T.C. FIRAT ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

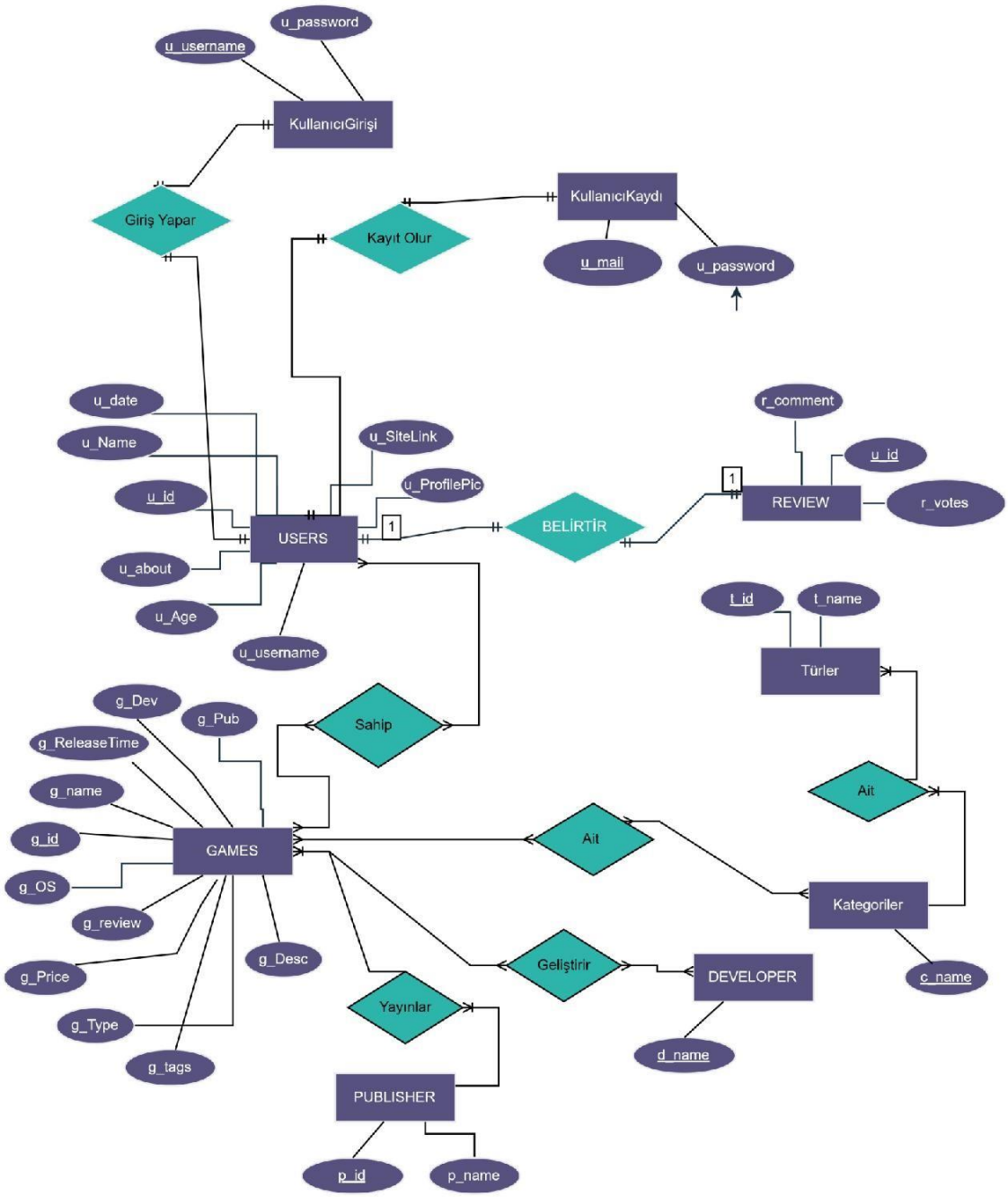
Oyun Kütüphanesi Veri Tabanı Projesi Raporu

220260036 Ahmet Salih Doğan

210260020 Sevgi PEKİN

210260060 Sıray TARIM

Projenin E-R Diyagramı



Projenin İlişkisel Şeması :

Yukarıdaki E-R Şemasına göre ilişkisel şemamızı oluşturuyoruz.

- Users(u_id (PK), u_Name, u_date, u_username, u_SiteLink, u_ProfilePic, u_about, u_age)
- Games(g_id (PK), g_name, g_dev(FK), g_pub(FK), g_releaseTime, g_os, g_price, g_review, g_Desc, g_type, g_tags)
- Developers(d_name(PK))
- Publishers(p_id (PK), p_name)
- Reviews(r_id (PK), g_id(FK), r_comment, r_votes, u_id (FK))
- KullaniciGirisi(u_username (PK), u_password)
- KullaniciKaydi(u_mail (PK), u_password)
- Turler(t_id (PK), t_name)
- Kategoriler(c_name (PK))

Normalizasyon İşlemi Hakkında :

Kullanıcı mail bilgisi de id'si de kullanıcıyı eşsiz tanımlayan özellikler olduğundan dolayı bunları ayrı tablolara koyduk.(BCNF)

Burada tekrar eden veri gruplarını ayrı tablolara bölerek veri tekrarını azalttık (TÜR,KATEGORİLER gibi).

Kullanıcının profil ve giriş bilgilerini birbirinden bağımsız olarak güncelleyebilmemiz için ayrı tablolarda tuttuk.

Tablolardaki sütunları sadece birincil anahtara bağlı olacak şekilde güncelledik. Mesela kategoriler tablosunda kategori adı sadece kategori id'sine bağlıdır. Zaten başka sütun da yok BCNF'e uygundur. Diğer tablolara da aynısını uyguladık.

VERİTABANINI VE TABLOLARI OLUŞTURMA

İlk olarak veritabanımızı **CREATE DATABASE** komutuyla oluşturuyoruz.

```
CREATE DATABASE GameLibrary
```

USERS TABLOSU :

```
CREATE TABLE Users (
    u_id INT PRIMARY KEY,
    u_Name VARCHAR(100) NOT NULL,
    u_date DATE NOT NULL,
    u_username VARCHAR(50) UNIQUE NOT NULL,
    u_SiteLink VARCHAR(255),
    u_ProfilePic VARCHAR(255),
    u_about VARCHAR(MAX),
    u_age INT CHECK (u_age >= 13)
);
```

DEVELOPERS TABLOSU :

```
CREATE TABLE Developers (
    d_name VARCHAR(100) PRIMARY KEY
);
```

PUBLİSHERS TABLOSU :

```
   CREATE TABLE Publishers (
   p_id INT PRIMARY KEY,
   p_name VARCHAR(100) NOT NULL
);
```

GAMES TABLOSU :

```
CREATE TABLE Games (
    g_id INT PRIMARY KEY,
    g_name VARCHAR(100) NOT NULL,
    g_dev VARCHAR(100) NOT NULL,
    g_pub VARCHAR(100) NOT NULL,
    g_releaseTime DATE NOT NULL,
    g_os VARCHAR(50),
    g_price DECIMAL(10, 2) CHECK (g_price >= 0),
    g_review VARCHAR(MAX),
    g_Desc VARCHAR(MAX),
    g_type VARCHAR(50),
    g_tags VARCHAR(255),
    FOREIGN KEY (g_dev) REFERENCES Developers(d_name),
    FOREIGN KEY (g_pub) REFERENCES Publishers(p_id)
);
```

REVİEWS TABLOSU :

```
CREATE TABLE Reviews (
    r_id INT PRIMARY KEY,
    g_id INT NOT NULL,
    r_comment VARCHAR(MAX),
    r_votes INT DEFAULT 0,
    u_id INT NOT NULL,
    FOREIGN KEY (g_id) REFERENCES Games(g_id),
    FOREIGN KEY (u_id) REFERENCES Users(u_id)
);
```

KullaniciKaydi TABLOSU :

```
CREATE TABLE KullaniciKaydi (
    u_mail VARCHAR(255) PRIMARY KEY,
    u_password VARCHAR(255) NOT NULL,
    FOREIGN KEY (u_mail) REFERENCES Users(u_mail)
);
```

Türler TABLOSU :

```
CREATE TABLE Turler (
    t_id INT PRIMARY KEY,
    t_name VARCHAR(100) NOT NULL
);
```

Kategoriler TABLOSU :

```
CREATE TABLE Kategoriler (
    c_name VARCHAR(100) PRIMARY KEY
);
```

TABLOLARA DEĞERLER EKLEME

INSERT INTO Komutuyla Örnek Değerleri Ekleyebiliriz.

USER Tablosuna Değer Ekleme :

```
INSERT INTO Users (u_Name, u_password, u_mail, u_date,
u_username, u_SiteLink, u_ProfilePic, u_about, u_age)
VALUES
('Ahmet Salih', '123456', 'asalihdogan@example.com',
'2004-01-01', 'salih16', 'https://asalih.com',
'profilepic.jpg', 'Baklava severim.', 21),
('Ali Demir', 'sifre123', 'alidemir@example.com',
'1993-08-22', 'xkraltr',  'https://ademir.com ',
'profilepic1.png', 'Oyunları seven adam.', 32);
```

	u_id	u_Name	u_password	u_mail	u_date	u_username	u_SiteLink	u_ProfilePic	u_about	u_age
1	1	Ahmet Salih	123456	asalihdogan@example.com	2004-01-01	salih16	https://asalih.com	profilepic.jpg	Baklava severim.	21
2	2	Ali Demir	sifre123	alidemir@example.com	1993-08-22	xkraltr	https://ademir.com	profilepic1.png	Oyunları seven adam.	32

DEVELOPERS Tablosuna Değer Ekleme :

```
INSERT INTO Developers (d_name)
VALUES
('Obsidian Games'),
('Valve Corporation'),
('CD Projekt Red');
```

	d_name
1	CD Projekt Red
2	Obsidian Games
3	Valve Corporation

PUBLİSHERS Tablosuna Değer Ekleme :

```
INSERT INTO Publishers (p_name)
VALUES
('Electronic Arts'),
('Bethesda Games'),
('Criterion Games ');
```

	p_id	p_name
1	1	Electronic Arts
2	2	Bethesda Games
3	3	Criterion Games

GAMES Tablosuna Değer Ekleme :

```
INSERT INTO Games (g_id, g_name, g_dev, g_pub, g_releaseTime,
g_os, g_price, g_review, g_Desc, g_type, g_tags)
VALUES
('Counter Strike 2', 'Valve Corporation', 1, '2023-09-27',
'Windows', 0.00, 'Mostly Positive', 'FPS game.', 'FPS',
'Competitive', 'Multiplayer'),
('The Witcher 3', 'CD Projekt Red', 3, '2015-05-19', 'Windows',
29.99, 'Overwhelmingly Positive', 'Open-world RPG.', 'RPG',
'Open-world, Fantasy');
```

	g_id	g_name	g_dev	g_pub	g_releaseTime	g_os	g_price	g_review	g_Desc	g_type	g_tags
1	2	Counter Strike 2	Valve Corporation	1	2023-09-27	Windows	0.00	Mostly Positive	FPS game.	FPS	Multiplayer
2	3	The Witcher 3	CD Projekt Red	3	2015-05-19	Windows	29.99	Overwhelmingly Positive	Open-world RPG.	RPG	Open-world

REVIEWS Tablosuna Değer Ekleme :

```
INSERT INTO Reviews (g_id, r_comment, r_votes, u_id)
VALUES
(2, 'Oyuna güncelleme getirin artık.', 20, 1),
(3, 'Hayatımda oynadığım en iyi oyun.', 50, 2);
```

	r_id	g_id	r_comment	r_votes	u_id
1	1	2	Oyuna güncelleme getirin artık.	20	1
2	2	3	Hayatımda oynadığım en iyi oyun.	50	2

TURLER Tablosuna Değer Ekleme :

```
INSERT INTO Turler (t_name)
VALUES
('MOBA'),
('Action'),
('Visual Novel');
```

	t_id	t_name
1	1	MOBA
2	2	Action
3	3	Visual Novel

KATEGORİLER Tablosuna Değer Ekleme :

```
INSERT INTO Kategoriler (c_name)
VALUES
('18+'),
('Free to Play'),
('Multiplayer');
```

	c_name
1	18+
2	Free to Play
3	Multiplayer

TEMEL SQL KOMUTLARI İLE BASİT UYGULAMALAR

```sql
SELECT
    u.u_Name AS UserName,
    g.g_name AS GameName,
    r.r_comment AS ReviewComment,
    r.r_votes AS Votes
FROM
    Reviews r
JOIN
    Users u ON r.u_id = u.u_id
JOIN
    Games g ON r.g_id = g.g_id
ORDER BY
    r.r_votes DESC;
```

Mesela bu uygulamada incelemeyi yapan kullanıcının bilgileri USERS tablosundan getirilir ve incelemenin yapıldığı oyunun bilgileri de GAMES tablosundan getirilir. Sonra bu tablolar birleştirilip inceleme oylarına göre azalacak şekilde sıralanır. Aynı zamanda AS komutuyla da diğer tablolardan çektiğimiz verileri listelerken istediğimiz şekilde isimlendirebiliriz.

	UserName	GameName	ReviewComment	Votes
1	Ali Demir	The Witcher 3	Hayatımda oynadığım en iyi oyun.	50
2	Ahmet Salih	Counter Strike 2	Oyuna güncelleme getirin artık.	20
3	Enes Yılmaz	Counter Strike 2	Fena değil	6
4	Mustafa	The Witcher 3	Overrated	2

DELETE FROM Reviews

WHERE r_votes = 2 ;

Komutunu çalıştırdıktan sonra yukarıdaki ilk sorguyu tekrar yaparsak :

	UserName	GameName	ReviewComment	Votes
1	Ali Demir	The Witcher 3	Hayatımda oynadığım en iyi oyun.	50
2	Ahmet Salih	Counter Strike 2	Oyuna güncelleme getirin artık.	20
3	Enes Yılmaz	Counter Strike 2	Fena değil	6

Oy değeri 2 olan inceleme silindi ve kalanlar aynı şekilde listelendi.

```
SELECT
    g.g_name AS GameName,
    AVG(r.r_votes) AS AverageVotes
FROM
    Games g
INNER JOIN
    Reviews r ON g.g_id = r.g_id
GROUP BY
    g.g_name
ORDER BY
 AverageVotes DESC;
```

	GameName	AverageVotes
1	Half Life	102
2	Deadlock	54
3	The Witcher 3	50
4	Counter Strike 2	13

Bu örnek oyunlara yapılan incelemelerin oylamalarını azalan sırada listeler.

STORED PROCEDURE

Bir oyunun fiyatını güncelleme :

```
CREATE PROCEDURE UpdatePrice
    @GameID INT,
    @NewPrice DECIMAL(10, 2)
AS
BEGIN
    UPDATE Games
    SET g_price = @NewPrice
    WHERE g_id = @GameID;
END;
```

Artık bu prosedür yardımıyla ihtiyaç duyduğumuzda fiyat güncelleme işlemi yapabiliriz.

```
EXEC UpdateGamePrice @GameID = 3, @NewPrice = 15.99;
```

	g_id	g_name	g_price
1	2	Counter Strike 2	0.00
2	3	The Witcher 3	15.99
3	10	Half Life	0.00
4	11	Deadlock	29.99

Witcher 3'ün fiyatı, 24.99'dan 15.99'a güncellenmiş oldu.

TRIGGER KULLANIMI

Bir oyunun fiyatını güncellediğimizde onu kayıt eden (loglayan) bir sistem yapmak istiyoruz. Bunu trigger ile yapabiliriz.

```
CREATE TRIGGER LogPriceUpdate
ON Games
AFTER UPDATE
AS
BEGIN
    IF UPDATE(g_price)
    BEGIN
        INSERT INTO PriceUpdateLog (g_id, OldPrice, NewPrice,
UpdateDate)
        SELECT d.g_id, d.g_price, i.g_price, GETDATE()
        FROM deleted d
        JOIN inserted i ON d.g_id = i.g_id;
    END
END;
```

Triggerımızı oluşturduktan sonra bu logları görüntüleyebileceğimiz/tutabileceğimiz bir tabloya da ihtiyacımız olacak.

```
CREATE TABLE PriceUpdateLog (
    log_id INT PRIMARY KEY ,
    g_id INT NOT NULL,
    OldPrice DECIMAL(10, 2),
    NewPrice DECIMAL(10, 2),
    UpdateDate DATETIME NOT NULL,
    FOREIGN KEY (g_id) REFERENCES Games(g_id)
);
```

Tablo oluşturduktan sonra test edebilmek için önce bir fiyat güncelleme işlemi yapıyoruz.

```
EXEC UpdatePrice @GameID = 2, @NewPrice = 29.99;
```

	g_name	g_id	g_price
1	Counter Strike 2	2	29.99

Daha önce fiyatı 0 olan Counter Strike 2 oyununun fiyatı güncellenmiş. Şimdi bir de Trigger işleminin çalışıp çalışmadığını kontrol etmek için oluşturduğumuz log table'a sorgu işlemi yapalım.

```
SELECT * FROM PriceUpdateLog;
```

	log_id	g_id	OldPrice	NewPrice	UpdateDate
1	1	2	0.00	29.99	2025-01-05 00:02:38.213

Görüldüğü üzere işlem başarıyla loglandı.

TRANSACTION YÖNETİMİ

Birden fazla SQL komutunun çalışması sırasında çakışmaların önüne geçmek ve eğer bir aksilik olursa işlemin geri alınmasını sağlayabiliriz.

Bir oyunun fiyatı güncellendiğinde kullanıcıya bildirim yollayan bir sistem yapabiliriz. Bunu yapabilmek için bildirimler için ayrı bir tablo oluşturmak daha mantıklı olur.

```
CREATE TABLE Notifications (
    n_id INT PRIMARY KEY,
    u_id INT NOT NULL,
    message VARCHAR(MAX),
    created_at DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (u_id) REFERENCES Users(u_id)
);

BEGIN TRANSACTION;

BEGIN TRY
    -- Fiyatı güncelle.
    UPDATE Games
    SET g_price = 00.00
    WHERE g_id = 2;

    -- Kullanıcıya bildirim yolla.
    INSERT INTO Notifications (u_id, message)
    VALUES (1, 'Counter Strike 2 tekrardan ücretsiz oldu!');


    COMMIT TRANSACTION;
    PRINT 'İşlem başarılı.';
END TRY
BEGIN CATCH
    -- Hata varsa geri al
    ROLLBACK TRANSACTION;
    PRINT 'Hata. Rollback işlemi yapılıyor.';
    PRINT ERROR_MESSAGE();
END CATCH;
```

```
(1 row affected)

(1 row affected)

(1 row affected)
İşlem başarılı.

Completion time: 2025-01-05T00:31:23.5310635+03:00
```

Transaction işlemimiz başarılı. Şimdi bildirimin gelip gelmediğine ve fiyatın gerçekten güncellenip güncellenmediğine bakalım.

```
SELECT * FROM Notifications WHERE u_id = 1;
```

	n_id	u_id	message	created_at
1	1	1	Counter Strike 2 tekrardan ücretsiz oldu!	2025-01-05 00:31:23.517

Bildirimimiz geldi sırada fiyatı kontrol etmek var.

```
SELECT * FROM Games WHERE g_id = 2;
```

	g_id	g_name	g_dev	g_pub	g_releaseTime	g_os	g_price	g_review	g_Desc	g_type	g_tags
1	2	Counter Strike 2	Valve Corporation	1	2023-09-27	Windows	0.00	Mostly Positive	FPS game.	FPS	Multiplayer

Görüldüğü üzere daha önce fiyatını güncelleyip 29.99’a çektiğimiz Counter Strike 2 tekrardan ücretsiz. İşlem başarılı.

Peki hata vermesi gerektiği durumda veriyor mu?

```sql
BEGIN TRANSACTION;

BEGIN TRY
    -- Fiyatımızı güncelliyoruz.
    UPDATE Games
    SET g_price = 00.00
    WHERE g_id = 2;

    -- User'a bildirim yolla.
    INSERT INTO Notifications (u_id, message)
    VALUES (438025, 'Counter Strike 2 tekrardan ücretsiz oldu!');


    COMMIT TRANSACTION;
    PRINT 'İşlem başarılı.';
END TRY
BEGIN CATCH
    -- Hata olursa geri al.
    ROLLBACK TRANSACTION;
    PRINT 'Hata. Rollback işlemi yapılıyor.';
    PRINT ERROR_MESSAGE();
END CATCH;
```

Burada görüldüğü gibi Notifications tablosuna değerleri insert ederken u_id'nin değerine tablomda bulunmayan absürt bir değer ekledim.

```
(1 row affected)

(1 row affected)

(0 rows affected)
Hata. Rollback işlemi yapılıyor.
The INSERT statement conflicted with the FOREIGN KEY constraint "FK__Notificati__u_id__31B762FC". The conflict occurred in database "GameLibrary", table "dbo.Users"
```

İşlemler tam olarak tamamlanamadığı için rollback ifadesi burada başarılı olan işlemleri de geri alır.