Love it. Here’s a **thorough first draft** of the **Control Plane database schema** that fits your rules:

* Public API is **data-plane CRUD only** (no billing/DDL exposed).
* Control Plane manages **tenants, users, plans, billing, usage/metering**, and **DB-perimeter API tokens** (➜ **one token = one base/database**).
* Designed to start simple and scale to industrial.

I’m using neutral types (Postgres-flavored: uuid, timestamptz, jsonb, int, text, numeric) and calling out **PK/UK/FK**, **indexes**, and **notes** where it helps.

**Control Plane — Logical Model**

**Org hierarchy**

* **tenant** → **workspace** → **base** *(= database in the data plane)*

**Identity & access**

* **user**, **membership** (user ↔ workspace, role)
* **sso\_config**, **invitation**

**Plans & billing**

* **plan**, **plan\_feature**, **subscription**, **subscription\_item**, **invoice**, **payment\_method**
* **wallet**, **wallet\_ledger** (credits for automation runs)

**Access to data plane**

* **api\_token** *(scoped to a single* ***base****)*, optional **api\_token\_ip\_allowlist**

**Schema governance (control only)**

* **base\_schema\_version**, **schema\_migration\_job**

**Metering & quotas**

* **usage\_base\_daily**, **usage\_tenant\_monthly**, **api\_token\_usage\_daily**
* **quota\_override**, **rate\_limit** (per token and/or tenant)

**Ops & config**

* **region**, **feature\_flag\_assignment**

(Platform/admin & alarms are in other DBs, by design.)

**Tables & Fields**

**1) Tenancy & Structure**

**tenant**

* **id (uuid, pk)**
* slug (text, unique) — for URLs/emails
* name (text)
* status (text enum: active|suspended|deleted)
* default\_region\_id (uuid, fk→region.id)
* created\_at (timestamptz), updated\_at (timestamptz)
* **IDX:** (slug), (status)

**workspace**

* **id (uuid, pk)**
* tenant\_id (uuid, fk→tenant.id)
* name (text)
* status (text enum: active|archived)
* created\_at, updated\_at
* **UK:** (tenant\_id, name) — prevent dup names within tenant
* **IDX:** (tenant\_id, status)

**base *(= one logical database in the data plane)***

* **id (uuid, pk)**
* tenant\_id (uuid, fk→tenant.id)
* workspace\_id (uuid, fk→workspace.id)
* name (text)
* status (text enum: active|locked|deleted)
* region\_id (uuid, fk→region.id)
* data\_plane\_ref (text) — router key or DSN alias (no raw creds)
* created\_at, updated\_at
* **UK:** (workspace\_id, name)
* **IDX:** (tenant\_id, region\_id, status)

**Note:** In DB-per-tenant mode, data\_plane\_ref can be a logical reference to a connection broker; in schema-per-tenant mode, it’s a shard key.

**2) Users, Memberships, SSO**

**user**

* **id (uuid, pk)**
* email (citext, unique)
* name (text)
* status (text enum: active|invited|disabled)
* mfa\_enabled (boolean, default false)
* last\_login\_at (timestamptz)
* created\_at, updated\_at
* **IDX:** (status), (last\_login\_at)

**membership *(user’s role in a workspace)***

* **id (uuid, pk)**
* user\_id (uuid, fk→user.id)
* workspace\_id (uuid, fk→workspace.id)
* role (text enum: owner|admin|editor|viewer)
* created\_at, updated\_at
* **UK:** (user\_id, workspace\_id)
* **IDX:** (workspace\_id, role), (user\_id)

**invitation**

* **id (uuid, pk)**
* workspace\_id (uuid, fk→workspace.id)
* email (citext)
* role (text enum: as above)
* invited\_by\_user\_id (uuid, fk→user.id)
* token\_hash (text, unique)
* expires\_at (timestamptz)
* accepted\_at (timestamptz null)
* created\_at
* **IDX:** (workspace\_id, email)

**sso\_config *(Business+)***

(Per tenant; SAML/OIDC metadata)

* **id (uuid, pk)**
* tenant\_id (uuid, fk→tenant.id, unique)
* provider (text enum: saml|oidc)
* metadata\_json (jsonb)
* sso\_enforced (bool, default false)
* created\_at, updated\_at

**3) Plans, Billing & Credits**

**plan**

* **id (uuid, pk)**
* code (text, unique) — free|pro|business|enterprise
* name (text)
* price\_cents (int) — per editor / mo (annualized logic in app)
* monthly\_included\_runs (int)
* record\_cap (int) — per base
* storage\_cap\_gb (int)
* features\_json (jsonb) — toggles
* created\_at, updated\_at

**plan\_feature**

* **id (uuid, pk)**
* plan\_id (uuid, fk→plan.id)
* key (text) — e.g., sso, row\_level\_security, private\_templates
* value (jsonb)
* **UK:** (plan\_id, key)

**subscription**

* **id (uuid, pk)**
* tenant\_id (uuid, fk→tenant.id)
* plan\_id (uuid, fk→plan.id)
* status (text enum: trialing|active|past\_due|canceled)
* seats (int)
* period\_start (timestamptz), period\_end (timestamptz)
* external\_ref (text) — Stripe subscription id
* created\_at, updated\_at
* **IDX:** (tenant\_id, status), (period\_end)

**subscription\_item *(seat counts / add-ons)***

* **id (uuid, pk)**
* subscription\_id (uuid, fk→subscription.id)
* item\_type (text enum: seat|premium\_connectors|extra\_storage)
* quantity (int)
* price\_cents (int)
* **IDX:** (subscription\_id, item\_type)

**invoice**

* **id (uuid, pk)**
* tenant\_id (uuid, fk→tenant.id)
* external\_ref (text, unique) — Stripe invoice id
* amount\_due\_cents (int)
* amount\_paid\_cents (int)
* status (text enum: draft|open|paid|void|uncollectible)
* issued\_at (timestamptz), due\_at (timestamptz), paid\_at (timestamptz null)
* created\_at
* **IDX:** (tenant\_id, issued\_at DESC), (status)

**payment\_method**

* **id (uuid, pk)**
* tenant\_id (uuid, fk→tenant.id)
* provider (text enum: stripe)
* external\_ref (text) — PM id in Stripe (no PAN here)
* brand (text), last4 (text)
* is\_default (bool)
* created\_at, updated\_at
* **UK:** (tenant\_id, external\_ref)

**wallet *(Automation credits wallet)***

* **id (uuid, pk)**
* tenant\_id (uuid, fk→tenant.id, unique)
* balance\_cents (int) — $ balance for credits
* balance\_runs (int) — pre-purchased runs remaining
* auto\_refill\_enabled (bool, default true)
* refill\_threshold\_cents (int, default 100) — $1
* expires\_at (timestamptz) — **12-month rolling**
* created\_at, updated\_at

**wallet\_ledger**

* **id (uuid, pk)**
* wallet\_id (uuid, fk→wallet.id)
* entry\_type (text enum: topup|debit|expire|adjust)
* amount\_cents (int) — positive for topups
* runs\_delta (int) — positive for credit purchase, negative for usage
* ref (text) — external payment ref / usage batch id
* created\_at
* **IDX:** (wallet\_id, created\_at DESC)

**Pricing alignment:** $5 = 2,500 runs. App logic updates both amount\_cents and runs\_delta atomically.

**4) API Access (Per-Database Tokens)**

**api\_token**

* **id (uuid, pk)**
* tenant\_id (uuid, fk→tenant.id)
* **base\_id (uuid, fk→base.id)** — **required** ➜ **token is scoped to exactly one base**
* label (text) — user-friendly name
* token\_hash (text, unique) — store only a hash
* scopes (text[] enum: records:read, records:write, records:delete)
* status (text enum: active|revoked)
* last\_used\_at (timestamptz)
* last\_used\_ip (inet)
* created\_by\_user\_id (uuid, fk→user.id)
* created\_at, revoked\_at
* **IDX:** (tenant\_id, base\_id, status), (last\_used\_at DESC)

**api\_token\_ip\_allowlist *(optional)***

* **id (uuid, pk)**
* api\_token\_id (uuid, fk→api\_token.id, on delete cascade)
* cidr (cidr)
* **UK:** (api\_token\_id, cidr)

**Notes**

* A tenant can issue **several tokens**, each **locked to one base**.
* Scopes are **CRUD-only** for records. No billing or DDL scopes exist.

**5) Schema Governance (Control-side, not public)**

**base\_schema\_version**

* **id (uuid, pk)**
* base\_id (uuid, fk→base.id)
* version (int) — monotonic
* ddl\_hash (text) — checksum of applied DDL/spec
* spec\_json (jsonb) — canonical schema spec (tables/fields)
* created\_by\_user\_id (uuid, fk→user.id)
* created\_at
* **UK:** (base\_id, version)
* **IDX:** (base\_id, created\_at DESC)

**schema\_migration\_job**

* **id (uuid, pk)**
* base\_id (uuid, fk→base.id)
* from\_version (int), to\_version (int)
* status (text enum: queued|running|succeeded|failed|rolled\_back)
* requested\_by\_user\_id (uuid, fk→user.id)
* started\_at, finished\_at
* error (text)
* **IDX:** (base\_id, status)

Public API never hits these. Only the Portal/Admin services use them.

**6) Metering, Quotas & Rate Limits**

**usage\_base\_daily**

* **id (uuid, pk)**
* date (date)
* tenant\_id (uuid, fk→tenant.id)
* base\_id (uuid, fk→base.id)
* records\_count (int) — snapshot EOD
* storage\_gb (numeric(10,3)) — attachments/storage usage
* runs\_used (int) — automation runs consumed that day
* api\_calls (int) — API requests via api\_token
* updated\_at (timestamptz)
* **UK:** (date, base\_id)
* **IDX:** (tenant\_id, date), (base\_id, date)

**usage\_tenant\_monthly**

* **id (uuid, pk)**
* month (date) — first day of month
* tenant\_id (uuid, fk→tenant.id)
* runs\_used (int)
* api\_calls (int)
* records\_count\_max (int) — max across bases in month
* storage\_gb\_max (numeric(10,3))
* updated\_at
* **UK:** (month, tenant\_id)
* **IDX:** (tenant\_id, month)

**api\_token\_usage\_daily**

* **id (uuid, pk)**
* date (date)
* api\_token\_id (uuid, fk→api\_token.id)
* calls (int)
* bytes\_in (bigint), bytes\_out (bigint)
* p95\_latency\_ms (int)
* **UK:** (date, api\_token\_id)

**quota\_override**

* **id (uuid, pk)**
* tenant\_id (uuid, fk→tenant.id)
* scope (text enum: runs|storage|records|api\_calls)
* target (text enum: tenant|base)
* base\_id (uuid, fk→base.id, nullable unless target=base)
* value (int or numeric) — store as jsonb { "int": 50000 } if mixed types
* valid\_from (timestamptz), valid\_to (timestamptz null)
* reason (text)
* created\_by\_admin\_id (uuid) *(from Admin DB identity if separate)*
* **IDX:** (tenant\_id, base\_id, scope)

**rate\_limit**

* **id (uuid, pk)**
* api\_token\_id (uuid, fk→api\_token.id) *(or tenant\_id for tenant-level caps)*
* window (text enum: 1s|10s|1m|1h|1d)
* max\_requests (int)
* created\_at, updated\_at
* **UK:** (api\_token\_id, window)

**7) Regions & Feature Flags**

**region**

* **id (uuid, pk)**
* code (text, unique) — eu-west-1, us-east-1, etc.
* name (text)
* created\_at

**feature\_flag\_assignment**

* **id (uuid, pk)**
* tenant\_id (uuid, fk→tenant.id)
* key (text) — new\_automation\_ui, beta\_connectors
* value (jsonb) — { "enabled": true }
* valid\_to (timestamptz null)
* created\_at
* **UK:** (tenant\_id, key)

**Relationships (quick view)**

* tenant 1—N workspace
* workspace 1—N base
* user N—N workspace via membership
* base 1—N api\_token (**perimeter = base**)
* tenant 1—1 wallet, 1—N wallet\_ledger
* tenant 1—N subscription 1—N subscription\_item
* tenant 1—N invoice, 1—N payment\_method
* base 1—N base\_schema\_version, 1—N schema\_migration\_job
* Usage tables roll up **per base/day** and **per tenant/month**.

**Indexing & Partitioning (starter guidance)**

* **Hot paths**
  + api\_token(token\_hash) — **unique** & fast lookup at auth time.
  + api\_token(tenant\_id, base\_id, status) — listing tokens for a base.
  + usage\_\* tables — **partition by date** (monthly partitions) if you cross millions of rows.
* **Foreign key fan-outs**
  + membership(workspace\_id), membership(user\_id)
  + base(workspace\_id), base(tenant\_id)  
    Add B-trees; keep FKs “ON” (consistency > micro-perf at control plane scale).
* **Cardinality**
  + quota\_override(tenant\_id, base\_id) typically tiny — still index for admin queries.

**Security & Data Hygiene**

* **Token storage:** store only token\_hash (e.g., SHA-256 + salt). Show plaintext once on create.
* **PII:** user.email is PII. If you later add phone/address, separate user\_profile with encryption at rest (pgcrypto/KMS).
* **Multi-region:** base.region\_id governs data-plane residency; enforce via provisioning service.
* **Soft deletes:** Use status enums and keep historical rows for auditability; do hard deletes only on strict request.

**How tokens map to your rule (“one token = one DB”)**

* api\_token.base\_id is **NOT NULL** and **FK → base.id**.
* No tenant-scoped tokens exist; every token is **base-scoped**.
* Token’s scopes allow only **record CRUD** (records:read|write|delete).
* DDL/billing endpoints **do not exist** on this token plane.

**What you can build next with this**

* **OpenAPI for Control Plane** (internal): endpoints for issuing/revoking tokens, managing wallets/subscriptions, listing usage.
* **Provisioner** that reads base.region\_id + data\_plane\_ref to create/attach the actual DB.
* **Metering workers** that write usage\_base\_daily and usage\_tenant\_monthly, and debit wallet\_ledger in atomic batches.

If you want, I can turn this into:

* a **SQL DDL starter pack** (Postgres) with types, enums, FKs, and indices