

SAKARYA ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ
2025-2026 GÜZ DÖNEMİ
VERİ TABANI YÖNETİM SİSTEMLERİ DERSİ PROJE RAPORU

Hazırlayan: İsmail Fatih Çolak

Numara: B241210062

Tarih: 12 Aralık 2025

Projenin GitHub Reposu: <https://github.com/kuyash71/ttrpgDatabase>

a. Uygulamanın Kısa Tanıtımı ve İş Kuralları

Giriş

Bu proje kapsamında, masaüstü rol yapma oyunları (Tabletop Role-Playing Games – TTRPG) için çoklu evren ve sistem desteği sunan, ilişkisel veritabanı tabanlı bir karakter ve oyun yönetim sistemi tasarlanmıştır ve uygulanmıştır. Projenin amacı; farklı TTRPG sistemlerine ait evrenler, kampanyalar, karakterler, sınıflar, istatistikler ve envanter bilgilerinin tutarlı, genişletilebilir ve güvenli bir biçimde saklanmasıdır.

Bu doğrultuda PostgreSQL veritabanı yönetim sistemi kullanılarak; tablolar, birincil ve yabancı anahtarlar, stored procedure’ler ve trigger’lar aracılığıyla veri bütünlüğü sağlanmıştır. İş kuralları yalnızca uygulama seviyesinde değil, veritabanı seviyesinde de enforced edilerek hatalı veri girişlerinin önüne geçilmiştir. Ayrıca sistem, yeni oyun evrenleri ve kurallarının eklenmesine olanak tanıyacak şekilde modüler bir yapıda tasarlanmıştır.

İş Kuralları

Bu proje, masaüstü rol yapma oyunları (Tabletop Role-Playing Games – TTRPG) için birden fazla oyun sistemi ve evreni destekleyebilen bir karakter ve evren veritabanı yönetim sistemi geliştirmeyi amaçlamaktadır. Sistem; oyun evrenleri, kampanyalar, karakterler, sınıflar (class/preset), statlar, kaynaklar, yetenekler ve envanter gibi oyun bileşenlerinin düzenli, tutarlı ve genişletilebilir şekilde saklanması sağlar.

Aşağıda sistemin uyması gereken temel iş kuralları tanımlanmıştır.

1. Sistem ve Evren Kuralları

Sistem, birden fazla oyun sistemini (örneğin Umbra Caelis, Shwarzesonne gibi) destekleyebilmelidir.

Her oyun sistemi kendine özgü stat, kaynak ve class/preset tanımlarına sahiptir.

Bir evren, yalnızca bir oyun sistemine bağlıdır.

Bir evren birden fazla kampanya içerebilir.

Bir kampanya yalnızca bir evrene bağlıdır.

2. Karakter Kuralları

Her karakter yalnızca bir “campaign”e bağlıdır.

Her karakter yalnızca bir oyun sistemine ve bir class/preset’e sahiptir.

Bir kampanya içerisinde birden fazla karakter bulunabilir.

Karakterler oyuncu karakteri (PC) veya oyuncu olmayan karakter (NPC) olabilir.

Karakterin temel bilgileri (isim, hikâye, güçlü/zayıf yönler vb.) sistem tarafından saklanmalıdır.

3. Stat Kuralları

Statlar, oyun sistemine bağlı olarak tanımlanır.

Her oyun sistemi, kendine özgü bir stat kümesine sahiptir.

Her stat için minimum ve maksimum değerler tanımlanmalıdır.

Her karakter, bağlı olduğu oyun sistemindeki tüm statlara sahip olmak zorundadır.

Karakter stat değerleri, tanımlı minimum ve maksimum sınırların dışına çıkamaz.

Stat değerleri, karakter oluşturulurken otomatik olarak oluşturulmalıdır.

4. Class / Preset Kuralları

Her class/preset, yalnızca bir oyun sistemine bağlıdır.

Bir class/preset, karakterin başlangıç statlarını etkileyen stat değiştiricilere (arti/eksi değerler) sahip olabilir.

Class/preset stat değiştiricileri, karakter oluşturulurken otomatik olarak uygulanmalıdır.

Bir karakterin class/preset'i değiştirildiğinde, stat değiştiricileri yeniden hesaplanmalıdır.

Class/preset'ler, bir veya daha fazla yetenek (ability) içerebilir.

5. Kaynak (Resource) Kuralları

Kaynaklar (örneğin sağlık, esin puanı, yiyecek vb.) oyun sistemine bağlı olarak tanımlanır.

Her karakter, oyun sisteminde tanımlı tüm kaynaklara sahip olmalıdır.

Kaynak değerleri minimum ve maksimum sınırlar içinde tutulmalıdır.

Kaynak değerleri oyun sırasında güncellenebilir.

6. Yetenek (Ability) Kuralları

Yetenekler, oyun sistemine bağlı olarak tanımlanır.

Yetenekler açıklama ve kullanım kuralları içerebilir.

Bir yetenek birden fazla class/preset tarafından kullanılabilir.

Bir karakter, class/preset'i aracılığıyla birden fazla yeteneğe sahip olabilir.

7. Envanter Kuralları

Oyun sisteminde tanımlı eşyalar karakterlerin envanterine eklenebilir.

Bir karakter birden fazla eşyaya sahip olabilir.

Envanterdeki her eşyanın bir adet (quantity) bilgisi bulunmalıdır.

Bir eşyanın adedi sıfır olduğunda envanter kaydı otomatik olarak silinmelidir.

8. Veri Bütünlüğü ve Kayıt Kuralları

Sistem üzerinde yapılan ekleme, silme ve güncelleme işlemleri kayıt altına alınmalıdır.

Tanımlı olmayan bir stat, kaynak veya class/preset bir karaktere atanamaz.

Veritabanı bütünlüğü, yabancı anahtarlar, kısıtlamalar ve tetikleyiciler (trigger) ile korunmalıdır.

9. Uygulama İşlevleri Kuralları

Sistem; arama, ekleme, silme ve güncelleme (CRUD) işlemlerini desteklemelidir.

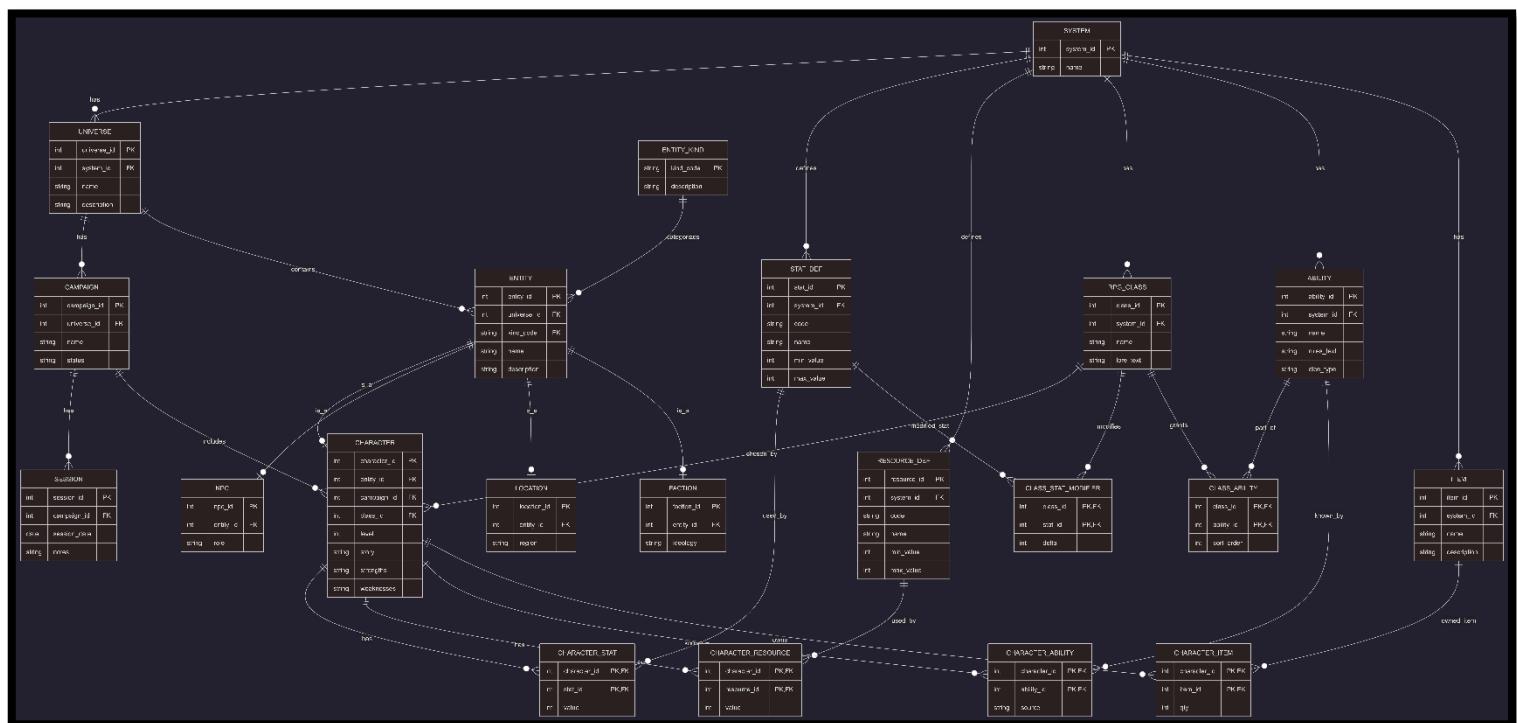
Kullanıcı, karakter bilgilerini ve statlarını görüntüleyebilmelidir.

Karakter oluşturma işlemleri veritabanı saklı yordamları (stored procedure) aracılığıyla gerçekleştirilmelidir.

İş kuralları, mümkün olduğunca veritabanı seviyesinde uygulanmalıdır.

b. Varlık Bağıntı Modeli

Projenin ER diyagramı aşağıda mevcuttur, görselin orijinali [github](#) reposunda yer almaktadır.



b.1. Projenin metinsel gösterim ile ilişkisel şeması:

```

CHARACTER_RESOURCE(
    character_id PK, FK → CHARACTER(character_id),
    resource_id PK, FK → RESOURCE_DEF(resource_id),
    value
)
RPG_CLASS(
    class_id PK,
    system_id FK → SYSTEM(system_id),
    name,
    lore_text
)
ABILITY(
    ability_id PK,
    system_id FK → SYSTEM(system_id),
    name,
    usage_text,
    dice_type
)
CLASS_STAT_MODIFIER(
    class_id PK, FK → RPG_CLASS(class_id),
    stat_id PK, FK → STAT_DEF(stat_id),
    delta
)
CLASS_ABILITY(
    class_id PK, FK → RPG_CLASS(class_id),
    ability_id PK, FK → ABILITY(ability_id),
    sort_order
)
CHARACTER_ABILITY(
    character_id PK, FK → CHARACTER(character_id),
    ability_id PK, FK → ABILITY(ability_id),
    source
)
ITEM(
    item_id PK,
    system_id FK → SYSTEM(system_id),
    name,
    description
)
CHARACTER_ITEM(
    character_id PK, FK → CHARACTER(character_id),
    item_id PK, FK → ITEM(item_id),
    qty
)

ENTITY(
    entity_id PK,
    universe_id FK → UNIVERSE(universe_id),
    kind_code FK → ENTITY_KIND(kind_code),
    name,
    description
)
CHARACTER(
    character_id PK,
    entity_id FK → ENTITY(entity_id),
    campaign_id FK → CAMPAIGN(campaign_id),
    class_id FK → RPG_CLASS(class_id),
    level,
    story,
    starting_loc,
    weaknesses
)
NPC(
    npc_id PK,
    entity_id FK → ENTITY(entity_id),
    role
)
LOCATION(
    location_id PK,
    entity_id FK → ENTITY(entity_id),
    region
)
SYSTEM(
    system_id PK,
    name
)
UNIVERSE(
    universe_id PK,
    system_id FK → SYSTEM(system_id),
    name,
    description
)
CAMPAIGN(
    campaign_id PK,
    universe_id FK → UNIVERSE(universe_id),
    name,
    status
)
SESSION(
    session_id PK,
    campaign_id FK → CAMPAIGN(campaign_id),
    session_date,
    notes
)
ENTITY_KIND(
    kind_code PK,
    description
)
FACTION(
    faction_id PK,
    entity_id FK → ENTITY(entity_id),
    ideology
)
STAT_DEF(
    stat_id PK,
    system_id FK → SYSTEM(system_id),
    code,
    name,
    min_value,
    max_value
)
RESOURCE_DEF(
    resource_id PK,
    system_id FK → SYSTEM(system_id),
    code,
    name,
    min_value,
    max_value
)
CHARACTER_STAT(
    character_id PK, FK → CHARACTER(character_id),
    stat_id PK, FK → STAT_DEF(stat_id),
    value
)

```

c.Veritabanını, içindeki veriler ile oluşturmayı sağlayan SQL ifadeleri

Veritabanını oluşturan tüm SQL ifadelerine “01_schema.sql” adlı dosyanın içerisindeinden erişilebilmektedir. Oluşturulan veritabanının içindeki başlangıç verilerini ise “02_seed.sql” adlı dosyanın içerisindeki SQL ifadeleri ile oluşturulur.

c.1. İlgili “01_schema.sql” dosyasının içindeki SQL ifadeleri:

```
40
41  CREATE TABLE campaign (
42    campaign_id  INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
43    universe_id   INTEGER NOT NULL REFERENCES universe(universe_id) ON DELETE CASCADE,
44    name          VARCHAR(160) NOT NULL,
45    status         VARCHAR(40),
46    CONSTRAINT uq_campaign_universe_name UNIQUE (universe_id, name)
47  );
48
49  CREATE TABLE session (
50    session_id    INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
51    campaign_id   INTEGER NOT NULL REFERENCES campaign(campaign_id) ON DELETE CASCADE,
52    session_date  DATE,
53    notes         TEXT
54  );
55  -- ENTITY KIND (Lookup) + ENTITY
56  CREATE TABLE entity_kind (
57    kind_code     VARCHAR(24) PRIMARY KEY,
58    description   VARCHAR(200)
59  );
60
61  CREATE TABLE entity (
62    entity_id     INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
63    universe_id   INTEGER NOT NULL REFERENCES universe(universe_id) ON DELETE CASCADE,
64    kind_code     VARCHAR(24) NOT NULL REFERENCES entity_kind(kind_code) ON DELETE RESTRICT,
65    name          VARCHAR(160) NOT NULL,
66    description   TEXT
67  );
68  -- KALITIM TABLOLARI
69  CREATE TABLE character (
70    character_id  INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
71    entity_id     INTEGER NOT NULL UNIQUE REFERENCES entity(entity_id) ON DELETE CASCADE,
72    campaign_id   INTEGER NOT NULL REFERENCES campaign(campaign_id) ON DELETE CASCADE,
73    class_id      INTEGER NOT NULL,
74    level         INTEGER NOT NULL DEFAULT 0 CHECK (level >= 0),
75    story         TEXT,
76    strengths    TEXT,
77    weaknesses   TEXT
78  );
79
80  CREATE TABLE npc (
81    npc_id        INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
82    entity_id     INTEGER NOT NULL UNIQUE REFERENCES entity(entity_id) ON DELETE CASCADE,
83    role          VARCHAR(120)
84  );
85
86  CREATE TABLE location (
87    location_id   INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
88    entity_id     INTEGER NOT NULL UNIQUE REFERENCES entity(entity_id) ON DELETE CASCADE,
89    region        VARCHAR(120)
90  );
91
92  CREATE TABLE faction (
93    faction_id   INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
94    entity_id     INTEGER NOT NULL UNIQUE REFERENCES entity(entity_id) ON DELETE CASCADE,
95    ideology      VARCHAR(160)
96  );
97  -- STAT / RESOURCE DEFINITIONS
98  CREATE TABLE stat_def (
99    stat_id       INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
100   system_id     INTEGER NOT NULL REFERENCES system(system_id) ON DELETE CASCADE,
101   code          VARCHAR(40) NOT NULL,
102   name          VARCHAR(120) NOT NULL,
103   min_value     INTEGER NOT NULL,
104   max_value     INTEGER NOT NULL,
105   CONSTRAINT ck_stat_def_minmax CHECK (min_value <= max_value),
106   CONSTRAINT uq_stat_def_system_code UNIQUE (system_id, code),
107   CONSTRAINT uq_stat_def_system_name UNIQUE (system_id, name)
108 );
109
110 CREATE TABLE resource_def (
111   resource_id   INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
112   system_id     INTEGER NOT NULL REFERENCES system(system_id) ON DELETE CASCADE,
113   code          VARCHAR(40) NOT NULL,
114   name          VARCHAR(120) NOT NULL,
115   min_value     INTEGER NOT NULL,
116   max_value     INTEGER NOT NULL,
117   CONSTRAINT ck_resource_def_minmax CHECK (min_value <= max_value),
118   CONSTRAINT uq_resource_def_system_code UNIQUE (system_id, code),
119   CONSTRAINT uq_resource_def_system_name UNIQUE (system_id, name)
120 );
```

```

122 CREATE TABLE character_stat (
123     character_id  INTEGER NOT NULL REFERENCES character(character_id) ON DELETE CASCADE,
124     stat_id       INTEGER NOT NULL REFERENCES stat_def(stat_id) ON DELETE RESTRICT,
125     value         INTEGER NOT NULL DEFAULT 0,
126     PRIMARY KEY (character_id, stat_id)
127 );
128
129 CREATE TABLE character_resource (
130     character_id  INTEGER NOT NULL REFERENCES character(character_id) ON DELETE CASCADE,
131     resource_id   INTEGER NOT NULL REFERENCES resource_def(resource_id) ON DELETE RESTRICT,
132     value         INTEGER NOT NULL DEFAULT 0,
133     PRIMARY KEY (character_id, resource_id)
134 );
135 -- CLASS / ABILITY / ITEM
136 CREATE TABLE rpg_class (
137     class_id      INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
138     system_id     INTEGER NOT NULL REFERENCES system(system_id) ON DELETE CASCADE,
139     name          VARCHAR(120) NOT NULL,
140     lore_text     TEXT,
141     CONSTRAINT uq_rpg_class__system_name UNIQUE (system_id, name)
142 );
143
144 ALTER TABLE character
145     ADD CONSTRAINT fk_character_class
146     FOREIGN KEY (class_id) REFERENCES rpg_class(class_id)
147     ON DELETE RESTRICT;
148
149 CREATE TABLE class_stat_modifier (
150     class_id      INTEGER NOT NULL REFERENCES rpg_class(class_id) ON DELETE CASCADE,
151     stat_id       INTEGER NOT NULL REFERENCES stat_def(stat_id) ON DELETE RESTRICT,
152     delta         INTEGER NOT NULL,
153     PRIMARY KEY (class_id, stat_id)
154 );
155
156 CREATE TABLE ability (
157     ability_id    INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
158     system_id     INTEGER NOT NULL REFERENCES system(system_id) ON DELETE CASCADE,
159     name          VARCHAR(160) NOT NULL,
160     rules_text    TEXT,
161     dice_type     VARCHAR(40),
162     CONSTRAINT uq_ability__system_name UNIQUE (system_id, name)
163 );
164
165 CREATE TABLE class_ability (
166     class_id      INTEGER NOT NULL REFERENCES rpg_class(class_id) ON DELETE CASCADE,
167     ability_id    INTEGER NOT NULL REFERENCES ability(ability_id) ON DELETE RESTRICT,
168     sort_order    INTEGER NOT NULL DEFAULT 0 CHECK (sort_order >= 0),
169     PRIMARY KEY (class_id, ability_id)
170 );
171
172 CREATE TABLE character_ability (
173     character_id  INTEGER NOT NULL REFERENCES character(character_id) ON DELETE CASCADE,
174     ability_id    INTEGER NOT NULL REFERENCES ability(ability_id) ON DELETE RESTRICT,
175     source        VARCHAR(60), -- e.g. 'CLASS', 'REWARD', 'ITEM', 'MANUAL'
176     PRIMARY KEY (character_id, ability_id)
177 );
178
179 CREATE TABLE item (
180     item_id       INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
181     system_id     INTEGER NOT NULL REFERENCES system(system_id) ON DELETE CASCADE,
182     name          VARCHAR(160) NOT NULL,
183     description   TEXT,
184     CONSTRAINT uq_item__system_name UNIQUE (system_id, name)
185 );
186
187 CREATE TABLE character_item (
188     character_id  INTEGER NOT NULL REFERENCES character(character_id) ON DELETE CASCADE,
189     item_id       INTEGER NOT NULL REFERENCES item(item_id) ON DELETE RESTRICT,
190     qty           INTEGER NOT NULL DEFAULT 1 CHECK (qty >= 0),
191     PRIMARY KEY (character_id, item_id)
192 );
193
194 CREATE TABLE audit_log (
195     audit_id      BIGINT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
196     table_name    TEXT NOT NULL,
197     action        TEXT NOT NULL,
198     changed_at    TIMESTAMPTZ NOT NULL DEFAULT now(),
199     row_pk        TEXT,
200     old_data      JSONB,
201     new_data      JSONB
202 );
203
204 CREATE INDEX idx_entity_universe_kind ON entity(universe_id, kind_code);
205 CREATE INDEX idx_character_campaign ON character(campaign_id);
206 CREATE INDEX idx_stat_def_system ON stat_def(system_id);
207 CREATE INDEX idx_resource_def_system ON resource_def(system_id);
208
209 COMMIT;
210

```

c.2.İlgili “02_seed.sql” dosyasının içerisindeki SQL ifadeleri:

```
1  BEGIN;
2
3  INSERT INTO entity_kind(kind_code, description) VALUES
4  ('CHARACTER', 'Player Character (PC)'),
5  ('NPC', 'Non-player character'),
6  ('LOCATION', 'Location / place'),
7  ('FACTION', 'Faction / organization')
8  ON CONFLICT (kind_code) DO NOTHING;
9
10 -- Systems
11 INSERT INTO system(name) VALUES
12 ('Umbra Caelis'),
13 ('Wolfenstein')
14 ON CONFLICT (name) DO NOTHING;
15
16 -- Universe
17 INSERT INTO universe(system_id, name, description)
18 SELECT s.system_id, 'Umbra Caelis Core', 'Main setting'
19 FROM system s
20 WHERE s.name = 'Umbra Caelis'
21 ON CONFLICT (system_id, name) DO NOTHING;
22
23 -- Campaign (basit)
24 INSERT INTO campaign(universe_id, name, status)
25 SELECT u.universe_id, 'Campaign 1', 'ACTIVE'
26 FROM universe u
27 WHERE u.name = 'Umbra Caelis Core'
28 ON CONFLICT (universe_id, name) DO NOTHING;
29
30 INSERT INTO stat_def(system_id, code, name, min_value, max_value)
31 SELECT s.system_id, v.code, v.name, 0, 6
32 FROM system s
33 JOIN (VALUES
34 ('HAK','Hakimiyet'),
35 ('SEB','Sebahat'),
36 ('CAZ','Cazibe'),
37 ('HIK','Hikmet'),
38 ('GOR','Görü'),
39 ('MAR','Marifet'),
40 ('YAL','Yalman'),
41 ('EFS','Efsun'),
42 ('KAC','Kaçan')
43 ) v(code,name) ON true
44 WHERE s.name='Umbra Caelis'
45 ON CONFLICT (system_id, code) DO NOTHING;
46
47 -- Umbra Caelis - resources (example 0..
48 INSERT INTO resource_def(system_id, code, name, min_value, max_value)
49 SELECT s.system_id, v.code, v.name, 0, 6
50 FROM system s
51 JOIN (VALUES
52 ('HP','Sağlık'),
53 ('ESIN','Esin Puanı'),
54 ('LUM','Lumre')
55 ) v(code,name) ON true
56 WHERE s.name='Umbra Caelis'
57 ON CONFLICT (system_id, code) DO NOTHING;
58
59 -- Sample class: Fallen Noble
60 INSERT INTO rpg_class(system_id, name, lore_text)
61 SELECT s.system_id, 'Fallen Noble', 'Preset example'
62 FROM system s
63 WHERE s.name='Umbra Caelis'
64 ON CONFLICT (system_id, name) DO NOTHING;
```

```

67  INSERT INTO rpg_class(system_id, name, lore_text)
68  SELECT s.system_id, v.name, v.lore
69  FROM system s
70  JOIN (VALUES
71  ('Arcanist', 'Magic-focused caster.'),
72  ('Warden', 'Defensive / tank archetype.'),
73  ('Vagabond', 'Mobile / skill-oriented archetype.')
74 ) v(name, lore) ON true
75  WHERE s.name='Umbra Caelis'
76  ON CONFLICT (system_id, name) DO NOTHING;
77
78
79  -- Arcanist class
80  INSERT INTO class_stat_modifier(class_id, stat_id, delta)
81  SELECT rc.class_id, sd.stat_id, v.delta
82  FROM rpg_class rc
83  JOIN system s ON s.system_id = rc.system_id
84  JOIN stat_def sd ON sd.system_id = s.system_id
85  JOIN (VALUES
86  ('EFS', 2),
87  ('HIK', 1),
88  ('YAL', -1)
89 ) v(code, delta) ON v.code = sd.code
90  WHERE s.name='Umbra Caelis' AND rc.name='Arcanist'
91  ON CONFLICT (class_id, stat_id) DO UPDATE SET delta = EXCLUDED.delta;
92
93  -- Warden class
94  INSERT INTO class_stat_modifier(class_id, stat_id, delta)
95  SELECT rc.class_id, sd.stat_id, v.delta
96  FROM rpg_class rc
97  JOIN system s ON s.system_id = rc.system_id
98  JOIN stat_def sd ON sd.system_id = s.system_id
99  JOIN (VALUES
100 ('HAK', 2),
101 ('MAR', 1),
102 ('CAZ', -1)
103 ) v(code, delta) ON v.code = sd.code
104 WHERE s.name='Umbra Caelis' AND rc.name='Warden'
105 ON CONFLICT (class_id, stat_id) DO UPDATE SET delta = EXCLUDED.delta;
106
107  -- Vagabond
108  INSERT INTO class_stat_modifier(class_id, stat_id, delta)
109  SELECT rc.class_id, sd.stat_id, v.delta
110  FROM rpg_class rc
111  JOIN system s ON s.system_id = rc.system_id
112  JOIN stat_def sd ON sd.system_id = s.system_id
113  JOIN (VALUES
114  ('SEB', 1),
115  ('GOR', 1),
116  ('HAK', -1)
117 ) v(code, delta) ON v.code = sd.code
118 WHERE s.name='Umbra Caelis' AND rc.name='Vagabond'
119 ON CONFLICT (class_id, stat_id) DO UPDATE SET delta = EXCLUDED.delta;
120
121  -- CAZ
122  INSERT INTO class_stat_modifier(class_id, stat_id, delta)
123  SELECT rc.class_id, sd.stat_id, v.delta
124  FROM rpg_class rc
125  JOIN system s ON s.system_id = rc.system_id
126  JOIN stat_def sd ON sd.system_id = s.system_id
127  JOIN (VALUES
128  ('CAZ', 2),
129  ('HAK', 1),
130  ('SEB', 1),
131  ('KAC', -1)
132 ) v(code, delta) ON v.code = sd.code
133 WHERE s.name='Umbra Caelis' AND rc.name='Fallen Noble'
134 ON CONFLICT (class_id, stat_id) DO UPDATE SET delta = EXCLUDED.delta;
135
136  -- Sample ability + mapping (optional, but useful)
137  INSERT INTO ability(system_id, name, rules_text, dice_type)
138  SELECT s.system_id, 'Noble Bearing', 'Sample ability granted by class.', 'd6'
139  FROM system s
140  WHERE s.name='Umbra Caelis'
141  ON CONFLICT (system_id, name) DO NOTHING;
142
143  INSERT INTO class_ability(class_id, ability_id, sort_order)
144  SELECT rc.class_id, a.ability_id, 1
145  FROM rpg_class rc
146  JOIN system s ON s.system_id = rc.system_id
147  JOIN ability a ON a.system_id = s.system_id
148  WHERE s.name='Umbra Caelis'
149  AND rc.name='Fallen Noble'
150  AND a.name='Noble Bearing'
151  ON CONFLICT (class_id, ability_id) DO NOTHING;
152
153  -- Sample item
154  INSERT INTO item(system_id, name, description)
155  SELECT s.system_id, 'Rope', 'Basic item'
156  FROM system s
157  WHERE s.name='Umbra Caelis'
158  ON CONFLICT (system_id, name) DO NOTHING;
159
160  COMMIT;
161
```

d. 4 Adet Saklı yordam ve 4 adet tetikleyici madde örneği ve işlevleri

```
1 SELECT * FROM fn_search_characters(1, 'Test');
2 SELECT * FROM fn_search_characters(1, NULL);
```

character_id	character_name	class_name	level
1	Test Karakter	Fallen Noble	0

d.1.“fn_search_characters” Fonksiyonu (Arama İşlevi)

Amaç: Kampanya içerisindeki karakterleri ada göre aramak ve listelemek.

Yapılan İşlem: fn_search_characters(p_campaign_id, p_name_like) fonksiyonu çağrılmıştır.

Fonksiyon, verilen kampanya ID'si içinde karakter adını ILIKE operatörü ile aramaktadır.

Arama parametresi NULL gönderildiğinde kampanyadaki tüm karakterler listelenmektedir.

Sonuç: Fonksiyon başarıyla çalışmış ve 'Test' parametresi ile yalnızca ilgili karakter, NULL parametresi ile kampanyadaki tüm karakterler doğru şekilde döndürülmüştür.

```
1 UPDATE character_stat
2 SET value = 6
3 WHERE character_id = 1
4 AND stat_id = (SELECT stat_id FROM stat_def ORDER BY stat_id LIMIT 1);
5
6 SELECT sd.code, sd.name, cs.value
7 FROM character_stat cs
8 JOIN stat_def sd ON sd.stat_id = cs.stat_id
9 WHERE cs.character_id = 1
10 ORDER BY sd.code;
```

code	name	value
CAZ	Cazibe	2
GOR	Görü	0
HAK	Hakimiyet	6
HIK	Hikmet	0
SEB	Sebahat	1

d.2.Stat Min/Max Kontrol Trigger’ı (t_check_stat_bounds)

Amaç: Karakter stat değerlerinin, sistemde tanımlanan minimum ve maksimum sınırlar之外 güncellenmesini engellemek.

Yapılan İşlem: Bir karakterin stat değeri üst sınır olan 6 değerine ayarlanmıştır. Trigger, bu güncellemeye kontrol ederek kurala uygunluğunu doğrulamıştır.

Sonuç: Güncelleme işlemi başarıyla gerçekleşmiş ve stat değeri sınırlar içinde kalmıştır.

Trigger’ın doğru şekilde çalıştığı doğrulanmıştır.

```

1 UPDATE character_resource
2 SET value = 6
3 WHERE character_id = 1
4 AND resource_id = (SELECT resource_id FROM resource_def ORDER BY resource_id LIMIT 1);
5
6 SELECT rd.code, rd.name, cr.value
7 FROM character_resource cr
8 JOIN resource_def rd ON rd.resource_id = cr.resource_id
9 WHERE cr.character_id = 1
10 ORDER BY rd.code;
11

```

Data Output Messages Notifications

Showing rows: 1 to 3

	code	name	value
1	ESIN	Esin Puanı	0
2	HP	Sağlık	6
3	LUM	Lumre	0

d.3.Resource Min/Max Kontrol Trigger'ı (t_check_resource_bounds)

Amaç: Karakter kaynak (resource) değerlerinin tanımlı aralık dışına çıkışmasını engellemek.

Yapılan İşlem: Karaktere ait bir resource değeri maksimum sınır olan 6 olarak güncellenmiştir.

Sonuç: Güncelleme başarıyla tamamlanmış, kaynak değerleri sınırlar içinde tutulmuştur.

Trigger doğru şekilde çalışmaktadır.

```

1 SELECT e.name AS character_name, i.name AS item_name, ci.qty
2 FROM character_item ci
3 JOIN character ch ON ch.character_id = ci.character_id
4 JOIN entity e ON e.entity_id = ch.entity_id
5 JOIN item i ON i.item_id = ci.item_id
6 WHERE ci.character_id = 2;
7

```

Data Output Messages Notifications

Showing rows: 1 to 1

character_name	item_name	qty
karakter adı (160)	eşya adı (160)	integer (160)

d.4.Envanter Temizleme Trigger'ı (t_item_qty_cleanup)

Amaç: Bir karakterin envanterindeki eşya miktarı 0 olduğunda, ilgili satırın otomatik olarak silinmesini sağlamak.

Yapılan İşlem: Bir karakterin sahip olduğu eşyanın miktarı 0 olacak şekilde güncellenmiştir.

Sonuç: Trigger devreye girerek ilgili eşya satırını otomatik olarak silmiştir. Envanterde qty = 0 olan kayıt kalmamıştır.

```

1 SELECT audit_id, table_name, action, changed_at, old_data, new_data
2 FROM audit_log
3 WHERE table_name = 'character'
4 ORDER BY audit_id DESC
5 LIMIT 5;
6

```

	audit_id	table_name	action	changed_at	old_data
1	9	character	UPDATE	2025-12-12 13:41:41.218721+...	{"level": 0, "story": null, "class_id": 1, "entity_id": 1, "strengths": null, "weaknesses": null, "campaign_id": 1, "character_i...}
2	8	character	INSERT	2025-12-12 13:39:26.682895+...	[null]
3	7	character	INSERT	2025-12-12 13:39:17.098146+...	[null]
4	6	character	INSERT	2025-12-12 13:38:34.216692+...	[null]
5	3	character	UPDATE	2025-12-12 13:34:29.536851+...	{"level": 0, "story": null, "class_id": 1, "entity_id": 1, "strengths": null, "weaknesses": null, "campaign_id": 1, "character_i..."}

	new_data
1	{"level": 1, "story": null, "class_id": 1, "entity_id": 1, "strengths": null, "weaknesses": null, "campaign_id": 1, "character_i..."}
2	{"level": 0, "story": null, "class_id": 1, "entity_id": 6, "strengths": null, "weaknesses": null, "campaign_id": 1, "character_i..."}
3	{"level": 0, "story": null, "class_id": 1, "entity_id": 5, "strengths": null, "weaknesses": null, "campaign_id": 1, "character_i..."}
4	{"level": 0, "story": null, "class_id": 1, "entity_id": 4, "strengths": null, "weaknesses": null, "campaign_id": 1, "character_i..."}
5	{"level": 0, "story": null, "class_id": 1, "entity_id": 1, "strengths": null, "weaknesses": null, "campaign_id": 1, "character_i..."}

d.5.Audit Log Trigger'i (t_audit_character)

Amaç: character tablosunda gerçekleşen INSERT ve UPDATE işlemlerinin otomatik olarak kayıt altına alınması.

Yapılan İşlem: Bir karakterin level alanı güncellenmiştir. Bu işlem sonrası audit_log tablosu sorgulanmıştır.

Sonuç:

Audit log tablosunda:

İşlem türü (INSERT / UPDATE)

İşlem zamanı

Eski ve yeni veri (JSON formatında)

bilgileri doğru şekilde kaydedilmiştir.

The screenshot shows a SQL query editor interface. At the top, there are tabs for 'Query' and 'Query History'. Below the tabs, the query text is displayed:

```
1 SELECT sp_create_character(
2     1, -- campaign_id
3     4, -- class_id
4     'Test Character',
5     'Kısa açıklama'
6 ) AS new_character_id;
```

Below the query text, there are three tabs: 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is selected. It displays the results of the query in a table format:

	new_character_id	integer
1	46	

At the bottom of the interface, there are several icons for file operations (New, Open, Save, Print, etc.) and a 'SQL' button.

d.6.”sp_create_character” fonksiyonu ile karakter oluşturma

Amaç: Yeni bir karakterin, seçilen kampanya ve class bilgilerine göre tüm bağımlı verileriyle birlikte (entity, stat, resource, ability) tek bir işlem olarak oluşturulmasını sağlamak.

Yaptığı İşlem: Karakter için entity tablosunda kayıt oluşturur, character tablosuna yeni karakteri ekler. İlgili oyun sistemine ait tüm statları otomatik oluşturur ve Kaynak (resource) değerlerini minimum değerleriyle ekler. Class'a ait stat modifier'ları uygular ve son olarak Class'a ait yetenekleri karaktere otomatik tanımlar

Sonuç: Karakter oluşturma işlemi, uygulama katmanından bağımsız olarak tamamen veritabanı seviyesinde ve atomik biçimde gerçekleştirilmiştir.

İşlem sırasında veri bütünlüğü bozulmamaktadır.

```

Query 1:
1 SELECT ch.character_id, e.name, ch.class_id
2 FROM character ch
3 JOIN entity e ON e.entity_id = ch.entity_id
4 ORDER BY ch.character_id;
5
6 SELECT class_id, name FROM rpg_class ORDER BY class_id;
7
8
9 CALL sp_apply_class_to_character(
10    1, -- p_character_id
11    1 -- p_new_class_id
12 );
13 SELECT * FROM fn_character_sheet(1);
14
15

Query 2:
1 SELECT ch.character_id, e.name, ch.class_id
2 FROM character ch
3 JOIN entity e ON e.entity_id = ch.entity_id
4 ORDER BY ch.character_id;
5
6 SELECT class_id, name FROM rpg_class ORDER BY class_id;
7
8
9 CALL sp_apply_class_to_character(
10    1, -- p_character_id
11    5 -- p_new_class_id
12 );
13 SELECT * FROM fn_character_sheet(1);
14
15

```

Data Output (Left):

character_id	character_name	campaign_name	class_name	level	stat_code	stat_name	stat_value
1	Test Karakter	Campaign 1	Fallen Noble	1	CAZ	Cazibe	5
2	Test Karakter	Campaign 1	Fallen Noble	1	GOR	Görü	3
3	Test Karakter	Campaign 1	Fallen Noble	1	HAK	Hakimiyet	4
4	Test Karakter	Campaign 1	Fallen Noble	1	HIK	Hikmet	3
5	Test Karakter	Campaign 1	Fallen Noble	1	SEB	Sebahat	4

Data Output (Right):

character_id	character_name	campaign_name	class_name	level	stat_code	stat_name	stat_value
1	Test Karakter	Campaign 1	Warden	1	CAZ	Cazibe	3
2	Test Karakter	Campaign 1	Warden	1	GOR	Görü	3
3	Test Karakter	Campaign 1	Warden	1	HAK	Hakimiyet	3
4	Test Karakter	Campaign 1	Warden	1	HIK	Hikmet	3
5	Test Karakter	Campaign 1	Warden	1	SEB	Sebahat	3

d.7.”sp_apply_class_to_character” fonksiyonu

Amaç: Mevcut bir karakterin class bilgisini değiştirirken, stat değerlerinin yeni class'a uygun şekilde yeniden hesaplanması sağlamak.

Yaptığı İşlem: Karakterin class_id bilgisini günceller, Karakterin tüm statlarını baz değere (BASE 3) resetler. Yeni class'a ait stat modifier'ları yeniden uygular ve Yeni class'in yeteneklerini karaktere ekler

Sonuç: Class değişikliği sonrası statlar hatalı veya çıkışmalı hale gelmemekte, tüm hesaplamlar merkezi olarak saklı yordam aracılığıyla yapılmaktadır.

```

Query 1:
1 SELECT *
2 FROM fn_character_inventory(41);
3

Query 2:
1 SELECT *
2 FROM fn_character_inventory(41);
3
4 CALL sp_add_item_to_character(
5    41, -- p_character_id
6    1, -- p_item_id
7    2 -- p_qty
8 );
9

Query 3:
1 CALL
2
3 Query returned successfully in 56 msec.

Data Output (Left):
Showing rows: 1 to 1 | Page No: 1 of 1 | < << >> >
character_id integer
character_name text
item_name text
qty integer
1 41 Hasan Rope 2

Data Output (Middle):
Showing rows: 1 to 1 | Page No: 1 of 1 | < << >> >
character_id integer
character_name text
item_name text
qty integer
1 41 Hasan Rope 26

Data Output (Right):
Showing rows: 1 to 1 | Page No: 1 of 1 | < << >> >
character_id integer
character_name text
item_name text
qty integer
1 41 Hasan Rope 26

```

d.8.”sp_transfer_item” fonksiyonu

Amaç: Bir karakterin envanterindeki eşyaların, başka bir karaktere kontrollü ve güvenli şekilde transfer edilmesini sağlamak.

Yaptığı İşlem: Gönderen karakterin yeterli eşya adedine sahip olup olmadığını kontrol eder, eşya adedi yetersizse işlemi hata ile iptal eder. Gönderen karakterin envanterinden düşürür ve alıcı karakterin envanterine ekler. İşlem sırasında qty = 0 durumuna düşen kayıtlar trigger tarafından temizlenir.

Sonuç: Envanter transfer işlemleri, uygulama seviyesinde kontrol gerektirmeden tamamen veritabanı seviyesinde güvenli şekilde yürütülmektedir.

e. Arama, Ekleme, Silme ve Güncelleme İşlemleri

Veri tabanının başlangıçtaki hali aşağıdaki sekildedir:

Karakterler

- Karakter silindi.

Campaign: Ara:

Yeni Karakter

Campaign: Class: İsim: Açıklama:

Liste

ID	Name	Class	Level	Detay
41	Hasan	Vagabond	0	Aç
7	Test	Vagabond	2	Aç
1	Test Karakter	Fallen Noble	1	Aç
4	Test Karakter 2	Fallen Noble	0	Aç

e.1. Arama İşlemi

Karakterler

Campaign: Ara:

Yeni Karakter

Campaign: Class: İsim: Açıklama:

Liste

ID	Name	Class	Level	Detay
41	Hasan	Vagabond	0	Aç

Kırmızı ile işaretlenmiş kutucuğa aranacak karakterin adı yazılır ve “Search” butonuna basılarak arama yapılır. Aratılan kriterde uygun karakter(ler) listelenir.

e.2. Ekleme İşlemi

Karakterler

Campaign: 1 - Campaign 1 Ara: Search

Yeni Karakter

Campaign: Campaign 1 Class: Warden İsim: Mehmet Açıklama: Yeni karakter Oluştur

Liste

ID	Name	Class	Level	Detay
41	Hasan	Vagabond	0	Aç
7	Test	Vagabond	2	Aç
1	Test Karakter	Fallen Noble	1	Aç
4	Test Karakter 2	Fallen Noble	0	Aç

Yukarıda yer alan “Yeni Karakter” başlığının altındaki alan kullanılarak yeni oluşturulacak karaktere ait temel bilgiler girilir ve “Oluştur” tuşu ile yeni karakter oluşturulur.

Karakterler

Campaign: 1 - Campaign 1 Ara: Search

Yeni Karakter

Campaign: Campaign 1 Class: Arcanist İsim: Açıklama: Oluştur

Liste

ID	Name	Class	Level	Detay
41	Hasan	Vagabond	0	Aç
42	Mehmet	Warden	0	Aç
7	Test	Vagabond	2	Aç
1	Test Karakter	Fallen Noble	1	Aç
4	Test Karakter 2	Fallen Noble	0	Aç

Yeni oluşturulan karakter Liste’nin altında listelenir.

e.3. Silme İşlemi

Karakterin detay penceresi “Detay” başlığı altındaki “Aç” tuşu ile açılır ve detay penceresinin altındaki “Karakter Sil” tuşu ile karakter silinir. Aşağıda buna örnek olarak ID numarası 7 olan “Test” karakteri silinmiştir:

Test

Campaign: Campaign 1 | Class: Vagabond | Level: 2

Level Güncelle

[Güncelle](#)

Statlar

Code	Name	Value
CAZ	Cazibe	3
EFS	Efsun	3
GOR	Görüş	3
HAK	Hakimiyet	3
HIK	Hikmet	3
KAC	Kaçma	3
MAR	Marifet	3
SEB	Şehabat	3
VAL	Yalman	3

Class Değiştir

[Vagabond](#) [Uygula](#)

Envanter

[Item Qty](#)

Item Transfer

From: To: Item: Qty: Transfer

Sil

[Karakter Sil](#)

Karakterler

- Karakter silindi.

Campaign: 1 - Campaign 1 Ara: Search

Yeni Karakter

Campaign: Campaign 1 Class: Arcanist İsim: Açıklama: Oluştur

Liste

ID	Name	Class	Level	Detay
41	Hasan	Vagabond	0	Aç
42	Mehmet	Warden	0	Aç
1	Test Karakter	Fallen Noble	1	Aç
4	Test Karakter 2	Fallen Noble	0	Aç

e.4. Güncelleme İşlemi

Güncelleme işlemine örnek olarak ID numarası 1 olan karakterin Class bilgisini güncelleyeceğiz. Öncelikle karakterin detay penceresini açıyoruz.

Karakterler

- Karakter silindi.

Campaign: 1 - Campaign 1 Ara: Search

Yeni Karakter

Campaign: Campaign 1 Class: Arcanist İsim: Açıklama: Oluştur

Liste

ID	Name	Class	Level	Detay
41	Hasan	Vagabond	0	Aç
42	Mehmet	Warden	0	Aç
1	Test Karakter	Fallen Noble	1	Aç
4	Test Karakter 2	Fallen Noble	0	Aç

Detay penceresindeki “Class Değiştir” kısmındaki ComboBox ile ilgili karakterin Class bilgisi değiştirilir:

Test Karakter

Campaign: Campaign 1 | Class: Fallen Noble | Level: 1

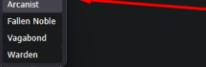
Level Güncelle

1

Statlar

Code	Name	Value
CAZ	Cazibe	2
GOR	Görü	0
HAK	Hakimiyet	6
HIK	Hikmet	0
SEB	Sebahat	1

Class Değiştir

Fallen Noble 

Arcanist
Fallen Noble
Vagabond
Warden
Rope 6

Item Transfer

From: 1 - Test Karakter To: 1 - Test Karakter Item: Rope Qty: 1 Transfer

Sil

Class bilgisi değiştirilen 1 ID numaralı karakterin Class bilgisi başarılı bir şekilde değiştirilmiş olduğu karakter listesinden net bir şekilde görüntülenebilir:

Karakterler

Campaign: 1 - Campaign 1 Ara: Search

Yeni Karakter

Campaign: Campaign 1 Class: Arcanist İsim: Açıklama: Oluştur

Liste

ID	Name	Class	Level	Detay
41	Hasan	Vagabond	0	Aç
42	Mehmet	Warden	0	Aç
1	Test Karakter	Arcanist	1	Aç
4	Test Karakter 2	Fallen Noble	0	Aç

f. Uygulamanın Kaynak Kodları

Uygulamanın kaynak kodlarına kapak sayfasında verilmiş olan GitHub reposunda (<https://github.com/kuyash71/ttrpgDatabase>) yer almaktadır. Aynı şekilde projenin SQL kodları ve ER diyagramı da GitHub reposunda yer almaktadır.

Sonuç:

Proje sonucunda, TTRPG oyunları için ölçülebilir ve yeniden kullanılabilir bir veritabanı mimarisi başarıyla geliştirilmiştir. Tasarlanan sistem; karakter oluşturma, sınıf bazlı istatistik uygulama, envanter yönetimi, arama işlemleri ve değişiklik kayıtlarının tutulması gibi temel oyun mekaniklerini desteklemektedir.

Stored procedure'ler kullanılarak karmaşık iş kuralları (örneğin sınıf seçimine bağlı otomatik stat atamaları ve eşya transferleri) merkezi bir yapıda toplanmış, trigger'lar ile istatistik ve kaynak sınırları denetlenmiş ve audit mekanizması sayesinde veri değişiklikleri kayıt altına alınmıştır. Bu sayede sistem hem güvenli hem de hataya dayanıklı hale getirilmiştir.

Geliştirilen veritabanı, ileride bir web veya masaüstü uygulaması ile entegre edilebilecek altyapıya sahiptir. Bu yönyle proje, yalnızca bir akademik çalışma değil, aynı zamanda gerçek hayatı kullanılabılır bir TTRPG yönetim sistemi altyapısı sunmaktadır.