Database Design

Database Design

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

---

Design Philosophy

Drop's database schema is designed around three principles:

1. **Simplicity over abstraction.** 19 tables for a well-scoped fintech app. No generic "entities" table, no EAV patterns. Each table maps to a clear domain concept.
2. **Compliance by design.** 7 of 19 tables exist solely for regulatory requirements (GDPR, AML, PSD2). They were added as a compliance infrastructure layer, not retrofitted.
3. **PostgreSQL-native.** Schema is defined in Drizzle ORM (`src/shared/db/schema.ts`) targeting PostgreSQL 16. PostgreSQL-native features (JSONB, `FOR UPDATE`, `RETURNING`, arrays) are available and used where beneficial. See ADR-014.

Why 19 Tables

The table count reflects the actual domain:

- **12 core tables** cover the business logic: users, their bank accounts, recipients, merchants, transactions, exchange rates, cards, sessions, notifications, settings, spending limits, and rate limits.
- **7 compliance tables** were added as a single compliance infrastructure layer: audit logging, AML alerts, STR reports, sanctions screening, consent tracking, data access requests, and complaints.

No table is redundant. No table combines unrelated concerns.

Complete Schema ERD

```
erDiagram
    users ||--|| settings : "1:1 preferences"
    users ||--o{ bank_accounts : "1:N linked accounts"
    users ||--o{ cards : "1:N payment cards"
    users ||--o{ recipients : "1:N saved recipients"
    users ||--o{ transactions : "1:N financial ops"
    users ||--o{ sessions : "1:N auth sessions"
    users ||--o{ notifications : "1:N alerts"
    users ||--o{ spending_limits : "1:N limits"
    users ||--o{ merchants : "1:N merchant profiles"
    users ||--o{ audit_log : "1:N audit entries"
    users ||--o{ aml_alerts : "1:N AML flags"
    users ||--o{ str_reports : "1:N STR filings"
    users ||--o{ screening_results : "1:N screenings"
    users ||--o{ consents : "1:N consents"
    users ||--o{ data_access_requests : "1:N DSARs"
    users ||--o{ complaints : "1:N complaints"

    transactions }o--o| recipients : "remittance target"
    transactions }o--o| merchants : "QR payment target"
    transactions ||--o{ aml_alerts : "triggers alert"
    aml_alerts ||--o{ str_reports : "escalates to STR"
    cards ||--o{ spending_limits : "card-level limits"

    users {
        text id PK "usr_ + 16 hex"
        text email UK "NOT NULL"
        text password_hash "NOT NULL, default EIDONLY"
        text auth_provider "default bankid"
        text first_name "NOT NULL"
        text last_name "NOT NULL"
        text phone "nullable"
        text date_of_birth "nullable"
        text kyc_status "CHECK pending|approved|rejected"
        text role "CHECK user|merchant"
        text risk_level "CHECK low|medium|high"
        text pep_status "CHECK not_checked|clear|match|pending_review"
        integer sanctions_cleared "default 0"
        text kyc_method "CHECK bankid|document|simplified"
        text kyc_verified_at "nullable"
```

```
    text national_id_hash "nullable, indexed WHERE NOT NULL"
    text deleted_at "nullable, soft delete"
    text created_at "default datetime now"
}

transactions {
    text id PK
    text user_id FK "NOT NULL"
    text type "CHECK remittance|qr_payment"
    text status "CHECK processing|completed|failed"
    integer amount "NOT NULL, in minor units"
    text currency "default NOK"
    integer fee "default 0"
    text recipient_id FK "nullable"
    text merchant_id FK "nullable"
    integer send_amount "nullable"
    text send_currency "nullable"
    integer receive_amount "nullable"
    text receive_currency "nullable"
    real exchange_rate "nullable"
    text purpose_code "nullable"
    text idempotency_key "UNIQUE WHERE NOT NULL"
    text created_at "default datetime now"
    text completed_at "nullable"
}

bank_accounts {
    text id PK
    text user_id FK "NOT NULL"
    text bank_name "NOT NULL"
    text account_number "NOT NULL"
    text iban "nullable"
    integer balance "default 0, cached AISP"
    text balance_synced_at "nullable"
    text currency "default NOK"
    integer is_primary "default 0"
    text connected_at "default datetime now"
}

merchants {
```

```
        text id PK
        text user_id FK "NOT NULL"
        text business_name "NOT NULL"
        text org_number "UNIQUE NOT NULL"
        text address "nullable"
        text bank_account "NOT NULL"
        real fee_rate "default 0.01"
        text status "default active"
        text qr_hmac_key "NOT NULL, random 32 bytes"
        text created_at "default datetime now"
    }

    recipients {
        text id PK
        text user_id FK "NOT NULL"
        text name "NOT NULL"
        text country "NOT NULL"
        text currency "NOT NULL"
        text bank_account "NOT NULL"
        text bank_name "nullable"
        text created_at "default datetime now"
    }
```

Table-by-Table Design Rationale

Core Tables

`users`

Normalization: 3NF. All columns are functionally dependent on the primary key.

Design decisions:

- `id` uses `usr_` prefix + 16 hex chars for readability and collision avoidance across distributed systems.
- `password_hash` defaults to `'EIDONLY'` sentinel value -- BankID-only users have no password. This avoids nullable password fields that complicate auth logic.
- `auth_provider` tracks how the user registered (`bankid`). Supports future Vipps Login without schema changes.

- `national_id_hash` stores SHA-256 of Norwegian fodselsnummer. Enables user deduplication across auth providers without storing the raw national ID.
- `deleted_at` enables soft delete for GDPR erasure while retaining records for AML legal obligations (5-year retention).
- `risk_level`, `pep_status`, `sanctions_cleared` are denormalized onto the user for fast access during transaction authorization -- these are checked on every financial operation.
- KYC fields (`kyc_status`, `kyc_method`, `kyc_verified_at`) are on the user table rather than a separate KYC table because there is a 1:1 relationship and the fields are accessed on every authenticated request.

`transactions`

Normalization: 3NF with intentional denormalization.

Design decisions:

- `amount`, `fee`, `send_amount`, `receive_amount` are stored as integers in minor units (ore for NOK, para for RSD, etc.) to avoid floating-point precision issues.
- Polymorphic reference: `recipient_id` is set for remittances, `merchant_id` for QR payments. Never both. This avoids a separate join table for a simple either/or relationship.
- `exchange_rate` is denormalized (snapshot at transaction time) because rates change. The rate at execution time must be preserved for audit and dispute resolution.
- `idempotency_key` with a unique partial index (`WHERE idempotency_key IS NOT NULL`) prevents duplicate transaction submission without requiring every transaction to have a key.
- `purpose_code` supports remittance regulatory requirements (some corridors require a transfer purpose).
- `completed_at` is separate from `created_at` to track processing duration.

`bank_accounts`

Normalization: 3NF.

Design decisions:

- `balance` is a cached read-only value from AISP, not a Drop-held balance. This is the most important design detail in the entire schema.
- `balance_synced_at` tracks when the balance was last refreshed from the bank via Open Banking.
- `is_primary` flag determines which account is used for transactions by default (1 = primary, 0 = secondary).
- No unique constraint on `account_number` because the same bank account could theoretically appear under different user records (shared accounts).

`recipients`

Normalization: 3NF.

Design decisions:

- Scoped to user (`user_id` FK) -- recipients are private, not shared.
- `country` and `currency` stored as free text validated at the API layer (not as FK to a countries table). This avoids over-engineering for 5-6 supported corridors.
- `bank_account` stores the full foreign account number. Format varies by country (IBAN for EU, local format for others).

`merchants`

Normalization: 3NF.

Design decisions:

- `org_number` is UNIQUE -- one merchant registration per Norwegian organization number (9 digits).
- `qr_hmac_key` is generated server-side (`hex(randomblob(32))`) for QR code integrity verification. Each merchant gets a unique key.
- `fee_rate` defaults to `0.01` (1%). Stored per merchant to allow variable pricing in the future.
- `user_id` FK links the merchant to the user who registered it. A user's role is upgraded to `merchant` upon registration.

`exchange_rates`

Normalization: 3NF.

Design decisions:

- `id` uses `INTEGER PRIMARY KEY AUTOINCREMENT` -- the only auto-increment ID in the schema. Exchange rates are system-managed, not user-created, so prefixed IDs are unnecessary.
- Only stores NOK-to-X rates (6 corridors). Inverse rates are calculated at runtime.
- No historical rate tracking in this table. Transaction records snapshot the rate at execution time.

`sessions`

Normalization: 3NF.

Design decisions:

- `token_hash` stores SHA-256 of the JWT, not the JWT itself. This prevents session hijacking even if the database is compromised.
- `revoked` flag (0/1) enables server-side session invalidation without waiting for JWT expiry.
- Multiple active sessions per user are allowed (different devices).

`settings`

Normalization: 3NF. 1:1 with `users`.

Design decisions:

- `user_id` as PRIMARY KEY enforces the 1:1 relationship at the database level.
- Created lazily on first `GET /api/settings` (INSERT default if not exists).
- Defaults: `currency='NOK'`, `language='nb'`, `push_enabled=1`, `email_enabled=1`.

`notifications`

Normalization: 3NF.

Design decisions:

- `read` flag (0/1) for marking notifications as read in batch.
- No foreign key to the triggering entity (transaction, system event) -- `type` field categorizes the notification source.
- Designed for high volume with eventual cleanup (no retention policy yet).

`cards` (FUTURE)

Normalization: 3NF.

Design decisions:

- Feature-flagged. Table exists in schema but endpoints return 404 when disabled.
- Only stores `last_four` and `token_ref` -- never full card number or CVV (PCI-DSS compliance).
- `pin_hash` added via runtime migration for backward compatibility.
- `status` supports freeze/unfreeze without deletion (`active` -> `frozen` -> `active`).

`spending_limits` (FUTURE)

Normalization: 3NF.

Design decisions:

- `card_id` is nullable -- supports user-level limits (no specific card) or card-level limits.
- `limit_type` values (`daily`, `weekly`, `monthly`, `transaction`) enforced at API level.
- Limits are replaced, not accumulated (PUT semantics per limit type per card).

`rate_limits`

Normalization: 3NF (trivial -- 3 columns).

Design decisions:

- `key` is the IP address (TEXT PK). Simple key-value store.
- `reset_at` is a Unix timestamp. Expired entries are cleaned every 100 rate limit checks in `middleware/rate-limit.ts`.
- Not a "real" domain table -- it is infrastructure. Could be replaced by Redis in production but works fine in SQLite/PostgreSQL.

Compliance Tables

`audit_log`

- `user_id` is nullable because some audit events occur before authentication (e.g., failed login attempts).
- `resource_type` + `resource_id` enable generic resource tracking without polymorphic FKs.
- `details` is a TEXT field (JSON string) for flexible event-specific data.
- `request_id` for correlating multiple audit entries from a single API request.
- Two indexes: `user_id` and `action` for the primary query patterns. (Note: no `timestamp` index exists in the implementation — only `idx_audit_log_user` and `idx_audit_log_action` are created in `db.ts`.)

`aml_alerts`

- `severity` (low/medium/high/critical) determines investigation priority and escalation timelines.
- `status` workflow: `open` -> `investigating` -> `resolved|escalated|filed`.
- `reviewed_by` and `reviewed_at` track the compliance officer's review.
- `transaction_id` FK links to the specific transaction that triggered the alert.

`str_reports`

- Filed with Okokrim/EFE (Norwegian financial intelligence unit).
- `reference_number` stores the authority-assigned reference after submission.
- `alert_id` links back to the originating AML alert.
- Immutable after `status = 'submitted'` -- regulatory requirement.

`screening_results`

- `screening_type` (pep/sanctions/adverse_media) supports multiple screening categories.
- `provider` tracks which screening service was used (future: Sumsub, Refinitiv, etc.).
- `match_details` stores full match information as TEXT (JSON) for review.
- Multiple results per user (periodic rescreening).

`consents`

- `consent_type` values: `terms`, `privacy`, `marketing`, `cookies_analytics`, `cookies_marketing`.
- `granted` (0/1) with separate `granted_at`/`withdrawn_at` timestamps for full consent lifecycle.

- `ip_address` stored as proof of consent action per GDPR requirements.

`data_access_requests`

- `request_type` covers GDPR data subject rights: export (Art. 15), erasure (Art. 17), rectification (Art. 16), restriction (Art. 18).
- `download_url` for data export files (temporary signed URLs).
- `status` workflow: `pending` -> `processing` -> `completed|rejected`.

`complaints`

- Required by Finansavtaleloven section 3-53 (15 business day response requirement).
- `category` (transaction/service/fees/privacy/technical/other) for routing and reporting.
- `resolution` text field filled when complaint is resolved.
- `resolved_at` timestamp for SLA compliance tracking.

Constraint Inventory

Table	Constraint Type	Column(s)	Value
`users`	PRIMARY KEY	`id`	-
`users`	UNIQUE	`email`	-
`users`	NOT NULL	`email`, `password_hash`, `first_name`, `last_name`	-
`users`	CHECK	`kyc_status`	`IN ('pending','approved','rejected')`
`users`	CHECK	`role`	`IN ('user','merchant')`
`users`	CHECK	`risk_level`	`IN ('low','medium','high')`
`users`	CHECK	`pep_status`	`IN ('not_checked','clear','match','pending_review')`
`users`	CHECK	`kyc_method`	`IN ('bankid','document','simplified')`
`transactions`	PRIMARY KEY	`id`	-
`transactions`	NOT NULL	`user_id`, `type`, `amount`	-
`transactions`	FK	`user_id`	`users(id)`
`transactions`	FK	`recipient_id`	`recipients(id)`
`transactions`	FK	`merchant_id`	`merchants(id)`
`transactions`	CHECK	`type`	`IN ('remittance','qr_payment')`

Table	Constraint Type	Column(s)	Value
transactions	CHECK	status	IN ('processing','completed','failed')
transactions	UNIQUE (partial)	idempotency_key	WHERE idempotency_key IS NOT NULL
merchants	PRIMARY KEY	id	-
merchants	UNIQUE	org_number	-
merchants	NOT NULL	user_id, business_name, org_number, bank_account, qr_hmac_key	-
merchants	FK	user_id	users(id)
bank_accounts	PRIMARY KEY	id	-
bank_accounts	NOT NULL	user_id, bank_name, account_number	-
bank_accounts	FK	user_id	users(id)
recipients	PRIMARY KEY	id	-
recipients	NOT NULL	user_id, name, country, currency, bank_account	-
recipients	FK	user_id	users(id)
sessions	PRIMARY KEY	id	-
sessions	NOT NULL	user_id, token_hash, expires_at	-
sessions	FK	user_id	users(id)
cards	PRIMARY KEY	id	-
cards	NOT NULL	user_id, last_four, expiry	-
cards	FK	user_id	users(id)
cards	CHECK	type	IN ('virtual','physical')
cards	CHECK	status	IN ('active','frozen','cancelled')
settings	PRIMARY KEY	user_id	1:1 with users
settings	FK	user_id	users(id)
exchange_rates	PRIMARY KEY	id	AUTOINCREMENT
exchange_rates	NOT NULL	to_currency, rate	-
notifications	PRIMARY KEY	id	-
notifications	NOT NULL	user_id, type, title, body	-

Table	Constraint Type	Column(s)	Value
`notifications`	FK	`user_id`	`users(id)`
`spending_limits`	PRIMARY KEY	`id`	-
`spending_limits`	NOT NULL	`user_id`, `limit_type`, `amount`	-
`spending_limits`	FK	`user_id`, `card_id`	`users(id)`, `cards(id)`
`rate_limits`	PRIMARY KEY	`key`	IP address
`audit_log`	PRIMARY KEY	`id`	-
`audit_log`	NOT NULL	`action`	-
`audit_log`	FK	`user_id`	`users(id)` (nullable)
`aml_alerts`	PRIMARY KEY	`id`	-
`aml_alerts`	NOT NULL	`user_id`, `alert_type`, `severity`	-
`aml_alerts`	FK	`user_id`, `transaction_id`	`users(id)`, `transactions(id)`
`aml_alerts`	CHECK	`severity`	`IN ('low','medium','high','critical')`
`aml_alerts`	CHECK	`status`	`IN ('open','investigating','resolved','escalated','filed')`
`str_reports`	PRIMARY KEY	`id`	-
`str_reports`	NOT NULL	`user_id`, `report_type`	-
`str_reports`	FK	`user_id`, `alert_id`	`users(id)`, `aml_alerts(id)`
`str_reports`	CHECK	`status`	`IN ('draft','submitted','acknowledged')`
`screening_results`	PRIMARY KEY	`id`	-
`screening_results`	NOT NULL	`user_id`, `screening_type`, `result`	-
`screening_results`	FK	`user_id`	`users(id)`
`screening_results`	CHECK	`screening_type`	`IN ('pep','sanctions','adverse_media')`
`screening_results`	CHECK	`result`	`IN ('clear','match','potential_match','error')`
`consents`	PRIMARY KEY	`id`	-
`consents`	NOT NULL	`user_id`, `consent_type`, `granted`	-

Table	Constraint Type	Column(s)	Value
`consents`	FK	`user_id`	`users(id)`
`data_access_requests`	PRIMARY KEY	`id`	-
`data_access_requests`	NOT NULL	`user_id`, `request_type`	-
`data_access_requests`	FK	`user_id`	`users(id)`
`data_access_requests`	CHECK	`request_type`	`IN ('export','erasure','rectification','restriction')`
`data_access_requests`	CHECK	`status`	`IN ('pending','processing','completed','rejected')`
`complaints`	PRIMARY KEY	`id`	-
`complaints`	NOT NULL	`user_id`, `category`, `subject`, `description`	-
`complaints`	FK	`user_id`	`users(id)`
`complaints`	CHECK	`status`	`IN ('received','investigating','resolved','escalated')`

Naming Conventions

Convention	Rule	Examples
Table names	Lowercase, plural, snake_case	`users`, `bank_accounts`, `aml_alerts`
Column names	Lowercase, snake_case	`user_id`, `first_name`, `created_at`
Primary keys	`id` (or entity-name for 1:1 tables like `settings.user_id`)	`users.id`, `settings.user_id`
Foreign keys	`{referenced_table_singular}_id`	`user_id`, `recipient_id`, `card_id`
Timestamps	`{action}_at` suffix	`created_at`, `completed_at`, `withdrawn_at`
Boolean flags	Descriptive name, INTEGER 0/1	`revoked`, `read`, `is_primary`, `granted`
Status columns	`status` with CHECK constraint	`status CHECK(... IN (...))`
Indexes	`idx_{table}_{column}` (exception: `idx_tx_idempotency` uses abbreviation)	`idx_transactions_user`, `idx_sessions_token`, `idx_tx_idempotency`
ID prefixes	3-letter prefix + underscore + 16 hex chars	`usr_`, `tx_`, `ba_`, `mer_`, `rec_`, `ses_`, `con_`, `cmp_`

Normalization Analysis

All tables are in **Third Normal Form (3NF)** with documented exceptions:

Table	NF Level	Deviation	Justification
`users`	3NF	`risk_level`, `pep_status`, `sanctions_cleared` could be in a separate risk_profile table	Accessed on every transaction check. Separate table would add a JOIN to the critical path. 1:1 relationship makes a separate table pointless.
`transactions`	3NF	`exchange_rate`, `send_amount`, `receive_amount` denormalized from exchange_rates	Rate at execution time must be preserved immutably. The `exchange_rates` table changes; the transaction record must not.
`transactions`	3NF	Polymorphic FK (`recipient_id` OR `merchant_id`)	Simple either/or. A join table or STI would add complexity for no benefit at this scale.
`bank_accounts`	3NF	`balance` denormalized from external bank (AISP)	This is a cache, not authoritative data. Drop cannot modify the real balance.
`audit_log`	3NF	`details` is unstructured TEXT (JSON)	Audit events have variable structure. A normalized schema would require dozens of event-specific tables.

No table violates 2NF (no partial key dependencies) because all tables use single-column primary keys.

Cross-References

- **Full schema:** DATABASE-SCHEMA.md
- **Data architecture overview:** data-architecture.md
- **Migration strategy:** migration-strategy.md
- **Dual-driver implementation:** `src/drop-api/src/lib/db.ts`

Revision #11
Created 2026-02-21 05:59:06 UTC by John
Updated 2026-05-23 10:57:12 UTC by John