Problem Statement - Building an analytical data warehouse for faster and efficient insight generation.

As part of **Amazon Books**, a data integration program initiated for merging print books and e-books to analyze the performance of overall books business.

You as a data engineer, are given a task to design an efficient data warehouse to analyze the books business performance and the stakeholders should be able to answer few questions like:

1. **What are the top 10 books sold weekly**
2. **What are the other products frequently bought with books**
3. **Who are the top Authors based on books sales**

We need Amazon books real time business data to answer these questions. Three tier application architecture is a client-server architecture which separates applications into a presentation tier, an application tier and a data tier. In the data tier, real time data which is processed by the application is retrieved, collected, stored and managed in a database.

Amazon uses DynamoDB as one of its database systems in its architecture that manages transactions performed over the ecommerce application by a customer. And this is an **online transaction processing system.**

What is OLTP?

OLTP is a data processing system that supports real time database transactions executed by a large number of people over the internet typically. A transaction could be an insert, delete, update of data in a database.

Features:

1. Response time should be faster in OLTP systems for users to remain productive. Say if a customer searching for a product in Amazon.com, the response time should be shorter for smooth interactions
2. High Concurrency - To perform large number of transactions concurrently
3. Large data volumes - OLTP systems store large volumes of data and can grow bigger over time.

While the OLTP system has numerous features and scaled significantly, it is still not suitable for our business requirements.

Why it is not suitable for Analytics

OLTP performs numerous small transactions that result in large data sets. OLTP contributes an immediate record of current business activity. However, providing insights from these data is challenging because,

1. To generate and validate business insights, historical data that is compiled over time is required but OLTP is designed for real time execution. If we access Amazon.com, we can only find the current price of a product and not its price history. As in OLTP the historical data won't be stored because storing history exponentially increases the storage and potentially slows down the response time. As general, OLTP should be light for fast performance.

2. In OLTP system data will be accessed by multiple users at same time. If two or more customers buy the same product at the same time on Amazon then the transaction data will get updated simultaneously by introducing two new records of transactions.

   And if we try to perform analytics like aggregate calculations of trying to identify overall sales on a particular day against these transaction data, that depends on millions of individual transactions, are very resource intensive for an OLTP system. This could again slow down the execution by blocking other transactions in the database that leads to slower response time which affects user experience and concurrency.

3. OLTP is primarily designed for collecting large data transactions and not necessarily to analyze that data in the aggregate. If we need to see a summary of sales performance over the years, we need to aggregate data to yearly detail. But In OLTP, we have real time data and to aggregate millions of transactions yearly and if required we need to join with other transaction tables like joining print book transactions with Kindle book transactions involves poor performance.

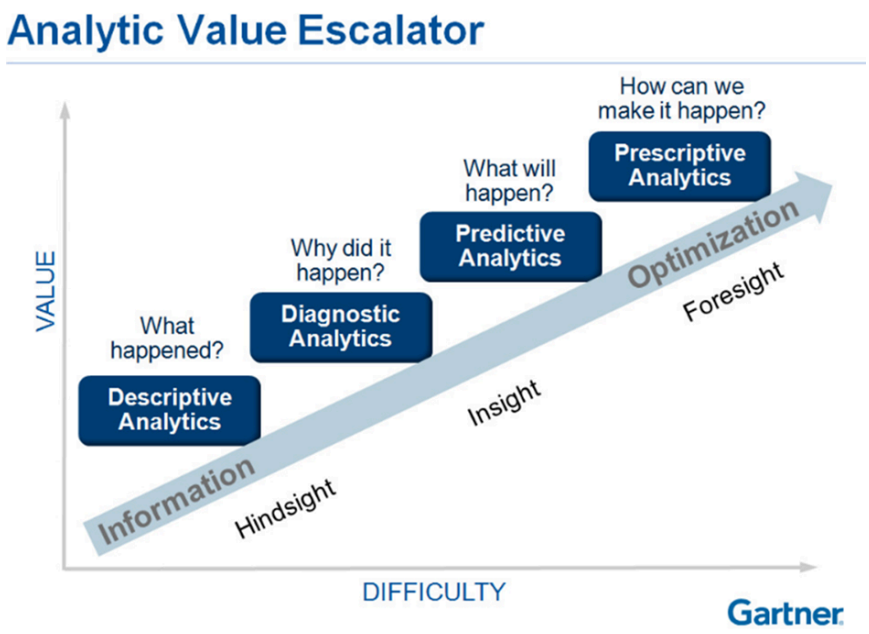
To summarize, performing analytics on top of OLTP involves a very resource intensive, slower query response and heavy OLTP system.

What is the better system for analytics

To support large business operations, complex analysis and allow users to access data in business context a high performance storage system is required.
Amazon uses **Datawarehouse** as its core component to store the business data and perform analytics on top of it.

Introduction to Data warehousing

The cumulative process of collecting an organization's business activities data, storing and transforming it into actionable insights is called **Business Intelligence(BI).**



For BI, there are multiple components involved like reporting, ETL etc., and one of the core components is a **Data Warehouse** which is a storage infrastructure.

This chapter covers Data warehousing, Dimensional Modeling problems and what are the best practices which distinguish a skilled Data Engineer from a novice one.

What is Data Warehouse

A data warehouse is a storage unit environment to support business analytics related activities like data analysis and reporting. This component is considered as the core of Business intelligence. Typically a data warehouse contains a large volume of current and historical business data.

How data is moved from Database to Data warehouse?

Extract Transform Load (ETL) is a data integration process to move data from one or more data source systems to a single data store target.

ETL is often used by an organization to:

- Extract data from source systems and in our case its DynamoDB

- Transform the raw data into reliable data to establish consistency and improve data quality. There are specific ETL tools to perform transformations.
- Load transformed data into a target system and we use a Data warehouse here.

Let's consider a raw transaction table of print books,

OrderId	CustomerId	InvoiceNo	Invoice Date	ShippingAddress	PaymentMethod	SoldBy	BillingAddress	PlaceOfSupply	PlaceOfDelivery	ASIN	Item.No	Description	UnitPrice	Qty	Net Amount	Tax Amount	Total Amount	Shipping
7146	101	0990	23 August 2022	Bangalore	COD	CT	Bangalore	Chennai	Bangalore	B0963RYQSW	1	Adapter	100	1	100	18	118	0

And as per amazon privacy policy, the customer details should be encrypted as this personal information should not be accessible or visible to engineers.
We can mask customer related details such as customer shipping address and billing address to abide with respect to privacy.
These encryption algorithms can be performed in the ETL layer. After performing the data might be like,

OrderId	CustomerId	InvoiceNo	Invoice Date	ShippingAddress	PaymentMethod	SoldBy	BillingAddress	PlaceOfSupply	PlaceOfDelivery	ASIN	Item.No	Description	UnitPrice	Qty	Net Amount	Tax Amount	Total Amount	Shipping
7146	101	0990	23 August 2022	sjkgfjsvfjsvn	COD	CT	sjkgfjsvfjsvn	Chennai	sjkgfjsvfjsvn	B0963RYQSW	1	Adapter	100	1	100	18	118	0

Other Examples For Instructor

1. The same transaction table splits to order details table and invoice details table
2. To join print books and kindle books transactions into a single table

Why is data warehouse a better option?

1. **Business Oriented -** Data warehouses are intended to contain large amounts of business data.
2. **Fast Query Response Time -** Data Warehouse is mainly designed for business analytics
3. **Data quality, consistency, and accuracy** - Data warehouse is nonvolatile as once data is moved to data warehouse, it shouldn't change.
4. **Integrated** - If the source data is from multiple data sources, we can combine all data and store in data warehouse
5. **Separate Data warehouse from OLTP** - Separating business analysis from transactional databases improves performance on both systems

For Instructor

1. **Business Oriented -** Data warehouses are intended to contain large amounts of business data rather than to focus on real time operations and transactions.

   In our case study, we are analysing Amazon's books business and a central repository is required to focus on books. This storage can be used not only for books sales performance included but not limited to analysing customer feedback, Customer personalized experience, most famous genre. So a single place where business user can analyse the data and fuel insights on multiple standpoints.

2. **Fast Query Response Time -** Data Warehouse has ability to boost query performance as we will be designing the datawarehouse intended for business analytics. In OLTP, the tables are designed to optimize the CRUD (Create, Read, Update and Delete) operations. And datawarehouses are designed for read operations to aggregate data, complex analysis, joining multiple tables to summarize business data and provide business insights.

3. **Data quality, consistency, and accuracy** - In the OLTP system, the data will keep getting updated based on user interactions and it won't be suitable for business reports.

   Consider, A business leader at amazon is accessing books business sales data. If the data is connected to OLTP, then based on real time transactions the revenue will keep changing. The first minute it shows $500M and next second it might turn to $550M. And this data is not reliable and consistent. We need data warehouse to store the summary data where the user can rely on. And this data will be refreshed periodically.

4. **Integrated** - Let's consider Amazon uses DynamoDb for print books but for ebooks it might use a different operational database like MySQL. To analyze the overall books business, we need all data in one place(data warehouse).

5. **Separate Data warehouse from OLTP** - Separating business analysis from transactional databases improves performance on both systems.

   Here we are analyzing books sales performance and to see insights of print and ebooks sales performance over the years. Let's consider the required data is available in OLTP and business users triggers a complex to join pbooks and ebooks. This activity can potentially block the user transaction and the database might go to a fixed state for some time in order to run a complex query and halt a user experience.

How to design a Data Warehouse

As a first step, we should understand the organization business and its requirements to design a data warehouse.

Let’s break down the given requirements and categorize what data needs to be stored in a data warehouse. There could be a wide range of business data. Here we are narrowing down to a few commonly-seen books' business requirements.

1. Books and Other Products transaction data to identify sales performance of books business and other products frequently bought with books
2. Books specific attributes like Name, Author and Publisher details
3. Other category Products related attributes
4. Vendor specific information
5. Marketplace related data

Now, we can’t accommodate everything into one big table which is not scalable, not robust, and does not separate different aspects of the model.

Each data point has a different use case – for example in a transaction, values like the total number of books sold weekly would be changing and somewhat unique, whereas information like the author of the book is not changing.

Considering large volumes of data and a wide range of business, there is a design required to store data efficiently for faster query retrieval and high performance.

Datawarehouse is a storage area and we need to organize stored business data to support complex analysis. **Dimensional Modeling** facilitates these analysis by designing a data warehouse efficiently for fast querying and data retrieval processes.

Dimensional Modeling is a technique which includes a set of methods, practices and concepts to design and store data in a data warehouse for storage optimization and boost performance.

Dimensional Modeling Approach:

Dimensional modeling is oriented to support end user queries in data warehouses and a strong requirement for a well designed data warehouse.This can be summarized as structuring data into Dimensions and Fact and form their relationship which enables users to fetch data efficiently. This approach focuses around faster querying, easy design and executing numerical analytics on business data.

Fact

Facts are the measurements/metrics or facts from your business process.
In our case, our facts are –

- QtySold
- OrderAmount

Dimension

Dimension provides the context surrounding a business process event. In simple terms, they give “Who, What, Where” of a fact.

- Example- In the Sales business process, for the fact quarterly sales number, dimensions would be
    - Who – Author, Publisher, Customer
    - Where – Location
    - What – Book Name

In our case, our dimensions are –

- Book Name
- Book Author
- Other Products Name
- Unit Price
- Marketplace
- Country
- Vendor Name

What is the Business Objective

Identifying the business process is the first step In dimensional modeling, the analysis starts with the purpose of the data and organization’s business interest(s).
In the given use case, the business process is to analyze the business performance of books and other products.

What is the granularity of data

Based on business purpose, Identifying the lowest level of data which is captured by business to analyze or resolve the problem.
In our case, Each row in the table represents a single transaction done by a customer to purchase one or more books or products from a vendor including the transaction amount, the number of items and order fulfillment type like whether order is successful or canceled. So a single row contains a transaction id with single ASIN(Amazon Standard Identification Number) and order fulfillment type along with vendor and marketplace details

What are the required Dimension and Fact tables

Dimension Table briefs about the business objects irrespective of any business event. Dimension provides descriptive information about the measures.
Let's consider the below Dimensions for Books Datawarehouse

1. PrintBooks
2. Ebooks
3. Date

4. OtherProducts
5. Marketplace

Other possible dimensions could be Publishers, Vendors, Customers

Fact Table
Fact table provides quantitative transactional metrics of the referenced dimension tables. These facts are required to understand the current business process and forecast it.

1. PTransactions - Fact table for Print book Transactions
2. ETransactions - Fact table for E Transactions

The primary key in Dimension table could be a foreign key in fact table

Why we need dimensional modeling

- To maintain a robust data warehouse for scaling along with data volume. Data warehouse volume grows over the time and an optimization technique required to organize data which helps in reducing cost along data management.

  In books business data, we have print books and ebooks data. If we store the data in the raw format instead of modeling, over the time data growth would be high and if we store all data in one table there would be redundancy. For example, consider the vendor name is 100 characters. Without modeling, we need to duplicate the vendor name over all the millions of records. But with modeling we will be invoking only id column

- Architecting the model as a solution specific to a business problem.

  The books transaction table contains multiple business information like vendor details, customers, Books, Authors, State tax, Sales.
  In our use case we are trying to identify only sales data so including other information in the table will lead to increased data volume. In this case even the aggregated data volume would be high as there are multiple attributes of data.

- Assembling the related tables to each other for optimized joining and easy querying. This yields a faster query response.

  In the book business, we are trying to solve sales business problems. But there is a requirement to analyze customer reviews. In this case, without disturbing the current architecture we can introduce new reviews fact tables. And can be easily joined with existing Customer PBooks and EBooks table

- Dimensional modeling enables flexible warehouse as the design is more extendable to accommodate any adjustments based on future business requirements

Let’s consider, we are required to capture ‘TotalTimeRequiredToRead’ field to understand how many minutes the user spends on a page in a book. To accomplish this we need to introduce new fields in the Books Dim Table along with a new fact table to analyze the user statistics.
Without dimensional modeling, we need to update a large volume of data to accommodate this change. And even if we apply the change,it might impact the existing business applications. But modeling enables us to make this required change only in dimension and no fact table would be impacted.

How to form relationships between tables

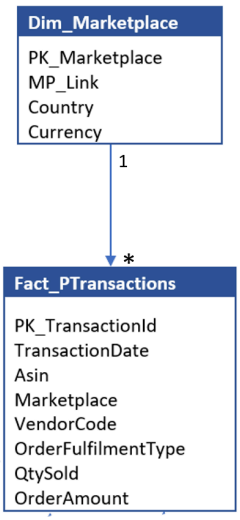
After identifying the dimension and fact tables, we need to form a relationship between tables to illustrate logical connections related to each other.

Types of Relationships

There are three type of relationships

One to Many Relationship

Such a relationship exists when each record of one table can be related to one or more records of the other table. This is the most common relationship form.
Let's consider, Marketplace Dimension table which has markeplace as PKey. And in PTransactions fact table many instances of marketplace id would be available based on customer transactions. And to form relationships between these two tables we need to logically relate them based on marketplace.
So a single record in the marketplace table would be connected to multiple records in fact.



One to One Relationship

A relationship between each record of one table is related to only one record of the other table.
We don't have this relationship in our use case.
[For the instructors] Other examples:

1. In Amazon.com, relating Customer_id from customer dimension and CutomerKYC dimension

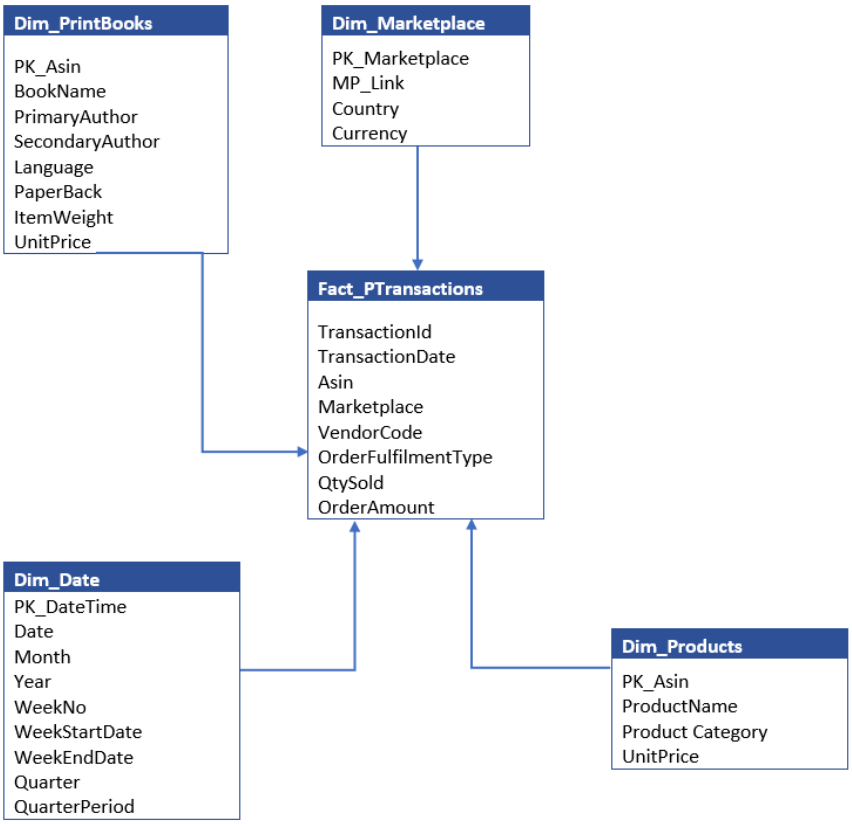
Many to Many Relationship

A relationship exists when each record of the first table can be related to one or more records of the second table and vice versa.
We don't have this relationship in our use case.
[For the instructors] Other examples:

1. In Amazon.com, A customer can have multiple addresses stored in CustomerAddress table and multiple payment details in CustomerPayment table. When joining these two tables on customer_id many to many forms

Star Schema

This is the simplest form of data relationship structure. In this schema, the Fact table in the center is surrounded by a series of dimensions.
To create a data model using the star schema framework, need to identify facts and dimension tables and connect them via foreign keys



Star schema is a schema that utilizes facts and dimensions to create a simpler yet efficient dimensional model. Few characteristics are as follows,

1. Every dimension in a star schema is represented with the only one-dimension table.
2. The dimension table should contain the set of attributes.
3. The dimension table is joined to the fact table using a foreign key
4. The dimension table are not joined to each other
5. Fact table would contain key and measure

Advantages of using star schema:

1. The Star schema is easy to understand and provides optimal disk usage
2. The schema is widely supported by BI Tools

Snowflake schema

Snowflake schema is an extended form of star schema by normalizing dimension tables. A single dimension table can be broken down into multiple dimensions through Normalization.
In snowflake schema, a single Fact table connected with multiple Dimension tables and each dimension connected to multiple sub dimensions

Snowflaking is a concept of applying normalization to dimension tables by creating separate dimensions through removal of low cardinality attributes.

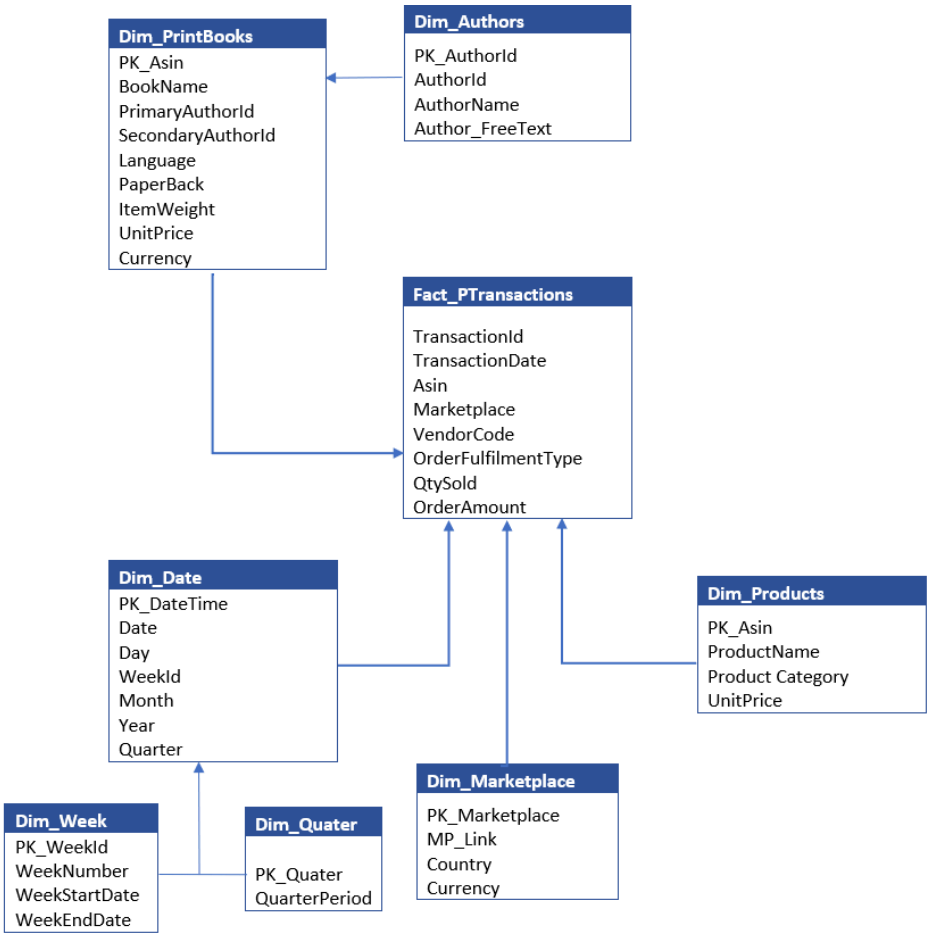
Advantages of using snowflake schema:

1. Consumes less storage
2. Low data redundancy

Disadvantages

1. Joining multiple tables leads to complexity due to multiple levels of attribute normalization to form sub dimensions
2. Query performance is lower
3. Design is complex comparing Star schema

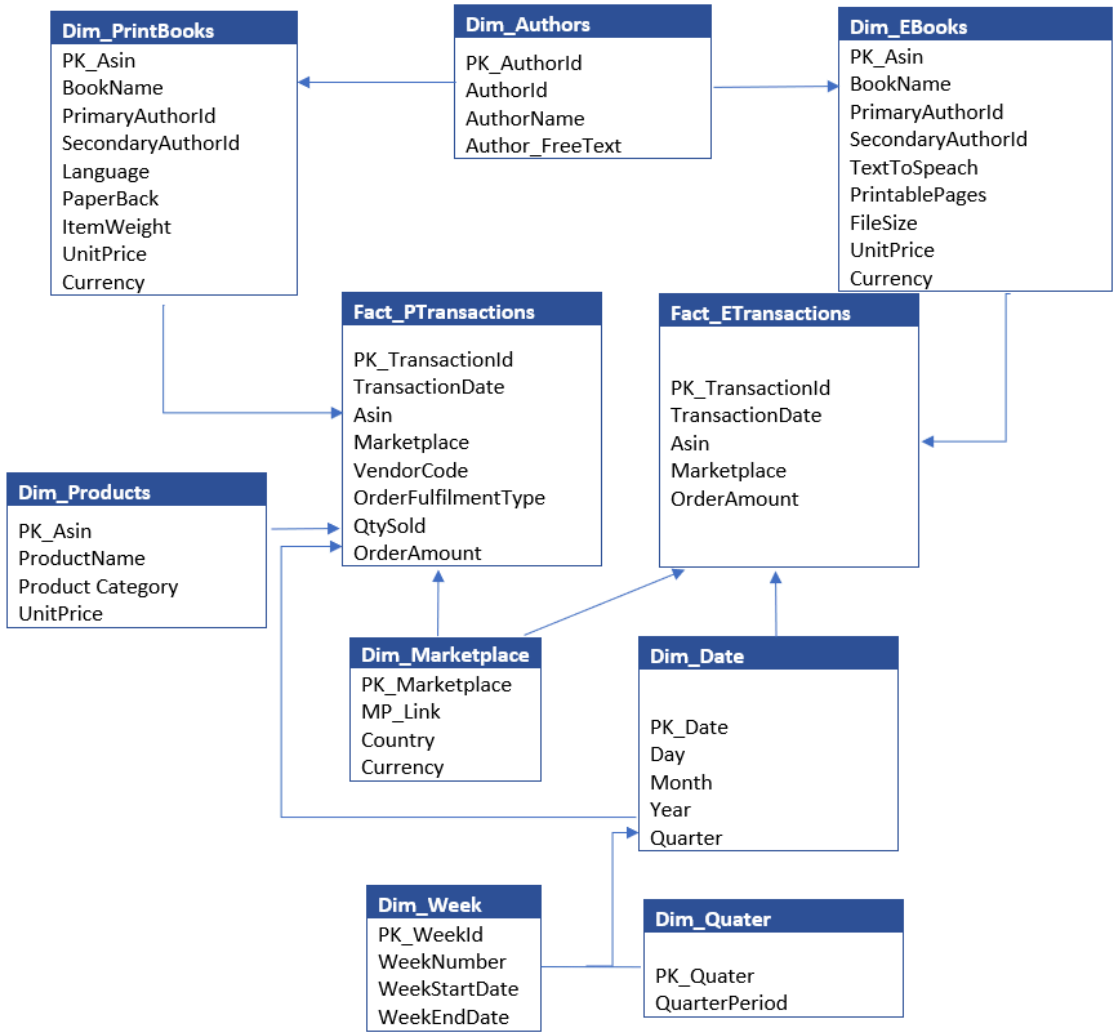
Designing Snowflake design for our case,
A book can have multiple authors and an author corresponds to multiple books. Inorder to eliminate duplicates of author information within books tables, let's break down books table to Author table. And Date Dim to Quarter and week dimension



Galaxy/Fact Constellation Schema

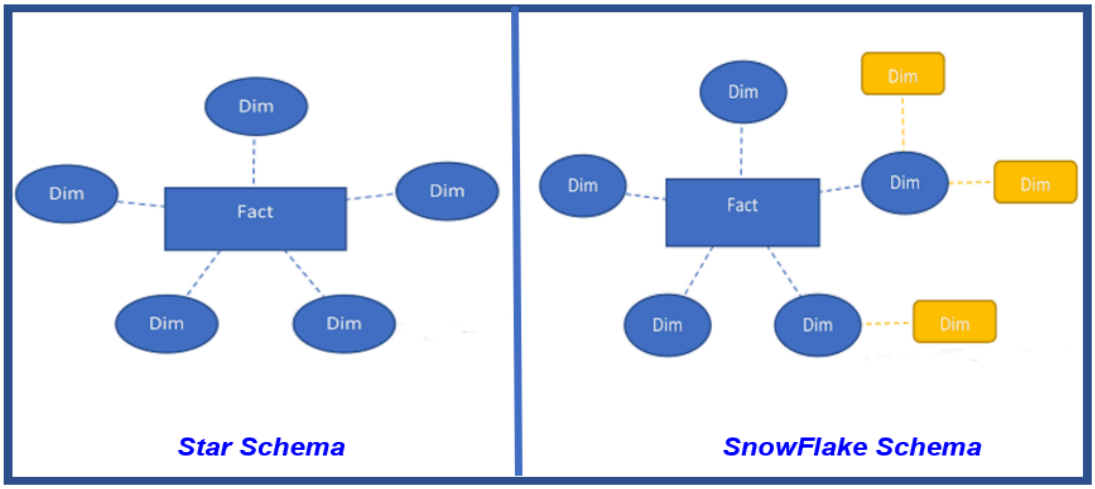
This is a compound schema consisting of more than one fact and normalized dimensions. Galaxy schema is more complex than the star and snowflake schema. When the business is large where it requires multiple fact tables and contains sophisticated applications, it is best to choose galaxy schema considering the complexity.

In our case study, we have two fact tables - PTransactions and ETransactions which have different attributes. In star and snowflake schema, no possibility to accommodate two or more fact tables



Which Schema to choose in our use case?

Based on the business purpose, one needs to choose the best schema. Dimension level analysis is faster in Snowflake like answering “How Many Books I have in Stock”.
Fact Level analysis is faster in Star schema such as “What is my Total revenue in Books”.

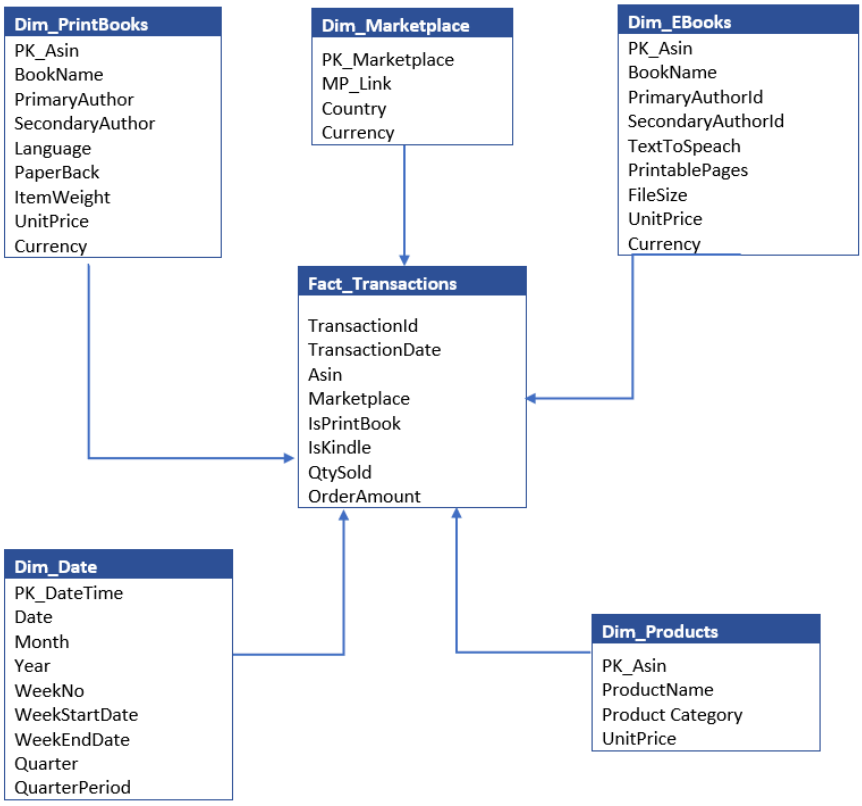


	Snowflake Schema	Star Schema
Ease of maintenance / change	No redundancy and hence more easy to maintain and change	Has redundant data and hence less easy to maintain/change
Ease of Use	More complex queries and hence less easy to understand	Less complex queries and easy to understand
Query Performance	More foreign keys-and hence more query execution time	Less no. of foreign keys and hence lesser query execution time
Type of Datawarehouse	Good to use for datawarehouse core to simplify complex relationships (many:many)	Good for data marts with simple relationships (1:1 or 1:many)
Joins	Higher number of Joins	Fewer Joins
Dimension table	It may have more than one dimension table for each dimension	Contains only single dimension table for each dimension
When to use	When the dimension table is relatively big in size, snowflaking is better as it reduces space.	When the dimension table contains less number of rows, we can go for the Star schema.
Normalization/ De-Normalization	Dimension Tables are in Normalized form but Fact Table is still in De-Normalized form	Both Dimension and Fact Tables are in De-Normalized form
Data model	Bottom up approach	Top down approach

In our use case, we have two Fact tables and it is not advisable to use Galaxy schema as it leads to complexity. In our use case we need to perform both fact and dimension level analysis. Considering the facts below, choosing a star schema is efficient.

1. Our dataset contains huge volumes of facts and less volume of dimensions
2. We need to perform more fact level analysis
3. Fast query performance

To achieve a star schema, two fact tables need to be aggregated into one by keeping only common attributes.



What If I need to track the price history of books

Slowly Changing Dimensions

Data warehouse is used to analyze historical data, hence it is essential to store the different states of data. A dimension in a database is any record which provides information about any fact data.

Dimensions could be static but sometimes the attribute data tends to change.

To capture the changes and store both historical and current data is one of the critical tasks in data warehousing.

Let's consider,
In the books table, the unit price of a book may change slowly over the period. The term slowly is used here as these dimensions are not prone to rapid changes. And there is a requirement to track the price history of books. As a data engineer we need to record all these historical changes.

For example, unitprice is changing from 999 INR to 1200 INR for Asin 345803787.

PK_Asin	BookName	PAuthorId	SAuthorId	Language	PaperBack	ItemWeight	UnitPrice	Currency
345803787	Crazy Rich Asians	1001	1780	English	544	372	999	INR
1760290785	China Rich Girlfriend	1038	6701	English	400	279	285	INR
1786091089	Rich People Problems	2479	5310	English	560	383	399	INR

[For the instructors] Other examples:

1. In Amazon.com, Customer changes home address, Default Payment details
2. In Hr Data Warehouse, update the employee's manager name, employee Position

The concept of Slowly Changing Dimensions (SCD) is crucial to designing real-world datasets. SCD allows us to model such updates to dimensions in multiple ways, each with its pros and cons. In this lecture we will delve a level deeper into data modeling and utilize these SCD techniques to handle the use case we just discussed above.

There are many approaches to deal with SCD.

SCD Type 0:The passive method

No change is captured. In Type 0 method, original data would be retained.

- In this method no special action is performed upon dimensional changes. Some dimension data can remain the same as it was first time inserted, others may be overwritten.
- Pros: require no extra considerations for designing data model
- Cons: provides no control over recording and analyzing historical data and state changes

After applying Type 0, no changes will be noticed in final table for the above given problem

PK_Asin	BookName	PAuthorId	SAuthorId	Language	PaperBack	ItemWeight	UnitPrice	Currency
345803787	Crazy Rich Asians	1001	1780	English	544	372	999	INR
1760290785	China Rich Girlfriend	1038	6701	English	400	279	285	INR
1786091089	Rich People Problems	2479	5310	English	560	383	399	INR

SCD Type 1: Overwriting the old value

When the attribute value is changed, the value will be overwritten and the old value will not be stored in the table.

- SCD type 1 methodology is implemented where there is no need to store historical data in the Dimension table. This method overwrites the old data in the dimension table with the new data.
- In this method no history of dimension changes is kept in the database. The old dimension value is simply overwritten by the new one. This type is easy to maintain and is often used for data which changes are caused by processing corrections(e.g. removal of special characters, correcting spelling errors).
- Pros:
    - Straightforward to implement
    - We can ignore previous values.
    - Uses less disk space since only one current record exists.
- Cons:
    - Stores no historical data for analysis.
    - Hard to find previous states before reaching current one.

Type 1 addresses the problem as below but change history would not be tracked.

PK_Asin	BookName	PAuthorId	SAuthorId	Language	PaperBack	ItemWeight	UnitPrice	Currency
345803787	Crazy Rich Asians	1001	1780	English	544	372	1200	INR
1760290785	China Rich Girlfriend	1038	6701	English	400	279	285	INR
1786091089	Rich People Problems	2479	5310	English	560	383	399	INR

SCD Type 2: Creating a new additional record

Upon the arrival of a new value to attribute, a new row would be inserted and versions of the records will be captured. In this methodology all history of dimension changes is kept in the database. As multiple new rows would be introduced based on the dimension values, we can't have two or more records for the same PKey and to address this problem we need to introduce a surrogate key.

Surrogate Keys

Keys like alternate, unique, candidate etc. have one thing in common, they are available within the data itself, i.e. business data. These are all called natural keys.

A Surrogate Key is an artificial key which is a unique identifier for a row used by different attributes of the same entity. We can randomly generate a key or it can be a combined form of any two or more attributes of an entity. This key is made for the express usage in data modeling and finds a crucial role when it comes to SCDs particularly.

Properties of Surrogate Key

- It is system generated
- It is not manipulable by the user or the application (once decided, it cannot change)
- It has no business context and is independent of domain
- It is used only for internal data modeling and is not visible to the end users

Why do we need a surrogate key?

1. The only significance of the surrogate key is to act as the primary key.

2. The basic attribute of a primary key is that it has to be non-null and unique. The normal attributes can be identified as Primary Keys and used in the context of current business data where each primary key corresponds to an entity and that entity is stored only once in the database. So we can't use a Primary key from the business data

Surrogate Key is a system generated and acts as the primary key. Surrogate key facilitates capturing all states of the data.
We capture data change by adding a new row with a new surrogate key to the dimension table.
In the PrintBooks dimension table, Asin is the primary key. We can generate surrogate keys using the identity column.

In addition to Surrogate Keys, we need below additional fields
> Flag column to identify the current records.
> ValidFrom - denotes effective date of dimension
> ValidTo - denotes expiry date of dimension

SCD Type 2 is the most commonly used method in data warehousing

- Pros:
  - Unlimited history is preserved for each insert.
  - We can store more information as to the date ranges of various states.
  - Original table schema does not change
- Cons:
  - Introducing changes to the dimensional model in type 2 could be very expensive database operation so it is not recommended to use it in dimensions where a new attribute could be added in the future.
  - It takes up more disk space since we are storing additional fields
  - Schema design must be decided at the time of data modeling to avoid expensive data backfills and model changes

For the given problem, SCD Type 2 is the best method since we can track all price history as both historical and current data is stored

PK_Skey	Asin	BookName	PAuthorId	SAuthorId	Language	PaperBack	ItemWeight	UnitPrice	Currency	CurrentFlag	ValidFrom	ValidTo
1	345803787	Crazy Rich Asians	1001	1780	English	544	372	999	INR	0	1/1/2019	1/1/2022
2	1760290785	China Rich Girlfriend	1038	6701	English	400	279	285	INR	1	2/11/2019	NULL
3	1786091089	Rich People Problems	2479	5310	English	560	383	399	INR	1	20/5/2020	NULL
4	345803787	Crazy Rich Asians	1001	1780	English	544	372	1200	INR	1	1/1/2022	NULL

SCD Type 3: Adding a new column

An additional column available to capture the previous value of the attribute and the existing column preserves the current value of the record by overwriting the old value.

- Pros:
  - Suitable for tracking only the most recent state change where previous state changes do not matter (for example, a dataset on employees who left the organization and their last held designation)
- Cons:
  - It is by design limited to the number of columns designated for storing historical data, hence it is not suitable for storing full history.
  - It requires addition of new columns which changes the original table structure

Not reliable to track all the last previous values for the given use case.

Asin	BookName	PAuthorId	SAuthorId	Language	PaperBack	ItemWeight	CurrentUnitPrice	PreviousUnitPrice	Currency
345803787	Crazy Rich Asians	1001	1780	English	544	372	1200	999	INR
1760290785	China Rich Girlfriend	1038	6701	English	400	279	285	NULL	INR
1786091089	Rich People Problems	2479	5310	English	560	383	399	NULL	INR
345803787	Crazy Rich Asians	1001	1780	English	544	372	1200	NULL	INR

SCD Type 4: Using historical table

Records only the current value in the main dimension table while old records move to the history table to track changes.

- In this method a separate historical table is used to track all dimension's attribute historical changes for each of the dimensions. The 'main' dimension table keeps only the current data e.g. customer and customer_history tables.
- Pros:
  - Provides a clear distinction between historical and current data for analytical vs transactional purposes
  - It resembles how an audit table stores change data
  - Querying on non-historical data is fast since they contain only the latest record
- Cons:
  - Two tables require more thoughtful schema design at the time of initial data modeling
  - Disk space usage is highest since same record is stored in multiple tables

SCD Type 6:

Hybrid method of SCD Type 1,2,3 to track history.

- Ralph Kimball calls this method "Unpredictable Changes with Single-Version Overlay" in The Data Warehouse Toolkit (the bible for data modeling)...
- In this method to capture attribute change we add a new record as in type 2. The current_type information is overwritten with the new one as in type 1. We store the history in a historical_column as in type 3.
- Pros:
    - Stores complete picture of the historical data in one table
    - Allows us to store multiple state changes while using Type 3 to store previous state in the same record
    - We can do "as at now", "as at transaction time" or "as at a point in time" queries by changing the date filter logic
    - If we use cust_key (surrogate key) as foreign key on the fact table, this type allows us to preserve referential integrity of the database
- Cons:
    - It is more complex to implement
    - Requires multiple additional columns to get complete picture of historical state changes

Best Practises in Dimensional Modeling

1. Analyze the organization business and how the data stored in warehouse will help in business decision
2. Avoid Many to Many relationship between fact Tables and Dimension Table and Many to One relationship from Dimension Tables to Fact.
3. Capture the lowest level of detail into dimension table
4. Choosing the schema based on the business purpose and avoid choosing Galaxy schema.
5. Don’t overcomplicate data as mostly Denormalized or Partly Denormalized tables are optimal in data warehouse
6. Consider all possible business scenarios while designing a dimensional model

Assignment Questions:

1. As part of Data engineering at Netflix streaming team, you need to design a data model that provide insights on
    a. Top 10 trending content by region
    b. What is the medium to view content whether Tv or Mobile etc.,
    c. How many users viewed the content completely
    d. And much more business questions

Interview Questions:

1. When should we use snowflake schema and star schema?
2. Consider you are giving an interview in swiggy - Rider analytics team

    a. Provide examples for Dimension tables and Fact table considering rider analytics insights
    b. How do you form relationship between identified tables
    c. How would you handle if there are many fact tables
3. Consider you are giving an interview in Amazon - Design a data model with facts and dimensions that could provide insights to track how many new customers joined amazon and what is their first order and order value. (Consider the huge volume of data Amazon have)
    a. What is your approach and mention your steps to solve this problem
    b. How do you implement SCD for the given model
    c. There is an in between requirement to identify new customers who joined via amazon kindle
    d. Justify why your data model works best for the given problem