DB-Dokumentation – Real Estate SaaS (Dokumente & Leads)

Stand: 2025-08-27
Scope: Leads, Properties, Dokumente (Platzhalter, Dateien, Notes), Zugriff (property_roles), Stammdaten (document_types)

---

1) Architekturüberblick

Die Datenbank bildet drei zentrale Bereiche ab:

1. **Leads** – Erfassung, Statusverlauf, Notizen; agentenbasiert sichtbar.
2. **Properties (Objekte)** – Stammdaten des Objekts, Medien (optional), Besitzer (Agent).
3. **Dokumenten-Handling** – je Objekt/Typ ein **Platzhalter** (property_documents), dazu **Dateien** (document_files), **interne Notes** (document_notes), optionale **Events**.
4. **Zugriffsmodell** – `profiles` (Rolle agent|customer) und Objekt-Zuordnung via `property_roles`.

Storage: Supabase Storage Bucket `` `, Pfadkonvention: documents/{property_id}/{document_type_key}/{property_document_id}/{uuid.ext}` ``

---

2) Entity-Relationship (vereinfacht)

```
profiles (id, role)           document_types (id, key)
          │                                 │
          │                                 │
       properties (id, agent_id)            │
          │            ▲                    │
          │            │                    │
 property_roles (property_id, user_id, role)│
          │
          ▼
 property_documents (id, property_id, type_id, status, due_date,
supplier_email, created_by)
          │
          ├── document_files (id, property_document_id, storage_path,
is_shared_with_customer, uploaded_by)
          └── document_notes (id, property_document_id, body, created_by)

Leads-Bereich:
profiles (id) ── leads (agent_id) ── lead_notes (lead_id, author_id) ;
lead_status_history (lead_id, changed_by)
```

---

3) Datenwörterbuch (Tabellen & Felder)

3.1 `profiles`

- **id** (uuid) – entspricht `auth.users.id`.
- **role** (text) – `agent` | `customer`.
- **full_name** (text), **created_at** (timestamptz).

Besonderheiten:
RLS sollte mindestens „eigener Datensatz sichtbar/bearbeitbar“ erzwingen. `profiles` dient als FK-Ziel für Ownership/Funktionen.

3.2 Leads-Domäne

`leads`

- **id** (uuid), **full_name**, **email**, **phone**, **source**.
- **status** (text) – `new|contacted|qualified|converted|archived`.
- **notes** (kurzer Freitext), **agent_id** (uuid → Besitzer/Betreuer).
- **street**, **postal_code**, **city**, **address_text** (generiert), **created_at**, **updated_at**.

Wichtig:

- **FK-Empfehlung:** `agent_id → profiles.id`.
- **Trigger-Empfehlung:** `updated_at` bei Änderungen setzen.
- **Indizes:** auf `agent_id`, `status`, `created_at`.

`lead_notes`

- **id** (uuid), **lead_id**, **author_id**, **body**, **created_at**.
  Indizes: `(lead_id, created_at desc)`.

`lead_status_history`

- **id** (uuid), **lead_id**, **from_status**, **to_status**, **changed_by**, **changed_at**.
  Indizes: `(lead_id, changed_at desc)`.

RLS (Leads-Domäne):

- Agent sieht/ändert **nur** Leads mit `agent_id = auth.uid()`.
- Notes/History nur im Kontext der sichtbaren Leads.
- Customer **kein** Zugriff.

3.3 Properties (Objekte)

`properties`

- **id** (uuid), **title** (text), **address** (text).

- **agent_id** (uuid → `profiles.id`), **created_at**, **updated_at**.
  Indizes: `agent_id`, `created_at desc`.
  Trigger: `updated_at` befüllen.

Optional: `property_media` **(falls verwendet)**

- **property_id**, **path**, **mime**, **is_cover**, **created_at**.
  Sichtbarkeit: analog zu `properties` (Owner ist Agent).

`property_roles`

- **property_id** (uuid), **user_id** (uuid), **role** (`agent|customer`).
- **assigned_by** (uuid), **created_at**.
  PK: `(property_id, user_id)`; **Indizes:** `user_id`, `(property_id, role)`.

Zweck: Customer-Zugriff auf einzelne Objekte abbilden – Grundlage für RLS der Dokumente.

RLS (Empfehlung):

- Agent (Owner des Objekts) darf Einträge zu *seinen* Objekten verwalten/sehen.
- Customer darf seine eigenen Einträge lesen (optional, meist nicht notwendig im Frontend).

3.4 Dokument-Stammdaten

`document_types`

- **id** (uuid), **key** (unique; z. B. `mietvertrag`, `grundbuch`, `nk`, `energie`), **label** (Anzeigename), **is_active**, **created_at**.
  Seeds: Vier Standard-Typen (MVP) sind eingetragen.
  Zweck: Klare, normalisierte Typreferenz für Platzhalter.

3.5 Dokumente (pro Objekt)

`property_documents` **(Platzhalter)**

- **id** (uuid), **property_id** (uuid), **type_id** (uuid).
- **status** (text) – `pending|uploaded|overdue`.
- **due_date** (date), **supplier_email** (text).
- **created_by** (uuid), **last_seen_at_agent** (timestamptz, optional).
- **created_at**, **updated_at**.
  Unique-Constraint: `(property_id, type_id)` – je Objekt/Typ genau ein Platzhalter.
  Indizes: `property_id`, `type_id`, `status`, `due_date`.
  Trigger: `updated_at` befüllen.

RLS:

- Agent: `select/insert/update/delete` **nur**, wenn ihm das Objekt gehört (`properties.agent_id = auth.uid()`).

- Customer: `select` **nur**, wenn eine Rolle in `property_roles` für das Objekt besteht (role=`customer`).
- Customer kann **keine** neuen Platzhalter anlegen (kein insert/update/delete für Customer).

`document_files` **(Dateien)**

- **id** (uuid), **property_document_id** (uuid), **storage_path** (text), **filename**, **ext**, **mime_type**, **size**.
- **is_shared_with_customer** (bool, default `true`).
- **uploaded_by** (uuid), **created_at**.
  Indizes: `(property_document_id, created_at desc)`, `uploaded_by`, `is_shared_with_customer`.

RLS:

- Agent: CRUD im Rahmen eigener Objekte.
- Customer:
- **select** nur, wenn er Zugriff aufs Objekt hat **und** (Datei freigegeben **oder** selbst hochgeladen).
- **insert** nur, wenn Zugriff aufs Objekt besteht; dabei muss `uploaded_by = auth.uid()` und `is_shared_with_customer = true` sein.
- **update/delete** nicht erlaubt.

Nebenwirkung (Status): Datei-Änderungen aktualisieren `property_documents.status` (siehe Trigger-Logik unten).

`document_notes` **(interne Notizen – nur Agent)**

- **id** (uuid), **property_document_id** (uuid), **body**, **created_by**, **created_at**, **edited_at**.
  Indizes: `(property_document_id, created_at desc)`.
  RLS: Nur Agent mit Objekt-Ownership darf lesen/schreiben.
  Trigger: `edited_at` bei Update setzen.

Optional: `document_events`

- **id**, **property_document_id**, **type** (text), **meta** (jsonb), **created_at**, **created_by**.
  Zweck: Audit & Reminder-Protokollierung (MVP optional; empfehlenswert für spätere Features).

---

4) RLS-Matrix (Kurzüberblick)

Tabelle	Agent (Owner des Objekts)	Customer (über `property_roles`)
profiles	Eigener Datensatz (min.)	Eigener Datensatz (min.)
leads	**SELECT/INSERT/UPDATE/ DELETE** eigene Leads	–
lead_notes	Sicht/Schreiben im Kontext eigener Leads	–
lead_status_history	Sicht im Kontext eigener Leads	–

Tabelle	Agent (Owner des Objekts)	Customer (über `property_roles`)
properties	**SELECT/INSERT/UPDATE/DELETE** eigene	Optional: SELECT (nur zugeordnete)
property_roles	**SELECT/INSERT/UPDATE/DELETE** für eigene	Optional: SELECT eigene Zuordnungen
document_types	SELECT	SELECT
property_documents	**SELECT/INSERT/UPDATE/DELETE** für eigene	**SELECT** (wenn `property_roles` existiert), kein CRUD
document_files	**SELECT/INSERT/UPDATE/DELETE** für eigene	**SELECT/INSERT** (geteilt/selbst), kein Update/Delete
document_notes	**SELECT/INSERT/UPDATE/DELETE** für eigene	–

> „Eigene“ bedeutet: Objekt gehört dem Agenten (`properties.agent_id = auth.uid()`) bzw. Lead besitzt `agent_id = auth.uid()`.

---

5) Trigger & Geschäftslogik

5.1 `updated_at` / `edited_at`

- `` ` – setzt `` updated_at `vor Update (u. a. bei` properties `,` property_documents`).
- `` ` – setzt `` edited_at `vor Update von` document_notes`.

5.2 Status-Automatik für `property_documents`

- **Ziel:** Status konsistent halten, ohne App-Logik zu duplizieren.
- **Quellen:** Anzahl zugehöriger Dateien und Fälligkeitsdatum.
- **Regeln:**
- **uploaded**: Wenn ≥ **1** Datei vorhanden.
- **pending**: Wenn **keine** Datei und `due_date` **nicht** überschritten.
- **overdue**: Wenn **keine** Datei und `due_date < today`.

Trigger-Ablauf:

1. **Nach INSERT** in `document_files` → Status auf `uploaded`.
2. **Nach DELETE** in `document_files` → Falls **keine** Datei übrig: `pending` / `overdue` abhängig von `due_date`.
3. **Nach UPDATE** von `due_date` in `property_documents` → Status neu berechnen.

Hinweis: `last_seen_at_agent` kann vom Frontend bei Ansicht des Dokuments aktualisiert werden, um „Neu“-Badges zu steuern.

---

6) Indizes & Performance-Hinweise

- `properties`: auf `agent_id`, `created_at desc`.
- `property_documents`: auf `property_id`, `type_id`, `status`, `due_date`, Unique `(property_id, type_id)`.
- `document_files`: auf `(property_document_id, created_at desc)`, `uploaded_by`, `is_shared_with_customer`.
- `document_notes`: auf `(property_document_id, created_at desc)`.
- Leads-Domäne: `leads(agent_id)`, `leads(status)`, `lead_notes(lead_id, created_at)`, `lead_status_history(lead_id, changed_at)`.

Praxis:

- Für Listenseiten Paginierung nutzen (limit/offset oder keyset), Filter (Status/Typ), selektive Spalten.

---

7) Seeds & Stammdaten

- `document_types`: vier Einträge (MVP): Mietvertrag, Grundbuchauszug, Nebenkostenabrechnung, Energieausweis.
- Aktivierungsflag `is_active` erlaubt spätere Erweiterung/Deaktivierung.

---

8) Sicherheit & Datenfluss

- **Owner-Modell:** `properties.agent_id` definiert die Hoheit des Agents über Objekt & Dokumente.
- **Kundenzugriff:** ausschließlich über `property_roles` (role=`customer`).
- **Dateifreigabe:** `is_shared_with_customer` steuert Sichtbarkeit pro Datei.
- **Customer-Uploads:** erlaubt **nur** in bestehende Platzhalter der zugeordneten Objekte; Uploads werden dem `uploaded_by` zugeordnet.

Nicht erlaubt (MVP):

- Customer darf **keine** neuen Platzhalter anlegen.
- Customer kann Dateien **nicht** aktualisieren oder löschen.

---

9) Typische Fehlerbilder & Diagnoseleitfaden

9.1 „Customer sieht keine Dokumente“

Checkliste:

1. Existiert ein Eintrag in `property_roles` für das Objekt/den Customer?
2. Gibt es Platzhalter (`property_documents`) für das Objekt?
3. Sind Dateien als `is_shared_with_customer = true` markiert? (Nur dann sichtbar – außer eigene Uploads.)

9.2 „Upload schlägt für Customer fehl“

Checkliste:

1. Besteht `property_roles`-Zugriff für das Objekt?
2. Upload erfolgt **gegen bestehenden Platzhalter** (gültige `property_document_id`)?
3. Wird `uploaded_by = auth.uid()` gesetzt?
4. Verhindern Policies evtl. das Insert (z. B. fehlende Rolle)?

9.3 „Status aktualisiert sich nicht“

Checkliste:

1. Existieren Trigger für Datei-Insert/Delete und Due-Date-Update?
2. Liegt wirklich **mindestens** eine Datei am Platzhalter?
3. Liegt `due_date` in der Vergangenheit, aber keine Datei vorhanden → erwartet: `overdue`.

9.4 „Agent sieht fremde Objekte/Dokumente“

Checkliste:

1. Stimmt `properties.agent_id`?
2. Greifen RLS-Policies auf `property_documents` / `document_files` (Bedingung `p.agent_id = auth.uid()`)?

9.5 „Customer kann Platzhalter hinzufügen“

Checkliste:

1. Existiert **keine** Insert/Update/Delete-Policy für Customer auf `property_documents`?
2. UI blendet Add-Modal in Customer-Sicht aus.

9.6 „Datei ist für Customer unsichtbar“

Checkliste:

1. `is_shared_with_customer = true`?
2. Falls `false`: Nur sichtbar, wenn `uploaded_by = Customer` selbst.
3. Customer hat `property_roles`-Zugriff auf das Objekt.

---

10) Betriebsleitfaden (Migrationen & Reihenfolge)

Empfohlene Reihenfolge für frische Setups:

1. `profiles` (falls noch nicht vorhanden) & RLS baseline.
2. Leads-Domäne (falls frisch): `leads`, `lead_notes`, `lead_status_history` (+ Trigger/Indizes/RLS).
3. `properties` (+ Trigger/Indizes/RLS).
4. `document_types` (+ Seeds).
5. `property_documents` (+ Trigger/Indizes/RLS).

6. `document_files` (+ RLS).
7. `document_notes` (+ Trigger/Indizes/RLS).
8. `property_roles` (+ RLS).
9. Optionale `document_events`.

Bestands-DB:

- Altlasten (z. B. `todos`) entfernen.
- FK-Ergänzungen (z. B. `leads.agent_id → profiles.id`).
- Index-Nachrüstungen & Trigger aktivieren.

---

11) Testfälle (manuell)

1. **Agent** legt Objekt an → sieht es in Liste/Detail.
2. Agent legt Platzhalter (z. B. „Mietvertrag“) an → Status `pending`.
3. Agent setzt `due_date` auf gestern → Status `overdue` (keine Datei vorhanden).
4. Customer erhält Objektzugriff via `property_roles` → sieht Platzhalter.
5. Customer lädt Datei in Platzhalter hoch → Datei sichtbar, Status `uploaded`.
6. Agent setzt `is_shared_with_customer = false` bei einer Datei → Customer sieht diese Datei nicht (es sei denn, es ist sein eigener Upload).
7. Agent erstellt/editiert/löscht **document_notes** → nur Agent sichtbar.
8. Lösche **alle** Dateien eines Platzhalters → Status wechselt zu `pending/overdue` (abhängig von `due_date`).

---

12) Erweiterungen (Backlog-fähig)

- **Automatische Erinnerungen**: täglich prüfen (`overdue`), E-Mail senden, `document_events` protokollieren.
- **Q&A pro Dokument**: `document_messages` mit resolved-Flag, Notifications.
- **Office-Preview**: Serverseitige Konvertierung nach PDF für Inline-Viewer.
- **Versionierung**: Versionen je Platzhalter, Restore/Compare.
- **Unkategorisierte Einreichung** (Customer-Inbox): `unassigned_submissions` + Zuordnung zu Platzhalter/Neuanlage `property_documents`.

---

13) Glossar

- **Platzhalter**: Einträge in `property_documents`, definieren „welche Dokumente“ pro Objekt erwartet/geführt werden.
- **Freigabe**: Sichtbarkeit einzelner Dateien für Customer via `is_shared_with_customer`.
- **Owner**: Der Agent, der in `properties.agent_id` als Besitzer geführt wird.
- **Zugriff Customer**: Über `property_roles` je Objekt.
- **Status**: Automatisch ermittelter Zustand eines Platzhalters basierend auf Dateibestand und Fälligkeitsdatum.