Chapter 4: Data Modeling & OLTP vs. OLAP

1. The Crucial Bridge from Data to Decisions: An Introduction to Data Modeling

In the field of data science, the journey from raw data to actionable business intelligence is not a straight line; it is a meticulously engineered process. At its core, this process requires a robust and well-defined **data model**. A data model is a strategic blueprint that defines how data is organized and connected to support a business's information needs. It is the architectural plan that transforms disparate, messy data into a clean, structured asset ready for analysis. While the concept of data modeling applies to all data systems, in this lecture, we will focus specifically on its application for **analytics and business intelligence (BI)**. This is a distinct and critical discipline, as the goals and design principles of analytical models are fundamentally different from those of transactional systems.

This lecture is positioned strategically within our course curriculum. We have previously explored the basics of data platforms and ingestion from various sources, including relational databases and APIs.[1] The data we have been ingesting resides in the raw, immutable Landing Zone and is then prepared in the Staging Zone of our data platform.[2] The data modeling techniques we will discuss today are the core of the subsequent step: creating a structured, analytics-ready dataset in the Integration Zone, which serves as the foundation for our data warehouse.[1] A well-executed data model is the single most important element in this transformation, ensuring that data is not just stored but is truly usable for decision-making.

For our lecture, we will anchor our discussion in a compelling, real-world scenario.[3] Imagine you have been hired as a Data Architect at "OmniCorp," a company that has recently acquired two smaller businesses: a digital music store called

Chinook and a food and beverage distributor named **Northwind**.[3] Both companies have been operating for years with their own distinct legacy database systems.[3] The central business challenge is to integrate the data from these two disparate systems into a single, unified data warehouse. The objective is not simply to combine the data, but to enable seamless

cross-business analytics and support strategic decision-making that was previously impossible.[3]

The initial analysis of the two legacy systems reveals that while their business domains are different, there are common entities: customers, employees, products (tracks in Chinook, products in Northwind), and sales transactions.[3] This commonality is what will form the basis of our unified data model.[3] However, a deeper look reveals inherent technical challenges. The naming conventions are different (e.g.,

Customer vs. Customers), and the data types and primary key formats may be inconsistent (e.g., a numeric CustomerID in one system and an alphanumeric one in another).[3] Our data model must not only be a logical representation of the business but also a practical solution that resolves these technical conflicts in a robust and scalable manner, preparing us for future growth, such as OmniCorp acquiring a third business.[3]

2. The Two Pillars of Data Systems: OLTP vs. OLAP

To effectively design our unified data warehouse, we must first understand the fundamental differences between the source systems we are dealing with and the target system we need to build. This distinction separates the world of operational systems from the world of analytical systems.

Online Transactional Processing (OLTP): The Engine of the Business

OLTP systems are the lifeblood of day-to-day business operations.[5] They are specifically designed to manage and process high-volume, real-time, short-term data transactions.[6] The primary purpose of an OLTP system is to execute and record transactions efficiently, reliably, and without compromising data integrity.[5]

The defining characteristics of OLTP systems are:

- **Purpose:** To manage and process real-time transactions.[6] Think of point-of-sale systems, ATM withdrawals, or online hotel bookings.[5]
- **Data Structure:** OLTP databases typically use a **normalized data model**, often adhering to Third Normal Form (3NF) to eliminate data redundancy.[6] This involves breaking down data into multiple, smaller tables, each with a specific purpose.
- **Workload:** OLTP systems are **write-heavy**, optimized for rapid INSERT, UPDATE, and

DELETE operations. This is crucial for handling the constant flow of new transactions.[6]
- **Data Volume:** The storage requirements are comparatively smaller, typically in the gigabytes (GB) range, as historical data is often archived or offloaded to other systems.[5]
- **Response Time:** Response times are measured in milliseconds, as a quick and seamless user experience is paramount.[5]
- **Intended Users:** The systems are used by frontline workers, customers, or internal applications.[5]

Our Chinook and Northwind databases are classic examples of OLTP systems.[3] They are relational databases (

all data is stored in several tables with links between the tables...) designed to handle the daily operations of a music store and a food distributor, respectively.[4] The Chinook tables, such as

Invoice and InvoiceLine, are structured to record individual sales transactions, while Northwind's Orders and Order Details serve the same purpose for food sales.[3] Each transaction involves updating a few specific rows, and the normalized schema ensures that these updates are performed quickly and without data inconsistencies.

Online Analytical Processing (OLAP): The Brain of the Business

In contrast, OLAP systems are not for running the business; they are for **understanding** the business.[5] Their design is optimized for complex, multi-dimensional analysis of large volumes of data.[6] The primary purpose of an OLAP system is to provide insights for strategic decision-making through activities like reporting, trend analysis, and predictive modeling.[6]

The key characteristics of OLAP systems are:

- **Purpose:** To analyze vast quantities of data to support business decisions.[6]
- **Data Structure:** OLAP databases use a **denormalized data model**, such as a Star Schema or Snowflake Schema.[6] This structure prioritizes query performance and ease of use for analysts by minimizing the number of joins required to retrieve data.
- **Workload:** OLAP systems are **read-heavy**, optimized for large, complex SELECT queries that aggregate data across many dimensions.[6]
- **Data Volume:** They have massive storage requirements, often in the terabytes (TB) or petabytes (PB) range, as they store historical and aggregated data from multiple sources.[5]
- **Response Time:** Response times are longer, ranging from seconds to minutes or even hours, depending on the complexity of the query.[5]
- **Intended Users:** These systems are used by data scientists, business analysts, and

executives who need to gain a deeper understanding of trends and performance.[5]

The Foundational Difference: Why We Separate the Two

A crucial principle in modern data architecture is the separation of OLTP and OLAP systems. This is not merely a technical choice but a strategic one. A common query in data science is why we can't simply run analytical reports directly on our live OLTP databases. There are two primary reasons why this approach is highly problematic.

First, there is a fundamental performance conflict.[6] A resource-intensive analytical query (e.g.,

What were our total sales by quarter and product category?) would require scanning and joining millions of rows across multiple tables. Running such a query on a live OLTP system would lock tables, consume significant processing resources, and could potentially slow down or even crash the real-time transactional operations of the business. This is an unacceptable risk.[6]

Second, the normalized schema of an OLTP system is highly inefficient for analytical queries.[5] To answer a seemingly simple question like

What were the total sales by employee and product genre?, a query would need to perform complex joins across tables like Invoice, InvoiceLine, Employee, Track, and Genre. An OLAP system, with its denormalized structure, simplifies this into a single, high-performance join.[3]

This critical distinction is summarized in the following table:

Criteria	OLAP	OLTP
Purpose	Analyze data to support decision-making	Manage and process real-time transactions
Data Source	Historical and aggregated data from multiple sources	Real-time and transactional data from a single source
Data Model	Denormalized (Star Schema, Snowflake Schema)	Normalized (3NF)

Data Structure	Multi-dimensional (cubes) or relational	Relational databases
Data Volume	Large (TB, PB)	Comparatively smaller (GB)
Response Time	Longer (seconds to minutes)	Shorter (milliseconds)
Intended Users	Data scientists, business analysts, executives	Front-line staff, customers
Example	Data warehouse for BI reports	Chinook and Northwind databases

The separation of these systems allows each to perform its intended function optimally. The OLTP system efficiently runs the business, while the OLAP system, a **data warehouse** or **data mart**, helps us derive value from the data and understand the business.[2]

3. Dimensional Modeling: A New Paradigm for Analytics

Dimensional modeling is a logical design technique for data warehouses that presents data in a standard, intuitive framework to support high-performance analytical queries.[6] Unlike the relational modeling used in OLTP systems, which focuses on data integrity and eliminating redundancy, dimensional modeling prioritizes query performance and ease of use for business analysts. It is a specific type of denormalized schema that strikes a perfect balance between simplicity and power.

The most common and widely adopted form of dimensional modeling is the **Kimball Star Schema**, championed by Ralph Kimball, a pioneer in the data warehousing industry.[6] It gets its name from its simple, elegant structure: a central

fact table surrounded by a set of **dimension tables**, resembling a star.[3]

The star schema is a significantly better fit for our OmniCorp analytical problem than a fully normalized schema for several reasons:

1. **Query Simplicity:** It makes it dramatically easier for business analysts to write queries. Instead of needing a deep understanding of complex joins across dozens of highly

normalized tables, an analyst can perform most queries by joining a central fact table to a few relevant dimension tables.[3] This empowers business users to perform self-service analytics, which is a key goal of our project.[3]

2. **Query Performance:** The denormalized nature of the star schema reduces the number of joins required to retrieve data. Fewer joins mean faster query execution, which is critical when dealing with the massive data volumes typical of an OLAP system.[3]
3. **Scalability and Flexibility:** The star schema is inherently flexible and easily extensible.[3] If OmniCorp were to acquire a third business in the future, we could simply ingest the new data, transform it, and load it into our existing dimensions and fact tables. We would not need to restructure the entire data model, as the addition would simply be another data source for our
   FactSales table, potentially with a new entry in a DimSourceSystem dimension.[3] This adaptability is a foundational principle of a robust data engineering solution.

The proposed schema for our OmniCorp project visualizes this concept perfectly: a central FactSales table at the core, with five surrounding dimensions tables: DimCustomer, DimEmployee, DimProduct, DimTime, and DimSourceSystem.[3] This structure will unify the disparate data from Chinook and Northwind into a single, cohesive, and easily queryable data model.[3]

4. Anatomy of the Star Schema: Facts, Dimensions, and Grain

Every dimensional model is composed of two primary components that work in tandem to provide both a quantitative record of business events and the qualitative context needed for analysis.

Fact Tables: The "What" of the Business

The fact table is the central hub of the star schema and contains the quantitative metrics, or **measures**, of a business process.[2] A fact is an event or a transaction (e.g., a sale, a temperature reading), and the measures are the numeric attributes of that event that we can aggregate or analyze.[2] Fact tables are typically large, containing many rows, and are often sparse, meaning they do not contain descriptive attributes, only keys and measures.

For our OmniCorp project, the central fact table is named FactSales.[3] Its purpose is to store the core metrics of sales transactions, enabling calculations like total revenue, average sales,

and per-item profit.[3] This table is designed to represent each line item of a sales transaction from both the Chinook and Northwind databases.[3]

The FactSales table will contain:

- **Measures:** The numeric quantities we want to analyze. Our key measures are SalesQuantity (the number of items sold in a transaction) and SalesAmount (the total price for that line item).[3]
- **Foreign Keys:** These are the columns that link the fact table to the dimension tables, providing the descriptive context.[3] For
  FactSales, these foreign keys will be DateKey (linking to DimTime), CustomerID (to DimCustomer), EmployeeID (to DimEmployee), ProductID (to DimProduct), and SourceSystemID (to DimSourceSystem).[3]

Dimension Tables: The "Who, What, When, Where, Why"

Dimension tables provide the descriptive context for the facts in the fact table. They answer the crucial "who, what, where, when, and how" questions about a business event.[2] A dimension is a collection of related, descriptive attributes used to "slice" and "dice" the measures. Dimension tables are typically much smaller than fact tables and contain non-numeric data.

Our OmniCorp project requires several key dimensions to provide a complete picture of our sales data [3]:

- DimCustomer: This dimension consolidates customer information from Chinook.Customer and Northwind.Customers.[3] It will include descriptive attributes like
  CustomerName, CompanyName, City, and Country. Its purpose is to allow us to analyze sales by customer, company, or location.[3]
- DimEmployee: This dimension stores information about the employees who handled the transactions, unifying data from Chinook.Employee and Northwind.Employees.[3] Attributes include
  EmployeeName, Title, and City. This allows for the analysis of sales performance by individual employee or by their reporting structure.[3]
- DimProduct: This dimension unifies details about the items sold, combining data from Chinook.Track and Northwind.Products.[3] It contains attributes like
  ProductName, CategoryName (e.g., 'Rock', 'Beverages'), and GenreName. Its purpose is to enable the analysis of sales by product, category, genre, or composer.[3]
- DimTime: This is a crucial temporal dimension for time-series analysis.[3] It contains granular time attributes like

DateKey (in YYYYMMDD format), FullDate, DayOfMonth, and Year. It allows us to analyze sales trends over time, for example, to see how much revenue was generated in a specific month or quarter.[3]

- DimSourceSystem: A simple but powerful dimension, this table distinguishes the origin of the data.[3] It contains attributes such as SourceSystemID and SourceSystemName (e.g., 'Chinook', 'Northwind').[3] This allows us to perform specialized analysis and to compare revenue directly between the two business units.[3]

It is important to recognize that the unification of these data sources presents significant data engineering challenges. For example, Chinook's CustomerID is an integer while Northwind's is a text string.[3] To create a single, unique

CustomerID in DimCustomer, a strategic approach must be taken, such as prefixing the Northwind IDs or using a hashing function to prevent primary key collisions.[3] Similarly, when consolidating product details, the

DimProduct table must handle disparate concepts like Track (Chinook) and Products (Northwind) by including all relevant attributes from both, such as Genre and Category.[3]

Understanding Grain: The Granularity of the Fact Table

The **grain** is the lowest level of detail represented in a fact table. It defines exactly what a single row in the table represents.[3] Determining the correct grain is arguably the most critical decision in dimensional modeling, as it fundamentally dictates the types of questions the data model can answer.

For our FactSales table, the chosen grain is **one row per line item per invoice or order**.[3] This means that if a customer purchased three different products in a single transaction, that single transaction would be represented by three separate rows in the

FactSales table.

To illustrate the importance of this decision, consider an alternative, and common, mistake: modeling the grain at the invoice or order level.[3] If each row represented an entire invoice, the fact table would contain the total sales amount for that invoice but would lose the detail for each individual product purchased.[3] We would no longer be able to analyze sales by product, category, or genre, and we would be unable to answer crucial business questions such as

Which products are our top sellers?.[3] The grain decision, therefore, must be made with the

end-user's analytical needs firmly in mind.

A single row in our FactSales table, based on the line-item grain, expresses a specific business event. For example: "On **January 15th, 2024**, customer **Bob Johnson** purchased **10 units** of **Chai Tea** for a total of **$180** from the **Northwind** business, with the transaction handled by employee **Nancy Davolio**".[3] This single sentence, which describes a row of data, clearly incorporates elements from five different dimensions (

DimTime, DimCustomer, DimProduct, DimSourceSystem, and DimEmployee), demonstrating the immense analytical power of the star schema.

The following table provides a more technical view of the source-to-target mapping, a crucial step in our data modeling process that connects the raw OLTP data to our final OLAP schema.[3]

Source Table	Source Column	Transformation Logic	Target Table	Target Column
Chinook.Customer	CustomerId	Use as-is, but a prefix or hash may be needed for unification.	DimCustomer	CustomerID
Northwind.Customers	CustomerID	Use as-is, but a prefix or hash may be needed for unification.	DimCustomer	CustomerID
Chinook.Customer	FirstName, LastName	Concatenate with a space.	DimCustomer	CustomerName
Northwind.Customers	CompanyName	Use as-is.	DimCustomer	CompanyName
Chinook.Invoice	InvoiceDate	Extract date and format to YYYYMMDD.	FactSales	DateKey
Northwind.Ord	Quantity	Use as-is.	FactSales	SalesQuantity

er Details				
Northwind.Order Details	UnitPrice, Quantity	Calculate UnitPrice * Quantity.	FactSales	SalesAmount
Chinook.InvoiceLine	UnitPrice, Quantity	Calculate UnitPrice * Quantity.	FactSales	SalesAmount
Chinook.Track	GenreId	Join to Chinook.Genre.	DimProduct	GenreName
Northwind.Products	CategoryId	Join to Northwind.Categories.	DimProduct	CategoryName

This mapping document is not just a technical exercise; it serves as a critical bridge between data understanding and schema design, ensuring that every piece of data from the source systems finds its correct and usable place in the unified data warehouse.

5. The Conceptual Link: From Knowledge Graphs to Star Schemas

To fully appreciate the elegance of dimensional modeling, it is valuable to connect it to a higher-level, more abstract concept from the field of knowledge representation: the distinction between the A-Box and the T-Box.[4] While some preliminary search results may incorrectly suggest this link is not available, it is a profound and foundational concept in our course materials and is directly applicable to our data modeling approach.[4]

The T-Box: The Terminological Foundation

In the world of knowledge graphs and ontology, a **T-Box** (Terminological Box) represents the **schema** or the **permanent knowledge** of a domain.[4] It defines the vocabulary, the classes,

and the properties of the entities within that domain. Think of the T-Box as the "theory" or the set of universal truths that govern the domain.[4] These statements are more permanent, hierarchical, and are used to classify and organize information.[4]

For example, a T-Box statement would be "An employee is a type of person," or "A product can be a track or a beverage".[4] This knowledge is a stable, unchanging foundation. It is the permanent structure that provides meaning and context.

The A-Box: The Assertion of Facts

An **A-Box** (Assertion Box), on the other hand, contains the specific **facts**, **events**, or **transactional data** about that domain.[4] It represents the "observation" part of the knowledge base. A-Box statements are dynamic, voluminous, and represent individual instances of the classes defined in the T-Box.[4]

An A-Box statement would be "On January 15th, 2024, customer Bob Johnson purchased 10 units of Chai Tea".[3] This is a specific observation, an event that occurred at a particular time. A-Boxes are where the high volume of daily transactional data is stored.[4]

The Profound Connection: How Dimensions are T-Boxes and Facts are A-Boxes

The central principle of dimensional modeling finds a deep conceptual parallel in this knowledge representation framework.[4]

- **Dimensions are T-Boxes:** Our dimension tables (DimCustomer, DimProduct, DimEmployee, etc.) are the **T-Boxes** of our data model. They define the terminology and descriptive attributes of our business domain.[4] The
  DimProduct table, for instance, is a T-Box that defines what a product *is* (e.g., its name, category, genre) and how it can be classified. This is the permanent, conceptual part of our data warehouse, our master data.[4]
- **Facts are A-Boxes:** Our FactSales table is the **A-Box** of our data model.[4] Every row in this table is an
  assertion or a **fact** about a specific sales event. It asserts that at a particular time, a specific customer bought a specific product for a certain amount. This is the dynamic, transactional, and voluminous part of our data warehouse.[4]

This powerful analogy shows that dimensional modeling is not just a practical design

technique; it is a form of engineering knowledge itself. By linking the A-Box (facts) to the T-Box (dimensions) via foreign keys, the star schema enables us to perform multi-hop reasoning.[4] We can go beyond simple facts to infer new, valuable knowledge. For instance, by linking

FactSales (A-Box) to DimProduct and DimTime (T-Boxes), we can infer trends like "sales of rock music are increasing in the second quarter".[3] This ability to reason about and derive new insights from data is the real magic of a well-designed data warehouse and is a skill that distinguishes a data practitioner from a data engineer.

6. Key Theories and Keywords

- **Online Transactional Processing (OLTP):** A class of software systems designed for high-volume, real-time, short-term data processing.[6] They are optimized for transactional workloads and use normalized data models to ensure data integrity and atomicity.[5] Examples include the Chinook and Northwind databases.
- **Online Analytical Processing (OLAP):** A class of software systems designed for complex, multi-dimensional analysis of large volumes of data.[6] They are optimized for read-heavy workloads and use denormalized schemas to support high-performance analytical queries.[5]
- **Data Modeling:** The process of creating a visual blueprint or logical representation of a database or data system. It defines how data is organized and connected to support specific business processes or analytical needs.[3]
- **Dimensional Modeling:** A logical design technique for data warehouses. Its primary goal is to present data in a standard, intuitive framework that supports high-performance analytical queries by using a structure of facts and dimensions.[3]
- **Kimball Star Schema:** The most common form of dimensional modeling.[6] It consists of a central fact table surrounded by and linked to a set of dimension tables, forming a star-like structure.[3]
- **Fact Table:** The central table in a star schema. It contains the quantitative measures of a business event and foreign keys that link it to the dimension tables. It is typically a very large and highly voluminous table.[3]
- **Dimension Table:** Tables that provide the descriptive context for the facts in a fact table.[3] They contain the attributes used to slice, dice, and filter data, answering the "who, what, where, when, and how" questions about an event.[3]
- **Fact:** A business event or transaction that occurred. Examples include a sale, a temperature reading, or a click on a website.[3]
- **Measure:** A quantitative attribute of a fact that can be aggregated or analyzed. Examples include SalesAmount, SalesQuantity, or DefectRate.[2]

- **Dimension:** A collection of related, descriptive attributes. Examples from the Chinook/Northwind case study include customer, product, and time.[3]
- **Grain:** The lowest level of detail represented in a fact table. It is the most important design decision in dimensional modeling, as it determines the specific level of granularity for analysis (e.g., one row per line item vs. one row per order).[3]
- **T-Box (Terminological Box):** A component of a knowledge graph or ontology that represents the permanent schema, vocabulary, and hierarchies of a domain.[4] Our dimension tables are the T-Boxes of a star schema.[4]
- **A-Box (Assertion Box):** A component of a knowledge graph or ontology that contains the specific, dynamic facts and events about a domain.[4] Our fact tables are the A-Boxes of a star schema.[4]

Works cited

1. dsi310_แบบฟอร์มเค้าโครงการบรรยาย 2025
2. dsi310 week01: Data Engineering & Data Lake
3. dsi310: Data Lake Modeling
4. 2025 dsi310 week02
5. OLTP vs. OLAP: Differences and Applications - Snowflake, accessed September 10, 2025, https://www.snowflake.com/en/fundamentals/olap-vs-oltp-the-differences/
6. OLTP vs OLAP - Difference Between Data Processing Systems - AWS, accessed September 10, 2025, https://aws.amazon.com/compare/the-difference-between-olap-and-oltp/
7. lerocha/chinook-database: Sample database for SQL Server, Oracle, MySQL, PostgreSQL, SQLite, DB2 - GitHub, accessed September 10, 2025, https://github.com/lerocha/chinook-database
8. Abox - Wikipedia, accessed September 10, 2025, https://en.wikipedia.org/wiki/Abox
9. Knowledge Graph Modeling: Governance & Project Scope - Oracle Blogs, accessed September 10, 2025, https://blogs.oracle.com/ateam/post/knowledge-graph-modeling-governance-project-scope