

개발환경설정

with



PL/SQL
ORACLE®

01. Oracle



DBMS
(Database Management System)

- 정의 (Definition) : 응용 프로그램이 요구하는 DB구조, 변경, 제거
- 조작 (Manipulation): 삽입, 갱신, 삭제
- 제어 (Control): DB 접근할 수 있는 사용자 제한 및 성능 관리

관계형 데이터베이스
(Relational Database)

- 통상적으로 RDB라고 함
- 행과 열로 구분하는 2차원 테이블 형태로 구성
- 오라클 DBMS, MS SQL SERVER, MYSQL, PostgreSQL

Oracle의 인기

- DB Engines Ranking 1위 (350개 DB) (2020 ~ 2021)¹
- 국내 시장 점유율 70%²

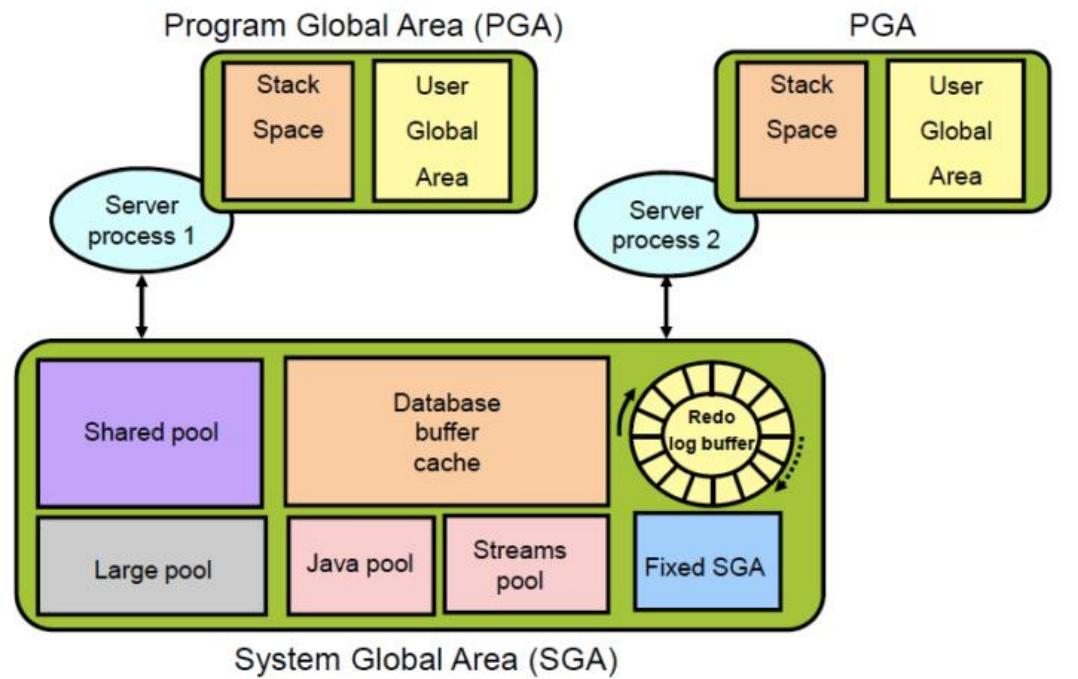
ORACLE의 장점

- 중앙 집중 방식, 쿼리 최적화 프로그램 등
- 다양한 플랫폼 지원 (Windows, MacOS, Linux 등)

출처 1: <https://db-engines.com/en/ranking>

출처 2: <https://www.sedaily.com/NewsView/1YXT5Z0V09>

02. 오라클 데이터베이스의 구조



출처 : 황선영 (2019). 오라클 데이터베이스의 구조

- PGA(Program Global Areas)
 - : 비공유 메모리, 각 서버 프로세스에 대한 데이터 및 정보 포함
 - : 유저의 글로벌 데이터나 세션 상태정보, 참조되는 포인터 정보
- SGA(System Global Area)
 - : Shared Pool – 공유 SQL 영역 및 라이브러리 캐시 옵티마이저, 하드 파싱, 소프트 파싱
 - : Large Pool – 백업이나 Recovery 작업 시 메모리 할당
 - : Java Pool – 오라클 DB에서 자바 이용할 수 있도록 도와줌

03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

오라클 다운로드 (19c / 19.3 Enterprise Edition)

(웹사이트 주소: <https://www.oracle.com/database/technologies/oracle-database-software-downloads.html#19c>)

Oracle Database 19c

Oracle Database 19c is the latest Long Term Release with the widest window of support duration. For details about database releases and their support timeframes, refer to Oracle Support Document 742060.1 (Release Schedule of Current Database Releases) on My Oracle Support.

19.5 - Enterprise Edition (also includes Standard Edition 2)

Name	Download	Note
Oracle Solaris (x86 systems, 64-bit)	 ZIP (2.7 GB)	See All

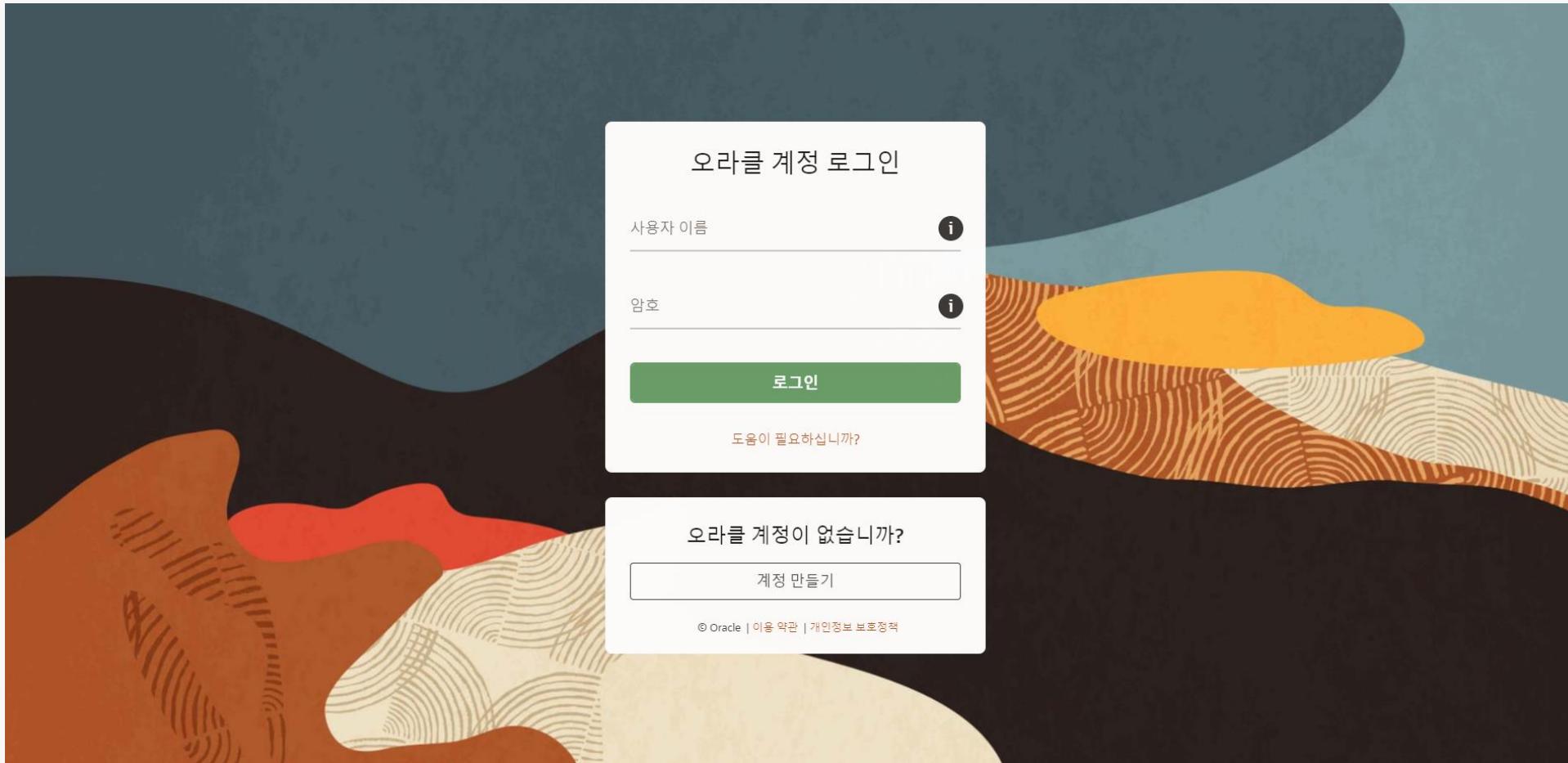
19.3 - Enterprise Edition (also includes Standard Edition 2)

Name	Download	Note
Microsoft Windows x64 (64-bit)	 ZIP (2.9 GB)	See All
Linux x86-64	 ZIP (2.8 GB)  RPM (2.5 GB)	See All
Oracle Solaris (SPARC systems, 64-bit)	 ZIP (2.8 GB)	See All
IBM AIX	 ZIP (4.1 GB)	See All
HP-UX ia64	 ZIP (4.7 GB)	See All
Linux on System z (64-bit)	 ZIP (2.6 GB)	See All

03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

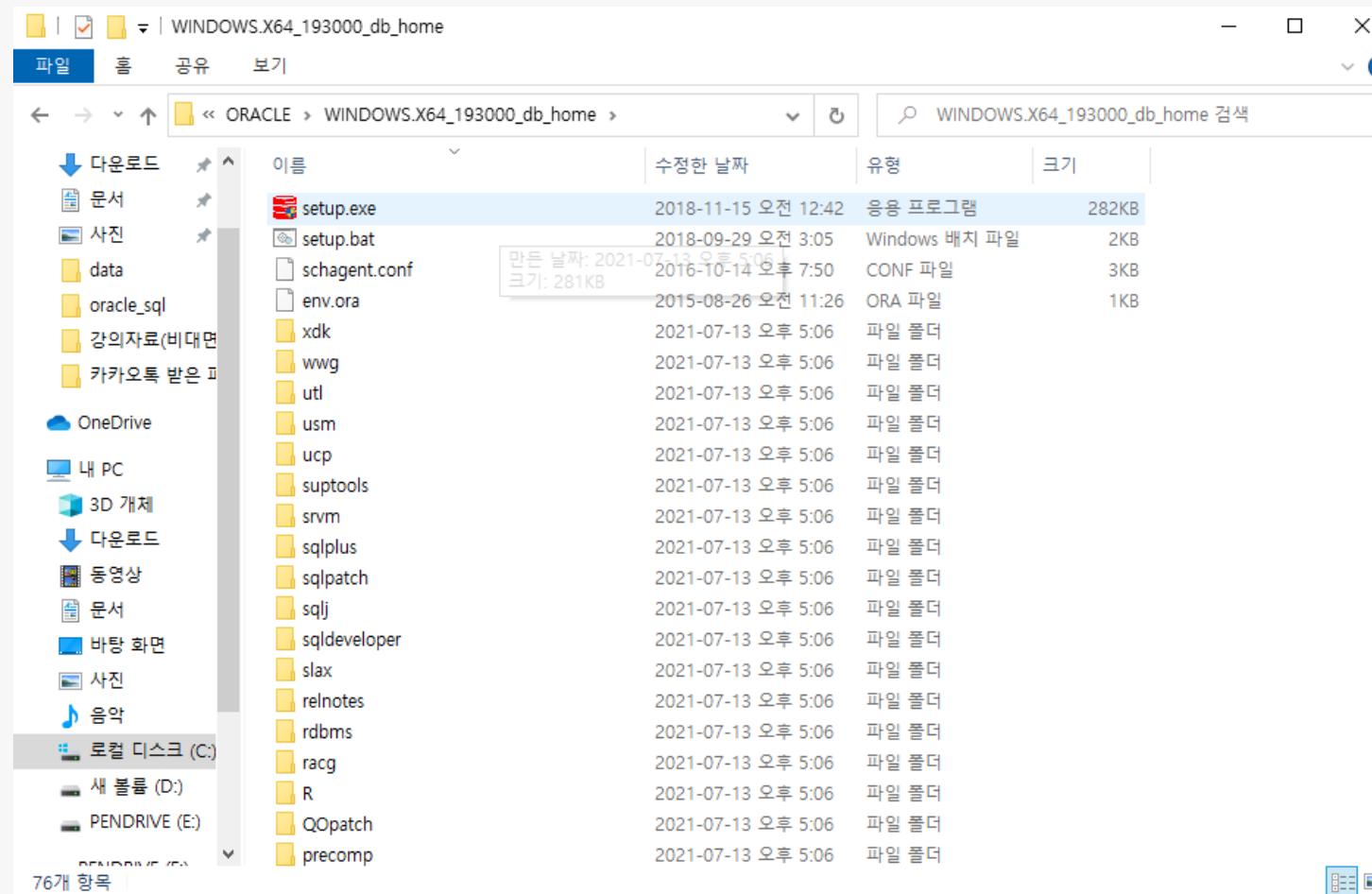
오라클 계정 로그인



03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

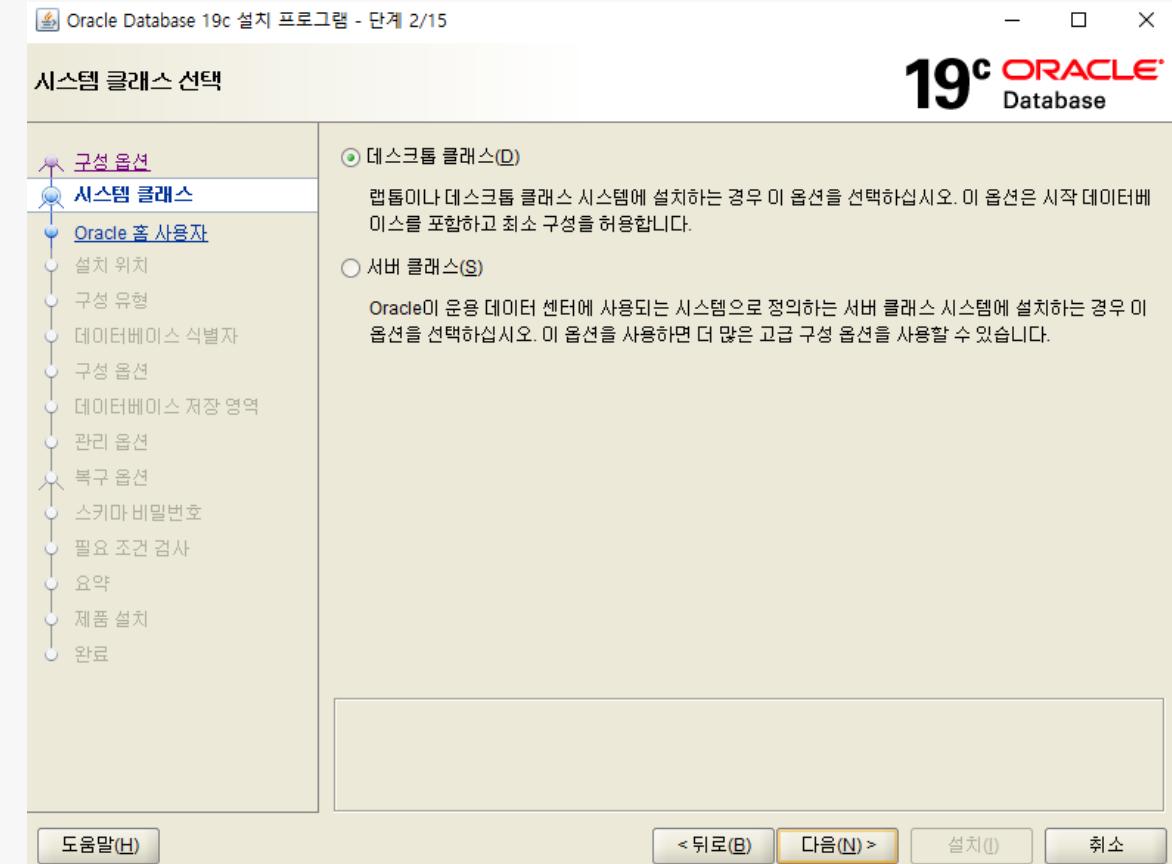
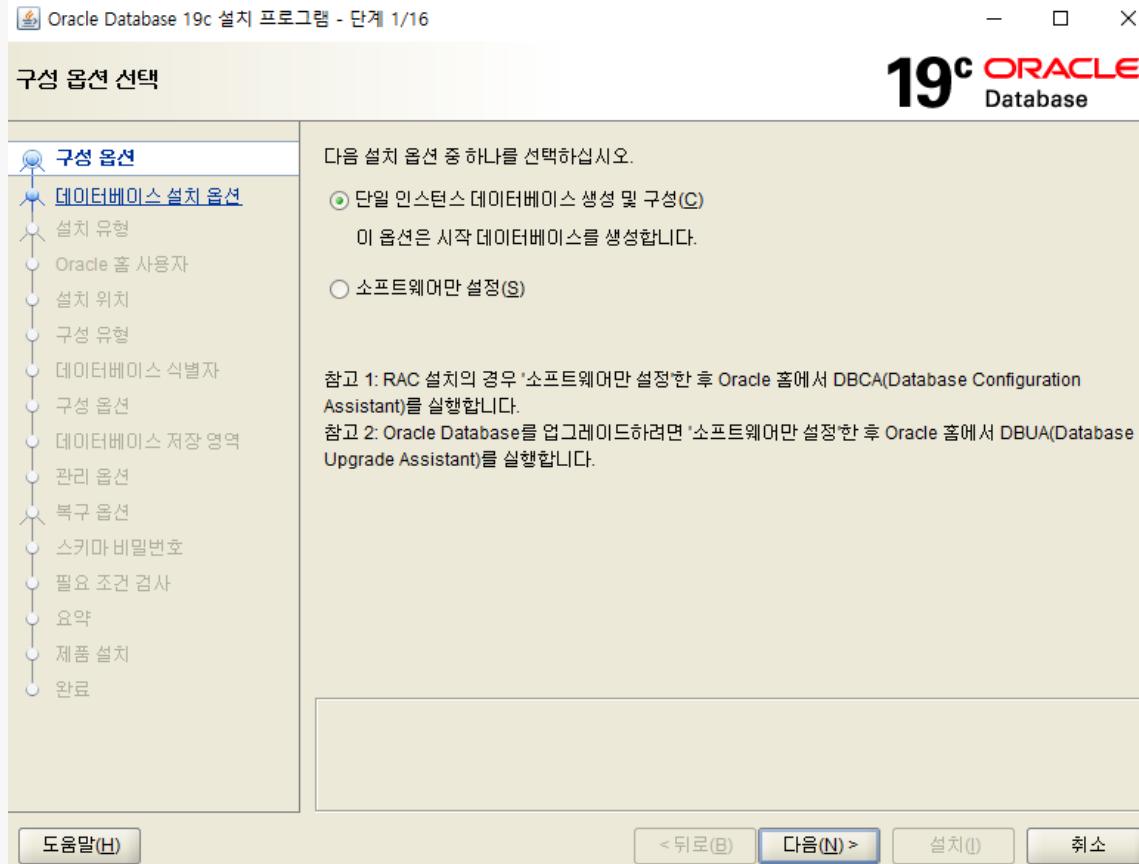
압축파일 해제 – 설치 프로그램 실행



03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

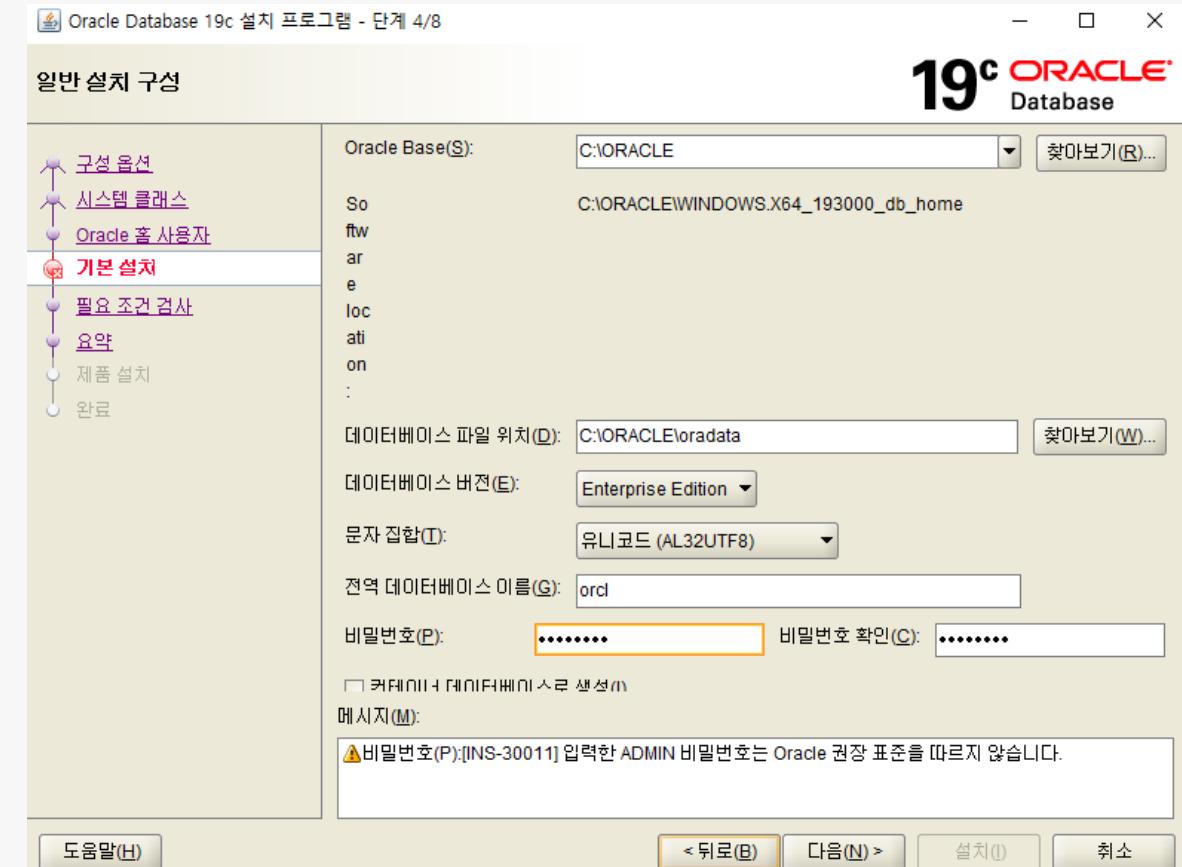
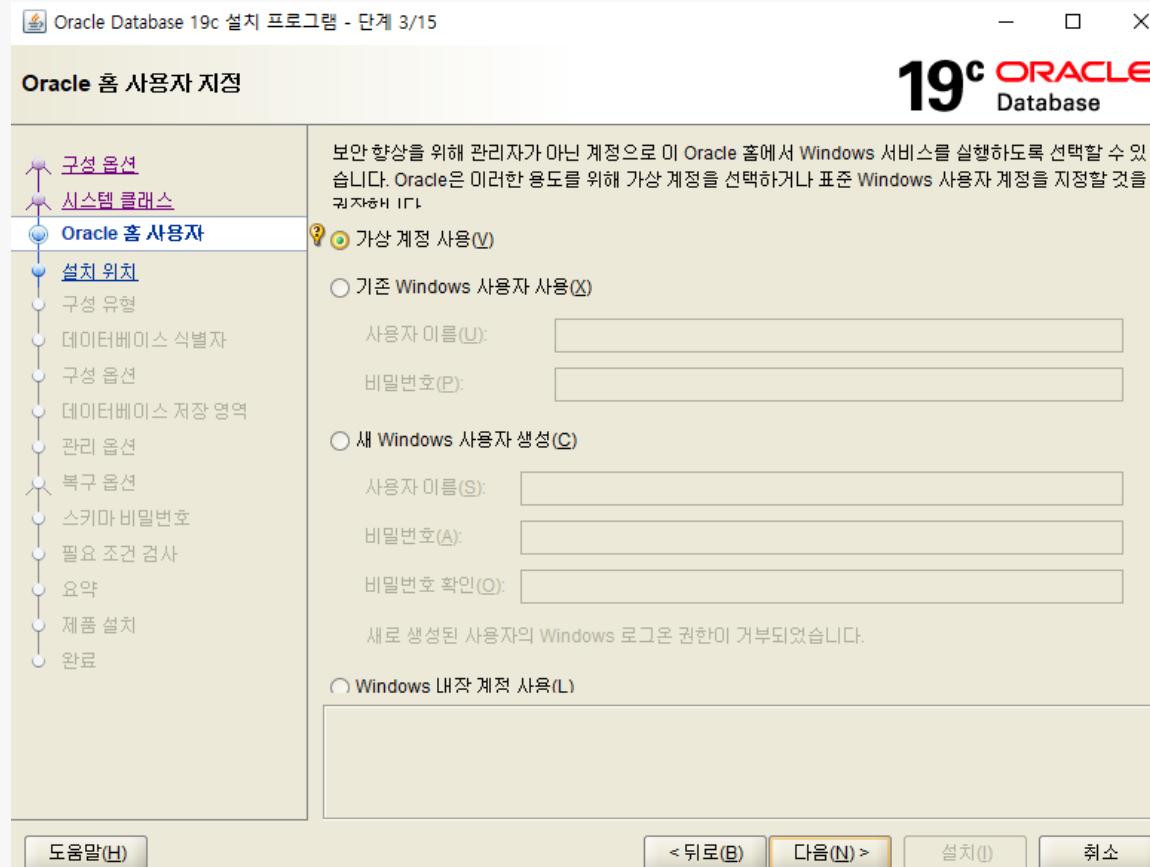
순차적으로 진행 (왼쪽에서 오른쪽 순으로)



03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

순차적으로 진행 (왼쪽에서 오른쪽 순으로)



03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

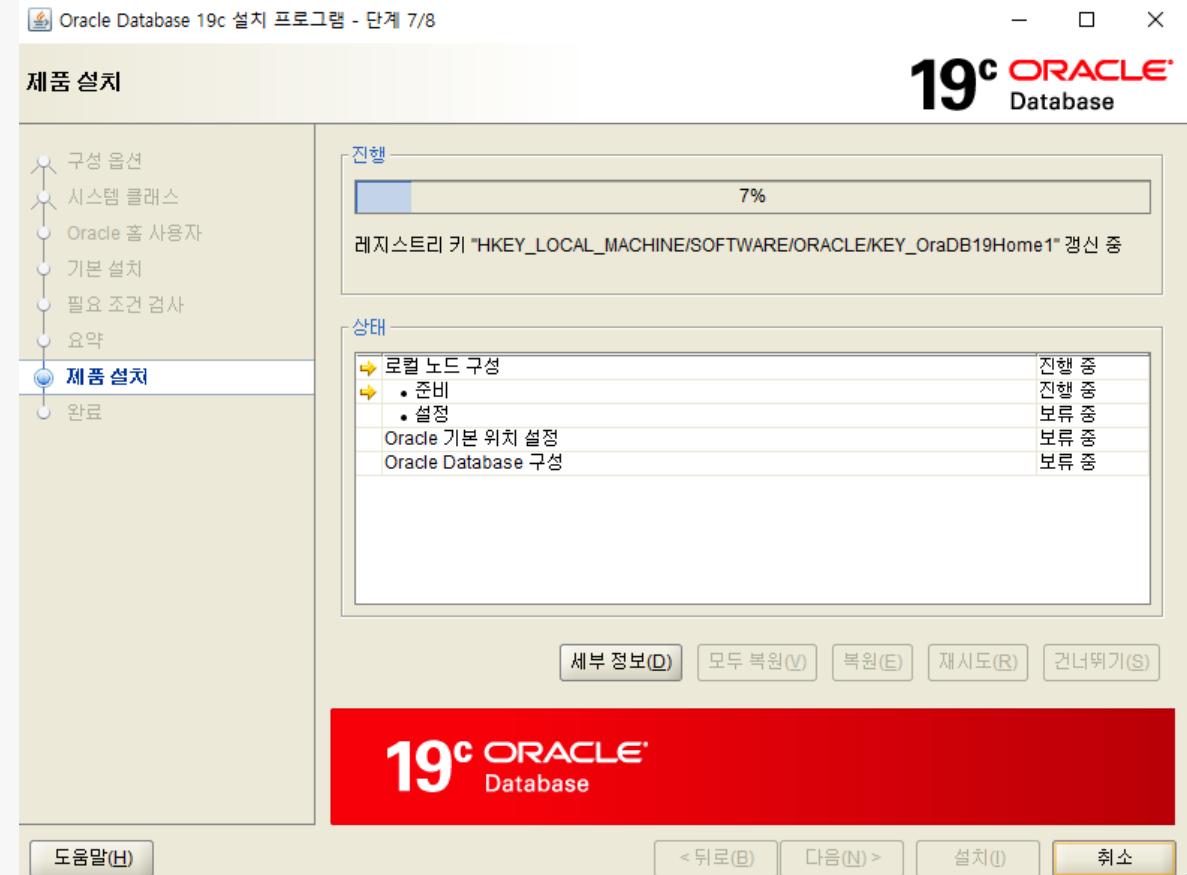
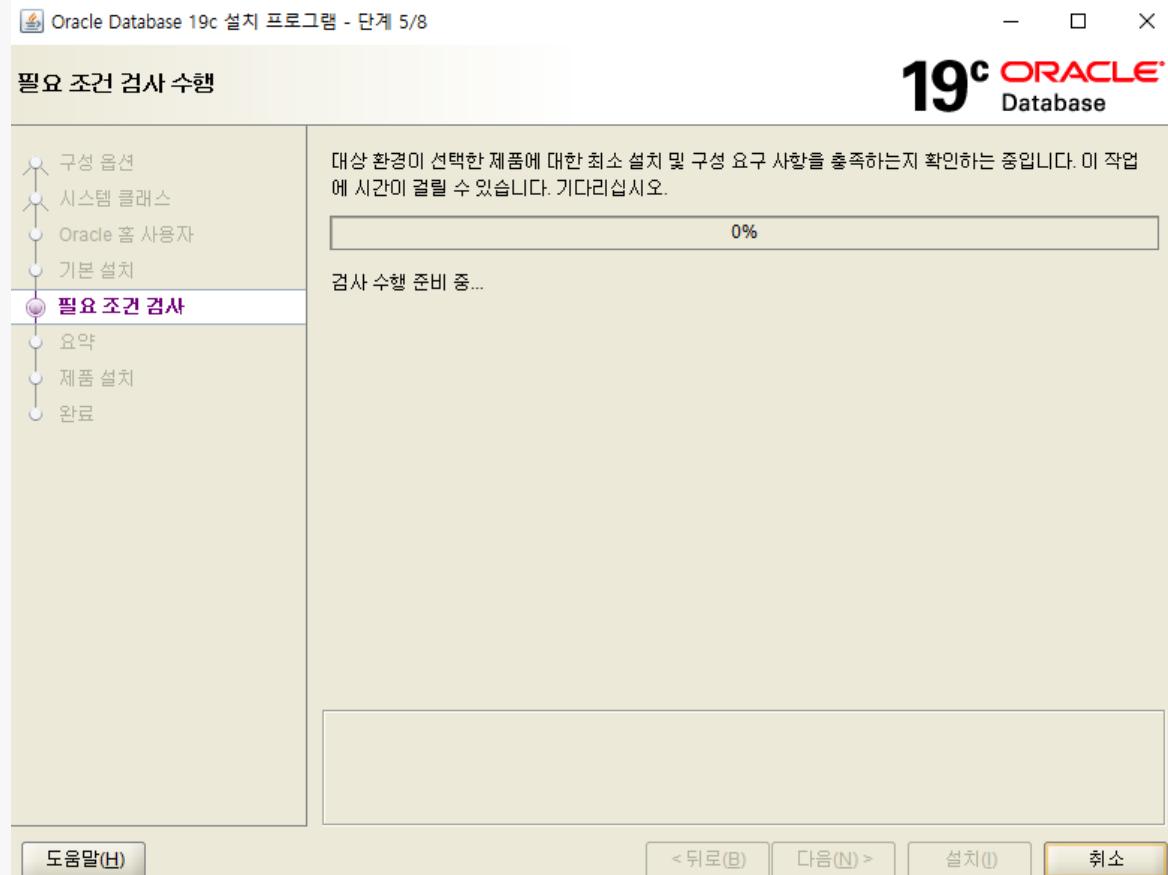
순차적으로 진행 (왼쪽에서 오른쪽 순으로)



03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

순차적으로 진행 (왼쪽에서 오른쪽 순으로)



03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

순차적으로 진행



03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

순차적으로 진행



- OracleServiceMyORACLE
 - OracleService + SID 형태로 구성된 서비스, 오라클 사용 시 반드시 맨 먼저 시작됨
- OracleOraBb19C_home1TNSListener
 - 리스너 서비스로 네트워크를 통해 클라이언트(오라클 사용자)와 오라클 서버와의 연결을 담당하는 관리 프로그램
- OracleDBConsolemyoracle
 - EM(Enterprise Manager)을 사용할 경우 시작해야 하는데, EM은 설치한 오라클을 관리하는 프로그램

05. SQLPlus 실행하기

(2021년 6월 기준)

- 원도우에서 명령창을 열고 'sqlplus'를 입력

```
명령 프롬프트 - sqlplus
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\human>sqlplus

SQL*Plus: Release 19.0.0.0.0 - Production on 목 6월 2 10:28:39 2022
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

사용자명 입력: system
비밀번호 입력:
마지막 성공한 로그인 시간: 목 6월 02 2022 10:28:27 +09:00

다음에 접속됨:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL>
```

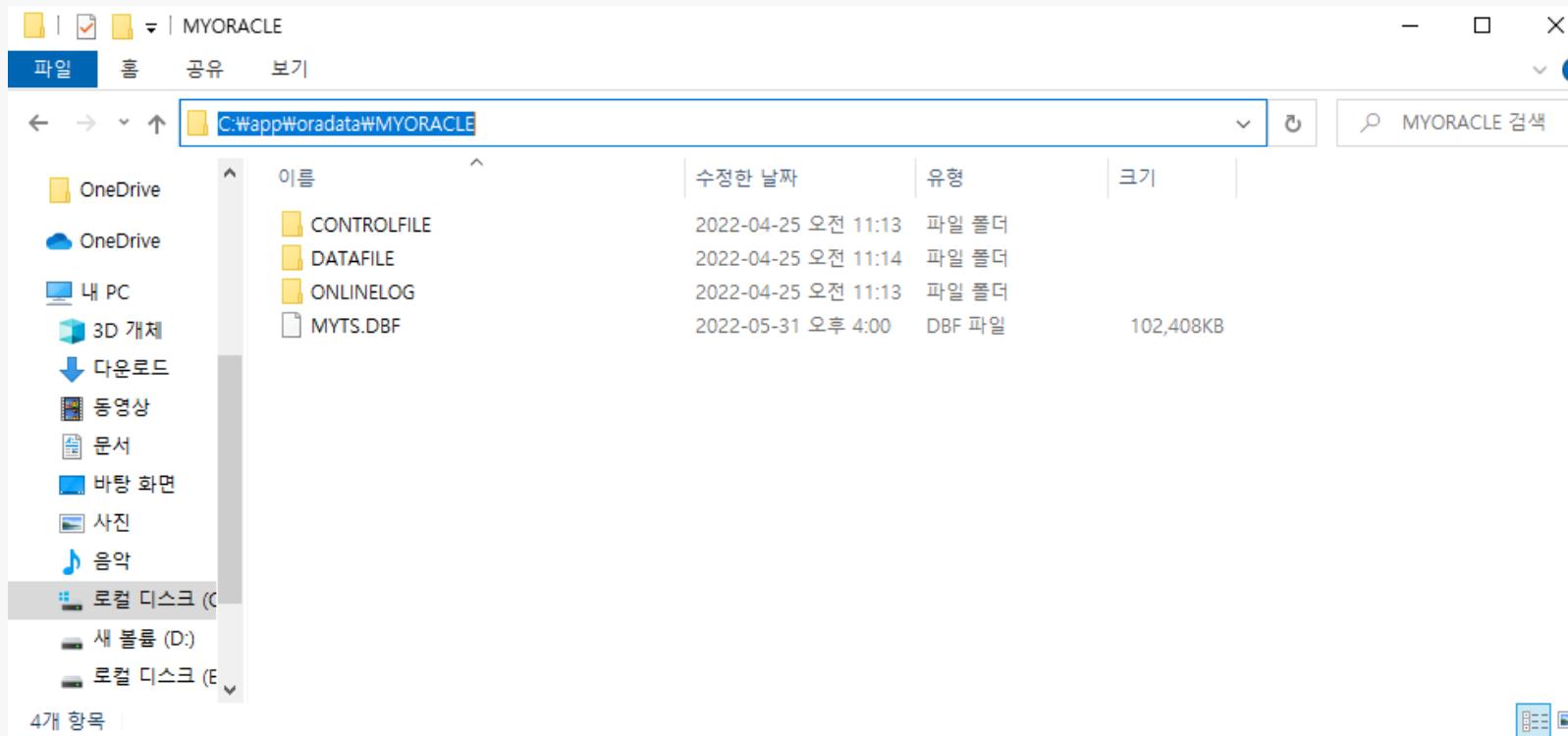
06. 테이블스페이스 생성하기

(2021년 6월 기준)

- 오라클이 설치된 경로에 myts라는 이름으로 100MB 크기로 생성

```
SQL> CREATE TABLESPACE myts DATAFILE  
2      'C:\app\your_name\oradata\myoracle\myts.dbf' SIZE 100M AUTOEXTEND ON NEXT 5M;
```

- 경로 작성시, 실제 oradata가 경로에 있는지 확인 필요



07. 사용자 생성

(2021년 6월 기준)

- ID : ora_user / PW : evan

```
SQL> CREATE USER ora_user IDENTIFIED BY evan
  2 DEFAULT TABLESPACE MYTS
  3 TEMPORARY TABLESPACE TEMP;
```

- 역할 부여

```
SQL> GRANT DBA TO ora_user;
```

- ora_user 로그인 화면

```
SQL> connect ora_user/evan;
연결되었습니다.
SQL> SELECT USER FROM dual;
```

USER

ORA_USER

- 두개의 파일 확인 (expall.dmp, expcust.dmp)
- C드라이브에 backup 폴더를 생성 후 해당 파일 저장
- 명령창을 열고 C:\backup 폴더로 이동
- imp 이하 명령어 입력 후 Enter 누름
 - ora_user/evan은 각자 설정한 ID와 비밀번호로 바꾼다.
- expall.dmp & expcust.dmp 파일을 업로드한다.

```
C:\backup> imp ora_user/evan file=expall.dmp log=empall.log ignore=y grants=y rows=y indexes=y full=y
```

```
C:\backup> imp ora_user/evan file=expcust.dmp log=expcust.log ignore=y grants=y rows=y indexes=y full=y
```

08. 파일 업로드

(2021년 6월 기준)

- sqlplus에서 table을 확인한다.

```
SQL> SELECT table_name FROM user_tables;
```

```
TABLE_NAME
```

```
-----
```

```
CUSTOMERS
```

```
SALES2
```

```
SALES
```

```
PRODUCTS
```

```
CHANNELS
```

```
COUNTRIES
```

```
JOB_HISTORY
```

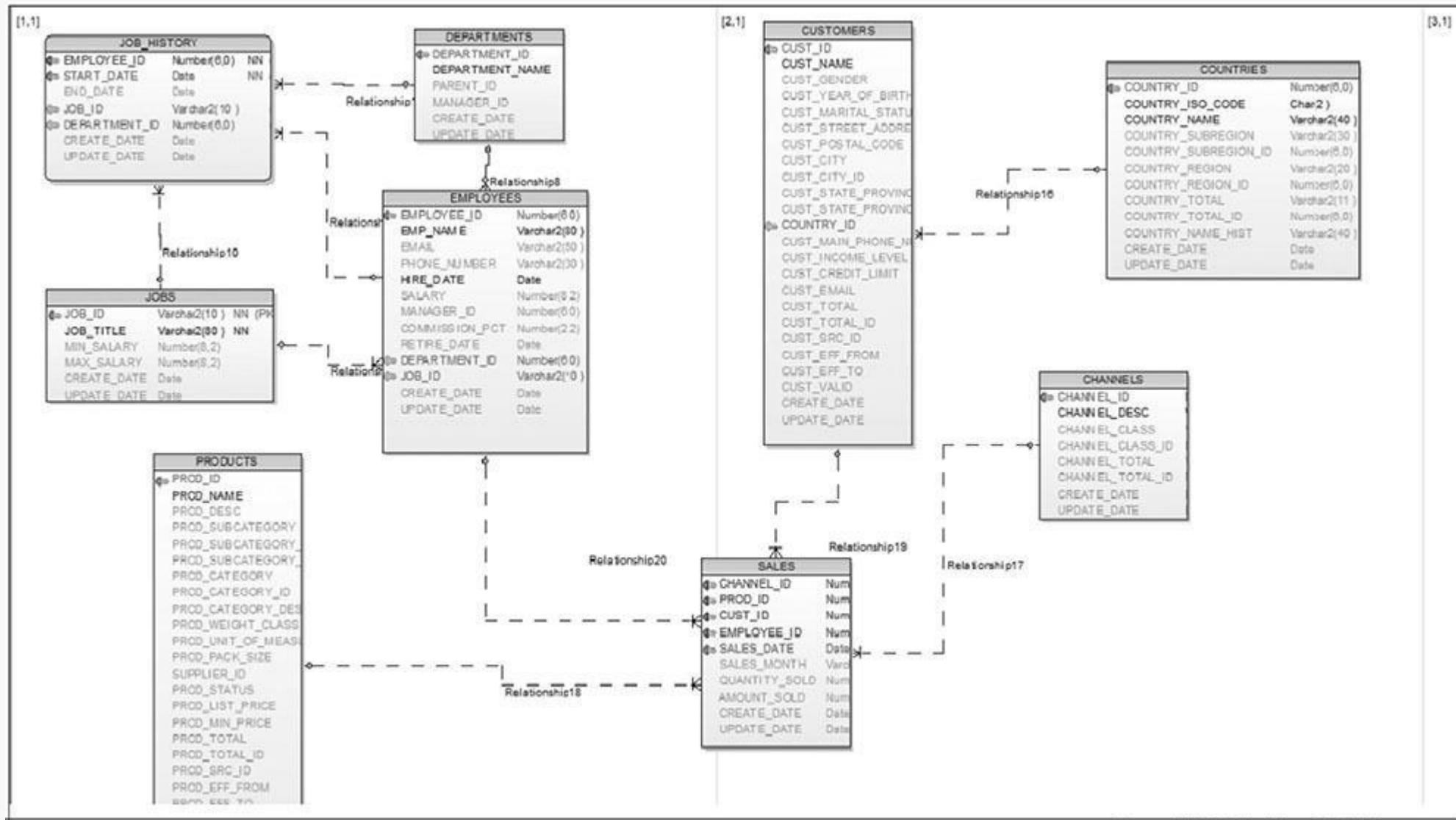
```
EMPLOYEES
```

```
JOBS
```

```
DEPARTMENTS
```

09. 스키마 비교

(2021년 6월 기준)



Copyright © Gilbut, Inc. All rights reserved.

10. SQL Developer 설치

(2021년 6월 기준)

SQL Developer 설치 (자바 환경설정 유무 필수 확인)

<https://www.oracle.com/tools/downloads/sqldev-downloads.html>

The screenshot shows the Oracle website's download section for SQL Developer. At the top, there's a navigation bar with links for Products, Industries, Resources, Support, Events, and Developer. Below the navigation is a search bar and account-related buttons for 'View Accounts' and 'Contact Sales'. The main content area is titled 'SQL Developer 20.4.1 Downloads' and specifies 'Version 20.4.1.407.0006 - February 22, 2021'. It lists two download options:

Platform	Download	Notes
Windows 64-bit with JDK 8 included	Download (423 MB)	<ul style="list-style-type: none">MD5: 73e7180ec8b494868c2fce4d99114630SHA1: 0ad1cc75a28c0ac3eb9797bfc167bfc6fea3212eInstallation Notes
Windows 32-bit/64-bit	Download (432 MB)	<ul style="list-style-type: none">MD5: 041709f01de2c6d176f37132089b61b8SHA1: df90320a3a6e15df90fafb9d0c603317f3a68b84Installation NotesJDK 8 or 11 required

10. SQL Developer 설치

(2021년 6월 기준)

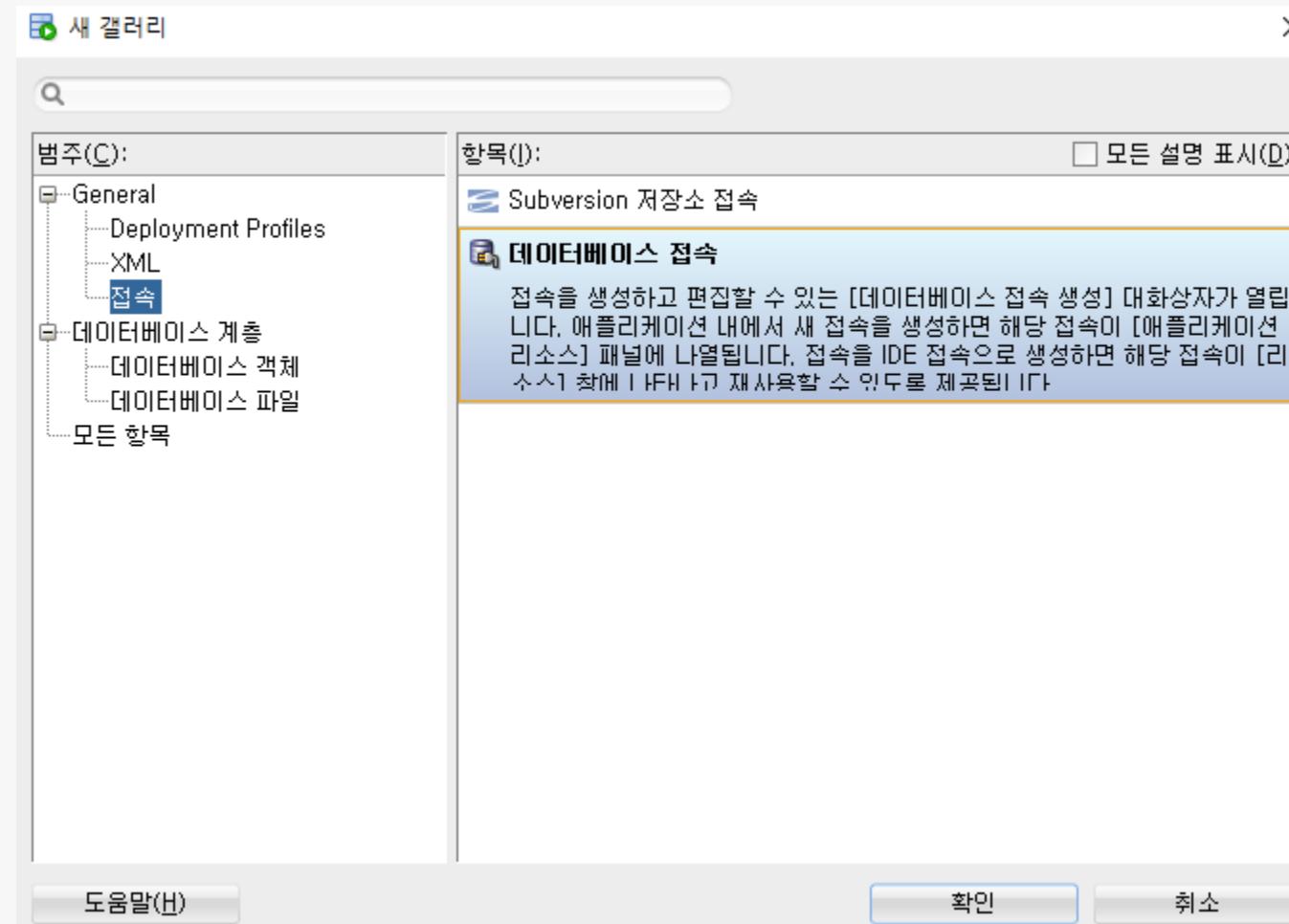
순차적으로 진행



10. SQL Developer 설치

(2021년 6월 기준)

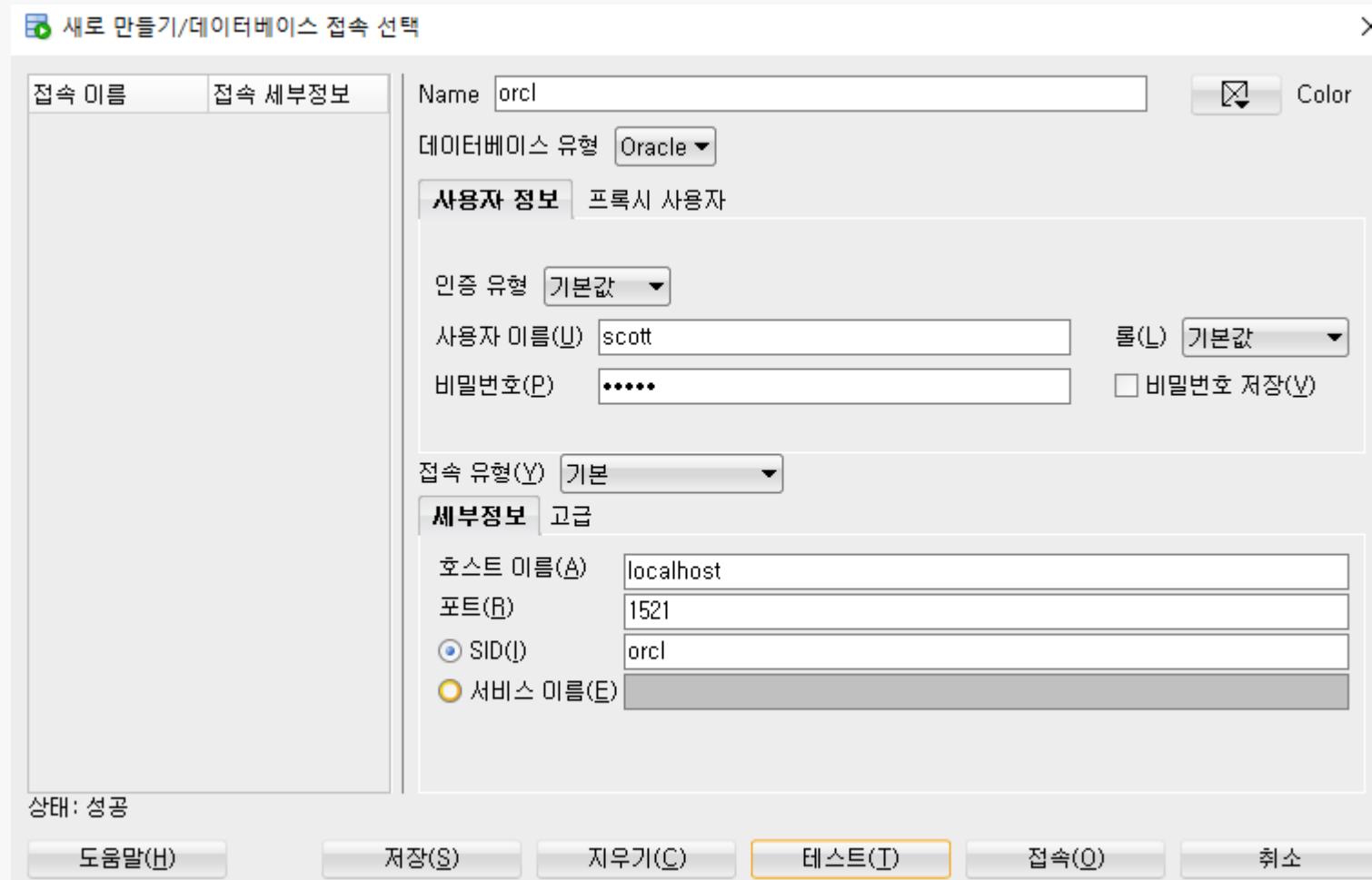
순차적으로 진행



10. SQL Developer 설치

(2021년 6월 기준)

순차적으로 진행



10. SQL Developer 설치

(2021년 6월 기준)

해당 코드 실행 (Ctrl + Enter)

The screenshot shows the Oracle SQL Developer interface with the following details:

- Toolbar:** Includes File, Edit, View, Tools, Database, Help, and a search bar.
- Left Sidebar:** Shows a connection tree under "Oracle 접속" (orcl) and a "데이터베이스 스키마 서비스 접속".
- Central Area:** A "워크시트" (Worksheet) tab is active, containing the following SQL code:

```
commit;

drop table salgrade;

create table salgrade
(
    grade   number(10),
    losal   number(10),
    hisal   number(10)
);

insert into salgrade values(1,700,1200);
insert into salgrade values(2,1201,1400);
insert into salgrade values(3,1401,2000);
insert into salgrade values(4,2001,3000);
insert into salgrade values(5,3001,9999);

commit;
```
- Bottom Area:** A "스크립트 출력" (Script Output) window shows the results of the executed script:

```
작업이 완료되었습니다.(0.437초)

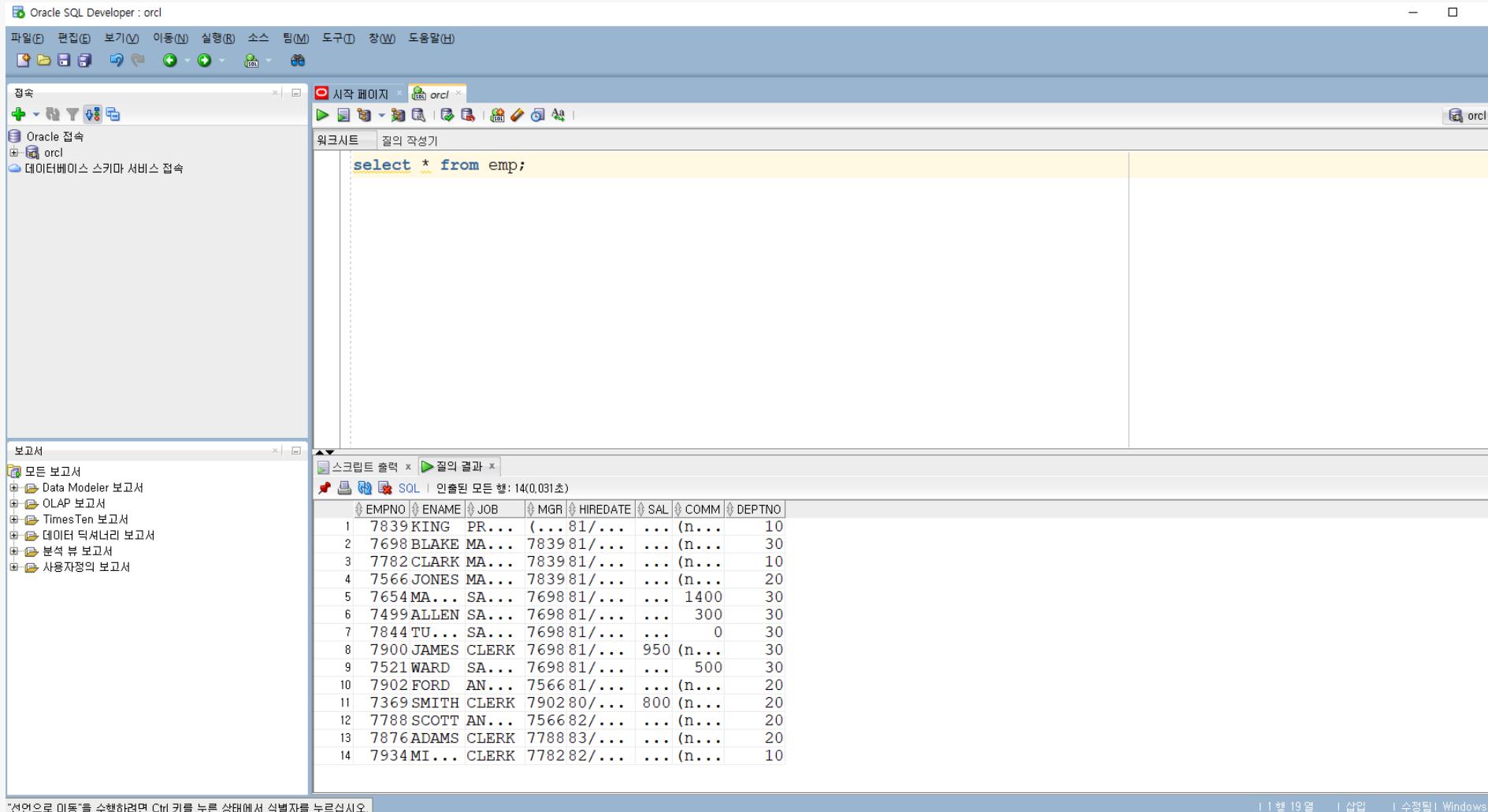
1 행 미 (가) 삽입되었습니다.

커밋 완료.
```

10. SQL Developer 설치

(2021년 6월 기준)

해당 코드 실행 (Ctrl + Enter)



오라클 SQL 입문 문법

with



01. 연산자

- 비교 연산자

연산자	의미
>	크다
<	작다

- 기타 비교 연산자

연산자	의미
BETWEEN AND	~ 사이에 있는
LIKE	일치하는 문자 패턴 검색

- 기타 비교 연산자

연산자	의미
>=	크거나 같다
<=	작거나 같다
=	같다
=	같지 않다
^=	같지 않다
<>	같지 않다

- 기타 비교 연산자

연산자	의미
IS NULL	NULL 값인지 여부
IN	값 리스트 중 일치하는 값 검색

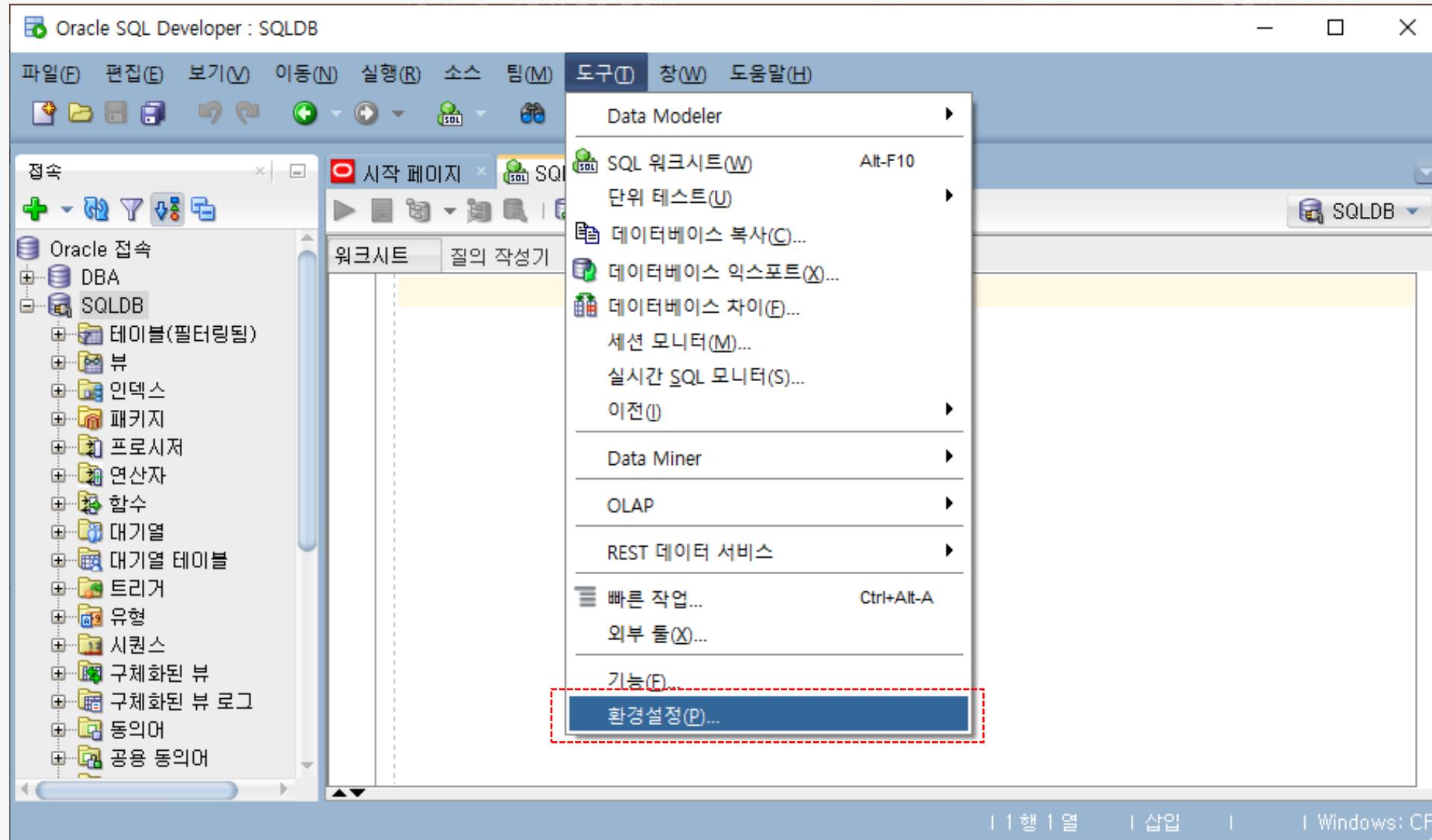
02. 날짜 형식

- 현재 날짜 형식 확인

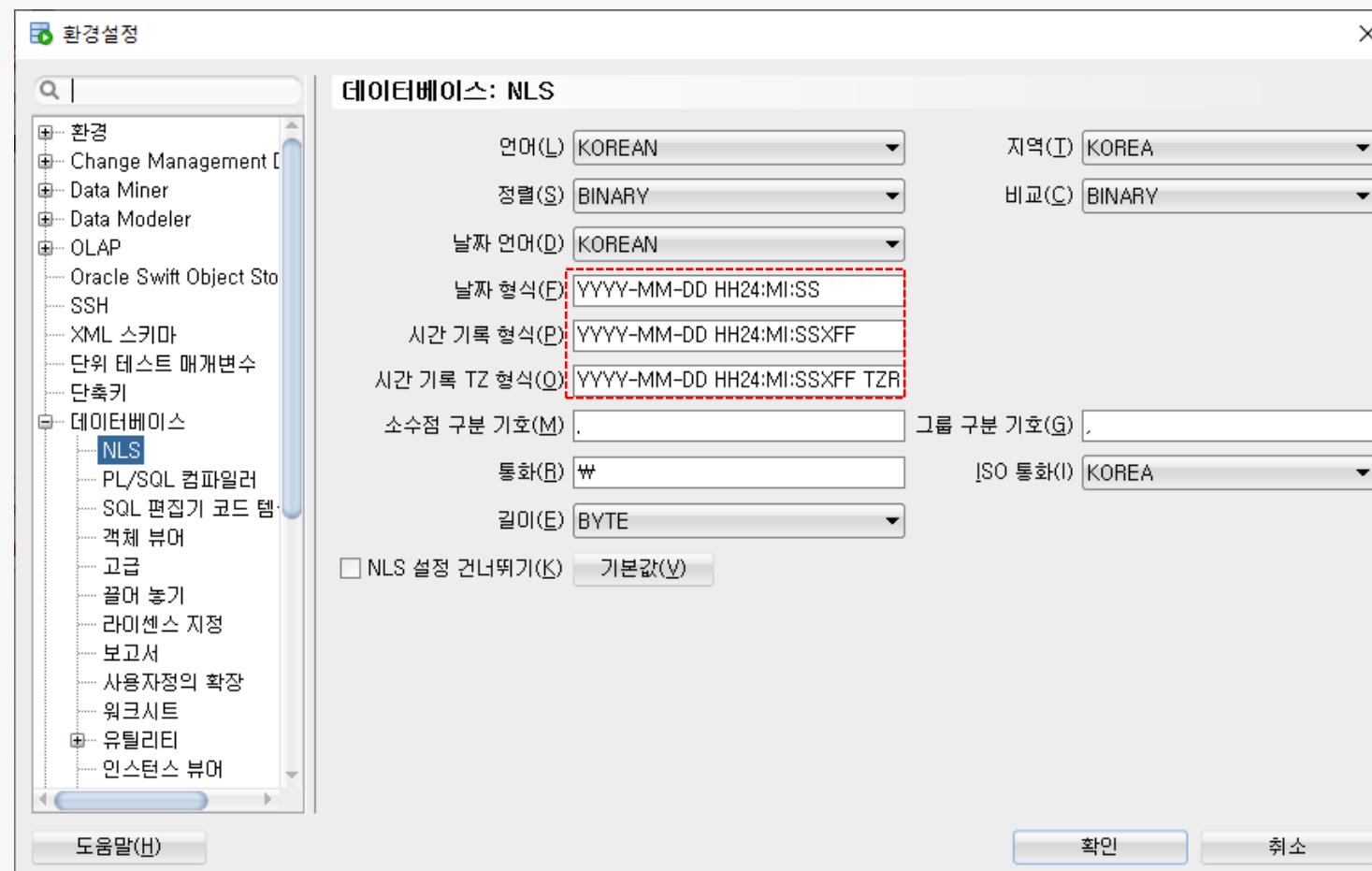
```
SQL> SELECT *
      FROM NLS_SESSION_PARAMETERS
     WHERE PARAMETER = 'NLS_DATE_FORMAT';
```

PARAMETER	VALUE
-----	-----
-----	-----
-----	-----
NLS_DATE_FORMAT	RR/MM/DD

02. 날짜 형식 수정 – SQL Developer



02. 날짜 형식 수정 – SQL Developer



The screenshot shows the SQL Developer interface with two panes. The left pane displays the results of the query 'SELECT SYSDATE FROM DUAL;' in RR/MM/DD format, showing the date as 19/03/24. The right pane displays the same query in YYYY/MM/DD HH24:MI:SS format, showing the date and time as 2019/03/24 16:07:20. Both results are highlighted with red boxes.

02. 날짜 형식

- 날짜 형식 수정

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT='YY/MM/DD';
```

```
세션이 변경되었습니다.
```

03. ADD OR NOT 연산자

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

OR	TRUE	FALSE	NULL
TRUE	FALSE	TRUE	NULL

오라클 SQL 기초 문법

with



01. 대소문자 변환 함수 배우기

- INTERVAL 표현식

함수의 종류	설명	
단일행 함수	정의	하나의 행을 입력받아 하나의 행을 반환하는 함수
	종류	문자, 숫자, 날짜, 변환, 일반 함수
다중 행 함수	정의	여러 개의 행을 입력받아 하나의 행을 반환하는 함수
	종류	그룹 함수

단일행 함수	함수
문자 함수	UPPER, LOWER, INITCAP, SUBSTR, LENGTH, CONCAT, INSTR, TRIM, LPAD, RPAD

02. INTERVAL 표현식

- INTERVAL 표현식

표현식	설명
INTERVAL '4' YEAR	An interval of 4 years 0 months
INTERVAL '123' YEAR	An interval of 123 years 0 months
INTERVAL '123' YEAR	An interval of 6 months
INTERVAL '600' MONTH(3)	An interval of 600 months
INTERVAL '400' DAY(3)	400 days
INTERVAL '10' HOUR	10 hours
INTERVAL '10' MINUTE	10 minutes
INTERVAL '4' DAY	4 days
INTERVAL '25' HOUR	25 hours
INTERVAL '40' MINUTE	'40' minute
INTERVAL '120' HOUR(3)	120 hours

03. 날짜 표현식

- 날짜를 문자로 출력할 때 사용할 수 있는 날짜 포맷

연도	RRRR, YYYY, RR, YY
월	MM, MON
일	DD
요일	DAY, DY

주	WW, IW, W
시간	HH, HH24
분	MI
초	SS

04. 데이터 분석 함수로 누적 데이터 출력하기(SUM OVER)

- ROWS

원도우 기준	원도우 방식	설명
ROWS	UNBOUNDED PRECEDING	맨 첫 번째 행을 가리킵니다.
	UNBOUNDED FOLLOWING	맨 마지막 행을 가리킵니다.
	CURRENT ROW	현재 행을 가리킵니다.

05. 데이터 분석 함수로 집계 결과 출력하기(GROUPING SETS)

- ROWS

GROUPING SETS	출력 결과
GROUPING SETS((deptno), (job), ())	부서 번호별 집계, 직업별 집계, 전체 집계
GROUPING SETS((deptno), (job))	부서 번호별 집계, 직업별 집계
GROUPING SETS((deptno, job), ())	부서 번호와 직업별 집계, 전체 집계
GROUPING SETS((deptno, job))	부서 번호와 직업별 집계

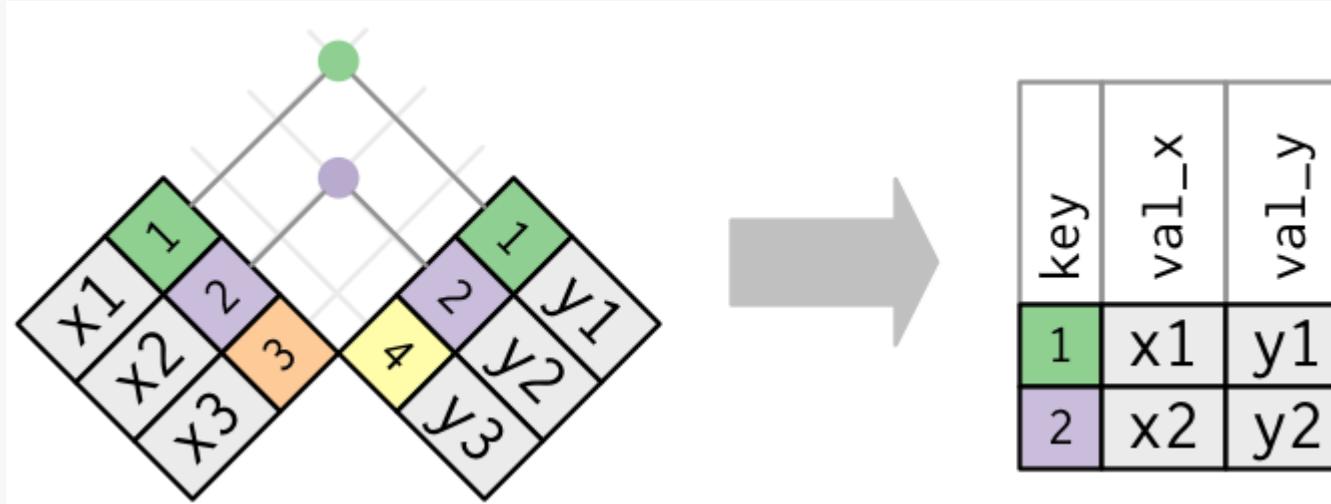
오라클 SQL 조인 문법

with



01. EQUI JOIN

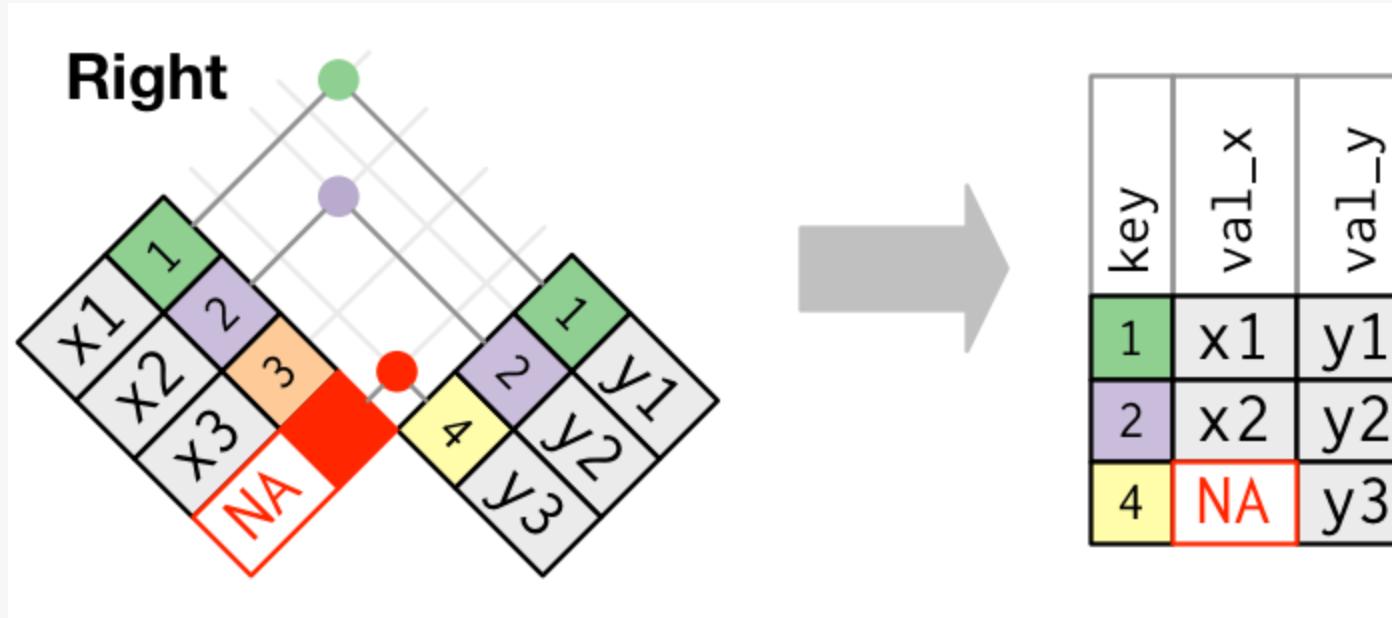
- INNER EQUI JOIN



출처: <https://r4ds.had.co.nz/relational-data.html>

02. RIGHT OUTER JOIN

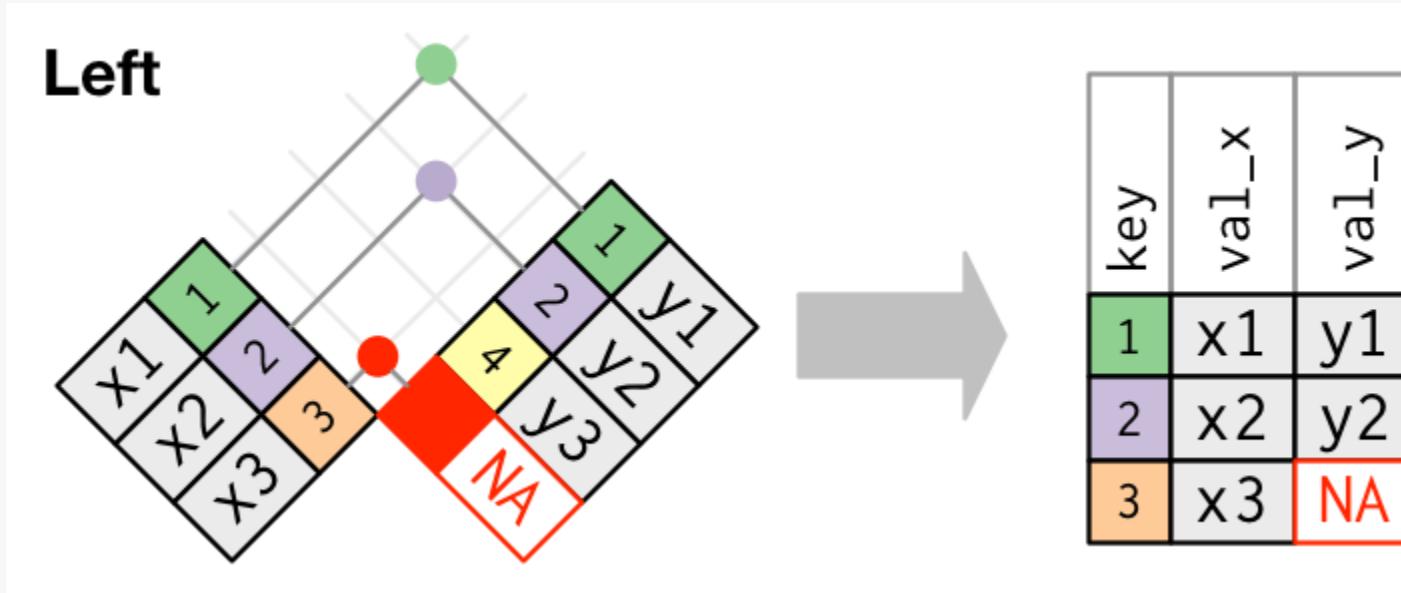
- RIGHT OUTER JOIN



출처: <https://r4ds.had.co.nz/relational-data.html>

03. LEFT OUTER JOIN

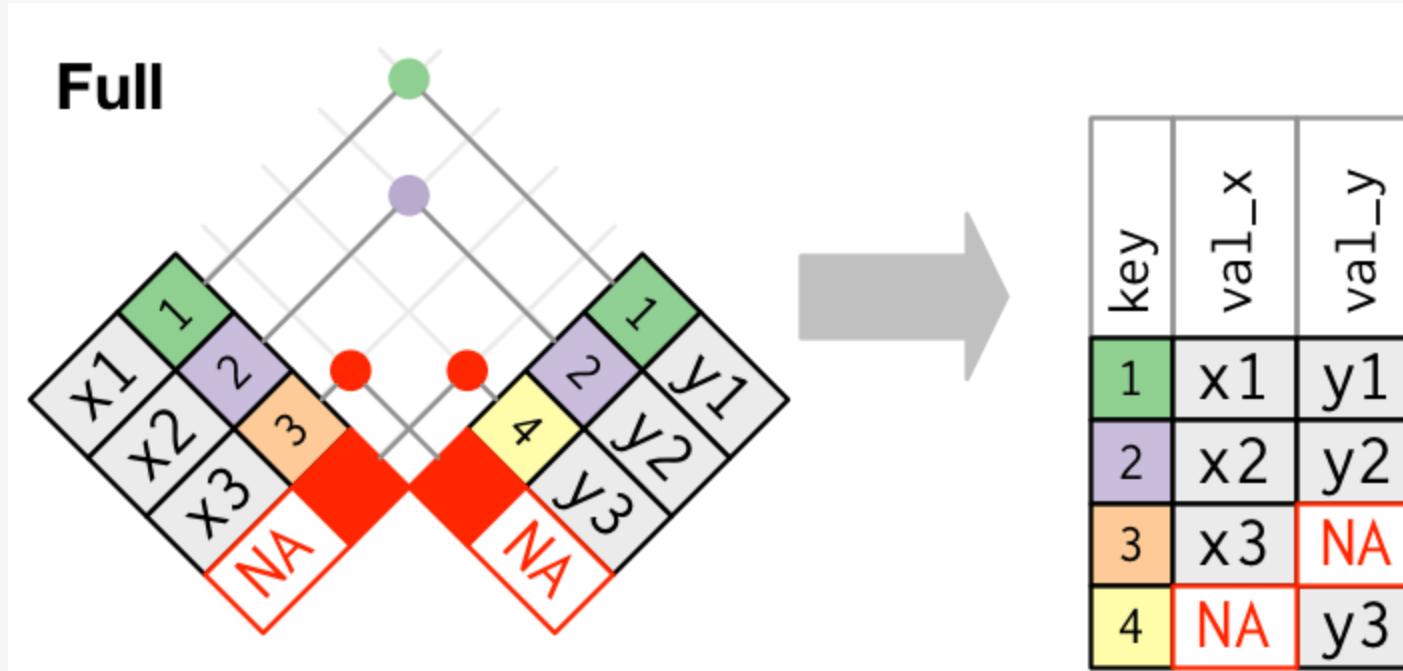
- LEFT OUTER JOIN



출처: <https://r4ds.had.co.nz/relational-data.html>

04. FULL JOIN

- FULL JOIN



출처: <https://r4ds.had.co.nz/relational-data.html>

05. 조인 문법의 종류

조인 문법의 종류	조인의 종류
ANSI/ISO SQL: 1999 standards	ON절을 사용한 JOIN
	LEFT/RIGHT/FULL OUTER JOIN
	USING절을 사용한 JOIN
	NATURAL JOIN
	CROSS JOIN

조인 문법의 종류	조인의 종류
오라클 조인	EQUI JOIN
	NON EQUI JOIN
	OUTER JOIN
	SELF JOIN

06. 집합 연산자

집합 연산자 작성 시 주의사항

UNION ALL 위쪽 쿼리와 아래쪽 쿼리 컬럼의 개수가 동일해야 한다.

UNION ALL 위쪽 쿼리와 아래쪽 쿼리 컬럼의 데이터 타입이 동일해야 한다.

결과로 출력되는 컬럼명은 위쪽 쿼리의 컬럼명으로 출력된다.

ORDER BY절은 제일 아래쪽 쿼리에만 작성할 수 있다.

UNION 연산자가 UNION ALL과 다른 점

중복된 데이터를 하나의 고유한 값으로 출력한다.

첫 번째 컬럼의 데이터를 기준으로 내림차순으로 정렬하여 출력한다.

오라클 DB에 로드하는 방법

with



01. CSV 데이터 오라클에 로드하는 방법

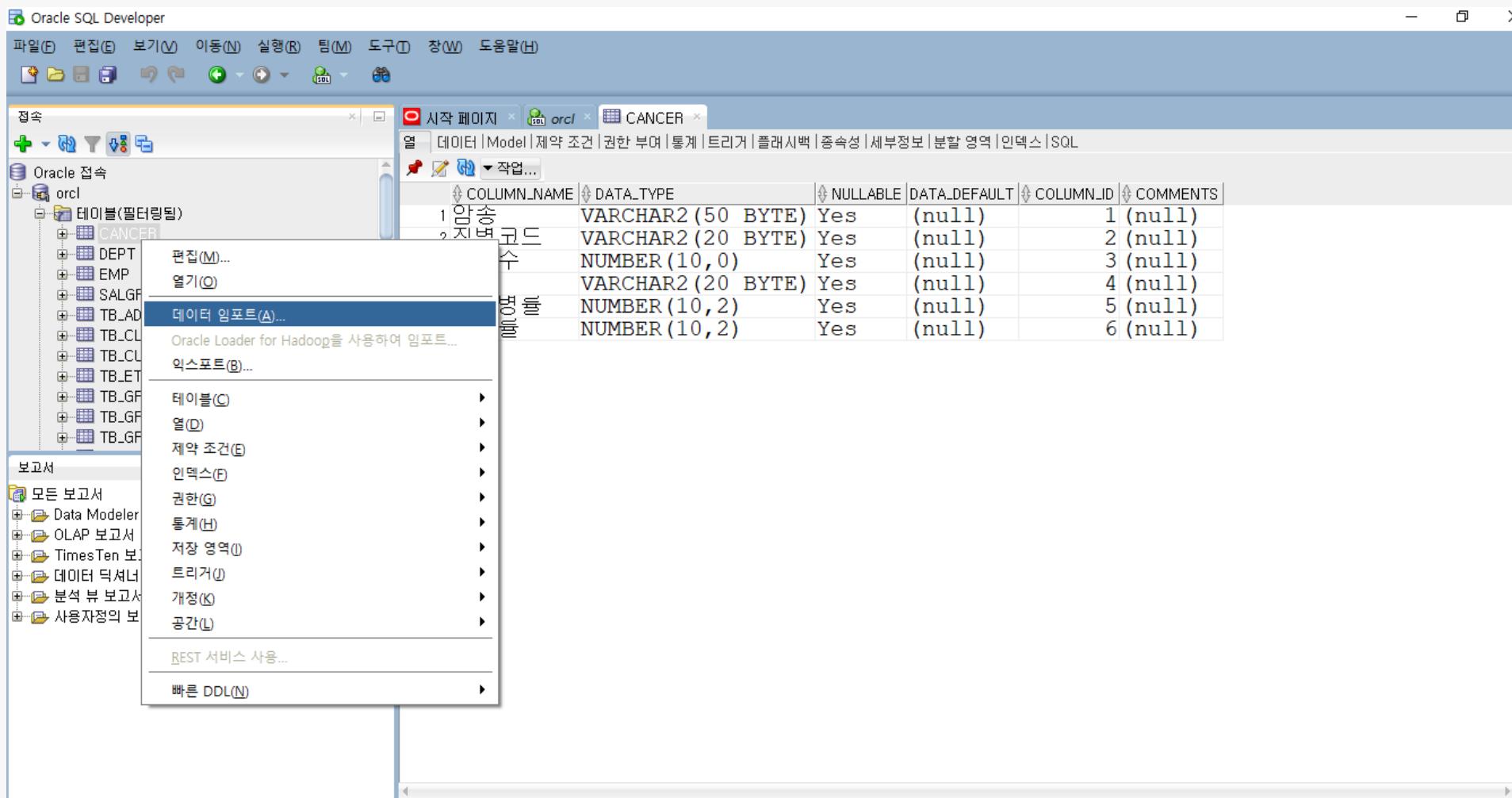
- 테이블 생성

```
DROP TABLE CANCER;
```

```
CREATE TABLE CANCER
(암종    VARCHAR2(50),
 질병코드  VARCHAR2(20),
 환자수   NUMBER(10),
 성별    VARCHAR2(20),
 조유병률 NUMBER(10,2),
 생존률   NUMBER(10,2));
```

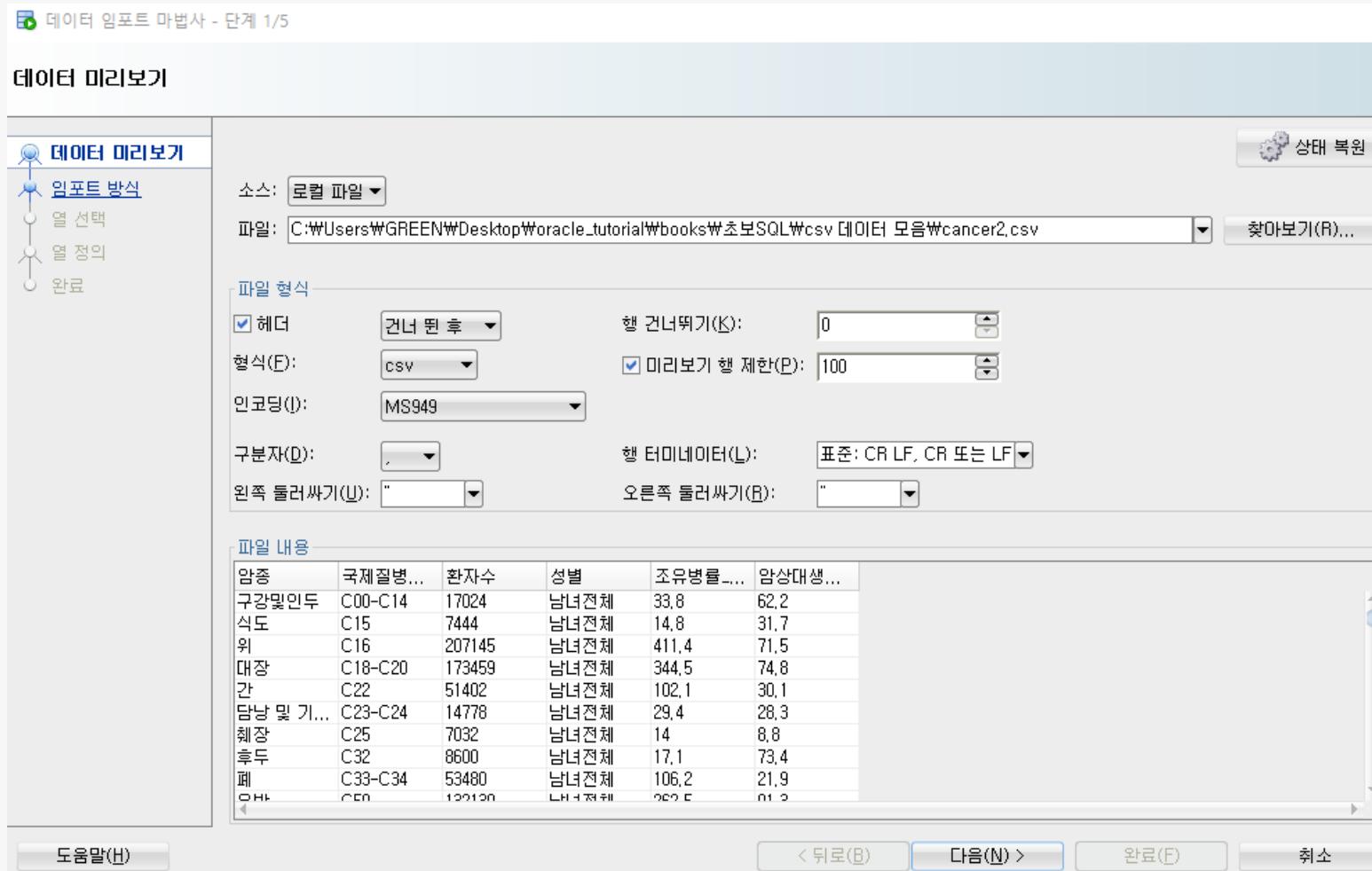
01. CSV 데이터 오라클에 로드하는 방법

- 데이터 임포트 가져오기



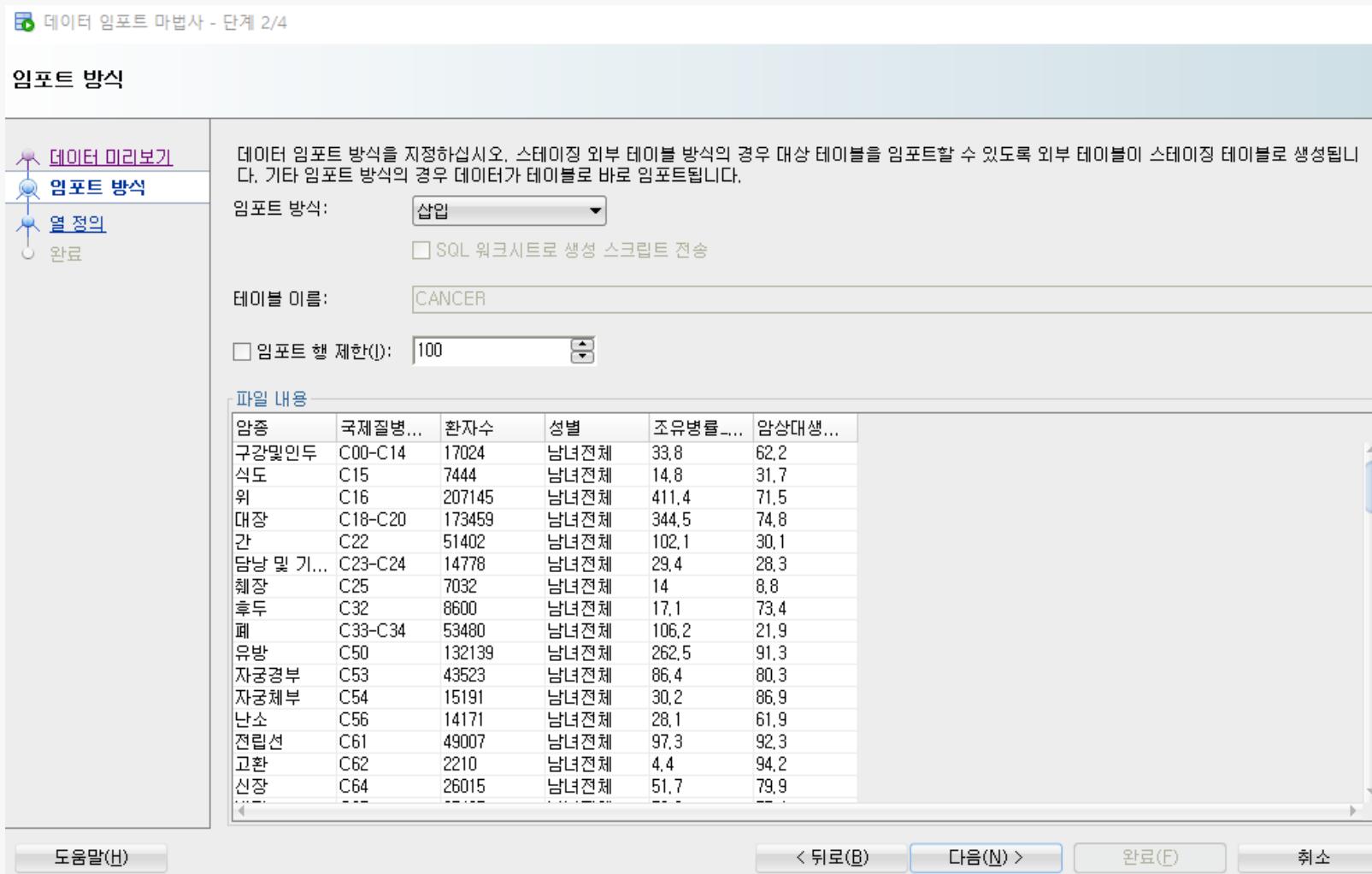
01. CSV 데이터 오라클에 로드하는 방법

- 데이터 임포트 가져오기



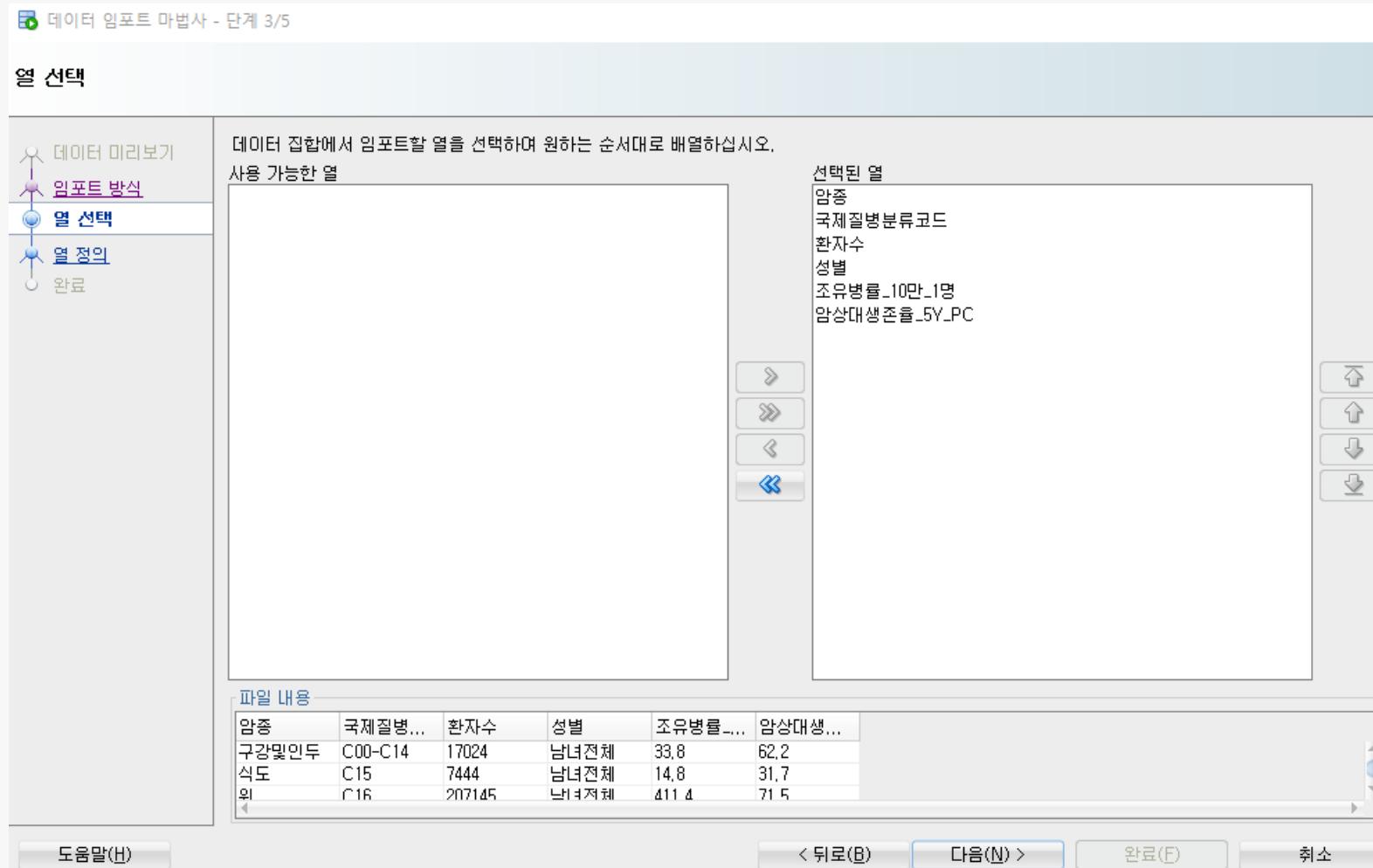
01. CSV 데이터 오라클에 로드하는 방법

- 데이터 임포트 가져오기



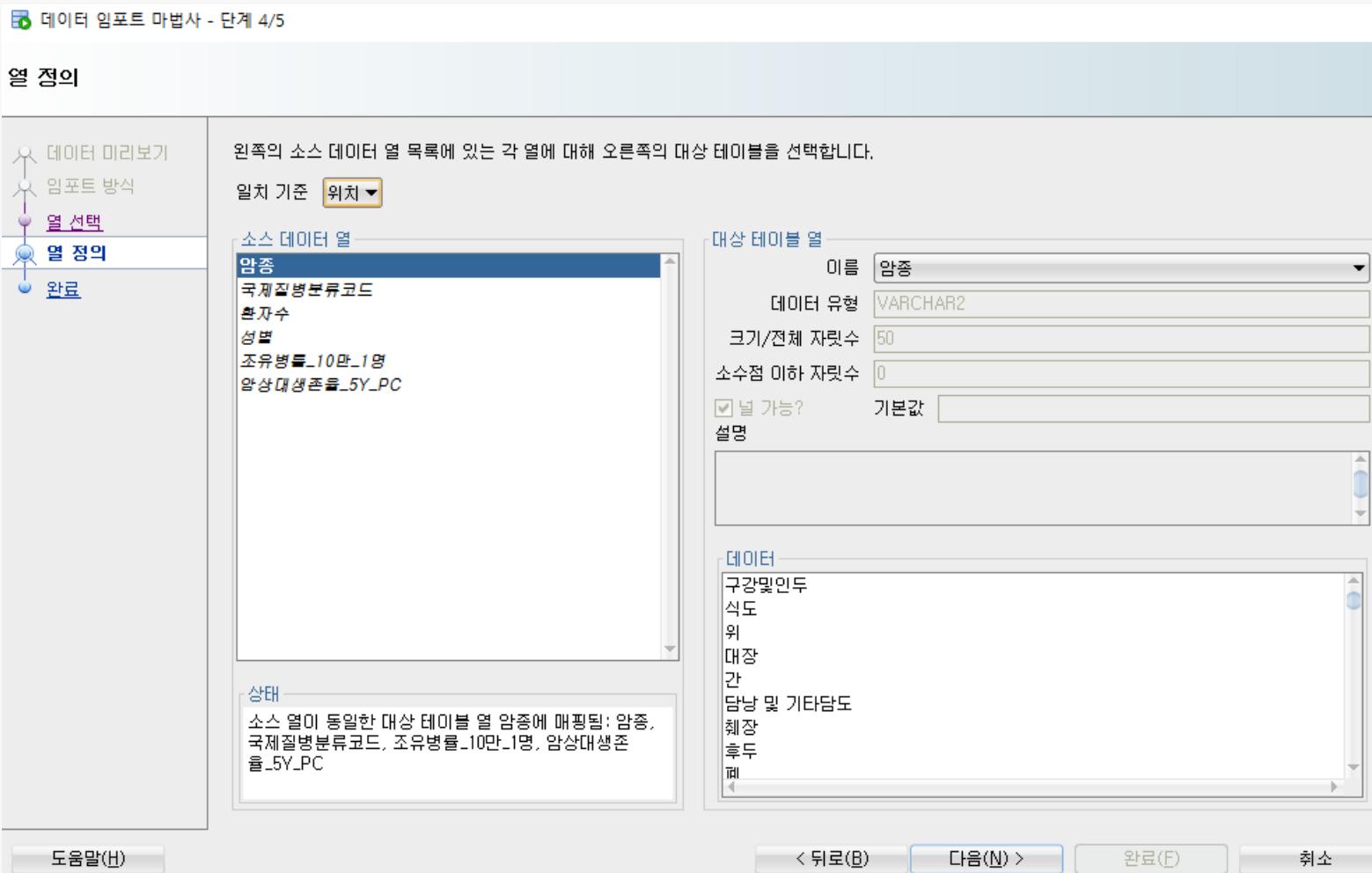
01. CSV 데이터 오라클에 로드하는 방법

- 데이터 임포트 가져오기



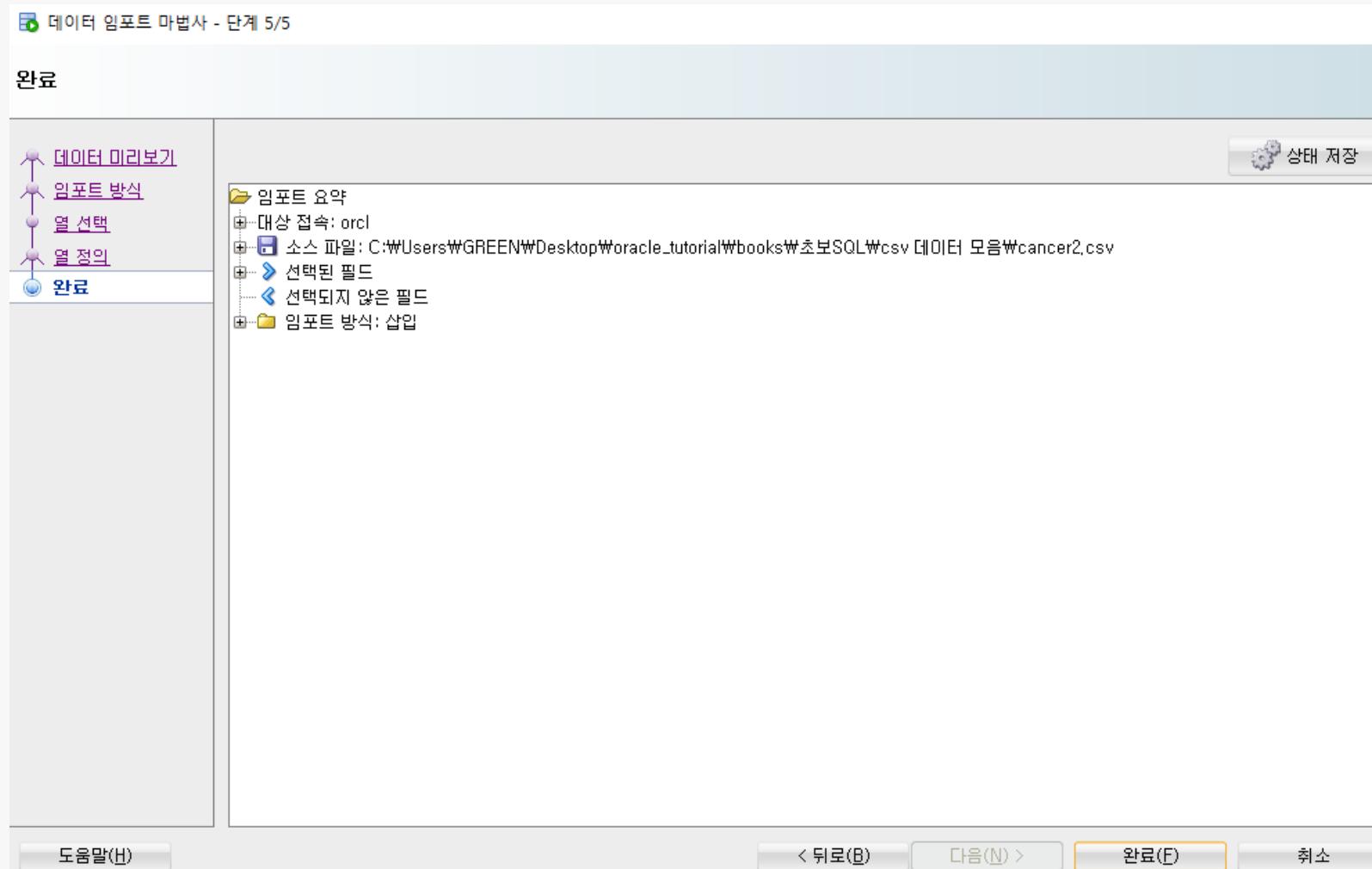
01. CSV 데이터 오라클에 로드하는 방법

- 이때 일치 기준은 위치로 정한다. (각 데이터마다 다르기 때문에 해당 옵션 선택 확인)



01. CSV 데이터 오라클에 로드하는 방법

- 이때 일치 기준은 위치로 정한다. (각 데이터마다 다르기 때문에 해당 옵션 선택 확인)



오라클 SQL 서브쿼리 문법

with



01. 서브쿼리의 종류 및 연산자

단일 행
서브쿼리

다중 행
서브쿼리

- 비교연산자(=, <, <=, >=, <>)
- IN, ANY, ALL, EXISTS 사용

02. 서브쿼리 종류 및 연산자

종류	설명
단일 행 서브 쿼리	서브 쿼리에서 메인 쿼리로 하나의 값이 반환됨
다중 행 서브 쿼리	서브 쿼리에서 메인 쿼리로 여러 개의 값이 반환됨
다중 컬럼 서브 쿼리	서브 쿼리에서 메인 쿼리로 여러 개의 컬럼 값이 반환됨

종류	연산자
단일 행 서브 쿼리	=, !=, >, <, >=, <=
다중 행 서브 쿼리	In, not in, >any, <any, >all, <all

03. 연산자 설명

연산자	설명
In	리스트의 값과 동일하다.
Not In	리스트의 값과 동일하지 않다.
>all	리스트에서 가장 큰 값보다 크다.
>any	리스트에서 가장 작은 값보다 크다
<all	리스트에서 가장 작은 값보다 작다
<any	리스트에서 가장 큰 값보다 작다.

04. SELECT문 6가지 절

SELECT문의 6가지 절	서브 쿼리 사용 여부	서브 쿼리 이름
SELECT	가능	스칼라(Scalar) 서브 쿼리
FROM	가능	IN LINE VIEW
WHERE	가능	서브 쿼리
GROUP BY	불가능	
HAVING	가능	서브 쿼리
ORDER BY	가능	스칼라(Scalar) 서브 쿼리

05. 쿼리 연습

- 2015년 전체 국가의 평균 수명을 계산하십시오.

05. 쿼리 연습

- 모든 데이터를 조회한다.
- 2015년 평균 기대 수명의 1.15배보다 높도록 설정한다.

▶ 질의 결과 ×

SQL | 인출된 모든 행: 5(0,002초)

	POP_ID	COUNTRY_CODE	YEAR	FERTILITY RATE	LIFE EXPECTANCY
1	21	AUS	2015	1.833	82.4512195121951
2	376	CHE	2015	1.54	83.1975609756098
3	356	ESP	2015	1.32	83.3804878048781
4	134	FRA	2015	2.01	82.6707317073171
5	170	HKG	2015	1.195	84.2780487804878

05. 쿼리 연습

- 서브 쿼리의 countires2 테이블에 있는 capital 필드를 사용한다.
- cities 테이블에서 name, country_code, urbanarea_pop 필드를 조회한다.

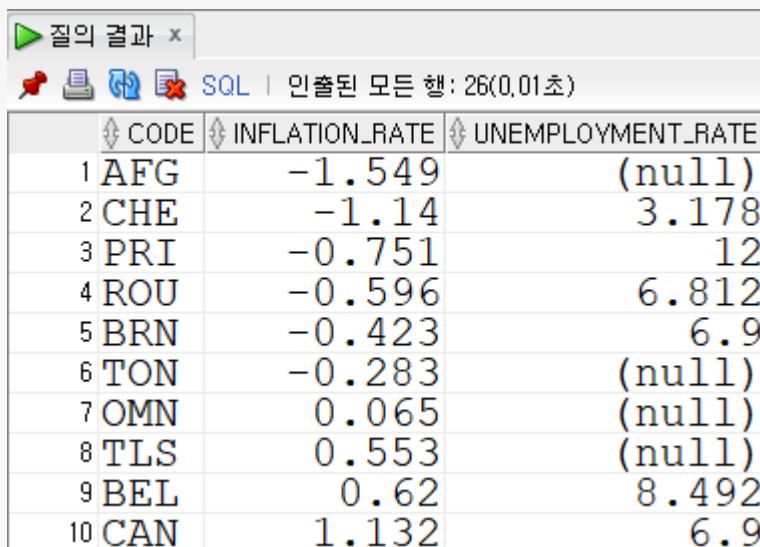


The screenshot shows a database query results window with the following details:

- Header: 질의 결과 x, SQL | 50개의 행이 인출됨(0.01초)
- Table Headers: NAME, COUNTRY_CODE, URBANAREA_POP
- Data Rows (10 rows):
 - 1 Beijing CHN 21516000
 - 2 Dhaka BGD 14543124
 - 3 Tokyo JPN 13513734
 - 4 Moscow RUS 12197596
 - 5 Cairo EGY 10230350
 - 6 Kinshasa COD 10130000
 - 7 Jakarta IDN 10075310
 - 8 Seoul KOR 9995784
 - 9 Mexico City MEX 8974724
 - 10 Lima PER 8852000

05. 쿼리 연습

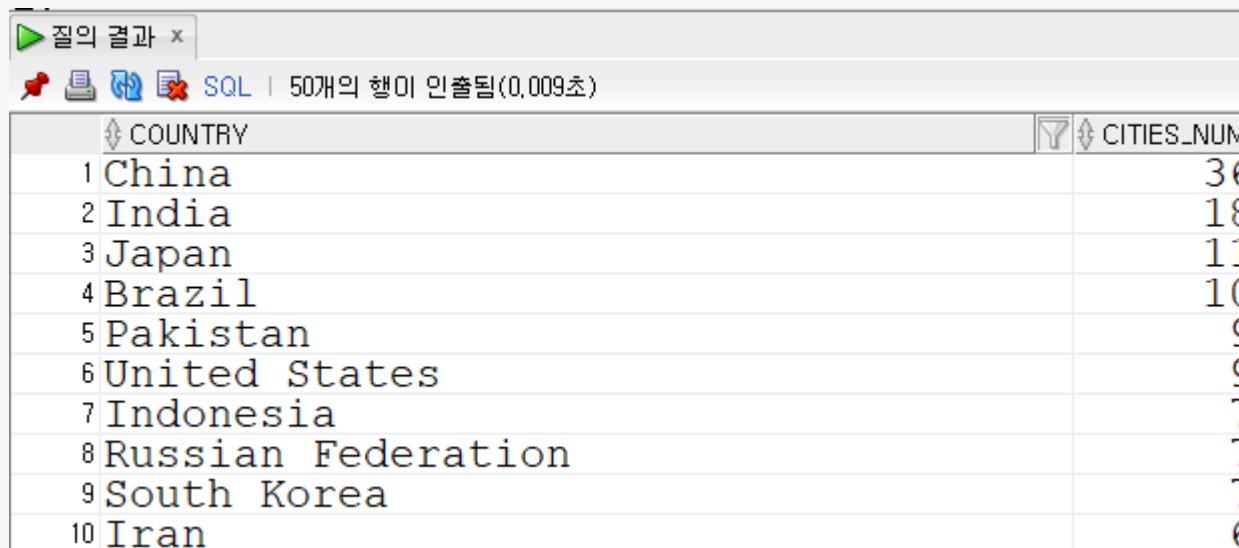
- economies 데이터에서 country code, inflation rate, unemployment rate를 조회한다.
- Inflation rate 오름차순으로 정렬한다. 연도는 2015년이다.
- Alias 사용하지 않는다.
- countries2 테이블 내 gov_form 칼럼에서 Constitutional Monarchy 또는 'Republic'이 들어간 국가는 제외한다.



	CODE	INFLATION_RATE	UNEMPLOYMENT_RATE
1	AFG	-1.549	(null)
2	CHE	-1.14	3.178
3	PRI	-0.751	12
4	ROU	-0.596	6.812
5	BRN	-0.423	6.9
6	TON	-0.283	(null)
7	OMN	0.065	(null)
8	TLS	0.553	(null)
9	BEL	0.62	8.492
10	CAN	1.132	6.9

05. 쿼리 연습

- 첫번째 주석을 풀고 실행합니다.
- GROUP BY 코드를 변환하여 SELECT 내부의 하위 쿼리를 사용한다. 첫번째, 쿼리에서 GROUP BY 코드를 사용하여 주어진 결과와 일치하는 코드를 작성한다.
- 다시 city_num 내림차순으로 결과를 정렬한 후, code 오름차순으로 정렬한다.



The screenshot shows a database query results window with the following details:

- WindowTitle: 질의 결과
- Toolbar Buttons: Refresh, Stop, SQL, and a status message: 50개의 행이 인출됨(0.009초)
- Table Headers: COUNTRY and CITIES_NUM
- Table Data:

COUNTRY	CITIES_NUM
1 China	36
2 India	18
3 Japan	11
4 Brazil	10
5 Pakistan	9
6 United States	9
7 Indonesia	7
8 Russian Federation	7
9 South Korea	7
10 Iran	6

05. 쿼리 연습

- 2015년의 continent 그룹별로 하여 continent, inflation_rate의 최댓값을 조회한다.
- 각 대륙별 inflation_rate가 가장 높은 나라를 추출하는 코드를 작성하도록 한다.

#	NAME	CONTINENT	INFLATION_RATE
1	Afghanistan	Asia	-1.549
2	Netherlands	Europe	0.22
3	Albania	Europe	1.896
4	Algeria	Africa	4.784
5	Angola	Africa	10.287
6	Antigua and Barbuda	North America	0.969
7	United Arab Emirates	Asia	4.07
8	Argentina	South America	(null)
9	Armenia	Asia	3.731
10	Australia	Oceania	1.461

#	CONTINENT	MAX_INF
1	North America	7.524
2	Oceania	9.784
3	Africa	21.858
4	Asia	39.403
5	Europe	48.684
6	South America	121.738

#	NAME	CONTINENT	INFLATION_RATE
1	Haiti	North America	7.524
2	Yemen	Asia	39.403
3	Malawi	Africa	21.858
4	Nauru	Oceania	9.784
5	Ukraine	Europe	48.684
6	Venezuela	South America	121.738

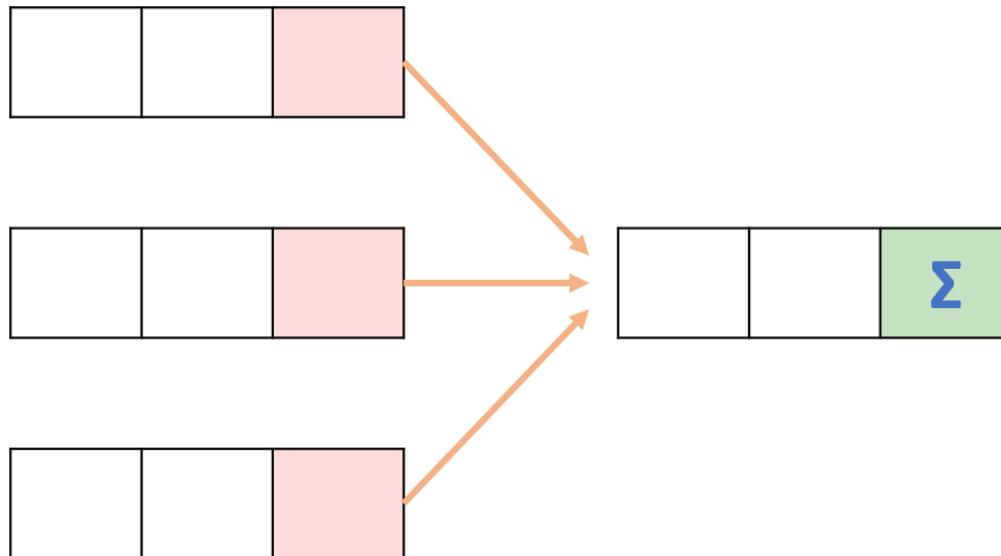
오라클 SQL 윈도우 함수 문법

with

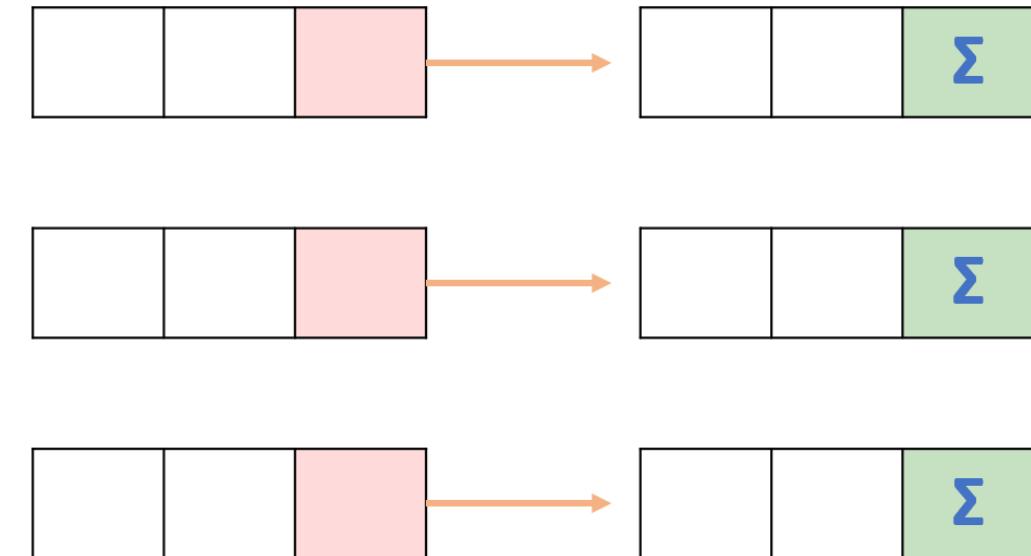


01. 기본 개념

Aggregation



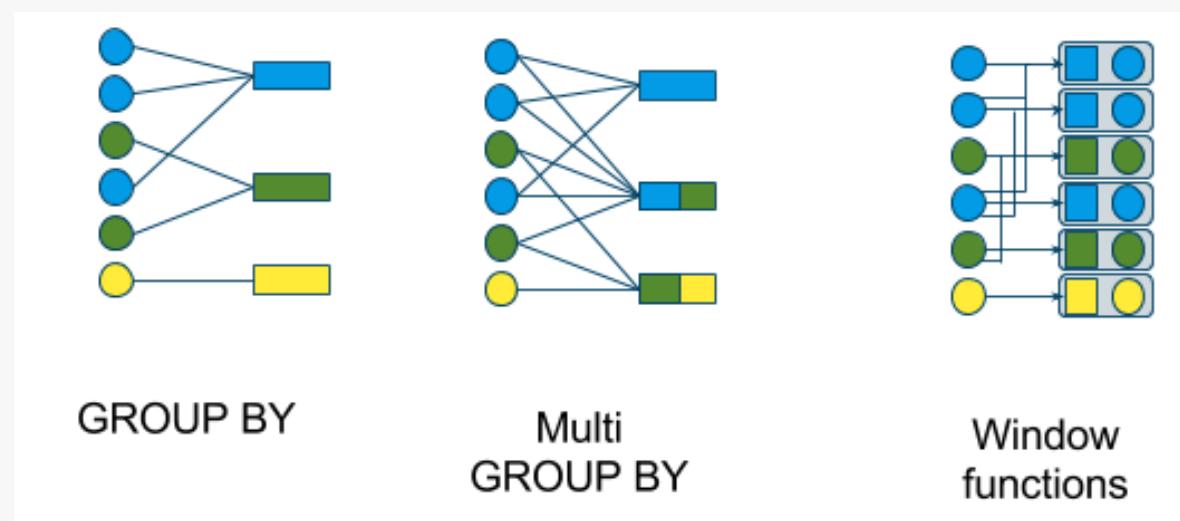
Window Function



출처: <https://towardsdatascience.com/sql-window-function-demonstrated-with-real-interview-questions-from-leetcode-e83e28edaabc>

01. 기본 개념

- 현재 행과 관련된 행 집합에서 쿼리 작업 수행
- GROUP BY 집계 함수와 유사하지만, 결괏값에서 모든 행이 존재 할 수
- Ranking, 누적 계산, 이동 평균 계산 진행 가능



출처: <https://sundaskhalid.medium.com/sql-window-function-vs-group-by-b246d14223d2>

02. 쿼리 연습

- 각 행에 숫자를 1, 2, 3, …, N 행태로 추가한다.
- ROW_N 기준으로 오름차순으로 진행한다.

▶ 질의 결과 x

SQL | 50개의 행이 인출됨(0.002초)

ROW_N	YEAR	CITY	SPORT	DISCIPLINE	ATHLETE	COUNTRY	GENDER	EVENT	MEDAL
1	1900	Paris	Athletics	Athletics	SHELDON Richard	USA	Men	Discus Throw	Bronze
2	1900	Paris	Athletics	Athletics	BAUER Rudolf	HUN	Men	Discus Throw	Gold
3	1900	Paris	Athletics	Athletics	JANDA Frantisek	BOH	Men	Discus Throw	Silver
4	1900	Paris	Athletics	Athletics	MCCRACKEN Josiah	USA	Men	Hammer Throw	Bronze
5	1900	Paris	Athletics	Athletics	FLANAGAN John Jesus	USA	Men	Hammer Throw	Gold
6	1900	Paris	Athletics	Athletics	HARE Thomas Truxton	USA	Men	Hammer Throw	Silver
7	1900	Paris	Athletics	Athletics	GNCZY Lajos	HUN	Men	High Jump	Bronze
8	1900	Paris	Athletics	Athletics	BAXTER Irving	USA	Men	High Jump	Gold
9	1900	Paris	Athletics	Athletics	LEAHY Patrick Joseph	GBR	Men	High Jump	Silver
10	1900	Paris	Athletics	Athletics	SHELDON Lewis Pendleton	USA	Men	High Jump Standing	Bronze

02. 쿼리 연습

- 하계 올림픽이 열린 각 연도에 번호를 할당한다.
- 가장 최근 연도를 가진 행이 더 낮은 숫자를 갖도록 한다.

질의 결과 x

SQL | 인출!

	YEAR	ROW_N
1	1896	1
2	1900	2
3	1904	3
4	1908	4
5	1912	5
6	1920	6
7	1924	7
8	1928	8
9	1932	9
10	1936	10

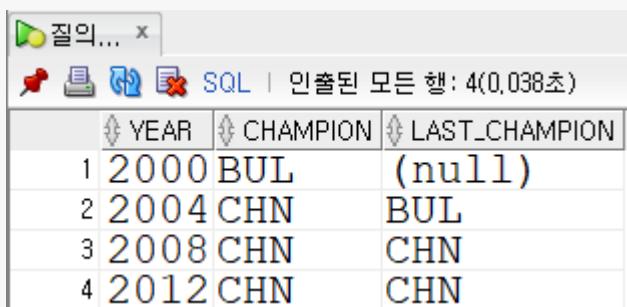
02. 쿼리 연습

- 각 운동선수들이 획득한 메달 개수를 내림차순으로 정렬 한다.
- 각 선수들의 랭킹을 추가한다.

	MEDALS	ATHLETE	ROW_N
1	22	PHELPS Michael	1
2	18	LATYNINA Larisa	2
3	15	ANDRIANOV Nikolay	3
4	13	ONO Takashi	4
5	13	SHAKHLIN Boris	5
6	13	MANGIAROTTI Edoardo	6
7	12	NURMI Paavo	7
8	12	COUGHLIN Natalie	8
9	12	THOMPSON Jenny	9
10	12	FISCHER Birgit	10

02. 쿼리 연습

- 남자 69KG 역도 경기에서 매년 금메달리스트 조회하도록 한다.
 - Discipline: Weightlifting
 - Event: 69kg
 - Gender: Men
 - Medal: Gold
- 기존 쿼리에서 매년 전년도 챔피언도 같이 조회하도록 한다
 - 이 때, LAG() 함수를 사용한다.



The screenshot shows a MySQL Workbench interface with a query editor window. The title bar says '질의...' (Query). The toolbar includes icons for connection, schema, and SQL. The status bar indicates 'SQL | 인출된 모든 행: 4(0.038초)'. The main area displays a table with four rows and three columns: YEAR, CHAMPION, and LAST_CHAMPION. The data is as follows:

	YEAR	CHAMPION	LAST_CHAMPION
1	2000	BUL	(null)
2	2004	CHN	BUL
3	2008	CHN	CHN
4	2012	CHN	CHN

03. PARTITION BY

- PARTITION BY는 테이블을 각 칼럼의 값에 따라서 분할을 합니다.
- ROW_NUMBER는 각 파티션에 따라 새롭게 정의를 내립니다.
- LAG는 이전 행이 동일한 파티션에 있는 경우에만 행의 이전 값을 가져옵니다.

03. PARTITION BY

Query

```
WITH Discus_Gold_Medal AS (
    SELECT
        Year, Event, Country AS Champion
    FROM summer_medals
    WHERE
        Year IN (2004, 2008, 2012)
        AND Gender = 'Men' AND Medal = 'Gold'
        AND Event IN ('Discus Throw', 'Triple Jump')
        AND Gender = 'Men')
```

```
SELECT
    YEAR, Event, Champion,
    LAG(Champion) OVER
        (ORDER BY Event ASC, Year ASC) AS Last_Champion
FROM Discus_Gold_Medal
ORDER BY Event ASC, Year ASC;
```

Result

YEAR	EVENT	CHAMPION	LAST_CHAMPION
1 2004	Discus Throw	LTU	(null)
2 2008	Discus Throw	EST	LTU
3 2012	Discus Throw	GER	EST
4 2004	Triple Jump	SWE	GER
5 2008	Triple Jump	POR	SWE
6 2012	Triple Jump	USA	POR

03. PARTITION BY

Query

```
WITH Discus_Gold_Medal AS (
    SELECT
        Year, Event, Country AS Champion
    FROM summer_medals
    WHERE
        Year IN (2004, 2008, 2012)
        AND Gender = 'Men' AND Medal = 'Gold'
        AND Event IN ('Discus Throw', 'Triple Jump')
        AND Gender = 'Men')
```

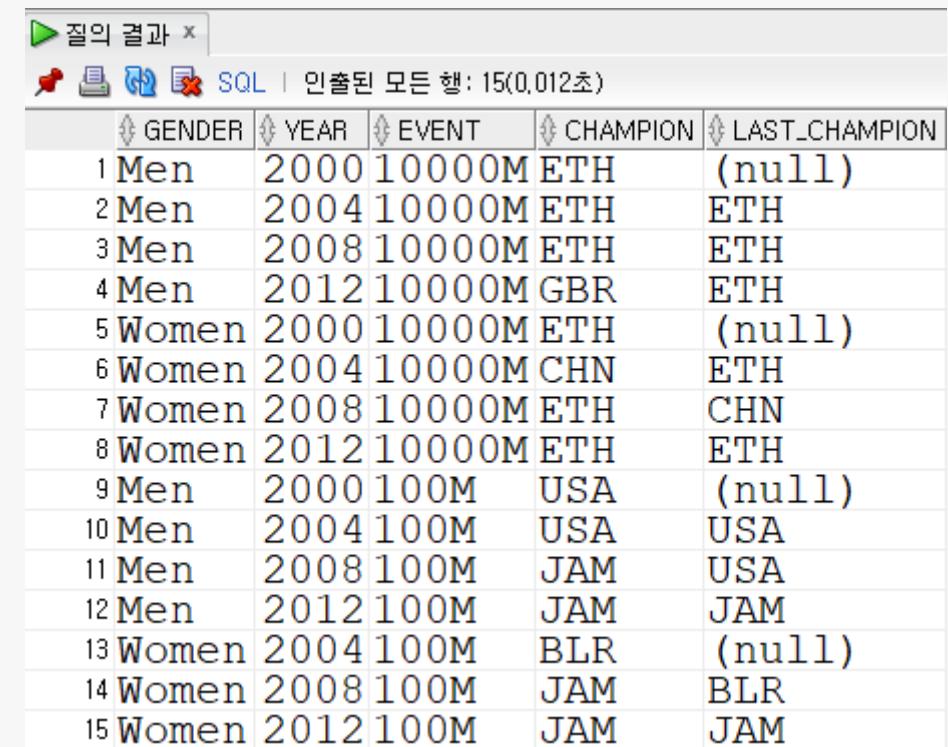
```
SELECT
    YEAR, Event, Champion,
    LAG(Champion) OVER
    (PARTITION BY Event ORDER BY Event ASC, Year ASC) AS
    Last_Champion
FROM Discus_Gold_Medal
ORDER BY Event ASC, Year ASC;
```

Result

YEAR	EVENT	CHAMPION	LAST_CHAMPION
1 2004	Discus Throw	LTU	(null)
2 2008	Discus Throw	EST	LTU
3 2012	Discus Throw	GER	EST
4 2004	Triple Jump	SWE	(null)
5 2008	Triple Jump	POR	SWE
6 2012	Triple Jump	USA	POR

04. 쿼리 연습

- 성별에 따라 각 이벤트의 이전 챔피언을 반환한다.
- 성별에 따른 현재 챔피언과 이전 챔피언을 반환한다.
- 이번에는 경기종목을 2개 추가한다.
- ('100M', '10000M')
- 이 때에는 IN() 함수를 사용한다.



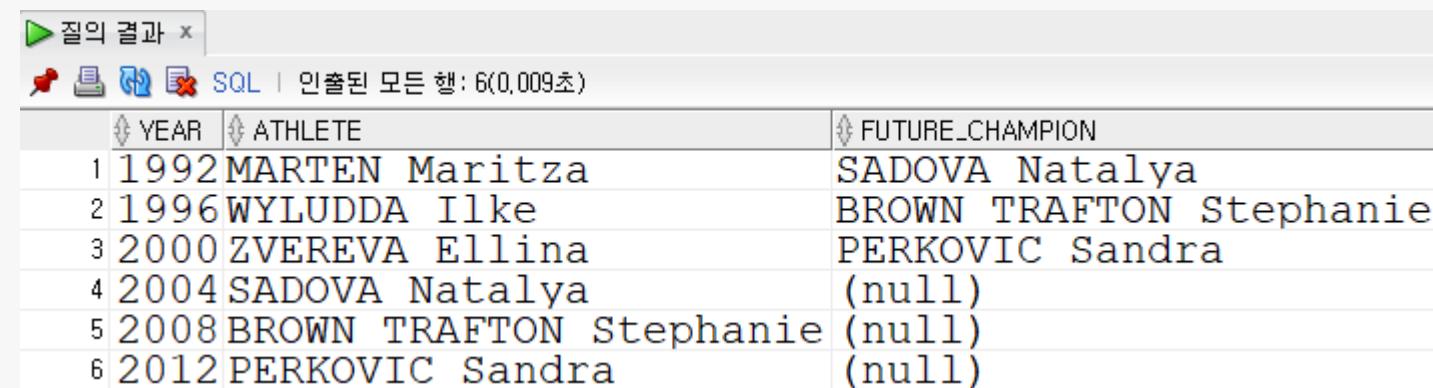
	GENDER	YEAR	EVENT	CHAMPION	LAST_CHAMPION
1	Men	2000	10000M	ETH	(null)
2	Men	2004	10000M	ETH	ETH
3	Men	2008	10000M	ETH	ETH
4	Men	2012	10000M	GBR	ETH
5	Women	2000	10000M	ETH	(null)
6	Women	2004	10000M	CHN	ETH
7	Women	2008	10000M	ETH	CHN
8	Women	2012	10000M	ETH	ETH
9	Men	2000	100M	USA	(null)
10	Men	2004	100M	USA	USA
11	Men	2008	100M	JAM	USA
12	Men	2012	100M	JAM	JAM
13	Women	2004	100M	BLR	(null)
14	Women	2008	100M	JAM	BLR
15	Women	2012	100M	JAM	JAM

05. Relative vs Absolute

- Relative
 - ✓ LAG(column, n) LAG()는 현재 행 앞의 행에 있는 열의 값을 반환합니다.
 - ✓ Lead(column, n) 현재 행 뒤의 행에 있는 열의 값을 반환합니다.
- Absolute
 - ✓ FIRST_VALUE(column) 테이블 또는 파티션의 첫번째 값을 반환합니다.
 - ✓ LAST_VALUE(column) 테이블 또는 파티션의 마지막 값을 반환합니다.

06. 쿼리 연습

- 각 연도마다 현재 금메달리스트와 3개 대회 이후의 메달리스트를 같이 조회합니다.



The screenshot shows a database query results window with the following data:

YEAR	ATHLETE	FUTURE_CHAMPION
1992	MARTEN Maritza	SADOVA Natalya
1996	WYLUDDA Ilke	BROWN TRAFTON Stephanie
2000	ZVEREVA Ellina	PERKOVIC Sandra
2004	SADOVA Natalya	(null)
2008	BROWN TRAFTON Stephanie	(null)
2012	PERKOVIC Sandra	(null)

06. 쿼리 연습

- 이번에는 FIRST_VALUE를 활용하여 모든 선수를 조회한다.
- 동시에 첫번째 선수를 조회하도록 합니다.
- 모든 선수 조회 시, 알파벳 순으로 정렬합니다.
- Medal = 'Gold'
- Gender = 'Men'

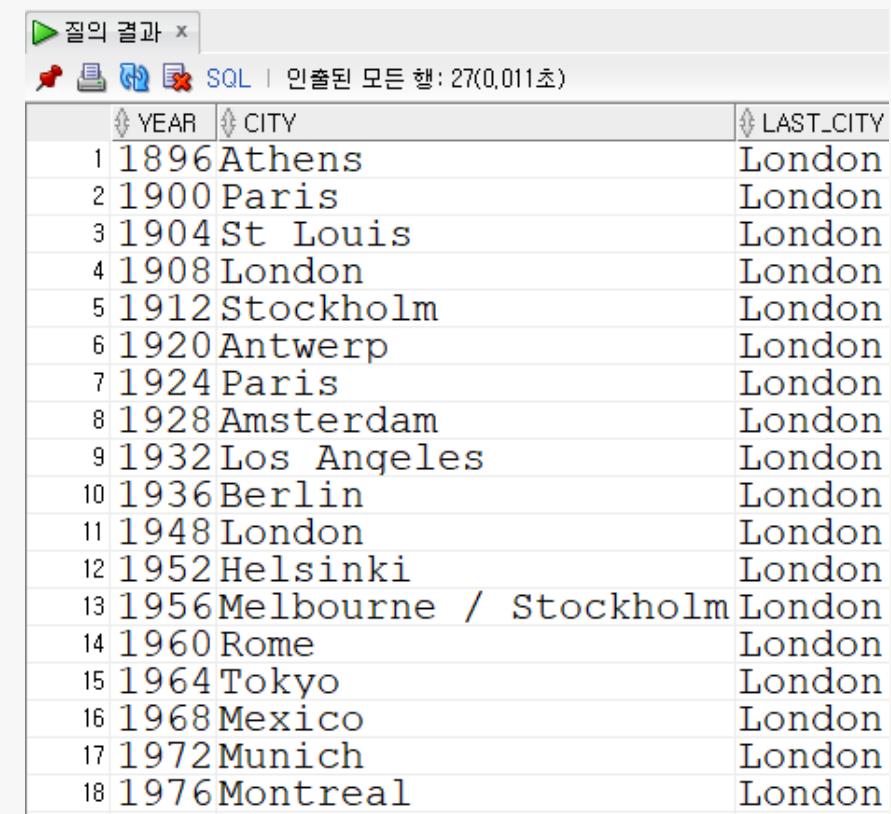
질의 결과 x

SQL | 50개의 행이 인출됨(0,018초)

ATHLETE	FIRST_ATHLETE
1 AABYE Edgar	AABYE Edgar
2 AALTONEN Paavo Johannes	AABYE Edgar
3 AAS Thomas Valentin	AABYE Edgar
4 ABALMASAU Aliaksei	AABYE Edgar
5 ABALO Luc	AABYE Edgar
6 ABANDA ETONG Patrice	AABYE Edgar
7 ABARCA Jose Maria	AABYE Edgar
8 ABASCAL GARCIA Alejandro	AABYE Edgar
9 ABATI Joel	AABYE Edgar
10 ABBAGNALE Agostino	AABYE Edgar
11 ABBAGNALE Carmine	AABYE Edgar
12 ABBAGNALE Giuseppe	AABYE Edgar
13 ABDOULLAYEV Mahamatkodir	AABYE Edgar
14 ABDUL Hamid	AABYE Edgar
15 ABDUL Rashid I	AABYE Edgar
16 ABDUL Rashid III	AABYE Edgar
17 ABDUL Rashid IV	AABYE Edgar
18 ABDUL Waheed	AABYE Edgar

06. 쿼리 연습

- 각 올림픽 경기가 열렸던 해외 도시를 조회합니다.
- 본 데이터에서 올림픽 경기가 열렸던 마지막 도시를 조회합니다.
- LAST_VALUE()



The screenshot shows a database query results window with the following details:

- WindowTitle: 질의 결과 x
- Toolbar Buttons: Refresh, Save, Print, SQL (selected), Exit
- Text: SQL | 인출된 모든 행: 27(0,011초)
- Table Headers: YEAR, CITY, LAST_CITY
- Table Data:

YEAR	CITY	LAST_CITY
1	1896 Athens	London
2	1900 Paris	London
3	1904 St Louis	London
4	1908 London	London
5	1912 Stockholm	London
6	1920 Antwerp	London
7	1924 Paris	London
8	1928 Amsterdam	London
9	1932 Los Angeles	London
10	1936 Berlin	London
11	1948 London	London
12	1952 Helsinki	London
13	1956 Melbourne / Stockholm	London
14	1960 Rome	London
15	1964 Tokyo	London
16	1968 Mexico	London
17	1972 Munich	London
18	1976 Montreal	London

07. Ranking 함수

- ROW_NUMBER()는 두 행의 값이 동일한 경우에도 항상 고유한 번호를 할당한다.
- RANK() 동일한 값을 가진 행에 같은 번호를 할당하고 이러한 경우 다음 숫자를 건너뛴다.
- DENSE_RANK() 값이 동일한 행에 같은 숫자를 할당함. 다음 숫자를 건너뛰지 않는다.

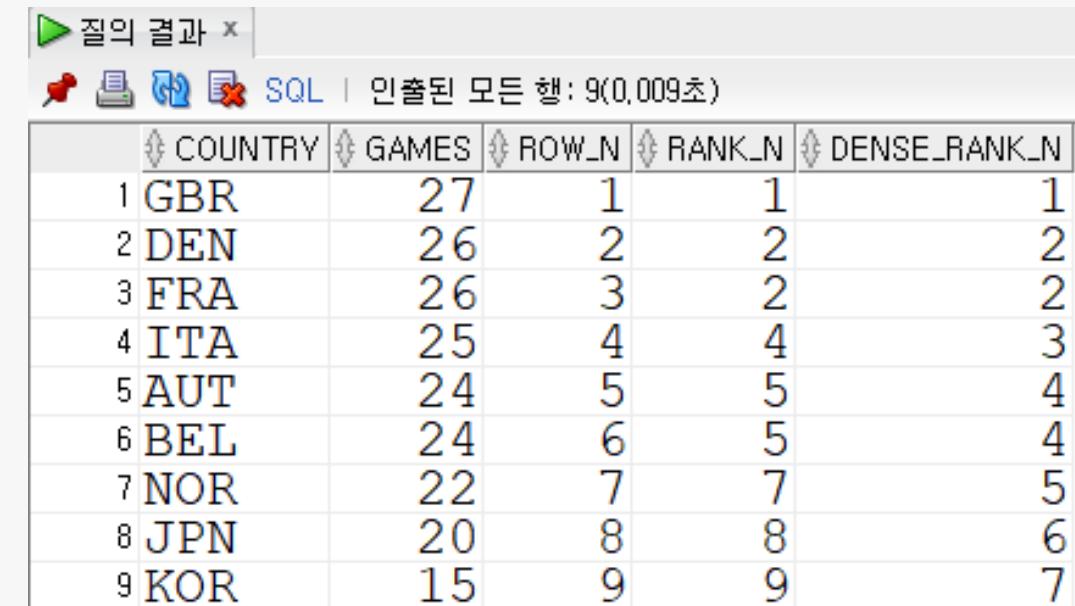
```
SELECT
    Country, COUNT(DISTINCT Year) AS Games
FROM summer_medals
Where
    Country IN ('GBR', 'DEN', 'FRA',
                 'ITA', 'AUT', 'BEL',
                 'NOR', 'JPN', 'KOR')
GROUP BY Country
ORDER BY GAMES DESC;
```

	COUNTRY	GAMES
1	GBR	27
2	FRA	26
3	DEN	26
4	ITA	25
5	BEL	24
6	AUT	24
7	NOR	22
8	JPN	20
9	KOR	15

07. Ranking 함수

```
WITH Country_Games AS (
    SELECT
        Country, COUNT(DISTINCT Year) AS Games
    FROM summer_medals
    WHERE
        Country IN ('GBR', 'DEN', 'FRA',
                    'ITA', 'AUT', 'BEL',
                    'NOR', 'JPN', 'KOR')
    GROUP BY Country
    ORDER BY GAMES DESC)

SELECT
    Country, Games,
    ROW_NUMBER()
        OVER (ORDER BY Games DESC) AS Row_N,
    RANK()
        OVER (ORDER BY Games DESC) AS Rank_N,
    DENSE_RANK()
        OVER (ORDER BY Games DESC) AS Dense_Rank_N
FROM Country_Games
ORDER BY Games DESC, Country ASC;
```



The screenshot shows a SQL query results window with the following details:

- Toolbar icons: Run, Stop, Refresh, SQL, and a status message "인출된 모든 행: 9(0.009초)".
- Table structure:
 - Columns: COUNTRY, GAMES, ROW_N, RANK_N, DENSE_RANK_N.
 - Row 1: GBR, 27, 1, 1, 1.
 - Row 2: DEN, 26, 2, 2, 2.
 - Row 3: FRA, 26, 3, 2, 2.
 - Row 4: ITA, 25, 4, 4, 3.
 - Row 5: AUT, 24, 5, 5, 4.
 - Row 6: BEL, 24, 6, 5, 4.
 - Row 7: NOR, 22, 7, 7, 5.
 - Row 8: JPN, 20, 8, 8, 6.
 - Row 9: KOR, 15, 9, 9, 7.

08. 쿼리연습

- 이번에는 국가별 Gold 메달 개수를 알파벳 순서대로 작업을 하도록 한다.
- 이 때, Gold 메달 누적 개수를 같이 조회하도록 한다.

질의 결과 ×

SQL | 50개의 행이 인출됨(0.005초)

	COUNTRY	MEDALS	CUM_MEDALS
1	ALG	2	2
2	ARG	47	49
3	AUS	159	208
4	AUT	6	214
5	AZE	6	220
6	BAH	11	231
7	BEL	2	233
8	BLR	16	249
9	BRA	46	295
10	BUL	8	303
11	CAN	19	322
12	CHI	3	325
13	CHN	221	546
14	CMR	20	566
15	COL	2	568
16	CRO	30	598

08. 쿼리연습

- 이번에는 한국과 일본 국가만 조회될 수 있도록 쿼리 작성한다.
- 조회된 기록 중, 가장 많은 메달을 기록 갯수가 나타나도록 한다.

▶ 질의 결과 x

SQL | 인출된 모든 행: 8(0.007초)

	COUNTRY	YEAR	MEDALS	MAX_MEDALS
1	JPN	2000	5	5
2	JPN	2004	21	21
3	JPN	2008	23	23
4	JPN	2012	7	23
5	KOR	2000	12	12
6	KOR	2004	14	14
7	KOR	2008	41	41
8	KOR	2012	18	41

08. 쿼리연습

- 국가별 금메달을 기준으로 랭킹을 구하는 쿼리 작성

질의 결과 x

SQL | 인출된 모든 행: 9(0.011초)

	COUNTRY	YEAR	MEDAL_CNT	RANK
1	FRA	2004	21	2
2	FRA	2008	25	3
3	FRA	2012	30	3
4	GBR	2004	17	3
5	GBR	2008	31	2
6	GBR	2012	48	1
7	GER	2004	41	1
8	GER	2008	42	1
9	GER	2012	45	2

오라클 PL/SQL

with



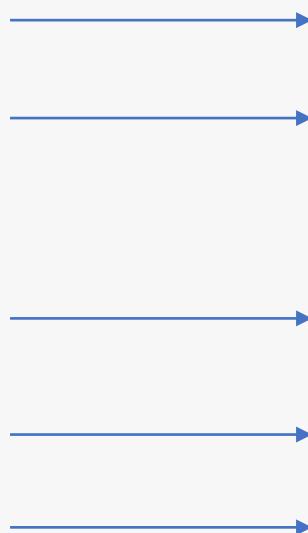
PL/SQL
ORACLE®

01. 기본 개념

- PL/SQL: Procedure Language SQL의 약자
- SQL을 확장한 절차적으로 처리하게 하는 DB 프로그래밍 언어
- PL/SQL 프로그램의 종류
 - ✓ Procedure
 - ✓ Function
 - ✓ Trigger
- 이름이 없는 익명 블록(Anonymous Block)

01. 기본 구조

- ✓ 인자값의 값 화면 출력
- ✓ accept: sqlplus 명령어
- ✓ v_sum: 로컬 변수
- ✓ := 할당 연산자
- ✓ dbms_output: 화면 출력 패키지
- ✓ put_line: dbms_output 패키지 내 함수



```
set serveroutput on  
accept p_num1 prompt '첫번째 숫자를 입력하세요 ~ '  
accept p_num2 prompt '두번째 숫자를 입력하세요 ~ '  
  
declare  
    v_sum number(10);  
begin  
    v_sum := &p_num1 + &p_num2 ;  
    dbms_output.put_line('총합은: ' || v_sum );  
end;  
/
```

영역	설명	옵션/필수
DECLARE (선언부)	PL/SQL에서 사용하는 모든 변수나 상수 선언	옵션
BEGIN (실행부)	제어문, 반복문, 함수 정의 로직 기술	필수
EXCEPTION (예외 처리부)	에러 발생 시, 예외 처리문	옵션
END (실행문 종료)	PL/SQL 블록 종료	필수

02. 데이터 타입

데이터 타입	설명
NATURAL	PLS_INTEGER 중 음수 제외 (0 포함)
NATURALN	PLS_INTEGER 중 음수 제외인데 NULL 할당 불가, 반드시 선언 시 초기화 필요
POSITIVE	PLS_INTEGER 중 양수 (0 미포함)
POSITIVEN	PLS_INTEGER 중 양수인데 NULL 할당 불가, 반드시 선언 시 초기화 필요
SIGNTYPE	PLS_INTEGER 중 -1, 0, 1
SIMPLE_INTEGER	PLS_INTEGER 중 NULL이 아닌 모든 값, 반드시 선언 시 초기화 필요

03. 연산자 우선순위

연산자	설명
**	제곱 연산자
+, -	양수, 음수 식별 연산자
*, /	곱셈, 나눗셈
+, -,	덧셈, 뺄셈, 문자열 연결 연산자
=, <, >, <=, >=, <>, !=, IS NULL, LIKE, BETWEEN, IN	비교 연산자
NOT	논리 연산자
AND	논리 연산자
OR	논리 연산자

04. 데이터 타입 최대 크기

데이터 타입	PL/SQL 최대 크기	SQL 최대 크기
CHAR	32,767 byte	2,000 byte
NCHAR	32,767 byte	2,000 byte
RAW	32,767 byte	2,000 byte
VARCHAR2	32,767 byte	4,000 byte
NVARCHAR2	32,767 byte	4,000 byte
LONG	32,767 byte	2 GB - 1
LONG RAW	32,767 byte	2 GB
BLOB	128 TB	4 GB -1
CLOB	128 TB	4 GB -1
NCLOB	128 TB	4 GB -1

오라클 함수 작성

with



01. 기본 작성 방법

CREATE [OR REPLACE] FUNCTION 함수 이름 (매개변수1, 매개변수2, …)

RETURN 데이터타입;

IS[AS]

변수, 상수 등 선언

BEGIN

실행부

RETURN 반환값;

[EXCEPTION 예외 처리부]

END [함수 이름];

01. 기본 작성 방법

함수 구문	설명
CREATE OR REPLACE FUNCTION	함수 생성 기본 구문
매개변수	매개변수명 데이터 타입 형태로 명시
RETURN 데이터 타입	함수가 반환할 데이터 타입 지정
RETURN 반환값	매개변수를 받아 특정 연산 수행 후 반환할 값 명시

- 사용 예시

```
SELECT 사용자 정의 함수(매개변수1, ...) FROM ...
```

오라클 프로시저 작성

with



PL/SQL
ORACLE®

01. 기본 작성 방법

```
CREATE [OR REPLACE] PROCEDURE 프로시저 이름  
  (매개변수명1[IN |OUT | IN OUT] 데이터타입[:= 디폴트 값],  
   매개변수명2[IN |OUT | IN OUT] 데이터타입[:= 디폴트 값],  
   …)
```

```
IS[AS]  
  변수, 상수 등 선언  
BEGIN  
  실행부
```

```
[EXCEPTION 예외 처리부]  
END [프로시저 이름];
```

- 사용 예시

```
EXEC 프로시저명(매개변수 1값, 매개변수 2값, …);
```

01. 기본 작성 방법

매개 변수 구분	내용
IN 매개변수	참조만 가능하며 값을 할당할 수 없음
OUT 매개변수	값을 전달할 수 있지만 의미는 없음
OUT, IN OUT 매개변수	디폴트 값을 설정할 수 없음
RETURN	수행을 종료한다는 의미 (LOOP의 EXIT와 유사)

02. 명명법

구분	내용
변수	Variable의 맨 앞 글자인 v + 데이터 타입의 앞 글자(문자는 s, 숫자는 n, 날짜는 d) + 변수 설명(테이블 컬럼 값을 사용할 때는 컬럼명) 예) vs_emp_name, vn_employee_id, vd_hire_date
상수	constant의 맨 앞 글자인 c + 나머지는 변수와 동일 예) cn_pi (파이 값), cn_inch_to_meter(인치를 미터로 변환하는 상수)
매개변수	parameter의 맨 앞 글자인 p + 나머지는 변수와 동일 예) pn_employee_id, pn_salary, pd_retire_date
함수	get (값을 가져오므로) + 함수 설명 약자 + fn 예) get_dep_name_fn (해당 부서에서 가장 높은 급여 값을 가져오는 함수)
프로시저	처리 유형 + 프로시저 설명 약자 + proc upd_emp_sal_proc(사원 급여를 갱신하는 프로시저)

PL/SQL 예외처리 개념

with



01. 기본 개념

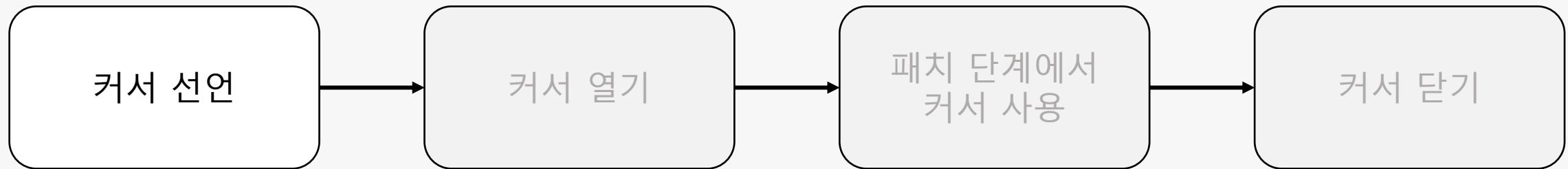
- 묵시적 커서(Implicit Cursor)
 - SQL 문장이 실행될 때마다 자동으로 만들어져 사용됨
- 명시적 커서(Explicit Cursor)
 - 사용자가 직접 정의해서 사용하는 커서를 말함



02. 목시적 커서 속성

속성명	설명
SQL%FOUND	결과 집합의 패치 로우 수가 1개 이상이면 TRUE, 아니면 FALSE를 반환
SQL%NOTFOUND	결과 집합의 패치 로우 수가 0이면 TRUE, 아니면 FALSE를 반환
SQL%ROWCOUNT	영향 받은 결과 집합의 로우 수 반환, 없으면 0을 반환
SQL%ISOPEN	목시적 커서는 항상 FALSE를 반환

03. 명시적 커서

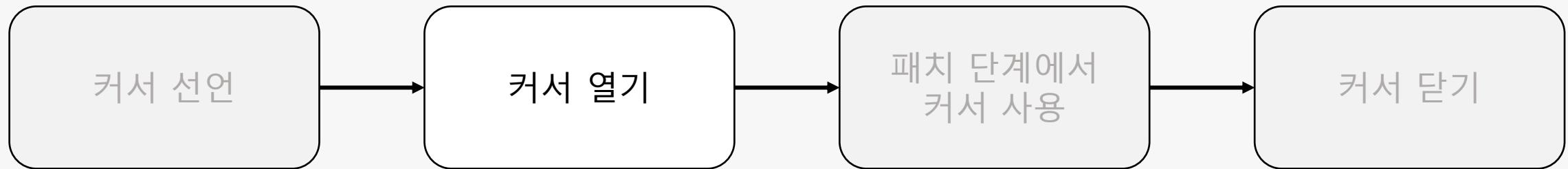


CURSOR 커서명[(매개변수1, 매개변수2, …)]

IS

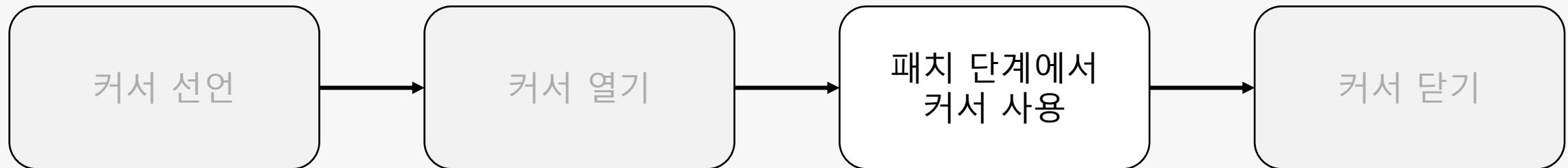
SELECT 문장;

03. 명시적 커서



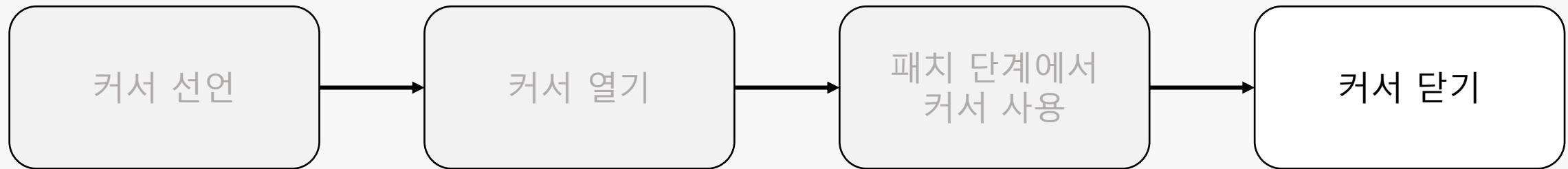
OPEN 커서명 [(매개변수1), (매개변수2), …];

03. 명시적 커서



```
LOOP
  FETCH 커서명 INTO 변수1, 변수2, …;
  EXIT WHEN 커서명%NOTFOUND;
END LOOP;
```

03. 명시적 커서



CLOSE 커서명;

04. 커서와 FOR문

```
FOR 레코드 IN 커서명(매개변수1, 매개변수2, …)
```

```
LOOP
```

```
처리문;
```

```
END LOOP;
```

05. 커서 변수

- 한 개 이상의 쿼리를 연결해 사용할 수 있다
- 변수처럼 커서 변수를 함수나 프로시저의 매개변수로 전달할 수 있다
- 커서 속성을 사용할 수 있다
- 커서 변수의 사용법 4가지
 - ✓ 커서 변수 선언
 - ✓ 커서 변수 사용
 - ✓ 커서 변수에서 결과 집합 가져오기
 - ✓ 커서 변수를 매개변수로 전달하기

PL/SQL 예외처리 개념

with



01. 기본 구문

- 오라클에서 발생시키는 시스템 예외
- 사용자가 의도적으로 발생시킬 수 있는 사용자 정의 예외

```
EXCEPTION WHEN 예외명1 THEN 예외처리 구문1
```

```
WHEN 예외명2 THEN 예외처리 구문2
```

```
...
```

```
WHEN OTHERS THEN 예외처리 구문n;
```

02. SQLCODE, SQLERRM 예외정보 참조

- 오라클에서 제공하는 Built-in 함수
- SQLCODE : 발생한 예외에 해당하는 코드 반환
- SQLERRM : 발생한 예외에 대한 오류 메시지 반환
- DBMS.Utility.Format_Error_Backtrace 사용 시, 에러코드 라인 숫자 반환

03. PL/SQL 미리 정의된 예외

- 공식 문서 :

https://docs.oracle.com/cd/B10501_01/appdev.920/a96624/07_errs.htm

예외명	예외 코드	설명
ACCESS_INTO_NULL	ORA-06530	LOB과 같은 객체 초기화 되지 않은 상태에서 사용
INVALID_NUMBER	ORA-01722	문자를 숫자로 변환할 때 실패할 경우
NO_DATA_FOUND	ORA-01403	SELECT INTO 시 데이터가 한 건도 없을 경우
STORAGE_ERROR	ORA-06500	프로그램 수행 시 메모리가 부족할 경우
...

04.1. 사용자 정의 예외

- 예외 정의 : 사용자 정의 예외명 EXCEPTION
 - ✓ PL/SQL 블록의 선언부에 예외 정의
- 예외 발생시키기 : RAISE 사용자 정의 예외명
 - ✓ **RAISE 예외명;** 형태로 사용
- 발생된 예외 처리 : EXCEPTION WHEN 사용자 정의 예외명 THEN
 - ✓ 시스템 예외와 동일한 방식으로 처리

04.2. 시스템 예외에 이름 부여

- 예외 정의 : 사용자 정의 예외명 EXCEPTION
 - ✓ 선언부에서 사용자 정의 예외 선언
- 사용자 정의 예외명과 시스템 예외 코드 연결
 - ✓ PRAGMA EXCEPTION_INIT(사용자 정의 예외명, 시스템 예외 코드)
- 발생된 예외 처리 : EXCEPTION WHEN 사용자 정의 예외명 THEN
 - ✓ 시스템 예외와 동일한 방식으로 처리

04.3. RAISE & RAISE_APPLICATION_ERROR

- 예외 코드와 예외 메시지는 사용자가 직접 정의할 수 있음
- 예외 코드는 -2000 ~ -20999번까지 사용 가능
 - ✓ 오라클에서 이미 사용하고 있는 시스템 예외는 위 번호 구간 미사용 중

```
RAISE_APPLICATION_ERROR (예외코드, 예외 메시지);
```

04.4. 효율적인 예외 처리 방법

- 시스템 예외인 경우는 OTHERS를 사용한다.
 - ✓ OTHERS는 모든 시스템 예외를 잡아냄
 - ✓ SQLCODE, SQLERRM 적극 활용
- 예외 처리 루틴을 공통 모듈화하고, 발생된 예외 로그 남긴다.
 - ✓ error_log 테이블과 같은 걸로 관리
- 사용자 정의 예외도 별도의 테이블로 미리 만들어 관리
 - ✓ 동일한 예외 코드를 사용하여 중복 발생 소지 우려
 - ✓ 실제 해당 예외 발생 시, 예외코드와 번호 등을 읽어오는 식으로 처리

05. 효율적인 예외 처리 방법

- 공식 문서 :

https://docs.oracle.com/cd/B10501_01/appdev.920/a96624/07_errs.htm

예외명	예외 코드	설명
ACCESS_INTO_NULL	ORA-06530	LOB과 같은 객체 초기화 되지 않은 상태에서 사용
INVALID_NUMBER	ORA-01722	문자를 숫자로 변환할 때 실패할 경우
NO_DATA_FOUND	ORA-01403	SELECT INTO 시 데이터가 한 건도 없을 경우
STORAGE_ERROR	ORA-06500	프로그램 수행 시 메모리가 부족할 경우
...

PL/SQL 트리거 개념

with



PL/SQL
ORACLE®

01. 기본 개념

- 트리거 (TRIGGER)의 사전적 의미 : 방아쇠, 촉발시키다, 야기하다, 유발하다 등
- DML 작업 (INSERT, UPDATE, DELETE) 작업이 일어날 때, 자동적으로 실행
- 주요 사용처
 - ✓ 자동으로 파생된 열 값 생성
 - ✓ 잘못된 트랜잭션 방지
 - ✓ 테이블 액세스 통계 수집
- 트리거 내에서는 COMMIT & ROLLBACK문 사용불가

02. 트리거의 유형

- DML 트리거
 - ✓ 단순 DML 트리거 (문장 단위 DML 트리거, 행 단위 DML 트리거)
 - ✓ INSTEAD OF DML 트리거
 - ✓ 복합 DML 트리거
- 시스템 트리거
 - ✓ 데이터베이스 트리거
 - ✓ 스키마 트리거
 - ✓ INSTEAD OF CREATE 트리거

03. 단순 DML 트리거

조건 술어	설명
INSERTING	INSERT 문이면 TRUE를 반환
UPDATING	UPDATE 문이면 TRUE를 반환
UPDATING ('column')	지정한 컬럼에 대한 UPDATE 문이면 TRUE를 반환
DELETING	DELETE 문이면 TRUE를 반환

- 네 가지 타이밍 포인트를 지정 (BEFORE, AFTER 키워드)

수행 시점	수행 단위	설명
BEFORE		DML 문 수행 전에 수행
AFTER		DML 문 수행 후에 수행
BEFORE	FOR EACH ROW	DML 문 수행 전에 행 단위로 수행
AFTER	FOR EACH ROW	DML 문 수행 후에 행 단위로 수행

03. 단순 DML 트리거

- BEFORE STATEMENT
 - ✓ SQL 구문이 실행되기 전에 그 문장에 대해 한 번 실행
- BEFORE ROW
 - ✓ SQL 구문이 실행되기 전(DML 작업을 수행 전에), 각 행(ROW)에 대해 한 번씩 실행
- AFTER STATEMENT
 - ✓ SQL 구문이 실행된 후에 그 문장에 대해 한 번 실행
- AFTER ROW
 - ✓ SQL 구문이 실행된 후에 (DML 작업 수행 후), 각 행(ROW)에 대해 한 번씩 실행

04. 행 단위 DML 트리거

- 행 단위(row-level) 트리거는 행 단위로 수행되는 트리거.
 - ✓ Before 트리거는 DML 문이 수행되기 전 수행
 - ✓ After 트리거는 DML 문이 수행된 후 행 수행
- 행 단위 DML 트리거는 OLD, NEW 상관명으로 변경 전후의 값을 참조할 수 있음

	OLD	NEW
INSERT		삽입 값
UPDATE	갱신 전 값	갱신 후 값
DELETE	삭제 값	

05. 기본 구문

```
CREATE [ OR REPLACE ] TRIGGER 트리거명
    [ BEFORE | AFTER ]
    이벤트1 [ OR 이벤트2 [ OR 이벤트3 ] ] ON 테이블명 -- 장착되는 테이블
    [ FOR EACH ROW [WHEN TRIGGER 조건] ]
    [ DECLARE ]
        -- 선언 구문
        ...
BEGIN
    -- 실행 구문
    ...
END;
```