과목II. 1장 - SQL 기본

■ SQL 문장 종류

명령어 종류	명령어	설명
데이터 조작어 DML Data Manipulation Language	SELECT	데이터를 조회하거나 검색하기 위한 명령어
	INSERT UPDATE DELETE	데이터에 변형을 가하는 종류의 명령어
데이터 정의어 DDL Data Definition Language	CREATE ALTER DROP RENAME	데이터 구조를 정의하는데 사용하는 명령어
데이터 제어어 DCL Data Control Language	GRANT REVOKE	데이터베이스에 접근하고 객체들을 사용하도록 권한을 주고 회수하는 명령어
트랜잭션 제어어 TCL Transaction Control Language	COMMIT ROLLBACK	논리적인 작업 단위를 묶어서 DML에 의해 조작된 결과를 작업단위(트랜잭션) 별로 제어하는 명령어

■ 제약조건 종류

- PRIMARY KEY (기본키) : 테이블당 1개만 생성 가능 (UNIQUE & NOT NULL)
- UNIQUE KEY (고유키) : 테이블 내에서 중복되는 값이 없으며 NULL 값이 가능함
- FOREIGN KEY (외래키) : 테이블당 여러개 생성 가능하며 NULL 값이 가능함
- NOT NULL : NULL 입력 방지
- CHECK : 값을 어떤 특정 범위로 제한 가능 (데이터 무결성)
- DEFAULT : 값을 입력하지 않아도 기본으로 입력되는 값

■ NULL

- 아직 정의되지 않은 미지의 값
- 현재 데이터를 입력하지 못하는 경우
- 공백이나 숫자 0과는 전혀 다른 값
- 조건에 맞는 데이터가 없을 때의 공집합과도 다름

■ master table DELETE / UPDATE 옵션

- NO ACTION: 투플을 삭제 X
- CASCADE: 관련 투플을 함께 삭제
- SET NULL: 관련 투플의 외래키 값을 NULL로 변경
- SET DEFAULT : 관련 투플의 외래키 값을 미리 지정한 기본값으로 변경
- RESTRICT : child table에 PK 값이 없는 경우만 허용

■ child table INSERT 옵션

- NO ACTION: 참조 무결성을 위반하는 입력 X
- AUTOMATIC: master table에 PK가 없는 경우 master PK 생성 후 child 입력
- SET NULL: master table에 PK가 없는 경우 child 외부키를 NULL 값으로 처리
- SET DEFAULT: master table에 PK가 없는 경우 child 외부키를 지정된 기본값으로 입력
- DEPENDENT: master table에 PK가 존재할 때만 child 입력 허용

■ 테이블 생성의 주의사항

- 테이블명은 객체를 의미할 수 있는 이름 사용 (가능한 단수형)
- 테이블명은 다른 테이블의 이름과 중복되지 않아야 함
- 한 테이블 내에서는 칼럼명이 중복되게 지정될 수 없음

- 테이블 이름을 지정하고 각 칼럼들은 괄호로 묶어 지정함
- 각 칼럼들은 콤마로 구분됨
- 테이블 생성문의 끝은 항상 세미콜론으로 끝남
- 칼럼에 대해서는 다른 테이블까지 고려하여 데이터베이스 내에서는 일관성있게 사용 (데이터 표준화 관점)
- 칼럼 뒤에 데이터 유형은 꼭 지정되어야함
- 테이블명과 칼럼명은 반드시 문자로 시작해야 하고 벤더별로 길이에 대한 한계가 있음
- 벤더에서 사전에 정의한 예약어는 쓸 수 없음
- A-Z, a-z, 0-9, _, $, # 문자만 허용됨

■ 데이터 구조 변경

■ 테이블 칼럼에 대한 정의 변경

- Oracle

ALTER TABLE 테이블명

MODIFY (칼럼명1 데이터유형 [DEFAULT] [NOT NULL], 칼럼명2 데이터유형 …);

- SQL Server

ALTER TABLE 테이블명

ALTER COLUMN 칼럼명1 데이터유형 [DEFAULT] [NOT NULL];

■ 테이블 칼럼 삭제

ALTER TABLE 테이블명;

DROP COLUMN 칼럼명;

■ 테이블 이름 변경

RENAME 원래 테이블명 TO 바꿀 테이블명;

■ 테이블에 데이터 입력

INSERT INTO 테이블명 VALUES(전체 COLUMN에 넣을 VALUE_LIST);

INSERT INTO 테이블명(COLUMN_LIST) VALUES(COLUMN_LIST에 넣을 VALUE_LIST);

■ 입력된 데이터 수정

UPDATE 테이블명 SET 칼럼명 = 새로운 값;

■ 입력된 데이터 조회

SELECT [ALL/DISTINCT] COLUMN_LIST FROM COLUMN_LIST가 있는 해당 테이블명;

- ALL: DEFAULT 옵션. 중복된 데이터가 있어도 모두 출력
- DISTINCT: 중복된 데이터가 있는 경우 1건으로 처리해서 출력

■ DROP / TRUNCATE / DELETE 비교

DROP	TRUNCATE	DELETE
DDL	DDL	DML
Rollback 불가능	Rollback 불가능	commit 이전 rollback 가능
Auto Commit	Auto Commit	사용자 Commit
테이블이 사용했던 Storage 모두 Release	테이블이 사용했던 Storage 중 최초 테이블 생성 시 할당된 Storage만 남기고 Release	데이터를 모두 Delete 해도 사용했던 Storage는 Release 되지 않음
로그를 남기지 않음	로그를 남기지 않음	로그를 남김
테이블의 정의 자체를 완전히 삭제	테이블을 최초 생성된 초기 상태로 만듬	데이터만 삭제

■ 트랜잭션 특성

트랜잭션: 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위

- 데이터베이스 시스템에서 병행 제어 및 회복 작업 시 처리되는 작업의 논리적 단위
- 사용자가 시스템에 대한 서비스 요구 시 시스템이 응답하기 위한 상태 변환 과정의 작업단위
- 하나의 트랜잭션은 Commit 되거나 Rollback 됨
- Commit : 데이터에 대한 변경사항을 데이터베이스에 영구적으로 반영하는 트랜잭션의 종료를 위한 명령어
- Rollback : 데이터에 대한 변경사항을 모두 폐기하고 변경 전의 상태로 되돌리는 트랜잭션의 종료를 위한 명령어

특성	설명
원자성	트랜잭션에서 정의된 연산들은 모두 성공적으로 실행되던지 아니면 전혀 실행되지 않은 상태로 남아 있어야 함.
일관성	트랜잭션 실행 전의 데이터베이스 내용이 잘못되어 있지 않다면 트랜잭션 실행 이후에도 데이터베이스의 내용에 잘못이 있으면 안됨.
고립성	트랜잭션이 실행되는 도중에 다른 트랜잭션의 영향을 받아 잘못된 결과를 만들어서는 안됨.
지속성	트랜잭션이 성공적으로 수행되면 그 트랜잭션이 갱신한 데이터베이스의 내용은 영구적으로 저장됨.

■ 트랜잭션에 대한 격리성이 낮은 경우 발생할 수 있는 문제점

- Dirty Read : 다른 트랜잭션에 의해 수정되었지만 아직 Commit 되지 않은 데이터를 읽는 것
- Non-Repeatable Read ; 한 트랜잭션 내에서 같은 쿼리를 두 번 수행했는데, 그 사이에 다른 트랜잭션이 값을 수정 또는 삭제해서 두 쿼리 결과가 다르게 나타나는 현상
- Phantom Read : 한 트랜잭션 내에서 같은 쿼리를 두 번 수행했는데, 첫 쿼리에서 없던 유령 레코드가 두 번째 쿼리에서 나타나는 현상

■ 롤백 ROLLBACK

- 테이블 내 입력한 데이터나 수정 또는 삭제한 데이터에 대하여 COMMIT 이전에는 변경사항 취소 가능
- 데이터 변경 사항이 취소되어 데이터의 이전 상태로 복구됨
- 관련된 행에 대한 잠금이 풀리고 다른 사용자들이 데이터 변경을 할 수 있게 됨
- ORACLE에서는 DDL 문장 수행 후 자동으로 COMMIT 수행함
- SQL Server에서는 DDL 문장 수행후 자동으로 COMMIT 수행 안함
- SQL Server에서는 CREATE TABLE 문장도 트랜잭션의 범주에 포함됨

■ TRANSACTION 시작 및 종료

BEGIN TRANSACTION : 트랜잭션 시작

COMMIT TRANSACTION / ROLLBACK TRANSACTION ; 트랜잭션 종료

* ROLLBACK 구문을 만나면 COMMIT 이후 최초의 BEGIN TRANSACTION 시점까지 모두 ROLLBACK 수행됨

■ SAVEPOINT

- SAVEPOINT를 정의하면 현 시점에서 SAVEPOINT까지 트랜잭션의 일부만 롤백할 수 있음
- ORACLE

```
SAVEPOINT SVPT1;
...
ROLLBACK TO SVPT1;
```

- SQL Server

```
SAVE TRANSACTION SVTR1;
...
ROLLBACK TRANSACTION SVTR1;
```

■ 연산자의 우선순위

① 괄호로 묶은 연산 ② 부정 연산자 (NOT) ③ 비교 연산자 ④ 논리 연산자 (AND, OR 순서)

■ NULL의 연산

- NULL 값과의 사칙연산은 NULL 값 리턴
- NULL 값과의 비교연산은 FALSE 리턴
- 특정 값보다 크다, 작다라고 표현할 수 없음

* NULL 값을 조건절에서 사용하는 경우 IS NULL, IS NOT NULL 키워드 사용함

■ 함수

- 벤더에서 제공하는 내장 함수와 사용자가 정의하는 함수로 나눌 수 있음
- 내장 함수는 단일행 함수와 다중행 함수로 나눌 수 있음 (함수의 입력 행수에 따라 구분)
- 단일행 함수는 SELECT, WHERE, ORDER BY, UPDATE의 SET절에 사용 가능
- 다중행 함수는 집계함수, 그룹함수, 윈도우 함수로 구분됨.
- 다중행 함수도 단일행 함수와 동일하게 단일 값만을 반환함

■ NULL 관련 단일행 함수

일반형 함수	함수 설명
NVL(표현식1, 표현식2) / ISNULL(표현식1, 표현식2)	표현식1의 결과값이 NULL이면 표현식2의 값을 출력함. 단, 표현식1과 표현식2의 결과 데이터 타입이 같아야 함.
NULLIF(표현식1, 표현식2)	표현식1이 표현식2와 같으면 NULL을, 같지 않으면 표현식1을 리턴함.
COALESCE(표현식1, 표현식2, …)	임의의 개수 표현식에서 NULL이 아닌 최초의 표현식을 나타냄. 모든 표현식이 NULL이라면 NULL을 리턴함.

■ SEARCHED/SIMPLE _CASE_EXPRESSION

[SEARCHED_CASE_EXPRESSION] [SIMPLE_CASE_EXPRESSION]

CASE WHEN LOC = 'NY' THEN 'EAST' CASE LOC WHEN 'NY' THEN 'EAST'

■ GROUP BY 절과 HAVING 절의 특성

- GROUP BY 절을 통해 소그룹별 기준을 정한 후, SELECT 절에 집계함수를 사용함
- 집계 함수의 통계 정보는 NULL 값을 가진 행을 제외하고 수행함
- GROUP BY 절에서는 SELECT 절과는 달리 ALIAS명을 사용할 수 없음
- 집계 함수는 WHERE 절에는 올 수 없음
- WHERE 절은 전체 데이터를 GROUP으로 나누기 전에 행들을 미리 제거함
- HAVING 절은 GROUP BY 절의 기준 항목이나 소그룹의 집계 함수를 이용한 조건을 표시할 수 있음
- GROUP BY 절에 의한 소그룹별로 만들어진 집계 데이터 중, HAVING 절에서 제한 조건을 두어 조건을 만족하는 내용만 출력함
- HAVING 절은 일반적으로 GROUP BY 절 뒤에 위치함

■ ORDER BY 절 특징

- 오름차순 정렬 기본
- 숫자형 데이터 타입은 작은 값부터 출력
- 날짜형 데이터 타입은 빠른 값부터 출력
- Oracle : NULL 값을 가장 큰 값으로 간주
- SQL Server : NULL 값을 가장 작은 값으로 간주
- ORDER BY 절에 칼럼명 대신 Alias명이나 칼럼 순서를 나타내는 정수를 혼용하여 사용 가능

* GROUP BY 절을 사용하는 경우 ORDER BY 절에 집계 함수를 사용할 수 있음

■ SELECT 문장 실행 순서

① 테이블 참조 (FROM)

② 발췌 대상 데이터가 아닌 것은 제거 (WHERE)

③ 행들을 소그룹화 (GROUP BY)

④ 그룹핑된 값의 조건에 맞는 것만을 출력 (HAVING)

⑤ 데이터 값을 출력/계산 (SELECT)

⑥ 데이터 정렬 (ORDER BY)

■ TOP(n) 예제

: 팀별성적 테이블에서 승리 건수가 높은 2팀을 내림차순으로 출력하는데 승리 건수가 같은 팀이 있다면 같이 출력함

```
SELECT TOP(2) WITH TIES 팀명, 승리건수
FROM 팀별성적
ORDER BY 승리건수 DESC;
```

* 함께 출력하도록 하는 WITH TIES 옵션을 ORDER BY 절과 함께 사용함

■ Join

- 일반적으로 PK와 FK 값의 연관성에 의해 성립됨
- EQUI Join은 테이블의 칼럼 값들이 정확하게 일치하는 경우에 사용되는 방법
- EQUI Join은 '=' 연산자에 의해서만 수행되며, 그 이외의 비교연산자를 사용하는 경우에는 모두 Non EQUI Join

* EQUI Join 문장

```
SELECT 테이블1.칼럼명, 테이블2.칼럼명, ...
FROM 테이블1, 테이블2
WHERE 테이블1.칼럼명1 = 테이블2.칼럼명2;
```

■ 데이터 정의

- 테이블 생성 : CREATE TABLE
- 테이블 변경 : ALTER TABLE
- 테이블 삭제 : DROP TABLE

■ 데이터 조작

- 기본 검색 : SELECT FROM
- 조건 검색 : SELECT FROM WHERE
- 정렬 검색 : SELECT FROM WHERE ORDER BY
- 집계함수를 이용한 검색 : COUNT, SUM, AVG, MAX, MIN
- 그룹 검색 : SELECT FROM WHERE GROUP BY HAVING (ORDER BY)
- 조인 검색 : 여러 테이블을 연결하여 검색
- 부속 질의문 검색 : SELECT 문 안에 또 다른 SELECT 문을 포함
- 데이터 삽입 : INSERT
- 데이터 수정 : UPDATE
- 데이터 삭제 : DELETE