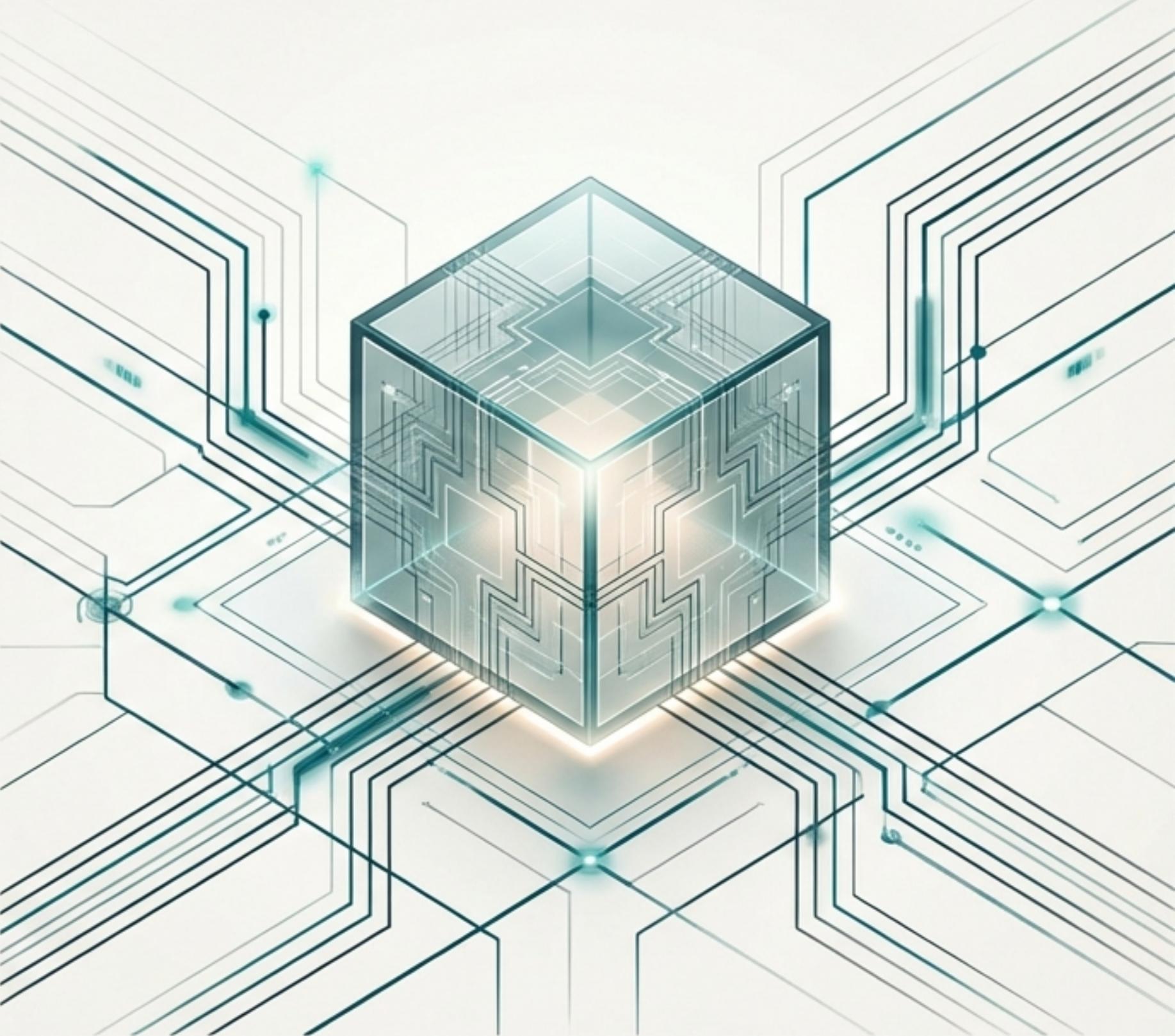
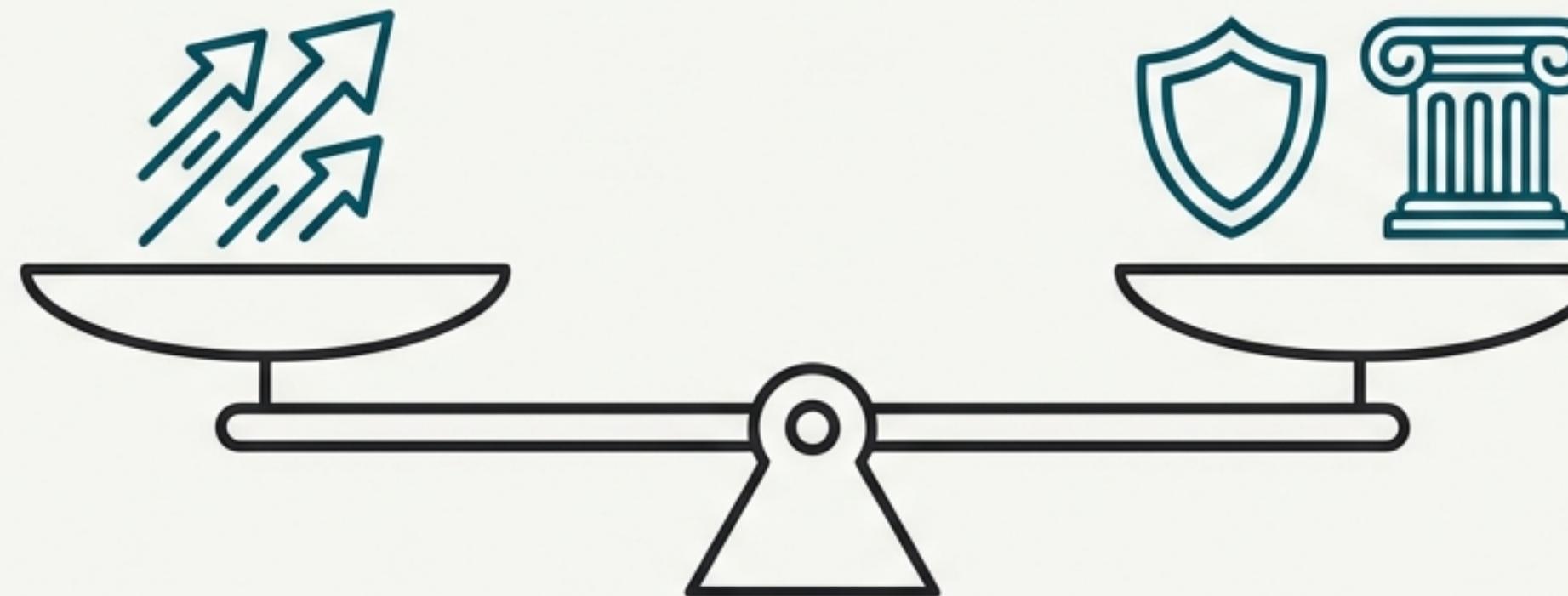


Banka Simülasyon Sistemi: Yüksek Performanslı ve Güvenli FinTech Mimarisi

- Geleneksel monolitik yapıların performans darboğazlarını aşmak için tasarlanmış modern bir Core Banking sistemi.
- Dolandırıcılık tespiti, regülasyon uyumluluğu ve denetim mekanizmalarını mimarının merkezine yerleştiren bütünsel bir yaklaşım.
- ACID prensiplerinden ödün vermeden saniyede binlerce işlemi (TPS) yönetme kapasitesi.
- Clean Architecture prensipleriyle geliştirilmiş, sürdürülebilir ve ölçeklenebilir bir altyapı.



FinTech'teki İkilem: Performans, Güvenlik ve Regülasyon Dengesi



Çözüm Yolculuğumuz

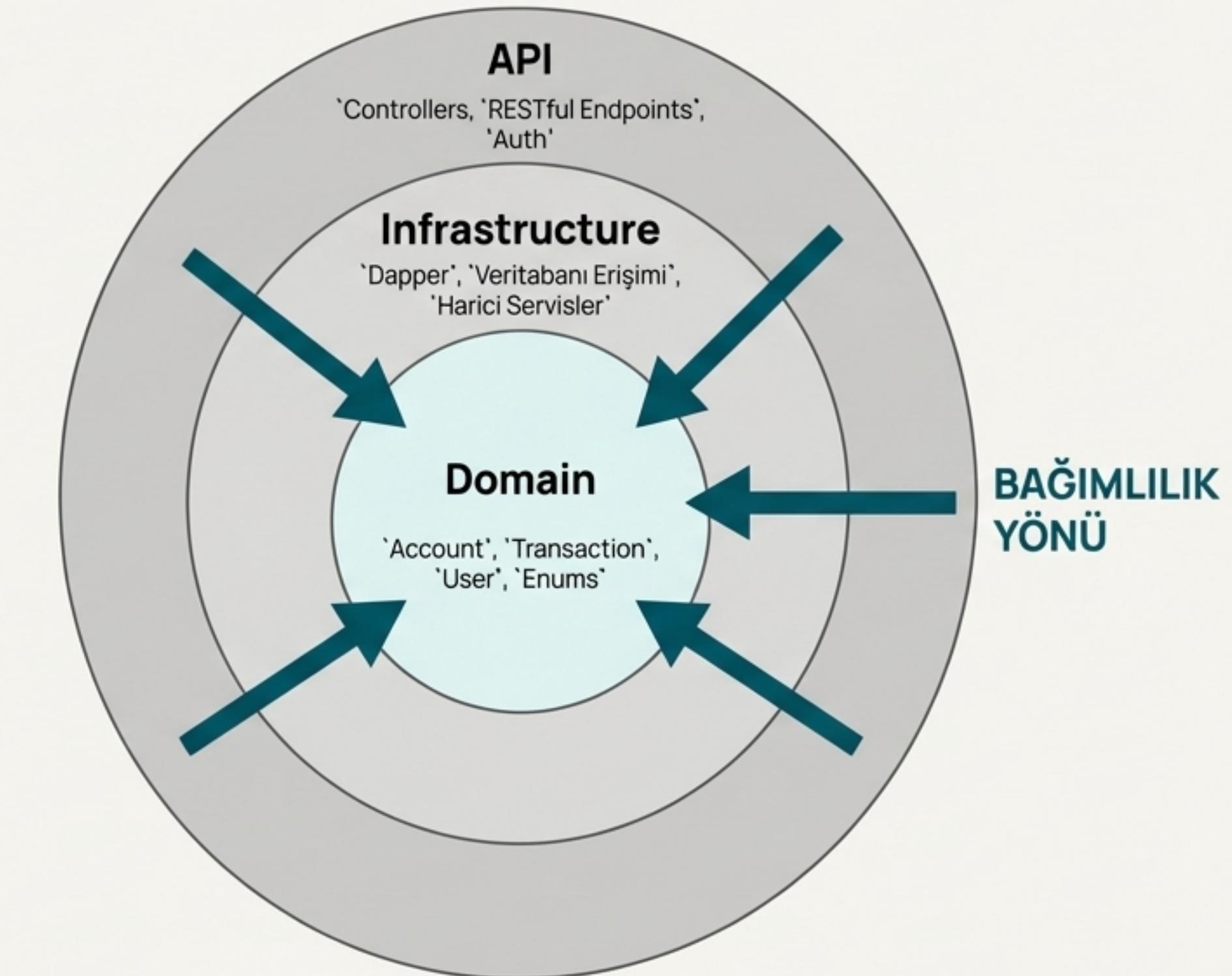
Problem: Ağır ORM'ler ve monolitik yapılar, yüksek işlem hacmi altında ölçeklenemiyor ve güvenlik zayıflıkları yaratıyor.

Çözüm Yolculuğumuz:

- Mimari Çözüm:** Esnek ve test edilebilir bir temel oluşturma.
- Veritabanı Derinliği:** Veri katmanında milisaniye düzeyinde performans mühendisliği.
- Güvenlik & Uyum:** Finansal varlıklarını korumak için çok katmanlı savunma.
- Sonuç & Vizyon:** Stratejik kazanımlar ve geleceğe yönelik yol haritası.

Mimari Felsefemiz: Temiz Mimari (Clean Architecture)

- **Katmanlı Bağımsızlık:** Bağımlılıklar tek yönde, dış katmanlardan merkeze (Domain) doğru ilerler. Teknoloji seçimleri iş kurallarını etkilemez.
- **Domain Katmanı (`BankSimulation.Domain`):** Sistemin kalbi. `Account`, `Transaction` gibi varlıkları içeren, harici bağımlılığı olmayan saf iş mantığı.
- **Infrastructure Katmanı (`BankSimulation.Infrastructure`):** Veritabanı erişimi (Dapper), harici servisler gibi “teknik detayları” barındırır ve Domain'i kirletmez.
- **API Katmanı (`BankSimulation.API`):** Dış dünya ile iletişimini sağlayan, RESTful prensiplerine uygun, ince bir kontrol ve giriş katmanı.

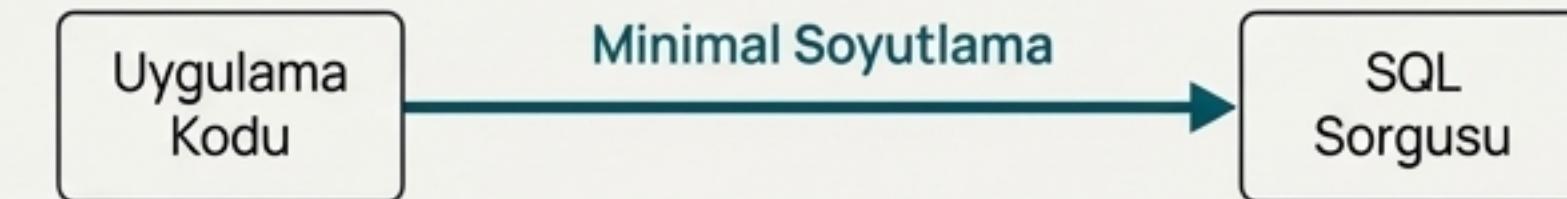


Stratejik Performans Kararı: Neden Entity Framework Değil, Dapper?

Entity Framework Core



Dapper (Micro-ORM)



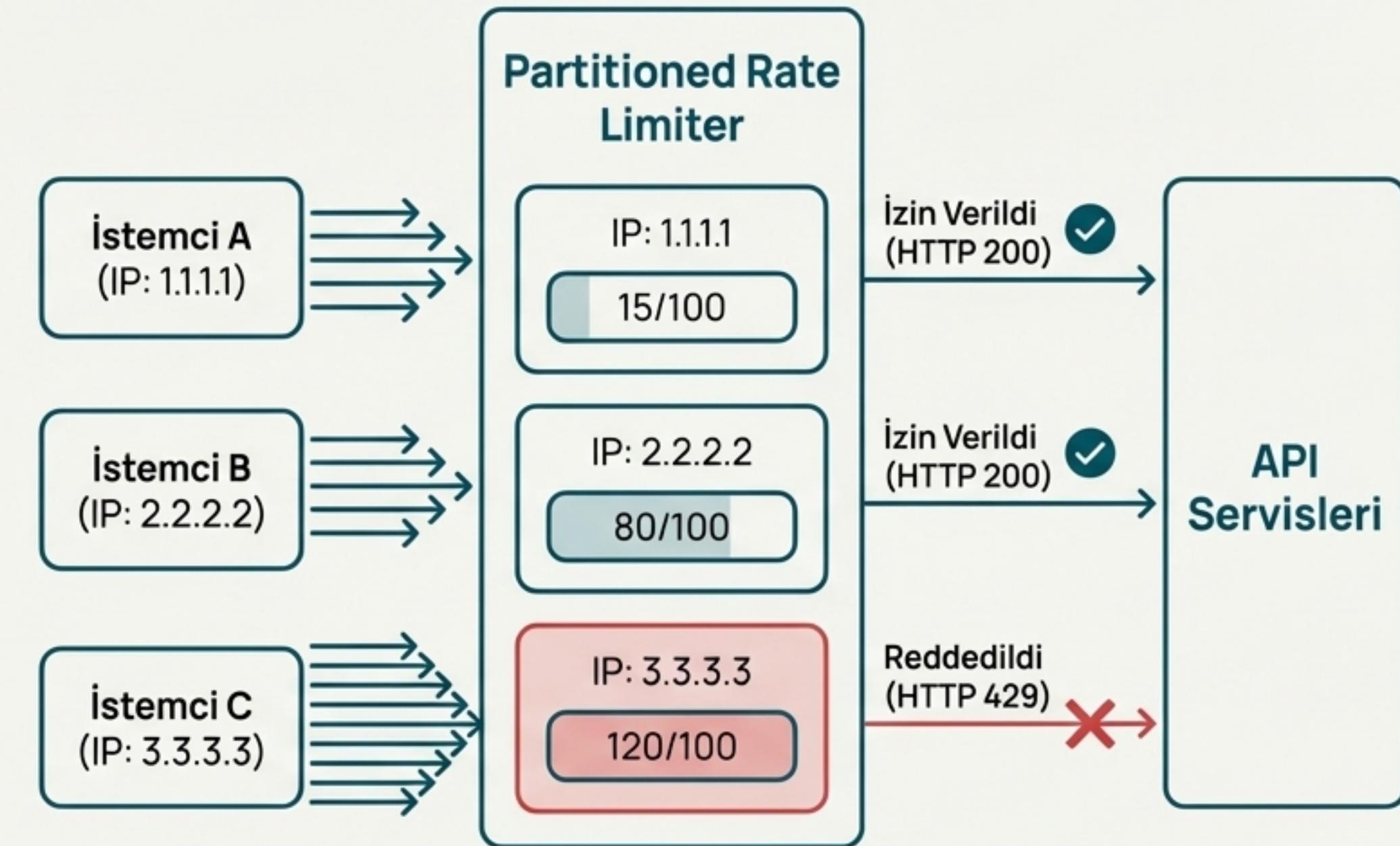
- **Sıfır "Change Tracking" Maliyeti:** EF Core'un her nesneyi izlemesinin getirdiği bellek (Memory Allocation) ve CPU yükü (overhead) Dapper'da yoktur. Finansal işlemlerde bu kritik bir avantajdır.
- **Tam SQL Hakimiyeti:** EF Core'un ürettiği potansiyel olarak sub-optimal sorgular yerine, elle optimize edilmiş, veritabanı Execution Plan'ına tam uyumlu SQL sorguları çalışma imkanı.

- **Minimal Soyutlama:** Dapper, bir Micro-ORM olarak, veriyi doğrudan POCO nesnesine map ederek gereksiz ara katmanları ve performans kaybını ortadan kaldırır.

Sonuç: Finansal işlemlerde kritik olan milisaniye düzeyindeki gecikmeyi (latency) minimize ederek daha yüksek TPS (Transaction Per Second) elde etmek.

API Koruması: Partitioned Rate Limiter ile Uygulama Seviyesinde Savunma

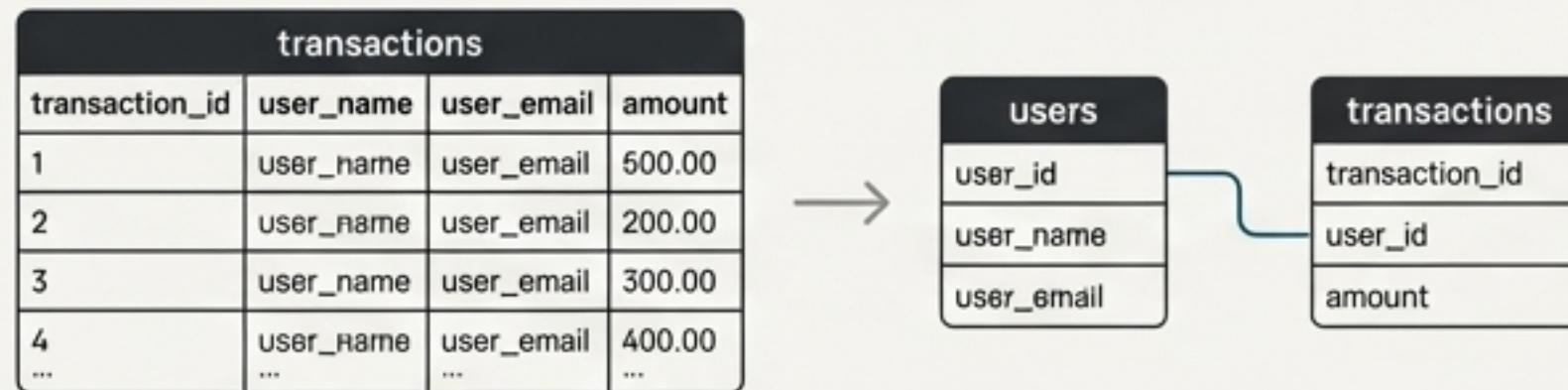
- Algoritma:** .NET 8'in sunduğu `PartitionedRateLimiter` kullanılarak her istemci (IP adresi veya API anahtarları bazlı) için ayrı bir istek sayacı tutulması.
- Strateji: Sabit Pencere (Fixed Window):** Her bir IP adresi için dakikada maksimum 100 istek limiti tanımlanmıştır.
- Kuyruk Yönetimi (Queue Limit):** Anlık yığınlamaları önlemek için her IP için 20 isteklik bir kuyruk limiti belirlenmiştir. Bu limit aşıldığında istekler direkt reddedilir (HTTP 429 Too Many Requests).
- Fayda:** Brute-force saldırıları, agresif botlar ve API'nin kötüye kullanımına karşı uygulama katmanında, daha alt katmanlara ulaşmadan proaktif bir savunma hattı oluşturur.



Veritabanı Mimarisi: 3. Normal Form ve İlişkisel Bütünlük

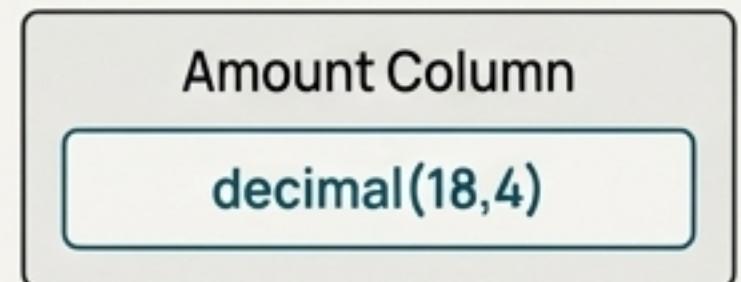
Normalizasyon (3NF)

38 tablonun tamamı, veri tekrarını (redundancy) ve güncelleme anomalilerini engellemek için 3. Normal Form (3NF) kurallarına uygun olarak tasarlanmıştır.



Veri Tipi Optimizasyonu

Her sütun için mümkün olan en dar ve en uygun veri tipi seçilerek (örn: para için `decimal(18,4)`) depolama alanı ve sorgu performansı optimize edilmiştir.



Referans Bütünlüğü

`FOREIGN KEY` kısıtlamaları ve `ON DELETE CASCADE` gibi kurallar, yetim kayıtların (Orphan Data) oluşmasını veritabanı seviyesinde engeller.

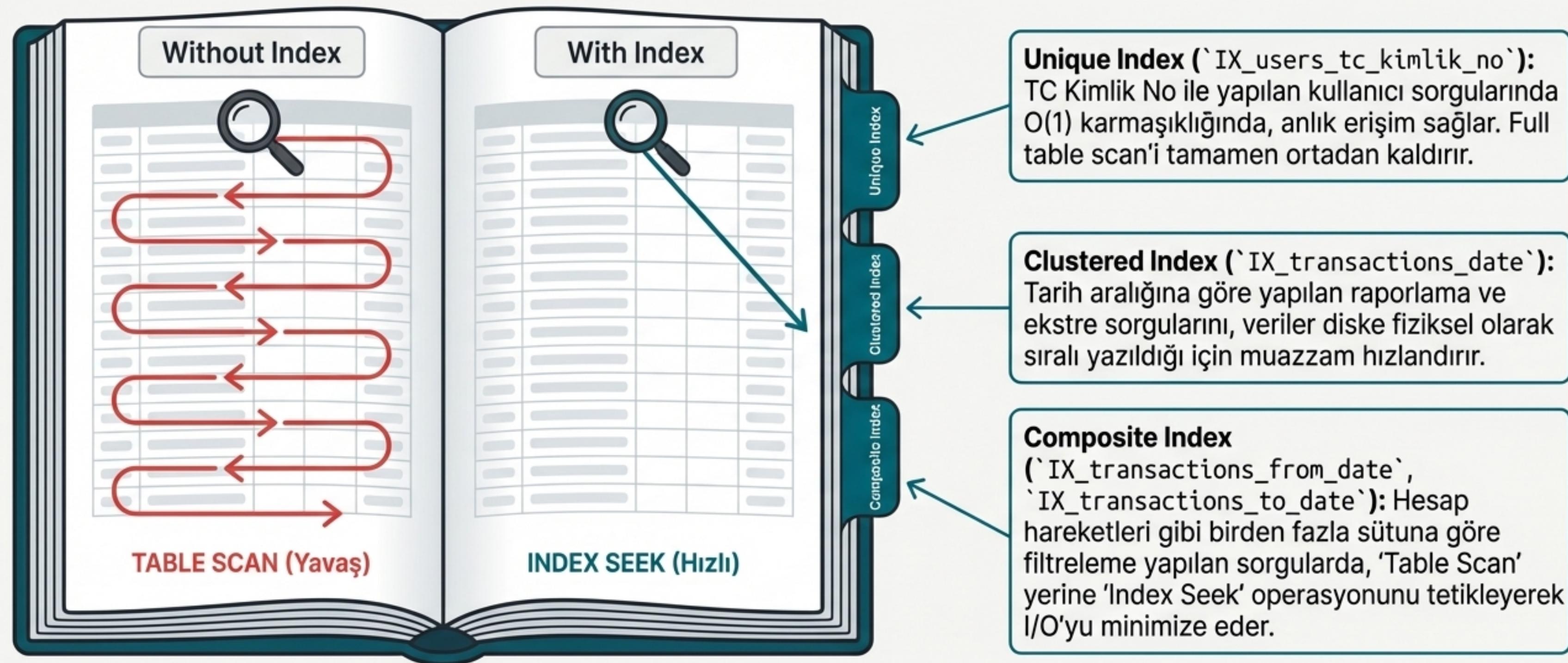


Soft Delete Mekanizması

Kritik veriler (`users`, `accounts`) `is_active` ve `deleted_at` alanları ile fiziksel olarak silinmez. Bu, denetim (audit) ve yasal uyumluluk (KVKK) için zorunludur.



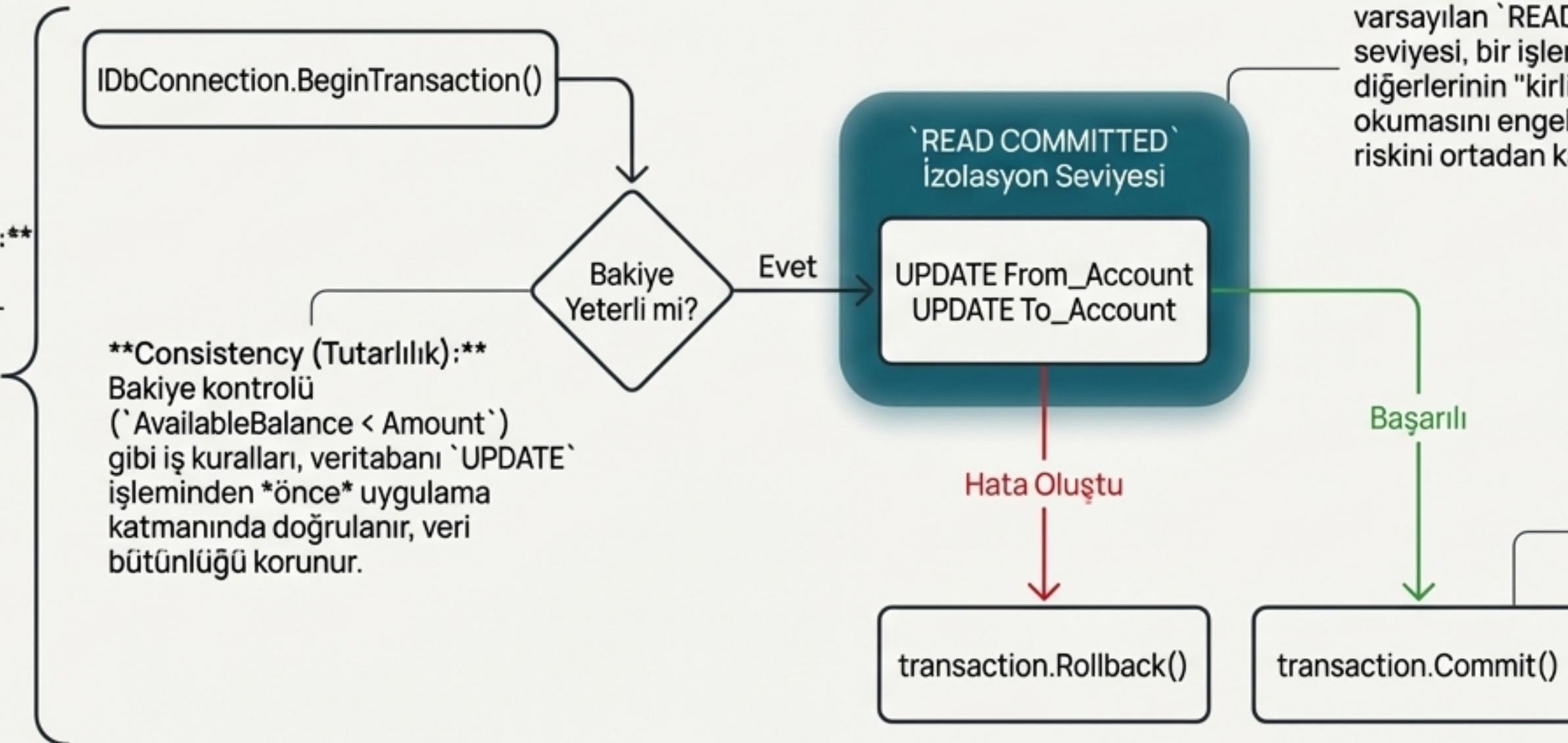
Sorgu Optimizasyonu: Stratejik Veritabanı İndekslemesi



Stratejik Sonuç: Yüksek hacimli tablolarda dahi en yaygın sorgu senaryolarında 'Table Scan' operasyonlarından kaçınarak, sorgu yanıt sürelerini milisaniyeler seviyesinde tutmak.

Finansal Bütünlüğün Garantisi: ACID Uyumlu Para Transferi

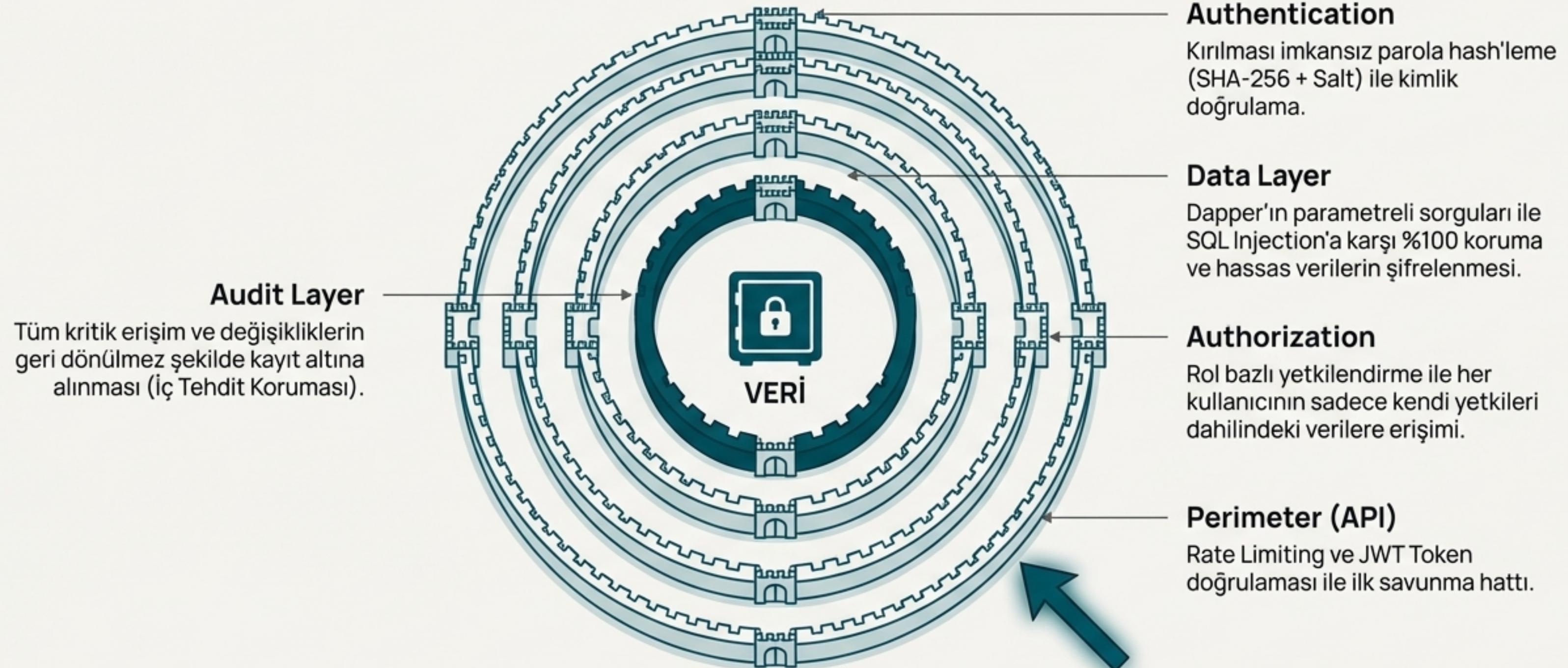
****Atomicity (Bölünmezlik):****
Her para transferi
'`IDbConnection.BeginTransaction()`' bloğu içinde
yürütülür.
Herhangi bir hata
durumunda
'`transaction.Rollback()`'
ile tüm işlem geri alınır.
"Ya hep ya hiç" prensibi.



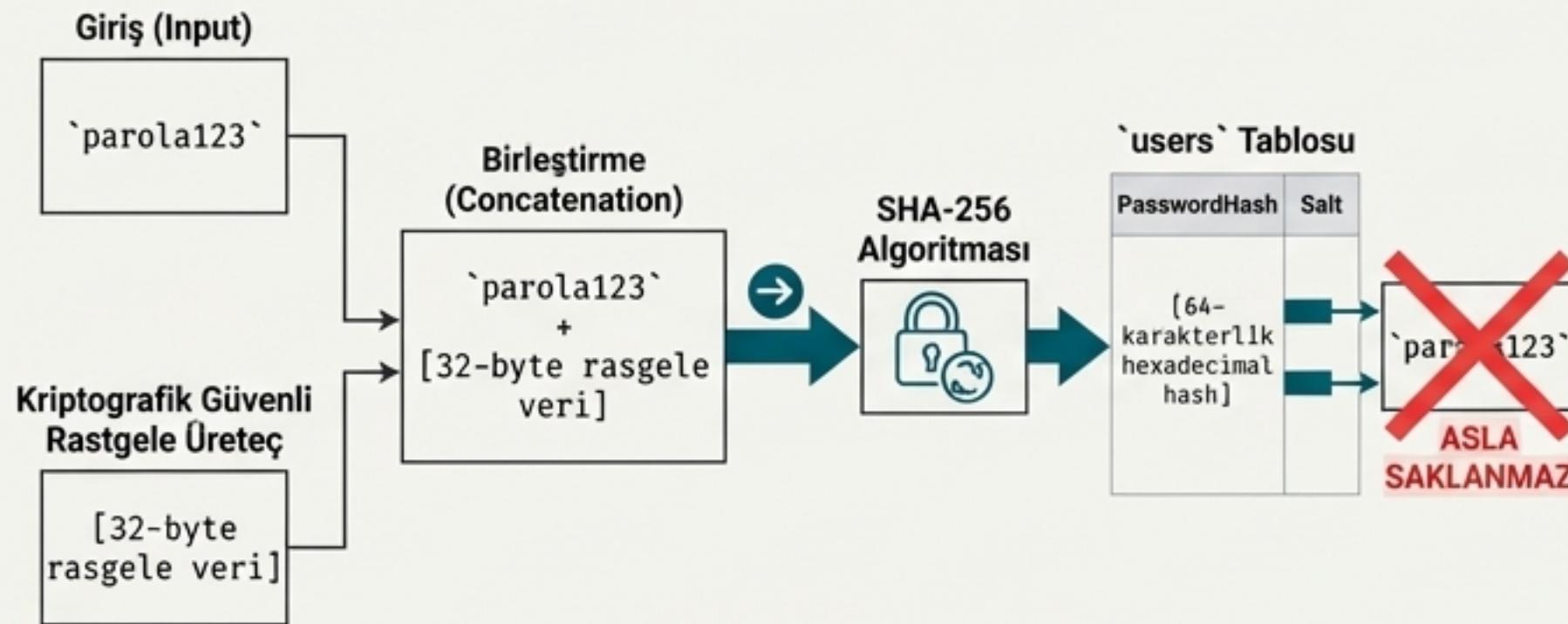
****Isolation (İzolasyon):**** SQL Server'in varsayılan 'READ COMMITTED' izolasyon seviyesi, bir işlem tamamlanmadan diğerlerinin "kirli veri" (dirty read) okumasını engeller ve "race condition" riskini ortadan kaldırır.

****Durability (Kalıcılık):**** '`transaction.Commit()`' başarılı olduğu anda, yapılan değişiklıkların sistem çokse bile kalıcı olacağı veritabanı tarafından garanti edilir.

Güvenlik Mimarisi: Defense in Depth (Derinlemesine Savunma)



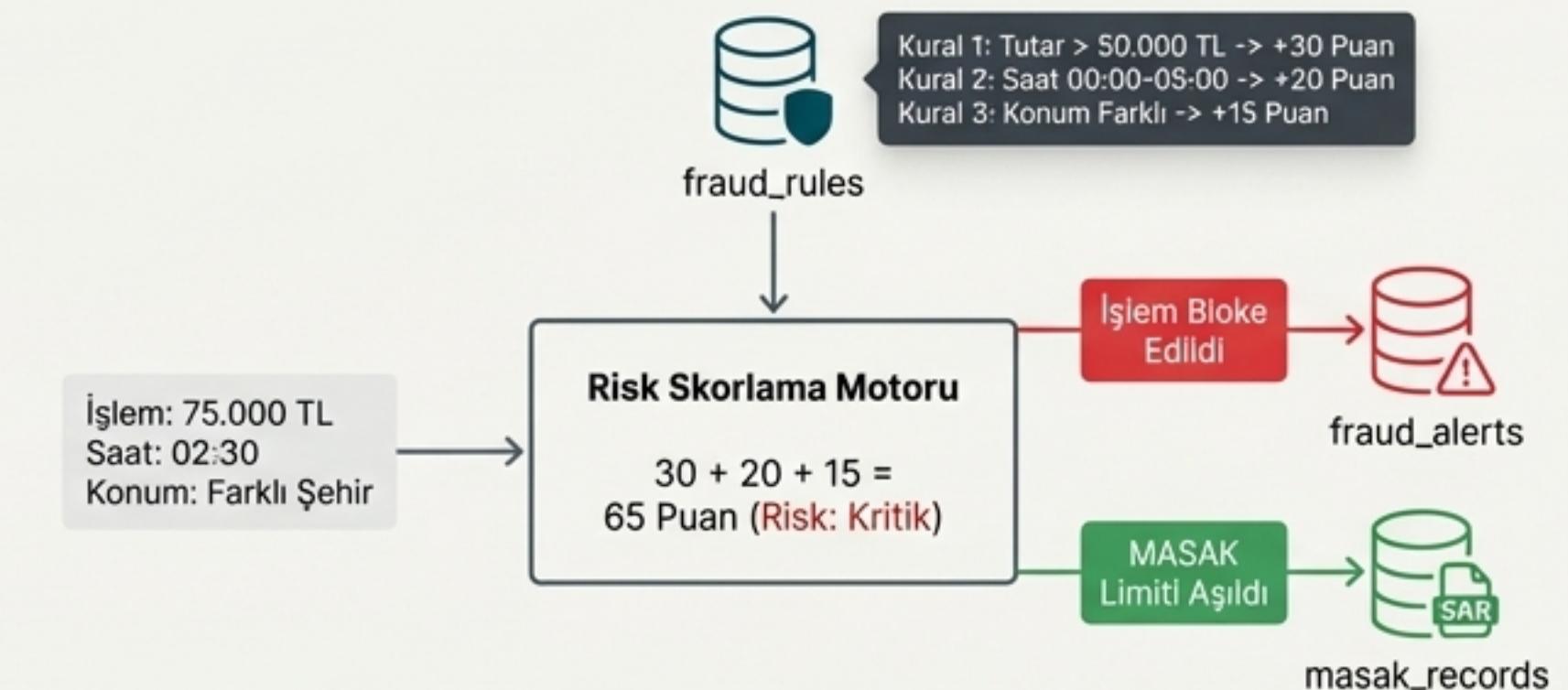
Kimlik Doğrulama Güvenliği: SHA-256 ve Salt (Tuzlama) Tekniği



- **Hashing Algoritması:** Endüstri standartı olan **SHA-256**, tek yönlü ve çarşılmaya (collision-resistant) dayanıklı bir hash algoritmasıdır. Paroladan hash üretilabilir, ama hash'ten parola üretilemez.
- **Salt (Tuzlama):** Her kullanıcı için özel olarak üretilen, kriptografik olarak güvenli, rastgele ve 32-byte uzunluğunda bir 'Salt' değeri, parola ile birleştirildikten sonra hash'lenir.
- **Depolama:** Veritabanında parolanın kendisi değil, PasswordHash(SHA256(parola + salt)) ve 'salt' değeri ayrı ayrı saklanır.
- **Koruma:** Bu yöntem, veritabanı sızdırılsa bile "Rainbow Table" saldırısını imkansız hale getirir, çünkü her kullanıcının hash'i kendine özgü 'salt' yüzünden farklıdır. Aynı parolaya sahip iki kullanıcının bile hash'leri farklıdır.

Dinamik Kural Motoru: Fraud Tespiti ve MASAK Uyumluluğu

- **Dinamik Kural Tanımlama (`fraud_rules`):** Riskli işlem paternleri (örn: gece yarısı yüksek meblağlı transfer) sistem çalışırken kod değişikliği gerektirmeden veritabanından dinamik olarak tanımlanabilir.
- **Gerçek Zamanlı Risk Skorlama:** Her işlem, bu aktif kurallara göre anlık olarak analiz edilir ve bir risk puanı alır.
- **Otomatik Aksiyon:** Risk puanı belirli bir eşigi (**Critical** seviyesi) aşan işlemler otomatik olarak bloke edilir ve **`fraud_alerts`** tablosuna alarm olarak kaydedilir.
- **MASAK Raporlama (`masak_records`):** Yasal limitleri (örn: 50.000 TL) aşan şüpheli işlemler, denetime hazır formatta otomatik olarak işaretlenir ve saklanır (Suspicious Activity Report - SAR).



Tam Denetlenebilirlik: Değişiklik Veri Yakalama ve Erişim Logları

- Değişiklik Veri Yakalama** (`audit_logs`): Kritik tablolardaki her `INSERT`, `UPDATE`, `DELETE` işleminin `OldValue` ve `NewValue`'sunu JSON formatında saklar. Bu, "kim, neyi, ne zaman, neyden neye değiştirdi?" sorusuna kesin cevap verir.
- Hassas Veri Erişim Kaydı** (`data_access_log`): Bir personelin müşteri bakiyesi gibi hassas bir veriye neden ve ne zaman eriştiği, IP adresi ile birlikte geri dönülmey şekilde kayıt altına alınır. Bu, iç tehditlere karşı en önemli savunmadır.
- PCI-DSS Simülasyonu** (`pci_audit_log`): Şifrelenmiş kredi kartı verisine yapılan her erişim, özel bir tabloda denetlenir.
- NoSQL Esnekliği**: Logların JSON formatında tutulması, ilişkisel bir veritabanı üzerinde NoSQL benzeri esnek sorgulama ve analiz imkanı sunar.

Denetim Kaydı: #10543

Timestamp: 2023-10-27 15:30:02

User: admin@bank.com

IP Address: 212.58.10.5

Action: UPDATE

Table: accounts

Record ID: 101

OldValue

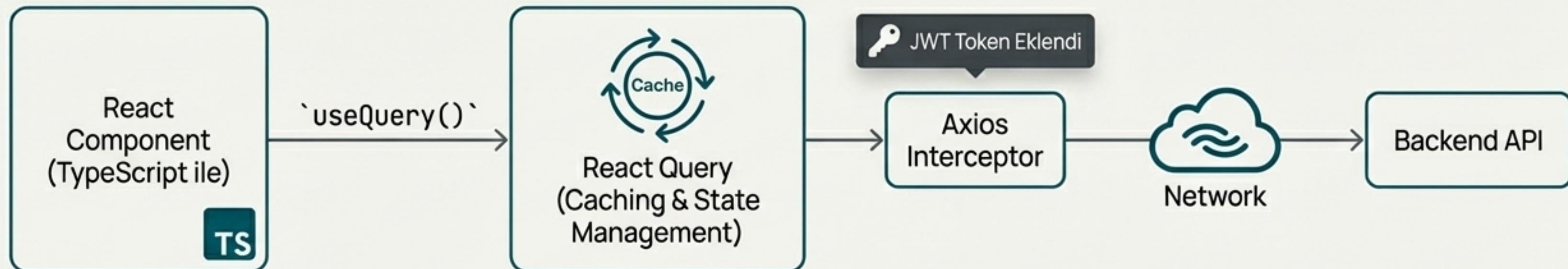
```
{  
    "balance": 1500.00,  
    "status": "active"  
}
```

NewValue

```
{  
    "balance": 1250.00,  
    "status": "active"  
}
```

Frontend Mimaris: React ve TypeScript ile Tip-Güvenli Arayüz

- **Teknoloji Yığını:** React, TypeScript, React Query, Axios.
- **Tip Güvenliği (Type Safety):** TypeScript, API'den gelen verinin yapısını derleme zamanında (compile-time) doğrulayarak "undefined is not a function" gibi çalışma zamanı (runtime) hatalarını büyük ölçüde ortadan kaldırır.
- **Veri Yönetimi (React Query):** Sunucu verisini (`server state`) akıllıca önbelleğe alarak (`caching`), gereksiz ağ isteklerini önler, arayüzün tepkisellliğini artırır ve kod karmaşıklığını azaltır.
- **Güvenli API İletişimi:** Axios Interceptor'ları, her giden isteğe JWT token'ını otomatik ve güvenli bir şekilde ekleyerek kimlik doğrulama sürecini merkezileştirir.



Stratejik Kazanımlar: Teknik Mükemmeliyetin İş Değerine Dönüşümü



Ölçeklenebilirlik

Dapper, Asenkron mimari ve doğru indeksleme sayesinde, donanım eklemesiyle doğrusal olarak artan işlem kapasitesi. Anlık trafik artışlarına hazır bir yapı.



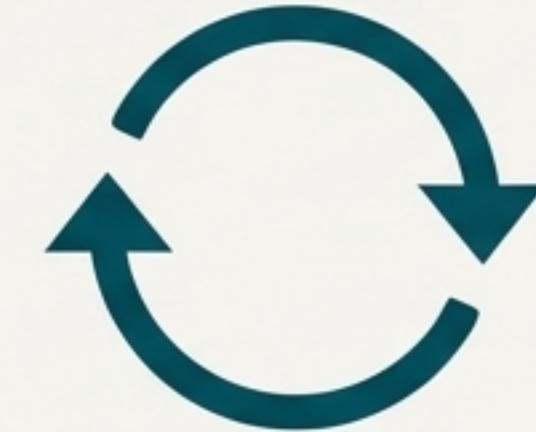
Güvenilirlik

ACID uyumlu atomik işlemler ve veritabanı bütünlük kuralları sayesinde sıfır veri kaybı riski ve finansal tutarlılığın garantisidir.



Güvenlik ve Uyumluluk

Çok katmanlı 'Defense in Depth' mimarisi ve dinamik kural motoru ile hem siber tehditlere hem de MASAK, KVKK gibi yasal gerekliliklere tam uyum.



Sürdürülebilirlik

Clean Architecture sayesinde düşük bakım maliyeti, yüksek test edilebilirlik ve yeni özelliklerin sisteme risk oluşturmadan, hızla entegre edilebilmesi.

Gelecek Vizyonu: Sağlam Temeller Üzerinde Evrimleşen Mimari



Bir Konsept Kanıtından Daha Fazlası: Üretime Üretime Hazır Bir Finansal Çekirdek

- Bu sistem, endüstri standartlarını karşılayan, canlıya alınmaya aday bir FinTech çekirdeğidir.
- Performans, güvenlik ve veri bütünlüğü arasındaki dengeyi, bilinçli ve **kanıta dayalı** mühendislik prensipleriyle kurar.
- Her teknik karar, somut bir problemi çözmek ve ölçülebilir bir fayda sağlamak amacıyla alınmıştır.
- Bu çalışma, hem teknik yatırımcılar için ölçülebilir bir değer önerisi, hem de akademik dünya için bir ‘best practice’ vaka çalışması sunmaktadır.