sqld 시험 정리 2과목

카테고리	SQLD
체크 표시됨	☐
태그	notion

2과목

2-1. 관계형 데이터베이스 개요

데이터베이스와 DBMS

- 데이터베이스: 데이터의 집합, 꼭 형식을 갖추지 않아도 엑셀 파일을 모아 둔다면 그것 또한 데이터베이스임
- DBMS: 데이터를 효과적으로 관리하기 위한 시스템
- 개인이 파일을 여러 개 묶어서 폴더에 보관하면 데이터를 찾고 관리하는데 많은 비용이 발생, 이를 보다 시스템적으로 작동하게 만든 시스템을 DBMS라고 한다.(ORACLE, MYSQL 등)

관계형 데이터베이스 구성 요소

-계정: 데이터의 접근 제한을 위한 여러 업무별/시스템별 계정이 존재

-테이블: DBMS의 DB안에서 데이터가 저장되는 형식

-스키마: 테이블이 어떠한 구성으로 되어있는지, 어떠한 정보를 가지고 있는지에 대한 기본적인 구조를 정의

테이블

1. **정의**

- 엑셀에서의 워크시트처럼 행(로우)과 열(칼럼)을 갖는 2차원 구조로 구성, 데이터를 입력하여 저장하는 최소 단위
- 컬럼은 속성이라고도 부름(모델링 단계마다 부르는 용어가 다름)

2. **특징**

- 하나의 테이블은 반드시 하나의 유저(계정) 소유여야 함
- 테이블간 관계는 일대일, 일대다, 다대다의 관계를 가질 수 있음

- **테이블명은 중복될 수 없지만, 소유자가 다른 경우 같은 이름으로 생성 가능**

ex) SCOTT 소유의 EMP 테이블 존재, HR 소유의 EMP 테이블 생성 가능(같은 계정 내 동일한 객체 생성 불가)

-테이블은 행 단위로 데이터가 입력, 삭제되며 수정은 값의 단위로 가능

ex) 사원테이블에 새로운 사원 정보를 사원번호, 사원이름 등의 테이블 내 모든 컬럼의 값 동시에 전달하여 입력

SQL(Structrued Query Language)

-관계형 데이터베이스에서 데이터 조회 및 조작, DBBMS 시스템 관리 기능을 명령하는 언어

-데이터 정의(DDL), 데이터 조작(DML), 데이터 제어 언어(DCL) 등으로 구분

-SQL 문법은 대,소문자를 구분하지 x

관계형 데이터베이스의 특징

-데이터의 분류, 정렬, 탐색 속도가 빠름

-신뢰성이 높고 데이터의 무결성 보장

-기존의 작성된 스키마를 수정하기 어려움

-데이터베이스의 부하를 분석하는 것이 어려움

데이터 무결성(integrity)

- 데이터의 정확성과 일관성을 유지하고, 데이터에 결손과 부정합이 없음을 보증하는 것

데이터 무결성 종류

1. 개체 무결성: 테이블의 기본키를 구성하는 컬럼(속성)은 NULL 값이나 중복값을 가질 수 없음
2. 참조 무결성: 외래키 값은 NULL이나 참조 테이블의 기본키 값과 동일해야 한다. (외래키란 참조 테이블의 기본키에 정의된 데이터만 허용되는 구조이므로)
3. 도메인 무결성: 주어진 속성 값이 정의된 도메인에 속한 값이어야 함
4. NULL 무결성: 특정 속성에 대해 NULL을 허용하지 않는 특징
5. 고유 무결성: 특정 속성에 대해, 값이 중복되지 않는 특징
6. 키 무결성: 하나의 릴레이션(관계)에는 적어도 하나의 키가 존재해야 함 (테이블이 서로 관계를 가질 경우 반드시 하나 이상의 조인키를 가짐)

*도메인: 각 컬럼(속성)이 갖는 범위

*릴레이션: 테이블간 관계를 말함

*튜플: 하나의 행을 의미함

*키: 식별자

ERD(Entity Relationship Diagram)

- ERD란 테이블 간 서로의 상관 관계를 그림으로 표현하는 것
- ERD의 구성요소에는 엔터티(Entity),관계(Relationship),속성(Atribute)이 있다.

→ 현실 세계의 데이터는 이 3가지의 구성으로 모두 표현 가능

SQL 종류

-SQL은 그 기능에 따라 다음과 같이 구분함

구분	종류
DDL	CREATE, ALTER, DROP, TRUNCATE
DML	INSERT, DELETE, UPDATE, MERGE
DCL	GRANT, REVOKE
TCL	COMMIT, ROLLBACK
DQL	SELECT

*사실 SELECT 문은 따로 SQL 종류 중 어디에도 속하지 않아서 SELECT문을 위한 DQL 등장

SELECT문 구조

-SELECT문은 다음과 같이 6개 절로 구성

-각 절의 순서대로 작성해야 함(GROUPBY 와 HAVING은 서로 바꿀 수 있지만 보통 사용 x)

-SELECT문의 내부 파싱(문법적 해석) 순서는 나열된 순서와는 다름

-FROM > WHERE > GROUP BY > HAVING > SELECT > ORDER BY 순서대로 실행됨

1. SELECT * | 컬럼명 | 표현식
2. FROM 테이블명 또는 뷰명
3. WHERE 조회 조건
4. GROUP BY 그룹핑컬럼명
5. HAVING 그룹핑 필터링 조건

6. ORDER BY 정렬컬럼명

특징

-SELECT절에서 표시할 대상 컬럼에 Alias(별칭) 지정 가능

-대소문자를 구분하지 않아도 인식한다.

컬럼 Alias(별칭)

-컬럼명 대신 출력할 암시 이름 지정(SELECT 절에서만 저으이 가능, 원본 컬럼명은 변경되지 x)

-컬렴명 뒤에 AS 와 함께 컬럼 별칭 전달(AS는 생략 가능)

특징 및 주의사항

-SELECT문보다 늦게 수령되는 ORDER BY 절에서만 컬럼 별칭 사용 가능 (그 외 절에서 사용시 에러 발생)

-한글 사용 가능(한글 지원 캐릭터셋 설정 시)

-이미 존재하는 예약어는 별칭으로 사용 불가

ex) avg, count, decode, SELECT, FROM 등

-다음의 경우 별칭에 반드시 쌍따옴표 전달 필요

1) 별칭에 공백을 포함하는 경우

2) 별칭에 특수문자를 포함하는 경우(”_” 제외)

3) 별칭 그대로 전달할 경우(입력한 대소를 그대로 출력하고자 할 때)

FROM 절

-데이터를 불러올 테이블명 또는 뷰명 전달

-테이블 여러 개 전달 가능(컴마로 구분)→ 조인 조건 없이 테이블명만 나열 시 카티시안 곱 발생 주의!

-테이블 별칭 선언 가능(ORACLE은 AS 사용 불가, SQL Server는 사용/생략가능)

*테이블 별칭 선언 시 컬럼 구분자는 테이블 별칭으로만 전달(테이블명으로 사용 시 에러 발생)

-ORACLE에서는 FROM 절 생략 불가(의미상 필요 없는 경우 DUAL 테이블 선언)

*ORACLE 23c 버전부터는 생략 가능

-SQL Server에서는 FROM 절 필요 없을 경우 생략 가능(오늘 날짜 조회 시)

*뷰: 테이블과 동일하게 데이터를 조회할 수 있는 객체이지만 테이블처럼 실제 데이터가 저장된 것이 아닌, SELECT 문 결과에 이름을 붙여 그 이름만으로 조회가 가능하도록 한 기능

함수

함수 정의

- input value가 있을 경우 그에 맞는 output value를 출력해주는 객체
- input value와 output value 의 관계를 정의한 객체
- from 절을 제외한 모든 절에서 사용 가능

함수 기능

- 기본적인 쿼리문을 더욱 강력하게 해줌
- 데이터의 계산을 수행
- 개별 데이터의 항목을 수정

함수의 종류(입력값의 수에 따라)

-단일행 함수: input과 output의 관계가 1:1

-복수행 함수: 여러 건의 데이터를 동시에 입력 받아서 하나의 요약값을 리턴

(그룹함수 또는 집계함수라고 함)

입/출력값의 타입에 따른 함수 분류

1) 문자형 함수

-문자열 결합, 추출, 삭제 등을 수행

-단일형 함수 형태

-output은 대부분 문자값(LENGTH,INSTR 제외)

***문자함수 종류**

SUBSTR(대상,m,n) : 문자열 중 m 위치에서 n개의 문자열 수출

INSTR(대상, 첫줄문자열,m,n): 대상에서 찾을 문자열 위치 변환(m위치에서 시작, n번째 발견된 문자열위치)

LTRIM(대상,삭제문자열): 문자열 중 특정 문자열을 왼쪽에서 삭제

RTIRM(대상,삭제문자열): 문자열 중 특정 문자열을 오른쪽에서 삭제

TRIM(대상): 문자열 중 특정 문자열을 앞쪽에서 삭제

LPAD(대상,n,문자열): 대상 왼쪽에 문자열을 추가하여 총 n의 길이 리턴

RPAD(대상,n,문자열): 대상 오른쪽에 문자열을 추가하여 총 n의 길이 리턴

COMCAT(대상1, 대상2): 문자열 결합

LENTH(대상): answkduf rlfdl

REPLACE(대상, 찾을문자열, 바꿀문자열): 문자열 치환 및 삭제

TRANSLATE(대상, 찾을문자열, 바꿀문자열): 글자를 1대1로 치환

SQL Server)

- SUBSTR → SUBSTRING
- LENGTH → LEN
- INSTR → CHARINDEX

2) 숫자형 함수

-숫자를 입력하면 숫자 값을 반환

-단일행 함수 형태의 숫자함수

***숫자함수 종류**

ABS(숫자): 절대값 변환

ROUND(숫자, 자리수): 소수점 특정 자리에서 반올림 → 자리수가 음수이면 정수자리에서 반올림(백의 자리에서 반올림 진행 ex) 123.456,-2 → 100)

TRUNC(숫자, 자리수): 소수점 특정 자리에서 버림

SIGN(숫자): 숫가자 양수면 1, 음수면 -1이면 0 반환

FLOOR(숫자): 작거나 같은 최대 정수 리턴

CEIL(숫자): 크거나 같은 최소 정수 리턴

MOD(숫자1, 숫자2): 숫자1을 숫자2로 나누어 나머지 반환

POWER(m,n): m의 n 거듭제곱

SQRT: 루트값 리턴

3)날짜형 함수

-날짜 연산과 관련된 함수

-ORACLE과 SQL Server 함수 거의 다름

***날짜함수 종류**

SYSDATE: 현재 날짜와 시간 리턴

CURRENT_TIME: 현재 날짜 리턴

CURRENT_TIMESTAMP: 현재 타임스탬프 리턴

ADD_MONTHS(날짜,n): 날짜에서 n개월 후 리턴

MONTHS_BETWEEN(날짜1,날짜2): 날짜1과 날짜2의 개월 수 리턴

LAST_DAY(날짜): 주어진 월의 마지막 날짜 리턴

NEXT_DAY(날짜,n): 주어진 날짜 이후 지정된 요일의 첫 번째 날짜 리턴 (ex) 1:일요일)

ROUND(날짜, 자리수): 날짜 반올림

TRUNC(날짜, 자리수): 날짜 버림

SQL Server)

- SYSDATE → GETDATE
- ADD_MONTHS → DATEADD(월 뿐만 아니라 모든 단위 날짜 연산 가능)
- MONTHS_BETWEEN → DATEDIFF(두 날짜 사이의 년, 월, 일 추출)

4) 변환함수

-값이 데이터 타입을 변환

-문자를 숫자로 숫자를 문자로 날짜를 문자로 변경

***변환함수 종류**

TO_NUMBER(문자): 숫자 타입으로 변경하여 리턴

TO_CHAR(대상,포맷): 1)날짜의 포맷 변경 2)숫자의 포맷 변경

TO_DATE(문자, 포맷): 주어진 문자를 포맷 형식에 맞게 읽어 날짜로 리턴

FORMAT(날짜, 포맷): 날짜의 포맷 변경

CAST(대상 AS 데이터타입): 대상을 주어진 데이터타입으로 변환

SQL Server)

- TO_NUMBER, TO_DATE, TO_CHAR → CONVERT(포맷 전달 시)
- 단순 변환일 경우 주로 CAST 사용

5) 그룹함수

-다중행 함수

-여러 값이 input값으로 들어가서 하나의 요약된 값으로 리턴

-GROUP BY와 함께 자주 사용됨

-ORACLE과 SQL Server 거의 동일

*그룹함수 종류

COUNT(대상): 행의 수 리턴

SUM(대상): 총 합 리턴

AVG(대상): 평균 리턴

MIN(대상): 최솟값 리턴

MAX(대상): 최댓값 리턴

VARIANCE(대상): 분산 리턴

STDDEV(대상): 표준편차 리턴

SQL Server)

- VARIANCE → VAR
- STDDEV → STDEV

6) 일반함수

-기타 함수(널 치환 함수 등)

***일반(기타)함수 종류**

DECODE(대상, 값1, 리턴1, 값2, 리턴2, _, 그외리턴): 대상이 값1이면 리턴1, 값1와 같으면 리턴2, .. 그외에는 그외리턴값 리턴

NVL(대상, 치환값): 대상이 널이면 치환값으로 치환하여 리턴

NVL2(대상, 치환값1, 치환값2): 대상이 널이면 치환값2로 치환, 널이 아니면 치환값 1로 치환하여 리턴

COALESCE(대상1, 대상2, _, 그외리턴): 대상들 중 널이 아닌값 출력(가장 첫번째 부터) 대상3, _, 모두가 널이면 그외리턴값이 리턴됨

ISNULL(대상, 치환값): 대상이 널이면 치환값이 리턴

NULLIF(대상1, 대상2): 두 값이 같으면 널 리턴, 다르면 대상1 리턴

CASE문: 조건별 치환 및 연산 수행

WHERE 절

- 테이블의 데이터 중 원하는 조건에 맞는 데이터만 조회하고 싶을 경우 사용
- 여러 조건 동시 전달 가능 (AND와 OR로 조건 연결)
- NULL 조회 시 IS NULL/ IS NOT NULL 연산자 사용(=연산자로 조회 불가)
- 연산자를 사용하여 다양한 표현이 가능
- 조건 전달 시 비교 대상의 데이터 타입 일치하는 것이 좋음

ex) EMP 테이블의 부서번호 컬럼의 데이터타입은 숫자인데 문자상수로 비교시 성능 문제가 발생할 수 있음

연산자 종류	설명
=	같은 조건을 검색
<>, !=	같지 않은 조건을 검색
>	큰 조건을 검색
BETWEEN a And b	A와 B 사이에 있는 범위 값을 모두 검색
IN(a,b,c)	A 이거나 B 이거나 C 인 조건을 검색
LIKE	특정 패턴을 가지고 있는 조건을 검색
is Null	Null 값을 검색
is Not Null	null이 아닌 값을 검색
A AND B	A 조건과 B 조건을 모두 만족하는 값만 검색
A OR B	A 조건이나 B 조건 중 한가지라도 만족하는 값을 검색
NOT A	A가 아닌 모든 조건을 검색

**주의사항

-문자나 날짜 상수 표현 시 반드시 홀따옴표 사용(다른 절에서도 동일 적용)

-ORACLE은 문자 상수의 경우 대소문자를 구분

-MSSQL은 기본적으로 문자상수의 대소문자를 구분하지 x

IN 연산자

-포함연산자로 여러 상수와 일치하는 조건 전달 시 사용

-상수를 괄호로 묶어서 동시에 전달(문자와 날짜 상수의 경우 반드시 홀따옴표와 함께)

IN 연산자를 사용하면 조건대상(ENAME)과 연산자(=)의 반복을 줄일 수 있음

이때, ()안의 상수도 문자상수와 날짜상수는 홀따옴표 필수

LIKE 연산자

-정확하게 일치하지 않아도 되는 패턴 조건 전달 시 사용

-%와 _와 함께 사용됨

1) %: 자리수 제한 없는 모든이라는 의미

2) _: _ 하나 당 한 자리수를 의미하여 모든 값을 표현함

예제) LIKE 연산자

ENAME LIKE 'S%': 이름이 S로 시작하는

ENAME LIKE '%S%': 이름이 S를 포함하는

ENAME LIKE 'S': 이름이 S로 끝나는

ENAME LIKE '_S%': 이름이 두 번째 글자가 S인(맨 앞이 _인 것 주의! %이면 자리수 상관 없이 S를 포함하기만 하면 됨)

ENAME LIKE '__S__: 이름의 가운데 글자가 S이며 이름의 길이가 5 글자인

GROUP BY 절

-각 행을 특정 조건에 따라 그룹으로 분리하여 계산하도록 하는 구문식

-GROUP BY 절에 그룹을 지정할 컬럼을 전달(여러 개 전달 가능)

-만약 그룹 연산에서 제외할 대상이 있다면 미리 WHERE 절에서 해당 행을 제외함 (WHERE 절이 GROUP BY 절보다 먼저 수행되므로)

-그룹에 대한 조건은 WHERE 절에서 사용할 수 없음

-SELECT 절에 집계 함수를 사용하여 그룹연산 결과 표현

-GROUP BY 절을 사용하면 데이터가 요약되므로 요약되기 전 데이터와 함께 출력할 수 없음

HAVING 절

-그룹 함수 결과를 조건으로 사용할 때 사용하는 절

-WHERE 절을 사용하여 그룹을 제한할 수 없으므로 HAVING 절에 전달

-HAVING 절이 GROUP BY 절 앞에 올 수 있지만 뒤에 쓰는 것을 권장

-내부적 연산 순서가 SELECT 절보다 먼저이므로 SELECT 절에서 선언된 Alias 사용 불가

ORDER BY절

-데이터는 입력된 순서대로 출력되나, 출력되는 행의 순서를 사용자가 변경하고자 할 때 ORDER BY 절을 사용

-ORDER BY 뒤에 명시된 컬럼 순서대로 정렬 → 1차 정렬, 2차 정렬 전달 가능

-정렬 순서를 오름차순(ASC), 내림차순(DESC)으로 전달(생략 시 오름차순 정렬)

-유일하게 SELECT 절에 정의한 컬럼 별칭 사용 가능

-SELECT 절에, 선언된 순서대로의 숫자 전달 가능(컬럼명과 숫자 혼합 사용가능)

복합 정렬

-먼저 정렬한 값의 동일한 결과가 있을경우 추가적으로 정렬 가능

→ 1차 정렬한 값이 같은 경우 그 값 안에서 2차 정렬 컬럼값의 정렬이 일어남

NULL의 정렬

-NULL을 포함한 값의 정렬 시 ORACLE은 기본적으로 NULL을 마지막에 배치(SQL Server는 처음에 배치)

-ORACLE은 ORDER BY 절에 NULLS LAST | NULLS FIRST 을 명시하여 NULL 정렬 순서 변경 가능

JOIN(조인)

-여러 테이블의 데이터를 사용하여 동시 출력하거나 참조 할 경우 사용

-FROM 절에 조인할 테이블 나열

-ORACLE 표준은 테이블 나열 순서 중요하지 x, ANSI 표준은 OUTER JOIN 시 순서 중요

-WHERE 절에서 조인 조건을 작성(ORACLE 표준)

-동일한 열 이름이 여러 테이블에 존재할 경우 열 이름 앞에 테이블 이름이나 테이블 Alias 붙임

-N 개의 테이블을 조인하려면 최소 N-1개의 조인 조건이 필요

-ORACLE 표준과 ANSI 표준이 서로 다름

조인 종류

1) 조건의 형태에 따라

1) EQUI JOIN(등가 JOIN): JOIN 조건이 동등 조건인 경우

2) NON EQUI JOIN: JOIN 조건이 동등 조건이 아닌 경우

2) 조인 결과에 따라

1)INNER JOIN: JOIN 조건에 성립하는 데이터만 출력하는 경우

2)OUTER JOIN: JOIN 조건에 성립하지 않는 데이터도 출력하는 경우
(LEFT/RIGHT/FULL OUTER JOIN 으로 나뉨)

3. NATUAL JOIN: 조인조건 생략 시 두 테이블에 같은 이름으로 자연 연결되는 조인
4. CROSS JOIN: 조인조건 생략 시 두 테이블의 발생 가능한 모든 행을 출력하는 조인
5. SELF JOIN: 하나의 테이블을 두 번 이상 참조하여 연결하는 조인

EQUI JOIN(등가 JOIN)

-조인 조건이 '='(equal) 비교를 통해 같은 값을 가지는 행을 연결하여 결과를 얻는 조인 방법

-SQL 명령문에서 가장 많이 사용하는 조인 방법

-FROM 절에 조인하고자 하는 테이블을 모두 명시

-FROM절에 명시하는 테이블은 테이블 별칭(Alias) 사용 가능

-WHERE 절에 두 테이블의 공동 컬럼에 대한 조인 조건을 나열

NON-EQUI JOIN

-테이블을 연결짓는 조인 컬럼에 대한 비교 조건이 '<', BETWEEN A AND B 와 같이 '=' 조건이 아닌 연션자를 사용하는 경우의 조인조건

세 테이블 이상의 조인

-관계를 잘 파악하여 모든 테이블이 연결되도록 조인 조건 명시

-N개 테이블의 경우 최소 N-1개의 조인 조건 필요

SELF JOIN

-한 테이블 내 각 행끼리 관계를 갖는 경우 조인 기법

-한 테이블을 참조할 때마다(필요할 때마다) 명시해야 함

-테이블명이 중복되므로 반드시 테이블 별칭 사용

표준 조인

-ANSI 표준으로 작성되는 INNER JOIN, CROSS JOIN, NATURAL JOIN, OUTER JOIN을 말함

INNER JOIN

-내부 조인이라고 하며, 조인 조건이 일치하는 행만 추출(ORACLE 조인 기본)

-ANSI 표준의 경우 FROM 절에 INNER JOIN 혹은 줄여서 JOIN을 명시

-ANSI 표준의 경우 USING이나 ON 조건절을 필수적으로 사용

ON절

-조인할 양 컬럼의 컬럼명이 서로 다르더라도 사용 가능

-ON 조건의 괄호는 옵션(생략가능)

-컬럼명이 같을 경우 테이블 이름이나 별칭을 사용하여 명확하게 지정(테이블 출처 명확히)

-ON 조건절에서 조인조건 명시, WHERE 절에서는 일반조건 명시(WHERE 절과 ON 절을 쓰임에 따라 구분)

USING 조건절

-조인할 컬럼명이 같을 경우 사용

NATURAL JOIN

-두 테이블 간의 동일한 이름을 가지는 모든 컬럼들에 대헤 EQUI JOIN을 수행

-USING, ON, WHER 절에서 조건 정의 불가

-JOIN에 사용된 컬럼들은 데이터 유형이 동일해야 하며 접두사를 사용불가

→ NATURAL JOIN은 동일한 이름의 모든 컬럼을 조인 컬럼으로 사용하므로 조인 컬럼의 값이 모든 결과가 리턴됨

→ SUTDENT와 PROFESSOR 테이블에는 NAME 컬럼과 PROFND 컬럼이 컬럼명이 서로 동일함

CROSS JOIN

-테이블 간 JOIN 조건이 없는 경우 생성 가능한 모든 데이터들의 조합

(Cartesian product(카타시안곱)출력)

-양쪽 테이블 행의 수의 곱한 수의 데이터 조합 발생(m*n)

OUTER JOIN

-INNER JOIN과 대비되는 조인방식

-JOIN 조건에서 동일한 값이 없는 행도 반환할 때 사용

-두 테이블 중 한쪽에 NULL을 가지면 EQUI JOIN 시 출력되지 않음 → 이를 출력 시 OUTER JOIN 사용

-테이블 기준 방향에 따라 LEFT OUTRER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN 으로 구분

-OUTER 생략 가능(LEFT OUTER JOIN 0> LEFT JOIN)

****OUTER JOIN 종류**

1) LEFT OUTER JOIN

-FROM 절에 나열된 왼쪽 테이블에 해당하는 데이터를 읽은 후, 우측 테이블에서 JOIN 대상 읽어옴

-즉, 왼쪽 테이블이 기준이 되어 오른쪽 테이블 데이터를 채우는 방식

-우측 값에서 같은 값이 없는 경우 NULL 값으로 출력

2) RIGHT OUTER JOIN

-LEFT OUTER JOIN의 반대

-즉, 오른쪽 테이블 기준으로 왼쪽 테이블 데이터를 채우는 방식

-FROM 절에 테이블 순서를 변경하면 LEFT OUTER JOIN으로 수행 가능

3) FULL OUTER JOIN

-두 테이블 전체 기준으로 결과를 생성하여 중복 데이터는 삭제 후 리턴

-LEFT OTUER JOIN 결과와 RIGHT OUTER JOIN 결과의 UNION 연산 리턴과 동일함

-ORACLE 표준에는 없음