**[ 들 어 가 기 ]**

1. **오라클(Oracle) 및 SQL 소개**

**1.1 DBMS?**

DBMS : DataBase Management System

처리할 데이터 양이 많아짐

데이터를 적절하고 효율적으로 관리할 필요가 생김

결국 데이터를 관리하기 위한 체계적인 시스템이 필요해짐(DBMS)

* **관계형 데이터베이스(Relation-Database)**

가장 대표적인 DB 시스템

2차원 구조 모델을 기반으로 한다.

데이터 무결성(Integrity), 트랜잭션 처리 등 DBMS 의 기본적인 기능면에서 뛰어난 성능을 보여준다.

질의어(Query Language)를 사용한 데이터 접근 방법이다.

* **객체지향형 데이터베이스**

좀 더 복잡한 데이터(사용자 정의 데이터, 오디오 비디오 등의 멀티미디어 데이터 등)에 대한 저장 관리의 필요성으로 발생된 모델

관계형 DB가 가지고 있지 않는 기능들

사용자 정의 타입을 지원함

비정형 복합 정보 모델이 가능

정 형 : 날짜, 문자, 숫자

비정형 : 멀티미디어, 이메일, 문서 등

객체들 사이의 참조 구조를 이용한 접근 가능

객체지향 프로그래밍 언어들과 스키마 구조가 유사

단점 : DB의 기본 기능인 트랜젝션 처리, 백업 및 복구 등의 기능이 기존 관계형 DB에 훨씬 미치지 못하는 성능을 보임

* **객체관계형(Object-relational Database) 데이터베이스 : Oracle이 여기에 해당**

위에서 언급한 두 가지 DB의 장점만을 살린 모델

특징 : 사용자 정의 타입 지원

참조 타입 지원(다른 레코드의 정보를 접근하는 것: 조인(관계형)/내비게이션 기반 접근(객체관계형))

중첩 테이블 지원

대단위 객체의 저장, 추출 가능

객체 간의 상속 관계를 지원함

**1.2 Oracle?**

DBMS의 일종

2001년 : Oracle 9i 출시 ( internet )

2003년 : Oracle 10g 출시 ( Grid Computing )

그리드 컨트롤(Grid Control : 여러 대의 서버를 하나의 논리적 서버로 사용) 환경 지원

프로비저닝(provisioning) 요청된 컴퓨터 자원들을 배치하고 제공

부하 관리(Load balancing) : 필요한 곳에 적잘한 작업을 분배

자동화(Automation) : 여러 대의 서버를 하나처럼 운영할 경우 필요한 관리들을 자동화

* 여러 대의 물리적 서버를 한 대의 논리적 서버로 구성하여 다운타임 최소화, 고가용성을 위한 기능을 지원

2007년 7월 : Oracle 11g 출시

**1.3 SQL(Structured Query Langauge)?**

구조화된 언어이다(데이터베이스에서 데이터를 조회, 입력, 수정, 삭제, 저장하는데 사용되는 언어)

- DB에는 데이터들이 물리적인 파일형태로 저장되어 있고, DBMS는 데이터와 관계된 일련의 작

업을 처리한다. 특히 데이터의 조회, 저장, 삭제와 같은 작업은 SQL을 통해 처리하게 된다.

- 구조화(선언적(declarative) 언어 : 틀과 패턴이 있다. 즉 일반 언어처럼 처리 과정을 일일이 기

술 하지 않는다.

- 표준 SQL 문법이 존재한다. DBMS의 종류에 얽매이지 않고 사용할 수 있는 장점이 있다.

1. **SQL의 이해**

**2.1 SQL 역할**

역할에 따라 DML과 DDL로 구분된다.

- DML(데이터 조작어, Data Manipulation Langue)

데이터 추출, 저장, 삭제 등 데이터를 조작하는 것에 관한 것

- DDL(데이터 정의어, Data Definition Language)

데이터가 저장되는 테이블이나 각종 객체들을 생성하는 역할

\*) DCL(데이터 제어어, Data Control Language) : DB에 있는 데이터에 접근을 제어하는 언어

**데이터 조작하기(DML)**

- 데이터 조회 : SELECT

- 데이터 삭제 : DELETE

- 데이터 저장 : INSERT(신규 데이터 입력) / UPDATE(기존 데이터 수정)

* SELECT : 데이터 조회 시 사용

컬럼(Column) : 엑셀 시트에서 가로로 A, B, C 등으로 구성된 셀들을 의미함

로우(Row) : 테이블의 세로

|  |
| --- |
| **SELECT** 컬럼1, 컬럼2, …  **FROM** 테이블1, 테이블2, …  **WHERE** 조건들; |

\*) WHERE 절은 필터 역할을 수행하며 생략 가능

\*) SELECT 와 FROM은 생략 불가

\*) EMPLOYEES라는 테이블에는 직원들의 이름(first\_name / last\_name), 입사일(hire\_date) 등의 정보가 있다.

2006년 이후에 입사한 직원들 이름을 조회하는 SQL문을 작성한다면?

SELECT last\_name, first\_name

FROM employees

WHERE hire\_date >= ‘2006-01-01’ ;

* INSERT : 테이블에 새로운 데이터를 넣을 때 사용, 두 가지 형식이 있다.

|  |
| --- |
| **INSERT INTO** 테이블명1 (컬럼1\_1, 컬럼2\_2), …)  **VALUES** (컬럼1\_1 값, 컬럼1\_2 값, … );  🡺 컬럼 리스트와 컬럼 값의 순서 및 개수 일치 |
| **INSERT INTO** 테이블명1  **VALUES** (컬럼1\_1 값, 컬럼1\_2값, … )  🡺 테이블의 모든 컬럼에 값 입력 |

제3의 테이블에 있는 데이터를 선택해서 새로운 데이터를 넣는 형태

|  |
| --- |
| **INSERT INTO** 테이블명1 (컬럼1, 컬럼2, …)  **SELECT** 컬럼2\_1, 컬럼2\_2, …  **FROM** 테이블2  **WHERE** … ; |

\*) INSERT 문장은 하나이지만 실제 입력되는 값은 1로우 이상이 가능하다.

* UPDATE : 기존 테이블에 저장되어 있는 데이터를 새로운 값으로 변경할 때 사용

|  |
| --- |
| **UPDATE** 테이블명  **SET** 변경 컬럼1 = 변경하고자 하는 값,  변경 컬럼2 = 변경하고자 하는 값,  …  **WHERE** 조건절 … ; |

\*) WHERE 절 생략 시 데이블 데이터 전체 변경

\*) WHERE 절 명시해서 조건에 일치하는 데이터만 변경 가능

* DELETE : 데이터를 삭제할 때 사용

|  |
| --- |
| **DELETE [FROM]** 테이블  **WHERE** 조건… ; |

\*) 테이블의 로우를 제거함

\*) FROM은 생략 가능

\*) WHERE문을 통해 특정 조건에 맞는 데이터만 삭제 가능

\*) WHERE문 삭제 시 테이블 전체 데이터가 삭제 됨

* COMMIT : 확정 개념

만약 중요한 데이터를 실수로 삭제나 수정했다면?

- INSERT, UPDATE, DELETE 문을 실행하더라도 곧바로 데이터가 변경되지 않는다.

변경된 데이터는 오라클 메모리 상에 존재하고 COMMIT 문장을 실행했을 경우 실제 DB에 존재하는 데이터가 변경되게 된다.

COMMIT : 데이터 변경작업(INSERT, UPDATE, DELETE)을 한 뒤 COMMIT을 실행하면 변경된 데이터들이 데이터 파일에 반영된다.

|  |
| --- |
| **COMMIT [ WORK] [TO SAVEPOINT** savepoint\_name**]**; |

\*) [ ] 생략 가능, ‘COMMIT’ 이나 ‘COMMIT WORK’ 는 동일한 효과를 가진다.

\*) SAVEPOINT는 특정 시점까지 작업한 내용을 반영할 때 사용

이전 COMMIT이나 ROLLBACK 을 실행한 시점부터 현재 COMMIT을 실행하는 동안에 이루어진 내용을 반영함

* ROLLBACK : 취소 개념

COMMIT과 반대 개념, 변경된 데이터를 변경 전 상태로 되돌리는 역할

ROLLBACK문을 실행하면 이전 ROLLBACK이나 COMMIT이 실행된 시점 이후에 변경된 사항은 원상태로 되돌아감

|  |
| --- |
| **ROLLBACK [WORK] [TO SAVEPOINT** savepoint\_name**];** |

ex) ROLLBACK과 COMMIT 예

|  |  |
| --- | --- |
| (A) | INSERT INTO temp\_emp (id, name) VALUES (1, ‘홍길동’) ;  SELECT name FROM temp\_emp  WHERE id = 1; ------------------------------------------------🡪 (1) ‘홍길동’ |
| (B) | COMMIT; |
| (C) | UPDATE temp\_emp  SET name = ‘김길동’;  WHERE id = 1; |
| (D) | INSERT INTO temp\_emp (id, name) VALUES(2, ‘공길동’); |
| (E) | COMMIT;  SELECT name FROM temp\_emp  WHERE id = 1; ---------------------------------------------🡪 (2) ‘김길동’ |
| (F) | DELETE temp\_emp; |
| (G) | ROLLBACK;  SELECT name FROM temp\_emp  WHERE id = 1; --------------------------------------------🡪 (3) ‘김길동’ |

1) COMMIT 실행 이전 시점으로 데이터의 최종 반영이 되지 않았지만, 동일한 트랜젝션 상에 있으므로 바로 전에 입력한 ‘홍길동’ 이라는 이름을 가져온다.

실제 이 이름을 가져오는 것은 데이터 파일이 아닌 오라클 메모리 상에서 가져오게 된다.

2) id = 1인 사람의 이름을 변경하고 COMMIT을 수행한 이후 조회했으므로 변경된 이름을 가져온다.

3) DELETE 문을 실행해서 모든 데이터를 삭제 했지만 ROLLBACK을 수행했으므로 이전 COMMIT이 실행된 (E)번 이후에 수행된 작업들을 모두 취소했으므로 테이블 상에 김길동이 존재하게 된다.

\*) 트랜젝션(transaction)

- COMMIT 과 ROLLBACK 은 데이터 베이스의 트랜잭션 처리를 담당한다.

- 트랜젝션이란 ‘거래’의 의미이다.

Ex) 계좌이체(A계좌에서 출금 B계좌로 입급)

계좌이체를 하나의 트랜젝션이라 함

만약, 2가지 일이 정상적으로 종료되면 COMMIT, 중간에 오류가 발생하면 ROLLBACK을 함

**객체 조작하기(DDL)**

데이터 베이스 객체(데이터베이스, 테이블, 인덱스, 뷰, 트리거 등) 들을 생성하는데 사용되는SQL 문

- CREATE, DROP, ALTER, TRUNCATE

* CREATE : 데이터 베이스 객체를 생성할 때 사용

|  |
| --- |
| CREATE 객체종류 객체명 … ; |

\*) 생성 객체에 따라 그 용법이 다른다.

\*) 기본 구조는 위와 동일

Ex) CREATE TABLE 테이블명…

CREATE INDEX 인덱스명…

* DROP : 이미 생성된 객체들을 삭제할 때 사용

DELETE는 테이블 상의 데이터 삭제 시 사용, DROP은 객체 자체를 데이터베이스에서 영구히 없앨 때 사용

|  |
| --- |
| DROP 객체종류 객체명 … ‘ |

\*) 객체에 따라 그 용법이 다른다.

\*) 기본 구조는 위와 동일

* ALTER : 기존 생성된 객체를 변경할 때 사용

Ex) 테이블을 생성해서 사용하는 중 컬럼을 추가 하는 등 테이블 구조를 변경해야 할 경우

기본 골격은 CREATE나 DROP과 비슷함

* TRUNCATE : DELETE 문과 동일한 역할을 수행함

즉, 테이블에 있는 데이터를 삭제할 때 사용

차이점 : TRUNCATE 문을 사용하여 데이터를 삭제할 경우 ROLLBACK을 사용하여 데이터를

복구할 수 없다.

Why? TRUNCATE 가 실행되면 자동 COMMIT 이 된다.

- DELETE 보다 수행 속도가 빠르며 수행 비용이 적다.

- 테이블에 있는 인덱스나 트리거에 영향을 주지 않고 데이터만 삭제한다.

- DROP은 테이블에 있는 데이터를 삭제함은 물론 테이블 구조까지 소멸시키는 반면 DELETE는 메모리 상에 존재하는 테이블에 데이터를 삭제한다. TRUNCATE는 테이블 구조는 그대로 두고 메모리뿐만 아니라 데이터 파일에 있는 데이터까지 삭제한다.

**데이터 접근 제어(DCL)**

일반적으로 SQL문장은 DDL과 DML로 구분함, 이 두 가지에 DCL을 추가하기도 함

- GRANT : 접근제어나 어떤 작업을 허용하는 권한을 주는 역할

- REVOKE : 허용된 권한을 없애는 역할

**2.1 SQL 문장의 구성 요소들**

- 예약어 / 연산자 / 의사컬럼 / 함수 / 표현식 / 조건

* 예약어 : 오라클 DBMS에서 특정 용도로 사용되는 단어
* 연산자 : 데이터 항목들을 조작해서 결과를 산출해내는 역할 수행
* 단항 혹은 일항(unary) 연산자
* 이항(binary) 연산자

|  |  |  |  |
| --- | --- | --- | --- |
| **연산자 구분** | **연산자** | **용도** | **사용 예** |
| 수식 연산자 | + - | 양수, 음수 표시 | SELECT \* FROM order\_items  WHERE quantity = -1; |
| + - | 덧셈, 뺼셈 | SELECT hire\_date  FROM employees  WHERE SYSDATE -hire\_date  > 365; |
| \* / | 곱셈, 나눗셈 | UPDATE employees  SET salary = salary \*1.1; |
| 문자 연산자 | || | 문자들을 합침 | SELECT ‘홍’ || ’길동’  FROM dual; |
| 논리 연산자 | >, <, =, >=, <= | 논리연산 수행 | SELECT \*  FROM employees  WHERE salary > 1000 |
| 계층형 쿼리  연산자 | PRIOR | 계층형 쿼리 |  |
| CONNECT\_BY\_ROOT | 계층형 쿼리 |  |
| Set 연산자 | UNION | 두 쿼리 결과에서 합집합에 속하는 로우만 선택 |  |
| UNION ALL | 두 커리 결과 모두를 선택, 중복 로우까지 선택 |  |
| INTERSECT | 두 쿼리 결과에서 교집합에 속하는 로우만 선택 |  |
| MINUS | 두 쿼리 결과에서 차집합에 속하는 로우만 선택 |  |

* 의사컬럼(Paeudocolumns) : 테이블에 있는 일반적인 컬럼 처럼 행동하기는 하지만 실제로 테이블에 저장되어 있지 않는 컬럼

- 의사컬럼값으로 SELECT 가능

- 의사컬럼값으로 INSERT, UPDATE, DELETE는 불가

- 자주 사용하는 의사컬럼

ROWNUM : 쿼리의 결과로 나오게 되는 각각의 로우들에 대한 순서값을 가리키는 의사컬

럼, 주로 특정 개수나 그 이하의 로우를 선택할 때 사용됨

Ex) SELECT employee\_id, last\_name, ROWNUM

FROM EMPLOYEES

WHERE ROWNUM <= 10

ROWID : 테이블에 저장된 각각의 로우들이 저장된 주소값을 가진 의사컬럼

보통 로우들을 식별하는 식별 ID로 사용가능

* 함수 : 특정 연산을 수행하고 그 결과를 돌려주는 기능을 수행하는 객체

일반 프로그래밍 언어들에서 사용되는 함수와 동일

- 오라클 자체에서 지원되는 내장 함수

일반적으로 SQL 함수라 하면 내장함수를 의미함

- 사용자가 직접 작성하여 사용하는 사용자 정의 함수로 구분

* 표현식 : 하나 이상의 상수나 컬럼의 값, 연산자, SQL 함수들을 조합하여 하나의 값을 평가하는 역할 , SQL에서 표현식을 사용할 수 있는 곳은 아래와 같다.

- SELECT 문장에서 select 리스트

- WHERE 와 HAVING 절에서의 조건절

- ORDER BY, CONNECT BY, START WHIT 절

- INSERT 문장의 VALUES 절

- UPDATE 문장의 SET 절

Ex)

- 상수와 연산자가 결합된 유형 : 2 \* 5

- 함수와 의사컬럼이 결합된 유형 : TO\_CHAR(SYSDATE)

- 함수와 컬럼값이 결합된 유형 : NVL(last\_name, ‘HONG’)

- 함수와 컬럼값이 결합된 유형 : DECODE(dept\_gubun, ‘1’, ‘상위부서’, ‘2’, ‘하위부서’)

* 조건(condition) : 하나 혹은 그 이상의 표현식과 논리 연산자가 결합되어 TRUE, FALSE, UNKNOWN의 값을 반환하는 형태

Ex) SELECT name FROM our\_class WHERE **length >= 180**;