Data Architecture

Data Architecture

Version: 1.0 **Date:** 2026-02-21 **Status:** Approved **Owner:** Database Architect

---

Overview

Drop's data architecture supports a PSD2 pass-through fintech application with two core functions: international remittances and QR merchant payments. The system manages 19 tables across 5 domains, backed by PostgreSQL 16 (all environments: development, CI, staging, production) via Drizzle ORM. See ADR-014.

Drop never holds customer funds. The `bank_accounts.balance` field is a cached AISP read from the user's real bank account -- not a Drop-held balance.

Domain Model

The 19 tables are organized into 5 logical domains:

```
erDiagram
    %% User Domain
    users ||--o{ bank_accounts : "links"
    users ||--o{ cards : "owns"
    users ||--o{ recipients : "saves"
    users ||--o{ transactions : "initiates"
    users ||--o{ sessions : "authenticates"
    users ||--o{ notifications : "receives"
    users ||--|| settings : "configures"
    users ||--o{ spending_limits : "sets"

    %% Financial Domain
    transactions }o--o| recipients : "sends to"
```

```
transactions }o--o| merchants : "pays"
users ||--o{ merchants : "registers as"
cards ||--o{ spending_limits : "limited by"

%% KYC/AML Domain
users ||--o{ screening_results : "screened"
users ||--o{ aml_alerts : "flagged"
aml_alerts ||--o{ str_reports : "escalated to"
transactions ||--o{ aml_alerts : "triggers"

%% GDPR Domain
users ||--o{ consents : "grants"
users ||--o{ data_access_requests : "submits"
users ||--o{ complaints : "files"

%% System Domain
users ||--o{ audit_log : "generates"

users {
    text id PK "usr_ prefix"
    text email UK
    text password_hash
    text first_name
    text last_name
    text phone
    text date_of_birth
    text kyc_status "pending|approved|rejected"
    text role "user|merchant"
    text risk_level "low|medium|high"
    text pep_status
    text national_id_hash
    text deleted_at
    text created_at
}

bank_accounts {
    text id PK
    text user_id FK
    text bank_name
    text account_number
```

```
    text iban
    integer balance "cached AISP read"
    text currency
    integer is_primary
}

transactions {
    text id PK
    text user_id FK
    text type "remittance|qr_payment"
    text status "processing|completed|failed"
    integer amount
    text currency
    integer fee
    text recipient_id FK
    text merchant_id FK
    real exchange_rate
    text idempotency_key UK
}

recipients {
    text id PK
    text user_id FK
    text name
    text country
    text currency
    text bank_account
}

merchants {
    text id PK
    text user_id FK
    text business_name
    text org_number UK
    text bank_account
    real fee_rate
    text qr_hmac_key
}

sessions {
```

```
    text id PK
    text user_id FK
    text token_hash
    text expires_at
    integer revoked
}

notifications {
    text id PK
    text user_id FK
    text type
    text title
    text body
    integer read
}

settings {
    text user_id PK
    text currency
    text language
    integer push_enabled
    integer email_enabled
}

exchange_rates {
    integer id PK
    text from_currency
    text to_currency
    real rate
}

cards {
    text id PK
    text user_id FK
    text type "virtual|physical"
    text last_four
    text status "active|frozen|cancelled"
}

spending_limits {
```

```
    text id PK
    text user_id FK
    text card_id FK
    text limit_type
    integer amount
}

audit_log {
    text id PK
    text user_id FK
    text action
    text resource_type
    text resource_id
    text ip_address
}

aml_alerts {
    text id PK
    text user_id FK
    text alert_type
    text severity "low|medium|high|critical"
    text transaction_id FK
    text status "open|investigating|resolved|escalated|filed"
}

str_reports {
    text id PK
    text user_id FK
    text alert_id FK
    text report_type
    text status "draft|submitted|acknowledged"
}

screening_results {
    text id PK
    text user_id FK
    text screening_type "pep|sanctions|adverse_media"
    text result "clear|match|potential_match|error"
}
```

```
    consents {
        text id PK
        text user_id FK
        text consent_type
        integer granted
        text ip_address
    }

    data_access_requests {
        text id PK
        text user_id FK
        text request_type "export|erasure|rectification|restriction"
        text status "pending|processing|completed|rejected"
    }

    complaints {
        text id PK
        text user_id FK
        text category
        text subject
        text status "received|investigating|resolved|escalated"
    }

    rate_limits {
        text key PK
        integer count
        integer reset_at
    }
```

Domain Groupings

1. User Domain (4 tables)

Core identity, authentication, and preferences.

Table	Purpose	Record Growth
`users`	User accounts with KYC/AML fields, BankID identity	1 per registered user

Table	Purpose	Record Growth
`settings`	Per-user preferences (currency, language, notifications)	1 per user (1:1)
`sessions`	JWT session tracking with revocation support	Multiple per user, prunable
`notifications`	In-app notification delivery	High volume, prunable

Key relationships: `users` is the central entity. Every other user-scoped table references `users(id)` via foreign key. `settings` has a 1:1 relationship using `user_id` as its primary key.

2. Financial Domain (7 tables)

Transaction processing, bank account linkage, exchange rates, and payment cards.

Table	Purpose	Record Growth
`transactions`	All financial operations (remittance + QR payment)	High volume, append-only
`bank_accounts`	Linked bank accounts with cached AISP balance	Few per user
`recipients`	Saved remittance recipients	Few per user
`merchants`	Registered merchant profiles	1 per merchant user
`exchange_rates`	NOK-to-foreign currency rates	6 corridor records, updated periodically
`cards`	Virtual/physical payment cards (FUTURE, feature-flagged)	Few per user
`spending_limits`	Card spending limits (FUTURE)	Few per card

Key relationships: `transactions` polymorphically references either `recipients` (for remittances) or `merchants` (for QR payments) -- never both simultaneously. `bank_accounts.balance` is a cached read-only value from AISP, not a Drop-held balance.

3. KYC/AML Domain (3 tables)

Anti-money laundering monitoring and regulatory screening.

Table	Purpose	Record Growth
`aml_alerts`	Flagged suspicious transaction patterns	Event-driven, low volume
`str_reports`	Suspicious Transaction Reports filed with Økokrim	Rare, legally retained

Table	Purpose	Record Growth
`screening_results`	PEP/sanctions/adverse media screening results	Per user, periodic rescreens

Key relationships: `aml_alerts` links to a triggering `transaction`. `str_reports` escalates from an `aml_alert`. Both reference the `user` under investigation.

4. GDPR/Compliance Domain (3 tables)

Data subject rights, consent management, and complaint handling.

Table	Purpose	Record Growth
`consents`	GDPR consent records (terms, privacy, marketing, cookies)	Few per user
`data_access_requests`	DSAR tracking (export, erasure, rectification, restriction)	Rare
`complaints`	Customer complaints per Finansavtaleloven section 3-53	Low volume

Key relationships: All reference `users(id)`. Consent withdrawal triggers downstream processing (e.g., marketing opt-out).

5. System Domain (2 tables)

Operational infrastructure for audit trails and rate limiting.

Table	Purpose	Record Growth
`audit_log`	User action audit trail for compliance	Very high volume
`rate_limits`	IP-based rate limiting counters	Ephemeral, auto-cleaned

Key relationships: `audit_log` optionally references `users(id)` (some system events are unauthenticated). `rate_limits` is standalone with no foreign keys.

Data Classification

Each table is classified by sensitivity level for security controls, encryption, and access policies.

Classification uses the 5-level taxonomy defined in [security-architecture.md](): CRITICAL, RESTRICTED, CONFIDENTIAL, INTERNAL, PUBLIC.

Table	Classification	PII	Financial	Compliance	Rationale
`users`	RESTRICTED	Yes	No	Yes	Contains name, email, phone, DOB, national ID hash
`bank_accounts`	RESTRICTED	Yes	Yes	Yes	Bank account numbers, IBAN, cached balance
`transactions`	CONFIDENTIAL	No	Yes	Yes	Financial records, amounts, exchange rates
`recipients`	RESTRICTED	Yes	Yes	No	Names and foreign bank account numbers
`merchants`	CONFIDENTIAL	No	Yes	No	Business details, org numbers, bank accounts
`sessions`	INTERNAL	No	No	No	Token hashes enabling authentication bypass if leaked
`cards`	RESTRICTED	Yes	Yes	Yes	Card last-four, token refs, PINs (FUTURE)
`aml_alerts`	CONFIDENTIAL	No	No	Yes	Regulatory investigation data
`str_reports`	CONFIDENTIAL	No	No	Yes	Filed with Økokrim, legally protected
`screening_results`	CONFIDENTIAL	No	No	Yes	PEP/sanctions match data
`audit_log`	INTERNAL	Partial	No	Yes	IP addresses, user agents, action descriptions
`consents`	RESTRICTED	Partial	No	Yes	IP addresses, consent timestamps
`data_access_requests`	INTERNAL	No	No	Yes	DSAR metadata and download URLs
`complaints`	INTERNAL	No	No	Yes	User-submitted text content

Table	Classification	PII	Financial	Compliance	Rationale
`notifications`	INTERNAL	No	No	No	Display text, no sensitive content
`settings`	INTERNAL	No	No	No	UI preferences only
`exchange_rates`	PUBLIC	No	No	No	Public market data
`spending_limits`	INTERNAL	No	No	No	User-configured limits
`rate_limits`	INTERNAL	No	No	No	Ephemeral IP counters

Data Flow

Remittance Flow

```
sequenceDiagram
    participant U as User (Mobile/Web)
    participant API as Hono API / Next.js
    participant Auth as Auth Middleware
    participant DB as Database (SQLite/PostgreSQL)
    participant AISP as Open Banking AISP
    participant PISP as Open Banking PISP

    U->>API: POST /transactions/remittance
    API->>Auth: Verify JWT + session
    Auth->>DB: SELECT sessions WHERE token_hash = ? AND revoked = 0
    Auth-->>API: userId, role

    API->>DB: SELECT * FROM recipients WHERE id = ? AND user_id = ?
    DB-->>API: Recipient (country, currency, bank_account)

    API->>DB: SELECT rate FROM exchange_rates WHERE to_currency = ?
    DB-->>API: Exchange rate

    API->>DB: SELECT * FROM bank_accounts WHERE user_id = ? AND is_primary = 1
    DB-->>API: Bank account (balance check)
```

```
    Note over API,DB: Atomic transaction begins
    API->>DB: UPDATE bank_accounts SET balance = balance - ? WHERE balance >= ?
    API->>DB: INSERT INTO transactions (type='remittance', status='processing', ...)
    API->>DB: INSERT INTO audit_log (action='transaction.create', ...)
    API->>DB: INSERT INTO notifications (type='transaction', ...)
    Note over API,DB: Atomic transaction commits

    API->>PISP: Initiate payment from user's bank (production)
    API-->>U: 201 { transaction details, ETA }
```

QR Payment Flow

```
sequenceDiagram
    participant U as User (Mobile)
    participant API as Hono API / Next.js
    participant DB as Database
    participant M as Merchant

    U->>U: Scan QR code (drop://pay/{merchantId})
    U->>API: POST /transactions/qr-payment { merchantId, amount }
    API->>DB: Verify JWT session
    API->>DB: SELECT * FROM merchants WHERE id = ?
    DB-->>API: Merchant details (fee_rate, bank_account)

    API->>DB: SELECT * FROM bank_accounts WHERE user_id = ? AND is_primary = 1
    DB-->>API: Primary bank account

    Note over API,DB: Atomic transaction
    API->>DB: UPDATE bank_accounts SET balance = balance - (amount + fee)
    API->>DB: INSERT INTO transactions (type='qr_payment', status='completed')
    API->>DB: INSERT INTO audit_log (action='qr_payment.create')
    API->>DB: INSERT INTO notifications (title='Betaling registrert')
    Note over API,DB: Commit

    API-->>U: 201 { payment confirmation }
```

Caching Strategy

Data	Cache Location	TTL	Invalidation	Rationale
Exchange rates	`exchange_rates` table	Updated periodically (external feed in production)	Table update replaces rows	Rates change infrequently; per-request DB lookup is sufficient
Bank account balance	`bank_accounts.balance` column	`balance_synced_at` tracks freshness	Re-synced via AISP on dashboard load	Cached AISP read; Drop never modifies this value except through sync
User session validity	`sessions` table lookup	Until `expires_at`	Set `revoked = 1` on logout	Every authenticated request checks session table
Rate limit counters	`rate_limits` table	`reset_at` Unix timestamp (60s window)	Auto-cleaned every 100 rate limit checks	Expired entries deleted in `middleware/rate-limit.ts`
JWT payload	In-cookie (client-side)	7d (all clients)	Cookie cleared on logout, session revoked server-side	Stateless token; server validates against sessions table
Feature flags	In-memory (process)	Process lifetime	Restart or env var change	Read from environment variables at startup

No external cache layer (Redis/Memcached): At current scale, PostgreSQL 16 with Drizzle ORM handles the expected query volume without an external cache. A caching layer will be evaluated when query volume exceeds PostgreSQL connection pool capacity (max 20 connections per App Runner instance).

Data Access Layer (Drizzle ORM)

> **NOTE:** The dual-driver abstraction (`db.ts`, `USE_PG`) was removed per ADR-014 (2026-03-03). The data access layer is now Drizzle ORM exclusively.

The database access layer (`src/shared/db/schema.ts` + Drizzle ORM) provides type-safe access to PostgreSQL 16:

Pattern	How

SELECT queries	`db.select().from(table).where(...)`
Single row SELECT	`db.select().from(table).limit(1)`
INSERT/UPDATE/DELETE	`db.insert(table).values(...)`, `db.update()`, `db.delete()`
Upsert	`db.insert(table).values(...).onConflictDoUpdate(...)`
Atomic operations	`db.transaction(async (tx) => { ... })`
Row locking	`db.select().from(table).for('update')`
Raw SQL escape hatch	`db.execute(sql\`SELECT ...\`)`

Connection string: `DATABASE_URL=postgresql://...` (required in all environments).

Cross-References

- **Schema details:** DATABASE-SCHEMA.md
- **Database design rationale:** database-design.md
- **Migration strategy:** migration-strategy.md
- **Data lifecycle and GDPR:** data-lifecycle.md
- **Audit architecture:** audit-architecture.md
- **Indexing strategy:** indexing-strategy.md
- **API reference:** API-REFERENCE.md
- **Security architecture:** SECURITY-ARCHITECTURE.md
- **Authentication:** AUTHENTICATION.md

Revision #9
Created 2026-02-21 05:58:49 UTC by John
Updated 2026-05-23 10:51:42 UTC by John