UNIVERSITÀ DI PISA

LABORATORY OF DSS
Group 12

Rustemi Ajla 598227
Toresan Stefano 649285
Wiwatwicha Na 682482

Contents

1 Introduction

This project is centered on the development of a data warehouse aimed at analyzing traffic incidents in Chicago. It simulates a decision support system for an insurance company, with the primary objective of creating a structured analytical framework. The framework facilitates the evaluation of claims, identification of risk patterns, and support for operational decisions within the insurance domain.

The initial phase of the project involves the creation and population of a database using data sourced from multiple files. This process includes the cleaning, transformation, and integration of raw data to ensure consistency and usability. The data handling primarily relies on SQL Server Integration Services (SSIS), with minimal use of SQL commands to maintain adherence to client-side computation requirements. SSIS nodes are utilized for data processing and transformation, offering a versatile and intuitive toolset for addressing complex analytical challenges.

By integrating and analyzing claims data along with contributing factors—such as road conditions, vehicle types, and accident locations—the system enables data-driven decision-making. This ultimately aids in optimizing operations and improving risk management strategies for the insurance sector.

2 Assignment 1 and 2 - Data Understanding and Cleaning

The three datasets were:

- **Crashes Dataset:** a table with data about road incidents between January 2014 and January 2019 in Chicago, USA. The same table also includes information about the causes of the incident, some road properties, and some information about the reported injuries. The table has *257925 rows × 36 columns* containing this informations with two candidate keys on a table, respectivly 'CRASH_DATE' and 'RD_NO'. The one selected as a primary key is 'RD_NO'

- **Vehicle Dataset:** contains information about the vehicle(s) involved in the incidents. This file includes information about the vehicle and the information collected by the police after the incident, having in total *460437 rows × 17 columns* with 'CRASH_UNIT_ID' as primary key and 'RD_NO' as a foreign key

- **People Dataset:** contains information about the people involved in the incidents, including their sex, age, city of residency, etc. Having in total *564565 rows × 19 columns*, with 'PERSON_ID' as primary key and 'RD_NO' and 'VEHICLE_ID' as foreign keys. Duplicate data about a person occurs when there are multiple damage values, meaning that person receives multiple claims.

The initial data sources included multiple tables containing detailed information about insurance claims, policyholders, vehicles, geographical regions, accident causes, and time-related attributes. For features, such as longitude and latitude *GeoHashing* was used to identify and visualize crash-prone areas, helping us look at the pinpoint high-risk locations, like the street 'MICHIGAN AVE' with a total of crashes equal to 7496.

Each dataset was carefully analyzed to ensure consistency and completeness. Data cleaning involved handling missing numeric and categorical values, such as '*damage_value*', '*vehicle_id*', '*beat_id*', '*age*', '*gender*', etc. Other transformations were, correcting inconsistent formats, and removing duplicates. For example, date fields were standardized to the 'YYYY-MM-DD' format, and textual fields were normalized to ensure uniformity.

Next, the data was transformed into dimensions and a fact table to align with the goals of the star schema.

3 Assignment 3 - Data-warehouse Schema

The schema that we choose to follow for this task is star schema. The star schema was designed to provide an efficient and scalable analytical framework for evaluating insurance claims. Its central purpose is to enable the comprehensive analysis of claims data in conjunction with factors such as accident causes, policyholder demographics, and geographical trends. This approach facilitates improved decision-making, particularly in exploring temporal patterns and tracking variations in claims over time.

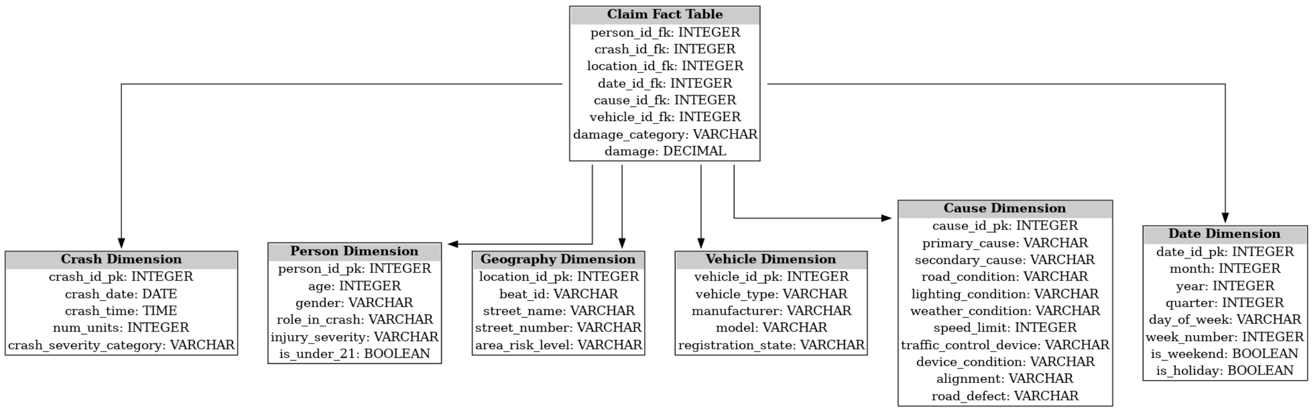

Figure 1: DW schema

Structure of the Star Schema

The star schema consists of a central fact table surrounded by several dimension tables. The fact table serves as the core repository for quantitative data on insurance claims, while the dimension tables provide contextual information to enrich the analysis.

- **Fact Table**: The fact table contains granular data at the level of individual crashes, including measurable metrics such as total damage amounts. It also features foreign keys linking to associated dimensions.
- **Dimension Tables**: These provide a multi-faceted context for analysis:
  - **Date Dimension**: Enables exploration of claims data across various timeframes, including monthly and annual trends. Attributes like holidays and weekends support seasonality studies.
  - **Geography Dimension**: Facilitates the evaluation of regional variations in claim frequency and severity, aiding in location-based risk management.
  - **Crashes Dimension**: Captures crashes details, to support segmentation and trend analysis.

- **Vehicle Dimension**: Provides insights into trends related to vehicle make, model, and type, which are critical for assessing vehicle-specific risk factors.
- **Cause Dimension**: Documents accident-related details, such as road conditions and weather, helping to identify risk patterns and inform preventative measures.
- **People Dimension**: Captures people details like age, sex, damage, crash, etc.

Justification for the Design

The star schema's simplicity and performance advantages were key factors in its selection. Its denormalized structure simplifies SQL queries, making data exploration more intuitive for analysts. Moreover, the schema is optimized for frequent and high-speed analytical queries, which are essential for business intelligence applications.

For the insurance company, this design supports actionable insights, such as identifying correlations between weather conditions and accident severity or evaluating the profitability of various policy types. Additionally, the schema's scalability ensures it can accommodate new dimensions or measures with minimal disruption, supporting long-term adaptability. After designing the schema we prepared our split files to meet the requirement in assignment 4

4 Assignment 4 - Data Preparation

The data was split into a fact table and multiple dimension tables, aligned with the star schema design principles. The primary key for the fact table is (`RD_NO`), and the table included metrics such as total damage, the number of vehicles involved (`NUM_UNITS`), and crash time.

The following dimensions were created to provide context for the fact table:

- **Date Dimension:** Extracted fields such as year, month, day, and quarter from the `CRASH_DATE`. A `holiday_flag` was added to indicate weekends and public holidays.
- **Person Dimension:** Categorized participants by age groups (e.g., `<21`, `21-40`) and roles such as driver or passenger.
- **Vehicle Dimension:** Stored vehicle attributes like type, make, model, and year of registration.
- **Geography Dimension:** Encoded `LATITUDE` and `LONGITUDE` into GeoHash values with a precision of 7 to group crashes by geographical areas.
- **Cause Dimension:** Included primary and secondary contributory causes, weather conditions, road conditions, and traffic control devices.
- **Crash Dimension:** Primary key was'RD_NO', and attributes such as time, 'NUM_UNITS', and, 'crash_severity_category'

Since in this project, the dataset includes latitude and longitude values for crash locations. These coordinates are difficult to group directly because they are floating-point values with high precision and they require a method to efficiently identify clusters of incidents in specific geographic areas. So to solve this problems we used GeoHash, library 'geohash2'. It was chosen because it simplifies these tasks by encoding the coordinates, *Latitude* and *longitude* were converted into a single alphanumeric string, making it easier to group or compare regions. GeoHash strings are hierarchical, so shorter strings represent larger areas, while longer strings represent smaller areas. This enables analysis at different spatial resolutions. It can also be used as keys in databases to quickly filter or query locations. At precision 7, GeoHash encodeded a location to a grid approximately 153 x 153 meters, which is fine-grained enough for traffic incident analysis in a city like Chicago.

The fact table and dimension tables are now ready for analytical tasks such as identifying high-risk geographical areas based on crash counts, computing age-based ratios for participants involved in crashes and analyzing crash trends by weather conditions and time of day.

This structured approach ensures that the data warehouse supports efficient querying and decision-making for the insurance company.

5 Assignment 5 - Data uploading with Python

In this section we populated the database using the python library called '*pyodbs*'. We utilized fast '*executemany*' function to insert the data in batches without creating a connection for every single row. Each dimension took approximately 15 minutes for the largest tables, such as people and vehicle dimensions and we implemented rollback at the table level. Some of the challenges that we faced were strings data being unexpectedly long, for this we had to go back to the data understanding task to check the maximum string length for string type columns and we adjusted the data specification accordingly. Other problems include mismatching of the data type such as boolean and float attributes which likewise were solved by the coordination of data cleaning and the table creations..

6 Assignment 6 - SSIS

In this section we will talk about how we answered four queries using SQL Server Integration Services (SSIS).

Query 6a

For every year, show all participants ordered by the total number of crashes

For this query, we generated a table with the following structure: **Column 1:** *Year*, **Column 2:** *PersonID*, **Column 3:** *CrashCount*

For this, we tried out two approaches, but for the final result, approach 2 was chosen since it is more optimized and faster than the join-based approach.

Approach 1: Using a Join

We started by merge-joining the **ClaimFactTable**, which already contains columns for *date*, *person*, and *crash data*, with the **DateDimension** table on the *date_id* field. This allowed us to extract the *year* from the **DateDimension**. To optimize the join process, both tables were pre-sorted by the *date_id* field. After the join, we performed an aggregation to group the data by *year* and *person_id*. For each grouping, we counted the distinct *crash_id* values within a given year per participant. This ensured that duplicate counts were avoided, as participants may receive multiple claims for the same crash. The final results were exported into a file named `assignment_6a`.

Approach 2: Using a Derived Table

Instead of performing a join, we utilized a "adding derived column to the table" node to simplify the process. In the derivation node, we created an expression to generate a new column, *crash_year*, directly from the **ClaimFactTable**. This was achieved by extracting the substring corresponding to the year from the `YYYYMMDD` format of the *date* field. Once the *crash_year* column was created, we grouped the data in the same manner as before, by *crash_year* and *person_id*. Finally, we calculated the count of distinct *crash_id* values for each grouping.

Query 7a

For every police beat, compute the day-night crash index, defined as the ratio between the number of vehicles (NUM_UNITS) involved in an incident between 9 pm and 8 am, and the number of vehicles involved in an incident between 8 am and 9 pm

To begin, we needed to integrate data from multiple tables. The following joins were performed: we joined **CrashDim** with **ClaimFactTable** using the *crash_id* column. Before the join: **CrashDim** was projected to retain only the *crash_id*, *crash_time*, and *num_units* columns. **ClaimFactTable** was projected to retain only the *crash_id* and *location_id* columns. Both datasets were sorted by the *crash_id* key. The result of this join was then merged with **GeographyDim** using the *location_id* column as the key. Before this the merged dataset was sorted by *location_id*. After, **GeographyDim** was similarly sorted by *location_id* and the final output of these operations included the columns: `crash_id`, *crash_time*, *num_units*, *location_id*, and *beat_id*. The next step was to create a derivation node that would distinguish each row of data as either a *day* or *night* crash based on the *crash_time* column. The *crash_time* column was of type `DBTIMESTAMP`, requiring expressions to extract the hour of the crash. A new column was generated that categorized crashes into *day* or *night* based on the extracted hour. The dataset was then split into two subsets: *day* crashes and *night* crashes. For each subset an aggregation operation was performed to `GROUP BY` *beat_id* and calculate the `SUM` of *num_units*. The aggregation results were two separate datasets: one for *day* crashes and one for *night* crashes. The aggregated datasets were then merged using a `FULL OUTER JOIN` on the *beat_id* key. The result included *beat_id* `SUM(num_units_day)`: The sum of *num_units* for day crashes. `SUM(num_units_night)`: The sum of *num_units* for night crashes. As every beat had crashes during both day and night, there were no `NULL` values in either the numerator or denominator in the subsequent ratio calculation. Finally, a derivation node was used to compute the *Day-Night Crash Index*:

$$\text{Day-Night Crash Index} = \frac{\text{SUM(num\_units\_day)}}{\text{SUM(num\_units\_night)}}$$

This index represents the ratio of the sum of *num_units* during the day to that during the night, calculated for each `beat`. The final output table included the following columns: *beat_id*, `SUM(num_units_day)`, `SUM(num_units_night)`, *Day-Night Crash Index*

Query 8a

For each quarter, weather condition, and beat, show the average ratio of people under 21 years old to people over 21 years old involved in crashes

For this query, we needed to join dimension tables through the **ClaimFactTable** to get all of the required variables i.e. *quarter*, *weather*, *beat*, and *person age category*, at the granularity of *crash*. First, we joined the **ClaimFactTable**, keeping only *crash_id* and *person_id*, with the **PersonDimension**, **DateDimension**, **CauseDimension**, and **GeographyDimension** respectively, using the left outer join. Then we split the dataset into 2 groups based on their *age category*. For each data compartment, we group by *quarter*, *weather*, and *beat*, then count the persons involved, in columns named *young_count* and *old_count* respectively, then merging them into a single table through full outer join. Finally, we compute the ratio between people under 21 years old and over 21 years old in crashes for each grouping, using a derived column node. The code currently has no issues with the output. In the future, we can implement the pipeline in the manner to support potential 0 as a denominator.

Query 9a

For this query we had to create an interesting query our own. THe query we invented was:

Show the number of crashes over days that are holidays vs. non-holidays for each Area_Risk_Level category

For this assignment, we were interested in studying the number of crashes over the *holiday* vs. crashes during *non-holiday* periods. We had previously imported a dataset from an external source in the data preparation stage with the *holiday dates* and *holiday names*, which we further processed by extending weekends that are adjacent to *holidays* that fall on a Friday or a Monday as a *holiday* as well for practical reasons, i.e., people may treat it as "long weekends" *holidays* in real life. We choose to inspect this crash count data over the existing geographic zones based on *Area Risk Level*, which was pre-computed in the data preparation stage with a definition derived from the sum of weighted counts of *severe injuries.*

In SSIS, we joined the *ClaimFactTable* (for *crash_id*, *location_id*, *date_id*) with the *DateDimension* table (for *date_id*, and *is_holiday*) with a left outer join. We then left-outer-joined it with the *GeographyDimension* table to get the *Area Risk Level* identification for each crash. Then we split the data conditionally based on whether the *is_holiday* value is *true* or *false.* After the split, for each compartment, we group by *Area Risk Level* and proceed to count the number of crashes in those areas. Finally, we export the file with the *Area Risk Level* and the number of crashes over *holiday* periods and over *non-holiday* periods.

This observation helps us understand whether *Area Risk Level* correlates with the number of crashes, and the proportion of *holiday crashes* vs. *non-holiday crashes.* In future iterations, we would like to study these accident phenomena as a ratio between *holiday crashes* and *non-holiday crashes*, and over other interesting spatial features such as unique streets in Chicago.