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

볼륨 1 • 학생용

40049KR11
제품 1.1
2002년 2월
D34384

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
Mozhe 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

발행인

Nita Brozowski

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

이 문서는 Oracle Corp.의 독점적 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.

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 널 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
중첩 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
동가 조인이란? 4-8

동가 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 널 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
표시자까지 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 톤백	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 그레프 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
- 롤이란? 13-9
- 롤 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 취소 방법 13-17
- 객체 권한 취소 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 향상된 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	

머리말

프로파일

시작하기 전에

이 과정을 시작하기 전에 먼저 GUI(그래픽 사용자 인터페이스)를 사용할 수 있어야 하며, 데이터 처리 개념 및 기술에 대한 충분한 지식을 가지고 있어야 합니다.

과정 구성

*Introduction to Oracle9i: SQL*은 강사가 지도하는 과정으로서 강의와 실제적인 연습 문제로 구성됩니다. 온라인 테모와 연습 세션을 통해 소개된 개념 및 기술을 강화합니다.

관련 서적

Oracle 참고 서적

제목	제품번호
<i>Oracle9i Reference, Release 1 (9.0.1)</i>	A90190-02
<i>Oracle9i SQL Reference, Release 1 (9.0.1)</i>	A90125-01
<i>Oracle9i Concepts, Release 1 (9.0.0)</i>	A88856-02
<i>Oracle9i Server Application Developer's Guide Fundamentals</i>	
<i>Release 1 (9.0.1)</i>	A88876-02
<i>iSQL*Plus User's Guide and Reference, Release 9.0.0</i>	A88826-01
<i>SQL*Plus User's Guide and Reference, Release 9.0.1</i>	A88827-02

참고 서적

- 시스템 릴리스 개시판
- 설치 및 사용 설명서
- *read.me* 파일
- International Oracle User's Group (IOUG) articles
- *Oracle Magazine*

표기법

다음의 두 목록은 텍스트나 코드 안에 사용되는 표기법을 보여줍니다.

텍스트 안의 표기법

규칙	객체 또는 용어	예제
대문자	명령, 함수, 열 이름, 테이블 이름, PL/SQL 객체, 스키마	SELECT 명령을 사용하여 EMPLOYEES 테이블의 LAST_NAME 열에 저장된 정보를 봅니다.
소문자, 기울임꼴	파일 이름, 구문 변수, 사용자 이름, 암호	where: <i>role</i> is the name of the role to be created.
머리 글자	트리거 및 버튼 이름	ORD 블록에 When-Validate-Item 트리거를 할당합니다. Cancel 을 선택합니다.
기울임꼴	책, 과정 및 설명서 이름 강조된 단어 또는 문	주제에 대한 자세한 내용은 <i>Oracle Server SQL Language Reference Manual</i> 을 참조하십시오.
따옴표	과정 안에서 참조되는 학습 모듈 제목	데이터베이스 변경 내용은 저장하지 마십시오. 이 주제는 3과, “객체 사용”에서 다릅니다.

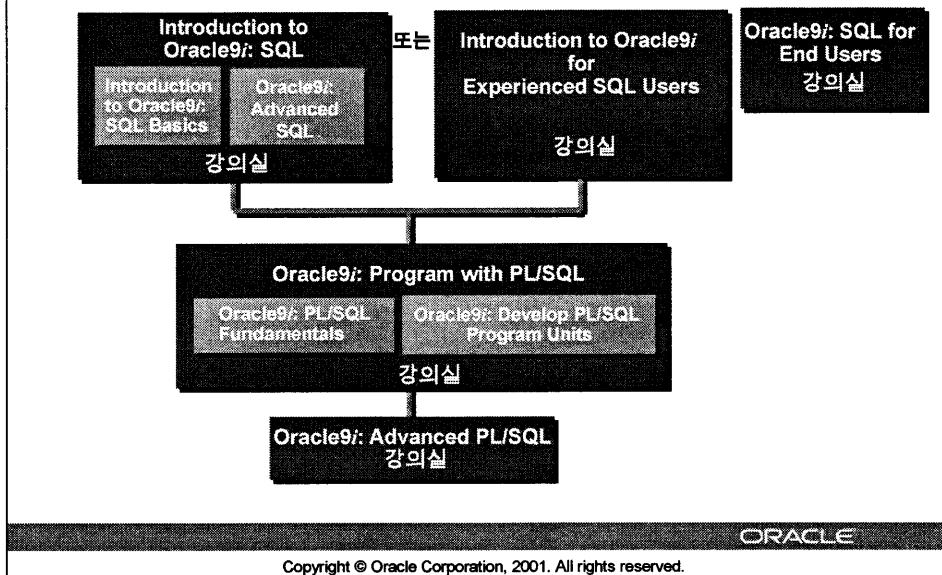
표기법(계속)

코드 안의 표기법

규칙	객체 또는 용어	예제
대문자	명령, 함수	<code>SELECT employee_id FROM employees;</code>
소문자, 기울임꼴	구문 변수	<code>CREATE ROLE role;</code>
머리 글자	폼 트리거	<code>Form module: ORD Trigger level: S_ITEM.QUANTITY item Trigger name: When-Validate-Item . . .</code>
소문자	열 이름, 테이블 이름, 파일 이름, PL/SQL 객체	<code>. . . OG_ACTIVATE_LAYER (OG_GET_LAYER ('prod_pie_layer')) . . . SELECT last_name FROM employees;</code>
굵은 체	사용자가 입력하는 텍스트	<code>CREATE USER scott IDENTIFIED BY tiger;</code>

교과
과정표

Oracle9i 언어 과정



통합된 언어 과정

*Introduction to Oracle9i: SQL*은 *Introduction to Oracle9i: SQL Basics*과 *Oracle9i: Advanced SQL*의 두 개 모듈로 구성됩니다. *Introduction to Oracle9i: SQL Basics*에서는 테이터베이스 구조의 생성과 관계형 테이터베이스에서의 테이터 저장, 검색, 조작 방법을 다룹니다. *Oracle9i: Advanced SQL*에서는 고급 SELECT 문, Oracle SQL 및 iSQL*Plus Reporting을 다릅니다.

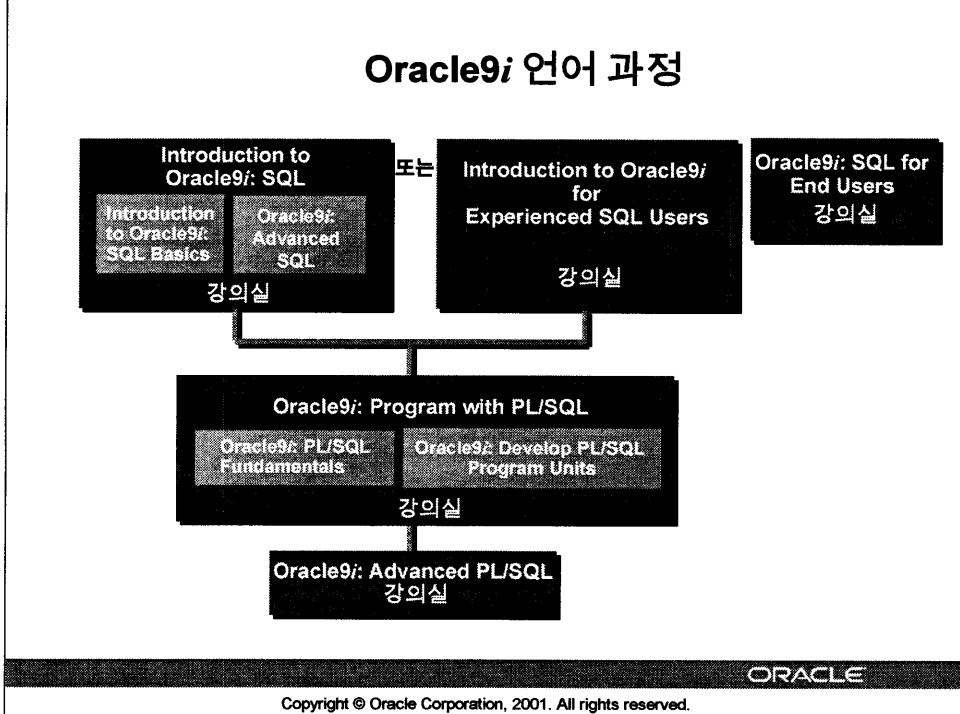
관계형 테이터베이스에 대한 경험이 있으며 SQL에도 사전 지식이 있다면 *Introduction to Oracle9i for Experienced SQL Users*가 적당합니다. 이 과정에서는 ANSI SQL에 속하지 않으며 오라클에 특정한 SQL 문에 대해 설명합니다.

*Oracle9i: Program with PL/SQL*은 *Oracle9i: PL/SQL Fundamentals*와 *Oracle9i: Develop PL/SQL Program Units*의 두 개 모듈로 구성됩니다. *Oracle9i: PL/SQL Fundamentals*에서는 PL/SQL 언어 구조, 실행 흐름, SQL과의 인터페이스를 비롯한 PL/SQL의 기초적인 부분을 다룹니다. *Oracle9i: Develop PL/SQL Program Units*에서는 내장 프로시저, 함수, 패키지, 트리거를 작성하는 방법과 함께 PL/SQL 프로그램 코드를 유지 관리하고 디버깅하는 방법을 다룹니다.

*Oracle9i: SQL for End Users*는 프로그래밍 관련 지식이 거의 없는 일반인을 대상으로 하는 과정으로 기본적인 SQL 문을 다룹니다. 이 과정은 기본적인 SQL 프로그래밍을 익히고자 하는 일반 사용자에게 적합합니다.

*Oracle9i: Advanced PL/SQL*은 PL/SQL 프로그래밍에 경험이 있는 일반인을 대상으로 하며 코드 효율성, 객체 지향 프로그래밍, 외부 코드를 사용한 작업, 오라클 지원 패키지의 고급 기능 등이 과정에서 다루는 내용입니다.

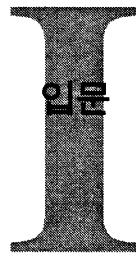
Oracle9i 언어 과정



통합된 언어 과정

슬라이드에서는 언어 과정에서 제공하는 다양한 모듈과 교육 과정을 보여 줍니다. 아래의 표에서는 TBT에서 유사한 과정을 제공하는 모듈 및 과정을 소개합니다.

과정 또는 모듈	TBT 과정
<i>Introduction to Oracle9i: SQL Basics</i>	<i>Oracle SQL: Basic SELECT Statements</i> <i>Oracle SQL: Data Retrieval Techniques</i> <i>Oracle SQL: DML and DDL</i>
<i>Oracle9i: Advanced SQL</i>	<i>Oracle SQL and SQL*Plus: Advanced SELECT Statements</i> <i>Oracle SQL and SQL*Plus: SQL*Plus and Reporting</i>
<i>Introduction to Oracle9i for Experienced SQL Users</i>	<i>Oracle SQL Specifics: Retrieving and Formatting Data</i> <i>Oracle SQL Specifics: Creating and Managing Database Objects</i>
<i>Oracle9i: PL/SQL Fundamentals</i>	<i>PL/SQL: Basics</i>
<i>Oracle9i: Develop PL/SQL Program Units</i>	<i>PL/SQL: Procedures, Functions, and Packages</i> <i>PL/SQL: Database Programming</i>
<i>Oracle9i: SQL for End Users</i>	<i>SQL for End Users: Part 1</i> <i>SQL for End Users: Part 2</i>
<i>Oracle9i: Advanced PL/SQL</i>	<i>Advanced PL/SQL: Implementation and Advanced Features</i> <i>Advanced PL/SQL: Design Considerations and Object Types</i>



ORACLE

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

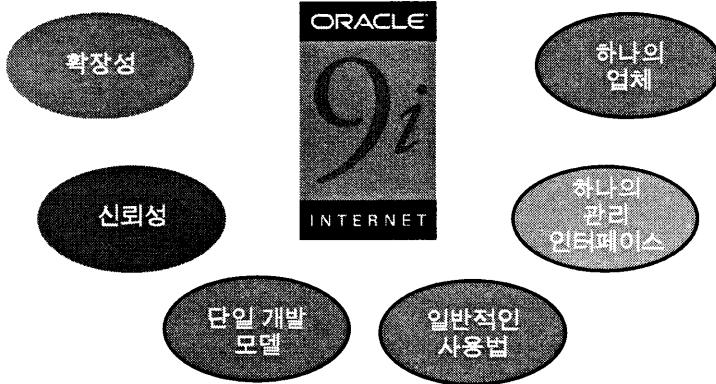
- Oracle9i의 특징 나열
- 관계형 데이터베이스의 이론적인 측면 및 물리적인 측면 설명
- RDBMS 및 ORDBMS의 오라클 구현 설명

단원 목표

이 단원에서는 RDBMS(관계형 데이터베이스 관리 시스템)와 ORDBMS(액체 관계형 데이터베이스 관리 시스템)를 설명하고 다음에 대해 소개합니다.

- 오라클 특유의 SQL 문
- SQL을 실행하고 형식 지정 및 보고 용도로 사용되는 iSQL*Plus

Oracle9*i*



I-3

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

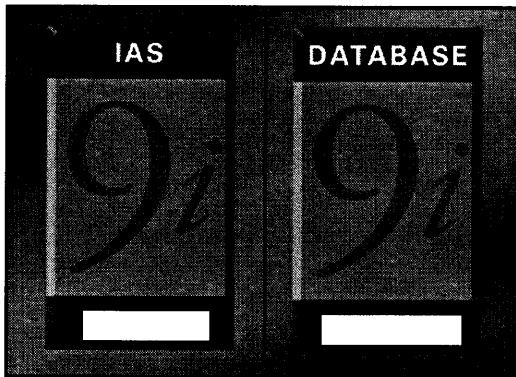
Oracle9*i*의 특징

오라클은 e-business에 사용할 수 있는 포괄적인 고성능 기반 구조를 제공합니다. 이를 Oracle9*i*라고 합니다. Oracle9*i*에는 인터넷 응용 프로그램을 개발, 배포 및 관리하는 데 필요한 모든 것이 포함되어 있습니다.

장점:

- 작게는 부서로부터 크게는 기업의 e-business 사이트까지 이르는 확장성
- 강력하고, 신뢰성 있고, 유용하며, 보안이 철저한 구조
- 단일 개발 모델, 간단한 배포 옵션
- 오라클 플랫폼(SQL, PL/SQL, Java, XML 등)을 통해 조직의 현재 업무 능력 향상
- 모든 응용 프로그램에 대해 관리 인터페이스 단일화
- 산업 표준 기술, 단독 소유권 없음

Oracle9*i*



ORACLE

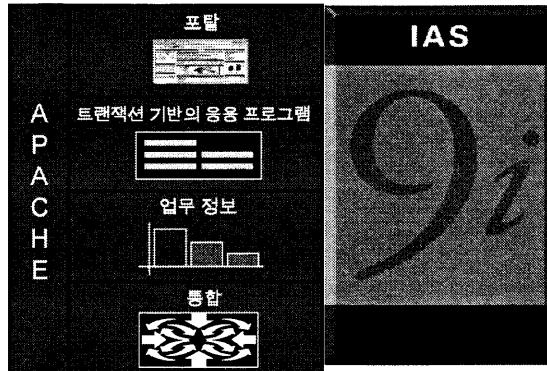
I-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9*i*

Oracle9*i*에는 인터넷 응용 프로그램에 완벽하고 단일한 기반 구조를 제공하는 두 가지 제품으로, Oracle9*i* Application Server와 Oracle9*i* Database가 있습니다.

Oracle9i Application Server



Oracle9i Application Server

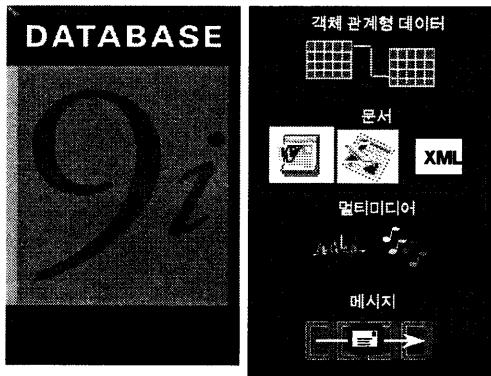
Oracle9i Application Server(Oracle9iAS)에서 모든 응용 프로그램을 실행할 수 있으며, Oracle9i Database에 모든 데이터를 저장할 수 있습니다.

Oracle9i Application Server는 실행하고자 하는 각기 다른 모든 서버 응용 프로그램의 서비스를 포함할 수 있는 유일한 응용 프로그램 서버입니다. Oracle9iAS에서는 다음을 실행할 수 있습니다.

- 포탈 또는 웹 사이트
- Java 트랜잭션 기반의 응용 프로그램
- Business intelligence 응용 프로그램

또한 조직 전체에 걸쳐 사용자, 응용 프로그램 및 데이터를 통합할 수 있도록 해줍니다.

Oracle9i Database



I-6

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

Oracle9i Database

두 제품의 역할은 매우 간단합니다. Oracle9i Database는 모든 형태의 데이터를 관리합니다. 기업 데이터베이스에서 사용하는 객체 관계형 데이터 뿐만 아니라 다음과 같이 구조화되지 않은 데이터도 관리할 수 있습니다.

- 스프레드시트
- Word 문서
- PowerPoint 프레젠테이션
- XML
- MP3, 그래픽, 비디오 등의 멀티미디어 데이터 유형

데이터가 반드시 데이터베이스에 존재하지 않아도 됩니다. Oracle9i Database에는 파일 시스템에 저장된 정보의 메타 데이터를 저장할 수 있는 서비스가 있습니다. 데이터베이스 서버를 사용하면 위치에 상관 없이 정보를 관리하고 서비스를 제공할 수 있습니다.

관계형 데이터베이스 관리 시스템 및 객체 관계형 데이터베이스 관리 시스템

- 관계형 모델 및 객체 관계형 모델
- 사용자 정의 데이터 유형 및 객체
- 관계형 데이터베이스와 완벽하게 호환 가능
- 멀티미디어 및 대형 객체 지원
- 고품질 데이터베이스 서버 기능

ORACLE

I-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Server 정보

Oracle9i 서버는 관계형 모델 및 객체 관계형 모델을 모두 지원합니다.

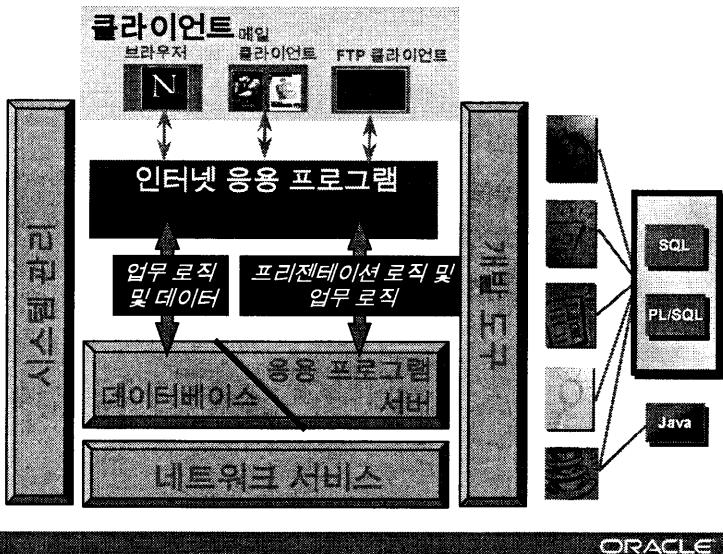
Oracle Server는 데이터 모델링 기능을 확장하여 객체 관계형 데이터베이스 모델을 제공함으로써 객체 지향 프로그래밍, 복합 데이터 유형, 복합 업무 객체 및 관계형 모델과의 완벽한 호환성을 지원할 수 있습니다.

Oracle Server는 향상된 런타임 데이터 구조 공유, 대용량 버퍼 캐시, 지역 가능한 제약 조건 등 OLTP(온라인 트랜잭션 프로세싱) 응용 프로그램의 향상된 성능 및 기능 몇 가지도 포함하고 있습니다. 또한 삽입, 생성, 삭제 작업 등의 병렬 실행과 분할, 병렬 인식 질의 최적화 등의 기술 향상은 데이터 웨어하우스 응용 프로그램에 매우 유용합니다. NCA(네트워크 컴퓨팅 구조) 프레임워크 내에서 작동되는 Oracle9i는 분산되어 여러 계층으로 구성된 클라이언트-서버 및 웹 기반 응용 프로그램을 지원합니다.

Oracle9i는 10,000여 명의 사용자가 동시에 사용할 수 있으며 최대 512페타바이트(1페타바이트는 1,000테라바이트임)의 데이터를 지원하고 구조화된 일반적인 데이터 뿐 아니라 텍스트, 공간, 이미지, 소리, 비디오, 일련의 시간 등 모든 유형의 데이터를 처리할 수 있습니다.

자세한 내용은 *Oracle9i Concepts*를 참조하십시오.

오라클 인터넷 플랫폼



I-8

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

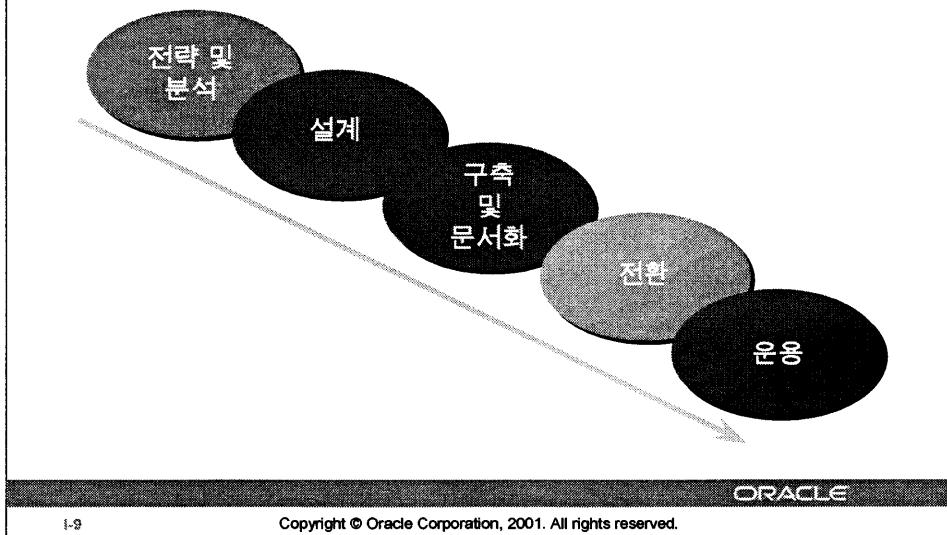
오라클 인터넷 플랫폼

오라클은 전자 상거래 및 데이터 웨어하우징을 위한 포괄적인 고성능 인터넷 플랫폼을 제공합니다. 이 통합 플랫폼에는 인터넷 응용 프로그램의 개발, 배포 및 관리에 필요한 모든 사항이 포함되어 있습니다. 오라클 인터넷 플랫폼은 다음 세 가지 핵심 요소를 기반으로 구축되었습니다.

- 프리젠테이션 로직을 처리하는 브라우저 기반 클라이언트
- 브라우저 기반 클라이언트에 대해 업무 로직을 실행하고 프리젠테이션 로직을 처리하는 응용 프로그램 서버
- 데이터베이스를 집중적으로 사용하는 업무 로직을 실행하고 데이터를 처리하는 데이터베이스

오라클은 업무 응용 프로그램을 구축할 수 있는 최고급 GUI(그래픽 사용자 인터페이스) 방식의 다양한 개발 툴과 많은 분야의 업무 및 산업에 필요한 다양한 응용 소프트웨어군을 제공합니다. 내장 프로시저, 함수 및 패키지는 SQL, PL/SQL 또는 Java를 사용하여 작성할 수 있습니다.

시스템 개발 주기



I-9

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

시스템 개발 주기

개념적인 단계부터 운영 단계에 이르기까지 여러 단계가 포함된 시스템 개발 주기에 따라 데이터베이스를 개발할 수 있습니다. 데이터베이스 개발에 대한 이러한 체계적인 하향식 접근 방법은 업무 정보 요구 사항을 운영 가능한 데이터베이스로 변형합니다.

전략 및 분석

- 업무 요구 사항을 연구하고 분석합니다. 사용자 및 관리자와의 면담을 통해 요구되는 정보를 확인합니다. 장래의 시스템 사양 뿐 아니라 앤터프라이즈 및 응용 프로그램이 수행할 역할을 구체화합니다.
- 시스템 모델을 구축합니다. 서술 형식의 업무 내용을 업무 정보 요구 사항과 규칙을 나타내는 그래픽으로 변형합니다. 분석가 및 전문가와 함께 모델을 확인하고 세분화합니다.

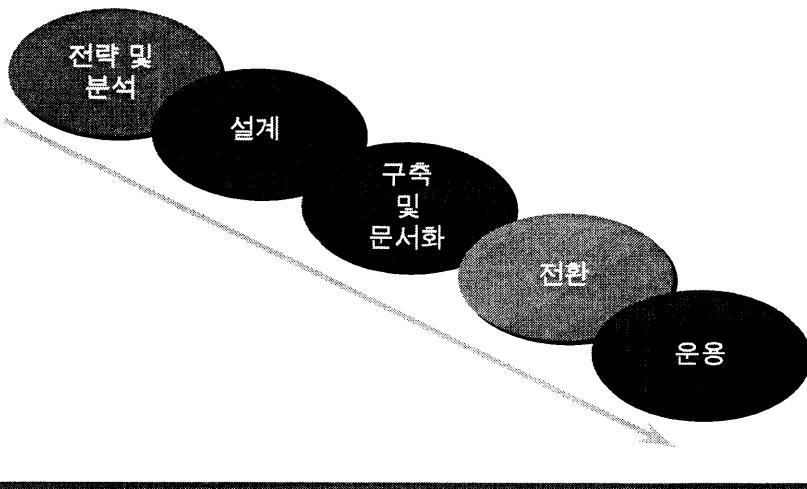
설계

전략 및 분석 단계에서 개발한 모델을 기반으로 데이터베이스를 설계합니다.

구축 및 문서화

- 프로토타입 시스템을 구축합니다. 데이터베이스에 대한 테이블 및 지원 객체를 만드는 명령을 작성하고 실행합니다.
- 사용자 설명서, 도움말 텍스트 및 작동 설명서를 개발하여 시스템을 사용 및 작동할 수 있도록 지원합니다.

시스템 개발 주기



ORACLE

I-10

Copyright © Oracle Corporation, 2001. All rights reserved.

시스템 개발 주기(계속)

전환

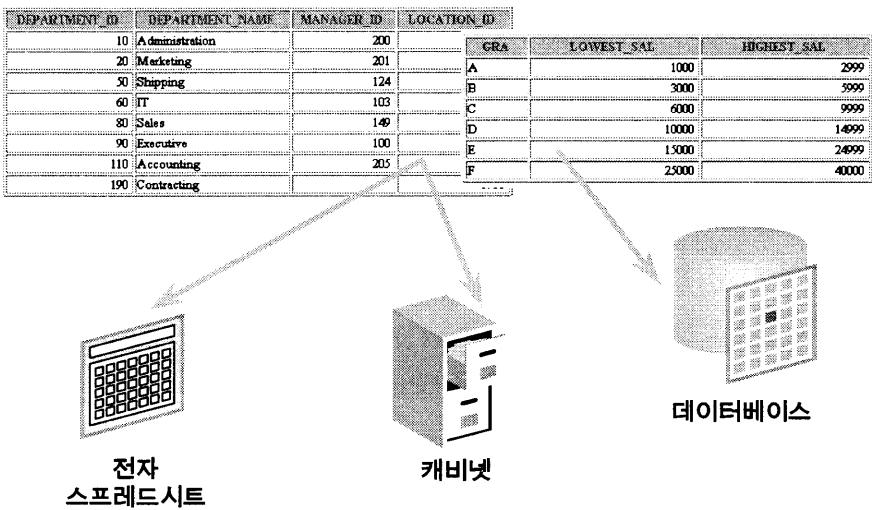
프로토타입을 정체화합니다. 사용자 승인 테스트, 기준 데이터 변환 및 병렬 작업을 통해 응용 프로그램을 운영 단계로 전환합니다. 필요하다면 수정 작업을 합니다.

운용

시스템을 사용자에게 공개하고 운영 시스템을 작동합니다. 시스템의 성능을 모니터하고 시스템을 보다 나은 형태로 개선합니다.

참고: 시스템 개발 주기의 각 단계는 반복적으로 수행할 수 있습니다. 이 과정에서는 시스템 개발 주기 중 구축 단계를 중점적으로 설명합니다.

다양한 데이터 저장 매체



I-11

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

정보 저장

모든 조직에는 정보가 필요합니다. 도서관에서는 회원, 책, 반납일, 벌금 등의 목록을 관리하고, 회사에서는 사원, 부서, 급여 등에 대한 정보를 저장해야 합니다. 이러한 정보를 데이터라고 합니다. 각 조직마다 데이터를 다양한 매체에 여러 형태로 저장할 수 있습니다. 예를 들면, 캐비넷에 문서를 보관하거나 전자 스프레드시트 또는 데이터베이스에 데이터를 저장할 수 있습니다.

데이터베이스는 조직화된 정보입니다.

데이터베이스를 관리 하려면 DBMS(데이터베이스 관리 시스템)가 필요합니다. DBMS는 요청에 따라 데이터베이스에서 데이터를 저장, 검색 및 수정하는 프로그램입니다. 데이터베이스에는 계층적 데이터베이스, 네트워크 데이터베이스, 관계형 데이터베이스, 그리고 최신의 객체 관계형 데이터베이스 등 네 가지 기본 유형이 있습니다.

관계형 데이터베이스 개념

- E.F. Codd 박사는 1970년에 데이터베이스 시스템을 위한 관계형 모델을 제안했습니다.
- 이것이 RDBMS(관계형 데이터베이스 관리 시스템)의 기초가 되었습니다.
- 관계형 모델은 다음으로 구성됩니다.
 - 객체 또는 관계의 모음
 - 관계에 적용되는 연산자 집합
 - 정확성과 일관성을 위한 데이터 무결성

ORACLE

I-12

Copyright © Oracle Corporation, 2001. All rights reserved.

관계형 모델

관계형 모델의 원리는 1970년 6월에 E. F. Codd 박사가 발표한 “A Relational Model of Data for Large Shared Data Banks”라는 논문에서 최초로 제시되었습니다. 이 논문에서 Codd 박사는 데이터베이스 시스템에 대한 관계형 모델을 제안했습니다.

당시에 더 많이 사용되던 모델은 계층적 데이터베이스와 네트워크 데이터베이스, 그리고 훨씬 간단한 플랫 파일 데이터 구조였습니다. RDBMS(관계형 데이터베이스 관리 시스템)는 사용하기 쉽고 융통성 있는 구조로 인해 곧 많은 사용자가 사용하게 되었습니다. 또한 오라클과 같은 혁신적인 여러 업체가 종합적인 솔루션을 제공하는 강력한 응용 프로그램군과 사용자 제품을 개발함으로써 RDBMS를 더욱 보완했습니다.

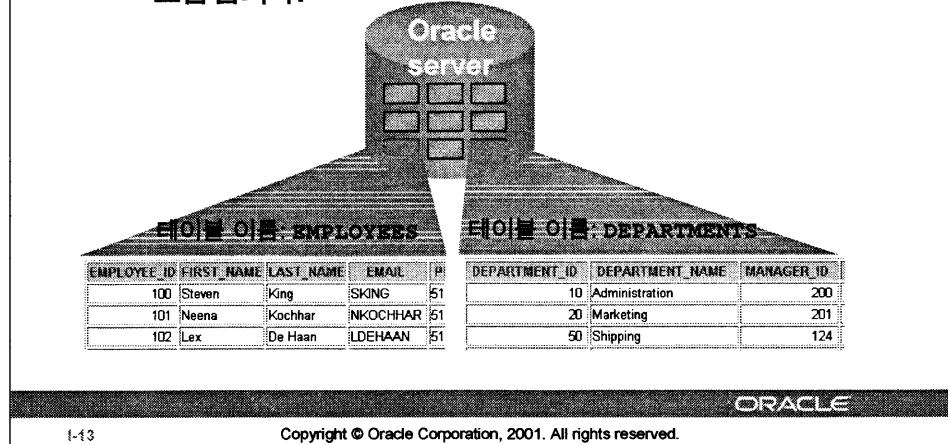
관계형 모델의 구성 요소

- 데이터를 저장하는 객체 또는 관계의 모음
- 관계에 적용되어 다른 관계 생성을 할 수 있도록 해주는 연산자 집합
- 정확성과 일관성을 위한 데이터 무결성

자세한 사항은 E. F. Codd의 *The Relational Model for Database Management Version 2*(Reading, Mass.: Addison-Wesley, 1990)를 참조하십시오.

관계형 데이터베이스 정의

관계형 데이터베이스는 관계 또는 2차원 테이블의 모음입니다.



I-13

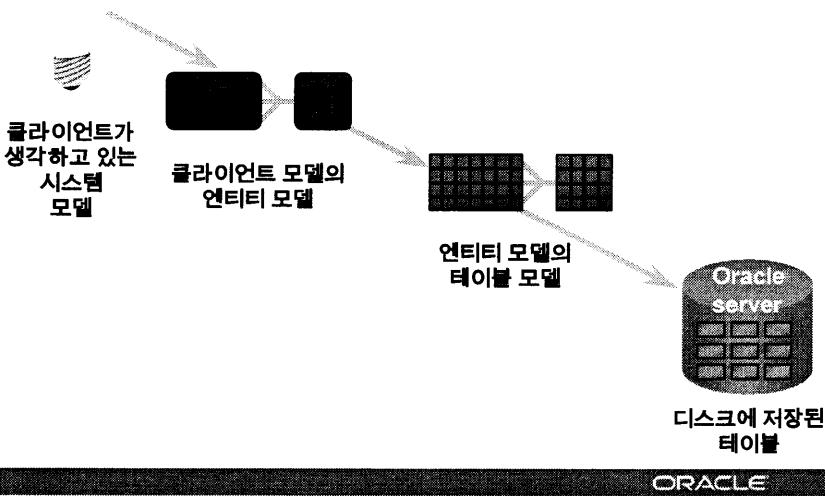
Copyright © Oracle Corporation, 2001. All rights reserved.

관계형 데이터베이스 정의

관계형 데이터베이스는 관계 또는 2차원 테이블을 사용하여 정보를 저장합니다.

예를 들어, 회사 내 모든 사원에 대한 정보를 저장해야 할 경우, 관계형 데이터베이스에서는 사원 테이블, 부서 테이블, 급여 테이블 등 여러 테이블을 생성하여 사원에 대한 각 정보를 저장합니다.

데이터 모델



I-14

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

데이터 모델

모델은 설계의 토대입니다. 예를 들어, 엔지니어는 자동차를 생산하기 전에 자동차 모델을 만들어 세부 사항에 대해 작업합니다. 동일한 방식으로 시스템 설계자는 모델을 개발하여 아이디어를 조사하고 데이터베이스 설계에 대한 이해를 향상시킵니다.

모델의 용도

모델은 사용자의 머리 속에 있는 개념을 전달하도록 도와 줍니다. 모델은 다음 작업에 사용될 수 있습니다.

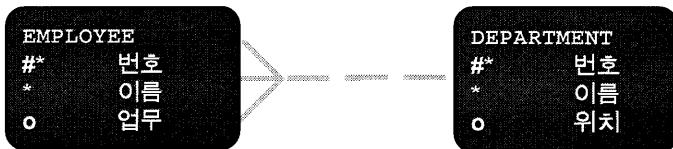
- 의사 소통
- 범주화
- 설명
- 지정
- 조사
- 발전
- 분석
- 모방

이러한 여러 가지 용도에 맞고, 일반 사용자가 이해할 수 있고, 개발자가 데이터베이스 시스템을 구축하기에 충분한 세부 사항을 포함하는 모델을 생산하는 데 그 목표가 있습니다.

엔티티 관계 모델

(ER)

- 업무 사양 또는 업무 내용 설명서를 토대로 엔티티 관계 다이어그램을 만듭니다.



• 시나리오

- "... 한 명 이상의 사원을 한 부서에 할당합니다..."
- "... 일부 부서에는 아직 할당된 사원이 없습니다..."

ORACLE

I-15

Copyright © Oracle Corporation, 2001. All rights reserved.

ER 모델링

효율적인 시스템에서는 데이터가 명확히 구별된 범주 또는 엔티티로 구분됩니다. ER(엔티티 관계) 모델은 업무 상 여러 엔티티와 그들 간의 관계를 보여줍니다. ER 모델은 업무 사양이나 업무 내용을 설명서를 통해 도출해 낼 수 있으며 시스템 개발 주기의 분석 단계에서 작성됩니다. ER 모델은 업무에 필요한 정보와 업무 내에서 수행되는 작업을 분리합니다. 비록 업무 상 수행되는 작업이 변경되더라도 정보의 유형은 그대로 남아 있는 경향이 있습니다. 따라서 데이터 구조도 그대로 남아 있게 됩니다.

ER 모델링의 장점

- 조직에 대한 정보를 정확하고 세밀한 형식으로 문서화
- 요구사항에 대한 명확한 범위를 제공
- 데이터베이스 설계를 이해하기 쉬운 그림 형태의 맵으로 제공
- 여러 응용 프로그램을 통합하는 효율적인 구조 제공

주요 구성 요소

- **엔티티:** 정보가 필요한 주요 대상(예: 부서, 사원, 주문 등)
- **속성:** 엔티티를 설명하거나 제한하는 내용. 예를 들어, 사원 엔티티에 대한 속성은 사원 번호, 이름, 업무, 입사일, 부서 번호 등이 될 수 있습니다. 각 속성은 반드시 필요하거나 선택적으로 필요한데 이러한 상태를 선택 성이라고 합니다.
- **관계:** 선택성과 그 관계차수를 나타내는 엔티티 간의 명명된 연관성(예: 사원과 부서, 주문과 품목 등)

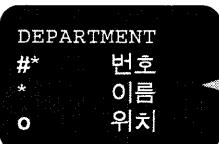
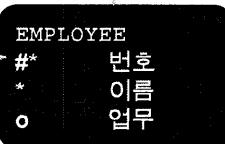
엔티티 관계 모델링 표기법

엔티티

소프트 상자
하나뿐인 고유한 이름
대문자
동의어는 괄호 안에 표기

속성

하나뿐인 단독 이름
소문자
필수 속성은 “*”로 표시
선택 속성은 “o”로 표시



할당

구성

UID(고유 식별자)

기본 UID는 “#”로 표시
보조 UID는 “(#)”로 표시

ORACLE

ER 모델링(계속)

엔티티

모델에서 엔티티를 표현할 때는 다음 표기법을 사용합니다.

- 차원에 제한이 없는 소프트 상자
- 하나뿐인 고유한 엔티티 이름
- 엔티티 이름은 대문자 사용
- 선택적인 동의어 이름은 괄호 안에 대문자로 표시: ()

속성

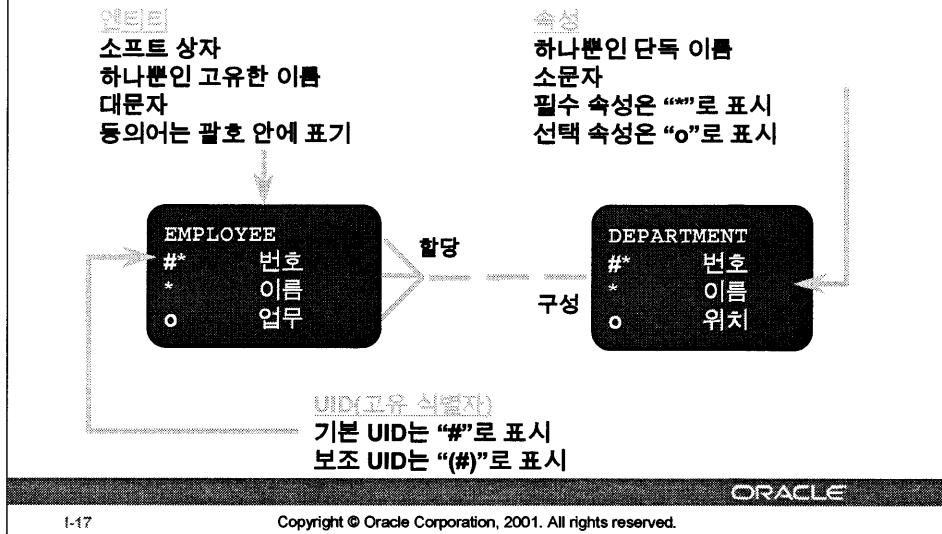
모델에서 속성을 표현할 때는 다음 표기법을 사용합니다.

- 하나뿐인 단독 이름은 소문자 사용
- 필수 속성이나 반드시 알아야 할 값에 별표로 태그 처리: *
- 선택 속성이나 알 수 있는 값에 문자 o로 태그 처리

관계

기호	설명
점선	“가능성(may be)”을 나타내는 선택 요소
실선	“당위성(must be)”을 나타내는 필수 요소
세줄 선(꼴모양)	“하나 이상(one or more)”을 나타내는 정도 요소
한줄 선(꼴모양)	“단 하나(one and only one)”를 나타내는 정도 요소

엔티티 관계 모델링 표기법



ER 모델링(계속)

관계

관계는 각각 다음과 같이 지정됩니다.

- 레이블: 예를 들어, ~에게 배우다 또는 ~에 할당되다
- 선택성: 당위성(must be) 또는 가능성(may be)
- 관계차수(degree): 단 하나(one and only one) 또는 하나 이상(one or more)

참고: 카드inality라는 용어는 관계차수(degree)라는 용어와 동의어입니다.

각 소스 엔티티 {may be | must be} 관계 이름 {one and only one | one or more} 대상 엔티티

참고: 이 표기법은 시계 방향으로 읽습니다.

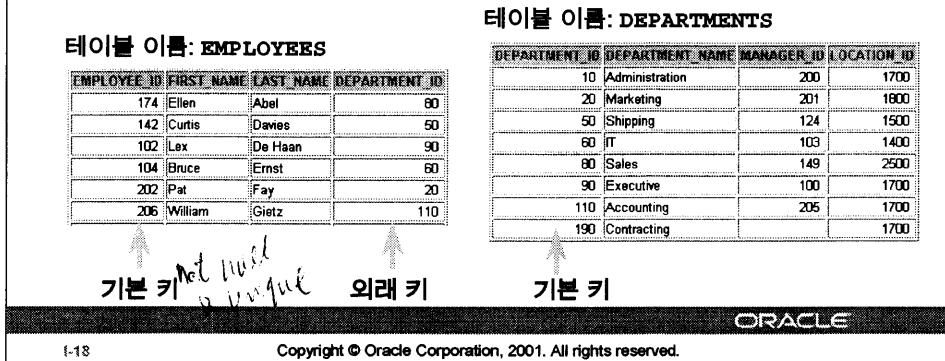
고유 식별자

UID(고유 식별자)는 속성이나 관계를 각각 조합하거나 함께 조합한 것으로 엔티티를 식별하는 역할을 합니다. 각 엔티티는 고유하게 식별될 수 있어야 합니다.

- UID를 구성하는 각 속성은 숫자 기호로 태그 처리: #
- 보조 UID는 팔호 안의 숫자 기호로 태그 처리: (#)

여러 테이블 관련시키기

- 테이블의 각 데이터 행(row)은 PK(기본 키)에 의해 고유하게 식별됩니다.
- FK(외래 키)를 사용하여 여러 테이블의 데이터를 논리적으로 관련시킬 수 있습니다.



여러 테이블 관련시키기

각 테이블에는 단 하나의 엔티티를 나타내는 데이터가 들어 있습니다. 예를 들어, EMPLOYEES 테이블에는 사원에 대한 정보가 들어 있습니다. 데이터 범주는 각 테이블의 맨 위에 나열되어 개별 사례가 그 아래에 나열됩니다. 테이블 형식을 사용하면 정보를 쉽게 보고, 이해하고, 사용할 수 있습니다.

엔티티가 서로 다르면 해당 데이터도 서로 다른 테이블에 저장되므로 특정 질의에 응답하려면 둘 이상의 테이블을 결합해야 합니다. 예를 들어, 한 사원이 근무하는 부서의 위치를 찾는 경우 사용자는 사원에 대한 데이터가 들어 있는 EMPLOYEES 테이블과 부서에 대한 정보가 들어 있는 DEPARTMENTS 테이블의 정보가 필요합니다. RDBMS에서는 외래 키를 사용하여 한 테이블의 데이터를 다른 테이블의 데이터에 관련시킬 수 있습니다. 외래 키는 동일한 테이블이나 다른 테이블의 기본 키를 참조하는 열 또는 열 집합입니다.

한 테이블의 데이터를 다른 테이블의 데이터에 관련시키는 기능을 사용하면 관리가 가능한 별도의 단위로 정보를 조직화할 수 있습니다. 사원 데이터를 별도의 테이블에 저장하여 부서 데이터와는 논리적으로 구분하여 관리할 수 있습니다.

기본 키 및 외래 키에 대한 지침

- 기본 키에 중복 값을 사용할 수 없습니다.
- 기본 키는 일반적으로 변경할 수 없습니다.
- 외래 키는 데이터 값을 기준으로 하며 물리적 포인터가 아닌 논리적 포인터입니다.
- 외래 키 값은 기존의 기본 키 값 또는 고유 키 값에 일치하거나 널 값이어야 합니다.
- 외래 키는 기본 키 또는 고유 키 열을 참조해야 합니다.

관계형 데이터베이스에서 사용되는 용어

2

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID
100	King	Steven	24000		90
101	Kochhar	Neena	17000		90
102	De Haan	Lex	17000		90
103	Hunold	Alexander	9000		60
104	Ernst	Bruce	6000		60
107	Lorentz	Diana	4200		60
124	Mourgos	Kevin	5800		50
141	Rajs	Trenna	3500		50
142	Davies	Curtis	3100		50
143	Matos	Randall	2600		50
144	Vargas	Peter	2500		50
149	Zlotkey	Eleni	10500	.2	80
174	Abel	Ellen	11000	.3	80
176	Taylor	Jonathon	8600	.2	80
178	Grant	Kimberely	7000	.15	80
200	Whalen	Jennifer	4400		10
201	Hartstein	Michael	13000		20
202	Fay	Pat	6000		20
205	Higgins	Shelley	12000		110
206	Gietz	William	8300		110

3

5

4

6

관계형 데이터베이스에서 사용되는 용어

관계형 데이터베이스는 하나 이상의 테이블을 포함할 수 있습니다. 테이블은 RDBMS의 기본 저장 구조로서 사원, 고객, 주문 등 현실에서 필요한 모든 데이터를 보유할 수 있습니다.

슬라이드는 EMPLOYEES 테이블 또는 관계의 내용을 보여줍니다. 여기서 각 번호는 다음을 나타냅니다.

- 특정 사원에 대해 필요한 모든 데이터를 나타내는 단일 행(row) 또는 테이블입니다. 테이블의 각 행은 기본 키로 식별되어야 하며 기본 키는 중복될 수 없습니다. 행의 순서는 중요하지 않습니다. 행 순서는 데이터를 검색할 때 지정합니다.
- 사원 번호가 들어 있는 열 또는 속성입니다. 사원 번호는 EMPLOYEES 테이블에서 사원을 고유하게 식별합니다. 예제에서는 사원 번호 열이 기본 키로 지정되어 있습니다. 기본 키에는 반드시 값이 포함되어야 하며 그 값은 고유해야 합니다.
- 키 값이 아닌 열이며 테이블에서 한 종류의 데이터(예: 모든 사원의 급여)를 나타냅니다. 데이터를 저장할 때는 열의 순서가 중요하지 않습니다. 열 순서는 데이터를 검색할 때 지정합니다.
- 부서 번호를 포함하는 열이며 외래 키기도 합니다. 외래 키는 테이블을 서로 관련시키는 방식을 정의하는 열입니다. 외래 키는 같은 테이블 또는 다른 테이블의 기본 키 또는 고유 키를 참조합니다. 예제에서 DEPARTMENT_ID는 DEPARTMENTS 테이블에서 부서를 고유하게 식별합니다.
- 필드에 값이 없을 경우 null이라고 합니다. EMPLOYEES 테이블에서는 영업 사원이라는 직책을 가진 사원만 COMMISSION_PCT(커미션) 필드에 값을 가집니다.
- 필드는 행과 열이 교차하는 부분으로 하나의 값만 넣을 수 있습니다.

관계형 데이터베이스 특성

관계형 데이터베이스

- SQL 문을 실행하여 액세스하고 수정할 수 있습니다.
- 물리적인 포인터가 없는 테이블들을 가지고 있습니다.
- 연산자 집합을 사용합니다.

ORACLE

I-20

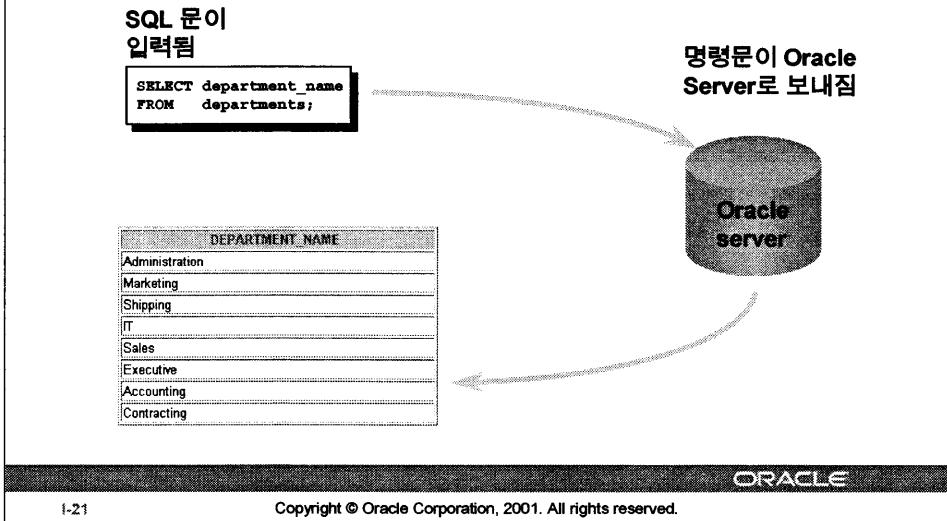
Copyright © Oracle Corporation, 2001. All rights reserved.

관계형 데이터베이스 특성

관계형 데이터 베이스에서는 테이블에 대한 액세스 경로를 지정하지 않으며 데이터가 물리적으로 배열되어 있는 방식을 알 필요가 없습니다.

데이터 베이스를 액세스하려면 SQL 문을 실행합니다. SQL 문은 관계형 데이터베이스를 사용하기 위한 ANSI(American National Standards Institute) 표준입니다. 이 언어에는 관계를 분할하고 결합하기 위한 연산자 집합이 포함되어 있습니다. 데이터베이스는 SQL 문을 사용하여 수정할 수 있습니다.

SQL을 사용하여 RDBMS와 통신



I-21

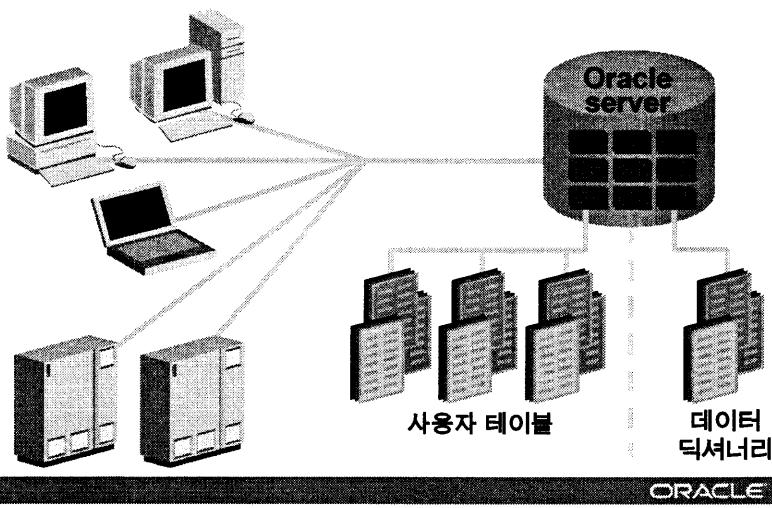
Copyright © Oracle Corporation, 2001. All rights reserved.

SQL

SQL을 사용하면 Oracle Server와 통신할 수 있으며 다음 장점이 있습니다.

- 효율적임
- 배우고 사용하기 쉬움
- 기능적으로 완벽함(SQL을 사용하여 테이블의 데이터를 정의, 검색 및 조작 가능)

관계형 데이터베이스 관리 시스템



I-22

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

관계형 데이터베이스 관리 시스템

오라클은 Oracle9i라는 융통성 있는 RDBMS를 제공합니다. Oracle9i 기능을 사용하면 관계형 구조와 PL/SQL의 모든 장점을 사용하여 데이터를 저장하고 관리할 수 있습니다. PL/SQL은 프로그램 단위를 저장하고 실행하는 기능을 제공하는 엔진입니다. Oracle9i는 Java 및 XML도 지원합니다. Oracle Server는 최적화 기술을 기반으로 하는 데이터 검색 옵션을 제공하며 데이터베이스의 액세스 및 사용 방식을 제어하는 보안 기능도 포함하고 있습니다. 그 외에 잠금 방식을 통해 일관성을 유지하고 데이터를 보호하는 기능이 있습니다.

Oracle9i 서버는 포괄적인 개방형 통합 접근 방식을 사용하여 정보를 관리합니다. Oracle Server는 오라클 데이터베이스와 Oracle Server 인스턴스로 구성됩니다. 데이터베이스가 시작될 때마다 시스템 글로벌 영역(SGA)이 할당되고 오라클 백그라운드 프로세스가 시작됩니다. 시스템 글로벌 영역은 데이터베이스 사용자들이 데이터베이스 정보를 공유할 수 있도록 해주는 메모리 영역입니다. 백그라운드 프로세스와 메모리 버퍼가 결합된 형태를 오라클 인스턴스라고 합니다.

SQL 문

SELECT	데이터 검색
INSERT UPDATE DELETE MERGE	DML(데이터 조작어)
CREATE ALTER DROP RENAME TRUNCATE	DDL(데이터 정의어)
COMMIT ROLLBACK SAVEPOINT	트랜잭션 제어
GRANT REVOKE	DCL(데이터 제어어)

ORACLE

i-23

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL 문

오라클 SQL은 산업 채택 표준을 준수합니다. 오라클 사는 SQL 표준 위원회의 핵심 인사를 적극적으로 참여시킴으로써 변화하는 표준을 준수합니다. 산업 인정 위원회는 ANSI(American National Standards Institute)와 ISO(International Standards Organization)입니다. ANSI와 ISO는 SQL을 관계형 데이터베이스를 위한 표준어로 승인했습니다.

명령문	설명
SELECT	데이터베이스에서 데이터를 검색합니다.
INSERT UPDATE DELETE MERGE	데이터베이스의 테이블에서 새 행(row) 입력, 기존 행 변경 및 필요 없는 행 제거를 수행합니다. 이 명령어들을 DML(데이터 조작어)이라고 합니다.
CREATE ALTER DROP RENAME TRUNCATE	테이블에서 데이터 구조를 설정, 변경 및 제거합니다. 이 명령어들을 DDL(데이터 정의어)이라고 합니다.
COMMIT ROLLBACK SAVEPOINT	DML 문이 변경한 내용을 관리합니다. 데이터에 대한 변경 내용은 논리적인 트랜잭션으로 그룹화될 수 있습니다.
GRANT REVOKE	오라클 데이터베이스 및 해당 구조에 대한 액세스 권한을 부여하거나 제거합니다. 이 명령어들을 DCL(데이터 제어어)이라고 합니다.

이 과정에서 사용되는 테이블

EMPLOYEES

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	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	60
107	Diana	Lorentz	DLORENTZ	590.423.4567	07-FEB-99	IT_PROG	42
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	58
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	35
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	31
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	26
				0.121.2004	09-JUL-98	ST_CLERK	25

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

DEPARTMENTS

JOB_GRADES

ORACLE

이 과정에서 사용되는 테이블

이 과정에서는 다음 기본 테이블이 사용됩니다.

- 모든 사원의 정보를 제공하는 EMPLOYEES 테이블
- 모든 부서의 정보를 제공하는 DEPARTMENTS 테이블
- 다양한 등급에 대한 급여 정보를 제공하는 JOB_GRADES 테이블

참고: 모든 테이블의 구조와 데이터는 부록 B에 들어 있습니다.

GRADE	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

요약

- Oracle9i Server는 인터넷 컴퓨팅을 위한 데이터 베이스입니다.
- Oracle9i는 객체 관계형 데이터베이스 관리 시스템을 기반으로 합니다.
- 관계형 데이터베이스는 관계로 구성되며 관계 연산 및 데이터 무결성 제약 조건에 따라 관리됩니다.
- Oracle Server에서 SQL 언어 및 PL/SQL 엔진을 사용하여 정보를 저장하고 관리할 수 있습니다.

ORACLE

I-25

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

관계형 데이터베이스 관리 시스템은 객체 또는 관계로 구성되며 연산 및 데이터 무결성 제약 조건에 따라 관리됩니다.

오라클은 관계형 데이터베이스 관리 시스템의 요구에 부응하는 제품 및 서비스를 생산하고 제공합니다. 주요 제품으로는 SQL을 사용하여 정보를 저장하고 관리할 수 있는 Oracle9i Database Server와 모든 사용자 응용 프로그램을 실행 할 수 있는 Oracle9i Application Server가 있습니다.

SQL

Oracle Server는 ANSI 표준 SQL을 지원하며 확장된 SQL을 포함하고 있습니다. SQL은 서버와 통신하여 데이터를 액세스하고, 조작하고, 제어하는 데 사용되는 언어입니다.

1

기본 SQL
SELECT 문 작성

ORACLE

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- **SQL SELECT 문의 기능 설명**
- **기본 SELECT 문 실행**
- **SQL 문과 iSQL*Plus 명령 구별**

ORACLE

1-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

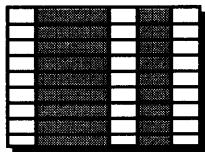
데이터베이스에서 데이터를 추출하려면 SQL SELECT 문을 사용해야 합니다. 이 때 표시되는 열을 제한해야 할 경우가 있습니다.

이 단원에서는 이러한 작업 수행에 필요한 모든 SQL 문을 설명합니다. 또한 여러 번 사용 가능한 SELECT 문을 작성할 수 있습니다. 이 단원에서는 SQL 문을 실행하는 iSQL*Plus 환경에 대해서도 설명합니다.

참고: iSQL*Plus는 Oracle9i 제품에 새롭게 추가되었으며, 브라우저 환경에서 SQL 명령을 실행하도록 제공되었습니다. 오라클의 이전 릴리스에서는 SQL*Plus가 SQL 명령을 실행하는 기본 환경이었습니다. SQL*Plus는 계속 사용할 수 있으며 이에 대해서는 부록 C에서 설명합니다.

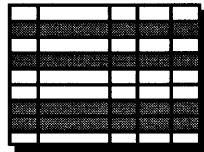
SQL SELECT 문의 기능

프로젝션



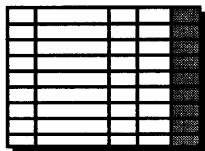
테이블 1

선택



테이블 1

선택 (Selection)



테이블 1

테이블 2

ORACLE

1-3

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL SELECT 문의 기능

SELECT 문은 데이터베이스에서 정보를 검색합니다. SELECT 문을 사용하여 다음을 수행할 수 있습니다.

- **프로젝션:** SQL의 프로젝션 기능을 사용하면 테이블에서 질의 결과로 반환될 열을 사용자가 필요한 만큼 선택할 수 있습니다.
- **선택:** SQL의 선택 기능을 사용하면 테이블에서 질의 결과로 반환될 행(row)을 선택할 수 있으며 다양한 조건을 사용하여 표시할 행을 제한할 수 있습니다.
- **조인:** SQL의 조인 기능을 사용하면 서로 다른 테이블 간에 링크를 생성하여 각 테이블에 저장된 데이터를 함께 가져올 수 있습니다. 조인에 대한 내용은 다른 단원에서 자세히 설명합니다.

기본 SELECT 문

```
SELECT * | { [DISTINCT] column|expression [alias], ... }  
FROM   table;
```

- **SELECT**는 표시할 대상 열을 지정합니다.
- **FROM**은 대상 열을 포함하는 해당 테이블을 지정합니다.

1-4

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

기본 SELECT 문

가장 간단한 형태의 SELECT 문은 다음을 포함해야 합니다.

- **SELECT** 절: 표시할 열을 지정합니다.
- **FROM** 절: SELECT 절에 나열된 열을 포함하는 테이블을 지정합니다.

구문 설명:

SELECT	하나 이상의 열로 구성되는 목록입니다.
*	모든 열을 선택합니다.
DISTINCT	중복되는 열을 생략합니다.
column/expression	지정한 열 또는 표현식을 선택합니다.
alias	선택된 열에 다른 머리를 이름을 줍니다.
FROM table	해당 열을 포함하는 테이블을 지정합니다.

참고: 이 과정 전체에서 키워드, 절, 문이라는 용어는 다음과 같이 사용됩니다.

- 키워드는 개별적인 SQL 요소입니다.
예를 들어, SELECT 및 FROM은 키워드입니다.
- 절은 SQL 문의 한 부분입니다.
예를 들어, SELECT employee_id, last_name, ... 은 절입니다.
- 문은 둘 이상의 절이 결합된 것입니다.
예를 들어, SELECT * FROM employees는 SQL 문입니다.

모든 열 선택

```
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.

ORACLE

1-5

Copyright © Oracle Corporation, 2001. All rights reserved.

모든 행의 모든 열 선택

SELECT 키워드 뒤에 별표(*)를 사용하면 테이블에 있는 데이터의 모든 열을 표시할 수 있습니다. 슬라이드 예제에서, 부서 테이블에는 DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID 및 LOCATION_ID라는 4개의 열이 포함되어 있으며 각 부서 당 하나씩 7개의 행이 포함되어 있습니다. 또한 SELECT 키워드 다음에 모든 열을 나열하여 테이블의 모든 열을 표시할 수도 있습니다. 예를 들어, 다음 SQL 문은 슬라이드 예제처럼 DEPARTMENTS 테이블의 모든 열과 모든 행을 표시합니다.

```
SELECT department_id, department_name, manager_id, location_id  
FROM departments;
```

특정 열 선택

```
SELECT department_id, location_id  
FROM departments;
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

ORACLE

1-6

Copyright © Oracle Corporation, 2001. All rights reserved.

모든 행의 특정 열 선택

SELECT 문을 사용할 때 해당 열 이름을 쉼표로 구분하여 지정하면 테이블의 특정 열을 표시할 수 있습니다. 슬라이드 예제는 DEPARTMENTS 테이블의 모든 부서 번호와 위치 번호를 표시합니다.

SELECT 절에서는 출력 결과에 표시할 순서대로 열을 지정합니다. 예를 들어, 왼쪽부터 오른쪽으로 나열할 때 부서 번호 앞에 위치를 표시하려면 다음 명령문을 사용합니다.

```
SELECT location_id, department_id  
FROM departments;
```

LOCATION_ID	DEPARTMENT_ID
1700	10
1800	20
1500	50

...

8 rows selected.

SQL 문 작성

- SQL 문은 대소문자를 구분하지 않습니다.
- SQL 문은 하나 이상의 줄에 입력할 수 있습니다.
- 키워드는 약어로 쓰거나 여러 줄에 나눠 쓸 수 없습니다.
- 절은 일반적으로 서로 다른 줄에 씁니다.
- 들여쓰기를 사용하면 SQL 문을 좀더 읽기 쉽게 작성할 수 있습니다.

ORACLE

1-7

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL 문 작성

다음의 간단한 규칙과 지침을 사용하면 읽기 쉽고 편집하기 쉬운 유용한 문을 작성할 수 있습니다.

- SQL 문은 특별히 표시하지 않는 한 대소문자를 구분하지 않습니다.
- SQL 문은 하나 이상의 줄에 입력할 수 있습니다.
- 키워드는 다음 줄에 나눠 쓰거나 약어로 쓸 수 없습니다.
- 절은 일반적으로 읽기 쉽고 편집하기 쉽도록 서로 다른 줄에 씁니다.
- 들여쓰기를 사용하면 좀더 읽기 쉬운 SQL 문을 작성할 수 있습니다.
- 일반적으로 키워드는 대문자로 입력하고 테이블 이름, 열 등 다른 단어는 모두 소문자로 입력합니다.

SQL 문 실행

iSQL*Plus를 사용하는 경우, 편집 창에서 Execute 버튼을 눌러 명령을 실행합니다.

열 머리글 기본값

• *iSQL*Plus:*

- 기본 머리글 정렬: 가운데
- 기본 머리글 표시: 대문자

• **SQL*Plus:**

- 문자 및 날짜 열 머리글: 왼쪽 정렬
- 숫자 열 머리글: 오른쪽 정렬
- 기본 머리글 표시: 대문자

1-8

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

열 머리글 기본값

*iSQL*Plus*에서, 열 머리글은 가운데로 정렬되어 대문자로 표시됩니다.

```
SELECT last_name, hire_date, salary
  FROM employees;
```

LAST_NAME	HIRE_DATE	SALARY
King	17-JUN-87	24000
Kochhar	21-SEP-89	17000
De Haan	13-JAN-93	17000
Hunold	03-JAN-90	9000
Ernst	21-MAY-91	6000
■ ■ ■		
Higgins	07-JUN-94	12000
Gietz	07-JUN-94	8300

20 rows selected.

열 머리글 표시에 별칭을 사용할 수 있습니다. 열 별칭에 대해서는 이 단원의 뒷부분에서 설명합니다.

산술식

산술 연산자를 사용하여 숫자 및 날짜 데이터에 대한 표현식을 작성합니다.

연산자	설명
+	더하기
-	빼기
*	곱하기
/	나누기

ORACLE

1-9

Copyright © Oracle Corporation, 2001. All rights reserved.

산술식

데이터 표시 방식을 수정하거나 계산을 수행하거나 가정 시나리오를 참고해야 할 경우 산술식을 사용할 수 있습니다. 산술식에는 열 이름, 상수 숫자 값 및 산술 연산자가 포함될 수 있습니다.

산술 연산자

슬라이드에는 SQL에서 사용 가능한 산술 연산자가 나와 있습니다. FROM 절을 제외한 SQL 문의 모든 절에서 산술 연산자를 사용할 수 있습니다.

산술 연산자 사용

```
SELECT last_name, salary, salary + 300  
FROM employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

Hartstein	13000	13300
Fay	6000	6300
Higgins	12000	12300
Gietz	8300	8600

20 rows selected.

ORACLE

산술 연산자 사용

슬라이드 예제는 덧셈 연산자를 사용하여 모든 사원에 대해 \$300의 급여 인상을 계산한 후 출력 결과에 새 열 SALARY+300을 표시합니다.

계산 결과로 나타나는 SALARY+300 열은 EMPLOYEES 테이블에 새로 생성되지 않고 표시만 됩니다. 기본적으로 새 열의 이름은 슬라이드 예제의 salary+300과 같이 해당 열을 생성하는 계산식을 기반으로 지정됩니다.

참고: Oracle9i server는 산술 연산자 앞뒤의 공백을 무시합니다.

연산자 우선순위

* / + -

- 곱셈과 나눗셈은 덧셈과 뺄셈보다 우선순위가 높습니다.
- 우선순위가 동일한 연산자는 왼쪽에서 오른쪽으로 계산됩니다.
- 괄호는 우선적으로 계산되며 명령문을 명확히 나타내는 역할을 합니다.

ORACLE

1-11

Copyright © Oracle Corporation, 2001. All rights reserved.

연산자 우선순위

산술식에 여러 연산자가 있을 경우 곱셈과 나눗셈이 먼저 계산되며 표현식 내에서 우선순위가 동일한 연산자는 왼쪽에서 오른쪽으로 계산됩니다.

괄호를 사용하면 괄호 안의 표현식이 먼저 계산됩니다.

연산자 우선순위

```
SELECT last_name, salary, 12*salary+100  
FROM employees;
```

LAST_NAME	SALARY	12*SALARY+100
King	24000	289100
Kochhar	17000	204100
De Haan	17000	204100
Hunold	9000	108100
Ernst	6000	72100

Hartstein	13000	156100
Fay	6000	72100
Higgins	12000	144100
Gietz	6300	99700

20 rows selected.

ORACLE®

연산자 우선순위(계속)

슬라이드 예제는 사원의 이름, 급여 및 연간 총수입을 표시합니다. 예제에서 연간 총수입은 월급에 12를 곱한 후 \$100의 상여금을 한번 더해 계산되며 곱셈이 덧셈보다 먼저 계산됩니다.

참고: 팔호를 사용하면 우선순위를 보다 명확하게 나타낼 수 있습니다. 예를 들어, 슬라이드의 산술식을 $(12*salary)+100$ 으로 작성해도 결과는 동일합니다.

괄호 사용

```
SELECT last_name, salary, 12*(salary+100)
FROM employees;
```

LAST_NAME	SALARY	12*(SALARY+100)
King	24000	288200
Kochhar	17000	205200
De Haan	17000	205200
Hunold	9000	109200
Ernst	6000	73200

Hartstein	13000	157200
Fay	6000	73200
Higgins	12000	145200
Gietz	6300	100600

20 rows selected.

ORACLE

1-13

Copyright © Oracle Corporation, 2001. All rights reserved.

괄호 사용

괄호를 사용하면 우선순위 규칙을 무시하고 연산자의 실행 순서를 지정할 수 있습니다.

슬라이드 예제는 사원의 이름, 급여 및 연간 총수입을 표시합니다. 예제에서 연간 총수입은 월급에 상여금 \$100를 더한 후 12를 곱해 계산되는데 괄호로 인해 덧셈의 우선순위가 곱셈 보다 높습니다.

널 값 정의

- 널 값은 알 수 없는 값, 사용할 수 없는 값, 할당할 수 없는 값, 적용할 수 없는 값을 의미합니다.
- 널은 0 또는 공백과 다릅니다.

```
SELECT last_name, job_id, salary, commission_pct  
FROM employees;
```

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	

Zlotkey	SA_MAN	10500	2
Abel	SA_REP	11000	3
Taylor	SA_REP	8800	2

Gietz	AC_ACCOUNT	8300	

20 rows selected.

ORACLE

널 값

한 행의 특정 열에 데이터 값이 없으면 그 값을 널이라고 하거나 널을 포함한다고 합니다.

널은 알 수 없는 값, 사용할 수 없는 값, 할당할 수 없는 값, 적용할 수 없는 값을 의미하며 0이나 공백과는 다릅니다. 0은 숫자고 공백은 하나의 문자입니다.

모든 데이터 유형의 열에는 널이 포함될 수 있습니다. 하지만 NOT NULL 및 PRIMARY KEY 등의 제약 조건이 지정된 열에는 널을 사용할 수 없습니다.

EMPLOYEES 테이블의 COMMISSION_PCT 열의 경우 영업 과장과 영업 사원만이 커미션을 받을 수 있습니다. 다른 사원들은 커미션을 받을 수 없는데, 이 경우 널을 사용합니다.

산술식의 널 값

널 값을 포함하는 산술식은 널로 평가됩니다.

```
SELECT last_name, 12*salary*commission_pct  
FROM employees;
```

LAST_NAME	12*Salary*Commission_Pct
King	
Kochhar	
...	
Zlotkey	25200
Abel	39600
Taylor	20640
...	
Gietz	

20 rows selected.

ORACLE

1-15

Copyright © Oracle Corporation, 2001. All rights reserved.

널 값(계속)

산술식에서 열의 값이 널이면 결과는 널입니다. 예를 들어, 숫자를 0으로 나누면 오류가 발생하지만 널로 나누면 결과는 널이거나 알 수 없습니다.

슬라이드 예제에서 사원 King은 커미션을 받지 않습니다. 이 경우 산술식에서 COMMISSION_PCT 열이 널이 되므로 결과는 널입니다.

자세한 내용은 *Oracle9i SQL Reference*, “Basic Elements of SQL”을 참조하십시오.

열 별칭 정의

Defn.

열 별칭:

- 열 머리글의 이름을 변경합니다.
- 계산식에 대한 열머리를 지정할 때 유용합니다.
- 열 이름 바로 뒤에 사용합니다. 열 이름과 별칭 사이에 선택적으로 AS 키워드를 사용할 수 있습니다.
- 공백 또는 특수 문자가 있거나 대소문자를 구분할 경우 큰 따옴표를 사용합니다.

ORACLE

열 별칭

질의 결과를 표시할 때 iSQL*Plus는 일반적으로 선택한 열의 이름을 열 머리글로 사용합니다. 이러한 머리글은 내용을 제대로 설명하지 못하므로 이해하기 어렵습니다. 열 별칭을 사용하면 열 머리글을 변경할 수 있습니다.

SELECT 목록에서 열 다음에 공백을 구분자로 사용하여 별칭을 지정합니다. 기본적으로 별칭 머리글은 대문자로 표시됩니다. 별칭이 공백 또는 특수 문자(# 또는 \$ 등)를 포함하거나 대소문자를 구분할 경우에는 큰 따옴표(" ")로 묶어야 합니다.

열 별칭 사용

```
SELECT last_name AS [name], commission_pct [comm]  
FROM employees;
```

NAME	COMM
King	
Kochhar	
De Haan	

20 rows selected.

```
SELECT last_name ["Name"], salary*12 ["Annual Salary"]  
FROM employees;
```

Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

20 rows selected.

ORACLE

1-17

Copyright © Oracle Corporation, 2001. All rights reserved.

열 별칭(계속)

첫번째 예제는 모든 사원의 이름과 커미션 비율을 표시합니다. 예제에서는 선택 사항인 AS 키워드가 열 별칭 이름 앞에 사용되었는데 질의 결과는 AS 키워드의 사용에 관계 없이 동일합니다. 또한 SQL 문에 열 별칭인 name 및 comm이 소문자로 되어 있지만 질의 결과에서는 열 머리글이 대문자로 표시됩니다. 이전 슬라이드에서 언급했듯이 열 머리글은 기본적으로 대문자로 표시됩니다.

두번째 예제는 모든 사원의 이름과 연봉을 표시합니다. Annual Salary에는 공백이 포함되어 있으므로 큰 따옴표로 묶여 있습니다. 출력 결과를 보면 열 머리글이 열 별칭과 같음을 알 수 있습니다.

연결 연산자

연결 연산자:

- 열 또는 문자열을 다른 열에 연결합니다.
- 두 개의 세로선(||)으로 표시합니다.
- 문자식에 해당하는 결과 열을 생성합니다.

ORACLE

1-18

Copyright © Oracle Corporation, 2001. All rights reserved.

연결 연산자

연결 연산자(||)를 사용하면 열을 다른 열, 산술식 또는 상수 값에 연결하여 문자식을 생성할 수 있습니다. 이 연산자의 좌우에 있는 열이 결합되어 하나의 열로 출력됩니다.

연결 연산자 사용

```
SELECT last_name||job_id AS "Employees"  
FROM employees;
```

Employees
KingAD_PRES
KochharAD_VP
De HaanAD_VP
HunoldIT_PROG
ErnstIT_PROG
LorentzIT_PROG
MourgosST_MAN
RajsST_CLERK
...

20 rows selected.

ORACLE

1-13

Copyright © Oracle Corporation, 2001. All rights reserved.

연결 연산자(계속)

예제에서 LAST_NAME과 JOB_ID가 연결되고 별칭으로 Employees가 지정되었습니다. 사원의 이름과 업무 코드가 결합되어 하나의 열로 출력됩니다.

별칭 이름 앞의 AS 키워드로 인해 SELECT 절을 파악하기가 쉽습니다.

리터럴 문자열

- 리터럴은 SELECT 목록에 포함된 문자, 숫자 또는 날짜입니다.
- 날짜 및 문자 리터럴 값은 작은 따옴표로 묶어야 합니다.
- 각 문자열은 각 행(row)이 반환될 때마다 한 번씩 출력됩니다.

ORACLE

1-20

Copyright © Oracle Corporation, 2001. All rights reserved.

리터럴 문자열

리터럴은 열 이름이나 열 별칭이 아니면서 SELECT 목록에 포함된 문자, 숫자 또는 날짜이며 각 행이 반환될 때마다 출력됩니다. 자유로운 형식의 텍스트인 리터럴 문자열은 질의 결과에 포함되어 SELECT 목록의 열과 동일하게 취급됩니다.

날짜 및 문자 리터럴은 반드시 작은 따옴표(' ')로 묶어야 하지만 숫자 리터럴은 묶지 않아도 됩니다.

리터럴 문자열 사용

```
SELECT last_name || ' is a ' || job_id  
      AS "Employee Details"  
  FROM employees;
```

Employee Details
King is a AD_PRES
Kochhar is a AD_VP
De Haan is a AD_VP
Hunold is a IT_PROG
Ernst is a IT_PROG
Lorentz is a IT_PROG
Mourgos is a ST_MAN
Rajs is a ST_CLERK
...
20 rows selected.

ORACLE

1-21

Copyright © Oracle Corporation, 2001. All rights reserved.

리터럴 문자열(계속)

슬라이드 예제는 모든 사원의 이름과 업무 코드를 표시하며 열 머리글은 Employee Details입니다. SELECT 문에서 작은 따옴표 사이에 공백을 두어 출력 결과를 알아보기 쉽도록 하였습니다.

다음 예제에서 각 사원의 이름과 급여는 리터럴로 연결되어 반환되는 행(row)에 더 많은 의미를 부여합니다.

```
SELECT last_name || ': 1 Month salary = ' || salary Monthly  
      FROM employees;
```

MONTHLY
King: 1 Month salary = 24000
Kochhar: 1 Month salary = 17000
De Haan: 1 Month salary = 17000
Hunold: 1 Month salary = 9000
Ernst: 1 Month salary = 6000
Lorentz: 1 Month salary = 4200
Mourgos: 1 Month salary = 5800
Rajs: 1 Month salary = 3500
...
20 rows selected.

중복 행

질의는 기본적으로 중복 행을 포함한 모든 행을 표시합니다.

```
SELECT department_id  
FROM employees;
```

DEPARTMENT ID	
	90
	90
	90
	90
	60
	60
	60
	50
	50
	50
	50

20 rows selected.

ORACLE

1-22

Copyright © Oracle Corporation, 2001. All rights reserved.

중복 행(row)

특별히 표시하지 않는 한 *iSQL*Plus*는 중복 행을 제거하지 않은 상태로 질의 결과를 표시합니다. 슬라이드 예제는 EMPLOYEES 테이블의 모든 부서 번호를 표시하며 부서 번호가 반복되어 있음을 볼 수 있습니다.

중복 행 제거

SELECT 절에서 DISTINCT 키워드를 사용하여 중복 행을 제거합니다.

```
SELECT DISTINCT department_id  
FROM employees;
```

DEPARTMENT_ID
10
20
30
40
50
60
70
80
90
110

8 rows selected.

ORACLE

1-23

Copyright © Oracle Corporation, 2001. All rights reserved.

중복 행(계속)

결과에서 중복 행을 제거하려면 SELECT 절에서 SELECT 키워드 바로 다음에 DISTINCT 키워드를 넣어야 합니다. 슬라이드 예제에서 EMPLOYEES 테이블에는 실제로 행이 20개 포함되어 있지만 고유한 부서 번호는 7개뿐입니다.

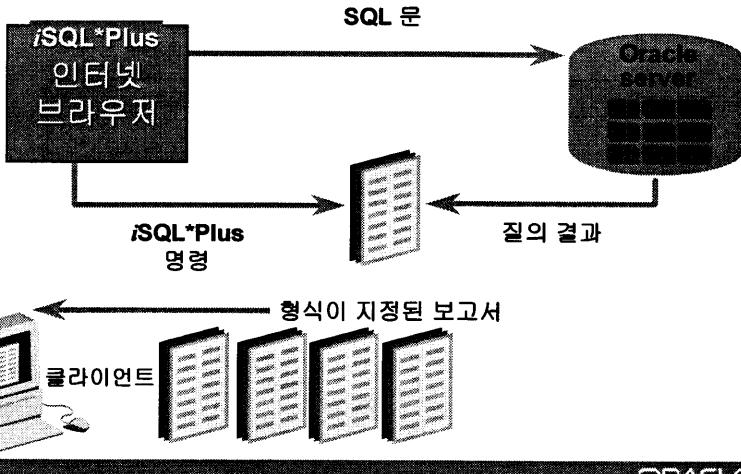
DISTINCT 수식자 다음에 여러 열을 지정할 수 있습니다. DISTINCT 수식자는 선택한 모든 열에 영향을 주므로 결과에는 모든 가능한 열 조합 중 고유한 열 조합이 모두 표시됩니다.

```
SELECT DISTINCT department_id, job_id  
FROM employees;
```

DEPARTMENT_ID	JOB_ID
10	AD_ASST
20	MK_MAN
20	MK_REP
50	ST_CLERK
50	ST_MAN
60	IT_PROG
...	
	SA_REP

13 rows selected.

SQL과 iSQL*Plus의 상호 작용



1-24

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

SQL 및 iSQL*Plus

SQL은 틀이나 응용 프로그램에서 Oracle server와 통신하기 위한 명령어입니다. 오라클 SQL은 많은 확장 기능을 갖추고 있습니다.

iSQL*Plus는 SQL 문을 인식하고 Oracle server에 전달하여 실행할 수 있도록 하는 오라클 틀이며 자체 명령어를 포함합니다.

SQL 특징

- 프로그래밍 경험이 거의 없거나 전혀 없는 사용자를 포함한 다양한 사용자 층에서 사용 할 수 있습니다.
- 비절차적 언어입니다.
- 시스템 생성 및 관리에 필요한 시간을 줄여줍니다.
- 영어 형식의 언어입니다.

iSQL*Plus 특징

- 브라우저에서 액세스합니다.
- 여러 SQL문을 하나의 목록으로 구성하여 사용할 수 있습니다.
- 온라인으로 SQL 문을 편집하여 수정할 수 있습니다.
- 환경 설정을 제어합니다.
- 질의 결과의 형식을 기본 보고서로 지정합니다.
- 로컬 및 원격 데이터베이스를 액세스합니다.

SQL 문과 iSQL*Plus 명령 비교

SQL

- 언어
- **ANSI 표준**
- 키워드를 약어로 쓸 수 없음
- 명령문으로 데이터베이스의 데이터 및 테이블 정의를 조작할 수 있음

iSQL*Plus

- 환경
- 오라클 전용
- 키워드를 약어로 쓸 수 있음
- 명령으로 데이터베이스의 값을 조작할 수 없음
- 브라우저에서 실행
- 중앙에서 로드되므로 각 컴퓨터에서 구현할 필요 없음

SQL
문

iSQL*Plus
명령

ORACLE

1-25

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL 및 iSQL*Plus(계속)

다음 표는 SQL과 iSQL*Plus를 비교한 것입니다.

SQL	iSQL*Plus
데이터 액세스를 위해 Oracle server와 통신하는 언어	SQL 문을 인식하여 서버에 전달
ANSI(American National Standards Institute) 표준 SQL 기반	SQL 문을 실행하기 위한 오라클 전용 인터페이스
데이터베이스의 데이터 및 테이블 정의를 조작할 수 있음	데이터베이스의 값을 조작할 수 없음
연결 문자 없음	명령이 한 줄보다 길 경우 대시(-)를 연결 문자로 사용
약어 사용 불가	약어 사용 가능
함수를 사용하여 일부 형식 지정 수행	명령을 사용하여 데이터 형식 지정

iSQL*Plus 개요

iSQL*Plus에 로그인하면 다음을 수행할 수 있습니다.

- 테이블 구조를 표시합니다.
- SQL 문을 편집합니다.
- iSQL*Plus에서 SQL을 실행합니다.
- SQL 문을 파일로 저장하고 파일에 SQL 문을 추가합니다.
- 파일에 저장된 명령문을 실행합니다.
- 텍스트 파일에서 iSQL*Plus 편집 창으로 명령을 로드합니다.

ORACLE

1-26

Copyright © Oracle Corporation, 2001. All rights reserved.

iSQL*Plus

iSQL*Plus는 다음 작업을 수행할 수 있는 환경입니다.

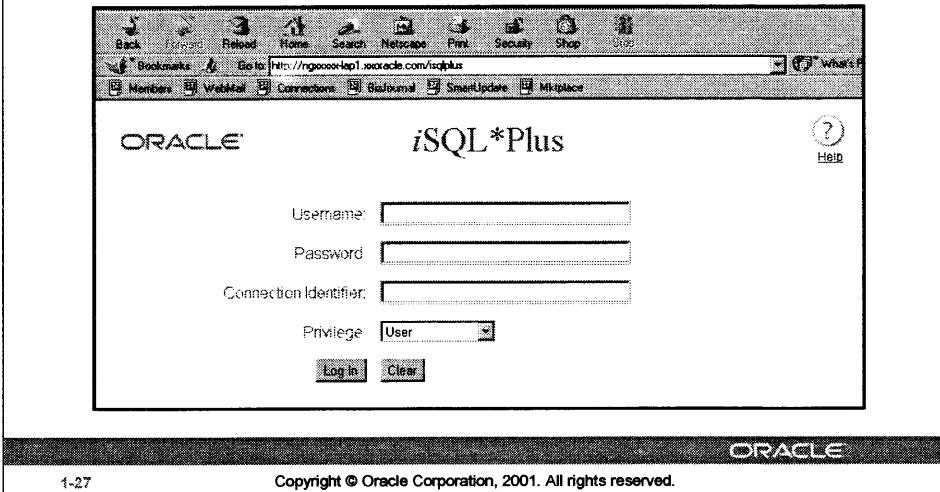
- SQL 문을 실행하여 데이터베이스에서 데이터를 검색, 수정, 추가 및 제거할 수 있습니다.
- 질의 결과에 대해 형식 지정, 계산 수행, 저장 및 보고서 형태로 인쇄할 수 있습니다.
- SQL 문을 저장하는 스크립트 파일을 작성하여 나중에 반복 사용할 수 있습니다.

iSQL*Plus 명령은 다음과 같은 기본 범주로 나뉩니다.

범주	용도
환경	해당 세션에서 SQL 문의 기본 동작에 영향을 미칩니다.
형식	질의 결과의 형식을 지정합니다.
파일 조작	명령문을 텍스트 스크립트 파일로 저장하고 텍스트 스크립트 파일에서 명령문을 실행합니다.
실행	브라우저에서 Oracle server로 SQL 문을 전달합니다.
편집	편집 창에서 SQL 문을 수정합니다.
상호 작용	변수를 작성하여 SQL 문에 전달하고, 변수 값을 출력하고, 메시지를 화면에 표시하도록 합니다.
기타	데이터베이스 연결, iSQL*Plus 환경 조작, 열 정의 표시 등을 위한 다양한 명령을 제공합니다.

iSQL*Plus에 로그인

Windows 브라우저 환경에서 로그인

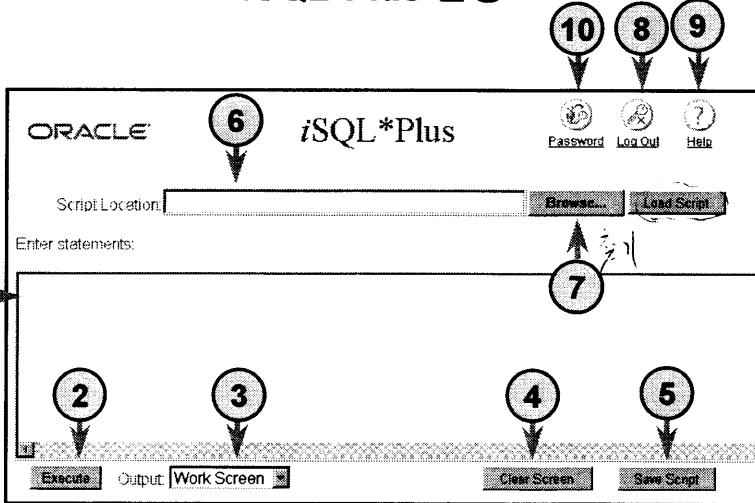


iSQL*Plus에 로그인

브라우저 환경을 통해 로그인 하려면

1. 브라우저를 시작합니다.
2. iSQL*Plus 환경의 URL 주소를 입력합니다.
3. username, password 및 Oracle Connection Identifier 항목을 입력 합니다.

iSQL*Plus 환경



iSQL*Plus 환경

Windows 브라우저 내의 iSQL*Plus 창에는 다음과 같은 몇 가지 주요 영역이 있습니다.

1. 편집 창: SQL 문과 iSQL*Plus 명령을 입력하는 영역입니다.
2. Execute 버튼: 편집 창의 명령문 및 명령을 실행할 때 누릅니다.
3. Output 옵션: 기본값은 Work Screen이며, SQL 문의 결과를 편집 창 아래에 표시합니다. 다른 옵션으로는 File 또는 Window가 있습니다. File 옵션은 내용을 지정된 파일로 저장하며 Window 옵션은 출력을 화면의 별도 창에 표시합니다.
4. Clear Screen 버튼: 편집 창의 텍스트를 지울 때 누릅니다.
5. Save Script 버튼: 편집 창의 내용을 파일로 저장합니다.
6. Script Location: 실행할 스크립트 파일의 이름과 위치를 지정합니다.
7. Browse 버튼: Windows 파일 열기 대화 상자를 사용하여 스크립트 파일을 선택하는 데 사용합니다.
8. 종료 아이콘: iSQL*Plus 세션을 종료하고 iSQL*Plus LogOn 창으로 돌아갈 때 누릅니다.
9. 도움말 아이콘: iSQL*Plus 도움말 문서에 대한 액세스를 제공합니다.
10. Password 버튼: 암호를 변경할 때 사용합니다.

테이블 구조 표시

iSQL*Plus DESCRIBE 명령을 사용하여 테이블 구조를 표시합니다.

```
DESC [RIBE] tablename
```

ORACLE

1-29

Copyright © Oracle Corporation, 2001. All rights reserved.

테이블 구조 표시

iSQL*Plus에서 DESCRIBE 명령을 사용하면 테이블 구조를 표시할 수 있습니다. 이 명령은 열 이름과 데이터 유형뿐 아니라 열에 반드시 데이터가 포함되어야 하는지 여부도 표시합니다.

구문 설명:

tablename 사용자가 액세스할 수 있는 기존 테이블, 뷰 또는 동의어의 이름입니다.

테이블 구조 표시

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)

ORACLE

테이블 구조 표시(계속)

슬라이드 예제는 DEPARTMENTS 테이블의 구조에 대한 정보를 표시합니다.

결과 설명:

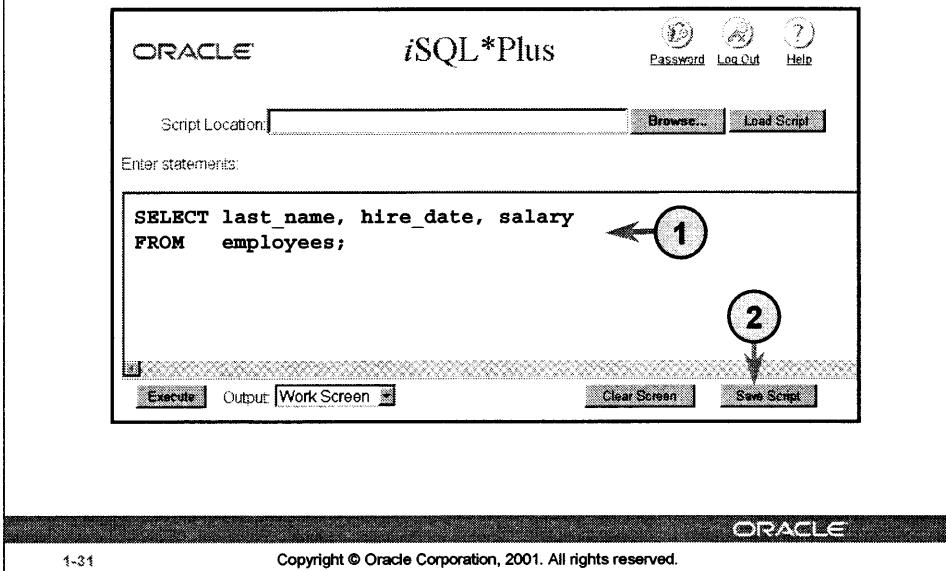
Null? 열에 반드시 데이터가 포함되어야 하는지를 나타냅니다. NOT NULL이면
열에 반드시 데이터가 포함되어야 합니다.

Type 열의 데이터 유형을 표시합니다.

다음 표는 데이터 유형을 설명합니다.

데이터 유형	설명
NUMBER (p, s)	최대 자릿수가 p고 소수점 이하 자릿수가 s인 숫자 값입니다.
VARCHAR2 (s)	최대 크기가 s인 가변 길이 문자 값입니다.
DATE	B.C. 4712년 1월 1일 ~ A.D. 9999년 12월 31일 사이의 날짜 및 시간 값입니다.
CHAR (s)	크기가 s인 고정 길이 문자 값입니다.

스크립트 파일과의 상호 작용



스크립트 파일과의 상호 작용

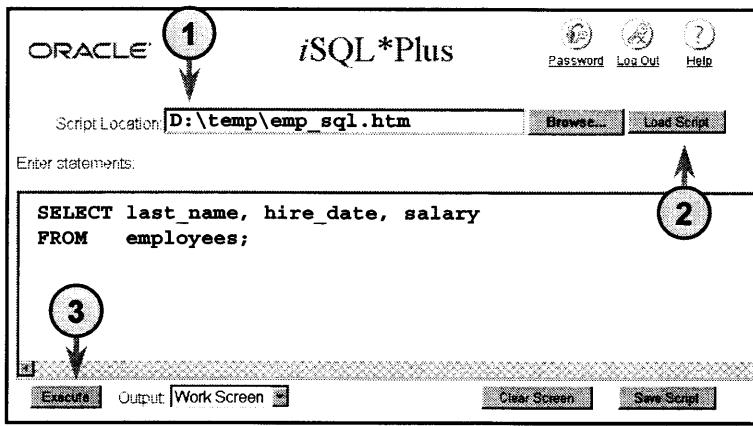
명령문 및 명령을 텍스트 스크립트 파일에 넣기

iSQL*Plus 편집 창의 명령 및 명령문을 텍스트 스크립트 파일로 저장하는 과정은 다음과 같습니다.

1. iSQL*Plus에서 SQL 문을 편집 창에 입력합니다.
2. Save Script 버튼을 누르면 Windows 파일 저장 대화상자가 열립니다. 파일의 이름을 지정합니다. 기본 확장자는 .html이며 확장을 변경하여 파일을 텍스트 파일이나 .sql 파일로 저장할 수 있습니다.



스크립트 파일과의 상호 작용



1-32

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

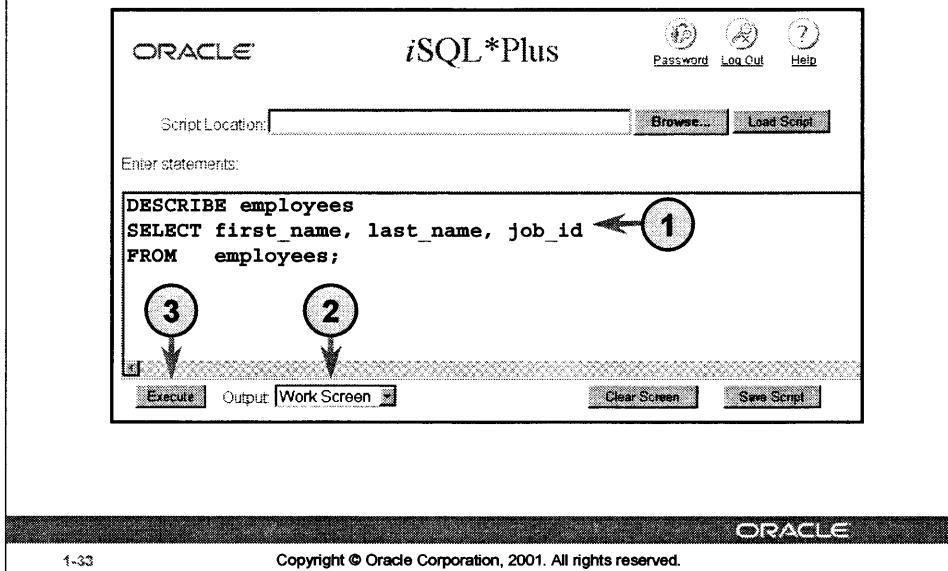
스크립트 파일과의 상호 작용

iSQL*Plus에서 스크립트 파일의 명령문 및 명령 사용하기

iSQL*Plus에서 이전에 저장한 스크립트 파일의 명령 및 명령문을 사용하는 방법은 다음과 같습니다.

1. 스크립트 이름과 위치를 입력합니다. 또는 Browse 버튼을 눌러 스크립트 이름과 위치를 찾을 수 있습니다.
2. Load Script 버튼을 누릅니다. 파일의 내용이 iSQL*Plus 편집 창으로 로드됩니다.
3. Execute 버튼을 눌러 iSQL*Plus 편집 창의 내용을 실행합니다.

스크립트 파일과의 상호 작용

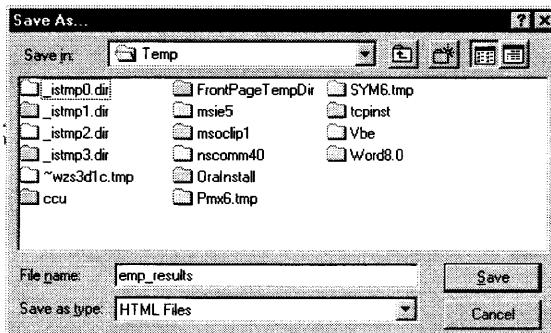


스크립트 파일과의 상호 작용

출력결과를 파일로 저장하기

SQL 문 또는 iSQL*Plus 명령으로 생성된 결과를 파일에 저장할 수 있습니다.

1. SQL 문 및 iSQL*Plus 명령을 iSQL*Plus의 편집 창에 입력합니다.
2. Output 옵션을 Save로 변경합니다.
3. Execute 버튼을 눌러 iSQL*Plus 편집 창의 내용을 실행하면 Windows 파일 저장 대화상자가 열립니다. 파일의 이름을 지정합니다. 기본 확장자는 .html이며 파일 형식을 변경할 수 있습니다. 결과가 지정된 파일로 전달됩니다.



요약

이 단원에서는 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- **SELECT 문 작성**
 - 테이블에서 모든 행(row) 및 열 반환
 - 테이블에서 특정 열 반환
 - 열 별칭을 사용하여 구체적인 열 머리를 부여
- **iSQL*Plus 환경을 사용하여 SQL 문 및 iSQL*Plus 명령을 작성, 저장 및 실행**

```
SELECT * | { [DISTINCT] column / expression [alias], ... }  
FROM   table;
```

ORACLE

1-34

Copyright © Oracle Corporation, 2001. All rights reserved.

SELECT 문

이 단원에서는 SELECT 문을 사용하여 데이터베이스 테이블에서 데이터를 검색하는 방법을 설명했습니다.

```
SELECT * | { [DISTINCT] column [alias], ... }  
FROM   table;
```

구문 설명:

SELECT	하나 이상의 열로 구성되는 목록입니다.
*	모든 열을 선택합니다.
DISTINCT	중복되는 열을 생략합니다.
column / expression	명명된 열 또는 표현식을 선택합니다.
alias	선택된 열에 다른 머리글을 부여합니다.
FROM table	해당 열을 포함하는 테이블을 지정합니다.

iSQL*Plus

iSQL*Plus는 SQL 문을 데이터베이스 서버로 전송하고 SQL 문을 편집 및 저장하는 데 사용되는 실행 환경입니다. 명령문은 SQL 프롬프트나 스크립트 파일에서 실행할 수 있습니다.

참고: SQL*Plus 환경은 부록 C에서 설명합니다.

연습 1 개요

이 연습에서는 다음 내용을 다룹니다.

- 여러 테이블에 있는 모든 데이터 선택
- 테이블 구조 표시
- 산술 계산 수행 및 열 이름 지정
- *SQL*Plus* 사용

1-35

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

연습 1 개요

이것은 여러 연습 과정 중 첫번째 과정입니다. 필요한 경우 부록 A의 해답을 참조하십시오. 연습은 단원에서 설명한 모든 내용을 소개하며 문제 2-4는 실습이 필요 없는 간단한 문제입니다.

모든 연습에는 “시간이 있을 때” 또는 “좀 더 연습이 필요한 경우” 풀어볼 수 있는 문제가 있습니다. 이러한 문제는 할당된 시간에 다른 문제를 모두 풀 후 사용법을 좀 더 익히고 싶을 때 풀어보십시오.

천천히 정확하게 연습하십시오. 명령 파일을 저장하고 실행해 볼 수 있습니다. 문제가 있으 면 언제든지 강사에게 물어 보십시오.

문제

문제 2-4는 True 또는 False로 답하십시오.

연습 1

1. 강사가 제공한 사용자 ID와 암호를 사용하여 iSQL*Plus 세션을 시작합니다.
2. iSQL*Plus 명령으로 데이터베이스에 액세스합니다.
True/False
3. 다음 SELECT 문은 성공적으로 실행됩니다.

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

True/False

4. 다음 SELECT 문은 성공적으로 실행됩니다.

```
SELECT *  
FROM   job_grades;
```

True/False

5. 이 명령문에는 코딩 오류가 네 개 있습니다. 네 가지의 코딩오류는 무엇입니까?

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

6. DEPARTMENTS 테이블의 구조를 표시하고 테이블의 모든 데이터를 검색하십시오.

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

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.

연습 1(계속)

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.

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)

8. lab1_7.sql 파일의 질의를 실행하십시오.

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

206	Gietz	AC_ACCOUNT	07-JUN-94

20 rows selected.

연습 1(계속)

9. EMPLOYEES 테이블의 업무 코드를 중복되지 않게 표시하는 질의를 작성하십시오.

JOB_ID
AC_ACCOUNT
AC_MGR
AD_ASST
AD_PRES
AD_VP
IT_PROG
MK_MAN
MK_REP
SA_MAN
SA_REP
ST_CLERK
ST_MAN

12 rows selected.

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.

Emp #	Employee	Job	Hire Date
100	King	AD_PRES	17-JUN-87
101	Kochhar	AD_VP	21-SEP-89
102	De Haan	AD_VP	13-JAN-93
103	Hunold	IT_PROG	03-JAN-90
104	Ernst	IT_PROG	21-MAY-91
107	Lorentz	IT_PROG	07-FEB-99
124	Mourgos	ST_MAN	16-NOV-99
141	Rajs	ST_CLERK	17-OCT-95
142	Davies	ST_CLERK	29-JAN-97
143	Matos	ST_CLERK	15-MAR-98
144	Vargas	ST_CLERK	09-JUL-98
206	Gietz	AC_ACCOUNT	07-JUN-94

20 rows selected.

연습 1(계속)

11. 업무 ID와 이름을 연결한 다음 쉼표 및 공백으로 구분하여 표시하고 열 이름을 Employee and Title로 지정하십시오.

Employee and Title	
King, AD_PRES	
Kochhar, AD_VP	
De Haan, AD_VP	
Hunold, IT_PROG	
Ernst, IT_PROG	
Lorentz, IT_PROG	
Mourgos, ST_MAN	
Rajs, ST_CLERK	
Davies, ST_CLERK	

Gietz, AC_ACCOUNT	

20 rows selected.

좀더 연습하려면 다음 문제를 풀어보십시오.

12. EMPLOYEES 테이블의 모든 데이터를 표시하는 질의를 작성하십시오. 각 열은 쉼표로 구분하고 열 이름은 THE_OUTPUT으로 지정하십시오.

THE_OUTPUT	
100,Steven,King,SKING,515.123.4567,AD_PRES,,17-JUN-87,24000,,90	
101,Neena,Kochhar,NKOCHHAR,515.123.4568,AD_VP,100,21-SEP-89,17000,,90	
102,Lex,De Haan,LDEHAAN,515.123.4569,AD_VP,100,13-JAN-93,17000,,90	
103,Alexander,Hunold,AHUNOLD,590.423.4567,IT_PROG,102,03-JAN-90,9000,,60	
104,Bruce,Ernst,BERNST,590.423.4568,IT_PROG,103,21-MAY-91,6000,,60	
107,Diana,Lorentz,DLORENTZ,590.423.5567,IT_PROG,103,07-FEB-99,4200,,60	
124,Kevin,Mourgos,KMOURGOS,650.123.5234,ST_MAN,100,16-NOV-99,5800,,50	
141,Trenna,Rajs,TRAJS,650.121.8009,ST_CLERK,124,17-OCT-95,3500,,50	
142,Curtis,Davies,CDAVIES,650.121.2994,ST_CLERK,124,29-JAN-97,3100,,50	
143,Randall,Matos,RMATOS,650.121.2874,ST_CLERK,124,15-MAR-98,2600,,50	
144,Peter,Vargas,PVARGAS,650.121.2004,ST_CLERK,124,09-JUL-98,2500,,50	

206,William,Gietz,WGIETZ,515.123.8181,AC_ACCOUNT,205,07-JUN-94,8300,,110	

20 rows selected.

데이터 제한 및 정렬

2

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 질의로 검색되는 행 제한
- 질의로 검색되는 행 정렬

ORACLE

2-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

데이터베이스에서 데이터 검색 중에 표시되는 데이터 행을 제한하거나 행 표시 순서를 지정해야 할 경우가 있습니다. 이 단원에서는 이러한 작업 수행에 사용되는 SQL 문을 설명합니다.

선택을 위한 행 제한

EMPLOYEES

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	Hunold	IT_PROG	60
104	Ernst	IT_PROG	60
107	Lorentz	IT_PROG	60
124	Mourgos	ST_MAN	50
...			

20 rows selected.

“부서 90의 모든 사원 검색”

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

ORACLE

2-3

Copyright © Oracle Corporation, 2001. All rights reserved.

선택을 위한 행 제한

슬라이드 예제에서 부서 90의 모든 사원을 표시하도록 하면 DEPARTMENT_ID 열에서 값이 90인 행만 반환됩니다. 이러한 제한 방법이 SQL에서 기본적인 WHERE 절의 기능입니다.

선택되는 행 제한

- WHERE 절을 사용하여 반환되는 행을 제한합니다.

```
SELECT * | { [DISTINCT] column / expression [alias], ... }  
FROM table  
[WHERE condition(s)];
```

- WHERE 절은 FROM 절 다음에 옵니다.

ORACLE

2-4

Copyright © Oracle Corporation, 2001. All rights reserved.

선택되는 행 제한

WHERE 절을 사용하여 질의에서 반환되는 행을 제한할 수 있습니다. WHERE 절은 만족해야 할 조건을 포함하며 FROM 절 바로 다음에 옵니다. 조건이 참일 경우, 조건을 만족하는 행이 반환됩니다.

구문 설명:

WHERE 조건을 만족하는 행만 질의하도록 제한합니다.
condition 열 이름, 표현식, 상수 및 비교 연산자로 구성됩니다.

WHERE 절은 열 값, 리터럴 값, 산술식 또는 함수를 비교할 수 있으며 다음 세 가지 요소로 구성됩니다.

- 열 이름
- 비교 조건
- 열 이름, 상수 또는 값 목록

WHERE 절 사용

```
SELECT employee_id, last_name, job_id, department_id  
FROM employees  
WHERE department_id = 90 ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

ORACLE

2-5

Copyright © Oracle Corporation, 2001. All rights reserved.

WHERE 절 사용

예제에서 SELECT 문은 업무 ID가 SA_REP인 모든 사원의 이름, 업무 ID 및 부서 번호를 검색 합니다.

SA_REP라는 업무는 EMPLOYEES 테이블의 업무 ID 열과 일치하도록 대문자로 지정되었습니다. 문자열은 대소문자를 구분합니다.

문자열 및 날짜

- 문자열 및 날짜 값은 작은 따옴표로 묶습니다.
- 문자 값은 대소문자를 구분하며 날짜 값은 날짜형식을 구분 합니다.
- 기본 날짜 형식은 ^{오늘날짜} ^{오늘날짜} ^{오늘날짜} DD-MON-RR입니다.

```
SELECT last_name, job_id, department_id  
FROM   employees  
WHERE  last_name = 'Whalen';
```

ORACLE

문자열 및 날짜

WHERE 절의 문자열 및 날짜는 작은 따옴표('')로 묶지만 숫자 상수는 작은 따옴표로 묶지 않습니다.
모든 문자 검색은 대소문자를 구분합니다. EMPLOYEES 테이블에는 모든 이름이 대소문자를 혼합하여 저장되어 있으므로 다음 예제의 경우 행이 반환되지 않습니다.

```
SELECT last_name, job_id, department_id  
FROM   employees  
WHERE  last_name = 'WHALEN';
```

오라클 데이터베이스는 세기, 연도, 월, 일, 시, 분, 초를 나타내는 내부 숫자 형식으로 날짜를 저장하는데 기본 날짜 표기 형식은 DD-MON-RR입니다.

참고: 기본 날짜 형식 변경은 이후 단원에서 설명합니다.

비교 조건

연산자	의미
=	같음
>	보다 큼
>=	크거나 같음
<	보다 작음
<=	작거나 같음
<>	같지 않음

ORACLE

2-7

Copyright © Oracle Corporation, 2001. All rights reserved.

비교 조건

비교 조건은 표현식을 다른 값이나 표현식과 비교하는 조건문에 사용되며 WHERE 절에서 다음 형식으로 사용됩니다.

구문

... WHERE *expr operator value*

예제

```
... WHERE hire_date='01-JAN-95'  
... WHERE salary>=6000  
... WHERE last_name='Smith'
```

WHERE 절에는 별칭을 사용할 수 없습니다.

참고: != 및 ^= 기호 역시 같지 않음 조건을 나타냅니다.

비교 조건 사용

```
SELECT last_name, salary  
FROM   employees  
WHERE  salary <= 3000;
```

	LAST NAME	SALARY
	Matos	2600
	Vargas	2500

ORACLE

비교 조건 사용

예제에서 SELECT 문은 EMPLOYEES 테이블에서 급여가 3000 이하인 사원의 이름과 급여를 검색합니다. WHERE 절에 명시적(explicit) 값이 제공되며, 명시적 값인 3000과 EMPLOYEES 테이블의 SALARY 열에 있는 급여 값을 비교합니다.

다른 비교 조건

연산자	의미
BETWEEN ...AND...	두 값 사이(지정한 값 포함)
IN (set)	값 목록 중의 값과 일치
LIKE	문자 패턴 일치
IS NULL	널 값

ORACLE

BETWEEN 조건 사용

BETWEEN 조건을 사용하여 값의 범위에 따라 행을 표시합니다.

```
SELECT last_name, salary  
FROM   employees  
WHERE  salary [BETWEEN 2500 AND 3500];
```

↑ ↑
하한 상한

LAST_NAME	SALARY
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

ORACLE

BETWEEN 조건

BETWEEN 범위 조건을 사용하면 값의 범위에 따라 행을 표시할 수 있으며 지정 범위에는 하한값과 상한값이 포함됩니다.

슬라이드의 SELECT 문은 EMPLOYEES 테이블에서 급여가 \$2,500 이상이고 \$3,500 이하인 사원의 행을 반환합니다.

BETWEEN 조건으로 지정한 값도 범위에 포함되며 하한값을 먼저 지정해야 합니다.

IN 조건 사용

IN 멤버 조건을 사용하면 값이 목록에 있는지 확인할 수 있습니다.

```
SELECT employee_id, last_name, salary, manager_id  
FROM   employees  
WHERE  manager_id IN (100, 101, 201);
```

EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
202	Fay	6000	201
200	Whalen	4400	101
205	Higgins	12000	101
101	Kochhar	17000	100
102	De Haan	17000	100
124	Mourgos	5800	100
149	Zlotkey	10500	100
201	Hartstein	13000	100

8 rows selected.

ORACLE

2-11

Copyright © Oracle Corporation, 2001. All rights reserved.

IN 조건

값이 특정 값 집합에 있는지 확인하려면 IN 조건을 사용합니다. IN 조건은 멤버 조건이라고도 합니다.

슬라이드 예제는 관리자의 사원 번호가 100, 101 또는 201인 모든 사원의 사원 번호, 이름, 급여 및 관리자의 사원 번호를 표시합니다.

IN 조건은 모든 데이터 유형에 사용할 수 있습니다. 다음 예제는 EMPLOYEES 테이블에서 WHERE 절의 이름 목록에 포함된 이름을 가진 사원의 행을 반환합니다.

```
SELECT employee_id, manager_id, department_id  
FROM   employees  
WHERE  last_name IN ('Hartstein', 'Vargas');
```

목록에 사용되는 문자 또는 날짜는 작은 따옴표('')로 묶어야 합니다.

LIKE 조건 사용

- LIKE 조건을 사용하면 유효한 검색 문자열 값인 대체 문자를 사용하여 검색할 수 있습니다.
- 검색 조건은 리터럴 문자 또는 숫자를 포함할 수 있습니다.
 - %에는 문자가 오지 않거나 여러 개 올 수 있습니다.
 - _에는 문자가 하나만 올 수 있습니다.

```
SELECT first_name  
FROM employees  
WHERE first_name LIKE 'S%';
```

ORACLE

2-12

Copyright © Oracle Corporation, 2001. All rights reserved.

LIKE 조건

검색할 값을 정확하게 알지 못하는 경우 LIKE 조건을 사용하여 문자 패턴이 일치하는 행을 선택할 수 있습니다. 문자 패턴 일치 연산을 대체 문자검색이라고도 하며 검색 문자열은 두 가지 기호를 사용하여 구성할 수 있습니다.

기호	설명
%	0개 이상의 일련의 문자를 나타냅니다.
_	문자 하나를 나타냅니다.

슬라이드의 SELECT 문은 EMPLOYEES 테이블에서 이름이 S로 시작하는 모든 사원의 이름을 반환합니다. 이 경우 대문자 S를 사용했으므로 소문자 s로 시작하는 이름은 반환되지 않습니다.

LIKE 조건은 일부 BETWEEN 비교 연산에 대한 단축 방법으로 사용될 수 있습니다. 다음 예제는 1995년 1월과 1995년 12월 사이에 입사한 모든 사원의 이름과 입사일을 표시합니다.

```
SELECT last_name, hire_date  
FROM employees  
WHERE hire_date LIKE '%95';
```

LIKE 조건 사용

- 패턴 일치 문자를 결합할 수 있습니다.

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

LAST_NAME
Kochhar
Lorentz
Mourgos

- ESCAPE 식별자를 사용하여 "%" 또는 "_" 자체를 검색할 수 있습니다.

2-13

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

대체 문자 결합

% 및 _ 기호는 리터럴 문자를 결합할 때 사용됩니다. 슬라이드 예제는 이름의 두번째 문자가 o인 모든 사원의 이름을 표시합니다.

ESCAPE 옵션

검색할 문자에 실제로 '%' 및 '_' 문자가 포함된 경우 ESCAPE 옵션을 사용하여 이스케이프 문자를 지정합니다. 'SA_'를 포함하는 문자열을 검색하는 경우 다음 SQL 문을 사용하여 검색 할 수 있습니다.

```
SELECT employee_id, last_name, job_id  
FROM employees  
WHERE job_id LIKE '%SA\_%' ESCAPE '\';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID
149	Zlotkey	SA_MAN
174	Abel	SA_REP
176	Taylor	SA_REP
178	Grant	SA_REP

ESCAPE 옵션은 백슬래시(\)를 이스케이프 문자로 식별합니다. 이 패턴에서 이스케이프 문자가 밀줄(＼) 앞에 있으므로 Oracle Server는 밀줄을 리터럴로 해석합니다.

NULL 조건 사용

IS NULL 연산자를 사용하여 널 여부를 테스트합니다.

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

LAST_NAME	MANAGER_ID
King	

ORACLE

2-14

Copyright © Oracle Corporation, 2001. All rights reserved.

NULL 조건

NULL 조건에는 IS NULL 조건과 IS NOT NULL 조건이 있습니다.

IS NULL 조건은 널 여부를 테스트합니다. 널 값은 알 수 없는 값으로서 사용, 할당 및 적용할 수 없습니다. 널 값은 어떤 값과도 동일성 여부를 판별할 수 없으므로 =를 사용하여 테스트할 수 없습니다. 슬라이드 예제는 관리자가 없는 모든 사원의 이름과 관리자를 검색합니다.

또 다른 예로, 커미션을 받지 않는 업무에 해당하는 모든 사원의 이름, 업무 ID 및 커미션을 표시하려면 다음 SQL 문을 사용합니다.

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

LAST_NAME	JOB_ID	COMMISSION_PCT
King	AD_PRES	
Kochhar	AD_VP	

Higgins	AC_MGR	
Gietz	AC_ACCOUNT	

16 rows selected.

논리 조건

연산자	의미
AND	구성 요소 조건이 모두 TRUE면 TRUE를 반환합니다.
OR	구성 요소 조건 중 하나라도 TRUE면 TRUE를 반환합니다.
NOT	뒤따르는 조건이 FALSE면 TRUE를 반환합니다.

ORACLE

2-15

Copyright © Oracle Corporation, 2001. All rights reserved.

논리 조건

논리 조건은 두 구성 요소 조건의 결과를 결합하여 이를 기반으로 하나의 결과를 생성하거나 단일 조건의 결과를 부정시키기도 합니다. 조건의 전체 결과가 참인 경우에만 행(row)이 반환됩니다. SQL에는 다음 세 가지 논리 연산자를 사용할 수 있습니다.

- AND
- OR
- NOT

이제까지는 모든 예제에서 WHERE 절에 하나의 조건만 지정했지만 이제 AND 및 OR 연산자를 사용하여 하나의 WHERE 절에 여러 조건을 지정할 수 있습니다.

AND 연산자 사용

AND는 조건이 모두 TRUE여야 합니다.

```
SELECT employee_id, last_name, job_id, salary  
FROM   employees  
WHERE  salary >=10000  
AND    job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
149	Zlotkey	SA_MAN	10500
201	Hartstein	MK_MAN	13000

ORACLE

2-16

Copyright © Oracle Corporation, 2001. All rights reserved.

AND 연산자

예제에서 조건이 모두 TRUE인 경우에만 레코드가 선택되므로 업무 ID에 문자열 MAN이 포함되고 급여가 \$10,000 이상인 사원만 선택됩니다.

모든 문자 검색은 대소문자를 구분하므로 MAN을 소문자로 표시하면 행이 반환되지 않습니다.
문자열은 따옴표로 묶어야 합니다.

AND 진리표

다음 표는 AND로 두 표현식을 결합한 결과를 표시합니다.

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

OR 연산자 사용

OR는 조건 중 하나가 TRUE면 됩니다.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR    job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
100	King	AD_PRES	24000
101	Kochhar	AD_VP	17000
102	De Haan	AD_VP	17000
124	Mourgos	ST_MAN	5600
149	Zlotkey	SA_MAN	10500
174	Abel	SA_REP	11000
201	Hartstein	MK_MAN	13000
205	Higgins	AC_MGR	12000

8 rows selected.

ORACLE

2-17

Copyright © Oracle Corporation, 2001. All rights reserved.

OR 연산자

예제에서 조건 중 하나가 TRUE면 레코드가 선택되므로 업무 ID에 MAN이 포함되거나 급여가 \$10,000 이상인 사원이 선택됩니다.

OR 진리표

다음 표는 OR로 두 표현식을 결합한 결과를 표시합니다.

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

NOT 연산자 사용

```
SELECT last_name, job_id  
FROM employees  
WHERE job_id  
    NOT IN ('IT_PROG', 'ST_CLERK', 'SA REP');
```

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Moungos	ST_MAN
Zlotkey	SA_MAN
Whalen	AD_ASST
Hartstein	MK_MAN
Fay	MK_REP
Higgins	AC_MGR
Gietz	AC_ACCOUNT

10 rows selected.

ORACLE

NOT 연산자

슬라이드 예제는 업무 ID가 IT_PROG, ST_CLERK 또는 SA_REP가 아닌 모든 사원의 이름과 업무 ID를 표시합니다.

NOT 진리표

다음 표는 NOT 연산자를 조건에 적용한 결과를 표시합니다.

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

참고: NOT 연산자는 BETWEEN, LIKE, NULL 등 다른 SQL 연산자와 함께 사용할 수 있습니다.

```
... WHERE job_id NOT IN ('AC_ACCOUNT', 'AD_VP')  
... WHERE salary NOT BETWEEN 10000 AND 15000  
... WHERE last_name NOT LIKE '%A%'  
... WHERE commission_pct IS NOT NULL
```

우선순위 규칙

계산 순서	연산자
1	산술 연산자
2	연결 연산자
3	비교 조건
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT 논리 조건
7	AND 논리 조건
8	OR 논리 조건

괄호를 사용하여 우선순위 규칙을 무시하고 우선순위를 변경할 수 있습니다.

ORACLE

2-19

Copyright © Oracle Corporation, 2001. All rights reserved.

우선순위 규칙

우선순위 규칙은 표현식을 평가하고 계산하는 순서를 결정합니다. 표에 기본 우선순위 순서가 나와 있습니다. 먼저 계산할 식의 좌우에 괄호를 사용하여 기본 순서를 무시하고 우선순위를 변경할 수 있습니다.

우선순위 규칙

```
SELECT last_name, job_id, salary  
FROM   employees  
WHERE  job_id = 'SA REP'  
OR    → job_id = 'AD PRES'  
AND   → salary > 15000;
```

LAST NAME	JOB ID	SALARY
King	AD PRES	24000
Abel	SA REP	11000
Taylor	SA REP	8600
Grant	SA REP	7000

ORACLE

AND 연산자의 우선순위 예제

슬라이드 예제에는 두 가지 조건이 있습니다.

- 첫 번째 조건은 업무 ID가 AD_PRES면서 급여가 15,000을 넘어야 합니다.
- 두 번째 조건은 업무 ID가 SA_REP여야 합니다.

따라서 SELECT 문은 다음과 같이 인식합니다.

“사장이면서 급여가 \$15,000를 넘는 사원 또는 영업 사원인 사원의 행(row)을 선택합니다.”

우선순위 규칙

괄호를 사용하여 우선순위를 강제로 지정합니다.

```
SELECT last_name, job_id, salary
  FROM employees
 WHERE (job_id = 'SA_REP'
    OR job_id = 'AD_PRES')
   AND salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000

ORACLE

2-21

Copyright © Oracle Corporation, 2001. All rights reserved.

괄호 사용

예제에는 두 가지 조건이 있습니다.

- 첫번째 조건은 업무 ID가 AD_PRES 또는 SA_REP여야 합니다.
- 두번째 조건은 급여가 \$15,000를 넘어야 합니다.

따라서 SELECT 문은 다음과 같이 인식합니다.

“사장 또는 영업 사원이면서 급여가 \$15,000를 넘는 사원의 행을 선택합니다.”

ORDER BY 절

- ORDER BY 절을 사용하여 행을 정렬합니다.
 - ASC: 오름차순, 기본값
 - DESC: 내림차순
- ORDER BY 절은 SELECT 문의 가장 끝에 둡니다.

```
SELECT last_name, job_id, department_id, hire_date  
FROM employees  
ORDER BY hire_date ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
King	AD_PRES		90 17-JUN-87
Whalen	AD_ASST		10 17-SEP-87
Kochhar	AD_VP		90 21-SEP-89
Hunold	IT_PROG		60 03-JAN-90
Ernst	IT_PROG		60 21-MAY-91

20 rows selected.

ORACLE

2-22

Copyright © Oracle Corporation, 2001. All rights reserved.

ORDER BY 절

질의 결과로 반환되는 행의 순서는 정의되어 있지 않으므로 ORDER BY 절을 사용하여 행을 정렬합니다. ORDER BY 절은 SQL 문의 가장 끝에 두어야 하며 정렬 조건으로 표현식, 별칭 또는 열 위치를 지정할 수 있습니다.

구문

```
SELECT      expr  
FROM        table  
[WHERE      condition(s)]  
[ORDER BY {column, expr} [ASC|DESC]];
```

구문 설명:

ORDER BY	검색된 행의 표시 순서를 지정합니다.
ASC	행을 오름차순(기본 순서)으로 정렬합니다.
DESC	행을 내림차순으로 정렬합니다.

ORDER BY 절을 사용하지 않으면 정렬 순서를 정의할 수 없으며 동일한 질의를 두 번 수행할 경우에도 행이 동일한 순서로 인출되지 않습니다. 그러나 ORDER BY 절을 사용하면 특정 순서로 행을 표시할 수 있습니다.

내림차순으로 정렬

```
SELECT last_name, job_id, department_id, hire_date  
FROM employees  
ORDER BY hire_date DESC ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
Zlotkey	SA_MAN	80	29-JAN-00
Mourgos	ST_MAN	50	16-NOV-99
Grant	SA_REP		24-MAY-99
Lorentz	IT_PROG	60	07-FEB-99
Vargas	ST_CLERK	50	09-JUL-96
Taylor	SA_REP	60	24-MAR-98
Matos	ST_CLERK	50	15-MAR-98
Fay	MK_REP		20
Davies	ST_CLERK	50	29-JAN-97

20 rows selected.

ORACLE

데이터 기본 정렬

기본 정렬 순서는 오름차순입니다.

- 숫자 값은 작은 값부터 표시됩니다(예: 1-999).
- 날짜 값은 이론 값부터 표시됩니다(예: 01-JAN-92가 01-JAN-95보다 먼저 표시됨).
- 문자 값은 영문자순으로 표시됩니다(예: A가 먼저 표시되고 Z가 나중에 표시됨).
- 널 값은 오름차순에서는 마지막에 표시되고 내림차순에서는 처음에 표시됩니다.

기본 순서 바꾸기

행 표시 순서를 바꾸려면 ORDER BY 절에서 열 이름 다음에 DESC 키워드를 지정합니다. 슬라이드 예제는 가장 최근에 입사한 사원 순으로 결과를 정렬합니다.

열 별칭을 기준으로 정렬

```
SELECT employee_id, last_name, salary*12 annsal  
FROM employees  
ORDER BY annsal;
```

EMPLOYEE_ID	LAST_NAME	ANNSAL
144	Vargas	30000
143	Matos	31200
142	Davies	37200
141	Rajs	42000
107	Lorentz	50400
200	Whalen	52800
124	Mourgos	69600
104	Ernst	72000
202	Fay	72000
178	Grant	84000

20 rows selected.

ORACLE

열 별칭을 기준으로 정렬

ORDER BY 절에 열 별칭을 사용할 수 있습니다. 슬라이드 예제는 연봉을 기준으로 데이터를 정렬합니다.

여러 열을 기준으로 정렬

- ORDER BY 목록의 순서가 정렬 순서입니다.

```
SELECT last_name, department_id, salary  
FROM employees  
ORDER BY department_id, salary DESC;
```

LAST_NAME	DEPARTMENT_ID	SALARY
Whalen	10	4400
Hartstein	20	13000
Fay	20	6000
Mourgos	50	5800
Rajs	50	3600
Davies	50	3100
Matos	50	2600
Vargas	50	2500
...		

20 rows selected.

- SELECT 목록에 없는 열을 기준으로 정렬할 수도 있습니다.

ORACLE

2-25

Copyright © Oracle Corporation, 2001. All rights reserved.

여러 열을 기준으로 정렬

여러 열을 기준으로 질의 결과를 정렬할 수 있으며 테이블의 열 수만큼 가능합니다.

ORDER BY 절에 열을 지정하고 쉼표로 열 이름을 구분합니다. 열 순서를 바꾸려면 이름 뒤에 DESC를 지정합니다. SELECT 절에 포함되지 않은 열을 기준으로 정렬할 수도 있습니다.

예제

모든 사원의 이름과 급여를 표시하고 부서 번호를 기준으로 정렬한 후 급여를 기준으로 내림차순으로 결과를 정렬합니다.

```
SELECT last_name, salary  
FROM employees  
ORDER BY department_id, salary DESC;
```

요약

이 단원에서는 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- WHERE 절을 사용한 출력 행 제한
 - 비교 조건 사용
 - BETWEEN, IN, LIKE 및 NULL 조건 사용
 - 논리 AND, OR 및 NOT 연산자 적용
- ORDER BY 절을 사용한 출력 행 정렬

```
SELECT      * | { [DISTINCT] column / expression [alias], ... }  
FROM        table  
[WHERE      condition(s)]  
[ORDER BY   {column, expr, alias} [ASC|DESC]];
```

ORACLE

요약

이 단원에서는 SELECT 문을 사용하여 반환 행을 제한하고 정렬하는 방법과 다양한 연산자 및 조건을 구현하는 방법을 설명했습니다.

연습 2 개요

이 연습에서는 다음 내용을 다릅니다.

- 데이터 선택 및 행 표시 순서 변경
- WHERE 절을 사용한 행 제한
- ORDER BY 절을 사용한 행 정렬

ORACLE

2-27

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 2 개요

이 연습에서는 WHERE 절 및 ORDER BY 절을 사용하는 다양한 연습 문제를 제공합니다.

연습 2

- 급여가 \$12,000를 넘는 사원의 이름과 급여를 표시하는 질의를 작성하여 lab2_1.sql이라는 이름의 텍스트 파일에 저장한 후 질의를 실행하십시오.

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hartstein	13000

- 사원 번호가 176인 사원의 이름과 부서 번호를 표시하는 질의를 작성하십시오.

LAST_NAME	DEPARTMENT_ID
Taylor	80

- 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.

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Lorentz	4200
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500
Whalen	4400
Hartstein	13000

10 rows selected.

연습 2(계속)

4. 1998년 2월 20일과 1998년 5월 1일 사이에 입사한 사원의 이름, 업무 ID 및 시작일을 표시하되, 시작일을 기준으로 오름차순으로 정렬하십시오.

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

5. 부서 20 및 50에 속하는 모든 사원의 이름과 부서 번호를 이름을 기준으로 영문자순으로 표시하십시오.

LAST_NAME	DEPARTMENT_ID
Davies	50
Fay	20
Hartstein	20
Matos	50
Mourgos	50
Rajs	50
Vargas	50

7 rows selected.

6. 급여가 \$5,000와 \$12,000 사이이고 부서 번호가 20 또는 50인 사원의 이름과 급여를 나열하도록 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.

Employee	Monthly_Salary
Mourgos	5800
Fay	6000

연습 2(계속)

7. 1994년에 입사한 모든 사원의 이름과 입사일을 표시하십시오.

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

8. 관리자가 없는 모든 사원의 이름과 업무를 표시하십시오.

LAST_NAME	JOB_ID
King	AD_PRES

9. 커미션을 받는 모든 사원의 이름, 급여 및 커미션을 급여 및 커미션을 기준으로 내림차순으로 정렬하여 표시하십시오.

LAST_NAME	SALARY	COMMISSION_PCT
Abel	11000	.3
Zlotkey	10500	.2
Taylor	8600	.2
Grant	7000	.15

시간이 있을 때 다음 문제를 풀어보십시오.

10. 이름의 세 번째 문자가 *a*인 모든 사원의 이름을 표시하십시오.

LAST_NAME
Grant
Whalen

11. 이름에 *a*와 *e*가 있는 모든 사원의 이름을 표시하십시오.

LAST_NAME
De Haan
Davies
Whalen
Hartstein

연습 2(계속)

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.

LAST_NAME	JOB_ID	SALARY
Davies	ST_CLERK	3100
Matos	ST_CLERK	2600
Abel	SA_REP	11000
Taylor	SA_REP	8600

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.

Employee	Monthly_Salary	Commission_Pct
Zlotkey	10500	.2
Taylor	8600	.2

3

단일 행 할수

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

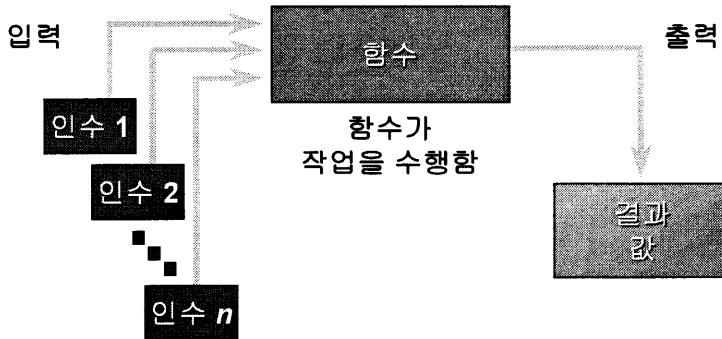
- **SQL**에서 사용 가능한 여러 함수 유형 설명
- **SELECT** 문에서 문자, 숫자 및 날짜 함수 사용
- 변환 함수 사용법 설명

ORACLE

단원 목표

함수를 사용하면 기본 질의 블록을 효율적으로 작성할 수 있을 뿐만 아니라 데이터 값을 조작 할 수도 있습니다. 이 단원은 함수를 설명하는 두 단원 중 첫번째 단원으로, 데이터 유형 변환 함수(예: 문자 데이터를 숫자 데이터로 변환)와 함께 단일 행 문자, 숫자 및 날짜 함수를 중점적으로 설명합니다.

SQL 함수



ORACLE

3-3

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL 함수

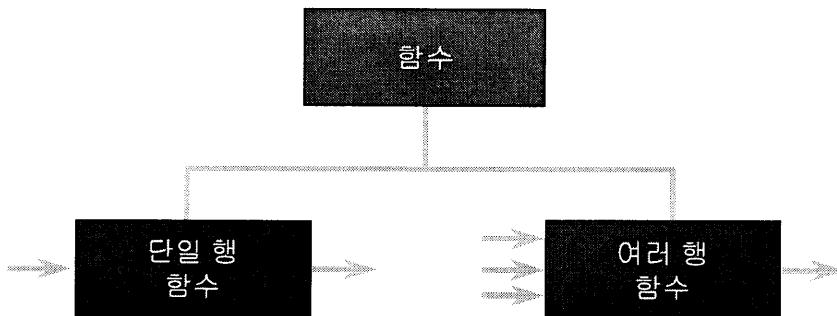
함수는 SQL에서 제공하는 강력한 기능으로, 다음과 같은 작업에 사용할 수 있습니다.

- 데이터의 계산 수행
- 개별 데이터 항목 수정
- 행 그룹에 대한 출력 결과 조작
- 표시할 날짜 및 숫자 형식 지정
- 열 데이터 유형 변환

SQL 함수는 경우에 따라 인수를 받아들이며 항상 값을 반환합니다.

참고: 이 단원에서 설명하는 대부분의 함수는 오라클의 SQL에서 사용되는 함수입니다.

SQL 함수의 두 유형



ORACLE

3-4

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL 함수(계속)

함수에는 두 가지 유형이 있습니다.

- 단일 행 함수
- 여러 행 함수

단일 행 함수

단일 행 함수는 단일 행만 연산하며 행 당 하나의 결과를 반환합니다. 단일 행 함수에도 여러 유형이 있는데 이 단원에서는 다음 유형을 설명합니다.

- 문자
- 숫자
- 날짜
- 변환

여러 행 함수

행 그룹을 조작하여 행 그룹 당 하나의 결과를 제공하도록 하는 함수도 있습니다. 이러한 함수를 그룹 함수라고 하며 다음 단원에서 설명합니다.

사용 가능한 함수의 전체 목록 및 구문에 대한 자세한 내용은 *Oracle9i SQL Reference*를 참조하십시오.

단일 행 함수

단일 행 함수

- 데이터 항목을 조작합니다.
- 인수를 사용하고 값을 하나 반환합니다.
- 반환되는 각 행에 대해 작업합니다.
- 행 당 하나의 결과를 반환합니다.
- 데이터 유형을 수정할 수 있습니다.
- 중첩될 수 있습니다.
- 열 또는 표현식을 인수로 사용합니다.

```
function_name [(arg1, arg2,...)]
```

ORACLE

3-5

Copyright © Oracle Corporation, 2001. All rights reserved.

단일 행 함수

단일 행 함수를 사용하여 데이터 항목을 조작할 수 있습니다. 단일 행 함수에는 인수를 하나 이상 사용할 수 있으며 질의에서 반환하는 행마다 하나의 값을 반환합니다. 인수로는 다음과 같은 것을 사용할 수 있습니다.

- 사용자 지원 상수
- 변수 값
- 열 이름
- 표현식

단일 행 함수의 기능은 다음과 같습니다.

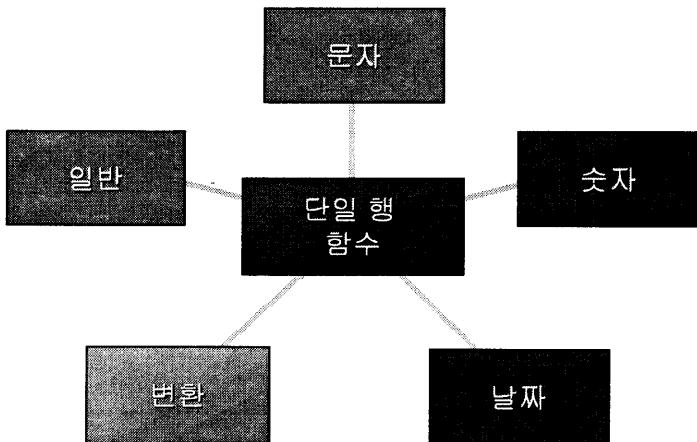
- 질의에서 반환되는 각 행에 대해 작업합니다.
- 행 당 하나의 결과를 반환합니다.
- 참조한 데이터 값과 다른 유형의 데이터 값을 반환할 수 있습니다.
- 인수를 하나 이상 사용할 수 있습니다.
- SELECT, WHERE 및 ORDER BY 절에 사용할 수 있으며 중첩될 수 있습니다.

구문 설명:

function_name 함수 이름입니다.

arg1, arg2 함수에서 사용하는 인수입니다. 열 이름 또는 표현식을 사용할 수 있습니다.

단일 행 함수



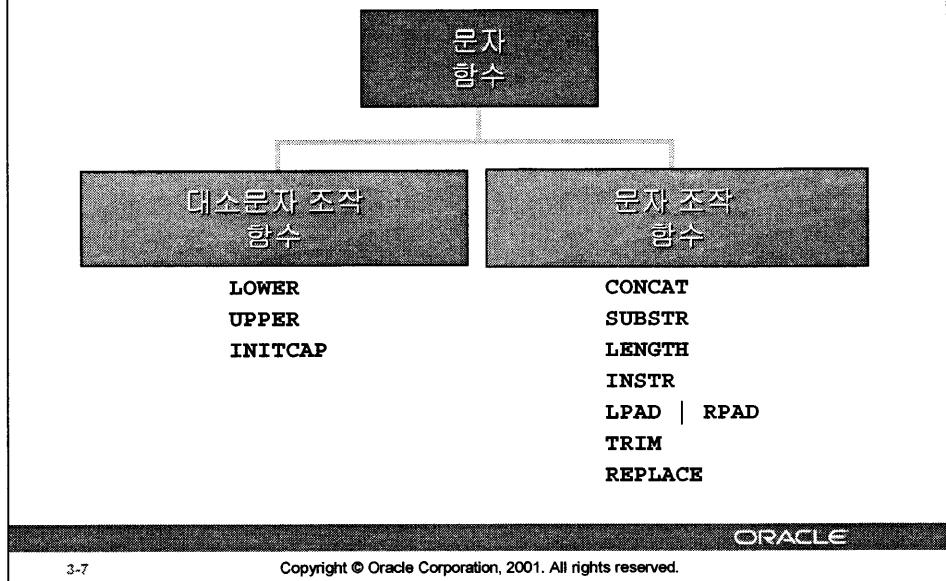
ORACLE

단일 행 함수(계속)

이 단원에서는 다음과 같은 단일 행 함수를 설명합니다.

- 문자 함수: 문자를 입력 값으로 받으며 문자 또는 숫자 값을 반환합니다.
- 숫자 함수: 숫자를 입력하면 숫자 값을 반환합니다.
- 날짜 함수: DATE 데이터 유형의 값에 동작합니다. 모든 날짜 함수는 DATE 데이터 유형 값을 반환하며 MONTHS_BETWEEN 함수만 숫자를 반환합니다.
- 변환 함수: 값의 데이터 유형을 변환합니다.
- 일반 함수
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
 - CASE
 - DECODE

문자 함수



3-7

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

문자 함수

단일 행 문자 함수는 문자 데이터를 입력 값으로 받으며 문자 또는 숫자 값을 모두 반환할 수 있습니다. 단일 행 문자 함수는 다음과 같이 나뉩니다.

- 대소문자 조작 함수
- 문자 조작 함수

함수	용도
LOWER (<i>column/expression</i>)	알파벳 값을 소문자로 변환합니다.
UPPER (<i>column/expression</i>)	알파벳 값을 대문자로 변환합니다.
INITCAP (<i>column/expression</i>)	알파벳 값을 각 단어의 첫 문자는 대문자로, 나머지 문자는 모두 소문자로 변환합니다.
CONCAT (<i>column1/expression1, column2/expression2</i>)	첫번째 문자 값을 두번째 문자 값에 연결합니다. 연결 연산자()와 동일합니다.
SUBSTR (<i>column/expression, m [,n]</i>)	문자 값의 위치 <i>m</i> 에서 <i>n</i> 까지 지정된 문자를 반환합니다. (<i>m</i> 이 음수면 문자 값의 끝부터 세며 <i>n</i> 을 생략하면 문자열의 끝까지 모든 문자가 반환됩니다.)

참고: 이 단원에서 설명하는 함수는 전체 사용 가능한 함수 중 일부입니다.

문자 함수

문자
함수

대소문자 조작
함수

LOWER
UPPER
INITCAP

문자 조작
함수

CONCAT
SUBSTR
LENGTH
INSTR
LPAD | RPAD
TRIM
REPLACE

ORACLE

문자 함수(계속)

함수	용도
LENGTH(<i>column/expression</i>)	표현식의 문자 수를 반환합니다.
INSTR(<i>column/expression</i> , ' <i>string</i> ', [<i>m</i>], [<i>n</i>])	지정된 문자열의 위치를 숫자로 반환합니다. 검색 시작 위치 <i>m</i> 과 문자열의 발생 횟수 <i>n</i> 을 지정할 수 있습니다 (선택 사항). <i>m</i> 과 <i>n</i> 의 기본값은 1이며, 기본값으로 지정된 경우 처음부터 검색을 시작하고 첫번째 발생 위치를 보고합니다.
LPAD(<i>column/expression</i> , <i>n</i> , ' <i>string</i> ') RPAD(<i>column/expression</i> , <i>n</i> , ' <i>string</i> ')	전체 폭(width)이 <i>n</i> 이 되도록 문자 값을 오른쪽 정렬하고 빈 곳을 지정한 <i>string</i> 으로 채웁니다. 전체 폭이 <i>n</i> 이 되도록 문자 값을 왼쪽 정렬하고 빈 곳을 지정한 <i>string</i> 으로 채웁니다.
TRIM(<i>leading/trailing/both</i> , ' <i>trim_character</i> ' FROM ' <i>trim_source</i> ')	문자열에서 접두어나 접미어 또는 두 가지 모두를 자릅니다. <i>trim_character</i> 또는 <i>trim_source</i> 가 문자 리터럴이면 작은 따옴표로 둑어야 합니다. Oracle8i 이상에서 사용 가능한 기능입니다.
REPLACE(<i>text</i> , ' <i>search_string</i> ', ' <i>replacement_string</i> ')	텍스트 표현식에서 문자열을 검색하여 해당 문자열을 발견한 경우 지정된 대체 문자열로 바꿉니다.

대소문자 조작 함수

문자열의 대소문자를 변환합니다.

함수	결과
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

ORACLE

3-3

Copyright © Oracle Corporation, 2001. All rights reserved.

대소문자 조작 함수

LOWER, UPPER, INITCAP 등 세 개 함수는 대소문자 변환 함수입니다.

- LOWER: 대소문자 또는 대문자 문자열을 소문자로 변환합니다.
- UPPER: 대소문자 또는 소문자 문자열을 대문자로 변환합니다.
- INITCAP: 각 단어의 첫 문자는 대문자로, 나머지 문자는 소문자로 변환합니다.

```
SELECT 'The job id for '||UPPER(last_name)||' is '
      ||LOWER(job_id) AS "EMPLOYEE DETAILS"
  FROM employees;
```

EMPLOYEE DETAILS
The job id for KING is ad_pres
The job id for KOCHHAR is ad_vp
The job id for DE HAAN is ad_vp
■ ■ ■
The job id for HIGGINS is ac_mgr
The job id for GIETZ is ac_account

20 rows selected.

대소문자 조작 함수 사용

사원 Higgins의 사원 번호, 이름 및 부서 번호를 표시합니다.

```
SELECT employee_id, last_name, department_id  
FROM   employees  
WHERE  last_name = 'higgins';  
no rows selected
```

```
SELECT employee_id, last_name, department_id  
FROM   employees  
WHERE  LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
205	Higgins	110

ORACLE

대소문자 조작 함수(계속)

슬라이드 예제는 사원 Higgins의 사원 번호, 이름 및 부서 번호를 표시합니다.

첫번째 SQL 문의 WHERE 절은 사원의 이름을 higgins로 지정합니다. EMPLOYEES 테이블의 모든 데이터는 대소문자가 구분된 상태로 저장되어 있으므로 질의를 실행해도 이 테이블에서 higgins와 일치하는 이름을 찾지 못하게 되고 따라서 행이 선택되지 않습니다.

두번째 SQL 문의 WHERE 절은 EMPLOYEES 테이블의 사원 이름과 higgins를 비교하되, 비교하기 전에 LAST_NAME 열을 소문자로 변환합니다. 이제 두 이름이 모두 소문자이므로 일치하는 이름을 찾을 수 있게 되며 한 행이 선택됩니다. WHERE 절을 다음과 같이 변경해도 동일한 결과를 얻을 수 있습니다.

```
...WHERE last_name = 'Higgins'
```

출력되는 이름은 데이터베이스에 저장된 이름과 동일하게 표시됩니다. 이름을 대문자로 표시하려면 SELECT 문에서 UPPER 함수를 사용합니다.

```
SELECT employee_id, UPPER(last_name), department_id  
FROM   employees  
WHERE  INITCAP(last_name) = 'Higgins';
```

문자 조작 함수

문자열을 조작합니다.

함수	결과
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld', 1, 5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary, 10, '***)	*****24000
RPAD(salary, 10, '***')	24000*****
TRIM('H' FROM 'HelloWorld')	elloWorld

ORACLE

3-11

Copyright © Oracle Corporation, 2001. All rights reserved.

문자 조작 함수

이 단원에서는 CONCAT, SUBSTR, LENGTH, INSTR, LPAD, RPAD, TRIM 등의 문자 조작 함수를 설명합니다.

- CONCAT: 값을 결합합니다. CONCAT에는 파라미터를 두 개만 사용할 수 있습니다.
- SUBSTR: 지정한 길이의 문자열을 추출합니다.
- LENGTH: 문자열의 길이를 숫자 값으로 표시합니다.
- INSTR: 지정한 문자의 위치를 숫자로 표시합니다.
- LPAD: 문자 값을 오른쪽 정렬하고 빈 곳을 지정한 문자열로 채웁니다.
- RPAD: 문자 값을 왼쪽 정렬하고 빈 곳을 지정한 문자열로 채웁니다.
- TRIM: 문자열에서 접두어나 접미어 또는 두 가지 모두를 자릅니다. *Trim_character* 또는 *trim_source*가 문자 리터럴이면 작은 따옴표로 묶어야 합니다.

문자 조작 함수 사용

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
       job_id, LENGTH(last_name),
       INSTR(last_name, 'a') "Contains 'a'?"
  FROM employees
 WHERE SUBSTR(job_id, 4) = 'REP';
```

EMPLOYEE_ID	NAME	JOB_ID	LENGTH(LAST_NAME)	Contains 'a'?
174	EllenAbel	SA_REP	4	0
176	JonathonTaylor	SA_REP	6	2
178	KimberelyGrant	SA_REP	5	3
202	PatFay	MK_REP	3	2

3-12

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

문자 조작 함수(계속)

슬라이드 예제는 업무 ID의 네번째 문자부터 REP라는 문자열이 포함된 모든 사원에 대해 성과 이름을 합친 전체 이름, 성의 길이, 성에서 문자 a의 위치를 표시합니다.

예제

성이 n으로 끝나는 사원에 대한 데이터를 표시하도록 슬라이드의 SQL 문을 수정합니다.

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
       LENGTH(last_name), INSTR(last_name, 'a') "Contains 'a'?"
  FROM employees
 WHERE SUBSTR(last_name, -1, 1) = 'n';
```

EMPLOYEE_ID	NAME	LENGTH(LAST_NAME)	Contains 'a'?
102	LexDe Haan	7	5
200	JenniferWhalen	6	3
201	MichaelHartstein	9	2

숫자 함수

- **ROUND:** 지정한 소수점 자리로 값을 반올림합니다.

ROUND(45.926, 2) → 45.93

- **TRUNC:** 지정한 소수점 자리까지 남기고 값을 버립니다.

TRUNC(45.926, 2) → 45.92

- **MOD:** 나눗셈의 나머지를 반환합니다.

MOD(1600, 300) → 100

ORACLE

3-13

Copyright © Oracle Corporation, 2001. All rights reserved.

숫자 함수

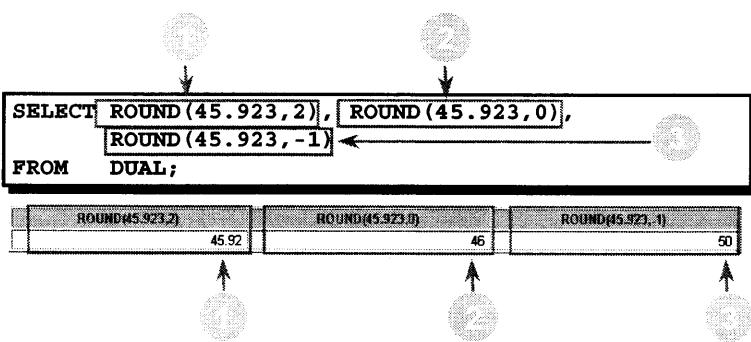
숫자 함수는 숫자 입력을 사용하고 숫자 값을 반환합니다. 이 단원에서는 몇몇 숫자 함수를 설명합니다.

함수	용도
ROUND(column expression, n)	열, 표현식 또는 값을 소수점 n째 자리로 반올림합니다. n을 지정하지 않은 경우 소수점 이하 값이 없어집니다. n이 음수이면 소수점 왼쪽의 수가 반올림됩니다.
TRUNC(column expression, n)	열, 표현식 또는 값을 소수점 n째 자리까지 남기고 버립니다. n을 지정하지 않은 경우에는 기본값 0이 지정됩니다.
MOD(m, n)	m을 n으로 나눈 나머지를 반환합니다.

참고: 이 목록은 전체 사용 가능한 숫자 함수 중 일부입니다.

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

ROUND 함수 사용



DUAL은 함수 및 계산 결과를 보는 데 사용할 수 있는 더미 테이블(dummy table)입니다.

ORACLE

ROUND 함수

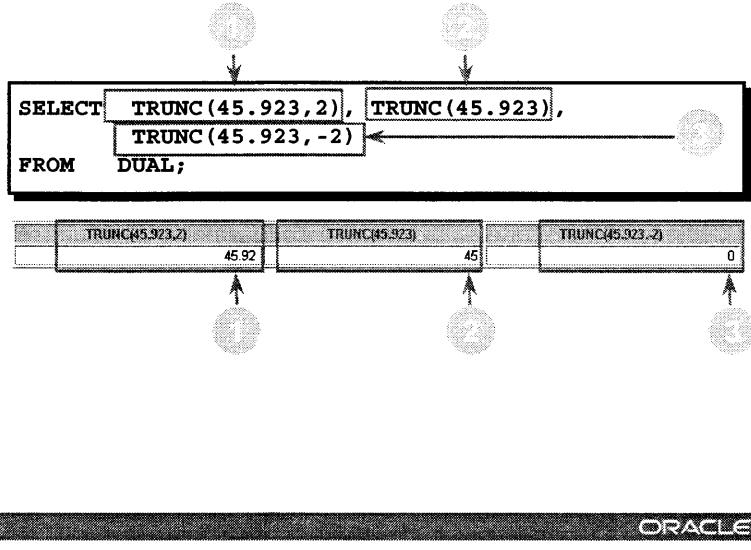
ROUND 함수는 열, 표현식 또는 값을 소수점 n 째 자리로 반올림합니다. 두번쨰 인수가 0이거나 없으면 값을 일의 자리로 반올림하고 2면 값을 소수점 둘째 자리로 반올림합니다. 반대로 두번쨰 인수가 -2면 값을 백의 자리로 반올림합니다.

ROUND 함수를 날짜 함수와 함께 사용할 수도 있습니다. 이에 대한 예제는 이 단원의 뒷부분에 나와 있습니다.

DUAL 테이블

DUAL 테이블은 사용자 SYS가 소유하며 모든 사용자가 액세스할 수 있습니다. 이 테이블은 DUMMY라는 열과 값 X가 들어 있는 행(row)을 하나씩 포함합니다. DUAL 테이블은 사용자 데이터가 들어 있는 테이블에서 파생되지 않는 표현식, 의사 열, 상수 값 등을 한 번만 반환할 때 유용합니다. 질의에서 SELECT 절과 FROM 절은 필수 항목이므로 DUAL 테이블을 사용하여 SELECT 절 구문을 완성하면 계산만 수행되는 몇몇 경우에 실제 테이블에서 관련 데이터를 선택할 필요가 없습니다.

TRUNC 함수 사용



3-15

Copyright © Oracle Corporation, 2001. All rights reserved.

TRUNC 함수

TRUNC 함수는 열, 표현식 또는 값을 소수점 n 째 자리까지 남기고 버립니다.

TRUNC 함수는 인수를 ROUND 함수와 유사한 방식으로 사용합니다. 두번쩨 인수가 0이거나 없으면 값을 일의 자리까지 남기고 버리며 2면 값을 소수점 둘째 자리까지 남기고 버립니다. 반대로 두번쩨 인수가 -2면 값을 백의 자리까지 남기고 버립니다.

ROUND 함수와 마찬가지로 TRUNC 함수도 날짜 함수와 함께 사용할 수 있습니다.

MOD 함수 사용

업무가 영업 사원인 모든 사원에 대해 급여를 5000으로
나눈 나머지를 계산합니다.

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

LAST_NAME	SALARY	MOD(SALARY,5000)
Abel	11000	1000
Taylor	8600	3600
Grant	7000	2000

ORACLE

MOD 함수

MOD 함수는 첫번째 값을 두번째 값으로 나눈 나머지를 반환합니다. 슬라이드 예제는 업무 ID가 SA_REP인 모든 사원에 대해 급여를 5000으로 나눈 나머지를 계산합니다.

참고: MOD 함수는 값이 홀수인지 짝수인지를 확인하는 데 많이 사용됩니다.

날짜 사용

- 오라클 데이터베이스는 다음과 같은 내부 숫자 형식으로 날짜를 저장합니다. 세기, 연도, 월, 일, 시, 분, 초
- 기본 날짜 표시 형식은 **DD-MON-RR**입니다.
 - 연도의 마지막 두 자리만 지정하여 20세기에 21세기 날짜를 저장할 수 있습니다.
 - 동일한 방식으로 21세기에 20세기 날짜를 저장할 수 있습니다.

```
SELECT last_name, hire_date  
FROM employees  
WHERE last_name like 'G%';
```

LAST_NAME	HIRE_DATE
Gietz	07-JUN-94
Grant	24-MAY-99

ORACLE

3-17

Copyright © Oracle Corporation, 2001. All rights reserved.

오라클 날짜 형식

오라클 데이터베이스는 세기, 연도, 월, 일, 시, 분, 초 형태의 내부 숫자 형식으로 날짜를 저장합니다.

날짜의 기본 표시 및 입력 형식은 DD-MON-RR이며 유효한 오라클 날짜 범위는 B.C. 4712년 1월 1일부터 A.D. 9999년 12월 31일입니다.

슬라이드 예제에서 사원 Gietz의 HIRE_DATE는 기본 형식인 DD-MON-RR로 표시됩니다. 그러나 데이터베이스에는 이 형식으로 저장되어 있지 않고 날짜 및 시간의 모든 요소가 저장되어 있습니다. 따라서 07-JUN-94와 같은 HIRE_DATE가 일, 월 및 연도로 표시되지만, 이와 관련된 시간 및 세기 정보도 있습니다. 완전한 데이터는 1994년 6월 7일 오후 5시 10분 43초입니다.

이 데이터는 내부적으로 다음과 같이 저장됩니다.

CENTURY	YEAR	MONTH	DAY	HOUR	MINUTE	SECOND
19	94	06	07	5	10	43

세기와 2000년

Oracle server는 2000년 문제를 해결했습니다. 날짜 열이 포함된 레코드가 테이블에 삽입되는 경우, 세기 정보가 SYSDATE 함수에서 추출됩니다. 하지만 날짜 열이 화면에 표시될 때 기본적으로 세기 구성 요소는 표시되지 않습니다.

DATE 데이터 유형은 내부적으로 연도 정보를 저장할 때 항상 네 자리 숫자로 저장하며, 이중 두 자리는 세기, 나머지 두 자리는 연도 정보입니다. 예를 들어, 오라클 데이터베이스는 연도를 96 또는 01이 아닌 1996 또는 2001로 저장합니다.

날짜 사용

SYSDATE는 다음을 반환하는 함수입니다.

- 날짜
- 시간

ORACLE

3-18

Copyright © Oracle Corporation, 2001. All rights reserved.

SYSDATE 함수

SYSDATE는 데이터베이스 서버의 현재 날짜 및 시간을 반환하는 날짜 함수입니다. 열 이름 사용 방식과 동일한 방식으로 **SYSDATE**를 사용합니다. 예를 들어, 테이블에서 **SYSDATE**를 선택하여 현재 날짜를 표시할 수 있습니다. 일반적으로 **SYSDATE**를 **DUAL**이라는 더미 테이블(dummy table)에서 선택합니다.

예제

DUAL 테이블을 사용하여 현재 날짜를 표시합니다.

```
SELECT SYSDATE  
FROM   DUAL;
```

SYSDATE
28-SEP-01

날짜 계산

- 날짜에 숫자를 더하거나 빼서 날짜 값을 계산합니다.
- 한 날짜에서 다른 날짜를 빼서 날짜 간의 일 수를 알 수 있습니다.
- 시간을 **24**로 나누어 날짜에 시간을 더합니다.

ORACLE

3-19

Copyright © Oracle Corporation, 2001. All rights reserved.

날짜 계산

데이터베이스는 날짜를 숫자로 저장하기 때문에 덧셈과 뺄셈과 같은 산술 연산자를 사용하여 계산할 수 있습니다. 날짜뿐만 아니라 숫자 상수와도 더하거나 뺄 수 있습니다.

다음 연산 작업을 수행 할 수 있습니다.

연산	결과	설명
date + number	날짜	날짜에 일 수를 더합니다.
date - number	날짜	날짜에서 일 수를 뺍니다.
date - date	일 수	한 날짜에서 다른 날짜를 뺍니다.
date + number/24	날짜	날짜에 시간 수를 더합니다.

날짜에 산술 연산자 사용

(X) 53)

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS
FROM employees
WHERE department_id = 90;
```

LAST_NAME	WEEKS
King	744.245395
Kochhar	626.102538
De Haan	453.245395

ORACLE

날짜 계산(계속)

슬라이드 예제는 부서 90에 있는 모든 사원의 이름과 근무한 주 수를 표시합니다. 현재 날짜(SYSDATE)에서 입사일을 뺀 후 결과를 7로 나누어 사원이 근무한 주 수를 계산합니다.

참고: SYSDATE는 현재 날짜 및 시간을 반환하는 SQL 함수로서 계산 결과가 예제와 다를 수 있습니다.

과거 날짜에서 현재 날짜를 빼는 경우 차이는 음수 값이 됩니다.

날짜 함수

함수	설명
MONTHS_BETWEEN	두 날짜 간의 달 수
ADD_MONTHS	날짜에 달 수 더하기
NEXT_DAY	지정한 날짜의 다음 날
LAST_DAY	해당 달의 마지막 날
ROUND	날짜 반올림
TRUNC	날짜 버림

ORACLE

3-21

Copyright © Oracle Corporation, 2001. All rights reserved.

날짜 함수

날짜 함수는 오라클 날짜를 연산합니다. 모든 날짜 함수는 DATE 데이터 유형 값을 반환하며 MONTHS_BETWEEN만 숫자 값을 반환합니다.

- MONTHS_BETWEEN(date1, date2): date1과 date2 간의 달 수를 계산합니다. 결과는 양수 또는 음수입니다. date1이 date2보다 나중이면 결과는 양수고 date1이 date2보다 먼저면 결과는 음수입니다. 결과 중 정수가 아닌 부분은 달의 일부를 나타냅니다.
- ADD_MONTHS(date, n): date에 n 달을 더합니다. n 값은 정수여야 하며 음수일 수 있습니다.
- NEXT_DAY(date, 'char'): date보다 이후 날짜며 지정한 요일 ('char')에 해당하는 날짜를 찾습니다. char의 값은 요일을 나타내는 숫자 또는 문자열입니다.
- LAST_DAY(date): date를 포함하는 달의 마지막 날짜를 찾습니다.
- ROUND(date[, 'fmt']): 형식 모델 fmt에 의해 지정된 단위로 반올림한 date를 반환합니다. 형식 모델 fmt를 생략하면 date는 가장 가까운 날로 반올림됩니다.
- TRUNC(date[, 'fmt']): 날짜의 시간 부분을 형식 모델 fmt로 지정한 단위까지 남기고 버린 후 반환합니다. 형식 모델 fmt를 생략하면 date는 가장 가까운 날에 맞추어 버려집니다.

이 목록은 전체 사용 가능한 날짜 함수 중 일부이며 형식 모델은 이 단원의 뒷부분에서 설명합니다. 형식 모델에는 달, 연도 등이 있습니다.

날짜 함수 사용

- `MONTHS_BETWEEN ('01-SEP-95', '11-JAN-94')`
→ 19.6774194
- `ADD_MONTHS ('11-JAN-94', 6)` → '11-JUL-94'
- `NEXT_DAY ('01-SEP-95', 'FRIDAY')`
→ '08-SEP-95'
- `LAST_DAY('01-FEB-95')` → '28-FEB-95'

ORACLE

3-22

Copyright © Oracle Corporation, 2001. All rights reserved.

날짜 함수(계속)

예를 들어, 36개월보다 적게 근무한 모든 사원의 사원 번호, 입사일, 근무한 달 수, 6개월 검토일, 입사 후 첫 금요일 및 입사한 달의 마지막 날을 표시합니다.

```
SELECT employee_id, hire_date,
       MONTHS_BETWEEN (SYSDATE, hire_date) TENURE,
       ADD_MONTHS (hire_date, 6) REVIEW,
       NEXT_DAY (hire_date, 'FRIDAY'), LAST_DAY(hire_date)
  FROM employees
 WHERE MONTHS_BETWEEN (SYSDATE, hire_date) < 36;
```

EMPLOYEE_ID	HIRE_DATE	TENURE	REVIEW	NEXT_DAY	LAST_DAY
107	07-FEB-99	31.6982407	07-AUG-99	12-FEB-99	28-FEB-99
124	16-NOV-99	22.4079182	16-MAY-00	19-NOV-99	30-NOV-99
149	29-JAN-00	19.9885633	29-JUL-00	04-FEB-00	31-JAN-00
178	24-MAY-99	28.1499536	24-NOV-99	28-MAY-99	31-MAY-99

날짜 함수 사용

Assume SYSDATE = '25-JUL-95':

- ROUND(SYSDATE, 'MONTH') → 01-AUG-95
- ROUND(SYSDATE, 'YEAR') → 01-JAN-96
- TRUNC(SYSDATE, 'MONTH') → 01-JUL-95
- TRUNC(SYSDATE, 'YEAR') → 01-JAN-95

ORACLE

3-23

Copyright © Oracle Corporation, 2001. All rights reserved.

날짜 함수(계속)

ROUND 및 TRUNC 함수는 숫자 값과 날짜 값에 사용할 수 있습니다. 날짜에 사용하면 지정한 형식 모델에 따라 함수가 반올림되거나 버려지므로 날짜를 가장 가까운 연도 또는 달로 반올림할 수 있습니다.

예제

1997년에 입사한 모든 사원의 입사일을 비교합니다. 사원 번호, 입사일 그리고 ROUND 및 TRUNC 함수를 사용하여 입사한 달을 표시합니다.

```
SELECT employee_id, hire_date,
       ROUND(hire_date, 'MONTH'), TRUNC(hire_date, 'MONTH')
  FROM employees
 WHERE hire_date LIKE '%97';
```

EMPLOYEE_ID	HIRE_DATE	ROUND(HIR)	TRUNC(HIR)
142	29-JAN-97	01-FEB-97	01-JAN-97
202	17-AUG-97	01-SEP-97	01-AUG-97

연습 3, 1부: 개요

이 연습에서는 다음 내용을 다룹니다.

- 현재 날짜를 표시하는 질의 작성
- 숫자, 문자 및 날짜 함수를 사용하는 질의 작성
- 사원의 근무 연수 및 달 수 계산

ORACLE

3-24

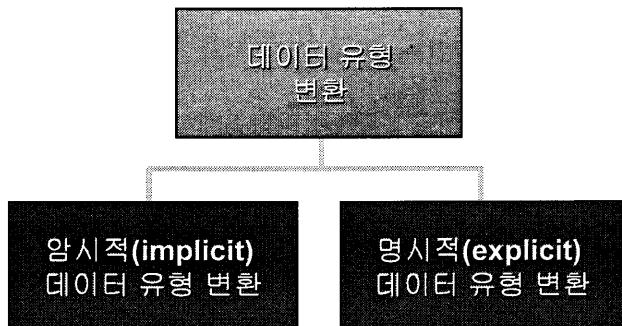
Copyright © Oracle Corporation, 2001. All rights reserved.

연습 3, 1부: 개요

이 연습은 문자, 숫자 및 날짜 데이터 유형에 사용 가능한 여러 함수를 사용하는 다양한 연습 문제를 제공합니다.

이 단원의 마지막에 있는 문제 1-5를 풀어보십시오.

변환 함수



3-25

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

변환 함수

Oracle9i 데이터베이스의 테이블 열은 오라클 데이터 유형 외에 ANSI, DB2 및 SQL/DS 데이터 유형을 사용하여 정의할 수 있습니다. 그러나 Oracle server는 이러한 데이터 유형도 내부적으로 오라클 데이터 유형으로 변환합니다.

Oracle server에서 기대한 것과는 다른 유형의 데이터를 사용하는 경우가 있습니다. 이 경우 Oracle server는 해당 데이터 유형을 원하는 유형으로 자동 변환할 수 있습니다. 이러한 데이터 유형 변환은 Oracle server에 의해 암시적으로 수행되거나 사용자에 의해 명시적으로 수행될 수 있습니다.

암시적(implicit) 데이터 유형 변환은 다음 두 슬라이드에서 설명하는 규칙에 따라 수행됩니다.

명시적(explicit) 데이터 유형 변환은 변환 함수를 사용하여 수행됩니다. 변환 함수는 한 데이터 유형을 다른 데이터 유형으로 변환하며 함수 이름의 형식은 일반적으로 데이터 유형 TO 데이터 유형 규칙을 따릅니다. 전자는 입력 데이터 유형이며 후자는 출력 데이터 유형입니다.

참고: 암시적 데이터 유형 변환을 사용할 수 있지만 SQL 문의 신뢰성을 높이기 위해서는 명시적 데이터 유형 변환을 사용하는 것이 좋습니다.

암시적(**implicit**) 데이터 유형 변환

할당문의 경우 Oracle server는 다음을 자동으로 변환합니다.

원본	대상
VARCHAR2 또는 CHAR	NUMBER
VARCHAR2 또는 CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

ORACLE

암시적(**implicit**) 데이터 유형 변환

Oracle server가 할당문에 사용되는 값의 데이터 유형을 할당 대상의 데이터 유형으로 변환할 수 있을 경우에는 성공적으로 할당됩니다.

암시적(**implicit**) 데이터 유형 변환

표현식 계산을 위해 **Oracle Server**는 다음을 자동으로 변환합니다.

원본	대상
VARCHAR2 또는 CHAR	NUMBER
VARCHAR2 또는 CHAR	DATE

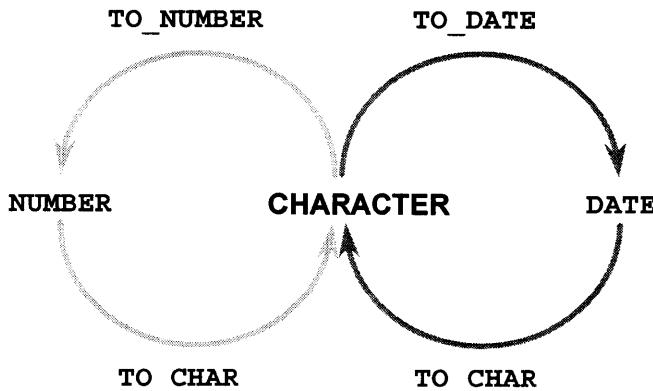
ORACLE

암시적 데이터 유형 변환(계속)

할당문 변환 규칙이 적용되지 않을 때 데이터 유형을 변환해야 하는 경우 Oracle server는 일반적으로 표현식 변환규칙을 적용합니다.

참고: CHAR에서 NUMBER로의 변환은 문자열이 유효한 숫자를 나타내는 경우에만 성공합니다.

명시적(explicit) 데이터 유형 변환



ORACLE

3-28

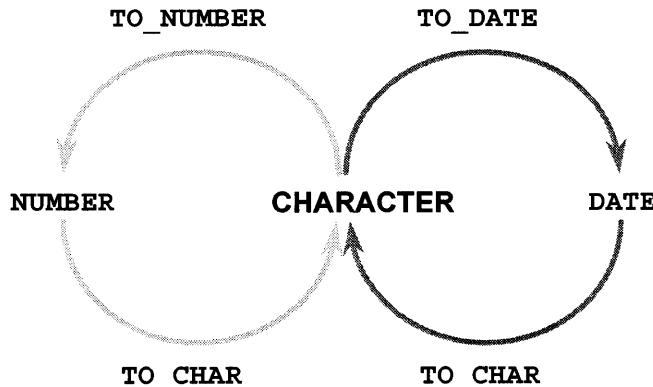
Copyright © Oracle Corporation, 2001. All rights reserved.

명시적(explicit) 데이터 유형 변환

SQL은 값의 데이터 유형을 변환하는 세 가지 함수를 제공합니다.

함수	용도
TO_CHAR (number date, [fmt], [nlsparams])	<p>형식 모델 <i>fmt</i>를 사용하여 숫자 또는 날짜 값을 VARCHAR2 문자열로 변환합니다.</p> <p>숫자 변환: <i>nlsparams</i> 파라미터는 숫자 형식 요소에 의해 반환되는 다음 문자를 지정합니다.</p> <ul style="list-style-type: none">• 싱글 문자• 그룹 구분자• 지역 통화 기호• 국제 통화 기호 <p><i>nlsparams</i> 또는 다른 파라미터가 생략된 경우 이 함수는 해당 세션의 기본 파라미터를 사용합니다.</p>

명시적(explicit) 데이터 유형 변환



ORACLE

3-29

Copyright © Oracle Corporation, 2001. All rights reserved.

명시적 데이터 유형 변환(계속)

함수	용도
<code>TO_CHAR(number date, [fmt], [nlsparams])</code>	날짜 변환: nlsparams 파라미터는 반환되는 월 및 일 이름과 약어에 사용되는 언어를 지정합니다. 이 파라미터가 생략된 경우 이 함수는 해당 세션의 기본 날짜 언어를 사용합니다.
<code>TO_NUMBER(char, [fmt], [nlsparams])</code>	숫자를 포함하는 문자열을 형식 모델 fmt(선택 사항)로 지정한 형식의 숫자로 변환합니다. nlsparams 파라미터는 숫자 변환의 경우 TO_CHAR 함수와 용도가 같습니다.
<code>TO_DATE(char, [fmt], [nlsparams])</code>	날짜를 나타내는 문자열을 지정된 fmt에 따라 날짜 값으로 변환합니다. fmt가 생략된 경우 형식은 DD-MON-YY입니다. nlsparams 파라미터는 날짜 변환의 경우 TO_CHAR 함수와 용도가 같습니다.

명시적 데이터 유형 변환(계속)

참고: 이 단원에 언급된 함수 목록은 전체 사용 가능한 변환 함수 중 일부입니다.

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

날짜에 TO_CHAR 함수 사용

```
TO_CHAR(date, 'format_model')
```

형식 모델:

- 작은 따옴표로 묶어야 하며 대소문자를 구분합니다.
- 모든 유효한 날짜 형식 요소를 포함할 수 있습니다.
- 채워진 공백을 제거하거나 실행 제로를 제거하는 fm 요소가 있습니다.
- 쉼표로 날짜 값과 구분합니다.

ORACLE

3-31

Copyright © Oracle Corporation, 2001. All rights reserved.

특정 형식으로 날짜 표시

이전에는 모든 오라클 날짜 값이 DD-MON-YY 형식으로 표시되었는데 TO_CHAR 함수를 사용하면 기본 형식의 날짜를 사용자가 지정하는 형식으로 변환할 수 있습니다.

지침

- 형식 모델은 작은 따옴표로 묶어야 하며 대소문자를 구분합니다.
- 형식 모델은 모든 유효한 날짜 형식 요소를 포함할 수 있으며 쉼표를 사용하여 형식 모델과 날짜 값을 구분합니다.
- 출력 결과에서 날짜 및 달 이름에는 공백이 자동으로 채워집니다.
- 채워진 공백을 제거하거나 실행 제로를 제거하려면 채우기 모드 fm 요소를 사용하십시오.
- 결과 문자 필드를 다음 단원에서 설명하는 iSQL*Plus COLUMN 명령을 사용하여 형식 지정할 수 있습니다.

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY') Month_Hired  
FROM   employees  
WHERE  last_name = 'Higgins';
```

EMPLOYEE_ID	MONTH
	06/94

날짜 형식 모델 요소

YYYY	네 자리 연도(숫자)
YEAR	연도(문자)
MM	두 자리 값으로 나타낸 달
MONTH	달 전체 이름 (<i>September</i>)
MON	세 자 약어로 나타낸 달
DY	세 자 약어로 나타낸 요일
DAY	요일 전체 이름
DD	숫자로 나타낸 달의 일

ORACLE

유효한 날짜 형식의 예제 형식 요소

요소	설명
SCC 또는 CC	세기. 기원전 날짜에는 -를 접두어로 붙임
날짜의 연도 YYYY 또는 SYYYY	연도. 기원전 날짜에는 -를 접두어로 붙임
YYY 또는 YY 또는 Y	연도의 마지막 세 자리, 두 자리 또는 한 자리
Y,YYY	해당 위치에 쉼표가 있는 연도
IYYY, IYY, IY, I	ISO 표준을 따르는 네 자리, 세 자리, 두 자리 또는 한 자리 연도
SYEAR 또는 YEAR	연도(문자). 기원전 날짜에는 -를 접두어로 붙임
BC 또는 AD	B.C./A.D. 표시자
B.C. 또는 A.D.	마침표가 있는 B.C./A.D. 표시자
Q	일년 중의 분기
MM	월: 두 자리 값
MONTH	9문자 길이가 되도록 공백을 채운 달의 이름
MON	달의 이름. 세 자 약어
RM	로마 숫자 달
WW 또는 W	일년 중의 주(week) 또는 한달 중의 주
DDD 또는 DD 또는 D	일년 중의 일(day), 한달 중의 일 또는 한주 중의 일
DAY	9문자 길이가 되도록 공백을 채운 요일의 이름
DY	요일의 이름. 세 자 약어
J	율리우스력의 일. 기원전 4713년 12월 31일부터의 일 수

날짜 형식 모델 요소

- 시간 요소는 날짜 중 시간 부분의 형식을 지정합니다.

HH24:MI:SS AM 15:45:32 PM

- 문자열은 큰 따옴표로 묶어 추가합니다.

DD "of" MONTH 12 OF OCTOBER

- 숫자 접미어는 숫자를 문자로 표기합니다.

ddspth fourteenth

ORACLE

3-34

Copyright © Oracle Corporation, 2001. All rights reserved.

날짜 형식 요소 - 시간 형식

표에 있는 형식을 사용하여 시간 정보 및 리터럴을 표시하고 숫자를 문자로 변경합니다.

요소	설명
AM 또는 PM	오전/오후 표시자
A.M. 또는 P.M.	마침표가 포함된 오전/오후 표시자
HH 또는 HH12 또는 HH24	하루 중의 시 또는 1-12 또는 0-23으로 표시되는 시
MI	분(0-59)
SS	초(0-59)
SSSSS	자정부터의 초(0-86399)

기타 형식

요소	설명
/ . ,	구두점을 결과에 재사용
“of the”	인용 문자열을 결과에 재사용

숫자 표시에 영향을 주는 접미어 지정

요소	설명
TH	서수(예를 들어, 4TH의 경우 DDTH)
SP	문자로 표현한 숫자(예를 들어, FOUR의 경우 DDSP)
SPTH 또는 THSP	문자로 표현한 서수(예를 들어, FOURTH의 경우 DDSPTH)

날짜에 TO_CHAR 함수 사용

```
SELECT last_name,  
       TO_CHAR(hire_date, 'fmDD Month YYYY')  
          AS HIREDATE  
FROM employees;
```

LAST_NAME	HIREDATE
King	17 June 1987
Kochhar	21 September 1989
De Haan	13 January 1993
Hunold	3 January 1990
Ernst	21 May 1991
Lorentz	7 February 1999
Mourgos	16 November 1999

20 rows selected.

ORACLE

3-36

Copyright © Oracle Corporation, 2001. All rights reserved.

날짜에 TO_CHAR 함수 사용

슬라이드의 SQL 문은 모든 사원의 이름과 입사일을 표시합니다. 입사일은 17 June 1987 형식으로 표시됩니다.

예제

날짜가 Seventh of June 1994 12:00:00 AM 형식으로 표시되도록 슬라이드 예제를 수정합니다.

```
SELECT last_name,  
       TO_CHAR(hire_date,  
              'fmDdspth "of" Month YYYY fmHH:MI:SS AM')  
          AS HIREDATE  
FROM employees;
```

LAST_NAME	HIREDATE
King	Seventeenth of June 1987 12:00:00 AM
Kochhar	Twenty-First of September 1989 12:00:00 AM
Higgins	Seventh of June 1994 12:00:00 AM
Gietz	Seventh of June 1994 12:00:00 AM

20 rows selected.

달 이름은 지정한 형식 모델에 따라 첫 자는 대문자고 나머지는 소문자입니다.

숫자에 TO_CHAR 함수 사용

TO_CHAR(number, 'format_model')

TO_CHAR 함수에는 숫자 값을 문자로 표시할 수 있는 다음과 같은 몇 가지 형식 요소가 있습니다.

9	숫자를 표시합니다.
0	0을 강제로 표시합니다.
\$	부동 \$ 기호를 넣습니다.
L	부동 지역 통화 기호를 사용합니다.
.	소수점을 출력합니다.
,	천 단위 구분자를 출력합니다.

ORACLE

3-37

Copyright © Oracle Corporation, 2001. All rights reserved.

숫자에 TO_CHAR 함수 사용

문자열과 같은 숫자 값을 사용할 때는 NUMBER 데이터 유형의 값을 VARCHAR2 데이터 유형으로 변환하는 TO_CHAR 함수를 사용하여 숫자를 문자 데이터 유형으로 변환해야 합니다. 이 기법은 연결할 때 특히 유용합니다.

숫자 형식 요소

숫자를 문자 데이터 유형으로 변환할 경우 다음 형식 요소를 사용하면 됩니다.

요소	설명	예제	결과
9	숫자 위치입니다. (9의 개수가 표시 폭(width)을 결정합니다.)	999999	1234
0	선행 제로를 표시합니다.	099999	001234
\$	부동 달러 기호입니다.	\$999999	\$1234
L	부동 지역 통화 기호입니다.	L999999	FF1234
.	지정된 위치의 소수점을입니다.	999999.99	1234.00
,	지정된 위치의 쉼표입니다.	999,999	1,234
MI	빼기 기호를 오른쪽에 붙입니다(음수 값).	999999MI	1234-
PR	음수를 괄호로 둡습니다.	999999PR	<1234>
EEEE	과학 표기(형식에 4개의 E를 지정해야 합니다.)	99.999EEEE	1.234E+03
V	10을 n번 곱합니다(n = V 뒤에 나오는 9의 개수).	9999V99	123400
B	0 값을 001 아닌 공백으로 표시합니다.	B9999.99	1234.00

숫자에 TO_CHAR 함수 사용

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY  
FROM employees  
WHERE last_name = 'Ernst';
```

SALARY
\$6,000.00

ORACLE

지침

- 정수가 형식 모델에서 지원하는 자릿수를 초과하는 경우 해시 기호(#)로 표시됩니다.
- Oracle server는 저장된 십진수를 형식 모델에서 지원하는 십진수로 반올림합니다.

TO_NUMBER 및 TO_DATE 함수 사용

- TO_NUMBER 함수를 사용하여 문자열을 숫자 형식으로 변환합니다.

```
TO_NUMBER(char[, 'format_model'])
```

- TO_DATE 함수를 사용하여 문자열을 날짜 형식으로 변환합니다.

```
TO_DATE(char[, 'format_model'])
```

- 이들 함수에는 fx 수정자가 있습니다. 이 수정자는 TO_DATE 함수의 문자 인수 및 날짜 형식 모델이 정확하게 일치하도록 지정합니다.

ORACLE

TO_NUMBER 및 TO_DATE 함수

문자열을 숫자 또는 날짜로 변환해야 하는 경우 TO_NUMBER 또는 TO_DATE 함수를 사용해야 하며 형식 모델은 이전에 설명한 형식 요소를 기반으로 합니다.

“fx” 수정자는 TO_DATE 함수의 문자 인수 및 날짜 형식 모델이 정확하게 일치하도록 지정합니다.

- 문자 인수의 구두점 및 인용 텍스트는 형식 모델의 해당 부분과 대소문자를 제외하고 정확히 일치해야 합니다.
- 문자 인수에 여분의 공백이 있으면 안됩니다. fx가 없는 경우 오라클에서는 여분의 공백을 무시합니다.
- 문자 인수의 숫자 데이터는 형식 모델의 해당 요소와 자릿수가 동일해야 합니다. fx가 없는 경우, 문자 인수의 숫자는 선행 제로를 생략할 수 있습니다.

TO_NUMBER 및 TO_DATE 함수 사용(계속)

예제

1999년 5월 24일에 입사한 모든 사원의 이름과 입사일을 표시합니다. fx 수정자가 사용되므로 정확히 일치해야 하며 'May' 단어 뒤의 공백은 인식되지 않습니다.

```
SELECT last_name, hire_date
  FROM employees
 WHERE hire_date = TO_DATE('May 24, 1999', 'fxMonth DD, YYYY');
```

```
WHERE hire_date = TO_DATE('May 24, 1999', 'fxMonth DD, YYYY')
      *
```

ERROR at line 3:

ORA-01858: a non-numeric character was found where a numeric was expected

RR 날짜 형식

현재 연도	지정한 날짜	RR 형식	YY 형식
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

		지정한 연도(두 자리)	
		0-49	50-99
현재 연도 (두 자리)	0-49	반환일이 현재 세기입니다.	반환일이 이전 세기입니다.
	50-99	반환일이 다음 세기입니다.	반환일이 현재 세기입니다.

ORACLE

3-41

Copyright © Oracle Corporation, 2001. All rights reserved.

RR 날짜 형식 요소

RR 날짜 형식은 YY 요소와 유사하지만 여러 세기를 지정할 수 있습니다. YY 대신 RR 날짜 형식 요소를 사용하면 지정한 두 자리 연도 및 현재 연도의 마지막 두 자리에 따라 반환되는 세기 값이 달라집니다. 슬라이드의 표는 RR 요소를 요약한 것입니다.

현재 연도	주어진 날짜	해석(RR)	해석(YY)
1994	27-OCT-95	1995	1995
1994	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017

RR 날짜 형식 예제

1990년 이전에 고용된 사원을 찾는 경우 RR 형식을 사용하면 명령을 1999년에 실행하든 지금 실행하든 동일한 결과를 얻습니다.

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
  FROM employees
 WHERE hire_date < TO_DATE('01-Jan-90', 'DD-Mon-RR');
```

LAST_NAME	TO_CHAR(HIRE_DATE)
King	17-Jun-1987
Kochhar	21-Sep-1989
Whalen	17-Sep-1987

ORACLE

RR 날짜 형식 요소 예제

1990년 이전에 고용된 사원을 찾는 경우 RR 형식을 사용할 수 있습니다. 현재 연도가 1999보다 크기 때문에 RR 형식은 날짜의 연도 부분을 1950에서 1999 사이로 해석합니다.

반면에 다음 명령은 YY 형식이 날짜의 연도 부분을 현재 세기로 해석하므로(2090), 행(row)이 선택되지 않습니다.

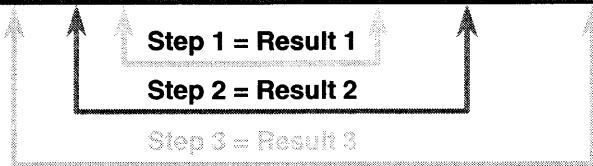
```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-yyyy')
  FROM employees
 WHERE TO_DATE(hire_date, 'DD-Mon-yy') < '01-Jan-1990';

no rows selected
```

중첩 함수

- 단일 행(row) 함수는 여러 번 중첩될 수 있습니다.
- 중첩 함수는 가장 안쪽부터 바깥쪽 순으로 계산됩니다.

F3 (F2 (F1 (col,arg1),arg2),arg3)



ORACLE

중첩 함수

단일 행 함수는 여러 번 중첩될 수 있으며 가장 안쪽부터 바깥쪽 순으로 계산됩니다.
다음 예제는 이러한 함수의 응용성을 보여줍니다.

중첩 함수

```
SELECT last_name,  
       NVL(TO_CHAR(manager_id), 'No Manager')  
  FROM employees  
 WHERE manager_id IS NULL;
```

LAST_NAME	NVL(TO_CHAR(MANAGER_ID), 'NOMANAGER')
King	No Manager

3-44

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

중첩 함수(계속)

슬라이드 예제는 관리자가 없는 회사 대표를 표시합니다. SQL 문 계산은 다음 두 단계를 포함합니다.

- 내부 함수를 실행하여 숫자 값을 문자열로 변환합니다.
 - Result1 = TO_CHAR(manager_id)
- 외부 함수를 실행하여 널 값을 텍스트 문자열로 바꿉니다.
 - NVL(Result1, 'No Manager')

열 별칭이 지정되지 않았으므로 전체 표현식이 열 머리글이 됩니다.

예제

입사일로부터 여섯 달 경과 후 첫번째 금요일의 날짜를 표시합니다. 날짜는 Friday, August 13th, 1999와 같은 형식으로 표시되어야 하며 입사일을 기준으로 결과를 정렬합니다.

```
SELECT    TO_CHAR(NEXT_DAY(ADD_MONTHS  
                           (hire_date, 6), 'FRIDAY'),  
                           'fmDay, Month DDth, YYYY')  
                      "Next 6 Month Review"  
  FROM      employees  
 ORDER BY  hire_date;
```

일반 함수

이들 함수에는 모든 데이터 유형을 사용할 수 있으며 널도 사용할 수 있습니다.

- NVL (*expr1, expr2*)
- NVL2 (*expr1, expr2, expr3*)
- NULLIF (*expr1, expr2*)
- COALESCE (*expr1, expr2, ..., exprn*)

ORACLE

3-45

Copyright © Oracle Corporation, 2001. All rights reserved.

일반 함수

이들 함수에는 모든 데이터 유형을 사용할 수 있으며 표현식 목록에 널 값을 사용할 수 있습니다.

함수	설명
NVL	널 값을 실제 값으로 변환합니다.
NVL2	<i>expr1</i> 이 널이 아닌 경우, NVL2는 <i>expr2</i> 를 반환합니다. <i>expr1</i> 이 널인 경우, NVL2는 <i>expr3</i> 을 반환합니다. 인수 <i>expr1</i> 에는 모든 데이터 유형을 사용할 수 있습니다.
NULLIF	두 표현식을 비교하여 동일한 경우 널을 반환하고 동일하지 않은 경우 첫 번째 표현식을 반환합니다.
COALESCE	표현식 목록에서 널이 아닌 첫 번째 표현식을 반환합니다.

참고: 사용 가능한 수백 가지의 함수에 대한 자세한 내용은 *Oracle9i SQL Reference*, “Functions”를 참조하십시오.

NVL 함수

널을 실제 값으로 변환합니다.

- 사용되는 데이터 유형은 날짜, 문자 및 숫자입니다.
- 데이터 유형은 서로 일치해야 합니다.
 - `NVL(commission_pct, 0)`
 - `NVL(hire_date, '01-JAN-97')`
 - `NVL(job_id, 'No Job Yet')`

ORACLE

3-46

Copyright © Oracle Corporation, 2001. All rights reserved.

NVL 함수

NVL 함수를 사용하여 널 값을 실제 값으로 변환할 수 있습니다.

구문

`NVL (expr1, expr2)`

구문 설명:

`expr1` 널을 포함할 수 있는 소스 값 또는 표현식입니다.

`expr2` 널을 변환할 대상 값입니다.

NVL 함수를 사용하여 모든 데이터 유형을 변환할 수 있지만 반환되는 값은 항상 `expr1`의 데이터 유형과 동일합니다.

여러 데이터 유형에 대한 NVL 변환

데이터 유형	변환 예제
NUMBER	<code>NVL(number_column, 9)</code>
DATE	<code>NVL(date_column, '01-JAN-95')</code>
CHAR 또는 VARCHAR2	<code>NVL(character_column, 'Unavailable')</code>

NVL 함수 사용

```
SELECT last_name, salary, NVL(commission_pct, 0),
       (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
  FROM employees;
```

LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
King	24000	0	268000
Kochhar	17000	0	204000
De Haan	17000	0	204000
Hunold	9000	0	108000
Ernst	6000	0	72000
Lorentz	4200	0	50400
Mourgos	5800	0	69600
Rajs	3500	0	42000
...			

20 rows selected.

ORACLE

3-47

Copyright © Oracle Corporation, 2001. All rights reserved.

NVL 함수

모든 사원의 연간 총수입을 계산하려면 월급에 12를 곱한 후 커미션 비율을 더해야 합니다.

```
SELECT last_name, salary, commission_pct,
       (salary*12) + (salary*12*commission_pct) AN_SAL
  FROM employees;
```

LAST_NAME	SALARY	COMMISSION_PCT	AN_SAL
Vargas	2500		
Zlotkey	10500	.2	151200
Abel	11000	.3	171600
Taylor	8600	.2	123840
...			
Gietz	8300		

20 rows selected.

연간 총수입은 커미션을 받는 사원에 대해서만 계산되며 표현식의 열에 널 값이 있으면 결과는 널입니다. 모든 사원에 대해 값을 계산하려면 산술 연산자를 적용하기 전에 널 값을 숫자 값으로 변환해야 합니다. 슬라이드 예제에서는 널 값을 0으로 변환하는 데 NVL 함수를 사용합니다.

NVL2 함수 사용

```
SELECT last_name, salary, commission_pct,  
      NVL2(commission_pct,  
            'SAL+COMM', 'SAL') income  
FROM employees WHERE department_id IN (50, 80);
```

LAST_NAME	SALARY	COMMISSION_PCT	INCOME
Zlotkey	10500	.2	SAL+COMM
Abel	11000	.3	SAL+COMM
Taylor	8600	.2	SAL+COMM
Mourgos	5000		SAL
Rajs	3500		SAL
Davies	3100		SAL
Matos	2600		SAL
Vargas	2500		SAL

8 rows selected.

ORACLE

3-48

Copyright © Oracle Corporation, 2001. All rights reserved.

NVL2 함수

NVL2 함수는 첫번째 표현식을 검사합니다. 첫번째 표현식이 널이 아닌 경우, NVL2 함수는 두번째 표현식을 반환하며 첫번째 표현식이 널인 경우, 세번째 표현식을 반환합니다.

구문

`NVL(expr1, expr2, expr3)`

구문 설명:

`expr1` 널을 포함할 수 있는 소스 값 또는 표현식입니다.

`expr2` `expr1`이 널이 아닌 경우 반환되는 값입니다.

`expr3` `expr1`이 널인 경우 반환되는 값입니다.

예제에서는 COMMISSION_PCT 열을 검사합니다. 값이 발견되면 두번째 표현식인 SAL+COMM이 반환됩니다. COMMISSION_PCT 열에 널 값이 있으면 세번째 표현식인 SAL이 반환됩니다.

인수 `expr1`에는 모든 데이터 유형을 사용할 수 있습니다. 인수 `expr2` 및 `expr3`에는 LONG을 제외한 모든 데이터 유형을 사용할 수 있습니다. `expr2` 및 `expr3`의 데이터 유형이 서로 다를 경우, Oracle server는 `expr3`이 널 상수가 아니라면 두 표현식을 비교하기 전에 `expr3`을 `expr2`의 데이터 유형으로 변환합니다. `expr3`이 널 상수인 경우에는 데이터 유형 변환이 필요없습니다.

반환 값의 데이터 유형은 항상 `expr2`의 데이터 유형과 동일하며 `expr2`가 문자 데이터인 경우에는 반환 값의 데이터 유형이 VARCHAR2입니다.

NULLIF 함수 사용

```
SELECT first_name, LENGTH(first_name) "expr1",
       last_name, LENGTH(last_name) "expr2",
       NULLIF(LENGTH(first name), LENGTH(last name)) result
  FROM employees;
```

FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
Steven	6	King	4	6
Neena	5	Kochhar	7	5
Lex	3	De Haan	7	3
Alexander	9	Hunold	6	9
Bruce	5	Ernst	5	5
Diana	5	Lorentz	7	5
Kevin	5	Moungos	7	5
Trenna	6	Rajs	4	6
Curtis	6	Davies	6	6

20 rows selected.

ORACLE

NULLIF 함수

NULLIF 함수는 두 표현식을 비교하여 동일한 경우 널을 반환하고 동일하지 않은 경우 첫번째 표현식을 반환합니다. 첫번째 표현식에 널 리터럴을 지정할 수 없습니다.

구문

NULLIF (expr1, expr2)

구문 설명:

expr1 expr2와 비교되는 소스 값입니다.

expr2 expr1과 비교되는 소스 값입니다. 이 값이 expr1과 동일하지 않은 경우 expr1이 반환됩니다.

예제에서는 EMPLOYEES 테이블의 업무 ID와 JOB_HISTORY 테이블의 업무 ID를 비교하여 두 테이블에 모두 존재하는 사원을 검색합니다. 출력은 사원의 현재 업무를 표시합니다. 사원이 두 번 이상 표시되면 해당 사원이 이전에 최소한 두 가지 업무를 가졌음을 의미합니다.

참고: NULLIF 함수는 다음의 CASE 표현식과 논리적으로 동일합니다. CASE 표현식은 다음 페이지에서 설명합니다.

```
CASE WHEN expr1 = expr 2 THEN NULL ELSE expr1 END
```

COALESCE 함수 사용

- COALESCE 함수가 NVL 함수보다 좋은 점은 여러 대체 값을 사용할 수 있다는 점입니다.
- 첫번째 표현식이 널이 아닌 경우 해당 표현식을 반환하며, 널인 경우에는 나머지 표현식에 대해 COALESCE 함수를 적용합니다.

ORACLE

3-50

Copyright © Oracle Corporation, 2001. All rights reserved.

COALESCE 함수

COALESCE 함수는 목록에서 널이 아닌 첫번째 표현식을 반환합니다.

구문

COALESCE (expr1, expr2, ... exprn)

구문 설명:

expr1 이 표현식이 널이 아닌 경우 이 표현식을 반환합니다.

expr2 첫번째 표현식이 널이고 이 표현식이 널이 아닌 경우 이 표현식을 반환합니다.

exprn 앞의 표현식이 모두 널인 경우 이 표현식을 반환합니다.

COALESCE 함수 사용

```
SELECT last_name,
       COALESCE(commission_pct, salary, 10) comm
  FROM employees
 ORDER BY commission_pct;
```

LAST_NAME	COMM
Grant	.15
Zlotkey	.2
Taylor	.2
Abel	.3
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000

20 rows selected.

ORACLE

COALESCE 함수

예제에서, COMMISSION_PCT 값이 널이 아닌 경우, 이 값이 표시됩니다. COMMISSION_PCT 값이 널인 경우, SALARY가 표시됩니다. COMMISSION_PCT와 SALARY 값이 모두 널인 경우, 값 10이 표시됩니다.

조건 표현식

- SQL 문 안에서 IF-THEN-ELSE 논리를 사용할 수 있도록 해줍니다.
- 다음 두 방법을 사용합니다.
 - CASE 표현식
 - DECODE 함수

ORACLE

3-62

Copyright © Oracle Corporation, 2001. All rights reserved.

조건 표현식

SQL 문 안에서 조건 처리(IF-THEN-ELSE 논리)를 구현하는 데 사용되는 두 방법은 CASE 표현식과 DECODE 함수입니다.

참고: CASE 표현식은 Oracle9i Server 릴리스에 새롭게 추가되었습니다. CASE 표현식은 ANSI SQL을 준수합니다. DECODE는 오라클 구문에만 있습니다.

CASE 표현식

IF-THEN-ELSE 문의 역할을 수행하여 조건부 조회를
손쉽게 수행할 수 있습니다.

```
CASE expr WHEN comparison_expr1 THEN return_expr1
           [WHEN comparison_expr2 THEN return_expr2
            WHEN comparison_exprn THEN return_exprn
            ELSE else_expr]
END
```

ORACLE

3-53

Copyright © Oracle Corporation, 2001. All rights reserved.

CASE 표현식

CASE 표현식을 사용하면 프로시저를 호출할 필요 없이 SQL 문에서 IF-THEN-ELSE 논리를 사용할 수 있습니다.

간단한 CASE 표현식의 경우 expr이 comparison_expr과 동일한 첫 번째 WHEN ... THEN 쌍을 찾아서 return_expr을 반환합니다. 이 조건을 만족하는 WHEN ... THEN 쌍이 없고 ELSE 절이 존재하는 경우, else_expr을 반환합니다. 그렇지 않은 경우 널을 반환합니다. return_expr과 else_expr에는 NULL 리터럴을 지정할 수 없습니다.

모든 표현식(expr, comparison_expr 및 return_expr)은 같은 데이터 유형이어야 하며 이러한 데이터 유형으로는 CHAR, VARCHAR2, NCHAR 또는 NVARCHAR2가 가능합니다.

CASE 표현식 사용

IF-THEN-ELSE 문의 역할을 수행하여 조건부 조회를
손쉽게 수행할 수 있습니다.

```
SELECT last_name, job_id, salary,  
       CASE job_id WHEN 'IT_PROG' THEN 1.10*salary  
                     WHEN 'ST_CLERK' THEN 1.15*salary  
                     WHEN 'SA_REP' THEN 1.20*salary  
                ELSE salary END      "REVISED_SALARY"  
FROM employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5600	5800
Rajs	ST_CLERK	3500	4025
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

ORACLE

CASE 표현식 사용

앞 SQL 문의 경우 JOB_ID의 값을 확인하여 JOB_ID가 IT_PROG면 급여가 10% 인상되고,
JOB_ID가 ST_CLERK이면 급여가 15% 인상되며 JOB_ID가 SA_REP면 급여가 20% 인상됩니다. 다른 업무에 대한 급여 인상은 없습니다.

동일한 명령문을 DECODE 함수를 사용하여 작성할 수 있습니다.

DECODE 함수

CASE 또는 IF-THEN-ELSE 문의 역할을 수행하여 조건부 조회를 손쉽게 수행할 수 있습니다.

```
DECODE(col / expression, search1, result1
       [, search2, result2, ...]
       [, default])
```

DECODE 함수

DECODE 함수는 여러 언어에서 사용되는 IF-THEN-ELSE 논리와 유사한 방식으로 표현식을 해독합니다. DECODE 함수는 *expression*을 각 *search* 값과 비교한 후 해독하여 *expression*이 *search* 값과 동일하면 *result*를 반환합니다.

기본값이 생략되면 검색 값에 결과 값과 일치하는 값이 없으므로 널 값이 반환됩니다.

DECODE 함수 사용

```
SELECT last_name, job_id, salary,  
       DECODE(job_id, 'IT_PROG', 1.10*salary,  
              'ST_CLERK', 1.15*salary,  
              'SA REP', 1.20*salary,  
              salary)  
          REVISED_SALARY  
FROM employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

ORACLE

DECODE 함수 사용

앞의 SQL 문에서, JOB_ID의 값이 테스트됩니다. JOB_ID가 IT_PROG면 급여가 10% 인상되고, JOB_ID가 ST_CLERK이면 급여가 15% 인상되며 JOB_ID가 SA REP면 급여가 20% 인상됩니다. 다른 업무에 대한 급여 인상은 없습니다.

동일한 명령문을 IF-THEN-ELSE 문과 같은 의사 코드로 표현할 수 있습니다.

```
IF job_id = 'IT_PROG'      THEN    salary = salary*1.10  
IF job_id = 'ST_CLERK'     THEN    salary = salary*1.15  
IF job_id = 'SA REP'       THEN    salary = salary*1.20  
ELSE salary = salary
```

DECODE 함수 사용

부서 80의 각 사원에 대해 적용할 수 있는 세율을 표시합니다.

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
               0, 0.00,
               1, 0.09,
               2, 0.20,
               3, 0.30,
               4, 0.40,
               5, 0.42,
               6, 0.44,
               0.45) TAX_RATE
  FROM employees
 WHERE department_id = 80;
```

ORACLE

3-57

Copyright © Oracle Corporation, 2001. All rights reserved.

예제

이 슬라이드는 DECODE 함수를 사용하는 다른 예제를 보여줍니다. 이 예제에서는 월급에 따라 부서 80의 각 사원에 대한 세율이 결정됩니다. 세율은 다음 데이터에 표시된 값에 따릅니다.

월급 범위	세율
\$0.00 - 1999.99	00%
\$2,000.00 - 3,999.99	09%
\$4,000.00 - 5,999.99	20%
\$6,000.00 - 7,999.99	30%
\$8,000.00 - 9,999.99	40%
\$10,000.00 - 11,999.99	42%
\$12,200.00 - 13,999.99	44%
\$14,000.00 or greater	45%

LAST_NAME	SALARY	TAX_RATE
Zlotkey	10500	.42
Abel	11000	.42
Taylor	8600	.4

요약

이 단원에서는 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- 함수를 사용하여 데이터에 대한 계산 수행
- 함수를 사용하여 개별 데이터 항목 수정
- 함수를 사용하여 행(row) 그룹에 대한 출력 결과 조작
- 함수를 사용하여 표시할 날짜 형식 변경
- 함수를 사용하여 열 데이터 유형 변환
- NVL 함수 사용
- IF-THEN-ELSE 논리 사용

ORACLE

3-68

Copyright © Oracle Corporation, 2001. All rights reserved.

단일 행(row) 함수

단일 행 함수는 여러 번 중첩될 수 있으며 다음과 같은 데이터에 사용할 수 있습니다.

- 문자 데이터: LOWER, UPPER, INITCAP, CONCAT, SUBSTR, INSTR, LENGTH
- 숫자 데이터: ROUND, TRUNC, MOD
- 날짜 데이터: MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND, TRUNC
- 날짜 값에도 산술 연산자를 사용할 수 있습니다.
- 변환 함수는 문자, 날짜 및 숫자 값을 변환합니다(TO_CHAR, TO_DATE, TO_NUMBER).
- NVL, NVL2, NULLIF, COALESCE 등을 포함하여 널을 사용할 수 있는 함수가 몇 가지 있습니다.
- IF-THEN-ELSE 논리는 CASE 표현식 또는 DECODE 함수를 사용하여 SQL 문 안에 적용할 수 있습니다.

SYSDATE 및 DUAL

SYSDATE는 현재 날짜 및 시간을 반환하는 날짜 함수입니다. 일반적으로 DUAL이라는 더미 테이블(dummy table)에서 SYSDATE를 선택합니다.

연습 3, 2부: 개요

이 연습에서는 다음 내용을 다릅니다.

- 숫자, 문자 및 날짜 함수를 사용하는 질의 작성
- 함수에 연결 사용
- 대소문자를 구분하지 않는 질의를 작성하여 문자 함수의 유용성 테스트
- 사원의 근무 연수 및 달 수 계산
- 사원에 대한 검토일 판별

ORACLE

3-59

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 3, 2부: 개요

이 연습은 문자, 숫자 및 날짜 데이터 유형에 사용 가능한 여러 함수를 사용하는 다양한 연습 문제를 제공합니다.

중첩 함수의 결과는 가장 안쪽 함수부터 바깥쪽 순으로 계산됩니다.

연습 3 - 1부(계속)

1. 현재 날짜를 표시하는 질의를 작성하고 열 레이블을 Date로 지정하십시오.

Date
28-SEP-01

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.
3. lab3_2.sql 파일의 질의를 실행하십시오.

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	
100	King	24000	27600	
101	Kochhar	17000	19550	
102	De Haan	17000	19550	
103	Hunold	9000	10350	
■ ■ ■				
202	Fay	6000	6900	
205	Higgins	12000	13800	
206	Gietz	8300	9545	

20 rows selected.

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.

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
100	King	24000	27600	3600
101	Kochhar	17000	19550	2550
102	De Haan	17000	19550	2550
103	Hunold	9000	10350	1350
104	Ernst	6000	6900	900
107	Lorentz	4200	4830	630
124	Mourgos	5800	6670	870
141	Rajs	3500	4025	525
142	Davies	3100	3565	465
143	Matos	2600	2990	390
■ ■ ■				
201	Hartstein	13000	14950	1950
202	Fay	6000	6900	900
205	Higgins	12000	13800	1800
206	Gietz	8300	9545	1245

20 rows selected.

연습 3 - 1부(계속)

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 지정 하십시오. 결과를 사원의 이름에 따라 정렬하십시오.

Name	Length
Abel	4
Matos	5
Mourgos	7

연습 3 - 2부

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 각기 다를 수 있습니다.

LAST_NAME	MONTHS_WORKED
Zlotkey	20
Mourgos	22
Grant	28
Lorentz	32
Vargas	39
Taylor	42
Matos	42
Fay	49
Davies	56
Abel	65
Hartstein	67
Rajs	71
Higgins	88
Gietz	88

LAST_NAME	MONTHS_WORKED
De Haan	105
Ernst	124
Hunold	141
Kochhar	144
Whalen	168
King	171

20 rows selected.

연습 3 - 2부(계속)

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.

Dream Salaries
King earns \$24,000.00 monthly but wants \$72,000.00.
Kochhar earns \$17,000.00 monthly but wants \$51,000.00.
De Haan earns \$17,000.00 monthly but wants \$51,000.00.
Hunold earns \$9,000.00 monthly but wants \$27,000.00.
Ernst earns \$6,000.00 monthly but wants \$18,000.00.
Lorentz earns \$4,200.00 monthly but wants \$12,600.00.
Mourgos earns \$5,800.00 monthly but wants \$17,400.00.
Rajs earns \$3,500.00 monthly but wants \$10,500.00.
Davies earns \$3,100.00 monthly but wants \$9,300.00.
Matos earns \$2,600.00 monthly but wants \$7,800.00.
Vargas earns \$2,500.00 monthly but wants \$7,500.00.
■ ■ ■
Gietz earns \$8,300.00 monthly but wants \$24,900.00.

20 rows selected.

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.

LAST_NAME	SALARY
King	\$\$\$\$\$\$\$\$\$\$24000
Kochhar	\$\$\$\$\$\$\$\$\$\$17000
De Haan	\$\$\$\$\$\$\$\$\$\$17000
Hunold	\$\$\$\$\$\$\$\$\$\$9000
Ernst	\$\$\$\$\$\$\$\$\$\$6000
Lorentz	\$\$\$\$\$\$\$\$\$\$4200
Mourgos	\$\$\$\$\$\$\$\$\$\$5800
Rajs	\$\$\$\$\$\$\$\$\$\$3500
■ ■ ■	
Higgins	\$\$\$\$\$\$\$\$\$\$12000
Gietz	\$\$\$\$\$\$\$\$\$\$8300

20 rows selected.

연습 3 - 2부(계속)

9. 사원의 이름, 입사일 및 급여 겸토일을 표시하십시오. 급여 겸토일은 여섯 달이 경과한 후 첫번째 월요일입니다. 열 레이블을 REVIEW로 지정하고 날짜는 “Monday, the Thirty-First of July, 2000”과 같은 형식으로 표시되도록 지정하십시오.

LAST_NAME	HIRE_DATE	REVIEW
King	17-JUN-87	Monday, the Twenty-First of December, 1987
Kochhar	21-SEP-89	Monday, the Twenty-Sixth of March, 1990
De Haan	13-JAN-93	Monday, the Nineteenth of July, 1993
Hunold	03-JAN-90	Monday, the Ninth of July, 1990
Ernst	21-MAY-91	Monday, the Twenty-Fifth of November, 1991
Lorentz	07-FEB-99	Monday, the Ninth of August, 1999
Mourgos	16-NOV-99	Monday, the Twenty-Second of May, 2000
Rajs	17-OCT-95	Monday, the Twenty-Second of April, 1996
Davies	29-JAN-97	Monday, the Fourth of August, 1997
■ ■ ■		
Gietz	07-JUN-94	Monday, the Twelfth of December, 1994

20 rows selected.

10. 이름, 입사일 및 업무 시작 요일을 표시하고 열 레이블을 DAY로 지정하십시오.
Monday를 시작으로 해서曜일을 기준으로 결과를 정렬하십시오.

LAST_NAME	HIRE_DATE	DAY
Grant	24-MAY-99	MONDAY
Ernst	21-MAY-91	TUESDAY
Mourgos	16-NOV-99	TUESDAY
Taylor	24-MAR-98	TUESDAY
Rajs	17-OCT-95	TUESDAY
Gietz	07-JUN-94	TUESDAY
Higgins	07-JUN-94	TUESDAY
King	17-JUN-87	WEDNESDAY
De Haan	13-JAN-93	WEDNESDAY
■ ■ ■		
Abel	11-MAY-96	SATURDAY
Lorentz	07-FEB-99	SUNDAY
Fay	17-AUG-97	SUNDAY
Matos	15-MAR-98	SUNDAY

20 rows selected.

연습 3 - 2부(계속)

좀 더 연습 하려면 다음 문제를 풀어보십시오.

11. 사원의 이름과 커미션 합계를 표시하는 질의를 작성하십시오. 커미션을 받지 않는 사원
일 경우 “No Commission”을 표시하십시오. 열 레이블은 COMM으로 지정하십시오.

LAST_NAME	COMM
King	No Commission
Kochhar	No Commission
De Haan	No Commission
Hunold	No Commission
Ernst	No Commission
Lorentz	No Commission
Mourgos	No Commission
Rajs	No Commission
Davies	No Commission
Matos	No Commission
Vargas	No Commission
Zlotkey	.2
Abel	.3
Taylor	.2
■ ■ ■	
Gietz	No Commission

20 rows selected.

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.

EMPLOYEE_AND_THEIR_SALARIES
King *****
Kochhar *****
De Haan *****
Hartstei *****
Higgins *****
Abel *****
■ ■ ■
Vargas **

20 rows selected.

연습 3 - 2부(계속)

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
기타	0

JOB_ID	G
AD_PRES	A
AD_VP	0
AD_VP	0
IT_PROG	C
IT_PROG	C
IT_PROG	C
ST_MAN	B
ST_CLERK	E
ST_CLERK	E
ST_CLERK	E
AC_MGR	0
AC_ACCOUNT	0

20 rows selected.

14. 앞 문제의 명령문을 CASE 구문을 사용하여 재작성하십시오.

4

여러 테이블의
데이터 표시

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 등가 조인 및 비등가 조인을 사용하여 여러 테이블의 데이터를 액세스하는 **SELECT** 문 작성
- 포괄 조인을 사용하여 일반적으로 조인 조건을 만족하지 않는 데이터 보기
- 자체 조인을 사용하여 테이블 자체 조인

ORACLE

단원 목표

이 단원에서는 여러 테이블에서 데이터를 얻는 방법을 설명합니다.

여러 테이블에서 데이터 얻기

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1900
60	IT	1400
80	Sales	2900
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

ORACLE

여러 테이블에서 데이터 얻기

여러 테이블의 데이터를 사용해야 하는 경우가 있습니다. 슬라이드 예제에서 보고서는 별개의 두 테이블에 있는 데이터를 표시합니다.

- 사원 ID는 EMPLOYEES 테이블에 존재합니다.
- 부서 ID는 EMPLOYEES 및 DEPARTMENTS 테이블 모두에 존재합니다.
- 위치 ID는 DEPARTMENTS 테이블에 존재합니다.

보고서를 생성하려면 EMPLOYEES 테이블과 DEPARTMENTS 테이블을 링크한 후 두 테이블에 있는 데이터를 액세스해야 합니다.

카티시안 곱(Cartesian Product)

- 카티시안 곱은 다음 경우에 생성됩니다.
 - 조인 조건을 생략한 경우
 - 조인 조건이 부적합한 경우
 - 첫번째 테이블의 모든 행이 두번째 테이블의 모든 행에 조인된 경우
- 카티시안 곱이 생성되지 않도록 하려면 WHERE 절에 항상 유효한 조인 조건을 포함시키십시오.

ORACLE

4-4

Copyright © Oracle Corporation, 2001. All rights reserved.

카티시안 곱(Cartesian Product)

조인 조건이 부적합하거나 조인 조건을 완전히 생략한 경우 행의 모든 조합을 표시하는 카티시안 곱이 생성됩니다. 첫번째 테이블의 모든 행이 두번째 테이블의 모든 행에 조인됩니다.

카티시안 곱은 너무 많은 행을 생성하므로 결과가 별로 유용하게 사용되지 않습니다. 특별히 모든 테이블에 있는 모든 행을 조합해야 하는 경우가 아니라면 항상 WHERE 절에 적합한 조인 조건을 포함시키도록 합니다.

카티시안 곱은 적정 양의 데이터를 시뮬레이트하기 위해 많은 수의 행을 생성해야 하는 경우와 같은 일부 테스트에서 유용합니다.

카티시안 곱(Cartesian Product) 생성

EMPLOYEES (20개 행)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

20 rows selected.

DEPARTMENTS (8개 행)

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
30	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
130	Contracting	1700

8 rows selected.

카티시안

곱:

20x8=160개 행

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
100	90	1700
101	90	1700
102	90	1700
103	60	1700
104	60	1700
107	60	1700

...

160 rows selected.

ORACLE

카티시안 곱(계속)

카티시안 곱은 조인 조건을 생략한 경우 생성됩니다. 슬라이드 예제는 EMPLOYEES 테이블과 DEPARTMENTS 테이블에서 사원의 이름과 부서 이름을 표시합니다. WHERE 절을 지정하지 않았으므로 EMPLOYEES 테이블의 모든 행(20개 행)이 DEPARTMENTS 테이블의 모든 행(8개 행)과 조인되어 결과적으로 160개 행(row)이 생성됩니다.

```
SELECT last_name, department_name dept_name
FROM   employees, departments;
```

LAST_NAME	DEPT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
...	

160 rows selected.

조인 유형

오라클 전용

조인(*8i* 이전):

- 등가 조인
- 비등가 조인
- 포괄 조인
- 자체 조인

SQL: 1999

표준 조인:

- 교차 조인
- 자연 조인
- **Using** 절
- 전체 또는 한쪽 포괄 조인
- 포괄 조인에 대한 임의의 조인 조건

조인 유형

Oracle*9i* 데이터베이스는 SQL: 1999 표준을 준수하는 조인 구문을 제공합니다. *9i* 릴리스 이전의 조인 구문은 ANSI 표준과 달랐습니다. 새로운 SQL: 1999 표준 조인 구문은 이전 릴리스에서 제공하던 오라클 전용 조인 구문의 성능 상의 이점을 제공하지 않습니다.

오라클 구문을 사용하여 테이블 조인

조인을 사용하여 여러 테이블의 데이터를 질의합니다.

```
SELECT    table1.column, table2.column  
FROM      table1, table2  
WHERE     table1.column1 = table2.column2;
```

- WHERE 절에서 조인 조건을 작성합니다.
- 동일한 열 이름이 여러 테이블에 있는 경우 열 이름에 테이블 이름을 접두어로 붙입니다.

ORACLE

4-7

Copyright © Oracle Corporation, 2001. All rights reserved.

조인 정의

데이터베이스에서 여러 테이블의 데이터가 필요한 경우 조인 조건을 사용합니다. 서로 대응되는 열 즉, 일반적으로 기본 키 및 외래 키 열에 존재하는 공통 값에 따라 한 테이블의 행(row)을 다른 테이블의 행에 조인할 수 있습니다.

관련된 둘 이상의 테이블에 있는 데이터를 표시하려면 WHERE 절에서 간단한 조인 조건을 작성하십시오.

구문 설명:

table1.column 데이터를 검색할 테이블 및 열입니다.
table1.column1 = 테이블을 조인하거나 관련시키는 조건입니다.
table2.column2

지침

- 테이블을 조인하는 SELECT 문을 작성할 때 열 이름에 테이블 이름을 접두어로 붙여 해당 열을 확실히 식별할 수 있도록 하여 데이터베이스 액세스 기능을 향상시킵니다.
- 여러 테이블에 동일한 열 이름이 있는 경우 열 이름에 테이블 이름을 접두어로 붙여야 합니다.
- n개의 테이블을 조인하려면 최소 n-1개의 조인 조건이 필요합니다. 예를 들어 네 개의 테이블을 조인하려면 최소 세 개의 조인 조건이 필요합니다. 이 규칙은 테이블에 복합 기본 키가 있으면 적용되지 않으며, 이 때 복합 기본 키는 각 행(row)을 고유하게 식별하기 위해 둘 이상의 열이 필요합니다.

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

등가 조인이란?

EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
200	10
201	20
202	20
124	50
141	50
142	50
143	50
144	50
103	60
104	60
107	60
149	80
174	80
176	80

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
60	IT
60	IT
60	IT
80	Sales
80	Sales
80	Sales

외래 키

기본 키

ORACLE

등가 조인

사원의 부서 이름을 판별하기 위해 EMPLOYEES 테이블의 DEPARTMENT_ID 열의 값을 DEPARTMENTS 테이블의 DEPARTMENT_ID 값과 비교합니다. EMPLOYEES 테이블과 DEPARTMENTS 테이블의 관계는 등가 조입니다. 즉 두 테이블의 DEPARTMENT_ID 열 값이 동일해야 합니다. 이 조인 유형은 주로 기본 키와 외래 키를 보조 수단으로 포함합니다.

참고: 등가 조인을 단순 조인 또는 내부 조인(*inner join*)이라고도 합니다.

등가 조인으로 레코드 검색

```
SELECT employees.employee_id, employees.last_name,  
       employees.department_id, departments.department_id,  
       departments.location_id  
  FROM employees, departments  
 WHERE employees.department_id = departments.department_id;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1600
202	Fay	20	20	1600
124	Moungos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500
144	Vargas	50	50	1500

19 rows selected.

ORACLE

4-9

Copyright © Oracle Corporation, 2001. All rights reserved.

등가 조인으로 레코드 검색

슬라이드 예제 설명:

- SELECT 절은 검색할 열 이름을 지정합니다.
 - EMPLOYEES 테이블에 있는 사원의 이름, 사원 번호 및 부서 번호 열
 - DEPARTMENTS 테이블에 있는 부서 번호, 부서 이름 및 위치 ID 열
- FROM 절은 데이터베이스가 액세스할 두 테이블을 지정합니다.
 - EMPLOYEES 테이블
 - DEPARTMENTS 테이블
- WHERE 절은 테이블 조인 방법을 지정합니다.

EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID

DEPARTMENT_ID 열은 두 테이블 모두에 있으므로 테이블 이름을 접두어로 붙여 확실히 식별할 수 있도록 합니다.

AND 연산자를 사용한 추가 검색 조건

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
Whalen	10
Hartstein	20
Fay	20
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Hunold	60
Ernst	60

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
60	IT
60	IT

ORACLE

추가 검색 조건

테이블을 조인할 경우 조인뿐 아니라 고려 대상인 행을 제한하기 위해 WHERE 절에 조건을 추가해야 하는 경우가 있습니다. 예를 들어, 사원 Matos의 부서 번호 및 부서 이름을 표시하려면 WHERE 절에 조건을 추가해야 합니다.

```
SELECT last_name, employees.department_id,
       department_name
  FROM employees, departments
 WHERE employees.department_id = departments.department_id
   AND last_name = 'Matos';
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Matos	50	Shipping

모호한 열 이름 자세히 지정

- 테이블 접두어를 사용하여 여러 테이블에 있는 열 이름을 자세히 지정합니다.
- 테이블 접두어를 사용하여 성능을 개선합니다.
- 열 별칭을 사용하여 다른 테이블에 있는 동일한 이름의 열을 구별합니다.

ORACLE

4-11

Copyright © Oracle Corporation, 2001. All rights reserved.

모호한 열 이름 자세히 지정

WHERE 절에서 테이블 이름을 사용해 열 이름을 자세히 지정하여 모호성을 방지해야 합니다. 테이블 접두어를 사용하지 않으면 DEPARTMENTS 테이블 또는 EMPLOYEES 테이블 중 하나에서 DEPARTMENT_ID 열을 가져오게 되므로 반드시 테이블 접두어를 추가하여 질의를 실행하도록 합니다.

두 테이블에 공통된 열 이름이 없으면 열 이름을 자세히 지정하지 않아도 되기는 하지만 테이블 접두어를 사용하면 열을 찾을 위치를 정확히 알 수 있으므로 성능이 개선됩니다.

SELECT 절이나 ORDER BY 절 등 다른 절에서도 모호한 열 이름을 자세히 지정해야 합니다.

테이블 별칭 사용

- 테이블 별칭을 사용하여 질의를 단순화합니다.
- 테이블 접두어를 사용하여 성능을 개선합니다.

```
SELECT [e].employee_id, [e].last_name, [e].department_id,  
       [d].department_id, [d].location_id  
  FROM   employees [e], departments [d]  
 WHERE  [e].department_id = [d].department_id;
```

ORACLE

4-12

Copyright © Oracle Corporation, 2001. All rights reserved.

테이블 별칭

테이블 이름을 사용하여 열 이름을 자세히 지정하는 경우 시간이 많이 걸리며 테이블 이름이 길 때는 더욱 오래 걸립니다. 이 때 테이블 이름 대신 테이블 별칭을 사용할 수 있습니다. 열 별칭이 열에 다른 이름을 부여하듯이 테이블 별칭은 테이블에 다른 이름을 부여합니다. 테이블 별칭을 사용하면 SQL 코드를 적게 작성해도 되므로 메모리 사용이 줄어듭니다.

예제는 FROM 절에서 테이블 별칭을 식별하는 방법을 나타냅니다. 테이블 이름은 전체 이름, 공백, 테이블 별칭 순으로 지정되어 있으며 EMPLOYEES 테이블에는 별칭 e가, DEPARTMENTS 테이블에는 별칭 d가 부여되었습니다.

지침

- 테이블 별칭은 최대 30자까지 가능하지만 짧을수록 좋습니다.
- FROM 절에서 특정 테이블 이름에 대해 테이블 별칭을 사용하는 경우 해당 테이블 별칭은 SELECT 문에서도 해당 테이블 이름을 대신합니다.
- 테이블 별칭은 의미 있는 것으로 지정해야 합니다.
- 테이블 별칭은 현재 SELECT 문에 대해서만 유효합니다.

세 개 이상의 테이블 조인

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
King	90
Kochhar	90
De Haan	90
Hunold	60
Ernst	60
Lorentz	60
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Zlotkey	60
Apel	60
Taylor	90
■ ■ ■	

20 rows selected.

DEPARTMENTS

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

LOCATIONS

LOCATION_ID	CITY
1400	Southlake
1500	South San Francisco
1700	Seattle
1800	Toronto
2500	Oxford

- * **n**개의 테이블을 조인하려면 최소 **n-1**개의 조인 조건이 필요합니다. 예를 들어 세 개의 테이블을 조인하려면 최소 두 개의 조인 조건이 필요합니다.

ORACLE

추가 검색 조건

세 개 이상의 테이블을 조인할 경우가 있습니다. 예를 들어, 각 사원의 이름, 부서 이름 및 도시를 표시 하려면 EMPLOYEES, DEPARTMENTS 및 LOCATIONS 테이블을 조인해야 합니다.

```
SELECT e.last_name, d.department_name, l.city
  FROM employees e, departments d, locations l
 WHERE e.department_id = d.department_id
   AND d.location_id = l.location_id;
```

LAST_NAME	DEPARTMENT_NAME	CITY
Hunold	IT	Southlake
Ernst	IT	Southlake
Lorentz	IT	Southlake
Mourgos	Shipping	South San Francisco
Rajs	Shipping	South San Francisco
Davies	Shipping	South San Francisco
■ ■ ■		

19 rows selected.

비등가 조인

EMPLOYEES

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000
Ernst	6000
Lorentz	4200
Moungos	5000
Rajs	3600
Davies	3100
Matos	2600
Vargas	2500
Zlotkey	10500
Abel	11000
Taylor	8600
...	

20 rows selected.

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



EMPLOYEES 테이블의
급여는 JOBGRADES
테이블의 최저 급여와
최고 급여 사이에
있어야 합니다.

ORACLE

비등가 조인

비등가 조인은 동등 연산자가 아닌 연산자를 포함하는 조인 조건입니다.

EMPLOYEES 테이블과 JOBGRADES 테이블의 관계에서 비등가 조인의 예를 볼 수 있습니다.

두 테이블의 관계는 EMPLOYEES 테이블의 SALARY 열이 JOBGRADES 테이블의
LOWEST_SALARY 열 값과 HIGHEST_SALARY 열 값 사이에 있어야 합니다. 관계는 등호(=) 이외의
연산자를 사용하여 얻습니다.

비등가 조인으로 레코드 검색

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e, job_grades j
WHERE  e.salary
       BETWEEN j.lowest_sal AND j.highest_sal;
```

LAST_NAME	SALARY	GRADE
Matos	2600	A
Vargas	2500	A
Lorentz	4200	B
Mourgos	5800	B
Rajs	3500	B
Davies	3100	B
Whalen	4400	B
Hunold	9000	C
Ernst	6000	C

20 rows selected.

ORACLE

4-15

Copyright © Oracle Corporation, 2001. All rights reserved.

비등가 조인(계속)

슬라이드 예제는 비등가 조인을 생성하여 사원의 급여 등급을 계산하는데, 급여는 해당 등급의 최저 급여와 최고 급여 사이에 있어야 합니다.

이 질의가 실행되면 모든 사원이 정확히 한 번만 표시된다는 점에 유의하십시오. 다음 두 가지 이유로 인해 사원이 목록에 반복 표시되지 않습니다.

- 업무 등급 테이블에 겹치는 등급을 포함하는 행이 없습니다. 즉 사원의 급여 값은 급여 등급 테이블의 한 행에 있는 최저 급여와 최고 급여 사이에만 있습니다.
- 모든 사원의 급여는 업무 등급 테이블에 의해 제한됩니다. 즉 LOWEST_SAL 열의 최저 급여보다 적게 받거나 HIGHEST_SAL 열의 최고 급여보다 많이 받는 사원은 없습니다.

참고: <= 및 >= 등의 다른 조건을 사용할 수 있지만 BETWEEN이 가장 간단합니다. BETWEEN을 사용할 경우 낮은 값을 먼저 지정하고 높은 값을 나중에 지정해야 합니다.

슬라이드 예제에서 테이블 별칭을 지정한 것은 모호성 때문이 아니고 성능 상의 이유 때문입니다.

포괄 조인

DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

8 rows selected.

EMPLOYEES

DEPARTMENT_ID	LAST_NAME
90	King
90	Kochhar
90	De Haan
60	Hunold
60	Ernst
60	Lorentz
50	Mourgos
50	Rajs
50	Davies
50	Matos
50	Vargas
80	Zlotkey

20 rows selected.

부서 190에는 사원이
없습니다.

ORACLE

포괄 조인을 사용하여 일치하는 항목이 없는 레코드 반환

행이 조인 조건을 만족하지 않으면 질의 결과에 나타나지 않습니다. 예를 들어, EMPLOYEES 및 DEPARTMENTS 테이블의 등가 조인 조건에서 사원 Grant의 부서 ID가 EMPLOYEES 테이블에 기록되어 있지 않기 때문에 Grant는 나타나지 않습니다. 결과 집합에는 20명의 사원 레코드가 표시되지 않고 19명의 사원 레코드가 표시됩니다.

```
SELECT e.last_name, e.department_id, d.department_name
  FROM employees e, departments d
 WHERE e.department_id = d.department_id;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping

19 rows selected.

포괄 조인 구문

- 포괄 조인을 사용하면 조인 조건을 만족하지 않는 행도 볼 수 있습니다.
- 포괄 조인 연산자는 더하기 기호(+)입니다.

```
SELECT table1.column, table2.column  
FROM   table1, table2  
WHERE  table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column  
FROM   table1, table2  
WHERE  table1.column = table2.column(+);
```

ORACLE

4-17

Copyright © Oracle Corporation, 2001. All rights reserved.

포괄 조인을 사용하여 일치하는 항목이 없는 레코드 반환

조인 조건에 포괄 조인 연산자를 사용하면 누락된 행을 반환할 수 있습니다. 이 연산자는 더하기 기호를 괄호로 묶어(+) 표시하며 정보가 부족한 조인 “옆”에 넣습니다. 이 연산자는 하나 이상의 널 행을 생성하는데 이러한 널 행은 완전한 테이블의 하나 이상의 행과 조인할 수 있습니다.

구문 설명:

table1.column = 테이블을 조인하거나 관련시키는 조건입니다.

table2.column (+) 포괄 조인 기호로서 WHERE 절 조건의 한 쪽에만 넣을 수 있으며 양쪽에 넣을 수 없습니다.(일치하는 행이 없는 테이블에서는 열 이름 다음에 포괄 조인 기호를 넣습니다.)

포괄 조인 사용

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

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mouros	50	Shipping
Rajs	50	Shipping
Davies	50	Shipping
Matos	50	Shipping

Gietz	110	Accounting
		Contracting

20 rows selected.

ORACLE

포괄 조인을 사용하여 일치하는 항목이 없는 레코드 반환(계속)

슬라이드 예제는 사원의 이름, 부서 ID 및 부서 이름을 표시합니다. Contracting 부서에는 사원이 없으므로 출력 결과에 공백 값이 표시됩니다.

포괄 조인 제한 사항

- 포괄 조인 연산자는 표현식의 한 쪽 즉, 정보가 누락된 쪽에만 표시할 수 있으며 상대편 테이블과 직접 일치하는 행이 없는 테이블의 행을 반환합니다.
- 포괄 조인을 포함하는 조건은 IN 연산자를 사용할 수 없으며 OR 연산자를 사용해 다른 조건에 링크할 수 없습니다.

자체 조인

EMPLOYEES (WORKER)

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos

WORKER 테이블의 MANAGER_ID는 MANAGER 테이블의 EMPLOYEE_ID와 같습니다.

ORACLE

테이블 자체 조인

테이블을 자체 조인해야 하는 경우가 있습니다. 각 사원의 관리자 이름을 찾으려면 EMPLOYEES 테이블 자체를 조인해야 합니다. 즉 자체 조인을 수행해야 합니다. 예를 들어, Whalen의 관리자 이름을 찾으려면 다음을 수행해야 합니다.

- EMPLOYEES 테이블의 LAST_NAME 열에서 Whalen을 찾습니다.
- MANAGER_ID 열에서 Whalen의 관리자 번호를 찾습니다. Whalen의 관리자 번호는 101입니다.
- LAST_NAME 열에서 EMPLOYEE_ID가 101인 관리자의 이름을 찾습니다. Kochhar의 사원 번호가 101이므로 Kochhar가 Whalen의 관리자입니다.

이 프로세스에서는 테이블을 두 번 검색합니다. 첫 번째는 테이블의 LAST_NAME 열에서 Whalen을 찾고 MANAGER_ID 열에서 101 값을 찾습니다. 두 번째는 EMPLOYEE_ID 열에서 101을 찾고 LAST_NAME 열에서 Kochhar를 찾습니다.

테이블 자체 조인

```
SELECT worker.last_name || ' works for '
    || manager.last_name
  FROM employees worker, employees manager
 WHERE worker.manager_id = manager.employee_id;
```

WORKER LAST NAME	WORKS FOR	MANAGER LAST NAME
Kochhar	works for	King
De Haan	works for	King
Morgos	works for	King
Zlotkey	works for	King
Hartstein	works for	King
Whalen	works for	Kochhar
Higgins	works for	Kochhar
Hunold	works for	De Haan
Ernst	works for	Hunold
...		

19 rows selected.

ORACLE

테이블 자체 조인(계속)

슬라이드 예제는 EMPLOYEES 테이블을 자체 조인합니다. FROM 절에 있는 두 테이블을 시뮬레이트하기 위해서 동일한 EMPLOYEES 테이블에 대해 w와 m이라는 두 별칭을 사용합니다.

이 예제에서 WHERE 절은 “사원의 관리자 번호가 관리자의 사원 번호와 일치하는 경우”를 의미하는 조인을 포함합니다.

연습 4, 1부: 개요

이 연습에서는 오라클 구문을 사용하여 테이블을 조인하는 질의 작성을 다룹니다.

ORACLE

4-21

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 4, 1부

이 연습은 이 단원에서 설명한 오라클 구문을 사용하여 테이블을 조인하는 다양한 연습 문제를 제공합니다.

이 단원의 마지막에 있는 연습 문제 1-4를 풀어보십시오.

SQL: 1999 구문을 사용한 테이블 조인

조인을 사용하여 여러 테이블의 데이터를 질의합니다.

```
SELECT    table1.column, table2.column
FROM      table1
[CROSS JOIN table2] |
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON(table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)];
```

ORACLE

4-22

Copyright © Oracle Corporation, 2001. All rights reserved.

조인 정의

SQL: 1999 구문을 사용하여 이전 페이지에 표시된 것과 동일한 결과를 얻을 수 있습니다.

구문 설명:

table1.column	데이터를 검색할 테이블 및 열입니다.
CROSS JOIN	두 테이블의 카티시안 곱(Cartesian Product)을 반환합니다.
NATURAL JOIN	동일한 열 이름을 기준으로 두 테이블을 조인합니다.
JOIN table	
USING column_name	열 이름을 기준으로 등가 조인을 수행합니다.
JOIN table ON	
table1.column_name = table2.column_name	ON 절의 조건을 기준으로 등가 조인을 수행합니다.
LEFT/RIGHT/FULL OUTER	

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

교차 조인 작성

- CROSS JOIN 절은 두 테이블 상호간의 조합을 생성합니다.
- 이것은 두 테이블 사이의 카티시안 곱(Cartesian Product)과 동일합니다.

```
SELECT last_name, department_name  
FROM employees  
CROSS JOIN departments ;
```

LAST_NAME	DEPARTMENT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
Hunold	Administration
...	

160 rows selected.

ORACLE

4-23

Copyright © Oracle Corporation, 2001. All rights reserved.

교차 조인 작성

슬라이드 예제는 다음과 동일한 결과를 생성합니다.

```
SELECT last_name, department_name  
FROM employees, departments;
```

LAST_NAME	DEPARTMENT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
Hunold	Administration
Ernst	Administration
...	

160 rows selected.

자연 조인 작성

- **NATURAL JOIN** 절은 두 테이블에서 동일한 이름을 가진 모든 열을 기준으로 합니다.
- 두 테이블의 일치하는 모든 열에서 같은 값을 가진 행을 선택합니다.
- 동일한 이름을 가진 열의 데이터 유형이 서로 다를 경우 오류가 반환됩니다.

ORACLE®

4-24

Copyright © Oracle Corporation, 2001. All rights reserved.

자연 조인 작성

오라클의 이전 릴리스에서는 해당 테이블의 열을 명시적으로 지정하지 않으면 조인을 수행할 수 없었습니다. Oracle9i의 경우, NATURAL JOIN 키워드를 사용하면 일치하는 데이터 유형 및 이름을 가진 두 테이블의 열을 기준으로 자동으로 조인을 수행할 수 있습니다.

참고: 조인은 두 테이블 모두에서 동일한 이름과 데이터 유형을 가진 열에서만 가능합니다. 열의 이름이 같지만 데이터 유형이 다를 경우 NATURAL JOIN 구문에서 오류가 발생합니다.

자연 조인으로 레코드 검색

```
SELECT department_id, department_name,
       location_id, city
  FROM departments
NATURAL JOIN locations ;
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
60	IT	1400	Southlake
60	Shipping	1500	South San Francisco
10	Administration	1700	Seattle
90	Executive	1700	Seattle
110	Accounting	1700	Seattle
190	Contracting	1700	Seattle
20	Marketing	1800	Toronto
80	Sales	2500	Oxford

8 rows selected.

ORACLE

4-25

Copyright © Oracle Corporation, 2001. All rights reserved.

자연 조인으로 레코드 검색

슬라이드 예제에서, LOCATIONS 테이블은 DEPARTMENT 테이블과 두 테이블 사이에서 유일하게 이름이 같은 열인 LOCATION_ID 열을 통해 조인됩니다. 다른 공통 열이 존재할 경우에는 이들 열도 함께 조인에 사용됩니다.

등가 조인

자연 조인을 등가 조인으로도 작성할 수 있습니다.

```
SELECT department_id, department_name,
       departments.location_id, city
  FROM departments, locations
 WHERE departments.location_id = locations.location_id;
```

WHERE 절을 포함하는 자연 조인

WHERE 절을 사용하여 자연 조인에 제한을 추가할 수 있습니다. 아래 예제는 부서 ID가 20 또는 50인 행(row)으로 출력 결과를 제한합니다.

```
SELECT department_id, department_name,
       location_id, city
  FROM departments
NATURAL JOIN locations
 WHERE department_id IN (20, 50);
```

USING 절을 포함하는 조인 작성

- 여러 열이 같은 이름을 가지지만 데이터 유형이 일치하지 않을 경우, NATURAL JOIN 절을 수정하여 USING 절을 포함시키면 등가 조인에 사용될 열을 지정할 수 있습니다.
- USING 절을 사용하면 둘 이상의 열이 일치할 때 한 열만 일치시킬 수 있습니다.
- 참조되는 열에 테이블 이름이나 별칭을 사용해서는 안됩니다.
- NATURAL JOIN 및 USING 절은 서로 배타적입니다.

ORACLE

4-26

Copyright © Oracle Corporation, 2001. All rights reserved.

USING 절

자연 조인은 이름과 데이터 유형이 일치하는 모든 열을 사용하여 테이블을 조인합니다. USING 절을 사용하면 등가 조인에 사용될 열만을 지정할 수 있습니다. USING 절에서 참조되는 열은 SQL 문 어디에서도 수식자(테이블 이름 또는 별칭)를 가져서는 안됩니다.

예를 들어, 다음 명령문은 유효합니다.

```
SELECT l.city, d.department_name
  FROM locations l JOIN departments d USING (location_id)
 WHERE location_id = 1400;
```

다음 명령문은 WHERE 절에서 LOCATION_ID가 자세히 지정되었기 때문에 유효하지 않습니다.

```
SELECT l.city, d.department_name
  FROM locations l JOIN departments d USING (location_id)
 WHERE d.location_id = 1400;
ORA-25154: column part of USING clause cannot have qualifier
```

동일한 제약이 자연 조인에도 적용됩니다. 따라서 두 테이블 모두에서 동일한 이름을 가진 열은 수식자 없이 사용되어야 합니다.

USING 절로 레코드 검색

```
SELECT e.employee_id, e.last_name, d.location_id  
FROM employees e JOIN departments d  
USING (department_id);
```

EMPLOYEE_ID	LAST_NAME	LOCATION_ID
200	Whalen	1700
201	Hartstein	1800
202	Fay	1800
124	Mourgos	1500
141	Rajs	1500
142	Davies	1500
143	Matos	1500
144	Vargas	1500
103	Hunold	1400
...		

19 rows selected.

ORACLE

4-27

Copyright © Oracle Corporation, 2001. All rights reserved.

USING 절(계속)

예제는 EMPLOYEES 및 DEPARTMENTS 테이블의 DEPARTMENT_ID 열을 조인하여 사원이 일하는 위치를 표시합니다.

이 예제는 다음과 같이 등가 조인으로도 작성할 수 있습니다.

```
SELECT employee_id, last_name,  
       employees.department_id, location_id  
  FROM employees, departments  
 WHERE employees.department_id = departments.department_id;
```

ON 절로 조인 작성

- 자연 조인의 조인 조건은 기본적으로 같은 이름을 가진 모든 열의 등가 조인입니다.
- 임의의 조건을 지정하거나 조인할 열을 지정하려면 ON 절을 사용합니다.
- 조인 조건이 다른 검색 조건과 분리됩니다.
- ON 절을 사용하면 코드가 이해하기 쉬워집니다.

ORACLE

4-28

Copyright © Oracle Corporation, 2001. All rights reserved.

ON 조건

조인 조건 지정에 ON 절을 사용하면 WHERE 절에서 조인 조건을 다른 검색 또는 필터 조건과 분리하여 지정할 수 있습니다.

ON 절로 레코드 검색

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
  FROM employees e JOIN departments d
 WHERE (e.department_id = d.department_id);
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500

19 rows selected.

ORACLE

4-29

Copyright © Oracle Corporation, 2001. All rights reserved.

ON 절로 조인 작성

ON 절은 다음과 같이 서로 다른 이름을 가진 열을 조인하는 데에도 사용할 수 있습니다.

```
SELECT e.last_name emp, m.last_name mgr
  FROM employees e JOIN employees m
 WHERE (e.manager_id = m.employee_id);
```

EMP	MGR
Kochhar	King
De Haan	King
Mourgos	King
Zlotkey	King
Hartstein	King
Whalen	Kochhar

19 rows selected.

앞의 예제는 EMPLOYEE_ID 및 MANAGER_ID 열을 기준으로 하는 EMPLOYEES 테이블의 자체 조인입니다.

ON 절로 3-way 조인 작성

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id;
```

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
103	Southlake	IT
104	Southlake	IT
107	Southlake	IT
124	South San Francisco	Shipping
141	South San Francisco	Shipping
142	South San Francisco	Shipping
143	South San Francisco	Shipping
144	South San Francisco	Shipping

19 rows selected.

ORACLE

3-Way 조인

3-way 조인은 세 테이블 조인입니다. SQL 1999 표준 구문에서, 조인은 왼쪽에서 오른쪽으로 수행되므로 수행될 첫번째 조인은 EMPLOYEES JOIN DEPARTMENTS입니다. 첫번째 조인 조건은 EMPLOYEES 및 DEPARTMENTS의 열을 참조할 수 있지만 LOCATIONS의 열은 참조할 수 없습니다. 두번째 조인 조건은 세 테이블 모두의 열을 참조할 수 있습니다.

이 예제는 다음과 같이 3-way 등가 조인으로도 작성할 수 있습니다.

```
SELECT employee_id, city, department_name
FROM   employees, departments, locations
WHERE  employees.department_id = departments.department_id
AND    departments.location_id = locations.location_id;
```

내부 조인(inner join)과 포괄 조인(outer join) 비교

- **SQL: 1999**에서 두 테이블을 조인해서 일치하는 열만 반환하는 조인이 내부 조인입니다.
- 두 테이블을 조인해서 내부 조인의 결과와 함께 일치하지 않는 원쪽(또는 오른쪽) 테이블의 행을 반환하는 조인이 원쪽(또는 오른쪽) 포괄 조인입니다.
- 두 테이블을 조인해서 내부 조인의 결과와 함께 원쪽 및 오른쪽 조인의 결과를 반환하는 조인이 전체 포괄 조인입니다.

ORACLE

조인 - SQL: 1999와 오라클 구문 비교

오라클	SQL: 1999
등가 조인	자연/내부 조인
포괄 조인	원쪽 포괄 조인
자체 조인	Join ON
비등가 조인	Join USING
카티시안 곱(Cartesian Product)	교차 조인

LEFT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name  
FROM employees e  
LEFT OUTER JOIN departments d  
ON (e.department_id = d.department_id);
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing

De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		

20 rows selected.

ORACLE

LEFT OUTER JOIN 예제

이 질의는 DEPARTMENTS 테이블에 일치하는 행이 없어도 왼쪽 테이블인 EMPLOYEES 테이블의 모든 행을 검색합니다.

이 질의는 이전 릴리스에서는 다음과 같이 작성되었습니다.

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

RIGHT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name  
FROM employees e  
RIGHT OUTER JOIN departments d  
ON (e.department_id = d.department_id);
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
King	90	Executive
Kochhar	90	Executive

Whaler	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Higgins	110	Accounting
Gietz	110	Accounting
		Contracting

20 rows selected.

ORACLE

4-33

Copyright © Oracle Corporation, 2001. All rights reserved.

RIGHT OUTER JOIN 예제

이 질의는 EMPLOYEES 테이블에 일치하는 행이 없어도 오른쪽 테이블인 DEPARTMENTS 테이블의 모든 행을 검색합니다.

이 질의는 이전 릴리스에서 다음과 같이 작성되었습니다.

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

FULL OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name  
FROM employees e  
FULL OUTER JOIN departments d  
ON (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
...		
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		Contracting

21 rows selected.

ORACLE

FULL OUTER JOIN 예제

이 질의는 DEPARTMENTS 테이블에 일치하는 행이 없어도 EMPLOYEES 테이블의 모든 행을 검색합니다. 또한 EMPLOYEES 테이블에 일치하는 행이 없어도 DEPARTMENTS 테이블의 모든 행을 검색합니다.

추가 조건

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
  FROM employees e JOIN departments d
 WHERE (e.department_id = d.department_id)
   AND e.manager_id = 149 ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
174	Abel	80	80	2500
176	Taylor	80	80	2500

ORACLE

추가 조건 적용

WHERE 절에 추가 조건을 적용할 수 있습니다. 예제는 EMPLOYEES 및 DEPARTMENTS 테이블에 대해 조인을 수행하고 관리자 ID가 149인 사원만 표시합니다.

요약

이 단원에서는 다음 구문의 조인을 사용하여 여러 테이블의 데이터를 표시하는 방법을 설명했습니다.

- 8*i*를 포함한 이전 버전에 사용된 오라클 전용 구문
- 9*i* 버전에 사용된 SQL: 1999 표준 구문

ORACLE®

4-36

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

테이블 조인에는 다음과 같은 여러 방법이 있습니다.

조인 유형

- 등가 조인
- 비등가 조인
- 포괄 조인
- 자체 조인
- 교차 조인
- 자연 조인
- 전체 또는 포괄 조인

카티시안 곱(Cartesian Product)

카티시안 곱이 수행되면 행의 모든 조합이 표시됩니다. WHERE 절을 생략하거나 CROSS JOIN 절을 지정하면 카티시안 곱이 수행됩니다.

테이블 별칭

- 테이블 별칭은 데이터베이스 액세스 속도를 높입니다.
- 테이블 별칭을 사용하면 SQL 코드를 작게 작성해도 되므로 메모리 사용이 줄어듭니다.

연습 4, 2부: 개요

이 연습에서는 다음 내용을 다룹니다.

- 등가 조인을 사용한 테이블 조인
- 포괄 조인 및 자체 조인 수행
- 조건 추가

ORACLE

4-37

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 4, 2부

이 연습은 여러 테이블에서 데이터를 추출하는 실제적인 연습 문제를 제공합니다. 오라클 전용 구문과 SQL: 1999 표준 구문 모두를 사용해 보십시오.

2부의 문제 5-8에서는 ANSI 구문을 사용하여 조인 문을 작성해 보십시오.

2부의 문제 9-11에서는 오라클 구문 및 ANSI 구문 모두를 사용하여 조인 문을 작성해 보십시오.

연습 4-1부

1. 모든 사원의 이름, 부서 번호, 부서 이름을 표시하는 질의를 작성하십시오.

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping
Rajs	50	Shipping
Davies	50	Shipping
Matos	50	Shipping
Vargas	50	Shipping
Hunold	60	IT
Ernst	60	IT
Lorentz	60	IT
Zlotkey	80	Sales
Abel	80	Sales

19 rows selected.

2. 부서 80에 속하는 모든 업무의 고유 목록을 작성하고 출력 결과에 부서의 위치를 포함시키십시오.

JOB_ID	LOCATION_ID
SA_MAN	2500
SA_REP	2500

3. 커미션을 받는 모든 사원의 이름, 부서 이름, 위치 ID 및 도시를 표시하는 질의를 작성하십시오.

LAST_NAME	DEPARTMENT_NAME	LOCATION_ID	CITY
Zlotkey	Sales	2500	Oxford
Abel	Sales	2500	Oxford
Taylor	Sales	2500	Oxford

연습 4 - 1부(계속)

4. 이름에 *a*(소문자)가 포함된 모든 사원의 이름과 부서 이름을 표시하고 해당 SQL 문을 lab4_4.sql 파일에 저장하십시오.

LAST_NAME	DEPARTMENT_NAME
Whalen	Administration
Hartstein	Marketing
Fay	Marketing
Rajs	Shipping
Davies	Shipping
Matos	Shipping
Vargas	Shipping
Taylor	Sales
Kochhar	Executive
De Haan	Executive

10 rows selected.

연습 4 - 2부

5. Toronto에서 근무하는 모든 사원의 이름, 업무, 부서 번호 및 부서 이름을 표시하는 질의를 작성하십시오.

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

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.

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

19 rows selected.

연습 4 - 2부(계속)

7. lab4_6.sql을 수정하여 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.

Employee	EMP#	Manager	Mgr#
King	100		
Kochhar	101	King	100
De Haan	102	King	100
Hunold	103	De Haan	102
Ernst	104	Hunold	103
Lorentz	107	Hunold	103
Mourgos	124	King	100

20 rows selected.

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.

DEPARTMENT	EMPLOYEE	COLLEAGUE
20	Fay	Hartstein
20	Hartstein	Fay
50	Davies	Matos
50	Davies	Mourgos
50	Davies	Rajs
50	Davies	Vargas
50	Matos	Davies
50	Matos	Mourgos
50	Matos	Rajs
50	Matos	Vargas
50	Mourgos	Davies
50	Mourgos	Matos
50	Mourgos	Rajs
50	Mourgos	Vargas

42 rows selected.

연습 4 - 2부(계속)

9. JOB_GRADES 테이블의 구조를 표시하고 모든 사원의 이름, 업무, 부서 이름, 급여 및 등급을 표시하는 질의를 작성하십시오.

Name	Null?	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRA
Matos	ST_CLERK	Shipping	2600	A
Vargas	ST_CLERK	Shipping	2500	A
Lorentz	IT_PROG	IT	4200	B
Mourgos	ST_MAN	Shipping	5800	B
Rajs	ST_CLERK	Shipping	3500	B
Davies	ST_CLERK	Shipping	3100	B
Whalen	AD_ASST	Administration	4400	B

19 rows selected.

좀더 연습하려면 다음 문제를 풀어보십시오.

10. Davies라는 사원보다 늦게 입사한 사원의 이름과 입사일을 표시하는 질의를 작성하십시오.

LAST_NAME	HIRE_DATE
Lorentz	07-FEB-99
Mourgos	16-NOV-99
Matos	15-MAR-98
Vargas	09-JUL-98
Zlotkey	29-JAN-00
Taylor	24-MAR-98
Grant	24-MAY-99
Fay	17-AUG-97

8 rows selected.

연습 4 - 2부(계속)

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.

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

9 rows selected.

5

그룹 함수를 사용한
데이터 질계

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 사용 가능한 그룹 함수 식별
- 그룹 함수 사용 설명
- GROUP BY 절을 사용한 데이터 그룹화
- HAVING 절을 사용하여 그룹화된 행 포함 또는 제외

ORACLE

5-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 함수에 대해 자세히 설명합니다. 행 그룹의 평균 등과 같은 요약 정보를 얻는 방법을 중점적으로 설명하며 테이블 행 그룹화 방법 및 행 그룹에 대한 검색 조건 지정 방법을 설명합니다.

그룹 함수란?

그룹 함수는 행 집합에 작용하여 그룹 당 하나의 결과를 생성합니다.

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3600
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
80	7000
10	4400

20 rows selected.

ORACLE

5-3

Copyright © Oracle Corporation, 2001. All rights reserved.

그룹 함수

단일 행 함수와 달리 그룹 함수는 행 집합에 작용하여 그룹 당 하나의 결과를 생성합니다. 이 때 집합은 전체 테이블 또는 그룹화된 테이블입니다.

그룹 함수 종류

- **AVG**
- **COUNT**
- **MAX**
- **MIN**
- **STDDEV**
- **SUM**
- **VARIANCE**

ORACLE

5-4

Copyright © Oracle Corporation, 2001. All rights reserved.

그룹 함수(계속)

각 함수는 인수를 사용합니다. 다음 표는 구문에서 사용 가능한 옵션을 설명합니다.

함수	설명
AVG ([DISTINCT ALL] <i>n</i>)	<i>n</i> 의 평균 값이며 널 값은 무시합니다.
COUNT ({ * [DISTINCT ALL] <i>expr</i> })	<i>expr</i> 이 널이 아닌 행의 수입니다. *를 사용하면 중복 행 및 널이 있는 행을 포함하여 선택한 모든 행을 셉니다.
MAX ([DISTINCT ALL] <i>expr</i>)	<i>expr</i> 의 최대값이며 널 값은 무시합니다.
MIN ([DISTINCT ALL] <i>expr</i>)	<i>expr</i> 의 최소값이며 널 값은 무시합니다.
STDDEV ([DISTINCT ALL] <i>n</i>)	<i>n</i> 의 표준 편차이며 널 값은 무시합니다.
SUM ([DISTINCT ALL] <i>n</i>)	<i>n</i> 의 합계이며 널 값은 무시합니다.
VARIANCE ([DISTINCT ALL] <i>n</i>)	<i>n</i> 의 분산이며 널 값은 무시합니다.

그룹 함수 구문

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

ORACLE

5-5

Copyright © Oracle Corporation, 2001. All rights reserved.

그룹 함수 사용 지침

- DISTINCT를 지정하면 함수는 중복되지 않는 값만 고려하고 ALL을 지정하면 중복 값을 포함한 모든 값을 고려합니다. 기본값은 ALL이므로 지정할 필요가 없습니다.
- expr 인수를 가진 함수의 데이터 유형은 CHAR, VARCHAR2, NUMBER 또는 DATE가 될 수 있습니다.
- 모든 그룹 함수는 널 값을 무시합니다. 널 값을 특정 값으로 치환하려면 NVL, NVL2 또는 COALESCE 함수를 사용합니다.
- GROUP BY 절을 사용하면 Oracle server가 결과 집합을 암시적(implicit)으로 오름차순으로 정렬합니다. 이러한 기본 순서를 무시하고 내림차순으로 정렬하려면 ORDER BY 절에 DESC를 사용하십시오.

AVG 및 SUM 함수 사용

숫자 데이터에 AVG 및 SUM을 사용할 수 있습니다.

```
SELECT AVG(salary), MAX(salary),  
       MIN(salary), SUM(salary)  
  FROM employees  
 WHERE job_id LIKE '%REP%';
```

Avg(Salary)	Max(Salary)	Min(Salary)	Sum(Salary)
8150	11000	6000	32600

ORACLE

그룹 함수

숫자 데이터를 저장하는 열에 대해 AVG, SUM, MIN 및 MAX 함수를 사용할 수 있습니다. 슬라이드 예제는 모든 영업 사원에 대한 월급 평균액, 최고액, 최저액 및 총액을 표시합니다.

MIN 및 MAX 함수 사용

모든 데이터 유형에 대해 MIN 및 MAX를 사용할 수 있습니다.

```
SELECT MIN(hire_date), MAX(hire_date)  
FROM employees;
```

MIN(HIRE_DATE)	MAX(HIRE_DATE)
17-JUN-87	29-JAN-00

ORACLE

5-7

Copyright © Oracle Corporation, 2001. All rights reserved.

그룹 함수(계속)

모든 데이터 유형에 대해 MAX 및 MIN 함수를 사용할 수 있습니다. 슬라이드 예제는 최근에 입사한 사원과 가장 오래전에 입사한 사원을 표시합니다.

다음 예제는 모든 사원 이름을 영문자순으로 나열했을 때 맨 처음 및 맨 마지막에 해당하는 사원 이름을 표시합니다.

```
SELECT MIN(last_name), MAX(last_name)  
FROM employees;
```

MIN(LAST_NAME)	MAX(LAST_NAME)
Abel	Zlotkey

참고: AVG, SUM, VARIANCE 및 STDDEV 함수는 숫자 데이터 유형에만 사용할 수 있습니다.

COUNT 함수 사용

COUNT (*)는 테이블의 행 수를 반환합니다.

```
SELECT COUNT(*)  
  FROM employees  
 WHERE department_id = 50;
```

COUNT(*)
5

ORACLE

COUNT 함수

COUNT 함수에는 다음 세 가지 형식이 있습니다.

- COUNT (*)
- COUNT (expr)
- COUNT (DISTINCT expr)

COUNT (*)는 열에 널 값이 있는 행 및 중복 행을 포함하여 테이블에서 SELECT 문의 조건을 만족하는 행 수를 반환합니다. SELECT 문에 WHERE 절이 포함된 경우 COUNT (*)는 WHERE 절의 조건을 만족하는 행 수를 반환합니다.

반대로 COUNT (expr)은 expr로 식별되는 열에서 널이 아닌 행 수를 반환합니다.

COUNT (DISTINCT expr)은 expr로 식별되는 열에서 중복되지 않는 널이 아닌 값의 수를 반환합니다.

슬라이드 예제는 부서 50의 사원 수를 표시합니다.

COUNT 함수 사용

- COUNT(expr)은 expr에 대해 널이 아닌 값을 가진 행 수를 반환합니다.
- EMPLOYEES 테이블에서 널 값을 제외한 부서 값의 수를 표시합니다.

```
SELECT COUNT(commission_pct)
  FROM employees
 WHERE department_id = 80;
```

COUNT(COMMISSION_PCT)

3

ORACLE

5-9

Copyright © Oracle Corporation, 2001. All rights reserved.

COUNT 함수(계속)

슬라이드 예제는 부서 80에서 커미션을 받는 사원의 수를 표시합니다.

예제

EMPLOYEES 테이블에서 부서 값의 수를 표시합니다.

```
SELECT COUNT(department_id)
  FROM employees;
```

COUNT(DEPARTMENT_ID)

19

DISTINCT 키워드 사용

- COUNT(DISTINCT expr)은 expr에 대해 중복되지 않는 널이 아닌 값의 수를 반환합니다.
- EMPLOYEES 테이블에서 중복되지 않는 부서 값의 수를 표시합니다.

```
SELECT COUNT(DISTINCT department_id)  
FROM employees;
```

```
COUNT(DISTINCTDEPARTMENT_ID)
```

```
7
```

ORACLE

DISTINCT 키워드

DISTINCT 키워드를 사용하면 열에서 중복되는 값을 세지 않도록 할 수 있습니다.

슬라이드 예제는 EMPLOYEES 테이블에서 중복되지 않는 부서 값의 수를 표시합니다.

그룹 함수 및 널 값

그룹 함수는 해당 열의 널 값을 무시합니다.

```
SELECT AVG(commission_pct)  
  FROM employees;
```

Avg(COMMISSION_PCT)
2125

ORACLE

그룹 함수 및 널 값

모든 그룹 함수는 해당 열의 널 값을 무시합니다. 슬라이드 예제에서 평균은 테이블의 COMMISSION_PCT 열에 유효한 값이 저장된 행만으로 계산되며, 모든 사원이 받는 커미션 총액을 커미션을 받는 사원 수(4)로 나눈 값입니다.

그룹 함수에 NVL 함수 사용

NVL 함수는 그룹 함수가 널 값을 포함하도록 강제로 지정합니다.

```
SELECT AVG(NVL(commission_pct, 0))  
FROM employees;
```

Avg(NVL(COMMISSION_PCT,0))

0425

ORACLE

5-12

Copyright © Oracle Corporation, 2001. All rights reserved.

그룹 함수 및 널 값(계속)

NVL 함수는 그룹 함수가 널 값을 포함하도록 강제로 지정합니다. 슬라이드 예제에서 평균은 COMMISSION_PCT 열의 널 값 저장 여부에 관계 없이 테이블의 모든 행을 기반으로 계산되며, 모든 사원이 받는 커미션 총액을 회사의 전체 사원 수(20)로 나눈 값입니다.

데이터 그룹 생성

EMPLOYEES

DEPARTMENT_ID	SALARY
10	4400
20	13000
20	6000
50	5800
50	3500
50	3100
50	2500
50	2600
60	9000
60	6000
60	4200
80	10500
80	8600
80	11000
90	24000
90	17000

...
20 rows selected.

DEPARTMENT_ID

AVG(SALARY)

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10160
	7000

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

데이터 그룹

지금까지 모든 그룹 함수는 테이블을 하나의 대형 정보 그룹으로 취급했습니다. 테이블 정보를 더 작은 그룹으로 나누어야 할 경우 GROUP BY 절을 사용하면 됩니다.

데이터 그룹 생성: GROUP BY 절 구문

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

GROUP BY 절을 사용하여 테이블 행을 더 작은 그룹으로 나눕니다.

ORACLE

5-14

Copyright © Oracle Corporation, 2001. All rights reserved.

GROUP BY 절

GROUP BY 절을 사용하여 테이블 행을 그룹으로 나눈 후 그룹 함수를 사용하여 각 그룹에 대한 요약 정보를 반환할 수 있습니다.

구문 설명:

group_by_expression 행 그룹화 기준을 결정하는 값을 가진 열을 지정 합니다.

지침

- SELECT 절에 그룹 함수를 포함시킨 경우 GROUP BY 절에 개별 열을 지정하지 않으면 개별 결과를 선택할 수 없으며, GROUP BY 절에 열 목록을 포함시키지 않으면 오류 메시지가 나타납니다.
- WHERE 절을 사용하면 그룹으로 나누기 전에 행을 제외시킬 수 있습니다.
- GROUP BY 절에 열을 포함시켜야 합니다.
- GROUP BY 절에는 열 별칭을 사용할 수 없습니다.
- 기본적으로 행은 GROUP BY 목록에 포함된 열의 오름차순으로 정렬되는데 ORDER BY 절을 사용하면 이 순서를 무시할 수 있습니다.

GROUP BY 절 사용

SELECT 목록의 열 중 그룹 함수에 없는 열은 모두 GROUP BY 절에 포함되어야 합니다.

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

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
30	3500
40	6400
50	10033.3333
60	19333.3333
70	10150
80	7000
90	
110	

8 rows selected.

ORACLE

5-15

Copyright © Oracle Corporation, 2001. All rights reserved.

GROUP BY 절(계속)

GROUP BY 절을 사용할 때 SELECT 목록의 열 중 그룹 함수가 아닌 열은 모두 GROUP BY 절에 포함되어야 합니다. 슬라이드 예제는 각 부서의 부서 번호 및 평균 급여를 표시합니다. GROUP BY 절을 포함하는 SELECT 문은 다음 방식으로 평가됩니다.

- SELECT 절은 검색할 열을 지정합니다.
 - EMPLOYEES 테이블의 부서 번호 열
 - GROUP BY 절에서 지정한 그룹 내의 모든 급여에 대한 평균
- FROM 절은 데이터베이스가 액세스할 테이블(EMPLOYEES 테이블)을 지정합니다.
- WHERE 절은 검색할 행을 지정하는데 예제에는 WHERE 절이 없으므로 기본적으로 모든 행을 검색합니다.
- GROUP BY 절은 행 그룹화 방식을 지정합니다. 행은 부서 번호에 따라 그룹화되므로 급여 열에 적용되는 AVG 함수는 각 부서의 평균 급여를 계산합니다.

GROUP BY 절 사용

GROUP BY 열을 SELECT 목록에 포함시키지 않아도 됩니다.

```
SELECT      AVG(salary)
FROM        employees
GROUP BY    department_id ;
```

	AVG(SALARY)
	4400
	9500
	3500
	6400
	10033.3333
	19333.3333
	10150
	7000

ORACLE

5-16

Copyright © Oracle Corporation, 2001. All rights reserved.

GROUP BY 절(계속)

GROUP BY 열을 SELECT 절에 포함시키지 않아도 됩니다. 예를 들어, 슬라이드의 SELECT 문은 해당 부서 번호는 표시하지 않고 각 부서의 평균 급여만 표시합니다. 그러나 부서 번호가 없으면 그 결과는 의미가 없습니다.

ORDER BY 절에 그룹 함수를 사용할 수 있습니다.

```
SELECT      department_id, AVG(salary)
FROM        employees
GROUP BY    department_id
ORDER BY    AVG(salary);
```

DEPARTMENT_ID	AVG(SALARY)
	3500
	4400
	6400
	19333.3333
90	

8 rows selected.

여러 열을 기준으로 그룹화

EMPLOYEES

DEPARTMENT_ID	JOB_ID	SALARY
90	AD PRES	24000
90	AD VP	17000
90	AD VP	17000
60	IT PROG	9000
60	IT PROG	6000
60	IT PROG	4200
50	ST MAN	5800
50	ST CLERK	3500
50	ST CLERK	3100
50	ST CLERK	2600
50	ST CLERK	2500
80	SA MAN	10500
80	SA REP	11000
80	SA REP	8600

20	MK REP	6000
110	AC MGR	12000
110	AC ACCOUNT	6800

20 rows selected.

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK MAN	13000
20	MK REP	6000
50	ST CLERK	11700
50	ST MAN	5800
60	IT PROG	19200
80	SA MAN	10600
80	SA REP	19600
90	AD PRES	24000
90	AD VP	34000
110	AC ACCOUNT	6800
110	AC MGR	12000
	SA REP	7000

13 rows selected.

ORACLE

그룹 내 그룹

그룹 내 그룹에 대한 결과를 확인해야 할 경우가 있습니다. 슬라이드는 각 부서 내에서 각 업무에 대해 지급되는 급여 총액을 표시하는 보고서를 보여줍니다.

EMPLOYEES 테이블은 먼저 부서 번호에 따라 그룹화된 후 그룹 내에서 다시 업무에 따라 그룹화됩니다. 예를 들어, 부서 50에서 일하는 네 명의 사무원(stock clerk)이 하나의 그룹으로 묶이고 이 그룹 내 모든 사무원에 대해 하나의 결과(총급여)가 생성됩니다.

여러 열에 GROUP BY 절 사용

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

DEPT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

ORACLE

그룹 내 그룹(계속)

하나 이상의 GROUP BY 열을 나열하여 그룹 및 하위 그룹에 대한 요약 결과를 반환할 수 있으며, GROUP BY 절에서의 열 순서에 따라 결과의 기본 정렬 순서를 결정할 수 있습니다. GROUP BY 절을 포함하는 슬라이드의 SELECT 문은 다음과 같이 평가됩니다.

- SELECT 절은 검색할 열을 지정합니다.
 - EMPLOYEES 테이블의 부서 번호
 - EMPLOYEES 테이블의 업무 ID
 - GROUP BY 절에서 지정한 그룹 내의 모든 급여 합계
- FROM 절은 데이터베이스가 액세스할 테이블(EMPLOYEES 테이블)을 지정합니다.
- GROUP BY 절은 행 그룹화 방식을 지정합니다.
 - 먼저 부서 번호를 기준으로 행을 그룹화합니다.
 - 그런 다음 해당 부서 번호 그룹 내에서 업무 ID를 기준으로 다시 그룹화합니다.

따라서 SUM 함수는 각 부서 번호 그룹 내에 있는 모든 업무 ID의 급여 열에 적용됩니다.

그룹 함수를 사용한 잘못된 질의

SELECT 목록의 열 또는 표현식 중 그룹 함수가 아닌 것은 GROUP BY 절에 포함시켜야 합니다.

```
SELECT department_id, COUNT(last_name)
FROM   employees;
```

```
SELECT department_id, COUNT(last_name)
*
ERROR at line 1:
ORA-00937: not a single-group group function
```

GROUP BY 절에 열이 없습니다.

ORACLE

5-19

Copyright © Oracle Corporation, 2001. All rights reserved.

그룹 함수를 사용한 잘못된 질의

개별 항목(DEPARTMENT_ID) 및 그룹 함수(COUNT)를 동일한 SELECT 문에 함께 사용할 때는 개별 항목(DEPARTMENT_ID)을 지정하는 GROUP BY 절을 항상 포함시켜야 합니다. GROUP BY 절이 없으면 “not a single-group group function”이라는 오류 메시지가 나타나며 잘못된 열에 별표(*)가 표시됩니다. GROUP BY 절을 추가하여 슬라이드에 나타난 오류를 수정할 수 있습니다.

```
SELECT    department_id, count(last_name)
FROM      employees
GROUP BY department_id;
```

DEPARTMENT_ID	COUNT(LAST_NAME)
10	1
20	2
...	
	1

8 rows selected.

SELECT 목록의 열 또는 표현식 중 집계 함수가 아닌 것은 GROUP BY 절에 포함시켜야 합니다.

그룹 함수를 사용한 잘못된 질의

- WHERE 절을 사용하여 그룹을 제한할 수 없습니다.
- HAVING 절을 사용하여 그룹을 제한할 수 있습니다.
- WHERE 절에서 그룹 함수를 사용할 수 없습니다.

```
SELECT department_id, AVG(salary)
FROM employees
WHERE AVG(salary) > 8000
GROUP BY department_id;
```

```
WHERE AVG(salary) > 8000
*
ERROR at line 3:
ORA-00934: group function is not allowed here
```

WHERE 절을 사용하여 그룹을 제한할 수 없습니다.

ORACLE

5-20

Copyright © Oracle Corporation, 2001. All rights reserved.

그룹 함수를 사용한 잘못된 질의(계속)

WHERE 절은 그룹을 제한하는 데 사용할 수 없습니다. 슬라이드의 SELECT 문은 WHERE 절을 사용하여 평균 급여가 \$8,000를 넘는 부서의 평균 급여를 표시하도록 제한하므로 오류가 발생합니다.

HAVING 절로 그룹을 제한하여 슬라이드 오류를 수정할 수 있습니다.

```
SELECT department_id, AVG(salary)
FROM employees
HAVING AVG(salary) > 8000
```

DEPARTMENT_ID	AVG(SALARY)
20	9500
80	10033.3333
90	19333.3333
110	10150

그룹 결과 제외

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	5000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	9500
...	
20	6000
110	12000
110	8300

20 rows selected.

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

ORACLE

그룹 결과 제한

WHERE 절을 사용하여 선택할 행을 제한하는 것과 동일한 방식으로 HAVING 절을 사용하여 그룹을 제한합니다. 각 부서의 최고 급여를 찾아 최고 급여가 \$10,000를 넘는 부서만 표시하려면 다음을 수행합니다.

1. 부서 번호를 기준으로 그룹화한 각 부서의 평균 급여를 찾습니다.
2. 최고 급여가 \$10,000를 넘는 부서로 그룹을 제한합니다.

그룹 결과 제외: HAVING 절

HAVING 절을 사용하여 그룹을 제한합니다.

1. 행이 그룹화됩니다.
2. 그룹 함수가 적용됩니다.
3. HAVING 절과 일치하는 그룹이 표시됩니다.

```
SELECT      column, group_function  
FROM        table  
[WHERE      condition]  
[GROUP BY   group_by_expression]  
[HAVING    group_condition]  
[ORDER BY   column];
```

ORACLE

5-22

Copyright © Oracle Corporation, 2001. All rights reserved.

HAVING 절

HAVING 절을 사용하여 표시할 그룹을 지정할 수 있으므로 집계 정보를 기준으로 그룹 제한을 강화합니다.

구문 설명:

group_condition 반환되는 행 그룹을 지정한 조건이 TRUE인 그룹으로 제한합니다.

HAVING 절을 사용하면 Oracle server가 다음 단계를 수행합니다.

1. 행이 그룹화됩니다.
2. 그룹 함수가 그룹에 적용됩니다.
3. HAVING 절의 조건에 일치하는 그룹이 표시됩니다.

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

HAVING 절 사용

```
SELECT department_id, MAX(salary)
FROM employees
GROUP BY department_id
HAVING MAX(salary) > 10000;
```

DEPARTMENT_ID	MAX(SALARY)
20	13000
60	11000
90	24000
110	12000

ORACLE

5-23

Copyright © Oracle Corporation, 2001. All rights reserved.

HAVING 절(계속)

슬라이드 예제는 최고 급여가 \$10,000를 넘는 부서의 부서 번호 및 최고 급여를 표시합니다.

SELECT 문에 그룹 함수를 사용하지 않고도 GROUP BY 절을 사용할 수 있습니다.

그룹 함수의 결과를 기반으로 행을 제한할 경우에는 GROUP BY 절 및 HAVING 절이 모두 있어야 합니다.

다음 예제는 최고 급여가 \$10,000를 넘는 부서의 부서 번호와 평균 급여를 표시합니다.

```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id
HAVING max(salary) > 10000;
```

DEPARTMENT_ID	AVG(SALARY)
20	9500
80	10033.3333
90	19333.3333
110	10150

HAVING 절 사용

```
SELECT      job_id, SUM(salary) PAYROLL
FROM        employees
WHERE       job_id NOT LIKE '%REP%'
GROUP BY   job_id
HAVING     SUM(salary) > 13000
ORDER BY   SUM(salary);
```

JOB_ID	PAYROLL
IT_PROG	19200
AD_PRES	24000
AD_VP	34000

ORACLE

HAVING 절(계속)

슬라이드 예제는 월급 총액이 \$13,000를 넘는 각 업무에 대해 업무 ID와 월급 총액을 표시하되, 영업 사원을 제외시킨 후 월급 총액에 따라 목록을 정렬합니다.

그룹 함수 중첩

최고 평균 급여를 표시합니다.

```
SELECT MAX(AVG(salary))
  FROM employees
 GROUP BY department_id;
```

MAX(AVG(SALARY))

19333.3333

ORACLE

S-25

Copyright © Oracle Corporation, 2001. All rights reserved.

그룹 함수 중첩

그룹 함수는 두 번까지 중첩될 수 있습니다. 슬라이드 예제는 최고 평균 급여를 표시합니다.

요약

이 단원에서는 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- 그룹 함수 COUNT, MAX, MIN, AVG 사용
- GROUP BY 절을 사용하는 질의 작성
- HAVING 절을 사용하는 질의 작성

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

ORACLE

5-26

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

SQL에서는 일곱 가지 그룹 함수를 사용할 수 있습니다.

- AVG
- COUNT
- MAX
- MIN
- SUM
- STDDEV
- VARIANCE

GROUP BY 절을 사용하여 하위 그룹을 생성하고 HAVING 절을 사용하여 그룹을 제외시킬 수 있습니다.

HAVING 및 GROUP BY 절은 명령문에서 WHERE 절 다음에 넣고 ORDER BY 절은 맨 마지막에 넣습니다.

Oracle server는 절을 다음 순서로 평가합니다.

1. 명령문이 WHERE 절을 포함하는 경우 서버가 후보 행을 설정합니다.
2. 서버가 GROUP BY 절에 지정된 그룹을 식별합니다.
3. HAVING 절은 HAVING 절의 그룹 조건을 만족하지 않는 결과 그룹을 제한합니다.

연습 5 개요

이 연습에서는 다음 내용을 다룹니다.

- 그룹 함수를 사용하는 질의 작성
- 여러 결과를 얻을 수 있도록 행 그룹화
- HAVING 절을 사용하여 그룹 제외

연습 5 개요

이 연습을 마치면 그룹 함수 사용 및 데이터 그룹 선택에 능숙해집니다.

문제

문제 1-3은 True 또는 False로 답하십시오.

참고: 질의에는 열 별칭이 사용됩니다.

연습 5

다음 세 문장의 유효성을 판별하여 True 또는 False로 답하십시오.

- 그룹 함수는 여러 행에 적용되어 그룹 당 하나의 결과를 출력합니다.

True/False

- 그룹 함수는 계산에 널을 포함합니다.

True/False

- WHERE 절은 그룹 계산에 행을 포함시키기 전에 행을 제한합니다.

True/False

- 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.

Maximum	Minimum	Sum	Average
24000	2500	175500	8775

- 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.

JOB_ID	Maximum	Minimum	Sum	Average
AC_ACCOUNT	8300	8300	8300	8300
AC_MGR	12000	12000	12000	12000
AD_ASST	4400	4400	4400	4400
AD_PRES	24000	24000	24000	24000
AD_VP	17000	17000	34000	17000
IT_PROG	9000	4200	19200	6400
MK_MAN	13000	13000	13000	13000
MK_REP	6000	6000	6000	6000
SA_MAN	10500	10500	10500	10500
SA_REP	11000	7000	26600	8867
ST_CLERK	3500	2500	11700	2925
ST_MAN	5800	5800	5800	5800

12 rows selected.

연습 5(계속)

6. 업무가 동일한 사원 수를 표시하는 질의를 작성하십시오.

JOB_ID	COUNT(*)
AC_ACCOUNT	1
AC_MGR	1
AD_ASST	1
AD_PRES	1
AD_VP	2
IT_PROG	3
MK_MAN	1
MK_REP	1
SA_MAN	1
SA_REP	3
ST_CLERK	4
ST_MAN	1

12 rows selected.

7. 관리자는 나열하지 말고 관리자 수를 확인하십시오. 열 레이블은 Number of Managers로 지정하십시오. 힌트: MANAGER_ID 열을 사용하여 관리자 수를 확인 하십시오.

Number of Managers
8

8. 최고 급여와 최저 급여의 차액을 표시하는 질의를 작성하고 열 레이블을 DIFFERENCE로 지정하십시오.

DIFFERENCE
21500

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 내림 차순으로 정렬하십시오.

MANAGER_ID	MIN(SALARY)
102	9000
205	8300
149	7000

연습 5(계속)

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.

Name	Location	Number of People	Salary
Accounting	1700	2	10150
Administration	1700	1	4400
Executive	1700	3	19333.33
IT	1400	3	6400
Marketing	1800	2	9500
Sales	2500	3	10033.33
Shipping	1500	5	3500

7 rows selected.

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.

TOTAL	1995	1996	1997	1998
20	1	2	2	3

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.

Job	Dept 20	Dept 50	Dept 80	Dept 90	Total
AC_ACCOUNT					8300
AC_MGR					12000
AD_ASST					4400
AD_PRES				24000	24000
AD_VP				34000	34000
IT_PROG					19200
MK_MAN	13000				13000
MK_REP	6000				6000
SA_MAN			10500		10500
SA_REP			19600		26600
ST_CLERK		11700			11700
ST_MAN		5800			5800

12 rows selected.

6

서브 쿼리

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 서브 쿼리가 해결할 수 있는 문제 유형 설명
- 서브 쿼리 정의
- 서브 쿼리 유형 나열
- 단일 행 서브 쿼리 및 다중 행 서브 쿼리 작성

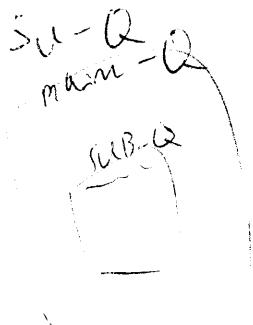
ORACLE

6-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 SELECT 문의 고급 기능을 설명합니다. 다른 SQL 문의 WHERE 절에 서브 쿼리를 작성하여 알 수 없는 조건 값을 기반으로 값을 얻을 수 있습니다. 이 단원에서는 단일 행 서브 쿼리 및 다중 행 서브 쿼리를 설명합니다.



문제 해결에 서브 쿼리 사용

Abel보다 급여가 많은 사원은 누구입니까?

메인쿼리



어느 사원의 급여가 Abel의 급여보다 많습니까?

서브 쿼리



Abel의 급여는 얼마입니까?

문제 해결에 서브 쿼리 사용

Abel보다 급여가 많은 사원을 찾는 질의를 작성한다고 가정합니다.

이 문제를 해결하려면 Abel의 급여를 찾는 첫번째 질의와 해당 급여보다 급여가 많은 사원을 찾는 두번째 질의 즉, 두 개의 질의가 필요합니다.

이 문제는 두 질의를 결합하여 한 질의를 다른 질의 내부에 포함시키면 해결됩니다.

내부 질의(서브 쿼리)는 외부 질의(메인쿼리)에 사용될 값을 반환합니다. 서브 쿼리 사용은 두 개의 순차적인 질의를 수행하여 첫번째 질의 결과를 두번째 질의의 검색 값으로 사용하는 것과 동일합니다.

서브 쿼리 구문

```
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 메인쿼리(외부 질의)에 사용 됩니다.

ORACLE

서브 쿼리

서브 쿼리는 다른 SELECT 문의 절에 삽입된 SELECT 문으로서 서브 쿼리를 사용하면 간단한 명령문으로 강력한 기능을 제공하는 명령문을 작성할 수 있습니다. 서브 쿼리는 테이블 자체의 데이터에 종속된 조건을 사용해 테이블에서 행을 선택할 때 유용합니다.

다음과 같은 여러 SQL 절에 서브 쿼리를 포함시킬 수 있습니다.

- WHERE 절
- HAVING 절
- FROM 절

구문 설명:

operator >, =, IN 등 비교 조건을 포함합니다.

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

서브 쿼리는 중첩 SELECT 문, 하위 SELECT 문 또는 내부 SELECT 문이라고도 합니다. 일반적으로 서브 쿼리가 먼저 실행되고 그 출력 결과를 사용하여 메인쿼리(외부 질의)에 대한 질의 조건을 완성합니다.

서브 쿼리 사용

```
SELECT last_name
  FROM employees 11000
 WHERE salary >
        (SELECT salary
          FROM employees
         WHERE last_name = 'Abel');
```

LAST_NAME
King
Kochhar
De Haan
Hartstein
Higgins

ORACLE

서브 쿼리 사용

슬라이드에서 내부 질의는 사원 Abel의 급여를 확인하고 외부 질의는 내부 질의의 결과를 사용하여 해당 급여보다 급여가 많은 사원을 모두 표시합니다.

서브 쿼리 사용 지침

- 서브 쿼리를 괄호로 묶습니다.
- 비교 조건의 오른쪽에 서브 쿼리를 넣습니다.
- 서브 쿼리의 ORDER BY 절은 Top-N 분석을 수행하지 않을 경우에는 필요가 없습니다.
- 단일 행 서브 쿼리에는 단일 행 연산자를 사용하고 다중 행 서브 쿼리에는 다중 행 연산자를 사용합니다.

ORACLE

서브 쿼리 사용 지침

- 서브 쿼리는 괄호로 묶어야 합니다.
- 읽기 쉽도록 비교 조건의 오른쪽에 서브 쿼리를 넣습니다.
- Oracle8i 릴리스 이전에는 서브 쿼리에 ORDER BY 절을 포함할 수 없었습니다. ORDER BY 절은 SELECT 문에 한번만 사용할 수 있었으며, 지정할 경우 기본 SELECT 문의 마지막 절이어야 했습니다. Oracle8i 릴리스부터는 Top-N 분석을 수행하는 데 필요하므로 ORDER BY 절을 서브 쿼리에 사용할 수 있게 되었습니다.
- 서브 쿼리에 사용되는 비교 조건은 단일 행 연산자 및 다중 행 연산자로 분류됩니다.

서브 쿼리 유형

- 단일 행 서브 쿼리



- 다중 행 서브 쿼리



서브 쿼리 유형

- 단일 행 서브 쿼리: 내부 SELECT 문에서 한 행만 반환하는 질의
- 다중 행 서브 쿼리: 내부 SELECT 문에서 여러 행을 반환하는 질의

참고: 다중 열 서브 쿼리: 내부 SELECT 문에서 여러 열을 반환하는 질의

단일 행 서브 쿼리

- 한 행만 반환합니다.
- 단일 행 비교 연산자를 사용합니다.

연산자	의미
=	같음
>	보다 큼
>=	크거나 같음
<	보다 작음
<=	작거나 같음
<>	같지 않음

ORACLE

6-8

Copyright © Oracle Corporation, 2001. All rights reserved.

단일 행 서브 쿼리

단일 행 서브 쿼리는 내부 SELECT 문에서 하나의 행을 반환하는 서브 쿼리로서 단일 행 연산자를 사용합니다. 슬라이드는 단일 행 연산자 목록을 보여줍니다.

예제

사원 141과 동일한 업무 ID를 가진 사원을 표시합니다.

```
SELECT last_name, job_id
  FROM employees
 WHERE job_id =
       (SELECT job_id
        FROM   employees
        WHERE  employee_id = 141);
```

LAST_NAME	JOB_ID
Rajs	ST_CLERK
Davies	ST_CLERK
Matos	ST_CLERK
Vargas	ST_CLERK

단일 행 서브 쿼리 실행

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = ST_CLERK
       (SELECT job_id
        FROM   employees
        WHERE employee_id = 141)
AND    salary > 2600
       (SELECT salary
        FROM   employees
        WHERE employee_id = 143);
```

LAST_NAME	JOB_ID	SALARY
Rajs	ST_CLERK	3500
Davies	ST_CLERK	3100

ORACLE

단일 행 서브 쿼리 실행

SELECT 문을 질의 블록으로 간주할 수 있습니다. 슬라이드 예제는 사원 141과 업무 ID가 동일하면서 사원 143보다 급여가 많은 사원을 표시합니다.

이 예제는 세 개의 질의(외부 질의의 한 개와 내부 질의 두 개) 블록으로 구성됩니다. 내부 질의 블록이 먼저 실행되어 각각의 질의 결과(ST_CLERK, 2600)를 생성합니다. 그런 다음 외부 질의 블록이 처리되면서 내부 질의에서 반환된 값을 사용하여 검색 조건을 완성합니다.

두 개의 내부 질의는 모두 하나의 값(ST_CLERK 및 2600)을 반환하므로 이 SQL 문을 단일 행 서브 쿼리라고 합니다.

참고: 내부 질의 및 외부 질의는 서로 다른 테이블의 데이터를 가져올 수 있습니다.

서브 쿼리에서 그룹 함수 사용

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =  
       (SELECT MIN(salary)
        FROM   employees);
```

LAST_NAME	JOB_ID	SALARY
Vargas	ST_CLERK	2500

ORACLE

서브 쿼리에서 그룹 함수 사용

단일 행을 반환하는 그룹 함수를 서브 쿼리에서 사용하여 기본 질의에서 데이터를 표시할 수 있습니다. 서브 쿼리는 괄호로 묶어야 하며 비교 조건 다음에 놓입니다.

슬라이드 예제는 최소 급여를 받는 모든 사원의 이름, 업무 ID 및 급여를 표시하며 MIN 그룹 함수는 단일 값(2500)을 외부 질의에 반환합니다.

HAVING 절에 서브 쿼리 사용

- Oracle server는 서브 쿼리를 먼저 실행합니다.
- Oracle server는 메인쿼리의 HAVING 절에 결과를 반환합니다.

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary) > 2500
      (SELECT MIN(salary)
       FROM employees
       WHERE department_id = 50);
```

6-11

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

HAVING 절에 서브 쿼리 사용

WHERE 절뿐만 아니라 HAVING 절에도 서브 쿼리를 사용할 수 있습니다. Oracle server는 서브 쿼리를 실행하여 결과를 메인쿼리의 HAVING 절에 반환합니다.

슬라이드의 SQL 문은 최소 급여가 부서 50의 최소 급여보다 많은 부서를 모두 표시합니다.

DEPARTMENT_ID	MIN(SALARY)
10	4400
20	6000
	7000

7 rows selected.

예제

평균 급여가 가장 적은 업무를 찾습니다.

```
SELECT job_id, AVG(salary)
FROM employees
GROUP BY job_id
HAVING AVG(salary) = (SELECT MIN(AVG(salary))
                      FROM employees
                      GROUP BY job_id);
```

이 명령문은 무엇이 잘못되었을까요?

```
SELECT employee_id, last_name  
FROM employees  
WHERE salary =  
      (SELECT MIN(salary)  
       FROM employees  
       GROUP BY department_id);
```

```
ERROR at line 4:  
ORA-01427: single-row subquery returns more than  
one row
```

다중 행 서브 쿼리에 단일 행 연산자를 사용했습니다.

ORACLE

6-12

Copyright © Oracle Corporation, 2001. All rights reserved.

서브 쿼리 오류

서브 쿼리의 일반적인 오류는 단일 행 서브 쿼리에 대해 여러 행이 반환되는 것입니다.

슬라이드의 SQL 문에서 서브 쿼리는 GROUP BY 절을 포함하는데 이는 서브 쿼리가 그룹 당 하나씩 여러 행을 반환함을 의미합니다. 이 예제에서 서브 쿼리의 결과는 4400, 6000, 2500, 4200, 7000, 17000, 8300입니다.

외부 질의는 서브 쿼리의 결과(4400, 6000, 2500, 4200, 7000, 17000, 8300)를 받아 WHERE 절에서 사용합니다. WHERE 절은 하나의 값만 처리하는 단일 행 비교 연산자인 등호(=) 연산자를 포함하는데 =연산자가 서브 쿼리로부터 여러 값을 받아 들일 수 없으므로 오류가 발생합니다.

이 문제를 수정하려면 =연산자를 IN으로 바꾸십시오.

이 명령문은 행을 반환할까요?

```
SELECT last_name, job_id  
FROM   employees  
WHERE  job_id =  
       (SELECT job_id  
        FROM   employees  
        WHERE  last_name = 'Haas');
```

```
no rows selected
```

서브 쿼리가 값을 반환하지 않습니다.

ORACLE

6-13

Copyright © Oracle Corporation, 2001. All rights reserved.

서브 쿼리에서 발생할 수 있는 문제

서브 쿼리에서 발생할 수 있는 일반적인 문제는 내부 질의에 서 행을 반환하지 않는 경우입니다.

슬라이드의 SQL 문에서 서브 쿼리는 이름이 Haas인 사원을 찾는 WHERE 절을 포함합니다. 이 명령문은 유효하지만 실행했을 때 선택되는 행이 없습니다.

이름이 Haas라는 사원이 없으므로 서브 쿼리에서 행을 반환하지 않습니다. 외부 질의는 서브 쿼리의 결과(널)를 받아 WHERE 절에서 사용합니다. 외부 질의는 업무 ID가 널인 사원을 찾지 못합니다. 널 값을 갖는 업무가 존재하더라도 널 값 두 개를 비교하면 널이 반환되므로 행이 반환되지 않습니다. 따라서 WHERE 조건이 True가 되지 않습니다.

다중 행 서브 쿼리

- 여러 행을 반환합니다.
- 여러 행 비교 연산자를 사용합니다.

연산자	의미
IN	목록에 있는 임의의 멤버와 동일합니다.
ANY	값을 서브 쿼리에 의해 반환된 각 값과 비교합니다.
ALL	값을 서브 쿼리에 의해 반환된 모든 값과 비교합니다.

ORACLE®

6-14

Copyright © Oracle Corporation, 2001. All rights reserved.

여러 행 서브 쿼리

여러 행을 반환하는 서브 쿼리를 여러 행 서브 쿼리라고 하는데 여러 행 서브 쿼리에서는 단일 행 연산자 대신 여러 값을 처리하는 여러 행 연산자를 사용합니다.

```
SELECT last_name, salary, department_id
  FROM employees
 WHERE salary IN (SELECT MIN(salary)
                   FROM employees
                   GROUP BY department_id);
```

예제

각 부서에서 최소 급여를 받는 사원을 찾습니다.

내부 질의가 먼저 실행되어 질의 결과를 생성합니다. 그런 다음 기본 질의 블록이 처리되면서 내부 질의에 의해 반환된 값을 사용해 검색 조건을 완성합니다. 실제로 Oracle server는 기본 질의를 다음과 같이 인식합니다.

```
SELECT last_name, salary, department_id
  FROM employees
 WHERE salary IN (2500, 4200, 4400, 6000, 7000, 8300, 8600, 17000);
```

다중 행 서브 쿼리에 ANY 연산자 사용

```
SELECT employee_id, last_name, job_id, salary
FROM   employees      3000, 6000, 4200
WHERE  salary < ANY
       (SELECT salary
        FROM   employees
        WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
124	Mourgos	ST_MAN	5800
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

10 rows selected.

ORACLE

여러 행 서브 쿼리(계속)

ANY 연산자(및 동의어인 SOME 연산자)는 값을 서브 쿼리가 반환하는 각각의 값과 비교합니다.

슬라이드 예제는 IT 프로그래머가 아니면서 급여가 임의의 IT 프로그래머보다 낮은 사원을 표시합니다. 프로그래머의 최고 급여는 \$9,000입니다.

<ANY는 최대값보다 작음을 나타내고, >ANY는 최소값보다 큼을 나타내며, =ANY는 IN과 동일합니다.

다중 행 서브 쿼리에 ALL 연산자 사용

```
SELECT employee_id, last_name, job_id, salary
  FROM employees
 WHERE salary < ALL
          8000, 5000, 4200
          (SELECT salary
             FROM employees
            WHERE job_id = 'IT_PROG')
AND      job_id <> 'IT_PROG';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

ORACLE

여러 행 서브 쿼리(계속)

ALL 연산자는 값을 서브 쿼리에서 반환하는 모든 값과 비교합니다. 슬라이드 예제는 업무 ID가 IT_PROG인 모든 사원보다 급여가 적으면서 업무가 IT_PROG가 아닌 사원을 표시합니다.

>ALL은 최대값보다 큼을 나타내고, <ALL은 최소값보다 작음을 나타냅니다.

NOT 연산자는 IN, ANY 및 ALL 연산자와 함께 사용할 수 있습니다.

서브 쿼리에서의 널 값

```
SELECT emp.last_name
  FROM employees emp
 WHERE emp.employee_id NOT IN
          (SELECT mgr.manager_id
            FROM   employees mgr);
no rows selected
```

ORACLE

6-17

Copyright © Oracle Corporation, 2001. All rights reserved.

서브 쿼리의 결과 집합에 널 반환

슬라이드의 SQL 문은 부하 직원이 없는 모든 사원을 표시합니다. 논리적으로는 이 SQL 문이 행을 12개 반환해야 하지만 아무 행도 반환되지 않습니다. 내부 질의에서 반환한 값 중 하나가 널 값이므로 전체 질의가 행을 반환하지 않는 것입니다. 널 값을 비교하는 모든 조건에 대한 결과는 널입니다. 따라서 서브 쿼리의 결과 집합에 널 값이 포함될 가능성이 있는 경우에는 NOT IN 연산자를 사용하지 마십시오. NOT IN 연산자는 <> ALL과 동일합니다.

IN 연산자를 사용할 때는 서브 쿼리 결과 집합에 널 값이 있어도 됩니다. IN 연산자는 =ANY와 동일합니다. 예를 들어, 부하 직원이 있는 모든 사원을 표시하려면 다음 SQL 문을 사용하십시오.

```
SELECT emp.last_name
  FROM employees emp
 WHERE emp.employee_id IN
          (SELECT mgr.manager_id
            FROM   employees mgr);
```

또는, 서브 쿼리에 다음과 같은 WHERE 절을 포함시켜 부하 직원이 없는 모든 사원을 표시할 수 있습니다.

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

요약

이 단원에서는 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- 서브 쿼리로 문제를 해결할 수 있는 경우 식별
- 질의가 알 수 없는 값을 기반으로 할 경우 서브 쿼리 작성

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

ORACLE

요약

이 단원에서는 서브 쿼리를 사용하는 방법을 설명했습니다. 서브 쿼리는 다른 SQL 문의 절에 삽입되는 SELECT 문으로서 중간 값을 알 수 없는 검색 조건을 기반으로 하는 질의에 유용하게 사용됩니다.

서브 쿼리의 특성은 다음과 같습니다.

- 하나의 행으로 이루어진 데이터를 =, <>, >, >=, <, <=와 같은 단일 행 연산자를 포함하는 기본 명령문에 전달할 수 있습니다.
- 여러 행 데이터를 IN과 같이 다중 행 연산자를 포함하는 기본 명령문에 전달할 수 있습니다.
- Oracle server에 의해 우선적으로 처리되며 그 결과는 WHERE 절 또는 HAVING 절에서 사용됩니다.
- 그룹 함수를 포함할 수 있습니다.

연습 6 개요

이 연습에서는 다음 내용을 다릅니다.

- 알 수 없는 조건을 기반으로 하는 값을 질의하는 서브 쿼리 작성
- 서브 쿼리를 사용하여 한 데이터 집합에는 존재하지만 다른 집합에는 존재하지 않는 값 찾기

ORACLE

6-19

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 6

이 연습에서는 중첩 SELECT 문을 사용하는 복잡한 질의를 작성해 봅니다.

문제

우선 이들 문제에 대해 내부 질의를 작성하는 것이 좋습니다. 외부 질의를 코딩하기 전에 내부 질의를 실행하고 예상 데이터를 생성하도록 하십시오.

연습 6

- Zlotkey와 동일한 부서에 속한 모든 사원의 이름과 입사일을 표시하는 질의를 작성하십시오. Zlotkey는 제외하십시오.

LAST_NAME	HIRE_DATE
Abel	11-MAY-96
Taylor	24-MAR-98

- 급여가 평균 급여보다 많은 모든 사원의 사원 번호와 이름을 표시하는 질의를 작성하고 결과를 급여에 대해 오름차순으로 정렬하십시오.

EMPLOYEE_ID	LAST_NAME	SALARY
103	Hunold	9000
149	Zlotkey	10500
174	Abel	11000
205	Higgins	12000
201	Hartstein	13000
101	Kochhar	17000
102	De Haan	17000
100	King	24000

8 rows selected.

- 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 실행하십시오.

EMPLOYEE_ID	LAST_NAME
124	Mourgos
141	Rajs
142	Davies
143	Matos
144	Vargas
103	Hunold
104	Ernst
107	Lorentz

8 rows selected.

연습 6(계속)

4. 부서 위치 ID가 1700인 모든 사원의 이름, 부서 번호 및 업무 ID를 표시하십시오.

LAST_NAME	DEPARTMENT_ID	JOB_ID
Whalen		10 AD_ASST
King		90 AD_PRES
Kochhar		90 AD_VP
De Haan		90 AD_VP
Higgins		110 AC_MGR
Gietz		110 AC_ACCOUNT

6 rows selected.

5. King에게 보고하는 모든 사원의 이름과 급여를 표시하십시오.

LAST_NAME	SALARY
Kochhar	17000
De Haan	17000
Mourgos	5800
Zlotkey	10500
Hartstein	13000

6. Executive 부서의 모든 사원에 대한 부서 번호, 이름 및 업무 ID를 표시하십시오.

DEPARTMENT_ID	LAST_NAME	JOB_ID
90	King	AD_PRES
90	Kochhar	AD_VP
90	De Haan	AD_VP

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.

EMPLOYEE_ID	LAST_NAME	SALARY
103	Hunold	9000



**iSQL*Plus로 일기 쉬운
출력 작성**

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 치환 변수가 필요한 질의 작성
- *SQL*Plus* 환경 사용자 정의
- 읽기 쉬운 결과 출력
- 스크립트 파일 작성 및 실행

ORACLE

7-2

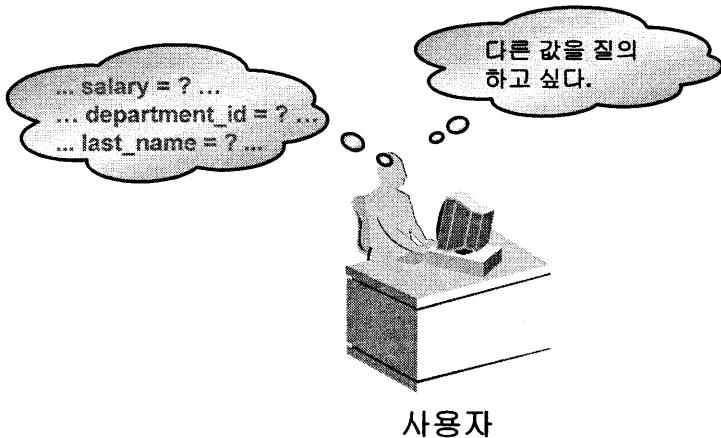
Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 *SQL*Plus* 명령을 포함하여 SQL 결과를 읽기 쉽게 출력하는 방법을 설명합니다.

WHERE 절을 포함하는 명령 파일을 생성하여 표시할 행 수를 제한할 수 있습니다. 명령 파일이 실행될 때마다 조건을 변경하려면 치환 변수를 사용하십시오. 치환 변수는 WHERE 절의 값, 텍스트 문자열 및 열 이름이나 테이블 이름까지도 바꿀 수 있습니다.

치환 변수



7-3

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

치환 변수

지금까지의 예제는 하드 코딩된 것이었습니다. 완성된 응용 프로그램에서 사용자가 보고서를 트리거하면 더 이상 프롬프트가 나타나지 않고 보고서가 실행되었습니다. 데이터 범위는 iSQL*Plus 스크립트 파일에 있는 고정된 WHERE 절에 의해 미리 정의된 것이었습니다.

iSQL*Plus에서 치환 변수를 사용하면, 반환되는 데이터의 범위를 제한하는 값을 사용자가 지정하도록 요구하는 보고서를 작성할 수 있습니다. 치환 변수는 명령 파일이나 단일 SQL 문에 포함시킬 수 있습니다. 변수는 값을 임시로 저장해 두는 컨테이너라고 생각하면 됩니다. 명령문을 실행하면 값이 치환됩니다.

치환 변수

iSQL*Plus 치환 변수를 사용하여 다음을 수행합니다.

- 값을 임시로 저장합니다.
 - 단일 앤퍼샌드(&)
 - 이중 앤퍼샌드(&&)
 - **DEFINE** 명령
- SQL 문 간에 변수 값을 전달합니다.
- 머리글과 바닥글을 동적으로 변경합니다.

ORACLE

7-4

Copyright © Oracle Corporation, 2001. All rights reserved.

치환 변수

iSQL*Plus에서는 단일 앤퍼샌드(&) 치환 변수를 사용하여 값을 임시로 저장할 수 있습니다.

iSQL*Plus에서는 DEFINE 명령을 사용하여 변수를 미리 정의할 수 있습니다. DEFINE은 값을 생성하여 변수에 할당합니다.

제한된 데이터 범위 예

- 현재 분기 또는 지정된 날짜 범위에 대한 수치 보고
- 보고서를 요청한 사용자와 관련된 데이터만 보고
- 지정된 부서의 사원만 표시

기타 대화식 효과

대화식 효과는 사용자가 직접 WHERE 절과 상호 작용할 수 있다는 것만으로 제한되지 않습니다. 이와 동일한 원칙을 사용하여 다른 목적을 이를 수 있습니다. 예를 들면 다음과 같습니다.

- 머리글과 바닥글 동적으로 변경
- 사용자가 값을 입력하는 대신 파일에서 입력 값 가져오기
- 한 SQL 문에서 다른 SQL 문으로 값 전달

iSQL*Plus는 사용자 입력에 대해 유효성 검사를 수행하지 않습니다(단, 데이터 유형은 예외).

& 치환 변수 사용

앰퍼샌드(&)가 접두어로 붙은 변수를 사용하여 사용자에게 값을 물습니다.

```
SELECT employee_id, last_name, salary, department_id  
FROM employees  
WHERE employee_id = &employee_num;
```

The screenshot shows the iSQL*Plus interface. At the top, there are navigation icons for Password, Log Out, and Help. Below that is a toolbar with 'Define Substitution Variables'. A text input field contains '&employee_num'. At the bottom are 'Submit for Execution' and 'Cancel' buttons. The Oracle logo is at the bottom right. The footer displays 'Copyright © Oracle Corporation, 2001. All rights reserved.'

단일 앰퍼샌드 치환 변수

보고서를 실행하는 동안 반환되는 데이터를 동적으로 제한하고 싶을 경우가 있습니다. iSQL*Plus는 사용자 변수를 통해 이러한 융통성을 제공합니다. SQL 문의 각 변수는 앰퍼샌드(&)를 사용하여 식별하며, 각 변수의 값을 정의할 필요는 없습니다.

표기법	설명
<code>&user_variable</code>	SQL 문의 변수를 나타냅니다. 해당 변수가 존재하지 않을 경우 iSQL*Plus에서는 사용자에게 값을 요구합니다. iSQL*Plus에서는 새 변수를 한 번 사용한 다음 버립니다.

슬라이드 예제는 사원 번호에 대해 iSQL*Plus 치환 변수를 생성합니다. 이 명령문을 실행하면 iSQL*Plus는 사용자에게 사원 번호를 요구한 다음 해당 사원의 사원 번호, 이름, 급여 및 부서 번호를 표시합니다.

단일 앤퍼샌드를 사용하면 해당 변수가 존재하지 않을 경우 명령이 실행될 때마다 프롬프트가 표시됩니다.

& 치환 변수 사용

The screenshot shows the Oracle iSQL*Plus interface. At the top, it says "Define Substitution Variables" and has a field containing "&employee_num". Below this is a table with four columns: EMPLOYEE_ID, LAST_NAME, SALARY, and DEPARTMENT_ID. A single row is shown with values 101, Kochhar, 17000, and 90 respectively. At the bottom of the screen, there is a copyright notice: "Copyright © Oracle Corporation, 2001. All rights reserved."

ORACLE iSQL*Plus

Define Substitution Variables

employee_num

Submit for Execution Cancel

old 3: WHERE employee_id = &employee_num
new 3: WHERE employee_id = 101

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
101	Kochhar	17000	90

Copyright © Oracle Corporation, 2001. All rights reserved.

단일 앤퍼샌드 치환 변수

iSQL*Plus에서는 &가 포함된 SQL 문을 발견하면 사용자에게 SQL 문의 치환 변수에 대한 값을 입력하도록 요구합니다. 사용자가 값을 입력하고 Submit for Execution 버튼을 누르면 결과가 iSQL*Plus 세션의 출력 영역에 표시됩니다.

치환 변수를 사용한 문자 및 날짜 값

날짜 및 문자 값에는 작은 따옴표를 사용합니다.

```
SELECT last_name, department_id, salary*12
FROM   employees
WHERE  job_id = '&job_title' ;
```

Define Substitution Variables

job_title IT_PROG

Submit for Execution

Cancle

LAST_NAME	DEPARTMENT_ID	SALARY*12
Hunold	60	108000
Ernst	60	72000
Lorentz	60	50400

ORACLE

7-7

Copyright © Oracle Corporation, 2001. All rights reserved.

치환 변수를 사용하여 문자 및 날짜 값 지정

WHERE 절에서 날짜 및 문자 값은 작은 따옴표로 묶어야 하는데 치환 변수에도 동일한 규칙이 적용됩니다.

변수도 SQL 문 안에서 작은 따옴표로 묶습니다.

슬라이드는 iSQL*Plus 치환 변수의 업무 값에 따라 모든 사원의 이름, 부서 번호 및 연봉을 검색하는 질의를 보여줍니다.

참고: UPPER 및 LOWER와 같은 함수도 앤페 샌드와 함께 사용할 수 있습니다.

UPPER('&job_title')를 사용하면 사용자가 업무를 대문자로 입력하지 않아도 됩니다.

열 이름, 표현식 및 텍스트 지정

다음 항목에 치환 변수를 사용합니다.

- WHERE 조건
- ORDER BY 절
- 열 표현식
- 테이블 이름
- 전체 SELECT 문

7-8

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

열 이름, 표현식 및 텍스트 지정

치환 변수는 SQL 문의 WHERE 절에 사용할 수 있을 뿐 아니라 열 이름, 표현식 또는 텍스트 치환에도 사용할 수도 있습니다.

예제

사원의 사원 번호, 다른 열 및 조건을 표시합니다.

```
SELECT employee_id, &column_name
FROM   employees
WHERE  &condition;
```

"column_name" job_id

"condition" department_id = 10

EMPLOYEE_ID	JOB_ID
200	AD_ASST

치환 변수 값을 입력하지 않으면 앞에 나온 명령문을 실행할 경우 오류가 발생합니다.

참고: 명령 프롬프트에서 첫번째 단어로 입력되는 경우를 제외하면 SELECT 문의 어디서나 치환 변수를 사용할 수 있습니다.

열 이름, 표현식 및 텍스트 지정

```
SELECT      employee_id, last_name, job_id,  
           &column_name  
FROM        employees  
WHERE       &condition  
ORDER BY    &order_column ;
```

Define Substitution Variables

*column_name salary
*condition salary > 15000
*order_column last_name

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
102	De Haan	AD_VP	17000
100	King	AD_PRES	24000
101	Kochhar	AD_VP	17000

ORACLE

7-9

Copyright © Oracle Corporation, 2001. All rights reserved.

열 이름, 표현식 및 텍스트 지정(계속)

슬라이드 예제는 실행 중에 사용자가 지정한 사원 번호, 이름, 업무 및 기타 열을 EMPLOYEES 테이블에서 가져와 표시합니다. 행 검색 조건과 결과 데이터의 정렬 기준이 될 열 이름을 지정할 수도 있습니다.

치환 변수 정의

- *iSQL*Plus* **DEFINE** 명령을 사용하여 변수를 미리 정의 할 수 있습니다.

DEFINE variable = value는 CHAR 데이터 유형의 사용자 변수를 생성합니다.

- 공백을 포함하는 변수를 미리 정의하려면 **DEFINE** 명령을 사용할 때 그 값을 작은 따옴표로 묶어야 합니다.
- 정의된 변수는 해당 세션에서 사용할 수 있습니다.

ORACLE

7-10

Copyright © Oracle Corporation, 2001. All rights reserved.

치환 변수 정의

SELECT 문을 실행하기 전에 사용자 변수를 미리 정의할 수 있습니다. *iSQL*Plus*는 치환 변수 정의 및 설정에 사용할 수 있는 **DEFINE** 명령을 제공합니다.

명령	설명
DEFINE variable = value	CHAR 데이터 유형의 사용자 변수를 생성하고 값을 할당합니다.
DEFINE variable	해당 변수와 해당 변수의 값 및 데이터 유형을 표시합니다.
DEFINE	모든 사용자 변수를 해당 값 및 데이터 유형과 함께 표시합니다.

DEFINE 및 UNDEFINE 명령

- 정의된 변수는 다음과 같은 경우에 손실됩니다.
 - UNDEFINE 명령을 사용하여 변수를 지운 경우
 - iSQL*Plus를 종료한 경우
- DEFINE 명령으로 변경 내용을 확인할 수 있습니다.

```
DEFINE job_title = IT_PROG
DEFINE job_title
DEFINE JOB_TITLE      = "IT_PROG" (CHAR)
```

```
UNDEFINE job_title
DEFINE job_title
SP2-0135: symbol job_title is UNDEFINED
```

ORACLE

7-11

Copyright © Oracle Corporation, 2001. All rights reserved.

DEFINE 및 UNDEFINE 명령

정의된 변수는 다음과 같은 경우에 손실됩니다.

- 변수에 UNDEFINE 명령을 실행한 경우
- iSQL*Plus를 종료한 경우

변수 정의를 취소한 경우 DEFINE 명령으로 변경 내용을 확인할 수 있습니다. iSQL*Plus를 종료하면 해당 세션에서 정의된 변수가 손실됩니다.

& 치환 변수에 DEFINE 명령 사용

- DEFINE 명령을 사용하여 치환 변수를 생성합니다.

```
DEFINE employee_num = 200
```

- 앤퍼샌드(&)가 접두어로 붙은 변수를 사용하여 SQL 문에서 값을 치환합니다.

```
SELECT employee_id, last_name, salary, department_id  
FROM   employees  
WHERE  employee_id = &employee_num ;
```

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
200	Whalen	4400	10

ORACLE

DEFINE 명령 사용

슬라이드 예제는 DEFINE 명령을 사용하여 사원 번호에 대한 iSQL*Plus 치환 변수를 생성하고 런타임에 해당 사원의 번호, 이름, 급여 및 부서 번호를 표시합니다.

변수가 iSQL*Plus DEFINE 명령을 사용하여 생성되었으므로 사용자에게 사원 번호 값을 입력하도록 요청하는 대신 정의된 변수 값이 자동으로 SELECT 문에서 치환됩니다.

EMPLOYEE_NUM 치환 변수는 사용자가 정의를 취소하거나 iSQL*Plus 세션을 종료할 때까지 존재합니다.

&& 치환 변수 사용

매번 사용자에게 묻지 않고 변수 값을 재사용하려면 이중 앤퍼샌드(&&)를 사용합니다.

```
SELECT employee_id, last_name, job_id, &&column_name  
FROM employees  
ORDER BY &column_name;
```

Define Substitution Variables

"column_name" department_id

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
200	Whalen	AD_ASST	10
201	Hartstein	MK_MAN	20

20 rows selected.

ORACLE

7-13

Copyright © Oracle Corporation, 2001. All rights reserved.

이중 앤퍼샌드 치환 변수

매번 사용자에게 묻지 않고 변수 값을 재사용하려면 이중 앤퍼샌드(&&) 치환 변수를 사용하면 됩니다. 이 치환 변수를 사용하면 사용자에게 해당 값을 묻는 프롬프트가 한 번만 표시됩니다. 슬라이드 예제의 경우 사용자에게 *column_name* 변수의 값을 지정하라는 프롬프트가 한 번만 표시됩니다. 사용자가 제공한 값(department_id)은 데이터 표시 및 데이터 순서 결정에 사용됩니다.

iSQL*Plus는 제공된 값을 DEFINE 명령을 사용하여 저장하며 해당 변수 이름이 참조될 때마다 저장된 값을 다시 사용합니다. 사용자 변수를 지정한 경우에 해당 변수를 삭제하려면 UNDEFINE 명령을 사용해야 합니다.

VERIFY 명령 사용

VERIFY 명령을 사용하면 iSQL*Plus가 치환 변수를 값으로 대체하기 전후의 치환 변수 값을 표시할 수 있습니다.

```
SET VERIFY ON  
SELECT employee_id, last_name, salary, department_id  
FROM   employees  
WHERE  employee_id = &employee_num;
```

employee_num [200]

```
old  3: WHERE employee_id = &employee_num  
new  3: WHERE employee_id = 200
```

ORACLE

7-14

Copyright © Oracle Corporation, 2001. All rights reserved.

VERIFY 명령

SQL 문에서 해당 변수가 치환되었는지 확인하려면 iSQL*Plus VERIFY 명령을 사용합니다. SET VERIFY ON을 설정할 경우 iSQL*Plus에서는 치환 변수를 값으로 바꾸기 전후의 명령 텍스트를 표시합니다.

슬라이드 예제는 EMPLOYEE_ID 열의 이전 값과 새로운 값을 모두 표시합니다.

iSQL*Plus 환경 사용자 정의

- SET 명령을 사용하여 현재 세션을 제어합니다.

```
SET system_variable value
```

- SHOW 명령을 사용하여 설정한 내용을 확인합니다.

```
SET ECHO ON
```

```
SHOW ECHO  
echo ON
```

ORACLE

7-15

Copyright © Oracle Corporation, 2001. All rights reserved.

iSQL*Plus 환경 사용자 정의

SET 명령을 사용하면 iSQL*Plus가 현재 작동 중인 환경을 제어할 수 있습니다.

구문

```
SET system_variable value
```

구문 설명:

 system_variable 세션 환경의 한 측면을 제어하는 변수입니다.

 value 해당 시스템 변수의 값입니다.

SHOW 명령을 사용하여 설정한 내용을 확인할 수 있으며 슬라이드의 SHOW 명령은 ECHO 설정 여부를 확인합니다.

모든 SET 변수 값을 보려면 SHOW ALL 명령을 사용하십시오.

자세한 내용은 *iSQL*Plus User's Guide and Reference*, “Command Reference”를 참조하십시오.

SET 명령 변수

↑ default

- **ARRAYSIZE** {20 | n}
- **FEEDBACK** {6 | n | OFF | ON}
- **HEADING** {OFF | ON}
- **LONG** {80 | n} | ON | text}

SET HEADING OFF

SHOW HEADING
HEADING OFF

ORACLE

7-16

Copyright © Oracle Corporation, 2001. All rights reserved.

SET 명령 변수

SET 변수 및 값	설명
ARRAY [SIZE] { <u>20</u> <u>n</u> }	데이터베이스 인출(Fetch) 크기를 설정합니다.
FEED [BACK] { <u>6</u> <u>n</u> OFF ON}	질의에서 최소 n개의 레코드를 선택할 경우 질의에서 반환 할 레코드 수를 표시합니다.
HEA [DING] {OFF <u>ON</u> }	보고서에 열 머리글을 표시할 것인지 여부를 결정합니다.
LONG { <u>80</u> <u>n</u> }	LONG 값 표시에 사용할 최대 폭(width)을 설정합니다.

참고: n 값은 숫자 값입니다. 밑줄 그어진 값이 기본값이며 변수에 아무 값도 입력하지 않으면 iSQL*Plus는 기본값을 사용합니다.

iSQL*Plus 형식 명령

〈 출력을 조정 명령어 〉

- COLUMN [column option]
- TTITLE [text | OFF | ON]
- BTITLE [text | OFF | ON]
- BREAK [ON report_element]

ORACLE

7-17

Copyright © Oracle Corporation, 2001. All rights reserved.

읽기 쉬운 보고서 얻기

다음 명령을 사용하여 보고서 기능을 제어할 수 있습니다.

명령	설명
COL [UMN] [column option]	열 형식을 제어합니다.
TTI [TLE] [text OFF ON]	보고서의 각 페이지 상단에 표시할 머리글을 지정합니다.
BTI [TLE] [text OFF ON]	보고서의 각 페이지 하단에 표시할 바닥글을 지정합니다.
BRE [AK] [ON report_element]	중복 값을 제거하고 구분 선을 사용하여 데이터 행 (row)을 구분합니다.

지침

- 모든 형식 명령은 해당 iSQL*Plus 세션이 끝날 때까지 또는 형식 설정이 겹쳐쓰여지거나 취소될 때까지 유효합니다.
- 보고서를 생성한 후에는 항상 해당 iSQL*Plus 설정을 기본값으로 재설정해야 합니다.
- iSQL*Plus 변수를 기본값으로 설정하는 명령은 없습니다. 특정 값을 기억해 두거나 로그아웃했다가 다시 로그인해야 합니다.
- 열에 별칭을 부여한 경우에는 열 이름이 아닌 별칭 이름을 참조해야 합니다.

COLUMN 명령

열 표시를 제어합니다.

```
COL [UNN] [{column|alias} [option]]
```

- CLE [AR]: 열 형식을 지웁니다.
- HEA [DING] *text*: 열 머리글을 설정합니다.
- FOR [MAT] *format*: 형식 모델을 사용하여 열 표시를 변경합니다.
- NOPRINT | PRINT
- NULL

ORACLE

7-18

Copyright © Oracle Corporation, 2001. All rights reserved.

COLUMN 명령 옵션

옵션	설명
CLE [AR]	열 형식을 지웁니다.
HEA [DING] <i>text</i>	열 머리글을 설정합니다. 정렬을 사용하지 않는 경우 세로선 ()으로 머리글에서 줄바꿈을 지정합니다.
FOR [MAT] <i>format</i>	열 데이터 표시를 변경합니다.
NOPRI [NT]	열을 숨깁니다.
NUL [L] <i>text</i>	값이 널일 경우 표시될 텍스트를 지정합니다.
PRI [NT]	열을 보여줍니다.

COLUMN 명령 사용

- 열 머리글을 생성합니다.

employee
Name

```
COLUMN last_name HEADING 'Employee|Name'  
COLUMN salary JUSTIFY LEFT FORMAT $99,990.00  
COLUMN manager FORMAT 999999999 NULL 'No manager'
```

- LAST_NAME 열의 현재 설정을 표시합니다.

```
COLUMN last_name
```

- LAST_NAME 열의 설정을 지웁니다.

```
COLUMN last_name CLEAR
```

ORACLE

7-19

Copyright © Oracle Corporation, 2001. All rights reserved.

설정 표시 또는 지우기

COLUMN 명령의 현재 설정을 표시하거나 지우려면 다음 명령을 사용하십시오.

명령	설명
COL [UMN] column	지정된 열의 현재 설정을 표시합니다.
COL [UMN]	모든 열의 현재 설정을 표시합니다.
COL [UMN] column CLE [AR]	지정된 열의 설정을 지웁니다.
CLE [AR] COL [UMN]	모든 열의 설정을 지웁니다.

COLUMN 형식 모델

요소	설명	예제	결과
9	0을 생략한 숫자입니다.	999999	1234
0	맨 앞에 0을 강제로 추가합니다.	099999	001234
\$	부동 달러 기호입니다.	\$9999	\$1234
L	지역 통화입니다.	L9999	L1234
.	소수점의 위치입니다.	9999.99	1234.00
,	천 단위 구분자입니다.	9,999	1,234

ORACLE

7-20

Copyright © Oracle Corporation, 2001. All rights reserved.

COLUMN 형식 모델

슬라이드는 COLUMN 형식 모델의 예제를 표시한 것입니다.

정수가 형식 모델에서 지원하는 자릿수를 초과하는 경우 파운드 기호(#)로 표시됩니다. 형식 모델은 문수자(alphanumeric)인데 실제 값이 숫자인 경우에는 해당 값 대신 파운드 기호가 표시됩니다.

BREAK 명령 사용

BREAK 명령을 사용하여 중복을 제거합니다.

```
BREAK ON job_id
```

ORACLE

7-21

Copyright © Oracle Corporation, 2001. All rights reserved.

BREAK 명령

행을 구분하고 중복 값을 제거하려면 BREAK 명령을 사용합니다. BREAK 명령을 효과적으로 적용하려면 ORDER BY 절을 사용하여 구분할 열의 순서를 지정하십시오.

구문

```
BREAK on column[|alias|row]
```

구문 설명:

column[|alias|row] 주어진 열의 중복 값이 표시되지 않도록 만듭니다.

CLEAR 명령을 사용하여 모든 BREAK 설정을 지웁니다.

```
CLEAR BREAK
```

TTITLE 및 BTITLE 명령 사용

- 머리글과 바닥글을 표시합니다.

TTI [TLE] [text|OFF|ON]

- 보고서 머리글을 설정합니다.

TTITLE 'Salary Report'

- 보고서 바닥글을 설정합니다.

BTITLE 'Confidential'

ORACLE

7-22

Copyright © Oracle Corporation, 2001. All rights reserved.

TTITLE 및 BTITLE 명령

TTITLE 명령을 사용하여 페이지 머리글 형식을 지정하고 BTITLE 명령을 사용하여 바닥글 형식을 지정합니다. 바닥글은 페이지 하단에 나타납니다.

BTITLE 및 TTITLE의 구문은 동일합니다. TTITLE의 구문만 표시되어 있습니다. 세로선(|)을 사용하여 제목의 텍스트를 여러 줄에 나눠 쓸 수 있습니다.

구문

TTI [TLE] | BTI [TLE] | [text|OFF|ON]

구문 설명:

text 제목 텍스트를 나타냅니다. 텍스트가 두 단어 이상이면 작은 따옴표를 사용하십시오.

OFF | ON 제목 표시를 켜거나 끕니다. OFF이면 제목이 표시되지 않습니다.

슬라이드의 TTITLE 예제는 보고서 머리글을 표시할 때 Salary를 한 줄의 가운데에 표시하고 Report는 그 다음 줄의 가운데에 표시하도록 설정하며, BTITLE 예제는 보고서 바닥글에 Confidential을 표시하도록 설정합니다. TTITLE은 보고서에 날짜와 페이지 번호를 자동으로 삽입합니다.

TTITLE 및 BTITLE 명령(계속)

참고: 슬라이드는 TTITLE 및 BTITLE의 구문을 축약하여 나타낸 것입니다. TTITLE 및 BTITLE의 다양한 옵션은 다른 SQL 과정에서 설명합니다.

스크립트 파일을 작성하여 보고서 실행

1. SQL SELECT 문을 작성하고 테스트합니다.
2. SELECT 문을 스크립트 파일에 저장합니다.
3. 스크립트 파일을 편집기로 읽어 들입니다.
4. SELECT 문 앞에 형식 명령을 추가합니다.
5. SELECT 문 다음에 종료 문자가 오는지 확인합니다.

ORACLE

7-24

Copyright © Oracle Corporation, 2001. All rights reserved.

스크립트 파일을 작성하여 보고서 실행

SQL 프롬프트에 각 iSQL*Plus 명령을 입력할 수도 있고 SELECT 문을 포함한 모든 명령을 명령(또는 스크립트) 파일에 포함시킬 수도 있습니다. 기본 스크립트는 최소한 한 개의 SELECT 문과 여러 개의 iSQL*Plus 명령으로 구성됩니다.

스크립트 파일 작성 방법

1. SQL 프롬프트에서 SQL SELECT 문을 작성합니다. 명령문을 파일에 저장하여 형식 명령을 적용하기 전에 보고서를 작성하는 데 필요한 데이터가 정확한 것인지 확인합니다. BREAK를 사용하려면 관련 ORDER BY 절을 포함시켜야 합니다.
2. SELECT 문을 스크립트 파일에 저장합니다.
3. 스크립트 파일을 편집하여 iSQL*Plus 명령을 입력합니다.
4. SELECT 문 앞에 필요한 형식 명령을 추가합니다. SELECT 문 안에 iSQL*Plus 명령을 포함시키지 않도록 합니다.
5. SELECT 문 뒤에 실행 문자인 세미콜론(;) 또는 슬래시(/)가 있는지 확인합니다.

스크립트 파일을 작성하여 보고서 실행

6. **SELECT** 문 다음에 있는 형식 명령을 지웁니다.
7. 스크립트 파일을 저장합니다.
8. 스크립트 파일을 **iSQL*Plus** 텍스트 창으로 로드하고 **Execute** 버튼을 누릅니다.

7-25

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

스크립트 파일 작성 방법(계속)

6. 실행 문자 다음에 **iSQL*Plus** 형식 지우기 명령을 추가합니다. 모든 형식 지우기 명령을 재설정 파일에 저장할 수도 있습니다.
7. 변경한 스크립트 파일을 저장합니다.
8. 스크립트 파일을 **iSQL*Plus** 텍스트 창으로 로드하고 **Execute** 버튼을 누릅니다.

지침

- 스크립트의 **iSQL*Plus** 명령들 사이에 빈 줄을 포함시킬 수 있습니다.
- **iSQL*Plus** 또는 **SQL*Plus** 명령이 길면 현재 줄 끝에 하이픈(-)을 넣어 다음 줄에 연결되도록 할 수 있습니다.
- **iSQL*Plus** 명령을 단축시킬 수 있습니다.
- 파일의 끝에 재설정 명령을 포함하여 원래 **iSQL*Plus** 환경을 복원해야 합니다.

참고: **REM**은 **iSQL*Plus**에서의 주 또는 주석을 나타냅니다.

예제 보고서

Fri Sep 28

Employee
Report

page 1

Job Category	Employee	Salary
AC_ACCOUNT	Gietz	\$8,500.00
AC_MGR	Higgins	\$12,000.00
AD_ASST	Whalen	\$4,400.00
IT_PROG	Ernst	\$6,000.00
	Hunold	\$9,000.00
	Lorentz	\$4,200.00
MK_MAN	Hartstein	\$13,000.00
MK_REP	Fay	\$6,000.00
SA_MAN	Zlotkey	\$10,500.00
SA_REP	Abel	\$11,000.00
	Grant	\$7,000.00
	Taylor	\$8,600.00

Confidential

ORACLE

7-26

Copyright © Oracle Corporation, 2001. All rights reserved.

예제

스크립트 파일을 생성하여 급여가 \$15,000 미만인 모든 사원의 업무 ID, 이름 및 급여를 표시하는 보고서를 작성합니다. 두 줄로 구성된 “Employee Report”라는 머리글과 “Confidential”이라는 바닥글을 가운데 정렬하여 추가합니다. 업무 열 이름을 “Job Category”로 바꿔 두 줄로 나누어 표시하고 사원 이름 열의 이름을 “Employee”로, 그리고 급여 열의 이름을 “Salary”로 바꾸고 형식을 \$2,500.00으로 지정합니다.

예제(계속)

```
SET FEEDBACK OFF
TTITLE 'Employee|Report'
BTITLE 'Confidential'
BREAK ON job_id
COLUMN job_id    HEADING 'Job|Category'
COLUMN last_name HEADING 'Employee'
COLUMN salary     HEADING 'Salary' FORMAT $99,999.99
REM ** Insert SELECT statement
SELECT    job_id, last_name, salary
FROM      employees
WHERE     salary < 15000
ORDER BY job_id, last_name
/ = ;
REM clear all formatting commands ...
SET FEEDBACK ON
COLUMN job_id CLEAR
COLUMN last_name CLEAR
COLUMN salary CLEAR
CLEAR BREAK
...
```

요약

이 단원에서는 다음과 같은 작업을 수행하는 방법에 대해 배웠습니다.

- *iSQL*Plus* 치환 변수를 사용하여 값을 임시로 저장
- SET 명령을 사용하여 현재 *iSQL*Plus* 환경 제어
- COLUMN 명령을 사용하여 열 표시 제어
- BREAK 명령을 사용하여 중복을 제거하고 행을 구분
- TTITLE 및 BTITLE 명령을 사용하여 머리글과 바닥글 표시

ORACLE

7-28

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

이 단원에서는 치환 변수 및 보고서 실행에서 치환 변수의 유용성에 대해 설명했습니다. 치환 변수는 WHERE 절, 열 이름 및 표현식의 값을 대체할 수 있는 유통성을 제공합니다. 다음을 사용하여 스크립트 파일을 작성하면 보고서를 사용자 정의할 수 있습니다.

- 단일 앤퍼샌드 치환 변수
- 이중 앤퍼샌드 치환 변수
- DEFINE 명령
- UNDEFINE 명령
- 명령행의 치환 변수

다음 명령을 사용하여 읽기 쉬운 보고서를 작성할 수 있습니다.

- COLUMN
- TTITLE
- BTITLE
- BREAK

연습 7 개요

이 연습에서는 다음 내용을 다룹니다.

- 치환 변수를 사용하여 값을 표시하는 질의 작성
- 변수를 포함하는 명령 파일 시작

ORACLE

7-29

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 7 개요

이 연습을 마치면 치환 변수를 사용해 대화식으로 실행할 수 있는 파일을 생성하여 실행 시간 선택 조건을 생성할 수 있습니다.

연습 7

다음 두 문장의 유효성을 판별하여 True 또는 False로 답하십시오.

1. 다음 명령문은 유효합니다.

```
DEFINE & p_val = 100
```

True/False

2. DEFINE 명령은 SQL 명령어입니다.

True/False

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.

```
DEFINE low_date = 01/01/1998
```

```
DEFINE high_date = 01/01/1999
```

EMPLOYEES	HIRE_DATE
Matos, ST_CLERK	15-MAR-98
Vargas, ST_CLERK	09-JUL-98
Taylor, SA REP	24-MAR-98

4. 지정한 위치에 근무하는 사원의 이름, 업무 ID 및 부서 이름을 표시하는 스크립트를 작성하십시오. 부서 위치에 대한 검색 조건은 대소문자를 구분하지 않도록 하고 스크립트 파일을 lab7_4.sql이라는 이름으로 저장하십시오.

EMPLOYEE NAME	JOB_ID	DEPARTMENT NAME
Whalen	AD_ASST	Administration
King	AD_PRES	Executive
Kochhar	AD_VP	Executive
De Haan	AD_VP	Executive
Higgins	AC_MGR	Accounting
Gietz	AC_ACCOUNT	Accounting

6 rows selected.

연습 7(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 줄에 표시되도록 하십시오. 이 스크립트를 lab7_5.sql이라는 이름으로 다시 저장하고 스크립트의 명령을 실행하십시오.

DEPARTMENT NAME	EMPLOYEE NAME	START DATE	SALARY	ANNUAL SALARY
Accounting	Higgins	07-JUN-94	\$12,000.00	\$144,000.00
	Gietz	07-JUN-94	\$8,300.00	\$99,600.00
Administration	Whalen	17-SEP-87	\$4,400.00	\$52,800.00
Executive	King	17-JUN-87	\$24,000.00	\$288,000.00
	Kochhar	21-SEP-89	\$17,000.00	\$204,000.00
	De Haan	13-JAN-93	\$17,000.00	\$204,000.00

8

데이터 조작

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 각 DML 문 설명
- 테이블에 행 삽입
- 테이블의 행 생성
- 테이블에서 행 삭제
- 테이블의 행 병합
- 트랜잭션 제어

ORACLE

8-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 테이블에 행을 삽입하는 방법과 테이블에서 기존 행을 생성 및 삭제하는 방법 그리고 COMMIT, SAVEPOINT 및 ROLLBACK 문을 사용하여 트랜잭션을 제어하는 방법을 설명합니다.

A (T)R
Autism (Autom) Consistency (Consistency) Isolation (Isolation) Durability (Durability)
인자성 일관성 고립성 영속성

데이터 조작어

- 다음 경우에 DML 문이 실행됩니다.
 - 테이블에 새 행 추가
 - 테이블의 기존 행 수정
 - 테이블에서 기존 행 삭제
- 트랜잭션은 논리 작업 단위를 형성하는 DML 문의 모음으로 구성됩니다.

ORACLE

8-3

Copyright © Oracle Corporation, 2001. All rights reserved.

데이터 조작어

DML(데이터 조작어)은 SQL의 핵심 요소로서 데이터베이스에 데이터를 추가, 생성 또는 삭제 할 때 실행합니다. 논리 작업 단위를 형성하는 DML 문의 모음을 트랜잭션이라고 합니다.

예를 들어, 은행 업무 데이터베이스에서 은행 고객이 보통 예금에서 당좌 예금으로 금액을 이체하는 경우 이 트랜잭션은 보통 예금 잔액 감소, 당좌 예금 잔액 증가 및 트랜잭션 저널(journal)에 트랜잭션 기록이라는 세 가지 작업으로 구성됩니다. Oracle server는 이 세 가지 SQL 문이 모두 수행되어 해당 계정에 정확한 잔액이 유지되도록 합니다. 트랜잭션 명령문 중 하나라도 실행되지 못하면 나머지 명령문의 실행도 취소되어야 합니다.

테이블에 새 행 추가

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

새
행

...DEPARTMENTS

테이블에
새 행을 추가
합니다...

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

70	Public Relations	100	1700
----	------------------	-----	------

ORACLE

테이블에 새 행 추가

슬라이드 그래픽은 DEPARTMENTS 테이블에 새 부서를 추가하는 것을 보여줍니다.

INSERT 문 구문

- INSERT 문을 사용하여 테이블에 새 행을 추가합니다.

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

- 이 구문으로는 한 번에 한 행만 추가됩니다.

ORACLE

테이블에 새 행 추가(계속)

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.

새 행 삽입

- 각 열에 대한 값을 포함하는 새 행을 삽입합니다.
- 테이블 열의 기본 순서대로 값을 나열합니다.
- INSERT 절에 열을 나열할 수 있습니다(선택 사항).

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

- 문자 및 날짜 값은 작은 따옴표로 묶습니다.

ORACLE

테이블에 새 행 추가(계속)

각 열의 값을 포함하는 새 행을 삽입할 수 있으므로 INSERT 절에 열 목록이 필요 없지만 열 목록을 사용하지 않는 경우에는 값을 테이블의 기본 열 순서에 따라 나열해야 하며 모든 열에 대해 각각 값을 제공해야 합니다.

```
DESCRIBE departments
```

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

명확하게 하기 위해서는 INSERT 절에 열 목록을 사용하십시오.

문자 및 날짜 값은 작은 따옴표로 묶고 숫자 값은 작은 따옴표로 묶지 않는 것이 좋습니다.

NUMBER 데이터 유형 열에 할당된 숫자 값을 작은 따옴표로 묶으면 암시적(implicit) 변환이 발생할 수 있으므로 작은 따옴표로 묶지 말아야 합니다.

널 값을 갖는 행 삽입

- 암시적(**implicit**) 방법: 열 목록에서 해당 열을 생략합니다.

```
INSERT INTO departments (department_id,  
                        department_name) // ] )  
VALUES      (30, 'Purchasing');  
1 row created.
```

- 명시적(**explicit**) 방법: VALUES 절에서 NULL 키워드를 지정합니다.

```
INSERT INTO departments  
VALUES      (100, 'Finance', NULL, NULL);  
1 row created.
```

ORACLE

8-7

Copyright © Oracle Corporation, 2001. All rights reserved.

널 값 삽입 방법

방법	설명
암시적(Implicit)	열 목록에서 열을 생략합니다.
명시적(Explicit)	VALUES 목록에서 NULL 키워드를 지정합니다. 문자열 및 날짜에 대해 VALUES 목록에서 공백 문자열('')을 지정합니다.

iSQL*Plus DESCRIBE 명령을 통해 Null? 상태를 확인하여 대상 열에서 널 값을 사용할 수 있는지 확인합니다.

Oracle Server는 자동으로 모든 데이터 유형, 데이터 범위를 사용하며 데이터 무결성 제약 조건을 보존합니다. 목록에 없는 열은 명시적(**explicit**)으로 새 행에 널 값을 갖게 됩니다.

다음은 사용자 입력 중에 발생할 수 있는 일반적인 오류입니다.

- NOT NULL 열에 필수 값 누락
- 중복 값으로 고유성 제약 조건 위반
- 외래 키 제약 조건 위반
- CHECK 제약 조건 위반
- 데이터 유형 불일치
- 값의 폭(width)이 너무 넓어 열에 맞지 않음

특정 값 삽입

SYSDATE 함수는 현재 날짜 및 시간을 기록합니다.

```
INSERT INTO employees (employee_id,
                      first_name, last_name,
                      email, phone_number,
                      hire_date, job_id, salary,
                      commission_pct, manager_id,
                      department_id)
VALUES          (113,
                  'Louis', 'Popp',
                  'LPOPP', '515.124.4567',
                  SYSDATE, 'AC_ACCOUNT', 6900,
                  NULL, 205, 100);
1 row created.
```

ORACLE

8-8

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL 함수를 사용하여 특정 값 삽입

함수를 사용하여 테이블에 특정 값을 입력할 수 있습니다.

슬라이드 예제는 EMPLOYEES 테이블에 사원 Popp에 대한 정보를 기록합니다. 이 때 HIRE_DATE 열에 현재 날짜 및 시간을 제공하기 위해 SYSDATE 함수를 사용합니다.

테이블에 행을 삽입할 때 USER 함수를 사용하여 현재 사용자 이름을 기록할 수도 있습니다.

테이블에 추가한 항목 확인

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

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	COMMISSION_PCT
113	Popp	AC_ACCOUNT	27-SEP-01	

특정 날짜 값 삽입

- 새 사원을 추가합니다.

```
INSERT INTO employees
VALUES      (114,
              'Den', 'Raphealy',
              'DRAPHEAL', '515.127.4561',
              TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
              'AC_ACCOUNT', 11000, NULL, 100, 30);
1 row created.
```

- 추가한 항목을 확인합니다.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_P
114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	AC_ACCOUNT	11000	

ORACLE

특정 날짜 및 시간 값 삽입

날짜 값을 삽입할 때 주로 사용하는 DD-MON-YY 형식에서는 세기 기본값으로 현재 세기를 사용한다는 점을 기억해 두어야 합니다. 날짜에도 시간 정보가 포함되므로 기본 시간은 자정(00:00:00)입니다.

날짜를 기본 형식이 아닌 다른 형식(예: 다른 세기 또는 특정 시간)으로 입력하려면 TO_DATE 함수를 사용해야 합니다.

슬라이드 예제는 사원 Raphealy의 정보를 EMPLOYEES 테이블에 기록하며 HIRE_DATE 열을 1999년 2월 3일로 설정합니다. 슬라이드에 표시된 명령문 대신 다음 명령문을 사용할 경우 hire_date의 연도는 2099로 해석됩니다.

```
INSERT INTO employees
VALUES      (114,
              'Den', 'Raphealy',
              'DRAPHEAL', '515.127.4561',
              '03-FEB-99',
              'AC_ACCOUNT', 11000, NULL, 100, 30);
```

RR 형식을 사용할 경우에는 현재 세기가 아닌 경우에도 자동으로 정확한 세기 정보를 제공합니다.

스크립트 작성

- SQL 문에 & 치환을 사용하여 사용자에게 값을 요구합니다.
- &는 변수 값에 대한 위치 표시자입니다.

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

Define Substitution Variables

department_id 40
department_name Human Resources
location 2500

1 row created.

ORACLE

데이터 조작 스크립트 작성

치환 변수가 포함된 명령을 파일에 저장한 후 이 파일에 있는 명령을 실행할 수 있습니다. 위의 예제는 부서 정보를 DEPARTMENTS 테이블에 기록합니다.

스크립트 파일을 실행하면 & 치환 변수의 값을 입력하라는 프롬프트가 나타납니다. 사용자가 입력한 값은 명령문으로 치환됩니다. 이런 방식으로 매번 실행할 때마다 다른 값을 제공하면서 동일한 스크립트 파일을 반복해서 실행할 수 있습니다.

다른 테이블에서 행 복사

- 서브 쿼리를 포함하는 INSERT 문을 작성합니다.

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

```
4 rows created.
```

- VALUES 절은 사용하지 않습니다.
- INSERT 절의 열 수와 서브 쿼리의 열 수를 일치시킵니다.

ORACLE

8-11

Copyright © Oracle Corporation, 2001. All rights reserved.

다른 테이블에서 행 복사

INSERT 문을 사용하면 기존 테이블에서 파생되는 값을 포함하는 행을 테이블에 추가할 수 있습니다. VALUES 절 대신 서브 쿼리를 사용하십시오.

구문

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

구문 설명:

table	테이블 이름입니다.
column	데이터를 삽입할 테이블의 열 이름입니다.
subquery	테이블에 삽입 할 행을 반환하는 서브 쿼리입니다.

INSERT 절의 열 목록에서 열 수 및 데이터 유형은 서브 쿼리의 열 수 및 데이터 유형과 일치해야 합니다. 테이블의 행에 대해 복사본을 생성하려면 서브 쿼리에서 SELECT *를 사용하십시오.

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

자세한 내용은 *Oracle9i SQL Reference*, “SELECT”의 서브 쿼리 단원을 참조하십시오.

테이블의 데이터 변경

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_PCT
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000		90
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000		90
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000		90
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000		60
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000		60
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200		60
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800		50

EMPLOYEES 테이블의 행 갱신

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_PCT
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000		90
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000		90
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000		90
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000		30
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000		30
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200		30
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800		50

ORACLE

테이블의 데이터 변경

슬라이드 그래픽은 부서 60에 있는 사원의 부서 번호를 부서 30으로 변경하는 것을 보여 줍니다.

UPDATE 문 구문

- UPDATE 문을 사용하여 기존 행을 수정합니다.

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

- 필요한 경우 한 번에 여러 행을 갱신합니다.

ORACLE

8-13

Copyright © Oracle Corporation, 2001. All rights reserved.

행 갱신

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.

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

참고: 일반적으로 하나의 행을 식별하려면 기본 키를 사용합니다. 다른 열을 사용할 경우 예기치 않게 여러 행이 갱신될 수 있습니다. 예를 들어, 동일한 이름을 가진 사원이 여러 명 있을 경우 EMPLOYEES 테이블에서 이름을 사용하여 하나의 행을 식별하는 것은 위험합니다.

테이블의 행 갱신

- WHERE 절을 지정하여 특정 행을 수정합니다.

```
UPDATE employees
SET department_id = 70
WHERE employee_id = 113;
1 row updated.
```

- WHERE 절을 생략하면 테이블의 모든 행이 수정됩니다.

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

ORACLE

8-14

Copyright © Oracle Corporation, 2001. All rights reserved.

행 갱신(계속)

WHERE 절을 지정하면 UPDATE 문이 특정 행을 수정합니다. 슬라이드 예제는 사원 113(Popp)을 부서 70으로 보냅니다.

WHERE 절을 생략하면 테이블의 모든 행이 수정됩니다.

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

LAST_NAME	DEPARTMENT_ID
King	110
Kochhar	110
De Haan	110
Hunold	110
Ernst	110
Lorentz	110
...	

22 rows selected.

참고: COPY_EMP 테이블과 EMPLOYEES 테이블의 데이터는 동일합니다.

서브 쿼리로 두 열 갱신

사원 114의 업무 및 급여를 사원 205의 업무 및 급여와 일치
하도록 갱신합니다.

```
UPDATE employees
SET job_id = (SELECT job_id
               FROM employees
              WHERE employee_id = 205),
    salary = (SELECT salary
               FROM employees
              WHERE employee_id = 205)
WHERE employee_id = 114;
1 row updated.
```

ORACLE

8-15

Copyright © Oracle Corporation, 2001. All rights reserved.

서브 쿼리로 두 열 갱신

여러 서브 쿼리를 작성하면 UPDATE 문의 SET 절에서 여러 열을 갱신할 수 있습니다.

구문

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

참고: 행이 갱신되지 않은 경우, “0 rows updated.”라는 메시지가 반환됩니다.

다른 테이블을 기반으로 행 갱신

UPDATE 문에 서브 쿼리를 사용하면 다른 테이블의 값을
기반으로 테이블의 행을 갱신할 수 있습니다.

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

ORACLE

다른 테이블을 기반으로 행 갱신

UPDATE 문에 서브 쿼리를 사용하여 테이블의 행을 갱신할 수 있습니다. 슬라이드 예제는 EMPLOYEES 테이블의 값을 기반으로 COPY_EMP 테이블을 갱신하며 사원 200과 업무 ID가 동일한 모든 사원의 부서 번호를 사원 100의 현재 부서 번호로 변경합니다.

행 간신: 무결성 제약 조건 오류

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

```
UPDATE employees  
*  
ERROR at line 1:  
ORA-02291: integrity constraint (HR.EMP_DEPT_FK)  
violated - parent key not found
```

부서 번호 55가 존재하지 않습니다.

ORACLE

8-17

Copyright © Oracle Corporation, 2001. All rights reserved.

무결성 제약 조건 오류

무결성 제약 조건의 영향을 받는 값을 포함하는 레코드를 갱신하면 오류가 반환됩니다.

슬라이드 예제에서 부모 테이블 DEPARTMENTS에는 부서 번호 55가 없으므로 parent key 위반(ORA-02291)이 발생합니다.

참고: 무결성 제약 조건은 데이터가 미리 정의된 규칙을 따르도록 보장해 줍니다. 무결성 제약 조건에 대한 내용은 뒷 단원에서 자세하게 설명합니다.

테이블에서 행 제거

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
100	Finance		
50	Shipping	124	1500
60	IT	103	1400

DEPARTMENTS 테이블에서 행 삭제

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
50	Shipping	124	1500
60	IT	103	1400

ORACLE

테이블에서 행 제거

슬라이드 그래픽은 DEPARTMENTS 테이블에서 Finance 부서를 제거합니다. DEPARTMENTS 테이블에 제약 조건이 정의되어 있지 않다고 가정합니다.

DELETE 문

DELETE 문을 사용하여 테이블에서 기존 행을 제거할 수 있습니다.

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

ORACLE

8-18

Copyright © Oracle Corporation, 2001. All rights reserved.

행 삭제

DELETE 문을 사용하여 기존 행을 제거할 수 있습니다.

구문 설명:

<i>table</i>	테이블 이름입니다.
<i>condition</i>	삭제될 행을 식별하며 열 이름, 표현식, 상수, 서브 쿼리 및 비교 연산자로 구성됩니다.

참고: 행이 삭제되지 않은 경우, “0 rows deleted.”라는 메시지가 반환됩니다.

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

테이블에서 행 삭제

- WHERE 절을 지정하면 특정 행이 삭제됩니다.

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

- WHERE 절을 생략하면 테이블의 모든 행이 삭제됩니다.

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

ORACLE

8-20

Copyright © Oracle Corporation, 2001. All rights reserved.

행 삭제(계속)

DELETE 문에 WHERE 절을 지정하여 특정 행을 삭제할 수 있습니다. 슬라이드 예제는 DEPARTMENTS 테이블에서 Finance 부서를 삭제합니다. SELECT 문을 사용하여 삭제된 행을 표시해 보면 삭제 작업을 확인할 수 있습니다.

```
SELECT *
FROM departments
WHERE department_name = 'Finance';
no rows selected.
```

WHERE 절을 생략하면 테이블의 모든 행이 삭제됩니다. 슬라이드의 두번째 예제에는 WHERE 절이 지정되지 않았으므로 COPY_EMP 테이블의 모든 행이 삭제됩니다.

예제

WHERE 절에서 지정한 행을 제거합니다.

```
DELETE FROM employees
WHERE employee_id = 114;
1 row deleted.
```

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

다른 테이블을 기반으로 행 삭제

DELETE 문에 서브 쿼리를 사용하여 테이블에서 다른 테이블의 값을 기반으로 하는 행을 제거합니다.

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

ORACLE

다른 테이블을 기반으로 행 삭제

서브 쿼리를 사용하면 다른 테이블의 값을 기반으로 행을 삭제할 수 있습니다. 슬라이드 예제는 부서 이름에 문자열 “Public”이 포함된 부서의 모든 사원을 삭제합니다. 서브 쿼리는 문자열 “Public”이 포함된 부서 이름을 기준으로 DEPARTMENTS 테이블을 검색하여 기본 질의에 부서 번호를 제공합니다. 기본 질의는 이 부서 번호를 기준으로 EMPLOYEES 테이블에서 데이터 행을 삭제합니다.

행 삭제: 무결성 제약 조건 오류

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

```
DELETE FROM departments  
*  
ERROR at line 1:  
ORA-02292: integrity constraint (HR.EMP_DEPT_FK)  
violated - child record found
```

다른 테이블에서 외래 키로 사용되는 기본 키를 포함하는 행은 삭제할 수 없습니다.

ORACLE

8-22

Copyright © Oracle Corporation, 2001. All rights reserved.

무결성 제약 조건 오류

무결성 제약 조건의 영향을 받는 값을 포함하는 레코드를 삭제하면 오류가 반환됩니다.

슬라이드 예제의 경우 DEPARTMENTS 테이블에서 부서 번호 60을 삭제하려고 시도하면 부서 번호가 EMPLOYEES 테이블에서 외래 키로 사용되므로 오류가 발생합니다. 삭제하려는 부모 레코드가 자식 레코드를 포함하는 경우 child record found 위반(ORA-02292)이 발생합니다.

다음 명령문은 부서 70에 사원이 없으므로 제대로 동작합니다.

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

1 row deleted.

INSERT 문에 서브 쿼리 사용

```
INSERT INTO
  (SELECT employee_id, last_name,
         email, hire_date, job_id, salary,
         department_id
    FROM   employees
   WHERE  department_id = 50)
VALUES (99999, 'Taylor', 'DTAYLOR',
        TO_DATE('07-JUN-99', 'DD-MON-RR'),
        'ST_CLERK', 5000, 50);

1 row created.
```

ORACLE

8-23

Copyright © Oracle Corporation, 2001. All rights reserved.

INSERT 문에 서브 쿼리 사용

INSERT 문의 INTO 절에 테이블 이름 대신 서브 쿼리를 사용할 수 있습니다.

이 서브 쿼리의 select 목록에는 VALUES 절의 열 목록과 동일한 수의 열 목록이 있어야 합니다. INSERT 문이 성공적으로 동작하려면 기본 테이블의 열에 적용되는 모든 규칙을 따라야 합니다. 예를 들어, 중복되는 사원 ID를 입력할 수도 없고 널이 아닌 필수 열에 대해 값을 생략할 수도 없습니다.

INSERT 문에 서브 쿼리 사용

```
SELECT employee_id, last_name, email, hire_date,  
       job_id, salary, department_id  
FROM   employees  
WHERE  department_id = 50;
```

EMPLOYEE_ID	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
124	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5000	50
141	Rajs	TRAJS	17-OCT-96	ST_CLERK	3500	50
142	Davies	CDAVIES	29-JAN-97	ST_CLERK	3100	50
143	Matos	RMATOS	15-MAR-98	ST_CLERK	2600	50
144	Vargas	PVARGAS	09-JUL-98	ST_CLERK	2500	50
99999	Taylor	DTAYLOR	07-JUN-99	ST_CLERK	5000	50

6 rows selected.

ORACLE

INSERT 문에 서브 쿼리 사용

예제는 INSERT 문이 수행될 테이블을 식별하는 데 사용된 서브 쿼리의 결과를 보여줍니다.

DML 문에 WITH CHECK OPTION 키워드 사용

- DML 문의 테이블 및 열을 식별하는 데 서브 쿼리를 사용합니다.
- WITH CHECK OPTION 키워드를 사용하면 서브 쿼리에 없는 행이 변경되는 것을 막을 수 있습니다.

```
INSERT INTO (SELECT employee_id, last_name, email,
               hire_date, job_id, salary
              FROM employees
             WHERE department_id = 50 WITH CHECK OPTION)
VALUES (99998, 'Smith', 'JSMITH',
        TO_DATE('07-JUN-99', 'DD-MON-RR'),
        'ST_CLERK', 5000);
INSERT INTO *
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

ORACLE

WITH CHECK OPTION 키워드

WITH CHECK OPTION을 지정하면 INSERT, UPDATE 또는 DELETE 문에서 테이블 대신 서브 쿼리가 사용된 경우 서브 쿼리에 포함되지 않은 행이 해당 테이블에 생성되는 것을 막을 수 있습니다.

이 예제는 WITH CHECK OPTION 키워드를 사용합니다. 서브 쿼리가 부서 50인 행을 식별하지만 부서 ID가 SELECT 목록에 없고 VALUES 목록에서 이에 대한 값이 제공되어 있지 않습니다. 이 행을 삽입하면 부서 ID가 널인 행이 만들어지는데 이 행은 서브 쿼리의 결과에 포함되지 않는 행입니다.

명시적(**Explicit**) 기본 기능 개요

- 명시적 기본 기능을 사용하면 열 기본값이 필요한 경우 **DEFAULT** 키워드를 열 값으로 사용할 수 있습니다.
- 이 기능은 **SQL: 1999** 표준을 준수하기 위해 추가되었습니다.
- 이 기능을 통해 데이터에 기본값이 적용될 위치 및 시기 를 제어할 수 있습니다.
- 명시적 기본값은 **INSERT** 및 **UPDATE** 문에 사용할 수 있습니다.

ORACLE

8-25

Copyright © Oracle Corporation, 2001. All rights reserved.

명시적 기본값

DEFAULT 키워드는 **INSERT** 및 **UPDATE** 문에서 기본 열 값을 지정하는 데 사용할 수 있습니다.
기본값이 존재하지 않는 경우, 널 값이 사용됩니다.

명시적(**Explicit**) 기본값 사용

- INSERT에 사용된 DEFAULT

```
INSERT INTO departments
(department_id, department_name, manager_id)
VALUES (300, 'Engineering', DEFAULT);
```

- UPDATE에 사용된 DEFAULT

```
UPDATE departments
SET manager_id = DEFAULT WHERE department_id = 10;
```

ORACLE

8-27

Copyright © Oracle Corporation, 2001. All rights reserved.

명시적 기본값 사용

이전에 열의 기본값으로 지정해 둔 값을 열에 설정하려면 DEFAULT를 지정합니다. 해당 열에 기본값이 지정되지 않은 경우 널이 설정됩니다.

첫번째 예제에서 INSERT 문은 MANAGER_ID 열에 대해 기본값을 사용합니다. 이 열에 정의된 기본값이 없는 경우 널 값이 대신 삽입됩니다.

두번째 예제는 UPDATE 문을 사용하여 MANAGER_ID 열을 기본값인 부서 10으로 설정합니다. 이 열에 정의된 기본값이 없는 경우 값을 널로 변경합니다.

참고: 테이블을 생성할 때 열의 기본값을 지정할 수 있습니다. 이에 대해서는 “테이블 생성 및 관리” 단원에서 설명합니다.

MERGE 문

- 데이터베이스 테이블에서 조건에 따라 데이터를 갱신하거나 삽입하는 기능을 제공합니다.
- 해당 행이 존재하는 경우 UPDATE를 수행하고, 새로운 행일 경우 INSERT를 수행합니다.
 - 별도의 갱신을 수행할 필요가 없습니다.
 - 성능이 향상되고 사용하기 편리합니다.
 - 데이터 웨어하우징 응용 프로그램에 유용합니다.

ORACLE

3-28

Copyright © Oracle Corporation, 2001. All rights reserved.

MERGE 문

SQL이 확장되어 MERGE 문이 포함되었습니다. 이 명령문을 사용하면 행을 조건에 따라 테이블에서 갱신하거나 삽입할 수 있으므로 UPDATE 문을 여러 번 사용하지 않아도 됩니다. 대상 테이블에 대한 갱신 또는 삽입 여부는 ON 절의 조건에 의해 결정됩니다.

MERGE 명령은 INSERT 및 UPDATE 명령을 결합한 것으로 대상 테이블에 대해서는 INSERT 및 UPDATE 권한을 모두 가지고 있어야 하고 원본 테이블에 대해서는 SELECT 권한을 가지고 있어야 합니다.

MERGE 문은 결정론적입니다. 동일한 MERGE 문으로 대상 테이블의 동일 행을 여러 번 갱신할 수 없습니다.

다른 접근 방법은 PL/SQL 루프와 여러 DML 문을 사용하는 것입니다. 하지만 MERGE 문이 사용하기 쉽고 하나의 SQL 문으로 더욱 간단하게 표현됩니다.

MERGE 문은 여러 데이터 웨어하우징 응용 프로그램에 적합합니다. 예를 들어, 데이터 웨어하우징 응용 프로그램에서 여러 원본의 데이터를 사용하여 작업해야 할 경우도 있으며 중복된 데이터가 있을 수도 있습니다. MERGE 문을 사용하면 조건에 따라 행을 추가하거나 수정할 수 있습니다.

MERGE 문 구문

MERGE 문을 사용하면 테이블에서 조건에 따라 행을 삽입 또는 갱신할 수 있습니다.

```
MERGE INTO table_name table_alias
  USING (table/view/sub_query) alias
  ON (join condition)
 WHEN MATCHED THEN
   UPDATE SET
     col1 = col_val1,
     col2 = col2_val
 WHEN NOT MATCHED THEN
   INSERT (column_list)
   VALUES (column_values);
```

ORACLE

8-29

Copyright © Oracle Corporation, 2001. All rights reserved.

행 병합

MERGE 문을 사용하면 조건에 따라 기존 행을 갱신하고 새로운 행을 삽입할 수 있습니다.

구문 설명:

INTO 절	갱신 또는 삽입할 대상 테이블입니다.
USING 절	갱신 또는 삽입될 데이터의 원본을 지정하며 테이블, 뷰 또는 서브 쿼리일 수 있습니다.
ON 절	MERGE 연산이 갱신 또는 삽입을 수행하는 기준이 되는 조건입니다.
WHEN MATCHED WHEN NOT MATCHED	서버가 조인 조건의 결과에 응답할 방식을 지정합니다.

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

행 병합

COPY_EMP 테이블에서 행을 삽입 또는 갱신하여 EMPLOYEES 테이블과 일치시킵니다.

```
MERGE INTO copy_emp c
  USING employees e
    ON (c.employee_id = e.employee_id)
  WHEN MATCHED THEN
    UPDATE SET
      c.first_name      = e.first_name,
      c.last_name       = e.last_name,
      ...
      c.department_id  = e.department_id
  WHEN NOT MATCHED THEN
    INSERT VALUES(e.employee_id, e.first_name, e.last_name,
                  e.email, e.phone_number, e.hire_date, e.job_id,
                  e.salary, e.commission_pct, e.manager_id,
                  e.department_id);
```

ORACLE

8-30

Copyright © Oracle Corporation, 2001. All rights reserved.

행 병합 예제

```
MERGE INTO copy_emp c
  USING employees e
    ON (c.employee_id = e.employee_id)
  WHEN MATCHED THEN
    UPDATE SET
      c.first_name      = e.first_name,
      c.last_name       = e.last_name,
      c.email           = e.email,
      c.phone_number    = e.phone_number,
      c.hire_date       = e.hire_date,
      c.job_id          = e.job_id,
      c.salary          = e.salary,
      c.commission_pct = e.commission_pct,
      c.manager_id     = e.manager_id,
      c.department_id   = e.department_id
  WHEN NOT MATCHED THEN
    INSERT VALUES(e.employee_id, e.first_name, e.last_name,
                  e.email, e.phone_number, e.hire_date, e.job_id,
                  e.salary, e.commission_pct, e.manager_id,
                  e.department_id);
```

이 예제는 COPY_EMP 테이블의 EMPLOYEE_ID와 EMPLOYEES 테이블의 EMPLOYEE_ID를 비교하여 일치하면 COPY_EMP 테이블의 행을 갱신하여 EMPLOYEES 테이블의 행과 일치되도록 만듭니다. 일치하지 않으면 EMPLOYEES 테이블의 해당 행이 COPY_EMP 테이블에 삽입됩니다.

행 병합

```
SELECT *
FROM COPY_EMP;

no rows selected

MERGE INTO copy_emp c
  USING employees e
  ON (c.employee_id = e.employee_id)
WHEN MATCHED THEN
    UPDATE SET
      ...
WHEN NOT MATCHED THEN
    INSERT VALUES...;

SELECT *
FROM COPY_EMP;

20 rows selected.
```

ORACLE

행 병합 예제

c.employee_id = e.employee_id 조건이 평가됩니다. COPY_EMP 테이블이 비어 있으므로 조건은 False를 반환합니다(일치하는 행 없음). 이에 따라 WHEN NOT MATCHED 절이 수행되어 MERGE 명령은 EMPLOYEES 테이블의 행을 COPY_EMP 테이블에 삽입합니다.

COPY_EMP 테이블에 행이 존재하고 사원 ID가 양쪽 테이블(COPY_EMP 및 EMPLOYEES 테이블)에서 일치하는 경우에는 COPY_EMP 테이블의 기존 행이 갱신되어 EMPLOYEES 테이블과 일치하게 됩니다.

데이터베이스 트랜잭션

데이터베이스 트랜잭션은 다음 중 하나로 구성됩니다.

- 데이터를 일관성 있게 변경하는 하나 이상의 **DML** 문
- **DDL** 문 하나
- **DCL** 문 하나

ORACLE

8-32

Copyright © Oracle Corporation, 2001. All rights reserved.

데이터베이스 트랜잭션

Oracle server는 트랜잭션을 기반으로 데이터 일관성을 보장합니다. 트랜잭션은 데이터를 변경 할 때 보다 많은 융통성과 제어 기능을 제공하며 사용자 프로세스가 중단되거나 시스템 장애가 발생한 경우에도 데이터 일관성을 보장합니다.

트랜잭션은 데이터를 일관성 있게 변경하는 DML 문으로 구성됩니다. 예를 들어, 두 계좌 사이에 자금을 이체하는 경우 한 계좌의 출금 금액과 다른 계좌의 입금 금액이 동일해야 하며 반드시 두 작업이 함께 처리되거나 취소되어야 합니다. 즉 출금이 처리되지 않은 상태에서 입금이 처리되어서는 안됩니다.

트랜잭션 유형

유형	설명
DML (데이터 조작어)	Oracle server에서 단일 항목 또는 작업의 논리 단위로 취급되는 여러 DML 문으로 구성됩니다.
DDL (데이터 정의어)	DDL 문 하나만으로 구성됩니다.
DCL (데이터 제어어)	DCL 문 하나만으로 구성됩니다.

데이터베이스 트랜잭션

- 첫번째 **DML SQL** 문이 실행될 때 시작됩니다.
- 다음 이벤트가 발생하면 종료됩니다.
 - **COMMIT** 또는 **ROLLBACK** 문이 실행되는 경우
 - **DDL** 또는 **DCL** 문이 실행되는 경우(자동 커밋)
 - **iSQL*Plus**를 종료하는 경우
 - 시스템이 고장난 경우

ORACLE

8-33

Copyright © Oracle Corporation, 2001. All rights reserved.

트랜잭션 시작 및 종료 시기

트랜잭션은 첫번째 DML 문이 실행될 때 시작되어 다음 이벤트가 발생하면 종료됩니다.

- **COMMIT** 또는 **ROLLBACK** 문이 실행되는 경우
- **DDL** 문(예: **CREATE**)이 실행되는 경우
- **DCL** 문이 실행되는 경우
- **iSQL*Plus**를 종료하는 경우
- 시스템에 장애가 있거나 시스템이 고장난 경우

트랜잭션이 종료되면 실행 가능한 다음 SQL 문이 다음 트랜잭션을 자동으로 시작합니다.

DDL 문 또는 **DCL** 문은 자동으로 커밋되므로 트랜잭션이 암시적(**implicit**)으로 종료됩니다.

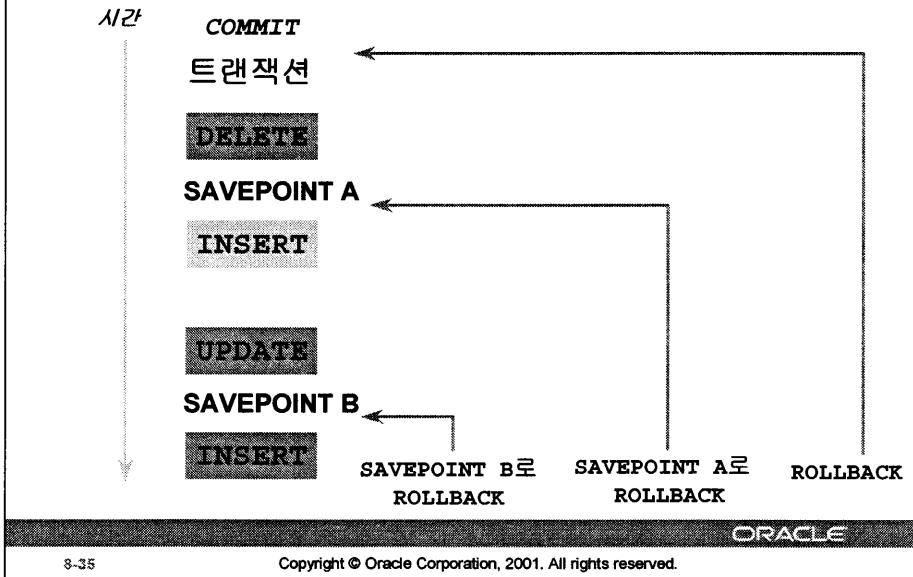
COMMIT 및 ROLLBACK 문의 장점

COMMIT 및 ROLLBACK 문으로 다음을 수행할 수 있습니다.

- 데이터 일관성을 보장합니다.
- 데이터 변경 내용을 영구히 저장하기 전에 미리 볼 수 있습니다.
- 논리적으로 관련된 작업을 묶습니다.

ORACLE

트랜잭션 제어



8-35

Copyright © Oracle Corporation, 2001. All rights reserved.

명시적(Explicit) 트랜잭션 제어문

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

명령문	설명
COMMIT	보류 중인 모든 데이터 변경 내용을 영구히 저장하고 현재 트랜잭션을 종료합니다.
SAVEPOINT name	현재 트랜잭션 내에 저장점을 표시합니다.
ROLLBACK	ROLLBACK은 보류 중인 모든 데이터 변경 내용을 버리고 현재 트랜잭션을 종료합니다.
ROLLBACK TO SAVEPOINT name	ROLLBACK TO SAVEPOINT는 현재 트랜잭션을 지정된 저장점으로 롤백하여 롤백하는 저장점 이후에 생성된 모든 변경 내용 및 저장점을 버립니다. TO SAVEPOINT 절을 생략할 경우 ROLLBACK 문은 전체 트랜잭션을 롤백합니다. 저장점은 논리적인 것이므로 작성한 저장점을 나열할 수는 없습니다.

참고: SAVEPOINT는 ANSI 표준 SQL이 아닙니다.

표시자까지 변경 내용 풀백

- **SAVEPOINT** 문을 사용하여 현재 트랜잭션에 표시자를 생성합니다.
- **ROLLBACK TO SAVEPOINT** 문을 사용하여 해당 표시자 까지 풀백합니다.

```
UPDATE...
SAVEPOINT update_done;
Savepoint created.
INSERT...
ROLLBACK TO update_done;
Rollback complete.
```

ORACLE

저장점까지 변경 내용 풀백

SAVEPOINT 문을 사용하여 현재 트랜잭션에 표시자를 생성하면 트랜잭션을 더 작은 부분으로 나눌 수 있습니다. 그런 다음 **ROLLBACK TO SAVEPOINT** 문을 사용하면 보류 중인 변경 내용 중에서 해당 표시자 이후에 변경된 내용은 버릴 수 있습니다.

두 번째 표시자를 이전 표시자와 동일한 이름으로 생성하면 이전 표시자가 삭제됩니다.

암시적(**implicit**) 트랜잭션 처리

- 자동 커밋은 다음 상황에서 발생합니다.
 - DDL 문이 실행되는 경우
 - DCL 문이 실행되는 경우
 - COMMIT 또는 ROLLBACK 문이 명시적(**explicit**)으로 실행되지 않은 채 *iSQL*Plus*가 정상적으로 종료되는 경우
- 자동 롤백은 *iSQL*Plus*가 비정상적으로 종료되거나 시스템에 장애가 있을 때 발생합니다.

ORACLE

8-37

Copyright © Oracle Corporation, 2001. All rights reserved.

암시적 트랜잭션 처리

상태	상황
자동 커밋	DDL 문 또는 DCL 문이 실행되는 경우. COMMIT 또는 ROLLBACK 명령이 명시적(explicit)으로 실행되지 않은 채 <i>iSQL*Plus</i> 가 정상적으로 종료되는 경우
자동 롤백	<i>iSQL*Plus</i> 가 비정상적으로 종료되거나 시스템 장애가 발생한 경우

참고: *iSQL*Plus*에서는 또 다른 명령을 사용할 수도 있습니다. AUTOCOMMIT 명령은 ON 또는 OFF로 토글할 수 있는데 ON으로 설정하면 각 개별 DML 문이 실행되자마자 커밋되어 변경 내용을 롤백할 수 없으며 OFF로 설정하면 COMMIT을 명시적으로 실행할 수 있습니다. COMMIT 문은 DDL 문이 실행될 때 또는 *iSQL*Plus*가 종료될 때 실행됩니다.

시스템 장애

시스템 장애로 인해 트랜잭션이 중단되면 트랜잭션 전체가 자동으로 롤백되어 잘못된 데이터 변경으로 인한 오류 발생을 방지하며 테이블을 마지막 커밋된 상태로 되돌립니다. Oracle server는 이러한 방식으로 테이블의 무결성을 보존합니다.

*iSQL*Plus*에서 Exit 버튼을 누르면 세션이 정상적으로 종료됩니다. *SQL*Plus*에서는 프롬프트에 EXIT 명령을 입력하면 정상적으로 종료됩니다. 창을 닫는 것은 비정상적인 종료로 해석됩니다.

COMMIT 또는 ROLLBACK

실행 이전의 데이터 상태

- 데이터를 이전 상태로 복구할 수 있습니다.
- 현재 사용자는 **SELECT** 문을 사용하여 **DML** 작업 결과를 검토할 수 있습니다.
- 현재 사용자가 사용 중인 **DML** 문의 결과를 다른 사용자는 볼 수 없습니다.
- 관련 행이 잠기므로 다른 사용자는 관련 행의 데이터를 변경할 수 없습니다.

ORACLE

변경 내용 커밋

트랜잭션에서 변경한 데이터는 트랜잭션을 커밋할 때까지는 임시 데이터입니다.

COMMIT 또는 ROLLBACK 문이 실행되기 이전의 데이터 상태

- 데이터 조작 작업은 주로 데이터베이스 베퍼에 영향을 주기 때문에 데이터를 이전 상태로 복구할 수 있습니다.
- 현재 사용자는 테이블을 질의하여 데이터 조작 결과를 검토할 수 있습니다.
- 현재 사용자가 수행한 데이터 조작 결과를 다른 사용자는 볼 수 없습니다. Oracle server는 읽기 일관성을 제공하여 각 사용자가 마지막 커밋된 상태의 데이터를 볼 수 있도록 합니다.
- 관련 행이 잠기므로 다른 사용자는 관련 행의 데이터를 변경할 수 없습니다.

COMMIT 실행 이후의 데이터 상태

- 데이터 변경 내용이 데이터베이스에 영구히 저장됩니다.
- 데이터의 이전 상태는 완전히 없어집니다.
- 모든 사용자가 결과를 볼 수 있습니다.
- 관련 행 잠금이 해제되어 다른 사용자가 행을 조작할 수 있습니다.
- 모든 저장점이 지워집니다.

ORACLE

8-38

Copyright © Oracle Corporation, 2001. All rights reserved.

변경 내용 커밋(계속)

COMMIT 문을 사용하여 보류 중인 변경 내용을 모두 영구히 저장합니다. 다음은 COMMIT 문이 수행된 이후의 상태입니다.

- 데이터 변경 내용이 데이터베이스에 기록됩니다.
- 데이터의 이전 상태는 완전히 없어집니다.
- 모든 사용자가 트랜잭션 결과를 볼 수 있습니다.
- 관련 행 잠금이 해제되어 다른 사용자가 행을 조작할 수 있습니다.
- 모든 저장점이 지워집니다.

데이터 커밋

- 변경을 수행합니다.

```
DELETE FROM employees  
WHERE employee_id = 99999;  
1 row deleted.  
  
INSERT INTO departments  
VALUES (290, 'Corporate Tax', NULL, 1700);  
1 row inserted.
```

- 변경 내용을 커밋합니다.

```
COMMIT;  
Commit complete.
```

ORACLE

8-40

Copyright © Oracle Corporation, 2001. All rights reserved.

변경 내용 커밋(계속)

슬라이드 예제는 EMPLOYEES 테이블에서 행을 삭제하고 DEPARTMENTS 테이블에 새로운 행을 삽입합니다. 그런 다음 COMMIT 문을 실행하여 변경 내용을 영구히 저장합니다.

예제

DEPARTMENTS 테이블에서 부서 290과 300을 제거하고 COPY_EMP 테이블에서 행을 생성합니다. 데이터 변경 내용을 영구히 저장합니다.

```
DELETE FROM departments  
WHERE department_id IN (290, 300);  
2 rows deleted.
```

```
UPDATE copy_emp  
SET department_id = 80  
WHERE employee_id = 206;  
1 row updated.
```

```
COMMIT;  
Commit Complete.
```

ROLLBACK 실행 이후의 데이터 상태

ROLLBACK 문을 사용하여 보류 중인 변경 내용을 모두 버립니다.

- 데이터 변경이 취소됩니다.
- 데이터가 이전 상태로 복구됩니다.
- 관련 행에 대한 잠금이 해제됩니다.

```
DELETE FROM copy_emp;
22 rows deleted.
ROLLBACK;
Rollback complete.
```

ORACLE

8-41

Copyright © Oracle Corporation, 2001. All rights reserved.

변경 내용 롤백

ROLLBACK 문을 사용하여 보류 중인 변경 내용을 모두 버립니다. 다음은 ROLLBACK 문이 실행된 이후의 상태입니다.

- 데이터 변경이 취소됩니다.
- 데이터가 이전 상태로 복구됩니다.
- 관련 행에 대한 잠금이 해제됩니다.

예제

TEST 테이블에서 레코드를 제거하다가 실수로 테이블의 내용이 삭제되는 경우가 발생할 수 있는데, 이러한 경우 오류를 수정하고 적합한 명령문을 재실행하여 데이터 변경 내용을 영구히 저장할 수 있습니다.

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

ROLLBACK;
Rollback complete.

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

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

COMMIT;
Commit complete.
```

명령문 레벨 롤백

- 트랜잭션 실행 중 단일 DML 문 실행에 실패하면 해당 명령문만 롤백됩니다.
- Oracle server는 암시적(implicit) 저장점을 구현합니다.
- 다른 모든 변경 내용은 유지됩니다.
- 사용자는 COMMIT 또는 ROLLBACK 문을 실행하여 트랜잭션을 명시적(explicit)으로 종료해야 합니다.

ORACLE

명령문 레벨 롤백

명령문 실행 오류가 감지된 경우 암시적 롤백에 의해 트랜잭션 일부가 버려질 수 있습니다.

트랜잭션을 실행하는 도중 단일 DML 문 실행에 실패하면 명령문 레벨의 롤백에 의해 DML 문 실행은 취소되지만 해당 트랜잭션의 이전 DML 문에 의해 변경된 내용은 그대로 유지됩니다. 따라서 이를 변경 내용은 사용자가 명시적으로 커밋 또는 롤백할 수 있습니다.

오라클은 DDL(데이터 정의어) 문 전후에 암시적으로 커밋을 실행하므로 DDL 문이 성공적으로 실행되지 않았을지라도 서버가 실행한 커밋 때문에 이전 명령문을 롤백할 수 없게 됩니다.

COMMIT 또는 ROLLBACK 문을 실행하여 트랜잭션을 명시적으로 종료하십시오.

읽기 일관성

- 읽기 일관성을 통해 항상 일관성 있는 데이터 뷰를 볼 수 있습니다.
- 한 사용자가 변경한 내용이 다른 사용자가 변경한 내용과 충돌하지 않습니다.
- 읽기 일관성은 동일한 데이터에 대해 다음을 보장합니다.
 - 데이터를 읽는 중에는 쓸 수 없습니다.
 - 데이터를 쓰는 중에는 읽을 수 없습니다.

ORACLE

읽기 일관성

데이터베이스 사용자는 다음 두 방법으로 데이터베이스에 액세스합니다.

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

읽기 일관성이 필요한 이유는 다음과 같습니다.

- 데이터베이스를 읽거나 쓰는 사용자에게 일관성 있는 데이터 뷰를 제공하기 위해
- 데이터를 읽는 사용자가 변경 중인 데이터를 보지 못하도록 하기 위해
- 데이터를 쓰는 사용자가 일관성 있는 방식으로 데이터베이스 데이터를 변경하도록 하기 위해
- 한 사용자가 변경한 데이터와 다른 사용자가 변경한 데이터 간의 충돌을 방지하기 위해

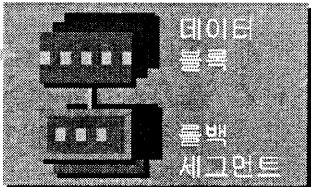
읽기 일관성은 각 사용자가 DML 작업을 시작하기 전에 마지막 커밋된 상태의 데이터를 볼 수 있도록 합니다.

읽기 일관성 구현

사용자 A

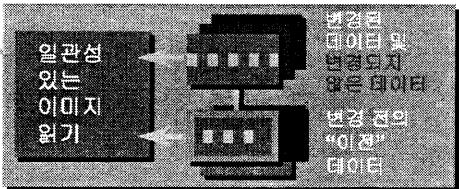


```
UPDATE employees  
SET salary = 7000  
WHERE last_name = 'Goyal';
```



사용자 B

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



ORACLE

읽기 일관성 구현

읽기 일관성은 자동으로 구현되며 데이터베이스 복사본 중 일부를 언두 세그먼트에 저장합니다.

데이터베이스가 삽입, 개신 또는 삭제될 경우 Oracle server는 변경 이전 데이터의 복사본을 가져와 언두 세그먼트에 기록합니다.

내용을 변경한 사용자 이외의 모든 사용자는 변경 이전의 데이터베이스인 른백 세그먼트의 데이터 “스냅샷”을 보게 됩니다.

데이터베이스의 변경 내용이 커밋되기 전까지는 데이터를 수정하는 사용자만 변경된 데이터베이스를 볼 수 있으며 다른 사용자는 언두 세그먼트의 스냅샷을 보게 됩니다. 따라서 데이터를 읽는 사용자는 현재 변경 중인 데이터가 아닌 일관성 있는 데이터를 볼 수 있습니다.

DML 문이 커밋되면 SELECT 문을 실행하는 모든 사용자가 데이터베이스의 변경 내용을 볼 수 있으며 언두 세그먼트 파일의 이전 데이터가 있던 공간은 재사용할 수 있도록 비워집니다.

트랜잭션이 롤백되면 변경이 취소됩니다.

- 언두 세그먼트에 있는 데이터 원본인 이전 버전이 테이블에 다시 기록됩니다.
- 모든 사용자는 트랜잭션 시작 이전의 데이터베이스를 보게 됩니다.

장금

오라클 데이터베이스에서 잠금은 다음과 같은 기능을 가집니다.

- 동시에 수행되는 트랜잭션 간에 파괴적인 상호 작용을 방지합니다.
- 사용자 작업이 필요 없습니다.
- 자동으로 최대한 낮은 레벨의 제한을 사용합니다.
- 트랜잭션 실행 기간 동안 유지됩니다.
- 명시적(**explicit**) 잠금과 암시적(**implicit**) 잠금이라는 두 가지 유형이 있습니다.

ORACLE

8-45

Copyright © Oracle Corporation, 2001. All rights reserved.

잠금이란?

잠금은 사용자 객체(테이블 또는 행) 또는 사용자가 볼 수 없는 시스템 객체(공유 데이터 구조 및 데이터 딕셔너리 행)와 같은 동일한 자원을 액세스하는 트랜잭션 간에 발생할 수 있는 파괴적인 상호 작용을 방지하는 방식입니다.

오라클 데이터베이스에서 데이터를 잠그는 방법

오라클은 자동으로 수행되며 사용자 작업이 필요 없습니다. 암시적 잠금은 요청된 작업에 따라 필요한 경우 SQL 문에서 발생합니다. 암시적 잠금은 SELECT를 제외한 모든 SQL 문에서 발생합니다.

사용자가 수동으로 데이터를 잠글 수도 있으며 이를 명시적 잠금이라고 합니다.

암시적(Implicit) 잠금

- 두 가지 잠금 모드
 - 배타적(exclusive): 다른 사용자가 액세스하지 못하도록 잠금
 - 공유(share): 다른 사용자의 액세스를 허용
- 높은 레벨의 데이터 동시성
 - DML: 테이블은 공유 잠금, 행은 배타적 잠금
 - 질의: 잠금 필요 없음
 - DDL: 객체 정의 보호
- 잠금은 커밋 또는 롤백이 수행될 때까지 유효합니다.

ORACLE

8-46

Copyright © Oracle Corporation, 2001. All rights reserved.

DML 잠금

DML(데이터 조작어) 작업을 수행할 때, Oracle server는 DML 잠금을 통해 데이터 동시성을 제공합니다. DML 잠금에는 두 가지 레벨이 있습니다.

- 공유 잠금(share lock)은 DML 작업 도중 테이블 레벨에서 자동으로 수행됩니다. 공유 잠금 모드에서는 여러 트랜잭션이 동일한 자원에 대해 공유 잠금을 획득할 수 있습니다.
- 배타적 잠금(exclusive lock)은 DML 문에서 수정하는 각 행에 자동으로 수행됩니다. 배타적 잠금은 트랜잭션이 커밋되거나 롤백될 때까지 다른 트랜잭션에서 해당 행을 변경하지 못하도록 합니다. 이 잠금은 다른 사용자가 동시에 동일한 행을 수정하지 못하도록 하며 아직 커밋하지 않은 변경 내용을 다른 사용자가 덮어 쓰지 못하도록 합니다.
- DDL 잠금은 테이블과 같은 데이터베이스 객체를 수정할 경우 발생합니다.

요약

이 단원에서는 **DML** 문 사용 방법과 트랜잭션 제어 방법에 대해 배웠습니다.

명령문	설명
INSERT	데이터에 새 행 추가
UPDATE	데이터의 기존 행 수정
DELETE	데이터에서 기존 행 제거
MERGE	조건에 따라 데이터를 데이터에 삽입 또는 갱신
COMMIT	보류 중인 모든 변경 내용을 영구히 저장
SAVEPOINT	저장점 표시자까지 롤백하는 데 사용
ROLLBACK	보류 중인 데이터 변경 내용을 모두 버림

ORACLE

요약

이 단원에서는 INSERT, UPDATE 및 DELETE 문을 사용하여 오라클 데이터베이스에서 데이터를 조작하는 방법을 설명했습니다. COMMIT, SAVEPOINT 및 ROLLBACK 문을 사용하여 데이터 변경 내용을 제어합니다.

Oracle server는 항상 일관성 있는 데이터 뷰를 제공합니다.

잠금은 암시적(implicit) 잠금이나 명시적(explicit) 잠금 중 하나에 속합니다.

연습 8 개요

이 연습에서는 다음 내용을 다룹니다.

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

ORACLE

3-48

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 8 개요

이 연습에서는 MY_EMPLOYEE 테이블에 행을 추가하고 이 테이블에서 데이터를 생성 및 삭제하며 해당 트랜잭션을 제어합니다.

연습 8

MY_EMPLOYEE 테이블에 데이터를 삽입하십시오.

1. lab8_1.sql 스크립트의 명령문을 실행하여 실습에 사용할 MY_EMPLOYEE 테이블을 생성하십시오.
2. MY_EMPLOYEE 테이블의 구조를 표시하여 열 이름을 식별하십시오.

Name	Null?	Type
ID	NOT NULL	NUMBER(4)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
USERID		VARCHAR2(8)
SALARY		NUMBER(9,2)

3. 다음 예제 데이터의 첫번째 행을 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

4. 위의 목록에 있는 예제 데이터의 두번째 행을 MY_EMPLOYEE 테이블에 추가하십시오. 이번에는 INSERT 절에 열을 명시적으로 나열하십시오.
5. 테이블에 추가한 항목을 확인하십시오.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860

연습 8(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.
7. 작성한 스크립트의 insert 문을 실행하여 예제 데이터의 그 다음 두 행을 테이블에 추가하십시오.
8. 테이블에 추가한 항목을 확인하십시오.

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

9. 추가한 데이터를 영구히 저장하십시오.

MY_EMPLOYEE 테이블의 데이터를 생성 및 삭제하십시오.

10. 사원 3의 성을 Drexler로 변경하십시오.
11. 급여가 900 미만인 모든 사원의 급여를 1000으로 변경하십시오.
12. 테이블의 변경 내용을 확인하십시오.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
2	Dancs	Betty	bdancs	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000

13. MY_EMPLOYEE 테이블에서 Betty Dancs를 삭제하십시오.

14. 테이블의 변경 내용을 확인하십시오.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000

연습 8(계속)

15. 보류 중인 변경 내용을 모두 커밋하십시오.
- MY_EMPLOYEE 테이블에 대한 데이터 트랜잭션을 제어하십시오.
16. 6 단계에서 작성한 스크립트의 명령문을 수정하여 예제 데이터의 마지막 행을 테이블에 추가하고 스크립트의 명령문을 실행하십시오.
17. 테이블에 추가한 항목을 확인하십시오.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000
5	Ropeburn	Audrey	aropebur	1550

18. 트랜잭션 수행 중에 저장점을 표시하십시오.
19. 테이블의 내용을 모두 삭제하십시오.
20. 테이블 내용이 비어 있는지 확인하십시오.
21. 이전의 INSERT 작업은 버리지 말고 최근의 DELETE 작업만 버리십시오.
22. 새 행이 그대로 있는지 확인하십시오.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000
5	Ropeburn	Audrey	aropebur	1550

23. 추가한 데이터를 영구히 저장하십시오.

읽기 일관성 예제

출력	시간	세션 1	세션 2
24000	t1	SELECT salary FROM employees WHERE last_name='King';	
	t2		UPDATE employees SET salary=salary+10000 WHERE last_name='King';
24000	t3	SELECT salary FROM employees WHERE last_name='King';	
	t4		COMMIT;
34000	t5	SELECT salary FROM employees WHERE last_name='King';	

ORACLE

9

테이블 생성 및 관리

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 기본 데이터베이스 객체 설명
- 테이블 생성
- 열 정의를 지정할 때 사용 가능한 데이터 유형 설명
- 테이블 정의 변경
- 테이블 삭제, 이름 변경 및 절단

ORACLE

단원 목표

이 단원에서는 기본 데이터베이스 객체 및 객체의 상호 관계를 설명하고 테이블을 생성, 변경 및 삭제하는 방법을 설명합니다.

데이터베이스 객체

객체	설명
테이블	기본 저장 단위며 행과 열로 구성됩니다.
뷰	논리적으로 하나 이상의 테이블에 있는 데이터의 부분 집합을 나타냅니다.
시퀀스	숫자 값 생성기입니다.
인덱스	질의의 성능을 향상시킵니다.
동의어	객체에 다른 이름을 제공합니다.

ORACLE

9-3

Copyright © Oracle Corporation, 2001. All rights reserved.

데이터베이스 객체

오라클 데이터베이스에는 여러 데이터 구조가 포함될 수 있습니다. 각 구조는 데이터베이스 설계 시 윤곽이 정해져서 데이터베이스 개발의 구축 단계에서 생성될 수 있도록 해야 합니다.

- 테이블: 데이터를 저장합니다.
- 뷰: 하나 이상의 테이블에 있는 데이터의 부분 집합입니다.
- 시퀀스: 숫자 값 생성기입니다.
- 인덱스: 질의의 성능을 향상시킵니다.
- 동의어: 객체에 다른 이름을 제공합니다.

Oracle9i 테이블 구조

- 테이블은 언제든지 생성할 수 있으며 심지어 사용자가 데이터베이스를 사용하는 중에도 생성할 수 있습니다.
- 테이블 크기는 데이터베이스에 전체적으로 할당된 공간만큼 정의되므로 지정할 필요가 없지만 시간이 경과하면서 테이블에 어느 정도의 공간이 사용될지 예측하는 것은 중요합니다.
- 테이블 구조는 온라인으로 수정할 수 있습니다.

참고: 사용 가능한 데이터베이스 객체가 더 있지만 이 과정에서는 설명하지 않습니다.

이름 지정 규칙

테이블 이름 및 열 이름:

- 문자로 시작해야 합니다.
- 1자부터 30자까지 가능합니다.
- A-Z, a-z, 0-9, _, \$, #만 포함해야 합니다.
- 동일한 사용자가 소유한 다른 객체의 이름과 중복되지 않아야 합니다.
- Oracle server의 예약어가 아니어야 합니다.

ORACLE

이름 지정 규칙

오라클 데이터베이스 객체의 이름 지정 표준 규칙에 따라 데이터베이스 테이블과 열 이름을 지정합니다.

- 테이블 이름과 열 이름은 문자로 시작해야 하며 1자부터 30자까지 가능합니다.
- 이름에는 A-Z, a-z, 0-9, _(밑줄), \$, #(적합한 문자지만 사용하지 않는 것이 좋음) 문자만 사용할 수 있습니다.
- 이름은 동일한 Oracle server 사용자가 소유한 다른 객체의 이름과 중복되지 않아야 합니다.
- 이름은 Oracle server 예약어가 아니어야 합니다.

이름 지정 지침

테이블 및 다른 데이터베이스 객체에는 기술적인 이름을 사용합니다.

참고: 이름은 대소문자를 구분하지 않습니다. 예를 들어, EMPLOYEES는 eMPloyees 또는 eMpLOYEES와 동일한 이름으로 취급됩니다.

자세한 내용은 *Oracle9i SQL Reference*, “Object Names and Qualifiers”를 참조하십시오.

CREATE TABLE 문

- 다음 사항이 필요합니다.

- CREATE TABLE 권한
- 저장 영역

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

- 다음을 지정합니다.

- 테이블 이름
- 열의 이름, 데이터 유형 및 크기

ORACLE

9-5

Copyright © Oracle Corporation, 2001. All rights reserved.

CREATE TABLE 문

SQL CREATE TABLE 문을 실행하여 데이터를 저장할 테이블을 생성합니다. 이 명령문은 DDL(데이터 정의어) 문의 하나로 나중 단원에서 설명합니다. DDL 문은 Oracle9i 데이터베이스 구조의 생성, 수정 또는 제거에 사용되는 SQL 문의 일부입니다. 이러한 명령문은 데이터베이스에 직접 영향을 주며 또한 데이터 딕셔너리에 정보를 기록합니다.

테이블을 생성하려면 사용자에게 CREATE TABLE 권한이 필요하며 객체를 생성할 저장 영역이 필요합니다. 데이터베이스 관리자는 사용자에게 권한을 부여하기 위해 DCL(데이터 케어어) 문을 사용하는데 DCL 문에 대해서는 나중 단원에서 설명합니다.

구문 설명:

<i>schema</i>	소유자 이름과 동일합니다.
<i>table</i>	테이블 이름입니다.
<i>DEFAULT expr</i>	INSERT 문에 값이 생략될 경우 사용될 기본값을 지정합니다.
<i>column</i>	열 이름입니다.
<i>datatype</i>	열의 데이터 유형 및 길이입니다.

다른 사용자의 테이블 참조

- 다른 사용자에게 속한 테이블은 해당 사용자의 스키마에 없습니다.
- 다른 사용자의 테이블을 사용할 때는 소유자 이름을 접두어로 붙여야 합니다.

ORACLE

9-6

Copyright © Oracle Corporation, 2001. All rights reserved.

다른 사용자의 테이블 참조

스키마는 객체의 collection이고 스키마 객체는 데이터베이스의 데이터를 직접 참조하는 논리 구조입니다. 스키마 객체는 테이블, 뷰, 동의어, 시퀀스, 내장 프로시저, 인덱스, 클러스터 및 데이터베이스 링크를 포함합니다.

테이블이 이를 사용 중인 사용자에게 속하지 않은 경우 테이블에 소유자 이름을 접두어로 붙여야 합니다. 예를 들어, USER_B라는 이름의 스키마가 있고, USER_B에 EMPLOYEES 테이블이 있는 경우 이 테이블에서 데이터를 검색하려면 다음과 같이 지정합니다.

```
SELECT *
FROM    user_b.employees;
```

DEFAULT 옵션

- 삽입을 수행할 때 사용할 열의 기본값을 지정합니다.

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

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

ORACLE

DEFAULT 옵션

DEFAULT 옵션을 사용하여 열에 기본값을 부여할 수 있습니다. 이 옵션을 사용하면 열 값이 없는 행을 삽입할 경우 열에 널 값이 입력되는 것을 방지합니다. 리터럴, 표현식 또는 SYSDATE 및 USER와 같은 SQL 함수는 기본값으로 사용할 수 있지만 NEXTVAL 또는 CURRVAL과 같은 다른 열 이름이나 의사 열은 기본값으로 사용할 수 없습니다. 기본 표현식은 해당 열의 데이터 유형과 일치해야 합니다.
참고: CURRVAL 및 NEXTVAL은 나중에 설명합니다.

테이블 생성

- 테이블을 생성합니다.

```
CREATE TABLE dept
  (deptno NUMBER(2),
   dname  VARCHAR2(14),
   loc    VARCHAR2(13));
Table created.
```

- 테이블 생성을 확인합니다.

```
DESCRIBE dept
```

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

ORACLE

테이블 생성

슬라이드 예제는 DEPTNO, DNAME 및 LOC의 세 열로 구성된 DEPT 테이블을 생성하고 DESCRIBE 명령을 실행하여 해당 테이블의 생성을 확인합니다.

테이블 생성 명령문은 DDL 문이므로 이 명령문을 실행하면 자동 ~~파티~~이 수행됩니다.

오라클 데이터베이스의 테이블

- 사용자 테이블
 - 사용자가 생성 및 유지 관리하는 테이블의 **collection**입니다.
 - 사용자 정보를 포함합니다.
- 데이터 딕셔너리
 - **Oracle Server**가 생성 및 유지 관리하는 테이블의 **collection**입니다.
 - 데이터베이스 정보를 포함합니다.

ORACLE

9-9

Copyright © Oracle Corporation, 2001. All rights reserved.

오라클 데이터베이스의 테이블

사용자 테이블은 EMPLOYEES와 같이 사용자가 생성하는 테이블입니다. 데이터 딕셔너리로 알려진 오라클 데이터베이스는 다른 테이블 및 뷰의 collection을 포함합니다. 이 collection은 Oracle server가 생성 및 유지 관리하며 데이터베이스에 관한 정보를 포함합니다.

SYS 사용자는 모든 데이터 딕셔너리 테이블을 소유합니다. 기본 테이블의 정보는 이해하기 어렵기 때문에 사용자는 기본 테이블을 거의 액세스하지 않으며 일반적으로 이해하기 쉬운 형식의 정보를 제공하는 데이터 딕셔너리 뷰를 액세스합니다. 데이터 딕셔너리에 저장된 정보는 Oracle server의 사용자 이름, 사용자에게 부여된 권한, 데이터베이스 객체 이름, 테이블 제약 조건, 감사 정보 등을 포함합니다.

데이터 딕셔너리 뷰에는 다음 네 가지 범주가 있는데 각 범주는 사용 목적을 반영하는 접두어를 포함합니다.

접두어	설명
USER_	이 뷰는 사용자가 소유하는 객체에 관한 정보를 포함합니다.
ALL_	이 뷰는 사용자가 액세스할 수 있는 모든 테이블(객체 테이블 및 관계형 테이블)에 관한 정보를 포함합니다.
DBA_	이 뷰는 제한된 뷰로서 DBA 를 활동 받은 사용자만 액세스할 수 있습니다.
V\$	이들 뷰는 데이터베이스 서버 성능, 메모리 및 임금에 대한 동적 성능 뷰입니다.

데이터 딕셔너리 질의

- 사용자가 소유한 테이블의 이름을 봅니다.

```
SELECT table_name  
FROM user_tables ;
```

- 사용자가 소유한 개별 객체 유형을 봅니다.

```
SELECT DISTINCT object_type  
FROM user_objects ;
```

- 사용자가 소유한 테이블, 뷰, 동의어 및 시퀀스를 봅니다.

```
SELECT *  
FROM user_catalog ;
```

ORACLE

데이터 딕셔너리 질의

데이터 딕셔너리 테이블을 질의하여 사용자가 소유한 다양한 데이터베이스 객체를 볼 수 있습니다.
자주 사용되는 데이터 딕셔너리 테이블은 다음과 같습니다.

- USER_TABLES
- USER_OBJECTS
- USER_CATALOG

참고: USER_CATALOG은 CAT라는 동의어를 가지고 있는데 SQL 문에서는 USER_CATALOG 대신에
이 동의어를 사용할 수 있습니다.

```
SELECT *  
FROM CAT ;
```

데이터 유형

데이터 유형	설명
VARCHAR2 (<i>size</i>)	가변 길이 문자 데이터
CHAR (<i>size</i>)	고정 길이 문자 데이터
NUMBER (<i>p, s</i>)	가변 길이 숫자 데이터
DATE	날짜 및 시간 값
LONG	최대 2GB의 가변 길이 문자 데이터
CLOB	최대 4GB의 문자 데이터
RAW 및 LONG RAW	원시 이진 데이터
BLOB	최대 4GB의 이진 데이터
BFILE	외부 파일에 저장된 이진 데이터(최대 4GB)
ROWID	테이블에서 행의 고유 주소를 나타내는 64진수

ORACLE

8-11

Copyright © Oracle Corporation, 2001. All rights reserved.

데이터 유형

데이터 유형	설명
VARCHAR2 (<i>size</i>)	가변 길이 문자 데이터며 <i>size</i> 에 최대 길이를 지정해야 합니다 (<i>size</i> 의 최소값: 1, <i>size</i> 의 최대값: 4000).
CHAR [(<i>size</i>)]	길이의 바이트 수가 <i>size</i> 인 고정 길이 문자 데이터입니다(<i>size</i> 의 기본값 및 최소값: 1, <i>size</i> 의 최대값: 2000).
NUMBER [(<i>p, s</i>)]	십진 자릿수가 <i>p</i> 고 소수점 이하 자릿수가 <i>s</i> 인 숫자입니다. 십진 자릿수는 십진수의 총 개수고 소수점 이하 자릿 수는 소수점의 오른쪽에 있는 숫자 개수입니다. 십진 자릿수의 범위는 1부터 38까지고 소수점 이하 자릿수의 범위는 -84부터 127까지입니다.
DATE	B.C. 4712년 1월 1일과 A.D. 9999년 12월 31일 사이에 있는 초 단위까지의 날짜 및 시간 값입니다.
LONG	최대 2GB의 가변 길이 문자 데이터입니다.
CLOB	최대 4GB의 문자 데이터입니다.

데이터 유형(계속)

데이터 유형	설명
RAW(<i>size</i>)	길이가 <i>size</i> 인 원시 이진 데이터이며 <i>size</i> 에 최대 길이를 지정해야 합니다(<i>size</i> 의 최대값: 2000).
LONG RAW	최대 2GB의 가변 길이 원시 이진 데이터입니다.
BLOB	최대 4GB의 이진 데이터입니다.
BFILE	외부 파일에 저장된 이진 데이터입니다(최대 4GB).
ROWID	테이블에서 행(row)의 고유 주소를 나타내는 64진수입니다.

- 서브 쿼리를 사용하여 테이블을 생성한 경우 LONG 열은 복사되지 않습니다.
- LONG 열은 GROUP BY 또는 ORDER BY 절에 포함시킬 수 없습니다.
- LONG 열은 테이블 당 하나만 사용할 수 있습니다.
- LONG 열에는 제약 조건을 정의할 수 없습니다.
- LONG 열보다는 CLOB 열을 사용하는 것이 좋습니다.

Date**Time** 데이터 유형

Oracle9i에서의 Datetime 기능 향상

- 새로운 **Datetime** 데이터 유형이 도입되었습니다.
- 새로운 데이터 유형 저장이 가능합니다.
- 시간대 및 지역 시간대에 관한 기능이 향상되었습니다.

데이터 유형	설명
TIMESTAMP	소수점 이하 초까지 포함하는 날짜
INTERVAL YEAR TO MONTH	연 수 및 개월 수로 기간을 저장
INTERVAL DAY TO SECOND	날짜 수, 시간 수, 분 수, 초 수로 기간을 저장

ORACLE

다른 Date**Time** 데이터 유형

데이터 유형	설명
TIMESTAMP	시간을 소수점 이하 초까지 있는 날짜로 저장할 수 있습니다. 이 데이터 유형에는 몇 가지 변형이 있습니다.
INTERVAL YEAR TO MONTH	시간을 연 수와 개월 수로 저장할 수 있습니다. 두 datetime 값에서 연도와 월만 중요한 값일 경우 이들 두 값의 차이를 나타내는데 사용됩니다.
INTERVAL DAY TO SECOND	시간을 날짜 수, 시간 수, 분 수, 초 수로 저장할 수 있습니다. 두 datetime 값의 정확한 차이를 나타내는데 사용됩니다.

DateType 데이터 유형

- **TIMESTAMP** 데이터 유형은 **DATE** 데이터 유형을 확장한 것입니다.
- **DATE** 데이터 유형의 연도, 월, 일뿐 아니라 시, 분, 초 및 소수점 이하 초 값까지 저장합니다.
- **TIMESTAMP** 데이터 유형은 다음과 같이 지정합니다.

```
TIMESTAMP [(fractional_seconds_precision)]
```

ORACLE

9-14

Copyright © Oracle Corporation, 2001. All rights reserved.

DateType 데이터 유형

`fractional_seconds_precision`은 선택 사항으로 SECOND datetime 필드의 소수점 이하 부분의 자릿수를 지정하며 0부터 9까지 범위의 숫자가 올 수 있습니다. 기본값은 6입니다.

예제

```
CREATE TABLE new_employees
(employee_id NUMBER,
 first_name VARCHAR2(15),
 last_name VARCHAR2(15),
 ...
 start_date TIMESTAMP(7),
 ...);
```

앞의 예제에서 데이터 유형이 `TIMESTAMP`인 `start_date` 열이 있는 `NEW_EMPLOYEES` 테이블을 생성합니다. 정밀도 '7'은 초의 소수점 이하 자릿수를 나타내며 지정되지 않은 경우 기본값은 '6'입니다.

두 행이 `NEW_EMPLOYEES` 테이블에 삽입되었다고 가정합니다. 출력을 보면 날짜와 시간이 표시되는 차이를 알 수 있습니다. `DATE` 데이터 유형은 기본적으로 DD-MON-RR 형식으로 표시됩니다.

```
SELECT start_date
FROM   new_employees;
```



```
17-JUN-87 12.00.00.0000000 AM
21-SEP-89 12.00.00.0000000 AM
```

TIMESTAMP WITH TIME ZONE 데이터 유형

- TIMESTAMP WITH TIME ZONE은 TIMESTAMP 값에 시간대 변위 값을 포함시킨 TIMESTAMP의 변형입니다.
- 시간대 변위 값은 지역 시간과 UTC 사이의 차이를 시와 분으로 나타낸 것입니다.

```
TIMESTAMP [(fractional_seconds_precision)]  
WITH TIME ZONE
```

ORACLE

9-15

Copyright © Oracle Corporation, 2001. All rights reserved.

Datetime 데이터 유형

UTC는 세계 협정 시(Coordinated Universal Time)를 나타내며 이전의 그리니치 표준시(GMT: Greenwich Mean Time)를 말합니다. 두 TIMESTAMP WITH TIME ZONE 값이 UTC에서 동일한 시간을 나타낼 경우에 이 두 값은 데이터에 저장된 TIME ZONE 오프셋과 상관 없이 동일한 것으로 간주됩니다.

TIMESTAMP WITH TIME ZONE은 시간대 정보를 저장할 수 있으므로 특히 지리적으로 서로 떨어져 있는 곳의 날짜 정보를 수집 및 조정을 거쳐 기록해야 하는 경우 적합합니다.

예를 들어

```
TIMESTAMP '1999-04-15 8:00:00 -8:00'
```

은 다음과 같습니다.

```
TIMESTAMP '1999-04-15 11:00:00 -5:00'
```

즉, 미국 태평양 표준시로 오전 8시는 미국 동부 표준시로 오전 11시와 같습니다.

다음과 같이 지정할 수도 있습니다.

```
TIMESTAMP '1999-04-15 8:00:00 US/Pacific'
```

참고: fractional_seconds_precision은 선택 사항이며 SECOND datetime 필드의 소수점 이하 부분의 자릿수를 지정하며 0부터 9까지 범위의 숫자가 올 수 있습니다. 기본값은 6입니다.

TIMESTAMP WITH LOCAL TIME 데이터 유형

- **TIMESTAMP WITH LOCAL TIME ZONE**은 **TIMESTAMP** 값에 시간대 변위 값을 포함시킨 **TIMESTAMP**의 변형입니다.
- 데이터베이스에 저장되는 데이터는 데이터베이스의 시간대에 맞게 정규화됩니다.
- 시간대 변위 값은 열 데이터의 일부로 저장되지 않습니다. 오라클에서 사용자의 지역 세션 시간대에 맞게 해당 데이터를 반환합니다.
- **TIMESTAMP WITH LOCAL TIME ZONE** 데이터 유형은 다음과 같이 지정합니다.

```
TIMESTAMP [(fractional_seconds_precision)]  
WITH LOCAL TIME ZONE
```

ORACLE

9-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Date/Time 데이터 유형

TIMESTAMP WITH TIME ZONE과 달리 **TIMESTAMP WITH LOCAL TIME ZONE** 유형의 열은 기본 키 또는 고유 키의 일부로 지정할 수 있습니다. 시간대 변위 값은 지역 시간과 UTC 사이의 차이를 시와 분으로 나타낸 것입니다. **TIMESTAMP WITH LOCAL TIME ZONE**에 사용되는 리터럴은 없습니다.

참고: **fractional_seconds_precision**은 선택 사항으로 SECOND datatype 필드의 소수점 이하 부분의 자릿수를 지정하며 0부터 9까지 범위의 숫자가 올 수 있습니다. 기본값은 6입니다.

예제

```
CREATE TABLE time_example  
(order_date TIMESTAMP WITH LOCAL TIME ZONE);  
  
INSERT INTO time_example VALUES('15-NOV-00 09:34:28 AM');  
  
SELECT *  
FROM   time_example;  
order_date  
-----  
15-NOV-00 09.34.28.000000 AM
```

TIMESTAMP WITH LOCAL TIME ZONE 유형은 클라이언트 시스템의 시간대를 사용하여 날짜 및 시간을 표시하는 2 계층 응용 프로그램에 적합합니다.

INTERVAL YEAR TO MONTH 데이터 유형

- INTERVAL YEAR TO MONTH는 YEAR 및 MONTH datetime 필드를 사용하여 기간을 저장합니다.

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

INTERVAL '123-2' YEAR(3) TO MONTH
123년 2개월의 기간을 나타냅니다.

INTERVAL '123' YEAR(3)
123년 0개월의 기간을 나타냅니다.

INTERVAL '300' MONTH(3)
300개월의 기간을 나타냅니다.

INTERVAL '123' YEAR
기본 자릿수는 2인데 '123'에 세 자리가 있기 때문에 오류가 반환됩니다.

ORACLE

3-17

Copyright © Oracle Corporation, 2001. All rights reserved.

INTERVAL YEAR TO MONTH 데이터 유형

INTERVAL YEAR TO MONTH는 YEAR 및 MONTH datetime 필드를 사용하여 기간을 저장합니다.
두 datetime 값에서 연도와 월만이 중요한 값일 경우 이들의 차이를 나타낼 때 INTERVAL YEAR TO MONTH를 사용합니다. 예를 들어, 이 값을 사용하여 앞으로 120개월 후의 날짜를 통보하도록 설정하거나 특정 날짜로부터 6개월이 경과했는지를 확인할 수 있습니다.

INTERVAL YEAR TO MONTH는 다음과 같이 지정합니다.

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

구문 설명:

year_precision YEAR datetime 필드의 자릿수입니다. year_precision의 기본값은 2입니다.

예제

```
CREATE TABLE time_example2
(loan_duration INTERVAL YEAR (3) TO MONTH);

INSERT INTO time_example2 (loan_duration)
VALUES (INTERVAL '120' MONTH(3));

SELECT TO_CHAR( sysdate+loan_duration, 'dd-mon-yyyy')
FROM   time_example2;
--today's date is 26-Sep-2001
```

TO_CHAR(SY)

26-sep-2011

INTERVAL DAY TO SECOND 데이터 유형

- **INTERVAL DAY TO SECOND**는 기간을 날짜 수, 시간 수, 분수 및 초 수로 저장합니다.

**INTERVAL DAY [(day_precision)]
TO SECOND [(fractional seconds precision)]**

INTERVAL '4 5:12:10.222' DAY TO SECOND(3)

4월 5시간 12분 10.222초를 나타냅니다.

INTERVAL '7' DAY

7일을 나타냅니다.

INTERVAL '180' DAY(3)

180일을 나타냅니다.

ORACLE

9-18

Copyright © Oracle Corporation, 2001. All rights reserved.

INTERVAL DAY TO SECOND 데이터 유형

INTERVAL DAY TO SECOND는 기간을 날짜 수, 시간 수, 분수, 초수로 저장합니다.

두 **datetime** 값 사이의 정확한 차이를 나타내려면 **INTERVAL DAY TO SECOND**를 사용합니다. 예를 들어, 이 값을 사용하여 앞으로 36시간 후의 시간을 통보하도록 설정하거나 경주에서 시작 및 종료 사이의 시간 간격을 기록할 수 있습니다. 몇 년에 걸쳐 오랜 시간 간격을 높은 정밀도로 나타내려면 날짜 부분에 큰 값을 사용하면 됩니다.

INTERVAL DAY TO SECOND는 다음과 같이 지정합니다.

```
INTERVAL DAY [(day_precision)]  
TO SECOND [(fractional seconds precision)]
```

구문 설명·

day precision

DAY **datetime** 필드의 자릿수입니다. 허용되는
값은 0부터 9까지이며 기본값은 2입니다.

`fractional_seconds_precision` SECOND datatype 필드의 소수점 이하 부분의 자릿수입니다. 허용되는 값은 0부터 9까지이며 기본값은 6입니다.

INTERVAL DAY TO SECOND 데이터 유형

- * INTERVAL DAY TO SECOND는 기간을 날짜 수, 시간 수, 분 수, 초 수로 저장합니다.

```
INTERVAL '4 5:12:10.222' DAY TO SECOND(3)  
4일 5시간 12분 10.222초를 나타냅니다.
```

```
INTERVAL '4 5:12' DAY TO MINUTE  
4일 5시간 12분을 나타냅니다.
```

```
INTERVAL '400 5' DAY(3) TO HOUR  
400일 5시간을 나타냅니다.
```

```
INTERVAL '11:12:10.2222222' HOUR TO SECOND(7)  
11시간 12분 10.2222222초를 나타냅니다.
```

ORACLE

9-19

Copyright © Oracle Corporation, 2001. All rights reserved.

INTERVAL DAY TO SECOND 데이터 유형

예제

```
CREATE TABLE time_example3  
(day_duration INTERVAL DAY (3) TO SECOND);  
  
INSERT INTO time_example3 (day_duration)  
VALUES (INTERVAL '180' DAY(3));  
  
SELECT sysdate + day_duration "Half Year"  
FROM   time_example3;           --today's date is 26-Sep-2001
```

Half Year

25-MAR-02

서브 쿼리 구문을 사용한 테이블 생성

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

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

- 지정한 열 수를 서브 쿼리 열 수와 일치시켜야 합니다.
- 열 이름 및 기본값을 사용하여 열을 정의합니다.

ORACLE

9-20

Copyright © Oracle Corporation, 2001. All rights reserved.

다른 테이블의 행으로부터 테이블 생성

테이블을 생성하는 두 번째 방법은 AS *subquery* 절을 적용하여 테이블 생성과 서브 쿼리에 의해 반환되는 행 삽입을 모두 수행하는 것입니다.

구문 설명:

table	테이블 이름입니다.
column	열 이름, 기본값 및 무결성 제약 조건입니다.
subquery	새 테이블에 삽입될 행의 집합을 정의하는 SELECT 문입니다.

지침

- 지정한 열 이름을 사용하여 테이블을 생성하며 SELECT 문에 의해 검색된 행을 테이블에 삽입합니다.
- 열 정의에는 열 이름과 기본값만 포함시킬 수 있습니다.
- 열 명세가 제공되는 경우 열 수는 서브 쿼리의 SELECT 목록에 있는 열 수와 동일해야 합니다.
- 열 명세가 제공되지 않는 경우 해당 테이블의 열 이름은 서브 쿼리의 열 이름과 동일합니다.
- 무결성 규칙은 새로운 테이블에 전달되지 않으며 열 데이터 유형 정의만 전달됩니다.

서브 쿼리를 사용한 테이블 생성

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

Table created.

```
DESCRIBE dept80
```

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

ORACLE

9-21

Copyright © Oracle Corporation, 2001. All rights reserved.

다른 테이블의 행으로부터 테이블 생성(계속)

슬라이드 예제는 부서 80에서 일하는 모든 사원에 대한 자세한 정보를 포함하는 DEPT80 테이블을 생성합니다. DEPT80 테이블의 데이터는 EMPLOYEES 테이블에서 가져옵니다.

iSQL*Plus DESCRIBE 명령을 사용하여 데이터베이스 테이블의 유무 및 열 정의를 확인할 수 있습니다.

표현식을 선택할 때 열 별칭을 지정해야 합니다. 표현식 SALARY*12에는 별칭 ANNSAL이 지정되었습니다. 별칭이 없으면 다음 오류가 생성됩니다.

ERROR at line 3:

ORA-00998: must name this expression with a column alias

ALTER TABLE 문

다음 작업에 ALTER TABLE 문을 사용합니다.

- 새 열 추가
- 기존 열 수정
- 새 열의 기본값 정의
- 열 삭제

ORACLE

9-22

Copyright © Oracle Corporation, 2001. All rights reserved.

ALTER TABLE 문

테이블 생성 후에 열을 빼뜨렸거나, 열 정의를 변경해야 하거나, 열을 제거해야 할 상황이 발생하여 테이블 구조를 변경해야 할 경우가 있습니다. ALTER TABLE 문을 사용하면 테이블 구조를 변경할 수 있습니다.

ALTER TABLE 문

ALTER TABLE 문을 사용하여 열을 추가, 수정 또는 삭제할 수 있습니다.

```
ALTER TABLE table
ADD          (column datatype [DEFAULT expr]
[, column datatype] ...);
```

```
ALTER TABLE table
MODIFY       (column datatype [DEFAULT expr]
[, column datatype] ...);
```

```
ALTER TABLE table
DROP         (column);
```

ORACLE

9-23

Copyright © Oracle Corporation, 2001. All rights reserved.

ALTER TABLE 문(계속)

ALTER TABLE 문을 사용하여 테이블에 열을 추가, 수정 및 삭제할 수 있습니다.

구문 설명:

<i>table</i>	테이블 이름입니다.
<i>ADD MODIFY DROP</i>	수정 유형입니다.
<i>column</i>	새 열의 이름입니다.
<i>datatype</i>	새 열의 데이터 유형 및 길이입니다.
<i>DEFAULT expr</i>	새 열의 기본값을 지정합니다.

참고: 슬라이드는 ALTER TABLE의 구문을 축약하여 나타낸 것입니다. ALTER TABLE에 대한 자세한 내용은 나중 단원에서 설명합니다.

열 추가

DEPT80

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
149	Zlotkey	126000	29-JAN-00
174	Abel	132000	11-MAY-96
176	Taylor	103200	24-MAR-98

새 열

JOB_ID

“DEPT80
테이블에
새 열을
추가합니다.”

DEPT80

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

ORACLE

열 추가

그래픽은 JOB_ID 열을 DEPT80 테이블에 추가합니다. 새 열은 테이블의 마지막 열이 됩니다.

열 추가

- ADD 절을 사용하여 열을 추가합니다.

```
ALTER TABLE dept80
ADD          (job_id VARCHAR2(9));
Table altered.
```

- 새 열은 마지막 열이 됩니다.

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

ORACLE

열 추가 지침

- 열을 추가 또는 수정할 수 있습니다.
- 열의 표시 위치를 지정할 수는 없습니다. 새 열은 마지막 열이 됩니다.

슬라이드 예제는 JOB_ID 열을 DEPT80 테이블에 추가하는데 JOB_ID 열은 테이블의 마지막 열이 됩니다.

참고: 열을 추가할 때 테이블이 이미 행을 포함하고 있으면 새 열의 모든 행은 초기에 널 값을 가집니다.

열 수정

- 열의 데이터 유형, 크기 및 기본값을 변경할 수 있습니다.

```
ALTER TABLE dept80
MODIFY      (last_name VARCHAR2(30));
Table altered.
```

- 기본값을 변경하면 변경 이후에 테이블에 삽입되는 항목에만 영향을 줍니다.

ORACLE

9-26

Copyright © Oracle Corporation, 2001. All rights reserved.

열 수정

ALTER TABLE 문을 MODIFY 절과 함께 사용하여 열의 데이터 유형, 크기 및 기본값을 포함한 열 정의를 수정할 수 있습니다.

지침

- 숫자 열의 폭(width) 또는 전체 자릿수를 늘릴 수 있습니다.
- 숫자 또는 문자열의 폭을 늘릴 수 있습니다.
- 열이 널 값만 포함하거나 테이블에 행이 없는 경우에만 열의 폭을 줄일 수 있습니다.
- 열이 널 값을 포함하는 경우에만 데이터 유형을 변경할 수 있습니다.
- 열이 널 값을 포함하거나 열 크기를 변경하지 않는 경우에만 CHAR 열을 VARCHAR2 데이터 유형으로 변환하거나 VARCHAR2 열을 CHAR 데이터 유형으로 변환할 수 있습니다.
- 열의 기본값을 변경하면 변경 이후에 테이블에 삽입되는 항목에만 영향을 줍니다.

열 삭제

DROP COLUMN 절을 사용하여 테이블에서 더 이상 필요하지 않은 열을 삭제합니다.

```
ALTER TABLE dept80
DROP COLUMN job_id;
Table altered.
```

ORACLE

9-27

Copyright © Oracle Corporation, 2001. All rights reserved.

열 삭제

ALTER TABLE 문을 DROP COLUMN 절과 함께 사용하여 테이블에서 열을 삭제할 수 있습니다. 이 기능은 Oracle8i부터 사용할 수 있습니다.

지침

- 열에 데이터가 포함되어 있어도 되고 포함되어 있지 않아도 됩니다.
- ALTER TABLE 문을 사용하면 한 번에 한 열만 삭제할 수 있습니다.
- 테이블을 변경한 후 테이블에 열이 하나 이상 있어야 합니다.
- 삭제된 열은 복구할 수 없습니다.

SET UNUSED 옵션

- SET UNUSED 옵션을 사용하여 하나 이상의 열을 UNUSED로 표시합니다.
- DROP UNUSED COLUMNS 옵션을 사용하여 UNUSED로 표시된 열을 제거합니다.

```
ALTER TABLE table  
SET UNUSED (column);
```

또는

```
ALTER TABLE table  
SET UNUSED COLUMN column;
```

```
ALTER TABLE table  
DROP UNUSED COLUMNS;
```

ORACLE

3-28

Copyright © Oracle Corporation, 2001. All rights reserved.

SET UNUSED 옵션

SET UNUSED 옵션은 시스템 자원에 대한 요구가 적을 때 열을 삭제할 수 있도록 하나 이상의 열을 UNUSED로 표시합니다. 이 기능은 Oracle8i부터 사용할 수 있습니다. 이 절을 지정한다고 해서 실제로 테이블의 각 행에서 대상 열이 제거되지는 않습니다. 즉 이 열이 사용하는 디스크 공간이 비워지는 것은 아닙니다. 따라서 DROP 절 실행에 소요되는 시간보다 응답 시간이 빨라집니다. UNUSED로 표시된 열은 테이블의 행에 열 데이터가 남아 있는 경우에도 삭제될 것으로 처리되므로 액세스할 수 없습니다. SELECT * 절의는 UNUSED 열의 데이터를 검색하지 않습니다. 또한 UNUSED 열의 이름 및 유형은 DESCRIBE 문을 실행해도 표시되지 않으며, UNUSED 열과 동일한 이름을 가진 새 열을 테이블에 추가할 수 있습니다. SET UNUSED 정보는 USER_UNUSED_COL_TABS 딕셔너리 뷰에 저장되어 있습니다.

DROP UNUSED COLUMNS 옵션

DROP UNUSED COLUMNS는 테이블에서 현재 UNUSED로 표시된 모든 열을 제거합니다. 테이블의 UNUSED 열로부터 디스크 공간을 회수하려고 할 경우 이 명령문을 사용하면 됩니다. 테이블에 UNUSED 열이 없으면 오류 발생 없이 명령문이 반환됩니다.

```
ALTER TABLE dept80  
SET UNUSED (last_name);  
Table altered.
```

```
ALTER TABLE dept80  
DROP UNUSED COLUMNS;  
Table altered.
```

테이블 삭제

- 테이블의 모든 데이터 및 구조를 삭제합니다.
- 보류 중인 트랜잭션을 모두 커밋합니다.
- 인덱스를 모두 삭제합니다.
- **DROP TABLE 문은 뒤판할 수 없습니다.**

```
DROP TABLE dept80;
Table dropped.
```

ORACLE

9-29

Copyright © Oracle Corporation, 2001. All rights reserved.

테이블 삭제

DROP TABLE 문은 오라클 테이블의 정의를 제거합니다. 테이블을 삭제하면 데이터베이스는 테이블의 모든 데이터 및 이와 관련된 모든 인덱스를 잊게 됩니다.

구문

```
DROP TABLE table
```

구문 설명:

table 테이블 이름입니다.

지침

- 테이블의 데이터를 모두 삭제합니다.
- 모든 뷰와 동의어는 그대로 남아 있지만 사용할 수 없습니다.
- 보류 중인 트랜잭션을 모두 커밋합니다.
- 해당 테이블을 생성한 사용자와 DROP ANY TABLE 권한을 가진 사용자만이 테이블을 제거할 수 있습니다.

참고: 실행한 DROP TABLE 문은 되돌릴 수 없습니다. Oracle server는 DROP TABLE 문을 실행할 경우 사용자에게 삭제를 확인하는 메시지 등을 표시하지 않고 삭제 작업을 실행합니다. 해당 테이블을 소유하고 있거나 높은 레벨의 권한이 있는 경우 테이블을 즉시 제거할 수 있습니다. 다른 DDL 문처럼 DROP TABLE도 자동으로 커밋됩니다.

객체 이름 변경

- 테이블, 뷰, 시퀀스 또는 동의어의 이름을 변경하려면 **RENAME** 문을 실행하십시오.

```
RENAME dept TO detail_dept;  
Table renamed.
```

- 사용자가 해당 객체의 소유자이어야 합니다.

ORACLE

9-30

Copyright © Oracle Corporation, 2001. All rights reserved.

테이블 이름 변경

추가 DDL 문으로 테이블, 뷰, 시퀀스 또는 동의어의 이름을 변경에 사용되는 RENAME 문이 있습니다.

구문

RENAME *old name* TO *new name*;

구문 설명:

old name 테이블, 뷰, 시퀀스 또는 동의어의 기존 이름입니다.

new_name 테이블, 뷰, 시퀀스 또는 동의어의 새 이름입니다.

사용자는 이름을 변경할 객체의 소유자이어야 합니다.

테이블 절단

- **TRUNCATE TABLE 문:**

- 테이블에서 모든 행을 제거합니다.
- 해당 테이블이 사용하는 저장 공간을 해제합니다.

```
TRUNCATE TABLE detail_dept;  
Table truncated.
```

- TRUNCATE를 사용한 행 제거 작업은 룰백할 수 없습니다.
- 대신 DELETE 문을 사용하여 행을 제거할 수 있습니다.

ORACLE

9-31

Copyright © Oracle Corporation, 2001. All rights reserved.

테이블 절단

TRUNCATE TABLE 문은 테이블에서 모든 행을 제거하고 해당 테이블이 사용하는 저장 공간을 해제할 때 사용하는 DDL 문입니다. TRUNCATE TABLE 문을 사용한 행 제거 작업은 룰백할 수 없습니다.

구문

```
TRUNCATE TABLE table;
```

구문 설명:

table 테이블 이름입니다.

테이블을 절단하면 테이블의 소유자이거나 테이블을 절단할 수 있는 DELETE TABLE 시스템 권한이 있어야 합니다.

DELETE 문 역시 테이블에서 모든 행을 제거할 수 있지만 저장 공간을 해제하지는 않으며 TRUNCATE 명령보다 느립니다. TRUNCATE 문으로 행을 제거하는 것이 DELETE 문으로 제거하는 것보다 빠른 이유는 다음과 같습니다.

- TRUNCATE 문은 DDL(데이터 정의어) 문이므로 롤백 정보를 생성하지 않습니다.
- 테이블을 절단하면 테이블의 삭제 트리거가 실행되지 않습니다.
- 테이블이 참조 무결성 제약 조건의 부모인 경우 테이블을 절단할 수 없습니다. TRUNCATE 문을 실행하기 전에 제약 조건을 해제하십시오.

테이블에 주석 추가

- COMMENT 문을 사용하여 테이블 또는 열에 주석을 추가할 수 있습니다.

```
COMMENT ON TABLE employees  
IS 'Employee Information';  
Comment created.
```

- 주석은 데이터 딕셔너리 뷰를 통해 볼 수 있습니다.
 - ALL_COL_COMMENTS
 - USER_COL_COMMENTS
 - ALL_TAB_COMMENTS
 - USER_TAB_COMMENTS

ORACLE

9-32

Copyright © Oracle Corporation, 2001. All rights reserved.

테이블에 주석 추가

COMMENT 문을 사용하면 열, 테이블, 뷰 또는 스냅샷에 관한 주석을 최대 2000바이트까지 추가할 수 있습니다. 주석은 데이터 딕셔너리에 저장되며 다음 데이터 딕셔너리 뷰의 COMMENTS 열을 통해 볼 수 있습니다.

- ALL_COL_COMMENTS
- USER_COL_COMMENTS
- ALL_TAB_COMMENTS
- USER_TAB_COMMENTS

구문

```
COMMENT ON TABLE table | COLUMN table.column  
IS 'text';
```

구문 설명:

<i>table</i>	테이블 이름입니다.
<i>column</i>	테이블의 열 이름입니다.
<i>text</i>	주석의 텍스트입니다.

주석을 빈 문자열(' ')로 설정하여 데이터베이스에서 주석을 삭제할 수 있습니다.

```
COMMENT ON TABLE employees IS ' ';
```

요약

이 단원에서는 **DML** 문을 사용하여 테이블을 생성, 변경 및 삭제하고 이름을 변경하는 방법에 대해 배웠습니다.

명령문	설명
CREATE TABLE	테이블 생성
ALTER TABLE	테이블 구조 수정
DROP TABLE	행 및 테이블 구조 제거
RENAME	테이블, 뷰, 시퀀스 또는 동의어의 이름 변경
TRUNCATE	테이블에서 모든 행을 제거하고 저장 공간을 해제
COMMENT	테이블 또는 뷰에 주석 추가

ORACLE

9-33

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

이 단원에서는 DDL 문을 사용하여 테이블을 생성, 변경 및 삭제하고 이름을 변경하는 방법을 설명했습니다. 또한 테이블을 절단하고 테이블에 주석을 추가하는 방법을 설명했습니다.

CREATE TABLE

- 테이블을 생성합니다.
- 서브 쿼리를 사용하여 다른 테이블을 기반으로 하는 테이블을 생성합니다.

ALTER TABLE

- 테이블 구조를 수정합니다.
- 열 폭(width)과 열 데이터 유형을 변경하고 열을 추가합니다.

DROP TABLE

- 행 및 테이블 구조를 제거합니다.
- 이 명령문을 실행한 후에는 롤백할 수 없습니다.

RENAME

- 테이블, 뷰, 시퀀스 또는 동의어의 이름을 변경합니다.

TRUNCATE

- 테이블에서 모든 행을 제거하고 해당 테이블이 사용하는 저장 공간을 해제합니다.
- DELETE 문은 행만 제거합니다.

COMMENT

- 테이블 또는 열에 주석을 추가합니다.
- 데이터 딕셔너리를 질의하여 주석을 봅니다.

연습 9 개요

이 연습에서는 다음 내용을 다룹니다.

- 새 테이블 생성
- CREATE TABLE AS 구문을 사용하여 새 테이블 생성
- 열 정의 수정
- 테이블의 존재 확인
- 테이블에 주석 추가
- 테이블 삭제
- 테이블 변경

ORACLE

9-34

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 9 개요

CREATE TABLE 문을 사용하여 새 테이블을 생성한 후 새 테이블이 데이터베이스에 추가되었는지 확인합니다. 명령 파일에 구문을 작성한 후 해당 명령 파일을 실행하여 테이블을 생성합니다.

연습 9

1. 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

Name	Null?	Type
ID		NUMBER(7)
NAME		VARCHAR2(25)

2. DEPARTMENT 테이블의 데이터를 DEPT 테이블에 추가하십시오.
필요한 열만 추가하십시오.
3. 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

Name	Null?	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

연습 9(계속)

4. 긴 성을 가진 사원의 성을 표시할 수 있도록 EMP 테이블을 수정한 후 수정 내용을 확인하십시오.

Name	Null?	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(50)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

5. DEPT 및 EMP 테이블이 모두 데이터 딕셔너리에 저장되었는지 확인하십시오(힌트: USER_TABLES).

TABLE_NAME
DEPT
EMP

6. EMPLOYEES 테이블 구조를 기반으로 EMPLOYEES2 테이블을 생성하십시오. EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY 및 DEPARTMENT_ID 열만 포함시키고 새 테이블의 열 이름을 각각 ID, FIRST_NAME, LAST_NAME, SALARY 및 DEPT_ID로 지정하십시오.
7. EMP 테이블을 삭제하십시오.
8. EMPLOYEES2 테이블의 이름을 EMP로 변경하십시오.
9. DEPT 및 EMP 테이블 정의에 테이블을 설명하는 주석을 추가한 후 데이터 딕셔너리에서 추가한 항목을 확인하십시오.
10. EMP 테이블에서 FIRST_NAME 열을 삭제한 후 테이블 설명을 참조하여 수정 내용을 확인하십시오.
11. EMP 테이블의 DEPT_ID 열을 UNUSED로 표시한 후 테이블 설명을 참조하여 수정 내용을 확인하십시오.
12. EMP 테이블에 있는 UNUSED 열을 모두 삭제한 후 테이블 설명을 참조하여 수정 내용을 확인하십시오.

10

제작 조건 포함

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

목표

이 단원을 마치면 다음을 수행할 수 있습니다.

- 제약 조건 설명
- 제약 조건 생성 및 유지 관리

ORACLE

10-2

Copyright © Oracle Corporation, 2001. All rights reserved.

단원 목표

이 단원에서는 무결성 제약 조건을 포함시켜 업무 규칙을 구현하는 방법을 설명합니다.

제약 조건이란?

constraint

- 제약 조건은 테이블 레벨로 규칙을 적용합니다.
- 제약 조건은 종속된 테이블의 삭제를 방지합니다.
- 다음은 유효한 제약 조건 유형입니다.
 - NOT NULL
 - UNIQUE
 - PRIMARY KEY
 - FOREIGN KEY (참조키, 외래키)
 - CHECK

ORACLE

10-3

Copyright © Oracle Corporation, 2001. All rights reserved.

제약 조건

Oracle Server는 제약 조건을 사용하여 테이블에 유효하지 않은 데이터가 입력되는 것을 방지합니다. 제약 조건을 사용하여 다음을 수행할 수 있습니다.

- 해당 테이블에서 행이 삽입, 개신 또는 삭제될 때마다 테이블의 데이터에 규칙을 적용합니다. 제약 조건을 만족시켜야 작업이 수행됩니다.
- 다른 테이블에 종속된 테이블의 삭제를 방지합니다.
- Oracle Developer와 같은 오라클 툴에 대한 규칙을 제공합니다.

데이터 무결성 제약 조건

제약 조건	설명
NOT NULL	열이 널 값을 포함하지 못하도록 지정합니다.
UNIQUE	테이블의 모든 행(row)에서 고유한 값을 갖는 열 또는 열 조합을 지정합니다.
PRIMARY KEY	테이블의 각 행을 고유하게 식별합니다.
FOREIGN KEY	한 열과 참조된 테이블의 열 간에 외래 키 관계를 설정하고 시험합니다.
CHECK	참(true)이어야 하는 조건을 지정합니다.

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

제약 조건 지침

- 제약 조건에 이름을 지정하지 않으면 Oracle server가 **SYS_Cn** 형식의 이름을 생성합니다.
- 제약 조건 생성 시기
 - 테이블이 생성될 때, 또는
 - 테이블이 생성된 후에
- 열 레벨 또는 테이블 레벨로 제약 조건을 정의합니다.
- 데이터 딕셔너리에서 제약 조건을 봅니다.

ORACLE

10-4

Copyright © Oracle Corporation, 2001. All rights reserved.

제약 조건 지침

모든 제약 조건은 데이터 딕셔너리에 저장됩니다. 제약 조건 이름은 표준 객체 이름 지정 규칙을 따라야 하며 제약 조건에 의미 있는 이름을 지정하면 참조하기가 쉬워집니다. 제약 조건에 이름을 지정하지 않으면 Oracle server가 **SYS_Cn** 형식의 이름을 생성하는데 여기서 *n*은 제약 조건 이름을 고유하게 만들어 주는 정수입니다.

제약 조건은 테이블이 생성될 때나 테이블이 생성된 후에 정의할 수 있습니다.

USER_CONSTRAINTS 데이터 딕셔너리 테이블을 통해 특정 테이블에 정의된 제약 조건을 볼 수 있습니다.

제약 조건 정의

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

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

ORACLE

10-5

Copyright © Oracle Corporation, 2001. All rights reserved.

제약 조건 정의

슬라이드는 테이블 생성 중에 제약 조건을 정의하기 위한 구문을 나타낸 것입니다.

구문 설명:

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

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

제약 조건 정의

- **열 제약 조건 레벨**

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

- **테이블 제약 조건 레벨**

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

ORACLE

10-6

Copyright © Oracle Corporation, 2001. All rights reserved.

제약 조건 정의(계속)

제약 조건은 일반적으로 테이블이 생성될 때 생성되지만 테이블이 생성된 후에도 추가할 수 있으며 일시적으로 사용할 수 없게 만들 수도 있습니다.

제약 조건은 다음 두 가지 레벨 중 하나로 정의할 수 있습니다.

제약 조건 레벨	설명
열	단일 열을 참조하며 해당 제약 조건을 소유하는 열의 명세 부분에서 정의됩니다. 모든 무결성 제약 유형을 정의할 수 있습니다.
테이블	하나 이상의 열을 참조하며 테이블의 열 정의와는 별도로 정의됩니다. NOT NULL을 제외한 모든 제약 조건을 정의할 수 있습니다.

구문 설명:

constraint_name 제약 조건 이름입니다.

constraint_type 제약 조건 유형입니다.

NOT NULL 제약 조건

해당 열에 널 값을 사용할 수 없도록 합니다.

EMPLOYEE_ID	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
100	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000	90
101	Kochhar	NKOCHHAR	515.123.4568	21-SEP-69	AD_VP	17000	90
102	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	90
103	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000	60
104	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000	60
178	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA REP	7000	
200	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400	10

20 rows selected.

NOT NULL 제약 조건
(이 열의 모든 행이 널
값을 가질 수 없음)

NOT NULL
제약 조건

NOT NULL 제약 조건
없음
(이 열의 모든 행이
널 값을 가질 수 있음)

ORACLE®

NOT NULL 제약 조건

NOT NULL 제약 조건을 사용하면 해당 열에 널 값이 사용되지 않습니다. NOT NULL 제약 조건이 없는 열은 기본적으로 널 값을 가질 수 있습니다.

NOT NULL 제약 조건

열 레벨로 정의합니다.

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,           ← 시스템이
    salary           NUMBER(8,2),                      이름 지정
    commission_pct   NUMBER(2,2),
    hire_date        DATE,
    CONSTRAINT emp_hire_date_nn
    NOT NULL,                                         ← 사용자가
    ...                                                 이름 지정
```

NOT NULL 제약 조건(계속)

NOT NULL 제약 조건은 테이블 레벨이 아닌 열 레벨로만 지정할 수 있습니다.

슬라이드 예제는 NOT NULL 제약 조건을 EMPLOYEES 테이블의 LAST_NAME 및 HIRE_DATE 열에 적용합니다. 이 제약 조건에 이름을 지정하지 않았으므로 Oracle server가 제약 조건의 이름을 생성합니다.

제약 조건을 지정하면서 제약 조건의 이름을 지정할 수 있습니다.

```
... last_name VARCHAR2(25)
      CONSTRAINT emp_last_name_nn NOT NULL...
```

참고: 이 단원에서 설명하는 제약 조건 예제 중에는 이 과정에서 제공한 예제 테이블에 없는 것도 있습니다. 필요하면 제약 조건을 테이블에 추가할 수 있습니다.

UNIQUE 제약 조건

UNIQUE 제약 조건

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	EMAIL
100	King	SKing
101	Kochhar	NKOCHHAR
102	De Haan	LDEHAAN
103	Hunold	AHUNOLD
104	Ernst	BERNST

...



208	Smith	JSMITH
209	Smith	JSMITH

허용됨

허용되지 않음:
이미 존재

ORACLE

UNIQUE 제약 조건

UNIQUE 키 무결성 제약 조건은 열 또는 열 집합(키)의 모든 값이 고유하게 유지되도록 합니다. 즉 한 테이블에 있는 두 행은 지정된 열 또는 열 집합에서 중복된 값을 가질 수 없습니다. UNIQUE 키 제약 조건이 정의된 열 또는 열 집합을 고유 키라고 합니다. UNIQUE 제약 조건이 하나 이상의 열을 포함하는 경우 해당 열 그룹을 조합 고유 키라고 합니다.

UNIQUE 제약 조건을 지정한 열에 NOT NULL 제약 조건을 정의하지 않으면 이 열에는 널 값이 허용됩니다. 널 값은 어떠한 값과도 동일한 것으로 취급되지 않으므로, NOT NULL 제약 조건이 없는 열에는 널 값이 포함된 행이 여러 개 존재할 수 있습니다. 열에 널 값이 있어도(또는 조합 UNIQUE 키가 적용되는 모든 열에 널 값이 있어도) UNIQUE 제약 조건은 항상 만족됩니다.

참고: 둘 이상의 열에 대한 UNIQUE 제약 조건 검색 방식 때문에 부분적으로 널을 포함할 수 있는 조합 UNIQUE 키 제약 조건이 적용될 경우 널이 아닌 열(들)은 동일한 값을 가질 수 없습니다.

UNIQUE 제약 조건

테이블 레벨 또는 열 레벨로 정의합니다.

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

```

ORACLE

10-10

Copyright © Oracle Corporation, 2001. All rights reserved.

UNIQUE 제약 조건(계속)

UNIQUE 제약 조건은 열 레벨 또는 테이블 레벨로 정의할 수 있습니다. 조합 고유 키는 테이블 레벨 정의를 사용하여 생성합니다.

슬라이드 예제는 UNIQUE 제약 조건을 EMPLOYEES 테이블의 EMAIL 열에 적용하며 제약 조건의 이름은 EMP_EMAIL_UK입니다.

참고: Oracle server는 암시적(**implicit**)으로 고유 키 열에 고유 인덱스를 생성하여 UNIQUE 제약 조건을 시행합니다.

PRIMARY KEY 제약 조건

DEPARTMENTS

PRIMARY KEY

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1600
60	IT	103	1400
80	Sales	149	2500

...

허용되지 않음
(널 값)



INSERT INTO

Public Accounting	1400
Finance	124

허용되지 않음
(50 이미 존재)

ORACLE

10-11

Copyright © Oracle Corporation, 2001. All rights reserved.

PRIMARY KEY 제약 조건

PRIMARY KEY 제약 조건은 테이블의 기본 키를 생성하는데, 기본 키는 테이블 당 하나만 생성할 수 있습니다. PRIMARY KEY 제약 조건은 테이블의 각 행을 고유하게 식별하는 열 또는 열 집합입니다. 이 제약 조건은 열 또는 열 조합에 고유성을 부여하며 기본 키에 속하는 열이 널 값을 가질 수 없도록 합니다.

PRIMARY KEY 제약 조건

테이블 레벨 또는 열 레벨로 정의합니다.

```
CREATE TABLE departments (
    department_id      NUMBER(4),
    department_name    VARCHAR2(30)
        CONSTRAINT dept_name_nn NOT NULL,
    manager_id         NUMBER(6),
    location_id        NUMBER(4),
    CONSTRAINT dept_id_pk PRIMARY KEY(department_id));
```

ORACLE

10-12

Copyright © Oracle Corporation, 2001. All rights reserved.

PRIMARY KEY 제약 조건(계속)

PRIMARY KEY 제약 조건은 열 레벨 또는 테이블 레벨로 정의할 수 있습니다. 조합 PRIMARY KEY는 테이블 레벨 정의를 사용하여 생성합니다.

하나의 테이블에 PRIMARY KEY 제약 조건은 하나만 있을 수 있지만 UNIQUE 제약 조건은 여러 개가 있을 수 있습니다.

슬라이드 예제는 DEPARTMENTS 테이블의 DEPARTMENT_ID 열에 PRIMARY KEY 제약 조건을 정의하며 제약 조건의 이름은 DEPT_ID_PK입니다.

참고: PRIMARY KEY 열에 대해서는 UNIQUE 인덱스가 자동으로 생성됩니다.

FOREIGN KEY 제약 조건

DEPARTMENTS

**PRIMARY
KEY**

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1900
60	IT	103	1400
80	Sales	149	2500

...

EMPLOYEES

**FOREIGN
KEY**

**PRIMARY
KEY**

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90
103	Hunold	60
104	Ernst	60
107	Lorentz	60

...



INSERT INTO

200	Ford	9
201	Ford	60

허용되지 않음
(9가 존재
하지 않음)
허용됨

ORACLE

10-13

Copyright © Oracle Corporation, 2001. All rights reserved.

FOREIGN KEY 제약 조건

FOREIGN KEY 즉, 참조 무결성 제약 조건은 열 또는 열 조합을 외래 키로 지정하고 동일한 테이블이나 다른 테이블에 있는 기본 키 또는 고유 키와의 관계를 설정합니다. 슬라이드 예제에서 DEPARTMENT_ID는 EMPLOYEES 테이블(종속 또는 자식 테이블)에서 외래 키로 정의되어 있으며 DEPARTMENTS 테이블(참조 또는 부모 테이블)의 DEPARTMENT_ID 열을 참조합니다.

외래 키 값은 부모 테이블의 기존 값과 일치하거나 NULL이어야 합니다.

외래 키는 데이터 값을 기준으로 하며 물리적 포인터가 아닌 논리적 포인터입니다.

FOREIGN KEY 제약 조건

테이블 레벨 또는 열 레벨로 정의합니다.

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

ORACLE

10-14

Copyright © Oracle Corporation, 2001. All rights reserved.

FOREIGN KEY 제약 조건(계속)

FOREIGN KEY 제약 조건은 열 또는 테이블 제약 조건 레벨로 정의할 수 있습니다. 조합 외래 키는 테이블 레벨 정의를 사용하여 생성해야 합니다.

슬라이드 예제는 테이블 레벨 구문을 사용하여 EMPLOYEES 테이블의 DEPARTMENT_ID 열에 FOREIGN KEY 제약 조건을 정의하며 제약 조건의 이름은 EMP_DEPTID_FK입니다.

제약 조건이 단일 열을 기반으로 할 경우 열 레벨로 외래 키를 정의할 수도 있는데 이때 구문에 키워드인 FOREIGN KEY가 나타나지 않는다는 점이 다릅니다. 예를 들면 다음과 같습니다.

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

FOREIGN KEY 제약 조건 키워드

- FOREIGN KEY: 테이블 제약 조건 레벨로 자식 테이블의 열을 정의합니다.
- REFERENCES: 부모 테이블 및 부모 테이블에서의 해당 열을 식별합니다.
- ON DELETE CASCADE: 부모 테이블의 행이 삭제되는 경우 자식 테이블의 종속 행을 삭제합니다.
- ON DELETE SET NULL: 종속 외래 키 값을 널로 변환 합니다.

ORACLE

10-15

Copyright © Oracle Corporation, 2001. All rights reserved.

FOREIGN KEY 제약 조건(계속)

외래 키는 자식 테이블에서 정의됩니다. 참조되는 열을 포함하는 테이블은 부모 테이블입니다. 다음 키워드를 조합하여 외래 키를 정의합니다.

- FOREIGN KEY를 사용하여 테이블 제약 조건 레벨로 자식 테이블의 열을 정의합니다.
- REFERENCES는 부모 테이블 및 부모 테이블에서의 해당 열을 식별합니다.
- ON DELETE CASCADE는 부모 테이블의 행이 삭제될 경우 자식 테이블의 종속 행도 삭제되어야 함을 나타냅니다.
- ON DELETE SET NULL은 부모 값이 제거되는 경우 외래 키 값을 널로 변환합니다.

기본 동작을 제한 규칙이라고 하는데 이 규칙에서는 참조되는 데이터의 개선 또는 삭제가 허용되지 않습니다.

ON DELETE CASCADE 또는 ON DELETE SET NULL 옵션을 사용하지 않으면 자식 테이블에서 참조되는 부모 테이블의 행을 삭제할 수 없습니다.

CHECK 제약 조건

- 각 행이 만족시켜야 하는 조건을 정의합니다.
- 다음 표현식은 허용되지 않습니다.
 - CURRVAL, NEXTVAL, LEVEL 및 ROWNUM 의사 열 참조
 - SYSDATE, UID, USER 및 USERENV 함수 호출
 - 다른 행의 다른 값을 참조하는 질의

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

ORACLE

10-16

Copyright © Oracle Corporation, 2001. All rights reserved.

CHECK 제약 조건

CHECK 제약 조건은 각 행이 만족시켜야 하는 조건을 정의합니다. 이 조건에서는 질의 조건과 동일한 구조를 사용할 수 있지만 다음과 같은 예외가 있습니다.

- CURRVAL, NEXTVAL, LEVEL 및 ROWNUM 의사 열 참조
- SYSDATE, UID, USER 및 USERENV 함수 호출
- 다른 행의 다른 값을 참조하는 질의

단일 열이 자신을 참조하는 CHECK 제약 조건을 열 정의에 여러 개 포함시킬 수 있습니다. 열에 정의할 수 있는 CHECK 제약 조건의 수에는 제한이 없습니다.

CHECK 제약 조건은 열 레벨 또는 테이블 레벨로 정의할 수 있습니다.

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

제약 조건 추가 구문

다음 작업에 ALTER TABLE 문을 사용합니다.

- 제약 조건을 추가 또는 삭제할 수 있지만 구조를 수정할 수는 없습니다.
- 제약 조건을 활성화 또는 비활성화합니다.
- MODIFY 절을 사용하여 NOT NULL 제약 조건을 추가합니다.

```
ALTER TABLE table  
ADD [CONSTRAINT constraint] type (column);
```

10-17

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

제약 조건 추가

ALTER TABLE 문을 ADD 절과 함께 사용하면 기존 테이블에 제약 조건을 추가할 수 있습니다.

구문 설명:

table	테이블 이름입니다.
constraint	제약 조건 이름입니다.
type	제약 조건의 유형입니다.
column	제약 조건의 영향을 받는 열의 이름입니다.

제약 조건 이름 구문은 선택사항이지만 사용하는 것이 좋습니다. 제약 조건 이름을 지정하지 않으면 시스템이 제약 조건의 이름을 생성합니다.

지침

- 제약 조건을 추가 또는 삭제하거나 활성화 또는 비활성화할 수 있지만 구조를 수정할 수는 없습니다.
- ALTER TABLE 문의 MODIFY 절을 사용하여 기존 열에 NOT NULL 제약 조건을 추가할 수 있습니다.

참고: NOT NULL 열은 테이블이 비어 있거나 해당 열의 모든 행에 값이 있는 경우에만 정의할 수 있습니다.

제약 조건 추가

EMPLOYEES 테이블에 FOREIGN KEY 제약 조건을 추가하는 다음 작업은 관리자가 EMPLOYEES 테이블에 유효한 사원으로 존재해야 함을 나타냅니다.

```
ALTER TABLE employees
ADD CONSTRAINT emp_manager_fk
FOREIGN KEY(manager_id)
REFERENCES employees(employee_id);
Table altered.
```

ORACLE

10-18

Copyright © Oracle Corporation, 2001. All rights reserved.

제약 조건 추가(계속)

슬라이드의 예제는 EMPLOYEES 테이블에 FOREIGN KEY 제약 조건을 생성합니다. 이 제약 조건은 관리자가 EMPLOYEES 테이블에 유효한 사원으로 존재해야 함을 나타냅니다.

제약 조건 삭제

- EMPLOYEES 테이블에서 관리자 제약 조건을 삭제합니다.

```
ALTER TABLE employees
DROP CONSTRAINT emp_manager_fk;
Table altered.
```

- DEPARTMENTS 테이블에서 PRIMARY KEY 제약 조건을 삭제하고 EMPLOYEES.DEPARTMENT_ID 열에서 연관된 FOREIGN KEY 제약 조건을 삭제합니다.

```
ALTER TABLE departments
DROP PRIMARY KEY CASCADE;
Table altered.
```

ORACLE

10-19

Copyright © Oracle Corporation, 2001. All rights reserved.

제약 조건 삭제

USER_CONSTRAINTS 및 USER_CONS_COLUMNS 테이터 딕셔너리 뷰에서 제약 조건 이름을 확인한 후, ALTER TABLE 문을 DROP 절과 함께 사용하여 제약 조건을 삭제합니다. DROP 절의 CASCADE 옵션을 사용하면 종속 제약 조건을 모두 삭제할 수 있습니다.

구문

```
ALTER TABLE table
DROP PRIMARY KEY | UNIQUE (column) |
CONSTRAINT constraint [CASCADE];
```

구문 설명:

table 테이블 이름입니다.

column 제약 조건의 영향을 받는 열의 이름입니다.

constraint 제약 조건 이름입니다.

무결성 제약 조건을 삭제하면 Oracle server가 해당 제약 조건을 더 이상 시행하지 않으며 해당 제약 조건이 데이터 딕셔너리에서 더 이상 나타나지 않습니다.

제약 조건 비활성화

- ALTER TABLE 문의 DISABLE 절을 실행하여 무결성 제약 조건을 비활성화합니다.
- CASCADE 옵션을 적용하여 종속 무결성 제약 조건을 비활성화합니다.

```
ALTER TABLE      employees
DISABLE CONSTRAINT emp_emp_id_pk CASCADE;
Table altered.
```

ORACLE

10-20

Copyright © Oracle Corporation, 2001. All rights reserved.

제약 조건 비활성화

ALTER TABLE 문을 DISABLE 절과 함께 사용하면 제약 조건을 삭제하거나 다시 생성하지 않고도 제약 조건을 비활성화할 수 있습니다.

구문

```
ALTER TABLE table
DISABLE CONSTRAINT constraint [CASCADE] ;
```

구문 설명:

table 테이블 이름입니다.
constraint 제약 조건 이름입니다.

지침

- CREATE TABLE 문과 ALTER TABLE 문 모두에서 DISABLE 절을 사용할 수 있습니다.
- CASCADE 절은 종속 무결성 제약 조건을 비활성화합니다.
- 고유 또는 기본 키 제약 조건을 비활성화하면 고유 인덱스가 제거됩니다.

제약 조건 활성화

- 현재 테이블 정의에서 비활성화되어 있는 무결성 제약 조건을 활성화하려면 **ENABLE** 절을 사용합니다.

```
ALTER TABLE      employees
ENABLE CONSTRAINT emp_emp_id_pk;
Table altered.
```

- UNIQUE 키 또는 PRIMARY KEY 제약 조건을 활성화하면 UNIQUE 또는 PRIMARY KEY 인덱스가 자동으로 생성됩니다.

ORACLE

10-21

Copyright © Oracle Corporation, 2001. All rights reserved.

제약 조건 활성화

ALTER TABLE 문을 ENABLE 절과 함께 사용하면 제약 조건을 삭제하거나 다시 생성하지 않고도 제약 조건을 활성화할 수 있습니다.

구문

```
ALTER TABLE      table
ENABLE CONSTRAINT constraint;
```

구문 설명:

table 테이블 이름입니다.
constraint 제약 조건 이름입니다.

지침

- 제약 조건을 활성화하면 해당 제약 조건이 테이블에 있는 모든 데이터에 적용됩니다. 테이블의 모든 데이터는 제약 조건을 만족시켜야 합니다.
- UNIQUE 키 또는 PRIMARY KEY 제약 조건을 활성화하면 UNIQUE 또는 PRIMARY KEY 인덱스가 자동으로 생성됩니다.
- CREATE TABLE 문과 ALTER TABLE 문 모두에서 ENABLE 절을 사용할 수 있습니다.
- CASCADE 옵션으로 비활성화했던 기본 키 제약 조건을 다시 활성화할 경우 해당 기본 키에 종속된 외래 키는 활성화되지 않습니다.

제약 조건 연쇄화

- CASCADE CONSTRAINTS 절은 DROP COLUMN 절과 함께 사용됩니다.
- CASCADE CONSTRAINTS 절을 사용하면 삭제되는 열에 정의된 기본 키 및 고유 키를 참조하는 모든 참조 무결성 제약 조건이 삭제됩니다.
- CASCADE CONSTRAINTS 절은 삭제되는 열에 정의된 다중 열 제약 조건도 모두 삭제합니다.

ORACLE

10-22

Copyright © Oracle Corporation, 2001. All rights reserved.

제약 조건 연쇄화

이 명령문은 CASCADE CONSTRAINTS 절의 사용 방법을 보여줍니다. TEST1 테이블이 다음과 같이 생성되었다고 가정합니다.

```
CREATE TABLE test1 (
    pk NUMBER PRIMARY KEY,
    fk NUMBER,
    col1 NUMBER,
    col2 NUMBER,
    CONSTRAINT fk_constraint FOREIGN KEY (fk) REFERENCES test1,
    CONSTRAINT ck1 CHECK (pk > 0 and col1 > 0),
    CONSTRAINT ck2 CHECK (col2 > 0));
```

다음 명령문을 사용하면 오류가 발생합니다.

```
ALTER TABLE test1 DROP (pk); -- pk는 부모 키입니다.
```

```
ALTER TABLE test1 DROP (col1); -- col1은 다중 열 제약 조건인 ck1에서 참조합니다.
```

제약 조건 연쇄화

예제:

```
ALTER TABLE test1
DROP (pk) CASCADE CONSTRAINTS;
Table altered.
```

```
ALTER TABLE test1
DROP (pk, fk, coll) CASCADE CONSTRAINTS;
Table altered.
```

ORACLE

10-23

Copyright © Oracle Corporation, 2001. All rights reserved.

제약 조건 연쇄화(계속)

다음 명령문을 입력하면 기본 키 제약 조건이 적용되는 PK 열, 외래 키 제약 조건인 fk_constraint 및 CHECK 제약 조건인 CK1이 삭제됩니다.

```
ALTER TABLE test1 DROP (pk) CASCADE CONSTRAINTS;
```

삭제되는 열에 정의되어 있는 제약 조건을 참조하는 열이 모두 삭제되는 경우에는 CASCADE CONSTRAINTS가 필요 없습니다. 예를 들어, 다른 테이블의 참조 제약 조건이 PK 열을 참조하지 않으면 CASCADE CONSTRAINTS 절을 사용하지 않고 다음과 같이 명령문을 실행해도 됩니다.

```
ALTER TABLE test1 DROP (pk, fk, coll);
```

제약 조건 보기

USER_CONSTRAINTS 테이블을 질의하여 모든 제약 조건
의 정의 및 이름을 봅니다.

```
SELECT constraint_name, constraint_type,  
       search_condition  
  FROM user_constraints  
 WHERE table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	C	SEARCH_CONDITION
EMP_LAST_NAME_NN	C	"LAST_NAME" IS NOT NULL
EMP_EMAIL_NN	C	"EMAIL" IS NOT NULL
EMP_HIRE_DATE_NN	C	"HIRE_DATE" IS NOT NULL
EMP_JOB_NN	C	"JOB_ID" IS NOT NULL
EMP_SALARY_MIN	C	salary > 0
EMP_EMAIL_UK	U	

...

ORACLE

10-24

Copyright © Oracle Corporation, 2001. All rights reserved.

제약 조건 보기

테이블을 생성한 후 DESCRIBE 명령을 실행하면 테이블의 생성 여부를 확인할 수 있지만 이 명령을 사용하면 NOT NULL 제약 조건만 확인할 수 있습니다. 테이블의 제약 조건을 모두 보려면 USER_CONSTRAINTS 테이블을 질의하십시오.

슬라이드의 예제는 EMPLOYEES 테이블의 제약 조건을 표시합니다.

참고: 테이블 소유자가 제약 조건 이름을 지정하지 않으면 시스템이 이름을 할당합니다. 제약 조건 유형에서 C는 CHECK를, P는 PRIMARY KEY를, R는 참조 무결성을, U는 UNIQUE 키를 나타냅니다. NOT NULL 제약 조건은 사실상 CHECK 제약 조건입니다.

제약 조건과 연관된 열 보기

USER_CONS_COLUMNS 뷰를 통해 제약 조건 이름과 연관된
열을 봅니다.

```
SELECT constraint_name, column_name
FROM user_cons_columns
WHERE table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	COLUMN_NAME
EMP_DEPT_FK	DEPARTMENT_ID
EMP_EMAIL_NN	EMAIL
EMP_EMAIL_UK	EMAIL
EMP_EMP_ID_PK	EMPLOYEE_ID
EMP_HIRE_DATE_NN	HIRE_DATE
EMP_JOB_FK	JOB_ID
EMP_JOB_NN	JOB_ID

ORACLE

10-25

Copyright © Oracle Corporation, 2001. All rights reserved.

제약 조건 보기(계속)

USER_CONS_COLUMNS 데이터 덕셔너리 뷰를 질의하여 제약 조건과 연관된 열의 이름을 볼 수 있습니다. 이 뷰는 특히 시스템이 할당한 이름을 사용하는 제약 조건에 유용합니다.

요약

이 단원에서는 제약 조건을 생성하는 방법에 대해 배웠습니다.

- 제약 조건 유형:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK

- USER_CONSTRAINTS 테이블을 질의하면 모든 제약 조건의 정의 및 이름을 볼 수 있습니다.

ORACLE

10-26

Copyright © Oracle Corporation, 2001. All rights reserved.

요약

이 단원에서는 Oracle server에서 제약 조건을 사용하여 테이블에 잘못된 데이터가 입력되는 것을 막는 방법을 설명했습니다. 또한 DDL 문에서 제약 조건을 구현하는 방법도 설명했습니다.

다음은 유효한 제약 조건 유형입니다.

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK

USER_CONSTRAINTS 테이블을 질의하면 모든 제약 조건의 정의 및 이름을 볼 수 있습니다.

연습 10 개요

이 연습에서는 다음 내용을 다릅니다.

- 기존 테이블에 제약 조건 추가
- 테이블에 여러 열 추가
- 데이터 딕셔너리 뷰의 정보 표시

ORACLE

10-27

Copyright © Oracle Corporation, 2001. All rights reserved.

연습 10 개요

이 연습에서는 이 단원에서 설명한 명령문을 사용하여 테이블에 제약 조건 및 여러 열을 추가합니다.

참고: 연습 도중에 정의하는 제약 조건의 이름을 지정하는 것이 좋습니다.

연습 10

- 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.
힌트: 제약 조건은 ALTER TABLE 명령이 성공적으로 실행되자마자 활성화됩니다.
- 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.
힌트: 제약 조건은 ALTER TABLE 명령이 성공적으로 실행되자마자 활성화됩니다.
- 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.
- USER_CONSTRAINTS 뷰를 질의하여 제약 조건이 추가되었는지 확인하고 제약 조건의 유형 및 이름을 적어두십시오. 명령문 텍스트를 lab10_4.sql이라는 파일에 저장하십시오.

CONSTRAINT_NAME	C
MY_DEPT_ID_PK	P
SYS_C002541	C
MY_EMP_ID_PK	P
MY_EMP_DEPT_ID_FK	R

- USER_OBJECTS 데이터 덕셔너리 뷰에서 EMP 및 DEPT 테이블의 객체 이름 및 유형을 표시하십시오. 새 테이블 및 새 인덱스가 생성된 것을 볼 수 있습니다.

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.