

MySQL



Chapter 01

SQL 비긴즈



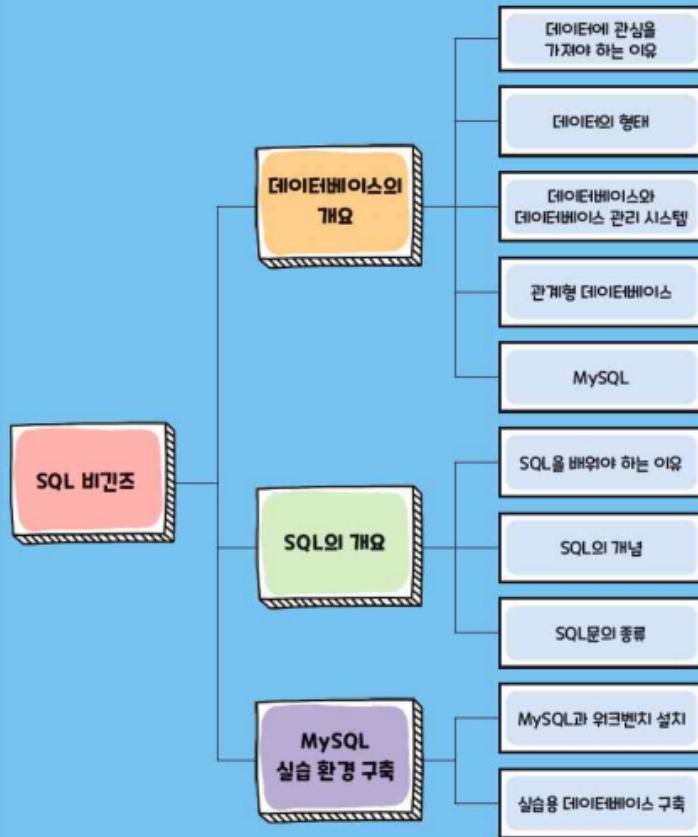
목차

1. 데이터베이스의 개요
2. SQL의 개요
3. MySQL 실습 환경 구축

학습목표

- 데이터의 중요성을 이해할 수 있습니다.
- 데이터베이스와 SQL의 개념을 이해할 수 있습니다.
- SQL 실습을 위한 환경을 구축합니다

Preview



Section 01

데이터베이스의 개요

1. 데이터에 관심을 가져야 하는 이유

■ 데이터

- 측정, 관찰, 연구, 분석 등을 통해 수집된 정보
- 활용분야
 - ✓ 금융 부문에서는 위험과 사기 감지
 - ✓ 의료 부문에서는 유전학 및 질병을 분석
 - ✓ 기업에서는 고객의 행동과 선호도 이해
 - ✓ 새로운 시장 기회 발견
 - ✓ 비즈니스 전략 수립

1. 데이터에 관심을 가져야 하는 이유

■ 폭발적으로 증가하는 데이터

- 정보 기술의 발전과 디지털화로 인해 막대한 양의 데이터가 생성됨
- 전 세계 디지털 정보량 60제타바이트(ZB) 수준(2020년 기준)

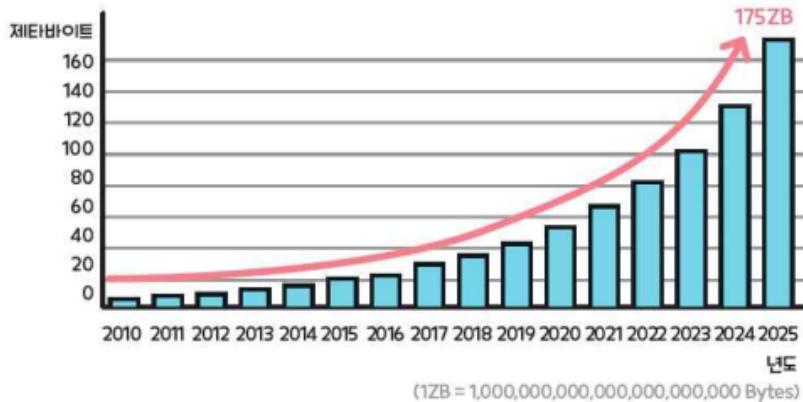


그림 1-1 데이터 예상 증가율

1. 데이터에 관심을 가져야 하는 이유

■ 4차 산업혁명 기술과 데이터

- 데이터가 핵심 자원으로 적용되는 분야
 - ✓ 인공지능 및 머신러닝
 - ✓ 빅데이터 분석
 - ✓ 사물인터넷
 - ✓ 블록체인
 - ✓ 로봇공학
 - ✓ 가상 현실 및 증강 현실



그림 1-2 데이터가 활용되는 분야

2. 데이터의 형태

■ 정형 데이터(Structured Data)

- 미리 정해진 구조에 따라 저장된 데이터
- 손쉽게 데이터 검색 및 삽입, 수정, 삭제 연산을 수행할 수 있음
- 엑셀의 스프레드시트나 관계형 데이터베이스의 테이블에 저장된 데이터 등

A	B	C	D
1 사용일자	노선명	역명	승차증승객수
2 20220101 3호선	수서		7370
3 20220101 3호선	학여울		461
4 20220101 3호선	대청		3224
5 20220101 3호선	일원		3321
6 20220101 경원선	창동		1
7 20220101 1호선	신설동		4939
8 20220101 1호선	동대문		6486
9 20220101 1호선	종로5가		7975
10 20220101 1호선	서울역		18398
11 20220101 1호선	동묘앞		7240
12 20220101 1호선	시청		5604
13 20220101 1호선	종각		9977
14 20220101 1호선	종로3가		11017
15 20220101 2호선	구로디지털단지		18363
16 20220101 2호선	신대방		11652
17 20220101 2호선	신림		29564
18 20220101 2호선	봉천		9207
19 20220101 2호선	서울대입구(관악)		20142

그림 1-3 정형 데이터의 예(엑셀 파일)

2. 데이터의 형태

■ 반정형 데이터(Semi-Structured Data)

- 정형 데이터와는 달리 구조화되어 있지 않아서 연산이 불가능함
- 데이터 내부의 데이터 구조에 대한 메타 정보를 함께 제공하는 파일 형식의 데이터
- HTML, XML, JSON 파일이나 웹 로그, 센서 데이터 등

```
{  
    "crewId": "110135",  
    "qualificationCode": "737TCE",  
    "renewalType": "RECURRENT",  
    "fromDateTime": "2099-04-30 00:00:00",  
    "toDateTime": "2099-05-30 00:00:00",  
    "qualificationStatus": "PLANNED",  
    "checkDateTime": "2022-04-07 00:00:00",  
    "comments": null,  
    "internalCrewID": "108207",  
    "odsChangeTimestamp": "2023-08-14 13:37:57.668287"  
}
```

그림 1-4 반정형 데이터의 예(JSON 파일)

2. 데이터의 형태

■ 비정형 데이터(Unstructured Data)

- 정해진 구조나 규칙이 없고, 연산에 사용할 수 없는 형태의 데이터
- 소설 데이터 및 워드나 PDF 문서, 이미지, 영상 등



(a) 페이스북

(b) 인스타그램

(c) 트위터

그림 1-5 비정형 데이터의 예(SNS)

3. 데이터베이스와 데이터베이스 관리 시스템

■ 데이터베이스(Database)

- 여러 사람이 공유하고 운영할 목적으로 관리되는 통합적인 정보의 집합
- 컴퓨터 시스템에 전자적으로 저장되는 구조화된 데이터 모음
- 데이터베이스는 데이터베이스 관리 시스템(DBMS)에 의해 관리됨

■ DBMS

- 최종 사용자 또는 응용 프로그램이 데이터베이스에 접근할 수 있는 인터페이스 역할
- 저장된 데이터를 보다 체계적으로 관리하고 이용할 수 있게 해주는 소프트웨어

3. 데이터베이스와 데이터베이스 관리 시스템

■ 사용 방식에 따른 데이터베이스 유형

표 1-1 데이터베이스 유형

데이터베이스 유형	설명
관계형 데이터베이스	열과 행이 있는 테이블 집합으로 구성되며, 정형 데이터에 액세스하는 가장 효율적이고 유연한 방법을 제공한다.
객체 지향 데이터베이스	객체 지향 패러다임을 사용하는 객체 지향 프로그래밍으로부터 영향을 받아 생성된 데이터베이스로 정보를 객체의 형태로 표현한다.
분산 데이터베이스	물리적으로 떨어진 데이터베이스를 네트워크로 연결하여 단일 데이터베이스 이미지를 보여준다. 분산된 작업 처리를 수행하는 데이터베이스로 사용자는 시스템이 분산되어 있는지 인식하지 못한 채 사용할 수 있다.
데이터 웨어하우스	조직 내 서로 다른 다양한 소스들의 정보를 집계하고 저장하는 시스템으로 최종 사용자가 전사적 정보를 분석할 수 있게 하여 의사 결정을 도모할 수 있도록 설계된다.
NoSQL 데이터베이스	관계형 데이터베이스 이외의 형식으로 데이터를 저장하는 데이터베이스로 대량의 분산된 데이터를 저장하고 조회하는데 특화되어 있다.
클라우드 데이터베이스	클라우드 플랫폼을 통해 구축·접근할 수 있는 데이터베이스로 기존 데이터베이스와 동일한 기능에 클라우드 컴퓨팅의 유연성이 추가되었다. 사용자가 클라우드 인프라에서 DBMS를 선택하여 데이터베이스를 구현할 수 있다.
자율 운영 데이터베이스	클라우드를 기반으로 머신러닝을 사용하여 데이터베이스 튜닝, 보안, 백업, 업데이트 및 기타 데이터베이스 관리자가 전통적으로 수행해 온 일상적인 관리 작업을 자동화한 데이터베이스이다.

4. 관계형 데이터베이스

■ 관계형 데이터베이스(Relational Database)

- 관계형 데이터 모델을 기반으로 한 데이터베이스

■ 테이블(Table)

- 릴레이션(Relation)이라고도 함
- 1개 이상의 열과 0개 이상의 행으로 이루어진 데이터 집합

■ 열(Column)

- 속성(Attribute), 필드(Field)라고도 함
- 의미가 더 이상 분리되지 않는 최소의 데이터 단위

4. 관계형 데이터베이스

■ 행(Row)

- 투플(Tuple), 레코드(Record)라고도 함
- 관련 있는 열의 묶음
- 한 테이블에 저장된 모든 행은 동일한 수의 열을 가짐

■ 관계(Relationship)

- 두 테이블 간에 서로 연결되는 방식
- 데이터의 종속성을 표현



■ 기본키(Primary key)

- 주된 식별자
- 한 테이블 내에서 행을 구별할 수 있는 열 또는 열의 묶음

4. 관계형 데이터베이스

■ 관계형 데이터베이스의 기본 용어



그림 1-6 관계형 데이터베이스의 기본 용어

5. MySQL

■ MySQL

- 1995년에 발표된 오픈 소스 RDBMS
- 웹 애플리케이션용으로 설계 및 최적화되어 모든 플랫폼에서 실행할 수 있음
- 4대 RDBMS : MySQL, Oracle, MS-SQL, PostgreSQL

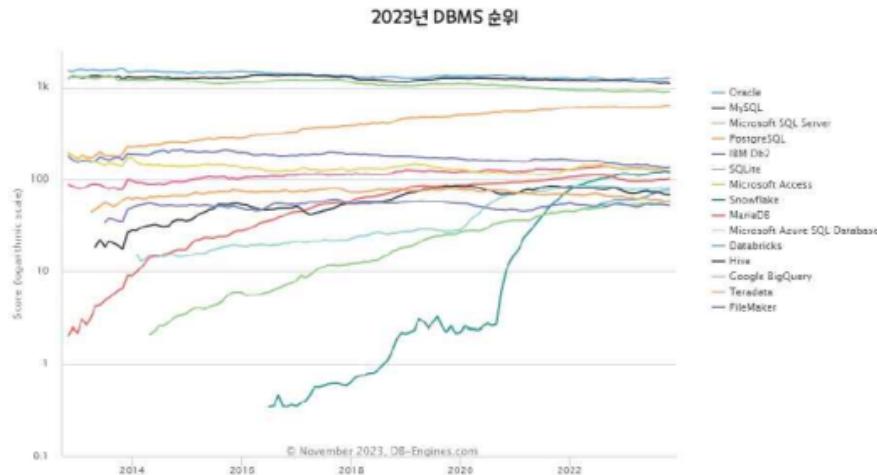


그림 1-7 2023년 기준 DBMS 순위(©https://db-engines.com/en/ranking_trend/relational+dbms)

Section 02

SQL의 개요

1. SQL을 배워야 하는 이유

■ SQL(Structured Query Language)

- IBM이 1970년대에 개발한 SEQUEL을 기반으로 만들어진 언어
- 데이터에 접근하기 위한 가장 보편적인 수단
- SQL이 사용되는 클라우드 기반의 머신러닝 플랫폼
 - ✓ 아마존 레드시프트 ML
 - ✓ 구글 빅쿼리 ML
 - ✓ 스노플레이크
 - ✓ 오라클 클라우드 인프라 데이터 사이언스
 - ✓ 마이크로소프트 SQL 서버 머신러닝 서비스

2. SQL의 개념

■ SQL의 역할

- 관계형 데이터베이스 관리 시스템(RDBMS)의 데이터를 관리하고 다양한 데이터 동작을 수행하는 데 사용되는 표준화된 프로그래밍 언어
- 데이터베이스 및 테이블 생성, 데이터 검색, 추가, 수정, 삭제 및 트랜잭션 처리, 보안 및 권한 제어 등 데이터베이스의 모든 측면을 관리하는 데 사용됨

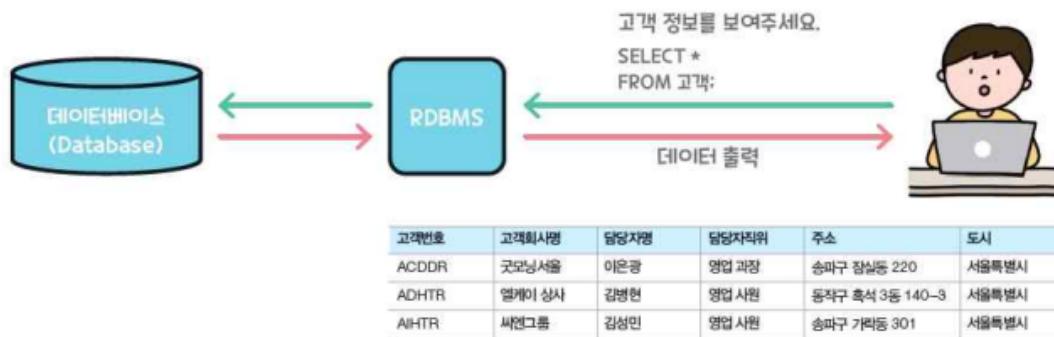


그림 1-8 SQL 역할

■ SQL의 특징

- 배우기 쉽고 사용하기 용이함
- 데이터 연산에 대한 처리 과정이 절차적이지 않으며 집합 단위로 처리됨
- 사용자 편의 중심 언어
- 미국표준협회(ANSI)의 표준에 맞게 SQL을 작성한다면 Oracle, MySQL, MS-SQL, PostgreSQL, MariaDB 등 상용 RDBMS 간에 호환과 전환이 용이함

3. SQL문의 종류

■ 데이터 정의어(DDL)

- 데이터베이스의 구조를 정의할 때 사용하는 언어
- 데이터베이스나 테이블, 인덱스와 같은 데이터베이스 객체를 생성, 변경, 삭제할 때 사용함

표 1~2 DDL문의 종류

종류	설명
CREATE	데이터베이스 객체를 생성한다(① 데이터베이스, 테이블, 뷰, 인덱스 등).
ALTER	생성된 객체의 구조를 변경한다.
DROP	생성된 객체를 삭제한다.
TRUNCATE	테이블에 있는 데이터를 모두 삭제한다.

3. SQL문의 종류

■ 데이터 조작어(DML)

- 데이터를 관리하는 데 사용하는 언어
- 데이터 조작어는 레코드를 추가, 삭제하거나 데이터를 변경할 때 사용함
- 데이터를 검색하는 데 사용하는 SELECT문은 데이터 조작어에 포함하기도 하고 데이터 질의어(DQL)로 분리하기도 함

표 1-3 DML문의 종류

종류	설명
SELECT	테이블에서 조건에 맞는 데이터를 검색한다.
INSERT	테이블에 새로운 레코드를 삽입한다.
UPDATE	테이블에 있는 데이터 값을 수정한다.
DELETE	테이블에 있는 레코드를 삭제한다.

3. SQL문의 종류

■ 데이터 제어어(DCL)

- 데이터에 대한 액세스를 제어하기 위한 언어
- 데이터베이스 객체나 데이터에 대한 권한을 관리하는 데 사용함

■ 트랜잭션 제어어(TCL)

- INSERT, UPDATE, DELETE문에 의해 수행된 변경 사항을 관리하는 데 사용됨

표 1-4 DCL문과 TCL문의 종류

종류		설명
DCL	<u>GRANT</u>	특정 사용자 또는 특정 객체에 대해 생성, 수정 등 특정 작업을 할 수 있도록 권한을 부여한다.
	<u>REVOKE</u>	GRANT문으로 부여한 권한을 철회할 때 사용한다.
TCL	<u>COMMIT</u>	트랜잭션 작업 내용을 실제 데이터베이스에 영구 저장한다.
	<u>ROLLBACK</u>	트랜잭션 처리 과정에서 발생한 변경 사항을 취소한다.

Section 03

MySQL 실습 환경 구축

1. MySQL과 워크벤치의 설치

■ 실습 환경

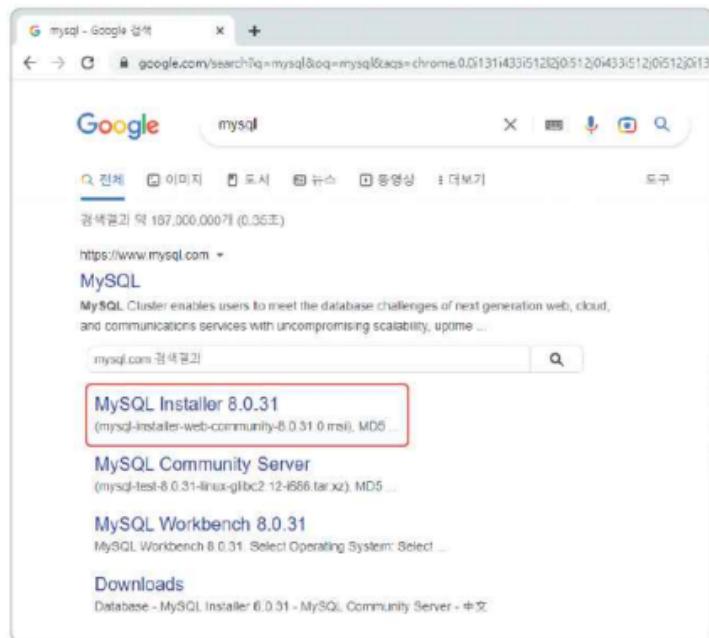
- MySQL 8.0
- 워크벤치(Workbench) 8.0
 - ✓ SQL 개발과 관리, 데이터베이스 설계와 생성 등을 위해 단일 개발 통합 환경을 제공하는 비주얼 데이터베이스 설계 도구

1. MySQL과 워크벤치의 설치

■ MySQL 설치하기

① MySQL 홈페이지의 다운로드 페이지

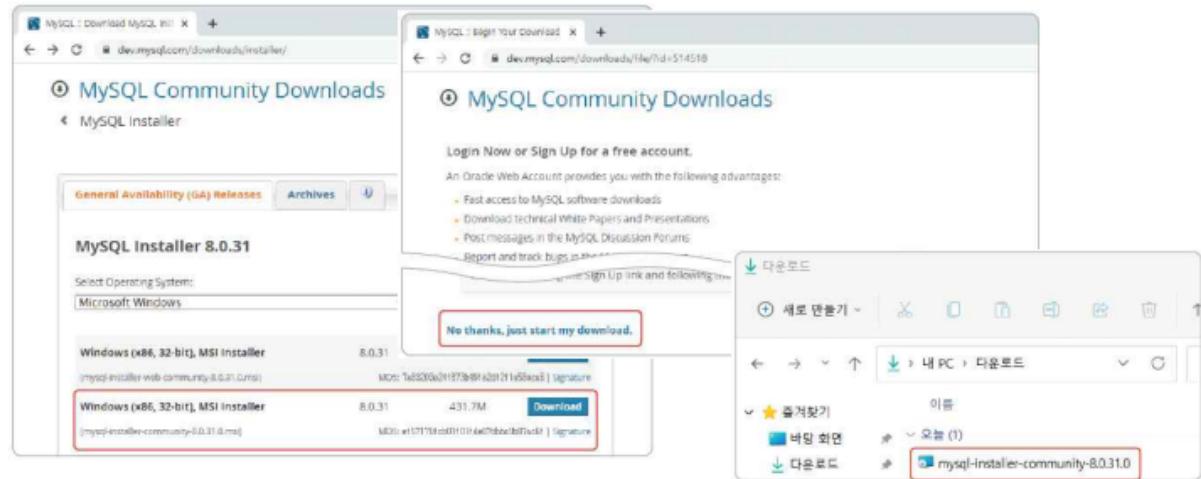
(<https://dev.mysql.com/downloads/installer/>)에 접속하기



1. MySQL과 워크벤치의 설치

■ MySQL 설치하기

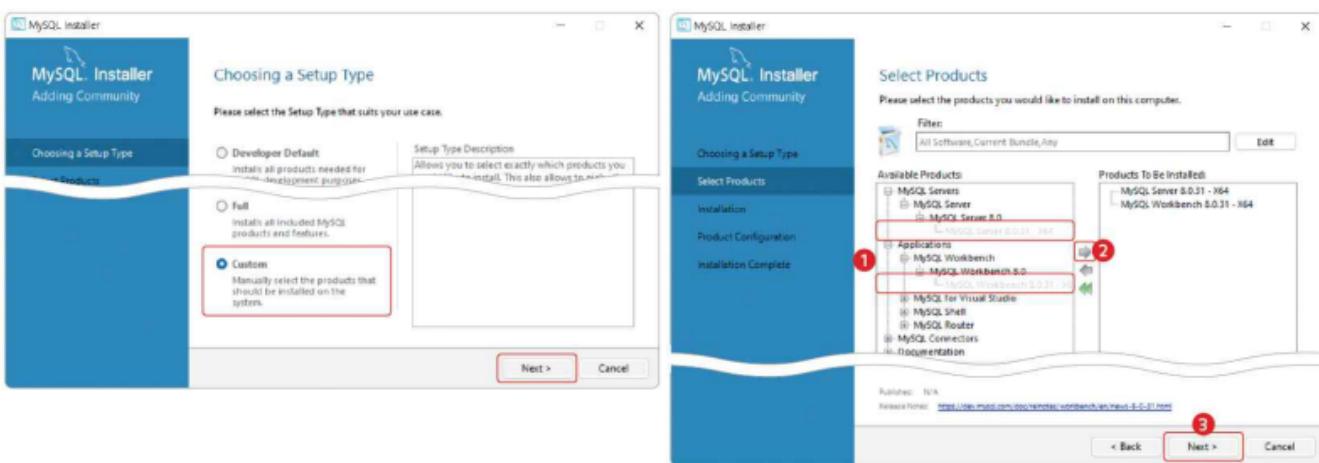
- ② [MySQL Community Downloads] 창에서 'mysql-installer-community-8.0.31.0.msi'의 [Download] 버튼 클릭
- ③ [No thanks, just start my download.] 링크를 클릭하면 로그인 없이 바로 다운로드할 수 있음. [다운로드] 폴더에서 내려받은 파일을 더블클릭하여 실행



1. MySQL과 워크벤치의 설치

■ MySQL 설치하기

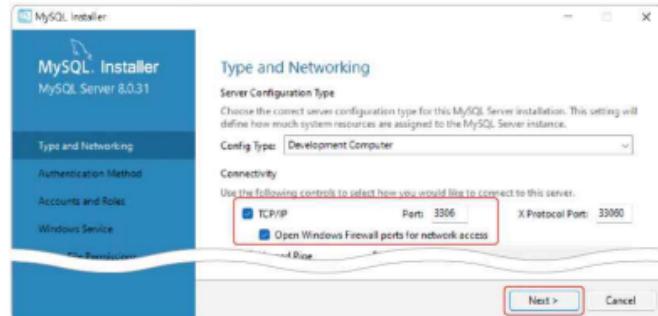
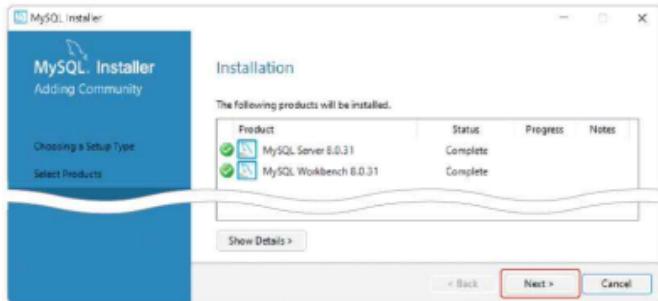
- ④ 설치 타입을 선택하는 [Choosing a Setup Type] 창에서 [Custom] 항목을 클릭하고 <Next> 버튼 클릭
- ⑤ [Select Products] 창의 'Available Products'에서 'MySQL Server 8.0.31-x64'와 'MySQL Workbench 8.0.31-x64'를 'Products To Be Installed'로 옮긴 후 <Next> 버튼 클릭



1. MySQL과 워크벤치의 설치

■ MySQL 설치하기

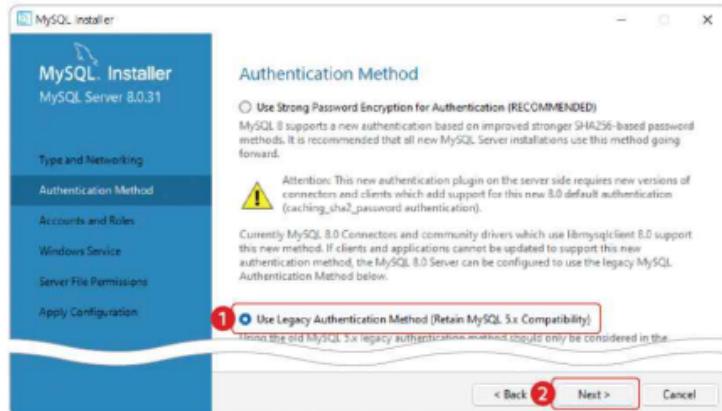
- ⑥ <Execute> 버튼을 클릭하여 설치를 진행하고, 설치가 끝나면 <Next> 버튼 클릭
- ⑦ [Type and Networking] 창에서 'TCP/IP'와 'Open Windows Firewall ports for network access' 항목에 체크되어 있는지 확인한 후, <Next> 버튼 클릭



1. MySQL과 워크벤치의 설치

■ MySQL 설치하기

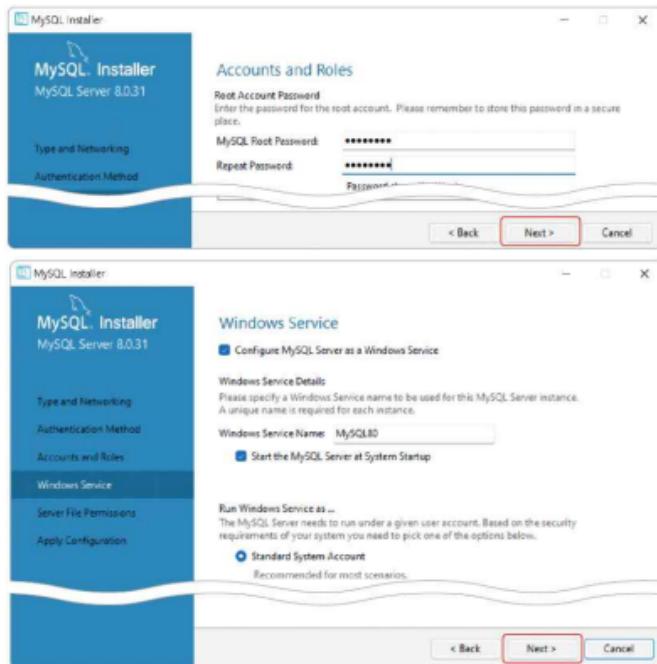
- ⑧ [Authentication Method] 창에서는 기존 프로그램들과의 호환성을 위해서 MySQL 5.x 버전에서 사용하던 방식을 선택. 'Use Legacy Authentication Method(Retain MySQL 5.x Compatibility)'에 체크하고 <Next> 버튼 클릭



1. MySQL과 워크벤치의 설치

■ MySQL 설치하기

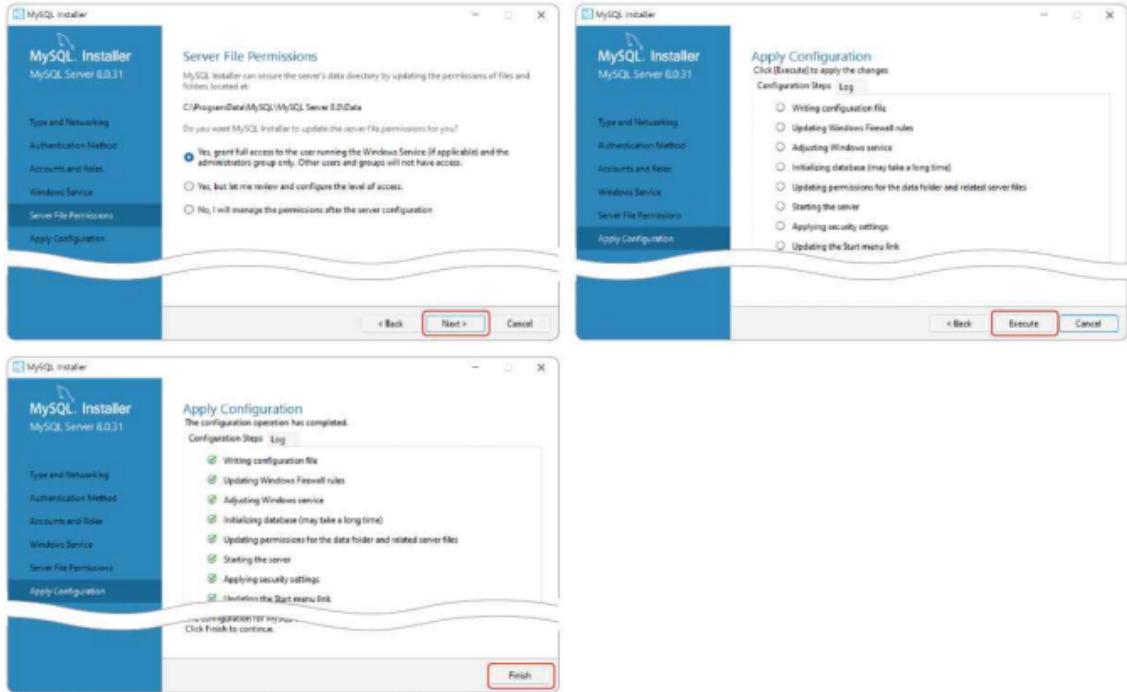
- ⑨ MySQL 관리자의 비밀번호(mysql123)를 설정하고 <Next> 버튼 클릭
- ⑩ [Windows Service] 창에서 <Next> 버튼 클릭



1. MySQL과 워크벤치의 설치

■ MySQL 설치하기

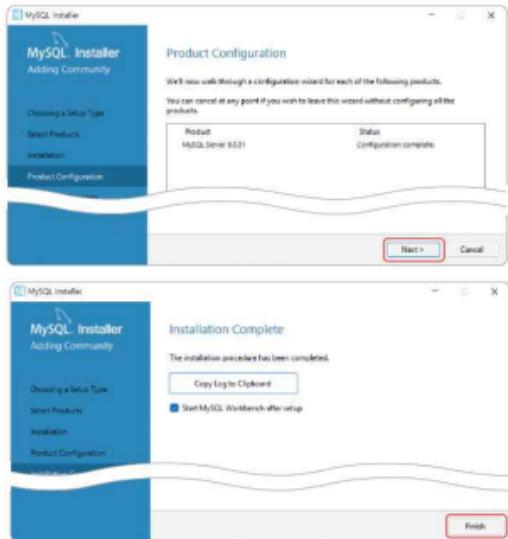
⑪ 이어지는 화면에서 <Next> 버튼과 <Execute> 버튼을 클릭하여 설치 계속 진행



1. MySQL과 워크벤치의 설치

■ MySQL 설치하기

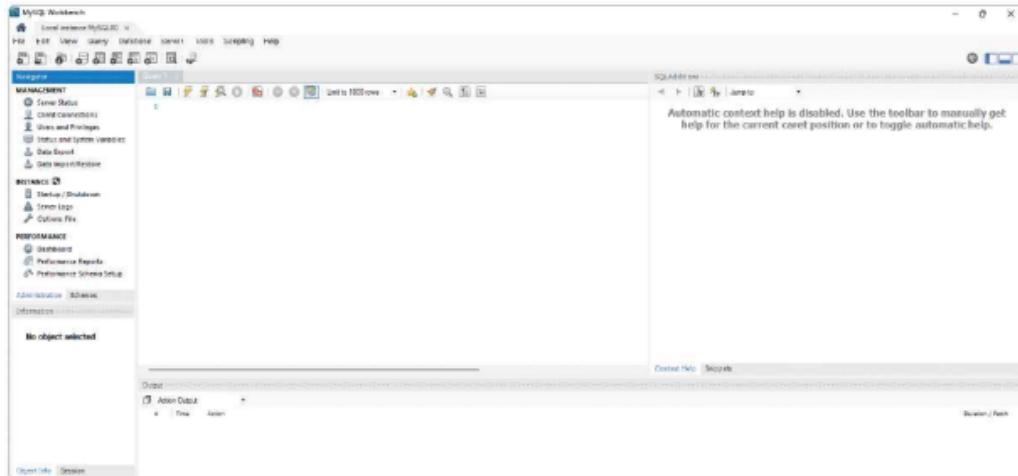
- ⑫ [Installation Complete] 화면에서 <Finish> 버튼을 클릭하면 워크벤치의 시작화면이 나타남
- ⑬ 워크벤치의 시작화면에서 하단의 [Local instance MySQL80]을 클릭
- ⑭ 비밀번호(mysql123) 입력하고 <OK>버튼 클릭



1. MySQL과 워크벤치의 설치

■ MySQL 설치하기

- ⑯ MySQL 워크벤치 화면이 나타나면 설치가 성공적으로 완료됨



2. 실습용 데이터베이스 구축

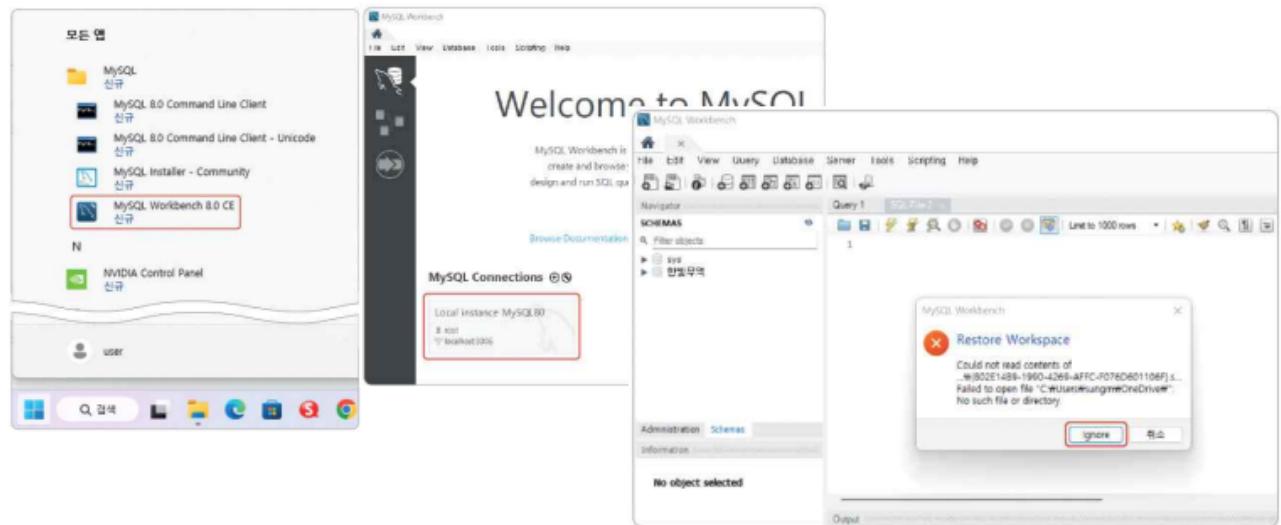
■ 실습 데이터베이스 '한빛무역'

- 한빛무역 데이터베이스
 - ✓ 고객, 사원, 부서, 주문, 주문세부, 제품에 대한 데이터를 관리하게 됨
 - ✓ 고객은 '거래처' 회사를 의미함
- 실습 데이터베이스 생성 순서
 - ① 스크립트 파일을 실행하여 데이터베이스와 테이블을 일괄 생성함
 - ✓ 실습 데이터베이스를 생성하기 위해 한빛아카데미 자료실에서 실습 예제 압축파일을 다운로드하기
 - ✓ [C] 드라이브에 [데이터] 폴더를 새로 생성하여 다운로드한 파일을 옮기고 압축 해제
 - ② 데이터베이스와 테이블을 생성한 후 CSV 파일에 있는 데이터를 각각의 테이블로 가져옴

2. 실습용 데이터베이스 구축

■ 실습 데이터베이스 생성하기

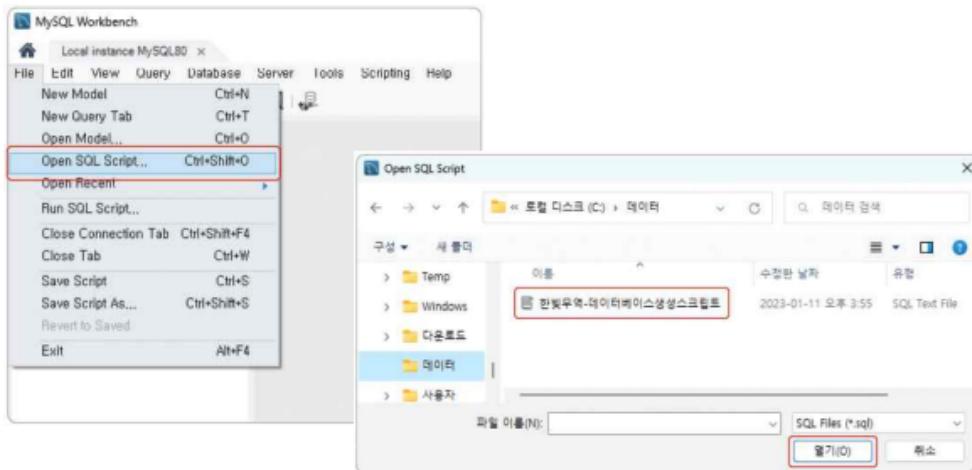
- ① 원도우에서 [MySQL]-[MySQL Workbench 8.0 CE]를 클릭
- ② 시작화면에서 'Local instance MySQL80'을 클릭
 ① 아직 연결된 데이터베이스가 없어서 오류가 발생할 수 있는데, 이런 경우에는
 <Ignore> 버튼을 클릭하여 넘어가기



2. 실습용 데이터베이스 구축

■ 실습 데이터베이스 생성하기

- ③ 워크벤치 시작화면에서 [File]-[Open SQL Script]를 클릭
- ④ [데이터] 폴더에서 데이터베이스 생성 스크립트 파일인 '한빛무역-데이터베이스 생성스크립트.sql'을 선택하고 <열기> 버튼 클릭



2. 실습용 데이터베이스 구축

■ 실습 데이터베이스 생성하기

⑤ 데이터베이스와 테이블 생성을 위한 스크립트가 나타남. 스크립트의 11행을 선택하여 실행하기

- ✓ 11행의 CREATE DATABASE...을 드래그하고 실행버튼을 클릭
- ✓ 실행이 완료되면 하단에 초록색 체크표시가 나타남

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays various management categories like Server Status, Client Connections, and Performance. The central area is a Query Editor titled 'Query 1' containing the following SQL script:

```
1 * # 고려할: 난상작성을 SQL
2 * # 차이: 내용은
3 * # 같은 DATABASE를: 한빛무역
4 * # 한상일: 2023-01-11
5 *
6 *****
7
8
9 DROP DATABASE IF EXISTS 한빛무역;
10
11 * CREATE DATABASE 한빛무역 DEFAULT CHARSET utf8mb4 COLLATE utf8mb4_general_ci;
12
13 * USE 한빛무역;
14
15 * CREATE TABLE 부서(
    ...
)
```

The 11th line, which contains the 'CREATE DATABASE' command, is highlighted with a red box. Below the editor, the Action Output pane shows the execution results:

Time	Action	Message
1 11:23:28	CREATE DATABASE 한빛무역 DEFAULT CHARSET utf8mb4 COLLATE utf8mb4_general_ci;	1 row(s) affected

2. 실습용 데이터베이스 구축

■ 실습 데이터베이스 생성하기

- ⑥ 13행의 USE 한빛무역;을 실행한 후, 15행인 CREATE TABLE 부서부터 스크립트 파일에 있는 나머지 문장 끝까지 선택하고 실행버튼을 클릭
- ⑦ 테이블 목록은 [Schema] 탭(②)에서 확인할 수 있음. 생성된 테이블 목록(③)을 확인하기 위해서는 먼저 새로고침 버튼(①)을 클릭

The screenshot shows the MySQL Workbench interface with the following highlights:

- ①** Refresh button (top right of the query editor).
- ②** Schemas tab in the left navigation bar.
- ③** Tables section under the Schemas tab, showing the newly created tables: 고객, 고객_주문, 부서, 시점, 계정, 주문, 주문세부.
- ④** Table list in the Object Info panel, showing the columns for the '고객' table.
- ⑤** Execute button (top right of the query editor).

The Query 1 editor contains the following SQL script:

```
56     고객번호 INT PRIMARY KEY,
57     고객명 VARCHAR(30),
58     회장직위 VARCHAR(30),
59     담당 INT,
60     직급 INT
61     ] DEFAULT CHARSET=utf8mb4;
62
63
64
65
66 * CREATE TABLE 고객(
67     주문번호 CHAR(5) PRIMARY KEY,
68     고객번호 CHAR(5),
69     사원번호 CHAR(5),
70     주문일 DATE,
71     주문상품 VARCHAR(20),
72     주문수량 INT,
73     주문금액 DECIMAL(10,2),
74     주문날짜 DATETIME);
```

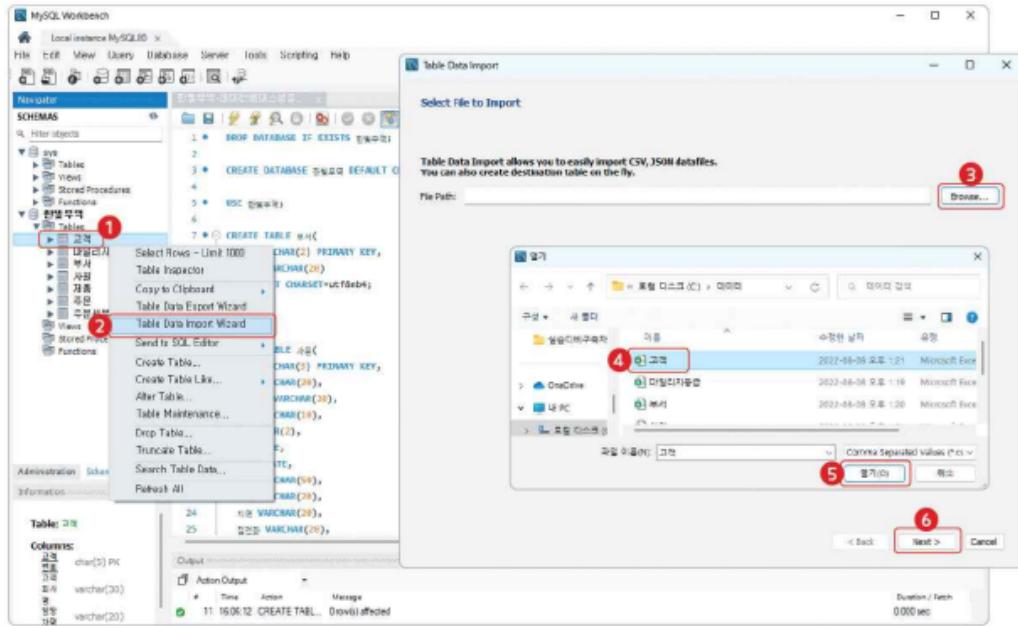
The Output panel shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	11:26:03	CREATE TABLE 고객(고객번호 CHAR(5)PRIMARY KEY, ...) rows affected	0 rows affected	0.015 sec
2	11:26:03	CREATE TABLE 고객_주문(고객번호 INT PRIMARY KEY, 고객명 VARCHAR(30), ...) rows affected	0 rows affected	0.015 sec
3	11:26:03	CREATE TABLE 부서(부서번호 CHAR(5) PRIMARY KEY, ...) rows affected	0 rows affected	0.015 sec
4	11:26:03	CREATE TABLE 시점(시점번호 CHAR(5) PRIMARY KEY, ...) rows affected	0 rows affected	0.015 sec
5	11:26:03	CREATE TABLE 계정(계정번호 CHAR(5) PRIMARY KEY, ...) rows affected	0 rows affected	0.015 sec
6	11:26:03	CREATE TABLE 주문(주문번호 CHAR(5) PRIMARY KEY, 고객번호 INT, ...) rows affected	0 rows affected	0.015 sec
7	11:26:03	CREATE TABLE 주문세부(주문세부번호 CHAR(5) PRIMARY KEY, ...) rows affected	0 rows affected	0.015 sec
8	11:26:03	CREATE TABLE 주문상품(주문상품번호 CHAR(5) PRIMARY KEY, ...) rows affected	0 rows affected	0.015 sec
9	11:26:03	CREATE TABLE 주문금액(주문금액 DECIMAL(10,2) PRIMARY KEY, ...) rows affected	0 rows affected	0.015 sec
10	11:26:03	CREATE TABLE 주문날짜(주문날짜 DATETIME PRIMARY KEY, ...) rows affected	0 rows affected	0.015 sec

2. 실습용 데이터베이스 구축

■ CSV 파일에 들어있는 데이터를 각 테이블에 넣기

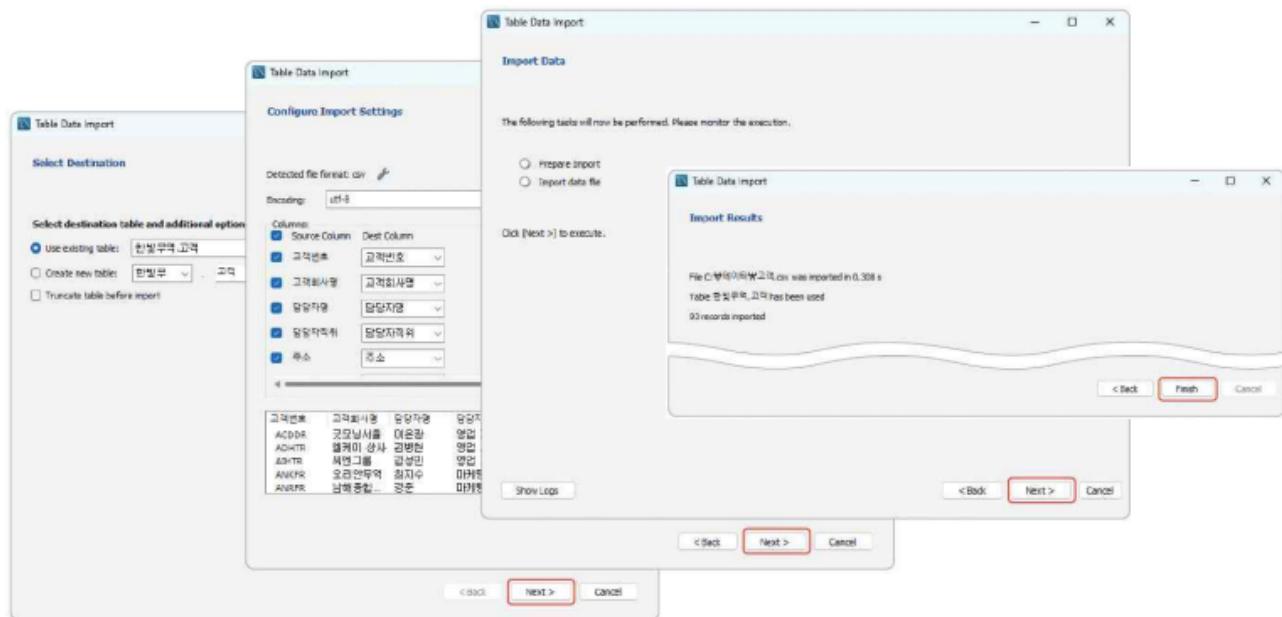
- ⑧ 고객 테이블에서 마우스 오른쪽 클릭 → [Table Data Import Wizard] 선택 → [Select File to Import] 창에서 [Browse] 버튼 클릭 → [데이터] 폴더에 있는 고객.csv 파일 선택 → <열기> 버튼 클릭 → <Next> 버튼 클릭



2. 실습용 데이터베이스 구축

■ CSV 파일에 들어있는 데이터를 각 테이블에 넣기

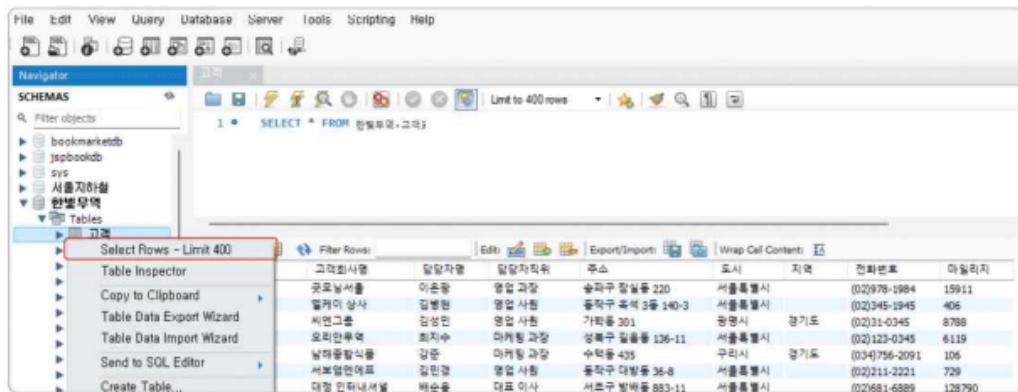
- ⑨ 각 설치 단계마다 기본 설정을 그대로 두고 <Next> 버튼 클릭
- ⑩ 마지막 화면에서 <Finish> 버튼을 클릭하면 데이터 삽입이 완료됨



2. 실습용 데이터베이스 구축

■ 고객 테이블에 삽입된 레코드를 확인하기

- ⑪ 고객 테이블에서 마우스 오른쪽 클릭하여 [Select Rows-Limit 1000]을 선택
 - ⑫ 부서, 사원, 제품, 주문, 주문세부, 마일리지등급 테이블에 대하여 동일하게 작업
을 진행하면 실습을 위한 데이터베이스 구축 작업이 완료됨

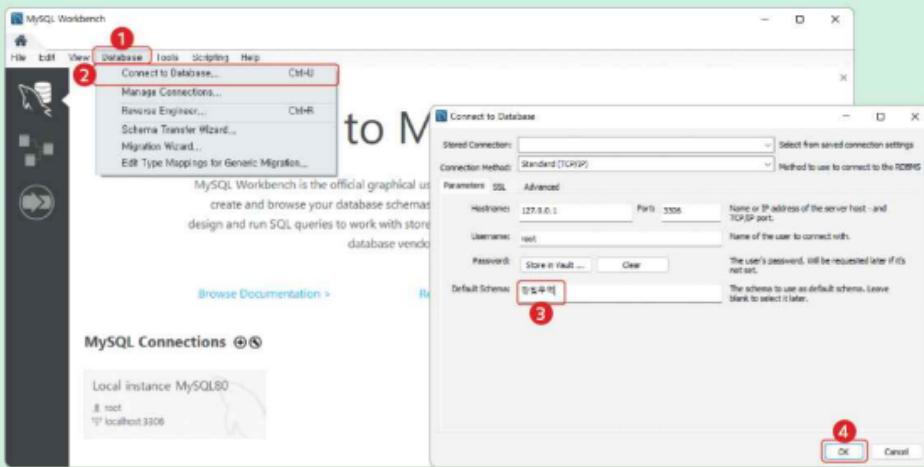


2. 실습용 데이터베이스 구축

하나 더 알기

워크벤치 실행 오류

만일 다시 워크벤치를 실행할 때 연결 오류가 나타난다면 다음과 같이 작업을 진행합니다. [Database]–[Connect to Database]를 선택하고, [Default Schema]에 ‘한빛무역’ 데이터베이스명을 입력한 후, [OK] 버튼을 클릭합니다. 이제 한빛무역 데이터베이스가 디폴트 데이터베이스로 설정되었습니다.

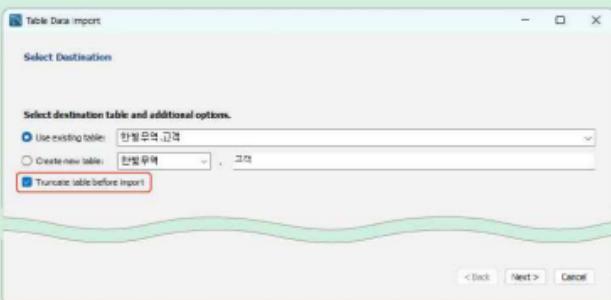


2. 실습용 데이터베이스 구축

하나 더 알기 V

실습 데이터 초기화

만일 데이터를 초기화하고 싶다면 다시 가져올 테이블에서 마우스 오른쪽 버튼을 클릭하여 [Table Data Import Wizard]를 클릭하고 [Select Destination]창에서 'Truncate table before import'에 체크합니다. 이후 데이터 가져오기를 다시 진행합니다.



Chapter 02

SQL의 기본 질의문과 연산자



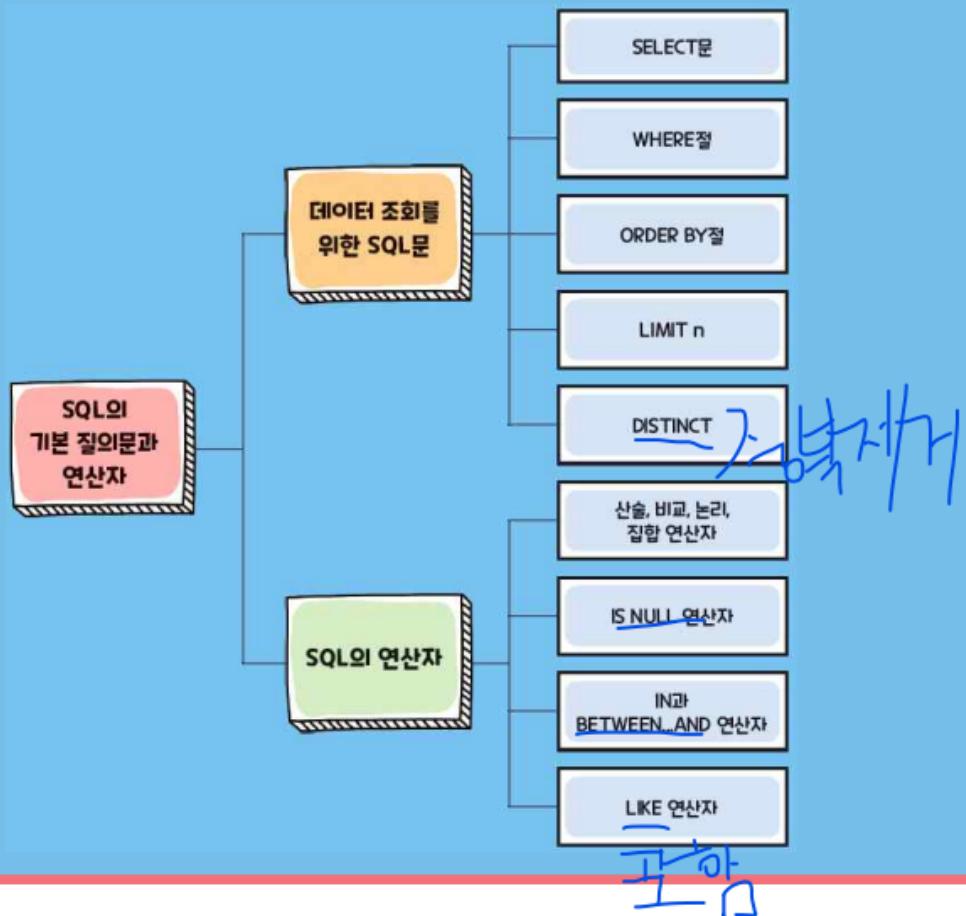
목차

1. 데이터 조회를 위한 SQL문
2. SQL 연산자

학습목표

- 데이터 조회를 위한 SQL 문법을 이해할 수 있습니다.
- SELECT절, WHERE절, ORDER BY절 및 LIMIT n에 대한 문법을 이해할 수 있습니다. X
- SQL문에서 연산자를 활용할 수 있습니다

Preview



Section 01

데이터 조회를 위한 SQL문

1. SELECT문

■ SELECT문

- 형식

```
SELECT    컬럼1 [AS 별명1]           검색할 컬럼명  
          ,컬럼2 [별명2]  
          ,컬럼3  
FROM      테이블명 *             데이터를 검색할 테이블명  
[WHERE     조건]  
[ORDER BY 컬럼]  
[LIMIT    n];
```

- SELECT절에는 컬럼명 뿐만 아니라 수식이나 함수가 들어갈 수도 있음

1. SELECT문

■ SELECT문

- 컬럼에는 별명(Alias)을 붙여줄 수 있음
 - ✓ 별명은 컬럼명을 실제로 변경하는 것이 아니라 현재 SELECT 문장에서만 적용되는 컬럼의 제목

✓ AS 다음에 별명을 입력하거나 AS를 생략하고 바로 별명을 입력해도 됨
만약 별명에 공백이 있는 경우 또는 특수문자가 포함된 경우에는 반드시 쌍따옴표(" ")나 작은따옴표(' ') 안에 별명을 넣어줌

내소문자

구분 X

(DB 안에서만)

그림 2-1 별명을 붙이는 이유

내 이름은
"김수한무 거북이와 두루미
삼천갑자 동방삭" 이에요.



매번 부르기에는
이름이 너무 길어~!



1. SELECT문

■ [예제 2-1] 고객 테이블에 있는 모든 데이터를 조회하시오

```
SELECT *  
FROM 고객;
```

▶ 실행 결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
ACDDR	굿모닝서울	이 음갈	영업 과장	송파구 갈실동 220	서울특별시	(02)978-1984	15911	
ADHTR	엘케이 상사	길 병현	영업 사원	동작구 흑석 3동 140-3	서울특별시	(02)345-1945	406	
AIHTR	시연그룹	김성민	영업 사원	기학동 301	광명시	경기도	(02)31-0345	8788
ANKFR	오리안무역	최지수	마케팅 과장	상복구 길동동 136-11	서울특별시	(02)123-0345	6119	
ANRFR	남하증합식품	강준	마케팅 과장	수락동 435	구리시	경기도	(034)756-2091	106
ANSFR	서보알ян스프	김 민경	영업 사원	동작구 마방동 36-6	서울특별시	(02)211-2221	729	
ATRAN	대정 인디나서울	배 순웅	대표 이사	시즈구 방배동 883-11	서울특별시	(02)681-6889	128790	
AUSBL	계원인터넷교	박지혜	영업 사원	서대문구 남가� 1동 121	서울특별시	(02)811-1234	47865	
BEVBS	케이리탈리스	오선주	영업 사원	구산동 17-111	김해시	경상남도	(052)342-3333	8758
BQOZA	비큐큐루드	활수경	대표 이사	서초구 서초동 120-3	서울특별시	(02)456-9876	100	
CARRI	진가식품	이두원	영업 사원	비전 1동 118-36	평택시	경기도	(033)945-1932	122
CCOPT	진산푸드	길효정	영업 과장	구로구 오류 1동 101	서울특별시	(02)100-1670	1034	
CORIA	한술F&B	이살훈	영업 사원	구로구 가리봉 1동 22-1	서울특별시	(02)123-7981	1331	
CSUR	유로이이지	이소경	영업 과장	서대문구 창전동 167-3	서울특별시	(02)712-8931	6608	
CTEVI	한국맥널트 무역	김유환	영업 사원	중구 단동 79	인천광역시	(032)158-1984	939	
CTUCA	유성물산교역	우지선	영업 사원	강남구 매치동 315-11	서울특별시	(02)943-4486	7232	
DOSGO	프루트크리아	길태욱	영업 과장	중구 광희동 1가 188-3	서울특별시	(02)545-8911	713	
DWOOL	하이畏惧	길예진	영업 사원	도봉구 미아 7동 334-7	서울특별시	(02)774-1290	6935	

조회된 행수 : 1066

1. SELECT문

- [예제 2-2] 고객 테이블에서 고객번호, 담당자명, 고객회사명, 마일리지, 10% 인상된 마일리지를 조회하시오. 이때 마일리지는 '포인트'로, 인상된 마일리지는 '10%인상된 마일리지'로 별명을 붙입니다.

```
SELECT 고객번호  
      ,담당자명  
      ,고객회사명  
      ,마일리지 AS 포인트  
      ,마일리지 * 1.1 AS "10%인상된 마일리지"  
FROM 고객;
```

▶ 실행결과

고객번호	담당자명	고객회사명	포인트	10%인상된 마일리지
ACDDR	이윤광	굿모닝서울	15911	17502.1
ADHTR	김병현	알레이 상사	406	446.6
AIIHR	김성민	서연그룹	8788	9666.8
ANKFR	최지수	오리안우역	6119	6730.9
ANRFR	강준	남해풀빌식물	106	116.6
ANSFR	길민경	서보艰辛에프	729	801.9
ATRAN	배운종	다정 인력나서널	128790	141669.0
AUSBL	박지혜	제원인디네쇼날	47865	52651.5
BVBBS	오선주	케미리탈리스	8758	9633.8
BQQ2A	황수영	비큐큐두드	100	110.0
CARRI	이두원	진가식품	122	134.2
GHQJL	김호정		1034	1137.4

결정명인 마일리지 대신
별명인 '포인트'가 보여짐

2. WHERE절

- [예제 2-3] 고객 테이블에서 마일리지가 100,000점 이상인 고객의 고객번호, 담당자명, 마일리지를 조회하시오.

```
SELECT 고객번호  
      ,담당자명  
      ,마일리지  
FROM 고객  
WHERE 마일리지 >= 100000;
```

▶ 실행 결과

고객번호	담당자명	마일리지
ATRAN	배준동	128790
PRDSU	한현구	115898

3. ORDER BY절

■ ORDER BY

- 레코드를 순서대로 정렬하고자 할 때 사용함
- 컬럼명 이외에도 수식이나 함수 또는 컬럼의 순번이 올 수도 있음
- 오름차순 정렬(ASC)
 - ✓ 작은 값부터 순서대로 보여줌
 - ✓ ASC는 생략 가능함
- 내림차순 정렬(DESC)
 - ✓ 큰 값부터 순서대로 보여줌
- ORDER절에서 컬럼명 대신 별명이나 컬럼의 순서를 넣을 수도 있음

ORDER BY 포인트 DESC; (O)

ORDER BY 4 DESC; (O)

3. ORDER BY절

- [예제 2-4] '서울특별시'에 사는 고객에 대해 고객번호, 담당자명, 도시, 마일리지를 조회하시오. 이때 마일리지가 많은 고객부터 순서대로 보입니다.

```
SELECT 고객번호  
      ,담당자명  
      ,도시  
      ,마일리지 AS 포인트  
  FROM 고객  
 WHERE 도시 = '서울특별시'  
 ORDER BY 마일리지 DESC;
```

▶ 실행 결과

고객번호	담당자명	도시	포인트
ATRAN	배순봉	서울특별시	128790
PRDSU	한현구	서울특별시	116898
HNSLE	조효주	서울특별시	78001
AUSBL	박지혜	서울특별시	47865
LMKWT	조가영	서울특별시	36960
LASLI	지승환	서울특별시	22260
ACDDR	이은광	서울특별시	15911

4. LIMIT n

■ LIMIT n

- 사용하여 반환되는 레코드의 개수를 지정할 수 있음
- 1행부터 시작하여 n개의 레코드를 가져오기 위해서는 문장의 맨 마지막에 LIMIT n을 추가함
- 시작행의 위치를 지정할 수도 있음
- ORDER BY절 뒤에 LIMIT와 가져올 레코드의 수를 넣으면 상위 또는 하위 n개의 레코드를 조회할 수 있음

4. LIMIT n

- [예제 2-5] 고객 테이블에서 1행부터 시작하여 3개의 고객 정보를 조회하시오입니다.

```
SELECT *  
FROM 고객  
LIMIT 3;
```

▶ 실행결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
ACDR	굿모닝서울	이은경	영업 과장	송파구 잠실동 220	서울특별시	(02)978-1984	15911	
ADHTR	얼케이 상사	김병현	영업 사원	동작구 흥선 3동 140-3	서울특별시	(02)345-1945	406	
AHTR	씨엔그룹	김성민	영업 사원	가락동 301	광명시	경기도	(02)31-0345	8788

- [예제 2-6] 마일리지가 많은 고객부터 상위 3명의 고객에 대한 모든 정보를 조회하시오.

```
SELECT *  
FROM 고객  
ORDER BY 마일리지 DESC  
LIMIT 3;
```

▶ 실행결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
ATRAN	대성 인터내셔널	배순율	대표 이사	서초구 방배동 883-11	서울특별시	(02)681-6889	128790	
PRODSU	풀인 텐플라이시스	한정구	회계 과장	용산구 서빙고동 111-16	서울특별시	(02)934-1984	116898	
EBITH	그린로더스	홍소현	마케팅 과장	남동구 간석동 264-11	인천광역시	(032)945-1846	81479	

5. DISTINCT

DISTINCT

- 데이터를 조회할 때 중복된 데이터를 한 번만 보이고자 할 때 사용함
- SELECT절에 DISTINCT를 넣으면 중복된 값은 제거한 결과를 보여줌

The diagram illustrates the use of the DISTINCT keyword. On the left, a tall vertical table labeled '도시' (City) contains eight rows of data, with the values '서울특별시', '서울특별시', '부산광역시', '서울특별시', '부산광역시', '서울특별시', and '서울특별시' repeated. A large red arrow points from this tall table to a shorter vertical table on the right, also labeled '도시'. This second table only contains two distinct rows: '서울특별시' and '부산광역시', demonstrating that the original data has been filtered to remove duplicates.

도시
서울특별시
서울특별시
부산광역시
서울특별시
부산광역시
서울특별시
서울특별시

→

도시
서울특별시
부산광역시

그림 2-2 DISTINCT 사용으로 중복 데이터 제거

5. DISTINCT

- [예제 2-7] 고객 테이블의 도시 컬럼에 들어있는 값 중 중복되는 도시 데이터를 한 번씩만 보이시오.

```
SELECT DISTINCT 도시  
FROM 고객;
```

▶ 실행결과

도시
서울특별시
충청시
구하이
김해시
광택시
인천광역시
대전광역시
상주시

Section 02

SQL 연산자

1. 산술 연산자

■ 연산자

- 변수나 값의 연산을 위해 사용되는 부호
- SQL에서 연산자는 단독으로 사용할 수 없음
- 컬럼이나 상수 등과 함께 사용하여 여러 가지 계산 작업을 수행할 수 있음

■ 산술 연산자

- 더하기(+), 빼기(-), 곱하기(*), 나누기(/), 나머지(%)
- DIV 연산자 : 나누기에서 정수 결과를 얻을 때 사용
- % 또는 MOD 연산자 : 나머지 구하기

1. 산술 연산자

■ [예제 2-8] 두 개의 숫자 23과 5로 산술 연산자 +, -, *, /, %를 사용한 결과를 나타내시오. DIV, MOD 연산자의 사용 결과도 함께 확인해봅니다.

SELECT 23 + 5 AS 더하기 •————— 클럽에 보여질 별명

,23 - 5 AS 빼기

,23 * 5 AS 곱하기

,23 / 5 AS 실수나누기

,23 DIV 5 AS 정수나누기

,23 % 5 AS 나머지1

,23 MOD 5 AS 나머지2;

테이블에서 데이터를 가져오지 않는 경우에는 FROM절을 생략하고 SELECT절만 작성합니다.

▶ 실행 결과

더하기	빼기	곱하기	실수나누기	정수나누기	나머지1	나머지2
28	18	115	4.6000	4	3	3

2. 비교 연산자

■ 비교 연산자

- 두 값을 비교하여 조건에 맞는 결과를 얻고자 할 때 사용하는 연산자
- 1(True)이나 0(False) 값이 반환됨
- 크거나 같다(\geq), 작거나 같다(\leq), 크다($>$), 작다($<$), 같다($=$), 같지 않다(\neq 또는 $<>$)가 있음

2. 비교 연산자

- [예제 2-9] 두 개의 숫자 23과 5로 비교 연산자 $>=$, $<=$, $>$, $<$, $=$, $!=$, $<>$ 를 사용한 결과를 보이시오.

```
SELECT 23 >= 5  
, 23 <= 5  
, 23 > 23  
, 23 < 23  
, 23 = 23  
, 23 != 23  
, 23 <> 23;
```

▶ 실행결과

23 >= 5	23 <= 5	23 > 23	23 < 23	23 = 23	23 != 23	23 <> 23
1	0	0	0	1	0	0

- [예제 2-10] 담당자가 '대표 이사'가 아닌 고객의 모든 정보를 보이시오.

```
SELECT *  
FROM 고객  
WHERE 담당자직위 <> '대표 이사';
```

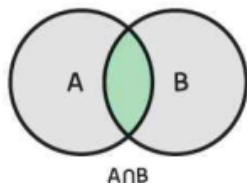
▶ 실행결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
ACDOR	굿모닝서울	이은우	영업 과장	송파구 잠실동 220	서울특별시	(02)978-1984	15911	
ADHTR	헬케이 상사	김병현	영업 사원	용적구 흑석 3동 140-3	서울특별시	(02)345-1945	406	
ADPTR	씨엔그룹	김성민	영업 사원	가평군 301	광명시	경기도	(02)31-0345	8788
ANKFR	오피안무역	최지수	마케팅 과장	성북구 길을동 136-11	서울특별시	(02)123-0345	6119	
ANRFR	남화동한식당	김준	마케팅 과장	수락동 435	구리시	경기도	(034)756-2091	106

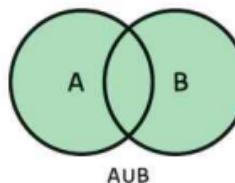
3. 논리 연산자

■ 논리 연산자

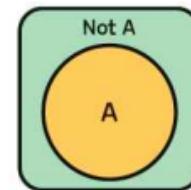
- 표현식이 참인지 거짓인지를 확인하기 위해 사용하는 연산자
- AND
 - ✓ 조건이 두 개 이상 있는 경우에 사용
 - ✓ 모든 조건이 참(True)인 레코드를 반환함
- OR
 - ✓ 여러 조건 중에서 하나 이상의 조건이 참인 레코드를 반환함
- NOT
 - ✓ 부정을 의미하는 연산자
- AND, OR, NOT 연산자를 교집합, 합집합, 여집합으로 대응시킬 수 있음



(a) AND 연산



(b) OR 연산



(c) NOT 연산

그림 2-3 논리 연산자의 집합 표현

3. 논리 연산자

- [예제 2-11] 도시가 '부산광역시'이면서 마일리지가 1,000점보다 작은 고객의 모든 정보를 보이시오.

```
SELECT *
```

```
FROM 고객
```

```
WHERE 도시 = '부산광역시'  
AND 마일리지 < 1000;
```

→ 레코드에 해당

▶ 실행 결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
LLWE	태신무역	박소영	영업 과장	부산진구 양정 1동 462-...	부산광역시		(051)555-5111	806

두 조건을 만족하는 레코드

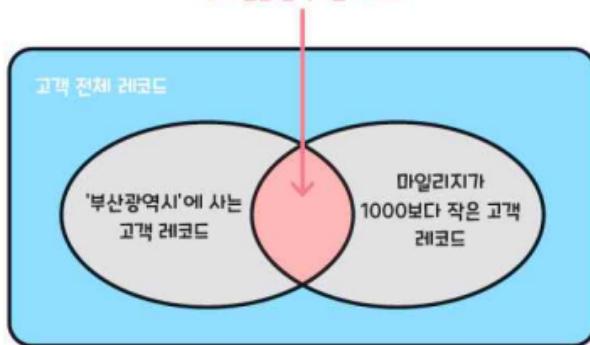


그림 2-4 [예제 2-11]의 두 가지 조건

4. 집합 연산자

■ 집합 연산자

- MySQL은 집합 연산자 중에서 합집합에 대한 연산자만 제공함
- 집합 연산자를 사용한 쿼리문에는 두 개 이상의 SELECT문이 들어있어야 함
- 두 개의 SELECT문 사이에 집합 연산자를 넣어줌
- UNION
 - ✓ 합집합 연산자
 - ✓ UNION ALL : 중복된 레코드까지 모두 다 나타내는 합집합 연산자
- UNION 연산자를 사용할 때는 주의할 점
 - ① 각 SELECT문에서 컬럼의 개수는 동일해야 한다.
 - ② 각 SELECT문에서 같은 위치에 존재하는 컬럼의 데이터 타입은 동일하거나, 상호 호환이 가능해야 함

4. 집합 연산자

■ 집합 연산자

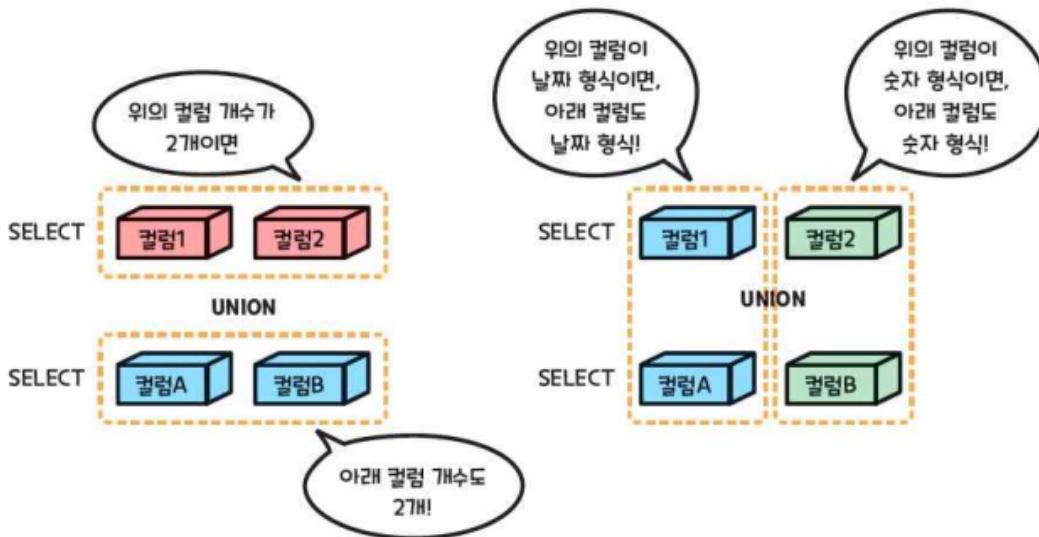


그림 2-5 UNION 연산자 사용 시 주의사항

4. 집합 연산자

- [예제 2-12] '부산광역시'에 살거나 마일리지가 1,000점보다 작은 고객에 대하여 고객번호, 담당자명, 마일리지, 도시를 보이시오. 이때 결과는 고객번호 순으로 정렬합니다.

```
SELECT 고객번호
      ,담당자명
      ,마일리지
      ,도시
  FROM 고객
 WHERE 도시 = '부산광역시'
UNION
SELECT 고객번호
      ,담당자명
      ,마일리지
      ,도시
  FROM 고객
 WHERE 마일리지 < 1000
ORDER BY 1; •———— ORDER BY 절은 제일 마지막에 넣어줌
```

▶ 실행결과

고객번호	담당자명	마일리지	도시
TTTCA	주협율	892	의정부시
TTIBO	안수인	7329	부산광역시
TTRAN	김희원	448	서울특별시
URLGO	김하경	865	성남시
VESSE	전혜연	414	서울특별시
ZYKLA	장재상	261	원주시

A

B

4. 집합 연산자

■ [예제 2-12] '부산광역시'에 살거나 마일리지가 1,000점보다 작은 고객에 대하여 고객번호, 담당자명, 마일리지, 도시를 보이시오. 이때 결과는 고객번호 순으로 정렬합니다.

- OR 연산자를 사용한 해결법

```
SELECT 고객번호
      ,담당자명
      ,마일리지
      ,도시
FROM 고객
WHERE 도시 = '부산광역시'
OR 마일리지 < 1000
ORDER BY 1;
```

5. IS NULL 연산자

■ NULL(Unknown Value)

- NULL은 알 수 없는 값을 의미함
- 0이나 빈 문자열(Empty String)과는 다른 의미함
- 컬럼에 값이 들어있지 않은 데이터를 검색하기 위해서는 **IS NULL**을 사용해야 함
 - ✓ = NULL이 아님

5. IS NULL 연산자

■ [예제 2-13] 지역에 값이 들어있지 않는 고객의 정보를 보이시오

```
SELECT *
FROM 고객
WHERE 지역 IS NULL;
```

▶ 실행 결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
ACDDR	굿모닝서울	이몽근	영업 과장	송파구 잠실동 220	서울특별시		(02)978-1984	15911

null

- 실행 결과 0개의 레코드가 나옴
- 빈 문자열을 검색했을 때는 레코드가 검색됨

```
SELECT *
FROM 고객
WHERE 지역 = '';
```

▶ 실행 결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
ADHTR	엘제이 상사	김병현	영업 사원	동작구 흑석 3동 140-3	서울특별시		(02)345-1945	406
ANIKFR	オリ인후부	최지수	마케팅 과장	성북구 칠동동 136-11	서울특별시		(02)123-0345	6119
ANISFR	서로�펠엔에프	길민강	영업 사원	용산구 대방동 36-8	서울특별시		(02)211-2221	729
ATRAN	대장 안티내셔널	배준용	대표 이사	서초구 방배동 883-11	서울특별시		(02)681-6889	128790
AUSBL	제원인터넷쇼날	박지혜	영업 사원	서대문구 남가좌 1동 121	서울특별시		(02)811-1234	47865
BAKJGA	비쥬루루드	최수경	대표 이사	서초구 방배동 883-3	서울특별시		(02)681-9876	100

빈 문자열

5. IS NULL 연산자

■ [예제 2-13] 지역에 값이 들어있지 않는 고객의 정보를 보이시오

- IS NULL을 사용했을 때 검색되지 않은 이유
 - ✓ NULL과 빈 문자열은 의미가 서로 다르기 때문
 - ✓ CSV 파일에서 테이블로 가져오기를 했을 때 값이 들어있지 않은 셀은 NULL이 아닌 빈 문자열로 저장됨
 - ✓ 따라서 IS NULL을 사용하여 레코드를 조회하면 결과가 나오지 않음
- 해결 방법) UPDATE문으로 문자열을 NULL로 변경하기

UPDATE 고객
SET 지역 = NULL
WHERE 지역 = '';

→ 대입 update
→ 같다

- 지역 컬럼의 값이 NULL인 레코드를 다시 검색하기

▶ 실행결과

```
SELECT *
FROM 고객
WHERE 지역 IS NULL;
```

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
ACDIR	굿모닝서울	이윤수	영업 과장	송파구 잠실동 220	서울특별시	서울	(02)978-1984	15911
ADHTR	알케미 산사	김병현	영업 사원	성북구 흑석 3동 140-3	서울특별시	서울	(02)345-1945	406
ANKFR	오리간동역	최지수	마케팅 과장	성북구 길음동 136-11	서울특별시	서울	(02)123-0345	6119
ANSFR	서보珉현에프	김민경	영업 사원	종로구 대청동 36-8	서울특별시	서울	(02)211-2221	729
ATRAN	대장 인터내셔널	백순용	대표 이사	서초구 방배동 883-11	서울특별시	서울	(02)681-6889	128790
AUSBL	계원인터넷내셔널	박지혜	영업 사원	서대문구 남기회 1동 121	서울특별시	서울	(02)811-1234	47865
BCPTD	비컴퓨트드	대표 이사	대표 이사	서대문구 남기회 1동 121	서울특별시	서울	(02)811-9876	100

5. IS NULL 연산자

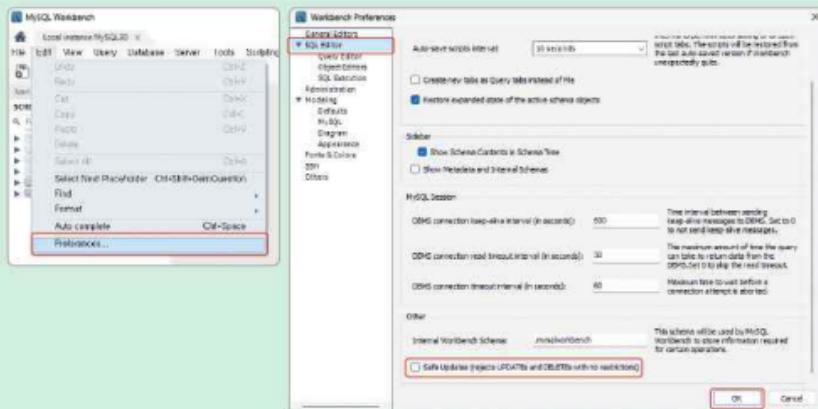
하나 더 알기 Y

UPDATE! DELETE가 제대로 동작하지 않는 경우

MySQL에서는 테이블에서 키를 이용한 UPDATE 또는 DELETE만을 허용합니다. 즉, UPDATE나 DELETE문에서 WHERE절이 없거나 WHERE절에 키 컬럼 외의 컬럼을 사용한 조건문이 사용되면 오류가 발생합니다.

Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column. To disable safe mode, toggle the option in Preferences -> SQL Editor and reconnect.

이를 해결하기 위해서는 Safe Mode 옵션을 해제해주어야 합니다. 워크벤치의 [Edit]-[Preferences] 메뉴에서 [SQL Editor]를 선택한 다음, 체크되어 있는 [Safe Updates] 항목을 클릭하여 체크 해제하고 <OK> 버튼을 클릭한 후 워크벤치를 재실행합니다.



만일 일시적으로 Safe Mode 해제 옵션을 적용하려면 다음과 같이 작성합니다.

```
SET SQL_SAFE_UPDATES = 0;
```

6. IN과 BETWEEN...AND 연산자

■ IN과 BETWEEN...AND 연산자

- WHERE절을 조금 더 쉽고 효율적으로 사용할 수 있게 도와주는 연산자

■ IN

- 동일한 컬럼에 대해서 OR을 수행해야 하는 경우 더 간단히 해결해주는 연산자

■ BETWEEN...AND

- 동일한 컬럼에 대해서 '~이상 ~이하'라는 조건이 있는 경우에 더 간단히 해결해주는 연산자

6. IN과 BETWEEN...AND 연산자

■ [예제 2-14] 담당자직위가 '영업 과장'이거나 '마케팅 과장'인 고객에 대하여
고객번호, 담당자명, 담당자직위를 보이시오.

```
SELECT 고객번호
      ,담당자명
      ,담당자직위
FROM 고객
WHERE 담당자직위 = '영업 과장'
  OR 담당자직위 = '마케팅 과장';
```

- IN을 사용한 해결 방법

```
SELECT 고객번호
      ,담당자명
      ,담당자직위
FROM 고객
WHERE 담당자직위 IN ('영업 과장', '마케팅 과장');
```

▶ 실행 결과

고객번호	담당자명	담당자직위
ACD0R	이은숙	영업 과장
ANKFR	최지수	마케팅 과장
ANRFR	김준	마케팅 과장
CCOPI	김희정	영업 과장
CSURI	이소영	영업 과장

6. IN과 BETWEEN...AND 연산자

- [예제 2-15] 마일리지가 100,000점 이상 200,000점 이하인 고객에 대해 담당자명, 마일리지를 보이시오.

```
SELECT 담당자명  
      ,마일리지  
FROM 고객  
WHERE 마일리지 >= 100000  
      AND 마일리지 <= 200000;
```

- BETWEEN...AND를 사용한 해결 방법

```
SELECT 담당자명  
      ,마일리지  
FROM 고객  
WHERE 마일리지 BETWEEN 100000 AND 200000;
```

포함

▶ 실행결과

담당자명	마일리지
배순봉	128790
한현구	116898

7. LIKE 연산자

■ LIKE 연산자

- 특정 문자열이 지정된 패턴과 일치하는지 확인할 때 사용함
- 와일드카드 문자(%, _)와 함께 사용하여 원하는 결과를 얻을 수 있음
- LIKE와 와일드카드 문자를 함께 사용하지 않으면 = 기호를 사용한 것과 동일한 결과가 나옴

표 2-1 와일드카드 문자와 사용 예

와일드카드 문자	설명	예시 문제	SQL 코드
%	0개 이상의 문자를 가진 문자열을 포함	주소 컬럼 값 중 '혹석'이 들어가는 것	주소 LIKE '%혹석%'
		도시 컬럼 값 중 '광역시'로 끝나는 것	도시 LIKE '%광역시'
_(밀줄)	정확히 한 개의 문자를 포함	고객번호 두 번째가 'C'인 것	고객번호 LIKE '_C'
		고객번호 세 번째가 'C'인 것	고객번호 LIKE '__C'
		고객번호가 5글자이면서 세 번째가 'C'인 것	고객번호 LIKE '__C__'

하나 더 알기 V

와일드카드 문자란?

SQL에서 와일드카드(Wildcard) 문자는 문자열을 비교하거나 검색할 때 패턴 매칭을 위해 사용되는 특수 문자를 나타냅니다. 와일드카드 문자를 사용하면 일부 문자열을 일치시키거나 패턴에 따라 문자열을 검색할 수 있어 데이터 검색 및 필터링에 유용합니다.



7. LIKE 연산자

- [예제 2-16] 도시가 '광역시'이면서 고객번호 두 번째 글자 또는 세 번째 글자가 'C'인 고객의 모든 정보를 보이시오.

```
SELECT *
FROM 고객
WHERE 도시 LIKE '%광역시'
AND (고객번호 LIKE '_C%' OR 고객번호 LIKE '__C%');
```

▶ 실행 결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
IOKOU	인티비스	정대왕	회계 과장	서구 연희동 400-16	인천광역시	02-628-12345	(032)44-1114	857
NTCCE	코스리아 무역	유석희	마케팅 과장	남동구 간석 3동 270-8	인천광역시	02-628-12345	(032)944-3687	112
RUCPE	뉴가든상사	김지호	영업 사원	금정구 청룡동 168-14	부산광역시	051-123-45678	(051)12-1122	1177

7. LIKE 연산자

■ 조건의 우선순위

- WHERE절에서 조건의 우선순위를 지정해야 하는 경우 소괄호(())를 사용함
- 우선순위를 표현하지 않은 경우에는 잘못된 결과가 반환될 수 있음
- [예] [예제 2-16]에서 괄호를 넣지 않으면 A가 먼저 수행된 후에 B가 수행됨

WHERE 도시 LIKE '%광역시' AND 고객번호 LIKE '_C%' OR 고객번호 LIKE '__C%';



- 따라서 C와 같이 괄호를 사용하여 우선순위를 명시적으로 지정해야 함

WHERE 도시 LIKE '%광역시' AND (고객번호 LIKE '_C%' OR 고객번호 LIKE '__C%');



7. LIKE 연산자

하나 더 알기 V

REGEXP 연산자

REGEXP 연산자 또는 REGEXP_LIKE 함수를 사용하면 LIKE보다 좀 더 정교한 패턴 매칭을 통해 데이터를 검색할 수 있습니다. 예를 들어 '푸드'가 들어가 있는 고객회사명을 검색해보겠습니다. REGEXP 연산자를 사용하면 컬럼명 REGEXP 패턴 형식으로 작성합니다.

```
SELECT *
FROM 고객
WHERE 고객회사명 REGEXP '푸드';
```

REGEXP_LIKE 함수를 사용하면 함수명 안에 컬럼명과 패턴을 함께 넣습니다.

```
SELECT *
FROM 고객
WHERE REGEXP_LIKE(고객회사명, '푸드');
```

▶ 실행 결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
BQQZA	비큐큐푸드	할수영	대표 이사	서초구 서초동 120-3	서울특별시		(02)456-9876	100
COOPT	친상푸드	김효정	영업 과장	구로구 오류 1동 101	서울특별시		(02)100-1670	1034
FCRTH	플라워푸드	최옥현	영업 사원	종로구 내각동 190-53	서울특별시		(02)954-1397	364

7. LIKE 연산자

[표 2-2]를 통해 몇 가지 패턴에 대한 사용 방법을 확인해봅시다.

표 2-2 자주 사용되는 패턴 문자와 예시

패턴 문자	설명	예시 문제 → SQL 코드
.	단일 문자	고객회사명이 5글자 이상 → WHERE 고객회사명 REGEXP '.....'
	OR	도시가 '과천시' 또는 '오산시' → WHERE 도시 REGEXP '과천시 오산시'
[]	[] 안에 나열된 패턴 에 해당하는 문자열	도시가 '인천광역시' 또는 '부산광역시' 또는 '대전광역시' → WHERE 도시 REGEXP '[인천 부산 대전]광역시'
^	시작하는 문자열	'한'으로 시작하는 고객회사명 → WHERE 고객회사명 REGEXP '^한'
\$	끝나는 문자열	고객회사명이 5글자이면서 데이터 중 4번째 글자가 '푸'인 고객회사명 → WHERE 고객회사명 REGEXP '^...푸.\$'
*	0번 이상	'정'이 0번 이상 들어있는 담당자명 → WHERE 담당자명 REGEXP '정*'
+	1번 이상	'정'이 1번 이상 들어있는 담당자명 → WHERE 담당자명 REGEXP '정+'
[^문자]	괄호 안의 문자 제외	고객번호 맨 뒷자리에 'H'~'Z'가 들어가지 않는 고객번호 → WHERE 고객번호 REGEXP '[^H-Z]\$'
{m}	m회	고객회사명이 5글자 → WHERE 고객회사명 REGEXP '^.{5}\$'; 'T'나'S'가 2회 나타나는 고객번호(TT, SS, TS, ST) → WHERE 고객번호 REGEXP '[TS]{2}';
{m,n}	m회 이상 n회 이하	'T'나'S'가 1회 이상 2회 이하 나타나는 고객번호(T, S, TT, SS, TS, ST) → WHERE 고객번호 REGEXP '[TS]{1,2}';

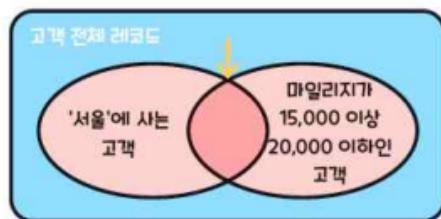
패턴을 다양하게 응용하면 복잡한 조건의 데이터도 잘 매칭하여 검색할 수 있습니다.

점검문제

점검문제

문제 1

다음 실행 결과를 참고하여 '서울'에 사는 고객 중에 마일리지가 15,000점 이상 20,000점 이하인 고객의 모든 정보를 보이시오.



▶ 실행결과

고객번호	고객회원ID	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
ACDDDR	웃모님서울	이은숙	영업 관광	송파구 청담동 220	서울특별시	08:00~18:00	(02)978-1984	15911

문제 2

한빛무역의 고객들은 어느 지역, 어느 도시에 사는지 지역과 도시를 한 번씩만 보이시오. 이때 결과를 지역 순으로 나타내고, 동일 지역에 대해서는 도시 순으로 나타내보ınız.

▶ 실행 결과

지역	도시
대전광역시	대전
부산광역시	부산
서울특별시	서울
인천광역시	인천
원주시	강원도
춘천시	강원도
과천시	경기도
광명시	경기도
구리시	경기도
군현시	경기도

점검문제

문제 3

'춘천시'나 '과천시' 또는 '광명시'에 사는 고객 중에서 담당자직위에 '이사' 또는 '사원'이 들어가는 고객의 모든 정보를 보이시오.

▶ 실행 결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
AIHTR	시엔그룹	김성민	영업 사원	가락동 301	광명시	경기도	(02)31-0345	8788
LAAHI	설민식 품앗사	신민주	영업 사원	충암로 3가 150-1	춘천시	강원도	(0369)39-1243	382
LKFCFO	진영주택	조희진	대표 이사	중앙동 427-10	과천시	경기도	(02)768-7688	8607
NSHCO	운해식품	이승연	영업 사원	소하동 11-3	광명시	경기도	(02)489-9184	387

문제 4

'광역시'나 '특별시'에 살지 않는 고객들 중에서 마일리지가 많은 상위 고객 3명의 모든 정보를 보이시오.

▶ 실행 결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
LZAWO	오투락	수민경	대표 이사	건입동 111-16	제주시	제주도	(064)643-2871	8838
AHTR	시엔그룹	김성민	영업 사원	가락동 301	광명시	경기도	(02)31-0345	8788
BEVBS	케미리탈리스	오선주	영업 사원	구산동 17-111	김해시	경상남도	(0525)342-3333	8758

점검문제

문제 5

지역에 값이 들어있는 고객 중에서 담당자직위가 '대표 이사'인 고객을 빼고 보이시오.

▶ 실행(결과)

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
AIHTR	씨엔그룹	김성민	영업 사원	가락동 301	광명시	경기도	(02)31-0345	8788
ANRFR	남해중합식품	강준	마케팅 과장	수락동 435	구리시	경기도	(034)756-2091	106
BEVBS	케이라할리스	오선주	영업 사원	구산동 17-111	김해시	경상남도	(052)342-3333	8758
	친가식품		영업 사원	108-36			(033)945-1022	

Chapter 03

함수1: 단일 행 함수



목차

1. SQL 함수의 개요
2. 단일 행 함수의 종류와 사용법
3. 기타 단일 행 함수

학습목표

- 함수의 개념과 종류를 이해할 수 있습니다.
- 단일 행 함수인 문자형 함수, 숫자형 함수, 날짜/시간형 함수를 활용할 수 있습니다.
- 기타 유용한 함수에 대해 이해할 수 있습니다.

Preview



Section 01

SQL 함수의 개요

1. 함수의 개념

■ 함수

- 특별한 목적의 작업을 수행하기 위해 독립적으로 설계된 프로그램 코드의 집합
- 사용자는 어떤 로직에 의해서 함수가 동작하는지에 대해 알 필요 없이 필요한 값을 주고, 원하는 값을 얻으면 됨
- 함수를 실행할 때 필요한 값은 매개변수를 통해 전달됨
- [예] 문자열의 길이를 반환하는 LENGTH 함수
 - ✓ 함수 사용에 필요한 매개변수 값으로 문자열을 입력함

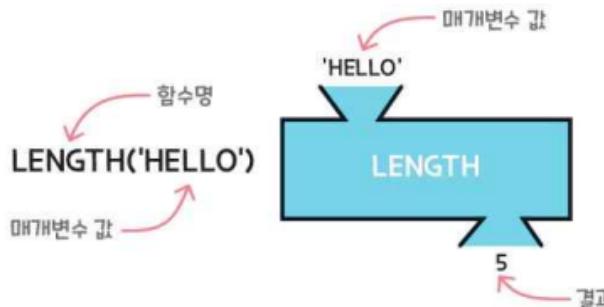


그림 3-1 함수의 사용 방법

2. SQL 함수의 개념과 종류

■ SQL 함수

- 데이터를 조회하거나 집계, 저장, 수정하는 과정에서 값을 가공하기 위해 제공되는 모듈화된 기능
- SQL 함수는 DBMS에 미리 만들어져 저장되어 있는지, 사용자가 필요에 따라서 직접 만드는지에 따라 내장 함수와 사용자 정의 함수로 구분함

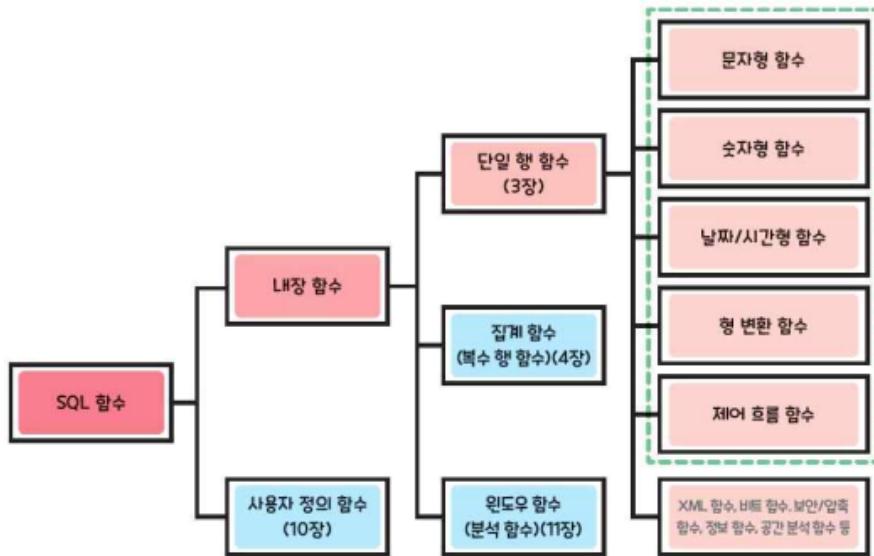


그림 3-2 SQL 함수의 종류

Section 02

단일 행 함수의
종류와 사용법

1. 문자형 함수

■ 문자형 함수

- 문자열을 입력하면 문자열이나 숫자 값을 반환하는 함수

■ CHAR_LENGTH(), LENGTH()

- CHAR_LENGTH() : 문자의 개수를 반환하는 함수
- LENGTH() : 문자열에 할당된 바이트(Byte) 수를 반환하는 함수
- 형식

CHAR_LENGTH(문자열)

LENGTH(문자열)

1. 문자형 함수

■ [예제 3-1] 영문 'HELLO'와 한글 '안녕'의 문자 개수를 세고 결과를 확인하시오.

```
SELECT CHAR_LENGTH('HELLO') •———— 영문자 개수 반환  
, LENGTH('HELLO') •———— 영문자의 바이트 수 반환  
, CHAR_LENGTH('안녕') •———— 문자 개수 반환  
, LENGTH('안녕'); •———— UTF-8의 바이트 수 반환
```

▶ 실행 결과

CHAR_LENGTH('HELLO')	LENGTH('HELLO')	CHAR_LENGTH('안녕')	LENGTH('안녕')
5	5	2	6

1. 문자형 함수

■ CONCAT(), CONCAT_WS()

- CONCAT() : 문자열을 연결할 때 사용하는 함수
- CONCAT_WS() : 구분자와 함께 문자열을 연결할 때 사용하는 함수
- 형식

CONCAT(문자열1, 문자열2, ...)

CONCAT_WS(구분자, 문자열1, 문자열2, ...)

■ [예제 3-2] CONCAT() 함수와 CONCAT_WS() 함수를 사용하여 문자열을 연결했을 때의 결괏값을 비교하시오.

```
SELECT CONCAT('DREAMS', 'COME', 'TRUE') •———— 문자열 연결  
,CONCAT_WS('-', '2023', '01', '29'); •———— 구분자 '-'로 연결
```

▶ 실행결과

CONCAT('DREAMS', 'COME', 'TRUE')	CONCAT_WS('-', '2023', '01', '29')
DREAMSCOMETRUE	2023-01-29

1. 문자형 함수

■ LEFT(), RIGHT(), SUBSTR()

- LEFT() : 문자열의 왼쪽부터 길이만큼 문자열을 반환하는 함수
- RIGHT() : 문자열의 오른쪽부터 길이만큼 문자열을 반환하는 함수
- SUBSTR() : 지정한 위치로부터 길이만큼의 문자열을 반환하는 함수
- 형식

LEFT(문자열, 길이)

RIGHT(문자열, 길이)

SUBSTR(문자열, 시작_위치, 길이) 또는 SUBSTRING(문자열, 시작_위치, 길이)

1. 문자형 함수

■ [예제 3-3] 'SQL 완전정복' 문자열에서 지정한 길이만큼 문자열을 반환하시오.

SELECT LEFT('SQL 완전정복', 3) • 원쪽부터 3문자

,RIGHT('SQL 완전정복', 4) • 오른쪽부터 4문자

,SUBSTR('SQL 완전정복', 2, 5) • 2번째 문자부터 5문자

,SUBSTR('SQL 완전정복', 2); • 2번째 문자부터 끝까지

S E

▶ 실행결과

LEFT('SQL 완전정복', 3)	RIGHT('SQL 완전정복', 4)	SUBSTR('SQL 완전정복', 2, 5)	SUBSTR('SQL 완전정복', 2)
SQL	완전정복	QL 완전	QL 완전정복



1. 문자형 함수

■ SUBSTRING_INDEX()

- SUBSTRING_INDEX() : 지정한 구분자를 기준으로 문자열을 분리해서 가져올 때 사용하는 함수
- 형식

SUBSTRING_INDEX(문자열, 구분자, 인덱스)

■ [예제 3-4] '서울시 동작구 흑석로'에서 앞/뒤 2번째 위치에 있는 공백(' ')을 기준으로 문자열을 분리한 결과를 확인하시오.

```
SELECT SUBSTRING_INDEX('서울시 동작구 흑석로', ' ', 2) •———— 원쪽부터 두 번째 공백 이후 제거  
      ,SUBSTRING_INDEX('서울시 동작구 흑석로', ' ', -2); •———— 오른쪽부터 두 번째 공백 이전 제거
```

▶ 실행 결과

SUBSTRING_INDEX('서울시 동작구 흑석로', '', 2)	SUBSTRING_INDEX('서울시 동작구 흑석로', '', -2)
서울시 동작구	동작구 흑석로

1. 문자형 함수

LPAD() RPAD()

- LPAD()와 RPAD() : 지정한 길이에서 문자열을 제외한 빈칸을 특정 문자로 채울 때 사용하는 함수
- LPAD()는 왼쪽에, RPAD()는 오른쪽에 특정 문자를 채움
- 형식

LPAD(문자열, 길이, 채울_문자열)

RPAD(문자열, 길이, 채울_문자열)

- [예제 3-5] 'SQL' 문자열의 앞 7칸에 # 기호를 채우고, SQL 뒤 2칸에는 * 기호를 채우시오.

SELECT LPAD('SQL', 10, '#') •———— 문자열 왼쪽에 # 채우기
, RPAD('SQL', 5, '*'); •———— 문자열 오른쪽에 * 채우기

▶ 실행결과

LPAD('SQL', 10, '#')	RPAD('SQL', 5, '*')
#####SQL	SQL**

1. 문자형 함수

LTRIM(), RTRIM()

- LTRIM() : 왼쪽의 공백을 제거할 때 사용하는 함수
- RTRIM() : 오른쪽 공백을 제거할 때 사용하는 함수
- 형식

LTRIM(문자열)

RTRIM(문자열)

[예제 3-6] 'SQL' 문자열의 앞/뒤에 있는 공백을 제거하고, 결과를 문자열의 길이로 확인하시오.

```
SELECT LENGTH(LTRIM(' SQL ')) •———— 왼쪽 공백을 지운 후 길이 반환  
      ,LENGTH(RTRIM(' SQL ')) •———— 오른쪽 공백을 지운 후 길이 반환  
      ,LENGTH(TRIM(' SQL ')); •———— 양쪽 공백을 지운 후 길이 반환
```

▶ 실행결과

LENGTH(LTRIM(' SQL '))	LENGTH(RTRIM(' SQL '))	LENGTH(TRIM(' SQL '))
4	4	3

1. 문자형 함수

한국어

■ TRIM(BOTH/LEADING/TRAILING)

- 양쪽에 있는 동일 문자열을 제거하고자 할 때에는 TRIM()의 첫 번째 매개변수에 BOTH를 넣음
- BOTH 대신 LEADING을 넣으면 왼쪽에 있는 문자열이, TRAILING을 넣으면 오른쪽에 있는 문자열이 제거됨
- 형식

TRIM(문자열)

TRIM(제거할_문자열의_방향(BOTH/LEADING/TRAILING) 제거할_문자열 FROM 문자열)

■ [예제 3-7] 'abcSQLabcbc'에서 BOTH/LEADING/TRAILING을 사용하여 앞뒤에 있는 'abc' 문자열을 제거하시오.

```
SELECT TRIM(BOTH 'abc' FROM 'abcSQLabcbc') •———— 양쪽의 모든 abc 제거  
,TRIM(LEADING 'abc' FROM 'abcSQLabcbc') •———— 왼쪽의 abc 제거  
,TRIM(TRAILING 'abc' FROM 'abcSQLabcbc'); •———— 오른쪽의 abc 제거
```

▶ 실행 결과

TRIM(BOTH '*' FROM '**SQL***)	TRIM(LEADING 'abc' FROM 'abcSQLabbc')	TRIM(TRAILING 'abc' FROM 'abcSQLabcbc')
SQL	SQLab	abcSQL

1. 문자형 함수

■ FIELD(), FIND_IN_SET(), INSTR(), LOCATE()

- FIELD() : 여러 문자열 중에서 찾는 문자열이 있으면 문자열의 위치 값을 반환하는 함수. 만약 찾는 문자열이 없으면 0을 반환함

- 형식

```
FIELD(찾을_문자열, 문자열1, 문자열2, ...)
```

- FIND_IN_SET() : 문자열 리스트에서 지정한 문자열을 찾아서 위치 값을 반환함

- 형식

```
FIND_IN_SET(찾을_문자열, 문자열_리스트)
```

- INSTR(), LOCATE() : 기준 문자열 중에서 부분 문자열을 찾아서 위치 값을 반환함

- 형식

```
INSTR(기준_문자열, 부분_문자열)
```

```
LOCATE(부분_문자열, 기준_문자열)
```

1. 문자형 함수

- [예제 3-8] 문자열의 위치를 찾는 4가지 함수 FIELD(), FIND_IN_SET(), INSTR(), LOCATE()를 사용하고 결과를 확인하시오.

```
SELECT FIELD('JAVA', 'SQL', 'JAVA', 'C') •———— 첫 매개변수인 JAVA의 위치  
, FIND_IN_SET('JAVA', 'SQL,JAVA,C') •———— ';'로 구분된 두 번째 매개변수 값 중 JAVA의 위치  
, INSTR('네 인생을 살아라', '인생') •———— 인생의 위치  
, LOCATE('인생', '네 인생을 살아라');
```

▶ 실행 결과

FIELD('JAVA', 'SQL', 'JAVA', 'C')	FIND_IN_SET('JAVA', 'SQL,JAVA,C')	INSTR('네 인생을 살아라', '인생')	LOCATE('인생', '네 인생을 살아라')
2	2	3	3

배열

1. 문자형 함수

■ ELT()

- ELT(): 지정한 위치에 있는 문자열을 반환하는 함수
- FIELD()는 첫 매개변수 값에 찾고자 하는 문자열을 넣어주지만, ELT()는 찾을 문자열의 위치 값을 넣음
- 형식

```
ELT(찾을_문자열_위치, 문자열1, 문자열2, ...)
```

■ [예제 3-9] 문자열의 위치를 찾는 4가지 함수 FIELD(), FIND_IN_SET(), INSTR(), LOCATE()를 사용하고 결과를 확인하시오.

```
SELECT ELT(2, 'SQL', 'JAVA', 'C'); ●———— 두 번째 위치에 있는 문자열
```

▶ 실행 결과

ELT(2, 'SQL', 'JAVA', 'C')
JAVA

1. 문자형 함수

■ REPEAT()

- REPEAT() : 문자열을 반복하고자 할 때 사용하는 함수
 - ✓ 매개변수에 반복할 문자열과 반복할 횟수를 넣음
- 형식

REPEAT(문자열, 횟수)

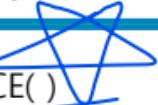
■ [예제 3-10] REPEAT()를 사용하여 '*' 5개를 반복하시오.

SELECT REPEAT('*', 5);

▶ 실행결과

REPEAT('*', 5)

1. 문자형 함수



■ REPLACE()

- REPLACE() : 문자열의 일부를 다른 문자열로 대체하고자 할 때 사용하는 함수
- 형식

REPLACE(문자열, 원래_문자열, 바꿀_문자열)

■ [예제 3-11] '.'로 구분되어 있는 전화번호의 구분자를 '-'로 대체하시오.

```
SELECT REPLACE('010.1234.5678', '.', '-');
```

▶ 실행결과

REPLACE('010.1234.5678', '.', '-')

010-1234-5678

1. 문자형 함수

REVERSE()

- REVERSE() : 문자열을 거꾸로 뒤집을 때 사용하는 함수
- 형식

REVERSE(문자열)

- [예제 3-12] 'OLLEH'를 맨 뒤쪽의 문자부터 나타내시오.

SELECT REVERSE('OLLEH');

▶ 실행결과

REVERSE('OLLEH')

HELLO

2. 숫자형 함수

■ 숫자형 함수

- 숫자를 다루는 단일 행 함수를 통해 올림, 버림, 반올림, 절댓값 반환, 나머지, 제곱승 등의 기능을 수행할 수 있음

■ CEILING(), FLOOR(), ROUND(), TRUNCATE()

- CEILING() : 올림, FLOOR() : 버림, ROUND() : 지정한 위치에서 반올림,
TRUNCATE() : 지정한 위치에서 버림 작업을 수행하는 함수
- 형식

CEILING(숫자)

FLOOR(숫자)

ROUND(숫자)

ROUND(숫자, 반올림할_자릿수)

TRUNCATE(숫자, 버림할_자릿수)

2. 숫자형 함수

- [예제 3-13] 123.56에 대해 CEILING(), FLOOR(), ROUND(), TRUNCATE()를 사용한 결과를 확인하시오.

```
SELECT CEILING(123.56)
      ,FLOOR(123.56)
      ,ROUND(123.56)
      ,ROUND(123.56, 1)
      ,TRUNCATE(123.56, 1);
```

▶ 실행결과

CEILING(123.56)	FLOOR(123.56)	ROUND(123.56)	ROUND(123.56, 1)	TRUNCATE(123.56, 1)
124	123	124	123.6	123.5

2. 숫자형 함수

■ ABS(), SIGN()

- ABS() : 절댓값을 반환하는 함수
- SIGN() : 양수의 경우 1, 음수의 경우 -1을 반환하는 함수
- 형식

ABS(숫자)

SIGN(숫자)

■ [예제 3-14] -120과 120의 절댓값과 음수, 양수 여부를 판별하시오.

```
SELECT ABS(-120)
      ,ABS(120)
      ,SIGN(-120)
      ,SIGN(120);
```

▶ 실행결과

ABS(-120)	ABS(120)	SIGN(-120)	SIGN(120)
120	120	-1	1

2. 숫자형 함수

■ MOD()

- MOD(): 나머지를 구하는 함수로 세 가지 방법을 사용할 수 있음
- 형식

MOD(숫자1, 숫자2)

숫자1 % 숫자2

숫자1 MOD 숫자2

■ [예제 3-15] 203을 4로 나눈 나머지를 확인하시오

```
SELECT MOD(203, 4)
      ,203 % 4
      ,203 MOD 4;
```

▶ 실행결과

MOD(203, 4)	203 % 4	203 MOD 4
3	3	3

2. 숫자형 함수

■ POWER(), SQRT(), RAND()

- POWER() : n제곱승 값을 반환하는 함수
- SQRT() : 제곱근 값을 반환하는 함수
- RAND() : 0과 1사이 임의의 실수 값을 반환하는 함수
 - ✓ 매개변수 값을 넣지 않으면 실행할 때마다 0과 1 사이에 있는 임의의 실수를 반환함
 - ✓ RAND() 안에 시드(Seed)를 설정하면 매번 동일한 임의의 값을 얻을 수 있음
 - ✓ ROUND(), TRUNCATE() 등의 함수를 같이 사용하면 임의의 정수를 만들 수 있음
- 형식

POWER(숫자1, 숫자2) ●———— 숫자1의 숫자2 제곱승

SQRT(숫자)

RAND()

RAND(숫자)

2. 숫자형 함수

■ [예제 3-16] 제곱과 제곱근, 랜덤값을 구하는 함수를 사용하고, 그 결과를 보아시오.

```
SELECT POWER(2, 3) }  
    ,SQRT(16)  
    ,RAND()  
    ,RAND(100)  
    ,ROUND(RAND() * 100) ————— 0~100 사이의 정수
```

▶ 실행결과

POWER(2, 3)	SQRT(16)	RAND()	RAND(100)	ROUND(RAND() * 100)
8	4	0.838829533046884	0.17353134804734155	76

3. 날짜/시간형 함수

■ 현재 날짜/시간 반환 함수 날짜 시간

- NOW(), SYSDATE(), CURDATE(), CURTIME()
 - ✓ 현재 날짜와 시간을 반환하는 함수
- NOW(), SYSDATE() : 시스템의 현재 날짜와 시간을 반환함
- CURDATE() : 시스템의 현재 날짜를 반환함
- CURTIME() : 시스템의 현재 시간을 반환함

■ [예제 3-17] 현재 날짜와 시간을 다양한 형태로 반환하시오.

```
SELECT NOW()
      ,SYSDATE()
      ,CURDATE()
      ,CURTIME();
```

▶ 실행 결과

NOW()	SYSDATE()	CURDATE()	CURTIME()
2023-01-31 14:31:07	2023-01-31 14:31:07	2023-01-31	14:31:07

3. 날짜/시간형 함수

■ 연도, 분기, 월, 일, 시, 분, 초 반환 함수

- YEAR(), QUARTER(), MONTH(), DAY(), HOUR(), MINUTE(), SECOND()
 - ✓ 날짜 중 해당 값을 반환함
 - ✓ [예] YEAR()는 날짜 중 연도를, QUARTER()는 날짜 중 분기를 반환함

■ [예제 3-18] 날짜에 관한 다양한 함수의 결과를 확인하시오.

```
SELECT NOW()          •———— 현재 날짜  
,YEAR(NOW())      •———— 연도  
,QUARTER(NOW())   •———— 분기  
,MONTH(NOW())     •———— 월  
,DAY(NOW())        •———— 일  
,HOUR(NOW())       •———— 시  
,MINUTE(NOW())    •———— 분  
,SECOND(NOW());  •———— 초
```

▶ 실행결과

NOW()	YEAR(NOW())	QUARTER(NOW())	MONTH(NOW())	DAY(NOW())	HOUR(NOW())	MINUTE(NOW())	SECOND(NOW())
2023-01-29 20:23:32	2023	1	1	29	20	23	32

3. 날짜/시간형 함수

■ 기간 반환 함수

- DATEDIFF(), TIMESTAMPDIFF()

✓ 지정한 기간을 반환하는 함수

날짜 (DATEDIFF() : 기간을 일자 기준으로 반환함

• TIMESTAMPDIFF() : 기간을 지정한 단위 기준으로 보여줌

✓ DATEDIFF()는 끝 일자를 앞에 넣어주고, TIMESTAMPDIFF()는 끝 일자를 뒤에 넣음

• 형식

DATEDIFF(끝_일자, 시작_일자)

TIMESTAMPDIFF(단위, 시작_일자, 끝_일자)

표 3-1 TIMESTAMPDIFF() 함수에 매개변수로 들어가는 단위

단위	의미	단위	의미
SECOND	초	WEEK	주
MINUTE	분	MONTH	월
HOUR	시	QUARTER	분기
DAY	일	YEAR	연도

3. 날짜/시간형 함수

■ [예제 3-19] 현재 날짜를 기준으로 날짜 사이의 기간을 확인하시오.

```
SELECT NOW()
    ,DATEDIFF('2025-12-20', NOW()) •————①
    ,DATEDIFF(NOW(), '2025-12-20')
    ,TIMESTAMPDIFF(YEAR, NOW(), '2025-12-20')
    ,TIMESTAMPDIFF(MONTH, NOW(), '2025-12-20')
    ,TIMESTAMPDIFF(DAY, NOW(), '2025-12-20'); •————②
```

▶ 실행결과

NOW()	DATEDIFF('2025-12-20', NOW())	DATEDIFF(NOW(), '2025-12-20')	TIMESTAMPDIFF(YEAR, NOW(), '2025-12-20')	TIMESTAMPDIFF(MONTH, NOW(), '2025-12-20')	TIMESTAMPDIFF(DAY, NOW(), '2025-12-20')
2023-01-31 14:35:11	1054	-1054	2	34	1053

3. 날짜/시간형 함수

■ 기간을 반영하는 날짜 함수

- ADDDATE() : 지정한 날짜를 기준으로 그 기간만큼 더한 날짜를 반환하는 함수
- SUBDATE() : 기간만큼 뺀 날짜를 반환함
- 형식

ADDDATE(날짜, 기간)

또는 ADDDATE(날짜, INTERVAL 기간 단위)

SUBDATE(날짜, 기간)

또는 SUBDATE(날짜, INTERVAL 기간 단위)

■ [예제 3-20] 오늘 날짜 및 오늘 날짜로부터 50일 후, 50개월 후, 50시간 전의 날짜를 확인하시오.

SELECT NOW()

,ADDDATE(NOW(), 50) ● ① 50일 후
,ADDDATE(NOW(), INTERVAL 50 DAY) ● ② 50일 후
,ADDDATE(NOW(), INTERVAL 50 MONTH) ● 50개월 후
,SUBDATE(NOW(), INTERVAL 50 HOUR); ● 50시간 전

자동생성

▶ 실행결과

NOW()	ADDDATE(NOW(), 50)	ADDDATE(NOW(), INTERVAL 50 DAY)	ADDDATE(NOW(), INTERVAL 50 MONTH)	SUBDATE(NOW(), INTERVAL 50 HOUR)
2023-01-31 14:59:41	2023-03-22 14:59:41	2023-03-22 14:59:41	2027-03-31 14:59:41	2023-01-29 12:59:41

3. 날짜/시간형 함수

■ 기타 날짜 반환 함수

- LAST_DAY() : 해당 월의 마지막 일자를 반환함
- DAYOFYEAR() : 현재 연도에서 며칠이지났는지를 반환함
- MONTHNAME()은 월을 영문으로, WEEKDAY()는 요일을 정수로 보여줌
- 형식

LAST_DAY(날짜)

DAYOFYEAR(날짜)

MONTHNAME(날짜)

WEEKDAY(날짜)

3. 날짜/시간형 함수

- [예제 3-21] 오늘 날짜를 기준으로 이번 달 마지막 날짜와 일 년 중 몇 일째인지, 그리고 월 이름과 요일을 확인하시오.

```
SELECT NOW()
      ,LAST_DAY(NOW())
      ,DAYOFYEAR(NOW())
      ,MONTHNAME(NOW())
      ,WEEKDAY(NOW());
```

▶ 실행결과

NOW()	LAST_DAY(NOW())	DAYOFYEAR(NOW())	MONTHNAME(NOW())	WEEKDAY(NOW())
2023-01-31 15:09:37	2023-01-31	31	January	1

Section 03

기타 단일 행 함수

1. 형 변환 함수

■ 형 변환 함수

- 데이터를 검색, 삽입할 때 컬럼에 맞는 형식으로 지정하지 않으면 오류가 나는 경우에 사용하는 함수

■ CAST(), CONVERT()

- 원하는 형태로 데이터타입을 변경하여 처리하거나 확인할 수 있음
- 형식

CAST(값 AS 데이터타입)

CONVERT(값, 데이터타입)

1. 형 변환 함수

- [예제 3-22] 문자 '1'을 부호 없는 숫자 형식으로, 숫자 2를 문자 형식으로 변환하시오.

```
SELECT CAST('1' AS UNSIGNED)
      ,CAST(2 AS CHAR(1))
      ,CONVERT('1' AS UNSIGNED)
      ,CONVERT(2, CHAR(1));
```

▶ 실행 결과

CAST('1' AS UNSIGNED)	CAST(2 AS CHAR(1))	CONVERT('1', UNSIGNED)	CONVERT(2, CHAR(1))
1	2	1	2

2. 제어 흐름 함수

■ 제어 흐름 함수

- 프로그램의 흐름을 제어할 때 사용함
- 조건에 따른 결과를 SQL 문장 하나로 얻을 수 있음

■ IF

- IF(조건, 수식1, 수식2) : 조건의 결과가 참이면 수식1을 반환하고, 그렇지 않으면 수식2의 결과를 반환함
- 형식

IF(조건, 수식1, 수식2)

■ [예제 3-23] 12,500원짜리 제품을 450개 이상 주문한 금액이 5,000,000원 이상이면 '초과달성', 미만이면 '미달성'이라고 보이시오.

SELECT IF(12500 * 450 > 5000000, '초과달성', '미달성');

조건 참 거짓

▶ 실행결과

IF(12500 * 450 > 5000000, '초과달성', '미달성')

초과달성

2. 제어 흐름 함수

■ IFNULL(), NULLIF()

- NULL과 관련된 제어 흐름 함수

~~• IFNULL()~~: 수식1이 NULL이 아니면 수식1의 값을 반환하고, NULL이면 수식2의 값을 반환함

• NULLIF() : 두 수식 값을 비교하여 값이 같으면 NULL을 반환하고, 값이 다르면 수식1의 값을 반환함

- 형식

IFNULL(수식1, 수식2)

NULLIF(수식1, 수식2)

2. 제어 흐름 함수

- [예제 3-24] IFNULL()의 첫 매개변수 값이 NULL인지 여부에 따라 어떻게 결과가 다르게 나오는지 확인하시오.

```
SELECT IFNULL(1, 0)
      ,IFNULL(NULL, 0)
      ,IFNULL(1/0, 'OK');
```

▶ 실행결과

IFNULL(1, 0)	IFNULL(NULL, 0)	IFNULL(1/0, 'OK')
1	0	OK

- [예제 3-25] NULLIF()을 사용하여 두 결과를 비교하시오.

```
SELECT NULLIF(12 * 10, 120)
      ,NULLIF(12 * 10, 1200);
```

▶ 실행결과

NULLIF(12 * 10, 120)	NULLIF(12 * 10, 1200)
NULL	120

3. CASE문

■ CASE문

- 함수는 아니지만 조건 비교가 여러 개일 때 유용하게 사용할 수 있음
- WHEN 조건1 THEN 값1 WHEN 조건2 THEN 값2...
 - ✓ 모든 조건을 만족하지 않으면 ELSE 다음에 값을 넣어줌
 - ✓ CASE문은 END로 마무리되어야 함

■ [예제 3-26] 주문 금액이 5,000,000원 이상이면 '초과달성', 4,000,000원 이상이면 '달성' 그 나머지는 '미달성'이라고 할 때, 12,500원짜리 제품을 450개 이상 주문했다면 어디에 해당하는지 보이시오.

```
SELECT CASE WHEN 12500 * 450 > 5000000 THEN '초과달성'  
          WHEN 2500 * 450 > 4000000 THEN '달성'  
          ELSE '미달성'  
END;
```

▶ 실행결과

CASE WHEN 12500 * 450 > 5000000 THEN '초과달성' WHEN 2500 * 450 > 4000000 THEN '달성' ELSE '미달성' END 초과달성
--

점검문제

문제 1
문제 1

다음 조건에 따라 고객 테이블에서 고객회사명과 전화번호를 다른 형태로 보이도록 함수를 사용해봅시다. 고객회사명2와 전화번호2를 만드는 조건은 다음과 같습니다.

조건

- 고객회사명2 : 기존 고객회사명 중 앞의 두 자리를 *로 변환한다.
- 전화번호2 : 기존 전화번호의 (xxx)xxx-xxxx 형식을 xxx-xxx-xxxx 형식으로 변환한다.

▶ 실행결과

고객회사명	고객회사명2	전화번호	전화번호2
굿모닝서울	**닝서울	(02)978-1984	02-978-1984
얼케이 상사	**이 상사	(02)345-1945	02-345-1945
씨언그룹	**그룹	(02)31-0345	02-31-0345



점검문제

문제 2

다음 조건에 따라 주문세부 테이블의 모든 컬럼과 주문금액, 할인금액, 실제 주문금액을 보이시오. 이때 모든 금액은 1의 단위에서 버림을 하고 10원 단위까지 보이도록 합니다.

조건

- 주문금액 : 주문수량 * 단가
- 할인금액 : 주문수량 * 단가 * 할인율
- 실주문금액 : 주문금액 – 할인금액



▶ 실행결과

주문번호	제품번호	단가	주문수량	할인율	주문금액	할인금액	실주문금액
H0250	51	4200	35	0.15	147000	22050	124950
H0250	65	1700	15	0.15	25500	3820	21680
H0251	22	1700	6	0.05	10200	510	9690
H0251	57	1600	15	0.05	24000	1200	22800
	65	2000	0	0	0	0	0

문제 3

사원 테이블에서 전체 사원의 이름, 생일, 만나이, 입사일, 입사일수, 입사한 지 500일 후의 날짜를 보이시오.

▶ 실행결과

이름	생일	만나이	입사일	입사일수	500일후
이소미	1985-12-05	38	2019-04-13	1706	2020-08-25
배재웅	1973-02-17	50	2019-01-01	1808	2020-05-15
유대현	1988-08-27	35	2019-03-14	1736	2020-07-26
최소민	1987-09-17	36	2019-04-15	1704	2020-08-27
	1980-03-01	40	2018-12-29	1844	2021-05-12



점검문제

문제 4

고객 테이블에서 도시 컬럼의 데이터를 다음 조건에 따라 '대도시'와 '도시'로 구분하고, 마일리지 점수에 따라서 'VVIP', 'VIP', '일반 고객'으로 구분하시오.



조건

- 도시 구분 : '특별시'나 '광역시'는 '대도시'로, 그 나머지 도시는 '도시'로 구분한다.
- 마일리지 구분 : 마일리지가 100,000점 이상이면 'VVIP고객', 10,000점 이상이면 'VIP고객', 그 나머지는 '일반고객'으로 구분한다.

▶ 실행 결과

담당자명	고객회사명	도시	도시구분	마일리지	마일리지구분
이운광	굿모닝서울	서울특별시	대도시	15911	VIP고객
김병현	열체이 상사	서울특별시	대도시	406	일반고객
김성민	씨엔그룹	광명시	도시	8788	일반고객

점검문제

문제 5

주문 테이블에서 주문번호, 고객번호, 주문일 및 주문년도, 분기, 월, 일, 요일, 한글요일을 보이시오.

▶ 실행 결과

주문번호	고객번호	주문일	주문년도	주문분기	주문월	주문일	주문요일	한글요일
H0248	NETVI	2020-03-12	2020	1	3	12	3	화요일
H0249	MSPTO	2020-03-13	2020	1	3	13	4	수요일
H0250	NARHA	2020-03-16	2020	1	3	16	0	목요일
H0251	CTEVI	2020-03-16	2020	1	3	16	0	목요일
H0252	PRDSU	2020-03-17	2020	1	3	17	1	금요일

문제 6

주문 테이블에서 요청일보다 발송일이 7일 이상 늦은 주문내역을 보이시오.

▶ 실행 결과

주문번호	고객번호	사용번호	주문일	요청일	발송일	지연일수
H0380	NGOHO	E07	2020-08-20	2020-09-17	2020-09-24	7
H0423	URLGO	E06	2020-10-01	2020-10-15	2020-11-02	18
H0427	CCOPT	E04	2020-10-05	2020-11-02	2020-11-09	7
H0451	ICKQU	E04	2020-10-28	2020-11-11	2020-11-18	7

Chapter 04

함수2: 집계 함수



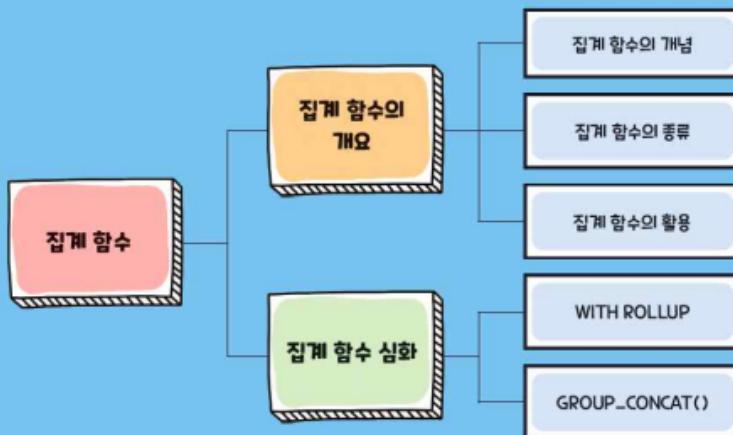
목차

1. 집계 함수의 개요
2. 집계 함수 심화

학습목표

- 집계 함수를 사용하여 데이터를 요약할 수 있습니다.
- 소계와 총계를 한번에 요약하는 문법에 대해 이해할 수 있습니다.

Preview



Section 01

집계 함수의 개요

1. 집계 함수의 개념

■ 집계 함수(Aggregate Function)

- 여러 행에 걸쳐 있는 값을 묶어서 계산을 수행하여 단일 값을 반환하는 함수
- 레코드의 개수나 합계, 평균, 최댓값, 최솟값 등을 구할 때 사용함

```
SELECT 담당자명  
      ,도시  
      ,마일리지  
FROM 고객  
WHERE 도시 = '대전광역시';
```

담당자명	도시	마일리지
박준선	대전광역시	139
이소연	대전광역시	428
박건호	대전광역시	57746
정수현	대전광역시	975

그림 4-1 집계 함수의 활용

```
SELECT SUM(마일리지) AS 마일리지합  
FROM 고객  
WHERE 도시 = '대전광역시';
```

마일리지합
59258



2. 집계 함수의 종류

■ 집계 함수를 사용하는 SELECT문의 문법

- 형식

```
SELECT 집계_함수  
FROM 테이블명  
[WHERE절];
```

- 집계 함수 중 자주 사용되는 함수의 종류

표 4-1 주요 집계 함수

집계 함수	설명	사용 예
COUNT(* 컬럼)	레코드 개수	COUNT(*), COUNT(마일리지), COUNT(고객번호)
SUM(컬럼 수식)	합	SUM(마일리지), SUM(단가 * 주문수량)
AVG(컬럼 수식)	평균	AVG(단가), AVG(단가 * 주문수량)
MAX(컬럼 수식)	최댓값	MAX(고객번호), MAX(단가 * 주문수량)
MIN(컬럼 수식)	최솟값	MIN(담당자명), MIN(단가)
STDDEV(컬럼 수식)	표준편차	STDDEV(단가)

2. 집계 함수의 종류

- [예제 4-1] 고객 테이블에서 고객번호, 도시, 지역의 개수를 조회하시오.

```
SELECT COUNT(*)
      ,COUNT(고객번호)
      ,COUNT(도시)
      ,COUNT(지역)
  FROM 고객;
```

▶ 실행결과

COUNT(*)	COUNT(고객번호)	COUNT(도시)	COUNT(지역)
93	93	93	27

- [예제 4-2] 고객 테이블의 마일리지 컬럼에 대하여 마일리지 합과 평균 마일리지, 최소 마일리지와 최대 마일리지를 조회하시오.

```
SELECT SUM(마일리지)
      ,AVG(마일리지)
      ,MIN(마일리지)
      ,MAX(마일리지)
  FROM 고객;
```

▶ 실행결과

SUM(마일리지)	AVG(마일리지)	MIN(마일리지)	MAX(마일리지)
802027	8623.9462	100	128790

2. 집계 함수의 활용

■ WHERE절

- WHERE절에 조건을 넣으면 조건에 맞는 레코드에 한해서 값을 요약할 수 있음

■ [예제 4-3] 고객 테이블에서 '서울특별시' 고객에 대해 마일리지합, 평균마일리지, 최소마일리지, 최대마일리지를 조회하시오.

```
SELECT SUM(마일리지)
      ,AVG(마일리지)
      ,MIN(마일리지)
      ,MAX(마일리지)
FROM 고객
WHERE 도시 = '서울특별시';
```

▶ 실행 결과

SUM(마일리지)	AVG(마일리지)	MIN(마일리지)	MAX(마일리지)
556854	11137.0800	100	128790

2. 집계 함수의 활용

■ GROUP BY절

- 그룹별로 묶어서 요약할 때 사용함
- SELECT절에 그룹으로 묶을 컬럼명과 집계 함수를 넣어줌
- SELECT절의 집계 함수를 제외한 나머지 컬럼이나 수식은 반드시GROUP BY절에
도 넣어야 오류가 발생하지 않음
- 형식

```
SELECT 그룹으로_묶을_컬럼명  
      ,집계_함수  
  FROM 테이블명  
  [WHERE절]  
 GROUP BY 그룹으로_묶을_컬럼명 또는 컬럼_순번;
```

2. 집계 함수의 활용

- [예제 4-4] 고객 테이블에서 도시별 고객의 수와 해당 도시 고객들의 평균마일리지를 조회하시오.

```
SELECT 도시
    ,COUNT(*) AS 고객수
    ,AVG(마일리지) AS 평균마일리지
FROM 고객
GROUP BY 도시;
```

▶ 실행결과

도시	고객수	평균마일리지
서울특별시	50	11137.0800
충청시	2	4587.5000
구리시	1	106.0000
김해시	1	8758.0000
광주시	1	122.0000
하남시	7	*****

- GROUP BY절에 컬럼명 대신 SELECT절에 나열되어 있는 컬럼의 순번을 넣을 수도 있음

```
SELECT 도시
    ,COUNT(*) AS 고객수
    ,AVG(마일리지) AS 평균마일리지
FROM 고객
GROUP BY 1;
```

2. 집계 함수의 활용

- [예제 4-5] 담당자직위별로 묶고, 같은 담당자직위에 대해서는 도시별로 묶어서 집계한 결과(고객수와 평균마일리지)를 보이시오. (이때 담당자직위 순, 도시 순으로 정렬하기)



```
SELECT 담당자직위
      ,도시
      ,COUNT(*) AS 고객수
      ,AVG(마일리지) AS 평균마일리지
FROM 고객
GROUP BY 담당자직위
      ,도시
ORDER BY 1, 2;
```

▶ 실행 결과

담당자직위	도시	고객수	평균마일리지
대표 이사	광주시	1	8607.0000
대표 이사	부산광역시	1	2819.0000
대표 이사	서울특별시	13	15741.3846

2. 집계 함수의 활용

■ HAVING절

- GROUP BY의 결과에 대하여 추가 조건을 넣고자 한다면 HAVING절을 사용함
- SELECT절에 있는 컬럼과 함수에 대한 조건만 넣을 수 있음
- 형식

```
SELECT 그룹으로_묶을_컬럼명  
      ,집계_함수  
  FROM 테이블명  
 [WHERE절]  
 GROUP BY 그룹으로_묶을_컬럼명  
 [HAVING절];
```

2. 집계 함수의 활용

- [예제 4-6] 고객 테이블에서 도시별로 그룹을 묶어서 고객수와 평균마일리지를 구하고, 이 중에서 고객수가 10명 이상인 레코드만 걸러내시오.

```
SELECT 도시  
    ,COUNT(*) AS 고객수  
    ,AVG(마일리지) AS 평균마일리지  
FROM 고객  
GROUP BY 도시  
HAVING COUNT(*) >= 10;
```

▶ 실행 결과

도시	고객수	평균마일리지
서울특별시	50	11137.0800

HAVING COUNT(*) >= 10

도시	고객수	평균마일리지
서울특별시	50	11137.0800
광명시	2	4597.5000
구리시	1	106.0000
김해시	1	8758.0000
광주시	1	122.0000
인천광역시	7	13088.0000

SELECT 도시
 ,COUNT(*) 고객수
 ,AVG(마일리지) 평균마일리지
FROM 고객
GROUP BY 도시

그림 4-2 [예제 4-6] SQL문의 흐름

2. 집계 함수의 활용

- [예제 4-7] 고객번호가 'T'로 시작하는 고객에 대해 도시별로 묶어서 고객의 마일리지 합을 구하시오. 이때 마일리지 합이 1,000점 이상인 레코드만 보이시오.

```
SELECT 도시  
      ,SUM(마일리지)  
FROM 고객  
WHERE 고객번호 LIKE 'T%'  
GROUP BY 도시  
HAVING SUM(마일리지) >= 1000;
```

▶ 실행 결과

도시	SUM(마일리지)
서울특별시	1661
부산광역시	7329

```
SELECT 고객번호, 도시, 마일리지  
FROM 고객  
WHERE 고객번호 LIKE 'T%';
```

고객번호	도시	마일리지
TKKOT	안양시	113
TOMAN	서울특별시	1213
TSSLE	의정부시	712
TTICRA	서울시	892
TTMBO	부산광역시	7329
TTTRAN	서울특별시	448

(a) 'T'로 시작하는 고객

그림 4-3 HAVING절의 절차

```
SELECT 도시, SUM(마일리지)  
FROM 고객  
WHERE 고객번호 LIKE 'T%'  
GROUP BY 도시;
```

도시	SUM(마일리지)
안양시	113
서울특별시	1661
의정부시	712
서울시	892
부산광역시	7329

(b) 도시별로 묶은 결과

```
SELECT 도시, SUM(마일리지)  
FROM 고객  
WHERE 고객번호 LIKE 'T%'  
GROUP BY 도시  
HAVING SUM(마일리지) >= 1000;
```

도시	SUM(마일리지)
서울특별시	1661
부산광역시	7329

(c) 도시 중 마일리지 합 ≥ 1000 의 결과

Section 02

집계 함수 심화

1. WITH ROLLUP

■ WITH ROLLUP

- 그룹별 소계와 전체 총계를 한번에 확인하고 싶을 때 사용함
- GROUP BY절 다음에 WITH ROLLUP을 사용하면 그룹별 소계와 전체 합계를 같이 나타낼 수 있음
- 형식

```
SELECT 그룹으로_묶을_컬럼명  
    ,집계_함수  
FROM 테이블명  
GROUP BY 그룹으로_묶을_컬럼명  
WITH ROLLUP;
```



그림 4-4 WITH ROLLUP의 역할

1. WITH ROLLUP

- [예제 4-8] 지역이 NULL인 고객에 대해 도시별로 고객수와 평균마일리지를 보이시오. 이때 맨 마지막 행에 전체 고객수와 전체 고객에 대한 평균마일리지도 함께 볼 수 있도록 작성하시오.

```
SELECT 도시
      ,COUNT(*) AS 고객수
      ,AVG(마일리지) AS 평균마일리지
  FROM 고객
 WHERE 지역 IS NULL
 GROUP BY 도시
    WITH ROLLUP;
```

▶ 실행 결과

도시	고객수	평균마일리지
대한민국역시	4	14822.0000
부산광역시	5	3985.2000
서울특별시	50	11137.0800
인천광역시	7	13088.0000
전국	66	11025.5152

실행 결과가 나오지 않는다면 2장의 [예제 2-13]과 같이 고객 테이블의 지역 컬럼에 값이 들어가 있지 않은 빈 문자열을 NULL로 변경 했는지 확인하세요.

1. WITH ROLLUP

■ [예제 4-8] 지역이 NULL인 고객에 대해 도시별로 고객수와 평균마일리지를 보이시오. 이때 맨 마지막 행에 전체 고객수와 전체 고객에 대한 평균마일리지도 함께 볼 수 있도록 작성하시오.

- SELECT절에 IFNULL()을 추가하여 [예제 4-8]을 보완한 SQL문

```
SELECT IFNULL(도시,'총계') AS 도시  
    ,COUNT(*) AS 고객수  
    ,AVG(마일리지) AS 평균마일리지  
FROM 고객  
WHERE 지역 IS NULL  
GROUP BY 도시  
WITH ROLLUP;
```

▶ 실행결과

도시	고객수	평균마일리지
부산광역시	5	3985.2000
서울특별시	50	11137.0800
인천광역시	7	13088.0000
총계	66	11025.5152

1. WITH ROLLUP

- [예제 4-9] 담당자직위에 '마케팅'이 들어가 있는 고객에 대해 고객(담당자직위, 도시)별 고객수를 보이시오. 담당자직위별 고객수와 전체 고객수도 함께 볼 수 있도록 조회하시오.

```
SELECT 담당자직위
      ,도시
      ,COUNT(*) AS 고객수
  FROM 고객
 WHERE 담당자직위 LIKE '%마케팅%'
 GROUP BY 담당자직위
      ,도시
  WITH ROLLUP;
```

▶ 실행결과

담당자직위	도시	고객수
마케팅 과장	구리시	1
마케팅 담당	나주시	1
마케팅 담당	서울특별시	3
마케팅 담당	대전	4
	전체	16

A

B

1. WITH ROLLUP

- [예제 4-9] 담당자직위에 '마케팅'이 들어가 있는 고객에 대해 고객(담당자직위, 도시)별 고객수를 보이시오. 담당자직위별 고객수와 전체 고객수도 함께 볼 수 있도록 조회하시오.



그림 4-5 다중 컬럼이 사용된 WITH ROLLUP

1. WITH ROLLUP

■ GROUPING()

- WITH ROLLUP의 결과로 나온 NULL에 대해서는 1을 반환하고, 그렇지 않은 NULL에 대해서는 0을 반환함

■ [예제 4-10] 담당자직위가 '대표 이사'인 고객에 대하여 지역별로 묶어서 고객수를 보이고, 전체 고객수도 함께 보이시오.

```
SELECT 지역  
      ,COUNT(*) AS 고객수  
  FROM 고객  
 WHERE 담당자직위 = '대표 이사'  
 GROUP BY 지역  
 WITH ROLLUP;
```

▶ 실행 결과

지역	고객수
NULL	14
경기도	4
제주도	1
NULL	19

A : 제3행 NULL
B : 제5행 NULL

- 의미가 서로 다른 두 NULL을 어떻게 구별할 수 있을까?

1. WITH ROLLUP

■ [예제 4-10] 담당자직위가 '대표 이사'인 고객에 대하여 지역별로 묶어서 고객수를 보이고, 전체 고객수도 함께 보이시오.

- 의미가 서로 다른 두 NULL을 어떻게 구별할 수 있을까?
 - ✓ GROUPING()을 사용함

```
SELECT 지역
      ,COUNT(*) AS 고객수
      ,GROUPING(지역) AS 구분
  FROM 고객
 WHERE 담당자직위 = '대표 이사'
 GROUP BY 지역
    WITH ROLLUP;
```

▶ 실행 결과

지역	고객수	구분
서울	14	0
경기도	4	0
제주도	1	0
전체	19	1

2. GROUP_CONCAT()

■ GROUP_CONCAT()

- 각 행에 있는 값을 결합함

■ [예제 4-11] GROUP_CONCAT()을 사용하여 사원 테이블에 들어있는 이름을 한 행에 나열하시오.

```
SELECT GROUP_CONCAT(이름)
FROM 사원;
```

▶ 실행결과

GROUP_CONCAT(이름)
이소미, 배재율, 유대현, 최소민, 안주홍, 이현진,...

■ [예제 4-12] 고객 테이블에 들어있는 지역을 한 행에 나열하되 중복되는 지역은 한 번씩만 보이시오.

```
SELECT GROUP_CONCAT(DISTINCT 지역)
FROM 고객;
```

▶ 실행결과

GROUP_CONCAT(DISTINCT 지역)
, 강원도, 경기도, 경상남도, 경상북도, 전라남도,...

2. GROUP_CONCAT()

■ [예제 4-13] 고객 테이블에서 도시별로 고객회사명을 나열하시오.

```
SELECT 도시  
      ,GROUP_CONCAT(고객회사명) AS 고객회사명목록  
FROM 고객  
GROUP BY 도시;
```

▶ 실행결과

도시	고객회사명목록
공주시	오키이식품
과천시	대원에스엠,진영무역
광명시	은혜식품,씨엔그룹
구리시	남해종합식품
고양시	케이리필리스

점검문제

점검문제

문제 1

고객 테이블의 도시 컬럼에는 몇 개의 도시가 들어있을까요? 도시 수와 중복 값을 제외한 도시 수를 보이시오.

▶ 실행결과

COUNT(도시)	COUNT(DISTINCT 도시)
93	27

문제 2

제품 테이블에서 주문년도별로 주문건수를 조회하시오.

▶ 실행결과

주문년도	주문건수
2020	270
2021	529
2022	31

점검문제

문제 3

결과 화면을 참고하여 주문 테이블에서 (주문년도, 분기)별 주문건수, 주문년도별

▶ 실행결과

주문건수, 전체 주문건수를 한번에 조회해봅시다.

(주문년도, 분기)별 주문건수

주문년도별 주문건수

전체 주문건수

주문년도	분기	주문건수
2020	1	16
2020	2	72
2020	3	87
2020	4	95
2020	NULL	270
2021	1	91
2021	2	106
2021	3	133
2021	4	199
2021	NULL	529
2022	1	31
2022	NULL	31
2022	NULL	830

점검문제

문제 4

주문 테이블에서 요청일보다 발송이 늦어진 주문내역이 월별로 몇 건씩인지
요약하여 조회하시오. 이때 주문월 순서대로 정렬하여 보이시오.

▶ 실행결과

주문월	주문건수
1	3
3	3
4	3
5	4
6	4
7	3
8	3
9	5
10	4
11	4
12	2



점검문제

문제 5

제품 테이블에서 '아이스크림' 제품들에 대하여 제품명별로 재고합을 보이시오.

▶ 실행 결과

제품명	재고합
볼루 바닐라 아이스크림	0
볼루 초콜릿 아이스크림	9
이倔 얼른 아이스크림	112

문제 6

고객 테이블에서 마일리지가 50,000점 이상인 고객은 'VIP고객', 나머지 고객은 '일반고객'으로 구분하고, 고객구분별로 고객수와 평균마일리를 보이시오.

▶ 실행 결과

고객구분	고객수	평균마일리지
일반고객	88	3853.5568
VIP고객	5	92582.8000

Chapter 05

조인



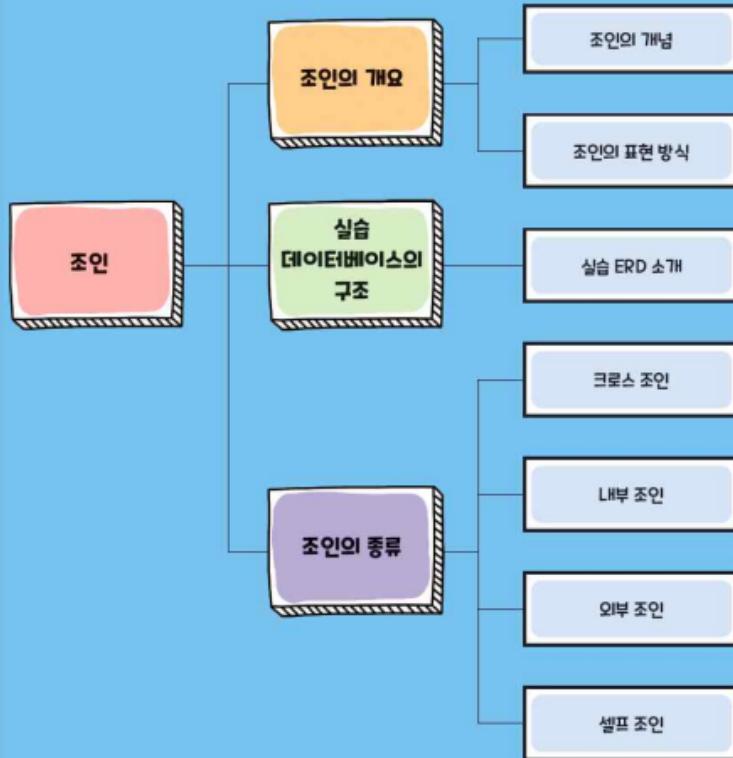
목차

1. 조인의 개요
2. 실습 데이터베이스의 구조
3. 조인의 종류

학습목표

- 조인의 개념과 표현 방식을 이해할 수 있습니다.
- 조인의 종류를 이해하고 활용할 수 있습니다.

Preview



Section 01

조인의 개요

1. 조인의 개념

■ 조인(Join)

- 두 개 이상의 테이블을 연결하여 데이터를 검색하는 방법
- 서로 다른 테이블에 저장된 관련된 데이터를 함께 가져와 하나의 결과로 표시함
- 검색하고 싶은 컬럼이 서로 다른 테이블에 있을 때 사용함
- 조인을 사용하면 여러 개의 테이블을 마치 하나의 테이블인 것처럼 쓸 수 있음



그림 5-1 조인을 하는 이유

2. 조인의 표현 방식

■ ANSI SQL 조인 방식

- FROM절에 있는 두 테이블명 사이에 조인 종류에 따라 CROSS, INNER, OUTER와 함께 JOIN 키워드를 넣어줌
- 조인에 대한 조건은 ON절에 작성하고 나머지 조건은 WHERE절에 작성함
- CROSS, INNER, OUTER 키워드는 생략할 수 있음
- 형식

```
SELECT *
FROM 테이블A
[CROSS|INNER|OUTER] JOIN 테이블B
ON 조인_조건
WHERE 기타_조건;
```

2. 조인의 표현 방식

■ Non-ANSI SQL 조인 방식

- 테이블을 쉼표(,)로 구분하여 FROM절에 작성함
- 조인 조건과 기타 조건을 구분하지 않고 모두 WHERE절에 기술함
- 형식

```
SELECT *
FROM 테이블A
    ,테이블B
WHERE 조인_조건 및 기타_조건;
```

Section 02

실습 데이터베이스의 구조

1. 실습 ERD 소개

■ ERD(Entity Relationship Diagram)

- 테이블 간의 관계를 설명해주는 다이어그램
 - ERD를 통해 데이터베이스의 구조를 한눈에 파악할 수 있음
 - ERD를 통해 서로 관계가 있는 테이블들을 파악할 수 있음
- ✓ 그들 간의 대응수가 1:1인지 또는 1:N인지 등을 확인할 수 있음



그림 5-2 고객 테이블과 주문 테이블의 관계

1. 실습 ERD 소개

■ ERD

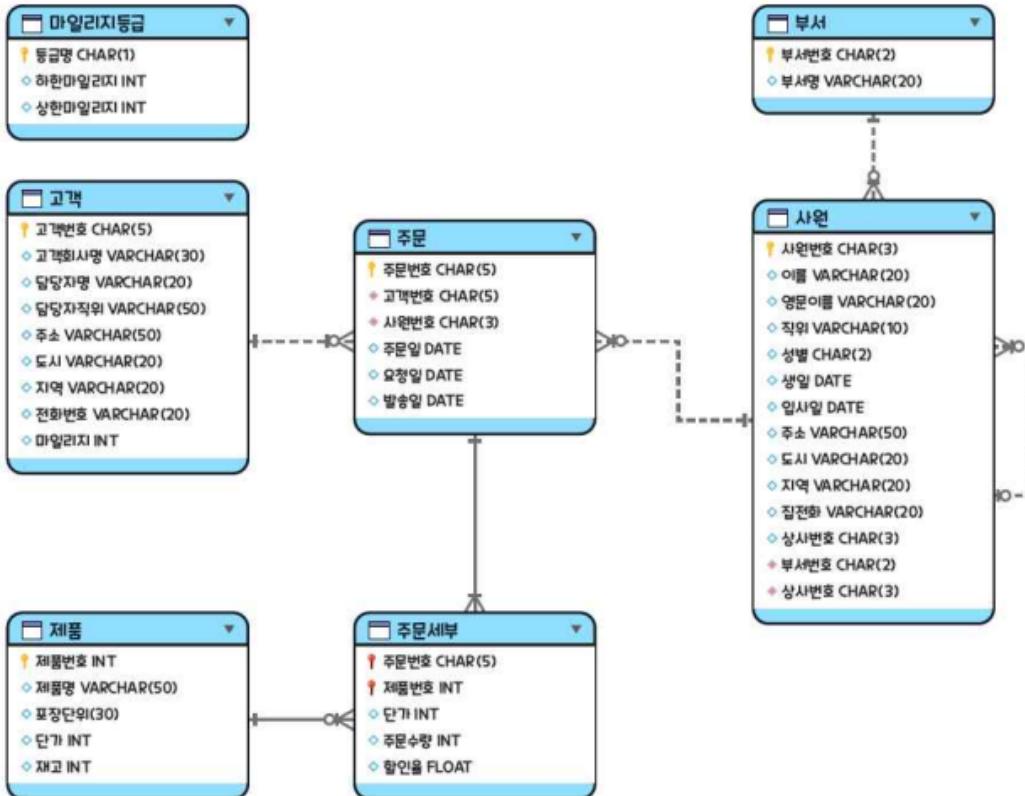


그림 5-3 한빛무역 ERD

1. 실습 ERD 소개



주문세부 테이블에는 기본키가 두 개 있는 건가요?

주문세부	
■	주문번호 CHAR(5)
■	제품번호 INT
△	단가 INT
△	주문수량 INT
△	합인율 FLOAT

모든 테이블은 오직 한 개의 기본키만 가질 수 있습니다.

이 테이블의 기본키는 '주문번호 + 제품번호'로, 동일한 '주문번호 + 제품번호'를 가진 레코드는 오직 한 개만 존재해야 함을 의미합니다. 이와 같이 두 개 이상의 컬럼으로 이루어진 기본키를 복합키 (Composite Key)라고 합니다.



Section 03

조인의 종류

1. 크로스 조인

■ 조인의 종류

- 크로스 조인, 내부 조인, 외부 조인, 셀프 조인

■ 크로스 조인(Cross Join)

- 한쪽 테이블의 각 행마다 다른 쪽 테이블의 모든 행이 한 번씩 각각 매칭되는 조인을 의미하며, 카티션 곱(Cartesian Product)이라고도 부름
- 크로스 조인의 결과 행의 개수 : 테이블A 행의 개수×테이블B 행의 개수

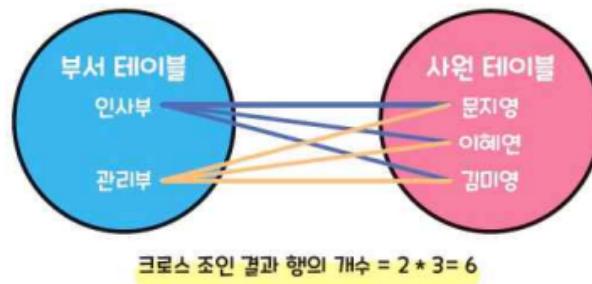


그림 5-4 크로스 조인(카티션 곱)의 형태

1. 크로스 조인

■ 크로스 조인의 문법

- ANSI SQL 조인 형식
 - ✓ CROSS는 생략할 수 있음

```
SELECT *
FROM 테이블A
[CROSS] JOIN 테이블B
WHERE 기타_조건;
```

- Non-ANSI SQL 조인

```
SELECT *
FROM 테이블A
,테이블B
WHERE 기타_조건;
```

1. 크로스 조인

- [예제 5-1] 사원 테이블과 부서 테이블을 크로스 조인하여 '배재용' 사원에 대한 정보(이름, 사원 테이블의 부서번호, 부서 테이블의 부서번호, 부서명)를 보이시오.

- ANSI SQL 조인

```
SELECT 부서.부서번호  
      ,부서명  
      ,이름  
      ,사원.부서번호  
FROM 부서  
CROSS JOIN 사원  
WHERE 이름 = '배재용';
```

▶ 실행 결과

부서번호	부서명	이름	부서번호
A1	영업부	배재용	A2
A2	기획부	배재용	A2
A3	개발부	배재용	A2
A4	홍보부	배재용	A2

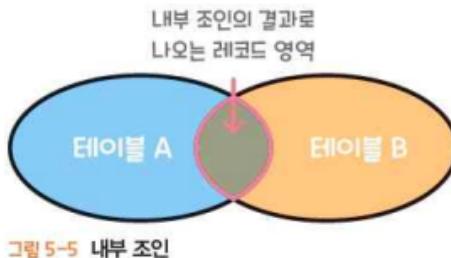
- Non-ANSI SQL 조인

```
SELECT 부서.부서번호 ①  
      ,부서명  
      ,이름  
      ,사원.부서번호  
FROM 부서  
      ,사원  
WHERE 이름 = '배재용';
```

2. 내부 조인

■ 내부 조인(Inner Join)

- 각 테이블에서 조인 조건에 일치되는 데이터만 가져오는 조인
- 내부 조인에는 이쿼 조인과 비이쿼 조인이 있음
- 이쿼 조인(Equi Join)
 - ✓ 조인 조건에 = 연산자를 사용함
- 비이쿼 조인(Non-Equi Join)
 - ✓ 조인 조건에 = 연산자 이외의 비교 연산자를 사용함
- 여러 테이블을 사용할 때 조인 조건을 제대로 기술하지 않으면 카티션 곱(크로스 조인)을 한 결과가 나오게 됨



2. 내부 조인

■ 내부 조인의 문법

- ANSI SQL 조인

```
SELECT *
FROM 테이블A
[INNER] JOIN 테이블B
ON 조인_조건
WHERE 기타_조건;
```

- Non-ANSI SQL 조인

```
SELECT *
FROM 테이블A
,테이블B
WHERE 조인_조건 및 기타_조건;
```

2. 내부 조인

■ [예제 5-2] '이소미' 사원의 사원번호, 직위, 부서번호, 부서명을 보이시오.

- 사원 테이블과 부서 테이블의 ERD

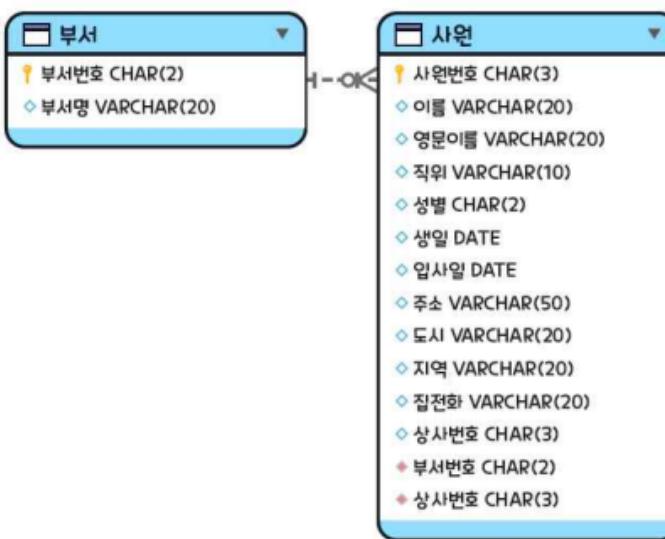


그림 5-6 사원 테이블과 부서 테이블의 ERD

2. 내부 조인

■ [예제 5-2] '이소미' 사원의 사원번호, 직위, 부서번호, 부서명을 보이시오.

- ANSI SQL 조인

```
SELECT 사원번호  
      ,직위  
      ,사원.부서번호  
      ,부서명  
FROM 사원  
INNER JOIN 부서  
ON 사원.부서번호 = 부서.부서번호 •———— 조인 조건  
WHERE 이름 = '이소미'; •———— 기타 조건
```

▶ 실행 결과

사원번호	직위	부서번호	부서명
E01	사원	A1	영업부

- Non-ANSI SQL 조인

```
SELECT 사원번호  
      ,직위  
      ,사원.부서번호  
      ,부서명  
FROM 사원  
      ,부서  
WHERE 사원.부서번호 = 부서.부서번호  
      AND 이름 = '이소미';
```

조인 조건과 기타 조건

2. 내부 조인

■ [예제 5-3] 고객 회사들이 주문한 주문건수를 주문건수가 많은 순서대로 보이시오. 이때 고객 회사의 정보로는 고객번호, 담당자명, 고객회사명을 보이시오.

- 고객 테이블과 주문 테이블의 ERD



그림 5-7 고객 테이블과 주문 테이블의 ERD

2. 내부 조인

- [예제 5-3] 고객 회사들이 주문한 주문건수를 주문건수가 많은 순서대로 보이시오. 이때 고객 회사의 정보로는 고객번호, 담당자명, 고객회사명을 보이시오.

- ANSI SQL 조인

```
SELECT 고객.고객번호
      ,담당자명
      ,고객회사명
      ,COUNT(*) AS 주문건수
  FROM 고객
 INNER JOIN 주문
    ON 고객.고객번호 = 주문.고객번호
 GROUP BY 고객.고객번호
      ,담당자명
      ,고객회사명
 ORDER BY COUNT(*) DESC;
```

2. 내부 조인

■ [예제 5-3] 고객 회사들이 주문한 주문건수를 주문건수가 많은 순서대로 보이시오. 이때 고객 회사의 정보로는 고객번호, 담당자명, 고객회사명을 보이시오.

- Non-ANSI SQL 조인

```
SELECT 고객.고객번호
      ,담당자명
      ,고객회사명
      ,COUNT(*) AS 주문건수
  FROM 고객
    ,주문
 WHERE 고객.고객번호 = 주문.고객번호
 GROUP BY 고객.고객번호
      ,담당자명
      ,고객회사명
 ORDER BY COUNT(*) DESC;
```

▶ 실행결과

고객번호	담당자명	고객회사명	주문건수
VEASA	장시현	크림푸드	31
NSHER	정다경	슬인티내셔널	30
ICKQU	정대왕	인터넷	28
NGOHU	오유진	케이티에스씨	19
LKOFO	조희진	진영무역	19
LAAHI	신민주	성민식품상사	18
TTCRA	술지윤	리보인트	18
RGSBE	박건희	탈루드코리아	18
NAPOB	신풀수	만성 푸에	17
RTHWHA	박규민	국월식품	15
ANKFR	최지수	오리안무역	15
HMSLE	조효주	대화유리피스	15
NARHA	손창민	청운유통	14

2. 내부 조인

■ [예제 5-4] 고객별(고객번호, 담당자명, 고객회사명)로 주문금액 합을 보이되, 주문금액 합이 많은 순서대로 보이시오.

- 고객 테이블, 주문 테이블, 주문세부 테이블의 ERD



그림 5-8 고객 테이블, 주문 테이블, 주문세부 테이블의 ERD

2. 내부 조인

■ [예제 5-4] 고객별(고객번호, 담당자명, 고객회사명)로 주문금액 합을 보이되, 주문금액 합이 많은 순서대로 보이시오.

- ANSI SQL 조인

```
SELECT 고객.고객번호
      ,담당자명
      ,고객회사명
      ,SUM(주문수량 * 단가) AS 주문금액합
   FROM 고객
INNER JOIN 주문
  ON 고객.고객번호 = 주문.고객번호
INNER JOIN 주문세부
  ON 주문.주문번호 = 주문세부.주문번호
 GROUP BY 고객.고객번호
      ,담당자명
      ,고객회사명
 ORDER BY 4 DESC;
```

2. 내부 조인

■ [예제 5-4] 고객별(고객번호, 담당자명, 고객회사명)로 주문금액 합을 보이되, 주문금액 합이 많은 순서대로 보이시오.

- Non-ANSI SQL 조인

```
SELECT 고객.고객번호
      ,담당자명
      ,고객회사명
      ,SUM(주문수량 * 단가) AS 주문금액합
   FROM 고객
        ,주문
        ,주문세부
 WHERE 고객.고객번호 = 주문.고객번호
   AND 주문.주문번호 = 주문세부.주문번호
 GROUP BY 고객.고객번호
        ,담당자명
        ,고객회사명
 ORDER BY 4 DESC;
```

▶ 실행 결과

고객번호	담당자명	고객회사명	주문금액합
ICKQU	정대웅	인터넷 서비스	11742000
VEASA	장시현	코월루드	11611600
NSPER	정다경	솔인티나세일	11356100
NGOHU	오유진	케이티에스씨	5739300
TTORA	송지운	허브인트	5086600
NARHA	손창민	청운유통	3418800
...			

2. 내부 조인

- [예제 5-5] 고객 테이블과 마일리지등급 테이블을 크로스 조인하시오. 그 다음 고객 테이블에서 담당자가 '이은광'인 고객에 대하여 고객번호, 담당자명, 마일리지와 마일리지등급 테이블의 모든 컬럼을 보이시오.

- ANSI SQL 조인

```
SELECT 고객번호  
      ,담당자명  
      ,마일리지  
      ,등급.*  
FROM 고객  
CROSS JOIN 마일리지등급 AS 등급  
WHERE 담당자명 = '이은광';
```

▶ 실행결과

고객번호	담당자명	마일리지	등급명	하한마일리지	상한마일리지
ACDDR	이은광	15911	S	100000	999999999
ACDDR	이은광	15911	D	0	99
ACDDR	이은광	15911	C	100	999
ACDDR	이은광	15911	B	1000	9999
ACDDR	이은광	15911	A	10000	99999

- Non-ANSI SQL 조인

```
SELECT 고객번호  
      ,담당자명  
      ,마일리지  
      ,등급.*  
FROM 고객  
      ,마일리지등급 AS 등급  
WHERE 담당자명 = '이은광';
```

2. 내부 조인

■ [예제 5-6] 고객 테이블에서 담당자가 '이은광'인 경우의 고객번호, 고객회사명, 담당자명, 마일리지와 마일리지등급을 보이시오.

- 고객 테이블과 마일리지등급 테이블의 ERD



그림 5-9 고객 테이블과 마일리지등급 테이블의 ERD

2. 내부 조인

■ [예제 5-6] 고객 테이블에서 담당자가 '이은광'인 경우의 고객번호, 고객회사명, 담당자명, 마일리지와 마일리지등급을 보이시오.

- ANSI SQL 조인

```
SELECT 고객번호
      ,고객회사명
      ,담당자명
      ,마일리지
      ,등급명
  FROM 고객
 INNER JOIN 마일리지등급
    ON 마일리지 >= 하한마일리지
   AND 마일리지 <= 상한마일리지
 WHERE 담당자명 = '이은광';
```

✓ ON절을 다음과 같이 작성할 수 있음

```
FROM 고객
INNER JOIN 마일리지등급
ON 마일리지 BETWEEN 하한마일리지 AND 상한마일리지
```

2. 내부 조인

■ [예제 5-6] 고객 테이블에서 담당자가 '이은광'인 경우의 고객번호, 고객회사명, 담당자명, 마일리지와 마일리지등급을 보이시오.

- Non-ANSI SQL 조인

```
SELECT 고객번호
      ,고객회사명
      ,담당자명
      ,마일리지
      ,등급명
FROM 고객
     ,마일리지등급
WHERE 마일리지 BETWEEN 하한마일리지 AND 상한마일리지
  AND 담당자명 = '이은광';
```

▶ 실행결과

고객번호	고객회사명	담당자명	마일리지	등급명
ACD001	굿모닝서울	이은광	15911	A

3. 외부 조인

■ 외부 조인(Outer Join)

- 외부 조인을 통해 조건에 맞지 않는 행도 함께 출력할 수 있음
- 외부 조인은 두 테이블에서 한쪽에는 데이터가 있고 한쪽에는 데이터가 없는 경우 데이터가 있는 쪽의 테이블을 기준으로 데이터를 출력함
- MySQL에서 외부 조인은 ANSI SQL 방식으로만 표현할 수 있음

3. 외부 조인

■ LEFT/RIGHT 외부 조인

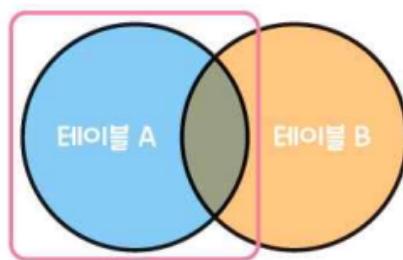
- 외부 조인은 조인할 테이블명 사이에 LEFT 또는 RIGHT와 함께 OUTER JOIN을 넣음
- OUTER는 생략 가능함
- LEFT JOIN
 - ✓ 왼쪽에 있는 테이블의 결과를 기준으로 오른쪽 테이블의 데이터를 매칭함
 - ✓ 매칭되는 데이터가 없는 경우에는 NULL로 표시됨
- RIGHT JOIN
 - ✓ 오른쪽에 있는 테이블 결과를 기준으로 왼쪽 테이블의 데이터를 매칭함
 - ✓ 매칭되는 데이터가 없는 경우에는 NULL로 표시됨

3. 외부 조인

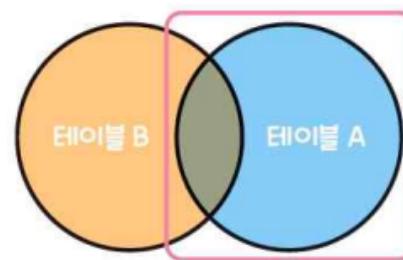
■ LEFT/RIGHT 외부 조인

- 형식

```
SELECT *
FROM 테이블A
LEFT|RIGHT [OUTER] JOIN 테이블B
ON 조인_조건
WHERE 기타_조건;
```



(a) LEFT 조인



(b) RIGHT 조인

그림 5-10 매칭되는 것과 안 되는 행을 다 가져오는 LEFT/RIGHT 외부 조인

3. 외부 조인

하나 더 알기 ✓

FULL OUTER JOIN

FULL OUTER JOIN은 LEFT OUTER JOIN과 RIGHT OUTER JOIN을 합한 형태입니다. MySQL에서는 이것을 지원하지 않습니다. 대신 UNION 집합 연산자를 사용하여 동일한 결과를 얻을 수 있습니다.

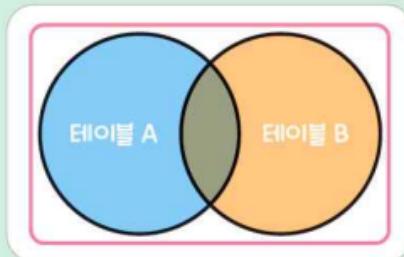


그림 5-11 FULL OUTER JOIN

형식

```
SELECT *  
FROM 테이블A  
LEFT [OUTER] JOIN 테이블B  
ON 조인_조건  
WHERE 기타_조건  
UNION  
SELECT *  
FROM 테이블A  
RIGHT [OUTER] JOIN 테이블B  
ON 조인_조건  
WHERE 기타_조건;
```



3. 외부 조인

■ [예제 5-7] 고객 테이블에서 담당자가 '이은광'인 경우의 고객번호, 고객회사명, 담당자명, 마일리지와 마일리지등급을 보이시오.

- 이 예제는 이퀴 조인을 하면 부서번호가 NULL인 '여' 사원의 정보는 얻을 수 없음
 - ✓ 예를 들어 '정수진' 사원은 부서번호가 없기 때문에 이퀴 조인을 하면 '정수진' 사원의 레코드는 결과에 나오지 않음

사원번호	이름	영문이름	직위	성별	생일	입사일	주소	도시	지역	집전화	실사번호	부서번호
E01	정수진	Jung Su Jin	수습사원 여	1993-10-07	2022-03-19	동작구 흥복동 현대아파트 1-1501	서울특별시	경인	(02)824-2898			

(a) 사원 테이블에서 부서번호가 없는 정수진 사원의 정보

사원번호	이름	부서명
E01	이소미	영업부
E04	최소민	영업부
E08	선하라	개발부
E09	유가율	영업부

(b) 이퀴 조인 결과

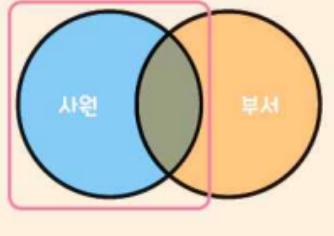
그림 5-12 사원 테이블과 부서 테이블에 대한 이퀴 조인 결과

3. 외부 조인

■ [예제 5-7] 고객 테이블에서 담당자가 '이은광'인 경우의 고객번호, 고객회사명, 담당자명, 마일리지와 마일리지등급을 보이시오.

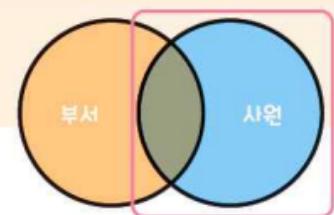
- 외부 조인을 사용해야 부서번호가 없는 '정수진' 사원의 정보도 함께 얻을 수 있음

```
SELECT 사원번호  
      ,이름  
      ,부서명  
FROM 사원  
LEFT OUTER JOIN 부서  
ON 사원.부서번호 = 부서.부서번호  
WHERE 성별 = '여';
```



✓ 사원 테이블명이 우측에 기술되어 있을 때는 RIGHT를 넣으면 됨

```
FROM 부서  
RIGHT OUTER JOIN 사원  
ON 사원.부서번호 = 부서.부서번호
```



▶ 실행 결과

사원번호	이름	부서명
E01	이소미	영업부
E04	최소인	영업부
E08	선하라	개발부
E09	유가을	영업부
E10	정수진	

3. 외부 조인

■ [예제 5-8] 부서명과 해당 부서의 소속 사원 정보를 보이시오. 이때 사원이 한 명도 존재하지 않는 부서명이 있다면 그 부서명도 함께 보이시오.

- 이퀴 조인을 한다면 'A4' 부서에 대한 정보는 확인할 수 없음
 - ✓ 부서 테이블에는 'A4' 부서의 레코드가 존재하는데,
사원 테이블에는 'A4' 부서에 소속된 사원이 존재하지 않기 때문

부서번호	부서명
E01	영업부
E02	기획부
E03	개발부
E04	홍보부

사원번호	이름	부서번호
E01	이소미	A1
E02	배재용	A2
E03	유대현	A1
E04	최소민	A1
E05	안주홍	A1
E06	이현진	A1
E07	오영수	A1
E08	선하라	A3
E09	유가을	A1
E10	정수진	

(a) 부서 테이블

(b) 사원 테이블

그림 5-13 부서 테이블과 사원 테이블

3. 외부 조인

■ [예제 5-8] 부서명과 해당 부서의 소속 사원 정보를 보이시오. 이때 사원이 한 명도 존재하지 않는 부서명이 있다면 그 부서명도 함께 보이시오.

- 'A4' 부서의 정보를 함께 확인하려면 외부 조인을 사용해야 함
 - ✓ 테이블을 기준으로 외부 조인을 작성함

```
SELECT 부서명
      ,사원.*
  FROM 사원
RIGHT OUTER JOIN 부서
    ON 사원.부서번호 = 부서.부서번호;
```

▶ 실행 결과

부서명	사원번호	이름	영문이름	직위	성별	생일	입사일	주소	도시	지역	집전화	상사번호	부서번호
영업부	E05	유가을	Yoo Ga Eul	부장	여	1990-0...	2019-03-29	남구 대평동 19-7	대구광...	영남	(053)465-1248	E02	A1
영업부	E07	오경수	O Young Soo	서원	남	1989-0...	2019-12-15	동화동 76-77	경주시	종부	(051)219-1284	E06	A1
영업부	E08	이현진	Lee Hyun Jin	대리	남	1983-0...	2019-09-26	대구구 이현동 577-1	대전광...	충무부	(042)218-3876	E05	A1
영업부	E03	안주홍	Ahn Ju Hong	과장	남	1980-0...	2010-12-25	남구 토곡동 270-31	인천광...	강진	(032)915-9278	E09	A1
영업부	E04	최소민	Choi So Min	사원	여	1987-0...	2019-04-15	중구 을당동 57-14	부산광...	영남	(051)587-4793	E06	A1
	E03	손아영	Son Ae Young	사원	여			울산구 동정동 100-14	울산광...	동부	(052)573-0294		
기획부	E02	백자룡	Bae Jea Yong	책임자	남	1973-0...	2019-01-01	한미동 10-11	부경시	경인	(032)999-01249		A2
기획부	E08	신희란	Sohn Hee Ran	회사동장	여	1988-01-	2018-12-31	목동구 고йт로 116	전주시	호서	(062)568-15885	E02	A1
홍보부	G05											G05	

3. 외부 조인

■ [예제 5-9] 사원이 한 명도 존재하지 않는 부서명을 보이시오.

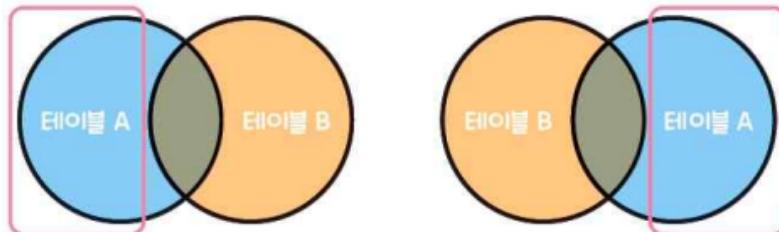


그림 5-14 LEFT/RIGHT 외부 조인에서 매칭이 안 되는 행만 선택

```
SELECT 부서명
      ,사원.* •———— 설명을 위해 사원 테이블의 정보도 함께 나타냄
  FROM 사원
RIGHT OUTER JOIN 부서
  ON 사원.부서번호 = 부서.부서번호
 WHERE 사원.부서번호 IS NULL;
```

▶ 실행결과

부서명	사원번호	이름	영문이름	직위	성별	생일	입사일	주소	도시	지역	집전화	상사번호	부서번호
홍보부	1001	홍길동	Hong Gil Dong	부장	남	1985-01-01	2005-01-01	서울특별시 강남구 테헤란로 123	서울	서울	02-1234-5678	1001	

3. 외부 조인

■ [예제 5-10] 소속 부서가 없는 사원의 이름을 보이시오.

```
SELECT 이름  
      ,부서.* ●———— 설명을 위해 부서 테이블의 정보도 함께 나타냄  
FROM 사원  
LEFT OUTER JOIN 부서  
ON 사원.부서번호 = 부서.부서번호  
WHERE 부서.부서번호 IS NULL;
```

▶ 실행 결과

이름	부서번호	부서명
정수진	NULL	NULL

4. 셀프 조인

■ 셀프 조인(Self Join)

- 동일한 테이블 내에서 한 컬럼이 다른 컬럼을 참조하는 조인
- 셀프 조인을 하려면 조인 조건에 동일한 테이블명이 두 번 나타나게 됨
- 이때 테이블명을 다른 별명으로 지정하여 다른 테이블인 것처럼 사용함
- 또한 컬럼명도 모두 동일하므로 테이블의 별명과 컬럼명을 함께 써 컬럼의 소속을 구분해주어야 함

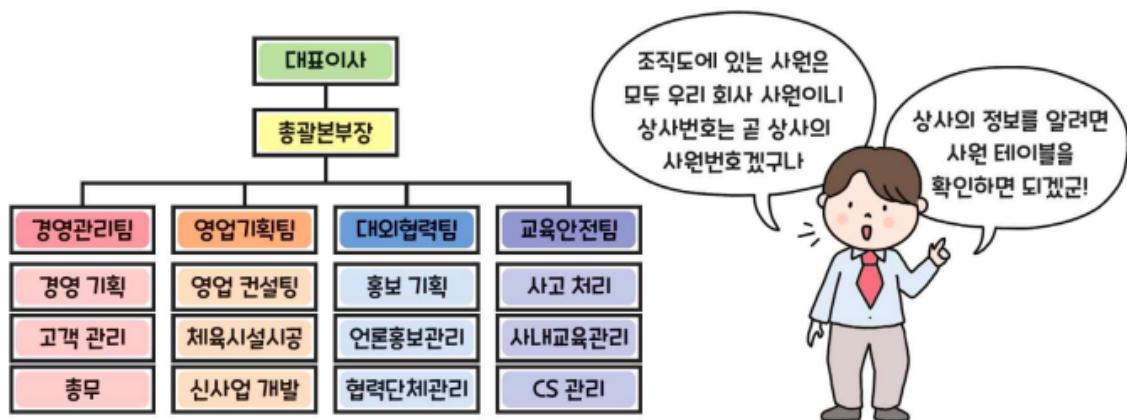


그림 5-15 셀프 조인

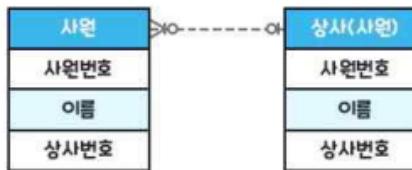
4. 셀프 조인

■ [예제 5-11] 사원번호, 사원의 이름, 상사의 사원번호, 상사의 이름을 보이시오.



(a) 사원 테이블 ERD

그림 5-16 사원 테이블의 ERD



(b) 개념상 분리한 사원 테이블과 상사(사원) 테이블

4. 셀프 조인

■ [예제 5-11] 사원번호, 사원의 이름, 상사의 사원번호, 상사의 이름을 보이시오.

- ANSI SQL 조인

```
SELECT 사원.사원번호  
      ,사원.이름  
      ,상사.사원번호 AS '상사의 사원번호'  
      ,상사.이름 AS '상사의 이름'  
FROM 사원  
INNER JOIN 사원 AS 상사  
ON 사원.상사번호 = 상사.사원번호;
```

- Non-ANSI SQL 조인

```
SELECT 사원.사원번호  
      ,사원.이름  
      ,상사.사원번호 AS '상사의 사원번호'  
      ,상사.이름 AS '상사의 이름'  
FROM 사원  
      ,사원 AS 상사  
WHERE 사원.상사번호 = 상사.사원번호;
```

▶ 실행 결과

사원번호	이름	상사의 사원번호	상사의 이름
E01	이소미	E06	이현진
E03	유대현	E06	이현진
E04	최소민	E06	이현진
E05	안주훈	E06	유가율

4. 셀프 조인

■ [예제 5-12] 사원이름, 직위, 상사이름을 상사이름 순으로 정렬하여 나타내시오. 이때 상사가 없는 사원의 이름도 함께 보이시오.

- 셀프 조인과 외부 조인을 함께 사용하여 해결

- 셀프 조인으로 상사와 사원의 정보를 확인하고, 외부 조인으로 상사가 없는 사원의 정보를 확인

- ANSI SQL 조인

```
SELECT 사원.이름 AS 이름
      ,사원.직위
      ,상사.이름 AS 상사이름
  FROM 사원 AS 상사
 RIGHT OUTER JOIN 사원
    ON 사원.상사번호 = 상사.사원번호
   ORDER BY 상사이름;
```

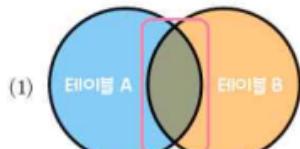
▶ 실행결과

이름	직위	상사이름
배재웅	대표이사	NULL
정수진	수습사원	NULL
선하라	전산팀장	배재웅
유가을	부처	배재웅
이원진	대리	안주론

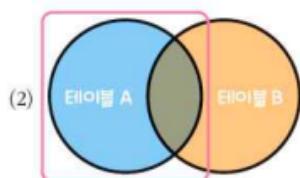
4. 셀프 조인

확인문제

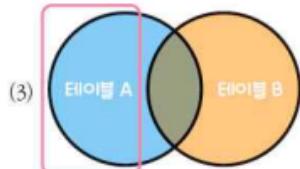
다음 그림에서 붉은 박스에 해당하는 레코드를 검색하고자 합니다. 어떤 SQL문으로 작성해야 하는지 연결시키시오.



- (A)
SELECT *
FROM 테이블A
INNER JOIN 테이블B
ON 테이블A.컬럼 = 테이블B.컬럼;



- (B)
SELECT *
FROM 테이블A
LEFT OUTER JOIN 테이블B
ON 테이블A.컬럼 = 테이블B.컬럼
WHERE 테이블B.컬럼 IS NULL;



- (C)
SELECT *
FROM 테이블A
LEFT OUTER JOIN 테이블B
ON 테이블A.컬럼 = 테이블B.컬럼;

정답

1-A, 2-C, 3-B

점검문제

점검문제

문제 1

한빛무역 데이터베이스의 제품 테이블과 주문세부 테이블을 조인하여 제품명별로 주문수량합과 주문금액합을 보이시오.

〈ERD〉



▶ 실행 결과

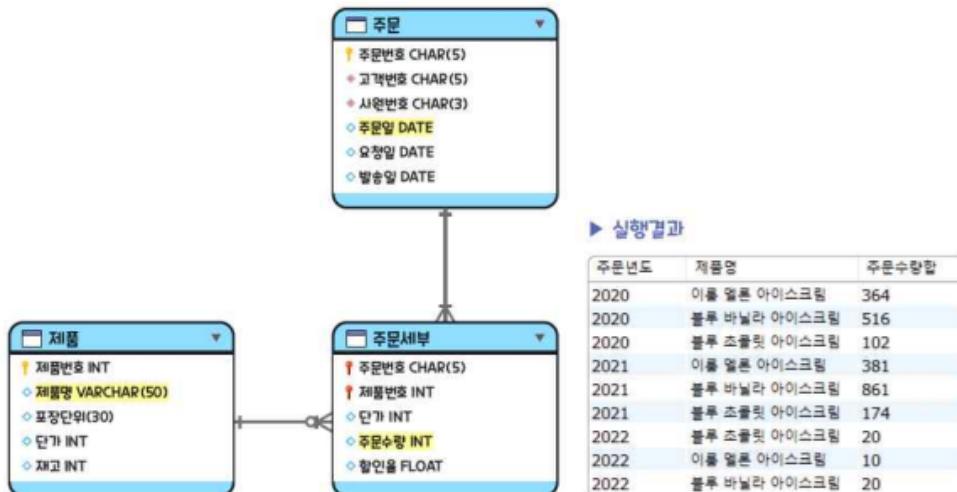
제품명	주문수량합	주문금액합
스마트 폰 A	1000	1000000
엔 100% 레몬 주스	1033	1802300
엔 체리 시럽	318	298000
베리 100% 블루바 시럽	452	945200
베리 100% 파인애플 시럽	298	574200
할시 블루베리 정	300	732000
할시 건과(베)	767	2243400

점검문제

문제 2

주문, 주문세부, 제품 테이블을 활용하여 '아이스크림' 제품에 대하여 (주문년도 제품명)별로 주문수량합을 보이시오.

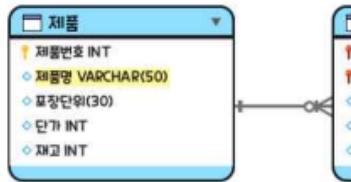
<ERD>



문제 3

제품, 주문세부 테이블을 활용하여 제품명별로 주문수량합을 보이시오. 이때 주문이 한 번도 안 된 제품에 대한 정보도 함께 나타내시오.

〈ERD〉



▶ 실행 결과

제품명	주문수량
한국 순상에일	291
서울 구이집	297
노르웨이 치즈	1151
찰스 콤플릿 드링크	981
노르웨이 살라드 도저히	792
밀푀유 달고 우유	6008

문제 4

고객 회사 중 마일리지 등급이 'A'인 고객의 정보를 조회하시오. 조회할 컬럼은 고객번호, 담당자명, 고객회사명, 등급명, 마일리지입니다.

〈ERD〉



▶ 실행결과

고객번호	고객회사명	담당자명	등급명	마일리지
ACDOR	굿도닝서울	이준광	A	15911
AUSBL	제원인터넷쇼날	박지혜	A	47865
EBITH	그린트더스	줄소현	A	81479
HMSLE	대화유파스	조희주	A	78001
LASLI	조종식률	지승관	A	22260
LMKWI	태광부드	조가영	A	36960
NAPBO	민성 무역	신종수	A	10006
RGSBE	탑투드코리아	박건희	A	57746

Chapter 06

서브쿼리



목차

1. 서브쿼리의 개요
2. 반환 값에 따른 서브쿼리
3. 사용 위치에 따른 서브쿼리
4. 상관 서브쿼리와 다중 컬럼 서브쿼리

학습목표

- 서브쿼리에 대해 설명할 수 있습니다.
- 서브쿼리의 종류를 이해하고, 작성할 수 있습니다.

Preview



Section 01

서브쿼리의 개요

1. 서브쿼리의 사용 목적

■ 서브쿼리(SubQuery)

- SQL문 내부에서 사용하는 SELECT문
- 서브쿼리는 복잡한 데이터 추출 및 조작 작업에 유용하며, SQL문의 유연성과 기능 확장에 도움됨
- 서브쿼리는 괄호 안에 기술해야 하며, WHERE절, SELECT절, FROM절, HAVING절 등에 넣어 사용할 수 있음
- 일반적으로 서브쿼리가 먼저 실행된 후 메인 쿼리가 실행되는 순서로 작업이 이루어짐

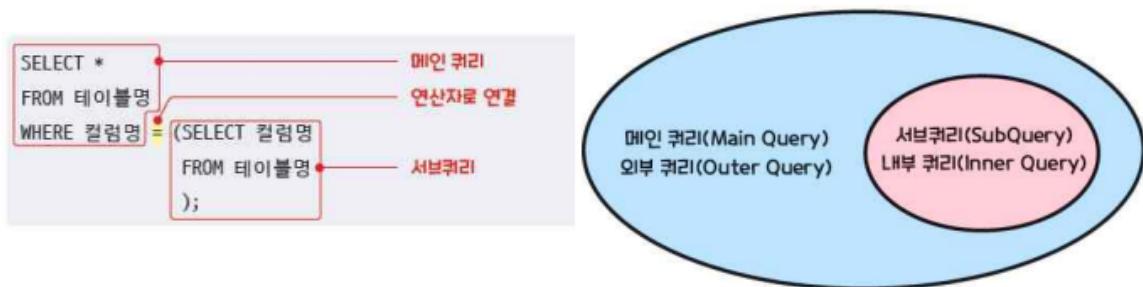


그림 6-1 서브쿼리의 구조

1. 서브쿼리의 사용 목적

■ 서브쿼리의 사용 목적

- 데이터 필터링
- 집계 및 계산
- 비교 및 검증
- 중첩된 데이터 추출
- EXISTS 절

2. 서브쿼리와 조인의 차이점

■ 서브쿼리와 조인의 차이점

- 데이터를 추출하는 접근 방식이 서로 다름
 - ✓ 서브쿼리 : 한 문장 안에 1개의 메인 쿼리와 1개 이상의 서브쿼리가 있는 구조
조인 : 1개의 쿼리문 안에 2개 이상의 테이블을 연결한 후 필요한 컬럼을 조회
- 서브쿼리
 - ✓ 작은 데이터셋이나 간단한 필터링 작업에 효율적
 - ✓ 간단한 쿼리에 적합
 - ✓ 데이터 필터링, 계산에 적합
- 조인
 - ✓ 큰 테이블이면서 적절한 인덱스가 있는 경우에 효율적
 - ✓ 복잡한 쿼리에 적합
 - ✓ 여러 테이블 간의 관계 처리에 적합

Section 02

반환 값에 따른 서브쿼리

1. 단일 행 서브쿼리

■ 단일 행 서브쿼리(Single-Row SubQuery)

- 서브쿼리의 결과로 단일 행을 반환함
- $=$, $<$, \leq , $>$, \geq , $<>$ 등의 단일 행 비교 연산자를 사용하여 메인 쿼리와 서브 쿼리를 연결할 수 있음

■ [예제 6-1] 최고 마일리지를 보유한 고객의 정보를 보이시오.

```
SELECT 고객번호
      ,고객회사명
      ,담당자명
      ,마일리지
  FROM 고객
 WHERE 마일리지 = (SELECT MAX(마일리지)
                      FROM 고객
                     );
```

▶ 실행 결과

고객 번호	고객회사명	담당자명	마일리지
ATRAN	대정 인터내셔널	배순용	128790

1. 단일 행 서브쿼리

■ [예제 6-2] 주문번호 'H0250'을 주문한 고객에 대해 고객회사명과 담당자명을 보이시오.

- 서브쿼리 사용

```
SELECT 고객회사명  
      ,담당자명  
FROM 고객  
WHERE 고객번호 = (SELECT 고객번호  
                      FROM 주문  
                     WHERE 주문번호 = 'H0250'  
                    );
```

- 조인 사용

```
SELECT 고객회사명  
      ,담당자명  
FROM 고객  
INNER JOIN 주문  
ON 고객.고객번호 = 주문.고객번호  
WHERE 주문번호 = 'H0250';
```

▶ 실행 결과

고객회사명	담당자명
청운유통	손장민

1. 단일 행 서브쿼리

- [예제 6-3] '부산광역시'고객의 최소 마일리지보다 더 큰 마일리지를 가진 고객 정보를 보이시오

```
SELECT 담당자명
      ,고객회사명
      ,마일리지
FROM 고객
WHERE 마일리지 > (SELECT MIN(마일리지)
                      FROM 고객
                     WHERE 도시 = '부산광역시'
                    );
```

▶ 실행 결과

담당자명	고객회사명	마일리지
이은경	굿모닝서울	15911
김성민	씨엔그룹	8788
최지수	오리안무역	6119

2. 복수 행 서브쿼리

복수 행 서브쿼리(Multi-Row SubQuery)

- 서브쿼리의 결과가 여러 행이 나오는 쿼리
- IN, ALL, ANY, SOME, EXISTS와 같은 복수 행 비교 연산자를 사용하여 메인 쿼리와 서브쿼리를 연결함

표 6-1 복수 행 비교 연산자

연산자	설명
IN	메인 쿼리의 비교 조건이 서브쿼리 결과 중 일치하는 것이 하나라도 있으면 참이다. 서브쿼리의 각 결괏값을 = 연산자로 비교한다.
ANY, SOME	서브쿼리의 각 결괏값을 >, <, >=, <= 등의 비교 연산자로 비교하여 하나 이상 일치하면 참이다. 즉, 서브쿼리의 최소 결괏값과 비교한다.
ALL	서브쿼리의 각 결괏값을 >, <, >=, <= 등의 비교 연산자로 비교하여 모두 일치하면 참이다. 즉, 서브쿼리의 최대 결괏값과 비교한다.
EXISTS	서브쿼리에 비교 조건을 만족하는 결과가 존재하면 참이다. 컬럼으로 비교하지 않고 행의 존재 여부로 비교하기 때문에 EXISTS 연산자 앞에는 컬럼명을 넣지 않는다.

2. 복수 행 서브쿼리

■ [예제 6-4] '부산광역시' 고객이 주문한 주문건수를 보이시오.

```
SELECT COUNT(*) AS 주문건수
FROM 주문
WHERE 고객번호 IN (SELECT 고객번호
                     FROM 고객
                     WHERE 도시 = '부산광역시'
                  );
```

▶ 실행 결과

주문건수
51

- IN은 서브쿼리의 각 결과마다 = 연산자를 사용하여 비교함



그림 6-2 복수 행 비교 연산자 IN

2. 복수 행 서브쿼리

■ [예제 6-5] '부산광역시' 전체 고객의 마일리지보다 마일리지가 큰 고객의 정보를 보이시오.

```
SELECT 담당자명
      ,고객회사명
      ,마일리지
  FROM 고객
 WHERE 마일리지 > ANY (SELECT 마일리지
                           FROM 고객
                          WHERE 도시 = '부산광역시'
                         );
```

▶ 실행 결과

담당자명	고객회사명	마일리지
이준왕	굿모닝서울	15911
김성민	씨엔그룹	8788
최지수	오리안무역	6119
배순율	대장 인터내셔널	128790
박지혜	제원 인터네셔널	47865

- ANY를 사용하여 서브쿼리의 각 결과에 대해 > 연산자를 사용하여 비교한 후, 하나 이상 만족하는 고객의 정보를 반환함

마일리지
2819
806
7795
1177
7329

(a) 부산광역시 고객의 마일리지

고객회사명	마일리지	비교	결과
굿모닝서울	15911	15911>2819 OR 15911>806 OR 15911>7795 OR 15911>1177 OR 15911>7329	TRUE
엘케이 상사	406	406>2819 OR 406>806 OR 406>7795 OR 406>1177 OR 406> 7329	FALSE
씨엔그룹	8788	8788>2819 OR 8788>806 OR 8788>7795 OR 8788>1177 OR 8788>7329	TRUE
오리안무역	6119	6119>2819 OR 6119>806 OR 6119>7795 OR 6119>1177 OR 6119>7329	TRUE

(b) 고객 테이블의 각 고객 마일리지에 대하여 부산광역시 고객의 마일리지와 각각 비교한 후 결과 반환

그림 6-3 ANY를 사용한 서브쿼리의 처리 방식

2. 복수 행 서브쿼리

- [예제 6-6] 각 지역의 어느 평균 마일리지보다도 마일리지가 큰 고객의 정보를 보이시오.

```
SELECT 담당자명
      ,고객회사명
      ,마일리지
  FROM 고객
 WHERE 마일리지 > ALL (SELECT AVG(마일리지)
                           FROM 고객
                          GROUP BY 지역
                         );
```

▶ 실행결과

담당자명	고객회사명	마일리지
이은광	굿모닝서울	15911
박순용	대성 인터내셔널	128790
박지혜	제원인터넷스쿨	47865
홍소현	그린트머스	81479
조효주	대화유피스	78001
지승환	조종식풀	22260
조가영	태풍푸드	36960
한현구	찰인터프라이시스	116898
박근희	탈루드코리아	57746

2. 복수 행 서브쿼리

■ [예제 6-6] 각 지역의 어느 평균 마일리지보다도 마일리지가 큰 고객의 정보를 보이시오.

- ALL은 서브쿼리의 결과로 나온 각 지역의 평균 마일리지 중 최대 값보다 큰 마일리지를 가진 고객의 정보를 반환함

지역	AVG(마일리지)
경기도	11025.5152
경상남도	2290.7059
충상남도	8758.0000
경상북도	7065.0000
전라북도	943.0000
강원도	321.5000

(a) 지역별 평균 마일리지

고객회사명	마일리지	비교	결과
굿모닝서울	15911	15911 > 11025 AND 15911 > 2290 AND 15911 > 8758 AND 15911 > 7065 ...	TRUE
엘케이 상사	406	406 > 11025 AND 406 > 2290 AND 406 > 8758 AND 406 > 7065 ...	FALSE
씨엔그룹	8788	8788 > 11025 AND 8788 > 2290 AND 8788 > 8758 AND 8788 > 7065 ...	FALSE
오리안무역	6119	6119 > 11025 AND 6119 > 2290 AND 6119 > 8758 AND 6119 > 7065 ...	FALSE

(b) 고객 테이블의 각 고객 마일리지에 대하여 지역별 평균 마일리지를 각각 비교한 후 결과 반환

그림 6-4 ALL을 사용한 서브쿼리의 처리 방식

2. 복수 행 서브쿼리

- [예제 6-7] 한 번이라도 주문한 적이 있는 고객의 정보를 보이시오.

```
SELECT 고객번호  
      ,고객회사명  
FROM 고객  
WHERE EXISTS (SELECT *  
               FROM 주문  
              WHERE 고객번호 = 고객.고객번호  
            );
```

▶ 실행 결과

고객번호	고객회사명
ACDOR	굿모닝서울
ADHTR	얼케이 상사
AJHTR	씨언그룹
ANKFR	오리안무역
ANRFR	남화동합식품
ANSFR	서보일엔에프
ATRAN	대정 인티내셔널 제작

- IN 사용

```
SELECT 고객번호  
      ,고객회사명  
FROM 고객  
WHERE 고객번호 IN (SELECT DISTINCT 고객번호  
                     FROM 주문  
                   );
```

2. 복수 행 서브쿼리

■ [예제 6-7] 한 번이라도 주문한 적이 있는 고객의 정보를 보이시오.

- 조인 사용

```
SELECT DISTINCT 고객.고객번호  
      ,고객회사명  
  FROM 고객  
INNER JOIN 주문  
ON 고객.고객번호 = 주문.고객번호;
```

▶ 실행결과

고객번호	고객회사명
ACDDR	굿모닝서울
ADHTR	알케이 상사
AIHTR	씨엔그룹
ANIKFR	오티한우역
ANRFR	남해종합식품
ANSFR	서보암엔에프
ATRAN	대장 인터내셔널 제작이전

Section 03

사용 위치에 따른 서브쿼리

1. 조건절에서 사용하는 서브쿼리

- [예제 6-8] 고객 전체의 평균마일리지보다 평균마일리지가 큰 도시에 대해 도시명과 도시의 평균마일리지를 보이시오.

```
SELECT 도시
      ,AVG(마일리지) AS 평균마일리지
  FROM 고객
 GROUP BY 도시
 HAVING AVG(마일리지) > (SELECT AVG(마일리지)
                               FROM 고객
                             );
```

▶ 실행결과

도시	평균마일리지
서울특별시	11137.0800
김해시	8758.0000
인천광역시	13088.0000
대전광역시	14822.0000
제주시	8838.0000

2. FROM절에서 사용하는 서브쿼리

■ 인라인 뷰(Inline View)

- FROM절에서도 서브쿼리를 사용할 수 있는데, 이를 인라인 뷰라고 함
- 뷰에는 반드시 별명을 지정해주어야 함
- 인라인 뷰의 별명은 테이블명처럼 사용할 수 있음

2. FROM절에서 사용하는 서브쿼리

- [예제 6-9] 담당자명, 고객회사명, 마일리지, 도시, 해당 도시의 평균마일리지를 보이시오. 그리고 고객이 위치하는 도시의 평균마일리지와 각 고객의 마일리지 간의 차이도 함께 보이시오.

```
SELECT 담당자명
      ,고객회사명
      ,마일리지
      ,고객.도시 ● ①
      ,도시_평균마일리지
      ,도시_평균마일리지 - 마일리지 AS 차이
FROM 고객
     ,(SELECT 도시
          ,AVG(마일리지) AS 도시_평균마일리지 ● 인라인뷰
       FROM 고객
      GROUP BY 도시
     ) AS 도시별요약 ● 별명
WHERE 고객.도시 = 도시별요약.도시;
```

▶ 실행결과

담당자명	고객회사명	마일리지	도시	도시_평균마일리지	차이
이은정	굿모닝서울	15911	서울특별시	11137.0800	-4773.9200 ● 11,137-15,911
길병현	얼케어 상사	406	서울특별시	11137.0800	10731.0800 ● 11,137-406
김성민	씨엔그룹	8788	광주시	4587.5000	-4200.5000
최지수	오피안푸역	6119	서울특별시	11137.0800	5018.0800
강준	날해풀합식품	106	구리시	106.0000	0.0000
... 10명 ...	서보밀레액션	10408.0000	서울특별시	10408.0000	

3. SELECT절에서 사용하는 서브쿼리

■ 스칼라 서브쿼리(Scalar SubQuery)

- SELECT절 내에서도 서브쿼리를 사용할 수 있음. 이때 서브쿼리가 하나의 값을 생성하는 형태를 스칼라 서브쿼리라고 함
- 쿼리를 실행하여 반환되는 값은 메인 쿼리에서 사용됨
- 만약 서브쿼리의 결과로 행을 0개 반환한다면 메인 쿼리의 결과는 NULL이고, 행이 2개 이상 반환되면 오류가 발생함

3. SELECT절에서 사용하는 서브쿼리

■ [예제 6-10] 고객번호, 담당자명과 고객의 최종 주문일을 보이시오

```
SELECT 고객번호  
      ,담당자명  
      ,(SELECT MAX(주문일)  
        FROM 주문  
       WHERE 주문.고객번호 = 고객.고객번호  
     ) AS 최종주문일  
  FROM 고객;
```

▶ 실행결과

고객번호	담당자명	최종주문일
ACDDR	이은광	2021-01-11
ADHTR	김병렬	2021-09-27
AJHTR	김성민	2021-09-16
ANKFR	최지수	2021-12-17
AKNED	김준	2021-12-01

4. CTE

■ CTE(Common Table Expression)

- 쿼리로 만든 임시 데이터셋으로 WITH절에서 정의함
- 인라인 뷰와 마찬가지로 파생 테이블(Derived Table)처럼 사용할 수 있음
- 하나의 쿼리를 논리적인 블록으로 나눌 수 있어 가독성이 좋음
- 쿼리 내에서 여러 번 참조할 수 있기 때문에 재사용성이 좋음
- 형식

```
WITH CTE명 AS  
(  
    SELECT문  
)
```

4. CTE

■ [예제 6-11] [예제 6-9]를 CTE를 사용하여 작성하시오.

```
WITH 도시별요약 AS
(
    SELECT 도시
        ,AVG(마일리지) AS 도시_평균마일리지
    FROM 고객
    GROUP BY 도시
)

SELECT 담당자명
    ,고객회사명
    ,마일리지
    ,고객.도시
    ,도시_평균마일리지
    ,도시_평균마일리지 - 마일리지 AS 차이
FROM 고객
    ,도시별요약
WHERE 고객.도시 = 도시별요약.도시;
```

Section 04

상관 서브쿼리와
다중 컬럼 서브쿼리

1. 상관 서브쿼리

■ 상관 서브쿼리(Correlated SubQuery)

- 메인 쿼리와 서브쿼리 간의 상관관계를 포함하는 형태의 쿼리
- 상관 서브쿼리는 메인 쿼리를 한 행씩 처리함
- 이때 연관이 있는 컬럼을 기준으로 서브쿼리에서 값을 찾음
- 이로 인해 데이터가 많아지면 속도가 느려질 수 있으니 주의해야 함
- 상관 서브쿼리는 SELECT절, FROM절, WHERE절 등에서 사용할 수 있음

■ [예제 6-12] 사원 테이블에서 사원번호, 사원의 이름, 상사의 사원번호, 상사의 이름을 보이시오.

```
SELECT 사원번호  
      ,이름  
      ,상사번호  
      ,(SELECT 이름  
          FROM 사원 AS 상사  
         WHERE 상사.사원번호 = 사원.상사번호  
              ) AS 상사이름  
      FROM 사원;
```

▶ 실행결과

사원번호	이름	상사번호	상사 이름
E01	이소미	E06	이현진
E02	백자율		NULL
E03	유대현	E06	이현진
E04	최소민	E06	이현진
E05	안주홍	E09	유가을
	이현진		안주홍

2. 다중 컬럼 서브쿼리

■ 다중 컬럼 서브쿼리(Multi-Column SubQuery)

- 서브쿼리에서 여러 개의 컬럼을 사용하여 다중 비교를 하는 쿼리
- 다시 말해 서브쿼리의 결과로 나오는 여러 컬럼을 메인 쿼리의 값과 비교하여 결과를 출력함

■ [예제 6-13] 각 도시마다 최고 마일리지를 보유한 고객의 정보를 보이시오

```
SELECT 도시
      ,담당자명
      ,고객회사명
      ,마일리지
  FROM 고객
 WHERE (도시, 마일리지) IN (SELECT 도시, MAX(마일리지)
                               FROM 고객
                               GROUP BY 도시);
```

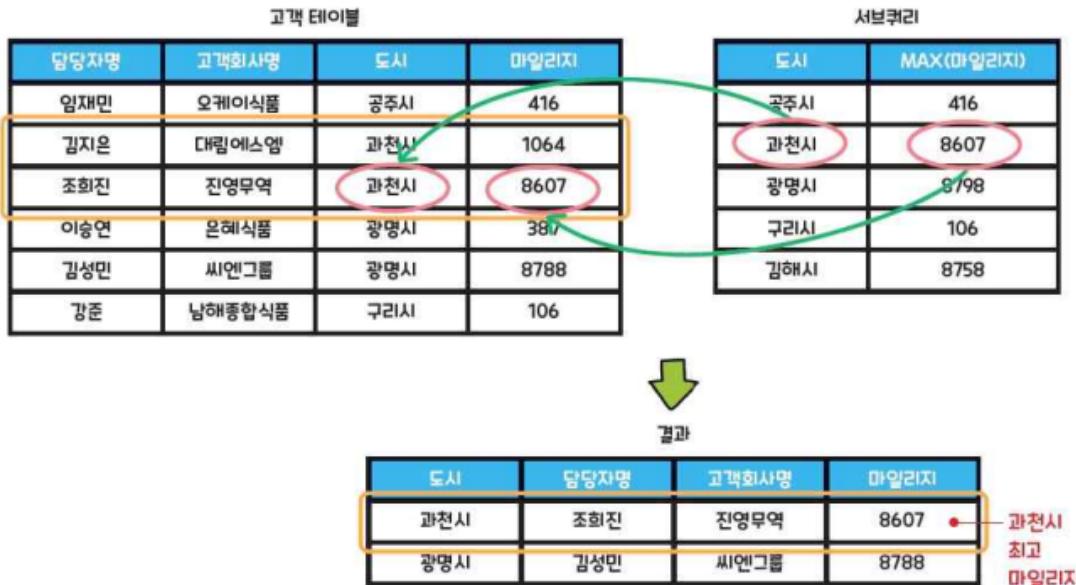
순서와 개수가 같아야 함

▶ 실행결과

도시	담당자명	고객회사명	마일리지
광명시	김성민	씨엔그룹	8788
구리시	강준	남해풍한식품	106
서울특별시	배순용	대장 인터내셔널	128790
김해시	오선주	케이리탈리스	8758
평택시	이두원	진가식품	122
인천광역시	홍소현	그린로더스	81479
상주시	김기호	태원 산업	7065
전주시	박동찬	시드니식품	943
동두천시	김경태	선인 상사	631
충주시	신민주	성민식품상사	382
부천시	최해주	세미원푸드	7106
과천시	조희진	진영우역	8607
제주시	주민영	오포락	8838
오산시	이수호	뉴서울제과	409
수원시	조연희	시대 통상	7135
제천시	손창민	청운유봉	3852
안산시	홍지우	신푸드	1149
부산광역시	오유진	제이티에스씨	7795

2. 다중 컬럼 서브쿼리

■ [예제 6-13] 각 도시마다 최고 마일리지를 보유한 고객의 정보를 보이시오



점검문제

점검문제

문제 1

'배재용' 사원의 부서명을 보이시오.

▶ 실행결과

부서명
기획부

문제 2

한 번도 주문한 적이 없는 제품의 정보를 보이시오.

▶ 실행결과

제품번호	제품명	포장단위	단가	재고	재고금액
78	얼론트 망고 주스	0.5l bottles	3000	80	240000

점검문제

문제 3

담당자명, 고객회사명, 주문건수, 최초주문일과 최종주문일을 보이시오.

▶ 실행 결과

담당자명	고객회사명	주문건수	최초주문일	최종주문일
이은광	굿모닝서울	6	2020-08-04	2022-01-11
김병현	엘케이 상사	6	2020-05-06	2021-09-27
김성민	씨엔그룹	3	2021-02-25	2021-09-16
최지수	오리안무역	15	2020-04-06	2021-12-17
강준	남해종합식품	3	2021-05-26	2021-12-01
김민경	서보암연애프	6	2020-09-30	2022-01-07
배승용	대장 인터내셔널	4	2020-05-27	2021-11-11
박지혜	제원인터넷소설	7	2020-12-16	2022-01-06

Chapter 07

DML: 데이터 조작어



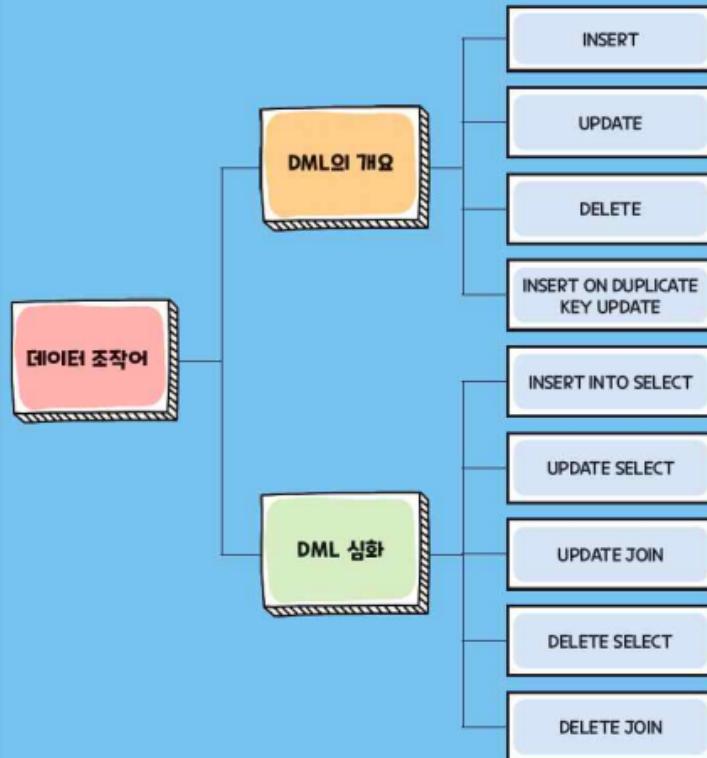
목차

1. DML의 개요
2. DML 심화

학습목표

- 데이터 조작어를 사용하여 SQL문을 작성할 수 있습니다.
- 서브쿼리를 활용한 데이터 조작어 문장을 작성할 수 있습니다.

Preview



Section 01

DML의 개요

1. INSERT

■ 데이터 조작어(Data Manipulation Language, DML)

- 데이터를 관리하는 데 사용하는 언어
- 데이터를 추가(INSERT), 변경(UPDATE), 삭제(DELETE)할 때 사용함

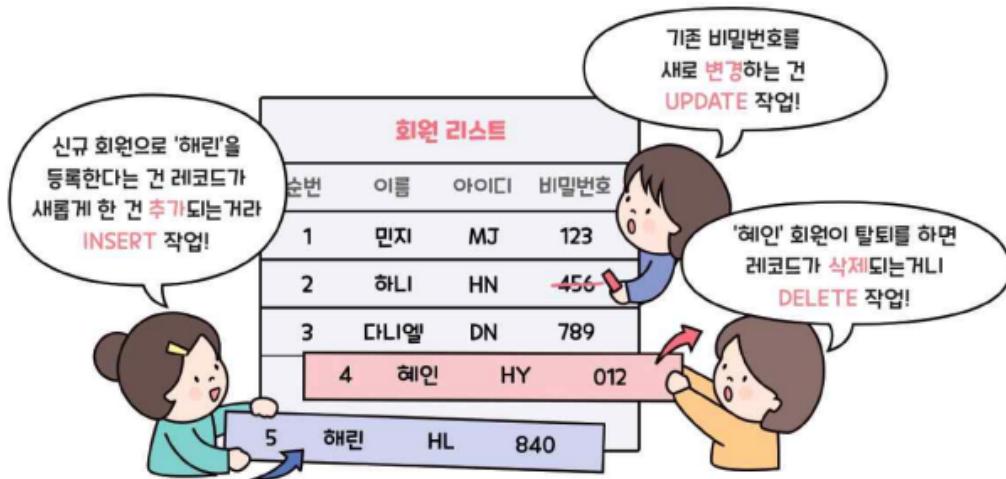


그림 7-1 INSERT, UPDATE, DELETE 작업

1. INSERT

■ INSERT

- 테이블에 새로운 행을 삽입하는 경우에 사용함
- 삽입할 컬럼명이 생략된 경우 삽입할 값이 테이블의 컬럼 수와 같아야 하며, 순서도 동일해야 함
- 형식

```
INSERT INTO 테이블명  
VALUES(값1, 값2, ...);
```

■ [예제 7-1] 부서 테이블에 다음 레코드를 삽입하시오.

- 부서번호: A5, 부서명: 마케팅부

```
INSERT INTO 부서  
VALUES('A5', '마케팅부');
```

▶ 실행결과

#	Time	Action
1	09:28:24	INSERT INTO 부서 VALUES('A5','마케팅부')

부서번호	부서명
A1	영업부
A2	기획부
A3	개발부
A4	홍보부
A5	마케팅부

그림 7-2 [예제 7-1] 레코드 삽입 결과

1. INSERT

■ [예제 7-2] 제품 테이블에 다음 레코드를 추가하시오

- 제품번호: 91, 제품명: 연어피클소스, 단가: 5000, 재고: 40

```
INSERT INTO 제품  
VALUES(91, '연어피클소스', NULL, 5000, 40);
```

▶ 실행 결과

제품번호	제품명	포장단위	단가	재고
73	천왕 순 상어알	150 g jars	3500	101
74	서울 구이 길	5 kg pkgs.	1000	4
75	뉴트리인 양주	0.5 l bottles	2400	125
76	찰스 츠플릿 드링크	500 ml	600	57
77	뉴트리 셀러드 드레싱	12 boxes	1300	32
78	밀른트 맥고 주스	0.5 l bottles	3000	80
91	연어피클소스		5000	40

1. INSERT

■ [예제 7-3] 제품 테이블에 다음 레코드를 추가하시오

- 제품번호: 90, 제품명: 연어핫소스, 단가: 4000, 재고: 50

```
INSERT INTO 제품(제품번호, 제품명, 단가, 재고)
VALUES(90, '연어핫소스', 4000, 50);
```

▶ 실행결과

제품번호	제품명	포장단위	단가	재고
65	핫오스스	150 g tars	100	100
77	큐브리 글러드 드레싱	12 boxes	1300	32
78	펩시콜라 주스	0.5l bottles	3000	80
90	연어핫소스	500g	4000	50
91	연어파울소스	500g	5000	40

1. INSERT

■ [예제 7-4] 사원 테이블에 한 문장으로 세 명의 레코드를 추가하시오.

- 사원번호: E20, 이름: 김사과, 직위: 수습사원, 성별: 남, 입사일: 현재 날짜
- 사원번호: E21, 이름: 박바나나, 직위: 수습사원, 성별: 여, 입사일: 현재 날짜
- 사원번호: E22, 이름: 정오렌지, 직위: 수습사원, 성별: 여, 입사일: 현재 날짜

```
INSERT INTO 사원(사원번호, 이름, 직위, 성별, 입사일)
VALUES('E20', '김사과', '수습사원', '남', CURDATE())
      ,('E21', '박바나나', '수습사원', '여', CURDATE())
      ,('E22', '정오렌지', '수습사원', '여', CURDATE());
```

▶ 실행 결과

사원번호	이름	성별	직위	생년	망월	입사일	주소	도시	지번	동전화	생년코드	부서번호
E01	이소미	Lee So Mi	사원	여	1985-12-05	2019-04-13	강남구 역삼동 36-8	서울특별시	강인	(02)578-8988	E06	A1
E02	배가용	Bae Ga Yong	대표이사	남	1973-02-17	2019-01-01	동탄동 16-11	부천시	갈인	(032)699-0136	E02	A2
E03	유다현	Yoo Da Hyun	사원	남	1988-08-27	2019-03-14	광진구 송정동 100-11	경주광역시	포모	(062)73-0256	E06	A1
E04	최소민	Choi So Min	사원	여	1987-09-17	2019-04-15	중구 중앙동 57-14	부산광역시	명봉	(051)587-4783	E06	A1
E05	안주훈	Ahn Ju Hoon	직원	남	1980-03-01	2018-12-29	남구 도화동 275-31	인천광역시	간인	(032)515-0278	E09	A1
E06	이현진	Lee Hyun Jin	직원	남	1983-06-30	2019-09-29	대구구 이현동 577-1	대구광역시	중부	(042)518-3876	E05	A1
E07	오경수	O Yeong Soo	사원	남	1989-05-27	2019-12-15	반기동 76-77	경주시	우부	(043)119-1784	E06	A1
E08	신하라	Seon Ha Ra	전산팀장	여	1982-01-07	2019-02-16	백현구 고당동 116	한주시	포남	(065)2983-1985	E02	A3
E09	유가을	Yoo Ga Eul	부정	여	1980-01-25	2019-10-29	남구 대종동 19-7	대구광역시	명봉	(053)465-1248	E02	A1
E10	정소진	Jung So Jin	수습사원	여	1993-10-07	2022-03-19	용산구 협곡동 협곡아파트 1-1501	서울특별시	3동	(02)824-2898		
E20	김사과	Kim Sa Gwa	수습사원	남	2023-03-03	2023-03-03	00000	00000	00000	00000	00000	00000
E21	박바나나	Park Ba Nana	수습사원	여	2023-03-03	2023-03-03	00000	00000	00000	00000	00000	00000
E22	정오렌지	Jung O Reungi	수습사원	여	2023-03-03	2023-03-03	00000	00000	00000	00000	00000	00000

2. UPDATE

■ UPDATE

- 기존 행에 있는 데이터 값을 변경할 때 사용함
- UPDATE문에 WHERE절이 없으면 모든 행의 값이 변경되므로 주의해야 함
- 형식

UPDATE 테이블명

SET 컬럼1 = 값1

, 컬럼2 = 값2

[WHERE 조건];

만약 UPDATE문을 실행했을 때 오류가 발생한다면
2장 61p의 <하나 더 알기>를 참고하세요.

■ [예제 7-5] 사원번호가 'E20'인 사원의 데이터를 다음과 같이 수정하시오.

- (사원번호: E20, 이름: 김사과) → (사원번호: E20, 이름: 김레몬)

UPDATE 사원

SET 이름 = '김레몬'

WHERE 사원번호 = 'E20';

▶ 실행결과

사원번호	이름
E10	정수진
E20	김사과
E21	박바나나



사원번호	이름
E10	정수진
E20	김레몬
E21	박바나나

2. UPDATE

■ [예제 7-6] 제품번호가 91인 제품에 대하여 포장단위를 추가하시오.

- (제품번호: 91, 포장단위: NULL) → (제품번호: 91, 포장단위: 200 ml bottles)

UPDATE 제품

SET 포장단위 = '200 ml bottles'

WHERE 제품번호 = 91;

▶ 실행 결과

제품번호	제품명	포장단위
77	뉴트리 셀러드 드레싱	12 boxes
78	알몬트 망고 주스	0.5 l bottles
90	연어 핫소스	NULL
91	연어 피플소스	NULL



제품번호	제품명	포장단위
77	뉴트리 셀러드 드레싱	12 boxes
78	알몬트 망고 주스	0.5 l bottles
90	연어 핫소스	NULL
91	연어 피플소스	200 ml bottles

2. UPDATE

- [예제 7-7] 제품번호가 91인 제품에 대하여 단가를 10% 인상하고, 재고에서 10을 뺀 값으로 변경하시오.

UPDATE 제품

SET 단가 = 단가 * 1.1

,재고 = 재고 - 10

WHERE 제품번호 = 91;

▶ 실행 결과

제품번호	제품명	포장단위	단가	재고
77	뉴트리 살러드 드레싱	12 boxes	1300	32
78	알몬트 망고 주스	0.5 l bottles	3000	80
90	연어핫소스	100ml	4000	50
91	연어피클소스	200 ml bottles	5000	40



제품번호	제품명	포장단위	단가	재고
77	뉴트리 살러드 드레싱	12 boxes	1300	32
78	알몬트 망고 주스	0.5 l bottles	3000	80
90	연어핫소스	100ml	4000	50
91	연어피클소스	200 ml bottles	5500	30

3. DELETE

■ DELETE

- 기준에 있는 행을 삭제할 때 사용함
- 만약 DELETE문에 WHERE절이 없으면 모든 행이 삭제되므로 주의해야 함
- 형식

```
DELETE FROM 테이블명  
[WHERE 조건];
```

■ [예제 7-8] 제품 테이블에서 제품번호가 91인 레코드를 삭제하시오.

```
DELETE FROM 제품  
WHERE 제품번호 = 91;
```

▶ 실행결과

제품번호	제품명	포장단위	단가	재고
77	뉴트리 샐러드 드레싱	12 boxes	1300	32
78	얼몬트 망고 주스	0.5 l bottles	3000	80
90	연어핫소스	100ml	4000	50
91	연어피클소스	200 ml bottles	5500	30



제품번호	제품명	포장단위	단가	재고
77	뉴트리 샐러드 드레싱	12 boxes	1300	32
78	얼몬트 망고 주스	0.5 l bottles	3000	80
90	연어핫소스	100ml	4000	50

3. DELETE

- [예제 7-9] 사원 테이블에서 입사일이 가장 늦은 사원 3명의 레코드를 삭제하시오.

```
DELETE FROM 사원  
ORDER BY 입사일 DESC  
LIMIT 3;
```

- [예제 7-4]에서 추가한 '김레몬', '박바나나', '정오렌지'가 삭제된 것을 확인하는 쿼리

```
SELECT *  
FROM 사원  
WHERE 이름 IN('김레몬', '박바나나', '정오렌지');
```

▶ 실행결과

사원번호	이름	영문이름	직위	성별	성별	입사일	주소	도시	지역	집전화	상사번호	우편번호
1001						1990-01-01						

4. INSERT ON DUPLICATE KEY UPDATE

■ INSERT ON DUPLICATE KEY UPDATE

- 레코드가 없다면 새롭게 추가하고, 이미 있다면 데이터를 변경하는 경우에 사용
- 형식

```
INSERT INTO 테이블명(컬럼1, 컬럼2, 컬럼3, ...) • ①  
VALUES(값1, 값2, 값3, ...)  
ON DUPLICATE KEY UPDATE  
컬럼2 = 값2, 컬럼3 = 값3, ...;
```

4. INSERT ON DUPLICATE KEY UPDATE

■ [예제 7-10] 91번 제품이 없다면 레코드를 추가하고, 이미 존재한다면 값을 변경하시오.

- 제품번호: 91, 제품명: 연어피클핫소스, 단가: 6000, 재고: 50

```
INSERT INTO 제품(제품번호, 제품명, 단가, 재고)
VALUES(91, '연어피클핫소스', 6000, 50)
ON DUPLICATE KEY UPDATE
제품명 = '연어피클핫소스', 단가 = 6000, 재고 = 50;
```

▶ 실행결과

제품번호	제품명	포장단위	단가	재고
75	뉴트리인 맥주	0.5 l bottles	2400	125
76	찰스 조플릿 드링크	500 ml	600	57
77	뉴트리 살러드 드레싱	12 boxes	1300	32
78	알콘트 맥고 주스	0.5 l bottles	3000	80
90	연어핫소스	NULL	4000	50
91	연어피클핫소스	NULL	6000	50

Section 02

DML 심화

1. INSERT INTO SELECT

■ INSERT INTO SELECT

- SELECT문의 결과를 다른 테이블에 삽입하려면 INSERT문에서 VALUES 대신 SELECT 문장을 넣음
- 이때 SELECT문의 컬럼 수와 INSERT문에 있는 컬럼 수는 동일해야 하고, 순서도 같아야 함
- 형식

```
INSERT INTO 테이블명(컬럼A, 컬럼B)
    SELECT 컬럼1, 컬럼2
    FROM 테이블명
    [WHERE 조건];
```

1. INSERT INTO SELECT

■ [예제 7-11] 고객주문요약 테이블을 만들고, 레코드를 추가하시오.

- 테이블 생성에 관한 내용은 8장에서 다룸

표 7-1 고객주문요약 테이블의 컬럼과 자료형

컬럼명	자료형
고객번호(기본키)	CHAR(5)
고객회사명	VARCHAR(50)
주문건수	INT
최종주문일	DATE

- 고객주문요약 테이블 생성

```
CREATE TABLE 고객주문요약
(
    고객번호 CHAR(5) PRIMARY KEY
    ,고객회사명 VARCHAR(50)
    ,주문건수 INT
    ,최종주문일 DATE
);
```

1. INSERT INTO SELECT

■ [예제 7-11] 고객주문요약 테이블을 만들고, 레코드를 추가하시오.

- 레코드 삽입

```
INSERT INTO 고객주문요약
    SELECT 고객.고객번호
          ,고객회사명
          ,COUNT(*)
          ,MAX(주문일)
     FROM 고객
          ,주문
   WHERE 고객.고객번호 = 주문.고객번호
 GROUP BY 고객.고객번호
          ,고객회사명;
```

▶ 실행결과

고객번호	고객회사명	주문건수	최종주문일
ACDDR	굿모닝서울	6	2022-01-11
ADHTR	얼케이상사	6	2021-09-27
AIHTR	씨언그룹	3	2021-09-16
ANKFR	오리안무역	15	2021-12-17
ANRFR	남해중합식품	3	2021-12-01
AMCER	서보열연예프	5	2022-01-07

2. UPDATE SELECT

■ UPDATE SELECT

- UPDATE문에서 서브쿼리를 사용할 수도 있음
- 변경할 값을 얻기 위해서는 SET절에서 SELECT문을 사용함
- 서브쿼리의 SELECT문에는 반드시 별명을 붙여야 하고, 서브쿼리의 결과는 단일 값이어야 함
- 형식

```
UPDATE 테이블명  
SET 컬럼명 = (  
    SELECT *  
    FROM (  
        SELECT 컬럼A  
        FROM ...  
    ) AS 별명  
    WHERE 조건  
);
```

2. UPDATE SELECT

- [예제 7-12] 제품번호가 91인 제품의 단가를 '소스' 제품들의 평균단가로 변경하시오.

```
UPDATE 제품
SET 단가 = (
    SELECT *
    FROM (
        SELECT AVG(단가)
        FROM 제품
        WHERE 제품명 LIKE '%소스%'
    ) AS t
)
WHERE 제품번호 = 91;
```

▶ 실행 결과

제품번호	제품명	단가	재고
75	뉴트리인 맥주	2400	125
76	찰스 조플릿 드링크	600	57
77	뉴트리 쌀러드 드레싱	1300	32
78	월든트 망고 주스	3000	80
90	연어핫소스	4000	50
91	연어파울릿소스	6000	50



제품번호	제품명	단가	재고
75	뉴트리인 맥주	2400	125
76	찰스 조플릿 드링크	600	57
77	뉴트리 쌀러드 드레싱	1300	32
78	월든트 망고 주스	3000	80
90	연어핫소스	4000	50
91	연어파울릿소스	2831	50

2. UPDATE SELECT

■ UPDATE SELECT

- WHERE절에서 서브쿼리 사용
- 형식

```
UPDATE 변경할_테이블명  
    ,서브쿼리  
SET 컬럼1 = 변경할 값  
WHERE 컬럼2 IN (서브쿼리.컬럼);
```

2. UPDATE SELECT

■ [예제 7-13] 한 번이라도 주문한 적이 있는 고객의 마일리지를 10% 인상하시오.

```
UPDATE 고객
  (
    SELECT DISTINCT 고객번호
    FROM 주문
  ) AS 주문고객
SET 마일리지 = 마일리지 * 1.1
WHERE 고객.고객번호 IN (주문고객.고객번호);
```

- 결과를 확인하기 위한 SELECT문

```
SELECT *
FROM 고객
WHERE 고객번호 IN (SELECT DISTINCT 고객번호
                     FROM 주문
                   );
```

▶ 실행결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
ACDDR	국도님서울	이준동	영업 과장	송파구 잠실동 220	서울특별시	서울	(02)978-1984	17502
ADHTR	알케이 상사	김병현	영업 사원	동작구 읍석 3동 140-3	서울특별시	서울	(02)345-1945	447
AHTR	씨언그룹	김성민	영업 사원	가락동 301	광명시	경기도	(02)31-0345	9667
ANWFR	오리안무역	최지수	마케팅 과장	성북구 길음동 136-11	서울특별시	서울	(02)123-0345	6731
ANWFO	날개풀판스	김준	마케팅 과장	***-435	부산시	경기	(034)756-200*	12345

3. UPDATE JOIN

■ UPDATE JOIN

- INNER JOIN을 사용하여 다른 테이블의 행과 일치하는 행을 수정할 수 있음
- LEFT OUTER JOIN을 사용하여 그렇지 않은 행을 수정할 수도 있음
- 변경할 값에는 상수나 수식뿐만 아니라 조인할 테이블에 있는 컬럼에 기반하여 값을 넣을 수도 있음
- ANSI SQL 표현

```
UPDATE 변경할_테이블A  
[INNER|LEFT| OUTER] JOIN 조인할_테이블B  
ON 조인_조건  
SET 테이블A.컬럼명 = 값  
WHERE 기타_조건;
```

- Non-ANSI SQL 표현

```
UPDATE 변경할_테이블A  
,조인할_테이블B  
SET 테이블A.컬럼명 = 값  
WHERE 조인_조건  
AND 기타_조건;
```

3. UPDATE JOIN

■ [예제 7-14] 마일리지등급이 'S'인 고객의 마일리지에 1000점씩 추가하시오.

- ANSI SQL 표현

```
UPDATE 고객
INNER JOIN 마일리지등급
ON 마일리지 BETWEEN 하한마일리지 AND 상한마일리지
SET 마일리지 = 마일리지 + 1000
WHERE 등급명 = 'S';
```

- Non-ANSI SQL 표현

```
UPDATE 고객
,마일리지등급
SET 마일리지 = 마일리지 + 1000
WHERE 마일리지 BETWEEN 하한마일리지 AND 상한마일리지
AND 등급명 = 'S';
```

3. UPDATE JOIN

■ [예제 7-14] 마일리지등급이 'S'인 고객의 마일리지에 1000점씩 추가하시오.

- 결과를 확인하기 위한 SELECT문

```
SELECT 고객번호  
      ,고객회사명  
      ,마일리지  
FROM 고객  
INNER JOIN 마일리지등급  
ON 마일리지 BETWEEN 하한마일리지AND 상한마일리지  
WHERE 등급명 = 'S';
```

▶ 실행결과

고객번호	고객회사명	마일리지
ATRAN	대정 인터내셔널	141669
PRDSU	풀인더프라이시스	128588



고객번호	고객회사명	마일리지
ATRAN	대정 인터내셔널	142669
PRDSU	풀인더프라이시스	129588

4. DELETE SELECT

■ DELETE SELECT

- DELETE문에서도 삭제할 레코드를 찾기 위하여 서브쿼리를 사용할 수 있음
- 형식

```
DELETE FROM 테이블A  
WHERE 컬럼명 IN (  
    SELECT 컬럼명  
    FROM 테이블B  
);
```

■ [예제 7-15] 주문 테이블에는 존재하나 주문세부 테이블에는 존재하지 않는 주문번호를 주문 테이블에서 삭제하시오.

```
DELETE FROM 주문  
WHERE 주문번호 NOT IN (  
    SELECT DISTINCT 주문번호  
    FROM 주문세부  
);
```

4. DELETE SELECT

- [예제 7-15] 주문 테이블에는 존재하나 주문세부 테이블에는 존재하지 않는 주문번호를 주문 테이블에서 삭제하시오.

주문 테이블

주문번호	고객번호	사원번호	주문일	요청일	발송일
H1071	LASLI	E01	2022-01-12	2022-02-09	2022-01-20
H1072	NSHER	E04	2022-01-12	2022-02-09	2022-01-20
H1073	RICPE	E03	2022-01-12	2022-02-09	2022-01-20
H1074	MORSI	E07	2022-01-13	2022-02-10	2022-01-23
H1075	CSURI	E07	2022-01-13	2022-02-10	2022-01-23
H1076	NAPRO	E04	2022-01-13	2022-02-10	2022-01-23
H1077	TTCRA	E01	2022-01-13	2022-02-10	2022-01-23

주문세부 테이블

주문번호	제품번호	단가	주문수량	할인율
H0248	11	1400	12	0
H0248	42	1000	10	0
H0248	72	3500	5	0
H0249	14	1900	9	0
H0249	51	4200	40	0
H0250	41	800	10	0
H0250	51	4200	15	0.15
H0250	65	1700	15	0.15
H0251	22	1700	6	0.05

주문 테이블에는 존재하지만
주문세부 테이블에는
존재하지 않는 주문번호

주문번호	고객번호	사원번호	주문일	요청일	발송일
H1077	TTCRA	E01	2022-01-13	2022-02-10	2022-01-23

주문 테이블에서 삭제

그림 7-3 [예제 7-15] 진행 과정

4. DELETE SELECT

하나 더 알기 V

Collation 에러가 발생하는 경우

조인을 하거나 IF, CASE 등으로 값을 비교할 때, 양쪽 테이블의 Collation이 다를 때 다음과 같은 오류가 발생할 수 있습니다.

Error Code: 1267. Illegal mix of collations

이런 오류가 발생할 때는 다음과 같이 테이블의 Collation을 변경시켜주면 문제를 해결할 수 있습니다.

```
ALTER TABLE 고객 CONVERT TO CHAR SET utf8mb4 COLLATE utf8mb4_general_ci;  
ALTER TABLE 주문 CONVERT TO CHAR SET utf8mb4 COLLATE utf8mb4_general_ci;  
ALTER TABLE 주문세부 CONVERT TO CHAR SET utf8mb4 COLLATE utf8mb4_  
general_ci;  
ALTER TABLE 제품 CONVERT TO CHAR SET utf8mb4 COLLATE utf8mb4_general_ci;  
...
```



5. DELETE JOIN

■ DELETE JOIN

- INNER JOIN을 사용하여 두 테이블에서 일치하는 행을 모두 삭제할 수 있음
- LEFT OUTER JOIN을 사용하여 일치하지 않는 행을 삭제할 수도 있음
- ANSI SQL 표현

```
DELETE 테이블A  
      ,테이블B  
FROM 테이블A  
[INNER|LEFT OUTER] JOIN 테이블B  
ON 조인_조건  
WHERE 삭제할_조건;
```

- Non-ANSI SQL 표현

```
DELETE 테이블A  
      ,테이블B  
FROM 테이블A  
      ,테이블B  
WHERE 조인_조건  
AND 삭제할_조건;
```

5. DELETE JOIN

■ [예제 7-16] 주문번호 'H0248'에 대한 내역을 주문 테이블과 주문세부 테이블에서 모두 삭제하시오. 하나의 문장으로 작업을 수행해보시오.

- 주문 테이블, 주문세부 테이블에 주문번호 'H0248'이 존재하는지 확인하기

```
SELECT *  
FROM 주문  
WHERE 주문번호 = 'H0248';
```

▶ 실행 결과

주문번호	고객번호	사원번호	주문일	요청일	발송일
H0248	NETVI	E04	2020-03-12	2020-04-09	2020-03-24

```
SELECT *  
FROM 주문세부  
WHERE 주문번호 = 'H0248';
```

▶ 실행 결과

주문번호	제품번호	단가	주문수량	확인율
H0248	11	1400	12	0
H0248	42	1000	10	0
H0248	72	3500	5	0

5. DELETE JOIN

■ [예제 7-16] 주문번호 'H0248'에 대한 내역을 주문 테이블과 주문세부 테이블에서 모두 삭제하시오. 하나의 문장으로 작업을 수행해보시오.

- 주문 테이블, 주문세부 테이블에 주문번호 'H0248'를 동시에 삭제하기
- ANSI SQL 표현

```
DELETE 주문
      ,주문세부
FROM 주문
INNER JOIN 주문세부
ON 주문.주문번호 = 주문세부.주문번호
WHERE 주문.주문번호 = 'H0248';
```

- Non-ANSI SQL 표현

```
DELETE 주문
      ,주문세부
FROM 주문
      ,주문세부
WHERE 주문.주문번호 = 주문세부.주문번호
AND 주문.주문번호 = 'H0248';
```

▶ 실행 결과

• 주문 테이블에서 주문번호 'H0248' 레코드 확인

주문번호	고객번호	사원번호	주문일	요청일	발송일
H0248	H0001	H0001	H0001	H0001	H0001

• 주문세부 테이블에서 주문번호 'H0248' 레코드 확인

주문번호	제품번호	단가	주문수량	합인출
H0248	H0001	H0001	H0001	H0001

5. DELETE JOIN

■ [예제 7-17] 한 번도 주문한 적이 없는 고객의 정보를 삭제하시오

- 한 번도 주문한 적이 없는 고객 검색하기

```
SELECT 고객.*  
FROM 고객  
LEFT OUTER JOIN 주문  
ON 고객.고객번호 = 주문.고객번호  
WHERE 주문.고객번호 IS NULL;
```

▶ 실행 결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
BQQZA	비큐ью푸드	황수영	대표 이사	서초구 서초동 120-3	서울특별시	서울	(02)456-9876	100
RISPA	한림통상	김경현	대표 이사	송파구 창실본동 150	서울특별시	서울	(02)909-2094	6942
SSAFI	대동 무역	정수관	회계 과장	동구 마산동 250-1	대전광역시	대전	(042)33-3433	975
TTRAN	해피트리푸드	김희원	영업 과장	동작구 죽석동 220-12	서울특별시	서울	(02)756-9787	448

5. DELETE JOIN

■ [예제 7-17] 한 번도 주문한 적이 없는 고객의 정보를 삭제하시오

- 주문한 적이 없는 고객의 레코드를 고객 테이블에서 삭제하기

```
DELETE 고객
FROM 고객
LEFT JOIN 주문
ON 고객.고객번호 = 주문.고객번호
WHERE 주문.고객번호 IS NULL;
```

- 삭제된 고객 정보가 고객 테이블에 존재하는지 확인하기

```
SELECT *
FROM 고객
WHERE 고객번호 IN ('BQQZA', 'RISPA', 'SSAFI', 'TTRAN');
```

▶ 실행 결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

점검문제

점검문제

문제 1

제품 테이블에 레코드를 추가하시오.

- 제품번호: 95, 제품명: 망고베리 아이스크림, 포장단위: 400g, 단가: 800, 재고: 30

▶ 실행 결과

제품번호	제품명	포장단위	단가	재고
78	월든트 망고 주스	0.5l bottles	3000	80
90	연어핫소스	100ml	4000	50
91	연어피클핫소스	100ml	3283	50
95	망고베리 아이스크림	400g	800	30

문제 2

제품 테이블에 레코드를 추가하시오.

- 제품번호: 96, 제품명: 눈꽃빙수맛 아이스크림, 단가: 2000

▶ 실행 결과

제품번호	제품명	포장단위	단가	재고
90	연어핫소스	100ml	4000	50
91	연어피클핫소스	100ml	3283	50
95	망고베리 아이스크림	400g	800	30
96	눈꽃빙수맛 아이스크림	100ml	2000	100

점검문제

문제 3

[문제 2]에서 추가한 96번 제품의 재고를 30으로 변경하시오.

▶ 실행결과

제품번호	제품명	포장단위	단가	재고
90	연어핫소스	EA	4000	50
91	연어파클릿소스	EA	3283	50
95	망고베리 아이스크림	400g	800	30
96	눈꽃빙수맛 아이스크림	EA	2000	30

문제 4

사원이 한 명도 존재하지 않는 부서를 부서 테이블에서 삭제하시오.

▶ 실행결과

부서번호	부서명
A1	영업부
A2	기획부
A3	개발부
A4	홍보부
A5	마케팅부



부서번호	부서명
A1	영업부
A2	기획부
A3	개발부

Chapter 08

DDL: 데이터 정의어



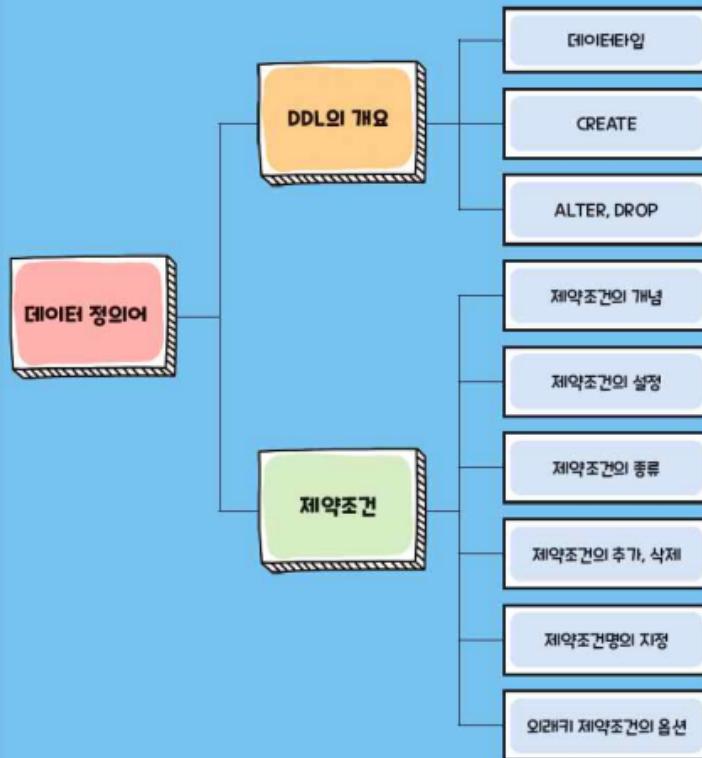
목차

1. DDL의 개요
2. 데이터타입
3. CREATE
4. ALTER, DROP
5. 제약조건

학습목표

- 데이터 정의어를 작성할 수 있습니다.
- 제약조건을 이해하고 적용할 수 있습니다

Preview



Section 01

DDL의 개요

1. DDL의 개요

■ 데이터 정의어(Data Definition Language, DDL)

- 데이터베이스 내에 테이블이나 인덱스, 뷰 등의 객체를 만들거나 수정, 삭제할 때 사용하는 언어를 의미함
- DDL에는 CREATE, ALTER, DROP이 있음

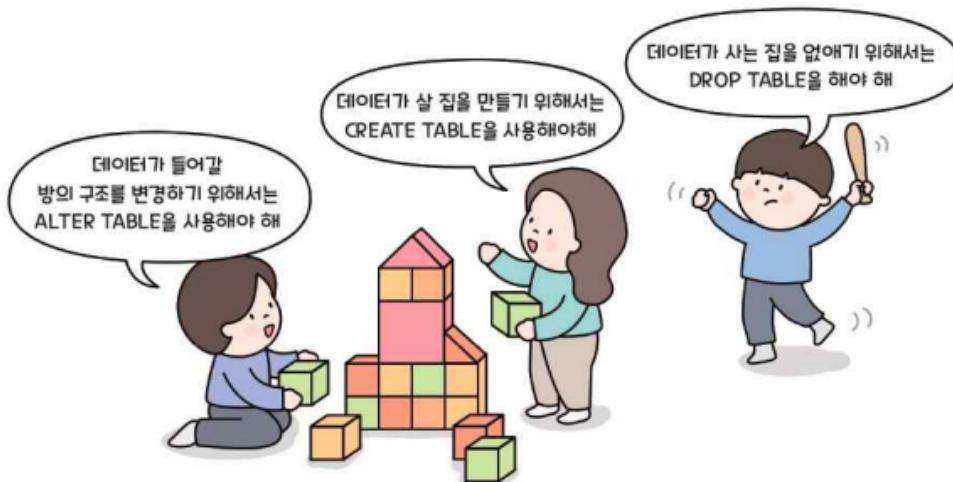


그림 8-1 CREATE, ALTER, DROP의 개념

Section 02

데이터타입

1. 문자형 데이터타입

■ 문자형 데이터타입

- CHAR

고정길이 문자형 데이터타입

- ✓ 지정한 길이보다 데이터 길이가 작으면 빈칸만큼 공백(Blank)이 들어감
- ✓ [예] 고객번호, 주민등록번호

- VARCHAR

- ✓ 가변길이 문자형 데이터타입
- ✓ 데이터의 길이만큼의 메모리만 차지함
- ✓ [예] 고객회사명, 주소

- TEXT

- ✓ VARCHAR과 달리 길이를 지정하지 않음
- ✓ 컬럼의 최대 길이를 모르는 경우에 사용하기 적합함

1. 문자형 데이터타입

■ 문자형 데이터타입

표 8-1 문자형 데이터타입

데이터타입	설명	최대크기(단위: Bytes)
CHAR(길이)	고정길이 문자형 데이터타입	255
VARCHAR(길이)	가변길이 문자형 데이터타입	65,535
TINYTEXT	문자열 데이터타입	255
TEXT	문자열 데이터타입	65,535
MEDIUMTEXT	문자열 데이터타입	16,777,215
LONGTEXT	문자열 데이터타입	4,294,967,295
JSON	JSON 문자열 데이터타입	1GB

2. 숫자형 데이터타입

■ 숫자형 데이터타입

- 정수형

- ✓ 들어갈 데이터의 크기에 따라 TINYINT부터 BIGINT 중에서 선택하여 지정할 수 있음

표 8-2 정수형 데이터타입

데이터타입	최대크기(단위: Bytes)
TINYINT	1
SMALLINT	2
MEDIUMINT	3
INT	4
BIGINT	8

- 실수형

- ✓ FLOAT형, DOUBLE형, DECIMAL형 중에서 선택하여 지정할 수 있음

표 8-3 실수형 데이터타입

데이터타입	표현 형식	최대크기
FLOAT	부동 소수점 형식	4Bytes 소수점 아래 7자리까지 표현
DOUBLE	부동 소수점 형식	8Bytes 소수점 아래 15자리까지 표현
DECIMAL(M,D) M: 전체 자릿수 D: 소수점 자릿수	고정 소수점 형식 DEC, NUMERIC이라고도 함	전체 자릿수(M)은 65까지 표현 소수점 자릿수(D)는 30까지 표현 ❶ DECIMAL(4,2): -99.99 ~ 99.99까지 저장

3. 날짜시간형 데이터타입

■ 날짜시간형 데이터타입

- DATE는 날짜, TIME은 시간을 저장하기 위한 데이터타입
- DATETIME과 TIMESTAMP는 날짜와 시간을 함께 저장할 수 있는 데이터타입
 - ✓ DATETIME과 TIMESTAMP의 차이는 시간대(Time Zone)의 적용 여부임
TIMESTAMP는 내부적으로 시간을 가져올 때 시간대를 적용시켜 보여줌
 - ✓ 즉, 데이터베이스가 글로벌 서비스에서 사용된다면 DATETIME 대신
TIMESTAMP를 사용해야 함

표 8-4 날짜시간형 데이터타입

데이터타입	형식	데이터 범위
DATE	YYYY-MM-DD	1000-01-01 ~ 9999-12-31
TIME	HH:MI:SS	-838:59:59 ~ 838:59:59
DATETIME	YYYY-MM-DD HH:MI:SS	1000-01-01 00:00:00 ~ 9999-12-31 23:59:59
TIMESTAMP	YYYY-MM-DD HH:MI:SS	1970-01-01 00:00:01 UTC ~ 2038-01-19 03:14:07 UTC

3. 날짜시간형 데이터타입

하나 더 알기 V

시간대

시간대(Time Zone)란 지역 사이에 생기는 낮과 밤의 차이를 인위적으로 조정하기 위해 고안된 시간의 구분선을 의미합니다. 이는 영국의 그리니치 천문대를 기준으로 지역에 따른 시간의 차이를 계산하여 적용하기 때문에 GMT(Greenwich Mean Time)라 지칭합니다.

UTC(Coordinated Universal Time)은 1972년 1월 1일부터 시행된 국제 표준시로 UTC와 GMT는 초의 소수점 단위에서만 차이가 나기 때문에 흔용되어 사용되기도 하지만 기술적인 표기에서는 UTC가 사용됩니다. 우리나라에서는 KST(Korea Standard Time)을 표준시로 사용하는데, 이는 UTC에 9시간을 더한 시간대입니다.



4. 이진형 데이터타입, 공간형 데이터타입

■ 이진형 데이터타입

- BINARY 또는 BLOB(Binary Large OBject) 데이터타입
- BINARY(n)와 BYTE(n)는 CHAR 형태의 이진형 데이터타입
- VARBINARY(n)는 VARCHAR 형태의 이진형 데이터타입
- BLOB도 저장 크기에 따라 TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB 등의 데이터타입을 사용할 수 있음

■ 공간형 데이터타입

- 공간 데이터를 저장하기 위한 데이터타입
- GEOMETRY, POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION 등이 있음

Section 03

CREATE

1. 데이터베이스 생성하기

■ CREATE

- 데이터베이스나 테이블, 뷰, 인덱스 등 객체를 만들 때 사용함
- 형식

```
CREATE DATABASE [IF NOT EXISTS] 데이터베이스명;
```

- MySQL에서 한글을 사용하려면 Character Set 설정

```
CREATE DATABASE 데이터베이스명 [CHAR SET utf8mb4 COLLATE utf8mb4_general_ci];
```

■ [예제 8-1] 한빛학사 데이터베이스를 생성하시오.

```
CREATE DATABASE 한빛학사;
```

2. 테이블 생성하기

■ 테이블 생성

- 테이블을 생성할 때 모든 컬럼에는 데이터타입을 지정해주어야 함
- 형식

```
CREATE TABLE 테이블명  
(  
    컬럼1 데이터타입  
    ,컬럼2 데이터타입  
);
```

2. 테이블 생성하기

- [예제 8-2] 다음과 같이 한빛학사 데이터베이스에 학과 테이블을 생성하고, 레코드 3건을 삽입하시오.

■ 학과 테이블 명세서

컬럼명	데이터타입
학과번호	고정문자형 2자
학과명	기변문자형 20자
학과장명	기변문자형 20자

■ 학과 데이터

학과번호	학과명	학과장명
AA	컴퓨터공학과	배경민
BB	소프트웨어학과	김남준
CC	디자인융합학과	박선영

2. 테이블 생성하기

- [예제 8-2] 다음과 같이 한빛학사 데이터베이스에 학과 테이블을 생성하고, 레코드 3건을 삽입하시오.

- 학과 테이블을 생성

```
CREATE TABLE 학과
(
    학과번호 CHAR(2)
    ,학과명 VARCHAR(20)
    ,학과장명 VARCHAR(20)
);
```

- 학과 테이블에 3건의 레코드를 삽입

```
INSERT INTO 학과
VALUES('AA','컴퓨터공학과','배경민')
      ,('BB','소프트웨어학과','김남준')
      ,('CC','디자인융합학과','박선영');
```

▶ 실행 결과

학과번호	학과명	학과장명
AA	컴퓨터공학과	배경민
BB	소프트웨어학과	김남준
CC	디자인융합학과	박선영

2. 테이블 생성하기

■ [예제 8-3] 다음과 같이 학생 테이블을 생성하고, 레코드 3건을 삽입하시오.

■ 학생 테이블 명세서

컬럼명	데이터타입
학번	고정문자형 5자
이름	가변문자형 20자
나이	숫자형
연락처	숫자형
학과명	가변문자형 20자

- Q) 학생 테이블 명세서에서 적절하지 않은 컬럼은?
- A) 나이, 연락처, 학과명

2. 테이블 생성하기

■ [예제 8-3] 다음과 같이 학생 테이블을 생성하고, 레코드 3건을 삽입하시오.

■ 변경된 학생 테이블 명세서

컬럼명	데이터타입
학번	고정문자형 5자
이름	기변문자형 20자
생일	날짜형
연락처	기변문자형 20자
학과번호	고정문자형 2자

■ 학생 데이터

학과번호	이름	생일	연락처	학과번호
S0001	이윤주	2020-01-30	01033334444	AA
S0001	이승은	2021-02-23	NULL	AA
S0003	백재용	2018-03-31	01077778888	DD

2. 테이블 생성하기

■ [예제 8-3] 다음과 같이 학생 테이블을 생성하고, 레코드 3건을 삽입하시오.

- 학생 테이블 생성

```
CREATE TABLE 학생
(
    학번 CHAR(5)
    ,이름 VARCHAR(20)
    ,생일 DATE
    ,연락처 VARCHAR(20)
    ,학과번호 CHAR(2)
);
```

- 학생 데이터 표를 참고하여 3건의 레코드 삽입

```
INSERT INTO 학생
VALUES('S0001', '이윤주', '2020-01-30', '01033334444', 'AA')
      ,('S0001', '이승은', '2021-02-23', NULL, 'AA')
      ,('S0003', '백재용', '2018-03-31', '01077778888', 'DD');
```

▶ 실행 결과

학번	이름	생일	연락처	학과번호
S0001	이윤주	2020-01-30	01033334444	AA
S0001	이승은	2021-02-23	NULL	AA
S0003	백재용	2018-03-31	01077778888	DD

2. 테이블 생성하기

■ 테이블의 구조와 데이터 복사

- 형식

```
CREATE TABLE 테이블명 AS  
SELECT문;
```

■ [예제 8-4] 학생 테이블을 사용하여 휴학생 테이블을 생성하시오. 이때 데이터는 복사하지 않고, 구조만 복사하시오.

```
CREATE TABLE 휴학생 AS  
SELECT *  
FROM 학생  
WHERE 1 = 2;
```

▶ 실행 결과

학번	이름	성별	연락처	학과번호

3. GENERATED 컬럼

■ GENERATED 컬럼

- 테이블에 있는 컬럼을 기반으로 하여 계산된 값을 저장할 수 있는 컬럼
- 형식

컬럼 데이터타입 [GENERATED ALWAYS] AS 계산식 [VIRTUAL | STORED]

- VIRTUAL 방식

- ✓ 기본값
 - ✓ 데이터는 저장하지 않고 정의만 데이터 사전에 추가하는 방식

- STORED 방식

- ✓ 레코드가 삽입되거나 수정될 때마다 값이 계산되어 저장되기 때문에 추가 저장 공간이 필요하지만, SELECT 검색 시에는 계산 없이 바로 값을 읽어올 수 있음

3. GENERATED 컬럼

- [예제 8-5] 휘트니스센터의 회원을 관리하는 테이블을 만드시오. 이때 체질량 지수가 자동 계산되어 저장되도록 GENERATED 컬럼으로 설정하시오.

■ 회원 테이블 명세서

컬럼명	데이터타입	제약조건
아이디	기본문자형 20자	기본키
회원명	기본문자형 20자	
키	정수형	
몸무게	정수형	
체질량지수	실수형	몸무게(Kg) / POWER(?)^(cm), 2)

3. GENERATED 컬럼

■ [예제 8-5] 휘트니스센터의 회원을 관리하는 테이블을 만드시오. 이때 체질량
지수가 자동 계산되어 저장되도록 GENERATED 컬럼으로 설정하시오.

- 회원 테이블 생성

```
CREATE TABLE 회원
(
    아이디 VARCHAR(20) PRIMARY KEY
    ,회원명 VARCHAR(20)
    ,키 INT
    ,몸무게 INT
    ,체질량지수 DECIMAL(4,1) AS (몸무게 / POWER(키 / 100, 2)) STORED
);
```

- 레코드 삽입

```
INSERT INTO 회원(아이디, 회원명, 키, 몸무게)
VALUES('APPLE','김사과',178, 70);
```

▶ 실행 결과

아이디	회원명	키	몸무게	체질량지수
APPLE	김사과	178	70	22.1

Section 04

ALTER, DROP

1. ALTER

■ 컬럼 추가

- 형식

```
ALTER TABLE 테이블명 ADD COLUMN new_컬럼명 데이터타입;
```

■ [예제 8-6] 학생 테이블에 성별 컬럼을 추가하시오.

컬럼명	데이터타입
성별	고정문자형 1자

```
ALTER TABLE 학생 ADD 성별 CHAR(1);
```

▶ 실행결과

학번	이름	생일	연락처	학과번호	성별
S0001	이윤주	2020-01-30	01033334444	AA	Male
S0001	이승준	2021-02-23	01033334444	AA	Male
S0003	박재윤	2018-03-31	01077778888	DD	Male

1. ALTER

■ 컬럼 데이터타입 변경

- 형식

```
ALTER TABLE 테이블명 MODIFY COLUMN 컬럼명 new_데이터타입;
```

■ [예제 8-7] 학생 테이블에 성별 컬럼을 추가하시오.

컬럼명	데이터타입
성별	가변문자형 2자

```
ALTER TABLE 학생 MODIFY COLUMN 성별 VARCHAR(2);
```

1. ALTER

■ 컬럼명 변경

- 형식

```
ALTER TABLE 테이블명 CHANGE COLUMN old_컬럼명 new_컬럼명 데이터타입;
```

■ [예제 8-8] 학생 테이블에서 연락처 컬럼명을 휴대폰번호로 변경하시오

```
ALTER TABLE 학생 CHANGE COLUMN 연락처 휴대폰번호 VARCHAR(20);
```

▶ 실행결과

학번	이름	성별	휴대폰번호	학과번호	성적
50001	이윤주	2020-01-30	01033334444	AA	None

1. ALTER

■ 컬럼 삭제

- 형식

```
ALTER TABLE 테이블명 DROP COLUMN 컬럼명;
```

■ [예제 8-9] 학생 테이블에서 성별 컬럼을 삭제하시오

```
ALTER TABLE 학생 DROP COLUMN 성별;
```

▶ 실행결과

학번	이름	생일	휴대폰번호	학과번호
50001	이윤주	2020-01-30	01033334444	AA
50001	이승은	2021-02-23	01044445555	AA
50003	백자용	2018-03-31	01077778888	DD

1. ALTER

■ 테이블명 변경

- 형식

```
ALTER TABLE old_테이블명 RENAME new_테이블명;
```

■ [예제 8-10] 휴학생 테이블명을 졸업생 테이블명으로 변경하시오

```
ALTER TABLE 휴학생 RENAME 졸업생;
```

2. DROP

■ DROP

- 데이터베이스, 테이블 및 기타 여러 객체를 삭제할 때 사용함
- 형식

```
DROP DATABASE 데이터베이스명;
```

```
DROP TABLE 테이블명;
```

■ [예제 8-11] 학과 테이블과 학생 테이블을 삭제하시오

```
DROP TABLE 학과;
```

```
DROP TABLE 학생;
```

Section 05

제약조건

1. 제약조건의 개념

■ 제약조건

- 데이터베이스에는 무결한 데이터가 들어가야 함
- [예제 8-3]에서 보았듯이 테이블에 아무런 제약 사항을 두지 않으면 적합하지 않은 데이터가 저장되어 무결성 위배가 발생할 수 있음
- 이를 위해 테이블에 제약조건을 설정하여 데이터 무결성을 유지할 수 있음

■ 제약조건의 특징

- 제약조건은 데이터 사전에 저장됨
- 제약조건은 CREATE문으로 테이블을 생성할 때 지정할 수 있고, ALTER문으로 테이블의 구조를 변경할 때 지정할 수도 있음
- 제약조건은 고유한 이름을 붙여서 식별할 수 있음
- 한 컬럼에 여러 개의 제약조건을 설정할 수 있음

2. 제약조건의 설정

■ 제약조건의 설정

- 형식

```
CREATE TABLE 테이블명
(
    컬럼1 데이터타입 제약조건 • ①
    ,컬럼2 데이터타입 • ②
    ,제약조건(컬럼2) • ③
);
```

- 테이블 레벨 제약조건 설정

- ✓ ① 컬럼 레벨 제약조건 설정: 컬럼의 데이터타입 바로 다음에 기술함
- ✓ ②, ③ 테이블 레벨 제약조건 설정: 컬럼의 정의를 끝낸 후 제약조건을 별도로 지정

하나 더 알기 ▾

복합키와 대리키

레코드를 식별하기 위해서는 기본키를 설정해야 합니다. 기본키는 한 개의 컬럼 이상으로 구성이 될 수 있으며, 두 개 이상의 컬럼으로 생성된 기본키를 복합키(Composite Key)라고 합니다.

대리키(Surrogate key)는 기본키를 대체하는 키로 인공키(Artificial key)라고도 합니다. 기본키가 여러 개의 컬럼으로 구성되어야 하거나 기본키로 사용할 속성이 없을 경우에 대리키를 사용할 수 있습니다. 또는 데이터 길이가 너무 길거나 보안이 필요한 컬럼을 기본키로 사용해야 하는 경우 기본키를 대체할 수 있도록 일련번호와 같은 컬럼을 생성하여 기본키로 사용할 수 있습니다.



3. 제약조건의 종류

■ PRIMARY KEY 제약

- 기본키를 설정하는 제약조건
- 기본키는 레코드를 대표하는 키로 한 개 이상의 컬럼으로 구성됨
- 기본키는 테이블당 한 개여야 함
- 기본키를 생성하면 자동으로 인덱스가 생성됨
- 기본키는 NOT NULL이어야 하고, 유일한 값을 가져야 함
 - ✓ (NOT NULL 제약조건 + UNIQUE 제약조건)

■ NOT NULL 제약

- NOT NULL 제약조건이 설정된 컬럼에는 반드시 값을 넣어야 함
- NOT NULL 제약조건은 반드시 컬럼 레벨로 설정해야 함

3. 제약조건의 종류

■ UNIQUE 제약

- UNIQUE 제약조건이 설정된 컬럼에는 반드시 유일한 값을 넣어야 함
- 자동으로 인덱스가 생성됨

■ CHECK 제약

- CHECK 제약조건이 설정된 컬럼에는 설정된 조건에 맞는 값만 넣어야 함
- 조건으로는 특정 값이나 범위, 특정 패턴의 숫자나 문자 값 등을 설정할 수 있음

■ DEFAULT 제약

- 값을 넣지 않을 경우 지정한 값이 자동으로 들어감

■ FOREIGN KEY 제약

- 외래키를 설정하는 제약조건
- 한 테이블의 외래키는 참조하는 테이블의 기본키이거나 NULL이어야 함
- 외래키의 컬럼과 참조하는 테이블의 기본키 컬럼의 데이터타입과 크기는 서로 동일해야 함

3. 제약조건의 종류

하나 더 알기

개체 무결성과 참조 무결성

개체 무결성과 참조 무결성은 데이터 무결성을 보장하는 중요한 규칙으로 이를 지키지 않으면 데이터의 정확성과 일관성이 손상될 수 있으므로 무결성 규칙을 엄격하게 준수해야 합니다.

■ 개체 무결성

개체 무결성(Entity Integrity)은 기본키와 관련이 있습니다. 개체 무결성은 기본키 컬럼에 중복된 값이나 NULL 값을 허용하지 않는 원칙을 나타냅니다.

■ 참조 무결성

참조 무결성(Referential Integrity)은 외래키와 관련이 있습니다. 특정 테이블의 외래키는 다른 테이블의 기본키를 참조함으로써 관계가 항상 일관성 있게 유지되어야 함을 의미합니다.



3. 제약조건의 종류



테이블에 기본키를 꼭 지정해야 하나요?



A

기본키는 각 레코드를 고유하게 식별하는 역할을 합니다. 기본키를 지정하면 정확하고 빠르게 데이터를 검색하고 관리할 수 있으며, 레코드 간에 관계를 설정하거나 데이터를 조작할 때도 중요한 역할을 합니다. 또한 기본키는 중복을 방지하고 데이터의 무결성을 보장하는 데에도 도움이 됩니다. 기본키를 지정하지 않으면 데이터베이스 시스템이 중복 레코드를 허용할 수 있는데, 이는 데이터 관리와 정확성에 대해 문제를 초래할 수 있습니다. 따라서 특정 상황을 제외하고는 기본키를 지정하여 데이터의 고유성을 보장하고 데이터베이스의 성능을 향상시키는 것이 좋습니다.



3. 제약조건의 종류

■ [예제 8-12] 제약조건을 추가하여 학과 테이블을 다시 생성하시오

■ 학과 테이블 명세서(제약조건 추가)

컬럼명	데이터타입	제약조건
학과번호	고정문자형 2자	기본키
학과명	가변문자형 20자	필수 입력
학과장명	가변문자형 20자	

3. 제약조건의 종류

■ [예제 8-12] 제약조건을 추가하여 학과 테이블을 다시 생성하시오

- 방법1: 컬럼 레벨의 제약조건 설정

```
CREATE TABLE 학과
(
    학과번호 CHAR(2) PRIMARY KEY
    ,학과명 VARCHAR(20) NOT NULL
    ,학과장명 VARCHAR(20)
);
```

- 방법2: 테이블 레벨로 제약조건 설정

```
CREATE TABLE 학과
(
    학과번호 CHAR(2)
    ,학과명 VARCHAR(20) NOT NULL •————②
    ,학과장명 VARCHAR(20)
    ,PRIMARY KEY(학과번호) •————①
);
```

3. 제약조건의 종류

■ [예제 8-13] 제약조건을 추가하여 학생 테이블을 다시 생성하시오

■ 학생 테이블 명세서(제약조건 추가)

컬럼명	데이터타입	제약조건
학번	고정문자형 5자	기본키
이름	가변문자형 20자	필수 입력
생일	날짜형	필수 입력
연락처	가변문자형 20자	유일한 값만 입력 가능
학과번호	고정문자형 2자	학과 테이블의 학과번호 참조
성별	고정문자형 1자	남 또는 여만 입력 가능
등록일	날짜형	오늘 날짜 자동 입력

3. 제약조건의 종류

■ [예제 8-13] 제약조건을 추가하여 학생 테이블을 다시 생성하시오

- 방법1: 컬럼 레벨의 제약조건 설정

```
CREATE TABLE 학생
(
    학번 CHAR(5) PRIMARY KEY
    ,이름 VARCHAR(20) NOT NULL
    ,생일 DATE NOT NULL
    ,연락처 VARCHAR(20) UNIQUE
    ,학과번호 CHAR(2) REFERENCES 학과(학과번호)
    ,성별 CHAR(1) CHECK(성별 IN ('남','여'))
    ,등록일 DATE DEFAULT(CURDATE())
    ,FOREIGN KEY(학과번호) REFERENCES 학과(학과번호)
);
```

3. 제약조건의 종류

■ [예제 8-13] 제약조건을 추가하여 학생 테이블을 다시 생성하시오

- 방법2: 테이블 레벨로 제약조건 설정

```
CREATE TABLE 학생
(
    학번 CHAR(5)
    ,이름 VARCHAR(20) NOT NULL
    ,생일 DATE NOT NULL
    ,연락처 VARCHAR(20)
    ,학과번호 CHAR(2)
    ,성별 CHAR(1)
    ,등록일 DATE DEFAULT(CURDATE())
    ,PRIMARY KEY(학번)
    ,UNIQUE(연락처)
    ,CHECK(성별 IN ('남','여'))
    ,FOREIGN KEY (학과번호) REFERENCES 학과(학과번호)
);
```

3. 제약조건의 종류

■ [예제 8-14] 제약조건을 추가하여 과목 테이블을 생성하시오

■ 과목 테이블 명세서

컬럼명	데이터타입	제약조건
과목번호	고정문자형 5자	기본키
과목명	가변문자형 20자	필수 입력
학점	정수형	필수 입력, 2학점부터 4학점까지만 입력 가능
구분	가변문자형 20자	전공, 교양, 일반만 입력 가능

```
CREATE TABLE 과목
(
    과목번호 CHAR(5) PRIMARY KEY
    ,과목명 VARCHAR(20) NOT NULL
    ,학점 INT NOT NULL CHECK(학점 BETWEEN 2 AND 4) •————①
    ,구분 VARCHAR(20) CHECK(구분 IN ('전공','교양','일반'))
);
```

3. 제약조건의 종류

■ [예제 8-15] 제약조건을 추가하여 수강_1 테이블을 생성하시오. 기본키는 수강년도, 수강학기, 학번, 과목번호를 모두 포함하시오

■ 수강_1 테이블 명세서

컬럼명	데이터타입	제약조건
수강년도	고정문자형 4자	필수 입력
수강학기	기변문자형 20자	필수 입력. 1학기, 2학기, 여름학기, 겨울학기만 입력 가능
학번	고정문자형 5자	필수 입력. 학생 테이블의 학번 참조
과목번호	고정문자형 5자	필수 입력. 과목 테이블의 과목번호 참조
성적	실수형	0부터 4.5까지 입력 가능

```
CREATE TABLE 수강_1
(
    수강년도 CHAR(4) NOT NULL
    ,수강학기 VARCHAR(20) NOT NULL CHECK(수강학기 IN ('1학기','2학기','여름학기','겨울학기'))
    ,학번 CHAR(5) NOT NULL
    ,과목번호 CHAR(5) NOT NULL
    ,성적 NUMERIC(3,1) CHECK(성적 BETWEEN 0 AND 4.5)
    ,PRIMARY KEY(수강년도, 수강학기, 학번, 과목번호) •————①
    ,FOREIGN KEY (학번) REFERENCES 학생(학번)
    ,FOREIGN KEY (과목번호) REFERENCES 과목(과목번호)
);
```

3. 제약조건의 종류

- [예제 8-16] 수강번호라는 대리키를 기본키로 하는 제약조건을 추가하여 수강_2 테이블을 생성하시오. 수강번호는 일련번호로 생성하시오.

```
CREATE TABLE 수강_2
(
    수강번호 INT PRIMARY KEY AUTO_INCREMENT
    ,수강년도 CHAR(4) NOT NULL
    ,수강학기 VARCHAR(20) NOT NULL CHECK(수강학기 IN ('1학기','2학기','여름학기','겨울학기'))
    ,학번 CHAR(5) NOT NULL
    ,과목번호 CHAR(5) NOT NULL
    ,성적 NUMERIC(3,1) CHECK(성적 BETWEEN 0 AND 4.5)
    ,FOREIGN KEY (학번) REFERENCES 학생(학번)
    ,FOREIGN KEY (과목번호) REFERENCES 과목(과목번호)
);
```

3. 제약조건의 종류

■ [예제 8-17] 학과 테이블에 다음과 같이 레코드 3건을 추가하시오. 오류가 발생한다면 이유가 무엇인지 생각해봅니다.

```
INSERT INTO 학과  
VALUES ('AA', '컴퓨터공학과', '배경민');
```

```
INSERT INTO 학과  
VALUES ('AA', '소프트웨어학과', '김남준'); ────────── ① 오류 발생
```

```
INSERT INTO 학과  
VALUES ('CC', '디자인융합학과', '박선영');
```

3. 제약조건의 종류

■ [예제 8-17] 학과 테이블에 다음과 같이 레코드 3건을 추가하시오. 오류가 발생한다면 이유가 무엇인지 생각해봅니다.

```
INSERT INTO 학과  
VALUES ('AA', '컴퓨터공학과', '배경민');
```

```
INSERT INTO 학과  
VALUES ('AA', '소프트웨어학과', '김남준'); •———— ① 오류 발생
```

```
INSERT INTO 학과  
VALUES ('CC', '디자인융합학과', '박선영');
```

- 문장①에서 오류 발생

Error Code: 1062. Duplicate entry 'AA' for key '학과.PRIMARY'

▼ 해결

```
INSERT INTO 학과  
VALUES ('BB', '소프트웨어학과', '김남준');
```

▶ 실행 결과

학과번호	학과명	학과장명
AA	컴퓨터공학과	배경민
BB	소프트웨어학과	김남준
CC	디자인융합학과	박선영

3. 제약조건의 종류

- [예제 8-18] 학생 테이블에 다음과 같이 레코드 3건을 추가하시오. 오류가 발생한다면 이유가 무엇인지 생각해봅니다.

```
INSERT INTO 학생(학번, 이름, 생일, 학과번호)  
VALUES ('S0001','이윤주','2020-01-30', 'AA');
```

```
INSERT INTO 학생(이름, 생일, 학과번호)  
VALUES ('이승은','2021-02-23', 'AA');
```

① 오류 발생

```
INSERT INTO 학생(학번, 이름, 생일, 학과번호)  
VALUES ('S0003','백재용','2018-03-31','DD');
```

② 오류 발생

3. 제약조건의 종류

■ [예제 8-18] 학생 테이블에 다음과 같이 레코드 3건을 추가하시오. 오류가 발생한다면 이유가 무엇인지 생각해봅니다.

```
INSERT INTO 학생(학번, 이름, 생일, 학과번호)
VALUES ('S0001', '이윤주', '2020-01-30', 'AA');
```

```
INSERT INTO 학생(이름, 생일, 학과번호)
VALUES ('이승은', '2021-02-23', 'AA');
```

① 오류 발생

```
INSERT INTO 학생(학번, 이름, 생일, 학과번호)
VALUES ('S0003', '백재용', '2018-03-31', 'DD');
```

② 오류 발생

- 문장①에서 오류 발생

```
Error Code: 1364. Field '학번' doesn't have a default value
```



해결

```
INSERT INTO 학생(학번, 이름, 생일, 학과번호)
VALUES ('S0002', '이승은', '2020-01-30', 'AA');
```

3. 제약조건의 종류

■ [예제 8-18] 학생 테이블에 다음과 같이 레코드 3건을 추가하시오. 오류가 발생한다면 이유가 무엇인지 생각해봅니다.

```
INSERT INTO 학생(학번, 이름, 생일, 학과번호)
VALUES ('S0001', '이윤주', '2020-01-30', 'AA');
```

```
INSERT INTO 학생(이름, 생일, 학과번호)
VALUES ('이승은', '2021-02-23', 'AA');
```

① 오류 발생

```
INSERT INTO 학생(학번, 이름, 생일, 학과번호)
VALUES ('S0003', '백재용', '2018-03-31', 'DD');
```

② 오류 발생

- 문장②에서 오류 발생

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails('한빛학사','학생',CONSTRAINT '학생_ibfk_1'FOREIGN KEY ('학과번호') REFERENCES '학과' ('학과번호'))



해결

```
INSERT INTO 학생(학번, 이름, 생일, 학과번호)
VALUES ('S0003', '백재용', '2018-03-31', 'CC');
```

3. 제약조건의 종류

- [예제 8-19] 과목 테이블에 다음과 같이 레코드 3건을 추가하시오. 오류가 발생한다면 이유가 무엇인지 생각해봅니다.

```
INSERT INTO 과목(과목번호, 과목명, 구분)
```

```
VALUES ('C0001','데이터베이스실습', '전공');
```

① 오류 발생

```
INSERT INTO 과목(과목번호, 과목명, 구분, 학점)
```

```
VALUES ('C0002','데이터베이스 설계와 구축', '전공', 5);
```

② 오류 발생

```
INSERT INTO 과목(과목번호, 과목명, 구분, 학점)
```

```
VALUES ('C0003','데이터 분석', '전공', 3);
```

3. 제약조건의 종류

■ [예제 8-19] 과목 테이블에 다음과 같이 레코드 3건을 추가하시오. 오류가 발생한다면 이유가 무엇인지 생각해봅니다.

```
INSERT INTO 과목(과목번호, 과목명, 구분)
```

```
VALUES ('C0001','데이터베이스실습', '전공');
```

① 오류 발생

```
INSERT INTO 과목(과목번호, 과목명, 구분, 학점)
```

```
VALUES ('C0002','데이터베이스 설계와 구축', '전공', 5);
```

② 오류 발생

```
INSERT INTO 과목(과목번호, 과목명, 구분, 학점)
```

```
VALUES ('C0003','데이터 분석', '전공', 3);
```

- 문장①에서 오류 발생

Error Code: 1364. Field '학점' doesn't have a default value



해결

```
INSERT INTO 과목(과목번호, 과목명, 구분, 학점)
```

```
VALUES ('C0001','데이터베이스실습', '전공', 3);
```

3. 제약조건의 종류

■ [예제 8-19] 과목 테이블에 다음과 같이 레코드 3건을 추가하시오. 오류가 발생한다면 이유가 무엇인지 생각해봅니다.

```
INSERT INTO 과목(과목번호, 과목명, 구분)
```

```
VALUES ('C0001','데이터베이스실습', '전공');
```

① 오류 발생

```
INSERT INTO 과목(과목번호, 과목명, 구분, 학점)
```

```
VALUES ('C0002','데이터베이스 설계와 구축', '전공', 5);
```

② 오류 발생

```
INSERT INTO 과목(과목번호, 과목명, 구분, 학점)
```

```
VALUES ('C0003','데이터 분석', '전공', 3);
```

- 문장②에서 오류 발생

Error Code: 3819. Check constraint '과목_chk_1'is violated.



해결

```
INSERT INTO 과목(과목번호, 과목명, 구분, 학점)
```

```
VALUES ('C0002','데이터베이스 설계와 구축', '전공', 3);
```

▶ 실행결과

과목번호	과목명	학점	구분
C0001	데이터베이스실습	3	전공
C0002	데이터베이스 설계와 구축	3	전공
C0003	데이터 분석	3	전공

3. 제약조건의 종류

■ [예제 8-20] 수강_1 테이블에 다음과 같이 레코드 4건을 추가하시오. 오류가 발생한다면 이유가 무엇인지 생각해봅니다.

```
INSERT INTO 수강_1(수강년도, 수강학기, 학번, 과목번호, 성적)
VALUES('2023', '1학기', 'S0001', 'C0001', 4.3);
```

```
INSERT INTO 수강_1(수강년도, 수강학기, 학번, 과목번호, 성적)
VALUES('2023', '1학기', 'S0001', 'C0001', 4.5);
```

① 오류 발생

```
INSERT INTO 수강_1(수강년도, 수강학기, 학번, 과목번호, 성적)
VALUES('2023', '1학기', 'S0001', 'C0002', 4.6);
```

② 오류 발생

```
INSERT INTO 수강_1(수강년도, 수강학기, 학번, 과목번호, 성적)
VALUES('2023', '1학기', 'S0002', 'C0009', 4.3);
```

③ 오류 발생

3. 제약조건의 종류

■ [예제 8-20] 수강_1 테이블에 다음과 같이 레코드 4건을 추가하시오. 오류가 발생한다면 이유가 무엇인지 생각해봅니다.

- 문장①에서 오류 발생

```
Error Code: 1062. Duplicate entry '2023-1학기-S0001-C0001' for key '수강_1.PRIMARY'
```

- 문장②에서 오류 발생

```
Error Code: 3819. Check constraint '수강_1_chk_2' is violated.
```



해결

```
INSERT INTO 수강_1(수강년도, 수강학기, 학번, 과목번호, 성적)  
VALUES('2023','1학기','S0001','C0002',4.4);
```

- 문장③에서 오류 발생

```
Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('한빛  
학사`.`수강_1`, CONSTRAINT `수강_1_ibfk_2` FOREIGN KEY (`과목번호`) REFERENCES `과목` (`과  
목번호`))
```



해결

▶ 실행결과

```
INSERT INTO 수강_1(수강년도, 수강학기, 학번, 과목번호, 성적)  
VALUES('2023','1학기','S0002','C0002',4.3);
```

수강년도	수강학기	학번	과목번호	성적
2023	1학기	S0001	C0001	4.3
2023	1학기	S0001	C0002	4.4
2023	1학기	S0002	C0002	4.3

3. 제약조건의 종류

- [예제 8-21] 수강_1에서 넣은 것과 동일하게 수강_2 테이블에 2개의 레코드를 추가해보시오.

```
INSERT INTO 수강_2(수강년도, 수강학기, 학번, 과목번호, 성적)
VALUES('2023','1학기','S0001','C0001',4.3);
```

```
INSERT INTO 수강_2(수강년도, 수강학기, 학번, 과목번호, 성적)
VALUES('2023','1학기','S0001','C0001',4.5);
```

▶ 실행결과

수강번호	수강년도	수강학기	학번	과목번호	성적
1	2023	1학기	S0001	C0001	4.3
2	2023	1학기	S0001	C0001	4.5

4. 제약조건의 추가, 삭제

■ 제약조건의 추가

- 형식

```
ALTER TABLE 테이블명 ADD CONSTRAINT 제약조건(컬럼명);
```

■ [예제 8-22] 학생 테이블의 학번 컬럼에 CHECK 제약조건을 추가하시오.

- 제약조건: 모든 학번은 'S'로 시작한다.

```
ALTER TABLE 학생 ADD CONSTRAINT CHECK(학번 LIKE 'S%');
```

■ [예제 8-23] 학생 테이블에 설정되어 있는 제약조건 명세를 확인하시오.

```
SELECT *  
FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS  
WHERE CONSTRAINT_SCHEMA = '한빛학사'  
AND TABLE_NAME = '학생';
```

▶ 실행결과

CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE	ENFORCED
def	한빛학사	PRIMARY	한빛학사	학생	PRIMARY KEY	YES
def	한빛학사	연락처	한빛학사	학생	UNIQUE	YES
def	한빛학사	학생_ibfk_1	한빛학사	학생	FOREIGN KEY	YES
def	한빛학사	학생_chk_1	한빛학사	학생	CHECK	YES
def	한빛학사	학생_chk_2	한빛학사	학생	CHECK	YES

4. 제약조건의 추가, 삭제

■ 제약조건의 삭제

- 형식

```
ALTER TABLE 테이블명 DROP CONSTRAINT 제약조건명;
```

■ [예제 8-24] 학생 테이블에 설정되어 있는 제약조건 중 연락처에 설정되어 있는 제약조건을 삭제하시오.

```
ALTER TABLE 학생 DROP CONSTRAINT 연락처;
```

■ [예제 8-25] 성별 컬럼에 설정되어 있는 CHECK 제약조건을 삭제하시오.

- 학생 테이블에 걸려있는 CHECK 제약조건 2개를 삭제하고 학번에 설정했던 CHECK제약조건은 다시 설정

```
ALTER TABLE 학생 DROP CONSTRAINT 학생_chk_1;
```

```
ALTER TABLE 학생 DROP CONSTRAINT 학생_chk_2;
```

```
ALTER TABLE 학생 ADD CHECK (학번 LIKE 'S%');
```

▶ 실행결과

CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE	ENFORCED
def	한빛학사	PRIMARY	한빛학사	학생	PRIMARY KEY	YES
def	한빛학사	학생_ibfk_1	한빛학사	학생	FOREIGN KEY	YES
def	한빛학사	학생_chk_1	한빛학사	학생	CHECK	YES

5. 제약조건명의 지정

■ 제약조건명의 지정

- 제약조건에는 고유한 이름을 지정하여 관리할 수 있음
- 형식

컬럼 레벨:

컬럼명 데이터타입 [CONSTRAINT 제약조건명] 제약조건

테이블 레벨:

[CONSTRAINT 제약조건명] 제약조건

5. 제약조건명의 지정

■ [예제 8-26] [예제 8-13]에 있는 학생 테이블의 구조를 사용하여 학생_2 테이블을 생성하시오. 이때 제약조건명을 지정하여 생성하시오.

- 학생_2 테이블 생성

```
CREATE TABLE 학생_2
(
    학번 CHAR(5)
    ,이름 VARCHAR(20) NOT NULL
    ,생일 DATE NOT NULL
    ,연락처 VARCHAR(20)
    ,학과번호 CHAR(2)
    ,성별 CHAR(1)
    ,등록일 DATE DEFAULT(CURDATE())
    ,PRIMARY KEY(학번)
    ,CONSTRAINT UK_학생2_연락처 UNIQUE(연락처)
    ,CONSTRAINT CK_학생2_성별 CHECK(성별 IN ('남','여'))
    ,CONSTRAINT FK_학생2_학과번호 FOREIGN KEY (학과번호) REFERENCES 학과(학과번호)
);
```

5. 제약조건명의 지정

■ [예제 8-26] [예제 8-13]에 있는 학생 테이블의 구조를 사용하여 학생_2 테이블을 생성하시오. 이때 제약조건명을 지정하여 생성하시오.

- 제약조건 명세 확인

```
SELECT *
FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
WHERE CONSTRAINT_SCHEMA = '한빛학사'
AND TABLE_NAME = '학생_2';
```

▶ 실행 결과

CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE	ENFORCED
def	한빛학사	PRIMARY	한빛학사	학생_2	PRIMARY KEY	YES
def	한빛학사	UK_학생2_연락처	한빛학사	학생_2	UNIQUE	YES
def	한빛학사	FK_학생2_학과번호	한빛학사	학생_2	FOREIGN KEY	YES
def	한빛학사	CK_학생2_성별	한빛학사	학생_2	CHECK	YES

6. 외래키 제약조건의 옵션

■ 외래키 제약조건의 옵션

표 8-5 외래키 제약조건 옵션

	ON DELETE	ON UPDATE
CASCADE	부모 레코드 삭제 시 자식 레코드도 연쇄적으로 삭제된다.	부모 레코드의 기본키 수정 시 자식 레코드의 외래키 값도 연쇄적으로 수정된다.
SET NULL	부모 레코드 삭제 시 자식 레코드의 외래키 값이 NULL로 변경된다.	부모 레코드의 기본키 수정 시 자식 레코드의 외래키 값이 NULL로 변경된다.
SET DEFAULT	부모 레코드 삭제 시 자식 레코드의 외래키 값이 기본값으로 변경된다.	부모 레코드의 기본키 수정 시 자식 레코드의 외래키 값이 기본값으로 변경된다.
NO ACTION(기본값)	자식 레코드가 있으면 부모 레코드를 삭제할 수 없다.	자식 레코드가 있으면 부모 레코드의 기본키 값을 수정할 수 없다.

6. 외래키 제약조건의 옵션

- [예제 8-27] 수강평가 테이블을 생성하시오. 이때 평가순번은 자동 번호로 생성합니다. 또한 과목 테이블에서 레코드를 삭제하면 수강평가 테이블에서도 해당 과목에 대한 평가 레코드가 삭제되도록 옵션을 설정하시오.

▶ 수강평가 테이블 명세서

컬럼명	데이터타입	제약조건 및 조건
평가순번	정수형	기본키, 자동번호
학번	고정문자형 5자	학생 테이블의 학번 참조
과목번호	고정문자형 5자	과목 테이블의 과목번호 참조
평점	정수형	0부터 5까지 입력 가능
과목평가	기변문자형 500자	
평가일시	날짜시간형	현재 일시 자동 등록

6. 외래키 제약조건의 옵션

■ [예제 8-27] 수강평가 테이블을 생성하시오. 이때 평가순번은 자동 번호로 생성합니다. 또한 과목 테이블에서 레코드를 삭제하면 수강평가 테이블에서도 해당 과목에 대한 평가 레코드가 삭제되도록 옵션을 설정하시오.

- 수강평가 테이블 생성

```
CREATE TABLE 수강평가
(
    평가순번 INT PRIMARY KEY AUTO_INCREMENT
    ,학번 CHAR(5) NOT NULL
    ,과목번호 CHAR(5) NOT NULL
    ,평점 INT CHECK(평점 BETWEEN 0 AND 5)
    ,과목평가 VARCHAR(500)
    ,평가일시 DATETIME DEFAULT CURRENT_TIMESTAMP
    ,FOREIGN KEY (학번) REFERENCES 학생(학번)
    ,FOREIGN KEY (과목번호) REFERENCES 과목(과목번호) ON DELETE CASCADE
);
```

6. 외래키 제약조건의 옵션

- [예제 8-28] 수강평가 테이블에 4건의 레코드를 추가하시오.

```
INSERT INTO 수강평가(학번, 과목번호, 평점, 과목평가)
VALUES('S0001','C0001',5,'SQL학습에 도움이 되었습니다.')
      ,('S0001','C0003',5,'SQL 활용을 배워서 좋았습니다.')
      ,('S0002','C0003',5,'데이터 분석에 관심이 생겼습니다.')
      ,('S0003','C0003',5,'머신러닝과 시각화 부분이 유용했습니다.');
```

▶ 실행결과

평가순번	학번	과목번호	평점	과목평가	평가일시
1	S0001	C0001	5	SQL학습에 도움이 되었습니다.	2023-12-27 11:25:38
2	S0001	C0003	5	SQL 활용을 배워서 좋았습니다.	2023-12-27 11:25:38
3	S0002	C0003	5	데이터 분석에 관심이 생겼습니다.	2023-12-27 11:25:38
4	S0003	C0003	5	머신러닝과 시각화 부분이 유용했습니다.	2023-12-27 11:25:38

6. 외래키 제약조건의 옵션

■ [예제 8-29] 과목 테이블에서 'C0003' 과목을 삭제하시오. 그리고 과목 테이블과 수강평가 테이블에서 결과를 확인해봅니다.

```
DELETE FROM 과목 WHERE 과목번호 = 'C0003';
```

```
SELECT * FROM 과목;  
SELECT * FROM 수강평가;
```

▶ 실행 결과

• 과목 테이블

과목 번호	과목명	학점	구분
C0001	데이터베이스실습	3	전공
C0002	데이터베이스 설계와 구축	3	전공

• 수강평가 테이블

평가순번	학번	과목번호	평점	과목평가	평가일시
1	S0001	C0001	5	SQL학습에 도움이 되었습니다.	2023-12-27 11:25:38

6. 외래키 제약조건의 옵션

- [예제 8-30] 과목 테이블에서 'C0001' 과목을 삭제하시오. 오류가 난다면 이유를 생각해봅니다.

```
DELETE FROM 과목 WHERE 과목번호 = 'C0001';
```

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('한빛학사'. '수강_1', CONSTRAINT '수강_1_ibfk_2' FOREIGN KEY ('과목번호') REFERENCES '과목' ('과목번호'))

점검문제

점검문제

문제 1

제품 테이블의 재고 컬럼에 CHECK 제약조건을 추가하시오.

- 제약조건: 재고는 0보다 크거나 같아야 합니다.

문제 2

제품 테이블에 재고금액 컬럼을 추가하시오. 이때 재고금액은 '단가 * 재고'가 자동 계산되어 저장되도록 합니다.

문제 3

제품 테이블에서 제품 레코드를 삭제하면 주문세부 테이블에 있는 관련 레코드도 함께 삭제되도록 주문세부 테이블의 제품번호 컬럼에 외래키 제약조건과 옵션을 설정하시오.

Chapter 09

뷰와 인덱스



목차

1. 뷰
2. 뷰를 통한 데이터 삽입
3. 인덱스
4. 옵티마이저

학습목표

- 뷰의 개념과 사용법을 이해하고 뷰를 생성할 수 있습니다.
- 인덱스의 개념과 B-트리 인덱스를 이해할 수 있습니다.
- 옵티마이저의 개념을 이해하고 EXPLAIN, EXPLAIN ANALYZE를 이해할 수 있습니다.

Preview



Section 01

뷰

1. 뷰의 개념

■ 뷰(view)

- 한 개 이상의 테이블을 기반으로 생성된 가상의 테이블
- 뷰는 실제 데이터를 저장하지 않으며, 쿼리 실행 시점에 생성된 쿼리 결과를 가상의 테이블로 만들어서 제공함
- 다시 말해 뷰는 물리적으로 데이터를 저장하는 것이 아니라 SELECT문을 저장한 형태로서 스토어드 쿼리(Stored Queries)라고도 함
- 뷰 실행 시에는 테이블을 조회한 것과 유사한 형태로 데이터를 확인할 수 있음

1. 뷰의 개념

■ 뷰의 장점

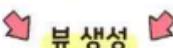
- 데이터 중복성 최소화
- 데이터 보안성
- 간편한 데이터 접근
- 데이터 가상화

고객 테이블

고객번호	고객회사명	도시	전화번호
ACCO01	금성仪表	서울특별시	(02)578-0984
ADMT02	알마이 식당	서울특별시	(02)105-0198
ALY03	시강그룹	광명시	(031)433-0145
ANAS04	오리간부회	서울특별시	(02)123-1245
ANAS05	노성종합나들	구리시	(034)759-2081
ANAS06	서보컴퓨터	서울특별시	(02)211-2221
ATM07	다정인하나세계	서울특별시	(02)561-0889
BS08	개인인력나눔	서울특별시	(02)011-1234

주문 테이블

주문번호	고객번호	주문일	포장일	발송일
2020-03-13	MPTD	2020-03-13	2020-04-24	2020-03-18
2020-03-16	MARNA	2020-03-16	2020-04-13	2020-03-29
2020-03-17	CTEV	2020-03-17	2020-04-14	2020-03-23
2020-03-17	PROSU	2020-03-17	2020-04-14	2020-03-19
2020-03-18	MARNA	2020-03-18	2020-04-21	2020-03-24
2020-03-19	OPICH	2020-03-19	2020-04-16	2020-03-31
2020-03-20	CSLR	2020-03-20	2020-04-17	2020-03-23
2020-03-23	BSHE	2020-03-23	2020-04-26	2020-03-26



view_고객_주문

고객번호	고객회사명	도시	전화번호	주문번호	주문일	포장일	발송일
MSPT01	포스미	부천시	(032)110-1238	H0249	2020-03-13	2020-04-24	2020-03-18
MARNA	청운유물	제주시	(044)3110-2181	H0250	2020-03-16	2020-04-13	2020-03-20
CTEV01	한국탁구프트 우연	한국탁구역시	(032)568-1984	H0251	2020-03-16	2020-04-13	2020-03-23
PROSU	활란타운리레이스	서울특별시	(02)904-1984	H0252	2020-03-16	2020-04-14	2020-03-19
MARNA	청운유물	제주시	(044)3110-2181	H0253	2020-03-18	2020-04-01	2020-03-24



그림 9-1 뷰의 개념

2. 뷰 생성 및 변경하기

■ 뷰 생성

- 형식1: OR REPLACE 사용

```
CREATE [OR REPLACE] VIEW 뷰명  
AS  
SELECT문;
```

- 형식2: ALTER 사용

```
ALTER VIEW 뷰명  
AS  
SELECT문;
```

2. 뷰 생성 및 변경하기

- [예제 9-1] 한빛무역 데이터베이스에서 사원 테이블을 사용하여 사원의 이름, 집전화, 입사일, 주소를 보이는 뷰를 작성하시오.

- 뷰명: view_사원

- ✓ 뷰를 생성할 때 컬럼명에 별명을 붙일 수 있으며, 뷰를 사용할 때에는 컬럼명에 붙인 별명을 사용해야 함

```
CREATE OR REPLACE VIEW view_사원
AS
SELECT 이름
    ,집전화 AS 전화번호
    ,입사일
    ,주소
FROM 사원;
```

- ✓ 컬럼 또는 컬럼의 별명을 뷰 이름 옆에 나열할 수도 있음

```
CREATE OR REPLACE VIEW view_사원(이름, 전화번호, 입사일, 주소)
AS
SELECT 이름
    ,집전화
    ,입사일
    ,주소
FROM 사원;
```

2. 뷰 생성 및 변경하기

- [예제 9-1] 한빛무역 데이터베이스에서 사원 테이블을 사용하여 사원의 이름, 집전화, 입사일, 주소를 보이는 뷰를 작성하시오.

- 뷰명: view_사원

- ✓ 뷰를 생성한 이후에는 뷰를 테이블인 것처럼 조회할 수 있음

```
SELECT *  
FROM view_사원;
```

▶ 실행결과

이름	전화번호	입사일	주소
이소미	(02)578-8988	2019-04-13	강남구 역삼동 36-8
배재용	(032)69-0136	2019-01-01	원미동 16-11
유대현	(062)73-0256	2019-03-14	광산구 송정동 100-11
최소연	(051)587-4783	2019-04-15	중구 중앙동 57-14

2. 뷰 생성 및 변경하기

■ [예제 9-2] 제품 테이블, 주문세부 테이블을 조인하여 제품명과 주문수량합을 보이는 뷰를 작성하시오

- 뷰명: view_제품별주문수량합

- ✓ 조인이나 서브쿼리 등 여러 테이블을 사용한 쿼리문으로 뷰를 생성 가능

```
CREATE OR REPLACE VIEW view_제품별주문수량합
AS
SELECT 제품명
      ,SUM(주문수량) AS 주문수량합
  FROM 제품
 INNER JOIN 주문세부
    ON 제품.제품번호 = 주문세부.제품번호
 GROUP BY 제품명;
```

```
SELECT *
  FROM view_제품별주문수량합;
```

▶ 실행결과

제품명	주문수량합
컨 100% 오현지 주스	828
컨 100% 레몬 주스	1033
컨 제리 시럽	318

2. 뷰 생성 및 변경하기

■ [예제 9-3] '여' 사원에 대하여 사원의 이름, 집전화, 입사일, 주소, 성별을 보이는 뷰를 작성하시오

- 뷰명: view_사원_여

```
CREATE OR REPLACE VIEW view_사원_여
AS
SELECT 이름
    ,집전화 AS 전화번호
    ,입사일
    ,주소
    ,성별
FROM 사원
WHERE 성별 = '여';
```

```
SELECT *
FROM view_사원_여;
```

▶ 실행결과

이름	전화번호	입사일	주소	성별
이소미	(02)578-8988	2019-04-13	강남구 역삼동 36-8	여
최소민	(051)587-4783	2019-04-15	중구 풍정동 57-14	여
선화라	(0652)983-1985	2019-02-16	온진구 고황동 116	여
유가율	(053)465-1248	2019-10-29	남구 대명동 19-7	여

3. 뷰 조회하기

■ 뷰 조회하기

- 형식

```
SELECT *  
FROM 뷰명;
```

- [예제 9-4] [예제 9-3]에서 생성한 'view_사원_여' 뷰로 전화번호에 '88'이 들어간 사원의 정보를 검색하시오.

```
SELECT *  
FROM view_사원_여  
WHERE 전화번호 LIKE '%88%';
```

▶ 실행결과

이름	전화번호	입사일	주소	성별
이소미	(02)578-8988	2019-04-13	강남구 역삼동 36-8	여

- [예제 9-5] [예제 9-2]에서 생성한 'view_제품별주문수량합' 뷰로 주문수량합이 1,200개 이상인 레코드를 검색하시오.

```
SELECT *  
FROM view_제품별주문수량합  
WHERE 주문수량합 >= 1200;
```

▶ 실행결과

제품명	주문수량합
볼루 바닐라 아이스크림	1397
대왕 옥수수 가루	1263
그린 포장 치즈	1496
그린 파예살 치즈	1575

4. 뷰 메타 정보 확인하기

■ 뷰 메타 정보 확인하기

- 형식1: INFORMATION_SCHEMA.VIEWS 사용

```
SELECT *  
FROM INFORMATION_SCHEMA.VIEWS  
WHERE TABLE_NAME = '뷰명';
```

- 형식2: SHOW CREATE VIEW 사용

```
SHOW CREATE VIEW 뷰명;
```

4. 뷰 메타 정보 확인하기

- [예제 9-6] 'view_사원' 뷰의 메타 정보를 확인하시오.

```
SELECT *
FROM INFORMATION_SCHEMA.VIEWS
WHERE TABLE_NAME = 'view_사원';
```

▶ 실행결과

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	VIEW_DEFINITION	CHECK_OPTION	IS_UPDATABLE	DEFINER	SECURITY_TYPE	CHARACTER_SET_CLIENT	COLLATION_CONNECTION
def	동일구역	view_사원	select `동일구역`.`사원`.`이름` AS `이름`, ...	NONE	YES	root@localhost	DEFINER	utf8mb4	utf8mb4_0900_ai_ci

```
SHOW CREATE VIEW view_사원;
```

▶ 실행결과

View	Create View	character_set_client	collation_connection
view_사원	CREATE ALGORITHM=UNDEFINED DEFINER='root' @'localhost' SQL SECURITY DEFINER VIEW `view_사원` (`이름`, `전화번호` , ...)	utf8mb4	utf8mb4_0900_ai_ci

5. 뷰 삭제하기

■ 뷰 삭제하기

- 형식

```
DROP VIEW 뷰명;
```

■ [예제 9-7] 'view_사원' 뷰를 삭제하시오.

```
DROP VIEW view_사원;
```

Section 02

뷰를 통한 데이터 삽입

1. 데이터 삽입 조건

■ 데이터 삽입 조건

- 삽입할 대상의 컬럼 명시
- NOT NULL 제약조건
- PRIMARY KEY 제약조건
- CHECK 제약조건
- WITH CHECK OPTION 조건
- 함수, 수식, JOIN, GROUP BY, UNION 등 사용 여부

1. 데이터 삽입 조건

■ [예제 9-8] [예제 9-3]에서 작성한 'view_사원_여' 뷰를 사용하여 다음 레코드를 삽입하시오. 오류가 발생한다면 이유를 생각해보시오

- 이름: 황여름, 전화번호: (02)587-4989, 입사일: 2023-02-10, 주소: 서울시 강남구 청담동 23-5, 성별: 여

```
INSERT INTO view_사원_여(이름, 전화번호, 입사일, 주소, 성별)
VALUES('황여름','(02)587-4989','2023-02-10','서울시 강남구 청담동 23-5','여');
```

Error Code: 1423. Field of view '한빛무역.view_사원_여' underlying table doesn't have a default value

1. 데이터 삽입 조건

- [예제 9-8] [예제 9-3]에서 작성한 'view_사원_여' 뷰를 사용하여 다음 레코드를 삽입하시오. 오류가 발생한다면 이유를 생각해보시오
 - 오류 해결1

```
CREATE OR REPLACE VIEW view_사원_여
AS
SELECT 사원번호
    ,이름
    ,집전화 AS 전화번호
    ,입사일
    ,주소
    ,성별
FROM 사원
WHERE 성별 = '여';
```

1. 데이터 삽입 조건

- [예제 9-8] [예제 9-3]에서 작성한 'view_사원_여' 뷰를 사용하여 다음 레코드를 삽입하시오. 오류가 발생한다면 이유를 생각해보시오
 - 오류 해결2

```
ALTER VIEW view_사원_여
AS
SELECT 사원번호
    ,이름
    ,집전화 AS 전화번호
    ,입사일
    ,주소
    ,성별
FROM 사원
WHERE 성별 = '여';
```

1. 데이터 삽입 조건

■ [예제 9-8] [예제 9-3]에서 작성한 'view_사원_여' 뷰를 사용하여 다음 레코드를 삽입하시오. 오류가 발생한다면 이유를 생각해보시오

- 사원번호: E12, 이름: 황여름, 전화번호: (02)587-4989, 입사일: 2023-02-10, 주소: 서울시 강남구 청담동 23-5, 성별: 여

```
INSERT INTO view_사원_여(사원번호, 이름, 전화번호, 입사일, 주소, 성별)  
VALUES('E12','황여름','(02)587-4989','2023-02-10','서울시 강남구 청담동 23-5','여');
```

- 뷰를 통한 레코드 삽입 확인

```
SELECT *  
FROM view_사원_여  
WHERE 사원번호 = 'E12';
```

▶ 실행 결과

사원번호	이름	전화번호	입사일	주소	성별
E12	황여름	(02)587-4989	2023-02-10	서울시 강남구 청담동 23-5	여

1. 데이터 삽입 조건

- [예제 9-8] [예제 9-3]에서 작성한 'view_사원_여' 뷰를 사용하여 다음 레코드를 삽입하시오. 오류가 발생한다면 이유를 생각해보시오
 - 뷰를 사용하여 레코드를 추가하였지만 실제로는 사원 테이블에 레코드가 삽입됨

```
SELECT *
FROM 사원
WHERE 사원번호 = 'E12';
```

▶ 실행결과

사원번호	이름	성별	직위	생년	월사진	주소	도시	지역	집전화	상사번호	부서번호
E12	장어른	여	대리	1988-01-01	2020-02-10	서울시 강남구 테헤란로 123-5	서울시	강남구	(02) 555-4909	0000	0000

1. 데이터 삽입 조건

- [예제 9-9] [예제 9-2]에서 작성한 'view_제품별주문수량합' 뷰를 사용하여 레코드를 삽입하시오. 오류가 난다면 이유를 생각해보시오
 - 제품명: 단짠 새우깡, 주문수량합: 250

```
INSERT INTO view_제품별주문수량합  
VALUES('단짠 새우깡', 250);
```

Error Code: 1471. The target table view_제품별주문수량합 of the INSERT is not insertable-into

2. WITH CHECK OPTION

■ WITH CHECK OPTION

- 뷰에서 사용하는 옵션 중 하나
- 특정 뷰에 대해 INSERT 또는 UPDATE 작업을 수행할 때 해당 작업으로 인해 뷰에서 정의한 조건을 만족하지 않는 행이 생성되거나 수정되는 것을 방지함
- WITH CHECK OPTION을 사용하면 뷰를 통해 데이터를 수정하거나 삽입할 때 뷰의 조건을 만족하는 데이터만 작업이 가능함
- 즉, 뷰에서 사용된 조건식을 기반으로 데이터의 일관성을 보장하기 위한 제약조건
- 형식

```
CREATE [OR REPLACE] VIEW 뷰명  
AS  
SELECT문  
WITH CHECK OPTION;
```

2. WITH CHECK OPTION

■ [예제 9-10] 'view_사원_여' 뷰를 사용하여 '남' 사원 정보를 추가하고, 결과를 확인하시오.

- 사원번호: E13, 이름: 강겨울, 입사일: 2023-02-10, 주소: 서울시 성북구 장위동 123-7, 성별: 남

```
INSERT INTO view_사원_여(사원번호, 이름, 입사일, 주소, 성별)
VALUES('E13','강겨울','2023-02-10','서울시 성북구 장위동 123-7','남');
```

- 삽입된 레코드를 뷰로 확인하기

```
SELECT *
FROM view_사원_여
WHERE 사원번호 = 'E13';
```

▶ 실행결과

사원번호	이름	전화번호	입사일	주소	성별
E13	강겨울	010-1234-5678	2023-02-10	서울시 성북구 장위동 123-7	남

- ✓ 삽입은 정상적으로 이루어졌으나 검색되지 않음

2. WITH CHECK OPTION

■ [예제 9-10] 'view_사원_여' 뷰를 사용하여 '남' 사원 정보를 추가하고, 결과를 확인하시오.

- 테이블에서 레코드를 확인하기

```
SELECT *
FROM 사원
WHERE 사원번호 = 'E13';
```

▶ 실행결과

사원번호	이름	성별	직위	성별	성별	입사일	주소	도시	지역	집전화	실사번호	부서번호
E13	김가을	여	대리	여	여	2023-02-10	서울시 성북구 강원동 123-7	서울	성북	02-1234-5678	1234567890	0000

- ✓ 테이블에는 존재함

2. WITH CHECK OPTION

■ [예제 9-10] 'view_사원_여' 뷰를 사용하여 '남' 사원 정보를 추가하고, 결과를 확인하시오.

- 해결방안

```
CREATE OR REPLACE VIEW view_사원_여
AS
SELECT 사원번호
      ,이름
      ,집전화 AS 전화번호
      ,입사일
      ,주소
      ,성별
FROM 사원
WHERE 성별 = '여'
WITH CHECK OPTION;
```

2. WITH CHECK OPTION

■ [예제 9-10] 'view_사원_여' 뷰를 사용하여 '남' 사원 정보를 추가하고, 결과를 확인하시오.

- 새로운 '남' 사원을 추가해보기
 - ✓ 사원번호: E14, 이름: 유봄, 성별: 남

```
INSERT INTO view_사원_여(사원번호, 이름, 성별)  
VALUES('E14','유봄','남');
```

Error Code: 1369. CHECK OPTION failed '한빛무역.view_사원_여'

- ✓ 이름: 황여름, 성별: 남

```
UPDATE view_사원_여  
SET 성별 = '남'  
WHERE 이름 = '황여름';
```

Error Code: 1369. CHECK OPTION failed '한빛무역.view_사원_여'

Section 03

인덱스

1. 인덱스의 개념

■ 인덱스(Index)

- 데이터베이스 테이블에서 특정 컬럼이나 컬럼의 집합에 대한 검색 성능을 향상시키기 위해 사용되는 자료구조
- 데이터베이스의 성능 향상을 위해 중요한 역할을 수행함

■ 인덱스의 장점

- 검색 속도 향상
- 정렬 및 순서 유지
- 중복 제거
- 조인 성능 향상
- 쿼리 성능 향상

2. 인덱스 종류와 주의사항

■ 기본 인덱스(Primary Index)

- 클러스터형 인덱스(Clustered Index)라고도 함
- 테이블의 기본키 컬럼에 대해 행 데이터가 오름차순으로 정렬되는 인덱스
- 기본 인덱스는 테이블당 한 개만 생성할 수 있음
- 기본 인덱스는 사전과 유사한 방식으로 저장됨



그림 9-2 기본 인덱스와 유사한 사전

2. 인덱스 종류와 주의사항

■ 보조 인덱스(Secondary Index)

- 기본 인덱스 이외 인덱스로, 비클러스터형 인덱스(Non-Clustered Index)라고도 함
- 보조 인덱스는 책의 찾아보기 페이지와 유사함



그림 9-3 보조 인덱스와 유사한 색인

2. 인덱스 종류와 주의사항

■ 인덱스 생성 시 주의사항

- 쿼리 패턴 분석
- 테이블 크기와 메모리 제한
- 업데이트 비용 주의
- 복합 인덱스
- 중복 및 NULL 값

3. MySQL 인덱스의 자료구조

■ B-트리 인덱스

- 가장 일반적으로 사용되는 인덱스 형식으로 좌우 자식 간에 항상 균형을 이루는 균형 트리(Balanced Tree)임
- B-트리에서 노드(Node)는 블록(Block) 또는 페이지(Page)라고 부름
- M차 B-트리
 - ✓ 최대 M개의 자식 노드를 가질 수 있는 B-트리
- 각 노드의 키들은 좌우로 다른 노드를 가리키는 포인터를 가지고 있음
 - ✓ 좌측 포인터는 키보다 작은 데이터를 가진 노드를 가리키고,
우측 포인터는 키보다 큰 데이터를 가진 노드를 가리킴
- B-트리 인덱스는 테이블 데이터를 정렬하여 B-트리 형태로 구성하며, 각 노드는 정렬된 값의 범위를 나타냄
- 이러한 구조로 인해 효율적인 범위 검색과 정렬이 가능함

3. MySQL 인덱스의 자료구조

■ B-트리 인덱스

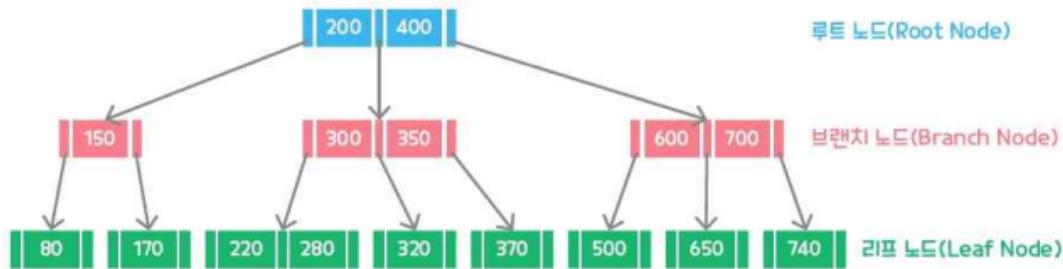


그림 9-4 3차 B-트리의 형태

3. MySQL 인덱스의 자료구조

■ MySQL의 B-트리 인덱스

- 데이터는 루트 노드부터 탐색이 시작되고, 정렬된 인덱스키를 따라서 리프 노드까지 도달함
- 기본 인덱스는 리프 노드에 키가 포함된 행의 모든 데이터를 포함하므로 추가적인 데이터 조회가 필요하지 않음
- 반면 보조 인덱스는 먼저 인덱스키와 쌍이 되는 기본키 값을 찾은 후 기본 인덱스에서 검색함

3. MySQL 인덱스의 자료구조

■ MySQL의 B-트리 인덱스

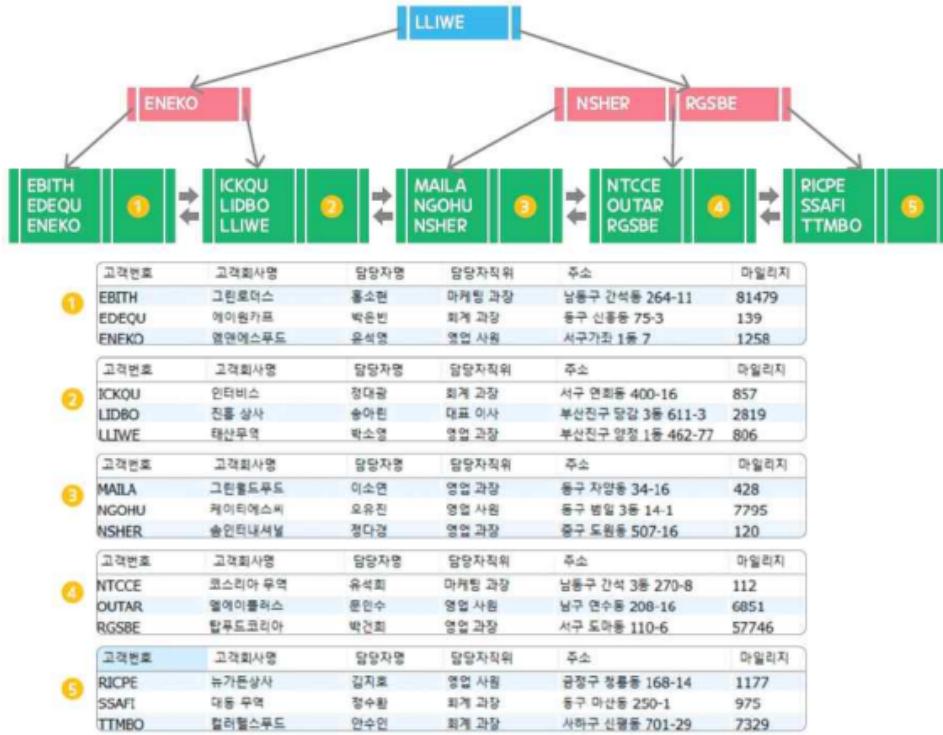


그림 9-5 고객번호를 기본키로 한 기본 인덱스의 형태

3. MySQL 인덱스의 자료구조

■ MySQL의 B-트리 인덱스

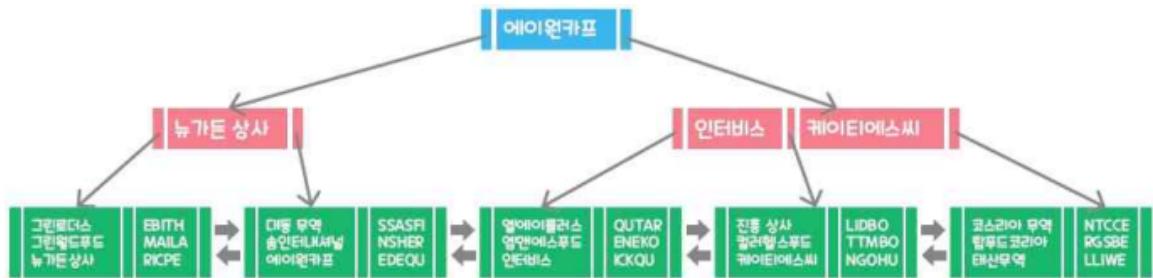


그림 9-6 고객회사명을 인덱스키로 한 보조 인덱스의 형태

4. 인덱스 생성과 사용

■ 인덱스 생성하기

- 형식1

```
CREATE [UNIQUE] INDEX 인덱스명 ON 테이블명(컬럼명);
```

- 형식2: 테이블을 생성하면서 인덱스를 함께 생성

```
CREATE TABLE 테이블명  
{  
    컬럼1 데이터타입  
    , 컬럼2 데이터타입  
    , 컬럼3 데이터타입  
    , 컬럼4 데이터타입  
    , PRIMARY KEY(컬럼1, 컬럼2, 컬럼3)  
    , INDEX 인덱스명 (컬럼4) •————①  
};
```

4. 인덱스 생성과 사용

■ 인덱스 생성하기

- 테이블에 인덱스를 추가하는 형식

```
ALTER TABLE 테이블명 ADD INDEX 인덱스명 (컬럼명);
```

- 테이블에서 인덱스를 삭제하는 형식

```
ALTER TABLE 테이블명 DROP INDEX 인덱스명 (컬럼명);
```

- 테이블에 걸려있는 인덱스를 확인하는 형식

```
SHOW INDEX FROM 테이블명;
```

4. 인덱스 생성과 사용

■ 인덱스 사용 시 고려 사항

- 인덱스가 다중 컬럼으로 구성되어 있다면 옵티마이저는 인덱스의 가장 왼쪽 접두사(Leftmost Prefix)를 사용해서 컬럼을 찾게 됨
- LIKE 비교에서 인수가 상수 문자열인 경우에는 인덱스를 사용할 수 있으나, '%'와 같은 와일드카드 문자로 시작하는 경우에는 인덱스를 사용할 수 없음
- 인덱스의 가장 왼쪽 접두사가 반드시 모든 AND 그룹에서 사용되어야 인덱스를 사용할 수 있음

4. 인덱스 생성과 사용

■ [예제 9-11] 날씨 테이블과 인덱스를 생성하시오.

- 컬럼: 년도 INT, 월 INT, 일 INT, 도시 VARCHAR(20), 기온 NUMERIC(3,1), 습도 INT
- 기본키: 년도 + 월 + 일 + 도시
- 보조 인덱스: 기온, 도시

```
CREATE TABLE 날씨
(
    년도 INT
    ,월 INT
    ,일 INT
    ,도시 VARCHAR(20)
    ,기온 NUMERIC(3,1)
    ,습도 INT
    ,PRIMARY KEY(년도, 월, 일, 도시)
    ,INDEX 기온인덱스(기온)
    ,INDEX 도시인덱스(도시)
);
```

점검문제

문제 1

다음 실행결과와 같이 피벗 형식으로 결과가 보이도록 뷰를 작성하시오.

▶ 실행결과

도시	대표 이사	영업	마케팅	회계
서울특별시	11	23	10	3
충청시	0	2	0	0
구리시	0	0	1	0
김해시	0	1	0	0
광택시	0	1	0	0
인천광역시	0	4	2	1
대전광역시	0	2	0	1
상주시	0	1	0	0
천주시	0	1	0	0
	0	0	0	0

Chapter 10

프로시저와 함수, 트리거



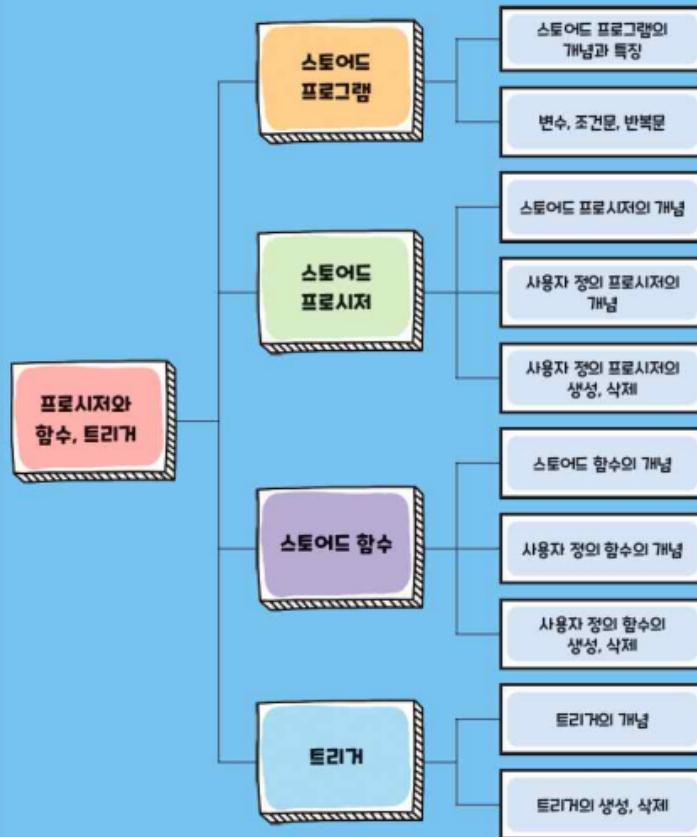
목차

1. 스토어드 프로그램
2. 스토어드 프로시저
3. 스토어드 함수
4. 트리거

학습목표

- 스토어드 프로그램의 개념과 특징을 이해할 수 있습니다.
- 스토어드 프로그램 작성을 위한 문법을 작성할 수 있습니다.
- 스토어드 프로시저와 스토어드 함수의 개념과 사용법을 이해할 수 있습니다.
- 트리거의 개념과 사용법을 이해할 수 있습니다.

Preview



Section 01

스토어드 프로그램

1. 스토어드 프로그램의 개념과 특징

■ SQL 스토어드 프로그램(Stored Program)

- 데이터베이스에서 실행되는 프로그램으로 일련의 SQL문을 포함하고 있는 데이터베이스 객체
- 주로 프로시저나 함수, 트리거 형태로 사용됨

■ 스토어드 프로그램의 특징

- 프로그래밍 언어적 특성
- 코드 캡슐화와 유지 보수
- 재사용성
- 성능 향상
- 보안
- 트랜잭션 관리
- 비즈니스 로직 분리

2. 변수

■ 변수

- 값을 저장하고 참조하는 데 사용되는 식별자

표 10-1 MySQL의 변수 유형

변수 유형	설명	예
사용자 정의 변수 (User-defined Variables)	<ul style="list-style-type: none">세션 내에서 사용자가 정의한 변수를 의미하며, 세션이 종료될 때까지 값을 유지한다.@를 접두사로 사용하며, 변수에 값을 할당하기 위해서는 SET 또는 SELECT를 사용한다.	<ul style="list-style-type: none">값 할당 SET @from = 100, @to = 10000; 또는 SELECT @from = 100, @to = 10000;사용 SELECT * FROM 고객 WHERE 마일리지 BETWEEN @from AND @to
로컬 변수 (Local Variables)	<ul style="list-style-type: none">스토어드 프로시저나 함수와 같은 스토어드 프로그램 내에서 사용되는 변수로, 해당 프로그램 실행이 끝나면 값이 소멸된다.DECLARE문을 사용하여 변수를 선언하고, SELECT나 SET을 사용하여 값을 할당한다.	<pre>DECLARE city VARCHAR DEFAULT '과천시';</pre>
시스템 변수 (System Variables)	<ul style="list-style-type: none">MySQL 서버의 동작을 제어하고 구성하는 데 사용되는 변수로 @@를 접두사로 가진다.실행 중인 서버에서 사용되는 현재 값을 보려면 SHOW 또는 SELECT를 사용하며, 값을 변경하려면 SET을 사용한다.	<ul style="list-style-type: none">시스템 변수 확인 SHOW VARIABLES LIKE '%sort_buffer_size%'; 또는 SELECT @@sort_buffer_size;값 할당 SET @@sort_buffer_size = 262144;

3. 조건문

■ IF문

- 조건에 따라 다른 명령을 처리하고자 할 때에는 IF문을 사용함
- 조건이 여러 개일 때에는 ELSE IF로 조건을 추가할 수 있음
- ELSEIF절은 여러 개 작성이 가능하지만, ELSE는 문장 내에서 한 번만 작성 가능함
- 형식

```
IF 조건1 THEN
    조건1이 참일 때 실행할 코드
[ELSEIF 조건2 THEN
    조건2가 참일 때 실행할 코드]
[ELSE
    모든 조건이 참이 아닐 때 실행할 코드]
END IF;
```

3. 조건문

■ [예제 10-1] IF문을 사용하여 두 개의 숫자 10과 5의 크기를 비교하는 프로시저를 작성하시오

```
DELIMITER $$  
CREATE PROCEDURE proc_if()  
BEGIN  
    DECLARE x INT; • ①  
    DECLARE y INT DEFAULT 5; • ②  
    SET x = 10; • ③  
  
    IF x > y THEN  
        SELECT 'x는 y보다 큽니다.' AS 결과;  
    ELSE  
        SELECT 'x는 y보다 작거나 같습니다.' AS 결과;  
    END IF;  
END $$  
DELIMITER ;
```

①
②
③
④

스토어드 프로시저에 대한 자세한
내용은 다음 절에서 살펴봅니다.

- 스토어드 프로시저를 실행할 때는 CALL을 사용함

```
CALL proc_if();
```

▶ 실행결과

결과
x는 y보다 큽니다.

3. 조건문

■ CASE문

- 주로 복잡한 다중 조건을 처리하기 위해 사용됨
- 형식

```
CASE
WHEN 조건1 THEN
    조건1이 참일 때 실행할 코드
[WHEN 조건2 THEN
    조건2가 참일 때 실행할 코드]
[ELSE
    모든 조건이 참이 아닐 때 실행할 코드]
END CASE;
```

3. 조건문

■ [예제 10-2] CASE문을 사용하여 숫자 10이 짝수인지 홀수인지를 판별하는 프로시저를 작성하시오.

```
DELIMITER $$  
CREATE PROCEDURE proc_case()  
BEGIN  
    DECLARE x INT DEFAULT 10;  
    DECLARE y INT;  
    SET y = 10 mod 2; • ①  
  
    CASE  
        WHEN y = 0 THEN  
            SELECT 'x는 짝수입니다' AS '결과';  
        ELSE  
            SELECT 'x는 홀수입니다' AS '결과';  
    END CASE;  
END $$  
DELIMITER ;
```



The diagram shows a brace grouping the WHEN y = 0 THEN block and the ELSE block of the CASE statement. A red dot labeled '①' is placed on the line before the SET statement. A red brace labeled '②' is placed around the WHEN and ELSE blocks.

```
CALL proc_case();
```

▶ 실행 결과

결과
x는 짝수입니다

4. 반복문

■ WHILE

- 주어진 조건이 참인 동안 반복적으로 코드를 실행하다가, 조건이 거짓이 되면 반복문을 종료함
- WHILE문은 반복 시작하기 전에 조건을 평가하기 때문에 조건이 처음부터 거짓이면 코드가 실행되지 않음
- 형식

```
WHILE 조건 DO  
    반복적으로 실행할 코드  
END WHILE;
```

4. 반복문

■ [예제 10-3] 1부터 10까지의 합을 출력하는 프로시저를 WHILE문으로 작성하시오

```
DELIMITER $$  
CREATE PROCEDURE proc_while()  
BEGIN  
    DECLARE x INT DEFAULT 0;  
    DECLARE y INT DEFAULT 0;  
  
    WHILE x < 10 DO  
        SET x = x + 1; •————①  
        SET y = y + x; •————②  
    END WHILE;  
    SELECT x, y; •————③  
END $$  
DELIMITER ;
```

```
CALL proc_while();
```

▶ 실행결과

x	y
10	55

4. 반복문

■ LOOP은

- 탈출 조건이 없으면 무한 반복되기 때문에 반복문 내에서 LEAVE문을 사용하여 루프를 종료해야 함
- 이때 루프에 레이블명을 지정하고, 지정한 레이블명은 LEAVE문에 기술함
- 형식

```
레이블명:LOOP  
반복적으로 실행할 코드  
IF 조건 THEN  
    LEAVE 레이블명;  
END IF;  
END LOOP;
```

4. 반복문

■ [예제 10-4] 1부터 10까지의 합을 출력하는 프로시저를 LOOP문으로 작성하시오.

```
DELIMITER $$  
CREATE PROCEDURE proc_loop()  
BEGIN  
    DECLARE x INT DEFAULT 0;  
    DECLARE y INT DEFAULT 0;  
  
    loop_sum:LOOP  
        SET x = x + 1; •—————①  
        SET y = y + x;  
        IF x > 10 THEN •—————②  
            LEAVE loop_sum; •—————③  
        END IF;  
        SELECT x, y;  
    END LOOP;  
END $$  
DELIMITER ;
```

```
CALL proc_loop();
```

▶ 실행 결과

x	y
10	55

4. 반복문

하나 더 알기 ✓

ITERATE

ITERATE도 LEAVE와 같이 반복문의 흐름을 제어하는 데 사용할 수 있습니다. LEAVE문은 현재 실행 중인 반복문을 종료하고, 반복문 바로 다음 문장으로 제어를 이동합니다. 반면 ITERATE문은 현재 반복문의 나머지 부분을 건너뛰고, 다음 반복 단계로 제어를 이동합니다.

labelA:LOOP

실행할 코드1;

IF 조건 THEN

LEAVE labelA;

END IF;

실행할 코드2;

END LOOP;

실행할 코드3

labelA:LOOP

실행할 코드1;

IF 조건 THEN

ITERATE labelA;

END IF;

실행할 코드2;

END LOOP;

실행할 코드3

그림 10-1 LEAVE와 ITERATE의 차이점



4. 반복문

■ REPEAT

- 조건이 참이 될 때까지 코드를 실행함
- REPEAT문은 반복문 내에서 조건을 비교하기 전에 최소 한 번은 코드가 실행됨
- 형식

REPEAT

반복적으로 실행할 코드

UNTIL 조건;

4. 반복문

■ [예제 10-5] 1부터 10까지의 합을 출력하는 프로시저를 REPEAT문으로 작성 하시오

```
DELIMITER $$  
CREATE PROCEDURE proc_repeat()  
BEGIN  
    DECLARE x INT DEFAULT 0;  
    DECLARE y INT DEFAULT 0;  
  
    REPEAT  
        SET x = x + 1; ●————①  
        SET y = y + x; ●————②  
        UNTIL x >= 10 END REPEAT; ●————③  
        SELECT x, y;  
    END $$  
DELIMITER ;
```

```
CALL proc_repeat()
```

▶ 실행 결과

x	y
10	55

Section 02

스토어드 프로시저

1. 스토어드 프로시저의 개념

■ SQL 스토어드 프로시저(Stored Procedure)

- 데이터베이스 객체 중 하나로 데이터베이스에서 수행할 수 있는 일련의 SQL문과 제어문을 저장한 SQL 스토어드 프로그램을 의미함
- SQL 스토어드 프로시저를 사용하면 하나의 요청으로 여러 개의 SQL문을 실행할 수 있음
- 데이터 검색이나 조작, 업데이트, 삭제 등과 같은 다양한 작업을 수행할 수 있음
- 스토어드 프로시저는 시스템 스토어드 프로시저와 사용자 정의 프로시저로 구분 할 수 있음

1. 스토어드 프로시저의 개념

■ 시스템 스토어드 프로시저

- MySQL 데이터베이스 시스템에 내장되어 있는 특수한 유형의 저장 프로그램
- 백업, 복원, 성능 최적화, 사용자 관리 등 주로 데이터베이스 시스템의 내부 동작을 제어하고 관리하는 데 사용됨
- 데이터베이스에 내장되어 있기 때문에 별도로 설치할 필요가 없음
- 사용자가 직접 프로시저를 생성할 수도 있는데, 이러한 프로시저를 사용자 정의 프로시저라고 함

2. 사용자 정의 프로시저의 개념

■ 사용자 정의 프로시저

- 자주 사용하는 SQL문을 프로시저로 생성하여 저장한 후 필요 시에 호출해서 사용 할 수 있음

■ 사용자 정의 프로시저의 구성 요소

- 프로시저 정의
 - ✓ IN 매개변수
 - ✓ OUT 매개변수
 - ✓ INOUT 매개변수
- 변수
- SQL문
- 제어문

3. 사용자 정의 프로시저의 생성과 삭제

■ 사용자 정의 프로시저의 생성, 호출, 삭제

- 생성 형식

```
DELIMITER $$  
CREATE PROCEDURE 사용자 정의 프로시저명  
(  
    [IN|OUT|INOUT 매개변수]  
)  
BEGIN  
    실행코드  
END $$  
DELIMITER ;
```

- 호출 형식

```
CALL 프로시저명;
```

- 삭제 형식

```
DROP PROCEDURE 프로시저명;
```

3. 사용자 정의 프로시저의 생성과 삭제

■ [예제 10-6] 고객 정보와 고객 수를 보이는 프로시저를 작성하시오.

```
DELIMITER $$  
CREATE PROCEDURE proc_고객정보()  
BEGIN  
    SELECT *  
    FROM 고객;  
  
    SELECT COUNT(*) AS 고객수  
    FROM 고객;  
END $$  
DELIMITER ;
```

```
CALL proc_고객정보();
```

▶ 실행결과

Result Grid						
고객번호	Filter Rows:		Export:	Wrap Cell Content:	Result Grid	
	고객회사명	담당자명			도시	지역
ACDOR	굿도닝서플	이준우	영업 과장	송파구 잠실동 220	서울특별시	서울특별시
ADHTR	알케이 상사	김병현	영업 사원	동작구 흑석 3동 140-3	서울특별시	서울특별시
ANKFR	씨엔그룹	김성민	영업 사원	가락동 301	광명시	경기도
ANRFR	오리안무역	최지수	마케팅 과장	성북구 길동동 136-11	서울특별시	서울특별시
	남해풀향식품	강준	마케팅 과장	수락동 435	구리시	경기도

Result Grid	
고객수	Result 1
89	Result 2

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-7] 도시를 입력하면 해당 도시의 고객 정보와 고객 수를 보이는 프로시저를 작성하시오

```
DELIMITER $$  
CREATE PROCEDURE proc_도시고객정보  
(  
    IN city VARCHAR(50) •————①  
)  
BEGIN  
    SELECT *  
    FROM 고객  
    WHERE 도시 = city; •————②  
    SELECT COUNT(*) AS 고객수  
    FROM 고객  
    WHERE 도시 = city;  
END $$  
DELIMITER ;
```

```
CALL proc_도시고객정보('부산광역시');
```

3. 사용자 정의 프로시저의 생성과 삭제

■ [예제 10-7] 도시를 입력하면 해당 도시의 고객 정보와 고객 수를 보이는 프로시저를 작성하시오

▶ 실행결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
LIDBO	진홍 상사	송아린	대표 이사	부산진구 달길 3동 611-3	부산광역시	NULL	(051)784-1945	3101
LLIWE	태산무역	박소영	영업 과장	부산진구 양정 1동 462-77	부산광역시	NULL	(051)555-5111	887
NGOHU	케이티에스씨	오유진	영업 사원	동구 통일 3동 14-1	부산광역시	NULL	(051)820-1993	8575
RICPE	뉴가든상사	김지호	영업 사원	금정구 청룡동 168-14	부산광역시	NULL	(051)12-1122	1295
TTMBO	컬러풀스튜드	안수인	회계 과장	사하구 신봉동 701-29	부산광역시	NULL	(051)845-9483	8062

Result 3 ×

Result 4

고객수
5

Result 3

Result 4 ×

Error Code 12670이 발생하면 207P의
<하나 더 알기>를 참고하세요.

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-8] 주문년도와 고객의 도시를 입력하면 해당 년도에 해당 도시의 고객이 주문한 내역에 대하여 주문고객별로 주문건수를 보이는 프로시저를 작성하시오.

```
DELIMITER $$  
CREATE PROCEDURE proc_주문년도시_고객정보  
(  
    IN order_year INT  
    ,IN city VARCHAR(50)  
)  
BEGIN  
    SELECT 고객.고객번호  
        ,고객회사명  
        ,도시  
        ,COUNT(*) AS 주문건수  
    FROM 고객 JOIN 주문  
    ON 고객.고객번호 = 주문.고객번호  
    WHERE YEAR(주문일) = order_year  
    AND 도시 = city  
    GROUP BY 고객.고객번호  
        ,고객회사명;  
END $$  
DELIMITER ;
```

▶ 실행 결과

CALL proc_주문년도시_고객정보(2021, '공주시');

고객번호	고객회사명	도시	주문건수
STCEA	모케이식품	공주시	4

3. 사용자 정의 프로시저의 생성과 삭제

■ [예제 10-9] 고객회사명과 추가할 마일리지를 입력하면 해당 고객에 대하여
입력한 마일리지만큼 추가하는 프로시저를 작성하시오.

```
DELIMITER $$  
CREATE PROCEDURE proc_고객회사명_마일리지추가  
(  
    IN company VARCHAR(50)  
    ,IN add_mileage INT  
)  
BEGIN  
    SELECT 고객번호  
        ,고객회사명  
        ,마일리지 AS 변경전마일리지  
    FROM 고객  
    WHERE 고객회사명 = company;  
  
    UPDATE 고객  
    SET 마일리지 = 마일리지 + add_mileage  
    WHERE 고객회사명 = company;  
  
    SELECT 고객번호  
        ,고객회사명  
        ,마일리지 AS 변경후마일리지  
    FROM 고객  
    WHERE 고객회사명 = company;  
END $$  
DELIMITER ;
```

만약 UPDATE나 DELETE문을 실행
했을 때 오류가 발생한다면 2장 61p의
〈하나 더 일기〉를 참고하세요.

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-9] 고객회사명과 추가할 마일리지를 입력하면 해당 고객에 대하여 입력한 마일리지만큼 추가하는 프로시저를 작성하시오.

```
CALL proc_고객회사명_마일리지추가('진영무역',1000);
```

▶ 실행 결과

고객번호	고객회사명	변경전마일리지	고객번호	고객회사명	변경후마일리지
LKOFO	진영무역	9468	LKOFO	진영무역	10468

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-10] 고객회사명을 입력하면 해당 고객의 마일리지를 변경하는 프로시저를 작성하시오. 이때 만일 고객의 마일리지가 전체 고객의 평균마일리지 보다 크다면 100점을 추가하고, 그렇지 않다면 전 고객의 평균마일리지만큼으로 변경하시오.

```
DELIMITER $$  
CREATE PROCEDURE proc_고객회사명_평균마일리지로변경  
(  
    IN company VARCHAR(50)  
)  
BEGIN  
    DECLARE 평균마일리지 INT;  
    DECLARE 보유마일리지 INT;  
  
    SELECT 고객회사명  
        , 마일리지 AS 변경전마일리지  
    FROM 고객  
    WHERE 고객회사명 = company;  
  
    SET 평균마일리지 = (SELECT AVG(마일리지)  
        FROM 고객);  
    SET 보유마일리지 = (SELECT 마일리지  
        FROM 고객  
        WHERE 고객회사명 = company);  
    ①
```

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-10] 고객회사명을 입력하면 해당 고객의 마일리지를 변경하는 프로시저를 작성하시오. 이때 만일 고객의 마일리지가 전체 고객의 평균마일리지 보다 크다면 100점을 추가하고, 그렇지 않다면 전 고객의 평균마일리지만큼으로 변경하시오.

```
IF (보유마일리지 > 평균마일리지) THEN
    UPDATE 고객
    SET 마일리지 = 마일리지 + 100
    WHERE 고객회사명 = company;
ELSE
    UPDATE 고객
    SET 마일리지 = 평균마일리지
    WHERE 고객회사명 = company;
END IF;
```

②

```
SELECT 고객회사명
      , 마일리지 AS 변경후마일리지
   FROM 고객
  WHERE 고객회사명 = company;
END $$

DELIMITER ;
```

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-10] 고객회사명을 입력하면 해당 고객의 마일리지를 변경하는 프로시저를 작성하시오. 이때 만일 고객의 마일리지가 전체 고객의 평균마일리지 보다 크다면 100점을 추가하고, 그렇지 않다면 전 고객의 평균마일리지만큼으로 변경하시오.

```
CALL proc_고객회사명_평균마일리지로변경('굿모닝서울');
```

▶ 실행 결과

고객회사명	변경전마일리지	고객회사명	변경후마일리지
굿모닝서울	17502	굿모닝서울	17602

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-11] 고객회사명을 입력하면 고객의 보유 마일리지에 따라서 등급을 보이는 프로시저를 작성하시오. 이때 고객의 마일리지가 100,000점 이상이면 '최우수고객회사', 50,000점 이상이면 '우수고객회사', 그 나머지는 '관심고객회사'라고 보이시오.

```
DELIMITER $$  
CREATE PROCEDURE proc_고객등급  
(  
    IN company VARCHAR(50),  
    OUT grade VARCHAR(20) • ①  
)  
BEGIN  
    DECLARE 보유마일리지 INT;  
  
    SELECT 마일리지  
    INTO 보유마일리지  
    FROM 고객  
    WHERE 고객회사명 = company;  
  
    IF 보유마일리지 > 100000 THEN  
        SET grade = '최우수고객회사';  
    ELSEIF 보유마일리지 >= 50000 THEN  
        SET grade = '우수고객회사';  
    ELSE  
        SET grade = '관심고객회사';  
    END IF;  
END $$  
DELIMITER ;
```

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-11] 고객회사명을 입력하면 고객의 보유 마일리지에 따라서 등급을 보이는 프로시저를 작성하시오. 이때 고객의 마일리지가 100,000점 이상이면 '최우수고객회사', 50,000점 이상이면 '우수고객회사', 그 나머지는 '관심고객회사'라고 보이시오.

```
CALL proc_고객등급('그린로더스', @그린로더스등급); •————③
```

```
CALL proc_고객등급('오투락', @오투락등급);
```

```
SELECT @그린로더스등급, @오투락등급; •————④
```

▶ 실행결과

②그린로더스등급	③오투락등급
우수고객회사	관심고객회사

3. 사용자 정의 프로시저의 생성과 삭제

■ [예제 10-12] 인상율과 금액을 입력하면 인상금액을 계산하고, 그 결과를 확인할 수 있는 프로시저를 작성하시오.

```
DELIMITER $$  
CREATE PROCEDURE proc_인상금액  
(  
    IN increase_rate INT  
    ,INOUT price INT • ①  
)  
BEGIN  
    SET price = price * (1 + increase_rate / 100); •— ④  
END $$  
DELIMITER ;
```

```
SET @금액 = 10000; • ②  
CALL proc_인상금액(10, @금액); • ③  
SELECT @금액; • ⑤
```

▶ 실행결과

@금액
11000

```
CALL proc_인상금액(10, @금액);  
SELECT @금액;
```

▶ 실행결과

총금액
12100

Section 03

스토어드 함수

1. 스토어드 함수의 개념

■ SQL 스토어드 함수(Stored Function)

- 데이터베이스에서 사용할 수 있는 함수
- SQL 스토어드 함수는 SQL문의 일부로 호출되며, 함수 내에서 로직을 실행하고 값을 반환할 수 있음

■ MySQL의 주요 스토어드 함수의 유형

- 내장 함수
 - ✓ MySQL을 설치하면 자동으로 제공되는 함수
- 사용자 정의 함수
 - ✓ 사용자가 필요에 따라 직접 정의하는 함수

2. 사용자 정의 함수의 개념

■ 사용자 정의 함수의 구성 요소

- 함수 정의
- 입력 매개변수
- 반환 값
- 제어문

3. 사용자 정의 함수의 생성과 삭제

■ 사용자 정의 함수의 생성, 삭제

- 생성 형식

```
DELIMITER $$  
CREATE FUNCTION 사용자정의함수명(매개변수)  
RETURNS 반환형식  
BEGIN  
    실행코드  
    RETURN 반환값;  
END $$  
DELIMITER ;
```

- 실행 형식

```
SELECT 함수명();
```

- 삭제 형식

```
DROP FUNCTION 함수명;
```

3. 사용자 정의 함수의 생성과 삭제

- [예제 10-13] 수량과 단가를 입력하면 두 수를 곱하여 금액을 반환하는 함수를 생성하시오.

```
DELIMITER $$  
CREATE FUNCTION func_금액(quantity INT, price INT)  
RETURNS INT  
BEGIN  
DECLARE amount INT;  
SET amount = quantity * price;  
RETURN amount;  
END $$  
DELIMITER ;
```

```
SELECT func_금액(100, 4500);
```

▶ 실행 결과

func_금액(100, 4500)
450000

```
SELECT *  
,func_금액(주문수량, 단가) AS 주문금액  
FROM 주문세부;
```

▶ 실행 결과

주문번호	제품번호	단가	주문수량	할인율	주문금액
H0249	14	1900	9	0	17100
H0249	51	4200	40	0	168000
H0250	41	800	10	0	8000
H0250	51	4200	35	0.15	147000

3. 사용자 정의 함수의 생성과 삭제

하나 더 알기 ✓

Error Code: 1418 에러 발생

log_bin_trust_function_creators 변수가 0(OFF)으로 설정된 경우에는 사용자는 함수를 생성하거나 수정할 수 없기 때문에 다음과 같은 오류가 발생합니다.

Error Code: 1418. This function has none of DETERMINISTIC, NO SQL, or
READS SQL DATA in its declaration and binary logging is enabled.

이때 함수 생성에 제약을 받지 않도록 1(ON)로 설정 값을 변경하면 문제를 해결할 수 있습니다.

```
SET GLOBAL log_bin_trust_function_creators = 1;
```



Section 04

트리거

1. 트리거의 개념

■ 트리거(Trigger)

- 데이터베이스에서 데이터 삽입, 변경 또는 삭제와 같은 특정 이벤트가 발생할 때마다 자동으로 실행되는 작업을 의미함
- INSERT, UPDATE, DELETE와 같은 이벤트가 발생할 때마다 트리거에 정의된 SQL문이 자동 실행됨
- 트리거는 복잡한 비즈니스 규칙 구현 및 데이터 검증, 보안, 로깅 등에 사용됨
- 트리거를 통해 데이터의 일관성을 유지할 수 있으며 다양한 작업을 자동화할 수 있음

■ 트리거의 구성 요소

- 트리거 정의
- 이벤트
- 테이블
- 타이밍
- 본문

2. 트리거의 생성과 삭제

■ 트리거 생성, 삭제

- 생성 형식

```
DELIMITER $$  
CREATE TRIGGER 트리거명  
BEFORE|AFTER INSERT|UPDATE|DELETE ON 테이블명  
FOR EACH ROW  
BEGIN  
    실행코드  
END $$  
DELIMITER ;
```

표 10-2 트리거 이벤트에 따른 OLD, NEW 키워드 사용 가능 여부

트리거 이벤트	OLD.컬럼명	NEW.컬럼명
INSERT	×	○
UPDATE	○	○
DELETE	○	×

- 생성 확인 형식

```
SHOW 트리거명;
```

- 삭제 형식

```
DROP TRIGGER 트리거명;
```

2. 트리거의 생성과 삭제

- [예제 10-14] 제품로그 테이블을 생성하시오. 그리고 제품을 추가할 때마다 로그 테이블에 정보를 남기는 트리거를 작성하시오.

```
CREATE TABLE 제품로그
(
    로그번호 INT AUTO_INCREMENT PRIMARY KEY,
    처리 VARCHAR(10),
    내용 VARCHAR(100),
    처리일 TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
DELIMITER $$

CREATE TRIGGER trigger_제품추가로그
AFTER INSERT ON 제품
FOR EACH ROW
BEGIN
    INSERT INTO 제품로그(처리, 내용)
    VALUES('INSERT', CONCAT('제품번호:', NEW.제품번호, '제품명:', NEW.제품명));
END $$

DELIMITER ;
```

2. 트리거의 생성과 삭제

■ [예제 10-14] 제품로그 테이블을 생성하시오. 그리고 제품을 추가할 때마다 로그 테이블에 정보를 남기는 트리거를 작성하시오.

- 트리거 동작 여부는 제품 테이블에 레코드를 추가하고 제품로그 테이블을 검색하여 확인함

```
INSERT INTO 제품(제품번호,제품명,단가,재고)
VALUES(99, '레몬캔디', 2000, 10);
```

```
SELECT *
FROM 제품
WHERE 제품번호 = 99;
```

▶ 실행 결과

제품번호	제품명	포장단위	단가	재고	재고금액
99	레몬캔디	EA	2000	10	20000

```
SELECT *
FROM 제품로그;
```

▶ 실행 결과

로그번호	처리	내용	처리일
1	INSERT	제품번호:99 제품명:레몬캔디	2023-11-24 14:17:15

2. 트리거의 생성과 삭제

- [예제 10-15] 제품 테이블에서 단가나 재고가 변경되면 변경된 사항을 제품로그 테이블에 저장하는 트리거를 생성하시오.

```
DELIMITER $$  
CREATE TRIGGER trigger_제품변경로그  
AFTER UPDATE ON 제품  
FOR EACH ROW  
BEGIN  
    IF (NEW.단가 <> OLD.단가) THEN •————①  
        INSERT INTO 제품로그(처리, 내용)  
        VALUES ('UPDATE', CONCAT('제품번호:', OLD.제품번호, ' 단가:', OLD.단가, '->', NEW.단가));  
    END IF;  
  
    IF (NEW.재고 <> OLD.재고) THEN •————②  
        INSERT INTO 제품로그(처리, 내용)  
        VALUES ('UPDATE', CONCAT('제품번호:', OLD.제품번호, ' 재고:', OLD.재고, '->', NEW.재고));  
    END IF;  
END $$  
DELIMITER ;
```

2. 트리거의 생성과 삭제

■ [예제 10-15] 제품 테이블에서 단가나 재고가 변경되면 변경된 사항을 제품로그 테이블에 저장하는 트리거를 생성하시오.

- 트리거 동작 여부는 제품 테이블에서 단가나 재고 값을 변경한 후, 제품로그 테이블을 확인함

```
UPDATE 제품
SET 단가 = 2500
WHERE 제품번호 = 99;
SELECT *
FROM 제품로그;
```

▶ 실행 결과

로그번호	처리	내용	처리일
1	INSERT	제품번호:99 제품명:레몬캔디	2023-11-24 14:17:15
2	UPDATE	제품번호:99 단가:2000->2500	2023-11-24 14:18:06

2. 트리거의 생성과 삭제

- [예제 10-16] 제품 테이블에서 제품 정보를 삭제하면 삭제된 레코드의 정보를 제품로그 테이블에 저장하는 트리거를 생성하시오.

```
DELIMITER $$  
CREATE TRIGGER trigger_제품삭제로그  
AFTER DELETE ON 제품  
FOR EACH ROW  
BEGIN  
    INSERT INTO 제품로그(처리, 내용)  
    VALUES ('DELETE', CONCAT('제품번호:', OLD.제품번호, ' 제품명:', OLD.제품명));  
END $$  
DELIMITER ;
```

- 트리거 동작 여부는 제품 테이블에서 레코드를 삭제한 후, 제품로그 테이블을 확인함

```
DELETE FROM 제품  
WHERE 제품번호 = 99;
```

```
SELECT * FROM 제품로그;
```

▶ 실행 결과

로그번호	처리	내용	처리일
1	INSERT	제품번호:99 제품명:레온천디	2023-11-24 14:17:15
2	UPDATE	제품번호:99 단가:2000->2500	2023-11-24 14:18:06
3	DELETE	제품번호:99 제품명:레온천디	2023-11-24 14:18:30

점검문제

점검문제

문제 1

제품명의 일부를 입력하면 해당 제품들에 대하여 제품명별로 주문수량합, 주문금액합을 보이는 프로시저를 작성하시오.

▶ 실행결과

제품명	주문수량합	주문금액합
뉴그린 토로피질 캔디	138	153900
칼스 계피 캔디	520	950000
미미 스카치 캔디	799	963700

문제 2

생일을 입력하면 연령구분을 반환하는 함수를 생성하시오.

조건

- ❶ 나이가 20살보다 적으면 '미성년'이다.
- ❷ 나이가 20살보다 많고 30살보다 적으면 '청년층'이다.
- ❸ 나이가 30살보다 많고 55살보다 적으면 '중년층'이다.
- ❹ 나이가 55살보다 많고 70살보다 적으면 '장년층'이다.
- ❺ 그 이외는 '노년층'이다.

▶ 실행결과

사원번호	이름	생일	나이	연령구분
E01	이소미	1985-12-05	38	중년층
E02	배재율	1973-02-17	50	중년층
E03	유대현	1988-08-27	35	중년층
E04	최소인	1987-09-17	36	중년층

Chapter 11

윈도우 함수



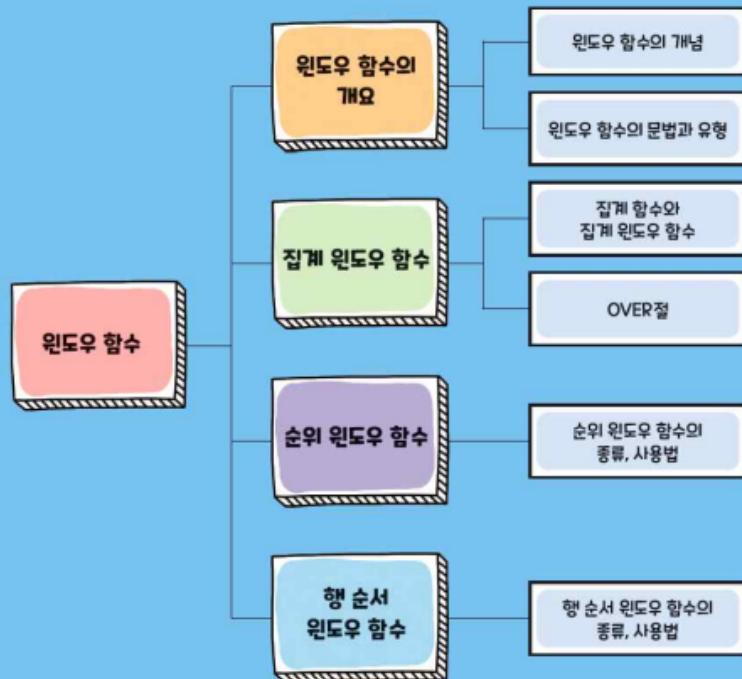
목차

1. 윈도우 함수의 개요
2. 집계 윈도우 함수
3. 순위 윈도우 함수
4. 행 순서 윈도우 함수

학습목표

- 원도우 함수의 개념과 사용법을 이해할 수 있습니다.
- 집계 원도우 함수를 이해하고 활용할 수 있습니다.
- 순위 원도우 함수를 이해하고 활용할 수 있습니다.
- 행 순서 원도우 함수를 이해하고 활용할 수 있습니다.

Preview



Section 01

원도우 함수의 개요

1. 원도우 함수의 개념

■ 원도우 함수(Window Function)

- 원도우 함수는 데이터 웨어하우스에서 발전한 기능으로, 분석 함수 또는 순위 함수로도 알려져 있음
- 원도우 함수는 데이터베이스 쿼리나 데이터 분석에서 주로 사용되는 함수임
- 데이터셋을 그룹화하거나 정렬한 후에 특정 연산을 적용하는 데 사용함
- 원도우란?
 - ✓ 연산이 수행되는 데이터셋의 부분 집합으로 지정한 분할 기준에 의해서 정의되며, 함수가 적용되는 범위를 지정하는 역할
- 원도우 함수는 OLAP과 같은 데이터 분석 분야에서 자주 사용됨

1. 원도우 함수의 개념

■ 원도우 함수(Window Function)

담당자직위	도시	고객번호	고객회사명	마일리지	순위	Avg() OVER()	Avg() OVER (PARTITION BY 도시)
영업 사원	부산광역시	NGOHU	케이티에스씨	8575	1	6948	4935
영업 사원	부산광역시	RICPE	뉴가든상사	1295	2	6948	4935
영업 사원	상주시	FKIAL	태월 산업	7772	1	6948	7772
영업 사원	서귀포시	VEASA	크릴부드	1114	1	6948	1114
영업 사원	서울특별시	HMSLE	대파유피스	85801	1	6948	10772
영업 사원	서울특별시	AUSBL	제한 인터내셔널	52652	2	6948	10772
영업 사원	서울특별시	NCHRA	쉐프스풀드	9589	3	6948	10772
영업 사원	서울특별시	UGBLA	금진 투역	8026	4	6948	10772
영업 사원	서울특별시	CTUCA	유성률산고 역	7955	5	6948	10772
영업 사원	서울특별시	DWWOOL	하이人寿 퍼퓸	7629	6	6948	10772
영업 사원	서울특별시	NDKWA	크리아유통	4364	7	6948	10772

그림 11-1 원도우 함수의 사용 예

2. 윈도우 함수의 문법과 유형

■ 윈도우 함수 사용

- 반드시 OVER절이 필요함
- OVER절은 윈도우 함수가 작동하는 윈도우를 정의하며, 파티션과 순서를 지정할 수 있음
- 윈도우 함수의 종류에 따라서 0개부터 n개의 인수(Arguments)가 지정될 수 있음
- 윈도우 함수의 유형

집계 윈도우 함수

- ✓ 순위 윈도우 함수
 - ✓ 행 순서 윈도우 함수
- 형식

```
SELECT 윈도우함수명(인수)
      OVER([PARTITION BY 컬럼] [ORDER BY 컬럼] [WINDOWING절])
FROM 테이블명;
```

Section 02

집계 윈도우 함수

1. 집계 함수와 집계 윈도우 함수

■ 집계 함수와 집계 윈도우 함수의 차이점

- 집계 함수에서 사용하는 GROUP BY절은 특정 컬럼을 기준으로 데이터를 집계하고자 할 때 사용함
 - ✓ GROUP BY절을 수행한 후에는 기존의 상세 데이터들은 확인할 수 없으며, 집계된 결과만 알 수 있음
- OVER절을 사용하여 정의하는 집계 윈도우 함수는 PARTITION BY절을 사용하여 그룹화함
 - ✓ GROUP BY와는 달리 기준 행의 데이터와 집계된 값을 함께 볼 수 있음
- 모든 집계 함수는 윈도우 함수로 사용할 수 있음

2. OVER절

■ 윈도우(파티션)

- 윈도우 함수는 결과를 생성하기 위해 입력으로 고려해야 하는 행의 일부를 OVER 절에서 정의함
- 예) 집계 함수를 사용하여 '부산광역시' 고객 테이블에서 평균마일리지 구하기

```
SELECT AVG(마일리지) AS 평균마일리지  
FROM 고객  
WHERE 도시 = '부산광역시';
```

▶ 실행결과

평균마일리지
4384.0000

■ [예제 11-1] '부산광역시' 고객에 대하여 고객번호, 고객회사명, 마일리지와 평균 마일리지를 함께 보이시오.

```
SELECT 고객번호  
, 고객회사명  
, 마일리지  
, AVG(마일리지) OVER() AS 평균마일리지  
FROM 고객  
WHERE 도시 = '부산광역시';
```

▶ 실행결과

고객번호	고객회사명	마일리지	평균마일리지
L1060	진출 상사	3101	4384.0000
LL114E	태산무역	887	4384.0000
NGOHU	케이티에스씨	8575	4384.0000
R1C1PE	뉴가든상사	1295	4384.0000
TTMBO	컬리풀스푸드	8062	4384.0000

2. OVER절

- [예제 11-2] '부산광역시' 고객에 대하여 고객번호, 고객회사명, 마일리지, 평균마일리지, 그리고 각 고객의 마일리지와 평균마일리지 간의 차이를 보이시오.

```
SELECT 고객번호
      ,고객회사명
      ,마일리지
      ,AVG(마일리지) OVER() AS 평균마일리지
      ,마일리지 - AVG(마일리지) OVER() AS 차이
  FROM 고객
 WHERE 도시 = '부산광역시';
```

▶ 실행결과

고객번호	고객회사명	마일리지	평균마일리지	차이
LIDBO	진흥상사	3101	4384.0000	-1283.0000
LLIWE	태산무역	887	4384.0000	-3497.0000
NGOHU	케이티에스씨	8575	4384.0000	4191.0000
RICPE	뉴가든상사	1295	4384.0000	-3089.0000
WWDNO	월드헬스푸드	5042	4384.0000	-658.0000

3,101 - 4,384

887 - 4,384

2. OVER절

■ PARTITION BY절

- 전체 집합을 특정 기준에 의해 소그룹으로 나누고자 할 때는 OVER절 내에서 PARTITION BY절을 사용함
- 예) 집계 함수를 사용하여 '경기도' 지역의 각 도시별로 평균 마일리지 확인하기

```
SELECT 도시
      ,AVG(마일리지) AS 평균마일리지
  FROM 고객
 WHERE 지역 = '경기도'
 GROUP BY 도시;
```

▶ 실행 결과

도시	평균마일리지
광명시	5046.5000
구리시	117.0000
광택시	134.0000
풀무천시	694.0000
부천시	3976.0000
고양시	5819.0000

2. OVER절

- [예제 11-3] '경기도' 고객에 대하여 고객번호, 도시, 마일리지, 도시의 평균마일리지, 그리고 각 고객의 마일리지와 도시의 평균마일리지 간 차이를 보이시오.

```
SELECT 고객번호
      ,도시
      ,마일리지 AS 고객마일리지
      ,AVG(마일리지) OVER(PARTITION BY 도시) AS 도시평균마일리지
      ,마일리지 - AVG(마일리지) OVER(PARTITION BY 도시) AS 차이
FROM 고객
WHERE 지역 = '경기도';
```

▶ 실행결과

고객번호	도시	고객마일리지	도시평균마일리지	차이
LYKFO	과천시	10468	5819.0000	4649.0000 → 10,468 - 5,819
RIBFU	과천시	1170	5819.0000	-4649.0000
AIHTR	광명시	9667	5046.5000	4620.5000 → 9,667 - 5,046.5
NSHCO	광명시	426	5046.5000	-4620.5000
ANRFR	구리시	117	117.0000	0.0000

2. OVER절

■ ORDER BY절

- OVER절에 ORDER BY절을 사용하면 정렬 기준을 지정할 수 있음
- 특히 SUM 윈도우 함수의 OVER절에 ORDER BY절을 넣으면 지정한 컬럼을 기준으로 누적 합을 구할 수 있음

■ [예제 11-4] '부산광역시' 고객에 대하여 고객번호, 마일리지, 모든 고객의 마일리지 합, 그리고 각 고객번호 순으로 마일리지의 누적 합을 보이시오.

```
SELECT 고객번호
      ,마일리지 AS 고객마일리지
      ,SUM(마일리지) OVER() AS 마일리지합
      ,SUM(마일리지) OVER(ORDER BY 고객번호) AS 누적마일리지
  FROM 고객
 WHERE 도시 = '부산광역시';
```

▶ 실행결과

고객번호	고객마일리지	마일리지합	누적마일리지
LIDBO	3101	21920	3101
LLIWE	887	21920	3988
NGOCHU	8575	21920	12563
RICPE	1295	21920	13858
TTMBO	8062	21920	21920

3,101 + 887

3,988 + 8,575

Section 03

순위 윈도우 함수

1. 순위 윈도우 함수의 종류

■ 순위 윈도우 함수(Rank Window Function)

- 행 간의 값을 비교하여 순위를 매기는 함수

표 11-1 순위 윈도우 함수의 종류

함수	설명	예
RANK()	결과 집합의 각 행에 대해 순위를 반환한다. RANK는 동률 순위에 대해 동일한 숫자 값을 제공한다.	1, 2, 2, 4, 5
DENSE_RANK()	결과 집합의 각 행에 대해 순위를 반환한다. DENSE_RANK는 바로 앞의 해당 특정 행에 대한 순위 값에 1을 더한 수를 제공한다.	1, 2, 2, 3, 4
ROW_NUMBER()	결과 집합의 각 행에 대해 순위를 반환한다. ROW_NUMBER는 모든 행에 대해 순차적으로 번호를 지정한다.	1, 2, 3, 4, 5
PERCENT_RANK()	백분율 순위 값을 의미한다. 파티션 내에서 행의 상대 순위를 계산하며 0과 1 사이의 값으로 표현된다.	0, 0.25, 0.5, 0.5, 1
CUME_DIST()	파티션 내에서 값의 누적 분포(Cumulative Distribution)를 계산한다. 즉, 파티션의 전체 건수에서 현재 행보다 작거나 같은 건수에 대한 누적 백분율을 의미한다.	0.2, 0.4, 0.8, 0.8, 1
NTILE(n)	정렬된 파티션의 행을 지정된 수의 그룹으로 나누며, 해당 행이 속한 그룹 번호를 반환한다. 파티션 별 전체 건수를 n등분하여 표현한다.	1, 1, 1, 2, 2

2. 순위 원도우 함수의 사용

■ RANK(), DENSE_RANK(), ROW_NUMBER()

- 순위를 반환하는 함수
 - ✓ ORDER BY 절에서 지정한 컬럼을 기준으로 순위를 매김
 - ✓ 세 개의 함수 모두 순위를 나타내지만 동일한 값이 있는 경우 결과가 다르게 반환됨
- 형식

```
RANK() OVER([PARTITION BY 컬럼] ORDER BY 컬럼)  
DENSE_RANK() OVER([PARTITION BY 컬럼] ORDER BY 컬럼)  
ROW_NUMBER() OVER([PARTITION BY 컬럼] ORDER BY 컬럼)
```



성적	순위		
	RANK()	DENSE_RANK()	ROW_NUMBER()
100	1	1	1
95	2	2	2
95	2	2	3
90	4	3	4
88	5	4	5

그림 11-2 함수별 동일한 값의 순위 처리 방식

2. 순위 원도우 함수의 사용

- [예제 11-6] '부산광역시' 고객에 대하여 마일리지가 많은 고객부터 순위를 매기시오.

```
SELECT 고객번호  
      ,고객회사명  
      ,담당자명  
      ,마일리지  
      ,RANK() OVER(ORDER BY 마일리지 DESC) AS 순위  
FROM 고객  
WHERE 도시 = '부산광역시';
```

▶ 실행결과

고객번호	고객회사명	담당자명	마일리지	순위
NGOHU	케이티에스씨	오유진	8575	1
TTMBO	컬러월스푸드	한수연	8062	2
LIDBO	진홀 상사	송아린	3101	3
RICPE	뉴가든상사	김지호	1295	4
LLIWE	태산무역	박소영	887	5

2. 순위 원도우 함수의 사용

■ PERCENT_RANK()

- 현재 행의 순위를 백분율로 나타낼 때 사용함
- 현재 행의 상대적 위치를 확인할 수 있음
- 형식

```
PERCENT_RANK() OVER([PARTITION BY 컬럼] ORDER BY 컬럼)
```

- 수식 : $(\text{순위} - 1) / (\text{원도우 행 수} - 1)$
- PERCENT_RANK() 함수의 결과 첫 행은 항상 0이고, 마지막 행은 항상 1이 됨



그림 11-3 PERCENT_RANK 함수가 사용되는 경우

2. 순위 원도우 함수의 사용

■ PERCENT_RANK()

- 예) 몸무게가 적을수록 높은 순위가 되도록 순위 백분율 값

표 11-2 몸무게에 대한 순위 백분율 계산 예

몸무게	RANK()	PERCENT_RANK()	수식
51	1	0	$(1 - 1) / (5 - 1)$
53	2	0.25	$(2 - 1) / (5 - 1)$
59	3	0.5	$(3 - 1) / (5 - 1)$
59	3	0.5	$(3 - 1) / (5 - 1)$
75	5	1	$(5 - 1) / (5 - 1)$

■ [예제 11-7] '시럽' 제품에 대해 단가를 기준으로 백분율 순위를 보이시오

```
SELECT 제품명
      ,단가
      ,PERCENT_RANK() OVER(ORDER BY 단가) AS 백분율순위
   FROM 제품
 WHERE 제품명 LIKE '%시럽';
```

▶ 실행 결과

제품명	단가	백분율순위
번 제리 시럽	1000	0
방그레 바닐라 시럽	1380	0.25
파블로바 피넛 시럽	1700	0.5
베리 100% 파인애플 시럽	2100	0.75
베리 100% 복숭아 시럽	2200	1

2. 순위 원도우 함수의 사용

■ CUME_DIST()

- 누적 분포를 나타낼 때 사용함
- 해당 행의 값이 윈도우 내에서 어느 위치에 있는지에 대한 상대적 위치를 누적 백분율로 표현함
- 결괏값의 범위 : $0 < \text{결괏값} \leq 1$
- 형식

```
CUME_DIST() OVER([PARTITION BY 컬럼] ORDER BY 컬럼)
```

- 예) 몸무게를 기준으로 누적 분포

표 11-3 몸무게에 대한 누적 분포 예시

몸무게	CUME_DIST()	수식
51	0.2	1 / 5
53	0.4	2 / 5
59	0.8	4 / 5
59	0.8	4 / 5
75	1	5 / 5

2. 순위 원도우 함수의 사용

- [예제 11-8] '부산광역시' 고객에 대하여 고객번호, 고객회사명, 마일리지 및 마일리지를 기준으로 한 누적 분포 값을 보이시오.

```
SELECT 고객번호
      ,고객회사명
      ,마일리지
      ,CUME_DIST() OVER(ORDER BY 마일리지) AS 누적분포
   FROM 고객
 WHERE 도시 = '부산광역시';
```

▶ 실행 결과

고객번호	고객회사명	마일리지	누적분포
LTIWE	태산무역	887	0.2
RICPE	뉴가든살사	1295	0.4
LIDBO	진출 상사	3101	0.6
TTMBO	컬러풀스풀드	8062	0.8
NGOHU	케이티에스씨	8575	1

2. 순위 원도우 함수의 사용

■ NTILE()

- 데이터를 특정 기준에 따라 그룹으로 나누고 차례대로 그룹 번호를 부여함
- PARTITION BY절을 생략하면 전체 행을 한 그룹으로 하여 계산함
- PARTITION BY절을 지정하면 해당 윈도우 내에서 그룹화를 진행하여 그룹 번호를 부여함
- ORDER BY절을 지정하면 해당 컬럼 순으로 정렬한 후에 그룹 번호를 부여함
- 형식

```
NTILE(n) OVER([PARTITION BY 컬럼] ORDER BY 컬럼)
```

2. 순위 원도우 함수의 사용

- [예제 11-9] '인천광역시' 고객에 대해 마일리지를 기준으로 하여 3개 그룹으로 나누시오

```
SELECT 고객번호  
      ,도시  
      ,마일리지  
      ,NTILE(3) OVER(ORDER BY 마일리지) AS 그룹  
FROM 고객  
WHERE 도시 = '인천광역시';
```

▶ 실행결과

고객번호	도시	마일리지	그룹
NTCCE	인천광역시	123	1
NSHER	인천광역시	132	1
ICKQU	인천광역시	943	1
CTEVI	인천광역시	1033	2
ENBKO	인천광역시	1384	2
OUTAR	인천광역시	7536	3
EBITH	인천광역시	89627	3

- [예제 11-10] 각 도시별로 고객에 대하여 마일리지를 기준으로 두 그룹씩 나누시오

```
SELECT 고객번호  
      ,도시  
      ,마일리지  
      ,NTILE(2) OVER(PARTITION BY 도시 ORDER BY 마일리지) AS 그룹  
FROM 고객;
```

▶ 실행결과

고객번호	도시	마일리지	그룹
RICPE	부산광역시	1295	1
LIDBO	부산광역시	3101	1
TTMBO	부산광역시	8062	2
NGOHU	부산광역시	8575	2
MSPTO	부천시	135	1
LIRSP	부천시	7817	2
	상주시		1

Section 04

행 순서 원도우 함수

1. 행 순서 윈도우 함수의 종류

■ 행 순서 윈도우 함수

- 특정 행의 위치에 있는 값을 가져올 때 사용함

표 11-4 행 순서 윈도우 함수의 종류

함수명	설명
FIRST_VALUE()	정렬된 값 집합의 첫 번째 값을 반환한다.
LAST_VALUE()	정렬된 값 집합의 마지막 값을 반환한다.
LAG()	명시된 값을 기준으로 이전 행의 값을 반환한다.
LEAD()	명시된 값을 기준으로 이후 행의 값을 반환한다.
NTH_VALUE(n)	윈도우의 n번째 행의 값을 반환한다.

2. 행 순서 윈도우 함수의 사용

■ FIRST_VALUE()

- 정렬된 컬럼 또는 수식 값 중 첫 번째 값을 반환함
- 형식

```
FIRST_VALUE(수식) OVER([PARTITION BY 컬럼] [ORDER BY 컬럼] [WINDOWING절])
```

■ [예제 11-11] '부산광역시' 고객에 대하여 고객 회사명, 마일리지, 최소 마일리지를 가진 고객회사명, 최소 마일리지, 마일리지와 최소 마일리지와의 차이를 보이시오.

```
SELECT 고객회사명  
      ,마일리지  
      ,FIRST_VALUE(고객회사명) OVER(ORDER BY 마일리지) AS 최소마일리지보유고객  
      ,FIRST_VALUE(마일리지) OVER(ORDER BY 마일리지) AS 최소마일리지  
      ,마일리지 - FIRST_VALUE(마일리지) OVER(ORDER BY 마일리지) AS 차이  
FROM 고객  
WHERE 도시 = '부산광역시';
```

▶ 실행결과

고객회사명	마일리지	최소마일리지보유고객	최소마일리지	차이	
태산무역	887	태산무역	887	0	887 - 887
뉴가든상사	1295	태산무역	887	408	1,295 - 887
진흥상사	3101	태산무역	887	2214	
컬러풀스튜드	8062	태산무역	887	7175	
케이티에스씨	8575	태산무역	887	7688	

2. 행 순서 윈도우 함수의 사용

- [예제 11-12] 각 도시별로 고객회사명, 마일리지와 최소 마일리지를 가진 고객회사명, 도시의 최소 마일리지, 그리고 고객의 마일리지와 최소마일리지 간의 차이를 보이시오.

```
SELECT 도시  
      ,고객회사명  
      ,마일리지  
      ,FIRST_VALUE(고객회사명) OVER(PARTITION BY 도시 ORDER BY 마일리지) AS 최소마일리지보  
      유 고객  
      ,FIRST_VALUE(마일리지) OVER(PARTITION BY 도시 ORDER BY 마일리지) AS 최소마일리지  
      ,마일리지 - FIRST_VALUE(마일리지) OVER(PARTITION BY 도시 ORDER BY 마일리지) AS 차이  
FROM 고객;
```

▶ 실행결과

도시	고객회사명	마일리지	최소마일리지보유고객	최소마일리지	차이
광주시	오키아식품	458	오키아식품	458	0
과천시	대원에스엠	1170	대원에스엠	1170	0
과천시	진영우역	10468	대원에스엠	1170	9298
광명시	준해식품	426	준해식품	426	0
광명시	씨엔그룹	9667	준해식품	426	9241
기타	남하종합식품	---	남하종합식품	117	0

1,170 - 1,170

10,468 - 1,170

2. 행 순서 윈도우 함수의 사용

■ LAST_VALUE()

- 정렬된 수식 값 중 마지막 값을 반환하는 함수
- 형식

```
LAST_VALUE(수식) OVER([PARTITION BY 컬럼] [ORDER BY 컬럼] [WINDOWING절])
```

■ [예제 11-13] '부산광역시' 고객에 대하여 고객회사명, 마일리지와 최대 마일리지를 가진 고객회사명, 최대 마일리지를 보이시오.

```
SELECT 고객회사명
      ,마일리지
      ,LAST_VALUE(고객회사명) OVER(ORDER BY 마일리지) AS 최대마일리지보유고객
      ,LAST_VALUE(마일리지) OVER(ORDER BY 마일리지) AS 최대마일리지
   FROM 고객
 WHERE 도시 = '부산광역시';
```

▶ 실행 결과

고객회사명	마일리지	최대마일리지보유고객	최대마일리지
태산무역	887	태산무역	887
뉴가든상사	1295	뉴가든상사	1295
진종 상사	3101	진종 상사	3101
컬러풀스튜드	8062	컬러풀스튜드	8062
케이티에스씨	8575	케이티에스씨	8575

2. 행 순서 윈도우 함수의 사용

■ [예제 11-13] '부산광역시' 고객에 대하여 고객회사명, 마일리지와 최대 마일리지를 가진 고객회사명, 최대 마일리지를 보이시오.

- 윈도우의 범위를 인수로 지정해주지 않았기 때문에 실행결과가 예상과 다름
 - ✓ OVER절에서 함수의 대상이 되는 행 기준의 범위를 명시적으로 지정하면 문제를 해결할 수 있음

표 11-5 범위 기준 옵션에 사용되는 키워드의 종류

키워드	설명
ROW	물리적인 단위로 행 집합을 지정한다. ORDER BY로 정렬된 컬럼 중 같은 값을 가지는 행이 있다면 묶어서 한번에 연산한다.
RANGE	논리적인 주소에 의해 행 집합을 지정한다. ORDER BY로 정렬된 컬럼에 대해 각 행마다 계산한다.
BETWEEN ~ AND ~	윈도우의 시작과 끝 위치를 지정한다.
UNBOUNDED PRECEDING	윈도우에서 첫 번째 행을 의미한다.
UNBOUNDED FOLLOWING	윈도우에서 마지막 행을 의미한다.
CURRENT ROW	윈도우에서 현재 행을 의미한다.

2. 행 순서 윈도우 함수의 사용

- [예제 11-13] '부산광역시' 고객에 대하여 고객회사명, 마일리지와 최대 마일리지를 가진 고객회사명, 최대 마일리지를 보이시오.
- 각 행마다 최대 마일리지를 보이도록 SQL 문장 수정

```
SELECT 고객회사명
      ,마일리지
      ,LAST_VALUE(마일리지) OVER(ORDER BY 마일리지
                                    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS 최대마일리지
   FROM 고객
 WHERE 도시 = '부산광역시';
```

▶ 실행 결과

고객회사명	마일리지	최대마일리지
태산루역	887	8575
뉴가든상사	1295	8575
진출 상사	3101	8575
컬러풀스푸드	8062	8575
케이티에스씨	8575	8575

2. 행 순서 윈도우 함수의 사용

■ LAG(), LEAD()

- 윈도우에서 이전 행의 값과 다음 행에 대한 값을 알고 싶을 때 사용함
- 형식

```
LAG(수식) OVER([PARTITION BY 컬럼] [ORDER BY 컬럼] [WINDOWING절])
```

```
LEAD(수식) OVER([PARTITION BY 컬럼] [ORDER BY 컬럼] [WINDOWING절])
```

■ [예제 11-14] 이전 행의 고객번호, 현재 행의 고객번호, 다음 행의 고객번호를 보이시오.

```
SELECT LAG(고객번호) OVER(ORDER BY 고객번호) AS 이전행고객번호  
      ,고객번호  
      ,LEAD(고객번호) OVER(ORDER BY 고객번호) AS 다음행고객번호  
FROM 고객;
```

▶ 실행결과

이전행고객번호	고객번호	다음행고객번호
ACDOR	ADHTR	ADHTR
ACDOR	ADHTR	ADHTR
ADHTR	ADHTR	ANKFR
ADHTR	ANKFR	ANKFR

2. 행 순서 윈도우 함수의 사용

■ NTH_VALUE()

- 윈도우에서 n번째 행의 값을 가져오기 위해 사용함
- 형식

```
NTH_VALUE(수식, 행위치) OVER([PARTITION BY 컬럼] [ORDER BY 컬럼] [WINDOWING절])
```

■ [예제 11-15] '부산광역시' 고객에 대하여 고객의 정보와 마일리지가 세 번째로 적은 고객의 정보를 함께 보이시오.

```
SELECT 고객번호  
      ,고객회사명  
      ,마일리지  
      ,NTH_VALUE(고객회사명, 3) OVER(ORDER BY 마일리지) AS "3번째로 마일리지가 적은 고객"  
      ,NTH_VALUE(마일리지, 3) OVER(ORDER BY 마일리지) AS "3번째 마일리지"  
FROM 고객  
WHERE 도시 = '부산광역시';
```

▶ 실행결과

고객번호	고객회사명	마일리지	3번째로 마일리지가 적은 고객	3번째 마일리지
LLIWE	태산무역	887	NULL	NULL
RICEPE	뉴가든상사	1295	NULL	NULL
LIDBO	진종 상사	3101	진종 상사	3101
TTMBO	컬러풀스튜드	8062	진종 상사	3101
NGOHU	케이티에스씨	8575	진종 상사	3101

점검문제

점검문제

문제 1

2021년도의 주문에 대해 월별 주문금액합과 월별 주문금액합에 대한 누적합을 보이시오.

▶ 실행 결과

주문월	주문금액합	누적주문금액합
1	5522100	5522100
2	4347000	9869100
3	4701300	14570400
4	5542500	20112900
5	5021000	25133900
6	7876700	33010600
7	4616100	37626700
8	6503100	44129800
9	10103900	54233700
10	11018800	65252500
11	8422300	73674800
12	16083400	89758200

문제 2

2021년도의 주문에 대해 분기별 주문건수, 첫 분기 주문건수, 마지막 분기 주문건수, 현 분기와 첫 분기 간의 주문 건수 차이, 현 분기와 마지막 분기 간 주문건수의 차이를 보이시오.

▶ 실행 결과

분기	주문건수	첫분기주문건수	마지막분기주문건수	첫분기와의차이	마지막분기와의차이
1	91	91	199	0	-108
2	106	91	199	15	-93
3	133	91	199	42	-66
4	199	91	199	108	0

문제 3

2021년도의 주문에 대해 월별로 당월 주문건수, 차월 주문건수, 차차월 주문건수를 보이시오.

▶ 실행 결과

주문월	당월주문건수	차월주문건수	차차월주문건수
1	30	28	33
2	28	33	34
3	33	34	35
4	34	35	37
5	35	37	36
6	37	36	42
7	36	42	55
8	42	55	53
9	55	53	69
10	53	69	77
11	69	77	NONE
12	77	NONE	NONE

Chapter 12

데이터 모델링



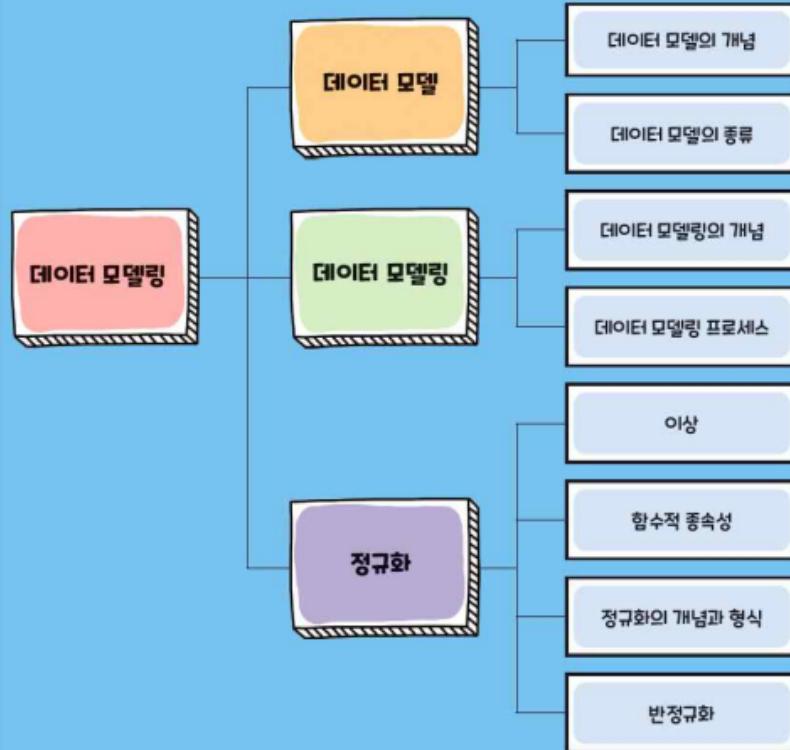
목차

1. 데이터 모델의 개요
2. 데이터 모델링의 개요
3. 정규화

학습목표

- 데이터 모델의 개념과 종류를 이해할 수 있습니다.
- 데이터 모델링의 개념과 프로세스를 이해할 수 있습니다.
- 이상과 정규화의 개념을 살펴보고 실습을 통해 정규화 과정을 직접 수행할 수 있습니다.

Preview



Section 01

데이터 모델의 개요

1. 데이터 모델의 개념

■ 데이터 모델(Data Model)

- 현실 세계를 추상화하여 사람, 장소, 사물, 범주 등에 대한 데이터 구조와 이들 간의 관계를 시각적으로 표현한 것
- 데이터베이스 시스템에서 사용되는 데이터의 논리적 구조를 정의함

■ 데이터 모델링(Data Modeling)

- 데이터 모델을 개발하기 위한 과정으로 현실 세계의 데이터를 추상화하여 데이터 모델로 변환하는 작업
- 데이터 모델을 통해 기업은 비즈니스 프로세스의 맥락에서 데이터를 정의하고 구조화함으로써 효과적인 정보 시스템을 개발할 수 있음

2. 데이터 모델의 종류

■ 데이터 모델의 종류

- 계층적 데이터 모델
- 네트워크 데이터 모델
- 개체-관계형 데이터 모델
- 관계형 데이터 모델
- 차원 데이터 모델
- 객체지향 데이터 모델
- 그래프 데이터 모델

2. 데이터 모델의 종류

■ 계층적 데이터 모델(Hierarchical Data Model)

- 상위 및 하위 레코드에 대하여 트리(Tree) 구조의 배열로 데이터를 구성함
- 자식 레코드는 하나의 부모만 가질 수 있으며 일대다(1:N)로 모델링됨
- 계층적 접근 방식은 메인프레임 데이터베이스에서 시작되었음
- 대부분 관계형 데이터 모델로 대체됨

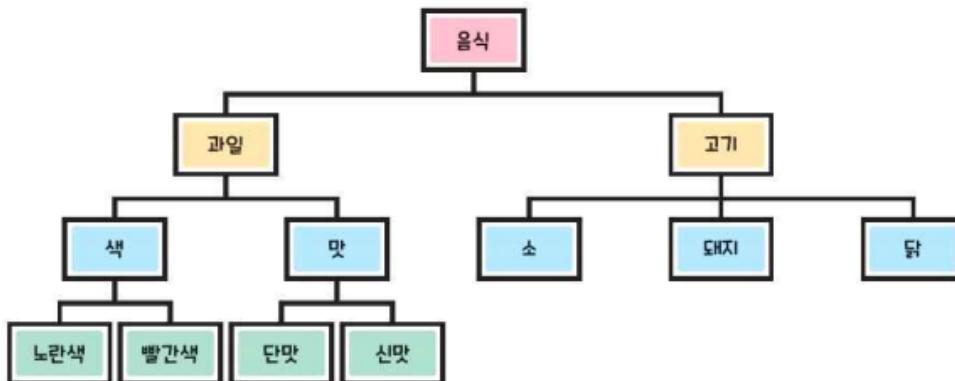


그림 12-1 계층적 데이터 모델의 예

2. 데이터 모델의 종류

■ 네트워크 데이터 모델(Network Data Model)

- 계층적 데이터 모델이 확장된 형태로 하위 레코드를 여러 상위 레코드에 연결하여 다대다(N:M)로 모델링이 가능한 모델
- 네트워크 데이터 모델을 CODASYL 모델이라고도 함



그림 12-2 네트워크 데이터 모델의 예

2. 데이터 모델의 종류

■ 개체-관계형 데이터 모델(Entity-Relationship Data Model)

- 지정된 소프트웨어 시스템에 대한 데이터 요소 및 해당 관계를 정의하는 상위 수준의 데이터 모델
- 개체-관계형 데이터 모델은 커뮤니케이션 도구로 사용될 수 있음
- 개체(Entity), 속성(Attribute) 및 관계(Relationship)를 ER 다이어그램(ERD)과 같이 시각적으로 표현함으로써 개체 간의 관계를 쉽게 이해할 수 있음

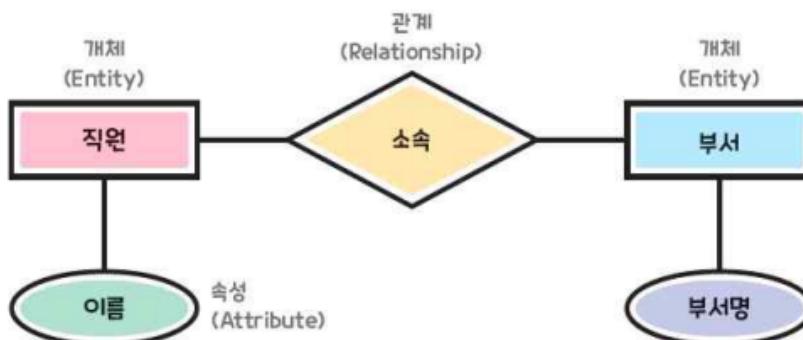


그림 12-3 개체-관계형 데이터 모델의 예

2. 데이터 모델의 종류

■ 관계형 데이터 모델(Relational Data Model)

- 계층 및 네트워크 데이터 모델에 대한 보다 유연한 대안으로 만들어짐
- 테이블 간의 관계를 나타내는 데 사용되며, 개체-관계형 데이터 모델에 비해 테이블 간의 관계를 도출하기가 좀 더 쉬움
- 관계형 데이터 모델에서 데이터는 테이블(릴레이션(Relation))에 저장됨
 - ✓ 테이블은 최소 0개 이상의 행(Row)과 1개 이상의 열(Column)로 구성됨
 - ✓ 열은 테이블의 항목들로 필드(Field), 속성(Attribute)이라고도 함
 - ✓ 행은 관련된 열의 집합으로 레코드(Record) 또는 튜플(Tuple)이라고도 함
 - ✓ 테이블의 각 레코드는 기본키(Primary Key) 속성에 의해 구별됨

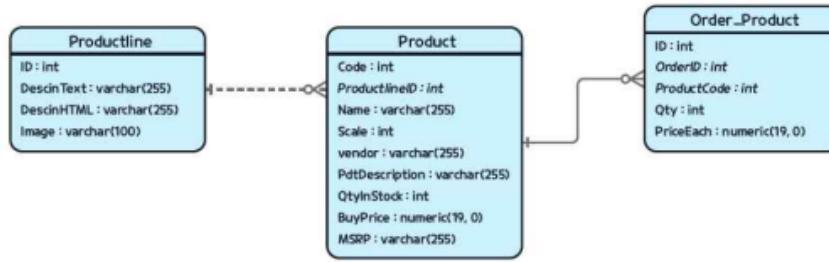


그림 12-4 관계형 데이터 모델의 예

2. 데이터 모델의 종류

■ 차원 데이터 모델(Dimensional Data Model)

- 주로 비즈니스 인텔리전스 애플리케이션을 지원하는 데이터 웨어하우스나 데이터 마트에서 사용됨
- 차원 데이터 모델은 비즈니스 측정값을 포함하는 팩트(Fact) 테이블과 비즈니스 관점이나 분석 대상을 포함하는 차원(Dimension) 테이블로 구성됨

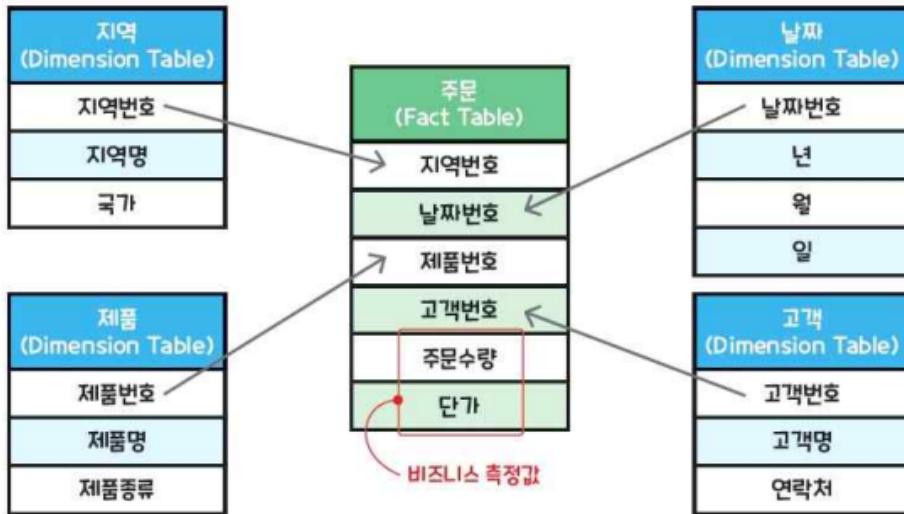


그림 12-5 차원 데이터 모델의 예

2. 데이터 모델의 종류

■ 객체지향 데이터 모델(Object Oriented Data Model)

- 객체를 하나 이상의 속성(Property)과 동작(Method)을 갖는 객체(Object)로 모델링함
- 속성과 동작이 동일한 여러 객체는 클래스로 그룹화할 수 있으며, 새 클래스는 기존 클래스의 속성과 동작을 상속함
- 복잡한 구조의 정보 모델링이 가능하다는 장점에도 안정성과 성능이 떨어져 지금은 특수한 전문 분야를 제외하고는 잘 쓰이지 않음

2. 데이터 모델의 종류

■ 그래프 데이터 모델(Graph Data Model)

- 네트워크 및 계층형 데이터 모델로부터 파생된 모델
- 주로 소셜 네트워크, 추천 엔진 및 사기 탐지 애플리케이션 등 복잡한 관계가 포함된 데이터 집합을 표현하는 데 사용됨
 - ✓ 노드(Node): 그래프 데이터 모델은 개별 엔티티를 나타냄
엣지(Edge): 노드 간의 관계를 나타냄
 - ✓ 속성(Property): 노드의 특성
 - ✓ 필요에 따라서 새로운 노드와 엣지를 그래프에 추가할 수 있음
- 특정 비즈니스 요구사항을 수용하기 위해 유연하고 쉽게 모델링할 수 있고, 데이터 내의 관계를 이해하기 쉬워 복잡한 관계의 데이터 집합과 실시간 쿼리가 필요한 애플리케이션에 적합함

2. 데이터 모델의 종류

■ 그래프 데이터 모델(Graph Data Model)

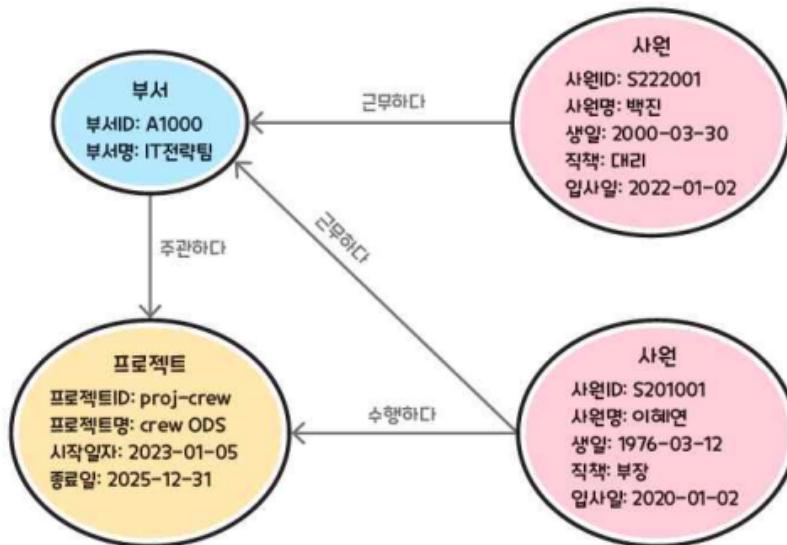


그림 12-6 그래프 데이터 모델의 예

Section 02

데이터 모델링의 개요

1. 데이터 모델링의 개념

■ 데이터 모델링

- 현실 세계의 엔티티(개체), 속성, 관계를 분석하고 이를 데이터 모델로 변환하는 프로세스
- 데이터 모델링 과정에서는 비즈니스 요구사항을 이해하고 데이터 요구사항을 수집하며, 데이터의 구조와 관계를 정의하고 문서화함
- 데이터 모델링은 데이터 모델의 설계, 개발, 유지보수를 위한 체계적인 방법론을 사용하여 진행됨
- 데이터 모델링은 데이터베이스 설계와 밀접한 관련이 있으며 데이터베이스 시스템의 성능, 효율성 및 유지보수 용이성에 영향을 미침

1. 데이터 모델링의 개념

■ 데이터 모델링 프로세스

- ① 요구사항 분석
- ② 개념적 데이터 모델링
- ③ 논리적 데이터 모델링
- ④ 물리적 데이터 모델링
- ⑤ 구현 및 유지보수
- ⑥ 검증 및 최적화
- ⑦ 문서화

2. 요구사항 분석

■ 요구사항 분석

- 데이터 모델링의 시작 단계
- 시스템 또는 응용프로그램에서 필요한 사용자 및 시스템의 요구사항을 정의하고 문서화하는 단계
- 주요 산출물
 - ✓ 요구사항 분석서, 인터뷰 기록 문서 등

3. 개념적 데이터 모델링

■ 개념적 데이터 모델링(Conceptual Data Modeling)

- 요구사항 분석 결과를 기반으로 개념적 데이터 모델을 구축함
- 특정 DBMS와 독립적인 개념적 스키마를 기술하는 단계로 업무 중심적이고 포괄적인 수준의 모델링을 진행함
- 엔티티와 관계 위주의 모델링을 통해 전반적인 골격을 파악하는 데 중점을 둠
- 이 단계에서 주제 영역 및 주요 엔티티, 주요 속성, 식별자와의 관계를 도출함
- 개념적 데이터 모델은 ERD 등으로 표현할 수 있으며, 개념적 데이터 모델로 표현한 결과를 '개념적 스키마'라고 함

3. 개념적 데이터 모델링

■ 주요 속성 도출

- 속성(Attribute): 엔티티 집합에 포함되는 최소의 데이터 단위
- 엔티티 집합의 각 엔티티는 각각의 속성에 대해 단일 속성값을 가져야 함

■ 식별자 도출

- 식별자(Identifier): 하나의 엔티티 집합에서 각 엔티티를 구분할 수 있는 속성이나 속성의 집합
- 엔티티 집합은 반드시 식별자를 가져야 함

3. 개념적 데이터 모델링

■ 관계 도출

- 관계(Relationship): 엔티티 간의 관련성을 의미하는 것으로 존재 관계와 행위 관계로 구분할 수 있음
- 존재 관계(Existence Relationship)
 - ✓ 두 엔티티 간의 소속이나 소유의 관계
 - ✓ [예] 사원과 가족, 학과와 학생, 은행과 계좌 엔티티 간의 관계
- 행위 관계(Behavioral Relationship)
 - ✓ 두 엔티티 간의 상호작용이나 동작의 관계
 - ✓ [예] 교수와 학생, 고객과 주문 엔티티 간의 관계

3. 개념적 데이터 모델링

■ 관계 도출

- 관계의 카디널리티(Cardinality, Degree)
 - ✓ 두 테이블 간의 관계에서 카디널리티는 대응수라고도 함
 - ✓ 한 테이블의 행이 다른 테이블에서 두 개 이상의 행과 연결될 수 있는지의 여부를 나타냄
 - ✓ 종류: 일대일(1:1), 일대다(1:N), 다대일(N:1), 다대다(N:M)

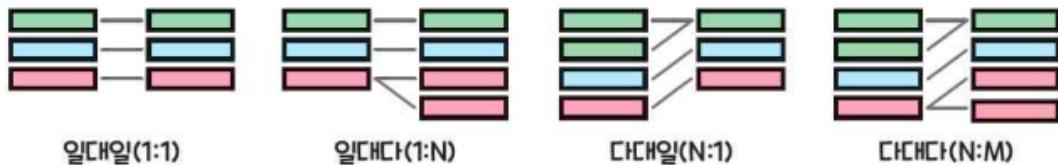


그림 12-9 카디널리티의 종류

3. 개념적 데이터 모델링

■ 관계 도출

- 관계 이름(Membership)
 - ✓ 객체 간에 관계가 맺어지는 형태를 의미함
 - ✓ 관계 시작점(The Beginning): 관계가 시작되는 쪽의 엔티티
 - ✓ 관계 끝점(The End): 관계가 끝나는 쪽의 엔티티
- 관계 선택 사양(Optionality)
 - ✓ 선택적(Optional) 참여 관계: 두 엔티티의 인스턴스 간의 관계가 선택적으로 존재할 수 있음
 - ✓ 필수적(Mandatory) 참여 관계: 두 엔티티의 인스턴스 간에 관계가 반드시 존재해야 함

3. 개념적 데이터 모델링

하나 더 알기 Y

ERD 표기법

ERD 표기법에는 피터 첸(Peter Chen) 표기법, IE(Information Engineering) 표기법, 바커(Baker) 표기법 등이 있습니다. IE 표기법은 관계를 나타내는 선의 끝 모양이 새의 발 모양처럼 생겼다고 하여 새발 표기법(Crow's Foot Notation)이라고도 합니다.

피터 첸 방식의 ERD 표기법은 직관적으로 이해하기는 쉬우나 실무에서는 거의 사용하지 않습니다. 주요 표기법은 다음과 같습니다.

표 12-1 ERD 주요 표기법(피터 첸 방식)

의미	기호
객체 타입	
키 속성	
속성	
관계	

3. 개념적 데이터 모델링

학과에 소속된 학생이 과목을 수강신청하는 시나리오에 대해 엔티티 간의 관계를 피터 천 표기법으로 표현한 개념적 데이터 모델 ERD의 예는 다음과 같습니다.

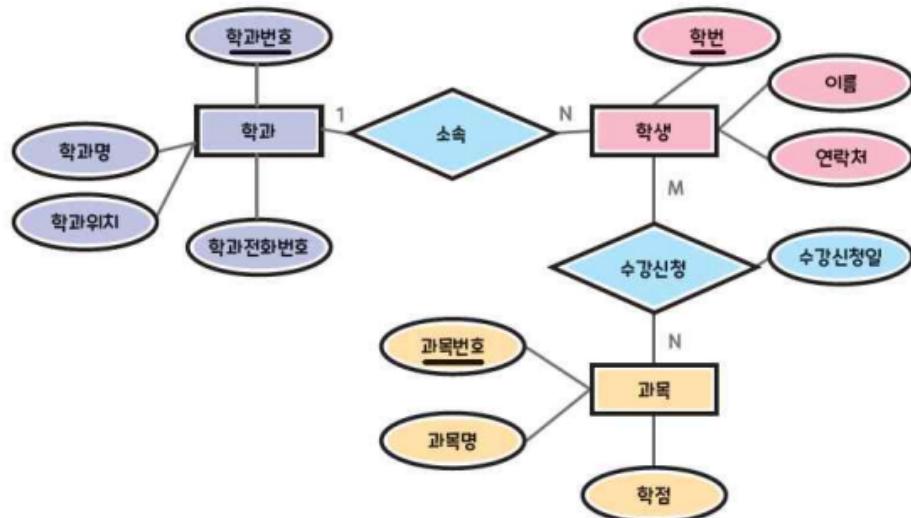


그림 12-10 피터 천 표기법으로 표현한 개념적 데이터 모델 ERD의 예

3. 개념적 데이터 모델링

IE 표기법은 실무에서 가장 많이 사용하는 ERD 표기법 중 하나입니다. IE 표기법은 논리적 데이터 모델이나 물리적 데이터 모델을 작성하는 데에도 사용됩니다. [표 12-2]는 IE의 주요 표기법을 보여줍니다.

표 12-2 ERD 주요 표기법(IE 방식)

관계	선택 여부	IE 표기법
1:1	필수	+ — +
1:1	선택	+ — o-
1:N	필수	+ — ->
1:N	선택	+ — o->



4. 논리적 데이터 모델링

■ 논리적 데이터 모델링(Logical Data Modeling)

- 개념적 데이터 모델을 더 상세하게 정의하는 단계
- 개념적 데이터 모델을 개발에 사용하는 특정 DBMS에 적합한 모델로 변환함
- DBMS가 관계형 데이터 모델인 경우에는 개념적 스키마를 릴레이션 스키마 등으로 변환해야 함
- 릴레이션 스키마와 같이 논리적 데이터 모델로 표현된 결과를 '논리적 스키마'라고 함

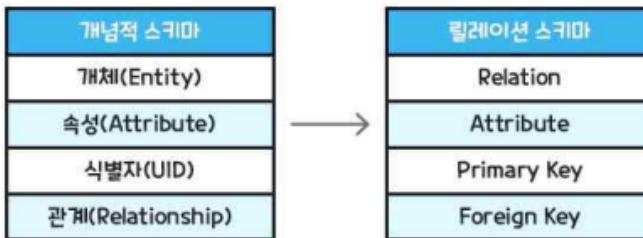


그림 12-11 개념적 데이터 모델을 관계형 데이터 모델로 매핑

4. 논리적 데이터 모델링

■ 논리적 데이터 모델링(Logical Data Modeling)

- 개념적 스키마를 릴레이션 스키마로 변환하기 위한 규칙
 - ✓ 모든 엔티티는 릴레이션으로 변환함
 - ✓ 일대일 관계(1:1)는 외래키로 표현함
 일대다 관계(1:N)는 외래키로 표현함
 - ✓ 다대다 관계(N:M)는 릴레이션으로 변환함

4. 논리적 데이터 모델링

■ 논리적 데이터 모델링(Logical Data Modeling)

- 규칙을 적용하여 [그림 12-10]의 개념적 데이터 모델을 릴레이션 스키마로 변환한 결과

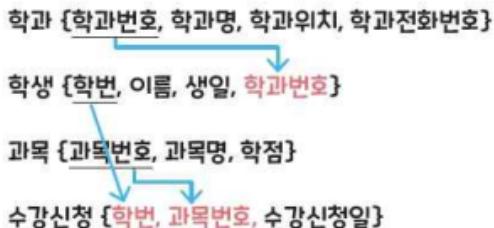


그림 12-12 개념적 데이터 모델을 릴레이션 스키마로 변환한 예

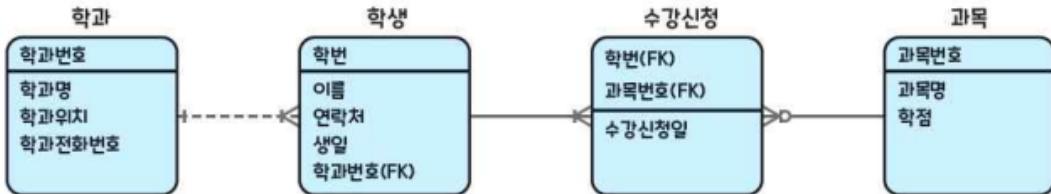


그림 12-13 IE 표기법을 사용하여 ERD로 표현한 논리적 데이터 모델의 예

4. 논리적 데이터 모델링

■ 논리적 데이터 모델링(Logical Data Modeling)

- 식별 관계와 비식별 관계

- ✓ [그림 12-13]의 점선: 비식별 관계, 실선: 식별 관계

이 두 관계는 부모 릴레이션과 자식 릴레이션 간의 관계를 나타냄

식별 관계는 부모 릴레이션의 기본키가 자식 릴레이션의 기본키의 일부로 사용되는 관계로 두 릴레이션 간의 속성이 서로에 대해 필수적일 때 적합함

비식별 관계는 부모 릴레이션의 기본키가 자식 릴레이션의 기본키에 포함되지 않는 관계로, 두 릴레이션 간의 관계가 서로에 대해 필수 정보가 아닐 때 적합함

5. 물리적 데이터 모델링

■ 물리적 데이터 모델링(Physical Data Modeling)

- 논리적 데이터 모델을 구체적인 데이터베이스 시스템에 맞게 변환함
- 테이블에 대해 기본키 컬럼 순서, 영문 컬럼명 매핑, 데이터타입 및 크기 등을 결정함
- 또한 인덱스, 참조 무결성 규칙 등의 제약조건, 파티셔닝과 같은 세부 사항을 정의하고, 저장 구조와 접근 경로 등을 결정함

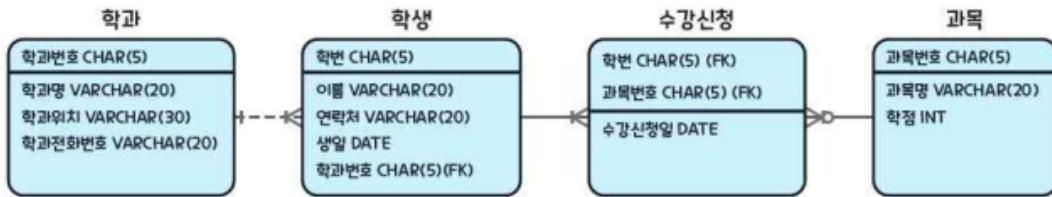


그림 12-14 물리적 데이터 모델의 예

6. 구현 및 유지보수

■ 구현 단계 및 유집보수

- 물리적 데이터 모델을 기반으로 데이터베이스를 생성함
- 이후 데이터베이스 시스템을 운영하고 유지보수하며, 필요에 따라 모델을 조정하거나 확장함
- 주요 산출물: 데이터베이스 데이터베이스 객체 정의문, 데이터 마이그레이션 계획서 등

7. 검증 및 최적화, 문서화

■ 검증 및 최적화, 문서화

- 데이터 모델링 단계가 완료된 후에는 모델의 검증과 최적화를 수행함
- 데이터 모델의 정확성, 일관성, 성능 등을 평가하고, 필요하다면 수정 및 개선 작업을 진행함
- 문서화는 각 단계에서 산출된 문서를 정리하여 모델을 이해하고 유지보수할 수 있도록 만들어줌

Section 03

정규화

1. 이상

■ 이상(Anomaly)

- 관계형 데이터베이스에서 데이터 삽입, 삭제 및 수정과 관련하여 발생할 수 있는 불일치 또는 오류
- 데이터베이스의 정상적인 동작을 방해하거나 데이터의 무결성과 일관성을 저해할 수 있음

■ 이상의 유형

- 삽입 이상(Insertion Anomaly)
 - ✓ 데이터를 삽입할 때 발생하는 이상으로 데이터를 삽입할 때 원하지 않는 불필요한 데이터를 반복해서 입력해야 하는 경우
- 삭제 이상(Deletion Anomaly)
 - ✓ 데이터를 삭제할 때 발생하는 이상으로 테이블에서 일부 레코드를 삭제하면 삭제할 필요가 없는 다른 정보도 함께 삭제되는 경우
- 수정 이상(Update Anomaly)
 - ✓ 데이터를 변경할 때 발생하는 이상으로 테이블의 일부 레코드의 값만 수정되어 데이터 일관성이 깨지는 경우

1. 이상

- [예] 동아리, 학생, 학과 정보를 한번에 관리하도록 설계된 동아리가입학생학과 테이블

동아리번호	동아리명	동아리개설일	학번	이름	동아리가입일	학관번호	학과명
C1	지구한바퀴여행	2000-02-01	231001	문지영	2023-04-01	D1	화학공학과
C1	지구한바퀴여행	2000-02-01	231002	배경민	2023-04-03	D4	경영학과
C2	클래식연주동아리	2010-06-05	232001	김명희	2023-03-22	D2	동양학과
C3	워너비골프	2020-03-01	232002	전은정	2023-03-07	D2	동양학과
C3	워너비골프	2020-03-01	231002	배경민	2023-04-02	D4	경영학과
C4	댄스밸류	2021-07-01	231001	문지영	2023-04-30	D1	화학공학과
C4	댄스밸류	2021-07-01	233001	이현경	2023-03-27	D3	역사학과

그림 12-15 기본키가 {동아리번호, 학번}인 정규화되지 않은 동아리가입학생학과 테이블

2. 함수적 종속성

■ 함수적 종속성(Functional Dependency, FD)

- 관계형 데이터베이스의 릴레이션에서 속성 간의 관계
- 특정 릴레이션에서 속성 X와 속성 Y 사이에 함수적 종속성이 존재한다
 - ✓ = X의 값이 주어지면 Y의 값이 항상 유일하게 결정된다
 - ✓ $X \rightarrow Y$
 - ✓ X: 결정자(Determinant), Y: 종속자(Independent)

2. 함수적 종속성

■ 함수적 종속성(Functional Dependency, FD)

- 함수적 종속성 다이어그램(Functional Dependency Diagram, FDD)
 - ✓ 함수적 종속성을 나타내는 표기법
 - ✓ 이것을 통해 속성 간의 관계를 쉽게 파악할 수 있음

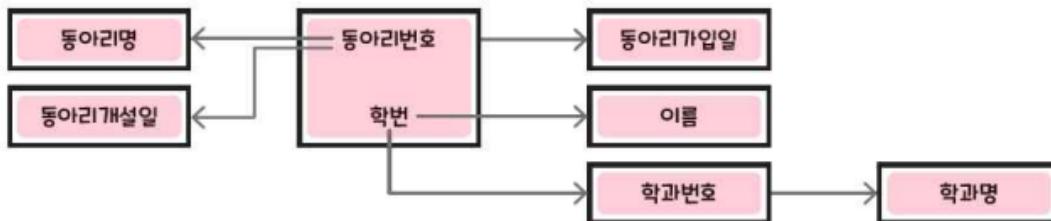


그림 12-16 동아리가입학생학과 테이블의 함수적 종속성 다이어그램

동아리번호 → 동아리명
동아리번호 → 동아리개설일
{동아리번호, 학번} → 동아리가입일
학번 → 이름
학번 → 학과번호
학과번호 → 학과명

2. 함수적 종속성

■ 완전 함수적 종속성(Full Functional Dependency)

- 기본키가 아닌 속성들이 기본키에 종속이 되는 종속 관계
- 만일 기본키가 복합키인 경우에는 기본키를 구성하는 속성 집합에 종속되어야 함

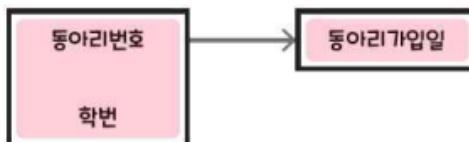


그림 12-17 완전 함수적 종속성의 예

■ 부분 함수적 종속성(Partial Functional Dependency)

- 기본키가 아닌 속성들이 기본키의 일부에 종속되는 관계

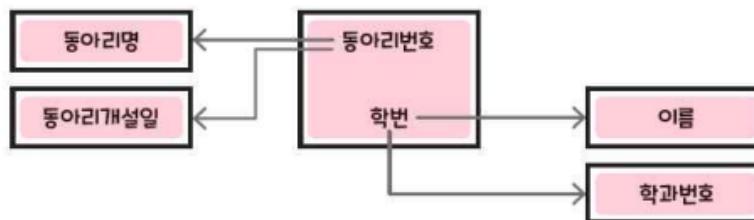


그림 12-18 부분 함수적 종속성의 예

2. 함수적 종속성

■ 이행 함수적 종속성(Transitive Functional Dependency)

- 세 개 이상의 속성이 연결되어 있을 때 발생하는 종속 관계
- X, Y, Z 라는 3개의 속성이 있을 때 $X \rightarrow Y, Y \rightarrow Z$ 와 같이 속성 간에 종속성이 연쇄적으로 전파되는 경우
- Y 가 X 에 함수적으로 종속되고, Z 가 Y 에 함수적으로 종속되면, Z 는 X 에 대해 이행 함수적으로 종속됨



그림 12-19 이행 함수적 종속성의 예

3. 정규화의 개념과 형식

■ 데이터베이스 정규화(Normalization)

- 데이터베이스 설계의 중요한 개념으로 이상을 제거하여 데이터베이스를 효율적으로 구조화하는 과정
- 정규화를 통해 데이터베이스를 구조화하면 데이터 삽입 이상, 수정 이상, 삭제 이상이 최소화됨
- 그 결과 데이터의 중복을 줄일 수 있고, 데이터 무결성을 강화하며 일관성을 유지할 수 있음
- 또한 데이터의 논리적 구조가 좀 더 간결하고 명확하게 표현되므로 데이터를 검색하는 데 효율적이며, 유지보수성이 향상됨

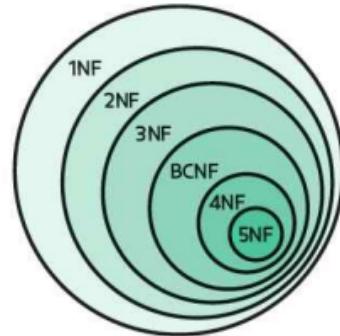


그림 12-20 정규화 단계

3. 정규화의 개념과 형식

■ 제1정규형(1NF)

- 테이블의 각 속성 값은 반복 그룹이 없는 원자 값으로만 구성되어야 함
- 이를 위해서 중복되는 속성이나 속성 그룹은 별도의 테이블로 분리함

■ 제2정규형(2NF)

- 제2정규형은 제1정규형을 만족하면서 부분 함수적 종속성을 제거해야 함

■ 제3정규형(3NF)

- 제3정규형은 제2정규형을 만족하면서 이행 함수적 종속성을 제거해야 함

■ 추가적인 정규화 형식

- 보이스-코드 정규형(Boyce-Codd NF, BCNF), 제4정규형(4NF), 제5정규형(5NF) 등
- 실무에서 데이터베이스 정규화는 주로 제3정규형까지 진행함

4. 반정규화

■ 반정규화(Denormalization)

- 중복 데이터를 하나 이상의 테이블에 추가하는 데이터베이스 최적화 기술
- 반정규화의 장점
 - ✓ 조인 개수를 줄일 수 있고, 이로 인해 검색 쿼리가 간단해짐
 - ✓ 빠른 읽기 작업의 수행으로 인해 검색 성능을 높일 수 있음
- 반정규화의 단점
 - ✓ 반정규화는 중복 저장으로 인해 더 많은 스토리지가 필요함
 - ✓ 데이터 불일치가 발생할 수 있음
 - ✓ 쓰기 작업에 추가적인 비용이 소요될 수 있음

4. 반정규화

■ [예제 12-1] 동아리가입학생학과 테이블에 대해 제1정규형을 적용하시오.

- 정규화 대상인 동아리가입학생학과 테이블

동아리번호	동아리명	동아리개설일	학번	이름	동아리가입일	학과번호	학과명
C1	자구한바퀴여행	2000-02-01	231001	문지영	2023-04-01	D1	화학공학과
C1	자구한바퀴여행	2000-02-01	231002	배경민	2023-04-03	D4	경영학과
C2	클라식연주동아리	2010-06-05	232001	길영희	2023-03-22	D2	통계학과
C3	워너비클래	2020-03-01	232002	전윤정	2023-03-07	D2	통계학과
C3	워너비클래	2020-03-01	231002	배경민	2023-04-02	D4	경영학과
C4	댄스바풀	2021-07-01	231001	문지영	2023-04-30	D1	화학공학과
C4	댄스바풀	2021-07-01	233001	이현경	2023-03-27	D3	역사학과

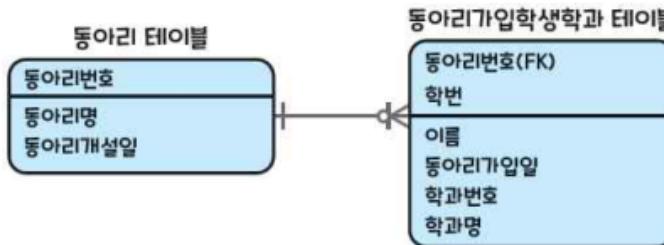
- 동아리가입학생학과 테이블

동아리번호
학번
동아리명
동아리개설일
이름
동아리가입일
학과번호
학과명

4. 반정규화

■ [예제 12-1] 동아리가입학생학과 테이블에 대해 제1정규형을 적용하시오.

- 제1정규형을 만족하려면 반복 속성 그룹을 별도의 테이블로 분할해야 함



- 분할된 동아리 테이블과 동아리가입학생학과 테이블 간의 카디널리티는 일대다

4. 반정규화

■ [예제 12-1] 동아리가입학생학과 테이블에 대해 제1정규형을 적용하시오.

- 동아리 테이블

동아리번호	동아리명	동아리개설일
C1	지구한바퀴여행	2000-02-01
C2	클래식연주동아리	2010-06-05
C3	워너비골퍼	2020-03-01
C4	댄스배틀	2021-07-01

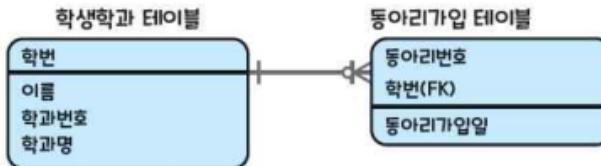
- 동아리가입학생학과 테이블

동아리번호	학번	이름	동아리가입일	학과번호	학과명
C1	231001	문지영	2023-04-01	D1	화학공학과
C1	231002	배경민	2023-04-03	D4	경영학과
C2	232001	김명희	2023-03-22	D2	통계학과
C3	232002	전은정	2023-03-07	D2	통계학과
C3	231002	배경민	2023-04-02	D4	경영학과
C4	231001	문지영	2023-04-30	D1	화학공학과
C4	233001	이현경	2023-03-27	D3	역사학과

4. 반정규화

■ [예제 12-2] 동아리가입학생학과 테이블에 대해 제2정규형을 적용하시오

- 제2정규형을 만족하기 위해서는 테이블 분할을 통해 부분 함수적 종속성을 제거해야 함



- 분할된 학생학과 테이블과 동아리가입 테이블 간의 카디널리티는 일대다

4. 반정규화

■ [예제 12-2] 동아리가입학생학과 테이블에 대해 제2정규형을 적용하시오

- 학생학과 테이블

학번	이름	학과번호	학과명
231001	문자영	D1	화학공학과
231002	배경민	D4	경영학과
232001	김명희	D2	통계학과
232002	전은정	D2	통계학과
233001	이현경	D3	역사학과

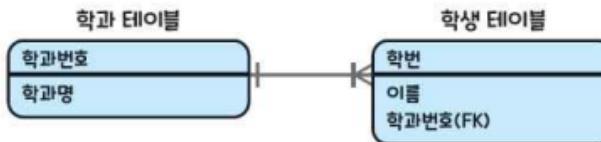
- 동아리가입 테이블

동아리번호	학번	동아리가입일
C1	231001	2023-04-01
C1	231002	2023-04-03
C2	232001	2023-03-22
C3	232002	2023-03-07
C3	231002	2023-04-02
C4	231001	2023-04-30
C4	233001	2023-03-27

4. 반정규화

■ [예제 12-3] 학생학과 테이블에 대해 제3정규형을 적용하시오

- 동아리가입 테이블은 학번 → 학과번호, 학과번호 → 학과명과 같이 속성의 종속성이 연쇄 발생하고 있어 테이블을 분할하여 종속성을 제거해야 함



- 분할된 학과 테이블과 학생 테이블 간의 카디널리티는 일대다
 - ✓ 이 경우에 학과 테이블의 기본키인 학과번호를 학생 테이블에 추가함
- 학과 테이블
- 학과 테이블

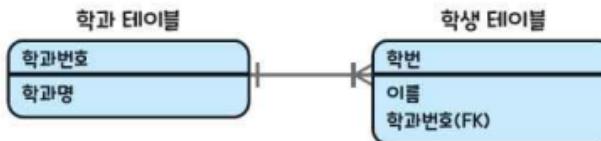
학과번호	학과명
D1	화학공학과
D2	통계학과
D3	역사학과
D4	경영학과

학번	이름	학과번호
231001	문지영	D1
231002	배경민	D4
232001	김명희	D2
232002	전은정	D2
233001	이현경	D3

4. 반정규화

■ [예제 12-3] 학생학과 테이블에 대해 제3정규형을 적용하시오

- 동아리가입 테이블은 학번 → 학과번호, 학과번호 → 학과명과 같이 속성의 종속성이 연쇄 발생하고 있어 테이블을 분할하여 종속성을 제거해야 함



- 분할된 학과 테이블과 학생 테이블 간의 카디널리티는 일대다
 - ✓ 이 경우에 학과 테이블의 기본키인 학과번호를 학생 테이블에 추가함

- 학과 테이블

학과번호	학과명
D1	화학공학과
D2	통계학과
D3	역사학과
D4	경영학과

- 학과 테이블

학번	이름	학과번호
231001	문지영	D1
231002	배경민	D4
232001	김명희	D2
232002	전은정	D2
233001	이현경	D3

4. 반정규화

■ [예제 12-3] 학생학과 테이블에 대해 제3정규형을 적용하시오

- 제3정규형을 만족하는 정규화 결과

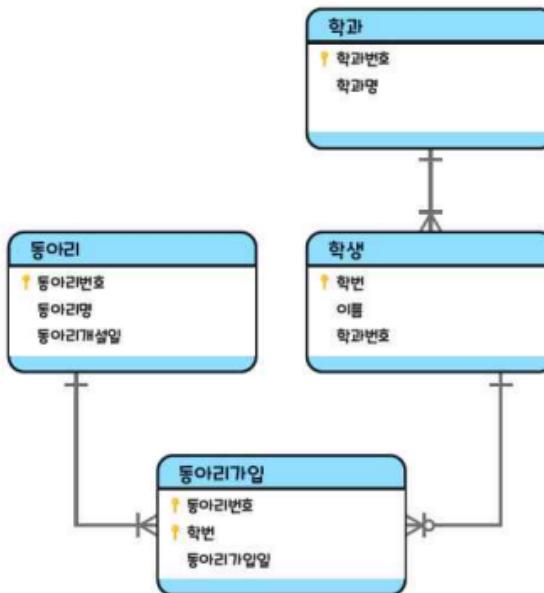


그림 12-21 제3정규형을 만족하는 정규화 결과

점검문제

문제 1

[예제 12-3]에서 도출된 논리적 모델에 대해 물리적 모델링을 수행한 후 DDL문을 사용하여 테이블을 생성하시오.
이때 데이터타입 및 크기는 임의로 지정하시오.

Chapter 13

[프로젝트]

공공데이터를 활용한
데이터 분석



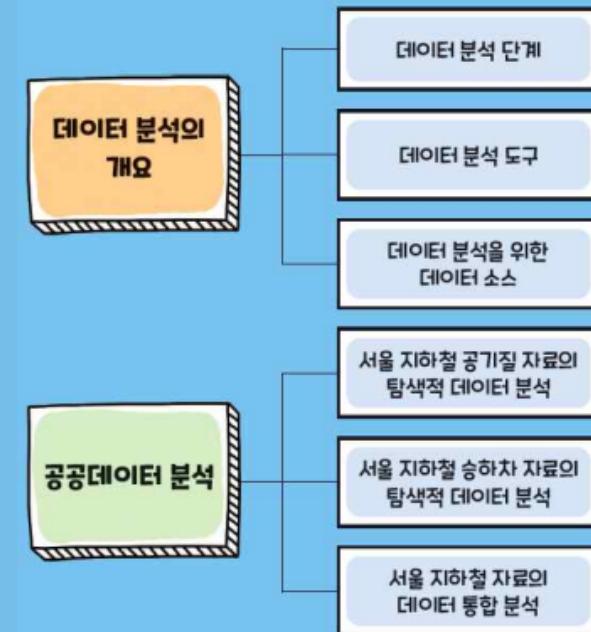
목차

1. 데이터 분석과 공공데이터
2. 공공데이터 전처리
3. 서울 지하철 공기질 자료의 탐색적 데이터 분석
4. 서울 지하철 승하차 자료의 탐색적 데이터 분석
5. 서울 지하철 자료의 데이터 통합 분석

학습목표

- 데이터 분석의 단계와 도구에 대해 이해할 수 있습니다.
- 공공데이터를 내려받고 전처리하는 과정을 이해할 수 있습니다.
- 공공데이터를 활용하여 서울 지하철 공기질과 서울 지하철 승하차 분석을 수행할 수 있습니다

Preview



Section 01

데이터 분석과 공공데이터

1. 데이터 분석 단계

■ 데이터 분석

- 다양한 데이터 분석 도구를 사용하여 데이터를 분석하고 해석하여 조직이 목표를 달성하는 데 도움이 되는 통찰력과 추세를 제시하는 것
- 잘 분석된 데이터는 조직이 자신있게 의사 결정을 내리고, 조직에 가치를 추가하고, 제품 및 서비스 개발에 더 나은 정보를 제공하는 데 도움을 줌

■ 데이터 분석 단계



그림 13-1 데이터 분석 단계

2. 데이터 분석 도구

■ 데이터 분석 도구

- 목적에 따라 통계 분석용 도구, 비즈니스 인텔리전스 도구, 프로그래밍 언어 기반 도구로 구분할 수 있음
- 통계 분석용 도구
 - ✓ 엑셀, R, SAS, SPSS 등
- 비즈니스 인텔리전스 도구
 - ✓ Power BI, 태블로(Tableau) 등
- 프로그래밍 언어 기반 도구
 - ✓ 파이썬, SQL 등

2. 데이터 분석 도구

■ 데이터 분석 도구 SQL

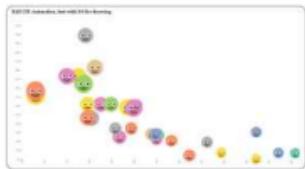
- 데이터 추출
- 데이터 필터링 및 정렬
- 집계 및 그룹화
- 조인 및 서브쿼리
- 데이터 변환 및 처리
- 데이터 시각화를 위한 기반 데이터 생성

2. 데이터 분석 도구

■ 데이터 시각화 분석 도구

표 13-1 주요 데이터 시각화 도구

시각화 도구	설명
Matplotlib, Seaborn	파이썬에서 사용되는 데이터 시각화 라이브러리
ggplot2	R언어에서 사용되는 데이터 시각화 라이브러리
D3.js	웹을 위한 자바스크립트 시각화 라이브러리
Power BI	マイクロ소프트의 비즈니스 인텔리전스 도구로 엔터프라이즈 및 클라우드 기반의 서비스 제공
Excel	스프레드시트 프로그램으로 차트 및 그래프 생성 지원
Tableau	엔터프라이즈 수준의 데이터 시각화 도구로 데이터 시각화, 대시보드 및 리포팅 지원
Redash	오픈 소스 데이터 시각화 도구
Google Data Studio	무료로 사용 가능한 클라우드 기반의 데이터 시각화 도구
Looker Studio	구글 클라우드에서 제공하는 데이터 시각화 및 비즈니스 인텔리전스 플랫폼으로 SQL 기반의 데이터 탐색과 시각화 지원
Amazon QuickSight	AWS(Amazon Web Services)에서 제공하는 데이터 시각화 및 비즈니스 분석 도구



(a) D3.js



(b) PowerBI



(c) Tableau

그림 13-2 데이터 시각화의 예

3. 데이터 분석을 위한 데이터 소스

■ 데이터 분석을 위한 데이터 소스

- 데이터 소스로부터 데이터를 얻을 때는 데이터의 품질, 출처의 신뢰성, 개인정보 보호 등을 고려해야 함
- 내부 데이터
- 외부 데이터
- 웹 스크래핑
- 사용자 생성 콘텐츠
- 업계 데이터
- 연구 데이터
- 애널리틱 데이터

4. 데이터 제공 사이트

■ 데이터 제공 사이트

표 13-2 국내 데이터 제공 사이트

사이트명	사이트	설명
공공데이터포털	https://www.data.go.kr/	한국 정부에서 운영하는 공공데이터 제공 포털로 다양한 분야의 데이터를 제공
국가통계포털	https://kosis.kr/	통계청에서 운영하는 국가 통계 포털
서울 열린데이터광장	https://data.seoul.go.kr/	서울시에서 운영하는 공공데이터 포털
K-ICT 빅데이터 센터	https://kbig.kr/portal/	과기부와 한국지능정보사회진흥원에서 지원하는 공식 사이트
통합데이터지도	https://www.bigdata-map.kr/	공공과 민간에서 제공하는 데이터를 쉽게 검색, 활용할 수 있도록 지원

표 13-3 해외 데이터 제공 사이트

사이트	설명
https://data.gov/	미국 정부에서 운영하는 미국의 공공데이터 포털
https://data.europa.eu/euodp/	유럽 연합에서 운영하는 오픈 데이터 포털
https://data.gov.uk/	영국 정부에서 운영하는 영국의 공공데이터 포털
https://www.kaggle.com/datasets	전 세계 다양한 분야의 방대한 데이터셋 공개
https://registry.opendata.aws/	아마존 AWS 데이터셋

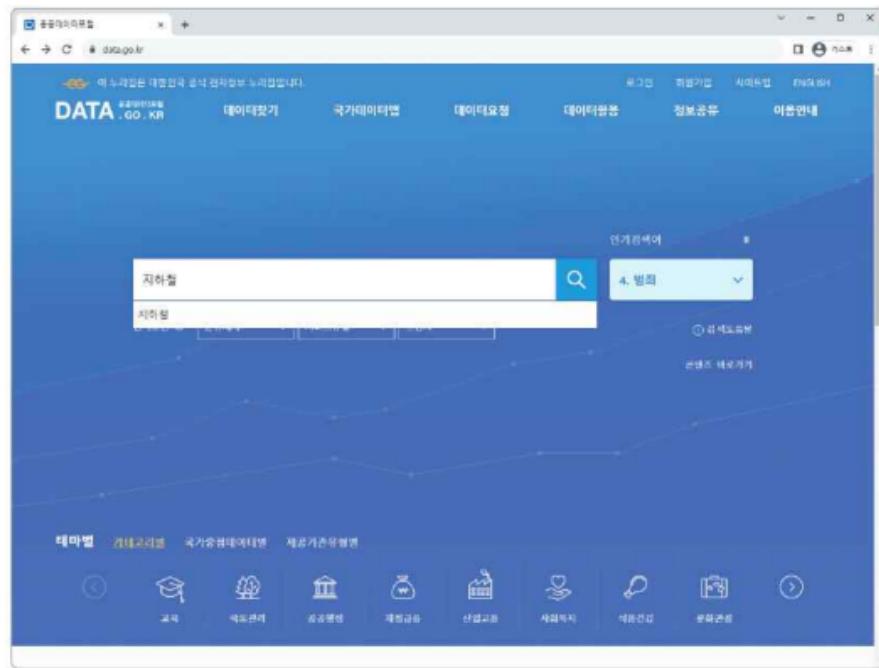
Section 02

공공데이터 전처리

1. 공공데이터포털에서 데이터셋 얻기

■ 공공데이터포털에서 데이터셋 얻기

- 공공데이터포털 사이트(<https://www.data.go.kr>)에 접속하여 '지하철' 검색



1. 공공데이터포털에서 데이터셋 얻기

■ 공공데이터포털에서 데이터셋 얻기

- [파일데이터] 탭을 클릭하여 '서울특별시_지하철 호선별 역별 승하차 인원 정보'의 [바로가기] 버튼 클릭
- 이후 서울 열린데이터 광장 페이지의 '파일내려받기'에서 버튼을 클릭하여 'CARD_SUBWAY_MONTH_2022.csv' 파일 다운로드

The screenshot shows the Seoul Open Data Portal interface. At the top, there's a search bar and a navigation menu. Below it, a main content area displays a specific dataset titled "서울시 지하철호선별 역별 승하차 인원 정보". This page includes a brief description, a preview of the data, and a large "파일내려받기" (File Download) button at the bottom. To the right, a separate window titled "파일내려받기" (File Download) is open, showing a table of file details. One row in the table is highlighted with a red border, and the "다운로드" (Download) button next to it is also highlighted with a red box. The table columns include ID, 번호 (Number), 파일명 (File Name), 사이즈 (Size), and 상태 (Status).

ID	번호	파일명	사이즈	상태
1	번호1	CARD_SUBWAY_MONTH_2022.csv	1.0M	다운로드
2	번호2	CARD_SUBWAY_MONTH_2020.csv	1.0M	다운로드
3	번호3	CARD_SUBWAY_MONTH_2021.csv	1.0M	다운로드
4	번호4	CARD_SUBWAY_MONTH_2020_06.csv	1.0M	다운로드
5	번호5	CARD_SUBWAY_MONTH_2020_07.csv	1.0M	다운로드
6	번호6	CARD_SUBWAY_MONTH_2020_08.csv	1.0M	다운로드
7	번호7	CARD_SUBWAY_MONTH_2020_09.csv	1.0M	다운로드
8	번호8	CARD_SUBWAY_MONTH_2020_10.csv	1.0M	다운로드
9	번호9	CARD_SUBWAY_MONTH_2020_11.csv	1.0M	다운로드
10	번호10	CARD_SUBWAY_MONTH_2020_12.csv	1.0M	다운로드
11	번호11	CARD_SUBWAY_MONTH_2021_01.csv	1.0M	다운로드
12	번호12	CARD_SUBWAY_MONTH_2021_02.csv	1.0M	다운로드

1. 공공데이터포털에서 데이터셋 얻기

■ 공공데이터포털에서 데이터셋 얻기

- [표준데이터셋] 탭을 클릭하여 '전국지하철공기질측정정보표준데이터' 클릭
 - 데이터 목록에서 '서울교통공사_지하철공기질측정정보'를 클릭하여 나타나는 데이터 상세정보 창에서 [CSV] 버튼을 클릭하여 CSV 파일 다운로드

2. 전처리하기

■ 지하철승하차 데이터의 전처리 수행 작업

- ① 등록일자 필드 삭제
 - ② 역명에 붙은 (역명) 삭제
 - ③ '노선명'을 '호선명'으로 변경
 - ④ 1~9호선 이외의 정보 삭제
 - ⑤ 사용일자를 날짜 타입으로 변환
 - ⑥ '승차총승객수'를 '승차승객수'로, '하차총승객수'를 '하차승객수'로 변경
- ✓ 전처리 전 파일명: CARD_SUBWAY_MONTH_2022.CSV

전처리 후 파일명: 지하철승하차.CSV

A	B	C	D	E	F	G
1	사용일자	노선명	역명	승차승승객수	하차승승객수	등록일자
2	2022/01/01	3호선	서울	7373	7076	2022/01/01
3	2022/01/01	8호선	인천	461	473	2022/01/01
4	2022/01/01	9호선	대전	3224	2903	2022/01/01
5	2022/01/01	7호선	일본	3221	2403	2022/01/01
6	2022/01/01	5호선	장동	7	0	2022/01/01
7	2022/01/01	9호선	신도동	4929	4733	2022/01/01
8	2022/01/01	1호선	동대문	6488	6290	2022/01/01
9	2022/01/01	9호선	송파구	1915	1922	2022/01/01
10	2022/01/01	1호선	기장역	18398	18295	2022/01/01
11	2022/01/01	9호선	동작역	7243	7366	2022/01/01
12	2022/01/01	9호선	사당	5684	5506	2022/01/01

(a) 전처리 전 파일(CARD_SUBWAY_MONTH_2022.CSV)

그림 13-3 지하철승하차 전처리 전/후 파일

A	B	C	D	E	F	G
1	사용일자	호선명	역명	승차승객수	하차승객수	
2	2022/01/01	1호선	인천	4939	4733	
3	2022/01/01	8호선	동대문	6406	6390	
4	2022/01/01	9호선	송포구	7975	7922	
5	2022/01/01	7호선	서용역	18198	18026	
6	2022/01/01	5호선	동묘앞	7240	7308	
7	2022/01/01	9호선	시청	5604	5006	
8	2022/01/01	1호선	종각	9977	8854	
9	2022/01/01	9호선	송로구	11017	9680	
10	2022/01/01	1호선	방왕리	9400	8553	
11	2022/01/01	9호선	제기동	7244	7259	
12	2022/01/01	9호선	송월리	10719	10816	

(b) 전처리 후 파일(지하철승하차.CSV)

2. 전처리하기

■ 지하철공기질 데이터의 전처리 수행 작업

- ① 측정연도, 측정위치, 이산화질소, 라돈, 휘발성유기화합물, 관리기관명, 데이터기준일자 필드 삭제
 - ② 지하철역사명에 붙은 (호선) 삭제
 - ③ '지하철역사명'을 '역명', '지하철호선명'을 '호선명'으로 변경
 - ✓ 전처리 전 파일명: 서울교통공사_지하철공기질측정정보_20230711.CSV
 - ✓ 전처리 후 파일명: 지하철공기질.CSV

(a) 전처리 전 파일(서울교통공사_지하철공기질측정정보_20230711.CSV)

(b) 전처리 후 파일(지하철공기질.CSV)

그림 13-4 지하철공기질 전처리 전/후 파일

3. 스키마 생성하기

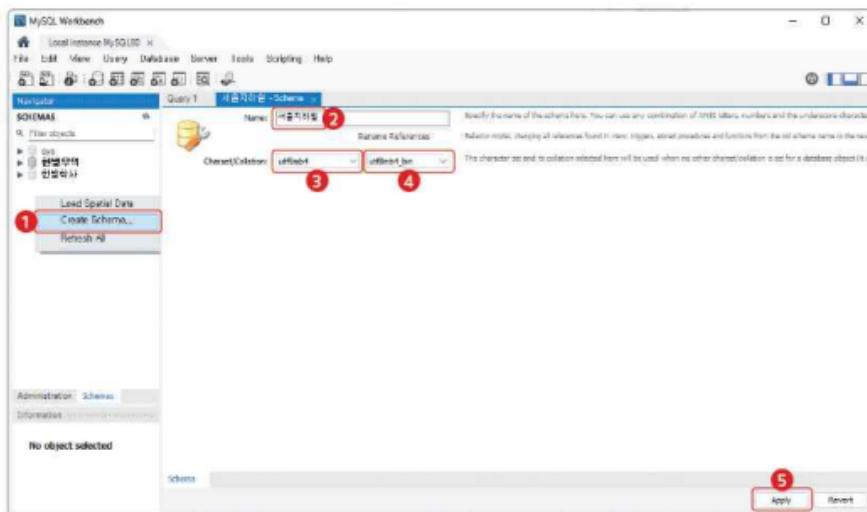
■ 스키마 생성하기

- 워크벤치의 [Navigator] 창에서 마우스 오른쪽 버튼을 클릭하여 [Create Schema...]를 클릭

- ✓ 스키마 이름: '서울지하철'
- ✓ Charset: 'utf8mb4'

Collation: 'utf8mb4_bin'

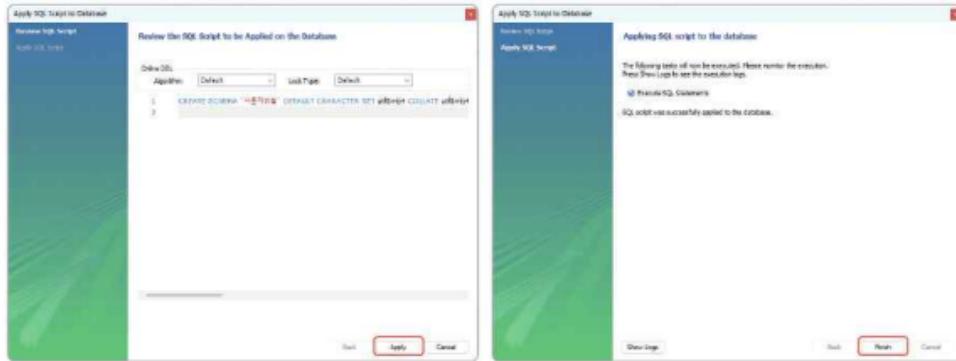
스키마를 생성하는 대신 2장을 참고하여 데이터베이스를 생성해도 동일하게 작업할 수 있습니다.



3. 스키마 생성하기

■ 스키마 생성하기

- [Apply SQL Script to Database] 창에서 <Apply>와 <Finish> 버튼 클릭하여 스키마 생성 완료



- Navigator 창에서 새로고침 버튼을 클릭해 '서울지하철' 스키마 생성 확인



4. 데이터 가져오기

■ 데이터 가져오기

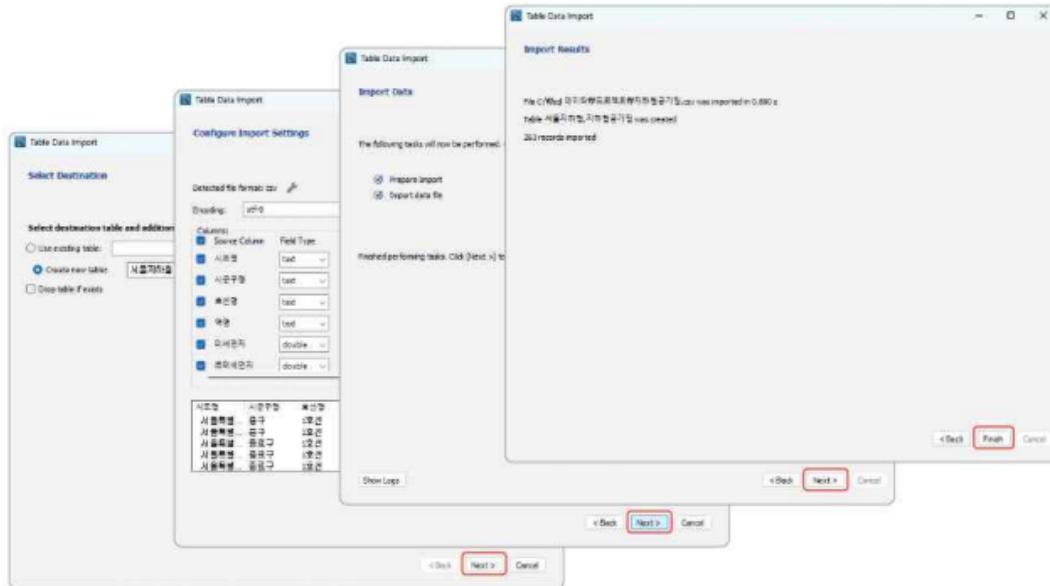
- 서울지하철 스키마의 [Tables]에서 마우스 오른쪽 버튼을 클릭하고 [Table Data Import Wizard] 선택
- Table Data Import 창에서 <Browse...> 버튼을 클릭하여 '지하철공기질.csv' 파일을 선택하고, <Next> 버튼 클릭



4. 데이터 가져오기

■ 데이터 가져오기

- 이후 창에서 [Next], [Finish] 버튼 클릭하여 데이터 가져오기와 테이블 생성 완료



4. 데이터 가져오기

■ 데이터 가져오기

- 새로 생성된 지하철공기질 테이블과 지하철승하차 테이블에 있는 컬럼 중 일부 컬럼의 데이터타입과 크기 변경

```
ALTER TABLE 지하철공기질 MODIFY 호선명 VARCHAR(30);
ALTER TABLE 지하철공기질 MODIFY 역명 VARCHAR(30);
ALTER TABLE 지하철공기질 MODIFY 시도명 VARCHAR(30);
ALTER TABLE 지하철공기질 MODIFY 시군구명 VARCHAR(30);

ALTER TABLE 지하철승하차 MODIFY 사용일자 DATE;
ALTER TABLE 지하철승하차 MODIFY 호선명 VARCHAR(30);
ALTER TABLE 지하철승하차 MODIFY 역명 VARCHAR(30);
```

- 지하철공기질 테이블은 '호선명 + 역명' 컬럼으로, 지하철승하차 테이블은 '사용일자 + 호선명 + 역명' 컬럼으로 기본키 설정

```
ALTER TABLE 지하철공기질 ADD PRIMARY KEY(호선명, 역명);
ALTER TABLE 지하철승하차 ADD PRIMARY KEY(사용일자, 호선명, 역명);
```

4. 데이터 가져오기

■ 데이터 가져오기

- Navigator 창에서 새로고침 버튼을 클릭하면 생성된 테이블과 컬럼을 확인할 수 있음



Section 03

서울 지하철 공기질 자료의
탐색적 데이터 분석

1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

■ 서울 지하철 공기질 자료의 탐색적 데이터 분석



그림 13-5 지하철 공기질에 대한 탐색적 데이터 분석 프로젝트

1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

■ [예제 13-1] 지하철공기질 테이블의 모든 정보를 보이시오.

- 작업 데이터베이스를 '서울지하철'로 지정

```
USE 서울지하철;
```

- 지하철공기질 테이블 검색

```
SELECT *  
FROM 지하철공기질;
```

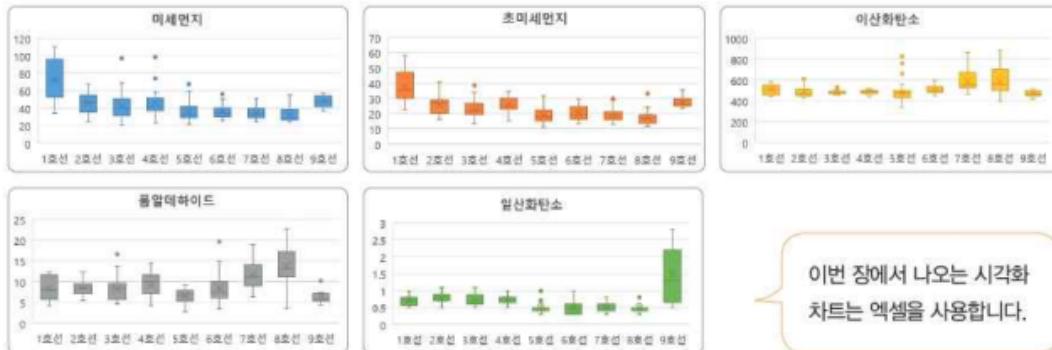
▶ 실행 결과

시도별	시군구별	호선명	역명	미세먼지	초미세먼지	이산화탄소	용알레하이드	질산화탄소
서울특별시	중구	1호선	서울역	39.9	27.3	566	11.5	0.7
서울특별시	중구	1호선	시청	83.5	34.9	545	12	0.7
서울특별시	종로구	1호선	종각	110	58	500	12.2	0.7
서울특별시	종로구	1호선	종로3가	96	44.6	507	10.3	0.8
서울특별시	종로구	1호선	종로5가	73.2	37.6	525	8.1	0.7
서울특별시	종로구	1호선	한대	44.6	54	79	*	*

1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

■ [예제 13-1] 지하철공기질 테이블의 모든 정보를 보이시오.

- 상자그림(Box Plot) 차트를 사용한 공기질 정보 시각화



이번 장에서 나오는 시각화 차트는 엑셀을 사용합니다.

그림 13-6 상자그림 차트를 사용하여 지하철 공기질 데이터를 시각화한 예

하나 더 알기 ✓

상자그림 차트

상자그림 차트(Box Plot Chart)는 최솟값, 제1사분위 수(25%), 제2사분위 수(50%), 제3사분위 수(75%), 최댓값 등 5가지 통계 묘약 수치 및 이상치를 시각적으로 표현합니다. 이 차트는 주로 데이터 분포를 이해하고 여러 그룹 간의 데이터 분포를 비교하여 통계적 차이를 시각화함으로써 데이터의 특성을 파악하는 데 활용됩니다.



1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

■ [예제 13-1] 지하철공기질 테이블의 모든 정보를 보이시오.

- 분산형 차트를 사용해 공기질 항목 간의 관련성 확인

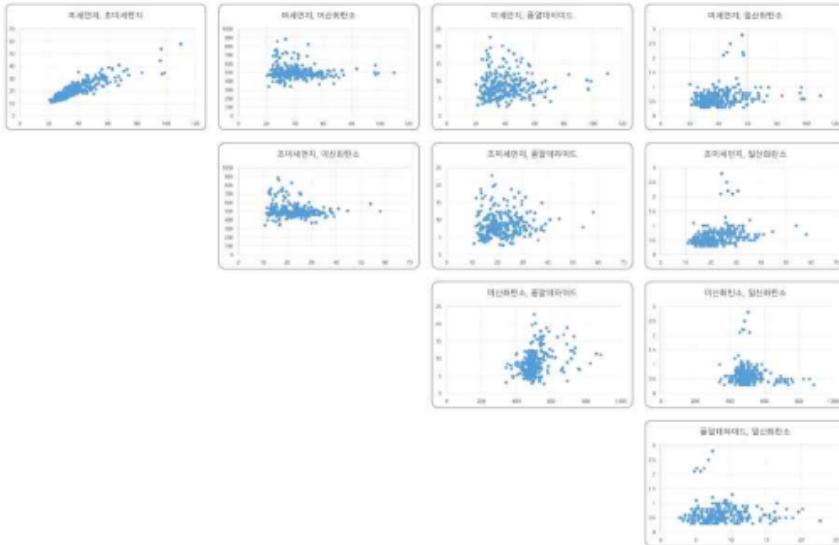


그림 13-7 분산형 차트를 사용하여 시각화한 예

하나 더 알기 ▾

분산형 차트

분산형 차트(Scatter Chart)는 데이터 간의 상관관계 및 패턴을 시각적으로 파악하거나 이상치를 식별하기 위해 주로 활용됩니다. 또한 분포와 집단 간의 비교를 통해 데이터의 특성을 빠르게 파악하는데도 도움이 됩니다.



1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

- [예제 13-2] 지하철공기질 테이블에 있는 총 레코드 건수 및 각 공기질 측정 항목에 대하여 평균 수치값을 보이시오.

```
SELECT COUNT(*) AS 총건수  
    ,ROUND(AVG(미세먼지), 1) AS 평균미세먼지  
    ,ROUND(AVG(초미세먼지), 1) AS 평균초미세먼지  
    ,ROUND(AVG(이산화탄소), 1) AS 평균이산화탄소  
    ,ROUND(AVG(폼알데하이드), 1) AS 평균폼알데하이드  
    ,ROUND(AVG(일산화탄소), 1) AS 평균일산화탄소  
FROM 지하철공기질;
```

▶ 실행 결과

총건수	평균미세먼지	평균초미세먼지	평균이산화탄소	평균폼알데하이드	평균일산화탄소
263	40.4	22.3	507.9	8.7	0.6

1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

■ [예제 13-3] 호선별로 공기질의 각 항목에 대한 평균값을 보이시오.

```
SELECT 호선명
      ,ROUND(AVG(미세먼지), 1) AS 평균미세먼지
      ,ROUND(AVG(초미세먼지), 1) AS 평균초미세먼지
      ,ROUND(AVG(이산화탄소), 1) AS 평균이산화탄소
      ,ROUND(AVG(폼알데하이드), 1) AS 평균폼알데하이드
      ,ROUND(AVG(일산화탄소), 1) AS 평균일산화탄소
  FROM 지하철공기질
 GROUP BY 호선명;
```

▶ 실행결과

호선명	평균미세먼지	평균초미세먼지	평균이산화탄소	평균풀알데하이드	평균일산화탄소
1호선	72.6	37.9	509.6	8.4	0.7
2호선	45.8	26	483.4	8.4	0.8
3호선	42	22.8	481.9	8.3	0.7
4호선	45.9	25.8	483.9	9.2	0.7
5호선	36.1	19	474.6	6.6	0.5
6호선	36.1	20.3	508.8	8.3	0.5
7호선	34.5	19.1	589.0	11.4	0.5
8호선	32.6	17.7	588.9	13.7	0.5
9호선	47.7	27.5	469.8	6.2	1.5

1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

■ [예제 13-3] 호선별로 공기질의 각 항목에 대한 평균값을 보이시오.

- 데이터셋을 세로 막대형 차트로 시각화



그림 13-8 세로 막대형 차트를 사용하여 시각화한 예

하나 더 알기

막대형 차트

막대형 차트(Bar Chart)는 데이터 시각화에서 가장 일반적으로 사용되는 그래프 유형 중 하나로 수치의 크기를 막대 모양의 길이로 나타낸 그래프입니다. 이 차트는 길이나 높이로 나타낸 막대들을 비교하여 데이터 간의 상대적 크기나 양을 파악할 수 있습니다. 막대를 세로로 할 수도 있고 가로로 할 수도 있는데, 항목 수가 적거나 항목 간 수치 비교를 통해 순위를 강조하고 싶을 때에는 세로 막대 차트가 좀 더 이해하기 좋습니다. 항목 수가 많거나 항목 레이블이 길다면 가로 막대 차트가 좀 더 직관적이고 보기 좋습니다.



1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

- [예제 13-4] 1호선 역에 대하여 미세먼지 수치가 높은 역부터 10개 역의 순위를 매겨 나타내시오.

```
SELECT RANK() OVER(ORDER BY 미세먼지 DESC) AS 순위  
    ,역명  
    ,미세먼지  
FROM 지하철공기질  
WHERE 호선명 = '1호선'  
LIMIT 10;
```

▶ 실행결과

순위	역명	미세먼지
1	줄각	110
2	동대문	96.6
3	종로3가	96
4	시청	83.5
5	풀로5가	73.2
6	제기동	71
7	신설동	64.8
8	정왕리	56.8
9	서풀역	39.9
10	동묘앞	34.3

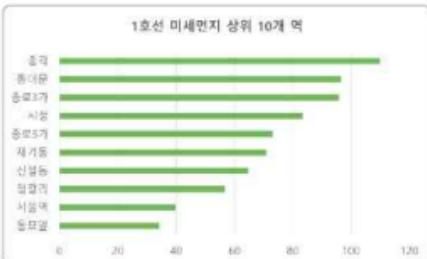


그림 13-9 가로 막대형 차트를 사용하여 시각화한 예

1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

- [예제 13-5] 1호선의 평균 미세먼지 수치보다 높은 수치를 보이는 호선 및 역에 대한 정보를 보이시오.

```
SELECT 호선명
      ,역명
      ,미세먼지
FROM 지하철공기질
WHERE 미세먼지 > (
      SELECT AVG(미세먼지)
      FROM 지하철공기질
      WHERE 호선명 = '1호선'
    );
```

▶ 실행 결과

호선명	역명	미세먼지
1호선	동대문	96.6
1호선	시청	83.5
1호선	종각	110
1호선	종로3가	96
1호선	종로5가	73.2
3호선	출지로3가	96.9
4호선	수유	98.7
4호선	이촌	74.7

1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

- [예제 13-6] 시군구명별로 역 개수와 공기질 각 항목에 대한 평균값을 보이시오. 이때 역의 개수가 많은 레코드부터 순서대로 나타내시오.

```
SELECT 시군구명
      ,COUNT(*) AS 역개수
      ,ROUND(AVG(미세먼지), 1) AS 평균미세먼지
      ,ROUND(AVG(초미세먼지), 1) AS 평균초미세먼지
      ,ROUND(AVG(이산화탄소), 1) AS 평균이산화탄소
      ,ROUND(AVG(폼알데하이드), 1) AS 평균폼알데하이드
      ,ROUND(AVG(일산화탄소), 1) AS 평균일산화탄소
FROM 지하철공기질
GROUP BY 시군구명
ORDER BY 2 DESC;
```

▶ 실행 결과

시군구명	역개수	평균미세먼지	평균초미세먼지	평균이산화탄소	평균폼알데하이드	평균일산화탄소
송파구	27	36.2	21.2	518.4	9.1	0.7
중구	23	49.4	26.5	488.9	9	0.7
강남구	21	39	22.2	523.7	9.4	1
마포구	16	38.5	20.6	494.5	6.8	0.6
종로구	15	54.7	29.2	495.7	9.1	0.6
은평구	13	38.7	22.9	492.3	8.4	0.5
강동구	13	39.1	21.5	511.0	8.4	0.5
관악구	12	--	22	--	0.7	0.7

1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

■ [예제 13-6] 시군구명별로 역 개수와 공기질 각 항목에 대한 평균값을 보이시오. 이때 역의 개수가 많은 레코드부터 순서대로 나타내시오.

- 데이터셋 시각화

시군구명	역개수	평균미세먼지	평균초미세먼지	평균이산화탄소	평균총알데하이드	평균일산화탄소
송파구	27	36.2	21.2	518.4	9.1	0.7
중구	23	49.4	26.5	488.9	9	0.7
강남구	21	39	22.2	523.7	9.4	1
마포구	16	38.5	20.6	484.5	6.8	0.6
문로구	15	54.7	29.2	495.7	9.1	0.6
은평구	13	38.7	22.9	492.3	8.4	0.5
강동구	13	39.1	21.5	511	8.4	0.5
서초구	12	37	22	524.7	9.7	0.7

그림 13-10 조건부 서식(데이터 막대)을 사용하여 시각화한 예

하나 더 알기

조건부 서식

엑셀의 조건부 서식(Conditional Formatting)은 데이터 시각화를 위해서도 유용하게 사용할 수 있습니다. 조건부 서식을 통하여 특정 조건을 만족하는 데이터를 시각적으로 강조할 수 있습니다. 특히 막대 차트와 같이 수치의 크기를 비교하거나 데이터를 범주화하고 관련된 항목을 같은 서식으로 표시하여 데이터 집계 및 분석을 쉽게 할 수 있어 데이터를 직관적으로 이해하고 효과적으로 표현할 수 있습니다.



1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

■ [예제 13-7] 시군구명별로 역의 개수와 역명 목록을 보이시오. 이때 중복된 역명은 한번씩만 보이며, 역의 개수가 많은 레코드부터 순서대로 나타내시오.

```
SELECT 시군구명  
    ,COUNT(DISTINCT 역명) AS 역개수  
    ,GROUP_CONCAT(DISTINCT 역명) AS 역명  
FROM 지하철공기질  
GROUP BY 시군구명  
ORDER BY 역개수 DESC;
```

▶ 실행결과

시군구명	역개수	역명
송파구	22	가락시장, 강들구정, 개통, 거여, 강갈병원, 마천, 물관로성, 문점, 방이, 폭정, 삼전, 신촌, 석촌고분, 솔파, 솔파나루, 오금, 올림픽공원, 원당, 원당역, 원당호수공원, 원당호수공원역, 원당호수역, 원당호수정류장, 원당호수정류장역, 원당호수정류장2호선역, 원당호수정류장3호선역, 원당호수정류장4호선역, 원당호수정류장5호선역, 원당호수정류장6호선역, 원당호수정류장7호선역, 원당호수정류장8호선역, 원당호수정류장9호선역, 원당호수정류장10호선역, 원당호수정류장11호선역, 원당호수정류장12호선역, 원당호수정류장13호선역, 원당호수정류장14호선역, 원당호수정류장15호선역, 원당호수정류장16호선역, 원당호수정류장17호선역, 원당호수정류장18호선역, 원당호수정류장19호선역, 원당호수정류장20호선역, 원당호수정류장21호선역, 원당호수정류장22호선역
강남구	21	강남, 강남구청, 논현, 대방, 대치, 도곡, 마봉, 불암사, 삼성, 삼성중앙, 선릉, 선릉역, 수서, 신사, 압구정, 언주, 역삼, 일원, 청담, 학동, 흰재, 흰재역, 흰재정류장, 흰재정류장역, 흰재정류장2호선역, 흰재정류장3호선역, 흰재정류장4호선역, 흰재정류장5호선역, 흰재정류장6호선역, 흰재정류장7호선역, 흰재정류장8호선역, 흰재정류장9호선역, 흰재정류장10호선역, 흰재정류장11호선역, 흰재정류장12호선역, 흰재정류장13호선역, 흰재정류장14호선역, 흰재정류장15호선역, 흰재정류장16호선역, 흰재정류장17호선역, 흰재정류장18호선역, 흰재정류장19호선역, 흰재정류장20호선역, 흰재정류장21호선역, 흰재정류장22호선역
마포구	14	광역, 광진장, 대종, 마포, 마포구정, 망원, 상수, 신촌, 아현, 애오개, 홀드컵경기장, 이대, 합정, 흥대입구
중구	14	동대문역사문화공원, 동대입구, 명동, 버티고개, 서울역, 시청, 신동, 약수, 을지로3가, 을지로4가, 을지로입구, 청구, 충무로, 회현
강동구	12	강동, 강일, 그제, 금은다리, 갈동, 돈촌동, 돈촌오름, 명일, 상일동, 암사, 중앙보통병원, 전포
은평구	11	구산, 구파발, 녹번, 톡바위, 디자밸미디어시티, 불량, 새암, 역촌, 연신내, 풍암, 증산
종로구	11	경복궁, 광화문, 동대문, 동묘앞, 서대문, 안국, 종각, 종로3가, 종로5가, 창신, 해화
서초구	10	고속터미널, 고대, 남부터미널, 낭태정, 내방, 반포, 방배, 서초, 양재, 잠원
영등포구	10	대월, 무리, 보라매, 신길, 신릉, 양평, 여의나루, 여의도, 영등포구청, 영등포시장

1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

- [예제 13-8] 환승역과 역개수, 환승 호선명을 목록으로 함께 보이시오. 이때 환승 역개수가 많은 역부터 순서대로 나타내시오.

```
SELECT 역명 AS 환승역  
    ,COUNT(*) AS 역개수  
    ,GROUP_CONCAT(호선명) AS 호선명  
FROM 지하철공기질  
GROUP BY 역명  
HAVING COUNT(*) >= 2  
ORDER BY 2 DESC;
```

▶ 실행결과

환승역	역개수	호선명
동대문역사문화공원	3	2호선,4호선,5호선
종로5가	3	1호선,3호선,5호선
고속터미널	2	3호선,7호선
광화	2	5호선,6호선
교대	2	2호선,3호선
구자	2	4호선,7호선

1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

■ [예제 13-9] 환승역과 환승역이 아닌 역을 구분하여 각 공기질 항목에 대한 평균값을 보이시오.

```
WITH 환승역 AS
(
SELECT 역명
,COUNT(*) AS 환승역수
FROM 지하철공기질
GROUP BY 역명
HAVING COUNT(*) >= 2
)
```

```
SELECT '환승역' AS 구분
,ROUND(AVG(미세먼지), 1) AS 평균미세먼지
,ROUND(AVG(초미세먼지), 1) AS 평균초미세먼지
,ROUND(AVG(이산화탄소), 1) AS 평균이산화탄소
,ROUND(AVG(폼알데하이드), 1) AS 평균폼알데하이드
,ROUND(AVG(일산화탄소), 1) AS 평균일산화탄소
FROM 지하철공기질
WHERE 역명 IN (

```

```
    SELECT 역명
    FROM 환승역
)
```

```
UNION
```

```
    SELECT '단일역'
    ,ROUND(AVG(미세먼지), 1) AS 평균미세먼지
    ,ROUND(AVG(초미세먼지), 1) AS 평균초미세먼지
    ,ROUND(AVG(이산화탄소), 1) AS 평균이산화탄소
    ,ROUND(AVG(폼알데하이드), 1) AS 평균폼알데하이드
    ,ROUND(AVG(일산화탄소), 1) AS 평균일산화탄소
    FROM 지하철공기질
    WHERE 역명 NOT IN (
        SELECT 역명
        FROM 환승역
    );
)
```

▶ 실행 결과

구분	평균미세먼지	평균초미세먼지	평균이산화탄소	평균풀알데하이드	평균일산화탄소
환승역	42.2	23.4	508.1	8.6	0.6
단일역	39.8	21.9	507.8	8.8	0.6

1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

■ [예제 13-9] 환승역과 환승역이 아닌 역을 구분하여 각 공기질 항목에 대한 평균값을 보이시오.

- 데이터셋 시각화

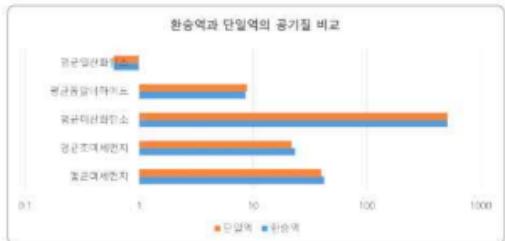


그림 13-11 가로 막대형 로그 차트를 사용하여 시각화한 예

하나 더 알기

로그 막대 차트

로그 막대 차트는 아주 큰 값과 작은 값이 함께 존재할 때 유용하게 사용할 수 있습니다. 이 차트는 작은 값과 큰 값 간의 비율을 강조합니다. 이를 통해 작은 변화나 간격을 뚜렷하게 나타낼 수 있으며, 데이터 간의 상대적 비교가 용이해집니다. 엑셀에서는 '로그 눈금 간격' 옵션을 지정하여 적용할 수 있습니다.



1. 서울 지하철 공기질 자료의 탐색적 데이터 분석

- [예제 13-10] 각 호선별로 미세먼지 수치가 가장 높은 역에 대하여 역명과 미세먼지 수치를 보이시오.

```
SELECT 호선명
      ,역명
      ,미세먼지
FROM 지하철공기질
WHERE (호선명, 미세먼지) IN (SELECT 호선명, MAX(미세먼지)
                                FROM 지하철공기질
                                GROUP BY 호선명);
```

▶ 실행결과

호선명	역명	미세먼지
1호선	종각	110
2호선	신림	67.6
3호선	을지로3가	96.9
4호선	수유	98.7
5호선	방화	67.4
6호선	역촌	56
7호선	상도	50.9
8호선	동촌로7성	55.2
9호선	삼성중앙	57.1

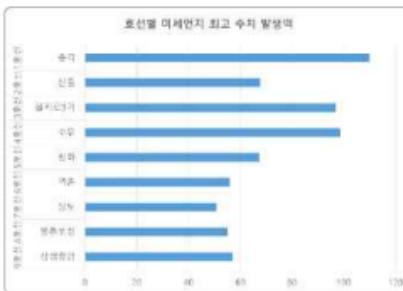


그림 13-12 가로 막대형 차트를 사용하여 시각화한 예

Section 04

서울 지하철 승하차 자료의
탐색적 데이터 분석

1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

■ [예제 13-11] 호선명별로 승차승객수합, 하차승객수합, 승하차승객수합을 보아시오.

```
SELECT 호선명  
      ,SUM(승차승객수) AS 승차승객수합  
      ,SUM(하차승객수) AS 하차승객수합  
      ,SUM(승차승객수 + 하차승객수) AS 승하차승객수합  
FROM 지하철승하차  
GROUP BY 호선명;
```

▶ 실행결과

호선명	승차승객수합	하차승객수합	승하차승객수합
1호선	74538614	72891546	147430160
2호선	441861854	447823921	889685775
3호선	167796748	167588713	335375461
4호선	158029620	160093030	318122650
5호선	206065776	204965294	411031070
6호선	108027486	106623231	214650717
7호선	191420784	188305212	379725996
8호선	60647370	61098112	121745482
9호선	89484922	90306325	179791247
9호선2~3단계	32062985	31802541	63865526

1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

■ [예제 13-11] 호선명별로 승차승객수합, 하차승객수합, 승하차승객수합을 보아시오.

- 데이터셋 시각화



그림 13-13 누적 세로 막대형 차트를 사용하여 시각화한 예

하나 더 알기

누적 막대형 차트

누적 막대형 차트(Stacked Column Chart)는 데이터의 부분과 전체 간의 상대적 비율을 시각적으로 나타내는 데 사용됩니다. 항목 간 상대적 크기나 비율을 비교하거나 전체에서 각 부분이 차지하는 비율을 보여주어 데이터의 전반적인 구조를 이해하는 데 도움이 됩니다.



1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

■ [예제 13-12] 호선명별로 역개수, 역명, 승하차승객수합, 역당 평균승하차승객수를 보이시오. 이때 역당 평균승하차승객수가 많은 레코드부터 순서대로 나타내시오.

```
SELECT 호선명
    ,COUNT(DISTINCT 역명) AS 역개수
    ,GROUP_CONCAT(DISTINCT 역명) AS 역명
    ,SUM(승차승객수 + 하차승객수) AS 승하차승객수합
    ,ROUND(SUM(승차승객수 + 하차승객수) / COUNT(DISTINCT 역명), 0) AS 역당_평균승하차승객수
FROM 지하철승하차
GROUP BY 호선명
ORDER BY 5 DESC;
```

▶ 실행 결과

호선명	역개수	역명	승하차승객수합	역당_평균승하차승객수
2호선	50	길동, 강현, 전대입구, 고대, 구포디지털단지, 구포, 낙성대, 은산, 대월, 도원천, 흥덕동역사문화공원, 화암, 문래, 방화, 불현, 사당, 일...	8886885775	1791735
1호선	30	동대문, 동교앞, 서울역, 시청, 선릉역, 계기동, 광진역, 광교역, 광교5가, 한성역	147490160	14749016
4호선	26	길동, 남대현, 노원, 도고역, 흥대동, 흥아증주사랑병원역, 홍제, 영통, 디아, 미아사거리역, 시설, 실리지, 강제, 서울역, 신진여대입구, ...	318123630	12235487
3호선	34	기동역, 삼각, 경복궁, 경찰병원, 그린리버빌, 고려, 구포역, 광교, 남부극장역, 노원, 대원, 디원, 도곡, 노원역, 등디입구, 미용, 무의역, ...	333375461	9862984
7호선	51	가산디지털단지, 강남구청, 간접입구, 고 솔리미 네, 풍물, 양재사거리, 군자, 쌤포역, 기지역, 날구포, 날사, 날역, 노원, 논현, 대월, 도...	337235966	7446608
5호선	36	길동, 강원, 개봉, 경화산, 거여, 고덕, 광교, 경나루, 경마로, 군자, 굽은다리, 길동, 괴포공원, 기사단, 길십리, 몽내역역사문화공원, 등...	411031070	7309641
9호선	25	기암, 개화, 고속터미널, 광한역, 광한로역, 광한역, ...	179791247	7193650
8호선	38	기동역, 강동구청, 남위례, 남한신성역구, 민묘역거리, 광진, 광진역, 광진역, 광진역, 광진역, 광진역, 광진역, 광진역, 광진역, 광진역, ...	121745462	6703638
6호선	39	고려대, 광교, 광진역, 구신, 축사역, 대월, 뚜렷위, 둘꽃이, 둘꽃이, 디자인미디어비전, 마포구청, 말동, 배드민턴, 보문, 광진역, 광진역, 광진역, ...	214650717	5503665
9호선~2호선	12	동촌오류, 봉은사, 삼성중앙, 삼천, 석촌, 석촌고분, 선정로, 송파나루, 연주, 을밀로, 을밀로, 을밀로, 을밀로, 을밀로, 을밀로, 을밀로, 을밀로, ...	63365526	4912733

1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

■ [예제 13-12] 호선명별로 역개수, 역명, 승하차승객수합, 역당 평균승하차승객수를 보이시오. 이때 역당 평균승하차승객수가 많은 레코드부터 순서대로 나타내시오.

- 데이터셋 시각화

호선명	역개수	승하차승객수합	역당_평균승하차승객수
2호선	50	889,585,775	17,792,716
1호선	10	147,430,160	14,743,016
4호선	25	318,122,650	12,725,487
3호선	34	335,375,461	9,863,984
7호선	51	379,725,996	7,445,600
5호선	56	411,011,070	7,389,841
9호선	25	179,791,247	7,191,650
8호선	19	121,745,482	6,763,639
6호선	39	214,650,717	5,559,865
9호선 2~3단계	13	63,865,526	4,912,733

그림 13-14 조건부 서식(색조)을 사용하여 시각화한 예



그림 13-15 원형 차트를 사용하여 시각화한 예

하나 더 알기 V

원형 차트

원형 차트(Pie Chart)는 데이터의 상대적인 부분과 전체 간의 비율을 시각적으로 나타내는데 사용됩니다. 원형 차트를 사용하면 부분-전체 관계에서 상대적 비율을 강조하여 시각화할 수 있습니다.



1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

■ [예제 13-13] 월별, 호선명별로 승하차승객수합을 보이시오

```
SELECT MONTH(사용일자) AS 월
      ,호선명
      ,SUM(승차승객수 + 하차승객수) AS 승하차승객수합
  FROM 지하철승하차
 GROUP BY MONTH(사용일자)
      ,호선명
 ORDER BY 1, 2;
```

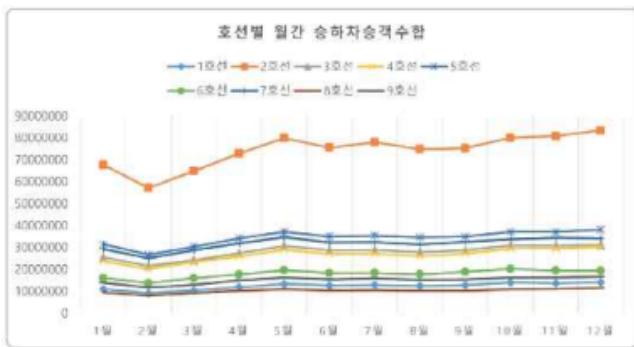
▶ 실행 결과

월	호선명	승하차승객수합
1	1호선	10910954
1	2호선	67662511
1	3호선	25383519
1	4호선	23726386
1	5호선	31344826
1	6호선	15846303

1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

■ [예제 13-13] 월별, 호선명별로 승하차승객수합을 보이시오

- 데이터셋 시각화



하나 더 알기

꺾은선 차트

꺾은선 차트(Line Chart)는 데이터의 추세, 패턴, 상관관계를 시각적으로 보여주는 데 사용됩니다. 꺾은선 차트는 시계열 데이터를 표시하여 시간에 따른 값의 변화를 분석함으로써 시간 내에서의 추세, 주기성, 계절성 등을 파악할 수 있으며, 서로 다른 변수 간의 상관관계를 시각화하여 선형 또는 비선형 관계를 파악할 수도 있습니다. 또한 과거 데이터를 기반으로 미래 추세를 예측하고 데이터의 흐름을 이해하는 데에도 유용하게 사용할 수 있습니다.



1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

- [예제 13-14] 어느 역, 어느 호선의 승하차승객수합이 많은지 상위 10개의 정 보를 보이시오.

```
SELECT 역명
      ,호선명
      ,SUM(승차승객수 + 하차승객수) AS 승하차승객수합
  FROM 지하철승하차
 GROUP BY 역명
      ,호선명
 ORDER BY 3 DESC
 LIMIT 10;
```

▶ 실행결과

역명	호선명	승하차승객수합
길동	2호선	51764650
잠실	2호선	49660855
홍대입구	2호선	43253951
신월	2호선	39184576
구로디지털단지	2호선	36820457
삼성	2호선	33684518
...	2호선547

1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

■ [예제 13-15] 2호선에 대하여 요일별로 승하차승객수합을 보이되 월요일부터 순서대로 보이시오

```
SELECT MONTH(사용일자) AS 월
      ,WEEKDAY(사용일자) AS 정수요일
      ,CASE WEEKDAY(사용일자)
        WHEN 0 THEN '월요일'
        WHEN 1 THEN '화요일'
        WHEN 2 THEN '수요일'
        WHEN 3 THEN '목요일'
        WHEN 4 THEN '금요일'
        WHEN 5 THEN '토요일'
        ELSE '일요일'
      END AS 요일
      ,SUM(승차승객수 + 하차승객수) AS 승하차승객수합
FROM 지하철승하차
WHERE 호선명 = '2호선'
GROUP BY MONTH(사용일자)
      ,WEEKDAY(사용일자)
      ,CASE WEEKDAY(사용일자)
        WHEN 0 THEN '월요일'
        WHEN 1 THEN '화요일'
        WHEN 2 THEN '수요일'
        WHEN 3 THEN '목요일'
        WHEN 4 THEN '금요일'
        WHEN 5 THEN '토요일'
        ELSE '일요일'
      END
ORDER BY 1,2;
```

▶ 실행 결과

월	정수요일	요일	승하차승객수합
1	0	월요일	11104750
1	1	화요일	10611443
1	2	수요일	10712608
1	3	목요일	10685467
1	4	금요일	10802850
1	5	토요일	7836806
1	6	일요일	5559587

1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

- [예제 13-15] 2호선에 대하여 요일별로 승하차승객수합을 보이되 월요일부터 순서대로 보이시오
 - 데이터셋 시각화



그림 13-17 누적 세로 막대형 차트에서 행/열 전환을 사용하여 시각화한 예

1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

- [예제 13-16] 강남역과 홍대입구역, 잠실역, 명동역의 12월 데이터에 대하여 역명, 사용일자, 승하차승객수 및 승하차승객수누적합을 보이시오.

```
SELECT 역명
      ,사용일자
      ,SUM(승차승객수 + 하차승객수) AS 승하차승객수합
      ,SUM(승차승객수 + 하차승객수)
          OVER(PARTITION BY 역명 ORDER BY 사용일자) 승하차승객수누적합
  FROM 지하철승하차
 WHERE 역명 IN ('강남','홍대입구','잠실','명동')
   AND MONTH(사용일자) = 12
 GROUP BY 역명
      ,사용일자
 ORDER BY 1, 2;
```

▶ 실행결과

역명	사용일자	승하차승객수합	승하차승객수누적합
길동	2022-12-01	160084	160084
길동	2022-12-02	173348	333432
길동	2022-12-03	121627	455059
길동	2022-12-04	521978	
길동		166894	
강남	2022-12-31	94064	4615577
명동	2022-12-01	53912	53912
명동	2022-12-02	57983	111895
명동	2022-12-03	61408	173303
명동	2022-12-04	44485	217788

160,084 + 173,348

53,912 + 57,983

1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

■ [예제 13-16] 강남역과 홍대입구역, 잠실역, 명동역의 12월 데이터에 대하여 역명, 사용일자, 승하차승객수 및 승하차승객수누적합을 보이시오.

- 데이터셋 시각화



그림 13-18 꺾은선 차트에서 주이 비교를 시각화한 예



에러 코드 1055가 발생하는 경우

[예제 13-16]을 실행했을 때 오류가 발생할 수 있습니다. MySQL 5.7.5 이상에서는 ONLY_FULL_GROUP_BY 모드가 활성화된 경우 선택 목록, HAVING 조건 또는 ORDER BY 목록이 GROUP BY 절에 지정되지 않았거나 기능적으로 종속되지 않은 집계되지 않은 열을 참조하는 쿼리는 허용하지 않는 것으로 변경되었기 때문입니다. 이 문제는 sql_mode에 설정되어 있는 ONLY_FULL_GROUP_BY 모드를 다음과 같이 비활성으로 변경하여 해결할 수 있습니다.

```
SET sql_mode=(SELECT REPLACE(@@sql_mode,'ONLY_FULL_GROUP_BY',''));
```



1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

- [예제 13-17] 환승역에 대하여 승하차승객수합을 보이시오. 이때 승하차승객 수합이 많은 역부터 나타내시오.

```
WITH 환승역 AS
(
SELECT 역명
,COUNT(*) AS 역개수
FROM (
      SELECT DISTINCT 호선명
      ,역명
      FROM 지하철승하차
      ) AS t
GROUP BY 역명
HAVING COUNT(*) >= 2
)

SELECT 역명
, SUM(승차승객수 + 하차승객수) AS 승하차승객수합
, GROUP_CONCAT(DISTINCT 호선명) AS 호선명
FROM 지하철승하차
WHERE 역명 IN (
      SELECT 역명
      FROM 환승역
      )
GROUP BY 역명
ORDER BY 2 DESC;
```

▶ 실행결과

역명	승하차승객수합	호선명
갈월	60624028	2호선,8호선
고속터미널	52696727	3호선,7호선,9호선
사당	42905316	2호선,4호선
서울역	40343499	1호선,4호선
여의도	35053176	5호선,9호선
...	34390287	...

1. 서울 지하철 승하차 자료의 탐색적 데이터 분석

- [예제 13-18] 호선명, 역명, 승차승객수합, 하차승객수합, 승하차승객수합을 보이는 뷰를 지하철승하차뷰라는 이름으로 생성하시오.

```
CREATE VIEW 지하철승하차뷰
AS
SELECT 호선명
      ,역명
      ,SUM(승차승객수) AS 승차승객수합
      ,SUM(하차승객수) AS 하차승객수합
      ,SUM(승차승객수+하차승객수) AS 승하차승객수합
FROM 지하철승하차
GROUP BY 호선명
      ,역명;
```

Section 05

서울 지하철 자료의 데이터
통합 분석

1. 서울 지하철 자료의 데이터 통합 분석

- [예제 13-19] 호선명, 역명, 승차승객수합, 하차승객수합, 승하차승객수합 및 공기질 항목의 수치를 보이시오. 이때 [예제 13-18]에서 생성한 뷰를 사용하시오.

- ANSI SQL 조인

```
SELECT 승하차.*  
    ,공기질.미세먼지  
    ,공기질.초미세먼지  
    ,공기질.이산화탄소  
    ,공기질.폼알데하이드  
    ,공기질.일산화탄소  
FROM 지하철승하차뷰 AS 승하차  
JOIN 지하철공기질 AS 공기질  
ON 승하차.호선명 = 공기질.호선명  
AND 승하차.역명 = 공기질.역명  
ORDER BY 호선명  
    ,역명;
```

1. 서울 지하철 자료의 데이터 통합 분석

■ [예제 13-19] 호선명, 역명, 승차승객수합, 하차승객수합, 승하차승객수합 및 공기질 항목의 수치를 보이시오. 이때 [예제 13-18]에서 생성한 뷰를 사용하시오.

- Non-ANSI SQL 조인

```
SELECT 승하차.*  
    ,공기질.미세먼지  
    ,공기질.초미세먼지  
    ,공기질.이산화탄소  
    ,공기질.폼알데하이드  
    ,공기질.일산화탄소  
FROM 지하철승하차뷰 AS 승하차  
    ,지하철공기질 AS 공기질  
WHERE 승하차.호선명 = 공기질.호선명  
AND 승하차.역명 = 공기질.역명  
ORDER BY 호선명  
    ,역명;
```

▶ 실행결과

호선종	역명	승차승객수합	하차승객수합	승하차승객수합	미세먼지	초미세먼지	이산화탄소	폼알데하이드	일산화탄소
1호선	동대문	3863307	3680871	7544078	96.6	54	585	7.9	1
1호선	동묘앞	3196812	3317739	6514551	34.3	22.4	457	4.7	0.5
1호선	서울역	15575611	15107638	30683249	39.9	27.3	566	11.5	0.7
1호선	서청	7472550	7549153	15021703	83.5	34.9	545	12	0.7
1호선	신설동	4469939	4326111	8796050	64.8	31	517	6.9	0.6
1호선	제기동	5200885	5188885	10389730	76.7	36.7	42	4.2	0.6

1. 서울 지하철 자료의 데이터 통합 분석

■ [예제 13-20] 공기질을 측정하지 않은 역에 대하여 호선명, 역명을 보이시오

- 먼저 몇 개의 역이 공기질 측정을 하지 않았는지 확인하기

```
SELECT COUNT(*) AS 공기질미측정역개수
FROM 지하철승하차뷰 AS 승하차
WHERE NOT EXISTS (
    SELECT *
    FROM 지하철공기질 AS 공기질
    WHERE 공기질.호선명 = 승하차.호선명
    AND 공기질.역명 = 승하차.역명
);
```

▶ 실행결과

공기질미측정역개수
72

- 공기질 측정을 하지 않은 호선명과 역명 확인하기

```
SELECT 호선명
      ,역명
FROM 지하철승하차뷰 AS 승하차
WHERE NOT EXISTS (
    SELECT *
    FROM 지하철공기질 AS 공기질
    WHERE 공기질.호선명 = 승하차.호선명
    AND 공기질.역명 = 승하차.역명
);
```

▶ 실행결과

호선명	역명
2호선	강변
2호선	건대입구
2호선	구로디지털단지
2호선	구의
2호선	당산
2호선	대월
2호선	풀남

1. 서울 지하철 자료의 데이터 통합 분석

- [예제 13-21] 공기질 측정을 한 역을 대상으로 호선명별로 역개수, 승하차승객수합, 각 호선의 역당 평균승하차승객수 및 공기질 각 항목에 대한 평균 수치를 보이시오. 이때 역당 평균승하차승객수가 많은 레코드부터 순서대로 나타내시오.

- ANSI SQL 조인

```
SELECT 승하차.호선명
      ,COUNT(승하차.역명) AS 역개수
      ,SUM(승하차승객수합) AS 승하차승객수합
      ,ROUND(승하차승객수합 / COUNT(승하차.역명), 0) AS 역당_평균_승하차인원수
      ,ROUND(AVG(미세먼지), 1) AS 평균미세먼지
      ,ROUND(AVG(초미세먼지), 1) AS 평균초미세먼지
      ,ROUND(AVG(이산화탄소), 1) AS 평균이산화탄소
      ,ROUND(AVG(품알데하이드), 1) AS 평균품알데하이드
      ,ROUND(AVG(일산화탄소), 1) AS 평균일산화탄소
  FROM 지하철승하차뷰 AS 승하차
  JOIN 지하철공기질 AS 공기질
  ON 승하차.호선명 = 공기질.호선명
  AND 승하차.역명 = 공기질.역명
 GROUP BY 승하차.호선명
 ORDER BY 4 DESC;
```

1. 서울 지하철 자료의 데이터 통합 분석

- [예제 13-21] 공기질 측정을 한 역을 대상으로 호선명별로 역개수, 승하차승객수합, 각 호선의 역당 평균승하차승객수 및 공기질 각 항목에 대한 평균 수치를 보이시오. 이때 역당 평균승하차승객수가 많은 레코드부터 순서대로 나타내시오.

- Non-ANSI SQL 조인

```
SELECT 승하차.호선명
      ,COUNT(승하차.역명) AS 역개수
      ,SUM(승하차승객수합) AS 승하차승객수합
      ,ROUND(승하차승객수합 / COUNT(승하차.역명), 0) AS 역당_평균_승하차인원수
      ,ROUND(AVG(미세먼지), 1) AS 평균미세먼지
      ,ROUND(AVG(초미세먼지), 1) AS 평균초미세먼지
      ,ROUND(AVG(이산화탄소), 1) AS 평균이산화탄소
      ,ROUND(AVG(폼알데하이드), 1) AS 평균폼알데하이드
      ,ROUND(AVG(일산화탄소), 1) AS 평균일산화탄소

  FROM 지하철승하차뷰 AS 승하차
      ,지하철공기질 AS 공기질
 WHERE 승하차.호선명 = 공기질.호선명
   AND 승하차.역명 = 공기질.역명
 GROUP BY 승하차.호선명
 ORDER BY 4 DESC;
```

1. 서울 지하철 자료의 데이터 통합 분석

- [예제 13-21] 공기질 측정을 한 역을 대상으로 호선명별로 역개수, 승하차승객수합, 각 호선의 역당 평균승하차승객수 및 공기질 각 항목에 대한 평균 수치를 보이시오. 이때 역당 평균승하차승객수가 많은 레코드부터 순서대로 나타내시오.

▶ 실행결과

호선명	역개수	승하차승객수합	역당_평균_승하차인원수	평균미세먼지	평균초미세먼지	평균미산화탄소	평균증발데하이드	평균일산화탄소
2호선	37	687,052,418	18,990,45	45.8	26	483.4	8.4	0.8
1호선	39	147,430,160	3,754,05	72.6	37.9	509.6	8.4	0.7
7호선	39	365,656,235	9,301,99	34.5	19.1	589.0	11.4	0.5
4호선	21	268,007,128	12,800,00	45.9	25.8	483.9	9.2	0.7
8호선	17	118,271,146	6,904,05	32.6	17.7	508.9	13.7	0.5
5호선	36	411,031,070	11,417,55	36.1	19	474.6	6.6	0.5
3호선	32	326,664,850	10,195,33	42	22.8	461.9	8.3	0.7
6호선	38	214,650,587	5,643,62	36.1	20.4	508.8	8.3	0.5

호선명	역개수	승하차승객수합	역당_평균_승하차인원수	평균미세먼지	평균초미세먼지	평균미산화탄소	평균증발데하이드	평균일산화탄소
2호선	37	687,052,418	18,990,453	45.8	26	483.4	8.4	0.8
1호선	39	147,430,160	3,754,065	72.6	37.9	509.6	8.4	0.7
4호선	21	268,007,128	12,800,078	45.9	25.8	483.9	9.2	0.7
3호선	36	326,664,850	9,073,57	36.1	22.8	481.9	8.3	0.7
8호선	17	118,271,146	6,901,275	32.6	17.7	588.9	13.7	0.5
7호선	39	365,656,235	9,301,902	34.5	19.1	589	11.4	0.5
6호선	38	214,650,587	5,643,673	36.1	20.4	508.8	8.3	0.5
5호선	56	411,031,070	7,379,068	36.1	19	474.6	6.6	0.5

그림 13-19 조건부 서식(색조, 아이콘 집합)을 사용하여 시각화한 예

1. 서울 지하철 자료의 데이터 통합 분석

- [예제 13-21] 공기질 측정을 한 역을 대상으로 호선명별로 역개수, 승하차승객수합, 각 호선의 역당 평균승하차승객수 및 공기질 각 항목에 대한 평균 수치를 보이시오. 이때 역당 평균승하차승객수가 많은 레코드부터 순서대로 나타내시오.



그림 13-20 콤보 차트(막대형과 깍은선)를 사용하여 시각화한 예