

|  |
| --- |
| **NYC taxi trip** |
|  |



Contents

[1 Business Description 3](#_Toc155614186)

[1.1 Business background 3](#_Toc155614187)

[1.2 Problems because of poor data management 3](#_Toc155614188)

[1.3 Benefits from implementing a Data Warehouse 3](#_Toc155614189)

[1.4 DATASETS DESCRIPTION 3](#_Toc155614190)

[1.5 GRAIN / DIM / FACT 4](#_Toc155614191)

[2 Business Layer 3NF 4](#_Toc155614192)

[2.1 Definitions & Acronyms 3](#_Toc62212635)

[2.2 Logical Scheme 3](#_Toc62212636)

[2.3 Objects 3](#_Toc62212637)

[3 Business Layer Dimensional Model 4](#_Toc155614193)

[2.1 Logical Scheme 3](#_Toc62212635)

[2.2 Metrics description 3](#_Toc62212636)

[4 Logical Scheme 4](#_Toc155614194)

[5 Data Flow 4](#_Toc155614195)

[6 Fact Table Partitioning Strategy 4](#_Toc155614196)

# 

# Business Description

## Business background

Taxis are a vital part of the transportation infrastructure in any large city, including New York. In this industry, there are many operators offering taxi services for various types of trips, from short city rides to long trips to airports. Competition in this sector is high, so successful companies must effectively manage their data to improve service and meet customer needs.

## Problems because of poor data management

Poor data management hampers effective business operations due to insufficient information for decision-making. Without using data analysis tools, companies cannot develop a strategy focused on customer needs and operational optimization. This can lead to decreased market competitiveness and lower quality of services provided.

## Benefits from implementing a Data Warehouse

Implementing a data warehouse can help solve the described problems and answer the following questions:

* What times of day are most in demand for taxi rides?
* Which areas of the city (by longitude and latitude) are most frequently served?
* How does demand for taxis change depending on the day of the week or month?
* Which types of trips (business, private) are more popular?

## DATASETS DESCRIPTION

In the dataset, Yellow Taxi is only associated with the Street method, so it will not have a booking\_type column but can be inferred by the method of booking. Booking type hierarchical structure can be useful for understanding the relationship between taxi types and booking methods.

Booking Type:

This represents the type of taxi service.

* Values: Green Taxi (Yellow Taxi is only available through the Street method).
* Description: The category of taxi associated with the booking. In this case, Yellow Taxi is always associated with street bookings, while Green Taxi can have multiple booking methods.

Booking Method (Subcategory):

This indicates the method or channel used to book the taxi.

* Values: Street, Phone.
* Description: The specific booking method used for each taxi type. For instance, booking via the street or through a phone call.

Example Hierarchy:

Booking Type = Yellow Taxi

* Street — Booking made through the street (Yellow Taxi can only be booked this way).

Booking Type = Green Taxi

* Street — Booking made through the street.
* Phone — Booking made through a phone call.

Datasets:

**The First Dataset: Yellow Taxi Data** **(SRC\_GREEN\_TAXI\_DATA.csv)**

Information about Yellow Taxi trips:

1. **Vendor Information**:

vendor\_src\_id: unique identifier

vendor\_name: Vendor name

street: Vendor street, part of the vendor’s address

house: Vendor house, part of the vendor’s address

city: Vendor city, part of the vendor’s address

house: Vendor house, part of the vendor’s address

postal\_code: Vendor postal code, part of the vendor’s address

vendor\_telephone: Vendor telephone

1. **Payment Information**:

payment\_src\_id: unique identifier

payment\_type: card or cash

payment\_time: payment time

1. **Rate Information**:

rate\_src\_id: unique identifier

base\_fare: base fare

rate\_per\_mile: rate per mile

1. **Trip Information**:

trip\_src\_id: unique identifier

pickup\_longitude: pickup longitude

pickup\_latitude: pickup latitude

dropoff\_longitude: dropoff longitude

dropoff\_latitude: dropoff latitude

pickup\_datetime: Pickup date and time

dropoff\_datetime: Dropoff date and time

passenger\_count: Number of passengers

distance\_miles: distance\_miles

trip\_duration: trip duration

**The Second Dataset: Green Taxi Data (SRC\_YELLOW\_TAXI\_DATA.csv)**

Information about Green Taxi trips:

1. **Vendor Information**:

vendor\_src\_id: unique identifier

vendor: Vendor name

street: Vendor street, part of the vendor’s address

house: Vendor house, part of the vendor’s address

city: Vendor city, part of the vendor’s address

house: Vendor house, part of the vendor’s address

postal\_code: Vendor postal code, part of the vendor’s address

vendor\_telephone: Vendor telephone

1. **Customer Information**:

customer\_src\_id: unique identifier

customer\_type: individual, or business

customer telephone: telephone number used to place the ordrer

1. **Booking Information**:

booking\_src\_id: unique identifier

booking\_type: phone, or street

booking\_datetime: booking time

1. **Payment Information**:

payment\_src\_id: unique identifier

payment\_type: card or cash

1. **Rate Information**:

rate\_src\_id: unique identifier

base\_fare: base fare

rate\_per\_mile: rate per mile

1. **Promotion Information**:

promo\_src\_id: unique identifier

promo code: promo code

discount\_percentage: 10%, 20%, none

1. **Trip Information**:

trip\_src\_id: unique identifier

pickup\_datetime: Pickup date and time

dropoff\_datetime: Dropoff date and time

passenger\_count: Number of passengers

trip\_duration: Duration of the trip

pickup\_longitude: Pickup longitude

pickup\_latitude: Pickup latitude

dropoff\_longitude: Dropoff longitude

dropoff\_latitude: Dropoff latitude

## **GRAIN / DIM / FACT**

#### Grain

Each record represents one taxi trip. This will be the finest level of detail. We will be analyzing this in this assignment.

#### Dimensions

The granularity of the datasets is defined at the level of individual trips. Each row in the fact table represents one taxi trip: vendor\_id, customer\_id, booking\_id, payment\_id, transmission\_id, rate\_id, promo\_id, trip\_id

* **DIM\_VENDORS**

The taxi company transports people from point A to point B

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| vendor\_src\_id | unique identifier for vendor | bigint |
| vendor\_name | vendor name | varchar |
| vendor\_street | part of full address | varchar |
| vendor\_house | part of full address | varchar |
| vendor\_city | part of full address | varchar |
| vendor\_country | part of full address | varchar |
| vendor\_postal code | address postal code | varchar |
| vendor\_telephone | telephone number | varchar |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| vendor\_id | vendor\_name | street | house | city | country | postal code | vendor\_telephone |
| 3 | Green Taxi | Beaver Street | 33 | New York | USA | NY 10004 | +1 212-639-9675 |

Example with filled data

* **DIM\_CUSTOMERS**

A Customer is a client who can be either an individual or a business.

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| customer\_src\_id | unique identifier for customer | bigint |
| customer\_type | Business/Individual | varchar |
| customer\_telephone | telephone number | varchar |

|  |  |  |
| --- | --- | --- |
| customer\_id | customer\_type | customer\_telephone |
| 101 | Business | +1 212-639-9675 |
| 1 | Individual | NULL |

Example with filled data

* **DIM\_BOOKINGS**

Green Taxi can be ordered from the street and via phone. Yellow Taxi can only be ordered from the street, therefore, the call is only made on the street.

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| booking\_src\_id | unique identifier for booking | bigint |
| booking\_type | phone/street | varchar |
| booking\_datetime | booking time | datetime |

|  |  |  |
| --- | --- | --- |
| booking\_id | booking\_type | booking\_datetime |
| 52 | Phone | 3/26/2016 13:18 |

Example with filled data

* **DIM\_PAYMENTS**

For ride can by paid card or cash

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| payment\_src\_id | unique identifier for payment | bigint |
| payment\_type | Card/Cash | varchar |
| payment\_datetime | payment time | datetime |

|  |  |  |
| --- | --- | --- |
| payment\_id | payment\_type | payment\_time |
| 1015 | Card | 6/12/2016 0:57 |
| 1158 | Cash | NULL |

Example with filled data

* **DIM\_RATES**

Taxi rates base fare plus rate per mile.

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| rate\_src\_id | unique identifier for rate | bigint |
| base\_fare | base fare | decimal(10,2) |
| rate\_per\_mile | rate per mile | decimal(10,2) |

|  |  |  |
| --- | --- | --- |
| rate\_id | base\_fare | rate\_per\_mile |
| 888 | 2.50 | 1.50 |

Example with filled data

* **DIM\_PROMOTIONS**

Taxi promo codes. With discount: 10%, 20%.

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| promo\_src\_id | unique identifier for promotion | int |
| promo\_code | promo code with discount | varchar |
| discount\_percentage | discount percentage | decimal(5,2) |

|  |  |  |
| --- | --- | --- |
| promo\_id | promo\_code | discount\_percentage |
| 666 | Promo10 | 10 |

Example with filled data

* **DIM\_TRIP\_LOCATIONS**

Information about taxi trip locations point A and point B

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| location\_src\_id | unique identifier for location | varchar |
| pickup\_longitude | pickup longitude | decimal (10, 6) |
| pickup\_latitude | pickup latitude | decimal (10, 6) |
| dropoff\_longitude | dropoff longitude | decimal (10, 6) |
| dropoff\_latitude | dropoff latitude | decimal (10, 6) |
| pickup\_datetime | pickup datetime | datetime |
| dropoff\_datetime | dropoff datetime | datetime |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| trip\_id | pickup\_longitude | pickup\_latitude | dropoff\_longitude | dropoff\_latitude | pickup\_datetime | dropoff\_datetime |
| id18133569 | -73.98041534 | 40.73856354 | -73.9994812 | 40.73115158 | 6/12/2016 0:43 | 6/12/2016 0:54 |

Example with filled data

#### Facts

* **FCT\_TAXI\_TRIPS**

Fact table with references on all dimensional tables (context). Plus measurements: distance, duration (fact table = context + measurements)

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| vendor\_src\_id | vendor id (FK) | bigint |
| customer\_id | customer id (FK) | bigint |
| booking\_id | booking id (FK) | bigint |
| payment\_id | payment id (FK) | bigint |
| rate\_id | rate id (FK) | bigint |
| promo\_id | promotion id (FK) | bigint |
| location\_id | location id pickup and dropoff (FK) | bigint |
| distance\_miles | measure; trip distance | decimal(10, 2) |
| amount | measure; trip amount | decimal(10,2) |

Example with filled data

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vendor\_id | customer\_id | booking\_id | payment\_id | rate\_id | promo\_id | trip\_id | distance\_miles | amount |
| 1 | 101 | 5001 | 2001 | 5 | 7001 | 9001 | 12.5 | 25.75 |

# Business Layer 3NF

## Definitions & Acronyms

• SQL (Structured Query Language): The standard language used to communicate with and manipulate databases.

• PK (Primary Key): A unique identifier for a record in a table.

• FK (Foreign Key): A field in one table that uniquely identifies a row of another table, creating a relationship between the two tables.

NOT NULL: attribute in a table must have a value.

• PL/pgSQL: A procedural language supported by PostgreSQL that allows for more complex operations and control structures in SQL scripts.

m:m

: Many-to-many relationship in the database, where multiple records from one table can be associated with multiple records from another table.

1:m (m:1)

: One-to-many (many-to-one) relationship in the database, where one record from a table is associated with many records from another table.

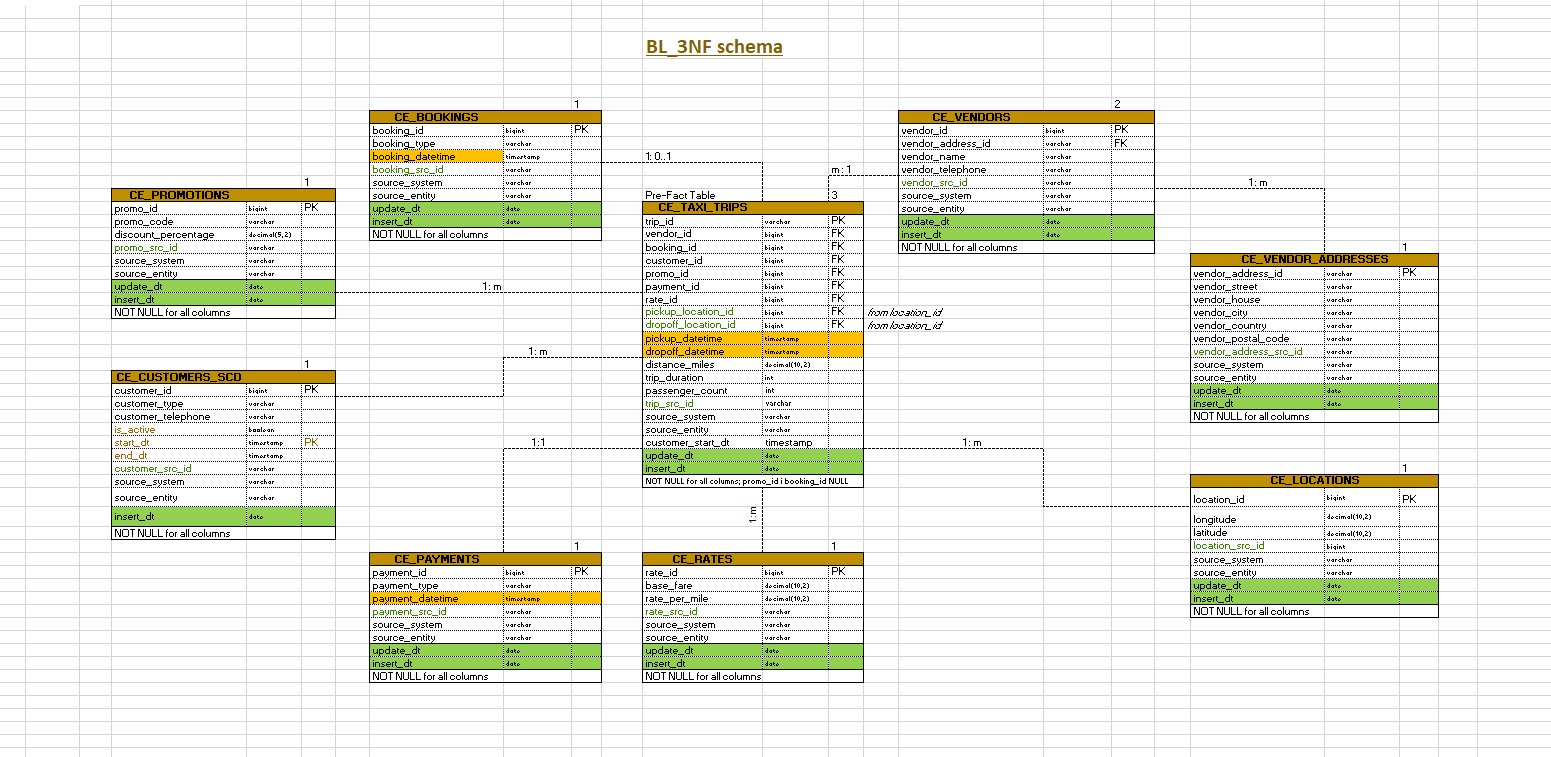
1:1

: One-to-one relationship in the database, where one record from a table is associated with one record from another table.

1:0...1

: Indicates an optional one-to-one relationship. In SQL, this means that each record in Table A can be associated with at most one record in Table B, but the association is optional.

## Logical Scheme



## Objects

1. **CE\_VENDORS table description**

This table stores information about taxi companies. It also includes metadata for tracking the source of the data and the entity within the source system.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| CE\_VENDORS | vendor\_id | surrogate key for for each vendor (PK) | bigint |
| vendor\_address\_id | reference to the vendor record in ce\_vendor\_addresses (FK) | varchar |
| vendor\_name | vendor name | varchar |
| vendor\_telephone | vendor telephone number | varchar |
| vendor\_src\_id | business key of the vendor record from the source system | varchar |
| source\_system | system where the data originates from (datasets) | varchar |
| source\_entity | entity within the source system | varchar |
| update\_dt | date of update data | date |
| insert\_dt | date of first insert of data | date |

Comments on table relationships:

* CE\_VENDORS ↔ CE\_TAXI\_TRIPS: one-to-many (1:m) relationship;
* CE\_VENDORS ↔ CE\_VENDOR\_ADDRESSES: one-to-many (1:m) relationship.

Additional constraints: NOT NULL for all columns

1. **CE\_VENDOR\_ADDRESSES table description**

This table stores information about taxi companies. It also includes metadata for tracking the source of the data and the entity within the source system.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| CE\_VENDOR\_ADDRESSES | vendor\_address\_id | surrogate key for each vendor address (PK) | varchar |
| vendor\_street | vendor street | varchar |
| vendor\_house | vendor telephone house | varchar |
| vendor\_city | vendor city | varchar |
| vendor\_country | vendor country | varchar |
| vendor\_postal\_code | vendor postal code | varchar |
| vendror\_address\_src\_id | business key of the vendor address record from the source system | varchar |
| source\_system | system where the data originates from (datasets) | varchar |
| source\_entity | entity within the source system | varchar |
| update\_dt | date of update data | date |
| insert\_dt | date of first insert of data | date |

Comments on table relationships:

* CE\_VENDORS ↔ CE\_VENDOR\_ADDRESSES: one-to-many (1:m) relationship.

Additional constraints: NOT NULL for all columns

1. **CE\_CUSTOMERS\_SCD table description**

The CE\_CUSTOMERS table stores information about customers, including a unique identifier, customer type, number of passengers, and contact details. It also includes metadata for tracking the source of the data and the entity within the source system.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| CE\_CUSTOMERS\_SCD | customer\_id | surrogate key for the customer record (PK) | bigint |
| customer\_type | Individual/Business | varchar |
| customer\_telephone | customer telephone | varchar |
| is\_active | Indicates if the rate is currently active | boolean |
| start\_dt | Start date of the rate's validity (SCD Type 2 field) (PK) | timestamp |
| end\_dt | End date of the rate's validity (SCD Type 2 field) | timestamp |
| customer\_src\_id | business key of the customer record from the source system | varchar |
| source\_system | system where the data originates from (datasets) | varchar |
| source\_entity | entity within the source system | varchar |
| insert\_dt | date of first insert of data | date |

Comments on table relationships:

* CE\_CUSTOMERS ↔ CE\_TAXI\_TRIPS: one-to-many (1:m) relationship.

Additional constraints: NOT NULL for all columns

1. **CE\_BOOKINGS table description**

The CE\_BOOKINGS table contains information about customer bookings, including a unique identifier, associated customer, trip, rate and promo IDs, booking type, and the time of booking. It also tracks the source of the data and the entity within the source system.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| CE\_BOOKINGS | booking\_id | surrogate key for each booking (PK) | bigint |
| booking\_type | business key of the booking record from the source system (natural key) | varchar |
| booking\_datetime | booking time | timestamp |
| booking\_src\_id | business key of the booking record from the source system | varchar |
| source\_system | system where the data originates from (datasets) | varchar |
| source\_entity | entity within the source system | varchar |
| update\_dt | date of update data | date |
| insert\_dt | date of first insert of data | date |

Comments on table relationships:

* CE\_BOOKINGS ↔ CE\_TAXI\_TRIPS: many-to-one (m:1) relationship.

Additional constraints: NOT NULL for all columns

1. **CE\_PAYMENTS table description**

The **CE\_PAYMENTS** table stores information about payments made for bookings, including a unique payment identifier, the booking ID it relates to, payment type, and the time of payment. It also includes metadata for tracking the source of the data and the entity within the source system.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| CE\_PAYMENTS | payment\_id | surrogate key for each payment (PK) | bigint |
| payment\_type | payment type | varchar |
| payment\_datetime | payment time | timestamp |
| payment\_src\_id | business key of the payment record from the source system | varchar |
| source\_system | system where the data originates from (datasets) | varchar |
| source\_entity | entity within the source system | varchar |
| update\_dt | date of update data | date |
| insert\_dt | date of first insert of data | date |

Comments on table relationships:

* CE\_TAXI\_TRIPS ↔ CE\_PAYMENTS: one-to-one (1: 1) relationship.

Additional constraints: NOT NULL for all columns

1. **CE\_RATES table description**

The **CE\_RATES** table stores pricing information for taxi trips, including base fare, per-mile rates.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| CE\_RATES | rate\_id | surrogate key for each rate (PK) | bigint |
| base\_fare | base fare for the transportation service | decimal(10,2) |
| rate\_per\_mile | Rate per mile for the transportation service | decimal(10,2) |
| rate\_src\_id | business key of the rate record from the source system | varchar |
| source\_system | system where the data originates from (datasets) | varchar |
| source\_entity | entity within the source system | varchar |
| update\_dt | date of update data | date |
| insert\_dt | date of first insert of data | date |

Comments on table relationships:

• CE\_TAXI\_TRIPS ↔ CE\_RATES\_SCD: many-to-one (m:1) relationship.

Additional constraints: NOT NULL for all columns

1. **CE\_PROMOTIONS table description**

This table stores promotional information related to transportation services. It tracks active and past promotions, including the promotional code, the discount offered, and metadata about the source system from which the data originates.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| CE\_PROMOTIONS | promo\_id | surrogate key for each promotion (PK) | bigint |
| promo\_code | promotion code | varchar |
| discount\_percentage | discount percentage | decimal(5,2) |
| promo\_src\_id | business key of the promo record from the source system | varchar |
| source\_system | system where the data originates from (datasets) | varchar |
| source\_entity | entity within the source system | varchar |
| update\_dt | date of update data | date |
| insert\_dt | date of first insert of data | date |

Comments on table relationships:

• CE\_TAXI\_TRIPS ↔ CE\_PROMOTIONS: many-to-one (m:1) relationship.

Additional constraints: NOT NULL for all columns

1. **CE\_TAXI\_TRIPS table description**

This table stores information about individual trips, including geographic data related to pickup and drop-off locations, as well as timestamps for the trip's start and end. It serves as a core component for tracking the details of transportation journeys. It also includes metadata for tracking the source of the data and the entity within the source system.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| CE\_TRIPS | trip\_id | surrogate key for each trip (PK) | bigint |
| vendor\_id | Reference to the vendor record in CE\_VENDORS (FK) | bigint |
| booking\_id | Reference to the booking record in CE\_BOOKING (FK) | bigint |
| customer\_id | Reference to the customer record in CE\_CUSTOMERS\_SCD (FK) | bigint |
| promo\_id | promotion id (FK) | bigint |
| payment\_id | Reference to the payment record in CE\_PAYMENTS (FK) | bigint |
| rate\_id | Reference to the rates record in CE\_RATES (FK) | bigint |
| pickup\_location\_id | Foreign key referencing the location\_src\_id in the CE\_LOCATIONS tabl e(FK) | bigint |
| dropoff\_location\_id | Foreign key referencing the location\_src\_id in the CE\_LOCATIONS table (FK) | bigint |
| pickup\_datetime | Date and time when the trip was picked up | timestamp |
| dropoff\_datetime | Date and time when the trip was completed (dropped off) | timestamp |
| distance\_miles | The total distance traveled during the trip (in miles) | decimal(10,2) |
| trip\_duration | Total duration of the trip (in seconds) | int |
| passenger\_count | Number of passengers in the trip | int |
| trip\_src\_id | business key of the trip record from the source system | varchar |
| source\_system | system where the data originates from (datasets) | varchar |
| source\_entity | entity within the source system | varchar |
| customer\_start\_dt | reference to customers start\_dt | timestamp |
| update\_dt | date of update data | date |
| insert\_dt | date of first insert of data | date |

Comments on table relationships:

* CE\_TAXI\_TRIPS ↔ CE\_VENDORS: many-to-one (m:1) relationship;
* CE\_BOOKINGS ↔ CE\_TAXI\_TRIPS: many-to-one (m:1) relationship;
* CE\_PROMOTIONS ↔ CE\_TAXI\_TRIPS: one-to-one (1:m) relationship;
* CE\_TAXI\_TRIPS ↔ CE\_LOCATIONS: one-to-many (1:m) relationship;
* CE\_TAXI\_TRIPS ↔ CE\_RATES: many-to-one (m:1) relationship;
* CE\_TAXI\_TRIPS ↔ CE\_PAYMENTS: one-to-one (1:1) relationship;
* CE\_TAXI\_TRIPS ↔ CE\_CUSTOMERS\_SCD: many-to-one (m:1) relationship.

Additional constraints: NOT NULL for all columns; promo\_id and booking\_id NULL

1. CE\_LOCATIONS table description

The **CE\_LOCATIONS** table is a dimension table that stores information about geographical locations, including their coordinates (latitude and longitude) and unique identifiers from various source systems. It is used to standardize location data across different trips, bookings, and vendors.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| CE\_LOCATIONS | location\_id | Surrogate key for the location, uniquely identifying each record in this table (PK) | bigint |
| longitude | The longitude coordinate of the location | decimal(10,2) |
| latitude | The latitude coordinate of the location | decimal(10,2) |
| location\_src\_id | The original location identifier from the source system | varchar |
| source\_system | The system that provided the location data | varchar |
| source\_entity | The specific entity within the source system that generated the location data | varchar |
| update\_dt | date of update data | date |
| insert\_dt | date of first insert of data | date |

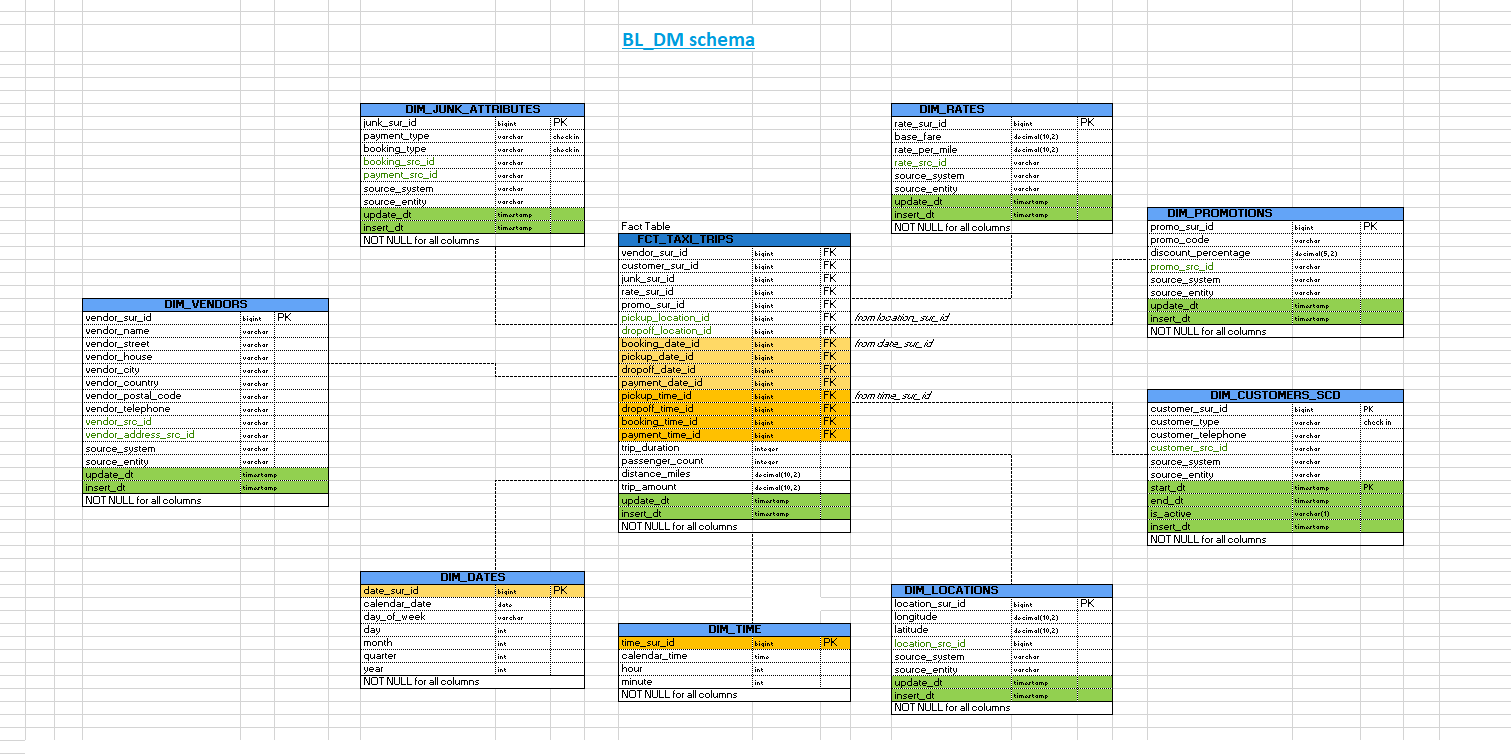
Comments on table relationships:

* CE\_TAXI\_TRIPS ↔ CE\_LOCATIONS: one-to-many (1:m) relationship.

Additional constraints: NOT NULL for all columns

# Business Layer Dimensional Model

## Logical Scheme



## **METRICS Description**

What Are Metrics in BL\_DM?

Metrics are measurable quantitative indicators used for analysis, evaluation, and monitoring of data. In the context of the BL\_DM schema, metrics are stored in fact tables and represent numerical or categorical information that helps analyze business data.

Key Properties of Metrics:

Measurable: Metrics are numerical or have fixed categories (e.g., sums, counts, durations).

Aggregation: Metrics can be aggregated using functions like SUM, COUNT, AVG, MAX, MIN.

Stored in Fact Tables: Metrics are typically found in fact tables (e.g., FCT\_TAXI\_TRIPS) and are linked to dimensions (dimension tables) that provide analytical context.

**Metrics in my tables:**

1. DIM\_JUNK\_ATTRIBUTES

This table does not directly contain metrics but holds categorical attributes like payment\_type and booking\_type that can be used for filtering or grouping data in analytical queries.

2. DIM\_RATES

This dimension contains no direct metrics but provides reference values such as:

* base\_fare: Base cost for a trip.
* rate\_per\_mile: Cost per mile traveled.

These attributes are used in calculating metrics in the fact table, such as trip\_amount.

3. DIM\_PROMOTIONS

No direct metrics in this table. However, discount\_percentage is a key attribute used for calculating the final trip cost (trip\_amount) in the fact table.

4. DIM\_VENDORS

No direct metrics in this table. It contains descriptive data like vendor details, which can be used to analyze vendor-specific metrics in the fact table.

5. DIM\_CUSTOMERS\_SCD

This table does not store metrics. It is primarily used for tracking customer attributes (e.g., type, telephone) and managing historical data with start\_dt and end\_dt.

6. DIM\_DATES

Contains no direct metrics. It is a date reference table used to connect and analyze metrics in the fact table by day, month, quarter, or year.

7. DIM\_TIME

Contains no direct metrics. It is a time reference table used to analyze time-related metrics in the fact table, such as trip\_duration.

8. DIM\_LOCATIONS

Contains no direct metrics. It provides reference information for trip pickup and dropoff locations.

9. FCT\_TAXI\_TRIPS (Fact Table)

This is the primary table containing metrics for analysis. Metrics include:

* trip\_duration: Duration of the trip (in minutes).
* passenger\_count: Number of passengers for the trip.
* distance\_miles: Distance covered during the trip (in miles).
* trip\_amount: Total cost of the trip (calculated using base fare, rate per mile, and any promotions).

The fact table is linked to the dimension tables to enable multidimensional analysis of these metrics.

Metrics in BL\_DM are primarily stored in the fact table (FCT\_TAXI\_TRIPS).

Dimension tables do not directly store metrics but provide descriptive or categorical data that enrich and contextualize the metrics in the fact table.

Metrics such as trip\_duration, trip\_amount, and distance\_miles are the primary values analyzed for insights, using the dimensions for filtering and aggregation.

These metrics cover the various elements of a taxi trip, such as rates, vendors, customers, locations, time and promotions. The main fact table (FCT\_TAXI\_TRIPS) will likely be used to store transactional data about each trip.

# Logical Scheme

# Data Flow

# Fact Table Partitioning Strategy

Question:

• Please reiterate the purpose of staging area and explain why it is not possible to load data directly into the 3NF layer?

Answer:

Loading into the PostgreSQL database is the staging area for source tables (Staging Source Tables):

• Data is physically loaded into the database.

• It is an intermediate layer (Staging Area) before transformation into the storage model.

• Allows for validation, deduplication, and enrichment of data before loading into the warehouse.

The Staging Area is used as a temporary storage for raw data. Here, data is extracted from source systems, cleaned (duplicates are removed, missing values are filled, and formats are standardized), and integrated into a unified schema.

Loading data directly into the 3NF (Third Normal Form) layer is not recommended because:

Performance issues: Direct loading can cause performance problems due to complex relationships and constraints.

Data quality risks: Raw data may contain errors, inconsistencies, or duplicates, which can negatively impact data integrity in the warehouse.

Idempotency: The staging area ensures that the ETL process is idempotent (re-running the load does not create duplicate records) and allows for pre-validation and data cleansing before data reaches the 3NF layer.