📕

[AI초급] SQLD 자격증 코스 - 챕터 7

매 주차 강의자료 시작에 PDF파일을 올려두었어요!

▼ PDF 파일

[수업 목표]

- SQL에 대해 알아보고 기본기를 학습합니다.

[목차]

💡 모든 토글을 열고 닫는 단축키
Windows : `Ctrl` + `alt` + `t`
Mac : `⌘` + `⌥` + `t`

---

01. 관계형 데이터베이스 개요

✔️ **관계형 데이터베이스를 통해 가능한 것들에 대해 학습합니다**

▼ 1) 데이터베이스란?

☑️ 데이터베이스 (Database)

데이터베이스는 통상 넓은 의미로 '정보의 모음'을 의미합니다. 핸드폰이 없던 시절 우리는 어떤 방식으로 전화번호를 기억했을까요? 전화번호를 노트에 하나하나 적어두고 필요할 때마다 다시 확인했습니다. '전화번호'라는 정보를 한곳에 모아서 관리한다고 볼 수 있습니다. 다들 학창 시절 알림장을 썼던 기억이 있으실 텐데요, 필요한 준비물을 까먹지 않기 위해 필요한 정보를 저장하는 용도로 사용했습니다. 알림장도 한 곳에 정보를 모아서 관리한다는 측면에서 데이터베이스로 볼 수 있습니다.

이번에는 데이터베이스의 의미를 일상에서 보다 업무적인 영역으로 옮겨와서 생각해 보겠습니다. 온라인 플랫폼에서 물건을 구매하는 행위는 판매자의 데이터베이스에 접근하여 특정한 데이터를 저장하는 것이라고 볼 수 있습니다. 만약 구매한 물건의 목록을 확인한다면 저장된 데이터를 조회하는 행위라고 볼 수 있겠죠. 온라인 서비스에 로그인하거나 은행 계좌에서 돈을 인출하는 행위 모두 본질적으로는 같은 방식으로 데이터베이스와 소통하는 과정입니다. 결과적으로 일상에서 하는 수많은 행위는 정보로 저장되고 해당 정보를 여러 가지 방식으로 활용하는 과정은 모두 데이터베이스와 소통한다고 볼 수 있습니다. 매 순간 만들어지는 정보를 그저 쌓아두기만 하는 것이 아니라 효율적으로 관리하는 것이 필요합니다. 그러면 여기서 말하는 '효율적인 관리'란 어떤 것을 의미하는지 알아보도록 하겠습니다.

❕ **참고 - 데이터베이스 용어의 유래**

데이터베이스라는 용어는 미국에서 군비 관리를 효율적으로 하기 위해 도서관처럼 데이터를 모아두고 관리한다는 뜻으로 '데이터의 기지'라는 의미를 담아 만들어졌습니다.

☑️ 관계형데이터 베이스 (Relational Database)

현재 사용하는 보편적인 데이터베이스가 등장하기 이전에는 파일 시스템을 활용하여 데이터를 관리했습니다. 파일 시스템은 데이터를 파일 단위로 관리하기 위해 파일을 생성, 삭제, 수정, 조회할 수 있는 기능을 제공해 주는 소프트웨어를 의미합니다. 파일 1개가 수많은 데이터를 저장할 수 있는 구조인데, 이러한 데이터의 관리에는 몇 가지 문제점이 있었습니다.

우선 구조화되지 않고 데이터를 일방적으로 저장하기 때문에 동일한 내용의 정보가 여러 파일에 반복적으로 저장되는 문제가 있습니다. 중복되는 정보로 인해 불필요한 저장 공간을 사용해야 했고 원하는 데이터를 어디서 찾아야 하는지 알 수 없어 비효율적입니다. 해당 파일을 사용하는 프로그램에 종속적이라 프로그램이 바뀌면 정보를 저장하는 데이터의 형식도 바꿔야 하는 불편함이 있습니다. 나아가 저장해야 하는 데이터의 형식 등을 통제할 수 없어 축적된 데이터를 원하는 방향으로 사용하기 위해서는 많은 비용을 들여 처리하는 과정이 필요합니다. 결론적으로 파일 시스템을 통한 정보의 관리는 중복되는 데이터를 관리하기 어렵고 일관성 있는 데이터의 축적에 큰 한계가 있습니다.

여러 문제점을 고려하는 과정에서 데이터를 조금 더 체계적으로 정의하고 조작하는 방법을 고민하게 되었고 IBM사에 의해 관계형 데이터베이스가 등장했습니다. 관계형 데이터 베이스는 데이터베이스를 체계적으로 관리하고 운영하는 소프트웨어입니다. 사용자가 정해진 방식으로 요청하면 행과 열의 2차원 관계로 정의된 데이터의 **생성, 수정, 삭제, 그리고 조회**가 가능합니다. 그럼 구체적으로 어떤 방식으로 데이터베이스 관리 시스템의 주요 기능을 활용할 수 있는지 알아보겠습니다.

☑️ 관계형 데이터베이스 관리 시스템

- 관계형 데이터베이스 구조에 기반한 관계형 데이터베이스 관리 시스템 (Relational Database Management System, RDBMS)는 메타 데이터를 총관 관리할 수 있기 때문에 데이터의 성격, 속성 또는 표현 방법 등을 체계화할 수 있습니다. 또한 데이터 표준화를 통해 데이터 품질을 확보할 수 있는 장점을 가지고 있습니다.
- RDBMS는 인증된 사용자만이 참조할 수 있도록 보안 기능을 제공합니다. 사용자가 실수로 조건에 위배되는 행동을 할 경우 이를 방지하여 데이터 무결성(Intergrity)을 보장합니다.
- RDBMS는 시스템의 갑작스러운 장애로부터 사용자가 입력, 수정, 삭제하던 데이터가 제대로 반영될 수 있도록 보장해주는 기능과 시스템 다운, 재해 등의 상황에서도 데이터를 회복/복구할 수 있는 기능을 제공합니다.

▼ 2) SQL

지금까지 왜 데이터베이스를 사용해야 하는지, 나아가 관계형 데이터베이스를 통해 어떤 것이 가능해졌는지 알게 되었습니다. 이 시점에서 한 가지 궁금한 점이 생길 수 있습니다. 데이터를 보관하는 방법이 어떤 형태로 구성되어 있는지는 알겠는데, 새로운 정보를 데이터베이스에 넣고 수정하고 삭제하는 등의 행위는 어떤 과정을 통해 가능할까요? 이런 일을 수행하기 위해 고안된 언어가 바로 SQL입니다.

SQL은 Structured Query Language의 약자로 관계형 데이터베이스와 통신할 때 사용하는 구조화된 질의 언어를 의미합니다. 여기서 '질의(query)'라고 하는 것은 '질문'을 떠올리시면 됩니다. 궁금한 점을 누군가에게 질문하듯 SQL이라는 언어를 사용하여 질문하면 그에 대한 응답으로 어떠한 결과를 받을 수 있습니다. 질문은 데이터를 생성, 수정, 삭제, 그리고 조회하는 문장으로 구성되어 있으며 특정 권한을 부여받은 사람만 이러한 행동을 할 수 있도록 제어할 수도 있습니다. 다음과 같은 형태로 분류 가능합니다.

☑️ SQL 언어 분류

데이터 정의 언어(DDL: Data Definition Language)

> 관계형 데이터베이스의 구조를 정의하는 데 사용하는 언어입니다.

- 데이터를 저장하기 위한 데이터베이스의 기본 구조를 잡는데 사용하는 언어입니다.
- 데이터를 저장하는 테이블을 만들거나 수정할 수 있습니다.
- 테이블을 생성(CREATE), 변경(ALTER, RENAME), 삭제(DROP) 할 수 있습니다.

데이터 조작 언어(DML: Data Manipulation Language)

> 테이블에 저장된 데이터를 조작하는 데 사용하는 언어입니다.

- 데이터 정의 언어로 생성된 구조를 실제 데이터로 채우는데 사용하는 언어입니다.
- 데이터를 삽입(INSERT), 조회(SELECT), 수정(UPDATE), 삭제(DELETE) 할 수 있습니다.

데이터 제어 언어(DCL: Data Control Language)

- 데이터베이스에 접근하여 이를 제어하기 위한 권한을 관리하는 데 사용하는 언어입니다.
- 특정한 사용자의 데이터베이스 사용 권한을 부여하거나 박탈할 수 있습니다.
- 데이터베이스 접근하고 객체를 사용하도록 권한을 부여(GRANT)하거나 회수(REVOKE)할 수 있습니다.

트랜잭션 제어어(TCL: Transaction Control Language)

- 논리적인 작업의 단위를 묶어서 DML에 의해 조작된 결과를 작업단위(트랜잭션)별로 제어하는 언어입니다.
- 논리적인 작업의 단위를 묶어서 DML에 의해 조작된 결과를 작업단위(트랜잭션) 별로 제어할 수 있습니다. 이때 COMMIT, ROLLBACK 등의 명령어를 수행할 수 있습니다.
- 트랜잭션은 다음과 같은 특징을 갖습니다. (ACID)
  - **원자성(Atomicity)**
    - 하나의 논리적인 단위로 묶여있는 트랜잭션이 데이터베이스에 모두 적용되거나 아니면 적용되지 않거나 둘 중 하나의 선택지만 있어야 합니다.
  - **일관성(Consistency)**
    - 트랜잭션의 작업 처리 결과는 항상 일관되어야 합니다.
  - **독립성(Isolation)**
    - 둘 이상의 트랜잭션이 동시에 진행되고 있는 상황에서 하나의 트랜잭션의 결과가 다른 트랜잭션의 결과에 영향을 줄 수 없습니다.
  - **지속성(Durability)**
    - 트랜잭션을 통해 특정한 작업이 마무리된 경우 결과는 데이터베이스에 영구적으로 반영되어야 합니다.

SQL은 특정한 규칙에 맞춰서 구문을 작성해야 하는 자체 표준이 존재합니다. 다만, 관계형 데이터베이스를 제공하는 회사마다 특징이 있기 때문에 완전하게 통일된 형태로 제공하지 않습니다. 따라서 데이터베이스를 제공하는 회사에 따라 다른 처리가 필요하다는 점을 숙지하고 있어야 합니다.

SQL Syntax

모든 SQL 문(statement)는 SELECT, INSERT, UPDATE 등과 같은 키워드로 시작하고,
하나의 statement는 세미콜론(;)으로 끝납니다.

```
SELECT col_name FROM table_name;
```

- 세미콜론(;)은 각각의 SQL 문을 구분하는 표준 방법
- SQL 키워드는 대소문자를 구분하지 않습니다.
  - `SELECT` == `select`
  - 하지만 대문자로 작성하는것을 권장합니다 🙂

    → 그냥 외우기보다 익숙해지는 것이 필요!

SQL 문(Statement) & 절(Clause)

- 문(Statement)
  - 독립적으로 실행할 수 있는 완전한 SQL 코드 조각
  - 즉, 문(Statement)은 절(Clause)로 구성됨
- 절(Clause)
  - 문(Statement)의 하위 단위

```
SELECT col_name FROM table_name;
```

→ SELECT 문이라고도 불러요

→ 위 문(Statement)은 두개의 절(Clause)로 구성되어 있어요

- `SELECT col_name`
- `FROM table_name`

▼ 3) 테이블

관계형 데이터베이스는 데이터베이스의 기본 단위인 테이블의 형태로 데이터를 저장합니다. '테이블'이라는 말이 다소 생소하지만 우리가 자주 사용하는 MS사의 Excel 프로그램을 떠올리면 비교적 쉽게 이해할 수 있습니다. Excel에서 제공하는 시트를 데이터베이스에서 하나의 테이블에 대응시켜 생각해 볼 수 있습니다. 시트는 행과 열로 구성되어 있으며 열은 어떤 정보가 들어갈지 정의하고 행은 실제 데이터가 담기는 구조입니다. 이는 관계형 데이터베이스 테이블의 구조와 동일하며 행과 열의 2차원의 형태로 어떤 정보를 어떻게 담을지 통해 구조화 시킨 것이라고 할 수 있습니다. 정리하면 테이블은 관계형 데이터베이스에서 데이터를 저장하는 기본 단위이며 관계형 데이터베이스에서 모든 데이터는 칼럼과 행의 2차원 구조로 표현할 수 있습니다. 세로 방향을 칼럼(Column), 가로 방향을 행(Row)이라고 부르고 이들이 교차하는 하나의 공간을 필드(Field)라고 부릅니다.

종류	설명
테이블(Table)	행과 열로 구성된 2차원의 데이터 저장 구조
column/열/속성	테이블에 저장되는 데이터의 특징 더 이상 나눌 수 없는 하나하나의 속성
row/행/레코드	테이블에 저장되는 실제 데이터 (인스턴스)

2차원 관계형 테이블을 통해서 어떠한 데이터를 저장한다고 하더라도 그 데이터가 완전한 형태로 저장되었다고 확신할 수는 없습니다. 저장 과정에서 중복된 정보가 들어갈 수도 있고 불필요한 칼럼이 존재하여 저장 공간의 낭비가 발생할 수 있습니다. 이러한 문제점 등을 해결하여 데이터의 불필요한 중복을 줄이는 것을 정규화(Normalization)라고 합니다.

우리가 사는 세상은 하나의 테이블로 정보를 저장하여 표현하기 어려운 경우가 많습니다. 예를 들면, '학교'라는 대상을 표현하기 위해 필요한 정보를 생각해 봅시다. 학생 정보가 담긴 테이블 선생님 정보가 담긴 테이블이 필요합니다. 그렇다면 이 두 대상의 정보만 있다면 온전하게 학교를 표현할 수 있을까요? 학생과 선생님뿐만 아니라 교실, 운동장, 등과 같은 사람이 아닌 대상도 존재하기 때문에 위의 테이블 정보만 가지고는 이러한 정보를 표현하기는 어렵습니다. 따라서 대상에 따라 각기 다른 정보를 저장하기 위한 테이블을 여러 개 구성합니다.

테이블은 어떠한 관계를 가질 수 있습니다. 예시를 하나 들어보겠습니다. 게시판 서비스를 많이 활용해 보셨을 텐데, 게시글에는 보통 댓글이 있습니다. 게시글은 게시글을 표현하기 위한 정보가 있고 댓글을 표현하기 위한 정보가 존재합니다. 그럼 각자 이런 정보를 가지고 있는 것만으로 두 정보의 관계가 '연결'되어 있다고 볼 수 있을까요? 아닙니다. 단순하게 보면 그저 정보가 나열되어 있는 것이라고 볼 수 있습니다. 관계 설정을 위해서는 몇 가지 추가적인 작업이 필요합니다.

우선 특정한 정보를 담고 있는 테이블에서 특정한 row (데이터)를 구분할 수 있는 구분자가 필요합니다. 테이블에 있는 특정한 데이터에서는 그래야 다른 정보를 담고 있는 테이블에서 특정 정보를 참고할 때 이를 유일하게 식별할 수 있는 어떤 값을 찾아낼 수 있습니다. 대한민국 사람은 모두 주민등록번호라는 고유한 값을 통해 구분될 수 있는 것을 생각해 보면 쉽게 이해할 수 있습니다. 이렇게 특정한 데이터를 유일하게 구분하는 식별자를 기본키(Primary Key)라고 합니다. 기본키(Primary Key)는 테이블에 존재하는 하나의 행(row)을 한 가지 의미로 특정할 수 있는 한 개 이상의 칼럼을 의미합니다. 하나의 테이블에서 중복되지 않으며 고유하게 식별할 수 있어야 합니다. 예를 들어, 게시글 테이블에서 게시글 번호는 행을 구분하는 고유하고 중복되지 않은 값이기 때문에 기본키라고 할 수 있습니다.

다음으로 이 식별 정보를 활용해서 연결 관계를 표현하는 과정이 필요합니다. 관계 표현을 위해서는 외부키(Foreign Key)에 대한 이해가 필요합니다. 외부키(Foreign Key)는 다른 테이블의 기본키(Primary Key)로 활용되고 있는 열과의 관계를 연결하는 칼럼을 의미합니다. 예를 들어, 게시글 테이블에서 게시글 번호는 댓글 테이블에서 해당 댓글이 어떤 게시글에 달린 댓글인지 구분하는 역할을 하기 때문에 외부키라고 할 수 있습니다. 결과적으로 댓글 테이블에 있는 게시글 번호 칼럼(외부키)을 통해 테이블 간 연결 정보를 표현할 수 있습니다.

기본키🔑
(Primary Key)

외부키🔐
(Foreign Key)

기본키🔑
(Primary Key)

댓글 번호	게시글 번호	내용	댓글 생성 시각	댓글 수정 시각
1				
2				
3				

< 댓글 테이블 >

게시글 번호	제목	내용	게시글 생성 시각	게시글 수정 시각
1				
2				
3				
4				
5				

< 게시글 테이블 >

기본키🔑
(Primary Key)

외부키🔐
(Foreign Key)

기본키🔑
(Primary Key)

게시글 번호	댓글번호	제목	내용
1			
2			
3			
4			
5			

< 게시글 테이블 >

댓글번호	내용	댓글생성 시각	댓글 수정 시각
1			
2			
3			

< 댓글 테이블 >

▼ 4) ERD

테이블 간 상관 관계를 이미지로 도식화한 것을 'Entity-Relationship Diagram'이라고 하며 간략하게 ERD로 줄여서 부릅니다. ERD는 **Entity**(개체), **Relationship**(관계), 그리고 **Attribute**(속성)으로 구성되어 있으며 현실 세계의 데이터는 이렇게 3가지 구성 요소로 모두 표현 가능합니다. SQL의 표준이 존재하기는 하지만 DBMS를 제공하는 회사에 따라 상세 내용은 상이한 것처럼 ERD도 이를 제공하는 회사에 따라 조금씩 다르게 표현합니다. 대표적으로 바커 표기법과 I/E 표기법으로 구분되는데, 이에 대해 하나씩 알아보겠습니다.

☑️ 바커 표기법(Baker Notation)

바커 표기법은 영국 컨설팅 회사 CACI에 의해 처음 개발되었고 리처드 바커에 의해 지속적으로 개선되어 왔습니다. 바커 표기법에서 **Entity(개체)**는 데이터베이스를 사용하는 주체가 지속적으로 보관하고 관리하는 대상을 의미합니다. 조금 더 넓은 의미로 표현하면 세상에 존재하는 무언가를 지칭합니다. 예를 들어, 사람, 회사의 사원, 학생, 친구 등이 될 수 있습니다. 그리고 데이터베이스 상에서 Entity가 의미를 갖기 위해서는 반드시 두 개 이상의 속성을 가져야 합니다. **Attribute(속성)는** 하나의 Entity에 종속되는 명사적 단어를 의미합니다. 일반적으로 명사적 단어 중에서 구성 요소를 포함하고 있는 명사들은 Entity가 되고 그렇지 않은 명사는 Attribute가 됩니다. 어떤 속성에 특정한 값을 반드시 저장해야 하는 경우는 `*`(Mandatory)을 표시해야 하며 값이 존재할 수도 있고 아닐 수도 있는 경우는 `o`(Optional)로 표시해야 합니다. **Relationship(관계)**은 두 개의 Entity 간의 조건 관계를 표기한 후 해당하는 Entity의 가까운 위치에 관계 명칭을 표기하고 관계(Relationship)는 실세계의 해당 Entity에서 발생하는 동사적 단어들을 표기합니다.

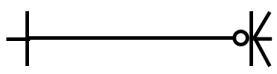
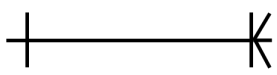
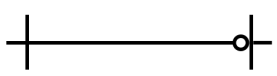
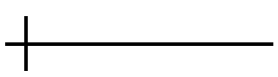
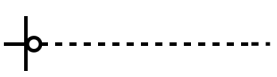
☑️ I/E 표기법(Information Engineering Notation)

Information Engineering(I/E)은 정보 시스템을 구축하는 데 있어 데이터 분석과 데이터베이스의 설계 과정에 매우 중요한 기법으로 자리 잡은 표기법입니다. 이 모델은 관계를 표현하는 데 있어 다수(many)에 해당하는 대상을 나타내기 위해 까마귀 발 모양을 사용하기 때문에 까마귀 발 모델(Crow's Foot Model)이라고 부르기도 합니다. 많은 문서에서 까마귀 발 표기법을 사용하기 때문에 정확하게 알아 두어야 합니다.

Entity(개체)는 바커 표기법과 마찬가지로 사용자가 추적하고자 하는 어떤 대상을 의미합니다.

Attribute(속성) 또한 바커 표기법과 마찬가지로 Entity의 특징을 기술해 주는 여러 개의 attribute를 가질 수 있습니다. 속성은 Entity 안에 위치해야 합니다.

Relationship(관계)는 까마귀 발 부호를 통해서 나타낼 수 있습니다. 관계의 다(Many)를 보여주는데 까마귀 발이 사용되며 타원(Oval), 해쉬 마크 등을 활용한 다양한 조합은 아래 표에 나온 것처럼 나타낼 수 있습니다.

ER win 부호	Identifying	Non-Identifying	의미
타원, 해쉬마크 및 까마귀 발			0,1 또는 그 이상의 개체 허용
까마귀 발이 있는 해쉬마크			1 또는 그 이상의 개체 허용
해쉬마크만 있는 타원			0 또는 1 개체 허용
해쉬마크			정확히 1 개체 허용

▼ Identifying (식별관계)

- 식별 관계의 경우 A 개체의 기본키가 B 개체의 외래키이면서 동시에 기본키가 되는 관계를 의미
- 비식별 관계의 경우 A 개체의 기본키가 B 개체의 비기본키 영역에서 외래키가 되는 관계를 의미

▼ 연습문제

☑️ 문제 1

Q. 문제	트랜잭션의 특징 4가지에 해당하지 않는 것은?
A. (1)	중복성
A. (2)	원자성
A. (3)	지속성
A. (4)	일관성

▼ 정답

(1) 트랜잭션의 특징 4가지는 원자성, 일관성, 독립성, 지속성이다.

☑️ 문제 2

Q. 문제	SQL언어의 종류와 명령어가 바르게 짝지어지지 않은 것은?
A. (1)	DML : SELECT

A. (2)	DDL : ALTER
A. (3)	DCL : RENAME
A. (4)	TCL : COMMIT

▼ 정답

(3) RENAME은 DDL의 명령어이다.

☑️ 문제 3

Q. 문제	테이블과 관련된 용어에 대한 설명으로 바르지 않은 것은?
A. (1)	정규화 : 데이터의 불필요한 중복을 줄이는 프로세스
A. (2)	테이블 : 2차원 구조의 데이터 저장 구조
A. (3)	칼럼 : 테이블에 저장되는 실제 데이터 인스턴스
A. (4)	기본키 : 특정한 데이터를 유일하게 구분하는 식별자

▼ 정답

(3) 칼럼은 테이블에서 세로 방향으로 이루어진 더 이상 나눌 수 없는 하나하나의 속성이다.

02. DDL

> ✔️ **SQL의 한 종류인 DDL에 대해 학습합니다**

▼ 1) DDL의 정의와 데이터 유형

☑️ DDL의 정의

DDL(Data Definition Language)는 SQL의 한 종류로 데이터베이스를 구성하는 다양한 요소를 정의하거나 변경하고 제거하는데 활용합니다. DDL은 주로 데이터를 저장하는 데이터베이스의 골격인 테이블을 다루는 언어로 구성되어 있습니다.

☑️ 데이터 유형

테이블의 구성 요소 중 가장 중요한 것은 칼럼을 정의하는 일입니다. 칼럼은 데이터베이스에 어떤 데이터를 어떤 방식으로 저장할 지 결정하기 때문에 테이블을 설계하는 단계에서 해야 하는 가장 기본적인 일입니다. 이에 대해 자세하게 알아보겠습니다.

1. 데이터 유형의 정의

- 데이터 유형

  데이터 유형은 데이터베이스에 데이터를 저장할 때 어떠한 데이터를 받을지 결정하는 기준을 의미합니다. 예를 들어, '학생'이라는 테이블에 '이름'이라는 칼럼이 있다고 하면 해당 칼럼은 어떤 데이터 타입을 갖게 될지를 설명해 준다고 생각하면 됩니다. 숫자, 문자열, 참/거짓 등의 자료형이 있으며 숫자 및 문자열 등도 데이터베이스를 제공하는 업체마다 세부적인 항목을 다르게 제공하는 경우가 있습니다. 만약 사전에 약속한 데이터와 다른 유형의 데이터를 저장하려고 하면 에러가 발생하거나 의도하지 않은 데이터가 저장될 수 있습니다.

- 데이터 크기

  데이터를 저장할 때는 데이터 크기(SIZE)도 지정할 수 있습니다. 특정 칼럼을 설정할 때 데이터 종류를 설정하면서 크기를 얼마만큼 배정할지 설정할 수 있습니다. 사전에 지정한 데이터의 크기를 넘어서는 자료가 입력되는 경우 에러가 발생하거나 의도하지 않은 데이터가 저장될 수 있습니다.

- NUMERIC Type

  숫자 형 타입은 DB를 제공하는 회사별로 다른 기준을 제공합니다. ISO/ANSI 기준으로는 숫자형 타입을 NUMERIC, DECIMAL, DEC, SMALLINT, INTEGER, INT, BIGINT, FLOAT, REAL 등으로 구분합니다. SQL Server의 경우 ISO/ANSI의 기준에 맞춰 작은 정수, 정수, 큰 정수, 실수 등으로 제공합니다. Oracle의 경우 숫자 형 타입은 NUMBER 타입이 대표적입니다.

> 💡 ❕ **참조 - ISO(International Organization for Standardization)**
>
> ISO는 국제 표준화 기구로 여러 나라의 표준 제정 단체들의 대표들로 이루어진 국제적인 표준화 기구입니다. 나라마다 다른 산업군에서 국제적으로 통용되는 기준을 개발하고 보급하는 일을 합니다.

▼ 한 걸음 더

- NUMERIC/DECIMAL : 고정 소수점 숫자를 나타내는 데 사용
  이 둘은 거의 동일한 의미를 가지며, 몇몇 DBMS에서는 이 둘을 상호 교환하여 사용
- SMALLINT : 2Byte로 표현되는 작은 범위 정수
- BIGINT : 4Byte로 표현되는 큰 범위 정수
- FLOAT : 부동 소수점 숫자를 나타내는 데 사용
- REAL : 4Byte 부동 소수점 숫자를 나타내는데 사용

→ 저장하고자 하는 숫자의 크기와 정밀도를 고려하여 선택하는 것

2. 대표적인 데이터 유형

   위에서 언급한 자료형의 구체적인 종류에 대해서 살펴보겠습니다.

   - **CHARACTER(S)**
     - 고정 길이의 문자열 정보이며 **CHAR**로 표현합니다.
     - 기본 길이를 1 Byte로 설정합니다.
     - 최대 길이는 Oracle 2,000 Bytes, SQL Server는 8,000 Bytes로 둡니다.
     - **고정된 길이**를 갖기 때문에 입력되는 값의 길이가 해당 길이보다 작은 경우 남는 부분의 차이만큼은 **공백**으로 채워지게 됩니다.
   - **VARCHAR(S)**
     - 가변 길이의 문자열 정보이며 VARCHAR로 표현합니다.
     - Oracle의 경우 **VARCHAR2**로 표현합니다.
     - 기본 길이를 1 Byte로 설정합니다.
     - 최대 길이는 Oracle 4,000 Bytes, SQL Server는 8,000 Bytes로 둡니다.
     - 최대 길이만큼의 크기를 가질 수 있지만 길이는 **가변적으로 조정**되기 때문에 입력되는 값의 Byte만큼만 적용됩니다.
   - **NUMERIC**
     - 정수, 실수 등의 숫자 정보입니다.

   ▼ SQL Server의 경우 보다 다양한 숫자 타입을 사용하지만, Oracle은 **NUMBER** 숫자 타입이 대표적입니다.

   → 보다 정확하게는 Oracle도 다양한 숫자타입을 제공하고, 이를 사용해서 테이블 생성도 가능하지만 내부적으로는 Number 타입으로 변환되어 처리됩니다. 이러한 특징은 DBMS마다 상이하므로, 내가 사용하는 DBMS에 맞게 알아두는 것이 필요합니다.

```
-- Oracle
-- NUMBER(p,s) precision(정밀도)/scale(소수 자리수)

NUMBER -- 입력 변형 없이 허용가능한 범위 내에서 가변 저장
NUMBER(5) -- 5자리 숫자를 표현할 수 있음 (유효숫자)
NUMBER(5, 3) -- 5자리 숫자중 소수점을 포함해 3자리까지 소수를 나타낼 수 있음

/*
예제
123.56
precision 5, scale 2 -> NUMBER(5, 2)

NUMBER(3) -> NUMBER(3,[0]) -> 124
NUMBER(3, 2) -> 유효숫자가 5자리인데 3자리까지 밖에 표현을 못하므로 ERR
```

```
NUMBER(5, 1) -> 123.6 scale에 따라 반올림 되어 .5까지 표시
*/
```

- **DATETIME**
  - 날짜, 시각 등과 관련된 정보입니다.
  - SQL Server의 경우 DATETIME으로 표현하고 Oracle은 **DATE**로 표현합니다.

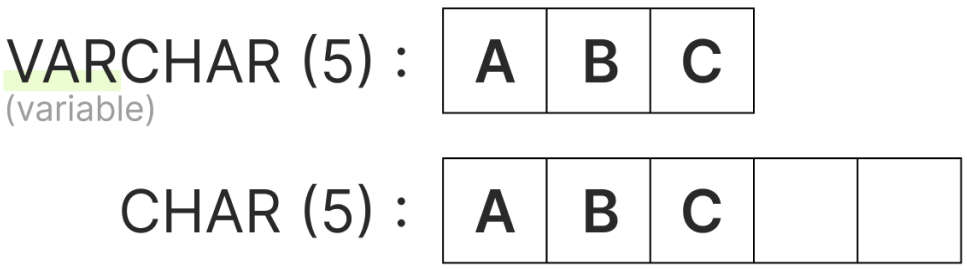
3. 문자열 비교

   문자열은 가장 많이 사용하는 데이터 타입입니다. 따라서 어떻게 저장하는지 이해하는 것은 매우 중요합니다. ANSI를 기준으로 크게 CHAR와 VARCHAR 두 가지로 구분할 수 있습니다.

   CHAR의 경우 남는 부분을 '공백(BLANK)'으로 채워서 비교합니다. 짧은 쪽의 끝에 공백을 추가하여 비교하는 2개의 문자열의 길이가 같아지도록 만들고 앞에서부터 하나씩 비교합니다. 끝의 공백만 다른 경우로 판단되는 문자열의 경우 같다고 판단합니다. 고정 길이의 문자열이기 때문에 헤더에 레코드의 길이에 대한 정보를 포함합니다.

   VARCHAR는 맨 처음부터 하나의 문자 단위로 비교합니다. 이 과정에서 공백도 하나의 문자로 취급하기 때문에 끝의 공백이 다르면 다른 문자로 판단합니다. 가변 길이의 문자열이기 때문에 헤더에 레코드의 길이에 대한 정보를 포함하지 않습니다.

   예시를 통해 조금 더 자세하게 알아보겠습니다. 'ABC'라는 문자열을 할 때 CHAR와 VARCHAR는 각각 아래와 같은 방식으로 문자를 저장합니다.



   Oracle의 경우 VARCHAR2로 문자열을 표현할 수 있습니다. 이때 주어지는 인자값에 따라 조금 다른 의미로 쓰이게 됩니다. 주어지는 인자값이 'VARCHAR2(11 BYTE)'이라면 11 Byte 크기의 문자열을 저장할 수 있습니다. 'VARCHAR2(11 CHAR)'으로 지정하면 11개의 문자열을 지정할 수 있습니다. 이때 지정하는 문자의 크기와 관계없이 11개의 문자열을 지정할 수 없습니다.

   💡 ❕참고 - 일반적으로 1개의 문자는 4 Bytes의 크기를 갖습니다.

▼ 2) 제약 조건

제약조건은 테이블에 부적절한 값이 들어오지 않도록 하는 규칙입니다. 제약조건은 마치 컬럼의 속성처럼 사용하지만 제약조건 또한 데이터베이스 입장에서는 각각의 개체이므로 **고유의 이름(제약조건 명)을 지정해주어야 합니다.** 사용자가 지정하지 않는 경우, DBMS가 자동으로 부여합니다.

- NOT NULL
  - NOT NULL 제약조건을 명시하면 해당 컬럼에는 반드시 데이터를 입력해야만 합니다. 필수로 데이터가 입력되어야 하는 컬럼에 설정합니다.

- UNIQUE
  - 유니크 단어 뜻 그대로 해당 컬럼에 들어가는 값이 테이블 전체에서 유일해야합니다. 즉, 중복값을 허용하지 않습니다. NOT NULL과 함께 사용할 수 있습니다.

- PRIMARY KEY

- 테이블의 기본키를 설정합니다. PRIMARY KEY는 기본적으로 NOT NULL + UNIQUE 제약조건을 함께 갖으며 테이블당 1개의 기본키만 설정할 수 있습니다. 여러 칼럼을 묶어 하나의 기본키로 사용할 수 있습니다.

```
CONSTRAINT PK_example PRIMARY KEY(col1, col2, col3)
```

- FOREIGN KEY
  - 외래키를 지정합니다. 참조 관계를 이용해 테이블의 데이터를 조작할 수 있습니다.
  - 참조하는 테이블이 먼저 생성되어 있어야합니다.
  - 외래키가 참조하는 컬럼은 참조하는 테이블의 기본키(PK)여야 합니다.

```
CONSTRAINT FK_example FOREIGN KEY(col4) REFERENCES table_example(col1)
```

- DEFAULT
  - 해당 속성에 값을 입력하지 않을 경우 부여될 기본 값을 지정합니다.

```
...
column2 VARCHAR2(50) DEFAULT 'Default Text',
...
```

- CHECK
  - 미리 지정한 조건에 맞는 데이터를 입력해야합니다. 주로 특정한 값, 범위를 지정합니다.

```
-- 컬럼 정의에 함께 사용할 경우

...
age NUMBER(2) CHECK(age > 14),
...
```

```
-- 제약 조건을 따로 명시할 경우

...
CONSTRAINT CK_age CHECK(age > 14)
...
```

▼ 3) CREATE TABLE

☑️ 테이블 생성

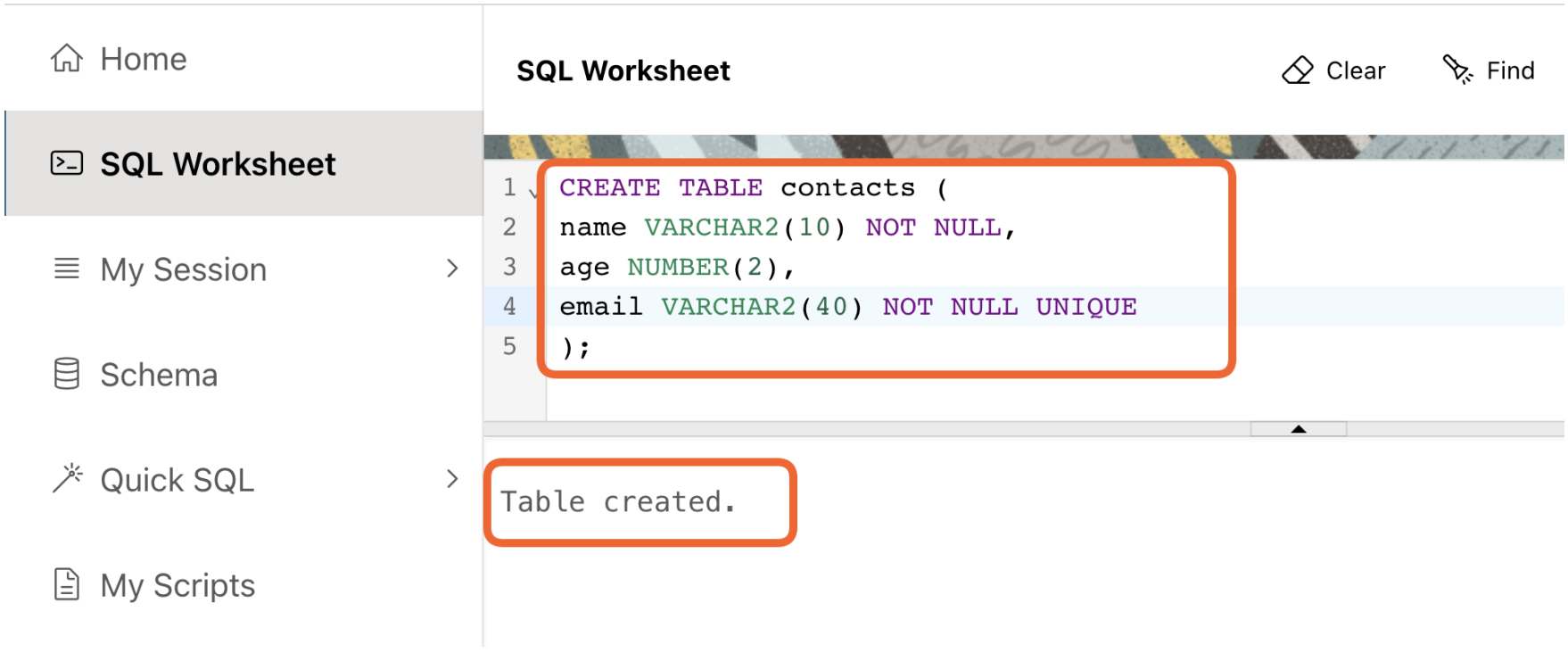
지금까지 살펴본 제약 조건들을 잘 활용하여 실제 테이블을 작성을 해봅시다. 테이블 작성에 앞서 우선 어떤 데이터를 저장할 것인지 잘 분류를 해야 합니다. 이후 어떤 제약조건이 필요한지 설정한 다음에 테이블을 생성해 주면 됩니다. 다음은 TABLE 작성 방법에 대한 예시입니다.

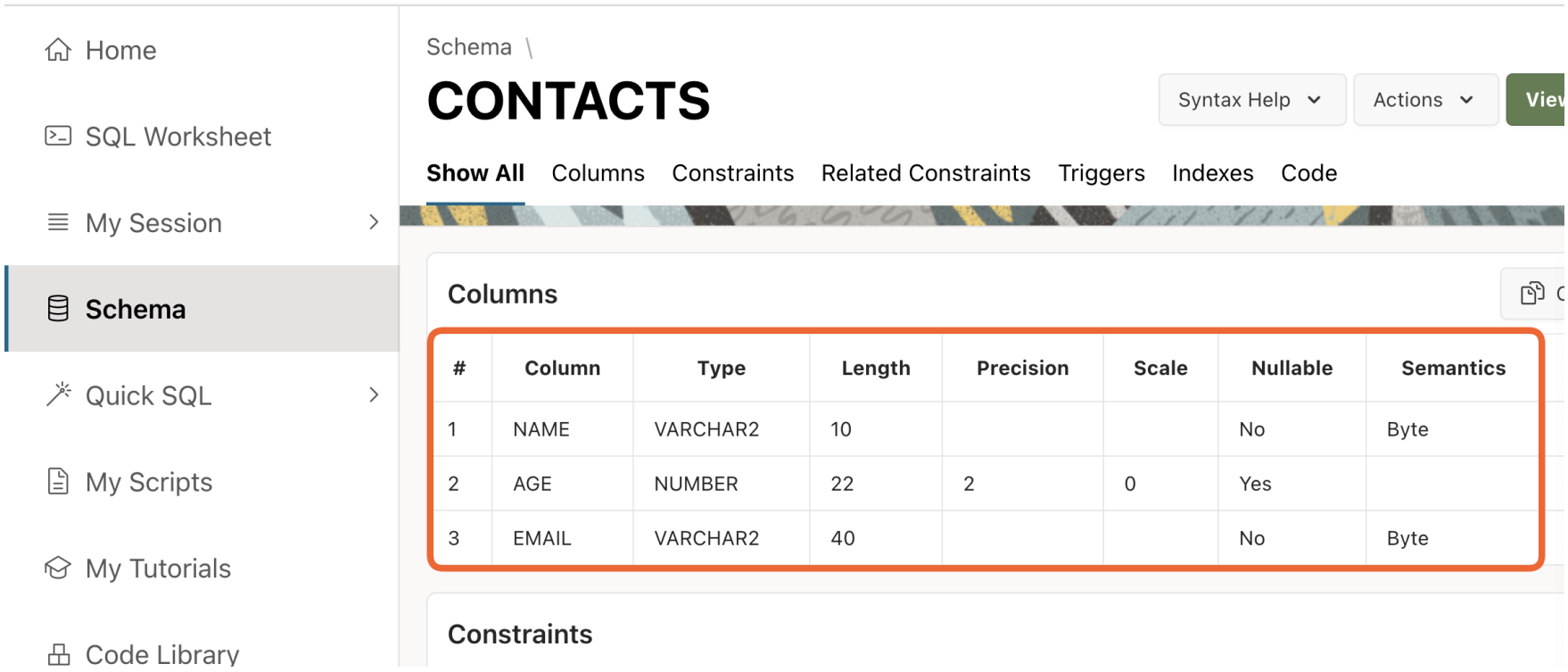
```
-- Create a new table in the database

CREATE TABLE table_name (
column_1 data_type constraints,
column_2 data_type constraints,
column_3 data_type constraints
);
```

연락처에 대한 테이블이라면

```
CREATE TABLE contact (
name VARCHAR2(10) NOT NULL,
age NUMBER(2),
email VARCHAR2(40) NOT NULL UNIQUE
);
```

▼ 백문이불여일타 !





CREATE TABLE 작성 예시

▼ 테이블 구조

```
-- 테이블 구조

테이블명 : PLAYER
테이블 설명 : K-리그 선수들의 정보를 가지고 있는 테이블
칼럼명 :
  PLAYER_ID (선수ID) 문자 고정 자릿수 7자리,
  PLAYER_NAME (선수명) 문자 가변 자릿수 20자리,
  TEAM_ID (팀ID) 문자 고정 자릿수 3자리,
  E_PLAYER_NAME (영문선수명) 문자 가변 자릿수 40자리,
  NICKNAME (선수별명) 문자 가변 자릿수 30자리,
  JOIN_YYYY (입단년도) 문자 고정 자릿수 4자리,
  POSITION (포지션) 문자 가변 자릿수 10자리,
  BACK_NO (등번호) 숫자 2자리,
  NATION (국적) 문자 가변 자릿수 20자리,
  BIRTH_DATE (생년월일) 날짜,
  SOLAR (양/음) 문자 고정 자릿수 1자리,
```

```
  HEIGHT (신장) 숫자 3자리,
  WEIGHT (몸무게) 숫자 3자리,

제약조건 :
  기본키(PRIMARY KEY) → PLAYER_ID (제약조건명은 PLAYER_ID_PK)
  값이 반드시 존재 (NOT NULL) → PLAYER_NAME, TEAM_ID
```

▼ Oracle 테이블 생성 구문

```
-- Oracle 테이블 생성 구문

CREATE TABLE PLAYER (
  PLAYER_ID CHAR(7) NOT NULL,
  PLAYER_NAME VARCHAR2(20) NOT NULL,
  TEAM_ID CHAR(3) NOT NULL,
  E_PLAYER_NAME VARCHAR2(40),
  NICKNAME VARCHAR2(30),
  JOIN_YYYY CHAR(4),
  POSITION VARCHAR2(10),
  BACK_NO NUMBER(2),
  NATION VARCHAR2(20),
  BIRTH_DATE DATE,
  SOLAR CHAR(1),
  HEIGHT NUMBER(3),
  WEIGHT NUMBER(3),
  CONSTRAINT PLAYER_PK PRIMARY KEY (PLAYER_ID)
);
```

▼ (참고) SQL Server 생성 구문

```
-- SQL Server 생성 구문

CREATE TABLE PLAYER (
  PLAYER_ID CHAR(7) NOT NULL,
  PLAYER_NAME VARCHAR(20) NOT NULL,
  TEAM_ID CHAR(3) NOT NULL,
  E_PLAYER_NAME VARCHAR(40),
  NICKNAME VARCHAR(30),
  JOIN_YYYY CHAR(4),
  POSITION VARCHAR(10),
  BACK_NO TINYINT,
  NATION VARCHAR(20),
  BIRTH_DATE DATE,
  SOLAR CHAR(1),
  HEIGHT SMALLINT,
  WEIGHT SMALLINT,
  CONSTRAINT PLAYER_PK PRIMARY KEY (PLAYER_ID)
);
```

☑️ 테이블 생성 실습

실습 문제에 맞는 SQL 문을 작성해 보세요.

> **1. 아래 정보를 참고하여 테이블을 생성하세요.**
>
> ***** 테이블명
> - pokemon
>
> * 컬럼명
> **-** pokemon_id
> - name
> **-** attr
> **-** age

> * 제약 조건
> pokemon_id
> - 문자 고정 자릿수 10자리 / NOT NULL / PRIMARY KEY
> name
> - 문자 고정 자릿수 10자리 / NOT NULL
> attr
> - 문자 가변 자릿수 10자리 / NOT NULL
> age
> - 숫자 3자리 / NOT NULL

```
-- Oracle

CREATE TABLE pokemon (
  pokemon_id CHAR(10) NOT NULL,
  name CHAR(10) NOT NULL,
  attr VARCHAR2(10) NOT NULL,
  age NUMBER(3) NOT NULL,
  CONSTRAINT pokemon_pk PRIMARY KEY (pokemon_id)
);
```

> **2. 아래 정보를 참고하여 `summer_olympic` 테이블을 생성해 보세요.**
> **- olympiad 컬럼을 제외한 모든 컬럼은 NULL 값을 허용합니다.**
>
> * 테이블명
> - summer_olympic
>
> * 컬럼명
> - olympiad
> - host_city
> - country
> - competitors

```
-- 데이터
(1992/Barcelona/Spain/9356),
(1996/Atlanta/USA/10318),
(2000/Sydney/Australia/10651),
(2004/Athens/Greece/10625),
(2008/Beijing/China/10942),
(2012/London/UK/10768),
(2016/Riode janeiro/Brazil/11238),
(2020/Tokyo/Japan/11656),
(2024/Paris/France/ ),
(2028/LA/USA/ ),
(2032/Brisbane/Australia/),
(2036/ / /)
```

```
-- Oracle

CREATE TABLE summer_olympic  (
  olympiad CHAR(10) NOT NULL,
  host_city VARCHAR2(30),
  country VARCHAR2(30),
  competitors NUMBER(20),
```

```
  CONSTRAINT olympiad_pk PRIMARY KEY (olympiad)
);
```

▼ 4) ALTER TABLE

☑️ COLUMN / CONSTRAINT 추가

처음부터 테이블 구조를 완벽하게 설계하기는 어렵습니다. 업무적인 요구 사항이 발생하거나 시스템 운영상 데이터를 저장하는 도중에 수정 사항이 발생하는 경우가 빈번하기 때문입니다. 이때 테이블의 칼럼을 직접 추가, 수정 및 삭제를 할 수 있으며 제약조건도 추가 혹은 삭제가 가능하기 때문에 보다 유연하게 테이블을 운영할 수 있습니다.

1. **COLUMN 추가**

   테이블에 추가 정보를 저장할 필요가 생겼을 때 아래와 같은 방법을 사용하면 됩니다. 새롭게 추가되는 칼럼은 테이블의 마지막에 위치하게 되며 칼럼의 위치를 우리가 직접 지정할 수 없습니다.

   - 구조

   ```
   ALTER TABLE 테이블명 ADD 칼럼명 데이터_유형;
   ```

- 예시
  - player 테이블에 address 칼럼 추가하기

```
-- Oracle

ALTER TABLE player ADD address VARCHAR2(80);
```

```
-- SQL Server

ALTER TABLE player ADD address VARCHAR(20);
```

- 실습

```
CREATE TABLE summer_olympic (
  olympiad NUMBER PRIMARY KEY, -- 속성과 함께 PK 설정
  host_city VARCHAR2(20),
  country VARCHAR2(20),
  competitors NUMBER
);
```

> **위 SQL문을 활용하여 테이블을 생성 후, summer_olympic 테이블의 올림픽 개회일 칼럼을 추가하세요.**
>
> *** 컬럼명**
> start_date

```
-- Oracle

ALTER TABLE summer_olympic ADD start_date DATE;
```

2. **CONSTRAINT 추가**

> 💡 **CONSTRAINT(제약조건)이란?**
> 데이터의 무결성을 유지하기 위한 방법으로 테이블의 특정 칼럼에 설정하는 제약
>
> **제약 조건의 종류**
>
> - 기본 키(Primary Key) : UNIQUE & NOT NULL
> - 고유 키(Unique Key) : 고유키 정의
> - NOT NULL : NULL 값 입력금지
> - CHECK : 입력 값 범위 제한
> - 외래 키(Foreign Key) : NULL 가능, 여러속성가능

칼럼을 생성하고 미처 제약조건을 추가하지 못했다면 ADD CONSTRAINT를 통하여 제약조건을 추가할 수 있습니다.

- 구조

```
-- ADD Constraint

ALTER TABLE 테이블_이름 ADD CONSTRAINT 제약_조건_이름 제약조건 (컬럼_이름);
```

- 예시
  - 아래의 SQL문을 활용하여 테이블을 하나 더 추가하기

```
CREATE TABLE TEAM (
  team_id CHAR(3) PRIMARY KEY,
  team_name VARCHAR2(20)
);
```

  - player 테이블에 team 테이블에서 team_id를 FK로 추가하기

```
-- ADD Constraint

ALTER TABLE player ADD CONSTRAINT player_fk FOREIGN KEY team_id
REFERENCES TEAM(team_id);
```

☑️ COLUMN / CONSTRAINT 삭제

1. **COLUMN 삭제**

   테이블을 생성할 때 필요할 것이라고 생각하고 만들었는데, 생성한 칼럼이 쓸모가 없어지는 경우가 있습니다. 이때는 테이블에서 해당 칼럼을 삭제해야 합니다.

   - 구조

```
ALTER TABLE 테이블_이름 DROP COLUMN 삭제_할_컬럼_이름;
```

   - 예시
     - player 테이블에서 address 칼럼 삭제하기

```
-- Oracle

ALTER TABLE player DROP COLUMN address;
```

```
-- SQL Server
```

```
ALTER TABLE player DROP COLUMN address;
```

- 실습

> **summer_olympic 테이블의 올림픽 개회일(start_date) 컬럼을 삭제하세요.**

```
CREATE TABLE summer_olympic (
  olympiad INT PRIMARY KEY,
  host_city VARCHAR(20),
  country VARCHAR(20),
  competitors INT,
  start_date DATE
);
```

```
-- Oracle

ALTER TABLE summer_olympic DROP COLUMN start_date;
```

2. **CONSTRAINT 삭제**

특정 칼럼에서 더는 제약조건이 필요하지 않다면 DROP CONSTRAINT를 통하여 설정된 제약조건을 삭제할 수 있습니다.

- 구조

```
-- DROP Constraint

ALTER TABLE 테이블_이름 DROP CONSTRAINT 제약_조건_이름;
```

- 예시
  - player 테이블에서 player_fk 라는 제약조건을 삭제하기

```
-- DROP Constraint

ALTER TABLE player DROP CONSTRAINT player_fk;
```

- 실습

> **friend 테이블에서 name칼럼에 설정된 Primary Key 제약 조건을 삭제하세요.**

```
CREATE TABLE friend (
  name VARCHAR(20) ,
  age VARCHAR(3),
  phone_number VARCHAR(20) NOT NULL,
  address VARCHAR(30),
  CONSTRAINT phone_number_PK PRIMARY KEY(Phone_number)
);
```

```
-- Oracle

ALTER TABLE friend DROP CONSTRAINT phone_number_PK;
```

☑️ COLUMN 수정

1. **COLUMN 수정**

칼럼의 데이터 유형, DEFAULT 값, NOT NULL 제약조건에 대한 변경을 할 수 있습니다. 칼럼의 데이터의 크기를 변경할 때 크기를 늘릴 수는 있지만 데이터 훼손 우려로 인하여 줄이지는 못합니다.

- 구조

```
-- Oracle

ALTER TABLE 테이블_이름 MODIFY (
  컬럼_이름_1 데이터_유형 [DEFAULT 식] [NOT NULL],
  컬럼_이름_2 데이터_유형 [DEFAULT 식] [NOT NULL]
  ...);
```

```
-- SQL Server

ALTER TABLE 테이블_이름 ALTER COLUMN (
  컬럼_이름_1 데이터_유형 [DEFAULT 식] [NOT NULL],
  컬럼_이름_2 데이터_유형 [DEFAULT 식] [NOT NULL]
  ...);
```

- 예시
  - `ORIG_YYYY CHAR(4),` 설정을 변경하기

```
-- Oracle
ALTER TABLE TEAM_TEMP
MODIFY (ORIG_YYYY VARCHAR2(8) DEFAULT '20020129' NOT NULL);
```

```
-- SQL Server

ALTER TABLE TEAM_TEMP
ALTER COLUMN ORIG_YYYY VARCAHR(8) NOT NULL;
```

- 실습

> **summer_olympic 테이블의 참가자(competitors) 컬럼의 데이터 유형을 고정 길이(10) 문자형으로 변경하세요.**

```
CREATE TABLE summer_olympic (
  olympiad INT PRIMARY KEY,
  host_city VARCHAR(20),
  country VARCHAR(20),
  competitors NUMBER,
  start_date DATE
);
```

```
-- Oracle
ALTER TABLE summer_olympic
MODIFY (competitors CHAR(10));
```

☑️ COLUMN / TABLE 이름 변경

1. **COLUMN 이름 변경**

   칼럼 이름을 수정할 수 있습니다.

   - 구조

```
-- Oracle

ALTER TABLE 테이블_이름 RENAME COLUMN 변경_할_컬럼_이름 TO 새로운_컬럼_이름;
```

```
-- SQL Server
```

```
SP_RENAME 테이블_명.변경_할_컬럼_이름, 새로운_컬럼_이름, 'COLUMN';
```

- 예시
  - player_name 칼럼 이름을 name 으로 수정하기

```
-- oracle

ALTER TABLE player
RENAME COLUMN player_name TO name;
```

```
-- SQL Server

SP_RENAME player.player_name, name, 'COLUMN';
```

- 실습

> **summer_olympic 테이블의 olympiad 칼럼 이름을 year로 변경하세요.**

```
CREATE TABLE summer_olympic (
  olympiad INT PRIMARY KEY,
  host_city VARCHAR(20),
  country VARCHAR(20),
  competitors INT,
  start_date DATE
);
```

```
-- Oracle

ALTER TABLE summer_olympic
RENAME COLUMN olympiad TO year;
```

```
-- SQL Server

SP_RENAME summer_olympic.olympiad, year, 'COLUMN';
```

2. **TABLE 이름 변경**

   테이블의 이름을 수정할 필요가 있을 때 사용합니다. ALTER TABLE 없이 이름을 변경할 수 있습니다.

   - 구조

```
-- Oracle

RENAME 기존_테이블_이름 TO 새로운_테이블_이름;
```

```
-- SQL Server

SP_RENAME 기존_테이블_이름, 새로운_테이블_이름;
```

   - 예시
     - player 테이블 명을 football_player로 변경하기

```
-- Oracle

RENAME player TO football_player;
```

```
-- SQL Server

SP_RENAME player, football_player;
```

- 실습

> **friend 테이블의 이름을 address_book 으로 변경하세요.**

```
CREATE TABLE friend (
  name VARCHAR(20) NOT NULL,
  age VARCHAR(3),
  phone_number VARCHAR(20) NOT NULL,
  address VARCHAR(30)
);
```

```
-- Oracle

RENAME friend TO address_book;
```

```
-- SQL Server

SP_RENAME friend, address_book;
```

▼ 5) DROP TABLE , TRUNCATE TABLE

☑️ DROP TABLE

테이블 전체를 삭제할 때 사용하며 저장된 데이터는 같이 삭제됩니다. `CASCADE CONSTRAINT` 옵션은 해당 테이블을 참조하고 있는 제약조건에 대해서 해당 테이블이 삭제될 때 같이 삭제될지 여부를 결정하는 옵션입니다. 다만, SQL Server에서는 해당 옵션이 존재하지 않기 때문에 참조 테이블 관계가 있을 시 제약 조건 혹은 참조 테이블을 먼저 삭제해야 테이블을 삭제할 수 있습니다.

▼ 참조 무결성

참조 무결성이란 데이터베이스 상의 **참조가 모두 유효함**을 말합니다. 즉, 관계형 데이터베이스에서 하나의 속성이 다른 테이블의 속성을 참조하고 있다면, 참조하고 있는 해당 속성이 반드시 존재해야 한다는거죠.

상품과 주문이라는 각각의 테이블이 있을 때 모든 주문은 반드시 한 개의 상품을 FK로 가지고 있을 거예요. 이 상태에서 만약 참조하고 있는 상품이 사라지면 해당 주문은 어떻게 되어야 하는 걸까요? 이럴 때 사용하는 옵션이 바로 CASCADE입니다. 참조가 걸려있는 값을 수정/삭제할 때, 해당 값을 참조하고 있는 모든 레코드 역시 종속적으로 수정/삭제를 가능하게 하는 옵션이에요.

```
-- 참조하고 있는 모든 레코드도 함께 수정
ALTER TABLE ORDERS ADD FOREIGN KEY (product_id) REFERENCES Product(id) ON UPDATE CASCADE;
```

```
-- 참조하고 있는 모든 레코드도 함께 삭제
ALTER TABLE ORDERS ADD FOREIGN KEY (product_id) REFERENCES Product (id) ON DELETE CASCADE;
```

만약, 상품이 사라졌다고 해서 주문도 삭제하고 싶지 않다면?

상품이 사라진 주문이 있다면 해당 FK 값은 `NULL` 처리 해주세요 (`SET NULL`)

```
-- 참조하던 레코드가 사라지면 NULL 값으로 변경
ALTER TABLE ORDERS ADD FOREIGN KEY (product_id) REFERENCES Product (id) ON DELETE SET NULL;
```

이러한 조건들은 테이블을 생성할 때 FK 필드에 한 번에 지정할 수 있습니다.

```
-- Orders 테이블 생성
CREATE TABLE Orders (
    order_id NUMBER PRIMARY KEY
```

```
    -- 다른 컬럼들...
);

-- OrderItems 테이블 생성
CREATE TABLE OrderItems (
    item_id NUMBER PRIMARY KEY,
    order_id NUMBER,
    item_name VARCHAR2(50),
    CONSTRAINT fk_order_id FOREIGN KEY (order_id) REFERENCES Orders(order_id) ON DELETE CASCADE
);
```

- 구조
  - Oracle / SQL Server 공통

```
DROP TABLE 테이블_이름 [CASCADE CONSTRAINT];
```

- 예시
  - player 테이블 삭제하기

```
DROP TABLE player;
```

- 실습

> **friend 테이블을 삭제해보세요.**

```
CREATE TABLE friend (
  name VARCHAR(20) NOT NULL,
  age VARCHAR(3),
  phone_number VARCHAR(20) NOT NULL,
  address VARCHAR(30)
);
```

```
-- Oracle / SQL Server

DROP TABLE friend;
```

☑️ TRUNCATE TABLE

테이블은 남기고 데이터만 지우고 싶을 때 사용합니다. 테이블에 저장된 모든 데이터는 삭제되며 테이블 구조는 그대로 남아있게 됩니다.

- 구조
  - Oracle / SQL Server 공통

```
TRUNCATE TABLE 테이블_이름;
```

- 예시
  - player 테이블의 데이터 지우기

```
TRUNCATE TABLE player;
```

- 실습

> 다음 friend 테이블에 대하여 테이블은 그대로 유지하고 데이터만 삭제해보세요.

```
CREATE TABLE friend (
  name VARCHAR2(20) NOT NULL,
  age NUMBER,
  phone_number VARCHAR2(20)
);

INSERT INTO friend VALUES ('KIM', 20, '010-1234-1567');
INSERT INTO friend VALUES ('LEE', 21, '010-1111-1234');
INSERT INTO friend VALUES ('PARK', 20, '010-1234-5675');
```

```
-- Oracle / SQL Server

SELECT COUNT(*) FROM friend; -- 데이터 조회해보기 (데이터 존재)
TRUNCATE TABLE friend;
SELECT COUNT(*) FROM friend; -- 데이터 조회해보기 (데이터 없음)
```

▼ 6) 학습 요약

- **데이터 유형**
  - CHAR(s) : 고정 길이 문자열 정보 최대 . 길이 만큼 공간 채움 ‘AA’ = ‘AA ’
  - VARCHAR2(s) : 가변 길이 문자열 정보 할당된 변수 값의 바이트만 적용 ‘AA’ != ‘AA ’
  - NUMBER : 정수, 실수 등 숫자 정보
  - DATE : 날짜와 시각 정보
- **테이블 생성 규칙**
  - 테이블 명은 다른 테이블의 이름과 중복되면 안된다.
  - 테이블 내의 칼럼명은 중복될 수 없다.
  - 각 칼럼들은 ',' 로 구분되고 ';' 로 끝난다
  - 칼럼 뒤에 데이터 유형은 꼭 지정되어야 한다.
  - 테이블명과 칼럼명은 반드시 문자로 시작해야한다.
  - A-Z,a-z,0-9,_,$,#만 사용 가능
- **테이블 생성**
  - CREATE TABLE PLAYER (
  - PLAYER_ID CHAER(7) NOT NULL,
  - PLAYER_NAME VARCHAR2(20) NOT NULL);
  - 테이블 구조 변경
    - 추가 : ALTER TABLE PLAYER ADD(ADDRESS VARCHAR2(80));
    - 삭제 : ALTER TABLE PLAYER DROP COLUMN ADDRESS;
    - 수정 : ALTER TABLE TEAM_TEMP MODIFY
      (ORIG_YYYY VARCHAR2(8) DEFAULT '20020129' NOT NULL);

- **제약 조건의 종류**
  - PRIMARY KEY(기본키) : UNIQUE & NOT NULL
  - UNIQUE KEY(고유키) : 고유키 정의
  - NOT NULL : NULL 값 입력금지
  - CHECK : 입력 값 범위 제한
  - FOREIGN KEY(외래키) : NULL 가능, 여러속성가능
- **제약 조건 SQL**

- 제약조건 삭제 : DROP CONSTRAINT 조건명;
- 제약조건 추가 : ADD CONSTRAINT 조건명 조건 (칼럼명);
- 테이블명 변경 : RENAME PLAYER TO PLAYER_BACKUP;
- 테이블 삭제 : DROP TABLE PLAYER;
- 테이블 데이터 삭제 : TRUNCATE TABLE PLAYER;
- 컬럼명 변경 : RENAME COLUMN TEAM_ID TO T_ID;

▼ 연습문제

☑️ 문제 1

Q. 문제	대표적인 데이터 유형으로 옳지 않은 것은?
A. (1)	CHARACTER
A. (2)	VARCHAR
A. (3)	NUMERIC
A. (4)	Standardization

▼ 정답

(4) 대표적인 데이터 유형으로으로는 CHARACTER, VARCHAR, NUMERIC, DATETIME 이 있다

☑️ 문제 2

Q. 문제	CONSTRAINT (제약 조건)의 종류로 옳지 않은 것은?
A. (1)	PRIMARY KEY
A. (2)	SUBKEY
A. (3)	UNIQUE
A. (4)	CHECK

▼ 정답

(2) 제약 조건의 종류에는 Primary Key, Unique Key, NOT NULL, CHECK, Foreign Key가 있다

☑️ 문제 3

Q. 문제	테이블은 남기고 데이터만 지우고 싶을 때 사용하는 SQL은?
A. (1)	CREATE TABLE
A. (2)	DROP TABLE
A. (3)	TRUNCATE TABLE
A. (4)	ALTER TABLE

▼ 정답

(3)