Question 4 - Banking System ETL/DWH Design (20 Marks)

4.1 Designing measure columns for different currencies and interest calculations (5 Marks)

Solution: To handle three different currencies and interest calculation methods, I would design the fact table with the following approach:

1. **Store amounts in multiple columns:**
    - original_amount - Amount in source currency
    - original_currency - Currency code (USD, AUD, CAD)
    - standardized_amount_usd - All amounts converted to USD for comparison
    - interest_amount_original - Interest in original currency
    - interest_amount_usd - Interest converted to USD
2. **Add calculation metadata:**
    - interest_calculation_method - Type of calculation used (MONTH_END, 30TH_DAY, CUSTOM_CALENDAR)
    - interest_rate - Applied interest rate
    - interest_included_flag - Boolean to indicate if interest is included
3. **Currency conversion rates:**
    - Link to a separate currency conversion dimension table with daily rates
    - conversion_rate_to_usd - Rate used for conversion at transaction time

4.2 Handling different extraction times from source systems (5 Marks)

Solutions:

1. **Implement a staging area with timestamps:**
    - Create separate staging tables for each country (A_staging, B_staging, C_staging)
    - Add extraction_timestamp and processing_status columns
    - Process data asynchronously as it arrives
2. **Use a batch window approach:**
    - Define a common cut-off time (e.g., 2 AM local time for each country)
    - Hold data in staging until all sources have delivered
    - Process all data together in a single ETL run
3. **Implement incremental loading:**
    - Use CDC (Change Data Capture) or timestamp-based extraction
    - Each system maintains its own extraction watermark
    - Merge data in staging area before loading to DWH
4. **Add operational metadata:**
    - source_system_date - Business date from source
    - etl_load_date - When data was loaded to DWH
    - data_effective_date - Standardized business date across all systems

4.3 Linking interest data to customer details from CRM system (5 Marks)

Solution:

1. **Create a conformed Customer Dimension:**
    - Extract customer data from CRM system regularly
    - Create surrogate keys for customer dimension
    - Include customer attributes: CustomerID, Name, Type, Region, etc.
2. **Implement customer matching logic:**
    - Use business keys (customer IDs) from banking systems
    - Create a mapping table between banking system customer IDs and CRM customer IDs
    - Handle cases where customer IDs differ across systems
3. **Design the integration:**

CRM System → Customer_Staging → Customer_Dimension

↓

Banking Systems → Interest_Fact ← Customer_Key

4. **Handle missing customers:**
    - Create "Unknown Customer" records for unmatched cases
    - Implement a data quality process to identify and resolve mismatches
    - Regular reconciliation between systems

4.4 Incorporating monthly currency conversion rates from web service (5 Marks)

Design approach:

1. **Create Currency Exchange Rate infrastructure:**
    - Design a Currency_Exchange_Rate table:
    - - rate_id (SK)- from_currency- to_currency - exchange_rate- effective_date- expiry_date- source_system
2. **Implement web service integration:**
    - Create a separate ETL job for currency rates
    - Schedule to run monthly (or when rates are published)
    - Use web service connector in ETL tool (e.g., SSIS Web Service Task)
3. **ETL Process flow:**

Web Service → JSON/XML Parser → Staging_Currency_Rates

↓

Validation & Transformation

↓

Currency_Exchange_Rate (Type 2 SCD)

4. **Integration with main ETL:**
    - Main ETL job checks for latest rates before processing
    - Use effective dating to get correct rate for transaction date
    - Implement fallback logic if web service fails:
        - Use previous month's rates
        - Send alerts to administrators
        - Log missing rate periods
5. **Error handling:**
    - Validate rate ranges (e.g., rates should be positive)
    - Check for completeness (all required currency pairs)
    - Maintain audit trail of rate changes

This design ensures accurate currency conversion while maintaining data quality and system reliability.

JULY 2024 PAPER - QUESTION 2 ANSWERS

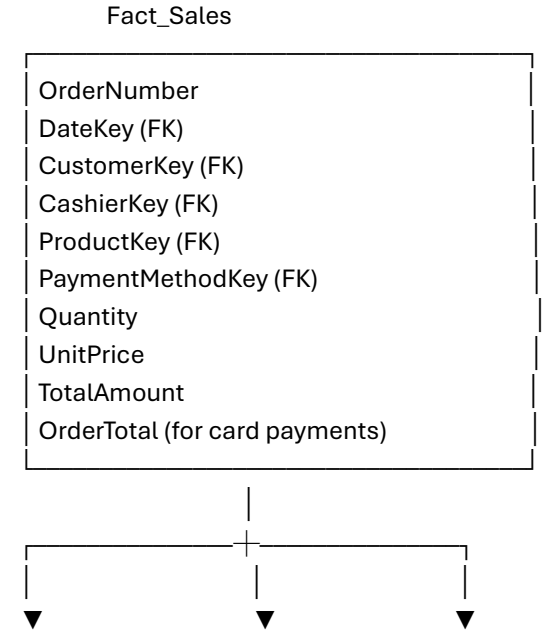
2.1 Dimension Tables with Relevant Attributes (5 Marks)

Based on the dataset, the following dimension tables should be created:

1. **Dim_Date**
    - DateKey (SK)
    - Date
    - Day
    - Month
    - Quarter
    - Year
    - DayOfWeek
    - IsWeekend
    - IsHoliday
2. **Dim_Customer**
    - CustomerKey (SK)
    - CustomerCode
    - CustomerName
    - CustomerType
    - Address
    - City
    - Country
3. **Dim_Cashier**
    - CashierKey (SK)
    - CashierCode
    - CashierName
    - Department
    - HireDate

4. **Dim_Product**
    - ProductKey (SK)
    - ItemCode
    - ProductName
    - Category
    - SubCategory
    - UnitCost
    - Supplier
5. **Dim_PaymentMethod**
    - PaymentMethodKey (SK)
    - PaymentType (CRD/CASH)
    - CardType (VISA/MASTER/NULL)
    - Description

2.2 Star Schema Design (5 Marks)



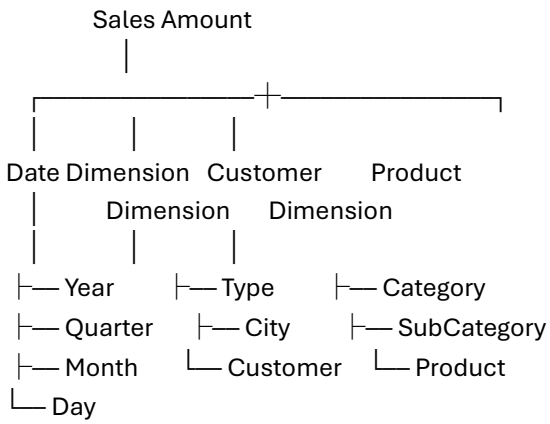
2.3 Pseudo Code/Logic for ETL Process (10 Marks)

```
PROCEDURE ProcessOrderTextFile(filePath)
  // Step 1: Create temporary tables
  CREATE TEMP TABLE Header_Records (
    OrderNumber, OrderDate, CustomerCode, CashierCode
  )
  CREATE TEMP TABLE Detail_Records (
    OrderNumber, ItemCode, Quantity, UnitPrice
  )
  CREATE TEMP TABLE Total_Records (
    OrderNumber, TotalAmount, PaymentType, CardType
  )

  // Step 2: Read and parse the file
  file = OPEN(filePath)
  WHILE NOT EOF(file)
    line = READLINE(file)
    columns = SPLIT(line, ',')

    orderNumber = columns[0]
```

```
    orderType = columns[1]

    IF orderType = 'H' THEN
        // Process Header Record
        orderDate = columns[2]
        customerCode = columns[3]
        cashierCode = columns[4]
        INSERT INTO Header_Records VALUES(orderNumber, orderDate, customerCode, cashierCode)

    ELSE IF orderType = 'D' THEN
        // Process Detail Record
        itemCode = columns[2]
        quantity = CAST(columns[3] AS INTEGER)
        unitPrice = CAST(columns[4] AS DECIMAL)
        INSERT INTO Detail_Records VALUES(orderNumber, itemCode, quantity, unitPrice)

    ELSE IF orderType = 'T' THEN
        // Process Total Record
        totalAmount = CAST(columns[2] AS DECIMAL)
        paymentType = columns[3]
        cardType = columns[4]
        INSERT INTO Total_Records VALUES(orderNumber, totalAmount, paymentType, cardType)
    END IF
END WHILE
CLOSE(file)

// Step 3: Join tables and insert into fact table
INSERT INTO Fact_Sales
SELECT
    h.OrderNumber,
    d.DateKey,
    c.CustomerKey,
    cs.CashierKey,
    p.ProductKey,
    COALESCE(pm.PaymentMethodKey, 1), -- Default to CASH if no T record
    det.Quantity,
    det.UnitPrice,
    det.Quantity * det.UnitPrice as TotalAmount,
    t.TotalAmount as OrderTotal
FROM Header_Records h
INNER JOIN Detail_Records det ON h.OrderNumber = det.OrderNumber
LEFT JOIN Total_Records t ON h.OrderNumber = t.OrderNumber
LEFT JOIN Dim_Date d ON h.OrderDate = d.Date
LEFT JOIN Dim_Customer c ON h.CustomerCode = c.CustomerCode
LEFT JOIN Dim_Cashier cs ON h.CashierCode = cs.CashierCode
LEFT JOIN Dim_Product p ON det.ItemCode = p.ItemCode
LEFT JOIN Dim_PaymentMethod pm ON t.PaymentType = pm.PaymentType
    AND (t.CardType = pm.CardType OR (t.CardType IS NULL AND pm.CardType IS NULL))

// Step 4: Clean up temporary tables
DROP TABLE Header_Records
DROP TABLE Detail_Records
DROP TABLE Total_Records
```

```
    // Step 5: Log ETL execution
    INSERT INTO ETL_Log VALUES(CURRENT_TIMESTAMP, 'Order File Processing', 'Success', ROW_COUNT())
END PROCEDURE
```

2.4 OLAP Cube Design (5 Marks)



Measures: Quantity, Sales Amount, Order Count, Average Order Value

2.5 Traditional Challenges with Text Files and Workarounds (5 Marks)

Challenges:

1. **Data Type Inconsistencies** - String values in numeric fields
    - *Workaround:* Implement data validation and type casting with error handling
2. **Missing/Incomplete Records** - Missing header or detail records
    - *Workaround:* Create orphan record tables and reconciliation reports
3. **File Format Changes** - Column order or delimiter changes
    - *Workaround:* Use configuration files to define file formats dynamically
4. **Character Encoding Issues** - Special characters, different encodings
    - *Workaround:* Standardize to UTF-8, implement encoding detection
5. **Large File Processing** - Memory issues with large files
    - *Workaround:* Implement streaming/batch processing, use staging tables

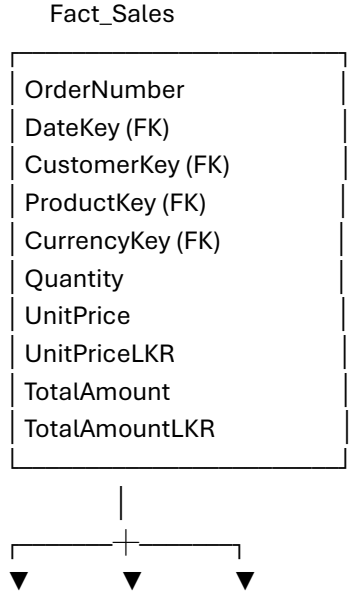
---

AUGUST 2023 PAPER - QUESTION 2 ANSWERS

2.1 Dimension Tables (3 Marks)

1. **Dim_Date** - DateKey, Date, Day, Month, Year, Quarter
2. **Dim_Customer** - CustomerKey, CustomerCode, CustomerName, CustomerType (Local/International)
3. **Dim_Product** - ProductKey, ItemCode, ProductName, Category, UnitCost
4. **Dim_Currency** - CurrencyKey, CurrencyCode (LKR/USD), ExchangeRate, EffectiveDate

2.2 Star Schema (5 Marks)



Dim_Date Dim_Customer Dim_Product
Dim_Currency

2.3 Pseudo Code/Logic (7 Marks)

```
PROCEDURE ProcessInternationalOrderFile(filePath)
  // Step 1: Create temporary tables
  CREATE TEMP TABLE Header_Records (
    OrderNumber, OrderDate, CustomerCode, CustomerType
  )
  CREATE TEMP TABLE Detail_Records (
    OrderNumber, ItemCode, Quantity, UnitPrice, Currency
  )

  // Step 2: Get current exchange rate
  currentUSDRate = SELECT ExchangeRate FROM Dim_Currency
            WHERE CurrencyCode = 'USD'
            AND EffectiveDate <= CURRENT_DATE
            ORDER BY EffectiveDate DESC LIMIT 1

  // Step 3: Process file
  file = OPEN(filePath)
  currentHeader = NULL

  WHILE NOT EOF(file)
    line = READLINE(file)
    columns = SPLIT(line, ',')

    orderNumber = TRIM(columns[0])
    orderType = TRIM(columns[1])

    IF orderType = 'H' THEN
      // Store header info
      currentHeader = {
        orderDate: columns[2],
        customerCode: columns[3],
        customerType: columns[4]
      }
      INSERT INTO Header_Records VALUES(orderNumber, currentHeader)

    ELSE IF orderType = 'D' AND currentHeader IS NOT NULL THEN
      // Process detail with currency conversion
      itemCode = columns[2]
      quantity = CAST(columns[3] AS INTEGER)
      unitPrice = CAST(columns[4] AS DECIMAL)

      IF currentHeader.customerType = 'International' THEN
        currency = 'USD'
      ELSE
        currency = 'LKR'
      END IF
      INSERT INTO Detail_Records VALUES(orderNumber, itemCode, quantity, unitPrice, currency)
    END IF
  END WHILE
```

```
  // Step 4: Insert into fact table with currency conversion
  INSERT INTO Fact_Sales
  SELECT
    h.OrderNumber,
    d.DateKey,
    c.CustomerKey,
    p.ProductKey,
    cur.CurrencyKey,
    det.Quantity,
    det.UnitPrice,
    CASE
      WHEN det.Currency = 'USD' THEN det.UnitPrice * currentUSDRate
      ELSE det.UnitPrice
    END as UnitPriceLKR,
    det.Quantity * det.UnitPrice as TotalAmount,
    CASE
      WHEN det.Currency = 'USD' THEN det.Quantity * det.UnitPrice * currentUSDRate
      ELSE det.Quantity * det.UnitPrice
    END as TotalAmountLKR
  FROM Header_Records h
  INNER JOIN Detail_Records det ON h.OrderNumber = det.OrderNumber
  LEFT JOIN Dim_Date d ON h.OrderDate = d.Date
  LEFT JOIN Dim_Customer c ON h.CustomerCode = c.CustomerCode
  LEFT JOIN Dim_Product p ON det.ItemCode = p.ItemCode
  LEFT JOIN Dim_Currency cur ON det.Currency = cur.CurrencyCode

  // Step 5: Cleanup
  DROP TABLE Header_Records
  DROP TABLE Detail_Records
END PROCEDURE
```

2.4 Solutions for String Values in Quantity Column (5 Marks)

1. **Data Validation Layer**
   - Add TRY-CATCH blocks to handle conversion errors
   - Log errors to an error table with original values
2. **Data Cleansing Rules**
   - Remove non-numeric characters (e.g., "10pcs" → 10)
   - Handle common patterns (e.g., "dozen" → 12)
3. **Default Value Strategy**
   - Use NULL or 0 for invalid quantities
   - Flag records for manual review
4. **Staging Table Approach**
   - Load all data as strings initially
   - Apply transformation rules in staging
   - Only move clean data to fact table
5. **Business Rule Implementation**
   - Create lookup table for text-to-number mappings
   - Implement alerts for new unmapped values

Understanding INNER JOIN and LEFT OUTER JOIN in SQL

Overview

Joins are used to combine rows from two or more tables based on a related column between them. Let's understand the two most common types using the order processing example from your questions.

Sample Data Tables

Header_Records Table (H records)

OrderNumber	OrderDate	CustomerCode	CashierCode
100020	2006-11-01	CUS001	CASHIER001
100021	2006-11-01	CUS002	CASHIER002
100022	2006-11-01	CUS003	CASHIER003

Detail_Records Table (D records)

OrderNumber	ItemCode	Quantity	UnitPrice
100020	CN-6137	10	50.00
100020	AR-5381	10	25.00
100021	AR-5381	5	10.00
100021	CA-5965	3	15.00

Total_Records Table (T records - for card payments only)

OrderNumber	TotalAmount	PaymentType	CardType
100021	195.00	CRD	VISA

1. INNER JOIN

Definition

INNER JOIN returns **only the rows that have matching values in both tables**. If there's no match, the row is excluded from the result.

Syntax

```
SELECT columns
FROM table1
INNER JOIN table2 ON table1.column = table2.column
```

Example: Join Header and Detail Records

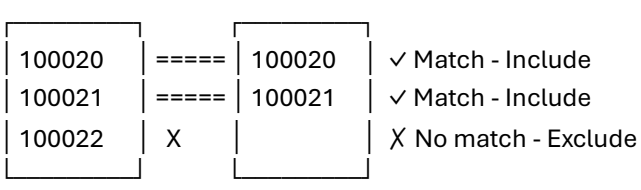
```
SELECT
  h.OrderNumber,
  h.OrderDate,
  h.CustomerCode,
  d.ItemCode,
  d.Quantity,
  d.UnitPrice
FROM Header_Records h
INNER JOIN Detail_Records d ON h.OrderNumber = d.OrderNumber
```

Result:

OrderNumber	OrderDate	CustomerCode	ItemCode	Quantity	UnitPrice
100020	2006-11-01	CUS001	CN-6137	10	50.00
100020	2006-11-01	CUS001	AR-5381	10	25.00
100021	2006-11-01	CUS002	AR-5381	5	10.00
100021	2006-11-01	CUS002	CA-5965	3	15.00

Note: Order 100022 is NOT in the result because it has no matching detail records.

Visual Representation:



2. LEFT OUTER JOIN (or LEFT JOIN)

Definition

LEFT JOIN returns **all rows from the left table**, and the matched rows from the right table. If there's no match, NULL values are returned for columns from the right table.

Syntax

```
SELECT columns
FROM table1
LEFT JOIN table2 ON table1.column = table2.column
```

Example: Join All Orders with Total Records

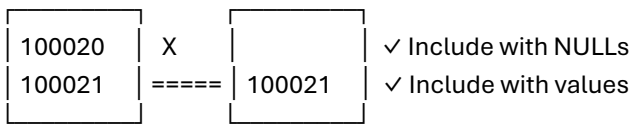
```
SELECT
  h.OrderNumber,
  h.CustomerCode,
  d.ItemCode,
  d.Quantity,
  t.TotalAmount,
  t.PaymentType,
  t.CardType
FROM Header_Records h
INNER JOIN Detail_Records d ON h.OrderNumber = d.OrderNumber
LEFT JOIN Total_Records t ON h.OrderNumber = t.OrderNumber
```

Result:

OrderNumber	CustomerCode	ItemCode	Quantity	TotalAmount	PaymentType	CardType
100020	CUS001	CN-6137	10	NULL	NULL	NULL
100020	CUS001	AR-5381	10	NULL	NULL	NULL
100021	CUS002	AR-5381	5	195.00	CRD	VISA
100021	CUS002	CA-5965	3	195.00	CRD	VISA

Note: Order 100020 appears with NULL values for Total_Records columns because it was a cash payment (no T record).

Visual Representation:



Key Differences Summary

Aspect	INNER JOIN	LEFT JOIN
Returns	Only matching rows	All rows from left table
Non-matching rows	Excluded	Included with NULLs
Use when	You need only complete data	You need all records regardless of match
Example use case	Find all orders WITH details	Find all orders, including those without payment info

Why Use This Pattern in ETL?

In the order processing example:

1. **INNER JOIN for Header + Detail**:
   - Every order MUST have at least one detail line
   - Orders without details are likely errors
2. **LEFT JOIN for Total Records**:
   - Not all orders have card payments (only card payments have T records)
   - Cash payments won't have a Total record
   - We still want to process cash orders

Complete ETL Example

```
-- This query processes all orders correctly
INSERT INTO Fact_Sales
SELECT
  h.OrderNumber,
  h.OrderDate,
  h.CustomerCode,
  d.ItemCode,
  d.Quantity,
  d.UnitPrice,
  d.Quantity * d.UnitPrice as LineTotal,
  COALESCE(t.PaymentType, 'CASH') as PaymentType,  -- Default to CASH if NULL
  t.CardType
FROM Header_Records h
INNER JOIN Detail_Records d ON h.OrderNumber = d.OrderNumber  -- Must have details
LEFT JOIN Total_Records t ON h.OrderNumber = t.OrderNumber    -- May or may not have total
```

Practice Questions

1. What would happen if we used INNER JOIN for both Detail and Total records?
    - Answer: We would lose all cash payment orders!
2. What if we used LEFT JOIN for both?
    - Answer: We might include orders without any detail lines (which could be data errors).
3. How do we handle NULL values from LEFT JOIN?
    - Answer: Use COALESCE() or ISNULL() to provide default values.

DIGITAL SIGNATURE

Document Information

Document: Question 2 both papers.pdf

Company: Tech Solutions Lanka (Pvt) Ltd

Signature Details

Signed by: David

Print Name: David

Email: david.anderson@gmail.com

Date: 2025-07-17

IP Address: 127.0.0.1

Timestamp: 2025-07-17 12:53:18 UTC