
**Introduction to Oracle9i: SQL
(한글판)**

볼륨 2 • 학생용

40049KR11
제품 1.1
2002년 2월
D34385

ORACLE®

만든이

Nancy Greenberg
Priya Nathan

기술 제공자 및 검토자

Josephine Turner
Martin Alvarez
Anna Atkinson
Don Bates
Marco Berbeek
Andrew Brannigan
Laszlo Czinkoczi
Michael Gerlach
Sharon Gray
Rosita Hanoman
Mozheh Jalali
Sarah Jones
Charbel Khouri
Christopher Lawless
Diana Lorentz
Nina Minchen
Cuong Nguyen
Daphne Nougier
Patrick Odell
Laura Pezzini
Stacey Procter
Maribel Renau
Bryan Roberts
Helen Robertson
Sunshine Salmon
Casa Sharif
Bernard Soleillant
Craig Spoonemore
Ruediger Steffan
Karla Villasenor
Andree Wheeley
Lachlan Williams

Copyright © Oracle Corporation, 2000, 2001. All rights reserved.

이 문서는 Oracle Corp.의 등록된 정보를 포함하고 있습니다. 이 정보는 사용 제한 및 기밀 유지 규정을 포함하는 사용권 계약에 따라 제공되며 저작권법에 의해 보호됩니다. 이 소프트웨어를 리버스 엔지니어링 하는 것은 금지되어 있습니다. 이 문서를 미국 국방성 내의 정부 기관에 제공할 때는 제한된 권리 규정이 적용되며 다음 범례를 적용 할 수 있습니다.

Restricted Rights Legend

Use, duplication or disclosure by the Government is subject to restrictions for commercial computer software and shall be deemed to be Restricted Rights software under Federal law, as set forth in subparagraph (c)(1)(ii) of DFARS 252.227-7013, Rights in Technical Data and Computer Software (October 1988).

This material or any portion of it may not be copied in any form or by any means without the express prior written permission of Oracle Corporation. Any other copying is a violation of copyright law and may result in civil and/or criminal penalties.

If this documentation is delivered to a U.S. Government Agency not within the Department of Defense, then it is delivered with "Restricted Rights," as defined in FAR 52.227-14, Rights in Data-General, including Alternate III (June 1987).

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them in writing to Education Products, Oracle Corporation, 500 Oracle Parkway, Box SB-6, Redwood Shores, CA 94065. Oracle Corporation does not warrant that this document is error-free.

Oracle and all references to Oracle products are trademarks or registered trademarks of Oracle Corporation.

인용된 모든 다른 회사명 또는 제품명은 명시의 목적으로만 사용되었고 각 소유 회사들의 상표일 수 있습니다.

발행인

Nita Brozowski

목차

머리말

교과 과정표

입문

목표	I-2
Oracle9i	I-3
Oracle9i Application Server	I-5
Oracle9i Database	I-6
관계형 데이터베이스 관리 시스템 및 객체 관계형 데이터베이스 관리 시스템	I-7
오라클 인터넷 플랫폼	I-8
시스템 개발 주기	I-9
다양한 데이터 저장 매체	I-11
관계형 데이터베이스 개념	I-12
관계형 데이터베이스 정의	I-13
데이터 모델	I-14
엔티티 관계 모델	I-15
엔티티 관계 모델링 표기법	I-16
여러 테이블 관련시키기	I-18
관계형 데이터베이스에서 사용되는 용어	I-19
관계형 데이터베이스 특성	I-20
SQL을 사용하여 RDBMS와 통신	I-21
관계형 데이터베이스 관리 시스템	I-22
SQL 문	I-23
이 과정에서 사용되는 테이블	I-24

1 기본 SQL SELECT 문 작성

목표	1-2
SQL SELECT 문의 기능	I-3
기본 SELECT 문	1-4
모든 열 선택	1-5
특정 열 선택	1-6
SQL 문 작성	1-7
열 머리글 기본 값	1-8
산술식	1-9
산술 연산자 사용	1-10
연산자 우선순위	1-11
괄호 사용	1-13
널 값 정의	1-14
산술식의 낼 값	1-15
열 별칭 정의	1-16
열 별칭 사용	1-17
연결 연산자	1-18
연결 연산자 사용	1-19
리터럴 문자열	1-20
리터럴 문자열 사용	1-21
중복 행(row)	1-22
중복 행(row) 제거	1-23

SQL과 iSQL*Plus의 상호 작용	1-24
SQL 문과 iSQL*Plus 명령 비교	1-25
iSQL*Plus 개요	1-26
iSQL*Plus에 로그인	1-27
iSQL*Plus 환경	1-28
테이블 구조 표시	1-29
스크립트 파일과의 상호 작용	1-31
요약	1-34
연습 1 개요	1-35
2 데이터 제한 및 정렬	
목표	2-2
선택을 사용한 행(row) 제한	2-3
선택되는 행(row) 제한	2-4
WHERE 절 사용	2-5
문자열 및 날짜	2-6
비교 조건	2-7
비교 조건 사용	2-8
다른 비교 조건	2-9
BETWEEN 조건 사용	2-10
IN 조건 사용	2-11
LIKE 조건 사용	2-12
NULL 조건 사용	2-14
논리 조건	2-15
AND 연산자 사용	2-16
OR 연산자 사용	2-17
NOT 연산자 사용	2-18
우선순위 규칙	2-19
ORDER BY 절	2-22
내림차순으로 정렬	2-23
열 별칭을 기준으로 정렬	2-24
여러 열을 기준으로 정렬	2-25
요약	2-26
연습 2 개요	2-27
3 단일 행(row) 함수	
목표	3-2
SQL 함수	3-3
SQL 함수의 두 유형	3-4
단일 행(row) 함수	3-5
단일 행(row) 함수	3-6
문자 함수	3-7
문자 함수	3-8
대소문자 조작 함수	3-9
대소문자 조작 함수 사용	3-10

- 문자 조작 함수 3-11
 - 문자 조작 함수 사용 3-12
 - 숫자 함수 3-13
 - ROUND 함수 사용 3-14
 - TRUNC 함수 사용 3-15
 - MOD 함수 사용 3-16
 - 날짜 사용 3-17
 - 날짜 계산 3-19
 - 날짜에 산술 연산자 사용 3-20
 - 날짜 함수 3-21
 - 날짜 함수 사용 3-22
 - 연습 3, 1부: 개요 3-24
 - 변환 함수 3-25
 - 암시적(implicit) 데이터 유형 변환 3-26
 - 명시적(explicit) 데이터 유형 변환 3-28
 - 날짜에 TO_CHAR 함수 사용 3-31
 - 날짜 형식 모델 요소 3-32
 - 날짜에 TO_CHAR 함수 사용 3-36
 - 숫자에 TO_CHAR 함수 사용 3-37
 - TO_NUMBER 및 TO_DATE 함수 사용 3-39
 - RR 날짜 형식 3-41
 - RR 날짜 형식 예제 3-42
 - 증점 함수 3-43
 - 일반 함수 3-45
 - NVL 함수 3-46
 - NVL 함수 사용 3-47
 - NVL2 함수 사용 3-48
 - NULLIF 함수 사용 3-49
 - COALESCE 함수 사용 3-50
 - 조건 표현식 3-52
 - CASE 표현식 3-53
 - CASE 표현식 사용 3-54
 - DECODE 함수 3-55
 - DECODE 함수 사용 3-56
 - 요약 3-58
 - 연습 3, 2부: 개요 3-59
- 4 여러 테이블의 데이터 표시
- 목표 4-2
 - 여러 테이블에서 데이터 얻기 4-3
 - 카티시안 곱 4-4
 - 카티시안 곱(Cartesian Product) 생성 4-5
 - 조인 유형 4-6
 - 오라클 구문을 사용하여 테이블 조인 4-7
 - 등가 조인이란? 4-8

등가 조인으로 레코드 검색	4-9
AND 연산자를 사용한 추가 검색 조건	4-10
모호한 열 이름 자세히 지정	4-11
테이블 별칭 사용	4-12
세 개 이상의 테이블 조인	4-13
비동가 조인	4-14
비동가 조인으로 레코드 검색	4-15
포괄 조인	4-16
포괄 조인 구문	4-17
포괄 조인 사용	4-18
자체 조인	4-19
테이블 자체 조인	4-20
연습 4, 1부: 개요	4-21
SQL: 1999 구문을 사용한 테이블 조인	4-22
교차 조인 작성	4-23
자연 조인 작성	4-24
자연 조인으로 레코드 검색	4-25
USING 절을 포함하는 조인 작성	4-26
USING 절로 레코드 검색	4-27
ON 절을 조인 작성	4-28
ON 절로 레코드 검색	4-29
ON 절로 3-way 조인 작성	4-30
내부 조인(inner join)과 포괄 조인 비교	4-31
LEFT OUTER JOIN	4-32
RIGHT OUTER JOIN	4-33
FULL OUTER JOIN	4-34
추가 조건	4-35
요약	4-36
연습 4, 2부: 개요	4-37
5 그룹 함수를 사용한 데이터 집계	
목표	5-2
그룹 함수란?	5-3
그룹 함수 종류	5-4
그룹 함수 구문	5-5
AVG 및 SUM 함수 사용	5-6
MIN 및 MAX 함수 사용	5-7
COUNT 함수 사용	5-8
DISTINCT 키워드 사용	5-10
그룹 함수 및 널 값	5-11
그룹 함수에 NVL 함수 사용	5-12
데이터 그룹 생성	5-13
데이터 그룹 생성: GROUP BY 절 구문	5-14
GROUP BY 절 사용	5-15
여러 열을 기준으로 그룹화	5-17

여러 열에 GROUP BY 절 사용	5-18
그룹 함수를 사용한 잘못된 질의	5-19
그룹 결과 제외	5-21
그룹 결과 제외: HAVING 절	5-22
HAVING 절 사용	5-23
그룹 함수 중첩	5-25
요약	5-26
연습 5 개요	5-27
6 서브 쿼리	
목표	6-2
문제 해결에 서브 쿼리 사용	6-3
서브 쿼리 구문	6-4
서브 쿼리 사용	6-5
서브 쿼리 사용 지침	6-6
서브 쿼리 유형	6-7
단일 행(row) 서브 쿼리	6-8
단일 행(row) 서브 쿼리 실행	6-9
서브 쿼리에서 그룹 함수 사용	6-10
HAVING 절에 서브 쿼리 사용	6-11
이 명령문은 무엇이 잘못되었을까요?	6-12
이 명령문은 행(row)을 반환할까요?	6-13
여러 행(row) 서브 쿼리	6-14
여러 행(row) 서브 쿼리에 ANY 연산자 사용	6-15
여러 행(row) 서브 쿼리에 ALL 연산자 사용	6-16
서브 쿼리에서의 NULL 값	6-17
요약	6-18
연습 6 개요	6-19
7 iSQL*Plus로 읽기 쉬운 출력 작성	
목표	7-2
치환 변수	7-3
& 치환 변수 사용	7-5
치환 변수를 사용한 문자 및 날짜 값	7-7
열 이름, 표현식 및 텍스트 지정	7-8
치환 변수 정의	7-10
DEFINE 및 UNDEFINE 명령	7-11
& 치환 변수에 DEFINE 명령 사용	7-12
&& 치환 변수 사용	7-13
VERIFY 명령 사용	7-14
iSQL*Plus 환경 사용자 정의	7-15
SET 명령 변수	7-16
iSQL*Plus 형식 명령	7-17
COLUMN 명령	7-18
COLUMN 명령 사용	7-19

COLUMN 형식 모델 7-20
BREAK 명령 사용 7-21
TTITLE 및 BTITLE 명령 사용 7-22
스크립트 파일을 작성하여 보고서 실행 7-24
예제 보고서 7-26
요약 7-28
연습 7 개요 7-29

8 데이터 조작

목표 8-2
데이터 조작어 8-3
테이블에 새 행(row) 추가 8-4
INSERT 문 구문 8-5
새 행(row) 삽입 8-6
널 값을 갖는 행(row) 삽입 8-7
특정 값 삽입 8-8
특정 날짜 값 삽입 8-9
스크립트 작성 8-10
다른 테이블에서 행(row) 복사 8-11
테이블의 데이터 변경 8-12
UPDATE 문 구문 8-13
테이블의 행(row) 갱신 8-14
서브 쿼리로 두 열 갱신 8-15
다른 테이블을 기반으로 행(row) 갱신 8-16
행(row) 갱신: 무결성 제약 조건 오류 8-17
테이블에서 행(row) 제거 8-18
DELETE 문 구문 8-19
테이블에서 행(row) 삭제 8-20
다른 테이블을 기반으로 행(row) 삭제 8-21
행(row) 삭제: 무결성 제약 조건 오류 8-22
INSERT 문에 서브 쿼리 사용 8-23
DML 문에 WITH CHECK OPTION 키워드 사용 8-25
명시적(Explicit) 기본 기능 개요 8-26
명시적(Explicit) 기본값 사용 8-27
MERGE 문 8-28
MERGE 문 구문 8-29
행(row) 병합 8-30
데이터베이스 트랜잭션 8-32
COMMIT 및 ROLLBACK 문의 장점 8-34
트랜잭션 제어 8-35
표 시자까지 변경 내용 끌백 8-36
암시적(implicit) 트랜잭션 처리 8-37
COMMIT 또는 ROLLBACK 실행 이전의 데이터 상태 8-38
COMMIT 실행 이후의 데이터 상태 8-39
데이터 커밋 8-40

ROLLBACK 실행 이후의 데이터 상태	8-41
명령문 레벨 롤백	8-42
읽기 일관성	8-43
읽기 일관성 구현	8-44
잠금	8-45
암시적(Implicit) 잠금	8-46
요약	8-47
연습 8 개요	8-48
읽기 일관성 예제	8-53
9 테이블 생성 및 관리	
목표	9-2
데이터베이스 객체	9-3
이름 지정 규칙	9-4
CREATE TABLE 문	9-5
다른 사용자의 테이블 참조	9-6
DEFAULT 옵션	9-7
테이블 생성	9-8
오라클 데이터베이스의 테이블	9-9
데이터 딕셔너리 질의	9-10
데이터 유형	9-11
DateTime 데이터 유형	9-13
TIMESTAMP WITH TIME ZONE 데이터 유형	9-15
TIMESTAMP WITH LOCAL TIME 데이터 유형	9-16
INTERVAL YEAR TO MONTH 데이터 유형	9-17
INTERVAL DAY TO SECOND 데이터 유형	9-18
서브 쿼리 구문을 사용한 테이블 생성	9-20
서브 쿼리를 사용한 테이블 생성	9-21
ALTER TABLE 문	9-22
열 추가	9-24
열 수정	9-26
열 삭제	9-27
SET UNUSED 옵션	9-28
테이블 삭제	9-29
객체 이름 변경	9-30
테이블 절단	9-31
테이블에 주석 추가	9-32
요약	9-33
연습 9 개요	9-34

10 제약 조건 포함

- 목표 10-2
- 제약 조건이란? 10-3
- 제약 조건 지침 10-4
- 제약 조건 정의 10-5
- NOT NULL 제약 조건 10-7
- UNIQUE 제약 조건 10-9
- PRIMARY KEY 제약 조건 10-11
- FOREIGN KEY 제약 조건 10-13
- FOREIGN KEY 제약 조건 키워드 10-15
- CHECK 제약 조건 10-16
- 제약 조건 추가 구문 10-17
- 제약 조건 추가 10-18
- 제약 조건 삭제 10-19
- 제약 조건 비활성화 10-20
- 제약 조건 활성화 10-21
- 제약 조건 연쇄화 10-22
- 제약 조건 보기 10-24
- 제약 조건과 연관된 열 보기 10-25
- 요약 10-26
- 연습 10 개요 10-27

11 뷰 생성

- 목표 11-2
- 데이터베이스 객체 11-3
- 뷰란? 11-4
- 뷰 사용 목적 11-5
- 단순 뷰 및 복합 뷰 11-6
- 뷰 생성 11-7
- 뷰에서 데이터 검색 11-10
- 뷰 질의 11-11
- 뷰 수정 11-12
- 복합 뷰 생성 11-13
- 뷰를 통한 DML 작업 수행에 관한 규칙 11-14
- WITH CHECK OPTION 절 사용 11-17
- DML 작업 거부 11-18
- 뷰 제거 11-20
- 인라인 뷰 11-21
- "Top-N" 분석 11-22
- "Top-N" 분석 수행 11-23
- Top-N 분석 예제 11-24
- 요약 11-25
- 연습 11 개요 11-26

- 12 기타 데이터베이스 객체**
목표 12-2
데이터베이스 객체 12-3
시퀀스란? 12-4
CREATE SEQUENCE 문 구문 12-5
시퀀스 생성 12-6
시퀀스 확인 12-7
NEXTVAL 및 CURRVAL 의사 열 12-8
시퀀스 사용 12-10
시퀀스 수정 12-12
시퀀스 수정에 대한 지침 12-13
시퀀스 제거 12-14
인덱스란? 12-15
인덱스 생성 방법 12-16
인덱스 생성 12-17
인덱스 생성이 필요한 경우 12-18
인덱스를 생성하지 않아야 할 경우 12-19
인덱스 확인 12-20
함수 기반 인덱스 12-21
인덱스 제거 12-23
동의어 12-24
동의어 생성 및 제거 12-25
요약 12-26
연습 12 개요 12-27

- 13 사용자 액세스 제어**
목표 13-2
사용자 액세스 제어 13-3
권한 13-4
시스템 권한 13-5
사용자 생성 13-6
사용자 시스템 권한 13-7
시스템 권한 부여 13-8
를 이란? 13-9
를 생성 및 권한 부여 13-10
암호 변경 13-11
객체 권한 13-12
객체 권한 부여 13-14
WITH GRANT OPTION 및 PUBLIC 키워드 사용 13-15
부여된 권한 확인 13-16
객체 권한 추소 방법 13-17
객체 권한 추소 13-18
데이터베이스 링크 13-19
요약 13-21
연습 13 개요 13-22

14 SQL 강습

강습 개요 14-2

15 SET 연산자 사용

목표 15-2

SET 연산자 15-3

이 단원에서 사용되는 테이블 15-4

UNION 연산자 15-7

UNION 연산자 사용 15-8

UNION ALL 연산자 15-10

UNION ALL 연산자 사용 15-11

INTERSECT 연산자 15-12

INTERSECT 연산자 사용 15-13

MINUS 연산자 15-14

SET 연산자 지침 15-16

Oracle Server와 SET 연산자 15-17

SELECT 문 일치 15-18

행(row) 순서 제어 15-20

요약 15-21

연습 15 개요 15-22

16 Oracle9i Datetime 함수

목표 16-2

시간대 16-3

Oracle9i datetime 지원 16-4

TZ_OFFSET 16-6

CURRENT_DATE 16-8

CURRENT_TIMESTAMP 16-9

LOCALTIMESTAMP 16-10

DBTIMEZONE 및 SESSIONTIMEZONE 16-11

EXTRACT 16-12

FROM_TZ를 사용하여 TIMESTAMP 변환 16-13

TO_TIMESTAMP 및 TO_TIMESTAMP_TZ를 사용하여

STRING을 TIMESTAMP로 변환 16-14

TO_YMINTERVAL을 사용하여 시간 간격 변환 16-15

요약 16-16

연습 16 개요 16-17

17 GROUP BY 절의 항상된 기능

목표 17-2

그룹 함수 복습 17-3

GROUP BY 절 복습 17-4

HAVING 절 복습 17-5

GROUP BY에 ROLLUP 및 CUBE 연산자 사용 17-6

ROLLUP 연산자 17-7

ROLLUP 연산자 예제 17-8

CUBE 연산자 17-9
CUBE 연산자: 예제 17-10
GROUPING 함수 17-11
GROUPING 함수: 예제 17-12
GROUPING SETS 17-13
GROUPING SETS: 예제 17-15
조합 열 17-17
조합 열: 예제 17-19
연결된 그룹화 17-21
연결된 그룹화 예제 17-22
요약 17-23
연습 17 개요 17-24

18 고급 서브 쿼리

목표 18-2
서브 쿼리란? 18-3
서브 쿼리 18-4
서브 쿼리 사용 18-5
여러 열 서브 쿼리 18-6
열 비교 18-7
쌍(pairwise) 비교 서브 쿼리 18-8
비쌍 비교 서브 쿼리 18-9
FROM 절에 서브 쿼리 사용 18-10
스칼라 서브 쿼리식 18-11
스칼라 서브 쿼리: 예제 18-12
상관 서브 쿼리 18-14
상관 서브 쿼리 사용 18-16
EXISTS 연산자 사용 18-18
NOT EXISTS 연산자 사용 18-20
상관 UPDATE 18-21
상관 DELETE 18-24
WITH 절 18-26
WITH 절: 예제 18-27
요약 18-29
연습 18 개요 18-31

19 계층 검색

목표 19-2
EMPLOYEES 테이블의 예제 데이터 19-3
자연 트리 구조 19-4
계층 질의 19-5
트리 검색 19-6
트리 검색: 아래에서 위로 19-8
트리 검색: 위에서 아래로 19-9
LEVEL 의사 열을 사용하여 순위 결정 19-10

LEVEL 및 LPAD를 사용하여 계층 보고서 형식 지정	19-11
분기 제거	19-13
요약	19-14
연습 19 개요	19-15
20 Oracle9i에서 확장된 DML 및 DDL 문 기능	
목표	20-2
INSERT 문 복습	20-3
UPDATE 문 복습	20-4
다중 테이블 INSERT 문 개요	20-5
다중 테이블 INSERT 문 개요	20-6
다중 테이블 INSERT 문 유형	20-7
다중 테이블 INSERT 문	20-8
무조건 INSERT ALL	20-10
조건 INSERT ALL	20-11
조건 FIRST INSERT	20-13
피벗 INSERT	20-15
외부 테이블	20-18
외부 테이블 생성	20-19
외부 테이블 생성 예제	20-20
외부 테이블 질의	20-23
CREATE TABLE 문 내의 CREATE INDEX	20-24
요약	20-25
연습 20 개요	20-26
A 해답	
B 테이블 설명 및 데이터	
C SQL* Plus 사용	
D 고급 스크립트 작성	
E 오라클의 구조적 구성 요소	
인덱스	
추가 연습	
추가 연습 해답	
추가 연습 테이블과 설명	

11

뷰 생성

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 뷰 설명
- 뷰의 정의 생성 및 변경, 뷔 삭제
- 뷔를 통해 데이터 검색
- 뷔를 통해 데이터 삽입, 갱신 및 삭제
- 인라인 뷔 생성 및 사용
- “Top-N” 분석 수행

ORACLE

11-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 뷔를 생성 및 사용하는 방법을 설명하며, 관련 데이터 딕셔너리 객체를 질의하여 뷔에 관한 정보를 검색하는 방법도 설명합니다. 마지막으로 인라인 뷔를 생성 및 사용하는 방법과 인라인 뷔를 사용하여 Top-N 분석을 수행하는 방법을 설명합니다.

데이터베이스 객체

객체	설명
테이블	기본 저장 단위며 행과 열로 구성됩니다.
뷰	논리적으로 하나 이상의 테이블에 있는 데이터의 부분 집합을 나타냅니다.
시퀀스	기본 키 값을 생성합니다.
인덱스	질의의 성능을 향상시킵니다.
동의어	객체의 다른 이름입니다.

ORACLE

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	240
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	170
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	170
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	90
104	Alfred	Bergl	ABERGL	590.423.4568	01-JAN-95	SA_MAN	60
105	Galen	Carter	GCARTER	590.423.4569	29-JUL-95	SA_REP	42
106	Marcia	Frederick	MFREDERICK	590.423.4570	08-DEC-95	SA_REP	58
107	Victoria	Fuller	VFULMER	590.423.4571	23-JAN-96	SA_REP	35
108	Robert	Kingsley	RKINGSLEY	590.423.4572	03-JAN-96	SA_REP	31
109	David	Lund	DLUND	590.423.4573	20-APR-96	SA_REP	28
110	Michael	Schoen	MSCHOEN	590.423.4574	20-APR-96	SA_REP	25
111	Patricia	Wilman	PWILMAN	511.414.1234	01-MAY-99	SA_MAN	105
149	Dotkey			10500		SA_REP	110
174	Abel			11000		SA_REP	66
176	Taylor			8600		SA_REP	70
177	Patricia	Wilman	PWILMAN	511.414.1234	01-MAY-99	SA_REP	44
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	44
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	130
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	60
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	120
206	William	Gietz	WGRIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	65

20 rows selected.

ORACLE

11-4

Copyright © Oracle Corporation, 2001. All rights reserved.

뷰란?

테이블 뷰를 생성하면 데이터의 논리적인 부분 집합 또는 조합을 나타낼 수 있습니다. 뷔는 테이블 또는 다른 뷔를 기반으로 하는 논리 테이블로서 자체적으로 데이터를 포함하지는 않지만 창과 마찬가지로 뷔를 통해 테이블의 데이터를 보거나 변경할 수 있습니다. 뷔의 기반이 되는 테이블을 기본 테이블이라고 하며 뷔는 데이터 덕셔너리에 SELECT 문으로 저장됩니다.

뷰 사용 목적

- 데이터 액세스를 제한하기 위해
- 복잡한 질의를 쉽게 작성하기 위해
- 데이터 독립성을 제공하기 위해
- 동일한 데이터로부터 다양한 결과를 얻기 위해

- 무관련 공간 X
논리적 Tab / 물리 Tab
- create or replace view 이름
as (sub Q)

ORACLE

11-5

Copyright © Oracle Corporation, 2001. All rights reserved.

뷰의 장점

- 뷰는 테이블의 열을 선택적으로 표시할 수 있으므로 데이터 액세스를 제한합니다.
- 뷰를 사용하면 복잡한 질의의 결과를 검색하는 단순한 질의를 만들 수 있습니다. 예를 들어, 조인 문 작성 방법을 몰라도 뷰를 사용하여 여러 테이블에 있는 정보를 질의할 수 있습니다.
- 뷰는 임시 사용자와 응용 프로그램에 대해 데이터 독립성을 제공하며 하나의 뷰를 사용하여 여러 테이블에 있는 데이터를 검색할 수 있습니다.
- 뷰를 사용하면 사용자 그룹이 특정 기준에 따라 데이터를 액세스할 수 있습니다.

자세한 내용은 *Oracle9i SQL Reference*, “CREATE VIEW”를 참조하십시오.

단순 뷰 및 복합 뷰

특징	단순 뷰	복합 뷰
테이블 수	하나	하나 이상
함수 포함	아니오	예
데이터 그룹 포함	아니오	예
뷰를 통한 DML 작업	예	불가능한 경우도 있음

ORACLE

단순 뷰와 복합 뷰 비교

뷰는 단순 뷰와 복합 뷰로 구분되는데 기본적인 차이는 INSERT, UPDATE 및 DELETE 등의 DML 작업과 관련되어 있습니다.

- 단순 뷰의 특징은 다음과 같습니다.
 - 한 테이블에서만 데이터를 얻습니다.
 - 함수 또는 데이터 그룹을 포함하지 않습니다.
 - 뷰를 통해 DML 작업을 수행할 수 있습니다.
- 복합 뷰의 특징은 다음과 같습니다.
 - 여러 테이블에서 데이터를 얻습니다.
 - 함수 또는 데이터 그룹을 포함합니다.
 - 뷰를 통해 DML 작업을 수행할 수 없는 경우도 있습니다.

서의 복합화
can

뷰 생성

- CREATE VIEW 문에 서브 쿼리를 포함시킵니다.

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

- 서브 쿼리는 복합 SELECT 구문을 포함할 수 있습니다.

ORACLE

11-7

Copyright © Oracle Corporation, 2001. All rights reserved.

뷰 생성

CREATE VIEW 문에 서브 쿼리를 포함시켜 뷰를 생성할 수 있습니다.

구문 설명:

OR REPLACE	뷰가 이미 있어도 다시 생성합니다.
FORCE	기본 테이블의 존재 여부에 관계 없이 뷰를 생성합니다.
NOFORCE	기본 테이블이 있는 경우만 뷰를 생성합니다(기본값).
view	뷰 이름입니다.
alias	뷰 질의에 의해 선택되는 표현식의 이름을 지정합니다. (별칭 수는 뷰에 의해 선택되는 표현식 수와 일치해야 합니다.)
subquery	완전한 SELECT 문입니다. (SELECT 목록에서 열에 대해 별칭을 사용할 수 있습니다.)
WITH CHECK OPTION	뷰를 통해 액세스할 수 있는 행만 삽입 또는 갱신할 수 있도록 지정합니다.
constraint	CHECK OPTION 제약 조건에 지정된 이름입니다.
WITH READ ONLY	이 뷰를 통해서는 DML 작업을 수행할 수 없도록 합니다.

뷰 생성

- 부서 80의 사원에 대한 자세한 정보를 포함하는 뷰 **EMPVU80**을 생성합니다.

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

- iSQL*Plus의 DESCRIBE 명령을 사용하여 뷰의 구조를 표시합니다.

```
DESCRIBE empvu80
```

ORACLE

11-8

Copyright © Oracle Corporation, 2001. All rights reserved.

뷰 생성(계속)

슬라이드의 예제는 부서 80에 있는 각 사원의 사원 번호, 이름 및 급여를 포함하는 뷰를 생성합니다.

iSQL*Plus의 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)

뷰 생성 규칙:

- 뷰를 정의하는 서브 쿼리는 조인, 그룹, 서브 쿼리 등의 복합 SELECT 구문을 포함할 수 있습니다.
- 뷰를 정의하는 서브 쿼리는 ORDER BY 절을 포함할 수 없습니다. ORDER BY 절은 뷰에서 데이터를 검색할 때 지정합니다.
- WITH CHECK OPTION을 사용하여 생성한 뷰의 제약 조건 이름을 지정하지 않으면 *SYS_Cn* 형식의 기본 이름이 지정됩니다.
- OR REPLACE 옵션을 사용하면 뷰를 삭제하여 다시 생성하거나 이전에 부여 받은 객체 권한을 다시 부여 하지 않고도 뷰의 정의를 변경할 수 있습니다.

뷰 생성

- 서브 쿼리에 열 별칭을 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;
View created.
```

- 주어진 별칭 이름을 사용하여 이 뷰에서 열을 선택합니다.

ORACLE

11-3

Copyright © Oracle Corporation, 2001. All rights reserved.

뷰 생성(계속)

서브 쿼리에 열 별칭을 포함시켜 열 이름을 제어할 수 있습니다.

슬라이드의 예제는 부서 50의 모든 사원에 대해 별칭이 ID_NUMBER인 사원 번호(EMPLOYEE_ID), 별칭이 NAME인 이름(LAST_NAME), 별칭이 ANN_SALARY인 연봉(SALARY)을 포함하는 뷰를 생성합니다.

또는 별칭을 CREATE 문 뒤이면서 SELECT 서브 쿼리 앞인 위치에 사용할 수도 있습니다. 나열되는 별칭의 수는 서브 쿼리에서 선택되는 표현식의 수와 일치해야 합니다.

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

뷰에서 데이터 검색

```
SELECT *  
FROM salvu50;
```

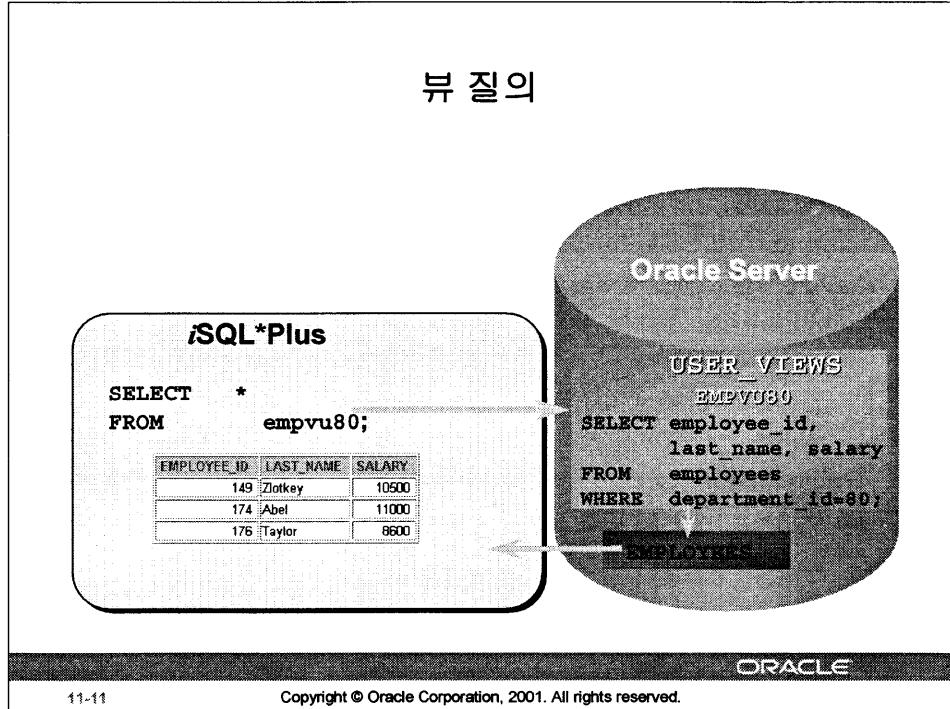
ID NUMBER	NAME	ANN SALARY
124	Mourgos	69600
141	Rajs	42000
142	Daves	37200
143	Matos	31200
144	Vargas	30000

ORACLE

뷰에서 데이터 검색

데이터에서 데이터를 검색하는 것처럼 뷰에서도 데이터를 검색할 수 있습니다. 전체 뷰의 내용을 표시할 수도 있고 특정 행 및 열만 표시할 수도 있습니다.

뷰 질의



11-11

Copyright © Oracle Corporation, 2001. All rights reserved.

데이터 덕셔너리의 뷰

뷰가 생성되면 USER_VIEWS라는 데이터 덕셔너리 뷰를 질의하여 뷰의 이름 및 정의를 볼 수 있습니다. 뷔를 구성하는 SELECT 문의 텍스트는 LONG 열에 저장됩니다.

뷰를 사용한 데이터 액세스

뷰를 사용하여 데이터를 액세스하면 Oracle server가 다음 작업을 수행합니다.

1. 데이터 덕셔너리 테이블인 USER_VIEWS에서 뷔 정의를 검색합니다.
2. 뷔의 기본 테이블에 대한 액세스 권한을 확인합니다.
3. 뷔 질의를 기본 테이블을 기반으로 하는 동등한 작업으로 변환합니다. 즉 데이터를 기본 테이블에서 검색 또는 간접합니다.

뷰 수정

- CREATE OR REPLACE VIEW 절을 사용하여 EMPVU80 뷰를 수정하고 각 열 이름에 대해 별칭을 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;
View created.
```

- CREATE VIEW 절의 열 별칭은 서브 쿼리의 열과 동일한 순서로 나열해야 합니다.

ORACLE

11-12

Copyright © Oracle Corporation, 2001. All rights reserved.

뷰 수정

OR REPLACE 옵션을 사용하면 해당 이름의 뷰가 이미 있더라도 뷰를 생성하여 해당 소유자의 이전 버전의 뷰를 대체할 수 있습니다. 즉 이전 객체 권한을 삭제하고 다시 생성하여 권한을 다시 부여하지 않고도 뷰를 변경할 수 있습니다.

참고: CREATE VIEW 절에서 열 별칭을 지정하는 경우 서브 쿼리의 열과 동일한 순서로 나열해야 합니다.

복합 뷰 생성

그룹 함수를 포함하는 복합 뷰를 생성하여 두 테이블의 값을 표시합니다.

```
CREATE 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, departments d
       WHERE      e.department_id = d.department_id
      GROUP BY    d.department_name;
View created.
```

ORACLE

11-13

Copyright © Oracle Corporation, 2001. All rights reserved.

복합 뷰 생성

슬라이드의 예제는 부서별로 부서 이름, 최저 급여, 최고 급여 및 평균 급여를 표시하는 복합 뷰를 생성합니다. 뷰에 대체 이름이 지정되어 있는데 이것은 뷰의 열이 함수나 표현식으로부터 파생될 경우 반드시 필요합니다.

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

```
SELECT *
  FROM    dept_sum_vu;
```

NAME	MIN SAL	MAX SAL	AVG SAL
Accounting	8300	12000	10150
Administration	4400	4400	4400
Executive	17000	24000	19333.3333
IT	4200	9000	6400
Marketing	6000	13000	9500
Sales	8600	11000	10033.3333
Shipping	2500	5800	3500

7 rows selected.

뷰를 통한 DML 작업 수행에 관한 규칙

- 단순 뷰를 통해 DML 작업을 수행할 수 있습니다.
- 뷰에 다음이 포함된 경우 행을 제거할 수 없습니다.
 - 그룹 함수
 - GROUP BY 절
 - DISTINCT 키워드
 - 의사 열 ROWNUM 키워드

ORACLE

11-14

Copyright © Oracle Corporation, 2001. All rights reserved.

뷰를 통해 DML 작업 수행

DML 작업이 특정 규칙을 따르는 경우 뷰를 통해 데이터에 대한 DML 작업을 수행할 수 있습니다.

뷰가 다음을 포함하는 경우 행을 제거할 수 없습니다.

- 그룹 함수
- GROUP BY 절
- DISTINCT 키워드
- 의사 열 ROWNUM 키워드

뷰를 통한 DML 작업 수행에 관한 규칙

뷰에 다음이 포함된 경우 데이터를 수정할 수 없습니다.

- 그룹 함수
- GROUP BY 절
- DISTINCT 키워드
- 의사 열 ROWNUM 키워드
- 표현식에 의해 정의된 열

ORACLE

11-15

Copyright © Oracle Corporation, 2001. All rights reserved.

뷰를 통해 DML 작업 수행(계속)

뷰에 이전 슬라이드에서 설명한 조건이 포함되어 있지 않고 표현식으로 정의된 열(예: SALARY * 12)도 포함되어 있지 않으면 뷰를 통해 데이터를 수정할 수 있습니다.

뷰를 통한 DML 작업 수행에 관한 규칙

뷰에 다음이 포함된 경우 뷰를 통해 데이터를 추가할 수 없습니다.

- 그룹 함수
- GROUP BY 절
- DISTINCT 키워드
- 의사 열 ROWNUM 키워드
- 표현식에 의해 정의된 열
- 기본 테이블에서 뷰에 의해 선택되지 않은 열에 NOT NULL 제약 조건이 있는 경우

ORACLE

11-18

Copyright © Oracle Corporation, 2001. All rights reserved.

뷰를 통해 DML 작업 수행(계속)

슬라이드에 나열된 항목이 포함되어 있거나, 기본 테이블의 열 중에 뷰에 의해 선택되지 않은 NOT NULL 열이 있는데 이 열에 기본값이 지정되어 있지 않은 경우에는 뷰를 통해 데이터를 추가할 수 없습니다. 뷰에 필수 항목에 대한 값이 모두 있어야 데이터를 추가할 수 있습니다. 뷰를 통해 값을 직접 기본 테이블에 추가한다는 점을 기억하십시오.

자세한 내용은 *Oracle9i SQL Reference*, “CREATE VIEW”를 참조하십시오.

WITH CHECK OPTION 절 사용

- * WITH CHECK OPTION 절을 사용하면 뷰를 통한 DML 작업이 뷰의 도메인 내에서 수행됩니다.

```
CREATE OR REPLACE VIEW empvu20
AS SELECT *
  FROM employees
 WHERE department_id = 20
  WITH CHECK OPTION CONSTRAINT empvu20_ck ;
View created.
```

- * 뷰를 통해 행의 부서 번호 변경을 시도하면 WITH CHECK OPTION 제약 조건에 위배되므로 오류가 발생합니다.

ORACLE

11-17

Copyright © Oracle Corporation, 2001. All rights reserved.

WITH CHECK OPTION 절 사용

뷰를 통해 참조 무결성을 검사할 수 있으며 제약 조건을 데이터베이스 레벨로 적용할 수도 있습니다. 뷰는 데이터 무결성 보존에 사용할 수 있지만 사용이 매우 제한되어 있습니다.

WITH CHECK OPTION 절을 사용하면 뷰를 통한 INSERT 및 UPDATE 작업이 수행될 경우 해당 뷰가 선택할 수 없는 행은 생성되지 않습니다. 따라서 삽입 또는 갱신되는 데이터에 대해 무결성 제약 조건을 적용하고 데이터의 유효성을 검사할 수 있습니다.

뷰가 선택하지 않은 행에 대해 DML 작업을 시도하면 지정된 제약 조건 이름과 함께 다음 오류가 표시됩니다.

```
UPDATE empvu20
   SET department_id = 10
 WHERE employee_id = 201;
UPDATE empvu20
*
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

참고: 부서 번호를 10으로 변경하면 뷰에서 해당 사원을 더 이상 볼 수 없으므로 어떤 행도 갱신되지 않습니다. 이 뷰에서는 WITH CHECK OPTION 절을 사용하므로 부서 20의 사원만 볼 수 있으며 뷰를 통해 해당 사원의 부서 번호를 변경할 수는 없습니다.

DML 작업 거부

- 뷰 정의에 **WITH READ ONLY** 옵션을 추가하면 DML 작업을 거부할 수 있습니다.
- 뷰를 통해 행에 DML 작업을 수행하면 **Oracle server 오류**가 발생합니다.

ORACLE

11-18

Copyright © Oracle Corporation, 2001. All rights reserved.

DML 작업 거부

WITH READ ONLY 옵션을 사용하여 뷰를 생성하면 해당 뷰에서 DML 작업이 수행되지 않도록 할 수 있습니다. 슬라이드의 예제는 뷰를 통해 DML 작업이 수행되지 않도록 EMPVU10 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;
View created.
```

ORACLE

11-18

Copyright © Oracle Corporation, 2001. All rights reserved.

DML 작업 거부

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

```
DELETE FROM empvu10
WHERE employee_number = 200;
DELETE FROM empvu10
*
ERROR at line 1:
ORA-01752: cannot delete from view without exactly one key-
preserved table
```

읽기 전용 제약 조건이 있는 뷰를 사용하여 행을 삽입하거나 수정하려고 하면 다음 Oracle server 오류가 발생합니다.

01733: virtual column not allowed here.

뷰 제거

뷰는 데이터베이스의 기본 테이블을 기반으로 하므로
데이터 손실 없이 뷰를 제거할 수 있습니다.

```
DROP VIEW view;
```

```
DROP VIEW empvu80;  
View dropped.
```

ORACLE

11-20

Copyright © Oracle Corporation, 2001. All rights reserved.

뷰 제거

DROP VIEW 문을 사용하면 뷰를 제거할 수 있는데 이 명령문은 데이터베이스에서 뷰의 정의를 제거하는 것입니다. 뷰를 삭제하면 뷰의 기반이 되는 테이블은 아무 영향도 받지 않지만 삭제된 뷰를 기반으로 하는 뷰나 다른 응용 프로그램은 사용할 수 없게 됩니다. 뷰의 생성자 또는 DROP ANY VIEW 권한을 가진 사용자만 뷰를 제거할 수 있습니다.

구문 설명:

view 뷰 이름입니다.

인라인 뷰

- 인라인 뷰는 SQL 문에서 사용 가능한 별칭(또는 상관 이름)을 사용하는 서브 쿼리입니다.
- 기본 질의의 FROM 절에 있는 명명된 서브 쿼리는 인라인 뷰의 한 예입니다.
- 인라인 뷰는 스키마 객체가 아닙니다.

11-21

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

인라인 뷰

FROM 절에 서브 쿼리를 만들어 이 서브 쿼리에 별칭을 부여할 경우 인라인 뷰가 생성됩니다. 서브 쿼리는 기본 질의에서 참조할 수 있는 데이터 소스를 정의합니다. 다음 예제에서, 인라인 뷰 b는 EMPLOYEES 테이블의 각 부서에 대해 모든 부서 번호 및 최고 급여에 대한 세부 사항을 반환합니다. 기본 질의의 WHERE a.department_id = b.department_id AND a.salary < b.maxsal 절은 자신의 부서에서 최고 급여보다 적게 버는 모든 사원의 사원 이름, 급여, 부서 번호 및 최고 급여를 표시합니다.

```
SELECT a.last_name, a.salary, a.department_id, b.maxsal
  FROM employees a, (SELECT department_id, max(salary) maxsal
                        FROM employees
                       GROUP BY department_id) b
 WHERE a.department_id = b.department_id
   AND a.salary < b.maxsal;
```

LAST_NAME	SALARY	DEPARTMENT_ID	MAXSAL
Fay	6000	20	13000
Rajs	3600	50	5800
Davies	3100	50	5800
Matos	2600	50	5800
Vargas	2500	50	5800

■ ■ ■

12 rows selected.

“Top-N” 분석

- **Top-N** 질의는 열에서 가장 큰 n 개의 값 또는 가장 작은 n 개의 값을 요청합니다. 예를 들어
 - 가장 많이 팔린 제품 10가지는?
 - 가장 적게 팔린 제품 10가지는?
- 최대값 집합 및 최소값 집합은 모두 **Top-N** 질의에 해당합니다.

ORACLE

11-22

Copyright © Oracle Corporation, 2001. All rights reserved.

“Top-N” 분석

Top-N 질의는 테이블에서 조건에 맞는 최상위 레코드 n 개 또는 최하위 레코드 n 개를 표시하는 시나리오에 유용합니다. 이 결과 집합은 다른 분석에도 사용할 수 있습니다. 예를 들어, Top-N 분석을 사용하여 다음 유형의 질의를 수행할 수 있습니다.

- 회사의 최상위 소득자 세 명
- 회사에 가장 최근에 입사한 신입 사원 네 명
- 제품을 가장 많이 판매한 영업 사원 두 명
- 최근 6개월 동안 가장 많이 팔린 제품 세 가지

“Top-N” 분석 수행

Top-N 분석 질의의 상위 레벨 구조는 다음과 같습니다.

```
SELECT [column_list], ROWNUM
FROM   (SELECT [column_list]
        FROM table
        ORDER BY Top-N_column)
WHERE  ROWNUM <= N;
```

ORACLE

11-23

Copyright © Oracle Corporation, 2001. All rights reserved.

“Top-N” 분석 수행

Top-N 질의는 다음에 설명된 요소와 함께 일관성 있는 중첩 질의 구조를 사용합니다.

- 정렬된 데이터 목록을 생성하는 서브 쿼리 또는 인라인 뷰. 서브 쿼리 또는 인라인 뷰는 ORDER BY 절을 포함하므로 데이터를 원하는 순서로 정렬할 수 있습니다. 최대값을 검색하려면 DESC 파라미터가 필요합니다.
- 최종 결과 집합의 행 수를 제한하는 외부 질의. 외부 질의는 다음 구성 요소를 포함합니다.
 - ROWNUM 의사 열: 서브 쿼리에서 반환되는 각 행에 1부터 시작하는 순차 값을 할당합니다.
 - WHERE 절: 반환될 n개 행을 지정합니다. 외부 WHERE 절은 < 또는 <= 연산자를 사용해야 합니다.

Top-N 분석 예제

EMPLOYEES 테이블에서 최상위 소득자 세 명의 이름 및 급여를 표시합니다.

```
SELECT ROWNUM as RANK, last_name, salary  
FROM (SELECT last_name,salary FROM employees  
      ORDER BY salary DESC)  
WHERE ROWNUM <= 3;
```

RANK	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000

ORACLE

11-24

Copyright © Oracle Corporation, 2001. All rights reserved.

"Top-N" 분석 예제

슬라이드의 예제는 EMPLOYEES 테이블에서 최상위 소득자 세 명의 이름 및 급여를 표시하는 방법을 보여줍니다. 서브 쿼리는 EMPLOYEES 테이블에서 모든 사원의 이름 및 급여에 대한 세부 사항을 반환하여 급여에 대해 내림차순으로 정렬합니다. 기본 질의에 있는 WHERE ROWNUM < 3 절에 따라 결과 집합 중에서 처음 세 개의 레코드만 표시됩니다.

다음은 인라인 뷰를 사용하는 Top-N 분석의 또 다른 예제입니다. 다음 예제는 인라인 뷰 E를 사용하여 회사의 최장기 근무 사원 네 명을 표시합니다.

```
SELECT ROWNUM as SENIOR,E.last_name, E.hire_date  
FROM (SELECT last_name,hire_date FROM employees  
      ORDER BY hire_date)E  
WHERE rownum <= 4;
```

SENIOR	LAST_NAME	HIRE_DATE
1	King	17-JUN-87
2	Whalen	17-SEP-87
3	Kochhar	21-SEP-89
4	Hunold	03-JAN-90

요약

이 단원에서는 뷰가 다른 테이블 또는 뷰에서 파생되며 다음의 이점을 제공한다는 것을 배웠습니다.

- 데이터베이스 액세스를 제한
- 질의를 단순화
- 데이터 독립성 제공
- 동일한 데이터에 대해 다양한 뷰 제공
- 기본 데이터를 제거하지 않고도 삭제 가능
- 인라인 뷰는 별칭 이름을 가진 서브 쿼리임
- **Top-N** 분석은 서브 쿼리 및 외부 질의를 통해 수행됨

ORACLE

11-25

Copyright © Oracle Corporation, 2001. All rights reserved.

뷰란?

뷰는 테이블 또는 다른 뷰를 기반으로 하여 뷰를 통해 테이블의 데이터를 보거나 변경할 수 있으므로 창의 역할을 합니다. 뷔는 데이터를 포함하지 않으며 뷔의 정의는 데이터 덕셔너리에 저장됩니다. USER_VIEWS 데이터 덕셔너리 테이블에서 뷔의 정의를 볼 수 있습니다.

뷰의 장점

- 데이터베이스 액세스를 제한합니다.
- 질의를 단순화합니다.
- 데이터 독립성을 제공합니다.
- 동일한 데이터에 대해 다양한 뷔를 제공합니다.
- 기본 데이터에 영향을 주지 않고도 제거할 수 있습니다.

뷰 선택 사항

- 하나의 테이블을 기반으로 하는 단순 뷔를 만들 수 있습니다.
- 둘 이상의 테이블을 기반으로 하는 복합 뷔를 만들 수 있고 함수 그룹을 포함할 수도 있습니다.
- 동일한 이름의 다른 뷔로 대체할 수 있습니다.
- 검색 제약 조건을 포함할 수 있습니다.
- 읽기 전용으로 설정할 수 있습니다.

연습 11 개요

이 연습에서는 다음 내용을 다룹니다.

- 단순 뷰 생성
- 복합 뷰 생성
- 검색 제약 조건을 사용한 뷰 생성
- 뷰를 통한 데이터 수정
- 뷰 정의 표시
- 뷰 제거

ORACLE

11-26

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 11 개요

이 연습에서는 단순 뷰 및 복합 뷰를 생성하여 뷰를 통해 DML 문을 실행합니다.

연습 11

- EMPLOYEES 테이블에서 사원 번호, 사원 이름 및 부서 번호를 기반으로 하는 EMPLOYEES_VU라는 뷰를 생성하십시오. 사원 이름의 머리글을 EMPLOYEE로 변경 하십시오.
- EMPLOYEES_VU 뷰의 내용을 표시하십시오.

EMPLOYEE_ID	EMPLOYEE	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90
103	Hunold	60
104	Ernst	60
107	Lorentz	60
...		
206	Gietz	110

20 rows selected.

- USER_VIEWS 데이터 덕셔너리 뷰에서 뷰 이름 및 텍스트를 선택하십시오.

참고: 다른 뷰가 이미 존재합니다. EMP_DETAILS_VIEW가 스키마의 일부로 생성되었습니다.

참고: LONG 열의 내용을 더 많이 보려면 iSQL*Plus 명령인 SET LONG n을 사용하십시오. 여기서 n은 보려는 LONG 열의 문자 수를 나타내는 값입니다.

VIEW_NAME	TEXT
EMPLOYEES_VU	SELECT employee_id, last_name employee, department_id FROM employees
EMP_DETAILS_VIEW	SELECT e.employee_id, e.job_id, e.manager_id, e.department_id, d.location_id, l.country_id, e.first_name, e.last_name, e.salary, e.commission_pct, d.department_name, j.job_title, l.city, l.state_province, c.country_name, r.region_name FROM employees e, departments d, jobs j, locations l, countries c, regions r WHERE e.department_id = d.department_id AND d.location_id = l.location_id AND l.country_id = c.country_id AND c.region_id = r.region_id AND j.job_id = e.job_id WITH READ ONLY

- EMPLOYEES_VU 뷰를 사용하여 모든 사원의 이름 및 부서 번호를 표시하는 질의를 작성 하십시오.

EMPLOYEE	DEPARTMENT_ID
King	90
Kochhar	90
...	
Gietz	110

20 rows selected.

연습 11(계속)

5. 부서 50의 모든 사원에 대한 사원 번호, 사원 이름 및 부서 번호를 포함하는 DEPT50이라는 뷰를 생성하고 뷰의 열 레이블을 EMPNO, EMPLOYEE 및 DEPTNO로 지정하십시오. 뷰를 통해 한 사원이 다른 부서에 중복 할당되지 않도록 하십시오.
6. DEPT50 뷰의 구조와 내용을 표시하십시오.

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

7. Matos를 부서 80에 다시 할당해 보십시오.

시간이 있을 때 다음 문제를 풀어보십시오.

8. 모든 사원의 이름, 부서 이름, 급여 및 급여 등급을 기반으로 하는 SALARY_VU라는 뷰를 생성하십시오. EMPLOYEES, DEPARTMENTS 및 JOB_GRADES 테이블을 사용하고 열 레이블을 각각 Employee, Department, Salary 및 Grade로 지정하십시오.

12

기타 데이터베이스 객체

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 시퀀스 생성, 유지 관리 및 사용
- 인덱스 생성 및 유지 관리
- 전용(**private**) 동의어 및 공용(**public**) 동의어 생성

ORACLE

12-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 시퀀스, 인덱스, 동의어를 포함하여 일반적으로 사용되는 기타 데이터베이스 객체를 생성 및 유지 관리하는 방법에 대해 설명합니다.

데이터베이스 객체

객체	설명
테이블	기본 저장 단위며 행과 열로 구성됩니다.
뷰	논리적으로 하나 이상의 테이블에 있는 데이터의 부분 집합을 나타냅니다.
시퀀스	기본 키 값을 생성합니다.
인덱스	질의의 성능을 향상시킵니다.
동의어	객체의 다른 이름입니다.

ORACLE

데이터베이스 객체

대부분의 응용 프로그램에서는 고유 번호를 기본 키 값으로 사용해야 합니다. 응용 프로그램에 코드를 작성하거나 시퀀스를 사용하여 고유 번호를 생성할 수 있습니다.

효율적으로 질의하려면 인덱스를 생성해야 하는데 인덱스를 사용하면 열 또는 열 collection의 고유성을 보존할 수도 있습니다.

동의어를 사용하면 객체에 다른 이름을 제공할 수 있습니다.

시퀀스란?

시퀀스는

- 고유 번호를 자동으로 생성합니다.
- 공유 가능한 객체입니다.
- 일반적으로 기본 키 값을 생성하는 데 사용됩니다.
- 응용 프로그램 코드를 대체합니다.
- 시퀀스 값을 메모리에 캐시하면 액세스 효율이 높아집니다.

ORACLE

12-4

Copyright © Oracle Corporation, 2001. All rights reserved.

시퀀스란?

시퀀스는 사용자가 생성하는 데이터베이스 객체로서 여러 사용자가 공유할 수 있으며 고유한 정수를 생성합니다.

시퀀스는 일반적으로 각 행의 고유한 기본 키 값을 생성하는 데 사용되며 오라클 내부 루틴에 의해 생성되거나 증가 또는 감소됩니다. 또한 시퀀스는 시퀀스 생성 루틴 작성에 필요한 응용 프로그램 코드의 양을 줄여주므로 시간 절약 객체입니다.

시퀀스 번호는 테이블과 별도로 저장 및 생성되므로 여러 테이블에 동일한 시퀀스를 사용할 수 있습니다.

CREATE SEQUENCE 문 구문

시퀀스를 정의하여 시퀀스 번호를 자동으로 생성합니다.

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

ORACLE

12-5

Copyright © Oracle Corporation, 2001. All rights reserved.

시퀀스 생성

CREATE SEQUENCE 문을 사용하여 시퀀스 번호를 자동으로 생성합니다.

구문 설명:

sequence	시퀀스 생성기 이름입니다.
INCREMENT BY n	시퀀스 번호 사이의 간격을 지정합니다. 여기서 n은 정수입니다. (이 절을 생략하면 시퀀스는 1씩 증가합니다.)
START WITH n	생성할 첫번째 시퀀스 번호를 지정합니다. (이 절을 생략하면 시퀀스는 1부터 시작됩니다.)
MAXVALUE n	시퀀스가 생성할 수 있는 최대값을 지정합니다.
NOMAXVALUE	오름차순 시퀀스의 최대값으로 10의 27승을 지정하고 내림차순 시퀀스의 최대값으로 -1을 지정합니다(기본 옵션).
MINVALUE n	시퀀스의 최소값을 지정합니다.
NOMINVALUE	오름차순 시퀀스의 최소값으로 1을 지정하고 내림차순 시퀀스의 최소값으로 -10의 26승을 지정합니다(기본 옵션).
CYCLE NOCYCLE	최대값 또는 최소값에 도달한 이후에도 시퀀스가 계속 값을 생성할지 여부를 지정합니다. (NOCYCLE이 기본 옵션입니다.)
CACHE n NOCACHE	Oracle server가 미리 할당하여 메모리에 저장할 값의 개수를 지정합니다. (기본적으로 Oracle server는 20개의 값을 캐시합니다.)

시퀀스 생성

- DEPARTMENTS 테이블의 기본 키로 사용할 DEPT_DEPTID_SEQ 시퀀스를 생성합니다.
- CYCLE 옵션은 사용하지 마십시오.

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

ORACLE

12-6

Copyright © Oracle Corporation, 2001. All rights reserved.

시퀀스 생성(계속)

슬라이드의 예제는 DEPARTMENTS 테이블의 DEPARTMENT_ID 열에 사용할 DEPT_DEPTID_SEQ 시퀀스를 생성합니다. 시퀀스 값은 120에서 시작하며 캐시를 허용하지 않고 순환하지 않습니다.

시퀀스 순환보다 빠르게 이전 행을 지우는 안정적인 처리 방법이 없는 경우에는 시퀀스를 사용하여 기본 키 값을 생성할 때 CYCLE 옵션을 사용하지 마십시오.

자세한 내용은 *Oracle9i SQL Reference*, “CREATE SEQUENCE”를 참조하십시오.

참고: 시퀀스가 하나의 테이블에 전적으로 연결되어 있는 것이 아닙니다. 일반적으로, 시퀀스는 용도에 따라 이름을 정하게 되지만 이름과 상관 없이 어디서나 사용할 수 있습니다.

시퀀스 확인

- * **USER_SEQUENCES** 데이터 딕셔너리 테이블에서 시퀀스 값을 확인합니다.

```
SELECT sequence_name, min_value, max_value,  
       increment_by, last_number  
FROM   user_sequences;
```

- * NOCACHE가 지정된 경우 LAST_NUMBER 열에는 사용 가능한 다음 시퀀스 번호가 표시됩니다.

ORACLE

12-7

Copyright © Oracle Corporation, 2001. All rights reserved.

시퀀스 확인

생성된 시퀀스는 데이터 딕셔너리에 기록됩니다. 시퀀스는 데이터베이스 객체이므로 **USER_OBJECTS** 데이터 딕셔너리 테이블에서 식별할 수 있습니다.

USER_SEQUENCES 데이터 딕셔너리 뷰에서 시퀀스 설정을 선택하여 확인할 수도 있습니다.

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPARTMENTS_SEQ	1	9990	10	280
DEPT_DEPTID_SEQ	1	9999	10	120
EMPLOYEES_SEQ	1	1.0000E+27	1	207
LOCATIONS_SEQ	1	9900	100	3300

NEXTVAL 및 CURRVAL 의사 열

- NEXTVAL은 사용 가능한 다음 시퀀스 값을 반환하며, 참조될 때마다(서로 다른 사용자일지라도) 고유한 값을 반환합니다.
- CURRVAL은 현재 시퀀스 값을 반환합니다.
- CURRVAL이 값을 포함하려면 먼저 해당 시퀀스에 대해 NEXTVAL이 실행되어야 합니다.

ORACLE

12-3

Copyright © Oracle Corporation, 2001. All rights reserved.

시퀀스 사용

시퀀스를 생성하면 시퀀스가 테이블에서 사용할 시퀀스 번호를 생성합니다. NEXTVAL 및 CURRVAL 의사 열을 사용하여 시퀀스 값을 참조합니다.

NEXTVAL 및 CURRVAL 의사 열

NEXTVAL 의사 열은 지정한 시퀀스에서 연속적인 시퀀스 번호를 추출하는 데 사용되며, NEXTVAL에는 시퀀스 이름을 지정해야 합니다. *sequence.NEXTVAL*을 참조하면 새 시퀀스 번호가 생성되며 현재 시퀀스 번호는 CURRVAL에 위치하게 됩니다.

CURRVAL 의사 열은 현재 사용자가 방금 생성한 시퀀스 번호를 참조하는 데 사용됩니다. CURRVAL을 참조하려면 그 전에 NEXTVAL을 사용하여 현재 사용자 세션에서 시퀀스 번호를 생성해야 하며, CURRVAL에는 시퀀스 이름을 지정해야 합니다. *sequence.CURRVAL*을 참조하면 해당 사용자 프로세스에 마지막으로 반환된 값이 표시됩니다.

NEXTVAL 및 CURRVAL 사용 규칙

다음과 같은 경우에 NEXTVAL 및 CURRVAL을 사용할 수 있습니다.

- 서브 쿼리에 속하지 않은 SELECT 문의 SELECT 목록
- INSERT 문에 있는 서브 쿼리의 SELECT 목록
- INSERT 문의 VALUES 절
- UPDATE 문의 SET 절

다음과 같은 경우에는 NEXTVAL 및 CURRVAL을 사용할 수 없습니다.

- 뷰의 SELECT 목록
- DISTINCT 키워드가 있는 SELECT 문
- GROUP BY, HAVING 또는 ORDER BY 절이 있는 SELECT 문
- SELECT, DELETE, UPDATE 문의 서브 쿼리
- CREATE TABLE 또는 ALTER TABLE 문의 DEFAULT 표현식

자세한 내용은 *Oracle9i SQL Reference*, “Pseudocolumns” 단원과 “CREATE SEQUENCE”를 참조하십시오.

시퀀스 사용

- 위치 ID 2500에 “Support”라는 이름의 새 부서를 추가합니다.

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

- DEPT_DEPTID_SEQ 시퀀스의 현재 값을 봅니다.

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

ORACLE

12-10

Copyright © Oracle Corporation, 2001. All rights reserved.

시퀀스 사용

슬라이드의 예제는 새 부서를 DEPARTMENTS 테이블에 삽입하며 새 부서 번호 생성을 위해 DEPT_DEPTID_SEQ 시퀀스를 사용합니다.

시퀀스의 현재 값을 볼 수 있습니다.

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

CURRVAL

120

이제 신입 사원을 고용하여 새 부서에 배치한다고 가정합니다. 신입 사원 모두에 대해 실행될 INSERT 문에 다음 코드를 포함시킬 수 있습니다.

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

참고: 앞의 예제에서는 새 사원 번호를 생성하기 위해 EMPLOYEE_SEQ라는 시퀀스가 이미 생성되었다고 가정합니다.

시퀀스 사용

- 시퀀스 값을 메모리에 캐시하면 해당 값을 더 빠르게 액세스할 수 있습니다.
- 다음과 같은 경우 시퀀스 값 사이에 공백(gap)이 생깁니다.
 - 룰백이 발생하는 경우
 - 시스템이 고장난 경우
 - 시퀀스가 다른 테이블에서 사용되는 경우
- NOCACHE로 시퀀스를 생성한 경우 USER_SEQUENCES 테이블을 질의하여 사용 가능한 다음 값을 볼 수 있습니다.

ORACLE

12-11

Copyright © Oracle Corporation, 2001. All rights reserved.

시퀀스 값 캐시

시퀀스를 메모리에 캐시하면 해당 시퀀스 값을 더 빠르게 액세스할 수 있습니다. 해당 시퀀스를 처음 참조할 때 캐시가 채워지며 다음 시퀀스 값을 요청할 때마다 캐시된 시퀀스에서 값이 검색됩니다. 마지막 시퀀스 값을 사용한 이후에 시퀀스를 요청하면 메모리에 시퀀스의 또 다른 캐시가 만들어집니다.

시퀀스 공백(gap)

시퀀스 생성기는 공백(gap) 없이 차례로 시퀀스 번호를 발행하지만 이 작업은 커밋이나 룰백과는 별도로 이루어집니다. 따라서 시퀀스를 포함하는 명령문을 룰백하면 해당 번호를 잃게 됩니다.

시퀀스에 공백이 생기는 또 다른 원인은 시스템 고장입니다. 시퀀스가 메모리에 값을 캐시한 후에 시스템이 고장나면 해당 값을 잃게 됩니다.

시퀀스가 하나의 테이블에 전적으로 연결되어 있는 것이 아니므로 동일한 시퀀스가 여러 테이블에서 사용될 수 있습니다. 따라서 개별 테이블 내에서는 시퀀스 번호에 공백이 생길 수 있습니다.

시퀀스 값을 증가시키지 않고 사용 가능한 다음 시퀀스 값 보기

NOCACHE를 사용하여 시퀀스를 생성한 경우 USER_SEQUENCES 테이블을 질의하면 시퀀스를 증가시키지 않고 사용 가능한 다음 시퀀스 값을 볼 수 있습니다.

시퀀스 수정

증분 값, 최대값, 최소값, CYCLE 옵션 또는 CACHE 옵션을 변경합니다.

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

ORACLE

12-12

Copyright © Oracle Corporation, 2001. All rights reserved.

시퀀스 변경

시퀀스가 MAXVALUE 한계에 도달하면 해당 시퀀스의 추가 값을 할당할 수 없으며 시퀀스가 MAXVALUE를 초과함을 알리는 오류 메시지가 나타납니다. 시퀀스를 계속 사용하려면 ALTER SEQUENCE 문을 사용하여 수정합니다.

구문

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

구문 설명:

*sequence*는 시퀀스 생성기 이름입니다.

자세한 내용은 *Oracle9i SQL Reference*, “ALTER SEQUENCE”를 참조하십시오.

시퀀스 수정에 대한 지침

- 시퀀스 소유자이거나 시퀀스에 대한 ALTER 권한이 있어야 합니다.
- 이후 시퀀스 번호에만 영향을 줍니다.
- 시퀀스를 다른 번호로 다시 시작하려면 시퀀스를 삭제한 후 다시 생성해야 합니다.
- 일부 검증이 수행됩니다.

ORACLE

12-13

Copyright © Oracle Corporation, 2001. All rights reserved.

시퀀스 수정에 대한 지침

- 시퀀스를 수정하려면 시퀀스 소유자이거나 시퀀스에 대한 ALTER 권한이 있어야 합니다.
- ALTER SEQUENCE 문은 이후 시퀀스 번호에만 영향을 줍니다.
- START WITH 옵션은 ALTER SEQUENCE를 사용하여 변경할 수 없습니다. 시퀀스를 다른 번호로 다시 시작하려면 시퀀스를 삭제한 후 다시 생성해야 합니다.
- 일부 검증이 수행됩니다. 예를 들어, 새로운 MAXVALUE 값을 현재 시퀀스 번호보다 작게 지정할 수 없습니다.

```
ALTER SEQUENCE dept_deptid_seq
    INCREMENT BY 20
    MAXVALUE 90
    NOCACHE
    NOCYCLE;
ALTER SEQUENCE dept_deptid_seq
*
ERROR at line 1:
ORA-04009: MAXVALUE cannot be made to be less than the current
           value
```

시퀀스 제거

- **DROP SEQUENCE** 문을 사용하여 데이터 딕셔너리에서 시퀀스를 제거합니다.
- 제거된 시퀀스는 더 이상 참조할 수 없습니다.

```
DROP SEQUENCE dept_deptid_seq;
Sequence dropped.
```

ORACLE

12-14

Copyright © Oracle Corporation, 2001. All rights reserved.

시퀀스 제거

데이터 딕셔너리에서 시퀀스를 제거하려면 **DROP SEQUENCE** 문을 사용하십시오. 시퀀스를 제거하려면 시퀀스 소유자이거나 **DROP ANY SEQUENCE** 권한이 있어야 합니다.

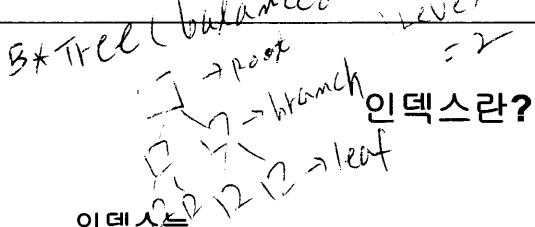
구문

```
DROP SEQUENCE sequence;
```

구문 설명:

*sequence*는 시퀀스 생성기 이름입니다.

자세한 내용은 *Oracle9i SQL Reference*, “**DROP SEQUENCE**”를 참조하십시오.



인덱스는

- 스키마 객체입니다.
- Oracle server에서 포인터를 사용하여 행 검색 속도를 높이기 위해 사용합니다.
- 데이터 위치를 빠르게 찾는 신속한 경로 액세스 방법을 사용하여 디스크 I/O를 줄여 줍니다.
- 인덱스화된 테이블과 독립되어 존재합니다.
- Oracle server에 의해 사용되며 자동으로 유지 관리됩니다.

ORACLE

인덱스

Oracle server 인덱스는 포인터를 사용하여 행 검색 속도를 높일 수 있는 스키마 객체입니다. 인덱스는 명시적(explicit)으로 또는 자동으로 생성할 수 있으며 열에 인덱스가 없는 경우에는 테이블 전체가 스캔됩니다.

인덱스는 테이블의 행에 대해 직접적이고 빠른 액세스를 제공하며 인덱스화된 경로를 사용하여 데이터 위치를 빠르게 찾음으로써 디스크 I/O를 줄여 줍니다. 인덱스는 Oracle server에 의해 사용되며 자동으로 유지 관리됩니다. 인덱스가 생성되면 사용자가 인덱스에 직접적인 작업을 할 필요가 없습니다.

인덱스는 논리적으로 또는 물리적으로 인덱스화된 테이블과 독립되어 존재합니다. 즉 인덱스는 언제든지 생성 또는 삭제할 수 있으며 기본 테이블이나 기타 인덱스에 아무런 영향도 주지 않습니다.

참고: 테이블을 삭제하면 해당하는 인덱스도 삭제됩니다.

자세한 내용은 *Oracle9i Concepts*, “Schema Objects” 단원의 “Indexes” 항목을 참조하십시오.

인덱스 생성 방법

- 자동: 테이블 정의에 PRIMARY KEY 또는 UNIQUE 제약 조건을 정의하면 고유 인덱스가 자동으로 생성됩니다.
- 수동: 사용자가 열에 고유하지 않은 인덱스를 생성하여 행에 대한 액세스 시간을 줄일 수 있습니다.

ORACLE

12-16

Copyright © Oracle Corporation, 2001. All rights reserved.

인덱스 유형

두 가지 유형의 인덱스를 생성할 수 있는데 그 중 하나가 고유 인덱스입니다. 테이블의 열이 PRIMARY KEY 또는 UNIQUE 키 제약 조건을 포함하도록 정의하면 Oracle server는 이러한 고유 인덱스를 자동으로 생성합니다. 인덱스의 이름은 제약 조건에 지정한 이름과 동일합니다.

사용자가 생성할 수 있는 또 하나의 인덱스 유형은 고유하지 않은 인덱스입니다. 예를 들어, 질의에서 조인할 FOREIGN KEY 열 인덱스를 생성하여 검색 속도를 향상시킬 수 있습니다.

참고: 고유 인덱스를 수동으로 생성할 수 있지만 고유 제약 조건을 생성하여 암시적(implicit)으로 고유 인덱스를 생성하는 것이 좋습니다.

인덱스 생성

- 하나 이상의 열에 대해 인덱스를 생성합니다.

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

- EMPLOYEES 테이블의 LAST_NAME 열에 대한 질의 액세스 속도를 향상시킵니다.

```
CREATE INDEX emp_last_name_idx  
ON employees(last_name);  
Index created.
```

ORACLE

12-17

Copyright © Oracle Corporation, 2001. All rights reserved.

인덱스 생성

CREATE INDEX 문을 실행하여 하나 이상의 열에 대해 인덱스를 생성합니다.

구문 설명:

index	인덱스 이름입니다.
table	테이블 이름입니다.
column	인덱스화될 테이블의 열 이름입니다.

자세한 내용은 *Oracle9i SQL Reference*, “CREATE INDEX”를 참조하십시오.

인덱스 생성이 필요한 경우

다음의 경우에 인덱스를 생성해야 합니다.

- 열에 광범위한 값이 포함된 경우
- 열에 널 값이 많이 포함된 경우
- WHERE 절 또는 조인 조건에서 하나 이상의 열이 함께 자주 사용되는 경우
- 큰 테이블에서 대부분의 질의에 의해 검색되는 행이 2%-4% 미만인 경우

ORACLE

12-18

Copyright © Oracle Corporation, 2001. All rights reserved.

인덱스가 많을수록 좋은 것은 아닙니다

테이블에 인덱스가 많을수록 질의 속도가 빨라지는 것은 아닙니다. 인덱스를 포함하는 테이블에서는 DML 작업이 커밋될 때마다 해당 인덱스도 갱신되어야 합니다. 따라서 테이블과 연관된 인덱스가 많을수록 Oracle server는 DML 작업 후 더 많은 인덱스를 갱신해야 합니다.

인덱스 생성이 필요한 경우

따라서 다음의 경우에만 인덱스를 생성해야 합니다.

- 열에 광범위한 값이 포함된 경우
- 열에 널 값이 많이 포함된 경우
- WHERE 절 또는 조인 조건에서 하나 이상의 열이 함께 자주 사용되는 경우
- 큰 테이블에서 대부분의 질의에 의해 검색되는 행이 2%-4% 미만인 경우

고유성을 보존하려면 테이블 정의에 고유 제약 조건을 정의해야 합니다. 그러면 고유 인덱스가 자동으로 생성됩니다.

인덱스를 생성하지 않아야 할 경우

다음의 경우에는 인덱스를 생성하지 않는 것이 좋습니다.

- 테이블이 작은 경우
- 열이 질의의 조건으로 자주 사용되지 않는 경우
- 대부분의 질의가 테이블에 있는 행의 **2%-4%** 이상을 검색할 경우
- 테이블이 자주 갱신되는 경우 → DML 속도↓ : *성능 저하로 때문*
- 인덱스화된 열이 표현식의 일부로 참조되는 경우

ORACLE

12-19

Copyright © Oracle Corporation, 2001. All rights reserved.

인덱스 확인

- USER_INDEXES 데이터 딕셔너리 뷰는 인덱스 이름 및 고유성을 포함합니다.
- USER_IND_COLUMNS 뷰는 인덱스 이름, 테이블 이름 및 열 이름을 포함합니다.

```
SELECT    ic.index_name, ic.column_name,
          ic.column_position col_pos, ix.uniqueness
FROM      user indexes ix, user ind columns ic
WHERE     ic.index_name = ix.index_name
AND       ic.table_name = 'EMPLOYEES';
```

ORACLE

12-28

Copyright © Oracle Corporation, 2001. All rights reserved.

인덱스 확인

USER_INDEXES 데이터 딕셔너리 뷰에서 인덱스의 존재를 확인할 수 있습니다.

USER_IND_COLUMNS 뷰를 질의하여 인덱스와 관련된 열을 확인할 수도 있습니다.

슬라이드의 예제는 EMPLOYEES 테이블에서 이전에 생성된 모든 인덱스, 관련된 열 이름 및 인덱스의 고유성을 나타냅니다.

INDEX_NAME	COLUMN_NAME	COL_POS	UNIQUENESS
EMP_EMAIL_UK	EMAIL	1	UNIQUE
EMP_EMP_ID_PK	EMPLOYEE_ID	1	UNIQUE
EMP_DEPARTMENT_IX	DEPARTMENT_ID	1	NONUNIQUE
EMP_JOB_IX	JOB_ID	1	NONUNIQUE
EMP_MANAGER_IX	MANAGER_ID	1	NONUNIQUE
EMP_NAME_IX	LAST_NAME	1	NONUNIQUE
EMP_NAME_IX	FIRST_NAME	2	NONUNIQUE
EMP_LAST_NAME_IDX	LAST_NAME	1	NONUNIQUE

8 rows selected.

함수 기반 인덱스

- 함수 기반 인덱스는 표현식을 기반으로 하는 인덱스입니다.
- 인덱스 표현식은 테이블 열, 상수, SQL 함수 및 사용자 정의한 함수로부터 생성됩니다.

```
CREATE INDEX upper_dept_name_idx  
ON departments(UPPER(department_name)) ;
```

```
Index created.
```

```
SELECT *  
FROM   departments  
WHERE  UPPER(department_name) = 'SALES' ;
```

ORACLE

12-21

Copyright © Oracle Corporation, 2001. All rights reserved.

함수 기반 인덱스

UPPER(column_name) 또는 LOWER(column_name) 키워드와 함께 정의된 함수 기반 인덱스를 사용하면 대소문자 구분 없이 검색할 수 있습니다. 다음과 같은 인덱스를 예로 들 수 있습니다.

```
CREATE INDEX upper_last_name_idx ON employees (UPPER(last_name)) ;
```

이 인덱스는 다음과 같은 질의를 쉽게 처리할 수 있도록 해줍니다.

```
SELECT * FROM employees WHERE UPPER(last_name) = 'KING' ;
```

Oracle server가 전체 테이블을 스캔하는 대신 인덱스를 사용하도록 만들려면 후속 질의의 함수 값이 널이 아니어야 합니다. 예를 들어, 다음 명령문은 인덱스를 사용하도록 되어 있지만 WHERE 절이 없으면 Oracle server가 전체 테이블을 스캔합니다.

```
SELECT  *  
FROM    employees  
WHERE   UPPER (last_name) IS NOT NULL  
ORDER BY UPPER (last_name) ;
```

함수 기반 인덱스(계속)

Oracle server는 열에 DESC가 표시된 인덱스를 함수 기반 인덱스로 처리합니다. DESC가 표시된 열은 내림차순으로 정렬됩니다.

인덱스 제거

- **DROP INDEX** 명령을 사용하여 데이터 딕셔너리에서 인덱스를 제거합니다.

```
DROP INDEX index;
```

- 데이터 딕셔너리에서 **UPPER_LAST_NAME_IDX** 인덱스를 제거합니다.

```
DROP INDEX upper_last_name_idx;  
Index dropped.
```

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

ORACLE

12-23

Copyright © Oracle Corporation, 2001. All rights reserved.

인덱스 제거

인덱스는 수정할 수 없습니다. 인덱스를 변경하려면 삭제한 후 다시 생성해야 합니다. **DROP INDEX** 문을 실행하여 데이터 딕셔너리에서 인덱스 정의를 제거합니다. 인덱스를 삭제하려면 인덱스 소유자이거나 **DROP ANY INDEX** 권한이 있어야 합니다.

구문 설명:

index 인덱스 이름입니다.

참고: 테이블을 삭제할 경우 인덱스와 제약 조건은 자동으로 삭제되지만 뷰와 시퀀스는 그대로 남습니다.

동의어

동의어(액체의 다른 이름)를 생성하여 객체 액세스를 단순화 합니다. 동의어를 사용하여 다음을 수행할 수 있습니다.

- 다른 사용자가 소유한 테이블을 쉽게 참조합니다.
- 긴 객체 이름을 짧게 만듭니다.

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

ORACLE

12-24

Copyright © Oracle Corporation, 2001. All rights reserved.

객체에 대한 동의어 생성

다른 사용자가 소유한 테이블을 참조하려면 테이블 이름에 해당 테이블을 생성한 사용자의 이름과 마침표를 접두어로 붙여야 합니다. 동의어를 생성하면 객체 이름을 스키마와 함께 지정할 필요가 없으며 테이블, 뷰, 시퀀스, 프로시저 및 기타 객체에 대해 다른 이름을 사용할 수 있습니다. 이 방법은 특히 뷰와 같은 긴 객체 이름을 사용할 때 유용합니다.

구문 설명:

PUBLIC	모든 사용자가 액세스할 수 있는 동의어를 생성합니다.
<i>synonym</i>	생성될 동의어의 이름입니다.
<i>object</i>	동의어가 생성될 객체입니다.

지침

- 객체는 패키지에 포함시킬 수 없습니다.
- 전용(private) 동의어 이름은 동일한 사용자가 소유한 다른 모든 객체와 구별되어야 합니다.

자세한 내용은 *Oracle9i SQL Reference*, “CREATE SYNONYM”을 참조하십시오.

동의어 생성 및 제거

- * DEPT_SUM_VU 뷰의 간략한 이름을 생성합니다.

```
CREATE SYNONYM d_sum  
FOR dept_sum_vu;  
Synonym Created.
```

- * 동의어를 삭제합니다.

```
DROP SYNONYM d_sum;  
Synonym dropped.
```

12-25

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

액체에 대한 동의어 생성(계속)

슬라이드의 예제는 더 빠른 참조를 위해 DEPT_SUM_VU 뷰의 동의어를 생성합니다.

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

```
CREATE PUBLIC SYNONYM dept  
FOR alice.departments;  
Synonym created.
```

동의어 제거

동의어를 삭제하려면 DROP SYNONYM 문을 사용합니다. 데이터베이스 관리자만이 공용(public)
동의어를 삭제할 수 있습니다.

```
DROP PUBLIC SYNONYM dept;  
Synonym dropped.
```

자세한 내용은 *Oracle9i SQL Reference*, “DROP SYNONYM”을 참조하십시오.

요약

이 단원에서는 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- 시퀀스 생성기를 사용한 시퀀스 번호 자동 생성
- **USER_SEQUENCES** 데이터 딕셔너리 테이블에서 시퀀스 정보 보기
- 인덱스를 생성하여 질의 검색 속도 향상시키기
- **USER_INDEXES** 딕셔너리 테이블에서 인덱스 정보 보기
- 동의어를 사용하여 객체에 다른 이름 제공

ORACLE

12-26

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

이 단원에서는 시퀀스, 인덱스 및 뷰를 포함한 다른 데이터베이스 객체에 대해 설명했습니다.

시퀀스

시퀀스 생성기를 사용하여 테이블의 행에 대한 시퀀스 번호를 자동으로 생성함으로써 시간을 절약하고 응용 프로그램 코드의 양도 줄일 수 있습니다.

시퀀스는 다른 사용자와 공유할 수 있는 데이터베이스 객체입니다. 시퀀스에 대한 정보는 데이터 딕셔너리의 **USER_SEQUENCES** 테이블을 참조하십시오.

시퀀스를 사용하려면 **NEXTVAL** 또는 **CURRVAL** 의사 열을 사용하여 참조하십시오.

- *sequence.NEXTVAL*을 참조하여 시퀀스에서 다음 번호를 검색합니다.
- *sequence.CURRVAL*을 참조하여 현재 사용 가능한 번호를 반환합니다.

인덱스

인덱스는 질의 검색 속도를 향상시키는데 사용됩니다. 인덱스의 정의는 **USER_INDEXES** 테이터 딕셔너리 뷰에서 볼 수 있습니다. 인덱스는 해당 인덱스를 생성한 사람이나 **DROP ANY INDEX** 권한을 가진 사람이 **DROP INDEX** 문을 사용하여 삭제할 수 있습니다.

동의어

CREATE SYNONYM 문을 사용하여 데이터베이스 관리자는 공용(**public**) 동의어를 생성할 수 있고 사용자는 편의를 위해 전용(**private**) 동의어를 생성할 수 있습니다. 동의어를 사용하면 객체에 대해 간략한 이름이나 다른 이름을 부여할 수 있습니다. **DROP SYNONYM** 문을 사용하여 동의어를 제거합니다.

연습 12 개요

이 연습에서는 다음 내용을 다룹니다.

- 시퀀스 생성
- 시퀀스 사용
- 고유하지 않은 인덱스 생성
- 시퀀스 및 인덱스에 대한 데이터 딕셔너리 정보 표시
- 인덱스 삭제

ORACLE

12-27

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 12 개요

이 연습에서는 테이블에 데이터를 채울 때 사용되는 시퀀스를 생성하며 암시적(implicit) 및 명시적(explicit) 인덱스도 생성합니다.

연습 12

- DEPT 테이블의 기본 키 열에 사용할 시퀀스를 생성하십시오. 시퀀스 값은 200부터 시작하여 10씩 증가하며 최대 1000까지 가능합니다. 시퀀스 이름은 DEPT_ID_SEQ로 지정하십시오.
- 시퀀스 이름, 최대값, 증가분, 마지막 번호와 같은 시퀀스 정보를 표시하는 질의를 스크립트로 작성하여 lab12_2.sql이라는 이름을 지정한 후 스크립트의 명령문을 실행하십시오.

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPARTMENTS_SEQ	9990	10	280
DEPT_ID_SEQ	1000	10	200
EMPLOYEES_SEQ	1.0000E+27	1	207
LOCATIONS_SEQ	9900	100	3300

- DEPT 테이블에 두 행을 삽입하는 스크립트를 작성하고 이름을 lab12_3.sql로 지정하십시오. ID 열에 대해서는 이전에 생성한 시퀀스를 사용하십시오. Education 및 Administration이라는 두 개의 부서를 추가하고 결과를 확인한 후 스크립트의 명령을 실행하십시오.
- EMP 테이블의 외래 키 열(DEPT_ID)에 대해 비고유 인덱스를 생성하십시오.
- 데이터 덕셔너리에 있는 EMP 테이블의 인덱스 및 고유성을 표시하십시오. 명령문을 lab12_5.sql이라는 스크립트에 저장하십시오.

INDEX_NAME	TABLE_NAME	UNIQUENESS
EMP_DEPT_ID_IDX	EMP	NONUNIQUE
MY_EMP_ID_PK	EMP	UNIQUE

13

사용자 액세스 제어

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 사용자 생성
- 보안 모델 설정 및 유지 관리를 용이하게 해주는 를 생성
- GRANT 및 REVOKE 문을 사용하여 객체 권한 부여 및 취소
- 데이터베이스 링크 생성 및 액세스

ORACLE

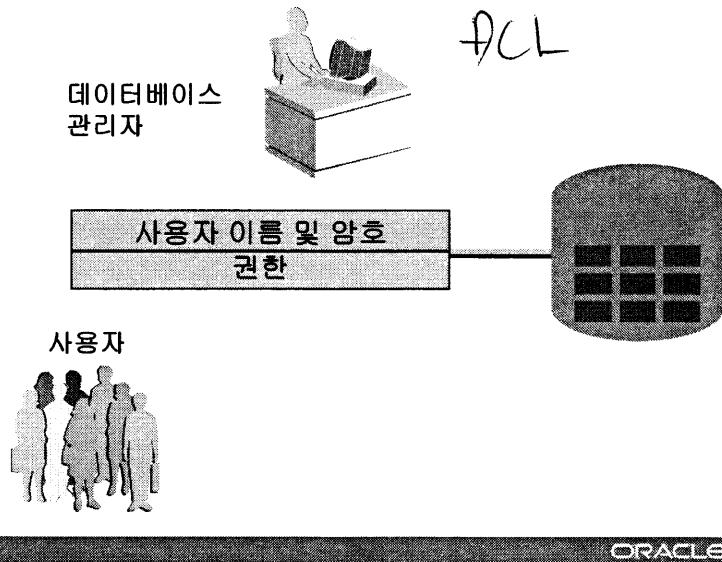
13-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 특정 객체에 대한 데이터베이스 액세스를 제어하는 방법과 서로 다른 레벨의 액세스 권한을 가진 새 사용자를 추가하는 방법을 설명합니다.

사용자 액세스 제어



13-3

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

사용자 액세스 제어

다중 사용자 환경에서는 데이터베이스 액세스 및 사용에 대한 보안을 유지 관리해야 합니다. Oracle server 데이터베이스 보안을 사용하면 다음을 수행할 수 있습니다.

- 데이터베이스 액세스 제어
- 데이터베이스의 특정 객체에 대한 액세스 권한 부여
- 오라클 데이터 딕셔너리에서 주고 받은 권한 확인
- 데이터베이스 객체의 동의어 생성

데이터베이스 보안은 시스템 보안 및 데이터 보안의 두 가지 범주로 나눌 수 있습니다. 시스템 보안은 사용자 이름 및 암호, 사용자에게 할당된 디스크 공간, 사용자가 수행할 수 있는 시스템 작업 등 시스템 레벨의 데이터베이스 액세스 및 사용을 다루고, 데이터 보안은 데이터베이스 객체에 대한 액세스와 사용 및 각 객체에 대해 사용자가 수행할 수 있는 작업을 다룹니다.

권한

- 데이터베이스 보안:
 - 시스템 보안
 - 데이터 보안
- 시스템 권한: 데이터베이스를 액세스할 수 있습니다.
- 객체 권한: 데이터베이스 객체의 내용을 조작할 수 있습니다.
- 스키마: 테이블, 뷰, 시퀀스 등과 같은 객체의 **Collection**입니다.

schema = oracle user

ORACLE

권한

권한은 특정 SQL 문을 실행할 수 있는 권리입니다. DBA(데이터베이스 관리자)는 사용자에게 데이터베이스 및 데이터베이스 객체에 대한 액세스 권한을 부여할 수 있는 상위 레벨 사용자입니다. 사용자는 데이터베이스에 액세스할 수 있는 시스템 권한 및 데이터베이스 객체의 내용을 조작할 수 있는 객체 권한이 필요합니다. 또한 사용자는 다른 사용자 또는 그룹(관련 권한을 따로 분류하여 명명한 그룹)에 추가 권한을 부여할 수 있는 권한을 부여 받을 수도 있습니다.

스키마

스키마는 테이블, 뷰, 시퀀스 등과 같은 여러 객체의 collection으로서 데이터베이스 사용자가 소유하며 해당 사용자와 동일한 이름을 갖습니다.

자세한 내용은 *Oracle9i Application Developer's Guide - Fundamentals*, “Establishing a Security Policy” 단원 및 *Oracle9i Concepts*, “Database Security” 항목을 참조하십시오.

시스템 권한

- 100가지 이상의 권한을 사용할 수 있습니다.
- 데이터베이스 관리자는 다음과 같은 작업에 대해 상위 레벨의 시스템 권한을 가집니다.
 - 새 사용자 생성
 - 사용자 제거
 - 테이블 제거
 - 테이블 백업

ORACLE

13-5

Copyright © Oracle Corporation, 2001. All rights reserved.

시스템 권한

사용자 및 틀에 대해 사용할 수 있는 고유 시스템 권한은 100가지가 넘는데 이 권한은 대부분 데이터베이스 관리자가 제공합니다.

일반적인 DBA 권한

시스템 권한	승인된 작업
CREATE USER	다른 오라클 사용자를 생성할 수 있습니다(DBA 틀에 필요한 권한).
DROP USER	다른 사용자를 삭제할 수 있습니다.
DROP ANY TABLE	스키마에 있는 테이블을 삭제할 수 있습니다.
BACKUP ANY TABLE	Export 유ти리티로 스키마에 있는 테이블을 백업할 수 있습니다.
SELECT ANY TABLE	스키마에 있는 테이블, 뷰 또는 스냅샷을 질의할 수 있습니다.
CREATE ANY TABLE	스키마에 테이블을 생성할 수 있습니다.

사용자 생성

DBA는 CREATE USER 문을 사용하여 사용자를 생성합니다.

```
CREATE USER user  
IDENTIFIED BY password;
```

```
CREATE USER scott  
IDENTIFIED BY tiger;  
User created.
```

ORACLE

13-6

Copyright © Oracle Corporation, 2001. All rights reserved.

사용자 생성

DBA는 CREATE USER 문을 실행하여 사용자를 생성하는데 처음 생성된 사용자는 아무런 권한도 갖지 않습니다. DBA는 해당 사용자에게 권한을 부여할 수 있는데 이 때 부여된 권한에 따라 사용자가 데이터베이스 레벨에서 수행할 수 있는 작업이 결정됩니다.

슬라이드는 사용자 생성 구문을 축약하여 나타낸 것입니다.

구문 설명:

user	생성될 사용자 이름입니다.
password	사용자가 로그인 할 때 사용할 암호입니다.

자세한 내용은 *Oracle9i SQL Reference*, “GRANT” 및 “CREATE USER”를 참조하십시오.

사용자 시스템 권한

- DBA는 생성된 사용자에게 특정 시스템 권한을 부여할 수 있습니다.

```
GRANT privilege [, privilege...]
TO user [, user| role, PUBLIC...];
```

- 응용 프로그램 개발자는 예를 들어 다음과 같은 시스템 권한을 갖습니다.
 - CREATE SESSION
 - CREATE TABLE
 - CREATE SEQUENCE
 - CREATE VIEW
 - CREATE PROCEDURE

ORACLE

三

Copyright © Oracle Corporation 2001. All rights reserved.

일반적인 사용자 권한

DBA는 생성된 사용자에게 권한을 할당할 수 있습니다.

시스템 권한	승인된 작업
CREATE SESSION	데이터베이스에 연결합니다.
CREATE TABLE	사용자의 스키마에 테이블을 생성합니다.
CREATE SEQUENCE	사용자의 스키마에 시퀀스를 생성합니다.
CREATE VIEW	사용자의 스키마에 뷰를 생성합니다.
CREATE PROCEDURE	사용자의 스키마에 내장 프로시저, 함수 또는 패키지를 작성합니다.

구문 설명

privilege 부여할 시스템 권한입니다.

user | role | PUBLIC 사용자 이름 또는 룰 이름을 지정할 수도 있고 모든 사용자에게 권한이 부여되었음을 나타내는 PUBLIC을 지정할 수도 있습니다.

참고: 현재 시스템 권한은 딕셔너리 뷰 SESSION_PRIVS에서 볼 수 있습니다.

시스템 권한 부여

DBA는 사용자에게 특정 시스템 권한을 부여할 수 있습니다.

```
GRANT create session, create table,  
       create sequence, create view  
TO      scott;  
Grant succeeded.
```

ORACLE

13-8

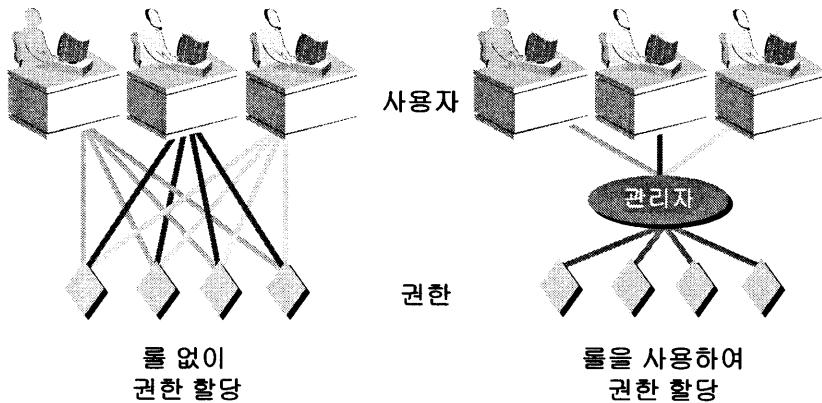
Copyright © Oracle Corporation, 2001. All rights reserved.

시스템 권한 부여

DBA는 GRANT 문을 사용하여 사용자에게 시스템 권한을 할당하며 권한을 부여 받은 사용자는 해당 권한을 곧바로 사용할 수 있습니다.

슬라이드의 예제에서는 사용자 Scott에게 세션, 테이블, 시퀀스 및 뷰를 생성할 수 있는 권한이 할당되었습니다.

룰이란?



13-9

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

룰이란?

룰은 사용자에게 부여할 수 있는 관련 권한을 하나로 묶어 명명한 그룹으로서 룰을 사용하면 권한 취소 및 유지 관리를 쉽게 수행할 수 있습니다.

사용자는 여러 룰을 액세스할 수 있으며 동일한 룰이 여러 사용자에게 할당될 수도 있습니다. 일반적으로 데이터베이스 응용 프로그램에 대해 룰을 생성합니다.

룰 생성 및 할당

DBA가 룰을 생성한 후 룰에 특정 권한 및 사용자를 할당합니다.

구문

```
CREATE ROLE role;
```

구문 설명:

role 생성할 룰 이름입니다.

룰을 생성한 후 DBA는 GRANT 문을 사용하여 룰에 권한 및 사용자를 할당할 수 있습니다.

룰 생성 및 권한 부여

- 룰을 생성합니다.

```
CREATE ROLE manager;  
Role created.
```

- 룰에 권한을 부여합니다.

```
GRANT create table, create view  
TO manager;  
Grant succeeded.
```

- 사용자에게 룰을 부여합니다.

```
GRANT manager TO DEHAAN, KOCHHAR;  
Grant succeeded.
```

ORACLE

룰 생성

슬라이드의 예제는 **manager** 를을 생성한 후 **manager** 를에 테이블 및 뷰 생성 권한을 부여합니다. 그런 다음 DeHaan 및 Kochhar에게 **manager** 를을 부여하여 테이블과 뷰를 생성할 수 있도록 합니다.

사용자에게 부여된 룰이 여러 개일 경우 해당 사용자는 부여된 룰과 관련된 권한을 모두 받게 됩니다.

암호 변경

- DBA는 사용자 계정을 생성하고 암호를 초기화합니다.
- ALTER USER 문을 사용하여 암호를 변경할 수 있습니다.

```
ALTER USER scott  
IDENTIFIED BY lion;  
User altered.
```

암호 변경

DBA는 모든 사용자의 계정을 생성하고 암호를 초기화합니다. 사용자는 ALTER USER 문을 사용하여 암호를 변경할 수 있습니다.

구문

```
ALTER USER user IDENTIFIED BY password;
```

구문 설명:

user 사용자 이름입니다.

password 새 암호를 지정합니다.

ALTER USER 문을 사용하면 암호 이외에도 여러 가지 옵션을 변경할 수 있지만 ALTER USER 권한이 있어야 합니다.

자세한 내용은 *Oracle9i SQL Reference*, “ALTER USER”를 참조하십시오.

객체 권한

객체 권한	테이블	뷰	시퀀스	프로시저
ALTER	✓		✓	
DELETE	✓	✓		
EXECUTE				✓
INDEX	✓			
INSERT	✓	✓		
REFERENCES	✓	✓		
SELECT	✓	✓	✓	
UPDATE	✓	✓		

ORACLE

13-12

Copyright © Oracle Corporation, 2001. All rights reserved.

객체 권한

객체 권한은 특정 테이블, 뷰, 시퀀스 또는 프로시저에 대해 특정 작업을 수행할 수 있는 권한이며 각 객체는 부여할 수 있는 특정 권한 집합을 포함합니다. 슬라이드의 테이블은 다양한 객체에 대한 권한을 나열합니다. 시퀀스에는 SELECT 및 ALTER 권한만 적용할 수 있습니다. 생성 가능한 일부 열을 지정하여 UPDATE, REFERENCES 및 INSERT 작업을 제한할 수 있으며, 일부 열을 사용하여 뷰를 생성하고 해당 뷰에만 SELECT 권한을 부여해서 SELECT 작업을 제한할 수 있습니다. 동의어에 부여된 권한은 해당 동의어가 참조하는 기본 테이블에 대한 권한으로 변환됩니다.

객체 권한

- 객체 권한은 객체마다 다릅니다.
- 소유자는 객체에 대한 모든 권한을 갖습니다.
- 소유자는 자신의 객체에 대한 특정 권한을 부여할 수 있습니다.

```
GRANT      object_priv [(columns)]
ON         object
TO         {user|role|PUBLIC}
[WITH GRANT OPTION];
```

13-13

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

객체 권한 부여

다른 유형의 스키마 객체에 대해 여러 가지 객체 권한을 사용할 수 있습니다. 사용자는 자동으로 자신의 스키마에 포함된 스키마 객체에 대한 모든 객체 권한을 가지며 또한 자신이 소유한 스키마 객체에 대한 객체 권한을 다른 사용자나 룰에 부여할 수 있습니다. 권한을 부여할 때 WITH GRANT OPTION 절을 포함시키면 권한을 부여 받은 사용자가 해당 객체 권한을 다른 사용자에게 부여할 수 있지만, 이 옵션을 포함시키지 않으면 권한을 부여 받은 사용자가 해당 권한을 사용할 수는 있지만 다른 사용자에게 부여할 수는 없습니다.

구문 설명:

<i>object_priv</i>	부여할 객체 권한입니다.
<i>ALL</i>	모든 객체 권한을 지정합니다.
<i>columns</i>	권한을 부여할 테이블 또는 뷰의 열을 지정합니다.
<i>ON object</i>	권한을 부여할 객체입니다.
<i>TO</i>	권한을 부여 받을 사용자입니다.
<i>PUBLIC</i>	객체 권한을 모든 사용자에게 부여합니다.
<i>WITH GRANT OPTION</i>	권한을 부여 받은 사용자가 해당 객체 권한을 다른 사용자 및 룰에 부여할 수 있도록 허용합니다.

객체 권한 부여

- EMPLOYEES 테이블에 대한 질의 권한을 부여합니다.

```
GRANT select  
ON   employees  
TO   sue, rich;  
Grant succeeded.
```

- 사용자 및 틀에 특정 열을 갱신할 수 있는 권한을 부여 합니다.

```
GRANT update (department_name, location_id)  
ON   departments  
TO   scott, manager;  
Grant succeeded.
```

ORACLE

13-14

Copyright © Oracle Corporation, 2001. All rights reserved.

지침

- 사용자가 객체에 대한 권한을 다른 사용자나 틀에 부여하려면 해당 객체가 사용자 자신의 스키마에 있거나 해당 객체 권한을 WITH GRANT OPTION 절과 함께 부여 받았어야 합니다.
- 객체 소유자는 객체에 대한 모든 객체 권한을 데이터베이스 내의 다른 사용자나 틀에 부여 할 수 있습니다.
- 객체 소유자는 자신이 소유한 객체에 대한 모든 객체 권한을 자동으로 갖게 됩니다.

슬라이드의 첫번째 예제는 사용자 Sue와 Rich에게 EMPLOYEES 테이블을 질의할 수 있는 권한을 부여하며, 두번째 예제는 Scott와 manager 틀에 DEPARTMENTS 테이블의 특정 열에 대한 UPDATE 권한을 부여합니다.

Sue 또는 Rich가 EMPLOYEES 테이블에서 데이터를 선택할 경우에는 다음 구문을 사용해야 합니다.

```
SELECT *  
FROM   scott.employees;
```

또는 테이블의 동의어를 생성하여 이 동의어를 FROM 절에 지정할 수도 있습니다.

```
CREATE SYNONYM emp FOR scott.employees;  
SELECT * FROM emp;
```

참고: 일반적으로 DBA는 시스템 권한을 할당하고 객체 소유자는 객체 권한을 부여합니다.

WITH GRANT OPTION 및 PUBLIC 키워드 사용

- 사용자에게 권한을 부여할 수 있는 권한(authority)을 제공합니다.

```
GRANT select, insert  
ON departments  
TO scott  
WITH GRANT OPTION;  
Grant succeeded.
```



- 시스템의 모든 사용자가 Alice의 DEPARTMENTS 테이블에 있는 데이터를 질의할 수 있도록 합니다.

```
GRANT select  
ON alice.departments  
TO PUBLIC; → 모든 사용자가  
Grant succeeded.
```

ORACLE

13-15

Copyright © Oracle Corporation, 2001. All rights reserved.

WITH GRANT OPTION 키워드

WITH GRANT OPTION 절을 통해 권한을 부여 받은 사용자는 해당 권한을 다른 사용자 및 테이블에 전달할 수 있는데 권한 부여자의 권한이 취소되면 WITH GRANT OPTION 절을 통해 부여된 객체 권한도 취소됩니다.

슬라이드의 예제는 사용자 Scott에게 DEPARTMENTS 테이블에 대한 액세스 권한 및 해당 테이블을 질의하고 테이블에 행을 추가할 수 있는 권한을 부여하며 또한 다른 사용자에게 동일한 권한을 부여할 수 있는 권한도 부여합니다.

PUBLIC 키워드

테이블의 소유자는 PUBLIC 키워드를 사용하여 모든 사용자에게 액세스 권한을 부여할 수 있습니다.

두번째 예제는 시스템의 모든 사용자가 Alice의 DEPARTMENTS 테이블에 있는 데이터를 질의할 수 있도록 합니다.

부여된 권한 확인

데이터 딕셔너리 뷰	설명
ROLE_SYS_PRIVS	롤에 부여된 시스템 권한입니다.
ROLE_TAB_PRIVS	롤에 부여된 테이블 권한입니다.
USER_ROLE_PRIVS	사용자가 액세스할 수 있는 룰입니다.
USER_TAB_PRIVS_MADE	사용자 객체에 대해 부여된 객체 권한입니다.
USER_TAB_PRIVS_RECV	사용자에게 부여된 객체 권한입니다.
USER_COL_PRIVS_MADE	사용자 객체의 열에 대해 부여된 객체 권한입니다.
USER_COL_PRIVS_RECV	특정 열에 대해 사용자에게 부여된 객체 권한입니다.
USER_SYS_PRIVS	사용자에게 부여된 시스템 권한을 나열합니다.

ORACLE

부여된 권한 확인

DELETE 권한이 없는 테이블에서 행 삭제와 같은 승인되지 않은 작업을 시도하면 Oracle server가 해당 작업의 수행을 거부합니다.

다음 작업 중 하나를 수행하면 “table or view does not exist”라는 Oracle server 오류 메시지가 나타납니다.

- 존재하지 않는 테이블 또는 뷰에 이름을 지정한 경우
- 권한이 없는 테이블 또는 뷰에서 작업을 수행한 경우

데이터 딕셔너리를 액세스하여 부여 받은 권한을 볼 수 있습니다. 슬라이드의 차트는 다양한 데이터 딕셔너리 테이블을 설명한 것입니다.

객체 권한 취소 방법

- REVOKE 문을 사용하여 다른 사용자에게 부여된 권한을 취소합니다.
- WITH GRANT OPTION 절을 통해 다른 사용자에게 부여된 권한도 취소됩니다.

```
REVOKE {privilege [, privilege...]}|ALL}
ON      object
FROM   {user[, user...]|role|PUBLIC}
[CASCADE CONSTRAINTS];
```

ORACLE

13-17

Copyright © Oracle Corporation, 2001. All rights reserved.

객체 권한 취소

REVOKE 문을 사용하여 다른 사용자에게 부여된 권한을 취소할 수 있습니다. REVOKE 문을 사용하면 지정한 사용자 및 WITH GRANT OPTION 절을 통해 권한을 부여 받은 사용자의 권한을 취소할 수 있습니다.

구문 설명:

CASCADE CONSTRAINTS	REFERENCES 권한을 통해 객체에 생성된 참조 무결성 제약 조건을 제거할 때 필요합니다.
------------------------	---

자세한 내용은 *Oracle9i SQL Reference*, “REVOKE”를 참조하십시오.

객체 권한 취소

사용자 **Alice**의 입장이 되어 DEPARTMENTS 테이블에 대해 사용자 **Scott**에게 부여된 SELECT 및 INSERT 권한을 취소 합니다.

```
REVOKE select, insert  
ON departments  
FROM scott;  
Revoke succeeded.
```

ORACLE

13-18

Copyright © Oracle Corporation, 2001. All rights reserved.

객체 권한 취소(계속)

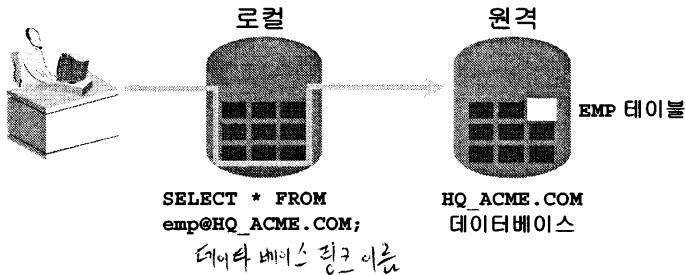
슬라이드의 예제는 DEPARTMENTS 테이블에 대해 사용자 **Scott**에게 부여된 SELECT 및 INSERT 권한을 취소합니다.

참고: WITH GRANT OPTION 절을 통해 권한을 부여 받은 사용자는 WITH GRANT OPTION 절을 사용하여 이 권한을 다른 사용자에게 부여할 수 있으므로 이 권한을 갖는 사용자가 긴 체인 형식으로 늘어날 수 있습니다. 그러나 동일한 권한을 부여 받은 사용자가 순환 형식으로 존재할 수는 없습니다. 다른 사용자에게 권한을 부여한 사용자의 권한을 취소하면 부여된 모든 권한이 연쇄적으로 취소됩니다.

예를 들어, 사용자 A가 WITH GRANT OPTION 절을 포함시켜 테이블에 대한 SELECT 권한을 사용자 B에게 부여하면 사용자 B는 사용자 C에게 WITH GRANT OPTION을 사용하여 SELECT 권한을 부여하고 마찬가지로 사용자 C는 사용자 D에게 SELECT 권한을 부여할 수 있습니다. 사용자 A가 사용자 B에게 부여한 권한을 취소하면 사용자 C 및 D에게 부여된 권한도 취소됩니다.

데이터베이스 링크

데이터베이스 링크 연결을 사용하면 로컬 사용자가 원격 데이터베이스의 데이터를 액세스할 수 있습니다.



ORACLE

13-19

Copyright © Oracle Corporation, 2001. All rights reserved.

데이터베이스 링크

데이터베이스 링크는 오라클 데이터베이스 서버에서 다른 데이터베이스 서버로의 일방 통행 경로를 정의하는 포인터입니다. 링크 포인터는 실제로는 데이터 딕셔너리 테이블의 항목으로 정의됩니다. 링크를 액세스하려면 해당 데이터 딕셔너리 항목을 포함하는 로컬 데이터베이스에 연결되어 있어야 합니다.

데이터베이스 링크 연결은 로컬 데이터베이스 A에 연결된 클라이언트가 데이터베이스 A에 저장된 링크를 사용하여 원격 데이터베이스 B에 있는 정보를 액세스할 수 있지만, 데이터베이스 B에 연결된 사용자는 동일한 링크를 사용하여 데이터베이스 A에 있는 정보를 사용할 수 없기 때문에 일방 통행입니다. 데이터베이스 B의 로컬 사용자가 데이터베이스 A에 있는 데이터를 액세스하려면 데이터베이스 B의 데이터 딕셔너리에 저장되는 링크를 정의해야 합니다.

데이터베이스 링크 연결을 사용하면 로컬 사용자가 원격 데이터베이스에 있는 데이터를 액세스 할 수 있습니다. 이러한 연결이 가능하려면 분산 시스템의 각 데이터베이스에 고유한 전역 데이터베이스 이름이 있어야 합니다. 전역 데이터베이스 이름은 분산 시스템에서 데이터베이스 서버를 고유하게 식별합니다.

데이터베이스 링크의 커다란 이점은 사용자로 하여금 원격 데이터베이스에 있는 다른 사용자의 객체를 액세스하도록 허용하되, 객체 소유자가 지정하는 권한 집합에 따라 사용자를 제한 할 수 있다는 것입니다. 다시 말하면, 로컬 사용자가 해당 원격 데이터베이스의 사용자가 아니라도 원격 데이터베이스를 액세스할 수 있습니다.

예제는 사용자 SCOTT가 전역 이름이 HQ.ACME.COM인 원격 데이터베이스에 있는 EMP 테이블을 액세스하는 것을 보여줍니다.

참고: 일반적으로 DBA가 데이터베이스 링크를 생성해야 합니다. 딕셔너리 뷰 USER_DB_LINKS에는 사용자가 액세스할 수 있는 링크 정보가 포함됩니다.

데이터베이스 링크

- 데이터베이스 링크를 생성합니다.

```
CREATE PUBLIC DATABASE LINK hq.acme.com  
USING 'sales'; 데이터 베이스 링크  
Database link created.
```

- 데이터베이스 링크를 사용하는 SQL 문을 작성합니다.

```
SELECT *  
FROM emp@HQ.ACME.COM;
```

ORACLE

13-20

Copyright © Oracle Corporation, 2001. All rights reserved.

데이터베이스 링크 사용

예제는 데이터베이스 링크를 생성합니다. USING 절은 원격 데이터베이스의 서비스 이름을
식별합니다.

데이터베이스 링크가 생성되면 원격 사이트의 데이터를 사용하는 SQL 문을 작성할 수 있습니다.
동의어가 설정된 경우 동의어를 사용하여 SQL 문을 작성할 수 있습니다.

예를 들면 다음과 같습니다.

```
CREATE PUBLIC SYNONYM HQ_EMP FOR emp@HQ.ACME.COM;
```

그런 다음 동의어를 사용하는 SQL 문을 작성합니다.

```
SELECT * FROM HQ_EMP;
```

원격 객체에 대한 권한은 부여할 수 없습니다.

요약

이 단원에서는 데이터베이스 및 데이터베이스 객체에 대한 액세스를 제어하는 **DCL** 문에 대해 배웠습니다.

명령문	기능
CREATE USER	사용자 생성(일반적으로 DBA가 수행)
GRANT	다른 사용자에게 자신의 객체를 액세스 할 수 있는 권한 부여
CREATE ROLE	권한 collection을 생성 (일반적으로 DBA가 수행)
ALTER USER	사용자 암호 변경
REVOKE	사용자가 가진 객체 권한 취소

ORACLE

요약

DBA는 사용자에게 권한을 할당하여 사용자의 초기 데이터베이스 보안을 설정합니다.

- DBA는 반드시 암호를 사용하는 사용자를 생성해야 하며 각 사용자의 초기 시스템 권한을 설정해야 합니다.
- 사용자는 객체를 생성한 후 GRANT 문을 사용하여 다른 사용자 또는 모든 사용자에게 사용 가능한 객체 권한을 부여할 수 있습니다.
- DBA는 CREATE ROLE 문을 사용하여 룰을 생성해서 시스템 또는 객체 권한 collection을 여러 사용자에게 부여할 수 있습니다. 룰을 사용하면 권한 부여 및 취소를 쉽게 유지 관리할 수 있습니다.
- ALTER USER 문을 사용하여 암호를 변경할 수 있습니다.
- REVOKE 문을 사용하여 사용자의 권한을 취소할 수 있습니다.
- 데이터 딕셔너리 뷰를 통해 사용자는 자신에게 부여된 권한 및 자신의 객체에 부여된 권한을 볼 수 있습니다.
- 데이터베이스 링크를 통해 원격 데이터베이스에 있는 데이터를 액세스할 수 있습니다. 원격 객체에 대한 권한은 부여할 수 없습니다.

연습 13 개요

이 연습에서는 다음 내용을 다룹니다.

- 다른 사용자에게 테이블에 대한 권한 부여
- 부여 받은 권한을 통해 다른 사용자의 테이블 수정
- 동의어 생성
- 권한과 관련된 데이터 딕셔너리 뷰 질의

ORACLE

13-22

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 13 개요

다른 학생과 팀을 구성하여 데이터베이스 객체에 대한 액세스 제어를 연습해 봅니다.

연습 13

1. 사용자가 Oracle Server에 로그인하기 위해 필요한 권한은 무엇입니까? 시스템 권한과 객체 권한 중 어느 것입니까?

2. 테이블을 생성할 때 필요한 권한은 무엇입니까?

3. 생성한 테이블에 대한 권한을 다른 사용자에게 부여할 수 있는 사용자는 누구입니까?

4. DBA가 동일한 시스템 권한이 필요한 사용자를 여러 명 생성할 때 간편하게 작업할 수 있는 방법은 무엇입니까?

5. 암호를 변경할 때 사용하는 명령은 무엇입니까?

6. 자신이 생성한 DEPARTMENTS 테이블에 대한 액세스 권한을 다른 사용자에게 부여하고 해당 사용자의 DEPARTMENTS 테이블에 대한 질의 액세스 권한을 부여 반도록 하십시오.
7. 자신의 DEPARTMENTS 테이블에 있는 모든 행을 질의하십시오.

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

8. DEPARTMENTS 테이블에 새 행을 추가하십시오. 팀 1은 Education을 부서 번호 500으로, 팀 2는 Human Resources를 부서 번호 510으로 추가한 후 서로 상대 팀의 테이블을 질의하십시오.
9. 상대 팀의 DEPARTMENTS 테이블에 대한 동의어를 생성하십시오.

연습 13(계속)

10. 동의어를 사용하여 상대 팀의 DEPARTMENTS 테이블에 포함된 모든 행을 질의 하십시오.

팀 1의 SELECT 문:

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700
500	Education		

9 rows selected.

팀 2의 SELECT 문:

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700
510	Human Resources		

9 rows selected.

연습 13(계속)

11. 소유한 테이블에 대한 정보를 표시하도록 USER_TABLES 데이터 딕셔너리를 질의하십시오.

TABLE_NAME
COUNTRIES
DEPARTMENTS
DEPT
EMP
EMPLOYEES
JOB
JOB_GRADES
JOB_HISTORY
LOCATIONS
REGIONS

10 rows selected.

12. 액세스할 수 있는 모든 테이블에 대한 정보를 표시하도록 ALL_TABLES 데이터 딕셔너리 뷰를 질의하십시오(현재 소유한 테이블 제외).

참고: 나열되는 목록이 아래에 표시된 목록과 다를 수 있습니다.

TABLE_NAME	OWNER
DEPARTMENTS	owner

13. 상대 팀에게 부여했던 자신의 테이블에 대한 SELECT 권한을 취소하십시오.

14. 8 단계에서 DEPARTMENTS 테이블에 삽입한 행을 제거하고 변경 내용을 저장하십시오.

14

SQL 강습

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

강습 개요

이 강습에서는 다음 내용을 다릅니다.

- 테이블 및 시퀀스 생성
- 테이블 데이터 수정
- 테이블 정의 수정
- 뷰 생성
- SQL 및 iSQL*Plus 명령을 포함하는 스크립트 작성
- 단순 보고서 생성

ORACLE

14-2

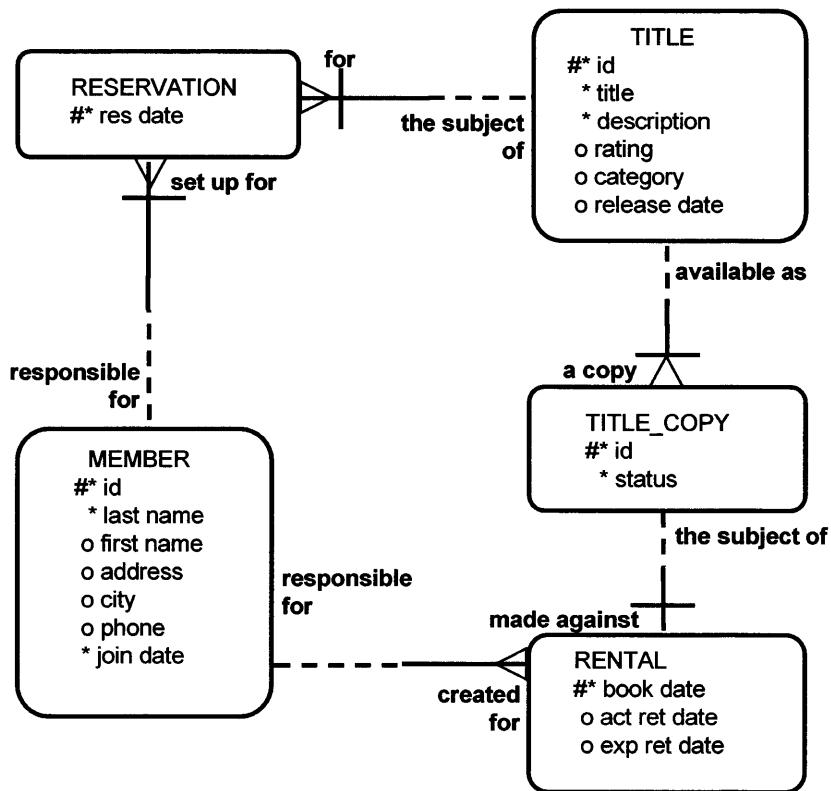
Copyright © Oracle Corporation, 2001. All rights reserved.

강습 개요

이 강습에서는 비디오 응용 프로그램에 대한 데이터베이스 테이블을 생성한 후 비디오 대여점 데이터베이스의 레코드를 삽입, 갱신 및 삭제하고 보고서를 생성합니다. 데이터베이스에는 꼭 필요한 테이블만 포함됩니다.

참고: 테이블을 생성하려면 iSQL*Plus에서 buildtab.sql 스크립트의 명령을 실행하고 테이블을 삭제하려면 iSQL*Plus에서 dropvid.sql 스크립트의 명령을 실행하십시오. 그런 다음 iSQL*Plus에서 buildvid.sql 스크립트의 명령을 실행하여 테이블을 생성하고 채울 수 있습니다. buildvid.sql 스크립트를 사용하여 테이블을 생성하고 추가할 경우에는 6b 단계부터 시작하십시오.

비디오 응용 프로그램 엔티티 관계 다이어그램



연습 14

1. 다음 테이블 인스턴스 차트를 기반으로 테이블을 생성하십시오. 적합한 데이터 유형을 선택하고 무결성 제약 조건을 추가해야 합니다.

a. 테이블 이름: MEMBER

열 이름	MEMBER_ID	LAST_NAME	FIRST_NAME	ADDRESS	CITY	PHONE	JOIN_DATE
키 유형	PK						
널/고유	NN,U	NN					NN
기본값							System Date
데이터 유형	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
길이	10	25	25	100	30	15	

b. 테이블 이름: TITLE

열 이름	TITLE_ID	TITLE	DESCRIPTION	RATING	CATEGORY	RELEASE_DATE
키 유형	PK					
널/고유	NN,U	NN	NN			
확인				G, PG, R, NC17, NR	DRAMA, COMEDY, ACTION, CHILD, SCI FI, DOCUMENTARY	
데이터 유형	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
길이	10	60	400	4	20	

연습 14(계속)

c. 테이블 이름: TITLE_COPY

열 이름	COPY_ID	TITLE_ID	STATUS
키 유형	PK	PK,FK	
넓/고유	NN,U	NN,U	NN
확인			AVAILABLE, DESTROYED, RENTED, RESERVED
FK 참조 테이블		TITLE	
FK 참조 열		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
기본값	System Date				System Date + 2 days	
FK 참조 테이블		MEMBER	TITLE_COPY			TITLE_COPY
FK 참조 열		MEMBER_ID	COPY_ID			TITLE_ID
데이터 유형	DATE	NUMBER	NUMBER	DATE	DATE	NUMBER
길이		10	10			10

연습 14(계속)

e. 테이블 이름: RESERVATION

열 이름	RES_DATE	MEMBER_ID	TITLE_ID
키 유형	PK	PK_FK1	PK_FK2
널/고유	NN,U	NN,U	NN
FK 참조 테이블		MEMBER	TITLE
FK 참조 열		MEMBER_ID	TITLE_ID
데이터 유형	DATE	NUMBER	NUMBER
길이		10	10

2. 데이터 딕셔너리를 참조하여 테이블 및 제약 조건이 제대로 생성되었는지 확인하십시오.

TABLE_NAME
MEMBER
RENTAL
RESERVATION
TITLE
TITLE_COPY

CONSTRAINT_NAME	C	TABLE_NAME
MEMBER_LAST_NAME_NN	C	MEMBER
MEMBER_JOIN_DATE_NN	C	MEMBER
MEMBER_MEMBER_ID_PK	P	MEMBER
RENTAL_BOOK_DATE_COPY_TITLE_PK	P	RENTAL
RENTAL_MEMBER_ID_FK	R	RENTAL
RENTAL_COPY_ID_TITLE_ID_FK	R	RENTAL
RESERVATION_RESDATE_MEM_TIT_PK	P	RESERVATION
RESERVATION_MEMBER_ID	R	RESERVATION
RESERVATION_TITLE_ID	R	RESERVATION
TITLE_TITLE_NN	C	TITLE

18 rows selected.

연습 14(계속)

3. MEMBER 테이블 및 TITLE 테이블의 각 행(row)을 고유하게 식별하는 시퀀스를 생성하십시오.
- MEMBER 테이블의 회원 번호는 101부터 시작하고 값이 캐시되지 않도록 하십시오. 시퀀스 이름은 MEMBER_ID_SEQ로 지정하십시오.
 - TITLE 테이블의 제목 번호는 92부터 시작하고 캐시되지 않도록 하십시오. 시퀀스 이름은 TITLE_ID_SEQ로 지정하십시오.
 - 데이터 딕셔너리에서 시퀀스의 존재를 확인하십시오.

SEQUENCE_NAME	INCREMENT_BY	LAST_NUMBER
TITLE_ID_SEQ	1	92
MEMBER_ID_SEQ	1	101

4. 테이블에 데이터를 추가하고 추가할 각 데이터 집합에 대한 스크립트를 작성하십시오.
- TITLE 테이블에 영화 제목을 추가하십시오. 영화 정보를 입력할 스크립트를 작성하여 lab14_4a.sql이라는 이름으로 저장하십시오. 시퀀스를 사용하여 각 제목을 고유하게 식별하고 출시일을 DD-MON-YYYY 형식으로 입력하십시오. 문자 필드에 사용된 작은 따옴표는 특별하게 처리된다는 점을 기억하십시오. 추가한 항목을 확인하십시오.

TITLE
Willie and Christmas Too
Alien Again
The Glob
My Day Off
Miracles on Ice
Soda Gang

6 rows selected.

연습 14(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

연습 14(계속)

- b. MEMBER 테이블에 다음 레이터를 추가하십시오. INSERT 문을 lab14_4b.sql이라는 이름의 스크립트에 추가하고 스크립트의 명령을 실행하십시오. 시퀀스를 사용하여 회원 번호를 추가해야 합니다.

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 14(계속)

- c. TITLE_COPY 테이블에 다음 영화 복사본을 추가하십시오.
참고: 이 연습 문제에서는 TITLE_ID 번호를 사용할 수 있습니다.

Title	Copy_Id	Status
Willie and Christmas Too	1	AVAILABLE
Alien Again	1	AVAILABLE
	2	RENTED
The Glob	1	AVAILABLE
My Day Off	1	AVAILABLE
	2	AVAILABLE
	3	RENTED
Miracles on Ice	1	AVAILABLE
Soda Gang	1	AVAILABLE

- d. RENTAL 테이블에 다음 대여 항목을 추가하십시오.
참고: 제목 번호는 시퀀스 번호에 따라 달라집니다.

Title_Id	Copy_Id	Member_Id	Book_date	Exp_Ret_Date	Act_Ret_Date
92	1	101	3 days ago	1 day ago	2 days 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	2 days ago

연습 14(계속)

5. 영화 제목, 각 복사본의 대여 가능 여부, 반환 예정일(대여된 경우)을 표시하는 TITLE_AVAIL이라는 이름의 뷰를 생성하고 뷰의 모든 행(row)을 질의한 후 결과를 제목별로 정렬하십시오.

참고: 결과는 각기 다를 수 있습니다.

TITLE	COPY_ID	STATUS	EXP_RET_D
Alien Again	1	AVAILABLE	
Alien Again	2	RENTED	26-SEP-01
Miracles on Ice	1	AVAILABLE	
My Day Off	1	AVAILABLE	
My Day Off	2	AVAILABLE	
My Day Off	3	RENTED	27-SEP-01
Soda Gang	1	AVAILABLE	25-SEP-01
The Glob	1	AVAILABLE	
Willie and Christmas Too	1	AVAILABLE	26-SEP-01

9 rows selected.

6. 테이블의 데이터를 변경하십시오.

- a. 새 제목을 추가하십시오. 이 영화의 제목은 “Interstellar Wars”이고 등급은 PG이며 SCIFI(공상 과학 영화)로 분류됩니다. 출시일은 1977년 7월 7일이며 영화에 대한 정보는 “Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?”입니다. 두 개의 복사본에 대해 TITLE_COPY 레코드를 추가해야 합니다.
- b. 예약 항목 두 개를 입력하십시오. 하나는 “Interstellar Wars”를 빌리려는 Carmen Velasquez에 대한 항목이고 다른 하나는 “Soda Gang”을 빌리려는 Mark Quick-to-See에 대한 항목입니다.

연습 14(계속)

- c. 고객 Carmen Velasquez가 영화 “Interstellar Wars”의 복사본 1을 빌렸으므로 영화 예약 항목에서 Carmen Velasquez를 제거하고 대여 정보를 기록하십시오. 반환 예정일에는 기본값을 사용할 수 있도록 하고, 작성한 뷰를 사용하여 대여 정보가 기록되었는지 확인하십시오.

참고: 결과는 각기 다를 수 있습니다.

TITLE	COPY_ID	STATUS	EXP_RET_D
Alien Again	1	AVAILABLE	
Alien Again	2	RENTED	26-SEP-01
Interstellar Wars	1	RENTED	29-SEP-01
Interstellar Wars	2	AVAILABLE	
Miracles on Ice	1	AVAILABLE	
My Day Off	1	AVAILABLE	
My Day Off	2	AVAILABLE	
My Day Off	3	RENTED	27-SEP-01
Soda Gang	1	AVAILABLE	25-SEP-01
The Glob	1	AVAILABLE	
Willie and Christmas Too	1	AVAILABLE	26-SEP-01

11 rows selected.

7. 테이블 중 하나를 수정하십시오.

- a. TITLE 테이블에 PRICE 열을 추가하여 비디오 구입 가격을 기록하십시오. 열은 소수 둘째 자리까지 포함하여 총 8자리로 표시하십시오. 수정 내용을 확인하십시오.

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)

연습 14(계속)

- b. 다음 목록에 따라 각 비디오의 가격을 개신하는 UPDATE 문을 포함하는 lab14_7b.sql이라는 스크립트를 생성하고 스크립트의 명령을 실행하십시오.
참고: 이 연습 문제에서는 TITLE_ID 번호를 사용할 수 있습니다.

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

- c. 이후에는 모든 제목에 가격이 포함되도록 하십시오. 제약 조건을 확인하십시오.

CONSTRAINT_NAME	C	SEARCH_CONDITION
TITLE_TITLE_NN	C	"TITLE" IS NOT NULL
TITLE_PRICE_NN	C	"PRICE" IS NOT NULL

6 rows selected.

8. Customer History Report라는 제목의 보고서를 작성하십시오. 이 보고서에는 고객 이름, 대여한 영화, 대여 날짜 및 대여 기간 같은 각 고객의 비디오 대여 기록이 포함됩니다. 또한 보고 기간 동안 모든 고객에게 대여된 총 대여 수를 계산합니다. 이 보고서를 생성하는 명령을 lab14_8.sql이라는 스크립트 파일로 저장하십시오.

참고: 결과는 각기 다를 수 있습니다.

Thu Sep 27

Customer History Report

page 1

MEMBER	TITLE	BOOK_DATE	DURATION
Carmen Velasquez	Willie and Christmas Too	24-SEP-01	1
	Alien Again	26-SEP-01	
	Interstellar Wars	27-SEP-01	
LaDoris Ngao	My Day Off	25-SEP-01	
Molly Urguhart	Soda Gang	23-SEP-01	2

15
SAP 연산자 사용

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- SET 연산자 설명
- SET 연산자를 사용하여 여러 질의를 하나의 질의로 결합
- 반환되는 행 순서 제어

ORACLE

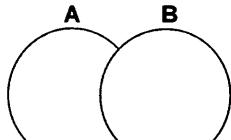
15-2

Copyright © Oracle Corporation, 2001. All rights reserved.

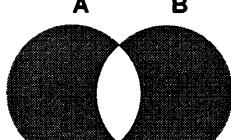
단원 목표

이 단원에서는 SET 연산자를 사용하여 질의를 작성하는 방법을 설명합니다.

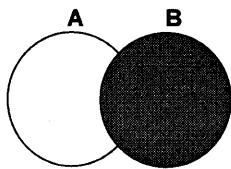
SET 연산자



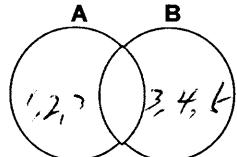
UNION/UNION ALL



INTERSECT



MINUS



$$A \cup B = \{1, 2, 3, 4, 5\}$$

$$A \cup B \text{ All } B = \{1, 2, 3, 3, 4, 5\}$$

$$A \cap B = \{3\}$$

$$A \setminus B = \{1, 2\}$$

$$B \setminus A = \{4, 5\}$$

ORACLE

15-3

Copyright © Oracle Corporation, 2001. All rights reserved.

SET 연산자

SET 연산자는 둘 이상의 구성 요소 질의 결과를 하나로 결합합니다. SET 연산자가 들어 있는 질의를 복합 질의라고 합니다.

연산자	반환 결과
UNION	두 질의 중 어느 것 하나에 의해서라도 선택된 모든 구분(distinct) 행 (row)
UNION ALL	중복 행을 포함하여 두 질의 중 어느 것 하나에 의해서라도 선택된 모든 행
INTERSECT	두 질의 모두에 의해 선택된 모든 구분 행
MINUS	첫째 SELECT 문에 의해 선택되고 둘째 SELECT 문에서 선택되지 않은 모든 구분 행

모든 SET 연산자는 동등한 우선순위를 갖습니다. SQL 문에 여러 개의 SET 연산자가 들어 있는 경우 명시적(explicit)으로 순서를 지정하는 괄호가 없으면 Oracle server에서는 이를 왼쪽(위)에서 오른쪽(아래)로 평가합니다. INTERSECT 연산자를 다른 SET 연산자와 함께 사용하는 질의에서 명시적으로 평가 순서를 지정하려면 괄호를 사용해야 합니다.

참고: 슬라이드에서 다이어그램의 밝은 색(회색) 부분은 질의 결과를 나타냅니다.

이 단원에서 사용되는 테이블

이 단원에서 사용되는 테이블은 다음과 같습니다.

- * **EMPLOYEES:** 현재의 모든 사원에 대한 세부 사항을 제공합니다.
- * **JOB_HISTORY:** 사원의 업무가 변경된 경우 이전 업무의 시작 및 종료 일자, 업무 식별 번호 및 부서에 관한 세부 사항을 기록합니다.

ORACLE

15-4

Copyright © Oracle Corporation, 2001. All rights reserved.

이 단원에서 사용되는 테이블

이 단원에서는 EMPLOYEES 테이블과 JOB_HISTORY 테이블을 사용합니다.

EMPLOYEES 테이블은 사원 세부 사항을 저장합니다. 이 테이블은 인적 자원 레코드를 관리하기 위한 것으로, 각 사원에 대한 고유 식별 번호와 전자 우편 주소를 비롯하여 사원의 업무 식별 번호, 급여 및 관리자에 관한 세부 사항이 저장됩니다. 일부 사원은 급여뿐 아니라 커미션을 지급받는데 이 정보 또한 추적할 수 있습니다. 회사에서는 사원의 역할을 업무로 구분합니다. 회사에 오래 근무하면서 업무가 변경된 사원이 있을 수 있는데 이러한 내역은 JOB_HISTORY 테이블을 사용하여 모니터 할 수 있습니다. 사원의 업무가 변경되면 이전 업무의 시작 및 종료 일자, 업무 식별 번호 및 부서에 관한 세부 사항이 JOB_HISTORY 테이블에 기록됩니다.

EMPLOYEES 및 JOB_HISTORY 테이블의 구조 및 데이터는 다음 페이지에서 보여줍니다.

제작 중에 같은 업무를 여러 번 담당할 수도 있습니다. 예를 들어, Taylor라는 사원이 1998년 3월 24일에 입사했다고 가정합니다. Taylor는 98년 3월 24일부터 98년 12월 31일까지 업무가 SA_REP이고 99년 1월 1일부터 99년 12월 31일까지 업무가 SA_MAN이었습니다. 현재 Taylor는 SA_REP로 다시 업무가 변경된 상태입니다.

비슷한 경우로 1987년 9월 17일에 입사한 Whalen이란 사원이 있다고 가정합니다. Whalen은 87년 9월 17일부터 93년 6월 17일까지 업무가 AD_ASST였고 94년 7월 1일부터 98년 12월 31일 까지 업무가 AC_ACCOUNT였습니다. 현재 Whalen은 AD_ASST로 다시 업무가 변경된 상태입니다.

이 단원에서 사용되는 테이블(계속)

DESC 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)
DEPARTMENT_NAME		VARCHAR2(14)

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

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
100	King	AD_PRES	17-JUN-87	90
101	Kochhar	AD_VP	21-SEP-89	90
102	De Haan	AD_VP	13-JAN-93	90
103	Hunold	IT_PROG	03-JAN-90	60
104	Ernst	IT_PROG	21-MAY-91	60
107	Lorentz	IT_PROG	07-FEB-99	60
124	Mourgos	ST_MAN	16-NOV-99	50
141	Rajs	ST_CLERK	17-OCT-95	50
142	Davies	ST_CLERK	29-JAN-97	50
143	Matos	ST_CLERK	15-MAR-98	50
144	Vargas	ST_CLERK	09-JUL-98	50
149	Zlotkey	SA_MAN	29-JAN-00	80
174	Abel	SA REP	11-MAY-96	80
176	Taylor	SA REP	24-MAR-98	80
EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
178	Grant	SA REP	24-MAY-99	
200	Whalen	AD_ASST	17-SEP-87	10
201	Hartstein	MK_MAN	17-FEB-96	20

20 rows selected.

이 단원에서 사용되는 테이블(계속)

DESC 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;

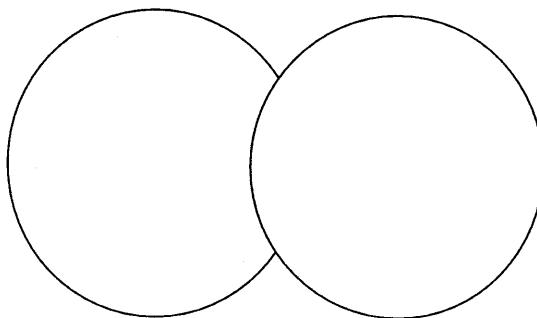
EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
102	13-JAN-93	24-JUL-98	IT_PROG	60
101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
101	28-OCT-93	15-MAR-97	AC_MGR	110
201	17-FEB-96	19-DEC-99	MK_REP	20
114	24-MAR-98	31-DEC-99	ST_CLERK	50
122	01-JAN-99	31-DEC-99	ST_CLERK	50
200	17-SEP-87	17-JUN-93	AD_ASST	90
176	24-MAR-98	31-DEC-98	SA REP	80
176	01-JAN-99	31-DEC-99	SA_MAN	80
200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

10 rows selected.

UNION 연산자

A

B



UNION 연산자는 두 질의의 결과를 중복을 제거한 후 반환합니다.

ORACLE

15-7

Copyright © Oracle Corporation, 2001. All rights reserved.

UNION 연산자

UNION 연산자는 두 질의 중 어느 것 하나에 의해서라도 선택된 행은 모두 반환합니다. 여러 테이블의 행을 모두 반환한 후 중복 행을 제거하려면 UNION 연산자를 사용합니다.

지침

- 선택될 열의 개수와 데이터 유형은 질의에 사용된 모든 SELECT 문에서 같아야 합니다. 열 이름은 같지 않아도 됩니다.
- UNION은 선택될 열 전체에 대해 작동합니다.
- 중복을 확인하는 동안에도 NULL 값은 무시되지 않습니다.
- IN 연산자는 UNION 연산자보다 높은 우선순위를 갖습니다.
- 기본적으로 결과는 SELECT 질의 첫째 열에 대해 오름차순으로 정렬됩니다.

UNION 연산자 사용

모든 사원의 현재 및 이전 업무에 대한 세부 사항을 표시합니다. 각 사원에 대해 한 번씩만 표시합니다.

```
SELECT employee_id, job_id  
FROM employees  
UNION  
SELECT employee_id, job_id  
FROM job_history;
```

EMPLOYEE_ID	JOB_ID
100	AD_PRES
101	AC_ACCOUNT
...	
200	AC_ACCOUNT
200	AD_ASST
...	
205	AC_MGR
206	AC_ACCOUNT

ORACLE

15-8

Copyright © Oracle Corporation, 2001. All rights reserved.

UNION SET 연산자 사용

UNION 연산자를 사용하면 중복 레코드가 모두 제거됩니다. EMPLOYEES 및 JOB_HISTORY 테이블 모두에 같은 레코드가 있으면 한 번만 표시됩니다. 슬라이드에 표시된 출력에서, EMPLOYEE_ID가 200인 사원에 대한 레코드는 JOB_ID가 서로 다르므로 두 번 표시됩니다.

다음 예제를 살펴봅니다.

```
SELECT employee_id, job_id, department_id  
FROM employees  
UNION  
SELECT employee_id, job_id, department_id  
FROM job_history;
```

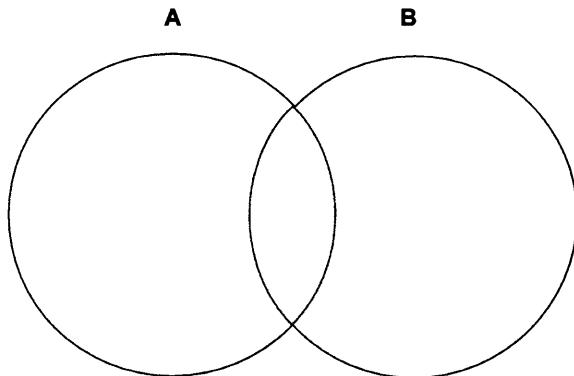
EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
200	AC_ACCOUNT	90
200	AD_ASST	10
200	AD_ASST	90

29 rows selected.

UNION SET 연산자 사용(계속)

앞의 출력에서 사원 ID가 200인 레코드가 세 번 표시됩니다. 그 이유는 사원 ID가 200인 레코드의 DEPARTMENT_ID 값 때문입니다. 첫째 행은 DEPARTMENT_ID가 90이고 둘째 행은 10, 셋째 행은 90입니다. 업무 ID와 부서 ID의 조합이 모두 다르므로 사원 ID가 200인 행은 모두 고유하며 따라서 중복된 것으로 간주되지 않습니다. 결과는 SELECT 절의 첫째 열 즉, EMPLOYEE_ID에 대해 오름차순으로 정렬된다는 점에 주목하십시오.

UNION ALL 연산자



UNION ALL 연산자는 두 질의의 결과를 중복을 포함하여 반환합니다.

ORACLE

15-10

Copyright © Oracle Corporation, 2001. All rights reserved.

UNION ALL 연산자

여러 질의의 행을 모두 반환하려면 UNION ALL 연산자를 사용합니다.

지침

- UNION과 달리 중복 행이 제거되지 않으며 기본적으로 출력이 정렬되지 않습니다.
- DISTINCT 키워드는 사용할 수 없습니다.

참고: 위에 나온 예외를 제외하면 UNION과 UNION ALL의 지침은 동일합니다.

UNION ALL 연산자 사용

모든 사원의 현재 및 이전 부서를 표시합니다.

```
SELECT employee_id, job_id, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
100	AD_PRES	90
101	AD_VP	90
200	AD_ASST	10
200	AD_ASST	90
200	AC_ACCOUNT	90
205	AC_MGR	110
206	AC_ACCOUNT	110

30 rows selected.

ORACLE

15-11

Copyright © Oracle Corporation, 2001. All rights reserved.

UNION ALL 연산자(계속)

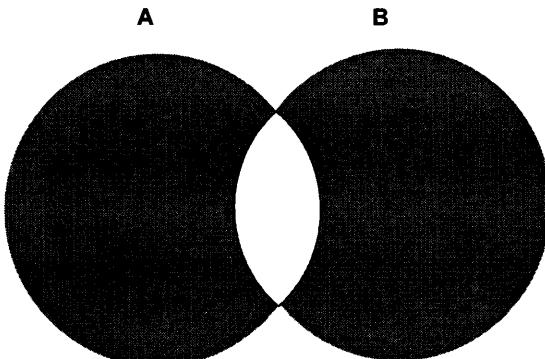
이 예제에서는 30행이 선택되었습니다. 두 테이블을 결합하면 총 30행이 됩니다. UNION ALL 연산자를 사용하면 중복 행이 제거되지 않습니다. 슬라이드의 출력에서 중복 행이 강조 표시되어 있습니다. UNION은 두 질의 중 어느 것 하나에 대해서만이라도 선택된 구분(distinct) 행을 모두 반환합니다. UNION ALL은 두 질의 중 어느 것 하나에 대해서만이라도 선택된 행은 중복 행을 포함하여 모두 반환합니다. 슬라이드의 질의를 UNION 질을 사용하여 다음과 같이 다시 작성합니다.

```
SELECT employee_id, job_id, department_id
FROM employees
UNION
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

마지막에 ← 이 질의는 다음과 같은 중복 행을 제거하므로 29행을 반환합니다.
↑ 이 질의는 다음과 같은 중복 행을 제거하므로 29행을 반환합니다.

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
176	SA_REP	80

INTERSECT 연산자



ORACLE

15-12

Copyright © Oracle Corporation, 2001. All rights reserved.

INTERSECT 연산자

INTERSECT 연산자는 여러 질의에 공통적인 행을 모두 반환합니다.

지침

- 질의에 있는 SELECT 문에 의해 선택될 열의 개수와 데이터 유형은 질의에 사용된 모든 SELECT 문에서 같아야 합니다. 열 이름은 같지 않아도 됩니다.
- INTERSECT에 사용될 테이블의 순서를 바꿔도 결과는 달라지지 않습니다.
- INTERSECT는 NULL 값을 무시하지 않습니다.

INTERSECT 연산자 사용

입사 이후 현재 업무와 같은 업무를 담당한 적이 있는
사원의 사원 ID와 업무 ID를 표시합니다.

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```

EMPLOYEE_ID	JOB_ID
176	SA_REP
200	AD_ASST

ORACLE

15-13

Copyright © Oracle Corporation, 2001. All rights reserved.

INTERSECT 연산자(계속)

이 슬라이드의 예제 질의에서는 양쪽 테이블에서 선택된 열이 같은 값을 가지는 레코드만
반환합니다.

EMPLOYEES 테이블에 대한 SELECT 문에 DEPARTMENT_ID 열을 추가하고 JOB_HISTORY 테이블에
대한 SELECT 문에 DEPARTMENT_ID 열을 추가한 다음 이 질의를 실행하면 추가된
행 값의 중복 여부가 달라지므로 결과가 달라질 수 있습니다.

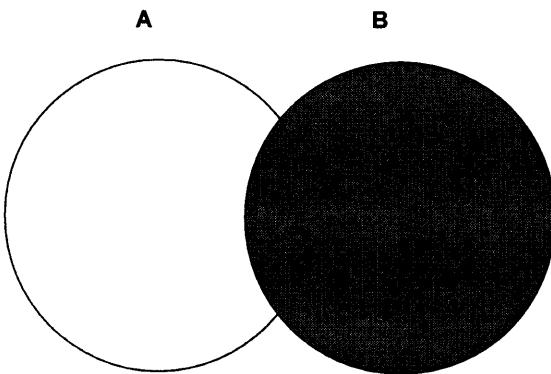
예제

```
SELECT employee_id, job_id, department_id
FROM employees
INTERSECT
SELECT employee_id, job_id, department_id
FROM job_history;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
176	SA_REP	80

사원 ID가 200인 행은 EMPLOYEES.DEPARTMENT_ID 값이 JOB_HISTORY.DEPARTMENT_ID 값과
다르므로 더 이상 결과에 출력되지 않습니다.

MINUS 연산자



ORACLE

15-14

Copyright © Oracle Corporation, 2001. All rights reserved.

MINUS 연산자

MINUS 연산자를 사용하면 첫째 질의가 반환한 행으로부터 둘째 질의가 반환한 행을 제외한 행이 반환됩니다(첫째 SELECT 문 MINUS 둘째 SELECT 문).

지침

- 질의에 있는 SELECT 문에 의해 선택될 열의 개수와 데이터 유형은 질의에 사용된 모든 SELECT 문에서 같아야 합니다. 열 이름은 같지 않아도 됩니다.
- WHERE 절에 있는 모든 열이 SELECT 절에 있어야 MINUS 연산자가 작동합니다.

MINUS 연산자

업무가 변경된 적이 없는 사원의 사원 ID를 표시합니다.

```
SELECT employee_id, job_id  
FROM   employees  
MINUS  차집합  
SELECT employee_id, job_id  
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID
100	AD_PRES
101	AD_VP
102	AD_VP
103	IT_PROG
...	
201	MK_MAN
202	MK_REP
205	AC_MGR
206	AC_ACCOUNT

18 rows selected.

ORACLE

15-15

Copyright © Oracle Corporation, 2001. All rights reserved.

MINUS 연산자(계속)

슬라이드의 예제에서는 EMPLOYEES 테이블의 사원 ID 및 업무 ID에서 JOB_HISTORY 테이블의 사원 ID 및 업무 ID를 뺍니다. 결과 집합에서는 EMPLOYEES 테이블에 있지만 JOB_HISTORY 테이블에는 없는 행을 표시합니다. 이것은 업무를 바꾼 적이 없는 사원에 대한 레코드입니다.

SET 연산자 지침

- SELECT 목록에 있는 표현식의 개수와 데이터 유형이 서로 일치해야 합니다.
- 실행 순서를 바꾸려면 팔호를 사용합니다.
- ORDER BY 절:
 - 명령문의 끝에만 사용할 수 있습니다.
 - 열 이름, 첫째 SELECT 문에서 사용한 별칭 또는 위치 표기법을 사용할 수 있습니다.

ORACLE

15-15

Copyright © Oracle Corporation, 2001. All rights reserved.

SET 연산자 지침

- 질의의 SELECT 목록에 있는 표현식의 개수와 데이터 유형이 서로 일치해야 합니다. WHERE 절에 UNION, UNION ALL, INTERSECT 및 MINUS SET 연산자를 사용하는 질의는 해당 SELECT 목록에도 WHERE 절과 같은 개수와 유형의 열을 사용해야 합니다. 예를 들면 다음과 같습니다.

```
SELECT employee_id, department_id
  FROM employees
 WHERE (employee_id, department_id)
       IN (SELECT employee_id, department_id
            FROM   employees
            UNION
            SELECT employee_id, department_id
            FROM   job_history);
```

- ORDER BY 절:
 - 명령문의 끝에만 사용할 수 있습니다.
 - 열 이름, 별칭 또는 위치 표기법을 사용할 수 있습니다.
- ORDER BY 절에 사용된 열 이름 또는 별칭은 첫째 SELECT 목록과 일치해야 합니다.
- SET 연산자는 서브 쿼리에 사용할 수 있습니다.

Oracle Server와 SET 연산자

- UNION ALL을 제외한 다른 연산자를 사용할 경우에는 중복 행이 자동으로 제거됩니다.
- 첫째 질의의 열 이름이 결과에 표시됩니다.
- UNION ALL을 제외한 다른 연산자의 출력은 기본적으로 오름차순으로 정렬됩니다.

15-17

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

Oracle Server와 SET 연산자

질의에서 SET 연산자를 사용하면 Oracle Server에서는 UNION ALL 연산자의 경우를 제외하고 자동으로 중복 행을 제거합니다. 출력되는 열 이름은 첫째 SELECT 문의 열 목록에 따라 결정됩니다. 기본적으로 출력은 SELECT 질의 첫째 열에 대해 오름차순으로 정렬됩니다.

혼합 질의의 여러 구성 요소 질의에 있는 SELECT 목록들의 표현식은 개수와 데이터 유형이 서로 일치해야 합니다. 구성 요소 질의에서 문자 데이터를 선택하면 반환 값의 데이터 유형은 다음과 같이 결정됩니다.

- 두 질의에서 모두 데이터 유형이 CHAR인 값을 사용하면 반환 값의 데이터 유형은 CHAR입니다.
- 두 질의 중 하나에서 데이터 유형이 VARCHAR2인 값을 사용하면 반환 값의 데이터 유형은 VARCHAR2입니다.

SELECT 문 일치

UNION 연산자를 사용하여 모든 사원의 부서 ID, 위치 및 입사 일자를 표시합니다.

```
SELECT department_id, TO_NUMBER(null)
      location, hire_date
  FROM employees
UNION
SELECT department_id, location_id, TO_DATE(null)
  FROM departments;
```

DEPARTMENT_ID	LOCATION	HIRE_DATE
10	1700	
10		17-SEP-87
20	1800	
20		17-FEB-96
...		
110	1700	
110		07-JUN-94
190	1700	
		24-MAY-99

27 rows selected.

ORACLE

15-18

Copyright © Oracle Corporation, 2001. All rights reserved.

SELECT 문 일치

두 질의의 SELECT 목록에 있는 표현식의 개수를 일치시켜야 하므로 더미(dummy) 열 및 데이터 유형 변환 함수를 사용하여 이 규칙을 적용할 수 있습니다. 슬라이드에서 location이라는 이름은 더미 열 머리글로 주어진 것입니다. 첫째 질의에 TO_NUMBER 함수를 사용하여 둘째 질의에서 검색하는 LOCATION_ID 열의 NUMBER 데이터 유형에 일치시킵니다. 마찬가지로 둘째 질의에도 TO_DATE 함수를 사용하여 첫째 질의에서 검색하는 HIRE_DATE 열의 DATE 데이터 유형에 일치시킵니다.

SELECT 문 일치

- UNION 연산자를 사용하여 모든 사원의 사원 ID, 업무 ID 및 급여를 표시합니다.

```
SELECT employee_id, job_id, salary  
FROM employees
```

UNION

```
SELECT employee_id, job_id, 0 이 Out-로 가기 때문,  
FROM job_history;
```

EMPLOYEE_ID	JOB_ID	SALARY
100	AD_PRES	24000
101	AC_ACCOUNT	0
101	AC_MGR	0
...		
205	AC_MGR	12000
206	AC_ACCOUNT	8300

30 rows selected.

ORACLE

15-19

Copyright © Oracle Corporation, 2001. All rights reserved.

SELECT 문 일치: 예제

EMPLOYEES 및 JOB_HISTORY 테이블에는 EMPLOYEE_ID, JOB_ID 및 DEPARTMENT_ID와 같이 공통적인 열이 여러 개 있습니다. 그러나 UNION 연산자를 사용하여 EMPLOYEE_ID, JOB_ID 및 SALARY를 표시하는 질의를 작성하려고 하는데 급여가 EMPLOYEES 테이블에만 존재하면 어떻게 하시겠습니까?

슬라이드에 있는 코드 예제에서는 EMPLOYEES 및 JOB_HISTORY 테이블의 EMPLOYEE_ID 열과 JOB_ID 열을 일치시키고, 리터럴 값 0을 JOB_HISTORY SELECT 문에 추가하여 EMPLOYEES SELECT 문에 있는 숫자 유형의 SALARY 열과 일치시킵니다.

이 출력 결과에서는 JOB_HISTORY 테이블의 레코드에 해당하는 각 행의 SALARY 열에 0이 들어 있습니다.

행 순서 제어

두 UNION 연산자를 사용하여 영어 문장을 만듭니다.

```
COLUMN a_dummy NOPRINT
SELECT 'sing' AS "My dream", 3 a_dummy
FROM dual
UNION
SELECT 'I''d like to teach', 1
FROM dual
UNION
SELECT 'the world to', 2
FROM dual
ORDER BY 2;
```

My dream
I'd like to teach
the world to
sing

ORACLE

15-20

Copyright © Oracle Corporation, 2001. All rights reserved.

행 순서 제어

기본적으로 출력은 첫째 열에 대해 오름차순으로 정렬됩니다. ORDER BY 절을 사용하면 이를 변경할 수 있습니다.

ORDER BY를 사용하여 행 정렬

ORDER BY 절은 혼합 질의에서 한 번만 사용할 수 있습니다. ORDER BY 절은 질의의 끝에 와야 하며 ORDER BY 절에는 열 이름, 별칭 또는 위치 표기법을 사용할 수 있습니다. ORDER BY 절 없이 슬라이드에 있는 코드 예제를 실행하면 첫째 열의 문자순으로 다음과 같이 출력됩니다.

My dream
I'd like to teach
sing
the world to

참고: UNION SET 연산자를 두 번 이상 사용한 혼합 질의에서는 ORDER BY 절에 명시적인(explicit) 표현식 대신 위치만 사용할 수 있습니다.

요약

이 단원에서는 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- UNION을 사용하여 모든 구분 행 반환
- UNION ALL을 사용하여 중복을 포함한 모든 행 반환
- INTERSECT를 사용하여 양쪽 질의에서 공유하는 모든 행 반환
- MINUS를 사용하여 첫째 질의에서 선택되지만 둘째 질의에서는 선택되지 않는 모든 구분 행 반환
- ORDER BY는 명령문의 끝에만 사용

ORACLE

15-21

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

- UNION 연산자는 두 질의 중 어느 것 하나에 의해서라도 선택된 행은 모두 반환합니다. 여러 테이블의 행을 모두 반환한 후 중복 행을 제거하려면 UNION 연산자를 사용합니다.
- 여러 질의의 행을 모두 반환하려면 UNION ALL 연산자를 사용합니다. UNION 연산자는 달리 중복 행이 제거되지 않고 출력이 기본적으로 정렬되지 않습니다.
- INTERSECT 연산자를 사용하여 여러 질의에 대해 공통적인 행을 모두 반환합니다.
- MINUS 연산자를 사용하면 첫째 질의가 반환한 행으로부터 둘째 질의가 반환한 행을 제외한 행을 반환할 수 있습니다.
- ORDER BY 절은 혼합문의 끝에만 사용한다는 점을 명심하십시오.
- SELECT 목록에 있는 표현식의 개수와 데이터 유형은 서로 일치해야 합니다.

연습 15 개요

이 연습에서는 Oracle9i datetime 함수 사용을 다룹니다.

ORACLE

15-22

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 15 개요

이 연습에서는 SET 연산자를 사용하는 질의를 작성합니다.

연습 15

1. SET 연산자를 사용하여 업무 ID ST_CLERK을 포함하지 않는 부서의 ID를 나열하십시오.

DEPARTMENT_ID
10
20
60
80
90
110
190

7 rows selected.

2. SET 연산자를 사용하여 해당 지역에 부서가 없는 지역 ID와 지역 이름을 표시하십시오.

CO	COUNTRY_NAME
DE	Germany

3. 부서 ID가 10, 50 및 20인 부서의 업무 목록을 해당 순서대로 나열하고, SET 연산자를 사용하여 업무 ID와 부서 ID를 표시하십시오.

JOB_ID	DEPARTMENT_ID
AD_ASST	10
ST_CLERK	50
ST_MAN	50
MK_MAN	20
MK_REP	20

4. 입사 이후 현재 업무와 같은 업무를 담당한 적이 있는 사원의 사원 ID와 업무 ID를 나열하십시오.

EMPLOYEE_ID	JOB_ID
176	SA_REP
200	AD_ASST

연습 15(계속)

5. 다음을 나열하는 혼합 질의를 작성하십시오.

- 소속된 부서에 상관 없이 EMPLOYEES 테이블에 있는 모든 사원의 이름과 부서 ID
- 소속된 사원에 상관 없이 DEPARTMENTS 테이블에 있는 모든 부서의 부서 ID와 부서 이름

LAST_NAME	DEPARTMENT_ID	TO_CHAR(NULL)
Abel	80	
Davies	50	
De Haan	90	
Ernst	60	
Fay	20	
Gietz	110	
Grant		
Hartstein	20	
Higgins	110	
Hunold	60	
King	90	
Kochhar	90	
Lorentz	60	
Matos	50	

LAST_NAME	DEPARTMENT_ID	TO_CHAR(NULL)
Moungos	50	
Rajs	50	
Taylor	80	
Vargas	50	
Whalen	10	
Zlotkey	80	
	10	Administration
	20	Marketing
	50	Shipping
	60	IT
	80	Sales
	90	Executive
	110	Accounting
	190	Contracting

28 rows selected.

16

Oracle9i Datetime 함수

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음 datetime 함수를 사용할 수 있습니다.

- TZ_OFFSET
- CURRENT_DATE
- CURRENT_TIMESTAMP
- LOCALTIMESTAMP
- DBTIMEZONE
- SESSIONTIMEZONE
- EXTRACT
- FROM_TZ
- TO_TIMESTAMP
- TO_TIMESTAMP_TZ
- TO_YMINTERVAL

ORACLE

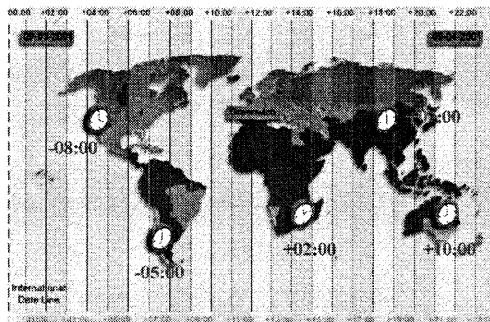
16-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 Oracle9i에 도입된 datetime 함수 몇 가지를 설명합니다.

시간대



이 이미지는 그리니치 시간이
12:00일 때 각 시간대의 시간을
나타냅니다.

16-3

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

시간대

Oracle9i에서는 날짜 및 시간 데이터에 시간대를 포함시킬 수 있으며 초를 소수점 이하 자릿수까지 표시할 수도 있습니다. 이 단원에서는 새 `datetime` 함수를 사용하여 Oracle9i에 포함된 새 `datetime` 데이터 유형을 조작하는 방법을 중점적으로 설명합니다. 이러한 함수 작업을 이해하려면 시간대 및 그리니치 표준시의 개념에 익숙해져야 합니다. 그리니치 표준시(GMT: Greenwich Mean Time)는 현재 협정 세계시(UTC: Coordinated Universal Time)로 사용되고 있습니다.

하루의 시간은 지구의 자전에 의해 측정되며 현재 위치에 따라 달라집니다. 영국 그리니치 시간으로 정오이면 날짜 변경선 상의 시간은 자정입니다. 지구는 24개의 시간대로 나뉘어져 있으며 각 시간대는 하루의 시간을 따로 나타냅니다. 영국 그리니치의 기본 자오선 상에 있는 지역의 시간을 그리니치 표준시 또는 GMT라고 합니다. GMT는 세계적으로 다른 시간대에서 참조하는 시간 표준입니다. 이는 일년 내내 동일하며 서머 타임(summer time)이나 일광 절약 시간(daylight savings time)에 의해 영향을 받지 않습니다. 자오선은 북극에서 남극을 잇는 가상

의 선입니다. 이 선의 경도는 0이며 이 선을 기준으로 다른 경도를 측정합니다. 모든 시간은 그리니치 표준시를 기준으로 상대적으로 측정되며 모든 위치는 위도(북위와 남위로 측정되는 적도까지의 거리)와 경도(동경과 서경으로 측정되는 그리니치 자오선까지의 거리)로 표시할 수 있습니다.

일광 절약 시간(daylight savings time)

대부분의 서방 국가에서는 여름 동안 시계를 한 시간 앞에 맞춥니다. 이 기간을 일광 절약 시간이라고 하며 미국, 멕시코 및 캐나다의 대부분 지역에서 4월 첫째 일요일부터 10월 마지막 일요일까지 지속됩니다. 유럽 연합의 국가들은 일광 절약 시간을 준수하지만 대신 서머 타임 기간이라고 부릅니다. 유럽의 서머 타임 기간은 북미보다 1주 먼저 시작하지만 같은 시기에 끝납니다.

Oracle9i Datetime 지원

- Oracle9i에서는 날짜 및 시간 데이터에 시간대를 포함 시킬 수 있으며 소수점 이하의 초를 지원합니다.
- DATE에 추가된 세 가지 데이터 유형
 - TIMESTAMP
 - TIMESTAMP WITH TIME ZONE (TSTZ)
 - TIMESTAMP WITH LOCAL TIME ZONE (TSLTZ)
- Oracle9i는 서버에서 datetime 데이터 유형에 대해 일광 절약 시간을 지원합니다.

ORACLE

16-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9i datetime 지원

Oracle9i에서는 DATE에 다음과 같은 세 가지 데이터 유형이 새로 추가되었습니다.

데이터 유형	시간대	소수점 이하의 초 표시
DATE	아니오	아니오
TIMESTAMP	아니오	예
TIMESTAMP (fractional_seconds_precision) WITH TIMEZONE	UTC(현정 세계시, 이전의 그리니치 표준 시)와의 시간 차이를 시와 분으로 나타내는 시간대 범위 값 및 TIMESTAMP의 모든 값입니다.	fractional_seconds_precision은 SECOND datetime 필드의 소수점 이하 초 부분의 자릿수입니다. 사용할 수 있는 값은 0부터 9까지이며 기본값은 6입니다.
TIMESTAMP (fractional_seconds_precision) WITH LOCAL TIME ZONE	TIMESTAMP WITH TIME ZONE의 모든 값에서 다음 경우가 제외된 값입니다. <ul style="list-style-type: none">• 데이터는 데이터베이스에 저장될 때 데이터베이스 시간대에 맞게 정규화됩니다.• 데이터가 검색될 때 사용자는 해당 세션의 시간대에 맞는 데이터를 보게 됩니다.	예

Oracle9i datetime 지원(계속)

TIMESTAMP WITH LOCAL TIME ZONE은 데이터베이스 시간대에 맞게 저장됩니다. 사용자가 데이터를 선택하면 해당 값은 사용자의 세션 시간대에 맞게 조정됩니다.

예제

샌프란시스코 데이터베이스의 시스템 시간대는 -8:00입니다. 세션 시간대가 -5:00인 뉴욕 클라이언트가 샌프란시스코 데이터베이스에서 데이터를 삽입하거나 선택하면 TIMESTAMP WITH LOCAL TIME ZONE 데이터가 다음과 같이 조정됩니다.

- 뉴욕 클라이언트가 샌프란시스코 데이터베이스의 TIMESTAMP WITH LOCAL TIME ZONE 열에 '1998-1-23 6:00:00-5:00'을 삽입하면 이 데이터는 샌프란시스코에서 이진 값 '1998-1-23 3:00:00'으로 저장됩니다.
- 뉴욕 클라이언트가 샌프란시스코 데이터베이스에서 삽입된 데이터를 선택하면 뉴욕에서는 값이 '1998-1-23 6:00:00'으로 표시됩니다.
- 샌프란시스코 클라이언트가 같은 데이터를 선택하면 값이 '1998-1-23 3:00:00'으로 표시됩니다.

일광 절약 시간(daylight savings time) 지원

Oracle Server는 모든 주어진 시간대 지역에 대해 일광 절약 시간이 유효한지 여부를 자동으로 확인하여 해당 지역 시간을 반환합니다. `datetime` 값을 사용하면 서버에서 주어진 지역에 대해 일광 절약 시간이 경계 시간을 제외한 모든 시간에 대해 유효한지 여부를 충분히 확인할 수 있습니다. 경계 시간은 일광 절약 시간이 효력을 발휘하기 시작할 때와 끝날 때 발생합니다. 예를 들어, 미국-태평양 지역에서 일광 절약 시간이 시행되면 오전 2시가 오전 3시로 변경되며 2시와 3시 사이의 한 시간 간격은 없어지게 됩니다. 일광 절약 시간이 종료되면 오전 2시는 오전 1시로 변경되며 1시와 2시 사이의 한 시간이 되풀이됩니다.

또한 Oracle9i에서는 단일 데이터베이스 인스턴스에서 전세계적인 응용 프로그램을 개발하고 배포할 수 있도록 함으로써 관련 비용을 상당히 줄일 수 있도록 했습니다. 응용 프로그램을 여러 지역에서 사용하려면 유니코드를 통해 명명된 시간대 및 다중 언어를 지원해야 합니다. `datetime` 데이터 유형인 TSLTZ 및 TSTZ는 시간대를 인식합니다. `datetime` 값을 특정 오프셋으로 지정하는 대신 특정 지역의 해당 시간으로 지정할 수 있습니다. 주어진 지역에 대한 시간대 규칙 테이블을 사용하면 일광 절약 시간을 고려한 지역 시간의 시간대 오프셋을 계산하고 이를 다른 작업에 사용할 수 있습니다.

이 단원에서는 Oracle9i에 새로 도입된 `datetime` 함수 몇 가지를 설명합니다.

TZ_OFFSET

- 'US/Eastern' 시간대의 시간대 오프셋을 표시합니다.

```
SELECT TZ_OFFSET('US/Eastern') FROM DUAL;
```

TZ_OFFSET
-04:00

- 'Canada/Yukon' 시간대의 시간대 오프셋을 표시합니다.

TZ_OFFSET
07:00

- 'Europe/London' 시간대의 시간대 오프셋을 표시합니다.

```
SELECT TZ_OFFSET('Europe/London') FROM DUAL;
```

TZ_OFFSET
+01:00

ORACLE

16-6

Copyright © Oracle Corporation, 2001. All rights reserved.

TZ_OFFSET

TZ_OFFSET 함수는 입력한 값에 해당하는 시간대 오프셋을 반환합니다. 반환 값은 문을 실행한 날짜에 따라 다릅니다. 예를 들어, TZ_OFFSET 함수가 -08:00을 반환하면 명령을 실행한 위치가 UTC로부터 8시간 뒤의 시간대라고 반환 값을 해석할 수 있습니다. 유효한 시간대 이름, UTC와의 시간대 오프셋(자신을 반환함) 또는 키워드인 SESSIONTIMEZONE 또는 DBTIMEZONE을 입력할 수 있습니다. TZ_OFFSET 함수의 구문은 다음과 같습니다.

```
TZ_OFFSET ( ['time_zone_name'] '[+ | -] hh:mm' )
[ SESSIONTIMEZONE ] [DBTIMEZONE])
```

슬라이드의 예제는 다음과 같이 해석할 수 있습니다.

- 'US/Eastern' 시간대는 UTC보다 4시간이 늦습니다.
- 'Canada/Yukon' 시간대는 UTC보다 7시간 늦습니다.
- 'Europe/London' 시간대는 UTC보다 1시간 빠릅니다.

유효한 시간대 이름 값의 목록을 보려면 V\$TIMEZONE_NAMES 동적 성능 뷰를 질의합니다.

```
DESC V$TIMEZONE_NAMES
```

Name	Null?	Type
TZNAME		VARCHAR2(64)
TZABBREV		VARCHAR2(64)

TZ_OFFSET(계속)

SELECT * FROM V\$TIMEZONE_NAMES;

TZNAME	TZABBREV
Africa/Cairo	LMT
Africa/Cairo	EET
Africa/Cairo	EEST
Africa/Tripoli	LMT
Africa/Tripoli	CET
Africa/Tripoli	CEST
Africa/Tripoli	EET
America/Adak	LMT
America/Adak	NST
America/Adak	NWWT
America/Adak	BST
America/Adak	BDT
America/Adak	HAST
America/Adak	HADT
TZNAME	TZABBREV
America/Anchorage	LMT
America/Anchorage	CAT
America/Anchorage	CAWT
America/Anchorage	AHST
America/Anchorage	AHDT
America/Anchorage	AKST
■ ■ ■	
W-SU	MDST
W-SU	S
W-SU	MSD
W-SU	MSK
W-SU	EET
W-SU	EEST
WET	WEST
WET	WET

616 rows selected.

CURRENT_DATE

- 해당 세션의 시간대의 현재 날짜와 시간을 표시합니다.

```
ALTER SESSION  
SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
```

```
ALTER SESSION SET TIME_ZONE = '-5:0';  
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

SESSIONTIMEZONE	CURRENT_DATE
05:00	03-OCT-2001 09:37:06

```
ALTER SESSION SET TIME_ZONE = '-8:0';  
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

SESSIONTIMEZONE	CURRENT_DATE
08:00	03-OCT-2001 06:38:07

- CURRENT_DATE는 세션 시간대에 따라 달라집니다.
- 반환 값은 그레고리력의 날짜입니다.

ORACLE

16-3

Copyright © Oracle Corporation, 2001. All rights reserved.

CURRENT_DATE

CURRENT_DATE 함수는 해당 세션 시간대의 현재 날짜를 반환합니다. 반환 값은 그레고리력의 날짜입니다.

슬라이드 예제는 CURRENT_DATE가 세션 시간대에 따라 변한다는 것을 보여줍니다. 첫번째 예제에서 TIME_ZONE 파라미터를 -5:0으로 설정하여 세션을 변경합니다. TIME_ZONE 파라미터는 현재 SQL 세션에 대한 기본 지역 시간대 범위 값을 지정합니다. TIME_ZONE은 초기화 파라미터가 아니라 세션 파라미터입니다. TIME_ZONE 파라미터는 다음과 같이 설정 됩니다.

```
TIME_ZONE = '[+ | -] hh:mm'
```

포맷 마스크([+ | -] hh:mm)는 UTC와의 시간 차이를 시와 분으로 나타냅니다.

두번째 예제에서 TIME_ZONE 파라미터 값을 -8:0으로 변경하여 출력해 보면 CURRENT_DATE의 값이 변경되어 있음을 알 수 있습니다.

참고: ALTER SESSION 명령은 세션의 날짜 형식을 'DD-MON-YYYY HH24:MI:SS'로 설정합니다. 이는 일(1-31)-월 이름 약어-4자리 연도 시(0-23):분(0-59):초(0-59)를 나타냅니다.

CURRENT_TIMESTAMP

- 세션 시간대의 현재 날짜와 소수점 이하 초 단위의 시간을 표시합니다.

```
ALTER SESSION SET TIME_ZONE = '-5:0';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP
FROM DUAL;
```

SESSIONTIMEZONE	CURRENT_TIMESTAMP
-05:00	03-OCT-01 09.40.59.000000 AM -05:00

```
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP
FROM DUAL;
```

SESSIONTIMEZONE	CURRENT_TIMESTAMP
-08:00	03-OCT-01 06.41.38.000000 AM -08:00

- CURRENT_TIMESTAMP는 세션 시간대에 따라 달라집니다.
- 반환 값의 데이터 유형은 TIMESTAMP WITH TIME ZONE입니다.

ORACLE

16-3

Copyright © Oracle Corporation, 2001. All rights reserved.

CURRENT_TIMESTAMP

CURRENT_TIMESTAMP 함수는 해당 세션 시간대의 현재 날짜와 시간을 TIMESTAMP WITH TIME ZONE 데이터 유형의 값으로 반환합니다. 시간대 변위 값에 SQL 세션의 현재 지역 시간이 반영됩니다. CURRENT_TIMESTAMP 함수의 구문은 다음과 같습니다.

CURRENT_TIMESTAMP (*precision*)

여기서 *precision*은 반환되는 시간 값에서 소수점 이하 초의 정밀도를 지정하는 선택적인 인수입니다. 이 값을 생략할 경우 기본값은 6입니다.

슬라이드 예제는 CURRENT_TIMESTAMP가 세션 시간대에 따라 변한다는 것을 보여줍니다. 첫번째 예제에서 TIME_ZONE 파라미터를 -5:0으로 설정하여 세션을 변경합니다. 두번째 예제에서 TIME_ZONE 파라미터 값이 -8:0으로 변경된 다음 출력해 보면 CURRENT_TIMESTAMP의 값이 변경되어 있음을 알 수 있습니다.

LOCALTIMESTAMP

- 세션 시간대의 현재 날짜와 시간을 TIMESTAMP 데이터 유형의 값으로 표시합니다.

```
ALTER SESSION SET TIME_ZONE = '-5:0';
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP
FROM DUAL;
```

CURRENT_TIMESTAMP	LOCALTIMESTAMP
03-OCT-01 09:44:21.000000 AM -05:00	03-OCT-01 09:44:21.000000 AM

```
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP
FROM DUAL;
```

CURRENT_TIMESTAMP	LOCALTIMESTAMP
03-OCT-01 06:45:21.000001 AM -08:00	03-OCT-01 06:45:21.000001 AM

- CURRENT_TIMESTAMP가 TIMESTAMP WITH TIMEZONE 값을 반환하는 반면 LOCALTIMESTAMP는 TIMESTAMP 값을 반환합니다.

ORACLE

16-10

Copyright © Oracle Corporation, 2001. All rights reserved.

LOCALTIMESTAMP

LOCALTIMESTAMP 함수는 해당 세션 시간대의 현재 날짜와 시간을 TIMESTAMP 데이터 유형의 값으로 반환합니다. 이 함수와 CURRENT_TIMESTAMP의 차이점은 CURRENT_TIMESTAMP가 TIMESTAMP WITH TIMEZONE 값을 반환하는 반면 LOCALTIMESTAMP는 TIMESTAMP 값을 반환한다는 것입니다. TIMESTAMP WITH TIMEZONE은 TIMESTAMP 값에 시간대 범위 값을 포함시킨 TIMESTAMP의 변형입니다. 시간대 범위 값은 지역 시간과 UTC 사이의 차이(시와 분)입니다. TIMESTAMP WITH TIMEZONE 데이터 유형의 형식은 다음과 같습니다.

TIMESTAMP [(fractional_seconds_precision)] WITH TIMEZONE

여기서 *fractional_seconds_precision*은 SECOND datetime 필드의 소수점 이하 초 부분의 자릿수를 0부터 9까지 범위의 숫자를 사용하여 선택적으로 지정합니다. 기본값은 6입니다. 예를 들어, TIMESTAMP WITH TIMEZONE을 다음과 같은 리터럴로 지정할 수 있습니다.

TIMESTAMP '1997-01-31 09:26:56.66 +02:00'

LOCAL_TIMESTAMP 함수의 구문은 다음과 같습니다.

LOCAL_TIMESTAMP (TIMESTAMP_precision)

여기서 *TIMESTAMP_precision*은 반환되는 TIMESTAMP 값에서 소수점 이하 초의 정밀도를 지정하는 선택적인 인수입니다.

슬라이드 예제는 LOCALTIMESTAMP와 CURRENT_TIMESTAMP의 차이점을 보여줍니다. CURRENT_TIMESTAMP는 시간대 값을 표시하지만 LOCALTIMESTAMP는 이를 표시하지 않음을 알 수 있습니다.

DBTIMEZONE 및 SESSIONTIMEZONE

- 데이터베이스의 시간대 값을 표시합니다.

```
SELECT DBTIMEZONE FROM DUAL;
```

DBTIMEZONE
-05:00

- 세션의 시간대 값을 표시합니다.

```
SELECT SESSIONTIMEZONE FROM DUAL;
```

SESSIONTIMEZONE
-06:00

ORACLE

16-11

Copyright © Oracle Corporation, 2001. All rights reserved.

DBTIMEZONE 및 SESSIONTIMEZONE

기본 데이터베이스 시간대는 운영 체제의 시간대와 같습니다. CREATE DATABASE 문에 SET TIME_ZONE 절을 지정하여 데이터베이스의 기본 시간대를 설정할 수 있습니다. 이 절을 생략하면 기본 데이터베이스 시간대는 운영 체제 시간대와 같아집니다. 데이터베이스 시간대는 세션에 대해 ALTER SESSION 문을 사용하여 변경할 수 있습니다.

DBTIMEZONE 함수는 데이터베이스의 시간대 값을 반환합니다. 반환 유형은 시간대 오프셋('[+|-] TZH:TZM' 형식의 문자 유형) 또는 시간대 지역 이름입니다. 이는 사용자가 가장 최근에 CREATE DATABASE 또는 ALTER DATABASE 문을 사용하여 데이터베이스 시간대 값을 지정한 방법에 따라 달라집니다. 슬라이드의 예제는 TIME_ZONE 파라미터가 다음 형식으로 되어 있으므로 데이터베이스 시간대가 UTC로 설정되어 있음을 보여줍니다.

```
TIME_ZONE = '[+ | -] hh:mm'
```

SESSIONTIMEZONE 함수는 현재 세션 시간대의 값을 반환합니다. 반환 유형은 시간대 오프셋('[+|-] TZH:TZM' 형식의 문자 유형) 또는 시간대 지역 이름입니다. 이는 사용자가 가장 최근에 ALTER SESSION 문을 사용하여 세션 시간대 값을 지정한 방법에 따라 달라집니다. 슬라이드의 예제는 세션 시간대가 UTC로 설정되어 있음을 보여줍니다.

데이터베이스 시간대가 현재 세션의 시간대와 다르다는 것을 알 수 있습니다.

EXTRACT

- SYSDATE에서 YEAR 구성 요소를 표시합니다.

```
SELECT EXTRACT (YEAR FROM SYSDATE) FROM DUAL;
```

```
EXTRACT(YEARFROMSYSDATE)
```

```
2001
```

- MANAGER_ID가 100인 사원의 HIRE_DATE에서 MONTH 구성 요소를 표시합니다.

```
SELECT last_name, hire_date,  
       EXTRACT (MONTH FROM HIRE_DATE)  
  FROM employees  
 WHERE manager_id = 100;
```

LAST_NAME	HIRE_DATE	EXTRACT(MONTHFROMHIRE_DATE)
Kochhar	21-SEP-89	9
De Haan	13-JAN-93	1
Mourgos	16-NOV-99	11
Zlotkey	29-JAN-00	1
Hartstein	17-FEB-96	2

ORACLE

16-12

Copyright © Oracle Corporation, 2001. All rights reserved.

EXTRACT

EXTRACT 식은 datetime 또는 간격 값 표현식에 지정된 datetime 필드의 값을 추출하여 반환합니다. EXTRACT 함수를 사용하여 다음 구문에 언급된 구성 요소를 추출할 수 있습니다. EXTRACT 함수의 구문은 다음과 같습니다.

```
SELECT EXTRACT ([YEAR] [MONTH] [DAY] [HOUR] [MINUTE] [SECOND]  
                  [TIMEZONE_HOUR] [TIMEZONE_MINUTE]  
                  [TIMEZONE_REGION] [TIMEZONE_ABBR]  
  FROM  [datetime_value_expression]  
        [interval_value_expression]);
```

TIMEZONE_REGION 또는 TIMEZONE_ABBR(abbreviation)을 추출하면 적절한 시간대 이름 또는 약어를 포함하는 문자열 값이 반환됩니다. 다른 값을 추출하면 그雷고리력의 값이 반환됩니다. datetime에서 시간대 값을 추출하면 UTC의 값이 반환됩니다. 시간대 이름과 해당 약어 목록을 보려면 V\$TIMEZONE_NAMES 동적 성능 뷰를 질의합니다. 슬라이드의 첫 번째 예제는 EXTRACT 함수를 사용하여 SYSDATE에서 YEAR를 추출합니다.

슬라이드의 두 번째 예제는 EMPLOYEE_ID가 100인 관리자에게 보고를 하는 사원에 대해 EXTRACT 함수를 사용하여 EMPLOYEES 테이블의 HIRE_DATE 열에서 MONTH를 추출합니다.

FROM_TZ를 사용하여 TIMESTAMP 변환

- TIMESTAMP 값 '2000-03-28 08:00:00' 을 TIMESTAMP WITH TIME ZONE 값으로 표시합니다.

```
SELECT FROM_TZ(TIMESTAMP  
  '2000-03-28 08:00:00', '3:00')  
FROM DUAL;
```

```
FROM_TZ(TIMESTAMP '2000-03-28 08:00:00', '3:00')  
28-MAR-00 08:00:00 00000000 AM +03:00
```

- TIMESTAMP 값 '2000-03-28 08:00:00' 을 'Australia/North' 시간대 지역에 대한 TIMESTAMP WITH TIME ZONE 값으로 표시합니다.

```
SELECT FROM_TZ(TIMESTAMP  
  '2000-03-28 08:00:00', 'Australia/North')  
FROM DUAL;
```

```
FROM_TZ(TIMESTAMP '2000-03-28 08:00:00', 'AUSTRALIA/NORTH')  
28-MAR-00 08:00:00 00000000 AM AUSTRALIA/NORTH
```

ORACLE

16-13

Copyright © Oracle Corporation, 2001. All rights reserved.

FROM_TZ를 사용하여 TIMESTAMP 변환

FROM_TZ 함수는 TIMESTAMP 값을 TIMESTAMP WITH TIME ZONE 값으로 변환합니다.

FROM_TZ 함수의 구문은 다음과 같습니다.

```
FROM_TZ(TIMESTAMP timestamp_value, time_zone_value)
```

여기서 *time_zone_value*는 'TZH:TZM' 형식의 문자열이거나 선택적인 TZD(TZD는 일광 절약 시간 정보가 들어 있는 시간대 문자열 약어) 형식과 함께 TZR(시간대 지역)의 문자열을 반환하는 문자식입니다. TZR은 datetime 입력 문자열에 있는 시간대 지역을 나타냅니다. 예를 들면 'Australia/North', 'UTC', 'Singapore' 등이 있습니다. TZD는 일광 절약 시간 정보와 함께 시간대 지역을 약어 형태로 나타냅니다. 예를 들면 'PST'는 US/Pacific 표준시를 나타내며 'PDT'는 US/Pacific 일광 절약 시간을 나타냅니다. TZR 및 TZD 형식 요소에 대해 유효한 값 목록을 보려면 V\$TIMEZONE_NAMES 동적 성능 뷰를 질의합니다.

슬라이드의 예제는 TIMESTAMP 값을 TIMESTAMP WITH TIME ZONE으로 변환합니다.

TO_TIMESTAMP 및 TO_TIMESTAMP_TZ를 사용하여 STRING을 TIMESTAMP로 변환

- 문자열 '2000-12-01 11:00:00'을 TIMESTAMP 값으로 표시합니다.

```
SELECT TO_TIMESTAMP ('2000-12-01 11:00:00',
                      'YYYY-MM-DD HH:MI:SS')
FROM DUAL;
```

```
TO_TIMESTAMP('2000-12-01 11:00:00','YYYY-MM-DD HH:MI:SS')
01-DEC-00 11:00:00.000000000 AM
```

- 문자열 '1999-12-01 11:00:00 -8:00'을 TIMESTAMP WITH TIME ZONE 값으로 표시합니다.

```
SELECT
      TO_TIMESTAMP_TZ('1999-12-01 11:00:00 -8:00',
                       'YYYY-MM-DD HH:MI:SS TZH:TZM')
FROM DUAL;
```

```
TO_TIMESTAMP_TZ('1999-12-01 11:00:00','YYYY-MM-DD HH:MI:SS TZH:TZM')
01-DEC-99 11:00:00.000000000 AM -08:00
```

ORACLE

16-14

Copyright © Oracle Corporation, 2001. All rights reserved.

TO_TIMESTAMP 및 TO_TIMESTAMP_TZ를 사용하여 STRING을 TIMESTAMP로 변환

TO_TIMESTAMP 함수는 CHAR, VARCHAR2, NCHAR 또는 NVARCHAR2 데이터 유형의 문자열을 TIMESTAMP 데이터 유형의 값으로 변환합니다. TO_TIMESTAMP 함수의 구문은 다음과 같습니다.

TO_TIMESTAMP (char, [fmt], ['nlsparam'])

선택적인 인수 *fmt*은 *char*의 형식을 지정합니다. *fmt*를 생략하려면 문자열이 TIMESTAMP 데이터 유형의 기본 형식이어야 합니다. 선택적인 인수 *nlsparam*은 반환되는 월, 일의 이름 및 약어에 사용되는 언어를 지정합니다. 이 인수는 다음과 같은 형태가 될 수 있습니다.

'NLS_DATE_LANGUAGE = language'

*nlsparams*를 생략하면 이 함수는 현재 세션의 기본 날짜 언어를 사용합니다. 슬라이드의 예제는 문자열을 TIMESTAMP의 값으로 변환합니다.

TO_TIMESTAMP_TZ 함수는 CHAR, VARCHAR2, NCHAR 또는 NVARCHAR2 데이터 유형의 문자열을 TIMESTAMP WITH TIME ZONE 데이터 유형의 값으로 변환합니다. TO_TIMESTAMP_TZ 함수의 구문은 다음과 같습니다.

TO_TIMESTAMP_TZ (char, [fmt], ['nlsparam'])

선택적인 인수 *fmt*은 *char*의 형식을 지정합니다. *fmt*를 생략하려면 문자열이 TIMESTAMP WITH TIME ZONE 데이터 유형의 기본 형식이어야 합니다. 선택적인 인수 *nlsparam*은 TO_TIMESTAMP 함수에서와 동일하게 사용됩니다. 슬라이드의 예제는 문자열을 TIMESTAMP WITH TIME ZONE의 값으로 변환합니다.

참고: TO_TIMESTAMP_TZ 함수는 문자열을 TIMESTAMP WITH LOCAL TIME ZONE으로 변환하지는 않습니다.

TO_YMINTERVAL을 사용하여 시간 간격 변환

- DEPARTMENT_ID가 20인 부서에서 일하는 사원에 대해 입사일에서 1년 2개월 후의 날짜를 표시합니다.

```
SELECT hire_date,          1년 2개월
      hire_date + TO_YMINTERVAL('01-02') AS
      HIRE_DATE_YMININTERVAL
FROM EMPLOYEES
WHERE department_id = 20;
```

HIRE_DATE	HIRE_DATE_YMININTERVAL
17-FEB-1996 00:00:00	17-APR-1997 00:00:00
17-AUG-1997 00:00:00	17-OCT-1998 00:00:00

ORACLE

16-15

Copyright © Oracle Corporation, 2001. All rights reserved.

TO_YMINTERVAL을 사용하여 시간 간격 변환

TO_YMINTERVAL 함수는 CHAR, VARCHAR2, NCHAR 또는 NVARCHAR2 데이터 유형의 문자열을 INTERVAL YEAR TO MONTH 데이터 유형으로 변환합니다. INTERVAL YEAR TO MONTH 데이터 유형은 YEAR 및 MONTH datetime 필드를 사용하여 시간 간격을 저장합니다. INTERVAL YEAR TO MONTH의 형식은 다음과 같습니다.

INTERVAL YEAR [(year_precision)] TO MONTH

여기서 *year_precision*은 YEAR datetime 필드의 자릿수이며 기본값은 2입니다.
TO_YMINTERVAL 함수의 구문은 다음과 같습니다.

TO_YMINTERVAL (char)

여기서 *char*는 변환할 문자열입니다.

슬라이드의 예제는 EMPLOYEES 테이블에서 부서 ID가 20인 사원에 대해 입사일로부터 1년 2개월 후의 날짜를 계산합니다.

TO_YMINTERVAL 함수를 사용하여 다음과 같이 반대로 계산할 수도 있습니다.

```
SELECT hire_date, hire_date + TO_YMINTERVAL('-02-04') AS
      HIRE_DATE_YMININTERVAL
FROM employees WHERE department_id = 20;
```

이 경우에는 TO_YMINTERVAL 함수에 음수 값의 문자열을 전달합니다. 예제에서는 EMPLOYEES 테이블에서 부서 ID가 20인 사원에 대해 입사일로부터 2년 4개월 전의 날짜를 반환합니다.

요약

이 단원에서 다음 함수를 사용하는 방법에 대해 배웠습니다.

- **TZ_OFFSET**
- **FROM_TZ**
- **TO_TIMESTAMP**
- **TO_TIMESTAMP_TZ**
- **TO_YMINTERVAL**
- **CURRENT_DATE**
- **CURRENT_TIMESTAMP**
- **LOCALTIMESTAMP**
- **DBTIMEZONE**
- **SESSIONTIMEZONE**
- **EXTRACT**

ORACLE

16-16

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

이 단원에서는 Oracle9i에 새로 도입된 **datetime** 함수 몇 가지를 설명했습니다.

연습 16 개요

이 연습에서는 Oracle9i datetime 함수 사용을 다릅니다.

ORACLE

16-17

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 16 개요

이 연습에서는 시간대 오프셋, CURRENT_DATE, CURRENT_TIMESTAMP 및 LOCALTIMESTAMP를 표시합니다. 또한 시간대를 설정하고 EXTRACT 함수를 사용합니다.

연습 16

1. NLS_DATE_FORMAT을 DD-MON-YYYY HH24:MI:SS로 설정하여 세션을 변경하십시오.
2. a. 다음 시간대에 대해 시간대 오프셋(TZ_OFFSET)을 표시하는 질의를 작성하십시오.
 - US/Pacific-New

TZ_OFFSET
-07:00

- Singapore

TZ_OFFSET
+08:00

- Egypt

TZ_OFFSET
+02:00

- b. TIME_ZONE 파라미터 값을 US/Pacific-New의 시간대 오프셋으로 설정하여 세션을 변경하십시오.

- c. 이 세션에 대한 CURRENT_DATE, CURRENT_TIMESTAMP 및 LOCALTIMESTAMP를 표시하십시오.

참고: 명령을 실행하는 날짜에 따라 결과가 다를 수 있습니다.

CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
01-OCT-2001 13:40:54	01-OCT-01 01.40.54.000001 PM -07:00	01-OCT-01 01.40.54.000001 PM

- d. TIME_ZONE 파라미터 값을 Singapore의 시간대 오프셋으로 설정하여 세션을 변경하십시오.

- e. 이 세션에 대한 CURRENT_DATE, CURRENT_TIMESTAMP 및 LOCALTIMESTAMP를 표시하십시오. 참고: 명령을 실행하는 날짜에 따라 결과가 다를 수 있습니다.

CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
02-OCT-2001 04:42:34	02-OCT-01 04.42.34.000000 AM +08:00	02-OCT-01 04.42.34.000000 AM

참고: 앞의 연습에서 CURRENT_DATE, CURRENT_TIMESTAMP 및 LOCALTIMESTAMP가 모두 세션 시간대에 따라 달라진다는 점을 알 수 있습니다.

3. DBTIMEZONE 및 SESSIONTIMEZONE을 표시하는 질의를 작성하십시오.

DBTIME	SESSIONTIMEZONE
-05:00	+08:00

연습 16(계속)

4. 부서 ID가 80인 사원에 대해 EMPLOYEES 테이블의 HIRE_DATE 열에서 YEAR를 추출하는
질의를 작성하십시오.

LAST_NAME	EXTRACT(YEARFROMHIRE_DATE)
Zlotkey	2000
Abel	1996
Taylor	1998

5. NLS_DATE_FORMAT을 DD-MON-YYYY로 설정하여 세션을 변경하십시오.

17

GROUP BY 절의
향상된 기능

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- ROLLUP 연산을 사용하여 하위 총계 값 산출
- CUBE 연산을 사용하여 교차 도표화 값 산출
- GROUPING 함수를 사용하여 ROLLUP 또는 CUBE를 통해 만들어진 행 값 식별
- GROUPING SETS를 사용하여 하나의 결과 값 산출

ORACLE

17-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 다음 작업을 수행하는 방법을 배웁니다.

- 데이터를 그룹화하여 다음 결과를 산출합니다.
 - ROLLUP 연산자를 사용한 하위 총계 값
 - CUBE 연산자를 사용한 교차 도표화 값
- GROUPING 함수를 사용하여 ROLLUP 또는 CUBE 연산자를 통해 산출된 결과 집합에서 집계 레벨을 식별합니다.
- GROUPING SETS를 사용하면 UNION ALL을 사용한 결과와 동일한 단일 결과 집합이 산출됩니다.

그룹 함수 복습

그룹 함수는 행 집합에 작용하여 그룹 당 하나의 결과를 생성합니다.

```
SELECT      [column,] group_function(column) . . .
FROM        table
[WHERE       condition]
[GROUP BY   group_by_expression]
[ORDER BY   column];
```

예제:

```
SELECT AVG(salary), STDDEV(salary),
       COUNT(commission_pct), MAX(hire_date)
  FROM employees
 WHERE job_id LIKE 'SA%';
```

ORACLE

17-3

Copyright © Oracle Corporation, 2001. All rights reserved.

그룹 함수

GROUP BY 절을 사용하면 테이블의 행을 여러 그룹으로 나눌 수 있습니다. 그런 다음 그룹 함수를 사용하면 각 그룹에 대한 요약 정보를 반환할 수 있습니다. 그룹 함수는 select 목록에서 사용할 수도 있고 ORDER BY 및 HAVING 절에서 사용할 수도 있습니다. Oracle Server에서는 그룹 함수를 각 행 그룹에 적용한 다음 각 그룹에 대해 하나의 결과 행을 반환합니다.

그룹 함수 종류

그룹 함수인 AVG, SUM, MAX, MIN, COUNT, STDDEV 및 VARIANCE는 모두 하나의 인수를 받아 들입니다. AVG, SUM, STDDEV 및 VARIANCE 함수는 숫자 값에만 사용할 수 있으며 MAX 및 MIN 함수는 숫자, 문자 또는 날짜 데이터 값에 사용할 수 있습니다. COUNT 함수는 주어진 표현식에 대해 널이 아닌 숫자 행을 반환합니다. 슬라이드의 예제는 JOB_ID가 SA로 시작하는 사원에 대해 평균 급여, 급여의 표준 편차, 커미션을 받는 사원의 수 및 가장 최근 고용일을 계산합니다.

그룹 함수 사용 지침

- 인수로 사용할 수 있는 데이터 유형은 CHAR, VARCHAR2, NUMBER 또는 DATE입니다.
- COUNT(*)를 제외한 모든 그룹 함수는 널 값을 무시합니다. 널 값을 특정 값으로 치환 하려면 NVL 함수를 사용하십시오. COUNT는 숫자 또는 0을 반환합니다.
- GROUP BY 절을 사용하면 암시적(implicit)으로 Oracle Server에서 결과 집합을 지정된 그룹화 열에 대해 오름차순으로 정렬합니다. 기본 순서를 무시하고 내림차순으로 정렬 하려면 ORDER BY 절에 DESC를 사용하십시오.

GROUP BY 절 복습

구문:

```
SELECT      [column,] group_function(column) . . .
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[ORDER BY   column];
```

예제:

```
SELECT      department_id, job_id, SUM(salary),
            COUNT(employee_id)
FROM        employees
GROUP BY   department_id, job_id ;
```

ORACLE

17-4

Copyright © Oracle Corporation, 2001. All rights reserved.

GROUP BY 절 복습

슬라이드의 예제는 Oracle Server에서 다음과 같이 계산됩니다.

- SELECT 절은 검색할 다음과 같은 열을 지정합니다.
 - EMPLOYEES 테이블의 부서 ID 및 업무 ID 열
 - GROUP BY 절에서 지정한 각 그룹에 대한 모든 급여의 합과 사원 수
- GROUP BY 절은 테이블에서 행을 그룹화하는 방법을 지정합니다. 총 급여와 사원 수는 각 부서 내에서 업무 ID별로 계산됩니다. 행은 부서 ID별로 그룹화된 다음 각 부서 내에서 업무별로 그룹화됩니다.

DEPARTMENT_ID	JOB_ID	SUM(SALARY)	COUNT(EMPLOYEE_ID)
10	AD_ASST	4400	1
20	MK_MAN	13000	1
20	MK_REP	6000	1
50	ST_CLERK	11700	4

110	AC_ACCOUNT	8300	1
110	AC_MGR	12000	1
	SA_REP	7000	1

13 rows selected.

HAVING 절 복습

```
SELECT      [column,] group_function(column)...
FROM        table
[WHERE       condition]
[GROUP BY   group_by_expression]
[HAVING     having_expression]
[ORDER BY   column];
```

- HAVING 절을 사용하여 표시할 그룹을 지정합니다.
- 제한 조건을 사용하여 그룹을 더욱 제한할 수 있습니다.

ORACLE

17-5

Copyright © Oracle Corporation, 2001. All rights reserved.

HAVING 절

그룹이 형성되고 그룹 함수가 계산된 후 그룹에 HAVING 절이 적용됩니다. HAVING 절이 GROUP BY 절 앞에 올 수는 있지만 GROUP BY 절을 먼저 두는 것이 더 논리적이므로 이를 권장합니다.

HAVING 절을 사용하면 Oracle Server에서 다음 단계를 수행합니다.

1. 행 그룹화
2. 그룹 함수를 그룹에 적용한 다음 HAVING 절의 조건에 맞는 그룹을 표시합니다.

```
SELECT department_id, AVG(salary)
FROM   employees
GROUP  BY department_id
HAVING AVG(salary) >9500;
```

DEPARTMENT_ID	AVG(SALARY)
80	10033 3333
90	19333 3333
110	10150

예제에서는 평균 급여가 \$9,500보다 큰 부서에 대한 부서 ID와 평균 급여를 표시합니다.

8-4 사용 가능 GROUP BY에 ROLLUP 및 CUBE 연산자 사용

- ROLLUP 또는 CUBE를 GROUP BY에 사용하여 상호 참조 열에 따라 상위 집계 행을 산출합니다.
- ROLLUP 그룹화는 정규 그룹화 행과 하위 총계 값을 포함하는 결과 집합을 산출합니다.
- CUBE 그룹화는 ROLLUP의 결과 행 및 교차 도표화 행을 포함하는 결과 집합을 산출합니다.

ORACLE

17-6

Copyright © Oracle Corporation, 2001. All rights reserved.

GROUP BY에 ROLLUP 및 CUBE 연산자 사용

ROLLUP 및 CUBE 연산자를 질의의 GROUP BY 절에 지정할 수 있습니다. ROLLUP 그룹화는 정규 그룹화 행과 하위 총계 값을 포함하는 결과 집합을 산출합니다. CUBE 연산을 GROUP BY 절에서 수행하면 지정된 표현식에서 가능한 모든 조합 값에 따라 선택된 행이 그룹화되고 각 그룹에 대한 요약 정보를 나타내는 하나의 행이 반환됩니다. CUBE 연산자를 사용하면 교차 도표화 행을 산출할 수 있습니다.

참고: ROLLUP 및 CUBE를 사용하여 작업할 때는 GROUP BY 절 뒤에 나오는 열이 실질적으로 서로 연관성이 있는 열인지 확인해야 합니다. 서로 관계가 없는 열을 사용하면 아무 의미가 없는 정보가 반환됩니다.

ROLLUP 및 CUBE 연산자는 Oracle8i 이후 릴리스에서만 사용할 수 있습니다.

ROLLUP 연산자

```
SELECT      [column,] group_function(column) . . .
FROM        table
[WHERE      condition]
[GROUP BY   [ROLLUP] group_by_expression]
[HAVING    having_expression];
[ORDER BY   column];
```

- ROLLUP은 GROUP BY 절의 확장 기능입니다.
- ROLLUP 연산을 사용하면 하위 총계와 같은 누적
집계를 산출할 수 있습니다.

ORACLE

17-7

Copyright © Oracle Corporation, 2001. All rights reserved.

ROLLUP 연산자

ROLLUP 연산자는 GROUP BY 문의 표현식에 대한 집계 및 상위 집계를 표시합니다. ROLLUP 연산자는 보고서를 작성할 때 결과 집합에서 통계 및 요약 정보를 추출하는 데 사용할 수 있습니다. 누적 집계는 보고서, 차트 및 그래프에 사용할 수 있습니다.

ROLLUP 연산자는 GROUP BY 절에 지정된 열 목록을 따라 오른쪽에서 왼쪽 방향으로 하나씩 그룹을 만듭니다. 그런 다음 집계 함수를 이러한 그룹에 적용합니다.

참고: ROLLUP 연산자 없이 n (GROUP BY 절의 열 수)차원의 하위 총계를 산출하려면 $n+1$ 개의 SELECT 문을 UNION ALL로 연결해야 합니다. 그러면 모든 SELECT 문이 각각 테이블에 액세스하므로 질의가 비효율적으로 실행됩니다. ROLLUP 연산자는 단 한 번 테이블에 액세스하여 해당 결과를 취합합니다. ROLLUP 연산자는 하위 총계를 산출하는 데 필요한 열이 많은 경우 유용합니다.

ROLLUP 연산자 예제

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 60
GROUP BY ROLLUP(department_id, job_id);
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
10		4400
20	MK_MAN	13000
20	MK_REP	6000
20		19000
50	ST_CLERK	11700
50	ST_MAN	5800
50		17500
50		40900

1

2

3

9 rows selected.

ORACLE

17-8

Copyright © Oracle Corporation, 2001. All rights reserved.

ROLLUP 연산자 예제

슬라이드 예제 설명:

- 부서 ID가 60보다 작은 부서에 대해 해당 부서에 속한 업무 ID별 총 급여를 GROUP BY 절을 사용하여 표시합니다(1).
- ROLLUP 연산자를 사용하면 다음과 같은 내용이 표시됩니다.
 - 부서 ID가 60보다 작은 부서에 대한 부서별 총 급여(2)
 - 업무 ID와 상관 없이 부서 ID가 60보다 작은 부서에 대한 전체 총 급여(3)
- 1로 표시된 행은 정규 행이며 2와 3으로 표시된 행은 상위 집계 행입니다.

ROLLUP 연산자는 GROUP BY 절에 지정된 그룹화 목록에 따라 가장 하위 레벨부터 시작하여 최상위 총계까지 차례로 계산합니다. 먼저 GROUP BY 절에 지정된 그룹에 대해 표준 집계 값을 계산합니다. 이 예제의 경우 부서 내 각 업무별로 그룹화된 급여의 합계입니다. 그런 다음 그룹화 열 목록을 따라 오른쪽에서 왼쪽 방향으로 이동하면서 한 레벨씩 높은 하위 총계를 만들어 갑니다. 이 예제의 경우 각 부서에 대한 급여의 합을 계산한 다음 모든 부서에 대한 급여의 합을 계산합니다.

- GROUP BY 절의 ROLLUP 연산자에 n 개의 표현식이 있으면 연산 결과는 $n+1 = 2 + 1 = 3$ 번 그룹화됩니다.
- 처음 n 개 표현식의 값에 따라 표시된 행을 행 또는 정규 행이라고 하며 다른 행을 상위 집계 행이라고 합니다.

CUBE 연산자

```
SELECT      [column,] group_function(column) ...
FROM        table
[WHERE       condition]
[GROUP BY   [CUBE] group_by_expression]
[HAVING      having_expression]
[ORDER BY   column];
```

- * CUBE는 GROUP BY 절의 확장 기능입니다.
- * CUBE 연산자를 사용하면 하나의 SELECT 문으로 교차
도표화(cross tabulation) 값을 산출할 수 있습니다.

ORACLE

17-3

Copyright © Oracle Corporation, 2001. All rights reserved.

CUBE 연산자

CUBE 연산자는 SELECT 문의 GROUP BY 절에 사용할 수 있는 또 다른 전환 기능입니다. CUBE 연산자는 AVG, SUM, MAX, MIN 및 COUNT를 포함한 모든 집계 함수에 적용할 수 있으며 일반적으로 교차 표 형식 보고서에 사용되는 결과 집합을 산출하는 데 사용됩니다. ROLLUP은 가능한 하위 총계 조합 중 일부만 산출하지만 CUBE는 GROUP BY 절에 지정된 모든 그룹의 조합에 대해 하위 총계와 최상위 총계를 산출합니다.

CUBE 연산자를 집계 함수에 사용하면 결과 집합에 추가 행이 만들어집니다. GROUP BY 절에 포함된 열이 상호 참조되어 그룹의 대집합이 산출됩니다. select 목록에 지정된 집계 함수가 이러한 그룹에 적용되어 추가 상위 집계 행에 대한 요약 값이 산출됩니다. 결과 집합의 추가 그룹 수는 GROUP BY 절에 포함된 열 수에 따라 결정됩니다.

실제적으로 GROUP BY 절에 있는 열 또는 표현식의 모든 조합이 상위 집계를 산출하는 데 사용됩니다. GROUP BY 절에 n 개의 열 또는 표현식이 있을 경우 가능한 상위 집계 조합 수는 2^n 개입니다. 수학적으로 이러한 결합은 n 차원 큐브를 형성하며 연산자의 이름도 이에 따른 것입니다. 응용 프로그램 또는 프로그래밍 블에서 이러한 상위 집계 값을 사용하여 결과 및 관계를 시각적이면서도 효율적으로 전달하는 차트 및 그래프를 만들 수 있습니다.

group by A, B

{ group by a

group by b

group by ()

CUBE 연산자: 예제

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 60
GROUP BY CUBE (department_id, job_id);
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
10		4400
20	MK_MAN	13000
20	MK_REP	6000
20		19000
50	ST_CLERK	11700
50	ST_MAN	5800
50		17500
	AD_ASST	4400
	MK_MAN	13000
	MK_REP	6000
	ST_CLERK	11700
	ST_MAN	5800
		40300

14 rows selected.

ORACLE

17-10

Copyright © Oracle Corporation, 2001. All rights reserved.

CUBE 연산자: 예제

예제에 있는 SELECT 문의 출력은 다음과 같이 해석할 수 있습니다.

- 부서 ID가 60보다 작은 부서에 대해 해당 부서에 속한 업무별 총 급여를 GROUP BY 절을 사용하여 표시합니다(1).
- 부서 ID가 60보다 작은 부서에 대한 부서별 총 급여(2)
- 부서와 상관 없는 업무별 총 급여(3)
- 업무와 상관 없이 부서 ID가 60보다 작은 부서에 대한 전체 총 급여(4)

앞의 예제에서 1로 표시된 행은 정규 행이고 2와 4로 표시된 행은 상위 집계 행이며 3으로 표시된 행은 고차 도표화 값입니다.

CUBE 연산자는 ROLLUP 연산도 수행하여 업무에 상관 없이 부서 ID가 60보다 작은 부서에 대한 하위 총계 및 부서 ID가 60보다 작은 부서에 대한 총 급여를 표시합니다. 또한 CUBE 연산자는 부서에 상관 없이 업무별 총 급여를 표시합니다.

참고: ROLLUP 연산자의 경우와 마찬가지로 CUBE 연산자 없이 하위 총계를 n (GROUP BY 절의 열 수) 차원으로 산출하려면 2^n 개의 SELECT 문을 UNION ALL로 연결해야 합니다. 따라서 3차원 보고서를 작성하려면 $2^3 = 8$ 개의 SELECT 문을 UNION ALL로 연결해야 합니다.

GROUPING 함수

```
SELECT      [column,] group_function(column) . ,
            GROUPING(expr)
FROM        table
[WHERE      condition]
[GROUP BY  [ROLLUP] [CUBE] group_by_expression]
[HAVING    having_expression]
[ORDER BY  column];
```

- GROUPING 함수는 CUBE 또는 ROLLUP 연산자와 함께 사용합니다.
- GROUPING 함수를 사용하면 행에서 하위 총계를 형성한 그룹을 찾을 수 있습니다.
- GROUPING 함수를 사용하면 ROLLUP 또는 CUBE를 통해 만들어진 NULL 값과 저장된 NULL 값을 구별할 수 있습니다.
- GROUPING 함수는 0 또는 1을 반환합니다.

ORACLE

17-11

Copyright © Oracle Corporation, 2001. All rights reserved.

GROUPING 함수

GROUPING 함수를 CUBE 또는 ROLLUP 연산자와 함께 사용하면 요약 값을 산출한 방법을 이해하는 데 도움이 됩니다.

GROUPING 함수는 하나의 열을 인수로 사용합니다. GROUPING 함수에 있는 expr은 GROUP BY 절에 있는 표현식 중 하나와 일치해야 합니다. 이 함수는 0 또는 1을 반환합니다.

GROUPING 함수의 반환 값은 다음과 같은 작업에 유용합니다.

- 주어진 하위 총계의 집계 레벨, 즉 하위 총계를 산출한 기준이 된 그룹을 확인합니다.
- 결과 집합 행의 표현식 열에 있는 NULL 값이 다음 중 어떤 값을 나타내는지 식별합니다.
 - 기본 테이블의 NULL 값(저장된 NULL 값)
 - ROLLUP/CUBE를 통해 만들어진 NULL 값(해당 표현식에 있는 그룹 함수의 결과)

GROUPING 함수의 표현식에 따라 반환된 0의 값은 다음 중 하나를 나타냅니다.

- 표현식을 사용하여 집계 값을 계산했습니다.
- 표현식 열에 있는 NULL 값이 저장된 NULL 값입니다.

GROUPING 함수의 표현식에 따라 반환된 1의 값은 다음 중 하나를 나타냅니다.

- 표현식을 사용하지 않고 집계 값을 계산했습니다.
- 표현식 열에 있는 NULL 값이 그룹화의 결과로 ROLLUP 또는 CUBE를 통해 만들어졌습니다.

GROUPING 함수: 예제

```
SELECT department_id DEPTID, job_id JOB,
       SUM(salary),
       GROUPING(department_id) GRP_DEPT,
       GROUPING(job_id) GRP_JOB
  FROM employees
 WHERE department_id < 50
 GROUP BY ROLLUP(department_id, job_id);
```

DEPTID	JOB	SUM(SALARY)	GRP_DEPT	GRP_JOB
10	AD_ASST	4400	0	0
10		4400	0	1
20	MK_MAN	13000	0	0
20	MK_REP	6000	0	0
20		19000	0	1
		23400	1	1

6 rows selected.

ORACLE

17-12

Copyright © Oracle Corporation, 2001. All rights reserved.

GROUPING 함수: 예제

슬라이드의 예제에서 요약 값이 4400인 첫 번째 행을 살펴봅니다(1). 이 요약 값은 부서 ID 10 내에서의 업무 ID AD_ASST에 대한 총 급여입니다. 이 요약 값을 계산하려면 DEPARTMENT_ID 및 JOB_ID 열을 모두 고려해야 합니다. 따라서 GROUPING(department_id) 및 GROUPING(job_id) 표현식 모두에 대해 0의 값이 반환됩니다.

요약 값이 4400인 두 번째 행을 살펴봅니다(2). 이 값은 부서 ID 10에 대한 총 급여이며 DEPARTMENT_ID 열을 고려하여 계산됩니다. 따라서 GROUPING(department_id)에서 0의 값이 반환됩니다. 이 값을 계산하는 데 JOB_ID 열이 고려되지 않았으므로 GROUPING(job_id)에서는 1의 값이 반환됩니다. 다섯 번째 행의 출력도 마찬가지입니다.

슬라이드의 예제에서 요약 값이 23400인 마지막 행을 살펴봅니다(3). 이 값은 부서 ID가 50보다 작은 부서에 대한 전체 총 급여입니다. 이 요약 값을 계산하는 데 DEPARTMENT_ID 및 JOB_ID 열이 모두 고려되지 않습니다. 따라서 GROUPING(department_id) 및 GROUPING(job_id) 표현식 모두에 대해 1의 값이 반환됩니다.

GROUPING SETS

9. 데이터가능한

- GROUPING SETS는 GROUP BY 절의 확장 기능입니다.
- GROUPING SETS를 사용하면 같은 질의에서 여러 그룹화를 정의할 수 있습니다.
- Oracle Server에서는 GROUPING SETS 절에 지정된 모든 그룹화를 계산하고 UNION ALL 연산을 통해 각 그룹화의 결과를 결합합니다.
- 그룹화 집합을 사용하면 다음과 같은 면에서 효율적입니다.
 - 기본 테이블을 한 번만 검색합니다.
 - 복잡한 UNION 문을 작성할 필요가 없습니다.
 - GROUPING SETS에 요소가 많을수록 성능이 좋아집니다.

ORACLE

17-13

Copyright © Oracle Corporation, 2001. All rights reserved.

GROUPING SETS

GROUPING SETS는 GROUP BY 절의 확장 기능으로, 데이터를 여러 개로 그룹화하도록 지정할 수 있습니다. 이렇게 하면 집계를 효율적으로 수행할 수 있으며 따라서 여러 차원에 걸쳐 데이터를 분석하기가 용이해집니다.

이제 UNION ALL 연산자를 통해 여러 개의 SELECT 문을 결합하는 대신 GROUPING SETS를 사용하여 하나의 SELECT 문을 작성함으로써 다양한 그룹화(ROLLUP 또는 CUBE 연산자 포함)를 지정할 수 있습니다. 예를 들면 다음과 같습니다.

```
SELECT department_id, job_id, manager_id, AVG(salary)
  FROM employees
 GROUP BY GROUPING SETS
    ((department_id, job_id, manager_id),
     (department_id, manager_id), (job_id, manager_id));
```

이 명령문에서는 다음 세 그룹화에 대한 집계를 계산합니다.

```
(department_id, job_id, manager_id), (department_id, manager_id)
 및 (job_id, manager_id)
```

향상된 Oracle9i의 이 기능을 사용하지 않고 앞에 나오는 SELECT 문의 결과를 얻으려면 UNION ALL을 사용하여 여러 질의를 결합해야 합니다. 여러 질의를 사용하는 방식은 같은 데이터를 여러 번 스캔하므로 비효율적입니다.

GROUPING SETS(계속)

앞에 나오는 명령문을 다음 명령문과 비교합니다.

```
SELECT department_id, job_id, manager_id, AVG(salary)
  FROM employees
 GROUP BY CUBE(department_id, job_id, manager_id);
```

앞에 나오는 명령문은 $8(2^2 * 2^2)$ 개의 그룹화를 계산하지만 실제 관심 있는 그룹은 $(\text{department_id}, \text{job_id}, \text{manager_id})$, $(\text{department_id}, \text{manager_id})$ 및 $(\text{job_id}, \text{manager_id})$ 뿐입니다.

다음 명령문을 사용할 수도 있습니다.

```
SELECT department_id, job_id, manager_id, AVG(salary)
  FROM employees
 GROUP BY department_id, job_id, manager_id
 UNION ALL
SELECT department_id, NULL, manager_id, AVG(salary)
  FROM employees
 GROUP BY department_id, manager_id
 UNION ALL
SELECT NULL, job_id, manager_id, AVG(salary)
  FROM employees
 GROUP BY job_id, manager_id;
```

이 명령문은 기본 테이블을 세 번 스캔하므로 비효율적입니다.

CUBE 및 ROLLUP은 매우 특별한 의미를 지닌 그룹화 집합입니다. 다음 내용을 보면 이 사실을 확인할 수 있습니다.

CUBE(a, b, c) 이는 다음과 같습니다.	GROUPING SETS ((a, b, c), (a, b), (a, c), (b, c), (a), (b), (c), ())
ROLLUP(a, b, c) 이는 다음과 같습니다.	GROUPING SETS ((a, b, c), (a, b), (a), ())

GROUPING SETS: 예제

```
SELECT department_id, job_id,
       manager_id, avg(salary)
  FROM employees
 GROUP BY GROUPING SETS
 ((department_id,job_id), (job_id,manager_id));
```

DEPARTMENT_ID	JOB_ID	MANAGER_ID	AVG(SALARY)
10	AD_ASST		4400
20	MK_MAN		13000
20	MK_REP		6000
50	ST_CLERK		2925
	SA_MAN	100	10500
	SA_REP	149	8866.66667
	ST_CLERK	124	2925
	ST_MAN	100	5800

26 rows selected.

1

2

ORACLE

17-15

Copyright © Oracle Corporation, 2001. All rights reserved.

GROUPING SETS: 예제

슬라이드의 질의는 두 그룹화에 대한 집계를 계산합니다. 테이블은 다음 그룹으로 나눠집니다.

- 부서 ID, 업무 ID
- 업무 ID, 관리자 ID

해당 그룹에 대해 각각 평균 급여가 계산되어 결과 집합에 두 그룹에 대한 평균 급여가 각각 표시됩니다.

출력에서 1로 표시된 그룹을 다음과 같이 해석할 수 있습니다.

- 부서 ID가 10이고 업무 ID가 AD_ASST인 사원의 평균 급여는 4400입니다.
- 부서 ID가 20이고 업무 ID가 MK_MAN인 사원의 평균 급여는 13000입니다.
- 부서 ID가 20이고 업무 ID가 MK_REP인 사원의 평균 급여는 6000입니다.
- 부서 ID가 50이고 업무 ID가 ST_CLERK인 사원의 평균 급여는 2925입니다.

나머지도 마찬가지로 해석할 수 있습니다.

GROUPING SETS: 예제(계속)

출력에서 2로 표시된 그룹을 다음과 같이 해석할 수 있습니다.

- 관리자 ID가 201인 관리자에게 보고를 하고 업무 ID가 MK_REP인 사원의 평균 급여는 6000입니다.
- 관리자 ID가 100인 관리자에게 보고를 하고 업무 ID가 SA_MAN인 사원의 평균 급여는 10500입니다. 나머지도 마찬가지로 해석할 수 있습니다.

슬라이드의 예제를 다음과 같이 작성할 수도 있습니다.

```
SELECT      department_id, job_id, NULL as manager_id,  
           AVG(salary) as AVGSAL  
  FROM        employees  
 GROUP BY    department_id, job_id  
UNION ALL  
SELECT      NULL, job_id, manager_id, avg(salary) as AVGSAL  
  FROM        employees  
 GROUP BY    job_id, manager_id;
```

질의 블록을 검색하여 실행 계획을 생성하는 최적기가 없을 경우 앞의 질의는 기본 테이블인 EMPLOYEES를 두 번 스캔합니다. 이는 매우 비효율적이므로 GROUPING SETS 문을 사용하는 것이 좋습니다.

조합 열

- 조합 열은 한 단위로 처리되는 열의 collection입니다.
`ROLLUP (a, (b, c), d)`
- 조합 열을 지정하려면 GROUP BY 절을 사용하여 해당 열들을 괄호로 묶어 그룹화합니다. Oracle Server에서는 ROLLUP 또는 CUBE 연산을 계산하는 동안 조합 열을 한 단위로 처리합니다.
- 조합 열을 ROLLUP 또는 CUBE와 함께 사용하면 특정 레벨에 대해 집계를 건너뛴다는 것을 의미합니다.

17-17

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

조합 열

조합 열은 그룹화를 계산하는 동안 한 단위로 처리되는 열의 collection입니다. 다음 명령문에서와 같이 괄호 안에 열을 지정할 수 있습니다.

`ROLLUP (a, (b, c), d)`

여기서 (b, c)는 조합 열을 형성하며 한 단위로 처리됩니다. 일반적으로 조합 열은 ROLLUP, CUBE 및 GROUPING SETS에서 유용하게 사용됩니다. 예를 들어, CUBE 또는 ROLLUP에서 조합 열을 사용하면 특정 레벨에 대해 집계를 건너뛴다는 것을 의미합니다.

즉, `GROUP BY ROLLUP(a, (b, c))`는

다음과 같습니다.

```
GROUP BY a, b, c UNION ALL  
GROUP BY a UNION ALL  
GROUP BY ()
```

여기서 (b, c)는 한 단위로 처리되므로 (b, c)에 대해서는 ROLLUP이 적용되지 않습니다. 이는 (b, c)에 대해 별칭 z를 사용하여 GROUP BY 표현식을 GROUP BY ROLLUP(a, z)로 줄이는 것과 같습니다.

참고: GROUP BY()는 일반적으로 a 및 b 열에 대해서는 NULL 값을 가지면서 집계 함수만을 포함하는 SELECT 문입니다. 이는 대개 최상위 총계를 생성하는 데 사용됩니다.

```
SELECT NULL, NULL, aggregate_col  
FROM <table_name>  
GROUP BY ( );
```

조합 열(계속)

이것을 다음과 같은 보통 ROLLUP과 비교합니다.

GROUP BY ROLLUP(a, b, c)

이것은 다음과 같습니다.

```
GROUP BY a, b, c UNION ALL  
GROUP BY a, b UNION ALL  
GROUP BY a UNION ALL  
GROUP BY () .
```

마찬가지로

GROUP BY CUBE((a, b), c)

이것은 다음과 같습니다.

```
GROUP BY a, b, c UNION ALL  
GROUP BY a, b UNION ALL  
GROUP BY c UNION ALL  
GROUP BY ()
```

다음 표에서는 그룹화 집합 명세와 해당 GROUP BY 명세를 보여줍니다.

GROUPING SETS 문	동등한 GROUP BY 문
GROUP BY GROUPING SETS(a, b, c)	GROUP BY a UNION ALL GROUP BY b UNION ALL GROUP BY c
GROUP BY GROUPING SETS(a, b, (b, c)) (GROUPING SETS 표현식에 조합 열이 있습니다.)	GROUP BY a UNION ALL GROUP BY b UNION ALL GROUP BY b, c
GROUP BY GROUPING SETS((a, b, c))	GROUP BY a, b, c
GROUP BY GROUPING SETS(a, (b), ())	GROUP BY a UNION ALL GROUP BY b UNION ALL GROUP BY ()
GROUP BY GROUPING SETS (a, ROLLUP(b, c)) (GROUPING SETS 표현식에 조합 열이 있습니다.)	GROUP BY a UNION ALL GROUP BY ROLLUP(b, c)

조합 열: 예제

```
SELECT department_id, job_id, manager_id,
       SUM(salary)
  FROM employees
 GROUP BY ROLLUP( department_id, (job_id, manager_id));
```

DEPARTMENT_ID	JOB_ID	MANAGER_ID	SUM(SALARY)
10	AD_ASST	101	4400
10			4400
20	MK_MAN	100	13000
20	MK_REP	201	6000
20			19000
50	ST_CLERK	124	11700
...			
			175500

23 rows selected.

ORACLE

17-19 Copyright © Oracle Corporation, 2001. All rights reserved.

조합 열: 예제

다음 예제를 살펴봅니다.

```
SELECT department_id, job_id, manager_id, SUM(salary)
  FROM employees
 GROUP BY ROLLUP( department_id, job_id, manager_id);
```

앞의 질의를 실행하면 Oracle Server에서 다음 그룹화를 계산합니다.

1. (department_id, job_id, manager_id)
2. (department_id, job_id)
3. (department_id)
4. ()

앞의 예제에 있는 (1), (3) 및 (4)의 그룹화만 필요한 경우에는 조합 열을 사용해야 이러한 그룹화만 계산하도록 제한할 수 있습니다. 조합 열을 사용하면 를업하는 동안 JOB_ID 및 MANAGER_ID 열을 한 단위로 처리할 수 있습니다. 괄호로 묶인 열은 ROLLUP 및 CUBE를 계산하는 동안 한 단위로 처리됩니다. 이러한 사항이 슬라이드의 예제에 나와 있습니다. JOB_ID 및 MANAGER_ID 열을 괄호로 묶으면 Oracle Server에서 JOB_ID와 MANAGER_ID를 한 단위의 조합 열로 처리합니다.

조합 열: 예제(계속)

슬라이드의 예제는 다음 그룹화를 계산합니다.

- (department_id, job_id, manager_id)
- (department_id)
- ()

슬라이드의 예제는 다음을 표시합니다.

- 각 부서의 총 급여(1)
- 각 부서, 업무 ID 및 관리자에 대한 총 급여(2)
- 최상위 총계(3)

슬라이드의 예제를 다음과 같이 작성할 수도 있습니다.

```
SELECT      department_id, job_id, manager_id, SUM(salary)
FROM        employees
GROUP BY    department_id, job_id, manager_id
UNION ALL
SELECT      department_id, TO_CHAR(NULL), TO_NUMBER(NULL),  SUM(salary)
FROM        employees
GROUP BY    department_id
UNION ALL
SELECT      TO_NUMBER(NULL),  TO_CHAR(NULL), TO_NUMBER(NULL),  SUM(salary)
FROM        employees
GROUP BY  () ;
```

질의 블록을 검색하여 실행 계획을 생성하는 최적기가 없을 경우 앞의 질의는 기본 테이블인 EMPLOYEES를 세 번 스캔합니다. 이는 매우 비효율적이므로 조합 열을 사용하는 것이 좋습니다.

연결된 그룹화

- 연결된 그룹화를 사용하면 간단하게 유용한 그룹화를 결합할 수 있습니다.
- 연결된 그룹화 집합을 지정하려면 여러 그룹화 집합, ROLLUP 및 CUBE 연산을 쉼표로 구분함으로써 Oracle Server에서 이를 하나의 GROUP BY 절로 결합할 수 있도록 해야 합니다.
- 각 그룹화 집합으로부터 그룹화 상호 곱이 만들어져 결과로 표시됩니다.

```
GROUP BY GROUPING SETS(a, b), GROUPING SETS(c, d)
```

17-21

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

연결된 열

연결된 그룹화를 사용하면 간단하게 유용한 그룹화를 결합할 수 있습니다. 연결된 그룹화는 여러 그룹화 집합, 큐브 및 블업을 나열하고 쉼표로 구분하여 쉽게 지정할 수 있습니다. 다음은 연결된 그룹화 집합의 예제를 보여줍니다.

GROUP BY GROUPING SETS(a, b), GROUPING SETS(c, d)

앞의 SQL에서는 다음 그룹화를 정의합니다.

(a, c), (a, d), (b, c), (b, d)

그룹화 집합을 연결하면 다음과 같은 면에서 매우 유용합니다.

- 질의 개발 용이: 모든 그룹화를 직접 열거할 필요가 없습니다.
- 응용 프로그램에서 사용: OLAP 응용 프로그램을 통해 생성된 SQL에는 종종 그룹화 집합의 연결이 포함됩니다. 이 때 각 그룹화 집합은 차원에 필요한 그룹화를 정의합니다.

연결된 그룹화 예제

```
SELECT department_id, job_id, manager_id,
       SUM(salary)
  FROM employees
 GROUP BY department_id,
           ROLLUP (job_id),
           CUBE (manager_id);
```

	DEPARTMENT_ID	JOB_ID	MANAGER_ID	SUM(SALARY)
1	→	10 AD_ASST	101	4400
		20 MK_MAN	100	13000
2	→	10	101	4400
		20	100	13000
3	→	10 AD_ASST		4400
		10		4400
4	→	... SA_REP		7000
				7000

49 rows selected.

ORACLE

17-22

Copyright © Oracle Corporation, 2001. All rights reserved.

연결된 그룹화 예제

슬라이드의 예제는 다음 그룹화를 생성합니다.

- (department_id, manager_id, job_id)
- (department_id, manager_id)
- (department_id, job_id)
- (department_id)

해당 그룹에 대해 각각 총 급여가 계산됩니다.

슬라이드의 예제는 다음을 표시합니다.

- 각 부서, 업무 ID 및 관리자에 대한 총 급여
- 각 부서, 관리자 ID에 대한 총 급여
- 각 부서, 업무 ID에 대한 총 급여
- 각 부서에 대한 총 급여

쉽게 이해할 수 있도록 출력에서 부서 ID가 10인 부분이 강조 표시되어 있습니다.

요약

이 단원에서 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- ROLLUP 연산을 사용하여 하위 총계 값 산출
- CUBE 연산을 사용하여 교차 도표화 값 산출
- GROUPING 함수를 사용하여 ROLLUP 또는 CUBE를 통해 만들어진 행 값 식별
- GROUPING SETS 구문을 사용하여 같은 질의에서 여러 그룹화 정의
- GROUP BY 절을 사용하여 표현식을 다양한 방법으로 결합
 - 조합 열
 - 연결된 그룹화 집합

ORACLE

17-23

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

- ROLLUP 및 CUBE는 GROUP BY 절의 확장 기능입니다.
- ROLLUP은 하위 총계 값 및 최상위 총계 값을 표시하는데 사용됩니다.
- CUBE는 교차 도표화 값을 표시하는데 사용됩니다.
- GROUPING 함수는 행이 CUBE 또는 ROLLUP 연산자를 통해 산출된 집계 값인지 여부를 확인하는데 도움이 됩니다.
- GROUPING SETS 구문을 사용하면 같은 질의에서 여러 그룹화를 정의할 수 있습니다. GROUP BY는 지정된 그룹화를 모두 계산하여 UNION ALL로 결합합니다.
- GROUP BY 절에서 표현식을 다음과 같이 다양한 방법으로 결합할 수 있습니다.
 - 조합 열을 지정하려면 열을 괄호로 묶어 그룹화합니다. Oracle Server에서는 ROLLUP 또는 CUBE 연산을 계산하는 동안 조합 열을 한 단위로 처리합니다.
 - 연결된 그룹화 집합을 지정하려면 여러 그룹화 집합, ROLLUP 및 CUBE 연산을 쉼표로 구분함으로써 Oracle Server에서 이를 하나의 GROUP BY 절로 결합할 수 있도록 해야 합니다. 그러면 각 그룹화 집합으로부터 그룹화 상호 꼽이 만들어져 결과로 표시됩니다.

연습 17 개요

이 연습에서는 다음 내용을 다릅니다.

- ROLLUP 연산자 사용
- CUBE 연산자 사용
- GROUPING 함수 사용
- GROUPING SETS 사용

ORACLE

17-24

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 17 개요

이 연습에서는 GROUP BY 절의 확장 기능으로 ROLLUP 및 CUBE 연산자를 사용하며 GROUPING SETS도 사용하게 됩니다.

연습 17

1. 관리자 ID가 120보다 작은 관리자를 둔 사원에 대해 다음을 표시하는 질의를 작성하십시오.

- 관리자 ID
- 같은 관리자를 둔 사원에 대한 업무 ID별 총 급여와 해당 업무 ID
- 해당 관리자의 총 급여
- 업무 ID에 상관 없는 해당 관리자의 총 급여

MANAGER_ID	JOB_ID	SUM(SALARY)
100	AD_VP	34000
100	MK_MAN	13000
100	SA_MAN	10500
100	ST_MAN	5800
100		63300
101	AC_MGR	12000
101	AD_ASST	4400
101		16400
102	IT_PROG	9000
102		9000
103	IT_PROG	10200
103		10200
		98900

13 rows selected.

연습 17(계속)

2. 1번 질문의 출력을 살펴본 다음, GROUPING 함수를 사용하여 GROUP BY 표현식의 해당 열에 있는 NULL 값이 ROLLUP 연산으로 인한 것인지 확인하는 질의를 작성하십시오.

MGR	JOB	SUM(SALARY)	GROUPING(MANAGER_ID)	GROUPING(JOB_ID)
100	AD_VP	34000	0	0
100	MK_MAN	13000	0	0
100	SA_MAN	10500	0	0
100	ST_MAN	5800	0	0
100		63300	0	1
101	AC_MGR	12000	0	0
101	AD_ASST	4400	0	0
101		16400	0	1
102	IT_PROG	9000	0	0
102		9000	0	1
103	IT_PROG	10200	0	0
103		10200	0	1
		98900	1	1

13 rows selected.

연습 17(계속)

3. 관리자 ID가 120보다 작은 관리자를 둔 사원에 대해 다음을 표시하는 질의를 작성하십시오.

- 관리자 ID
- 같은 관리자를 둔 사원에 대한 업무별 총 급여와 해당 업무
- 해당 관리자의 총 급여
- 관리자에 상관 없이, 업무별 총 급여를 표시하는 교차 도표화 값
- 업무에 상관 없는 전체 총 급여

MANAGER_ID	JOB_ID	SUM(SALARY)
100	AD_VP	34000
100	MK_MAN	13000
100	SA_MAN	10500
100	ST_MAN	5800
100		63300
101	AC_MGR	12000
101	AD_ASST	4400
101		16400
102	IT_PROG	9000
102		9000
103	IT_PROG	10200
103		10200
	AC_MGR	12000
	AD_ASST	4400
MANAGER_ID	JOB_ID	SUM(SALARY)
	AD_VP	34000
	IT_PROG	19200
	MK_MAN	13000
	SA_MAN	10500
	ST_MAN	5800
		98900

20 rows selected.

연습 17(계속)

4. 3번 질문의 출력을 살펴본 다음, GROUPING 함수를 사용하여 GROUP BY 표현식의 해당 열에 있는 NULL 값이 CUBE 연산으로 인한 것인지 확인하는 질의를 작성하십시오.

MGR	JOB	SUM(SALARY)	GROUPING(MANAGER_ID)	GROUPING(JOB_ID)
100	AD_VP	34000	0	0
100	MK_MAN	13000	0	0
100	SA_MAN	10500	0	0
100	ST_MAN	5800	0	0
100		63300	0	1
101	AC_MGR	12000	0	0
101	AD_ASST	4400	0	0
101		16400	0	1
102	IT_PROG	9000	0	0
102		9000	0	1
103	IT_PROG	10200	0	0
103		10200	0	1
	AC_MGR	12000	1	0
	AD_ASST	4400	1	0
MGR	JOB	SUM(SALARY)	GROUPING(MANAGER_ID)	GROUPING(JOB_ID)
	AD_VP	34000	1	0
	IT_PROG	19200	1	0
	MK_MAN	13000	1	0
	SA_MAN	10500	1	0
	ST_MAN	5800	1	0
		98900	1	1

20 rows selected.

연습 17(계속)

5. GROUPING SETS를 사용하여 다음 그룹화를 표시하는 질의를 작성하십시오.

- department_id, manager_id, job_id
- department_id, job_id
- manager_id, job_id

질의에서 해당 그룹별 급여의 합을 계산해야 합니다.

DEPARTMENT_ID	MANAGER_ID	JOB_ID	SUM(SALARY)
10	101	AD_ASST	4400
20	100	MK_MAN	13000
20	201	MK_REP	6000
50	124	ST_CLERK	11700
50	100	ST_MAN	5800
60	102	IT_PROG	9000
60	103	IT_PROG	10200
80	100	SA_MAN	10500
80	149	SA_REP	19600
90		AD_PRES	24000
90	100	AD_VP	34000
110	205	AC_ACCOUNT	8300
110	101	AC_MGR	12000
	149	SA_REP	7000

	100	MK_MAN	13000
	100	SA_MAN	10500
	100	ST_MAN	5800
	101	AC_MGR	12000
	101	AD_ASST	4400
	102	IT_PROG	9000
	103	IT_PROG	10200
	124	ST_CLERK	11700
	149	SA_REP	26600
	201	MK_REP	6000
	205	AC_ACCOUNT	8300
		AD_PRES	24000

40 rows selected.

18
고급 서브 쿼리

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 여러 열 서브 쿼리 작성
- 널 값이 검색되는 경우 서브 쿼리의 기능 설명
- FROM 절에 서브 쿼리 작성
- SQL에서 스칼라 서브 쿼리 사용
- 상관 서브 쿼리로 해결할 수 있는 문제 유형 설명
- 상관 서브 쿼리 작성
- 상관 서브 쿼리를 사용한 행 갱신 및 삭제
- EXISTS 및 NOT EXISTS 연산자 사용
- WITH 절 사용

ORACLE

18-2

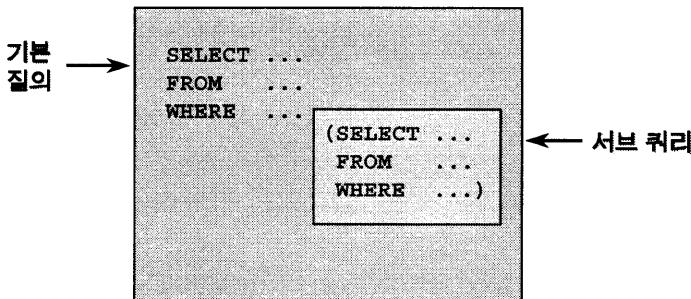
Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 여러 열 서브 쿼리 및 SELECT 문의 FROM 절에서 서브 쿼리를 작성하는 방법을 배웁니다. 또한 스칼라, 상관 서브 쿼리 및 WITH 절을 사용하여 문제를 해결하는 방법을 배웁니다.

서브 쿼리란?

서브 쿼리는 다른 SQL 문의 절에 내장된 SELECT 문입니다.



18-3

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

서브 쿼리란?

서브 쿼리는 상위 문이라고 하는 다른 SQL 문의 절에 내장된 SELECT 문입니다.

서브 쿼리(내부 질의)는 상위 문에 사용될 값을 반환합니다. 중첩된 서브 쿼리를 사용하는 것은 두 개의 순차적인 질의를 수행하여 내부 질의 결과를 외부 질의(메인 쿼리)의 검색 값으로 사용하는 것과 동일합니다.

서브 쿼리는 다음과 같은 경우에 사용할 수 있습니다.

- SELECT 문의 WHERE, HAVING 및 START WITH 절의 조건에 값 제공
- INSERT 또는 CREATE TABLE 문에서 대상 테이블에 삽입될 행 집합 정의
- CREATE VIEW 또는 CREATE SNAPSHOT 문에서 뷰 또는 스냅샷에 포함될 행 집합 정의
- UPDATE 문에서 기존 행에 할당될 하나 이상의 값 정의
- 포함하는 질의에서 작업할 대상 테이블 정의(이 경우 FROM 절에 서브 쿼리를 사용합니다. INSERT, UPDATE 및 DELETE 문에서도 이러한 작업을 수행할 수 있습니다.)

참고: 서브 쿼리는 전체 상위 문에 대해 한 번 평가됩니다.

서브 쿼리

```
SELECT select_list  
FROM table  
WHERE expr operator (SELECT select_list  
                      FROM table);
```

- 서브 쿼리(내부 질의)는 기본 질의 실행 전에 한 번 실행됩니다.
- 서브 쿼리의 결과는 기본 질의(메인 쿼리)에 사용됩니다.

ORACLE

18-4

Copyright © Oracle Corporation, 2001. All rights reserved.

서브 쿼리

서브 쿼리를 사용하면 단순한 명령문을 토대로 강력한 명령문을 구축할 수 있습니다. 서브 쿼리는 해당 테이블 자체 또는 다른 테이블에 있는 데이터에 따라 조건을 달리하여 테이블에서 행을 선택할 필요가 있는 경우 매우 유용합니다. 서브 쿼리는 조건값을 알 수 없는 SQL 문을 작성하는 데 매우 유용합니다.

구문 설명:

operator >, =, IN 등 비교 연산자를 포함합니다.

참고: 비교 연산자는 단일 행 연산자(>, =, >=, <, <>, <=)와 다중 행 연산자(IN, ANY, ALL)로 분류됩니다.

서브 쿼리는 중첩 SELECT 문, 하위 SELECT 문, 내부 SELECT 문이라고도 합니다. 내부 질의 및 외부 질의는 같은 테이블 또는 다른 테이블에서 데이터를 검색할 수 있습니다.

서브 쿼리 사용

```
SELECT last_name
  FROM employees
 WHERE salary > 10500
      (SELECT salary
       FROM employees
      WHERE employee_id = 149) ;
```

LAST_NAME
King
Kochhar
De Haan
Abel
Hartstein
Higgins

6 rows selected.

ORACLE

18-5

Copyright © Oracle Corporation, 2001. All rights reserved.

서브 쿼리 사용

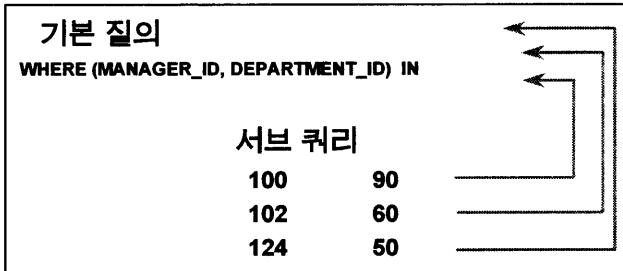
슬라이드의 예제에서 내부 질의는 사원 번호가 149인 사원의 급여를 반환합니다. 외부 질의는 내부 질의의 결과를 사용하여 이 사원보다 급여가 많은 사원의 이름을 모두 표시합니다.

예제

회사의 평균 급여보다 적은 급여를 받는 사원의 이름을 모두 표시합니다.

```
SELECT last_name, job_id, salary
  FROM employees
 WHERE salary < (SELECT AVG(salary)
                  FROM employees);
```

다중 열 서브 쿼리



기본 질의의 각 행은 여러 행 및 여러 열 서브 쿼리의
값과 비교됩니다.

ORACLE

다중 열 서브 쿼리

지금까지는 내부 SELECT 문에서 한 열만 반환하는 단일 행 서브 쿼리와 다중 행 서브 쿼리를 작성했으며 이는 상위 SELECT 문의 표현식을 평가하는 데 사용되었습니다. 두 개 이상의 열을 비교하려면 논리 연산자를 사용하여 혼합 WHERE 절을 작성해야 합니다. 다중 열 서브 쿼리를 사용하면 중복된 WHERE 조건을 하나의 WHERE 절로 결합할 수 있습니다.

구문

```
SELECT column, column, ...
FROM   table
WHERE  (column, column, ...) IN
       (SELECT column, column, ...
        FROM   table
        WHERE  condition);
```

슬라이드의 그림은 기본 질의의 MANAGER_ID 및 DEPARTMENT_ID 값과 서브 쿼리에서 검색된 MANAGER_ID 및 DEPARTMENT_ID 값의 비교를 보여줍니다. 비교될 열 수가 둘 이상 이므로 이 예제는 여러 열 서브 쿼리로 볼 수 있습니다.

열 비교

여러 열 서브 쿼리의 열 비교 방식은 다음과 같을 수 있습니다.

- 쌍(pairwise) 비교
- 비쌍(nonpairwise) 비교

ORACLE

18-7

Copyright © Oracle Corporation, 2001. All rights reserved.

쌍 비교와 비쌍 비교

다중 열 서브 쿼리에서의 열 비교는 쌍 비교 또는 비쌍 비교입니다.

다음 슬라이드의 예제에서는 WHERE 절에서 쌍 비교가 실행됩니다. SELECT 문의 후보 행은 EMPLOYEE_ID가 178 또는 174인 사원과 MANAGER_ID 및 DEPARTMENT_ID 열이 모두 같아야 합니다.

다중 열 서브 쿼리는 비쌍 비교도 될 수 있습니다. 비쌍 비교에서 상위 SELECT 문의 WHERE 절에 있는 각 열은 내부 SELECT 문에서 검색된 여러 값에 각각 비교됩니다. 각 열은 내부 SELECT 문에서 검색된 값 중 어느 것과도 일치할 수 있습니다. 그러나 전체적으로 볼 때 기본 SELECT 문의 여러 조건을 모두 만족시키는 행만이 표시됩니다. 다음 페이지의 예제는 비쌍 비교를 보여줍니다.

쌍(pairwise) 비교 서브 쿼리

EMPLOYEE_ID가 178 또는 174인 사원의 관리자 및 부서와 같은 관리자 및 부서를 갖는 사원의 정보를 표시합니다.

```
SELECT employee_id, manager_id, department_id
  FROM employees
 WHERE (manager_id, department_id) IN
       (SELECT manager_id, department_id
        FROM employees
        WHERE employee_id IN (178,174))
AND   employee_id NOT IN (178,174);
```

ORACLE

18-8

Copyright © Oracle Corporation, 2001. All rights reserved.

쌍 비교 서브 쿼리

슬라이드에 있는 예제는 서브 쿼리가 두 개 이상의 열을 반환하므로 다중 열 서브 쿼리입니다. 이 서브 쿼리는 EMPLOYEES 테이블의 각 행에 있는 MANAGER_ID 열 및 DEPARTMENT_ID 열의 값을 EMPLOYEE_ID가 178 또는 174인 사원의 MANAGER_ID 열 및 DEPARTMENT_ID 열 값과 비교합니다. 먼저 EMPLOYEE_ID가 178 또는 174인 사원의 MANAGER_ID 및 DEPARTMENT_ID 값을 검색하는 서브 쿼리가 실행됩니다. 그 다음 이 값이 EMPLOYEES 테이블의 각 행에 있는 MANAGER_ID 열 및 DEPARTMENT_ID 열과 비교됩니다. 값이 일치하면 행이 표시됩니다. 출력에서 EMPLOYEE_ID가 178 또는 174인 사원의 레코드는 표시되지 않습니다. 슬라이드에 있는 질의의 출력 결과는 다음과 같습니다.

EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
176	149	80

비쌍 비교 서브 쿼리

EMPLOYEE_ID가 174 또는 141인 사원과 관리자가 같으며
EMPLOYEE_ID가 174 또는 141인 사원과 부서가 같은 사원
정보를 표시합니다.

```
SELECT employee_id, manager_id, department_id
  FROM employees
 WHERE manager_id IN
       (SELECT manager_id
        FROM employees
       WHERE employee_id IN (174,141))
 AND department_id IN
       (SELECT department_id
        FROM employees
       WHERE employee_id IN (174,141))
 AND employee_id NOT IN(174,141);
```

ORACLE

18-9

Copyright © Oracle Corporation, 2001. All rights reserved.

비쌍 비교 서브 쿼리

예제에서는 열의 비쌍 비교를 보여줍니다. 여기서는 MANAGER_ID는 사원 ID가 174 또는 141인 사원의 관리자 ID 중 하나와 일치하고, DEPARTMENT_ID는 사원 ID가 174 또는 141인 사원의 부서 ID 중 하나와 일치하는 사원의 EMPLOYEE_ID, MANAGER_ID 및 DEPARTMENT_ID를 표시합니다.

먼저 EMPLOYEE_ID가 174 또는 141인 사원에 대한 MANAGER_ID 값을 검색하는 서브 쿼리가 실행됩니다. 비슷한 방식으로 EMPLOYEE_ID가 174 또는 141인 사원에 대한 DEPARTMENT_ID 값을 검색하는 두번쩨 서브 쿼리가 실행됩니다. 그런 다음 검색된 MANAGER_ID 및 DEPARTMENT_ID 열의 값이 EMPLOYEES 테이블의 각 행에 대한 MANAGER_ID 및 DEPARTMENT_ID 열에 비교됩니다. EMPLOYEES 테이블의 행에 있는 MANAGER_ID 열이 내부 서브 쿼리에서 검색된 MANAGER_ID 값 중 하나와 일치하고 EMPLOYEES 테이블의 행에 있는 DEPARTMENT_ID 열이 두번쩨 서브 쿼리에서 검색된 DEPARTMENT_ID 값 중 하나와 일치하면 해당 레코드가 표시됩니다. 슬라이드에 있는 질의의 출력 결과는 다음과 같습니다.

EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
142	124	50
143	124	50
144	124	50
176	149	80

FROM 절에 서브 쿼리 사용

```
SELECT a.last_name, a.salary,
       a.department_id, b.salavg
  FROM employees a,
       (SELECT department_id,
              AVG(salary) salavg
             FROM employees
            GROUP BY department_id) b
 WHERE a.department_id = b.department_id
   AND a.salary > b.salavg;
```

LAST_NAME	SALARY	DEPARTMENT_ID	SAL_AVG
Hartstein	13000	20	9500
Mourgos	5800	50	3600
Hunold	9000	60	6400
Zlotkey	10500	80	10033.3333
Abel	11000	80	10033.3333
King	24000	90	19333.3333
Higgins	12000	110	10150

7 rows selected.

ORACLE

FROM 절에 서브 쿼리 사용

SELECT 문의 FROM 절에 서브 쿼리를 사용할 수 있습니다. 이는 뷰를 사용하는 방식과 매우 비슷합니다. SELECT 문의 FROM 절에 있는 서브 쿼리는 인라인 뷰라고도 합니다. SELECT 문의 FROM 절에 있는 서브 쿼리는 해당 SELECT 문에 대해서만 데이터 소스를 정의합니다. 슬라이드 예제는 사원의 이름, 급여, 부서 번호 및 소속 부서의 평균 급여보다 많은 급여를 받는 모든 사원의 평균 급여를 표시합니다. FROM 절에 있는 서브 쿼리를 b라고 명명하고 외부 질의가 이 별칭을 사용하여 SALAVG 열을 참조합니다.

스칼라 서브 쿼리식

- 스칼라 서브 쿼리식은 하나의 행에서 하나의 열 값만 반환하는 서브 쿼리입니다.
- Oracle8i에서는 다음과 같은 제한된 경우에만 스칼라 서브 쿼리가 지원되었습니다.
 - SELECT 문(FROM 및 WHERE 절)
 - INSERT 문의 VALUES 목록
- Oracle9i에서는 스칼라 서브 쿼리를 다음 경우에 사용할 수 있습니다.
 - DECODE 및 CASE의 조건 및 표현식 부분
 - GROUP BY를 제외한 SELECT의 모든 절

ORACLE

18-11

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL의 스칼라 서브 쿼리

하나의 행에서 하나의 열 값만 반환하는 서브 쿼리를 스칼라 서브 쿼리라고 합니다. 혼합 WHERE 절과 논리 연산자를 사용하여 두 개 이상의 열을 비교하기 위해 작성된 여러 열 서브 쿼리는 스칼라 서브 쿼리라고 볼 수 없습니다.

스칼라 서브 쿼리식의 값은 서브 쿼리의 SELECT 목록에 있는 항목의 값입니다. 서브 쿼리가 0개의 행을 반환하면 스칼라 서브 쿼리식의 값은 NULL입니다. 서브 쿼리가 두 개 이상의 행을 반환하면 오류가 반환됩니다. Oracle Server에서는 항상 SELECT 문에 스칼라 서브 쿼리를 사용할 수 있습니다. Oracle9i에서는 스칼라 서브 쿼리의 사용이 더욱 향상되어 다음과 같은 경우에 스칼라 서브 쿼리를 사용할 수 있습니다.

- DECODE 및 CASE의 조건 및 표현식 부분
- GROUP BY를 제외한 SELECT의 모든 절
- UPDATE 문의 SET 절 및 WHERE 절에서 연산자의 왼쪽

그러나 다음과 같은 경우에는 스칼라 서브 쿼리를 사용할 수 없습니다.

- 클러스터의 열 및 해시 표현식에 대한 기본값
- DML 문의 RETURNING 절
- 함수 기반 인덱스의 기준
- GROUP BY 절, CHECK 제약 조건, WHEN 조건
- HAVING 절
- START WITH 및 CONNECT BY 절
- CREATE PROFILE과 같이 질의와 연관되지 않은 명령문

스칼라 서브 쿼리: 예제

CASE 표현식의 스칼라 서브 쿼리

```
SELECT employee_id, last_name,
       (CASE
        WHEN department_id = 20
        THEN 'Canada' ELSE 'USA' END) location
  FROM employees;
```

← 20
(SELECT department_id FROM departments
WHERE location_id = 1800)

ORDER BY 절의 스칼라 서브 쿼리

```
SELECT employee_id, last_name
  FROM employees e
 ORDER BY (SELECT department_name
            FROM departments d
           WHERE e.department_id = d.department_id);
```

ORACLE

18-12

Copyright © Oracle Corporation, 2001. All rights reserved.

스칼라 서브 쿼리: 예제

슬라이드에 있는 첫번째 예제는 CASE 표현식에 사용된 스칼라 서브 쿼리를 보여줍니다. 내부 질의는 위치 ID가 1800인 부서의 부서 ID인 20을 반환합니다. 외부 질의의 CASE 표현식에서 는 내부 질의 결과를 사용하여 외부 질의에서 검색된 레코드의 부서 ID가 20이면 Canada를 그렇지 않으면 USA를 표시하되, 사원 ID 및 이름과 함께 표시합니다.

앞 예제의 결과는 다음과 같습니다.

EMPLOYEE_ID	LAST_NAME	LOCATION
100	King	USA
101	Kochhar	USA
102	De Haan	USA
...		
201	Hartstein	Canada
202	Fay	Canada
205	Higgins	USA
206	Gietz	USA

20 rows selected.

스칼라 서브 쿼리: 예제(계속)

슬라이드의 두번째 예제는 ORDER BY 절에 사용된 스칼라 서브 쿼리를 보여줍니다. 이 예제에서는 EMPLOYEES 테이블의 DEPARTMENT_ID를 DEPARTMENTS 테이블의 DEPARTMENT_ID와 대응시켜 DEPARTMENT_NAME을 알아낸 뒤 이 열을 기준으로 결과를 정렬합니다. 이 비교 작업은 ORDER BY 절의 스칼라 서브 쿼리에서 수행됩니다. 두번째 예제의 결과는 다음과 같습니다.

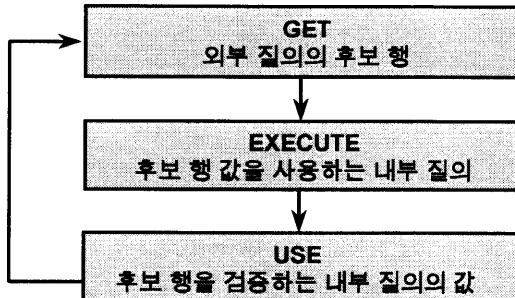
EMPLOYEE_ID	LAST_NAME
205	Higgins
206	Gietz
200	Whalen
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
201	Hartstein
202	Fay
149	Zlotkey
176	Taylor
174	Abel
EMPLOYEE_ID	LAST_NAME
124	Mourgos
141	Rajs
142	Davies
143	Matos
144	Vargas
178	Grant

20 rows selected.

두번째 예제에서는 상호관련 서브 쿼리를 사용합니다. 상호관련 서브 쿼리에서 서브 쿼리는 상위 문에서 참조된 테이블의 열을 참조합니다. 상호관련 서브 쿼리는 나중에 설명합니다.

상호관련 서브 쿼리 (correlated subquery)

상관 서브 쿼리는 행 단위 처리에 사용됩니다. 각 서브 쿼리는 외부 질의의 모든 행에 대해 한 번씩 실행됩니다.



ORACLE

상호관련 서브 쿼리(correlated subquery)

Oracle Server는 서브 쿼리가 상위 문에서 참조되는 테이블의 열을 참조할 때 상호관련 서브 쿼리를 수행합니다. 상호관련 서브 쿼리는 상위 문에서 처리되는 각 행에 대해 한 번씩 평가됩니다. 이러한 상위 문으로서는 SELECT, UPDATE 또는 DELETE 문이 가능합니다.

중첩 서브 쿼리와 상호관련 서브 쿼리의 비교

일반 중첩 서브 쿼리를 사용하면 내부 SELECT 질의가 먼저 한 번 실행됨으로써 기본 질의에 사용될 값을 반환합니다. 그러나 상호관련 서브 쿼리는 외부 질의에서 고려되는 각 후보 행에 대해 한 번씩 실행됩니다. 즉, 내부 질의가 외부 질의에 의해 실행됩니다.

중첩 서브 쿼리 실행

- 내부 질의가 먼저 실행되어 값을 찾습니다.
- 외부 질의는 내부 질의의 값을 사용하여 한 번만 실행됩니다.

상호관련 서브 쿼리 실행

- 후보 행을 가져옵니다(외부 질의에서 인출(fetch)).
- 후보 행의 값을 사용하여 내부 질의를 실행합니다.
- 내부 질의의 결과 값을 사용하여 후보 행을 검증합니다.
- 후보 행이 남지 않을 때까지 반복합니다.

상호관련 서브 쿼리 (correlated subquery)

```
SELECT column1, column2, ...
  FROM table1 [outer]
 WHERE column1 operator
       (SELECT column1, column2
        FROM table2
        WHERE expr1 =
              [outer].expr2);
```

서브 쿼리는 상위 질의에 있는 테이블의 열을 참조합니다.

ORACLE

18-15

Copyright © Oracle Corporation, 2001. All rights reserved.

상호관련 서브 쿼리(계속)

상호관련 서브 쿼리는 테이블에서 모든 행을 읽어서 각 행의 값을 관련된 데이터와 비교하는 방법 중 하나입니다. 이 방법은 기본 질의에서 고려된 각 후보 행에 대해 서브 쿼리가 다른 결과 또는 결과 집합을 반환해야 하는 경우에 사용됩니다. 즉, 상위 문에서 처리되는 각 행의 값에 따라 응답이 달라지는 다중 질의에 응답할 때 상관 서브 쿼리를 사용합니다.

Oracle Server는 서브 쿼리가 상위 질의에 있는 테이블의 열을 참조할 때 상호관련 서브 쿼리를 수행합니다.

참고: 상호관련 서브 쿼리에는 ANY 및 ALL 연산자를 사용할 수 있습니다.

상호관련 서브 쿼리 사용

소속 부서의 평균 급여보다 많은 급여를 받는 사원을 모두 찾습니다.

```
SELECT last_name, salary, department_id
FROM   employees outer
WHERE  salary > (
    SELECT AVG(salary)
    FROM   employees
    WHERE  department_id =
           outer.department_id) ;
```

외부 질의의
행이
처리될 때마다
내부 질의가
평가됩니다.

ORACLE

18-16

Copyright © Oracle Corporation, 2001. All rights reserved.

상호관련 서브 쿼리 사용

슬라이드의 예제에서는 소속 부서의 평균 급여보다 많은 급여를 받는 사원을 확인합니다.
이 경우 상호관련 서브 쿼리가 각 부서의 평균 급여를 계산합니다.

외부 질의와 내부 질의의 FROM 절에서 모두 EMPLOYEES 테이블을 사용하므로 확실히 하기 위해
외부 SELECT 문의 EMPLOYEES에 별칭을 사용합니다. 별칭을 사용하면 전체 SELECT 문을 읽기
쉬워지며 별칭을 사용하지 않는 경우 내부 문에서 외부 테이블 열과 내부 테이블 열을 구별할 수
없으므로 질의가 제대로 작동하지 않을 수 있습니다.

상호관련 서브 쿼리 사용

업무를 두 번 이상 바꾼 사원에 대한 정보를 표시합니다.

```
SELECT e.employee_id, last_name, e.job_id
  FROM employees e
 WHERE 2 <= (SELECT COUNT(*)
      FROM job_history
     WHERE employee_id = e.employee_id);
```

EMPLOYEE_ID	LAST_NAME	JOB_ID
101	Kochhar	AD_VP
176	Taylor	SA_REP
200	Whalen	AD_ASST

ORACLE

18-17

Copyright © Oracle Corporation, 2001. All rights reserved.

상호관련 서브 쿼리 사용

슬라이드의 예제에서는 업무를 두 번 이상 바꾼 사원에 대한 정보를 표시합니다. Oracle Server에서는 다음과 같이 상관 서브 쿼리를 평가합니다.

- 외부 질의에 지정된 테이블에서 행을 선택합니다. 이 행이 현재 후보 행입니다.
 - 이 후보 행의 서브 쿼리에서 참조하는 행의 값을 저장합니다. 슬라이드 예제의 경우 서브 쿼리에서 참조하는 행은 E.EMPLOYEE_ID입니다.
 - 외부 질의의 후보 행 값을 참조하는 조건을 사용하여 서브 쿼리를 실행합니다. 슬라이드 예제의 경우 그룹 함수 COUNT(*)가 2단계에서 얻은 E.EMPLOYEE_ID 열 값에 대해 계산됩니다.
 - 3단계에서 수행된 서브 쿼리의 결과를 바탕으로 외부 질의의 WHERE 절을 평가합니다. 이 단계가 후보 행의 출력 여부를 결정합니다. 예제의 경우 서브 쿼리에서 평가된 사원의 업무 변경 횟수를 외부 질의의 WHERE 절에서 2와 비교합니다. 조건을 만족하면 해당 사원 레코드가 표시됩니다.
 - 테이블에 있는 행이 모두 처리될 때까지 다음 후보 행에 대해 같은 과정을 반복합니다.
- 상관 관계는 서브 쿼리에 있는 외부 질의 요소를 사용하여 성립됩니다. 이 예제에서는 서브 쿼리의 테이블에 있는 EMPLOYEE_ID를 외부 질의의 테이블에 있는 EMPLOYEE_ID와 비교하는 EMPLOYEE_ID = E.EMPLOYEE_ID 문에서 상관 관계가 성립됩니다.

EXISTS 연산자 사용

- EXISTS 연산자는 서브 쿼리의 결과 집합에 행이 있는지 여부를 검사합니다.
- 서브 쿼리 행 값이 발견되는 경우:
 - 내부 질의에서 더 이상 검색하지 않습니다.
 - 조건 플래그가 TRUE가 됩니다.
- 서브 쿼리 행 값이 발견되지 않는 경우:
 - 조건 플래그가 FALSE가 됩니다.
 - 내부 질의에서 검색을 계속합니다.

ORACLE

18-18

Copyright © Oracle Corporation, 2001. All rights reserved.

EXISTS 연산자

중첩 SELECT 문에는 논리 연산자를 모두 사용할 수 있으며 EXISTS 연산자도 사용할 수 있습니다. 이 연산자는 외부 질의에서 검색된 값이 내부 질의에서 검색된 값의 결과 집합에 존재하는지 여부를 검사하기 위한 상호관련 서브 쿼리에 자주 사용됩니다. 이 연산자는 서브 쿼리가 한 행 이상 반환하면 TRUE를 반환합니다. 해당 값이 없으면 FALSE를 반환합니다. 마찬가지로 NOT EXISTS는 외부 질의에서 검색된 값이 내부 질의에서 검색된 값의 결과 집합에 속하지 않는지 여부를 검사합니다.

EXISTS 연산자 사용

부하 사원을 가진 사원을 찾습니다.

```
SELECT employee_id, last_name, job_id, department_id
  FROM employees outer
 WHERE EXISTS ( SELECT 'x' FROM employees
      WHERE manager_id =
        outer.employee_id);
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90
103	Hundl	IT_PROG	60
124	Mourgos	ST_MAN	50
149	Zlotkey	SA_MAN	80
201	Hartstein	MK_MAN	20
205	Higgins	AC_MGR	110

8 rows selected.

ORACLE

18-13

Copyright © Oracle Corporation, 2001. All rights reserved.

EXISTS 연산자 사용

EXISTS 연산자를 사용하면 다음 조건을 만족하는 행이 하나라도 발견될 경우 내부 질의를 더 이상 검색하지 않습니다.

```
WHERE manager_id = outer.employee_id.
```

내부 SELECT 질의가 특정 값을 반환할 필요가 없으므로 상수를 선택할 수 있다는 점에 주목하십시오. 성능면에서 볼 때 열을 선택하는 것보다 상수를 선택하는 게 더 빠릅니다.

참고: 내부 질의의 SELECT 질에 EMPLOYEE_ID가 있으므로 테이블에서 해당 열을 스캔하게 됩니다. 열을 X 또는 다른 상수로 바꾸면 성능이 향상됩니다. 이는 IN 연산자를 사용하는 것 보다 훨씬 효율적입니다.

IN 구문을 EXISTS 연산자 대신 사용할 수 있으며, 이에 대한 예제는 다음과 같습니다.

```
SELECT employee_id, last_name, job_id, department_id
  FROM employees
 WHERE employee_id IN (SELECT manager_id
      FROM employees
     WHERE manager_id IS NOT NULL);
```

NOT EXISTS 연산자 사용

사원이 없는 부서를 모두 찾습니다.

```
SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS (SELECT 'X'
                   FROM employees
                   WHERE department_id
                     = d.department_id);
```

DEPARTMENT_ID	DEPARTMENT_NAME
190	Contracting

ORACLE

18-20

Copyright © Oracle Corporation, 2001. All rights reserved.

NOT EXISTS 연산자 사용

다른 해결책

NOT IN 구문을 NOT EXISTS 연산자 대신 사용할 수 있으며, 이에 대한 예제는 다음과 같습니다.

```
SELECT department_id, department_name
FROM departments
WHERE department_id NOT IN (SELECT department_id
                             FROM employees);
no rows selected
```

그러나 NOT IN은 결과 집합에 NULL 값이 있는 경우 FALSE를 반환합니다. 따라서 부서 테이블에 WHERE 조건을 만족하는 행이 있을지라도 질의에서 행을 반환하지 않을 수 있습니다.

상호관련 UPDATE

```
UPDATE table1 alias1
SET    column = (SELECT expression
                  FROM   table2 alias2
                  WHERE  alias1.column =
                         alias2.column);
```

상관 서브 쿼리를 사용하면 다른 테이블의 행에 따라 테이블의 행을 갱신할 수 있습니다.

ORACLE

18-21

Copyright © Oracle Corporation, 2001. All rights reserved.

상호관련 UPDATE

UPDATE 문에서 상호관련 서브 쿼리를 사용하여 다른 테이블의 행에 따라 테이블의 행을 갱신할 수 있습니다.

상호관련 UPDATE

- 부서 이름을 저장하기 위한 열을 추가하여 EMPLOYEES 테이블을 비정규화합니다.
- 상호관련 update를 사용하여 테이블을 채웁니다.

```
ALTER TABLE employees  
ADD(department_name VARCHAR2(14));
```

```
UPDATE employees e  
SET department_name =  
      (SELECT department_name  
       FROM departments d  
      WHERE e.department_id = d.department_id);
```

ORACLE

18-22

Copyright © Oracle Corporation, 2001. All rights reserved.

상호관련 UPDATE(계속)

슬라이드의 예제는 부서 이름을 저장하기 위한 열을 추가하여 EMPLOYEES 테이블을 비정규화한 다음 상관 간선을 사용하여 테이블을 채웁니다.

다음은 상호관련 update의 또 다른 예입니다.

문제 문

상호관련 서브 쿼리를 사용하여 REWARDS 테이블의 행에 따라 다음과 같이 EMPLOYEES 테이블의 행을 갱신합니다.

```
UPDATE employees  
SET salary = (SELECT employees.salary + rewards.pay_raise  
              FROM rewards  
             WHERE employee_id = employees.employee_id  
           AND payraise_date =  
                  (SELECT MAX(payraise_date)  
                   FROM rewards  
                  WHERE employee_id = employees.employee_id))  
WHERE employees.employee_id  
IN (SELECT employee_id  
     FROM rewards);
```

상호관련 UPDATE(계속)

이 예제에서는 REWARDS 테이블을 사용합니다. REWARDS 테이블에는 EMPLOYEE_ID, PAY_RAISE 및 PAYRAISE_DATE 열이 있습니다. 사원의 급여가 인상되면 REWARDS 테이블에 사원 ID 정보, 인상 급여 폭, 급여 인상일이 삽입됩니다. REWARDS 테이블에는 한 사원에 대한 레코드가 여러 개 있을 수 있습니다. PAYRAISE_DATE 열을 사용하여 해당 사원에 대한 최근의 인상 급여를 식별합니다.

예제에서는 EMPLOYEES 테이블의 SALARY 열을 갱신하여 최근에 인상된 급여를 반영합니다. 이 작업은 현재 급여에 REWARDS 테이블의 해당 인상 급여 폭을 추가하여 수행됩니다.

상호관련 DELETE

```
DELETE FROM table1 alias1
WHERE column operator
      (SELECT expression
       FROM   table2 alias2
       WHERE  alias1.column = alias2.column);
```

상호관련 서브 쿼리를 사용하면 다른 테이블의 행에 따라
테이블의 행을 삭제할 수 있습니다.

ORACLE

18-24

Copyright © Oracle Corporation, 2001. All rights reserved.

상호관련 DELETE

DELETE 문에서 상관 서브 쿼리를 사용하여 다른 테이블에도 존재하는 행을 삭제할 수 있습니다.
JOB_HISTORY 테이블에 최근 네 개의 업무 기록만 유지하려면 사원이 다섯번쩨로 업무를 바꿨을
때 JOB_HISTORY 테이블에서 해당 사원에 대해 MIN(START_DATE)를 찾아 가장 오래된
JOB_HISTORY 열을 삭제합니다. 다음 코드는 상관 DELETE를 사용하여 앞의 작업을 수행하는
방법을 보여줍니다.

```
DELETE FROM job_history JH
WHERE employee_id =
      (SELECT employee_id
       FROM   employees E
       WHERE  JH.employee_id = E.employee_id
       AND    start_date =
              (SELECT MIN(start_date)
               FROM   job_history JH
               WHERE  JH.employee_id = E.employee_id)
       AND 5 >  (SELECT COUNT(*)
                  FROM   job_history JH
                  WHERE  JH.employee_id = E.employee_id
                  GROUP  BY employee_id
                  HAVING COUNT(*) >= 4));
```

상호관련 DELETE

상호관련 서브 쿼리를 사용하여 EMPLOYEES 테이블의 행 중 EMP_HISTORY 테이블에도 존재하는 행을 삭제합니다.

```
DELETE FROM employees E
WHERE employee_id =
      (SELECT employee_id
       FROM   emp_history
       WHERE  employee_id = E.employee_id);
```

ORACLE

18-25

Copyright © Oracle Corporation, 2001. All rights reserved.

상호관련 DELETE(계속)

예제

이 예제에는 다음 두 테이블이 사용됩니다.

- 현재 모든 사원의 세부 정보를 제공하는 EMPLOYEES 테이블
- 퇴사한 사원의 세부 정보를 제공하는 EMP_HISTORY 테이블

EMP_HISTORY에는 퇴사한 사원에 대한 데이터가 들어 있으므로 EMPLOYEES 및 EMP_HISTORY 테이블 모두에 같은 사원의 레코드가 존재하면 이는 잘못된 경우입니다. 슬라이드에 있는 상관 서브 쿼리를 사용하면 이러한 잘못된 레코드를 삭제할 수 있습니다.

WITH 절

SELECT 문에서만 사용 가능

- WITH 절을 사용하면 복합 질의에서 여러 번 발생하는 같은 질의 블록을 SELECT 문에서 사용할 수 있습니다.
- WITH 절은 질의 블록의 결과를 검색한 다음 이를 사용자 임시 테이블스페이스에 저장합니다.
- WITH 절을 사용하면 성능이 향상됩니다.

ORACLE

18-26

Copyright © Oracle Corporation, 2001. All rights reserved.

WITH 절

WITH 절을 사용하면 질의 블록을 정의하여 질의에서 사용할 수 있습니다. WITH 절(이전의 `subquery_factoring_clause`)을 사용하면 복합 질의에서 여러 번 발생하는 같은 질의 블록을 SELECT 문에서 다시 사용할 수 있습니다. 이는 질의가 같은 질의 블록을 여러 번 참조하거나 조인 및 집계를 해야 하는 경우 매우 유용합니다.

해당 질의 블록을 평가하는 데 많은 비용이 들고 복합 질의에서 해당 질의 블록이 여러 번 발생하는 경우 WITH 절을 사용하면 같은 질의를 다시 사용할 수 있습니다. WITH 절을 사용하면 Oracle Server에서는 질의 블록의 결과를 검색하여 사용자 임시 테이블스페이스에 저장합니다. 이렇게 하면 성능이 향상될 수 있습니다.

WITH 절의 이점

- 질의를 읽기 쉽게 만듭니다.
- 해당 절이 질의에서 여러 번 사용될지라도 한 번만 평가하므로 성능이 향상됩니다.

WITH 절: 예제

WITH 절을 사용하여 전체 부서의 평균 총 급여보다 총 급여가 많은 부서의 부서 이름 및 총 급여를 표시하는 질의를 작성합니다.

ORACLE

18-27

Copyright © Oracle Corporation, 2001. All rights reserved.

WITH 절: 예제

슬라이드의 문제에서는 다음 중간 계산이 필요합니다.

1. WITH 절을 사용하여 모든 부서의 총 급여를 계산한 다음 결과를 저장합니다.
2. WITH 절을 사용하여 전체 부서의 평균 총 급여를 계산한 다음 결과를 저장합니다.
3. 1단계에서 계산한 총 급여를 2단계에서 계산한 평균 총 급여와 비교합니다. 특정 부서의 총 급여가 전체 부서의 평균 총 급여보다 많으면 해당 부서 이름 및 총 급여를 표시합니다.

앞의 문제에 대한 답은 다음 페이지에 나와 있습니다.

WITH 절: 예제

```
WITH
dept_costs AS (
  SELECT d.department_name, SUM(e.salary) AS dept_total
  FROM employees e, departments d
  WHERE e.department_id = d.department_id
  GROUP BY d.department_name),
avg_cost AS (
  SELECT SUM(dept_total)/COUNT(*) AS dept_avg
  FROM dept_costs)
SELECT *
FROM dept_costs
WHERE dept_total >
  (SELECT dept_avg
  FROM avg_cost)
ORDER BY department_name;
```

ORACLE

18-28

Copyright © Oracle Corporation, 2001. All rights reserved.

WITH 절: 예제(계속)

슬라이드에 있는 SQL 코드는 WITH 절을 사용하여 성능을 향상시키고 SQL을 보다 간단하게 작성할 수 있는 상황을 보여주는 예제입니다. 이 질의에서는 DEPT_COSTS 및 AVG_COST라는 이름의 질의를 만든 다음 이를 기본 질의의 본문에 사용합니다. 내부적으로 WITH 절은 인라인 뷰 또는 임시 테이블로 해석됩니다. 최적기는 WITH 절의 결과를 임시로 저장하는 테드는 비용이나 이점은 따져서 적절한 해석 방법을 선택합니다.

참고: SELECT 문의 FROM 절에 있는 서브 쿼리를 인라인 뷰라고도 합니다.

슬라이드의 SQL 코드에서 생성되는 출력 결과는 다음과 같습니다.

DEPARTMENT_NAME	DEPT_TOTAL
Executive	58000
Sales	30100

WITH 절 사용의 참고 사항

- SELECT 문에서만 사용됩니다.
- 질의 이름은 뒤에 정의된 모든 WITH 요소 질의 블록(해당 서브 쿼리 블록 포함) 및 기본 질의 블록(해당 서브 쿼리 블록 포함) 자체에서 사용할 수 있습니다.
- 질의 이름이 기존 테이블 이름과 같을 경우 구문 분석기는 안쪽에서 바깥쪽으로 검색하므로 질의 블록 이름이 테이블 이름보다 높은 우선순위를 가집니다.
- WITH 절에는 여러 개의 질의를 쉼표로 구분하여 사용할 수 있습니다.

요약

이 단원에서 다음과 같은 사항을 배웠습니다.

- 다중 열 서브 쿼리는 여러 열을 반환
- 다중 열 비교는 쌍 비교 또는 비쌍 비교가 가능
- 다중 열 서브 쿼리를 SELECT 문의 FROM 절에서도 사용 가능
- Oracle9i에서의 향상된 스칼라 서브 쿼리 기능

ORACLE

18-29

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

다중 열 서브 쿼리를 사용하면 여러 WHERE 조건을 하나의 WHERE 절로 결합할 수 있습니다. 다중 열 서브 쿼리에서의 열 비교는 쌍 비교 또는 비쌍 비교가 될 수 있습니다.

서브 쿼리를 사용하면 포함하는 질의에서 작업할 대상 테이블을 정의할 수 있습니다.

Oracle 9i에서는 스칼라 서브 쿼리의 기능이 향상되었습니다. 이제 스칼라 서브 쿼리를 다음과 같은 경우에 사용할 수 있습니다.

- DECODE 및 CASE의 조건 및 표현식 부분
- GROUP BY를 제외한 SELECT의 모든 절
- UPDATE 문의 SET 절 및 WHERE 절

요약

- 상호관련 서브 쿼리는 서브 쿼리가 각 후보 행에 대해 다른 결과를 반환해야 하는 경우 유용
- EXISTS 연산자는 값의 존재 여부를 검사하는 부울 연산자
- 상호관련 서브 쿼리를 SELECT, UPDATE 및 DELETE 문에 사용 가능
- WITH 절을 사용하면 여러 번 발생하는 같은 질의 블록을 SELECT 문에서 사용 가능

ORACLE

18-30

Copyright © Oracle Corporation, 2001. All rights reserved.

요약(계속)

Oracle Server는 서브 쿼리가 상위 문에서 참조되는 테이블의 열을 참조할 때 상관 서브 쿼리를 수행합니다. 상호관련 서브 쿼리는 상위 문에서 처리되는 각 행에 대해 한 번씩 실행됩니다. 상위 문으로는 SELECT, UPDATE 또는 DELETE 문을 사용할 수 있습니다. WITH 절을 사용하면 다시 평가하는 데 비용이 많이 들고 복합 질의에서 여러 번 발생하는 같은 질의 블록을 다시 사용할 수 있습니다.

연습 18 개요

이 연습에서는 다음 내용을 다룹니다.

- 다중 열 서브 쿼리 작성
- 상호관련 서브 쿼리 작성
- EXISTS 연산자 사용
- 스칼라 서브 쿼리 사용
- WITH 절 사용

18-31

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

연습 18 개요

이 연습에서는 다중 열 서브 쿼리, 상호관련 서브 쿼리 및 스칼라 서브 쿼리를 작성합니다.
또한 WITH 절을 작성하여 문제를 해결합니다.

연습 18

1. 커미션을 받는 사원과 부서 번호 및 급여가 일치하는 사원의 이름, 부서 번호 및 급여를 표시하는 질의를 작성하십시오.

LAST_NAME	DEPARTMENT_ID	SALARY
Taylor	80	8600
Zlotkey	80	10500
Abel		11000

2. 위치 ID가 1700인 사원과 급여 및 커미션이 일치하는 사원의 이름, 부서 이름 및 급여를 표시하십시오.

LAST_NAME	DEPARTMENT_NAME	SALARY
Whalen	Administration	4400
Gietz	Accounting	8300
Higgins	Accounting	12000
Kochhar	Executive	17000
De Haan	Executive	17000
King	Executive	24000

6 rows selected.

3. Kochhar와 동일한 급여 및 커미션을 받는 모든 사원의 이름, 입사일 및 급여를 표시하는 질의를 작성하십시오.

참고: 결과 집합에 Kochhar는 표시하지 마십시오.

LAST_NAME	HIRE_DATE	SALARY
De Haan	13-JAN-93	17000

4. 모든 영업 관리자(JOB_ID = 'SA_MAN')보다 급여를 많이 받는 사원을 표시하는 질의를 작성하고 결과를 최고 급여에서 최저 급여의 순으로 정렬하십시오.

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Kochhar	AD_VP	17000
De Haan	AD_VP	17000
Hartstein	MK_MAN	13000
Higgins	AC_MGR	12000
Abel	SA_REP	11000

6 rows selected.

연습 18(계속)

5. T로 시작하는 도시에 사는 사원의 세부 정보인 사원 ID, 이름 및 부서 ID를 표시하십시오.

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
201	Hartstein	20
202	Fay	20

6. 소속 부서의 평균 급여보다 많은 급여를 받는 사원을 모두 찾는 질의를 작성하십시오. 이름, 급여, 부서 ID 및 소속 부서의 평균 급여를 표시하되, 평균 급여에 따라 정렬하십시오. 예제 출력과 같이 질의에 의해 검색되는 열에 별칭을 사용하십시오.

ENAME	SALARY	DEPTNO	DEPT_AVG
Moungos	5800	50	3500
Hunold	9000	60	6400
Hartstein	13000	20	9500
Abel	11000	80	10033.3333
Zlotkey	10500	80	10033.3333
Higgins	12000	110	10150
King	24000	90	19333.3333

7 rows selected.

7. 관리자가 아닌 사원을 모두 찾으십시오.

- a. 먼저 NOT EXISTS 연산자를 사용하여 이 작업을 수행하십시오.

LAST_NAME
Ernst
Lorentz
Rajs
Davies
Matos
Vargas
Abel
Taylor
Grant
Whalen
Fay
Gietz

12 rows selected.

- b. NOT IN 연산자를 사용하여 이 작업을 수행할 수 있습니까? 수행할 수 있는 경우 방법을 설명하고 수행할 수 없는 경우 이유를 설명하십시오.

연습 18(계속)

8. 소속 부서의 평균 급여보다 적은 급여를 받는 사원의 이름을 표시하는 질의를 작성하십시오.

LAST_NAME
Kochhar
De Haan
Ernst
Lorentz
Davies
Matos
Vargas
Taylor
Fay
Gietz

10 rows selected.

9. 소속 부서에 입사일이 늦지만 더 많은 급여를 받는 동료가 있는 사원의 이름을 표시하는 질의를 작성하십시오.

LAST_NAME
Rajs
Davies
Matos
Vargas
Taylor

연습 18(계속)

10. 모든 사원의 사원 ID, 이름 및 부서 이름을 표시하는 질의를 작성하십시오.

참고: 스칼라 서브 쿼리를 사용하여 SELECT 문에서 부서 이름을 검색하십시오.

EMPLOYEE_ID	LAST_NAME	DEPARTMENT
205	Higgins	Accounting
206	Gietz	Accounting
200	Whalen	Administration
100	King	Executive
101	Kochhar	Executive
102	De Haan	Executive
103	Hunold	IT
104	Ernst	IT
107	Lorentz	IT
201	Hartstein	Marketing
202	Fay	Marketing
149	Zlotkey	Sales
176	Taylor	Sales
174	Abel	Sales
EMPLOYEE_ID	LAST_NAME	DEPARTMENT
124	Mourgos	Shipping
141	Rajs	Shipping
142	Davies	Shipping
143	Matos	Shipping
144	Vargas	Shipping
178	Grant	

20 rows selected.

11. 총 급여가 전체 회사 총 급여의 1/8보다 많은 부서의 이름을 표시하는 질의를 작성하십시오.
이 질의를 WITH 절을 사용하여 작성하되, SUMMARY라는 이름을 부여하십시오.

DEPARTMENT_NAME	DEPT_TOTAL
Executive	58000
Sales	30100

제 총 19

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 계층 질의의 개념 설명
- 트리 구조로 된 보고서 생성
- 계층 데이터의 형식 지정
- 트리 구조에서 분기(**branch**) 제거

ORACLE

19-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 계층 질의를 사용하여 트리 구조의 보고서를 생성하는 방법을 설명합니다.

EMPLOYEES 테이블의

예제 데이터

EMPLOYEE_ID	LAST_NAME	JOB_ID	MANAGER_ID
100	King	AD_PRES	
101	Kochhar	AD_VP	100
102	De Haan	AD_VP	100
103	Hunold	IT_PROG	102
104	Ernst	IT_PROG	103
107	Lorentz	IT_PROG	103
124	Mourgos	ST_MAN	100
141	Rajs	ST_CLERK	124
142	Davies	ST_CLERK	124
143	Matos	ST_CLERK	124
144	Vargas	ST_CLERK	124
149	Zlotkey	SA_MAN	100
174	Abel	SA REP	149
176	Taylor	SA REP	149
EMPLOYEE_ID	LAST_NAME	JOB_ID	MANAGER_ID
178	Grant	SA REP	149
200	Whalen	AD_ASST	101
201	Hartstein	MK_MAN	100
202	Fay	MK REP	201
205	Higgins	AC_MGR	101
206	Gietz	AC_ACCOUNT	205

20 rows selected.

ORACLE

19-3

Copyright © Oracle Corporation, 2001. All rights reserved.

EMPLOYEES 테이블의 예제 데이터

계층 질의를 사용하면 테이블의 행 사이에 존재하는 자연스러운 계층 관계를 바탕으로 데이터를 검색할 수 있습니다. 관계형 데이터베이스에서는 계층 방식으로 레코드를 저장하지 않습니다. 그러나 단일 테이블의 행 사이에 계층 관계가 존재할 경우에는 트리 검색(tree walking)이라는 프로세스를 통해 계층 구조를 생성할 수 있습니다. 계층 질의는 트리의 분기(branch)를 순서대로 보고하는 방법 중 하나입니다.

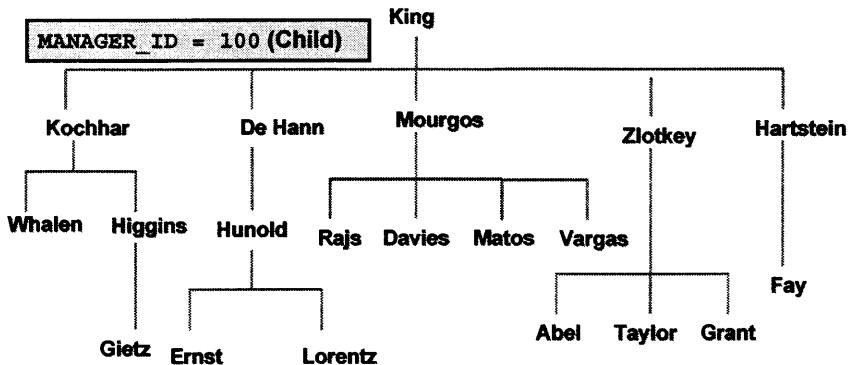
이를테면 가족 트리가 하나 있는데, 가장 윗사람은 트리의 시작 부분이나 줄기 쪽에 위치하고 가장 아랫사람은 트리의 가지(분기) 쪽에 있는 것과 같은 경우입니다. 가지에서는 다시 다른 가지가 뻗어 나올 수 있고 작은 가지에서도 다시 가지가 나올 수 있습니다.

계층 질의는 테이블의 행 사이에 관계가 존재할 때 사용할 수 있습니다. 예를 들어, 슬라이드에서 AD_VP, ST_MAN, SA_MAN 및 MK_MAN이라는 업무 ID를 가진 사원은 회사의 사장에게 직접 보고합니다. 그 이유는 이러한 레코드의 MANAGER_ID 열에 사장(AD_PRES)을 나타내는 사원 ID인 100이 들어 있기 때문입니다.

참고: 계층 트리는 가계도(가족 트리), 목축업(품종 개량용), 회사 관리(관리 계층), 제조업(부품 조립), 전화론 연구(종 전화) 및 과학적 연구와 같은 다양한 분야에서 사용할 수 있습니다.

자연 트리 구조

EMPLOYEE_ID = 100 (Parent)



ORACLE

19-4

Copyright © Oracle Corporation, 2001. All rights reserved.

자연 트리 구조

EMPLOYEES 테이블은 관리 보고 라인을 나타내는 트리 구조를 가질 수 있습니다. EMPLOYEE_ID 및 MANAGER_ID 열의 값이 같을 경우에 이들 사이에 존재하는 관계를 찾아 계층을 만들 수 있으며, 이 테이블을 자체 조인하여 이러한 관계를 이용할 수 있습니다. MANAGER_ID 열에는 관리자의 사원 번호가 들어 있습니다.

트리 구조의 부모-자식 관계를 이용하면 다음을 제어할 수 있습니다.

- 계층 검색 방향
- 계층 내 시작점

참고: 슬라이드는 EMPLOYEES 테이블에 있는 사원들의 관리 계층에 대한 트리 구조를 역으로 표시한 것입니다.

계층 질의

```
SELECT [LEVEL], column, expr...
FROM table
[WHERE condition(s)]
[START WITH condition(s)]
[CONNECT BY PRIOR condition(s)] ;
```

WHERE 조건:

```
expr comparison_operator expr
```

ORACLE

19-5

Copyright © Oracle Corporation, 2001. All rights reserved.

키워드 및 절

질의가 계층 질의인지 여부는 CONNECT BY 및 START WITH 절로 식별할 수 있습니다.

구문 설명:

SELECT

일반 SELECT 절입니다.

LEVEL

계층 질의에서 반환되는 각 행에 대해 LEVEL 의사 열은 루트 행에 대해서는 1, 루트의 자식 행에 대해서는 2와 같은 방식으로 값을 반환합니다.

FROM table

해당 열을 포함하는 테이블, 뷰 또는 스냅샷을 지정합니다. 하나의 테이블에서만 선택 가능합니다.

WHERE

질의에서 반환되는 행을 계층의 다른 행에 영향을 주지 않고 제한 합니다.

condition

표현식을 사용하여 비교합니다.

START WITH

계층의 루트 행(시작점)을 지정합니다. 이 절은 계층 질의에서 실제 계층을 반영하는 데 필수적입니다.

CONNECT BY

부모 행과 자식 행 사이에 관계가 존재하는 열을 지정합니다.

PRIOR

이 절은 계층 질의에 필수적입니다.

SELECT 문은 조인이나 조인을 포함하는 뷰에 대한 질의를 포함할 수 없습니다.

트리 검색

시작점

- 만족시켜야 할 조건을 지정합니다.
- 유효한 조건을 모두 받아들입니다.

```
START WITH column1 = value
```

EMPLOYEES 테이블을 사용하여 이름이 **Kochhar**인 사원
부터 시작합니다.

```
...START WITH last_name = 'Kochhar'
```

ORACLE

19-6

Copyright © Oracle Corporation, 2001. All rights reserved.

트리 검색

START WITH 절에서 트리의 루트로 사용될 행이 결정됩니다. START WITH 절에서는 유효한
조건을 결합하여 사용하는 것이 가능합니다.

예제

EMPLOYEES 테이블을 사용하여 회사 사장인 King부터 시작합니다.

```
... START WITH manager_id IS NULL
```

EMPLOYEES 테이블을 사용하여 Kochhar부터 시작합니다. START WITH 조건에 서브 쿼리를
포함시킬 수 있습니다.

```
... START WITH employee_id = (SELECT employee_id  
                           FROM   employees  
                           WHERE   last_name = 'Kochhar')
```

START WITH 절을 생략하면 테이블 내 모든 행을 루트 행으로 간주하고 트리 검색이 시작됩
니다. WHERE 절을 사용하면 WHERE 조건을 만족하는 모든 행부터 검색이 시작됩니다. 이러한 방식은
실제 계층을 반영하지 못합니다.

참고: CONNECT BY PRIOR 및 START WITH 절은 ANSI SQL 표준이 아닙니다.

트리 검색

```
CONNECT BY PRIOR column1 = column2
```

EMPLOYEES 테이블을 사용하여 위에서 아래로 검색합니다.

```
... CONNECT BY PRIOR employee_id = manager_id
```

방향

위에서 아래로 → Column1 = 부모 키
Column2 = 자식 키

아래에서 위로 → Column1 = 자식 키
Column2 = 부모 키

ORACLE

15-7

Copyright © Oracle Corporation, 2001. All rights reserved.

트리 검색(계속)

질의의 방향 즉, 질의가 부모부터 시작하여 자식 방향으로 수행될지 또는 자식부터 시작하여 부모 방향으로 수행될지는 CONNECT BY 절의 PRIOR 바로 뒤에 어떤 열이 오는가에 따라 결정됩니다. 이 경우는 PRIOR 연산자가 부모 행을 가리킵니다. Oracle Server에서는 이 부모 행의 자식을 찾기 위해 부모 행에 대해서는 PRIOR 식을 평가하고, 테이블 내 각 행에 대해 나머지 표현식을 평가합니다. 각 행에 대해 조건을 평가한 결과가 참이면 그 행은 해당 부모 행의 자식 행이 됩니다. Oracle Server는 항상 현재 부모 행을 기준으로 CONNECT BY 조건을 평가하여 자식 행을 선택합니다.

예제

EMPLOYEES 테이블을 사용하여 위에서 아래로 검색합니다. 부모 행의 EMPLOYEE_ID 값이 자식 행의 MANAGER_ID 값과 같은 계층 관계를 정의합니다.

```
... CONNECT BY PRIOR employee_id = manager_id
```

EMPLOYEES 테이블을 사용하여 아래에서 위로 검색합니다.

```
... CONNECT BY PRIOR manager_id = employee_id
```

PRIOR 연산자를 반드시 CONNECT BY 바로 뒤에 둘 필요는 없습니다. 따라서 다음 CONNECT BY PRIOR 절은 앞의 예제와 결과가 같습니다.

```
... CONNECT BY employee_id = PRIOR manager_id
```

참고: CONNECT BY 절은 서브 쿼리를 포함할 수 없습니다.

트리 검색: 아래에서 위로

```
SELECT employee_id, last_name, job_id, manager_id  
FROM employees  
START WITH employee_id = 101  
CONNECT BY PRIOR manager_id = employee_id ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	MANAGER_ID
101	Kochhar	AD_VP	100
100	King	AD_PRES	

ORACLE

19-8

Copyright © Oracle Corporation, 2001. All rights reserved.

트리 검색: 아래에서 위로

슬라이드의 예제는 사원 ID가 101인 사원부터 시작하여 관리자의 목록을 표시합니다.

예제

다음 예제의 경우 부모 행에 대해서는 EMPLOYEE_ID 값이 평가되고 자식 행에 대해서는 MANAGER_ID 및 SALARY 값이 평가됩니다. PRIOR 연산자는 EMPLOYEE_ID 값에만 적용됩니다.

```
... CONNECT BY PRIOR employee_id = manager_id  
          AND salary > 15000;
```

자식 행이 되려면 해당 행의 MANAGER_ID 값이 부모 행의 EMPLOYEE_ID 값과 같고 SALARY 값이 \$15,000보다 커야 합니다.

트리 검색: 위에서 아래로

```
SELECT last_name||' reports to '||  
PRIOR last_name "Walk Top Down"  
FROM employees  
START WITH last_name = 'King'  
CONNECT BY PRIOR employee_id = manager_id ;
```

Walk Top Down
King reports to
Kochhar reports to King
Whalen reports to Kochhar
Higgins reports to Kochhar

Zlotkey reports to King
Abel reports to Zlotkey
Taylor reports to Zlotkey
Grant reports to Zlotkey
Hartstein reports to King
Fay reports to Hartstein

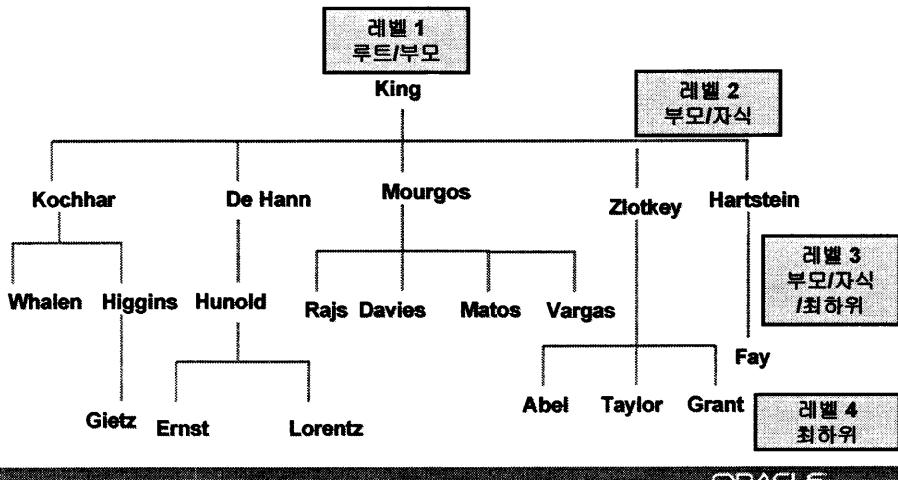
20 rows selected.

ORACLE

트리 검색: 위에서 아래로

위에서 아래로 검색하여 사원과 해당 관리자의 이름을 표시합니다. King을 시작점으로 사용하고 열은 하나만 출력합니다.

LEVEL 의사 열을 사용하여 순위 결정



19-10

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

LEVEL 의사 열을 사용하여 순위 결정

LEVEL 의사 열을 사용하면 계층에서 열의 순위 또는 레벨을 명시적(explicit)으로 표시할 수 있습니다. 이렇게 하면 보고서를 읽기가 쉬워집니다. 보다 큰 분기에서 하나 이상의 분기로 나뉘지는 지점을 노드라고 하며 분기의 끝을 최하위(leaf) 또는 최하위 노드라고 합니다. 슬라이드의 다이어그램은 역 트리의 노드와 해당 LEVEL 값을 보여줍니다. 예를 들어, 사원 Higgins는 부모 행이면서 동시에 자식 행이며 사원 Davies는 자식 행이면서 동시에 최하위 행입니다.

LEVEL 의사 열

값	레벨
1	루트 노드
2	루트 노드의 자식 노드
3	자식 노드의 자식 노드, 등

참고: 루트 노드는 역 트리에서 최상위의 노드이며 자식 노드는 루트가 아닌 노드입니다. 자식 노드가 있는 노드를 부모 노드라고 합니다. 최하위 노드는 자식 노드가 없는 노드입니다. 계층 질의에서 반환되는 레벨 숫자는 사용 가능한 사용자 메모리에 의해 제한을 받습니다.

슬라이드에서 King은 루트 또는 부모 행입니다(LEVEL = 1). Kochhar, De Hann, Mourgos, Zlotkey, Hartstein, Higgins 및 Hunold는 자식 행이면서 부모 행입니다(LEVEL = 2). Whalen, Rajs, Davies, Matos, Vargas, Gietz, Ernst, Lorentz, Abel, Taylor, Grant 및 Fay는 자식 행이면서 최하위 행입니다(LEVEL = 3 및 LEVEL = 4).

LEVEL 및 LPAD를 사용하여 계층 보고서 형식 지정

회사 관리 레벨을 가장 높은 레벨부터 시작하여 다음 레벨을 각각 들여쓰기하여 표시하는 보고서를 작성합니다.

```
COLUMN org_chart FORMAT A12
SELECT LPAD(last_name, LENGTH(last_name)+(LEVEL*2)-2, '_')
      AS org_chart
FROM   employees
START WITH last_name='King'
CONNECT BY PRIOR employee_id=manager_id
```

ORACLE

19-11

Copyright © Oracle Corporation, 2001. All rights reserved.

LEVEL을 사용하여 계층 보고서 형식 지정

트리의 노드는 루트부터 레벨 숫자가 할당됩니다. LPAD 함수를 의사 열 LEVEL과 함께 사용하면 계층 보고서를 들여쓰기된 트리 형태로 표시할 수 있습니다.

슬라이드 예제 설명:

- LPAD(char1, n [,char2])는 char1의 왼쪽에 char2 문자들을 채워 n의 길이로 반환합니다. 인수 n은 현재 단말기 화면에 표시될 반환 값의 총 길이입니다.
- LPAD(last_name, LENGTH(last_name)+(LEVEL*2)-2, '_')는 표시 형식을 정의합니다.
- char1은 LAST_NAME이고, n은 반환 값의 총 길이로 LAST_NAME +(LEVEL*2)-2이며, char2는 '_'입니다.

즉, SQL에서 LAST_NAME을 가져와서 결과 문자열의 길이가 LENGTH(last_name)+(LEVEL*2)-2가 될 때까지 왼쪽에 '_' 문자를 채워 넣습니다.

King은 LEVEL이 1입니다. 따라서 $(2 * 1) - 2 = 2 - 2 = 0$ 이므로 King 앞에 '_' 문자가 채워지지 않은 상태로 첫번째 열에 표시됩니다.

Kochhar는 LEVEL이 2입니다. 따라서 $(2 * 2) - 2 = 4 - 2 = 2$ 이므로 Kochhar 앞에 두 개의 '_' 문자가 채워진 상태로 들여쓰기되어 표시됩니다.

EMPLOYEES 테이블에 있는 나머지 코드도 같은 방식으로 표시됩니다.

LEVEL을 사용하여 계층 보고서 형식 지정(계속)

ORG_CHART

King
Kochhar
Whalen
Higgins
Gietz
De Haan
Hunold
Ernst
Lorentz
Mourgos
Rajs
Davies
Matos
Vargas

ORG_CHART

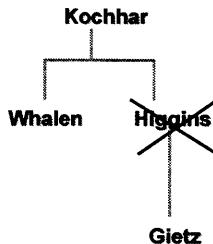
Zlotkey
Abel
Taylor
Grant
Hartstein
Fay

20 rows selected.

분기 제거

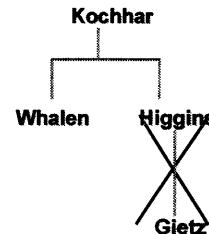
WHERE 절을 사용하여 노드를 제거합니다.

```
WHERE last_name != 'Higgins'
```



CONNECT BY 절을 사용하여 분기를 제거합니다.

```
CONNECT BY PRIOR  
employee_id = manager_id  
AND last_name != 'Higgins'
```



19-13

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

분기 제거

WHERE 및 CONNECT BY 절을 사용하여 트리의 일부를 제거할 수 있습니다. 즉, 표시될 노드 또는 행을 제어할 수 있습니다. 사용한 술어는 부울 조건처럼 동작합니다.

예제

루트부터 시작하여 위에서 아래로 검색한 결과에서 Higgins를 제거합니다. 이 때 자식 행은 그대로 듭니다.

```
SELECT department_id, employee_id, last_name, job_id, salary  
FROM employees  
WHERE last_name != 'Higgins'  
START WITH manager_id IS NULL  
CONNECT BY PRIOR employee_id = manager_id;
```

루트부터 시작하여 위에서 아래로 검색한 결과에서 Higgins 및 그 자식 행을 모두 제거합니다.

```
SELECT department_id, employee_id, last_name, job_id, salary  
FROM employees  
START WITH manager_id IS NULL  
CONNECT BY PRIOR employee_id = manager_id  
AND last_name != 'Higgins';
```

요약

이 단원에서 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- 계층 질의를 사용하여 테이블 내 행 사이의 계층 관계 보기
- 질의의 수행 방향 및 시작점 지정
- 노드 또는 분기 제거 가능

ORACLE

19-14

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

계층 질의를 사용하여 테이블 내 행 사이의 자연스러운 계층 관계에 따라 데이터를 검색할 수 있습니다. LEVEL 의사 열은 계층 트리에서 루트와의 거리를 나타냅니다. CONNECT BY PRIOR 절을 사용하여 질의가 수행될 방향을 지정할 수 있으며 START WITH 절을 사용하여 시작점을 지정할 수 있습니다. WHERE 및 CONNECT BY 절을 사용하여 트리 분기를 제거할 수 있습니다.

연습 19 개요

이 연습에서는 다음 내용을 다룹니다.

- 계층 질의와 비계층 질의 구별
- 트리 검색
- LEVEL 의사 열을 사용하여 들여쓰기된 보고서 작성
- 트리 구조의 일부 제거
- 출력 정렬

ORACLE

19-15

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 19 개요

이 연습에서는 계층 보고서를 작성해 봅니다.

문제

문제 1은 실습이 필요 없는 간단한 문제입니다.

연습 19

- 다음 출력을 보고 결과 출력이 계층 질의의 결과인지 여부를 판단하고 그 이유를 설명하십시오.

제출물 1:

EMPLOYEE_ID	LAST_NAME	MANAGER_ID	SALARY	DEPARTMENT_ID
100	King		24000	90
101	Kochhar	100	17000	90
102	De Haan	100	17000	90
201	Hartstein	100	13000	20
205	Higgins	101	12000	110
174	Abel	149	11000	80
149	Zlotkey	100	10500	80
103	Hunold	102	9000	60
■ ■ ■				
200	Whalen	101	4400	10
107	Lorentz	103	4200	60
141	Rajs	124	3500	50
142	Davies	124	3100	50
143	Matos	124	2600	50
144	Vargas	124	2500	50

20 rows selected.

제출물 2:

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
205	Higgins	110	Accounting
206	Gietz	110	Accounting
100	King	90	Executive
101	Kochhar	90	Executive
102	De Haan	90	Executive
149	Zlotkey	80	Sales
174	Abel	80	Sales
176	Taylor	80	Sales
103	Hunold	60	IT
104	Ernst	60	IT
107	Lorentz	60	IT

11 rows selected.

연습 19(계속)

제출물 3:

RANK	LAST_NAME
1	King
2	Kochhar
2	De Haan
3	Hunold
4	Ernst

2. Mourgos의 부서에 대한 조직 차트를 보여주는 보고서를 작성하십시오. 이름, 급여 및 부서 ID를 출력하십시오.

LAST_NAME	SALARY	DEPARTMENT_ID
Mourgos	5800	50
Rajs	3500	50
Davies	3100	50
Matos	2600	50
Vargas	2500	50

3. Lorentz의 관리자 계층을 보여주는 보고서를 작성하십시오. 직속 관리자를 먼저 표시 하십시오.

LAST_NAME
Hunold
De Haan
King

연습 19(계속)

4. LAST_NAME이 Kochhar인 사원부터 시작하여 관리 계층을 보여주는 들여쓰기된 보고서를 작성하십시오. 사원의 이름, 관리자 ID 및 부서 ID를 출력하십시오. 예제 출력처럼 열 이름에 별칭을 사용하십시오.

NAME	MGR	DEPTNO
Kochhar	100	90
Whalen	101	10
Higgins	101	110
Gietz	205	110

시간이 있을 때 다음 문제를 풀어보십시오.

5. 관리 계층을 보여주는 회사 조직 차트를 작성하십시오. 최상위 레벨부터 시작하되, 업무 ID가 IT_PROG인 사원, De Haan 및 De Haan이 관리하는 사원은 모두 제외시키십시오.

LAST_NAME	EMPLOYEE_ID	MANAGER_ID
King	100	
Kochhar	101	100
Whalen	200	101
Higgins	205	101
Gietz	206	205
Mourgos	124	100
Rajs	141	124
Davies	142	124
Matos	143	124
Vargas	144	124
Zlotkey	149	100
Abel	174	149
Taylor	176	149
Grant	178	149
LAST_NAME	EMPLOYEE_ID	MANAGER_ID
Hartstein	201	100
Fay	202	201

16 rows selected.

20

Oracle9i에서 확장된
DML 및 DDL 문 기능

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 다중 테이블 삽입 기능 설명
- 아래와 같은 다중 테이블 삽입 유형 사용
 - 무조건 **INSERT**
 - 피벗 **INSERT**
 - 조건 **ALL INSERT**
 - 조건 **FIRST INSERT**
- 외부 테이블 생성 및 사용
- 기본 키 제약 조건 생성 시 인덱스 명명

ORACLE

20-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 Oracle9i에서 확장된 DDL 및 DML 문의 기능에 대해 설명합니다. 여기서는 주로 다중 테이블 **INSERT** 문, 다중 테이블 **INSERT** 문의 유형 및 외부 테이블에 대해 설명하고 기본 키 제약 조건 생성 시 인덱스 이름을 지정하는 방법을 설명합니다.

INSERT 문 복습

- INSERT 문을 사용하여 테이블에 새 행을 추가합니다.

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

- 이 구문을 사용하면 한 번에 한 행만 삽입됩니다.

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

ORACLE

20-3

Copyright © Oracle Corporation, 2001. All rights reserved.

INSERT 문 복습

INSERT 문을 실행하여 테이블에 새 행을 추가할 수 있습니다.

구문 설명:

table 테이블 이름입니다.

column 데이터를 삽입할 테이블의 열 이름입니다.

value 열 값입니다.

참고: VALUES 절을 포함하는 명령문은 한 번에 한 행만 테이블에 추가합니다.

UPDATE 문 복습

- UPDATE 문을 사용하여 기존 행을 수정합니다.

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

- 한 번에 여러 행을 갱신할 수 있습니다.
- WHERE 절을 지정하여 특정 행을 수정합니다.

```
UPDATE employees  
SET department_id = 70  
WHERE employee_id = 142;  
1 row updated.
```

ORACLE

20-4

Copyright © Oracle Corporation, 2001. All rights reserved.

UPDATE 문 복습

UPDATE 문을 사용하여 기존 행을 수정할 수 있습니다.

구문 설명:

table	테이블 이름입니다.
column	데이터를 갱신할 테이블의 열 이름입니다.
value	열의 값 또는 서브 쿼리입니다.
condition	갱신될 행을 식별하며 열 이름, 표현식, 상수, 서브 쿼리 및 비교 연산자로 구성됩니다.

테이블을 질의해서 갱신된 행을 표시하여 갱신 작업을 확인합니다.

다중 테이블 INSERT 문 개요

- **INSERT...SELECT** 문을 단일 DML 문에 포함시키면 여러 테이블에 행을 삽입할 수 있습니다.
- 다중 테이블 **INSERT** 문을 데이터 웨어하우징 시스템에 사용하면 운영 중인 하나 이상의 소스에서 대상 테이블 집합으로 데이터를 전송할 수 있습니다.
- 다음을 비교해 보면 이 기능이 제공하는 성능 향상 효과를 확인할 수 있습니다.
 - 단일 DML과 다중 **INSERT...SELECT** 문의 비교
 - 단일 DML과 **IF...THEN** 구문을 사용한 다중 삽입 절차의 비교

ORACLE

20-6

Copyright © Oracle Corporation, 2001. All rights reserved.

다중 테이블 INSERT 문 개요

다중 테이블 **INSERT** 문은 서브 쿼리에서 반환한 행에 대해 계산을 수행한 다음 이 행을 하나 이상의 테이블에 삽입합니다.

다중 테이블 **INSERT** 문은 데이터 웨어하우스 시나리오에서 매우 유용한 역할을 수행할 수 있습니다. 데이터 웨어하우스는 정기적으로 로드되어 업무 분석에 이용되어야 하는데, 이렇게 하려면 운영 중인 하나 이상의 시스템에서 데이터를 추출하여 웨어하우스로 복사해야 합니다. 소스 시스템에서 데이터를 추출하여 데이터 웨어하우스로 가져오는 프로세스를 일반적으로 ETL이라고 합니다. 이는 추출(extraction), 변형(transformation) 및 로드/loading)를 나타냅니다.

추출 작업 동안에는 데이터베이스 시스템 및 응용 프로그램과 같은 다양한 소스에서 원하는 데이터를 식별하여 추출합니다. 추출된 데이터는 다음 작업을 위해 대상 시스템 또는 중간 시스템에 실제로 전송되어야 합니다. 선택한 전송 수단에 따라 이 프로세스에서 데이터 변형이 이루어질 수도 있습니다. 예를 들어, 게이트웨이를 통해 원격 대상에 직접 액세스하는 SQL 문은 SELECT 문에서 두 열을 연결할 수 있습니다.

일단 데이터가 Oracle9i 데이터베이스에 로드되면 SQL 작업을 통해 데이터가 변형됩니다. Oracle9i에서 다중 테이블 **INSERT** 문을 사용하는 것도 SQL 데이터 변형을 구현하는 방법 중 하나입니다.

다중 테이블 INSERT 문 개요(계속)

다중 테이블이 대상으로 포함된 경우 다중 테이블 INSERT 문에서 INSERT ... SELECT 문의 장점을 활용할 수 있습니다. Oracle9*i* 이전 버전에서 이 기능을 사용하면 *n*개의 INSERT ... SELECT 문을 처리해야 하므로 같은 소스 데이터가 *n*번 처리되며 변형 작업 로드도 *n*배 증가하게 됩니다.

기존 INSERT ... SELECT 문을 사용하면 빠른 성능을 위해 새 명령문이 병렬화되고 직접 로드 방식이 함께 사용되기도 합니다.

비관계형 데이터베이스 테이블과 같은 입력 스트림의 각 레코드가 관계형 데이터베이스 테이블 환경에 적합한 다중 레코드로 변환될 수 있습니다. Oracle9*i* 이전 버전에서 이 기능을 구현하려면 여러 개의 INSERT 문을 작성해야 합니다.

다중 테이블 INSERT 문의 유형

Oracle9i에서는 다음과 같은 유형의 다중 테이블 INSERT 문을 사용할 수 있습니다.

- 무조건 INSERT
- 조건 ALL INSERT
- 조건 FIRST INSERT
- 파벗 INSERT

ORACLE

20-7

Copyright © Oracle Corporation, 2001. All rights reserved.

다중 테이블 INSERT 문의 유형

Oracle9i에서는 다음 유형의 다중 테이블 INSERT 문이 도입되었습니다.

- 무조건 INSERT
- 조건 ALL INSERT
- 조건 FIRST INSERT
- 파벗 INSERT

서로 다른 절을 사용하여, 실행할 INSERT의 유형을 나타냅니다.

다중 테이블 INSERT 문

구문

```
INSERT [ALL] [conditional_insert_clause]  
[insert_into_clause values_clause] (subquery)
```

conditional_insert_clause

```
[ALL] [FIRST]  
[WHEN condition THEN] [insert_into_clause values_clause]  
[ELSE] [insert_into_clause values_clause]
```

다중 테이블 INSERT 문

슬라이드는 다중 테이블 INSERT 문의 일반적인 형태를 보여줍니다. 다중 테이블 INSERT 문에는 다음 네 가지 유형이 있습니다.

- 무조건 INSERT
- 조건 ALL INSERT
- 조건 FIRST INSERT
- 피벗 INSERT

무조건 INSERT: ALL into_clause

ALL 뒤에 여러 개의 insert_into_clause를 지정하여 무조건 다중 테이블 삽입을 수행합니다. Oracle Server는 서브 쿼리에서 반환된 각 행에 대해 각 insert_into_clause를 한 번씩 실행합니다.

조건 INSERT: conditional_insert_clause

conditional_insert_clause를 지정하여 조건 다중 테이블 삽입을 수행합니다. Oracle Server는 insert_into_clause의 실행 여부를 결정하는 해당 WHEN 조건을 통해 각 insert_into_clause를 필터링합니다. 하나의 다중 테이블 INSERT 문에는 WHEN 절을 최대 127개까지 사용할 수 있습니다.

조건 INSERT: ALL

ALL을 지정하면 Oracle Server는 다른 WHEN 절의 평가 결과에 상관 없이 각 WHEN 절을 평가합니다. Oracle Server는 조건이 참인 각 WHEN 절에 대해 해당 INTO 절 목록을 실행합니다.

다중 테이블 INSERT 문(계속)

조건 FIRST: INSERT

FIRST를 지정하면 Oracle Server는 나열된 순서대로 WHEN 절을 각각 평가합니다. 주어진 행이 첫번째 WHEN 절에서 참이면 Oracle Server는 해당 INTO 절을 실행한 다음 해당 행에 대해 이후의 WHEN 절을 건너뜁니다.

조건 INSERT: ELSE 절

주어진 행에 대해 WHEN 절이 모두 거짓이면 다음을 수행합니다.

- ELSE 절을 지정한 경우 Oracle Server는 ELSE 절과 관련된 INTO 절 목록을 실행합니다.
- ELSE 절을 지정하지 않은 경우 Oracle Server는 해당 행에 대해 아무 작업도 수행하지 않습니다.

다중 테이블 INSERT 문에 대한 제한

- 다중 테이블 삽입은 테이블에 대해서만 수행할 수 있으며 뷰나 구체화된 뷰에 대해서는 수행할 수 없습니다.
- 원격 테이블에는 다중 테이블 삽입을 수행할 수 없습니다.
- 다중 테이블 삽입을 수행할 때에는 테이블 collection 식을 지정할 수 없습니다.
- 다중 테이블 삽입에서는 모든 insert_into_clause를 통틀어 999개보다 많은 대상 열을 지정할 수 없습니다.

무조건 INSERT ALL

- EMPLOYEES 테이블에서 EMPLOYEE_ID가 200보다 큰 사원의 EMPLOYEE_ID, HIRE_DATE, SALARY 및 MANAGER_ID 값을 선택합니다.
- 다중 테이블 INSERT를 사용하여 이 값을 SAL_HISTORY 및 MGR_HISTORY 테이블에 삽입합니다.

```
INSERT ALL
```

```
  INTO sal_history VALUES(EMPID,HIREDATE,SAL)
  INTO mgr_history VALUES(EMPID,MGR,SAL)
 SELECT employee_id EMPID, hire_date HIREDATE,
        salary SAL, manager_id MGR
    FROM employees
   WHERE employee_id > 200;
8 rows created.
```

ORACLE

20-10

Copyright © Oracle Corporation, 2001. All rights reserved.

무조건 INSERT ALL

슬라이드의 예제는 SAL_HISTORY 및 MGR_HISTORY 테이블에 행을 삽입합니다.

SELECT 문을 사용하여 EMPLOYEES 테이블에서 사원 ID가 200보다 큰 사원의 사원 ID, 입사일, 급여 및 관리자 ID를 검색합니다. 그 다음 사원 ID, 입사일 및 급여를 SAL_HISTORY 테이블에 삽입하고 사원 ID, 관리자 ID 및 급여를 MGR_HISTORY 테이블에 삽입합니다.

이 INSERT 문은 무조건 INSERT라고 하며, SELECT 문에서 검색된 행에 제한이 적용되지 않습니다. SELECT 문에서 검색된 모든 행이 SAL_HISTORY 및 MGR_HISTORY 테이블에 삽입됩니다. INSERT 문의 VALUES 절은 SELECT 문에서 가져온 것으로서, 각 테이블에 삽입되어야 할 열을 지정합니다. SELECT 문에서 반환된 각 행은 SAL_HISTORY 테이블과 MGR_HISTORY 테이블에 한 번씩, 두 번 삽입됩니다.

피드백 “8 rows created”는 기본 테이블인 SAL_HISTORY 및 MGR_HISTORY에 대해 총 여덟 번의 삽입이 수행되었음을 나타냅니다.

조건 INSERT ALL

- EMPLOYEES 테이블에서 EMPLOYEE_ID가 200보다 큰 사원의 EMPLOYEE_ID, HIRE_DATE, SALARY 및 MANAGER_ID 값을 선택합니다.
- SALARY가 \$10,000보다 많으면 조건 다중 테이블 INSERT 문을 사용하여 SAL_HISTORY 테이블에 이 값을 삽입합니다.
- MANAGER_ID가 200보다 크면 조건 다중 테이블 INSERT 문을 사용하여 MGR_HISTORY 테이블에 이 값을 삽입합니다.

ORACLE

20-11

Copyright © Oracle Corporation, 2001. All rights reserved.

조건 INSERT ALL

조건 INSERT ALL 문에 대한 질문이 슬라이드에 나와 있습니다. 이 문제에 대한 답은 다음 페이지에서 보여줍니다.

조건 INSERT ALL

```
INSERT ALL
  WHEN SAL > 10000 THEN
    INTO sal_history VALUES (EMPID, HIREDATE, SAL)
  WHEN MGR > 200 THEN
    INTO mgr_history VALUES (EMPID, MGR, SAL)
  SELECT employee_id EMPID, hire_date HIREDATE,
        salary SAL, manager_id MGR
  FROM employees
  WHERE employee_id > 200;
4 rows created.
```

ORACLE

조건 INSERT ALL(계속)

슬라이드의 예제는 SAL_HISTORY 및 MGR_HISTORY 테이블에 행을 삽입하는 이전 슬라이드 예제와 비슷합니다. SELECT 문을 사용하여 EMPLOYEES 테이블에서 사원 ID가 200보다 큰 사원의 사원 ID, 입사일, 급여 및 관리자 ID를 검색합니다. 그 다음 사원 ID, 입사일 및 급여를 SAL_HISTORY 테이블에 삽입하고 사원 ID, 관리자 ID 및 급여를 MGR_HISTORY 테이블에 삽입합니다.

이 INSERT 문은 조건 ALL INSERT라고 하며, SELECT 문에서 검색된 행에 제한이 적용됩니다. SELECT 문에서 검색된 행 중 SAL 열의 값이 10000보다 큰 행만 SAL_HISTORY 테이블에 삽입되고, 마찬가지로 MGR 열의 값이 200보다 큰 행만 MGR_HISTORY 테이블에 삽입됩니다.

테이블에 여덟 개의 행이 삽입된 이전 예제와 달리 이 예제에서는 네 개의 행만 삽입됩니다.

피드백 “4 rows created”는 기본 테이블인 SAL_HISTORY 및 MGR_HISTORY에 대해 총 네 번의 삽입이 수행되었음을 나타냅니다.

조건 FIRST INSERT

- EMPLOYEES 테이블에서 DEPARTMENT_ID, SUM(SALARY) 및 MAX(HIRE_DATE)를 선택합니다.
- SUM(SALARY) 가 \$25,000보다 많으면 조건 FIRST 다중 테이블 INSERT를 사용하여 SPECIAL_SAL에 이 값을 삽입합니다.
- 첫번째 WHEN 절이 참이면 이 행에 대해서 이후의 WHEN 절을 건너뜁니다.
- 첫번째 WHEN 조건을 만족하지 않는 행은 조건 다중 테이블 INSERT를 사용하여 HIRE_DATE 열의 값에 따라 HIREDATE_HISTORY_00, HIREDATE_HISTORY_99 또는 HIREDATE_HISTORY 테이블에 삽입합니다.

ORACLE

20-13

Copyright © Oracle Corporation, 2001. All rights reserved.

조건 FIRST INSERT

조건 FIRST INSERT 문에 대한 질문이 슬라이드에 나와 있습니다. 이 문제에 대한 답은 다음 페이지에서 보여줍니다.

조건 FIRST INSERT

첫 번째 WHEN 절이 참이면 이전 WHEN 절은 거치지 않음.

```
INSERT FIRST
WHEN SAL > 25000           THEN
    INTO special_sal VALUES(DEPTID, SAL)
WHEN HIREDATE like ('%00%') THEN
    INTO hiredate_history_00 VALUES(DEPTID, HIREDATE)
WHEN HIREDATE like ('%99%') THEN
    INTO hiredate_history_99 VALUES(DEPTID, HIREDATE)
ELSE
    INTO hiredate_history VALUES(DEPTID, HIREDATE)
SELECT department_id DEPTID, SUM(salary) SAL,
       MAX(hire_date) HIREDATE
FROM   employees
GROUP BY department_id;
8 rows created.
```

ORACLE

20-14

Copyright © Oracle Corporation, 2001. All rights reserved.

조건 FIRST INSERT(계속)

슬라이드의 예제는 단일 INSERT 문을 사용하여 여러 테이블에 행을 삽입합니다. 먼저 SELECT 문을 사용하여 EMPLOYEES 테이블의 모든 부서에 대해 부서 ID, 총 급여 및 최근 입사일을 검색합니다.

이 INSERT 문을 조건 FIRST INSERT라고 하며, 총 급여가 \$25,000보다 많은 부서에 대해서는 예외를 적용합니다. WHEN ALL > 25000이라는 조건이 먼저 평가되어 부서의 총 급여가 \$25,000보다 많으면 해당 레코드가 입사일에 상관 없이 SPECIAL_SAL 테이블에 삽입됩니다. 이 첫 번째 WHEN 절이 참이면 Oracle Server는 해당 INTO 절을 실행하고 이 행에 대해서는 이후의 WHEN 절을 건너뜁니다.

첫 번째 WHEN 조건(WHEN SAL > 25000)을 만족하지 않는 행은 조건 INSERT 문에서처럼 나머지 조건이 평가됩니다. 따라서 SELECT 문에서 검색된 레코드는 HIREDATE 열의 값에 따라 HIREDATE_HISTORY_00, HIREDATE_HISTORY_99 또는 HIREDATE_HISTORY 테이블에 삽입됩니다.

피드백 “8 rows created”는 기본 테이블인 SPECIAL_SAL, HIREDATE_HISTORY_00, HIREDATE_HISTORY_99 및 HIREDATE_HISTORY에 대해 총 여덟 번의 INSERT 문이 수행되었음을 나타냅니다.

피벗 INSERT

- 비관계형 데이터베이스 테이블
`SALES_SOURCE_DATA`에서 다음 형식으로 판매
레코드 집합을 받았다고 가정
합니다.

`EMPLOYEE_ID, WEEK_ID, SALES_MON,`
`SALES_TUE, SALES_WED, SALES_THUR,`
`SALES_FRI`

- 이 레코드를 다음과 같이 일반적인 관계형 형식으로
`SALES_INFO` 테이블에 저장하려고 합니다.

`EMPLOYEE_ID, WEEK, SALES`

- 피벗 INSERT를 사용하면 비관계형 데이터베이스
테이블의 판매 레코드 집합을 관계형 형식으로 변환할
수 있습니다.

ORACLE

20-15

Copyright © Oracle Corporation, 2001. All rights reserved.

피벗 INSERT

피벗은 비관계형 데이터베이스 테이블과 같은 입력 스트림의 각 레코드를 관계형 데이터
베이스 테이블 환경에 적합한 여러 레코드로 변환해야 하는 경우에 필요한 작업입니다.

슬라이드에 언급된 문제를 해결하려면 원래 비관계형 데이터베이스 테이블인
`SALES_SOURCE_DATA`의 각 레코드를 데이터 웨어하우스의 `SALES_INFO` 테이블에 적합한 다섯
개의 레코드로 변환해야 합니다. 이 작업을 피벗이라고 합니다.

피벗 INSERT 문에 대한 질문이 슬라이드에 나와 있습니다. 이 문제에 대한 답은 다음 페이지
에서 보여줍니다.

피벗 INSERT 가로로 진행액을 세로로 길게 펼쳐서

```
INSERT ALL
INTO sales_info VALUES (employee_id, week_id, sales_MON)
INTO sales_info VALUES (employee_id, week_id, sales_TUE)
INTO sales_info VALUES (employee_id, week_id, sales_WED)
INTO sales_info VALUES (employee_id, week_id, sales_THUR)
INTO sales_info VALUES (employee_id, week_id, sales_FRI)
SELECT employee_id, week_id, sales_MON, sales_TUE,
       sales_WED, sales_THUR, sales_FRI
  FROM sales_source_data;
5 rows created.
```

ORACLE

20-16

Copyright © Oracle Corporation, 2001. All rights reserved.

피벗 INSERT(계속)

슬라이드의 예제는 비관계형 데이터베이스 테이블인 SALES_SOURCE_DATA에서 판매 데이터를 받습니다. 이 데이터는 특정 주 ID를 갖는 주에 대해 요일별로 영업 사원의 판매 실적 세부 정보를 포함하고 있습니다.

```
DESC SALES_SOURCE_DATA
```

Name	Null?	Type
EMPLOYEE_ID		NUMBER(6)
WEEK_ID		NUMBER(2)
SALES_MON		NUMBER(8,2)
SALES_TUE		NUMBER(8,2)
SALES_WED		NUMBER(8,2)
SALES_THUR		NUMBER(8,2)
SALES_FRI		NUMBER(8,2)

파벳 INSERT(계속)

```
SELECT * FROM SALES_SOURCE_DATA;
```

EMPLOYEE_ID	WEEK_ID	SALES_MON	SALES_TUE	SALES_WED	SALES_THUR	SALES_FRI
176	6	2000	3000	4000	5000	6000

```
DESC SALES_INFO
```

Name	Null?	Type
EMPLOYEE_ID		NUMBER(6)
WEEK		NUMBER(2)
SALES		NUMBER(8,2)

```
SELECT * FROM sales_info;
```

EMPLOYEE_ID	WEEK	SALES
176	6	2000
176	6	3000
176	6	4000
176	6	5000
176	6	6000

파벳 INSERT를 사용하는 앞의 예제에서는 SALES_SOURCE_DATA 테이블의 한 행이 관계형 테이블인 SALES_INFO에서 다섯 개의 레코드로 변환됩니다.

외부 테이블

- 외부 테이블은 데이터베이스 외부의 플랫 파일에 데이터가 저장되어 있는 읽기 전용 테이블입니다.
- 외부 테이블에 대한 메타 데이터는 **CREATE TABLE** 문을 사용하여 만듭니다.
- 외부 테이블을 이용하여 오라클 데이터를 플랫 파일에 저장하거나 언로드할 수 있습니다.
- **SQL**을 사용하여 데이터를 질의할 수 있지만 **DML**을 사용하거나 인덱스를 생성할 수는 없습니다.

ORACLE

20-18

Copyright © Oracle Corporation, 2001. All rights reserved.

외부 테이블

외부 테이블이란 메타 데이터는 데이터베이스에 저장되었지만 데이터는 데이터베이스 외부에 저장되어 있는 읽기 전용 테이블입니다. Oracle9i 외부 테이블 기능을 사용하면 외부 데이터를 가상 테이블로 사용할 수 있습니다. 외부 데이터를 먼저 데이터베이스에 로드하지 않고도 이 데이터를 직접 병렬로 조인하거나 질의할 수 있습니다. 외부 테이블의 데이터를 질의하려면 SQL, PL/SQL 및 Java를 사용합니다.

외부 테이블과 정규 테이블의 주된 차이점은 외부에서 구성된 테이블은 읽기 전용이라는 것입니다. 따라서 DML 작업(UPDATE, INSERT 또는 DELETE)을 수행할 수 없으며 인덱스를 생성할 수도 없습니다.

외부 테이블에 대한 메타 데이터를 정의하려면 CREATE TABLE ... ORGANIZATION EXTERNAL 문을 사용합니다. 이 외부 테이블 정의는 외부 데이터를 먼저 데이터베이스에 로드하지 않고도 외부 데이터에 대해 SQL 질의를 실행할 수 있도록 해주는 뷰로 생각할 수 있습니다.

Oracle Server는 외부 테이블에 대한 기본 액세스 드라이버를 두 개 제공합니다. 하나는 로더 액세스 드라이버 즉, ORACLE_LOADER입니다. 이것은 오라클로더 기술을 사용하여 외부 파일로부터 데이터를 읽어오는 데 사용됩니다. 이 액세스 드라이버를 사용하면 Oracle Server에서도 SQL*Loader 유ти리티를 통해야만 해석 가능한 형식의 데이터 소스 데이터에 액세스할 수 있습니다. 오라클에 포함된 또 하나의 액세스 드라이버는 임포트/엑스포트 액세스 드라이버 즉, ORACLE_INTERNAL입니다. 이것은 플랫폼에 구애 받지 않는 형식의 사용하여 데이터를 임포트 및 엑스포트하는 데 사용됩니다.

외부 테이블 생성

- **CREATE TABLE** 구문과 함께 **external_table_clause**를 사용하여 외부 테이블을 생성합니다.
- **ORGANIZATION**을 **EXTERNAL**로 지정하여 테이블이 데이터베이스 외부에 있음을 나타냅니다.
- **external_table_clause**는 액세스 드라이버 **TYPE**, **external_data_properties** 및 **REJECT LIMIT**로 구성됩니다.
- **external_data_properties**는 다음과 같은 요소로 구성됩니다.
 - **DEFAULT DIRECTORY**
 - **ACCESS PARAMETERS**

LOCATION

ORACLE

20-19

Copyright © Oracle Corporation, 2001. All rights reserved.

외부 테이블 생성

CREATE TABLE 문의 **ORGANIZATION EXTERNAL** 절을 사용하여 외부 테이블을 생성합니다. 사실 테이블을 생성하는 것이 아니라 외부 데이터에 액세스하는 데 사용할 수 있는 메타 데이터를 데이터 딕셔너리에 만드는 것입니다. **ORGANIZATION** 절에는 테이블의 데이터 행이 저장되는 순서를 지정합니다. **ORGANIZATION** 절에 **EXTERNAL**을 지정하여 해당 테이블이 데이터베이스 외부에 있는 읽기 전용 테이블이라는 것을 나타냅니다.

TYPE access_driver_type은 외부 테이블의 액세스 드라이버를 나타냅니다. 액세스 드라이버는 외부 데이터를 데이터베이스에 맞게 해석하는 API(Application Program Interface)입니다. **TYPE**을 지정하지 않으면 기본 액세스 드라이버인 **ORACLE_LOADER**가 사용됩니다.

REJECT LIMIT 절에서는 외부 데이터의 질의를 수행하는 동안 몇 개의 변환 오류가 발생해야 오라클 오류를 반환하고 질의를 중지할 것인지를 지정합니다. 기본값은 0입니다.

DEFAULT DIRECTORY를 사용하면 외부 데이터 소스가 상주할 파일 시스템의 디렉토리에 대응되는 기본 디렉토리 객체를 하나 이상 지정할 수 있습니다. 기본 디렉토리는 액세스 드라이버에서 오류 로그와 같은 보조 파일을 저장하는 데도 사용할 수 있습니다. 여러 디스크 드라이브에서의 로드 벨련성을 위해 여러 기본 디렉토리를 사용할 수 있습니다.

선택 사항인 **ACCESS PARAMETERS** 절을 사용하면 이 외부 테이블에 대한 특정 액세스 드라이버의 파라미터에 값을 할당할 수 있습니다. 오라클에서는 이 절에 대해 해석 작업을 수행하지 않으며, 이 정보를 외부 데이터의 상황에 맞게 해석하는 것은 액세스 드라이버입니다.

LOCATION 절을 사용하면 각 외부 데이터 소스에 대해 외부 위치자를 하나씩 지정할 수 있습니다. 일반적으로 **location_specifier**는 파일이지만 반드시 파일일 필요는 없습니다. 오라클에서는 이 절을 해석하지 않으며, 이 정보를 외부 데이터의 상황에 맞게 해석하는 것은 액세스 드라이버입니다.

외부 테이블 생성 예제

외부 데이터 소스가 상주할 파일 시스템의 디렉토리에 대응되는 DIRECTORY 객체를 생성합니다.

```
CREATE DIRECTORY emp_dir AS '/flat_files' ;
```

OS File의 디렉토리명

ORACLE

20-20

Copyright © Oracle Corporation, 2001. All rights reserved.

외부 테이블 생성 예제

CREATE DIRECTORY 문을 사용하여 디렉토리 객체를 생성합니다. 디렉토리 객체에는 외부 데이터 소스가 상주하는 서버의 파일 시스템 디렉토리에 대한 별칭을 지정합니다. 외부 데이터 소스를 참조할 때 파일 관리를 유연하게 하기 위해 운영 체제 경로 이름 대신 디렉토리 이름을 사용할 수도 있습니다.

디렉토리를 생성하려면 CREATE ANY DIRECTORY 시스템 권한이 있어야 합니다. 디렉토리를 생성하면 자동으로 READ 객체 권한이 부여되며 다른 사용자와 룰(role)에 READ 권한을 부여 할 수 있습니다. DBA도 이 권한을 다른 사용자와 룰에 부여할 수 있습니다.

구문

```
CREATE [OR REPLACE] DIRECTORY AS 'path_name';
```

구문 설명:

OR REPLACE OR REPLACE를 지정하면 이미 존재하는 디렉토리 데이터베이스 객체를 다시 생성할 수 있습니다. 이 절을 사용하면 기존 디렉토리를 삭제하고 다시 생성하여 이전에 해당 디렉토리에 부여되었던 데이터베이스 객체 권한을 다시 부여하지 않고도 기존 디렉토리의 정의를 변경할 수 있습니다. 다시 정의된 디렉토리에 대한 권한을 이전에 부여받았던 사용자는 권한을 다시 부여받지 않아도 해당 디렉토리에 액세스할 수 있습니다.

directory 생성될 디렉토리 객체의 이름을 지정합니다. 디렉토리 이름의 최대 길이는 30바이트입니다. 디렉토리 객체에 스키마 이름을 지정할 수 없습니다.

'path_name' 운영 체제 디렉토리의 전체 경로 이름(대/소문자 구분)을 지정합니다.

외부 테이블 생성 예제

```
CREATE TABLE oldemp (
    empno NUMBER, empname CHAR(20), birthdate DATE)
ORGANIZATION EXTERNAL 외부 테이블입니다.
(TYPE ORACLE_LOADER
DEFAULT DIRECTORY emp_dir 디렉토리 명
ACCESS PARAMETERS
(RECORDS DELIMITED BY NEWLINE
BADFILE 'bad_emp'
LOGFILE 'log_emp'
FIELDS TERMINATED BY ','
(empno CHAR,
empname CHAR,
birthdate CHAR date_format date mask "dd-mon-yyyy"))
LOCATION ('empl.txt')
PARALLEL 5 파일이 있는 폴더명.
REJECT LIMIT 200;
Table created.
```

ORACLE

20-21

Copyright © Oracle Corporation, 2001. All rights reserved.

외부 테이블 생성 예제(계속)

다음 형식의 레코드를 포함하는 플랫 파일이 있다고 가정합니다.

10,jones,11-Dec-1934
20,smith,12-Jun-1972

레코드는 줄 바꿈으로 구분되며 필드는 모두 쉼표(,)로 끝납니다. 파일 이름은 /flat_files/empl.txt입니다.

이 파일을 외부 테이블에 대한 데이터 소스로 변환하고, 해당 메타 데이터를 데이터베이스에 상주하도록 하려면 다음 단계를 수행해야 합니다.

1. 다음과 같이 디렉토리 객체 emp_dir을 생성합니다.

```
CREATE DIRECTORY emp_dir AS '/flat_files' ;
```

2. 슬라이드에 표시된 것처럼 CREATE TABLE 명령을 실행합니다.

슬라이드의 예제는 다음 파일에 대한 외부 테이블을 생성하는 테이블 명세를 보여줍니다.

```
/flat_files/empl.txt
```

예제의 TYPE 명세는 사용 방법을 보여주기 위해 표시한 것입니다. TYPE을 지정하지 않으면 ORACLE_LOADER를 기본 액세스 드라이버로 사용합니다. ACCESS PARAMETERS는 특정 액세스 드라이버의 파라미터에 값을 제공하며 Oracle Server가 아닌 액세스 드라이버에 의해 해석됩니다.

PARALLEL 절을 사용하면 INSERT INTO TABLE 문을 실행할 때 다섯 개의 병렬 실행 서버로 하여금 외부 데이터 소스를 동시에 스캔하도록 만들 수 있습니다. 예를 들어, PARALLEL=5를 지정하면 하나 이상의 병렬 실행 서버가 데이터 소스에 대해 작업을 수행합니다. 외부 테이블의 크기가 매우 클 수 있으므로 질의에 PARALLEL 절 또는 병렬 힌트를 지정하여 성능을 향상 시키는 것이 좋습니다.

외부 테이블 정의 예제

REJECT LIMIT 절에서 지정하는 내용은 외부 데이터를 질의하는 동안 200개 이상의 변환 오류가 발생할 경우 질의를 중지하고 오류를 반환한다는 것입니다. 이러한 변환 오류는 액세스 드라이버가 외부 테이블 정의에 맞춰 데이터 파일의 데이터를 변환하려고 할 때 발생할 수 있습니다.

일단 CREATE TABLE 명령이 성공적으로 실행되면 외부 테이블 OLDEMP를 관계형 테이블처럼 구조를 표시하고 질의할 수 있습니다.

```
DESC oldemp
```

Name	Null?	Type
EMPNO		NUMBER
EMPNAME		CHAR(20)
BIRTHDATE		DATE

다음 예제는 INSERT INTO TABLE 문을 사용하여 외부 데이터 소스의 데이터를 해당 데이터를 처리할 Oracle SQL 엔진으로 보내는 데이터 흐름을 생성합니다. 외부 테이블에서 데이터가 추출될 때 이 외부 데이터 표현은 ORACLE_LOADER 액세스 드라이버에 의해 오라클 고유의 동등한 표현으로 투명하게 변환됩니다. 다음 INSERT 문은 외부 테이블 OLDEMP의 데이터를 BIRTHDAYS 테이블에 삽입합니다.

```
INSERT INTO birthdays(empno, empname, birthdate)
    SELECT empno, empname, birthdate
      FROM   oldemp;
```

```
2 rows created.
```

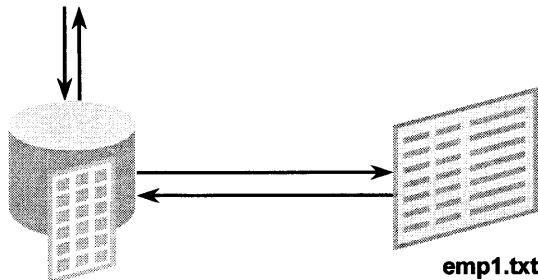
이제 BIRTHDAYS 테이블에서 선택할 수 있습니다.

```
SELECT * FROM birthdays;
```

EMPNO	EMPNAME	BIRTHDATE
10	jones	11-DEC-34
20	smith	12-JUN-97

외부 테이블 질의

```
SELECT *
FROM oldemp
```



20-23

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

외부 테이블 질의

외부 테이블에는 데이터베이스에 저장된 데이터에 관한 정보도 없고, 외부 소스에 데이터가 어떤 방식으로 저장되었는지에 대한 정보도 없습니다. 대신 외부 테이블 쿼리에서 서버에 어떤 방식으로 데이터를 제공해야 하는지를 기술하고 있습니다. 외부 테이블 정의와 일치하도록 데이터 파일의 데이터를 변환하는 작업은 액세스 드라이버와 외부 테이블 쿼리에서 담당합니다.

데이터베이스 서버가 외부 소스의 데이터에 액세스할 때는 적절한 액세스 드라이버를 호출하여 데이터베이스 서버가 원하는 형식으로 외부 소스의 데이터를 가져옵니다.

데이터 소스에 들어 있는 데이터는 외부 테이블의 정의와 다르다는 점을 명심해야 합니다. 소스 파일의 필드 수는 테이블 열의 수보다 많거나 적을 수 있습니다. 데이터 소스 필드의 데이터 유형도 테이블 열의 데이터 유형과 다를 수 있습니다. 액세스 드라이버는 데이터 소스의 데이터를 처리하여 외부 테이블의 정의와 일치되도록 만듭니다.

CREATE TABLE 문 내의 CREATE INDEX

```
CREATE TABLE NEW_EMP
(employee_id NUMBER(6)
 PRIMARY KEY USING INDEX
 (CREATE INDEX emp_id_idx ON
 NEW_EMP(employee_id)),
first_name VARCHAR2(20),
last_name VARCHAR2(25));
Table created.
```

```
SELECT INDEX_NAME, TABLE_NAME
FROM USER_INDEXES
WHERE TABLE_NAME = 'NEW_EMP';
```

INDEX_NAME	TABLE_NAME
EMP_ID_IDX	NEW_EMP

ORACLE

20-24

Copyright © Oracle Corporation, 2001. All rights reserved.

CREATE TABLE 문 내의 CREATE INDEX

슬라이드의 예제는 CREATE INDEX 절을 CREATE TABLE 문과 함께 사용하여 기본 키 인덱스를 명시적(explicit)으로 생성합니다. 이 향상된 기능은 Oracle9i에서 제공됩니다. Oracle Server가 인덱스를 생성하였으나 사용자가 인덱스 이름을 제어할 수 없었던 이전 버전과 달리 지금은 PRIMARY 키를 생성할 때 인덱스의 이름을 지정할 수 있습니다. 다음 예제는 이러한 내용을 보여줍니다.

```
CREATE TABLE EMP_UNNAMED_INDEX
(employee_id NUMBER(6) PRIMARY KEY ,
 first_name VARCHAR2(20),
last_name VARCHAR2(25));
Table created.
```

```
SELECT INDEX_NAME, TABLE_NAME
FROM USER_INDEXES
WHERE TABLE_NAME = 'EMP_UNNAMED_INDEX';
```

INDEX_NAME	TABLE_NAME
SYS_C002835	EMP_UNNAMED_INDEX

Oracle Server가 PRIMARY KEY 열에 대해 인덱스를 생성하여 이름을 부여하는 것을 보면 이름이 복잡하여 쉽게 이해하기 어렵습니다. Oracle9i에서는 CREATE TABLE 문을 통해 테이블을 생성할 때 PRIMARY KEY 열의 인덱스에 이름을 지정할 수 있습니다. 그러나 Oracle9i 이전 버전의 경우 제약 조건을 생성할 때 기본 키 제약 조건의 이름을 지정하면 인덱스도 제약 조건과 같은 이름으로 생성됩니다.

요약

이 단원에서 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- **INSERT...SELECT** 문을 단일 **DML** 문에 포함시키면 여러 테이블에 행 삽입 가능
- 외부 테이블 생성
- **CREATE TABLE** 문에 **CREATE INDEX** 문을 함께 사용하여 인덱스의 이름 지정

ORACLE

20-25

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

Oracle 9i에서는 다음 유형의 다중 테이블 **INSERT** 문을 사용할 수 있습니다.

- 무조건 **INSERT**
- 조건 **ALL INSERT**
- 조건 **FIRST INSERT**
- 피벗 **INSERT**

external_table_clause를 사용하여 외부 테이블을 생성합니다. 이 테이블은 읽기 전용이며 해당 메타 데이터가 데이터베이스에 저장되지만 데이터는 데이터베이스 외부에 저장됩니다. 외부 테이블을 사용하면 데이터를 먼저 데이터베이스에 로드할 필요 없이 데이터를 질의할 수 있습니다.

Oracle9i에서는 **CREATE TABLE** 문을 통해 테이블을 생성할 때 **PRIMARY KEY** 열 인덱스의 이름을 지정할 수 있습니다.

연습 20 개요

이 연습에서는 다음 내용을 다룹니다.

- 무조건 **INSERT** 문 작성
- 조건 **ALL INSERT** 문 작성
- 피벗 **INSERT** 문
- **CREATE TABLE** 명령과 함께 인덱스 생성

ORACLE

20-28

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 20 개요

이 연습에서는 다중 테이블 **INSERT** 문을 작성하고 **CREATE TABLE** 명령을 통해 테이블을 생성할 때 **CREATE INDEX** 명령을 사용합니다.

연습 20

1. lab 폴더에서 `cre_sal_history.sql` 스크립트를 실행하여 `SAL_HISTORY` 테이블을 생성하십시오.
2. `SAL_HISTORY` 테이블의 구조를 표시하십시오.

Name	Null?	Type
EMPLOYEE_ID		NUMBER(6)
HIRE_DATE		DATE
SALARY		NUMBER(8,2)

3. lab 폴더에서 `cre_mgr_history.sql` 스크립트를 실행하여 `MGR_HISTORY` 테이블을 생성하십시오.
4. `MGR_HISTORY` 테이블의 구조를 표시하십시오.

Name	Null?	Type
EMPLOYEE_ID		NUMBER(6)
MANAGER_ID		NUMBER(6)
SALARY		NUMBER(8,2)

5. lab 폴더에서 `cre_special_sal.sql` 스크립트를 실행하여 `SPECIAL_SAL` 테이블을 생성하십시오.
6. `SPECIAL_SAL` 테이블의 구조를 표시하십시오.

Name	Null?	Type
EMPLOYEE_ID		NUMBER(6)
SALARY		NUMBER(8,2)

7. a. 다음을 수행하는 질의를 작성하십시오.

- `EMPLOYEES` 테이블에서 사원 ID가 125보다 작은 사원의 사원 ID, 입사일, 급여 및 관리자 ID를 검색하십시오.
- 급여가 \$20,000보다 많으면 `SPECIAL_SAL` 테이블에 사원 ID 및 급여를 삽입하십시오.
- `SAL_HISTORY` 테이블에 사원 ID, 입사일 및 급여를 삽입하십시오.
- `MGR_HISTORY` 테이블에 사원 ID, 관리자 ID 및 급여를 삽입하십시오.

연습 20(계속)

b. SPECIAL_SAL 테이블의 레코드를 표시하십시오.

EMPLOYEE_ID	SALARY
100	24000

c. SAL_HISTORY 테이블의 레코드를 표시하십시오.

EMPLOYEE_ID	HIRE_DATE	SALARY
101	21-SEP-89	17000
102	13-JAN-93	17000
103	03-JAN-90	9000
104	21-MAY-91	6000
107	07-FEB-99	4200
124	16-NOV-99	5800

6 rows selected.

d. MGR_HISTORY 테이블의 레코드를 표시하십시오.

EMPLOYEE_ID	MANAGER_ID	SALARY
101	100	17000
102	100	17000
103	102	9000
104	103	6000
107	103	4200
124	100	5800

6 rows selected.

연습 20(계속)

8. a. lab 폴더에서 cre_sales_source_data.sql 스크립트를 실행하여 SALES_SOURCE_DATA 테이블을 생성하십시오.
- b. lab 폴더에서 ins_sales_source_data.sql 스크립트를 실행하여 SALES_SOURCE_DATA 테이블에 레코드를 삽입하십시오.
- c. SALES_SOURCE_DATA 테이블의 구조를 표시하십시오.

Name	Null?	Type
EMPLOYEE_ID		NUMBER(6)
WEEK_ID		NUMBER(2)
SALES_MON		NUMBER(8,2)
SALES_TUE		NUMBER(8,2)
SALES_WED		NUMBER(8,2)
SALES_THUR		NUMBER(8,2)
SALES_FRI		NUMBER(8,2)

- d. SALES_SOURCE_DATA 테이블의 레코드를 표시하십시오.

EMPLOYEE_ID	WEEK_ID	SALES_MON	SALES_TUE	SALES_WED	SALES_THUR	SALES_FRI
178	6	1750	2200	1500	1500	3000

- e. lab 폴더에서 cre_sales_info.sql 스크립트를 실행하여 SALES_INFO 테이블을 생성하십시오.
- f. SALES_INFO 테이블의 구조를 표시하십시오.

Name	Null?	Type
EMPLOYEE_ID		NUMBER(6)
WEEK		NUMBER(2)
SALES		NUMBER(8,2)

연습 20(계속)

g. 다음을 수행하는 질의를 작성하십시오.

SALES_SOURCE_DATA 테이블에서 사원 ID, 주 ID, 월요일 판매 실적, 화요일 판매 실적, 수요일 판매 실적, 목요일 판매 실적 및 금요일 판매 실적을 검색하십시오.

SALES_SOURCE_DATA 테이블에서 검색된 각 레코드를 SALES_INFO 테이블의 여러 레코드로 변환하십시오.

힌트: 괴짜 INSERT 문을 사용하십시오.

h. SALES_INFO 테이블의 레코드를 표시하십시오.

EMPLOYEE_ID	WEEK	SALES
178	6	1750
178	6	2200
178	6	1500
178	6	1500
178	6	3000

9. a. 다음 테이블 인스턴스 차트에 따라 DEPT_NAMED_INDEX 테이블을 생성하십시오.

PRIMARY KEY 열에 대한 인덱스의 이름을 DEPT_PK_IDX로 지정하십시오.

열 이름	Deptno	Dname
기본 키	예	
데이터 유형	Number	VARCHAR2
길이	4	30

b. USER_INDEXES 테이블을 질의하여 DEPT_NAMED_INDEX 테이블의 INDEX_NAME을 표시하십시오.

INDEX_NAME	TABLE_NAME
DEPT_PK_IDX	DEPT_NAMED_INDEX

A

해답

연습 1 해답

1. 강사가 제공한 사용자 ID와 암호를 사용하여 iSQL*Plus 세션을 시작합니다.
2. iSQL*Plus 명령으로 데이터베이스에 액세스합니다.
False
3. 다음 SELECT 문은 성공적으로 실행됩니다.

True

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

4. 다음 SELECT 문은 성공적으로 실행됩니다.

True

```
SELECT *
FROM   job_grades;
```

5. 이 명령문에는 코딩 오류가 네 개 있습니다. 식별할 수 있습니까?

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

- EMPLOYEES 테이블에는 sal이라는 열이 없습니다. 해당 열의 이름은 SALARY입니다.
- 꼽하기 연산자는 2번 줄에 나타나 있는 x가 아니라 *입니다.
- 별칭 ANNUAL SALARY는 공백을 포함할 수 없습니다. ANNUAL_SALARY로 쓰거나 큰 따옴표로 묶어야 합니다.
- LAST_NAME 열 다음에 쉼표가 없습니다.

6. DEPARTMENTS 테이블의 구조를 표시하고 테이블의 모든 데이터를 선택하십시오.

```
DESCRIBE departments
```

```
SELECT *
FROM   departments;
```

7. EMPLOYEES 테이블의 구조를 표시하십시오. 사원 번호가 가장 앞에 오고 이어서 각 사원의 이름, 업무 코드, 입사일이 오도록 질의를 작성하십시오. HIRE_DATE 열에 STARTDATE라는 별칭을 지정하십시오. SQL 문을 lab1_7.sql이라는 파일에 저장하십시오.

```
DESCRIBE employees
```

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

연습 1 해답(계속)

8. lab1_7.sql 파일의 질의를 실행하십시오.

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

9. EMPLOYEES 테이블의 업무 코드를 중복되지 않게 표시하는 질의를 작성하십시오.

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

시간이 있을 때 다음 문제를 풀어보십시오.

10. lab1_7.sql의 명령문을 iSQL*Plus 편집 창으로 복사하십시오. 머리글을 각각 Emp #, Employee, Job 및 Hire Date로 명명한 다음 질의를 다시 실행하십시오.

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

11. 업무 ID와 이름을 연결한 다음 쉼표 및 공백으로 구분하여 표시하고 열 이름을 Employee and Title로 지정하십시오.

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

좀 더 연습하려면 다음 문제를 풀어보십시오.

12. EMPLOYEES 테이블의 모든 데이터를 표시하는 질의를 작성하십시오. 각 열은 쉼표로 구분하고 열 이름은 THE_OUTPUT으로 지정하십시오.

```
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;
```

연습 2 해답

1. 급여가 \$12,000를 넘는 사원의 이름과 급여를 표시하는 질의를 작성하여 lab2_1.sql이라는 이름의 텍스트 파일에 저장한 후 질의를 실행하십시오.

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

2. 사원 번호가 176인 사원의 이름과 부서 번호를 표시하는 질의를 작성하십시오.

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

3. 급여가 \$5,000에서 \$12,000 사이에 포함되지 않는 모든 사원의 이름과 급여를 표시하도록 lab2_1.sql 파일을 수정한 다음 이 SQL 문을 lab2_3.sql이라는 이름의 파일에 저장하십시오.

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

4. 1998년 2월 20일과 1998년 5월 1일 사이에 입사한 사원의 이름, 업무 ID 및 시작일을 표시하되, 시작일을 기준으로 오름차순으로 정렬하십시오.

```
SELECT last_name, job_id, hire_date
FROM   employees
WHERE  hire_date BETWEEN '20-Feb-1998' AND '01-May-1998'
ORDER BY hire_date;
```

연습 2 해답(계속)

5. 부서 20 및 50에 속하는 모든 사원의 이름과 부서 번호를 이름을 기준으로 영문자순으로 표시하십시오.

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

6. 급여가 \$5,000와 \$12,000 사이이고 부서 번호가 20 또는 50인 사원의 이름과 급여를 나열 하도록 lab2_3.sql을 수정하십시오. 열 레이블을 Employee와 Monthly Salary로 각각 지정하고 lab2_3.sql을 lab2_6.sql로 다시 저장한 다음 lab2_6.sql의 명령문을 실행하십시오.

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

7. 1994년에 입사한 모든 사원의 이름과 입사일을 표시하십시오.

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

8. 관리자가 없는 모든 사원의 이름과 업무를 표시하십시오.

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

9. 커미션을 받는 모든 사원의 이름, 급여 및 커미션을 급여 및 커미션을 기준으로 내림차순으로 정렬하여 표시하십시오.

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

연습 2 해답(계속)

시간이 있을 때 다음 문제를 풀어보십시오.

10. 이름의 세번째 문자가 *a*인 모든 사원의 이름을 표시하십시오.

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

11. 이름에 *a*와 *e*가 있는 모든 사원의 이름을 표시하십시오.

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

좀 더 연습하려면 다음 문제를 풀어보십시오.

12. 업무가 영업 사원 또는 사무원이면서 급여가 \$2,500, \$3,500 또는 \$7,000가 아닌 모든 사원의 이름, 업무 및 급여를 표시하십시오.

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

13. 커미션 비율이 20%인 모든 사원의 이름, 급여 및 커미션을 표시하도록 lab2_6.sql을 수정하십시오. lab2_6.sql을 lab2_13.sql로 다시 저장한 다음 lab2_13.sql의 명령문을 다시 실행하십시오.

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

연습 3 해답

1. 현재 날짜를 표시하는 질의를 작성하고 열 레이블을 Date로 지정하십시오.

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

2. 각 사원에 대해 사원 번호, 이름, 급여 및 15% 인상된 급여를 정수로 표시하십시오. 인상된 급여 열의 레이블을 New Salary로 지정하십시오. SQL 문을 lab3_2.sql이라는 이름의 텍스트 파일에 저장하십시오.

```
SELECT    employee_id, last_name, salary,  
         ROUND(salary * 1.15, 0) "New Salary"  
FROM      employees;
```

3. lab3_2.sql 파일의 질의를 실행하십시오.

```
SELECT    employee_id, last_name, salary,  
         ROUND(salary * 1.15, 0) "New Salary"  
FROM      employees;
```

4. lab3_2.sql의 질의를 수정하여 새 급여에서 이전 급여를 빼는 새 열을 추가하고 레이블을 Increase로 지정하십시오. 파일의 내용을 lab3_4.sql로 저장하고 수정한 질의를 실행하십시오.

```
SELECT    employee_id, last_name, salary,  
         ROUND(salary * 1.15, 0) "New Salary",  
         ROUND(salary * 1.15, 0) - salary "Increase"  
FROM      employees;
```

5. 이름이 J, A 또는 M으로 시작하는 모든 사원의 이름(첫 글자는 대문자로, 나머지 글자는 소문자로 표시) 및 이름 길이를 표시하는 질의를 작성하고 각 열에 적합한 레이블을 지정하십시오. 결과를 사원의 이름에 따라 정렬하십시오.

```
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 해답(계속)

6. 각 사원의 이름을 표시하고 근무 달 수(입사일로부터 현재까지의 달 수)를 계산하여 열 레이블을 MONTHS_WORKED로 지정하십시오. 결과는 정수로 반올림하여 표시하고 근무 달 수를 기준으로 정렬하십시오.

참고: 결과가 다를 수 있습니다.

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

7. 각 사원에 대해 다음 항목을 생성하는 질의를 작성하십시오.

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

```
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;
```

시간이 있을 때 다음 문제를 풀어보십시오.

8. 모든 사원의 이름과 급여를 표시하는 질의를 작성하십시오. 급여는 15자 길이로 왼쪽에 \$ 기호가 채워진 형식으로 표기하고 열 레이블을 SALARY로 지정하십시오.

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

9. 사원의 이름, 입사일 및 급여 검토일을 표시하십시오. 급여 검토일은 여섯 달이 경과한 후 첫번째 월요일입니다. 열 레이블을 REVIEW로 지정하고 날짜는 “Monday, the Thirty-First of July, 2000”과 같은 형식으로 표시되도록 지정하십시오.

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

10. 이름, 입사일 및 업무 시작 요일을 표시하고 열 레이블을 DAY로 지정하십시오. Monday를 시작으로 해서曜일을 기준으로 결과를 정렬하십시오.

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

연습 3 해답(계속)

좀 더 연습하려면 다음 문제를 풀어보십시오.

11. 사원의 이름과 커미션 합계를 표시하는 질의를 작성하십시오. 커미션을 받지 않는 사원일 경우 “No Commission”을 표시하십시오. 열 레이블은 COMM으로 지정하십시오.

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

12. 사원의 이름을 표시하고 급여 총액을 별표(*)로 나타내는 질의를 작성하십시오. 각 별표는 1,000달러를 나타냅니다. 급여를 기준으로 데이터를 내림차순으로 정렬하고 열 레이블을 EMPLOYEES_AND_THEIR_SALARIES로 지정하십시오.

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

13. DECODE 함수를 사용하여 다음 데이터에 따라 JOB_ID 열의 값을 기준으로 모든 사원의 등급을 표시하는 질의를 작성하십시오.

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

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

연습 3 해답(계속)

14. 앞 문제의 명령문을 CASE 구문을 사용하여 재작성하십시오.

```
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 'O' END GRADE
FROM employees;
```

연습 4 해답

- 모든 사원의 이름, 부서 번호, 부서 이름을 표시하는 질의를 작성하십시오.

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id;
```

- 부서 80에 속하는 모든 업무의 고유 목록을 작성하고 출력 결과에 부서의 위치를 포함시키십시오.

```
SELECT DISTINCT job_id, location_id
FROM employees, departments
WHERE employees.department_id = departments.department_id
AND employees.department_id = 80;
```

- 커미션을 받는 모든 사원의 이름, 부서 이름, 위치 ID 및 도시를 표시하는 질의를 작성하십시오.

```
SELECT e.last_name, d.department_name, d.location_id, l.city
FROM employees e, departments d, locations l
WHERE e.department_id = d.department_id
AND
d.location_id = l.location_id
AND e.commission_pct IS NOT NULL;
```

- 이름에 *a*(소문자)가 포함된 모든 사원의 이름과 부서 이름을 표시하고 해당 SQL 문을 lab4_4.sql 파일에 저장하십시오.

```
SELECT last_name, department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id
AND last_name LIKE '%a%';
```

연습 4 해답(계속)

5. Toronto에서 근무하는 모든 사원의 이름, 업무, 부서 번호 및 부서 이름을 표시하는 질의를 작성하십시오.

```
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';
```

6. 사원의 이름 및 사원 번호를 관리자의 이름 및 관리자 번호와 함께 표시하고 각각의 열 레이블을 Employee, Emp#, Manager, Mgr#로 지정하십시오. 해당 SQL 문을 lab4_6.sql 파일에 저장하십시오.

```
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);
```

연습 4 해답(계속)

7. lab4_6.sql을 수정하되 King을 포함하여 관리자가 없는 모든 사원을 표시하도록 하고 결과를 사원 번호를 기준으로 정렬하십시오.
해당 SQL 문을 lab4_7.sql 파일에 저장하고 lab4_7.sql의 질의를 실행하십시오.

```
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);
```

시간이 있을 때 다음 문제를 풀어보십시오.

8. 지정한 사원의 이름, 부서 번호 및 지정한 사원과 동일한 부서에서 근무하는 모든 사원을 표시하도록 질의를 작성하고 각 열에 적합한 레이블을 지정하십시오.

```
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;
```

9. JOB_GRADES 테이블의 구조를 표시하고 모든 사원의 이름, 업무, 부서 이름, 급여 및 등급을 표시하는 질의를 작성하십시오.

```
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;  
-- OR  
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);
```

연습 4 해답(계속)

좀 더 연습하려면 다음 문제를 풀어보십시오.

10. Davies라는 사원보다 늦게 입사한 사원의 이름과 입사일을 표시하는 질의를 작성하십시오.

```
SELECT e.last_name, e.hire_date
FROM   employees e, employees davies
WHERE  davies.last_name = 'Davies'
AND    davies.hire_date < e.hire_date
-- OR
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;
```

11. 관리자보다 먼저 입사한 모든 사원의 이름 및 입사일을 관리자의 이름 및 입사일과 함께 표시하고 열 레이블을 각각 Employee, Emp Hired, Manager, Mgr Hired로 지정하십시오.

```
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;
-- OR
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;
```

연습 5 해답

다음 세 문장의 유효성을 판별하여 True 또는 False로 답하십시오.

- 그룹 함수는 여러 행에 적용되어 그룹 당 하나의 결과를 출력합니다.

True

- 그룹 함수는 계산에 널을 포함합니다.

False. 그룹 함수는 널 값을 무시합니다. 널 값을 포함시키려면 NVL 함수를 사용하십시오.

- WHERE 절은 그룹 계산에 행(row)을 포함시키기 전에 행을 제한합니다.

True

- 모든 사원의 급여 최고액, 최저액, 총액 및 평균액을 표시하십시오. 열 레이블을 각각 Maximum, Minimum, Sum 및 Average로 지정하고 결과를 정수로 반올림한 후 작성한 SQL 문을 lab5_4.sql이라는 파일에 저장하십시오.

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

- lab5_4.sql의 질의를 수정하여 각 업무 유형별로 급여 최고액, 최저액, 총액 및 평균액을 표시하십시오. lab5_4.sql을 lab5_5.sql로 다시 저장하고 lab5_5.sql의 명령문을 실행하십시오.

```
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;
```

연습 5 해답(계속)

6. 업무가 동일한 사원 수를 표시하는 질의를 작성하십시오.

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

7. 관리자는 나열하지 말고 관리자 수를 확인하십시오. 열 레이블은 Number of Managers로 지정하십시오. 힌트: MANAGER_ID 열을 사용하여 관리자 수를 확인하십시오.

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

8. 최고 급여와 최저 급여의 차액을 표시하는 질의를 작성하고 열 레이블을 DIFFERENCE로 지정하십시오.

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

시간이 있을 때 다음 문제를 풀어보십시오.

9. 관리자 번호 및 해당 관리자에 속한 사원의 최저 급여를 표시하십시오.

관리자를 알 수 없는 사원 및 최저 급여가 \$6,000 미만인 그룹은 제외시키고 결과를 급여에 대한 내림차순으로 정렬하십시오.

```
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;
```

10. 각 부서에 대해 부서 이름, 위치, 사원 수, 부서 내 모든 사원의 평균 급여를 표시하는 질의를 작성하고, 열 레이블을 각각 Name, Location, Number of People 및 Salary로 지정하십시오. 평균 급여는 소수점 둘째 자리로 반올림하십시오.

```
SELECT    d.department_name "Name", d.location_id "Location",
          COUNT(*) "Number of People",
          ROUND(AVG(salary),2) "Salary"
FROM      employees e, departments d
WHERE     e.department_id = d.department_id
GROUP BY d.department_name, d.location_id;
```

연습 5 해답(계속)

좀 더 연습하려면 다음 문제를 풀어보십시오.

11. 총 사원 수 및 1995, 1996, 1997, 1998년에 입사한 사원 수를 표시하는 질의를 작성하고 적합한 열 머리글을 작성하십시오.

```
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;
```

12. 업무를 표시한 다음 해당 업무에 대해 부서 번호별 급여 및 부서 20, 50, 80 및 90의 급여 총액을 각각 표시하는 행렬 질의를 작성하고 각 열에 적합한 머리글을 지정하십시오.

```
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;
```

연습 6 해답

- Zlotkey와 동일한 부서에 속한 모든 사원의 이름과 입사일을 표시하는 질의를 작성하십시오.
오. Zlotkey는 제외하십시오.

```
SELECT last_name, hire_date
  FROM employees
 WHERE department_id = (SELECT department_id
                           FROM employees
                          WHERE last_name = 'Zlotkey')
   AND last_name <> 'Zlotkey';
```

- 급여가 평균 급여보다 많은 모든 사원의 사원 번호와 이름을 표시하는 질의를 작성하고 결과를
급여에 대해 오름차순으로 정렬하십시오.

```
SELECT employee_id, last_name, salary
  FROM employees
 WHERE salary > (SELECT AVG(salary)
                   FROM employees)
 ORDER BY salary;
```

- 이름에 *u*가 포함된 사원과 같은 부서에서 일하는 모든 사원의 사원 번호와 이름을 표시하는
질의를 작성하고 작성한 SQL 문을 lab6_3.sql이라는 파일에 저장한 다음 질의를 실행하
십시오.

```
SELECT employee_id, last_name
  FROM employees
 WHERE department_id IN (SELECT department_id
                           FROM employees
                          WHERE last_name like '%u%');
```

- 부서 위치 ID가 1700인 모든 사원의 이름, 부서 번호 및 업무 ID를 표시하십시오.

```
SELECT last_name, department_id, job_id
  FROM employees
 WHERE department_id IN (SELECT department_id
                           FROM departments
                          WHERE location_id = 1700);
```

연습 6 해답(계속)

5. King에게 보고하는 모든 사원의 이름과 급여를 표시하십시오.

```
SELECT last_name, salary
  FROM employees
 WHERE manager_id = (SELECT employee_id
                        FROM employees
                       WHERE last_name = 'King');
```

6. Executive 부서의 모든 사원에 대한 부서 번호, 이름 및 업무 ID를 표시하십시오.

```
SELECT department_id, last_name, job_id
  FROM employees
 WHERE department_id IN (SELECT department_id
                           FROM departments
                          WHERE department_name = 'Executive');
```

시간이 있을 때 다음 문제를 풀어보십시오.

7. lab6_3.sql의 질의를 수정하여 평균 급여보다 많은 급여를 받고 이름에 *u*가 포함된 사원과 같은 부서에서 근무하는 모든 사원의 사원 번호, 이름 및 급여를 표시하십시오.
lab6_3.sql을 lab6_7.sql로 다시 저장하고 lab6_7.sql의 명령문을 실행하십시오.

```
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);
```

연습 7 해답

다음 문장의 유효성을 판별하여 True 또는 False로 답하십시오.

1. 다음 명령문은 유효합니다.

```
DEFINE & p_val = 100
```

False

DEFINE은 **DEFINE p_val=100**과 같이 지정해야 유효한 문장입니다. &는 SQL 코드 내에서 사용됩니다.

2. DEFINE 명령은 SQL 명령입니다.

False

DEFINE 명령은 **SQL*Plus** 명령입니다.

3. 지정한 범위 내에서 업무를 시작한 모든 사원의 이름, 업무 및 입사일을 표시하는 스크립트를 작성하십시오. 이름과 업무는 공백과 쉼표로 분리하여 연결하고 열 레이블을 Employees로 지정하십시오. 별도의 SQL 스크립트 파일에서 DEFINE 명령을 사용하여 두 범위를 제공하되, MM/DD/YYYY 형식을 사용하십시오. 스크립트 파일을 lab7_3a.sql 및 lab7_3b.sql이라는 이름으로 저장하십시오.

```
-- lab file lab7_3a.sql
SET ECHO OFF
SET VERIFY OFF
DEFINE low_date = 01/01/1998
DEFINE high_date = 01/01/1999

-- lab file lab7_3a.sql
SELECT last_name ||', '|| job_id EMPLOYEES, hire_date
FROM employees
WHERE hire_date BETWEEN TO_DATE('&low_date', 'MM/DD/YYYY')
                     AND TO_DATE('&high_date', 'MM/DD/YYYY')
/
UNDEFINE low_date
UNDEFINE high_date
SET VERIFY ON
SET ECHO ON
```

연습 7 해답(계속)

4. 지정한 위치에 근무하는 사원의 이름, 업무 ID 및 부서 이름을 표시하는 스크립트를 작성 하십시오. 부서 위치에 대한 검색 조건은 대소문자를 구분하지 않도록 하고 스크립트 파일 을 lab7_4.sql이라는 이름으로 저장하십시오.

```
SET ECHO OFF
SET VERIFY OFF
COLUMN last_name HEADING "EMPLOYEE NAME"
COLUMN department_name HEADING "DEPARTMENT NAME"
SELECT e.last_name, e.job_id, d.department_name
FROM employees e, departments d, locations l
WHERE e.department_id = d.department_id
AND l.location_id = d.location_id
AND l.city = INITCAP('&p_location')
/
COLUMN last_name CLEAR
COLUMN department_name CLEAR
SET VERIFY ON
SET ECHO ON
```

연습 7 해답(계속)

5. lab7_4.sql을 수정하여 지정한 위치에 근무하는 각 사원에 대한 부서 이름, 사원 이름, 입사일, 급여 및 연봉을 포함하는 보고서를 작성하십시오. 열 레이블을 DEPARTMENT NAME, EMPLOYEE NAME, START DATE, SALARY 및 ANNUAL SALARY로 지정하여 여러 줄에 표시되도록 하십시오. 이 스크립트를 lab7_5.sql이라는 이름으로 다시 저장하고 스크립트의 명령을 실행하십시오.

```
SET ECHO OFF
SET FEEDBACK OFF
SET VERIFY OFF
BREAK ON department_name
COLUMN department_name HEADING "DEPARTMENT|NAME"
COLUMN last_name HEADING "EMPLOYEE|NAME"
COLUMN hire_date HEADING "START|DATE"
COLUMN salary HEADING "SALARY" FORMAT $99,990.00
COLUMN asal HEADING "ANNUAL|SALARY" FORMAT $99,990.00
SELECT d.department_name, e.last_name, e.hire_date,
       e.salary, e.salary*12 asal
  FROM departments d, employees e, locations l
 WHERE e.department_id = d.department_id
   AND d.location_id    = l.location_id
   AND l.city            = '&p_location'
 ORDER BY d.department_name
/
COLUMN department_name CLEAR
COLUMN last_name CLEAR
COLUMN hire_date CLEAR
COLUMN salary CLEAR
COLUMN asal CLEAR
CLEAR BREAK
SET VERIFY ON
SET FEEDBACK ON
SET ECHO ON
```

연습 8 해답

MY_EMPLOYEE 테이블에 데이터를 삽입하십시오.

1. lab8_1.sql 스크립트의 명령문을 실행하여 실습에 사용할 MY_EMPLOYEE 테이블을 생성하십시오.

```
CREATE TABLE my_employee
(id NUMBER(4) CONSTRAINT my_employee_id_nn NOT NULL,
last_name VARCHAR2(25),
first_name VARCHAR2(25),
userid VARCHAR2(8),
salary NUMBER(9,2));
```

2. MY_EMPLOYEE 테이블의 구조를 표시하여 열 이름을 식별하십시오.

```
DESCRIBE my_employee
```

3. 다음 예제 데이터의 첫 번째 데이터 행(row)을 MY_EMPLOYEE 테이블에 추가하십시오. 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. 위의 목록에 있는 예제 데이터의 두 번째 행을 MY_EMPLOYEE 테이블에 추가하십시오. 이번에는 INSERT 절에 열을 명시적으로 나열하십시오.

```
INSERT INTO my_employee (id, last_name, first_name,
                           userid, salary)
VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

5. 테이블에 추가한 항목을 확인하십시오.

```
SELECT *
FROM     my_employee;
```

연습 8 해답(계속)

6. loademp.sql이라는 텍스트 파일에 MY_EMPLOYEE 테이블로 행을 로드하는 insert 문을 작성하십시오. 이름의 첫 글자와 성의 처음 일곱 글자를 연결하여 사용자 ID를 만드십시오.

```
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
```

7. 작성한 스크립트의 insert 문을 실행하여 예제 데이터의 그 다음 두 행(row)을 테이블에 추가하십시오.

```
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
```

8. 테이블에 추가한 항목을 확인하십시오.

```
SELECT *
FROM my_employee;
```

9. 추가한 데이터를 영구히 저장하십시오.

```
COMMIT;
```

연습 8 해답(계속)

MY_EMPLOYEE 테이블의 데이터를 생성 및 삭제하십시오.

10. 사원 3의 성을 Drexler로 변경하십시오.

```
UPDATE my_employee  
SET last_name = 'Drexler'  
WHERE id = 3;
```

11. 급여가 900 미만인 모든 사원의 급여를 1000으로 변경하십시오.

```
UPDATE my_employee  
SET salary = 1000  
WHERE salary < 900;
```

12. 테이블의 변경 내용을 확인하십시오.

```
SELECT last_name, salary  
FROM my_employee;
```

13. MY_EMPLOYEE 테이블에서 Betty Dancs를 삭제하십시오.

```
DELETE  
FROM my_employee  
WHERE last_name = 'Dancs';
```

14. 테이블의 변경 내용을 확인하십시오.

```
SELECT *  
FROM my_employee;
```

15. 보류 중인 변경 내용을 모두 커밋하십시오.

```
COMMIT;
```

MY_EMPLOYEE 테이블에 대한 데이터 트랜잭션을 제어하십시오.

16. 6 단계에서 작성한 스크립트의 명령문을 수정하여 예제 데이터의 마지막 행(row)을 테이블에 추가하고 스크립트의 명령문을 실행하십시오.

```
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
```

연습 8 해답(계속)

17. 테이블에 추가한 항목을 확인하십시오.

```
SELECT *  
FROM my_employee;
```

18. 트랜잭션 수행 중에 저장점을 표시하십시오.

```
SAVEPOINT step_18;
```

19. 테이블의 내용을 모두 삭제하십시오.

```
DELETE  
FROM my_employee;
```

20. 테이블 내용이 비어 있는지 확인하십시오.

```
SELECT *  
FROM my_employee;
```

21. 이전의 INSERT 작업은 버리지 말고 최근의 DELETE 작업만 버리십시오.

```
ROLLBACK TO step_18;
```

22. 새 행이 그대로 있는지 확인하십시오.

```
SELECT *  
FROM my_employee;
```

23. 추가한 데이터를 영구히 저장하십시오.

```
COMMIT;
```

연습 9 해답

- 다음 테이블 인스턴트 차트를 기반으로 DEPT 테이블을 생성하십시오. Lab9_1.sql이라는 스크립트에 구문을 입력하고 스크립트의 명령문을 실행하여 테이블을 생성한 후, 테이블이 생성되었는지 확인하십시오.

열 이름	ID	NAME
키 유형		
널/고유		
FK 테이블		
FK 열		
데이터 유형	Number	VARCHAR2
길이	7	25

```
CREATE TABLE dept
(id NUMBER(7),
name VARCHAR2(25));
```

```
DESCRIBE dept
```

- DEPARTMENT 테이블의 데이터를 DEPT 테이블에 추가하십시오. 필요한 열만 추가하십시오.

```
INSERT INTO dept
SELECT department_id, department_name
FROM departments;
```

- 다음 테이블 인스턴스 차트를 기반으로 EMP 테이블을 생성하십시오. Lab9_3.sql이라는 스크립트에 구문을 입력하고 스크립트의 명령문을 실행하여 테이블을 생성한 후, 테이블이 생성되었는지 확인하십시오.

열 이름	ID	LAST_NAME	FIRST_NAME	DEPT_ID
키 유형				
널/고유				
FK 테이블				
FK 열				
데이터 유형	Number	VARCHAR2	VARCHAR2	Number
길이	7	25	25	7

연습 9 해답(계속)

```
CREATE TABLE emp
(id          NUMBER(7),
last_name    VARCHAR2(25),
first_name   VARCHAR2(25),
dept_id      NUMBER(7));
```

```
DESCRIBE emp
```

4. 긴 성을 가진 사원의 성을 표시할 수 있도록 EMP 테이블을 수정한 후 수정 내용을 확인 하십시오.

```
ALTER TABLE emp
MODIFY (last_name  VARCHAR2(50));
```

```
DESCRIBE emp
```

5. DEPT 및 EMP 테이블이 모두 데이터 딕셔너리에 저장되었는지 확인하십시오(힌트: USER_TABLES).

```
SELECT  table_name
FROM    user_tables
WHERE   table_name IN ('DEPT', 'EMP');
```

6. EMPLOYEES 테이블 구조를 기반으로 EMPLOYEES2 테이블을 생성하십시오. EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY 및 DEPARTMENT_ID 열만 포함 시키고 새 테이블의 열 이름을 각각 ID, FIRST_NAME, LAST_NAME, SALARY 및 DEPT_ID로 지정하십시오.

```
CREATE TABLE employees2 AS
SELECT  employee_id id, first_name, last_name, salary,
        department_id dept_id
FROM    employees;
```

7. EMP 테이블을 삭제하십시오.

```
DROP TABLE emp;
```

8. EMPLOYEES2 테이블의 이름을 EMP로 변경하십시오.

```
RENAME employees2 TO emp;
```

연습 9 해답(계속)

9. DEPT 및 EMP 테이블 정의에 테이블을 설명하는 주석을 추가한 후 데이터 딕셔너리에서 추가한 항목을 확인하십시오.

```
COMMENT ON TABLE emp IS 'Employee Information';
COMMENT ON TABLE dept IS 'Department Information';
SELECT *
FROM   user_tab_comments
WHERE  table_name = 'DEPT'
OR     table_name = 'EMP';
```

10. EMP 테이블에서 FIRST_NAME 열을 삭제한 후 테이블 설명을 참조하여 수정 내용을 확인하십시오.

```
ALTER TABLE emp
DROP COLUMN FIRST_NAME;

DESCRIBE emp
```

11. EMP 테이블의 DEPT_ID 열을 UNUSED로 표시한 후 테이블 설명을 참조하여 수정 내용을 확인하십시오.

```
ALTER TABLE emp
SET UNUSED (dept_id);

DESCRIBE emp
```

12. EMP 테이블에 있는 UNUSED 열을 모두 삭제한 후 테이블 설명을 참조하여 수정 내용을 확인하십시오.

```
ALTER TABLE emp
DROP UNUSED COLUMNS;

DESCRIBE emp
```

연습 10 해답

- EMP 테이블의 ID 열에 테이블 레벨의 PRIMARY KEY 제약 조건을 추가하십시오. 제약 조건을 생성할 때 이름을 지정해야 합니다. 제약 조건의 이름을 my_emp_id_pk로 지정하십시오.

```
ALTER TABLE emp  
ADD CONSTRAINT my_emp_id_pk PRIMARY KEY (id);
```

- ID 열을 사용하여 DEPT 테이블에 PRIMARY KEY 제약 조건을 생성하십시오. 제약 조건을 생성할 때 이름을 지정해야 합니다. 제약 조건의 이름을 my_dept_id_pk로 지정하십시오.

```
ALTER TABLE dept  
ADD CONSTRAINT my_dept_id_pk PRIMARY KEY(id);
```

- EMP 테이블에 DEPT_ID 열을 추가하십시오. 존재하지 않는 부서에 사원이 배정되지 않도록 외래 키 참조를 EMP 테이블에 추가하십시오. 제약 조건의 이름을 my_emp_dept_id_fk로 지정하십시오.

```
ALTER TABLE emp  
ADD (dept_id NUMBER(7));
```

```
ALTER TABLE emp  
ADD CONSTRAINT my_emp_dept_id_fk  
FOREIGN KEY (dept_id) REFERENCES dept(id);
```

- USER_CONSTRAINTS 뷰를 질의하여 제약 조건이 추가되었는지 확인하고 제약 조건의 유형 및 이름을 적어두십시오. 명령문 텍스트를 lab10_4.sql이라는 파일에 저장하십시오.

```
SELECT constraint_name, constraint_type  
FROM user_constraints  
WHERE table_name IN ('EMP', 'DEPT');
```

- USER_OBJECTS 데이터 딕셔너리 뷰에서 EMP 및 DEPT 테이블의 객체 이름 및 유형을 표시하십시오. 새 테이블 및 새 인덱스가 생성된 것을 볼 수 있습니다.

```
SELECT object_name, object_type  
FROM user_objects  
WHERE object_name LIKE 'EMP%'  
OR object_name LIKE 'DEPT%';
```

시간이 있을 때 다음 문제를 풀어보십시오.

- EMP 테이블을 수정하여 십진 자릿수 2, 소수점 이하 자릿수 2인 NUMBER 데이터 유형의 COMMISSION 열을 추가하십시오. 커미션 값이 0보다 크도록 커미션 열에 제약 조건을 추가하십시오.

```
ALTER TABLE EMP  
ADD commission NUMBER(2,2)  
CONSTRAINT my_emp_comm_ck CHECK (commission >= 0);
```

연습 11 해답

- EMPLOYEES 테이블에서 사원 번호, 사원 이름 및 부서 번호를 기반으로 하는 EMPLOYEES_VU라는 뷰를 생성하십시오. 사원 이름의 머리글을 EMPLOYEE로 변경하십시오.

```
CREATE OR REPLACE VIEW employees_vu AS
SELECT employee_id, last_name employee, department_id
FROM employees;
```

- EMPLOYEES_VU 뷰의 내용을 표시하십시오.

```
SELECT *
FROM employees_vu;
```

- USER_VIEWS 데이터 덕셔너리 뷰에서 뷰 이름 및 텍스트를 선택하십시오.

참고: 다른 뷰가 이미 존재합니다. EMP_DETAILS_VIEW가 스키마의 일부로 생성되었습니다.

참고: LONG 열의 내용을 더 많이 보려면 SQL*Plus 명령인 SET LONG n을 사용하십시오. 여기서 n은 보려는 LONG 열의 문자 수를 나타내는 값입니다.

```
SET LONG 600
SELECT view_name, text
FROM user_views;
```

- EMPLOYEES_VU 뷰를 사용하여 모든 사원의 이름 및 부서 번호를 표시하는 질의를 작성하십시오.

```
SELECT employee, department_id
FROM employees_vu;
```

- 부서 50의 모든 사원에 대한 사원 번호, 사원 이름 및 부서 번호를 포함하는 DEPT50이라는 뷰를 생성하고 뷰의 열 레이블을 EMPNO, EMPLOYEE 및 DEPTNO로 지정하십시오. 뷰를 통해 한 사원이 다른 부서에 중복 할당되지 않도록 하십시오.

```
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;
```

연습 11 해답(계속)

6. DEPT50 뷰의 구조와 내용을 표시하십시오.

```
DESCRIBE dept50
SELECT *
FROM dept50;
```

7. Matos를 부서 80에 다시 할당해 보십시오.

```
UPDATE dept50
SET deptno = 80
WHERE employee = 'Matos';
```

시간이 있을 때 다음 문제를 풀어보십시오.

8. 모든 사원의 이름, 부서 이름, 급여 및 급여 등급을 기반으로 하는 SALARY_VU라는 뷰를 생성하십시오. EMPLOYEES, DEPARTMENTS 및 JOB_GRADES 테이블을 사용하고 열 레이블을 각각 Employee, Department, Salary 및 Grade로 지정하십시오.

```
CREATE OR REPLACE VIEW salary_vu
AS
SELECT e.last_name "Employee",
       d.department_name "Department",
       e.salary "Salary",
       j.grade_level "Grades"
  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;
```

연습 12 해답

1. DDEPT 테이블의 기본 키 열에 사용할 시퀀스를 생성하십시오. 시퀀스 값은 200부터 시작하여 10씩 증가하며 최대 1000까지 가능합니다. 시퀀스 이름은 DEPT_ID_SEQ로 지정하십시오.

```
CREATE SEQUENCE dept_id_seq
START WITH 200
INCREMENT BY 10
MAXVALUE 1000;
```

2. 시퀀스 이름, 최대값, 증가분, 마지막 번호와 같은 시퀀스 정보를 표시하는 질의를 스크립트로 작성하여 lab12_2.sql이라는 이름을 지정한 후 스크립트의 명령문을 실행하십시오.

```
SELECT sequence_name, max_value, increment_by, last_number
FROM user_sequences;
```

3. DEPT 테이블에 두 행(row)을 삽입하는 스크립트를 작성하고 이름을 lab12_3.sql로 지정하십시오. ID 열에 대해서는 이전에 생성한 시퀀스를 사용하십시오. Education 및 Administration이라는 두 개의 부서를 추가하고 결과를 확인한 후 스크립트의 명령을 실행하십시오.

```
INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Education');
```

```
INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Administration');
```

4. EMP 테이블의 외래 키 열(DEPT_ID)에 대해 비고유 인덱스를 생성하십시오.

```
CREATE INDEX emp_dept_id_idx ON emp (dept_id);
```

5. 데이터 덕셔너리에 있는 EMP 테이블의 인덱스 및 고유성을 표시하십시오. 명령문을 lab12_5.sql이라는 스크립트에 저장하십시오.

```
SELECT index_name, table_name, uniqueness
FROM user_indexes
WHERE table_name = 'EMP';
```

연습 13 해답

- 사용자가 Oracle Server에 로그인하기 위해 필요한 권한은 무엇입니까? 시스템 권한과 객체 권한 중 어느 것입니까?

CREATE SESSION 시스템 권한

- 테이블을 생성할 때 필요한 권한은 무엇입니까?

CREATE TABLE 권한

- 생성한 테이블에 대한 권한을 다른 사용자에게 부여할 수 있는 사용자는 누구입니까?

생성자 및 생성자가 **WITH GRANT OPTION**을 사용하여 해당 권한을 부여한 사람

- DBA가 동일한 시스템 권한이 필요한 사용자를 여러 명 생성할 때 간편하게 작업할 수 있는 방법은 무엇입니까?

시스템 권한을 포함하는 틀을 생성하여 사용자에게 부여합니다.

- 암호를 변경할 때 사용하는 명령은 무엇입니까?

ALTER USER 문

- 자신이 생성한 DEPARTMENTS 테이블에 대한 액세스 권한을 다른 사용자에게 부여하고 해당 사용자의 DEPARTMENTS 테이블에 대한 질의 액세스 권한을 부여 받도록 하십시오.

팀 2에서 실행할 GRANT 문:

```
GRANT select  
ON   departments  
TO   <user1>;
```

팀 1에서 실행할 GRANT 문:

```
GRANT select  
ON   departments  
TO   <user2>;
```

WHERE user1 is the name of team 1 and user2 is the name of team 2.

- 자신의 DEPARTMENTS 테이블에 있는 모든 행(row)을 질의하십시오.

```
SELECT *  
FROM   departments;
```

연습 13 해답(계속)

8. DEPARTMENTS 테이블에 새 행(row)을 추가하십시오. 팀 1은 Education을 부서 번호 500으로, 팀 2는 Human Resources를 부서 번호 510으로 추가한 후 서로 상대 팀의 테이블을 질의하십시오.

팀 1에서 실행할 INSERT 문:

```
INSERT INTO departments(department_id, department_name)
VALUES (500, 'Education');
COMMIT;
```

팀 2에서 실행할 INSERT 문:

```
INSERT INTO departments(department_id, department_name)
VALUES (510, 'Administration');
COMMIT;
```

9. 상대 팀의 DEPARTMENTS 테이블에 대한 동의어를 생성하십시오.

팀 1에서 team2라는 동의어를 생성하는 명령문:

```
CREATE SYNONYM      team2
FOR <user2>.DEPARTMENTS;
```

팀 2에서 team1이라는 동의어를 생성하는 명령문:

```
CREATE SYNONYM      team1
FOR <user1>. DEPARTMENTS;
```

10. 동의어를 사용하여 상대 팀의 DEPARTMENTS 테이블에 포함된 모든 행(row)을 질의하십시오.

팀 1에서 실행할 SELECT 문:

```
SELECT *
FROM      team2;
```

팀 2에서 실행할 SELECT 문:

```
SELECT *
FROM      team1;
```

연습 13 해답(계속)

11. 소유한 테이블에 대한 정보를 표시하도록 USER_TABLES 데이터 딕셔너리를 질의하십시오.

```
SELECT table_name  
FROM user_tables;
```

12. 액세스할 수 있는 모든 테이블에 대한 정보를 표시하도록 ALL_TABLES 데이터 딕셔너리 뷰를 질의하십시오(현재 소유한 테이블 제외).

```
SELECT table_name, owner  
FROM all_tables  
WHERE owner <> <your account>;
```

13. 상대 팀에게 부여했던 자신의 테이블에 대한 SELECT 권한을 취소하십시오.

팀 1에서 권한을 취소하는 명령문:

```
REVOKE select  
ON departments  
FROM user2;
```

팀 2에서 권한을 취소하는 명령문:

```
REVOKE select  
ON departments  
FROM user1;
```

14. 8 단계에서 DEPARTMENTS 테이블에 삽입한 행(row)을 제거하고 변경 내용을 저장하십시오.

팀 1에서 실행할 INSERT 문:

```
DELETE FROM departments  
WHERE department_id = 500;  
COMMIT;
```

팀 2에서 실행할 INSERT 문:

```
DELETE FROM departments  
WHERE department_id = 510;  
COMMIT;
```

연습 14 해답

- 다음 테이블 인스턴스 차트를 기반으로 테이블을 생성하십시오. 적합한 데이터 유형을 선택하고 무결성 제약 조건을 추가해야 합니다.
 - 테이블 이름: MEMBER

열 이름	MEMBER_ID	LAST_NAME	FIRST_NAME	ADDRESS	CITY	PHONE	JOIN_DATE
키 유형	PK						
널/고유	NN,U	NN					NN
기본값							System Date
데이터 유형	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
길이	10	25	25	100	30	15	

```
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);
```

연습 14 해답(계속)

b. 테이블 이름: TITLE

열 이름	TITLE_ID	TITLE	DESCRIPTION	RATING	CATEGORY	RELEASE_DATE
키 유형	PK					
필/ 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	

```

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);

```

연습 14 해답(계속)

c. 테이블 이름: TITLE_COPY

열 이름	COPY_ID	TITLE_ID	STATUS
키 유형	PK	PK,FK	
널/ 고유	NN,U	NN,U	NN
확인			AVAILABLE, DESTROYED, RENTED, RESERVED
FK 참조 테이블		TITLE	
FK 참조 열		TITLE_ID	
데이터 유형	NUMBER	NUMBER	VARCHAR2
길이	10	10	15

```

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 14 해답(계속)

4. 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
기본값	System Date				System Date + 2 days	
FK 참조 테이블		MEMBER	TITLE_COPY			TITLE_COPY
FK 참조 열		MEMBER_ID	COPY_ID			TITLE_ID
데이터 유형	DATE	NUMBER	NUMBER	DATE	DATE	NUMBER
길이		10	10			10

```

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));

```

연습 14 해답(계속)

e. 테이블 이름: RESERVATION

열 이름	RES_DATE	MEMBER_ID	TITLE_ID
키 유형	PK	PK,FK1	PK,FK2
널/ 고유	NN,U	NN,U	NN
FK 참조 테이블		MEMBER	TITLE
FK 참조 열		MEMBER_ID	TITLE_ID
데이터 유형	DATE	NUMBER	NUMBER
길이		10	10

```
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));
```

연습 14 해답(계속)

2. 데이터 딕셔너리를 참조하여 테이블 및 제약 조건이 제대로 생성되었는지 확인하십시오.

```
SELECT    table_name
FROM      user_tables
WHERE     table_name IN ('MEMBER', 'TITLE', 'TITLE_COPY',
                        'RENTAL', 'RESERVATION');
```

```
SELECT    constraint_name, constraint_type, table_name
FROM      user_constraints
WHERE     table_name IN ('MEMBER', 'TITLE', 'TITLE_COPY',
                        'RENTAL', 'RESERVATION');
```

3. MEMBER 테이블 및 TITLE 테이블의 각 행(row)을 고유하게 식별하는 시퀀스를 생성하십시오.

a. MEMBER 테이블의 회원 번호는 101부터 시작하고 값이 캐시되지 않도록 하십시오.

시퀀스 이름은 MEMBER_ID_SEQ로 지정하십시오.

```
CREATE SEQUENCE member_id_seq
START WITH 101
NOCACHE;
```

b. TITLE 테이블의 제목 번호는 92부터 시작하고 캐시되지 않도록 하십시오. 시퀀스

이름은 TITLE_ID_SEQ로 지정하십시오.

```
CREATE SEQUENCE title_id_seq
START WITH 92
NOCACHE;
```

c. 데이터 딕셔너리에서 시퀀스의 존재를 확인하십시오.

```
SELECT    sequence_name, increment_by, last_number
FROM      user_sequences
WHERE     sequence_name IN ('MEMBER_ID_SEQ', 'TITLE_ID_SEQ');
```

연습 14 해답(계속)

4. 테이블에 데이터를 추가하고 추가할 각 데이터 집합에 대한 스크립트를 작성하십시오.
- a. TITLE 테이블에 영화 제목을 추가하십시오. 영화 정보를 입력할 스크립트를 작성하여 lab14_4a.sql이라는 이름으로 저장하십시오. 시퀀스를 사용하여 각 제목을 고유하게 식별하고 출시일을 DD-MON-YYYY 형식으로 입력하십시오. 문자 필드에 사용된 작은 따옴표는 특별하게 처리된다는 점을 기억하십시오. 추가한 항목을 확인하십시오.

```
SET ECHO OFF
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'))
/
...
COMMIT
/
SET ECHO ON

SELECT title
FROM title;
```

연습 14 해답(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

연습 14 해답(계속)

- b. MEMBER 테이블에 다음 데이터를 추가하십시오.

INSERT 문을 lab14_4b.sql이라는 이름의 스크립트에 추가하고 스크립트의 명령을 실행하십시오. 시퀀스를 사용하여 회원 번호를 추가해야 합니다.

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

```
SET ECHO OFF
SET VERIFY OFF
INSERT INTO member(member_id, first_name, last_name, address,
       city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, '&first_name', '&last_name',
       '&address', '&city', '&phone', TO_DATE('&join_date',
       'DD-MM-YYYY'));
COMMIT;
SET VERIFY ON
SET ECHO ON
```

연습 14 해답(계속)

c. TITLE_COPY 테이블에 다음 영화 복사본을 추가하십시오.

참고: 이 연습 문제에서는 TITLE_ID 번호를 사용할 수 있습니다.

Title	Copy_Id	Status
Willie and Christmas Too	1	AVAILABLE
Alien Again	1	AVAILABLE
	2	RENTED
The Glob	1	AVAILABLE
My Day Off	1	AVAILABLE
	2	AVAILABLE
	3	RENTED
Miracles on Ice	1	AVAILABLE
Soda Gang	1	AVAILABLE

```
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 14 해답(계속)

d. RENTAL 테이블에 다음 대여 항목을 추가하십시오.

참고: 제목 번호는 시퀀스 번호에 따라 달라집니다.

Title_Id	Copy_Id	Member_Id	Book_date	Exp_Ret_Date	Act_Ret_Date
92	1	101	3 days ago	1 day ago	2 days 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	2 days ago

```
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;
```

연습 14 해답(계속)

5. 영화 제목, 각 복사본의 대여 가능 여부, 반환 예정일(대여된 경우)을 표시하는 TITLE_AVAIL이라는 이름의 뷰를 생성하고 뷰의 모든 행(row)을 질의한 후 결과를 제목별로 정렬하십시오.

```
CREATE VIEW title_avail AS
SELECT    t.title, c.copy_id, c.status, r.exp_ret_date
FROM      title t, title_copy c, rental r
WHERE     t.title_id = c.title_id
AND       c.copy_id = r.copy_id(+)
AND       c.title_id = r.title_id(+);

SELECT    *
FROM      title_avail
ORDER BY title, copy_id;
```

6. 테이블의 데이터를 변경하십시오.

a. 새 제목을 추가하십시오. 이 영화의 제목은 “Interstellar Wars”이고 등급은 PG이며 SCIFI(공상 과학 영화)로 분류됩니다. 출시일은 1977년 7월 7일이며 영화에 대한 정보는 “Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?”입니다. 두 개의 복사본에 대해 TITLE_COPY 레코드를 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. 예약 항목 두 개를 입력하십시오. 하나는 “Interstellar Wars”를 빌리려는 Carmen Velasquez에 대한 항목이고 다른 하나는 “Soda Gang”을 빌리려는 Mark Quick-to-See에 대한 항목입니다.

```
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);
```

연습 14 해답(계속)

- c. 고객 Carmen Velasquez가 영화 “Interstellar Wars”의 복사본 1을 빌렸으므로 영화 예약 항목에서 Carmen Velasquez를 제거하고 대여 정보를 기록하십시오. 반환 예정일에는 기본값을 사용할 수 있도록 하고, 작성한 뷰를 사용하여 대여 정보가 기록되었는지 확인하십시오.

```
INSERT INTO rental(title_id, copy_id, member_id)
VALUES (98,1,101);
UPDATE title_copy
SET status= 'RENTED'
WHERE title_id = 98
AND copy_id = 1;
DELETE
FROM reservation
WHERE member_id = 101;
SELECT *
FROM title_avail
ORDER BY title, copy_id;
```

7. 테이블 중 하나를 수정하십시오.

- a. TITLE 테이블에 PRICE 열을 추가하여 비디오 구입 가격을 기록하십시오. 열은 소수 둘째 자리까지 포함하여 총 8자리로 표시하십시오. 수정 내용을 확인하십시오.

```
ALTER TABLE title
ADD (price NUMBER(8,2));
DESCRIBE title
```

연습 14 해답(계속)

- b. 다음 목록에 따라 각 비디오의 가격을 개선하는 UPDATE 문을 포함하는 lab14_7b.sql이라는 스크립트를 생성하고 스크립트의 명령을 실행하십시오.
참고: 이 연습 문제에서는 TITLE_ID 번호를 사용할 수 있습니다.

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

```
SET ECHO OFF
SET VERIFY OFF
DEFINE price=
DEFINE title_id=
UPDATE title
SET price = &price
WHERE title_id = &title_id;
SET VERIFY OFF
SET ECHO OFF
```

- c. 이후에는 모든 제목에 가격이 포함되도록 하십시오. 제약 조건을 확인하십시오.

```
ALTER TABLE title
MODIFY (price CONSTRAINT title_price_nn NOT NULL);
SELECT constraint_name, constraint_type,
       search_condition
  FROM user_constraints
 WHERE table_name = 'TITLE';
```

연습 14 해답(계속)

8. Customer History Report라는 제목의 보고서를 작성하십시오. 이 보고서에는 고객 이름, 대여한 영화, 대여 날짜 및 대여 기간 등과 같은 각 고객의 비디오 대여 기록이 포함됩니다. 또한 보고 기간 동안 모든 고객에게 대여된 총 대여 수를 계산합니다. 이 보고서를 생성하는 명령을 lab14_8.sql이라는 스크립트 파일로 저장하십시오.

```
SET ECHO OFF
SET VERIFY OFF
TTITLE 'Customer History Report'
BREAK ON member SKIP 1 ON REPORT
SELECT      m.first_name||' '||m.last_name MEMBER, t.title,
            r.book_date, r.act_ret_date - r.book_date
            DURATION
FROM        member m, title t, rental r
WHERE       r.member_id = m.member_id
AND         r.title_id = t.title_id
ORDER BY member;

CLEAR BREAK
TTITLE OFF
SET VERIFY ON
SET ECHO ON
```

연습 15 해답

- SET 연산자를 사용하여 업무 ID ST_CLERK을 포함하지 않는 부서의 ID를 나열하십시오.

```
SELECT department_id  
  FROM departments  
 MINUS  
SELECT department_id  
  FROM employees  
 WHERE job_id = 'ST_CLERK';
```

- SET 연산자를 사용하여 해당 지역에 부서가 없는 지역 ID와 지역 이름을 표시하십시오.

```
SELECT country_id, country_name  
  FROM countries  
 MINUS  
SELECT l.country_id, c.country_name  
  FROM locations l,    countries c  
 WHERE l.country_id = c.country_id;
```

- 부서 ID가 10, 50 및 20인 부서의 업무 목록을 해당 순서대로 나열하고, SET 연산자를 사용하여 업무 ID와 부서 ID를 표시하십시오.

```
COLUMN dummy NOPRINT  
SELECT job_id, department_id, 'x' dummy  
  FROM employees  
 WHERE department_id = 10  
UNION  
SELECT job_id, department_id, 'y'  
  FROM employees  
 WHERE department_id = 50  
UNION  
SELECT job_id, department_id, 'z'  
  FROM employees  
 WHERE department_id = 20  
 ORDER BY 3;  
COLUMN dummy PRINT
```

연습 15 해답(계속)

4. 입사 이후 현재 업무와 같은 업무를 담당한 적이 있는 사원의 사원 ID와 업무 ID를 나열하십시오.

```
SELECT      employee_id, job_id
FROM        employees
INTERSECT
SELECT      employee_id, job_id
FROM        job_history;
```

5. 다음을 나열하는 혼합 질의를 작성하십시오.

- 소속된 부서에 상관 없이 EMPLOYEES 테이블에 있는 모든 사원의 이름과 부서 ID
- 소속된 사원에 상관 없이 DEPARTMENTS 테이블에 있는 모든 부서의 부서 ID와 부서 이름

```
SELECT last_name,department_id,TO_CHAR(null)
FROM   employees
UNION
SELECT TO_CHAR(null),department_id,department_name
FROM   departments;
```

연습 16 해답

1. NLS_DATE_FORMAT을 DD-MON-YYYY HH24:MI:SS로 설정하여 세션을 변경하십시오.

```
ALTER SESSION SET NLS_DATE_FORMAT =
'DD-MON-YYYY HH24:MI:SS';
```

2. a. 다음 시간대에 대해 시간대 오프셋(TZ_OFFSET)을 표시하는 질의를 작성하십시오.

US/Pacific-New

```
SELECT TZ_OFFSET ('US/Pacific-New') from dual;
```

Singapore

```
SELECT TZ_OFFSET ('Singapore') from dual;
```

Egypt

```
SELECT TZ_OFFSET ('Egypt') from dual;
```

- b. TIME_ZONE 파라미터 값을 US/Pacific-New의 시간대 오프셋으로 설정하여 세션을 변경하십시오.

```
ALTER SESSION SET TIME_ZONE = '-7:00';
```

- c. 이 세션에 대한 CURRENT_DATE, CURRENT_TIMESTAMP 및 LOCALTIMESTAMP를 표시하십시오.

참고: 명령을 실행하는 날짜에 따라 결과가 다를 수 있습니다.

```
SELECT CURRENT_DATE, CURRENT_TIMESTAMP, LOCALTIMESTAMP
FROM DUAL;
```

- d. TIME_ZONE 파라미터 값을 Singapore의 시간대 오프셋으로 설정하여 세션을 변경하십시오.

```
ALTER SESSION SET TIME_ZONE = '+8:00';
```

- e. 이 세션에 대한 CURRENT_DATE, CURRENT_TIMESTAMP 및 LOCALTIMESTAMP를 표시하십시오.

참고: 명령을 실행하는 날짜에 따라 결과가 다를 수 있습니다.

```
SELECT CURRENT_DATE, CURRENT_TIMESTAMP, LOCALTIMESTAMP
FROM DUAL;
```

3. DBTIMEZONE 및 SESSIONTIMEZONE을 표시하는 질의를 작성하십시오.

```
SELECT DBTIMEZONE, SESSIONTIMEZONE
FROM DUAL;
```

연습 16 해답(계속)

4. 부서 ID가 80인 사원에 대해 EMPLOYEES 테이블의 HIRE_DATE 열에서 YEAR를 추출하는 질의를 작성하십시오.

```
SELECT last_name, EXTRACT (YEAR FROM HIRE_DATE)
FROM employees
WHERE department_id = 80;
```

5. NLS_DATE_FORMAT을 DD-MON-YYYY로 설정하여 세션을 변경하십시오.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';
```

연습 17 해답

1. 관리자 ID가 120보다 작은 관리자를 둔 사원에 대해 다음을 표시하는 질의를 작성하십시오.

- 관리자 ID
- 같은 관리자를 둔 사원에 대한 업무 ID별 총 급여와 해당 업무 ID
- 해당 관리자의 총 급여
- 업무 ID에 상관 없는 해당 관리자의 총 급여

```
SELECT manager_id, job_id, sum(salary)
FROM   employees
WHERE manager_id < 120
GROUP BY ROLLUP(manager_id, job_id);
```

2. 1번 질문의 출력을 살펴본 다음, GROUPING 함수를 사용하여 GROUP BY 표현식의 해당 열에 있는 NULL 값이 ROLLUP 연산으로 인한 것인지 확인하는 질의를 작성하십시오.

```
SELECT manager_id MGR ,job_id JOB,
sum(salary), GROUPING(manager_id), GROUPING(job_id)
FROM   employees
WHERE manager_id < 120
GROUP BY ROLLUP(manager_id, job_id);
```

3. 관리자 ID가 120보다 작은 관리자를 둔 사원에 대해 다음을 표시하는 질의를 작성하십시오.

- 관리자 ID
- 같은 관리자를 둔 사원에 대한 업무별 총 급여와 해당 업무
- 해당 관리자의 총 급여
- 관리자에 상관 없이, 업무별 총 급여를 표시하는 교차 도표화 값
- 업무에 상관 없는 전체 총 급여

```
SELECT manager_id, job_id, sum(salary)
FROM   employees
WHERE manager_id < 120
GROUP BY CUBE(manager_id, job_id);
```

연습 17 해답(계속)

4. 3번 질문의 출력을 살펴본 다음, GROUPING 함수를 사용하여 GROUP BY 표현식의 해당 열에 있는 NULL 값이 CUBE 연산으로 인한 것인지 확인하는 질의를 작성하십시오.

```
SELECT manager_id MGR ,job_id JOB,
       sum(salary),GROUPING(manager_id),GROUPING(job_id)
  FROM   employees
 WHERE manager_id < 120
 GROUP BY CUBE(manager_id,job_id);
```

5. GROUPING SETS를 사용하여 다음 그룹화를 표시하는 질의를 작성하십시오.

- department_id, manager_id, job_id
- department_id, job_id
- Manager_id, job_id

질의에서 해당 그룹별 급여의 합을 계산해야 합니다.

```
SELECT department_id, manager_id, job_id,  SUM(salary)
  FROM employees
 GROUP BY
 GROUPING SETS ((department_id, manager_id, job_id),
 (department_id, job_id),(manager_id,job_id));
```

연습 18 해답

1. 커미션을 받는 사원과 부서 번호 및 급여가 일치하는 사원의 이름, 부서 번호 및 급여를 표시하는 질의를 작성하십시오.

```
SELECT last_name, department_id, salary
  FROM employees
 WHERE (salary, department_id) IN
       (SELECT salary, department_id
        FROM employees
       WHERE commission_pct IS NOT NULL);
```

2. 위치 ID가 1700인 사원과 급여 및 커미션이 일치하는 사원의 이름, 부서 이름 및 급여를 표시하십시오.

```
SELECT e.last_name, d.department_name, e.salary
  FROM employees e, departments d
 WHERE e.department_id = d.department_id
   AND (salary, NVL(commission_pct,0)) IN
        (SELECT salary, NVL(commission_pct,0)
         FROM employees e, departments d
        WHERE e.department_id = d.department_id
          AND d.location_id = 1700);
```

3. Kochhar와 동일한 급여 및 커미션을 받는 모든 사원의 이름, 입사일 및 급여를 표시하는 질의를 작성하십시오.

참고: 결과 집합에 Kochhar는 표시하지 마십시오.

```
SELECT last_name, hire_date, salary
  FROM employees
 WHERE (salary, NVL(commission_pct,0)) IN
        (SELECT salary, NVL(commission_pct,0)
         FROM employees
        WHERE last_name = 'Kochhar')
AND last_name != 'Kochhar';
```

4. 모든 영업 관리자(JOB_ID = 'SA_MAN')보다 급여를 많이 받는 사원을 표시하는 질의를 작성하고 결과를 최고 급여에서 최저 급여의 순으로 정렬하십시오.

```
SELECT last_name, job_id, salary
  FROM employees
 WHERE salary > ALL
        (SELECT salary
         FROM employees
        WHERE job_id = 'SA_MAN')
 ORDER BY salary DESC;
```

연습 18 해답(계속)

5. T로 시작하는 도시에 사는 사원의 세부 정보인 사원 ID, 이름 및 부서 ID를 표시하십시오.

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                           FROM departments
                           WHERE location_id IN
                                 (SELECT location_id
                                  FROM locations
                                  WHERE city LIKE 'T%'));
```

6. 소속 부서의 평균 급여보다 많은 급여를 받는 사원을 모두 찾는 질의를 작성하십시오. 이름, 급여, 부서 ID 및 소속 부서의 평균 급여를 표시하되, 평균 급여에 따라 정렬하십시오. 예제 출력과 같이 질의에 의해 검색되는 열에 별칭을 사용하십시오.

```
SELECT e.last_name ename, e.salary salary,
       e.department_id deptno, AVG(a.salary) dept_avg
  FROM   employees e, employees a
 WHERE  e.department_id = a.department_id
 AND    e.salary > (SELECT AVG(salary)
                      FROM   employees
                      WHERE  department_id = e.department_id )
 GROUP BY e.last_name, e.salary, e.department_id
 ORDER BY AVG(a.salary);
```

7. 관리자가 아닌 사원을 모두 찾으십시오.

- a. 먼저 NOT EXISTS 연산자를 사용하여 이 작업을 수행하십시오.

```
SELECT outer.last_name
  FROM   employees outer
 WHERE  NOT EXISTS (SELECT 'X'
                      FROM employees inner
                      WHERE inner.manager_id =
                            outer.employee_id);
```

연습 18 해답(계속)

- b. NOT IN 연산자를 사용하여 이 작업을 수행할 수 있습니까? 수행할 수 있는 경우 방법을 설명하고 수행할 수 없는 경우 이유를 설명하십시오.

```
SELECT outer.last_name
  FROM employees outer
 WHERE outer.employee_id
    NOT IN (SELECT inner.manager_id
            FROM employees inner);
```

이 방법은 좋은 방법이 아닙니다. 서브 쿼리에서 NULL 값을 선택하므로 전체 질의에서 아무런 행도 반환하지 않습니다. 그 이유는 NULL 값을 비교하는 조건에서는 결과가 항상 NULL이기 때문입니다. 값 집합에 NULL 값이 포함될 가능성이 있을 경우에는 NOT EXISTS 대신 NOT IN을 사용하지 않도록 하십시오.

8. 소속 부서의 평균 급여보다 적은 급여를 받는 사원의 이름을 표시하는 질의를 작성하십시오.

```
SELECT last_name
  FROM employees outer
 WHERE outer.salary < (SELECT AVG(inner.salary)
                        FROM employees inner
                       WHERE inner.department_id
                         = outer.department_id);
```

9. 소속 부서에 입사일이 늦지만 더 많은 급여를 받는 동료가 있는 사원의 이름을 표시하는 질의를 작성하십시오.

```
SELECT last_name
  FROM employees outer
 WHERE EXISTS (SELECT 'X'
                  FROM employees inner
                 WHERE inner.department_id =
                      outer.department_id
                   AND inner.hire_date > outer.hire_date
                   AND inner.salary > outer.salary);
```

10. 모든 사원의 사원 ID, 이름 및 부서 이름을 표시하는 질의를 작성하십시오.

참고: 스칼라 서브 쿼리를 사용하여 SELECT 문에서 부서 이름을 검색하십시오.

```
SELECT employee_id, last_name,
       (SELECT department_name
        FROM departments d
       WHERE e.department_id =
             d.department_id) department
  FROM employees e
 ORDER BY department;
```

연습 18 해답(계속)

11. 총 급여가 전체 회사 총 급여의 1/8보다 많은 부서의 이름을 표시하는 질의를 작성하십시오. 이 질의를 WITH 절을 사용하여 작성하되, SUMMARY라는 이름을 부여하십시오.

```
WITH
summary AS (
    SELECT d.department_name, SUM(e.salary) AS dept_total
    FROM employees e, departments d
    WHERE e.department_id = d.department_id
    GROUP BY d.department_name)
SELECT department_name, dept_total
FROM summary
WHERE dept_total > (
    SELECT SUM(dept_total) * 1/8
    FROM summary )
ORDER BY dept_total DESC;
```

연습 19 해답

1. 다음 출력을 보고 결과 출력이 계층 질의의 결과인지 여부를 판단하고 그 이유를 설명 하십시오.

제출물 1: 이것은 계층 질의가 아닙니다. 이 보고서는 단순히 SALARY에 대해 내림차순으로 정렬한 것입니다.

제출물 2: 이것은 계층 질의가 아닙니다. 왜냐하면 두 개의 테이블이 관련되어 있기 때문입니다.

제출물 3: 이것은 EMPLOYEES 테이블에서 관리 보고선을 나타내는 트리 구조를 표시하므로 계층 질의가 맞습니다.

2. Mourgos의 부서에 대한 조직 차트를 보여주는 보고서를 작성하십시오. 이름, 급여 및 부서 ID를 출력하십시오.

```
SELECT last_name, salary, department_id
FROM employees
START WITH last_name = 'Mourgos'
CONNECT BY PRIOR employee_id = manager_id;
```

3. Lorentz의 관리자 계층을 보여주는 보고서를 작성하십시오. 직속 관리자를 먼저 표시 하십시오.

```
SELECT last_name
FROM employees
WHERE last_name != 'Lorentz'
START WITH last_name = 'Lorentz'
CONNECT BY PRIOR manager_id = employee_id;
```

4. LAST_NAME이 Kochhar인 사원부터 시작하여 관리 계층을 보여주는 들여쓰기된 보고서를 작성하십시오. 사원의 이름, 관리자 ID 및 부서 ID를 출력하십시오. 예제 출력처럼 열 이름에 별칭을 사용하십시오.

```
COLUMN name FORMAT A20
SELECT LPAD(last_name, LENGTH(last_name)+(LEVEL*2)-2, '_')
      ,name,manager_id mgr, department_id deptno
FROM employees
START WITH last_name = 'Kochhar'
CONNECT BY PRIOR employee_id = manager_id
/
COLUMN name CLEAR
```

연습 19 해답(계속)

시간이 있을 때 다음 문제를 풀어보십시오.

5. 관리 계층을 보여주는 회사 조직 차트를 작성하십시오. 최상위 레벨부터 시작하되, 업무 ID가 IT_PROG인 사원, De Haan 및 De Haan이 관리하는 사원은 모두 제외시키십시오.

```
SELECT last_name,employee_id, manager_id
  FROM employees
 WHERE job_id != 'IT_PROG'
 START WITH manager_id IS NULL
 CONNECT BY PRIOR employee_id = manager_id
 AND last_name != 'De Haan';
```

연습 20 해답

1. lab 폴더에서 cre_sal_history.sql 스크립트를 실행하여 SAL_HISTORY 테이블을 생성하십시오.

```
@ \lab\cre_sal_history.sql
```

2. SAL_HISTORY 테이블의 구조를 표시하십시오.

```
DESC sal_history
```

3. lab 폴더에서 cre_mgr_history.sql 스크립트를 실행하여 MGR_HISTORY 테이블을 생성하십시오.

```
@ \lab\cre_mgr_history.sql
```

4. MGR_HISTORY 테이블의 구조를 표시하십시오.

```
DESC mgr_history
```

5. lab 폴더에서 cre_special_sal.sql 스크립트를 실행하여 SPECIAL_SAL 테이블을 생성하십시오.

```
@ \lab\cre_special_sal.sql
```

6. SPECIAL_SAL 테이블의 구조를 표시하십시오.

```
DESC special_sal
```

7. a. 다음을 수행하는 질의를 작성하십시오.

- EMPLOYEES 테이블에서 사원 ID가 125보다 작은 사원의 사원 ID, 입사일, 급여 및 관리자 ID를 검색하십시오.
- 급여가 \$20,000보다 많으면 SPECIAL_SAL 테이블에 사원 ID 및 급여를 삽입하십시오.
- SAL_HISTORY 테이블에 사원 ID, 입사일 및 급여를 삽입하십시오.
- MGR_HISTORY 테이블에 사원 ID, 관리자 ID 및 급여를 삽입하십시오.

```
INSERT ALL
WHEN SAL > 20000 THEN
  INTO special_sal VALUES (EMPID, SAL)
ELSE
  INTO sal_history VALUES (EMPID, HIREDATE, SAL)
  INTO mgr_history VALUES (EMPID, MGR, SAL)
SELECT employee_id EMPID, hire_date HIREDATE,
       salary SAL, manager_id MGR
  FROM employees
 WHERE employee_id < 125;
```

연습 20 해답(계속)

- b. SPECIAL_SAL 테이블의 레코드를 표시하십시오.

```
SELECT * FROM special_sal;
```

- c. SAL_HISTORY 테이블의 레코드를 표시하십시오.

```
SELECT * FROM sal_history;
```

- d. MGR_HISTORY 테이블의 레코드를 표시하십시오.

```
SELECT * FROM mgr_history;
```

8. a. lab 폴더에서 cre_sales_source_data.sql 스크립트를 실행하여 SALES_SOURCE_DATA 테이블을 생성하십시오.

```
@ \lab\cre_sales_source_data.sql
```

- b. lab 폴더에서 ins_sales_source_data.sql 스크립트를 실행하여 SALES_SOURCE_DATA 테이블에 레코드를 삽입하십시오.

```
@ \lab\ins_sales_source_data.sql
```

- c. SALES_SOURCE_DATA 테이블의 구조를 표시하십시오.

```
DESC sales_source_data
```

- d. SALES_SOURCE_DATA 테이블의 레코드를 표시하십시오.

```
SELECT * FROM SALES_SOURCE_DATA;
```

- e. lab 폴더에서 cre_sales_info.sql 스크립트를 실행하여 SALES_INFO 테이블을 생성하십시오.

```
@ \lab\cre_sales_info.sql
```

- f. SALES_INFO 테이블의 구조를 표시하십시오.

```
DESC sales_info
```

- g. 다음을 수행하는 질의를 작성하십시오.

- SALES_SOURCE_DATA 테이블에서 사원 ID, 주 ID, 월요일 판매 실적, 화요일 판매 실적, 수요일 판매 실적, 목요일 판매 실적 및 금요일 판매 실적을 검색하십시오.
- SALES_SOURCE_DATA 테이블에서 검색된 각 레코드를 SALES_INFO 테이블의 여러 레코드로 변환하십시오.

힌트: 꾸밈 INSERT 문을 사용하십시오.

연습 20 해답(계속)

```
INSERT ALL
INTO sales_info VALUES (employee_id, week_id, sales_MON)
INTO sales_info VALUES (employee_id, week_id, sales_TUE)
INTO sales_info VALUES (employee_id, week_id, sales_WED)
INTO sales_info VALUES (employee_id, week_id, sales_THUR)
INTO sales_info VALUES (employee_id, week_id, sales_FRI)
SELECT employee_id, week_id, sales_MON, sales_TUE,
sales_WED, sales_THUR, sales_FRI FROM sales_source_data;
```

h. SALES_INFO 테이블의 레코드를 표시하십시오.

```
SELECT * FROM sales_info;
```

9. a. 다음 테이블 인스턴스 차트에 따라 DEPT_NAMED_INDEX 테이블을 생성하십시오.
PRIMARY KEY 열에 대한 인덱스의 이름을 DEPT_PK_IDX로 지정하십시오.

열 이름	Deptno	Dname
기본 키	예	
데이터 유형	Number	VARCHAR2
길이	4	30

```
CREATE TABLE DEPT_NAMED_INDEX
(deptno NUMBER(4)
PRIMARY KEY USING INDEX
(CREATE INDEX dept_pk_idx ON
DEPT_NAMED_INDEX(deptno)),
dname VARCHAR2(30));
```

b. USER_INDEXES 테이블을 질의하여 DEPT_NAMED_INDEX 테이블의 INDEX_NAME
을 표시하십시오.

```
SELECT INDEX_NAME, TABLE_NAME
FROM USER_INDEXES
WHERE TABLE_NAME = 'DEPT_NAMED_INDEX';
```

연습 D 해답

1. 테이블의 데이터를 기술하고 선택하는 스크립트를 작성하십시오. SELECT 목록에서 CHR(10) 과 함께 연결 문자(||)를 사용하여 보고서에서 줄 바꿈이 되도록 하십시오. 스크립트의 출력을 my_file1.sql에 저장하십시오. 파일을 저장하려면 Output 옵션으로 File을 선택하고 코드를 실행합니다. 파일을 저장할 때는 .sql 확장자를 사용해야 한다는 점에 유의하십시오. my_file1.sql을 실행하려면 해당 스크립트를 찾아 로드한 다음 실행합니다.

```
SET PAGESIZE 0

SELECT  'DESC ' || table_name || CHR(10) ||
        'SELECT * FROM ' || table_name || ';'
FROM    user_tables
/
SET PAGESIZE 24
SET LINESIZE 100
```

2. SQL을 사용하여 사용자 권한을 취소하는 SQL 문을 생성하십시오. 데이터 딕셔너리 뷰 USER_TAB_PRIVS_MADE 및 USER_COL_PRIVS_MADE를 사용하십시오.
 - a. \Lab\privs.sql 스크립트를 실행하여 SYSTEM 사용자에게 권한을 부여하십시오.
 - b. 데이터 딕셔너리 뷰를 질의하여 권한을 확인하십시오. 예제 출력에서 GRANTOR 열의 데이터는 GRANTOR에 따라 달라질 수 있습니다. 또한 잘린 마지막 행(row)은 GRANTABLE 열입니다.

```
COLUMN      grantee  FORMAT A10
COLUMN      table_name  FORMAT A10
COLUMN      column_name  FORMAT A10
COLUMN      grantor  FORMAT A10
COLUMN      privilege  FORMAT A10
SELECT      *
FROM        user_tab_privs_made
WHERE       grantee = 'SYSTEM';

SELECT      *
FROM        user_col_privs_made
WHERE       grantee = 'SYSTEM';
```

연습 D 해답(계속)

- c. 권한을 취소하는 스크립트를 작성하십시오. 스크립트의 출력을 my_file2.sql에 저장하십시오. 파일을 저장하려면 Output 옵션으로 File을 선택하고 코드를 실행합니다. .sql 확장자를 사용하여 파일을 저장해야 한다는 점에 유의하십시오. my_file2.sql을 실행하려면 해당 스크립트를 찾아 로드한 다음 실행합니다.

```
SET VERIFY OFF
SET PAGESIZE 0

SELECT      'REVOKE ' || privilege || ' ON ' ||
table_name || ' FROM system;'
FROM    user_tab_privs_made
WHERE   grantee = 'SYSTEM'
/
SELECT      DISTINCT   'REVOKE ' || privilege || ' ON ' ||
table_name || ' FROM system;'
FROM    user_col_privs_made
WHERE   grantee = 'SYSTEM'
/

SET VERIFY ON
SET PAGESIZE 24
```

B

테이블 설명
및 데이터

COUNTRIES 테이블

```
DESCRIBE countries
```

Name	Null?	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

```
SELECT * FROM countries;
```

CO	COUNTRY_NAME	REGION_ID
CA	Canada	2
DE	Germany	1
UK	United Kingdom	1
US	United States of America	2

DEPARTMENTS 테이블

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
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

EMPLOYEES 테이블

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
100	Steven	King	SKING	515.123.4567	17-JUN-87
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98
144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00
174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-96
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-98
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94

20 rows selected.

EMPLOYEES 테이블(계속)

JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
AD_PRES	24000			90
AD_VP	17000		100	90
AD_VP	17000		100	90
IT_PROG	9000		102	60
IT_PROG	6000		103	60
IT_PROG	4200		103	60
ST_MAN	5800		100	50
ST_CLERK	3500		124	50
ST_CLERK	3100		124	50
ST_CLERK	2600		124	50
ST_CLERK	2500		124	50
SA_MAN	10500	.2	100	80
SA REP	11000	.3	149	80
SA REP	8600	.2	149	80
SA REP	7000	.15	149	
AD_ASST	4400		101	10
MK_MAN	13000		100	20
MK REP	6000		201	20
AC_MGR	12000		101	110
AC_ACCOUNT	8300		205	110

20 rows selected.

JOBs 테이블

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
AD_PRES	President	20000	40000
AD_VP	Administration Vice President	15000	30000
AD_ASST	Administration Assistant	3000	6000
AC_MGR	Accounting Manager	8200	16000
AC_ACCOUNT	Public Accountant	4200	9000
SA_MAN	Sales Manager	10000	20000
SA_REP	Sales Representative	6000	12000
ST_MAN	Stock Manager	5500	8500
ST_CLERK	Stock Clerk	2000	5000
IT_PROG	Programmer	4000	10000
MK_MAN	Marketing Manager	9000	15000
MK_REP	Marketing Representative	4000	9000

12 rows selected.

JOB_GRADES 테이블

```
DESCRIBE job_grades
```

Name	Null?	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

```
SELECT * FROM job_grades;
```

GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

6 rows selected.

JOB_HISTORY 테이블

```
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
102	13-JAN-93	24-JUL-98	IT_PROG	60
101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
101	28-OCT-93	15-MAR-97	AC_MGR	110
201	17-FEB-96	19-DEC-99	MK_REP	20
114	24-MAR-98	31-DEC-99	ST_CLERK	50
122	01-JAN-99	31-DEC-99	ST_CLERK	50
200	17-SEP-87	17-JUN-93	AD_ASST	90
176	24-MAR-98	31-DEC-98	SA REP	80
176	01-JAN-99	31-DEC-99	SA_MAN	80
200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

10 rows selected.

LOCATIONS 테이블

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
1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
1500	2011 Interiors Blvd	99236	South San Francisco	California	US
1700	2004 Charade Rd	98199	Seattle	Washington	US
1800	460 Bloor St. W.	ON M5S 1XB	Toronto	Ontario	CA
2500	Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK

REGIONS 테이블

DESCRIBE regions

Name	Null?	Type
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

SELECT * FROM regions;

REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa

C

SQL*Plus 사용

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 부록을 마치면 다음을 수행할 수 있습니다.

- **SQL*Plus**에 로그인
- **SQL** 명령 편집
- **SQL*Plus** 명령을 사용한 출력 형식 지정
- 스크립트 파일과 상호 작용

ORACLE

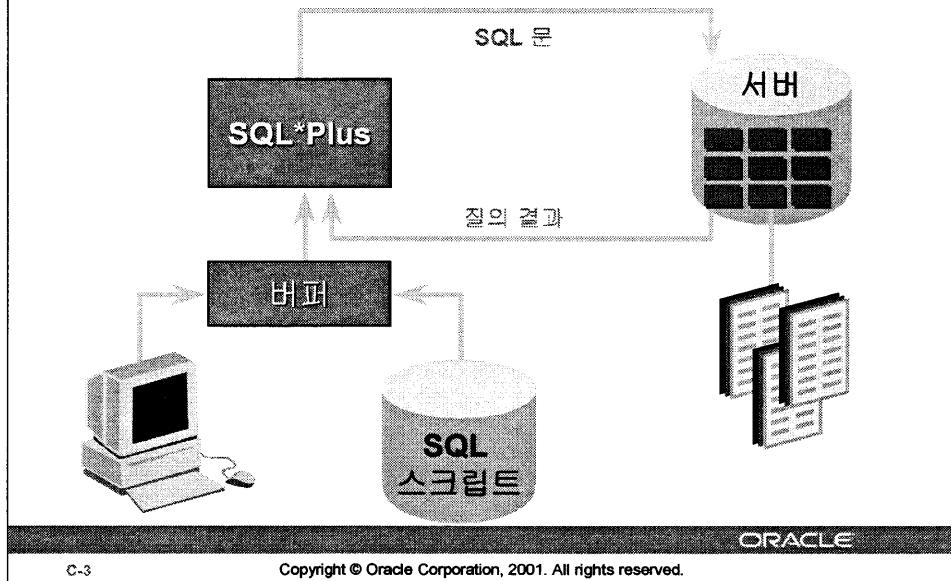
C-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

반복하여 사용할 수 있는 `SELECT` 문을 만들고자 할 경우가 있습니다. 이 단원에서는 이러한 내용과 더불어 `SQL*Plus` 명령을 사용하여 SQL 문을 실행하는 방법을 설명합니다. 또한 `SQL*Plus` 명령을 사용하여 출력 형식을 지정하는 방법, `SQL` 명령을 편집하는 방법 및 `SQL*Plus` 스크립트를 저장하는 방법을 설명합니다.

SQL 및 SQL*Plus 상호 작용



C-3

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

SQL 및 SQL*Plus

SQL은 틀 또는 응용 프로그램에서 Oracle9i Server와 통신하기 위한 명령어입니다. 오라클 SQL은 뛰어난 확장성을 갖추고 있습니다. SQL 문을 입력하면 SQL 버퍼라는 메모리 부분에 저장되며 새 SQL 문을 입력할 때까지 남아 있습니다.

SQL*Plus는 SQL 문을 인식하여 이를 Oracle9i Server에서 실행할 수 있도록 전달해 주는 오라클 툴이며 자체 명령어를 포함합니다.

SQL 특징

- SQL은 프로그래밍 경험이 별로 없거나 전혀 없는 사용자를 포함하여 다양한 사용자 층에서 사용할 수 있습니다.
- 비절차적 언어입니다.
- 시스템 생성 및 유지 관리에 필요한 시간을 줄여 줍니다.
- 영어 형식의 언어입니다.

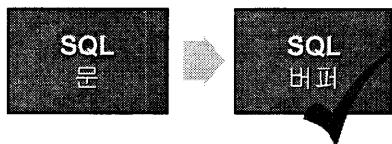
SQL*Plus 특징

- SQL*Plus에서는 명령문의 임시 항목을 사용할 수 있습니다.
- 파일을 통한 SQL 입력을 사용할 수 있습니다.
- SQL 문을 수정할 수 있도록 줄 편집기를 제공합니다.
- 환경 설정을 제어합니다.
- 질의 결과를 기본 보고서 형식으로 지정합니다.
- 로컬 및 원격 데이터베이스에 액세스합니다.

SQL 문과 SQL*Plus 명령 비교

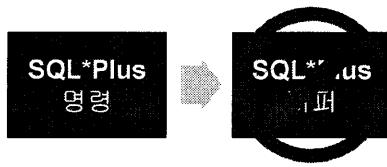
SQL

- 언어
- ANSI 표준**
- 키워드를 약어로 쓸 수 없음
- 명령문이 데이터베이스의 데이터 및 테이블 정의를 조작할 수 있음



SQL*Plus

- 환경
- 오라클 전용
- 키워드를 약어로 쓸 수 있음
- 명령이 데이터베이스의 값을 조작할 수 없음



C-4

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL 및 SQL*Plus(계속)

다음 표에서는 SQL과 SQL*Plus를 비교합니다.

SQL	SQL*Plus
Oracle Server와 통신하여 데이터에 액세스하기 위한 언어	SQL 문을 인식하여 이를 서버로 전송
ANSI(American National Standards Institute) 표준 SQL 기반	SQL 문을 실행하기 위한 오라클 전용의 인터페이스
데이터베이스의 데이터 및 테이블 정의를 조작할 수 있음	데이터베이스의 값을 조작할 수 없음
SQL 버퍼에 하나 이상의 줄이 입력됨	한 번에 한 줄이 입력되며 SQL 버퍼에 저장되지 않음
연결 문자 없음	명령이 여러 줄인 경우 대시(-)를 연결 문자로 사용
약어를 사용할 수 없음	약어를 사용할 수 있음
종료 문자를 사용하여 명령 즉시 실행	종료 문자가 필요 없으며 명령을 즉시 실행
함수를 사용하여 일부 형식 지정을 수행	명령을 사용하여 데이터 형식을 지정

SQL*Plus 개요

- SQL*Plus에 로그인합니다.
- 테이블 구조를 표시합니다.
- SQL 문을 편집합니다.
- SQL*Plus에서 SQL을 실행합니다.
- SQL 문을 파일로 저장하고 파일에 SQL 문을 추가합니다.
- 저장한 파일을 실행합니다.
- 파일에서 버퍼로 명령을 로드하여 편집합니다.

ORACLE

C-5

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL*Plus

SQL*Plus는 다음 작업을 수행할 수 있는 환경입니다.

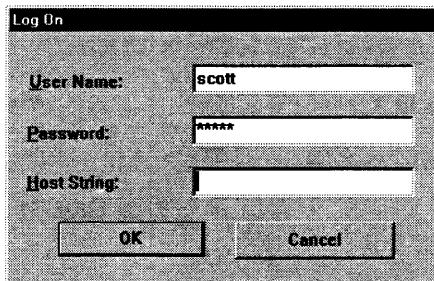
- SQL 문을 실행하여 데이터베이스에서 데이터를 검색, 수정, 추가 및 제거할 수 있습니다.
- 형식 지정, 계산 수행, 저장 및 보고서 형태로 질의 결과를 출력할 수 있습니다.
- SQL 문을 저장하는 스크립트 파일을 작성하여 반복 사용할 수 있습니다.

SQL*Plus 명령은 다음 기본 범주로 구분됩니다.

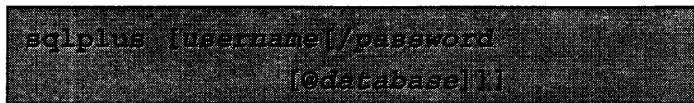
범주	용도
환경	세션에서 SQL 문의 일반적인 작업 방식에 영향을 미침
형식	질의 결과에 대한 형식 지정
파일 조작	스크립트 파일 저장, 로드 및 실행
실행	SQL 버퍼에서 Oracle Server로 SQL 문 전송
편집	버퍼에 있는 SQL 문 수정
상호 작용	변수를 생성하여 SQL 문으로 전달, 변수 값 출력 및 화면에 메시지 출력
기타	데이터베이스에 연결, SQL*Plus 환경 조작 및 열 정의 표시

SQL*Plus에 로그인

- Windows 환경의 경우



- 명령행의 경우



C-6

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

SQL*Plus에 로그인

SQL*Plus 헉출 방법은 실행 중인 운영 체제나 Windows 환경의 유형에 따라 달라집니다.

Windows 환경을 통해 로그인하려면:

- 시작 > 프로그램 > Oracle for Windows NT > SQL*Plus를 선택합니다.
- 사용자 이름, 암호 및 데이터베이스 이름을 입력합니다.

명령행 환경을 통해 로그인하려면:

- 시스템에 로그온합니다.
- 슬라이드에 표시된 것처럼 SQL*Plus 명령을 입력합니다.

구문 설명:

username 데이터베이스 사용자 이름입니다.

password 데이터베이스 암호입니다(여기서 암호를 입력하면 볼 수 있음).

@database 데이터베이스 접속 문자열입니다.

참고: 암호 무결성을 보존하려면 운영 체제 프롬프트에서는 암호를 입력하지 말고 사용자 이름만 입력한 후 Password 프롬프트에 암호를 입력하십시오.

SQL*Plus에 로그인하면 다음 메시지가 표시됩니다(SQL*Plus 버전 9i의 경우).

SQL*Plus: Release 9.0.1.0.0 - Development on Tue Jan 9 08:44:28 2001

(c) Copyright 2000 Oracle Corporation. All rights reserved.

테이블 구조 표시

SQL*Plus DESCRIBE 명령을 사용하여 테이블 구조를 표시합니다.

```
DESC[RIBE] tablename
```

ORACLE

C-7

Copyright © Oracle Corporation, 2001. All rights reserved.

테이블 구조 표시

SQL*Plus에서 DESCRIBE 명령을 사용하면 테이블 구조를 표시할 수 있습니다. 이 명령의 결과로 열 이름 및 데이터 유형이 표시되며 열에 반드시 데이터가 포함되어야 하는지 여부도 알 수 있습니다.

구문 설명:

tablename 사용자가 액세스할 수 있는 기준 테이블, 뷰 또는 동의어의 이름입니다.

JOB_GRADES 테이블의 구조를 표시하려면 다음 명령을 사용합니다.

```
SQL> DESCRIBE job_grades
Name          Null?    Type
-----
GRADE_LEVEL           VARCHAR2(3)
LOWEST_SAL            NUMBER
HIGHEST_SAL           NUMBER
```

테이블 구조 표시

```
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)

ORACLE

C-8

Copyright © Oracle Corporation, 2001. All rights reserved.

테이블 구조 표시(계속)

슬라이드의 예제는 DEPARTMENTS 테이블의 구조에 대한 정보를 표시합니다.

결과 설명:

Null? 열에 반드시 데이터가 포함되어야 하는지를 나타냅니다. NOT NULL이면
 열에 반드시 데이터가 포함되어야 합니다.

Type 열의 데이터 유형을 표시합니다.

다음 표는 데이터 유형을 설명한 것입니다.

데이터 유형	설명
NUMBER (p, s)	최대 자리수가 p, 소수점 이하 자리수가 s인 숫자 값
VARCHAR2 (s)	최대 크기가 s인 가변 길이 문자 값
DATE	B.C. 4712년 1월 1일과 A.D. 9999년 12월 31일 사이의 날짜 및 시간 값
CHAR (s)	크기가 s인 고정 길이 문자 값

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

C-9

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL*Plus 편집 명령

SQL*Plus 명령은 한 번에 한 줄이 입력되어 SQL 버퍼에 저장되지 않습니다.

명령	설명
A [PPEND] text	현재 줄의 끝에 텍스트를 추가합니다.
C [HANGE] / old / new	현재 줄에서 텍스트 <i>old</i> 를 <i>new</i> 로 변경합니다.
C [HANGE] / text /	현재 줄에서 <i>text</i> 를 삭제합니다.
CL [EAR] BUFF [ER]	SQL 버퍼에서 모든 줄을 삭제합니다.
DEL	현재 줄을 삭제합니다.
DEL n	<i>n</i> 번 줄을 삭제합니다.
DEL m n	<i>m</i> 번 줄부터 <i>n</i> 번 줄까지 삭제합니다.

지침

- 명령을 완료하기 전에 [Enter] 키를 누르면 줄 번호가 있는 SQL*Plus 프롬프트가 나타납니다.
- 종료 문자(세미콜론 또는 슬래시) 중 하나를 입력하거나 [Enter] 키를 두 번 눌러 SQL 버퍼를 종료하면 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***
- **O *text***

ORACLE

C-10

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL*Plus 편집 명령(계속)

명령	설명
I [NPUT]	정해지지 않은 개수의 줄을 삽입합니다.
I [NPUT] <i>text</i>	<i>text</i> 로 구성된 줄을 삽입합니다.
L [IST]	SQL 버퍼의 모든 줄을 나열합니다.
L [IST] <i>n</i>	<i>n</i> 번 줄을 나열합니다.
L [IST] <i>m n</i>	<i>m</i> 번 줄부터 <i>n</i> 번 줄까지 나열합니다.
R [UN]	버퍼에서 현재 SQL 문을 표시하고 실행합니다.
<i>n</i>	줄을 지정하여 현재 줄로 만듭니다.
<i>n text</i>	<i>n</i> 번 줄을 <i>text</i> 로 바꿉니다.
O <i>text</i>	1번 줄 앞에 한 줄을 삽입합니다.

참고: SQL 프롬프트 당 하나의 SQL*Plus 명령만 입력할 수 있으며 SQL*Plus 명령은 버퍼에 저장되지 않습니다. SQL*Plus 명령을 다음 줄에 계속하려면 첫째 줄의 끝에 하이픈(-)을 입력하십시오.

LIST, n 및 APPEND 사용

```
SQL> LIST
```

```
1  SELECT last_name  
2* FROM employees
```

```
SQL> 1
```

```
1* SELECT last_name
```

```
SQL> A , job_id
```

```
1* SELECT last_name, job_id
```

```
SQL> L
```

```
1  SELECT last_name, job_id  
2* FROM employees
```

ORACLE

LIST, n 및 APPEND 사용

- L[IST] 명령을 사용하여 SQL 버퍼의 내용을 표시합니다. 버퍼에서 2번 줄 옆에 있는 *는 2번 줄이 현재 줄임을 나타냅니다. 편집한 내용은 현재 줄에 적용됩니다.
- 편집하려는 줄의 번호를 입력하여 현재 줄로 변경합니다. 새로운 현재 줄이 표시됩니다.
- A[PPEND] 명령을 사용하여 현재 줄에 텍스트를 추가합니다. 새로 편집된 줄이 표시됩니다. LIST 명령을 사용하여 버퍼에 새로 적용된 내용을 확인합니다.

참고: LIST 및 APPEND를 포함한 다수의 SQL*Plus 명령을 첫번째 문자만 표시하여 약어로 사용할 수 있습니다. LIST는 L, APPEND는 A로 대신 사용할 수 있습니다.

CHANGE 명령 사용

```
SQL> L
```

```
1* SELECT * from employees
```

```
SQL> C/employees/departments
```

```
1* SELECT * from departments
```

```
SQL> L
```

```
1* SELECT * from departments
```

ORACLE®

CHANGE 명령 사용

- L[IST]를 사용하여 버퍼의 내용을 표시합니다.
- C[HANGE] 명령을 사용하여 SQL 버퍼에 있는 현재 줄의 내용을 변경합니다.
이 경우 사원 테이블을 부서 테이블로 바꿉니다. 새로운 현재 줄이 표시됩니다.
- L[IST] 명령을 사용하여 버퍼의 새로운 내용을 확인합니다.

SQL*Plus 파일 명령

- **SAVE filename**
- **GET filename**
- **START filename**
- **@ filename**
- **EDIT filename**
- **SPOOL filename**
- **EXIT**

ORACLE

C-13

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL*Plus 파일 명령

SQL 문은 Oracle Server와 통신합니다. SQL*Plus 명령은 환경을 제어하고 질의 결과의 형식을 지정하며 파일을 관리합니다. 이를 위해 다음 표에 설명된 명령을 사용할 수 있습니다.

명령	설명
SAV[E] filename [.ext] [REP[LACE]APP[END]]	현재 SQL 버퍼 내용을 파일에 저장합니다. APPEND를 사용하여 기존 파일에 추가하거나 REPLACE를 사용하여 기존 파일을 덮어씁니다. 기본 확장자는 .sql입니다.
GET filename [.ext]	이전에 저장된 파일의 내용을 SQL 버퍼로 읽어들입니다. 파일 이름의 기본 확장자는 .sql입니다.
STA[RT] filename [.ext]	이전에 저장된 명령 파일을 실행합니다.
@ filename	이전에 저장된 명령 파일을 실행합니다(START와 동일).
ED[IT]	편집기를 호출하고 버퍼 내용을 afiedt.buf 파일에 저장합니다.
ED[IT] [filename [.ext]]	편집기를 호출하여 저장된 파일의 내용을 편집합니다.
SPO[OL] [filename [.ext]] OFF OUT	질의 결과를 파일에 저장합니다. OFF를 사용하면 스플파일을 닫습니다. OUT을 사용하면 스플파일을 닫고 파일 결과를 시스템 프린터로 전송합니다.
EXIT	SQL*Plus를 종료합니다.

SAVE 및 START 명령 사용

```
SQL> L
  1 SELECT last_name, manager_id, department_id
  2* FROM employees
SQL> SAVE my_query
```

```
Created file my_query
```

```
SQL> START my_query
```

LAST_NAME	MANAGER_ID	DEPARTMENT_ID
King		90
Kochhar	100	90
...		
20 rows selected.		

ORACLE

SAVE

SAVE 명령을 사용하여 버퍼의 현재 내용을 파일에 저장합니다. 이러한 방식으로 자주 사용하는 스크립트를 저장하여 나중에 사용할 수 있습니다.

START

START 명령을 사용하여 SQL*Plus에서 스크립트를 실행합니다.

EDIT

EDIT 명령을 사용하여 기존 스크립트를 편집합니다. 이 명령을 사용하면 편집기가 열리고 그 안에 해당 스크립트 파일이 열립니다. 스크립트를 변경한 후 편집기를 종료하면 SQL*Plus 명령행으로 되돌아갑니다.

요약

SQL*Plus 환경을 사용하여 다음 작업을 수행합니다.

- **SQL** 문을 실행합니다.
- **SQL** 문을 편집합니다.
- 출력 형식을 지정합니다.
- 스크립트 파일과 상호 작용합니다.

ORACLE

C-15

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

SQL*Plus는 SQL 명령을 데이터베이스 서버로 전송하고 SQL 명령을 편집 및 저장하는 데 사용할 수 있는 실행 환경입니다. 명령은 SQL 프롬프트 또는 스크립트 파일에서 실행할 수 있습니다.

D

고급 스크립트 작성

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 부록을 마치면 다음을 수행할 수 있습니다.

- **SQL**을 생성하는 **SQL**을 사용하여 해결할 수 있는 문제 유형 설명
- **DROP TABLE** 문의 스크립트를 생성하는 스크립트 작성
- **INSERT INTO** 문의 스크립트를 생성하는 스크립트 작성

ORACLE

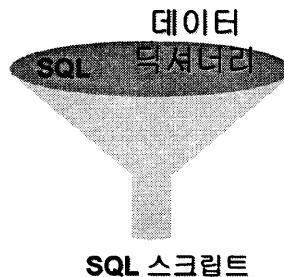
D-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 부록에서는 **SQL** 스크립트를 생성하는 **SQL** 스크립트를 작성하는 방법을 설명합니다.

SQL을 생성하는 SQL 사용



- **SQL을 사용하여 SQL 스크립트를 생성할 수 있습니다.**
- **데이터 딕셔너리**
 - 데이터베이스 정보가 들어 있는 테이블 및 뷰의 **collection**입니다.
 - **Oracle Server**에 의해 생성되며 유지 관리됩니다.

ORACLE

D-3

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL을 생성하는 SQL 사용

SQL은 다른 SQL 문을 작성할 수 있는 강력한 툴로서, 대부분의 경우 스크립트 파일을 작성하는 데 사용됩니다. SQL에서 생성된 SQL은 다음과 같은 작업에 유용합니다.

- 반복적인 코딩을 피합니다.
- 데이터 딕셔너리의 정보에 액세스합니다.
- 데이터베이스 객체를 삭제하거나 다시 생성합니다.
- 런타임 파라미터를 포함하는 동적 쿼리를 생성합니다.

이 단원에 사용된 예제는 데이터 딕셔너리의 정보를 선택하는 내용입니다. 데이터 딕셔너리는 데이터베이스 정보를 포함하는 테이블 및 뷰의 collection이며 Oracle Server에 의해 생성되고 유지 관리됩니다. 모든 데이터 딕셔너리 테이블은 SYS 사용자 소유입니다. 데이터 딕셔너리

에는 Oracle Server 사용자 이름, 사용자 권한, 데이터베이스 객체 이름, 테이블 제약 조건 및 감사 정보가 들어 있습니다. 데이터 딕셔너리 뷰에는 네 가지 범주가 있으며, 각 범주 이름에는 사용 목적을 나타내는 고유 접두어가 붙습니다.

접두어	설명
USER_	사용자 소유의 객체에 대한 세부 정보를 포함합니다.
ALL_	사용자 소유의 객체 뿐 아니라 사용자가 액세스 권한을 가지고 있는 객체의 세부 정보를 포함합니다.
DBA_	데이터베이스의 모든 객체에 액세스할 수 있는 DBA 권한을 가진 사용자의 세부 정보를 포함합니다.
V\$_	데이터베이스 서버 성능 및 잠금에 대한 저장 정보입니다. DBA만 사용할 수 있습니다.

기본 스크립트 작성

```
SELECT 'CREATE TABLE ' || table_name || '_test '
      || 'AS SELECT * FROM ' || table_name
      || ' WHERE 1=2;'
      AS "Create Table Script"
FROM user_tables;
```

Create Table Script
CREATE TABLE COUNTRIES_test AS SELECT * FROM COUNTRIES WHERE 1=2;
CREATE TABLE DEPARTMENTS_test AS SELECT * FROM DEPARTMENTS WHERE 1=2;
CREATE TABLE EMPLOYEES_test AS SELECT * FROM EMPLOYEES WHERE 1=2;
CREATE TABLE JOBS_test AS SELECT * FROM JOBS WHERE 1=2;
CREATE TABLE JOB_GRADES_test AS SELECT * FROM JOB_GRADES WHERE 1=2;
CREATE TABLE JOB_HISTORY_test AS SELECT * FROM JOB_HISTORY WHERE 1=2;
CREATE TABLE LOCATIONS_test AS SELECT * FROM LOCATIONS WHERE 1=2;
CREATE TABLE REGIONS_test AS SELECT * FROM REGIONS WHERE 1=2;

8 rows selected.

ORACLE

D-4

Copyright © Oracle Corporation, 2001. All rights reserved.

기본 스크립트

슬라이드의 예제는 사용자 소유의 모든 테이블에 대해 CREATE TABLE 문을 사용하여 보고서를 작성합니다. 보고서에 작성된 각 CREATE TABLE 문에는 테이블을 생성하는 구문이 포함되어 있는데, 테이블 이름에는 _test 접미어가 붙고 테이블 구조는 대응되는 기존 테이블의 구조를 사용합니다. 기존 테이블 이름은 데이터 덕셔너리 뷰 USER_TABLES의 TABLE_NAME 열에서 가져옵니다.

다음 단계는 프로세스를 자동화하도록 보고서의 기능을 향상시키는 것입니다.

참고: 데이터 덕셔너리 테이블을 질의하여 사용자가 소유한 다양한 데이터베이스 객체를 볼 수 있습니다. 자주 사용되는 데이터 덕셔너리 테이블은 다음과 같습니다.

- **USER_TABLES:** 사용자 소유의 테이블에 대한 설명을 표시합니다.
- **USER_OBJECTS:** 사용자 소유의 객체를 모두 표시합니다.
- **USER_TAB_PRIVS_MADE:** 사용자 소유의 객체에 대한 권한을 모두 표시합니다.
- **USER_COL_PRIVS_MADE:** 사용자가 소유한 객체의 열에 대한 권한을 모두 표시합니다.

환경 제어

```
SET ECHO OFF  
SET FEEDBACK OFF  
SET PAGESIZE 0  
  
SPOOL dropem.sql  
  
SQL STATEMENT  
  
SPOOL OFF  
  
SET FEEDBACK ON  
SET PAGESIZE 24  
SET ECHO ON
```

시스템 변수를 적절한
값으로 설정합니다.

시스템 변수를 기본값
으로 다시 설정합니다.

ORACLE

D-5

Copyright © Oracle Corporation, 2001. All rights reserved.

환경 제어

생성된 SQL 문을 실행하려면 이를 스펠 파일에 캡처하여 실행해야 합니다. 또한 생성된 출력
을 정리하거나 머리글, 피드백 메시지, 상단 제목 등의 요소가 표시되지 않도록 해야 합니다.
iSQL*Plus 명령을 사용하면 이러한 작업을 모두 수행할 수 있습니다.

완성된 결과

```
SET ECHO OFF
SET FEEDBACK OFF
SET PAGESIZE 0

SELECT 'DROP TABLE ' || object_name || '';
FROM   user_objects
WHERE  object_type = 'TABLE'
/

SET FEEDBACK ON
SET PAGESIZE 24
SET ECHO ON
```

ORACLE

D-6

Copyright © Oracle Corporation, 2001. All rights reserved.

완성된 결과

슬라이드에 있는 명령의 출력은 iSQL*Plus에서 File이라는 Output 옵션을 사용하여 dropem.sql이라는 파일에 저장됩니다. 이 파일에는 다음 데이터가 포함됩니다. 이제 이 스크립트 파일을 찾아 로드한 다음 실행함으로써 이 파일을 iSQL*Plus에서 시작할 수 있습니다.

'DROPTABLE' OBJECT_NAME ';'
DROP TABLE COUNTRIES;
DROP TABLE DEPARTMENTS;
DROP TABLE EMPLOYEES;
DROP TABLE JOBS;
DROP TABLE JOB_GRADES;
DROP TABLE JOB_HISTORY;
DROP TABLE LOCATIONS;
DROP TABLE REGIONS;

참고: 기본적으로 파일은 Windows NT의 ORACLE_HOME\ORANT\BIN 폴더에 스플립니다.

테이블 내용을 파일로 덤프

```
SET HEADING OFF ECHO OFF FEEDBACK OFF
SET PAGESIZE 0

SELECT
  'INSERT INTO departments_test VALUES
  (' || department_id || ', "' || department_name ||
  ' ", "' || location_id || '")';
  AS "Insert Statements Script"
FROM   departments
/

SET PAGESIZE 24
SET HEADING ON ECHO ON FEEDBACK ON
```

ORACLE

D-7

Copyright © Oracle Corporation, 2001. All rights reserved.

테이블 내용을 파일로 덤프

경우에 따라 테이블 행(row)의 값을 INSERT INTO VALUES 문의 형식으로 텍스트 파일에 저장해 놓으면 유용하게 사용할 수 있습니다. 실수로 테이블이 삭제된 경우 이 스크립트를 실행하여 테이블을 채울 수 있습니다.

슬라이드의 예제는 DEPARTMENTS_TEST 테이블에 대한 INSERT 문을 생성하는데, 이 결과는 iSQL*Plus에서 File이라는 Output 옵션을 사용하여 data.sql 파일에 캡처됩니다.

data.sql 스크립트 파일의 내용은 다음과 같습니다.

```
INSERT INTO departments_test VALUES
  (10, 'Administration', 1700);
INSERT INTO departments_test VALUES
  (20, 'Marketing', 1800);
INSERT INTO departments_test VALUES
  (50, 'Shipping', 1500);
INSERT INTO departments_test VALUES
  (60, 'IT', 1400);
...
```

테이블 내용을 파일로 덤프

원본	결과
' ''X'' '	'X'
' '' '' '	' '
' '' '' department_name '' '' '	'Administration'
' '' , '' '	', '
' '') ; '	') ;

ORACLE

D-8

Copyright © Oracle Corporation, 2001. All rights reserved.

테이블 내용을 파일로 덤프(계속)

이전 페이지의 슬라이드에서 매우 많은 작은 따옴표(')를 볼 수 있습니다. 마지막 명령문에 있는 네 개의 작은 따옴표는 하나의 작은 따옴표로 표시됩니다. 또한 문자 및 날짜 값을 작은 따옴표로 묶어야 한다는 점도 기억하십시오.

문자열 내에서 하나의 작은 따옴표를 표시하려면 다른 작은 따옴표를 앞에 붙여야 합니다. 예를 들어, 슬라이드의 다섯번째 예제에서 맨 앞과 맨 뒤의 작은 따옴표는 전체 문자열을 묶은 것입니다. 두번쩨 작은 따옴표는 세번쩨 작은 따옴표를 표시하기 위해 앞에 붙인 것입니다. 따라서 결과적으로 하나의 작은 따옴표, 팔호 및 세미콜론(:)이 표시됩니다.

동적 술어 생성

```
COLUMN my_col NEW_VALUE dyn_where_clause

SELECT DECODE('&&deptno', null,
DECODE ('&&hiredate', null, '',
'WHERE hire_date=TO_DATE('' || '&hiredate'', ''DD-MON-YYYY'')'),
DECODE ('&&hiredate', null,
'WHERE department_id = ' || '&&deptno',
'WHERE department_id = ' || '&&deptno' ||
' AND hire_date = TO_DATE('' || '&hiredate'', ''DD-MON-YYYY'')))
AS my_col FROM dual;
```

```
SELECT last_name FROM employees &dyn_where_clause;
```

#

ORACLE

D-9

Copyright © Oracle Corporation, 2001. All rights reserved.

동적 술어 생성

슬라이드의 예제는 한 부서에서 특정 날짜에 입사한 모든 사원 데이터를 검색하는 SELECT 문을 생성합니다. 이 스크립트는 WHERE 절을 동적으로 생성합니다.

참고: 이전에 만들어진 사용자 변수가 있을 경우 UNDEFINE 명령을 사용하여 삭제해야 합니다.

첫번째 SELECT 문은 사용자에게 부서 번호를 입력하라는 프롬프트를 표시합니다. 부서 번호를 입력하지 않으면 DECODE 함수에 의해 부서 번호가 널로 처리되며 다음으로 입사일을 입력하라는 프롬프트가 표시됩니다. 입사일을 입력하지 않으면 DECODE 함수에 의해 입사일이 널로 처리되고, 따라서 생성된 동적 WHERE 절 또한 널이 됩니다. 그 결과 두번째 SELECT 문은 EMPLOYEES 테이블에서 모든 행(row)을 검색합니다.

참고: NEW_V[ALUE] 변수는 열 값을 보유할 변수를 지정합니다. 이 변수는 TTITLE 명령에서 참조할 수 있습니다. NEW_VALUE를 사용하여 상단 제목에 열 값 또는 날짜를 표시합니다. 이 열을 SKIP PAGE 작업과 함께 BREAK 명령에 포함시켜야 합니다. 변수 이름은 파운드 기호(#)를 포함할 수 없습니다. NEW_VALUE는 각 페이지가 새로운 마스터 레코드를 갖는 마스터/디테일 보고서에 유용합니다.

동적 쿼리 생성(계속)

참고: 여기서 입사일은 DD-MON-YYYY 형식으로 입력해야 합니다.

이전 슬라이드의 SELECT 문은 다음과 같이 해석할 수 있습니다.

```
IF (<<deptno>>가 입력되지 않으면) THEN  
  IF (<<hiredate>>가 입력되지 않으면) THEN  
    빈 문자열을 반환합니다.  
  ELSE  
    'WHERE hire_date = TO_DATE('<<hiredate>>', 'DD-MON-YYYY')' 문자열을  
    반환합니다.  
  ELSE  
    IF (<<hiredate>>가 입력되지 않으면) THEN  
      'WHERE department_id = 입력된<<deptno>>' 문자열을 반환합니다.  
    ELSE  
      'WHERE department_id = 입력된<<deptno>>  
        AND hire_date = TO_DATE('<<hiredate>>', 'DD-MON-YYYY')'  
      문자열을 반환합니다.  
    END IF
```

반환된 문자열은 변수 DYN_WHERE_CLAUSE의 값이 되며 이 값은 둘째 SELECT 문에서 사용됩니다.

슬라이드의 첫번째 예제를 실행하면 다음과 같이 DEPTNO 및 HIREDATE 값을 입력하라는 프롬프트가 표시됩니다.



Define Substitution Variables

"deptno"	10
"hiredate"	17-SEP-1987

다음과 같이 MY_COL 값이 생성됩니다.

MY_COL
WHERE department_id = 10 AND hire_date = TO_DATE('17-SEP-1987', 'DD-MON-YYYY')

슬라이드의 두번째 예제를 실행하면 다음과 같은 결과가 생성됩니다.

LAST_NAME
Whalen

요약

이 부록에서 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- 다른 **SQL** 스크립트를 생성하는 **SQL** 스크립트 작성
- 스크립트 파일에 종종 데이터 덕셔너리 사용
- 출력을 파일에 캡처

ORACLE

D-11

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

SQL을 사용하여 SQL 스크립트를 생성할 수 있습니다. 이러한 스크립트를 사용하면 반복되는 코딩을 피할 수 있고, 객체를 삭제하거나 다시 생성할 수 있으며, 데이터 덕셔너리를 이용할 수 있고, 런타임 파라미터를 포함하는 동적 술어를 생성할 수 있습니다.

iSQL*Plus 명령을 사용하면 SQL 문에서 생성된 보고서를 캡처할 수 있으며 생성된 출력을 정리하여 머리글, 피드백 메시지 등이 표시되지 않도록 할 수 있습니다.

연습 D 개요

이 연습에서는 다음 내용을 다룹니다.

- 테이블의 데이터를 기술하고 선택하는 스크립트 작성
- 사용자 권한을 취소하는 스크립트 작성

ORACLE

D-42

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 D 개요

이 연습에서는 SQL을 생성하는 SQL을 작성해 봅니다.

연습 D

1. 테이블의 데이터를 기술하고 선택하는 스크립트를 작성하십시오. SELECT 목록에서 CHR(10)과 함께 연결 문자(||)를 사용하여 보고서에서 줄 바꿈이 되도록 하십시오. 스크립트의 출력을 my_file1.sql에 저장하십시오. 파일을 저장하려면 Output 옵션으로 File을 선택하고 코드를 실행합니다. 파일을 저장할 때는 .sql 확장자를 사용해야 한다는 점에 유의하십시오. my_file1.sql을 실행하려면 해당 스크립트를 찾아 로드한 다음 실행합니다.
2. SQL을 사용하여 사용자 권한을 취소하는 SQL 문을 생성하십시오. 데이터 딕셔너리 뷰 USER_TAB_PRIVS_MADE 및 USER_COL_PRIVS_MADE를 사용하십시오.
 - a. \Lab\privs.sql 스크립트를 실행하여 SYSTEM 사용자에게 권한을 부여하십시오.
 - b. 데이터 딕셔너리 뷰를 질의하여 권한을 확인하십시오. 예제 출력에서 GRANTOR 열의 데이터는 GRANTOR에 따라 달라질 수 있습니다. 또한 잘린 마지막 행(row)은 GRANTABLE 열입니다.

GRANTEE	TABLE_NAME	GRANTOR	PRIVILEGE	GRA	HIE
SYSTEM	DEPARTMENT S	SQL2	ALTER	NO	NO
SYSTEM	DEPARTMENT S	SQL2	DELETE	NO	NO
SYSTEM	DEPARTMENT S	SQL2	INDEX	NO	NO
SYSTEM	DEPARTMENT S	SQL2	INSERT	NO	NO
SYSTEM	DEPARTMENT S	SQL2	SELECT	NO	NO
SYSTEM	DEPARTMENT S	SQL2	UPDATE	NO	NO
SYSTEM	DEPARTMENT S	SQL2	REFERENCES	NO	NO
SYSTEM	DEPARTMENT S	SQL2	ON COMMIT REFRESH	NO	NO
SYSTEM	DEPARTMENT S	SQL2	QUERY REWRITE	NO	NO
SYSTEM	DEPARTMENT S	SQL2	DEBUG	NO	NO

10 rows selected.

GRANTEE	TABLE_NAME	COLUMN_NAM	GRANTOR	PRIVILEGE	GRA
SYSTEM	EMPLOYEES	JOB_ID	SQL2	UPDATE	NO
SYSTEM	EMPLOYEES	SALARY	SQL2	UPDATE	NO

연습 D(계속)

- c. 권한을 취소하는 스크립트를 작성하십시오. 스크립트의 출력을 my_file2.sql에 저장하십시오. 파일을 저장하려면 Output 옵션으로 File을 선택하고 코드를 실행합니다. .sql 확장자를 사용하여 파일을 저장해야 한다는 점에 유의하십시오. my_file2.sql을 실행하려면 해당 스크립트를 찾아 로드한 다음 실행합니다.

```
'REVOKE||PRIVILEGE||'ON'||TABLE_NAME||'FROM SYSTEM;'
```

```
REVOKE ALTER ON DEPARTMENTS FROM system;
```

```
REVOKE DELETE ON DEPARTMENTS FROM system;
```

```
REVOKE INDEX ON DEPARTMENTS FROM system;
```

```
REVOKE INSERT ON DEPARTMENTS FROM system;
```

```
REVOKE SELECT ON DEPARTMENTS FROM system;
```

```
REVOKE UPDATE ON DEPARTMENTS FROM system;
```

```
REVOKE REFERENCES ON DEPARTMENTS FROM system;
```

```
REVOKE ON COMMIT REFRESH ON DEPARTMENTS FROM system;
```

```
REVOKE QUERY REWRITE ON DEPARTMENTS FROM system;
```

```
REVOKE DEBUG ON DEPARTMENTS FROM system;
```

10 rows selected.

```
'REVOKE||PRIVILEGE||'ON'||TABLE_NAME||'FROM SYSTEM;'
```

```
REVOKE UPDATE ON EMPLOYEES FROM system;
```

Oracle의 구조적 구성 요소

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 부록을 마치면 다음을 수행할 수 있습니다.

- Oracle Server 구조 및 기본 구성 요소 설명
- 사용자를 오라클 인스턴스에 연결할 때 관련되는 구조
나열
- 다음의 처리 단계 나열
 - 질의
 - DML 문
 - 커밋

ORACLE®

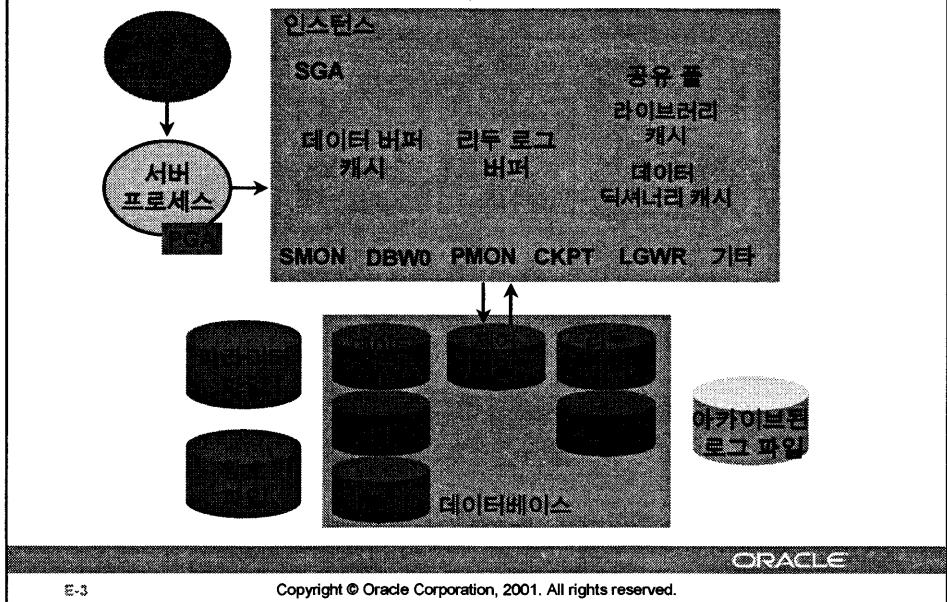
E-2

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 부록에서는 데이터베이스 연결 및 SQL 명령 실행과 관련된 파일, 프로세스 및 메모리 구조를 설명함으로써 Oracle Server 구조를 설명합니다.

개요



E-3

Copyright © Oracle Corporation, 2001. All rights reserved.

개요

Oracle Server는 개방적이면서도 광범위한 통합 정보 관리 방식을 제공하는 객체 관계형 데이터베이스 관리 시스템입니다.

기본 구성 요소

Oracle Server에는 여러 개의 프로세스, 메모리 구조 및 파일이 있지만 이들이 모두 SQL 문을 처리하는 테 사용되지는 않습니다. 이를 중 일부는 데이터베이스의 성능을 향상시키거나, 소프트웨어 또는 하드웨어 오류가 발생한 경우 데이터베이스를 복구하거나, 데이터베이스의 유지 관리에 필요한 다른 작업을 수행하는 테 사용됩니다. Oracle Server는 오라클 인스턴스와 오라클 데이터베이스로 구성됩니다.

오라클 인스턴스

오라클 인스턴스는 백그라운드 프로세스와 메모리 구조가 결합된 것입니다. 인스턴스가 시작되어야 데이터베이스의 데이터에 액세스할 수 있습니다. 인스턴스가 시작될 때마다 시스템 글로벌 영역(SGA)이 할당되고 오라클 백그라운드 프로세스가 시작됩니다. SGA는 데이터베이스 프로세스에서 공유하는 데이터베이스 정보를 저장하는 테 사용되는 메모리 영역입니다.

백그라운드 프로세스는 프로세스를 호출하는 대신 기능을 수행합니다. 백그라운드 프로세스는 각 사용자가 여러 개의 오라클 프로그램을 별개적으로 실행하여 처리해야 할 기능들을 통합하는 역할을 합니다. 백그라운드 프로세스는 I/O를 수행하며 다른 오라클 프로세스를 모니터하여 성능 및 신뢰성이 높은 보다 향상된 병렬화를 제공합니다.

기본 구성 요소(계속)

기타 프로세스

사용자 프로세스는 SQL 문을 생성하는 응용 프로그램입니다. 서버 프로세스는 사용자 프로세스로부터 받은 SQL 문을 실행합니다.

데이터베이스 파일

데이터베이스 파일은 데이터베이스 정보에 대한 실제 물리적인 저장 영역을 제공하는 운영 체제 파일입니다. 데이터베이스 파일을 사용하면 데이터 일관성이 보장되며 인스턴스 실패 시 데이터를 복구할 수 있습니다.

기타 파일

데이터베이스 파일이 아닌 파일은 인스턴스를 구성하고, 권한이 있는 사용자를 인증하며, 디스크 고장 시 데이터베이스를 복구하는 데 사용됩니다.

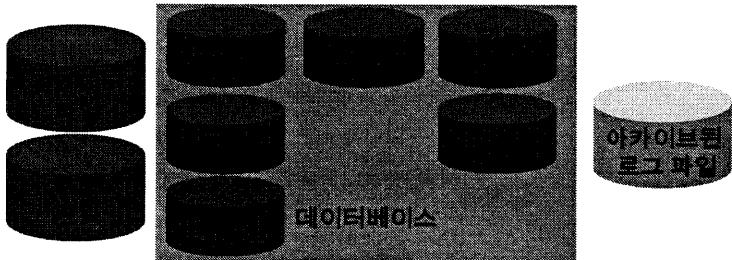
SQL 문 처리

사용자 프로세스와 서버 프로세스는 SQL 문이 실행될 때 사용되는 기본 프로세스지만 기타 프로세스는 서버가 SQL 문 처리를 완료하는 데 도움을 줍니다.

오라클 데이터베이스 관리자

데이터베이스 관리자는 서버에서 사용자 요청을 처리할 수 있도록 Oracle Server를 유지 관리합니다. 효율적인 관리를 위해서는 오라클 구조에 대해 이해하고 있어야 합니다.

오라클 데이터베이스 파일



ORACLE

E-5

Copyright © Oracle Corporation, 2001. All rights reserved.

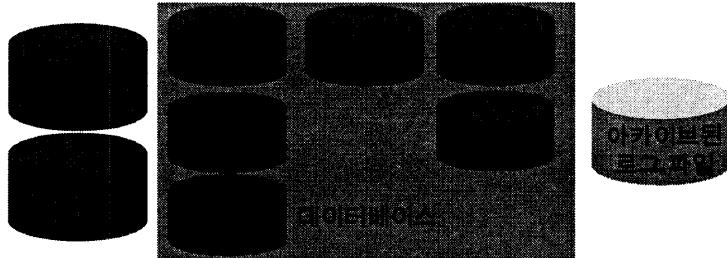
오라클 데이터베이스 파일

오라클 데이터베이스는 한 단위로 처리되는 데이터의 collection입니다. 일반적으로 데이터베이스는 관련된 정보를 저장하고 검색하는 데 사용됩니다. 데이터베이스에는 논리적 구조와 물리적 구조가 있습니다. 데이터베이스의 물리적 구조는 데이터베이스에 있는 운영 체제 파일의 집합입니다. 오라클 데이터베이스는 다음과 같은 세 가지 파일 유형으로 구성됩니다.

데이터 파일은 데이터베이스의 실제 데이터를 포함합니다. 데이터는 사용자 정의 테이블에 저장되지만 데이터 파일에는 데이터 딕셔너리, 수정된 데이터의 이전 이미지, 인덱스 및 다른 유형의 구조도 포함됩니다. 데이터베이스에는 적어도 하나의 데이터 파일이 있습니다. 데이터 파일의 특징은 다음과 같습니다.

- 하나의 데이터 파일은 하나의 데이터베이스에만 관련될 수 있습니다. 데이터 파일에는 몇 가지 특성 집합이 있어서 데이터베이스 공간이 부족할 때 자동으로 확장됩니다. 하나 이상의 데이터 파일이 모여 데이터베이스 저장 영역의 논리적인 단위를 구성하며 이를 테이블스페이스라고 합니다. 리두 로그는 데이터베이스에 대한 변경 내용이 기록된 레코드를 포함하고 있어서 실패가 발생한 경우 데이터를 복구할 수 있도록 해줍니다. 하나의 데이터베이스에는 적어도 두 개의 리두 로그 파일이 있어야 합니다.
- 제어 파일은 데이터베이스 무결성을 유지 관리하고 확인하는 데 필요한 정보를 포함합니다. 예를 들어, 어떤 제어 파일은 데이터 파일과 리두 로그 파일을 식별하는 데 사용됩니다. 데이터베이스에는 적어도 하나의 제어 파일이 있어야 합니다.

기타 중요한 물리적 구조



ORACLE

기타 중요한 파일

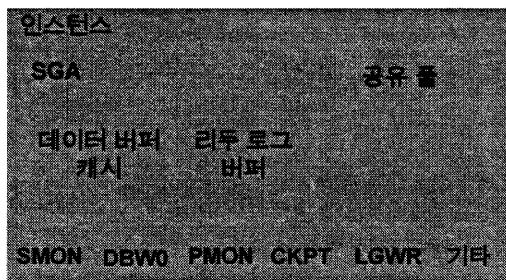
Oracle Server는 데이터베이스의 일부가 아닌 다른 파일도 사용합니다.

- 파라미터 파일은 오라클 인스턴스의 특성을 정의합니다. 예를 들어, SGA의 메모리 구조 중 일부의 크기를 조정하는 파라미터를 포함합니다.
- 암호 파일은 오라클 인스턴스의 시작 및 종료 권한이 있는 사용자를 인증합니다.
- 아카이브된 리두 로그 파일은 매체가 고장난 경우 복구하는 데 필요한 리두 로그 파일의 오프라인 복사본입니다.

오라클 인스턴스

오라클 인스턴스

- 오라클 데이터베이스에 액세스하기 위한 수단입니다.
- 항상 하나의 데이터베이스만 열니다.



오라클 인스턴스

오라클 인스턴스는 데이터베이스를 관리하는 데 사용되는 SGA 메모리 구조 및 백그라운드 프로세스로 구성됩니다. 인스턴스는 운영 체제별로 다른 방법에 의해 식별됩니다. 인스턴스는 한번에 하나의 데이터베이스만 열고 사용할 수 있습니다.

시스템 글로벌 영역

SGA는 데이터베이스 프로세스에서 공유하는 데이터베이스 정보를 저장하는 데 사용되는 메모리 영역입니다. 여기에는 Oracle Server에 대한 데이터 및 제어 정보가 들어 있습니다. SGA는 Oracle server가 상주하는 컴퓨터의 가상 메모리에 할당되며, 여러 개의 메모리 구조로 구성됩니다.

- 공유 풀은 가장 최근에 실행된 SQL 문과 데이터 딕셔너리에서 가장 최근에 사용된 데이터를 저장하는 데 사용됩니다. 이러한 SQL 문은 사용자 프로세스에 의해 제출되었거나 내장 프로시저의 경우에는 데이터 딕셔너리로부터 읽은 것입니다.
- 데이터베이스 버퍼 캐시는 가장 최근에 사용된 데이터를 저장하는 데 사용됩니다. 이 데이터는 데이터 파일에서 읽어 오거나 데이터 파일에 쓰여집니다.
- 리두 로그 버퍼는 서버 및 백그라운드 프로세스에 의해 변경된 데이터베이스의 변경 내용을 추적하는 데 사용됩니다.

오라클 인스턴스

시스템 글로벌 영역(계속)

이러한 구조를 사용하는 목적은 이 단원의 후반부에서 자세하게 설명합니다.

SGA에는 다음과 같은 두 가지의 선택적인 메모리 구조도 있습니다.

- Java 풀: Java 코드를 저장하는 데 사용됩니다.
- 대용량 풀: SQL 문 처리에 직접적으로 관련되지 않은 대용량 메모리 구조를 저장하는 데 사용됩니다. 이러한 예로는 백업 및 복원 작업 동안 복사되는 데이터 블록이 있습니다.

백그라운드 프로세스

백그라운드 프로세스는 인스턴스에서 무결성 및 시스템 성능에 영향을 미치지 않고 동시 사용자의 요청을 처리하는 데 필요한 일반적인 기능을 수행합니다. 백그라운드 프로세스는 각 사용자가 여러 개의 오라클 프로그램을 개별적으로 실행하여 처리해야 할 기능들을 통합하는 역할을 합니다.

백그라운드 프로세스는 I/O를 수행하며 다른 오라클 프로세스를 모니터하여 성능 및 신뢰성이 높은 보다 항상된 병렬화를 제공합니다.

오라클 인스턴스는 구성에 따라 여러 백그라운드 프로세스를 포함할 수 있지만 모든 인스턴스에는 다음과 같은 다섯 가지 필수 백그라운드 프로세스가 포함됩니다.

- 데이터베이스 기록자(DBW0)는 데이터베이스 버퍼 캐시에서 변경된 데이터를 데이터 파일에 씁니다.
- 로그 기록자(LGWR)는 리두 로그 버퍼에 등록된 변경 내용을 리두 로그 파일에 씁니다.
- 시스템 모니터(SMON)는 데이터베이스의 일관성을 확인하고, 필요한 경우 데이터베이스가 열릴 때 데이터베이스 복구를 시작합니다.
- 프로세스 모니터(PMON)는 오라클 프로세스 중 하나가 실패하면 자원을 정리합니다.
- 체크포인트 프로세스(CKPT)는 버퍼 캐시의 변경 내용이 영구적으로 데이터베이스에 기록될 때마다 제어 파일 및 데이터 파일의 데이터베이스 상태 정보를 갱신합니다.

이 단원의 다음 부분에서는 서버 프로세스가 오라클 인스턴스 및 데이터베이스의 몇몇 구성 요소를 사용하여 사용자 프로세스에서 제출한 SQL문을 처리하는 방법을 설명합니다.

SQL 문 처리

- 다음을 사용하여 인스턴스에 연결합니다.
 - 사용자 프로세스
 - 서버 프로세스
- 다음과 같은 SQL 문의 유형에 따라 사용되는 Oracle Server 구성 요소가 달라집니다.
 - 질의의 경우 행(row) 반환
 - DML 문의 경우 변경 내용을 기록
 - 커밋의 경우 트랜잭션 복구를 보장
- 일부 Oracle Server 구성 요소는 SQL 문 처리에 참여하지 않습니다.

ORACLE

E-3

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL 처리에 사용되는 구성 요소

SQL 문을 처리하는 데 오라클 인스턴스의 모든 구성 요소가 사용되지는 않습니다. 사용자 프로세스 및 서버 프로세스는 사용자를 오라클 인스턴스에 연결하는 데 사용됩니다. 이러한 프로세스는 오라클 인스턴스의 일부가 아니며 SQL 문을 처리하는 데 반드시 필요하지는 않습니다.

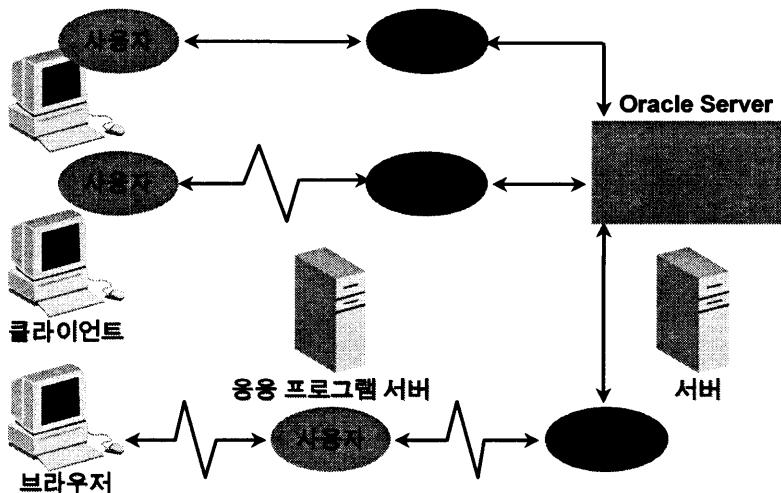
일부 백그라운드 프로세스, SGA 구조 및 데이터베이스 파일이 SQL 문을 처리하는 데 사용됩니다. 다음과 같은 SQL 문의 유형에 따라 서로 다른 구성 요소가 사용됩니다.

- 질의의 경우 사용자에게 행(row)을 반환하는 추가 처리가 필요합니다.
- DML(데이터 조작어)문의 경우 데이터의 변경 내용을 기록하는 추가 처리가 필요합니다.
- 커밋을 처리할 경우 트랜잭션에서 수정된 데이터가 복구될 수 있도록 보장해야 합니다.

일부 필수 백그라운드 프로세스는 SQL 문을 처리하는 데 직접 참여하지는 않지만 데이터베이스의 성능을 향상시키고 데이터베이스를 복구하는 데 사용됩니다.

선택적인 백그라운드 프로세스인 ARC0은 운영 중인 데이터베이스의 복구를 보장하기 위해 사용됩니다.

인스턴스에 연결



E-10

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

인스턴스 연결에 사용되는 프로세스

사용자가 SQL 문을 Oracle Server에 제출하려면 먼저 인스턴스에 연결해야 합니다.

사용자는 iSQL*Plus와 같은 툴을 시작하거나 Oracle Forms와 같은 툴을 사용하여 개발한 응용 프로그램을 실행합니다. 이러한 응용 프로그램이나 툴은 사용자 프로세스에서 실행됩니다.

가장 기본적인 구성에서는 사용자가 Oracle Server에 로그인하면 Oracle Server가 실행되고 있는 컴퓨터에서 프로세스가 생성됩니다. 이 프로세스를 서버 프로세스라고 하며, 이 프로세스는 클라이언트에서 실행되는 사용자 프로세스를 대신하여 오라클 인스턴스와 통신합니다. 서버 프로세스는 사용자 프로세스 대신 SQL 문을 실행합니다.

연결

연결은 사용자 프로세스와 Oracle Server 사이의 통신 경로입니다. 데이터베이스 사용자는 다음 세 가지 방법 중 하나를 사용하여 Oracle Server에 연결할 수 있습니다.

- 사용자는 오라클 인스턴스를 실행하고 있는 운영 체제에 로그인한 다음 해당 시스템의 데이터베이스에 액세스하는 응용 프로그램이나 툴을 시작합니다. 통신 경로는 호스트 운영 체제에서 사용할 수 있는 프로세스 간 통신 방법을 사용하여 설정됩니다.

인스턴스 연결에 사용되는 프로세스

연결(계속)

- 사용자는 로컬 컴퓨터에서 응용 프로그램이나 룰을 시작한 다음 네트워크를 통해 오라클 인스턴스를 실행하고 있는 컴퓨터에 연결합니다. 클라이언트-서버라고 불리는 이 구성에서는 사용자와 Oracle Server 간의 통신에 네트워크 소프트웨어가 사용됩니다.
- 3 계층 연결에서 사용자의 컴퓨터는 네트워크를 통해 응용 프로그램 또는 네트워크 서버와 통신하고, 이러한 응용 프로그램 또는 네트워크 서버는 다시 네트워크를 통해 오라클 인스턴스를 실행하고 있는 컴퓨터에 연결됩니다. 예를 들면, 사용자가 네트워크 컴퓨터에서 브라우저를 실행하여 NT 서버에 상주하는 응용 프로그램을 사용하고, 이 응용 프로그램은 UNIX 호스트에서 실행되고 있는 오라클 데이터베이스에서 데이터를 검색하는 경우입니다.

세션

세션은 사용자와 Oracle Server 사이의 특정 연결입니다. 세션은 Oracle Server에서 사용자를 검증할 때 시작되며 사용자가 로그아웃하거나 비정상적인 종료가 발생할 경우 끝납니다. 데이터베이스 사용자가 동시에 여러 룰, 응용 프로그램 또는 단말기에 로그인하면 동시에 여러 세션이 생길 수 있습니다. 일부 특정 데이터베이스 관리 룰 이외의 경우에는 데이터베이스 세션을 시작할 때 Oracle Server가 사용 가능한 상태여야 합니다.

참고: 여기서 설명한 것과 같이 사용자 프로세스와 서버 프로세스 사이에 1:1 대응이 성립하는 연결 유형을 전용 서버 연결이라고 합니다.

질의 처리

- **구문 분석(Parse):**
 - 동일한 명령문을 검색합니다.
 - 구문, 객체 이름 및 권한을 확인합니다.
 - 구문 분석하는 동안 사용되는 객체를 잠금니다.
 - 실행 계획을 생성하고 저장합니다.
- **실행(Execute):** 선택된 행(row)을 식별합니다.
- **인출(Fetch):** 사용자 프로세스에 행을 반환합니다.

ORACLE

E-12

Copyright © Oracle Corporation, 2001. All rights reserved.

질의 처리 단계

질의는 성공할 경우 데이터를 반환한다는 점에서 다른 유형의 SQL 문과 구분됩니다. 다른 명령문이 단순히 성공 또는 실패 여부를 반환하는 반면 질의는 하나의 행에서 수천 개의 행까지 반환할 수 있습니다.

질의를 처리할 때는 다음과 같은 기본적인 세 단계를 거칩니다.

- 구문 분석(Parse)
- 실행(Execute)
- 인출(Fetch)

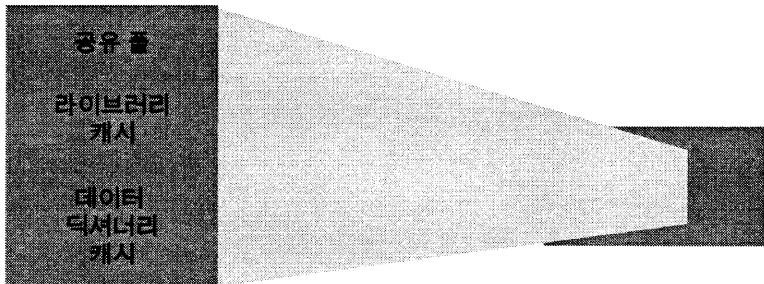
SQL 문 구문 분석

구문 분석 단계에서는 SQL 문이 사용자 프로세스에서 서버 프로세스로 전달되며 SQL 문의 구문 분석된 표현이 공유 SQL 영역에 로드됩니다.

구문을 분석하는 동안 서버 프로세스는 다음 기능을 수행합니다.

- 해당 SQL 문의 기존 복사본이 공유 풀에 있는지 검색합니다.
- SQL 문의 구문을 검증합니다.
- 데이터 딕셔너리를 검색하여 테이블 및 행(row) 정의를 검증합니다.

공유 풀



- 라이브러리 캐시에는 SQL 문 텍스트, 구문 분석된 코드 및 실행 계획이 포함됩니다.
- 데이터 딕셔너리 캐시에는 테이블, 열 및 다른 객체의 정의와 권한이 포함됩니다.
- 공유 풀의 크기는 `SHARED_POOL_SIZE`에 의해 지정 됩니다.

ORACLE

E-13

Copyright © Oracle Corporation, 2001. All rights reserved.

공유 풀 구성 요소

구문 분석 단계 동안 서버 프로세스는 공유 풀이라고 알려진 SGA 내 영역을 사용하여 SQL 문을 컴파일합니다. 공유 풀에는 다음과 같은 두 가지 기본 구성 요소가 있습니다.

- 라이브러리 캐시
- 데이터 딕셔너리 캐시

라이브러리 캐시

라이브러리 캐시는 가장 최근에 사용된 SQL 문에 대한 정보를 공유 SQL 영역이라는 메모리 구조에 저장합니다. 공유 SQL 영역에는 다음이 포함됩니다.

- SQL 문의 텍스트
- 구문 분석 트리: 명령문의 컴파일된 버전
- 실행 계획: 명령문을 실행할 단계

최적기는 최적의 실행 계획을 결정하는 Oracle Server 함수입니다.

SQL 문이 다시 실행될 경우 공유 SQL 영역에 이미 해당 명령문에 대한 실행 계획이 있으면 서버 프로세스가 이 명령문을 다시 구문 분석 할 필요가 없습니다. 라이브러리 캐시는 SQL 문을 다시 사용하는 응용 프로그램에 대해 구문 분석 시간 및 필요한 메모리를 감소시킴으로써 성능을 향상시켜 줍니다. SQL 문은 다시 사용되지 않을 경우 결국은 라이브러리 캐시에서 지워집니다.

공유 풀 구성 요소(계속)

데이터 딕셔너리 캐시

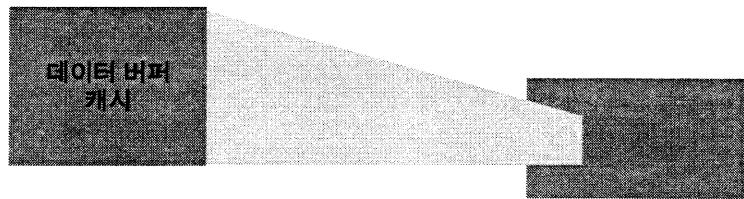
데이터 딕셔너리 캐시는 데이터 캐시 또는 행(row) 캐시라고도 하며 데이터베이스에서 가장 최근에 사용된 정의의 collection입니다. 여기에는 데이터베이스 파일, 테이블, 인덱스, 열, 사용자, 권한 및 다른 데이터베이스 객체 정보가 포함됩니다.

구문 분석 단계 동안 서버 프로세스는 딕셔너리 캐시에서 정보를 검색하여 SQL 문에 지정된 객체 이름을 분석하고 액세스 권한을 검증합니다. 필요한 경우 서버 프로세스가 데이터 파일에서 이 정보의 로드를 시작합니다.

공유 풀 크기 지정

공유 풀의 크기는 초기화 파라미터 SHARED_POOL_SIZE에 의해 지정됩니다.

데이터베이스 버퍼 캐시



- 가장 최근에 사용된 블록을 저장합니다.
- DB_BLOCK_SIZE에 따라 버퍼 크기를 조정합니다.
- 버퍼 수는 DB_BLOCK_BUFFERS에 의해 정의됩니다.

ORACLE

데이터베이스 버퍼 캐시의 기능

질의가 처리될 때 서버 프로세스는 데이터베이스 버퍼 캐시에서 필요 한 블록을 검색합니다. 해당 블록이 데이터베이스 버퍼 캐시에 없으면서 서버 프로세스는 해당 블록을 데이터 파일에서 읽어 버퍼 캐시에 복사합니다. 이후에 동일한 블록이 요청될 경우 메모리에서 해당 블록을 찾을 수 있으므로 물리적으로 읽을 필요가 없습니다. Oracle Server는 LRU(least Recently Used) 알고리즘을 사용하여 최근에 액세스된 적이 없는 버퍼를 지움으로써 버퍼 캐시에 새로운 블록을 위한 공간을 확보합니다.

데이터베이스 버퍼 캐시 크기 지정

버퍼 캐시에 있는 각 버퍼의 크기는 오라클 블록의 크기와 같으며, 이 크기는 DB_BLOCK_SIZE 파라미터에 의해 지정됩니다. 버퍼의 수는 DB_BLOCK_BUFFERS 파라미터의 값과 같습니다.

프로그램 글로벌 영역(PGA)

- 공유되지 않습니다.
- 서버 프로세서에서만 쓸 수 있습니다.
- 다음을 포함합니다.
 - 정렬 영역
 - 세션 정보
 - 커서 상태
 - 스택 공간



ORACLE

E-16

Copyright © Oracle Corporation, 2001. All rights reserved.

프로그램 글로벌 영역 구성 요소

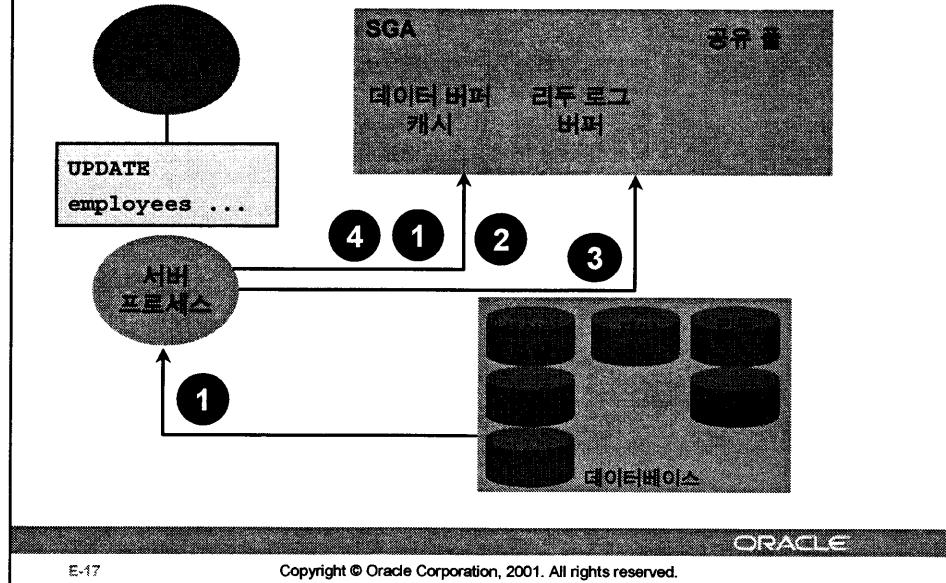
프로그램 글로벌 영역(PGA)은 서버 프로세스에 대한 데이터 및 제어 정보를 포함하는 메모리 영역입니다. 이 영역은 비공유 메모리로서 서버 프로세스가 시작될 때 오라클에서 생성합니다. 해당 서버 프로세스는 이 영역에 액세스할 수 있으며 Oracle Server 코드가 서버 프로세스를 대신하여 이 영역을 읽고 씁니다. 오라클 인스턴스에 연결된 각 서버 프로세스에 할당된 PGA 메모리를 해당 인스턴스에 할당된 총 PGA 메모리라고 합니다.

전용 서버 구성에서 서버의 PGA에는 다음 구성 요소가 포함됩니다.

- 정렬 영역: SQL 문을 처리하는 데 필요한 정렬에 사용됩니다.
- 세션 정보: 세션에 대한 사용자 권한 및 성능 통계를 포함합니다.
- 커서 상태: 세션에서 현재 사용하고 있는 SQL 문의 처리 단계를 나타냅니다.
- 스택 공간: 다른 세션 변수를 포함합니다.

PGA는 프로세스가 생성될 때 할당되고 프로세스가 종료될 때 해제됩니다.

DML 문 처리



DML 처리 단계

DML(데이터 조작어) 문은 다음과 같이 두 단계에 걸쳐 처리됩니다.

- 구문 분석(Parse) 단계는 질의 처리에 사용되는 구문 분석 단계와 같습니다.
- 실행(Execute) 단계에서는 데이터 변경 사항을 추가로 처리해야 합니다.

DML 실행 단계

DML 문을 실행하려면:

- 해당 데이터 및 블록이 버퍼 캐시에 없으면 서버 프로세스가 데이터 파일에서 버퍼 캐시로 이들을 읽어옵니다.
- 수정될 행(row)을 서버 프로세스가 잠금니다.
- 서버 프로세스는 블록 및 데이터에 대한 변경 내용을 리두 로그 버퍼에서 기록합니다.
- 블록 변경 내용에는 수정되기 전의 데이터 값이 기록됩니다. 블록은 필요할 경우 DML 문을 블록할 수 있도록 데이터의 이전 이미지를 저장하는 데 사용됩니다.
- 데이터 블록 변경 내용에는 데이터의 새 값이 기록됩니다.

DML 처리 단계

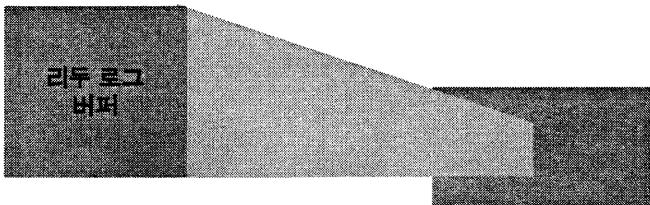
DML 실행 단계(계속)

서버 프로세스는 블록을 블록하기 위한 이전 이미지를 기록하고 데이터 블록을 갱신합니다. 이러한 변경 내용은 모두 데이터베이스 버퍼 캐시에서 수행됩니다. 버퍼 캐시에서 변경된 블록은 더티(dirty) 버퍼로 표시됩니다. 즉, 버퍼가 디스크의 해당 블록과 다르다는 의미입니다.

DELETE 또는 INSERT 명령 처리도 비슷한 단계를 거칩니다. DELETE에 대한 이전 이미지에는 삭제된 행(row)의 열 값이 포함되며 INSERT에 대한 이전 이미지에는 행 위치 정보가 포함됩니다.

블록에 대한 변경 내용은 메모리 구조에만 기록되고 디스크에 즉시 쓰여지지 않으므로 컴퓨터가 고장나면 SGA가 손실될 뿐 아니라 이러한 변경 내용도 손실됩니다.

리두 로그 버퍼



- **LOG_BUFFER**에 의해 크기가 정의됩니다.
- 인스턴스를 통해 변경된 내용을 기록합니다.
- 순차적으로 사용됩니다.
- 순환 버퍼입니다.

ORACLE

E-19

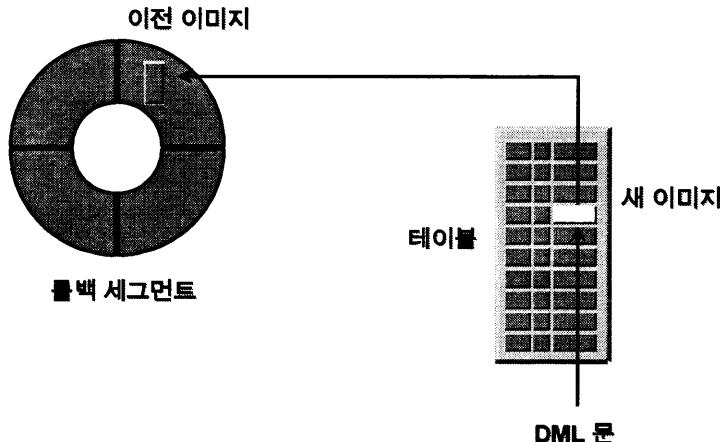
Copyright © Oracle Corporation, 2001. All rights reserved.

리두 로그 버퍼의 특징

서버 프로세스는 데이터 파일 블록에 대한 변경 내용 대부분을 SGA의 일부인 리두 로그 버퍼에 기록합니다. 리두 로그 버퍼는 다음과 같은 특성이 있습니다.

- **LOG_BUFFER** 파라미터에 의해 크기(바이트 단위)가 정의됩니다.
- 변경된 블록, 변경된 위치 및 새 값이 리두 항목에 기록됩니다. 리두 항목은 변경된 블록 유형을 구별하지 않고 단순히 해당 블록에서 변경된 바이트를 기록합니다.
- 리두 로그 버퍼는 순차적으로 사용되므로 서로 다른 트랜잭션에 의해 변경된 내용이 섞여 있을 수 있습니다.
- 리두 로그 버퍼는 가득 찰 경우 다시 사용되는 순환 버퍼입니다. 그러나 이전 리두 항목이 리두 로그 파일에 기록되어야만 다시 사용할 수 있습니다.

롤백 세그먼트



롤백 세그먼트

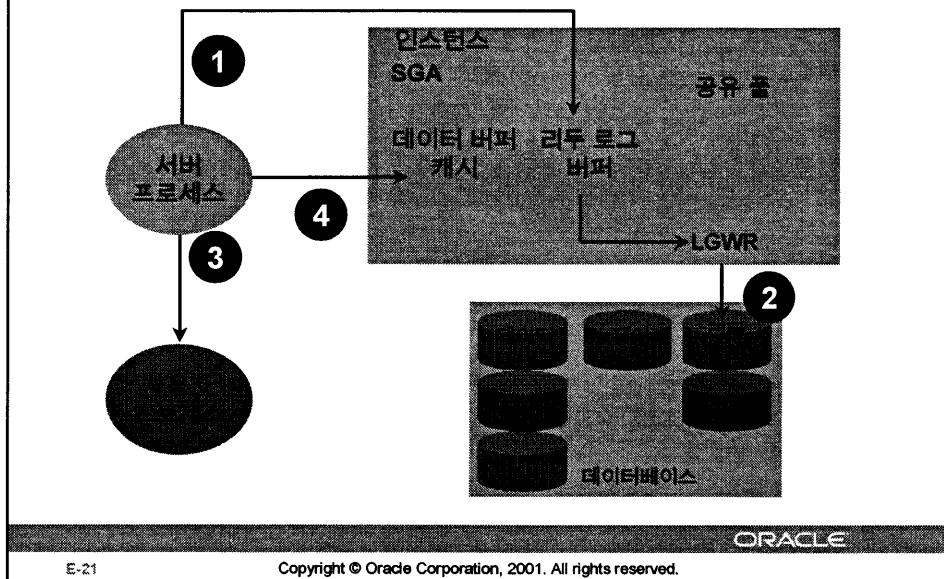
서버 프로세스는 데이터를 변경하기 전에 이전 데이터 값을 롤백 세그먼트에 저장합니다. 이러한 이전 이미지는 다음 작업에 사용됩니다.

- 트랜잭션이 롤백될 경우 변경 내용을 취소합니다.
- 커밋되지 않은 DML 문의 변경 내용을 다른 트랜잭션에서 보지 못하도록 함으로써 읽기 일관성을 제공합니다.
- 작업이 실패한 경우 데이터베이스를 일관성 있는 상태로 복구합니다.

롤백 세그먼트는 테이블 및 인덱스처럼 파일에 존재하며 롤백 블록은 필요에 따라 데이터베이스 베패 캐시로 읽혀집니다. 롤백 세그먼트는 DBA가 생성합니다.

롤백 세그먼트에 대한 변경 내용은 리두 로그 베패에 기록됩니다.

COMMIT 처리



E-21

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

빠른 COMMIT

Oracle Server는 빠른 커밋 방식을 사용하여 인스턴스 실패 시 커밋된 변경 내용이 복구될 수 있도록 합니다.

시스템 변경 번호

트랜잭션이 커밋할 때마다 Oracle Server는 커밋 시스템 변경 번호(SCN)를 트랜잭션에 할당합니다. SCN은 하나씩 증가하며 데이터베이스 내에서 고유합니다. Oracle Server는 SCN을 내부 시간 기록으로 사용하여 데이터 파일에서 데이터가 검색될 때 데이터를 동기화하고 읽기 일관성을 제공합니다. SCN을 사용하면 운영 체제의 날짜 및 시간에 구애받지 않고 Oracle Server에서 일관성 검사를 수행할 수 있습니다.

COMMIT 처리 단계

COMMIT을 실행하면 다음 단계가 수행됩니다.

- 서버 프로세스가 커밋 레코드와 SCN을 리두 로그 버퍼로 가져옵니다.
- LGWR는 커밋 레코드를 포함하여 모든 리두 로그 버퍼 항목을 연속적으로 리두 로그 파일에 씁니다. 이 작업이 이루어진 이후에는 Oracle Server에서 인스턴스가 실패해도 변경 내용이 손실되지 않습니다.

빠른 COMMIT

COMMIT 처리 단계(계속)

- 사용자에게 COMMIT이 완료되었음을 알립니다.
- 서버 프로세스는 트랜잭션이 완료되었고 자원 잠금을 해제할 수 있다는 것을 나타내는 정보를 기록합니다.

더티 버퍼의 내용을 데이터 파일에 쓰는 작업은 DBW0에 의해 독립적으로 수행되며 커밋 이전이나 이후에 발생할 수 있습니다.

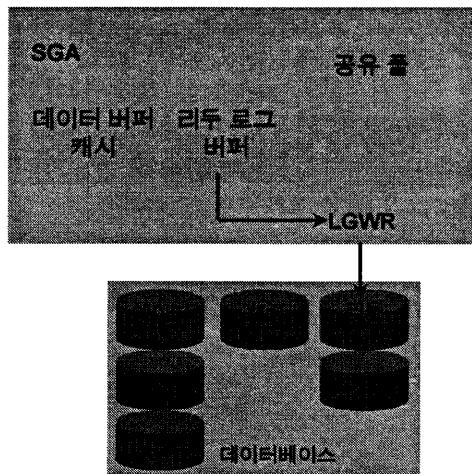
빠른 COMMIT의 장점

빠른 커밋 방식은 변경 내용을 데이터 파일 대신 리두 로그 버퍼에 쓰임으로써 데이터를 복구를 보장해 줍니다. 이런 방식은 다음과 같은 장점이 있습니다.

- 로그 파일에 순차적으로 쓰면 데이터 파일의 서로 다른 블록에 쓰는 것보다 속도가 훨씬 빠릅니다.
- 데이터 파일에 쓰려면 데이터의 전체 블록을 써야 하지만 로그 파일에는 변경 내용을 기록하는데 필요한 최소한의 정보만 쓰여집니다.
- 여러 트랜잭션이 동시에 커밋을 요청하는 경우 인스턴스는 리두 로그 레코드를 단일 쓰기(piggyback)로 처리합니다.
- 리두 로그 버퍼가 특별히 가득 찬 경우가 아니라면 트랜잭션 당 한 번의 동기 쓰기만 수행하면 됩니다. 단일 쓰기(piggyback)로 처리하면 트랜잭션 당 동기 쓰기를 반드시 한 번씩 수행할 필요가 없어지므로 동기 쓰기가 한 번 이하로 수행될 수 있습니다.
- 리두 로그 버퍼가 COMMIT 전에 비워지므로 트랜잭션 크기는 COMMIT 작업에 필요한 실제 시간에 영향을 미치지 않습니다.

참고: 트랜잭션 를백이 LGWR로 하여금 디스크에 쓰기를 시작하도록 만드는 것은 아닙니다. Oracle Server는 실패를 복구할 때 커밋되지 않은 변경 내용을 항상 를백합니다. 를백한 후 를백 항목이 디스크에 기록되기 전에 실패가 발생하면 커밋 레코드가 없으므로 트랜잭션으로 인한 변경 내용이 절로 를백됩니다.

로그 기록자(LGWR)



LGWR는 다음과 같은 경우에 쓰기를 수행합니다.

- COMMIT이 발생한 경우
- 리두 버퍼 로그의 1/3이 채워진 경우
- 리두가 1MB 이상인 경우
- DBW0가 쓰기 전에

ORACLE

로그 기록자

LGWR는 다음과 같은 경우 리두 로그 버퍼에서 리두 로그 파일로 순차 쓰기를 수행합니다.

- 트랜잭션이 커밋한 경우
- 리두 버퍼 로그의 1/3이 채워진 경우
- 리두 로그 버퍼에 기록된 변경 내용이 1MB 이상인 경우
- DBW0가 데이터베이스 버퍼 캐시에 있는 수정된 블록을 데이터 파일에 쓰기 전에

리두는 복구에 필요하므로 LGWR는 리두가 디스크에 쓰여진 다음에야 COMMIT를 확정합니다.

기타 인스턴스 프로세스

- **기타 필수 프로세스**
 - 데이터베이스 기록자(DBW0)
 - 프로세스 모니터(PMON)
 - 시스템 모니터(SMON)
 - 체크포인트(CKPT)
- **아카이브 프로세스(ARC0)**는 주로 운용 중인 데이터베이스에서 생성됩니다.

ORACLE

기타 필수 프로세스

다음과 같은 기타 필수 프로세스 네 가지는 SQL 문 처리에 직접 참여하지 않습니다.

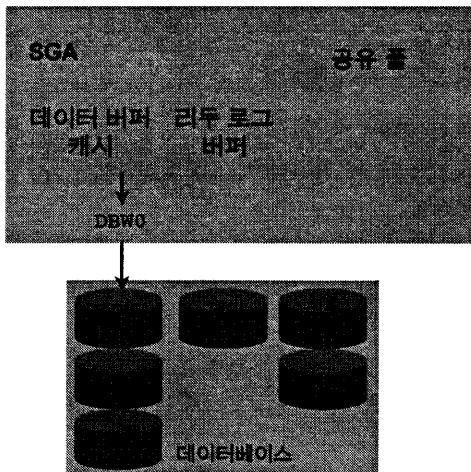
- 데이터베이스 기록자(DBW0)
- 프로세스 모니터(PMON)
- 시스템 모니터(SMON)
- 체크포인트(CKPT)

체크포인트 프로세스는 데이터베이스 파일을 동기화하는 데 사용됩니다.

아카이브 프로세스

필수 프로세스 이외의 백그라운드 프로세스는 모두 데이터베이스 구성에 따라 선택적이지만 ARC0는 디스크가 손실된 경우 데이터베이스 복구 작업에서 매우 중요한 역할을 합니다. ARC0 프로세스는 주로 운용 중인 데이터베이스에서 생성됩니다.

데이터베이스 기록자(DBW0)



DBW0는 다음과 같은 경우에 쓰기를 수행합니다.

- 더티 버퍼가 많은 경우
- 사용 가능한 버퍼가 적은 경우
- 시간 초과가 발생한 경우
- 체크포인트가 발생한 경우

데이터베이스 기록자

서버 프로세스는 룰백 및 데이터 블록에 대한 변경 내용을 버퍼 캐시에 기록합니다. 데이터베이스 기록자(DBW0)는 데이터베이스 버퍼 캐시의 더티 버퍼를 데이터 파일에 씁니다. 그러면 데이터베이스 버퍼 캐시에서 충분한 수의 버퍼(서버 프로세스가 데이터 파일에서 블록을 읽어야 할 때 덮어쓸 수 있는 버퍼)를 사용할 수 있게 됩니다. 서버 프로세스는 버퍼 캐시에서만 데이터를 변경하고 DBW0는 다음 이벤트 중 하나가 발생해야만 데이터 파일에 쓰기를 시작하므로 데이터베이스 성능이 좋아집니다.

- 더티 버퍼의 수가 임계값에 도달한 경우
- 프로세스가 사용 가능한 버퍼를 스캔할 때 특정 개수의 블록을 스캔하고도 사용 가능한 버퍼를 찾지 못한 경우
- 시간 초과가 발생한 경우(3초 간격)
- 체크포인트가 발생한 경우(체크포인트는 데이터베이스 버퍼 캐시를 데이터 파일과 동기화하는 수단임)

SMON: 시스템 모니터

- 다음과 같이 인스턴스를 자동 복구합니다.
 - 리두 로그의 변경 내용을 룰포워드합니다.
 - 사용자가 액세스할 수 있도록 데이터베이스를 업니다.
 - 커밋되지 않은 트랜잭션을 롤백합니다.
- 사용 가능 영역을 합칩니다.
- 임시 세그먼트의 할당을 해제합니다.

ORACLE

E-26

Copyright © Oracle Corporation, 2001. All rights reserved.

SMON: 시스템 모니터

오라클 인스턴스가 실패하면 디스크에 쓰여지지 않은 SGA 정보는 손실됩니다. 예를 들어, 운영 체계가 실패하면 인스턴스도 실패합니다. 인스턴스가 손실된 후 데이터베이스가 다시 열리면 SMON은 자동으로 인스턴스를 복구합니다. 인스턴스 복구 단계는 다음과 같습니다.

- 룰포워드를 수행하여 데이터 파일에는 기록되지 않고 온라인 리두 로그에만 기록된 데이터를 복구합니다. 이러한 데이터는 인스턴스가 실패하여 SGA가 손실되었으므로 디스크에 쓰여지지 않은 것입니다. 룰포워드 프로세스 동안 SMON은 리두 로그 파일을 읽어 리두 로그에 기록된 변경 내용을 데이터 블록에 적용합니다. 커밋된 트랜잭션은 모두 리두 로그에 기록되어 있으므로 이 프로세스를 통해 이들 트랜잭션을 완벽하게 복구할 수 있습니다.
- 사용자가 로그인할 수 있도록 데이터베이스가 열립니다. 복구되지 않은 트랜잭션에 의해 잠긴 데이터는 즉시 사용할 수 있습니다.
- 커밋되지 않은 트랜잭션이 롤백됩니다. 이들 트랜잭션이 잠긴 데이터에 액세스할 때 SMON 또는 개별 서버 프로세스에서 이들 트랜잭션을 롤백합니다.

SMON은 일부 영역 관리 기능도 수행합니다.

- 데이터 파일에서 인접해 있는 사용 가능 영역을 합칩니다.
- 임시 세그먼트의 할당을 해제하여 데이터 파일의 사용 가능 영역으로 반환합니다.
임시 세그먼트는 SQL 문을 처리하는 동안 데이터를 저장하는 데 사용됩니다.

PMON: 프로세스 모니터

프로세스가 실패하면 다음과 같이 정리합니다.

- 트랜잭션 롤백
- 잠금 해제
- 기타 자원 해제

ORACLE

E-27

Copyright © Oracle Corporation, 2001. All rights reserved.

PMON의 기능

백그라운드 프로세스 PMON은 프로세스가 실패할 경우 다음과 같이 정리합니다.

- 사용자의 현재 트랜잭션을 롤백
- 현재 사용 중인 테이블 또는 행(row) 잠금을 모두 해제
- 사용자가 현재 예약해 둔 기타 자원 해제

요약

이 부록에서 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- 데이터베이스 파일 식별: 데이터 파일, 제어 파일, 온라인 리두 로그
- SGA 메모리 구조 설명: DB 버퍼 캐시, 공유 SQL 풀 및 리두 로그 버퍼
- 기본 백그라운드 프로세스 설명: DBW0, LGWR, CKPT, PMON, SMON 및 ARC0
- SQL 처리 단계 설명: 구문 분석(Parse), 실행(Execute), 인출(Fetch)

ORACLE

요약

오라클 데이터베이스에는 다음과 같은 파일이 있습니다.

- 제어 파일: 데이터베이스에 있는 다른 파일 이름을 포함하여 데이터베이스 무결성을 확인하는 데 필요한 정보가 들어 있습니다.(제어 파일은 일반적으로 이중화됩니다.)
- 데이터 파일: 테이블, 인덱스, 룰백 세그먼트 및 임시 세그먼트를 포함한 데이터베이스의 데이터가 들어 있습니다.
- 온라인 리두 로그: 데이터 파일에 대한 변경 내용이 들어 있습니다.(온라인 리두 로그는 복구 작업에 사용되며 일반적으로 이중화됩니다.)

데이터베이스에서 일반적으로 사용되는 기타 파일은 다음과 같습니다.

- 파라미터 파일: 오라클 인스턴스의 특성을 정의합니다.
- 암호 파일: 권한이 있는 데이터베이스 사용자를 인증합니다.
- 아카이브된 리두 로그: 온라인 리두 로그의 백업입니다.

SGA 메모리 구조

시스템 글로벌 영역(SGA)은 다음 세 가지 기본 구조로 이루어집니다.

- 공유 풀: 가장 최근에 실행된 SQL 문 및 데이터 덕셔너리 중 가장 최근에 사용된 데이터를 저장합니다.
- 데이터베이스 버퍼 캐시: 가장 최근에 사용된 데이터를 저장합니다.
- 리두 로그 버퍼: 인스턴스를 통해 데이터베이스에 대해 변경한 내용을 기록합니다.

백그라운드 프로세스

운용 중인 오라클 인스턴스에는 다음과 같은 프로세스가 포함되어 있습니다.

- 데이터베이스 기록자(DBW0): 변경된 데이터를 데이터 파일에 씁니다.
- 로그 기록자(LGWR): 데이터 파일에 대한 변경 내용을 온라인 리두 로그 파일에 기록합니다.
- 시스템 모니터(SMON): 데이터베이스가 열릴 때 데이터베이스의 일관성을 검사하고 복구를 시작합니다.
- 프로세스 모니터(PMON): 프로세스 중 하나가 실패하면 자원을 정리합니다.
- 체크포인트(CKPT): 체크포인트 후 데이터베이스 상태 정보를 갱신합니다.
- 아카이버(ARCO): 온라인 리두 로그를 백업하여 매체 고장 시 복구할 수 있도록 합니다.
(이 프로세스는 선택적이지만 일반적으로 운영 중인 인스턴스에 포함됩니다.)

구성에 따라 인스턴스에 다른 프로세스가 포함될 수도 있습니다.

SQL 문 처리 단계

SQL 문의 처리 단계는 다음과 같습니다.

- 구문 분석(Parse): SQL 문을 컴파일합니다.
- 실행(Execute): 선택된 행(row)을 식별하거나 DML 변경 내용을 데이터에 적용합니다.
- 인출(Fetch): SELECT 문에서 질의한 행을 반환합니다.

인덱스

참고: 굵게 표시된 문자 또는 숫자는 전체 단원 또는 부록을 나타냅니다.

숫자

2000년 준수 3-17

3-Way 조인 4-30

A

ACCESS PARAMETER 20-19

ADD_MONTHS 기능 3-21

ALL INSERT(조건) 20-7

ALL 연산자 6-16

ALL_COL_COMMENT 데이터 딕셔너리 뷰 9-32

ALL_TAB_COMMENT 데이터 딕셔너리 뷰 9-32

ALTER SEQUENCE 문 12-12

ALTER TABLE 문 9-22, 9-23, 10-17, 10-20, 10-21, 13-11

ALTER USER 문 13-11

American National Standards Institute I-24

ANSI I-24

ANY 연산자 6-15

APPEND 명령 C-11

AS Subquery 절 9-20

AVG 함수 5-6, 5-7

B

BETWEEN 연산자 2-10

BREAK 명령 7-18

BTITLE 명령 7-19

C

CASCADE CONSTRAINTS 절 10-22

CASE 표현식 3-51, 3-52, 18-12

CHANGE 명령 C-12

C

CHECK 제약 조건 10-16
CLEAR BREAK 명령 7-18
COALESCE 함수 3-49
COLUMN 명령 7-16, 7-17
COMMENT 문 9-32
COMMIT 문 8-2, 8-33, 8-35, 8-39, 8-40, 9-8
CONCAT 함수 3-11
CONNECT BY 절 19-5, 19-7, 19-13
COUNT 함수 5-8
CREATE DATABASE 문 16-9
CREATE DIRECTORY 문 20-20
CREATE INDEX 문 12-17, 20-24
CREATE SEQUENCE 문 12-5
CREATE TABLE 문 9
CREATE USER 문 13-6
CREATE VIEW 문 11-7
CUBE 연산자 17-2, 17-6, 17-9
CURRENT_DATE 함수 16-6
CURRENT_TIMESTAMP 함수 16-7
CURRVAL 9-7, 12-8
CYCLE 절(시퀀스) 12-6

D

Datetime 데이터 유형 9-14
Datetime 함수 16-2
DBTIMEZONE 함수 16-9
DCL(데이터 제어어) 문 8-33, 9
DDL(데이터 정의어) 문 8-33, 9-5, 13

D

DECODE 표현식 3-51, 3-54
DEFAULT DIRECTORY 20-19
DEFAULT 절 8-26, 8-27, 9-7
DEFINE 명령 7-5, 7-11
DELETE 문 8-19, 8-20, 13-16
DESCRIBE 명령 1-29, 8-7, 10-24, 11-13, C-7
DISABLE 절 10-20
DISTINCT 키워드 1-4, 1-23, 5-5, 5-10
DML(데이터 조작어) 문 8
 뷰를 통한 DML 작업 11-14
DROP ANY INDEX 문 12-2
DROP ANY VIEW 문 11-20
DROP COLUMN 절 9-27
DROP INDEX 문 12
DROP SEQUENCE 문 12-14
DROP SYNONYM 12-24
DROP TABLE 문 9-29
DROP UNUSED COLUMNS 절 9-28
DROP VIEW 문 11-20
DUAL 테이블 3-14, 3-18

E

EDIT 명령 C-14
ESCAPE 옵션 2-13
Execute 버튼(iSQL*Plus에서) 1-7, 1-32
EXISTS 연산자 18-18, 18-19
EXTRACT 함수 16-10

F

FOREIGN KEY 제약 조건 10-13, 10-14, 10-15, I-19

FRACTIONAL_SECONDS_PRECISION 9-15

FROM 절 1

FROM 절 질의 11-21, 18-2, 18-10

FROM_TZ 함수 16-11

G

GRANT 문 13

GROUP BY ROLLUP 17-17

GROUP BY 절 5-13, 5-14, 5-15, 5-16, 17-3, 17-4

GROUPING SETS 절 17-12, 17-11, 17-13

H

HAVING 절 5-21, 5-22, 5-23, 6-11, 17-5

I

If-Then-Else 논리 3-51

INITCAP 함수 3-9

INSERT 문 8-5, 8-6, 8-11, 13-18, 20-2, 20-7

VALUES 절 8-5

무조건 INSERT 20-7, 20-10

조건 FIRST INSERT 20-7, 20-13, 20-14

조건 INSERT ALL 20-7, 20-11

피벗 INSERT 20-7, 20-15

INSTR 함수 3-11

INTERSECT 연산자 15-12

INTERVAL YEAR TO MONTH 데이터 유형 9-17

IS NOT NULL 연산자 2-14

IS NULL 연산자 2-14

iSQL*Plus 1-24

ISO(International Standards Organization) I-24

J

Java I-23

L

LAST_DAY 함수 3-21
LENGTH 함수 3-11
LEVEL 의사 열 19-10
LIKE 연산자 2-12
LIST 명령 C-11
LOCALTIMESTAMP 함수 16-8
LOWER 함수 3-9
LPAD 함수 3-11

M

MAX 함수 5-6, 5-7
MERGE 문 8-28, 8-29
WHEN NOT MATCHED 절 8-31
MIN 함수 5-6, 5-7
MINUS 연산자 15-14
MODIFY 절 9-26
MONTHS_BETWEEN 함수 3-6, 3-21

N

NEXT_DAY 함수 3-21
NEXTVAL 의사 열 9-7, 12-8
NOT EXISTS 연산자 18-20
NOT IN 연산자 18-20
NOT NULL 제약 조건 10-7
NULL 1-14, 1-15, 2-14, I-19
NULLIF 함수 3-48
Number 함수 3-13
NVL 함수 3-45, 3-46, 5-5, 5-12
NVL2 함수 3-47

O

ON DELETE CASCADE 절 10-15
ON DELETE SET NULL 절 10-15
ON 절 4-28, 4-29
OR REPLACE 절 11-12
Oracle9i Application Server I-4
Oracle9i Database I-4
ORDER BY 절 2, 15-20
 기본 정렬 순서 2-23
ORDER BY 절을 사용한 정렬 결과 2
 기본 정렬 순서 2-23
ORGANIZATION EXTERNAL 절 20-18, 20-19

P

PRIOR 절 19-7
PUBLIC 키워드 13-5

R

READ ONLY 제약 조건 11-19
REFERENCE 제약 조건 10-13, 10-15
REJECT LIMIT 절 20-19
REM 명령 7-21
RENAME 명령 9-28
REVOKE 명령 13-17
ROLLBACK 문 8-2, 8-33, 8-35, 8-38, 8-41, E-20
ROLLUP 절 17-2, 17-6, 17-7, 17-8
ROUND 함수 3-14, 3-21, 3-23
ROWNUMBER 의사 열
RR 날짜 형식 3-41

S

SAVE 명령 C-14
SAVEPOINT 문 8-2, 8-35, 8-36
SELECT 문 1
Server 구조 E-2
SESSIONTIMEZONE 함수 16-9
SET TIME_ZONE 절 16-9
SET UNUSED 절 9-28
SET VERIFY ON 명령 7-7
SET 명령 7-12
SET 연산자 15-2, 15-3
SET 절 8-15
SMON 프로세스 E-8
SOME 연산자 6-15
SQL 버퍼 C-3
SQL 스크립트 D-2
SQL 실행 1-26
SQL(구조적 질의어) I-2, I-21, I-22, 1-2, 1-24, 1-25
SQL*Plus C
SQL*Plus 명령 C-2
SQL*Plus 스크립트 파일 7-3
SQL: 1999 준수 4-6, 4-22, 4-30
START WITH 절 19-5, 19-6
START 명령 C-14
STDDEV 함수 5-7
SUBSTR 함수 3-11
SUM 함수 5-6, 5-7
SYS 함수 9-9
SYSDATE 함수 3-18, 3-20, 9-7

T

TIMESTAMP 데이터 유형 9-16

INTERVAL YEAR TO MONTH 9-17

TIMESTAMP WITH LOCAL TIME 9-16

TIMESTAMP WITH TIME ZONE 9-15

TIMEZONE_ABBR 16-10

TIMEZONE_REGION 16-10

TO_CHAR 함수 3-31, 3-37, 3-39

TO_DATE 함수 3-39

TO_NUMBER 함수 3-39

TO_TIMESTAMP 함수 16-12

TO_YMINTERVAL 함수 16-13

Top-n 분석 11-2, 11-22, 11-23, 11-24

TRIM 함수 3-11

TRUNC 함수 3-15, 3-21, 3-23

TRUNCATE TABLE 문 9-31

TTITLE 명령 7-19

TYPE ACCESS_DRIVER_TYPE 20-19

TZOFFSET 함수 16-14

U

UNDEFINE 명령 7-11

UNION 연산자 15-10, 15-11

UNION 연산자 15-7, 15-8, 15-11

UNIQUE 제약 조건 10-9, 10-10

UPDATE 문 8, 13-14

SET 절 8-15

상관 UPDATE 18-22

UPPER 함수 3-9, 3-10

USER 함수 9-7

USER_CATALOG 딕셔너리 뷰 9-10

U

USER_COL_COMMENTS 딕셔너리 뷰 9-32
USER_COL_PRIVS_MADE 딕셔너리 뷰 D-4
USER_CONS_COLUMNS 딕셔너리 뷰 10-19, 10-25
USER_CONSTRAINTS 딕셔너리 뷰 10-4, 10-19, 10-24
USER_DB_LINKS 딕셔너리 뷰 13-19
USER_INDEXES 딕셔너리 뷰 12-20
USER_OBJECTS 딕셔너리 뷰 9-10, D-4
USER_SEQUENCES 딕셔너리 뷰 12-7
USER_TAB_COMMENTS 딕셔너리 뷰 9-30
USER_TAB_PRIVS_MADE 딕셔너리 뷰 D-4
USER_TABLES 딕셔너리 뷰 9-10, D-4
USER_UNUSED_COL_TABS 딕셔너리 뷰 9-28
USING 절 4-26, 13-20

V

V\$TIMEZONE_NAME 딕셔너리 뷰 16-11
VALUES 절 8-5
VERIFY 명령 7-7

W

WHEN NOT MATCHED 절 8-31
WHERE 절 2
 행(row) 제한 2-2
WITH CHECK OPTION 절 8-25, 11-17, 13-13, 13-14, 13-15, 13-18
WITH READ ONLY 절 11-18
WITH 절 18-2, 18-26

X

XML I-23

ㄱ

값 반환 3-3

객체 관계형 데이터베이스 관리 시스템(ORDBMS) I-2, I-7, I-12

객체 권한 13-2

객체 지향 프로그래밍 I-7

검색 2-12

계층 질의 19

CONNECT BY 절 19-5, 19-7, 19-13

PRIOR 절 19-7

START WITH 절 19-5, 19-6

자식 노드 19-10

트리 제거 19-13

고유 번호 생성 12-3

고유 식별자 I-18

고유 인덱스 10-10, 12-6

고유하지 않은 인덱스 12-16

공유 SQL 영역 E-14

공유 글로벌 영역 I-23, E-7

관계 I-16

관계형 데이터베이스 관리 시스템(RDBMS) I-2, I-13, I-14

교차 도표화 값 17-10

교차 도표화 행 17-6

교차 표 형식 보고서 17-9

권한 13

객체 권한 13-2

권한 할당 13-7

그레고리력(Gregorian Calendar) 16-10

그룹 함수 5

그룹 함수와 NULL 값 5-11

ㄱ

- 그리니치 표준시(GMT: Greenwich Mean Time) 16-3
- 기본 날짜 표시 2-6, 3-17
- 기본 정렬 순서 2-23
- 기본 키 10-11
- 기수(Cardinality) I-18

ㄴ

- 날짜 변환 함수 3-4, 3-35
- 날짜 함수 3-6
- 내부 질의 6-3, 6-4, 6-5, 18-5
- 논리 조건 2-15
- 논리적 부분 집합 11-4

ㄷ

- 다중 테이블 삽입 20-2, 20-5, 20-7
- 단일 앤퍼샌드 치환 7-4
- 단일 행(row) 서브 쿼리 6-2, 6-7
- 단일 행(row) 연산자 6-8
- 단일 행(row) 함수 3-4
- 대체 문자 2-12
- 데이터 그룹화 5, 17-2
- 데이터 딕셔너리 캐시 E-13, E-14
- 데이터 딕셔너리 테이블 9-9, D-3
- 데이터 웨어하우스 응용 프로그램 I-8
- 데이터 유형 3-25
- 데이터 파일 E-5
- 데이터베이스 기록자 E-8
- 데이터베이스 링크 13-19
- 데이터베이스 액세스 제어 13
- 동의어 9-3, 12-2, 12-3, 12-23, 13-3
- 동가 조인 4-8, 4-27

ㄹ

라이브러리 캐시 E-13
로그 기록자(LGWR) E-6, E-8
롤백 세그먼트 E-20
루트 노드 19-10
리터럴 값 1-20

ㅁ

명령 혹은 스크립트 파일 7-20
명령문 레벨 롤백 8-42
명시적(explicit) 데이터 유형 변환 3-25
모호한 열 이름 4-11
무결성 제약 조건 8-17, 10-2
문 1-4
문자열 2-5, 2-6

ㅂ

배타적 잠금(exclusive lock) 8-46
백그라운드 프로세스 E-3, E-7
별칭 1-4, 1-17, 1-16, 2-7, 2-24, 11-9
 테이블 별칭 4-12
부모-자식 관계 19-4
부분 집합, 논리적 11-4
분산 5-7
뷰 9-3, 11
 OR REPLACE 절 11-12
 USING 절 4-26
 WITH READ ONLY 절 11-18
 단순 및 복합 11-6
 뷰 생성 지침 11-8
 뷰에서 데이터 검색 11-10
 인라인 뷰 11-2, 11-21

H

- 뷰 생성 지침 11-8
- 뷰를 통한 데이터 추가 11-16
- 뷰에서 데이터 검색 11-10
- 비교 연산자, 비교 조건 2-7, 18-4
- 비등가 조인 4-14, 4-15
- 비쌍 비교 18-7

I

- 사용자 - (작성) 13-6
- 사용자 프로세스 E-10
- 산술 연산자 1-9
- 산술식 1-9
- 상관 18-17
- 상관 UPDATE 18-22
- 상관 서브 쿼리 18-2, 18-13, 18-14, 18-15, 18-21, 18-24
- 상위 집계 행 17-7, 17-8, 17-9
- 서브 쿼리 6, 8-16, 8-21, 8-23, 9-18, 11-21, 18-2, 18-3, 18-10
 - AS Subquery 절 9-20
 - FROM 절 질의 11-21, 18-2, 18-10
 - 내부 질의 6-3, 6-4, 6-5, 18-5
 - 단일 행(row) 서브 쿼리 6-2, 6-7
 - 상관 UPDATE 18-22
 - 상관 서브 쿼리 18-2, 18-13, 18-14, 18-15, 18-21, 18-24
 - 서브 쿼리 배치 6-4
 - 서브 쿼리에서 반환되는 열 없음 6-13
 - 서브 쿼리에서의 그룹 함수 6-10
 - 스칼라 서브 쿼리 18-11
 - 여러 열 서브 쿼리 6-7, 18-2, 18-8
 - 여러 행 서브 쿼리 6-2, 6-7, 6-14, 18-6
 - 외부 질의 6-5, 18-5
 - 중첩 질의 6-4, 18-4

人

서브쿼리에서의 그룹 함수 6-10
선택 1-3
속성 I-16, I-19
스칼라 서브 쿼리 18-11
스크립트 또는 명령 파일 7-20, 7-22, C-2

 스크립트 로딩 1-32

 스크립트 작성 1-26

스크립트 로딩 1-32

스크립트 작성 1-26

스키마 9-6, 13-4

스풀 파일 D-5

시간대 16-3

시스템 개발 주기 I-10

시스템 글로벌 영역 I-23, E-3, E-8

시퀀스 9-13, 12

 CREATE SEQUENCE 문 12-5

 CURRVAL 9-7, 12-8

 CYCLE 절 12-6

 NEXTVAL 9-7, 12-8

 고유 번호 생성 12-3

 시퀀스 값 캐시 12-11

시퀀스 캐시 12-1

쌍(pairwise) 비교 18-7

o

아카이브된 리두 로그 파일 E-6
암시적(implicit) 데이터 유형 변환 3-25
암호 파일 E-6

o

- 엔티티 I-16, I-17, I-18
- 엔티티 관계 다이어그램 I-16, I-17, I-16
- 여러 열 서브 쿼리 6-7, 18-2, 18-8
- 여러 테이블의 데이터(조인) 4
- 여러 행 서브 쿼리 6-2, 6-7, 6-14, 18-6
- 여러 행 함수 3-4
- 연결 연산자 1-18
- 연결된 그룹화 17-21
- 열 레벨 제약 조건 10-8
- 열 수정 9-25
- 오라클 데이터베이스의 데이터 구조 9-3, 9-5
- 오라클 인스턴스 E-3, E-7, I-23
- 온라인 트랜잭션 프로세싱(OLTP) I-8
- 외부 질의 6-5, 18-5
- 외부 테이블 20
 - ORGANIZATION EXTERNAL 절 20-18, 20-19
 - REJECT LIMIT 절 20-19
 - TYPE ACCESS_DRIVER_TYPE 20-19
 - 무조건 INSERT 20-7, 20-10
 - 조건 FIRST INSERT 20-7, 20-13, 20-14
 - 조건 INSERT ALL 20-7, 20-11
 - 피벗 INSERT 20-7, 20-15
- 우선순위 규칙 1-13, 2-19
- 우선순위 순서 1-12
- 응용 프로그램 서버 I-5

○

인덱스 9-3, 12

CREATE INDEX 문 12-17, 20-24

고유 인덱스 10-10, 12-6

고유하지 않은 인덱스 12-16

인덱스 생성이 필요한 경우 12-18

인덱스 이름 지정 20-2

인덱스 이름 지정 20-2

인라인 뷰 11-2, 11-21

인수 3-3, 3-5

인터넷 기능 I-7

일광 절약 시간 16-5

읽기 일관성 8-43, 8-44

×

자식 노드 19-10

자연 조인 4-24, 4-26

잠금 8-45

배타적 잠금(exclusive lock) 8-46

전자 상거래 19-6, I-3

제약 조건 10

CASCADE CONSTRAINTS 절 10-22

CHECK 제약 조건 10-16

FOREIGN KEY 10-13, 10-14, 10-15, I-19

NOT NULL 제약 조건 10-7

READ ONLY 제약 조건 11-19

REFERENCE 제약 조건 10-15

UNIQUE 제약 조건 10-9, 10-10

X

- 기본 키 10-11
- 무결성 제약 조건을 가진 레코드 삭제 8-22
- 비활성화 10-20
- 열 레벨 제약 조건 10-8
- 제약 조건 삭제 10-19
- 제약 조건 정의 10-5
- 참조 무결성 제약 조건 10-13
- 테이블 레벨 제약 조건 10-8
- 제약 조건 삭제 10-19
- 제약 조건 정의 10-5
- 제어 파일 E-5
- 조건 FIRST INSERT 20-7, 20-13, 20-14
- 조건 If-Then-Else 논리 3-51
- 조건 INSERT ALL 20-7, 20-11
- 조건 처리 3-51
- 조건, 논리 2-15
- 조합 고유 키 10-10
- 조합 열 17-17
- 중복 레코드 15-11
- 중첩 질의 6-4, 18-4
- 중첩 함수 3-42

Y

- 참조 무결성 제약 조건 10-13
- 체크포인트 프로세스 E-8
- 치환 변수 7-2, 7-3

Z

- 카티시안 곱 4-4, 4-5
- 키워드 1-4, 1-7

E

- 테이블 레벨 제약 조건 10-8
- 테이블 별칭 4-12
- 테이블 접두어 4-11
- 테이블 조인 1-3, 4
 - 3-Way 조인 4-30
 - ON 절 4-28, 4-29
 - RIGHT 테이블 4-33
 - 동가 조인 4-8, 4-27
 - 비동가 조인 4-14, 4-15
 - 세 개 이상의 테이블 조인 4-13
 - 왼쪽 테이블 4-32
 - 일치하는 레코드가 없을때 조인 4-34
 - 자연 조인 4-24, 4-26
 - 카티시안 곱(Cartesian Product) 4-4, 4-5
 - 테이블 자체 조인 4-19
 - 포괄 조인 4-17, 4-18
- 테이블에 대한 이름 지정 규칙 9-4
- 튜플 I-19
- 트랜잭션 8-32
- 트리 구조의 보고서 19

I

- 파라미터 파일 E-6
- 포괄 조인 4-17, 4-18
- 표현식
 - CASE 표현식 3-51, 3-52, 18-12
 - DECODE 표현식 3-51, 3-54
 - If-Then-Else 논리 3-51
 - 표현식에서의 계산 1-9

II

표현식의 계산 1-9
프로그램 글로벌 영역 E-16
프로세스 모니터 E-8
프로젝션 1-3
피벗 INSERT 20-7, 20-15

III

함수 3, 5
 AVG(Average) 5-6, 5-7
 COALESCE 함수 3-49
 CONCAT 함수 3-11
 COUNT 함수 5-8
 CURRENT_DATE 함수 16-6
 CURRENT_TIMESTAMP 함수 16-7
 Datetime 함수 16-2
 DBTIMEZONE 함수 16-9
 EXTRACT 함수 16-10
 TIMEZONE_ABBR 16-10
 TIMEZONE_REGION 16-10
 FROM_TZ 함수 16-11
 INITCAP 함수 3-9
 INSTR 함수 3-11
 LAST_DAY 함수 3-21
 LENGTH 함수 3-11
 LOCALTIMESTAMP 함수 16-8
 LOWER 함수 3-9
 LPAD 함수 3-11
 MAX 함수 5-6, 5-7
 MIN 함수 5-6, 5-7
 MONTHS_BETWEEN 함수 3-6, 3-21

함수 3, 5

NEXT_DAY 함수 3-21
NULLIF 함수 3-48
Number 함수 3-13
NVL 함수 3-45, 3-46, 5-5, 5-12
NVL2 함수 3-47
ROUND 함수 3-14, 3-21, 3-23
SESSIONTIMEZONE 함수 16-9
STDDEV 함수 5-7
SUBSTR 함수 3-11
SUM 함수 5-6, 5-7
SYS 함수 9-9
SYSDATE 함수 3-18, 3-20, 9-7
TO_CHAR 함수 3-31, 3-37, 3-39
TO_DATE 함수 3-39
TO_NUMBER 함수 3-39
TO_TIMESTAMP 함수 16-12
TO_YMINTERVAL 함수 16-13
TRIM 함수 3-11
TRUNC 함수 3-15, 3-21, 3-23
TZOFFSET 함수 16-14
UPPER 함수 3-9, 3-10
USER 함수 9-7
값 반환 3-3
날짜 변환 함수 3-4, 3-35
여러 행 함수 3-4
함수에서의 문자 데이터 유형 3-4

▶

- 함수 기반 인덱스 12-21
- 함수 내에서 문자 데이터 유형 3-4
- 해시 기호 3-38
- 행(row) 1-19, 17-8
- 행(row) 그룹에 대한 요약 결과 5-18
- 행(row) 제한 2-2
- 행(row) 집합 5-3
- 협정 세계시(UTC) 9-15
- 형식 모드(Format Mode) 3-31

