

## **Oracle Database 11g: SQL Fundamentals I(한글판)**

**블록 II • 학생용**

D49996KR20

Edition 2.0

2009년 12월

D64453

**ORACLE®**

## 저자

Salome Clement

Brian Pottle

Puja Singh

## 기술 제공자 및 검토자

Anjulapponni Azhagulekshmi

Clair Bennett

Zarko Cesljas

Yanti Chang

Gerlinde Frenzen

Steve Friedberg

Joel Goodman

Nancy Greenberg

Pedro Neves

Surya Rekha

Helen Robertson

Lauran Serhal

Tulika Srivastava

**Copyright © 2009, Oracle. All rights reserved.**

## Disclaimer

본 문서는 독점적 정보를 포함하고 있으며 저작권법 및 기타 지적 재산법에 의해 보호됩니다. 본 문서는 오라클 교육 과정에서 자신이 사용할 목적으로만 복사하고 인쇄할 수 있습니다. 어떤 방법으로도 본 문서를 수정하거나 변경할 수 없습니다. 저작권법에 따라 "공정"하게 사용하는 경우를 제외하고, 오라클의 명시적 허가 없이 본 문서의 전체 또는 일부를 사용, 공유, 다운로드, 업로드, 복사, 인쇄, 표시, 실행, 재생산, 게시, 라이센스, 우편 발송, 전송 또는 배포할 수 없습니다.

본 문서의 내용은 사전 공지 없이 변경될 수 있습니다. 만일 본 문서의 내용상 문제점을 발견하면 서면으로 통지해 주기 바랍니다. Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. 오라클은 본 문서에 오류가 존재하지 않음을 보증하지 않습니다.

## Restricted Rights Notice

만일 본 문서를 미국 정부나 또는 미국 정부를 대신하여 문서를 사용하는 개인이나 법인에게 배송하는 경우, 다음 공지 사항이 적용됩니다.

### U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

## Trademark Notice

Oracle은 Oracle Corporation 또는 그 자회사의 등록 상표입니다. 기타의 명칭들은 각 해당 명칭을 소유한 회사의 상표일 수 있습니다.

## 편집자

Aju Kumar

Arijit Ghosh

## 그래픽 디자이너

Rajiv Chandrabhanu

## 발행인

Pavithran Adka

Veena Narasimhan

# 목차

## I 소개

- 단원 목표 I-2
- 단원 내용 I-3
- 과정 목표 I-4
- 과정 일정 I-5
- 본 과정에 사용되는 부록 I-7
- 단원 내용 I-8
- Oracle Database 11g: 핵심 영역 I-9
- Oracle Database 11g I-10
- Oracle Fusion Middleware I-12
- Oracle Enterprise Manager Grid Control I-13
- Oracle BI Publisher I-14
- 단원 내용 I-15
- 관계형 및 객체 관계형 데이터베이스 관리 시스템 I-16
- 다양한 미디어의 데이터 저장 영역 I-17
- 관계형 데이터베이스 개념 I-18
- 관계형 데이터베이스 정의 I-19
- 데이터 모델 I-20
- 엔티티 관계 모델 I-21
- 엔티티 관계 모델링 규칙 I-23
- 여러 테이블 연관짓기 I-25
- 관계형 데이터베이스 용어 I-27
- 단원 내용 I-29
- SQL을 사용하여 데이터베이스 조회 I-30
- SQL 문 I-31
- SQL 개발 환경 I-32
- 단원 내용 I-33
- Human Resources(HR) 스키마 I-34
- 연습 과정에 사용되는 테이블 I-35
- 단원 내용 I-36
- Oracle Database 11g 설명서 I-37
- 추가 자료 I-38
- 요약 I-39
- 연습 I: 개요 I-40

**1 SQL SELECT 문을 사용하여 데이터 검색**

- 목표 1-2
- 단원 내용 1-3
- SQL SELECT 문의 기능 1-4
- 기본 SELECT 문 1-5
- 모든 열 선택 1-6
- 특정 열 선택 1-7
- SQL 문 작성 1-8
- 열 머리글 기본값 1-9
- 단원 내용 1-10
- 산술식 1-11
- 산술 연산자 사용 1-12
- 연산자 우선 순위 1-13
- Null 값 정의 1-14
- 산술식의 Null 값 1-15
- 단원 내용 1-16
- 열 alias 정의 1-17
- 열 alias 사용 1-18
- 단원 내용 1-19
- 연결 연산자 1-20
- 리터럴 문자열 1-21
- 리터럴 문자열 사용 1-22
- 대체 인용(q) 연산자 1-23
- 중복 행 1-24
- 단원 내용 1-25
- 테이블 구조 표시 1-26
- DESCRIBE 명령 사용 1-27
- 퀴즈 1-28
- 요약 1-29
- 연습 1: 개요 1-30

**2 데이터 제한 및 정렬**

- 목표 2-2
- 단원 내용 2-3
- 선택을 사용하여 행 제한 2-4
- 선택되는 행 제한 2-5
- WHERE 절 사용 2-6
- 문자열 및 날짜 2-7
- 비교 연산자 2-8
- 비교 연산자 사용 2-9
- BETWEEN 연산자를 사용하는 범위 조건 2-10
- IN 연산자를 사용하는 멤버 조건 2-11

LIKE 연산자를 사용하여 패턴 일치	2-12
대체 문자 결합	2-13
NULL 조건 사용	2-14
논리 연산자를 사용하여 조건 정의	2-15
AND 연산자 사용	2-16
OR 연산자 사용	2-17
NOT 연산자 사용	2-18
단원 내용	2-19
우선 순위 규칙	2-20
단원 내용	2-22
ORDER BY 절 사용	2-23
정렬	2-24
단원 내용	2-26
치환 변수	2-27
단일 앤퍼샌드 치환 변수 사용	2-29
문자 및 날짜 값을 치환 변수로 지정	2-31
열 이름, 표현식 및 텍스트 지정	2-32
이중 앤퍼샌드 치환 변수 사용	2-33
단원 내용	2-34
DEFINE 명령 사용	2-35
VERIFY 명령 사용	2-36
퀴즈	2-37
요약	2-38
연습 2: 개요	2-39

### 3 단일 행 함수를 사용하여 출력 커스터마이즈

목표	3-2
단원 내용	3-3
SQL 함수	3-4
SQL 함수의 두 가지 유형	3-5
단일 행 함수	3-6
단원 내용	3-8
문자 함수	3-9
대소문자 변환 함수	3-11
대소문자 변환 함수 사용	3-12
문자 조작 함수	3-13
문자 조작 함수 사용	3-14
단원 내용	3-15
숫자 함수	3-16
ROUND 함수 사용	3-17
TRUNC 함수 사용	3-18
MOD 함수 사용	3-19

단원 내용 3-20  
 날짜 작업 3-21  
 RR 날짜 형식 3-22  
 SYSDATE 함수 사용 3-24  
 날짜 연산 3-25  
 날짜에 산술 연산자 사용 3-26  
 단원 내용 3-27  
 날짜 조작 함수 3-28  
 날짜 함수 사용 3-30  
 날짜에 ROUND 및 TRUNC 함수 사용 3-31  
 퀴즈 3-32  
 요약 3-33  
 연습 3: 개요 3-34

#### **4 변환 함수 및 조건부 표현식 사용**

목표 4-2  
 단원 내용 4-3  
 변환 함수 4-4  
 암시적 데이터 유형 변환 4-5  
 명시적 데이터 유형 변환 4-7  
 단원 내용 4-10  
 날짜에 TO\_CHAR 함수 사용 4-11  
 날짜 형식 모델의 요소 4-12  
 날짜에 TO\_CHAR 함수 사용 4-16  
 숫자에 TO\_CHAR 함수 사용 4-17  
 TO\_NUMBER 및 TO\_DATE 함수 사용 4-20  
 RR 날짜 형식으로 TO\_CHAR 및 TO\_DATE 함수 사용 4-22  
 단원 내용 4-23  
 함수 중첩 4-24  
 함수 중첩: 예제 1 4-25  
 함수 중첩: 예제 2 4-26  
 단원 내용 4-27  
 일반 함수 4-28  
 NVL 함수 4-29  
 NVL 함수 사용 4-30  
 NVL2 함수 사용 4-31  
 NULLIF 함수 사용 4-32  
 COALESCE 함수 사용 4-33  
 단원 내용 4-36  
 조건부 표현식 4-37  
 CASE 식 4-38  
 CASE 식 사용 4-39

DECODE 함수 4-40

DECODE 함수 사용 4-41

퀴즈 4-43

요약 4-44

연습 4: 개요 4-45

## 5 그룹 함수를 사용하여 집계된 데이터 보고

목표 5-2

단원 내용 5-3

그룹 함수란? 5-4

그룹 함수 유형 5-5

그룹 함수: 구문 5-6

AVG 및 SUM 함수 사용 5-7

MIN 및 MAX 함수 사용 5-8

COUNT 함수 사용 5-9

DISTINCT 키워드 사용 5-10

그룹 함수 및 null 값 5-11

단원 내용 5-12

데이터 그룹 생성 5-13

데이터 그룹 생성: GROUP BY 절 구문 5-14

GROUP BY 절 사용 5-15

두 개 이상의 열로 그룹화 5-17

여러 열에서 GROUP BY 절 사용 5-18

그룹 함수를 사용한 잘못된 Query 5-19

그룹 결과 제한 5-21

HAVING 절을 사용하여 그룹 결과 제한 5-22

HAVING 절 사용 5-23

단원 내용 5-25

그룹 함수 중첩 5-26

퀴즈 5-27

요약 5-28

연습 5: 개요 5-29

## 6 조인을 사용하여 여러 테이블의 데이터 표시

목표 6-2

단원 내용 6-3

여러 테이블에서 데이터 가져오기 6-4

조인 유형 6-5

SQL:1999 구문을 사용한 테이블 조인 6-6

모호한 열 이름 한정 6-7

단원 내용 6-9

Natural Join 생성 6-10

Natural join으로 레코드 검색	6-11
USING 절로 조인 생성	6-12
열 이름 조인	6-13
USING 절을 사용하여 레코드 검색	6-14
USING 절에 테이블 Alias 사용	6-15
ON 절로 조인 생성	6-17
ON 절을 사용하여 레코드 검색	6-18
ON 절을 사용하여 3-Way 조인 생성	6-19
조인에 추가 조건 적용	6-20
단원 내용	6-21
테이블 자체 조인	6-22
ON 절을 사용하는 Self Join	6-23
단원 내용	6-24
Nonequijoin	6-25
Nonequijoin을 사용하여 레코드 검색	6-26
단원 내용	6-27
OUTER Join과 직접 일치하지 않는 레코드 반환	6-28
INNER Join과 OUTER Join	6-29
LEFT OUTER JOIN	6-30
RIGHT OUTER JOIN	6-31
FULL OUTER JOIN	6-32
단원 내용	6-33
Cartesian Product	6-34
Cartesian Product 생성	6-35
Cross Join 생성	6-36
퀴즈	6-37
요약	6-38
연습 6: 개요	6-39

## 7 Subquery를 사용하여 Query 해결

목표	7-2
단원 내용	7-3
Subquery를 사용하여 문제 해결	7-4
Subquery 구문	7-5
Subquery 사용	7-6
Subquery 사용 지침	7-7
Subquery 유형	7-8
단원 내용	7-9
단일 행 Subquery	7-10
단일 행 Subquery 실행	7-11
Subquery에서 그룹 함수 사용	7-12
Subquery가 있는 HAVING 절	7-13

이 명령문에서 잘못된 점은?	7-14
Inner query에서 반환된 행이 없음	7-15
단원 내용	7-16
여러 행 Subquery	7-17
여러 행 Subquery에서 ANY 연산자 사용	7-18
여러 행 Subquery에서 ALL 연산자 사용	7-19
EXISTS 연산자 사용	7-20
단원 내용	7-21
Subquery의 null 값	7-22
퀴즈	7-24
요약	7-25
연습 7: 개요	7-26

## 8 집합 연산자 사용

목표	8-2
단원 내용	8-3
집합 연산자	8-4
집합 연산자 지침	8-5
Oracle 서버 및 집합 연산자	8-6
단원 내용	8-7
이 단원에 사용되는 테이블	8-8
단원 내용	8-12
UNION 연산자	8-13
UNION 연산자 사용	8-14
UNION ALL 연산자	8-16
UNION ALL 연산자 사용	8-17
단원 내용	8-18
INTERSECT 연산자	8-19
INTERSECT 연산자 사용	8-20
단원 내용	8-21
MINUS 연산자	8-22
MINUS 연산자 사용	8-23
단원 내용	8-24
SELECT 문 일치	8-25
SELECT 문 일치: 예제	8-26
단원 내용	8-27
집합 연산에서 ORDER BY 절 사용	8-28
퀴즈	8-29
요약	8-30
연습 8: 개요	8-31

## 9 데이터 조작

- 목표 9-2
- 단원 내용 9-3
- DML(데이터 조작어) 9-4
- 테이블에 새 행 추가 9-5
- INSERT 문 구문 9-6
- 새 행 삽입 9-7
- Null 값을 가진 행 삽입 9-8
- 특수 값 삽입 9-9
- 특정 날짜 및 시간 값 삽입 9-10
- 스크립트 작성 9-11
- 다른 테이블에서 행 복사 9-12
- 단원 내용 9-13
- 테이블의 데이터 변경 9-14
- UPDATE 문 구문 9-15
- 테이블의 행 갱신 9-16
- Subquery를 사용하여 두 개의 열 갱신 9-17
- 다른 테이블을 기반으로 행 갱신 9-19
- 단원 내용 9-20
- 테이블에서 행 제거 9-21
- DELETE 문 9-22
- 테이블에서 행 삭제 9-23
- 다른 테이블을 기반으로 행 삭제 9-24
- TRUNCATE 문 9-25
- 단원 내용 9-26
- 데이터베이스 트랜잭션 9-27
- 데이터베이스 트랜잭션: 시작과 종료 9-28
- COMMIT 및 ROLLBACK 문의 이점 9-29
- 명시적 트랜잭션 제어문 9-30
- 변경 사항을 표시자로 롤백 9-31
- 암시적 트랜잭션 처리 9-32
- COMMIT 또는 ROLLBACK 전의 데이터 상태 9-34
- COMMIT 후의 데이터 상태 9-35
- 데이터 커밋 9-36
- ROLLBACK 후의 데이터 상태 9-37
- ROLLBACK 후의 데이터 상태: 예제 9-38
- 명령문 레벨 롤백 9-39
- 단원 내용 9-40
- 읽기 일관성 9-41
- 읽기 일관성 구현 9-42
- 단원 내용 9-43
- SELECT 문의 FOR UPDATE 절 9-44

FOR UPDATE 절: 예제 9-45  
 퀴즈 9-47  
 요약 9-48  
 연습 9: 개요 9-49

## 10 DDL 문을 사용하여 테이블 생성 및 관리

목표 10-2  
 단원 내용 10-3  
 데이터베이스 객체 10-4  
 이름 지정 규칙 10-5  
 단원 내용 10-6  
 CREATE TABLE 문 10-7  
 다른 유저의 테이블 참조 10-8  
 DEFAULT 옵션 10-9  
 테이블 생성 10-10  
 단원 내용 10-11  
 데이터 유형 10-12  
 Datetime 데이터 유형 10-14  
 단원 내용 10-15  
 제약 조건 포함 10-16  
 제약 조건 지침 10-17  
 제약 조건 정의 10-18  
 NOT NULL 제약 조건 10-20  
 UNIQUE 제약 조건 10-21  
 PRIMARY KEY 제약 조건 10-23  
 FOREIGN KEY 제약 조건 10-24  
 FOREIGN KEY 제약 조건: 키워드 10-26  
 CHECK 제약 조건 10-27  
 CREATE TABLE: 예제 10-28  
 제약 조건 위반 10-29  
 단원 내용 10-31  
 Subquery를 사용하여 테이블 생성 10-32  
 단원 내용 10-34  
 ALTER TABLE 문 10-35  
 읽기 전용 테이블 10-36  
 단원 내용 10-37  
 테이블 삭제 10-38  
 퀴즈 10-39  
 요약 10-40  
 연습 10: 개요 10-41

## 11 기타 스키마 객체 생성

- 목표 11-2
- 단원 내용 11-3
- 데이터베이스 객체 11-4
- 뷰란? 11-5
- 뷰의 이점 11-6
- 단순 뷰와 복합 뷰 11-7
- 뷰 생성 11-8
- 뷰에서 데이터 검색 11-11
- 뷰 수정 11-12
- 복합 뷰 생성 11-13
- 뷰에 대한 DML 작업 수행 규칙 11-14
- WITH CHECK OPTION 절 사용 11-17
- DML 작업 거부 11-18
- 뷰 제거 11-20
- 연습 11: 1부 개요 11-21
- 단원 내용 11-22
- 시퀀스 11-23
- CREATE SEQUENCE 문: 구문 11-25
- 시퀀스 생성 11-26
- NEXTVAL 및 CURRVAL Pseudocolumn 11-27
- 시퀀스 사용 11-29
- 시퀀스 값 캐시 11-30
- 시퀀스 수정 11-31
- 시퀀스 수정 지침 11-32
- 단원 내용 11-33
- 인덱스 11-34
- 인덱스가 생성되는 방식 11-36
- 인덱스 생성 11-37
- 인덱스 생성 지침 11-38
- 인덱스 제거 11-39
- 단원 내용 11-40
- 동의어 11-41
- 객체의 동의어 생성 11-42
- 동의어 생성 및 제거 11-43
- 퀴즈 11-44
- 요약 11-45
- 연습 11: 2부 개요 11-46

**부록 A: 연습 해답****부록 B: 테이블 설명****부록 C: SQL Developer 사용**

- 목표 C-2
- Oracle SQL Developer란? C-3
- SQL Developer 사양 C-4
- SQL Developer 1.5 인터페이스 C-5
- 데이터베이스 연결 생성 C-7
- 데이터베이스 객체 탐색 C-10
- 테이블 구조 표시 C-11
- 파일 탐색 C-12
- 스크리마 객체 생성 C-13
- 새 테이블 생성: 예제 C-14
- SQL Worksheet 사용 C-15
- SQL 문 실행 C-18
- SQL 스크립트 저장 C-19
- 저장된 SQL 스크립트 실행: 방법 1 C-20
- 저장된 SQL 스크립트 실행: 방법 2 C-21
- SQL 코드 형식 지정 C-22
- Snippet 사용 C-23
- Snippet 사용: 예제 C-24
- 프로시저 및 함수 디버깅 C-25
- 데이터베이스 보고 C-26
- 유저 정의 보고서 작성 C-27
- 검색 엔진 및 External 도구 C-28
- 환경 설정 C-29
- SQL Developer 레이아웃 재설정 C-30
- 요약 C-31

**부록 D: SQL\*Plus 사용**

- 목표 D-2
- SQL과 SQL\*Plus의 상호 작용 D-3
- SQL 문과 SQL\*Plus 명령 비교 D-4
- SQL\*Plus 개요 D-5
- SQL\*Plus에 로그인 D-6
- 테이블 구조 표시 D-7
- SQL\*Plus 편집 명령 D-9
- LIST, n 및 APPEND 사용 D-11
- CHANGE 명령 사용 D-12
- SQL\*Plus 파일 명령 D-13

SAVE 및 START 명령 사용	D-15
SERVEROUTPUT 명령	D-16
SQL*Plus SPOOL 명령 사용	D-17
AUTOTRACE 명령 사용	D-18
요약	D-19

## 부록 E: JDeveloper 사용

목표	E-2
Oracle JDeveloper	E-3
Database Navigator	E-4
연결 생성	E-5
데이터베이스 객체 탐색	E-6
SQL 문 실행	E-7
프로그램 단위 생성	E-8
컴파일	E-9
프로그램 단위 실행	E-10
프로그램 단위 삭제	E-11
Structure window	E-12
Editor window	E-13
Application Navigator	E-14
Java 내장 프로시저 배치	E-15
PL/SQL에 Java 게시(Publishing)	E-16
JDeveloper 11g에 대해 자세히 배울 수 있는 방법	E-17
요약	E-18

## 부록 F: Oracle 조인 구문

목표	F-2
여러 테이블에서 데이터 가져오기	F-3
Cartesian Product	F-4
Cartesian Product 생성	F-5
Oracle 고유의 조인 유형	F-6
Oracle 구문을 사용하여 테이블 조인	F-7
모호한 열 이름 한정	F-8
Equijoin	F-10
Equijoin을 사용하여 레코드 검색	F-11
Equijoin을 사용하여 레코드 검색: 예제	F-12
AND 연산자를 사용한 추가 검색 조건	F-13
세 개 이상의 테이블 조인	F-14

Nonequijoin F-15

Nonequijoin을 사용하여 레코드 검색 F-16

Outer Join을 사용하여 직접 일치되지 않는 레코드 반환 F-17

Outer Join: 구문 F-18

Outer Join 사용 F-19

Outer Join: 다른 예제 F-20

테이블 자체 조인 F-21

Self Join: 예제 F-22

요약 F-23

연습 F: 개요 F-24

## 추가 연습 및 해답

### 색인



# 9

## 데이터 조작

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 각 DML(데이터 조작어) 문 설명
- 테이블에 행 삽입
- 테이블의 행 갱신
- 테이블에서 행 삭제
- 트랜잭션 제어

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 목표

이 단원에서는 DML(데이터 조작어) 문을 사용하여 테이블에 행을 삽입하고, 테이블의 기존 행을 갱신하고, 테이블에서 기존 행을 삭제하는 방법을 배웁니다. 또한 COMMIT, SAVEPOINT 및 ROLLBACK 문을 사용하여 트랜잭션을 제어하는 방법을 배웁니다.

## 단원 내용

- 테이블에 새 행 추가
  - INSERT 문
- 테이블의 데이터 변경
  - UPDATE 문
- 테이블에서 행 제거
  - DELETE 문
  - TRUNCATE 문
- COMMIT, ROLLBACK 및 SAVEPOINT를 사용하여 데이터베이스 트랜잭션 제어
- 읽기 일관성
- SELECT 문의 FOR UPDATE 절

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# DML(데이터 조작어)

- DML 문은 다음과 같은 경우에 실행합니다.
  - 테이블에 새 행 추가
  - 테이블의 기존 행 수정
  - 테이블에서 기존 행 제거
- 트랜잭션은 논리적 작업 단위를 형성하는 DML 문의 모음으로 구성됩니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## DML(데이터 조작어)

DML(데이터 조작어)은 SQL의 핵심 부분입니다. 데이터베이스에서 데이터를 추가, 생성 또는 삭제하려는 경우 DML 문을 실행하십시오. 논리적 작업 단위를 형성하는 DML 문의 모음을 트랜잭션이라고 합니다.

은행 데이터베이스를 생각해 봅시다. 은행 고객이 예금 계좌에서 결제 계좌로 현금을 이체하는 경우 트랜잭션은 예금 계좌 감소, 결제 계좌 증가, 트랜잭션 저널에 트랜잭션 기록이라는 세 가지 별도 작업으로 구성됩니다. Oracle 서버에서 세 개의 SQL 문을 수행할 때 계좌 잔액이 정확히 유지되어야 합니다. 특정 문제로 인해 트랜잭션의 명령문 중 하나가 실행되지 못하면 트랜잭션의 다른 명령문도 연두되어야 합니다.

### 참고

- 이 단원에 나오는 대부분의 DML 문에서는 테이블에 대한 제약 조건이 위반되지 않은 것으로 가정합니다. 제약 조건은 본 과정의 뒷부분에서 다릅니다.
- SQL Developer에서 DML 문을 실행하려면 Run Script 아이콘 또는 [F5]를 누릅니다. 피드백 메시지가 Script Output 탭 페이지에 표시됩니다.

# 테이블에 새 행 추가

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10 Administration	200	1700	
2	20 Marketing	201	1800	
3	50 Shipping	124	1500	
4	60 IT	103	1400	
5	80 Sales	149	2500	
6	90 Executive	100	1700	
7	110 Accounting	205	1700	
8	190 Contracting	(null)	1700	

70 Public Relations

100

1700

새  
행

DEPARTMENTS 테이블에  
새 행을 삽입합니다.

Copyright © 2009, Oracle. All rights reserved.

ORACLE®

## 테이블에 새 행 추가

슬라이드의 그림은 DEPARTMENTS 테이블에 새 부서를 추가하는 과정을 보여줍니다.

# INSERT 문 구문

- **INSERT 문을 사용하여 테이블에 새 행을 추가합니다.**

```
INSERT INTO table [(column [, column...])]  
VALUES  
      (value [, value...]);
```

- **이 구문을 사용할 경우 한 번에 한 행만 삽입됩니다.**

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## INSERT 문 구문

INSERT 문을 사용하여 테이블에 새 행을 추가할 수 있습니다.

이 구문에서 다음이 적용됩니다.

*table* 테이블의 이름입니다.

*column* 테이블에 채울 열의 이름입니다.

*value* 열의 해당 값입니다.

**참고:** VALUES 절이 있는 이 명령문은 한 번에 한 행만 테이블에 추가합니다.

## 새 행 삽입

- 각 열에 대한 값을 포함하는 새 행을 삽입합니다.
- 테이블에 있는 열의 기본 순서로 값을 나열합니다.
- 선택적으로 INSERT 절에 열을 나열합니다.

```
INSERT INTO departments(department_id,
                       department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
1 rows inserted
```

- 문자와 날짜 같은 작은 따옴표로 묶습니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 새 행 삽입

각 열에 대한 값을 포함하는 새 행을 삽입할 수 있기 때문에 INSERT 절에서 열 리스트가 필요하지 않습니다. 그러나 열 리스트를 사용하지 않을 경우 테이블에 있는 열의 기본 순서에 따라 값이 나열되어야 하고 각 열에 대해 값이 제공되어야 합니다.

DESCRIBE departments

Name	Null	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

명확성을 위해 INSERT 절에 열 리스트를 사용하십시오.

문자와 날짜 같은 작은 따옴표로 묶습니다. 그러나 숫자 같은 작은 따옴표로 묶지 않는 것이 좋습니다.

## null 값을 가진 행 삽입

- 암시적 방법: 열 리스트에서 열을 생략합니다.

```
INSERT INTO departments (department_id,
                        department_name)
VALUES      (30, 'Purchasing');
1 rows inserted
```

- 명시적 방법: VALUES 절에서 NULL 키워드를 지정합니다.

```
INSERT INTO departments
VALUES      (100, 'Finance', NULL, NULL);
1 rows inserted
```

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### null 값을 가진 행 삽입

메소드	설명
암시적	열 리스트에서 열을 생략합니다.
명시적	VALUES 리스트에 NULL 키워드를 지정합니다. 문자열과 날짜에 대해 VALUES 리스트에 빈 문자열(' ')을 지정합니다.

대상 열에서 null 값을 사용할 수 있는지 확인하려면 DESCRIBE 명령을 사용하여 Null 상태를 확인해야 합니다.

Oracle 서버는 모든 데이터 유형, 데이터 범위 및 데이터 무결성 제약 조건을 자동으로 강제 적용합니다. 명시적으로 나열되지 않은 열은 새 행에서 null 값을 가집니다.

다음 순서에 따라 유저가 입력하는 동안 발생할 수 있는 일반적인 오류를 검사합니다.

- NOT NULL 열에 누락된 필수 값
- unique key 또는 primary key 제약 조건을 위반하는 중복 값
- CHECK 제약 조건을 위반하는 값
- Foreign Key에 대해 유지되는 참조 무결성 제약 조건
- 데이터 유형 불일치 또는 열 크기를 초과하는 값

참고: 열 리스트를 통해 보다 읽기 쉽고 신뢰할 수 있으며 실수가 발생할 확률을 최소화한 INSERT 문을 작성할 수 있으므로 열 리스트를 사용하는 것이 좋습니다.

## 특수 값 삽입

SYSDATE 함수는 현재 날짜와 시간을 기록합니다.

```
INSERT INTO employees (employee_id,
                      first_name, last_name,
                      email, phone_number,
                      hire_date, job_id, salary,
                      commission_pct, manager_id,
                      department_id)
VALUES
      (113,
       'Louis', 'Popp',
       'LPOPP', '515.124.4567',
       SYSDATE, 'AC_ACCOUNT', 6900,
       NULL, 205, 110);
```

1 rows inserted

ORACLE

Copyright © 2009, Oracle. All rights reserved.

### 특수 값 삽입

함수를 사용하여 테이블에 특수 값을 입력할 수 있습니다.

슬라이드의 예제는 EMPLOYEES 테이블에 사원 Popp의 정보를 기록합니다. 이 예제는 HIRE\_DATE 열에 현재 날짜와 시간을 제공합니다. 이 예제에서는 데이터베이스 서버의 현재 날짜와 시간을 반환하는 SYSDATE 함수를 사용합니다. CURRENT\_DATE 함수를 사용하여 해당 세션 시간대의 현재 날짜를 구할 수도 있습니다. 또한 테이블에 행을 삽입할 때 USER 함수를 사용할 수 있습니다. USER 함수는 현재 username을 기록합니다.

#### 테이블에 추가된 내용 확인

```
SELECT employee_id, last_name, job_id, hire_date, commission_pct
  FROM   employees
 WHERE  employee_id = 113;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	COMMISSION_PCT
1	113	Popp	AC_ACCOUNT	10-JUL-09	(null)

## 특정 날짜 및 시간 값 삽입

- 새 사원을 추가합니다.

```
INSERT INTO employees
VALUES
(114,
'Den', 'Raphealy',
'DRAPHEAL', '515.127.4561',
TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
'SA REP', 11000, 0.2, 100, 60);
1 rows inserted
```

- 추가한 내용을 확인합니다.

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
1	114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	SA REP	11000	0.2

ORACLE

Copyright © 2009, Oracle. All rights reserved.

### 특정 날짜 및 시간 값 삽입

일반적으로 DD-MON-RR 형식을 사용하여 날짜 값을 삽입합니다. RR 형식을 사용하는 경우 시스템은 정확한 세기를 자동으로 제공합니다.

날짜 값을 DD-MON-YYYY 형식으로 제공할 수도 있습니다. 이 형식은 정확한 세기를 지정하는 내부 RR 형식 논리에 의존하지 않으며 세기를 명확히 지정하므로 이 형식을 사용하는 것이 좋습니다.

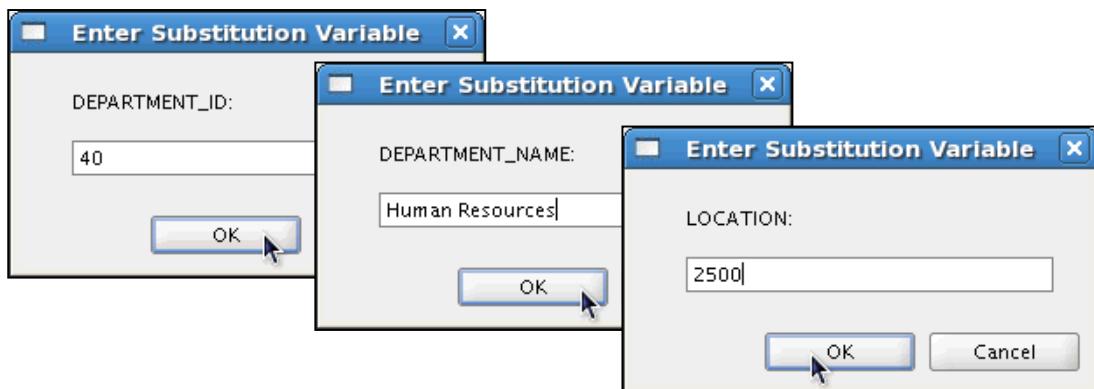
기본 형식이 아닌 다른 형식으로 날짜를 입력해야 하는 경우(예를 들어, 다른 세기나 특정 시간) TO\_DATE 함수를 사용해야 합니다.

슬라이드의 예제는 EMPLOYEES 테이블에 Raphealy 사원의 정보를 기록합니다. 이 예제에서는 HIRE\_DATE 열을 February 3, 1999로 설정합니다.

## 스크립트 작성

- SQL 문에서 & 치환을 사용하여 값을 입력하도록 요구합니다.
- &는 변수 값에 대한 위치 표시자입니다.

```
INSERT INTO departments
    (department_id, department_name, location_id)
VALUES  (&department_id, '&department_name', &location);
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

### 스크립트 작성

치환 변수를 사용하여 파일에 명령을 저장하고 파일의 명령을 실행할 수 있습니다. 슬라이드의 예제는 DEPARTMENTS 테이블에 부서에 대한 정보를 기록합니다.

스크립트 파일을 실행하면 각 앤피샌드(&) 치환 변수에 대해 값을 입력하도록 요구합니다. 치환 변수에 대한 값을 입력한 후에 OK 버튼을 누릅니다. 그러면 명령문 내에서 치환 변수가 입력한 값으로 치환됩니다. 이렇게 하면 동일한 스크립트 파일을 반복적으로 실행할 수 있고 파일을 실행할 때마다 다른 값 집합을 제공할 수 있습니다.

## 다른 테이블에서 행 복사

- INSERT 문을 subquery로 작성합니다.

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
SELECT employee_id, last_name, salary, commission_pct
FROM   employees
WHERE  job_id LIKE '%REP%';

4 rows inserted
```

- VALUES 절을 사용하지 마십시오.
- INSERT 절의 열 개수를 subquery의 열 개수와 일치시킵니다.
- subquery에서 반환되는 모든 행을 sales\_reps 테이블에 삽입합니다.

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### 다른 테이블에서 행 복사

INSERT 문을 사용하여 기존 테이블에서 파생된 값으로 테이블에 행을 추가할 수 있습니다. 슬라이드 예제에서 INSERT INTO 문이 작동하도록 하려면 먼저 CREATE TABLE 문을 사용하여 sales\_reps 테이블을 만들어 두어야 합니다. CREATE TABLE에 대해서는 "DDL 문을 사용하여 테이블 생성 및 관리" 단원에서 설명합니다.  
VALUES 절 자리에 subquery를 사용합니다.

#### 구문

```
INSERT INTO table [ column (, column) ] subquery;
```

이 구문에서 다음이 적용됩니다.

*table* 테이블의 이름입니다.

*column* 테이블에 채울 열의 이름입니다.

*subquery* 테이블에 행을 반환하는 subquery입니다.

INSERT 절의 열 리스트에 나오는 열 개수 및 해당 데이터 유형은 subquery의 값 개수 및 해당 데이터 유형과 일치해야 합니다. subquery에 의해 반환되는 행의 개수에 따라 하나 이상의 행이 추가되거나 어떠한 행도 추가되지 않습니다. 테이블의 행 복사본을 생성하려면 subquery에서 SELECT \*를 사용합니다.

```
INSERT INTO copy_emp
SELECT *
FROM employees;
```

## 단원 내용

- 테이블에 새 행 추가
  - INSERT 문
- 테이블의 데이터 변경
  - UPDATE 문
- 테이블에서 행 제거
  - DELETE 문
  - TRUNCATE 문
- COMMIT, ROLLBACK 및 SAVEPOINT를 사용하여 데이터베이스 트랜잭션 제어
- 읽기 일관성
- SELECT 문의 FOR UPDATE 절

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 테이블의 데이터 변경

## EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	60
104	Bruce	Ernst	6000	103	(null)	60
107	Diana	Lorentz	4200	103	(null)	60
124	Kevin	Mourgos	5800	100	(null)	50

EMPLOYEES 테이블의 행을 갱신합니다.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	80
104	Bruce	Ernst	6000	103	(null)	80
107	Diana	Lorentz	4200	103	(null)	80
124	Kevin	Mourgos	5800	100	(null)	50

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 테이블의 데이터 변경

슬라이드는 부서 60의 사원에 대해 부서 번호를 부서 80으로 변경하는 과정을 보여줍니다.

## UPDATE 문 구문

- UPDATE 문을 사용하여 테이블의 기존 값을 수정합니다.

```
UPDATE          table
SET            column = value [, column = value, ...]
[WHERE          condition];
```

- 필요한 경우 한 번에 두 개 이상의 행을 갱신합니다.

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### UPDATE 문 구문

UPDATE 문을 사용하여 테이블의 기존 값을 수정할 수 있습니다.

이 구문에서 다음이 적용됩니다.

*table* 테이블의 이름입니다.

*column* 테이블에 채울 열의 이름입니다.

*value* 열의 해당 값 또는 subquery입니다.

*condition* 갱신할 행을 식별하며 열 이름, 표현식, 상수, subquery 및 비교 연산자로 구성됩니다.

테이블을 질의하여 갱신된 행을 표시하고 갱신 작업을 확인합니다.

자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 "UPDATE" 관련 섹션을 참조하십시오.

**참고:** 갱신할 단일 행을 식별하려면 일반적으로 WHERE 절에 Primary Key 열을 사용합니다. 다른 열을 사용하면 예상치 못한 여러 행이 갱신될 수도 있습니다. 예를 들어, EMPLOYEES 테이블의 단일 행을 이름으로 식별하는 것은 위험합니다. 두 명 이상의 사원이 동일한 이름을 가질 수도 있기 때문입니다.

## 테이블의 행 갱신

- WHERE 절을 지정하면 특정 행에서 값이 수정됩니다.

```
UPDATE employees
SET department_id = 50
WHERE employee_id = 113;
1 rows updated
```

- WHERE 절을 생략하면 테이블의 모든 행에서 값이 수정됩니다.

```
UPDATE copy_emp
SET department_id = 110;
22 rows updated
```

- 열 값을 NULL로 갱신하려면 SET column\_name= NULL을 지정합니다.

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### 테이블의 행 갱신

WHERE 절이 지정된 경우 UPDATE 문은 특정 행의 값을 수정합니다. 슬라이드의 예제는 사원 113(Popp)을 부서 50으로 이동하는 것을 보여줍니다.

WHERE 절을 생략하면 테이블의 모든 행에서 값이 수정됩니다. COPY\_EMP 테이블의 갱신된 행을 살펴봅니다.

```
SELECT last_name, department_id
FROM copy_emp;
```

LAST_NAME	DEPARTMENT_ID
Whalen	110
Hartstein	110
Fay	110

...

예를 들어, 직무가 SA\_REPO였던 사원이 IT\_PROG로 직무를 변경한 경우를 가정해 보겠습니다. 이에 따라 해당 사원의 JOB\_ID가 갱신되고 커미션 필드는 NULL로 설정되어야 합니다.

```
UPDATE employees
SET job_id = 'IT_PROG', commission_pct = NULL
WHERE employee_id = 114;
```

참고: COPY\_EMP 테이블에는 EMPLOYEES 테이블과 동일한 데이터가 있습니다.

# Subquery를 사용하여 두 개의 열 갱신

사원 113의 직무와 급여를 사원 205와 일치하도록 갱신합니다.

```
UPDATE employees
SET job_id = (SELECT job_id
               FROM employees
               WHERE employee_id = 205),
    salary = (SELECT salary
               FROM employees
               WHERE employee_id = 205)
WHERE employee_id = 113;
1 rows updated
```

Copyright © 2009, Oracle. All rights reserved.

## Subquery를 사용하여 두 개의 열 갱신

여러 subquery를 작성하여 UPDATE 문의 SET 절에서 여러 열을 갱신할 수 있습니다. 구문은 다음과 같습니다.

```
UPDATE table
SET column =
        (SELECT column
         FROM table
         WHERE condition)
[ ,
column =
        (SELECT column
         FROM table
         WHERE condition)]
[WHERE condition];
```

## Subquery를 사용하여 두 개의 열 갱신(계속)

슬라이드의 예제를 다음과 같이 작성할 수도 있습니다.

```
UPDATE employees
SET ( job_id, salary ) = (SELECT job_id, salary
                           FROM   employees
                           WHERE  employee_id = 205)
WHERE    employee_id      = 113;
```

## 다른 테이블을 기반으로 행 갱신

UPDATE 문에서 subquery를 사용하여 다른 테이블의 값을 기반으로 테이블의 행 값을 갱신합니다.

```
UPDATE copy_emp
SET department_id = (SELECT department_id
                      FROM employees
                      WHERE employee_id = 100)
WHERE job_id = (SELECT job_id
                 FROM employees
                 WHERE employee_id = 200);

1 rows updated
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

### 다른 테이블을 기반으로 행 갱신

UPDATE 문에서 subquery를 사용하여 테이블의 값을 갱신할 수 있습니다. 슬라이드의 예제는 EMPLOYEES 테이블의 값을 기반으로 COPY\_EMP 테이블을 갱신합니다. 이 예제는 사원 200의 직무 ID를 가진 모든 사원의 부서 번호를 사원 100의 현재 부서 번호로 변경합니다.

## 단원 내용

- 테이블에 새 행 추가
  - INSERT 문
- 테이블의 데이터 변경
  - UPDATE 문
- 테이블에서 행 제거
  - DELETE 문
  - TRUNCATE 문
- COMMIT, ROLLBACK 및 SAVEPOINT를 사용하여 데이터베이스 트랜잭션 제어
- 읽기 일관성
- SELECT 문의 FOR UPDATE 절

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 테이블에서 행 제거

## DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

DEPARTMENTS 테이블에서 행을 삭제합니다.

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 테이블에서 행 제거

슬라이드의 그림에 나오는 것처럼 DEPARTMENTS 테이블에서 Contracting 부서가 제거되었습니다.  
본 과정에서 DEPARTMENTS 테이블에 대한 제약 조건이 위반되지 않은 것으로 가정합니다.

# **DELETE 문**

**DELETE 문을 사용하여 테이블에서 기존 행을 제거할 수 있습니다.**

```
DELETE [FROM]    table  
[WHERE          condition];
```

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

## **DELETE 문 구문**

DELETE 문을 사용하여 테이블에서 기존 행을 제거할 수 있습니다.

이 구문에서 다음이 적용됩니다.

*table*                      테이블의 이름입니다.

*condition*                삭제할 행을 식별하며 열 이름, 표현식, 상수, subquery 및 비교 연산자로 구성됩니다.

**참고:** 삭제된 행이 없는 경우 SQL Developer의 Script Output 탭에 "0 rows deleted"라는 메시지가 반환됩니다.

자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 "DELETE" 관련 섹션을 참조하십시오.

## 테이블에서 행 삭제

- WHERE 절을 지정하면 특정 행이 삭제됩니다.

```
DELETE FROM departments
WHERE department_name = 'Finance';
1 rows deleted
```

- WHERE 절을 생략하면 테이블의 모든 행이 삭제됩니다.

```
DELETE FROM copy_emp;
22 rows deleted
```

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### 테이블에서 행 삭제

DELETE 문에 WHERE 절을 지정하여 특정 행을 삭제할 수 있습니다. 슬라이드의 첫번째 예제는 DEPARTMENTS 테이블에서 Accounting 부서를 삭제합니다. SELECT 문을 사용하여 삭제된 행을 표시하는 방식으로 삭제 작업을 확인할 수 있습니다.

```
SELECT *
FROM departments
WHERE department_name = 'Finance';
0 rows selected
```

그러나 WHERE 절을 생략하면 테이블의 모든 행이 삭제됩니다. 슬라이드의 두번째 예제는 WHERE 절이 지정되지 않았기 때문에 COPY\_EMP 테이블에서 모든 행을 삭제합니다.

**예제:**

WHERE 절에서 식별된 행을 제거합니다.

```
DELETE FROM employees WHERE employee_id = 114;
1 rows deleted
```

```
DELETE FROM departments WHERE department_id IN (30, 40);
2 rows deleted
```

## 다른 테이블을 기반으로 행 삭제

DELETE 문에서 subquery를 사용하여 다른 테이블의 값을 기반으로 테이블에서 행을 제거합니다.

```
DELETE FROM employees
WHERE department_id =
    (SELECT department_id
     FROM departments
     WHERE department_name
           LIKE '%Public%');

1 rows deleted
```

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 다른 테이블을 기반으로 행 삭제

subquery를 사용하여 다른 테이블의 값을 기반으로 테이블에서 행을 삭제할 수 있습니다. 슬라이드의 예제는 부서 이름에 문자열 Public이 포함된 부서에서 근무하는 모든 사원을 삭제합니다.

subquery는 문자열 Public을 포함한 부서 이름을 기반으로 DEPARTMENTS 테이블을 검색하여 부서 번호를 찾습니다. 그러면 subquery가 main query에 부서 번호를 전달하고 main query는 이 부서 번호를 기반으로 EMPLOYEES 테이블에서 데이터 행을 삭제합니다.

## TRUNCATE 문

- 테이블은 빈 상태로, 테이블 구조는 그대로 남겨둔 채 테이블에서 모든 행을 제거합니다.
- DML 문이 아니라 DDL(데이터 정의어) 문이므로 쉽게 언두할 수 없습니다.
- 구문:

```
TRUNCATE TABLE table_name;
```

- 예제:

```
TRUNCATE TABLE copy_emp;
```

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

## TRUNCATE 문

테이블을 비우는 좀 더 효율적인 방법은 TRUNCATE 문을 사용하는 것입니다. TRUNCATE 문을 사용하면 테이블이나 클러스터에서 모든 행을 신속하게 제거할 수 있습니다. TRUNCATE 문으로 행을 제거하는 방법이 DELETE 문으로 행을 제거하는 방법보다 빠른 이유는 다음과 같습니다.

- TRUNCATE 문은 DDL(데이터 정의어) 문이며 롤백 정보를 생성하지 않습니다. 롤백 정보는 이 단원 뒷부분에서 다룹니다.
- 테이블 자르기는 테이블의 삭제 트리거를 유발하지 않습니다.

테이블이 참조 무결성 제약 조건의 상위 요소인 경우 해당 테이블을 truncate 할 수 없습니다. TRUNCATE 문을 실행하기 전에 제약 조건을 비활성화해야 합니다. 제약 조건 비활성화에 대해서는 "DDL 문을 사용하여 테이블 생성 및 관리" 단원에서 설명합니다.

## 단원 내용

- 테이블에 새 행 추가
  - INSERT 문
- 테이블의 데이터 변경
  - UPDATE 문
- 테이블에서 행 제거
  - DELETE 문
  - TRUNCATE 문
- COMMIT, ROLLBACK 및 SAVEPOINT를 사용하여 데이터베이스 트랜잭션 제어
- 읽기 일관성
- SELECT 문의 FOR UPDATE 절

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 데이터베이스 트랜잭션

**데이터베이스 트랜잭션은 다음 중 하나로 구성됩니다.**

- 데이터를 일관되게 변경하는 여러 DML 문
- 하나의 DDL 문
- 하나의 DCL(데이터 제어어) 문

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

## 데이터베이스 트랜잭션

Oracle 서버는 트랜잭션에 준하여 데이터 일관성이 유지되도록 합니다. 트랜잭션은 데이터를 변경할 때 많은 유연성과 제어 기능을 제공하며 유저 프로세스 failure 또는 시스템 failure 시 데이터 일관성을 보장합니다.

트랜잭션은 데이터를 일관되게 변경하는 여러 DML 문으로 구성됩니다. 예를 들어, 두 계좌 간에 자금을 이체할 때 한 계좌의 차변과 다른 계좌의 대변에서 변경되는 금액이 동일해야 합니다. 이 두 작업은 동시에 성공하거나 실패해야 하며 차변 없이 대변만 커밋될 수는 없습니다.

### 트랜잭션 유형

유형	설명
DML(데이터 조작어)	Oracle 서버가 단일 엔티티나 논리적 작업 단위로 취급하는 임의 개수의 DML 문으로 구성됩니다.
DDL(데이터 정의어)	하나의 DDL 문으로만 구성됩니다.
DCL(데이터 제어어)	하나의 DCL 문으로만 구성됩니다.

## 데이터베이스 트랜잭션: 시작과 종료

- 첫번째 DML SQL 문이 실행될 때 시작됩니다.
- 다음 상황 중 하나가 발생하면 종료됩니다.
  - COMMIT 또는 ROLLBACK 문 실행
  - DDL 또는 DCL 문 실행(자동 커밋)
  - 유저가 SQL Developer 또는 SQL\*Plus를 종료
  - 시스템 중단

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 데이터베이스 트랜잭션: 시작과 종료

데이터베이스 트랜잭션이 언제 시작되고 종료되는지 알아보겠습니다.

트랜잭션은 첫번째 DML 문을 만나면 시작되고 다음 상황 중 하나가 발생하면 종료됩니다.

- COMMIT 또는 ROLLBACK 문 실행
- CREATE와 같은 DDL 문 실행
- DCL 문 실행
- 유저가 SQL Developer 또는 SQL\*Plus를 종료
- 시스템 failure 또는 시스템 중단

한 트랙잭션이 끝나면 다음 실행 가능한 SQL 문이 다음 트랜잭션을 자동으로 시작합니다.

DDL 문 또는 DCL 문은 자동으로 커밋되기 때문에 트랜잭션을 암시적으로 종료합니다.

## COMMIT 및 ROLLBACK 문의 이점

COMMIT 및 ROLLBACK 문을 사용할 경우 다음과 같은 이점이 있습니다.

- 데이터 일관성 보장
- 변경 사항을 영구 적용하기 전에 데이터 변경 사항 검토
- 논리적으로 관련된 작업 그룹화

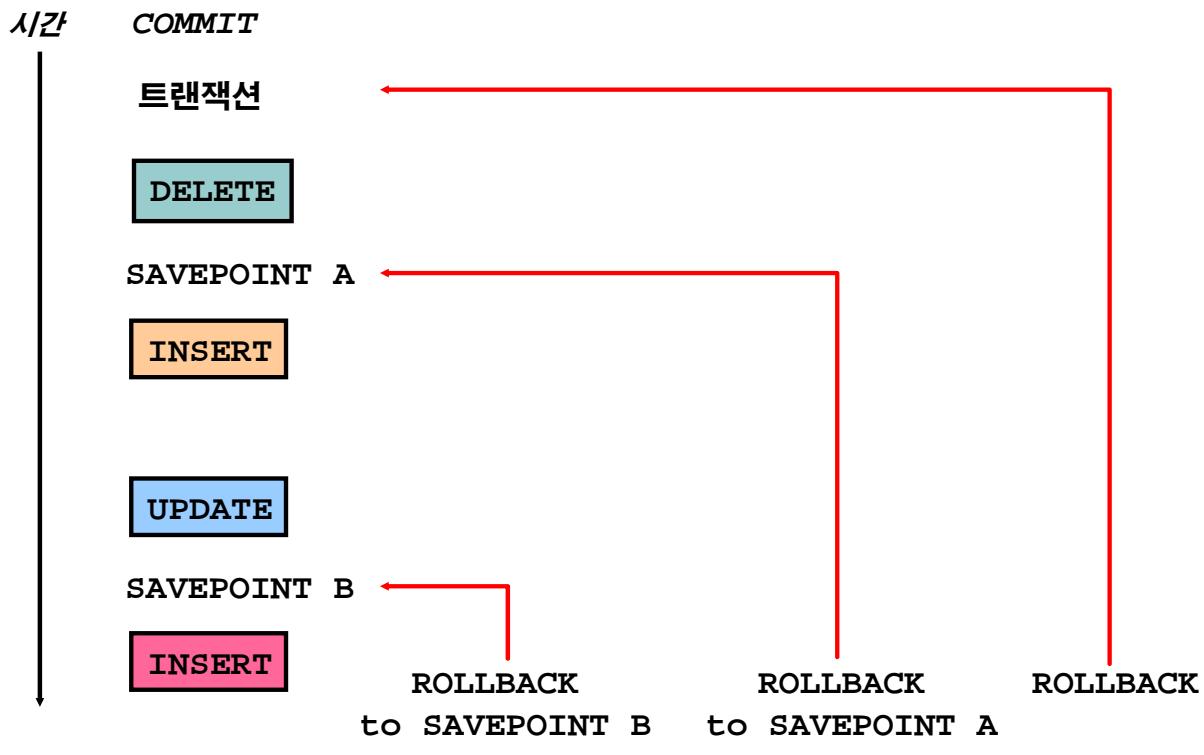
ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## COMMIT 및 ROLLBACK 문의 단점

COMMIT 및 ROLLBACK 문을 사용하면 데이터 변경 사항의 영구 적용 여부를 제어 할 수 있습니다.

# 명시적 트랜잭션 제어문



ORACLE

Copyright © 2009, Oracle. All rights reserved.

## 명시적 트랜잭션 제어문

`COMMIT`, `SAVEPOINT` 및 `ROLLBACK` 문을 사용하여 트랜잭션 논리를 제어할 수 있습니다.

명령문	설명
<code>COMMIT</code>	<code>COMMIT</code> 은 보류 중인 모든 데이터 변경 사항을 영구적으로 적용하여 현재 트랜잭션을 종료합니다.
<code>SAVEPOINT name</code>	<code>SAVEPOINT name</code> 은 현재 트랜잭션 내에 저장점을 표시합니다.
<code>ROLLBACK</code>	<code>ROLLBACK</code> 은 보류 중인 모든 데이터 변경 사항을 폐기하여 현재 트랜잭션을 종료합니다.
<code>ROLLBACK TO SAVEPOINT name</code>	<code>ROLLBACK TO SAVEPOINT</code> 는 현재 트랜잭션을 지정된 저장점으로 롤백합니다. 이에 따라 롤백 중인 저장점 이후에 생성된 변경 사항 및/또는 저장점은 폐기됩니다. <code>TO SAVEPOINT</code> 절을 생략하면 <code>ROLLBACK</code> 문은 전체 트랜잭션을 롤백합니다. 저장점은 논리적이기 때문에 생성한 저장점을 리스트로 표시할 수 없습니다.

참고: `SAVEPOINT`로 커밋 할 수는 없습니다. `SAVEPOINT`는 ANSI 표준 SQL이 아닙니다.

## 변경 사항을 표시자로 롤백

- **SAVEPOINT 문을 사용하여 현재 트랜잭션에서 표시자를 생성합니다.**
- **ROLLBACK TO SAVEPOINT 문을 사용하여 해당 표시자로 롤백합니다.**

```
UPDATE...
SAVEPOINT update_done;
SAVEPOINT update_done succeeded.

INSERT...
ROLLBACK TO update_done;
ROLLBACK TO succeeded.
```



Copyright © 2009, Oracle. All rights reserved.

### 변경 사항을 표시자로 롤백

트랜잭션을 작은 섹션으로 나누는 SAVEPOINT 문을 사용하여 현재 트랜잭션에서 표시자를 생성할 수 있습니다. 그런 다음 ROLLBACK TO SAVEPOINT 문을 사용하여 해당 표시자에 보류 중인 변경 사항을 폐기할 수 있습니다.

이전의 저장점과 동일한 이름으로 두번째 저장점을 만들면 이전의 저장점이 삭제됩니다.

## 암시적 트랜잭션 처리

- 자동 커밋은 다음 상황에서 발생합니다.
  - DDL 문이 실행되는 경우
  - DCL 문이 실행되는 경우
  - COMMIT 또는 ROLLBACK 문을 명시적으로 실행하지 않은 채 SQL Developer 또는 SQL\*Plus가 정상적으로 종료된 경우
- SQL Developer 또는 SQL\*Plus가 비정상적으로 종료되거나 시스템 failure가 발생된 경우 자동 롤백이 발생합니다.

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### 암시적 트랜잭션 처리

상태	상황
자동 커밋	DDL 문 또는 DCL 문 실행 COMMIT 또는 ROLLBACK 명령을 명시적으로 실행하지 않은 채 SQL Developer 또는 SQL*Plus를 정상적으로 종료
자동 롤백	SQL Developer 또는 SQL*Plus가 비정상적으로 종료되거나 시스템 failure가 발생

**참고:** SQL\*Plus에서는 AUTOCOMMIT 명령에 대한 설정을 ON 또는 OFF로 변경할 수 있습니다. ON으로 설정된 경우 각 DML 문은 실행되는 즉시 커밋됩니다. 변경 사항은 롤백할 수 없습니다. OFF로 설정된 경우 COMMIT 문은 계속 명시적으로 실행될 수 있습니다. 또한 COMMIT 문은 DDL 문이 실행되거나 SQL\*Plus가 종료되는 경우에도 실행됩니다. SQL Developer에서는 SET AUTOCOMMIT ON/OFF 명령을 실행하지 않고 건너뜁니다. Autocommit 환경 설정을 활성화한 경우에만 SQL Developer 정상 종료 시 DML 문이 커밋됩니다. Autocommit를 활성화하려면 다음을 수행합니다.

- Tools 메뉴에서 Preferences를 선택합니다. Preferences 대화상자에서 Database를 확장하고 Worksheet Parameters를 선택합니다.
- 오른쪽 창에서 "Autocommit in SQL Worksheet" 옵션을 선택합니다. OK를 누릅니다.

## 암시적 트랜잭션 처리(계속)

### 시스템 failure

시스템 failure로 트랜잭션이 중단될 경우 전체 트랜잭션은 자동으로 롤백됩니다. 이렇게 하면 오류로 인해 데이터가 원하지 않는 방식으로 변경되는 것을 방지할 수 있으며 마지막 커밋 작업 시의 상태로 테이블을 반환합니다. Oracle 서버는 이러한 방식으로 테이블의 무결성을 보호합니다.

SQL Developer에서 세션을 정상적으로 종료하려면 File 메뉴에서 Exit를 선택하고 SQL\*Plus에서 세션을 정상적으로 종료하려면 프롬프트에서 EXIT 명령을 입력합니다. window를 닫는 것은 비정상적인 종료로 간주됩니다.

## COMMIT 또는 ROLLBACK 전의 데이터 상태

- 이전의 데이터 상태를 복구할 수 있습니다.
- 현재 유저는 SELECT 문을 사용하여 DML 작업의 결과를 확인할 수 있습니다.
- 다른 유저는 현재 유저가 실행한 DML 문의 결과를 볼 수 없습니다.
- 영향을 받는 행이 잠기므로 다른 유저가 영향을 받는 행의 데이터를 변경할 수 없습니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### COMMIT 또는 ROLLBACK 전의 데이터 상태

트랜잭션 동안의 모든 데이터 변경 사항은 트랜잭션이 커밋되기 전까지는 임시적인 상태입니다.

다음은 COMMIT 또는 ROLLBACK 명령이 실행되기 전의 데이터 상태에 대한 설명입니다.

- 데이터 조작 작업은 기본적으로 데이터베이스 버퍼에 영향을 주기 때문에 이전의 데이터 상태를 복구할 수 있습니다.
- 현재 유저는 테이블을 query하여 데이터 조작 작업의 결과를 확인할 수 있습니다.
- 다른 유저는 현재 유저의 데이터 조작 작업 결과를 확인할 수 없습니다. Oracle 서버는 읽기 일관성을 제공하여 각 유저가 마지막 커밋된 시점의 상태대로 데이터를 볼 수 있도록 보장합니다.
- 영향을 받는 행이 잠기므로 다른 유저가 영향을 받는 행의 데이터를 변경할 수 없습니다.

## COMMIT 후의 데이터 상태

- 데이터 변경 사항이 데이터베이스에 저장됩니다.
- 이전의 데이터 상태를 겹쳐씁니다.
- 모든 유저가 결과를 확인할 수 있습니다.
- 영향을 받는 행의 Lock이 해제되어 이러한 행을 다른 유저가 조작할 수 있습니다.
- 모든 저장점이 지워집니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### COMMIT 후의 데이터 상태

COMMIT 명령을 사용하여 보류 중인 모든 변경 사항을 영구 적용합니다. 다음은 COMMIT 문이 실행된 이후에 발생하는 상황입니다.

- 데이터 변경 사항이 데이터베이스에 기록됩니다.
- 더 이상 정상적인 SQL query를 통해 이전의 데이터 상태를 사용할 수 없습니다.
- 모든 유저가 트랜잭션 결과를 확인할 수 있습니다.
- 영향 받는 행의 잠금이 해제되고 다른 유저가 해당 행에 대해 새로운 데이터 변경 작업을 수행할 수 있습니다.
- 모든 저장점이 지워집니다.

# 데이터 커밋

- 데이터를 변경합니다.

```
DELETE FROM employees
WHERE employee_id = 99999;
1 rows deleted

INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 rows inserted
```

- 변경 사항을 커밋합니다.

```
COMMIT;
COMMIT succeeded.
```

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 데이터 커밋

슬라이드의 예제에서는 EMPLOYEES 테이블에서 행이 삭제되고 DEPARTMENTS 테이블에 새 행이 삽입됩니다. 이러한 변경 사항은 COMMIT 문을 실행하여 저장됩니다.

예제:

DEPARTMENTS 테이블에서 부서 290 및 300을 제거하고 EMPLOYEES 테이블의 행을 생성합니다. 데이터 변경 사항을 저장합니다.

```
DELETE FROM departments
WHERE department_id IN (290, 300);

UPDATE employees
SET department_id = 80
WHERE employee_id = 206;

COMMIT;
```

## ROLLBACK 후의 데이터 상태

ROLLBACK 문을 사용하여 보류 중인 모든 변경 사항을 폐기합니다.

- 데이터 변경 사항이 실행 취소됩니다.
- 이전의 데이터 상태가 복원됩니다.
- 영향 받는 행의 잠금이 해제됩니다.

```
DELETE FROM copy_emp;  
ROLLBACK ;
```

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### ROLLBACK 후의 데이터 상태

ROLLBACK 문을 사용하여 보류 중인 모든 변경 사항을 폐기합니다. 다음과 같은 결과가 발생합니다.

- 데이터 변경 사항이 실행 취소됩니다.
- 이전의 데이터 상태가 복원됩니다.
- 영향 받는 행의 잠금이 해제됩니다.

## ROLLBACK 후의 데이터 상태: 예제

```
DELETE FROM test;
25,000 rows deleted.

ROLLBACK;
Rollback complete.

DELETE FROM test WHERE id = 100;
1 row deleted.

SELECT * FROM test WHERE id = 100;
No rows selected.

COMMIT;
Commit complete.
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

### ROLLBACK 후의 데이터 상태: 예제

TEST 테이블에서 레코드를 제거하려다 실수로 테이블을 모두 비웠습니다. 그러나 이러한 실수를 바로잡고 적절한 명령문을 다시 실행하여 데이터 변경 사항을 영구 적용할 수 있습니다.

## 명령문 레벨 롤백

- 단일 DML 문을 실행하는 중에 오류가 발생하면 해당 명령문만 롤백됩니다.
- Oracle 서버는 저장점을 암시적으로 구현합니다.
- 다른 모든 변경 사항은 보존됩니다.
- 유저는 COMMIT 또는 ROLLBACK 문을 실행하여 트랜잭션을 명시적으로 종료해야 합니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 명령문 레벨 롤백

명령문 실행 오류가 감지되면 암시적 롤백으로 트랜잭션의 일부가 폐기될 수 있습니다. 트랜잭션 실행 중 단일 DML 문에 오류가 발생하면 명령문이 미치는 영향은 명령문 레벨 롤백에 의해 언두되지만, 트랜잭션에서 이전 DML 문에 의해 변경된 사항은 폐기되지 않습니다. 이러한 이전의 변경 사항은 유저가 명시적으로 커밋하거나 롤백할 수 있습니다.

Oracle 서버는 DDL 문을 실행하기 전과 실행한 후에 암시적 커밋을 실행합니다. 따라서 DDL 문이 성공적으로 실행되지 않더라도 서버가 이미 커밋을 실행했기 때문에 이전 명령문을 롤백할 수 없습니다.

COMMIT 또는 ROLLBACK 문을 실행하여 트랜잭션을 명시적으로 종료합니다.

## 단원 내용

- 테이블에 새 행 추가
  - INSERT 문
- 테이블의 데이터 변경
  - UPDATE 문
- 테이블에서 행 제거
  - DELETE 문
  - TRUNCATE 문
- COMMIT, ROLLBACK 및 SAVEPOINT를 사용하여 데이터베이스 트랜잭션 제어
- 읽기 일관성
- SELECT 문의 FOR UPDATE 절

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 읽기 일관성

- 읽기 일관성은 데이터에 대한 일관성 있는 뷰를 보장합니다.
- 한 유저가 변경한 사항이 다른 유저가 변경한 사항과 충돌하지 않습니다.
- 읽기 일관성은 동일한 데이터에 대해 다음 사항을 보장합니다.
  - 읽는 사람은 쓰는 사람의 작업이 완료되기를 기다릴 필요가 없습니다.
  - 쓰는 사람은 읽는 사람의 작업이 완료되기를 기다릴 필요가 없습니다.
  - 쓰는 사람은 다른 쓰는 사람의 작업이 완료되기를 기다려야 합니다.

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### 읽기 일관성

데이터베이스 유저는 다음과 같은 두 가지 방식으로 데이터베이스를 액세스합니다.

- 읽기 작업(SELECT 문)
- 쓰기 작업(INSERT, UPDATE, DELETE 문)

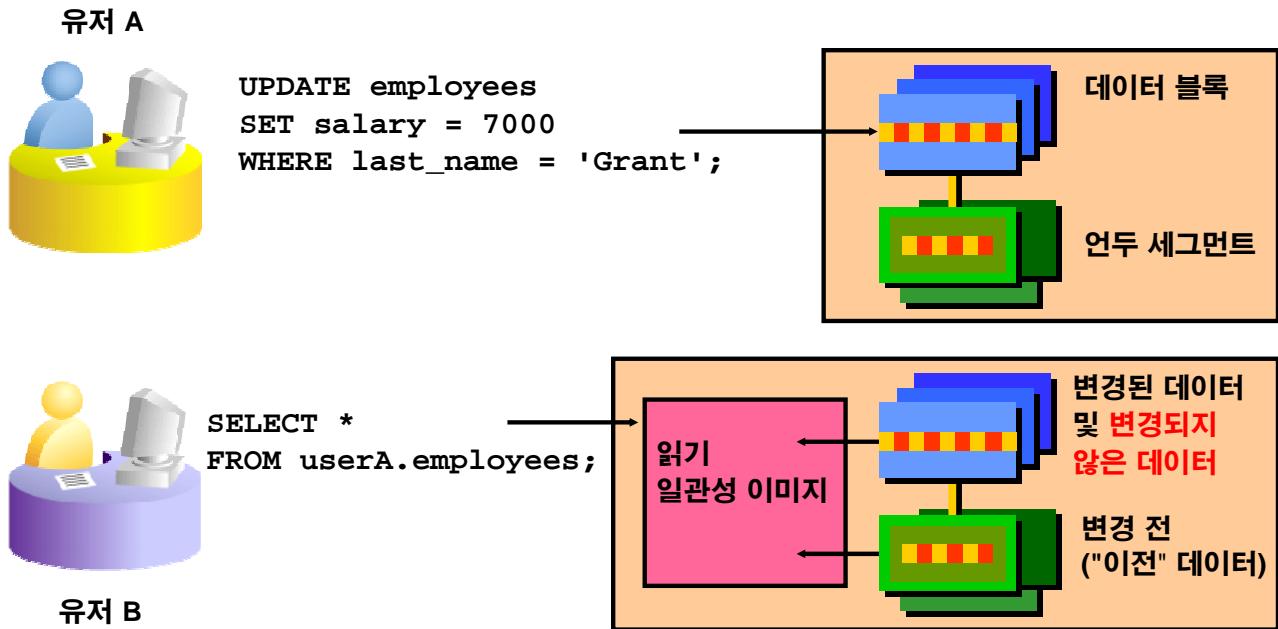
읽기 일관성을 유지할 경우 다음과 같은 효과를 볼 수 있습니다.

- 데이터베이스 기록자와 읽는 사람이 일관된 데이터 뷰를 볼 수 있습니다.
- 현재 변경 중인 데이터는 읽는 사람이 볼 수 없습니다.
- 쓰는 사람은 데이터베이스 변경 작업을 일관되게 수행할 수 있습니다.
- 쓰는 사람이 변경한 사항은 다른 쓰는 사람이 변경하는 사항과 충돌하지 않습니다.

읽기 일관성의 목적은 유저가 DML 작업을 시작하기 전 마지막 커밋 시점의 상태대로 데이터를 볼 수 있도록 보장하는 것입니다.

**참고:** 동일한 유저가 서로 다른 세션으로 로그인할 수 있습니다. 여러 세션에서 유저가 동일한 경우에도 각 세션은 위에 설명된 방식으로 읽기 일관성을 유지합니다.

## 읽기 일관성 구현



ORACLE

Copyright © 2009, Oracle. All rights reserved.

### 읽기 일관성 구현

읽기 일관성은 자동으로 구현됩니다. 읽기 일관성은 데이터베이스의 일부 복사본을 언두 세그먼트에 보관합니다. 읽기 일관성을 제공하는 이미지는 테이블에서 커밋된 데이터와 언두 세그먼트에서 아직 커밋되지 않은 변경 중인 이전 데이터로 구성됩니다.

데이터베이스에 대한 삽입, 갱신, 삭제 등의 작업을 수행하면 Oracle 서버에서는 데이터가 변경되기 전에 데이터 복사본을 만들어 이를 언두 세그먼트에 기록합니다.

변경 작업을 직접 수행한 유저 이외의 모든 읽는 사람에게는 변경 전 상태의 데이터베이스, 즉 언두 세그먼트의 데이터 "스냅샷"이 표시됩니다.

변경 사항이 데이터베이스에 커밋되기 전에는 해당 데이터를 수정하는 유저만 데이터베이스에서 변경 사항을 볼 수 있습니다. 그 밖의 모든 사람은 언두 세그먼트의 스냅샷을 볼 수 있습니다. 이렇게 하면 해당 데이터를 읽는 사람들은 현재 변경 중이 아닌 일관된 데이터를 보게 됩니다.

DML 문이 커밋되는 경우 커밋이 완료된 후에 SELECT 문을 실행하는 모든 사람이 데이터베이스에 대한 변경 사항을 볼 수 있습니다. 언두 세그먼트 파일의 이전 데이터가 점유한 공간은 다시 사용할 수 있도록 해제됩니다.

트랜잭션을 롤백하면 변경 내용이 취소됩니다.

- 언두 세그먼트에 들어 있던 이전의 원래 데이터가 테이블에 다시 기록됩니다.
- 모든 유저는 데이터베이스를 트랜잭션 시작 전의 상태로 보게 됩니다.

## 단원 내용

- 테이블에 새 행 추가
  - INSERT 문
- 테이블의 데이터 변경
  - UPDATE 문
- 테이블에서 행 제거
  - DELETE 문
  - TRUNCATE 문
- COMMIT, ROLLBACK 및 SAVEPOINT를 사용하여 데이터베이스 트랜잭션 제어
- 읽기 일관성
- SELECT 문의 FOR UPDATE 절

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## SELECT 문의 FOR UPDATE 절

- EMPLOYEES 테이블에서 job\_id가 SA\_REP인 행을 잠금니다.

```
SELECT employee_id, salary, commission_pct, job_id
FROM employees
WHERE job_id = 'SA_REP'
FOR UPDATE
ORDER BY employee_id;
```

- ROLLBACK 또는 COMMIT를 실행하는 경우에만 Lock이 해제됩니다.
- 다른 유저가 접근 행을 SELECT 문에서 잠그려고 하면 데이터베이스는 해당 행을 사용할 수 있을 때까지 기다린 다음 SELECT 문의 결과를 반환합니다.

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### SELECT 문의 FOR UPDATE 절

일부 레코드를 query하기 위해 데이터베이스에 대해 SELECT 문을 실행하는 경우 선택된 행에는 Lock이 걸리지 않습니다. 일반적으로 특정 시점에서 잠겨 있는 레코드 수는 기본적으로 절대 최소치로 유지되므로, 즉 변경되었지만 아직 커밋되지 않은 레코드만 잠기므로 SELECT 문에 의해 선택된 행이 잠기지 않아야 합니다. 레코드가 잠긴 경우에도 다른 유저는 변경되기 전의 모습, 즉 데이터의 "이전 이미지"로 해당 레코드를 읽을 수 있습니다. 그러나 때로는 사용 중인 프로그램에서 레코드 집합을 변경하지 않았어도 해당 레코드 집합을 잠그려는 경우가 있습니다. Oracle은 이러한 잠금을 수행할 수 있도록 SELECT 문의 FOR UPDATE 절을 제공합니다.

SELECT...FOR UPDATE 문을 실행하면 RDBMS(관계형 데이터베이스 관리 시스템)는 SELECT 문에 의해 식별된 모든 행에 대해 배타적 행 레벨 잠금(exclusive row-level lock)을 자동으로 획득하여 "해당 유저만 변경할 수 있는" 레코드를 보유하게 됩니다. 해당 유저가 ROLLBACK 또는 COMMIT를 수행하기 전까지 다른 유저는 이러한 레코드를 변경할 수 없습니다. FOR UPDATE 절에 선택적 키워드인 NOWAIT를 추가하면 다른 유저가 해당 테이블을 접근 경우 Oracle 서버가 기다리지 않도록 할 수 있습니다. 이 경우 사용 중인 프로그램이나 SQL Developer 환경으로 제어가 즉시 반환되므로 유저는 다른 작업을 수행하거나 일정 시간 동안 기다린 후 다시 시도할 수 있습니다. NOWAIT 절을 사용하지 않으면 해당 테이블을 사용할 수 있을 때까지, 즉 다른 유저가 COMMIT 또는 ROLLBACK 명령을 실행하여 Lock이 해제될 때까지 프로세스가 차단됩니다.

## FOR UPDATE 절: 예제

- SELECT 문에서 여러 테이블에 대해 FOR UPDATE 절을 사용할 수 있습니다.

```
SELECT e.employee_id, e.salary, e.commission_pct
FROM employees e JOIN departments d
USING (department_id)
WHERE job_id = 'ST_CLERK'
AND location_id = 1500
FOR UPDATE
ORDER BY e.employee_id;
```

- EMPLOYEES 테이블 및 DEPARTMENTS 테이블의 행이 잠깁니다.
- FOR UPDATE OF *column\_name*을 사용하여 변경할 열을 지정하면 특정 테이블의 해당 행만 잠깁니다.

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### FOR UPDATE 절: 예제

슬라이드의 예제에서 명령문은 EMPLOYEES 테이블에서 JOB\_ID가 ST\_CLERK으로 설정되고 LOCATION\_ID가 1500으로 설정된 행을 잠그고 DEPARTMENTS 테이블에서는 LOCATION\_ID가 1500으로 설정된 행을 잠깁니다.

FOR UPDATE OF *column\_name*을 사용하여 변경할 열을 지정할 수 있습니다. 선택된 행에서 변경할 수 있는 열은 FOR UPDATE 절의 OF 리스트에 나열된 열로 제한되지 않습니다. 모든 행에는 여전히 Lock이 걸려 있으므로 query에 FOR UPDATE만 지정하고 OF 키워드 뒤에 하나 이상의 열을 포함하지 않는 경우 데이터베이스는 FROM 절에 나열된 모든 테이블에서 식별된 모든 행을 잠깁니다.

다음 명령문은 EMPLOYEES 테이블에서 LOCATION\_ID 1500에 거주하는 ST\_CLERK을 값으로 가지는 행만 잠깁니다. DEPARTMENTS 테이블의 행은 잠기지 않습니다.

```
SELECT e.employee_id, e.salary, e.commission_pct
FROM employees e JOIN departments d
USING (department_id)
WHERE job_id = 'ST_CLERK' AND location_id = 1500
FOR UPDATE OF e.salary
ORDER BY e.employee_id;
```

### FOR UPDATE 절: 예제(계속)

다음 예제에서 데이터베이스는 행을 사용할 수 있을 때까지 5초간 기다린 후 유저에게 제어를 반환합니다.

```
SELECT employee_id, salary, commission_pct, job_id  
FROM employees  
WHERE job_id = 'SA_REP'  
FOR UPDATE WAIT 5  
ORDER BY employee_id;
```

## 퀴즈

다음 명령문은 동일한 결과를 생성합니다.

```
DELETE FROM copy_emp;
```

```
TRUNCATE TABLE copy_emp;
```

1. 맞습니다.
2. 틀립니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

정답: 2

# 요약

이 단원에서는 다음 명령문의 사용 방법에 대해 설명했습니다.

함수	설명
INSERT	테이블에 새 행 추가
UPDATE	테이블의 기존 행 수정
DELETE	테이블에서 기존 행 제거
TRUNCATE	테이블에서 모든 행 제거
COMMIT	보류 중인 변경 사항을 영구적으로 적용
SAVEPOINT	저장점 표시자로 롤백하는 데 사용
ROLLBACK	보류 중인 모든 데이터 변경 사항 폐기
FOR UPDATE clause in SELECT	SELECT query에 의해 식별된 행 잠그기

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 요약

이 단원에서는 INSERT, UPDATE, DELETE, TRUNCATE 문을 사용하여 오라클 데이터베이스의 데이터를 조작하는 방법과 COMMIT, SAVEPOINT, ROLLBACK 문을 사용하여 데이터 변경 사항을 제어하는 방법을 배웠습니다. 또한 SELECT 문의 FOR UPDATE 절을 사용하여 다른 유저가 변경할 수 없도록 행을 잠그는 방법도 배웠습니다.

Oracle 서버는 항상 데이터를 일관되게 볼 수 있도록 보장한다는 사실을 기억하십시오.

## 연습 9: 개요

이 연습에서는 다음 내용을 다룹니다.

- 테이블에 행 삽입
- 테이블에서 행 생성 및 삭제
- 트랜잭션 제어

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 연습 9: 개요

이 연습에서는 MY\_EMPLOYEE 테이블에 행을 추가하고, 테이블에서 데이터를 생성 및 삭제하고, 트랜잭션을 제어하는 과정을 다룹니다. 스크립트를 실행하여 MY\_EMPLOYEE 테이블을 생성합니다.



# **DDL 문을 사용하여 테이블 생성 및 관리**

10

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

## 목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 기본 데이터베이스 객체 분류
- 테이블 구조 검토
- 열에 사용할 수 있는 데이터 유형 나열
- 간단한 테이블 생성
- 테이블 생성 시 제약 조건이 생성되는 방식 설명
- 스키마 객체의 작동 방식 설명

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 목표

이 단원에서는 DDL(데이터 정의어) 문을 소개합니다. 간단한 테이블을 생성, 변경 및 제거하는 방법을 배웁니다. DDL에서 사용할 수 있는 데이터 유형을 살펴보고 스키마 개념이 소개됩니다. 이 단원에서 제약 조건에 대해 설명합니다. DML 작업 동안 제약 조건 위반으로 생성되는 예외 메시지에 대해 살펴봅니다.

## 단원 내용

- 데이터베이스 객체:
  - 이름 지정 규칙
- CREATE TABLE 문:
  - 다른 유저의 테이블에 액세스
  - DEFAULT 옵션
- 데이터 유형
- 제약 조건 개요: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK 제약 조건
- subquery를 사용하여 테이블 생성
- ALTER TABLE
  - 읽기 전용 테이블
- DROP TABLE 문

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 데이터베이스 객체

객체	설명
테이블	기본 저장 단위이며 행으로 구성되어 있습니다.
뷰	하나 이상의 테이블에 있는 데이터의 부분 집합을 논리적으로 나타냅니다.
시퀀스	숫자 값을 생성합니다.
인덱스	일부 query 성능을 향상시킵니다.
동의어	객체에 다른 이름을 부여합니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 데이터베이스 객체

오라클 데이터베이스는 여러 데이터 구조를 포함할 수 있습니다. 각 구조는 데이터베이스 개발 과정의 구축 단계에서 생성될 수 있도록 데이터베이스 설계 시에 기본적인 틀을 만들어야 합니다.

- **테이블:** 데이터를 저장합니다.
- **뷰:** 하나 이상의 테이블에 있는 데이터의 부분 집합입니다.
- **시퀀스:** 숫자 값을 생성합니다.
- **인덱스:** 일부 질의 성능을 향상시킵니다.
- **동의어:** 객체에 다른 이름을 부여합니다.

### Oracle 테이블 구조

- 유저가 데이터베이스를 사용하고 있는 경우를 포함하여 언제든지 테이블을 생성할 수 있습니다.
- 테이블의 크기를 지정할 필요는 없습니다. 크기는 최종적으로 데이터베이스에 전체적으로 할당된 공간의 크기로 정의됩니다. 하지만 시간이 지남에 따라 테이블이 사용할 공간을 산정하는 것이 중요합니다.
- 테이블 구조는 온라인으로 수정할 수 있습니다.

**참고:** 이 밖에도 많은 데이터베이스 객체가 있지만 본 과정에서는 다루지 않습니다.

## 이름 지정 규칙

**테이블 이름 및 열 이름은 다음 규칙을 따라야 합니다.**

- 문자로 시작해야 합니다.
- 길이는 1–30자 사이여야 합니다.
- A–Z, a–z, 0–9, \_, \$, #만 포함할 수 있습니다.
- 동일한 유저가 소유한 다른 객체의 이름과 중복되지 않아야 합니다.
- Oracle 서버 예약어는 사용할 수 없습니다.

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### 이름 지정 규칙

오라클 데이터베이스 객체에 대한 이름 지정 표준 규칙에 따라 데이터베이스 테이블 및 열 이름을 다음과 같이 지정합니다.

- 테이블 이름과 열 이름은 문자로 시작하고 길이는 1–30자 사이여야 합니다.
- 이름에는 문자 A–Z, a–z, 0–9, \_(밑줄), \$, #(규칙에 위배되지 않지만 권장되지 않음) 만 사용할 수 있습니다.
- 이름은 동일한 Oracle 서버 유저가 소유한 다른 객체의 이름과 중복되면 안됩니다.
- Oracle 서버 예약어는 이름으로 사용할 수 없습니다.
  - 따옴표로 묶은 식별자를 사용하여 객체의 이름을 나타낼 수도 있습니다. 따옴표로 묶은 식별자는 앞과 뒤에 큰 따옴표("")를 붙입니다. 따옴표로 묶은 식별자를 사용하여 스키마 객체에 이름을 지정하면 해당 객체를 참조할 때마다 큰 따옴표를 사용해야 합니다. 권장 사항은 아니지만 따옴표로 묶은 식별자를 예약어로 사용할 수 있습니다.

### 이름 지정 지침

테이블 및 기타 데이터베이스 객체에 대해 설명형 이름을 사용합니다.

**참고:** 이름은 대소문자를 구분하지 않습니다. 예를 들어, EMPLOYEES는 eMployees 또는 eMpLOYEES와 동일한 이름으로 간주됩니다. 그러나 따옴표로 묶은 식별자는 대소문자를 구분합니다.

자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 *Schema Object Names and Qualifiers* 관련 섹션을 참조하십시오.

## 단원 내용

- 데이터베이스 객체:
  - 이름 지정 규칙
- CREATE TABLE 문:
  - 다른 유저의 테이블에 액세스
  - DEFAULT 옵션
- 데이터 유형
- 제약 조건 개요: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK 제약 조건
- subquery를 사용하여 테이블 생성
- ALTER TABLE
  - 읽기 전용 테이블
- DROP TABLE 문

ORACLE®

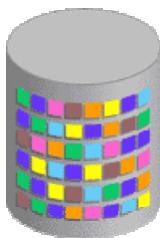
Copyright © 2009, Oracle. All rights reserved.

## CREATE TABLE 문

- 다음이 필요합니다.
  - CREATE TABLE 권한
  - 저장 영역

```
CREATE TABLE [schema.]table  
  (column datatype [DEFAULT expr][, ...]);
```

- 다음을 지정합니다.
  - 테이블 이름
  - 열 이름, 열 데이터 유형 및 열 크기



ORACLE

Copyright © 2009, Oracle. All rights reserved.

### CREATE TABLE 문

SQL CREATE TABLE 문을 실행하여 데이터를 저장할 테이블을 생성합니다. DDL 문 중 하나인 이 명령문은 오라클 데이터베이스 구조를 생성, 수정 또는 제거하는 데 사용되는 SQL 문의 일종입니다. 이러한 명령문은 데이터베이스에 즉시 적용되고 데이터 덕셔너리에 정보를 기록하기도 합니다.

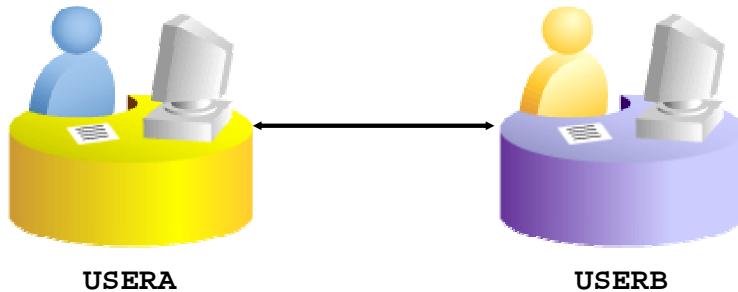
테이블을 생성하려면 유저는 CREATE TABLE 권한이 있어야 하며 객체를 생성할 저장 영역이 있어야 합니다. DBA(데이터베이스 관리자)는 DCL(데이터 제어어) 문을 사용하여 유저에게 권한을 부여합니다.

이 구문에서 다음이 적용됩니다.

schema	소유자 이름과 동일합니다.
table	테이블의 이름입니다.
DEFAULT expr	INSERT 문에서 값이 생략된 경우 기본값을 지정합니다.
column	열 이름입니다.
datatype	열의 데이터 유형 및 길이입니다.

## 다른 유저의 테이블 참조

- 다른 유저가 소유한 테이블은 유저의 스키마에 없습니다.
- 이러한 테이블에는 소유자의 이름을 접두어로 사용해야 합니다.



```
SELECT *
FROM userB.employees;
```

```
SELECT *
FROM userA.employees;
```

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### 다른 유저의 테이블 참조

스키마는 데이터의 논리적 구조 또는 스키마 객체의 모음입니다. 스키마는 데이터베이스 유저가 소유하며 해당 유저와 동일한 이름을 가지고 있습니다. 각 유저는 단일 스키마를 소유합니다.

스키마 객체에는 테이블, 뷰, 동의어, 시퀀스, 내장 프로시저, 인덱스, 클러스터 및 데이터베이스 링크가 있으며 SQL을 사용하여 스키마 객체를 생성 및 조작할 수 있습니다.

테이블이 유저 소유가 아닌 경우 테이블 이름 앞에 소유자의 이름을 추가해야 합니다. 예를 들어, USERA 및 USERB라는 스키마가 있고 양쪽에 모두 EMPLOYEES 테이블이 있는 경우 USERB가 소유한 EMPLOYEES 테이블에 USERA가 액세스하려면 USERA는 다음과 같이 테이블 이름 앞에 스키마 이름을 추가해야 합니다.

```
SELECT *
FROM userb.employees;
```

USERA가 소유한 EMPLOYEES 테이블에 USERB가 액세스하려면 USERB는 다음과 같이 테이블 이름 앞에 스키마 이름을 추가해야 합니다.

```
SELECT *
FROM usera.employees;
```

## DEFAULT 옵션

- 삽입 시 열의 기본값을 지정합니다.

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- 리터럴 값, 표현식 또는 SQL 함수는 올바른 값입니다.
- 다른 열의 이름이나 의사 열은 잘못된 값입니다.
- 기본 데이터 유형은 열 데이터 유형과 일치해야 합니다.

```
CREATE TABLE hire_dates
  (id          NUMBER(8),
   hire_date DATE DEFAULT SYSDATE);
```

```
CREATE TABLE succeeded.
```

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### DEFAULT 옵션

테이블을 정의할 때 열에 기본값이 제공되도록 DEFAULT 옵션을 사용하여 지정할 수 있습니다. 이 옵션은 열에 대한 값이 없는 행이 삽입되는 경우 해당 열에 null 값이 입력되는 것을 방지합니다. 기본값으로 리터럴, 표현식 또는 SYSDATE 및 USER와 같은 SQL 함수를 사용할 수 있지만 다른 열의 이름이나 NEXTVAL 또는 CURRVAL과 같은 의사 열은 사용할 수 없습니다. 기본 표현식은 열의 데이터 유형과 일치해야 합니다.

다음 예제를 살펴보십시오.

```
INSERT INTO hire_dates values(45, NULL);
```

위의 명령문에서는 기본값 대신 null 값을 삽입합니다.

```
INSERT INTO hire_dates(id) values(35);
```

위의 명령문에서는 HIRE\_DATE 열에 SYSDATE를 삽입합니다.

**참고:** SQL Developer에서 DDL 문을 실행하려면 Run Script 아이콘 또는 [F5]를 누릅니다. 피드백 메시지가 Script Output 탭 페이지에 표시됩니다.

# 테이블 생성

- 테이블 생성:

```
CREATE TABLE dept
  (deptno      NUMBER(2),
   dname       VARCHAR2(14),
   loc         VARCHAR2(13),
   create_date DATE DEFAULT SYSDATE);
```

CREATE TABLE succeeded.

- 테이블 생성 확인:

```
DESCRIBE dept
```

Name	Null	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)
CREATE_DATE		DATE

4 rows selected

ORACLE

Copyright © 2009, Oracle. All rights reserved.

## 테이블 생성

슬라이드의 예제에서는 네 개의 열(DEPTNO, DNAME, LOC, CREATE\_DATE)을 가진 DEPT 테이블을 생성합니다. CREATE\_DATE 열은 기본값을 가지고 있습니다. INSERT 문에 값이 제공되지 않으면 시스템 날짜가 자동으로 삽입됩니다.

테이블이 생성되었는지 확인하려면 DESCRIBE 명령을 실행합니다.

테이블 생성은 DDL 문 작업이기 때문에 이 명령문이 실행되면 자동 커밋이 발생합니다.

**참고:** 데이터 딕셔너리를 query하여 자신이 소유한 테이블 리스트를 조회할 수 있습니다. 예를 들면 다음과 같습니다.

```
select table_name from user_tables
```

데이터 딕셔너리 뷰를 사용하여 뷰, 인덱스 등과 같은 다른 데이터베이스 객체에 대한 정보를 찾아볼 수도 있습니다. 데이터 딕셔너리에 대해서는 *Oracle Database 11g: SQL Fundamentals II* 과정에서 자세히 설명합니다.

## 단원 내용

- 데이터베이스 객체:
  - 이름 지정 규칙
- CREATE TABLE 문:
  - 다른 유저의 테이블에 액세스
  - DEFAULT 옵션
- 데이터 유형
- 제약 조건 개요: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK 제약 조건
- subquery를 사용하여 테이블 생성
- ALTER TABLE
  - 읽기 전용 테이블
- DROP TABLE 문

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 데이터 유형

데이터 유형	설명
VARCHAR2( <i>size</i> )	가변 길이 문자 데이터
CHAR( <i>size</i> )	고정 길이 문자 데이터
NUMBER( <i>p,s</i> )	가변 길이 숫자 데이터
DATE	날짜 및 시간 값
LONG	가변 길이 문자 데이터(최대 2GB)
CLOB	문자 데이터(최대 4GB)
RAW and LONG RAW	원시 이진 데이터
BLOB	바이너리 데이터(최대 4GB)
BFILE	외부 파일에 저장된 바이너리 데이터(최대 4GB)
ROWID	테이블에 있는 행의 고유한 주소를 나타내는 base-64 숫자 체계

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 데이터 유형

테이블에 대해 열을 식별할 때 열의 데이터 유형을 제공해야 합니다. 다음과 같은 여러 데이터 유형을 사용할 수 있습니다.

데이터 유형	설명
VARCHAR2( <i>size</i> )	가변 길이 문자 데이터(최대 <i>size</i> 를 지정해야 합니다. 최소 <i>size</i> 는 1이고 최대 <i>size</i> 는 4,000입니다.)
CHAR [( <i>size</i> )]	길이가 <i>size</i> 바이트인 고정 길이 문자 데이터(기본 및 최소 <i>size</i> 는 1이며 최대 <i>size</i> 는 2,000입니다.)
NUMBER [( <i>p,s</i> )]	자릿수가 <i>p</i> 이고 소수점 이하 자릿수가 <i>s</i> 인 숫자(자릿수는 십진 숫자의 총 수이고 소수점 이하 자릿수는 소수점 오른쪽에 있는 숫자의 수입니다. 자릿수는 1~38까지 될 수 있고 소수점 이하 자릿수는 -84~127까지 될 수 있습니다.)
DATE	기원전 4712년 1월 1일부터 서기 9999년 12월 31일 사이의 가장 가까운 초 단위에 대한 날짜 및 시간 값
LONG	가변 길이 문자 데이터(최대 2GB)
CLOB	문자 데이터(최대 4GB)

## 데이터 유형(계속)

데이터 유형	설명
RAW( <i>size</i> )	길이가 <i>size</i> 인 원시 바이너리 데이터(최대 <i>size</i> 를 지정해야 합니다. 최대 <i>size</i> 는 2,000입니다.)
LONG RAW	가변 길이의 원시 바이너리 데이터(최대 2GB)
BLOB	바이너리 데이터(최대 4GB)
BFILE	외부 파일에 저장된 바이너리 데이터(최대 4GB)
ROWID	테이블에 있는 행의 고유한 주소를 나타내는 base-64 숫자 체계

### 지침

- LONG 열은 subquery를 사용하여 테이블을 생성할 때 복사되지 않습니다.
- LONG 열은 GROUP BY 또는 ORDER BY 절에 포함될 수 없습니다.
- 각 테이블당 하나의 LONG 열만 사용할 수 있습니다.
- LONG 열에 대해 제약 조건을 정의할 수 없습니다.
- LONG 열 대신 CLOB 열을 사용할 수 있습니다.

## Datetime 데이터 유형

다음과 같은 여러 Datetime 데이터 유형을 사용할 수 있습니다.

데이터 유형	설명
TIMESTAMP	소수 표시 초 단위의 날짜
INTERVAL YEAR TO MONTH	년, 월 간격으로 저장됨
INTERVAL DAY TO SECOND	일, 시, 분, 초 간격으로 저장됨



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### Datetime 데이터 유형

데이터 유형	설명
TIMESTAMP	소수 표시 초 단위의 날짜로 시간을 저장할 수 있습니다. 소수 표시 초 값뿐만 아니라 DATE 데이터 유형의 년, 월, 일, 시, 분 및 초를 저장합니다. 이 데이터 유형은 WITH TIMEZONE, WITH LOCALTIMEZONE 등과 같은 여러 변형이 있습니다.
INTERVAL YEAR TO MONTH	년, 월 간격으로 시간을 저장할 수 있습니다. 년과 월로만 유효한 부분을 나타내는 두 datetime 값 사이의 차이를 표시하는 데 사용됩니다.
INTERVAL DAY TO SECOND	일, 시, 분, 초 간격으로 시간을 저장할 수 있습니다. 두 datetime 값 사이의 정밀한 차이를 나타내는 데 사용됩니다.

**참고:** 이러한 Datetime 데이터 유형은 Oracle9i 이상의 버전에서 사용할 수 있습니다. Datetime 데이터 유형은 *Oracle Database 11g: SQL Fundamentals II* 과정의 "다른 시간대에서 데이터 관리" 단원에서 자세히 설명합니다.

또한 Datetime 데이터 유형에 대한 자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*의 "TIMESTAMP Datatype", "INTERVAL YEAR TO MONTH Datatype" 및 "INTERVAL DAY TO SECOND Datatype" 관련 섹션을 참조하십시오.

## 단원 내용

- 데이터베이스 객체:
  - 이름 지정 규칙
- CREATE TABLE 문:
  - 다른 유저의 테이블에 액세스
  - DEFAULT 옵션
- 데이터 유형
- 제약 조건 개요: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK 제약 조건
- subquery를 사용하여 테이블 생성
- ALTER TABLE
  - 읽기 전용 테이블
- DROP TABLE 문

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 제약 조건 포함

- 제약 조건은 테이블 레벨에서 규칙을 강제 적용합니다.
- 제약 조건은 테이블에 종속 관계가 있는 경우 삭제를 방지합니다.
- 유효한 제약 조건 유형은 다음과 같습니다.
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK



**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### 제약 조건

Oracle 서버는 제약 조건을 사용하여 테이블에 잘못된 데이터를 입력하는 것을 방지합니다.

제약 조건을 사용하여 다음을 수행할 수 있습니다.

- 테이블에서 행을 삽입, 갱신 또는 삭제할 때마다 테이블의 데이터에 규칙을 강제 적용합니다. 작업이 성공하려면 제약 조건이 충족되어야 합니다.
- 다른 테이블과 종속 관계가 있는 경우 테이블 삭제를 방지합니다.
- Oracle Developer와 같은 Oracle 도구에 규칙을 제공합니다.

### 데이터 무결성 제약 조건

제약 조건	설명
NOT NULL	열에 null 값을 포함할 수 없음을 지정합니다.
UNIQUE	테이블의 모든 행에 대해 값이 고유해야 하는 열 또는 열 조합을 지정합니다.
PRIMARY KEY	테이블의 각 행을 고유하게 식별합니다.
FOREIGN KEY	특정 테이블의 열과 참조 테이블의 열 간에 참조 무결성을 설정하고 적용하여 한 테이블의 값이 다른 테이블의 값과 일치하도록 합니다.
CHECK	참이어야 하는 조건을 지정합니다.

## 제약 조건 지침

- **유저가 제약 조건의 이름을 지정하거나 Oracle 서버가 SYS\_Cn 형식을 사용하여 이름을 생성할 수 있습니다.**
- **다음 시점 중 하나에서 제약 조건을 생성합니다.**
  - 테이블이 생성되는 시점
  - 테이블 생성 후
- **열 또는 테이블 레벨에서 제약 조건을 정의합니다.**
- **데이터 딕셔너리에서 제약 조건을 확인합니다.**

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 제약 조건 지침

모든 제약 조건은 데이터 딕셔너리에 저장됩니다. 제약 조건에 의미 있는 이름을 제공하면 쉽게 참조할 수 있습니다. 제약 조건 이름은 표준 객체 이름 지정 규칙을 따라야 하며 예외적으로 동일한 유저가 소유한 다른 객체와 이름이 같지 않아야 한다는 규칙은 적용되지 않습니다. 제약 조건의 이름을 지정하지 않으면 Oracle 서버가 SYS\_Cn 형식으로 이름을 생성합니다. 여기서 *n*은 제약 조건 이름에 고유하게 사용되는 정수입니다.

테이블 생성 시 또는 테이블이 생성된 후에 제약 조건을 정의할 수 있습니다. 제약 조건은 열 또는 테이블 레벨에서 정의할 수 있습니다. 기능적인 면에서 테이블 레벨의 제약 조건은 열 레벨의 제약 조건과 동일합니다.

자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 "Constraints" 관련 섹션을 참조하십시오.

# 제약 조건 정의

- 구문:

```
CREATE TABLE [schema.]table
  (column datatype [DEFAULT expr]
   [column_constraint],
   ...
   [table_constraint][,...]);
```

- 열 레벨 제약 조건 구문:

```
column [CONSTRAINT constraint_name] constraint_type,
```

- 테이블 레벨 제약 조건 구문:

```
column, ...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

## 제약 조건 정의

슬라이드는 테이블 생성 시 제약 조건을 정의하는 구문을 제공합니다. 열 레벨이나 테이블 레벨에서 제약 조건을 생성할 수 있습니다. 열 레벨에서 정의되는 제약 조건은 열이 정의될 때 포함됩니다. 테이블 레벨 제약 조건은 테이블 정의의 끝에서 정의되며 제약 조건이 괄호로 묶인 열을 참조해야 합니다. 두 제약 조건은 구문에서 주된 차이점을 보이지만 기능적인 면에서 열 레벨 제약 조건은 테이블 레벨 제약 조건과 동일합니다.

NOT NULL 제약 조건은 열 레벨에서 정의되어야 합니다.

두 개 이상의 열에 적용되는 제약 조건은 테이블 레벨에서 정의되어야 합니다.

이 구문에서 다음이 적용됩니다.

schema	소유자 이름과 동일합니다.
table	테이블의 이름입니다.
DEFAULT expr	INSERT 문에서 값이 생략된 경우 사용할 기본값을 지정합니다.
column	열 이름입니다.
datatype	열의 데이터 유형 및 길이입니다.
column_constraint	열 정의의 일부인 무결성 제약 조건입니다.
table_constraint	테이블 정의의 일부인 무결성 제약 조건입니다.

# 제약 조건 정의

- 열 레벨 제약 조건 예제:

```
CREATE TABLE employees(
    employee_id  NUMBER(6)
        CONSTRAINT emp_emp_id_pk PRIMARY KEY,
    first_name    VARCHAR2(20),
    ...
);
```

1

- 테이블 레벨 제약 조건 예제:

```
CREATE TABLE employees(
    employee_id  NUMBER(6),
    first_name    VARCHAR2(20),
    ...
    job_id        VARCHAR2(10) NOT NULL,
    CONSTRAINT emp_emp_id_pk
        PRIMARY KEY (EMPLOYEE_ID));
```

2

ORACLE

Copyright © 2009, Oracle. All rights reserved.

## 제약 조건 정의(계속)

제약 조건은 대개 테이블과 동시에 생성됩니다. 제약 조건은 테이블 생성 후에 테이블에 추가할 수 있고 일시적으로 비활성화할 수도 있습니다.

슬라이드의 예제에서는 모두 EMPLOYEES 테이블의 EMPLOYEE\_ID 열에 Primary key 제약 조건을 생성합니다.

1. 첫번째 예제는 열 레벨 구문을 사용하여 제약 조건을 정의합니다.
2. 두번째 예제는 테이블 레벨 구문을 사용하여 제약 조건을 정의합니다.

Primary key 제약 조건에 대한 자세한 내용은 이 단원의 뒷부분에서 제공합니다.

## NOT NULL 제약 조건

열에 null 값이 허용되지 않도록 보장합니다.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID	EMAIL	PHONE_NUMBER	HIRE_DATE
100	Steven	King	24000	(null)	90	SKING	515.123.4567	17-JUN-87
101	Neena	Kochhar	17000	(null)	90	NKOCHHAR	515.123.4568	21-SEP-89
102	Lex	De Haan	17000	(null)	90	LDEHAAN	515.123.4569	13-JAN-93
103	Alexander	Hunold	9000	(null)	60	AHUNOLD	590.423.4567	03-JAN-90
104	Bruce	Ernst	6000	(null)	60	BERNST	590.423.4568	21-MAY-91
107	Diana	Lorentz	4200	(null)	60	DLORENTZ	590.423.5567	07-FEB-99
124	Kevin	Mourgos	5800	(null)	50	KMOURGOS	650.123.5234	16-NOV-99
141	Trenna	Rajs	3500	(null)	50	TRAJS	650.121.8009	17-OCT-95
142	Curtis	Davies	3100	(null)	50	CDAVIES	650.121.2994	29-JAN-97
143	Randall	Matos	2600	(null)	50	RMATOS	650.121.2874	15-MAR-98
144	Peter	Vargas	2500	(null)	50	PVARGAS	650.121.2004	09-JUL-98
149	Eleni	Zlotkey	10500	0.2	80	EZLOTKEY	011.44.1344.429018	29-JAN-00
174	Ellen	Abel	11000	0.3	80	EABEL	011.44.1644.429267	11-MAY-96
176	Jonathon	Taylor	8600	0.2	80	JTAYLOR	011.44.1644.429265	24-MAR-98
178	Kimberely	Grant	7000	0.15	(null)	KGRANT	011.44.1644.429263	24-MAY-99
200	Jennifer	Whalen	4400	(null)	10	JWHALEN	515.123.4444	17-SEP-87
201	Michael	Hartstein	13000	(null)	20	MHARTSTE	515.123.5555	17-FEB-96
202	Pat	Fay	6000	(null)	20	PFAY	603.123.6666	17-AUG-97
205	Shelley	Higgins	12000	(null)	110	SHIGGINS	515.123.8080	07-JUN-94
206	William	Gietz	8300	(null)	110	WGIETZ	515.123.8181	07-JUN-94

↑  
NOT NULL 제약 조건  
(Primary Key는 NOT NULL  
제약 조건을 적용합니다.)

↑  
NOT NULL  
제약 조건

↑  
NOT NULL 제약 조건 없음(모든 행에서 이  
열에 대해 null 값을 포함할 수 있습니다.)

ORACLE®

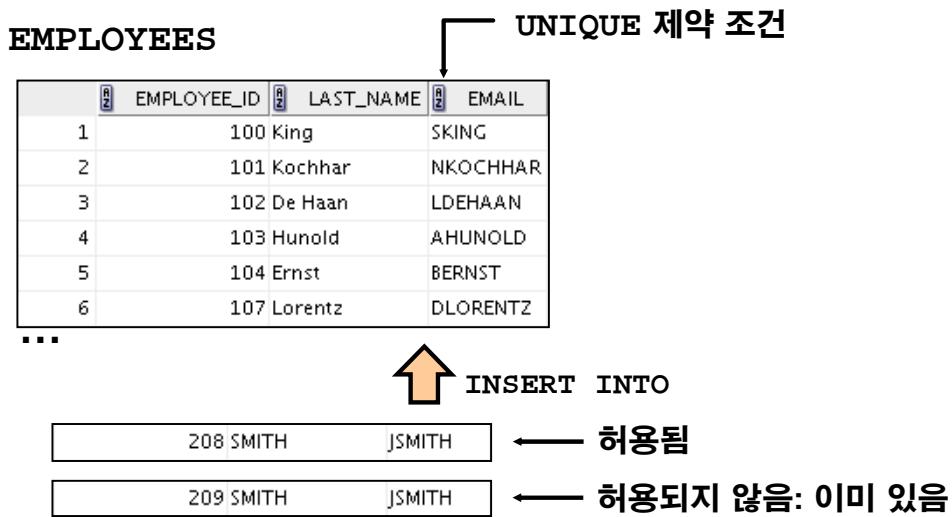
Copyright © 2009, Oracle. All rights reserved.

### NOT NULL 제약 조건

NOT NULL 제약 조건은 열이 null 값을 포함하지 않도록 보장합니다. NOT NULL 제약 조건이 없는 열은 기본적으로 null 값을 포함할 수 있습니다. NOT NULL 제약 조건은 열 레벨에서 정의되어야 합니다. EMPLOYEES 테이블에서 EMPLOYEE\_ID 열은 Primary key로 정의되었으므로 NOT NULL을 상속합니다. 그렇지 않으면 LAST\_NAME, EMAIL, HIRE\_DATE 및 JOB\_ID 열에 NOT NULL 제약 조건이 적용됩니다.

참고: Primary key 제약 조건은 이 단원의 뒷부분에서 자세히 설명합니다.

## UNIQUE 제약 조건



ORACLE

Copyright © 2009, Oracle. All rights reserved.

### UNIQUE 제약 조건

UNIQUE Key 무결성 제약 조건에서는 하나의 열 또는 여러 열에 있는 모든 값(키)이 고유해야 합니다. 즉, 테이블의 두 행은 지정된 열 또는 열 집합에서 중복된 값을 가질 수 없습니다. UNIQUE Key 제약 조건 정의에 포함된 열 또는 열 집합을 *Unique key*라고 합니다. UNIQUE 제약 조건은 두 개 이상의 열로 구성되며 이러한 열 그룹을 조합 Unique Key라고 합니다.

UNIQUE 제약 조건에서는 동일한 열에 대해 NOT NULL 제약 조건을 정의하지 않는 한, null을 입력할 수 있습니다. 실제로 null은 어떤 값과도 같지 않기 때문에 모든 행에서 NOT NULL 제약 조건이 없는 열에 대해 null을 포함할 수 있습니다. 열 또는 조합 UNIQUE Key의 모든 열에서 null은 항상 UNIQUE 제약 조건을 충족합니다.

**참고:** 두 개 이상의 열에 대한 UNIQUE 제약 조건의 검색 메커니즘 때문에 부분적으로 null인 조합 UNIQUE Key 제약 조건의 null이 아닌 열에서 동일한 값을 가질 수 없습니다.

## UNIQUE 제약 조건

테이블 레벨 또는 열 레벨에서 정의됩니다.

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
    ...
    CONSTRAINT emp_email_uk UNIQUE(email));
```

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

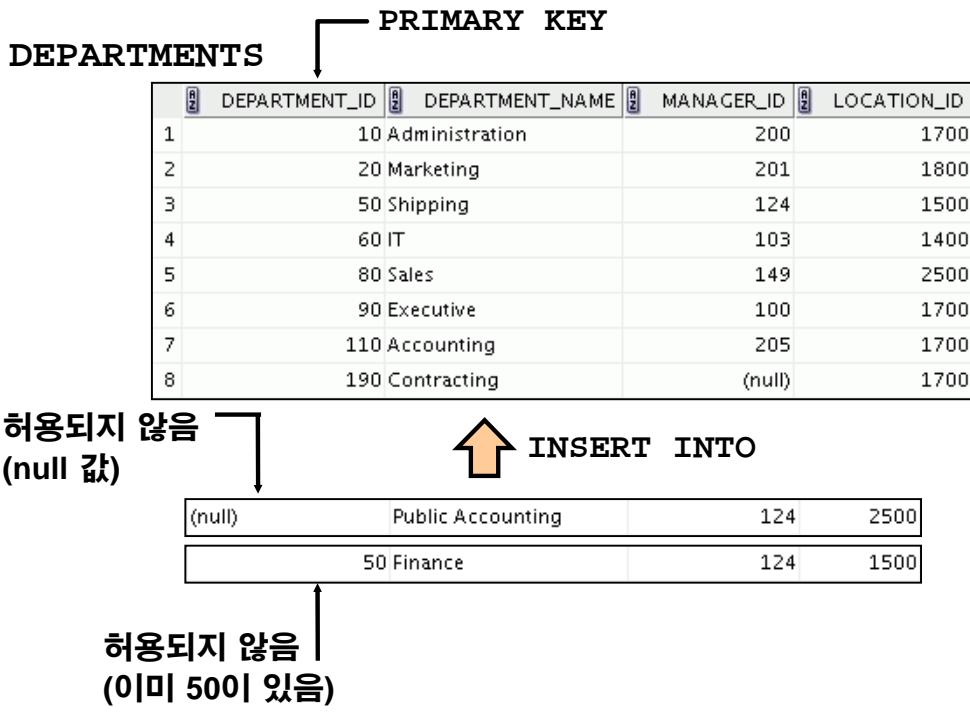
### UNIQUE 제약 조건(계속)

UNIQUE 제약 조건은 열 레벨 또는 테이블 레벨에서 정의될 수 있습니다. 조합 Unique key를 생성하려는 경우 테이블 레벨에서 제약 조건을 정의합니다. 행을 고유하게 식별할 수 있는 속성이 하나도 없는 경우 조합 키가 정의됩니다. 이 경우 두 개 이상의 열로 구성된, 항상 고유하고 행을 식별할 수 있는 값의 결합인 Unique key를 사용할 수 있습니다.

슬라이드의 예제는 EMPLOYEES 테이블의 EMAIL 열에 UNIQUE 제약 조건을 적용합니다. 제약 조건의 이름은 EMP\_EMAIL\_UK입니다.

**참고:** Oracle 서버는 Unique Key 열에서 고유 인덱스를 암시적으로 생성하는 방식으로 UNIQUE 제약 조건을 적용합니다.

## PRIMARY KEY 제약 조건



ORACLE

Copyright © 2009, Oracle. All rights reserved.

### PRIMARY KEY 제약 조건

PRIMARY KEY 제약 조건은 테이블에 대해 Primary key를 생성합니다. 각 테이블에는 하나의 Primary key만 생성할 수 있습니다. PRIMARY KEY 제약 조건은 테이블의 각 행을 고유하게 식별하는 열 또는 열 집합입니다. 이 제약 조건은 열 또는 열 조합에 고유성을 적용하고 Primary key에 속하는 열이 null 값을 포함할 수 없도록 합니다.

**참고:** 고유성은 Primary Key 제약 조건 정의의 일부이기 때문에 Oracle 서버는 암시적으로 Primary Key 열에 고유 인덱스를 생성하는 방식으로 고유성을 적용합니다.

## FOREIGN KEY 제약 조건

### DEPARTMENTS

**PRIMARY KEY** 

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500

... 

### EMPLOYEES

**FOREIGN KEY** 

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
4	205	Higgins	110
5	206	Gietz	110

... 

 INSERT INTO

200 Ford	9
200 Ford	60

 허용되지  
 않음(9가 없음)  
 허용됨

ORACLE

Copyright © 2009, Oracle. All rights reserved.

### FOREIGN KEY 제약 조건

FOREIGN KEY(또는 참조 무결성) 제약 조건은 열 또는 열 조합을 Foreign key로 지정하고 동일한 테이블이나 다른 테이블의 Primary key 또는 Unique key와의 관계를 설정합니다.

슬라이드의 예제에서 DEPARTMENT\_ID는 EMPLOYEES 테이블(중속 또는 하위 테이블)의 Foreign key로 정의되었습니다. 이 테이블은 DEPARTMENTS 테이블(참조 또는 상위 테이블)의 DEPARTMENT\_ID 열을 참조합니다.

#### 자침

- Foreign key값은 상위 테이블의 기존 값과 일치하거나 NULL이어야 합니다.
- Foreign key는 데이터 값을 기반으로 하며 물리적 포인터가 아니라 순전히 논리적 포인터입니다.

## FOREIGN KEY 제약 조건

테이블 레벨 또는 열 레벨에서 정의됩니다.

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
    ...
    department_id    NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
        REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

### FOREIGN KEY 제약 조건(계속)

FOREIGN KEY 제약 조건은 열 또는 테이블 제약 조건 레벨에서 정의될 수 있습니다.

조합 Foreign key는 테이블 레벨 정의를 사용하여 생성되어야 합니다.

슬라이드의 예제는 테이블 레벨 구문을 사용하여 EMPLOYEES 테이블의 DEPARTMENT\_ID 열에 FOREIGN KEY 제약 조건을 정의합니다. 제약 조건의 이름은 EMP\_DEPT\_FK입니다.

제약 조건이 단일 열을 기반으로 하는 경우 Foreign key는 열 레벨에서도 정의될 수 있습니다. 이 구문은 FOREIGN KEY 키워드가 나타나지 않는다는 점에서 차이가 있습니다. 예를 들면 다음과 같습니다.

```
CREATE TABLE employees
(
    ...
    department_id NUMBER(4) CONSTRAINT emp_deptid_fk
        REFERENCES departments(department_id),
    ...
)
```

## FOREIGN KEY 제약 조건: 키워드

- FOREIGN KEY: 테이블 제약 조건 레벨에서 하위 테이블의 열을 정의합니다.
- REFERENCES: 테이블 및 상위 테이블의 열을 식별합니다.
- ON DELETE CASCADE: 상위 테이블의 행이 삭제될 때 하위 테이블의 종속 행을 삭제합니다.
- ON DELETE SET NULL: 종속 Foreign key 값을 null로 변환합니다.

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### FOREIGN KEY 제약 조건: 키워드

Foreign Key는 하위 테이블에서 정의되며 참조되는 열을 포함하는 테이블은 상위 테이블입니다. Foreign Key는 다음 키워드 조합을 사용하여 정의됩니다.

- FOREIGN KEY는 테이블 제약 조건 레벨에서 하위 테이블의 열을 정의하는 데 사용됩니다.
- REFERENCES는 테이블 및 상위 테이블의 열을 식별합니다.
- ON DELETE CASCADE는 상위 테이블의 행이 삭제될 때 하위 테이블의 종속 행도 삭제됨을 나타냅니다.
- ON DELETE SET NULL은 상위 테이블의 행이 삭제될 때 Foreign key 값이 null로 설정됨을 나타냅니다.

기본 동작을 *Restrict Rule*이라고 하며 참조되는 데이터의 생성이나 삭제를 허용하지 않습니다.

ON DELETE CASCADE 또는 ON DELETE SET NULL 옵션이 없으면 상위 테이블의 행을 하위 테이블에서 참조하는 경우 삭제할 수 없습니다.

## CHECK 제약 조건

- 각 행이 충족해야 하는 조건을 정의합니다.
- 다음 표현식은 허용되지 않습니다.
  - CURRVAL, NEXTVAL, LEVEL, ROWNUM 의사 열에 대한 참조
  - SYSDATE, UID, USER, USERENV 함수에 대한 호출
  - 다른 행의 다른 값을 참조하는 query

```
...., salary NUMBER(2)
CONSTRAINT emp_salary_min
    CHECK (salary > 0),...
```

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### CHECK 제약 조건

CHECK 제약 조건은 각 행이 충족해야 하는 조건을 정의합니다. 이 조건은 다음의 경우를 제외하고 query 조건과 동일한 구성을 사용할 수 있습니다.

- CURRVAL, NEXTVAL, LEVEL, ROWNUM 의사 열에 대한 참조
- SYSDATE, UID, USER, USERENV 함수에 대한 호출
- 다른 행의 다른 값을 참조하는 query

단일 열에서 해당 정의의 열을 참조하는 여러 CHECK 제약 조건을 가질 수 있습니다. 열에 정의할 수 있는 CHECK 제약 조건의 수에는 제한이 없습니다.

CHECK 제약 조건은 열 레벨이나 테이블 레벨에서 정의될 수 있습니다.

```
CREATE TABLE employees
(
    ...
    salary NUMBER(8,2) CONSTRAINT emp_salary_min
        CHECK (salary > 0),
    ...
)
```

## CREATE TABLE: 예제

```
CREATE TABLE employees
  ( employee_id      NUMBER(6)
    CONSTRAINT emp_employee_id PRIMARY KEY
  , first_name        VARCHAR2(20)
  , last_name         VARCHAR2(25)
    CONSTRAINT emp_last_name_nn NOT NULL
  , email             VARCHAR2(25)
    CONSTRAINT emp_email_nn     NOT NULL
    CONSTRAINT emp_email_uk     UNIQUE
  , phone_number      VARCHAR2(20)
  , hire_date         DATE
    CONSTRAINT emp_hire_date_nn NOT NULL
  , job_id            VARCHAR2(10)
    CONSTRAINT emp_job_nn      NOT NULL
  , salary             NUMBER(8,2)
    CONSTRAINT emp_salary_ck    CHECK (salary>0)
  , commission_pct    NUMBER(2,2)
  , manager_id        NUMBER(6)
    CONSTRAINT emp_manager_fk REFERENCES
      employees (employee_id)
  , department_id     NUMBER(4)
    CONSTRAINT emp_dept_fk      REFERENCES
      departments (department_id));
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

### CREATE TABLE: 예제

슬라이드의 예제에서는 HR 스키마에서 EMPLOYEES 테이블을 생성하는 데 사용되는 명령문을 보여줍니다.

## 제약 조건 위반

```
UPDATE employees
SET department_id = 55
WHERE department_id = 110;
```

```
Error starting at line 1 in command:
UPDATE employees
SET department_id = 55
WHERE department_id = 110
Error report:
SQL Error: ORA-02291: integrity constraint (ORA1.EMP_DEPT_FK) violated - parent key not found
02291. 00000 - "integrity constraint (%s.%s) violated - parent key not found"
*Cause:    A foreign key value has no matching primary key value.
```

부서 55가 존재하지 않습니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 제약 조건 위반

열에 제약 조건이 있는 경우 제약 조건 규칙을 위반하려고 하면 오류가 반환됩니다. 예를 들어, 무결성 제약 조건이 적용된 값이 있는 레코드를 생성하려고 하면 오류가 반환됩니다.

슬라이드의 예제에서 상위 테이블 DEPARTMENTS에 부서 55가 없기 때문에 "parent key not found"라는 위반 오류 ORA-02291이 발생합니다.

## 제약 조건 위반

**다른 테이블에서 Foreign key로 사용되는 Primary key를 포함한 행은 삭제할 수 없습니다.**

```
DELETE FROM departments
WHERE department_id = 60;
```

```
Error starting at line 1 in command:
DELETE FROM departments
WHERE department_id = 60
Error report:
SQL Error: ORA-02292: integrity constraint (ORA1.JHIST_DEPT_FK) violated - child record found
02292. 00000 - "integrity constraint (%s.%s) violated - child record found"
*Cause:    attempted to delete a parent key value that had a foreign
           dependency.
*Action:   delete dependencies first then parent or disable constraint.
```

Copyright © 2009, Oracle. All rights reserved.

### 제약 조건 위반(계속)

무결성 제약 조건이 적용된 값이 있는 레코드를 삭제하려고 하면 오류가 반환됩니다. 슬라이드의 예제는 DEPARTMENTS 테이블에서 부서 60을 삭제하려고 하지만 해당 부서 번호가 EMPLOYEES 테이블에서 Foreign key로 사용되기 때문에 오류가 발생합니다. 삭제하려고 하는 상위 레코드가 하위 레코드를 가지고 있는 경우 "child record found"라는 위반 오류 ORA-02292가 발생합니다.

다음 명령문은 부서 70에 사원이 없기 때문에 제대로 실행됩니다.

```
DELETE FROM departments
WHERE department_id = 70;
```

1 rows deleted

## 단원 내용

- 데이터베이스 객체:
  - 이름 지정 규칙
- CREATE TABLE 문:
  - 다른 유저의 테이블에 액세스
  - DEFAULT 옵션
- 데이터 유형
- 제약 조건 개요: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK 제약 조건
- **subquery를 사용하여 테이블 생성**
- ALTER TABLE
  - 읽기 전용 테이블
- DROP TABLE 문

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# Subquery를 사용하여 테이블 생성

- CREATE TABLE 문과 AS subquery 옵션을 결합하여 테이블을 생성하고 행을 삽입합니다.

```
CREATE TABLE table
    [(column, column...)]
AS subquery;
```

- 지정된 열 개수와 subquery 열 개수를 일치시킵니다.
- 열 이름과 기본값을 가진 열을 정의합니다.

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

## Subquery를 사용하여 테이블 생성

테이블을 생성하는 두번째 방법은 AS subquery 절을 적용하는 것입니다. 이 절은 테이블을 생성하고 subquery에서 반환된 행을 삽입합니다.

이 구문에서 다음이 적용됩니다.

table 테이블의 이름입니다.

column 열 이름, 기본값, 무결성 제약 조건입니다.

subquery 새 테이블에 삽입될 행 집합을 정의하는 SELECT 문입니다.

### 지침

- 지정된 열 이름으로 테이블이 생성되고 SELECT 문으로 검색한 행이 테이블에 삽입됩니다.
- 열 정의에는 열 이름과 기본값만 포함할 수 있습니다.
- 열 사양이 주어진 경우 열 개수는 subquery SELECT 리스트의 열 개수와 같아야 합니다.
- 열 사양이 주어지지 않은 경우 테이블의 열 이름은 subquery의 열 이름과 동일합니다.
- 열 데이터 유형 정의 및 NOT NULL 제약 조건이 새 테이블로 전달됩니다. 명시적 NOT NULL 제약 조건만 상속됩니다. PRIMARY KEY 열은 NOT NULL 기능을 새 열에 전달하지 않습니다. 다른 제약 조건 규칙은 새 테이블로 전달되지 않습니다. 하지만 열 정의에 제약 조건을 추가할 수 있습니다.

## Subquery를 사용하여 테이블 생성

```
CREATE TABLE dept80
AS
SELECT employee_id, last_name,
       salary*12 ANNSAL,
       hire_date
  FROM employees
 WHERE department_id = 80;
CREATE TABLE succeeded.
```

```
DESCRIBE dept80
```

Name	Null	Type
EMPLOYEE_ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
ANNSAL		NUMBER
HIRE_DATE	NOT NULL	DATE

ORACLE

Copyright © 2009, Oracle. All rights reserved.

### Subquery를 사용하여 테이블 생성(계속)

슬라이드의 예제에서는 부서 80에서 근무하는 모든 사원의 세부 정보가 담긴 DEPT80이라는 테이블을 생성합니다. DEPT80 테이블의 데이터는 EMPLOYEES 테이블에서 가져옵니다.

DESCRIBE 명령을 사용하여 데이터베이스 테이블이 있는지 확인하고 열 정의를 검사할 수 있습니다.

그러나 표현식을 선택할 때 열 alias를 제공해야 합니다. SALARY\*12 표현식에는 alias ANNSAL이 제공됩니다. alias가 없으면 다음과 같은 오류가 발생합니다.

```
Error starting at line 1 in command:
CREATE TABLE dept80
AS      SELECT employee_id, last_name,
           salary*12 ,
           hire_date FROM employees WHERE department_id = 80
Error at Command Line:3 Column:18
Error report:
SQL Error: ORA-00998: must name this expression with a column alias
00998. 00000 - "must name this expression with a column alias"
*Cause:
*Action:
```

## 단원 내용

- 데이터베이스 객체:
  - 이름 지정 규칙
- CREATE TABLE 문:
  - 다른 유저의 테이블에 액세스
  - DEFAULT 옵션
- 데이터 유형
- 제약 조건 개요: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK 제약 조건
- subquery를 사용하여 테이블 생성
- ALTER TABLE
  - 읽기 전용 테이블
- DROP TABLE 문

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## **ALTER TABLE 문**

**ALTER TABLE 문을 사용하여 다음을 수행합니다.**

- 새 열 추가
- 기존 열 정의 수정
- 새 열의 기본값 정의
- 열 삭제
- 열 이름 바꾸기
- 읽기 전용 상태로 테이블 변경

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### **ALTER TABLE 문**

테이블을 생성한 후에 다음과 같은 이유로 인해 테이블 구조를 변경해야 할 수 있습니다.

- 열을 생략했습니다.
- 열 정의 또는 이름을 변경해야 합니다.
- 열을 제거하려고 합니다.
- 테이블을 읽기 전용 모드로 변경하려 합니다.

이 작업은 ALTER TABLE 문을 사용하여 수행할 수 있습니다.

## 읽기 전용 테이블

**ALTER TABLE** 구문을 사용하여 다음을 수행할 수 있습니다.

- 테이블을 읽기 전용 모드로 설정하여 테이블을 유지 관리하는 동안 DDL 문 또는 DML 문에 의한 변경을 방지합니다.
- 테이블을 다시 읽기/쓰기 모드로 설정합니다.

```
ALTER TABLE employees READ ONLY;

-- perform table maintenance and then
-- return table back to read/write mode

ALTER TABLE employees READ WRITE;
```

Copyright © 2009, Oracle. All rights reserved.

### 읽기 전용 테이블

Oracle Database 11g에서는 READ ONLY를 지정하여 테이블을 읽기 전용 모드로 둘 수 있습니다. 테이블이 READ-ONLY 모드이면 테이블에 영향을 주는 DML 문 또는 SELECT ...FOR UPDATE 문을 실행할 수 없습니다. 테이블의 데이터를 수정하지 않는 한 DDL 문은 실행할 수 있습니다. 테이블과 연결된 인덱스에 대한 작업은 테이블이 READ ONLY 모드일 때 가능합니다.

읽기 전용 테이블을 읽기/쓰기 모드로 되돌리려면 READ/WRITE를 지정합니다.

**참고:** READ ONLY 모드인 테이블을 삭제할 수 있습니다. DROP 명령은 데이터 딕셔너리에서만 실행되므로 테이블 내용에 액세스할 필요가 없습니다. 테이블에서 사용한 공간은 테이블스페이스를 읽기/쓰기로 다시 변경하기 전에는 회수되지 않으며 그 이후 블록 세그먼트 헤더 등에 필요한 변경 사항을 적용할 수 있습니다.

ALTER TABLE 문에 대한 자세한 내용은 *Oracle Database 10g SQL Fundamentals II* 과정을 참조하십시오.

## 단원 내용

- 데이터베이스 객체:
  - 이름 지정 규칙
- CREATE TABLE 문:
  - 다른 유저의 테이블에 액세스
  - DEFAULT 옵션
- 데이터 유형
- 제약 조건 개요: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK 제약 조건
- Subquery를 사용하여 테이블 생성
- ALTER TABLE
  - 읽기 전용 테이블
- DROP TABLE 문

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 테이블 삭제

- 테이블을 Recycle bin으로 이동
- PURGE 절이 지정되면 테이블 및 해당 데이터를 완전히 제거
- 종속 객체 무효화 및 테이블의 객체 권한 제거

```
DROP TABLE dept80;
```

`DROP TABLE dept80 succeeded.`

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 테이블 삭제

DROP TABLE 문은 테이블을 Recycle bin으로 이동하거나 테이블 및 해당 데이터를 데이터베이스에서 완전히 제거합니다. PURGE 절을 지정하지 않으면 DROP TABLE 문에 의해 공간이 테이블스페이스로 회수되지 않으므로 해당 공간을 다른 객체에서 사용하지 못하며 해당 공간은 유저의 공간 할당량 계산에 계속 포함됩니다. 테이블을 삭제하면 종속 객체가 무효화되며 테이블의 객체 권한이 제거됩니다.

테이블을 삭제하면 데이터베이스에서 테이블의 모든 데이터 및 해당 테이블과 연관된 모든 인덱스를 잃게 됩니다.

### 구문

`DROP TABLE table [ PURGE ]`

이 구문에서 `table`은 테이블 이름입니다.

### 지침

- 테이블에서 모든 데이터가 삭제됩니다.
- 뷰와 동의어는 그대로 남아있지만 더 이상 유효하지 않습니다.
- 보류 중인 모든 트랜잭션이 커밋됩니다.
- 테이블 생성자나 DROP ANY TABLE 권한을 가진 유저만 테이블을 제거할 수 있습니다.

**참고:** 삭제된 테이블을 Recycle bin에서 복원하려면 FLASHBACK TABLE 문을 사용합니다.

이에 대해서는 *Oracle Database 11g: SQL Fundamentals II* 과정에서 자세히 설명합니다.

## 퀴즈

제약 조건을 사용하여 다음을 수행할 수 있습니다.

1. 행을 삽입, 갱신 또는 삭제할 때마다 테이블의 데이터에 규칙을 강제 적용합니다.
2. 테이블이 삭제되지 않도록 방지합니다.
3. 테이블이 생성되지 않도록 방지합니다.
4. 테이블에 데이터가 생성되지 않도록 방지합니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

정답: 1, 2, 4

## 요약

이 단원에서는 CREATE TABLE 문을 사용하여 테이블을 생성하고 제약 조건을 포함시키는 방법을 배웠습니다.

- 기본 데이터베이스 객체 분류
- 테이블 구조 검토
- 열에 사용할 수 있는 데이터 유형 나열
- 간단한 테이블 생성
- 테이블 생성 시 제약 조건이 생성되는 방식 설명
- 스키마 객체의 작동 방식 설명

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 요약

이 단원에서는 다음 항목에 대해 설명했습니다.

### CREATE TABLE

- CREATE TABLE 문을 사용하여 테이블을 생성하고 제약 조건을 포함시킵니다.
- subquery를 사용하여 다른 테이블을 기반으로 테이블을 생성합니다.

### DROP TABLE

- 행과 테이블 구조를 제거합니다.
- 이 명령문이 실행되면 롤백할 수 없습니다.

## 연습 10: 개요

이 연습에서는 다음 내용을 다룹니다.

- 새 테이블 생성
- CREATE TABLE AS 구문을 사용하여 새 테이블 생성
- 테이블이 있는지 확인
- 읽기 전용 상태로 테이블 설정
- 테이블 삭제

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 연습 10: 개요

CREATE TABLE 문을 사용하여 새 테이블을 생성합니다. 데이터베이스에 새 테이블이 추가되었는지 확인합니다. 또한 테이블 상태를 READ ONLY로 설정한 다음 READ/WRITE로 복귀시키는 방법을 배웁니다.

**참고:** 모든 DDL 및 DML 문의 경우 SQL Developer에서 query를 실행하려면 Run Script 아이콘 또는 [F5]를 누릅니다. 이렇게 하면 Script Output 탭 페이지에서 피드백 메시지를 볼 수 있습니다. SELECT query의 경우 계속해서 Execute Statement 아이콘을 누르거나 [F9]를 누르면 Results 탭 페이지에서 형식이 지정된 출력을 볼 수 있습니다.



# 기타 스키마 객체 생성

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 단순 뷰 및 복합 뷰 생성
- 뷰에서 데이터 검색
- 시퀀스 생성, 유지 관리 및 사용
- 인덱스 생성 및 유지 관리
- 전용(private) 및 공용(public) 동의어 생성

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 목표

이 단원에서는 뷰, 시퀀스, 동의어 및 인덱스 객체를 소개합니다. 뷰, 시퀀스 및 인덱스를 생성하고 사용하는 기본적인 방법을 배웁니다.

# 단원 내용

- 뷰 개요:
  - 뷰에서 데이터 생성, 수정 및 검색
  - 뷰에서 DML(데이터 조작어) 작업
  - 뷰 삭제
- 시퀀스 개요:
  - 시퀀스 생성, 사용 및 수정
  - 시퀀스 값을 캐시에 저장
  - NEXTVAL 및 CURRVAL Pseudocolumn
- 인덱스 개요
  - 인덱스 생성, 삭제
- 동의어 개요
  - 동의어 생성, 삭제

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 데이터베이스 객체

객체	설명
테이블	기본 저장 단위이며 행으로 구성되어 있습니다.
뷰	하나 이상의 테이블에 있는 데이터의 부분 집합을 논리적으로 나타냅니다.
시퀀스	숫자 값을 생성합니다.
인덱스	데이터 검색 query의 성능을 향상시킵니다.
동의어	객체에 다른 이름을 부여합니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 데이터베이스 객체

데이터베이스에는 테이블 외에도 기타 여러 객체가 있습니다.

뷰를 사용하면 테이블의 데이터를 표시하거나 숨길 수 있습니다.

많은 응용 프로그램에서는 Primary Key 값으로 고유 번호를 사용해야 합니다. 응용 프로그램에 이러한 요구 사항을 처리하는 코드를 작성하거나 시퀀스를 사용하여 고유 번호를 생성할 수 있습니다.

데이터 검색 query의 성능을 향상시키려면 인덱스를 생성하는 것을 고려해 보십시오. 인덱스를 사용하여 열 또는 열 모음에서 고유성을 강화할 수 있습니다.

동의어를 사용하여 객체에 다른 이름을 지정할 수 있습니다.

# 뷰란?

## EMPLOYEES 테이블

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000
104	Bruce	Ernst	BERNSTEIN	515.123.4568	20-APR-98	IT_PROG	6000
105	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	SA_REP	4200
106	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	6900
107	Mark	Tucker	MTUCKER	515.123.8080	17-MAY-99	SA_MAN	5800
108	James	Frederickson	JFREDERICKSON	515.123.8080	17-SEP-87	SA_ASST	3500
109	David	Martini	DMARTINI	515.123.8080	17-FEB-96	MK_MAN	3100
110	Patricia	Chen	PACHEN	515.123.8080	17-AUG-97	MK_REP	2600
111	John	Elis	JELIS	515.123.8080	07-JUN-94	AC_MGR	2500
112	Michael	Schober	MSCHOBER	515.123.8080	07-JUN-94	AC_ACCOUNT	10500
113	Matthew	Austin	MAUSTIN	515.123.8080	07-JUN-94	AC_ACCOUNT	11000
114	Robert	Frederickson	RFREDERICKSON	515.123.8080	07-JUN-94	AC_ACCOUNT	8600
115	Jeffrey	Whitney	JWHITNEY	515.123.8080	07-JUN-94	AC_ACCOUNT	7000
116	Michael	Ernst	MERNSTEIN	515.123.8080	07-JUN-94	AC_ACCOUNT	4400
117	Patricia	Chen	PACHEN	515.123.8080	07-JUN-94	AC_ACCOUNT	13000
118	John	Elis	JELIS	515.123.8080	07-JUN-94	AC_ACCOUNT	12000
119	Michael	Schober	MSCHOBER	515.123.8080	07-JUN-94	AC_ACCOUNT	8300

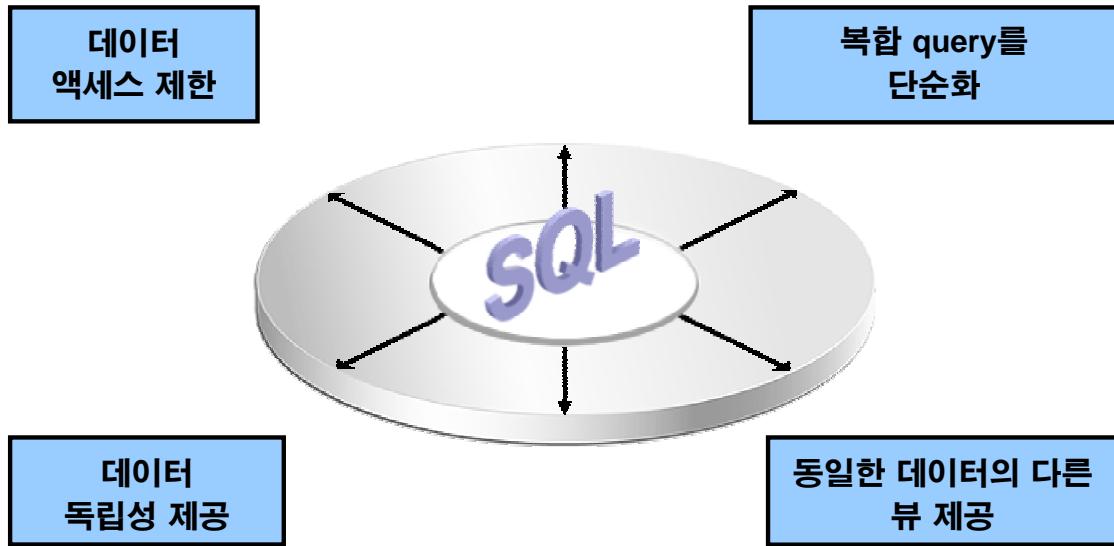
ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 뷰란?

테이블의 뷰를 생성하여 데이터의 논리적 하위 집합 또는 조합을 나타낼 수 있습니다. 뷰는 테이블 또는 다른 뷰를 기반으로 하는 논리적 테이블입니다. 뷰는 자체적으로 데이터를 갖고 있지 않지만 테이블의 데이터를 보거나 변경할 수 있는 window와 같습니다. 뷰의 기반이 되는 테이블을 기본 테이블이라고 합니다. 뷰는 데이터 딕셔너리에 SELECT 문으로 저장됩니다.

## 뷰의 이점



**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### 뷰의 이점

- 뷰는 테이블의 열을 선택적으로 표시하므로 데이터 액세스를 제한합니다.
- 뷰를 사용하여 복잡한 query 결과를 검색하는 간단한 query를 만들 수 있습니다. 예를 들어, 조인 문을 작성하는 방법을 몰라도 뷰를 사용하여 여러 테이블에서 정보를 query할 수 있습니다.
- 뷰는 특정 유저와 응용 프로그램에 대해 데이터 독립성을 제공합니다. 하나의 뷰를 사용하여 여러 테이블의 데이터를 검색할 수 있습니다.
- 뷰는 특정 기준에 따라 유저 그룹에게 데이터 액세스 권한을 부여합니다.

자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 "CREATE VIEW" 관련 섹션을 참조하십시오.

## 단순 뷰와 복합 뷰

기능	단순 뷰	복합 뷰
테이블 수	한 개	하나 이상
함수 포함	아니오	예
데이터 그룹 포함	아니오	예
뷰를 통해 DML 작업	예	항상은 아님

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 단순 뷰와 복합 뷰

뷰에는 단순 뷰와 복합 뷰의 두 가지 유형이 있습니다. 기본적인 차이점은 DML 작업(INSERT, UPDATE 및 DELETE)과 관련이 있습니다.

- 단순 뷰의 특징은 다음과 같습니다.
  - 하나의 테이블에서만 데이터를 가져옵니다.
  - 함수나 데이터 그룹을 포함하지 않습니다.
  - 뷰를 통해 DML 작업을 수행할 수 있습니다.
- 복합 뷰의 특징은 다음과 같습니다.
  - 여러 테이블에서 데이터를 가져옵니다.
  - 함수나 데이터 그룹을 포함합니다.
  - 뷰를 통한 DML 작업을 허용하지 않는 경우도 있습니다.

# 뷰 생성

- CREATE VIEW 문에 subquery를 포함시킵니다.

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
[(alias[, alias]...)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

- 0| subquery에 복합 SELECT 문을 포함할 수 있습니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 뷰 생성

CREATE VIEW 문에 subquery를 포함하여 뷰를 생성할 수 있습니다.

이 구문에서 다음이 적용됩니다.

OR REPLACE	뷰가 이미 있는 경우 다시 생성합니다.
FORCE	기본 테이블의 존재 여부에 관계없이 뷰를 생성합니다.
NOFORCE	기본 테이블이 있는 경우에만 뷰를 생성합니다(기본값).
<i>view</i>	뷰의 이름입니다.
<i>alias</i>	뷰의 query에서 선택한 표현식의 이름을 지정합니다. (alias의 수와 뷰에서 선택한 표현식의 수가 일치해야 합니다.)
<i>subquery</i>	완전한 SELECT 문입니다. (SELECT 리스트에 열의 alias를 사용할 수 있습니다.)
WITH CHECK OPTION	뷰에서 액세스할 수 있는 행만 삽입하거나 갱신할 수 있도록 지정합니다.
<i>constraint</i>	CHECK OPTION 제약 조건에 할당되는 이름입니다.
WITH READ ONLY	현재 뷰에서 DML 작업을 수행하지 못하도록 합니다.

참고: SQL Developer에서 Run Script 아이콘을 누르거나 [F5]를 눌러 DDL(데이터 정의어) 문을 실행합니다. 피드백 메시지가 Script Output 탭 페이지에 표시됩니다.

# 뷰 생성

- 부서 80의 사원에 대한 세부 정보를 포함하는 EMPVU80 뷰를 생성합니다.

```
CREATE VIEW empvu80
AS SELECT employee_id, last_name, salary
FROM employees
WHERE department_id = 80;
```

CREATE VIEW succeeded.

- SQL\*Plus DESCRIBE 명령을 사용하여 뷰의 구조를 설명합니다.

```
DESCRIBE empvu80
```

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 뷰 생성(계속)

슬라이드의 예제는 부서 80의 각 사원에 대한 사원 번호, 성 및 급여를 포함하는 뷰를 생성합니다.  
DESCRIBE 명령을 사용하여 뷰의 구조를 표시할 수 있습니다.

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
SALARY		NUMBER(8,2)

## 지침

- 뷰를 정의하는 subquery는 조인, 그룹 및 subquery를 포함한 복합 SELECT 구문을 포함할 수 있습니다.
- WITH CHECK OPTION을 사용하여 생성한 뷰에 대해 제약 조건 이름을 지정하지 않으면 시스템이 SYS\_Cn 형식으로 기본 이름을 할당합니다.
- OR REPLACE 옵션을 사용하면 뷰를 삭제 및 재생성하거나 이전에 부여한 객체 권한을 다시 부여하지 않고서도 뷰의 정의를 변경할 수 있습니다.

# 뷰 생성

- **subquery에서 열 alias를 사용하여 뷰를 생성합니다.**

```
CREATE VIEW salvu50
AS SELECT employee_id ID_NUMBER, last_name NAME,
           salary*12 ANN_SALARY
      FROM employees
     WHERE department_id = 50;
CREATE VIEW succeeded.
```

- **제공된 alias 이름으로 이 뷰에서 열을 선택합니다.**

ORACLE

Copyright © 2009, Oracle. All rights reserved.

## 뷰 생성(계속)

subquery에 열 alias를 포함하여 열 이름을 제어할 수 있습니다.

슬라이드의 예제는 부서 50의 모든 사원에 대해 alias가 ID\_NUMBER인 사원 번호(EMPLOYEE\_ID), alias가 NAME인 이름(LAST\_NAME) 및 alias가 ANN\_SALARY인 연봉(SALARY)을 포함하는 뷰를 생성합니다.

또는 CREATE 문과 SELECT subquery 사이에서 alias를 사용할 수 있습니다. 나열된 alias의 수와 subquery에서 선택한 표현식의 수가 일치해야 합니다.

```
CREATE OR REPLACE VIEW salvu50 (ID_NUMBER, NAME, ANN_SALARY)
AS SELECT employee_id, last_name, salary*12
      FROM employees
     WHERE department_id = 50;
```

```
CREATE VIEW succeeded.
```

## 뷰에서 데이터 검색

```
SELECT *  
FROM salvu50;
```

	ID_NUMBER	NAME	ANN_SALARY
1	124	Mourgos	69600
2	141	Rajs	42000
3	142	Davies	37200
4	143	Matos	31200
5	144	Vargas	30000

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 뷰에서 데이터 검색

테이블처럼 뷰에서도 데이터를 검색할 수 있습니다. 전체 뷰나 특정 행과 열의 내용을 표시할 수 있습니다.

## 뷰 수정

- CREATE OR REPLACE VIEW 절을 사용하여 EMPVU80 뷰를 수정합니다. 각 열 이름에 alias를 추가합니다.

```
CREATE OR REPLACE VIEW empvu80
  (id_number, name, sal, department_id)
AS SELECT employee_id, first_name || ' '
      || last_name, salary, department_id
  FROM employees
 WHERE department_id = 80;
CREATE OR REPLACE VIEW succeeded.
```

- CREATE OR REPLACE VIEW 절의 열 alias는 subquery의 열과 동일한 순서로 나열됩니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 뷰 수정

OR REPLACE 옵션을 사용하면 이미 동일한 이름의 뷰가 있는 경우에도 뷰가 생성되므로 이전 버전의 뷰를 해당 소유자에 맞게 대체할 수 있습니다. 즉, 뷰를 삭제하고 다시 생성하고 객체 권한을 다시 부여하지 않고서도 뷰를 변경할 수 있습니다.

참고: CREATE OR REPLACE VIEW 절에서 열 alias를 지정할 때 alias가 subquery의 열과 동일한 순서로 나열되어야 합니다.

## 복합 뷰 생성

그룹 함수를 포함하는 복합 뷰를 생성하여 두 테이블의 값을 표시합니다.

```
CREATE OR REPLACE VIEW dept_sum_vu
  (name, minsal, maxsal, avgsal)
AS SELECT      d.department_name, MIN(e.salary),
                MAX(e.salary), AVG(e.salary)
  FROM        employees e JOIN departments d
  ON          (e.department_id = d.department_id)
 GROUP BY    d.department_name;
CREATE OR REPLACE VIEW succeeded.
```

Copyright © 2009, Oracle. All rights reserved.

### 복합 뷰 생성

슬라이드의 예제에서는 각 부서별로 부서 이름, 최소 급여, 최대 급여 및 평균 급여를 포함하는 복합 뷰를 생성합니다. 뷰에 다른 이름이 지정되었습니다. 뷰의 모든 열이 함수나 표현식으로부터 파생되는 경우에는 다른 이름을 지정해야 합니다.

DESCRIBE 명령을 사용하여 뷰의 구조를 볼 수 있습니다. SELECT 문을 실행하여 뷰의 내용을 표시합니다.

```
SELECT *
FROM dept_sum_vu;
```

	NAME	MINSAL	MAXSAL	AVGSAL
1	Administration	4400	4400	4400
2	Accounting	8300	12000	10150
3	IT	4200	9000	6400
4	Executive	17000	24000	19333.333333333...
5	Shipping	2500	5800	3500
6	Sales	8600	11000	10033.333333333...
7	Marketing	6000	13000	9500

## 뷰에 대한 DML 작업 수행 규칙

- 단순 뷰에서는 대개 DML 작업을 수행할 수 있습니다. 
- 뷰에 다음 항목이 포함되어 있으면 행을 제거할 수 없습니다.
  - 그룹 함수
  - GROUP BY 절
  - DISTINCT 키워드
  - Pseudocolumn ROWNUM 키워드

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 뷰에 대한 DML 작업 수행 규칙

- 해당 작업이 특정 규칙을 따르는 경우 뷰를 통해 데이터에 DML 작업을 수행할 수 있습니다.
- 뷰에 다음 항목들이 포함되어 있지 않으면 뷰에서 행을 제거할 수 있습니다.
  - 그룹 함수
  - GROUP BY 절
  - DISTINCT 키워드
  - Pseudocolumn ROWNUM 키워드

## 뷰에 대한 DML 작업 수행 규칙

뷰에 다음 항목이 포함되어 있으면 뷰의 데이터를 수정할 수 없습니다.

- 그룹 함수
- GROUP BY 절
- DISTINCT 키워드
- Pseudocolumn ROWNUM 키워드
- 표현식으로 정의되는 열



Copyright © 2009, Oracle. All rights reserved.

### 뷰에 대한 DML 작업 수행 규칙(계속)

뷰에 이전 슬라이드에서 언급한 조건이나 표현식으로 정의된 열(예: SALARY \* 12)이 포함되어 있지 않으면 뷰를 통해 데이터를 수정할 수 있습니다.

## 뷰에 대한 DML 작업 수행 규칙

뷰에 다음 항목이 포함되어 있으면 뷰를 통해 데이터를 추가할 수 없습니다.

- 그룹 함수
- GROUP BY 절
- DISTINCT 키워드
- Pseudocolumn ROWNUM 키워드
- 표현식으로 정의되는 열
- 뷰에서 선택되지 않은 기본 테이블의 NOT NULL 열

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 뷰에 대한 DML 작업 수행 규칙(계속)

슬라이드에 나열된 항목이 뷰에 포함되어 있지 않으면 뷰를 통해 데이터를 추가할 수 있습니다. 기본 테이블의 기본값이 없는 NOT NULL 열이 뷰에 포함되어 있으면 뷰에 데이터를 추가할 수 없습니다. 필요한 모든 값이 뷰에 있어야 합니다. 뷰를 통해 기본 테이블에 직접 값을 추가한다는 것을 기억하십시오.

자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 "CREATE VIEW" 관련 섹션을 참조하십시오.

## WITH CHECK OPTION 절 사용

- WITH CHECK OPTION 절을 사용하여 뷰에 수행된 DML 작업이 뷰 영역에만 적용되도록 할 수 있습니다.

```
CREATE OR REPLACE VIEW empu20
AS SELECT      *
   FROM employees
  WHERE department_id = 20
    WITH CHECK OPTION CONSTRAINT empu20_ck ;
CREATE OR REPLACE VIEW succeeded.
```

- department\_id가 200이 아닌 행을 INSERT하거나 뷰에 있는 임의의 행에서 부서 번호를 UPDATE하려는 시도는 WITH CHECK OPTION 제약 조건에 위반되므로 실패합니다.

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### WITH CHECK OPTION 절 사용

뷰를 통해 참조 무결성 검사를 수행할 수 있습니다. 또한 데이터베이스 레벨에서 제약 조건을 적용할 수도 있습니다. 매우 제한적이기는 하지만 뷰를 사용하여 데이터 무결성을 보호할 수 있습니다.

WITH CHECK OPTION 절은 뷰를 통해 수행된 INSERT 및 UPDATE 문이 해당 뷰에서 선택할 수 없는 행을 생성하지 못하도록 지정합니다. 따라서 삽입 또는 갱신될 데이터에 대해 무결성 제약 조건과 데이터 유효성 검사를 강제로 적용할 수 있습니다. 뷰에서 선택하지 않은 행에서 DML 작업을 수행하려고 시도하면 제약 조건 이름이 지정된 경우 해당 이름과 함께 오류가 표시됩니다.

```
UPDATE empu20
   SET department_id = 10
 WHERE employee_id = 201;
```

원인:

```
Error report:
SQL Error: ORA-01402: view WITH CHECK OPTION where-clause violation
01402. 00000 -  "view WITH CHECK OPTION where-clause violation"
```

참고: 부서 번호가 10으로 변경된 경우 뷰에 더 이상 해당 사원을 표시할 수 없으므로 행은 갱신되지 않습니다. 따라서 WITH CHECK OPTION 절을 사용할 경우 뷰에서 부서 20에 있는 사원만 볼 수 있으며 이러한 사원의 부서 번호를 뷰를 통해 변경할 수 없습니다.

## DML 작업 거부

- **WITH READ ONLY 옵션을 뷰 정의에 추가하여 DML 작업이 발생하지 않도록 할 수 있습니다.**
- **뷰에 있는 임의의 행에서 DML 작업을 수행하려고 시도하면 Oracle 서버 오류가 발생합니다.**



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### DML 작업 거부

WITH READ ONLY 옵션을 사용하여 뷰를 생성하면 뷰에서 DML 작업이 발생하지 않도록 할 수 있습니다. 다음 슬라이드의 예제는 EMPVU10 뷰의 모든 DML 작업을 차단하도록 뷰를 수정합니다.

## DML 작업 거부

```
CREATE OR REPLACE VIEW empvu10
  (employee_number, employee_name, job_title)
AS SELECT      employee_id, last_name, job_id
   FROM        employees
  WHERE      department_id = 10
    WITH READ ONLY ;
CREATE OR REPLACE VIEW succeeded.
```

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### DML 작업 거부(계속)

읽기 전용 제약 조건이 있는 뷰에서 행을 제거하려고 시도하면 오류가 발생합니다.

```
DELETE FROM empvu10
WHERE employee_number = 200;
```

마찬가지로 읽기 전용 제약 조건이 있는 뷰를 사용하여 행을 삽입하거나 수정하려고 하면 같은 오류가 발생합니다.

Error report:
SQL Error: ORA-42399: cannot perform a DML operation on a read-only view

## 뷰 제거

**뷰는 데이터베이스의 기본 테이블을 기반으로 하기 때문에 뷰를 제거해도 데이터는 손실되지 않습니다.**

```
DROP VIEW view;
```

```
DROP VIEW empvu80;
```

```
DROP VIEW empvu80 succeeded.
```

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 뷰 제거

DROP VIEW 문을 사용하여 뷰를 제거할 수 있습니다. 이 명령문은 데이터베이스에서 뷰 정의를 제거합니다. 그러나 뷰를 삭제해도 뷰의 기반이 되는 테이블에는 영향을 미치지 않습니다. 또한 삭제된 뷰를 기반으로 하는 뷰나 기타 응용 프로그램은 사용할 수 없게 됩니다. 생성자나 DROP ANY VIEW 권한을 가진 유저만 뷰를 제거할 수 있습니다.

이 구문에서 *view*는 뷰 이름입니다.

## 연습 11: 1부 개요

이 연습에서는 다음 내용을 다룹니다.

- 단순 뷰 생성
- 복합 뷰 생성
- CHECK 제약 조건을 가진 뷰 생성
- 뷰의 데이터 수정 시도
- 뷰 제거

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 연습 11: 1부 개요

이 단원의 1부 연습에서는 뷰를 생성, 사용 및 제거하는 다양한 연습을 제공합니다. 이 단원 끝부분에서 문제 1-6을 완성하십시오.

## 단원 내용

- 뷰 개요:
  - 뷰에서 데이터 생성, 수정 및 검색
  - 뷰에 대한 DML 작업
  - 뷰 삭제
- 시퀀스 개요:
  - 시퀀스 생성, 사용 및 수정
  - 시퀀스 값을 캐시에 저장
  - NEXTVAL 및 CURRVAL pseudocolumn
- 인덱스 개요
  - 인덱스 생성, 삭제
- 동의어 개요
  - 동의어 생성, 삭제

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 시퀀스

객체	설명
테이블	기본 저장 단위이며 행으로 구성되어 있습니다.
뷰	하나 이상의 테이블에 있는 데이터의 부분 집합을 논리적으로 나타냅니다.
시퀀스	숫자 값을 생성합니다.
인덱스	일부 query 성능을 향상시킵니다.
동의어	객체에 다른 이름을 부여합니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

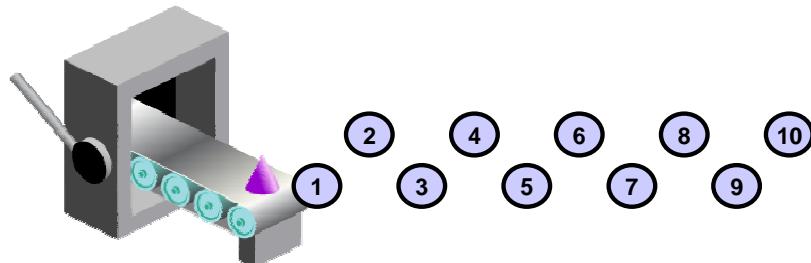
## 시퀀스

시퀀스는 정수 값을 생성하는 데이터베이스 객체입니다. 시퀀스를 생성한 다음 이 시퀀스를 사용하여 번호를 생성합니다.

# 시퀀스

## 시퀀스:

- 고유 번호를 자동으로 생성할 수 있습니다.
- 공유할 수 있는 객체입니다.
- Primary key 값을 생성하는 데 사용할 수 있습니다.
- 응용 프로그램 코드를 대체합니다.
- 시퀀스 값이 메모리에서 캐시된 경우 액세스 속도가 향상됩니다.



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 시퀀스(계속)

시퀀스는 유저가 생성한 데이터베이스 객체이며 정수를 생성하는 여러 유저가 공유할 수 있습니다. 시퀀스를 정의하여 고유 값을 생성하거나 동일한 번호를 재사용할 수 있습니다.

시퀀스의 일반적 용도는 각 행에 고유해야 하는 Primary key 값을 생성하는 경우입니다. 시퀀스는 내부 Oracle 루틴에 의해 생성되고 순차적으로 증가하거나 감소합니다. 이것은 시퀀스 생성 루틴을 작성하는 데 필요한 응용 프로그램 코드의 양을 줄일 수 있기 때문에 시간 절약형 객체라 할 수 있습니다.

시퀀스 번호는 테이블과 별도로 저장되고 생성됩니다. 따라서 여러 테이블에 동일한 시퀀스를 사용할 수 있습니다.

## CREATE SEQUENCE 문: 구문

자동으로 일련 번호를 생성하도록 시퀀스를 정의합니다.

```
CREATE SEQUENCE sequence
  [INCREMENT BY n]
  [START WITH n]
  [ {MAXVALUE n | NOMAXVALUE} ]
  [ {MINVALUE n | NOMINVALUE} ]
  [ {CYCLE | NOCYCLE} ]
  [ {CACHE n | NOCACHE} ];
```

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### CREATE SEQUENCE 문: 구문

CREATE SEQUENCE 문을 사용하여 자동으로 일련 번호를 생성합니다.

이 구문에서 다음이 적용됩니다.

<i>sequence</i>	시퀀스 생성기의 이름입니다.
<i>INCREMENT BY n</i>	시퀀스 번호 사이의 간격을 지정하며, 여기서 <i>n</i> 은 정수입니다. (이 절을 생략하면 시퀀스는 1씩 증가합니다.)
<i>START WITH n</i>	생성할 첫번째 시퀀스 번호를 지정합니다. (이 절을 생략하면 시퀀스는 1부터 시작합니다.)
<i>MAXVALUE n</i>	시퀀스가 생성할 수 있는 최대값을 지정합니다.
<i>NOMAXVALUE</i>	오름차순 시퀀스의 경우 최대값 $10^{27}$ , 내림차순 시퀀스의 경우 -1을 지정합니다(기본 옵션).
<i>MINVALUE n</i>	최소 시퀀스 값을 지정합니다.
<i>NOMINVALUE</i>	오름차순 시퀀스의 경우 최소값 1을, 내림차순 시퀀스의 경우 $-(10^{26})$ 을 지정합니다(기본 옵션).

## 시퀀스 생성

- DEPARTMENTS 테이블의 Primary key에 사용할 DEPT\_DEPTID\_SEQ라는 시퀀스를 생성합니다.
- CYCLE 옵션을 사용하지 마십시오.

```
CREATE SEQUENCE dept_deptid_seq
    INCREMENT BY 10
    START WITH 120
    MAXVALUE 9999
    NOCACHE
    NOCYCLE;
```

CREATE SEQUENCE succeeded.

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### 시퀀스 생성(계속)

CYCLE | NOCYCLE

최대값이나 최소값에 도달한 후에 시퀀스를 계속 생성할지 여부를 지정합니다. (NOCYCLE이 기본 옵션입니다.)

CACHE n | NOCACHE

Oracle 서버가 메모리에 미리 할당하고 저장하는 값의 개수를 지정합니다. (Oracle 서버는 기본적으로 20개의 값을 캐시합니다.)

슬라이드의 예제는 DEPARTMENTS 테이블의 DEPARTMENT\_ID 열에 사용할 DEPT\_DEPTID\_SEQ라는 시퀀스를 생성합니다. 이 시퀀스는 120에서 시작하고 캐시에 저장할 수 있으며 순환하지 않습니다.

시퀀스를 사용하여 Primary key 값을 생성하는 경우 시퀀스 주기보다 더 빠르게 이전 행을 지우는 믿을 만한 방법이 없으면 CYCLE 옵션을 사용하지 마십시오.

자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 "CREATE SEQUENCE" 관련 섹션을 참조하십시오.

**참고:** 시퀀스는 테이블과 연관되지 않습니다. 일반적으로 시퀀스는 용도에 따라 이름을 지정해야 합니다. 하지만 이름과 상관없이 어디에나 사용할 수 있습니다.

## NEXTVAL 및 CURRVAL Pseudocolumn

- NEXTVAL은 사용 가능한 다음 시퀀스 값을 반환합니다. 다른 유저인 경우도 포함하여 참조될 때마다 고유 값을 반환합니다.
- CURRVAL은 현재 시퀀스 값을 구합니다.
- CURRVAL이 값을 포함하기 전에 해당 시퀀스에 대해 NEXTVAL이 실행되어야 합니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### NEXTVAL 및 CURRVAL Pseudocolumn

시퀀스를 생성한 후에 테이블에서 사용할 수 있도록 일련 번호가 생성됩니다. NEXTVAL 및 CURRVAL pseudocolumn을 사용하여 시퀀스 값을 참조합니다.

NEXTVAL pseudocolumn은 지정된 시퀀스에서 연속적인 시퀀스 번호를 추출하는 데 사용됩니다. NEXTVAL을 시퀀스 이름으로 한정해야 합니다. `sequence.NEXTVAL`을 참조하면 새 시퀀스 번호가 생성되고 현재 시퀀스 번호는 CURRVAL에 입력됩니다.

CURRVAL pseudocolumn은 현재 유저가 방금 생성한 시퀀스 번호를 참조하는 데 사용됩니다. 그러나 먼저 NEXTVAL을 사용하여 현재 유저의 세션에서 시퀀스 번호를 생성해야 CURRVAL을 참조할 수 있습니다. CURRVAL을 시퀀스 이름으로 한정해야 합니다. `sequence.CURRVAL`을 참조할 경우 유저의 프로세스로 반환된 마지막 값이 표시됩니다.

## NEXTVAL 및 CURRVAL Pseudocolumn(계속)

### NEXTVAL 및 CURRVAL 사용 규칙

다음과 같은 상황에서 NEXTVAL 및 CURRVAL을 사용할 수 있습니다.

- subquery의 일부가 아닌 SELECT 문의 SELECT 리스트
- INSERT 문에서 subquery의 SELECT 리스트
- INSERT 문의 VALUES 절
- UPDATE 문의 SET 절

다음과 같은 상황에서는 NEXTVAL 및 CURRVAL을 사용할 수 없습니다.

- 뷰의 SELECT 리스트
- DISTINCT 키워드가 있는 SELECT 문
- GROUP BY, HAVING 또는 ORDER BY 절이 있는 SELECT 문
- SELECT, DELETE 또는 UPDATE 문의 subquery
- CREATE TABLE 또는 ALTER TABLE 문의 DEFAULT 식

자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 "Pseudocolumns" 및 "CREATE SEQUENCE" 관련 섹션을 참조하십시오.

## 시퀀스 사용

- 위치 ID 2500에 "Support"라는 새 부서를 삽입합니다.

```
INSERT INTO departments(department_id,
                       department_name, location_id)
VALUES      (dept_deptid_seq.NEXTVAL,
              'Support', 2500);

1 rows inserted
```

- DEPT\_DEPTID\_SEQ 시퀀스의 현재 값을 확인합니다.

```
SELECT dept_deptid_seq.CURRVAL
FROM dual;
```

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### 시퀀스 사용

슬라이드의 예제는 DEPARTMENTS 테이블에 새 부서를 삽입합니다. 다음과 같이 DEPT\_DEPTID\_SEQ 시퀀스를 사용하여 새 부서 번호를 생성합니다.

슬라이드의 두번째 예제에 나와 있는 대로 *sequence\_name.CURRVAL*을 사용하여 시퀀스의 현재 값을 확인할 수 있습니다. 이 query의 출력은 다음과 같습니다.

AZ	CURRVAL
1	120

이제 사원을 채용하여 새 부서에 배치하려 한다고 가정합시다. 모든 신입 사원에 대해 실행되는 INSERT 문에 다음 코드를 포함할 수 있습니다.

```
INSERT INTO employees (employee_id, department_id, ...)
VALUES (employees_seq.NEXTVAL, dept_deptid_seq.CURRVAL, ...);
```

**참고:** 위의 예제에서는 새 사원 번호를 생성하기 위해 이미 EMPLOYEE\_SEQ라는 시퀀스가 생성되었다고 가정합니다.

## 시퀀스 값 캐시

- 시퀀스 값을 메모리에 캐시하면 해당 값에 빠르게 액세스할 수 있습니다.
- 다음과 같은 경우 시퀀스 값에 간격이 발생할 수 있습니다.
  - 롤백이 발생하는 경우
  - 시스템 작동이 중단되는 경우
  - 시퀀스가 다른 테이블에서 사용되는 경우

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 시퀀스 값 캐시

시퀀스를 메모리에 캐시하면 해당 시퀀스 값에 빠르게 액세스할 수 있습니다. 캐시는 처음 시퀀스를 참조할 때 채워집니다. 다음 시퀀스 값을 요청할 때마다 캐시된 시퀀스에서 검색합니다. 마지막 시퀀스 값이 사용된 후 다음 시퀀스 요청 시 다른 시퀀스 캐시를 메모리로 가져옵니다.

### 시퀀스 간격

시퀀스 생성기는 간격 없이 일련 번호를 생성하지만 이 작업은 커밋이나 롤백에 관계없을 때만 그렇습니다. 따라서 시퀀스가 포함된 명령문을 롤백하면 번호를 잃게 됩니다.

시퀀스에 간격을 발생시킬 수 있는 또 다른 이벤트는 시스템에 장애가 발생하는 경우입니다. 시퀀스가 메모리에 값을 캐시한 경우 시스템이 중단되면 해당 값을 잃게 됩니다.

시퀀스는 테이블과 직접 연관되지 않기 때문에 여러 테이블에 대해 동일한 시퀀스를 사용할 수 있습니다. 그러나 이렇게 하면 각 테이블의 일련 번호에 간격이 포함될 수 있습니다.

## 시퀀스 수정

**증분값, 최대값, 최소값, 순환 옵션 또는 캐시 옵션을 변경합니다.**

```
ALTER SEQUENCE dept_deptid_seq
    INCREMENT BY 20
    MAXVALUE 999999
    NOCACHE
    NOCYCLE;
```

```
ALTER SEQUENCE dept_deptid_seq succeeded.
```

Copyright © 2009, Oracle. All rights reserved.

### 시퀀스 수정

시퀀스의 MAXVALUE 한계에 도달하면 시퀀스에서 추가 값이 할당되지 않고 시퀀스가 MAXVALUE 한계를 초과했음을 나타내는 오류 메시지가 수신됩니다. ALTER SEQUENCE 문을 사용하여 시퀀스를 수정하면 계속 사용할 수 있습니다.

#### 구문

```
ALTER SEQUENCE      sequence
  [INCREMENT BY n]
  [ {MAXVALUE n | NOMAXVALUE} ]
  [ {MINVALUE n | NOMINVALUE} ]
  [ {CYCLE | NOCYCLE} ]
  [ {CACHE n | NOCACHE} ];
```

이 구문에서 *sequence*는 시퀀스 생성기의 이름입니다.

자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 "ALTER SEQUENCE" 관련 섹션을 참조하십시오.

## 시퀀스 수정 지침

- 시퀀스의 소유자이거나 시퀀스에 대해 ALTER 권한을 가져야 합니다.
- 다음 시퀀스 번호에만 적용됩니다.
- 다른 번호로 시퀀스를 다시 시작하려면 시퀀스를 삭제하고 다시 생성해야 합니다.
- 일부 유효성 검사가 수행됩니다.
- 시퀀스를 제거하려면 DROP 문을 사용합니다.

```
DROP SEQUENCE dept_deptid_seq;
DROP SEQUENCE dept_deptid_seq succeeded.
```

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### 시퀀스 수정 지침

- 시퀀스를 수정하려면 시퀀스의 소유자이거나 시퀀스에 대해 ALTER 권한을 가져야 합니다. 시퀀스를 제거하려면 시퀀스의 소유자이거나 DROP ANY SEQUENCE 권한을 가져야 합니다.
- ALTER SEQUENCE 문은 후속 시퀀스 번호에만 적용됩니다.
- START WITH 옵션은 ALTER SEQUENCE를 사용하여 변경할 수 없습니다. 다른 번호로 시퀀스를 다시 시작하려면 시퀀스를 삭제하고 다시 생성해야 합니다.
- 일부 유효성 검사가 수행됩니다. 예를 들어, 새 MAXVALUE를 현재 시퀀스 번호보다 작게 지정할 수 없습니다.

```
ALTER SEQUENCE dept_deptid_seq
    INCREMENT BY 20
    MAXVALUE 90
    NOCACHE
    NOCYCLE;
```

- 오류:

Error report:
SQL Error: ORA-04009: MAXVALUE cannot be made to be less than the current value 04009. 00000 - "MAXVALUE cannot be made to be less than the current value"
*Cause: the current value exceeds the given MAXVALUE
*Action: make sure that the new MAXVALUE is larger than the current value

# 단원 내용

- 뷰 개요:
  - 뷰에서 데이터 생성, 수정 및 검색
  - 뷰에 대한 DML 작업
  - 뷰 삭제
- 시퀀스 개요:
  - 시퀀스 생성, 사용 및 수정
  - 시퀀스 값을 캐시에 저장
  - NEXTVAL 및 CURRVAL pseudocolumn
- 인덱스 개요
  - 인덱스 생성, 삭제
- 동의어 개요
  - 동의어 생성, 삭제

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 인덱스

객체	설명
테이블	기본 저장 단위이며 행으로 구성되어 있습니다.
뷰	하나 이상의 테이블에 있는 데이터의 부분 집합을 논리적으로 나타냅니다.
시퀀스	숫자 값을 생성합니다.
인덱스	일부 query 성능을 향상시킵니다.
동의어	객체에 다른 이름을 부여합니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

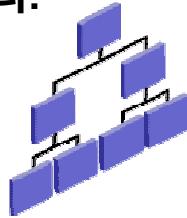
## 인덱스

데이터베이스 객체인 인덱스를 생성하면 일부 query의 성능을 향상시킬 수 있습니다. 인덱스는 유저가 Primary key 또는 Unique 제약 조건을 생성할 때 서버에 의해 자동으로 생성됩니다.

# 인덱스

## 인덱스:

- 스키마 객체입니다.
- Oracle 서버에서 포인터를 사용하여 행 검색 속도를 높이는 데 사용할 수 있습니다.
- 신속한 경로 액세스 방식을 사용하여 데이터를 빠르게 찾아 디스크 I/O(입/출력)를 줄일 수 있습니다.
- 인덱스의 대상인 테이블에 독립적입니다.
- Oracle 서버에서 자동으로 사용되고 유지 관리됩니다.



**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

## 인덱스(계속)

Oracle 서버 인덱스는 포인터를 사용하여 행 검색 속도를 높일 수 있는 스키마 객체입니다. 명시적으로 또는 자동으로 인덱스를 생성할 수 있습니다. 열에 인덱스가 없으면 전체 테이블 스캔이 수행됩니다.

인덱스를 사용하면 테이블의 행에 직접 빠르게 액세스할 수 있습니다. 인덱스는 인덱스화된 경로를 사용하여 데이터를 신속하게 찾음으로써 디스크 I/O를 줄이는 데 그 목적이 있습니다. 인덱스는 Oracle 서버에서 자동으로 사용되고 유지 관리됩니다. 인덱스가 생성된 후에는 유저가 직접 조작할 필요가 없습니다.

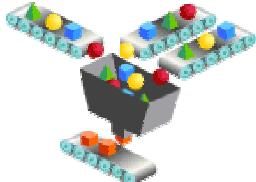
인덱스는 논리적, 물리적으로 인덱스의 대상인 테이블에 독립적입니다. 즉, 인덱스는 언제든지 생성하고 삭제할 수 있으며 기본 테이블이나 다른 인덱스에 영향을 미치지 않습니다.

**참고:** 테이블을 삭제하면 해당 인덱스 또한 삭제됩니다.

자세한 내용은 *Oracle Database Concepts 11g, Release 1 (11.1)*에서 "Schema Objects: Indexes" 관련 섹션을 참조하십시오.

## 인덱스가 생성되는 방식

- 자동으로: 테이블 정의에서 PRIMARY KEY 또는 UNIQUE 제약 조건을 정의하면 고유 인덱스가 자동으로 생성됩니다.



- 수동으로: 행에 액세스하는 속도를 높이기 위해 유저가 열의 비고유 인덱스를 생성할 수 있습니다.



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 인덱스가 생성되는 방식

두 가지 유형의 인덱스를 생성할 수 있습니다.

- 고유 인덱스: 테이블의 열이 PRIMARY KEY 또는 UNIQUE 제약 조건을 갖도록 정의하면 Oracle 서버가 자동으로 고유 인덱스를 생성합니다. 인덱스의 이름은 제약 조건에 지정된 이름입니다.
- 비고유 인덱스: 유저가 생성할 수 있는 인덱스입니다. 예를 들어, query의 조인을 위해 FOREIGN KEY 열 인덱스를 생성하여 검색 속도를 높일 수 있습니다.

**참고:** 고유 인덱스를 수동으로 생성할 수도 있지만, 고유 인덱스를 암시적으로 생성하는 고유 제약 조건을 생성할 것을 권장합니다.

## 인덱스 생성

- 하나 이상의 열에 인덱스를 생성합니다.

```
CREATE [UNIQUE][BITMAP] INDEX index
ON table (column[, column]...);
```

- EMPLOYEES 테이블의 LAST\_NAME 열에 대한 query 액세스 속도를 향상시킵니다.

```
CREATE INDEX emp_last_name_idx
ON employees(last_name);
CREATE INDEX succeeded.
```

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### 인덱스 생성

CREATE INDEX 문을 실행하여 하나 이상의 열에 인덱스를 생성합니다.

이 구문에서 다음이 적용됩니다.

- index 인덱스의 이름입니다.
- table 테이블의 이름입니다.
- column 인덱스화될 테이블의 열 이름입니다.

인덱스가 기반으로 하는 열의 값이 고유해야 함을 나타내려면 UNIQUE를 지정합니다. 각 행을 별도로 인덱스화하지 않고 각 구분 키에 대한 비트맵을 사용하여 인덱스가 생성되도록 하려면 BITMAP을 지정합니다. 비트맵 인덱스에서는 키 값과 연관된 rowid를 비트맵으로 저장합니다.

자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 "CREATE INDEX" 관련 섹션을 참조하십시오.

## 인덱스 생성 지침

### 다음 경우에 인덱스를 생성하십시오.

<input checked="" type="checkbox"/>	열에 광범위한 값이 포함된 경우
<input checked="" type="checkbox"/>	열에 많은 널 값이 포함된 경우
<input checked="" type="checkbox"/>	하나 이상의 열이 WHERE 절이나 조인 조건에서 함께 자주 사용되는 경우
<input checked="" type="checkbox"/>	테이블이 크고 대부분의 query가 테이블에서 2%~4% 미만의 행을 검색할 것으로 예상되는 경우

### 다음 경우에는 인덱스를 생성하지 마십시오.

<input checked="" type="checkbox"/>	열이 query에서 조건으로 자주 사용되지 않는 경우
<input checked="" type="checkbox"/>	테이블이 작거나 대부분의 query가 테이블에서 2%~4% 이상의 행을 검색할 것으로 예상되는 경우
<input checked="" type="checkbox"/>	테이블이 자주 갱신되는 경우
<input checked="" type="checkbox"/>	인덱스화된 열이 표현식의 일부로 참조되는 경우

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 인덱스 생성 지침

### 많다고 좋은 것은 아님

테이블에 인덱스가 많다고 해서 query 속도가 빨라지는 것은 아닙니다. 각 DML 작업이 인덱스가 있는 테이블에 커밋되어 있다는 것은 인덱스가 갱신되어야 함을 의미합니다. 테이블과 연관된 인덱스가 많을수록 DML 작업 후에 모든 인덱스를 갱신해야 하므로 Oracle 서버의 부담이 늘어납니다.

### 인덱스를 생성해야 하는 경우

따라서 다음과 같은 경우에만 인덱스를 생성해야 합니다.

- 열에 광범위한 값이 포함된 경우
- 열에 많은 널 값이 포함된 경우
- 하나 이상의 열이 WHERE 절이나 조인 조건에서 함께 자주 사용되는 경우
- 테이블이 크고 대부분의 query가 2%~4% 미만의 행을 검색할 것으로 예상되는 경우

고유성을 강화하려면 테이블 정의에서 고유 제약 조건을 정의해야 합니다. 그러면 고유 인덱스가 자동으로 생성됩니다.

## 인덱스 제거

- **DROP INDEX 명령을 사용하여 데이터 딕셔너리에서 인덱스를 제거할 수 있습니다.**

```
DROP INDEX index;
```

- **데이터 딕셔너리에서 emp\_last\_name\_idx 인덱스를 제거합니다.**

```
DROP INDEX emp_last_name_idx;  
DROP INDEX emp_last_name_idx succeeded.
```

- **인덱스를 삭제하려면 인덱스의 소유자이거나 DROP ANY INDEX 권한이 있어야 합니다.**

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 인덱스 제거

인덱스는 수정할 수 없습니다. 인덱스를 변경하려면 인덱스를 삭제한 다음 다시 생성해야 합니다.

DROP INDEX 문을 실행하여 데이터 딕셔너리에서 인덱스 정의를 제거합니다. 인덱스를 삭제하려면 인덱스의 소유자이거나 DROP ANY INDEX 권한이 있어야 합니다.

이 구문에서 *index*는 인덱스의 이름입니다.

**참고:** 테이블을 삭제하면 인덱스와 제약 조건이 자동으로 삭제되지만 뷰와 시퀀스는 남아 있습니다.

## 단원 내용

- 뷰 개요:
  - 뷰에서 데이터 생성, 수정 및 검색
  - 뷰에 대한 DML 작업
  - 뷰 삭제
- 시퀀스 개요:
  - 시퀀스 생성, 사용 및 수정
  - 시퀀스 값을 캐시에 저장
  - NEXTVAL 및 CURRVAL pseudocolumn
- 인덱스 개요
  - 인덱스 생성, 삭제
- 동의어 개요
  - 동의어 생성, 삭제

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 동의어

객체	설명
테이블	기본 저장 단위이며 행으로 구성되어 있습니다.
뷰	하나 이상의 테이블에 있는 데이터의 부분 집합을 논리적으로 나타냅니다.
시퀀스	숫자 값을 생성합니다.
인덱스	일부 query 성능을 향상시킵니다.
동의어	객체에 다른 이름을 부여합니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 동의어

동의어는 다른 이름으로 테이블을 호출할 수 있는 데이터베이스 객체입니다. 동의어를 생성하여 테이블에 대체 이름을 부여할 수 있습니다.

## 객체의 동의어 생성

**동의어(객체의 또 다른 이름)를 생성하면 객체에 쉽게 액세스할 수 있습니다. 동의어를 사용하여 다음과 같은 작업을 할 수 있습니다.**

- **다른 유저가 소유한 테이블을 쉽게 참조할 수 있습니다.**
- **긴 객체 이름을 짧게 만듭니다.**

```
CREATE [PUBLIC] SYNONYM synonym
FOR      object;
```

Copyright © 2009, Oracle. All rights reserved.

### 객체의 동의어 생성

다른 유저가 소유한 테이블을 참조하려면 테이블 이름 앞에 해당 테이블을 만든 유저 이름을 추가하고 마침표를 입력해야 합니다. 동의어를 생성하면 객체 이름을 스키마로 제한할 필요가 없고 테이블, 뷰, 시퀀스, 프로시저 또는 다른 객체에 대해 다른 이름을 지정할 수 있습니다. 이 방법은 뷰와 같이 객체 이름이 긴 경우에 특히 유용합니다.

이 구문에서 다음이 적용됩니다.

PUBLIC	모든 유저가 액세스할 수 있는 동의어를 생성합니다.
<i>synonym</i>	생성할 동의어의 이름입니다.
<i>object</i>	동의어를 생성할 객체를 식별합니다.

#### 자침

- 객체는 패키지에 포함될 수 없습니다.
- 전용(private) 동의어 이름은 동일한 유저가 소유한 모든 다른 객체와 구분되어야 합니다.

자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 "CREATE SYNONYM" 관련 섹션을 참조하십시오.

## 동의어 생성 및 제거

- DEPT\_SUM\_VU 뷰의 짧은 이름을 생성합니다.

```
CREATE SYNONYM d_sum
FOR dept_sum_vu;
CREATE SYNONYM succeeded.
```

- 동의어를 삭제합니다.

```
DROP SYNONYM d_sum;
DROP SYNONYM d_sum succeeded.
```

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### 동의어 생성 및 제거

#### 동의어 생성

슬라이드의 예제는 빠른 참조를 위해 DEPT\_SUM\_VU 뷰의 동의어를 생성합니다.

데이터베이스 관리자는 모든 유저가 액세스할 수 있는 공용(public) 동의어를 생성할 수 있습니다.  
다음 예제는 Alice의 DEPARTMENTS 테이블에 대해 DEPT라는 공용(public) 동의어를 생성합니다.

```
CREATE PUBLIC SYNONYM dept
CREATE SYNONYM succeeded.
```

#### 동의어 제거

동의어를 제거하려면 DROP SYNONYM 문을 사용합니다. 데이터베이스 관리자만 공용(public) 동의어를 삭제할 수 있습니다.

```
DROP PUBLIC SYNONYM dept;
```

자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 "DROP SYNONYM" 관련 섹션을 참조하십시오.

## 퀴즈

**인덱스는 수동으로 만들어야 하며 테이블의 행에 액세스하는 속도를 빠르게 하는 역할을 합니다.**

- 1. 맞습니다.**
- 2. 틀립니다.**

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### 정답: 2

**참고:** 인덱스는 query 속도를 높이기 위해 만들어집니다. 하지만 모든 인덱스를 수동으로 만드는 것은 아닙니다. 테이블의 열이 PRIMARY KEY 또는 UNIQUE 제약 조건을 갖도록 정의하면 Oracle 서버가 자동으로 고유 인덱스를 생성합니다.

## 요약

이 단원에서는 다음 항목에 대해 설명했습니다.

- 뷰 생성, 사용 및 제거
- 시퀀스 생성기를 사용하여 자동으로 시퀀스 번호 생성
- 인덱스를 생성하여 query 검색 속도 향상
- 동의어를 사용하여 객체에 대체 이름 제공

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 요약

이 단원에서는 뷰, 시퀀스, 인덱스 및 동의어와 같은 데이터베이스 객체에 대해 배웠습니다.

## 연습 11: 2부 개요

이 연습에서는 다음 내용을 다룹니다.

- 시퀀스 생성
- 시퀀스 사용
- 비고유 인덱스 생성
- 동의어 생성

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 연습 11: 2부 개요

이 단원의 2부 연습에서는 다양한 예제를 통해 시퀀스, 인덱스 및 동의어를 생성하고 사용하는 과정을 실습합니다.

이 단원 끝부분에서 문제 7-10을 완성하십시오.

---

**부록 A**  
**연습 및 해답**

---

## 목차

단원 I의 연습 .....	3
연습 I-1: 소개 .....	4
연습 I-1 해답: 소개 .....	5
단원 1의 연습 .....	10
연습 1-1: SQL SELECT 문을 사용하여 데이터 검색 .....	11
연습 1-1 해답: SQL SELECT 문을 사용하여 데이터 검색 .....	15
단원 2의 연습 .....	18
연습 2-1: 데이터 제한 및 정렬 .....	19
연습 2-1 해답: 데이터 제한 및 정렬 .....	23
단원 3의 연습 .....	26
연습 3-1: 단일 행 함수를 사용하여 출력 커스터마이즈 .....	27
연습 3-1 해답: 단일 행 함수를 사용하여 출력 커스터마이즈 .....	31
단원 4의 연습 .....	34
연습 4-1: 변환 함수 및 조건부 표현식 사용 .....	35
연습 4-1 해답: 변환 함수 및 조건부 표현식 사용 .....	38
단원 5의 연습 .....	40
연습 5-1: 그룹 함수를 사용하여 집계 데이터 보고 .....	41
연습 5-1 해답: 그룹 함수를 사용하여 집계 데이터 보고 .....	44
단원 6의 연습 .....	46
연습 6-1: 조인을 사용하여 여러 테이블의 데이터 표시 .....	47
연습 6-1 해답: 조인을 사용하여 여러 테이블의 데이터 표시 .....	50
단원 7의 연습 .....	52
연습 7-1: Subquery 를 사용하여 Query 해결 .....	53
연습 7-1 해답: Subquery 를 사용하여 Query 해결 .....	55
단원 8의 연습 .....	57
연습 8-1: 집합 연산자 사용 .....	58
연습 8-1 해답: 집합 연산자 사용 .....	60
단원 9의 연습 .....	62
연습 9-1: 데이터 조작 .....	63
연습 9-1 해답: 데이터 조작 .....	66
단원 10의 연습 .....	70
연습 10-1: DDL 문을 사용하여 테이블 생성 및 관리 .....	71
연습 10-1 해답: DDL 문을 사용하여 테이블 생성 및 관리 .....	73
단원 11의 연습 .....	75
연습 11-1: 스키마 객체 관리 .....	76
연습 11-1 해답: 다른 스키마 객체 생성 .....	78
부록 F 해답 .....	80
부록 F-1: Oracle 조인 구문 .....	81
연습 F-1 해답: Oracle 조인 구문 .....	84

## 단원 I의 연습

이 연습에서는 다음을 수행합니다.

- Oracle SQL Developer를 시작하고 ora1 계정에 대한 새 연결을 생성합니다.
- Oracle SQL Developer를 사용하여 ora1 계정에서 데이터 객체를 검사합니다.  
ora1 계정에는 HR 스키마 테이블이 포함됩니다.

다음 lab 파일 위치를 기록해 두십시오.

\home\oracle\labs\sql1\labs

lab 파일을 저장하라는 요청을 받으면 이 위치에 저장하십시오.

각 연습에서 "시간 여유가 있을 경우" 또는 "심화 연습에 도전하려면"으로 시작하는 연습 과정이 제공될 수 있습니다. 이러한 연습은 할당된 시간 내에 다른 연습을 모두 완료한 다음 자신의 실력을 좀더 테스트해 보고자 하는 경우에만 수행하십시오.

연습은 천천히 정확하게 수행하십시오. 명령 파일을 저장하거나 실행해 볼 수 있습니다. 의문 사항이 있으면 언제든지 강사에게 문의하십시오.

### 참고

- 1) 모든 연습 문제는 Oracle SQL Developer를 개발 환경으로 사용합니다. Oracle SQL Developer를 사용하는 것이 좋지만 본 과정에서 사용 가능한 SQL\*Plus를 사용할 수도 있습니다.
- 2) Query의 경우 데이터베이스에서 검색된 행의 시퀀스가 표시된 스크린샷과 다를 수 있습니다.

## 연습 I-1: 소개

본 과정에 나오는 여러 연습 중 첫번째입니다. 필요한 경우 본 연습의 끝 부분에서 해답을 찾아볼 수 있습니다. 연습에서는 해당 단원에 나오는 대부분의 주제를 다룹니다.

### Oracle SQL Developer 시작

- 1) SQL Developer 바탕 화면 아이콘을 사용하여 Oracle SQL Developer 를 시작합니다.

### 새 Oracle SQL Developer 데이터베이스 연결 생성

- 2) 새 데이터베이스 연결을 생성하려면 Connections Navigator에서 Connections 를 마우스 오른쪽 버튼으로 누릅니다. 메뉴에서 New Connection 을 선택합니다. New>Select Database Connection 대화상자가 나타납니다.
- 3) 다음 정보를 사용하여 데이터베이스 연결을 생성합니다.
  - a) Connection Name: myconnection
  - b) Username: ora1
  - c) Password: ora1
  - d) Hostname: localhost
  - e) Port: 1521
  - f) SID: ORCL

Save Password 체크 박스를 선택합니다.

### Oracle SQL Developer 데이터베이스 연결을 사용하여 테스트 및 연결

- 4) 새 연결을 테스트합니다.
- 5) 상태가 Success 이면 새로운 이 연결을 사용하여 데이터베이스에 연결합니다.

### Connections Navigator에서 테이블 탐색

- 6) Connections Navigator 의 Tables 노드에서 사용할 수 있는 객체를 확인합니다. 다음 테이블이 있는지 확인합니다.

```
COUNTRIES
DEPARTMENTS
EMPLOYEES
JOB_GRADES
JOB_HISTORY
JOBS
LOCATIONS
REGIONS
```

- 7) EMPLOYEES 테이블의 구조를 살펴봅니다.
- 8) DEPARTMENTS 테이블의 데이터를 확인합니다.

## 연습 I-1 해답: 소개

### Oracle SQL Developer 시작

- 1) SQL Developer 바탕 화면 아이콘을 사용하여 Oracle SQL Developer를 시작합니다.
- a) SQL Developer 바탕 화면 아이콘을 두 번 누릅니다.

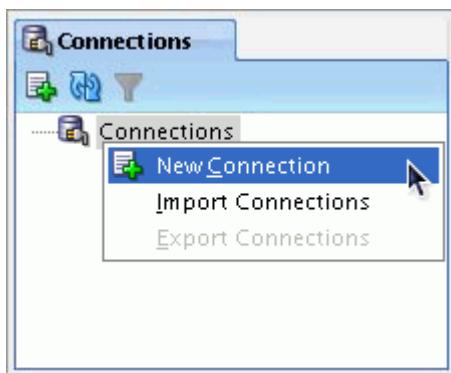


SQL Developer 인터페이스가 나타납니다.



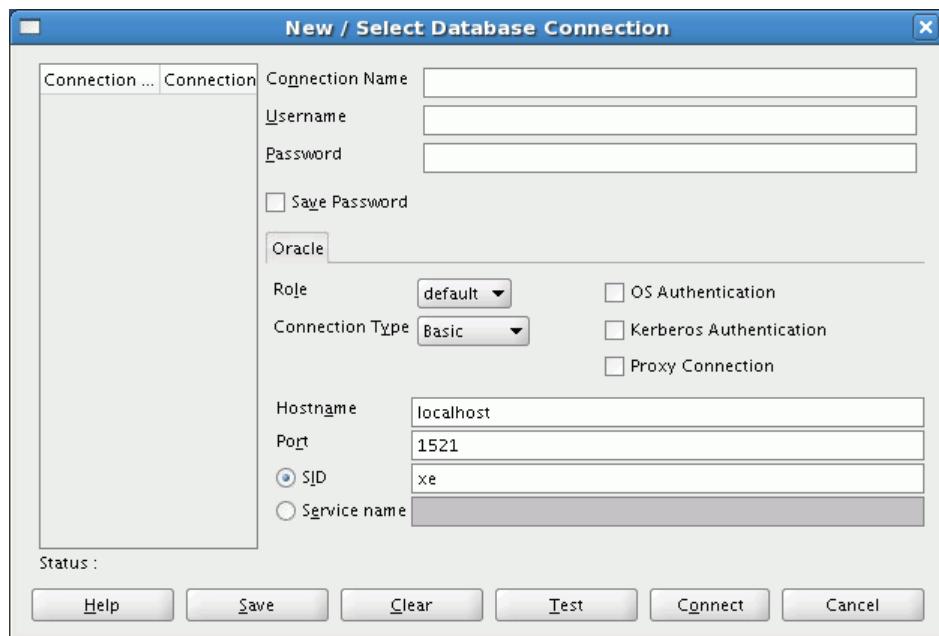
### 새 Oracle SQL Developer 데이터베이스 연결 생성

- 2) 새 데이터베이스 연결을 생성하려면 Connections Navigator에서 Connections를 마우스 오른쪽 버튼으로 누르고 메뉴에서 New Connection을 선택합니다.



## 연습 I-1 해답: 소개 (계속)

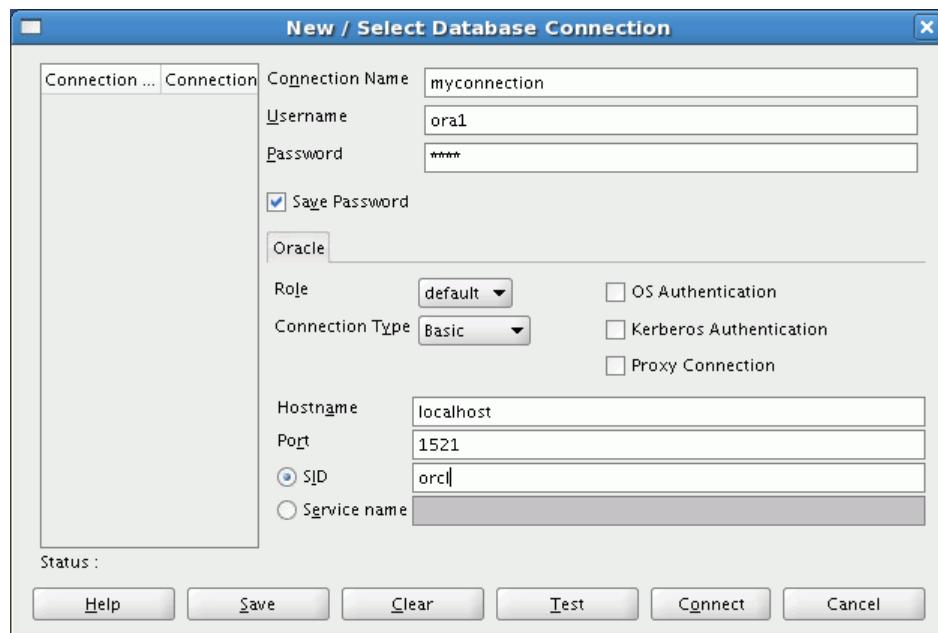
New>Select Database Connection 대화상자가 나타납니다.



3) 다음 정보를 사용하여 데이터베이스 연결을 생성합니다.

- a) Connection Name: myconnection
- b) Username: oral
- c) Password: oral
- d) Hostname: localhost
- e) Port: 1521
- f) SID: ORCL

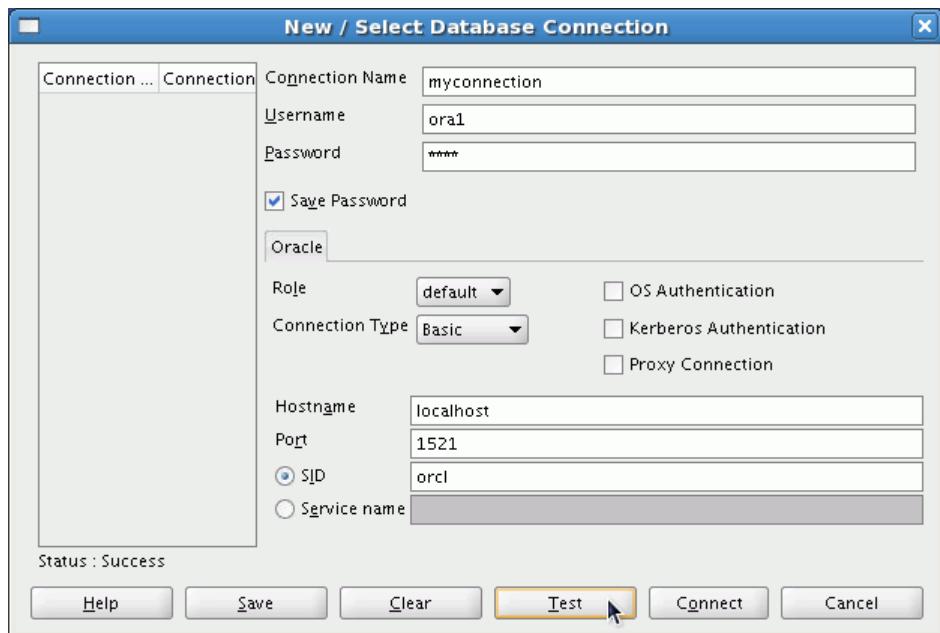
Save Password 체크 박스를 선택합니다.



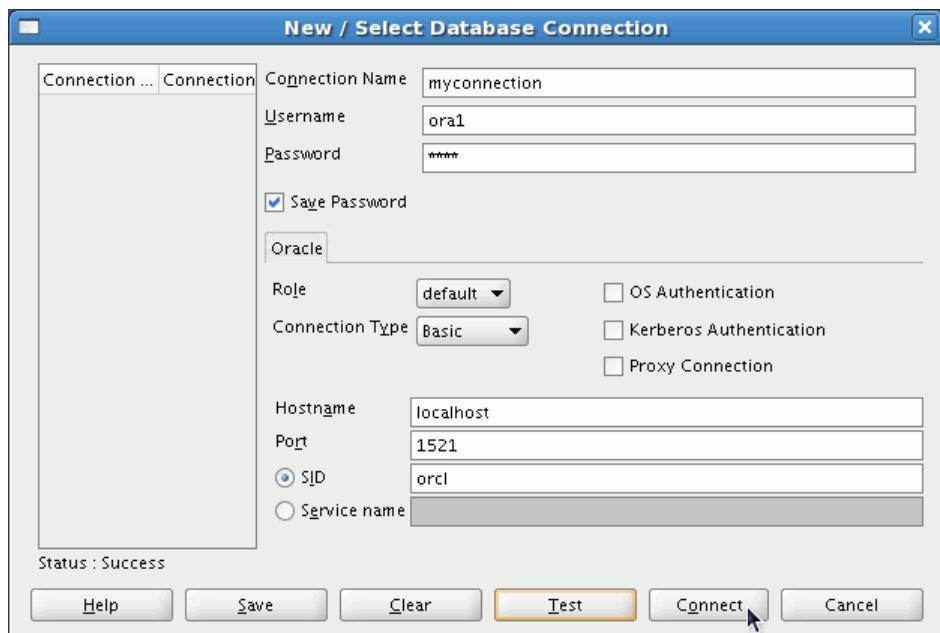
## 연습 I-1 해답: 소개 (계속)

Oracle SQL Developer 데이터베이스 연결을 사용하여 테스트 및 연결

- 4) 새 연결을 테스트합니다.

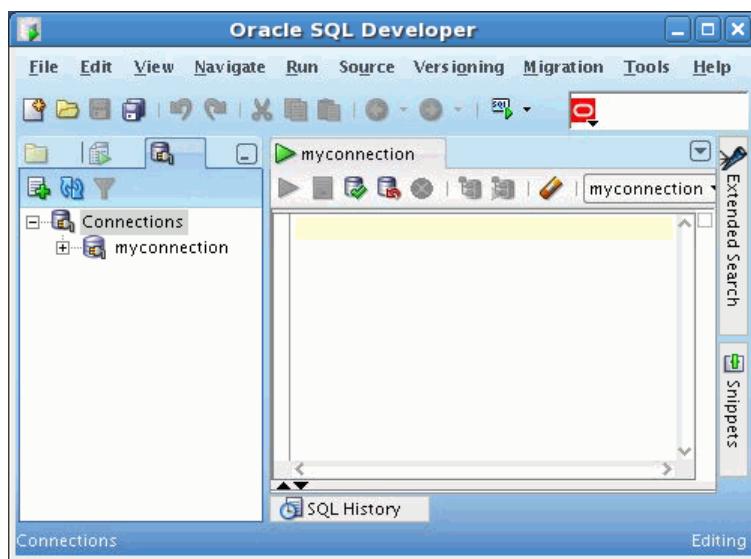


- 5) 상태가 Success이면 새로운 이 연결을 사용하여 데이터베이스에 연결합니다.



## 연습 I-1 해답: 소개 (계속)

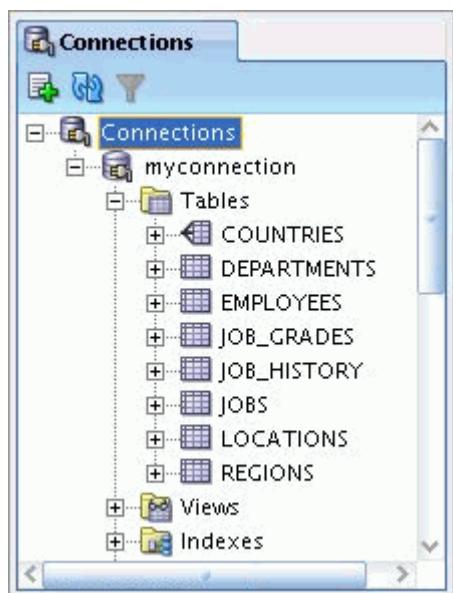
연결을 생성하면 해당 연결에 대한 SQL Worksheet가 자동으로 열립니다.



### Connections Navigator에서 테이블 탐색

- 6) Connections Navigator의 Tables 노드에서 사용할 수 있는 객체를 확인합니다. 다음 테이블이 있는지 확인합니다.

```
COUNTRIES
DEPARTMENTS
EMPLOYEES
JOB_GRADES
JOB_HISTORY
JOBS
LOCATIONS
REGIONS
```



## 연습 I-1 해답: 소개(계속)

7) EMPLOYEES 테이블의 구조를 살펴봅니다.

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Tables' tree view has 'EMPLOYEES' selected. The main panel displays the 'EMPLOYEES' table structure under the 'myconnection' connection. The 'Columns' tab is active, showing the following columns:

Column Name	Data Type	Nullable	Default Value
EMPLOYEE_ID	NUMBER(6,0)	No	(null)
FIRST_NAME	VARCHAR2(20 BYTE)	Yes	(null)
LAST_NAME	VARCHAR2(25 BYTE)	No	(null)
EMAIL	VARCHAR2(25 BYTE)	No	(null)
PHONE_NUMBER	VARCHAR2(20 BYTE)	Yes	(null)
HIRE_DATE	DATE	No	(null)
JOB_ID	VARCHAR2(10 BYTE)	No	(null)
SALARY	NUMBER(8,2)	Yes	(null)
COMMISSION_PCT	NUMBER(2,2)	No	(null)
MANAGER_ID	NUMBER(6,0)	No	(null)
DEPARTMENT_ID	NUMBER(3,0)	No	(null)

8) DEPARTMENTS 테이블의 데이터를 확인합니다.

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Tables' tree view has 'DEPARTMENTS' selected. The main panel displays the 'DEPARTMENTS' table data under the 'myconnection' connection. The 'Data' tab is active, showing the following data:

DEPARTMENT_ID	DEPARTMENT_NAME
1	10 Administration
2	20 Marketing
3	30 Shipping
4	40 IT
5	50 Sales
6	60 Executive
7	70 Accounting
8	80 Contracting

## 단원 1의 연습

이 연습에서는 간단한 SELECT query를 작성합니다. 이러한 query에는 이 단원에서 학습한 대부분의 SELECT 절과 연산이 포함됩니다.

## 연습 1-1: SQL SELECT 문을 사용하여 데이터 검색

### 1부

지식을 테스트해 보십시오.

- 1) 다음 SELECT 문은 성공적으로 실행됩니다.

```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```

맞음/틀림

- 2) 다음 SELECT 문은 성공적으로 실행됩니다.

```
SELECT *
FROM   job_grades;
```

맞음/틀림

- 3) 다음 명령문에 네 개의 코딩 오류가 있습니다. 식별할 수 있습니까?

```
SELECT      employee_id, last_name
sal x 12  ANNUAL SALARY
FROM       employees;
```

### 2부

연습을 시작하기 전에 다음과 같은 점에 주의합니다.

- 모든 연습 파일을 /home/oracle/labs/sql11/labs에 저장하십시오.
- SQL Worksheet에 SQL 문을 입력합니다. SQL Developer에서 스크립트를 저장하려면 필요한 SQL Worksheet가 활성화되어 있는지 확인한 다음 File 메뉴에서 Save As를 선택하여 SQL 문을 lab\_<lessonno>\_<stepno>.sql 스크립트로 저장합니다. 기존 스크립트를 수정하는 경우 Save As를 사용하여 다른 파일 이름으로 스크립트를 저장해야 합니다.
- query를 실행하려면 SQL Worksheet에서 Execute Statement 아이콘을 누릅니다. 또는 [F9]를 누를 수 있습니다. DML 및 DDL 문의 경우에는 Run Script 아이콘을 사용하거나 [F5]를 누릅니다.
- query를 실행한 후 동일한 워크시트에 다음 query를 입력하지 않도록 합니다. 새 워크시트를 엽니다.

## 연습 1-1: SQL SELECT 문을 사용하여 데이터 검색 (계속)

여러분은 Acme Corporation의 SQL 프로그래머로 채용되었습니다. 첫 작업은 Human Resources 테이블의 데이터를 기반으로 몇 가지 보고서를 작성하는 것입니다.

- 4) 첫 작업은 DEPARTMENTS 테이블의 구조와 해당 내용을 파악하는 것입니다.

Name	Null	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

4 rows selected

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	Administration	200	1700
2	Marketing	201	1800
3	Shipping	124	1500
4	IT	103	1400
5	Sales	149	2500
6	Executive	100	1700
7	Accounting	205	1700
8	Contracting	(null)	1700

- 5) EMPLOYEES 테이블의 구조를 확인합니다.

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8, 2)
COMMISSION_PCT		NUMBER(2, 2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

11 rows selected

HR 부서에서 사원 ID가 먼저 나타나고 이어서 각 사원에 대한 성, 직무 ID, 채용 날짜 및 사원 ID를 표시하는 query를 요구합니다. HIRE\_DATE 열에 대한 alias로 STARTDATE를 입력하십시오. 작성한 SQL 문을 HR 부서에 전달할 수 있도록 lab\_01\_05.sql이라는 파일에 저장합니다.

## 연습 1-1: SQL SELECT 문을 사용하여 데이터 검색 (계속)

- 6) lab\_01\_05.sql 파일의 query가 제대로 실행되는지 테스트합니다.

참고: query를 실행한 후 동일한 워크시트에 다음 query를 입력하지 않도록 합니다. 새 워크시트를 엽니다.

	EMPLOYEE_ID	LAST_NAME	JOB_ID	STARTDATE
1	200	Whalen	AD_ASST	17-SEP-87
2	201	Hartstein	MK_MAN	17-FEB-96
3	202	Fay	MK_REP	17-AUG-97
4	205	Higgins	AC_MGR	07-JUN-94
5	206	Gietz	AC_ACCOUNT	07-JUN-94

...

19	176	Taylor	SA_REP	24-MAR-98
20	178	Grant	SA_REP	24-MAY-99

- 7) HR 부서에서 EMPLOYEES 테이블의 모든 고유 직무 ID를 표시하는 query를 요구합니다.

	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD_PRES
5	AD_VP
6	IT_PROG
7	MK_MAN
8	MK_REP
9	SA_MAN
10	SA_REP
11	ST_CLERK
12	ST_MAN

### 3부

시간 여유가 있을 경우 다음 연습을 완료하십시오.

- 8) HR 부서에서 보고에 적합하도록 열 머리글의 사원 정보를 쉽게 표시해 달라고 합니다. lab\_01\_05.sql의 명령문을 새 SQL Worksheet로 복사합니다. 열 머리글의 이름을 각각 Emp #, Employee, Job, Hire Date로 지정합니다. 그런 다음 query를 다시 실행합니다.

	Emp #	Employee	Job	Hire Date
1	200	Whalen	AD_ASST	17-SEP-87
2	201	Hartstein	MK_MAN	17-FEB-96
3	202	Fay	MK_REP	17-AUG-97
4	205	Higgins	AC_MGR	07-JUN-94
5	206	Gietz	AC_ACCOUNT	07-JUN-94

...

## 연습 1-1: SQL SELECT 문을 사용하여 데이터 검색 (계속)

19	176 Taylor	SA_REP	24-MAR-98
20	178 Grant	SA_REP	24-MAY-99

- 9) HR 부서에서 모든 사원과 그들의 직무 ID에 대한 보고서를 요청했습니다. 성과 직무 ID를 이어서 표시하고(쉼표와 공백으로 구분) 열 이름을 Employee and Title로 지정합니다.

Employee and Title	
1	Abel, SA_REP
2	Davies, ST_CLERK
3	De Haan, AD_VP
4	Ernst, IT_PROG
5	Fay, MK_REP

...

19	Whalen, AD_ASST
20	Zlotkey, SA_MAN

다른 작업을 수행하려면 다음 연습을 완료하십시오.

- 10) EMPLOYEES 테이블의 데이터에 익숙해지도록 해당 테이블의 모든 데이터를 표시하는 query를 생성합니다. 각 열 출력은 쉼표로 구분합니다. 열 제목을 THE\_OUTPUT으로 지정합니다.

THE_OUTPUT	
1	200,Jennifer,Whalen,JWHALEN,515.123.4444,AD_ASST,101,17-SEP-87,4400,,10
2	201,Michael,Hartstein,MHARTSTE,515.123.5555,MK_MAN,100,17-FEB-96,13000,,20
3	202,Pat,Fay,PFAY,603.123.6666,MK_REP,201,17-AUG-97,6000,,20
4	205,Shelley,Higgins,SHIGGINS,515.123.8080,AC_MGR,101,07-JUN-94,12000,,110
5	206,William,Gietz,WGIETZ,515.123.8181,AC_ACCOUNT,205,07-JUN-94,8300,,110

...

19	176,Jonathon,Taylor,JTAYLOR,011.44.1644.429265,SA_REP,149,24-MAR-98,8600,.2,80
20	178,Kimberely,Grant,KGRANT,011.44.1644.429263,SA_REP,149,24-MAY-99,7000,.15,

## 연습 1-1 해답: SQL SELECT 문을 사용하여 데이터 검색

### 1부

지식을 테스트해 보십시오.

- 1) 다음 SELECT 문은 성공적으로 실행됩니다.

```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```

맞음/틀림

- 2) 다음 SELECT 문은 성공적으로 실행됩니다.

```
SELECT *
FROM   job_grades;
```

맞음/틀림

- 3) 다음 명령문에 네 개의 코딩 오류가 있습니다. 식별할 수 있습니까?

```
SELECT      employee_id, last_name
sal x 12  ANNUAL SALARY
FROM       employees;
```

- EMPLOYEES 테이블에 sal이라는 열이 없습니다. 열 이름은 SALARY입니다.
- 두번째 행의 곱하기 연산자는 x가 아니라 \*입니다.
- ANNUAL SALARY alias에 공백을 포함할 수 없습니다. alias는 ANNUAL\_SALARY로 표기하거나 큰 따옴표로 묶어야 합니다.
- LAST\_NAME 열 뒤에 쉼표가 누락되었습니다.

## 연습 1-1 해답: SQL SELECT 문을 사용하여 데이터 검색 (계속)

### 2부

여러분은 Acme Corporation의 SQL 프로그래머로 채용되었습니다. 첫 작업은 Human Resources 테이블의 데이터를 기반으로 몇 가지 보고서를 작성하는 것입니다.

- 4) 첫 작업은 DEPARTMENTS 테이블의 구조와 해당 내용을 파악하는 것입니다.

- a. DEPARTMENTS 테이블의 구조를 파악하려면 다음 명령문을 사용합니다.

```
DESCRIBE departments
```

- b. DEPARTMENTS 테이블에 포함된 데이터를 확인하려면 다음 명령문을 사용합니다.

```
SELECT *
FROM   departments;
```

- 5) EMPLOYEES 테이블의 구조를 확인합니다.

```
DESCRIBE employees
```

HR 부서에서 사원 ID가 먼저 나타나고 이어서 각 사원에 대한 성, 직무 ID, 채용 날짜 및 사원 ID를 표시하는 query를 요구합니다. HIRE\_DATE 열에 대한 alias로 STARTDATE를 입력하십시오. 작성한 SQL 문을 HR 부서에 전달할 수 있도록 lab\_01\_05.sql이라는 파일에 저장합니다.

```
SELECT employee_id, last_name, job_id, hire_date StartDate
FROM   employees;
```

- 6) lab\_01\_05.sql 파일의 query가 제대로 실행되는지 테스트합니다.

```
SELECT employee_id, last_name, job_id, hire_date StartDate
FROM   employees;
```

- 7) HR 부서에서 EMPLOYEES 테이블의 모든 고유 직무 ID를 표시하는 query를 요구합니다.

```
SELECT DISTINCT job_id
FROM   employees;
```

## 연습 1-1 해답: SQL SELECT 문을 사용하여 데이터 검색 (계속)

### 3부

시간 여유가 있을 경우 다음 연습을 완료하십시오.

- 8) HR 부서에서 보고에 적합하도록 열 머리글의 사원 정보를 쉽게 표시해 달라고 합니다. lab\_01\_05.sql의 명령문을 새 SQL Worksheet로 복사합니다. 열 머리글의 이름을 각각 Emp #, Employee, Job, Hire Date로 지정합니다. 그런 다음 query를 다시 실행합니다.

```
SELECT employee_id "Emp #", last_name "Employee",
       job_id "Job", hire_date "Hire Date"
  FROM employees;
```

- 9) HR 부서에서 모든 사원과 그들의 직무 ID에 대한 보고서를 요청했습니다. 성과 직무 ID를 이어서 표시하고(쉼표와 공백으로 구분) 열 이름을 Employee and Title로 지정합니다.

```
SELECT last_name||', '||job_id "Employee and Title"
  FROM employees;
```

다른 작업을 수행하려면 다음 연습을 완료하십시오.

- 10) EMPLOYEES 테이블의 데이터에 익숙해지도록 해당 테이블의 모든 데이터를 표시하는 query를 생성합니다. 각 열 출력은 쉼표로 구분합니다. 열 제목을 THE\_OUTPUT으로 지정합니다.

```
SELECT employee_id || ',' || first_name || ',' || last_name
      || ',' || email || ',' || phone_number || ',' || job_id
      || ',' || manager_id || ',' || hire_date || ',' ||
      || salary || ',' || commission_pct || ',' ||
department_id
      THE_OUTPUT
  FROM employees;
```

## 단원 2의 연습

이 연습에서는 WHERE 절과 ORDER BY 절을 사용하는 명령문을 비롯하여 추가 보고서를 작성하는 과정을 다룹니다. 앤퍼샌드 치환을 포함시키면 SQL 문의 재사용 및 범용성을 높일 수 있습니다.

## 연습 2-1: 데이터 제한 및 정렬

HR 부서에서 몇 가지 query 작성과 관련해 여러분의 도움을 요청합니다.

- 1) HR 부서에서 예산 문제 때문에 급여가 \$12,000가 넘는 사원의 성과 급여를 표시하는 보고서가 필요합니다. 작성한 SQL 문을 lab\_02\_01.sql이라는 텍스트 파일로 저장합니다. query를 실행합니다.

	LAST_NAME	SALARY
1	Hartstein	13000
2	King	24000
3	Kochhar	17000
4	De Haan	17000

- 2) 새 SQL Worksheet를 엽니다. 사원 번호 176의 성과 부서 ID를 표시하는 보고서를 작성합니다. query를 실행합니다.

	LAST_NAME	DEPARTMENT_ID
1	Taylor	80

- 3) HR 부서에서 급여가 높은 사원과 급여가 낮은 사원을 찾으려고 합니다. 급여가 \$5,000 ~ \$12,000 범위에 속하지 않는 사원의 성과 급여를 표시하도록 lab\_02\_01.sql을 수정합니다. 작성한 SQL 문을 lab\_02\_03.sql이라는 텍스트 파일로 저장합니다.

	LAST_NAME	SALARY
1	Whalen	4400
2	Hartstein	13000
3	King	24000
4	Kochhar	17000
5	De Haan	17000
6	Lorentz	4200
7	Rajs	3500
8	Davies	3100
9	Matos	2600
10	Vargas	2500

- 4) Matos 및 Taylor라는 성을 가진 사원에 대해 성, 직무 ID, 채용 날짜를 표시하는 보고서를 작성합니다. 채용 날짜를 기준으로 오름차순으로 query를 정렬합니다.

	LAST_NAME	JOB_ID	HIRE_DATE
1	Matos	ST_CLERK	15-MAR-98
2	Taylor	SA_REP	24-MAR-98

## 연습 2-1: 데이터 제한 및 정렬 (계속)

- 5) 부서 20 또는 50에 속하는 모든 사원의 성과 부서 ID를 이름별로 오름차순으로 정렬하여 표시합니다.

LAST_NAME	DEPARTMENT_ID
1 Davies	50
2 Fay	20
3 Hartstein	20
4 Matos	50
5 Mourgos	50
6 Rajs	50
7 Vargas	50

- 6) \$5,000 ~ \$12,000의 급여를 받고 부서 20 또는 50에 속하는 사원의 성과 급여를 표시하도록 lab\_02\_03.sql을 수정합니다. 열 레이블을 각각 Employee 및 Monthly Salary로 지정합니다. lab\_02\_03.sql을 lab\_02\_06.sql로 다시 저장합니다. lab\_02\_06.sql의 명령문을 실행합니다.

Employee	Monthly Salary
1 Fay	6000
2 Mourgos	5800

- 7) HR 부서에서 1994년에 채용된 모든 사원의 성과 채용 날짜를 표시하는 보고서를 요구합니다.

LAST_NAME	HIRE_DATE
1 Higgins	07-JUN-94
2 Gietz	07-JUN-94

- 8) 담당 관리자가 없는 모든 사원의 성과 직책을 표시하는 보고서를 작성합니다.

LAST_NAME	JOB_ID
1 King	AD_PRES

- 9) 커미션을 받는 모든 사원의 성, 급여 및 커미션을 표시하는 보고서를 작성합니다. 급여 및 커미션의 내림차순으로 데이터를 정렬합니다.  
ORDER BY 절에서 열의 숫자 위치를 사용합니다.

LAST_NAME	SALARY	COMMISSION_PCT
1 Abel	11000	0.3
2 Zlotkey	10500	0.2
3 Taylor	8600	0.2
4 Grant	7000	0.15

## 연습 2-1: 데이터 제한 및 정렬 (계속)

10) HR 부서의 멤버는 여러분이 작성 중인 query에 유연성이 확대되기를 원합니다.

그들은 유저가 프롬프트에 지정하는 액수보다 많은 급여를 받은 사원이 있을 경우 이들의 성과 급여를 표시하는 보고서를 기대합니다. 이 query를 lab\_02\_10.sql이라는 파일에 저장합니다. 프롬프트가 표시되었을 때 12000을 입력하면 보고서에 다음 결과가 표시됩니다.

	LAST_NAME	SALARY
1	Hartstein	13000
2	King	24000
3	Kochhar	17000
4	De Haan	17000

11) HR 부서에서 관리자를 기준으로 보고서를 실행하려고 합니다. 유저에게 관리자 ID 입력 프롬프트를 표시하고 해당 관리자에 속한 사원의 사원 ID, 성, 급여 및 부서를 생성하는 query를 작성합니다. HR 부서에서 선택한 열을 기준으로 보고서를 정렬하는 기능을 원합니다. 다음 값으로 데이터를 테스트할 수 있습니다.

manager\_id = 103, last\_name을 기준으로 정렬:

	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	104	Ernst	6000	60
2	107	Lorentz	4200	60

manager\_id = 201, salary를 기준으로 정렬:

	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	202	Fay	6000	20

manager\_id = 124, employee\_id를 기준으로 정렬:

	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	141	Rajs	3500	50
2	142	Davies	3100	50
3	143	Matos	2600	50
4	144	Vargas	2500	50

시간 여유가 있을 경우 다음 연습을 완료하십시오.

12) 성의 세번째 문자가 "a"인 모든 사원의 성을 표시합니다.

	LAST_NAME
1	Grant
2	Whalen

## 연습 2-1: DML 제한 및 정렬 (계속)

- 13) 성에 "a"와 "e"가 모두 포함된 모든 사원의 성을 표시합니다.

LAST_NAME
1 Davies
2 De Haan
3 Hartstein
4 Whalen

다른 작업을 수행하려면 다음 연습을 완료합니다.

- 14) 직무가 판매 사원이나 자재 담당자이고 급여가 \$2,500, \$3,500 또는 \$7,000가 아닌 모든 사원의 성, 직무 및 급여를 표시합니다.

LAST_NAME	JOB_ID	SALARY
1 Abel	SA_REP	11000
2 Taylor	SA_REP	8600
3 Davies	ST_CLERK	3100
4 Matos	ST_CLERK	2600

- 15) 커미션이 20%인 모든 사원의 성, 급여 및 커미션을 표시하도록 lab\_02\_06.sql을 수정합니다. lab\_02\_06.sql을 lab\_02\_15.sql로 다시 저장합니다. lab\_02\_15.sql의 명령문을 다시 실행합니다.

Employee	Monthly Salary	COMMISSION_PCT
1 Zlotkey	10500	0.2
2 Taylor	8600	0.2

## 연습 2-1 해답: 데이터 제한 및 정렬

HR 부서에서 몇 가지 query 작성과 관련해 여러분의 도움을 요청합니다.

- 1) HR 부서에서 예산 문제 때문에 급여가 \$12,000가 넘는 사원의 성과 급여를 표시하는 보고서가 필요합니다. 작성한 SQL 문을 lab\_02\_01.sql이라는 파일로 저장합니다. query를 실행합니다.

```
SELECT last_name, salary
FROM employees
WHERE salary > 12000;
```

- 2) 새 SQL Worksheet를 엽니다. 사원 번호 176의 성과 부서 ID를 표시하는 보고서를 작성합니다.

```
SELECT last_name, department_id
FROM employees
WHERE employee_id = 176;
```

- 3) HR 부서에서 급여가 높은 사원과 급여가 낮은 사원을 찾으려고 합니다. 급여가 \$5,000에서 \$12,000의 범위에 속하지 않는 모든 사원의 성 및 급여를 표시하도록 lab\_02\_01.sql을 수정합니다. 작성한 SQL 문을 lab\_02\_03.sql로 저장합니다.

```
SELECT last_name, salary
FROM employees
WHERE salary NOT BETWEEN 5000 AND 12000;
```

- 4) Matos 및 Taylor라는 성을 가진 사원에 대해 성, 직무 ID, 채용 날짜를 표시하는 보고서를 작성합니다. 채용 날짜를 기준으로 오름차순으로 query를 정렬합니다.

```
SELECT last_name, job_id, hire_date
FROM employees
WHERE last_name IN ('Matos', 'Taylor')
ORDER BY hire_date;
```

- 5) 부서 20 또는 50에 속하는 모든 사원의 성과 부서 ID를 이름별로 오름차순으로 정렬하여 표시합니다.

```
SELECT last_name, department_id
FROM employees
WHERE department_id IN (20, 50)
ORDER BY last_name ASC;
```

- 6) \$5,000 ~ \$12,000의 급여를 받고 부서 20 또는 50에 속하는 사원의 성과 급여를 나열하도록 lab\_02\_03.sql을 수정합니다. 열 레이블을 각각 Employee 및 Monthly Salary로 지정합니다. lab\_02\_03.sql을 lab\_02\_06.sql로 다시 저장합니다. lab\_02\_06.sql의 명령문을 실행합니다.

```
SELECT last_name "Employee", salary "Monthly Salary"
FROM employees
WHERE salary BETWEEN 5000 AND 12000
AND department_id IN (20, 50);
```

## 연습 2-1 해답: 데이터 제한 및 정렬 (계속)

- 7) HR 부서에서 1994년에 채용된 모든 사원의 성과 채용 날짜를 표시하는 보고서를 요구합니다.

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date LIKE '%94';
```

- 8) 담당 관리자가 없는 모든 사원의 성과 직책을 표시하는 보고서를 작성합니다.

```
SELECT last_name, job_id
FROM employees
WHERE manager_id IS NULL;
```

- 9) 커미션을 받는 모든 사원의 성, 급여 및 커미션을 표시하는 보고서를 작성합니다. 급여 및 커미션의 내림차순으로 데이터를 정렬합니다. ORDER BY 절에서 열의 숫자 위치를 사용합니다.

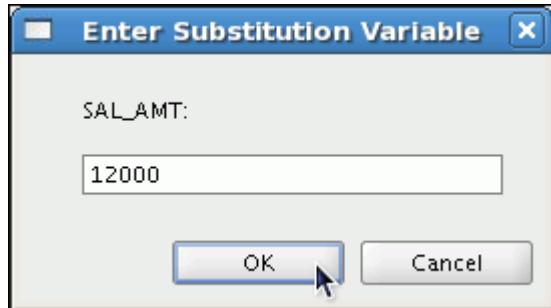
```
SELECT last_name, salary, commission_pct
FROM employees
WHERE commission_pct IS NOT NULL
ORDER BY 2 DESC, 3 DESC;
```

- 10) HR 부서의 멤버는 여러분이 작성 중인 query에 유연성이 확대되기를 원합니다.

그들은 유저가 프롬프트에 지정하는 액수보다 많은 급여를 받은 사원이 있을 경우 이들의 성과 급여를 표시하는 보고서를 기대합니다. (연습 1에서 생성한 query를 수정하여 사용할 수 있습니다.) 이 query를 lab\_02\_10.sql이라는 파일에 저장합니다.

```
SELECT last_name, salary
FROM employees
WHERE salary > &sal_amt;
```

프롬프트가 나타나면 대화상자에 값으로 12000을 입력합니다. OK를 누릅니다.



- 11) HR 부서에서 관리자를 기준으로 보고서를 실행하려고 합니다. 유저에게 관리자 ID 입력 프롬프트를 표시하고 해당 관리자에 속한 사원의 사원 ID, 성, 급여 및 부서를 생성하는 query를 작성합니다. HR 부서에서 선택한 열을 기준으로 보고서를 정렬하는 기능을 원합니다. 다음 값으로 데이터를 테스트할 수 있습니다.

manager\_id = 103, last\_name을 기준으로 정렬

manager\_id = 201, salary를 기준으로 정렬

manager\_id = 124, employee\_id를 기준으로 정렬

## 연습 2-1 해답: 데이터 제한 및 정렬 (계속)

```
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE manager_id = &mgr_num
ORDER BY &order_col;
```

시간 여유가 있을 경우 다음 연습을 완료하십시오.

- 12) 성의 세번째 문자가 "a"인 모든 사원의 성을 표시합니다.

```
SELECT last_name
FROM employees
WHERE last_name LIKE '__a%';
```

- 13) 성에 "a"와 "e"가 모두 포함된 모든 사원의 성을 표시합니다.

```
SELECT last_name
FROM employees
WHERE last_name LIKE '%a%'
AND last_name LIKE '%e%';
```

다른 작업을 수행하려면 다음 연습을 완료합니다.

- 14) 직무가 판매 사원이나 자재 담당자이고 급여가 \$2,500, \$3,500 또는 \$7,000가 아닌 모든 사원의 성, 직무 및 급여를 표시합니다.

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id IN ('SA_REP', 'ST_CLERK')
AND salary NOT IN (2500, 3500, 7000);
```

- 15) 커미션 금액이 20%인 모든 사원의 성, 급여 및 커미션을 표시하도록

`lab_02_06.sql`을 수정합니다. `lab_02_06.sql`을 `lab_02_15.sql`로 다시 저장합니다. `lab_02_15.sql`의 명령문을 다시 실행합니다.

```
SELECT last_name "Employee", salary "Monthly Salary",
commission_pct
FROM employees
WHERE commission_pct = .20;
```

## 단원 3의 연습

이 연습에서는 문자, 숫자 및 날짜 데이터 유형에 사용할 수 있는 다양한 함수의 사용법을 익힙니다.

### 연습 3-1: 단일 행 함수를 사용하여 출력 커스터마이즈

- 1) 시스템 날짜를 표시하기 위한 query를 작성합니다. 열 레이블을 Date로 지정합니다.

**참고:** 시간대가 다른 지역에 데이터베이스가 있는 경우 해당 데이터베이스가 상주하는 운영 체제의 날짜가 출력됩니다.

	Date
1	10-JUN-09

- 2) HR 부서에서 각 사원에 대해 사원 번호, 성, 급여 및 15.5% 인상된 급여(정수로 표현)를 표시하는 보고서가 필요합니다. 열 레이블을 New Salary로 지정합니다. 작성한 SQL 문을 lab\_03\_02.sql이라는 파일에 저장합니다.
- 3) lab\_03\_02.sql 파일의 query를 실행합니다.

	EMPLOYEE_ID	LAST_NAME	SALARY	New Salary
1	200	Whalen	4400	5082
2	201	Hartstein	13000	15015
3	202	Fay	6000	6930
4	205	Higgins	12000	13860
5	206	Gietz	8300	9587

...

19	176	Taylor	8600	9933
20	178	Grant	7000	8085

- 4) 새 급여에서 이전 급여를 뺀 차액을 추가하도록 lab\_03\_02.sql의 query를 수정합니다. 열 레이블을 Increase로 지정합니다. 파일 내용을 lab\_03\_04.sql로 저장합니다. 수정한 query를 실행합니다.

	EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
1	200	Whalen	4400	5082	682
2	201	Hartstein	13000	15015	2015
3	202	Fay	6000	6930	930
4	205	Higgins	12000	13860	1860
5	206	Gietz	8300	9587	1287

...

19	176	Taylor	8600	9933	1333
20	178	Grant	7000	8085	1085

### 연습 3-1: 단일 행 함수를 사용하여 출력 커스터마이즈 (계속)

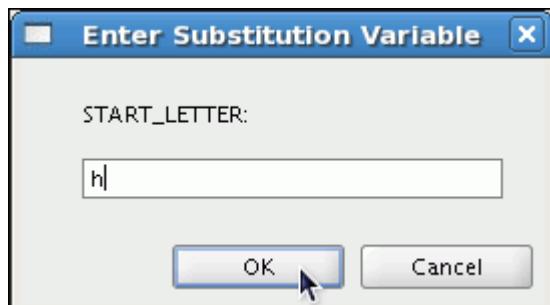
- 5) "J", "A" 또는 "M"으로 시작하는 이름을 가진 모든 사원의 성(첫번째 문자는 대문자, 나머지는 모두 소문자)과 성의 길이를 표시하는 query를 작성합니다. 각 열에 적절한 레이블을 지정합니다. 사원의 성을 기준으로 결과를 정렬합니다.

	Name	Length
1	Abel	4
2	Matos	5
3	Mourgos	7

유저에게 성의 첫 문자를 입력하는 프롬프트를 표시하도록 query를 재작성합니다. 예를 들어, 문자 입력 프롬프트가 표시되었을 때 유저가 "H"(대문자)를 입력하면 성이 "H"로 시작하는 모든 사원이 출력에 표시되어야 합니다.

	Name	Length
1	Hartstein	9
2	Higgins	7
3	Hunold	6

입력된 문자의 대소문자 여부에 따라 출력이 달라지지 않도록 쿼리를 수정합니다. 입력된 문자는 SELECT query에서 처리되기 전에 대문자로 변경해야 합니다.



	Name	Length
1	Hartstein	9
2	Higgins	7
3	Hunold	6

### 연습 3-1: 단일 행 함수를 사용하여 출력 커스터마이즈 (계속)

- 6) HR 부서에서 각 사원의 근속 기간을 파악하려고 합니다. 각 사원에 대해 성을 표시하고 채널 엔터테인먼트 오늘까지 경과한 개월 수를 계산합니다. 열 레이블을 MONTHS\_WORKED로 지정합니다. 재직 개월 수에 따라 결과를 정렬합니다. 개월 수를 가장 가까운 정수로 반올림합니다.

참고: 이 query는 실행된 날짜에 의존하므로 MONTHS\_WORKED 열의 값은 실행된 날짜에 따라 달라집니다.

	LAST_NAME	MONTHS_WORKED
1	Zlotkey	112
2	Mourgos	115
3	Grant	121
4	Lorentz	124
5	Vargas	131
...		
19	Whalen	261
20	King	264

시간 여유가 있을 경우 다음 연습을 완료하십시오.

- 7) 모든 사원의 성과 급여를 표시하기 위한 query를 작성합니다. 급여가 15자 길이로 표시되고 왼쪽에 \$ 기호가 채워지도록 형식을 지정합니다. 열 레이블을 SALARY로 지정합니다.

	LAST_NAME	SALARY
1	Whalen	\$\$\$\$\$\$\$\$\$\$\$\$\$\$4400
2	Hartstein	\$\$\$\$\$\$\$\$\$\$\$\$\$13000
3	Fay	\$\$\$\$\$\$\$\$\$\$\$\$\$6000
4	Higgins	\$\$\$\$\$\$\$\$\$\$\$\$\$12000
5	Gietz	\$\$\$\$\$\$\$\$\$\$\$\$\$8300
...		
19	Taylor	\$\$\$\$\$\$\$\$\$\$\$\$\$8600
20	Grant	\$\$\$\$\$\$\$\$\$\$\$\$\$7000

- 8) 사원의 성에서 처음 8자를 표시하고 급여 액수를 별표로 나타내는 query를 작성합니다. 각 별표는 \$1,000을 나타냅니다. 급여의 내림차순으로 데이터를 정렬합니다. 열 레이블을 EMPLOYEES\_AND\_THEIR\_SALARIES로 지정합니다.

	EMPLOYEES_AND_THEIR_SALARIES
1	King *****
2	Kochhar *****
3	De Haan *****
4	Hartstei *****
5	Higgins *****
...	
19	Matos ***
20	Vargas ***

### 연습 3-1: 단일 행 함수를 사용하여 출력 커스터마이즈 (계속)

- 9) 부서 90의 모든 사원에 대해 성 및 재직 기간(주 단위)을 표시하도록 query를 작성합니다. 주를 나타내는 숫자 열의 레이블을 TENURE로 지정합니다. 주를 나타내는 숫자 값을 소수점 왼쪽에서 truncate합니다. 직원 재직 기간의 내림차순으로 레코드를 표시합니다.

참고: TENURE 값은 query를 실행한 날짜에 의존하므로 실행한 날짜에 따라 달라집니다.

	LAST_NAME	TENURE
1	King	1147
2	Kochhar	1028
3	De Haan	856

## 연습 3-1 해답: 단일 행 함수를 사용하여 출력 커스터마이즈

- 1) 시스템 날짜를 표시하기 위한 query를 작성합니다. 열 레이블을 Date로 지정합니다.

**참고:** 시간대가 다른 지역에 데이터베이스가 있는 경우 해당 데이터베이스가 상주하는 운영 체제의 날짜가 출력됩니다.

```
SELECT sysdate "Date"
FROM   dual;
```

- 2) HR 부서에서 각 사원에 대해 사원 번호, 성, 급여 및 15.5% 인상된 급여(정수로 표현)를 표시하는 보고서가 필요합니다. 열 레이블을 New Salary로 지정합니다. 작성한 SQL 문을 lab\_03\_02.sql이라는 파일에 저장합니다.

```
SELECT employee_id, last_name, salary,
       ROUND(salary * 1.155, 0) "New Salary"
  FROM employees;
```

- 3) lab\_03\_02.sql 파일의 query를 실행합니다.

```
SELECT employee_id, last_name, salary,
       ROUND(salary * 1.155, 0) "New Salary"
  FROM employees;
```

- 4) 새 급여에서 이전 급여를 뺀 열을 추가하도록 lab\_03\_02.sql의 query를 수정합니다. 열 레이블을 Increase로 지정합니다. 파일 내용을 lab\_03\_04.sql로 저장합니다. 수정한 query를 실행합니다.

```
SELECT employee_id, last_name, salary,
       ROUND(salary * 1.155, 0) "New Salary",
       ROUND(salary * 1.155, 0) - salary "Increase"
  FROM employees;
```

- 5) "J", "A" 또는 "M"으로 시작하는 이름을 가진 모든 사원의 성(첫번째 문자는 대문자, 나머지는 모두 소문자)과 성의 길이를 표시하는 query를 작성합니다. 각 열에 적절한 레이블을 지정합니다. 사원의 성을 기준으로 결과를 정렬합니다.

```
SELECT INITCAP(last_name) "Name",
       LENGTH(last_name) "Length"
  FROM employees
 WHERE last_name LIKE 'J%'
   OR last_name LIKE 'M%'
   OR last_name LIKE 'A%'
 ORDER BY last_name ;
```

### 연습 3-1 해답: 단일 행 함수를 사용하여 출력 커스터마이즈 (계속)

유저에게 성의 첫 문자를 입력하는 프롬프트를 표시하도록 query를 재작성합니다. 예를 들어, 문자 입력 프롬프트가 표시되었을 때 유저가 H(대문자)를 입력하면 출력에 성이 "H"로 시작하는 모든 사원이 표시되어야 합니다.

```
SELECT INITCAP(last_name) "Name",
       LENGTH(last_name) "Length"
  FROM employees
 WHERE last_name LIKE '&start_letter%'
 ORDER BY last_name;
```

입력된 문자의 대소문자 여부에 따라 출력이 달라지지 않도록 쿼리를 수정합니다. 입력된 문자는 SELECT query에서 처리되기 전에 대문자로 변경해야 합니다.

```
SELECT INITCAP(last_name) "Name",
       LENGTH(last_name) "Length"
  FROM employees
 WHERE last_name LIKE UPPER('&start_letter%')
 ORDER BY last_name;
```

- 6) HR 부서에서 각 사원의 근속 기간을 파악하려고 합니다. 각 사원에 대해 성을 표시하고 채용일부터 오늘까지 경과한 개월 수를 계산합니다. 열 레이블을 MONTHS\_WORKED로 지정합니다. 재직 개월 수에 따라 결과를 정렬합니다. 개월 수를 가장 가까운 정수로 반올림합니다.

**참고:** 이 query는 실행된 날짜에 의존하므로 MONTHS\_WORKED 열의 값은 실행된 날짜에 따라 달라집니다.

```
SELECT last_name, ROUND(MONTHS_BETWEEN(
          SYSDATE, hire_date)) MONTHS_WORKED
  FROM employees
 ORDER BY months_worked;
```

시간 여유가 있을 경우 다음 연습을 완료하십시오.

- 7) 모든 사원의 성과 급여를 표시하기 위한 query를 작성합니다. 급여가 15자 길이로 표시되고 왼쪽에 \$ 기호가 채워지도록 형식을 지정합니다. 열 레이블을 SALARY로 지정합니다.

```
SELECT last_name,
       LPAD(salary, 15, '$') SALARY
  FROM employees;
```

### 연습 3-1 해답: 단일 행 함수를 사용하여 출력 커스터마이즈 (계속)

- 8) 사원의 성에서 처음 8자를 표시하고 급여 액수를 별표로 나타내는 query를 작성합니다. 각 별표는 \$1,000을 나타냅니다. 급여의 내림차순으로 데이터를 정렬합니다. 열 레이블을 EMPLOYEES\_AND THEIR\_SALARIES로 지정합니다.

```
SELECT rpad(last_name, 8)||' '||  
      rpad(' ', salary/1000+1, '*')  
          EMPLOYEES_AND_THEIR_SALARIES  
FROM   employees  
ORDER BY salary DESC;
```

- 9) 부서 90의 모든 사원에 대해 성 및 재직 기간(주 단위)을 표시하도록 query를 작성합니다. 주를 나타내는 숫자 열의 레이블을 TENURE로 지정합니다. 주를 나타내는 숫자 값을 소수점 왼쪽에서 truncate합니다. 직원 재직 기간의 내림차순으로 레코드를 표시합니다.

**참고:** TENURE 값은 query를 실행한 날짜에 따라 달라집니다.

```
SELECT last_name, trunc((SYSDATE-hire_date)/7) AS TENURE  
FROM   employees  
WHERE  department_id = 90  
ORDER BY TENURE DESC
```

## 단원 4의 연습

이 연습에서는 TO\_CHAR 및 TO\_DATE 함수와 DECODE 및 CASE와 같은 조건부 표현식을 사용하는 다양한 실습을 제공합니다. 중첩 함수의 경우 결과는 가장 안쪽 함수에서 가장 바깥쪽 함수로 평가됩니다.

## 연습 4-1: 변환 함수 및 조건부 표현식 사용

- 1) 각 사원에 대해 다음과 같이 출력하는 보고서를 작성합니다.

<employee last name> earns <salary> monthly but wants <3 times salary.> 열 레이블을 Dream Salaries로 지정합니다.

	Dream Salaries
1	Whalen earns \$4,400.00 monthly but wants \$13,200.00.
2	Hartstein earns \$13,000.00 monthly but wants \$39,000.00.
3	Fay earns \$6,000.00 monthly but wants \$18,000.00.
4	Higgins earns \$12,000.00 monthly but wants \$36,000.00.
5	Gietz earns \$8,300.00 monthly but wants \$24,900.00.
...	
19	Taylor earns \$8,600.00 monthly but wants \$25,800.00.
20	Grant earns \$7,000.00 monthly but wants \$21,000.00.

- 2) 각 사원의 성, 채용 날짜 및 근무 6개월 후 첫번째 월요일에 해당하는 급여 심의 날짜를 표시합니다. 열 레이블을 REVIEW로 지정합니다. 날짜 형식을 "Monday, the Thirty-First of July, 2000"과 유사한 형식으로 지정합니다.

	LAST_NAME	HIRE_DATE	REVIEW
1	Whalen	17-SEP-87	Monday, the Twenty-First of March, 1988
2	Hartstein	17-FEB-96	Monday, the Nineteenth of August, 1996
3	Fay	17-AUG-97	Monday, the Twenty-Third of February, 1998
4	Higgins	07-JUN-94	Monday, the Twelfth of December, 1994
5	Gietz	07-JUN-94	Monday, the Twelfth of December, 1994
...			
19	Taylor	24-MAR-98	Monday, the Twenty-Eighth of September, 1998
20	Grant	24-MAY-99	Monday, the Twenty-Ninth of November, 1999

- 3) 사원의 성, 채용 날짜, 근무 시작 요일을 표시합니다. 열 레이블을 DAY로 지정합니다. 월요일부터 시작하여曜일순으로 결과를 정렬합니다.

	LAST_NAME	HIRE_DATE	DAY
1	Grant	24-MAY-99	MONDAY
2	Ernst	21-MAY-91	TUESDAY
3	Taylor	24-MAR-98	TUESDAY
4	Rajs	17-OCT-95	TUESDAY
5	Mourgos	16-NOV-99	TUESDAY
...			
19	Matos	15-MAR-98	SUNDAY
20	Fay	17-AUG-97	SUNDAY

## 연습 4-1: 변환 함수 및 조건부 표현식 사용 (계속)

- 4) 사원의 성과 커미션 금액을 표시하는 query를 작성합니다. 사원이 커미션을 받지 않으면 "No Commission"을 표시합니다. 열 레이블을 COMM으로 지정합니다.

	LAST_NAME	COMM
1	Whalen	No Commission
2	Hartstein	No Commission
3	Fay	No Commission
4	Higgins	No Commission
5	Gietz	No Commission

...

16	Vargas	No Commission
17	Zlotkey	.2
18	Abel	.3
19	Taylor	.2
20	Grant	.15

시간 여유가 있을 경우 다음 연습을 완료하십시오.

- 5) 다음 데이터를 사용하여 DECODE 함수를 통해 JOB\_ID 열의 값을 기반으로 모든 사원의 등급을 표시하는 query를 작성합니다.

직책	등급
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA REP	D
ST_CLERK	E
None of the above	0

JOB_ID	GRADE
1 AC_ACCOUNT	0
2 AC_MGR	0
3 AD_ASST	0
4 AD_PRES	A
5 AD_VP	0
6 AD_VP	0
7 IT_PROG	C

...

14	SA REP	D
15	SA REP	D

...

19	ST_CLERK	E
20	ST_MAN	B

## 연습 4-1: 변환 함수 및 조건부 표현식 사용 (계속)

- 6) CASE 구문을 사용하여 앞의 연습에 나오는 명령문을 재작성합니다.

	JOB_ID	GRADE
1	AC_ACCOUNT	O
2	AC_MGR	O
3	AD_ASST	O
4	AD_PRES	A
5	AD_VP	O
6	AD_VP	O
7	IT_PROG	C
...		
14	SA_REP	D
15	SA_REP	D
...		
19	ST_CLERK	E
20	ST_MAN	B

## 연습 4-1 해답: 변환 함수 및 조건부 표현식 사용

- 1) 각 사원에 대해 다음과 같이 출력하는 보고서를 작성합니다.

<employee last name> earns <salary> monthly but wants <3 times salary> 열 레이블을 Dream Salaries로 지정합니다.

```
SELECT last_name || ' earns '
    || TO_CHAR(salary, 'fm$99,999.00')
    || ' monthly but wants '
    || TO_CHAR(salary * 3, 'fm$99,999.00')
    || '.' "Dream Salaries"
FROM employees;
```

- 2) 각 사원의 성, 채용 날짜 및 근무 6개월 후 첫번째 월요일에 해당하는 급여 십의 날짜를 표시합니다. 열 레이블을 REVIEW로 지정합니다. 날짜 형식을 "Monday, the Thirty-First of July, 2000"과 유사한 형식으로 지정합니다.

```
SELECT last_name, hire_date,
       TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'MONDAY'),
               'fmDay, "the" Ddspth "of" Month, YYYY') REVIEW
FROM employees;
```

- 3) 사원의 성, 채용 날짜, 근무 시작 요일을 표시합니다. 열 레이블을 DAY로 지정합니다. 월요일부터 시작하여曜일순으로 결과를 정렬합니다.

```
SELECT last_name, hire_date,
       TO_CHAR(hire_date, 'DAY') DAY
FROM employees
ORDER BY TO_CHAR(hire_date - 1, 'd');
```

- 4) 사원의 성과 커미션 금액을 표시하는 query를 작성합니다. 사원이 커미션을 받지 않으면 "No Commission"을 표시합니다. 열 레이블을 COMM으로 지정합니다.

```
SELECT last_name,
       NVL(TO_CHAR(commission_pct), 'No Commission') COMM
FROM employees;
```

- 5) 다음 데이터를 사용하여 DECODE 함수를 통해 JOB\_ID 열의 값을 기반으로 모든 사원의 등급을 표시하는 query를 작성합니다.

직책	등급
----	----

AD_PRES	A
ST_MAN	B
IT_PROG	C

## 연습 4-1 해답: 변환 함수 및 조건부 표현식 사용 (계속)

SA_REP	D
ST_CLERK	E
None of the above	0

```
SELECT job_id, decode (job_id,
                      'ST_CLERK',      'E',
                      'SA_REP',        'D',
                      'IT_PROG',       'C',
                      'ST_MAN',        'B',
                      'AD_PRES',       'A',
                      '0') GRADE
FROM employees;
```

- 6) CASE 구문을 사용하여 앞의 연습에 나오는 명령문을 재작성합니다.

```
SELECT job_id, CASE job_id
                  WHEN 'ST_CLERK' THEN 'E'
                  WHEN 'SA_REP'   THEN 'D'
                  WHEN 'IT_PROG'  THEN 'C'
                  WHEN 'ST_MAN'   THEN 'B'
                  WHEN 'AD_PRES'  THEN 'A'
                  ELSE '0' END GRADE
FROM employees;
```

## 단원 5의 연습

이 연습을 마친 후에는 그룹 함수 사용과 데이터 그룹 선택에 능숙해져야 합니다.

## 연습 5-1: 그룹 함수를 사용하여 집계 데이터 보고

다음 세 명령문의 유효성을 확인합니다. 맞음 또는 틀림에 O표 하십시오.

- 1) 그룹 함수는 다수 행에 대해 실행되어 그룹당 하나의 결과를 산출합니다.

맞음/틀림

- 2) 그룹 함수는 계산에 널을 포함시킵니다.

맞음/틀림

- 3) WHERE 절은 그룹 계산에 포함시키기 전에 행을 제한합니다.

맞음/틀림

HR 부서에서 다음 보고서를 요구합니다.

- 4) 모든 사원의 최고, 최저, 합계 및 평균 급여를 찾습니다. 열 레이블을 각각

Maximum, Minimum, Sum, Average로 지정합니다. 결과를 가장 가까운 정수로 반올림합니다. SQL 문을 lab\_05\_04.sql로 저장합니다. query를 실행합니다.

	Maximum	Minimum	Sum	Average
1	24000	2500	175500	8775

- 5) 각 직무 유형에 대해 최소, 최대, 합계 및 평균 급여를 표시하도록

lab\_05\_04.sql의 query를 수정합니다. lab\_05\_04.sql을

lab\_05\_05.sql로 다시 저장합니다. lab\_05\_05.sql의 명령문을 실행합니다.

JOB_ID	Maximum	Minimum	Sum	Average
1 AC_MGR	12000	12000	12000	12000
2 AC_ACCOUNT	8300	8300	8300	8300
3 IT_PROG	9000	4200	19200	6400
4 ST_MAN	5800	5800	5800	5800
5 AD_ASST	4400	4400	4400	4400
6 AD_VP	17000	17000	34000	17000
7 MK_MAN	13000	13000	13000	13000
8 SA_MAN	10500	10500	10500	10500
9 MK_REP	6000	6000	6000	6000
10 AD_PRES	24000	24000	24000	24000
11 SA_REP	11000	7000	26600	8867
12 ST_CLERK	3500	2500	11700	2925

## 연습 5-1: 그룹 함수를 사용하여 집계 데이터 보고 (계속)

- 6) 동일한 직무를 수행하는 사람 수를 표시하기 위한 query를 작성합니다.

JOB_ID	COUNT(*)
1 AC_ACCOUNT	1
2 AC_MGR	1
3 AD_ASST	1
4 AD_PRES	1
5 AD_VP	2
6 IT_PROG	3
7 MK_MAN	1
8 MK_REP	1
9 SA_MAN	1
10 SA_REP	3
11 ST_CLERK	4
12 ST_MAN	1

HR 부서의 유저에게 직무를 입력하는 프롬프트를 표시하도록 query를 일반화합니다. 이 스크립트를 lab\_05\_06.sql이라는 파일에 저장합니다. query를 실행합니다. 프롬프트가 나타나면 IT\_PROG를 입력합니다.

JOB_ID	COUNT(*)
1 IT_PROG	3

- 7) 관리자를 나열하지 않는 채로 관리자 수를 확인합니다. 열 레이블을 Number of Managers로 지정합니다.

힌트: MANAGER\_ID 열을 사용하여 관리자 수를 확인합니다.

Number of Managers
1 8

- 8) 최고 급여와 최저 급여의 차이를 알아냅니다. 열 레이블을 DIFFERENCE로 지정합니다.

DIFFERENCE
1 21500

시간 여유가 있을 경우 다음 연습을 완료하십시오.

- 9) 관리자 번호 및 해당 관리자의 부하 사원 중 최저 급여를 받는 사원의 급여를 표시하는 보고서를 작성합니다. 관리자가 알려져 있지 않은 모든 사원을 제외합니다. 최소 급여가 \$6,000 이하인 그룹을 제외시킵니다. 급여의 내림차순으로 출력을 정렬합니다.

MANAGER_ID	MIN(SALARY)
1 102	9000
2 205	8300
3 149	7000

## 연습 5-1: 그룹 함수를 사용하여 집계 데이터 보고 (계속)

다른 작업을 수행하려면 다음 연습을 완료합니다.

- 10) 사원의 총 수와 1995년, 1996년, 1997년, 1998년에 채용된 사원의 수를 표시하는 query를 작성합니다. 적절한 열 머리글을 지정하십시오.

	TOTAL	1995	1996	1997	1998
1	20	1	2	2	3

- 11) 부서 20, 50, 80 및 90에 대해 직무, 부서 ID별 해당 직무에 대한 급여 및 해당 직무에 대한 총 급여를 표시하고 각 열에 적절한 머리글을 지정하기 위한 행렬 query를 작성합니다.

	Job	Dept 20	Dept 50	Dept 80	Dept 90	Total
1	AC_MGR	(null)	(null)	(null)	(null)	12000
2	AC_ACCOUNT	(null)	(null)	(null)	(null)	8300
3	IT_PROG	(null)	(null)	(null)	(null)	19200
4	ST_MAN	(null)	5800	(null)	(null)	5800
5	AD_ASST	(null)	(null)	(null)	(null)	4400
6	AD_VP	(null)	(null)	(null)	34000	34000
7	MK_MAN	13000	(null)	(null)	(null)	13000
8	SA_MAN	(null)	(null)	10500	(null)	10500
9	MK_REP	6000	(null)	(null)	(null)	6000
10	AD_PRES	(null)	(null)	(null)	24000	24000
11	SA_REP	(null)	(null)	19600	(null)	26600
12	ST_CLERK	(null)	11700	(null)	(null)	11700

## 연습 5-1 해답: 그룹 함수를 사용하여 집계 데이터 보고

다음 세 명령문의 유효성을 확인합니다. 맞음 또는 틀림에 O표 하십시오.

- 1) 그룹 함수는 다수 행에 대해 실행되어 그룹당 하나의 결과를 산출합니다.

맞음/ 틀림

- 2) 그룹 함수는 계산에 널을 포함시킵니다.

맞음/ 틀림

- 3) WHERE 절은 그룹 계산에 포함시키기 전에 행을 제한합니다.

맞음/ 틀림

HR 부서에서 다음 보고서를 요구합니다.

- 4) 모든 사원의 최고, 최저, 합계 및 평균 급여를 찾습니다. 열 레이블을 각각 Maximum, Minimum, Sum, Average로 지정합니다. 결과를 가장 가까운 정수로 반올림합니다. SQL 문을 lab\_05\_04.sql로 저장합니다. query를 실행합니다.

```
SELECT ROUND(MAX(salary),0) "Maximum",
       ROUND(MIN(salary),0) "Minimum",
       ROUND(SUM(salary),0) "Sum",
       ROUND(AVG(salary),0) "Average"
  FROM   employees;
```

- 5) 각 직무 유형에 대해 최소, 최대, 합계 및 평균 급여를 표시하도록 lab\_05\_04.sql의 query를 수정합니다. lab\_05\_04.sql을 lab\_05\_05.sql로 다시 저장합니다. lab\_05\_05.sql의 명령문을 실행합니다.

```
SELECT job_id, ROUND(MAX(salary),0) "Maximum",
       ROUND(MIN(salary),0) "Minimum",
       ROUND(SUM(salary),0) "Sum",
       ROUND(AVG(salary),0) "Average"
  FROM   employees
 GROUP BY job_id;
```

- 6) 동일한 직무를 수행하는 사람 수를 표시하기 위한 query를 작성합니다.

```
SELECT job_id, COUNT(*)
  FROM   employees
 GROUP BY job_id;
```

HR 부서의 유저에게 직무를 입력하는 프롬프트를 표시하도록 query를 일반화합니다. 이 스크립트를 lab\_05\_06.sql이라는 파일에 저장합니다. query를 실행합니다. 프롬프트가 나타나면 IT\_PROG를 입력하고 OK를 누릅니다.

```
SELECT job_id, COUNT(*)
  FROM   employees
 WHERE  job_id = '&job_title'
 GROUP BY job_id;
```

## 연습 5-1 해답: 그룹 함수를 사용하여 집계 데이터 보고 (계속)

- 7) 관리자를 나열하지 않는 채로 관리자 수를 확인합니다. 열 레이블을 Number of Managers로 지정합니다.

**힌트:** MANAGER\_ID 열을 사용하여 관리자 수를 확인합니다.

```
SELECT COUNT(DISTINCT manager_id) "Number of Managers"
FROM employees;
```

- 8) 최고 급여와 최저 급여의 차이를 알아냅니다. 열 레이블을 DIFFERENCE로 지정합니다.

```
SELECT MAX(salary) - MIN(salary) DIFFERENCE
FROM employees;
```

시간 여유가 있을 경우 다음 연습을 완료하십시오.

- 9) 관리자 번호 및 해당 관리자의 부하 사원 중 최저 급여를 받는 사원의 급여를 표시하는 보고서를 작성합니다. 관리자가 알려져 있지 않은 모든 사원을 제외합니다. 최소 급여가 \$6,000 이하인 그룹을 제외시킵니다. 급여의 내림차순으로 출력을 정렬합니다.

```
SELECT manager_id, MIN(salary)
FROM employees
WHERE manager_id IS NOT NULL
GROUP BY manager_id
HAVING MIN(salary) > 6000
ORDER BY MIN(salary) DESC;
```

다른 작업을 수행하려면 다음 연습을 완료합니다.

- 10) 사원의 총 수와 1995년, 1996년, 1997년 및 1998년에 채용된 사원의 수를 표시하는 query를 작성합니다. 적절한 열 머리글을 지정하십시오.

```
SELECT COUNT(*) total,
       SUM(DECODE(TO_CHAR(hire_date,
'YYYY'), 1995, 1, 0)) "1995",
       SUM(DECODE(TO_CHAR(hire_date,
'YYYY'), 1996, 1, 0)) "1996",
       SUM(DECODE(TO_CHAR(hire_date,
'YYYY'), 1997, 1, 0)) "1997",
       SUM(DECODE(TO_CHAR(hire_date, 'YYYY'), 1998, 1, 0)) "1998"
  FROM employees;
```

- 11) 부서 20, 50, 80 및 90에 대해 직무, 부서 ID별 해당 직무에 대한 급여 및 해당 직무에 대한 총 급여를 표시하고 각 열에 적절한 머리글을 지정하기 위한 행렬 query를 작성합니다.

```
SELECT job_id "Job",
       SUM(DECODE(department_id, 20, salary)) "Dept 20",
       SUM(DECODE(department_id, 50, salary)) "Dept 50",
       SUM(DECODE(department_id, 80, salary)) "Dept 80",
       SUM(DECODE(department_id, 90, salary)) "Dept 90",
       SUM(salary) "Total"
  FROM employees
 GROUP BY job_id;
```

## 단원 6의 연습

이 연습은 SQL:1999 호환 조인을 사용하여 두 개 이상의 테이블에서 데이터를 추출하는 과정을 실습할 수 있도록 구성되었습니다.

## 연습 6-1: 조인을 사용하여 여러 테이블의 데이터 표시

- 1) HR 부서를 위해 모든 부서의 주소를 생성하는 query를 작성합니다. LOCATIONS 및 COUNTRIES 테이블을 사용합니다. 출력에 번지, 동/리, 구/군, 시/도 및 국가를 표시합니다. NATURAL JOIN을 사용하여 결과를 생성합니다.

	LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
2	1500	2011 Interiors Blvd	South San Francisco	California	United States of America
3	1700	2004 Charade Rd	Seattle	Washington	United States of America
4	1800	460 Bloor St. W.	Toronto	Ontario	Canada
5	2500	Magdalene Centre, The Oxford Science Park	Oxford	Oxford	United Kingdom

- 2) HR 부서에서 해당 부서가 있는 사원에 대해서만 보고서를 요구합니다. 이러한 사원의 성, 부서 ID 및 부서 이름을 표시하는 query를 작성합니다.

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping

...

18	Higgins	110	Accounting
19	Gietz	110	Accounting

- 3) HR 부서에서 Toronto에 근무하는 사원에 대한 보고서를 요구합니다. Toronto에서 근무하는 모든 사원의 성, 직무, 부서 ID 및 부서 이름을 표시합니다.

	LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	Hartstein	MK_MAN	20	Marketing
2	Fay	MK_REP	20	Marketing

- 4) 사원의 성 및 사원 번호를 해당 관리자의 성 및 관리자 번호와 함께 표시하는 보고서를 작성합니다. 열 레이블을 각각 Employee, Emp#, Manager, Mgr#으로 지정합니다. SQL 문을 lab\_06\_04.sql로 저장합니다. query를 실행합니다.

	Employee	EMP#	Manager	Mgr#
1	Hunold	103	De Haan	102
2	Fay	202	Hartstein	201
3	Gietz	206	Higgins	205
4	Lorentz	107	Hunold	103
5	Ernst	104	Hunold	103

...

18	Taylor	176	Zlotkey	149
19	Abel	174	Zlotkey	149

## 연습 6-1: 조인을 사용하여 여러 테이블의 데이터 표시 (계속)

- 5) King을 비롯하여 해당 관리자가 지정되지 않은 모든 사원을 표시하도록 lab\_06\_04.sql을 수정합니다. 사원 번호순으로 결과를 정렬합니다. SQL 문을 lab\_06\_05.sql로 저장합니다. lab\_06\_05.sql의 query를 실행합니다.

Employee	EMP#	Manager	Mgr#
1 King	100 (null)	(null)	
2 Kochhar	101 King	100	
3 De Haan	102 King	100	
4 Hunold	103 De Haan	102	
5 Ernst	104 Hunold	103	

...

19 Higgins	205 Kochhar	101
20 Gietz	206 Higgins	205

- 6) HR 부서를 위해 사원의 성과 부서 ID 및 주어진 사원과 동일한 부서에 근무하는 모든 사원을 표시하는 보고서를 작성합니다. 각 열에 적절한 레이블을 지정합니다. 이 스크립트를 lab\_06\_06.sql이라는 파일에 저장합니다.

DEPARTMENT	EMPLOYEE	COLLEAGUE
1	20 Fay	Hartstein
2	20 Hartstein	Fay
3	50 Davies	Matos
4	50 Davies	Mourgos
5	50 Davies	Rajs

...

41	110 Gietz	Higgins
42	110 Higgins	Gietz

- 7) HR 부서에서 직책 등급 및 급여에 대한 보고서를 요구합니다. JOB\_GRADES 테이블에 익숙해지도록 먼저 JOB\_GRADES 테이블의 구조를 표시합니다. 그런 다음 모든 사원의 이름, 직무, 부서 이름, 급여 및 등급을 표시하는 query를 작성합니다.

DESC JOB_GRADES		
Name	Null	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

3 rows selected

## 연습 6-1: 조인을 사용하여 여러 테이블의 데이터 표시 (계속)

	LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRADE_LEVEL
1	King	AD_PRES	Executive	24000	E
2	Kochhar	AD_VP	Executive	17000	E
3	De Haan	AD_VP	Executive	17000	E
4	Hartstein	MK_MAN	Marketing	13000	D
5	Higgins	AC_MGR	Accounting	12000	D

...

18	Matos	ST_CLERK	Shipping	2600	A
19	Vargas	ST_CLERK	Shipping	2500	A

다른 작업을 수행하려면 다음 연습을 완료합니다.

- 8) HR 부서에서 Davies 이후에 채용된 모든 사원의 이름을 파악하려고 합니다.

사원 Davies 이후로 채용된 모든 사원의 이름과 채용 날짜를 표시하기 위한 query를 작성합니다.

	LAST_NAME	HIRE_DATE
1	Fay	17-AUG-97
2	Lorentz	07-FEB-99
3	Mourgos	16-NOV-99
4	Matos	15-MAR-98
5	Vargas	09-JUL-98
6	Zlotkey	29-JAN-00
7	Taylor	24-MAR-98
8	Grant	24-MAY-99

- 9) HR 부서에서 관리자보다 먼저 채용된 모든 사원의 이름과 채용 날짜 및 해당 리자의 이름과 채용 날짜를 찾으려고 합니다. 이 스크립트를 lab\_06\_09.sql이라는 파일에 저장합니다.

	LAST_NAME	HIRE_DATE	LAST_NAME_1	HIRE_DATE_1
1	Whalen	17-SEP-87	Kochhar	21-SEP-89
2	Hunold	03-JAN-90	De Haan	13-JAN-93
3	Vargas	09-JUL-98	Mourgos	16-NOV-99
4	Matos	15-MAR-98	Mourgos	16-NOV-99
5	Davies	29-JAN-97	Mourgos	16-NOV-99
6	Rajs	17-OCT-95	Mourgos	16-NOV-99
7	Grant	24-MAY-99	Zlotkey	29-JAN-00
8	Taylor	24-MAR-98	Zlotkey	29-JAN-00
9	Abel	11-MAY-96	Zlotkey	29-JAN-00

## 연습 6-1 해답: 조인을 사용하여 여러 테이블의 데이터 표시

- 1) HR 부서를 위해 모든 부서의 주소를 생성하는 query를 작성합니다. LOCATIONS 및 COUNTRIES 테이블을 사용합니다. 출력에 번지, 동/리, 구/군, 시/도 및 국가를 표시합니다. NATURAL JOIN을 사용하여 결과를 생성합니다.

```
SELECT location_id, street_address, city, state_province,
       country_name
  FROM   locations
 NATURAL JOIN countries;
```

- 2) HR 부서에서 모든 사원에 대한 보고서를 요구합니다. 모든 사원의 성, 부서 ID 및 부서 이름을 표시하는 query를 작성합니다.

```
SELECT last_name, department_id, department_name
  FROM   employees
  JOIN   departments
  USING (department_id);
```

- 3) HR 부서에서 Toronto에 근무하는 사원에 대한 보고서를 요구합니다. Toronto에서 근무하는 모든 사원의 성, 직무, 부서 ID 및 부서 이름을 표시합니다.

```
SELECT e.last_name, e.job_id, e.department_id,
       d.department_name
  FROM   employees e JOIN departments d
  ON     (e.department_id = d.department_id)
  JOIN   locations l
  ON     (d.location_id = l.location_id)
 WHERE  LOWER(l.city) = 'toronto';
```

- 4) 사원의 성 및 사원 번호를 해당 관리자의 성 및 관리자 번호와 함께 표시하는 보고서를 작성합니다. 열 레이블을 각각 Employee, Emp#, Manager, Mgr#으로 지정합니다. SQL 문을 lab\_06\_04.sql로 저장합니다. query를 실행합니다.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id "Mgr#"
  FROM   employees w join employees m
  ON     (w.manager_id = m.employee_id);
```

- 5) King을 비롯하여 해당 관리자가 지정되지 않은 모든 사원을 표시하도록 lab\_06\_04.sql을 수정합니다. 사원 번호순으로 결과를 정렬합니다. SQL 문을 lab\_06\_05.sql로 저장합니다. lab\_06\_05.sql의 query를 실행합니다.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id "Mgr#"
  FROM   employees w
 LEFT   OUTER JOIN employees m
  ON     (w.manager_id = m.employee_id)
 ORDER BY 2;
```

## 연습 6-1 해답: 조인을 사용하여 여러 테이블의 데이터 표시 (계속)

- 6) HR 부서를 위해 사원의 성과 부서 ID 및 주어진 사원과 동일한 부서에 근무하는 모든 사원을 표시하는 보고서를 작성합니다. 각 열에 적절한 레이블을 지정합니다. 이 스크립트를 lab\_06\_06.sql이라는 파일에 저장합니다. query를 실행합니다.

```
SELECT e.department_id department, e.last_name employee,
       c.last_name colleague
  FROM employees e JOIN employees c
    ON (e.department_id = c.department_id)
 WHERE e.employee_id <> c.employee_id
ORDER BY e.department_id, e.last_name, c.last_name;
```

- 7) HR 부서에서 직책 등급 및 급여에 대한 보고서를 요구합니다. JOB\_GRADES 테이블에 익숙해지도록 먼저 JOB\_GRADES 테이블의 구조를 표시합니다. 그런 다음 모든 사원의 이름, 직무, 부서 이름, 급여 및 등급을 표시하는 query를 작성합니다.

```
DESC JOB_GRADES

SELECT e.last_name, e.job_id, d.department_name,
       e.salary, j.grade_level
  FROM employees e JOIN departments d
    ON (e.department_id = d.department_id)
   JOIN job_grades j
    ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

다른 작업을 수행하려면 다음 연습을 완료합니다.

- 8) HR 부서에서 Davies 이후에 채용된 모든 사원의 이름을 파악하려고 합니다. 사원 Davies 이후로 채용된 모든 사원의 이름과 채용 날짜를 표시하기 위한 query를 작성합니다.

```
SELECT e.last_name, e.hire_date
  FROM employees e JOIN employees davies
    ON (davies.last_name = 'Davies')
 WHERE davies.hire_date < e.hire_date;
```

- 9) HR 부서에서 관리자보다 먼저 채용된 모든 사원의 이름과 채용 날짜 및 해당 리자의 이름과 채용 날짜를 찾으려고 합니다. 이 스크립트를 lab\_06\_09.sql이라는 파일에 저장합니다.

```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
  FROM employees w JOIN employees m
    ON (w.manager_id = m.employee_id)
 WHERE w.hire_date < m.hire_date;
```

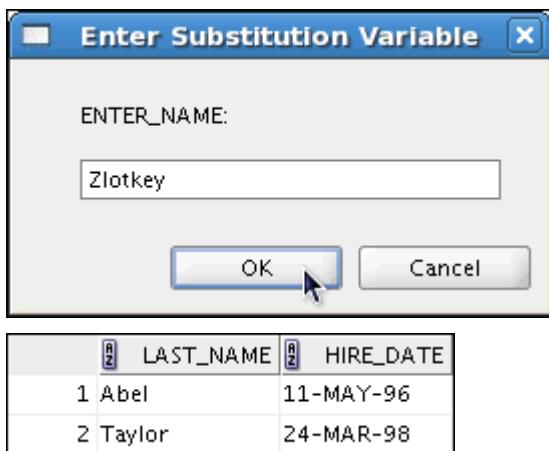
## 단원 7의 연습

이 연습에서는 중첩된 SELECT 문을 사용하여 복합 query를 작성합니다.

연습 문제의 경우 inner query를 먼저 작성할 수 있습니다. outer query를 코딩하기 전에 inner query를 실행하여 예상되는 데이터가 생성되는지 확인하십시오.

## 연습 7-1: Subquery를 사용하여 Query 해결

- 1) HR 부서에서 유저에게 사원의 성을 입력하라는 프롬프트를 표시하는 query를 요구합니다. 그런 다음 이 query는 유저가 입력한 이름의 사원과 동일한 부서에서 근무하는 사원의 성과 채용 날짜를 표시합니다(입력한 사원은 제외). 예를 들어, 유저가 Zlotkey를 입력하면 Zlotkey와 함께 근무하는 모든 사원을 찾습니다(Zlotkey는 제외).



- 2) 평균 급여 이상을 받는 모든 사원의 사원 번호, 성 및 급여를 표시하는 보고서를 작성합니다. 급여의 오름차순으로 결과를 정렬합니다.

	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000
2	149	Zlotkey	10500
3	174	Abel	11000
4	205	Higgins	12000
5	201	Hartstein	13000
6	102	De Haan	17000
7	101	Kochhar	17000
8	100	King	24000

- 3) 성에 문자 "u"가 포함된 사원과 같은 부서에 근무하는 모든 사원의 사원 번호와 성을 표시하는 query를 작성합니다. 작성한 SQL 문을 lab\_07\_03.sql로 저장합니다. query를 실행합니다.

	EMPLOYEE_ID	LAST_NAME
1	124	Mourgos
2	141	Rajs
3	142	Davies
4	143	Matos
5	144	Vargas
6	103	Hunold
7	104	Ernst
8	107	Lorentz

## 연습 7-1: Subquery 를 사용하여 Query 해결 (계속)

- 4) HR 부서에서 부서 위치 ID가 1700인 모든 사원의 성, 부서 ID 및 직무 ID를 표시하는 보고서를 요구합니다.

	LAST_NAME	DEPARTMENT_ID	JOB_ID
1	Whalen		10 AD_ASST
2	King		90 AD_PRES
3	Kochhar		90 AD_VP
4	De Haan		90 AD_VP
5	Higgins		110 AC_MGR
6	Gietz		110 AC_ACCOUNT

유저에게 번지를 입력하는 프롬프트를 표시하도록 query를 수정합니다.

입력한 내용을 lab\_07\_04.sql이라는 파일에 저장합니다.

- 5) HR을 위해 King에게 보고하는 모든 사원의 성과 급여를 표시하는 보고서를 작성합니다.

	LAST_NAME	SALARY
1	Hartstein	13000
2	Kochhar	17000
3	De Haan	17000
4	Mourgos	5800
5	Zlotkey	10500

- 6) HR을 위해 Executive 부서의 모든 사원에 대해 부서 ID, 성 및 직무 ID를 표시하는 보고서를 작성합니다.

	DEPARTMENT_ID	LAST_NAME	JOB_ID
1		King	AD_PRES
2		Kochhar	AD_VP
3		De Haan	AD_VP

- 7) 부서 60의 사원보다 급여가 많은 모든 사원 리스트를 표시하는 보고서를 작성합니다.

시간 여유가 있을 경우 다음 연습을 완료하십시오.

- 8) 평균보다 많은 급여를 받고 성에 "u"가 포함된 사원이 있는 부서에서 근무하는 모든 사원의 사원 번호, 성 및 급여를 표시하도록 lab\_07\_03.sql의 query를 수정합니다. lab\_07\_03.sql을 lab\_07\_08.sql로 다시 저장합니다.  
lab\_07\_08.sql의 명령문을 실행합니다.

	EMPLOYEE_ID	LAST_NAME	SALARY
1		Hunold	9000

## 연습 7-1 해답: Subquery를 사용하여 Query 해결

- 1) HR 부서에서 유저에게 사원의 성을 입력하라는 프롬프트를 표시하는 query를 요구합니다. 그런 다음 이 query는 유저가 입력한 이름의 사원과 동일한 부서에서 근무하는 사원의 성과 채용 날짜를 표시합니다(입력한 사원은 제외). 예를 들어, 유저가 Zlotkey를 입력하면 Zlotkey와 함께 근무하는 모든 사원을 찾습니다(Zlotkey는 제외).

```
UNDEFINE Enter_name

SELECT last_name, hire_date
FROM   employees
WHERE  department_id = (SELECT department_id
                        FROM   employees
                        WHERE  last_name = '&&Enter_name')
AND    last_name <> '&Enter_name';
```

- 2) 평균 급여 이상을 받는 모든 사원의 사원 번호, 성 및 급여를 표시하는 보고서를 작성합니다. 급여의 오름차순으로 결과를 정렬합니다.

```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  salary > (SELECT AVG(salary)
                  FROM   employees)
ORDER BY salary;
```

- 3) 성에 "u"가 포함된 사원과 같은 부서에 근무하는 모든 사원의 사원 번호와 성을 표시하는 query를 작성합니다. 작성한 SQL 문을 lab\_07\_03.sql로 저장합니다. query를 실행합니다.

```
SELECT employee_id, last_name
FROM   employees
WHERE  department_id IN (SELECT department_id
                          FROM   employees
                          WHERE  last_name like '%u%');
```

- 4) HR 부서에서 부서 위치 ID가 1700인 모든 사원의 성, 부서 ID 및 작업 ID를 표시하는 보고서를 요구합니다.

```
SELECT last_name, department_id, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                          FROM   departments
                          WHERE  location_id = 1700);
```

## 연습 7-1 해답: Subquery 를 사용하여 Query 해결 (계속)

유저에게 번지를 입력하는 프롬프트를 표시하도록 query를 수정합니다. 입력한 내용을 lab\_07\_04.sql이라는 파일에 저장합니다.

```
SELECT last_name, department_id, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                           FROM   departments
                           WHERE  location_id =
&Enter_location);
```

- 5) HR을 위해 King에게 보고하는 모든 사원의 성과 급여를 표시하는 보고서를 작성합니다.

```
SELECT last_name, salary
FROM   employees
WHERE  manager_id = (SELECT employee_id
                      FROM   employees
                      WHERE  last_name = 'King');
```

- 6) HR을 위해 Executive 부서의 모든 사원에 대해 부서 ID, 성 및 직무 ID를 표시하는 보고서를 작성합니다.

```
SELECT department_id, last_name, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                           FROM   departments
                           WHERE  department_name =
'Executive');
```

- 7) 부서 60의 사원보다 급여가 많은 모든 사원 리스트를 표시하는 보고서를 작성합니다.

```
SELECT last_name FROM employees
WHERE salary > ANY (SELECT salary
                     FROM employees
                     WHERE department_id=60);
```

시간 여유가 있을 경우 다음 연습을 완료하십시오.

- 8) 평균보다 많은 급여를 받고 성에 "u"가 포함된 사원이 있는 부서에서 근무하는 모든 사원의 사원 번호, 성 및 급여를 표시하도록 lab\_07\_03.sql의 query를 수정합니다. lab\_07\_03.sql을 lab\_07\_08.sql로 다시 저장합니다.  
lab\_07\_08.sql의 명령문을 실행합니다.

```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  department_id IN (SELECT department_id
                           FROM   employees
                           WHERE  last_name like '%u%')
AND    salary > (SELECT AVG(salary)
                  FROM   employees);
```

## 단원 8의 연습

이 연습에서는 집합 연산자를 사용하여 query를 작성합니다.

## 연습 8-1: 집합 연산자 사용

- 1) HR 부서에서 직무 ID ST\_CLERK을 포함하지 않는 부서에 대한 부서 ID 리스트를 요구합니다. 집합 연산자를 사용하여 이 보고서를 작성합니다.

	DEPARTMENT_ID
1	10
2	20
3	60
4	80
5	90
6	110
7	190

- 2) HR 부서에서 부서가 소재하지 않는 국가의 리스트를 요구합니다. 해당 국가의 국가 ID 및 이름을 표시합니다. 집합 연산자를 사용하여 이 보고서를 작성합니다.

	COUNTRY_ID	COUNTRY_NAME
1	DE	Germany

- 3) 부서 10, 50 및 20에 대한 직무 목록을 이 순서대로 생성합니다. 집합 연산자를 사용하여 직무 ID 및 부서 ID를 표시합니다.

	JOB_ID	DEPARTMENT_ID
1	AD_ASST	10
2	ST_MAN	50
3	ST_CLERK	50
4	MK_MAN	20
5	MK_REP	20

- 4) 현재 직책이 회사에 처음 입사할 때의 직책과 동일한 사원(즉, 직무가 변경된 적은 있지만 현재 원래 직무로 복귀한 사원)의 사원 ID와 직무 ID를 나열하는 보고서를 작성합니다.

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

- 5) HR 부서에서 다음과 같은 사양의 보고서를 요구합니다.

- 부서에 소속되었는지 여부에 관계없이 EMPLOYEES 테이블에 있는 모든 사원의 성 및 부서 ID
- 해당 부서에 근무 중인 사원이 있는지 여부에 관계없이 DEPARTMENTS 테이블에 있는 모든 부서의 부서 ID 및 부서 이름

## 연습 8-1: 집합 연산자 사용 (계속)

i]를 수행하는 복합 query를 작성합니다.

	LAST_NAME	DEPARTMENT_ID	TO_CHAR(NULL)
1	Abel		80 (null)
2	Davies		50 (null)
3	De Haan		90 (null)
4	Ernst		60 (null)
5	Fay		20 (null)
6	Gietz		110 (null)
7	Grant		(null) (null)
8	Hartstein		20 (null)
9	Higgins		110 (null)
10	Hunold		60 (null)
11	King		90 (null)
12	Kochhar		90 (null)
13	Lorentz		60 (null)
14	Matos		50 (null)
15	Mourgos		50 (null)
16	Rajs		50 (null)
17	Taylor		80 (null)
18	Vargas		50 (null)
19	Whalen		10 (null)
20	Zlotkey		80 (null)
21	(null)	10	Administration
22	(null)	20	Marketing
23	(null)	50	Shipping
24	(null)	60	IT
25	(null)	80	Sales
26	(null)	90	Executive
27	(null)	110	Accounting
28	(null)	190	Contracting

## 연습 8-1 해답: 집합 연산자 사용

- 1) HR 부서에서 직무 ID ST\_CLERK을 포함하지 않는 부서에 대한 부서 ID 리스트를 요구합니다. 집합 연산자를 사용하여 이 보고서를 작성합니다.

```
SELECT department_id
FROM departments
MINUS
SELECT department_id
FROM employees
WHERE job_id = 'ST_CLERK';
```

- 2) HR 부서에서 부서가 소재하지 않는 국가의 리스트를 요구합니다. 해당 국가의 국가 ID 및 이름을 표시합니다. 집합 연산자를 사용하여 이 보고서를 작성합니다.

```
SELECT country_id, country_name
FROM countries
MINUS
SELECT l.country_id, c.country_name
FROM locations l JOIN countries c
ON (l.country_id = c.country_id)
JOIN departments d
ON d.location_id=l.location_id;
```

- 3) 부서 10, 50 및 20에 대한 직무 목록을 이 순서대로 생성합니다. 집합 연산자를 사용하여 직무 ID 및 부서 ID를 표시합니다.

```
SELECT distinct job_id, department_id
FROM employees
WHERE department_id = 10
UNION ALL
SELECT DISTINCT job_id, department_id
FROM employees
WHERE department_id = 50
UNION ALL
SELECT DISTINCT job_id, department_id
FROM employees
WHERE department_id = 20
```

- 4) 현재 직책이 회사에 처음 입사할 때의 직책과 동일한 사원(즉, 직무가 변경된 적은 있지만 현재 원래 직무로 복귀한 사원)의 사원 ID와 직무 ID를 나열하는 보고서를 작성합니다.

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```

## 연습 8-1 해답: 집합 연산자 사용 (계속)

5) HR 부서에서 다음과 같은 사양의 보고서를 요구합니다.

- 부서에 소속되었는지 여부에 관계없이 EMPLOYEES 테이블에 있는 모든 사원의 성 및 부서 ID
- 해당 부서에 근무 중인 사원이 있는지 여부에 관계없이 DEPARTMENTS 테이블에 있는 모든 부서의 부서 ID 및 부서 이름

이를 수행하는 복합 query를 작성합니다.

```
SELECT last_name,department_id,TO_CHAR(null)
FROM   employees
UNION
SELECT TO_CHAR(null),department_id,department_name
FROM   departments;
```

## 단원 9의 연습

이 연습에서는 MY\_EMPLOYEE 테이블에 행을 추가하고, 테이블에서 데이터를 생성 및 삭제하고, 트랜잭션을 제어하는 과정을 다릅니다. 스크립트를 실행하여 MY\_EMPLOYEE 테이블을 생성합니다.

## 연습 9-1: DML 조작

HR 부서에서 사원 데이터를 삽입, 갱신 및 삭제하는 SQL 문을 작성해 달라고 합니다.  
HR 부서에 명령문을 제공하기에 앞서 프로토타입으로 MY\_EMPLOYEE 테이블을 사용합니다.

**참고:** 모든 DML 문의 경우에는 Run Script 아이콘 또는 [F5]를 사용하여 query를 실행합니다. 이렇게 하면 Script Output 탭 페이지에서 피드백 메시지를 볼 수 있습니다. SELECT 쿼리의 경우 계속해서 Execute Statement 아이콘을 사용하거나 [F9]를 누르면 Results 탭 페이지에서 형식이 지정된 출력을 볼 수 있습니다.

**MY\_EMPLOYEE 테이블에 데이터를 삽입합니다.**

- 1) lab\_09\_01.sql 스크립트의 명령문을 실행하여 이 연습에서 사용되는 MY\_EMPLOYEE 테이블을 생성합니다.
- 2) 열 이름을 식별하도록 MY\_EMPLOYEE 테이블의 구조를 기술합니다.

DESCRIBE my_employee		
Name	Null	Type
ID	NOT NULL	NUMBER(4)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
USERID		VARCHAR2(8)
SALARY		NUMBER(9, 2)
5 rows selected		

- 3) 다음 예제 데이터의 첫번째 데이터 행을 MY\_EMPLOYEE 테이블에 추가하는 INSERT 문을 작성합니다. INSERT 절에 열을 나열하지 마십시오. 아직 모든 행을 넣지 마십시오.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

- 4) 앞의 리스트에서 가져온 예제 데이터의 두번째 행으로 MY\_EMPLOYEE 테이블을 채웁니다. 이번에는 INSERT 절에 명시적으로 열을 나열합니다.

## 연습 9-1: 데이터 조작 (계속)

5) 테이블에 추가한 내용을 확인합니다.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dance	Betty	bdance	860

- 6) 나머지 행을 MY\_EMPLOYEE 테이블에 로드하는 INSERT 문을 재사용 가능한 동적 스크립트 파일에 작성합니다. 스크립트는 모든 열(ID, LAST\_NAME, FIRST\_NAME, USERID, SALARY)에 대해 프롬프트를 표시해야 합니다. 이 스크립트를 lab\_09\_06.sql 파일에 저장합니다.
- 7) 작성한 스크립트에서 INSERT 문을 실행하여 3단계에 나열된 예제 데이터의 다음 두 행으로 테이블을 채웁니다.
- 8) 테이블에 추가한 내용을 확인합니다.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dance	Betty	bdance	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750

9) 데이터 추가 내용이 영구적으로 적용되도록 합니다.

MY\_EMPLOYEE 테이블에서 데이터를 생성하고 삭제합니다.

- 10) 사원 3의 성을 Drexler로 변경합니다.
- 11) 급여가 \$900 미만인 모든 사원에 대해 급여를 \$1000로 변경합니다.
- 12) 테이블에 대한 변경 사항을 확인합니다.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
2	Dance	Betty	bdance	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000

13) MY\_EMPLOYEE 테이블에서 Betty Dance를 삭제합니다.

14) 테이블에 대한 변경 사항을 확인합니다.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
2	Drexler	Ben	bbiri	1100
3	Newman	Chad	cnewman	1000

15) 보류 중인 모든 변경 사항을 커밋합니다.

MY\_EMPLOYEE 테이블에 대한 데이터 트랜잭션을 제어합니다.

- 16) 6단계에서 작성한 스크립트의 명령문을 사용하여 3단계에 나열된 예제 데이터의 마지막 행으로 테이블을 채웁니다. 스크립트의 명령문을 실행합니다.

## 연습 9-1: 데이터 조작 (계속)

17) 테이블에 추가한 내용을 확인합니다.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
2	Drexler	Ben	bbiri	1100
3	Newman	Chad	cnewman	1000
4	Ropeburn	Audrey	aropebur	1550

18) 트랜잭션 처리의 중간 지점에 표시합니다.

19) 그런 다음 MY\_EMPLOYEE 테이블에서 모든 행을 query합니다.

20) 테이블이 비어 있는지 확인합니다.

21) 이전의 INSERT 작업을 삭제하지 않은 채로 가장 최근의 DELETE 작업을 삭제합니다.

22) 새 행이 여전히 원래 상태를 유지하는지 확인합니다.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
2	Drexler	Ben	bbiri	1100
3	Newman	Chad	cnewman	1000
4	Ropeburn	Audrey	aropebur	1550

23) 데이터 추가 내용이 영구적으로 적용되도록 합니다.

시간 여유가 있을 경우 다음 연습을 완료하십시오.

24) 이름의 첫번째 문자와 성의 앞부분 일곱 문자를 연결하여 USERID를 자동으로 생성하도록 lab\_09\_06.sql 스크립트를 수정합니다. 생성된 USERID는 소문자여야 합니다. 따라서 이 스크립트는 유저에게 USERID를 입력하도록 요구하지 않아야 합니다. lab\_09\_24.sql이라는 파일에 이 스크립트를 저장합니다.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

25) lab\_09\_24.sql 스크립트를 실행하여 다음 레코드를 삽입합니다.

26) 새 행에 올바른 USERID가 추가되었는지 확인합니다.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Anthony	Mark	manthony	1230

## 연습 9-1 해답: 데이터 조작

MY\_EMPLOYEE 테이블에 데이터를 삽입합니다.

- 1) lab\_09\_01.sql 스크립트의 명령문을 실행하여 이 연습에서 사용되는 MY\_EMPLOYEE 테이블을 생성합니다.
  - a) File 메뉴에서 Open을 선택합니다. Open 대화상자에서 /home/oracle/labs/sql1/labs 폴더를 찾은 다음 lab\_09\_01.sql을 두 번 누릅니다.
  - b) SQL Worksheet에서 명령문이 열리면 Run Script 아이콘을 눌러 스크립트를 실행합니다. Script Output 탭 페이지에 테이블 생성 성공 메시지가 나타납니다.
- 2) 열 이름을 식별하도록 MY\_EMPLOYEE 테이블의 구조를 기술합니다.

```
DESCRIBE my_employee
```

- 3) 다음 예제 데이터의 첫번째 데이터 행을 MY\_EMPLOYEE 테이블에 추가하는 INSERT 문을 작성합니다. INSERT 절에 열을 나열하지 마십시오.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

```
INSERT INTO my_employee
VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);
```

- 4) 앞의 리스트에서 가져온 예제 데이터의 두번째 행으로 MY\_EMPLOYEE 테이블을 채웁니다. 이번에는 INSERT 절에 명시적으로 열을 나열합니다.

```
INSERT INTO my_employee (id, last_name, first_name,
                        userid, salary)
VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

- 5) 테이블에 추가한 내용을 확인합니다.

```
SELECT *
FROM my_employee;
```

## 연습 9-1 해답: 데이터 조작 (계속)

- 6) 나머지 행을 MY\_EMPLOYEE 테이블에 로드하는 INSERT 문을 재사용 가능한 동적 스크립트 파일에 작성합니다. 스크립트는 모든 열(ID, LAST\_NAME, FIRST\_NAME, USERID, SALARY)에 대해 프롬프트를 표시해야 합니다.  
lab\_09\_06.sql이라는 파일에 이 스크립트를 저장합니다.

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
       '&p_userid', &p_salary);
```

- 7) 작성한 스크립트에서 INSERT 문을 실행하여 3단계에 나열된 예제 데이터의 다음 두 행으로 테이블을 채웁니다.

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
       '&p_userid', &p_salary);
```

- 8) 테이블에 추가한 내용을 확인합니다.

```
SELECT *
FROM my_employee;
```

- 9) 데이터 추가 내용이 영구적으로 적용되도록 합니다.

```
COMMIT;
```

**MY\_EMPLOYEE 테이블에서 데이터를 갱신하고 삭제합니다.**

- 10) 사원 3의 성을 Drexler로 변경합니다.

```
UPDATE my_employee
SET last_name = 'Drexler'
WHERE id = 3;
```

- 11) 급여가 \$900 미만인 모든 사원에 대해 급여를 \$1000로 변경합니다.

```
UPDATE my_employee
SET salary = 1000
WHERE salary < 900;
```

- 12) 테이블에 대한 변경 사항을 확인합니다.

```
SELECT *
FROM my_employee;
```

- 13) MY\_EMPLOYEE 테이블에서 Betty Dancs를 삭제합니다.

```
DELETE
FROM my_employee
WHERE last_name = 'Dancs';
```

- 14) 테이블에 대한 변경 사항을 확인합니다.

```
SELECT *
FROM my_employee;
```

## 연습 9-1 해답: 데이터 조작 (계속)

- 15) 보류 중인 모든 변경 사항을 커밋합니다.

```
COMMIT;
```

**MY\_EMPLOYEE**테이블에 대한 데이터 트랜잭션을 제어합니다.

- 16) 6단계에서 작성한 스크립트의 명령문을 사용하여 3단계에 나열된 예제 데이터의 마지막 행으로 테이블을 채웁니다. 스크립트의 명령문을 실행합니다.

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
'&p_userid', &p_salary);
```

- 17) 테이블에 추가한 내용을 확인합니다.

```
SELECT *
FROM my_employee;
```

- 18) 트랜잭션 처리의 중간 지점에 표시합니다.

```
SAVEPOINT step_17;
```

- 19) 그런 다음 **MY\_EMPLOYEE** 테이블에서 모든 행을 query합니다.

```
DELETE
FROM my_employee;
```

- 20) 테이블이 비어 있는지 확인합니다.

```
SELECT *
FROM my_employee;
```

- 21) 이전의 **INSERT** 작업을 삭제하지 않은 채로 가장 최근의 **DELETE** 작업을 삭제합니다.

```
ROLLBACK TO step_17;
```

- 22) 새 행이 여전히 원래 상태를 유지하는지 확인합니다.

```
SELECT *
FROM my_employee;
```

- 23) 데이터 추가 내용이 영구적으로 적용되도록 합니다.

```
COMMIT;
```

## 연습 9-1 해답: 데이터 조작 (계속)

시간 여유가 있을 경우 다음 연습을 완료합니다.

- 24) 이름의 첫번째 문자와 성의 앞부분 일곱 문자를 연결하여 USERID를 자동으로 생성하도록 lab\_09\_06.sql 스크립트를 수정합니다. 생성된 USERID는 소문자여야 합니다. 따라서 이 스크립트는 유저에게 USERID를 입력하도록 요구하지 않아야 합니다. lab\_09\_24.sql이라는 파일에 이 스크립트를 저장합니다.

```
SET ECHO OFF
SET VERIFY OFF
INSERT INTO my_employee
VALUES (&p_id, '&&p_last_name', '&&p_first_name',
        lower(substr('&p_first_name', 1, 1) ||
              substr('&p_last_name', 1, 7)), &p_salary);
SET VERIFY ON
SET ECHO ON
UNDEFINE p_first_name
UNDEFINE p_last_name
```

- 25) lab\_09\_24.sql 스크립트를 실행하여 다음 레코드를 삽입합니다.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

- 26) 새 행에 올바른 USERID가 추가되었는지 확인합니다.

```
SELECT *
FROM my_employee
WHERE ID='6';
```

## 단원 10의 연습

CREATE TABLE 문을 사용하여 새 테이블을 생성합니다. 데이터베이스에 새 테이블이 추가되었는지 확인합니다. 또한 테이블 상태를 READ ONLY로 설정한 다음 READ/WRITE로 복귀시키는 방법을 배웁니다.

**참고:** 모든 DDL 및 DML 문의 경우 SQL Developer에서 query를 실행하려면 Run Script 아이콘 또는 [F5]를 누릅니다. 이렇게 하면 Script Output 탭 페이지에서 피드백 메시지를 볼 수 있습니다. SELECT query의 경우 계속해서 Execute Statement 아이콘을 누르거나 [F9]를 누르면 Results 탭 페이지에서 형식이 지정된 출력을 볼 수 있습니다.

## 연습 10-1: DDL 문을 사용하여 테이블 생성 및 관리

- 1) 다음 테이블 instance 차트에 준하여 DEPT 테이블을 생성합니다.  
`lab_10_01.sql`이라는 스크립트에 명령문을 저장한 다음 해당 스크립트의 명령문을 실행하여 테이블을 생성합니다. 테이블이 생성되었는지 확인합니다.

열 이름	ID	NAME
키 유형	Primary Key	
Nulls/고유		
FK 테이블		
FK 열		
데이터 유형	NUMBER	VARCHAR2
길이	7	25

Name	Null	Type
ID	NOT NULL	NUMBER(7)
NAME		VARCHAR2(25)

- 2) DEPARTMENTS 테이블의 데이터로 DEPT 테이블을 채웁니다. 필요한 열만 포함시키십시오.
- 3) 다음 테이블 instance 차트에 준하여 EMP 테이블을 생성합니다.  
`lab_10_03.sql`이라는 스크립트에 명령문을 저장한 다음 해당 스크립트의 명령문을 실행하여 테이블을 생성합니다. 테이블이 생성되었는지 확인합니다.

열 이름	ID	LAST_NAME	FIRST_NAME	DEPT_ID
키 유형				
Nulls/고유				
FK 테이블				DEPT
FK 열				ID
데이터 유형	NUMBER	VARCHAR2	VARCHAR2	NUMBER
길이	7	25	25	7

Name	Null	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

## 연습 10-1: DDL 문을 사용하여 테이블 생성 및 관리 (계속)

- 4) EMPLOYEES 테이블의 구조에 준하여 EMPLOYEES2 테이블을 생성합니다.  
EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY 및 DEPARTMENT\_ID  
열만 포함시킵니다. 새 테이블의 열 이름을 각각 ID, FIRST\_NAME,  
LAST\_NAME, SALARY 및 DEPT\_ID로 지정합니다.
- 5) EMPLOYEES2 테이블 상태를 읽기 전용으로 변경합니다.
- 6) EMPLOYEES2 테이블에 다음 행을 삽입해 봅니다.

ID	FIRST_NAME	LAST_NAME	SALARY	DEPT_ID
34	Grant	Marcie	5678	10

다음 오류 메시지가 나타납니다.

```
Error starting at line 1 in command:
INSERT INTO employees2
VALUES (34, 'Grant','Marcie',5678,10)
Error at Command Line:1 Column:12
Error report:
SQL Error: ORA-12081: update operation not allowed on table "ORA1"."EMPLOYEES2"
12081. 00000 -  "update operation not allowed on table \"%s\".\"%s\""
*Cause: An attempt was made to update a read-only materialized view.
*Action: No action required. Only Oracle is allowed to update a
read-only materialized view.
```

- 7) EMPLOYEES2 테이블을 읽기/쓰기 상태로 복귀시킵니다. 이제 동일한 행을 다시  
삽입해 봅니다. 다음 메시지가 나타나야 합니다.

```
ALTER TABLE employees2 succeeded.
1 rows inserted
```

- 8) EMPLOYEES2 테이블을 삭제합니다.

## 연습 10-1 해답: DDL 문을 사용하여 테이블 생성 및 관리

- 1) 다음 테이블 instance 차트에 준하여 DEPT 테이블을 생성합니다.

lab\_10\_01.sql이라는 스크립트에 명령문을 저장한 다음 해당 스크립트의 명령문을 실행하여 테이블을 생성합니다. 테이블이 생성되었는지 확인합니다.

열 이름	ID	NAME
키 유형	Primary Key	
Nulls/고유		
FK 테이블		
FK 열		
데이터 유형	NUMBER	VARCHAR2
길이	7	25

```
CREATE TABLE dept
(id NUMBER(7) CONSTRAINT department_id_pk PRIMARY KEY,
name VARCHAR2(25));
```

테이블이 생성되었는지 확인하고 테이블 구조를 보려면 다음 명령을 실행합니다.

```
DESCRIBE dept
```

- 2) DEPARTMENTS 테이블의 데이터로 DEPT 테이블을 채웁니다. 필요한 열만 포함시키십시오.

```
INSERT INTO dept
SELECT department_id, department_name
FROM departments;
```

- 3) 다음 테이블 instance 차트에 준하여 EMP 테이블을 생성합니다.

lab\_10\_03.sql이라는 스크립트에 명령문을 저장한 다음 해당 스크립트의 명령문을 실행하여 테이블을 생성합니다. 테이블이 생성되었는지 확인합니다.

열 이름	ID	LAST_NAME	FIRST_NAME	DEPT_ID
키 유형				
Nulls/고유				
FK 테이블				DEPT
FK 열				ID
데이터 유형	NUMBER	VARCHAR2	VARCHAR2	NUMBER
길이	7	25	25	7

## 연습 10-1 해답: DDL 문을 사용하여 테이블 생성 및 관리 (계속)

```
CREATE TABLE emp
  (id          NUMBER(7),
   last_name   VARCHAR2(25),
   first_name  VARCHAR2(25),
   dept_id     NUMBER(7)
   CONSTRAINT emp_dept_id_FK REFERENCES dept (id)
  );
```

테이블이 생성되었는지 확인하고 테이블 구조를 보려면 다음 명령을 실행합니다.

```
DESCRIBE emp
```

- 4) EMPLOYEES 테이블의 구조에 준하여 EMPLOYEES2 테이블을 생성합니다. EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY 및 DEPARTMENT\_ID 열만 포함시킵니다. 새 테이블의 열 이름을 각각 ID, FIRST\_NAME, LAST\_NAME, SALARY 및 DEPT\_ID로 지정합니다.

```
CREATE TABLE employees2 AS
  SELECT employee_id id, first_name, last_name, salary,
         department_id dept_id
    FROM employees;
```

- 5) EMPLOYEES2 테이블 상태를 읽기 전용으로 변경합니다.

```
ALTER TABLE employees2 READ ONLY
```

- 6) EMPLOYEES2 테이블에 다음 행을 삽입해 봅니다.

ID	FIRST_NAME	LAST_NAME	SALARY	DEPT_ID
34	Grant	Marcie	5678	10

"Update operation not allowed on table"이라는 오류 메시지가 나타납니다. 테이블의 상태가 읽기 전용으로 지정되어 있으므로 행을 추가할 수 없습니다.

```
INSERT INTO employees2
VALUES (34, 'Grant', 'Marcie', 5678, 10)
```

- 7) EMPLOYEES2 테이블을 읽기/쓰기 상태로 복귀시킵니다. 이제 동일한 행을 다시 삽입해 봅니다.

테이블의 상태가 READ WRITE로 지정되어 있으므로 행을 삽입할 수 있습니다.

```
ALTER TABLE employees2 READ WRITE
```

```
INSERT INTO employees2
VALUES (34, 'Grant', 'Marcie', 5678, 10)
```

- 8) EMPLOYEES2 테이블을 삭제합니다.

**참고:** READ ONLY 모드의 테이블도 삭제할 수 있습니다. 이를 테스트하려면 테이블을 다시 READ ONLY 상태로 변경한 다음 DROP TABLE 명령을 실행합니다. EMPLOYEES2 테이블이 삭제됩니다.

```
DROP TABLE employees2;
```

## 단원 11의 연습

이 단원의 1부 연습에서는 뷰를 생성, 사용 및 제거하는 다양한 연습을 제공합니다.

이 단원의 1번부터 6번까지의 문제를 모두 푸십시오.

이 단원의 2부 연습에서는 다양한 예제를 통해 시퀀스, 인덱스 및 동의어를 생성하고 사용하는 과정을 실습합니다. 이 단원의 7번부터 10번까지의 문제를 모두 푸십시오.

## 연습 11-1: 스키마 객체 관리

### 1부

- 1) HR 부서의 임원이 EMPLOYEES 테이블의 일부 데이터를 숨길 것을 요구합니다. EMPLOYEES 테이블의 사원 번호, 사원 성 및 부서 번호를 기반으로 하는 EMPLOYEES\_VU라는 뷰를 생성합니다. 사원 이름의 머리글은 EMPLOYEE여야 합니다.
- 2) 뷰가 작동하는지 확인합니다. EMPLOYEES\_VU 뷰의 내용을 표시합니다.

	EMPLOYEE_ID	EMPLOYEE	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
4	205	Higgins	110
5	206	Gietz	110

...

19	205 Higgins	110
20	206 Gietz	110

- 3) EMPLOYEES\_VU 뷰를 사용하여 HR 부서를 위해 모든 사원 이름과 부서 번호를 표시하는 query를 작성합니다.

	EMPLOYEE	DEPARTMENT_ID
1	King	90
2	Kochhar	90
3	De Haan	90
4	Hunold	60
5	Ernst	60

...

19	Higgins	110
20	Gietz	110

- 4) 부서 50의 사원 데이터에 액세스하려고 합니다. 부서 50의 모든 사원에 대해 사원 번호, 사원 성 및 부서 번호를 포함하는 DEPT50이라는 뷰를 생성합니다. 뷰 열의 레이블을 EMPNO, EMPLOYEE 및 DEPTNO로 지정하라는 요청을 받았습니다. 보안상의 이유로 사원이 뷰를 통해 다른 부서에 재할당되는 것을 허용하지 않습니다.

## 연습 11-1: 스키마 객체 관리 (계속)

- 5) DEPT50 뷰의 구조 및 내용을 표시합니다.

DESCRIBE dept50		
Name	Null	Type
EMPNO	NOT NULL	NUMBER(6)
EMPLOYEE	NOT NULL	VARCHAR2(25)
DEPTNO		NUMBER(4)

EMPNO	EMPLOYEE	DEPTNO
124	Mourgos	50
141	Rajs	50
142	Davies	50
143	Matos	50
144	Vargas	50

- 6) 뷰를 테스트합니다. Matos를 부서 80에 재할당해 보십시오.

### 2부

- 7) DEPT 테이블의 PRIMARY KEY 열에 사용할 수 있는 시퀀스가 필요합니다. 시퀀스는 200부터 시작하고 최대값은 1,000이어야 합니다. 시퀀스 충분 값을 10으로 설정하고 시퀀스 이름을 DEPT\_ID\_SEQ로 지정합니다.
- 8) 시퀀스를 테스트하기 위해 DEPT 테이블의 두 행을 삽입하는 스크립트를 작성합니다. 스크립트 이름을 lab\_11\_08.sql로 지정합니다. ID 열에 대해 생성한 시퀀스를 사용해야 합니다. Education 및 Administration이라는 두 개의 부서를 추가합니다. 추가한 내용을 확인합니다. 스크립트의 명령을 실행합니다.
- 9) DEPT 테이블의 NAME 열에 비고유 인덱스를 생성합니다.
- 10) EMPLOYEES 테이블의 동의어를 생성하고 이름을 EMP로 지정합니다.

## 연습 11-1 해답: 다른 스키마 객체 생성

### 1부

- 1) HR 부서의 임원이 EMPLOYEES 테이블의 일부 데이터를 숨길 것을 요구합니다. EMPLOYEES 테이블의 사원 번호, 사원 성 및 부서 번호를 기반으로 하는 EMPLOYEES\_VU라는 뷰를 생성합니다. 사원 이름의 머리글은 EMPLOYEE여야 합니다.

```
CREATE OR REPLACE VIEW employees_vu AS
    SELECT employee_id, last_name employee, department_id
    FROM employees;
```

- 2) 뷰가 작동하는지 확인합니다. EMPLOYEES\_VU 뷰의 내용을 표시합니다.

```
SELECT *
FROM employees_vu;
```

- 3) EMPLOYEES\_VU 뷰를 사용하여 HR 부서를 위해 모든 사원 이름과 부서 번호를 표시하는 query를 작성합니다.

```
SELECT employee, department_id
FROM employees_vu;
```

- 4) 부서 50의 사원 데이터에 액세스하려고 합니다. 부서 50의 모든 사원에 대해 사원 번호, 사원 성 및 부서 번호를 포함하는 DEPT50이라는 뷰를 생성합니다. 뷰의 레이블을 각각 EMPNO, EMPLOYEE 및 DEPTNO로 지정하라는 요청을 받았습니다. 보안상의 이유로 사원이 뷰를 통해 다른 부서에 재할당되는 것을 허용하지 않습니다.

```
CREATE VIEW dept50 AS
    SELECT employee_id empno, last_name employee,
           department_id deptno
    FROM employees
    WHERE department_id = 50
    WITH CHECK OPTION CONSTRAINT emp_dept_50;
```

- 5) DEPT50 뷰의 구조 및 내용을 표시합니다.

```
DESCRIBE dept50
SELECT *
FROM dept50;
```

- 6) 뷰를 테스트합니다. Matos를 부서 80에 재할당해 보십시오.

```
UPDATE dept50
SET deptno = 80
WHERE employee = 'Matos';
```

DEPT50 뷰가 WITH CHECK OPTION 제약 조건을 사용하여 생성되었으므로 오류가 발생합니다. 이 제약 조건은 뷰의 DEPTNO 열이 변경되지 않도록 합니다.

## 연습 11-1 해답: 다른 스키마 객체 생성 (계속)

### 2부

- 7) DEPT 테이블의 primary key 열로 사용할 수 있는 시퀀스가 필요합니다. 시퀀스는 200부터 시작하고 최대값은 1,000이어야 합니다. 시퀀스 증분 값을 10으로 설정하고 시퀀스 이름을 DEPT\_ID\_SEQ로 지정합니다.

```
CREATE SEQUENCE dept_id_seq
    START WITH 200
    INCREMENT BY 10
    MAXVALUE 1000;
```

- 8) 시퀀스를 테스트하기 위해 DEPT 테이블의 두 행을 삽입하는 스크립트를 작성합니다. 스크립트 이름을 lab\_11\_08.sql로 지정합니다. ID 열에 대해 생성한 시퀀스를 사용해야 합니다. Education 및 Administration이라는 두 개의 부서를 추가합니다. 추가한 내용을 확인합니다. 스크립트의 명령을 실행합니다.

```
INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Education');

INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Administration');
```

- 9) DEPT 테이블의 NAME 열에 비고유 인덱스를 생성합니다.

```
CREATE INDEX dept_name_idx ON dept (name);
```

- 10) EMPLOYEES 테이블의 동의어를 생성하고 이름을 EMP로 지정합니다.

```
CREATE SYNONYM emp FOR EMPLOYEES;
```

## **부록 F 해답**

이 연습은 Oracle 조인 구문을 사용하여 두 개 이상의 테이블에서 데이터를 추출하는 과정을 실습할 수 있도록 구성되었습니다.

## 부록 F-1: Oracle 조인 구문

- 1) HR 부서를 위해 모든 부서의 주소를 생성하는 query를 작성합니다. LOCATIONS 및 COUNTRIES 테이블을 사용합니다. 출력에 번지, 동/리, 구/군, 시/도 및 국가를 표시합니다. query를 실행합니다.

LOCATION_ID	STREET_ADDRESS		CITY	STATE_PROVINCE	COUNTRY_NAME
1	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
2	1500	2011 Interiors Blvd	South San Francisco	California	United States of America
3	1700	2004 Charade Rd	Seattle	Washington	United States of America
4	1800	460 Bloor St. W.	Toronto	Ontario	Canada
5	2500	Magdalen Centre, The Oxford Science Park	Oxford		United Kingdom

- 2) HR 부서에서 모든 사원에 대한 보고서를 요구합니다. 모든 사원의 성, 부서 ID 및 부서 이름을 표시하는 query를 작성합니다. query를 실행합니다.

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Davies	50	Shipping
Vargas	50	Shipping

...

18 Higgins	110	Accounting
19 Gietz	110	Accounting

- 3) HR 부서에서 Toronto에 근무하는 사원에 대한 보고서를 요구합니다. Toronto에서 근무하는 모든 사원의 성, 직무, 부서 ID 및 부서 이름을 표시합니다.

LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
Hartstein	MK_MAN	20	Marketing
Fay	MK_REP	20	Marketing

- 4) 사원의 성 및 사원 번호를 해당 관리자의 성 및 관리자 번호와 함께 표시하는 보고서를 작성합니다. 열 레이블을 각각 Employee, Emp#, Manager, Mgr#으로 지정합니다. SQL 문을 lab\_f\_04.sql로 저장합니다.

Employee	EMP#	Manager	Mgr#
1 Hunold	103	De Haan	102
2 Fay	202	Hartstein	201
3 Gietz	206	Higgins	205
4 Lorentz	107	Hunold	103
5 Ernst	104	Hunold	103

...

18 Taylor	176	Zlotkey	149
19 Abel	174	Zlotkey	149

## 부록 F-1: Oracle 조인 구문 (계속)

- 5) King을 비롯하여 해당 관리자가 지정되지 않은 모든 사원을 표시하도록 lab\_f\_04.sql을 수정합니다. 사원 번호순으로 결과를 정렬합니다. SQL 문을 lab\_f\_05.sql로 저장합니다. lab\_f\_05.sql의 query를 실행합니다.

Employee	EMP#	Manager	Mgr#
1 Hunold	103 De Haan	102	
2 Fay	202 Hartstein	201	
3 Gietz	206 Higgins	205	
4 Lorentz	107 Hunold	103	
5 Ernst	104 Hunold	103	

...

19 Abel	174 Zlotkey	149
20 King	100 (null)	(null)

- 6) HR 부서를 위해 사원의 성과 부서 ID 및 해당 사원과 동일한 부서에 근무하는 든 사원을 표시하는 보고서를 작성합니다. 각 열에 적절한 레이블을 지정합니다. 이 크립트를 lab\_f\_06.sql이라는 파일에 저장합니다.

DEPARTMENT	EMPLOYEE	COLLEAGUE
1	20 Fay	Hartstein
2	20 Hartstein	Fay
3	50 Davies	Matos
4	50 Davies	Mourgos
5	50 Davies	Rajs

...

39	90 Kochhar	De Haan
40	90 Kochhar	King
41	110 Gietz	Higgins
42	110 Higgins	Gietz

- 7) HR 부서에서 직책 등급 및 급여에 대한 보고서를 요구합니다. JOB\_GRADES 테이블에 익숙해지도록 먼저 JOB\_GRADES 테이블의 구조를 표시합니다. 그런 다음 모든 사원의 이름, 직무, 부서 이름, 급여 및 등급을 표시하는 query를 작성합니다.

Name	Null	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

## 부록 F-1: Oracle 조인 구문 (계속)

LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRADE_LEVEL
1 King	AD_PRES	Executive	24000 E	
2 De Haan	AD_VP	Executive	17000 E	
3 Kochhar	AD_VP	Executive	17000 E	
4 Hartstein	MK_MAN	Marketing	13000 D	
5 Higgins	AC_MGR	Accounting	12000 D	
...				
18 Matos	ST_CLERK	Shipping	2600 A	
19 Vargas	ST_CLERK	Shipping	2500 A	

다른 작업을 수행하려면 다음 연습을 완료합니다.

- 8) HR 부서에서 Davies 이후에 채용된 모든 사원의 이름을 파악하려고 합니다.  
사원 Davies 이후로 채용된 모든 사원의 이름과 채용 날짜를 표시하기 위한 query를 작성합니다.

LAST_NAME	HIRE_DATE
1 Lorentz	07-FEB-99
2 Mourgos	16-NOV-99
3 Matos	15-MAR-98
4 Vargas	09-JUL-98
5 Zlotkey	29-JAN-00
6 Taylor	24-MAR-98
7 Grant	24-MAY-99
8 Fay	17-AUG-97

- 9) HR 부서에서 관리자보다 먼저 채용된 모든 사원의 이름과 채용 날짜 및 해당 리자의 이름과 채용 날짜를 찾으려고 합니다. 이 스크립트를 lab\_f\_09.sql이라는 파일에 저장합니다.

LAST_NAME	HIRE_DATE	LAST_NAME_1	HIRE_DATE_1
1 Whalen	17-SEP-87	Kochhar	21-SEP-89
2 Hunold	03-JAN-90	De Haan	13-JAN-93
3 Vargas	09-JUL-98	Mourgos	16-NOV-99
4 Matos	15-MAR-98	Mourgos	16-NOV-99
5 Davies	29-JAN-97	Mourgos	16-NOV-99
6 Rajs	17-OCT-95	Mourgos	16-NOV-99
7 Grant	24-MAY-99	Zlotkey	29-JAN-00
8 Taylor	24-MAR-98	Zlotkey	29-JAN-00
9 Abel	11-MAY-96	Zlotkey	29-JAN-00

## 연습 F-1 해답: Oracle 조인 구문

- 1) HR 부서를 위해 모든 부서의 주소를 생성하는 query를 작성합니다. LOCATIONS 및 COUNTRIES 테이블을 사용합니다. 출력에 번지, 동/리, 구/군, 시/도 및 국가를 표시합니다. query를 실행합니다.

```
SELECT location_id, street_address, city, state_province,
       country_name
  FROM   locations, countries
 WHERE  locations.country_id = countries.country_id;
```

- 2) HR 부서에서 모든 사원에 대한 보고서를 요구합니다. 모든 사원의 성, 부서 ID 및 부서 이름을 표시하는 query를 작성합니다. query를 실행합니다.

```
SELECT e.last_name, e.department_id, d.department_name
  FROM   employees e, departments d
 WHERE  e.department_id = d.department_id;
```

- 3) HR 부서에서 Toronto에 근무하는 사원에 대한 보고서를 요구합니다. Toronto에서 근무하는 모든 사원의 성, 직무, 부서 ID 및 부서 이름을 표시합니다.

```
SELECT e.last_name, e.job_id, e.department_id,
       d.department_name
  FROM   employees e, departments d , locations l
 WHERE  e.department_id = d.department_id
 AND    d.location_id = l.location_id
 AND    LOWER(l.city) = 'toronto';
```

- 4) 사원의 성 및 사원 번호를 해당 관리자의 성 및 관리자 번호와 함께 표시하는 보고서를 작성합니다. 열 레이블을 각각 Employee, Emp#, Manager, Mgr#으로 지정합니다. SQL 문을 lab\_f\_04.sql로 저장합니다.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id "Mgr#"
  FROM   employees w, employees m
 WHERE  w.manager_id = m.employee_id;
```

- 5) King을 비롯하여 해당 관리자가 지정되지 않은 모든 사원을 표시하도록 lab\_f\_04.sql을 수정합니다. 사원 번호순으로 결과를 정렬합니다. SQL 문을 lab\_f\_05.sql로 저장합니다. lab\_f\_05.sql의 query를 실행합니다.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id "Mgr#"
  FROM   employees w, employees m
 WHERE  w.manager_id = m.employee_id (+);
```

## 연습 F-1 해답: Oracle 조인 구문 (계속)

- 6) HR 부서를 위해 사원의 성과 부서 ID 및 주어진 사원과 동일한 부서에 근무하는 든 사원을 표시하는 보고서를 작성합니다. 각 열에 적절한 레이블을 지정합니다. 이 스크립트를 lab\_f\_06.sql이라는 파일에 저장합니다.

```
SELECT e1.department_id department, e1.last_name employee,
       e2.last_name colleague
  FROM employees e1, employees e2
 WHERE e1.department_id = e2.department_id
   AND e1.employee_id <> e2.employee_id
 ORDER BY e1.department_id, e1.last_name, e2.last_name;
```

- 7) HR 부서에서 직책 등급 및 급여에 대한 보고서를 요구합니다. JOB\_GRADES 테이블에 익숙해지도록 먼저 JOB\_GRADES 테이블의 구조를 표시합니다. 그런 다음 모든 사원의 이름, 직무, 부서 이름, 급여 및 등급을 표시하는 query를 작성합니다.

```
DESC JOB_GRADES

SELECT e.last_name, e.job_id, d.department_name,
       e.salary, j.grade_level
  FROM employees e, departments d, job_grades j
 WHERE e.department_id = d.department_id
   AND e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

다른 작업을 수행하려면 다음 연습을 완료합니다.

- 8) HR 부서에서 Davies 이후에 채용된 모든 사원의 이름을 파악하려고 합니다. 사원 Davies 이후로 채용된 모든 사원의 이름과 채용 날짜를 표시하기 위한 query를 작성합니다.

```
SELECT e.last_name, e.hire_date
  FROM employees e , employees davies
 WHERE davies.last_name = 'Davies'
   AND davies.hire_date < e.hire_date;
```

- 9) HR 부서에서 관리자보다 먼저 채용된 모든 사원의 이름과 채용 날짜 및 해당 관리자의 이름과 채용 날짜를 찾으려고 합니다. 열 레이블을 각각 Employee, Emp Hired, Manager, Mgr Hired로 지정합니다. 이 스크립트를 lab\_f\_09.sql이라는 파일에 저장합니다.

```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
  FROM employees w , employees m
 WHERE w.manager_id = m.employee_id
   AND w.hire_date < m.hire_date;
```



# B 테이블 설명

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 스키마 설명

### 전반적인 설명

오라클 데이터베이스 샘플 스키마는 여러 다양한 제품에 대한 주문을 이행하기 위해 전세계적으로 운영되는 샘플 회사를 보여줍니다. 이 회사에는 다음과 같은 3개의 부서가 있습니다.

- **Human Resources:** 사원 및 회사 시설에 대한 정보를 추적합니다.
- **Order Entry:** 다양한 통로를 통해 제품 재고 및 판매량을 추적합니다.
- **Sales History:** 용이한 업무 결정을 위해 업무 통계를 추적합니다.

이러한 부서는 각각 스키마로 표현됩니다. 본 과정에서 수강생들은 이러한 모든 스키마의 객체에 액세스할 수 있습니다. 그러나 예제, 데모 및 연습에서는 HR (Human Resources) 스키마를 중점적으로 다룹니다.

예제 스키마를 생성하는 데 필요한 모든 스크립트는 \$ORACLE\_HOME/demo/schema/ 폴더에 있습니다.

### HR (Human Resources)

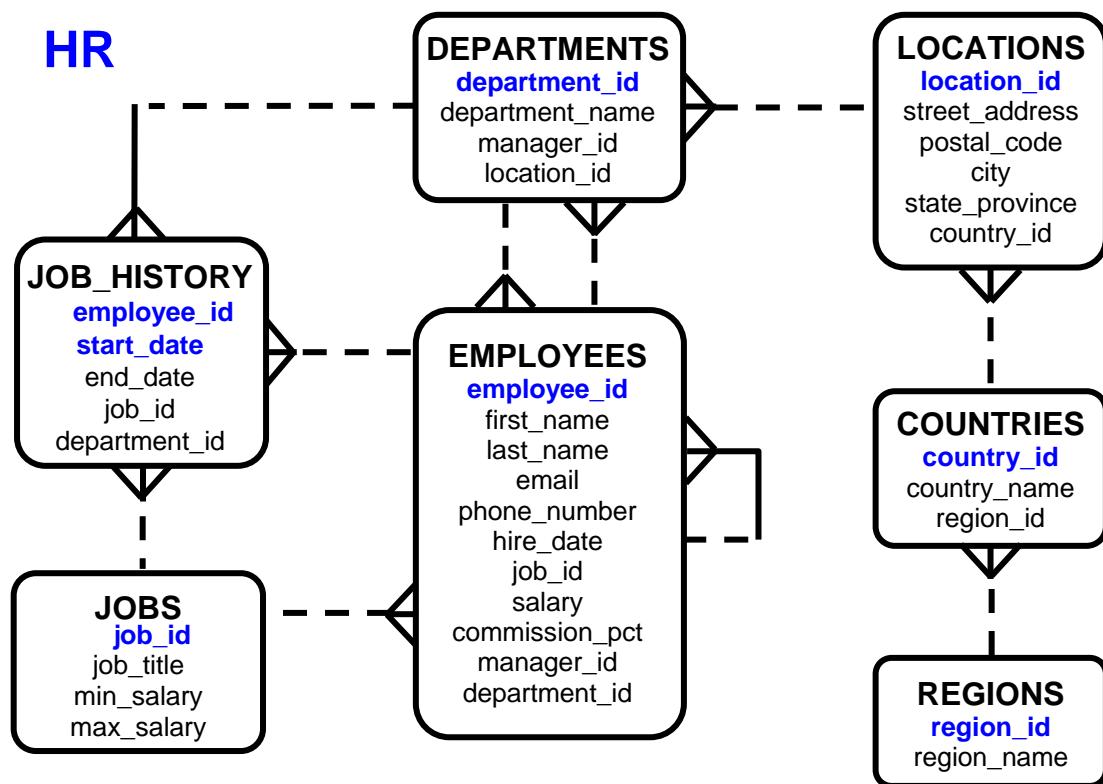
본 과정에서 사용되는 스키마입니다. 회사의 HR (Human Resource) 레코드에는 각 사원의 식별 번호, 전자 메일 주소, 업무 식별 코드, 급여 및 관리자가 기록되어 있습니다. 급여 외에 커미션을 받는 사원도 있습니다.

또한 조직 내에서 업무에 대한 정보를 추적합니다. 업무마다 해당 업무에 대한 식별 코드, 직위, 최소 및 최대 급여 범위가 있습니다. 일부 사원은 오랫동안 회사에 근무하며 사내에서 다양한 업무를 맡았습니다. 사원이 사임하면 사원이 근무한 기간, 직무 식별 번호 및 부서가 기록됩니다.

예제 회사는 여러 지역에 분포되어 있으므로 회사의 물류 창고 및 부서 위치를 추적합니다. 각 사원은 부서에 배정되고 각 부서는 고유한 부서 번호나 단축 이름으로 식별됩니다. 각 부서는 하나의 위치와 연관되고 각 위치는 번지, 우편 번호, 시, 도 및 국가 코드를 포함한 전체 주소를 가집니다.

회사는 부서 및 물류 창고가 위치한 장소에서 국가 이름, 통화 기호, 통화 이름을 비롯하여 해당 국가가 지리적으로 위치하고 있는 지역 등의 세부 사항을 기록합니다.

## HR 엔티티 관계 도표



## HR (Human Resources) 테이블 설명

DESCRIBE countries

Name	Null	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

SELECT \* FROM countries;

#	COUNTRY_ID	COUNTRY_NAME	REGION_ID
1	CA	Canada	2
2	DE	Germany	1
3	UK	United Kingdom	1
4	US	United States of America	2

## HR (Human Resources) 테이블 설명(계속)

DESCRIBE departments

Name	Null	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

SELECT \* FROM departments;

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10 Administration	200	1700
2	20 Marketing	201	1800
3	50 Shipping	124	1500
4	60 IT	103	1400
5	80 Sales	149	2500
6	90 Executive	100	1700
7	110 Accounting	205	1700
8	190 Contracting	(null)	1700

## HR (Human Resources) 테이블 설명(계속)

DESCRIBE employees

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

SELECT \* FROM employees;

#	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPART...
1	100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000	(null)	(null)	90
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000	(null)	100	90
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	(null)	100	90
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000	(null)	102	60
5	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000	(null)	103	60
6	107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	4200	(null)	103	60
7	124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	5800	(null)	100	50
8	141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	3500	(null)	124	50
9	142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	3100	(null)	124	50
10	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	2600	(null)	124	50
11	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	2500	(null)	124	50
12	149	Eleni	Zlotkey	EZLOTKEY	011.44.1344....	29-JAN-00	SA_MAN	10500	0.2	100	80
13	174	Ellen	Abel	EABEL	011.44.1644....	11-MAY-96	SA REP	11000	0.3	149	80
14	176	Jonathon	Taylor	JTAYLOR	011.44.1644....	24-MAR-98	SA REP	8600	0.2	149	80
15	178	Kimberely	Grant	KGRANT	011.44.1644....	24-MAY-99	SA REP	7000	0.15	149	(null)
16	200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400	(null)	101	10
17	201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK MAN	13000	(null)	100	20
18	202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK REP	6000	(null)	201	20
19	205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	12000	(null)	101	110
20	206	William	Gietz	WGIEWT	515.123.8181	07-JUN-94	AC_ACCOUNT	8300	(null)	205	110

...

## HR (Human Resources) 테이블 설명(계속)

DESCRIBE job\_history

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

SELECT \* FROM job\_history

#	EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
1	102	13-JAN-93	24-JUL-98	IT_PROG	60
2	101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
3	101	28-OCT-93	15-MAR-97	AC_MGR	110
4	201	17-FEB-96	19-DEC-99	MK_REP	20
5	114	24-MAR-98	31-DEC-99	ST_CLERK	50
6	122	01-JAN-99	31-DEC-99	ST_CLERK	50
7	200	17-SEP-87	17-JUN-93	AD_ASST	90
8	176	24-MAR-98	31-DEC-98	SA_REP	80
9	176	01-JAN-99	31-DEC-99	SA_MAN	80
10	200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

## HR (Human Resources) 테이블 설명(계속)

DESCRIBE jobs

Name	Null	Type
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

SELECT \* FROM jobs

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1 AD_PRES	President	20000	40000
2 AD_VP	Administration Vice President	15000	30000
3 AD_ASST	Administration Assistant	3000	6000
4 AC_MGR	Accounting Manager	8200	16000
5 AC_ACCOUNT	Public Accountant	4200	9000
6 SA_MAN	Sales Manager	10000	20000
7 SA_REP	Sales Representative	6000	12000
8 ST_MAN	Stock Manager	5500	8500
9 ST_CLERK	Stock Clerk	2000	5000
10 IT_PROG	Programmer	4000	10000
11 MK_MAN	Marketing Manager	9000	15000
12 MK_REP	Marketing Representative	4000	9000

## HR (Human Resources) 테이블 설명(계속)

DESCRIBE locations

Name	Null	Type
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

SELECT \* FROM locations

#	LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
1	1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
2	1500	2011 Interiors Blvd	99236	South San Francisco	California	US
3	1700	2004 Charade Rd	98199	Seattle	Washington	US
4	1800	460 Bloor St. W.	ON MSS 1X8	Toronto	Ontario	CA
5	2500	Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK

## HR (Human Resources) 테이블 설명(계속)

DESCRIBE regions

Name	Null	Type
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

SELECT \* FROM regions

	REGION_ID	REGION_NAME
1	1	Europe
2	2	Americas
3	3	Asia
4	4	Middle East and Africa



# **SQL Developer 사용**

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

# 목표

이 부록을 마치면 다음을 수행할 수 있습니다.

- Oracle SQL Developer의 주요 기능 숙지
- Oracle SQL Developer의 메뉴 항목 식별
- 데이터베이스 연결 생성
- 데이터베이스 객체 관리
- SQL Worksheet 사용
- SQL 스크립트 저장 및 실행
- 보고서 작성 및 저장

ORACLE®

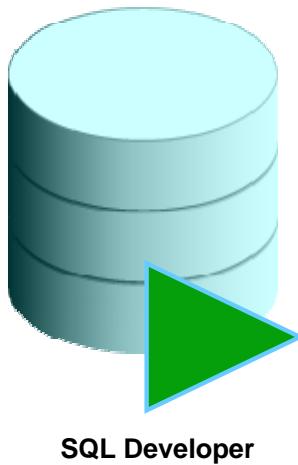
Copyright © 2009, Oracle. All rights reserved.

## 목표

이 부록에서는 SQL Developer라는 그래픽 도구에 대해 알아봅니다. 먼저, 데이터베이스 개발 작업에 SQL Developer를 사용하는 방법에 대해 알아보고, SQL Worksheet를 사용하여 SQL 문과 SQL 스크립트를 실행하는 방법을 학습합니다.

# Oracle SQL Developer란?

- **Oracle SQL Developer는 생산성을 향상하고 데이터베이스 개발 작업을 단순화하는 그래픽 도구입니다.**
- **표준 오라클 데이터베이스 인증을 사용하여 대상 오라클 데이터베이스 스키마에 연결할 수 있습니다.**



**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

## Oracle SQL Developer란?

Oracle SQL Developer는 생산성을 높이고 일상적인 데이터베이스 작업의 개발을 간소화하기 위해 무료로 제공되는 그래픽 도구입니다. 마우스를 몇 번 누르는 것만으로 간단하게 내장 프로시저를 생성 및 디버그하고, SQL 문을 테스트하고, 옵티마이저 계획을 볼 수 있습니다.

데이터베이스 개발을 위한 시각적 도구인 SQL Developer는 다음 작업을 단순화합니다.

- 데이터베이스 객체 탐색 및 관리
- SQL 문 및 스크립트 실행
- PL/SQL 문 편집 및 디버깅
- 보고서 작성

표준 오라클 데이터베이스 인증을 사용하여 대상 오라클 데이터베이스 스키마에 연결할 수 있습니다. 연결되면 데이터베이스의 객체에 대해 작업을 수행할 수 있습니다.

SQL Developer 1.2 버전은 유저가 단일 위치에서 third-party 데이터베이스의 데이터 및 데이터베이스 객체를 탐색하고 이러한 데이터베이스에서 오라클로 이전할 수 있도록 하는 Developer Migration Workbench와 긴밀하게 통합됩니다. 또한 MySQL, Microsoft SQL Server 및 Microsoft Access 등의 타사 데이터베이스를 선택하고 스키마에 연결하여 이러한 데이터베이스의 메타 데이터 및 데이터를 볼 수 있습니다.

뿐만 아니라 SQL Developer에서는 Oracle Application Express 3.0.1(Oracle APEX)도 지원합니다.

# SQL Developer 사양

- Oracle Database 11g Release 2와 함께 제공
- Java로 개발됨
- Windows, Linux 및 Mac OS X 플랫폼 지원
- JDBC Thin 드라이버를 사용하여 기본 연결 제공
- Oracle Database 9.2.0.1 이상 버전에 연결
- 다음 링크에서 무료 다운로드 가능
  - [http://www.oracle.com/technology/products/database/sql\\_developer/index.html](http://www.oracle.com/technology/products/database/sql_developer/index.html)



Copyright © 2009, Oracle. All rights reserved.

## SQL Developer 사양

Oracle SQL Developer 1.5는 Oracle Database 11g Release 2와 함께 제공됩니다. SQL Developer는 Oracle JDeveloper 통합 개발 환경(IDE)을 활용하여 Java로 개발되었습니다. 교차 플랫폼 도구로, 이 도구는 Windows, Linux 및 Mac OS(운영 체제) X 플랫폼에서 실행됩니다.

또한 JDBC(Java Database Connectivity) Thin 드라이버를 통해 데이터베이스에 대한 기본 연결이 제공되므로 Oracle Home이 필요하지 않습니다. SQL Developer에는 Installer가 필요 없으며 다운로드 한 파일의 압축을 풀기만 하면 됩니다. SQL Developer 유저는 Oracle Database 9.2.0.1 이상 버전 및 Express Edition을 포함한 모든 Oracle Database Edition에 연결할 수 있습니다.

### 참고

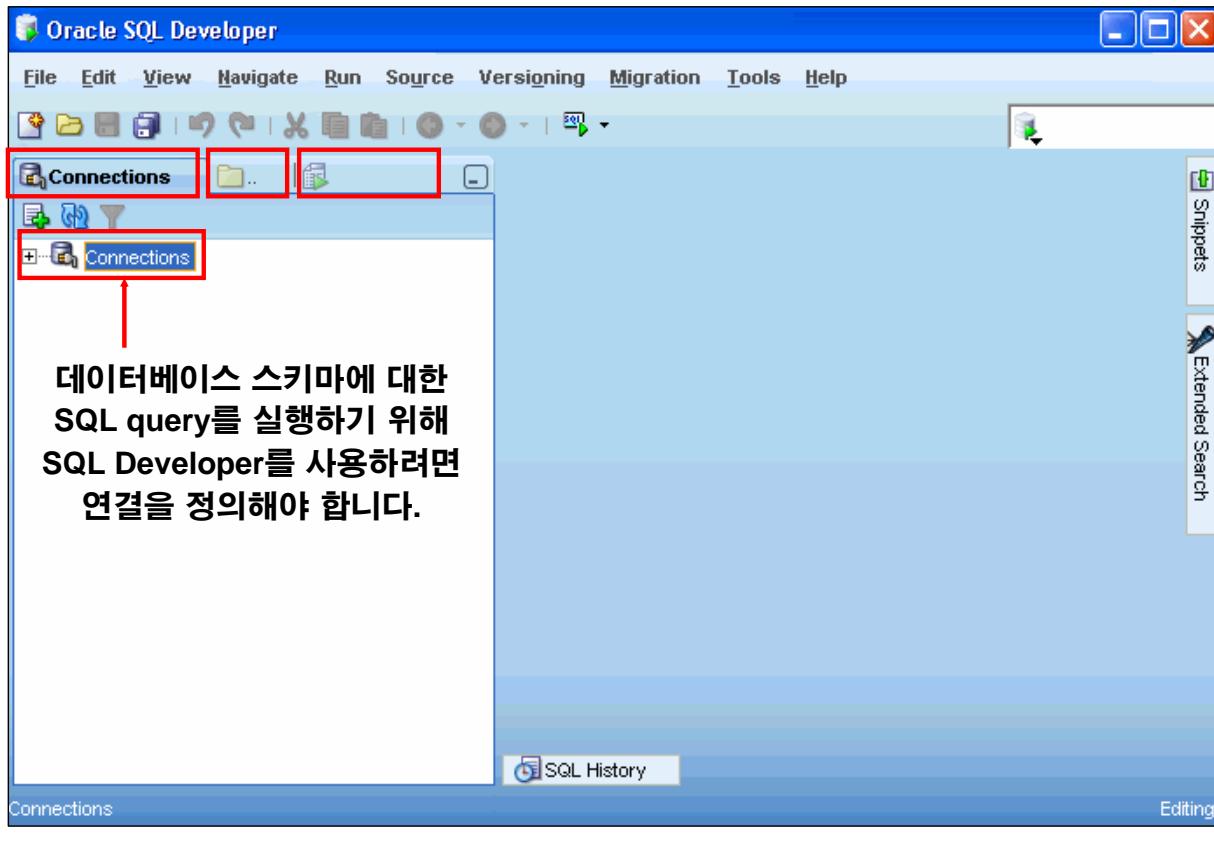
Oracle Database 11g Release 2 이전의 Oracle Database 버전을 사용하는 경우 SQL Developer를 다운로드하여 설치해야 합니다. SQL Developer 1.5는 다음 링크에서 무료로 다운로드할 수 있습니다.

[http://www.oracle.com/technology/products/database/sql\\_developer/index.html](http://www.oracle.com/technology/products/database/sql_developer/index.html).

SQL Developer를 설치하는 방법을 보려면 다음 웹 사이트를 참조하십시오.

[http://download.oracle.com/docs/cd/E12151\\_01/index.htm](http://download.oracle.com/docs/cd/E12151_01/index.htm)

# SQL Developer 1.5 인터페이스



## SQL Developer 1.5 인터페이스

SQL Developer 1.5에는 다음과 같은 세 개의 기본 탐색 템(왼쪽부터)이 있습니다.

- Connections 템:** 이 템을 사용하면 액세스할 수 있는 데이터베이스 객체 및 유저를 찾아볼 수 있습니다.
- Files 템:** Files 폴더 아이콘으로 식별됩니다. 이 템을 사용하면 File > Open 메뉴를 사용하지 않고도 로컬 시스템에서 파일에 액세스할 수 있습니다.
- Reports 템:** Reports 아이콘으로 식별됩니다. 이 템을 사용하면 미리 정의된 보고서를 실행하거나 사용자 보고서를 직접 작성하고 추가할 수 있습니다.

### 일반 탐색 및 사용

SQL Developer는 왼쪽 탐색 영역에서 객체를 찾아 선택하면 오른쪽 탐색 영역에 선택한 객체에 대한 정보가 표시됩니다. 환경 설정을 통해 SQL Developer의 다양한 모양과 동작을 커스터마이즈할 수 있습니다.

**참고:** 데이터베이스 스키마에 연결하여 SQL Query를 실행하거나 프로시저/함수를 실행하려면 연결을 하나 이상 정의해야 합니다.

## SQL Developer 1.5 인터페이스(계속)

### 메뉴

다음 메뉴에는 표준 항목과 SQL Developer 특정 기능에 대한 항목이 있습니다.

- **View:** SQL Developer 인터페이스에 표시되는 사항에 영향을 주는 옵션이 있습니다.
- **Navigate:** window 이동 및 서브 프로그램 실행을 위한 옵션이 있습니다.
- **Run:** 함수나 프로시저가 선택될 때 연관되는 Run File 및 Execution Profile 옵션과 디버깅 옵션이 있습니다.
- **Source:** 함수와 프로시저를 편집할 때 사용하는 옵션이 있습니다.
- **Versioning:** CVS(Concurrent Versions System) 및 Subversion과 같은 버전 관리 및 소스 제어 시스템을 통합 지원합니다.
- **Migration:** Third-party 데이터베이스를 오라클로 이전할 때 관련되는 옵션이 있습니다.
- **Tools:** SQL\*Plus, Preferences 및 SQL Worksheet 등의 SQL Developer 도구를 호출합니다.

**참고:** Run 메뉴에는 디버깅용으로 함수나 프로시저가 선택될 때 연관되는 옵션이 있습니다.

이러한 옵션은 1.2 버전의 Debug 메뉴에 있는 옵션과 동일합니다.

## 데이터베이스 연결 생성

- SQL Developer를 사용하려면 하나 이상의 데이터베이스 연결이 있어야 합니다.
- 다음 대상에 대해 연결을 생성하고 테스트할 수 있습니다.
  - 하나 이상의 데이터베이스
  - 하나 이상의 스키마
- SQL Developer는 시스템의 `tnsnames.ora` 파일에 정의된 모든 연결을 자동으로 임포트합니다.
- XML(Extensible Markup Language) 파일로 연결을 эксп포트할 수 있습니다.
- 추가로 생성된 각각의 데이터베이스 연결은 Connections Navigator 계층에 나열됩니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 데이터베이스 연결 생성

연결은 특정 데이터베이스에 해당 데이터베이스의 특정 유저로 연결하는 데 필요한 정보를 지정하는 SQL Developer 객체입니다. SQL Developer를 사용하려면 하나 이상의 데이터베이스 연결이 있어야 하는데, 이러한 연결은 기존 연결일 수도 있고 새로 생성하거나 임포트할 수도 있습니다.

여러 데이터베이스 및 여러 스키마에 대해 연결을 생성하고 테스트할 수 있습니다.

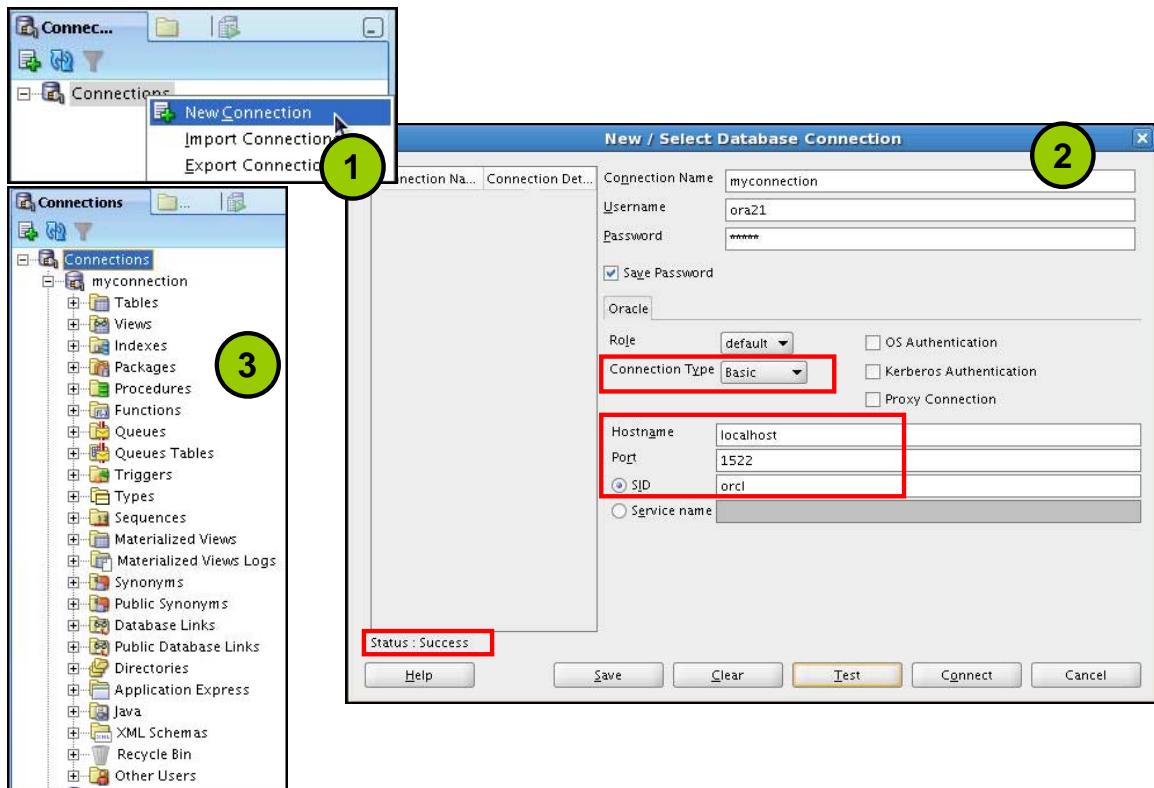
기본적으로 `tnsnames.ora` 파일은 `$ORACLE_HOME/network/admin` 디렉토리에 있지만, `TNS_ADMIN` 환경 변수 또는 레지스트리 값에 지정된 디렉토리에 있을 수도 있습니다. SQL Developer를 시작하고 Database Connections 대화상자를 표시하면 SQL Developer는 시스템의 `tnsnames.ora` 파일에 정의된 모든 연결을 자동으로 임포트합니다.

**참고:** Windows에서 `tnsnames.ora` 파일이 존재하지만 SQL Developer가 해당 연결을 사용하고 있지 않으면 `TNS_ADMIN`을 시스템 환경 변수로 정의하십시오.

연결을 XML 파일로 эксп포트하여 재사용할 수 있습니다.

동일한 데이터베이스에 다른 유저로 연결하거나 다른 데이터베이스에 연결하도록 추가 연결을 생성할 수 있습니다.

# 데이터베이스 연결 생성



ORACLE

Copyright © 2009, Oracle. All rights reserved.

## 데이터베이스 연결 생성(계속)

데이터베이스 연결을 생성하려면 다음 단계를 수행하십시오.

1. Connections 탭 페이지에서 **Connections**를 마우스 오른쪽 버튼으로 누르고 **New Connection**을 선택합니다.
2. New>Select Database Connection window에 연결 이름을 입력합니다. 연결하려는 스키마의 Username 및 암호를 입력합니다.

a) Role drop-down list에서 기본값 또는 SYSDBA를 선택할 수 있습니다. sys 유저 또는 데이터베이스 관리자 권한을 가진 유저의 경우 SYSDBA를 선택합니다.

b) 연결 유형은 다음 중에서 선택하면 됩니다.

**Basic:** 이 유형의 경우 연결하려는 데이터베이스의 호스트 이름 및 SID를 입력합니다. 포트는 이미 1521로 설정되어 있습니다. 원격 데이터베이스 연결을 사용하는 경우에는 서비스 이름을 직접 입력할 수도 있습니다.

**TNS:** tnsnames.ora 파일에서 임포트한 데이터베이스 alias 중 하나를 선택할 수 있습니다.

**LDAP:** Oracle Identity Management의 구성 요소인 Oracle Internet Directory에서 데이터베이스 서비스를 조회할 수 있습니다.

**Advanced:** 데이터베이스에 연결할 사용자 정의 JDBC URL을 정의할 수 있습니다.

c) Test를 눌러 연결이 올바르게 설정되었는지 확인합니다.

d) Connect를 누릅니다.

## 데이터베이스 연결 생성(계속)

Save Password 체크 박스를 선택하면 암호가 XML 파일에 저장됩니다. 따라서 SQL Developer 연결을 닫았다가 다시 열 때 암호를 입력하라는 메시지가 표시되지 않습니다.

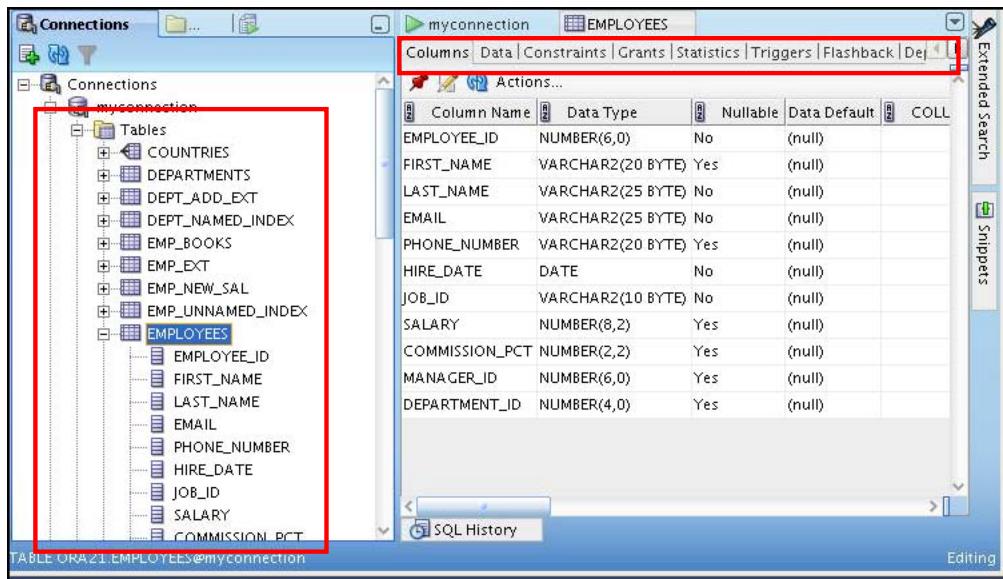
3. Connections Navigator에 연결이 추가됩니다. 연결을 확장하여 데이터베이스 객체를 볼 수 있으며 종속성, 상세 내역, 통계 등의 객체 정의를 볼 수 있습니다.

**참고:** 같은 New>Select Database Connection window에서 Access, MySQL 및 SQL Server 탭을 사용하여 third-party 데이터 소스에 대한 연결을 정의할 수 있습니다. 그러나 이러한 연결은 읽기 전용 연결이며 해당 데이터 소스에서 객체 및 데이터를 찾아볼 수만 있습니다.

## 데이터베이스 객체 탐색

Connections Navigator를 사용하여 다음을 수행할 수 있습니다.

- 데이터베이스 스키마에서 여러 객체 탐색
- 여러 객체의 정의를 한 번에 검토



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 데이터베이스 객체 탐색

데이터베이스 연결을 생성한 후 Connections Navigator를 사용하여 데이터베이스 스키마에서 테이블, 뷰, 인덱스, 패키지, 프로시저, 트리거, 유형 등의 여러 객체를 탐색할 수 있습니다.

SQL Developer는 왼쪽 탐색 영역에서 객체를 찾아 선택하면 오른쪽 탐색 영역에 선택한 객체에 대한 정보가 표시됩니다. 환경 설정을 통해 SQL Developer의 다양한 모양을 커스터마이즈할 수 있습니다.

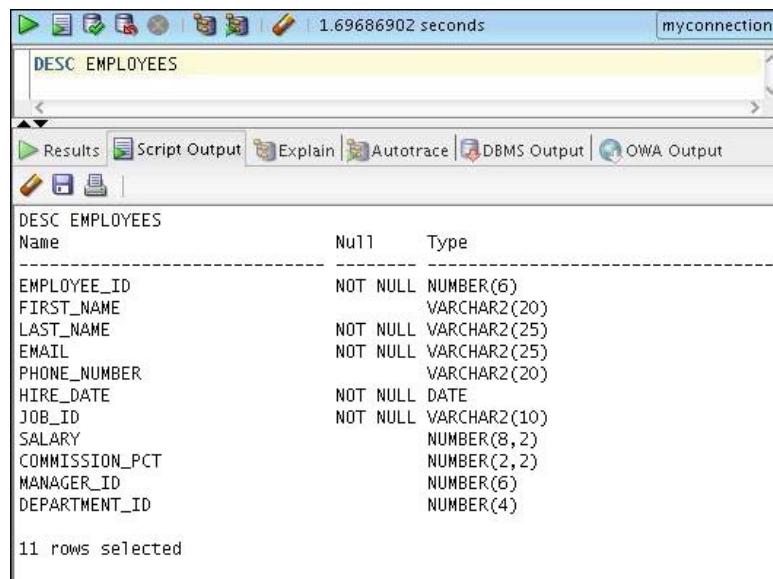
데이터 딕셔너리에서 추출된 객체 정의가 여러 정보 템에 분할되어 있는 것을 볼 수 있습니다. 예를 들어, Navigator에서 테이블을 선택하면 읽기 쉬운 템 페이지에 열, 제약 조건, 권한 부여, 통계, 트리거 등에 대한 세부 정보가 표시됩니다.

슬라이드에서처럼 EMPLOYEES 테이블의 정의를 보려면 다음 단계를 수행하십시오.

1. Connections Navigator에서 Connections 노드를 확장합니다.
2. Tables를 확장합니다.
3. EMPLOYEES를 누릅니다. 기본적으로 Columns 템이 선택되며, 이 템에는 해당 테이블의 열 설명이 표시됩니다. Data 템을 통해 테이블 데이터를 볼 수 있으며 새 행을 입력하고, 데이터를 갱신하고, 해당 변경 내용을 데이터베이스로 커밋할 수 있습니다.

# 테이블 구조 표시

**DESCRIBE 명령을 사용하여 테이블의 구조를 표시합니다.**



The screenshot shows the SQL Developer interface with the query `DESC EMPLOYEES` entered in the top text area. The results pane displays the structure of the `EMPLOYEES` table:

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8, 2)
COMMISSION_PCT		NUMBER(2, 2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

Below the table, it says `11 rows selected`.

ORACLE®

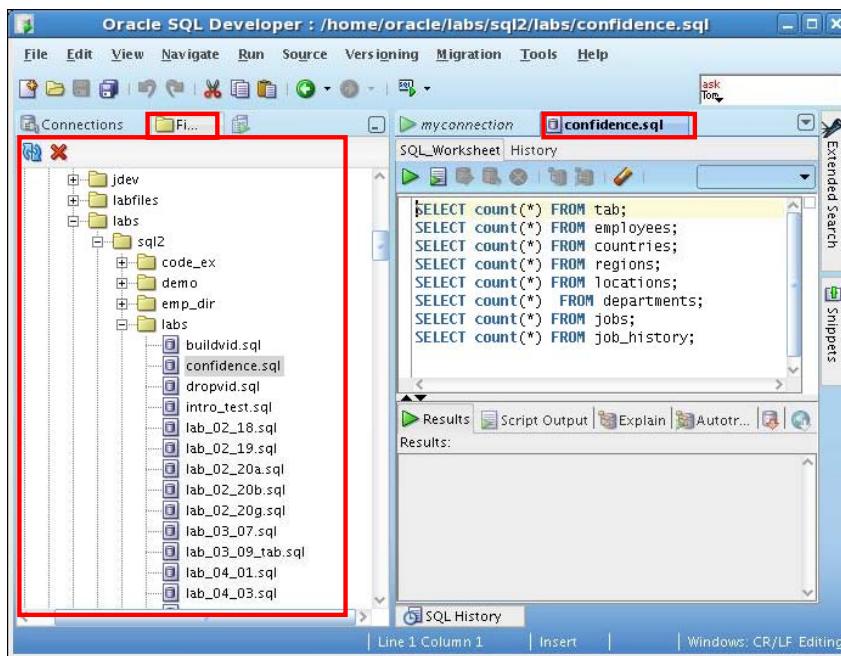
Copyright © 2009, Oracle. All rights reserved.

## 테이블 구조 표시

SQL Developer에서 DESCRIBE 명령을 사용하여 테이블 구조를 표시할 수도 있습니다. 이 명령의 결과로 열 이름, 데이터 유형 및 열에 반드시 데이터가 포함되어야 하는지 여부가 표시됩니다.

## 파일 탐색

File Navigator를 사용하여 파일 시스템을 탐색하고 시스템 파일을 열 수 있습니다.



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

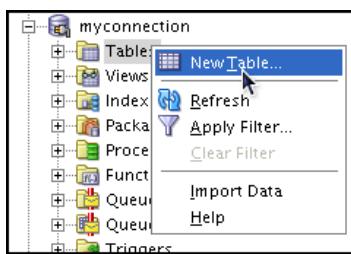
### 데이터베이스 객체 탐색

File Navigator를 사용하여 시스템 파일을 찾아서 열 수 있습니다.

- Files Navigator를 보려면 Files 탭을 누르거나 View > Files를 선택합니다.
- 파일 내용을 보려면 파일 이름을 두 번 눌러 SQL Worksheet 영역에 파일 내용을 표시합니다.

## 스키마 객체 생성

- SQL Developer는 다음 방법을 통한 스키마 객체 생성을 지원합니다.
  - SQL Worksheet에서 SQL 문 실행
  - 컨텍스트 메뉴 사용
- 편집 대화상자나 문맥에 따른 여러 가지 메뉴 중 하나를 사용하여 객체를 편집합니다.
- 새 객체 생성 또는 기존 스키마 객체 편집과 같은 조정을 위해 DDL(데이터 정의어)을 봅니다.



**ORACLE**

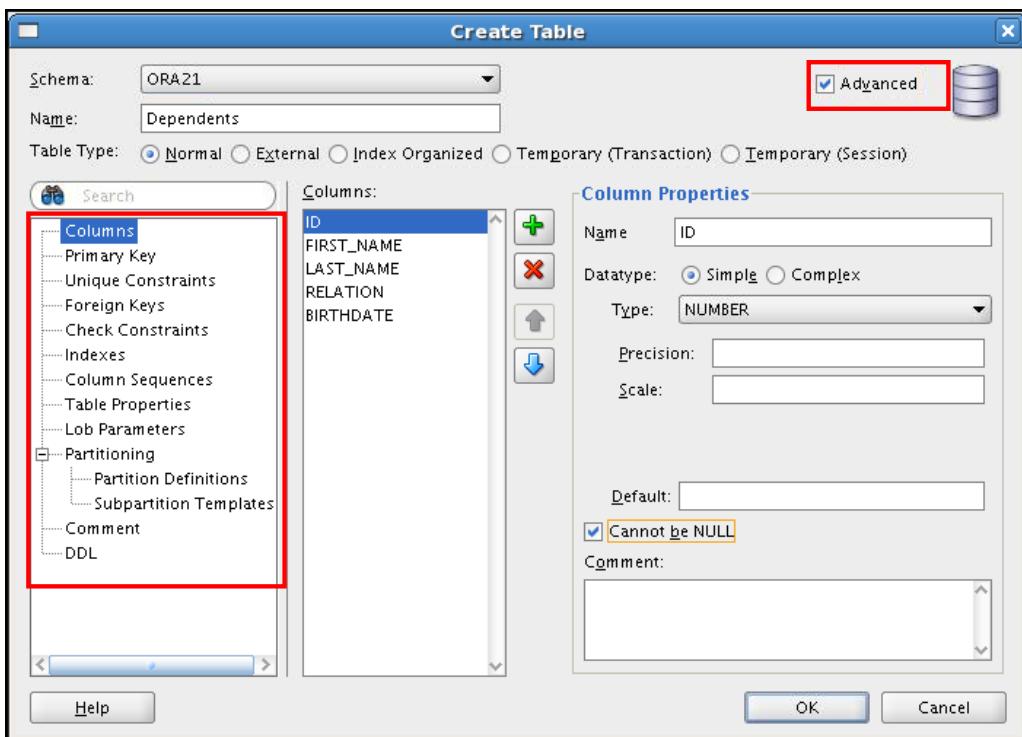
Copyright © 2009, Oracle. All rights reserved.

### 스키마 객체 생성

SQL Developer는 SQL Worksheet에서 SQL 문을 실행하여 스키마 객체 생성을 지원합니다. 또한 컨텍스트 메뉴를 사용하여 객체를 생성할 수도 있습니다. 객체를 생성한 후 편집 대화상자나 문맥에 따른 여러 메뉴 중 하나를 사용하여 객체를 편집할 수 있습니다. 새 객체를 생성하거나 기존 객체를 편집할 때 이러한 조정 작업을 위한 DDL을 확인할 수 있습니다. 스키마의 여러 객체에 대한 전체 DDL을 생성하려는 경우 Export DDL 옵션을 사용할 수 있습니다.

슬라이드에서는 컨텍스트 메뉴를 사용하여 테이블을 생성하는 방법을 보여줍니다. 새 테이블 생성 대화상자를 열려면 Tables를 마우스 오른쪽 버튼으로 누르고 New Table을 선택합니다. 데이터베이스 객체를 생성하고 편집하는 대화상자에는 여러 탭이 있으며 각 탭은 해당 객체 유형의 논리적 속성 그룹화를 반영합니다.

## 새 테이블 생성: 예제



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 새 테이블 생성: 예제

Create Table 대화상자에서 Advanced 체크 박스를 선택하지 않은 경우, 열과 자주 사용하는 일부 기능을 지정하여 빠르게 테이블을 생성할 수 있습니다.

Advanced 체크 박스를 선택한 경우에는 Create Table 대화상자에 여러 옵션이 표시되며, 여기에서 테이블을 생성하는 동안 확장된 기능을 지정할 수 있습니다.

슬라이드의 예제는 Advanced 체크 박스를 선택하여 DEPENDENTS 테이블을 생성하는 방법을 보여줍니다.

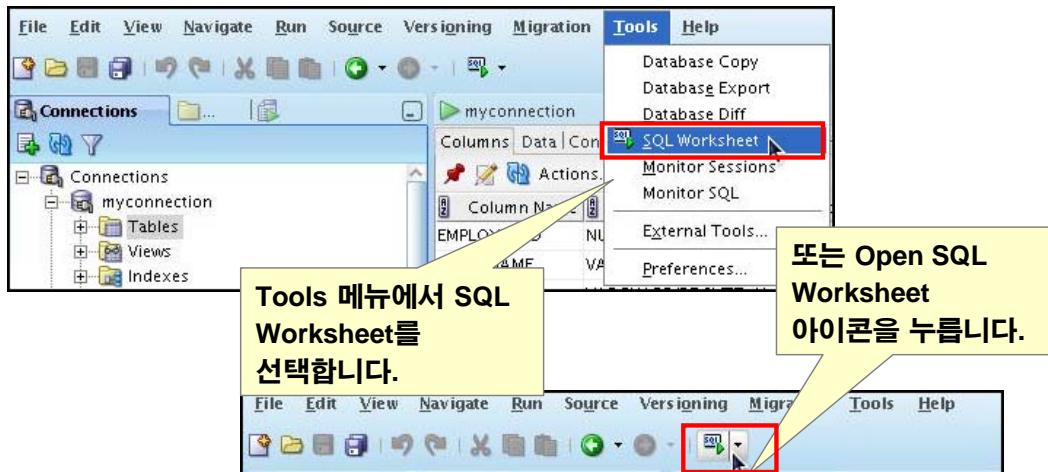
새 테이블을 생성하려면 다음 단계를 수행하십시오.

1. Connections Navigator에서 Tables를 마우스 오른쪽 버튼으로 누릅니다.
2. Create TABLE을 선택합니다.
3. Create Table 대화상자에서 Advanced를 선택합니다.
4. 열 정보를 지정합니다.
5. OK를 누릅니다.

또한 필수 사항은 아니지만 대화상자의 Primary Key 탭을 사용하여 Primary Key를 지정하는 것이 좋습니다. 생성한 테이블을 편집하려면 Connections Navigator에서 테이블을 마우스 오른쪽 버튼으로 누르고 Edit를 선택합니다.

# SQL Worksheet 사용

- SQL Worksheet를 사용하여 SQL, PL/SQL 및 SQL\*Plus 문을 입력하고 실행합니다.
- 워크시트와 연관된 데이터베이스 연결로 처리할 수 있는 모든 작업을 지정합니다.



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## SQL Worksheet 사용

데이터베이스에 연결하면 해당 연결에 대한 SQL Worksheet window가 자동으로 열립니다. SQL Worksheet를 사용하여 SQL, PL/SQL 및 SQL\*Plus 문을 입력하고 실행할 수 있습니다. SQL Worksheet는 특정 범위까지 SQL\*Plus 문을 지원합니다. SQL Worksheet에서 지원되지 않는 SQL\*Plus 문은 무시되고 데이터베이스에 전달되지 않습니다.

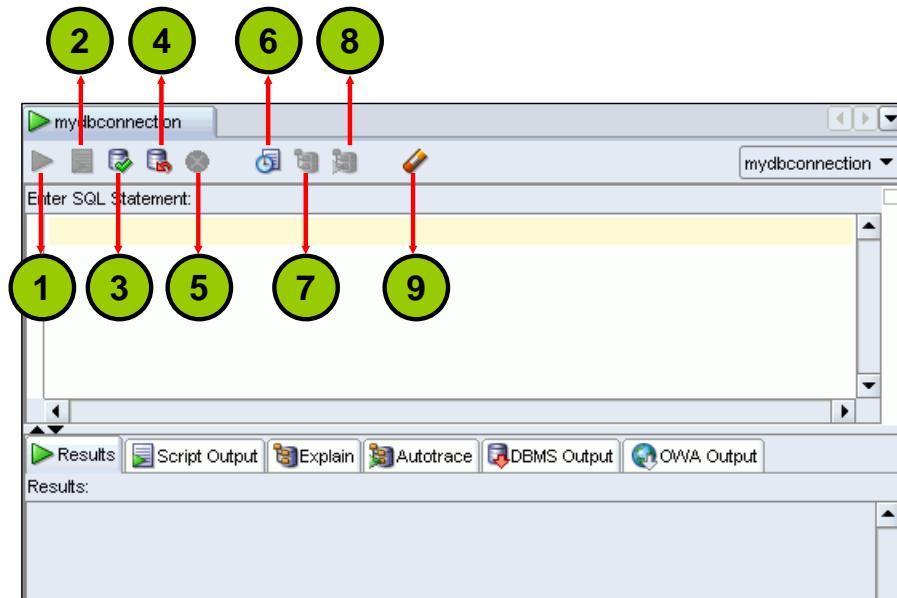
다음과 같이 워크시트와 연관된 데이터베이스 연결에 의해 처리될 수 있는 작업을 지정합니다.

- 테이블 생성
- 데이터 삽입
- 트리거 생성 및 편집
- 테이블에서 데이터 선택
- 파일에 선택한 데이터 저장

다음 중 하나를 사용하여 SQL Worksheet를 표시할 수 있습니다.

- Tools > SQL Worksheet를 선택합니다.
- Open SQL Worksheet 아이콘을 누릅니다.

# SQL Worksheet 사용



**ORACLE®**

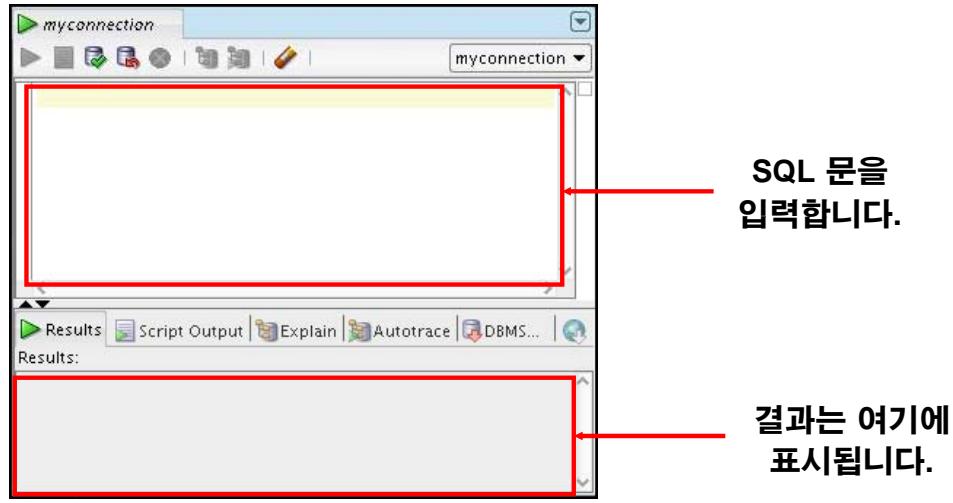
Copyright © 2009, Oracle. All rights reserved.

## SQL Worksheet 사용(계속)

단축키 또는 아이콘을 사용하여 SQL 문 실행, 스크립트 실행, 실행한 SQL 문 기록 보기 등의 특정 작업을 수행하는 경우가 있습니다. 여러 아이콘이 있는 SQL Worksheet 도구 모음을 사용하여 다음 작업을 수행할 수 있습니다.

1. **Execute Statement:** Enter SQL Statement 상자에서 커서가 위치한 명령문을 실행합니다. SQL 문에 바인드 변수를 사용할 수 있지만 치환 변수는 사용할 수 없습니다.
2. **Run Script:** Script Runner를 사용하여 Enter SQL Statement 상자에 있는 모든 명령문을 실행합니다. SQL 문에 치환 변수를 사용할 수 있지만 바인드 변수는 사용할 수 없습니다.
3. **Commit:** 데이터베이스에 대한 모든 변경 사항을 기록하고 트랜잭션을 종료합니다.
4. **Rollback:** 데이터베이스에 대한 모든 변경 사항을 데이터베이스에 기록하지 않은 채 폐기하고 트랜잭션을 종료합니다.
5. **Cancel:** 현재 실행 중인 모든 명령문의 실행을 정지합니다.
6. **SQL History:** 실행한 SQL 문에 대한 정보가 있는 대화상자를 표시합니다.
7. **Execute Explain Plan:** Explain 탭을 눌러 볼 수 있는 실행 계획을 생성합니다.
8. **Autotrace:** 명령문에 대한 추적 정보를 생성합니다.
9. **Clear:** Enter SQL Statement 상자에서 명령문을 지웁니다.

# SQL Worksheet 사용



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## SQL Worksheet 사용(계속)

데이터베이스에 연결하면 해당 연결에 대한 SQL Worksheet window가 자동으로 열립니다. SQL Worksheet를 사용하여 SQL, PL/SQL 및 SQL\*Plus 문을 입력하고 실행할 수 있습니다. 모든 SQL 및 PL/SQL 명령이 지원되며 이러한 명령은 SQL Worksheet에서 오라클 데이터베이스로 직접 전달됩니다. SQL Developer에서 사용되는 SQL\*Plus 명령은 데이터베이스로 전달하기 전에 SQL Worksheet에서 해석되어야 합니다.

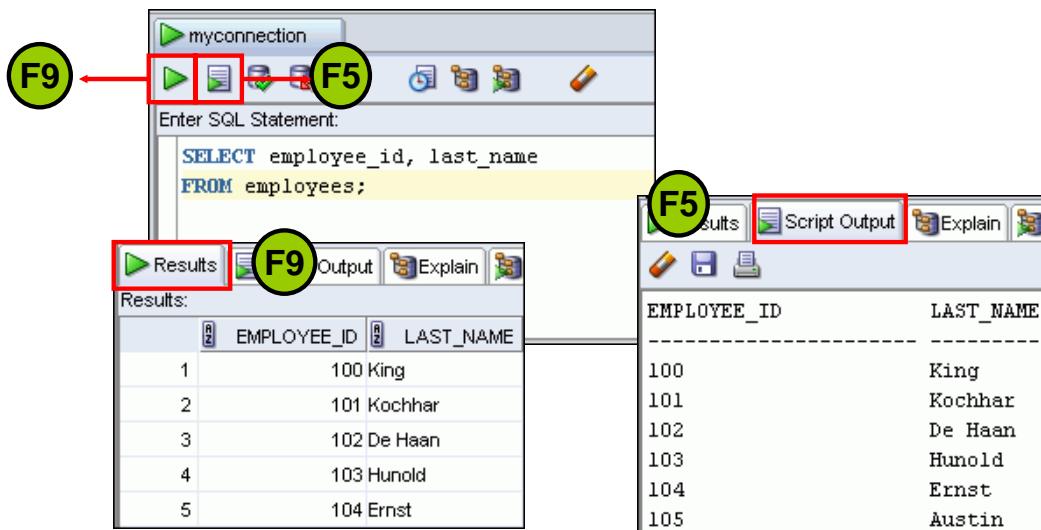
현재 SQL Worksheet에서는 다수의 SQL\*Plus 명령을 지원합니다. SQL Worksheet에서 지원되지 않는 명령은 무시되며 오라클 데이터베이스로 보내지지 않습니다. SQL Worksheet를 통해 SQL 문을 실행하거나 SQL\*Plus 명령 중 일부를 실행할 수 있습니다.

다음 옵션 중 하나를 사용하여 SQL Worksheet를 표시할 수 있습니다.

- Tools > SQL Worksheet를 선택합니다.
- Open SQL Worksheet 아이콘을 누릅니다.

# SQL 문 실행

Enter SQL Statement 상자를 사용하여 SQL 문을 하나 또는 여러 개 입력합니다.



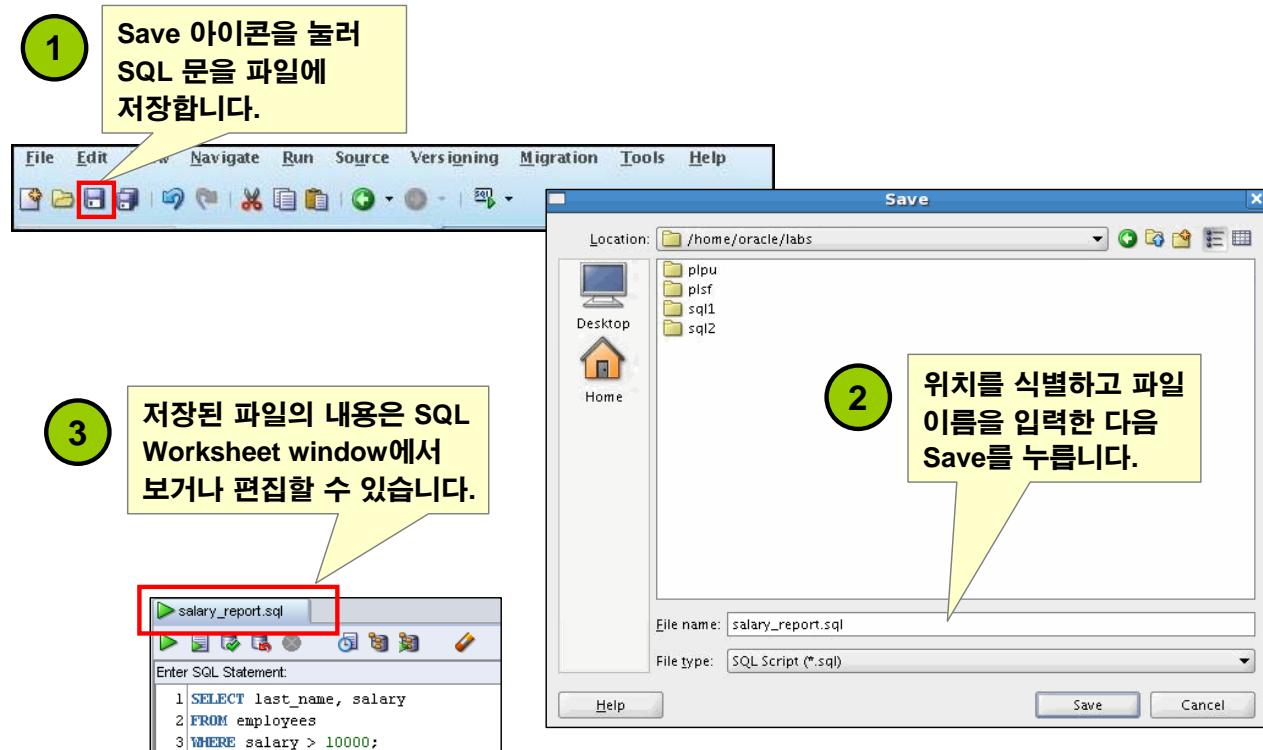
ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## SQL 문 실행

슬라이드의 예제에서는 동일한 Query에 대해 [F9] 키 또는 Execute Statement를 사용할 때와 [F5] 또는 Run Script를 사용할 때 출력의 차이를 보여줍니다.

# SQL 스크립트 저장



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## SQL 스크립트 저장

SQL 문을 SQL Worksheet에서 텍스트 파일로 저장할 수 있습니다. Enter SQL Statement 상자의 컨텐트를 저장하려면 다음 단계를 따르십시오.

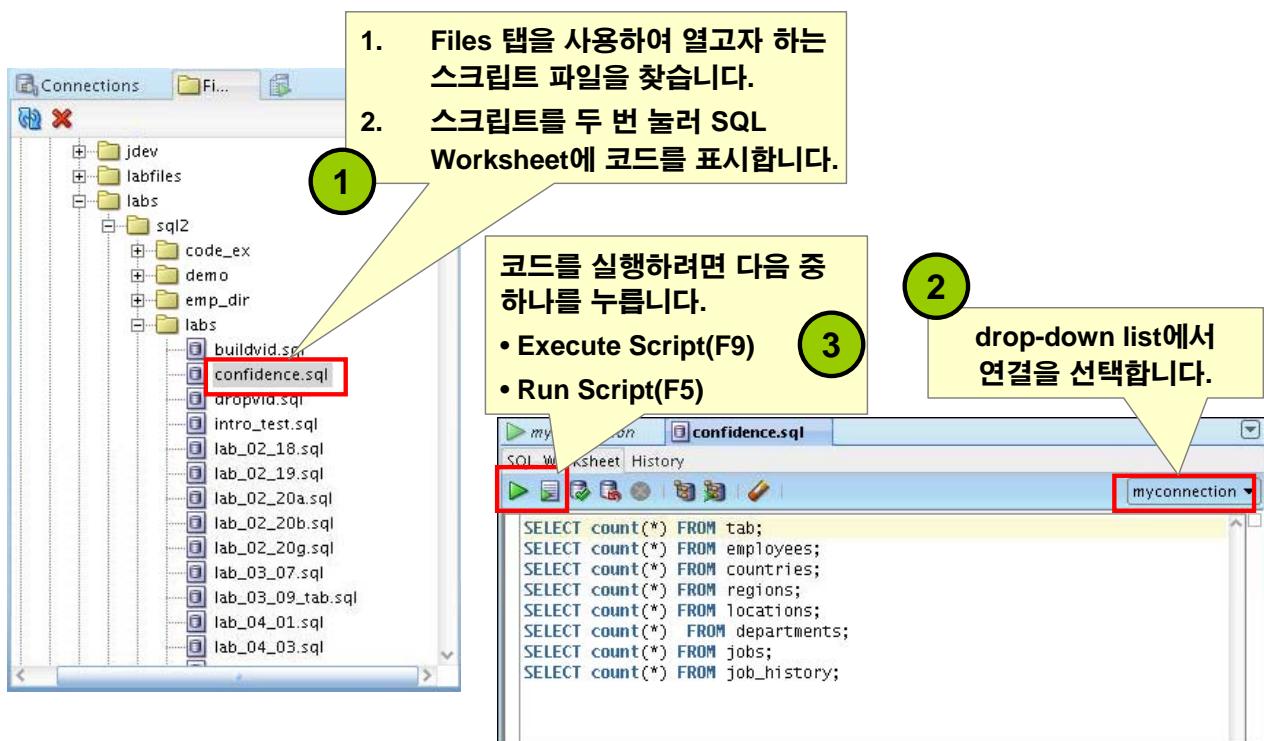
1. Save 아이콘을 누르거나 File > Save 메뉴 항목을 사용합니다.
2. Save 대화상자에서 파일 이름과 파일을 저장할 위치를 입력합니다.
3. Save를 누릅니다.

내용을 파일에 저장하면 Enter SQL Statement window가 파일 내용의 탭 페이지를 표시합니다. 여러 개의 파일을 동시에 열 수 있으며 각 파일은 탭 페이지로 표시됩니다.

## 스크립트 경로

스크립트를 찾고 저장할 기본 경로를 선택할 수 있습니다. Tools > Preferences > Database > Worksheet Parameters 아래에서 "Select default path to look for scripts" 필드에 값을 입력하십시오.

## 저장된 SQL 스크립트 실행: 방법 1



ORACLE

Copyright © 2009, Oracle. All rights reserved.

### 저장된 SQL 스크립트 실행: 방법 1

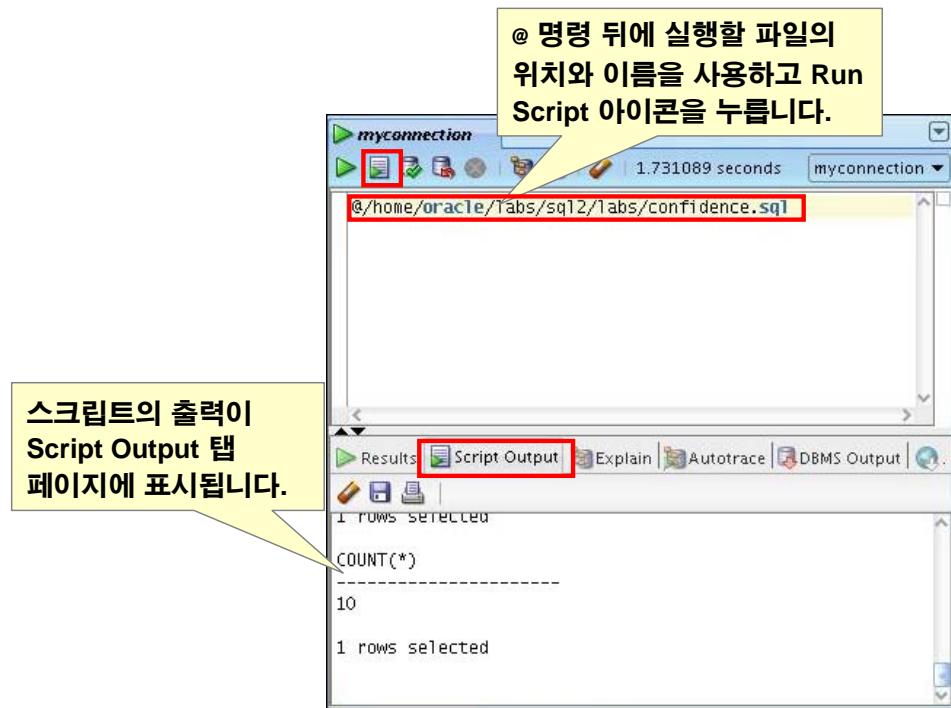
스크립트 파일을 열고 SQL Worksheet 영역에 코드를 표시하려면 다음 단계를 수행합니다.

1. Open 대화상자에서 열고자 하는 스크립트 파일을 선택합니다. 또는 해당 파일이 있는 위치로 이동합니다.
2. 두 번 눌러 엽니다. 스크립트 파일의 코드가 SQL Worksheet 영역에 표시됩니다.
3. connection drop-down list에서 연결을 선택합니다.
4. 코드를 실행하려면 SQL Worksheet 도구 모음에서 Run Script (F5) 아이콘을 누릅니다. connection drop-down list에서 연결을 선택하지 않은 경우 connection 대화상자가 나타납니다. 스크립트 실행에 사용할 연결을 선택합니다.

또는 다음을 수행할 수 있습니다.

1. File > Open을 선택합니다. Open 대화상자가 표시됩니다.
2. Open 대화상자에서 열고자 하는 스크립트 파일을 선택합니다. (또는 해당 파일이 있는 위치로 이동합니다.)
3. Open을 누릅니다. 스크립트 파일의 코드가 SQL Worksheet 영역에 표시됩니다.
4. connection drop-down list에서 연결을 선택합니다.
5. 코드를 실행하려면 SQL Worksheet 도구 모음에서 Run Script (F5) 아이콘을 누릅니다. connection drop-down list에서 연결을 선택하지 않은 경우 connection 대화상자가 나타납니다. 스크립트 실행에 사용할 연결을 선택합니다.

## 저장된 SQL 스크립트 실행: 방법 2



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

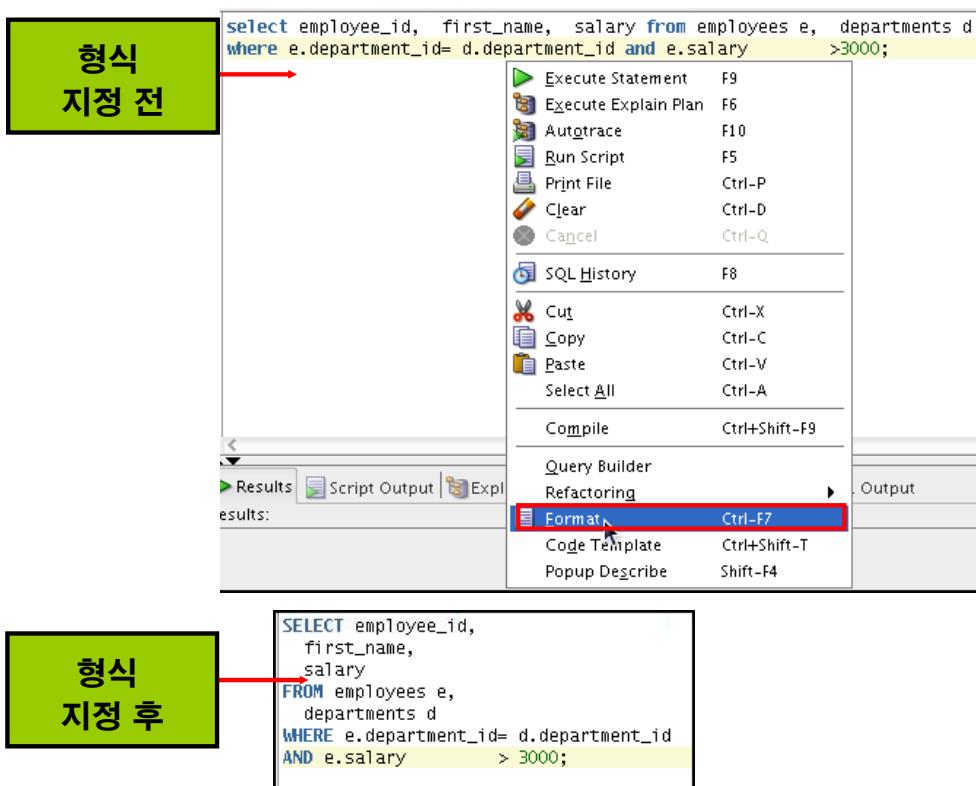
### 저장된 SQL 스크립트 실행: 방법 2

저장된 SQL 스크립트를 실행하려면 다음 단계를 수행합니다.

1. Enter SQL Statement window에서 @ 명령을 사용하고 그 뒤에 실행할 파일의 위치와 이름을 입력합니다.
2. Run Script 아이콘을 누릅니다.

파일의 실행 결과가 Script Output 탭 페이지에 표시됩니다. 또한 Script Output 탭 페이지에서 Save 아이콘을 눌러 스크립트 출력을 저장할 수도 있습니다. File Save 대화상자가 나타나고 파일의 이름 및 위치를 식별할 수 있습니다.

## SQL 코드 형식 지정



ORACLE

Copyright © 2009, Oracle. All rights reserved.

### SQL 코드 형식 지정

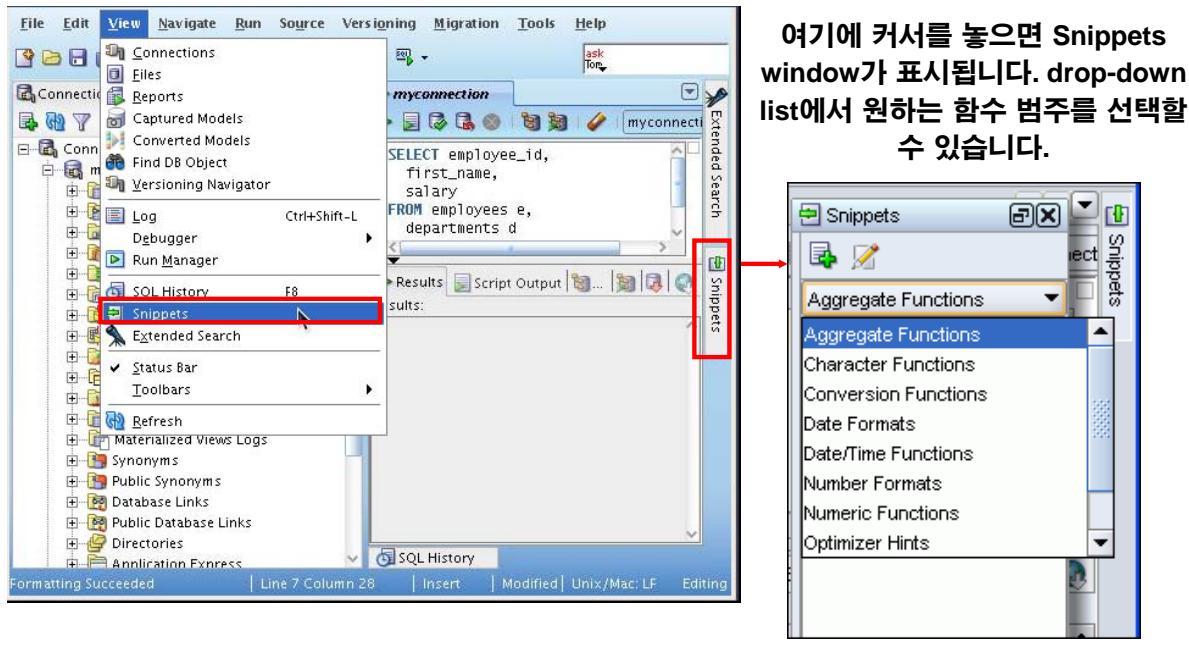
SQL 코드의 들여쓰기, 간격, 대소문자 및 줄 구분 형식을 지정해야 할 경우가 있습니다. SQL Developer에는 SQL 코드의 형식을 지정하는 기능이 있습니다.

SQL 코드에 형식을 지정하려면 명령문 영역을 마우스 오른쪽 버튼으로 누른 다음 Format SQL을 선택합니다.

슬라이드의 예제에서 형식이 지정되기 전의 SQL 코드에는 키워드가 대문자로 표시되지 않고 명령문의 들여쓰기가 제대로 되어 있지 않습니다. 형식 지정 이후의 SQL 코드에서는 키워드가 대문자로 표시되고 명령문이 적절히 들여쓰기되어 보기 좋게 다듬어졌습니다.

# Snippet 사용

**Snippet은 구문이거나 예제일 수 있는 코드 부분입니다.**



여기에서 커서를 놓으면 Snippets window가 표시됩니다. drop-down list에서 원하는 함수 범주를 선택할 수 있습니다.

Copyright © 2009, Oracle. All rights reserved.

ORACLE

## Snippet 사용

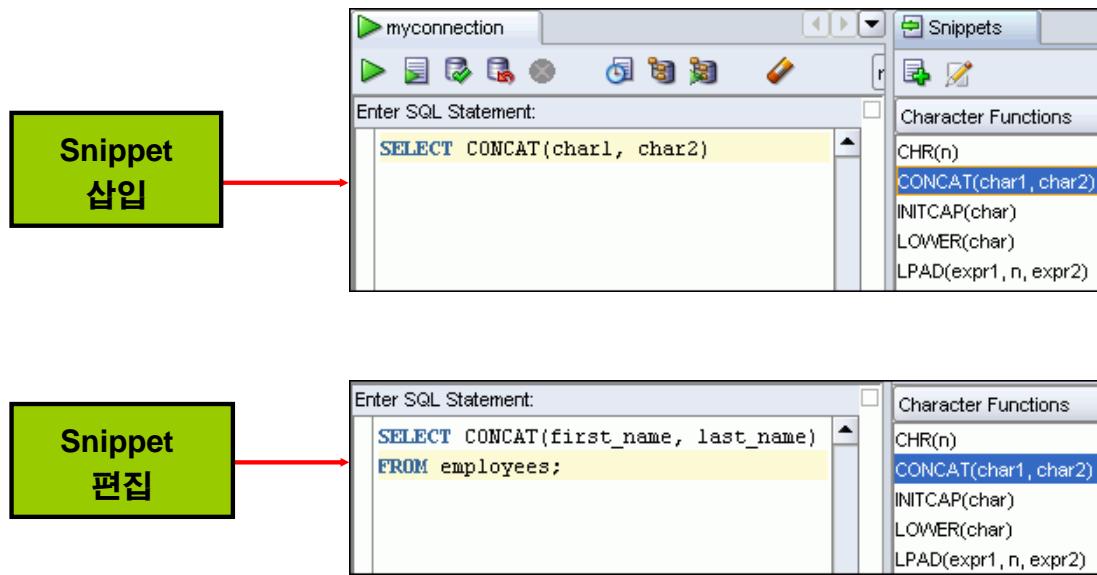
SQL Worksheet을 사용하거나 PL/SQL 함수 또는 프로시저를 생성하거나 편집할 때 특정 코드 부분을 사용해야 하는 경우가 있습니다. SQL Developer에는 Snippet이라는 기능이 있습니다.

Snippet은 SQL 함수, 옵티마이저 힌트 및 기타 PL/SQL 프로그래밍 기법과 같은 코드 부분입니다. Snippet을 Editor window로 끌어올 수 있습니다.

Snippet을 표시하려면 View > Snippets를 선택합니다.

그러면 Snippets window가 오른쪽에 표시됩니다. drop-down list를 사용하여 그룹을 선택할 수 있습니다. Snippets window가 숨겨진 경우 이를 표시할 수 있도록 오른쪽 window 여백에 Snippets 버튼이 표시됩니다.

## Snippet 사용: 예제



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### Snippet 사용: 예제

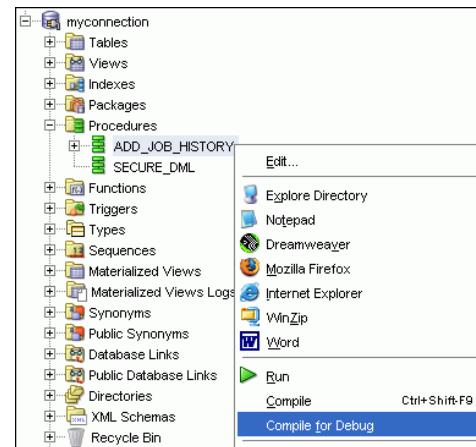
SQL Worksheet 또는 PL/SQL 함수나 프로시저의 코드에 Snippet을 삽입하려면 Snippets window에서 코드의 원하는 위치로 Snippet을 끌어옵니다. 그런 다음 SQL 함수가 현재 컨텍스트에서 유효하도록 구문을 편집할 수 있습니다. 도구 설명에서 SQL 함수에 대한 간단한 설명을 보려면 커서를 함수 이름 위에 둡니다.

슬라이드의 예제는 Snippets window의 Character Functions 그룹에서 CONCAT( char1, char2 )를 끌어오는 과정을 보여줍니다. 이어서 다음과 같이 CONCAT 함수 구문을 편집하고 나머지 명령문을 추가합니다.

```
SELECT CONCAT(first_name, last_name)
FROM employees;
```

## 프로시저 및 함수 디버깅

- **SQL Developer를 사용하여 PL/SQL 함수 및 프로시저를 디버그합니다.**
- **프로시저를 디버그할 수 있도록 PL/SQL 컴파일을 수행하려면 Compile for Debug 옵션을 사용합니다.**
- **중단점을 설정하고 Step Into 및 Step Over 작업을 수행하려면 Debug 메뉴 옵션을 사용합니다.**



**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### 프로시저 및 함수 디버깅

SQL Developer에서는 PL/SQL 프로시저 및 함수를 디버깅할 수 있습니다. Debug 메뉴 옵션을 사용하여 다음 디버깅 작업을 수행할 수 있습니다.

- **Find Execution Point**를 사용하면 다음 실행 지점으로 이동합니다.
- **Resume**을 사용하면 실행이 계속됩니다.
- **Step Over**를 사용하면 다음 메소드를 건너뛰고 해당 메소드 뒤의 다음 명령문으로 이동합니다.
- **Step Into**를 사용하면 다음 메소드의 첫번째 명령문으로 이동합니다.
- **Step Out**을 사용하면 현재 메소드에서 나와 다음 명령문으로 이동합니다.
- **Step to End of Method**를 사용하면 현재 메소드의 마지막 명령문으로 이동합니다.
- **Pause**를 사용하면 실행이 중지되지만 종료되지는 않으므로 나중에 실행을 재개할 수 있습니다.
- **Terminate**를 사용하면 실행이 중지 및 종료됩니다. 이 지점에서는 실행을 재개할 수 없습니다. 대신 함수나 프로시저 시작 부분부터 실행이나 디버깅을 시작하려면 Source 탭 도구 모음의 Run 또는 Debug 아이콘을 누르십시오.
- **Garbage Collection**을 사용하면 캐시에서 무효한 객체가 제거되고 자주 액세스하는 유효한 객체가 사용됩니다.

이러한 옵션은 디버깅 도구 모음에서 아이콘으로도 사용할 수 있습니다.

## 데이터베이스 보고

SQL Developer에서는 데이터베이스 및 해당 객체에 대한 여러 가지 미리 정의된 보고서가 제공됩니다.

The screenshot shows the SQL Developer interface. On the left, the 'Connections' sidebar has its 'Reports' tab selected, indicated by a red box around the icon. The main workspace displays a table titled 'Dependencies' for the connection 'myconnection'. The table lists various database objects with their owners, names, types, and referenced owners. A portion of the table data is as follows:

Owner	Name	Type	Referenced Owner	Referenced Name
CTXSYS	CTX_CLASSES	VIEW	CTXSYS	DR\$CLASS
CTXSYS	CTX_CLS	PACKAGE	SYS	STANDARD
CTXSYS	CTX_DOC	PACKAGE	SYS	STANDARD
CTXSYS	CTX_INDEX_SETS	VIEW	CTXSYS	DR\$INDEX_SET
CTXSYS	CTX_INDEX_SETS	VIEW	SYS	USER\$
CTXSYS	CTX_INDEX_SET_INDEXES	VIEW	CTXSYS	DR\$INDEX_SET
CTXSYS	CTX_INDEX_SET_INDEXES	VIEW	CTXSYS	DR\$INDEX_SET_INDEX
CTXSYS	CTX_INDEX_SET_INDEXES	VIEW	SYS	USER\$
CTXSYS	CTX_OBJECTS	VIEW	CTXSYS	DR\$CLASS
CTXSYS	CTX_OBJECTS	VIEW	CTXSYS	DR\$OBJECT
CTXSYS	CTX_OBJECT_ATTRIBUTES	VIEW	CTXSYS	DR\$CLASS
CTXSYS	CTX_OBJECT_ATTRIBUTES	VIEW	CTXSYS	DR\$OBJECT
CTXSYS	CTX_OBJECT_ATTRIBUTE_LOV	VIEW	CTXSYS	DR\$CLASS
CTXSYS	CTX_OBJECT_ATTRIBUTE_LOV	VIEW	CTXSYS	DR\$OBJECT
CTXSYS	CTX_OBJECT_ATTRIBUTE_LOV	VIEW	CTXSYS	DR\$OBJECT_ATTRIBUTE
CTXSYS	CTX_OBJECT_ATTRIBUTE_LOV	VIEW	CTXSYS	DR\$OBJECT_ATTRIBUTE_LOV
CTXSYS	CTX_PARAMETERS	VIEW	CTXSYS	DR\$PARAMETER

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 데이터베이스 보고

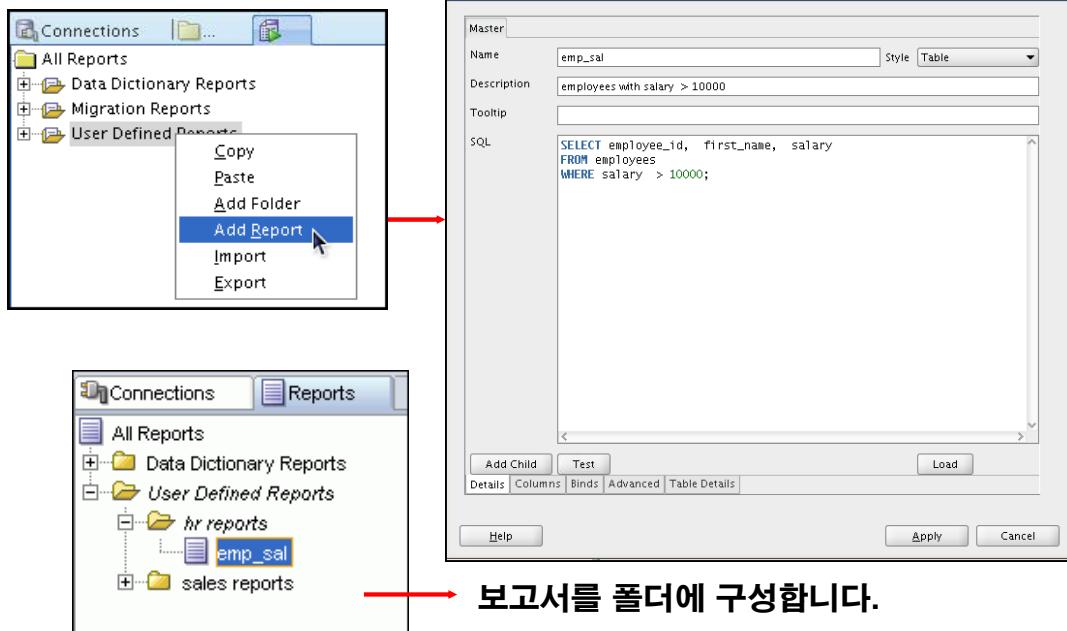
SQL Developer는 데이터베이스 및 객체에 대한 다양한 보고서를 제공합니다. 이러한 보고서는 다음 범주로 그룹화할 수 있습니다.

- About Your Database 보고서
- Database Administration 보고서
- Table 보고서
- PL/SQL 보고서
- Security 보고서
- XML 보고서
- Jobs 보고서
- Streams 보고서
- All Objects 보고서
- Data Dictionary 보고서
- User-Defined 보고서

보고서를 표시하려면 window 왼쪽에 있는 Reports 탭을 누릅니다. 그러면 개별 보고서가 window 오른쪽의 탭 창에 표시되며 각 보고서에 대해 drop-down list를 사용하여 보고서를 표시할 데이터베이스 연결을 선택할 수 있습니다. 객체에 대한 보고서의 경우 선택한 데이터베이스 연결과 연관된 데이터베이스 유저가 볼 수 있는 객체만 표시되고 행은 일반적으로 Owner별로 정렬됩니다. 고유한 유저 정의 보고서를 작성할 수도 있습니다.

## 유저 정의 보고서 작성

반복적으로 사용할 수 있도록 유저 정의 보고서를 생성하고 저장합니다.



**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### 유저 정의 보고서 작성

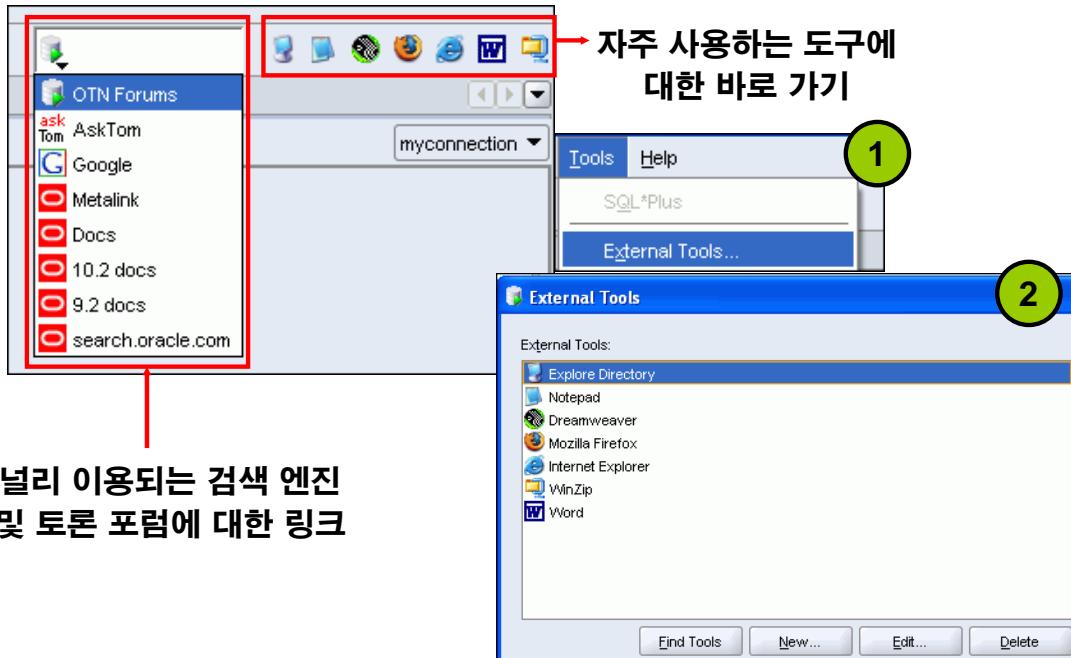
유저 정의 보고서는 SQL Developer 유저가 생성하는 보고서입니다. 유저 정의 보고서를 작성하려면 다음 단계를 수행하십시오.

1. Reports 아래에서 User Defined Reports 노드를 마우스 오른쪽 버튼으로 누르고 Add Report를 선택합니다.
2. Create Report 대화상자에서 보고서 이름을 지정하고 보고서 정보를 검색할 SQL query를 지정합니다. 그런 다음 Apply를 누릅니다.

슬라이드의 예제에서 보고서 이름은 emp\_sal로 지정됩니다. 보고서에 급여  $>= 10000$ 인 사원에 대한 세부 정보가 포함되어 있음을 나타내는 선택적 설명이 제공됩니다. 유저 정의 보고서에 표시할 정보를 검색하기 위한 전체 SQL 문이 SQL 상자에서 지정됩니다. Reports Navigator 화면에서 보고서 이름 위에 잠시 커서를 두면 표시되는 선택적 도구 설명을 포함할 수도 있습니다.

유저 정의 보고서를 폴더에 구성하고 폴더와 하위 폴더의 계층을 생성할 수 있습니다. 유저 정의 보고서에 대한 폴더를 생성하려면 User Defined Reports 노드나 해당 노드 아래에 있는 임의의 폴더 이름을 마우스 오른쪽 버튼으로 누르고 Add Folder를 선택합니다. 이러한 보고서에 대한 폴더를 포함한 유저 정의 보고서에 대한 정보는 유저 특정 정보를 위한 디렉토리 아래 UserReports.xml이라는 파일에 저장됩니다.

# 검색 엔진 및 External 도구



**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

## 검색 엔진 및 External 도구

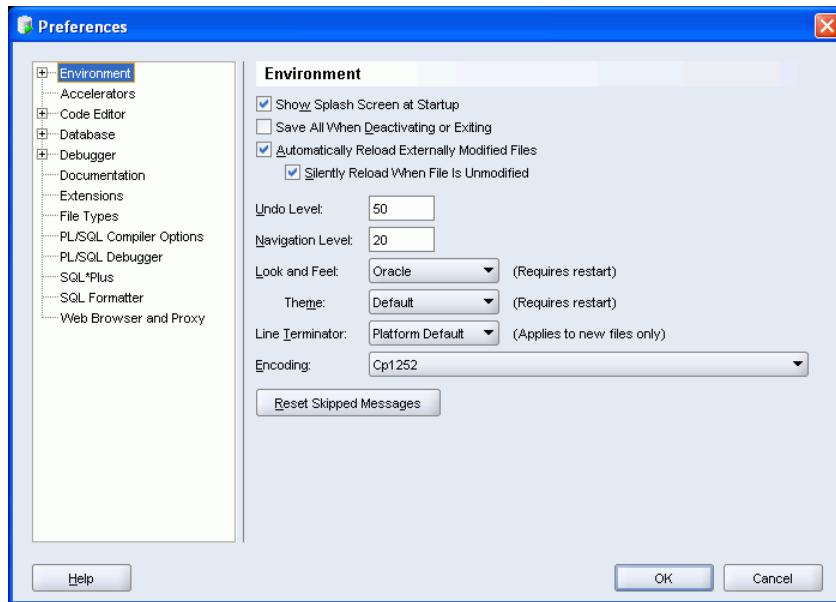
SQL 개발자의 생산성을 높이기 위해 SQL Developer에는 AskTom, Google 등의 유명 검색 엔진과 토의 포럼에 대한 빠른 링크가 포함되었습니다. 또한 메모장, Microsoft Word, Dreamweaver 등의 자주 사용하는 일부 도구에 대한 단축키 아이콘도 사용할 수 있습니다.

기존 리스트에 external 도구를 추가하거나 자주 사용하지 않는 도구에 대한 단축키를 삭제할 수도 있습니다. 이렇게 하려면 다음 단계를 수행하십시오.

1. Tools 메뉴에서 External Tools를 선택합니다.
2. 새 도구를 추가하려면 External Tools 대화상자에서 New를 선택합니다. 리스트에서 도구를 제거하려면 Delete를 선택합니다.

## 환경 설정

- SQL Developer 인터페이스와 환경을 커스터마이즈합니다.
- Tools 메뉴에서 Preferences를 선택합니다.



**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### 환경 설정

SQL Developer 환경 설정을 자신의 취향과 요구 사항에 맞게 수정함으로써 SQL Developer 인터페이스와 환경의 다양한 요소를 커스터마이즈할 수 있습니다. SQL Developer 환경 설정을 수정하려면 Tools와 Preferences를 차례로 선택합니다.

환경 설정은 다음 범주로 분류되어 있습니다.

- Environment
- Accelerators (Keyboard shortcuts)
- Code Editors
- Database
- Debugger
- Documentation
- Extensions
- File Types
- Migration
- PL/SQL Compilers
- PL/SQL Debugger

# SQL Developer 레이아웃 재설정

```
Terminal
File Edit View Terminal Tabs Help
[oracle@EDRSR5P1 ~]$ locate windowinglayout.xml
/home/oracle/.sqldeveloper/system1.5.4.59.40/o.ide.11.1.1.0.22.49.48/windowinglayout.xml
/home/oracle/.sqldeveloper/system1.5.4.59.41/o.ide.11.1.1.0.22.49.48/windowinglayout.xml
[oracle@EDRSR5P1 ~]$ cd /home/oracle/.sqldeveloper/system1.5.4.59.41/o.ide.11.1.1.0.22.49.48
[oracle@EDRSR5P1 o.ide.11.1.1.0.22.49.48]$ ls
Debugging.layout Editing.layout projects windowinglayout.xml
dtcache.xml preferences.xml settings.xml
[oracle@EDRSR5P1 o.ide.11.1.1.0.22.49.48]$ rm windowinglayout.xml
[oracle@EDRSR5P1 o.ide.11.1.1.0.22.49.48]$
```

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## SQL Developer 레이아웃 재설정

SQL Developer로 작업하는 동안 Connections Navigator가 사라지거나 Log window를 원래의 위치에 도킹할 수 없는 경우 다음 단계를 수행하여 문제를 해결합니다.

1. SQL Developer를 종료합니다.
2. 터미널 window를 열고 locate 명령을 사용하여 windowinglayout.xml의 위치를 찾습니다.
3. windowinglayout.xml이 있는 디렉토리로 이동하여 해당 파일을 삭제합니다.
4. SQL Developer를 재시작합니다.

## 요약

이 부록에서는 SQL Developer를 사용하여 다음 작업을 수행하는 방법을 배웠습니다.

- 데이터베이스 객체 탐색, 생성 및 편집
- SQL Worksheet에서 SQL 문 및 스크립트 실행
- 사용자 정의 보고서 작성 및 저장



Copyright © 2009, Oracle. All rights reserved.

### 요약

SQL Developer는 데이터베이스 개발 작업을 간편하게 수행할 수 있는 무료 그래픽 도구입니다. SQL Developer를 사용하여 데이터베이스 객체를 탐색, 생성 및 편집할 수 있습니다. 또한 SQL Worksheet를 사용하여 SQL 문 및 스크립트를 실행할 수 있습니다. SQL Developer를 통해 고유의 특수 보고서 집합을 생성하여 저장해 두었다가 반복해서 사용할 수 있습니다.



# D

## SQL\*Plus 사용

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 목표

이 부록을 마치면 다음을 수행할 수 있습니다.

- SQL\*Plus에 로그인
- SQL 명령 편집
- SQL\*Plus 명령을 사용하여 출력 형식 지정
- 스크립트 파일과 상호 작용

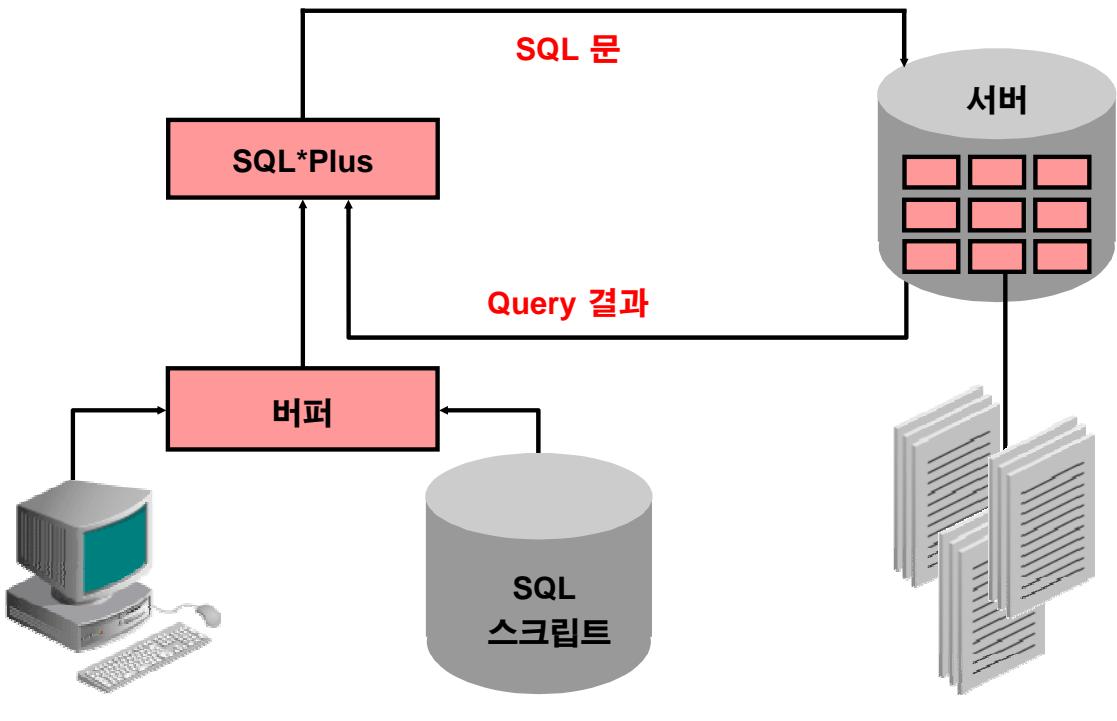
ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 목표

몇 번이고 사용할 수 있는 SELECT 문을 생성할 수 있습니다. 이 부록에서는 SQL\*Plus 명령을 사용하여 SQL 문을 실행하는 과정도 다릅니다. SQL\*Plus 명령을 사용하여 출력 형식을 지정하고, SQL 명령을 편집하고, SQL\*Plus로 스크립트를 저장하는 방법을 배웁니다.

# SQL과 SQL\*Plus의 상호 작용



**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

## SQL과 SQL\*Plus

SQL은 임의의 도구 또는 응용 프로그램에서 Oracle 서버와 통신하는 데 사용하는 명령어입니다. Oracle SQL은 많은 확장을 포함합니다. SQL 문을 입력하면 SQL 버퍼(SQL Buffer)라고 하는 메모리 부분에 저장되어 새 SQL 문을 입력할 때까지 그대로 남아 있습니다. SQL\*Plus는 SQL 문을 인식하여 실행을 위해 Oracle9i Server로 제출하는 오라클 도구로, 자체 명령어를 포함합니다.

## SQL의 특성

- 프로그래밍의 경험이 별로 없거나 아예 없는 유저를 포함하여 다양한 유저가 사용할 수 있습니다.
- 비절차적 언어입니다.
- 시스템 생성 및 유지 관리에 필요한 시간을 줄여 줍니다.
- 영어와 유사한 언어입니다.

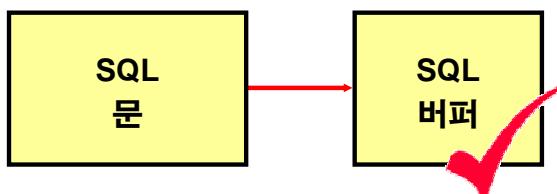
## SQL\*Plus의 특성

- 명령문의 임시 항목을 받아들입니다.
- 파일을 사용하여 SQL을 입력할 수 있습니다.
- SQL 문 수정을 위한 행 편집기를 제공합니다.
- 환경 설정을 제어합니다.
- Query 결과를 기본 보고서 형식으로 만듭니다.
- 로컬 및 원격 데이터베이스에 액세스합니다.

# SQL 문과 SQL\*Plus 명령 비교

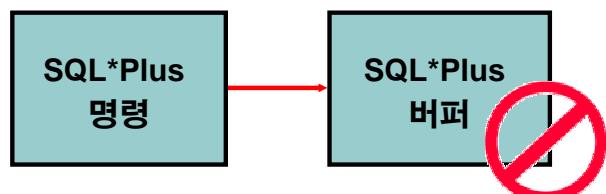
## SQL

- 언어
- ANSI 표준
- 키워드를 약어로 표기할 수 없음
- 명령문으로 데이터베이스의 데이터 및 테이블 정의를 조작함



## SQL\*Plus

- 환경
- Oracle 고유
- 키워드를 약어로 표시할 수 있음
- 명령으로 데이터베이스의 값을 조작할 수 없음



ORACLE

Copyright © 2009, Oracle. All rights reserved.

## SQL과 SQL\*Plus(계속)

다음 표에서는 SQL과 SQL\*Plus를 비교합니다.

SQL	SQL*Plus
데이터에 액세스하기 위해 Oracle 서버와 통신하기 위한 언어	SQL 문을 인식하고 서버로 보냄
ANSI(American National Standards Institute) 표준 SQL을 기반으로 함	SQL 문을 실행하기 위한 오라클 고유의 인터페이스
데이터베이스의 데이터 및 테이블을 조작함	데이터베이스의 값을 조작할 수 없음
SQL 버퍼에 하나 이상의 행이 입력됨	한 번에 한 행씩 입력되고 SQL 버퍼에 저장되지 않음
연속 문자가 없음	명령이 한 줄보다 긴 경우 연속 문자로 대시(-)를 사용
약어를 사용할 수 없음	약어를 사용할 수 있음
종료 문자를 사용하여 명령을 즉시 실행	종료 문자를 사용할 필요 없이 명령을 즉시 실행
함수를 사용하여 일부 형식 지정 수행	명령을 사용하여 데이터의 형식 지정

# SQL\*Plus 개요

- SQL\*Plus에 로그인합니다.
- 테이블 구조를 설명합니다.
- SQL 문을 편집합니다.
- SQL\*Plus에서 SQL을 실행합니다.
- SQL 문을 파일에 저장하고 첨부합니다.
- 저장된 파일을 실행합니다.
- 파일에서 버퍼로 명령을 로드하여 편집합니다.

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

## SQL\*Plus

SQL\*Plus는 다음 작업을 수행할 수 있는 환경입니다.

- 데이터베이스에서 데이터를 검색, 수정, 추가 및 제거하는 SQL 문을 실행합니다.
- query 결과의 형식을 지정하고 계산을 수행하고, 저장하고, 보고서 형식으로 인쇄합니다.
- 앞으로 반복 사용할 수 있도록 SQL 문을 저장하는 스크립트 파일을 생성합니다.

SQL\*Plus 명령은 다음의 주요 범주로 나눌 수 있습니다.

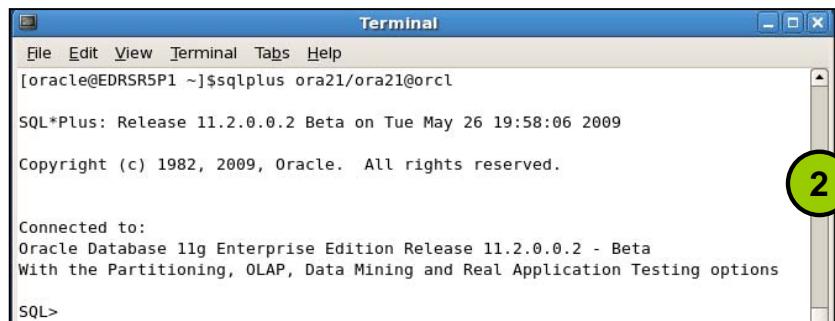
범주	목적
환경	세션에 대한 SQL 문의 일반 동작에 영향을 줍니다.
형식	Query 결과의 형식을 지정합니다.
파일 조작	스크립트 파일을 저장, 로드, 실행합니다.
실행	SQL 문을 SQL 버퍼에서 Oracle 서버로 보냅니다.
편집	버퍼의 SQL 문을 수정합니다.
상호 작용	변수를 생성하여 SQL 문에 전달하고, 변수 값을 인쇄하고, 화면에 메시지를 출력합니다.
기타	데이터베이스에 연결하고, SQL*Plus 환경을 조작하며, 열 정의를 표시합니다.

## SQL\*Plus에 로그인!



```
[oracle@EDRSR5P1 ~]$sqlplus
SQL*Plus: Release 11.2.0.0.2 Beta on Tue May 26 19:59:06 2009
Copyright (c) 1982, 2009, Oracle. All rights reserved.
Enter user-name: ora21@orcl
Enter password: 1
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.0.2 - Beta
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL>
```

**sqlplus [username[ /password[@database]]]**



```
[oracle@EDRSR5P1 ~]$sqlplus ora21/ora21@orcl
SQL*Plus: Release 11.2.0.0.2 Beta on Tue May 26 19:58:06 2009
Copyright (c) 1982, 2009, Oracle. All rights reserved.
2
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.0.2 - Beta
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL>
```

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### SQL\*Plus에 로그인

SQL\*Plus를 호출하는 방법은 오라클 데이터베이스를 실행 중인 운영 체제의 유형에 따라 다릅니다.

Linux 환경에서 로그인 하려면 다음 단계를 수행하십시오.

1. Linux 바탕 화면을 마우스 오른쪽 버튼으로 누르고 terminal을 선택합니다.
2. 슬라이드에 표시된 sqlplus 명령을 입력합니다.
3. Username, 암호 및 데이터베이스 이름을 입력합니다.

이 구문에서 다음이 적용됩니다.

*username* 데이터베이스 username입니다.

*password* 데이터베이스 암호입니다. 여기에 암호를 입력하면 암호가 보입니다.

*@database* 데이터베이스 연결 문자열입니다.

**참고:** 암호의 무결성을 보장하려면 운영 체제 프롬프트에서 암호를 입력하지 마십시오.

대신 Username만 입력하십시오. 암호는 암호 프롬프트에서 입력하십시오.

# 테이블 구조 표시

SQL\*Plus DESCRIBE 명령을 사용하여 테이블의 구조를 표시합니다.

```
DESC[RIBE] tablename
```

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 테이블 구조 표시

SQL\*Plus에서 DESCRIBE 명령을 사용하여 테이블의 구조를 표시할 수 있습니다. 이 명령의 결과로 열 이름, 데이터 유형 및 열에 반드시 데이터가 포함되어야 하는지 여부가 표시됩니다. 이 구문에서 다음이 적용됩니다.

**tablename** 유저가 액세스할 수 있는 기존의 테이블, 뷰 또는 동의어의 이름입니다.

DEPARTMENTS 테이블을 기술하려면 다음 명령을 사용합니다.

```
SQL> DESCRIBE DEPARTMENTS
Name          Null?    Type
-----
DEPARTMENT_ID      NOT NULL NUMBER(4)
DEPARTMENT_NAME    NOT NULL VARCHAR2(30)
MANAGER_ID         NUMBER(6)
LOCATION_ID        NUMBER(4)
```

# 테이블 구조 표시

```
DESCRIBE departments
```

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER( 4 )
DEPARTMENT_NAME	NOT NULL	VARCHAR2( 30 )
MANAGER_ID		NUMBER( 6 )
LOCATION_ID		NUMBER( 4 )

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 테이블 구조 표시(계속)

슬라이드의 예제는 DEPARTMENTS 테이블의 구조에 대한 정보를 표시합니다. 결과에서 다음이 적용됩니다.

Null?: 열에 데이터가 포함되어야 하는지 여부를 지정합니다. NOT NULL은 열에 데이터가 포함되어야 함을 나타냅니다.

Type: 열의 데이터 유형을 표시합니다.

# SQL\*Plus 편집 명령

- **A[PPEND] *text***
- **C[HANGE] / *old* / *new***
- **C[HANGE] / *text* /**
- **CL[EAR] BUFF[ER]**
- **DEL**
- **DEL *n***
- **DEL *m* *n***

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

## SQL\*Plus 편집 명령

SQL\*Plus 명령은 한 번에 한 줄씩 입력되고 SQL 버퍼에 저장되지 않습니다.

명령	설명
A[PPEND] <i>text</i>	현재 행의 끝에 텍스트를 추가합니다.
C[HANGE] / <i>old</i> / <i>new</i>	현재 행에서 <i>old</i> 텍스트를 <i>new</i> 로 변경합니다.
C[HANGE] / <i>text</i> /	현재 행에서 <i>text</i> 를 삭제합니다.
CL[EAR] BUFF[ER]	SQL 버퍼에서 모든 행을 삭제합니다.
DEL	현재 행을 삭제합니다.
DEL <i>n</i>	<i>n</i> 행을 삭제합니다.
DEL <i>m</i> <i>n</i>	<i>m</i> 행부터 <i>n</i> 행까지 삭제합니다.

### 지침

- 명령을 완료하기 전에 Enter를 누르면 SQL\*Plus에서 행 번호를 표시합니다.
- 종료 문자(세미콜론이나 슬래시) 중 하나를 입력하거나 [Enter]를 두 번 눌러 SQL 버퍼를 종료합니다. 그러면 SQL 프롬프트가 나타납니다.

# SQL\*Plus 편집 명령

- **I[NPUT]**
- **I[NPUT] *text***
- **L[IST]**
- **L[IST] *n***
- **L[IST] *m n***
- **R[UN]**
- ***n***
- ***n text***
- **0 *text***

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## SQL\*Plus 편집 명령(계속)

명령	설명
I[NPUT]	임의 개수의 행을 삽입합니다.
I[NPUT] <i>text</i>	<i>text</i> 로 구성된 행을 삽입합니다.
L[IST]	SQL 버퍼에 있는 모든 행을 나열합니다.
L[IST] <i>n</i>	한 행( <i>n</i> 으로 지정된 행)을 나열합니다.
L[IST] <i>m n</i>	일정 범위( <i>m-n</i> )의 행을 나열합니다.
R[UN]	버퍼에 있는 현재 SQL 문을 표시하고 실행합니다.
<i>N</i>	<i>n</i> 행을 현재 행으로 지정합니다.
<i>n text</i>	<i>n</i> 행을 <i>text</i> 로 대체합니다.
0 <i>text</i>	1행 앞에 행을 삽입합니다.

**참고:** 각 SQL 프롬프트에서 하나의 SQL\*Plus 명령만 입력할 수 있습니다. SQL\*Plus 명령은 버퍼에 저장되지 않습니다. SQL\*Plus 명령이 다음 행으로 이어지는 경우 첫 행의 끝에 하이픈(-)을 추가합니다.

## LIST, n 및 APPEND 사용

```
LIST
1  SELECT last_name
2* FROM employees
```

```
1
1* SELECT last_name
```

```
A , job_id
1* SELECT last_name, job_id
```

```
LIST
1  SELECT last_name, job_id
2* FROM employees
```

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### LIST, n 및 APPEND 사용

- L[IST] 명령을 사용하여 SQL 버퍼의 내용을 표시합니다. 버퍼에서 2행 옆의 별표(\*)는 해당 행이 현재 행임을 나타냅니다. 편집한 내용은 현재 행에 적용됩니다.
- 편집하려는 행 번호(n)를 입력하여 현재 행의 번호를 바꿉니다. 새로운 현재 행이 표시됩니다.
- A[PPEND] 명령을 사용하여 현재 행에 텍스트를 추가합니다. 새로 편집한 행이 표시됩니다. LIST 명령을 사용하여 버퍼의 새 내용을 확인합니다.

**참고:** LIST 및 APPEND를 비롯한 대부분의 SQL\*Plus 명령은 첫번째 문자만 사용하여 약어로 표기할 수 있습니다. LIST는 L, APPEND는 A라는 약어로 표기할 수 있습니다.

## CHANGE 명령 사용

```
LIST
```

```
1* SELECT * from employees
```

```
c/employees/departments
```

```
1* SELECT * from departments
```

```
LIST
```

```
1* SELECT * from departments
```

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### CHANGE 명령 사용

- L[IST]를 사용하여 버퍼의 내용을 표시합니다.
- C[HANGE] 명령을 사용하여 SQL 버퍼에서 현재 행의 내용을 변경합니다. 이 경우 employees 테이블을 departments 테이블로 대체합니다. 새로운 현재 행이 표시됩니다.
- L[IST] 명령을 사용하여 버퍼의 새 내용을 확인합니다.

# SQL\*Plus 파일 명령

- **SAVE *filename***
- **GET *filename***
- **START *filename***
- **@ *filename***
- **EDIT *filename***
- **SPOOL *filename***
- **EXIT**

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## SQL\*Plus 파일 명령

SQL 문은 Oracle 서버와 통신합니다. SQL\*Plus 명령은 환경을 제어하고 query 결과의 서식을 지정하고 파일을 관리합니다. 다음 표에 설명된 명령을 사용할 수 있습니다.

## SQL\*Plus 파일 명령 (계속)

명령	설명
SAV[E] <i>filename</i> [.ext] [REP[LACE]APP[END]]	SQL 버퍼의 현재 내용을 파일에 저장합니다. 기존 파일에 추가하려면 APPEND를 사용하고 기존 파일을 덮어 쓰려면 REPLACE를 사용합니다. 기본 확장자는 .sql입니다.
GET <i>filename</i> [.ext]	이전에 저장한 파일의 내용을 SQL 버퍼에 씁니다. 파일 이름의 기본 확장자는 .sql입니다.
STA[RT] <i>filename</i> [.ext]	이전에 저장한 명령 파일을 실행합니다.
@ <i>filename</i>	이전에 저장한 명령 파일을 실행합니다(START와 동일).
ED[IT]	편집기를 호출하여 버퍼 내용을 afiedt.buf라는 파일에 저장합니다.
ED[IT] [ <i>filename</i> [.ext]]	편집기를 호출하여 저장된 파일의 내용을 편집합니다.
SPO[OL] [ <i>filename</i> [.ext]]   OFF OUT]	Query 결과를 파일에 저장합니다. OFF는 스펄 파일을 닫습니다. OUT은 스펄 파일을 닫고 파일 결과를 프린터로 보냅니다.
EXIT	SQL*Plus를 종료합니다.

## SAVE 및 START 명령 사용

**LIST**

```
1  SELECT last_name, manager_id, department_id
2* FROM employees
```

**SAVE my\_query**

Created file my\_query

**START my\_query**

LAST_NAME	MANAGER_ID	DEPARTMENT_ID
King		90
Kochhar	100	90
...		
107 rows selected.		

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

## SAVE, START 및 EDIT 명령 사용

### SAVE

SAVE 명령을 사용하여 버퍼의 현재 내용을 파일에 저장합니다. 이와 같은 방법으로 자주 사용되는 스크립트를 나중에 사용할 수 있도록 저장할 수 있습니다.

### START

START 명령을 사용하여 SQL\*Plus에서 스크립트를 실행합니다. 또는 기호 @을 사용하여 스크립트를 실행할 수도 있습니다.

`@my_query`

## SERVEROUTPUT 명령

- SET SERVEROUT[PUT] 명령을 사용하여 내장 프로시저 또는 PL/SQL 블록의 출력을 SQL\*Plus에 표시할지 여부를 제어할 수 있습니다.
- DBMS\_OUTPUT 행 길이 제한이 255바이트에서 32767바이트로 늘어났습니다.
- 이제 기본 크기에 제한이 없습니다.
- SERVEROUTPUT를 설정하는 경우 리소스가 미리 할당되지 않습니다.
- Physical memory를 절약하려는 경우가 아니라면 UNLIMITED를 사용하십시오. 그래도 성능 저하가 발생하지 않습니다.

```
SET SERVEROUT[PUT] {ON | OFF} [SIZE {n | UNLIMTED}] [  
FOR[MAT] {WRA[PPED] | WOR[D_WRAPPED] | TRU[NATED]}]
```

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### SERVEROUTPUT 명령

대부분의 PL/SQL 프로그램에서는 SQL 문을 통해 입력 및 출력을 수행하여 데이터를 데이터베이스 테이블에 저장하거나 해당 테이블을 query합니다. 다른 모든 PL/SQL 입/출력은 다른 프로그램과 상호 작용하는 API를 통해 수행됩니다. 예를 들어, DBMS\_OUTPUT 패키지에는 PUT\_LINE과 같은 프로시저가 포함되어 있습니다. PL/SQL 외부에서 결과를 보려면 DBMS\_OUTPUT에 전달된 데이터를 읽고 표시할 수 있는 SQL\*Plus와 같은 다른 프로그램이 필요합니다.

SQL\*Plus에서 DBMS\_OUTPUT 데이터를 표시하려면 먼저 다음과 같이 SQL\*Plus 명령 SET SERVEROUTPUT ON을 실행해야 합니다.

```
SET SERVEROUTPUT ON
```

#### 참고

- SIZE는 오라클 데이터베이스 서버 내에서 버퍼될 수 있는 출력 크기를 바이트 수로 나타낸 값입니다. 기본값은 UNLIMITED입니다. N은 2000보다 작거나 1,000,000보다 클 수 없습니다.
- SERVEROUTPUT에 대한 자세한 내용은 *Oracle Database PL/SQL User's Guide and Reference 11g*를 참조하십시오.

## SQL\*Plus SPOOL 명령 사용

```
SPO[OL] [file_name[.ext]] [CRE[A TE] | REP[L ACE] |
APP[END]] | OFF | OUT]
```

옵션	설명
file_name[.ext]	출력을 지정된 파일 이름으로 스플립니다.
CRE[A TE]	지정한 이름으로 새 파일을 생성합니다.
REP[L ACE]	기존 파일의 내용을 바꿉니다. 파일이 존재하지 않을 경우 REPLACE를 사용하면 파일이 생성됩니다.
APP[END]	지정한 파일의 끝에 버퍼의 내용을 추가합니다.
OFF	스풀을 정지합니다.
OUT	스풀 작업을 정지하고 파일을 컴퓨터의 표준(기본) 프린터로 보냅니다.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

### SQL\*Plus SPOOL 명령 사용

SPOOL 명령은 query 결과를 파일에 저장하거나 선택적으로 파일을 프린터로 보냅니다. SPOOL 명령이 향상되었습니다. 이전에는 SPOOL 명령을 사용하여 파일을 생성하거나 바꿀 수만 있었지만 이제는 기존 파일을 바꾸는 것뿐 아니라 기존 파일에 내용을 추가할 수도 있습니다. 기본값은 REPLACE입니다.

스크립트의 명령에 의해 생성된 출력을 화면에 표시하지 않고 스플립려면 SET TERMOUT OFF를 사용합니다. SET TERMOUT OFF는 대화식으로 실행하는 명령의 출력에는 영향을 주지 않습니다.

공백이 포함된 파일 이름은 따옴표로 묶어야 합니다. SPOOL APPEND 명령을 사용하여 유효한 HTML 명령을 생성하려면 PROMPT 또는 유사한 명령을 사용하여 HTML 페이지 header와 footer를 생성해야 합니다. SPOOL APPEND 명령은 HTML 태그 구문을 분석하지 않습니다. CREATE, APPEND 및 SAVE 파라미터를 비활성화하려면 SET SQLPLUSCOMPAT[IBILITY]를 9.2 이하로 설정합니다.

## AUTOTRACE 명령 사용

- **SELECT, INSERT, UPDATE, DELETE 같은 SQL DML 문을 성공적으로 실행한 후에 보고서를 표시합니다.**
- **이제 보고서에 실행 통계 및 query 실행 경로가 포함될 수 있습니다.**

```
SET AUTOT[RACE] {ON | OFF | TRACE[ONLY]} [EXP[LAIN]]  
[STATISTICS]
```

```
SET AUTOTRACE ON  
-- The AUTOTRACE report includes both the optimizer  
-- execution path and the SQL statement execution  
-- statistics
```

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

### AUTOTRACE 명령 사용

EXPLAIN은 EXPLAIN PLAN을 수행하여 query 실행 경로를 보여줍니다. STATISTICS는 SQL 문 통계를 표시합니다. AUTOTRACE 보고서의 형식은 연결되어 있는 서버의 버전과 서버의 구성에 따라 달라질 수 있습니다. DBMS\_XPLAN 패키지를 사용하면 EXPLAIN PLAN 명령의 출력을 미리 정의된 여러 형식으로 간단히 표시할 수 있습니다.

#### 참고

- 패키지 및 서브 프로그램에 대한 자세한 내용은 *Oracle Database PL/SQL Packages and Types Reference 11g*를 참조하십시오.
- EXPLAIN PLAN에 대한 자세한 내용은 *Oracle Database SQL Reference 11g*를 참조하십시오.
- 실행 계획 및 통계에 대한 자세한 내용은 *Oracle Database Performance Tuning Guide 11g*를 참조하십시오.

## 요약

이 부록에서는 다음 작업을 수행하기 위한 환경으로 SQL\*Plus를 사용하는 방법을 배웠습니다.

- SQL 문 실행
- SQL 문 편집
- 출력 형식 지정
- 스크립트 파일과 상호 작용



Copyright © 2009, Oracle. All rights reserved.

### 요약

SQL\*Plus는 SQL 명령을 데이터베이스 서버로 보내고 SQL 명령을 편집 및 저장하는 데 사용할 수 있는 실행 환경입니다. SQL 프롬프트 또는 스크립트 파일에서 명령을 실행할 수 있습니다.



# JDeveloper 사용

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 목표

이 부록을 마치면 다음을 수행할 수 있습니다.

- Oracle JDeveloper의 주요 기능 나열
- JDeveloper에서 데이터베이스 연결 생성
- JDeveloper에서 데이터베이스 객체 관리
- JDeveloper를 사용하여 SQL 명령 실행
- PL/SQL 프로그램 단위 생성 및 실행

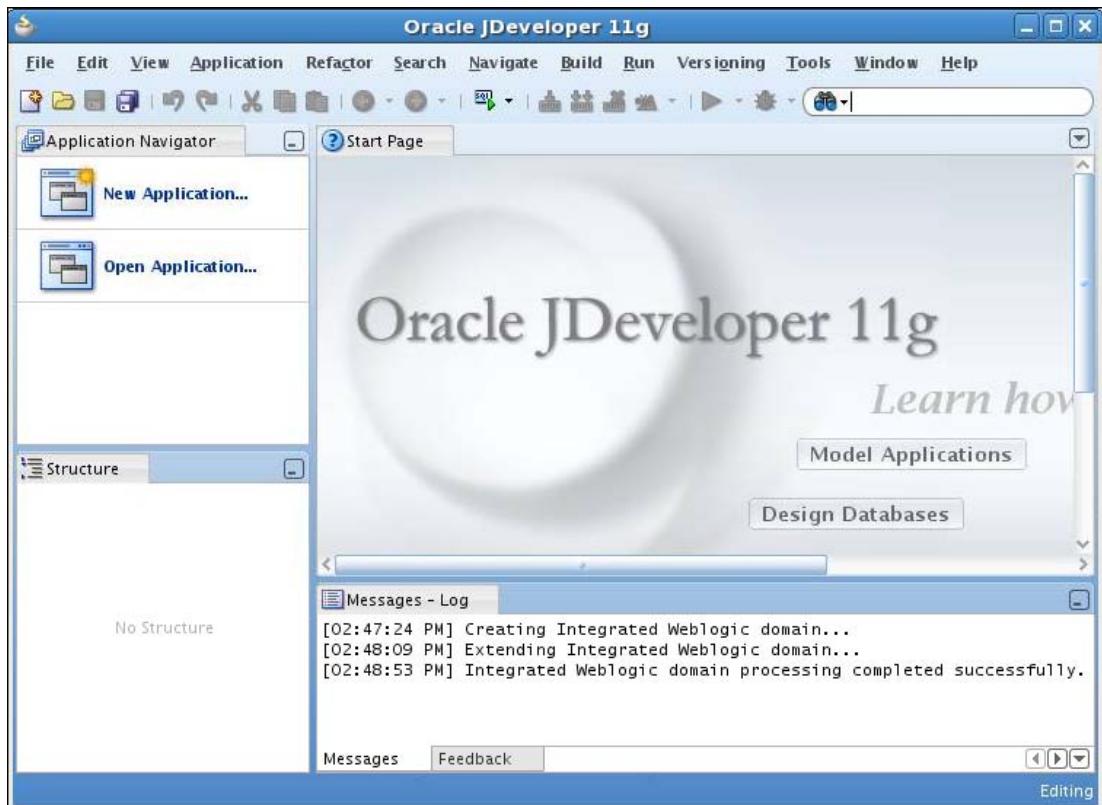
ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 목표

이 부록에서는 JDeveloper를 소개합니다. 먼저, 데이터베이스 개발 작업에 JDeveloper를 사용하는 방법에 대해 알아봅니다.

# Oracle JDeveloper



ORACLE

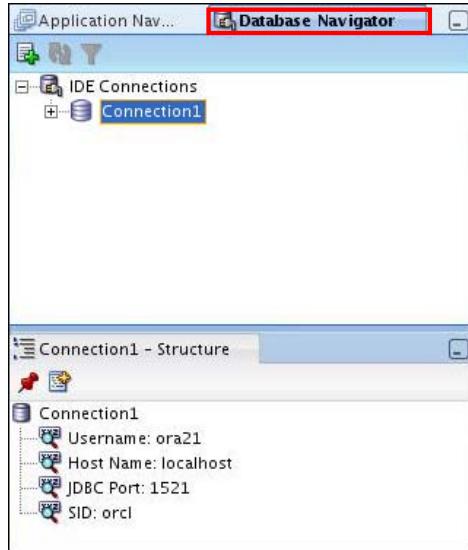
Copyright © 2009, Oracle. All rights reserved.

## Oracle JDeveloper

Oracle JDeveloper는 Java 응용 프로그램과 웹 서비스를 개발하여 배치할 수 있는 IDE(통합 개발 환경)입니다. 모델링에서 배치까지 전단계의 SDLC(소프트웨어 개발 주기)를 지원합니다. 이 도구에는 응용 프로그램 개발 시 Java, XML 및 SQL에 대한 최신 산업 표준을 사용하는 기능이 있습니다.

Oracle JDeveloper 11g는 시각적/선언적 개발을 지원하는 기능으로 J2EE 개발에 대한 새로운 접근법을 시도합니다. 이 혁신적 접근은 J2EE 개발을 간단하고 효율적으로 만듭니다.

# Database Navigator



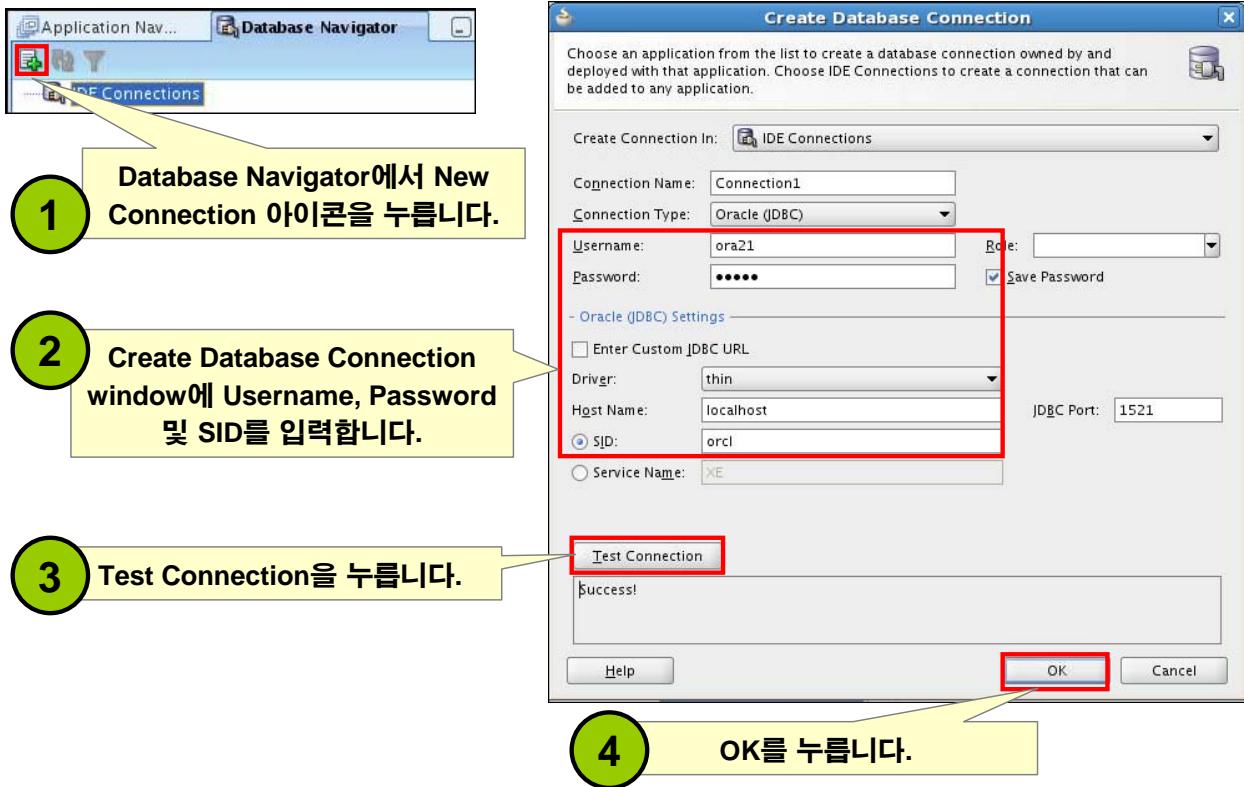
ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## Database Navigator

Oracle JDeveloper를 사용하면 데이터베이스에 연결하는 데 필요한 정보를 "connection"이라는 객체에 저장할 수 있습니다. 연결은 IDE 설정의 일부로 저장되며 유저 그룹 간에 공유하기 쉽도록 엑스포트하고 임포트할 수 있습니다. 또한 데이터베이스를 찾고 응용 프로그램을 구축하는 것에서부터 배치에 이르기까지 다양한 용도로 사용됩니다.

## 연결 생성



ORACLE

Copyright © 2009, Oracle. All rights reserved.

### 연결 생성

연결은 특정 데이터베이스에 해당 데이터베이스의 특정 유저로 연결하는 데 필요한 정보를 지정하는 객체입니다. 여러 데이터베이스 및 여러 스키마에 대해 연결을 생성하고 테스트할 수 있습니다.

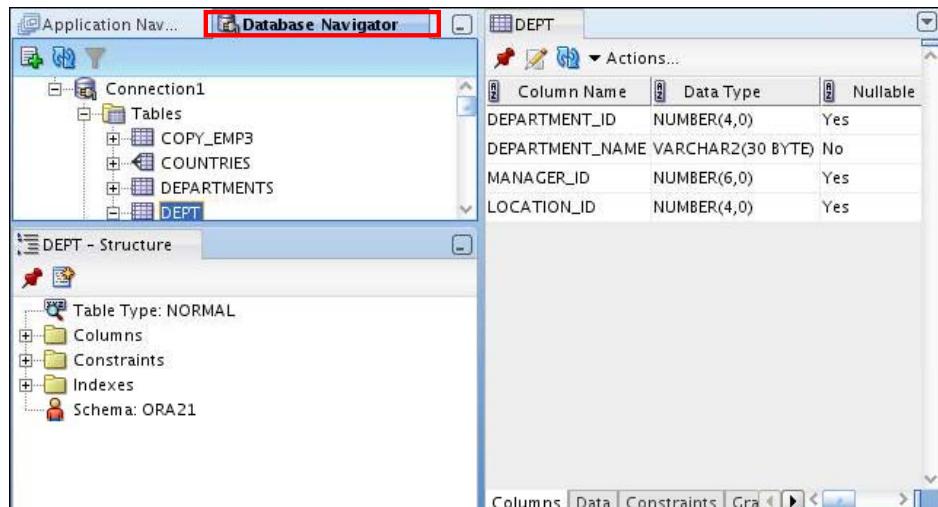
데이터베이스 연결을 생성하려면 다음 단계를 수행하십시오.

1. Database Navigator에서 New Connection 아이콘을 누릅니다.
2. Create Database Connection window에 연결 이름을 입력합니다. 연결하려는 스키마의 Username 및 암호를 입력합니다. 연결하려는 데이터베이스의 SID를 입력합니다.
3. Test를 눌러 연결이 올바르게 설정되었는지 확인합니다.
4. OK를 누릅니다.

## 데이터베이스 객체 탐색

Database Navigator를 사용하여 다음을 수행할 수 있습니다.

- 데이터베이스 스키마에서 여러 객체 탐색
- 여러 객체의 정의를 한 번에 검토



ORACLE

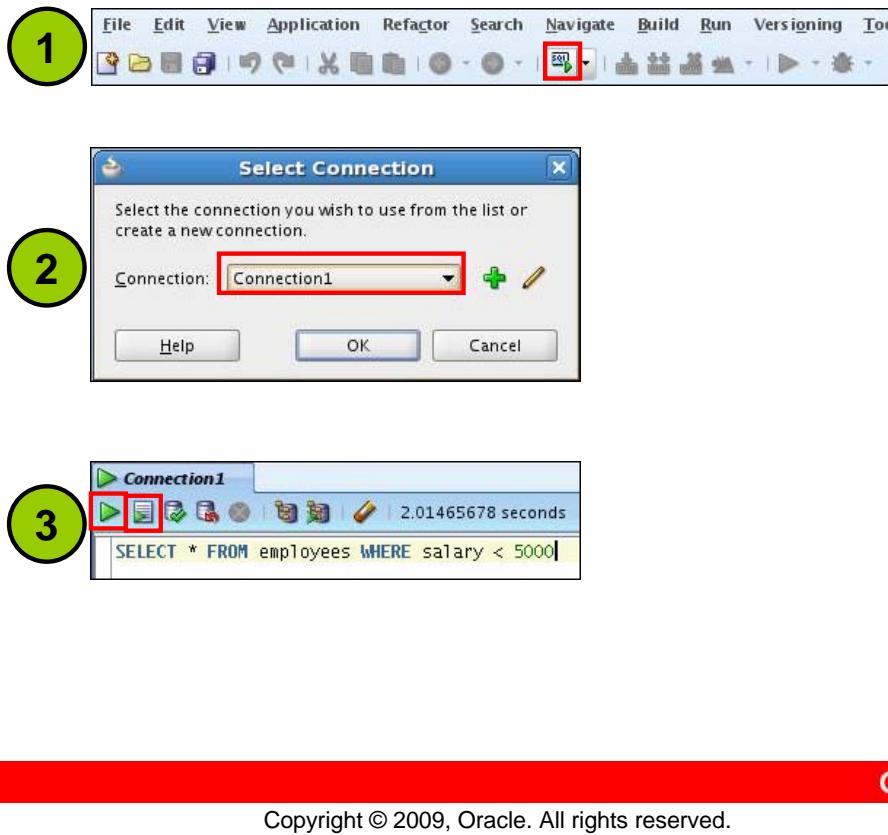
Copyright © 2009, Oracle. All rights reserved.

### 데이터베이스 객체 탐색

데이터베이스 연결을 생성한 후 Database Navigator를 사용하여 데이터베이스 스키마에서 테이블, 뷰, 인덱스, 패키지, 프로시저, 트리거, 유형 등의 여러 객체를 탐색할 수 있습니다.

데이터 딕셔너리에서 추출된 객체 정의가 여러 정보 탭에 분할되어 있는 것을 볼 수 있습니다. 예를 들어, Navigator에서 테이블을 선택하면 열, 제약 조건, 권한 부여, 통계, 트리거 등에 대한 세부 정보가 Database Navigator에 표시됩니다.

# SQL 문 실행



ORACLE

Copyright © 2009, Oracle. All rights reserved.

## SQL 문 실행

SQL 문을 실행하려면 다음 단계를 수행하십시오.

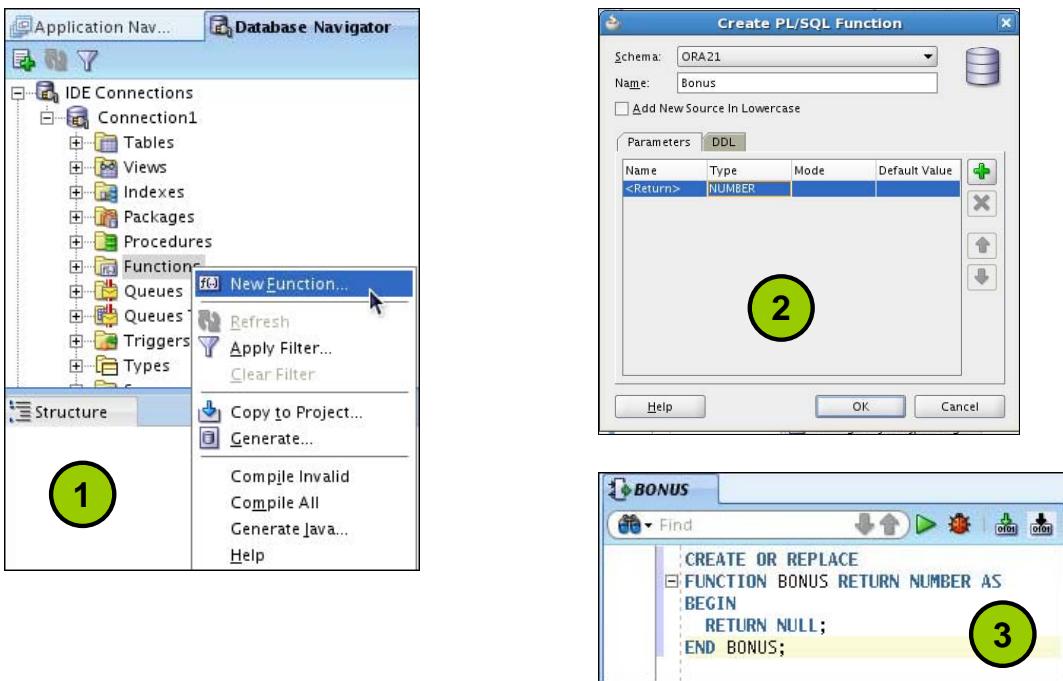
1. Open SQL Worksheet 아이콘을 누릅니다.
2. 연결을 선택합니다.
3. 다음을 눌러 SQL 명령을 실행합니다.
  1. Execute statement 버튼 또는 F9 키를 누릅니다. 출력 결과는 다음과 같습니다.

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME
1	100	Steven	King
2	101	Neena	Kochhar

2. Run Script 버튼 또는 F5 키를 누릅니다. 출력 결과는 다음과 같습니다.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
100	Steven	King

## 프로그램 단위 생성



함수의 기본 구조

ORACLE

Copyright © 2009, Oracle. All rights reserved.

### 프로그램 단위 생성

PL/SQL 프로그램 단위를 생성하려면 다음 단계를 수행하십시오.

1. View > Database Navigator를 선택합니다. 데이터베이스 연결을 선택하고 확장합니다. 객체 유형에 해당하는 폴더(Procedures, Packages, Functions)를 마우스 오른쪽 버튼으로 누릅니다. "New [Procedures | Packages | Functions]"를 선택합니다.
2. 함수, 패키지 또는 프로시저에 대한 유효한 이름을 입력하고 OK를 누릅니다.
3. 기본 구조 정의가 생성되어 Code Editor에서 열립니다. 그런 다음 요구에 맞게 서브 프로그램을 편집할 수 있습니다.

## 컴파일

Messages - Log

Compiling...

Context: MakeSelectedCommand selection=Element containing fi  
/home/oracle/Middleware/jdk160\_11/jre/bin/java -jar /  
[5:14:32 PM] Successful compilation: 0 errors, 0 warnings.

Messages Feedback <>

### 오류가 있는 컴파일

Compiler - Log

Project: /home/oracle/jdeveloper/mywork/Application1/Project1/Project1

/home/oracle/jdeveloper/mywork/Application1/Project1/src/project1

- Error(7,13): duplicate definition of variable a in constructor Hello()
- Error(7,15): ; expected
- Error(8,28): variable a might not have been initialized

Messages Feedback Compiler <>

### 오류가 없는 컴파일

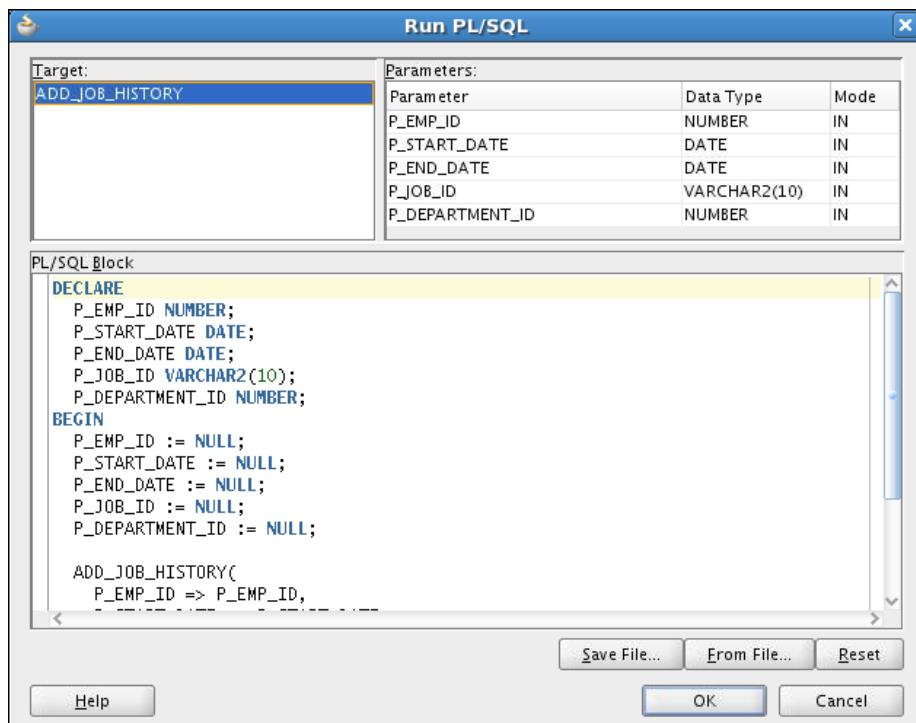
ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 컴파일

기본 구조 정의를 편집한 후 프로그램 단위를 컴파일해야 합니다. Connection Navigator에서 컴파일해야 하는 PL/SQL 객체를 마우스 오른쪽 버튼으로 누르고 Compile을 선택합니다. 또는 [Ctrl] + [Shift] + [F9]를 눌러 컴파일할 수도 있습니다.

## 프로그램 단위 실행



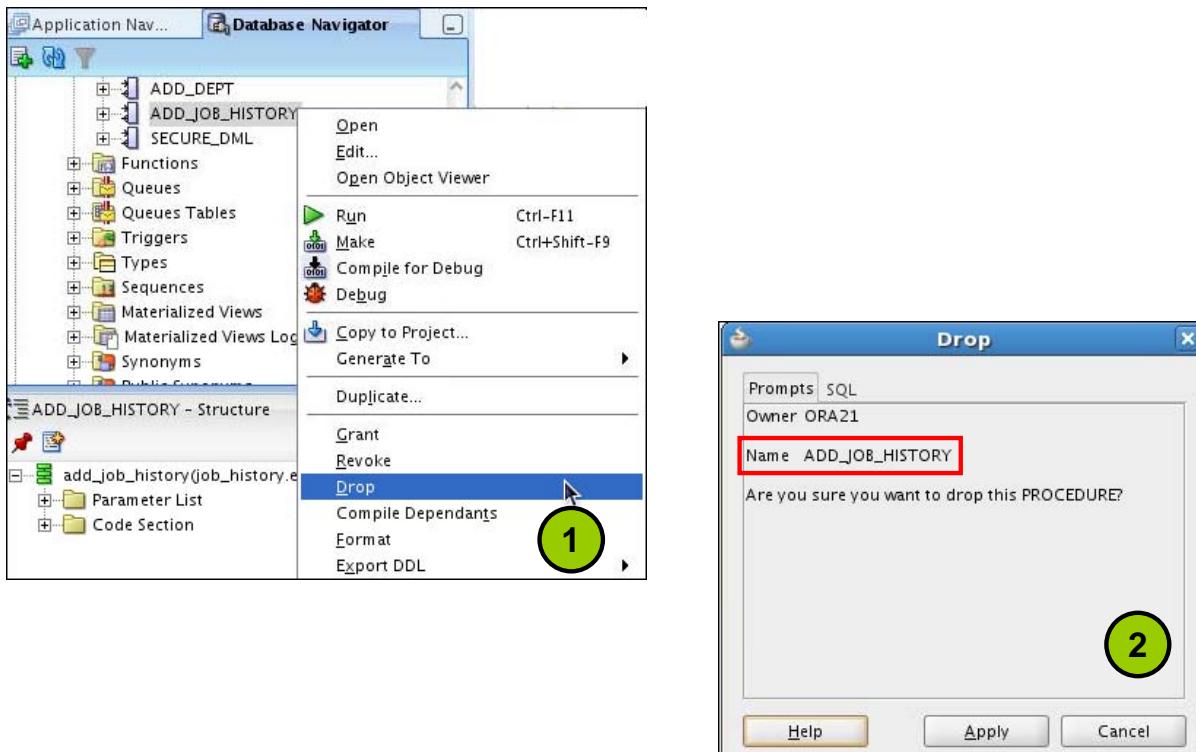
ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 프로그램 단위 실행

프로그램 단위를 실행하려면 객체를 마우스 오른쪽 버튼으로 누르고 Run을 누릅니다. Run PL/SQL 대화상자가 나타납니다. NULL 값을 프로그램 단위로 전달하기에 적절한 값으로 변경해야 할 수도 있습니다. 해당 값을 변경한 후 OK를 누릅니다. Message-Log window에 출력 결과가 표시됩니다.

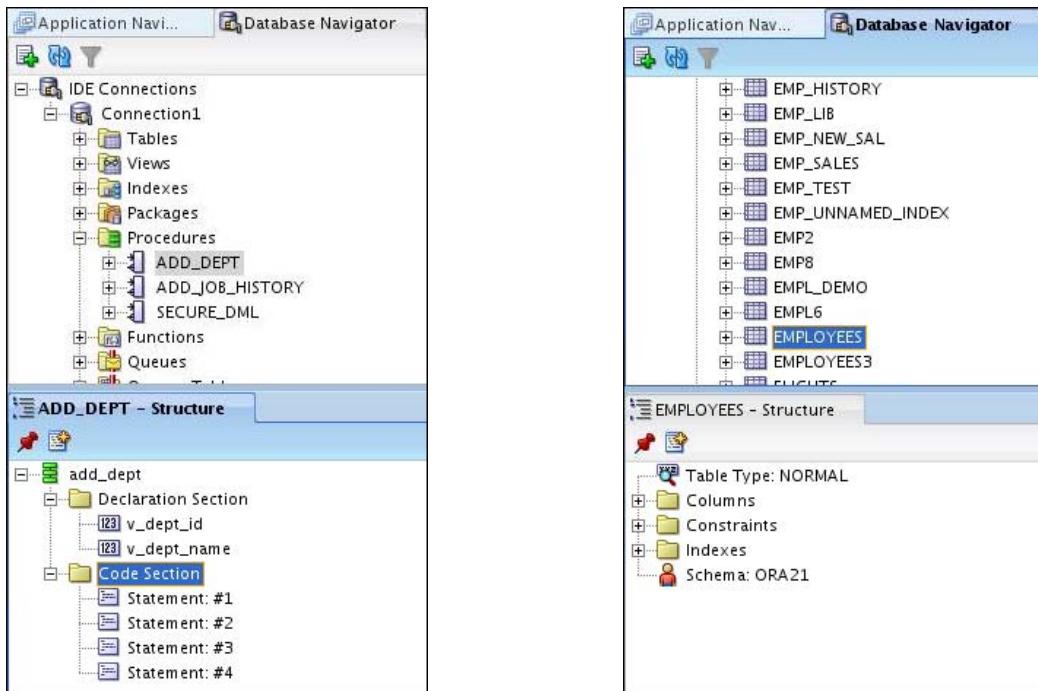
## 프로그램 단위 삭제



### 프로그램 단위 삭제

프로그램 단위를 삭제하려면 객체를 마우스 오른쪽 버튼으로 누르고 Drop을 선택합니다. Drop Confirmation 대화상자가 나타나면 Apply를 누릅니다. 객체가 데이터베이스에서 삭제됩니다.

# Structure window



**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

## Structure window

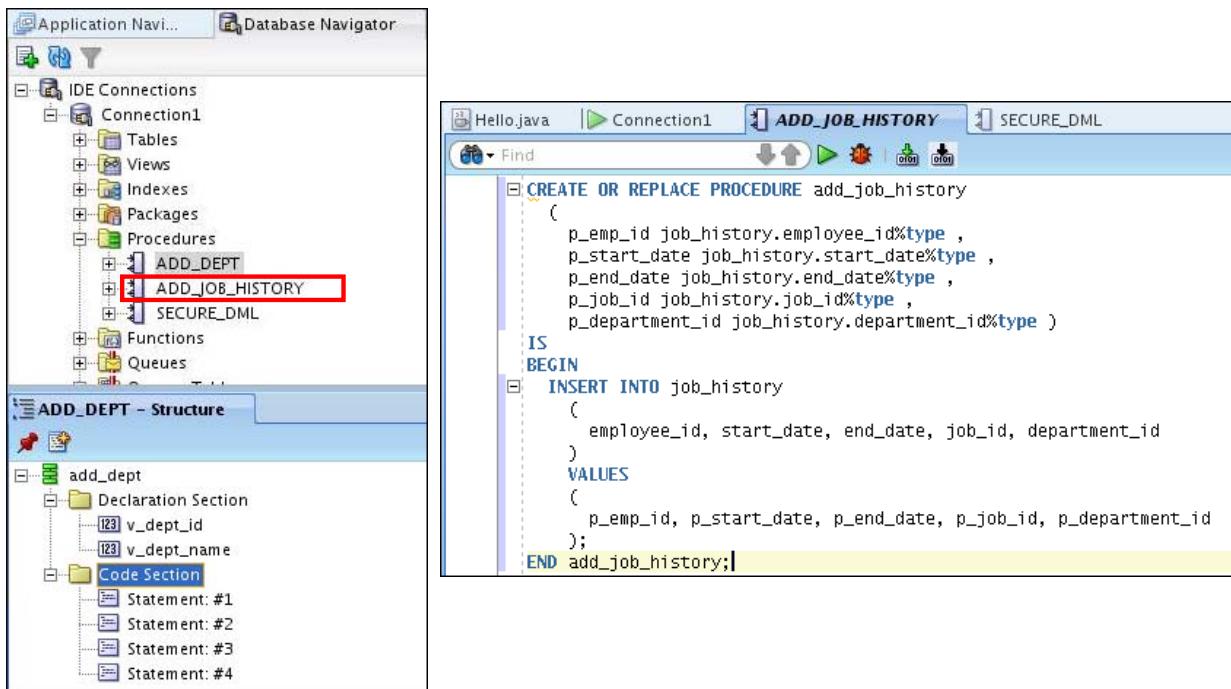
Structure window는 구조를 제공하는 데 관여하는 Navigator, Editor, Viewer, Property Inspector와 같은 window 중 활성 window에 현재 선택된 문서 데이터의 구조적 뷰를 제공합니다.

View > Structure window를 눌러 Structure window를 봅니다.

Structure window에서 다양한 방식으로 문서 데이터를 볼 수 있습니다. 표시할 수 있는 구조는 문서 유형에 따라 결정됩니다. Java 파일의 경우 코드 구조, UI 구조 또는 UI 모델 데이터를 볼 수 있습니다. XML 파일의 경우 XML 구조, 디자인 구조 또는 UI 모델 데이터를 볼 수 있습니다.

Structure window는 현재의 활성 Editor와 연관되어 특정 뷰에 window 내용을 고정하지 않는 한 항상 동적으로 활성 window의 현재 선택 사항을 추적합니다. 현재 선택 사항이 Navigator의 노드일 때를 기본 편집기로 간주합니다. 현재 선택 사항의 구조에 대한 뷰를 변경하려면 다른 구조 탭을 누릅니다.

# Editor window



**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

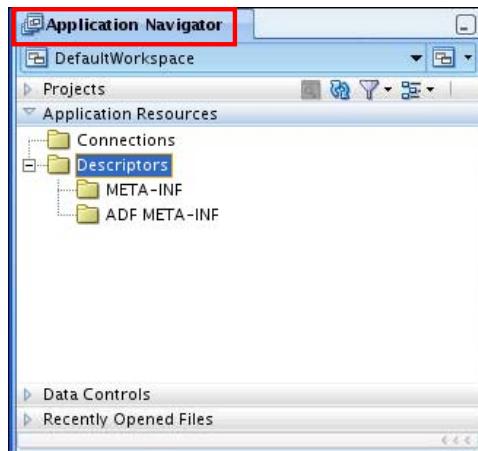
## Editor window

프로그램 단위의 이름을 두 번 눌러 Editor window에서 엽니다. 단일 Editor window에서 프로젝트 파일을 모두 보거나, 한 파일을 여러 뷰로 열거나, 다양한 파일을 여러 뷰로 열 수 있습니다.

편집기 window의 상단에 있는 탭은 문서 탭입니다. 문서 탭을 누르면 해당 파일을 현재 편집기의 window 맨 앞으로 가져와서 해당 파일에 집중시킵니다.

주어진 파일에 대해 편집기 window의 하단에 있는 탭은 편집기 탭입니다. 편집기 탭을 선택하면 해당 편집기에서 파일이 열립니다.

# Application Navigator



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

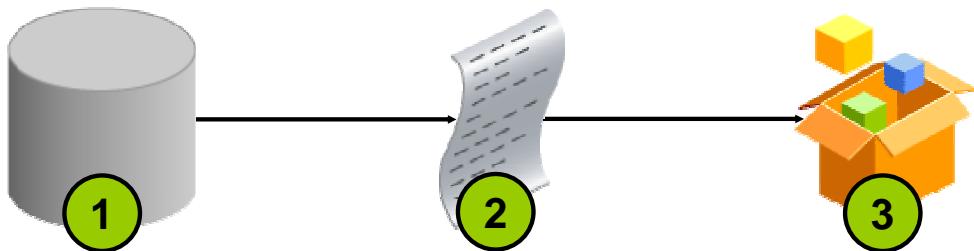
## Application Navigator

Applications - Navigator는 응용 프로그램 및 이 응용 프로그램이 포함하는 데이터의 논리적 뷰를 제공합니다. Applications - Navigator는 다양한 확장을 연결하여 일관되고 추상적인 방식으로 해당 데이터 및 메뉴를 구성하는 데 사용할 수 있는 Infrastructure를 제공합니다. Applications - Navigator는 Java 소스 파일과 같은 개별 파일을 포함할 수 있으며 이때 복잡한 데이터를 통합합니다. 엔티티 객체, UML 다이어그램, EJB 또는 웹 서비스와 같은 복잡한 데이터 유형이 Navigator에 단일 노드로 표시됩니다. 이러한 요약 노드를 구성하는 Raw File은 Structure window에 나타납니다.

# Java 내장 프로시저 배치

Java 내장 프로시저를 배치하기 전에 다음 단계를 수행하십시오.

1. 데이터베이스 접속을 생성합니다.
2. 배치 프로파일을 생성합니다.
3. 객체를 배치합니다.



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## Java 내장 프로시저 배치

Java 내장 프로시저에 대한 배치 프로파일을 생성한 다음 프로파일의 설정을 사용하여 JDeveloper에서 클래스를 배치하고 선택적으로 공용(public) 정적 메소드를 배치합니다.

데이터베이스에 배치할 때에는 Deployment Profile Wizard에 입력한 정보와 다음 두 개의 오라클 데이터베이스 유ти리티를 사용합니다.

- `loadjava`는 내장 프로시저를 포함하는 Java 클래스를 오라클 데이터베이스에 로드합니다.
- `publish`는 로드된 공용(public) 정적 메소드에 대해 PL/SQL 호출 특정 래퍼를 생성합니다. 게시(Publishing) 후에는 Java 메소드를 PL/SQL 함수 또는 프로시저로 호출할 수 있습니다.

# PL/SQL에 Java 게시(Publishing)

The screenshot shows two tabs in the Oracle SQL Developer interface: 'TrimLob.java' and 'TRIMLOBPROC'. The 'TrimLob.java' tab contains Java code for a main method that connects to an Oracle database using JDBC. The 'TRIMLOBPROC' tab contains the PL/SQL code for a procedure named 'TRIMLOBPROC' that calls the Java code.

```
public class TrimLob {
    public static void main (String args[]) throws SQLException {
        Connection conn=null;
        if (System.getProperty("oracle.jserver.version") != null)
        {
            conn = DriverManager.getConnection("jdbc:default:connection:");
        }
        else
        {
            DriverManager.registerDriver(new oracle.jdbc.OracleDriver());
            conn = DriverManager.getConnection("jdbc:oracle:thin:scott/tiger");
        }
    }
}
```

The screenshot shows the 'TRIMLOBPROC' tab in Oracle SQL Developer. It displays the PL/SQL code for the procedure, which includes the language declaration 'language java' and the name of the Java method it calls.

```
CREATE OR REPLACE PROCEDURE TRIMLOBPROC
as language java
name 'TrimLob.main(java.lang.String[])';
/
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

## PL/SQL에 Java 게시(Publishing)

위 슬라이드는 Java 코드 및/oracle Java 코드를 PL/SQL 프로시저에 게시(publish)하는 방법을 보여줍니다.

# JDeveloper 11g에 대해 자세히 배울 수 있는 방법

항목	웹 사이트
Oracle JDeveloper 제품 페이지	<a href="http://www.oracle.com/technology/products/jdev/index.html">http://www.oracle.com/technology/products/jdev/index.html</a>
Oracle JDeveloper 11g 자습서	<a href="http://www.oracle.com/technology/obe/obe11jdev/11/index.html">http://www.oracle.com/technology/obe/obe11jdev/11/index.html</a>
Oracle JDeveloper 11g 제품 설명서	<a href="http://www.oracle.com/technology/documentation/jdev.html">http://www.oracle.com/technology/documentation/jdev.html</a>
Oracle JDeveloper 11g 토의 포럼	<a href="http://forums.oracle.com/forums/forum.jspa?forumID=83">http://forums.oracle.com/forums/forum.jspa?forumID=83</a>

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 요약

이 부록에서는 다음 항목에 대해 설명했습니다.

- Oracle JDeveloper의 주요 기능 나열
- JDeveloper에서 데이터베이스 연결 생성
- JDeveloper에서 데이터베이스 객체 관리
- JDeveloper를 사용하여 SQL 명령 실행
- PL/SQL 프로그램 단위 생성 및 실행



Copyright © 2009, Oracle. All rights reserved.

# Oracle 조인 구문

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

# 목표

이 부록을 마치면 다음을 수행할 수 있습니다.

- **equijoin 및 nonequijoin을 사용하여 두 개 이상의 테이블에서 데이터에 액세스하는 SELECT 문 작성**
- **Self Join을 사용하여 테이블을 자체 조인**
- **Outer join을 사용하여 일반적으로 조인 조건을 충족하지 않는 데이터 보기**
- **두 개 이상의 테이블에서 모든 행의 Cartesian Product 생성**

ORACLE®

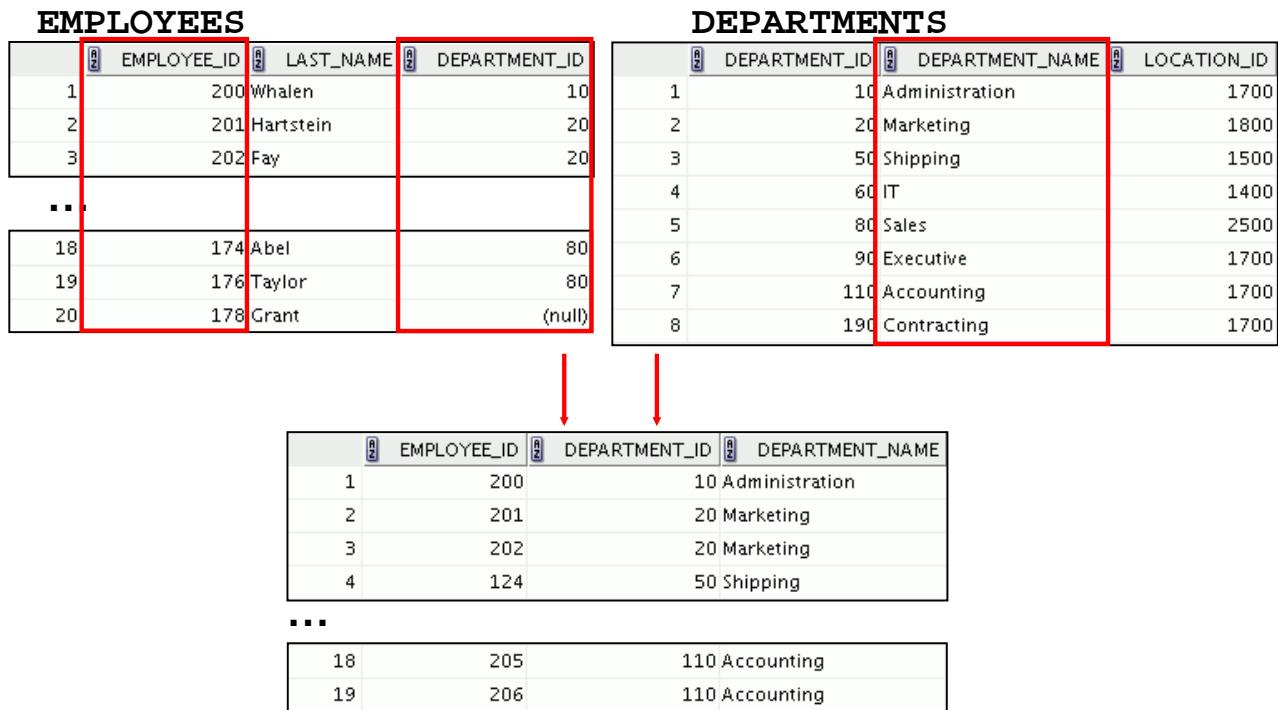
Copyright © 2009, Oracle. All rights reserved.

## 목표

이 단원에서는 두 개 이상의 테이블에서 데이터를 가져오는 방법에 대해 설명합니다. 조인은 여러 테이블의 정보를 보는 데 사용됩니다. 따라서 테이블을 조인하여 두 개 이상의 테이블에 있는 정보를 볼 수 있습니다.

**참고:** 조인에 대한 자세한 내용은 *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*에서 "SQL Queries and Subqueries: Joins" 관련 섹션을 참조하십시오.

## 여러 테이블에서 데이터 가져오기



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 여러 테이블에서 데이터 가져오기

때때로 두 개 이상의 테이블에서 데이터를 사용해야 할 경우가 있습니다. 슬라이드 예제에서는 별도의 두 테이블에서 가져온 데이터가 보고서에 표시됩니다.

- 사원 ID는 EMPLOYEES 테이블에 있습니다.
- 부서 ID는 EMPLOYEES 테이블과 DEPARTMENTS 테이블에 모두 있습니다.
- 부서 이름은 DEPARTMENTS 테이블에 있습니다.

이 보고서를 작성하려면 EMPLOYEES 및 DEPARTMENTS 테이블을 연결하고 두 테이블에서 데이터에 액세스해야 합니다.

# Cartesian Product

- 다음과 같은 경우에 Cartesian Product가 생성됩니다.
  - 조인 조건이 생략된 경우
  - 조인 조건이 잘못된 경우
  - 첫번째 테이블의 모든 행이 두번째 테이블의 모든 행에 조인되는 경우
- Cartesian Product를 피하려면 항상 WHERE 절에 유효한 조인 조건을 포함하십시오.



Copyright © 2009, Oracle. All rights reserved.

## Cartesian Product

조인 조건이 잘못되거나 완전히 생략된 경우 결과는 모든 행 조합이 표시되는 *Cartesian Product*로 나타납니다. 즉, 첫번째 테이블의 모든 행이 두번째 테이블의 모든 행에 조인됩니다.

Cartesian Product는 많은 행을 생성하므로 결과는 그다지 유용하지 않습니다. 따라서 특별히 모든 테이블에서 모든 행을 조합해야 하는 경우가 아니면 항상 유효한 조인 조건을 포함해야 합니다.

그러나 Cartesian Product는 일부 테스트에서 적당량의 데이터를 시뮬레이트하기 위해 많은 행을 생성해야 하는 경우에 유용합니다.

# Cartesian Product 생성

**EMPLOYEES (20행)**

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
4	205	Higgins	110
...			
19	176	Taylor	80
20	178	Grant	(null)

**DEPARTMENTS (8행)**

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

**Cartesian product:**

**20 x 8 = 160행**

	EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	200	10	1700
2	201	20	1700
...			
21	200	10	1800
22	201	20	1800
...			
159	176	80	1700
160	178	(null)	1700

**ORACLE®**

Copyright © 2009, Oracle. All rights reserved.

## Cartesian Product 생성

조인 조건을 생략하면 Cartesian Product가 생성됩니다. 슬라이드의 예제에서는 EMPLOYEES 및 DEPARTMENTS 테이블에서 가져온 사원의 성과 부서 이름을 표시합니다. 조인 조건이 지정되지 않았기 때문에 EMPLOYEES 테이블의 모든 행(20행)이 DEPARTMENTS 테이블의 모든 행(8행)과 조인되어 출력에 160행이 생성됩니다.

```
SELECT last_name, department_name dept_name
FROM employees, departments;
```

	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration
5	Fay	Administration
...		
158	Vargas	Contracting
159	Whalen	Contracting
160	Zlotkey	Contracting

# Oracle 고유의 조인 유형

- **Equijoin**
- **Nonequijoin**
- **Outer Join**
- **Self Join**



Copyright © 2009, Oracle. All rights reserved.

## 조인 유형

테이블을 조인하기 위해 Oracle의 조인 구문을 사용할 수 있습니다.

**참고:** Oracle9i 릴리스 이전에는 고유 조인 구문이 사용되었습니다. SQL:1999 호환 조인 구문은 Oracle 고유의 조인 구문과 비교하여 별다른 성능 이점을 제공하지 않습니다.

Oracle에는 SQL:1999 호환 조인 구문의 FULL OUTER JOIN을 지원하는 해당 구문이 없습니다.

# Oracle 구문을 사용하여 테이블 조인

조인을 사용하여 둘 이상의 테이블에서 데이터를 query합니다.

```
SELECT      table1.column, table2.column
FROM        table1, table2
WHERE       table1.column1 = table2.column2;
```

- WHERE 절에 조인 조건을 작성합니다.
- 둘 이상의 테이블에 동일한 열 이름이 나타날 때는 테이블 이름을 열 이름 앞에 붙입니다.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

## Oracle 구문을 사용하여 테이블 조인

데이터베이스에 있는 둘 이상의 테이블에서 데이터가 필요한 경우 조인 조건을 사용합니다. 해당 열(일반적으로 Primary Key 또는 Foreign Key 열)에 존재하는 공통 값에 따라 한 테이블의 행을 다른 테이블의 행에 조인할 수 있습니다.

둘 이상의 관련 테이블에서 데이터를 표시하려면 WHERE 절에 간단한 조인 조건을 작성합니다. 이 구문에서 다음이 적용됩니다.

<i>table1.column</i>	데이터를 검색할 대상 테이블과 열을 나타냅니다.
<i>table1.column</i> =	테이블을 조인 또는 연관시키는 조건입니다.
<i>table2.column2</i>	

### 지침

- 테이블을 조인하는 SELECT 문을 작성할 때는 열 이름 앞에 테이블 이름을 붙여서 표시를 명확히 하고 데이터베이스 액세스를 향상시킵니다.
- 둘 이상의 테이블에서 동일한 열 이름이 나타나면 열 이름 앞에 테이블 이름을 붙여야 합니다.
- n개의 테이블을 함께 조인하려면 최소 n-1개의 조인 조건이 필요합니다. 예를 들어, 네 개의 테이블을 조인하려면 최소 세 개의 조인이 필요합니다. 이 규칙은 테이블에 연결된 primary key가 있을 때는 적용되지 않습니다. 이 경우에는 각 행을 고유하게 식별하기 위해 둘 이상의 열이 필요합니다.

## 모호한 열 이름 한정

- 테이블 접두어를 사용하여 여러 테이블에 있는 열 이름을 한정합니다.
- 테이블 접두어를 사용하여 성능을 향상합니다.
- 전체 테이블 이름 접두어 대신 테이블 alias를 사용합니다.
- 테이블 alias로 테이블에 짧은 이름을 지정합니다.
  - SQL 코드 크기를 줄여 메모리를 적게 사용합니다.
- 열 alias를 사용하여 이름은 같지만 서로 다른 테이블에 상주하는 열을 구분합니다.

**ORACLE**

Copyright © 2009, Oracle. All rights reserved.

### 모호한 열 이름 한정

두 개 이상의 테이블을 조인하는 경우 모호성을 피하기 위해 열 이름을 테이블 이름으로 한정해야 합니다. 테이블 접두어를 사용하지 않으면 SELECT 리스트의 DEPARTMENT\_ID 열을 DEPARTMENTS 테이블이나 EMPLOYEES 테이블에서 가져올 수 있습니다. 따라서 query를 실행하려면 테이블 접두어를 추가해야 합니다. 두 테이블 간에 공통되는 열 이름이 없으면 열을 한정할 필요가 없습니다. 하지만 테이블 접두어를 사용하면 Oracle 서버에 열을 찾을 위치를 정확히 알려줄 수 있으므로 성능이 향상됩니다.

테이블 이름으로 열 이름을 한정하는 것은 시간이 많이 소요되는 작업이 될 수 있으며 테이블 이름이 길 경우 더욱 그렇습니다. 따라서 테이블 이름 대신 테이블 alias를 사용할 수 있습니다. 열 alias로 열에 다른 이름을 지정할 수 있는 것처럼 테이블에도 테이블 alias로 다른 이름을 지정합니다. 테이블 alias를 사용하면 SQL 코드를 더 쉽게 유지할 수 있으므로 메모리 사용량도 줄어듭니다.

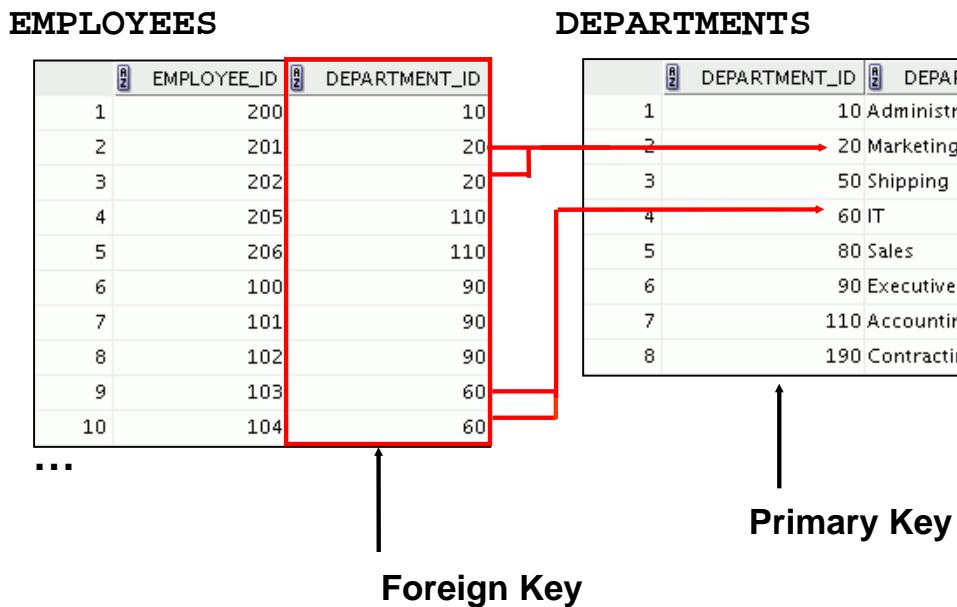
테이블 이름은 완전하게 지정되며 뒤에 공백이 온 후에 테이블 alias가 옵니다. 예를 들어 EMPLOYEES 테이블에는 e라는 alias를 지정하고 DEPARTMENTS 테이블에는 d라는 alias를 지정할 수 있습니다.

## 모호한 열 이름 한정(계속)

### 지침

- 테이블 alias는 30자까지 사용할 수 있지만 길이는 짧을수록 좋습니다.
- FROM 절의 특정 테이블 이름에 대해 테이블 alias가 사용될 경우 SELECT 문 전체에서 테이블 이름 대신 해당 테이블 alias를 사용해야 합니다.
- 테이블 alias는 의미 있는 단어여야 합니다.
- 테이블 alias는 현재 SELECT 문에 대해서만 유효합니다.

# Equijoin



ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## Equijoin

사원의 부서 이름을 확인하려면 EMPLOYEES 테이블의 DEPARTMENT\_ID 열의 값을 DEPARTMENTS 테이블의 DEPARTMENT\_ID 값과 비교합니다. EMPLOYEES 테이블과 DEPARTMENTS 테이블 사이의 관계는 *equijoin*입니다. 즉, 두 테이블의 DEPARTMENT\_ID 열에 있는 값이 같아야 합니다. 대개 이러한 유형의 조인에는 Primary Key 및 Foreign Key가 보완 요소로 포함됩니다.

참고: equijoin은 *simple join* 또는 *inner join*이라고도 합니다.

## Equijoin을 사용하여 레코드 검색

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
  FROM employees e, departments d
 WHERE e.department_id = d.department_id;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200 Whalen	10	10	1700
2	201 Hartstein	20	20	1800
3	202 Fay	20	20	1800
4	144 Vargas	50	50	1500
5	143 Matos	50	50	1500
6	142 Davies	50	50	1500
7	141 Rajs	50	50	1500
8	124 Mourgos	50	50	1500
9	103 Hunold	60	60	1400
10	104 Ernst	60	60	1400
11	107 Lorentz	60	60	1400
...				

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### Equijoin을 사용하여 레코드 검색

슬라이드의 예제는 다음과 같습니다.

- **SELECT** 절은 다음과 같이 검색할 열 이름을 지정합니다.
  - 사원의 성, 사원 번호 및 부서 번호(EMPLOYEES 테이블의 열)
  - 부서 번호, 부서 이름 및 위치 ID(DEPARTMENTS 테이블의 열)
- **FROM** 절은 데이터베이스에서 액세스해야 하는 다음의 두 테이블을 지정합니다.
  - EMPLOYEES 테이블
  - DEPARTMENTS 테이블
- **WHERE** 절은 테이블이 조인되는 방식을 지정합니다.
   
`e.department_id = d.department_id`

DEPARTMENT\_ID 열이 두 테이블에 공통되므로 명확하게 구분하기 위해서는 열 이름 앞에 테이블 alias를 붙여야 합니다. 두 테이블 모두에 있지 않은 다른 열은 테이블 alias로 한정할 필요는 없지만 더 나은 성능을 위해 이렇게 하는 것이 좋습니다.

참고: SQL Developer에서 Execute Statement 아이콘을 사용하여 query를 실행할 경우 두 개의 DEPARTMENT\_ID를 구분하기 위해 뒤에 "\_1"이 붙습니다.

## Equijoin을 사용하여 레코드 검색: 예제

```
SELECT d.department_id, d.department_name,
       d.location_id, l.city
  FROM departments d, locations l
 WHERE d.location_id = l.location_id;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	90	Executive	1700	Seattle
5	110	Accounting	1700	Seattle
6	190	Contracting	1700	Seattle
7	20	Marketing	1800	Toronto
8	80	Sales	2500	Oxford

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### Equijoin을 사용하여 레코드 검색: 예제

슬라이드의 예제에서는 LOCATIONS 테이블과 DEPARTMENT 테이블을 두 테이블에서 유일하게 이름이 같은 열인 LOCATION\_ID 열로 조인합니다. 열을 한정하고 모호성을 방지하기 위해 테이블 alias를 사용합니다.

## AND 연산자를 사용한 추가 검색 조건

```
SELECT d.department_id, d.department_name, l.city
FROM departments d, locations l
WHERE d.location_id = l.location_id
AND d.department_id IN (20, 50);
```

	DEPARTMENT_ID	DEPARTMENT_NAME	CITY
1	20	Marketing	Toronto
2	50	Shipping	South San Francisco

ORACLE

Copyright © 2009, Oracle. All rights reserved.

## AND 연산자를 사용한 추가 검색 조건

조인과 함께 WHERE 절에 대한 조건을 설정하여 조인에서 하나 이상의 테이블에 대해 고려 대상인 행을 제한할 수 있습니다. 슬라이드의 예제에서는 출력 행을 부서 ID가 20 또는 50인 행으로 제한합니다.

예를 들어, Matos라는 사원의 부서 번호와 부서 이름을 표시하려면 WHERE 절에서 다음과 같은 추가 조건이 필요합니다.

```
SELECT e.last_name, e.department_id,
       d.department_name
  FROM employees e, departments d
 WHERE e.department_id = d.department_id
   AND last_name = 'Matos';
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Matos	50	Shipping

## 세 개 이상의 테이블 조인

**EMPLOYEES**

LAST_NAME	DEPARTMENT_ID
King	90
Kochhar	90
De Haan	90
Hunold	60
Ernst	60
Lorentz	60
Mourgos	50
Rajs	50
Davies	50
Matos	50

...

**DEPARTMENTS**

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

**LOCATIONS**

LOCATION_ID	CITY
1400	Southlake
1500	South San Francisco
1700	Seattle
1800	Toronto
2500	Oxford

*n*개의 테이블을 함께 조인하려면 최소 *n*-1개의 조인 조건이 필요합니다. 예를 들어, 세 개의 테이블을 조인하려면 최소 두 개의 조인이 필요합니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 세 개 이상의 테이블 조인

때때로 세 개 이상의 테이블을 조인해야 할 필요가 있습니다. 예를 들어, 각 사원의 성, 부서 이름 및 근무지를 표시하려면 EMPLOYEES, DEPARTMENTS 및 LOCATIONS 테이블을 조인해야 합니다.

```
SELECT e.last_name, d.department_name, l.city
  FROM employees e, departments d, locations l
 WHERE e.department_id = d.department_id
   AND d.location_id = l.location_id;
```

LAST_NAME	DEPARTMENT_NAME	CITY
Abel	Sales	Oxford
Davies	Shipping	South San Francisco
De Haan	Executive	Seattle
Ernst	IT	Southlake
Fay	Marketing	Toronto

...

# Nonequijoin

**EMPLOYEES**

	LAST_NAME	SALARY
1	Whalen	4400
2	Hartstein	13000
3	Fay	6000
4	Higgins	12000
5	Gietz	8300
6	King	24000
7	Kochhar	17000
8	De Haan	17000
9	Hunold	9000
10	Ernst	6000
...		
19	Taylor	8600
20	Grant	7000

**JOB\_GRADES**

	GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
1	A	1000	2999
2	B	3000	5999
3	C	6000	9999
4	D	10000	14999
5	E	15000	24999
6	F	25000	40000

JOB\_GRADES 테이블에서는 각 GRADE\_LEVEL에 대해 LOWEST\_SAL 및 HIGHEST\_SAL 값의 범위를 정의합니다. 따라서 GRADE\_LEVEL 열을 사용하여 각 사원에 등급을 지정할 수 있습니다.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

## Nonequijoin

Nonequijoin은 등호 연산자 외의 다른 연산자를 포함하는 조인 조건입니다.

Nonequijoin의 한 예로 EMPLOYEES 테이블과 JOB\_GRADES 테이블 사이의 관계를 들 수 있습니다. EMPLOYEES 테이블의 SALARY 열은 JOB\_GRADES 테이블의 LOWEST\_SAL 열에 있는 값과 HIGHEST\_SAL 열에 있는 값 사이의 범위에 해당하는 값을 가집니다. 따라서 급여를 기준으로 각 사원의 등급을 지정할 수 있습니다. 이때 등호(=) 연산자가 아닌 다른 연산자를 사용하여 관계를 구합니다.

## Nonequijoin을 사용하여 레코드 검색

```
SELECT e.last_name, e.salary, j.grade_level
FROM employees e, job_grades j
WHERE e.salary
      BETWEEN j.lowest_sal AND j.highest_sal;
```

	LAST_NAME	SALARY	GRADE_LEVEL
1	Vargas	2500	A
2	Matos	2600	A
3	Davies	3100	B
4	Rajs	3500	B
5	Lorentz	4200	B
6	Whalen	4400	B
7	Mourgos	5800	B
8	Ernst	6000	C
9	Fay	6000	C
10	Grant	7000	C
...			

ORACLE

Copyright © 2009, Oracle. All rights reserved.

### Nonequijoin을 사용하여 레코드 검색

슬라이드의 예제에서는 사원의 급여 등급을 평가하는 nonequijoin을 생성합니다. 낮은 급여 범위와 높은 급여 범위 쌍(pair) 사이에 있어야 합니다.

이 query가 실행될 때 각 사원은 한 번만 나타납니다. 리스트에서 반복되는 사원은 없습니다. 여기에는 다음과 같은 두 가지 이유가 있습니다.

- 직무 등급 테이블의 어떠한 행도 중복되는 등급을 포함하지 않습니다. 즉, 한 사원에 대한 급여 값은 급여 등급 테이블의 한 행에서 낮은 급여 값과 높은 급여 값 범위에만 속할 수 있습니다.
- 모든 사원의 급여는 직무 등급 테이블에서 제공하는 제한 범위 내에 놓입니다. 즉, LOWEST\_SAL 열에 포함된 최저값보다 적은 급여를 받거나 HIGHEST\_SAL 열에 포함된 최고값보다 많은 급여를 받는 사원은 없습니다.

**참고:** <= 및 >=와 같은 다른 조건을 사용할 수 있지만 BETWEEN이 가장 간단합니다. BETWEEN 조건을 사용할 때는 맨 처음에 낮은 값을 지정하고 마지막에 높은 값을 지정해야 합니다. Oracle 서버는 BETWEEN 조건을 AND 조건 쌍(pair)으로 변환합니다. 따라서 BETWEEN을 사용해도 성능상 이점이 전혀 없으므로 논리적으로 간결한 SQL 문을 작성하는 경우에만 BETWEEN을 사용해야 합니다.

슬라이드의 예제에서는 모호성을 피하기 위해서가 아니라 성능상의 이유로 테이블 alias를 지정했습니다.

# Outer join을 사용하여 직접 일치되지 않는 레코드 반환

DEPARTMENTS

	DEPARTMENT_NAME	DEPARTMENT_ID
1	Administration	10
2	Marketing	20
3	Shipping	50
4	IT	60
5	Sales	80
6	Executive	90
7	Accounting	110
8	Contracting	190

EMPLOYEES

	DEPARTMENT_ID	LAST_NAME
1	10	Whalen
2	20	Hartstein
3	20	Fay
4	110	Higgins
5	110	Gietz
6	90	King
7	90	Kochhar
8	90	De Haan
9	60	Hunold
10	60	Ernst
...		
18	80	Abel
19	80	Taylor

부서 190에는 사원이 없습니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## Outer join을 사용하여 직접 일치되지 않는 레코드 반환

조인 조건을 충족하지 못하는 행은 query 결과에 나타나지 않습니다. 예를 들어, EMPLOYEES 테이블과 DEPARTMENTS 테이블의 Equijoin 조건에서 부서 ID 190은 나타나지 않습니다. 해당 부서 ID를 가진 사원이 EMPLOYEES 테이블에 기록되어 있지 않기 때문입니다. 또한 DEPARTMENT\_ID가 NULL로 설정된 사원이 있으므로 해당 행도 Equijoin의 query 결과에 표시되지 않습니다. 사원이 없는 부서 레코드를 반환하거나 부서에 속하지 않는 사원 레코드를 반환하려면 Outer join을 사용할 수 있습니다.

## Outer Join: 구문

- Outer join을 사용하면 조인 조건을 만족하지 않는 행을 볼 수 있습니다.
- outer join 연산자는 더하기 기호(+)입니다.

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column = table2.column(+);
```

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### Outer Join: 구문

조인 조건에서 *outer join* 연산자를 사용하면 누락된 행이 반환될 수 있습니다. 이 연산자는 더하기 기호를 괄호로 묶은 (+)이며, 조인에서 누락된 정보가 있는 "쪽"에 놓입니다. 이 연산자는 하나 이상의 null 행이 생성되는 결과를 가져오며 이 null 행에 누락되지 않은 테이블에 있는 하나 이상의 행을 조인할 수 있습니다.

이 구문에서 다음이 적용됩니다.

*table1.column* = 테이블을 조인 또는 연관시키는 조건입니다.

*table2.column* (+) Outer join 기호입니다. WHERE 절 조건의 양쪽 중 한쪽에 놓을 수 있습니다. 일치하는 행이 없는 테이블의 열 이름 다음에 Outer join 기호를 놓습니다.

## Outer Join 사용

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e, departments d
WHERE e.department_id(+) = d.department_id ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping
6	Rajs	50	Shipping
7	Mourgos	50	Shipping
8	Matos	50	Shipping
9	Hunold	60	IT
10	Ernst	60	IT
...			
19	Gietz	110	Accounting
20	(null)	(null)	Contracting

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### Outer Join 사용

슬라이드의 예제는 사원의 성, 부서 ID 및 부서 이름을 표시합니다. Contracting 부서에는 사원이 없습니다. 따라서 출력에 빈 값이 표시됩니다.

#### Outer Join 제한 사항

- Outer Join 연산자는 표현식의 한쪽, 즉 정보가 누락된 쪽에만 나타날 수 있습니다. 이 연산자는 테이블에서 다른 테이블과 직접 일치되는 항목이 없는 행을 반환합니다.
- Outer Join을 포함하는 조건은 IN 연산자를 사용할 수 없으며 OR 연산자를 통해 다른 조건에 연결될 수 없습니다.

참고: Oracle에는 SQL:1999 호환 조인 구문의 FULL OUTER JOIN에 해당하는 조인 구문이 없습니다.

## Outer Join: 다른 예제

```
SELECT e.last_name, e.department_id, d.department_name  
FROM employees e, departments d  
WHERE e.department_id = d.department_id(+);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping

16	Kochhar	90	Executive
17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### Outer Join: 다른 예제

슬라이드 예제의 query에서는 DEPARTMENTS 테이블에 일치하는 행이 없어도 EMPLOYEES 테이블의 모든 행을 검색합니다.

## 테이블 자체 조인

**EMPLOYEES (WORKER)**

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
200	Whalen	101
201	Hartstein	100
202	Fay	201
205	Higgins	101
206	Gietz	205
100	King	(null)
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103

...

**EMPLOYEES (MANAGER)**

EMPLOYEE_ID	LAST_NAME
200	Whalen
201	Hartstein
202	Fay
205	Higgins
206	Gietz
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst

...

WORKER 테이블의 MANAGER\_ID는 MANAGER 테이블의 EMPLOYEE\_ID와 같습니다.

Copyright © 2009, Oracle. All rights reserved.

### 테이블 자체 조인

때때로 테이블을 자체 조인해야 할 경우가 있습니다. 각 사원의 관리자 이름을 찾으려면 EMPLOYEES 테이블을 자체 조인하거나 Self Join을 수행해야 합니다. 예를 들어, Lorentz의 관리자 이름을 찾으려면 다음을 수행해야 합니다.

- EMPLOYEES 테이블에서 LAST\_NAME 열을 조사하여 Lorentz를 찾습니다.
- MANAGER\_ID 열을 조사하여 Lorentz의 관리자 번호를 찾습니다. Lorentz의 관리자 번호는 103입니다.
- LAST\_NAME 열을 조사하여 EMPLOYEE\_ID가 103인 관리자의 이름을 찾습니다. Hunold의 사원 번호가 103이므로 Hunold가 Lorentz의 관리자입니다.

본 과정에서 EMPLOYEES 테이블을 두 번 조사하게 됩니다. 먼저 테이블을 조사하여 LAST\_NAME 열에서 Lorentz를 찾고 MANAGER\_ID 값 103을 찾습니다. 이어서 EMPLOYEE\_ID 열을 조사하여 103을 찾고 LAST\_NAME 열에서 Hunold를 찾습니다.

## Self Join: 예제

```
SELECT worker.last_name || ' works for '
    || manager.last_name
  FROM employees worker, employees manager
 WHERE worker.manager_id = manager.employee_id ;
```

WORKER.LAST_NAME  'WORKSFOR'  MANAGER.LAST_NAME
1 Hunold works for De Haan
2 Fay works for Hartstein
3 Gietz works for Higgins
4 Lorentz works for Hunold
5 Ernst works for Hunold
6 Zlotkey works for King
7 Mourgos works for King
8 Kochhar works for King
9 Hartstein works for King
10 De Haan works for King
...

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### Self Join: 예제

슬라이드의 예제에서는 EMPLOYEES 테이블을 자체 조인합니다. FROM 절의 두 테이블을 시뮬레이트하기 위해 동일한 테이블 EMPLOYEES에 대해 두 개의 alias worker 및 manager를 사용합니다.

이 예제에서 WHERE 절은 "작업자의 관리자 번호가 해당 관리자에 대한 사원 번호와 일치함"을 의미하는 조인을 포함합니다.

# 요약

이 부록에서는 Oracle 고유의 구문을 통해 조인을 사용하여 여러 테이블의 데이터를 표시하는 방법을 배웠습니다.

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

## 요약

다양한 방법으로 테이블을 조인할 수 있습니다.

### 조인 유형

- Equijoin
- Nonequijoin
- Outer join
- Self Join

### Cartesian Product

Cartesian Product는 결과에 모든 행 조합을 표시합니다. WHERE 절을 생략하면 이러한 결과가 나타납니다.

### 테이블 alias

- 테이블 alias는 데이터베이스 액세스 속도를 높입니다.
- 테이블 alias를 사용하면 SQL 코드를 더 작게 유지할 수 있으므로 메모리 사용량도 줄어듭니다.

## 연습 F: 개요

이 연습에서는 다음 내용을 다룹니다.

- Equijoin을 사용하여 테이블 조인
- Outer Join 및 Self Join 수행
- 조건 추가

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

### 연습 F: 개요

이 연습은 Oracle 조인 구문을 사용하여 두 개 이상의 테이블에서 데이터를 추출하는 과정을 실습할 수 있도록 구성되었습니다.

---

---

**부록 AP**  
**추가 연습 및 해답**

---

## 목차

추가 연습 .....	3
연습 1-1 .....	4
연습 1-1 해답 .....	12
사례 연구 .....	17
연습 2-1 .....	19
연습 2-1 해답 .....	27

## **추가 연습**

## 연습 1-1

다음 연습은 SQL 기본 SQL SELECT 문, 기본 SQL Developer 명령, SQL 함수 등의 항목을 살펴 본 후에 추가로 수행할 수 있습니다.

- 1) HR 부서에서 1997년 이후 채용된 모든 clerk에 대한 데이터를 찾으려고 합니다.

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	2600
2	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	2500

- 2) HR 부서에서 커미션을 받는 사원에 대한 보고서를 요구합니다. 해당 사원의 성, 직무, 급여 및 커미션을 표시합니다. 급여의 내림차순으로 데이터를 정렬합니다.

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
Abel	SA_REP	11000	0.3
Zlotkey	SA_MAN	10500	0.2
Taylor	SA_REP	8600	0.2
Grant	SA_REP	7000	0.15

- 3) HR 부서에서 예산 책정을 위해 예상되는 급여 인상에 대한 보고서를 요구합니다.  
이 보고서는 커미션을 받지 않지만 급여가 10% 인상되는 사원을 표시해야 합니다(급여 반올림).

New salary
1 The salary of King after a 10% raise is 26400
2 The salary of Kochhar after a 10% raise is 18700
3 The salary of De Haan after a 10% raise is 18700
4 The salary of Hunold after a 10% raise is 9900
5 The salary of Ernst after a 10% raise is 6600
6 The salary of Lorentz after a 10% raise is 4620
7 The salary of Mourgos after a 10% raise is 6380
8 The salary of Rajs after a 10% raise is 3850
9 The salary of Davies after a 10% raise is 3410
10 The salary of Matos after a 10% raise is 2860
11 The salary of Vargas after a 10% raise is 2750
12 The salary of Whalen after a 10% raise is 4840
13 The salary of Hartstein after a 10% raise is 14300
14 The salary of Fay after a 10% raise is 6600
15 The salary of Higgins after a 10% raise is 13200
16 The salary of Gietz after a 10% raise is 9130

## 연습 1-1 (계속)

- 4) 사원 및 근속 기간에 대한 보고서를 작성합니다. 모든 사원들의 성 및 근무 기간(년, 개월)을 함께 표시합니다. 근속 기간별로 보고서를 정렬합니다.  
근속 기간이 가장 긴 사원이 리스트의 맨 위에 나타나야 합니다.

	LAST_NAME	YEARS	MONTHS
1	King	22	0
2	Whalen	21	9
3	Kochhar	19	9
4	Hunold	19	6
5	Ernst	18	1
6	De Haan	16	6
7	Higgins	15	1
8	Gietz	15	1
9	Rajs	13	8
10	Hartstein	13	4
11	Abel	13	2
12	Davies	12	5
13	Fay	11	10
14	Matos	11	4
15	Taylor	11	3
16	Vargas	11	0
17	Lorentz	10	5
18	Grant	10	1
19	Mourgos	9	7
20	Zlotkey	9	5

- 5) 성이 "J", "K", "L" 또는 "M"으로 시작하는 사원을 표시합니다.

	LAST_NAME
1	King
2	Kochhar
3	Lorentz
4	Matos
5	Mourgos

## 연습 1-1 (계속)

- 6) 모든 사원을 표시하고 각 사원이 커미션을 받는지 여부를 Yes 또는 No로 나타내는 보고서를 작성합니다. query에서 DECODE 식을 사용합니다.

	LAST_NAME	SALARY	COMMISSION
1	King	24000	No
2	Kochhar	17000	No
3	De Haan	17000	No
4	Hunold	9000	No
5	Ernst	6000	No
6	Lorentz	4200	No
7	Mourgos	5800	No
8	Rajs	3500	No
9	Davies	3100	No
10	Matos	2600	No
11	Vargas	2500	No
12	Zlotkey	10500	Yes
13	Abel	11000	Yes
14	Taylor	8600	Yes
15	Grant	7000	Yes
16	Whalen	4400	No
17	Hartstein	13000	No
18	Fay	6000	No
19	Higgins	12000	No
20	Gietz	8300	No

다음 연습은 기본 SQL SELECT 문, 기본 SQL Developer 명령, SQL 함수, 조인, 그룹 함수 등의 항목을 살펴본 후에 추가로 수행할 수 있습니다.

- 7) 특정 위치에서 근무하는 사원의 부서 이름, 위치 ID, 성, 직책 및 급여를 표시하는 보고서를 작성합니다. 유저에게 위치를 입력하는 프롬프트를 표시합니다. 예를 들어, 유저가 1800을 입력하면 출력 결과는 다음과 같습니다.

	DEPARTMENT_NAME	LOCATION_ID	LAST_NAME	JOB_ID	SALARY
1	Marketing	1800	Hartstein	MK_MAN	13000
2	Marketing	1800	Fay	MK_REP	6000

- 8) 성이 "n"으로 끝나는 사원의 수를 알아냅니다. 가능한 두 가지 해결책을 작성합니다.

	COUNT(*)
1	3

## 연습 1-1 (계속)

- 9) 각 부서에 대한 이름, 위치 및 사원 수를 보고서를 작성합니다.  
보고서에 사원이 없는 부서도 포함되는지 확인합니다.

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	COUNT(E.EMPLOYEE_ID)
1	80	Sales	2500	3
2	110	Accounting	1700	2
3	10	Administration	1700	1
4	60	IT	1400	3
5	20	Marketing	1800	2
6	90	Executive	1700	3
7	50	Shipping	1500	5
8	190	Contracting	1700	0

- 10) HR 부서에서 부서 번호 10 및 20에 있는 직책을 찾으려고 합니다. 해당 부서의 직무 ID를 표시하는 보고서를 작성합니다.

JOB_ID
AD_ASST
MK_MAN
MK_REP

- 11) Administration 및 Executive 부서에서 찾은 직무를 표시하는 보고서를 작성합니다.  
또한 해당 직무에 대한 사원 수도 표시합니다. 사원 수가 가장 많은 직무를 가장 먼저 표시합니다.

JOB_ID	FREQUENCY
AD_VP	2
AD_PRES	1
AD_ASST	1

다음 연습은 기본 SQL SELECT 문, 기본 SQL Developer 명령, SQL 함수, 조인, 그룹 함수, subquery 등의 항목을 살펴본 후에 추가로 수행할 수 있습니다.

- 12) 각 월의 16일 이전에 채용된 사원을 모두 표시합니다.

LAST_NAME	HIRE_DATE
De Haan	13-JAN-93
Hunold	03-JAN-90
Lorentz	07-FEB-99
Matos	15-MAR-98
Vargas	09-JUL-98
Abel	11-MAY-96
Higgins	07-JUN-94
Gietz	07-JUN-94

## 연습 1-1 (계속)

- 13) 모든 사원에 대해 성, 급여 및 \$1000 단위로 표현된 급여를 표시하는 보고서를 작성합니다.

	LAST_NAME	SALARY	THOUSANDS
1	King	24000	24
2	Kochhar	17000	17
3	De Haan	17000	17
4	Hunold	9000	9
5	Ernst	6000	6
6	Lorentz	4200	4
7	Mourgos	5800	5
8	Rajs	3500	3
9	Davies	3100	3
10	Matos	2600	2
11	Vargas	2500	2
12	Zlotkey	10500	10
13	Abel	11000	11
14	Taylor	8600	8
15	Grant	7000	7
16	Whalen	4400	4
17	Hartstein	13000	13
18	Fay	6000	6
19	Higgins	12000	12
20	Gietz	8300	8

- 14) 급여가 \$15,000 이상인 관리자 휘하의 모든 사원을 표시합니다. 사원 이름, 관리자 이름, 관리자 급여 및 관리자의 급여 등급을 표시합니다.

	LAST_NAME	MANAGER	SALARY	GRADE_LEVEL
1	De Haan	King	24000 E	
2	Hartstein	King	24000 E	
3	Higgins	Kochhar	17000 E	
4	Hunold	De Haan	17000 E	
5	Kochhar	King	24000 E	
6	Mourgos	King	24000 E	
7	Whalen	Kochhar	17000 E	
8	Zlotkey	King	24000 E	

## 연습 1-1 (계속)

- 15) 모든 부서의 부서 번호, 이름, 사원 수, 평균 급여와 각 부서에서 일하는 사원의 이름, 급여 및 직무를 표시합니다.

	DEPARTMENT_ID	DEPARTMENT_NAME	EMPLOYEES	Avg_Sal	Last_Name	Salary	Job_ID
1	10	Administration	1	4400.00	Whalen	4400	AD_ASST
2	20	Marketing	2	9500.00	Hartstein	13000	MK_MAN
3	20	Marketing	2	9500.00	Fay	6000	MK_REP
4	50	Shipping	5	3500.00	Davies	3100	ST_CLERK
5	50	Shipping	5	3500.00	Matos	2600	ST_CLERK
6	50	Shipping	5	3500.00	Rajs	3500	ST_CLERK
7	50	Shipping	5	3500.00	Mourgos	5800	ST_MAN
8	50	Shipping	5	3500.00	Vargas	2500	ST_CLERK
9	60	IT	3	6400.00	Hunold	9000	IT_PROG
10	60	IT	3	6400.00	Lorentz	4200	IT_PROG
11	60	IT	3	6400.00	Ernst	6000	IT_PROG
12	80	Sales	3	10033.33	Zlotkey	10500	SA_MAN
13	80	Sales	3	10033.33	Taylor	8600	SA_REP
14	80	Sales	3	10033.33	Abel	11000	SA_REP
15	90	Executive	3	19333.33	Kochhar	17000	AD_VP
16	90	Executive	3	19333.33	De Haan	17000	AD_VP
17	90	Executive	3	19333.33	King	24000	AD_PRES
18	110	Accounting	2	10150.00	Gietz	8300	AC_ACCOUNT
19	110	Accounting	2	10150.00	Higgins	12000	AC_MGR
20	(null)	(null)	0	No average	Grant	7000	SA_REP

- 16) 평균 급여가 가장 높은 부서의 부서 번호와 최저 급여를 표시하는 보고서를 작성합니다.

	DEPARTMENT_ID	MIN(SALARY)
1	90	17000

- 17) 영업 사원이 근무하지 않는 부서를 표시하는 보고서를 작성합니다. 출력에 부서 번호, 부서 이름, 관리자 ID 및 위치를 포함합니다.

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	90	Executive	100	1700
6	110	Accounting	205	1700
7	190	Contracting	(null)	1700

## 연습 1-1 (계속)

18) HR 부서를 위해 통계 보고서를 작성합니다. 이 보고서에는 다음 조건의 부서에 대한 부서 번호, 부서 이름 및 근무하는 사원 수를 표시합니다.

a) 사원 수가 3명 미만인 부서:

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	10	Administration	1
2	110	Accounting	2
3	20	Marketing	2

b) 사원 수가 가장 많은 부서:

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	50	Shipping	5

c) 사원 수가 가장 적은 부서:

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	10	Administration	1

19) 모든 사원에 대해 사원 번호, 성, 급여, 부서 번호 및 해당 부서의 평균 급여를 표시하는 보고서를 작성합니다.

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	SALARY	AVG(S.SALARY)
1	149	Zlotkey	80	10500	10033.333333333...
2	174	Abel	80	11000	10033.333333333...
3	144	Vargas	50	2500	3500
4	101	Kochhar	90	17000	19333.333333333...
5	100	King	90	24000	19333.333333333...
6	103	Hunold	60	9000	6400
7	142	Davies	50	3100	3500
8	205	Higgins	110	12000	10150
9	104	Ernst	60	6000	6400
10	143	Matos	50	2600	3500
11	102	De Haan	90	17000	19333.333333333...
12	107	Lorentz	60	4200	6400
13	141	Rajs	50	3500	3500
14	200	Whalen	10	4400	4400
15	202	Fay	20	6000	9500
16	176	Taylor	80	8600	10033.333333333...
17	201	Hartstein	20	13000	9500
18	206	Gietz	110	8300	10150
19	124	Mourgos	50	5800	3500

## 연습 1-1 (계속)

20) 사원 채용 수가 가장 많은 요일에 채용된 사원을 모두 표시합니다.

	LAST_NAME	DAY
1	Ernst	TUESDAY
2	Mourgos	TUESDAY
3	Rajs	TUESDAY
4	Taylor	TUESDAY
5	Higgins	TUESDAY
6	Gietz	TUESDAY

21) 사원의 채용일에 준하여 기념일 개요를 생성합니다. 기념일을 오름차순으로 정렬하십시오.

	LAST_NAME	BIRTHDAY
1	Hunold	January 03
2	De Haan	January 13
3	Davies	January 29
4	Zlotkey	January 29
5	Lorentz	February 07
6	Hartstein	February 17
7	Matos	March 15
8	Taylor	March 24
9	Abel	May 11
10	Ernst	May 21
11	Grant	May 24
12	Higgins	June 07
13	Gietz	June 07
14	King	June 17
15	Vargas	July 09
16	Fay	August 17
17	Whalen	September 17
18	Kochhar	September 21
19	Rajs	October 17
20	Mourgos	November 16

## 연습 1-1 해답

다음 연습은 SQL 기본 SELECT 문, 기본 SQL Developer 명령, SQL 함수 등의 항목을 살펴 본 후에 추가로 수행할 수 있습니다.

- 1) HR 부서에서 1997년 이후 채용된 모든 clerk에 대한 데이터를 찾으려고 합니다.

```
SELECT *
FROM   employees
WHERE  job_id = 'ST_CLERK'
AND    hire_date > '31-DEC-1997';
```

- 2) HR 부서에서 커미션을 받는 사원에 대한 보고서를 요구합니다. 해당 사원의 성, 직무, 급여 및 커미션을 표시합니다. 급여의 내림차순으로 데이터를 정렬합니다.

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees
WHERE  commission_pct IS NOT NULL
ORDER BY salary DESC;
```

- 3) HR 부서에서 예산 책정을 위해 예상되는 급여 인상에 대한 보고서를 요구합니다. 이 보고서는 커미션을 받지 않지만 급여가 10% 인상되는 사원을 표시해야 합니다(급여 반올림).

```
SELECT 'The salary of '||last_name||' after a 10% raise is '
      || ROUND(salary*1.10) "New salary"
FROM   employees
WHERE  commission_pct IS NULL;
```

- 4) 사원 및 근속 기간에 대한 보고서를 작성합니다. 모든 사원들의 성 및 근무 기간(년, 개월)을 함께 표시합니다. 근속 기간별로 보고서를 정렬합니다. 근속 기간이 가장 긴 사원이 리스트의 맨 위에 나타나야 합니다.

```
SELECT last_name,
       TRUNC(MONTHS_BETWEEN(SYSDATE, hire_date) / 12) YEARS,
       TRUNC(MOD(MONTHS_BETWEEN(SYSDATE, hire_date), 12))
             MONTHS
  FROM employees
 ORDER BY years DESC, months desc;
```

- 5) 성이 "J", "K", "L" 또는 "M"으로 시작하는 사원을 표시합니다.

```
SELECT last_name
  FROM   employees
 WHERE  SUBSTR(last_name, 1,1) IN ('J', 'K', 'L', 'M');
```

- 6) 모든 사원을 표시하고 각 사원이 커미션을 받는지 여부를 Yes 또는 No로 나타내는 보고서를 작성합니다. query에서 DECODE 식을 사용합니다.

```
SELECT last_name, salary,
       decode(commission_pct, NULL, 'No', 'Yes') commission
  FROM   employees;
```

## 연습 1-1 해답 (계속)

다음 연습은 기본 SQL SELECT 문, 기본 SQL Developer 명령, SQL 함수, 조인, 그룹 함수 등의 항목을 살펴본 후에 추가로 수행할 수 있습니다.

- 7) 특정 위치에서 근무하는 사원의 부서 이름, 위치 ID, 이름, 직책 및 급여를 표시하는 보고서를 작성합니다. 유저에게 위치를 입력하는 프롬프트를 표시합니다.

- a) 프롬프트가 표시되면 location\_id로 1800을 입력합니다.

```
SELECT d.department_name, d.location_id, e.last_name,
e.job_id, e.salary
FROM   employees e, departments d
WHERE   e.department_id = d.department_id
AND      d.location_id = &location_id;
```

- 8) 성이 "n"으로 끝나는 사원의 수를 알아냅니다. 가능한 두 가지 해결책을 작성합니다.

```
SELECT COUNT(*)
FROM   employees
WHERE  last_name LIKE '%n';
--or
SELECT COUNT(*)
FROM   employees
WHERE  SUBSTR(last_name, -1) = 'n';
```

- 9) 각 부서에 대한 이름, 위치 및 사원 수를 보여주는 보고서를 작성합니다. 보고서에 사원이 없는 부서도 포함되는지 확인합니다.

```
SELECT d.department_id, d.department_name,
       d.location_id, COUNT(e.employee_id)
FROM   employees e RIGHT OUTER JOIN departments d
ON     e.department_id = d.department_id
GROUP BY d.department_id, d.department_name, d.location_id;
```

- 10) HR 부서에서 부서 번호 10 및 20에 있는 직책을 찾으려고 합니다. 해당 부서의 직무 ID를 표시하는 보고서를 작성합니다.

```
SELECT DISTINCT job_id
FROM   employees
WHERE  department_id IN (10, 20);
```

- 11) Administration 및 Executive 부서에서 찾은 직무를 표시하는 보고서를 작성합니다. 또한 해당 직무에 대한 사원 수도 표시합니다. 사원 수가 가장 많은 직무를 가장 먼저 표시합니다.

```
SELECT e.job_id, count(e.job_id) FREQUENCY
FROM      employees e JOIN departments d
ON      e.department_id = d.department_id
WHERE    d.department_name IN ('Administration',
'Executive')
GROUP BY e.job_id
ORDER BY FREQUENCY DESC;
```

## 연습 1-1 해답 (계속)

다음 연습은 기본 SQL SELECT 문, 기본 SQL Developer 명령, SQL 함수, 조인, 그룹 함수, subquery 등의 항목을 살펴본 후에 추가로 수행할 수 있습니다.

- 12) 각 월의 16일 이전에 채용된 사원을 모두 표시합니다.

```
SELECT last_name, hire_date
FROM   employees
WHERE  TO_CHAR(hire_date, 'DD') < 16;
```

- 13) 모든 사원에 대해 성, 급여 및 \$1000 단위로 표현된 급여를 표시하는 보고서를 작성합니다.

```
SELECT last_name, salary, TRUNC(salary, -3)/1000  Thousands
FROM   employees;
```

- 14) 급여가 \$15,000 이상인 관리자 휘하의 모든 사원을 표시합니다. 사원 이름, 관리자 이름, 관리자 급여 및 관리자의 급여 등급을 표시합니다.

```
SELECT e.last_name, m.last_name manager, m.salary,
j.grade_level
FROM   employees e JOIN employees m
ON    e.manager_id = m.employee_id
JOIN   job_grades j
ON    m.salary BETWEEN j.lowest_sal AND j.highest_sal
AND   m.salary > 15000;
```

- 15) 모든 부서의 부서 번호, 이름, 사원 수, 평균 급여와 각 부서에서 일하는 사원의 이름, 급여 및 직무를 표시합니다.

```
SELECT d.department_id, d.department_name,
count(e1.employee_id) employees,
NVL(TO_CHAR(AVG(e1.salary), '99999.99'), 'No
average') avg_sal,
e2.last_name, e2.salary, e2.job_id
FROM   departments d RIGHT OUTER JOIN employees e1
ON    d.department_id = e1.department_id
RIGHT OUTER JOIN employees e2
ON    d.department_id = e2.department_id
GROUP BY d.department_id, d.department_name, e2.last_name,
e2.salary,
e2.job_id
ORDER BY d.department_id, employees;
```

- 16) 평균 급여가 가장 높은 부서의 부서 번호와 최저 급여를 표시하는 보고서를 작성합니다.

```
SELECT department_id, MIN(salary)
FROM   employees
GROUP BY department_id
HAVING AVG(salary) = (SELECT MAX(AVG(salary))
                      FROM   employees
                      GROUP BY department_id);
```

## 연습 1-1 해답 (계속)

- 17) 영업 사원이 근무하지 않는 부서를 표시하는 보고서를 작성합니다. 출력에 부서 번호, 부서 이름 및 위치를 포함합니다.

```
SELECT *
  FROM departments
 WHERE department_id NOT IN(SELECT department_id
                                FROM employees
                               WHERE job_id = 'SA_REP'
                                 AND department_id IS NOT NULL);
```

- 18) HR 부서를 위해 통계 보고서를 작성합니다. 이 보고서에는 다음 조건의 부서에 대한 부서 번호, 부서 이름 및 근무하는 사원 수를 표시합니다.

- a) 사원 수가 3명 미만인 부서:

```
SELECT d.department_id, d.department_name, COUNT(*)
  FROM departments d JOIN employees e
  ON d.department_id = e.department_id
 GROUP BY d.department_id, d.department_name
 HAVING COUNT(*) < 3;
```

- b) 사원 수가 가장 많은 부서:

```
SELECT d.department_id, d.department_name, COUNT(*)
  FROM departments d JOIN employees e
  ON d.department_id = e.department_id
 GROUP BY d.department_id, d.department_name
 HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                      FROM employees
                     GROUP BY department_id);
```

- c) 사원 수가 가장 적은 부서:

```
SELECT d.department_id, d.department_name, COUNT(*)
  FROM departments d JOIN employees e
  ON d.department_id = e.department_id
 GROUP BY d.department_id, d.department_name
 HAVING COUNT(*) = (SELECT MIN(COUNT(*))
                      FROM employees
                     GROUP BY department_id);
```

- 19) 모든 사원에 대해 사원 번호, 성, 급여, 부서 번호 및 해당 부서의 평균 급여를 표시하는 보고서를 작성합니다.

```
SELECT e.employee_id, e.last_name, e.department_id,
e.salary, AVG(s.salary)
  FROM employees e JOIN employees s
  ON e.department_id = s.department_id
 GROUP BY e.employee_id, e.last_name, e.department_id,
e.salary;
```

## 연습 1-1 해답 (계속)

20) 사원 채용 수가 가장 많은 요일에 채용된 사원을 모두 표시합니다.

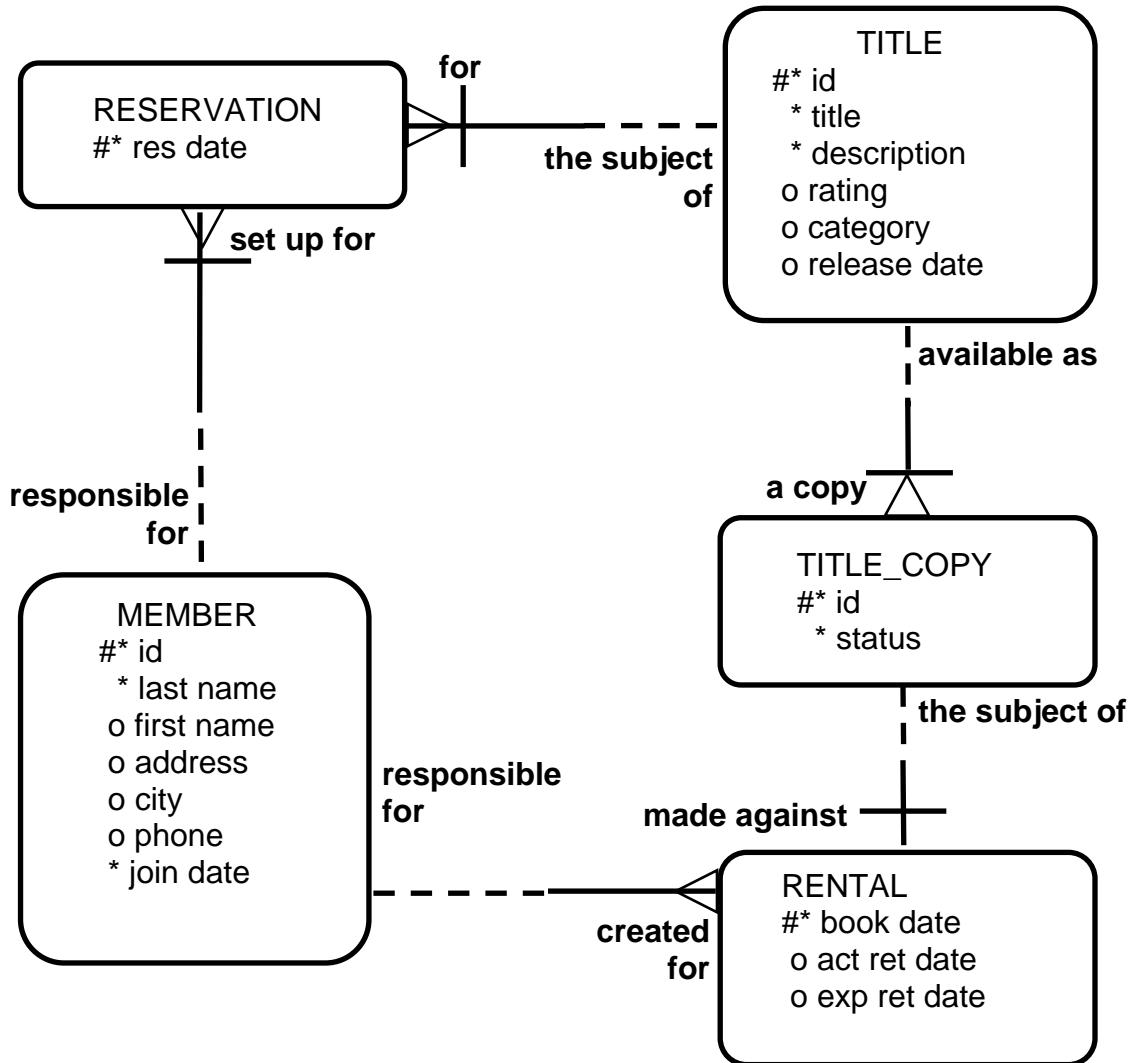
```
SELECT last_name, TO_CHAR(hire_date, 'DAY') day
FROM   employees
WHERE  TO_CHAR(hire_date, 'Day') =
       (SELECT TO_CHAR(hire_date, 'Day')
        FROM   employees
        GROUP BY TO_CHAR(hire_date, 'Day')
        HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                            FROM   employees
                            GROUP BY TO_CHAR(hire_date,
                                             'Day')));
```

21) 사원의 채용일에 준하여 기념일 개요를 생성합니다. 기념일을 오름차순으로 정렬하십시오.

```
SELECT last_name, TO_CHAR(hire_date, 'Month DD') BIRTHDAY
FROM   employees
ORDER BY TO_CHAR(hire_date, 'DDD');
```

## 사례 연구

이 사례 연구에서는 비디오 응용 프로그램에 사용할 일련의 데이터베이스 테이블을 구축합니다. 테이블을 생성한 후 비디오 대여점 데이터베이스의 레코드를 삽입, 갱신 및 삭제하고 보고서를 생성합니다. 데이터베이스에는 필수 테이블만 포함됩니다. 다음은 비디오 응용 프로그램의 엔티티 및 속성을 나타낸 다이어그램입니다.



## 연습 1-1 해답 (계속)

참고: 테이블을 생성하려면 SQL Developer에서 buildtab.sql 스크립트의 명령을 실행합니다. 참고: 테이블을 삭제하려면 SQL Developer에서 dropvid.sql 스크립트의 명령을 실행합니다. 그런 다음 SQL Developer에서 buildvid.sql 스크립트의 명령을 실행하여 테이블을 작성하고 채울 수 있습니다.

세 가지 SQL 스크립트는 모두 /home/oracle/labs/sql1/labs 폴더에 있습니다.

- buildtab.sql 스크립트를 사용하여 테이블을 생성하는 경우 단계 4부터 시작하십시오.
- dropvid.sql 스크립트를 사용하여 비디오 테이블을 제거하는 경우 1단계부터 시작하십시오.
- buildvid.sql 스크립트를 사용하여 테이블을 작성하고 채우는 경우 단계 6(b)부터 시작하십시오

## 연습 2-1

- 1) 다음 테이블 instance 차트에 준하여 테이블을 생성합니다. 해당 데이터 유형을 선택하고 무결성 제약 조건을 추가합니다.

a) 테이블 이름: MEMBER

Column Name	MEMBER_ID	LAST_NAME	FIRST_NAME	ADDRESS	CITY	PHONE	JOIN_DATE
키 유형	PK						
Null / 고유	NN,U	NN					NN
기본 값							시스템 날짜
데이터 유형	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
길이	10	25	25	100	30	15	

b) 테이블 이름: TITLE

Column Name	TITLE_ID	TITLE	DESCRIPTION	RATING	CATEGORY	RELEASE_DATE
키 유형	PK					
Null/ 고유	NN,U	NN	NN			
확인				G, PG, R, NC17, NR	DRAMA, COMEDY, ACTION, CHILD, SCIFI, DOCUMENTARY	
데이터 유형	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
길이	10	60	400	4	20	

## 연습 2-1 (계속)

c) 테이블 이름: TITLE\_COPY

열 이름	COPY_ID	TITLE_ID	STATUS
키 유형	PK	PK,FK	
Null/ 고유	NN,U	NN,U	NN
확인			AVAILABLE, DESTROYED, RENTED, RESERVED
FK Ref 테이블		TITLE	
FK Ref Col		TITLE_ID	
데이터 유형	NUMBER	NUMBER	VARCHAR2
길이	10	10	15

d) 테이블 이름: RENTAL

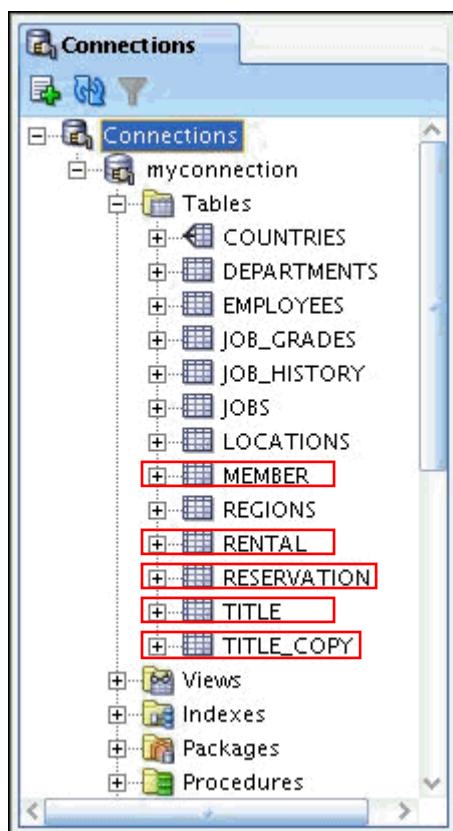
열 이름	BOOK_DATE	MEMBER_ID	COPY_ID	ACT_RET_DATE	EXP_RET_DATE	TITLE_ID
키 유형	PK	PK,FK1	PK,FK2			PK,FK2
기본 값	시스템 날짜				시스템 날짜 + 2일	
FK Ref 테이블		MEMBER	TITLE_COPY			TITLE_COPY
FK Ref Col		MEMBER_ID	COPY_ID			TITLE_ID
데이터 유형	DATE	NUMBER	NUMBER	DATE	DATE	NUMBER
길이		10	10			10

## 연습 2-1 (계속)

e) 테이블 이름: RESERVATION

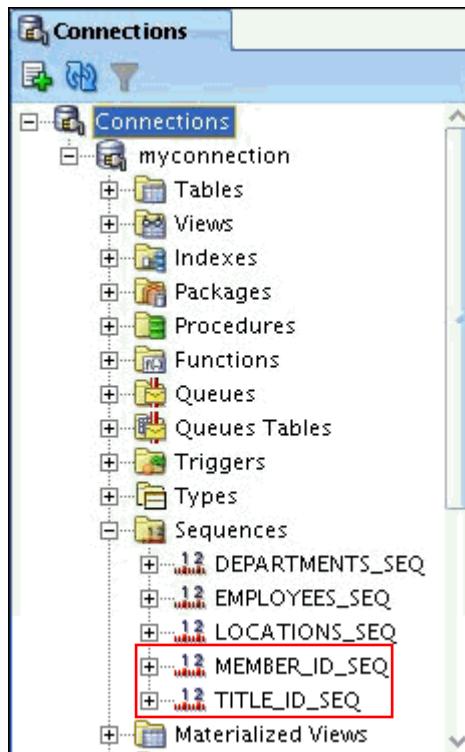
열 이름	RES_DATE	MEMBER_ID	TITLE_ID
키 유형	PK	PK,FK1	PK,FK2
Null/고유	NN,U	NN,U	NN
FK Ref 테이블		MEMBER	TITLE
FK Ref 열		MEMBER_ID	TITLE_ID
데이터 유형	DATE	NUMBER	NUMBER
길이		10	10

- 2) SQL Developer의 Connections Navigator에서 검사를 진행하여 해당 테이블이 제대로 생성되었는지 확인합니다.



## 연습 2-1 (계속)

- 3) MEMBER 테이블 및 TITLE 테이블의 각 행을 고유하게 식별하기 위한 시퀀스를 생성합니다.
- MEMBER 테이블의 멤버 번호: 101부터 시작하고 값을 캐시에 저장할 수 없습니다. 파일 이름을 시퀀스 MEMBER\_ID\_SEQ로 지정하십시오.
  - TITLE 테이블의 제목 번호: 92부터 시작하고 값을 캐시에 저장할 수 없습니다. 파일 이름을 시퀀스 TITLE\_ID\_SEQ로 지정하십시오.
  - SQL Developer의 Connections Navigator에 시퀀스가 존재하는지 확인합니다.



- 4) 테이블에 데이터를 추가합니다. 추가할 각 데이터 집합에 대해 스크립트를 작성합니다.
- TITLE 테이블에 영화 제목을 추가합니다. 영화 정보를 입력하기 위한 스크립트를 작성합니다. lab\_apcs\_4a.sql이라는 스크립트에 해당 명령문을 저장합니다. 시퀀스를 사용하여 각 영화 제목을 고유하게 식별합니다. 출시 날짜를 DD-MON-YYYY 형식으로 입력합니다. 문자 필드의 작은 따옴표는 특수하게 처리해야 한다는 점을 기억하십시오. 추가한 내용을 확인합니다.

	TITLE
1	Willie and Christmas Too
2	Alien Again
3	The Glob
4	My Day Off
5	Miracles on Ice
6	Soda Gang

## 연습 2-1 (계속)

Title	Description	Rating	Category	Release_date
Willie and Christmas Too	All of Willie's friends make a Christmas list for Santa, but Willie has yet to add his own wish list.	G	CHILD	05-OCT-1995
Alien Again	Yet another installation of science fiction history. Can the heroine save the planet from the alien life form?	R	SCIFI	19-MAY-1995
The Glob	A meteor crashes near a small American town and unleashes carnivorous goo in this classic.	NR	SCIFI	12-AUG-1995
My Day Off	With a little luck and a lot of ingenuity, a teenager skips school for a day in New York.	PG	COMEDY	12-JUL-1995
Miracles on Ice	A six-year-old has doubts about Santa Claus, but she discovers that miracles really do exist.	PG	DRAMA	12-SEP-1995
Soda Gang	After discovering a cache of drugs, a young couple find themselves pitted against a vicious gang.	NR	ACTION	01-JUN-1995

- b) MEMBER 테이블에 데이터를 추가합니다. lab\_apcs\_4b.sql이라는 스크립트에 insert 문을 저장합니다. 스크립트의 명령을 실행합니다. 시퀀스를 사용하여 멤버 번호를 추가하십시오.

First_Name	Last_Name	Address	City	Phone	Join_Date
Carmen	Velasquez	283 King Street	Seattle	206-899-6666	08-MAR-1990
LaDoris	Ngao	5 Modrany	Bratislava	586-355-8882	08-MAR-1990
Midori	Nagayama	68 Via Centrale	Sao Paolo	254-852-5764	17-JUN-1991
Mark	Quick-to-See	6921 King Way	Lagos	63-559-7777	07-APR-1990
Audry	Ropeburn	86 Chu Street	Hong Kong	41-559-87	18-JAN-1991
Molly	Urguhart	3035 Laurier	Quebec	418-542-9988	18-JAN-1991

## 연습 2-1 (계속)

c) 다음 영화 제목을 TITLE\_COPY 테이블에 추가합니다.

**참고:** 이 연습에 사용할 수 있는 TITLE\_ID 번호가 있어야 합니다.

Title	Copy_Id	Status	Title	Copy_Id
Willie and Christmas Too	1	Available	Willie and Christmas Too	1
Alien Again	1	Available	Alien Again	1
	2	Rented		2
The Glob	1	Available	The Glob	1
My Day Off	1	Available	My Day Off	1
	2	Available		2
	3	Rented		3
Miracles on Ice	1	Available	Miracles on Ice	1
Soda Gang	1	Available	Soda Gang	1

d) RENTAL 테이블에 다음 대여 비디오를 추가합니다.

**참고:** 제목 번호는 시퀀스에 따라 다를 수 있습니다.

Title_Id	Copy_Id	Member_Id	Book_date	Exp_Ret_Date
92	1	101	3 days ago	1 day ago
93	2	101	1 day ago	1 day from now
95	3	102	2 days ago	Today
97	1	106	4 days ago	2 days ago

## 연습 2-1 (계속)

- 5) 영화 제목, 각 복사본의 대여 가능성 및 대여된 경우 예상 반납 날짜를 표시하는 TITLE\_AVAIL이라는 뷰를 생성합니다. 뷰에서 모든 행을 질의합니다. 제목에 따라 결과를 정렬합니다.

참고: 결과는 다를 수 있습니다.

	TITLE	COPY_ID	STATUS	EXP_RET_DATE
1	Alien Again	1	AVAILABLE	(null)
2	Alien Again	2	RENTED	15-JUL-09
3	Miracles on Ice	1	AVAILABLE	(null)
4	My Day Off	1	AVAILABLE	(null)
5	My Day Off	2	AVAILABLE	(null)
6	My Day Off	3	RENTED	16-JUL-09
7	Soda Gang	1	AVAILABLE	14-JUL-09
8	The Glob	1	AVAILABLE	(null)
9	Willie and Christmas Too	1	AVAILABLE	15-JUL-09

- 6) 테이블의 데이터를 변경합니다.

- a) 새 제목을 추가합니다. 영화는 공상 과학 영화로 분류된 PG 등급의 "Interstellar Wars"입니다. 출시 날짜는 07-JUL-77입니다. 설명은 "Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?"입니다. 두 사본에 대해 영화 사본 기록을 추가해야 합니다.
- b) 두 건의 예약을 입력합니다. 하나는 "Interstellar Wars"를 대여하려는 Carmen Velasquez 고객이고 다른 하나는 "Soda Gang"을 대여하려는 Mark Quick-to-See 고객입니다.

- 7) 테이블 중 하나를 수정합니다.

- a) /home/oracle/labs/sql1/labs 폴더에 있는 lab\_apcs\_7a.sql 스크립트를 실행하여 TITLE 테이블에 비디오 구입 가격을 기록하는 PRICE 열을 추가합니다. 수정 사항을 확인합니다.

DESCRIBE title		
Name	Null	Type
TITLE_ID	NOT NULL	NUMBER(10)
TITLE	NOT NULL	VARCHAR2(60)
DESCRIPTION	NOT NULL	VARCHAR2(400)
RATING		VARCHAR2(4)
CATEGORY		VARCHAR2(20)
RELEASE_DATE		DATE
PRICE		NUMBER(8,2)

Title	Price
Willie and Christmas Too	25
Alien Again	35
The Glob	35
My Day Off	35
Miracles on Ice	30
Soda Gang	35
Interstellar Wars	29

## 연습 2-1 (계속)

- b) 앞의 리스트에 따른 가격으로 각 비디오를 갱신하는 update 문이 포함된 lab\_apcs\_7b.sql이라는 스크립트를 작성합니다. 스크립트에서 명령을 실행합니다.

**참고:** 이 연습에 사용할 수 있는 TITLE\_ID 번호가 있어야 합니다.

- 8) 각 고객의 비디오 대여 내역을 포함하는 보고서를 작성합니다. 고객 이름, 대여한 영화, 대여 날짜 및 대여 기간을 포함해야 합니다. Reporting 기간 동안 모든 고객이 대여한 비디오 수를 합산합니다. lab\_apcs\_8.sql이라는 스크립트 파일에 보고서를 생성하는 명령을 저장합니다.

**참고:** 결과는 다를 수 있습니다.

MEMBER	TITLE	BOOK_DATE	DURATION
1 Carmen Velasquez	Willie and Christmas Too	13-JUL-09	1
2 Carmen Velasquez	Alien Again	15-JUL-09	(null)
3 LaDoris Ngao	My Day Off	14-JUL-09	(null)
4 Molly Urguhart	Soda Gang	12-JUL-09	2

## 연습 2-1 해답

- 1) 다음 테이블 instance 차트에 준하여 테이블을 생성합니다. 해당 데이터 유형을 선택하고 무결성 제약 조건을 추가합니다.

a) 테이블 이름: MEMBER

```
CREATE TABLE member
  (member_id          NUMBER(10)
   CONSTRAINT member_member_id_pk PRIMARY KEY,
   last_name          VARCHAR2(25)
   CONSTRAINT member_last_name_nn NOT NULL,
   first_name         VARCHAR2(25),
   address            VARCHAR2(100),
   city               VARCHAR2(30),
   phone              VARCHAR2(15),
   join_date          DATE DEFAULT SYSDATE
   CONSTRAINT member_join_date_nn NOT NULL);
```

b) 테이블 이름: TITLE

```
CREATE TABLE title
  (title_id          NUMBER(10)
   CONSTRAINT title_title_id_pk PRIMARY KEY,
   title              VARCHAR2(60)
   CONSTRAINT title_title_nn NOT NULL,
   description        VARCHAR2(400)
   CONSTRAINT title_description_nn NOT NULL,
   rating             VARCHAR2(4)
   CONSTRAINT title_rating_ck CHECK
     (rating IN ('G', 'PG', 'R', 'NC17', 'NR')),
   category           VARCHAR2(20)
   CONSTRAINT title_category_ck CHECK
     (category IN ('DRAMA', 'COMEDY', 'ACTION',
                   'CHILD', 'SCIFI', 'DOCUMENTARY')),
   release_date       DATE);
```

c) 테이블 이름: TITLE\_COPY

```
CREATE TABLE title_copy
  (copy_id            NUMBER(10),
   title_id           NUMBER(10)
   CONSTRAINT title_copy_title_if_fk REFERENCES
   title(title_id),
   status              VARCHAR2(15)
   CONSTRAINT title_copy_status_nn NOT NULL
   CONSTRAINT title_copy_status_ck CHECK (status IN
     ('AVAILABLE', 'DESTROYED', 'RENTED', 'RESERVED')),
   CONSTRAINT title_copy_copy_id_title_id_pk
   PRIMARY KEY (copy_id, title_id));
```

## 연습 2-1 해답(계속)

d) 테이블 이름: RENTAL

```
CREATE TABLE rental
  (book_date      DATE DEFAULT SYSDATE,
   member_id      NUMBER(10)
    CONSTRAINT rental_member_id_fk REFERENCES
member(member_id),
   copy_id        NUMBER(10),
   act_ret_date   DATE,
   exp_ret_date  DATE DEFAULT SYSDATE + 2,
   title_id      NUMBER(10),
    CONSTRAINT rental_book_date_copy_title_pk
      PRIMARY KEY (book_date, member_id,
copy_id,title_id),
    CONSTRAINT rental_copy_id_title_id_fk
      FOREIGN KEY (copy_id, title_id)
      REFERENCES title_copy(copy_id, title_id));
```

e) 테이블 이름: RESERVATION

```
CREATE TABLE reservation
  (res_date       DATE,
   member_id      NUMBER(10)
    CONSTRAINT reservation_member_id REFERENCES
member(member_id),
   title_id       NUMBER(10)
    CONSTRAINT reservation_title_id REFERENCES
title(title_id),
    CONSTRAINT reservation_resdate_mem_tit_pk PRIMARY KEY
  (res_date, member_id, title_id));
```

- 2) SQL Developer의 Connections Navigator에서 검사를 진행하여 해당 테이블이 제대로 생성되었는지 확인합니다.
  - a) Connections Navigator에서 Connections > myconnection > Tables를 확장합니다.
- 3) MEMBER 테이블 및 TITLE 테이블의 각 행을 고유하게 식별하기 위한 시퀀스를 생성합니다.
  - a) MEMBER 테이블의 멤버 번호: 101부터 시작하고 값을 캐시에 저장할 수 없습니다. 파일 이름을 시퀀스 MEMBER\_ID\_SEQ로 지정하십시오.

```
CREATE SEQUENCE member_id_seq
START WITH 101
NOCACHE;
```

- b) TITLE 테이블의 제목 번호: 92부터 시작하고 값을 캐시에 저장할 수 없습니다. 파일 이름을 시퀀스 TITLE\_ID\_SEQ로 지정하십시오.

```
CREATE SEQUENCE title_id_seq
START WITH 92
NOCACHE;
```

## 연습 2-1 해답(계속)

- c) SQL Developer의 Connections Navigator에 시퀀스가 존재하는지 확인합니다.
- i) Connections Navigator에서 myconnection 노드가 확장되어 있으면 Sequences를 확장합니다.
- 4) 테이블에 데이터를 추가합니다. 추가할 각 데이터 집합에 대해 스크립트를 작성합니다.
- a) TITLE 테이블에 영화 제목을 추가합니다. 영화 정보를 입력하기 위한 스크립트를 작성합니다. lab\_apcs\_4a.sql이라는 스크립트에 해당 명령문을 저장합니다. 시퀀스를 사용하여 각 영화 제목을 고유하게 식별합니다. 출시 날짜를 DD-MON-YYYY 형식으로 입력합니다. 문자 필드의 작은 따옴표는 특수하게 처리해야 한다는 점을 기억하십시오. 추가한 내용을 확인합니다.

```

INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES  (title_id_seq.NEXTVAL, 'Willie and Christmas Too',
          'All of Willie''s friends make a Christmas list for
          Santa, but Willie has yet to add his own wish list.',
          'G', 'CHILD', TO_DATE('05-OCT-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id , title, description, rating,
                  category, release_date)
VALUES  (title_id_seq.NEXTVAL, 'Alien Again', 'Yet another
          installment of science fiction history. Can the
          heroine save the planet from the alien life form?',
          'R', 'SCIFI', TO_DATE( '19-MAY-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES  (title_id_seq.NEXTVAL, 'The Glob', 'A meteor crashes
          near a small American town and unleashes carnivorous
          goo in this classic.', 'NR', 'SCIFI',
          TO_DATE( '12-AUG-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES  (title_id_seq.NEXTVAL, 'My Day Off', 'With a little
          luck and a lot ingenuity, a teenager skips school
          for
          a day in New York.', 'PG', 'COMEDY',
          TO_DATE( '12-JUL-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES  (title_id_seq.NEXTVAL, 'Miracles on Ice', 'A six-
          year-old has      doubts about Santa Claus, but she
          discovers that miracles really do exist.', 'PG', 'DRAMA',
          TO_DATE('12-SEP-1995','DD-MON-YYYY'))
/

```

## 연습 2-1 해답(계속)

```

INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES      (title_id_seq.NEXTVAL, 'Soda Gang', 'After
discovering a cache of drugs, a young couple find themselves
pitted against a vicious gang.', 'NR', 'ACTION',
TO_DATE('01-JUN-1995', 'DD-MON-YYYY'))
/
COMMIT
/
SELECT  title
FROM    title;

```

- b) MEMBER 테이블에 데이터를 추가합니다. lab\_apcs\_4b.sql이라는 스크립트에 insert 문을 넣습니다. 스크립트의 명령을 실행합니다. 시퀀스를 사용하여 멤버 번호를 추가하십시오.

```

SET VERIFY OFF
INSERT INTO member(member_id, first_name, last_name,
                   address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Carmen', 'Velasquez',
        '283 King Street', 'Seattle', '206-899-6666',
        TO_DATE('08-MAR-1990',
                'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
                   address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'LaDoris', 'Ngao',
        '5 Modrany', 'Bratislava', '586-355-8882',
        TO_DATE('08-MAR-1990',
                'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
                   address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Midori', 'Nagayama',
        '68 Via Centrale', 'Sao Paolo', '254-852-5764',
        TO_DATE('17-JUN-1991',
                'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
                   address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Mark', 'Quick-to-See',
        '6921 King Way', 'Lagos', '63-559-7777',
        TO_DATE('07-APR-1990',
                'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
                   address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Audry', 'Ropeburn',
        '86 Chu Street', 'Hong Kong', '41-559-87',
        TO_DATE('18-JAN-1991',
                'DD-MM-YYYY'))

```

## 연습 2-1 해답(계속)

```
'DD-MM-YYYY' ) )
/
INSERT INTO member(member_id, first_name, last_name,
                   address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Molly', 'Urguhart',
        '3035 Laurier', 'Quebec', '418-542-9988',
        TO_DATE('18-JAN-1991',
                'DD-MM-YYYY') );
/
COMMIT
SET VERIFY ON
```

- c) 다음 영화 제목을 TITLE\_COPY 테이블에 추가합니다.

**참고:** 이 연습에 사용할 수 있는 TITLE\_ID 번호가 있어야 합니다.

```
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 92, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 93, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (2, 93, 'RENTED')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 94, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 95, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (2, 95, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (3, 95, 'RENTED')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 96, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 97, 'AVAILABLE')
/
```

## 연습 2-1 해답(계속)

- d) RENTAL 테이블에 다음 대여 비디오를 추가합니다.

**참고:** 제목 번호는 시퀀스에 따라 다를 수 있습니다.

```
INSERT INTO rental(title_id, copy_id, member_id,
                   book_date, exp_ret_date, act_ret_date)
VALUES (92, 1, 101, sysdate-3, sysdate-1, sysdate-2)
/
INSERT INTO rental(title_id, copy_id, member_id,
                   book_date, exp_ret_date, act_ret_date)
VALUES (93, 2, 101, sysdate-1, sysdate-1, NULL)
/
INSERT INTO rental(title_id, copy_id, member_id,
                   book_date, exp_ret_date, act_ret_date)
VALUES (95, 3, 102, sysdate-2, sysdate, NULL)
/
INSERT INTO rental(title_id, copy_id, member_id,
                   book_date, exp_ret_date, act_ret_date)
VALUES (97, 1, 106, sysdate-4, sysdate-2, sysdate-2)
/
COMMIT
/
```

- 5) 영화 제목, 각 복사본의 대여 가능성 및 대여된 경우 예상 반납 날짜를 표시하는 TITLE\_AVAIL이라는 뷰를 생성합니다. 뷰에서 모든 행을 질의합니다. 제목에 따라 결과를 정렬합니다.

**참고:** 결과는 다를 수 있습니다.

```
CREATE VIEW title_avail AS
  SELECT t.title, c.copy_id, c.status, r.exp_ret_date
    FROM title t JOIN title_copy c
      ON t.title_id = c.title_id
     FULL OUTER JOIN rental r
       ON c.copy_id = r.copy_id
      AND c.title_id = r.title_id;

  SELECT *
    FROM title_avail
   ORDER BY title, copy_id;
```

## 연습 2-1 해답(계속)

6) 테이블의 데이터를 변경합니다.

- a) 새 제목을 추가합니다. 영화는 공상 과학 영화로 분류된 PG 등급의 "Interstellar Wars"입니다. 출시 날짜는 07-JUL-77입니다. 설명은 "Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?"입니다. 두 사본에 대해 영화 사본 기록을 추가해야 합니다.

```
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Interstellar Wars',
        'Futuristic interstellar action movie. Can the
        rebels save the humans from the evil empire?',
        'PG', 'SCIFI', '07-JUL-77')
/
INSERT INTO title_copy (copy_id, title_id, status)
VALUES (1, 98, 'AVAILABLE')
/
INSERT INTO title_copy (copy_id, title_id, status)
VALUES (2, 98, 'AVAILABLE')
```

- b) 두 건의 예약을 입력합니다. 하나는 "Interstellar Wars"를 대여하려는 Carmen Velasquez 고객이고 다른 하나는 "Soda Gang"을 대여하려는 Mark Quick-to-See 고객입니다.

```
INSERT INTO reservation (res_date, member_id, title_id)
VALUES (SYSDATE, 101, 98)
/
INSERT INTO reservation (res_date, member_id, title_id)
VALUES (SYSDATE, 104, 97)
```

7) 테이블 중 하나를 수정합니다.

- a) /home/oracle/labs/sql1/labs 폴더에 있는 lab\_apcs\_7a.sql 스크립트를 실행하여 TITLE 테이블에 비디오 구입 가격을 기록하는 PRICE 열을 추가합니다. 수정 사항을 확인합니다.

```
ALTER TABLE title
ADD (price NUMBER(8,2));
DESCRIBE title
```

## 연습 2-1 해답(계속)

- b) 제공된 리스트에 따른 가격으로 각 비디오를갱신하는 update 문이 포함된 lab\_apcs\_7b.sql이라는 스크립트를 작성합니다. 스크립트에서 명령을 실행합니다.

**참고:** 이 연습에 사용할 수 있는 TITLE\_ID 번호가 있어야 합니다.

```
SET ECHO OFF
SET VERIFY OFF
UPDATE title
SET    price = &price
WHERE  title_id = &title_id;
SET VERIFY OFF
SET ECHO OFF
```

- 8) 각 고객의 비디오 대여 내역을 포함하는 보고서를 작성합니다. 고객 이름, 대여한 영화, 대여 날짜 및 대여 기간을 포함해야 합니다. Reporting 기간 동안 모든 고객이 대여한 비디오 수를 합산합니다. lab\_apcs\_8.sql이라는 스크립트 파일에 보고서를 생성하는 명령을 저장합니다.

**참고:** 결과는 다를 수 있습니다.

```
SELECT m.first_name||' '||m.last_name MEMBER, t.title,
       r.book_date, r.act_ret_date - r.book_date DURATION
  FROM member m
JOIN rental r
  ON r.member_id = m.member_id
JOIN title t
  ON r.title_id = t.title_id
 ORDER BY member;
```

---

**색인**

---

## ㄱ

객체 관계형 I-2, I-16, I-17

고유 식별자 I-23, I-24

관계형 데이터베이스 9-43, I-2, I-3, I-4, I-8, I-15, I-16, I-18, I-19,  
I-27, I-28, I-29, I-30, I-31, I-33, I-36, I-39

그룹 함수 3-5, 5-1, 5-2, 5-3, 5-4, 5-5, 5-6, 5-11, 5-12, 5-13, 5-14,  
5-15, 5-19, 5-20, 5-22, 5-25, 5-26, 5-27, 5-28, 5-29,  
7-3, 7-9, 7-12, 7-16, 7-21, 7-25, I-5

## ㄴ

날짜 1-9, 1-11, 1-20, 1-21, 2-7, 2-8, 2-11, 2-12, 2-23, 2-24,  
2-25, 2-28, 2-31, 3-2, 3-3, 3-4, 3-5, 3-7, 3-8, 3-15, 3-17,  
3-18, 3-20, 3-21, 3-22, 3-23, 3-24, 3-25, 3-26, 3-27, 3-28, 3-29,  
3-30, 3-32, 3-33, 4-2, 4-3, 4-5, 4-6, 4-7, 4-8, 4-9, 4-10,  
4-11, 4-12, 4-13, 4-14, 4-15, 4-16, 4-20, 4-21, 4-22, 4-23, 4-25,  
4-27, 4-29, 4-36, 4-43, 4-44, 4-45, 5-6, 5-8, 5-28, 8-8, 8-10,  
9-2, 9-3, 9-4, 9-7, 9-9, 9-10, 9-13, 9-14, 9-15, 9-16, 9-17,  
9-18, 9-19, 9-25, 9-29, 9-30, 9-35, 9-39, 9-40, 9-41, 9-42, 9-43,  
9-44, 9-45, 9-47, 9-48, 10-9, 10-10, 10-12, 10-14, 10-16, 10-20, 10-22,  
10-25, 10-26, 10-27, 10-28, 10-29, 10-33, 10-36, 10-38, 10-39, B-3,  
C-10, D-17, E-14, I-4, I-9, I-16, I-17, I-18, I-22, I-31, I-34

## ㄷ

대체 인용(q) 연산자 1-23

데이터 유형 1-11, 1-26, 1-27, 3-4, 3-33, 4-4, 4-29, 4-31, 4-44,  
5-6, 5-8, 6-9, 6-11, 8-13, 8-19, 8-22, 9-8, 9-12, 10-2,  
10-3, 10-6, 10-11, 10-12, 10-13, 10-14, 10-15, 10-31,  
10-34, 10-37, 10-40, C-11, D-7, E-14, I-9, I-16

**=(계속)**

데이터베이스 1-2, 1-4, 1-14, 1-15, 1-29, 2-2, 2-7, 3-4, 3-5, 3-10,  
 3-12, 3-16, 3-21, 3-24, 3-25, 3-26, 4-4, 4-9, 4-28, 5-15,  
 5-18, 5-28, 6-2, 6-6, 6-36, 7-8, 9-3, 9-4, 9-9, 9-13, 9-15,  
 9-19, 9-21, 9-25, 9-26, 9-27, 9-31, 9-33, 9-34, 9-39, 9-40, 9-41,  
 9-42, 9-43, 9-44, 9-45, 9-47, 10-2, 10-3, 10-4, 10-5, 10-6, 10-7,  
 10-8, 10-10, 10-11, 10-14, 10-15, 10-17, 10-31, 10-33, 10-34, 10-36,  
 10-37, 10-38, 10-40, 10-41, B-2, C-2, C-3, C-4, C-5, C-6, C-7, C-8,  
 C-9, C-10, C-12, C-13, C-15, C-16, C-17, C-19, C-26, C-29, C-31,  
 D-3, D-4, D-5, D-6, D-15, D-17, D-18, E-2, E-4, E-5, E-6, E-8,  
 E-11, E-15, E-18, F-2, F-7, F-10, F-22, I-2, I-3, I-4, I-8, I-9,  
 I-10, I-11, I-12, I-13, I-14, I-15, I-16, I-17, I-18, I-19, I-20, I-22,  
 I-27, I-28, I-29, I-30, I-31, I-32, I-33, I-34, I-36, I-37, I-38, I-39, I-40

데이터베이스 연결 생성 C-7, C-8, C-9

데이터베이스 트랜잭션 9-3, 9-13, 9-19, 9-25, 9-26, 9-27, 9-39, 9-42

동의어 1-24, 1-26, 7-18, 10-4, 10-8, 10-38, D-7, I-23, I-24

**■**

리터럴 1-3, 1-10, 1-14, 1-16, 1-19, 1-21, 1-22, 1-23, 1-25, 2-5,  
 2-12, 2-13, 3-13, 4-11, 4-14, 4-31, 4-32, 4-38, 8-26, 10-9

**▣**

명시적 데이터 유형 변환 4-3, 4-4, 4-7, 4-8, 4-9, 4-10, 4-23, 4-27, 4-36

문자열 1-3, 1-10, 1-16, 1-19, 1-20, 1-21, 1-22, 1-25,  
 2-7, 2-16, 3-11, 3-13, 4-14, 4-17, 4-43

**▣**

변환 함수 3-7, 3-9, 3-11, 3-12, 3-28, 4-1, 4-2, 4-4, 4-9, 4-44, 8-25, I-5

뷰 10-8, 10-10, 10-38, C-10, E-6, E-13

비교 연산자 2-8, 2-9, 7-10, 7-17, 9-15, 9-21

**^**

산술 연산자 1-11, 1-12, 2-20, 3-25, 3-26, 3-32  
 산술식 1-3, 1-10, 1-11, 1-15, 1-16, 1-19, 1-20, 1-25, 2-5  
 선택 1-4, 2-4, E-12  
 속성 I-22, I-23, I-24  
 순서 1-7, 1-13, 2-2, 2-3, 2-11, 2-19, 2-20, 2-22, 2-23, 2-24,  
     2-25, 2-26, 2-28, 2-32, 2-33, 2-34, 2-38, 2-39, 3-6, 4-25, 5-6,  
     5-14, 5-15, 5-16, 5-17, 5-18, 5-22, 5-24, 5-27, 5-28, 8-2, 8-3,  
     8-4, 8-5, 8-6, 8-7, 8-12, 8-13, 8-15, 8-17, 8-18, 8-19, 8-21,  
     8-24, 8-27, 8-28, 8-29, 8-30, 9-7, 9-8, 9-43, 9-44, 9-45, 10-13,  
     B-2, C-26, I-14, I-22, I-27  
 숫자 함수 3-3, 3-7, 3-8, 3-15, 3-16, 3-20, 3-27, 3-28, 4-3, 4-10, 4-23, 4-27, 4-36  
 스키마 10-2, 10-5, 10-7, 10-8, 10-18, 10-28, 10-40, B-2, C-3,  
     C-5, C-7, C-8, C-10, C-13, E-5, E-6, I-2, I-3, I-4, I-6,  
     I-8, I-15, I-29, I-33, I-34, I-36, I-40  
 시퀀스 2-24, 10-8

**o**

암시적 데이터 유형 변환 4-4, 4-5, 4-6  
 앤퍼샌드 치환 2-2, 2-29, 2-30, 2-33, 2-38, 2-39, 3-13  
 엔티티 관계 B-3, I-21, I-22, I-23  
 연결 연산자 1-3, 1-10, 1-16, 1-19, 1-20, 1-25, 2-20  
 열 Alias 1-3, 1-9, 1-10, 1-16, 1-17, 1-18, 1-19, 1-21, 1-25,  
     1-29, 2-6, 2-24, 4-25, 5-14, 5-27, 6-7, 10-33, F-8  
 우선 순위 규칙 1-12, 1-13, 2-3, 2-19, 2-20, 2-21, 2-22, 2-26, 2-34  
 이름 지정 10-3, 10-5, 10-6, 10-11, 10-15, 10-17, 10-31, 10-34, 10-37  
 인덱스 10-4, 10-8, 10-10, 10-22, 10-23, 10-36, 10-38, C-4, C-10, E-6, I-38  
 읽기 일관성 9-3, 9-13, 9-19, 9-25, 9-33, 9-39, 9-40, 9-41, 9-42  
 읽기 전용 테이블 10-3, 10-6, 10-11, 10-15, 10-31, 10-34, 10-36, 10-37

**❖**

정렬 2-1, 2-3, 2-19, 2-22, 2-24, 2-25, 2-26, 2-34, 2-38, 2-39, I-5

제약 조건 1-14, 9-4, 9-8, 9-20, 9-24, 10-2, 10-3, 10-6, 10-11, 10-13, 10-15, 10-16, 10-17, 10-18, 10-19, 10-20, 10-21, 10-22, 10-25, 10-27, 10-29, 10-30, 10-31, 10-32, 10-34, 10-37, 10-39, 10-40, C-10, E-6, I-16, I-39

중복 행 1-24, 5-9, 8-6, 8-13, 8-16, 8-17, 8-30, I-27

집합 연산자 8-1, 8-2, 8-3, 8-4, 8-5, 8-6, 8-7, 8-12, 8-18, 8-21, 8-24, 8-27, 8-31, I-5

**❖**

치환 변수 2-3, 2-19, 2-22, 2-26, 2-27, 2-28, 2-31, 2-32, 2-34, 2-36, 2-38, 2-39, 9-11, C-16

**☞**

키워드 1-5, 1-8, 2-23, 6-9, 10-25, 10-26, C-22, D-4

**≡**

테이블 조인 6-6, 6-27, 6-37, F-7, F-23

트랜잭션 9-2, 9-3, 9-4, 9-13, 9-19, 9-25, 9-26, 9-27, 9-29, 9-30, 9-31, 9-32, 9-33, 9-34, 9-38, 9-39, 9-41, 9-42, 9-48, 10-38, C-16, I-11, I-31

**▣**

프로젝션 1-4

**◁**

함수 2-5, 2-7, 3-1, 3-2, 3-3, 3-4, 3-5, 3-6, 3-7, 3-8, 3-9, 3-10, 3-11, 3-12, 3-13, 3-14, 3-15, 3-16, 3-17, 3-18, 3-20, 3-24, 3-27, 3-28, 3-29, 3-30, 3-31, 3-32, 3-33, 4-1, 4-2, 4-3, 4-4, 4-7, 4-9, 4-10, 4-20, 4-21, 4-23, 4-24, 4-25, 4-26, 4-27, 4-28, 4-36, 4-44, 4-45, 5-1, 5-2, 5-3, 5-4, 5-5, 5-6, 5-7, 5-8, 5-11, 5-12, 5-13, 5-14, 5-15, 5-19, 5-20, 5-22, 5-25, 5-26, 5-27, 5-28, 5-29, 7-3, 7-9, 7-12, 7-16, 7-21, 7-25, 8-25, 9-9, 10-9, 10-27, C-5, C-6, C-23, C-24, C-25, E-8, E-15, I-4, I-5

형식 모델 3-28, 3-30, 4-11, 4-12, 4-14, 4-15, 4-16, 4-19, 4-20, 4-43

**A**

Alias 1-3, 1-5, 1-9, 1-10, 1-16, 1-17, 1-18, 1-19, 1-20, 1-21,  
1-25, 1-29, 2-5, 2-6, 2-8, 2-23, 2-24, 2-38, 4-25, 5-14, 5-27,  
6-7, 6-14, 6-16, 6-24, 6-36, 8-28, 10-33, C-8, F-8, F-10, F-11,  
F-15, F-21, F-22

ALL 연산자 7-19, 8-3, 8-6, 8-7, 8-12, 8-16, 8-17, 8-18, 8-21,  
8-24, 8-27, 8-29, 8-30

ALTER TABLE 문 10-35, 10-36

AND 연산자 2-16, 2-21, F-12

ANSI(American National Standards Institute) 4-4, 6-5, I-30

ANY 연산자 7-3, 7-9, 7-18, 7-21

AVG 5-3, 5-5, 5-7, 5-8, 5-11, 5-12, 5-15, 5-16, 5-20, 5-23,  
5-25, 5-26, 5-28, 7-13

**B**

BETWEEN 연산자 2-10

BI Publisher I-14

**C**

Cartesian Product 6-2, 6-3, 6-6, 6-8, 6-19, 6-22, 6-25, 6-31, 6-32, 6-33,  
6-34, 6-35, 6-36, F-2, F-4, F-5, F-22

CASE 표현식 4-32, 4-37, 4-38, 4-39, 4-40, 4-44

CHECK 제약 조건 9-8, 10-3, 10-6, 10-11, 10-15, 10-27, 10-31, 10-34, 10-37

COALESCE 함수 4-33, 4-34, 4-35

COUNT 함수 5-9

CREATE TABLE 문 9-12, 10-3, 10-6, 10-7, 10-11, 10-15, 10-31,  
10-32, 10-34, 10-37, 10-40, 10-41

Cross Join 6-5, 6-34, 6-36

CURRENT\_DATE 3-24, 9-9

CURRVAL 10-9, 10-27

**D**

Datetime 데이터 유형 10-14

DBMS 9-43, D-15, D-17, I-2, I-17, I-18, I-25, I-27, I-39

DECODE 함수 4-37, 4-39, 4-40, 4-41, 4-42, 4-44, 5-6

DEFAULT 옵션 10-3, 10-6, 10-9, 10-11, 10-15, 10-31, 10-34, 10-37

DELETE 문 9-3, 9-13, 9-19, 9-21, 9-22, 9-23, 9-24, 9-25,  
9-39, 9-40, 9-42

DESCRIBE 명령 1-3, 1-10, 1-16, 1-19, 1-25, 1-26, 1-27, 9-8, 10-10, 10-33,  
C-11, D-7

DISTINCT 키워드 1-3, 1-10, 1-16, 1-19, 1-24, 1-25, 5-3, 5-10, 5-12, 5-25

DUAL 테이블 3-17

**E**

Equijoin 6-2, 6-3, 6-8, 6-12, 6-19, 6-22, 6-23, 6-24, 6-25, 6-35, 6-36,  
F-2, F-9, F-10, F-11, F-14, F-15, F-22

Execute Statement 아이콘 1-8, 6-16, 10-41, F-10

**F**

FOR UPDATE 절 9-3, 9-13, 9-19, 9-25, 9-39, 9-42, 9-43, 9-44, 9-45, 9-47

**G**

GROUP BY 절 5-2, 5-3, 5-12, 5-13, 5-14, 5-15, 5-16, 5-18,  
5-19, 5-22, 5-23, 5-25, 5-26, 5-27, 5-28, 7-14

**H**

HAVING 절 5-2, 5-3, 5-12, 5-20, 5-21, 5-22, 5-23, 5-24, 5-25,  
5-28, 5-29, 7-3, 7-5, 7-9, 7-13, 7-16, 7-21, 7-25

**I**

IN 연산자 2-11, 7-22, F-17, F-18

INSERT 문 9-3, 9-6, 9-8, 9-12, 9-13, 9-19, 9-25, 9-39, 9-42, 10-10, 10-18

INTERSECT 연산자 8-3, 8-4, 8-5, 8-7, 8-12, 8-18, 8-19, 8-20, 8-21, 8-24,  
8-27, 8-30, 8-31

INTERVAL YEAR TO MONTH 10-14

ISO(International Standards Organization) I-31

**J**

Java C-4, C-8, E-3, E-12, E-14, E-15, E-16, I-9, I-12, I-39

**L**

LENGTH 3-9, 3-10, 3-13, 3-14, 3-32, 4-32, 6-7, 8-6, 10-7, 10-12, 10-18, D-15, F-8

LIKE 연산자 2-12

LPAD 3-9, 3-13

**M**

MAX 5-3, 5-4, 5-5, 5-7, 5-8, 5-12, 5-21, 5-23, 5-25, 5-26, 5-28, 7-18, 7-19,  
B-2, B-3, I-10, I-34

MIN 1-8, 1-24, 1-30, 2-7, 2-20, 2-27, 3-13, 3-19, 3-21, 3-23,  
3-33, 4-31, 4-35, 4-42, 5-3, 5-5, 5-7, 5-8, 5-12, 5-14, 5-25,  
5-28, 6-7, 6-12, 6-37, 7-4, 7-6, 7-12, 7-13, 7-14, 7-17, 7-18,  
7-19, 7-24, 8-3, 8-4, 8-5, 8-6, 8-7, 8-12, 8-13, 8-14, 8-16,  
8-17, 8-18, 8-21, 8-22, 8-23, 8-24, 8-27, 8-30, 8-31, 9-9, 9-16,  
9-20, 9-31, 9-38, 9-43, 10-3, 10-5, 10-6, 10-7, 10-11, 10-14, 10-15,  
10-17, 10-27, 10-31, 10-34, 10-37, B-2, B-3, C-7, C-8, C-23, C-25,  
C-26, C-30, D-3, D-6, D-9, D-17, F-7, F-8, F-9, F-13, F-23, I-3,  
I-4, I-8, I-10, I-12, I-13, I-15, I-16, I-20, I-27, I-28, I-29, I-33, I-34,  
I-36, I-39, I-40

MINUS 연산자 8-3, 8-5, 8-7, 8-12, 8-18, 8-21, 8-22, 8-23, 8-24, 8-27,  
8-30, 8-31

MOD 함수 3-19

**N**

NEXTVAL 10-9, 10-27

Nonequijoin 6-2, 6-3, 6-8, 6-19, 6-22, 6-23, 6-24,  
6-25, 6-35, 6-36, F-2, F-14, F-15, F-22

NOT 연산자 2-3, 2-18, 2-19, 2-22, 2-26, 2-34, 2-38, 7-19

NOT NULL 제약 조건 1-26, 10-18, 10-20, 10-21, 10-32

Null 값 1-3, 1-10, 1-14, 1-15, 1-16, 1-19, 1-20, 1-25, 2-23, 2-24,  
4-28, 4-30, 5-3, 5-6, 5-9, 5-10, 5-11, 5-12, 5-25, 7-3, 7-9,  
7-15, 7-16, 7-21, 7-22, 7-23, 8-13, 8-19, 9-8, 10-9, 10-20, E-10  
Null 값 1-3, 1-10, 1-14, 1-15, 1-16, 1-19, 1-20, 1-25, 2-8, 2-14, 2-23,  
2-24, 4-28, 4-29, 4-30, 4-31, 4-32, 4-40, 5-3, 5-6, 5-9, 5-10,  
5-11, 5-12, 5-25, 7-3, 7-9, 7-15, 7-16, 7-21, 7-22, 7-23, 8-13,  
8-19, 9-8, 10-9, 10-20, 10-23, E-10, I-28

NULL 조건 2-3, 2-14

NULLIF 함수 4-32

NVL 함수 4-29, 4-30, 4-33, 4-44, 5-11

NVL2 함수 4-31

**O**

OLTP I-11, I-16

ON 절 6-3, 6-5, 6-6, 6-8, 6-15, 6-16, 6-17, 6-18, 6-19, 6-21, 6-22,  
6-25, 6-27, 6-31, 8-17

ON DELETE CASCADE 10-26

ON DELETE SET NULL 10-26

OR 연산자 2-3, 2-15, 2-17, 2-19, 2-22, 2-26, 2-34, F-18

Oracle 서버 1-12, 2-11, 2-13, 2-23, 4-4, 4-5, 4-6, 4-19, 4-20, 4-38,  
5-22, 5-28, 6-7, 6-24, 7-13, 7-17, 7-25, 8-4, 8-6, 9-4,  
9-8, 9-26, 9-32, 9-33, 9-38, 9-41, 9-43, 9-47, 10-5, 10-16,  
10-17, 10-22, 10-23, D-3, D-13, F-8, F-15, I-2, I-16, I-39

Oracle Database 11g 3-24, 7-8, 10-10, 10-14, 10-36, 10-38, C-4,  
I-2, I-3, I-4, I-8, I-9, I-10, I-11, I-14, I-15, I-29,  
I-32, I-33, I-36, I-37, I-38, I-39

**O(계속)**

Oracle Enterprise Manager Grid Control I-13, I-39  
 Oracle Fusion Middleware I-12, I-13, I-39  
 Oracle SQL Developer C-2, C-3, C-4, I-2, I-32, I-40  
 ORDBMS I-2, I-39  
 ORDER BY 절 2-3, 2-19, 2-22, 2-23, 2-24, 2-25, 2-26, 2-28, 2-34,  
     2-38, 2-39, 3-6, 5-15, 5-16, 5-18, 5-28, 8-3, 8-5, 8-7,  
     8-12, 8-18, 8-21, 8-24, 8-27, 8-28, 8-29, 8-30, 10-13

**P**

PRIMARY KEY 제약 조건 9-8, 10-19, 10-20, 10-23

Pseudocolumn 10-27

**Q**

q 연산자 1-23

Query 1-4, 1-8, 1-17, 1-18, 1-21, 1-24, 2-2, 2-5, 2-23, 2-25,  
     2-27, 2-31, 2-33, 3-2, 3-6, 3-26, 3-33, 4-21, 4-35, 5-15, 6-6,  
     6-7, 6-14, 6-16, 6-24, 6-26, 6-28, 6-29, 6-30, 7-3, 7-4, 7-5,  
     7-6, 7-7, 7-8, 7-9, 7-10, 7-11, 7-12, 7-13, 7-14, 7-15, 7-16,  
     7-17, 7-18, 7-19, 7-20, 7-21, 7-22, 7-23, 7-24, 7-25, 7-26, 8-2,  
     8-5, 8-6, 8-13, 8-17, 8-19, 8-20, 8-22, 8-25, 8-26, 8-28, 8-29,  
     8-30, 9-12, 9-15, 9-17, 9-23, 9-33, 9-43, 9-44, 9-47, 10-3, 10-6,  
     10-10, 10-11, 10-13, 10-15, 10-27, 10-31, 10-32, 10-33, 10-34,  
     10-37, 10-40, 10-41, C-18, C-27, D-3, D-5, D-13, D-14, D-15,  
     D-16, D-17, F-7, F-8, F-10, F-15, F-16, F-19, I-4, I-16, I-30

**R**

RDBMS 9-43, I-2, I-18, I-25, I-27, I-39

REFERENCES 1-14, 9-31, 10-24, 10-25, 10-26, 10-27, 10-28,  
     C-5, C-6, C-10, C-19, C-29

REPLACE 2-36, 3-9, 3-13, D-12, D-16

ROUND 및 TRUNC 함수 3-30, 3-32

ROUND 함수 3-17, 3-18, 4-26

RPAD 3-9, 3-13

RR 날짜 형식 3-22, 3-23, 4-22

**S**

SELECT 문 1-1, 1-2, 1-3, 1-4, 1-5, 1-7, 1-10, 1-16, 1-19, 1-21,  
 1-22, 1-25, 1-28, 1-29, 2-6, 2-9, 2-10, 2-12, 2-21,  
 2-23, 2-28, 2-32, 2-35, 2-38, 2-39, 3-2, 3-12, 4-2, 4-44,  
 5-9, 5-15, 5-16, 5-17, 5-18, 5-19, 5-20, 6-2, 6-7, 7-2, 7-5,  
 7-8, 7-10, 7-11, 7-25, 7-26, 8-3, 8-6, 8-7, 8-12, 8-18, 8-19,  
 8-20, 8-21, 8-22, 8-24, 8-25, 8-26, 8-27, 9-3, 9-13, 9-19, 9-22,  
 9-25, 9-33, 9-39, 9-40, 9-41, 9-42, 9-43, 9-44, 9-47, 10-32, D-2,  
 F-2, F-7, F-8, I-4, I-5

Self Join 6-35

SET VERIFY ON 2-36

Snippet 사용 C-23, C-24

SQL 실행 C-2, C-15, C-17, C-31, D-2, D-5, D-18, E-2, E-18

SQL Developer 1-6, 1-8, 1-9, 1-14, 1-17, 1-26, 2-28, 2-29, 2-30, 2-31, 2-33,  
 2-35, 2-36, 6-16, 9-4, 9-21, 9-27, 9-31, 9-32, 9-43, 10-9, 10-41,  
 C-1, C-2, C-3, C-4, C-5, C-6, C-7, C-9, C-10, C-11, C-13,  
 C-17, C-22, C-23, C-25, C-26, C-27, C-28, C-29, C-30, C-31,  
 F-10, I-2, I-7, I-9, I-32, I-37, I-40

subquery 7-3, 7-4, 7-5, 7-6, 7-7, 7-8, 7-9, 7-10, 7-11, 7-12, 7-13,  
 7-14, 7-15, 7-16, 7-17, 7-18, 7-19, 7-20, 7-21, 7-22,  
 7-23, 7-24, 7-25, 9-12, 9-15, 9-17, 9-23, 10-3, 10-6, 10-11,  
 10-13, 10-15, 10-31, 10-32, 10-33, 10-34, 10-37, 10-40

subquery의 그룹 함수 7-3, 7-9, 7-12, 7-16, 7-21

SUBSTR 3-9, 3-10, 3-13, 3-14, 3-32, 4-25

SYSDATE 함수 3-23, 3-24, 9-9

**T**

TO\_CHAR 4-2, 4-3, 4-7, 4-8, 4-9, 4-10, 4-11, 4-16, 4-17, 4-18, 4-19,  
 4-22, 4-23, 4-25, 4-26, 4-27, 4-34, 4-36, 4-44, 4-45, 8-25

TO\_DATE 4-2, 4-3, 4-7, 4-8, 4-9, 4-10, 4-20, 4-21, 4-22,  
 4-23, 4-27, 4-36, 4-44, 4-45, 9-10

TO\_NUMBER 4-2, 4-3, 4-7, 4-8, 4-9, 4-10, 4-20, 4-21, 4-23, 4-27, 4-36,  
 4-43, 4-44

TRIM 3-9, 3-10, 3-13

TRUNC 3-16, 3-18, 3-28, 3-30, 3-32, 4-42, 9-3, 9-13,  
 9-19, 9-24, 9-25, 9-39, 9-42, 9-46, 9-47, I-31

**U**

UNION 연산자 8-13, 8-14, 8-15, 8-25, 8-26, 8-30, 8-31

UNION ALL 8-3, 8-4, 8-5, 8-6, 8-7, 8-12, 8-16, 8-17,  
8-18, 8-21, 8-24, 8-27, 8-29, 8-30

UNIQUE 제약 조건 10-21, 10-22

UPDATE 문 9-3, 9-13, 9-15, 9-16, 9-17, 9-18, 9-19, 9-25, 9-39, 9-42,  
9-43, 10-36

USING 절 6-3, 6-5, 6-8, 6-11, 6-13, 6-14, 6-17, 6-19, 6-22, 6-25, 6-31

**V**

VARIANCE 5-5, 5-8, 5-28

VERIFY 명령 2-3, 2-19, 2-22, 2-26, 2-34, 2-36

**W**

WHERE 절 2-3, 2-4, 2-5, 2-6, 2-7, 2-8, 2-9, 2-11, 2-15, 2-19, 2-22,  
2-26, 2-27, 2-28, 2-31, 2-32, 2-34, 2-37, 2-38, 2-39,  
3-12, 5-9, 5-14, 5-15, 5-18, 5-20, 5-21, 5-22, 5-27, 5-28,  
6-10, 6-14, 6-15, 6-18, 6-36, 7-2, 7-5, 7-13, 7-14, 7-15, 7-23,  
8-5, 9-15, 9-16, 9-22, F-4, F-7, F-10, F-12, F-17, F-21, F-22

**X**

XML C-7, C-9, C-26, C-27, C-30, E-3, E-12, I-9, I-14, I-39