ASSIGNMENT 12.1

Explore Data Warehouses

Shubha, Adithya

2024-11-12

Question 1

Data warehouses are often constructed using relational databases. Explain the use of fact tables and star schemas to construct a data warehouse in a relational database. Also comment on whether a transactional database can and should be used to house an OLAP. Lastly, why would an organization use a relational database with a star schema rather than a dedicated NoSQL database specifically engineered for data warehousing and analytics?

In a data warehouse built on a relational database, data is typically structured as multidimensional, with two main types of attributes: **fact** attributes and **dimension** attributes. Fact attributes represent quantitative values that can be aggregated and are often referred to as facts. In contrast, dimension attributes provide the context by which the measure attributes and their summaries are analyzed.

Facts are stored in dedicated tables called fact tables, while dimension attributes are kept in separate tables known as dimension tables or lookup tables. The simplest schema for relational data warehouses is the star schema, which organizes data with a central fact table surrounded by individual dimension tables. A data warehouse may consist of multiple star schemas.

Fact tables are usually large, and their data is pre-computed during the ETL process. This pre-computation ensures faster lookups, as querying typically involves retrieving specific columns for a small number of rows. By eliminating the need for on-the-fly calculations, pre-computation enables efficient data exploration and quick access to facts in dashboards and analytics tools. However, the facts must be periodically recalculated when the data warehouse is refreshed with updated operational data.

Using transaction database to house an OLAP While transactional database can be used to house an OLAP it is generally **not** recommended for the following reasons. 1. Transactional databases are usually normalized to reduce redundancy and improve data integrity which makes analytical queires inefficient due to the need for joins. 2. Transactional databases are optimized for write operations and not for read-only operations which are common in OLAP. 3. Transactional databases are optimized for real-time processing and not for batch processing which is common in OLAP.

Using star schema instead of NoSQL database

Some organisations prefer to use Star Schema instead of NoSQL because of the following reasons: 1. Star Schema is build on relational databases which have been around for a long time and lot of tools such as BI tools available in the market making it easier for organisatons to use them. 2. The final data formed is structured and is easily maintainable than the semi-strucutred data in NoSQL databases. 3. Many complex queries of aggregations and joins can be easily executed on a star schema as compared to NoSQL databases.

Question 2

Explain the difference between a data warehouse, a data mart, and a data lake. Provide at least one example of their use from your experience or find out how they are generally used in

practice. Find at least one video, article, or tutorial online that explains the differences and embed that into your notebook.

Data Warehouse A data warehouse is a centralized and integrated repository that consolidates data from multiple sources. It is primarily designed for querying and analysis rather than transaction processing. Key characteristics of a data warehouse include:

a. Data remains stable and is not frequently updated or deleted.
b. It retains data over extended periods, enabling the analysis of long-term trends.
c. It offers users a single, consolidated interface to access data, supporting decision-making processes.

For example, banks utilize data warehouses to analyze the spending patterns of account holders or cardholders. This information helps them tailor special offers, discounts, and deals to specific customer behaviors.

Data Mart: Data warehouses can grow significantly in size and often include data that isn't relevant for every use case. To address specific needs, organizations extract targeted subsets of the data warehouse, known as **data marts**. Data marts are focused on particular subject areas and are designed to meet the requirements of specific users or teams.

For instance, a retail business might create separate data marts for sales and inventory. The sales data mart would include information related to sales transactions, while the inventory data mart would concentrate on product stock levels and supply chain details.

Data Lakes A data lake is a centralized repository designed to store raw and unstructured data. It allows organizations to store data first and process it later as needed. For example, Uber leverages data lakes to power real-time analytics that support route optimization, dynamic pricing strategies, and fraud detection. This real-time processing enables Uber to make instant, data-driven decisions.

Amazon Article Explaining the difference between Data Warehouse, Data Mart, and Data Lake

Question 3

Will be building a fact table on Film Stats which will help us understand which film is famous among customers to rent by calculating the number of times a film has been rented and also total revenue it generated for the store. The table will also have rental date split to rental_day, rental_month, rental_year and return date split to return_day,return_month,return_year to understand how the film is performing over time and to get granular insights across different time periods.

This is done with the One Big Table approach since currently the number of columns are fairly small and to focus more on the speed of the query.

The table film_rental_stats will have the following columns:

1. film_id PK INTEGER - The unique identifier for the film.
2. film_title VARCHAR(255) - The title of the film.
3. rental_count INTEGER - The number of times the film has been rented.
4. total_revenue NUMBER(5,2) - The total revenue generated by the film.
5. rental_day INT - The day the film was rented.
6. rental_month INT - The month the film was rented.
7. rental_year INT - The year the film was rented.
8. return_day INT - The day the film was returned.
9. return_month INT - The month the film was returned.
10. return_year INT - The year the film was returned.

Initially an OLAP table is created from the above defined structure. The required data is taken from the Sakila database and inserted into the OLAP table. The OLAP table is then used for further analysis.

Question 4

```
# Cleans the environment before beginning the execution
# referredFrom:https://northeastern.instructure.com/courses/192346/assignments/2351524
rm(list = ls())

# Install program required packages ============================
# Installs RSQLite and DBI to connect to SqlLite.
# Referred from: http://artificium.us/lessons/06.r/l-6-104-r4progs/l-6-104.html#Install_Packages_on_Dem
installRequiredPackages <- function() {
  packages <- c("RSQLite", "DBI")
  # Install packages that are not installed
  installed_packages <- packages %in% rownames(installed.packages())
  if (any(installed_packages == FALSE)) {
    install.packages(packages[!installed_packages])
  }


  # Load all packages
  library(RSQLite)
  library(DBI)
}

# Create a DB Connection ============================
# Creates a DB connection with the given dbName
# @param: dbName - Name of the db to connect to
# referredFrom: http://artificium.us/lessons/06.r/l-6-301-sqlite-from-r/l-6-301.html#Connect_to_Databas
createDBConnection <- function(dbName) {
  return(dbConnect(RSQLite::SQLite(), dbName))
}


# Creates the OLAP table in the database
# @param: dbCon - database connection to connect to
createOneBigTable <- function(dbCon) {
  invisible(dbExecute(dbCon, "DROP TABLE IF EXISTS film_rental_stats;"))
  invisible(
    dbExecute(
      dbCon,
      "CREATE TABLE IF NOT EXISTS film_rental_stats(
    film_id INTEGER PRIMARY KEY,
    film_title VARCHAR(255) NOT NULL,
    rental_count INTEGER NOT NULL,
    total_revenue NUMBER(5,2) NOT NULL,
    rental_day INT NOT NULL,
    rental_month INT NOT NULL ,
    rental_year INT NOT NULL,
    return_day INT,
    return_month INT,
    return_year INT
  )"
    )
  )
```

```
}

# Insert into database in batches
# @param: dbCon - database connection to connect to
# @param: batchSize - batch size to insert into the database
# @param: initialQuery - insert query with table and column names
# @param: values - values to insert into the database
insertInBatches <- function(dbCon, batchSize, initialQuery, values) {
  numBatches <- ceiling(length(values) / batchSize)
  for (i in 1:numBatches) {
    startIdx <- (i - 1) * batchSize + 1
    endIdx <- min(i * batchSize, length(values))
    batchValues <- values[startIdx:endIdx]
    completeQuery <- paste(initialQuery, paste(batchValues, collapse = ","))
    dbExecute(dbCon, completeQuery)
  }
}


# Gets the film rental details especially the rental count of a film along with the total revenue gener
# Data is split to month, day and year for rental and return date to understand how the film is perform
# @param: dbCon - database connection to connect to
getFilmRentalDetailsFromSakila <- function(dbCon) {
  # strftime referred from: https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/strptime
  return (
    dbGetQuery(
      dbCon,
      "SELECT
    f.film_id,
    f.title AS film_title,
    COUNT(r.rental_id) AS rental_count,
    SUM(p.amount) AS total_revenue,
    strftime('%m', r.rental_date)  AS rental_month,
    strftime('%d', r.rental_date)  AS rental_day,
    strftime('%Y', r.rental_date)  AS rental_year,
    strftime('%m', r.return_date)  AS return_month,
    strftime('%d', r.return_date)  AS return_day,
    strftime('%Y', r.return_date)  AS return_year
FROM
    film f
JOIN
    inventory i ON f.film_id = i.film_id
JOIN
    rental r ON i.inventory_id = r.inventory_id
LEFT JOIN
    payment p ON r.rental_id = p.rental_id
GROUP BY
    f.film_id, f.title;"
    )
  )
}

# Inserts the film rental details into the OLAP table
```

```r
# @param: sakilaDB - database connection to connect to the sakila database
# @param: sakilaOlapDB - database connection to connect to the OLAP database
insertToOneBigTable <- function(sakilaDB, sakilaOlapDB) {
  detailsFromSakila <- getFilmRentalDetailsFromSakila(sakilaDB)
  query <- "INSERT INTO film_rental_stats (film_id,film_title, rental_count, total_revenue, rental_day,
  # referredFrom: https://ademos.people.uic.edu/Chapter4.html
  values <- apply(detailsFromSakila, 1, function(row) {
    sprintf(
      "(%s, '%s', %s, %s, %s, %s, %s, %s, %s, %s)",
      row["film_id"],
      row["film_title"],
      row["rental_count"],
      row["total_revenue"],
      row["rental_day"],
      row["rental_month"],
      row["rental_year"],
      row["return_day"],
      row["return_month"],
      row["return_year"]
    )
  })
  insertInBatches(sakilaOlapDB, 200 , query, values)
}

# Main function to run the program
main <- function() {
  installRequiredPackages()
  sakilaDB <- createDBConnection("sakila.db")
  sakilaOlapDB <- createDBConnection("sakilaOlap.db")
  createOneBigTable(sakilaOlapDB)
  insertToOneBigTable(sakilaDB, sakilaOlapDB)
  dbDisconnect(sakilaDB)
  dbDisconnect(sakilaOlapDB)
}
main()
```

Question 5

```r
# Gets the top 10 films that were rented across all the stores
# @param: dbCon - database connection to connect to
getTopTenFilms <- function(dbCon) {
  return (
    dbGetQuery(
      dbCon,
      "
      SELECT
    film_title,
    rental_count
FROM
    film_rental_stats
ORDER BY
    rental_count DESC
```

```
LIMIT 10;
      "))
}

# Displays table in a readable format
# @param: resultTable - table to display
# @param: colNames - column names of the table
displayTable <- function(resultTable,colNames) {
  print(colNames,row.names = FALSE)
  print(resultTable,row.names = FALSE,col.names = FALSE)
}

# Gets the trend of rental count and revenue generated across different months for year 2005
# @param: dbCon - database connection to connect to
getTrendOfRentalCountAndRevenue <- function(dbCon) {
  return (
    dbGetQuery(
      dbCon,
      "
      Select CASE
    WHEN rental_month = 1 THEN 'January'
    WHEN rental_month = 2 THEN 'February'
    WHEN rental_month = 3 THEN 'March'
    WHEN rental_month = 4 THEN 'April'
    WHEN rental_month = 5 THEN 'May'
    WHEN rental_month = 6 THEN 'June'
    WHEN rental_month = 7 THEN 'July'
    WHEN rental_month = 8 THEN 'August'
    WHEN rental_month = 9 THEN 'September'
    WHEN rental_month = 10 THEN 'October'
    WHEN rental_month = 11 THEN 'November'
    WHEN rental_month = 12 THEN 'December'
  END AS month,
    SUM(rental_count),
    SUM(total_revenue)
FROM film_rental_stats
WHERE rental_year = '2005'
GROUP BY rental_month
ORDER BY rental_month;
"))}



sakilaOlapDB <- createDBConnection("sakilaOlap.db")
topTenFilms <- getTopTenFilms(sakilaOlapDB)
trendOfRentalCountAndRevenue <- getTrendOfRentalCountAndRevenue(sakilaOlapDB)
```

Below is the top 10 films that were rented across all the stores

```
## [1] "Film Title"   "Rental Count"
##           film_title rental_count
##   BUCKET BROTHERHOOD           34
##     ROCKETEER MOTHER           33
```

```
##       FORWARD TEMPLE           32
##       GRIT CLOCKWORK           32
##       JUGGLER HARDLY           32
##  RIDGEMONT SUBMARINE           32
##        SCALAWAG DUCK           32
##        APACHE DIVINE           31
##    GOODFELLAS SALUTE           31
##         HOBBIT ALIEN           31
```

Below is the trend of rental count and revenue generated across different months for year 2005. The total revenue is presumed to be on the date the movie is rented. If the month is not present then the rent count and revenue generated is 0 for that month.

```
## [1] "Month"         "Rental Count"  "Total Revenue"
##   month SUM(rental_count) SUM(total_revenue)
##     May              3968           17293.27
##    June              3713           15112.87
##    July              7019           29258.88
##  August              1344            5741.54
```