ChatGPT

Business-Level Feature Descriptions

This document summarizes the **business-visible capabilities** introduced or altered by the consolidated CRM change set. Each section is keyed to a **domain object** touched by the schema and references the exact SQL lines that drive the change. The descriptions remain high-level and avoid implementation details.

Enumerations

The change set introduces several Postgres enums to strongly type CRM concepts. These enums make values explicit, simplify validation and enable generic automation across entities.

CRM record type (`crm_record_type`)

- **Purpose:** Standardizes how records are identified across polymorphic tables. It enumerates the domain objects that can appear in generic tables such as watchers, automation actions and stage history. Values include contact, company, deal, lead, ticket, activity, pipeline and list.

Automation scope type (`automation_scope_type`)

- **Purpose:** Defines where an automation rule applies. It supports five scopes—record, pipeline, pipeline stage, list and entity. Each scope determines which target columns must be set in the automation_action table (e.g., a record rule requires `record_type` and `record_id`, while a pipeline rule requires `pipeline_id`).

Automation action type (`automation_action_type`)

- **Purpose:** Categorizes what an automation does. Allowed values are `WEBHOOK`, `WORKFLOW`, `EVENT` and `AIWORKER`. These types correspond to different execution engines (e.g., calling an external webhook or running an AI job).

Action execution status (`action_execution_status`)

- **Purpose:** Tracks the lifecycle of an automation execution. Statuses include `PENDING`, `IN_PROGRESS`, `SUCCEEDED` and `FAILED`.

Pipeline object type (`pipeline_object_type`)

- **Purpose:** Identifies which domain entity a pipeline is used for. Possible values are `COMPANY`, `CONTACT`, `DEAL`, `LEAD` and `TICKET`. This enables separate pipelines for different objects (e.g., deals and tickets) and allows the API to fetch pipelines by object type.

Pipeline movement mode (`pipeline_movement_mode`)

- **Purpose:** Specifies how stage transitions are enforced. A value of `FLEXIBLE` allows free movement between stages while `ENFORCED` requires that stages be traversed sequentially.

Pipeline stage state (`pipeline_stage_state`)

- **Purpose:** Replaces simple "won/lost" sales semantics with a general stage lifecycle. Stages can be `NOT_STARTED`, `IN_PROGRESS`, `DONE_SUCCESS` or `DONE_FAILED`. This standardizes stage semantics across pipelines.

Pipeline stage type (`pipeline_stage_type`)

- **Purpose:** Retained for backward compatibility. Values are `OPEN`, `SUCCESS` and `FAILED`. New integrations should use `pipeline_stage_state` instead.

List processing type (`list_processing_type`)

- **Purpose:** Indicates whether a list is **static** or **dynamic**. Static lists contain explicitly added members, while dynamic lists recalculate membership based on filter criteria. Allowed values are `STATIC` and `DYNAMIC`.

Deal type (`deal_type`)

- **Purpose:** Categorizes deals into `NEW`, `RENEWAL`, `UPSELL` or `OTHER`. This field supports pipeline reporting and segmentation.

Principal type (`principal_type`)

- **Purpose:** Used by the record watcher mechanism to identify the subscriber. Allowed values are `USER` and `GROUP`.

List object type (`list_object_type`)

- **Purpose:** Normalizes the `object_type` column on `list` and the `member_type` column on `list_membership`. Values include `CONTACT`, `COMPANY`, `DEAL`, `LEAD`, `TICKET` and `ACTIVITY`. Converting these columns from free-form text to an enum ensures consistent typing and enables efficient filtering.

New Domain Objects

Record watcher

- **Capability added:** Introduces a unified mechanism for users or groups to **follow any record**. Each watcher links a tenant, record type, record identifier, principal type (user or group) and principal identifier. It records when the watch was created and by whom. The primary key spans all of these columns, ensuring that a principal can subscribe to a record only once.

- **Behavior:** Applications can now offer “subscribe to changes” functionality on contacts, companies, deals, leads, tickets, activities, pipelines or lists. Indexes allow efficient retrieval of watchers by principal or by record. Polymorphic integrity (valid record and principal) is enforced in the application layer.

Automation action

- **Capability added:** Defines **declarative automation rules** in the form “when X happens, do Y.” Each action specifies:
- **entity_type** — the domain record type the rule applies to.
- **scope_type** — one of ENTITY, PIPELINE, PIPELINE_STAGE, LIST or RECORD. The scope determines which target identifier must be populated (e.g., pipeline_id for pipeline scope, record_id for record scope). A check constraint ensures that exactly one target column is set.
- **trigger_event** and optional **condition_json** — specify the event and any conditions that must be true.
- **action_type** and **config_json** — describe what to execute and its configuration. Supported action types include webhook calls, workflow invocations, event publications and AI worker tasks.
- **priority** and **enabled** flags — control execution order and whether the rule is active.
- **inherit_pipeline_actions** — for stage-scoped rules, indicates whether pipeline-level rules should also run.

Automation actions enable global automations (scope = ENTITY), pipeline-level rules, stage-specific rules, list membership triggers and record-specific rules. Indexes support efficient lookup by record, pipeline, stage or list, and foreign keys ensure that pipeline, stage and list identifiers refer to existing objects.

Automation action execution

- **Capability added:** Provides a **detailed execution log** for each run of an automation action. For every attempt the system records:
- **action_id** (foreign key to automation_action) and **tenant_id**.
- **entity_type** and **entity_id** of the affected record, along with optional **pipeline_id**, **from_stage_id**, **to_stage_id** and **list_id** for context.
- **trigger_event** and a unique **execution_key** used for idempotency; a unique constraint on `(tenant_id, execution_key)` prevents duplicate executions.
- **status** (`PENDING`, `IN_PROGRESS`, `SUCCEEDED`, `FAILED`), **response_code**, **response_body**, **error_message**, **triggered_at**, **started_at**, **completed_at** and audit timestamp **created_at**.

- **Behavior:** The table supports auditing, debugging, idempotency and operational reporting. It is append-heavy (rows are inserted on every run and updated as the status changes). Deleting an automation action cascades its execution history. Indexes enable queries by status and by entity.

Stage history

- **Capability added:** Tracks **stage transitions** for any stage-based entity (e.g., deal or ticket). Each record captures the entity type and ID, the pipeline involved, the previous and new stage identifiers, when the change occurred (`changed_at`), who initiated it (`changed_by_user_id`) and an optional **source** describing why the change happened.

- **Behavior:** Foreign keys ensure that pipeline and stage identifiers reference existing pipelines and stages and are nullable when not applicable. Indexes support queries by entity and by pipeline. This enables reporting on stage movements and dwell times.

Modified Domain Objects

Pipeline

- **New fields:** `object_type` (enum), `display_order`, `is_active`, `pipeline_key`, and `movement_mode`. These support multiple pipelines per object, stable identifiers, ordering, activation, and enforced movement rules. The change set backfills defaults (object_type = `DEAL`, pipeline_key built from the ID and first characters, display_order computed per object type) and then enforces non-null constraints.
- **Constraints:** Adds unique constraints on `(tenant_id, object_type, pipeline_key)` and on `(tenant_id, object_type, display_order)` and an index on `(tenant_id, object_type)`.

Pipeline stage

- **Renamed column:** `stage_order` is renamed to `display_order` if it exists.
- **New fields:** Adds `stage_state` (enum) with default `NOT_STARTED` and `inherit_pipeline_actions` (boolean). These fields support JIRA-style stage semantics and allow stage-level rules to inherit or override pipeline-level automations.
- **Constraint:** Adds a check to ensure `probability` values are within `[0,1]`.
- **Indexes:** Replaces the old ordering index with a unique index on `(pipeline_id, display_order)`.

List

- **New fields:** Adds `processing_type` (enum) with default `STATIC` and `is_archived` (boolean). These fields allow differentiating static versus dynamic lists and support list archiving.

Contact

- **New fields:** Adds `owned_by_user_id` and `owned_by_group_id`. These link a contact to the owning user or group. Foreign keys reference the tenant's user and group shadow tables and use `SET NULL` on delete. Indexes support filtering by owner.

Company

- **New fields:** Adds `owned_by_user_id` and `owned_by_group_id` with similar foreign keys and indexes as contacts. This allows companies to be assigned to account owners or teams.

Deal

- **New owner/assignment fields:** Adds `owned_by_user_id`, `owned_by_group_id`, `assigned_user_id` and `assigned_group_id`, enabling separate notions of ownership and active assignment.

- **Deal categorization:** Adds `deal_type` (enum), `forecast_probability` (numeric) and `close_date`. These fields support forecasting and pipeline reporting.
- **Constraints and indexes:** Foreign keys link owner and assignment fields to tenant shadow tables with `SET NULL` on delete. Indexes support queries by owner and assignee.

Lead

- **New fields:** Adds `owned_by_user_id` and `owned_by_group_id`. Ownership allows leads to be assigned to specific reps or teams. Foreign keys reference tenant shadow tables and use `SET NULL` on delete.

Activity

- **Column rename:** Renames the existing `type` column to `activity_type`. This prevents name collisions with reserved keywords and clarifies intent.
- **New fields:** Adds `activity_at` (timestamp), `created_by_user_id` (foreign key to tenant_user_shadow), `assigned_group_id` (foreign key to tenant_group_shadow) and `details_json` (JSONB). These fields allow recording when the activity occurred, who created it, group assignment and arbitrary details.
- **Indexes:** Adds indexes on `assigned_user_id`, `assigned_group_id` and `created_by_user_id` to support efficient filtering.

List/object conversions

- **Enum conversion:** Converts `list.object_type` and `list_membership.member_type` from free-form text to the new `list_object_type` enum. Before conversion, existing values are normalized to uppercase.
- **Indexes:** Adds helpful indexes on `(list_id, member_type)` and `(tenant_id, object_type)` to optimize queries by type.
