YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

BLM3041 - VERİ TABANI YÖNETİMİ

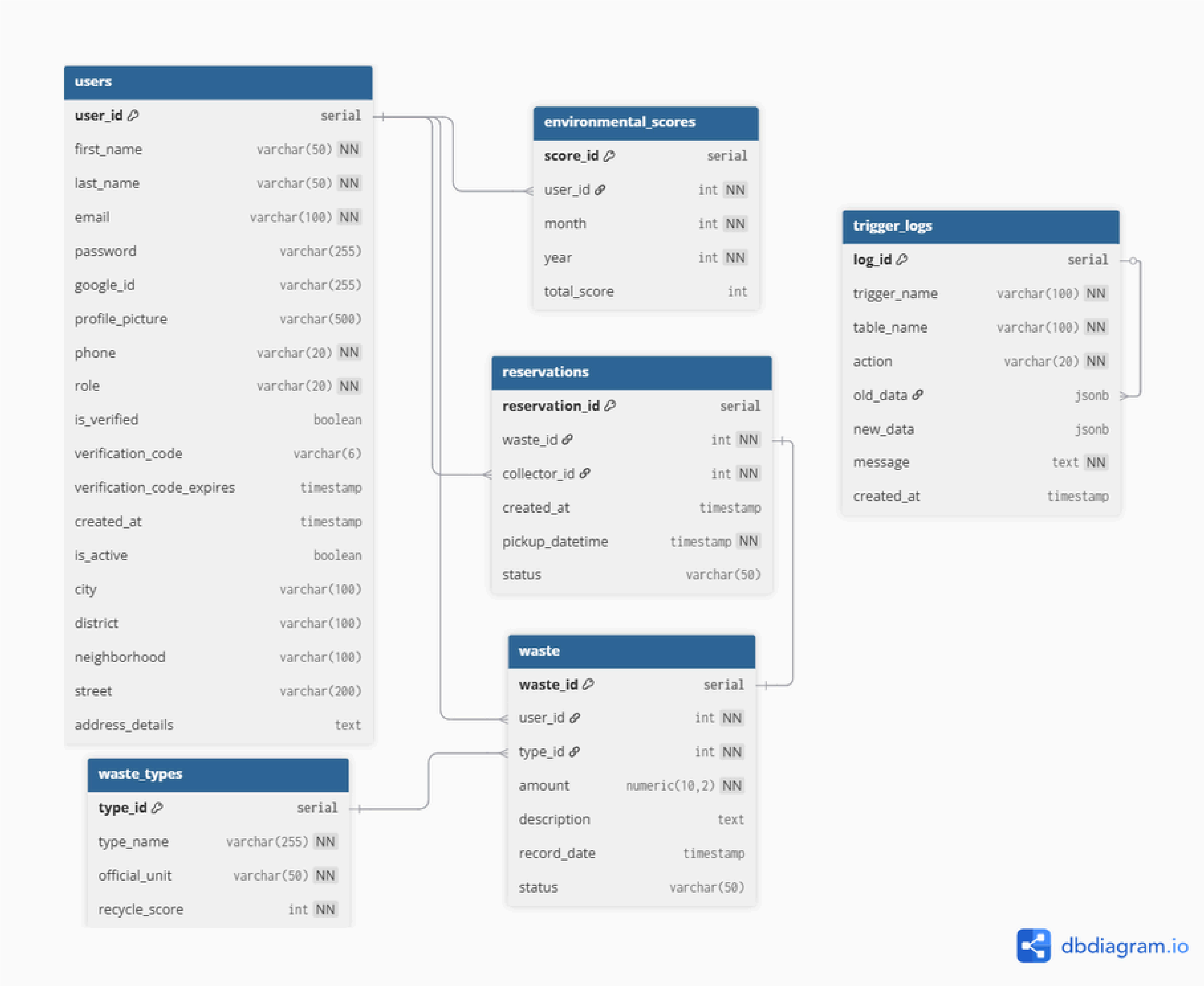
Mahalle Atık Paylaşım & Geri Dönüşüm Takas Platformu (RecycleShare)

Fatma Bera ÇOLAK - 23011111

Ayşe Sude CAMİ - 22011064

Sümeyye YOLDAŞ - 24023068

Tablolar


users
user_id serial
first_name varchar(50) NN
last_name varchar(50) NN
email varchar(100) NN
password varchar(255)
google_id varchar(255)
profile_picture varchar(500)
phone varchar(20) NN
role varchar(20) NN
is_verified boolean
verification_code varchar(6)
verification_code_expires timestamp
created_at timestamp
is_active boolean
city varchar(100)
district varchar(100)
neighborhood varchar(100)
street varchar(200)
address_details text
environmental_scores
score_id serial
user_id int NN
month int NN
year int NN
total_score int
trigger_logs
log_id serial
trigger_name varchar(100) NN
table_name varchar(100) NN
action varchar(20) NN
old_data jsonb
new_data jsonb
message text NN
created_at timestamp
reservations
reservation_id serial
waste_id int NN
collector_id int NN
created_at timestamp
pickup_datetime timestamp NN
status varchar(50)
waste
waste_id serial
user_id int NN
type_id int NN
amount numeric(10,2) NN
description text
record_date timestamp
status varchar(50)
waste_types
type_id serial
type_name varchar(255) NN
official_unit varchar(50) NN
recycle_score int NN
dbdiagram.io

Users Tablosu

Sistemde aktif kullanıcıları ve yetkilerini yönetir. Sadece doğrulama sürecini başarıyla tamamlamış üyelerin kalıcı bilgilerini saklar.

- **user_id** ile kullanıcı için sistem tarafından atanan benzersiz kullanıcı numarası tutulur. Diğer tüm tablolar bu ID üzerinden kullanıcıyla ilişkilendirilir.
- Kullanıcının profil bilgileri **first_name**, **last_name**, **email**, **phone** sütunları ile tutulur.
- **google_id** kullanıcının mevcut Google hesabı ile sisteme hızlıca entegre olmasını sağlar. Google ile "Devam Et" seçeneği kullanıldığında gelen benzersiz kimlik bilgisidir.
- **profile_picture** ise kullanıcının Google profilindeki fotoğrafını sisteme çekerek kişiselleştirilmiş bir deneyim sunar.
- Kullanıcı yetki seviyesini **role** sütunu ile tutulur. Bu sütun resident veya admin değerlerini alabilir. Varsayılan yetki seviyesi resident olarak tanımlanmıştır.
- Kullanıcının şifre bilgisi(**password**), hesabının doğrulanıp doğrulanmadığını (**is_verified**) ve hesabın dondurulup dondurulmadığını (**is_active**) kontrol eder.
- **verification_code** ve **verification_code_expires** şifre yenileme veya hesap güncelleme gibi durumlarda güvenlik teyidi için kullanılan geçici kod ve süresidir.
- **created_at** kullanıcının sisteme tam olarak ne zaman dahil olduğunu belirler.
- Kullanıcıya ait il, ilçe, semt, sokak gibi konum bilgileri **city, district, neighborhood, street, address_details** sütunlarına kaydedilir. Atık toplama işlemlerinin doğru adrese yapılabilmesi için gereken detaylı verilerdir.

Veriler (14 kayıt) 1 / 1

user_id	first_name	last_name	email	password	google_id	prof
38	Vera	Doe	fberacolak@gmail.com	$2b$12$tAOBvSCNmvW...	1049369041920327930...	http
12	Bera	Çolak	bera213342@gmail.com	$2b$12$TGuEA4vfHOz....	1036433669872857478...	http
2	Ayşe Sude	Cami	asudecmi.98@gmail.com	NULL	1085620905858059294...	http
37	Sümeyye	Yoldaş	sumeyyeyoldas3@gmail...	$2b$12$qd.8lc18V2lybY...	104150523006835820566	NU
13	Ahmet	Yılmaz	ahmet@test.com	$2b$12$axKyp9/fSu4In...	NULL	NU
14	Ayşe	Kaya	ayse@test.com	$2b$12$yabcGr9e8eLm...	NULL	NU
15	Mehmet	Demir	mehmet@test.com	$2b$12$yabcGr9e8eLm...	NULL	NU
16	Fatma	Şahin	fatma@test.com	$2b$12$yabcGr9e8eLm...	NULL	NU
18	Zeynep	Aydın	zeynep@test.com	$2b$12$yabcGr9e8eLm...	NULL	NU
19	Mustafa	Çelik	mustafa@test.com	$2b$12$yabcGr9e8eLm...	NULL	NU

Waste_types Tablosu

Sistemde takip edilen tüm atık türlerini ve bu türlerin geri dönüşüm değerlerini tanımlar. Atık kayıtları oluşturulurken bu tablodaki standart veriler referans alınır.

- **type_id** ile her atık türü için sistem tarafından atanan benzersiz bir atık numarası tutulur. Atık kayıtları yapılırken hangi türün seçildiği bu ID üzerinden ilişkilendirilir.
- **type_name** sütunu ile atığın cinsi (Örn: Plastik, Kağıt, Metal, Cam) kaydedilir. Kullanıcılar atık girişi yaparken bu isimleri liste olarak görürler.
- **official_unit** alanı, atığın hangi birimle ölçüleceğini belirler (Örn: kg, litre, adet). Bu, sistemdeki ölçüm birliği ve doğru raporlama için kritiktir.
- **recycle_score** ise ilgili atık türünün birim başına kazandırdığı çevresel puan değerini tutar. Kullanıcının kazandığı toplam puanlar hesaplanırken bu katsayı kullanılır.

Veriler (10 kayıt) 1 / 1

type_id	type_name	official_unit	recycle_score
9	Cables & Chargers	kg	180
10	Recyclable Textiles	kg	50
11	Glass Bottles & Jars	liter	30
1	Cardboard Boxes & Packaging	kg	40
2	Old Books & Newspapers	kg	60
3	PET Bottles	liter	100
4	Hard Plastic Packaging	kg	80
5	Metal Beverage Cans	kg	120
6	Kitchen Metal Waste	kg	150
7	Small Household Appliances	unit	250

Waste Tablosu

Sistemdeki kullanıcıların beyan ettiği atıkların detaylarını ve güncel durumlarını takip eder. Bu tablo, kullanıcı ile fiziksel atık arasındaki dijital bağı kurar.

- Her atık kaydı için sistem tarafından **waste_id** sütununda benzersiz bir takip numarası verilir. Bir atığın tüm süreçleri (oluşturulması, randevuya bağlanması ve toplanması) bu ID üzerinden izlenir.
- **user_id** atığı sisteme giren kullanıcıyı tanımlar. Bu alan users tablosuyla ilişkilidir bu tabloda foreign key görevindedir.
- Kullanıcılar tarafından girilen atığın türü **type_id** sütununa kaydedilir. waste_types tablosuna bağlıdır. Bu sayede atığın türü ve birim puanı bu ID üzerinden sisteme çekilir.
- **amount** kullanıcının beyan ettiği atık miktarını (kg, adet vb.) tutar. Puan hesaplamalarının doğrudan kullanılır.
- **description** kullanıcının atıkla ilgili eklemek istediği notlar için kullanılır.
- Atığın sisteme ilk girildiği tarih ve saat **record_date** sütununda otomatik olarak tutulur.
- **status** atığın o anki aşamasını gösterir. İş planında atıklar; 'beklemede' (waiting), 'randevusu alınmış' (reserved), 'toplanmış' (collected) veya 'iptal edilmiş' (cancelled) olarak bu sütun üzerinden yönetilir.

Veriler (21 kayıt) 1 / 2

waste_id	user_id	type_id	amount	description	record_date	status
1	4	1	5.50	Mixed packaging	2026-01-02T21:27:21.643Z	waiting
6	15	5	1.50	Eski piller ve aküler	2026-01-04T13:22:18.289Z	reserved
16	15	5	1.50	Eski piller ve aküler	2026-01-04T13:22:34.253Z	reserved
22	2	1	5.50	Temiz karton kutular, ürün a...	2026-01-04T13:23:18.089Z	waiting
25	14	4	10.00	PET şişeler ve plastik kaplar	2026-01-04T13:23:18.299Z	waiting
26	15	5	1.50	Eski piller ve aküler	2026-01-04T13:23:18.350Z	reserved
27	2	1	8.00	Gazete ve dergiler	2026-01-04T13:23:18.402Z	waiting
30	14	4	6.50	Plastik oyuncaklar	2026-01-04T13:23:18.621Z	waiting
43	12	9	1.00	USB kablo, adaptör, kulaklık k...	2026-01-08T07:58:26.376Z	waiting
36	37	9	2.00	NULL	2026-01-04T19:56:43.784Z	collected

Reservations Tablosu

Kullanıcıların atık girişlerini ve toplayıcıların atık rezervasyon süreçlerini yönetir.

- Her toplama randevusu için sistem tarafından atanan **reservation_id** benzersiz rezervasyon numarasıdır. Süreçlerin yönetimi ve denetiminde temel referans olarak kullanılır.
- Hangi atığın toplanacağını belirlemek için **waste_id** sütunu tutulur. Bu sütun waste tablosuna bağlanır. Bu sayede atık toplama aracının hangi tür ve miktarda atığı teslim alacağı önceden bilinir.
- **collector_id** atığı teslim alacak olan toplayıcıyı tanımlar. user tablosundaki user_id ile ilişkilendirilir.
- **created_at** rezervasyon kaydının sisteme girildiği anı otomatik olarak kaydeder.
- Atığın adresten alınması için planlanan tarih ve saati **pickup_datetime** ifade eder.
- **status** rezervasyonun güncel durumunu takip eder. Randevular; 'beklemede' (waiting), 'onaylandı' (reserved), 'toplandı' (collected) veya 'iptal edildi' (cancelled) gibi belirli değerler üzerinden yönetilir.
- Randevu zamanı (**pickup_datetime**) geldiğinde ve atık alındığında, hem rezervasyon tablosundaki hem de bağlı olan waste tablosundaki **status** eş zamanlı olarak güncellenir.

Veriler (15 kayıt) 1 / 1

reservation_id	waste_id	collector_id	created_at	pickup_datetime	status
21	1	16	2026-01-04T13:23:18.860Z	2026-01-05T14:00:00.000Z	waiting
22	6	17	2026-01-04T13:23:18.909Z	2026-01-06T10:00:00.000Z	waiting
23	16	18	2026-01-04T13:23:18.956Z	2026-01-07T16:00:00.000Z	reserved
24	22	16	2026-01-04T13:23:19.004Z	2026-01-08T11:00:00.000Z	waiting
26	24	18	2026-01-04T13:23:19.166Z	2026-01-10T15:00:00.000Z	waiting
27	25	16	2026-01-04T13:23:19.218Z	2026-01-04T13:00:00.000Z	collected
28	26	17	2026-01-04T13:23:19.265Z	2026-01-11T17:00:00.000Z	reserved
29	27	18	2026-01-04T13:23:19.312Z	2026-01-12T12:00:00.000Z	waiting
45	42	37	2026-01-07T08:44:23.092Z	2026-01-07T08:45:00.000Z	waiting
53	36	2	2026-01-08T16:55:39.807Z	2026-01-09T11:00:00.000Z	collected

Environmental_scores Tablosu

Kullanıcıların geri dönüşüm ve çevresel katkı aktivitelerini aylık olarak puanlar ve tarihsel olarak takip eder.

- **score_id** Her bir skor kaydını tekil olarak tanımlayan birincil anahtardır. İş planında, bir kullanıcının farklı aylardaki başarılarını birbirinden ayırmak için kullanılır.
- **user_id** bir foreign keyolup users tablosuna bağlanır. Puanın hangi kullanıcıya ait olduğunu belirler, kullanıcı bazlı skor takibi yapılır.
- month ve year puanların zamanını tanımlar. Bu alanlar sayesinde "Aylık En Çok Atık Toplayanlar" listesi oluşturulabilir veya kullanıcının yıllık geri dönüşüm performansı grafiklerle raporlanabilir.
- Kullanıcının ilgili ay içinde yaptığı tüm atık teslimatlarından kazandığı puanların toplamı **total_score** ile tutulur. Bu değer, waste_types tablosundaki **recycle_score** katsayıları kullanılarak hesaplanır.

Veriler (14 kayıt) 1 / 1

score_id	user_id	month	year	total_score
4	14	12	2025	120
10	15	1	2026	95
3	13	12	2025	175
9	14	1	2026	40
1	2	12	2025	150
2	12	12	2025	200
5	15	12	2025	300
8	13	1	2026	65
6	2	1	2026	301
31	37	1	2026	960

Trigger_logs Tablosu

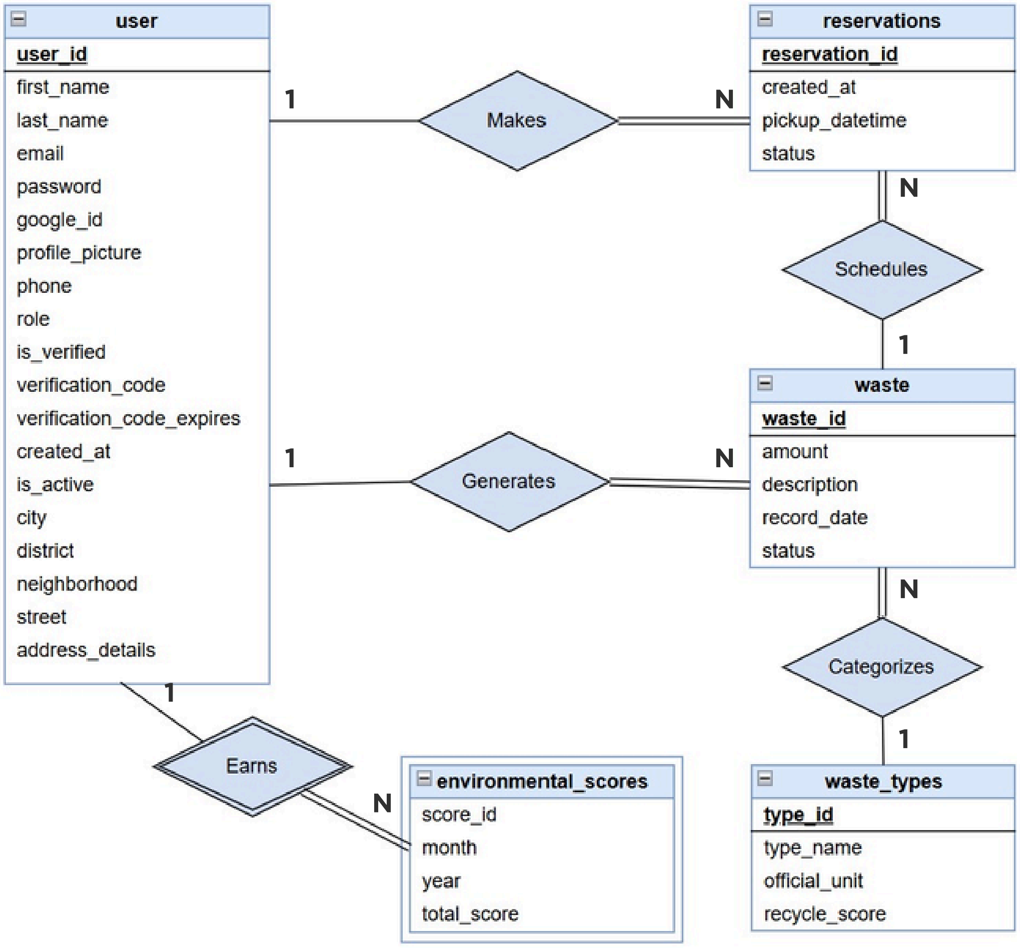
Veritabanında tetiklenen INSERT, UPDATE ve DELETE işlemlerini loglayarak sistemin güvenliğini ve denetimini sağlar.

- **log_id** gerçekleşen her bir log kaydı için sistem tarafından atanan benzersiz işlem numarasıdır.
- Değişikliği hangi tetikleyicinin gerçekleştirdiği ile **trigger_name** belirtir. Hata ayıklama süreçlerinde hangi mekanizmanın çalıştığını anlamaya yardımcı olur.
- **table_name** İşlemin hangi tablo üzerinde gerçekleştiği bilgisini tutar.
- **action** yapılan işlemin türünü kaydeder (INSERT, UPDATE veya DELETE). Bu, verinin nasıl bir değişikliğe uğradığını netleştirir.
- **old_data** ve **new_data**: JSONB formatında verinin işlemden önceki ve sonraki halini tam detaylı olarak saklar. Özellikle yanlışlıkla silinen veya güncellenen verilerin geri döndürülebilmesi için kritiktir.
- **message** işlemle ilgili sistem tarafından üretilen veya teknik ekibe yol gösterici olan özel açıklama mesajlarını içerir.
- **created_at** işlemin gerçekleştiği tam tarih ve saat bilgisini otomatik olarak kaydeder.

Veriler (38 kayıt) ‹ 1 / 2 ›

log_id	trigger_name	table_name	action	old_data	new_data	message	create
1	trg_reservation_status_c...	reservations	INSERT	NULL	[object Object]	Atığınız için yeni rezerva...	2026
2	trg_reservation_status_c...	reservations	INSERT	NULL	[object Object]	Atığınız için yeni rezerva...	2026
3	trg_reservation_status_c...	reservations	INSERT	NULL	[object Object]	Atığınız için yeni rezerva...	2026
4	trg_reservation_status_c...	reservations	INSERT	NULL	[object Object]	Atığınız için yeni rezerva...	2026
5	trg_reservation_status_c...	reservations	INSERT	NULL	[object Object]	Atığınız için yeni rezerva...	2026
6	trg_reservation_status_c...	reservations	INSERT	NULL	[object Object]	Atığınız için yeni rezerva...	2026
7	trg_reservation_status_c...	reservations	INSERT	NULL	[object Object]	Atığınız için yeni rezerva...	2026
8	trg_reservation_status_c...	reservations	INSERT	NULL	[object Object]	Atığınız için yeni rezerva...	2026
9	trg_reservation_status_c...	reservations	INSERT	NULL	[object Object]	Atığınız için yeni rezerva...	2026
10	trg_reservation_status_c	reservations	INSERT	NULL	[object Object]	Atığınız için yeni rezerva	2026

ER Diyagramı



Silme ve Sayı Kısıtları

- **users(user_id) ON DELETE CASCADE**: user_id sütununda kullanılmıştır; bir kullanıcı silindiğinde ona ait atık kayıtları da otomatik silinir.
- **waste_types(type_id) ON DELETE RESTRICT**: type_id sütununda kullanılmıştır; bir atık türü sistemde bir kayıtla ilişkiliyse, o türün silinmesini engeller.
- **CHECK (amount > 0 AND amount <= 1000)**: amount sütununa girilen değerin 0'dan büyük ve 1000'den küçük olması zorunludur.

```
CREATE TABLE IF NOT EXISTS waste (
    waste_id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL REFERENCES users(user_id) ON DELETE CASCADE,
    type_id INTEGER NOT NULL REFERENCES waste_types(type_id) ON DELETE RESTRICT,
    amount DECIMAL(10,2) NOT NULL CHECK (amount > 0 AND amount <= 1000),
    description TEXT,
    record_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(20) DEFAULT 'waiting' CHECK (status IN ('waiting', 'reserved', 'collected', 'cancelled'))
);
```

Sorgu Örneği

Kullanıcı arayüz üzerinden belirli bir atık türünü (örneğin: 'PET Bottles') ve durumunu ('waiting') seçtiğinde, arka planda waste, waste_types ve users tabloları JOIN edilerek kullanıcıya anlamlı bir liste sunulmaktadır.

```
SELECT w.waste_id, wt.type_name, w.amount, w.status, u.first_name || ' ' || u.last_name as namesurname
FROM waste w
JOIN waste_types wt ON w.type_id = wt.type_id
JOIN users u ON w.user_id = u.user_id
WHERE wt.type_name = 'PET Bottles'
AND w.status = 'waiting';
```

waste_id	type_name	amount	status	namesurname
24	PET Bottles	2.50	waiting	Ahmet Yılmaz

Roller

Sistemde 'admin' ve 'resident' olmak üzere iki farklı kullanıcı tipi tanımlanmıştır. CHECK kısıtı ile bu sütuna yanlışlıkla veya izinsiz başka bir rol ismi yazılması veritabanı seviyesinde engellenmiştir. Yeni kayıt olan her kullanıcı aksi belirtilmedikçe otomatik olarak 'resident' (standart kullanıcı) yetkisiyle başlatılır.

```
role VARCHAR(20) NOT NULL DEFAULT 'resident' CHECK (role IN ('admin', 'resident')),
```

View ve Sorgu Örneği

v_active_waste_details isimli bu view; waste, waste_types ve users tablolarını birbirine bağlayarak , atık bilgilerini, tür detaylarını ve kullanıcı iletişim bilgilerini tek bir yapıda birleştirir. Arayüz üzerinde 'İstanbul' gibi belirli bir şehre göre filtreleme yapıldığında, arka planda karmaşık JOIN sorguları yazmak yerine direkt olarak bu View üzerinden SELECT * FROM v_active_waste_details WHERE city = 'İstanbul'; sorgusu çalıştırılmaktadır.

```sql
CREATE OR REPLACE VIEW v_active_waste_details AS
SELECT
    w.waste_id,
    w.amount,
    w.description,
    w.status,
    w.record_date,
    wt.type_name,
    wt.official_unit,
    wt.recycle_score,
    u.user_id,
    u.first_name || ' ' || u.last_name AS owner_name,
    u.city,
    u.district,
    u.neighborhood,
    u.phone AS owner_phone
FROM waste w
JOIN waste_types wt ON w.type_id = wt.type_id
JOIN users u ON w.user_id = u.user_id
WHERE w.status IN ('waiting', 'reserved')
ORDER BY w.record_date DESC;
```

```sql
SELECT * FROM v_active_waste_details
WHERE city = 'İstanbul';
```

waste_id	amount	description	status	record_date	type_name	official_unit	recycle_score	user_id	owner_name	city
46	2.00	eski ütü	waiting	2026-01-13 09:39:53.899815	Small Household Appliances	unit	250	12	Bera Çolak	İstanbul
43	1.00	USB kablo, adaptör, kulaklık kablosu	waiting	2026-01-08 07:58:26.376189	Cables & Chargers	kg	180	12	Bera Çolak	İstanbul
38	4.00		waiting	2026-01-04 20:44:04.136924	Glass Bottles & Jars	liter	30	37	Sümeyye Yoldaş	İstanbul
32	3.50	Test karton kutu	reserved	2026-01-04 13:44:30.152201	Cardboard Boxes & Packaging	kg	40	13	Ahmet Yılmaz	İstanbul
30	6.50	Plastik oyuncaklar	waiting	2026-01-04 13:23:18.621371	Hard Plastic Packaging	kg	80	14	Ayşe Kaya	İstanbul
29	4.00	Kırık cam eşyalar	waiting	2026-01-04 13:23:18.572323	Old Books & Newspapers	kg	60	13	Ahmet Yılmaz	İstanbul
27	8.00	Gazete ve dergiler	waiting	2026-01-04 13:23:18.402949	Cardboard Boxes & Packaging	kg	40	2	Ayşe Sude Cami	İstanbul
25	10.00	PET şişeler ve plastik kaplar	waiting	2026-01-04 13:23:18.299204	Hard Plastic Packaging	kg	80	14	Ayşe Kaya	İstanbul
24	2.50	Eski bilgisayar parçaları	waiting	2026-01-04 13:23:18.24456	PET Bottles	liter	100	13	Ahmet Yılmaz	İstanbul
22	5.50	Temiz karton kutular, ürün ambalajları	waiting	2026-01-04 13:23:18.089652	Cardboard Boxes & Packaging	kg	40	2	Ayşe Sude Cami	İstanbul

Indexler

Veritabanı sorgu performansını artırmak için arayüzde sık kullanılan sütunlar üzerinde indexler tanımlanmıştır. Kullanıcı tablosunda email, phone, city ve role alanlarına eklenen indexler doğrulama ve şehir bazlı sorguları hızlandırırken, atık tablosunda status, type_id, record_date ve user_id alanlarına oluşturulan indexler atıkların durum, tür, tarih ve kullanıcıya göre hızlı filtrelenmesini sağlamaktadır. Bu sayede "İstanbul'daki aktif atıklar" veya "PET Şişe" gibi aramalar için tüm tabloyu aramak yerine indexler kullanılarak yüksek performansla gerçekleştirilmektedir.

```sql
CREATE INDEX IF NOT EXISTS idx_users_email ON users(email);
CREATE INDEX IF NOT EXISTS idx_users_phone ON users(phone);
CREATE INDEX IF NOT EXISTS idx_users_city ON users(city);
CREATE INDEX IF NOT EXISTS idx_users_role ON users(role);
CREATE INDEX IF NOT EXISTS idx_waste_status ON waste(status);
CREATE INDEX IF NOT EXISTS idx_waste_type ON waste(type_id);
CREATE INDEX IF NOT EXISTS idx_waste_user ON waste(user_id);
CREATE INDEX IF NOT EXISTS idx_waste_date ON waste(record_date DESC);
```

Sequence Örneği

reservation_number_seq sequence'i, daha önce oluşturulmadıysa 1000'den başlayacak şekilde tanımlanmıştır ve her yeni kayıt eklendiğinde değeri 1'er artırır. Üst sınırının olmaması sayesinde numaralar veritabanı kapasitesi el verdiği sürece artmaya devam eder. CACHE 1 kullanımı, her numaranın ihtiyaç anında üretilmesini sağlayarak beklenmedik durumlarda numara atlanması riskini azaltır.

Sistemde her yeni rezervasyon oluşturulduğunda bu sequence kullanılarak otomatik bir rezervasyon numarası atanır. Insert işlemi sırasında nextval('reservation_number_seq') fonksiyonu çağrılarak benzersiz ve sıralı bir numara üretilir. Böylece manuel müdahaleye gerek kalmadan, rezervasyon numaralarının benzersizliği ve düzenli artışı veritabanı seviyesinde güvence altına alınmış olur.

```sql
CREATE SEQUENCE IF NOT EXISTS reservation_number_seq
    START WITH 1000
    INCREMENT BY 1
    NO MAXVALUE
    CACHE 1;
```

→ INSERT işlemi

Kullanıcı "Rezerve Et" butonuna tıkladığında arka planda şu süreç işler. reservations tablosuna yeni bir kayıt eklenir.

INSERT INTO reservations (user_id, waste_id, pickup_datetime, status)
VALUES (mevcut_kullanici_id, secilen_atik_id, '2026-01-16 14:00', 'waiting');

Bu işlem yapıldığı anda, yazdığımız trg_fn_reservation_status_change fonksiyonu tetiklenir ve ilgili atığın durumunu otomatik olarak günceller.

→ UPDATE İşlemi

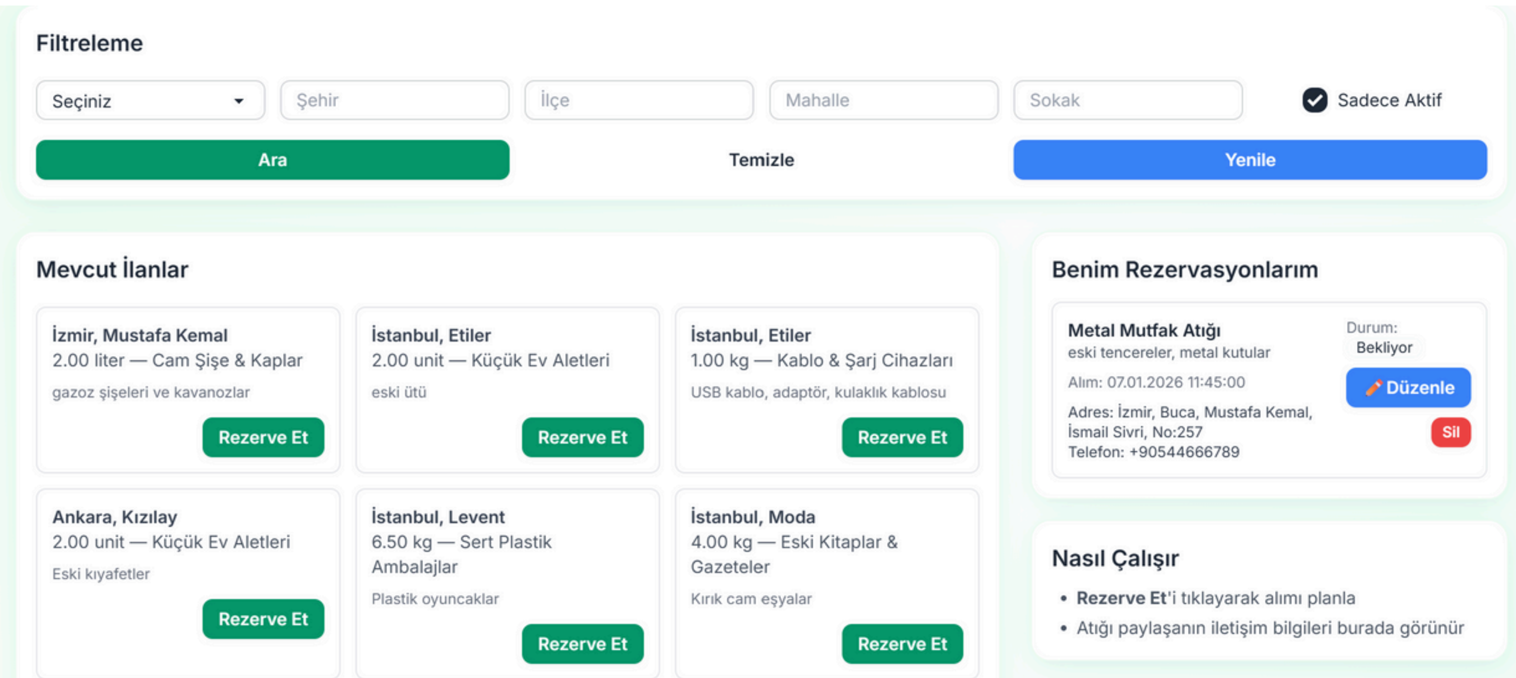
Arayüzün sağ tarafındaki "Benim Rezervasyonlarım" panelinde bulunan "Düzenle" butonu bu işlemi temsil eder. Kullanıcı alım tarihini veya rezervasyon notlarını değiştirmek istediğinde UPDATE komutu çalışır.

UPDATE reservations SET pickup_datetime = 'Yeni_Tarih'WHERE reservation_id = secilen_id;

->DELETE İŞLEMİ

"Benim Rezervasyonlarım" panelinde "Sil" butonu bu operasyonu gerçekleştirir. Kullanıcı yaptığı rezervasyondan vazgeçtiğinde ilgili kayıt veritabanından kaldırılır veya "İptal" durumuna çekilir.

DELETE FROM reservations WHERE reservation_id = secilen_id;



Sorgu Örneği (HAVING)

SQL sorgusu, öncelikle users ve waste tablolarını birleştirerek her kullanıcının paylaştığı atıkları ilişkilendirir. Daha sonra kullanıcı bazında gruplama yapılarak her bir kullanıcının toplam atık sayısı ve paylaştığı atıkların toplam miktarı hesaplanır. HAVING koşulu sayesinde yalnızca en az iki atık paylaşmış kullanıcılar tabloya dahil edilir ve ORDER BY ile en çok atık paylaşanlar üstte olacak şekilde sıralanır. Sonuç olarak, arayüzdeki tablo kullanıcıların isimlerini, şehirlerini, toplam atık sayılarını ve miktarlarını gösterir; böylece kullanıcıların atık paylaşım performansı kolayca takip edilebilir.

```sql
SELECT
    u.user_id,
    u.first_name || ' ' || u.last_name AS full_name,
    COUNT(w.waste_id) AS waste_count,
    SUM(w.amount) AS total_amount
FROM users u
JOIN waste w ON u.user_id = w.user_id
GROUP BY u.user_id, u.first_name, u.last_name
HAVING COUNT(w.waste_id) >= 2
ORDER BY waste_count DESC;
```


Aggregate + HAVING Sorgusu

En az 2 atık paylaşmış kullanıcılar

Min atık sayısı: 2 Filtrele

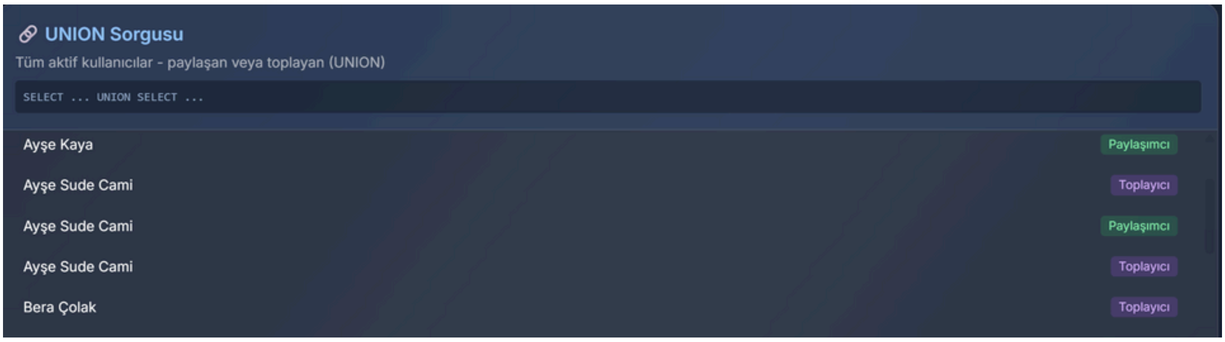
HAVING COUNT(w.waste_id) >= 2

Kullanıcı	Şehir	Atık Sayısı	Toplam Miktar	Toplanan	Puan
Mehmet Demir	Ankara	4	6.5	0	0
Sümeyye Yoldaş	İstanbul	3	9.0	2	480
Bera Çolak	İstanbul	3	7.5	1	270
Ahmet Yılmaz	İstanbul	3	10.0	0	0
Ayşe Kaya	İstanbul	2	16.5	0	0
Vera Doe	İzmir	2	3.0	0	0
Ayşe Sude Cami	İstanbul	2	13.5	0	0

Sorgu Örneği (UNION)

Bu SQL sorgusu, arayüzde platformda işlem yapmış kullanıcıları tek bir listede göstermek amacıyla kullanılır. Sorgu, waste tablosundaki kullanıcılar “Paylaşımcı”, reservations tablosundaki collector_id kullanıcıları ise “Toplayıcı” etiketiyle listelenir. UNION aynı olan satırları tekilleştirir; ancak rol bilgileri farklı olduğu için aynı kullanıcı arayüzde iki farklı rolde ayrı satırlar halinde görünebilir.

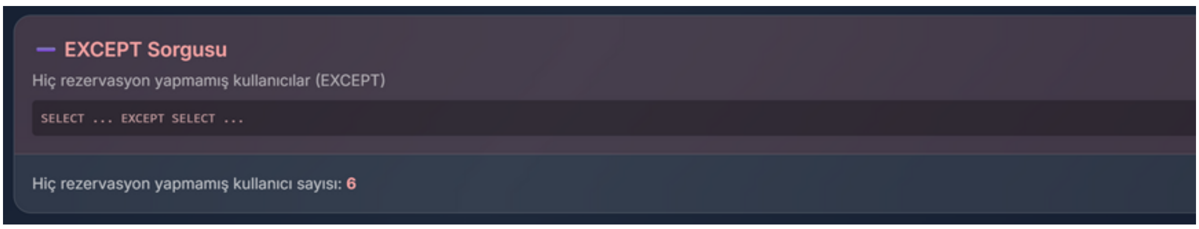
```
SELECT user_id, first_name, last_name, 'Paylaşımcı' as role_type
FROM users u
WHERE EXISTS (SELECT 1 FROM waste w WHERE w.user_id = u.user_id)
UNION
SELECT user_id, first_name, last_name, 'Toplayıcı' as role_type
FROM users u
WHERE EXISTS (SELECT 1 FROM reservations r WHERE r.collector_id = u.user_id);
```



Sorgu Örneği (EXCEPT)

Bu sorgu, EXCEPT kullanarak rezervasyon yapmış kullanıcıları tüm kullanıcılar arasından çıkarır ve geriye kalan, yani hiç rezervasyon yapmamış kullanıcıların sayısını hesaplar. Alt sorguda rezervasyon yapmayan kullanıcılar bulunur, dış sorguda ise bu kullanıcıların toplam sayısı alınır. Sonuç olarak arayüzde rezervasyon yapmayan kullanıcı sayısı gösterilir.

```
SELECT user_id, first_name, last_name FROM users
EXCEPT
SELECT DISTINCT u.user_id, u.first_name, u.last_name
FROM users u
SELECT COUNT(*) AS reservation_yapmayan_sayisi
FROM (
   SELECT user_id, first_name, last_name
   FROM users
   EXCEPT
   SELECT DISTINCT u.user_id, u.first_name, u.last_name
   FROM users u
   JOIN reservations r
   ON u.user_id = r.collector_id
) t;
```



Fonksiyon Örnekleri

→ fn_get_waste_by_city

Parametre olarak alınan şehir bilgisine (p_city) göre atıkları listelemek için kullanılır. Fonksiyon; waste, waste_types ve users tablolarını join ederek atığın türü, miktarı, sahibinin adı-soyadı, ilçesi ve mevcut durumu gibi arayüzde gerekli olan tüm bilgileri tek bir sonuç olarak döndürür. Sadece waiting ve reserved durumundaki atıklar filtrelenir ve kayıtlar en güncel olandan başlayacak şekilde sıralanır. Bu yapı, şehir bazlı atık listeleme sorgularının tekrar yazılmasını önler.

```sql
CREATE OR REPLACE FUNCTION fn_get_waste_by_city(p_city VARCHAR)
RETURNS TABLE(
    waste_id INTEGER,
    type_name VARCHAR,
    amount DECIMAL,
    owner_name TEXT,
    district VARCHAR,
    status VARCHAR
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        w.waste_id,
        wt.type_name,
        w.amount,
        (u.first_name || ' ' || u.last_name)::TEXT,
        u.district,
        w.status
    FROM waste w
    JOIN waste_types wt ON w.type_id = wt.type_id
    JOIN users u ON w.user_id = u.user_id
    WHERE u.city = p_city AND w.status IN ('waiting', 'reserved')
    ORDER BY w.record_date DESC;
END;
$$ LANGUAGE plpgsql;
```

→ fn_calculate_user_total_score

Belirli bir kullanıcının kazandığı toplam puanı hesaplamak için tasarlanmıştır. Fonksiyon, yalnızca collected durumundaki atıkları dikkate alır ve her atık için miktar (amount) ile atık türüne ait geri dönüşüm puanını (recycle_score) çarparak toplam puanı hesaplar. COALESCE kullanımı sayesinde, kullanıcının hiç toplanmış atığı olmasa bile sonuç 0 olarak döndürülür.

```sql
CREATE OR REPLACE FUNCTION fn_calculate_user_total_score(p_user_id INTEGER)
RETURNS INTEGER AS $$
DECLARE
    total INTEGER;
BEGIN
    SELECT COALESCE(SUM(w.amount * wt.recycle_score), 0)::INTEGER
    INTO total
    FROM waste w
    JOIN waste_types wt ON w.type_id = wt.type_id
    WHERE w.user_id = p_user_id AND w.status = 'collected';

    RETURN total;
END;
$$ LANGUAGE plpgsql;
```

→ fn_get_user_monthly_report

Bir kullanıcının belirli bir yıl ve ay içindeki atık toplama faaliyetlerini raporlamak için kullanılır. Fonksiyon, yalnızca “collected” durumundaki atıkları dikkate alır. Öncelikle atık türleri bazında, her tür için kaç adet atık toplandığı, toplam miktarı ve kazandığı puan hesaplanır. Cursor kullanılarak bu veriler tek tek alınır ve her atık türü için bir satır olarak rapora eklenir. Bu satırlar “WASTE_DETAIL” türünde olur ve kullanıcıya hangi türden kaç atık toplandığı gibi detayları gösterir.

Cursor döngüsü sırasında fonksiyon, tüm atıkları topluca sayarak ve miktarlarını ve puanlarını ekleyerek genel toplamları da hesaplar. Döngü bittikten sonra, fonksiyon “SUMMARY” türünde tek bir satır döndürür. Bu satır, seçilen ay için toplam atık sayısını, toplam miktarı ve toplam puanı gösterir. Böylece hem detaylı hem de özet bir rapor tek seferde elde edilmiş olur.

```sql
CREATE OR REPLACE FUNCTION fn_get_user_monthly_report(
    p_user_id INTEGER,
    p_year INTEGER,
    p_month INTEGER
)
RETURNS TABLE(
    report_type VARCHAR,
    type_name VARCHAR,
    item_count BIGINT,
    total_amount DECIMAL,
    total_score BIGINT,
    details TEXT
) AS $$
DECLARE
    waste_cursor CURSOR FOR
        SELECT
            wt.type_name,
            COUNT(*) as cnt,
            SUM(w.amount) as total_amt,
            SUM(w.amount * wt.recycle_score)::BIGINT as score
        FROM waste w
        JOIN waste_types wt ON w.type_id = wt.type_id
        WHERE w.user_id = p_user_id
          AND EXTRACT(YEAR FROM w.record_date) = p_year
          AND EXTRACT(MONTH FROM w.record_date) = p_month
          AND w.status = 'collected'
        GROUP BY wt.type_name;

    waste_record RECORD;
    total_items BIGINT := 0;
    grand_total_amount DECIMAL := 0;
    grand_total_score BIGINT := 0;
BEGIN
    -- Cursor ile atık verilerini oku
    OPEN waste_cursor;
    LOOP
        FETCH waste_cursor INTO waste_record;
        EXIT WHEN NOT FOUND;

        -- Her atık türü için bir satır döndür
        report_type := 'WASTE_DETAIL';
        type_name := waste_record.type_name;
        item_count := waste_record.cnt;
        total_amount := waste_record.total_amt;
        total_score := waste_record.score;
        details := 'Atık türü detayı: ' || waste_record.type_name;
        RETURN NEXT;

        -- Toplamları güncelle
        total_items := total_items + waste_record.cnt;
        grand_total_amount := grand_total_amount + waste_record.total_amt;
        grand_total_score := grand_total_score + waste_record.score;
    END LOOP;
    CLOSE waste_cursor;

    -- Özet satırı döndür
    report_type := 'SUMMARY';
    type_name := 'TOPLAM';
    item_count := total_items;
    total_amount := grand_total_amount;
    total_score := grand_total_score;
    details := p_year || '-' || p_month || ' dönemi özeti';
    RETURN NEXT;
END;
$$ LANGUAGE plpgsql;
```

Trigger Örnekleri

→ trg_fn_update_environmental_score

INSERT veya UPDATE işlemlerinde çalışacak şekilde tasarlanmıştır. Ancak puan yalnızca atığın durumu “collected” olduğunda hesaplanır. Yani, atık önceden toplanmadıysa ve şimdi toplandıysa işlem yapılır.Atığın türüne göre puan katsayısı (recycle_score) waste_types tablosundan alınır.Atığın miktarı ile bu katsayı çarpılarak kullanıcıya verilecek puan hesaplanır.

Environmental_scores tablosunu güncelleme yapılır. Fonksiyon, o ay ve yıl için kullanıcıya ait bir kayıt varsa puanı günceller. Eğer kayıt yoksa yeni bir satır ekler. Bu işlem UPSERT mantığıyla yapılır (INSERT … ON CONFLICT … DO UPDATE). Kullanıcının kazandığı puan için bir mesaj oluşturulur. Örnek: “Tebrikler! 15 çevresel puan kazandınız!” Bu mesaj ve işlem bilgisi trigger_logs tablosuna kaydedilir. Böylece tüm trigger aktiviteleri izlenebilir. Fonksiyon RETURN NEW ile, yapılan değişikliği veritabanına uygular.

```sql
CREATE OR REPLACE FUNCTION trg_fn_update_environmental_score()
RETURNS TRIGGER AS $$
DECLARE
    v_score INTEGER;
    v_current_month INTEGER;
    v_current_year INTEGER;
    v_message TEXT;
BEGIN
    -- Sadece status 'collected' olduğunda puan hesapla
    IF NEW.status = 'collected' AND (TG_OP = 'INSERT' OR OLD.status != 'collected') THEN
        -- Atık türünün puanını al
        SELECT recycle_score INTO v_score
        FROM waste_types
        WHERE type_id = NEW.type_id;

        v_current_month := EXTRACT(MONTH FROM CURRENT_DATE);
        v_current_year := EXTRACT(YEAR FROM CURRENT_DATE);

        -- Environmental score güncelle veya ekle
        INSERT INTO environmental_scores (user_id, month, year, total_score)
        VALUES (NEW.user_id, v_current_month, v_current_year, (NEW.amount * v_score)::INTEGER)
        ON CONFLICT (user_id, month, year)
        DO UPDATE SET total_score = environmental_scores.total_score + (NEW.amount * v_score)::INTEGER;

        v_message := 'Tebrikler! ' || (NEW.amount * v_score)::INTEGER || ' çevresel puan kazandınız!';

        -- Trigger log'a kaydet
        INSERT INTO trigger_logs (trigger_name, table_name, action, new_data, message)
        VALUES ('trg_update_environmental_score', 'waste', TG_OP, row_to_json(NEW), v_message);
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trg_update_environmental_score ON waste;
CREATE TRIGGER trg_update_environmental_score
    AFTER INSERT OR UPDATE ON waste
    FOR EACH ROW
    EXECUTE FUNCTION trg_fn_update_environmental_score();
```

→ trg_fn_reservation_status_change

Bu trigger, rezervasyon eklenip güncellendiğinde atıkların durumunu otomatik olarak değiştirmek için kullanılır. Yeni bir rezervasyon yapıldığında, ilgili atık otomatik olarak 'reserved' durumuna geçer. Trigger, rezervasyonu yapan kişinin bilgilerini alır ve kullanıcıya veya sistemde kaydedilecek bir mesaj hazırlar. Örneğin: "Atığınız için yeni rezervasyon! Toplayıcı: Ahmet Yılmaz, Tarih: 15.01.2026 14:00". Bu mesaj ve işlem bilgisi trigger_logs tablosuna kaydedilir.

Rezervasyon güncellendiğinde ise iki durum kontrol edilir: Eğer rezervasyon 'collected' yapılırsa, atık da otomatik olarak 'collected' durumuna geçer ve log tutulur. Eğer rezervasyon 'cancelled' olursa, atık tekrar 'waiting' durumuna gelir ve işlem yine loglanır. Böylece atıkların durumu her zaman doğru kalır ve yapılan tüm işlemler takip edilebilir.

```sql
CREATE OR REPLACE FUNCTION trg_fn_reservation_status_change()
RETURNS TRIGGER AS $$
DECLARE
    v_message TEXT;
    v_waste_owner_id INTEGER;
    v_collector_name TEXT;
BEGIN
    IF TG_OP = 'INSERT' THEN
        -- Yeni rezervasyon eklendiğinde atık durumunu 'reserved' yap
        UPDATE waste SET status = 'reserved' WHERE waste_id = NEW.waste_id;

        SELECT user_id INTO v_waste_owner_id FROM waste WHERE waste_id = NEW.waste_id;
        SELECT first_name || ' ' || last_name INTO v_collector_name FROM users WHERE user_id = NEW.collector_id;

        v_message := 'Atığınız için yeni rezervasyon! Toplayıcı: ' || v_collector_name ||
                     ', Tarih: ' || TO_CHAR(NEW.pickup_datetime, 'DD.MM.YYYY HH24:MI');

        INSERT INTO trigger_logs (trigger_name, table_name, action, new_data, message)
        VALUES ('trg_reservation_status_change', 'reservations', 'INSERT', row_to_json(NEW), v_message);

    ELSIF TG_OP = 'UPDATE' THEN
        IF NEW.status = 'collected' AND OLD.status != 'collected' THEN
            UPDATE waste SET status = 'collected' WHERE waste_id = NEW.waste_id;

            v_message := 'Atık başarıyla toplandı! Reservation ID: ' || NEW.reservation_id;

            INSERT INTO trigger_logs (trigger_name, table_name, action, old_data, new_data, message)
            VALUES ('trg_reservation_status_change', 'reservations', 'UPDATE', row_to_json(OLD), row_to_json(NEW), v_message);

        ELSIF NEW.status = 'cancelled' AND OLD.status != 'cancelled' THEN
            UPDATE waste SET status = 'waiting' WHERE waste_id = NEW.waste_id;

            v_message := 'Rezervasyon iptal edildi. Atık tekrar listeye eklendi.';

            INSERT INTO trigger_logs (trigger_name, table_name, action, old_data, new_data, message)
            VALUES ('trg_reservation_status_change', 'reservations', 'UPDATE', row_to_json(OLD), row_to_json(NEW), v_message);
        END IF;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```