FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ

2021-2022 AKADEMİK YILI GÜZ DÖNEMİ

BLM19303

Veri Tabanı Yönetim Sistemleri Final Projesi

DEMİRBAŞ VERİ TABANI MODELİ

Dr. Öğr. Üyesi Ali NİZAM

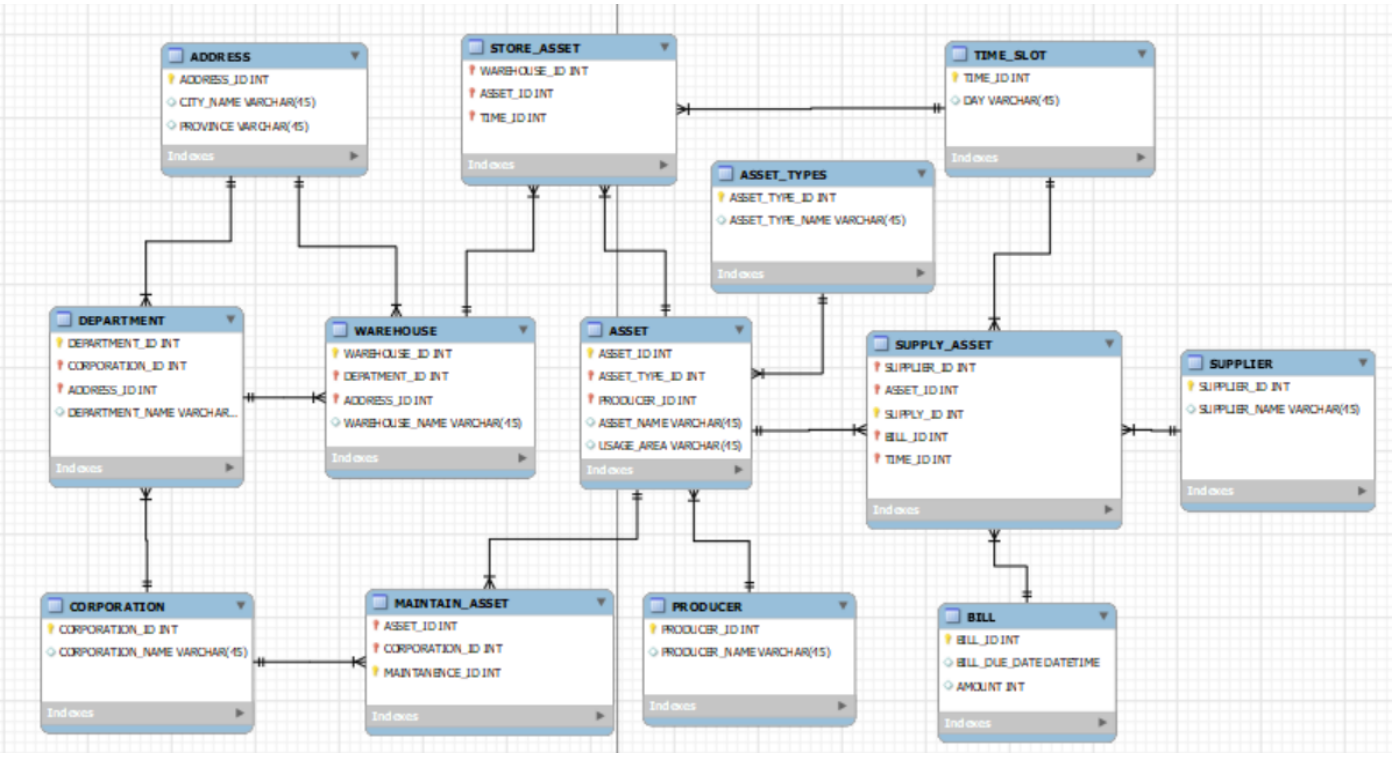
Kevser Büşra YILDIRIM

1821221029

Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

13.01.2022



SUPPLY_ASSET

- Tedarik edilmiş demirbaşların bilgisini tutar.
- Supplier, Time_Slot, Asset ve Bill tablolarının birbirleriyle olan çoka çok ilişkilerini sağlayan tablodur.

SUPPLIER

- Demirbaşları tedarik eden tedarikçilerin bilgisini tutar.

BILL

- Tedarik edilen demirbaşların fatura bilgilerini tutar.

ASSET

- Demirbaş tablosunun kendisidir.
- Üreticisinin ve türünün bilgisini tutar.

ASSET_TYPES

- Demirbaşların türlerinin bilgisini tutar.
- Bu türlerine örnek olarak, günlük hayatta kurumsal iş yerlerinde bulunan demirbaş ürünler örnek olarak verilebilir.

TIME_SLOT

- Gün ve zaman bilgisi tutar.

PRODUCER

- Demirbaş ürünlerinin üreticisinin bilgilerini tutar.
- Demirbaş ile bire çok ilişkisi vardır.

- Bir üretici birden fazla demirbaş urun üretebilir.

WAREHOUSE

- Demirbaş ürünleri alındıktan sonra tutulduğu depodur.
- İlgili kurumun departman ve adres bilgileri tutar

STORE_ASSET

- Warehouse ve demirbaş tablolarının çoka çok ilişkisini tanımlayan tablodur.
- Depolanmış demirbaş ürünleri tutar.
- Warehouse, demirbaş ve time bilgilerini tutar.

CORPORATION

- Demirbaşları alan kurum tablosudur.
- Kurum IDsi ve kurum ismi bilgilerini tutar.

MAINTAIN_ASSET

- Demirbaş ve kurum tablolarının çoka çok ilişkisini tanımlayan tablodur.
- Bakımları yapılan demirbaşların bilgisini tutar.
- Demirbaş id, kurum id ve bakim id'lerini tutar.

DEPARTMENT

- Kurumların departman bilgilerini tanımlar.
- Departman tablosu warehouse ve kurum tablolarıyla bire çok ilişkiye sahiptir.
- Bir departmana ait birden fazla warehouse olabilir.
- Bir kurumun da birden fazla departmanı olabilir.

ADDRESS

- Adres bilgilerini kaydeden tablodur.
- Şehir ismi ve konum bilgisi tutar.

*Producer(üretici) ve supplier(tedarik şirketi)

NORMALİZASYON

ASSET_ID	ASSET_TYPE_ID	ASSET_TYPE	PRODUCER_ID	PRODUCER_NAME	ASSET_NAME	USAGE_AREA
1	786, 435	ELEC,INDUSTRIAL	34	CAFEMARKT	ELECTRİC COOKER	KITCHEN
2	435	INDUSTRIAL	79	RAKLE	GLASS	SERVICE
3	6745	TEXTILE	46	BRILLANT	CURTAIN	LOBBY

Bu tabloda 1. satırda kategori isimlerinde bir hücrede birden fazla kaydolduğu için birinci normal forma uymamaktadır. Tekrar eden kayıtlar için, bir kayıt daha ekleyerek bu tekrarı önleyebiliriz.

ASSET_ID	ASSET_TYPE_ID	ASSET_TYPE	PRODUCER_ID	PRODUCER_NAME	ASSET_NAME	USAGE_AREA
1	786	ELECTRONIC	34	CAFEMARKT	ELECTRİC COOKER	KITCHEN
2	435	INDUSTRIAL	79	RAKLE	GLASS	SERVICE
3	6745	TEXTILE	46	BRILLANT	CURTAIN	LOBBY
1	435	INDUSTRIAL	34	CAFEMARKT	ELECTRİC COOKER	KITCHEN

- Birinci normal formdaki tabloyu ikinci normal forma getirelim.

{asset_id, asset_type_id, producer_id} →{asset_name, usage_area, asset_type}

Asset_id →asset_name, usage_area

R1 {asset_type_id }→{ asset_type }

R2 { producer_id }→{ producer_name }

R1	
ASSET_TYPE_ID	ASSET_TYPE
786	ELECTRONIC
6745	TEXTILE
435	INDUSTRIAL

R2	
PRODUCER_ID	PRODUCER_NAME
34	CAFEMARKT
79	RAKLE
46	BRILLANT

- İkinci normal formdan sonraki son durum:

ASSET_ID	ASSET_TYPE_ID	PRODUCER_ID	PRODUCER_NAME	ASSET_NAME	USAGE_AREA
1	786	34	CAFEMARKT	ELECTRİC COOKER	KITCHEN
2	435	79	RAKLE	GLASS	LOBBY
3	6745	46	BRILLANT	CURTAIN	KITCHEN

- Son durumdaki tabloyu üçüncü normal forma getirelim:

{ assed_id } → { assed_name, usage_area, assed_type_id, asset_type, producer_id, producer_name }

R1 { usage_area_id → usage_area }

R2 { assed_id → assed_name, usage_area_id, assed_type_id, asset_type, producer_id, producer_name }

ASSET_ID	ASSET_TYPE_ID	ASSET_TYPE_ID	ASSET_TYPE	PRODUCER_ID	USAGE_AREA_ID	PRODUCER_NAME
1	786	786	ELECTRONIC	34	1	CAFEMARKT
2	435	6745	TEXTILE	79	2	RAKLE
3	6745	435	INDUSTRIAL	46	3	BRILLANT

SORULAR

1. Bakım gören mutfak demirbaşlarını sıralayınız.
   SQL:
   SELECT * **FROM** assets a
   JOIN maintain_assets m **ON** m.asset_id = a.asset_id
   AND a.usage_area = 'KITCHEN'

   İlişkisel Cebir:
   Π asset.usage_area (**σ** asset.asset_id = maintain_asset.asset_id AND asset.usage_area = 'KITCHEN' (ASSET **x** MAINTAIN_ASSET))

2. Herhangi bir deposu olmayan kurumları bulunuz.
   SQL:
   SELECT c.corporation_id, c.corporation_name
   FROM corporation c, department d
   WHERE c.corporation_id = d.corporation_id
   AND d.department_id in (**SELECT** department_id **FROM** department
   MINUS
   SELECT department_id **FROM** warehouse)

   İlişkisel Cebir:
   Π corporation.corporation_name(**σ** corporation.corporation_id = department.corporation_id – warehouse.department_id (CORPORATION x DEPARTMENT)

3. Tedarik edilmiş demirbaş urunu olan üreticilerinin sayısını bulunuz.
   SQL:
   SELECT p.producer_id **COUNT**(p.producer_id) as producer_number
   FROM supply_asset sa
   JOIN asset a **ON** a.asset_id = sa.asset_id
   JOIN producer p **ON** p.producer_id = a.producer_id
   GROUP BY p.producer_id

   İlişkisel Cebir:
   γ asset**.** producer_id (producer ⋈ producer.asset_id = asset.asset_id asset ⋈ asset.ID = supply_asset.asset_id supply_asset)

PL SQL

1. Fonksiyon: Parametre olarak verilen üreticinin `Electronic` kategorisinde ürettiği demirbaşları var mı kontrol ediniz. Var ise demirbaşları yazdırınız.

```
CREATE OR REPLACE PROCEDURE supplied_electronic_assets(producer_name VARCHAR2(45)) AS
BEGIN
    FOR r_sup_assets (SELECT * FROM supply_asset) LOOP
        FOR r_assets (SELECT * FROM asset WHERE asset_id = r_sup_assets.asset_id) LOOP
            FOR r_asset_types (SELECT * FROM asset_types WHERE asset_type_id = r_assets.asset_type_id) LOOP
                FOR r_producer (SELECT * FROM producer WHERE producer_id = r_assets.producer_id) LOOP
                    IF r_asset_types.asset_type_name == 'Electronic' and r_producer.producer_name == producer_name THEN
                        DBMS.OUTPUT.PUT_LINE(r_sup_assets);
                    ELSE
                        DBMS.OUTPUT.PUT_LINE('Bu ureticinin Electronic turunde urettigi demirbas bulunmamaktadir..');
                    END IF;
                END LOOP;
            END LOOP;
        END LOOP;
    END LOOP;
END;
```

2. Tablo Trigger: Sisteme yeni bir demirbaş türü eklendiğinde hala 2'den az demirbaş türü olan üreticiler varsa siliniz.

```
CREATE OR REPLACE TRIGGER check_producer
BEFORE INSERT ON asset_types
FOR EACH ROW
BEGIN
    DELETE FROM producer
    WHERE producer_id IN (SELECT producer_id
                          FROM assets a, asset_types at, producer p
                          WHERE a.asset_type_id = at.asset_type_id
                          AND p.producer_id = a.producer_id
                          GROUP BY producer_id
                          HAVING COUNT (at.asset_type_id < 2))
```

DENORMALİZASYON

- Denormalizasyon normalleştirme sürecinin tersi işlemidir. Denormalizasyon, performansı optimize etmek için yedekli veriler ekleyerek veya verileri gruplayarak çalışır.
- Denormalizasyon genellikle veritabanının okuma performansını artırmak için gerçekleştirilir, ancak denormalizasyon için kullanılan ek kısıtlamalar nedeniyle, yazma işlemleri (yani ekleme, güncelleme ve silme işlemleri) yavaşlayabilir. Bu nedenle, normalleştirilmemiş bir veritabanı normalleştirilmiş bir veritabanından daha kötü yazma performansı sunabilir.

Tek Satır Denormalizasyon Örneği:

SELECT * **FROM** bill

WHERE bill_year = 2021

- Bu denormalizasyon işlemi ek alan, ekleme maliyetlerine sebep olmaktadır fakat sorgu performansı artmıştır.
- Çoklu satırlarda Denormalizasyon yapılırken tabloda kullanılmayan kolonlar yeni bir alanda toplanabilir.
- Birden fazla tablo ile denormalizasyon işlemi gerçekleştirilecekse örneğin Fatura bilgileri için müşteri_id ile müşteri adına her defasında JOIN ile ulaşmak performansı düşürmeye sebep olabilir. Bu durumda müşteri_adı Fatura tablosuna taşınabilir.
- Denormalizasyonda veri tutarlılığına dikkat edilmelidir.