PL/SQL Kapsamlı Çalışma Notu

Sınav Hazırlık Rehberi

1. PL/SQL'e Giriş

PL/SQL (Procedural Language/Structured Query Language), Oracle veritabanı için geliştirilmiş prosedürel bir programlama dilidir. SQL'e programlama özellikleri ekler ve veritabanı işlemlerini daha güçlü hale getirir.

PL/SQL'in Avantajları

- Blok yapısı sayesinde karmaşık işlemler tek bir birim olarak çalıştırılabilir
- Hata yakalama ve işleme mekanizmaları
- Değişkenler, döngüler, koşullar gibi programlama yapıları
- Veritabanı üzerinde performanslı çalışma

2. Blok Yapıları ve Mimarisi

PL/SQL kodları bloklar halinde yazılır. Bir blok üç temel bölümden oluşur:

1. **DECLARE (İsteğe Bağlı):** Değişkenlerin, sabitlerin ve cursor'ların tanımlandığı bölüm
2. **BEGIN (Zorunlu):** SQL ve PL/SQL komutlarının yazıldığı ana bölüm
3. **EXCEPTION (İsteğe Bağlı):** Hataların yakalandığı ve işlendiği bölüm

Basit Blok Örneği

```
DECLARE
    sayi NUMBER := 10;
BEGIN
    dbms_output.put_line('Sayı değeri: ' || sayi);
END;
```

İç İçe (Nested) Bloklar

PL/SQL'de bloklar iç içe yazılabilir. Her blok kendi kapsamına (scope) sahiptir.

```
<<dis_block>>
DECLARE
    sayi NUMBER := 10;
BEGIN
    <<ic_block>>
    DECLARE
        sayi NUMBER := 5;
    BEGIN
```

```
            dbms_output.put_line('İç blok: ' || ic_block.sayi);
        END;
        dbms_output.put_line('Dış blok: ' || dis_block.sayi);
    END;
```

Çıktı: İç blok: 5

Dış blok: 10

3. Değişken Çözünürlüğü (Variable Scope)

Değişken çözünürlüğü, bir değişkenin hangi blok içinde görünür ve erişilebilir olduğunu belirler.

Temel Kurallar

- Bir değişken tanımlandığı blok içinde ve alt bloklarda geçerlidir
- Alt blokta aynı isimli değişken tanımlanırsa, alt bloktaki değişken üst bloktakini gölgeler
- Blok etiketleri (<<blok_adi>>) kullanılarak gölgelenmiş değişkenlere erişilebilir
- PL/SQL büyük-küçük harf duyarlı DEĞİLDİR (case-insensitive)

Büyük-Küçük Harf Duyarsızlığı Örneği

```
DECLARE
    sayi NUMBER;
    SAYI NUMBER;  -- Aynı değişken!
BEGIN
    sayi := 15;
    SAYI := -2;  -- Önceki değerin üzerine yazar
    dbms_output.put_line('Toplam: ' || (sayi + SAYI));
END;
```

Çıktı: Toplam: -4 (çünkü her iki değişken de aynıdır ve son değer -2'dir)

4. Veri Tipleri

PL/SQL'de değişkenler tanımlanırken veri tipi belirtilmelidir. İşte temel veri tipleri:

Veri Tipi	Açıklama ve Kullanım

NUMBER	Sayısal değerler için kullanılır
	Örnekler:
	sayi NUMBER := 100;
	fiyat NUMBER(5,2) := 125.50;

VARCHAR2	Değişken uzunlukta karakter dizisi (maximum 32767 byte) *Örnekler:*

Veri Tipi	Açıklama ve Kullanım
	isim VARCHAR2(50) := 'Ahmet Yılmaz'; ders VARCHAR2(15) := 'PL/SQL';
DATE	Tarih ve saat bilgisi saklar *Örnekler:* dogum_tarihi DATE := '24-Apr-2002'; bugun DATE := SYSDATE;
BOOLEAN	Mantıksal değerler: TRUE, FALSE veya NULL *Örnekler:* aktif BOOLEAN := TRUE; varMi BOOLEAN := TRUE AND NULL; -- NULL döner

BOOLEAN Veri Tipi ile Çalışma

ÖNEMLİ: BOOLEAN değerler dbms_output ile doğrudan yazdırılamaz. IF-ELSE yapısı ile kontrol edilmelidir.

```
DECLARE
    varMi BOOLEAN := TRUE AND NULL;
BEGIN
    IF (varMi = TRUE) THEN
        dbms_output.put_line('TRUE');
    ELSIF (varMi = FALSE) THEN
        dbms_output.put_line('FALSE');
    ELSE
        dbms_output.put_line('NULL');
    END IF;
END;
```

Çıktı: NULL (çünkü TRUE AND NULL = NULL)

5. Operatörler

PL/SQL'de çeşitli operatörler kullanılarak işlemler yapılabilir.

Aritmetik Operatörler

Operatör	Açıklama	Örnek
+	Toplama	5 + 3 = 8
-	Çıkarma	10 - 4 = 6
*	Çarpma	7 * 2 = 14

Operatör	Açıklama	Örnek
/	Bölme	20 / 4 = 5
||	String birleştirme	'Merhaba' || ' Dünya'

Karşılaştırma Operatörleri

Operatör	Açıklama	Örnek
=	Eşit mi?	x = 5
!=, <>	Eşit değil mi?	x <> 5
>	Büyük mü?	x > 10
<	Küçük mü?	x < 20
>=, <=	Büyük/küçük eşit	x >= 5

Mantıksal Operatörler

- **AND:** Her iki koşul da doğru olmalı
- **OR:** Koşullardan en az biri doğru olmalı
- **NOT:** Mantıksal değili alır

6. IF-ELSE Yapıları

Koşullu işlemler için IF-ELSE yapıları kullanılır.

Basit IF Yapısı

```
IF koşul THEN
    -- İşlemler
END IF;
```

IF-ELSE Yapısı

```
IF koşul THEN
    -- Koşul doğruysa
ELSE
    -- Koşul yanlışsa
END IF;
```

IF-ELSIF-ELSE Yapısı

```
IF koşul1 THEN
    -- Koşul1 doğruysa
ELSIF koşul2 THEN
    -- Koşul2 doğruysa
```

```
ELSE
    -- Hiçbir koşul doğru değilse
END IF;
```

Detaylı Örnek

```
DECLARE
    puan NUMBER := 75;
    harf VARCHAR2(2);
BEGIN
    IF puan >= 90 THEN
        harf := 'AA';
    ELSIF puan >= 80 THEN
        harf := 'BA';
    ELSIF puan >= 70 THEN
        harf := 'BB';
    ELSIF puan >= 60 THEN
        harf := 'CB';
    ELSE
        harf := 'FF';
    END IF;
    dbms_output.put_line('Harf notu: ' || harf);
END;
```

7. SELECT INTO - Veritabanından Veri Okuma

SELECT INTO komutu, veritabanından veri çekip değişkenlere atamak için kullanılır.

Temel Söz Dizimi

```
SELECT kolon_adi
INTO değişken_adi
FROM tablo_adi
WHERE koşul;
```

Örnek

```
DECLARE
    v_first_name VARCHAR2(20);
BEGIN
    SELECT first_name
    INTO v_first_name
    FROM employees
    WHERE last_name = 'Vargas';
```

```
    DBMS_OUTPUT.PUT_LINE('İsim: ' || v_first_name);
END;
```

ÖNEMLİ: SELECT INTO sorgusu tam olarak BİR satır döndürmelidir. Hiç satır dönmezse NO_DATA_FOUND hatası, birden fazla satır dönerse TOO_MANY_ROWS hatası alınır.

8. DML İşlemleri (INSERT, UPDATE, DELETE)

DML (Data Manipulation Language) komutları veritabanındaki verileri eklemek, güncellemek ve silmek için kullanılır.

INSERT INTO - Veri Ekleme

Tabloya yeni kayıt eklemek için kullanılır.

```
-- Tüm kolonlara değer ekleme
INSERT INTO jobs (job_id, job_title, min_salary, max_salary)
VALUES ('ST_MIS', 'MIS STUDENT', 1000, 5000);

-- Sadece bazı kolonlara değer ekleme
INSERT INTO jobs (job_id, job_title)
VALUES ('TC_MIS', 'MIS TEACHER');

-- SELECT sonucunu INSERT etme
INSERT INTO bonuses(employee_id)
(SELECT employee_id FROM employees WHERE salary < 10000);
```

UPDATE - Veri Güncelleme

Mevcut kayıtları güncellemek için kullanılır.

```
UPDATE jobs
SET job_title = 'ZEKI HOCA'
WHERE job_title = 'MIS TEACHER';
```

DİKKAT: WHERE koşulu belirtilmezse tablodaki TÜM kayıtlar güncellenir!

DELETE - Veri Silme

Kayıtları silmek için kullanılır.

```
DELETE FROM jobs
WHERE job_id = 'ST_MIS';

-- Tüm kayıtları silme
DELETE FROM jobs2;
```

DİKKAT: WHERE koşulu belirtilmezse tablodaki TÜM kayıtlar silinir!

9. COMMIT ve ROLLBACK - Transaction Yönetimi

Transaction, bir veya birden fazla DML işleminin mantıksal bir bütün olarak ele alınmasıdır.

COMMIT

COMMIT komutu, yapılan değişiklikleri kalıcı hale getirir. COMMIT çalıştırıldıktan sonra değişiklikler geri alınamaz.

```
INSERT INTO jobs (job_id, job_title, min_salary, max_salary)
VALUES ('ST_MIS', 'MIS STUDENT', 1000, 5000);

COMMIT;  -- Değişiklik kalıcı oldu
```

ROLLBACK

ROLLBACK komutu, son COMMIT'ten sonra yapılan tüm değişiklikleri geri alır.

```
INSERT INTO jobs (job_id, job_title, min_salary, max_salary)
VALUES ('TC_MIS', 'MIS TEACHER', 5000, 15000);

ROLLBACK;  -- Değişiklik geri alındı, veri eklenmedi
```

Kullanım Senaryosu

Birden fazla işlem yaparken, hata durumunda tüm işlemleri geri almak için ROLLBACK kullanılır.

```
BEGIN
    INSERT INTO hesap VALUES (1, 1000);
    UPDATE hesap SET bakiye = bakiye - 500 WHERE id = 1;
    UPDATE hesap SET bakiye = bakiye + 500 WHERE id = 2;

    -- Tüm işlemler başarılı ise
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        -- Hata durumunda tüm işlemleri geri al
        ROLLBACK;
END;
```

10. MERGE İşlemi

MERGE komutu, bir tabloya veri eklerken veya güncellerken kullanılır. Eğer kayıt varsa günceller, yoksa ekler (UPSERT işlemi).

Temel Söz Dizimi

```
MERGE INTO hedef_tablo h
```

```
USING kaynak_tablo k
ON (eşleşme_koşulu)
WHEN MATCHED THEN
    UPDATE SET kolonlar
WHEN NOT MATCHED THEN
    INSERT VALUES (değerler);
```

Örnek

```
MERGE INTO bonuses b
USING employees e
ON (b.employee_id = e.employee_id)
WHEN MATCHED THEN
    UPDATE SET b.bonus = e.salary * 0.05;
```

Açıklama: Bu örnekte, employees tablosundaki her çalışan için bonuses tablosunda kayıt varsa, bonusu maaşın %5'i olarak günceller.

11. DDL İşlemleri (CREATE TABLE, DROP TABLE)

DDL (Data Definition Language) komutları, veritabanı nesnelerini oluşturmak, değiştirmek ve silmek için kullanılır.

CREATE TABLE - Tablo Oluşturma

```
CREATE TABLE bonuses
(
    employee_id NUMBER(6,0) NOT NULL,
    bonus NUMBER(8,2) DEFAULT 0
);
```

Açıklama:

- **NOT NULL:** Kolon boş bırakılamaz
- **DEFAULT:** Değer girilmezse varsayılan değer kullanılır

DROP TABLE - Tablo Silme

Tabloyu veritabanından tamamen siler.

```
DROP TABLE bonuses;
```

UYARI: DROP TABLE komutu geri alınamaz! Tablo ve içindeki tüm veriler kalıcı olarak silinir.

12. Kullanıcıdan Veri Alma (&)

& operatörü ile kullanıcıdan çalışma zamanında veri alınabilir.

```
DECLARE
```

```
    sayi1 NUMBER;
    sayi2 NUMBER;
BEGIN
    sayi1 := &sayi1;  -- Kullanıcıdan ilk sayıyı al
    sayi2 := &sayi2;  -- Kullanıcıdan ikinci sayıyı al
    dbms_output.put_line('Toplam: ' || (sayi1 + sayi2));
END;
```

13. Sınav İpuçları ve Önemli Noktalar

Dikkat Edilmesi Gereken Noktalar

4. **Blok Yapısı:** BEGIN zorunludur, DECLARE ve EXCEPTION isteğe bağlıdır
5. **Noktalı virgül:** Her komut noktalı virgül (;) ile biter
6. **Büyük-küçük harf:** PL/SQL case-insensitive'dir (değişken isimleri için)
7. **String birleştirme:** || operatörü kullanılır
8. **Atama operatörü:** := kullanılır (= karşılaştırma için)
9. **BOOLEAN:** Doğrudan yazdırılamaz, IF ile kontrol edilmelidir
10. **SELECT INTO:** Tam olarak bir satır döndürmelidir
11. **WHERE koşulu:** UPDATE ve DELETE'te WHERE kullanmazsanız tüm kayıtlar etkilenir
12. **COMMIT/ROLLBACK:** Transaction yönetimi için kritiktir
13. **dbms_output:** Çıktı almak için dbms_output.put_line kullanılır

14. Örnek Sınav Soruları

Soru 1: Blok Yapısı

Aşağıdaki kodun çıktısı nedir?

```
DECLARE
    x NUMBER := 5;
BEGIN
    DECLARE
        x NUMBER := 10;
    BEGIN
        dbms_output.put_line(x);
    END;
    dbms_output.put_line(x);
END;
```

Cevap: 10

5

Açıklama: İç blokta x değişkeni 10 değerine sahiptir. Dış blokta ise 5 değerine sahiptir.

Soru 2: IF-ELSE

Bir öğrencinin notuna göre geçti/kaldı durumunu yazdıran kodu tamamlayın (60 ve üzeri geçer).

```
DECLARE
    not NUMBER := 55;
BEGIN
    IF not >= 60 THEN
        dbms_output.put_line('Geçti');
    ELSE
        dbms_output.put_line('Kaldı');
    END IF;
END;
```

Soru 3: COMMIT ve ROLLBACK

Aşağıdaki koddan sonra tabloda kaç kayıt olur?

```
INSERT INTO test VALUES (1, 'A');
COMMIT;
INSERT INTO test VALUES (2, 'B');
INSERT INTO test VALUES (3, 'C');
ROLLBACK;
```

Cevap: 1 kayıt (sadece ilk INSERT kalıcı oldu)

15. Özet

Bu çalışma notunda PL/SQL'in temel konularını öğrendiniz:

- **Blok yapıları** - DECLARE, BEGIN, EXCEPTION
- **Veri tipleri** - NUMBER, VARCHAR2, DATE, BOOLEAN
- **Değişken çözünürlüğü** - Scope ve iç içe bloklar
- **Operatörler** - Aritmetik, karşılaştırma, mantıksal
- **IF-ELSE yapıları** - Koşullu işlemler
- **SELECT INTO** - Veritabanından veri okuma
- **DML işlemleri** - INSERT, UPDATE, DELETE
- **Transaction yönetimi** - COMMIT, ROLLBACK
- **MERGE işlemi** - UPSERT
- **DDL işlemleri** - CREATE TABLE, DROP TABLE

Başarılar!

Bu notları düzenli olarak gözden geçirin ve örnekleri kendiniz yazmaya çalışın.