Contents

1 Entity Relationship Diagrams (ERDs)

1.1 1. Document Control

- **Version:** 2.1
- **Author:** Henry Huerta
- **Date:** 2025-11-3
- **Reviewers:** Prof. Arnold Lau, T.A. Sneh Bhandari

This document expands the high-level ERD from the TDD into digestible domain diagrams with attributes and notes.

> **MVP Scope Note:** While this document describes the complete database schema supporting all planned features, the MVP focuses exclusively on core account and access card management (USER, MEMBER, ACCESS_CARD, MEMBERSHIP_PLAN, GYM, and related audit/status tables). See `MVP_SCOPE.md` for details on what features are implemented in the current phase versus future phases.

1.2 2. Notation Legend

This ERD uses **Crow's Foot notation** (also known as IE notation):

1.2.1 Cardinality Symbols

- `||--||` : One-to-one (each entity has exactly one related entity)
- `||--o|` : One-to-many (parent can have zero or many children)
- `}o--o{` : Many-to-many (requires junction table for implementation)

1.2.2 Symbols Key

- `||` : Required/mandatory participation (total participation)
- `o` : Optional participation (partial participation)
- **Direction** : Parent entity is on the left, child entity is on the right

1.2.3 Attribute Legend

- **PK** : Primary Key
- **FK** : Foreign Key

- **U** : Unique constraint
- **R** : Required (NOT NULL)
- Empty : Optional (NULL allowed)

1.2.4 Normalization Level

- **Current Normalization: Third Normal Form (3NF)**
- All tables are in 3NF with:
  - No partial key dependencies
  - No transitive dependencies
  - Proper foreign key relationships
- Some strategic denormalization in audit tables (`after_json` field) and reporting views for performance

Diagrams included - Overview (no audit tables, no *_STATUS_IND) - Gym & Equipment (+ audits) - Staff (+ audits) - Classes & Trainer Availability (+ audits) - Members, Plans, Bookings & Check-ins (+ audits) - Admins (+ audits) - Status Indicator Table Structure - Audit Table Structure

---

1.3 3. Overview (no audit or status indicator tables)

*Same as TDD overview, but **without** audit tables and **without** status indicator tables and **without attributes** for readability.*

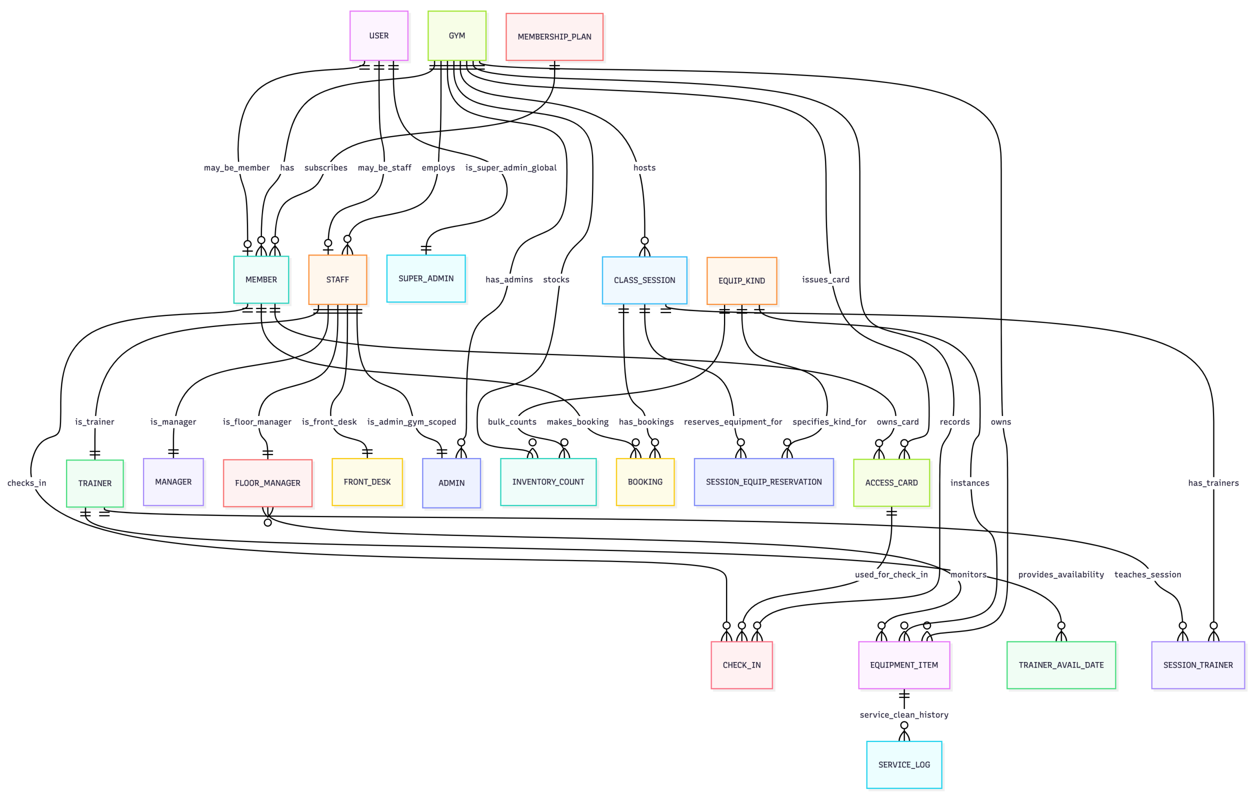


Figure 1: ERD Diagram 1: Overview

–>

1.4 4. Gym & Equipment (+ audits)

Equipment, items, inventory counts, service logs.

Attribute types - **Key:** `GYM.id`, `EQUIP_KIND.id`, `EQUIPMENT_ITEM.id`, `INVENTORY_COUNT.id`, `SERVICE_LOG.id` - **Composite:** none - **Multi-valued:** none - **Derived:** `next_clean_due_at` (from last_cleaned/intervals), `reorder_needed` (from quantities & threshold)

Relevant triggers - `equipment_item_clean_due_update` - recalculate `next_clean_due_at` / `cleaning_required` when cleaned - `service_log_item_flags` - on `SERVICE_LOG` insert: update `EQUIPMENT_ITEM.last_serviced_at`; set/reset `service_required` and `cleaning_required` based on `action` - `inventory_reorder_flag` - on `INVENTORY_COUNT` change: recalculate `reorder_needed` - `equip_item_status_guard` - for status to be `ok` whre must be no service/clean flags

Relevant checks - **Column-level**: - `EQUIP_KIND.mode IN ('per_item','bulk')` - `EQUIPMENT_ITEM.uses_count >= 0`, `rated_uses > 0`, `cleaning_interval_uses >= 0`, `cleaning_interval_days >= 0` - `INVENTORY_COUNT.qty_on_floo` `>= 0`, `qty_in_storage >= 0` - **Table-level**: - Unique `(gym_id, equip_kind_id)` in `INVENTORY_COUNT` - Partial unique on `(gym_id, serial_no)` where `serial_no IS NOT NULL` (or full unique if always present) - FKs `ON DELETE RESTRICT` for audits; `SERVICE_LOG.equipment_item_id ON DELETE RESTRICT`

Indexes - `GYM(name)`, `GYM(status_id)`, `GYM(created_at)` - `EQUIP_KIND(name) UNIQUE`, `EQUIP_KIND(mode)` - `EQUIPMENT_ITEM(gym_id, equip_kind_id)`, `EQUIPMENT_ITEM(status_id)`, `EQUIPMENT_ITEM(next_clean_due_at)` - `INVENTORY_COUNT(gym_id, equip_kind_id) UNIQUE`, `INVENTORY_COUNT(updated_snapshot_at)` - `SERVICE_LOG(equipment_item_id, serviced_at DESC)`, `SERVICE_LOG(staff_id)`

1.5 5. Staff (+ audits)

Users, staff specializations, and floor managers monitoring equipment.

Attribute types - **Key:** `USER.id`, `STAFF.id`, `TRAINER.id`, `MANAGER.id`, `FLOOR_MANAGER.id`, `FRONT_DESK.id`, `ADMIN.id`, `SUPER_ADMIN.id` - **Composite:** none - **Multi-valued:** none - **Derived:** none

Relevant triggers - `staff_gym_scope_guard` - ensure role's `STAFF` belongs to the target `GYM` - `user_status_login_guard` - prevent setting `USER.status='inactive'` if stil an active staff, etc. - `user_password_hash_enforce` - validate hash format

Relevant checks - **Column-level**: - `USER.status IN ('active','inactive','locked')` - `STAFF.status IN ('active','inactive')` - `ADMIN.scope = 'gym'`, `SUPER_ADMIN.scope = 'global'` - `FRONT_DESK.capabilities` subset check (enforced via permitted values or separate lookup table) - **Table-level**: - Unique `(user_id)` in `STAFF`, unique `(staff_id)` in each specialization table - FK `STAFF.user_id -> USER.id ON DELETE RESTRICT`; specialization tables `ON DELETE CASCADE` from `STAFF`

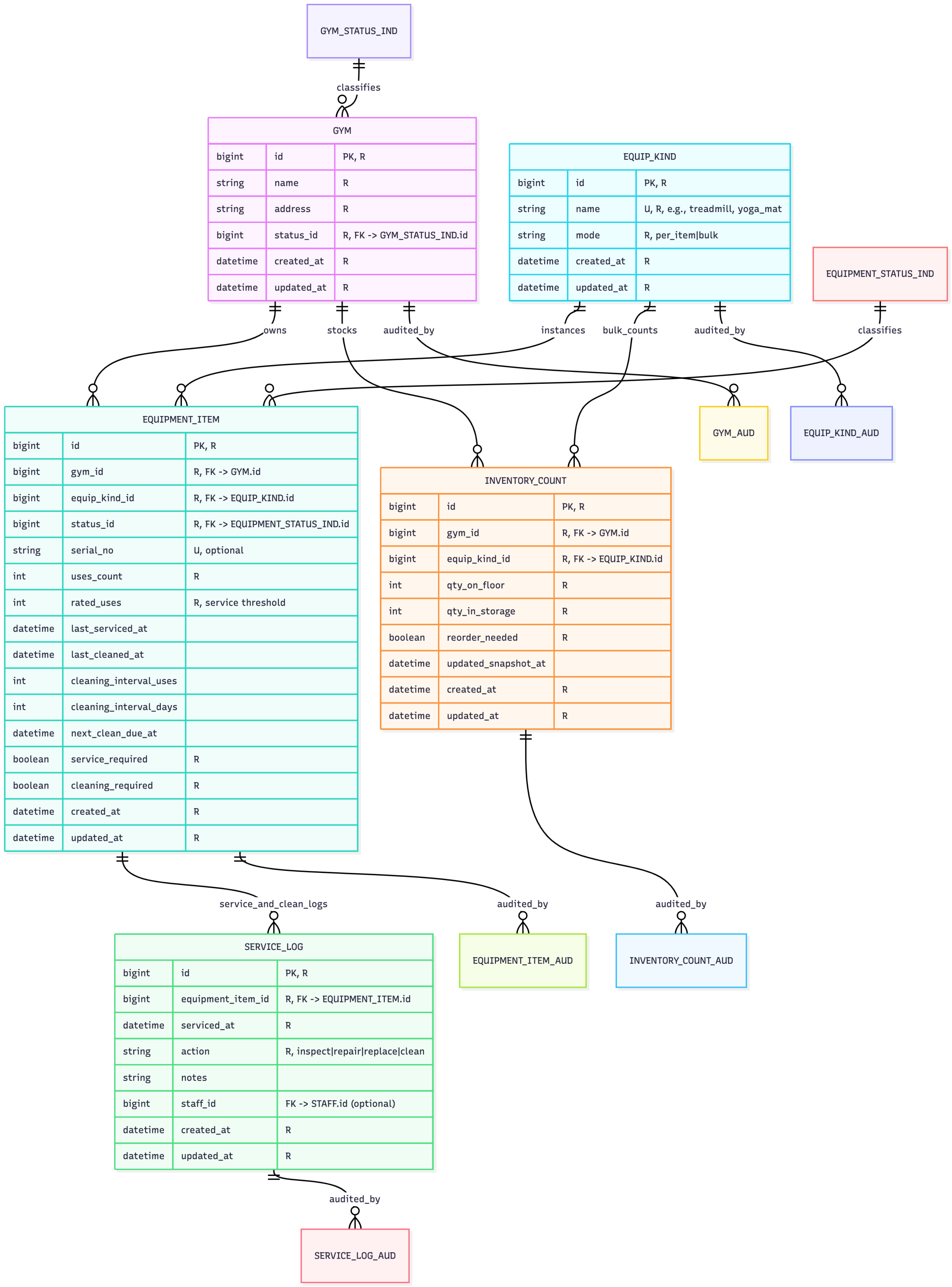
Indexes - `USER(username) UNIQUE`, `USER(email) UNIQUE`, `USER(status_id)`, `USER(last_login_at)` - `STAFF(user_id) UNIQUE`, `STAFF(gym_id, status_id)`, `STAFF(created_at)` - Each role table: `(<role>.staff_id) UNIQUE` - `EQUIPMENT_ITEM(status_id)`, `EQUIPMENT_ITEM(gym_id, equip_kind_id)`

1.6 6. Classes & Trainer Availability (+ audits)

Sessions, trainer availability, staffing, and per-session equipment reservations.

Attribute types - **Key:** `CLASS_SESSION.id`, `TRAINER_AVAIL_DATE.id` - **Composite:** `SESSION_TRAINER(session_id, trainer_id)`, `SESSION_EQUIP_RESERVATION(session_id, equip_kind_id)` - **Multi-valued:** none - **Derived:** none

Relevant triggers - `session_capacity_guard` - prevent inserts to `SESSION_TRAINER` beyond `CLASS_SESSION.max_trainers` - `session_booking_open_guard` - prevent reservations/bookings when `open_for_booking = false` - `availability_match_guard` - ensure trainer availability exists



GYM

bigint	id	PK, R
string	name	R
string	address	R
bigint	status_id	R, FK -> GYM_STATUS_IND.id
datetime	created_at	R
datetime	updated_at	R

EQUIP_KIND

bigint	id	PK, R
string	name	U, R, e.g., treadmill, yoga_mat
string	mode	R, per_item|bulk
datetime	created_at	R
datetime	updated_at	R

EQUIPMENT_ITEM

bigint	id	PK, R
bigint	gym_id	R, FK -> GYM.id
bigint	equip_kind_id	R, FK -> EQUIP_KIND.id
bigint	status_id	R, FK -> EQUIPMENT_STATUS_IND.id
string	serial_no	U, optional
int	uses_count	R
int	rated_uses	R, service threshold
datetime	last_serviced_at	
datetime	last_cleaned_at	
int	cleaning_interval_uses	
int	cleaning_interval_days	
datetime	next_clean_due_at	
boolean	service_required	R
boolean	cleaning_required	R
datetime	created_at	R
datetime	updated_at	R

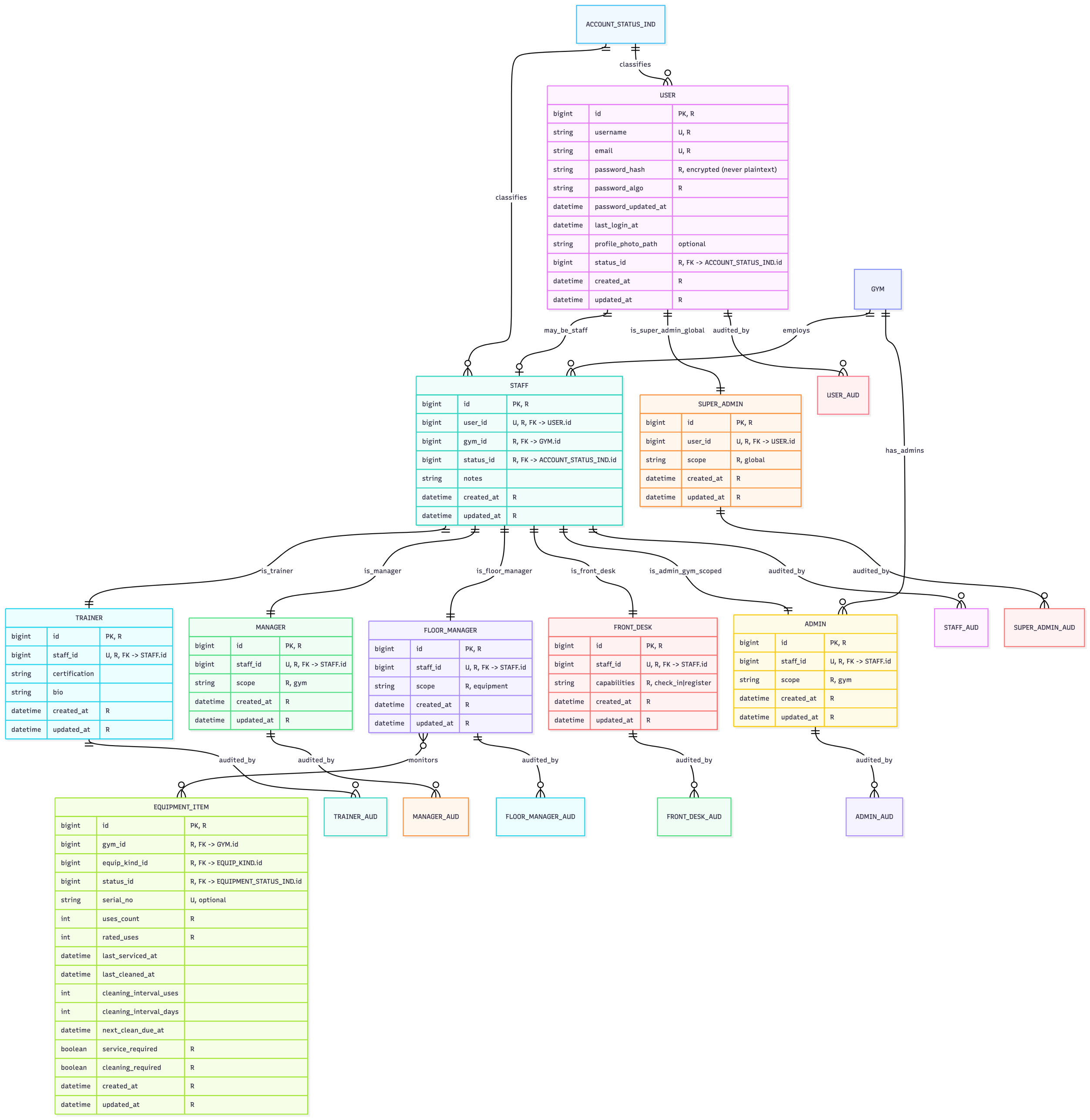
INVENTORY_COUNT

bigint	id	PK, R
bigint	gym_id	R, FK -> GYM.id
bigint	equip_kind_id	R, FK -> EQUIP_KIND.id
int	qty_on_floor	R
int	qty_in_storage	R
boolean	reorder_needed	R
datetime	updated_snapshot_at	
datetime	created_at	R
datetime	updated_at	R

SERVICE_LOG

bigint	id	PK, R
bigint	equipment_item_id	R, FK -> EQUIPMENT_ITEM.id
datetime	serviced_at	R
string	action	R, inspect|repair|replace|clean
string	notes	
bigint	staff_id	FK -> STAFF.id (optional)
datetime	created_at	R
datetime	updated_at	R

Figure 2: ERD Diagram 2: 4. Gym & Equipment



USER

bigint	id	PK, R
string	username	U, R
string	email	U, R
string	password_hash	R, encrypted (never plaintext)
string	password_algo	R
datetime	password_updated_at	
datetime	last_login_at	
string	profile_photo_path	optional
bigint	status_id	R, FK -> ACCOUNT_STATUS_IND.id
datetime	created_at	R
datetime	updated_at	R

STAFF

bigint	id	PK, R
bigint	user_id	U, R, FK -> USER.id
bigint	gym_id	R, FK -> GYM.id
bigint	status_id	R, FK -> ACCOUNT_STATUS_IND.id
string	notes	
datetime	created_at	R
datetime	updated_at	R

SUPER_ADMIN

bigint	id	PK, R
bigint	user_id	U, R, FK -> USER.id
string	scope	R, global
datetime	created_at	R
datetime	updated_at	R

TRAINER

bigint	id	PK, R
bigint	staff_id	U, R, FK -> STAFF.id
string	certification	
string	bio	
datetime	created_at	R
datetime	updated_at	R

MANAGER

bigint	id	PK, R
bigint	staff_id	U, R, FK -> STAFF.id
string	scope	R, gym
datetime	created_at	R
datetime	updated_at	R

FLOOR_MANAGER

bigint	id	PK, R
bigint	staff_id	U, R, FK -> STAFF.id
string	scope	R, equipment
datetime	created_at	R
datetime	updated_at	R

FRONT_DESK

bigint	id	PK, R
bigint	staff_id	U, R, FK -> STAFF.id
string	capabilities	R, check_in|register
datetime	created_at	R
datetime	updated_at	R

ADMIN

bigint	id	PK, R
bigint	staff_id	U, R, FK -> STAFF.id
string	scope	R, gym
datetime	created_at	R
datetime	updated_at	R

EQUIPMENT_ITEM

bigint	id	PK, R
bigint	gym_id	R, FK -> GYM.id
bigint	equip_kind_id	R, FK -> EQUIP_KIND.id
bigint	status_id	R, FK -> EQUIPMENT_STATUS_IND.id
string	serial_no	U, optional
int	uses_count	R
int	rated_uses	R
datetime	last_serviced_at	
datetime	last_cleaned_at	
int	cleaning_interval_uses	
int	cleaning_interval_days	
datetime	next_clean_due_at	
boolean	service_required	R
boolean	cleaning_required	R
datetime	created_at	R
datetime	updated_at	R

Figure 3: ERD Diagram 3: 5. Staff

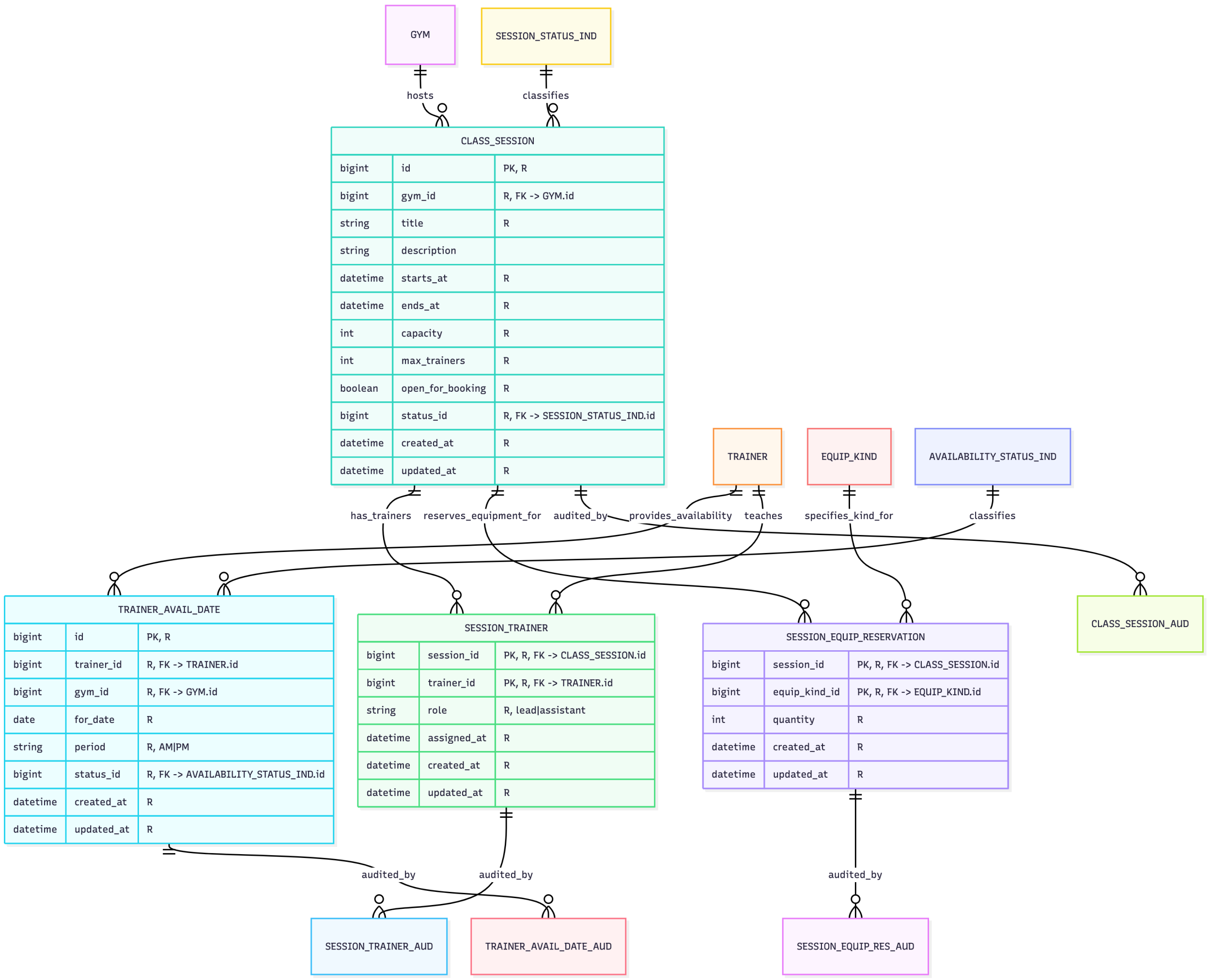


Figure 4: ERD Diagram 4: 6. Classes & Trainer Availability

for session date/period & gym before `SESSION_TRAINER` insert - `session_time_coherence` - ensure `ends_at > starts_at` and enforce allowed session windows (e.g., exclude 11:00–13:00 per policy) - `session_equip_res_guard` - ensure `quantity >= 0` for the equipment to reserve and is available inventory in gym - `session_cancel_cascade` - ensure that sessions that are canceled notify members and free up staff

Relevant checks - **Column-level**: - `CLASS_SESSION.capacity > 0`, `max_trainers >= 1` - `CLASS_SESSION.status IN ('scheduled','canceled','completed')` - `TRAINER_AVAIL_DATE.period IN ('AM','PM')`, `status IN ('available','unavailable')` - `SESSION_EQUIP_RESERVATION.quantity >= 0` - **Table-level**: - unique `(session_id, trainer_id)` in `SESSION_TRAINER` - unique `(session_id, equip_kind_id)` in `SESSION_EQUIP_RESERVATION` - `SESSION_TRAINER.session_id -> CLASS_SESSION.id ON DELETE CASCADE` (remove staffing if session is deleted)

Indexes - `CLASS_SESSION(gym_id, starts_at)`, `CLASS_SESSION(status_id, open_for_booking)`, `CLASS_SESSION(starts_at)` - `TRAINER_AVAIL_DATE(trainer_id, for_date, period) UNIQUE`, `TRAINER_AVAIL_DATE(gym` `for_date, period)`, `TRAINER_AVAIL_DATE(status_id)` - `SESSION_TRAINER(session_id, trainer_id) UNIQUE`, `SESSION_TRAINER(trainer_id, session_id)` - `SESSION_EQUIP_RESERVATION(session_id, equip_kind_id) UNIQUE` —

1.7 7. Members, Plans, Bookings & Check-ins (+ audits)

Memberships, bookings, access cards and check-ins (plus can check-in at any gym; trial/basic tied to home gym).

Attribute types - **Key:** `USER.id`, `MEMBERSHIP_PLAN.id`, `MEMBER.id`, `BOOKING.id`, `CHECK_IN.id`, `ACCESS_CARD.id`, `GYM.id` - **Composite:** none - **Multi-valued:** none - **Derived:** none

Relevant triggers - `booking_plus_only` - forbid `BOOKING` when member's plan tier `plus` - `booking_capacity_guard` - ensure confirmed bookings `CLASS_SESSION.capacity` - `booking_unique_member_session` - prevent multiple active bookings for the same `(member_id, session_id)` - `booking_window_guard` - restrict bookings to current and next month per policy - `checkin_scope_guard` - allow `trial|basic` only at `home_gym_id`; `plus` at any gym - `access_card_status_guard` - forbid check-ins with `ACCESS_CARD.status IN ('lost','revoked')` - `user_password_hash_enforce` - validate hash format

Relevant checks - **Column-level**: - `MEMBERSHIP_PLAN.tier IN ('trial','basic','plus')`, `billing_cycle IN ('monthly','annual')`, `price >= 0` - `MEMBER.status IN ('active','suspended','canceled')` - `BOOKING.status IN ('confirmed','canceled_member','canceled_system')` - `ACCESS_CARD.status IN ('active','lost','revoked')` - **Table-level**: - unique `(username)` and `(email)` in `USER` - unique `(user_id)` in `MEMBER` (1:1 user member) - unique `(member_id, session_id)` in `BOOKING` (active rows) - unique `(card_uid)` in `ACCESS_CARD`

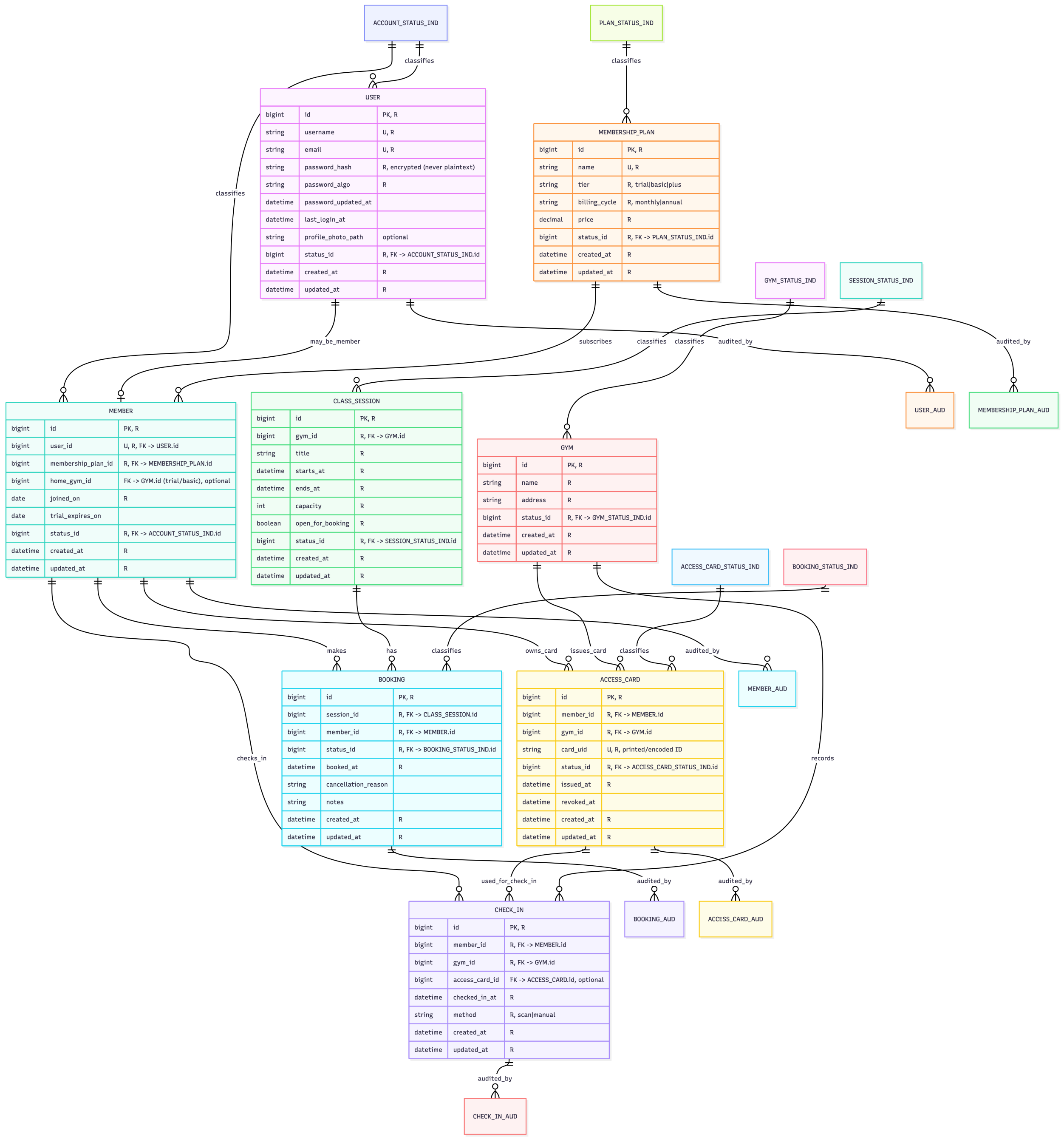
Indexes - `USER(username) UNIQUE`, `USER(email) UNIQUE`, `USER(status_id)`, `USER(last_login_at)` - `MEMBERSHIP_PLAN(name) UNIQUE`, `MEMBERSHIP_PLAN(tier, status_id)`, `MEMBERSHIP_PLAN(price)` - `MEMBER(user_id) UNIQUE`, `MEMBER(status_id)`, `MEMBER(membership_plan_id)`, `MEMBER(home_gym_id)` - `BOOKING(session_id, member_id) UNIQUE`, `BOOKING(member_id, booked_at)`, `BOOKING(status_id)` - `CHECK_IN(member_id, checked_in_at)`, `CHECK_IN(gym_id, checked_in_at)` - `ACCESS_CARD(card_uid) UNIQUE`, `ACCESS_CARD(member_id, status_id)`, `ACCESS_CARD(gym_id, status_id)`

1.8 8. Admins (+ audits)

Gym-scoped admins and global super-admins.

Attribute types - **Key:** `USER.id`, `STAFF.id`, `ADMIN.id`, `SUPER_ADMIN.id`, `GYM.id` - **Composite:** none - **Multi-valued:** none - **Derived:** none

Relevant triggers - `admin_scope_guard` - enforce privilege scope is bound to one gym - `user_admin_status_guard` - prevent orphaning of active users that have other roles - maybe add one so some admin actions need



USER

bigint	id	PK, R
string	username	U, R
string	email	U, R
string	password_hash	R, encrypted (never plaintext)
string	password_algo	R
datetime	password_updated_at	
datetime	last_login_at	
string	profile_photo_path	optional
bigint	status_id	R, FK -> ACCOUNT_STATUS_IND.id
datetime	created_at	R
datetime	updated_at	R

MEMBERSHIP_PLAN

bigint	id	PK, R
string	name	U, R
string	tier	R, trial|basic|plus
string	billing_cycle	R, monthly|annual
decimal	price	R
bigint	status_id	R, FK -> PLAN_STATUS_IND.id
datetime	created_at	R
datetime	updated_at	R

MEMBER

bigint	id	PK, R
bigint	user_id	U, R, FK -> USER.id
bigint	membership_plan_id	R, FK -> MEMBERSHIP_PLAN.id
bigint	home_gym_id	FK -> GYM.id (trial/basic), optional
date	joined_on	R
date	trial_expires_on	
bigint	status_id	R, FK -> ACCOUNT_STATUS_IND.id
datetime	created_at	R
datetime	updated_at	R

CLASS_SESSION

bigint	id	PK, R
bigint	gym_id	R, FK -> GYM.id
string	title	R
datetime	starts_at	R
datetime	ends_at	R
int	capacity	R
boolean	open_for_booking	R
bigint	status_id	R, FK -> SESSION_STATUS_IND.id
datetime	created_at	R
datetime	updated_at	R

GYM

bigint	id	PK, R
string	name	R
string	address	R
bigint	status_id	R, FK -> GYM_STATUS_IND.id
datetime	created_at	R
datetime	updated_at	R

BOOKING

bigint	id	PK, R
bigint	session_id	R, FK -> CLASS_SESSION.id
bigint	member_id	R, FK -> MEMBER.id
bigint	status_id	R, FK -> BOOKING_STATUS_IND.id
datetime	booked_at	R
string	cancellation_reason	
string	notes	
datetime	created_at	R
datetime	updated_at	R

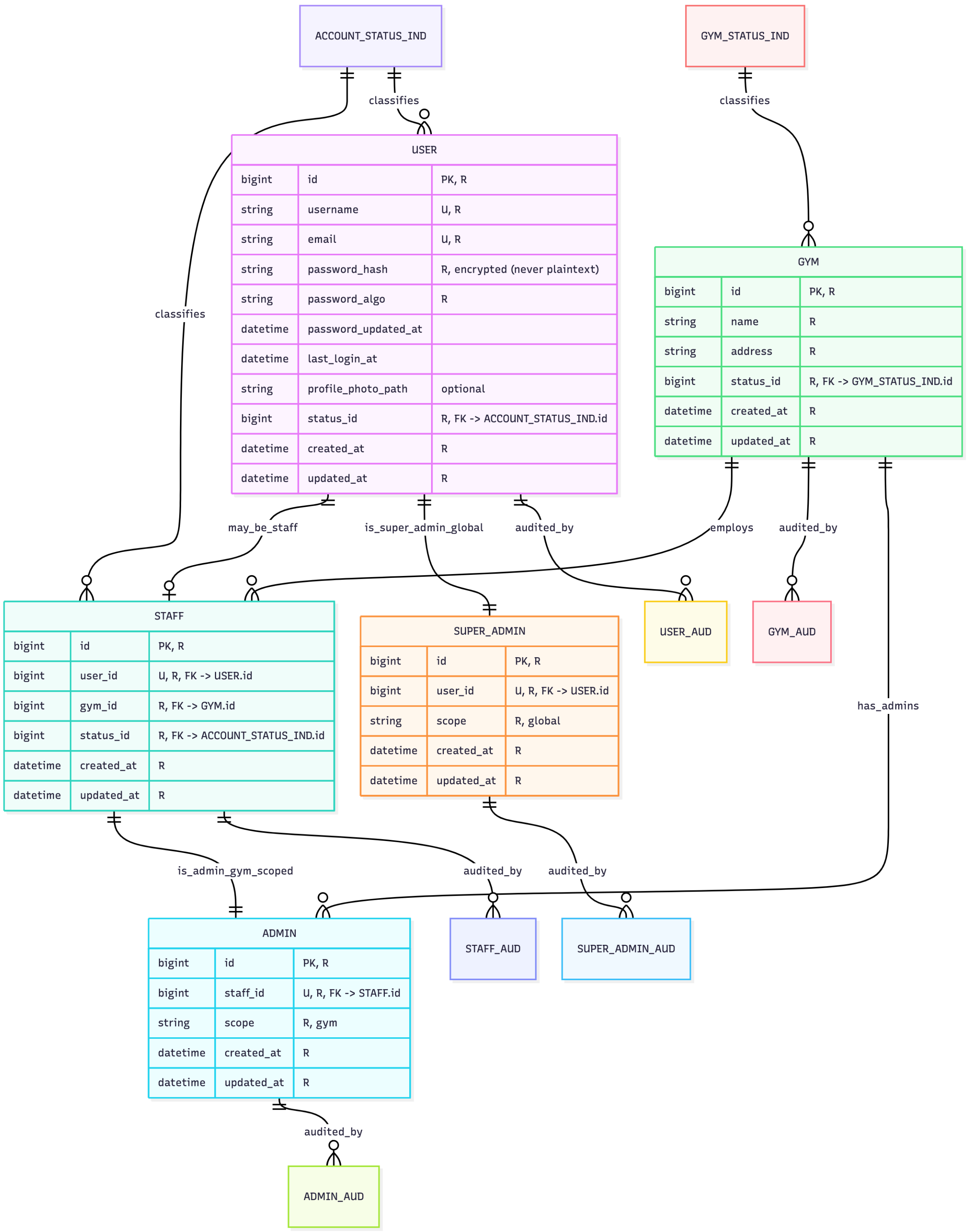
ACCESS_CARD

bigint	id	PK, R
bigint	member_id	R, FK -> MEMBER.id
bigint	gym_id	R, FK -> GYM.id
string	card_uid	U, R, printed/encoded ID
bigint	status_id	R, FK -> ACCESS_CARD_STATUS_IND.id
datetime	issued_at	R
datetime	revoked_at	
datetime	created_at	R
datetime	updated_at	R

CHECK_IN

bigint	id	PK, R
bigint	member_id	R, FK -> MEMBER.id
bigint	gym_id	R, FK -> GYM.id
bigint	access_card_id	FK -> ACCESS_CARD.id, optional
datetime	checked_in_at	R
string	method	R, scan|manual
datetime	created_at	R
datetime	updated_at	R

Figure 5: ERD Diagram 5: 7. Members, Plans, Bookings & Check-ins



USER

bigint	id	PK, R
string	username	U, R
string	email	U, R
string	password_hash	R, encrypted (never plaintext)
string	password_algo	R
datetime	password_updated_at	
datetime	last_login_at	
string	profile_photo_path	optional
bigint	status_id	R, FK -> ACCOUNT_STATUS_IND.id
datetime	created_at	R
datetime	updated_at	R

GYM

bigint	id	PK, R
string	name	R
string	address	R
bigint	status_id	R, FK -> GYM_STATUS_IND.id
datetime	created_at	R
datetime	updated_at	R

STAFF

bigint	id	PK, R
bigint	user_id	U, R, FK -> USER.id
bigint	gym_id	R, FK -> GYM.id
bigint	status_id	R, FK -> ACCOUNT_STATUS_IND.id
datetime	created_at	R
datetime	updated_at	R

SUPER_ADMIN

bigint	id	PK, R
bigint	user_id	U, R, FK -> USER.id
string	scope	R, global
datetime	created_at	R
datetime	updated_at	R

ADMIN

bigint	id	PK, R
bigint	staff_id	U, R, FK -> STAFF.id
string	scope	R, gym
datetime	created_at	R
datetime	updated_at	R

Figure 6: ERD Diagram 6: 8. Admins

approval from more admins

Relevant checks - **Column-level**: - `USER.status IN ('active','inactive','locked')` - `ADMIN.scope = 'gym'`, `SUPER_ADMIN.scope = 'global'` - **Table-level**: - Unique `(user_id)` in `STAFF`; unique `(staff_id)` in `ADMIN`

Indexes - `USER(username) UNIQUE`, `USER(email) UNIQUE`, `USER(status_id)` - `STAFF(user_id) UNIQUE`, `STAFF(gym_id, status_id)` - `ADMIN(staff_id) UNIQUE` - `GYM(name)`, `GYM(status_id)` —

1.9 9. Status Indicator Table Structure

Generic structure for all `*_STATUS_IND` tables (serves as lookup tables).

Indicator families (examples) - `ACCOUNT_STATUS_IND` → `USER.status_id`, `STAFF.status_id`, `MEMBER.status_id` Codes: `ACTIVE, INACTIVE, LOCKED, SUSPENDED, CANCELED` - `GYM_STATUS_IND` → `GYM.status_id` (e.g., `ACTIVE, INACTIVE`) - `EQUIPMENT_STATUS_IND` → `EQUIPMENT_ITEM.status_id` (`OK, NEEDS_SERVICE, OUT_OF_ORDER, RETIRED`) - `SESSION_STATUS_IND` → `CLASS_SESSION.status_id` (`SCHEDULED, CANCELED, COMPLETED`) - `AVAILABILITY_STATUS_IND` → `TRAINER_AVAIL_DATE.status_id` (`AVAILABLE, UNAVAILABLE`) - `PLAN_STATUS_IND` → `MEMBERSHIP_PLAN.status_id` (`ACTIVE, RETIRED`) - `ACCESS_CARD_STATUS_IND` → `ACCESS_CARD.status_id` (`ACTIVE, LOST, REVOKED`) - `BOOKING_STATUS_IND` → `BOOKING.status_id` (`CONFIRMED, CANCELED_MEMBER, CANCELED_SYSTEM`)

Querying pattern examples - Active members: `... JOIN ACCOUNT_STATUS_IND si ON m.status_id = si.id AND si.code = 'ACTIVE'` - Sessions open & scheduled: `... JOIN SESSION_STATUS_IND ssi ON cs.status_id = ssi.id AND ssi.code = 'SCHEDULED' AND cs.open_for_booking = TRUE` - Equipment out or needs service: `... JOIN EQUIPMENT_STATUS_IND esi ON ei.status_id = esi.id AND esi.code IN ('NEEDS_SERVICE','OUT_OF_ORDER')`

1.10 10. Audit Table Structure

Generic structure for all `*_AUD` tables.

How states are captured - Triggers write `after_json` for each event. Prior state can be reconstructed from the **previous audit row's `after_json`** (append-only history) - `seq_no` provides a reliable ordering of the audits

Constraints & integrity - `BASE_ENTITY_AUD(base_entity_id) → BASE_ENTITY(id)` - `UNIQUE(seq_no)` (per-table sequence), plus index `(base_entity_id, seq_no)` for efficient timelines.

Indexes - `(base_entity_id, seq_no)` **covering index** (primary read path) - `(actor_user_id, seq_no)` for actor timelines - `(occurred_at)` for time-range queries

Rollback considerations - Rollbacks are constructed using the last **valid `after_json`** snapshot prior to the target time/sequence number (transactions are applied in reverse order to avoid inconsistencies)

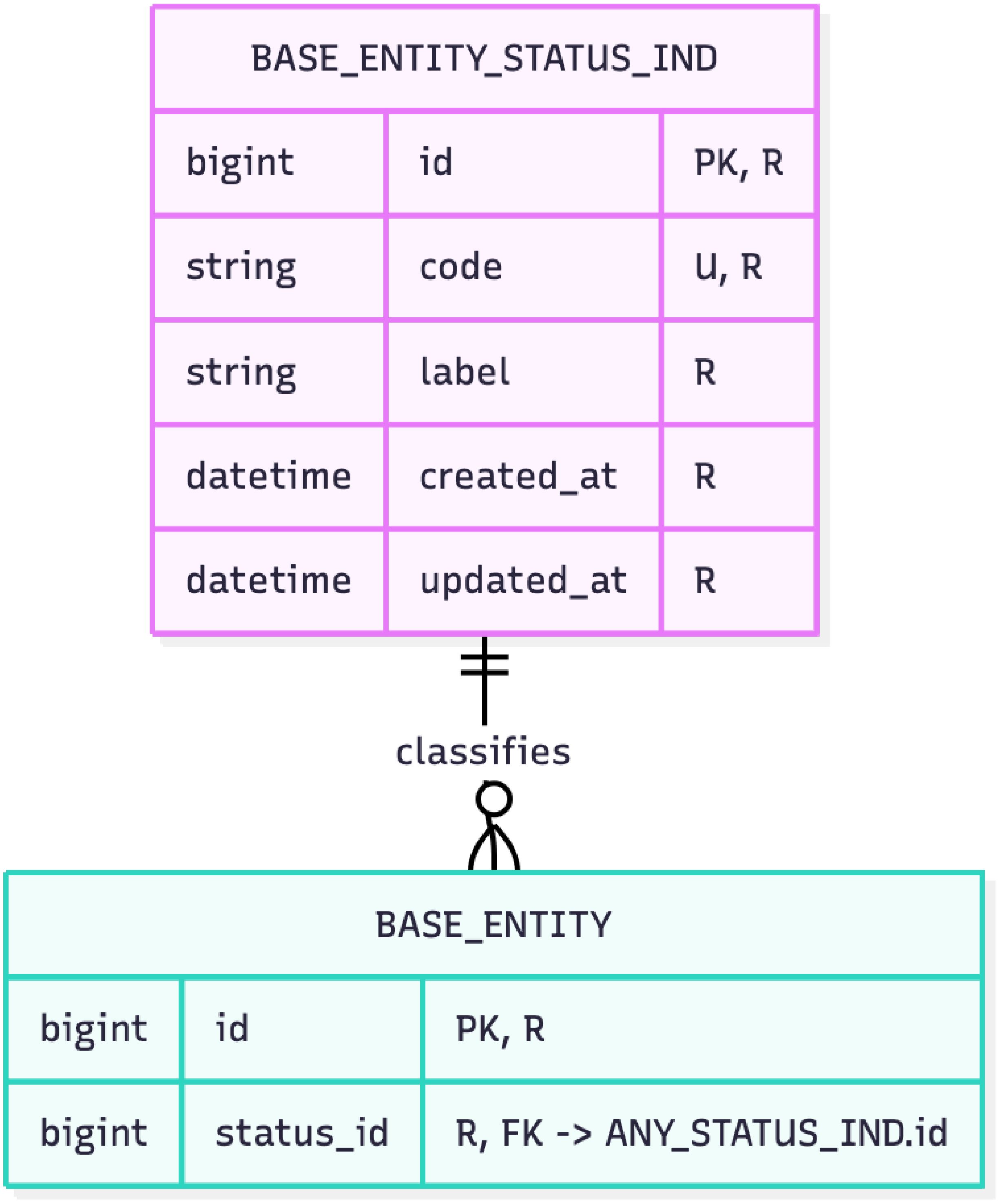


Figure 7: ERD Diagram 7: 9. Status Indicator Table Structure

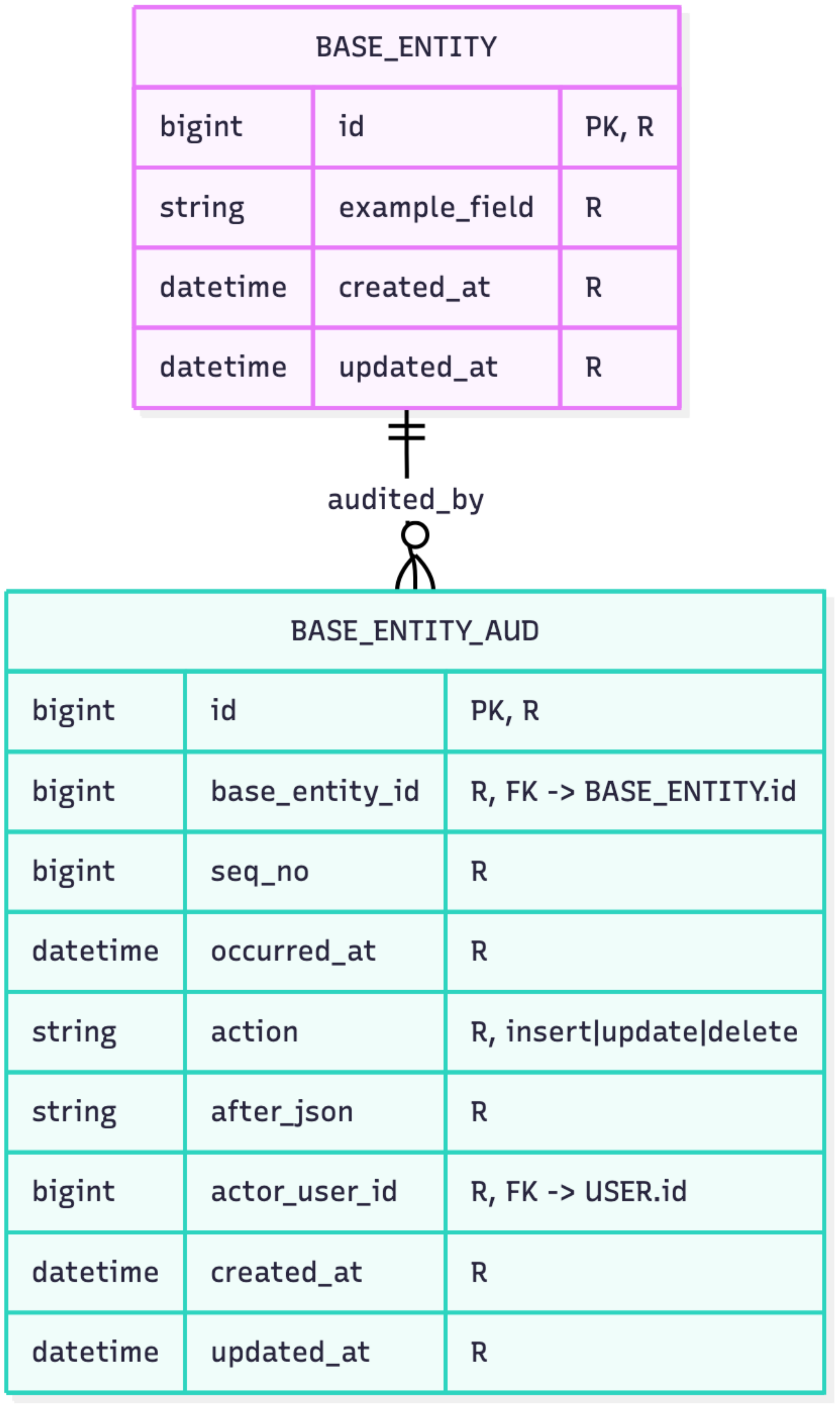

BASE_ENTITY
bigint | id | PK, R
string | example_field | R
datetime | created_at | R
datetime | updated_at | R
audited_by
BASE_ENTITY_AUD
bigint | id | PK, R
bigint | base_entity_id | R, FK -> BASE_ENTITY.id
bigint | seq_no | R
datetime | occurred_at | R
string | action | R, insert|update|delete
string | after_json | R
bigint | actor_user_id | R, FK -> USER.id
datetime | created_at | R
datetime | updated_at | R

Figure 8: ERD Diagram 8: 10. Audit Table Structure