<데이터베이스 실습 복습 정리>

161031 to 161107 (VIEW to USER/SCHEMA)

이번 CREATE 문에 쓰이는 새로운 개념들인 VIEW, SEQUENCE, SYNONYM을 공부 해 볼 계획이고, 차후에 DBA(데이터베이스관리자)의 입장으로서 사용자와 스키마에 대해서 관리 하는 연습 차원에서 쓰이는 USER/SCHEMA 개념에 대해서 숙지를 하는 기회를 가져보도록 하자.

[요약 중점 사항]

- VIEW를 이용해서 현재 테이블에 있는 값들에 관하여 컬럼들로 간략히 축소해서 보여주는 역할을 한다.
- SEQUENCE를 이용해서 각 세부 정보에 일정 번호를 유용하게 책정할 수 있다.
- SYNONYM을 이용해서 테이블의 이름을 대체할 수 있는 개념을 배워본다.
- USER/SCHEMA 개념을 배워봄으로서 GRANT 연산자를 이용하는 연습을 한다.

---

1. VIEW at 161031

필자는 동경 여행을 가고 싶다. 가령 동경을 3박 4일 동안 가게 된다면 동경의 전 지역을 돌아보는데 있어서 무리일 것이다. 그래서 필자가 가보고 싶은 주요 구경거리들을 골라서 아사쿠사, 오다이바, 아키하바라, 우에노, 시부야... 이런 식으로 지역을 선정해서 여행 계획을 짜면 4일 동안 비록 모든 지역을 갔다 오지는 못해도 대표적인 여행지를 다녀오게 되어 필자가 만족하게 될 것이다. 이처럼 View는 여행 도시 중에서 주요 관광지를 잡아서 짜듯이, 사용자가 테이블 내에서 꼭 필요로한 컬럼들을 골라내서 저장시키는 개념으로 생각하면 된다.

1) View의 주요 특성

(1) 가상의 테이블

-> 테이블 내부에는 데이터가 저장되지만, View 내부에는 SQL의 개념들로만 저장이 된다. 가령 쉽게 이야기하면 치킨을 시킬 때 사진에 있는 건 풍부해 보이는데 실제로는 뭐같이 배달이 오는 원리처럼, 사용자가 필요로 한 데이터들을 View를 통해 그 데이터 자체를 저장하는 것이 아닌 그 데이터의 포인터를 저장하는 원리로 생각을 하면 된다.

(2) 값이 변경되어도 최근에 변경된 값을 가져옴으로서 정확성을 보장한다.

-> 우리가 배웠던 CTAS(Create Table AS)를 이용하면 장땡인 줄 알았지만, 원본 테이블에서 변경되는 값에 대해서는 보장을 안 해준다. 그래서 View를 이용하게 되면 원본 테이블에 저장된 값이 변경되더라도 데이터의 포인터를 가리키기 됨으로서 갱신된 값에 대해서도 그대로 적용을 해서 보여주니깐 CTAS보다 매우 효율적으로 이용이 된다.

(3) Join을 View에서만 이용해서 사용자들에게 편의를 제공한다.

-> 테이블에서 또 다른 컬럼이 필요로 한 값들에 대해서는 join문을 써야 되어서 문장 처리 시간이 많이 걸리는 반면, View 하나에 join문을 테이블에서 써주는 원리를 쓰면 데이터가 아닌 포인터를 다루게 되어서 처리 속도가 오히려 뛰어나서 사용자들에게 편의를 제공하게 된다.

(4) DML을 수행할 수 있다.

-> View에서 또한 DML을 수행 할 수 있지만 일부 한정에서만 가능하다. 어떤 경우에 대해서 처리가 가능한지에 대해서 아래 정리한 내용을 참조하면 큰 도움이 된다.

(참고) 뷰를 통한 데이터 조작

♦ 단순 뷰에서는 대개 DML 작업을 수행할 수 있다.

♦ 뷰에 다음 항목이 포함되어 있으면 행을 제거할 수 없다.

- 그룹 함수
- GROUP BY 절
- DISTINCT 키워드

♦ 뷰에 다음 항목이 포함되어 있으면 뷰의 데이터를 수정할 수 없다.

- 그룹 함수
- GROUP BY 절
- DISTINCT 키워드
- 표현식으로 정의된 열

♦ 뷰에 다음 항목이 포함되어 있으면 뷰를 통해 데이터를 추가할 수 없다.

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

이는 지난 번 교수님이 정리한 글을 그대로 복붙하였다. 그래서 필자가 왜 이렇게 안 되는지에 대하여 유인물를 따로 준비하였으니 161103_VIEW_DML_적용 파일을 참조해서 원리를 이해하기 바란다.

2) View의 구조

View를 첨가할 때 쓰는 문장은 다음과 같이 살펴 볼 수가 있다.

```
Create [Or Replace](1) [force (참조)] View 뷰_이름 [AS] (4)
AS
  (SubQuery)
[With Check Option](2)
[With Read Only](3)
```

(1) Or Replace : 같은 이름의 View가 있는 경우 삭제 후 다시 생성하게 하는 문장이다. View는 애석하게도 Alter 문장을 쓰지 않는다. 그래서 이로 대체해서 다시 작성해 나가면 도니다.

(2) With Check Option : 주어진 제약조건에 맞는지 획인을 하는 역할을 하는데, 무엇보다 중요한건 여기서 제약 조건은 서브쿼리에서 쓰이는 Where문이 될 수 있다. Having 문은 애석하게도 Group 함수, Group By가 존재되어 수정/삭제/삽입 자체를 못하니...

아래 문장은 봉급이 5000이상인 직원들로만 저장하는 뷰를 만든 문장이다.

```
Create Or Replace View EMP_Sal_View
 AS
  Select employee_id, last_name, salary
  from employees
  Where salary>=5000
With Check Option;
```

이처럼 밑줄 친 부분이 with check option의 제약 조건이 되면서 이 조건을 위배하게 되면 값이 추가 되지도 않고 변경도 되지 않는다.

(3) With Read Only : Select만 가능하게 전용 뷰를 형성한다. 그렇다고 Select 문에서 컬럼을 함수로 표현해서 출력한다는 것을 못한다는 것이 아니다. 오로지 DML 문장 자체를 쓸 수가 없다는 의미로 받아들이자.

(4) AS : 늘 그래왔듯이 View도 별명을 지어줄 수 있다. 옆에 이용하고 싶은 이름을 작성해서 이용하자.

(참조) Force : Force는 강제로 삽입이라는 뜻이 있다. 본래 기본으로는 NoForce로 적용이 되는데 이는 기본 테이블, 즉 현재 소지하고 있는 테이블이 존재하게 되면 추가가 된다는 뜻이다. 허나 Force를 적용시키면 소지하고 있는 기본 테이블이 있든 말든 추가가 되는 개념이다.

3) Simple-View(단순 뷰)

어느 View를 추가할 때나 마찬가지겠다만, 생성할 서브쿼리에 join 문까지 쓰지 않고 간략히 사용자가 필요로 한 컬럼들에 대해 골라내서 Select문으로 서브쿼리를 만들어서 적용하는 View이다. 간략한 예로 직원 테이블에서 부서 번호가 50, 60, 80번인 직원 ID, 성, 부서번호를 출력하는 View를 만들어 보자.

```
Create Or Replace View EMP_VIEW_DEPT568
AS
  Select employee_id, last_name,
department_id
  from employees
  where department_id IN(50, 60, 80);
```

이처럼 작성하게 되면 Employees 테이블에 존재하는 컬럼들에 대해서 값을 불러오고 난 뒤에 Where 문에 해당되는 제약 조건들을 확인해서 값들의 포인터를 저장시키게 된다. 여기서 두 번 이상 말하게 되지만 뷰는 데이터가 저장되는 것이 아니라 포인터의 원리가 저장되어 평소에 뷰에는 아무런 데이터가 존재하지 않고 데이터에 대한 포인터들만 저장된다고 생각을 하면 된다. 그래서 View에서는 제약조건(Constraint), 인덱스(Index)를 절대로 추가할 수 없으니 참조하길 바란다.

4) Complex-View(복합 뷰)

복합 View는 한 마디로 우리가 컴퓨터를 이용하게 되면 모니터를 2개 이상 쓰는 경우가 있을 것이다. 하나는 작업용, 하나는 검색용 이런 식으로 말이다. 이처럼 2개 이상의 테이블을 공동된 컬럼을 통해서 값을 참조하여 Select 문으로만 번거롭게 작성하지 않고 뷰로 계속 참조할 수 있도록 도와준다. 다음과 같이 위의 뷰에서 부서 이름 컬럼만 새로 추가해 보겠다.

```
Create Or Replace View EMP_VIEW_DEPT568
AS
  Select e.employee_id, e.last_name,
e.department_id, d.department_name
  from employees e join departments d
  ON e.department_id=d.department_id
  where e.department_id IN(50, 60, 80);
```

여기서 Join 문장을 서브쿼리로 써서 뷰를 추가하게 되면 자연스럽게 View가 형성이 되어서 사용자에게 편의를 제공하게 된다.
허나 주의해야 할 점이 있다. 가령 아래의 문장을 살펴보자.

```
Update Emp_View_Dept568
set department_name='Administration'
where last_name='Sonoda';
```

이처럼 작성을 하게 된다면 성이 Sonoda인 직원의 부서를 Administration으로 바꿔버리는 문장이다. 허나 보존되지 않은 테이블로 대응한 열을 수정할 수가 없기 때문은 물론이고, 자동적으로 Sonoda의 부서 번호가 바뀌지도 않기 때문에 수정하는데 있어서 주의를 요하길 바란다...

5) Inline-View(인라인 뷰)

어렸을 때 인라인 스케이트를 많이 타봤을 것이다. 이는 신발 하단 부 내부에 바퀴 4개를 일직선으로 연결해준다. 이처럼 인라인 뷰는 사용자가 필요로 한 컬럼들을 인라인 스케이트의 바퀴처럼 묶어두는 원리로 from 문에서 적용을 시켜 필요로 한 컬럼들의 일부를 가상 테이블로 묶어서 일회성으로 이용되는 원리이다. 기존에 이용했던 서브쿼리(Subquery)와 비슷한 원리이지만, 서브쿼리는 Where문에서 쓰고, 인라인 뷰는 From 문에서 쓴다. 뷰는 가상 테이블로서 테이블의 일부로 인정하기 때문이다.
물론 사용자가 뷰를 따로 만들어서 from문에 박으면 되는데 일회용으로 쓰는 경우만 쓰길 권장한다.
아래는 부서번호, 부서이름, 부서별 봉급 평균을 골라내는 문장을 인라인 뷰를 이용해서 만들었다. 아래 문장을 살펴보자.

```
Select d.department_id, d.department_name,
e.sal
From (Select department_id,
        Trunc(AVG(Salary), 0) sal
        From employees
        Group By Department_id) e join
        departments d
On e.department_id=d.department_id;
```

이처럼 작성하게 되면 우선 괄호 안에 있는 인라인 뷰를 통해서 가상의 테이블을 추출해서 Employees 안에 있는 department_id, 부서 별 Salary의 평균을 정수로만 표현한 값들이 저장되어 있어서 이를 참조해 위의 문장을 출력하게 도와준다. 또한 인라인 뷰를 이용해서 병합도 할 수 있고, 등수도 매길 수 있는 다양한 응용방법이 있으니 심심하면 한 번 도전해 보는 것도 나쁘지 않겠다.
<추가 설명> 뷰와 서브쿼리에 대한 차이점을 알고 넘어가게 되면, 뷰는 테이블을 다루는 경우에 많이 쓰는 것이고, 서브쿼리는 컬럼에 대한 데이터를 다루는 경우에 많이 쓰게 된다.

6) Drop View

DDL로 만들어진 모든 객체들은 Drop를 언제든지 쓸 수 있다. 뷰도 이 문장 하나로 삭제를 하면 되니 참조하자.
Drop View 삭제할_뷰_이름

2. SEQUENCE at 161102

우리가 야채 이름 중 하나인 도시락 식당에서 밥 먹으러 가서 주문을 하면 직원 분께서 번호표를 줄 것이다. 가령 필자가 좋아하는 무슨 콤보 도시락을 시키게 되면 직원 분이 98번 무슨 콤보 도시락 나왔어요 라고 할 것이다. 이처럼 시퀀스는 많은 데이터들이 추가되면서 순서를 조정해주는 역할을 해준다. 방금 설명한 식당 주문 순서도 될 것이고, 학번, 지하철 역 번호, 도서관 책 번호... 실생활에서 없어선 안되는 개념들을 SQL에서는 이 녀석이 수행을 한다. 그래서 번호표를 매기는 역할을 해주는 Sequence에 대해 자세히 알아보도록 하자.

1) Sequence의 구조

Sequence를 쓰기 전에 어떤 구조로 되어 있는지에 대해 알아둬야 하겠다. 다음과 같은 문장을 통해 구성을 살펴보자.

```
Create Sequence 시퀀스_이름
  [Increment By 숫자] // 1
  [Start With 숫자] // 2
  [MaxValue 숫자] // 3
  [MinValue 숫자] // 4
  [Cycle / Nocycle] // 5
  [Cache 숫자 / NoCache] // 6
```

1 : Increment는 증가라는 뜻이다. 그래서 사용자가 증가하기 원하는 숫자를 입력하면 그만큼 숫자가 더해지는 개념이다. 기본 값은 1로 되어 있다. 가령 우리가 쓰고 있는 Departments의 Department_ID는 10씩 추가가 되어있는 것으로 봐서 이를 이용했다고 생각하면 된다.
2 : 시작하는 숫자의 시발점을 설정하는 것이다. 기본 값은 1부터 시작이 되지만, 현재 우리가 쓰고 있는 Employees 테이블의 Employee_ID는 100부터 시작을 하게끔 작성이 되어 있어서 이를 이용했다고 생각하면 된다.
3 : 생성이 가능한 시퀀스 최댓값이다. 가령 일리네어 콘서트에서 티켓 구매자가 너무 많아 좌석을 이용하는 인원의 티켓 번호를 한정하게끔 하는 역할을 해준다.
4 : 최솟값에 대해서 다루는데 경우에 따라 다룬다.
- Cycle를 도는 경우에 최솟값으로 돌아가서 시작을 하게끔 한다. 방금 좌석 티켓이 떨어진 경우에 입석 티켓 번호를 새로 지정해서 입석 예약번호로 새롭게 시작을 한다는 원리로 생각을 하면 된다.
- 물론 Increment에 음수도 작성이 가능해서 번호를 감소할 때 최솟값을 정의해서 형성하는 원리라고 생각하면 된다.
5 : Cycle는 시퀀스 번호를 순환할 때 이용한다. 신도림, 홍대, 을지로, 건대, 강남, 잠실을 지나가는 서울 2호선에서 각 역마다 번호를 매길 때 이용하는 원리로 생각하면 된다. 다만 반복 문장을 이용하기 위해서는 **Cache의 개수가 꼭 정의되어 있어야 한다.**
6 : Cache는 운영체제론 시간에 레지스터와 메인 메모리의 차이를 극복시켜주는 메모리 종류의 일부로 배웠다. 이런 원리로 시퀀스의 생성속도를 개선하기 위해 캐싱 여부를 지정해주는 원리로 생각하면 된다. 가령 일리네어 콘서트를 예약하는데 사용자가 너무 많아서 서버가 끊기는 경우가 있다. 사용자들의 희비가 갈라지는 점은 애석하겠지만... 번호 요청이 들어오게 되면 캐시를 이용해서 번호를 즉시 반환하는 역할이라 보면 된다. Cache를 이용하는 이유는 아래에서 상세히 설명하겠다.

2) Sequence 사용하기

Sequence의 숫자를 불러오기 위해서는 아래 2가지를 이용해서 불러온다는 원리를 알고 넘어가야 된다.

-> NextVal : 다음 값을 불러오는 키워드. NextValue를 줄여서 쓴 키워드이다.
-> CurrVal : 현재 시퀀스가 가리키는 값이다. CurrentValue를 줄여서 쓴 키워드이다.

시퀀스 이용하는 방법을 3가지 모델로 나눠어서 설명하겠다. 아래를 쉽게 설명하기 위해서 다음과 같은 테이블을 하나 만들었다.

```
Create Table Station
(ID Number(4), Station_Name Varchar2(20) Not Null);
```

쉽게 말해서 역 번호와 역 이름 2개의 컬럼으로 구성되었다. 역 번호에 시퀀스를 적용하여 값을 추가하는 원리로 생각하면 된다.

2-1) 일반적인 시퀀스 이용

```
시작 번호 : 409
증가 빈도: 1
최대 번호 : 415
반복 여부 : Not Yet...
캐시 수 : Not Yet...
```

시퀀스의 이름은 Station_SEQ_LINE4로 만들겠다.
다음과 같은 시퀀스는 다음과 같이 만들 수 있다.

```
Create Sequence Station_SEQ_Line4
  Increment By 1
  Start With 409
  MaxValue 415;
```

여기서 409번부터 각 역에 대한 정보를 추가하는 문장을 작성하겠다. 처음부터 이렇게만 작성을 해주면 이름만 조정하면 되니깐 편리하게 추가할 수 있다.

```
Insert Into Station
Values(station_SEQ_Line4.nextVal, '당고개');
Insert Into Station
Values(station_SEQ_Line4.nextVal, '상계');
....
```

=> 이처럼 추가해서 Station의 결과를 살펴보면 다음과 같이 된다.

```
409     당고개
410     상계
411     노원
412     창동
.... // 여기서 실제로 역번호가 415번인 미아역을 삽입한 뒤 다음 역인 미아사거리역을 추가하는 문장을 작성하면 어떻게 될까? 아래 문장을 통해 살펴보자.
Insert Into Station
Values(station_SEQ_Line4.nextVal, '미아');
```

-> 현재 415번인 미아역에서는 정상적으로 추가된다.

```
Insert Into Station
Values(station_SEQ_Line4.nextVal, '미아사거리');
```

-> 여기서 삽입을 하게 되면, nextVal인 값이 416으로 나오게 되어서 위의 MaxValue 조건을 보면 415

번까지만 삽입하게 되기 때문에 현재 시퀀스의 제약에 어긋나게 되어 최종적으로 값이 추가되지 않는다.

현재 적용 중인 시퀀스의 값이 어느 값인지에 대해서 알아보기 위한 문장은 아래와 같은 문장으로 작성하면 쉽게 알아볼 수 있다.
Select Station_SEQ_Line4.CurrVal
from dual;
그러면 현재 미아역까지만 추가가 된 상태이기 때문에 나오는 값은 415번이고, MaxValue의 값에 어긋나서 다음 값은 추가가 되지 않는다.

2-2) 순환 시퀀스 이용

시작 번호 : 547 증가 빈도 : 1 최대 번호 : 550 반복 여부 : Yes 반복 시 시작점 : 533 캐시 수 : 3

시퀀스의 이름은 Station_SEQ_Line5로 정의하였다. 그래서 5호선에 실제로 역 번호가 547인 천호역부터 추가해 보겠다.
Insert Into Station
Values(station_SEQ_Line5.nextVal, '천호');
Insert Into Station
Values(station_SEQ_Line5.nextVal, '강동');
Insert Into Station
Values(station_SEQ_Line5.nextVal, '길동');
Insert Into Station
Values(station_SEQ_Line5.nextVal, '굽은다리');
이처럼 계속 삽입을 하게 되면 굽은다리에서 역 번호가 550번으로 정의되고 끝나게 된다. 하지만 반복이 완료되면 다음 숫자는 533으로 정의가 된다. 그래서 다음 문장을 광화문역부터 시작을 하면 어떻게 될까?
Insert Into Station
Values(station_SEQ_Line5.nextVal, '광화문');
Insert Into Station
Values(station_SEQ_Line5.nextVal, '종로3가');
그렇게 되면 다음과 같이 삽입이 되어 결국 아래와 같은 결과가 나오게 된다.
547 천호
548 강동
549 길동
550 굽은다리 -> 여기서 시퀀스의 제약조건을 벗어나서 533으로 복귀를 시키고 난 뒤에
---
533 광화문
534 종로3가 -> 역번호가 533번부터 재정의해서 계속 순차적으로 값이 지정이 된다.

물론 533번부터는 계속 추가하게 되어 550번까지 도달을 하게 되면 다시 533번부터 재정의되어 추가되는 개념이라서, 이를 이용하는 컬럼이 UNIQUE, Primary Key 제약 조건이 걸리지 않는 곳에 쓰길 권장한다.

2-3) 역순 시퀀스 이용

시작 번호 : 352 증가 빈도 : -1 최소 번호 : 342 반복 여부 : Not Yet... 캐시 수 : 4

시퀀스의 이름은 Station_SEQ_Line3으로 정의하였다. 이번에는 증가하는 것이 아닌 감소하는 문장으로 재작성해 봤다. 시퀀스도 감소로서 정의를 할 수 있다는 점을 보여주기 위해서이다.
Create Sequence Station_SEQ_Line3
Increment By -1
MinValue 342
MaxValue 352
Start With 3
Cache 4;
여기서 감소를 하는 값에 대해서는 Start With 키워드가 아닌 MaxValue로 정의를 해야 된다는 점에 대해 알려둔다.

2-4) 시퀀스에 캐시를 이용하는 이유
왜 시퀀스에는 왜 캐시를 이용할까? 시퀀스의 값이 증가하는 속도와 실질적으로 DML에서 값을 추가할 때 중간에 서버가 끊기거나 다른 테이블에 작업을 하는 경우에 있어서 쓰는 것이다. 아래의 3가지 경우에 써서 이용한다고 생각을 하면 된다.
-> 롤백이 발생하는 경우
-> 시스템 작동이 중단되는 경우
-> 시퀀스가 다른 테이블에서 사용되는 경우
이런 경우들 중 대표적인 시스템 작동이 중단되는 경우에 살펴보겠다. 아래는 쉽게 설명하기 위해서 1부터 순차적으로 저장이 되면서 Cache 키워드가 어떤 원리로 적용되는지에 대해 설명하려고 작성하였다.
Create Sequence SEQ_SEV_TEST
Increment By 1
Start With 1
MaxValue 100
Cache 10;

서버에 이상이 나지 않는 이상 우리가 평소에 nextVal를 하게 되면 1, 2, 3... 이런 식으로 숫자가 나온다. 갑자기 서버 점검 때문에 SQL이 종료된 경우를 살펴보면 사용자가 현재 값이 다음과 같다면 캐시를 통해서 다음 값은 이를 가리키게 된다.

사용자가 반환한 값	서버가 끊기고 난 뒤... -->	캐시에 의한 다음 값
1~9		10
10~19		20
20~29		30
30~39		40
...		...

이를 살펴보게 되면 무슨 소리인지 잘 모를 것이다. 필자가 이해하기 쉽게 아래와 같이 그림으로 설명하겠다. 색칠한 부분은 서버가 끊기면 비상으로 불러낼 값이다.

(1) 사용자가 시퀀스를 정의하고 난 뒤에는 첫 번째 값은 0을 가리키게 된다.

0									

(2) 사용자가 1를 반환하고 난 뒤에는 0이 캐시 사이즈만큼 더해져서 바뀌게 되어 10으로 바뀌게 된다.

10	1								

(3) 사용자가 계속 입력해서 9까지 반환하고 10을 반환하게 되면 20으로 처음 캐시 값은 바뀌게 된다.

10	1	2	3	4	5	6	7	8	9

10을 반환하게 되면...

20	1	2	3	4	5	6	7	8	9

(4) 사용자가 다음 11를 반환하면 1은 11로 재저장이 되고, 나머지 값도 11처럼 값이 바뀌게 되는 원리이다.

20	11	...	...	...	...	...	...	...	...

...

20	11	12	13	14	15	...	...	...	...

...
(5) 만일 여기서 15까지 반환이 되고 서버가 끊기게 된다면 어떻게 될까? 우선 20을 반환하고, 계속 다음 값을 재저장하는 원리로 이용이 된다.

20	11	12	13	14	15	...	...	...	...

-> 여기서 15까지 저장이 되어도 서버가 끊기면 20을 반환시킨다.

30	21	22	23	24	25	...	...	...	...

-> 그 다음 21부터 계속 값을 추가시키고 색칠된 값은 30으로 바뀌게 된다.

이런 그림을 통해서 보면 서버가 끊기거나 롤백을 하는 경우에 비상으로 값을 불러오기 위해서 쓰는 원리로 생각을 하면 된다. 쉽게 이야기해서 우리가 알람을 맞춰놓고 자는 경우가 있는데 오전 5시 55분에 일어난다고 했는데 더 졸려서 알람 어플을 이용해서 오전 6시 15분으로 지연을 해두는 원리로 생각하면 된다.

3) Alter, Drop Sequence

물론 시퀀스도 조회가 가능하고, 수정/삭제가 당연하다. Alter 문을 사용하는 객체는 우리가 배워온 개념들 중에서는 사실 상 많지 않았다. 예를 들어 테이블의 구조를 변경하는 Alter Table, 날짜 형태를 변경하는 Alter Session 이렇게 말이다. 허나 Sequence라는 녀석도 엄연히 값을 변경할 수 있는 개념이니 Alter 문으로 수정하는 방법을 알아야겠다.

3-1) Alter문으로 수정하기
4페이지에 있는 SEQ_SEV_TEST를 다음과 같이 변경을 해 보겠다.
예를 들어 최댓값을 200으로 하고, 증가 빈도를 2로 바꿔보는 문장이다.

```
Alter Sequence SEQ_SEV_TEST
  Increment By 2
  MaxValue 200;
```

이처럼 시퀀스에 적용이 되는 값들을 Alter문으로 적게 되면 손쉽게 변경할 수 있으니 알아두길 바란다. 허나 문제는 여기서 start with라는 부분은 여기서 수정이 불가능하다. 이런 경우에는 시퀀스를 다시 만들든지 해야 한다.

3-2) 시퀀스 삭제하기
Drop Sequence 시퀀스_이름
Drop을 이용하면 되는데 여기서 주의할 점이 삭제되는 시퀀스를 사용하는 DML이 롤백 되어도 이미 시퀀스는 Recyclebin으로 들어가 버렸기 때문에 입력되는 번호의 차이가 생기게 된다.

3-3) 시퀀스 조회하기

```
Select sequence_name, min_value, max_value,
Increment_by, cycle_flag "순환 여부",
order_flag "정렬 여부", cache_size, last_number
"마지막 생성 값"
from user_sequences
...
```

각 시퀀스의 요소들로 사용자가 어떤 시퀀스를 추가했는지에 대해 알 수 있으니 어떤 시퀀스를 이용했는지 알고 싶다면 이를 이용하면 되겠다.

---

3. SYNONYM(동의어) at 161102

래퍼 dok2는 애칭이 도끼 하나만이 아니다. Gonzo, Notorious Kid라고도 칭하는 경우가 많다. 이처럼 우리가 컬럼에서 흔히 Alias의 개념들을 주로 활용을 했고 또한 join문에서 employees라고 쓰긴 너무 기니깐 emp, e 이런 식으로 축약을 한 경우를 많이 봤을 것이다. 하지만 Synonym을 이용하게 되면 테이블의 이름을 여태껏 설명한 Alias를 이용하지 않아도 사용자가 원하는 이름으로 별명을 붙어줄 수 있다. 어떤 원리로 이용하는지 살펴보자.

1) Synonym 구조

Create [public] Synonym 동의어_이름
for [스키마 이름].대상객체
우리가 쓰고 있는 스키마는 HR이다. 그래서 스키마 이름을 따로 정의를 하지 않는다면 HR 내에서만 그 별명을 칭할 수 있다. 허나 모든 이용자가 사용하도록 정의하고 싶다면 Create 옆에 public을 첨가해주면 된다. 가령 Employees 테이블 이름을 직원들로 바꾸고 싶다면(앞서 설명했지만 한글로도 테이블 이름이 정의가 가능한데, 이는 사용자의 편의를 위해 잠깐 이용하는 것이다.) 다음과 같이 작성하면 된다.

```
Create Synonym 직원아찌들들
for HR.Employees;
```

물론 대상 객체이니 테이블 이외에 제약 조건, 인덱스, 시퀀스, 뷰 등등 여러 객체들이 된다는 사실도 알아두자.

2) 스키마와 Synonym

유저는 사용자라는 뜻을 이미 알고 있을 것이다.
스키마는 일상에서 계획이나 구도를 스키마로 칭하는 경우가 대부분이지만 데이터베이스에서는 구성하는 모든 객체들에 대해 적용한 문장이다. 스키마는 사용자에게 하나씩 생기게 된다. 그리고 스키마의 구조에 대해서는 부지기수한데, 쉽게 나뉘어 보면 데이터의 실제 값, 데이터의 데이터 값(메타 데이터)로 2가지 분류로 나뉠 수 있다.
만일 사용자 이름이 User1, User2가 있다면 자동적으로 Employees 테이블이 저장되어 있다고 가정을 한다. 그러나 각 유저 별로 스키마 내에 있는 테이블의 값은 이름은 같은데 값들이 같은지 어찌한지에 대해서는 모른다. 이런 경우에 테이블의 이름이 서로 달라서 다른 이름으로 칭할 필요가 있다. 그래서 User1의 Employees 테이블 이름을 직원들로 지어

주면 User2와의 혼란을 방지하기 때문에 스키마 내부에서 사용자 권한을 주어서(Grant를 이용) 별명을 주어야지 아래의 문장에 대해 실행이 가능하다.

Grant Create Synonym To User1 // 일반적으로 User1이 동의어를 만들 수 있도록 권한을 부여한다.
Grant Create Public Synonym To User1 // User1이 사용자들을 위한 공용 동의어를 만들 수 있는 권한을 부여하는 문장이다.

User1>
Create [Public] Synonym 직원들
for Employees // 위 문장들 중에서 첫 번째 문장을 작성하게 되면 public 이외에 동의어를 작성할 수 있게 되는데, public을 포함시켜주면 모든 사용자들이 User1이 만든 동의어를 통해 그 객체에 대해 참조를 하게 된다.

3) Synonym 조회하기, 삭제하기

Select synonym_name, table_owner,
table_name
from User_Synonyms ...
이처럼 작성해 주면 현재 사용자가 저장한 Synonym이 무엇인지에 대해 알 수 있으니 참조하길 바란다.
물론 Drop 문을 이용해서 이처럼 작성하면 동의어를 삭제할 수 있으니 참조하길 바란다.
Drop Synonym 직원들;

---

4. User & Schema at 161107

우리가 여태껏 이용한 User의 이름들 중에서 sys, hr를 대표적으로 둘 수 있는데 이러한 User와 Schema에도 알아두면 피가 되고 살이 되는 개념들이 있다. 그래서 이번 차시에는 User를 통해서 계정을 생성하고, 계정에 대한 권한을 부여/해제를 할 수 있는 방법에 대해 공부해 보도록 하겠다.

1) User와 Schema의 정의

우리가 게임이나 이메일을 관리할 때 각 사용자 계정을 형성하고 사용자를 관리하게끔 형성을 하듯이 데이터베이스에도 User와 Schema의 개념으로 나뉘게 된다. 아래의 3가지 용어에 대해서는 숙지하고 갈 필요가 있으니 알아두자.
User : 오라클에 접속하기 위해 사용되는 사용자를 의미한다. 새로이 만들어진 계정은 DBA에게 권한을 받아야 Schema에 관련된 제어를 받을 수 있다.
Schema : 사용자들이 만든 Object(객체)들의 모음을 뜻한다. 객체에는 우리가 흔히 Create 문장으로 만드는 Table, Index, Sequence, Constraint 등등이 될 수가 있다. 그렇지만 User와 Schema의 개념은 애매모호하게 설명을 하는 경우가 대다수이다.
DBA(데이터베이스 관리자) : 사용자들에 대해서 관리를 하는 역할을 하면서, System 관련 권한을 줄 수 있는 역할을 한다고 보면 된다. 우리가 써왔던 DBA의 개념은 SYS 계정을 예로 살펴볼 수가 있다.

2) 권한(Privilege) 관리하기

예를 들어 필자가 집에서 자주 마시는 쌍화차 한 박스를 프젝실에 구비했다고 가정을 한다. 쌍화차를 마시고 싶은 사람들이 있겠지만 분명 쌍화차는 필자의 것이다. 그렇지만 필자에게 허락을 받으면 쌍화차를 타 마시는데 딱히 문제가 없다. 이처럼 사용자들이 DBA에게 테이블을 새로 만든다, 인덱스를 삭제한다 등등 보고를 하고 DBA이 그 사용자들에게 권한을 주는 원리를 쓰는 문장이 Grant이다. 지난번에 Synonym을 공부했을 때 잠깐 이야기했었지만 이번에는 주요 권한에 대해 대표적인 것들에 대해 알아보고 어떻게 권한을 주는지에 대해 설명하겠다.

2-1) System 관련 Privilege
여기서는 요약 정리를 위해 간략히 적을 테니 자세한 내용들에 대해서는 유인물 11페이지를 참조하도록 하자.
-> Index에 관련되어 소유자에 상관없이 인덱스를 새로이 만들거나 수정, 삭제하는 권한을 줄 수 있다.
Create Any Index, Alter Any Index, Drop Any Index
-> Table에 관련되어 소유자에 상관없이 인덱스를 새로이 만들거나 수정, 삭제하는 권한을 줄 수 있고 물론 DML 문장에 대해서도 제어가 가능하니 참조하도록 하자.
Create Any Table... Drop Any Table
Insert Any Table... Delete Any Table
-> 물론 Session에 대해서도 권한을 줄 수 있다. 이는 any가 붙지 않고 간략하게 다음과 같이 쓰면 된다. 우리가 지난번에 날짜 형식을 변경할 때 Alter Session에 대해 많이 살펴봤을 것이다. 이러한 권한들을 준다고 생각을 하면 된다.
Create Session(서버에 접속할 수 있는 권한)
Alter Session(접속 상태에서 환경 값을 변경할 수 있는 권한)
-> TableSpace에도 권한을 줄 수 있는데 이 개념에 대해서는 공부를 안 했는데 쉽게 말해 SQL계의 USB 메모리, 외장하드로 생각하면 쉽다. 이를 통해서 각 개체들에 대해서 본래 사용자에게 저장되는 것이 아닌 그 장소로 이동해 여러 객체들을 저장하는 원리로 생각하면 된다.
Create Tablespace, Alter Tablespace, Drop Tablespace....

2-2) SysOper, SysDBA 권한
이는 유인물 13페이지에 있는 문장을 참조하도록 한다.
SysOPER -> 우리가 컴퓨터를 켜고, 끄고, 백업하고, 포맷하고, 복구하고... 컴퓨터를 부팅하기 이전에 할 수 있는 역할들을 그 작업 이전에 시행을 할 수 있는 기능들을 컴퓨터 본체에서 제공하듯이, 데이터베이스를 실행할 때 서버에 접속하기 전에 실행할 수 있는 기능들에 대해 모든 권한들에 대한 묶음을 저장한 개념으로 생각하면 된다.
SysDBA - SysOper보다 센 권한들을 사용자들에게 제공을 할 수 있다. 방금 2-1에서 살펴본 권한들도 이를 통해서 다른 사용자들에게 권한을 주는 역할을 한다.

2-3) System 관련 권한 할당/해제
우리가 권한들에 대해서 간략하게 공부를 했다. 그래서 이러한 권한들을 사용자들에게 어떻게 권한을 주는지에 대한 문장들을 공부해 볼 필요가 있다. 유형들을 나누어서 공부를 해 보자.

(1) 사용자 계정을 새로 만들 경우
사용자 계정은 SYSDBA를 적용한 계정이 있는 sys 계정에서 새롭게 계정을 만들 수 있다. 아래와 같은 문장을 살펴보자.

Create User DBCOOL
identified by q1w2e3r4
본래 SYS 계정에 접속되어 있다면 마우스를 이용해서 추가해도 무방한데 이렇게 작성을 하게 되어도 기본 데이터베이스 테이블은 Users로, 임시 데이터베이스 테이블은 Temp로 지정되어 있다. 여기서 임시 데이터베이스 테이블을 왜 TEMP로 이용을 할까? 정렬작업, 집합작업, Distinct작업 등을 하는 문장들이 여기에 짬 때려져 있기 때문인데, 이를 이용해야 사용자가 이러한 옵션을 이용하는데 있어서 문제가 없게 된다.

(참고) 비밀번호는 알파벳으로부터 시작을 해야 된다. 가령 1q2w3e4r이런 식으로 숫자로 시작하면 누락된 옵션으로 오류가 뜬다. 주의하자.

(2) 사용자에게 권한 할당하기
지금 현재 SYSDBA 권한이 있는 SYS 계정이 방금 (1)번에 새로이 만든 DBCOOL라는 계정에 테이블, 시퀀스, 데이터베이스 서버 접속을 하는 권한을 주는 문장에 대해서 작성을 해보겠다. 아래를 참조하자.

Grant Create Table, Create Sequence, Create Session TO DBCOOL;

필자가 만일 문제에서 밑줄 친 부분을 없앤다고 생각을 하고 DBCOOL로 접속을 하면 어떻게 될까? 우리가 DBCOOL의 비밀번호를 이미 알고 있지만 애석하게도 서버 접속의 권한을 주지 않아 결국엔 로그인이 되지 않는다. 그래서 사용자를 생성해 접속을 새로이 추가하려면 Create Session을 꼭 넣어주길 바란다. 안 그러면 이 계정은 그림의 떡 밖에 안 된다.

(3) Role을 통해 권한 할당하기
본래 권한들을 적기에 번거로움을 줄이기 위해서 권한들을 묶어둔 개념들을 Role이라고 한다. Role은 역할이라는 뜻인데, 시스템 권한들을 묶어서 이러한 역할을 하겠다는 가상의 사용자(Virtual User)라고 생각하면 된다. 이번에는 Role에 데이터베이스 서버 접속 권한, 테이블 삽입, 삭제 권한을 주는 문장을 하도록 PlusMinusTable에게 할당을 해 보겠다.
Create Role PlusMinusTable;
// Role은 이렇게 간략하게만 작성해도 무방하다.
Grant Create Table, Drop Table, Create Session TO PlusMinusTable;
// 가상의 사용자 개념인 PlusMinusTable를 통해서 우선 가상의 사용자에게 권한들을 저장시켜 준다.
Grant PlusMinusTable TO DBCOOL;
// 최종적으로 PlusMinusTable에 대해 권한들을 할당시키면 DBCOOL은 그 권한들을 받게 된다.

(4) HR계정 내의 테이블(테이블을 이용해 DML 문장을 적용)을 관리할 수 있도록 권한을 주기
방금 DBCOOL 계정에게 HR 테이블에 있는 Employees 테이블을 select, insert, update를 할 수 있는 권한을 주는 문장을 작성해 보겠다. 이는 HR 계정에서 설정해줘야 적용이 된다.
Grant Select, Insert, Update ON HR.Employees
TO DBCOOL;
필자가 밑줄을 친 이유가 만일 SYS 계정이 아닌 타인의 계정에서 이용 중인 객체들에 대해서 이용을 할 수 있도록 권한을 주기 위해 ON 키워드를 작성해 줬다. ON은 INDEX에서 많이 봤을 것이다. 그렇지만 현재 계정의 테이블을 타인의 계정에 짬 때리기 위해서는 ON 키워드의 구조를 알아두고 가자.
On [계정이름].[계정의 객체] // 여기서 .은 그 계정 내에 있는 객체를 불러오기 위해 쓰는 연산자인데, 자바에서 인스턴스 변수 다음 .을 찍어주면 메소드를 불러올 수 있는 원리로 생각하면 쉽다.

(5) 사용자도 타 계정에게 권한을 부여할 수 있는 옵션 추가하기(With Grant Option/With Admin Option)

Grant Select, Delete ON HR.Departments
To DBCOOL with Grant Option;

Grant Select, Delete ON HR.Departments
To DBCOOL with Admin Option;

이는 쉽게 이야기해서 권한을 받은 계정인 DBCOOL이 다른 계정들에게 권한을 줄 수 있는 옵션이라고 생각하면 된다. 이를 작성해 주면 DBCOOL은 이러한 개념들이 필요로 한 계정들에게 계정을 줄 수 있다. 이 개념은 필자가 쉽게 설명하기 위해 드라마 일부 사진으로 준비했는데 파란 탑에서 나오는 김재우와 김호창, 정진욱을 예시로 들었으니 이를 참조해서 연상하면 되겠다.

병장 김재우 -> 계정 HR로 연상 | 상병 김호창 -> 계정 DBCOOL로 연상 | 일병 정진욱 -> DBCOOL에게 권한 받는 계정

그렇지만 With Grant Option과 With Admin Option의 차이는 무엇일까? With Grant Option을 통해 권한을 받은 DBCOOL 계정이 HR 계정에게 권한을 뺏으면 본래 DBCOOL에게 준 권한에 대해 타인들이 받은 권한도 빼앗기게 되지만, Admin Option을 주게 되면 DBCOOL의 권한만 뺏기지 DBCOOL에게 받은 타인의 권한들은 그대로 적용하게 되니깐 참조하도록 하자.

(6) 모든 사용자들에게 권한을 주기
권한을 쓰는 부분에 public을 쓰면 모든 사용자들에게 권한을 줄 수가 있다. 가령 예를 들어 모든 사용자들에게 HR 테이블의 Jobs 테이블 중 Select, Update를 할 수 있도록 하는 문장을 작성해 보겠다. 아래 문장을 참조하자.
Grant Select, Update ON HR.Jobs
To Public;
public은 방금 Synonym을 공부했을 때 참조했을 것이다. 가령 Create public Synonym를 하면 모든 사용자들이 이 테이블의 별명을 다 같이 부를 수 있다는 원리를 배웠듯이 public은 모든 사용자들을 뜻하는 키워드로 익혀두길 바란다. public의 원리는 파란 탑에 나오는 최종훈을 연상하면 쉽다.

(7) 권한에 대해 빼앗기
여태껏 Grant로서 권한만 계속 부여했었다. 허나 일부 사용자들에게 필요 없는 권한들을 다시 빼앗을 필요가 있다. 그래서 Revoke 키워드를 이용해서 권한을 빼앗는 문장에 대해 작성해 보겠다. DBCOOL 계정이 테이블 삭제, 테이블 수정을 하는 권한들을 뺏어보겠다.
Revoke Alter Table, Drop Table From DBCOOL;
우리가 Grant에서는 TO를 잘 써 왔지만, Revoke에서는 From이다. To를 쓴 이유는 누구에게 줄 것인가에 대해서를 썼고, From은 어느 계정에서 가져올까에 대해서 썼기 때문에 햇깔리는 일이 없도록 주의하자. 물론 HR계정의 Employees 테이블에서 테이블의 값을 수정하거나 삭제하지 못하도록 권한을 뺏는 예시로 문제를 풀면 다음과 같이 작성이 가능하다.
Revoke Update, Delete ON HR.Employees
from DBCOOL;
이렇게 되면 DBCOOL는 HR의 Employees 테이블에서 쓸 수 있는 개념들 중에서 Update, Delete를 못하게 된다. 만일 DBCOOL 계정 내에서 테이블을 참조하기 위해서는 아래와 같은 문장으로 작성해야 문제가 안 된다. HR.을 까먹으면 없는 테이블로 나오니 꼭 써두길 바란다.
DBCOOL> Select *
From HR.Employees;

(8) 사용자가 가진 권한/Role에 대해 알아보기
DBA_SYS_PRIVS를 통해서 현재 사용자가 받은 권한들에 대해서 출력을 한다. 구조는 다음과 같다.

Select grantee, privilege, admin_option
from DBA_SYS_PRIVS
where grantee='DBCOOL';

여기서 grantee는 우리가 흔히 쓰는 Employees가 고용 받는 직원들이라고 칭하듯이, 권한을 받는 사용자라는 뜻이다. 그래서 DBCOOL이 받는 권한들에 대해 출력을 하니 어느 권한들에 대해 받았는지 확인하려면 이를 작성해보자.
또한 Role에 대해서도 확인이 가능한데 구조는 위의 privilege가 아닌 granted_Role로서 현재 권한들을 할당받은 Role 이름을 출력하니 참조하면 된다.

<참조> 만일 밑줄 친 부분에 CONNECT, RESOURCES를 적으면 어떤 결과가 나올까? CONNECT를 쓰게 되면 가령 예를 들어 CONNECT로 적어서 검색을 하면 데이터베이스 서버를 연결하는 CREATE SESSION과 같은 원리의 권한이 나오게 된다. 이는 연결 관련된 권한들을 확인할 수 있고, RESOURCE는 CREATE TABLE와 같이 자원과 직속 연관된 권한들에 대해 출력을 하니 연결 관련 권한들과 자원 관련 권한들에 대해 나누어 출력이 가능하다는 것으로 숙지해두면 되겠다.

<좋은 기능> USER의 비밀 번호는 당연히 변경이 가능하다. 허나 방금 설명한 것처럼 알파벳을 우선시로 해서 작성을 해줘야 적용이 된다.
Alter User DBCOOL
identified by aqswdefr;

2016년 11월 9일에 배운 PL/SQL 관련된 문장은 새로운 요약정리에 정리하였으니 그 쪽부터 참조하시길 바랍니다.