

|  |
| --- |
| Business Template  **ChicaGo CICLISTIC 2023** |
|  |



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)

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

[4 Logical Scheme 4](#_Toc155614194)

[5 Data Flow 4](#_Toc155614195)

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

# 

# Business Description

## Business background

Cyclistic 2023 is a dynamic and innovative bike-sharing service operating in the vibrant city of Chicago. In collaboration with Divvy Bikes, the service provides convenient and eco-friendly transportation options to residents and visitors alike. The business thrives on the idea of making urban mobility seamless, efficient, and sustainable.

## Problems because of poor data management

Ineffective data management poses a significant hurdle for Cyclistic 2023. Without proper tools for analysis, the business risks insufficient insights into user behavior, popular routes, and market trends. This hampers strategic decision-making, puts the company at a competitive disadvantage, and limits the formulation of targeted business strategies. To thrive in the bike-sharing market, Cyclistic 2023 must prioritize data utilization and invest in advanced analytical tools for informed and proactive operations.

## Benefits from implementing a Data Warehouse

The integration of a data warehouse at Cyclistic 2023 unlocks key advantages, addressing prior data management challenges and enhancing strategic decision-making. Through this implementation, Cyclistic 2023 gains the ability to:

* **Optimize Bike Distribution:**

Identify demand patterns for electric and classic bikes, ensuring strategic placement for optimal user accessibility.Which ones have the widest distribution of prices?

* **User Behavior Patterns:**

Correlate user behaviors with ride patterns, enabling targeted initiatives to enhance user satisfaction and improve service offerings.

* **Diversity in Bike Usage:**

Understand user preferences between electric and classic bikes, facilitating adjustments to the fleet for more effective meeting of user demands.

* **Strategic Marketing Decisions:**

Utilize data insights to inform marketing strategies, targeting peak usage times, popular routes, and preferred bike types

* **And many other.**

## DATASETS DESCRIPTION

The first dataset contains the following information about rents on Chicagos market.

Cycle Information:

Cycle type: The type of cycle (electric or classic).

Location Information:

Start station: In which station ride started.

End station: In which station ride ended.

Start lat: Start coordinates.

Start lng long: Start coordinates.

End lat: End coordinates.

End lng: End coordinates.

Customer Information:

Name: Name of the customer

Last\_name: Last name of the customer

Gender: The gender who rented the bike (men or women).

Age: The age range of the customer.

Member: The customer registered in our system or not.

Time:

Start\_date: The date when ride started.

Start\_time: The time when ride started.

End\_date: The date when ride ended.

End\_time: The time when ride ended.

Ride\_time\_mins: Duration of the ride.

Price:

Price\_for\_min: The price per minute, it’s different for the electric and classic cycles

Dicount: Dicount by percent depending on rent duration

The second dataset is for online paid rides; it mostly contains the same dates, and additionally, it has the following information...

Payment:

Bank\_account:

Bank\_name:

Payment\_date:

Payment\_amount:

## GRAIN / DIM / FACT

Grain description 4 steps

1. Business process

The business process is related to rents and transactions.

The fact table (ride\_fact) stores quantitative data about ride and the dimension tables contextual to uniquely identify a single row. The business process is to manage and analyze the renal of bikes and improve performance. Here we have dim tables which are related with fact table with PK’s – FK’s to see where who, when and which bike was rented.

1. Grain Description

The fact table, "RideFact," represents individual online rides and captures detailed information about each ride event. Each row in the fact table corresponds to a unique online ride and contains specific details about the ride, including the ride type, location, date and time of the beginning and finish, ride duration, pricing information, customer details, ride-specific information, and payment details.

1. Identify the dimensions

* Dim\_Location:

This dimension captures information about the locations where rides start and finish, focusing on the unique identifier (LocationID) and the descriptive name of the location.

* Dim\_customer:

This dimension contains details about the customers who use the ride service, emphasizing customer demographic information.

* Dim\_time:

This dimension provides a time reference for each ride, capturing both the date and time of the beginning and finish events, along with a seasonal attribute.

* Dim\_ride:

This dimension focuses on ride-specific information, including a unique identifier (RideID) and the type of ride.

1. Identify facts

Here we Identify the measurable and numeric data that we want to analyze. These are the facts, in the fact table. In our dataset, the ride\_fact table contains facts such as "RideDuration," "MinutesPrice," "ChargePercent," etc.

**Ride\_fact**

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| online\_ride\_id | DD, natural PK | varchar(256) |
| location\_id | FK | Int |
| beginning\_date\_id | FK | date |
| ride\_duration | Measure | int |
| minutes\_price | Measure | Int |
| charge\_percent | Measure | Decimal |
| customer\_id | FK | Int |
| cycle\_id | FK | int |
| payment\_date\_time |  | timestamp |
| payment\_amount | Measure | int |
| Update\_dt | Last update, generated automatically | timestamp |

|  |  |  |  |
| --- | --- | --- | --- |
| Online\_ride\_id | Location\_id | Beginning\_date\_time\_id | Ride\_duration |
| BD88A2E670661CE53 | 1 | 2023-01-31 | 10 |

**Dim\_locations**

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| Location\_id | PK | bigint |
| beginning\_id | Identifier of street, there can be a lot of streets with the same name | int |
| beginning\_location | street | Varchar(256) |
| start\_latitude | Location details | Varchar(256) |
| start\_longitude | Location details | Varchar(256) |
| finish\_id |  | Int |
| finish\_location | street | Varchar(256) |
| end\_latitude | Location details | Varchar(256) |
| end\_longitude | Location details | Varchar(256) |
| update\_dt | Last update, generated automatically | timestamp |

|  |  |  |  |
| --- | --- | --- | --- |
| Location\_id | beginning\_id | beginning\_location | Start\_latitude |
| 23 | 1 | North Street | 10456 |

**Dim\_customers**

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| Customer\_id | PK | bigint |
| first\_name |  | Varchar(256) |
| last\_name |  | Varchar(256) |
| birth\_date |  | date |
| gender |  | Varchar(256) |
| bank\_account |  | Varchar(256) |
| bank\_name |  | Varchar(256) |
| customer\_status | member/non-member | Varchar(256) |
| update\_dt | Last update, generated automatically | timestamp |

|  |  |  |  |
| --- | --- | --- | --- |
| Customer\_id | First\_name | Last\_name | Birth\_date |
| 2 | Name | Surname | 2000-01-31 |

**Dim\_times**

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| beginning\_date\_id | PK | date |
| beginning\_time | Time when the ride started | time |
| finish\_date |  | date |
| finish\_time |  | time |

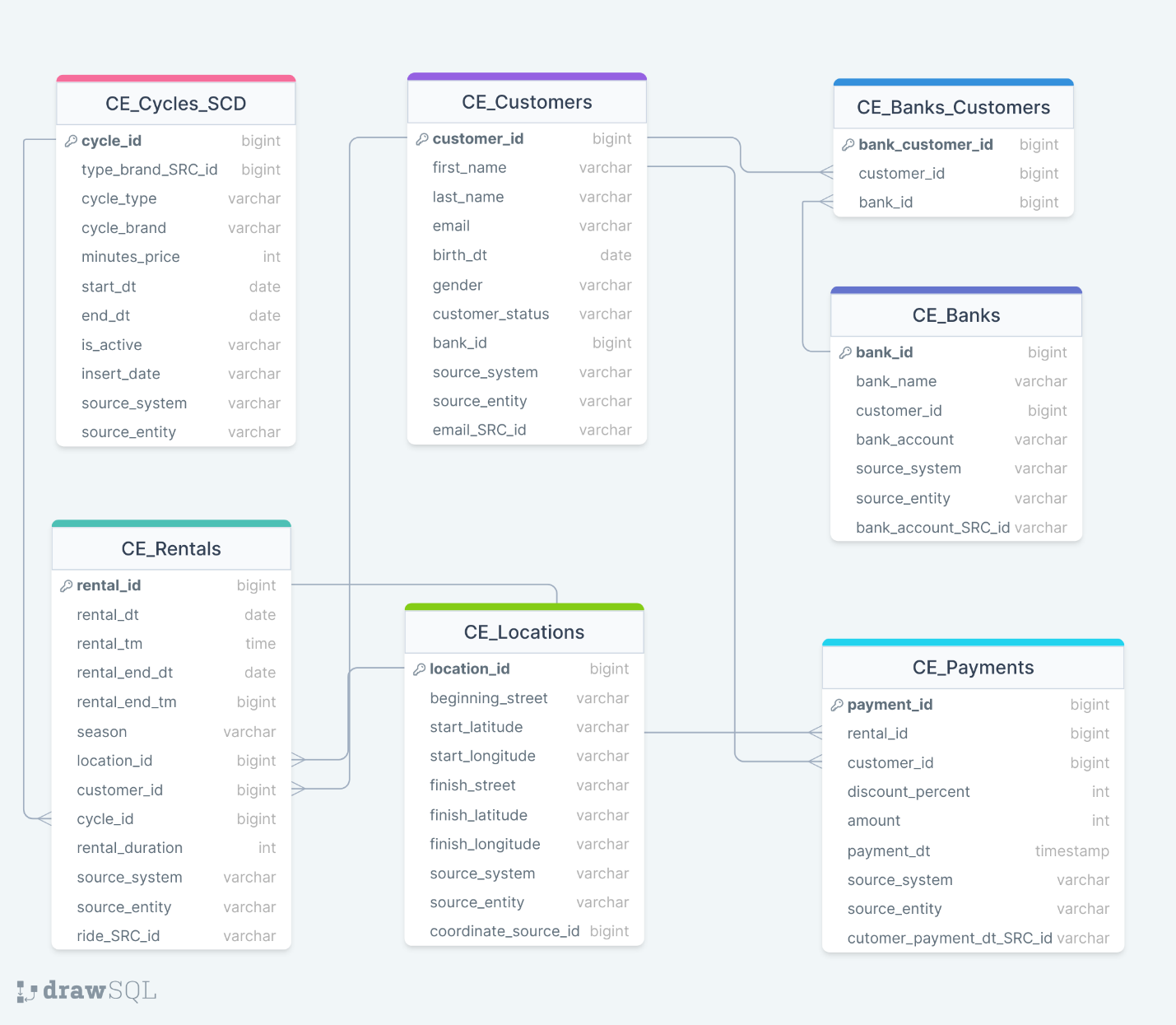
|  |  |  |  |
| --- | --- | --- | --- |
| Beginning\_date\_time | Beginning\_time | Finish\_date | Finish\_time |
| 2023-01-20 | 14:25:00 | 2023-01-20 | 15:25:00 |

**Dim\_cycle**

|  |  |  |
| --- | --- | --- |
| Column name | Description | Data Type |
| cycle\_id | PK | bigint |
| cycle\_type |  | Varchar(256) |
| cycle\_brand |  | Varchar(256) |
| update\_dt |  | timestamp |

|  |  |  |  |
| --- | --- | --- | --- |
| cycle\_id | cycle\_type | cycle\_branch | update\_dt |
| 3 | classic | trek | 2020-11-12 |

# Business Layer 3NF



Description source table to 3NF

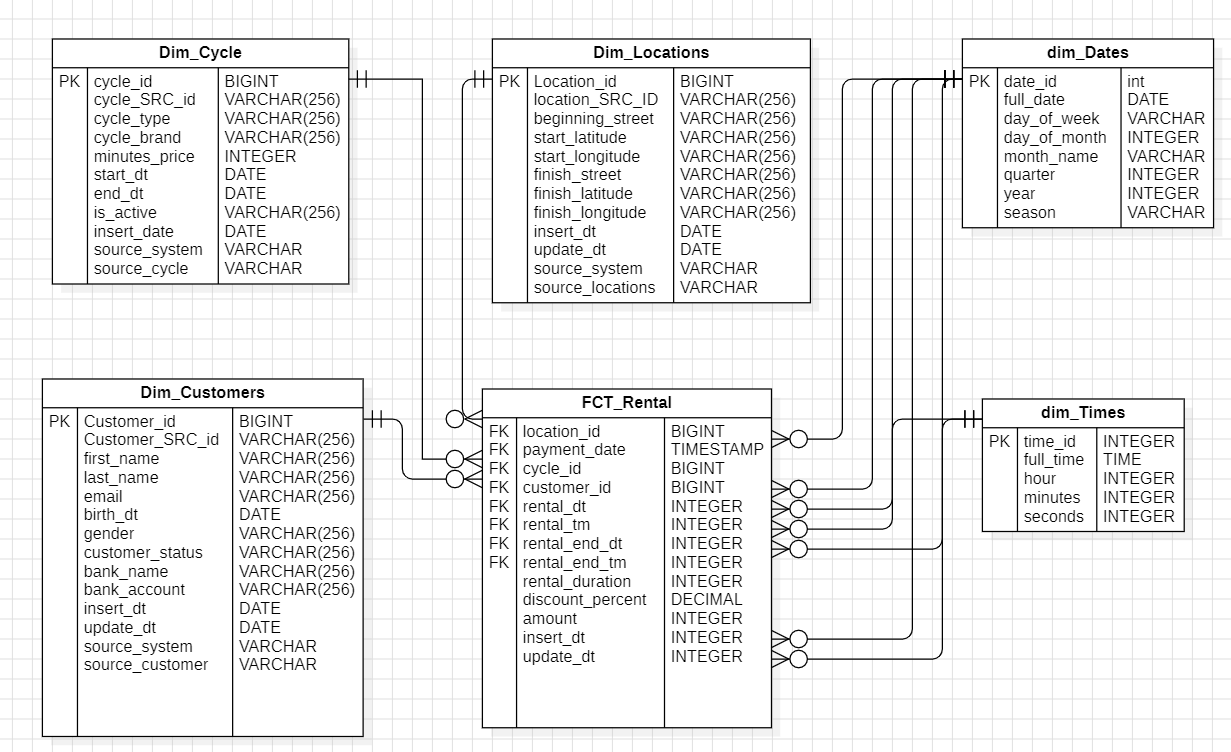
To bring it to 3NF we need to define our entities.   
Here we’ll have cycle, customer, bank, location, payment and rental entities. Considering that each attribute should be fully dependent on PK we split by entity and add PK for each table. E.g all attributes which are related with payment find place in payment table. The same for all entities. Here we have a bridge table CE\_Banks\_Customers, because a customer can have account in many banks and of course bank can have many customers. In CE\_Rentals we store exact info about rental (transactions) how long is rental rudation who rented, location, in what season was a cycle rented etc. Almost all tables have source triplet, except CE\_Banks\_Customers table which is a bridge table. In source\_system we’ll later add exact schema name where the data was captured from. Source\_entity will refer the table name where the data came from and finally in SRC\_id which is the natural key to identify exact row. Couple words about SRC\_ID. As we know SRC\_ID should be natural\_key. Initially the natural key we had it was ride\_id (online\_ride\_id) for the rest we create a natural key which will identify uniqueness of a row.

In CE\_Cycles\_SCD natural key will be the combination of cycle type and brand, it’s SCD type 2, which we will discuss later in this description. In CE\_Customers natural key will be email address, which is in general unique for each person, that’s why we decide to set it as email\_SRC\_ID. In CE\_Banks it will be bank\_account because it’s also unique for each person and can be a natural key, it’s called as bank\_account\_SRC\_ID. In CE\_Locations to identify unique row we can use concatenation of start lang, long and end land, long which we call coordinate\_SRC\_ID. In CE\_Payments the we create natural key using customer\_id and payment date, and we call it customer\_payment\_dt\_SRC\_ID. Finally, CE\_RENTALS we have natural key from our source which we’ll call as ride\_SRC\_ID.

According to our task we should have one table in SCD Type 2. Considering that CE\_Cycle is a table where the date is not changing every day. It’s not a part of transactions but we use it in our CE\_Rentals to know which type of bike was rented and the price. In this case the attribute that can be changed is minutes\_price, but we need to store old price for the future analytical purposes. Using type 2 we also store the date when the value was changed and end date of old value. The end of new value we set like 01/01/9999 an unreal date in the future. That’s the reason I have chosen CE\_CYCLE which can be as SCD TYPE 2 where can the minutes\_price be changed and the change history is important for us.

The rest tables in Type 1 (As it was mentioned in naming convention) which is to overwrite the old value but it depends on business. I this case the other tables can have a need for some changes to overwrite some values. That’s also one of the reasons I make other tables (CE\_Rentals, CE\_Customers, CE\_Payments and CE\_Locations) SCD Type 1.

# Business Layer Dimensional Model



describe all your metrics

In fact table we get amount that should be paid, it’s in USD,

We get amount multiplying rental\_duration \* disount\_percent \* minutes\_price

Discount is between 0,8 to 1. If the client don’t have discount then it multiplies by 1.  
Discount depends on rental\_duration. E.g. rental duration more than 10 minutes then discount is 0,9

More than 30 minutes 0,8.

# Logical Scheme

# Data Flow

# Fact Table Partitioning Strategy