SQLD 오답노트(1)

Ver. 제 30회 시험대비 기출문제

개발자: 꾸러기주

30회 기출문제

1-01 용어의 정의/특징

제약조건 : PK(기본키), Unique key(고유키), Foreign key(외래키), NOT NULL
Check(입력값 범위제한, NULL 무시)

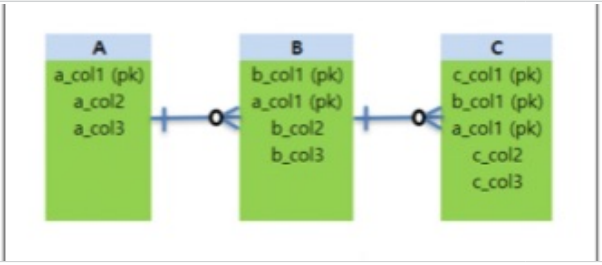
도메인 : 데이터 타입, 크기, 제약사항 지정함.

정규화 : 함수의 종속성 → 무손실 분해함.
(의미적 동등한 릴레이션 으로 분해, 자연조인하면 복원 O) ⇒이상현상방지
입력/수정/삭제 성능 향상 (반정규화는 조인 성능 향상)

식별자 : 주식별자 특징 (유최불준)
유일성 : 유일하게 인스턴스 구분
최천성 : 주식별자 속성 수 = 유일성 만족 최소 수
불변성 : 주식별자 값 바뀌지 X
존재성 : NOT NULL

ERD Entity 간의 필수, 선택 관계 살펴보기

1-06 ERD 이해



1) SQL작성 시 B를 제외할 경우 A와 C는 카테시안 조인.
↳ 조인 조건이 만족하지 않으면 카테시안 조인 발생

2) 3개 Table 모두 조인할 경우 조인 최소 조건은 3개
↳ n개 Table 모두 조인, 최소 n-1 조건 필요

3) B,C 에서 C의 데이터는 모두 B에 존재하므로 Outer Join 을 안 해도 된다.
↳ Table C에는 Table B의 데이터가 필수적 B.DATA C.DATA
↳ Table B에는 Table C의 데이터가 선택적 B.DATA NULL

4) B는 A의 데이터를 모두 포함하지 않는다.
↳ Table B에는 Table A의 데이터가 필수적
∴ A는 B의 데이터를 모두 포함한다.

1-10

Row chaining

: row의 길이가 길어서 여러 블록에 걸쳐서 저장

Row migration

: row의 데이터가 수정된 후에 다른 블록의 빈 공간에 저장된다.

↳ 같은 행의 데이터가 memory의 연속적인 위치에 저장되지 않음.

↳ 새로운 데이터의 추가될 시에 발생하는 현상이 아님.

↳ 디스크 I/O 발생 ↑ → 성능저하

So이) 1:1관계로 분리 (파티셔닝)

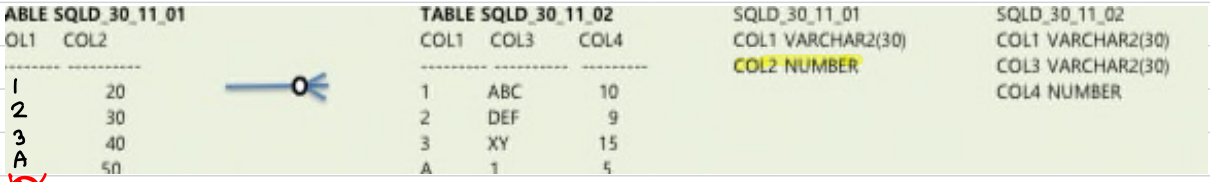
- 리스트 파티셔닝 : 기업 / 대량데이터
- 레인지 파티셔닝: 가장 많이 쓰임 / 숫자값
- 해시 파티셔닝 : 관리어렵 / 데이터 위치 모름.

2-11 SQL 오류

SQL오류1. WHERE절의 수치 부적절

문자형으로 선언된 column을 number 형과 비교연산할 때 Table 내 CHARACTER형과 number형이 혼재되어 있는 경우에 오류가 발생할 수 있다.

↳ORA-01722
: 수치가 부적절합니다.



1) SELECT A.COL1, A.COL2
FROM SQLD_30_11_01 A
WHERE A.COL1 > 0 → A.COL1 : VARCHAR (가변문자열) 비교연산
문자열 숫자 가변문자열과 숫자형 비교연산 (서로 다른 타입끼리 비교)

2) SELECT A.COL1, B.COL4
FROM (SQLD_30_11_01 A
INNER JOIN SQLD_30_11_02 B
ON (A.COL1 = B.COL1))
WHERE B.COL3 > 'X'

1	20	ABC	10
2	30	DEF	9
3	40	XY	15
A	50	1	5

3) SELECT COUNT(*)
FROM SQLD_30_11_01 A → 그냥 무시(?)
WHERE EXISTS (SELECT 'X'
FROM SQLD_30_11_02 B
WHERE A.COL2 = B.COL4)
↳조건을 만족하는 ROW Pair쌍이 존재 X
∴ RETURN NULL

재구성 : SELECT COUNT(*) → RETURN : NULL
FROM SQLD_30_11_01 A
WHERE EXISTS NULL

4) SELECT SUM (A.COL2)
FROM SQLD_30_11_01 A
INNER JOIN SQLD_30_11_02 B
ON (A.COL1 = B.COL1)
WHERE B.COL4 > '1'
문자열 숫자로 표현된 문자열

SQL 오류2.
서브쿼리 오류
↳ 1개 초과시 에러

2-12

서브쿼리에서는 1건을 초과한 레코드가 나오면 에러가 발생한다.

1) SELECT A.COL1
(SELECT COL3 FROM SQLD_30_12_02 B
WHERE A.COL1 = B.COL1) AS B_COL3,
FROM SQLD_30_12_01 A
WHERE A.COL2 > 10

↳ A.COL1이 'A'이고 B.COL1이 'A'일때 반환되는 COL3의 값이 (1과 2)로 1개를 초과하므로 오류가 발생한다.

2-13

그룹함수

- ROLLUP (A,B) : ① A,B 소계 ② A소계 ③ 전체 합계 ∴ 칼럼 순서에 따라 result가 달라짐.
- CUBE (A,B) : ① A,B (B,A) 소계 ② A 소계
③ B 소계 ③ 전체합계 ∴ 칼럼 순서에 영향을 받지 않음.
- GROUPING SET (A,B,C, ···)
: ① A 소계 ② B 소계 ③ C 소계 ··· ∴ 전체합계가 없음.

2-16

Optimizer 실행계획 → SQL 수행순서
↳ 액세스 기법 → 조인기법.
① INDEX → ② TABLE ACCESS (액세스 기법) → ③ NESTED LOOP JOIN (조인기법)

2-18

SQL 수행결과 (집계함수)

SQL>
SELECT COL1, SUM(COL2)
FROM SQLD_30_18_01
GROUP BY COL1

SQLD_30_18_01

COL1	COL2
1	100
1	NULL
2	100
2	200

⇓

COL1	COL2
1	SUM(100, NULL) = 100 (집계함수 연산 시, NULL 무시)
2	SUM(100, 200) = 300

SQLD_30_18_01

COL1	VARCHAR2(30)
COL2	NUMBER

2-20 Trigger (트리거)

- 테이블에서 DML 실행문이 실행되면 자동 실행
- COMMIT, ROLLBACK 불가
  ∵자동실행 되돌리기 X
- DELETE ON TRIGGER : OLD : 삭제 전 데이터, NEW : 삭제 후 데이터
- UPDATE TRIGGER : OLD : 업데 전 데이터, NEW : 업데 후 데이터
- DB 자체에 저장, 테이블. 뷰에 작성 및 저장 가능

2-21 NULL 처리 함수

- COALESCE ('AB', 'BC', 'CD') : NULL이 아닌 최초값 · AB
- CASE 'AB' WHEN 'BC' THEN 'CD' END = IF AB=BC THEN CD END : NULL
- DECODE ('AB', 'CD', 'DE') = IF AB=CD THEN DE END : NULL
- NULLIF ('AB', 'AB') => IF AB=AB THEN NULL : NULL

2-26 SQL 동일한 실행결과 (TOP N 쿼리 실행순서)

TOP N 쿼리 실행순서 : 정렬하고 일부데이터 추출

일반 SQL 쿼리에서는 ORDER BY를 가장 마지막에 수행한다.

↓

TOP() 쿼리에서는 ORDER BY 절의 데이터 정렬 후, 원하는 일부 데이터만 출력. (ORDER BY → select)

↓

ROWNUM 쿼리에서는
① ORDER BY가 있으면 ROWNUM을 WHERE 절에서 처리
② ORDER BY가 없으면 TOP() 과 같이 정렬 후 일부 데이터 추출.

[SQL] SELECT TOP(10) FIRST_NAME, JOB_ID ···③
FROM HR.EMPLOYEES ··· ①
ORDER BY SALARY ··· ②

↳ ① 테이블 HR.EMPLOYEES 에서
② SALARY 칼럼 값을 기준으로 오름차순 정렬하고
③ 상위 10개의 ROW를 추출

1) SELECT FIRST_NAME, JOB_ID ···③
FROM HR.EMPLOYEES ···①
WHERE ROWNUM <= 10 ···②
ORDER BY SALARY ···④

↳ ① 테이블 HR.EMPLOYEES 에서
② 상위 10개의 ROW를 추출해서
③ 출력할 칼럼을 FIRST_NAME 과 JOB_ID로 지정
④ 출력할 결과를 SALARY를 기준으로 정렬 <reverse>

2) SELECT TOP(10) WITH TIES FIRST_NAME, JOB_ID ···③
FROM HR.EMPLOYEES ···①
ORDER BY SALARY. ···②

↳ ① 테이블 HR.EMPLOYEES 에서
② SALARY 기준으로 오름차순 정렬
③ 상위 10위까지의 ROW를 추출 (단, 동석차 모두 추출) ⇒ 10개 초과할 수 있음.
∴ 순서는 올바른 순서이나, 추출되는 ROW의 개수가 10개를 초과할 수 있다.

3) SELECT FIRST_NAME, JOB_ID ···⑤
FROM (
SELECT FIRST_NAME, JOB_ID, ROWNUM RN ···②
FROM HR.EMPLOYEES ··· ①
ORDER BY SALARY ··· ③
)
WHERE RN <= 10 ···④

↳ ① 서브쿼리 / 테이블 HR.EMPLOYEES 에서
② / FIRST_NAME, JOB_ID 돌려, 컬럼으로 지정하고 ROW 나열순서에 따라 (~N 번호 부여
③ / SALARY 값을 기준으로 정렬
④ 메인쿼리 / RN이 10 이하인 10개의 ROW를 추출
⑤ / FIRST_NAME, JOB_ID 돌려,
RN 번호 부여가 SALARY 정렬 전에 이루어짐. (∴ 순서가 뒤바뀜!)

추출과정.

4) SELECT FIRST_NAME, JOB_ID
FROM (
SELECT FIRST_NAME, JOB_ID ···②
FROM HR.EMPLOYEES ··· ①
ORDER BY SALARY ··· ③
)
WHERE ROWNUM <= 10

↳ ① 서브쿼리 / HR.EMPLOYEES 에서
② 서브쿼리 / FIRST_NAME, JOB_ID 를 추출해서
③ 서브쿼리 / SALARY를 기준으로 정렬
↳ 메인쿼리 / 서브쿼리의 결과에서 ROWNUM을 부여해서
ROWNUM <= 10을 만족하는 ROW 10개를 추출

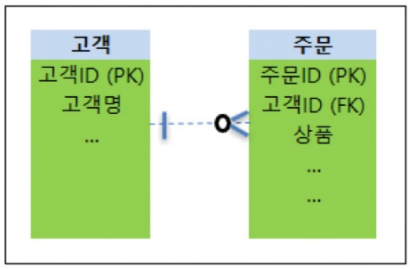
2-27 계층형 질의 / 셀프 조인 (START WITH ~ CONNECT BY ~)

- START WITH ~ : 시작위치 지정 , 루트 데이터 지정
- CONNECT BY PRIOR
  자식 데이터 지정 현재 읽은 칼럼
- 부모→자식 (순방향) : CONNECT BY PRIOR EMP_ID = MGR_ID.
  현재 직원ID 관리자번호
- 자식→부모 (역방향) : CONNECT BY PRIOR MGR_ID = EMP_ID.

2-30

2-34

34. 아래의 ERD 에서 아래 SQL 문을 순서대로 수행 할 경우 오류가 발생하는 구간은?



[테이블 정보] 주문 (고객ID) REFERENCES 고객 (고객 ID)

[SQL]
(1) INSERT INTO 고객 VALUES ('C001','AAA');
(2) INSERT INTO 주문 VALUES ('O001','C001','XXX');
(3) UPDATE 주문 SET 고객ID = NULL WHERE 주문ID = 'O001';
(4) INSERT INTO 주문 VALUES ('O002','C002','YYY');

[테이블 정보] : 테이블 생성 정보

(1) 고객 테이블에 ID='C001' 고객등록
(2) 주문 테이블에 고객ID='C001' 주문ID='O001' 등록
(3) 주문ID='O001'이 데이터의 고객ID=NULL로 변경 (고객의 주문취소요청)
(4) 주문테이블에 고객ID='C002'님의 주문ID='O002' 데이터 등록
Not Inserted in 고객.

∴ (4)

2-35

Cross Join / Natural Join 차이점에 대해 부적절한 것은?

- Cross Join = Cartesian product.
  ↳ 한쪽 테이블의 모든 행들과 다른 테이블의 모든 행을 조인.
  ∴ where 절에 조건 또는 on 조건절을 사용할수 없다.
- Natural Join은 동일한 이름의 컬럼에 대해 자동적으로 조인됨.
  ↳ 같은 이름이 아닌 다른 특정컬럼에 대해 필터링하고 싶으면 조건절을 추가할 수 있다.

"From A, B"
테이블 A, B에 대해서 어떠한 조건절도 없이 Join key가 없을 경우, From절은 두테이블의 Cross Join <Table가>을 반환.

update 발생
N ↓ ↓Y N
Next delete? →
row ↓Y

2-42 SQL 수행 → count(*)

SQLD_30_42_01

COL1	COL2	COL3
A	X	1
B	Y	2
C	Z	3
X	T	1

SQLD_30_42_02

COL1	COL2	COL3
A	X	1
B	Y	2
C	Z	3
D	가	4
E	나	5

[SQL]
```
MERGE INTO SQLD_30_42_01 A
USING SQLD_30_42_02 B
   ON (A.COL1 = B.COL1)
WHEN MATCHED THEN
   UPDATE SET A.COL3 = 4
      WHERE A.COL3 = 2
   DELETE WHERE A.COL3 <= 2
 WHEN NOT MATCHED THEN
   INSERT (A.COL1,A.COL2,A.COL3) VALUES(B.COL1,B.COL2,B.COL3);
```

① MERGE INTO ~ (테이블의 데이터를 병합)
② USING ~ ON ~ (1개 또는 2개의 테이블을 병합. 1개 테이블 이용시 USING DUAL 이라고 씀)
③-1. WHEN MATCHED THEN ~ 테이블에 존재하는 데이터이면...
종속적 (1) UPDATE 수정하고
(2) (DELETE) ★수정된 데이터에 대해 삭제할 수 있음.
③-2. WHEN NOT MATCHED THEN 테이블에 존재하지 않으면
INSERT 데이터 삽입

2-45

45. 아래와 같은 데이터가 들어있는 테이블에서 아래 sql 이 수행되었을 때의 결과 건수는? 2건

계층번호	상위계층번호
1	null
2	null

Table : SQLD_30_45

계층번호	상위계층번호
1	null
2	null
4	1
5	1
6	2
7	2
8	4
9	5
10	6
11	7

[SQL]
```
SELECT  LEVEL,
    LPAD('** ', (LEVEL -1 )*2 ,'  ') || 계층번호 AS 계층트리,
    계층번호,
    상위계층번호
FROM SQLD_30_45
START WITH 상위계층번호 IS NULL
CONNECT BY 계층번호 = PRIOR 상위계층번호;
```

· LPAD ('**', (LEVEL-1)*2, ' ') : 문자열의 길이가 (LEVEL-1)x2 를 만족할 때까지 값 문자열의 왼쪽 ' ' 을 채움문자로 채운다.
(값 / 총 문자열 길이 / 채움문자)

· START WITH : 계층형 질의에서 루트(시작) 지점 지정하기.
START WITH 상위계층번호 IS NULL : <1,2>
· CONNECT BY : 다음노드 연결방식 지정
CONNECT BY 계층번호 = PRIOR 상위계층번호
= CONNECT BY PRIOR 상위계층번호 = 계층번호 (PRIOR 부모 = 자식) // 역방향전개

root에서 시작 → (계층번호 : 1 or 2) PRIOR 상위계층번호 : NULL
CONNECT BY 계층번호 = NULL (존재안함)

2-46

46. 아래의 SQL 구문은 컬럼의 데이터 타입을 변경하는 SQL 구문이다. SQL 구문을 완성하시오. (SQL SERVER 기준임)

```
[SQL]          (     ) DEPT    (     ) VARCHAR(30) NOT NULL;
```

- 컬럼의 데이터 타입 변경 : 컬럼의 변경 → 테이블의 변경.

ALTER TABLE DEPT ALTER COLUMN VARCHAR(30) NOT NULL;

2-49

49. 아래와 같은 결과가 나오도록 SQL 을 작성하시오

```
[RESULT]          TABLE : SQLD_30_49
                    회원ID    RANK    주문금액
                    ----------------------------------
                      B         1        450
                      G         2        255
                      F         2        255
                      H         3        100

[SQL]
                  SELECT 회원ID,
                         DENSE_RANK() OVER(ORDER BY (          ) ),
                         주문금액
                  FROM SQLD_30_49;
```

- DENSE_RANK( ) : 동석차를 인정 / 다음순위는 현재순위+1

주문금액을 내림차순 정렬한 결과에 따른 RANK

⇒ ORDER BY 주문금액 DESC