7월 22일 SQL

JAVA와 DB를 연결하는 방법

1. JDBC(ODBC): Java Database Connectivity 사용
2. MyBas(iBatis) : Spring에서 사용
3. JPA: Java Persistence API : Spring Boot에서 사용

DB를 사용하는 이유

DB는 여러 업무에 사용할 목적으로 장치 내에 체계적으로 분류/정리하여 모아 놓은 데이터의 집합.

기존에는 파일을 사용했지만, 업무의 범위가 넓어지고 연계되는 일이 많아짐에 따라 여러 사용자가 데이터를 사용하면서 서로 다른 버전을 가지는 경우가 생김.

이로 인해 파일이 중복되거나, 최신 버전이 무엇인지 알 수 없게 됨

Database의 특징

1) Database는 통합된 데이터(자료 중복 배제)

--데이터의 중복을 최소화하여 효율성을 증대

-Database는 공용 데이터(여러 응용 시스템이 공용으로 소유 및 유지)

--사용자/시스템이 공동으로 소유/사용하는 데이터의 집합

-Database는 운영 데이터(존재 목적이나 유용성 면에서 필수적인 데이터)

--고유 기능 수행을 위해 필요한 데이터의 집합

-Database는 저장된 데이터(컴퓨터가 접근할 수 있는 저장매체에 저장된 데이터)

--컴퓨터에 의해 접근 가능한 매체에 저장

Schema

121 1)정의

122 -Database를 구성하는 데이터 객체, 이들의 성질, 이들 간에 존재하는 관계, 그리고

데이터의 조작 또는 이들 데이터 값들이 갖는 제약조건에 관한 정의를 총칭하는 용어

123 -Database의 구조와 제약 조건에 대한 명세(Specification)를 기술한 것

124 -현실세계의 특정한 한 부분의 표현으로서 특정 데이터 모델을 이용하여 만듦

125 -시간에 따라 불변

126 -Schema는 데이터의 구조적 특성을 의미하며 인스턴스에 의해 규정됨

SQL(Structured Query Language)

1. Database 에 접속하여 테이블 등 객체에 데이터를 입력, 수정, 삭제, 조회(CRUD)하기 위해 사용하는 언어

2. Database Server 와 통신하기 위한 명령 언어

3. RDBMS 를 사용하기 위해 ANSI 에서 책정한 표준언어

4. SEQUEL(Structured English QUEry Language)

5. DBMS 제품별로 SQL 에 대한 추가 및 확장

6. 비 절차식 언어 // 순차적으로 실행할 때, 검증하며 실행하는 것은 X, 오류가 발생해도 다음 구문 실행

7. 1970년 초에 IBM 의 SYSTEM R 프로젝트를 통해 개발

SQL 종류

1. Data Manipulation Language (CRUD 주요 기능, 개발자의 영역)

SELECT, INSERT, UPDATE, DELETE

2. Data Definition Language (DB, Table 등을 생성, DBA의 역할)

CREATE, ALTER, DROP

3. Data Control Language (DB에 대한 유지보수, 권한, 예외처리 등 DBA의 역할)

GRANT, REVOKE, COMMIT, ROLLBACK, SAVEPOINT

시스템의 영향을 주는 범위에 따라 아래 2가지로도 구분할 수 있음

// DQL, Data Query Language : 데이터 조회만

// TCL, Transaction Control language : 데이터 생성, 수정, 삭제

INSERT, UPDATE, DELETE, COMMIT, ROLLBACK

SQL문 작성방법

1. 특별히 대소문자를 구별하지는 않지만 유지보수/가독성을 위해 대문자 또는 소문자로 작성하

라. - **KEYWORD는 대문자, 개체(테이블, 필드 등)는 소문자**

2. SQL의 각 절은 되도록 다른 행에 작성하여 읽기 쉽고 편집하기 쉽도록 구분하라.

3. TAB과 들여쓰기를 사용하여 좀 더 읽기 쉬운 SQL로 작성하라.

4. 띄워쓰기 법칙을 지켜서 SQL문을 통일시켜라.

5. 컬럼명과 TABLE명은 SQL 절과 구분되도록 작성하라.

실습 테이블 소개

1. dept : 부서 정보를 저장. Primary Key는 deptno column.

2. emp : 사원 정보를 저장. Primary Key는 empno column.

- emp table의 deptno column은 dept table의 primary key인 deptno column을 참조하는 Foreign Key

3. salgrade : 급여 등급 정보를 저장. 급여의 범위에 따라 1,2,3... 등급을 정하고 있으며 이 table에는 Primary Key, Foreign Key가 없다.

4. 특정 사원의 부서명이나 부서의 위치 정보를 알기 위해서는 Foreign Key인 deptno column 값에 해당하는 dept table 정보를 검색해야 한다.

5. 특정 사원의 급여가 어떤 등급인지 확인하려면 SAL column 값이 salgrade table 의 losal 과 hisal column 값 사이에 있는 grade를 찾아야 한다.

각 테이블 정보를 확인해 보자.

DESC는 테이블의 필드 정보를 보여주는 명령어

mysql> DESC dept;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Deptno | Tinyint | No | Pri | Null |  |
| Dname | Varchar(14) | Yes |  | Null |  |
| Ioc | Varchar(13) | Yes |  | Null |  |
|  |  |  |  |  |  |

+--------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+--------+-------------+------+-----+---------+-------+

| deptno | tinyint | NO | PRI | NULL | |

| dname | varchar(14) | YES | | NULL | |

| loc | varchar(13) | YES | | NULL | |

+--------+-------------+------+-----+---------+-------+

mysql> DESC emp;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| empno | smallint | NO | PRI | NULL |  |
| ename | varchar(10) | YES |  | NULL |  |
| job | varchar(9) | YES |  | NULL |  |
| mgr | smallint | YES |  | NULL |  |
| hiredate | date | YES |  | NULL |  |
| sal | float(7,2) | YES |  | NULL |  |
| comm | float(7,2) | YES |  | NULL |  |
| deptno | tinyint | YES | MUL | NULL |  |

mysql> DESC salgrade;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| grade | tinyint(1) | YES |  | NULL |  |
| losal | smallint | YES |  | NULL |  |
| hisal | smallint | YES |  | NULL |  |

SELECT의 기능 및 사용 방법

SELECT의 기능

1. Selection : 조건검색, Row에 대한 필터링  
2. Projection : column에 대한 필터링  
3. Join : 여러 테이블에서의 검색

SELECT Syntax

SELECT [DISTINCT | ALL] { \* | column1, column2 [AS [alias]] | expr}  
FROM 테이블명  
WHERE 조건식 (  
ORDER BY column [ASC | DESC];

동작 순서

1. 가장 먼저 FROM을 사용해 테이블을 선택한다.

2. WHERE 절에 가져온 정보에 조건을 부여한다.

3. 원하는 정보를 SELECT한다.

4. 마지막에 ORDER BY로 정렬한다.

세부 설명

FROM 절 다음에는 조회할 테이블 이름을 적는다.

SELECT 절 다음에 질의하고 싶은 칼럼을 차례대로 나열한다.

여러 개의 칼럼을 가져올 때는 쉼표(,)로 구분한다.  
이때, 테이블 구조와 상관 없이 SELECT에서 정의한 칼럼 순으로 가져온다.

모든 칼럼을 조회할 때는 \*을 사용한다.

DISTINCT: 중복 제거 / ALL: 중복 허용

alias : 칼럼에 대한 별칭 사용, 다른 테이블에서 참조 시 별칭으로 호출 가능.

expr : SQL 함수를 사용하거나, 수학 연산을 포함한 표현식

WHERE 절 다음에는 칼럼에 대한 조건을 지정한다.

# DB에서 NULL 처리

## Null value

1. NULL 이란?
   * 1. 특정 행, 특정 열에 대한 아직 값을 알 수 없는 상태, 의미가 없는 상태를 표현
     2. 이용할 수 없거나, 지정되지 않거나, 알 수 없거나, 적용할 수 없는 값
     3. 아직 정의되지 않은 미지의 값
     4. 현재 데이터를 입력하지 못하는 경우
2. **NULL(ASCII 0)은 0(zero, ASCII 48) 또는 공백(blank, ASCII 32)과 다르다.**
3. **NULL 값을 포함한 산술 연산 식의 결과는 언제나 NULL**이다. -> **null 처리가 필요한 이유**
4. NOT NULL 또는 Primary Key 제약조건이 걸린 칼럼에서는 NULL VALUE가 나타날 수 없다.
5. NULL인 칼럼은 Length가 0이므로 data를 위한 물리적 공간을 차지 하지 않는다.

## IFNULL Function

1. NULL 값을 어떤 특정한 값으로 치환할 때 사용
2. 치환할 수 있는 값의 형태는 숫자형, 문자형, 날자형 모두 가능
3. IFNULL(expr1, expr2)

- 만약 expr1이 NULL이라면 expr2를 대체 출력

- expr1값이 NULL이 아니라면 expr1 값을 그대로 사용

SELECT IFNULL(1, 0); --> 1

SELECT IFNULL(NULL, 10); --> 10

SELECT IFNULL(1/0, 10); --> 10 // 1/0은 저장할 수 없어 Null

126 REM Alias 별칭

127 1. column header 에 별칭을 부여 할 수 있다.

128 2. SELECT 절에 expression 을 사용할 때 도움이 된다.

129 3. 별칭은 열 이름 바로 뒤에 기술한다. AS는 생략이 가능하지만 열 이름과 별칭 사이에 AS를 명시하여 가독성을 높인다.

130 4. 별칭에 띄어쓰기나 특수 문자나 한글을 사용할 때, 대소문자를 기술할 때(기본값은 모두 대문자)에는 "" 로 기술한다.

예시)

SELECT ename "Name", sal AS "Salary", sal \* 12 + IFNULL(comm, 0) AS "Annual Salary"

FROM emp;

# Name, Salary, Annual Salary

'ADAMS', '1100.00', '13200.00'

'ALLEN', '1600.00', '19500.00'

'BLAKE', '2850.00', '34200.00'

REM Concatenation Operator (연결 연산자)

142 1. Oracle에서는 문자열 리터럴을 이을 때에는 '||' 를 사용한다.

143 2. 하지만 MySQL에서는 연결연산자(||)가 없기 때문에 CONCAT()를 사용한다.

144 3. Character string들을 연결하여 하나의 결과 string을 만들어 낸다.

145

146 SELECT CONCAT(empno, ' ', ename) FROM emp;

# CONCAT(empno, ' ', ename)

'7369 SMITH'

'7499 ALLEN'

'7521 WARD'

'7566 JONES'

REM Literals (상수)

152 1. Literal은 상수 값을 의미.

153 2. Character literal은 작은 따옴표로 묶고, Number literal은 따옴표 없이 표현한다.

154 3. Character literal을 작은 따옴표로 묶어 주어야 MySQL Server는 keyword나 객체 이름을 구별할 수 있다.

147 SELECT CONCAT(ename, ‘의 봉급은 ’,sal, ‘입니다.’) AS 봉급 FROM emp;

# 봉급

'SMITH 의 봉급은 800.00입니다.'

'ALLEN 의 봉급은 1600.00입니다.'

'WARD 의 봉급은 1250.00입니다.'

148 --number이나 date값은 default 형태의 character 값으로 자동 변환한 후 연결된다.

REM Duplicate Values(중복 행 제거하기)

167 1. 기본적으로 Query는 ALL 을 사용하기 때문에 중복된 행이 출력된다.

168 2. DISTINCT를 사용하면 중복된 행의 값을 제거한다.

169 3. DISTINCT는 SELECT 바로 뒤에 기술한다.

170 4. DISTINCT 다음에 나타나는 column은 모두 DISTINCT에 영향을 받는다.

SELECT job FROM emp;

/\* 출력:

'CLERK'

'salESMAN' - 중복

'salESMAN' - 중복

'MANAGER'

\*/

SELECT DISTINCT job FROM emp;

/\* 출력:

'CLERK'

'salESMAN'

'MANAGER'

'ANALYST'

'PRESIDENT'

\*/

REM WHERE 절

182 1. 사용자들이 자신이 원하는 자료만을 검색하기 위해서 사용

183 2. Syntax

184

185 SELECT column ...

186 FROM 테이블\_명

187 WHERE conditions;

188

189 3. WHERE 절을 사용하지 않으면 FROM 절에 명시된 table의 모든 ROW를 조회하게 된다.

190 4. Table내의 특정 row만 선택하고 싶을 때 WHERE 절에 조건식을 사용한다.

191 5. MySQL Server는 table의 row를 하나씩 읽어 WHERE 절의 조건식을 평가하여 TRUE로

만족하는 것만을 선택한다.

192 6. Condition을 평가한 결과는 TRUE, FALSE, NULL 중의 하나이다.

193 7. Condition 내에서 character 와 date 값의 literal은 작은 따옴표를 사용하고, NUMBER 값은

그대로 사용한다.

194 8. 윈도우 에서는 Condition에서 사용하는 character 값은 대소문자를 구별하지 않는다. 즉 다음 두 조건은 동일한 조건.

195 1) WHERE ename = 'JAMES';

196 2) WHERE ename = 'james';

198 9. date 타입의 변경은 DATE\_FORMAT()를 사용한다.

select date\_format(hiredate,"%y년-%m월-%d일") from emp;

출력: 80년-12월-17일

199 10. WHERE는 FROM 다음에 와야 한다.

200 11. WHERE 절에 조건이 없는 FTS(Full Table Scan) 문장은 SQL Tunning의 1차적인 검토 대상이 된다.

201 12. WHERE 조건절의 조건식은 아래 내용으로 구성된다.

202 -Column 명(보통 조건식의 좌측에 위치)

203

- 비교 연산자 <, >, <=, >=, =, !=, <>(같지 않다)

- 논리 연산자 AND(&&), OR(||), NOT(!)

SELECT ename,sal,hiredate FROM emp

WHERE hiredate>='1987-01-01' AND hiredate<='1987-12-31';

SELECT ename, job FROM emp

WHERE job='CLERK' OR job='MANAGER';

- SQL 연산자

1. BETWEEN A AND B : A보다 같거나 크고, B보다 작거나 같은

SELECT ename,sal,hiredate FROM emp

WHERE hiredate BETWEEN '1987-01-01' AND '1987-12-31';

2. IN(list) : LIST 안에 있는 멤버들과 같은

SELECT ename, job FROM emp

WHERE job IN('CLERK','MANAGER');

3. A LIKE B [ESCAPE 'C']: A가 B의 패턴과 일치하면 TRUE, 보통 %(\* 여러 글자 대응), \_(? 한 글자 대응) 연산자와 같이 사용, escape 을 사용하면 B의 패턴 중에서 C를 상수로 취급한다.

SELECT ename,sal,hiredate FROM emp WHERE hiredate LIKE "1987%";

4. IS NULL / IS NOT NULL : NULL 여부를 테스트

SELECT ename, comm FROM emp WHERE comm IS NOT NULL;

// comm 지급 대상인 사원의 이름과 지급 금액

-문자, 숫자, 표현식(보통 조건식의 우측에 위치)

-비교 Column명 (JOIN 사용시)

샘플 코드 작성해보기

--1983년 이후에 입사한 사원의 사번, 이름, 입사일을 출력하시오.

**SELECT 사번, 이름, 입사일 FROM 사원테이블 WHERE 입사일 >=’1983-01-01’;**

--급여가 보너스(comm) 이하인 사원의 이름, 급여 및 보너스를 출력하시오

**SELECT 이름,급여, 보너스 FROM 사원테이블 WHERE 급여 < 보너스**

--10번 부서의 모든 사람들에게 급여의 13%를 보너스로 지급하기로 했다. 이름, 급여, 보너스 금액, 부서번호를 출력하시오.

**SLEECT 이름, 급여, 급여\*0.13 AS “보너스 금액”, 부서번호 FROM 사원테이블; WHERE 부서 = 10;**

--30번 부서의 연봉을 계산하여, 이름, 부서번호, 급여, 연봉을 출력하라. 단, 년말에 급여의 150%를 보너스로 지급한다.

**SELECT 이름, 부서번호, 급여, (급여\*12+급여\*1.5) AS “연봉” FROM 사원테이블 WHERE dept = 30;**

REM ORDER BY

428 1. 기본적으로 데이터는 정렬되지 않는다.

429 2. 같은 쿼리를 수행할 때마다 결과가 다르게 나올 수 있다.

430 3. 별칭은 정렬에 영향을 주지 않는다.

431 4. Syntax

432

433 SELECT column\_list

434 FROM table

435 [WHERE conditions]

436 [ORDER BY column[, column] {ASC | DESC};

437

438 5. 특징

439 1) 기본적으로 오름차순정렬한다.

440 --숫자인경우 ( 1 --> 999)

441 --날짜인경우 (옛날 --> 최근)

442 --문자인겨우 (알파벳순서, 유니코드순)

443 2) NULL은 오름차순일 경우는 제일 처음에, 내림차순인 경우에는 제일 마지막에 출력

444

445 6. ORDER BY 절에 정렬의 기준이 되는 column을 여러 개 지정할 수 있다.

ORDER BY hiredate ASC, ename DESC;

첫번째 column으로 hiredate로 정렬한 다음, 그 column 값이 같은 row들에 대해서는 두 번째 column 값 ename으로 정렬하도록 구성할 수 있다..

446 7. 오름차순(ASC) 정렬이 기본이며, 내림차순으로 정렬하고자 할 때에는 DESC를 사용한다.

447 8. ORDER BY 절에 column 이름 대신 별칭(AS)을 사용할 수도 있다.

SELECT empno, ename, sal, comm, sal \* 12 + IFNULL(comm, 0) AS "Annual"

456 FROM emp WHERE deptno = 20

457 ORDER BY "Annual" ASC;

REM SQL function

7 -A function is a stored program that you can pass parameters into and then return a value.

8 1. Built Function(내장함수)

9 2. Stored Function(사용자 정의 함수)

10

11 REM 단일행 함수(Single Row function)

12 1. Syntax

13 function\_name(column | expression [ arg1, arg2...])

14

15 2. 종류

16 1)제어흐름 함수

**2)숫자 함수**

**3)날짜, 시간 함수**

**4)문자열 함수**

20 5)집합 함수

21 6)변환 함수

22 7)기타 함수

23

24

REM 제어 흐름 함수(Flow Control Functions)

26 1. IF()

27 1)Definition

28 -Returns a value if a condition is TRUE, or another value if a condition is FALSE.

29

30 2)Syntax

31 IF(expr1, expr2, expr3)

32

33 3)만일 expr1이 참이면, expr2를 리턴한다.

34 4)그렇지 않으면 expr3을 리턴한다.

2. CASE : 정해진 조건에 따라 다른 가중치를 적용 할 경우

41 1)Definition

42 -Goes through conditions and return a value when the first condition is met.

43 -like an IF-THEN-ELSE statement.

44 -So, once a condition is true, it will stop reading and return the result.

45 -If no conditions are true, it will return the value in the ELSE clause.

46 -If there is no ELSE part and no conditions are true, it returns NULL.

47

48 2)Syntax

49 CASE

50 WHEN compare\_value1 THEN result1

51 WHEN compare\_value2 THEN result2

52 WHEN compare\_value3 THEN result3

53 ...

54 ELSE resultN

4. NULLIF

81 1)Definition

82 -Compares two expressions and returns NULL if they are equal. Otherwise, the first

expression is returned.

2)Syntax

85 NULLIF(expr1, expr2)

Expr1과 expr2가 같다면 Null, 같지 않다면 expr1 출력