Veri Tabanı Proje

Konu

***Maaş Otomasyonu***

Hazırlayanlar

183311047-Yasin HELVACI

Create Komutları

-- Departmanlar tablosu

CREATE TABLE Departmanlar (

departmanID NUMBER PRIMARY KEY,

departmanAdi VARCHAR2(50)

);

-- Personel tablosu

CREATE TABLE Personel (

personelID NUMBER PRIMARY KEY,

adi VARCHAR2(50),

soyadi VARCHAR2(50),

dogumTarihi DATE,

cinsiyet VARCHAR2(10),

topalmYillikIzinHakki NUMBER,

mevcutMaas NUMBER,

departmanID NUMBER REFERENCES Departmanlar(departmanID)

);

-- PersonelBankaHesaplari tablosu

CREATE TABLE PersonelBankaHesaplari (

hesapID NUMBER PRIMARY KEY,

personelID NUMBER REFERENCES Personel(personelID),

bankaAdi VARCHAR2(50),

hesapNumarasi VARCHAR2(20),

subeAdi VARCHAR2(50)

);

-- MaasZamlari tablosu

CREATE TABLE MaasZamlari (

zammID NUMBER PRIMARY KEY,

personelID NUMBER REFERENCES Personel(personelID),

miktar NUMBER,

tarih DATE

);

-- MaasOdeme tablosu

CREATE TABLE MaasOdeme (

maasID NUMBER PRIMARY KEY,

personelID NUMBER REFERENCES Personel(personelID),

miktar NUMBER,

tarih DATE

);

-- Avans tablosu

CREATE TABLE Avans (

avansID NUMBER PRIMARY KEY,

personelID NUMBER REFERENCES Personel(personelID),

miktar NUMBER,

tarih DATE

);

-- Izinler tablosu

CREATE TABLE Izinler (

izinID NUMBER PRIMARY KEY,

personelID NUMBER REFERENCES Personel(personelID),

ucretliMi CHAR(1),

baslangicTarihi DATE,

bitisTarihi DATE,

izinTuru VARCHAR2(50)

);

CREATE TABLE CalismaSaatleri (

calismaID NUMBER PRIMARY KEY,

personelID NUMBER REFERENCES Personel(personelID),

calismaTarihi DATE,

calismaSaati NUMBER

);

-- Sigorta tablosu

CREATE TABLE Sigorta (

sigortaID NUMBER PRIMARY KEY,

personelID NUMBER REFERENCES Personel(personelID),

sigortaTuru VARCHAR2(50),

sigortaBaslangicTarihi DATE,

sigortaBitisTarihi DATE,

primTutari NUMBER,

poliçeNumarasi VARCHAR2(20)

);

-- YillikIzinler tablosu

CREATE TABLE YillikIzinler (

izinID NUMBER PRIMARY KEY,

personelID NUMBER REFERENCES Personel(personelID),

baslangicTarihi DATE,

bitisTarihi DATE,

onayDurumu VARCHAR2(20)

);

Burada tablolarımızı oluşturduk. Toplamda 10 tane tablomuz var . Değişkenlerimizi tanımladık.

Bir Bir İlişki (One-to-One): Bir bir ilişkide, bir kayıt bir başka kayıt ile yalnızca bir kez ilişkilidir. Örneğin, "Personel" tablosundaki bir personel kaydının sadece bir "PersonelBankaHesaplari" kaydına sahip olabileceği bir bir ilişki olabilir.

Bir Çok İlişki (One-to-Many): Bu ilişkide bir kayıt, diğer bir kayıt ile birden çok ilişkilidir. Örneğin, "Departmanlar" tablosundaki bir departmanın birden çok "Personel" kaydına sahip olabileceği bir bir çok ilişki olabilir.

Çok Çok İlişki (Many-to-Many): Bir çok ilişkide, bir kayıt, diğer bir kayıt ile birden çok ilişkilidir ve aynı durum diğer yönde de geçerlidir. Ancak, bu tür bir ilişkiyi desteklemek için genellikle bir bağlantı tablosuna ihtiyaç duyulur. Örneğin, bir "Personel" birden çok "Izinler" kaydına sahip olabilir ve bir "Izinler" kaydı birden çok "Personel" ile ilişkilendirilebilir.

metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Diyagram

Insert Into Komutları

-- Departmanlar tablosu

INSERT INTO Departmanlar (departmanID, departmanAdi)

VALUES (1, 'Muhasebe');

INSERT INTO Departmanlar (departmanID, departmanAdi)

VALUES (2, 'Satış');

-- Personel tablosu

INSERT INTO Personel (personelID, adi, soyadi, dogumTarihi, cinsiyet, departmanID, topalmYillikIzinHakki, mevcutMaas)

VALUES (1, 'Ahmet', 'Yılmaz', TO\_DATE('1990-05-15', 'YYYY-MM-DD'), 'Erkek', 1, 20, 40000);

INSERT INTO Personel (personelID, adi, soyadi, dogumTarihi, cinsiyet, departmanID, topalmYillikIzinHakki, mevcutMaas)

VALUES (2, 'Ayşe', 'Demir', TO\_DATE('1985-08-22', 'YYYY-MM-DD'), 'Kadın', 2, 18, 50000);

-- PersonelBankaHesaplari tablosu

INSERT INTO PersonelBankaHesaplari (hesapID, personelID, bankaAdi, hesapNumarasi, subeAdi)

VALUES (1, 1, 'Ziraat Bankası', '123456789', 'Ankara Şubesi');

INSERT INTO PersonelBankaHesaplari (hesapID, personelID, bankaAdi, hesapNumarasi, subeAdi)

VALUES (2, 2, 'Garanti Bankası', '987654321', 'İstanbul Şubesi');

-- MaasZamlari tablosu

INSERT INTO MaasZamlari (zammID, personelID, miktar, tarih)

VALUES (1, 1, 5000, TO\_DATE('2024-01-01', 'YYYY-MM-DD'));

INSERT INTO MaasZamlari (zammID, personelID, miktar, tarih)

VALUES (2, 2, 6000, TO\_DATE('2024-01-05', 'YYYY-MM-DD'));

-- MaasOdeme tablosu

INSERT INTO MaasOdeme (maasID, personelID, miktar, tarih)

VALUES (1, 1, 40000, TO\_DATE('2024-01-15', 'YYYY-MM-DD'));

INSERT INTO MaasOdeme (maasID, personelID, miktar, tarih)

VALUES (2, 2, 50000, TO\_DATE('2024-01-20', 'YYYY-MM-DD'));

-- Avans tablosu

INSERT INTO Avans (avansID, personelID, miktar, tarih)

VALUES (1, 1, 1500, TO\_DATE('2024-01-10', 'YYYY-MM-DD'));

INSERT INTO Avans (avansID, personelID, miktar, tarih)

VALUES (2, 2, 2000, TO\_DATE('2024-01-12', 'YYYY-MM-DD'));

-- Izinler tablosu

INSERT INTO Izinler (izinID, personelID, ucretliMi, baslangicTarihi, bitisTarihi)

VALUES (1, 1, 'Y', TO\_DATE('2024-02-07', 'YYYY-MM-DD'), TO\_DATE('2024-02-15', 'YYYY-MM-DD'));

INSERT INTO Izinler (izinID, personelID, ucretliMi, baslangicTarihi, bitisTarihi)

VALUES (2, 2, 'N', TO\_DATE('2024-02-10', 'YYYY-MM-DD'), TO\_DATE('2024-02-17', 'YYYY-MM-DD'));

-- CalismaSaatleri tablosu

INSERT INTO CalismaSaatleri (calismaID, personelID, calismaTarihi, calismaSaati)

VALUES (1, 1, TO\_DATE('2024-02-01', 'YYYY-MM-DD'), 8);

INSERT INTO CalismaSaatleri (calismaID, personelID, calismaTarihi, calismaSaati)

VALUES (2, 2, TO\_DATE('2024-02-01', 'YYYY-MM-DD'), 7);

-- Sigorta tablosu

INSERT INTO Sigorta (sigortaID, personelID, sigortaTuru, sigortaBaslangicTarihi, sigortaBitisTarihi, primTutari, poliçeNumarasi)

VALUES (1, 1, 'Sağlık Sigortası', TO\_DATE('2024-01-01', 'YYYY-MM-DD'), TO\_DATE('2025-01-01', 'YYYY-MM-DD'), 2000, 'ABC123');

INSERT INTO Sigorta (sigortaID, personelID, sigortaTuru, sigortaBaslangicTarihi, sigortaBitisTarihi, primTutari, poliçeNumarasi)

VALUES (2, 2, 'İşsizlik Sigortası', TO\_DATE('2024-02-01', 'YYYY-MM-DD'), TO\_DATE('2025-02-01', 'YYYY-MM-DD'), 1500, 'XYZ789');

-- YillikIzinler tablosu

INSERT INTO YillikIzinler (izinID, personelID, baslangicTarihi, bitisTarihi, onayDurumu)

VALUES (1, 1, TO\_DATE('2024-06-01', 'YYYY-MM-DD'), TO\_DATE('2024-06-10', 'YYYY-MM-DD'), 'Onaylandı');

INSERT INTO YillikIzinler (izinID, personelID, baslangicTarihi, bitisTarihi, onayDurumu)

VALUES (2, 2, TO\_DATE('2024-07-01', 'YYYY-MM-DD'), TO\_DATE('2024-07-07', 'YYYY-MM-DD'), 'Onaylandı');

-- Personel tablosu

INSERT INTO Personel (personelID, adi, soyadi, dogumTarihi, cinsiyet, departmanID, topalmYillikIzinHakki, mevcutMaas)

VALUES (3, 'Mehmet', 'Demir', TO\_DATE('1988-11-18', 'YYYY-MM-DD'), 'Erkek', 1, 15, 6000);

INSERT INTO Personel (personelID, adi, soyadi, dogumTarihi, cinsiyet, departmanID, topalmYillikIzinHakki, mevcutMaas)

VALUES (4, 'Zeynep', 'Kaya', TO\_DATE('1995-07-25', 'YYYY-MM-DD'), 'Kadın', 2, 17, 9000);

-- PersonelBankaHesaplari tablosu

INSERT INTO PersonelBankaHesaplari (hesapID, personelID, bankaAdi, hesapNumarasi, subeAdi)

VALUES (3, 3, 'Akbank', '567890123', 'İzmir Şubesi');

INSERT INTO PersonelBankaHesaplari (hesapID, personelID, bankaAdi, hesapNumarasi, subeAdi)

VALUES (4, 4, 'Yapı Kredi', '987654321', 'Ankara Şubesi');

-- MaasZamlari tablosu

INSERT INTO MaasZamlari (zammID, personelID, miktar, tarih)

VALUES (3, 3, 700, TO\_DATE('2024-01-20', 'YYYY-MM-DD'));

INSERT INTO MaasZamlari (zammID, personelID, miktar, tarih)

VALUES (4, 4, 800, TO\_DATE('2024-01-25', 'YYYY-MM-DD'));

-- MaasOdeme tablosu

INSERT INTO MaasOdeme (maasID, personelID, miktar, tarih)

VALUES (3, 3, 4500, TO\_DATE('2024-01-28', 'YYYY-MM-DD'));

INSERT INTO MaasOdeme (maasID, personelID, miktar, tarih)

VALUES (4, 4, 5000, TO\_DATE('2024-01-30', 'YYYY-MM-DD'));

-- Avans tablosu

INSERT INTO Avans (avansID, personelID, miktar, tarih)

VALUES (3, 3, 1200, TO\_DATE('2024-01-15', 'YYYY-MM-DD'));

INSERT INTO Avans (avansID, personelID, miktar, tarih)

VALUES (4, 4, 900, TO\_DATE('2024-01-18', 'YYYY-MM-DD'));

-- Izinler tablosu

INSERT INTO Izinler (izinID, personelID, ucretliMi, baslangicTarihi, bitisTarihi)

VALUES (3, 3, 'Y', TO\_DATE('2024-03-01', 'YYYY-MM-DD'), TO\_DATE('2024-03-05', 'YYYY-MM-DD'));

INSERT INTO Izinler (izinID, personelID, ucretliMi, baslangicTarihi, bitisTarihi)

VALUES (4, 4, 'N', TO\_DATE('2024-03-10', 'YYYY-MM-DD'), TO\_DATE('2024-03-15', 'YYYY-MM-DD'));

-- CalismaSaatleri tablosu

INSERT INTO CalismaSaatleri (calismaID, personelID, calismaTarihi, calismaSaati)

VALUES (3, 3, TO\_DATE('2024-03-01', 'YYYY-MM-DD'), 6);

INSERT INTO CalismaSaatleri (calismaID, personelID, calismaTarihi, calismaSaati)

VALUES (4, 4, TO\_DATE('2024-03-01', 'YYYY-MM-DD'), 7);

-- Sigorta tablosu

INSERT INTO Sigorta (sigortaID, personelID, sigortaTuru, sigortaBaslangicTarihi, sigortaBitisTarihi, primTutari, poliçeNumarasi)

VALUES (3, 3, 'Kaza Sigortası', TO\_DATE('2024-02-01', 'YYYY-MM-DD'), TO\_DATE('2025-02-01', 'YYYY-MM-DD'), 250, 'KZ123');

INSERT INTO Sigorta (sigortaID, personelID, sigortaTuru, sigortaBaslangicTarihi, sigortaBitisTarihi, primTutari, poliçeNumarasi)

VALUES (4, 4, 'Hayat Sigortası', TO\_DATE('2024-02-15', 'YYYY-MM-DD'), TO\_DATE('2025-02-15', 'YYYY-MM-DD'), 300, 'HY456');

-- YillikIzinler tablosu

INSERT INTO YillikIzinler (izinID, personelID, baslangicTarihi, bitisTarihi, onayDurumu)

VALUES (3, 3, TO\_DATE('2024-08-01', 'YYYY-MM-DD'), TO\_DATE('2024-08-10', 'YYYY-MM-DD'), 'Onaylandı');

INSERT INTO YillikIzinler (izinID, personelID, baslangicTarihi, bitisTarihi, onayDurumu)

VALUES (4, 4, TO\_DATE('2024-09-01', 'YYYY-MM-DD'), TO\_DATE('2024-09-07', 'YYYY-MM-DD'), 'Onaylandı');

Burada tablolarımıza veri girdik . Toplam 4 tane personelimiz var .

Tabloları bilgiler ile doldurduk .

Select Sorguları

SELECT

P.personelID,

P.adi,

P.soyadi,

P.mevcutMaas,

M.miktar AS maasMiktari,

M.tarih AS maasTarihi,

I.baslangicTarihi AS izinBaslangic,

I.bitisTarihi AS izinBitis

FROM

Personel P

JOIN

MaasOdeme M ON P.personelID = M.personelID

LEFT JOIN

Izinler I ON P.personelID = I.personelID

ORDER BY

P.personelID;

SELECT

D.departmanID,

D.departmanAdi,

P.personelID,

P.adi,

P.soyadi,

M.miktar AS maasMiktari,

M.tarih AS maasTarihi

FROM

Departmanlar D

JOIN

Personel P ON D.departmanID = P.departmanID

JOIN

MaasOdeme M ON P.personelID = M.personelID

ORDER BY

D.departmanID, P.personelID, M.tarih;

SELECT

D.departmanID,

D.departmanAdi,

P.personelID,

P.adi,

P.soyadi,

I.baslangicTarihi AS izinBaslangic,

I.bitisTarihi AS izinBitis,

I.ucretliMi

FROM

Departmanlar D

JOIN

Personel P ON D.departmanID = P.departmanID

JOIN

Izinler I ON P.personelID = I.personelID

ORDER BY

D.departmanID, P.personelID, I.baslangicTarihi;

SELECT

D.departmanID,

D.departmanAdi,

P.personelID,

P.adi,

P.soyadi,

MZ.zammID,

MZ.miktar AS zamMiktari,

MZ.tarih AS zamTarihi

FROM

Departmanlar D

JOIN

Personel P ON D.departmanID = P.departmanID

JOIN

MaasZamlari MZ ON P.personelID = MZ.personelID

ORDER BY

D.departmanID, P.personelID, MZ.tarih;

Bu sorgu, departmanlar, personel ve maaş zamları tablolarını birleştirerek, her personelin çalıştığı departmanı, kişisel bilgilerini ve almış olduğu maaş zamlarını listeliyor .

Burada JOIN işlemleri kullanılmış 3 tablo birleştirilmiştir.

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, sayı, numara, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Procudere

CREATE OR REPLACE PROCEDURE ListDepartmanPersonelMaaş (

p\_departmanID IN NUMBER

) AS

CURSOR c\_personeller IS

SELECT

P.personelID,

P.adi,

P.soyadi,

M.miktar AS maas

FROM

Personel P

JOIN

MaasOdeme M ON P.personelID = M.personelID

WHERE

P.departmanID = p\_departmanID;

BEGIN

FOR personel\_rec IN c\_personeller LOOP

DBMS\_OUTPUT.PUT\_LINE('Personel ID: ' || personel\_rec.personelID || ', Adı: ' || personel\_rec.adi || ', Soyadı: ' || personel\_rec.soyadi || ', Maaş: ' || personel\_rec.maas);

END LOOP;

END;

/

BEGIN

ListDepartmanPersonelMaaş(1);

END;

/

metin, makbuz, yazı tipi, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu prosedür, belirli bir departmandaki personellerin maaş bilgilerini listeleyen bir prosedürdür.

Giriş parametresi (p\_departmanID): Departman ID'si. Bu ID'ye sahip departmandaki personellerin maaşları listelenecektir.

CURSOR c\_personeller: Departman ID'sine göre Personel ve MaasOdeme tablolarını birleştiren bir cursor. Personel ve MaasOdeme tablolarından gelen verilerin depolanması için kullanılır.

FOR personel\_rec IN c\_personeller LOOP: Cursor üzerinde döngü başlatılır ve her bir personel için maaş bilgileri ekrana yazdırılır.

CREATE OR REPLACE PROCEDURE ListPersonelMaasZamlari (

p\_personelID IN NUMBER

) AS

CURSOR c\_zamlar IS

SELECT

zammID,

miktar,

tarih

FROM

MaasZamlari

WHERE

personelID = p\_personelID;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Personel ID: ' || p\_personelID);

FOR zam\_rec IN c\_zamlar LOOP

DBMS\_OUTPUT.PUT\_LINE('Zam ID: ' || zam\_rec.zammID || ', Miktar: ' || zam\_rec.miktar || ', Tarih: ' || zam\_rec.tarih);

END LOOP;

END;

/

BEGIN

ListPersonelMaasZamlari(1);

END;

/

Çıktı:

metin, makbuz, yazı tipi, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu prosedür, belirli bir personelin aldığı maaş zamlarını listeleyen bir prosedürdür.

Giriş parametresi (p\_personelID): Personel ID'si. Bu ID'ye sahip personelin aldığı maaş zamları listelenecektir.

CURSOR c\_zamlar: Personel ID'sine göre MaasZamlari tablosundan veri çeken bir cursor. MaasZamlari tablosundan gelen verilerin depolanması için kullanılır. Maaş zamları ekrana yazdırılır.

CREATE OR REPLACE PROCEDURE ListPersonelMaasOdemeleri (

p\_personelID IN NUMBER

) AS

CURSOR c\_odemeler IS

SELECT

maasID,

miktar,

tarih

FROM

MaasOdeme

WHERE

personelID = p\_personelID;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Personel ID: ' || p\_personelID);

FOR odeme\_rec IN c\_odemeler LOOP

DBMS\_OUTPUT.PUT\_LINE('Ödeme ID: ' || odeme\_rec.maasID || ', Miktar: ' || odeme\_rec.miktar || ', Tarih: ' || odeme\_rec.tarih);

END LOOP;

END;

/

BEGIN

ListPersonelMaasOdemeleri(1);

END;

/

Çıktı:

Procedure created. Statement processed.  
Personel ID: 1  
Ödeme ID: 1, Miktar: 40000, Tarih: 15-JAN-24

Bu prosedür, belirli bir personelin aldığı maaş ödemelerini listeleyen bir prosedürdür.

Giriş parametresi (p\_personelID): Personel ID'si. Bu ID'ye sahip personelin aldığı maaş ödemeleri listelenecektir.

CURSOR c\_odemeler: Personel ID'sine göre MaasOdeme tablosundan veri çeken bir cursor. MaasOdeme tablosundan gelen verilerin depolanması için kullanılır. Maaş ödemeleri ekrana yazdırılır.

CREATE OR REPLACE PROCEDURE ListPersonelAvanslari (

p\_personelID IN NUMBER

) AS

CURSOR c\_avanslar IS

SELECT

avansID,

miktar,

tarih

FROM

Avans

WHERE

personelID = p\_personelID;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Personel ID: ' || p\_personelID);

FOR avans\_rec IN c\_avanslar LOOP

DBMS\_OUTPUT.PUT\_LINE('Avans ID: ' || avans\_rec.avansID || ', Miktar: ' || avans\_rec.miktar || ', Tarih: ' || avans\_rec.tarih);

END LOOP;

END;

/

BEGIN

ListPersonelAvanslari(1);

END;

/

Çıktı:

**Procedure created. Statement processed.  
Personel ID: 1  
Avans ID: 1, Miktar: 1500, Tarih: 10-JAN-24**

Fonksiyonlar

CREATE OR REPLACE FUNCTION GetPersonelToplamAvans (

p\_personelID IN NUMBER

) RETURN NUMBER AS

v\_toplamAvans NUMBER := 0;

BEGIN

SELECT

NVL(SUM(miktar), 0)

INTO

v\_toplamAvans

FROM

Avans

WHERE

personelID = p\_personelID;

RETURN v\_toplamAvans;

END;

/

-- 1. Personelin Toplam Avansını Hesaplayan Fonksiyon Kullanımı

DECLARE

v\_toplamAvans NUMBER;

BEGIN

v\_toplamAvans := GetPersonelToplamAvans(1); -- Personel ID'sini belirtin

DBMS\_OUTPUT.PUT\_LINE('Toplam Avans: ' || v\_toplamAvans);

END;

/

metin, yazı tipi, beyaz, cebir içeren bir resim

Açıklama otomatik olarak oluşturuldu

CREATE OR REPLACE FUNCTION GetPersonelToplamIzinGunleri (

p\_personelID IN NUMBER

) RETURN NUMBER AS

v\_toplamIzin NUMBER := 0;

BEGIN

SELECT

NVL(SUM(bitisTarihi - baslangicTarihi + 1), 0)

INTO

v\_toplamIzin

FROM

Izinler

WHERE

personelID = p\_personelID;

RETURN v\_toplamIzin;

END;

/

DECLARE

v\_personelID NUMBER := 1; -- İlgili personel ID'sini belirtin

v\_toplamIzin NUMBER;

BEGIN

v\_toplamIzin := GetPersonelToplamIzinGunleri(v\_personelID);

DBMS\_OUTPUT.PUT\_LINE('Personel ID: ' || v\_personelID || ', Toplam Kullandığı İzin Günleri: ' || v\_toplamIzin);

END;

/

CREATE OR REPLACE FUNCTION GetPersonelNetIzinGunleri (

p\_personelID IN NUMBER

) RETURN NUMBER AS

v\_toplamIzin NUMBER := 0;

v\_ucretliIzin NUMBER := 0;

v\_netIzin NUMBER := 0;

BEGIN

-- Toplam izin günlerini hesapla

SELECT

NVL(SUM(bitisTarihi - baslangicTarihi + 1), 0)

INTO

v\_toplamIzin

FROM

Izinler

WHERE

personelID = p\_personelID;

Çıktı:

metin, yazı tipi, beyaz, makbuz içeren bir resim

Açıklama otomatik olarak oluşturuldu

-- Ücretli izin günlerini hesapla

SELECT

NVL(SUM(bitisTarihi - baslangicTarihi + 1), 0)

INTO

v\_ucretliIzin

FROM

Izinler

WHERE

personelID = p\_personelID

AND ucretliMi = 'TRUE';

-- Net izin günlerini hesapla

v\_netIzin := v\_toplamIzin - v\_ucretliIzin;

RETURN v\_netIzin;

END;

/

DECLARE

v\_personelID NUMBER := 4; -- İlgili personel ID'sini belirtin

v\_netIzin NUMBER;

BEGIN

v\_netIzin := GetPersonelNetIzinGunleri1(v\_personelID);

DBMS\_OUTPUT.PUT\_LINE('Personel ID: ' || v\_personelID || ', Net İzin Günleri: ' || v\_netIzin);

END;

/

Çıktı:

metin, yazı tipi, beyaz, makbuz içeren bir resim

Açıklama otomatik olarak oluşturuldu

CREATE OR REPLACE FUNCTION GetPersonelNetMaas (

p\_personelID IN NUMBER

) RETURN NUMBER AS

v\_aylikMaas NUMBER := 0;

v\_toplamAvans NUMBER := 0;

v\_netMaas NUMBER := 0;

BEGIN

-- Aylık maaşı hesapla

SELECT

NVL(MAX(miktar), 0)

INTO

v\_aylikMaas

FROM

MaasOdeme

WHERE

personelID = p\_personelID;

-- Toplam avansı hesapla

SELECT

NVL(SUM(miktar), 0)

INTO

v\_toplamAvans

FROM

Avans

WHERE

personelID = p\_personelID;

-- Net maaşı hesapla

v\_netMaas := v\_aylikMaas - v\_toplamAvans;

RETURN v\_netMaas;

END;

/

DECLARE

v\_personelID NUMBER := 1; -- İlgili personel ID'sini belirtin

v\_netMaas NUMBER;

BEGIN

v\_netMaas := GetPersonelNetMaas(v\_personelID);

DBMS\_OUTPUT.PUT\_LINE('Personel ID: ' || v\_personelID || ', Net Maaş: ' || v\_netMaas);

END;

/

Çıktı:

metin, yazı tipi, beyaz, makbuz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Her bir fonksiyon, belirli bir amaca hizmet eder ve ilgili bilgileri çekerek işlemler gerçekleştirir. Kullanıcı, bu fonksiyonları çağırarak belirli personellerin avans, izin ve maaş bilgilerini elde edebilir.

DBMS\_OUTPUT.PUT\_LINE ifadeleri, bu bilgileri ekrana yazdırmak için kullanılır.

IF-ELSE: SQL dilinde, genellikle WHERE veya SELECT sorgularında kullanılır. Bir koşulun sağlanıp sağlanmadığını kontrol eder ve belirli bir eylemi gerçekleştirir.

INTO: SQL'de bir sorgunun sonucunu bir değişkene atamak için kullanılır. Özellikle SELECT sorgularında kullanılır.

NVL: Oracle veritabanı sistemine özgü bir ifadedir ve NULL değerleri kontrol etmek için kullanılır. Eğer belirtilen ifade NULL ise, belirtilen değeri döndürür; değilse orijinal değeri döndürür

Triggerlar

CREATE OR REPLACE TRIGGER DogumTarihiKontrolTrigger

BEFORE INSERT OR UPDATE ON Personel

FOR EACH ROW

DECLARE

v\_guncelTarih DATE := SYSDATE;

BEGIN

-- Eklenen veya güncellenen doğum tarihi şu andan büyükse exception fırlat

IF :NEW.dogumTarihi > v\_guncelTarih THEN

RAISE\_APPLICATION\_ERROR(-20006, 'Personelin doğum tarihi şu andan büyük olamaz.');

END IF;

END;

/

-- Bu tetikleyici, yeni bir personel kaydı eklenirken veya mevcut bir kayıt güncellenirken, doğum tarihinin şu anki tarihten büyük olup olmadığını kontrol eder. Eğer doğum tarihi şu anki tarihten büyükse, bir hata fırlatılır ve işlem gerçekleşmez.

v\_guncelTarih DATE := SYSDATE;: v\_guncelTarih adında bir değişken tanımlanır ve bu değişkene sistem tarihi (SYSDATE) atanır. Bu, mevcut tarihi temsil eder.

IF :NEW.dogumTarihi > v\_guncelTarih THEN: Yeni eklenen veya güncellenen kayıtın dogumTarihi alanı, mevcut tarihten büyükse, bir hata fırlatılır. :NEW öneki, yeni eklenen veya güncellenen değeri temsil eder.

RAISE\_APPLICATION\_ERROR(-20006, 'Personelin doğum tarihi şu andan büyük olamaz.');: Belirtilen hata kodu ve mesajı ile bir hata fırlatılır. Bu durumda, "Personelin doğum tarihi şu andan büyük olamaz." şeklinde bir hata mesajı verilir.

CREATE OR REPLACE TRIGGER MaasZammiTrigger

AFTER INSERT ON MaasZamlari

FOR EACH ROW

BEGIN

-- Yeni bir maaş zammı yapıldığında, Personel tablosundaki mevcut maaşı güncelle

UPDATE Personel

SET mevcutMaas = mevcutMaas + :NEW.miktar

WHERE personelID = :NEW.personelID;

END;

/

Amaç: Yeni bir maaş zammı yapıldığında, ilgili personelin mevcut maaşını güncellemek.

İşleyiş: AFTER INSERT ON MaasZamlari: MaasZamlari tablosuna yeni bir satır eklendikten sonra çalışır.

FOR EACH ROW: Her bir eklenen satır için tetikleyici çalışır.

UPDATE Personel: Personel tablosunu günceller.

SET mevcutMaas = mevcutMaas + :NEW.miktar: Yeni maaş zammını personelin mevcut maaşına ekler.

WHERE personelID = :NEW.personelID: Sadece ilgili personelin maaşını günceller.

CREATE OR REPLACE TRIGGER YillikIzinKontrolTrigger

BEFORE INSERT ON YillikIzinler

FOR EACH ROW

DECLARE

v\_toplamKullanilanIzin NUMBER;

v\_yillikIzinHakki NUMBER;

BEGIN

-- Personelin toplam kullanılan izin miktarını al

SELECT NVL(SUM(bitisTarihi - baslangicTarihi + 1), 0)

INTO v\_toplamKullanilanIzin

FROM YillikIzinler

WHERE personelID = :NEW.personelID;

-- Personelin yıllık izin hakkını al

SELECT topalmYillikIzinHakki

INTO v\_yillikIzinHakki

FROM Personel

WHERE personelID = :NEW.personelID;

-- Toplam kullanılan izin miktarı, yıllık izin hakkından fazlaysa exception fırlat

IF v\_toplamKullanilanIzin + (:NEW.bitisTarihi - :NEW.baslangicTarihi + 1) > v\_yillikIzinHakki THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Personelin yıllık izin hakkı aşıldı.');

END IF;

END;

/

Amaç: Yeni bir yıllık izin talebi yapıldığında, toplam kullanılan izin miktarını kontrol etmek ve izin hakkını aşan talepleri engellemek.

İşleyiş: BEFORE INSERT ON YillikIzinler: YillikIzinler tablosuna yeni bir satır eklenmeden önce çalışır.

FOR EACH ROW: Her bir eklenen satır için tetikleyici çalışır. DECLARE: Değişken tanımlamaları yapılır.

v\_toplamKullanilanIzin: İlgili personelin toplam kullanılan izin miktarını depolamak için kullanılır.

v\_yillikIzinHakki: İlgili personelin yıllık izin hakkını depolamak için kullanılır. SELECT: YillikIzinler ve Personel tablolarından gerekli bilgileri çeker.

IF: Toplam kullanılan izin miktarı, yıllık izin hakkını aşıyorsa, bir hata mesajı fırlatılır (RAISE\_APPLICATION\_ERROR)

CREATE OR REPLACE TRIGGER AvansKontrolTrigger

BEFORE INSERT ON Avans

FOR EACH ROW

DECLARE

v\_personelMaas NUMBER;

BEGIN

-- Personelin mevcut maaşını al

SELECT mevcutMaas

INTO v\_personelMaas

FROM Personel

WHERE personelID = :NEW.personelID;

-- Eklenen avans miktarı, personel maaşından büyükse exception fırlat

IF :NEW.miktar > v\_personelMaas THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Personelin maaşından büyük avans eklenemez.');

END IF;

END;

/

WHERE

d.departmanID = 1;

JOB:

CREATE OR REPLACE PROCEDURE YATIR\_MAAS\_JOB AS

BEGIN

FOR rec IN (SELECT personelID FROM Personel) LOOP

-- Personelin net maaşını al

DECLARE

v\_netMaas NUMBER;

BEGIN

v\_netMaas := GetPersonelNetMaas(rec.personelID);

-- Net maaşı MaasOdeme tablosuna ekle

BEGIN

INSERT INTO MaasOdeme (maasID, personelID, miktar, tarih)

VALUES (MaasOdemeSeq.NEXTVAL, rec.personelID, v\_netMaas, SYSTIMESTAMP);

-- Opsiyonel: İşlemi izlemek için çıktı ver

DBMS\_OUTPUT.PUT\_LINE('Personel ID ' || rec.personelID || ' için maaş yatırıldı. Net Maaş: ' || v\_netMaas);

END;

END;

END LOOP;

END;

Hesaplanan net maaş, "MaasOdeme" tablosuna eklenir.

DBMS\_OUTPUT.PUT\_LINE ifadesiyle işlemler takip edilebilir bir şekilde ekrana yazdırılır.

BEGIN

DBMS\_SCHEDULER.create\_job (

job\_name => 'MAAS\_YATIRMA\_JOB',

job\_type => 'PLSQL\_BLOCK',

job\_action => 'BEGIN YATIR\_MAAS\_JOB; END;',

start\_date => SYSTIMESTAMP,

repeat\_interval => 'FREQ=MONTHLY; BYMONTHDAY=25', -- Her ayın 25'inde çalışacak

enabled => TRUE

);

END;

--Bu iş, "MAAS\_YATIRMA\_JOB" adında bir zamanlanmış görev oluşturur. Bu görev, YATIR\_MAAS\_JOB prosedürünü belirtilen tarih ve tekrar aralığıyla çalıştıracaktır.

CREATE OR REPLACE PROCEDURE YILLIK\_ZAM\_JOB AS

BEGIN

FOR rec IN (SELECT personelID FROM Personel) LOOP

-- Personelin mevcut maaşını al

DECLARE

v\_mevcutMaas NUMBER;

BEGIN

SELECT mevcutMaas INTO v\_mevcutMaas FROM Personel WHERE personelID = rec.personelID;

-- Yıllık zammı hesapla (örneğin, %5)

BEGIN

DECLARE

v\_yillikZamOrani NUMBER := 0.05; -- %5 zam

v\_yillikZamMiktari NUMBER;

BEGIN

v\_yillikZamMiktari := v\_mevcutMaas \* v\_yillikZamOrani;

-- Yıllık zammı MaasZamlari tablosuna ekle

BEGIN

INSERT INTO MaasZamlari (zammID, personelID, miktar, tarih)

VALUES (MaasZamlariSeq.NEXTVAL, rec.personelID, v\_yillikZamMiktari, SYSTIMESTAMP);

-- Opsiyonel: İşlemi izlemek için çıktı ver

DBMS\_OUTPUT.PUT\_LINE('Personel ID ' || rec.personelID || ' için yıllık zam uygulandı. Zam Miktarı: ' || v\_yillikZamMiktari);

END;

END;

END;

END;

END LOOP;

END;

Bu prosedür, "Personel" tablosundaki her bir personel için bir döngü oluşturur. Her personelin mevcut maaşını alır ve belirli bir yıllık zam oranına göre zam miktarını hesaplar.

Hesaplanan zam miktarını, "MaasZamlari" tablosuna ekler. DBMS\_OUTPUT.PUT\_LINE ifadesiyle işlemler takip edilebilir bir şekilde ekrana yazdırılır.

BEGIN

DBMS\_SCHEDULER.create\_job (

job\_name => 'YILLIK\_ZAM\_JOB',

job\_type => 'PLSQL\_BLOCK',

job\_action => 'BEGIN YILLIK\_ZAM\_JOB; END;',

start\_date => TO\_TIMESTAMP('2024-01-01 00:00:00', 'YYYY-MM-DD HH24:MI:SS'),

repeat\_interval => 'FREQ=YEARLY; BYMONTH=1; BYMONTHDAY=1; BYHOUR=0', -- Her yılın 1 Ocak günü çalışacak

enabled => TRUE

);

END;

Bu iş, "YILLIK\_ZAM\_JOB" adında bir zamanlanmış görev oluşturur.

Bu görev, YILLIK\_ZAM\_JOB prosedürünü belirtilen tarih ve tekrar aralığıyla çalıştıracaktır.