

대분류/20  
정보통신

중분류/01  
정보기술

소분류/02  
정보기술개발

세분류/04  
DB엔지니어링

능력단위/13

NCS학습모듈

# SQL 활용

LM2001020413\_16v3



교육부

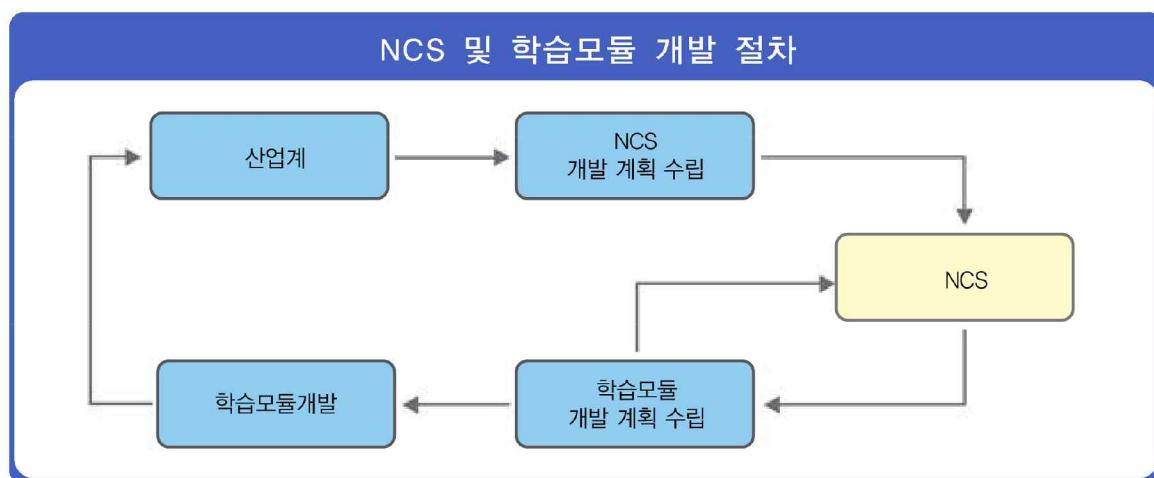
NCS 학습모듈은 교육훈련기관에서 출처를 명시하고 교육적 목적으로 활용할 수 있습니다. 다만 NCS 학습모듈에는 국가(교육부)가 저작재산권 일체를 보유하지 않은 저작물들(출처가 표기되어 있는 도표, 사진, 삽화, 도면 등)이 포함되어 있으므로 이러한 저작물들의 변형, 복제, 공연, 배포, 공중 송신 등과 이러한 저작물들을 활용한 2차 저작물의 생성을 위해서는 반드시 원작자의 동의를 받아야 합니다.

# NCS 학습모듈의 이해

\* 본 학습모듈은 「NCS 국가직무능력표준」 사이트(<http://www.ncs.go.kr>)에서 확인 및 다운로드 할 수 있습니다.

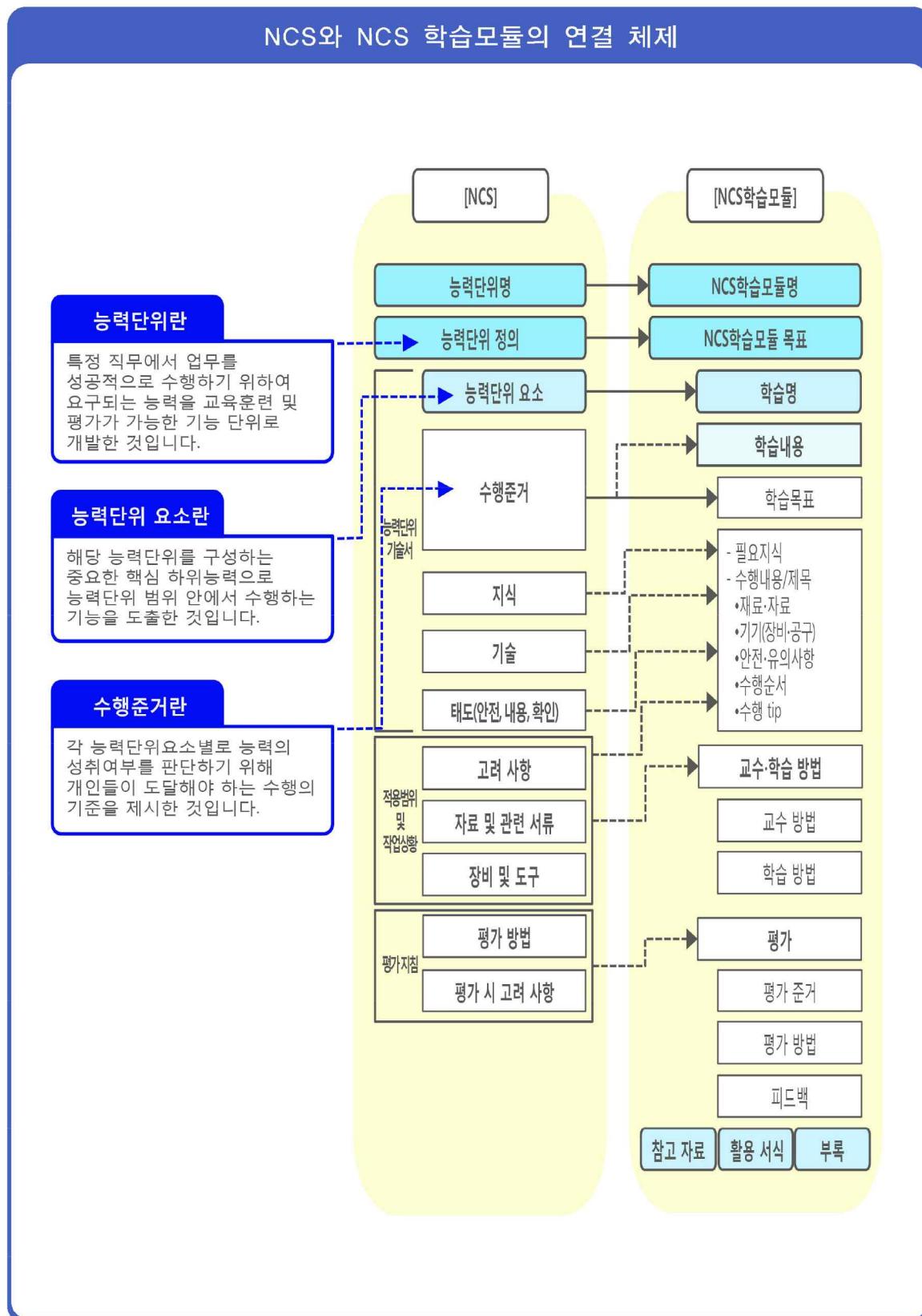
## (1) NCS 학습모듈이란?

- 국가직무능력표준(NCS: National Competency Standards)이란 산업현장에서 직무를 수행하기 위해 요구되는 지식·기술·소양 등의 내용을 국가가 산업부문별·수준별로 체계화한 것으로 산업현장의 직무를 성공적으로 수행하기 위해 필요한 능력(지식, 기술, 태도)을 국가적 차원에서 표준화한 것을 의미합니다.
- 국가직무능력표준(이하 NCS)이 현장의 ‘직무 요구서’라고 한다면, NCS 학습모듈은 NCS의 능력단위를 교육훈련에서 학습할 수 있도록 구성한 ‘교수·학습 자료’입니다. NCS 학습모듈은 구체적 직무를 학습할 수 있도록 이론 및 실습과 관련된 내용을 상세하게 제시하고 있습니다.



- NCS 학습모듈은 다음과 같은 특징을 가지고 있습니다.
- 첫째, NCS 학습모듈은 산업계에서 요구하는 직무능력을 교육훈련 현장에 활용할 수 있도록 성취목표와 학습의 방향을 명확히 제시하는 가이드라인의 역할을 합니다.
- 둘째, NCS 학습모듈은 특성화고, 마이스터고, 전문대학, 4년제 대학교의 교육기관 및 훈련기관, 직장교육기관 등에서 표준교재로 활용할 수 있으며 교육과정 개편 시에도 유용하게 참고할 수 있습니다.

- NCS와 NCS 학습모듈 간의 연결 체계를 살펴보면 아래 그림과 같습니다.



## (2) NCS 학습모듈의 체계

- NCS 학습모듈은 1.학습모듈의 위치, 2.학습모듈의 개요, 3.학습모듈의 내용 체계, 4.참고 자료, 5.활용 서식/부록으로 구성되어 있습니다.

### 1. NCS 학습모듈의 위치

- NCS 학습모듈의 위치는 NCS 분류 체계에서 해당 학습모듈이 어디에 위치하는지를 한 눈에 볼 수 있도록 그림으로 제시한 것입니다.

예시 : 이 · 미용 서비스 분야 중 네일미용 세분류

### NCS-학습모듈의 위치

대분류	이용 · 숙박 · 여행 · 오락 · 스포츠
중분류	이 · 미용
소분류	이·미용 서비스

#### 세분류

헤어미용	능력단위	학습모듈명
피부미용	네일 샵 위생 서비스	네일숍 위생서비스
메이크업	네일 화장을 제거	네일 화장을 제거
<b>네일미용</b>	<b>네일 기본 관리</b>	<b>네일 기본관리</b>
이용	네일 랩	네일 랩
	네일 팁	네일 팁
	젤 네일	젤 네일
	아크릴릭 네일	아크릴 네일
	평면 네일아트	평면 네일아트
	융합 네일아트	융합 네일아트
	네일 샵 운영관리	네일숍 운영관리

#### 학습모듈은

NCS 능력단위 1개당 1개의 학습모듈 개발을 원칙으로 합니다. 그러나 필요에 따라 고용 단위 및 교과단위를 고려하여 능력단위 몇 개를 묶어서 1개의 학습모듈로 개발할 수 있으며, NCS 능력단위 1개를 여러 개의 학습 모듈로 나누어 개발할 수도 있습니다.

## 2. NCS 학습모듈의 개요



### 구성

- NCS 학습모듈 개요는 학습모듈이 포함하고 있는 내용을 개략적으로 설명한 것으로서 **학습모듈의 목표**, **선수 학습**, **학습모듈의 내용 체계**, **핵심 용어**로 구성되어 있습니다.

#### 학습모듈의 목표

해당 NCS 능력단위의 정의를 토대로 학습목표를 작성한 것입니다.

#### 선수 학습

해당 학습모듈에 대한 효과적인 교수·학습을 위하여 사전에 이수해야 하는 학습모듈, 학습 내용, 관련 교과목 등을 기술한 것입니다.

#### 학습모듈의 내용 체계

해당 NCS 능력단위요소가 학습모듈에서 구조화된 방식을 제시한 것입니다.

#### 핵심 용어

해당 학습모듈의 학습 내용, 수행 내용, 설비·기자재 등 가운데 핵심적인 용어를 제시한 것입니다.



### 활용 안내

예시 : 네일미용 세분류의 ‘네일 기본관리’ 학습모듈

#### 네일 기본관리 학습모듈의 개요

##### 학습모듈의 목표

고객의 네일 보호와 미적 요구 충족을 위하여 효과적인 네일 관리로 프리에지 형태 만들기, 큐티를 정리하기, 컬러링하기, 보습제 도포하기, 마무리를 할 수 있다.

##### 선수학습

네일숍 위생서비스(LM1201010401\_14V2)

##### 학습모듈의 내용체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 프리에지 형태 만들기	1-1. 네일 파일에 대한 이해와 활용 1-2. 프리에지 형태 파일링	1201010403_12v2.1	프리에지 모양 만들기
2. 큐티를 정리하기	2-1. 네일 기본관리 매뉴얼 이해 2-2. 큐티를 관리	1201010403_14v2.2	큐티를 정리하기
3. 컬러링하기	3-1. 컬러링 매뉴얼 이해 3-2. 컬러링 방법 선정과 작업 3-3. 웰 컬러링 작업	1201010403_14v2.3	컬러링
4. 보습제 도포하기	4-1. 보습제 선정과 도포 4-2. 각질제거	1201010403_14v2.4	보습제 바르기
5. 네일 기본관리 마무리하기	5-1. 유분기 제거 5-2. 네일 기본관리 마무리와 정리	1201010403_14v2.5	마무리하기

##### 학습모듈의 목표는

학습자가 해당 학습모듈을 통해 성취해야 할 목표를 제시한 것으로, 교수자는 학습자가 학습모듈의 전체적인 내용흐름을 파악할 수 있도록 지도하는 것이 필요합니다.

##### 선수 학습은

교수자나 학습자가 해당 모듈을 교수 또는 학습하기 이전에 이수해야 할 학습내용, 교과목, 핵심 단어 등을 표기한 것입니다. 따라서 교수자는 학습자가 개별 학습, 자기 주도 학습, 방과 후 활동 등 다양한 방법을 통해 이수할 수 있도록 지도하는 것이 필요합니다.

##### 핵심 용어는

학습모듈을 통해 학습되고 평가되어야 할 주요 용어입니다. 또한 당해 모듈 또는 타 모듈에서도 핵심 용어를 사용하여 학습내용을 구성할 수 있으며, 「NCS 국가 직무능력표준」사이트([www.ncs.go.kr](http://www.ncs.go.kr))에서 색인(찾아보기) 중 하나로 이용할 수 있습니다.

##### 핵심 용어

프리에지, 니퍼, 퓨셔, 폴리시, 네일 파일, 스웨어형, 스웨어 오프형, 라운드형, 오발형, 포인트형

### 3. NCS 학습모듈의 내용 체계



- NCS 학습모듈의 내용은 크게 **학습**, **학습 내용**, **교수·학습 방법**, **평가**로 구성되어 있습니다.

#### 학습

해당 NCS 능력단위요소 명칭을 사용하여 제시한 것입니다.

학습은 크게 학습 내용, 교수·학습 방법, 평가로 구성되며 해당 NCS 능력단위의 능력단위 요소별 지식, 기술, 태도 등을 토대로 학습 내용을 제시한 것입니다.

#### 학습 내용

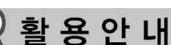
학습 내용은 학습 목표, 필요 지식, 수행 내용으로 구성하였으며, 수행 내용은 재료·자료, 기기(장비·공구), 안전·유의 사항, 수행 순서, 수행 tip으로 구성한 것입니다. 학습모듈의 학습 내용은 업무의 표준화된 프로세스에 기반을 두고 실제 산업현장에서 이루어지는 업무활동을 다양한 방식으로 반영한 것입니다.

#### 교수·학습 방법

학습 목표를 성취하기 위한 교수자와 학습자 간, 학습자와 학습자 간의 상호작용이 활발하게 일어날 수 있도록 교수자의 활동 및 교수 전략, 학습자의 활동을 제시한 것입니다.

#### 평가

평가는 해당 학습모듈의 학습 정도를 확인할 수 있는 평가 준거, 평가 방법, 평가 결과의 피드백 방법을 제시한 것입니다.



예시 : 네일미용 세분류의 ‘네일 기본관리’ 학습모듈의 내용

학습 1	프리에지 형태 만들기(LM1201010403_14v2.1)
학습 2	큐티클 정리하기(LM1201010403_14v2.2)
<b>학습 3 컬러링하기(LM1201010403_14v2.3)</b>	
학습 4	보습제 도포하기(LM1201010403_14v2.4)
학습 5	네일 기본관리 미무리하기(LM1201010403_14v2.5)

#### 학습은

해당 NCS 능력단위요소 명칭을 사용하여 제시하였습니다. 학습은 일반교과의 ‘대단원’에 해당되며, 모듈을 구성하는 가장 큰 단위가 됩니다. 또한 완성된 직무를 수행하기 위한 가장 기초적인 단위로 사용할 수 있습니다.

#### 학습내용은

요소 별 수행준거를 기준으로 제시하였습니다. 일반교과의 ‘중단원’에 해당합니다.

#### 학습목표는

모듈 내의 학습내용을 이수했을 때 학습자가 보여줄 수 있는 행동수준을 의미합니다. 따라서 일반 수업시간의 과목목표로 활용할 수 있습니다.

#### 3-1. 컬러링 매뉴얼 이해

##### 학습목표

- 고객의 요구에 따라 네일 폴리시 색상의 침착을 막기 위한 베이스코트를 아주 얇게 도포할 수 있다.
- 작업 매뉴얼에 따라 네일 폴리시를 일plex 없이 균일하게 도포할 수 있다.
- 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 톱코트를 바를 수 있다.

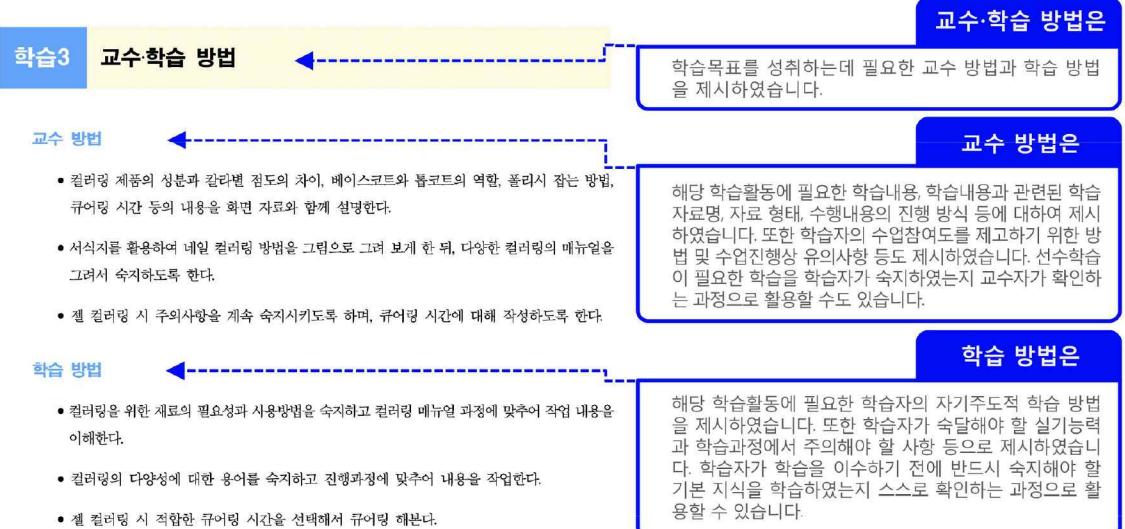
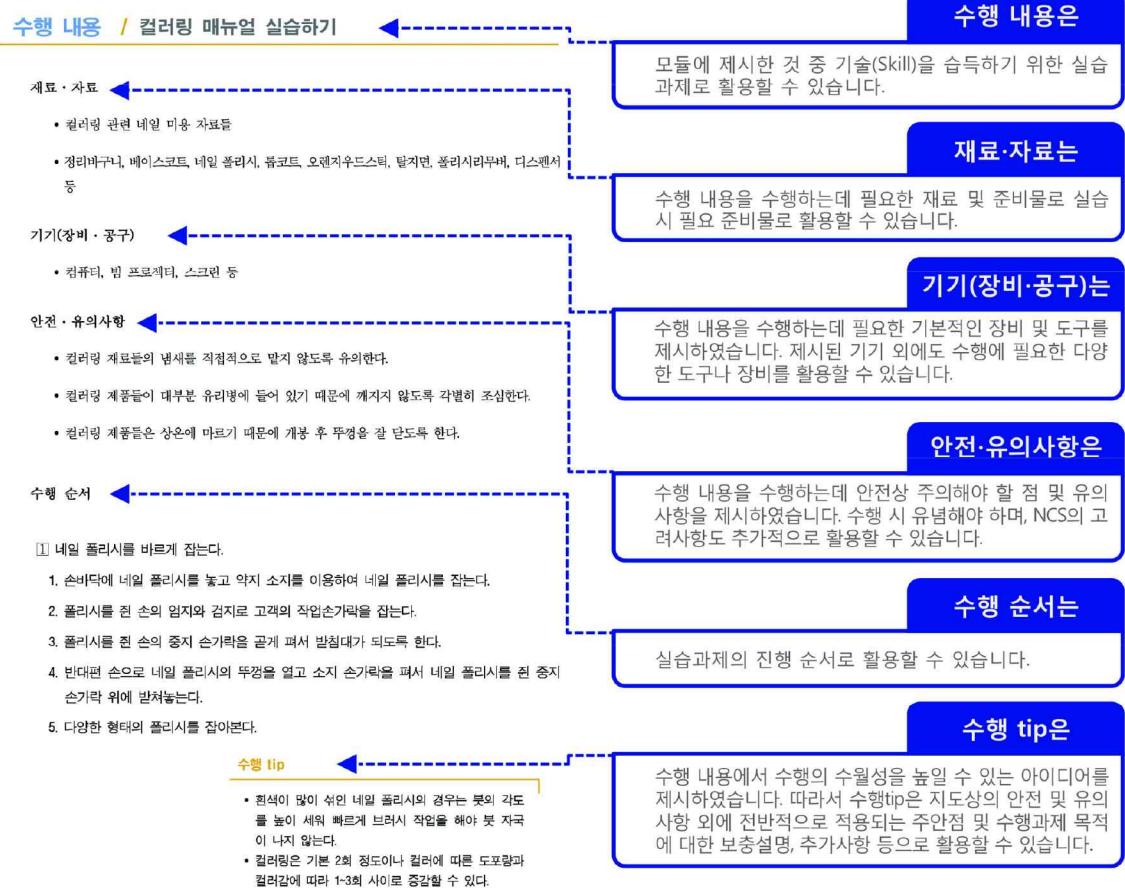
##### 필요 지식 /

##### ① 컬러링 매뉴얼

컬러링 작업 전, 아세톤 또는 네일 폴리시 리무버를 사용하여 손톱표면과 큐티클 주변, 손톱 밑 부분까지 깨끗하게 유분기를 제거해야 한다. 컬러링의 순서는 Base coating 1회 → Polishing 2회 → 컬러수정 → Top coating 1회 → 최종수정의 순서로 한다. 베이스코트는 착색을 방지하고 빌릴성 향상을 위해 가장 먼저 도포하며 컬러링의 마지막에 컬러의 유지와 광택을 위해 톱코트를 도포한다. 네일 보강제(Nail Strengthener)를 바를 시에는 베이스코트를 도포하기 전에 사용한다.

#### 필요지식은

해당 NCS의 지식을 토대로 해당 학습에 대한 이해와 성과를 높이기 위해 알아야 할 주요 지식을 제시하였습니다. 필요지식은 수행에 꼭 필요한 핵심 내용을 위주로 제시하여 교수자의 역할이 매우 중요하며, 이후 수행순서 내용과 연계하여 교수·학습으로 진행할 수 있습니다.



## 학습3 평가

## 평가 준거

- 평가는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.

- 평가는 다음 사항을 평가해야 한다.

학습내용	학습목표	성취수준
		상 중 하
컬러링 매뉴얼 이해	- 고객의 요구에 따라 네일 폴리시 색상의 침착을 막기 위한 베이스코트를 아주 알게 도포할 수 있다.	
	- 작업 매뉴얼에 따라 네일 폴리시를 얼룩 없이 균일하게 도포할 수 있다.	
	- 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 툴코트를 바를 수 있다.	

## 평가 방법

- 작업장 평가

학습내용	평가 항목	성취수준
		상 중 하
컬러링 매뉴얼 이해	- 고객의 요구에 따라 네일 폴리시 색상의 침착을 막기 위한 베이스코트를 아주 알게 도포할 수 있다.	
	- 작업 매뉴얼에 따라 네일 폴리시를 얼룩 없이 균일하게 도포할 수 있다.	
	- 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 툴코트를 바를 수 있다.	

## 피드백

## 1. 작업장 평가

- 작업 결과물을 확인하여 수정사항을 제시하고 수정 부분을 인지하도록 한다.

## 4. 참고 자료

## 참고 자료

• 김미원(2011).『Nail Study』. 서울: 사)한국네일자식서비스협회.

• 민방경(2015).『미용사(네일)필기』. 서울: 예문사.

• 박은주(2014).『네일미용』. 서울: 정담미디어.

## 5. 활용 서식/부록

## 활용서식

## 프리에지 형태 실습지

## 1. 프리에지 형태의 이해

모양	이름	특징
	( ) Square nail	- 강한 느낌의 사각형태 - 네일의 양끝 모서리 부분이 90° 사각의 형태이다. - 발톱의 형태 활용 - 내인성 발톱의 보정시에 적용

해당 NCS 능력단위 평가방법과 평가 시 고려 사항을 준용하여 작성하였습니다. 교수자 및 학습자가 평가항목 별 성취수준을 확인하는데 활용할 수 있습니다.

## 평가 준거는

학습자가 해당 학습을 어느 정도 성취하였는지를 평가하기 위한 기준을 제시하고 있습니다. 학습목표와 연계하여 단위수업 시간에 평가항목 별 성취수준을 평가하는데 활용할 수 있습니다.

## 평가 방법은

NCS 능력단위의 평가방법을 준용하였으며, 평가 준거에 따른 평가방법을 2개 이상 제시하였습니다. 평가방법으로는 포트폴리오, 문제해결 시나리오, 서술형 시험, 논술형 시험, 사례연구, 평가자 체크리스트, 작업장 평가 등이 있으며, NCS의 능력단위 요소 별 수행 수준을 평가하는데 가장 적절한 방법을 선정하여 활용할 수 있습니다.

## 피드백은

평가 후에 학습자들에게 평가 결과를 피드백하여 부족한 부분을 알려주고, 학습 결과가 미진한 경우, 해당 부분을 다시 학습하여 학습목표를 달성하는 데 활용할 수 있습니다.

## 참고자료는

해당 학습모듈의 필요지식에 대한 출처와 인용한 참고자료 및 사이트를 제시하였습니다.

## 활용서식은

평가 서식, 실험시트 등 교수학습 시 활용 가능한 다양한 서식들로 구성하였습니다. 과제 진행에서 평가에 이르기까지 필요한 서식을 해당 학습모듈의 특성에 맞춰 개발하거나 기존의 양식을 활용하여 제시하였습니다.

## 부록

## 네일 기본관리 도구와 재료 목록

목록	비고	준비
위생가운	흰색	작업자 착용
위생 미스트	흰색	작업자 착용
보호안경	투명한 렌즈 (안경으로 대체 가능)	작업자 착용
재료정리함	제질, 색상 무관	작업대

활용서식 이외에 교수학습과정에서 참고할 수 있는 자료가 있는 경우 제시하였습니다.

## 부록은

## [NCS-학습모듈의 위치]

대분류	정보통신
중분류	정보기술
소분류	정보기술개발

세분류	능력단위	학습모듈명
SW아키텍처	데이터베이스 요구사항 분석	데이터베이스 요구사항 분석
응용SW 엔지니어링	개념데이터 모델링	개념데이터 모델링
임베디드SW 엔지니어링	논리 데이터베이스 설계	논리 데이터베이스 설계
DB엔지니어링	물리 데이터베이스 설계	물리 데이터베이스 설계
NW엔지니어링	데이터베이스 구현	데이터베이스 구현
보안엔지니어링	데이터 품질관리	데이터 품질관리
UI/UX엔지니어링	데이터 전환 설계	데이터 전환 설계
시스템SW 엔지니어링	데이터 전환	데이터 전환
	데이터베이스 성능확보	데이터베이스 성능확보
	데이터 표준화	데이터 표준화
SQL활용	SQL활용	SQL활용
	SQL응용	SQL응용

---

# 차 례

---

학습모듈의 개요 ----- 1

학습 1. 기본 SQL 작성하기

1-1. DDL 활용 -----	3
1-2. DML 활용 -----	15
1-3. DCL 활용 -----	23
1-4. 데이터 사전 검색 -----	35
• 교수 · 학습 방법 -----	44
• 평가 -----	45

학습 2. 고급 SQL 작성하기

2-1. 인덱스 활용 -----	48
2-2. 뷰 활용 -----	58
2-3. 다중 테이블 검색 -----	68
• 교수 · 학습 방법 -----	80
• 평가 -----	81

참고 자료 ----- 83



# SQL 활용 학습모듈의 개요

## 학습모듈의 목표

관계형 데이터베이스에서 SQL을 사용하여 목적에 적합한 데이터를 정의하고, 조작하며, 제어할 수 있다.

## 선수학습

데이터베이스 개론, 데이터 모델링과 설계, 데이터베이스 관리

## 학습모듈의 내용 체계

학습	학습 내용	NCS 능력단위 요소	
		코드 번호	요소 명칭
	1-1. DDL 활용		
	1-2. DML 활용		
1. 기본 SQL 작성하기	1-3. DCL 활용	2001020413_16v3.1	기본 SQL 작성하기
	1-4. 데이터 사전 검색		
	2-1. 인덱스 활용		
2. 고급 SQL 작성하기	2-2. 뷰 활용	2001020413_16v3.2	고급 SQL 작성하기
	2-3. 다중 테이블 검색		

## 핵심 용어

테이블(Table), 기본키(Primary Key), 외래키(Foreign Key), 무결성 제약 조건(Integrity Constraint), 참조 무결성(Referential Integrity), 트랜잭션(Transaction), DDL(Data Definition Language), DML(Data Manipulation Language), DCL(Data Control Language), 인덱스(Index)



## 학습 1

# 기본 SQL 작성하기

## 학습 2

## 고급 SQL 작성하기

## 1-1. DDL 활용

### 학습 목표

- 테이블의 구조와 제약 조건을 생성, 삭제하고 수정하는 DDL(Data Definition Language) 명령문을 작성할 수 있다.

### 필요 지식 /

#### ① DDL 개요

##### 1. DDL 대상

DDL(Data Definition Language)은 ‘데이터를 정의하는 언어’로서, 보다 엄밀하게 말하면 ‘데이터를 담는 그릇을 정의하는 언어’이며, 이러한 그릇을 DBMS에서는 오브젝트라고 한다. DDL을 통해 정의할 수 있는 대상, 오브젝트 유형은 다음과 같다.

<표 1-1> DDL 대상

DDL 대상	설명	비고
스키마(Schema)	<ul style="list-style-type: none"><li>- DBMS 특성과 구현 환경을 감안한 데이터 구조</li><li>- 직관적으로 하나의 데이터베이스로 이해 가능</li></ul>	DBMS마다 차이
도메인(Domain)	<ul style="list-style-type: none"><li>- 속성의 데이터 타입과 크기, 제약 조건 등을 지정한 정보</li><li>- 속성이 가질 수 있는 값의 범위로 이해 가능</li></ul>	예를 들어, 주소를 VARCHAR(120)로 정의
테이블(Table)	<ul style="list-style-type: none"><li>- 데이터 저장 공간</li></ul>	본 학습 대상
뷰(View)	<ul style="list-style-type: none"><li>- 하나 이상의 물리 테이블에서 유도되는 가상의 논리 테이블</li></ul>	학습 2-2 참조
인덱스(Index)	<ul style="list-style-type: none"><li>- 검색을 빠르게 하기 위한 데이터 구조</li></ul>	학습 2-1 참조

## 2. DDL 조작 방법

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

<표 1-2> DDL 명령어

구분	DDL 명령어	내용
생성	CREATE	데이터베이스 오브젝트 생성
변경	ALTER	데이터베이스 오브젝트 변경
삭제	DROP	데이터베이스 오브젝트 삭제
	TRUNCATE	데이터베이스 오브젝트 내용 삭제

DDL 명령어로 분류되지는 않지만 DDL과 같이 사용되는 명령어가 있다. 비상용 제품인 M\*SQL의 경우, 생성된 오브젝트의 목록을 조회하기 위해서는 SHOW 명령문을 사용하며, 내용을 조회하기 위해서는 SELECT 문을 사용한다. 상용 제품인 O\*의 경우 SELECT로 목록과 내용을 조회한다.

### ② DDL 활용

데이터베이스를 구축하기 위해 스키마, 테이블, 도메인, 인덱스, 뷰와 같은 오브젝트에 대한 DDL 적용이 필요하나, 본 학습에서는 테이블만을 대상으로 한다.

#### 1. 테이블 생성

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

<표 1-3> 테이블 생성 SQL문

구분	문법
신규 생성	<pre>CREATE TABLE 테이블이름 (     열이름 데이터 타입 [DEFAULT 값] [NOT NULL]     {,열이름 데이터 타입 [DEFAULT 값] [NOT NULL] }*     [PRIMARY KEY (열 리스트),]     {[FOREIGN KEY (열 리스트) REFERENCES 테이블이름 [(열이름)]}         [ ON DELETE 옵션 ]         [ ON UPDATE 옵션 ] ], }*     [CHECK (조건식)   UNIQUE(열이름) ] ) ;</pre>
다른 테이블 정보를 이용한 테이블 생성 <sup>1)</sup>	CREATE TABLE 테이블이름 AS SELECT 문;

1) 다른 테이블을 이용해서 신규 테이블을 생성하는 방법은 DBMS 제품마다 차이가 있다.

## 2. 테이블 변경

ALTER를 이용하여 테이블 구조를 변경하는 문법은 다음과 같다.

<표 1-4> ALTER 이용한 테이블 변경 SQL문

구분	문법
열 추가	ALTER TABLE 테이블이름 ADD 열이름 데이터타입 [DEFAULT 값]
열 데이터 타입 변경	ALTER TABLE 테이블이름 MODIFY 열이름 데이터타입 [DEFAULT 값]
열 삭제	ALTER TABLE 테이블이름 DROP 열이름

## 3. 테이블 삭제, 절단, 이름 변경

DROP TABLE, TRUNCATE TABLE, RENAME TABLE 명령문을 사용하여 테이블을 삭제, 절단, 이름 변경을 할 수 있다. 테이블 및 테이블 내용을 삭제하기 위한 명령어의 사용 문법은 다음과 같다.

<표 1-5> 테이블 삭제 및 이름 변경 방법

구분	문법
테이블 삭제	DROP TABLE 테이블이름
테이블 내용 삭제	TRUNCATE TABLE 테이블이름
테이블이름 변경	RENAME TABLE 이전테이블이름 TO 새로운테이블이름 ALTER TABLE 이전테이블이름 RENAME 새로운테이블이름

## ③ 제약 조건 적용

### 1. 제약 조건 유형

다음과 같은 제약 조건을 테이블 생성 과정에 적용할 수 있다.

<표 1-6> 테이블 생성에 사용되는 제약 조건

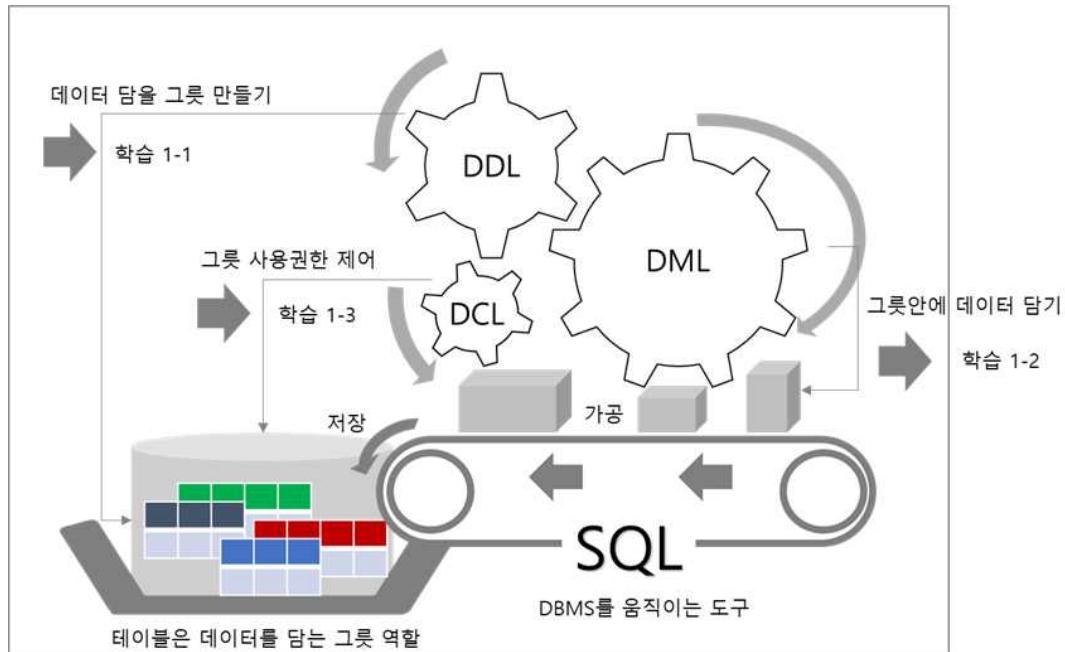
제약 조건	설명
PRIMARY KEY	테이블의 기본키를 정의함. 기본으로 NOT NULL, UNIQUE 제약이 포함됨.
FOREIGN KEY	외래키를 정의함. 참조 대상을 테이블이름(열이름)으로 명시해야 함. 참조 무결성 위배 상황 발생 시 처리 방법으로 옵션 지정 가능 - NO ACTION, SET DEFAULT, SET NULL, CASCADE
UNIQUE	테이블 내에서 같은 유일한 값을 가져야 함. 테이블 내에서 동일한 값을 가져서는 안 되는 항목에 지정함.
NOT NULL	테이블 내에서 관련 열의 값은 NULL일 수 없음. 필수 입력 항목에 대해 제약 조건으로 설정함.
CHECK	개발자가 정의하는 제약 조건 상황에 따라 다양한 조건 설정 가능

## 2. 제약 조건 활용

테이블 생성을 위한 CREATE 문에 제약 조건을 명시하는 형태로 사용하며, ALTER를 통해 테이블의 제약 조건을 변경할 수 있다.

### ④ SQL 활용 주요 내용

SQL은 DDL, DML 및 DCL과 같은 유형의 작업을 통해 데이터베이스 안에 그릇을 만들고 그 안에 데이터를 담거나 꺼내어 사용하는 도구다. 본 학습의 주요 내용과 이들 간의 관계는 다음 그림과 같다.



[그림 1-1] SQL 활용을 위한 주요 학습 내용

## 수행 내용 / DDL 활용하기

### 재료 · 자료

- 주문 정보, 회원 정보 관련 데이터 요구사항
- 개념 E-R 다이어그램

### 기기(장비 · 공구)

- E-R 모델링 도구
- DBMS(DataBase Management System) 프로그램

### 안전 · 유의 사항

- 한글로 번역하기 어색한 용어는 원어를 한글 발음으로 표현한다. (예시: ‘자료 유형’보다는 ‘데이터 타입’으로 표현함)
- 테이블을 생성하는 방법에 대한 학습 이전에 테이블을 구성하는 데이터의 주제 영역 및 모델링 등에 대한 이해가 필요하다.

### 수행 순서



[그림 1-2] DDL 활용을 위한 수행내용

## ① 테이블을 통해 관리할 데이터를 확인한다.

### 1. 데이터를 확인한다.

본 학습 주제인 데이터를 정의 (Data Definition) 한다는 것은 다음과 같은 데이터를 담는 그릇을 만드는 것과 같다.

주문테이블

주문번호	고객번호	주문일	주문가격	배송 도시	배송완료일	결제금액	할인금액
A0100	24680	20170704	55,000	서울	20170707	45,000	10,000
X0300	13579	20170801	70,000	부산	Null	50,000	20,000

[그림 1-3] 주문 데이터 예시

### 2. 데이터 타입을 결정한다.

데이터 타입은 자료를 표현할 때 필요한 공간과 형식을 규정하는 기준이다. SQL에서 컬럼을 정의할 때 규정되는 데이터 타입은 이후 해당 컬럼에 입력되는 데이터의 종류와 크기를 제한하며, 다른 종류의 데이터나 큰 데이터가 입력되면 오류가 나타난다.

SQL 데이터의 유형은 크게 숫자형, 문자형, 날짜 등의 유형으로 나눌 수 있으며, 구체적인 형식은 NUMERIC, CHARACTER, VARCHAR, DATETIME 등이 있으나, 실제 사용하는 SQL 제품에 따라 사용되는 표현은 조금씩 달라질 수 있다.

[그림 1-3]과 같은 데이터를 저장할 테이블을 만들고자 할 때, 각 컬럼의 값은 크게 문자열과 숫자로 구분할 수 있으므로 각각 다음과 같은 데이터 타입을 사용한다.

- 문자열 – VARCHAR
- 숫자 – DECIMAL

### 3. 제약 조건이 무엇인지 확인한다.

제약 조건으로는 주문 번호를 기본키(Primary Key)로 한다. 하지만 테이블의 변경으로 추가적인 제약 조건이 발생할 수 있다.

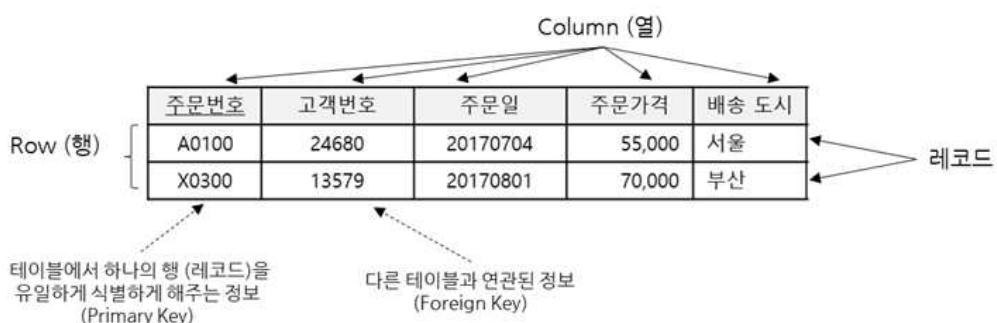
## ② DDL 관리 대상인 테이블에 대해 조사한다.

### 1. 테이블의 구성요소에 대해 조사한다.

- 눈에 보이지 않는 그 어떤 것을 문제 해결의 수단이나 목적으로 하는 경우, 이를 가시화 하는 것이 문제해결의 시작이다. 특히 데이터베이스 용어에 대한 개념을 형상화하는 것은 수행에 있어 필수적이다.
- 학습자는 테이블에 대해 조사하여 그 내용을 아래와 같은 테이블 모습 및 고려사항과 비교해 보도록 한다.

테이블은 행(Row)과 열(Column)로 구성되는 가장 기본적인 데이터베이스 객체로, 데이터베이스 내에서 모든 데이터는 테이블 안에 저장된다.

다음 그림에서 볼 수 있듯이, 하나의 행은 다른 행과 구별되는 정보를 가지도록 구성되어 있으며, 이러한 개별 행을 유일하게 식별하는 속성을 기본키(Primary Key)라고 한다. 또 데이터베이스는 정보의 중복을 최소화하기 위해 정보 종류를 테이블에 해당하는 엔티티(Entity) 단위로 분리하여 저장하며, 이렇게 분리된 정보 사이의 관계를 이어주는 수단이 외래키(Foreign Key)이다.



[그림 1-4] 테이블의 내부 구성 요소

- 참고로 [그림 1-4]와 같이 정보를 격자 형태의 테이블로 저장하고 테이블의 관계를 이용해 정보를 활용하는 데이터베이스를 ‘관계형 데이터베이스’라고 한다. 관계형 데이터베이스가 아닌 것은 어떤 것이 있으며, 어떻게 형상화가 되는지에 대해서도 조사해 본다.

## 2. 테이블에서 사용할 제약조건을 선택한다.

테이블 구조를 정의하면서 데이터 무결성을 유지하기 위해 제약 조건을 이용한다. 제약 조건이 테이블 생성 과정의 필수 요소는 아니지만 데이터베이스의 일관성(Consistency)을 유지하고 잘못된 데이터의 입력과 수정으로부터 데이터베이스를 보호하는 데 필수적인 요소이므로 반드시 정의하는 것이 좋다.

대표적인 제약 조건은 기본키(Primary Key)와 외래키(Foreign Key)에 의한 제약 조건이다. 그 밖에도 UNIQUE KEY, PRIMARY KEY, FOREIGN KEY, NOT NULL, CHECK 등이 있다.

## ③ 테이블을 생성하고 변경한다.

### 1. CREATE 문으로 테이블을 생성한다.

[그림 1-3]과 같은 데이터를 저장하기 위한 테이블을 생성하기 위해 다음과 같은 SQL문을 사용한다 (각 데이터 타입의 괄호 안 숫자는 최대 길이를 의미한다.).

<표 1-7> 테이블 생성 SQL문

위치	테이블 생성 SQL문		
1	CREATE TABLE	'주문테이블'	(
2	'주문번호'	varchar(16)	NOT NULL,
3	'고객번호'	varchar(16)	NOT NULL,
4	'주문일'	varchar(8)	NOT NULL,
5	'주문가격'	decimal(15,2)	NOT NULL,
6	'배송도시'	varchar(256),	
7	'배송완료일'	varchar(8),	
8	'결제금액'	varchar(8),	
9	'할인금액'	decimal(15,2)	NOT NULL,
10	'적립포인트'	decimal(15,2)	NOT NULL
11	PRIMARY KEY ('주문번호')		
12	)		

## 2. SELECT를 이용하여 테이블을 생성한다.

만일 생성하고자 하는 ‘기존테이블’과 동일한 컬럼을 가진 ‘신규테이블’이 있다면 다음과 같이 기존에 존재하는 테이블 정보를 이용하여 새로운 테이블을 만들 수 있다.

```
CREATE TABLE 신규테이블 AS SELECT * FROM 기존테이블;
```

위 SQL문의 SELECT 부분을 통해 기존테이블의 속성을 조회하여 신규테이블의 속성으로 정의하여 생성하는 방식이다. 이와 같이 SELECT 문을 이용해 테이블을 생성하는 방식은 <표 1-7>과 같은 테이블 생성 방법 대비 다음과 같은 특징이 있음에 유의한다.

- 생성된 테이블은 기존 테이블의 컬럼 및 데이터 유형과 길이 등을 그대로 적용함.
- NOT NULL의 정의는 그대로 적용함.
- 제약 조건은 적용되지 않음.
- ALTER TABLE을 사용하여 제약 조건을 추가해야 함.
- 동일한 컬럼들로 생성된 경우 '\*'를 사용함
- 필요한 컬럼만을 지정하여 테이블을 생성할 수 있음

## 3. 생성한 테이블을 확인한다.

테이블의 생성 유무를 확인하기 위해 다음과 같은 명령어를 사용한다.

<표 1-8> 테이블 확인 SQL문

DBMS 제품	테이블 목록 확인 SQL문
대표적인 상용 O 제품	SELECT * FROM user_tables;
대표적인 비상용 M 제품	SHOW TABLES;

결과에 앞서 생성한 ‘주문테이블’이 있으면 테이블 생성이 성공한 것이다. 또는 다음 명령어를 통해 테이블 구조를 확인할 수 있다.

<표 1-9> 테이블 구조 보기

구분	내용																																																		
테이블 구조 보기 명령	DESC ‘주문테이블’;																																																		
명령 결과																																																			
<table border="1"> <thead> <tr> <th>Field</th><th>Type</th><th>Null</th><th>Key</th><th>Default</th></tr> </thead> <tbody> <tr> <td>주문번호</td><td>varchar(16)</td><td>NO</td><td>PRI</td><td>NULL</td></tr> <tr> <td>고객번호</td><td>varchar(16)</td><td>NO</td><td></td><td>NULL</td></tr> <tr> <td>주문일</td><td>varchar(8)</td><td>NO</td><td></td><td>NULL</td></tr> <tr> <td>주문가격</td><td>decimal(15,2)</td><td>NO</td><td></td><td>NULL</td></tr> <tr> <td>배송도시</td><td>varchar(256)</td><td>YES</td><td></td><td>NULL</td></tr> <tr> <td>배송완료일</td><td>varchar(8)</td><td>YES</td><td></td><td>NULL</td></tr> <tr> <td>결제금액</td><td>varchar(8)</td><td>YES</td><td></td><td>NULL</td></tr> <tr> <td>할인금액</td><td>decimal(15,2)</td><td>NO</td><td></td><td>NULL</td></tr> <tr> <td>적립포인트</td><td>decimal(15,2)</td><td>NO</td><td></td><td>NULL</td></tr> </tbody> </table>		Field	Type	Null	Key	Default	주문번호	varchar(16)	NO	PRI	NULL	고객번호	varchar(16)	NO		NULL	주문일	varchar(8)	NO		NULL	주문가격	decimal(15,2)	NO		NULL	배송도시	varchar(256)	YES		NULL	배송완료일	varchar(8)	YES		NULL	결제금액	varchar(8)	YES		NULL	할인금액	decimal(15,2)	NO		NULL	적립포인트	decimal(15,2)	NO		NULL
Field	Type	Null	Key	Default																																															
주문번호	varchar(16)	NO	PRI	NULL																																															
고객번호	varchar(16)	NO		NULL																																															
주문일	varchar(8)	NO		NULL																																															
주문가격	decimal(15,2)	NO		NULL																																															
배송도시	varchar(256)	YES		NULL																																															
배송완료일	varchar(8)	YES		NULL																																															
결제금액	varchar(8)	YES		NULL																																															
할인금액	decimal(15,2)	NO		NULL																																															
적립포인트	decimal(15,2)	NO		NULL																																															

[그림 1-5] 주문테이블 구조

#### 4. CREATE 문에 사용된 제약 조건을 조사한다.

테이블 생성을 위해 사용한 <표 1-7>과 같은 SQL문에서 무결성을 보장하기 위해 사용된 제약 조건은 다음과 같다.

<표 1-10> 테이블 생성 시 제약 조건

위치	요소	의미
6 ~ 8		제약 조건 없음
2 ~ 5, 9 ~ 10	NOT NULL	각 컬럼의 값으로 Null 값 허용 여부 지정 - 제약 조건
11	PRIMARY KEY	기본키(Primary Key) 지정 - 제약 조건

#### ④ 테이블 구조를 변경한다(M\*-SQL 기준).

##### 1. 테이블 구조 변경 대상을 정의한다.

앞에서 생성한 ‘배송도시’를 ‘배송도시코드’로 속성 이름을 바꾸면서 동시에 데이터 타입 을 바꾸어 보자. 이때 새로운 데이터 타입을 정수형으로 정의한다.

## 2. 컬럼 이름과 타입을 변경한다.

‘배송도시’를 ‘배송도시코드’로 변경하는 SQL문은 다음과 같다.

```
ALTER TABLE ‘주문테이블’ CHANGE ‘배송도시’ ‘배송도시코드’ INT;
```

‘ALTER TABLE 테이블명 CHANGE 컬럼명 자료형’를 사용하면 열의 크기와 데이터 타입을 변경할 수 있다. 열의 크기 확대는 별 제한이 없지만 축소나 데이터 타입 변환은 열의 모든 데이터가 NULL 값을 가지는 경우에만 허용된다. DEFAULT 값을 수정할 수 있다. DEFAULT 값을 수정하면 수정 이후에 입력되는 값에만 수정된 DEFAULT 값이 적용된다.

## 3. 컬럼을 추가하고 삭제한다.

다음과 같은 명령어를 통해 컬럼의 추가 및 삭제가 가능하다.

<표 1-11> 테이블 변경을 위한 SQL문

구분	SQL문 형식
맨 뒤에	ALTER TABLE ‘테이블명’ ADD ‘컬럼명’ 자료형;
추가	맨 앞에      ALTER TABLE ‘테이블명’ ADD ‘새컬럼명’ 자료형 FIRST
	지정 컬럼 뒤    ALTER TABLE ‘테이블명’ ADD ‘새컬럼명’ 자료형 AFTER ‘앞컬럼명’
삭제	ALTER TABLE ‘테이블명’ DROP ‘컬럼명’;

‘ALTER TABLE 테이블명 ADD 컬럼명 자료형’ 문을 사용하여 테이블에 열을 추가하면 열은 테이블의 마지막에 추가된다. 테이블이 가지는 열의 개수에는 제한이 있어서 추가가 안 될 수도 있다.

## ⑤ 제약 조건을 변경한다(M\*-SQL 기준).

### 1. 제약 조건 변경 대상을 정의한다.

앞서 ‘배송도시’를 ‘배송도시코드’로 변경하면서 데이터 타입을 INT로 바꾸었다. 배송도시 명칭을 알기 위해서는 ‘도시코드테이블’에서 코드를 통해 조회해야 한다. 이를 위해 ‘배송도시코드’를 외래키(FOREIGN KEY)로 정의하고자 한다.

### 2. 제약 조건을 변경한다.

외래키(FOREIGN KEY)라는 제약 조건을 추가하는 SQL문은 다음과 같다.

```
ALTER TABLE ‘주문테이블’  
ADD FOREIGN KEY (‘배송도시코드’) REFERENCES ‘도시코드테이블’(code);
```

방금 생성한 외래키 제약 조건을 삭제하기 위한 SQL문은 다음과 같다.

```
ALTER TABLE '주문테이블' DROP FOREIGN KEY [제약조건 이름];
```

여기서 [제약조건 이름]은 시스템이 부여한 것으로 별도의 테이블에서 내용을 확인할 수 있다(information\_schema.table\_constraints 테이블).

### 3. 추가적으로 제약 조건 변경 SQL문을 조사한다.

ALTER TABLE을 사용하여 테이블에 제약 조건을 추가하거나 삭제할 수 있으나, 수정은 불가하다. 다음 표는 ALTER TABLE로 테이블에 제약 조건을 추가, 삭제, 비활성화하는 SQL 문법이다.

<표 1-12> 제약 조건 변경 SQL문

제약 조건 변경 내용	SQL 명령문
제약 조건 추가	ALTER TABLE 테이블이름 ADD [CONSTRAINT 제약조건이름] 제약조건(열이름)
제약 조건 삭제	ALTER TABLE 테이블이름 CONSTRAINT 제약조건이름 ! 테이블이름 PRIMARY KEY ! FOREIGN KEY(열이름) ! UNIQUE(열이름)
제약 조건 비활성화	ALTER TABLE 테이블이름 DISABLE CONSTRAINT 제약조건이름
제약 조건 활성화	ALTER TABLE 테이블이름 ENABLE CONSTRAINT 제약조건이름

제약 조건을 추가하려면 관련된 열의 데이터가 제약 조건에 부합해야만 한다. PRIMARY KEY 제약을 추가하려면 관련 열이 NULL 값을 가져도, 동일한 값을 가져도 안 된다. UNIQUE 제약은 관련 열이 유일한 값 또는 NULL 값을 가져야 한다. FOREIGN 키와 관련된 열은 해당 열의 값이 NULL이거나 참조하는 테이블에 있어야 한다.

제약 조건의 비활성화는 제약 조건을 없애는 것이 아니라 사용하지 못하게 하는 것이다. ‘ALTER TABLE 테이블이름 DISABLE CONSTRAINT 제약조건이름’은 제약 조건을 사용하지 못하게 비활성화한다. 비활성화된 제약 조건은 ‘ALTER TABLE 테이블이름 ENABLE CONSTRAINT 제약조건이름’으로 다시 활성화할 수 있다.

## 수행 tip

- SQL 명령문은 데이터베이스 및 SQL 언어의 종류에 따라 조금씩 달라질 수 있다. 그러므로 사용자가 보유한 데이터베이스 및 SQL 언어를 고려하여 수행해야 한다.
- 수행의 각 단계별 지속적인 구조 확인을 통해 의도한 수행이 이루어지고 있는지 확인하는 것이 좋다.
- DDL 대상으로 테이블 이외의 오브젝트에 대한 이해가 필요하다. 이들 오브젝트 하나하나가 학습 주제로서 추가적 관심이 필요하다.
- 본 학습에서는 주로 테이블을 생성하는 논리적 접근 방법에 대해 알아보았다. 테이블을 생성할 때, 테이블의 물리적 구성 역시 고려 대상이다. 테이블이 디스크에 저장되는 주요 방식으로 Heap Organized Table, Clustered Index Table, Partitioned Table 등이 있다. 저장 방식의 선택은 테이블의 성능 뿐 아니라 확장성, 가용성 등을 고려하여 테이블 저장 형태를 선택해야 한다.

# 1-2. DML 활용

## 학습 목표

- 한 개의 테이블에 대해 데이터를 삽입, 수정, 삭제하고 행을 조회하는 DML(Data Manipulation Language) 명령문을 작성할 수 있다.

## 필요 지식 /

### ① DML 개요

#### 1. DML 의미

데이터를 조작하는 명령어를 DML(Data Manipulation Language)이라고 한다. 여기서 조작은 데이터 관점에서 생명 주기를 제어하는 것을 의미한다.

#### 2. DML 유형

데이터의 생명 주기 관리 및 활용을 위해 사용하는 DML 명령어는 다음과 같다.

<표 1-13> DML 명령어

구분	DML 명령어	내용
데이터 생성	INSERT	삽입 형태로 신규 데이터를 테이블에 저장
데이터 조회	SELECT	테이블의 내용을 조회
데이터 변경	UPDATE	테이블의 내용을 변경
데이터 삭제	DELETE	테이블의 내용을 삭제

### ② DML 명령문

#### 1. 데이터 삽입(INSERT)

데이터를 삽입하기 위한 명령어로 다음과 같이 두 가지 형태의 명령문 형식을 제공한다. 이때 데이터 삽입 결과로 하나의 레코드가 추가된다. 따라서 삽입에 사용되는 정보는 하나의 레코드를 충분히 묘사해야 한다.

<표 1-14> INSERT 명령문 유형

형태	INSERT 명령문
A	INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);
B	INSERT INTO table_name VALUES (value1, value2, ...);

## 2. 데이터 조회(SELECT)

데이터의 내용을 조회할 때 사용하는 명령어이다. 가장 많이 사용되는 SQL 명령어로서, 다른 DML 명령어와 같이 사용되어 SQL의 활용을 풍부하게 한다. SELECT 명령어의 기본 형식은 다음과 같다.

```
SELECT [OPTION] columns FROM table [WHERE 절] ;
```

SELECT 문에 사용되는 각 정보는 다음과 같다.

<표 1-15> SELECT 문에서 사용되는 요소

SELECT 문 요소	요소값	내용
OPTION	• ALL • DISTINCT	• 중복 포함한 조회 결과 출력 • 중복 제거한 조회 결과 출력
columns	• 컬럼명 목록 • 와이드카드	• SELECT 통해 조회할 컬럼명 지정 • 모두 또는 전체를 의미하는 *

SELECT 문의 특징적 요소로 별명(Alias) 기능이 있다.

## 3. 데이터 수정(UPDATE)

데이터를 수정할 때 다음과 같은 형태의 UPDATE 명령문을 사용한다.

```
UPDATE table SET column1 = value1, column2 = value2, ... [WHERE 절] ;
```

UPDATE 명령문은 보통 WHERE 절을 통해 어떤 조건이 만족할 경우에만 특정 컬럼의 값을 수정하는 용도로 많이 사용된다.

## 4. 데이터 삭제(DELETE)

레코드를 삭제할 때 다음과 같은 형태의 DELETE 명령문을 사용한다.

```
DELETE FROM table [WHERE 절] ;
```

조건절 없이 DELETE를 사용하는 경우, 테이블 전체가 한 번에 삭제되는 위험이 있다.

## 수행 내용 / DML 활용하기

### 재료 · 자료

- 개념 E-R 다이어그램
- 테이블 설계서
- SQL 언어 문법 학습 자료

### 기기(장비 · 공구)

- E-R 모델링 도구
- DBMS(DataBase Management System) 프로그램

### 안전 · 유의 사항

- 데이터 조작을 위한 준비 과정으로 E-R 다이어그램을 통해 테이블을 생성한다.
- 사용상 주의 사항 또는 권장 사항을 알아보기 이전에 어떠한 방법이 있는지를 우선 살펴본다.
- 간접 홍보 효과를 방지하기 위하여 특정 제조사나 특정 제품에 의존적인 내용은 최소화하였으며, 제품에 특화된 내용을 구분하기 위하여 제품의 약어를 사용해 표기하였다.
- 제품 약어 M\*-SQL과 O\*는 대표적인 비상용 제품과 상용 제품을 의미한다.

### 수행 순서

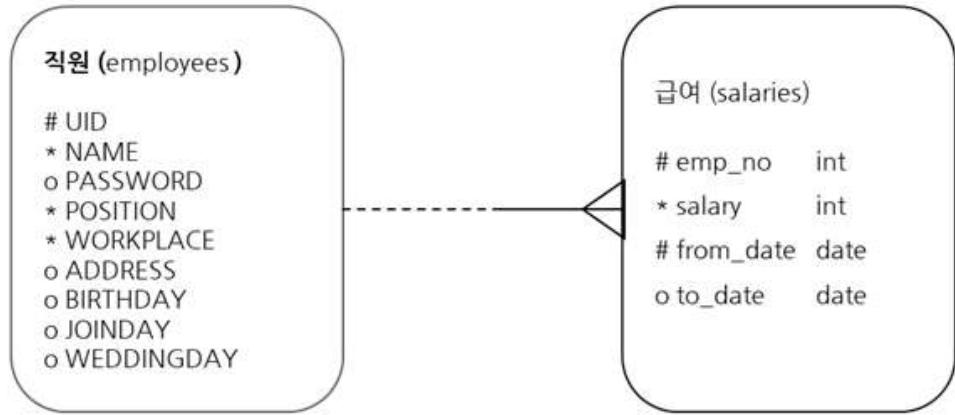


[그림 1-6] DML 활용을 위한 수행내용

## ① 테이블 설계서를 준비한다.

- 수행에 사용할 테이블을 생성한다.

DML을 이용하여, 데이터를 조작하기 위해 우선 다음과 같은 테이블을 생성한다.



[그림 1-7] DML 학습을 위한 대상 테이블의 개념 모델링 모습

위 그림에서 직원 테이블은 급여 테이블과 급여 지급 관계를 가지며, 또한 급여 테이블은 직원 테이블에 종속되는 구조를 가진다. 그림에서는 두 개의 테이블을 확인할 수 있으나, 다음 학습 과정에서는 급여 테이블만을 대상으로 데이터를 조작하는 학습을 수행한다.

급여 테이블을 생성하기 위해 다음과 같은 SQL문을 사용한다.

```
CREATE TABLE salaries ( -- 급여 테이블
    emp_no      INT          NOT NULL,
    salary       INT          NOT NULL,
    from_date   DATE         NOT NULL,
    to_date     DATE,
    FOREIGN KEY (emp_no) REFERENCES employees (UID) ON DELETE CASCADE,
    PRIMARY KEY (emp_no, from_date)
);
```

생성문에서 DML 통해 데이터를 조작하기 위해 유의 깊게 확인해야 할 것은 컬럼 이름과 데이터 타입이다. 또한 생성문에 있는 제약 조건의 확인을 통해 데이터 삽입이나 삭제 과정에 제약조건을 활용할 수 있다.

## ② 데이터를 삽입한다.

### 1. 단일 레코드를 삽입한다.

<표 1-14>의 삽입 명령문 유형 가운데 형태 A로, 하나의 레코드를 추가하는 방법은 다음과 같다.

```
INSERT INTO 'salaries' (emp_no, salary, from_date)  
VALUES (1002, 4000000, '2017-07-01');
```

앞서 급여 테이블 생성 과정에서 ‘to\_date’ 컬럼은 NULL 입력이 가능한 제약조건을 가지고 있기에 데이터를 삽입할 때 해당 컬럼을 생략하고 INSERT 문을 사용하였다.

또 직원 테이블은 직원 번호가 1002인 레코드가 미리 존재해야 위의 명령문이 정상적으로 동작한다. 그 이유는 테이블 생성 과정에서 외래키에 대한 제약 조건 때문이다.

INSERT 명령문 형태 B로 하나의 레코드를 다음과 같이 추가해 보자.

```
INSERT INTO 'salaries' VALUES (1002, 4000000, '2017-08-01'); -- 오류 발생
```

이 경우, 테이블에 정의된 컬럼은 4개인데, 입력값은 3개뿐이라 수행 중 오류가 발생한다. 하지만 테이블 생성 과정에서 ‘to\_date’ 컬럼에 Default 값을 지정하면 오류가 발생하지 않는다. 현재 상황에서 오류 없이 입력하기 위해서는 다음과 같이 4개의 값을 지정해야 한다.

```
INSERT INTO 'salaries' VALUES (1002, 4000000, '2017-08-01', null);
```

### 2. 복수 레코드를 삽입한다.

INSERT 명령문 형태 A로 복수개의 레코드를 삽입하는 명령문은 다음과 같다.

```
INSERT INTO 'salaries' (emp_no, salary, from_date, to_date) VALUES  
(1002, 2000000, '2012-07-01', '2013-06-30'),  
(1002, 3000000, '2013-07-01', '2014-06-30'),  
(1002, 4000000, '2014-07-01', '2015-06-30');
```

INSERT 명령문 형태 B로 복수개의 레코드를 삽입하는 명령문은 다음과 같다.

```
INSERT INTO 'salaries' VALUES  
(1002, 2000000, '2012-07-01', '2013-06-30'),  
(1002, 3000000, '2013-07-01', '2014-06-30'),  
(1002, 4000000, '2014-07-01', '2015-06-30');
```

하나의 SQL문으로 복수의 레코드를 삽입하려면 각각의 레코드 정보를 연속해서 정의하면 된다.

### 3. INSERT 명령문 오류 가능성은 조사한다.

INSERT 명령문은 테이블 구조에 민감하다. 즉, INSERT 명령문을 작성하기 위해서는 CREATE TABLE 문장의 내용을 보고 필수로 넣어야 할 값이 무엇인지, 컬럼의 순서가 어떤지를 조사해야 한다. 이러한 정보를 바탕으로 INSERT 문의 수행에 문제점을 찾을 수 있어야 한다. 또 데이터 타입이 틀린 값이 입력되지 않도록 유의해야 한다.

### ③ 데이터를 조회하고 수정한다.

#### 1. 데이터를 조회한다.

삽입한 데이터를 조회하기 위해 다음과 같은 명령문을 사용한다.

```
SELECT * FROM 'salaries' ;
```

급여 (salaries) 테이블의 모든 컬럼에 대해 아무런 조건 없이 검색하는 기본적인 조회 명령문이다. 데이터가 많아서 일부 데이터만 조회하기 위해서는 조건절이 필요하다. 또 컬럼 가운데 일부만을 보고 싶은 경우에는 와일드카드(\*) 대신에 특정 컬럼을 명시한다. 직원 번호가 1001인 사원의 급여 테이블 정보를 보기 위해서는 다음과 같이 조건을 명기한다.

```
SELECT * FROM 'salaries' WHERE emp_no = 1001;
```

직원 번호 1001번 직원의 급여 테이블 가운데 직원 번호와 급여만을 보고 싶은 경우에는 다음과 같이 특정 컬럼만을 명기한다.

```
SELECT emp_no, salary FROM 'salaries' WHERE emp_no = 1001;
```

조건절에 가장 최근의 날짜를 구하는 방법으로 다음과 같은 SQL문도 가능하다.

```
select * from salaries where from_date = (select max(from_date) from salaries);
```

동일한 결과를 가지는 다음과 같은 SQL문도 가능하다.

```
select * from salaries order by from_date DESC LIMIT 1; -- M*-SQL 경우
```

보통 SELECT를 이용한 조회는 복수 테이블을 대상으로 하는 경우가 많다. 이에 대해서는 ‘학습 2-3. 다중 테이블 검색’에서 보다 자세히 학습하도록 한다.

## 2. 데이터를 수정한다.

데이터를 수정하려면 다음과 같은 정보를 결정해야 한다.

- Condition: 특정 조건 지정
- What: 특정 대상 - 컬럼 지정
- How: 값 - 값 지정

이와 같은 정보를 표현하는 SQL문의 실제 사례는 다음과 같다.

```
UPDATE salaries SET salary = 1100000  
WHERE emp_no=1001 and from_date = '2014-07-01';
```

이 명령문에서 조건은 WHERE 절에 해당하며, 특정 대상과 값은 SET 절에 해당한다.

## ④ 데이터를 삭제한다.

### 1. 데이터를 삭제한다.

[그림 1-7]과 같은 개념 모델링 정보로 만들어진 테이블 구조에서 종속적 개념을 가지는 급여 테이블(salaries)의 특정 항목을 지우는 것은 용이하다. 다음과 같이 그 대상을 조건 절에 명시하면 된다.

```
DELETE FROM salaries WHERE emp_no=1001;
```

### 2. 데이터 삭제 오류가 발생한 이유를 조사한다.

다음과 같은 삭제 명령문을 수행해 보자. 급여 테이블이 아닌 직원 테이블에 있는 특정 직원의 레코드이다.

```
DELETE FROM employees where UID = 1001; -- 오류 발생
```

앞서 정상적으로 수행된 SQL문과 동일한 형태의 삭제 명령문이다. 그런데 이번에는 직원 테이블에 있는 1001번 레코드를 삭제하려는 경우 오류가 발생한다. 그 이유를 조사해 보도록 한다.

결론적으로 그 이유는 테이블 생성 시 지정한 외래키 제약 조건 때문이다. 즉, 급여 테이블에 1001번의 데이터가 존재할 경우, 직원 테이블에서 1001번의 데이터를 삭제할 수 없도록 한 것이다. 이는 무결성 보호 방법의 일종이다.

### 수행 tip

- 데이터베이스는 조직의 모든 자료가 통합 저장되어 공유하는 형태로 운영되는 데이터이므로 이를 사용 하려면 약속을 정하고 지키는 것이 중요하다. 이를 위해 조직의 데이터 아키텍처 수립이 필요하며, 표준화 작업이 필요하다. 이와 같은 작업은 모두 제한된 자원을 효율적으로 공유하기 위해서이다.
- DML 활용의 예시로 ‘select \*’ 형태의 명령문을 제시하고 있지만 이러한 사용 방법은 자제되어야 한다. 모든 컬럼을 조회 대상으로 하는 ‘\*’의 사용은 불필요한 컬럼 정보를 꺼내오는 시간과 이를 전달하는 네트워크상의 부하를 유발하므로 성능을 위해서는 사용 자제가 권고된다.
- INSERT 명령문의 경우, 즉 `INSERT INTO 'salaries' VALUES (1002, 2000000, '2012-07-01', '2013-06-30')`… 형태의 명령문은 대상 컬럼을 지정하지 않고 입력이 가능하다.  
하지만 해당 테이블의 컬럼이 추가 또는 삭제되어 구조가 변경되는 경우, 컬럼을 지정하지 않은 INSERT 문은 더 이상 유효하지 않게 된다. 따라서 이러한 INSERT 문의 사용 방법 역시 자제가 권고된다.

# 1-3. DCL 활용

## 학습 목표

- 업무 단위인 트랜잭션의 완료와 취소를 위한 DCL(Data Control Language) 명령문을 작성할 수 있다.

## 필요 지식 /

### ① DCL 개요

#### 1. DCL 유형

데이터베이스에서 데이터 이외의 오브젝트에 대해 조작할 필요가 있다. 이때 사용하는 SQL 명령을 DCL(Data Control Language)이라고 한다. DCL의 조작 대상, 오브젝트 유형은 다음과 같다.

<표 1-16> DCL 조작 대상

오브젝트	목적	내용
사용자 권한	접근 통제	사용자를 등록하고, 사용자에게 특정 데이터베이스를 사용할 수 있는 권리를 부여하는 작업
트랜잭션	안전한 거래 보장	동시에 다수의 작업을 독립적으로 안전하게 처리하기 위한 상호 작용 단위

트랜잭션 제어를 위한 명령어 TCL(Transaction Control Language)이 있다. TCL과 DCL은 대상이 달라 서로 별개의 개념으로 분류할 수 있으나, 제어 기능의 공통점으로 DCL의 일부로 분류하기도 한다. 각 유형별 DCL 명령어는 다음과 같다.

<표 1-17> 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 활용

### 1. 사용자 권한 부여

권한은 시스템 권한과 객체 권한으로 분류한다. 각 권한을 부여하기 위한 명령어 사용법은 다음과 같다.

<표 1-18> 권한 부여 명령어 문법

권한	명령어 문법
시스템 권한	GRANT 권한1, 권한2 TO 사용자계정
객체 권한	GRANT 권한1, 권한2 ON 객체명 TO 사용자계정

시스템 권한과 객체 권한의 종류는 다음과 같다.

<표 1-19> 권한 유형

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

### 2. 사용자 권한 회수

GRANT에 대응하는 권한 회수 명령은 REVOKE이며, 권한 유형별 대응하는 명령어 구조는 다음과 같다.

<표 1-20> 권한 회수 명령어 문법

권한	명령어 문법
시스템 권한	REVOKE 권한1, 권한2 FROM 사용자계정
객체 권한	REVOKE 권한1, 권한2 ON 객체명 FROM 사용자계정

### ③ DCL 이론적 배경인 접근통제

#### 1. 접근 통제 개념

데이터베이스의 보안을 구현하는 방법으로 접근 통제 방법을 사용한다. 접근 통제 정의는 아래와 같다.

<표 1-21> 접근 통제 정의

보안 정책에 따라 접근 객체(시스템 자원, 통신 자원 등)에 대한 접근 주체(사용자, 프로세스 등)의 접근 권한 확인 및 이를 기반으로 한 접근 제어를 통해 자원에 대한 비인가된 사용을 방지하는 정보 보호 기능

출처: TTA. <http://terms.tta.or.kr>에서 2017. 8. 7. 검색.

#### 2. 접근 통제 정책에 따른 유형

접근 통제 정책에는 크게 다음과 같은 2가지 유형이 있다.

<표 1-22> 접근 통제 유형

유형	정의	의미
임의 접근 통제 (DAC: Discretionary Access Control)	<ul style="list-style-type: none"><li>- 시스템 객체에 대한 접근을 사용자 개인 또는 그룹의 식별자를 기반으로 제한하는 방법</li><li>- 여기서 임의적이라는 말은 어떤 종류의 접근 권한을 갖는 사용자는 다른 사용자에게 자신의 판단에 의해서 권한을 줄 수 있다는 것임.</li><li>- 주체와 객체의 신분 및 임의적 접근 통제 규칙에 기초하여 객체에 대한 주체의 접근을 통제하는 기능</li></ul>	<ul style="list-style-type: none"><li>- 통제 권한이 주체에 있음.</li><li>- 주체가 임의적으로 접근 통제 권한을 배분하여 제어할 수 있음.</li></ul>
강제 접근 통제 (MAC: Mandatory Access Control)	<ul style="list-style-type: none"><li>- 정보시스템 내에서 어떤 주체가 특정 객체에 접근하려 할 때 양쪽의 보안 레이블(Security Label)에 기초하여 높은 보안 수준을 요구하는 정보(객체)가 낮은 보안 수준의 주체에게 노출되지 않도록 접근을 제한하는 통제 방법</li></ul>	<ul style="list-style-type: none"><li>- 통제 권한이 제3자에게 있음.</li><li>- 주체는 접근 통제 권한과 무관함.</li></ul>

출처: TTA. <http://terms.tta.or.kr>에서 2017. 8. 7. 검색.

### 3. 접근 통제와 DCL 관계

강제 접근 통제의 경우, 제3자의 종류에 따라 보다 세분화된 정책이 존재한다. 접근 통제 정책의 두 가지 가운데 데이터베이스관리시스템(DBMS)에서 채택한 접근 통제 정책은 임의 접근 통제, DAC 방식이다. 데이터베이스관리, 특히 접근 통제 용도로 SQL에서 사용하는 명령어가 바로 DCL(Data Control Language)인 것이다.

## ④ TCL 활용 방법

### 1. 트랜잭션 개념

트랜잭션은 ‘일 처리 단위’를 의미한다. 보다 다양한 관점에서 트랜잭션은 다음과 같은 모습을 가진다.

- 트랜잭션은 논리적 연산 단위이다.
- 한 개 이상의 데이터베이스 조작이다.
- 즉, 하나 이상의 SQL 문장이 포함된다.
- 트랜잭션은 ‘거래’다.
- 이때 거래 결과가 모두 반영되거나 또는 모두 취소되어야 한다.
- 즉, 데이터베이스에서의 트랜잭션은 특별한, 엄격한 거래를 의미한다.
- 분할할 수 없는 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 1-23> TCL 명령어

명령어	내용	비고
COMMIT	거래 내역을 확정함.	
ROLLBACK	거래 내역을 취소함.	
CHECKPOINT	저장점 설정	ROLLBACK할 위치를 지정함.

## 수행 내용 / DCL 활용하기

### 재료 · 자료

- 사용자 계정 정보
- 사용자 권한 설정 시나리오
- 트랜잭션 시나리오
- SQL 언어 문법 학습 자료
- DCL 및 TCL 활용 가이드

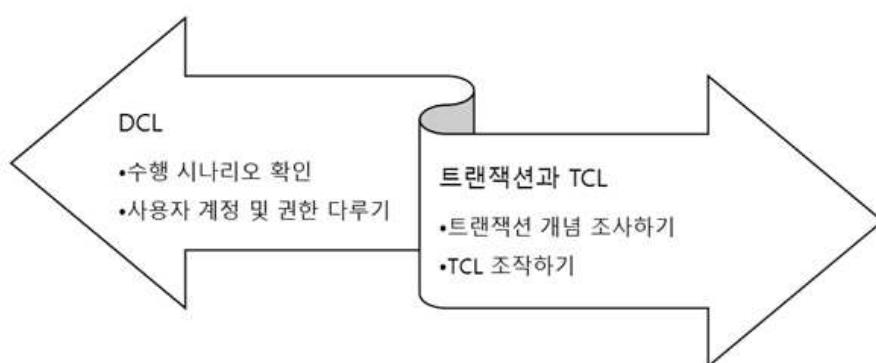
### 기기(장비 · 공구)

- DBMS(DataBase Management System) 프로그램

### 안전 · 유의 사항

- 사용자 권한 관리 방법의 배경 지식인 접근 통제에 대한 사전지식이 필요하다.
- 트랜잭션에 대한 개념과 특성에 대한 이해가 필요하다.
- 간접 홍보 효과를 방지하기 위하여 특정 제조사나 특정 제품에 의존적인 내용은 최소화하였으며, 제품에 특화된 내용을 구분하기 위하여 제품의 약어를 사용하여 표기하였다.
- 제품 약어 M\*-SQL과 O\*는 대표적인 비상용 제품과 상용 제품을 의미한다.

### 수행 순서



[그림 1-8] DCL 활용을 위한 수행내용

① 사용자 권한 관련 DCL을 시나리오에 따라 수행한다(M\*-SQL 기준).

1. DCL 수행 시나리오를 확인한다.

사용자 권한 제어에 대한 세부 수행 학습 내용은 다음과 같다.

- 사용자 계정 추가 및 제거
- 사용자에게 권한 부여
- 사용자의 권한 회수

2. 사용자 계정을 추가한다.

DBMS는 다수의 사용자가 공유하는 조직의 운영 데이터를 관리하는 시스템이다. 다수의 사용자를 지원하기 위하여 사용자 계정 정보를 관리한다. 이후 특정 사용자에게 제한된 권한을 부여하는 방식의 접근 통제를 통하여 데이터베이스에 대한 보안이 가능하다.

우선 사용자 계정을 다음과 같이 만들 수 있다.

```
CREATE USER super@localhost IDENTIFIED BY 'password';
```

여기서 super는 사용자 계정을, password는 사용자 비밀번호를 의미한다. 사용자 계정을 만들고 다음과 같이 계정 정보를 확인할 수 있다.

```
SHOW GRANTS FOR super@localhost;
```

그 결과는 다음과 같다. 아래 그림에서 USAGE는 접속 권한을 의미한다. 즉, 사용자의 super에게는 접속 권한만 주어진 상태이다.

```
> CREATE USER super@localhost IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.01 sec)

> SHOW GRANTS FOR super@localhost;
+-----+
| Grants for super@localhost          |
+-----+
| GRANT USAGE ON *.* TO 'super'@'localhost' |
+-----+
1 row in set (0.00 sec)
```

[그림 1-9] 사용자 계정 생성 후 사용자 권한 보기

다음 명령으로 사용자 계정을 제거할 수 있다.

```
DROP user super@localhost ;
```

### 3. 사용자에게 권한을 부여한다.

다음 명령을 통하여 사용자 super에게 GRANT 권한(권한을 부여할 수 있는 권한)을 부여하였다.

```
GRANT ALL ON *.* TO 'super'@'localhost' WITH GRANT OPTION;
```

SHOW GRANTS를 통한 계정 정보 확인 결과는 다음과 같다.

```
> GRANT ALL ON *.* TO 'super'@'localhost' WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

> SHOW GRANTS FOR super@localhost;
+-----+
| Grants for super@localhost |
+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'super'@'localhost' WITH GRANT OPTION |
+-----+
1 row in set (0.00 sec)
```

[그림 1-10] 권한 부여된 사용자 권한 보기

여기서 ON 절에 사용되는 정보는 database.table 정보를 의미하며, \*.\*은 모든 database의 모든 테이블에 대해 해당 권한이 부여되었음을 알 수 있다.

다른 예시로 다음과 같은 명령문은 super 유저가 biz라는 데이터베이스에 대해 SELECT와 UPDATE 권한만을 주는 경우이다.

```
GRANT SELECT, UPDATE ON biz.* TO super;
```

이 경우, super라는 사용자는 biz 데이터베이스의 모든 테이블에 대해 데이터를 삽입하거나 삭제할 수 없다. 다만 조회와 갱신만 가능하다.

### 4. 사용자 권한을 회수한다.

다음과 같은 명령어를 통해 권한을 회수할 수 있다.

```
REVOKE all ON *.* FROM super@localhost;
```

## ② 트랜잭션 제어 관련 TCL을 수행한다.

### 1. 트랜잭션을 분석한다.

다음과 같은 계좌 이체 과정을 참고하여, 트랜잭션을 분석하도록 한다.

- 내 계좌에서 100원을 인출한다.
- 상대방 계좌에 100원을 입금한다.

계좌 이체 과정은 최소 2번의 작업을 필요로 한다. (두 개의 작업을 하나로 해서 ‘인출하고 입금한다’라는 단위 작업을 말로 정의할 수 있지만, 이는 해당 작업을 한 문장으로 표현한 것이지 결코 하나의 조작이 될 수 없다) 두 개의 작업 도중에 문제가 발생했다면, 즉 1번이나 2번 조작이 실패했다면 어떠한 결과가 발생할까? 상대방 계좌에 근거 없는 100원이 나타나거나, 내 계좌에서 100원이 사라지는 현상이 발생할 것이다. 이러한 문제를 예방하기 위해 트랜잭션을 사용한다. 즉, 트랜잭션은 연관된 다수의 작업을 하나의 단위로 묶어서 하나의 단위가 바로 ‘분해할 수 없는 최소 단위’가 되는 것이다. 트랜잭션은 다음과 같은 특성을 가진다.

<표 1-24> 트랜잭션의 특성

특성	내용	비고
원자성 (Atomicity)	트랜잭션 안에 정의된 연산은 모두 실행되거나 모두 실행되지 않아야 함.	All or Nothing
일관성 (Consistency)	트랜잭션 실행 전과 후 동일하게 오류가 없어야 함.	무결성
고립성 (Isolation)	트랜잭션 실행 중 다른 트랜잭션에 영향을 받지 않아야 함.	독립성
지속성 (Durability)	트랜잭션 결과는 항상 보존됨.	장애 대응성

데이터베이스에서 일관성은 상태가 항상 같아야 함을 의미한다. 이때 상태라는 것은 이전에 오류가 있었으면 이후에도 오류가 있어야 하며, 이전에 오류가 없었으면 이후에도 오류가 없어야 함을 의미한다. 이 가운데 오류가 없는 상태를 무결성이라고 하며, 무결성은 정확성, 일관성, 유일성, 신뢰성 있는 상태를 한 번에 표현하는 것이다. 언제 어떠한 형태의 실패에도 안전한 거래를 보장하는 수단이 바로 트랜잭션이다.

## 2. 메모리 관점에서 트랜잭션 조작을 확인한다.

트랜잭션의 구현 원리를 이해하기 위하여, 우선 다음과 같은 데이터베이스 관리 방법에 대해 확인한다.

- DBMS의 모든 정보는 하드디스크에 저장된다.
- DBMS에서 이루어지는 모든 조작 또는 연산은 메모리에서 이루어진다.
- 하드디스크에 있는 정보를 메모리로 옮겨서 연산을 수행한다.
- 적당한 시점에 메모리 정보를 하드디스크로 옮긴다.

이와 같은 관리 방법과 동일한 개념으로 TCL의 조작 방법을 메모리 관점에서 이해하면 다음과 같다.

<표 1-25> TCL 명령어의 메모리 관점 동작

명령어	내용	메모리 조작
COMMIT	메모리의 내용을 하드디스크에 저장한다.	영구히 저장
ROLLBACK	메모리의 내용을 하드디스크에 저장하지 않고 버린다.	메모리 내용 무효화
CHECKPOINT	ROLLBACK 범위 설정을 위해 메모리상에 경계를 설정한다.	상태 기억

### 3. 트랜잭션 조작을 확인한다.

다음과 같이 트랜잭션을 설계한 문서를 살펴보자. 연속된 SQL문으로 이루어진 트랜잭션이 현재 6번째 위치까지 실행되고 7번째 순서에서 ROLLBACK을 하려고 한다.

<표 1-26> 트랜잭션 시나리오

순서	명령문	비고
1	트랜잭션 시작	
2	INSERT	
3	SAVEPOINT A	A라는 이름으로 복구 지점 설정
4	UPDATE	
5	SAVEPOINT B	B라는 이름으로 복구 지점 설정
6	DELETE	
7	(현재 위치에서 ROLLBACK 예정)	복구 지점 사용에 따라 결과 달라짐.

7번째 위치에서 ROLLBACK을 할 때 사용 명령에 따라 다음과 같이 결과가 달라진다.

<표 1-27> 트랜잭션 ROLLBACK 경우

7번째 명령문	결과
ROLLBACK	1번째 명령까지 취소됨. 즉, 현재 위치 이전 트랜잭션 처리 내용이 모두 취소됨.
ROLLBACK TO A	1~3까지의 명령은 유효하게 남고, 4~6까지의 내용이 취소됨.
ROLLBACK TO B	1~5까지의 명령은 유효하게 남고, 6 내용이 취소됨.

#### 4. Auto Commit을 이용한다.

TCL을 실무에서 사용하고자 할 경우, Auto Commit에 대한 이해가 필요하다. 대부분의 DBMS 제품이 Auto Commit 기능을 지원하며, MS SQL 경우에는 기본값으로 ‘사용하기’로 설정되어 있다. 이 기능은 트랜잭션 명령문에 COMMIT이 없어도 DML 문이 성공적으로 수행되면 자동으로 COMMIT되며, 또한 DML문이 실패하면 자동으로 ROLLBACK이 되는 기능이다.

M\*-SQL에서 Auto Commit 설정 여부를 확인하려면 다음 명령어를 사용한다.

```
SELECT @@AUTOCOMMIT;
```

이 명령의 결과가 1이면 Auto Commit이 설정된 것을 의미한다. 설정값을 바꾸기 위해 다음 명령을 이용한다(M\*-SQL 경우).

```
SET AUTOCOMMIT = TRUE; -- Auto Commit 설정  
SET AUTOCOMMIT = FALSE; -- Auto Commit 해지
```

#### 5. 트랜잭션을 제어한다(M\*-SQL 기준).

다음과 같은 트랜잭션에 대해 데이터의 변화 모습을 살펴보자.

<표 1-28> 트랜잭션 제어 SQL문

위치	트랜잭션 제어 SQL문
1	START TRANSACTION;
2	savepoint a;
3	INSERT INTO 'salaries' (emp_no, salary, from_date, to_date) VALUES (1001,900 ...
4	savepoint b;
5	UPDATE salaries set salary = 1000 where emp_no = 1001;
6	rollback to b;
7	rollback to a;

각 위치별 수행 결과는 다음과 같다.

위치 1~3 실행 후 결과는 salaries 테이블에 salary 값이 900인 레코드가 하나 추가된다.

```

> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

> savepoint a;
Query OK, 0 rows affected (0.00 sec)

> INSERT INTO salaries VALUES (1002, 900, '2014-07-01', '2015-06-30');
Query OK, 1 row affected (0.00 sec)

> select * from salaries;
+-----+-----+-----+
| emp_no | salary | from_date | to_date   |
+-----+-----+-----+
| 1002   |    900 | 2014-07-01 | 2015-06-30 |
+-----+-----+-----+
1 row in set (0.00 sec)

```

[그림 1-11] 위치 1~3 수행 후 데이터 모습

위치 5의 UPDATE 명령에 의해 salary 값이 1000으로 변경된다. 현재 값은 1000이다.

```

> savepoint b;
Query OK, 0 rows affected (0.00 sec)

> UPDATE salaries set salary = 1000 where emp_no = 1002;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

> select * from salaries;
+-----+-----+-----+
| emp_no | salary | from_date | to_date   |
+-----+-----+-----+
| 1002   |  1000  | 2014-07-01 | 2015-06-30 |
+-----+-----+-----+
1 row in set (0.00 sec)

```

[그림 1-12] 위치 4~5 수행 후 데이터 모습

저장 지점 b로 가서 직전의 UPDATE가 취소되면 salary 값은 900이 된다.

```

> rollback to b;
Query OK, 0 rows affected (0.00 sec)

> select * from salaries;
+-----+-----+-----+
| emp_no | salary | from_date | to_date   |
+-----+-----+-----+
| 1002   |    900 | 2014-07-01 | 2015-06-30 |
+-----+-----+-----+
1 row in set (0.00 sec)

```

[그림 1-13] 위치 6 수행 후 데이터 모습

저장 지점 a로 가서 INSERT를 취소하면 salaries에 데이터가 없는 최초 상태로 돌아가게 된다.

```
> rollback to a;  
Query OK, 0 rows affected (0.01 sec)  
  
> select * from salaries;  
Empty set (0.00 sec)
```

[그림 1-14] 위치 7 수행 후 데이터 모습

### 수행 tip

- 학습 목표는 DCL 명령문을 통해 트랜잭션을 제어하는 것으로 설명하고 있다. 일반적으로 TCL이라고 하는 트랜잭션 제어 기능을 DCL로 통칭하였기 때문이다. DCL과 TCL을 엄밀히 구분해야 하는지에 대해서는 판단하기 어렵다. 분명히 구분해야 할 것 같지만 일반적으로 그렇지 않기 때문이다.
- DCL 명령은 데이터베이스 제품에 따라 가장 변화가 많기 때문에 각 제품의 사용법을 확인해야 한다.

# 1-4. 데이터 사전 검색

## 학습 목표

- 생성된 테이블의 목록, 테이블의 구조와 제약 조건을 파악하기 위해 데이터 사전을 조회하는 명령문을 작성할 수 있다.

## 필요 지식 /

### ① 데이터 사전

#### 1. 데이터 사전 개념

데이터 사전(Data Dictionary)에는 데이터베이스의 데이터를 제외한 모든 정보가 있다. 데이터 사전의 내용을 변경하는 권한은 시스템이 가지며, 사용자에게는 읽기 전용 테이블 형태로 제공되므로 단순 조회만 가능하다.

데이터를 제외한(데이터를 구성하는) 모든 정보라는 것은 데이터의 데이터를 의미한다. 따라서 데이터 사전은 메타데이터(Meta data)로 구성되어 있다고 할 수 있다.

#### 2. 데이터 사전 내용

데이터 사전 안에 존재하는 메타데이터의 유형은 다음과 같다.

- 사용자 정보(아이디, 패스워드 및 권한 등)
- 데이터베이스 객체 정보(테이블, 뷰, 인덱스 등)
- 무결성 제약 정보
- 함수, 프로시저 및 트리거 등

데이터 사전 내용이 메타데이터라는 것은 모든 DBMS 제품에 공통이지만 데이터 사전을 구현하는 방법, 관리하는 방법 등의 차이로 메타데이터의 구체적인 내용은 제품마다 다르다.

#### 3. 데이터 사전 용도

사용자에게 데이터 사전은 단순 조회의 대상일 뿐이다. 하지만 데이터베이스 엔진을 이루는 컴파일러, 옵티마이저 등과 같은 구성 요소에 데이터 사전은 작업을 수행하는 데 필요한 참조 정보일 뿐만 아니라 작업의 대상이기도 하다.

### ② 데이터 사전 검색

#### 1. 오\*에서 데이터 사전 검색

오\* 사용자는 뷰로 데이터 사전에 접근할 수 있다. 오\*에서 데이터 사전과 관련된 뷰는 세 가지 영역이 있으며, 이때 오브젝트에 접근할 수 있는 사용자 권한에 따라 다음과 같이 구분된다.

## DBA\_ > ALL\_ > USER\_

오\*에서는 이와 같은 영역 지시자 뒤에 오브젝트명을 붙이는 형태로 뷰의 이름이 결정된다. 여기서 오브젝트는 테이블, 뷰, 인덱스와 같은 것을 의미하므로 다음과 같은 영역별 조회문 구성이 가능하다.

<표 1-29> 오\* 데이터 사전 영역

영역	검색 범위	데이터 사전 검색 쿼리문(예시)
DBA_	데이터베이스의 모든 객체 조회 가능 (DBA_는 시스템 접근 권한 의미)	<code>select * from DBA_TABLES</code> <code>select * from DBA_INDEXES</code> <code>select * from DBA_VIEWS</code>
ALL_	자신의 계정으로 접근 가능한 객체와 타 계정의 접근 권한을 가진 모든 객체 조회 가능	<code>select * from ALL_TABLES</code> <code>select * from ALL_INDEXES</code> <code>select * from ALL_VIEWS</code>
USER_	현재 자신의 계정이 소유한 객체 조회 가능	<code>select * from USER_TABLES</code> <code>select * from USER_INDEXES</code> <code>select * from USER_VIEWS</code>

## 2. M\*-SQL에서 데이터 사전 검색

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

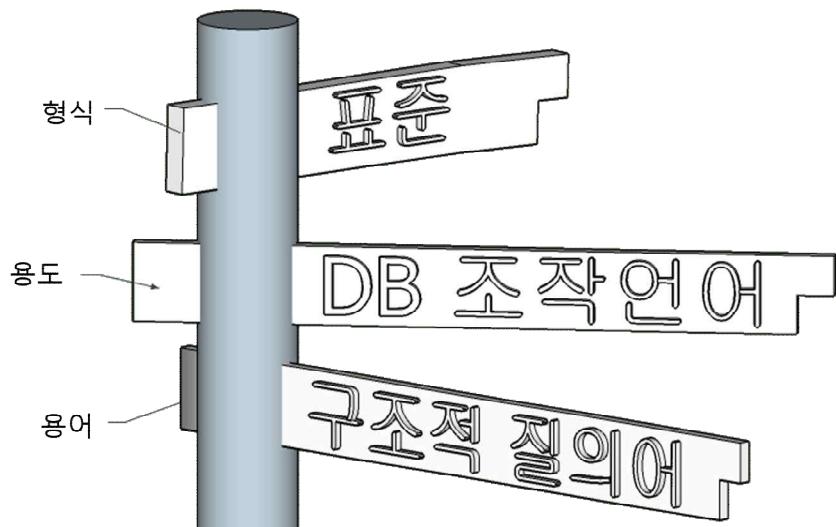
테이블 목록으로 데이터 사전을 구성하는 테이블 이름을 확인하고, SELECT 문을 통해 해당 테이블의 내용을 조회할 수 있다.

### ③ SQL 개요

이제까지 기본적인 SQL 활용 방법에 대해 알아보았다. DDL, DML, DCL 각 용도에 따라 SQL 명령문의 사용법을 살펴보며, 데이터베이스 제품에 따라 구체적인 SQL 사용 방법에 차이가 있는 것을 역시 살펴보았다. 왜 이러한 현상이 발생하는지에 대한 이해를 위해 SQL에 대해 알아볼 필요가 있다. 또한 SQL 자체에 대한 이해는 향후 학습 과정에 도움이 되기에 이러한 과정은 반드시 필요하다.

#### 1. SQL을 바라보는 관점

SQL을 바라보는 관점마다 SQL에 대한 다양한 정의가 가능하다. 우선 SQL을 바라보는 관점에 대해 알아본다.



[그림 1-15] SQL을 바라보는 관점

다양한 관점 가운데 위 그림과 같은 3가지 관점에 따른 정의 내용은 다음과 같다.

<표 1-30> SQL을 바라보는 관점별 정의

관점	SQL 의미	시사점
용어	구조적 질의 언어	구조적 개념에 대한 이해 필요
용도	데이터베이스 조작 언어	데이터베이스에 대한 이해 필요
형식	ISO/ANSI 표준으로 정의됨	표준의 변화에 대한 이해 필요

## 2. 용어 관점에서의 SQL

SQL은 Structured Query Language를 의미하며, 우리말로 옮기면 ‘구조적 질의어’가 된다, 구조적 질의어에 대한 다음과 같은 주장을 살펴보자.

- 구조적 질의어라는 용어를 직접 이해하기는 힘들다.
- 우선 분해해서 해석해 보자.
- 분해하면 ‘구조적’과 ‘질의어’라는 용어로 분리된다.
- 상대적으로 ‘질의어’라는 용어는 이해가 용이하다.
- 물어본다는 뜻으로 단순화시킬 수 있다.
- 그런데 ‘구조적’으로 물어본다는 것이 무슨 말인지 이해가 어렵다.
- 이 경우 반대 또는 상대 개념의 용어를 통해 이해하는 것이 적절하다.
- ‘구조적’의 상대되는 용어는 무엇일까?
- 비구조적은 아니다.
- 구조적의 상대되는 용어는 ‘절차적’이라는 용어가 적절하다.

즉, 구조적 질의어라는 말은 ‘非절차적 질의어’라는 것으로, 절차적으로 세세하게 묘사하는 것이 아니라는 뜻이다. 그 어떤 절차가 아닌 수학식과 같은 하나의 표현으로 목적하는 대상을 표현하는 수단으로 이해하는 것이 적절하다.

## 3. 표준 관점에서의 SQL

SQL은 데이터베이스를 조작하는 언어의 표준이다. SQL에 대한 표준은 ANSI(America National Standard Institute)라는 기관이 최초로 만들었으며, 이후 국제 표준화 기구(ISO: International Standard Organization)를 통해 관리되고 있다.

지속적으로 SQL 표준은 개선되고 있으며 다음과 같은 중요 표준 과정을 가지고 있다.

<표 1-31> SQL 표준화 과정

표준 명칭	특징	비고
SQL-86	ANSI에 의한 최초의 표준화	SQL1
SQL-89	폭발적 전성기 시대의 표준 벤더별 문법, 용어 차이로 상호 호환성에 문제 있음.	SQL2
SQL:1999	객체 지향 지원 SQL3 최초 상용화 버전은 오라클 8i	SQL3
SQL:2003	XML 관련 특징 도입 현재 출시 제품, 최소한의 데이터베이스 표준	SQL2003

표준 SQL:2003 이후에도 SQL2006, SQL2008 및 SQL2011 등이 추가되었으나, 현재 대부분의 DBMS 제품은 SQL:2003 표준을 지원하고 있다. SQL2 표준을 구현한 제품은 관계형 데이터베이스를 지원한다. 이러한 관점에서 SQL:2003 구현 제품은 SQL2 이후에 추가된 객체 지향과 XML 기능을 지원하고 있으므로 ‘객체 관계형 데이터베이스’ 제품이라고 한다. 즉, 현존하는 대부분의 DBMS는 객체를 지원하고 있다. 하지만 아직 객체의 사용은 활성화되지 않은 상태이며, 객체 관계형 데이터베이스를 도입하고 객체를 뺀 관계형 데이터베이스로 사용하고 있는 것이 지금의 현실이다. 따라서 본 학습모듈, SQL 활용의 내용은 SQL2를 기준으로 한다.

#### 4. 용도 관점에서의 SQL

‘SQL은 데이터베이스를 조작하는 언어다.’라고 SQL을 정의할 수 있다. 그렇다면 데이터베이스와 데이터베이스관리시스템이 무엇인지 알아보도록 하자.

##### (1) 데이터베이스 개념

데이터베이스는 ‘연관된 데이터의 모음’이라 이해할 수 있으며, 보다 염격하게 ‘데이터를 일정한 형태로 저장해 놓은 것’을 의미한다.

이러한 데이터 모음을 조작하는 것이 SQL이다. 하지만 SQL이 직접 데이터를 조작하는 것이 아니라, SQL과 데이터베이스의 중간에 데이터베이스관리시스템을 통해 데이터베이스를 조작하게 된다.

과거에는 데이터베이스 시스템 이전에 파일 시스템을 이용하여 데이터를 관리하였다. 파일 시스템을 통하여 데이터의 공유는 가능하였지만, 동시에 데이터의 입력, 수정 및 삭제와 같은 조작이 불가능하고 동일한 내용이 복사되어 사용, 관리되는 문제를 해결 할 수 없었다. 이와 같은 문제를 해결하기 위하여 파일 관리 시스템이 아닌 파일 시스템을 개선한 데이터 관점의 데이터베이스관리시스템(DBMS)이 탄생하게 된 것이다.

##### (2) 데이터베이스 관리시스템

데이터베이스관리시스템은 다음과 같은 관점에서 이해할 수 있다.

<표 1-32> 데이터베이스 정의

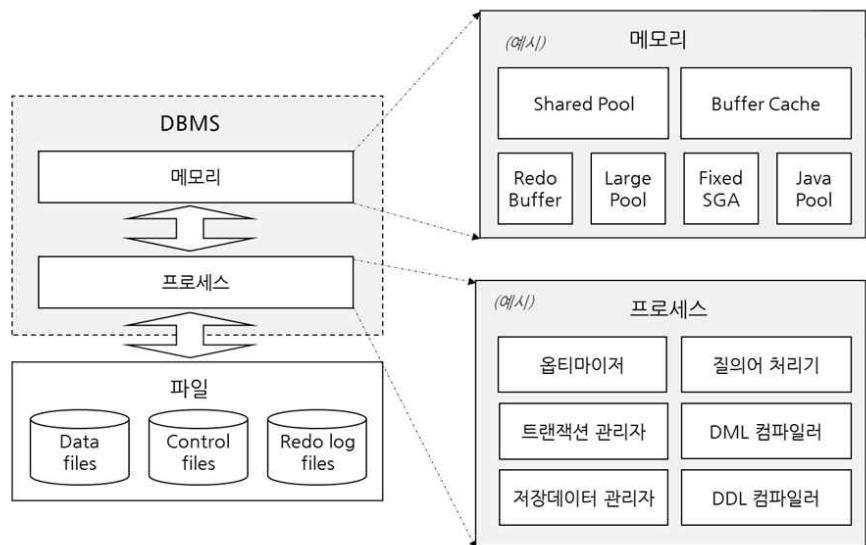
관점	내용	비고
저장 데이터	컴퓨터를 통해 접근 가능한 저장 매체에 저장된 데이터	Stored Data
통합 데이터	중복이 최소화된 데이터	Integrated Data
공유 데이터	여러 응용 프로그램이 공동으로 사용하는 데이터	Shared Data
운영 데이터	조직의 목적을 위해 존재 가치가 확실하고 반드시 필요한 데이터	Operational Data

이들 관점을 이용하여 DBMS를 정의하면, ‘조직의 목적을 위해 존재하는 운영 데이터를 통합 저장하여 공동으로 사용 가능하도록 관리하는 시스템’이라고 할 수 있다.

### (3) 데이터베이스 관리시스템 구조

DBMS는 대용량 공유 메모리와 프로세스로 구성된다. DBMS를 구성하는 메모리는 용도별 공유 영역을 구분하며, 기능별 프로세스가 이를 메모리 영역을 이용하는 구조이다. 공유 메모리 영역이나 프로세스 이름은 제품별 또는 버전별 차이가 있을 수 있지만, 일반적인 구성으로 공유 메모리와 용도별 프로세스로 구성되어 있다는 사실을 알 수 있다. 이와 같이, 메모리와 프로세스로 구성된 DBMS의 모습은 실행 중에 형성되기에 DBMS ‘인스턴스’라는 용어가 사용된다. 즉, DBMS ‘인스턴스’는 메모리와 프로세스로 구성되어 있다.

DBMS는 디스크에 저장된 데이터를 관리하는 시스템이다. 파일 또는 파일 시스템 자체는 DBMS에 포함되지 않으나, 이를 관리하는 ‘저장 데이터 관리자’와 같은 기능을 통해 파일을 관리한다. 일반적인 DBMS의 구조는 다음과 같다.



[그림 1-16] 데이터베이스 구조도

앞서 SQL은 ‘데이터베이스를 조작하는 언어’라는 정의와 조작하는 과정에 개입하는 데이터베이스관리시스템에 대해 알아보았다. 다시 이 개념을 통합하여 SQL은 ‘DBMS를 이용하여 DB를 조작하기 위한 그리고 언어 형태를 가진 수단이다.’라고 정의할 수 있다.

## 수행 내용 / 데이터 사전 검색하기

### 재료 · 자료

- SQL 언어 문법 학습 자료
- DBMS(DataBase Management System) 매뉴얼 및 튜토리얼 자료

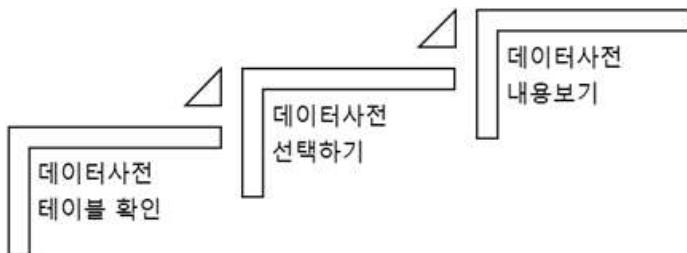
### 기기(장비 · 공구)

- DBMS(DataBase Management System) 프로그램

### 안전 · 유의 사항

- 데이터 사전 용도에 대한 추가 학습이 필요하다.
- 데이터베이스 제품별 사용 방법의 차이가 크다. 특히 데이터사전이란 개념을 구현하고 사용하는 방식에 있어 SQL 사용법은 제품별 큰 차이를 가지고 있는데, 각 제품별 데이터사전 활용방법에 대한 학습이 별도로 필요하다.

### 수행 순서



[그림 1-17] 데이터사전 검색 수행내용

#### ① 데이터 사전을 검색한다(M\*-SQL 기준).

##### 1. 데이터베이스를 선택한다.

데이터 사전 테이블을 관리하는 데이터베이스로 이동하여 테이블의 목록을 살펴보면 다음 그림과 같은 결과를 얻을 수 있다.

```

> use Information_schema;
Database changed
> show tables;
+-----+
| Tables_in_information_schema |
+-----+
| CHARACTER_SETS
| COLLATIONS
| COLLATION_CHARACTER_SET_APPLICABILITY
| COLUMNS
| COLUMN_PRIVILEGES
| ENGINES
| EVENTS
| FILES
.....

```

[그림 1-18] 데이터 사전 테이블 목록

## 2. 주요 데이터 사전 테이블을 살펴본다.

데이터 사전을 구성하는 주요 테이블의 핵심 주제는 다음과 같다.

<표 1-33> 데이터 사전 주요 테이블의 핵심 주제

데이터 사전 테이블	주요 내용
CHARACTER_SETS	사용 가능한 모든 문자 셋에 대한 정보
COLLATIONS	콜레션은 데이터베이스에 저장된 값들을 비교, 검색하거나 정렬 등의 작업을 위해 문자들을 서로 비교할 때 사용하는 규칙들의 집합. 사용 가능한 모든 콜레션에 대한 정보
COLUMNS	테이블 컬럼에 대한 콜레션 정보
COLUMN_PRIVILEGES	테이블 컬럼 권한에 대한 정보 제공
KEY_COLUMN_USAGE	제약사항을 가지고 있는 키 컬럼에 대한 정보 제공
ROUTINES	스토어드 루틴이란 DB상에 저장된 SQL 구문. 스토어드 루틴에 대한 정보
SCHEMATA	하나의 스키마는 하나의 데이터베이스로, SCHEMATA는 데이터베이스의 정보 제공
TABLES	데이터베이스에 존재하는 테이블에 대한 정보 제공
TABLE_CONSTRAINTS	테이블에 대한 제약사항에 대한 정보
TRIGGERS	트리거에 대한 정보 제공(트리거란 테이블에 대한 이벤트에 반응하여 자동으로 실행되는 작업을 의미)
VIEWS	DB에 있는 뷰에 대한 정보 제공

### 3. 데이터 사전 내용을 검색한다.

다음과 같은 SQL문을 통해 데이터 사전을 구성하는 개별 테이블에 대한 내용을 확인할 수 있다.

```
SELECT * FROM tables;
```

FROM 절에 사용된 tables라는 테이블은 다음 그림과 같이 데이터베이스의 모든 테이블 목록을 제공한다.

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE	ENGINE	VERSION	ROW_FORMAT	TABLE_ROWS
def	employees	주문테이블	BASE TABLE	InnoDB	10	Dynamic	0
def	employees	current deot emo	VIEW	NULL	NULL	NULL	NULL
def	employees	deoarments	BASE TABLE	InnoDB	10	Dynamic	9
def	employees	deot emo	BASE TABLE	InnoDB	10	Dynamic	331570
def	employees	deot emp latest date	VIEW	NULL	NULL	NULL	NULL
def	employees	deot manaoer	BASE TABLE	InnoDB	10	Dynamic	24
def	employees	emolovees	BASE TABLE	InnoDB	10	Dynamic	299113
def	employees	salaries	BASE TABLE	InnoDB	10	Dynamic	2838427
def	employees	titles	BASE TABLE	InnoDB	10	Dynamic	442248
def	mvsal	columns priv	BASE TABLE	MvISAM	10	Fixed	0
def	mvsal	db	BASE TABLE	MvISAM	10	Fixed	1
def	mvsal	enaine cost	BASE TABLE	InnoDB	10	Dynamic	2

[그림 1-19] 데이터 사전 가운데 tables 정보 조회 결과

### 수행 tip

- ‘데이터 사전 검색’에서 DBMS 구조를 살펴보는 이유는 트랜잭션, 인덱스, 복구와 같은 SQL 활용에서 DBMS의 구조가 해당 기술의 의도나 방법을 결정하는 데 영향을 끼치기 때문이다.
- 데이터 사전의 개념은 동일하나 구현 결과는 데이터베이스 제품마다 차이가 크다. 데이터 사전 설계 구조의 조그만 차이가 실제 사용 단계 활용 방법에서는 커다란 차이로 나타난다.
- 다른 유형의 SQL 명령문과 달리 데이터 사전에 대한 조회 방법은 데이터베이스 제품마다 전혀 다른 방법으로 내용을 조회하고 이해해야 한다.

## 학습 1 교수 · 학습 방법

### 교수 방법

- 교수자는 DDL 개념과 명령어의 종류에 대해 설명한다.
- 교수자는 DDL 명령어 각각의 사용법을 제시한다.
- 교수자는 DML 개념과 명령어의 종류에 대해 설명한다.
- 교수자는 DML 명령어 각각의 사용법을 제시한다.
- 교수자는 DCL 개념과 명령어의 종류에 대해 설명한다.
- 교수자는 DCL 명령어 각각의 사용법을 제시한다.
- 교수자는 데이터 사전의 개념과 필요성에 대해 설명한다.
- 교수자는 SQL 표준에 대해 설명한다.

### 학습 방법

- 학습자는 DDL 개념과 명령어의 종류에 대해 파악한다.
- 학습자는 DDL 명령어 각각의 사용법을 연습한다.
- 학습자는 DML 개념과 명령어의 종류에 대해 파악한다.
- 학습자는 DML 명령어 각각의 사용법을 연습한다.
- 학습자는 DCL 개념과 명령어의 종류에 대해 파악한다.
- 학습자는 DCL 명령어 각각의 사용법을 연습한다.
- 학습자는 데이터 사전의 개념과 필요성에 대해 파악한다.
- 학습자는 SQL 표준에 대해 파악한다.

## 학습 1 평 가

### 평가 준거

- 평가자는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습 내용	학습 목표	성취수준		
		상	중	하
DDL 활용	- 테이블의 구조와 제약 조건을 생성, 삭제하고 수정하는 DDL(Data Definition Language) 명령문을 작성할 수 있다.			
DML 활용	- 한 개의 테이블에 대해 데이터를 삽입, 수정, 삭제하고 행을 조회하는 DML(Data Manipulation Language) 명령문을 작성할 수 있다.			
DCL 활용	- 업무 단위인 트랜잭션의 완료와 취소를 위한 DCL(Data Control Language) 명령문을 작성할 수 있다.			
데이터 사전 검색	- 생성된 테이블의 목록, 테이블의 구조와 제약 조건을 파악하기 위해 데이터 사전을 조회하는 명령문을 작성할 수 있다.			

### 평가 방법

- 서술형 시험

학습 내용	평가 항목	성취수준		
		상	중	하
DDL 활용	- 데이터 타입별 표현 가능한 값의 범위에 대한 사전 지식 보유 - 테이블 생성 SQL문이 목적하는 테이블을 정확히 생성하고 있는지 평가 - 테이블 생성문 또는 ALTER 문을 통해 제약 조건을 추가할 수 있는지 평가 - 작성한 단건, 대량 데이터 삽입 SQL문이 정상적으로 동작하는지 평가			
DML 활용	- 삽입이나 삭제가 실패하는 경우, 원인 분석 및 대안 제시에 대해 평가 - 원하는 조건의 데이터만 선택적으로 조회할 수 있는 SQL문 작성 능력에 대해 평가			

학습 내용	평가 항목	성취수준		
		상	중	하
DCL 활용	- DCL 필요성에 대해 서술할 수 있는지 평가			
	- 사용자 생성 후 권한 부여할 수 있는지 평가			
데이터 사전 검색	- Auto Commit 설정 변경에 따른 Commit 효과를 구분할 수 있는지 평가			
	- 데이터 사전의 개념과 구성 요소를 서술할 수 있는지 평가			
	- 데이터 사전의 주요 내용을 검색하고 의미를 파악할 수 있는지 평가			

- 사례 연구

학습 내용	평가 항목	성취수준		
		상	중	하
DDL 활용	- 데이터 타입별 값의 범위 조사			
	- 테이블 정의서 조사			
DML 활용	- 제약 조건 표현 방식 조사			
	- 대량 데이터 삽입 SQL문			
DCL 활용	- CASCADE 제약 조건에 따른 삽입, 삭제 불가 상황 조사			
	- 복잡한 조건을 가지는 데이터 SQL문			
데이터 사전 검색	- 접근 통제 개념 조사			
	- 접근 통제 구현 사례 조사			
	- Commit 기능 및 동작 조건 조사			
	- DBMS별 데이터 사전의 구성 요소 조사			
	- DBMS별 데이터 사전의 요소별 내용 조사			

## 피드백

### 1. 서술형 시험

- 기본적인 SQL 명령문의 사용 능력을 평가한 후 미흡한 부분에 대해 설명해 준다. 이때 미흡한 부분은 이해 부족에서 발생하는 것이 대부분이기에, 기본 SQL 작성 주제 영역 별 SQL 사용 방법에 대한 지도 이외에, 배경 지식에 대한 설명이 요구된다.
- 본 학습 모듈에서는 배경지식을 설명하는 의도로 SQL 사용방법 이외에 SQL이 무엇인지 생각해 가는 과정을 제시하였다. 이러한 의도를 참고하여 배경 지식에 대한 설명 혹은 학습이 이루어지도록 한다.

### 2. 사례 연구

- SQL 활용의 배경 설명 또는 심화 학습 방향을 설명해 준다.
- 데이터 타입의 경우, 표현할 수 있는 값의 범위를 설명해 준다.
- 교육용 데이터베이스를 검색한 후 학습자에게 제공, 교육의 밀도를 높인다.

## 2-1. 인덱스 활용

### 학습 목표

- 테이블 조회 시간을 단축하기 위해 사용하는 인덱스의 개념을 이해하고, 인덱스를 생성하는 DDL(Data Definition Language) 명령문을 작성할 수 있다.

### 필요 지식 /

#### ① 인덱스 개요

##### 1. 인덱스 개념

인덱스는 데이터를 빠르게 찾을 수 있는 수단으로서, 테이블에 대한 조회 속도를 높여 주는 자료구조를 일컫는다. 인덱스는 다음 그림과 같이 테이블의 특정 레코드 위치를 알려주는 용도로 사용하는데, 이러한 인덱스는 자동으로 생성되지 않는다.

index_name		table_create_men			
Index (이름)	주소	일련번호	이름	생년월일	출생지
김구	5	1	장보고	07871111	전라남도 완도군
박문수	4	2	홍길동	14431111	전라남도 장성군
윤동주	6	3	이순신	15450428	서울시 중구 인현동
이순신	3	4	박문수	16911111	경기도 평택시
장보고	1	5	김구	18760829	황해도 해주시
홍길동	2	6	윤동주	19171230	중국 연변 연정시

[그림 2-1] 인덱스 개념

PK 컬럼은 PK를 생성할 때 자동으로 인덱스가 생성된다. 즉, PK 컬럼은 PK를 생성할 때 자동으로 PK 인덱스가 생성된다.

예를 들어 위의 그림과 같은 테이블에서 일련번호를 기본키(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’ 컬럼에 인덱스가 없는 경우, 테이블의 전체 내용을 검색(Table full scan)하게 된다.

반면, 인덱스가 생성되어 있다면 테이블의 일부분을 검색(Range scan)하여 데이터를 빠르게 찾을 수 있다. 조건절에 ‘=’로 비교되는 컬럼을 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 2-1]에서 ‘이름’ 컬럼에 대한 인덱스가 생성되어 있다면 데이터를 빠르게 찾을 수 있다. 이때 빠르게 찾는 행위의 주체는 DBMS이다. 즉, DBMS는 인덱스를 사용하여 빠른 검색을 수행한다. 이를 위해 DB 사용자는 DBMS가 인덱스를 사용할 수 있게 준비해 주어야 한다. 따라서 DB 사용자 입장에서는 인덱스를 사용하는 개념보다는 준비하는 개념으로 접근해야 한다. 인덱스 준비 방법에 대해 알아보도록 하자.

### 2. 인덱스 준비

DB 사용자가 인덱스에 대해 조작할 수 있는 방법으로는 ‘생성, 삭제 그리고 변경’ 조작이다. 참고로 인덱스 조작 명령은 SQL 표준화에 포함되지 않아 DBMS 제품 공급사마다 사용법이 약간씩 다르다. 여기서는 각 조작에 대한 개념을 익히도록 하자.

#### (1) 인덱스 생성

인덱스 생성 문법은 다음과 같다.

```
CREATE [UNIQUE] INDEX <index_name> ON <table_name> (<column(s)>);
```

여기서 각각의 파라미터가 의미하는 내용은 다음과 같다.

<표 2-1> 인덱스 명령문 요소

파라미터	내용	비고
[UNIQUE]	인덱스 걸린 컬럼에 중복값을 허용하지 않음 (생략 가능)	CREATE 테이블에서 사용하는 UNIQUE 제약 조건과 동일한 의미
<index_name>	생성하고자 하는 인덱스 테이블 이름	
<table_name>	인덱스 대상 테이블 이름	
<column(s)>	인덱스 대상 테이블의 특정 컬럼 이름(들)	복수 컬럼 지정 가능

## (2) 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 테이블의 변경을 가하는 형식의 명령을 사용한다. 즉, ALTER TABLE 명령 뒤에 DROP INDEX 명령이 추가되는 형태로 사용된다.

## (3) 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.

## 수행 내용 / 인덱스 활용하기

### 재료 · 자료

- 테이블 정의서
- 인덱스 설계서
- SQL 트레이스
- SQL 언어 문법 학습 자료

### 기기(장비 · 공구)

- DBMS(DataBase Management System) 프로그램

### 안전 · 유의 사항

- 인덱스는 목적이 아니라 수단이다.
- 간접 홍보 효과를 방지하기 위하여 특정 제조사나 특정 제품에 의존적인 내용은 최소화하였으며 제품에 특화된 내용을 구분하기 위하여 제품의 약어를 사용하여 표기하였다.
- 제품 약어 M\*-SQL과 O\*는 대표적인 비상용 제품과 상용 제품을 의미한다.
- 실사례 제시를 위하여 수행 내용의 핵심 부분은 명령행 작업 모습으로, 비핵심 부분은 상황 이해를 용이하게 하기 위하여 M\*-SQL 5.7 워크벤치 작업 모습을 사용하였다.

### 수행 순서



[그림 2-2] 인덱스 활용을 위한 수행내용

## ① 인덱스 대상 테이블을 확인한다.

다음과 같은 주문테이블이 있다. ‘주문번호’ 컬럼은 PK이므로 자동으로 인덱스가 생성된다. 하지만 PK 이외의 컬럼에는 인덱스가 생성되지 않은 상태이다. 주문테이블에 인덱스를 생성해 보자.

주문테이블

PK 컬럼

주문번호	고객번호	주문일	주문가격	배송 도시	배송완료일	결제금액	할인금액
A0100	24680	20170704	55,000	서울	20170707	45,000	10,000
X0300	13579	20170801	70,000	부산	Null	50,000	20,000

[그림 2-3] 인덱스 수행에 사용할 테이블 모습

## ② 인덱스를 설계한다.

인덱스 생성 전에 인덱스에 대한 설계가 필요하다. 즉, 인덱스 대상을 결정하는 과정이 필요하다.

### 1. 인덱스 설계 과정을 확인한다.

일반적인 인덱스 설계 과정은 다음과 같다.

<표 2-2> 인덱스 설계 과정

단계	인덱스 설계 과정	내용
1	접근 경로 수집	<ul style="list-style-type: none"><li>- 접근 경로는 테이블에서 데이터를 검색하는 방법을 의미함.</li><li>- 접근 경로 유형은 테이블 스캔과 인덱스 스캔이 있음.</li><li>- 접근 경로 수집 의미는 SQL이 최적화되었을 때 인덱스 스캔을 해야 하는 검색 조건들을 수집하는 것</li></ul>
2	통한 후보 컬럼 선정	<ul style="list-style-type: none"><li>- 수집된 접근 경로에 대해 분포도 조사</li><li>- 분포도 = 데이터별 평균 로 수 / 테이블의 총 로 수</li><li>- 일반적으로 분포도가 10~15% 정도이면 인덱스 컬럼 후보로 사용</li><li>- 인덱스 후보 목록을 이용하여 접근 유형에 따라 어떤 인덱스 후보를 사용할 것인가를 결정</li></ul>
3	접근 경로 결정	<ul style="list-style-type: none"><li>- 만약 누락된 접근 경로가 있다면 분포도 조사를 실시하고 인덱스 후보 목록에 추가 작업을 반복</li></ul>
4	컬럼 조합 및 순서 결정	<ul style="list-style-type: none"><li>- 단일 컬럼의 분포도가 양호하면 단일 컬럼 인덱스로 확정</li><li>- 하지만 하나 이상의 컬럼 조합이 필요한 경우는 추가 고려하여 인덱스 컬럼 순서를 결정</li><li>- 설계된 인덱스를 적용하고 접근 경로별로 인덱스가 사용되는지 시험해야 함.</li></ul>
5	적용 시험	<ul style="list-style-type: none"><li>- 여러 개의 접근 경로가 존재하는 테이블은 여러 개의 인덱스가 만들어지므로 의도한 실행 계획으로 동작하는지 확인해야 함.</li></ul>

출처: 인덱스 설계. DA 가이드(<http://www.dbguide.net>)에서 2017. 06. 24. 검색.

접근 경로라는 것은 ‘검색 조건’을 의미한다. 따라서 접근 경로를 수집한다는 것은 사용하는 질의문의 검색 조건을 살펴보기 위해 수집하는 것이며, 크게 두 가지 방법이 있다.

- 사용하고 있는 애플리케이션 소스 안에 있는 SQL 문장을 수집
- DBMS의 트레이스(Trace) 도구를 사용하여 SQL 문장을 수집

이와 같이 사용하는 SQL 질의문을 수집하고 분석하여 인덱스를 설계한다. 이때 대상 설정과 우선순위 설정에서 다음과 같은 판단 기준을 참고하도록 한다.

### (1) 접근 경로 유형

다음과 같은 분류에 속하는 접근 유형을 인덱스 대상으로 고려한다.

<표 2-3> 인덱스 접근 경로

접근 경로 유형	내용
반복 수행되는 접근 경로	대표적으로 조인 컬럼을 후보로 선택(주문 1건당 평균 50개의 주문 내역을 가진다면 주문테이블과 주문 내역 테이블을 이용하여 주문서를 작성하는 SQL은 조인을 위해 평균 50번의 주문 내역 테이블을 반복 액세스하기 때문)
분포도가 양호한 컬럼	주문 번호, 청구 번호, 주민 번호 등은 단일 컬럼 인덱스로도 충분한 수동 속도를 보장받을 수 있는 후보임.
조회 조건에 사용되는 컬럼	성명, 상품명, 고객명 등 명칭이나 주문 일자, 판매일, 입고일 등 일자와 같은 컬럼은 조회 조건으로 많이 이용되는 컬럼
자주 결합되어 사용되는 컬럼	판매일 + 판매 부서, 급여일 + 급여 부서와 같이 조합에 의해 사용되는 컬럼
데이터 정렬 순서와 그룹핑 컬럼	조건뿐만 아니라 순방향, 역방향 등의 정렬 순서 고려(인덱스는 구성 컬럼 값들이 정렬되어 있어 인덱스를 이용하면 별도의 ORDER BY, 정렬 작업이 불필요함.) 동일한 원리로 그룹핑 단위(GROUP BY)로 사용된 컬럼도 조사
일련번호를 부여한 컬럼	이력을 관리하기 위해서 일련번호를 부여한 컬럼에 대해서도 조사(마지막 일련번호를 찾는 경우가 빈번히 발생하므로 효과적인 액세스를 위해 필요)
통계 자료 추출 조건	통계 자료는 결과를 추출하기 위해서 넓은 범위의 데이터가 필요.(다양한 추출 조건을 사전에 확보하여 인덱스 생성에 반영)
조회 조건이나 조인 조건 연산자	위에 제시되는 유형의 컬럼과 함께 적용된 =, between, like 등의 비교 연산자를 병행 조사하여 인덱스 결합 순서를 결정할 때 중요한 정보로 사용하도록 함.

출처: 인덱스 설계. DA 가이드(<http://www.dbguide.net>)에서 2017. 06. 24. 검색.

### (2) 컬럼 조합 및 순서 결정

복수개의 컬럼(결합 컬럼)에 대해 인덱스를 사용할 때 순서가 중요하다. 인덱스 컬럼의 순서를 결정할 때 선두에 두어야 할 컬럼을 선택하는 판단 기준은 다음과 같다.

<표 2-4> 인덱스 선두 컬럼 결정 요건

선두 컬럼 요건	내용
항상 사용되는 컬럼	컬럼 A, B가 인덱스로 사용될 때 컬럼 A에는 항상 값이 있어야 함(인덱스로 사용될 때 선행되는 컬럼 값이 Null인 경우 인덱스를 이용하지 못함).
등호(=) 조건으로 사용되는 컬럼	부등호나 범위 연산(<, >, <=, >=, BETWEEN, LIKE)보다는 등호(=) 연산을 사용하는 컬럼을 선두에 배치하는 것이 좋음.
분포도가 좋은 컬럼	분포도가 좋다는 의미는 데이터값이 중복이 적은 것을 의미함. 즉, 값의 중복이 적은 컬럼을 선두로 사용하는 것이 좋음.
ORDER BY, GROUP BY 순서	ORDER BY, GROUP BY ORDER BY나 GROUP BY 절에 사용되는 컬럼 순으로 인덱스를 생성하는 것이 좋음.

출처: 인덱스 설계. DA 가이드(<http://www.dbguide.net>)에서 2017. 06. 24. 검색.

## 2. 인덱스 설계 단계를 확인한다.

테이블 구조만 보고 인덱스 대상을 선정할 수는 없다. 해당 테이블을 어떻게 사용하는지 질의문, 특히 검색 조건에 어떠한 컬럼이 사용되는지를 보고 결정해야 한다. 따라서 직접적인 실습 내용은 생략하며, 교수자와 학습자 사이에 해야 할 일을 다음과 같이 제시한다.

### (1) 인덱스 설계해 보기

앞서 수행 순서 ①에서 정의한 ‘주문테이블’에 대해 인덱스를 설계한다.

### (2) 인덱스 설계 필요성

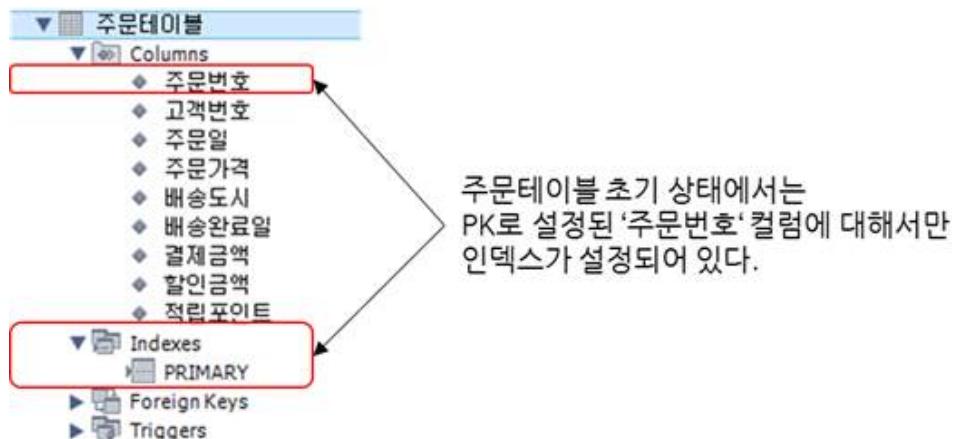
인덱스 설계 과정의 필요성에 대해 논의한다.

## ③ 인덱스를 생성한다.

인덱스 설계 과정을 통해 ‘주문테이블’의 다음과 같은 컬럼에 대해 인덱스가 필요하다고 결정하였다고 하자(수행 순서 ① 참조). 각 경우에 적합한 인덱스를 생성해 보자.

1. ‘배송도시’ 컬럼에 대해 인덱스 생성하기
2. ‘주문일 + 적립 포인트’ 결합 컬럼에 대해 인덱스 생성하기
3. ‘고객번호’ 컬럼에 대해 UNIQUE 인덱스 생성하기

참고로 인덱스를 생성하기 이전 주문테이블 상태는 다음과 같다.



[그림 2-4] 주문테이블 초기 상태

1. '배송도시' 컬럼에 대해 인덱스를 생성한다.

다음과 같은 SQL문을 통해 인덱스를 생성할 수 있다.

```
CREATE INDEX idx배송도시 on 주문테이블 (배송도시);
```

실행 결과 주문테이블의 상태에는 다음과 같은 변화가 생겼음을 확인할 수 있다.



[그림 2-5] 인덱스 생성 후 주문테이블 모습

M\*-SQL에서 인덱스 내용은 다음과 같은 SQL문을 통해 알 수 있다.

```
SHOW INDEX FROM 주문테이블;
```

인덱스 내용은 다음 그림과 같다,

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
주문테이블	0	PRIMARY	1	주문번호	A	0	NULL	NULL		BTREE
주문테이블	1	idx배송도시	1	배송도시	A	0	NULL	NULL	YES	BTREE

[그림 2-6] 인덱스 정보

2. ‘주문일 + 적립포인트’ 컬럼에 대해 인덱스를 생성한다.

다음과 같은 SQL문을 통해 인덱스를 생성할 수 있다.

```
CREATE INDEX idx주문일포인트 on 주문테이블 (주문일, 적립포인트);
```

실행 결과 인덱스의 상태가 다음과 같이 변한 것을 확인할 수 있다.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
주문테이블	0	PRIMARY	1	주문번호	A	0	NULL	NULL		BTREE
주문테이블	1	idx배송도시	1	배송도시	A	0	NULL	NULL	YES	BTREE
주문테이블	1	idx주문일포인트	1	주문일	A	0	NULL	NULL		BTREE
주문테이블	1	idx주문일포인트	2	적립포인트	A	0	NULL	NULL		BTREE

[그림 2-7] 주문일, 적립 포인트 추가 인덱스 정보

3. ‘고객번호’ 컬럼에 대해 UNIQUE 인덱스를 생성한다.

다음과 같은 SQL문을 통해 인덱스를 생성할 수 있다.

```
CREATE UNIQUE INDEX idx고객번호 on 주문테이블 (고객번호);
```

실행 결과 인덱스의 상태가 다음과 같이 변한 것을 확인할 수 있다.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
주문테이블	0	PRIMARY	1	주문번호	A	0	NULL	NULL		BTREE
주문테이블	0	idx고객번호	1	고객번호	A	0	NULL	NULL		BTREE
주문테이블	1	idx배송도시	1	배송도시	A	0	NULL	NULL	YES	BTREE
주문테이블	1	idx주문일포인트	1	주문일	A	0	NULL	NULL		BTREE
주문테이블	1	idx주문일포인트	2	적립포인트	A	0	NULL	NULL		BTREE

[그림 2-8] UNIQUE 인덱스 생성 정보

4. 기타 인덱스 생성 방법을 조사한다.

이제까지는 인덱스를 생성하기 위하여 CREATE INDEX 문을 사용하였다. 인덱스를 생성하는 기타 다른 방법에 대해 조사한다.

#### ④ 인덱스를 삭제한다.

인덱스 생성 과정을 통해 만든 인덱스를 다음 명령을 통해 삭제해 보자.

1. ‘배송도시’ 인덱스를 삭제한다.

```
ALTER TABLE 주문테이블 DROP INDEX idx배송도시;
```

2. ‘주문일 + 적립포인트’ 인덱스를 삭제한다.

```
ALTER TABLE 주문테이블 DROP INDEX idx주문일포인트;
```

3. ‘고객번호’ 인덱스를 삭제한다.

```
ALTER TABLE 주문테이블 DROP INDEX idx고객번호;
```

#### 수행 tip

- SQL 관련 수행은 데이터베이스 및 SQL 언어의 종류에 따라 조금씩 달라질 수 있다. 사용자가 보유한 데이터베이스 및 SQL 언어를 고려하여 수행해야 한다.
- 이제까지 인덱스를 생성하고 삭제하는 방법에 대해 알아보았다. 인덱스는 사용자가 직접 사용하지 않으므로 간접적으로 인덱스의 효용을 느낄 수밖에 없다. 본 수행 내용 과정을 통해 생성한 인덱스를 통해 검색 속도가 빨라졌는지는 상황에 따라 다르다. 그 이유는 축적된 데이터와 이를 사용하는 검색 조건에 인덱스는 의존적이기 때문이다. 인덱스 학습에서 중요한 것은 인덱스 대상을 결정하는 것이다. 이를 위해 ‘인덱스 설계’에 대한 필요와 이해가 가장 중요하며, 또한 인덱스를 둘러싼 관련 주제들 역시 중요하다.
- 스무고개 놀이를 통해 어떤 숫자를 찾아가는 과정을 생각해 보자. 가장 효과적인 방법은 중간값을 선택하는 것이다. 컴퓨터에서 이와 같은 방식의 해결 기법을 ‘이진 검색’이라고 한다. 데이터베이스에서 정보를 찾아가는 방법 역시 이진 검색을 응용한 것이다. B트리라는 자료구조를 통해 분기 횟수를 최소화하여 찾아간다. B트리에는 분기를 할 수 있는 정보가 있으며, 그 정보가 바로 인덱스이다.

## 2-2. 뷰 활용

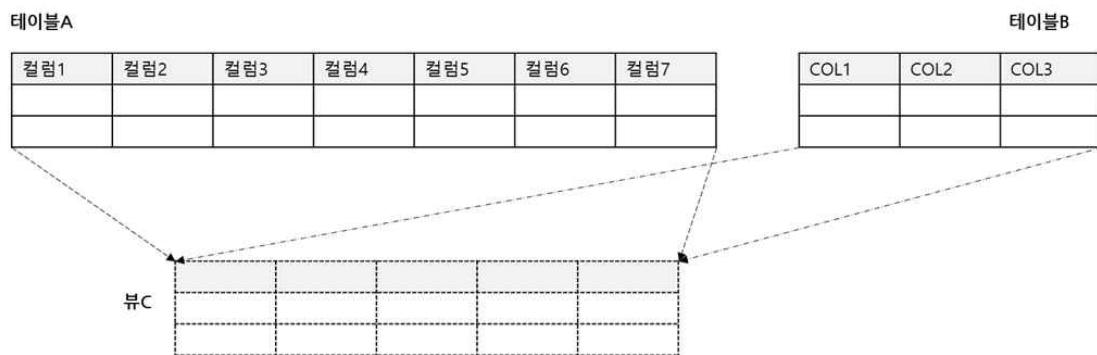
### 학습 목표

- 먼저 생성된 테이블들을 이용하여 새로운 테이블과 뷰를 생성하는 DDL(Data Definition Language) 명령문을 작성할 수 있다.

### 필요 지식 /

#### ① 뷰(View) 개요

뷰는 논리 테이블로서 사용자에게(생성 관점 아닌 사용 관점에서) 테이블과 동일하다. 아래 그림에서 ‘테이블A’와 ‘테이블B’는 물리 테이블을 의미하고, ‘뷰C’는 두 개의 테이블을 이용하여 생성한 뷰를 의미한다.



[그림 2-9] 뷰 개념도

뷰는 ‘테이블A’와 같은 하나의 물리 테이블로부터 생성 가능하며, 다수의 테이블 또는 다른 뷰를 이용해 만들 수 있다. 위 그림의 뷰와 같은 결과를 만들기 위해 다음 장에서 배울 조인(Join) 기능을 활용할 수 있으나, 뷰가 만들어져 있다면 사용자는 조인 없이 하나의 테이블을 대상으로 하는 단순한 질의어를 사용할 수 있다.

#### ② 뷰 활용 상세

##### 1. 뷰 사용

뷰를 사용하는 주된 이유는 다음과 같은 단순한 질의어를 사용할 수 있기 때문이다.

```
SELECT * FROM <View Name>;
```

즉, FROM 절에 있는 하나의 <뷰>를 통해 뷰를 구성하는 복수의 <테이블>을 대체하는 단순성에 그 의의가 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 테이블임(테이블의 구조가 변경되어도 뷰를 사용하는 응용 프로그램은 변경하지 않아도 됨).
사용자 데이터 관리 용이	복수 테이블에 존재하는 여러 종류의 데이터에 대해 단순한 질의에 사용이 가능함.
데이터 보안 용이	중요 보안 데이터를 저장 중인 테이블에는 접근 불허하고, 해당 테이블의 일부 정보만을 볼 수 있는 뷰에는 접근을 허용하는 방식으로 보안 데이터에 대한 접근 제어 가능

반면에 다음과 같은 단점이 있다.

<표 2-6> 뷰의 단점

뷰 단점	내용
뷰 자체 인덱스 불가	인덱스는 물리적으로 저장된 데이터를 대상으로 하기에 논리적 구성인 뷰 자체는 인덱스를 가지지 못함.
뷰 정의 변경 불가	뷰의 정의를 변경하려면 뷰를 삭제하고 재생성하여야 함.
데이터 변경 제약 존재	뷰의 내용에 대한 삽입, 삭제, 변경 제약이 있음.

뷰를 사용하기 위해서는 우선 뷰를 만들어야 한다. 뷰의 단순한 사용은 생성 과정에 의존적이다. 즉, 뷔를 어떻게 생성하였느냐에 따라 사용 방법이 달라진다. 뷔의 생성 방법에 대해 알아보도록 하자.

## 2. 뷔 생성

뷰 생성 명령의 일반 형태는 다음과 같다.

```
CREATE VIEW <뷰이름>(컬럼목록) AS <뷰를 통해 보여줄 데이터 조회용 쿼리문>
```

앞의 [그림 2-9]에서 상황별 뷔를 생성하는 방법은 다음과 같다.

<표 2-7> 뷔 생성 방법

상황	뷰 생성 쿼리문
테이블A 그대로	CREATE VIEW 뷔A AS select * from 테이블A;
테이블 A 일부 컬럼	CREATE VIEW 뷔X AS select 컬럼1,컬럼2,컬럼3 from 테이블A;
테이블A와 테이블B 조인 결과	CREATE VIEW 뷔Y AS select * from 테이블A a, 테이블B b where a.컬럼1=b.COL1;

조회문의 다양한 변형에 따라 뷰의 내용이 달라진다. 즉, [그림 2-9] 경우 ‘테이블A’와 ‘테이블B’로부터 조회 가능한 형태 모두가 뷰로 치환될 수 있다.

### 3. 뷰 삭제 및 변경

뷰 정의 자체를 변경하는 것은 불가능하다. 일단 뷔를 정의하면, 뷔의 물리적 내용은 뷔의 이름과 데이터를 조회하기 위한 쿼리문뿐이다. 이때 뷔의 이름이나 쿼리문을 변경하는 수단은 제공되지 않는다. 이 경우 뷔의 삭제와 재생성을 통해 뷔에 대한 정의 변경이 가능하다. 뷔를 삭제하는 쿼리문은 다음과 같다.

```
DROP VIEW <View Name>;
```

### 4. 뷔 내용 변경

뷰를 통해 접근 가능한 데이터에 대한 변경이 가능하다. 하지만 모든 경우에 데이터의 변경이 가능한 것이 아니라 일부 제약이 존재한다. 이러한 제약은 뷔 자체가 논리적 개념이기에 물리적 상황에 의존적임을 의미한다. 예를 들어 PK에 해당하는 컬럼이 뷔에 정의되어 있지 않은 경우 INSERT는 당연히 불가능하다.

## 수행 내용 / 뷰 활용하기

### 재료 · 자료

- 뷰 명세서
- E-R 모델링 다이어그램

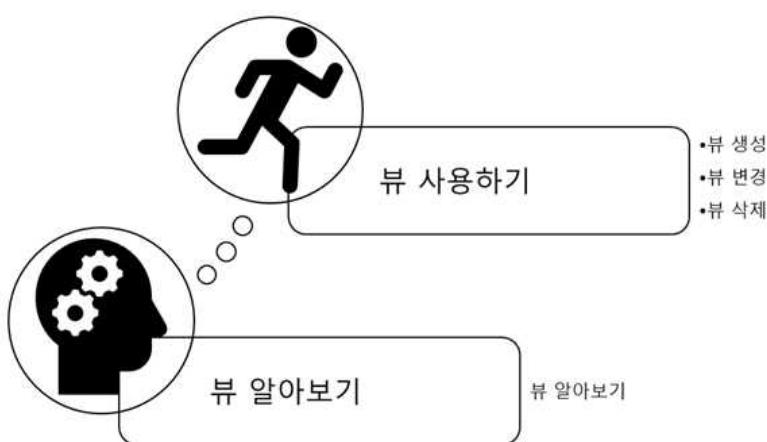
### 기기(장비 · 공구)

- DBMS(DataBase Management System) 프로그램

### 안전 · 유의 사항

- 쿼리문, 조회문 그리고 SQL문 모두 같은 의미로 사용 중이다.
- 간접 홍보 효과를 방지하기 위하여 특정 제조사나 특정 제품에 의존적인 내용은 최소화하였으며, 제품에 특화된 내용을 구분하기 위하여 제품의 약어를 사용하여 표기하였다.
- 제품 약어 M\*-SQL과 O\*는 대표적인 비상용 제품과 상용 제품을 의미한다.

### 수행 순서



[그림 2-10] 뷰 활용을 위한 수행내용

#### ① 수행에 사용할 뷰를 확인한다.

인사 부서에 다음과 같은 테이블이 있다. 이와 같은 인사 정보를 기반으로 다양한 업무에 활용 가능한 데이터베이스를 구축하고자 한다.

직원테이블

PK 컬럼

UID	NAME	PASSWORD	POSITION	WORKPLACE	ADDRESS	BIRTHDAY	JOINDAY	WEDDINGDAY
1010	홍길동	****	대표	서울	서울시 종로구 ...	19700707	2000.01.01	1999.09.09
1090	나길동	****	부장	대전	서울시 구로구 ...	19880808	2010.10.10	2012.12.12

[그림 2-11] 뷰 수행에 사용할 데이터

데이터가 채워진 테이블 모습은 다음과 같이 바커(Baker)식 표기법의 ER 다이어그램으로 표현할 수 있다. 보통 ERD 형태의 테이블 정의서를 보고 쿼리문을 설계하는 것이 일반적이다.



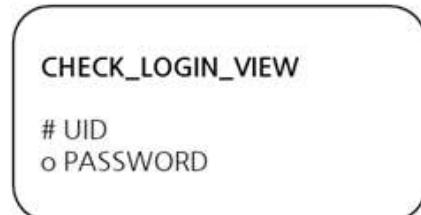
[그림 2-12] 개념 데이터 모델링 모습

쿼리문 설계 때 참조할 ERD는 논리 모델링 결과를 주로 이용하지만, 본 수행 내용에서는 개념 모델링 결과를 이용한다. 그 이유는 데이터 타입과 같은 정보가 필요하지 않기 때문이다.

## ② 뷰를 생성한다.

### 1. 특정 컬럼으로 구성된 뷰를 생성한다.

로그인 기능이 필요한 응용 프로그램은 직원 테이블에 정의된 내용 가운데 PASSWORD 관련 정보만 필요로 한다. 즉, 직원 테이블에 있는 수많은 개인 정보 가운데 다음과 같은 정보만을 해당 응용 프로그램에 제공하고자 할 경우 뷰를 생성해 보자.



[그림 2-13] 직원 테이블의 UID, PASSWORD 컬럼만으로 구성된 뷰

다음 쿼리문을 이용하여 뷰를 생성한다.

```
CREATE VIEW CHECK_LOGIN_VIEW AS select UID, PASSWORD from EMP;
```

이와 같이 생성된 뷰에 대해 응용 프로그램은 다음과 같은 쿼리문을 사용할 수 있다.

```
SELECT UID FROM CHECK_LOGIN_VIEW WHERE UID='user', PASSWORD='pw'
```

여기서 'user'와 'pw'는 사용자가 입력한 정보를 의미한다. 그리고 이와 같은 쿼리문의 결과로 반환되는 결과가 존재하면 로그인 성공으로 판정할 수 있다.

## 2. IN 조건절로 이루어진 뷰를 생성한다.

뷰의 원천 정보를 다양한 형태로 조회할 수 있다. 이러한 조회 형태 가운데 IN 조건절을 가지는 뷰를 생성해 보자. 뷰의 모습은 다음과 같다.

```
CHECK_WORKPLACE_VIEW  
# UID  
* NAME  
* WORKPLACE
```

[그림 2-14] 근무지 파악을 위한 뷰

다음 쿼리문을 이용하여 뷰를 생성한다.

```
CREATE VIEW CHECK_WORKPLACE_VIEW  
AS select UID, NAME, WORKPLACE from EMP  
WHERE WORKPLACE IN ('서울', '부산');
```

이와 같이 생성된 뷰를 통하여 서울과 부산 지역에 근무하는 직원 정보를 조건절 없는 쿼리문으로 조회할 수 있다.

## 3. 계산 컬럼을 포함하는 뷰를 생성한다.

입사일 이후 오늘까지 며칠이 지났는지를 보여 주는 뷰를 구성해 보자. 뷰의 모습은 다음과 같다.

### CHECK\_WORKDAY\_VIEW

```
# UID  
* NAME  
o JOINDAY  
o WORKDAYS
```

[그림 2-15] 근무 일수를 보여주는 뷰

기준의 뷰와 다른 점은 WORKDAYS 부분이다. WORKDAYS는 직원 테이블 정보에 포함되지 않은 컬럼으로 오늘 날짜와 입사일을 통해 계산하여 보여 주는 컬럼이다. 근무 일수는 개념적으로 다음과 같은 계산식을 가진다.

$$\text{근무일수} = \text{오늘날짜} - \text{입사일자}$$

이와 같은 계산이 뷰 생성문에 포함되도록 한다. 또 데이터의 형식에 주의해야 하는데, 일반적으로 날짜 컬럼은 처리 속도를 높이기 위해 CHAR형을 이용하며, 정확한 정보가 필요한 로그 데이터의 경우 Date 형식을 사용한다. 따라서 직원 테이블 경우 입사일자를 CHAR형으로 사용하였다는 가정에 따라 근무일수를 구하기 위해서는 입사일자의 데이터 형식을 Date 형식으로 바꾸어 계산해야 한다. 또 날짜 차이 결과를 날(day) 단위로 바꾸기 위한 함수를 사용해야 한다. 이와 같은 고려 사항이 포함된 뷰 생성문은 다음과 같다.

```
CREATE VIEW CHECK_WORKDAY_VIEW (UID, NAME, JOINDAY, WORKDAYS)  
AS select UID, NAME, JOINDAY, to_date( Now() - Date(JOINDAY) )  
from EMP;
```

또 WORKDAYS라는 컬럼 이름을 부여하기 위해 뷰를 생성할 때 이를 명시적으로 선언하는 형태로 뷰를 생성한다.

### ③ 뷰 내용을 변경한다.

사용자에게 ‘뷰는 테이블’이라고 하였다. 따라서 뷰의 내용을 변경하기 위해서는 UPDATE 문을 이용한다. 또 ‘뷰는 논리 테이블’이라고 하였다. 이와 같은 ‘논리’라는 제약에 의해 뷰의 내용에 대한 변경이 불가능한 경우가 발생한다.

#### 1. INSERT가 불가능한 경우를 조사한다.

앞의 직원(EMP) 테이블에 대해 다음과 같은 뷰를 생성하였다고 하자.

```
CREATE VIEW PARTIAL_VIEW (NAME, POSITION, WORKPLACE)
AS select NAME, POSITION, WORKPLACE) from EMP;
```

즉, 생성된 뷰에는 PK에 해당하는 UID가 포함되지 않았다. 이 경우 다음과 같은 레코드를 추가해 보자.

```
INSERT INTO PARTIAL_VIEW VALUES ('도길동', '대리', '부산');
```

이와 같은 쿼리는 실패한다. 그 이유는 INSERT 대상이 PARTIAL\_VIEW이지만 이는 논리 이름이고 실제 대상은 직원(EMP) 테이블이기 때문이다. 즉, EMP 테이블에 이와 같은 VALUES를 가지는 INSERT 문은 PK가 누락되었기 때문에 불가능하다.

INSERT가 성공하기 위해서는 반드시 PK가 포함되어야 하며, Not Null이나 제약 조건이 정의된 컬럼에도 반드시 대응되는 입력값이 있어야 한다.

## 2. UPDATE가 불가능한 경우를 조사한다.

INSERT가 불가능한 경우와 유사한 개념으로 무엇을 어떻게 개선해야 할지 모르는 커뮤니티나라도 존재하면 UPDATE도 불가능하다. 보다 구체적인 불가능한 상황은 다음과 같다.

<표 2-8> 뷰 업데이트가 불가능한 상황

UPDATE가 불가능한 상황	내용
뷰를 구성하는 테이블의 기본키를 포함하지 않은 경우	PK, FK 등이 포함되지 않은 뷰
뷰의 컬럼이 산술식, 집계 함수, 상수로부터 유도 된 경우	sum, count, avg 등 포함된 뷰
뷰 정의에서 Group By 절이 포함된 경우	물리적으로 존재하는 레코드를 특정할 수 없음.
다수 테이블로 뷰가 정의된 경우	대부분 불가능하지만 뷰 생성 조건이 엄격하다면 UPDATE가 가능할 수 있음.

뷰에 대한 UPDATE가 가능한 경우는 개선할 레코드를 뷰의 근원이 되는 테이블에서 식별 가능한 경우이다. 그 밖의 경우 테이블의 어떤 레코드가 업데이트 대상인지를 찾을 수 없는 경우 업데이트는 불가능하다. DELETE 명령도 UPDATE 경우와 같다.

앞서 생성한 PARTIAL\_VIEW에 대해 다음과 같은 UPDATE 문을 수행해 보자.

```
UPDATE PARTIAL_VIEW SET
    NAME = '도길동' ,
    POSITION= '대리' ,
    WORKPLACE = '부산' ;
```

위와 같은 경우 UPDATE가 실패할 것이다. 하지만 다음과 같은 CHECK\_WORKPLACE\_VIEW에 대한 UPDATE는 성공할 것이다.

```
UPDATE CHECK_WORKPLACE_VIEW SET  
    NAME = '도길동' ,  
    WORKPLACE = '부산' ;  
WHERE UID = 1090
```

### 3. 뷰 내용 변경 가능 및 불가능 조건을 확인한다.

어느 경우에 뷰의 내용 변경이 가능한 것인지, 어느 경우에 뷰의 내용 변경이 절대적으로 불가능한 것인지에 대해 알아보도록 한다.

<표 2-9> 뷰 변경이 가능 · 불가능한 경우

경우	변경 여부
뷰가 하나의 테이블에서 정의된 경우	가능
뷰 생성에 사용된 테이블의 PK를 포함하는 경우	가능
뷰 정의에서 집계 함수로 정의된 컬럼이 있는 경우	불가능
뷰 정의에서 DISTINCT가 포함된 경우	불가능
뷰 정의에서 GROUP BY 또는 HAVING이 포함된 경우	불가능
뷰 정의에서 서브쿼리가 포함된 경우	불가능
뷰 정의에 상수, 문자열 등이 포함된 경우	불가능

뷰를 구성하는 모든 테이블의 PK를 포함하고, 산술 연산이나 DISTINCT, GROUP BY, HAVING 및 서브쿼리를 포함하지 않는 뷰에 대해서는 내용 변경이나 삭제가 가능하다.

### ④ 뷰를 삭제한다.

#### 1. 기존에 생성한 뷰 삭제하기

앞서 생성한 뷰를 삭제해 보자.

```
DROP VIEW CHECK_LOGIN_VIEW;  
DROP VIEW CHECK_WORKPLACE_VIEW;  
DROP VIEW CHECK_WORKDAY_VIEW;
```

삭제 이후에는 해당 뷰에 대한 select와 같은 데이터 조회가 불가능하다.

### 수행 tip

- 앞서 “[② 뷰 생성](#)” 과정에서 사용한 다음과 같은 쿼리문(SELECT UID FROM CHECK\_LOGIN\_VIEW WHERE UID=' user' , PASSWORD=' pw' )의 사용은 문제가 많다.
- SW 보안 취약점 점검 관점에서 가장 위험한 사용 방법이다. 하지만 본 학습에서는 보안보다는 사용의 개념을 학습하기 위해 사용된 쿼리문으로 단순 이해의 용도로 사용되었다.

## 2-3. 다중 테이블 검색

### 학습 목표

- 조인, 서브쿼리, 집합 연산자를 사용하여 두 개 이상의 테이블로부터 데이터를 조회하는 DML(Data Manipulation Language) 명령문을 작성할 수 있다.

### 필요 지식 /

#### ① 다중 테이블 검색 방법

관계형 데이터베이스는 데이터의 중복을 최소화하기 위해 데이터를 분해하여 저장하고 통합하여 사용한다. 데이터를 분해하는 방법으로 정규화 기법이 사용되며, 통합하는 기법으로 다중 테이블에 대한 검색이 사용된다. 다중 테이블을 이용하는 보다 세부적인 기법은 다음과 같다.

<표 2-10> 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 연산	테이블을 집합 개념으로 조작하는 기법

#### ② 조인(JOIN)

##### 1. 조인 개념

조인은 결합을 의미하며, 관계형 데이터베이스에서의 조인은 교집합 결과를 가지는 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 컬럼을 조합하는 수단이다. 보통 PK와 FK 값을 결합하여 사용하는 것이 일반적이다. 보다 염밀하게 말하자면 PK, FK와 관계없이 논리적인 값들의 연관을 사용한다. 세 개 이상의 테이블에 대한 결합은 두 개의 테이블을 우선 결합하고 그 결과와 나머지 한 개의 테이블을 다시 결합한다.

##### 2. 조인 유형

조인은 관계형 데이터베이스의 가장 큰 장점이면서 대표적인 핵심 기능이라고 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 2-11> 조인 유형(대분류)

조인 분류	내용
논리적 조인	사용자의 SQL문에 표현되는 테이블 결합 방식 데이터베이스의 옵티マイ저에 의해 내부적으로 발생하는 테이블 결합 방식
물리적 조인	- 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 2-12> 조인 유형(세분류)

조인 유형	내용
내부 조인(INNER JOIN)	두 테이블에 공통으로 존재하는 컬럼을 이용하는 방식(공통 컬럼 기반)
동등 조인(EQUI JOIN)	공통 존재 컬럼의 값이 같은 경우를 추출
자연 조인(NATURAL JOIN)	두 테이블의 모든 컬럼을 비교하여 같은 컬럼명을 가진 모든 컬럼 값이 같은 경우를 추출
교차 조인(CROSS JOIN)	조인 조건이 없는 모든 데이터의 조합을 추출
외부 조인(OUTER JOIN)	특정 테이블의 모든 데이터를 기준으로 다른 테이블의 정보를 추출(다른 테이블에 값이 없어도 출력됨.)
왼쪽 외부 조인(LEFT OUTER JOIN)	왼쪽 테이블의 모든 데이터와 오른쪽 테이블의 동일 데이터를 추출
오른쪽 외부 조인(RIGHT OUTER JOIN)	오른쪽 테이블의 모든 데이터와 왼쪽 테이블의 동일 데이터를 추출
완전 외부 조인(FULL OUTER JOIN)	양쪽의 모든 데이터를 추출

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 컬럼 이름이 같은 값을 기준으로 하면 자연 조인이고, 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 테이블의 데이터를 함께 출력하고, 참조 테이블의 조인 데이터가 없는 경우에도 기준 테이블의 모든 데이터를 표시하고 싶은 경우에 사용한다. 보통 기준 테이블

의 모든 값에 대해 참조 테이블의 데이터가 반드시 존재한다는 보장이 없는 경우 외부 조인을 사용한다.

또 내부 조인에서 조인의 대상이 되는 컬럼을 명시적으로 선언하기 위하여 USING 조건절이나 ON 조건절이 사용된다.

### ③ 서브쿼리(Sub Query)

#### 1. 서브쿼리 개념

서브쿼리는 다음 그림과 같이 SQL문 안에 포함된 또 다른 SQL문을 의미한다. 서브쿼리의 용도는 알려지지 않은 기준을 위한 검색을 위해 사용한다.



[그림 2-16] 서브쿼리 개념도

메인쿼리와 서브쿼리 관계는 주종 관계로서, 서브쿼리에 사용되는 컬럼 정보는 메인쿼리의 컬럼 정보를 사용할 수 있으나 역으로는 성립하지 않는다.

#### 2. 서브쿼리 유형

서브쿼리는 동작하는 방식이나 반환되는 데이터의 형태에 따라 분류할 수 있다. 동작하는 방식에 따른 서브쿼리의 종류는 다음과 같다.

<표 2-13> 서브쿼리 유형 1

서브쿼리 종류	설명
비연관(Un-Correlated) 서브쿼리	서브쿼리가 메인쿼리의 컬럼을 가지고 있지 않은 형태(메인쿼리에 서브쿼리에서 실행된 결괏값의 제공 용도)
연관(Correlated) 서브쿼리	서브쿼리가 메인쿼리의 컬럼을 가지고 있는 형태(메인쿼리가 먼저 수행되어 얻은 데이터를 서브쿼리의 조건에 맞는지 확인하고자 할 경우에 사용)

반환되는 데이터 형태에 따른 서브쿼리의 종류는 다음과 같다.

<표 2-14> 서브쿼리 유형 2

서브쿼리 종류	설명
Single Row(단일 행) 서브쿼리	- 서브쿼리의 결과가 항상 1건 이하인 서브쿼리 - 단일 행 비교 연산자(=, <, <=, >, >=, <>)가 사용됨.
Multiple Row(다중 행) 서브쿼리	- 서브쿼리 실행 결과가 여러 건인 서브쿼리 - 다중 행 비교 연산자(IN, ALL, ANY, SOME, EXISTS)가 사용됨.
Multiple Column(다중 컬럼) 서브쿼리	- 서브쿼리 결과가 여러 컬럼으로 반환되는 서브쿼리 - 메인쿼리의 조건절에 여러 컬럼을 동시에 비교할 때, 서브쿼리와 메인쿼리에서 비교하는 컬럼 개수와 위치가 동일해야 함.

#### ④ 집합 연산

##### 1. 집합 연산 개념

테이블을 집합 개념으로 보고, 두 테이블 연산에 집합 연산자를 사용하는 방식이다. 집합 연산자는 여러 질의 결과를 연결하여 하나로 결합하는 방식을 사용한다. 즉, 집합 연산자는 2개 이상의 질의 결과를 하나의 결과로 만들어 준다. 일반적으로 집합 연산자를 사용하는 상황은 서로 다른 테이블에서 유사한 형태의 결과를 반환하는 것을 하나의 결과로 합치고자 할 때와 동일 테이블에서 서로 다른 질의를 수행하여 결과를 합치고자 할 때 사용할 수 있다.

##### 2. 집합 연산 유형

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

<표 2-15> 집합 연산자 유형

집합 연산자	설명
UNION	여러 SQL문의 결과에 대한 합집합(중복 행 제거함.)
UNION ALL	여러 SQL문의 결과에 대한 합집합(중복 행 제거하지 않음.)
INTERSECTION	여러 SQL문의 결과에 대한 교집합(중복 행 제거함.)
EXCEPT (MINUS)	앞의 SQL문의 결과와 뒤의 SQL문의 결과 사이의 차집합(중복 행 제거, 일부 제품의 경우 MINUS 사용)

## 수행 내용 / 다중 테이블 검색하기

### 재료 · 자료

- E-R 다이어그램
- 테이블 정의서
- SQL 언어 문법 학습 자료

### 기기(장비 · 공구)

- DBMS(DataBase Management System) 프로그램

### 안전 · 유의 사항

- 다양한 상황에 대한 많은 연습이 필요하다.
- 모든 DBMS 제품에 공통으로 사용 가능하도록 표준 SQL을 사용한다.

### 수행 순서

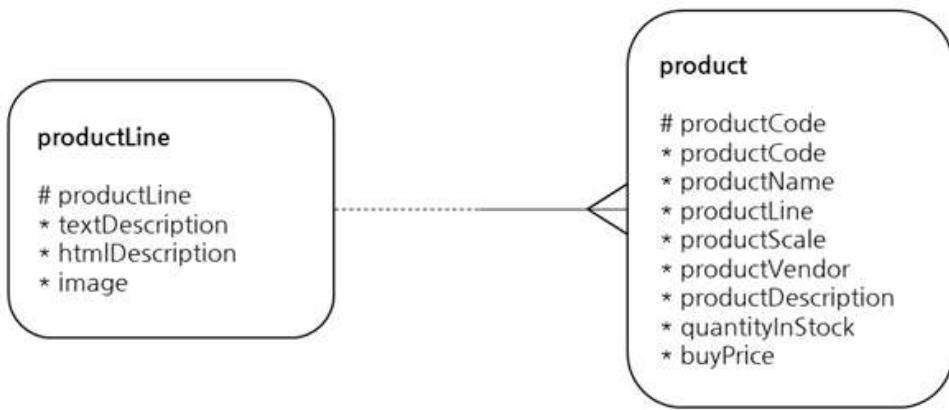


[그림 2-17] 다중 테이블 검색 수행내용

#### ① 다중 테이블에 대해 검색을 수행한다.

##### 1. 다중 테이블에 대해 조사한다.

제품은 제품 유형으로 분류할 수 있다. 이를 위한 테이블은 아래 그림과 같은 개념 모델링에 근거하여 제품 목록을 의미하는 product 테이블과 제품 유형을 의미하는 productLine 테이블로 정의할 수 있다.



[그림 2-18] 다중 테이블 예시

여기서 두 테이블의 관계는 각 테이블의 속성 `productLine`이라는 컬럼을 통해 유지된다. 이 관계를 통해 모든 제품의 상세한 유형 정보를 알 수 있다.

## 2. 다중 테이블에 대해 검색을 수행한다.

앞서 SQL은 구조적 언어라고 하였다. 다중 테이블 검색 과정에서 ‘구조적’에 대비되는 개념과의 비교를 통해 이의 의미를 알아보자.

### (1) 절차적 해결 방법

만일 ‘모든 제품의 상세한 유형 정보’ 조회를 구조적이 아닌 일반적인 프로그래밍 언어를 통해 구현한다면 다음과 같이 표현된다.

```

for (모든 제품에 대해, 하나의 제품선택) {
    for (모든 유형에 대해, 하나의 유형선택) {
        if (선택 제품.productLine = 선택 유형.productLine) {
            // 상세유형정보 = (제품정보 + 유형정보)
        }
    }
}
    
```

### (2) 구조적 해결 방법

동일한 문제를 해결하기 위해 SQL을 사용한 구조적 접근 방법은 다음과 같다.

제품의 상세유형정보 표시하라.

단, 선택 제품.productLine = 선택 유형.productLine 만족하는 조건으로

제품의 코드와 이름, 설명을 보기 위한 실질적인 SQL문은 다음과 같다.

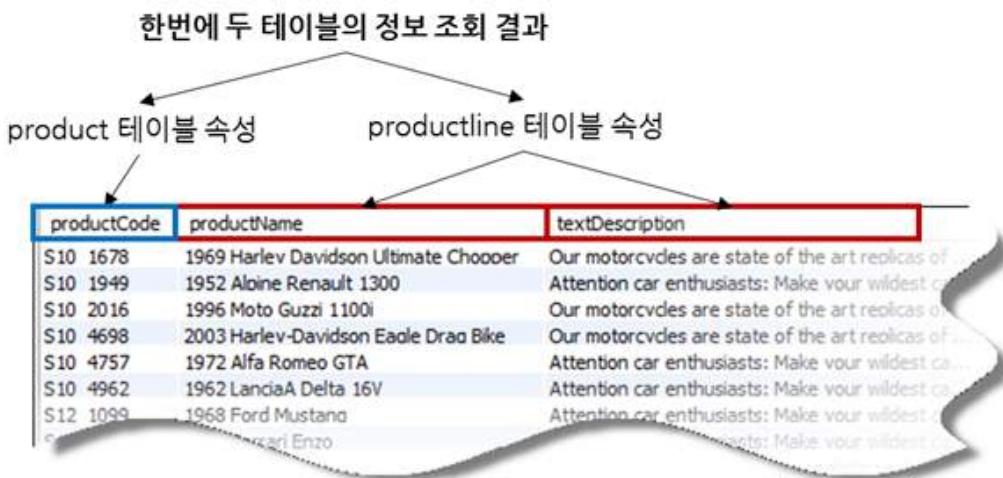
```

SELECT productCode, productName, textDescription
FROM   products t1,
       productlines t2
WHERE t1.productLine = t2.productLine;

```

### (3) 결과 분석하기

실행 결과는 다음 그림과 같으며, 결과적으로 두 테이블의 정보를 한 번에 볼 수 있다.



[그림 2-19] 다중 테이블 조회 결과 분석

### ② 조인으로 다중 테이블을 검색한다.

앞서 제품과 제품 유형 두 테이블의 productLine 컬럼을 이용하여 다중 테이블을 검색하는 방법에 대해 알아보았다. 동일한 결과를 조인을 통해서도 얻을 수 있다.

#### 1. 내부 조인이 무엇인지 확인한다.

내부 조인의 명시적 선언은 테이블 사이에 INNER JOIN이라는 명령어를 삽입함으로써 이루어진다. 동일한 결과를 얻을 수 있는 내부 조인 SQL문은 다음과 같다.

```

SELECT productCode, productName, textDescription
FROM   products t1
INNER JOIN
       productlines t2
ON t1.productLine = t2.productLine;

```

여기서 유의 깊게 보아야 할 부분은 FROM 이후의 부분이다. 여기서는 ON을 이용하여 조인 조건을 명시하였다.

## 2. 내부 조인에 USING을 이용한다.

동일한 결과를 내는 데 ON 대신에 USING을 이용한 SQL문은 다음과 같다.

```
SELECT productCode, productName, textDescription  
FROM products t1  
INNER JOIN  
    productlines t2  
USING (productLine);
```

USING을 이용하면 ON을 사용할 때보다 표현을 단순화할 수 있다. 이는 USING에 사용되는 컬럼의 값이 같을 때를 비교한다는 약속이 전제된 것이기 때문이다.

## 3. 자연 조인(NATURAL JOIN)을 수행한다.

동일한 결과를 얻을 수 있는 자연 조인 SQL문은 다음과 같다.

```
SELECT productCode, productName, textDescription  
FROM products t1  
NATURAL JOIN  
    productlines t2;
```

NATURAL 조인을 사용하기 위해서는 또 다른 암묵적인 약속이 필요하다. 조인 대상 테이블에 동일 컬럼 이름이 존재하며, 이 컬럼의 값이 같은 데이터를 구한다는 약속이 전제된 것이다. 이때 두 테이블에 동일 컬럼 이름이 복수개가 존재하면 이를 모두에 대한 동등 조건이 검색된다.

## 4. 교차 조인(CROSS JOIN)을 수행한다.

교차 조인은 곱하기 개념이다. 두 테이블 또는 집합 사이의 가능한 조합 모든 경우를 결과로 반환한다. 제품과 제품 유형 두 테이블에 대해 교차 조인을 하는 SQL문은 다음과 같다.

```
SELECT productCode, productName, textDescription  
FROM products t1  
CROSS JOIN  
    productlines t2;
```

두 테이블 사이의 모든 가능한 조합을 계산한다. 그런데 다음과 같이 INNER JOIN 문을 사용해도 동일한 결과를 얻을 수 있다.

```
SELECT productCode, productName, textDescription  
FROM products t1  
INNER JOIN  
    productlines t2;
```

즉, 조건이 없는 INNER JOIN은 가능한 모든 조합을 그 결과로 돌려준다.

### 5. 외부 조인(OUTER JOIN)을 수행한다.

외부 조인은 기준이 되는 테이블을 중심으로 데이터를 구성한다. 다음 SQL문을 살펴보자.

```
SELECT productCode, productName, textDescription  
FROM products t1  
LEFT OUTER JOIN  
    productlines t2  
ON t1.productLine = t2.productLine;
```

결과는 INNER JOIN과 같을 수 있다. 모든 제품(product)의 제품 유형 정보(productLine)가 제품 유형 테이블(productLine)에 속해 있다면 위의 결과는 INNER JOIN과 같으며, 제품 유형 테이블(productLine)에 속하지 않는 유형을 가진 제품이 있다면 그 결과는 다르다.

### ③ 서브쿼리로 다중 테이블을 검색한다.

#### 1. 비연관 서브쿼리를 수행한다.

다음과 같은 두 개의 SQL문을 살펴보자.

```
SELECT productCode, productName  
FROM products  
WHERE productLine NOT IN (  
    SELECT distinct productLine FROM productlines  
);  
  
SELECT productLine, textDescription  
FROM productlines  
WHERE productLine NOT IN (  
    SELECT productLine FROM products  
);
```

NOT IN 구절 안에 ( )로 둘러싸인 독립된 SQL문을 비연관 서브쿼리라고 한다. 이는 메인 쿼리와는 별도로 독립적 실행이 가능하기 때문에 비연관 서브쿼리라고 한다. 참고로 위의 두 가지 쿼리문 결과가 의미하는 것은 집합 연산의 일종인 차집합이다.

## 2. 연관 서브쿼리를 수행한다.

서브쿼리 단독으로는 실행이 불가능한 특징을 가지는 연관 쿼리의 사용 예는 다음과 같다.

```
SELECT productname, buyprice
FROM   products p1
WHERE  buyprice > (SELECT
                      AVG(buyprice)
                  FROM
                      products
                  WHERE
                      productline = p1.productline);
```

## 3. 서브쿼리와 조인의 차이를 살펴본다.

조인에 참여하는 모든 테이블이 대등한 관계이기 때문에 조인에 참여하는 모든 테이블의 컬럼을 위치와 무관하게 자유롭게 사용할 수 있다. 그러나 서브쿼리는 주-종 또는 부모-자식이라는 개념에서 제약이 있다. 자식에 해당하는 서브쿼리는 메인쿼리의 컬럼을 모두 사용할 수 있지만, 부모에 해당하는 메인쿼리는 서브쿼리의 컬럼을 사용할 수 없다. 따라서 질의 결과에 서브쿼리 컬럼을 표시해야 한다면 조인 방식으로 변환하거나 함수, 스칼라 서브쿼리(Scalar Subquery) 등을 사용해야 한다.

## ④ 집합 연산으로 다중 테이블을 검색한다.

### 1. 집합 연산의 제약 조건에 대해 알아본다.

집합 연산의 일종인 UNION의 사용 형식은 다음과 같다.

```
SELECT column1, column2
UNION [DISTINCT | ALL]
SELECT column1, column2
```

두 집합 사이에서 집합 연산자가 작동하도록 SELECT 문 사이에 연산자를 배치한다. 이때 UNION과 같은 집합 연산에서 다음과 같은 제약 조건이 존재한다.

- UNION으로 연결된 SELECT 문에 나타나는 컬럼의 숫자는 같아야 한다.
- UNION으로 대응되는 각 컬럼의 데이터 타입이 같아야 한다.

## 2. 집합 연산을 수행한다.

다음과 같은 UNION 연산자를 가진 SQL문을 실행해 보자.

```
SELECT productcode , productname  
FROM products  
UNION  
SELECT productLine ,textDescription  
FROM productlines;
```

UNION으로 루은 두 개의 집합, 테이블에서 선택된 컬럼의 의미가 달라도 제약 조건을 만족하므로 이 명령은 다음과 같이 적절히 수행된다.

앞 테이블의 컬럼 이름이 사용됨

productcode	productname	productLine	textDescription
S10 1678	1969 Harley Davidson Ultimate Choooser		
S10 1949	1952 Alpine Renault 1300		
S10 2016	1996 Moto Guzzi 1100i		
S10 4698	2003 Harley-Davidson Eagle Drag Bike		
S10 4757	1972 Alfa Romeo GTA		
S10 4962	1962 Lancia A Delta 16V		
S12 1099	1968 Ford Mustang		
S12 1108	2001 Ferrari Enzo		
Classic Cars	Attention car enthusiasts: Make your ...		
Motorcycles	Our motorcycles are state of the art re...		
Planes	Unique, diecast airplane and helicopter...		
Ships	The perfect holiday or anniversary gift ...		
Trains	Model trains are a rewarding hobby for...		

서로 다른 내용 (다른 컬럼 값)이 동시에 표현됨

[그림 2-20] UNION 연산에서 컬럼명 결정 모습

서로 이질적인 성격의 테이블을 강제로 UNION하였기에 동일 컬럼의 도메인이 달라지는 문제가 있다. 또 위의 그림과 같이 앞 테이블의 컬럼 이름이 결과에 사용된다. 이름을 동일하게 하기 위해 다음과 같은 SQL문을 실행해 보자.

```
SELECT productcode id, productname name  
FROM products  
UNION  
SELECT productLine id ,textDescription name  
FROM productlines;
```

SELECT 문에 별명을 사용하여 이전과는 다른 컬럼 이름을 확인할 수 있다.

별명으로 지정된 컬럼 이름 사용



id	name
S10 1678	1969 Harley Davidson Ultimate Chopper
S10 1949	1952 Alpine Renault 1300
S10 2016	1996 Moto Guzzi 1100i
S10 4698	2003 Harley-Davidson Eagle Drag Bike
S10 4757	1972 Alfa Romeo GTA
S10 4962	1962 Lancia Delta 16V
S12 1099	1968 Ford Mustang
S12 1108	2001 Ferrari Enzo

[그림 2-21] 별명을 이용한 컬럼명 지정

수행 tip

- SQL 관련 수행은 데이터베이스 및 SQL 언어의 종류에 따라 조금씩 달라질 수 있다. 사용자가 보유한 데이터베이스 및 SQL 언어를 고려하여 수행해야 한다.
- SQL문이 데이터베이스 내부적으로 절차적으로 바뀔 수 있지만, 우리는 구조적 언어 SQL을 사용하며 이를 위해 구조적으로 사고해야 한다.

## 학습 2 교수 · 학습 방법

### 교수 방법

- 교수자는 인덱스 개념에 대해 설명한다.
- 교수자는 인덱스 생성과 삭제 방법을 제시한다.
- 교수자는 인덱스 설계 과정에 대해 설명한다.
- 교수자는 뷰 개념에 대해 설명한다.
- 교수자는 뷰 생성과 삭제 방법을 제시한다.
- 교수자는 다중 테이블을 검색하는 일반적인 방법에 대해 설명한다.
- 교수자는 다중 테이블을 검색하는 조인 방법을 제시한다.
- 교수자는 다중 테이블을 검색하는 서브쿼리 방법을 제시한다.

### 학습 방법

- 학습자는 인덱스 개념을 파악한다.
- 학습자는 인덱스 생성과 삭제 방법을 연습한다.
- 학습자는 인덱스 설계 과정에 대해 파악한다.
- 학습자는 뷰 개념에 대해 파악한다.
- 학습자는 뷰 생성과 삭제 방법을 연습한다.
- 학습자는 다중 테이블에 대한 검색을 실습한다.
- 학습자는 다중 테이블을 검색하는 조인에 대해 실습한다.
- 학습자는 다중 테이블을 검색하는 서브쿼리를 실습한다.

## 학습 2 평 가

### 평가 준거

- 평가자는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습 내용	학습 목표	성취수준		
		상	중	하
인덱스 활용	- 테이블 조회 시간을 단축하기 위해 사용하는 인덱스의 개념을 이해하고, 인덱스를 생성하는 DDL(Data Definition Language) 명령문을 작성할 수 있다.			
뷰 활용	- 먼저 생성된 테이블들을 이용하여 새로운 테이블과 뷰를 생성하는 DDL(Data Definition Language) 명령문을 작성할 수 있다.			
다중 테이블 검색	- 조인, 서브쿼리, 집합 연산자를 사용하여 두 개 이상의 테이블로부터 데이터를 조회하는 DML(Data Manipulation Language) 명령문을 작성할 수 있다.			

### 평가 방법

- 서술형 시험

학습 내용	평가 항목	성취수준		
		상	중	하
인덱스 활용	- 검색 속도를 빠르게 하는 인덱스 원리			
	- 인덱스 설계 과정 및 단계별 유의 사항			
	- 인덱스 생성 방법			
뷰 활용	- 뷰가 필요한 상황			
	- 뷰 생성하기			
	- 뷰를 이용한 데이터 삽입 및 삭제			

다중 테이블 검색	- 다중 테이블에 대한 동일 조회 결과를 만들어 내는 다양한 조인 설계 및 최적의 조인 방법 평가			
	- 연관 서브쿼리와 비연관 서브쿼리 구문 작성하기			
	- 집합 연산의 장단점			

• 사례 연구

학습 내용	평가 항목	성취수준		
		상	중	하
인덱스 활용	- 인덱스 설계를 위한 접근 경로 수집			
	- 수집된 접근 경로에 대한 로그 분석			
	- 복수 컬럼 순서 변경으로 인덱스 성능 차이 구분			
뷰 활용	- 접근 통제가 필요한 보안 필드 구분			
	- 보안이 고려된 뷰, 편의성을 위한 뷰 설계			
	- 뷰를 이용한 데이터 삽입 및 변경 가능하도록 설계			
다중 테이블 검색	- 조인을 표현하는 방법			
	- 서브쿼리 결과 예측			
	- DBMS 제품별 집합 연산 지원 상황 조사			

## 피드백

### 1. 서술형 시험

- 고급 SQL 명령문의 사용 능력을 평가한 후 미흡한 부분에 대해 설명해 준다. 이때 미흡한 부분은 이해 부족에서 발생하는 것이 대부분이기에, 고급 SQL 작성 주제 영역별 SQL 사용 방법에 대한 지도 이외에, 개념적 지식에 대한 설명을 제공한다.
- 본 학습 모듈에서는 추가적인 배경지식을 설명하는 의도로 SQL 사용방법 이외에 SQL이 무엇인지 생각해 가는 과정을 제시하였다. 이러한 의도를 참고하여 배경 지식에 대한 설명 혹은 학습이 이루어지도록 한다.

### 2. 사례 연구

- 다양한 문제 상황에 대한 연습이 가능하도록 상황을 제시해 준다.
- 관련된 연관 주제에 대해 호기심을 가질 수 있도록 연관 관계를 제시해 준다.

# 참고자료

---



- 교육부(2016). 『데이터베이스 구현(2001020405\_16v3)』 . 세종: 한국직업능력개발원.
- 교육부(2016). 『SQL 활용(2001020410\_14v2)』 . 세종: 한국직업능력개발원.
- 데이터전문가 지식포털. DA가이드/SQL가이드(<http://www.dbguide.net>)에서 2017. 7. 23. 검색.
- 인덱스 설계. DA 가이드(<http://www.dbguide.net>)에서 2017. 06. 24. 검색.
- 한국정보통신기술협회. 정보통신용어사전(<http://terms.tta.or.kr>)에서 2017. 9. 7. 검색.
- Git hub. A sample MySQL database([https://github.com/datacharmer/test\\_db](https://github.com/datacharmer/test_db))에서 2017. 7. 23. 검색.
- My SQL. My SQL Document(<https://dev.mysql.com>)에서 2017. 7. 23. 검색.
- My SQL. Employees Sample Database(<https://dev.mysql.com/doc/employee/en/employees-installation.html>)에서 2017. 7. 23. 검색.
- My SQL Tutorial. My SQL Data Manipulation and Definition(<http://www.mysqltutorial.org>)에서 2017. 7. 23. 검색.
- TTA. <http://terms.tta.or.kr>에서 2017. 8. 7. 검색.
- W3 Resource. SQL, Oracle and MySQL Exercises(<http://www.w3resource.com>)에서 2017. 7. 23. 검색.
- W3 Schools. Learn SQL(<http://www.w3resource.com>)에서 2017. 7. 23. 검색.



## NCS학습모듈 개발이력

발행일	2015년 12월 31일		
세분류명	DB엔지니어링(20010204)		
개발기관	한국소프트웨어기술진흥협회, 한국직업능력개발원		
집필진	편홍열(에이비엔아이)* 강성구(명지대학교) 김승현(경희대학교) 박미화(투이컨설팅) 박준자(한국오라클) 임영섭(비투엔컨설팅) 장온순(한국IT컨설팅) 장인혁(청운)	검토진	김보운(이화여자대학교) 여권동(NDS시스템) 정금묵(베이스존) 주선태(T3Q) 진권기(이비스톰)
			* 표시는 대표집필자임
발행일	2017년 12월 31일		
학습모듈명	SQL 활용(LM2001020413_16v3)		
개발기관	(사)한국정보통신기술사협회, 한국직업능력개발원		
집필진	김광국((주)코스콤)* 김동욱(한국기술교육대학교) 김준범(SK주식회사) 송정현((주)씨에이에스)	검토진	이주연((주)씨에이에스) 장경애((사)한국정보통신기술사협회)
			* 표시는 대표집필자임
발행일	2018년 12월 31일		
학습모듈명	SQL 활용(LM2001020413_16v3)		
개발기관	한국직업능력개발원		

## SQL 활용(LM2001020413\_16v3)

저작권자	교육부
연구기관	한국직업능력개발원
발행일	2018.12.31

※ 이 학습모듈은 자격기본법 시행령(제8조 국가직무능력표준의 활용)에 의거하여 개발하였으며, NCS통합포털사이트(<http://www.ncs.go.kr>)에서 다운로드 할 수 있습니다.



[www.ncs.go.kr](http://www.ncs.go.kr)