Database

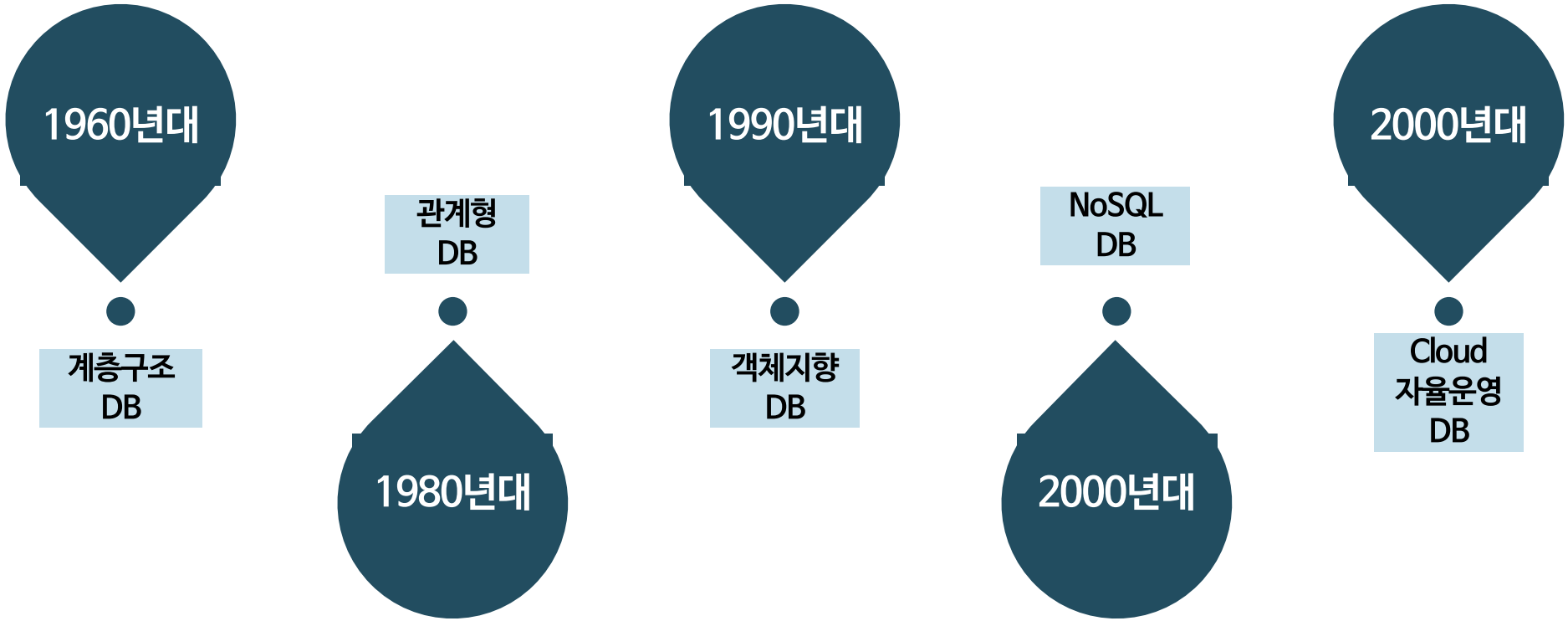
1

Database 기본개요

1. Database
2. DBMS
3. RDBMS
4. SQL

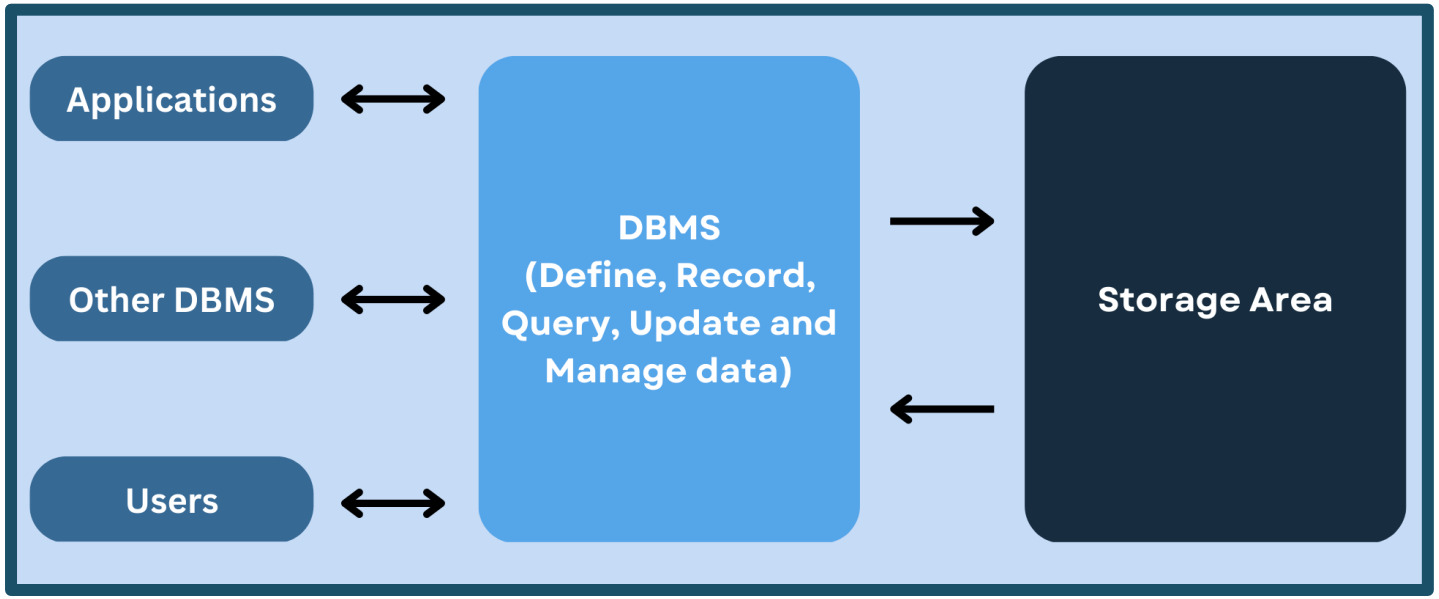
1.1. Database 개요

- 데이터베이스는 구조화된 정보 또는 데이터의 조직화된 모음으로서 일반적으로 컴퓨터 시스템에 전자적으로 저장된다
- 데이터베이스는 일반적으로 데이터베이스 관리 시스템(DBMS)에 의해 제어된다.
- 연결된 애플리케이션과 함께 데이터와 DBMS를 하나로 묶어 데이터베이스 시스템이라고 하며 단축하여 데이터베이스라고도 한다.



1.2. 데이터베이스 관리 시스템(DBMS)이란[1/4]

- 데이터베이스 관리 시스템(DBMS) : 데이터베이스를 효율적으로 관리하고 조작하기 위한 소프트웨어
- DBMS를 통해 데이터베이스를 관리하여 응용 프로그램들이 데이터베이스를 공유하고, 사용할 수 있는 환경 제공
- DBMS는 데이터베이스를 구축하는 틀을 제공하고, 효율적으로 데이터를 검색하고 저장하는 기능을 제공한다
- 응용 프로그램들이 데이터베이스에 접근할 수 있는 인터페이스를 제공
- 장애에 대한 복구 기능, 사용자 권한에 따른 보안성 유지 기능 등을 제공한다



1.2. DBMS의 역할 [2/4]

- 데이터 저장 및 관리:
  DBMS는 데이터를 디스크 또는 메모리에 저장하고, 데이터의 무결성과 일관성을 유지한다.
- 데이터 검색과 조회:
  사용자가 데이터를 검색하고 필요한 정보를 빠르게 조회할 수 있도록 SQL과 같은 질의 언어를 제공.
- 데이터 보안:
  데이터 액세스 권한을 관리하고, 사용자 및 역할에 따라 데이터에 대한 보안을 제공
- 데이터 무결성 유지:
  데이터의 무결성을 유지하고, 일관성을 제공하기 위해 데이터베이스 제약 조건을 관리
- 데이터 백업 및 복구:
  DBMS는 데이터를 주기적으로 백업하고 데이터 손실 시 데이터를 복구하는 기능을 제공
- 동시성 제어:
  여러 사용자가 동시에 데이터에 접근할 때 충돌을 방지하고, 데이터 일관성을 유지하는 기능 제공.
- 성능 최적화:
  DBMS는 쿼리 최적화, 인덱싱, 캐싱 등을 통해 데이터베이스 성능을 최적화
- 데이터 모델 지원: 관계형, 객체지향, 그래프 등 다양한 데이터 모델을 지원하여 다양한 유형의 데이터를 다룰 수 있다
- 데이터 무결성 검사: DBMS는 데이터가 정확하고 일관성 있는지를 검사하며, 오류를 방지하고 유지한다
- 데이터베이스 백업 및 복원: 시스템 장애 또는 오류 발생 시 데이터의 안전한 백업 및 복원을 수행.

1.2. DBMS 종류 [3/4]

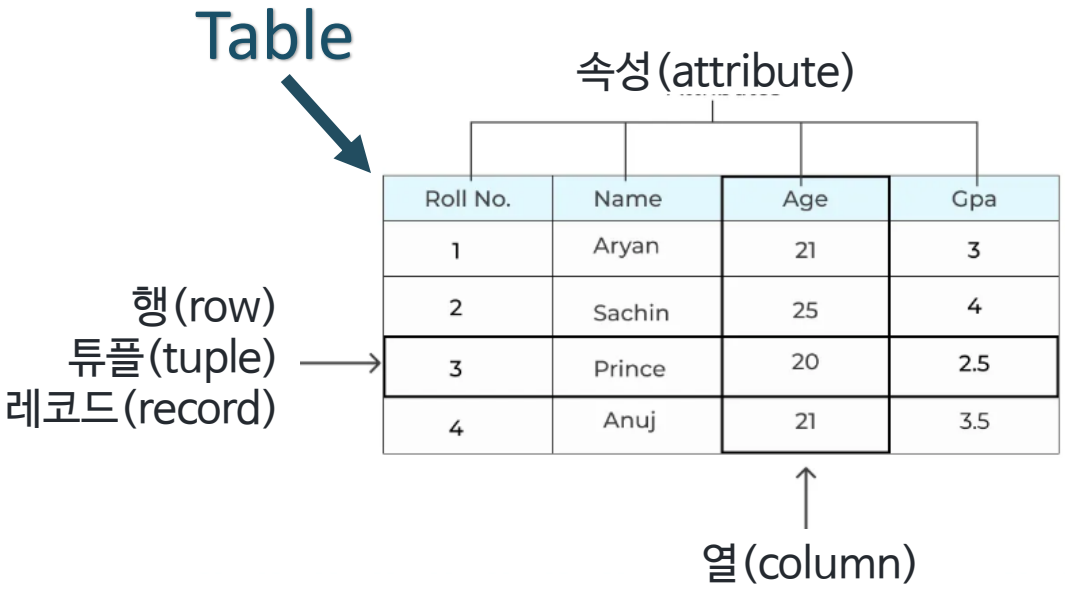
- 관계형 데이터베이스 (Relational Database):
  - 테이블 형태로 데이터를 저장하며, 데이터 간의 관계를 정의하는 데이터베이스
  - 대표적으로 MySQL, PostgreSQL, Oracle 등이 있다.
- NoSQL 데이터베이스 (Not Only SQL):
  - 비관계형 데이터 모델을 사용하는 데이터베이스로, 스키마가 유연하며 대용량 및 분산 데이터를 다룰 수 있다.
  - 대표적으로 MongoDB, Cassandra, Redis 등이 있다.
- 키-값 스토어 (Key-Value Store):
  - 간단한 키와 값으로 데이터를 저장하는 NoSQL 데이터베이스 유형.
  - Redis와 Amazon DynamoDB
- 문서 데이터베이스 (Document Database):
  - JSON 또는 XML과 같은 문서 형식으로 데이터를 저장하며, 복잡한 데이터 구조를 지원.
  - MongoDB와 Couchbase
- 그래프 데이터베이스 (Graph Database)
- 시계열 데이터베이스 (Time-Series Database)

1.2. NoSQL Database[4/4]

- NoSQL(Not Only SQL)은 관계형 데이터베이스 관리 시스템(RDBMS)과는 다른 데이터 저장 및 관리 방식을 지원하는 데이터베이스 유형이다
- NoSQL 데이터베이스는 스키마를 사용하지 않거나 유연한 스키마를 사용하며, 대량의 분산 데이터를 처리하고 확장성을 제공하는 데 주로 사용된다
- 주요 NoSQL 데이터베이스 유형에는 문서 지향 데이터베이스, 키-값 저장소, 열 지향 데이터베이스 및 그래프 데이터베이스 등이 있다
- NoSQL 데이터베이스는 다양한 용도로 사용되며, 웹 애플리케이션, 소셜 미디어 플랫폼, 센서 데이터 처리, 실시간 분석 등 다양한 분야에서 활용된다
- 각 NoSQL 데이터베이스는 데이터 모델과 사용 사례에 따라 선택해야 할 수 있으며, MongoDB, Cassandra, Redis, Neo4j 등이 널리 사용되는 NoSQL 데이터베이스이다
- NoSQL 데이터베이스는 관계형 데이터베이스와 비교하여 특정 사용 사례에 더 적합한 경우가 있으며, 빅 데이터 및 실시간 데이터 처리와 같은 현대적인 데이터 요구 사항을 충족시키기 위해 개발되었다

1.3. 관계형 데이터베이스(DBMS)[1/4]

- Relational Database Management System 즉, 관계형 데이터 베이스 관리 시스템은 데이터 베이스의 한 종류로 현재 가장 많이 사용된다.
- RDBMS는 테이블로 이루어져 있으며, 테이블은 열과 행으로 구성되어 있으며, 모든 데이터가 테이블에 저장되는 것이 가장 기본적이고 중요한 구성이다



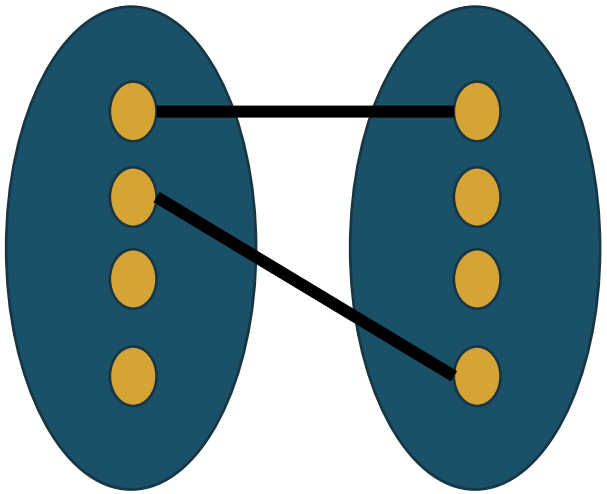
- **RDBMS의 특징**
  - 데이터의 분류, 정렬, 탐색 속도가 빠르다
  - 오랫동안 사용된 만큼 신뢰성이 높고, 어떤 상황에서도 데이터의 무결성을 보장해 준다.
  - 기존에 작성된 스키마를 수정하기가 어렵다.
  - 데이터베이스의 부하를 분석하는 것이 어렵다.

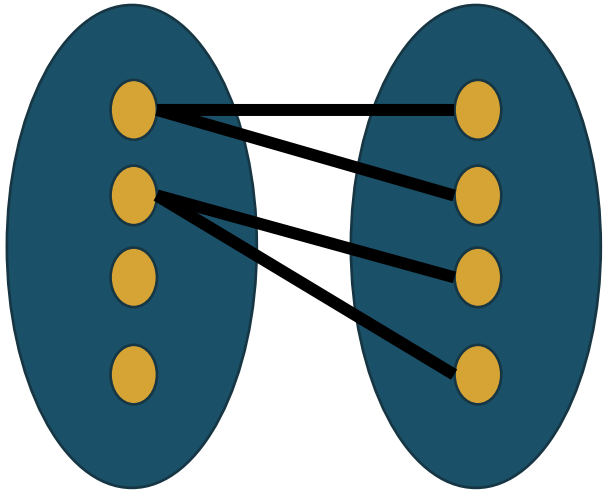
1.3. RDBMS 용어[2/4]

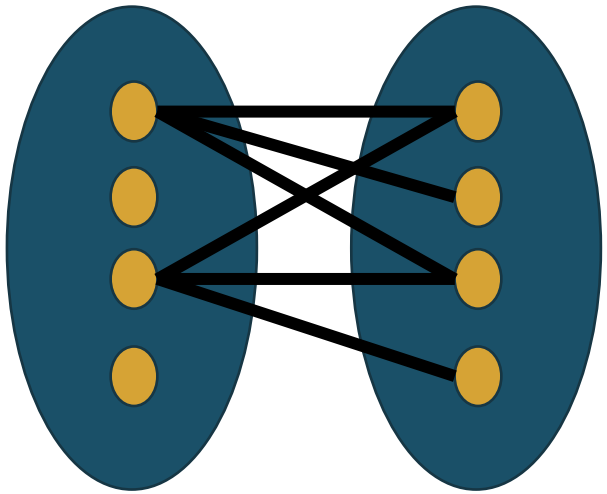
- **열(column)**
  - ✓ 각각의 열은 유일한 이름을 가지고 있으며, 자신만의 타입을 가지고 있다.
  - ✓ 이러한 열은 필드(field) 또는 속성(attribute)이라고도 불린다.
- **2. 행(row)**
  - ✓ 행은 관계된 데이터의 묶음을 의미한다.
  - ✓ 한 테이블의 모든 행은 같은 수의 열을 가지고 있다.
  - ✓ 이러한 행은 튜플(tuple) 또는 레코드(record)라고도 불린다.
- **3. 값(value)**
  - ✓ 테이블은 각각의 행과 열에 대응하는 값을 가지고 있습니다.
  - ✓ 이러한 값은 열의 타입에 맞는 값이어야 한다.
- **4. 키(key)**
  - ✓ 테이블에서 행의 식별자로 이용되는 열을 키(key) 또는 기본 키(primary key)라고 한다.
  - ✓ 즉, 테이블에 저장된 레코드를 고유하게 식별하는 후보 키(candidate key) 중에서 데이터베이스 설계자가 지정한 속성을 의미한다.

1.3. RDBMS - Table간의 관계(relationship) [3/4]

- 테이블 간의 관계는 관계를 맺는 테이블의 수에 따라 다음과 같이 나눌 수 있다.
  - ✓ 일대일(one-to-one) 관계
  - ✓ 일대다(one-to-many) 관계
  - ✓ 다대다(many-to-many) 관계
- RDBMS에서는 이러한 관계를 나타내기 위해 외래 키(foreign key)라는 것을 사용한다
  - ✓ 외래 키는 한 테이블의 키 중에서 다른 테이블의 행(row)을 식별할 수 있는 키를 의미







1.3. RDBMS - Table간의 관계(relationship) [4/4]

- RDMBS에서는 테이블간의 관계를 나타내기 위해 Primary Key와 Foreign key개념을 도입, 테이블간 join이 가능하다

Primary Key

회원 번호	회원 이름	휴대폰 번호
100	김장화	010-1234-5678
200	홍길동	010-7890-4321
300	사임당	010-2345-9080

User Table

Primary Key　Foreign Key

주문 번호	주문 회원 번호	주문 상품
230801001	100	양파
230801002	200	콩나물
230801003	100	배추

Order Table

2

MySQL

1. SQL 개요
2. MySQL 문법
3. 자료타입
4. 연산자와 함수
5. 다중테이블 연산
6. INDEX

2.1. SQL 개요

- SQL : 관계형 데이터베이스에 정보를 저장하고 처리하기 위한 프로그래밍 언어이다.
- 관계형 DB는 정보를 테이블 형식으로 저장하며, 행과 열은 다양한 데이터 속성과 데이터 값 간의 다양한 관계를 나타낸다.
- SQL 문을 사용하여 데이터베이스에서 정보를 저장, 업데이트, 제거, 검색 및 검색할 수 있다.
- 데이터베이스 성능을 유지 관리하고 최적화하는 데 SQL을 사용할 수도 있다.
- SQL 구문도 위의 목적에 맞게 크게 세 가지로 구분할 수 있다
  - DDL(Data Definition Language)
  - DML(Data Manipulation Language)
  - DCL(Data Control Language)

속성	설명	주요 명령어
DDL	데이터베이스나 테이블 등을 생성, 삭제하거나 그 구조를 변경하기 위한 명령어	CREATE, ALTER, DROP
DML	데이터베이스에 저장된 데이터를 처리하거나 조회, 검색하기 위한 명령어	INSERT, UPDATE, DELETE, SELECT 등
DCL	데이터베이스에 저장된 데이터를 관리하기 위하여 데이터의 보안성 및 무결성 등을 제어하기 위한 명령어	GRANT, REVOKE 등

2.2. MySQL 문법[1/5]

- MySQL에서 데이터베이스에 대한 작업 명령은 SQL 구문을 이용하여 처리
- SQL 구문을 구분하는 기준 - 일반적인 구문 뒤에는 세미콜론(;)을 붙인다
- MySQL은 키워드와 구문에서 대소문자를 구분하지 않는다
- 테이블 명과 필드의 이름은 대소문자를 구분하므로, 주의해서 사용해야 한다
- MySQL에서 주석 작성 방법
  - # 한 줄 주석
  - -- 한 줄 주석 (두 개 이상의 하이픈(-) 뒤에는 반드시 한 칸의 공백이 존재해야만 주석으로 정상 인식)
  - /* 두 줄
    이상의
    주석 */

2.2. MySQL 문법[2/5] - 데이터베이스와 테이블 생성

- MySQL에서는 CREATE 문을 사용하여 데이터베이스와 테이블을 만들 수 있다
  - CREATE DATABASE 데이터베이스이름
  - CREATE TABLE 테이블이름
- **테이블 생성**
  - 데이터베이스는 하나 이상의 테이블로 구성되며, 생성한 테이블에 데이터를 저장하여 관리할 수 있다.
  - 테이블을 생성하기 위해서는 테이블 이름, 필드(field) 목록과 각 필드의 타입을 명시해야 한다.
  - 필드의 타입이란 해당 필드에 저장될 데이터가 가질 수 있는 타입을 의미한다

- **문법**

```
CREATE TABLE 테이블이름
(
   필드이름1 필드타입1,
   필드이름2 필드타입2,
   ...
)
```

- **CREATE TABLE문 제약조건**

1. NOT NULL : 해당 필드는 NULL 값을 저장할 수 없다
2. UNIQUE : 해당 필드는 고유의 값을 가져야만 한다.
3. PRIMARY KEY : 해당 필드가 NOT NULL과 UNIQUE 제약 조건의 특징을 모두 가지게 다된.
4. FOREIGN KEY : 하나의 테이블을 다른 테이블에 의존하게 함
5. DEFAULT : 해당 필드의 기본값을 설정

2.2. MySQL 문법[3/5] - 테이블 수정/삭제

- **테이블 수정**

- 새로운 필드 추가
  - ALTER TABLE 테이블이름 ADD 필드이름 필드타입
- 기존 필드 삭제
  - ALTER TABLE 테이블이름 DROP 필드이름
- **필드 타입 변경**
  - ALTER TABLE 테이블이름 MODIFY COLUMN 필드이름 필드타입

- **테이블 삭제**

- DROP TABLE 문은 해당 테이블을 삭제해준다
  - DROP TABLE 테이블이름
  - 테이블을 삭제하면 해당 테이블의 모든 데이터도 다 같이 삭제되므로 주의

- **테이블내의 데이터만삭제**

- 테이블의 데이터만을 지우고 싶을 때는 TRUNCATE TABLE 문을 사용할 수 있다
  - TRUNCATE TABLE 테이블이름

2.2. MySQL 문법[4/5] - 레코드 추가/수정/삭제

• 테이블에 레코드 추가

- NSERT INTO 문과 함께 VALUES 절을 사용하여 해당 테이블에 새로운 레코드를 추가

```
INSERT INTO 테이블이름(필드이름1, 필드이름2, 필드이름3, ...)
VALUES (데이터값1, 데이터값2, 데이터값3, ...)
```

• 레코드 내용 수정

- UPDATE 문을 사용하여 레코드의 내용을 수정할 수 있다
- WHERE 절의 조건을 만족하는 레코드의 값만 수정

```
UPDATE 테이블이름
SET 필드이름1=데이터값1, 필드이름2=데이터값2, ...
WHERE 필드이름=데이터값
```

• 테이블내의 레코드 삭제

- DELETE 문을 사용하여 테이블의 레코드를 삭제
- 해당 테이블에서 WHERE 절의 조건을 만족하는 레코드만을 삭제. 테이블에서 명시된 필드와, 그 값이 일치하는 레코드만 삭제
  - DELETE FROM 테이블이름
  - WHERE 필드이름=데이터값

2.2. MySQL 문법[5/5] - 레코드 조회/선택

- SELECT 문을 사용하여 테이블의 레코드를 선택하고 조회할 수 있다
- FROM 절은 레코드를 선택할 테이블의 이름을 명시
- 해당 테이블에서 선택하고 싶은 필드의 이름을 SELECT 키워드 바로 뒤에 명시
- WHERE 절을 사용하면, 선택할 레코드의 조건을 좀 더 상세히 설정할 수 있다

```
SELECT 필드이름
FROM 테이블이름
[WHERE 조건]
```

• 중복된 값 제거

- 만약 같은 필드에 중복되는 값을 가지는 레코드가 있다면, DISTINCT 키워드를 사용하여 그 값이 한 번만 선택되도록 설정

```
SELECT DISTINCT Name
FROM Reservation;
```

• 선택된 결과 정렬

- SELECT 문으로 선택한 결과를 ORDER BY 절을 사용하여 정렬할 수 있다.
- ORDER BY 절의 기본 설정은 오름차순이며, ASC/DESC키워드를 사용하여 직접 오름(내림)차순을 명시해도 된다

```
SELECT 필드이름
FROM 테이블이름
ORDER BY 필드이름
```

1.3. MySQL 자료타입

- MySQL에서 테이블을 정의할 때는 필드별로 저장할 수 있는 타입까지 명시해야 한다
- MySQL에서 사용 가능한 자료타입은 다양하며, 데이터의 형태와 크기에 따라 선택할 수 있다.
- 사용하고자 하는 데이터에 맞게 적절한 자료타입을 선택하는 것이 중요하다

• MySQL의 자료타입

- 정수형 자료타입:
  - ✓ INT 또는 INTEGER: 정수를 저장하는 데 사용
  - ✓ 다양한 크기의 정수를 저장할 수 있다
  - ✓ TINYINT, SMALLINT, MEDIUMINT, BIGINT
- 실수형 자료타입:
  - ✓ FLOAT: 단정도 부동소수점 수를 저장
  - ✓ DOUBLE: 배정도 부동소수점 수를 저장
- 문자열 자료타입:
  - ✓ VARCHAR: 가변 길이 문자열을 저장
  - ✓ CHAR: 고정 길이 문자열을 저장
  - ✓ TEXT: 긴 텍스트 데이터를 저장

- 날짜와 시간 자료타입:
  - ✓ DATE: 날짜를 저장
  - ✓ TIME: 시간을 저장
  - ✓ DATETIME: 날짜와 시간을 저장
  - ✓ TIMESTAMP: 시간을 저장하며, 자동 업데이트 가능
- 불리언 자료타입:
  - ✓ BOOLEAN, BOOL, TINYINT(1): 참 또는 거짓 값 저장
- 이진 자료타입:
  - ✓ BLOB: 이진 데이터를 저장하는 데 사용
- 열거형과 집합 자료타입:
  - ✓ ENUM: 가능한 값 목록 중 하나를 저장
  - ✓ SET: 가능한 값 목록 중 여러 값을 저장
- 기타 자료타입:
  - ✓ JSON: JSON 데이터를 저장
  - ✓ GEOMETRY: 지리 정보를 저장

1.4. MySQL 연산자와 함수[1/3] - 기본 연산자

● 산술 연산자

- ✓ 더하기 (+)
- ✓ 빼기 (-)
- ✓ 곱하기 (*)
- ✓ 나누기 (/)

● 비교 연산자

- ✓ 같음 (==)
- ✓ 다름 (!=)
- ✓ 크다 (>)
- ✓ 작다 (<)
- ✓ 크거나 같다 (>=)
- ✓ 작거나 같다 (<=)

● 논리 연산자

- ✓ AND
- ✓ OR
- ✓ NOT

● 문자열 연산자

- ✓ CONCAT() - 문자열 결합
- ✓ LIKE - 패턴 매칭
- ✓ IN - 값 목록에 속함 여부 확인

● NULL검사 연산자

- ✓ IS NULL - 값이 NULL인지 확인
- ✓ IS NOT NULL - 값이 NULL이 아닌지 확인

● 대입 연산자

- ✓ = - 변수에 값을 할당

1.4. MySQL 연산자와 함수[2/3] - MySQL 흐름제어 연산자

- MySQL은 프로그램의 순차적인 흐름을 제어해야 할 때 사용할 수 있는 다양한 연산자와 함수를 제공한다

CASE

1. CASE value
   WHEN [compare_value] THEN result
   [WHEN [compare_value] THEN result]
...
   [ELSE result]
END
2. CASE
   WHEN [condition] THEN result
   [WHEN [condition] THEN result] ...
   [ELSE result]
END

IF() /IFNULL()

- **IF()**
- expr1이 참이면 expr2를 반환, 거짓이면 expr3를 반환
  IF(expr1, expr2, expr3)
- **IFNULL()**
- **expr1의 값이 NULL이 아니면 expr1 그 자체를 반환하고, NULL이면 expr2를 반환**
  - IFNULL(expr1, expr2)

1.4. MySQL 연산자와 함수[3/3] - MySQL 제약조건(constraint)

- 제약 조건(constraint)이란 데이터의 무결성을 지키기 위해, 데이터를 입력받을 때 실행되는 검사 규칙을 의미합니다.
- 이러한 제약 조건은 CREATE 문으로 테이블을 생성할 때나 ALTER 문으로 필드를 추가할 때도 설정할 수도 있습니다

NOT NULL

- NULL값을 저장할 수 없다
- 이 제약 조건이 설정된 필드는 무조건 데이터를 가지고 있어야 한다

UNIQUE

- 해당 필드는 고유한 값을 가져야 한다
- 중복된 값 저장할 수 없다

PRIMARY KEY

- 해당 필드는 NOT NULL과 UNIQUE 제약 조건의 특징을 모두 가진다
- 테이블당 오직 하나의 필드에만 설정할 수 있다

FOREIGN KEY

- 이 조건을 설정한 필드는 외래 키라고 부르며, 한 테이블을 다른 테이블과 연결해주는 역할을 한다
- FOREIGN KEY 제약 조건을 설정할 때 참조되는 테이블의 필드는 반드시 UNIQUE나 PRIMARY KEY 제약 조건이 설정되어 있어야 한다

1.5. 다중테이블 연산(JOIN)[1/2]

- 데이터베이스의 여러 테이블 간에 정보를 검색, 조작 및 결합하는 프로세스를 의미한다.
- **기본 키 (Primary Key)**: 각 테이블은 하나 이상의 열을 기본 키로 지정할 수 있다.
  - 각 행을 고유하게 식별하는 역할
- **외래 키 (Foreign Key)**: 하나의 테이블에 있는 열이 다른 테이블의 기본 키와 연결되는 경우, 이를 외래 키로 지정할 수 있다. 외래 키를 사용하면 테이블 간의 관계를 설정할 수 있다.
- **JOIN**:
  - 다중 테이블 연산의 핵심은 JOIN 연산이다.
  - JOIN을 사용하여 두 개 이상의 테이블을 결합하고 서로 관련된 데이터를 추출할 수 있다.
  - 일반적인 JOIN 종류로는 INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN 등이 있다.
  - 각각의 JOIN은 특정 조건에 따라 데이터를 가져오는 방식이다

- **INNER JOIN 예제**

```
SELECT users.user_id, users.username, orders.order_id
FROM users
INNER JOIN orders ON users.user_id = orders.user_id;
```

두 테이블에서
공통된 데이터만을 반환

1.5. 다중테이블 연산(JOIN)[2/2]

- **LEFT JOIN 예제**

```
SELECT users.user_id, users.username, orders.order_id
FROM users
LEFT JOIN orders ON users.user_id = orders.user_id;
```

사용자가 주문을 한 경우
해당 주문이 나타나고
주문을 하지 않은 사용자는
NULL 값

- **WHERE 조건 추가**

```
SELECT users.user_id, users.username, orders.order_id
FROM users
INNER JOIN orders ON users.user_id = orders.user_id
WHERE users.username = '사용자명';
```

특정 사용자의 주문만 검색

1.6. MySQL INDEX[1/2] - INDEX 생성

- 인덱스(index)는 테이블에서 원하는 데이터를 쉽고 빠르게 찾기 위해 사용
- MySQL은 데이터를 검색할 때 첫 번째 필드부터 차례대로 테이블 전체를 검색하므로 테이블이 크면 클수록 데이터를 탐색하는 시간도 많이 늘어나게 된다.
- 인덱스를 사용하면 테이블 전체를 읽지 않아도 되므로, 검색과 질의에 대한 처리가 빠르게 이루어진다

● 인덱스 생성

```
CREATE INDEX 인덱스이름
ON 테이블이름 (필드이름1, 필드이름2, ...)
```

● UNIQUE INDEX 생성

- UNIQUE INDEX는 중복 값을 허용하지 않는 인덱스

```
CREATE UNIQUE INDEX 인덱스이름
ON 테이블이름 (필드이름1, 필드이름2, ...)
```

● 인덱스 정렬

- 인덱스를 생성할 때 인덱스에 포함되는 필드의 정렬 방식을 설정할 수 있다
- DESC 키워드 : 내림차순 /ASC 키워드 : 오름차순

```
1. CREATE INDEX 인덱스이름
   ON 테이블이름 (필드이름 DESC)
2. CREATE INDEX 인덱스이름
   ON 테이블이름 (필드이름 ASC)
```

1.6. MySQL INDEX[2/2] - INDEX 추가

- ALTER 문을 사용하여 테이블에 인덱스를 추가할 수 있다.
- MySQL에서 추가할 수 있는 인덱스의 타입
  1. 기본 인덱스
  2. UNIQUE INDEX
  3. FULLTEXT INDEX

● 기본 인덱스 추가

- 기본 인덱스에서 필드의 값은 같은 값이 여러 번 저장될 수 있으며, NULL 값을 가질 수도 있다
- 기본 인덱스를 추가하는 문법
  ALTER TABLE 테이블이름
  ADD INDEX 인덱스이름 (필드이름)

● UNIQUE INDEX 추가

- UNIQUE INDEX에서 필드의 값은 중복될 수 없으나, NULL 값을 가질 수는 있다
- UNIQUE INDEX 추가하는 문법
  ALTER TABLE 테이블이름
  ADD UNIQUE 인덱스이름 (필드이름)

● FULL TEXT INDEX 추가

- FULLTEXT INDEX는 일반적인 인덱스와는 달리 매우 빠르게 테이블의 모든 텍스트 필드를 검색한다
- UNIQUE INDEX 추가하는 문법
  ALTER TABLE 테이블이름
  ADD FULLTEXT INDEX이름 (필드이름)

JDBC

- Java Database Connectivity의 약자로, Java 언어를 사용하여 데이터베이스에 연결하고 상호 작용하기 위한 자바 API이다
- 웹 애플리케이션은 정보 저장이 필요할 때 주로 DB를 이용한다.
- 또한 웹 애플리케이션 서버(web application server – WAS)는 애플리케이션에 Connection Pooling 등 DB 의존적인 서비스를 제공하기 위하여 DB와 통신할 필요가 있다. 이러한 필요를 보다 체계적이고 효율적으로 충족시키기 위해 DB 클라이언트들과 DB 간 인터페이스를 정의한 것이 바로 Java Database Connectivity(JDBC) 표준이다.
- JDBC 표준은 애플리케이션의 DB Connection 사용 방법 및 SQL 작업 수행 방법에 대해 기술하고 관련 API를 제공한다