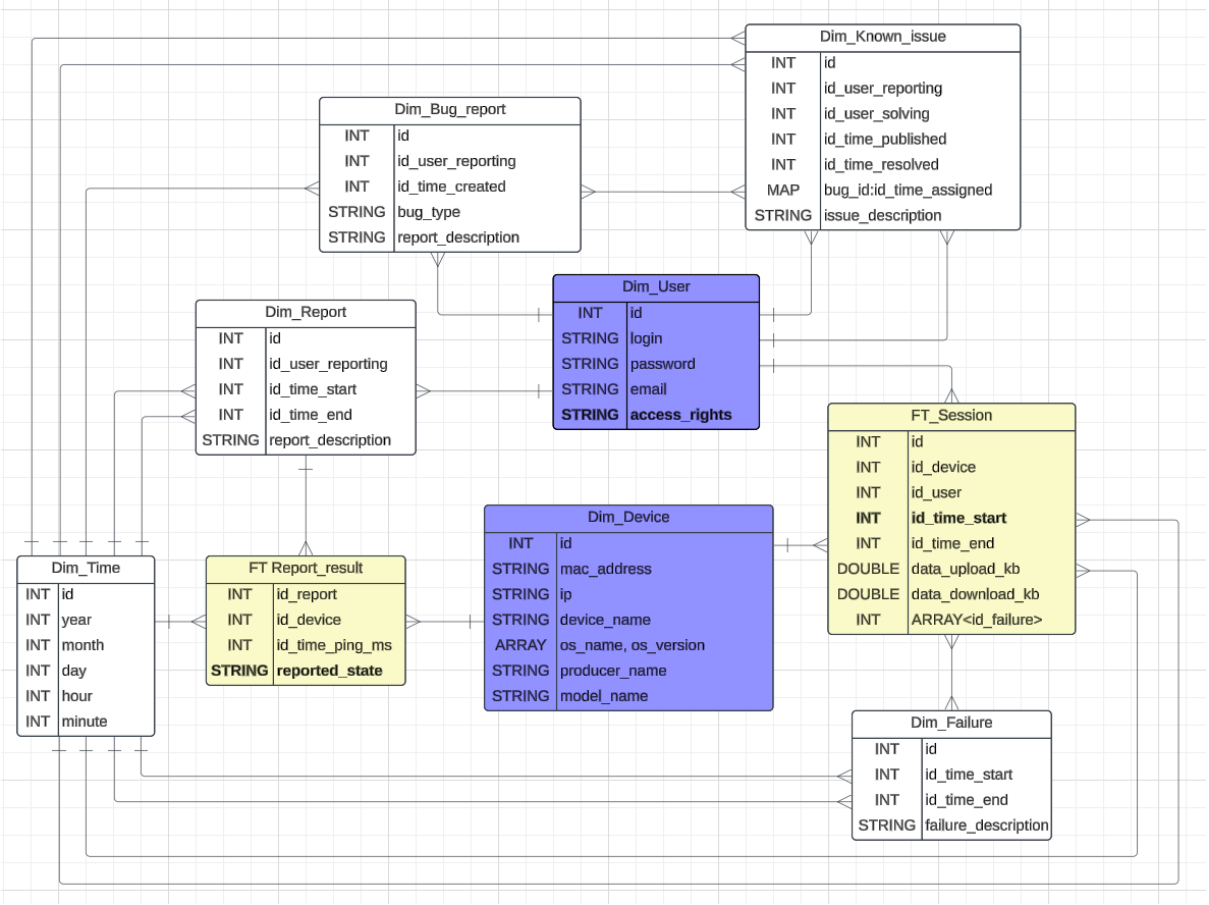
Network management data warehouse design in Apache/Hive



1. File storing types:

Dim_User and **Dim_Device** are stored as text files, for easier modification by other systems (e.g. adding, removing and modifying users or devices).

Dim_Time, Dim_Report, Dim_Bug_report, **Dim_Known_issue**, and **Dim_Failure** are stored as PARQUET because it stores data in a way that takes up less space and allows for faster searches.

FT_Report_result and **FT_Session** are stored as AVRO because it is also good for handling data that might change over time. AVRO helps to keep the data small and easy to store.

2. Internal and external tables:

Dim_User and **Dim_Device** are the only tables stored externally, because they are mostly maintained by external systems that might interfere.

Remaining tables are internal for better performance and ease of use.

3. Partitioning and bucketing

The table **Dim_User** is statically partitioned by **access_rights**, so that there are separate files for users with different permissions (admins, moderators, users and guests). This allows us to restrict access using the `hdfs dfs -chmod` command.

Table **FT_Report_result** is dynamically partitioned by **reported_state** for better querying performance, and splitting the large fact table into smaller, more manageable files..

Table **FT_Session** uses bucketing on id_time_start column, to improve performance, and it split up.

4. Complex data types

Dim_Known_issue.associated_bugs - MAP<INT,INT> is used to match related bugs, and the time at which they were associated. Each known issue might be related to one or more bugs. To solve this problem, ARRAY<STRUCT<INT,INT>> also can be used, but MAP has simpler structure and easier querries.

FT_Session.failures - ARRAY<INT> functions as a many-to-many relationship workaround, storing zero, one or multiple id_failure. It allows to list all failures related to a specific session, without creating an additional table.

Dim_Device.os_versions - ARRAY<STRUCT<os_name: STRING, os_version: STRING>> allows to describe devices using dual boot, that can run different operating systems.

5. Competency questions

1. Which devices have potential performance issues (increased ping time)?

```
SELECT
  d.device_name,
  d.ip,
  AVG(rr.time_ping_ms) as avg_ping_ms
FROM FT_Report_result rr
JOIN Dim_Device d ON rr.id_device = d.id
GROUP BY d.device_name, d.ip
HAVING avg_ping_ms > 100
ORDER BY avg_ping_ms DESC;
```

2. What is the distribution of devices by operating system version?

```
SELECT os_version.os_name, os_version.os_version, COUNT(DISTINCT d.id)
AS device_count
FROM Dim_Device d
LATERAL VIEW EXPLODE(d.os_versions) exploded_os_version AS os_version
GROUP BY os_version.os_name, os_version.os_version;
```

3. Who are the top 5 users who reported the most bugs?

```
SELECT
  u.login,
  COUNT(*) as bug_count
FROM Dim_Bug_report br
JOIN Dim_User u ON br.id_user_reporting = u.id
GROUP BY u.login
ORDER BY bug_count DESC
LIMIT 5;
```

4. How many failures occurred in 2022?

```
SELECT COUNT(*) AS failure_count
FROM Dim_Failure f
JOIN Dim_Time t ON f.id_time_start = t.id
WHERE t.year = 2022
  AND (
      (t.month = 1 AND t.day >= 1)
      OR (t.month > 1 AND t.month < 12)
      OR (t.month = 12 AND t.day <= 31)
  );
```

5. What are the average data upload and download volumes per session by users?

```
SELECT id_user, AVG(data_upload_kb) AS avg_upload_kb,
AVG(data_download_kb) AS avg_download_kb
FROM FT_Session
GROUP BY id_user;
```

6. What is the average response time of reports by device?

```
SELECT id_device, AVG(time_ping_ms) AS avg_ping_ms
FROM FT_Report_result
GROUP BY id_device;
```

7. Which devices reported the most failures during their sessions?

```
SELECT id_device, COUNT(DISTINCT failure_id) AS failure_count
FROM FT_Session f
LATERAL VIEW EXPLODE(failures) exploded_failures AS failure_id
GROUP BY id_device
ORDER BY failure_count DESC;
```

8. Which bug types are the most common?

```
SELECT bug_type, COUNT(*) AS bug_count
FROM Dim_Bug_report
GROUP BY bug_type
ORDER BY bug_count DESC;
```

9. What is the average duration of issues from publication to resolution?

```
WITH issue_times AS (
  SELECT
    ki.id,
    pub.year AS pub_year,
    pub.month AS pub_month,
    pub.day AS pub_day,
    pub.hour AS pub_hour,
    pub.minute AS pub_minute,
    res.year AS res_year,
    res.month AS res_month,
    res.day AS res_day,
    res.hour AS res_hour,
    res.minute AS res_minute
  FROM Dim_Known_issue ki
```

```
  JOIN Dim_Time pub ON ki.id_time_published = pub.id
  JOIN Dim_Time res ON ki.id_time_resolved = res.id
),
duration_calc AS (
  SELECT
    FLOOR(
      AVG(
        (res_year - pub_year) * 525600 + -- minutes in a year
        (res_month - pub_month) * 43800 + -- average minutes in a month
        (res_day - pub_day) * 1440 + -- minutes in a day
        (res_hour - pub_hour) * 60 + -- minutes in an hour
        (res_minute - pub_minute) -- minutes
      )
    ) AS total_minutes
  FROM issue_times
)
SELECT
  CONCAT(
    CAST(FLOOR(total_minutes / 1440) AS STRING), ' days, ',
    CAST(FLOOR((total_minutes % 1440) / 60) AS STRING), ' hours, ',
    CAST(total_minutes % 60 AS STRING), ' minutes'
  ) AS average_resolution_time
FROM duration_calc;
```

10. Which bugs often appear together in the known issues?

```
SELECT
    k1.key as bug_id1,
    k2.key as bug_id2,
    COUNT(*) as times_appeared_together
FROM Dim_Known_issue ki
LATERAL VIEW EXPLODE(ki.associated_bugs) k1
LATERAL VIEW EXPLODE(ki.associated_bugs) k2
WHERE k1.key < k2.key -- Avoid self-pairs and duplicates
GROUP BY k1.key, k2.key
HAVING times_appeared_together > 1
ORDER BY times_appeared_together DESC;
```