Multidimensional Design for the ACME Flying Data Warehouse

1. Deconstruction of Analytical Requirements and Key Metrics

1.1. The Business Imperative: Analyzing Operational Efficiency

The central objective of this project is to architect a Data Warehouse (DW) that empowers ACME Flying to transition from rudimentary operational reporting to sophisticated descriptive analytics.[1] The company's current data landscape is characterized by heterogeneous, siloed systems, including the Air Information Management System (AIMS) for flight operations and the Aircraft Maintenance Operation System (AMOS) for maintenance, repair, and overhaul (MRO) activities.[1] This fragmentation makes it exceedingly difficult to obtain a holistic, integrated view of aircraft performance, utilization, and reliability.

The proposed DW will serve as the single source of truth, enabling analysts and decision-makers to explore complex relationships between flight schedules, delays, cancellations, maintenance events, and aircraft availability. By integrating data from these disparate sources, the DW will facilitate the calculation and analysis of Key Performance Indicators (KPIs) that are critical for optimizing fleet management, improving service reliability, and controlling maintenance costs. The ultimate goal is to provide a robust analytical platform that supports strategic decision-making through Online Analytical Processing (OLAP) capabilities.[1]

1.2. KPI Specification and Data Source Mapping

A meticulous analysis of the project requirements reveals a set of 18 distinct KPIs that the multidimensional design must support.[1] These KPIs fall into three primary categories: Aircraft Utilization, Reliability & Performance, and Logbook Reporting. The design of the data warehouse, particularly the choice of fact tables and their grains, is directly driven by the need to efficiently compute these metrics.

A fundamental challenge identified at this stage is the inherent mismatch in the temporal granularity required for these KPIs. Metrics such as Flight Hours (FH) and Flight Cycles (TO) are naturally measured at a daily level, derived from individual flight records. In contrast, metrics like Aircraft Days Out-of-Service (ADOS) and various performance rates are defined as monthly or yearly aggregates.[1] Attempting to accommodate these different grains within a single fact table would lead to a fundamentally flawed design, characterized by data redundancy, aggregation errors, and an inability to satisfy all user queries. For instance, storing monthly ADOS values at a daily grain would involve repeating the same figure for every day of the month, while pre-aggregating daily flight data to a monthly level would make it impossible to perform daily analysis. This necessitates a multi-fact table architecture, a core principle that will guide the subsequent design.

The following table provides a comprehensive matrix that defines each KPI, specifies its calculation formula as per the source documentation, identifies the required data elements and their source systems, and anticipates the target fact table in the proposed DW architecture. This matrix serves as the definitive contract between the business requirements and the technical design.

Table 1: KPI Requirements Matrix

KPI Name	Business Definition	Calculation Formula	Required Granularity	Source System(s) & Table(s)	Target Fact Table
Aircraft Utilization KPIs					
Flight Hours (FH)	Airborne time from wheels-off to wheels-on.[1]	SUM(actualArrival - actualDeparture)	Daily	AIMS flights	Fact_Flight_Operations_Daily

Flight Cycles (TO)	Number of take-offs.[1]	COUNT(non-cancelled flights)	Daily	AIMS flights	Fact_Flight_Operations_Daily
Aircraft Days In-Service (ADIS)	Days an aircraft was available for operation.[1]	Days in Period - ADOS	Monthly / Yearly	AMOS MaintenanceEvents	Fact_Aircraft_Monthly_Snapshot
Aircraft Days Out-of-Service (ADOS)	Days an aircraft was unavailable due to maintenance.[1]	COUNT(unique days in maintenance intervals)	Monthly / Yearly	AMOS MaintenanceEvents	Fact_Aircraft_Monthly_Snapshot
ADOS Scheduled (ADOSS)	ADOS due to scheduled maintenance (Maintenance/Revision).[1]	COUNT(unique days in scheduled maintenance)	Monthly / Yearly	AMOS MaintenanceEvents	Fact_Aircraft_Monthly_Snapshot
ADOS Unscheduled (ADOSU)	ADOS due to unscheduled maintenance (Delay/AOG).[1]	COUNT(unique days in unscheduled maintenance)	Monthly / Yearly	AMOS MaintenanceEvents	Fact_Aircraft_Monthly_Snapshot
Daily Utilization (DU)	Average flight hours per day in service.[1]	$FH / ADIS$	Monthly / Yearly	AIMS, AMOS	Calculated in BI Layer
Daily	Average take-offs	$TO /	Monthly /	AIMS,	Calculated

Cycles (DC)	per day in service.[1]	ADIS$	Yearly	AMOS	in BI Layer
Reliability & Performance KPIs					
Delay Rate (DYR)	Delays per 100 departures.[1]	$(DY / TO) * 100$	Monthly / Yearly	AIMS flights	Calculated in BI Layer
Cancellation Rate (CNR)	Cancellations per 100 departures.[1]	$(CN / TO) * 100$	Monthly / Yearly	AIMS flights	Calculated in BI Layer
Technical Dispatch Reliability (TDR)	Percentage of on-time departures.[1]	$100 - ((DY + CN) / TO) * 100$	Monthly / Yearly	AIMS flights	Calculated in BI Layer
Average Delay Duration (ADD)	Average minutes of delay per 100 departures.[1]	(SUM(delay duration) / DY) * 100	Monthly / Yearly	AIMS flights	Calculated in BI Layer
Logbook Reporting KPIs					
Report Rate per hour (RRh)	Logbook entries per 1000 flight hours.[1]	$1000 * (logbook count) / FH$	Monthly / Yearly	AMOS, AIMS	Calculated in BI Layer
Report Rate per cycle	Logbook entries per	$100 * (logbook	Monthly / Yearly	AMOS, AIMS	Calculated in BI Layer

(RRc)	100 take-offs.[1]	count) / TO$			
PIREP Rate per hour (PRRh)	Pilot logbook entries per 1000 flight hours.[1]	$1000 * (Pilot logbook count) / FH$	Monthly / Yearly	AMOS, AIMS	Calculated in BI Layer
PIREP Rate per cycle (PRRc)	Pilot logbook entries per 100 take-offs.[1]	$100 * (Pilot logbook count) / TO$	Monthly / Yearly	AMOS, AIMS	Calculated in BI Layer
MAREP Rate per hour (MRRh)	Maintenance logbook entries per 1000 flight hours.[1]	$1000 * (Maint. logbook count) / FH$	Monthly / Yearly	AMOS, AIMS	Calculated in BI Layer
MAREP Rate per cycle (MRRc)	Maintenance logbook entries per 100 take-offs.[1]	$100 * (Maint. logbook count) / TO$	Monthly / Yearly	AMOS, AIMS	Calculated in BI Layer

1.3. Query Requirement Analysis

The project statement explicitly defines four analytical queries that the DW must support, which further clarifies the required dimensions and granularities.[1]

a) Give me FH and TO per aircraft (also per model and manufacturer) per day (also per month and per year).
This query confirms the need for a daily grain fact table. It also establishes the requirement for an Aircraft dimension with a Aircraft -> Model -> Manufacturer hierarchy and a Date dimension with a Day -> Month -> Year hierarchy.
b) Give me ADIS, ADOS, ADOSS, ADOSU, DYR, CNR, TDR, ADD per aircraft (also per model and

manufacturer) per month (also per year).
This query confirms the need for a monthly grain fact table to store the ADOS-related metrics. The other rate-based KPIs (DYR, CNR, etc.) will be calculated using atomic counts from the daily and monthly fact tables. It reinforces the need for the conformed Aircraft and Date dimensions.
c) Give me the RRh, RRC, PRRh, PRRC, MRRh and MRRc per aircraft (also per model and manufacturer) per month (also per year).
This query requires the counts of different logbook report types at a monthly grain, which can be housed in the same monthly snapshot fact table as the ADOS metrics.
d) Give me the MRRh and MRRc per airport of the reporting person per aircraft (also per model and manufacturer).
This query introduces a subtle but critical dimensional requirement. It demands analysis not by the airport where an event occurred, but by the home base of the maintenance personnel who filed the report. This cannot be satisfied by a standard flight-related airport dimension. It mandates the creation of a dedicated dimension for the reporter, which must contain an attribute for their assigned airport. This information will be sourced from the maintenance_personnel.csv file and linked to the reporter ID from the AMOS database.1 This requirement directly translates a specific business question into a concrete element of the schema, demonstrating a requirement-driven design process.

2. The Data Warehouse Bus Architecture for ACME Flying

2.1. Principles of the Bus Architecture

To address the analytical requirements of ACME Flying, this design adopts the Data Warehouse Bus Architecture, a methodology championed by Ralph Kimball. This approach organizes the data warehouse as a collection of interconnected, process-centric star schemas. Each star schema models a specific business process (e.g., flight operations, aircraft maintenance) and consists of a central fact table connected to a set of shared, standardized dimension tables.

The "bus" itself is the collection of these shared dimensions, known as conformed dimensions. By ensuring that different fact tables use the exact same dimension tables for common entities like Date, Aircraft, and Airport, the architecture guarantees analytical consistency and enables powerful cross-process analysis. For example, an analyst can seamlessly query flight cycles from a daily operations fact table and maintenance days from a monthly snapshot fact table, and slice and dice both sets of metrics by the same aircraft model or manufacturer. This modular, scalable, and integrated approach is ideal for an enterprise like ACME Flying, as it allows the data warehouse to grow incrementally, adding new star schemas for new business processes over time without disrupting the existing structure.

2.2. Justification: Why a Single Star Schema is Insufficient

As established in the analysis of KPIs and query requirements, a single star schema is an inadequate solution for this project. The primary reason is the problem of mixed granularity.[1] The requirements demand analysis of some metrics at a daily grain (FH, TO) and others at a monthly or yearly grain (ADOS, Logbook Rates).

Attempting to force these disparate grains into one fact table would lead to one of two undesirable outcomes:

1. **Adopting the Finest Grain (Daily):** If the fact table grain were set to "per aircraft per day," measures like monthly ADOS would have to be repeated for every day of the month for a given aircraft. This would lead to massive data redundancy and create a significant risk of incorrect aggregation by end-users, who might erroneously sum the repeated monthly value across all days.
2. **Adopting the Coarsest Grain (Monthly):** If the fact table grain were set to "per aircraft per month," all daily flight data would need to be pre-aggregated. This would make it impossible to fulfill query a, which explicitly requires the ability to analyze Flight Hours and Flight Cycles on a daily basis.[1]

Therefore, a multi-fact table approach is not merely a design preference but a technical necessity to accurately and efficiently meet all stated analytical objectives.

2.3. Proposed Architecture for ACME Flying

The proposed architecture is a constellation schema consisting of two distinct fact tables that share a common set of conformed dimensions. This design directly resolves the mixed-grain challenge while providing a comprehensive view of aircraft operations and maintenance.

The two core components are:

- **Fact_Flight_Operations_Daily:** This is a transactional-grain fact table designed to capture the aggregated results of an aircraft's flight activities. Its grain is "one row per aircraft per day." It will contain fundamental, additive measures like Flight_Hours and Flight_Cycles, which form the basis for many of the required KPIs.
- **Fact_Aircraft_Monthly_Snapshot:** This is a periodic snapshot fact table that records the status and summary metrics for each aircraft at the end of every month. Its grain is "one row per aircraft per month." It will house semi-additive measures like Aircraft_Days_Out_Of_Service and additive counts of maintenance logbook entries.

The power of this architecture lies in the conformed dimensions that bridge these two fact tables. Both Fact_Flight_Operations_Daily and Fact_Aircraft_Monthly_Snapshot will link to the same Dim_Date and Dim_Aircraft tables. This shared dimensional backbone is the "bus" that enables integrated analysis. A user can analyze daily utilization from the first fact table and monthly reliability from the second, slicing both by the same aircraft model or manufacturer, thereby creating a holistic and consistent view of fleet performance.

3. Core Fact Table Design: The Flight Operations Snapshot

This fact table is the cornerstone for analyzing the day-to-day operational performance of the ACME Flying fleet. It is designed to provide a granular view of flight activity, directly supporting the calculation of utilization metrics.

3.1. Business Process and Grain Definition

- **Business Process:** Daily Flight Operations. This process encompasses all revenue and non-revenue flights conducted by the airline's fleet.
- **Grain:** The grain of this fact table is defined with absolute precision as **one row per aircraft per day**. This means that all individual flights for a specific aircraft (identified by its AircraftRegistration) that have a scheduledDeparture on a given calendar date will be aggregated into a single row in this table. This choice of grain directly satisfies the requirements of query a.[1] The aggregation logic is simplified and standardized by Business Rule BR-20, which mandates that "All the hours of a Flight are imputed to the date of its scheduledDeparture".[1]

3.2. Dimensionality

The Fact_Flight_Operations_Daily table will be linked to the following conformed dimension tables via foreign key relationships:

- Date_Key: A foreign key referencing the primary key of the Dim_Date table. This link provides the temporal context for the facts.
- Aircraft_Key: A foreign key referencing the primary key of the Dim_Aircraft table. This link provides the context of which specific aircraft, model, and manufacturer the facts relate to.

3.3. Fact and Measure Specification

The table will contain a set of fully additive numeric measures. Additivity is a critical property in data warehousing, as it ensures that measures can be correctly summed across all dimensions. For example, Flight_Hours can be summed across all days in a month to get total monthly flight hours, or across all aircraft of a certain model to get the total flight hours for that model.

The measures are:

- **Flight_Hours**: The total airborne time for the aircraft on that day. Calculated as the sum of (actualArrival - actualDeparture) for all non-cancelled flights from the AIMS flights table.[1]
- **Flight_Cycles**: The total number of take-offs for the aircraft on that day. Calculated as the count of all non-cancelled flights from the AIMS flights table.[1]
- **Delayed_Flights_Count**: The total number of flights for the aircraft on that day that incurred a delay (i.e., where delayCode is not null in the AIMS flights table).[1] This serves as the numerator for the
  DYR KPI.
- **Cancelled_Flights_Count**: The total number of flights for the aircraft on that day that were cancelled (i.e., where cancelled is true in the AIMS flights table).[1] This serves as the numerator for the
  CNR KPI.
- **Total_Delay_Minutes**: The sum of the duration, in minutes, of all delays for the aircraft on that day. This is a crucial component for calculating the ADD KPI.
- **Passenger_Count**: The total number of passengers carried by the aircraft on that day.

Sourced from the passengers column in the AIMS flights table.[1]

Table 2: Fact_Flight_Operations_Daily Schema Specification

Column Name	Data Type	Role	Description
Date_Key	INTEGER	Foreign Key (to Dim_Date)	Links to the date of operations.
Aircraft_Key	INTEGER	Foreign Key (to Dim_Aircraft)	Links to the aircraft being measured.
Flight_Hours	DECIMAL(10, 2)	Measure	Total airborne time in hours.
Flight_Cycles	INTEGER	Measure	Total number of take-offs.
Delayed_Flights_Count	INTEGER	Measure	Count of flights with a delay code.
Cancelled_Flights_Count	INTEGER	Measure	Count of cancelled flights.
Total_Delay_Minutes	INTEGER	Measure	Sum of all delay durations in minutes.
Passenger_Count	INTEGER	Measure	Total number of passengers transported.

4. Ancillary Fact Table Design: Maintenance and Logbook Analysis

This second fact table is designed as a periodic snapshot to track longer-term trends in aircraft availability, maintenance status, and reliability as reflected in logbook reporting.

4.1. Business Process and Grain Definition

- **Business Process:** Monthly Aircraft Status and Maintenance Reporting. This process involves summarizing all maintenance activities and logbook entries over a calendar month to assess an aircraft's health and reliability.
- **Grain:** The grain is **one row per aircraft per month**. Each row represents a summary of an aircraft's status and reporting activity for a given month, capturing the state of affairs at the end of that period. This grain is chosen to directly support queries b and c.[1]

4.2. Dimensionality

This fact table will connect to the same conformed dimensions as the daily fact table, ensuring architectural consistency and enabling integrated analysis:

- Month_Key: A foreign key referencing the Dim_Date table. This key will link to the first day of the month (or a dedicated month-level key) to represent the entire period.
- Aircraft_Key: A foreign key referencing the primary key of the Dim_Aircraft table.

4.3. Fact and Measure Specification

This table contains a mix of semi-additive and fully additive measures.

Semi-Additive Measures:
These are measures that can be summed across some dimensions but not others. In this case, the ADOS metrics are additive across the Aircraft dimension (one can sum the ADOS for all aircraft in a fleet) but are not additive across the Date dimension (summing January's ADOS and February's ADOS does not produce a meaningful value).

- **Aircraft_Days_In_Service (ADIS)**: The total number of days in the month that the aircraft was available for operations. This is a calculated measure, derived as (Total Days in Month) - ADOS.[1]
- **Aircraft_Days_Out_Of_Service (ADOS)**: The total number of unique days in the month that the aircraft was unavailable due to any maintenance event. The calculation of this

measure is a non-trivial ETL task. It requires identifying all maintenance events for an aircraft from the AMOS MaintenanceEvents table, capturing their start and end timestamps, and then calculating the union of all these date ranges to find the total number of unique calendar days affected. A naive summation of event durations would lead to incorrect results if maintenance events overlap. For example, a scheduled Revision from Jan 10-25 and an unscheduled AOG event from Jan 22-28 must be resolved into a single continuous out-of-service period from Jan 10-28 (19 days), not the sum of their individual durations (15 + 6 = 21 days).

- **ADOS_Scheduled (ADOSS)**: A subset of ADOS, counting only the unique days the aircraft was unavailable due to scheduled events, specifically those of kind Maintenance or Revision.[1]
- **ADOS_Unscheduled (ADOSU)**: A subset of ADOS, counting only the unique days the aircraft was unavailable due to unscheduled events, such as Delays or Aircraft On Ground (AOG).[1]

Fully Additive Measures:

- **Logbook_Entries_Count**: The total count of TechnicalLogBookOrders filed for the aircraft during the month. Sourced from the AMOS database.[1]
- **PIREP_Count**: The count of logbook entries reported by pilots (ReporteurClass = 'PIREP' per BR-8).[1]
- **MAREP_Count**: The count of logbook entries reported by maintenance personnel (ReporteurClass = 'MAREP' per BR-8).[1]

Table 3: Fact_Aircraft_Monthly_Snapshot Schema Specification

Column Name	Data Type	Role	Description
Month_Key	INTEGER	Foreign Key (to Dim_Date)	Links to the month of the snapshot.
Aircraft_Key	INTEGER	Foreign Key (to Dim_Aircraft)	Links to the aircraft being measured.
ADIS	INTEGER	Measure (Semi-Additive)	Count of days in-service for the month.
ADOS	INTEGER	Measure (Semi-Additive)	Count of unique days out-of-service for the month.

ADOSS	INTEGER	Measure (Semi-Additive)	Count of unique scheduled out-of-service days.
ADOSU	INTEGER	Measure (Semi-Additive)	Count of unique unscheduled out-of-service days.
Logbook_Entries_Count	INTEGER	Measure	Total count of logbook entries for the month.
PIREP_Count	INTEGER	Measure	Count of pilot-reported logbook entries.
MAREP_Count	INTEGER	Measure	Count of maintenance-reported logbook entries.

5. Conformed Dimensional Models: The Context for Analysis

Dimension tables provide the descriptive context for the numeric facts. They answer the "who, what, where, when, and why" questions behind the business processes. The use of conformed dimensions is the cornerstone of the bus architecture, ensuring that analytical results are consistent and comparable across different fact tables.

5.1. Dim_Date

This is a standard, pre-populated dimension table that provides a rich set of attributes for temporal analysis.

- **Grain:** One row per calendar day.
- **Hierarchies:** The primary hierarchy is Day -> Month -> Quarter -> Year, which enables users to effortlessly roll up daily facts to monthly, quarterly, or yearly summaries.
- **Attributes:** The table will contain attributes such as Date_Key (PK), Full_Date, Day_Of_Week, Day_Name, Day_Of_Month, Month_Number, Month_Name, Quarter, Year, and flags like Is_Weekday or Is_Holiday.

5.2. Dim_Aircraft

This dimension provides the definitive list of all aircraft in the fleet and their key characteristics.

- **Grain:** One row per unique aircraft.
- **Hierarchies:** The primary hierarchy is Aircraft -> Model -> Manufacturer, allowing for analysis at different levels of fleet aggregation as required by the queries.[1]
- **Data Sources:** This is a composite dimension. The Aircraft_Registration_Code (the natural key) and Manufacturer_Serial_Number will be sourced from the operational AIMS and AMOS systems. This data will then be enriched by performing a lookup to the external aircraft-manufacturerinfo-lookup.csv file to populate the Aircraft_Model and Aircraft_Manufacturer attributes.[1]

Table 4: Dim_Aircraft Attribute Specification

Attribute Name	Business Description	Source System/File	Source Column	Data Type
Aircraft_Key	Surrogate Primary Key	DW	(Generated)	INTEGER
Aircraft_Registration_Code	Unique registration code of the aircraft (e.g., EC-MGF).	AIMS/AMOS	aircraftregistration	VARCHAR(10)
Manufacturer_	Serial number	PFR/Airbus	MSN	VARCHAR(20)

Serial_Number	assigned by the manufacturer.			
Aircraft_Model	The model of the aircraft (e.g., A320-200).	aircraft-manufacturerinfo-lookup.csv	model	VARCHAR(50)
Aircraft_Manufacturer	The manufacturer of the aircraft (e.g., Airbus).	aircraft-manufacturerinfo-lookup.csv	manufacturer	VARCHAR(50)

5.3. Dim_Airport

While not a primary dimension for the specified queries, a standard airport dimension is an essential component for any aviation data warehouse, providing geographical context.

- **Grain:** One row per unique airport.
- **Attributes:** Airport_Key (PK), Airport_Code (IATA 3-letter code), Airport_Name, City, Country. This dimension will be populated from the distinct departure and arrival airport codes found in the AIMS flights table and the executionplace from AMOS WorkOrders.[1]

5.4. Dim_Reporter

This specialized dimension is created specifically to satisfy the unique requirement of query d, which asks for analysis based on the location of the person filing a maintenance report.[1]

- **Grain:** One row per unique person who files a technical logbook report.
- **Data Sources:** This dimension is populated from the TechnicalLogBookOrders table in AMOS, which contains a reporteurid.[1] The
  Reporter_Type is derived based on the ReporteurClass attribute, as defined in BR-8 ('PIREP' for Pilot, 'MAREP' for Maintenance).[1] The crucial
  Assigned_Airport_Code attribute is populated by performing a lookup on the reporteurid

in the external maintenance_personnel.csv file.[1] This design enables analysts to aggregate
MAREP_Count from the monthly fact table by the maintenance base of the personnel, providing insights into which locations are reporting the most issues.

Table 5: Dim_Reporter Attribute Specification

Attribute Name	Business Description	Source System/File	Source Column	Data Type
Reporter_Key	Surrogate Primary Key	DW	(Generated)	INTEGER
Reporter_ID	Unique identifier of the reporting person.	AMOS TechnicalLogBookOrders	reporteurid	VARCHAR(20)
Reporter_Type	The role of the reporter.	AMOS TechnicalLogBookOrders	reporteurclass	VARCHAR(20)
Assigned_Airport_Code	The home base airport of the maintenance person.	maintenance_personnel.csv	airport_code	VARCHAR(3)

6. Strategic Implementation Guidance and ETL Considerations

A robust multidimensional design is only as effective as the Extract-Transform-Load (ETL) process that populates it. This section provides strategic guidance for developing an ETL process that ensures data quality, integrity, and performance.

6.1. High-Level ETL Flow

The ETL process will follow a standard, staged approach. Data will be extracted from the source systems (AIMS and AMOS databases, and the supplementary CSV files) into a staging area. In this area, all transformations, data cleansing, integration, and business rule enforcement will occur. Once the data is conformed to the target dimensional model, it will be loaded into the final dimension and fact tables in the DuckDB data warehouse. The general flow will be:

1. **Extract:** Pull raw data from AIMS, AMOS, and CSV files.
2. **Transform (Dimensions):** Cleanse, standardize, and integrate data to build the Dim_Date, Dim_Aircraft, Dim_Airport, and Dim_Reporter tables. This includes performing lookups to enrich aircraft and reporter data.
3. **Load (Dimensions):** Populate the dimension tables. Surrogate keys will be generated during this step.
4. **Transform (Facts):** Aggregate transactional data from AIMS and AMOS to the required grains (daily for flights, monthly for maintenance). This stage involves complex calculations (e.g., for ADOS) and the replacement of natural keys (like Aircraft_Registration_Code) with the newly generated surrogate keys from the dimension tables.
5. **Load (Facts):** Populate the Fact_Flight_Operations_Daily and Fact_Aircraft_Monthly_Snapshot tables.

6.2. Addressing Data Quality Mandates

The project statement mandates the correction of three specific data quality issues, which must be handled during the transformation stage.[1]

- **BR-23 Fix (Invalid Flight Times):** The rule states that actualArrival must be posterior to actualDeparture. The ETL logic must include a conditional check for each flight record. If actualArrival <= actualDeparture, the values must be swapped before calculating Flight_Hours.
- **BR-21 Fix (Overlapping Flights):** The rule prohibits two non-cancelled flights of the same aircraft from overlapping in time. To enforce this, the ETL process must be stateful. For each aircraft, the flight records should be processed in chronological order based on scheduledDeparture. The ETL logic must maintain the actualArrival time of the last valid flight processed for that aircraft. If the actualDeparture of the current flight is before the stored actualArrival of the previous flight, it constitutes an overlap. As per the requirement, this overlapping flight record must be ignored and its details written to a log

file for later review.

- **Aircraft Registration Validation:** Before processing a post-flight report from AMOS, the AircraftRegistration code in the report must be validated. This can be efficiently handled by performing a lookup against the already-populated Dim_Aircraft table. If the registration code does not exist in the dimension, the report is considered invalid and must be ignored, with its details logged.

6.3. Handling Slowly Changing Dimensions (SCDs)

Dimension attributes can change over time (e.g., an aircraft could be repainted or reconfigured). The strategy for handling these changes is known as a Slowly Changing Dimension (SCD) policy. For the scope of this project, a **Type 1 SCD** approach is sufficient and recommended for its simplicity. In a Type 1 approach, when an attribute changes in the source system, the corresponding record in the dimension table is simply overwritten with the new value. This approach does not preserve history but is often adequate for attributes that do not change frequently or where historical tracking is not a business requirement. For future enhancements, a Type 2 SCD approach—which preserves history by creating a new dimension record with a new surrogate key—could be considered if historical analysis of dimensional attributes becomes necessary.

6.4. Calculation of Derived KPIs

A critical aspect of this design is the distinction between measures stored in the fact tables and KPIs calculated in the presentation layer (e.g., a BI tool, reporting software, or a SQL query). The fact tables are designed to store fundamental, additive, atomic components (numerators and denominators). Complex, ratio-based KPIs like Delay Rate (DYR), Daily Utilization (DU), or Report Rate per hour (RRh) are **not** stored in the data warehouse.

This is a deliberate and crucial design choice based on best practices. Ratios and percentages are non-additive. For example, one cannot sum the DYR for January and the DYR for February to obtain the correct DYR for the first quarter. To calculate the quarterly DYR, one must sum the Delayed_Flights_Count for both months and the Flight_Cycles for both months, and *then* perform the division.

By storing the atomic counts (Delayed_Flights_Count, Cancelled_Flights_Count, Flight_Cycles, Logbook_Entries_Count) and sums (Flight_Hours, Total_Delay_Minutes) in the fact tables, this design empowers the BI tool or semantic layer to correctly calculate these KPIs at any level of aggregation (daily, monthly, quarterly, yearly, per model, per manufacturer) without error. This separation of concerns ensures analytical flexibility, accuracy, and prevents the storage of redundant, non-additive data in the warehouse.

Obras citadas

1. Statement.pdf