<혼자 공부하는 SQL>

Chapter 01. 데이터베이스와 SQL

01-1 데이터베이스 알아보기

데이터베이스(Database, DB) : '데이터의 집합'

데이터베이스와 DBMS

DBMS(Database Management System) : 데이터베이스를 관리하고 운영하는 소프트웨어.

- 다양한 데이터가 저장되어 있는 데이터베이스는 여러 명의 사용자나 응용 프로그램과 공유하고 동시에 접근이 가능해야 한다.
- 엑셀은 데이터의 집합을 관리하고 운영한다는 차원에서는 DBMS로 볼 수 있지만, 대용량 데이터를 관리하거나 여러 사용자와 공유하는 개념과는 거리가 있어 DBMS라고 부르지 않는다.
- DBMS와 같은 소프트웨어는 특정 목적을 처리하기 위한 프로그램.
  EX) 문서 작성 : HWP, Word / 표 계산 : Excel, Calc / 사진 편집 : PhotoShop, Gimp
- 데이터베이스를 사용하기 위해서도 소프트웨어, 즉 DBMS를 설치해야한다.
  EX) MySQL, Oracle, SQL Server, MariaDB 등

DBMS의 발전 과정

- 종이에 펜으로 기록 -> 컴퓨터에 파일로 저장 -> DBMS의 대두와 보급
- SQL(Structured Query Language)DBMS에 데이터를 구축 관리하고 활용하기 위해서 사용되는 언어.

이 SQL을 사용하면 DBMS를 통해 중요한 정보들을 입력, 관리하고 추출할 수 있다. 즉, SQL문을 잘 이해하고 사용해야만 DBMS를 원할하게 활용할 수 있다.

DBMS의 분류

- 계층형, 망형, 관계형, 객체지향형, 객체관계형 등으로 분류된다.
- 현재 사용되는 DBMS 중에는 관계형 DBMS가 가장 많은 부분을 차지하며, MySQL도 여기에 속한다.
- 계층형 DBMS(Hierarchical DBMS) : 각 계층은 트리 형태를 갖고, 처음 구성을 완료한 후에 이를 변경하기가 상당히 까다롭다. 그래서 현재는 사용하지 않는 형태이다.
- 망형 DBMS(Network DBMS) : 하위에 있는 구성원끼리도 연결된 유연한 구조고, 프로그래머가 모든 구조를 이해해야만 프로그램이 작성 가능하다는 단점으로, 지금은 거의 사용하지 않는 형태이다.
- 관계형 DBMS(Relational DBMS, RDBMS) : 테이블이라는 최소 단위로 구성되며, 이 테이블은 하나 이상의 열과 행으로 이뤄져 있다. 모든 데이터가 테이블에 저장된다.

DBMS에서 사용되는 언어 : SQL

- SQL(Structured Query Language) : 관계형 데이터베이스에서 사용되는 언어지만, 일반적인 프로그래밍 언어와는 조금 다른 특성을 갖고 있다.
- 표준 SQL : 국제표준화기구에서 SQL에 대한 표준을 정해서 발표하고 있다.
- SQL을 사용하는 DBMS를 만드는 회사가 여러 곳이기 때문에 표준 SQL이 각 회사 제품의 특성을 모두 포용하지 못해서 DBMS를 만드는 회사에서는 되도록 표준 SQL을 준수하되 각 제품의 특성을 반영한 SQL을 사용한다.

01-2 MySQL 설치하기

MySQL : 오라클 사에서 제공하는 데이터베이스 관리 소프트웨어로, 대용량의 데이터를 관리하고 운영하는 기능을 제공한다. MySQL은 교육용이나 개인에게는 무료로 제공되고, 영리를 목적으로 사용한다면 정해진 비용을 지불해야 한다. 만약 상용 목적인데 무료로 사용하고 싶다면 오픈 소스로 제고오디는 MariaDB를 사용할 것을 권장한다.

MySQL 설치를 위한 컴퓨터 환경

- 바탕화면 [시작] 버튼 클릭 - [시스템] 선택
- 정보 창에서 [에디션]과 [시스템 종류]를 확인

MySQL 다운로드 및 설치하기

- https://dev.mysql.com/downloads/windows/installer에 접속해 최신 버전으로 [Downloads] 버튼 클릭
- MySQL Community Downloads 화면이 나타나면 좌측 하단의 [No thanks, just start my download.]를 클릭해서 다운로드(로그인 하지 않아도 파일 다운로드 가능)
- 다운로드가 완료되면 실행, 폴더 열기, 다운로드 보기가 가능. [폴더 열기] 버튼을 클릭해서 다운로드된 파일을 확인
- 다운로드한 파일을 더블 클릭해서 설치를 시작한다. 로고가 잠깐 나타났다가 사라진다. 사용자 계정 컨트롤 창이 나타나면 [예] 버튼을 클릭
- MySQL Installer 창이 나타난다. [Choosing a Setup Type]에서는 설치 유형을 설치할 수 있는데 필요한 것들만 골라서 설치하기 위해 'Custom'을 선택한 후 [Next] 버튼을 클릭
- [Select Products and Features]에서는 설치할 제품들을 선택할 수 있다. [Available Products:]에서 [MySQL Servers] - [MySQL Server] - [MySQL Server 8.0] - [MySQL Server 8.0.21 - X64]를 선택하고 -> 버튼을 클릭
- 같은 방식으로 [Applications]의 [MySQL Workbench 8.0.21 - X64], [Documentation]의 [Sample and Examples 8.0.21 - X86]을 추가
- [Installation]에서 3개의 항목을 확인하고 [Execute] 버튼을 클릭해서 설치를 진행한다. 각 항목의 [Progress]에 설치 진행 과정이 숫자 %로 보인다.
- 설치가 완료되면 각 항목 앞에 초록색 체크가 표시되고 [Status]가 'Complete'로 변경된다. 추가 환경 설정을 위해 [Next] 버튼을 클릭
- [Product Configuration]에 2개 항목의 환경설정이 필요하다고 나온다. [Next] 버튼을 클릭한다.
- [High Availability]에서는 기본값인 'Standalone MySQL Server / Classic MySQL Replication'이 선택된 상태에서 [Next] 버튼을 클릭
- [Type and Networking]에서 [Config Type]을 'Development Computer'로 선택하고 [TCP/IP]가 체크된 상태에서 [Port]가 '3306'인 것을 확인. 이 번호는 자주 사용되므로 꼭 기억한다. [Open Windows Firewall ports for network access]도 체크되어 있어야 한다. [Next] 버튼을 클릭
- [Authentication Method]에서는 책 후반에 학습하는 파이썬과의 연동을 원활하게 하기 위해 'Use Legacy Authentication Method'를 선택하고 [Next] 버튼을 클릭
- [Accounts and Roles]에서는 MySQL 관리자(Root)의 비밀번호를 설정해야 한다. [MySQL User Accounts]에서 Root 외의 사용자를 추가할 수 있다. 지금은 그냥 비워두고 [Next] 버튼을 클릭
- [Windows Service]에서는 MySQL 서버를 윈도우즈의 서비스로 등록하기 위한 설정을 진행한다. [Windows Service Name]은 전통적으로 많이 사용하는 'MySQL'로 변경한다. 나머지는 그대로 두고 [Next] 버튼을 클릭
- [Apply Configuration]에서 설정된 내용을 적용하기 위해 [Execute] 버튼을 클릭. 각 항목에 모두 초록색 체크가 표시되면 [MySQL Server]에 대한 설정이 완료된 것이다. [Finish] 버튼을 클릭해서 설정을 종료
- [Product Configuration]이 나타난다. MySQL 8.0.21은 설정이 완료되었으며, 두 번째 Sample and Examples 8.0.21의 설정을 할 차례입니다. [Next] 버튼을 클릭
- [Connect To Server]에 연결할 서버가 보이고 [User name(사용자 이름)]에 'root'가 입력되어 있다. [Password(비밀번호)]를 앞에서 설정한 것으로 입력하고 [Check] 버튼을 클릭하면 [Status]가 'Connection succeeded'로 변경된다. 연결이 설정되었으니 [Next] 버튼을 클릭
- [Apply Configuration]에서 [Execute] 버튼을 클릭하면 설정된 내용이 적용된다. 모든 항목 앞에 초록색 체크가 표시되면 성공이다. Samples and Examples에 대한 설정이 완료되었다. [Finish] 버튼을 클릭해서 설정을 종료
- 다시 [Product Configuration]이 나온다. [Status]를 보면 모두 완료된 것이 확인. [Next] 버튼을 클릭
- [Installation Complete]에서 [start MySQL Workbench after Setup]을 체크 해제하고 [Finish] 버튼을 클릭. MySQL의 설치를 완료

MySQL 정상 작동 확인하기

- [MySQL] - [MySQL Workbench 8.0 CE]를 클릭해서 프로그램을 실행. MySQL Workbench(워크벤치) 창의 좌측 하단에서 [MySQL Connections]의 'Local instance MySQL'을 클릭
- Connect to MySQL Server 창이 나타난다. [User]는 'root'로 고정되어 있고 [Password]가 비어있다. MySQL을

설치할 때 지정한 것을 입력하고 [OK] 버튼을 클릭

- MySQL Workbench가 MySQL 서버에 접속된 화면이 나타난다. [SQL Additions] 패널은 사용할 일이 없어 패널을 숨기겠습니다.

- 최종적으로 완성된 MySQL Workbench 화면이다. 이 책이 끝날 때까지 주로 이 화면을 사용하게 될 것이다. 가운데 빈 공간은 쿼리 창이라고 부르며 메모장처럼 글자를 입력할 수 있는데, 여기에 앞으로 배울 SQL을 입력할 것이다.

- SQL을 한 번 맛보기 위해서 간단한 SQL을 입력해보인다. 빈 공간에 SHOW DATABASES 그리고 Execute the selected portion of the script or everything 아이콘을 클릭하면 아래쪽[Result Grid] 창에 SQL에 대한 결과가 나온다. MySQL 서버에 기본적으로 들어 있는 데이터베이스의 목록을 출력해준 것이다.

- 작업을 모두 마쳤다면 [Query 1] 또는 [SQL File 숫자] 탭의 닫기(X) 버튼을 클릭해서 창을 닫는다.

- MySQL Workbench 창에서 [File] - [Exit] 메뉴를 선택하면 프로그램이 완전히 종료

*** MariaDB 다운로드와 설치**

MySQL과 MariaDB는 핵심 개발자가 같고, 문법도 비슷하기 때문에 자매 제품으로 보면 된다. 이 책에서 사용하는 SQL도 MySQL과 MariaDB에서 모두 작동한다. MariaDB는 회사에서 상용으로 작업하는 것도 무료이므로 부담 없이 사용할 수 있다. 단점이라면, MySQL보다는 인지도가 조금 떨어지고, MySQL 워크벤치보다 기능이 부족한 HeidiSQL이라는 도구를 사용한다.

MariaDB는 https://mariadb.rg에서 다운로드할 수 있다. 다운로드한 파일을 더블 클릭해서 실행하면 설치 과정은 일반적인 소프트웨어와 크게 다르지 않는다. 설치가 완료되면 바탕화면의 HeidiSQL 아이콘을 클릭해서 MariaDB에 접속할 수 있다. 사용 방법은 MySQL 워크벤치와 비슷하다.

Chapter 02. 실전용 SQL 미리 맛보기

02-1 건물을 짓기 위한 설계도 : 데이터베이스 모델링

데이터베이스 모델링(database modeling) : 테이블의 구조를 미리 설계하는 개념으로 건축 설계도를 그리는 과정과 비슷하다. 프로젝트에서 데이터베이스 모델링이 잘 되어야 제대로 된 데이터베이스를 구축할 수 있다.

폭포수 모델(waterfall model) : 프로젝트를 진행하기 위해서는 대표적으로 사용한다. 데이터베이스 모델링은 폭포수 모델의 업무 분석과 시스템 설계 단계에 해당한다. 이 단계를 거치면서 사장 중요한 데이터베이스 개체인 테이블 구조가 결정된다.

프로젝트 진행 단계

- 프로젝트(project) : 현실 세계에서 일어나는 업무를 컴퓨터 시스템으로 옮겨놓는 과정, '대규모 소프트웨어를 작성하기 위한 전체 과정'
- 폭포수 모델(waterfall model) : 소프트웨어 공학에서 가장 기본적으로 언급되는 소프트웨어 개발 절차 중 하나로, 각 단계가 폭포 떨어지듯 진행되기 때문에 붙여진 이름. 각 단계가 구분되어 프로젝트의 진행 단계가 명확하다는 장점이 있지만, 앞 단계로 돌아가기가 어렵다

프로젝트 계획 : 슈퍼마켓의 물건들을 온라인으로 판매하기 위한 계획 단계

업무 분석 : 슈퍼마켓에서 업무가 어떻게 돌아가는지 파악하는 것.

시스템 설계 : 앞에서 정리한 업무 분석을 컴퓨터에 적용시키기 위해서 알맞은 형태로 다듬는 과정

프로그램 구현 : 앞에서 완성한 시스템 설계의 결과를 실제 프로그래밍 언어로 코딩하는 단계. 온라인으로 제공하기 위해서는 Javascript, PHP, JSP 등의 프로그래밍 언어를 사용

테스트 : 코딩된 프로그램에 오류가 없는지 확인하는 과정

유지보수 : 실제 온라인 쇼핑몰을 운영하면서 문제점을 보완하고 기능을 추가하는 과정

데이터베이스 모델링

- 데이터베이스 모델링(database modeling) : 우리가 살고 있는 세상에서 사용되는 사물이나 작업을 DBMS의 데이터베이스 개체로 옮기기 위한 과정. 현실에서 쓰이는 것을 테이블로 변경하기 위한 작업
- 테이블 : 사람, 물건뿐만 아니라 실체가 없는 '물건을 산다' 라는 행동(action)도 테이블로 변환할 수 있다.
- 데이터베이스 모델링에는 정답이 없다. 다만, 좋은 모델링과 나쁜 모델링은 분명히 존재한다.

전체 데이터베이스 구성도

- 데이터(data) : 하나하나의 단편적인 정보를 말한다.
- 테이블(table) : 회원이나 제품의 데이터를 입력하기 위해 표 형태로 표현한 것을 말한다.
- 데이터베이스(Database) : 테이블이 저장되는 저장소. 각 데이터베이스는 이름이 서로 달라야 한다.

테이블과 데이터베이스의 관계는 파일과 폴더랑 비슷한 개념으로, 폴더 안에 파일이 있듯이, 데이터베이스 안에 테이블이 있다고 보면 된다.

- DBMS(Database Management System) : 데이터베이스 관리 시스템 또는 소프트웨어를 말한다.
- 열(column) : 테이블의 세로. 각 테이블은 여러 개의 열(컬럼, 필드)로 구성된다.
- 열 이름 : 각 열을 구분하기 위한 이름. 열 이름은 각 테이블 내에서는 서로 달라야 한다.

실제로 열 이름은 영문으로 설정해야 문제가 없다.

- 데이터 형식 : 열에 저장될 데이터의 형식을 말한다. 데이터 형식은 테이블을 생성할 때 열 이름과 함께 지정해준다.
- 행(row) : 실질적인 진짜 데이터를 말한다. 하나의 행(로우, 레코드)으로 행 데이터라고도 부른다. 행의 개수가 데이터의 개수이다.
- 기본 키(Primary Key, PK) : 기본 키(또는 주키) 열은 각 행을 구분하는 유일한 열을 말한다. 기본 키는 중복되어서는 안 되며, 비어 있어서도 안 된다.
- SQL(Structured Query Language) : DBMS에서 작업을 하고 싶다면, DBMS가 알아듣는 언어로 해야 한다. 그것이 SQL(구조화된 질의 언어). 즉, SQL은 사람과 DBMS가 소통하기 위한 언어이다.

02-2 데이터베이스 시작부터 끝까지

데이터베이스 구축 절차 : 데이터베이스 만들기 -> 테이블 만들기 -> 데이터 입력/수정/삭제하기 -> 데이터 조회/활용하기

DBMS 설치하기

- 데이터베이스를 구축하기 위해서는 DBMS를 설치해야 한다.

데이터베이스 만들기

- MySQL Workbench 아이콘을 클릭하고 MySQL Workbench 창의 [MySQL Connections]에서 'Local instance MySQL'을 클릭. Connect to MySQL Server 창이 나타나면 [User]는 'root'로 지정되어 있고, [Password]에 설정한 값을 입력한 후 [OK] 버튼을 클릭
- 좌측 하단에 [Administration]과 [Schemas]가 탭으로 구분되어 있다. [Schemas] 탭을 클릭하면 MySQL에 기본적으로 들어 있는 데이터베이스가 2개 보인다. 지금은 그냥 무시하고 넘어간다.
- [SCHEMAS] 패널의 빈 부분에서 마우스 오른쪽 버튼을 클릭한 후 [Create Schema]를 선택
- 새로운 쿼리 땅에서 [Name]에 쇼핑몰 의미하는 'shop_db'를 입력하면 자동으로 탭 이름도 동일하게 변경. [Apply] 버튼을 클릭하면 Apply SQL Script to Database 창에 SQL문이 자동으로 생성된다. 다시 [Apply]와 [Finish] 버튼을 클릭하면 좌측 [SCHEMAS] 패널의 목록에 'shop_db'가 추가된다.
- 탭을 닫기 위해 [File] - [Close Tab] 메뉴를 클릭하고 만약 Close SQL Tab 창이 나타나면 [Don't Save]를 선택

테이블 만들기

- 테이블 설계하기 : 테이블의 열 이름과 데이터 형식을 지정하는 것
- 데이터 형식은 모두 문자로 지정. 문자는 CHAR(Character의 약자)라는 MySQL 문법상 이미 약속된 예약어를 사용해야 함. 문자의 최대 길이도 적절히 지정할 수 있다.

널(Null) : 빈 것을 의미하며, 널 허용 안 함(Not Null, NN)은 반드시 입력해야 한다.

INT : Integer의 약자로 소수점이 없는 정수를 의미

Date : 연, 월, 일을 입력

- 테이블 생성하기 : MySQL Workbench 창의 [SCHEMAS] 패널에서 'shop_db'의 ▶를 클릭해 확장하고 [Tables]를 마우스 오른쪽 버튼으로 클릭한 후 [Create Table]을 선택

-> [Table Name(테이블 이름)]에 'member'를 입력하고, [Column Name(열 이름)]의 첫 번째 항목을 더블 클릭

-> [Column Name(열 이름)]은 'memver_id'로 입력하고 [Datatype(데이터 형식)]은 문자 8글자이므로 'CHAR(8)'이라고 입력. 설계할 때 아이디 열을 기본 키로 설정하기로 했으므로 [PK(기본 키)]와 [NN(Not Null)]을 체크

-> 회원 이름(member_name, CHAR(5))과 주소(member_addr, CHAR(20))를 추가. [NN]은 member_name만 체크

-> Apply SQL Script to Database 창이 나타나고 자동으로 생성된 SQL문이 보인다. [Apply]와 [Finish] 버튼을 클릭해서 테이블 만들기를 완료. [File] - [Close Tab] 메뉴를 선택해서 테이블 생성 창을 닫기

-> 위 방법을 반복해 제품 테이블을 반복

-> [SCHEMAS] 패널에서 [shop_db] - [Tables]를 확장하면 회원 테이블(member)과 제품 테이블(product)을 확인할 수 있다.

데이터 입력하기

- 데이터는 행(가로) 단위로 입력한다.
- MySQL Workbench 창의 [SCHEMAS] 패널에서 [shop_db] - [Tables] - [member]를 선택하고 마우스 오른쪽 버튼을 클릭한 후 [Select Rows - Limits 1000]을 선택
- MySQL Workbench 창의 중앙에 [Result Grid] 창이 나타남. 아직은 모두 NULL로 표시되어 있음. 즉, 데이터가 한 건도 없는 상태

- [member_id], [member_name], [meber_addr] 항목의 'NULL' 부분을 클릭해서 데이터를 입력
- Apply SQL Script to Datavase 창에서 [Apply]와 [Finish] 버튼을 클릭하면 데이터가 입력
- 입력된 결과가 확인. [File] - [Close Tab] 메뉴를 클릭해서 데이터 입력 창을 닫기
- 위와 같은 방법으로 제품 테이블에 데이터 3건을 추가한다.
- 수정, 삭제를 진행하기 위해 먼저 데이터를 추가. 다시 [SCHEMAS] 패널의 회원 테이블에서 마우스 오른쪽 버튼을 클릭하고 [Select Rows - Limits 1000]을 선택. 연습용 데이터를 1건 입력한 후 [Apply] 버튼을 연속 2회 클릭하고 [Finish] 버튼을 클릭해서 입력 데이터를 적용
- 조금 전 입력한 데이터 내용을 수정해보려면, 수정할 데이터를 클릭하고 변경하면 된다. 변경한 후 다시 [Apply] 버튼을 클릭
- SQL이 자동으로 생성. 이번에는 수정이기 때문에 UPDATE 문이 생성된 것을 확인할 수 있다. [Apply]와 [Finish] 버튼을 클릭해서 수정한 내용을 적용
- 데이터를 삭제해보려면, 삭제하고자 하는 행의 제일 앞 부분을 클릭하면 행이 파란색으로 선택된다. 그 상태에서 마우스 오른쪽 버튼을 클릭하고 [Delete Row]를 선택. 삭제한 후에도 역시 [Apply] 버튼을 클릭
- SQL이 자동으로 생성. 이번에는 삭제이기 때문에 DELETE 문이 생성된 것을 확인. [Apply]와 [Finish] 버튼을 클릭해서 수정한 내용을 적용

테이블 활용하기

- 새 SQL을 입력하기 위해 툴바의 Create a new SQL tab for executing queries 아이콘을 클릭. 실행된 결과를 보기 위해 Output 아이콘을 클릭해 [Output] 패널도 활성화
- 작업할 데이터베이스를 선택하기 위해 [SCHEMAS] 패널의 'shop_db'를 더블 클릭. 진하게 변경되면 앞으로 커리 창에 입력할 SQL이 선택된 shop_dp에 적용된다는 의미. 처음 MySQL을 사용할 때 자주 빼먹는 부분이므로 주의
- SELECT 열_이름 FROM 테이블_이름 [WHERE 조건], Execute the selected portion of the script or everything 아이콘을 클릭하면 [Result Grid] 창에는 결과가, [Output] 패널에는 현재 결과의 건수와 조회하는 데 소요된 시간(초) 표시
- 회원 테이블 중에 이름과 주소만 출력하겠습니다. 기존 SQL을 지우고 다음과 같이 새로 입력한 후 실행. 열 이름 부분에 회원 이름과 주소만 나왔다. 이렇게 여러 개의 열 이름을 콤마로 분리하면 필요한 열만 추출된다.
- 이번에는 앞의 SQL을 지우지 말고 다음 줄에 이어서 다음과 같이 입력한 후 실행. WHERE 다음에 특정 조건을 입력해 출력. 그런데 [Result Grid] 창의 아래쪽 탭을 보면 2개의 SQL이 모두 실행된 것을 확인 할 수 있다.
- 위와 같은 경우를 방지하기 위해 1개의 SQL만 남기고 모두 지우는 방법이 있고, 필요한 부분만 마우스로 드래그해서 선택한 후에 실행하는 것.

02-3 데이터베이스 개체

테이블은 데이터베이스의 핵심 개체이지만, 데이터베이스에서는 테이블 외에 인덱스, 뷰, 스토어드 프로시저, 트리거, 함수, 커서 등의 개체도 필요하다.

인덱스

- 인덱스(index) : '찾아보기'와 비슷한 개념
- 데이터를 조회할 때 테이블에 데이터가 적다면 결과가 금방 나오지만, 데이터가 많아질수록 결과가 나오는 시간이 많이 소요된다. 인덱스는 이런 경우 결과가 나오는 시간을 대폭 줄여준다.

뷰

- 뷰(view) : '가상의 테이블', 테이블과 상당히 동일한 성격의 데이터베이스 개체
- 뷰를 활용하면 보안도 강화하고, SQL문도 간단하게 사용할 수 있다.
- 일반 사용자는 테이블과 동일하게 뷰를 취급하면 된다. 다만 뷰는 실제 데이터를 가지고 있지 않으며, 진짜 테이블에 링크된 개념이라고 생각하면 된다.
- 뷰의 실체는 SELECT 문이다.
- 보안에 도움되고 긴 SQL문을 간략하게 만들 수 있다.

스토어드 프로시저

- 스토어드 프로시저(stored procedure) : MySQL에서 제공하는 프로그래밍 기능, 여러 개의 SQL문을 하나로 묶어서 편리하게 사용할 수 있다.
- 일반 프로그래밍보다는 좀 불편하지만, 프로그래밍 로직을 작성할 수 있어 때론 유용하게 사용된다.

Chapter 03. SQL 기본 문법

03-1 기본 중에 기본 SELECT ~ FROM ~ WHERE

SELECT 문 : 구축이 완료된 테이블에서 데이터를 추출하는 기능

SELECT를 아무리 많이 사용해도 기존의 데이터가 변경되지는 않는다.

SELECT의 가장 기본 형식은 SELECT ~ FROM ~ WHERE

SELECT 다음에는 열 이름, FROM 다음에는 테이블 이름, WHERE 다음에는 조건식이 나오는데 조건식을 다양하게 조건식을 다양하게 표현함으로써 데이터베이스에서 원하는 데이터를 뽑아낼 수 있다.

기본 조회하기 : SELECT ~ FROM

- USE문 : SELECT 문을 실행하려면 머넷 사용할 데이터베이스를 지정해야 한다. 현재 사용하는 데이터베이스를 지정 또는 변경하는 형식은 USE 데이터베이스_이름;

지금부터 이 DB를 사용하겠으니 모든 커리는 이 DB에서 실행하라는 의미

- SELECT문의 기본 형식 : 처음에는 사용하기 간단하지만, 사실 상당히 복잡한 구조를 갖는다.

```
SELECT 열_이름
        FROM 테이블_이름
        WHERE 조건식
        GROUP BY 열_이름
        HAVING 조건식
        ORDER BY 열_이름
        LIMIT 숫자
```

특정한 조건만 조회하기 : SELECT ~ FROM ~ WHERE

- WHERE 없이 조회하기 : WHERE 없이 테이블을 조회하면 테이블의 모든 행이 출력된다.
- 기본적인 WHERE 절 : WHERE 절은 조회하는 결과에 특정한 조건을 추가해서 원하는 데이터만 보고 싶을 때 사용.

SELECT 열_이름 FROM 테이블_이름 WHERE 조건식;

- 관계 연산자, 논리 연산자의 사용 : 숫자로 표현된 데이터는 범위를 지정할 수 있다.

관계 연산자는 >, <, >=, <=, = 등

논리 연산자는 AND, OR, NOT 등

- BETWEEN ~ AND : 범위에 있는 값을 구하는 경우
- IN() : BETWEEN ~ AND를 사용할 수 있지만, 어느 범위에도 들어 있지 않은 것을 표현하기 위해 OR도 일일이 써줘야 한다. 하지만, IN()을 사용하면 코드를 훨씬 간결하게 작성할 수 있다.
- LIKE : 문자열의 이름 글자를 검색할 때 사용한다. 문자열 비교 시 무엇이든(%) 허용한다. 한 글자와 매치하기 위해서는 언더바(_) 사용.

03-2 좀 더 깊게 알아보는 SELECT문

ORDER BY 절

- ORDER BY 절 : 결과의 값이나 개수에 대해서는 영향을 미치지 않지만, 결과가 출력되는 순서를 조절한다.

```
SELECT 열_이름
        FROM 테이블_이름
        WHERE 조건식
        GROUP BY 열_이름
        HAVING 조건식
        ORDER BY 열_이름
        LIMIT 숫자
```

- 기본 값은 ASC 오름차순을 의미하고, DESC는 내림차순 의미한다.

- LIMIT : 출력의 개수를 제한.
LIMIT 시작, 개수
LIMIT 개수 OFFSET 시작
- DISTINCT : 중복된 결과를 제거
GROUP BY 절
- GROUP BY 절 : 그룹을 묶어주는 역할을 하고, 집계 함수는 주로 GROUP BY 절과 함께 쓰이며 데이터를 그룹화 해주는 기능을 한다.
SELECT 열_이름
    FROM 테이블_이름
    WHERE 조건식
    GROUP BY 열_이름
    HAVING 조건식
    ORDER BY 열_이름
    LIMIT 숫자
- 집계 함수
SUM() : 합계를 구합니다.
AVG() : 평균을 구합니다.
MIN() ; 최소값을 구합니다.
MAX() : 최대값을 구합니다.
COUNT() : 행의 개수를 셉니다.
COUNT(DISTINCT) : 행의 개수를 셉니다(중복은 1개만 인정).
- Having 절 : WHERE 대신에 사용되는 것으로, 집계 함수와 관련된 조건을 제한하며, GROUP BY 다음에 나온다.

03-3 데이터 변경을 위한 SQL 문

데이터 입력 : INSERT

- INSERT 문의 기본 문법 : 테이블에 데이터를 삽입하는 명령
INSERT INTO 테이블[(열1, 열2, ...)] VALUES (값1, 값2, ...)
- 테이블 이름 다음에 나오는 열은 생략 가능. 열 이름을 생략할 경우에 VALUES 다음에 나오는 값들의 순서 및 개수는 테이블을 정의할 때의 열 순서 및 개수와 동일해야 한다.
- 자동으로 증가하는 AUTO_INCREMENT : 열을 정의할 때 1부터 증가하는 값을 입력해준다. INSERT에서는 해당 열이 없다고 생각하고 입력하면 된다. 꼭 PRIMARY KEY로 지정해줘야 한다.
- 다른 테이블의 데이터를 한 번에 입력하는 INSERT INTO ~ SELECT : 다른 테이블에 이미 데이터가 입력되어 있다면 구문을 사용해 해당 테이블의 데이터를 가져와서 한 번에 입력할 수 있다.
INSERT INTO 테이블_이름 (열_이름1, 열_이름2, ...)
    SELECT 문;

데이터 수정 : UPDATE

- UPDATE 문의 기본 문법 : 기존에 입력되어 있는 값을 수정하는 명령
UPDATE 테이블_이름
    SET 열1=값1, 열2=값2, ...
    WHERE 조건;
- WHERE가 없는 UPDATE 문 : UPDATE 문에서 WHERE 절은 문법상 생략이 가능하지만, WHERE 절을 생략하면 테이블의 모든 행의 값이 변경된다. 일반적으로 전체 행의 값을 변경하는 경우는 별로 없으므로 주의해야 한다.
데이터 삭제 : DELETE
- DELETE : 행 데이터를 삭제하는 명령

DELETE FROM 테이블이름 WHERE 조건 ;

- DELETE도 UPDATE와 마찬가지로 WHERE 절이 생략되면 전체 행 데이터를 삭제하므로 주의해야 한다.

*** 대용량 테이블의 삭제**

동일한 내용의 대용량 테이블 3개를 DELETE, DROP, TRUNCATE 각각 다른 방법으로 삭제해보면, DELETE 문은 삭제가 오래 걸린다. DROP 문은 테이블 자체를 삭제한다. 그래서 순식간에 삭제된다. TRUNCATE 문도 DELETE와 동일한 효과를 내지만 속도가 무척 빠르다.

DROP은 테이블이 아예 없어지지만, DELETE와 TRUNCATE는 빈 테이블을 남긴다. 즉, 대용량 테이블의 전체 내용을 삭제할 때 테이블 자체가 필요 없을 경우에는 DROP으로 삭제하고, 테이블의 구조는 남겨놓고 싶다면 TRUNCATE로 삭제하는 것이 효율적이다.

Chapter 04. SQL 고급 문법

04-1 MySQL의 데이터 형식

데이터 형식

- 정수형 : 소수점이 없는 숫자, 즉 인원수, 가격, 수량 등에 많이 사용

데이터 형식	바이트 수	숫자 범위
TINYINT	1	-128 ~ 127
SMALLINT	2	-32,768 ~ 32,767
INT	4	약 -21억 ~ +21억
BIGINT	8	약 -900경 ~ +900경

- 문자형 : 글자를 저장하기 위해 사용하며, 입력할 최대 글자의 개수를 지정

데이터 형식	바이트 수
CHAR(개수)	1 ~ 255
VARCHAR(개수)	1 ~ 16383

- 대량의 데이터 형식

데이터 형식		바이트 수
TEXT 형식	TEXT	1 ~ 65535
	LONGTEXT	1 ~ 4294967295
BLOB 형식	BLOB	1 ~ 65535
	LONGBLOB	1 ~ 4294967295

- 실수형 : 소수점이 있는 숫자를 저장할 때 사용

데이터 형식	바이트 수	설명
FLOAT	4	소수점 아래 7자리까지 표현
DOUBLE	8	소수점 아래 15자리까지 표현

- 날짜형 : 날짜 및 시간을 저장할 때 사용

데이터 형식	바이트 수	설명
DATE	3	날짜만 저장. YYY-MM-DD 형식으로 사용
TIME	3	시간만 저장. HH:MM:SS 형식으로 사용
DATETIME	8	날짜 및 시간을 저장. YYYY-MM-DD HH:MM:SS 형식으로 사용

변수의 사용

- MySQL에서 변수는 @변수이름 형식으로 만들고, SET 문으로 변수에 값을 대입한다.

```
SET @변수이름 = 변수의 값;        -> 변수의 선언 및 값 대입
SELECT @변수이름 ;                -> 변수의 값 출력
```

- LIMIT : SELECT 문에서 행의 개수를 제한
- PREPARE와 EXECUTE : PREPARE는 실행하지 않고 SQL 문만 준비해 놓고 EXECUTE에서 실행하는 방식, EXECUTE는 SELECT 문을 실행할 때, USING으로 물음표(?)에 @count 변수의 값을 대입하는 것이다.

데이터 형 변환

- 데이터의 형 변환(type conversion) : 문자형을 정수형으로 바꾸거나, 반대로 정수형을 문자형으로 바꾸는 것
- 함수를 이용한 명시적인 변환(explicit conversion) : 직접 함수를 사용해서 변환

```
CAST ( 값 AS 데이터_형식 [ (길이) ] )
CONVERT ( 값, 데이터_형식 [ (길이) ] )
```

- 암시적인 변환(implicit conversion) : 별도의 지시 없이 자연스럽게 변환

04-2 두 테이블을 묶는 조인

조인(join) : 두 개의 테이블을 서로 묶어서 하나의 결과를 만들어 내는 것

내부 조인

- 내부 조인 : 두 테이블을 연결할 때 가장 많이 사용되는 것

- 일대다 관계의 이해 : 두 테이블의 조인을 위해서는 테이블이 일대다(one to many) 관계로 연결되어야 한다. 데이터베이스의 테이블은 하나로 구성되는 것보다는 여러 정보를 주제에 따라 분리해서 저장하는 것이 효율적이다. 이 분리된 테이블은 서로 관계를 맺고 있다.

- 내부 조인의 기본 : 조인 중에서 가장 많이 사용된다.

```
SELECT <열 목록>
FROM <첫 번째 테이블>
        INNER JOIN <두 번째 테이블>
        ON <조인될 조건>
[WHERE 검색 조건]
```

- INNER JOIN을 그냥 JOIN이라고만 써도 INNER JOIN으로 인식

- 내부 조인의 간결한 표현 : 테이블_이름.열_이름

외부 조인

- 외부 조인 : 두 테이블을 조인할 때 필요한 내용이 한쪽 테이블에만 있어도 결과를 추출할 수 있다. 자주 사용되지는 않지만, 가끔 사용되는 방식이다.

```
SELECT <열 목록>
FROM <첫 번째 테이블(LEFT 체이블)>
        <LEFT : RIGHT : FULL> OUTER JOIN <두 번째 테이블(RIGHT 테이블)>
        ON <조인될 조건>
[WHERE 검색 조건] ;
```

- LEFT OUTER JOIN : 왼쪽 테이블의 내용은 모두 출력되어야 한다.

- RIGHT OUTER JOIN : 동일한 결과를 출력하려면 단순히 왼쪽과 오른쪽 테이블의 위치만 바꿔주면 된다.

- FULL OUTER JOIN : 왼쪽 외부 조인과 오른쪽 외부 조인이 합쳐진 것이다. 왼쪽이든 오른쪽이든 한쪽에 들어 있는 내용이면 출력한다.

기타 조인

- 상호 조인 : 한쪽 테이블의 모든 행과 다른 쪽 테이블의 모든 행을 조인시키는 기능을 말한다. 상호 조인 결과의 전체 행 개수는 두 테이블의 각 행의 개수를 곱한 개수가 된다.

ON 구문을 사용할 수 없다. 결과의 내용은 의미가 없다. (랜덤으로 조인함) 상호 조인의 주 용도는 테스트하기 위해 대용량의 데이터를 생성할 때이다.

- 자체 조인 : 자신이 자신과 조인한다. 즉 한 개의 테이블을 사용한다.

```
SELECT <열 목록>
FROM <테이블> 별칭A
        INNER JOIN <테이블> 별칭B
        ON <조인될 조건>
[WHERE 검색 조건]
```

04-3 SQL 프로그래밍

스토어드 프로시저는 MySQL에서 프로그래밍 기능이 필요할 때 사용하는 데이터베이스 개체이다. SQL 프로그래밍은 기본적으로 스토어드 프로시저 안에 만들어야 한다.

```
DELIMITER $$
CREATE PROCEDURE 스토어드_프로시저_이름()
BEGIN
        이 부분에 SQL 프로그래밍 코딩
```

END $$
DELIMITER ;
CALL 스토어드_프로시저_이름();

IF 문

- IF 문 : 조건문으로 가장 많이 사용되는 프로그래밍 문법 중 하나
- IF 문의 기본 형식 : IF 문은 조건식이 참이라면 'SQL문장들' 을 실행하고, 그렇지 않으면 그냥 넘어간다.

```
IF <조건문> THEN
        SQL문장들
END IF;
```

'SQL문장들' 이 한 문장이라면 그 문장만 써도 되지만, 두 문장 이상이 처리되어야 할 때는 BEGIN~END로 묶어줘야 한다.

- IF~ELSE 문 : IF~ELSE 문 조건에 따라 다른 부분을 수행한다. 조건식이 참이라면 'SQL문장들1' 을 실행하고, 그렇지 않으면 'SQL문장들2' 를 실행한다.

CASE 문

- CASE 문의 기본 형식 : IF 문은 참 아니면 거짓 두 가지만 있기 때문에 2중 분기라는 용어를 사용. CASE 문은 두 가지 이상의 여러 가지 경우일 때 처리가 가능하므로 '다중 분기' 라고 부른다.

```
CASE
        WHEN 조건1 THEN
                SQL문장들1
        WHEN 조건2 THEN
                SQL문장들2
        WHEN 조건3 THEN
                SQL문장들3
        ELSE
                SQL문장들4
END CASE;
```

WHILE 문

- WHILE 문의 기본 형식 : WHILE 문은 조건식이 참인 동안에 'SQL문장들' 을 계속 반복한다.

```
WHILE <조건식> DO
        SQL 문장들
END WHILE;
```

- WHILE 문의 응용 : ITERATE[레이블] - 지정한 레이블로 가서 계속 진행합니다.
LEAVE[레이블] - 지정한 레이블을 빠져나간다. 즉 WHILE 문이 종료된다.

동적 SQL

- SQL 문은 내용이 고정되어 있는 경우가 대부분이다. 하지만, 상황에 따라 내용 변경이 필요할 때 동적 SQL을 사용하면 변경되는 내용을 실시간으로 적용시켜 사용할 수 있다.
- PREPARE는 SQL문을 실행하지는 않고 미리 준비만 해놓고, EXECUTE는 준비한 SQL 문을 실행한다. 그리고 실행 후에는 DEALLOCATE PREFARE로 문장을 해제해주는 것이 바람직하다.

Chapter 05. 테이블과 뷰

05-1 테이블 만들기

테이블(table)은 표 형태로 구성된 2차원 구조로, 행과 열로 구성되어 있다.
행은 로우(row)나 레코드(record)라 부르며, 열은 컬럼(column) 또는 필드(field)라고 부른다.

데이터베이스와 테이블 설계하기

- 테이블을 만들기 전에 설계를 먼저 해야 한다. 테이블 설계는 테이블 이름, 열 이름, 데이터 형식, 기본 키 등을 설정하는 것이다.
- 데이터 형식을 활용해서 각 열에 가장 적합한 데이터 형식을 지정한다.

GUI 환경에서 테이블 만들기

- 데이터베이스 생성하기 : 접속 후 새 쿼리 창을 생성한다. CREATE DATABASE 데이터베이스 명; 입력하고 Execute the selected portion of the script or everything을 클릭한다. SQL로 만든 데이터베이스는 화면에 바로 적용되지 않기 때문에 SCHEMAS 패널에 보이지 않는다. Refresh ALL을 선택하면 보인다.
- 테이블 생성하기 : Tables를 선택 후 Create Table을 선택한다. 설계한 대로 테이블을 구성하고 완료되었으면 Apply 버튼을 클릭해 적용한다. Apply SQL Script to Database 창에서 생성된 CREATE TABLE 코드를 확인할 수 있다.
- 데이터 입력하기 : SCHEMAS 패널에서 테이블 명 - Tables - 컬럼 명을 선택하고 Select Rows - Limit 1000을 선택한다. SELECT 문이 자동으로 생성되고 Result Grid 창에 같은 결과도 보인다.

SQL로 테이블 만들기

- 데이터베이스 생성하기 : 새 쿼리 창을 하나 준비하고 앞과 같이 실행한다.
- 테이블 생성하기 : CREATE TABLE 구문으로 옵션을 추가해서 테이블을 생성한다. 이어서 테이블에 기본 키를 설정해 준다. 열 이름과 데이터 형식을 먼저 지정한 후에, 나머지 조건들을 차근차근 설정하면 SQL로도 어렵지 않게 테이블을 만들 수 있다.
- 데이터 입력하기 : INSERT INTO 구문을 통해 데이터를 입력한다.

* 열에 입력될 값이 1부터 자동 증가하도록 설정하면 GUI에서는 AI를 체크하고, SQL에서는 AUTO_INCREMENT 입력한다. 열에 빈 값을 허용하지 않으려면 GUI에서는 NN을 체크하고, SQL에서는 NOT NULL을 입력한다. 열을 기본 키로 지정하려면 GUI에서는 PK를 체크하고, SQL에서는 PRIMARY KEY를 입력한다. 열을 외래 키로 지정하려면 FOREIGN KEY 예약어를 입력한다.

용어	설명
GUI	Graphical User Interface의 약자로, 윈도에서 진행하는 작업을 의미
로우(row)	테이블의 행, 레코드(record)라고도 부름
컬럼(column)	테이블의 열, 필드(field)라고도 부름
UNSIGNED	정수형 뒤에 붙이면 0부터 양의 정수만 입력됨
백틱(`)	테이블 이름이나 열 이름을 묶을 때 사용
NULL	열에 비어 있는 값을 허용할 때 설정함(별도로 지정하지 않으면 기본은 NULL)
기본 키-외래 키	두 테이블이 일대다로 연결되는 관계
주석(remark)	하이픈(-) 2개 이후에 한 칸을 띄고 설명을 써야 함

05-2 제약조건으로 테이블을 견고하게

테이블을 만들 때는 테이블의 구조에 필요한 제약조건을 설정해줘야 한다. 제약조건에는 대표적으로 기본 키와 외래 키, 고유 키, default 값, NOT NILL 등이 있다.

제약조건의 기본 개념과 종류

- 제약조건(constrint) : 데이터의 무결성을 지키기 위해 제한하는 조건이다. 데이터의 무결성이란, 데이터의 결함이 없음을 의미한다.
- PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, DEFAULT, NULL

기본 키 제약조건

- 기본 키(Primary Key) : 데이터를 구분할 수 있는 식별자
- 기본 키에 입력되는 값은 중복될 수 없으며, NULL 값이 입력될 수 없다.
- 대부분의 테이블은 기본 키를 가져야 한다. 기본 키가 없어도 테이블 구성이 가능하지만 실무에서 사용하는 테이블에는 기본 키를 설정해야 중복된 데이터가 입력되지 않는다.
- CREATE TABLE에서 설정하는 기본 키 제약조건은 열 이름 뒤에 PRIMARY KEY를 붙여주면 기본 키로 설정된다.
- ALTER TABLE에서 설정하는 기본 키 제약조건은 테이블 수정하고 제약조건을 추가한다.

외래 키 제약조건

- 외래 키(Foreign Key) : 두 테이블 사이의 관계를 연결해주고, 그 결과 데이터의 무결성을 보장해주는 역할을 한다. 외래 키가 설정된 열은 꼭 다른 테이블의 기본 키와 연결된다.
- 기준 테이블 : 기본 키가 있는 테이블, 참조 테이블 : 외래 키가 있는 테이블
- 참조 테이블이 참조하는 기준 테이블의 열은 반드시 기본 키나 고유 키로 설정되어 있어야 한다.
- CREATE TABLE에서 설정하는 외래 키 제약조건은 CREATE TABLE 끝에 FOREIGN KEY 키워드를 설정한다.

FOREIGN KEY(열_이름) REFERENCES 기준_테이블(열_이름)

- ALTER TABLE에서 설정하는 외래 키 제약조건은 테이블을 수정하고 제약조건을 추가한다.

기타 제약조건

- 고유 키(Unique) 제약조건 : 중복되지 않는 유일한 값을 입력해야 하는 조건이다. 기본 키 제약조건과 거의 비슷하지만, 차이점은 고유 키 제약조건은 NULL 값을 허용한다는 것이다.
- 체크(Check) 제약조건 : 입력되는 데이터를 점검하는 기능을 한다. 열의 정의 뒤에 CHECK(조건)을 추가해주면 된다.
- 기본값(Default) 정의 : 값을 입력하지 않았을 때 자동으로 입력될 값을 미리 지정해 놓는 방법이다. 기본값이 설정된 열에 기본값을 입력하려면 default라고 써주고, 원하는 값을 입력하려면 해당 값을 써주면 된다.
- 널 값 허용 : 널 값을 허용하려면 생략하거나 NULL을 사용하고, 허용하지 않으려면 NOT NULL을 사용한다. PRIMARY KEY가 설정된 열에는 NULL 값이 있을 수 없으므로 생략하면 자동으로 NOT NULL로 인식한다.

용어	설명
제약조건	데이터의 무결성을 지키기 위한 제한된 조건
ALTER TABLE	이미 만들어진 테이블을 수정하는 SQL 문
ADD CONSTRAINT	제약조건을 추가하는 SQL 문
기준 테이블	기본 키-외래 키 관계가 맺어진 테이블 중 기본 키가 설정된 테이블
참조 테이블	기본 키-외래 키 관계가 맺어진 테이블 중 외래 키가 설정된 테이블
ON UPDATE CASCADE	기준 테이블의 기본 키를 변경하면 참조 테이블의 외래 키도 변경되는 기능
ON DELETE CASCADE	기준 테이블의 기본 키를 삭제하면 참조 테이블의 외래 키도 삭제되는 기능

05-3 가상의 테이블 : 뷰

뷰(view)는 데이터베이스 개체 중 하나다. 테이블과 아주 밀접한 연관이 있다. 한 번 생성해 놓으면 테이블이라고 생각하고 사용해도 될 정도로 사용자 입장에서는 테이블과 거의 동일한 개체로 취급한다. SELECT 문으로 만들어져 있기 때문에 뷰에 접근하는 순간 SELECT가 실행되고 그 결과가 화면에 출력되는 방식이다.

단순 뷰는 하나의 테이블과 연관된 뷰를 말하고, 복합 뷰는 2개 이상의 테이블과 연관된 뷰를 말한다.

뷰의 개념

- 뷰의 기본 생성

```
CREATE VIEW 뷰_이름
AS
    SELECT 문;
```

뷰를 만든 후에 뷰에 접근하는 방식은 테이블과 동일하게 SELECT 문을 사용한다. 전체에 접근할 수도 있고, 필요하면 조건식도 테이블과 동일하게 사용할 수 있다.

```
SELECT 열_이름 FROM 뷰_이름
```

[WHERE 조건];

- 사용자 -> 조회 또는 변경 -> SELECT 문(뷰) -> 쿼리 실행 -> 데이터(테이블) -> 쿼리 결과값 -> SELECT 문(뷰) -> 결과 -> 사용자

- 뷰를 사용하는 이유? 보안에 도움이 된다. 복잡한 SQL을 단순하게 만들 수 있다.

뷰의 실제 작동

- 뷰의 실제 생성, 수정, 삭제 : 뷰를 생성하면서 뷰에서 사용될 열 이름을 테이블과 다르게 지정할 수도 있다. 별칭을 사용하면 되는데, 중간에 띄어쓰기 사용이 가능하다. 또한, 형식상 AS를 붙여준다. 뷰를 조회할 때는 열 이름에 공백이 있으면 백틱으로 묶어줘야 한다.

수정은 ALTER VIEW 구문을 사용하며, 열 이름에 한글을 사용해도 된다.

삭제는 DROP VIEW를 사용한다.

- 뷰의 정보 확인 : DESCRIBE 문으로 기존 뷰의 정보를 확인할 수 있다.

SHOW CREATE VIEW 문으로 뷰의 소스 코드도 확인할 수 있다.

- 뷰를 통한 데이터의 수정/삭제 : UPDATE로 데이터를 수정할 수 있다.

INSERT를 통해 데이터를 입력할 수 있다.

- 뷰를 통한 데이터 입력 : 뷰에서 보이지 않는 테이블의 열에 NOT NULL이 없어야 한다.

WITH CHECK OPTION을 통해 뷰에 설정된 값의 범위가 벗어나는 값은 입력되지 않도록 할 수 있다. 설정한 범위의 데이터만 입력되도록 제한한다.

- 뷰가 참조하는 테이블의 삭제 : DROP을 통해 테이블을 삭제할 수 있다. 삭제후 SELECT를 통해 조회를 해보면 참조하는 테이블이 없으면 조회할 수 없다는 메시지가 나온다. 관련 뷰가 있더라도 테이블은 쉽게 삭제가 된다. 뷰가 조회되지 않으면 CHECK TABLE 문으로 뷰의 상태를 확인해볼 수 있다.

용어	설명
CREATE VIEW	뷰를 생성하는 SQL
별칭	뷰에서 사용될 열의 이름을 별칭을 사용해서 테이블과 다르게 지정할 수도 있음
백틱	뷰를 조회할 때 열 이름에 공백이 있으면 붙여주는 기호
ALTER VIEW	뷰를 수정하는 SQL
DROP VIEW	뷰를 삭제하는 SQL
CREATE OR REPLACE VIEW	기존에 뷰가 있으면 덮어쓰고, 없으면 새로 생성하는 SQL
DESCRIBE	뷰 또는 테이블의 정보를 조회하는 SQL
SHOW CREATE VIEW	뷰의 소스 코드를 보여주는 SQL
WITH CHECK OPTION	뷰에 설정된 조건만 입력되도록 지정하는 SQL
CHECK TABLE	뷰 또는 테이블의 상태를 확인하는 SQL

Chapter 06. 인덱스

06-1 인덱스 개념을 파악하자

인덱스(index)는 데이터를 빠르게 찾을 수 있도록 도와주는 도구로, 실무에서는 현실적으로 인덱스 없이 데이터베이스 운영이 불가능하다.

인덱스의 개념

- 인덱스 : 책의 '찾아보기', '색인'과 비슷한 개념이다. 즉, 인덱스는 데이터를 빠르게 찾을 수 있도록 해주는 도구이다. 데이터를 찾을 때, 인덱스의 사용 여부에 따른 결과값의 차이는 없다. 책의 찾아보기가 없다고 글자를 못 찾는 것은 아니다. 단지 시간이 오래 걸릴 뿐이다.
- 인덱스의 문제점 : 필요 없는 인덱스를 만들면 데이터베이스가 차지하는 공간만 더 늘어나고, 인덱스를 이용해서 데이터를 찾는 것이 전체 테이블을 찾아보는 것보다 느려진다.
- 인덱스의 장점과 단점 : SELECT에서 즉각적인 효과를 내는 빠른 방법 중 하나로 적절한 인덱스를 생성하고 인덱스를 사용하는 SQL을 만든다면 기존보다 아주 빠른 응답 속도를 얻을 수 있다. 컴퓨터 입장에서는 적은 처리량으로 요청한 결과를 빨리 얻을 수 있으니 여유가 생기고 추가로 더 많은 일을 할 수 있게 된다. 결과적으로 전체 시스템의 성능이 향상되는 효과도 얻게 된다.
- 장점 : SELECT 문으로 검색하는 속도가 매우 빨라진다. 그 결과 컴퓨터의 부담이 줄어들어서 결국 전체 시스템의 성능이 향상된다.
- 단점 : 인덱스도 공간을 차지해서 데이터베이스 안에 추가적인 공간이 필요하다. 처음에 인덱스를 만드는 데 시간이 오래 걸릴 수 있다. SELECT가 아닌 데이터의 변경 작업(INSERT, UPDATE, DELETE)이 자주 일어나면 오히려 성능이 나빠질 수도 있다.

인덱스의 종류

- 클러스트형 인덱스(Clustered Index) : 기본 키로 지정하면 자동 생성되며 테이블에 1개만 만들 수 있다. 기본 키로 지정한 열을 기준으로 자동 정렬된다.
- 보조 인덱스(Secondary Index) : 고유 키로 지정하면 자동 생성되며 여러 개를 만들 수도 있지만, 자동 정렬되지는 않는다.
- 자동으로 생성되는 인덱스 : 하나의 열에 여러 개의 인덱스를 생성할 수도 있고, 여러 개의 열을 묶어서 하나의 인덱스를 생성할 수도 있다. 하지만 상당히 드문 경우이므로 하나의 열에 하나의 인덱스를 생성한다.
- 고유 인덱스(Unique Index) : 인덱스의 값이 중복되지 않다는 의미고, 단순 인덱스는 인덱스의 값이 중복되어도 된다는 의미이다. 기본 키나 고유 키로 지정하면 값이 중복되지 않으므로 고유 인덱스가 생성된다. 그 외의 인덱스는 단순 인덱스로 생성된다.
- 보조 인덱스 : 고유 키로 지정하면 자동으로 생성되며, 테이블당 여러 개 만들 수 있다.
- 자동으로 정렬되는 클러스트형 인덱스 : 클러스트형 인덱스가 생성된 열로 데이터가 자동 정렬된다. 클러스트형 인덱스는 테이블에 1개만 생성할 수 있다. 기본 키가 테이블에 1개인 것과 마찬가지이다.
- 정렬되지 않는 보조 인덱스 : 고유 키로 지정하면 보조 인덱스가 생성된다. 그리고 보조 인덱스는 테이블에 여러 개 설정할 수 있다. 고유 키를 테이블에 여러 개 지정할 수 있는 것과 마찬가지이다.

* 기본 키 변경 시 주의할 점 : 이미 대용량의 데이터가 있는 상태에서 기본 키를 지정하면 시간이 엄청 오래 걸릴 수 있다. 기본 키는 중복되지 않아야 한다.

	클러스트형 인덱스	보조 인덱스
영문	Clustered Index	Secondary Index
관련 제약조건	기본 키(Primary Key)	고유 키(Unique Key)
테이블당 개수	1개	여러 개
정렬	지정한 열로 정렬됨	정렬되지 않음
비유	영어사전	일반 책의 찾아보기

06-2 인덱스 내부 작동

클러스트형 인덱스와 보조 인덱스는 모두 내부적으로 균형 트리로 만들어진다. 균형 트리(Balanced tree, B-tree)는 자료 구조에 나오는 범용적으로 사용되는 데이터의 구조이다. 나무를 거꾸로 표현한 자료 구조로, 트리에서 제일 상단의 뿌리를 루트, 줄기를 중간, 끝에 달린 잎을 리프라고 부른다.

인덱스의 내부 작동 원리

- 균형 트리의 개념 : 균형 트리 구조에서 데이터가 저장되는 공간을 노트라고 한다. 루트 노드는 노드의 가장 상위 노드, 모든 출발은 루트 노드에서 시작된다. 리프 노드는 제일 마지막에 존재하는 노드를 말한다. 루트 노드와 리프 노드의 중간에 끼인 노드들은 중간 노드라 부른다.
- 균형 트리로 구성하지 않으면(인덱스가 없으면) 전체 페이지를 검색하는 방법밖에 없다. 데이터를 처음부터 끝까지 검색하는 것을 전체 테이블 검색(Full Table Scan)이라고 부른다.
- 균형 트리로 구성하면(인덱스가 있으면) 데이터를 빠르게 검색할 수 있다.
- 균형 트리의 페이지 분할 : 새로운 페이지를 준비해서 데이터를 나누는 작업을 말한다. 인덱스를 구성하면 데이터 변경 작업(INSERT, UPDATE, DELETE) 시 성능이 나빠진다. 인덱스가 있으면 데이터 변경 작업은 오히려 느려진다.

인덱스의 구조

- 클러스터형 인덱스 구성하기 : 기본 키로 지정하면 클러스터형 인덱스가 생성되고 데이터가 자동으로 정렬된다. 행 데이터를 지정한 열로 정렬한다. 각 페이지의 인덱스로 지정된 열의 첫 번째 값을 가지고 루트 페이지를 만든다. 루프 페이지와 리프 페이지(중간 페이지가 있다면 중간 페이지도 포함)로 구성되어 있다. 또한 인덱스 페이지의 리프 페이지는 데이터 그 자체이다.
- 보조 인덱스 구성하기 : 보조 인덱스는 데이터 페이지를 건드리지 않는다. 인덱스가 별도의 공간에 만들 진다. 데이터 페이지는 변경되지 않는다. 우선 인덱스 페이지의 리프 페이지에 인덱스로 구성한 열을 정렬한다. 실제 데이터가 있는 위치를 준비한다. 데이터의 위치는 페이지 번호 +#위치로 기록되어 있다.
- 인덱스에서 데이터 검색하기 : 인덱스 검색(Index Scan)을 통해 클러스터형 인덱스 또는 보조 인덱스를 이용해서 데이터를 검색한다. 속도는 인덱스를 사용하지 않았을 때보다 빠르다.
- 전체 테이블 검색 : 데이터를 처음부터 끝까지 검색하는 것이다. 인덱스가 없으면 전체 페이지를 검색하는 방법밖에 없다.
- 페이지 분할 : 데이터를 입력할 때, 입력할 페이지에 공간이 없어서 2개 페이지로 데이터가 나뉘지는 것을 말한다.
- 인덱스 검색 : 클러스터형 또는 보조 인덱스를 이용해서 데이터를 검색하는 것이다. 인덱스를 사용하지 않았을 때보다 빠르다.

06-3 인덱스의 실제 사용

인덱스를 생성하기 위해서는 CREATE INDEX 문을 사용하고, 제거하기 위해서는 DROP INDEX 문을 사용한다.
보조 인덱스는 데이터의 중복 여부에 따라 단순 보조 인덱스와 고유 보조 인덱스로 나뉜다.

인덱스 생성과 제거 문법

- 인덱스 생성 문법 :

```
CREATE [UNIQUE] INDEX 인덱스_이름
    ON 테이블_이름 (열_이름) [ASC | DESC]
```

- 인덱스 제거 문법 : 기본 키, 고유 키로 자동 생성된 인덱스는 DROP INDEX로 제거하지 못한다. ALTER TABLE 문으로 기본 키나 고유 키를 제거하면 자동으로 생성된 인덱스도 제거할 수 있다.

```
DROP INDEX 인덱스_이름 ON 테이블_이름
```

- 단순 보조 인덱스 : 중복을 허용하는 보조 인덱스, CREATE INDEX 문을 사용한다.
- 고유 보조 인덱스 : 중복을 허용하지 않는 보조 인덱스, CREATE UNIQUE INDEX문을 사용한다.

*** 인덱스를 효과적으로 사용하는 방법?**

- 인덱스는 열 단위에 생성된다.
- WHERE 절에서 사용되는 열에 인덱스를 만들어야 한다.
- WHERE 절에 사용되더라도 자주 사용해야 가치가 있다.
- 데이터의 중복이 높은 열은 인덱스를 만들어도 별 효과가 없다.
- 클러스터형 인덱스는 테이블당 하나만 생성할 수 있다.
- 사용하지 않는 인덱스는 제거한다.

Chapter 07. 스토어드 프로시저

07-1 스토어드 프로시저 사용 방법

SQL + 프로그래밍 기능 -> 스토어드 프로시저

스토어드 프로시저를 사용하면 복잡한 SQL을 한 번에 실행할 수 있습니다.

스토어드 프로시저 기본

- 스토어드 프로시저의 개념과 형식 : 스토어드 프로시저(stored procedure)란 MySQL에서 제공하는 프로그래밍 기능이다. 커피 문의 집합으로도 볼 수 있으며, 어떠한 동작을 일괄 처리하기 위한 용도로도 사용한다. 스토어드 프로시저도 데이터베이스의 개체 중 한 가지입니다. 즉, 테이블처럼 각 데이터베이스 내부에 저장됩니다.

```
DELIMITER $$
CREATE PROCEDURE 스토어드_프로시저_이름(IN 또는 OUT 매개변수)
BEGIN
        SQL 프로그래밍 코드
END $$
DELIMITER ;
```

- 스토어드 프로시저의 생성 : CALL 스토어드_프로시저_이름();

GREATE PROCEURE~는 스토어드 프로시저를 만들고, CALL~은 스토어드 프로시저를 실행(호출) 합니다.

- 스토어드 프로시저의 삭제 : DROP PROCEURE~는 스토어드 프로시저를 삭제합니다.

스토어드 프로시저 실습

- 매개변수의 사용 : 스토어드 프로시저에서는 실행 시 입력 매개변수를 지정할 수 있다. 매개변수는 다른 용어로 파라미터(paramete)라고도 부릅니다.

```
IN 입력_매개변수_이름 데이터_형식
CALL 프로시저_이름(전달_값);
```

스토어드 프로시저에서 처리된 결과를 출력 매개변수를 통해 얻을 수도 있다.

```
OUT 출력_매개변수_이름 데이터_형식
```

출력 매개변수에 값을 대입하기 위해서는 주로 SELECT~INTO 문을 사용한다.

```
CALL 프로시저_이름(@변수명);
SELECT @변수명;
```

- 입력 매개변수 : 스토어드 프로시저에 값을 전달한다. 형식은 IN을 앞에 붙인다.
- 출력 매개변수 : 스토어드 프로시저에서 계산된 결과를 돌려받는다. 형식 out을 앞에 붙인다.
- 동적 SQL : 다이나믹하게 SQL을 생성한 후 실행한다. PREPARE 문과 EXECUTE 문을 사용한다.

07-2 스토어드 함수와 커서

스토어드 함수는 MySQL에서 제공하는 내장 함수 외에 직접 함수를 만드는 기능을 제공한다. 즉, MySQL이 제공하는 함수를 그대로 사용할 수 없는 경우가 발생한다면 직접 스토어드 함수를 작성해서 사용할 수 있다.

스토어드 함수와 커서를 활용하면 편리하고 강력한 SQL을 활용할 수 있다.

스토어드 함수

- 스토어드 함수의 개념과 형식 : MySQL은 다양한 내장 함수로 제공되지 않는 기능을 스토어드 함수로 만들어서 사용할 수 있다.

스토어드 함수는 RETURNS 문으로 반환할 값의 데이터 형식을 지정하고, 본문 안에서는 RETURN 문으로 하나의 값을 반환해야 한다.

스토어드 함수의 매개변수는 모두 입력 매개변수이다. 그리고 IN을 붙이지 않는다.

스토어드 프로시저는 CALL로 호출하지만, 스토어드 함수는 SELECT 문 안에서 호출된다.

스토어드 프로시저 안에서는 SELECT 문을 사용할 수 있지만, 스토어드 함수 안에서는 SELECT를 사용할 수 없다.

스토어드 프로시저는 여러 SQL 문이나 숫자 계산 등의 다양한 용도로 사용하지만, 스토어드 함수는 어떤 계산을 통해서 하나의 값을 반환하는데 주로 사용한다.

- 스토어드 함수의 사용 : 스토어드 함수를 사용하기 위해서 먼저 다음 SQL로 스토어드 함수 생성 권한을 허용해줘야 한다.

함수의 반환 값을 SELECT~INTO~로 저장했다가 사용할 수도 있다. 함수의 반환 값을 각 변수에 저장한 후, 그 차이를 계산해서 출력한다.

커서로 한 행씩 처리하기

- 커서의 기본 개념 : 테이블에서 한 행씩 처리하기 위한 방식이다. 첫 번째 행을 처리한 후에 마지막 행까지 한 행씩 접근해서 값을 처리한다. 처음에는 커서가 행의 시작을 가리킨 후에 한 행씩 차례대로 접근한다.

커서 선언하기 -> 반복 조건 선언하기 -> 커서 열기 -> 데이터 가져오기 -> 데이터 처리하기 -> 커서 닫기

- 커서의 단계별 실습

1. 사용할 변수 준비하기 : DEFAULT 문을 사용해서 초기값을 0으로 설정한다. 추가로 행의 끝을 파악하기 위한 변수 endOfRow를 준비한다. 처음에는 당연히 행의 끝이 아닐 테니 FALSE로 초기화한다.

2. 커서 선언하기 : 커서를 선언한다.

3. 반복 조건 선언하기 : 행의 끝에 다다르면 앞에서 선언한 endOfRow 변수를 TRUE로 설정한다. DECLARE CONTINUE HANDLER는 반복 조건을 준비하는 예약어이다. 그리고 FOR NOT FOUND는 더 이상 행이 없을 때 이어진 문장을 수행한다. 즉, 행이 끝나면 endOfRow에 TRUE를 대입한다.

4. 커서 열기 : 앞에서 준비한 커서를 간단히 OPEN으로 열면 된다.

5. 행 반복하기 : 커서의 끝까지 한 행씩 접근해서 반복할 수 있다.

```
cursor_loop: LOOP
        이 부분을 반복
END LOOP cursor_loop
```

cursor_loop는 반복할 부분의 이름을 지정한 것이다. 그리고 코드는 무한 반복하기 때문에 코드 안에 반복문을 빠져나갈 조건이 필요하다. 앞에서 행의 끝에 다다르면 endOfRow를 TRUE로 변경하기로 설정했다.

LEAVE는 반복할 이름을 빠져나간다. 결국 행의 끝에 다다르면 반복 조건을 선언한 것에 의해서 endOfRow가 TRUE로 변경되고 반복하는 부분을 빠져나가게 된다.

FETCH는 한 행씩 읽어온다. 커서는 행의 끝에 다다르면 반복을 종료하도록 구성해야 한다.

6. 커서 닫기 : 모든 작업이 끝났으면 커서를 닫는다.

07-3 자동 실행되는 트리거

트리거(trigger)는 자동으로 수행해 사용자가 추가 작업을 잊어버리는 실수를 방지해준다. 트리거를 활용하면 데이터가 삭제될 때 해당 데이터를 다른 곳에 자동으로 백업할 수 있다.

트리거

- 트리거의 개요 : 트리거란 테이블에 INSERT나 UPDATE 또는 DELETE 작업이 발생하면 실행되는 코드이다.

- 트리거의 기본 작동 : 테이블에서 DML(Data Manipulation Language)문 (INSERT, UPDATE, DELETE 등)의 이벤트가 발생할 때 작동한다. 테이블에 미리 부착되는 프로그램 코드라고 생각하면 된다.

트리거는 스토어드 프로시저와 문법이 비슷하지만, CALL 문으로 직접 실행시킬 수는 없고 오직 테이블에 INSERT, UPDATE, DELETE 등의 이벤트가 발생할 경우에만 자동으로 실행된다. 또한 스토어드 프로시저와 달리 트리거에는 IN, OUT 매개변수를 사용할 수 없다.

트리거 활용

- 트리거는 테이블에 입력/수정/삭제되는 정보를 백업하는 용도로 활용할 수 있다.

- 테이블에 이벤트가 먼저 적용된 후에 트리거가 실행된다.

- 트리거에서 기존 테이터는 OLD 테이블에, 새로운 데이터는 NEW 테이블에 잠깐 저장된다. OLE 및 NEW 테이블은 MySQL이 내부적으로 관리한다.

*** 트리거가 사용하는 임시 테이블**

NEW 테이블 : INSERT(새 값) -> NEW 테이블(새 값) -> 테이블(새 값)

INSERT(새 값) 형태로 테이블에 새값이 바로 들어간다. 하지만 사실 새 값은 테이블에 들어가기 전에 NEW 테이블에 잠깐 들어가 있다. NEW 테이블은 많이 사용되지 않는다. 어차피 NEW 테이블에 들어간 값은 테이블에 들어가 있기 때문이다.

OLE 테이블 : DELETE(예전 값) -> 테이블(~~예전 값~~) -> OLD 테이블(예전 값)

삭제될 예전 값이 삭제되기 전에 OLD 테이블에 잠깐 들어가 있다. AFTER DELETE 트리거를 만들어도 삭제된 후에 OLD.열 이름 형식으로 예전 값에 접근할 수 있었다.

UPDATE(새 값, 예전 값) -> NEW 테이블(새 값) -> 테이블(~~예전 값~~ 새값) -> OLD 테이블(예전 값)

Chapter 08. SQL과 파이썬 연결

08-1 파이썬 개발 환경 준비

Python <-> MySQL : 파이썬을 MySQL과 연동하기 위해서는 PyMySQL 라이브러리가 필요하다.

파이썬 소개

- 파이썬(python) : 귀도 반 로섬이라는 프로그래머가 C언어를 기반으로 제작
- 다른 프로그래밍 언어에 비해 좀 더 직관적인 문법으로 배우기 쉽다는 장점을 갖고 있다.
- 무료로 강력한 기능을 사용할 수 있다.
- 설치와 사용 환경 구축이 쉽다.
- 다양하고 강력한 외부 라이브러리들이 많다.

파이썬 설치

- 파이썬 다운로드하기 : https://www.python.org에 접속하고 [Downloads] 한다.
- 파이썬 설치하기 : 다운로드한 python-3.x.x-amd64.exe 파일을 더블 클릭하고 초기 화면에서 제일 하단의 'Add Python 3.x to PATH'를 체크한 후 [Install Now]를 클릭한다. 설치가 완료되면 [Disable path length limit]을 클릭한 후 [Close] 버튼을 클릭한다.
- 외부 라이브러리 설치하기 : Windows+R을 누른 후에 실행 창에서 cmd를 입력하고 [확인] 버튼을 클릭한다. 명령 프롬프트가 나타나면 다음 명령어를 실행해서 pymysql을 설치한다. 'Successfully installed pymysql-x.x.x'와 같은 메시지가 나오면 성공이다.

```
pip install pymysql
exit
```

파이썬 사용 방법

- 대화형 모드 : 한 줄씩 실행하기
- 스크립트 모드 : 여러 줄을 한 번에 실행하기

08-2 파이썬과 MySQL의 연동

파이썬과 pymysql 라이브러리를 설치한 후에는 MySQL과 연동하는 데이터베이스 연동 프로그램을 작성할 수 있다.

연동 프로그래밍 기본

- 연동 프로그램을 위한 쇼핑몰 생성 : MySQL 워크벤치를 실행해서 DB를 생성한다.
- 파이썬에서 데이터 입력 : MySQL 연결하기(연결자 = pymysql.connect(연결 옵션)) -> 커서 생성하기(커서이름 = 연결자.cursor()) -> 테이블 만들기(커서이름.execute("CREATE TABLE 문장")) -> 데이터 입력하기(커서이름.execute("INSERT 문장")) -> 입력한 데이터 저장하기(연결자.commit()) -> MySQL 연결 종료하기(연결자.close())

연동 프로그래밍 활용

- MySQL의 데이터 조회를 위한 파이썬 코딩 순서 : MySQL 연결하기(연결자 = pymysql.connect(연결 옵션)) -> 커서 생성하기(커서이름 = 연결자.cursor()) -> 데이터 조회하기(커서이름.execute("SELECT 문장")) -> 조회한 데이터 추출하기(커서이름.fetchone()) -> MySQL 연결 종료하기(연결자.close())
- fetchone()은 한 행씩 접근하며, fetchall()은 모든 행에 한꺼번에 접근한다.

08-3 GUI 응용 프로그램

GUI(Graphical User Interface)는 윈도우에 그래픽 환경으로 제공되는 화면을 통틀어서 말한다. 파이썬을 통해 윈도에 출력되는 GUI 응용 프로그램을 작성하는 것을 도와주는 라이브러리는 tkinter다.

GUI 기본 프로그래밍

- 기본 윈도의 구성

```
from tkinter import *
root = Tk()
```

# 앞으로 이 부분에 코딩을 추가해서 화면을 구성하고 처리한다.
root.mainloop()

- from tkinter import * 문을 처음에 입력하면 위도 및 관련 내용을 사용할 수 있다.
- 라벨(label) : 문자를 표현할 수 있는 위젯으로, label(부모윈도, 옵션…) 형식을 사용한다. 윗젯(widget)은 윈도에 나오는 버튼, 텍스트, 라디오 버튼, 이미지 등을 통합해서 지칭하는 용어이다. 모든 위젯들은 pack() 함수를 사용해야 화면에 나타난다.
- 버튼(button) : 마우스로 클릭하면 지정한 작업이 실행되도록 사용되는 위젯으로, Button(qnah윈도, 옵션…) 형식을 사용한다. command 옵션으로 사용자가 버튼을 눌렀을 때 지정한 작업을 처리해야 한다.
- 위젯의 정렬 : pack() 함수의 옵션 중에서 가로로 정렬하는 방법으로 side=LEFT 또는 RIGHT 방식이 있다. TOP 또는 BOTTOM 방식을 사용하면 수직으로 정렬할 수 있다.
- 위젯 사이에 여백 추가 : pack() 함수의 옵션 중 padx=픽셀값 또는 pady=픽셀값 방식을 사용한다.
- 프레임, 엔트리, 리스트 박스 : 프레임(frame)은 화면을 여러 구역으로 나눌 때 사용, 엔트리(entry)는 입력 상자를 표현, 리스트 박스(list box)는 목록을 표현한다.

*** GUI란?**

GUI는 사용자가 편리하게 사용할 수 있도록 텍스트가 아닌 그래픽 환경을 제공한다. 간단히는 윈도 화면이 나오는 환경이라고 생각하면 된다.

Appendix. MySQL 연동을 위한 파이썬 필수 문법

01 변수와 print()

- 변수의 선언 : 파이썬은 별도의 변수 선언이 없다. 변수에 값을 대입하는 순간 변수가 선언된다.

변수의 종류를 확인하기 위해서는 type() 함수를 사용한다.

- print() 함수 : 서식을 지원하는 함수이다. 모니터에 큰따옴표 또는 작은따옴표 안의 내용을 출력한다는 의미이다.

서식 앞에는 %가 붙는데, %d는 정수를 의미한다. 또한 서식의 개수와 따옴표 뒤에 나오는 숫자(또는 문자)의개수는 동일해야 한다.

서식	값의 예	설명
%d, %x, %o	10, 100, 1234	정수(10진수, 16진수, 8진수)
%f	0.5, 1.0, 3.14, 3.14e3	실수(소수점이 붙은 수)
%c	"b", "한", '파'	문자 한 글자
%s	"안녕", 'abcdefg', "a"	한 글자 이상의 문자열

02 연산자

- 산술 연산자

산술 연산자	설명	사용 예	예 설명
=	대입 연산자	a = 3	정수 3을 a에 대입
+	더하기	a = 5 + 3	5와 3을 더한 값을 a에 대입
-	빼기	a = 5 - 3	5에서 3을 뺀 값을 a에 대입
*	곱하기	a = 5 * 3	5와 3을 곱한 값을 a에 대입
/	나누기	a = 5 / 3	5를 3으로 나눈 값을 a에 대입
//	나누기(몫)	a = 5 // 3	5를 3으로 나눈 뒤 소수점을 버리고 a에 대입
%	나머지 값	a = 5 % 3	5를 3으로 나눈 뒤 나머지 값을 a에 대입
**	제곱	a = 5 ** 3	5의 3제곱을 a에 대입

- 대입 연산자

연산자	설명	사용 예	예 설명
+=	대입 연산자	a += 3	a = a + 3과 동일
-=	대입 연산자	a -= 3	a = a - 3과 동일
*=	대입 연산자	a *= 3	a = a * 3과 동일
/=	대입 연산자	a /= 3	a = a / 3과 동일
//=	대입 연산자	a //=3	a = a // 3과 동일
%=	대입 연산자	a %= 3	a = a % 3과 동일
**=	대입 연산자	a **=3	a = a ** 3과 동일

파이썬에는 C, C++, 자바 등의 언어에 있는 증가 연산자 ++이나 감소 연산자 —는 없습니다.

- 관계 연산자 : 관계 연산자(또는 비교 연산자)는 어떤 것이 큰지, 작은지, 같은지를 비교하는 것으로 그 결과는 참(True)이나 거짓(False)이 된다. 그러므로 주로 조건문(if)이나 반복문(while)에서 사용되며, 단독으로 사용되는 경우는 별로 없다.

관계 연산자	의미	설명
==	같다	두 값이 동일하면 참
!=	같지 않다	두 값이 다르면 참
>	크다	왼쪽이 크면 참
<	작다	왼쪽이 작으면 참
>=	크거나 같다	왼쪽이 크거나 같으면 참
<=	작거나 같다	왼쪽이 작거나 같으면 참

- 문자열과 숫자의 상호 변환 : 문자열이 숫자로 구성되어 있을 때, int() 또는 float() 함수를 이용해서 정수나 실수로 변환할 수 있다. 파이썬은 정수 크기의 제한이 없다.

03 조건문

- 기본 if문 : 참일 때는 실행하고, 거짓일 때는 아무것도 실행하지 않는 가장 단순한 if문의 형태이다. 조건식이 참이라면 실행할 문장을 실행하고, 조건식이 거짓이라면 실행할 것이 없다.

```
if 조건식:
        실행할 문장
```

- if ~ else문 : 참일 때 실행하는 것과 거짓일 때 실행하는 것이 다를 때는 if ~ else문을 사용한다. 조건식이 참이라면 실행할 문자1을 실행하고, 그렇지 않으면 실행할 문장2를 실행한다.

```
if 조건식:
        실행할 문장1
else:
        실행할 문장2
```

04 반복문

- for문의 개념과 작동

```
for 변수 in range(시작 값, 끝값+1, 증가값) :
        이 부분을 반복
```

range() 함수는 지정된 범위의 값을 반환한다. 증가 값은 생략할 경우 1로 인식한다.

- 무한 루프를 위한 while 문 : while 문에 무한 루프(무한 반복)를 사용할 수 있는데, 무한 루프를 적용하려면 조건식을 True로 지정한다.

```
While True:
        반복할 문장들
```

- 반복문을 탈출하는 break 문 : for 문의 range() 함수를 이용하면 지정한 범위에서 벗어나면 for 문이 종료된다. while 문은 조건식이 False가 되면 while 문을 종료하거나, 무한 반복의 경우 Ctrl+C를 누르면 프로그램이 종료된다. 하지만, 계속되는 반복을 논리적으로 빠져나가는 방법이 break 문이다. 반복문 안에서 break 문을 만나면 무조건 반복문을 빠져나온다.

05 리스트

- 리스트의 개념 : 하나씩 사용하던 변수를 한 줄로 붙여 놓은 것이다. 다른 언어의 배열과 비슷한 개념이다.
- 리스트의 생성과 사용 방법

```
리스트_이름 = [값1, 값2, 값3, …]
```

- 빈 리스트와 리스트 추가 : 리스트_이름.append(값)
- 여러 리스트를 동시에 순회하는 zip() 함수
- 리스트 조작 함수

함수	설명	사용법
append()	리스트 제일 뒤에 항목을 추가한다.	리스트이름.append(값)
pop()	리스트 제일 뒤의 항목을 빼내고, 빼낸 항목은 삭제한다.	리스트이름.pop()
sort()	리스트의 항목을 정렬한다.	리스트이름.sort()
reverse()	리스트 항목의 순서를 역순으로 만든다.	리스트이름.reverse()
index()	지정한 값을 찾아서 그 위치를 반환한다.	리스트이름.index(찾을 값)
insert()	지정된 위치에 값을 삽입한다.	리스트이름.insert(위치, 값)
remove()	리스트에서 지정한 값을 제거한다. 단, 지정한 값이 여러 개일 경우 첫 번째 값만 지운다.	리스트이름.remove(지울 값)
extend()	리스트의 뒤에 리스트를 추가한다. 리스트의 더하기(+) 연산과 동일한 기능을 한다.	리스트이름.extend(추가할 리스트)
count()	리스트에 해당 값의 개수를 센다.	리스트이름.count(찾을 값)
clear()	리스트의 내용을 모두 제거한다.	리스트이름.clesr()

del()	리스트에서 해당 위치의 항목을 삭제한다.	del(리스트이름[위치])
len()	리스트에 포함된 전체 항목의 개수를 센다.	len(리스트리므)
copy()	리스트의 내용을 새로운 리스트에 복사한다.	새리스트=리스트이름.copy()
sorted()	리스트의 항목을 정렬해서 새로운 리스트에 대입한다.	새리스트=stored(리스트)

정확히는 함수와 메소드를 구분해야 하지만, 객체지향에 대한 언급을 하지 않기에 모두 함수로 통일해서 부름.

06 문자열

- 문자열 기본 : 리스트를 접근하는 것과 문자열을 접근하는 것은 크게 다르지 않습니다. 리스트는 대괄호로 묶어서 나오고 문자열은 작은따옴표로 묶여서 출력되는 차이점만 있다.

문자열도 리스트와 마찬가지로 연결하는 것은 더하기를 사용하고, 곱하기는 문자열을 반복해준다.

문자열의 길이를 파악할 때도 리스트와 같이 *len()* 함수를 사용한다. 개수를 파악할 수 있기 때문에, 리스트처럼 *for*문을 사용해서 처리하는 것이 가능하다.

- 문자열 찾기 함수

count('찾을 문자열') : 찾을 문자열이 몇 개 들었는지 개수를 센다.

find('찾을 문자열') : 찾을 문자열이 왼쪽부터 몇 번째 위치하는지 찾는다.

rfind('찾을 문자열') : 오른쪽부터 센다.

find('찾을 문자열' , 시작 위치) : 시작 위치부터 문자열을 찾는데, *find()* 함수는 찾을 문자열이 없으면 *-1*을 반환한다.

- 문자열 분리, 결합

split() : 문자열을 공백이나 다른 문자로 분리해서 리스트를 반환한다.

join() : 문자열을 합쳐준다.

07 함수

- 함수의 개념 : 함수(*function*)란 무엇을 넣으면 어떤 것을 돌려주는 블랙박스이다. 파이썬 프로그램 자체에서도 제공하지만, 사용자가 직접 만들어서 사용하기도 한다.

함수이름()

- 함수의 모양과 활용 : 함수는 매개변수를 입력받은 후, 그 매개변수를 가공 및 처리한 후에 반환 값을 돌려준다.

- 파이썬을 별도의 변수 선언이나 메인 함수가 없기 때문에 코드가 길어지면 가독성이 떨어진다. 즉, 함수 선언부, 전역 변수부, 메인 코드부로 분할해서 코드를 작성하면 코드를 읽기가 수월해지는 효과가 있다.

- 지역 변수와 전역 변수의 이해

지역 변수란 말 그대로 한정된 지역에서만 사용되는 변수이다.

전역 변수는 프로그램 전체에서 사용되는 변수이다.

※ 한빛미디어 '혼자 공부하는 SQL' 의 내용을 바탕으로 정리한 내용입니다.