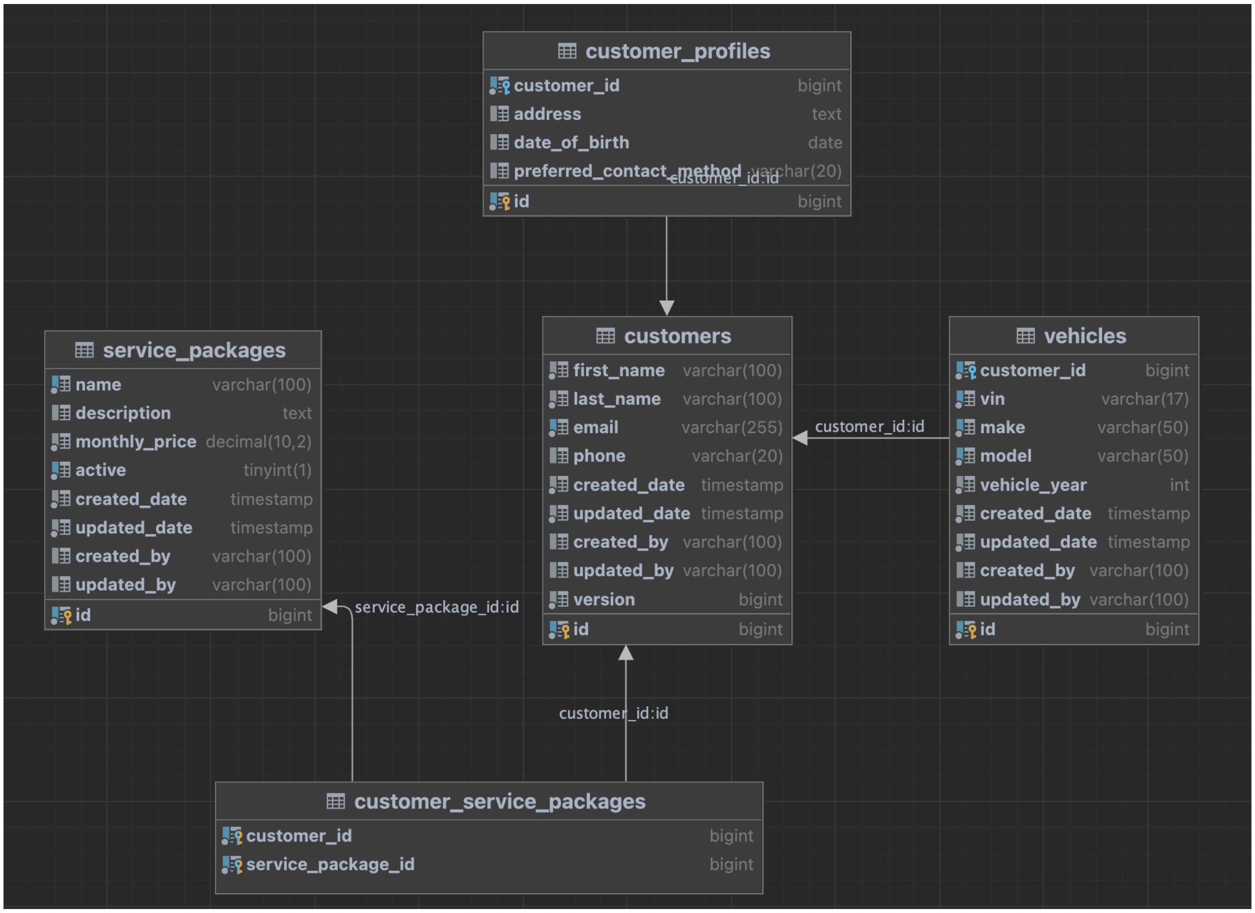
Database Schema Documentation

Customer Management API - Tekmetric Interview

Entity Relationship Diagram



customer_profiles	
customer_id	bigint
address	text
date_of_birth	date
preferred_contact_method	varchar(20)
id	bigint

customers	
first_name	varchar(100)
last_name	varchar(100)
email	varchar(255)
phone	varchar(20)
created_date	timestamp
updated_date	timestamp
created_by	varchar(100)
updated_by	varchar(100)
version	bigint
id	bigint

service_packages	
name	varchar(100)
description	text
monthly_price	decimal(10,2)
active	tinyint(1)
created_date	timestamp
updated_date	timestamp
created_by	varchar(100)
updated_by	varchar(100)
id	bigint

vehicles	
customer_id	bigint
vin	varchar(17)
make	varchar(50)
model	varchar(50)
vehicle_year	int
created_date	timestamp
updated_date	timestamp
created_by	varchar(100)
updated_by	varchar(100)
id	bigint

customer_service_packages	
customer_id	bigint
service_package_id	bigint

Entity Relationship Diagram showing all tables, columns, data types, and foreign key relationships

Table Specifications

CUSTOMERS

Primary Entity: Core customer information with optimistic locking

Column	Type	Constraints	Description
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier
version	BIGINT	NOT NULL, DEFAULT 0	Optimistic locking version
first_name	VARCHAR(100)	NOT NULL	Customer first name
last_name	VARCHAR(100)	NOT NULL	Customer last name
email	VARCHAR(255)	NOT NULL, UNIQUE	Contact email
phone	VARCHAR(20)	NULL	Contact phone number
created_date	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Creation timestamp
updated_date	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Last update timestamp
created_by	VARCHAR(100)	NULL	User who created record
updated_by	VARCHAR(100)	NULL	User who last updated record

Indexes:

- `idx_customers_email` ON (email) - For email lookups and uniqueness
- PRIMARY KEY ON (id)

Relationships:

- 1:1 with CUSTOMER_PROFILES
- 1:N with VEHICLES
- N:M with SERVICE_PACKAGES via junction table

CUSTOMER_PROFILES

Extension Entity: Optional detailed customer information

Column	Type	Constraints	Description
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier
customer_id	BIGINT	NOT NULL, UNIQUE, FK	Reference to customers table
address	TEXT	NULL	Customer address
date_of_birth	DATE	NULL	Birth date
preferred_contact_method	VARCHAR(20)	DEFAULT 'EMAIL'	Contact preference (EMAIL/PHONE/SMS)

Indexes:

- `idx_customer_profiles_customer_id` ON (customer_id)
- PRIMARY KEY ON (id)

Foreign Keys:

- customer_id REFERENCES customers(id) ON DELETE CASCADE

VEHICLES

Related Entity: Customer vehicle inventory

Column	Type	Constraints	Description
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier
customer_id	BIGINT	NOT NULL, FK	Owner reference
vin	VARCHAR(17)	NOT NULL, UNIQUE	Vehicle identification number
make	VARCHAR(50)	NOT NULL	Vehicle manufacturer

model	VARCHAR(50)	NOT NULL	Vehicle model
vehicle_year	INTEGER	NOT NULL	Manufacturing year
created_date	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Creation timestamp
updated_date	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Last update timestamp
created_by	VARCHAR(100)	NULL	User who created record
updated_by	VARCHAR(100)	NULL	User who last updated record

Indexes:

- `idx_vehicles_customer_id` ON (customer_id) - For customer vehicle queries
- `idx_vehicles_vin` ON (vin) - For VIN lookups and uniqueness
- `idx_vehicles_make_model` ON (make, model) - For search functionality
- PRIMARY KEY ON (id)

Foreign Keys:

- customer_id REFERENCES customers(id) ON DELETE CASCADE

SERVICE_PACKAGES

Business Entity: Available service offerings with soft delete

Column	Type	Constraints	Description
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier
name	VARCHAR(100)	NOT NULL, UNIQUE	Package name
description	TEXT	NULL	Package description
monthly_price	DECIMAL(10,2)	NOT NULL	Monthly subscription price

active	BOOLEAN	NOT NULL, DEFAULT TRUE	Soft delete flag
created_date	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Creation timestamp
updated_date	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Last update timestamp
created_by	VARCHAR(100)	NULL	User who created record
updated_by	VARCHAR(100)	NULL	User who last updated record

Indexes:

- `idx_service_packages_name` ON (name) - For name lookups and uniqueness
- `idx_service_packages_active` ON (active) - For active package filtering
- PRIMARY KEY ON (id)

CUSTOMER_SERVICE_PACKAGES

Junction Entity: Many-to-many relationship between customers and service packages

Column	Type	Constraints	Description
customer_id	BIGINT	NOT NULL, FK	Customer reference
service_package_id	BIGINT	NOT NULL, FK	Service package reference

Indexes:

- `idx_customer_service_packages_customer_id` ON (customer_id)
- `idx_customer_service_packages_service_package_id` ON (service_package_id)
- PRIMARY KEY ON (customer_id, service_package_id)

Foreign Keys:

- customer_id REFERENCES customers(id) ON DELETE CASCADE
- service_package_id REFERENCES service_packages(id) ON DELETE CASCADE

Entity Relationship Cascade Behavior

JPA-Level Cascade Operations

Customer → CustomerProfile (1:1)

```
@OneToOne(cascade = {CascadeType.PERSIST, CascadeType.MERGE}, orphanRemoval = true)
```

- **JPA Behavior**: `orphanRemoval = true` deletes CustomerProfile when Customer is deleted OR when `customer.setCustomerProfile(null)`
- **No CASCADE.REMOVE needed**: `orphanRemoval` handles deletion automatically
- **Reason**: Profile has no meaning without the customer

Customer → Vehicles (1:N)

```
@OneToMany(mappedBy = "customer", fetch = FetchType.Lazy, orphanRemoval = true)
```

- **JPA Behavior**: `orphanRemoval = true` deletes Vehicles when Customer is deleted OR when vehicles are removed from the collection
- **No CASCADE.REMOVE needed**: `orphanRemoval` handles deletion automatically
- **Reason**: In this domain model, vehicles belong exclusively to one customer

Customer ↔ ServicePackages (M:N)

```
@ManyToMany(fetch = FetchType.Lazy)
@JoinTable(name = "customer_service_packages", ...)
```

- **JPA Behavior**: No cascade operations - entities are independent
- **Relationship Management**: Only junction table records are affected
- **Reason**: Customers and ServicePackages exist independently

Database-Level Cascade Operations

Vehicles Table

```
FOREIGN KEY (customer_id) REFERENCES customers (id) ON DELETE CASCADE
```

- **Database Safety Net**: If Customer is deleted at DB level, Vehicles are also deleted
- **Primary Mechanism**: JPA `orphanRemoval` handles this in application code

Junction Table (customer_service_packages)

```
FOREIGN KEY (customer_id) REFERENCES customers (id) ON DELETE CASCADE
FOREIGN KEY (service_package_id) REFERENCES service_packages (id) ON DELETE CASCADE
```

- **Database Behavior**: Junction table records are deleted when either Customer or ServicePackage is deleted
- **Reason**: Maintains referential integrity at database level

Soft Delete vs Hard Delete

ServicePackages - Soft Delete

- Uses `active` boolean field instead of physical deletion
- **Reason**: Preserve historical subscription data

Other Entities - Hard Delete

- Physical removal from database using JPA `orphanRemoval` and repository delete methods
- **Reason**: Clean data model for demo purposes

Database Features

Audit Trail

All main entities include comprehensive audit fields:

- **created_date/updated_date**: Automatic timestamp management
- **created_by/updated_by**: User tracking via Spring Security context
- Implemented using JPA `@EntityListeners(AuditingEntityListener.class)`

Optimistic Locking

- **customers** table includes `version` field
- Prevents concurrent update conflicts
- Implemented via JPA `@Version` annotation
- Returns HTTP 409 Conflict on version mismatch

Data Integrity

- **Referential Integrity**: Proper foreign key constraints with CASCADE rules
- **Business Rules**: CHECK constraints and application-level validation
- **Unique Constraints**: Email uniqueness, VIN uniqueness, package name uniqueness

Performance Optimization

- **Strategic Indexes**: Covering primary query patterns
- **Entity Graphs**: Prevention of N+1 query problems
- **Connection Pooling**: HikariCP with production tuning

Migration Strategy

- **Flyway**: Version-controlled schema changes
- **Backward Compatibility**: Additive changes only
- **Sample Data**: Profile-specific data loading