서 해당하는 열의 값은 NULL 불가능 - 필수적으로 입력해야 하는 항목에 설정함
CHECK	- 사용자가 직접 정의하는 제약 조건 - 발생 가능한 상황에 따라 여러 가지 조건을 설정 가능

② 다중 테이블 조회(DML문)

관계형 데이터베이스는 데이터의 중복을 최소화하기 위해 데이터를 분해하여 저장하고 통합하여 사용한다. 데이터를 분해하는 방법에는 정규화 기법이 있고, 데이터를 통합하는 방법에는 다중 테이블 검색 기법이 있다. 다중 테이블을 이용할 수 있는 세부적인 기법은 다음과 같다

〈표 1-9〉 다중 테이블 검색 방법

다중 테이블 검색 기법	내용
조인	두 개의 테이블을 결합하여 데이터를 추출하는 기법
서브쿼리	SQL문 안에 포함된 SQL문 형태의 사용 기법
집합 연산자	테이블을 집합 개념으로 조작하는 기법

1. 조인(JOIN)

(1) 조인 개념

조인은 영문 의미 그대로는 결합을 의미하며, 관계형 데이터베이스에서의 조인은 교집합 결과로 데이터 결합 방법을 의미한다. 교집합이 되는 공통점은 다양한 관점에서 정의될 수 있다. 여기서 그 관점을 정의하는 것이 바로 조인의 조건이 된다. 조인은 두 테이블의 공통값을 이용하여 칼럼을 조합하는 수단이다. 보통 PK와 FK값을 결합하여 사용하는 것이 일반적이다. 보다 엄밀하게 말하자면 PK, FK와 관계없이 논리적인 값들의 연관을 사용한다. 세 개 이상의 테이블에 대한 조인(결합)은 두 개의 테이블을 우선 조인하고 그 결과와 나머지 한 개의 테이블을 다시 조인한다

출처: 집필진 제작(2021)
[그림 1-5] 조인 개념도

(2) 조인 유형

조인은 관계형 데이터베이스의 가장 핵심 기능이자 큰 장점이라 할 수 있다. 조인은 크게 물리적 조인과 논리적 조인으로 구분할 수 있으며, 물리적 조인은 데이터베이스의 성능을 높이기 위한 튜닝 관점에서 다루는 주제로서, 본 학습에서는 사용자가 직접 제어할 수 있는 논리적 조인에 대해 알아본다.

〈표 1-10〉 조인 유형(대분류)

조인 분류	내용
논리적 조인	사용자가 작성한 SQL문에 의해서 표현한 테이블 결합 방식
물리적 조인	데이터베이스 관리시스템(DBMS)의 옵티마이저 엔진에 의해 발생하는 테이블 결합 방식 - Nested Loop Join - Merge Join - Hash Join

논리적 조인은 크게 내부 조인과 외부 조인으로 구분할 수 있으며, 각각의 세부 유형은 다음과 같이 구분할 수 있다

〈표 1-11〉 조인 유형(세분류)

조인 유형	내용
내부 조인 (Inner Join)	두 테이블에 공통으로 있는 칼럼(열)을 활용하는 유형(공통 칼럼 기반)
동등 조인 (Equi Join)	공통으로 있는 칼럼값이 같은 경우에 레코드 추출
자연 조인 (Natural Join)	두 테이블에 있는 동일한 칼럼명을 기준으로, 모든 칼럼값이 같은 경우에 레코드 추출
교차 조인 (Cross Join)	조인 조건이 없는 모든 데이터의 조합을 추출
외부 조인 (Outer Join)	특정한 테이블의 모든 데이터를 기준으로 다른 테이블의 정보와 비교하여 추출(단, 다른 테이블에 동일한 값이 없어도 특정한 테이블은 출력됨)
왼쪽 외부 조인 (Left Outer Join)	왼쪽 테이블의 모든 데이터와 오른쪽 테이블의 동일한 데이터를 추출(단, 오른쪽 테이블에 동일한 값이 없어도 왼쪽 테이블의 레코드는 출력됨)
오른쪽 외부 조인 (Right Outer Join)	오른쪽 테이블의 모든 데이터와 왼쪽 테이블의 동일 데이터를 추출(단, 왼쪽 테이블에 동일한 값이 없어도 오른쪽 테이블의 레코드는 출력됨)
완전 외부 조인 (Full Outer Join)	양쪽의 모든 데이터를 추출(단, 오른쪽과 왼쪽 테이블에 동일한 값이 없어도 오른쪽, 왼쪽 테이블의 레코드는 출력됨)

내부 조인에서 내부의 의미는 외부 조인에 대비하기 위해 사용되었으므로 수식어가 없는 조인은 내부 조인을 의미한다. 내부 조인의 세부 유형은 조인의 조건에 따라 세분된다. 명시적으로 지정하지 않고, 두 테이블의 컬럼명이 같은 값을 기준으로 하면 자연 조인, 특정 컬럼을 지정하면서 같은 값을 비교하면 동등 조인이며, 값이 다른 것을 비교하면 비동등 조인이 된다.

외부 조인은 기준 테이블이 참조 테이블과 조인하되, 참조 테이블에 조인할 데이터가 있는 경우는 참조 테이블 데이터를 함께 출력하고, 참조 테이블의 조인 데이터가 없는 경우에도 기준 테이블의 모든 데이터를 표시하고 싶은 경우에 사용한다. 보통 기준 테이블의 모든 값에 대해 참조 테이블의 데이터가 반드시 존재한다는 보장이 없는 경우 외부 조인을 사용한다.

2. 서브쿼리(Sub-Query)

(1) 서브쿼리 개념

서브쿼리는 다음과 같이 SQL문 안에 포함된 또 다른 SQL문을 말하며, 아직 확인되지 않은 기준을 위한 검색하는 용도로 사용한다

출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.14.
[그림 1-6] 서브쿼리 개념도

가장 밖에 위치한 메인쿼리와 그 안에 포함된 서브쿼리 관계는 주종(Master-Slave) 관계를 이루고 있다, 서브쿼리에 작성된 칼럼명은 메인쿼리의 칼럼명을 가져와서 사용할 수 있으나
그 역은 성립하지 않는다.

(2) 서브쿼리 유형

서브쿼리는 자동하는 방식과 반환되는 값의 형태에 따라 서브쿼리 종류를 각각 분류할 수 있다.
우선 동작하는 방식에 따른 서브쿼리의 종류는 다음과 같다

〈표 1-12〉 서브쿼리 유형 1

서브쿼리 종류	설명
비연관(Un-Correlated) 서브쿼리	서브쿼리 안에 메인쿼리의 칼럼 정보를 가지고 있지 않은 형태 (서브쿼리는 메인쿼리 없이 독자적으로 실행)
연관(Correlated) 서브쿼리	서브쿼리 안에 메인쿼리의 칼럼 정보를 가지고 있는 형태 (메인쿼리의 실행된 결과를 통해서 서브쿼리의 조건이 맞는지 확인)

반환되는 데이터 형태에 따른 서브쿼리의 종류는 다음과 같다.

〈표 1-13〉 서브쿼리 유형 2

서브쿼리 종류	설명
Single Row(단일 행) 서브쿼리	- 서브쿼리의 결과가 항상 1건 이하인 서브쿼리 - 단일 행의 비교 연산자에는 **=, 〈, 〈=, 〉, 〉=, 〈〉** 등이 사용
Multiple Row(다중 행) 서브쿼리	- 서브쿼리 실행 결과가 여러 건인 서브쿼리 - 다중 행의 비교 연산자에는 **IN, EXISTS, ALL, ANY** 등이 사용
Multiple Column(다중 칼럼) 서브쿼리	- 서브쿼리의 실행 결과가 2개 이상 칼럼으로 반환되는 쿼리 - 메인쿼리의 조건절에 다수 칼럼을 비교할 때, 서브쿼리와 메인쿼리에서 비교하는 칼럼 개수, 위치가 동일해야 함

3. 서브쿼리(Sub-Query)

① 집합 연산 개념

집합 연산자는 테이블을 하나의 집합이라고 가정하고, 두 테이블 간의 집합 연산자를 사용해 계산하는 방식이다. 집합 연산자는 2개 이상의 쿼리를 연결하여 하나로 통합해 주는 역할을 수행한다. 즉 집합 연산자는 다수의 쿼리 실행 결과를 하나의 결과로 만든다. 일반적으로 집합 연산자를 사용하는 사례는 서로 다른 테이블 간에 실행 결과를 하나로 합치려고 할 때와 같은 테이블에서 서로 다른 조건으로 쿼리를 실행하여 결과를 합치려고 할 때 사용할 수 있다

② 집합 연산 개념

테이블을 집합의 개념으로 보고 두 테이블 사이에 가능한 집합 연산은 다음과 같은 종류가 있다

〈표 1-14〉 집합 연산자 유형

집합 연산자	설명
UNION	2개 이상 SQL문의 실행 결과에 대한 중복을 제거한 합집합
UNION ALL	2개 이상 SQL문의 실행 결과에 대한 중복을 제거하지 않은 합집합
INTERSECTION	2개 이상 SQL문의 실행 결과에 대한 중복을 제거한 교집합
EXCEPT (MINUS)	선행 SQL문의 실행 결과와 후행 SQL문의 실행 결과 사이의 중복을 제거한 차집합(일부 DBMS는 MINUS로 사용)

③ 데이터 제어(DCL문)

1. DCL 개념

데이터베이스에서 데이터 외의 오브젝트를 조작하려고 하는 경우에 DCL 명령을 사용한다.
이는 Data Control Language라는 약자로, 데이터 제어 언어라고 한다.

〈표 1-15〉 DCL 조작 대상

오브젝트	목적	내용
사용자 권한	접근 통제	DBMS에 접근할 사용자를 등록하고, 특정 사용자에게 데이터베이스의 사용 권한을 부여하는 작업
트랜잭션	안전하고 무결한 거래 보장	동시 다발적으로 발생하는 다수 작업을 독립적이고 안전하게 처리하기 위한 데이터베이스 작업 단위

추가적으로 트랜잭션 제어를 위한 명령어 TCL(Transaction Control Language)라는 용어가 있다.
DCL과 TCL은 작업 대상이 상이하기 때문에 서로 다른 개념으로 구분하나, 제어 기능이라는 공통점으로 TCL은 DCL의 일부로 분류하기도 한다. 다음을 통해 유형에 따른 DCL 명령어를 확인한다

〈표 1-16〉 DCL 명령어

유형	명령어	용도
DCL	GRANT	데이터베이스 사용자 권한 부여
	REVOKE	데이터베이스 사용자 권한 회수
TCL	COMMIT	트랜잭션 확정
	ROLLBACK	트랜잭션 취소
	CHECKPOINT	복귀지점 설정

2. DCL 작성

데이터베이스에서 데이터 외의 오브젝트를 조작하려고 하는 경우에 DCL 명령을 사용한다. 이는 Data Control Language라는 약자로, 데이터 제어 언어라고 한다.

(1) 사용자 권한 부여

사용자 권한은 시스템 권한과 객체 권한으로 구분한다.

〈표 1-17〉 권한 부여 명령어 문법

권한	명령어 문법
시스템 권한	GRANT 권한-1, 권한-2 TO 사용자 계정
객체 권한	GRANT 권한-1, 권한-2 ON 객체명 TO 사용자 계정

(2) 사용자 권한 회수

GRANT 명령어에 대응하는 권한 회수/해지 명령어는 REVOKE이다.

〈표 1-18〉 권한 회수 명령어 문법

권한	명령어 문법
시스템 권한	REVOKE 권한-1, 권한-2 FROM 사용자 계정
객체 권한	REVOKE 권한-1, 권한-2 ON 객체명 FROM 사용자 계정

3. TCL 활용

(1) 트랜잭션 개념

트랜잭션은 '일을 처리하는 단위'를 의미한다. 더 나아가 트랜잭션의 다양한 관점은 다음과 같다

- 트랜잭션은 논리적인 연산 단위이다.
- 하나 이상의 SQL문이 포함된다.
- 트랜잭션은 거래를 의미한다.
- 거래의 모든 결과가 모두 반영되거나 모두 취소되어야 한다.
- 분해되지 않는 최소 단위이다.

(2) 트랜잭션 제어

트랜잭션을 제어한다는 것은 흐름의 구조를 바꾼다는 것이 아니라 트랜잭션의 결과를 수용하거나 취소하는 것을 의미한다. 이러한 작업을 수행하는 TCL 관련 명령어는 다음과 같다

〈표 1-19〉 TCL 명령어

명령어	내용	비고
COMMIT	작업거래 내용 확정	
ROLLBACK	작업거래 내용 취소	
CHECKPOINT	작업거래의 저장 시점 설정	ROLLBACK 위치 지정

그리고 사용자 계정, 객체 생성에 관련된 시스템 권한과 해당 객체에 대한 권한은 다음과 같다

〈표 1-20〉 권한 유형

구분	권한	내용
시스템 권한	CREATE USER	계정 생성
	DROP USER	계정 삭제
	DROP ANY TABLE	테이블 삭제
	CREATE SESSION	데이터베이스 접속
	CREATE TABLE	테이블 생성
	CREATE VIEW	뷰 생성
	CREATE SEQUENCE	시퀀스 생성
	CREATE PROCEDURE	함수 생성
객체 권한	ALTER	테이블 변경
	INSERT	데이터 조작
	DELETE	
	SELECT	
	UPDATE	
	EXECUTE	PROCEDURE 실행

2. 고급 SQL 작성하기

2 - 1 테이블 외의 데이터 사전 조회

① 테이블 외의 데이터 사전 검색

데이터 사전에는 데이터베이스 객체인 테이블 외에도 인덱스, 뷰에 대한 메타 데이터 (Meta data)를 확인할 수 있다. 테이블뿐만 아니라 인덱스와 뷰에 대한 데이터 사전도 단순히 조회만 가능하며, DBMS 제품에 따라서 데이터 사전의 조회 방식, 관리 방식에 차이가 있다.

1. Oracle에서 데이터 사전 검색

앞서 1-1에서 설명한 것과 같이 Oracle사용자는 뷰로 데이터 사전에 접근할 수 있다. Oracle에서 뷰로 만들어진 데이터 사전은 3가지 영역으로 구분한다

DBA_ > ALL_ > USER

테이블, 뷰, 인덱스에 대한 오브젝트의 데이터 사전은 다음과 같다.

〈표 2-1〉 오* 데이터 사전 영역

영역	검색 범위	테이블 관련 데이터 사전	용도
DBA_	데이터베이스의 모든 객체 조회 가능 (DBA_는 시스템 접근 권한 의미)	DBA_TABLES	모든 테이블 목록 확인
		DBA_INDEXES	모든 인덱스 목록 확인
		DBA_IND_COLUMNS	모든 인덱스의 구성 칼럼 확인
		DBA_VIEWS	모든 뷰 목록 확인
ALL_	자신의 계정으로 접근 가능한 객체와 타 계정의 접근 권한을 가진 모든 객체 조회 가능	ALL_TABLES	권한 있는 테이블 목록 확인
		ALL_INDEXES	권한 있는 인덱스 목록 확인
		ALL_IND_COLUMNS	권한 있는 인덱스의 구성 칼럼 확인
		ALL_VIEWS	권한 있는 뷰 목록 확인
USER_	현재 자신의 계정이 소유한 객체 조회 가능	USER_TABLES	자기 계정의 테이블 목록 확인
		USER_INDEXES	자기 계정의 인덱스 목록 확인
		USER_IND_COLUMNS	자기 계정의 인덱스 구성 칼럼 확인
		USER_VIEWS	자기 계정의 뷰 목록 확인

2. MY-SQL에서 데이터 사전 검색

MY-SQL에서 데이터 사전은 Information_schema라는 데이터베이스 안에 존재한다. 또는 SQL문에서 information_schema라는 데이터베이스를 지정하고 데이터 사전 목록을 조회할 수 있다

- use Information_schema; -- information_schema로 이동
- show tables; -- 데이터 사전 목록 보기

또는

select * from information_schema.tables

테이블 목록으로 데이터 사전을 구성하는 테이블 이름을 확인하고, SELECT문을 통해 해당 테이블의 내용을 조회할 수 있다. 테이블과 인덱스, 뷰에 관련된 데이터 사전은 다음과 같다

〈표 2-2〉 M*-SQL 데이터 사전 영역

데이터베이스 명	데이터 사전 테이블	용도
information_schema	TABLES	데이터베이스에 존재하는 테이블 정보
	TABLE_CONSTRAINTS	테이블에 설정된 제약 사항 정보
	KEY_COLUMN_USAGE	제약 사항을 가지고 있는 키 칼럼에 대한 정보
	COLUMNS	테이블을 구성하는 칼럼에 대한 정보
	VIEWS	DB에 있는 뷰에 대한 정보
	STATISTICS	테이블의 인덱스에 대한 정보

2 – 2 인덱스와 뷰 작성

① 인덱스 활용

1. 인덱스 개념

인덱스는 저장된 데이터를 빠르게 검색할 수 있는 수단이자, 테이블에 대한 조회 속도를 높여 주는 자료 구조를 말한다. 다음 그림과 같은 인덱스(이름 칼럼)는 테이블에 있는 특정 레코드 위치를 알려주는 수단으로 사용하며, 사용자에 의해서 인덱스가 생성되어야 활용할 수 있다

출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.42.
[그림 2-6] 인덱스 개념

반면 PK 칼럼(일련번호 칼럼)으로 만들어진 PK 인덱스는 자동으로 생성된다. 요약하면 PK 칼럼은 기본 키를 생성할 때 데이터베이스에 의해서 자동으로 PK 인덱스가 생성된다

예를 들어 위의 [그림 2-6]과 같은 테이블에서 일련번호를 기본 키(Primary Key)로 하는 경우, 일련번호에 대한 인덱스는 자동으로 생성되나, 생년월일이나 이름을 기준으로 하는 인덱스는 자동으로 생성되지 않는다. 다음 질의문을 보자

```
select * from table_great_men where 이름 = '이순신'
```

조건문 where 절에서 `이름'을 비교하고 있다. 이 경우 해당 테이블의 `이름' 칼럼에 인덱스가 없는 경우, 테이블의 전체 내용을 검색(Full Table scan)하게 된다. 반면, 인덱스가 생성되어 있다면 테이블의 일부분을 검색(Unique Scan 또는 Range scan)하여 데이터를 빠르게 찾을 수 있다. 조건절에 `='로 비교되는 칼럼을 대상으로 인덱스를 생성하면 검색 속도를 높일 수 있다. 하지만 자동으로 생성되지 않기 때문에 DB 사용자는 질의문을 분석하여 인덱스를 생성해야 한다

2. 인덱스 사용

(1) 인덱스 사용 주체

[그림 2-7]에서 `이름' 칼럼에 대한 인덱스가 생성되어 있다면 데이터를 빠르게 찾을 수 있다. 이 때 DBMS는 인덱스를 사용하여 빠른 검색을 수행한다. 이를 위해 DB사용자는 DBMS가 인덱스를 사용할 수 있게 준비해 주어야 한다. 따라서 사용자 입장에서는 인덱스를 사용한다는 개념보다는 검색 속도를 향상시키기 위해 준비한다는 개념으로 DBMS에 접근해야 한다

(2) 인덱스 준비

DB 사용자가 인덱스에 대해 조작할 수 있는 방법으로는 '생성, 삭제, 그리고 변경' 조작이 있다. 참고로 인덱스를 사용하는 명령어는 SQL 표준화에 포함되지 않으므로 DBMS 제품 공급사마다 약간의 차이가 있다.

(가) 인덱스 생성

인덱스 생성 문법은 다음과 같다.

```
CREATE [UNIQUE] INDEX <인덱스명> ON <테이블명> (<칼럼명 나열>);
```

여기서 각각의 파라미터가 의미하는 내용은 다음과 같다.

〈표 2-4〉 인덱스 명령문 요소

파라미터	내용	비고
[UNIQUE]	인덱스 걸린 칼럼에 중복값을 허용하지 않음 (생략 가능)	CREATE 테이블에서 사용하는 UNIQUE 제약 조건과 동일한 의미
〈인덱스명〉	생성하고자 하는 인덱스 이름을 작성	
〈테이블명〉	인덱스 대상인 테이블 이름을 작성	
〈칼럼명 나열〉	인덱스 대상 테이블의 특정 칼럼 이름(들)	복수 칼럼 지정 가능

(나) 인덱스 삭제

인덱스 삭제 명령 형식은 다음과 같다.

DROP INDEX <index name>;

<index name>은 생성된 인덱스 이름을 의미한다. 인덱스 관련 명령어에 대한 SQL 표준이 없기에 제품별 Drop 명령문의 사용법은 약간씩 다르다. 보통 인덱스를 테이블의 종속 구조로 생각하여 인덱스를 삭제하기 위해 테이블에 변경을 가하는 형식의 명령을 사용한다.
즉, ALTER TABLE 명령 뒤에 DROP INDEX 명령이 추가되는 형태로 사용된다.

(다)인덱스 변경

인덱스에 대한 정의를 변경하는 명령문 형식은 다음과 같다..

ALTER [UNIQUE] INDEX <index name> ON <table name> (<column(s)>);

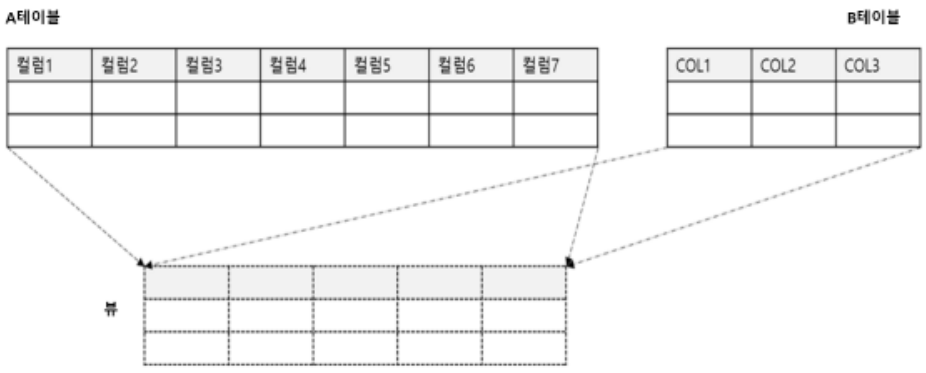
한 번 생성된 인덱스에 대해 변경이 필요한 경우는 드물다. 또 인덱스 관련 SQL문은 표준화가 안 되었으므로 인덱스 변경에 대한 명령문 지원 여부 및 방법은 벤더별로 다르다. 일부 제품은 인덱스에 대한 변경 SQL문이 없다. 이 경우 기존 인덱스를 삭제하고 신규 인덱스를 생성하는 방식으로 사용이 권고되고 있다.

② 뷰(View) 개요

1. 뷰 개념

뷰는 기본 테이블로부터 유도된 가상 테이블로서, 사용자에게(생성 관점 아닌 사용 관점에서)는 물리적으로 존재하지 않지만 테이블로 있는 것처럼 간주된다. 아래 [그림 2-7]은 물리 테이블인 'A테이블'과 'B테이블'을 활용하여 뷰를 만드는 모습을 나타내고 있다. 뷰는 'A테이블'과 같은 하나의 물리 테이블로부터 생성 가능하며, 다수의 테이블 또는 다른 뷰를 이용해 만들 수 있다. 위에서 생성한 뷰는 다음 장에서 학습할 조인 (Join) 기능을 사용하며, 이미 만들어진 뷰가 있다면 조인 없이 하나의 테이블을 대상으로 질의하는 것처럼 SQL문을 작성하면 된다

SELECT * FROM <View Name>;



출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.44.
[그림 2-7] 뷰 개념도

즉 FROM 절에 작성한 <뷰>를 통해 뷰를 만들 때 활용한 다수의 <테이블>을 대체할 수 있다.
또 이러한 기능을 통해 테이블의 중요 데이터 일부만을 제공할 수 있는 등 다음과 같은 장점이 있다

〈표 2-5〉 뷰의 장점

뷰 장점	내용
논리적 독립성 제공	뷰는 논리 테이블(테이블의 구조가 변경되더라도 뷰를 활용하는 응용 프로그램도 항상 수정할 필요는 없음)
사용자 데이터 관리 용이	다수 테이블에 있는 다양한 데이터에 대해 단순한 질의어 활용 가능
데이터 보안 용이	중요한 보안 데이터가 있는 테이블은 접근하지 못하도록 설정하고, 접근이 가능한 일부 데이터만을 조회할 수 있도록 뷰를 생성함

반면에 다음과 같은 단점이 있다.

〈표 2-6〉 뷰의 단점

뷰 단점	내용
뷰의 인덱스 사용 불가	논리적 생성한 뷰에는 인덱스를 만들 수 없음
뷰 구조 변경 불가	뷰는 삭제 후 재생성을 통해서 뷰의 구조 변경이 가능함
데이터 변경 제약 존재	뷰로 조회된 데이터에 대한 삽입, 변경, 삭제 제약이 있음.

뷰의 단순한 사용은 생성 과정에 의존적이며 뷰를 어떻게 생성했느냐에 따라 사용 방법이 달라진다.

2. 뷰 생성

뷰 생성 명령의 일반 형태는 다음과 같다

CREATE VIEW <뷰 이름>(칼럼 목록) AS <뷰를 통해 보여줄 데이터 조회용 쿼리문>

〈표 2-7〉 뷰 생성 방법

상황	뷰 생성 쿼리문
테이블A 그대로	CREATE VIEW 뷰A AS select * from 테이블A;
테이블A 일부 칼럼	CREATE VIEW 뷰X AS select 칼럼1, 칼럼2, 칼럼3 from 테이블A;
테이블A와 테이블B 조인 결과	CREATE VIEW 뷰Y AS select * from 테이블A a, 테이블B b where a.칼럼1=b.COL1;

조회문의 다양한 변형에 따라 뷰의 내용이 달라진다. [그림 2-7]의 경우 '테이블A'와 '테이블B'로부터 조회 가능한 형태 모두가 뷰로 치환될 수 있다.

3. 뷰 삭제 및 변경

뷰의 구조(정의)를 변경하는 것은 불가능하다. 뷰의 구조가 만들어졌으면, 물리적인 요소는 뷰 이름과 뷰를 조회하기 위한 쿼리문만 해당된다. 이때 뷰의 이름이나 쿼리문을 변경하는 수단은 제공되지 않고 이 경우 뷰의 삭제와 재생성을 통해 뷰에 대한 정의 변경이 가능하다.

DROP VIEW <View Name>;

4. 뷰 내용 변경

뷰를 통해 접근 가능한 데이터에 대한 변경이 가능하다. 하지만 모든 경우에 데이터의 변경이 가능한 것이 아닌 일부 제약이 존재하며 이러한 제약은 뷰가 논리적으로 생성된 가상 테이블이기 때문에 물리적 상황에 의존적임을 의미한다. 예를 들어 PK(기본 키)에 해당하는 칼럼이 뷰에 정의되어 있지 않다면 INSERT는 당연히 불가 능하다.