Middle East Technical University

Faculty of Arts and Sciences, Department of Statistics

Fall 2025-2026

STAT311

Modern Database Systems

Term Project

Dairy Enterprise Resource Planning Information System

By

Bartu Batum 2614469
Ozan Durmaz 2614584
Serhan Yavlal 2499820
Tuna Gökyokuş 2614626
Cihangir Osman Dokucu 2614576

Contents

Introduction

The dairy industry is one of the most critical sectors in food production, requiring precise management. Different dairy products should be manufactured, prepared and stored in different yet precise ways. As an enterprise, a dairy company also needs to manage its raw materials, production processes, inventory, logistics, and financial operations. However, most existing systems focus on isolated aspects of management and lack integration for a comprehensive and centralized enterprise resource planning.

In the Dairy Products Database Management System, our aim is to build a database structure that can achieve the following goals:

- **Comprehensive Supply Chain Management**: Track milk intake from multiple suppliers with quality metrics and pooling capabilities.
- **Production Process Control**: Manage production orders, steps, and co-production of multiple products (main products and byproducts)
- **Inventory Management**: Track batches with weak entity design, support batch splitting, and maintain location history
- **Sales and Logistics**: Handle sales orders, invoicing, shipments with multi-invoice capability, and fulfillment tracking
- **Financial Integration**: Link financial records to various sources (invoices, machines, payroll, etc.) with posting status control
- **Asset Management**: Track machines, vehicles, carriers, and their maintenance histories
- **Party Management**: Unified model for persons and organizations that can act as suppliers, customers, or drivers

Our database is built using PostgeSQL 18.0 and implements advanced database features including:

- Temporal data management with non-overlapping interval constraints using GiST indexes. GiST (Generalized Search Tree) indexes allow us to index non-linear relationships such as time intervals. For example:

```
ALTER TABLE public.vehicle_carrier_assignments
ADD CONSTRAINT ex_vca_no_overlap
EXCLUDE USING gist (vehicle_id WITH =, tstzrange(start_time,
    end_time) WITH &&);
```

  This code block allows us to ensure that two vehicles with the same id, i.e. the exact same vehicle cannot be in two different carrier assignments that happen at the exact same time. A vehicle cannot be in two places at once. Furthermore, tstzrange(start_time, end_time) by default is [start, end). It includes the starting time, but not ending time. With this, we can also ensure that a vehicle can embark on a new carrier assignment as soon as the previous carrier assignment ends.

- Deferred constraint triggers for complex business rules
- Automatic computation of invoice totals through triggers
- Immutability constraints for posted financial records

```
CREATE OR REPLACE FUNCTION public.
   fn_block_changes_when_posted()
RETURNS trigger LANGUAGE plpgsql AS $$
BEGIN
    IF (TG_OP = 'UPDATE' OR TG_OP = 'DELETE') AND OLD.status
       = 'POSTED' THEN
        RAISE EXCEPTION 'Finance record % is POSTED and
           immutable.', OLD.record_id;
    END IF;
        RETURN COALESCE(NEW, OLD);
END $$;
```

- Referential integrity with appropriate cascade behaviours: The child's existence is functionally dependent on the parent.
- Weak entity design for batches and invoice items: A batch is the batch of a given product, on a given date, with a given split. Furthermore, invoice items are parts of an invoice, not entities on their own.

The system demonstrates professional database design principles including normalization, constraint enforcement, and data integrity mechanisms that ensure reliable operation in a production environment.

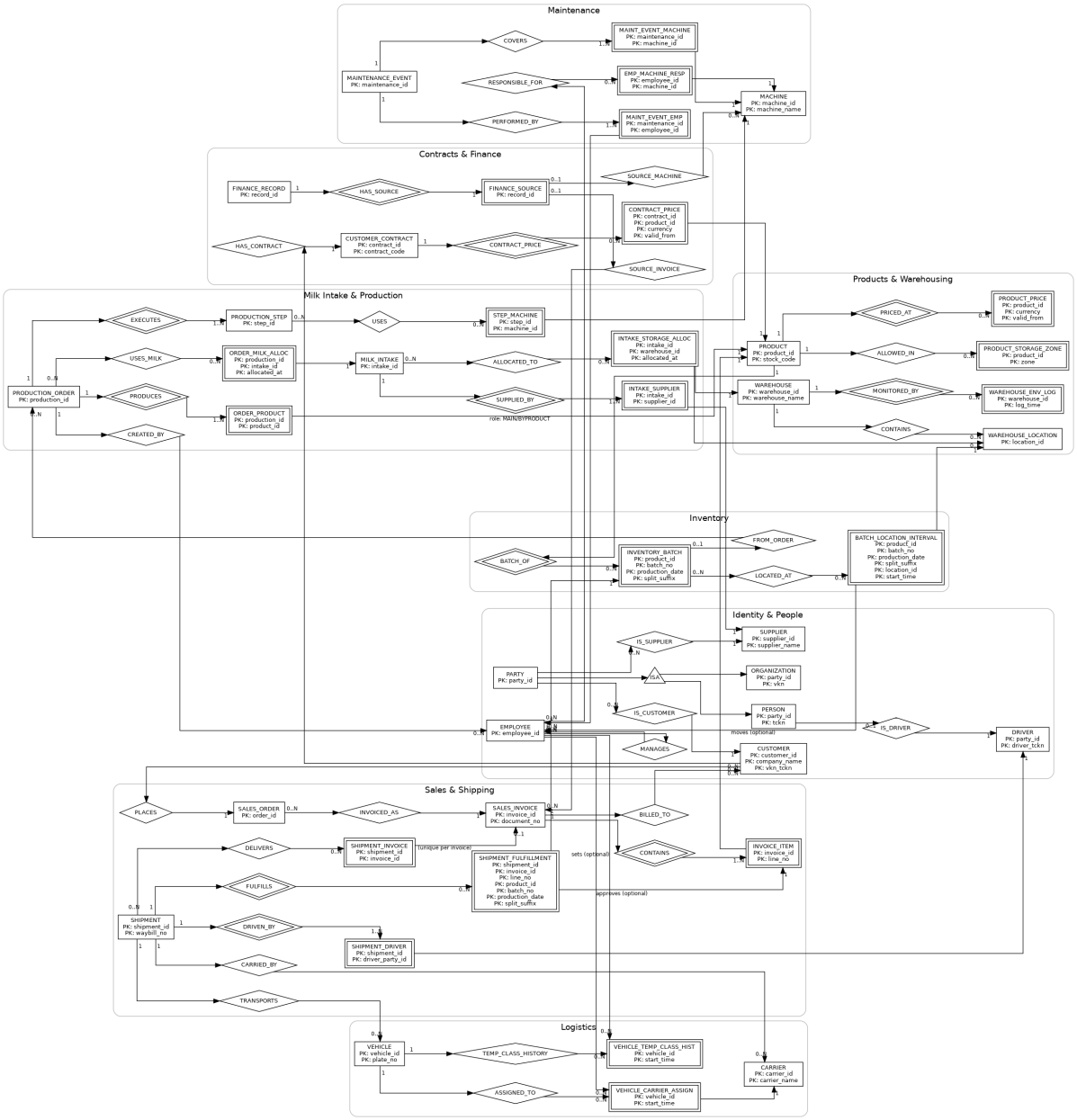
Data Disclaimer

Due to KVKK data protection constraints, we were not able to use real operational data from the company we collaborated with. Instead, we generated representative sample data using Cursor AI and populated the tables accordingly. The generated dataset is designed to be realistic and to satisfy all constraints and business rules, but it does not contain any personally identifiable information or proprietary company records.

ER Diagram

1. **Party Model**: Parties (persons/organizations), Persons, Organizations
2. **Employee Management**: Employees (self-referencing for managers)
3. **Supply Chain**: Suppliers, Customers, Carriers, Vehicles
4. **Product Management**: Products, Product Storage Zones, Product Prices
5. **Warehouse Management**: Warehouses, Warehouse Locations, Warehouse Environment Logs
6. **Machine Management**: Machines, Employee Machine Responsibility, Maintenance Events
7. **Milk Intake**: Milk Intakes, Milk Intake Suppliers, Milk Intake Storage Allocations
8. **Production**: Production Orders, Production Order Products, Production Steps, Production Milk Allocations
9. **Inventory**: Inventory Batches (weak entity), Batch Location Intervals
10. **Sales**: Sales Orders, Sales Invoices, Invoice Items (weak entity)
11. **Logistics**: Shipments, Shipment Invoices, Shipment Drivers, Shipment Fulfillments
12. **Pricing**: Customer Contracts, Contract Prices
13. **Finance**: Finance Records, Finance Sources

The relationships include one-to-many, many-to-many, and weak entity relationships with appropriate cardinalities and participation constraints.


Maintenance
COVERS
MAINT_EVENT_MACHINE
PK: maintenance_id
PK: machine_id
MAINTENANCE_EVENT
PK: maintenance_id
RESPONSIBLE_FOR
EMP_MACHINE_RESP
PK: employee_id
PK: machine_id
MACHINE
PK: machine_id
PK: machine_name
PERFORMED_BY
MAINT_EVENT_EMP
PK: maintenance_id
PK: employee_id
Contracts & Finance
FINANCE_RECORD
PK: record_id
HAS_SOURCE
FINANCE_SOURCE
PK: record_id
SOURCE_MACHINE
CONTRACT_PRICE
PK: contract_id
PK: product_id
PK: currency
PK: valid_from
HAS_CONTRACT
CUSTOMER_CONTRACT
PK: contract_id
PK: contract_code
CONTRACT_PRICE
SOURCE_INVOICE
Products & Warehousing
PRICED_AT
PRODUCT_PRICE
PK: product_id
PK: currency
PK: valid_from
PRODUCT
PK: product_id
PK: stock_code
ALLOWED_IN
PRODUCT_STORAGE_ZONE
PK: product_id
PK: zone
WAREHOUSE
PK: warehouse_id
PK: warehouse_name
MONITORED_BY
WAREHOUSE_ENV_LOG
PK: warehouse_id
PK: log_time
CONTAINS
WAREHOUSE_LOCATION
PK: location_id
Milk Intake & Production
EXECUTES
PRODUCTION_STEP
PK: step_id
USES
STEP_MACHINE
PK: step_id
PK: machine_id
USES_MILK
ORDER_MILK_ALLOC
PK: production_id
PK: intake_id
PK: allocated_at
MILK_INTAKE
PK: intake_id
ALLOCATED_TO
INTAKE_STORAGE_ALLOC
PK: intake_id
PK: warehouse_id
PK: allocated_at
PRODUCTION_ORDER
PK: production_id
PRODUCES
SUPPLIED_BY
INTAKE_SUPPLIER
PK: intake_id
PK: supplier_id
role: MAIN/BYPRODUCT
ORDER_PRODUCT
PK: production_id
PK: product_id
CREATED_BY
Inventory
FROM_ORDER
BATCH_OF
INVENTORY_BATCH
PK: product_id
PK: batch_no
PK: production_date
PK: split_suffix
LOCATED_AT
BATCH_LOCATION_INTERVAL
PK: product_id
PK: batch_no
PK: production_date
PK: split_suffix
PK: location_id
PK: start_time
Identity & People
IS_SUPPLIER
SUPPLIER
PK: supplier_id
PK: supplier_name
PARTY
PK: party_id
ISA
ORGANIZATION
PK: party_id
PK: vkn
IS_CUSTOMER
PERSON
PK: party_id
PK: tckn
moves (optional)
IS_DRIVER
DRIVER
PK: party_id
PK: driver_tckn
EMPLOYEE
PK: employee_id
MANAGES
CUSTOMER
PK: customer_id
PK: company_name
PK: vkn_tckn
Sales & Shipping
PLACES
SALES_ORDER
PK: order_id
INVOICED_AS
SALES_INVOICE
PK: invoice_id
PK: document_no
BILLED_TO
DELIVERS
SHIPMENT_INVOICE
PK: shipment_id
PK: invoice_id
1 unique per invoice
SHIPMENT_FULFILLMENT
PK: shipment_id
PK: invoice_id
PK: line_no
PK: product_id
PK: batch_no
PK: production_date
PK: split_suffix
sets (optional)
CONTAINS
INVOICE_ITEM
PK: invoice_id
PK: line_no
FULFILLS
approves (optional)
SHIPMENT
PK: shipment_id
PK: waybill_no
DRIVEN_BY
SHIPMENT_DRIVER
PK: shipment_id
PK: driver_party_id
CARRIED_BY
TRANSPORTS
Logistics
VEHICLE
PK: vehicle_id
PK: plate_no
TEMP_CLASS_HISTORY
VEHICLE_TEMP_CLASS_HIST
PK: vehicle_id
PK: start_time
CARRIER
PK: carrier_id
PK: carrier_name
ASSIGNED_TO
VEHICLE_CARRIER_ASSIGN
PK: vehicle_id
PK: start_time

Schema Design

The schema and table properties are developed in PostgreSQL 18.0.

ENUM Types

An ENUM restricts a column to one value from a predefined list, enforced by PostgreSQL itself. ENUM types in our database are:

```
CREATE TYPE public.storage_zone AS ENUM ('PLUS_4', 'MINUS_18', '
    AMBIENT');
CREATE TYPE public.milk_type AS ENUM ('RAW', 'PASTEURIZED');
CREATE TYPE public.expense_type AS ENUM ('FIXED', 'DYNAMIC', '
    INCOME');
CREATE TYPE public.machine_status AS ENUM ('RUNNING', '
    MAINTENANCE', 'BROKEN', 'IDLE');
CREATE TYPE public.vehicle_temp_class AS ENUM ('COLD_PLUS_4', '
    FROZEN_MINUS_18', 'AMBIENT');
CREATE TYPE public.process_step_type AS ENUM (
  'PASTEURIZATION', 'FERMENTATION', 'STABILIZATION',
  'SALTING', 'MATURATION', 'PORTIONING', 'PACKAGING'
);

CREATE TYPE public.party_kind AS ENUM ('PERSON', 'ORGANIZATION');
CREATE TYPE public.contract_product_role AS ENUM ('MAIN', '
    BYPRODUCT');
CREATE TYPE public.finance_posting_status AS ENUM ('DRAFT', '
    POSTED', 'VOID');
CREATE TYPE public.finance_source_kind AS ENUM (
  'SALES_INVOICE',
  'MACHINE',
  'PAYROLL',
  'RENT_CONTRACT',
  'OTHER'
);
```

Parties Schema

```sql
CREATE TABLE public.parties (
  party_id    BIGSERIAL PRIMARY KEY,
  party_kind  public.party_kind NOT NULL,
  created_at  TIMESTAMPTZ NOT NULL DEFAULT now()
);

CREATE TABLE public.persons (
  party_id   BIGINT PRIMARY KEY REFERENCES public.parties(
     party_id) ON DELETE CASCADE,
  tckn       VARCHAR UNIQUE,
  full_name  VARCHAR NOT NULL,
  phone      VARCHAR,
  email      VARCHAR UNIQUE
);

CREATE TABLE public.organizations (
  party_id    BIGINT PRIMARY KEY REFERENCES public.parties(
     party_id) ON DELETE CASCADE,
  legal_name  VARCHAR NOT NULL,
  tax_office  VARCHAR,
  vkn         VARCHAR UNIQUE,
  phone       VARCHAR
);
```

Column Name	Data Type	Allow Nulls	Description
parties			
party_id	BIGSERIAL	NO	Primary key, auto-incrementing
party_kind	party_kind	NO	ENUM: PERSON or ORGANIZATION
created_at	TIMESTAMPTZ	NO	Timestamp of record creation (default: now())
persons			
party_id	BIGINT	NO	Primary key, references parties(party_id) ON DELETE CASCADE
tckn	VARCHAR	YES	Turkish ID number (unique)
full_name	VARCHAR	NO	Full name of the person
phone	VARCHAR	YES	Phone number
email	VARCHAR	YES	Email address (unique)
organizations			
party_id	BIGINT	NO	Primary key, references parties(party_id) ON DELETE CASCADE

Column Name	Data Type	Allow Nulls	Description
legal_name	VARCHAR	NO	Legal name of the organization
tax_office	VARCHAR	YES	Tax office name
vkn	VARCHAR	YES	Tax identification number (unique)
phone	VARCHAR	YES	Phone number

Employees Schema

```sql
CREATE TABLE public.employees (
  employee_id          BIGSERIAL PRIMARY KEY,
  full_name            VARCHAR(150) NOT NULL,
  role                 VARCHAR(120) NOT NULL,
  department           VARCHAR(80)  NOT NULL,
  location             VARCHAR(120),
  salary_monthly       NUMERIC(12,2) NOT NULL CHECK (
     salary_monthly >= 0),
  phone                VARCHAR(40),
  email                VARCHAR(120) UNIQUE,
  manager_id           BIGINT REFERENCES public.employees(
     employee_id) ON DELETE SET NULL,
  CONSTRAINT ck_employees_manager_not_self CHECK (manager_id IS
     NULL OR manager_id <> employee_id)
);

CREATE INDEX idx_employees_dept ON public.employees(department);
```

Column Name	Data Type	Allow Nulls	Description
employee_id	BIGSERIAL	NO	Primary key, auto-incrementing
full_name	VARCHAR(150)	NO	Full name of the employee
role	VARCHAR(120)	NO	Job role/title
department	VARCHAR(80)	NO	Department name
location	VARCHAR(120)	YES	Work location
salary_monthly	NUMERIC(12,2)	NO	Monthly salary (CHECK: $\geq$ 0)
phone	VARCHAR(40)	YES	Phone number
email	VARCHAR(120)	YES	Email address (unique)
manager_id	BIGINT	YES	References employees(employee_id) ON DELETE SET NULL

Constraints:

- `ck_employees_manager_not_self`: Prevents an employee from being their own manager
- Index on `department` for efficient queries

Suppliers Schema

Suppliers are role tables that reference parties. A party can be both a supplier and a customer.

```
CREATE TABLE public.suppliers (
  supplier_id    BIGSERIAL PRIMARY KEY,
  party_id       BIGINT NOT NULL REFERENCES public.parties(
     party_id) ON DELETE RESTRICT,
  supplier_name  VARCHAR(200) NOT NULL UNIQUE,
  phone          VARCHAR(40),
  city           VARCHAR(80),
  district       VARCHAR(80)
);
```

Column Name	Data Type	Allow Nulls	Description
supplier_id	BIGSERIAL	NO	Primary key
party_id	BIGINT	NO	References parties(party_id) ON DELETE RESTRICT
supplier_name	VARCHAR(200)	NO	Supplier name (unique)
phone	VARCHAR(40)	YES	Phone number
city	VARCHAR(80)	YES	City
district	VARCHAR(80)	YES	District

Customers Schema

```
CREATE TABLE public.customers (
  customer_id    BIGSERIAL PRIMARY KEY,
  party_id       BIGINT NOT NULL REFERENCES public.parties(
     party_id) ON DELETE RESTRICT,
  company_name   VARCHAR(200) NOT NULL UNIQUE,
  address_text   TEXT NOT NULL,
  postal_code    VARCHAR(12),
  city           VARCHAR(80),
  district       VARCHAR(80),
  tax_office     VARCHAR(120),
  vkn_tckn       VARCHAR(20) UNIQUE
);
```

Column Name	Data Type	Allow Nulls	Description
customer_id	BIGSERIAL	NO	Primary key
party_id	BIGINT	NO	References parties(party_id) ON DELETE RESTRICT
company_name	VARCHAR(200)	NO	Company name (unique)
address_text	TEXT	NO	Full address
postal_code	VARCHAR(12)	YES	Postal code
city	VARCHAR(80)	YES	City
district	VARCHAR(80)	YES	District
tax_office	VARCHAR(120)	YES	Tax office name
vkn_tckn	VARCHAR(20)	YES	Tax/ID number (unique)

Carriers Schema

```
CREATE TABLE public.carriers (
  carrier_id    BIGSERIAL PRIMARY KEY,
  carrier_name  VARCHAR(200) NOT NULL UNIQUE,
  phone         VARCHAR(40)
);
```

Column Name	Data Type	Allow Nulls	Description
carrier_id	BIGSERIAL	NO	Primary key
carrier_name	VARCHAR(200)	NO	Carrier name (unique)
phone	VARCHAR(40)	YES	Phone number

Vehicles Schema

```
CREATE TABLE public.vehicles (
  vehicle_id       BIGSERIAL PRIMARY KEY,
  plate_no         VARCHAR(20) NOT NULL UNIQUE,
  vehicle_type     VARCHAR(80) NOT NULL,
  temp_class       public.vehicle_temp_class NOT NULL,
  max_capacity_kg  NUMERIC(12,2) CHECK (max_capacity_kg >= 0)
);

CREATE TABLE public.vehicle_carrier_assignments (
  vehicle_id              BIGINT NOT NULL REFERENCES public.
     vehicles(vehicle_id) ON DELETE CASCADE,
  carrier_id              BIGINT NOT NULL REFERENCES public.
     carriers(carrier_id) ON DELETE RESTRICT,
  start_time              TIMESTAMPTZ NOT NULL,
  end_time                TIMESTAMPTZ,
  approved_by_employee_id BIGINT REFERENCES public.employees(
     employee_id) ON DELETE SET NULL,
  PRIMARY KEY (vehicle_id, start_time),
  CONSTRAINT ck_vca_time CHECK (end_time IS NULL OR end_time >
     start_time)
);

ALTER TABLE public.vehicle_carrier_assignments
  ADD CONSTRAINT ex_vca_no_overlap
  EXCLUDE USING gist (vehicle_id WITH =, tstzrange(start_time,
     end_time) WITH &&);

CREATE TABLE public.vehicle_temp_class_history (
  vehicle_id          BIGINT NOT NULL REFERENCES public.vehicles(
     vehicle_id) ON DELETE CASCADE,
  temp_class          public.vehicle_temp_class NOT NULL,
  start_time          TIMESTAMPTZ NOT NULL,
  end_time            TIMESTAMPTZ,
  set_by_employee_id  BIGINT REFERENCES public.employees(
     employee_id) ON DELETE SET NULL,
  PRIMARY KEY (vehicle_id, start_time),
  CONSTRAINT ck_vtch_time CHECK (end_time IS NULL OR end_time >
     start_time)
);

ALTER TABLE public.vehicle_temp_class_history
  ADD CONSTRAINT ex_vtch_no_overlap
  EXCLUDE USING gist (vehicle_id WITH =, tstzrange(start_time,
     end_time) WITH &&);
```

Column Name	Data Type	Allow Nulls	Description
vehicle_id	BIGSERIAL	NO	Primary key
plate_no	VARCHAR(20)	NO	License plate number (unique)
vehicle_type	VARCHAR(80)	NO	Type of vehicle (e.g., "Truck", "Van")
temp_class	vehicle_temp_class	NO	Temperature classification ENUM
max_capacity_kg	NUMERIC(12,2)	YES	Maximum capacity in kilograms (CHECK: ≥ 0)

Related Tables:

- `vehicle_carrier_assignments`: Historical assignment of vehicles to carriers with temporal intervals
- `vehicle_temp_class_history`: History of temperature class changes

Products Schema

```
CREATE TABLE public.products (
  product_id    BIGSERIAL PRIMARY KEY,
  product_name  VARCHAR(180) NOT NULL UNIQUE,
  category      VARCHAR(80) NOT NULL,
  stock_code    VARCHAR(50) NOT NULL UNIQUE,
  vat_rate      NUMERIC(5,2) NOT NULL DEFAULT 1 CHECK (vat_rate
     >= 0),
  base_unit     VARCHAR(20) NOT NULL DEFAULT 'KG'
);

CREATE TABLE public.product_storage_zones (
  product_id BIGINT NOT NULL REFERENCES public.products(
     product_id) ON DELETE CASCADE,
  zone       public.storage_zone NOT NULL,
  PRIMARY KEY (product_id, zone)
);

CREATE TABLE public.product_prices (
  product_id   BIGINT NOT NULL REFERENCES public.products(
     product_id) ON DELETE CASCADE,
  currency     VARCHAR(10) NOT NULL DEFAULT 'TRY',
  valid_from   DATE NOT NULL DEFAULT CURRENT_DATE,
  valid_to     DATE,
  unit_price   NUMERIC(12,2) NOT NULL CHECK (unit_price >= 0),
  PRIMARY KEY (product_id, currency, valid_from),
  CONSTRAINT ck_pp_valid CHECK (valid_to IS NULL OR valid_to >
     valid_from)
);

ALTER TABLE public.product_prices
  ADD CONSTRAINT ex_pp_no_overlap
  EXCLUDE USING gist (
    product_id WITH =,
    currency WITH =,
    daterange(valid_from, COALESCE(valid_to, 'infinity'::date), '
       []') WITH &&
  );
```

Column Name	Data Type	Allow Nulls	Description
product_id	BIGSERIAL	NO	Primary key
product_name	VARCHAR(180)	NO	Product name (unique)
category	VARCHAR(80)	NO	Product category
stock_code	VARCHAR(50)	NO	Stock keeping unit code (unique)
vat_rate	NUMERIC(5,2)	NO	VAT rate percentage (default: 1, CHECK: ≥ 0)
base_unit	VARCHAR(20)	NO	Base unit of measurement (default: 'KG')

Related Tables:

- `product_storage_zones`: Allowed storage zones for each product (M:N relationship)
- `product_prices`: Price history with temporal validity

Warehouses Schema

```
CREATE TABLE public.warehouses (
  warehouse_id    BIGSERIAL PRIMARY KEY,
  warehouse_name  VARCHAR(120) NOT NULL UNIQUE,
  zone            public.storage_zone NOT NULL,
  address_text    TEXT
);

CREATE TABLE public.warehouse_locations (
  location_id    BIGSERIAL PRIMARY KEY,
  warehouse_id   BIGINT NOT NULL REFERENCES public.warehouses(
     warehouse_id) ON DELETE CASCADE,
  location_code  VARCHAR(40) NOT NULL,
  UNIQUE (warehouse_id, location_code)
);

CREATE TABLE public.warehouse_env_logs (
  warehouse_id    BIGINT NOT NULL REFERENCES public.warehouses(
     warehouse_id) ON DELETE CASCADE,
  log_time        TIMESTAMPTZ NOT NULL DEFAULT now(),
  temperature_c   NUMERIC(6,2),
  humidity_pct    NUMERIC(6,2),
  note            VARCHAR(200),
  PRIMARY KEY (warehouse_id, log_time)
);

CREATE INDEX idx_envlogs_wh_time ON public.warehouse_env_logs(
   warehouse_id, log_time DESC);
```

Column Name	Data Type	Allow Nulls	Description
warehouse_id	BIGSERIAL	NO	Primary key
warehouse_name	VARCHAR(120)	NO	Warehouse name (unique)
zone	storage_zone	NO	Storage zone ENUM
address_text	TEXT	YES	Warehouse address

Related Tables:

- `warehouse_locations`: Specific locations within warehouses
- `warehouse_env_logs`: Environmental monitoring logs (weak entity by warehouse_id, log_time)

Machines Schema

```
CREATE TABLE public.machines (
  machine_id           BIGSERIAL PRIMARY KEY,
  machine_name         VARCHAR(180) NOT NULL UNIQUE,
  brand                VARCHAR(100),
  status               public.machine_status NOT NULL DEFAULT 'IDLE',
  zone                 public.storage_zone,
  installed_location   VARCHAR(120),
  service_company      VARCHAR(150),
  last_maintenance_date DATE,
  next_maintenance_date DATE
);

CREATE INDEX idx_machines_status ON public.machines(status);

CREATE TABLE public.employee_machine_responsibility (
  employee_id  BIGINT NOT NULL REFERENCES public.employees(employee_id) ON DELETE RESTRICT,
  machine_id   BIGINT NOT NULL REFERENCES public.machines(machine_id) ON DELETE RESTRICT,
  assigned_at  TIMESTAMPTZ NOT NULL DEFAULT now(),
  PRIMARY KEY (employee_id, machine_id)
);

CREATE TABLE public.machine_maintenance_events (
  maintenance_id    BIGSERIAL PRIMARY KEY,
  maintenance_date  DATE NOT NULL DEFAULT CURRENT_DATE,
  maintenance_type  VARCHAR(80) NOT NULL,
  description       TEXT,
  cost_amount       NUMERIC(12,2) CHECK (cost_amount >= 0),
  scheduled         BOOLEAN NOT NULL DEFAULT TRUE
);

CREATE TABLE public.maintenance_event_machines (
  maintenance_id BIGINT NOT NULL REFERENCES public.machine_maintenance_events(maintenance_id) ON DELETE CASCADE,
  machine_id     BIGINT NOT NULL REFERENCES public.machines(machine_id) ON DELETE RESTRICT,
  PRIMARY KEY (maintenance_id, machine_id)
);

CREATE TABLE public.maintenance_event_employees (
  maintenance_id BIGINT NOT NULL REFERENCES public.machine_maintenance_events(maintenance_id) ON DELETE CASCADE,
  employee_id    BIGINT NOT NULL REFERENCES public.employees(employee_id) ON DELETE RESTRICT,
  PRIMARY KEY (maintenance_id, employee_id)
```

```
41 );
```

Column Name	Data Type	Allow Nulls	Description
machine_id	BIGSERIAL	NO	Primary key
machine_name	VARCHAR(180)	NO	Machine name (unique)
brand	VARCHAR(100)	YES	Brand name
status	machine_status	NO	Status ENUM (default: 'IDLE')
zone	storage_zone	YES	Storage zone where machine is located
installed_location	VARCHAR(120)	YES	Installation location
service_company	VARCHAR(150)	YES	Service provider company
last_maintenance_date	DATE	YES	Last maintenance date
next_maintenance_date	DATE	YES	Next scheduled maintenance

Related Tables:

- `employee_machine_responsibility`: M:N relationship for employee-machine assignments
- `machine_maintenance_events`: Maintenance events (can involve multiple machines and employees)

Milk Intakes Schema

```
CREATE TABLE public.milk_intakes (
  intake_id          BIGSERIAL PRIMARY KEY,
  intake_time        TIMESTAMPTZ NOT NULL DEFAULT now(),
  milk_kind          public.milk_type NOT NULL,
  quantity_liters    NUMERIC(12,2) NOT NULL CHECK (quantity_liters > 0),
  ph_level           NUMERIC(4,2),
  fat_ratio_pct      NUMERIC(5,2),
  total_cost         NUMERIC(12,2) NOT NULL CHECK (total_cost >= 0),
  storage_target     VARCHAR(80),
  note               VARCHAR(200),
  density_kg_per_l   NUMERIC(12,6) NOT NULL CHECK (density_kg_per_l > 0)
);

CREATE INDEX idx_milk_intakes_time ON public.milk_intakes(intake_time DESC);

CREATE TABLE public.milk_intake_suppliers (
  intake_id          BIGINT NOT NULL REFERENCES public.milk_intakes(intake_id) ON DELETE CASCADE,
  supplier_id        BIGINT NOT NULL REFERENCES public.suppliers(supplier_id) ON DELETE RESTRICT,
  contributed_liters NUMERIC(12,2) NOT NULL CHECK (contributed_liters > 0),
  cost_share         NUMERIC(12,6) NOT NULL CHECK (cost_share >= 0),
  supplier_ph_level  NUMERIC(4,2),
  supplier_fat_ratio_pct NUMERIC(5,2),
  PRIMARY KEY (intake_id, supplier_id)
);

CREATE OR REPLACE FUNCTION public.fn_milk_intake_must_have_supplier()
RETURNS trigger LANGUAGE plpgsql AS $$
DECLARE v_cnt INT;
BEGIN
  SELECT COUNT(*) INTO v_cnt FROM public.milk_intake_suppliers
    WHERE intake_id = COALESCE(NEW.intake_id, OLD.intake_id);
  IF v_cnt = 0 THEN
    RAISE EXCEPTION 'Milk intake % must have at least one supplier.', COALESCE(NEW.intake_id, OLD.intake_id);
  END IF;
  RETURN NULL;
END $$;

CREATE CONSTRAINT TRIGGER trg_intake_has_supplier
AFTER INSERT OR UPDATE OR DELETE ON public.milk_intake_suppliers
```

```
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW EXECUTE FUNCTION public.
   fn_milk_intake_must_have_supplier();

CREATE TABLE public.milk_intake_storage_allocations (
  intake_id        BIGINT NOT NULL REFERENCES public.milk_intakes
     (intake_id) ON DELETE CASCADE,
  warehouse_id     BIGINT NOT NULL REFERENCES public.warehouses(
     warehouse_id) ON DELETE RESTRICT,
  location_id      BIGINT REFERENCES public.warehouse_locations(
     location_id) ON DELETE RESTRICT,
  allocated_liters NUMERIC(12,2) NOT NULL CHECK (allocated_liters
      > 0),
  allocated_at     TIMESTAMPTZ NOT NULL DEFAULT now(),
  PRIMARY KEY (intake_id, warehouse_id, allocated_at)
);
```

Column Name	Data Type	Allow Nulls	Description
intake_id	BIGSERIAL	NO	Primary key
intake_time	TIMESTAMPTZ	NO	Intake timestamp (default: now())
milk_kind	milk_type	NO	Milk type ENUM (RAW or PASTEURIZED)
quantity_liters	NUMERIC(12,2)	NO	Quantity in liters (CHECK: > 0)
ph_level	NUMERIC(4,2)	YES	pH level
fat_ratio_pct	NUMERIC(5,2)	YES	Fat ratio percentage
total_cost	NUMERIC(12,2)	NO	Total cost (CHECK: ≥ 0)
storage_target	VARCHAR(80)	YES	Storage target description
note	VARCHAR(200)	YES	Additional notes
density_kg_per_l	NUMERIC(12,6)	NO	Density in kg per liter (CHECK: > 0)

Related Tables:

- `milk_intake_suppliers`: M:N relationship for milk pooling from multiple suppliers with individual quality metrics and cost shares
- `milk_intake_storage_allocations`: Allocation of intake to specific warehouse locations

Business Rule: Each milk intake must have at least one supplier (enforced by deferred constraint trigger).

Production Orders Schema

```
CREATE TABLE public.production_orders (
  production_id            BIGSERIAL PRIMARY KEY,
  created_time             TIMESTAMPTZ NOT NULL DEFAULT now(),
  planned_date             DATE NOT NULL DEFAULT CURRENT_DATE,
  fat_standard             VARCHAR(40),
  status                   VARCHAR(40) NOT NULL DEFAULT 'PLANNED'
    ,
  created_by_employee_id   BIGINT NOT NULL REFERENCES public.
    employees(employee_id) ON DELETE RESTRICT
);

CREATE INDEX idx_prod_orders_status ON public.production_orders(
   status);

CREATE TABLE public.production_order_products (
  production_id   BIGINT NOT NULL REFERENCES public.
    production_orders(production_id) ON DELETE CASCADE,
  product_id      BIGINT NOT NULL REFERENCES public.products(
    product_id) ON DELETE RESTRICT,
  role            public.contract_product_role NOT NULL,
  planned_qty_kg  NUMERIC(12,3) CHECK (planned_qty_kg IS NULL OR
    planned_qty_kg >= 0),
  actual_qty_kg   NUMERIC(12,3) CHECK (actual_qty_kg IS NULL OR
    actual_qty_kg >= 0),
  yield_pct       NUMERIC(6,3)  CHECK (yield_pct IS NULL OR (
    yield_pct >= 0 AND yield_pct <= 100)),
  PRIMARY KEY (production_id, product_id)
);

CREATE TABLE public.production_milk_allocations (
  production_id    BIGINT NOT NULL REFERENCES public.
    production_orders(production_id) ON DELETE CASCADE,
  intake_id        BIGINT NOT NULL REFERENCES public.milk_intakes
    (intake_id) ON DELETE RESTRICT,
  allocated_liters NUMERIC(12,2) NOT NULL CHECK (allocated_liters
     > 0),
  allocated_at     TIMESTAMPTZ NOT NULL DEFAULT now(),
  PRIMARY KEY (production_id, intake_id, allocated_at)
);

CREATE TABLE public.production_steps (
  step_id        BIGSERIAL PRIMARY KEY,
  production_id  BIGINT NOT NULL REFERENCES public.
    production_orders(production_id) ON DELETE CASCADE,
  step_no        INT NOT NULL,
  step_type      public.process_step_type NOT NULL,
  start_time     TIMESTAMPTZ DEFAULT now(),
  end_time       TIMESTAMPTZ,
  note           VARCHAR(200),
```

```
38    UNIQUE (production_id, step_no),
39    CONSTRAINT ck_step_time CHECK (end_time IS NULL OR end_time >=
          start_time)
40  );
41
42  CREATE INDEX idx_steps_prod ON public.production_steps(
        production_id);
43
44  CREATE TABLE public.production_step_machines (
45    step_id    BIGINT NOT NULL REFERENCES public.production_steps(
          step_id) ON DELETE CASCADE,
46    machine_id BIGINT NOT NULL REFERENCES public.machines(
          machine_id) ON DELETE RESTRICT,
47    PRIMARY KEY (step_id, machine_id)
48  );
49
50  CREATE OR REPLACE FUNCTION public.
        fn_production_order_must_have_step()
51  RETURNS trigger LANGUAGE plpgsql AS $$
52  DECLARE v_pid BIGINT;
53  DECLARE v_cnt INT;
54  BEGIN
55    v_pid := COALESCE(NEW.production_id, OLD.production_id);
56    SELECT COUNT(*) INTO v_cnt FROM public.production_steps WHERE
          production_id = v_pid;
57    IF v_cnt = 0 THEN
58      RAISE EXCEPTION 'Production order % must have at least one
            step.', v_pid;
59    END IF;
60    RETURN NULL;
61  END $$;
62
63  CREATE CONSTRAINT TRIGGER trg_order_has_step
64  AFTER INSERT OR UPDATE OR DELETE ON public.production_steps
65  DEFERRABLE INITIALLY DEFERRED
66  FOR EACH ROW EXECUTE FUNCTION public.
        fn_production_order_must_have_step();
```

Column Name	Data Type	Allow Nulls	Description
production_id	BIGSERIAL	NO	Primary key
created_time	TIMESTAMPTZ	NO	Creation timestamp (default: now())
planned_date	DATE	NO	Planned production date (default: CURRENT_DATE)
fat_standard	VARCHAR(40)	YES	Fat standard specification
status	VARCHAR(40)	NO	Production status (default: 'PLANNED')
created_by_employee_id	BIGINT	NO	References employees(employee_id) ON DELETE RESTRICT

Related Tables:

- `production_order_products`: Co-production products with roles (MAIN/BYPRODUCT), planned/actual quantities, and yield percentages
- `production_milk_allocations`: Allocation of milk intakes to production orders
- `production_steps`: Ordered sequence of production steps

Business Rule: Each production order must have at least one step (enforced by deferred constraint trigger).

Inventory Batches Schema

```
CREATE TABLE public.inventory_batches (
  product_id       BIGINT NOT NULL REFERENCES public.products(
     product_id) ON DELETE RESTRICT,
  batch_no         VARCHAR(60) NOT NULL,
  production_date  DATE NOT NULL DEFAULT CURRENT_DATE,
  split_suffix     VARCHAR(16) NOT NULL DEFAULT 'A',
  produced_by_production_id BIGINT REFERENCES public.
     production_orders(production_id) ON DELETE SET NULL,
  quantity_kg      NUMERIC(12,3) NOT NULL CHECK (quantity_kg >= 0)
     ,
  expiry_date      DATE NOT NULL,
  entry_time       TIMESTAMPTZ NOT NULL DEFAULT now(),
  entry_temperature_c NUMERIC(6,2),
  entry_humidity_pct  NUMERIC(6,2),
  PRIMARY KEY (product_id, batch_no, production_date,
     split_suffix)
);

CREATE INDEX idx_batches_pid_exp ON public.inventory_batches(
   product_id, expiry_date);

CREATE TABLE public.batch_location_intervals (
  product_id       BIGINT NOT NULL,
  batch_no         VARCHAR(60) NOT NULL,
  production_date  DATE NOT NULL,
  split_suffix     VARCHAR(16) NOT NULL,
  location_id      BIGINT NOT NULL REFERENCES public.
     warehouse_locations(location_id) ON DELETE RESTRICT,
  start_time       TIMESTAMPTZ NOT NULL,
  end_time         TIMESTAMPTZ,
  qty_kg           NUMERIC(12,3) NOT NULL CHECK (qty_kg >= 0),
  moved_by_employee_id BIGINT REFERENCES public.employees(
     employee_id) ON DELETE SET NULL,
  move_temperature_c NUMERIC(6,2),
  move_humidity_pct  NUMERIC(6,2),
  PRIMARY KEY (product_id, batch_no, production_date,
     split_suffix, location_id, start_time),
  CONSTRAINT fk_bli_batch
    FOREIGN KEY (product_id, batch_no, production_date,
       split_suffix)
    REFERENCES public.inventory_batches(product_id, batch_no,
       production_date, split_suffix)
    ON DELETE CASCADE,
  CONSTRAINT ck_bli_time CHECK (end_time IS NULL OR end_time >
     start_time)
);

ALTER TABLE public.batch_location_intervals
  ADD CONSTRAINT ex_bli_no_overlap
```

```
  EXCLUDE USING gist (
    product_id WITH =,
    batch_no WITH =,
    production_date WITH =,
    split_suffix WITH =,
    location_id WITH =,
    tstzrange(start_time, end_time) WITH &&
  );

CREATE OR REPLACE FUNCTION public.
   fn_batch_current_location_qty_not_exceed_batch()
RETURNS trigger LANGUAGE plpgsql AS $$
DECLARE v_total NUMERIC;
DECLARE v_batch_qty NUMERIC;
BEGIN
  SELECT COALESCE(SUM(qty_kg),0)
    INTO v_total
  FROM public.batch_location_intervals
  WHERE product_id = NEW.product_id
    AND batch_no = NEW.batch_no
    AND production_date = NEW.production_date
    AND split_suffix = NEW.split_suffix
    AND end_time IS NULL;

  SELECT quantity_kg INTO v_batch_qty
  FROM public.inventory_batches
  WHERE product_id = NEW.product_id
    AND batch_no = NEW.batch_no
    AND production_date = NEW.production_date
    AND split_suffix = NEW.split_suffix;

  IF v_total > v_batch_qty THEN
    RAISE EXCEPTION 'Current location qty % exceeds batch qty %
       for batch (%,%,%,%).',
      v_total, v_batch_qty, NEW.product_id, NEW.batch_no, NEW.
         production_date, NEW.split_suffix;
  END IF;

  RETURN NEW;
END $$;

CREATE TRIGGER trg_bli_current_qty_check
AFTER INSERT OR UPDATE OF qty_kg, end_time ON public.
   batch_location_intervals
FOR EACH ROW EXECUTE FUNCTION public.
   fn_batch_current_location_qty_not_exceed_batch();
```

This is a **weak entity** identified by the combination of (product_id, batch_no, production_date, split_suffix).

Column Name	Data Type	Allow Nulls	Description
product_id	BIGINT	NO	Part of primary key, references products(product_id)
batch_no	VARCHAR(60)	NO	Part of primary key, batch number
production_date	DATE	NO	Part of primary key (default: CURRENT_DATE)
split_suffix	VARCHAR(16)	NO	Part of primary key, for batch splitting (default: 'A')
produced_by_production_id	BIGINT	YES	References production_orders(production_id) ON DELETE SET NULL
quantity_kg	NUMERIC(12,3)	NO	Batch quantity in kilograms (CHECK: ≥ 0)
expiry_date	DATE	NO	Expiry date
entry_time	TIMESTAMPTZ	NO	Entry timestamp (default: now())
entry_temperature_c	NUMERIC(6,2)	YES	Entry temperature
entry_humidity_pct	NUMERIC(6,2)	YES	Entry humidity percentage

Related Tables:

- `batch_location_intervals`: Temporal history of batch locations with quantities (supports splitting and movement tracking)

Constraints:

- Non-overlapping intervals for same (batch, location) using GiST exclusion constraint
- Trigger ensures current location quantities do not exceed batch quantity

Sales Orders Schema

```
CREATE TABLE public.sales_orders (
  order_id      BIGSERIAL PRIMARY KEY,
  customer_id   BIGINT NOT NULL REFERENCES public.customers(
     customer_id) ON DELETE RESTRICT,
  order_date    DATE NOT NULL DEFAULT CURRENT_DATE,
  note          VARCHAR(200)
);
```

Column Name	Data Type	Allow Nulls	Description
order_id	BIGSERIAL	NO	Primary key
customer_id	BIGINT	NO	References customers(customer_id) ON DELETE RESTRICT
order_date	DATE	NO	Order date (default: CURRENT_DATE)
note	VARCHAR(200)	YES	Order notes

Sales Invoices Schema

```
    CREATE TABLE public.sales_invoices (
  invoice_id    BIGSERIAL PRIMARY KEY,
  order_id      BIGINT NOT NULL REFERENCES public.sales_orders(
     order_id) ON DELETE RESTRICT,
  customer_id   BIGINT NOT NULL REFERENCES public.customers(
     customer_id) ON DELETE RESTRICT,
  invoice_date  TIMESTAMPTZ NOT NULL DEFAULT now(),
  document_no   VARCHAR(60) NOT NULL UNIQUE,
  description   VARCHAR(200),
  total_net     NUMERIC(14,2) NOT NULL DEFAULT 0,
  total_vat     NUMERIC(14,2) NOT NULL DEFAULT 0,
  total_gross   NUMERIC(14,2) NOT NULL DEFAULT 0
);

CREATE INDEX idx_invoices_date ON public.sales_invoices(
   invoice_date DESC);
```

Column Name	Data Type	Allow Nulls	Description
invoice_id	BIGSERIAL	NO	Primary key
order_id	BIGINT	NO	References sales_orders(order_id) ON DELETE RESTRICT
customer_id	BIGINT	NO	References customers(customer_id) ON DELETE RESTRICT
invoice_date	TIMESTAMPTZ	NO	Invoice date (default: now())
document_no	VARCHAR(60)	NO	Invoice document number (unique)
description	VARCHAR(200)	YES	Invoice description
total_net	NUMERIC(14,2)	NO	Total net amount (default: 0, computed by trigger)
total_vat	NUMERIC(14,2)	NO	Total VAT amount (default: 0, computed by trigger)
total_gross	NUMERIC(14,2)	NO	Total gross amount (default: 0, computed by trigger)

Computed Fields: The totals are automatically computed by trigger `trg_invoice_header_after_` when invoice items are inserted, updated, or deleted.

Invoice Items Schema

```
1
2  CREATE TABLE public.invoice_items (
3    invoice_id    BIGINT NOT NULL REFERENCES public.sales_invoices(
       invoice_id) ON DELETE CASCADE,
4    line_no       INT NOT NULL,
5    product_id    BIGINT NOT NULL REFERENCES public.products(
       product_id) ON DELETE RESTRICT,
6    stock_code    VARCHAR(50) NOT NULL,
7    quantity_kg   NUMERIC(12,3) NOT NULL CHECK (quantity_kg > 0),
8    unit_price    NUMERIC(12,2) NOT NULL CHECK (unit_price >= 0),
9    vat_rate      NUMERIC(5,2) NOT NULL DEFAULT 1 CHECK (vat_rate >=
       0),
10   exit_temperature_c NUMERIC(6,2),
11   exit_time     TIMESTAMPTZ,
12   line_net      NUMERIC(14,2) NOT NULL DEFAULT 0,
13   line_vat      NUMERIC(14,2) NOT NULL DEFAULT 0,
14   line_gross    NUMERIC(14,2) NOT NULL DEFAULT 0,
15   PRIMARY KEY (invoice_id, line_no)
16 );
17
18 CREATE INDEX idx_items_invoice ON public.invoice_items(invoice_id
     );
19 CREATE INDEX idx_items_product ON public.invoice_items(product_id
     );
20
21 CREATE OR REPLACE FUNCTION public.fn_compute_invoice_item_totals
     ()
22 RETURNS trigger LANGUAGE plpgsql AS $$
23 BEGIN
24   NEW.line_net   := ROUND((NEW.quantity_kg * NEW.unit_price)::
       numeric, 2);
25   NEW.line_vat   := ROUND((NEW.line_net * (NEW.vat_rate / 100.0))
       ::numeric, 2);
26   NEW.line_gross := NEW.line_net + NEW.line_vat;
27   IF NEW.exit_time IS NULL THEN
28     NEW.exit_time := now();
29   END IF;
30   RETURN NEW;
31 END $$;
32
33 CREATE TRIGGER trg_invoice_item_totals
34 BEFORE INSERT OR UPDATE ON public.invoice_items
35 FOR EACH ROW EXECUTE FUNCTION public.
     fn_compute_invoice_item_totals();
36
37 CREATE OR REPLACE FUNCTION public.fn_recompute_invoice_header()
38 RETURNS trigger LANGUAGE plpgsql AS $$
39 DECLARE v_invoice_id BIGINT;
40 BEGIN
```

```
  v_invoice_id := COALESCE(NEW.invoice_id, OLD.invoice_id);

  UPDATE public.sales_invoices i SET
    total_net = COALESCE((SELECT SUM(line_net) FROM public.
       invoice_items WHERE invoice_id = v_invoice_id), 0),
    total_vat = COALESCE((SELECT SUM(line_vat) FROM public.
       invoice_items WHERE invoice_id = v_invoice_id), 0),
    total_gross = COALESCE((SELECT SUM(line_gross) FROM public.
       invoice_items WHERE invoice_id = v_invoice_id), 0)
  WHERE i.invoice_id = v_invoice_id;

  RETURN NULL;
END $$;

CREATE TRIGGER trg_invoice_header_after_items
AFTER INSERT OR UPDATE OR DELETE ON public.invoice_items
FOR EACH ROW EXECUTE FUNCTION public.fn_recompute_invoice_header
   ();

CREATE TABLE public.drivers (
  party_id    BIGINT PRIMARY KEY REFERENCES public.persons(
     party_id) ON DELETE CASCADE,
  driver_tckn VARCHAR(20) NOT NULL UNIQUE
);
```

invoice_items is a **weak entity** identified by (invoice_id, line_no).

Column Name	Data Type	Allow Nulls	Description
invoice_id	BIGINT	NO	Part of primary key, references sales_invoices(invoice_id)
line_no	INT	NO	Part of primary key, line number
product_id	BIGINT	NO	References products(product_id)
stock_code	VARCHAR(50)	NO	Stock code (denormalized for performance)
quantity_kg	NUMERIC(12,3)	NO	Quantity in kilograms (CHECK: > 0)
unit_price	NUMERIC(12,2)	NO	Unit price (CHECK: ≥ 0)
vat_rate	NUMERIC(5,2)	NO	VAT rate (default: 1, CHECK: ≥ 0)
exit_temperature_c	NUMERIC(6,2)	YES	Exit temperature
exit_time	TIMESTAMPTZ	YES	Exit timestamp (set by trigger if NULL)
line_net	NUMERIC(14,2)	NO	Line net amount (computed by trigger)
line_vat	NUMERIC(14,2)	NO	Line VAT amount (computed by trigger)

line_gross	NUMERIC(14,2)	NO	Line gross amount (computed by trigger)

Computed Fields: Line totals are automatically computed by trigger `trg_invoice_item_totals` before insert or update.

Shipments Schema

```
CREATE TABLE public.shipments (
  shipment_id     BIGSERIAL PRIMARY KEY,
  waybill_type    VARCHAR(60) NOT NULL DEFAULT 'Sevk  rsaliyesi  ',
  waybill_no      VARCHAR(60) NOT NULL UNIQUE,
  shipment_date   DATE NOT NULL,
  shipment_time   TIME NOT NULL,
  departure_time  TIMESTAMPTZ NOT NULL DEFAULT now(),
  carrier_id      BIGINT NOT NULL REFERENCES public.carriers(carrier_id) ON DELETE RESTRICT,
  vehicle_id      BIGINT NOT NULL REFERENCES public.vehicles(vehicle_id) ON DELETE RESTRICT
);

CREATE TABLE public.shipment_invoices (
  shipment_id BIGINT NOT NULL REFERENCES public.shipments(shipment_id) ON DELETE CASCADE,
  invoice_id  BIGINT NOT NULL REFERENCES public.sales_invoices(invoice_id) ON DELETE RESTRICT,
  PRIMARY KEY (shipment_id, invoice_id)
);

CREATE TABLE public.shipment_drivers (
  shipment_id    BIGINT NOT NULL REFERENCES public.shipments(shipment_id) ON DELETE CASCADE,
  driver_party_id BIGINT NOT NULL REFERENCES public.drivers(party_id) ON DELETE RESTRICT,
  driver_role    VARCHAR(40) NOT NULL DEFAULT 'DRIVER',
  PRIMARY KEY (shipment_id, driver_party_id)
);

CREATE OR REPLACE FUNCTION public.fn_shipment_must_have_driver()
RETURNS trigger LANGUAGE plpgsql AS $$
DECLARE v_sid BIGINT;
DECLARE v_cnt INT;
BEGIN
  v_sid := COALESCE(NEW.shipment_id, OLD.shipment_id);
  SELECT COUNT(*) INTO v_cnt FROM public.shipment_drivers WHERE shipment_id = v_sid;
  IF v_cnt = 0 THEN
    RAISE EXCEPTION 'Shipment % must have at least one driver.', v_sid;
  END IF;
```

```
  RETURN NULL;
END $$;

CREATE CONSTRAINT TRIGGER shipment_drivers_has_driver
AFTER INSERT OR UPDATE OR DELETE ON public.shipment_drivers
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW EXECUTE FUNCTION public.fn_shipment_must_have_
```

Column Name	Data Type	Allow Nulls	Description
shipment_id	BIGSERIAL	NO	Primary key
waybill_type	VARCHAR(60)	NO	Waybill type (default: 'Sevk İrsaliyesi')
waybill_no	VARCHAR(60)	NO	Waybill number (unique)
shipment_date	DATE	NO	Shipment date
shipment_time	TIME	NO	Shipment time
departure_time	TIMESTAMPTZ	NO	Departure timestamp (default: now())
carrier_id	BIGINT	NO	References carriers(carrier_id) ON DELETE RESTRICT
vehicle_id	BIGINT	NO	References vehicles(vehicle_id) ON DELETE RESTRICT

Related Tables:

- `shipment_invoices`: M:N relationship (one shipment can carry multiple invoices)
- `shipment_drivers`: M:N relationship (co-drivers allowed)
- `shipment_fulfillments`: Ternary relationship linking shipments, invoice items, and batches

Business Rules:

- Each shipment must have at least one driver (enforced by deferred constraint trigger)
- An invoice can be linked to at most one vehicle across all shipments (enforced by triggers)

Finance Records Schema

```
CREATE TABLE public.finance_records (
  record_id    BIGSERIAL PRIMARY KEY,
  record_time  TIMESTAMPTZ NOT NULL DEFAULT now(),
  record_type  public.expense_type NOT NULL,
  category     VARCHAR(120) NOT NULL,
  description  VARCHAR(200),
  amount       NUMERIC(14,2) NOT NULL,
  status       public.finance_posting_status NOT NULL DEFAULT 'DRAFT',
  posted_at    TIMESTAMPTZ,
  posted_by_employee_id BIGINT REFERENCES public.employees(employee_id) ON DELETE SET NULL
);

CREATE INDEX idx_finance_time ON public.finance_records(record_time DESC);
CREATE INDEX idx_finance_type ON public.finance_records(record_type);

CREATE TABLE public.finance_sources (
  record_id          BIGINT PRIMARY KEY REFERENCES public.finance_records(record_id) ON DELETE CASCADE,
  source_kind        public.finance_source_kind NOT NULL,
  invoice_id         BIGINT REFERENCES public.sales_invoices(invoice_id) ON DELETE RESTRICT,
  machine_id         BIGINT REFERENCES public.machines(machine_id) ON DELETE RESTRICT,
  payroll_ref        VARCHAR,
  rent_contract_ref  VARCHAR,
  other_ref          VARCHAR,
  CONSTRAINT ck_fin_source_one CHECK (
    (invoice_id IS NOT NULL)::int +
    (machine_id IS NOT NULL)::int +
    (payroll_ref IS NOT NULL)::int +
    (rent_contract_ref IS NOT NULL)::int +
    (other_ref IS NOT NULL)::int
    = 1
  )
);
```

Column Name	Data Type	Allow Nulls	Description
record_id	BIGSERIAL	NO	Primary key
record_time	TIMESTAMPTZ	NO	Record timestamp (default: now())
record_type	expense_type	NO	Expense type ENUM
category	VARCHAR(120)	NO	Category name
description	VARCHAR(200)	YES	Description

amount	NUMERIC(14,2)	NO	Amount
status	finance_posting_status	NO	Posting status (default: 'DRAFT')
posted_at	TIMESTAMPTZ	YES	Posting timestamp
posted_by_employee_id	BIGINT	YES	References employees(employee_id) ON DELETE SET NULL

Related Tables:

- `finance_sources`: Links finance records to their source (exactly one source must be present)

Business Rules:

- POSTED records are immutable (cannot be updated or deleted, enforced by triggers)
- Each finance record must have exactly one source (enforced by CHECK constraint)

Populated Tables

This section provides sample data for key tables to demonstrate the database structure. The data is realistic and follows all constraints and business rules.

Parties Table

party_id	party_kind	created_at
1	PERSON	2024-01-15 10:30:00+03
2	ORGANIZATION	2024-01-15 11:00:00+03
3	PERSON	2024-01-16 09:15:00+03
4	ORGANIZATION	2024-01-16 14:20:00+03
5	PERSON	2024-01-17 08:45:00+03

Persons Table

party_id	tckn	full_name	email
1	12345678901	Ahmet Yılmaz	ahmet.yilmaz@email.com
3	98765432109	Fatma Demir	fatma.demir@email.com
5	11223344556	Mehmet Kaya	mehmet.kaya@email.com

Organizations Table

party_id	legal_name	vkn	tax_office
2	ABC Gıda A.Ş.	1234567890	Çankaya VD
4	XYZ Süt Ürünleri Ltd.	9876543210	Keçiören VD

Employees Table

employee_id	full_name	role	department	salary_monthly
1	Ali Öztürk	Production Manager	Production	25000.00
2	Ayşe Çelik	Quality Control	Quality	18000.00
3	Mustafa Şahin	Warehouse Manager	Warehouse	22000.00
4	Zeynep Arslan	Sales Manager	Sales	24000.00

Suppliers Table

supplier_id	supplier_name	party_id	city
1	Çiftlik Süt Tedarik	26	Ankara
2	Doğal Süt Üreticileri	14	İstanbul

Customers Table

customer_id	company_name	party_id	city
1	Market Zinciri A.Ş.	58	Ankara
2	Restoran Grubu Ltd.	44	İstanbul

Products Table

product_id	product_name	category	stock_code	vat_rate
1	Tam Yağlı Süt	Dairy	ST-MLK-001	1.00
2	Beyaz Peynir	Cheese	ST-CHZ-001	1.00
3	Tereyağı	Butter	ST-BTR-001	1.00
4	Yoğurt	Dairy	ST-YGT-001	1.00

Milk Intakes Table

intake_id	intake_time	milk_kind	quantity_liters	total_cost	density_kg_per_l
1	2024-01-20 08:00:00+03	RAW	5000.00	25000.00	1.032
2	2024-01-21 09:30:00+03	RAW	3000.00	15000.00	1.030
3	2024-01-22 07:45:00+03	PASTEURIZED	2000.00	12000.00	1.028

Production Orders Table

production_id	created_time	planned_date	status	created_by_employee_id
1	2024-01-20 10:00:00+03	2024-01-21	PLANNED	1
2	2024-01-21 11:00:00+03	2024-01-22	IN_PROGRESS	1
3	2024-01-22 09:00:00+03	2024-01-23	COMPLETED	1

Sales Orders Table

order_id	customer_id	order_date	note
1	1	2024-01-25	Regular monthly order
2	2	2024-01-26	Urgent delivery required

Sales Invoices Table

invoice_id	order_id	customer_id	document_no	total_gross
1	1	1	INV-2024-001	15000.00
2	2	2	INV-2024-002	8500.00

User Interface

Our database handles a broad range of applications going on at a dairy company. As such, it is designed to be used by users from many different aspects of the company. It is unreasonable to assume everyone who has to use this database knows PostgreSQL, nor is it feasible to teach them. This raised the use for an easy-to-use user interface (UI). Since we have no experience in designing or coding UI, we used ChatGPT to create one in Python.

The main menus of our UI are: s

- Dashboard
- Employees
- Products & Prices
- Suppliers / Customers / Drivers
- Milk Intakes
- Production
- Warehouses & Locations

- Inventory
- Sales (Orders / Invoices / Shipments)
- Machines & Maintenance
- Finance
- SQL Console (to access data with precise specifications by someone who knows SQL syntax)

We created a dashboard on the entry screen of our program. At a quick glance, the user can see the amount of total raw milk (in liters) the company currently has, total income and outcome (in Turkish Lira), and the number of malfunctioning machinery across the facilities of the company. We figured that these items are the most crucial in keeping a dairy company running. Furthermore, one can see the composition of the company's inventory and a time-series graph of the company's finance.

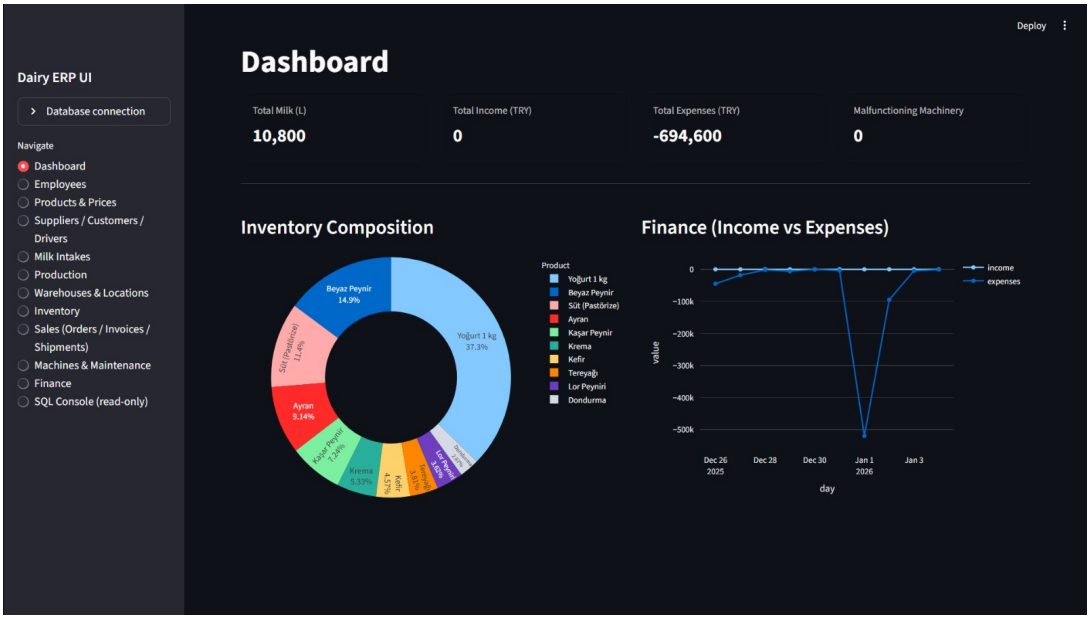


Figure 1: UI Dashboard

The main usage of this UI is to allow users to easily view, add, update and remove data rows without engaging in PostgreSQL coding. The "Employees" tab possesses each of these utilities, so we thought it would be the best candidate to show examples of the utilities of our UI.

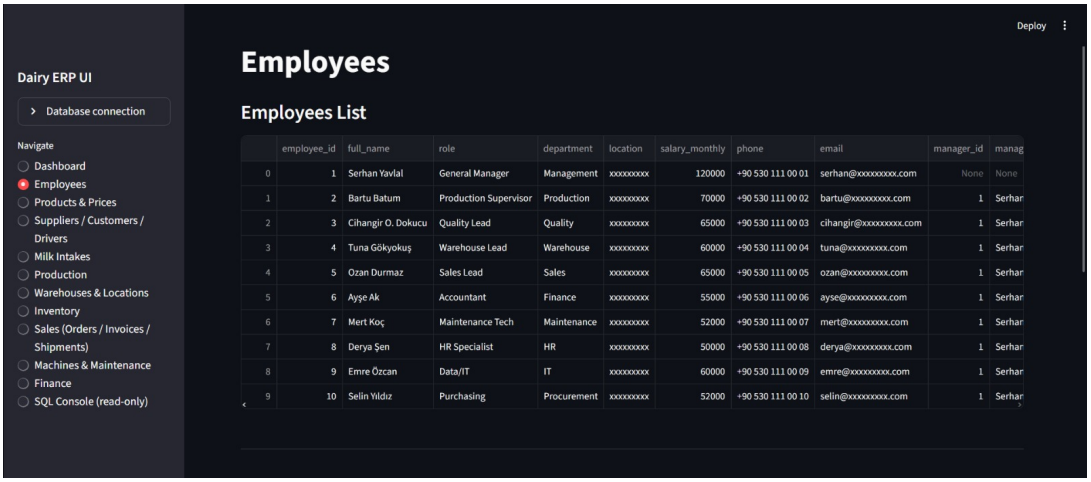


Figure 2: Viewing Data

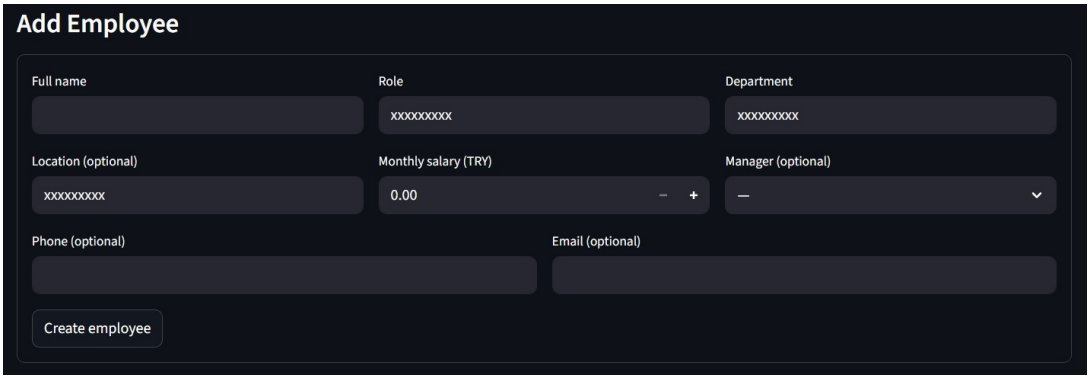


Figure 3: Adding New Data

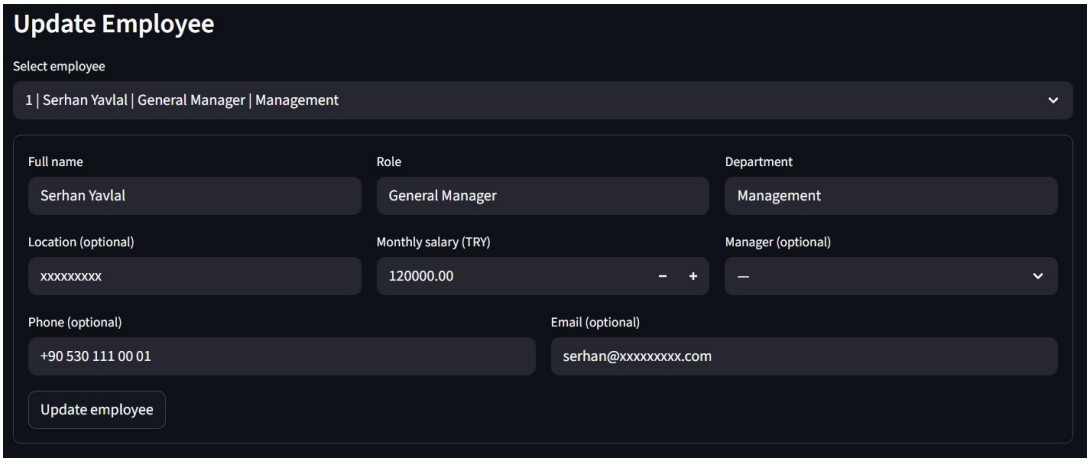


Figure 4: Updating Existing Data

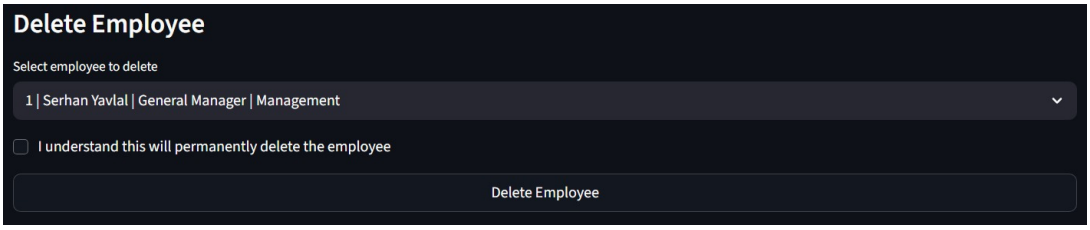


Figure 5: Deleting Data

Enforcing Constraints

Our database implements triggers to enforce constraints and keep our data sensible and accurate.

1. **Deferred Constraint Triggers**: Used for complex rules that require checking across multiple tables:
   - `trg_intake_has_supplier`: Ensures every milk intake has at least one supplier
   - `trg_order_has_step`: Ensures every production order has at least one step
   - `trg_shipment_has_driver`: Ensures every shipment has at least one driver
2. **Immediate Triggers**: Used for data validation and computation:
   - `trg_invoice_item_totals`: Computes line totals before inserting a new invoice or updating an existing invoice
   - `trg_invoice_header_after_items`: Computes invoice header totals again after item changes
   - `trg_bli_current_qty_check`: Ensures location quantities are not greater than batch quantity
   - `trg_no_negative_stock_fulfillment`: Prevents negative stock in fulfillments
   - `trg_finance_immutable_u/d`: Prevents modification of finance records of POSTED status
3. **Exclusion Constraints**: Prevent overlapping intervals using GiST indexes:
   - `ex_vca_no_overlap`: Vehicle-carrier assignments
   - `ex_vtch_no_overlap`: Vehicle temperature class history
   - `ex_bli_no_overlap`: Batch location intervals
   - `ex_pp_no_overlap`: Product price validity periods
   - `ex_cp_no_overlap`: Contract price validity periods

Cascade on Delete and Update and Referential Integrity

Our database includes a lot of rows that interact with or depend on other rows. So, if a row is deleted, these interactions and/or dependencies must be handled accordingly. So, we implemented appropriate CASCADE and RESTRICT behaviours to keep these interactions under control.

Table	Referenced Table	Action on Delete/Update
persons	parties	ON DELETE CASCADE
organizations	parties	ON DELETE CASCADE
suppliers	parties	ON DELETE RESTRICT
customers	parties	ON DELETE RESTRICT
employees	employees (manager)	ON DELETE SET NULL
vehicle_carrier_assignments	vehicles	ON DELETE CASCADE
vehicle_carrier_assignments	carriers	ON DELETE RESTRICT
milk_intake_suppliers	milk_intakes	ON DELETE CASCADE
milk_intake_suppliers	suppliers	ON DELETE RESTRICT
production_order_products	production_orders	ON DELETE CASCADE
production_order_products	products	ON DELETE RESTRICT
invoice_items	sales_invoices	ON DELETE CASCADE
invoice_items	products	ON DELETE RESTRICT
shipment_invoices	shipments	ON DELETE CASCADE
shipment_invoices	sales_invoices	ON DELETE RESTRICT
shipment_fulfillments	shipments	ON DELETE CASCADE
shipment_fulfillments	invoice_items	ON DELETE RESTRICT
shipment_fulfillments	inventory_batches	ON DELETE RESTRICT

Key Principles:

- **CASCADE**: Used when child records are meaningless without parent (for example, invoice items without invoice records)
- **RESTRICT**: Used when parent deletion would violate business rules (for example, cannot delete supplier that has an active intake currently)
- **SET NULL**: Used for optional relationships.

Conclusion

Our database implements a design that tries to meet the complicated demands of a dairy company. As such, the system has:

- **Data Integrity**: Constraints are enforced tightly by the usage of CHECK constraints, foreign keys, and triggers.
- **Temporal Data Management**: Support for historical data through interval-based tables with non-overlapping constraints, enforced with GiST indexing.
- **Business Rule Enforcement**: Deferred and immediate triggers ensure complex business rules are maintained
- **Performance Optimization**
- **Flexible Design**: Database supports co-production, multi-role parties and batch splitting so that different functions can run concurrently without causing each other trouble.

The database is production-ready and can serve as the foundation for a full ERP application with appropriate application-layer interfaces. The design principles demonstrated here can be extended to other enterprise resource planning domains.

Participation

Throughout the entire project, everyone was involved in every step of building this database in one way or another. While everyone had shared ideas and insights on the project's conception, design, coding the database in PostgreSQL and coding the UI in Python, there were fields in which everyone excelled:

- **Bartu Batum**: Conceptualized the idea, handled overall system design and communication with the dairy company
- **Ozan Durmaz and Tuna Gökyokuş**: Focused on heavy coding in PostgreSQL and handled constraint implementation
- **Serhan Yavlal and Cihangir Osman Dokucu**: Handled the visualization aspects of the project, including the ER diagrams and the UI design