Database Schema Document

Database Schema Document

> **Project:** {{PROJECT_NAME}} **Database:** {{DATABASE_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}} **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Related Standards

> Cross-references to ALAI DATABASE/ standards. Apply these alongside this schema document.

Document	Path	Purpose
alai-db-standards.md	DATABASE/	Master naming conventions, type standards, column defaults
indexing-strategy.md	DATABASE/	Index design guide — B-tree, GIN, partial, composite
migration-strategy.md	DATABASE/	Migration tooling, zero-downtime patterns, rollback procedures
rls-policy-guide.md	DATABASE/	Row Level Security for multi-tenant schemas
connection-pooling.md	DATABASE/	PgBouncer and application pool configuration

---

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

1. Database Technology & Version

Property	Value
Technology	{{DB_TECHNOLOGY}} (e.g., PostgreSQL, MySQL, MongoDB, DynamoDB)
Version	{{DB_VERSION}}
Hosting	{{HOSTING}} (e.g., AWS RDS, Cloud SQL, self-hosted)
Instance type	{{INSTANCE_TYPE}}
Storage	{{STORAGE_SIZE}} — auto-scaling: {{YES/NO}}
Read replicas	{{N}} replicas in {{REGIONS}}
Connection pooling	{{POOLER}} (e.g., PgBouncer, RDS Proxy) — pool size: {{POOL_SIZE}}
Encoding	UTF-8
Timezone	UTC (all timestamps in UTC)
Migration tool	{{MIGRATION_TOOL}} (e.g., Flyway, Liquibase, Prisma Migrate, custom)

2. ER Diagram

```
erDiagram
    TENANT {
        uuid id PK
        string name
        string slug UK
        string plan
        timestamptz created_at
        timestamptz deleted_at
    }

    USER {
        uuid id PK
        uuid tenant_id FK
        string email UK
        string full_name
```

```
    string password_hash
    string role
    boolean is_verified
    timestamptz last_login_at
    timestamptz created_at
    timestamptz deleted_at
}

{{ENTITY_1}} {
    uuid id PK
    uuid tenant_id FK
    uuid created_by FK
    string {{field_1}}
    string {{field_2}}
    string status
    int version
    timestamptz created_at
    timestamptz updated_at
    timestamptz deleted_at
}

{{ENTITY_2}} {
    uuid id PK
    uuid {{entity_1_id}} FK
    string {{field_1}}
    decimal {{amount_field}}
    timestamptz created_at
}

AUDIT_LOG {
    uuid id PK
    uuid tenant_id FK
    uuid actor_id FK
    string entity_type
    uuid entity_id
    string action
    jsonb old_values
    jsonb new_values
    string ip_address
    timestamptz created_at
```

```
    }

    TENANT ||--o{ USER : "has"
    TENANT ||--o{ {{ENTITY_1}} : "owns"
    USER ||--o{ {{ENTITY_1}} : "creates"
    {{ENTITY_1}} ||--o{ {{ENTITY_2}} : "contains"
    USER ||--o{ AUDIT_LOG : "generates"
```

---

3. Schema Conventions

3.1 Naming Conventions

Element	Convention	Example
Tables	`snake_case`, plural	`user_profiles`, `order_items`
Columns	`snake_case`	`created_at`, `tenant_id`
Primary keys	Always `id` (UUID)	`id UUID PRIMARY KEY`
Foreign keys	`{referenced_table_singular}_id`	`user_id`, `tenant_id`
Indexes	`idx_{table}_{column(s)}`	`idx_users_email`
Unique indexes	`uq_{table}_{column(s)}`	`uq_users_tenant_email`
Enum types	`snake_case`	`user_role`, `order_status`
Junction tables	`{table1}_{table2}` (alphabetical)	`role_permissions`
Sequences	Auto (via `gen_random_uuid()`)	

3.2 Standard Columns (all tables)

Column	Type	Nullable	Default	Description
`id`	`UUID`	NO	`gen_random_uuid()`	Surrogate primary key
`created_at`	`TIMESTAMPTZ`	NO	`NOW()`	Immutable — set on insert
`updated_at`	`TIMESTAMPTZ`	NO	`NOW()`	Auto-updated via trigger
`deleted_at`	`TIMESTAMPTZ`	YES	`NULL`	Soft delete (NULL = active)

Column	Type	Nullable	Default	Description
`version`	`INTEGER`	NO	`1`	Optimistic lock counter

3.3 Data Type Standards

Data	PostgreSQL Type	Notes
Primary keys	`UUID`	`gen_random_uuid()` default
Short strings	`VARCHAR(N)`	Specify max length
Long text	`TEXT`	No length limit
Money / currency	`NUMERIC(19, 4)`	Never FLOAT for money
Booleans	`BOOLEAN`	NOT NULL with DEFAULT
Enums	`custom ENUM type`	Define in migrations
JSON data	`JSONB`	Prefer JSONB over JSON
IP addresses	`INET`	Native IP type
URLs	`TEXT`	Validated at app layer
Timestamps	`TIMESTAMPTZ`	Always with timezone
Dates (no time)	`DATE`	
Durations	`INTERVAL`	

---

4. Tables by Domain

4.1 Identity & Access Domain

Table: `tenants`

Purpose: Top-level multi-tenancy isolation unit. Every resource belongs to a tenant.

Column	Type	Nullable	Default	Constraints	Description
`id`	`UUID`	NO	`gen_random_uuid()`	PK	Tenant identifier
`name`	`VARCHAR(255)`	NO		NOT NULL	Display name
`slug`	`VARCHAR(100)`	NO		UNIQUE, NOT NULL	URL-safe identifier
`plan`	`tenant_plan`	NO	`'free'`	NOT NULL	Subscription plan

Column	Type	Nullable	Default	Constraints	Description
`settings`	`JSONB`	NO	`'{}'::jsonb`		Tenant configuration
`created_at`	`TIMESTAMPTZ`	NO	`NOW()`		
`deleted_at`	`TIMESTAMPTZ`	YES	`NULL`		

```
-- Enum
CREATE TYPE tenant_plan AS ENUM ('free', 'starter', 'pro', 'enterprise');

-- Table
CREATE TABLE tenants (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(255) NOT NULL,
    slug VARCHAR(100) NOT NULL,
    plan tenant_plan NOT NULL DEFAULT 'free',
    settings JSONB NOT NULL DEFAULT '{}',
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    deleted_at TIMESTAMPTZ
);

-- Indexes
CREATE UNIQUE INDEX uq_tenants_slug ON tenants(slug) WHERE deleted_at IS NULL;
```

Table: `users`

Purpose: System users. Authentication identity.

Column	Type	Nullable	Default	Constraints	Description
`id`	`UUID`	NO	`gen_random_uuid()`	PK	
`tenant_id`	`UUID`	NO		FK → `tenants(id)`	Tenant membership
`email`	`VARCHAR(320)`	NO		NOT NULL	Normalized lowercase
`password_hash`	`VARCHAR(255)`	YES	`NULL`		bcrypt/Argon2 hash. NULL for SSO users
`full_name`	`VARCHAR(255)`	NO		NOT NULL	
`role`	`user_role`	NO	`'member'`	NOT NULL	RBAC role
`is_verified`	`BOOLEAN`	NO	`FALSE`	NOT NULL	Email verified

Column	Type	Nullable	Default	Constraints	Description
`last_login_at`	`TIMESTAMPTZ`	YES	`NULL`		
`mfa_enabled`	`BOOLEAN`	NO	`FALSE`	NOT NULL	
`mfa_secret`	`TEXT`	YES	`NULL`		Encrypted TOTP secret
`created_at`	`TIMESTAMPTZ`	NO	`NOW()`		
`updated_at`	`TIMESTAMPTZ`	NO	`NOW()`		
`deleted_at`	`TIMESTAMPTZ`	YES	`NULL`		
`version`	`INTEGER`	NO	`1`		

```
CREATE TYPE user_role AS ENUM ('owner', 'admin', 'member', 'viewer', 'api');

CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE RESTRICT,
    email VARCHAR(320) NOT NULL,
    password_hash VARCHAR(255),
    full_name VARCHAR(255) NOT NULL,
    role user_role NOT NULL DEFAULT 'member',
    is_verified BOOLEAN NOT NULL DEFAULT FALSE,
    last_login_at TIMESTAMPTZ,
    mfa_enabled BOOLEAN NOT NULL DEFAULT FALSE,
    mfa_secret TEXT,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    deleted_at TIMESTAMPTZ,
    version INTEGER NOT NULL DEFAULT 1
);

CREATE UNIQUE INDEX uq_users_tenant_email ON users(tenant_id, lower(email))
    WHERE deleted_at IS NULL;
CREATE INDEX idx_users_tenant_id ON users(tenant_id)
    WHERE deleted_at IS NULL;
```

4.2 {{DOMAIN_NAME}} Domain

Table: `{{table_name}}`

Purpose: {{TABLE_PURPOSE}}

Column	Type	Nullable	Default	Constraints	Description
`id`	`UUID`	NO	`gen_random_uuid()`	PK	
`tenant_id`	`UUID`	NO		FK → `tenants(id)`	
`created_by`	`UUID`	NO		FK → `users(id)`	
`{{field_1}}`	`{{TYPE}}`	{{YES/NO}}	`{{DEFAULT}}`	{{CONSTRAINTS}}	{{DESCRIPTION}}
`{{field_2}}`	`{{TYPE}}`	{{YES/NO}}	`{{DEFAULT}}`	{{CONSTRAINTS}}	{{DESCRIPTION}}
`status`	`{{status_enum}}`	NO	`'{{DEFAULT_STATUS}}'`	NOT NULL	
`created_at`	`TIMESTAMPTZ`	NO	`NOW()`		
`updated_at`	`TIMESTAMPTZ`	NO	`NOW()`		
`deleted_at`	`TIMESTAMPTZ`	YES	`NULL`		
`version`	`INTEGER`	NO	`1`		

---

5. Enums & Lookup Tables

5.1 Enum Types

```
CREATE TYPE user_role AS ENUM ('owner', 'admin', 'member', 'viewer', 'api');
CREATE TYPE tenant_plan AS ENUM ('free', 'starter', 'pro', 'enterprise');
CREATE TYPE {{entity_1}}_status AS ENUM ('draft', 'active', 'suspended', 'archived');
CREATE TYPE {{entity_2}}_type AS ENUM ('{{VALUE_1}}', '{{VALUE_2}}', '{{VALUE_3}}');
```

5.2 Lookup Tables

Table: `{{lookup_table}}`

Column	Type	Description
`code`	`VARCHAR(50)` PK	Machine-readable identifier
`label`	`VARCHAR(255)`	Human-readable label
`description`	`TEXT`	Detailed description
`sort_order`	`INTEGER`	Display ordering

Column	Type	Description
`is_active`	`BOOLEAN`	Whether selectable

6. Views & Materialized Views

6.1 Views

View: `active_{{entities}}`

Purpose: Filter out soft-deleted records for common queries **Refreshed:** N/A (standard view)

```
CREATE VIEW active_{{entities}} AS
    SELECT * FROM {{table_name}}
    WHERE deleted_at IS NULL;
```

6.2 Materialized Views

Materialized View: `{{entity}}_summary`

Purpose: Pre-aggregated summary for dashboard queries **Refreshed:** Every {{INTERVAL}} via scheduled job **Staleness acceptable:** Up to {{MAX_STALENESS}}

```
CREATE MATERIALIZED VIEW {{entity}}_summary AS
    SELECT
        tenant_id,
        DATE_TRUNC('day', created_at) AS date,
        COUNT(*) AS total,
        COUNT(*) FILTER (WHERE status = 'active') AS active_count
    FROM {{table_name}}
    WHERE deleted_at IS NULL
    GROUP BY tenant_id, DATE_TRUNC('day', created_at);

CREATE UNIQUE INDEX ON {{entity}}_summary(tenant_id, date);

-- Refresh command (run by scheduler):
-- REFRESH MATERIALIZED VIEW CONCURRENTLY {{entity}}_summary;
```

7. Stored Procedures & Functions

`updated_at_trigger()`

Purpose: Auto-update `updated_at` column on any row update

```
CREATE OR REPLACE FUNCTION updated_at_trigger()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = NOW();
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Apply to every table with updated_at:
CREATE TRIGGER set_updated_at
    BEFORE UPDATE ON {{table_name}}
    FOR EACH ROW EXECUTE FUNCTION updated_at_trigger();
```

---

8. Migration Strategy & Tooling

Tool: {{MIGRATION_TOOL}} (e.g., Flyway, Liquibase, Prisma Migrate) **Convention:** `V{timestamp}__{description}.sql` or `{NNN}_{description}.{up|down}.sql` **Location:** `db/migrations/` **Executed by:** CI/CD pipeline before application deployment

Zero-Downtime Migration Checklist

Before every DDL migration:

- ☐ Can this run on a live table without locking? (Use `CONCURRENTLY` for index creation)
- ☐ Does this add a column with a default? (Avoid volatile defaults on large tables)
- ☐ Does this remove a column? (Ensure app code is already deployed without references first)
- ☐ Does this rename? (Use multi-step: add new, backfill, update app, remove old)
- ☐ What's the estimated lock time? (Test on staging data of production size)

Expansion-Contraction Pattern

```
Step 1 (Expand): Add new_column alongside old_column
Step 2 (App deploy): Write to both, read from old
Step 3 (Backfill): Copy data from old to new
Step 4 (App deploy): Read from new, write to both
Step 5 (App deploy): Write to new only
Step 6 (Contract): Drop old_column
```

---

9. Seed Data Requirements

9.1 Required Seed Data (production)

```
-- System tenant (for internal operations)
INSERT INTO tenants (id, name, slug, plan)
VALUES ('00000000-0000-0000-0000-000000000001', 'System', 'system', 'enterprise')
ON CONFLICT DO NOTHING;

-- Default lookup values
INSERT INTO {{lookup_table}} (code, label, sort_order) VALUES
    ('{{VALUE_1}}', '{{LABEL_1}}', 1),
    ('{{VALUE_2}}', '{{LABEL_2}}', 2)
ON CONFLICT (code) DO UPDATE SET label = EXCLUDED.label;
```

9.2 Development Seed Data

Script: `db/seeds/development.sql` **Volume:** {{N}} tenants, {{N}} users per tenant, {{N}} sample records **Command:** `npm run db:seed` or `make seed-dev`

---

10. Performance Considerations

10.1 Partitioning

Table	Partition Strategy	Partition Key	Partition Size

`audit_logs`	Range (time)	`created_at`	Monthly
`{{events_table}}`	Range (time)	`created_at`	Weekly
`{{large_table}}`	List (tenant)	`tenant_id`	Per tenant

```
CREATE TABLE audit_logs (
    id UUID NOT NULL,
    tenant_id UUID NOT NULL,
    created_at TIMESTAMPTZ NOT NULL
) PARTITION BY RANGE (created_at);

CREATE TABLE audit_logs_2024_01
    PARTITION OF audit_logs
    FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');
```

10.2 Query Performance Standards

Query Pattern	Target (p99)	Optimization
PK lookup	< 5ms	B-tree index on id
Tenant-scoped list	< 50ms	Composite index (tenant_id, created_at)
Full-text search	< 200ms	GIN index on `search_vector`
Aggregation (dashboard)	< 500ms	Materialized view
Cross-tenant report	< 30s	Data warehouse

10.3 Connection Pooling

```
Application connections → PgBouncer (transaction mode) → PostgreSQL
Pool size: min={{MIN_POOL}}, max={{MAX_POOL}} per application instance
Max DB connections: {{MAX_DB_CONNECTIONS}} (= pool_size × instances + 10 reserve)
```

---

11. Backup & Recovery Procedures

Backup Type	Method	Frequency	Retention	Location
Continuous WAL	pg_wal_archive	Continuous	{{N}} days	{{BACKUP_LOCATION}}

Backup Type	Method	Frequency	Retention	Location
Base snapshot	pg_basebackup / cloud snapshot	Daily	{{N}} days	{{BACKUP_LOCATION}}
Logical dump	pg_dump (select tables)	Weekly	{{N}} weeks	{{COLD_STORAGE}}
Schema-only	pg_dump --schema-only	On every migration	Indefinite	Git repository

RTO target: {{RTO}} | **RPO target:** {{RPO}}

Recovery test schedule: Monthly ({{DAY_OF_MONTH}}) **Recovery runbook:** {{LINK_TO_RUNBOOK}}

```
# Point-in-time recovery command
pg_restore \
  --host={{HOST}} \
  --port=5432 \
  --username={{USER}} \
  --dbname={{DB}} \
  --target-time="{{TIMESTAMP}}" \
  {{BACKUP_FILE}}
```

Approval

Role	Name	Date	Signature
Author			
DBA / Platform			
Security Review			
Tech Lead			

Revision #8
Created 2026-02-23 12:05:07 UTC by John
Updated 2026-05-25 07:32:13 UTC by John