What is data modeling?

Discover how data modeling uses abstraction to represent and better understand the nature of data flow within an enterprise information system

Discover watsonx.ai →

Your privacy choices

Let’s talk

Take the next step

What is data modeling?

Data modeling is the process of creating a visual representation of either a whole information system or parts of it to communicate connections between data points and structures. The goal is to illustrate the types of data used and stored within the system, the relationships among these data types, the ways the data can be grouped and organized and its formats and attributes.

Data models are built around business needs. Rules and requirements are defined upfront through feedback from business stakeholders so they can be incorporated into the design of a new system or adapted in the iteration of an existing one.

Data can be modeled at various levels of abstraction. The process begins by collecting information about business requirements from stakeholders and end users. These business rules are then translated into data structures to formulate a concrete database design. A data model can be compared to a roadmap, an architect's blueprint or any formal diagram that facilitates a deeper understanding of what is being designed.

Data modeling employs standardized schemas and formal techniques. This provides a common, consistent, and predictable way of defining and managing data resources across an organization, or even beyond.

Ideally, data models are living documents that evolve along with changing business needs. They play an important role in supporting business processes and planning IT architecture and strategy. Data models can be shared with vendors, partners, and/or industry peers.

Let's talk

Related content

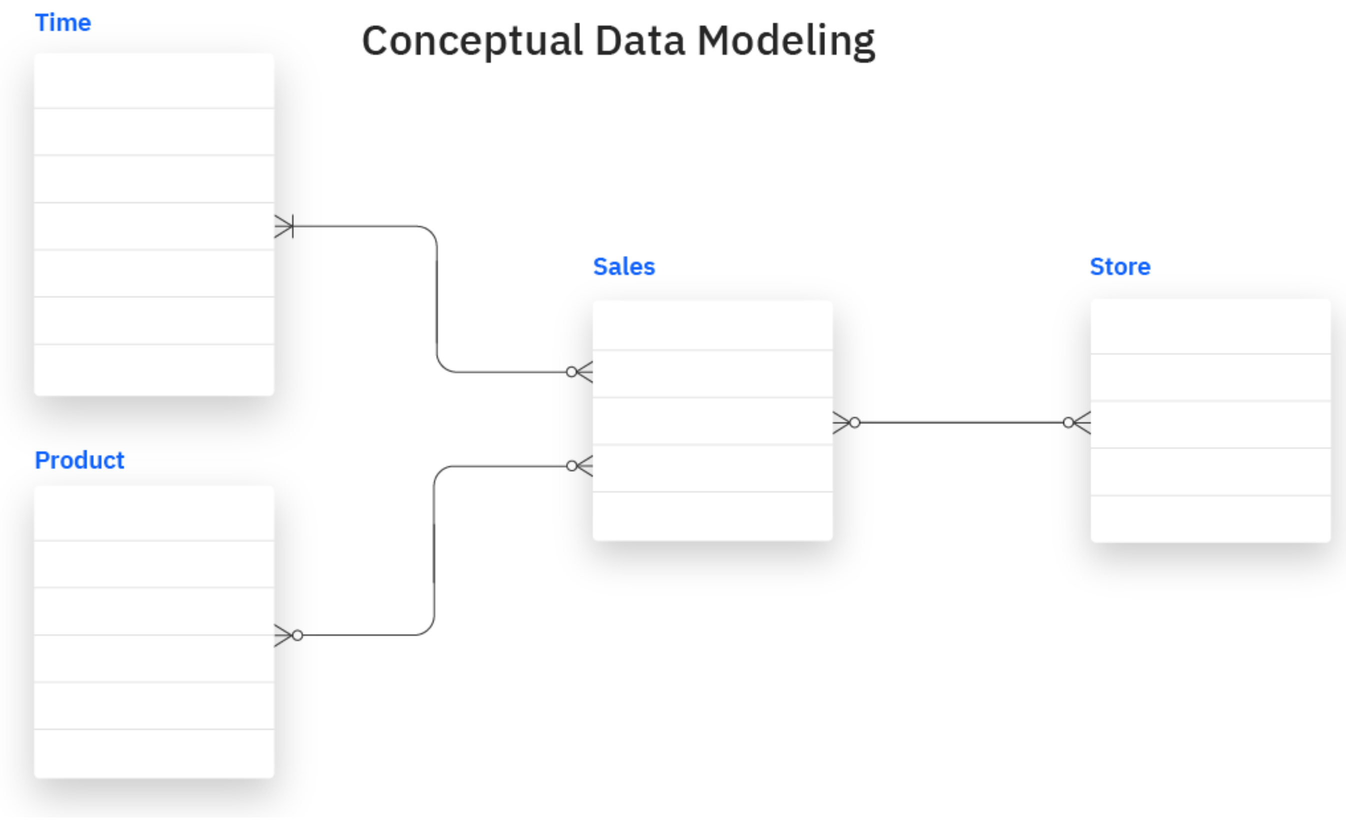
Explore data consulting services →

Types of data models

Like any design process, database and information system design begins at a high level of abstraction and becomes increasingly more concrete and specific. Data models can generally be divided into three categories, which vary according to their degree of abstraction. The process will start with a conceptual model, progress to a logical model and conclude with a physical model. Each type of data model is discussed in more detail in subsequent sections:

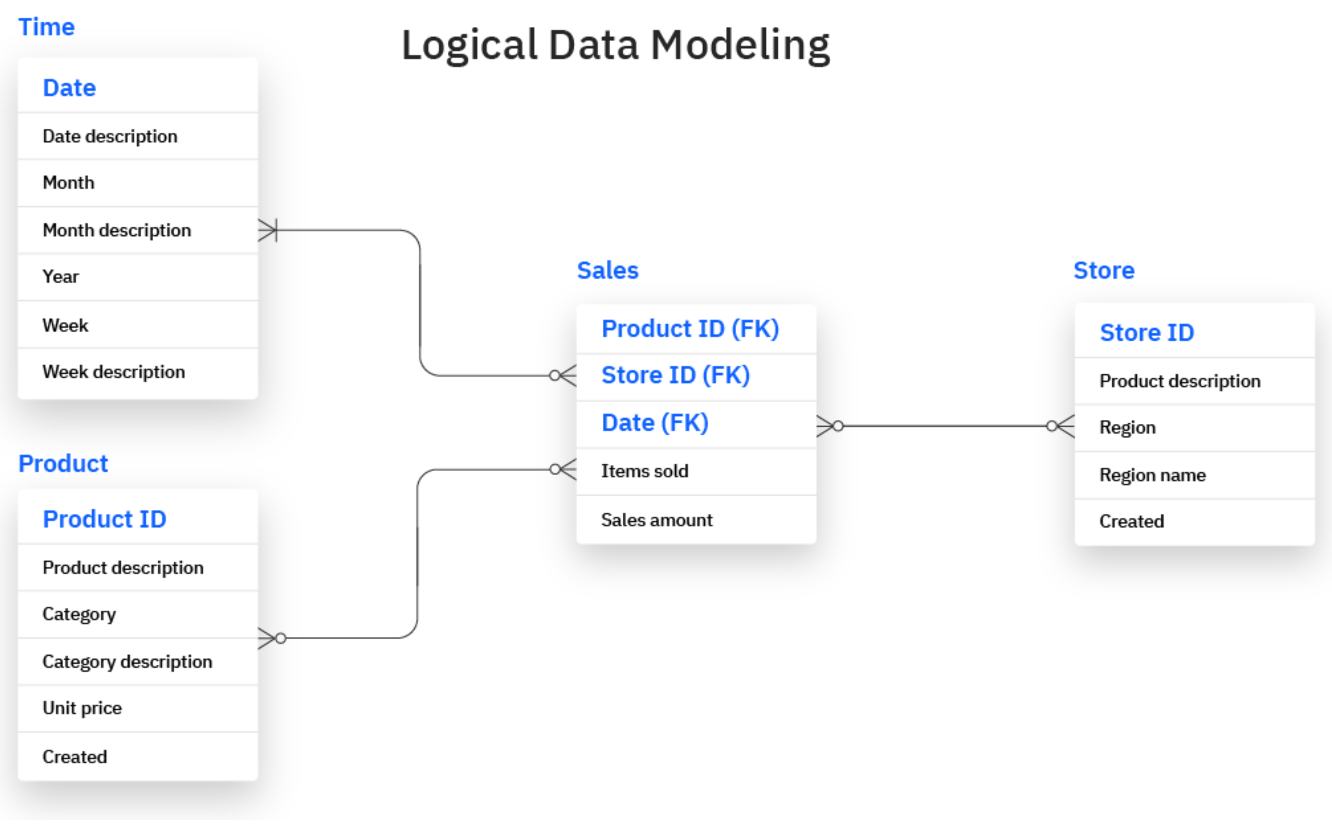
Conceptual data models

They are also referred to as domain models and offer a big-picture view of what the system will contain, how it will be organized, and which business rules are involved. Conceptual models are usually created as part of the process of gathering initial project requirements. Typically, they include entity classes (defining the types of things that are important for the business to represent in the data model), their characteristics and constraints, the relationships between them and relevant security and data integrity requirements. Any notation is typically simple.



L odels

They are less abstract and provide greater detail about the concepts and relationships in the domain under consideration. One of several formal data modeling notation systems is followed. These indicate data attributes, such as data types and their corresponding lengths, and show the relationships among entities. Logical data models don't specify any technical system requirements. This stage is frequently omitted in agile or DevOps practices. Logical data models can be useful in highly procedural implementation environments, or for projects that are data-oriented by nature, such as data warehouse design or reporting system development.



Physical data models

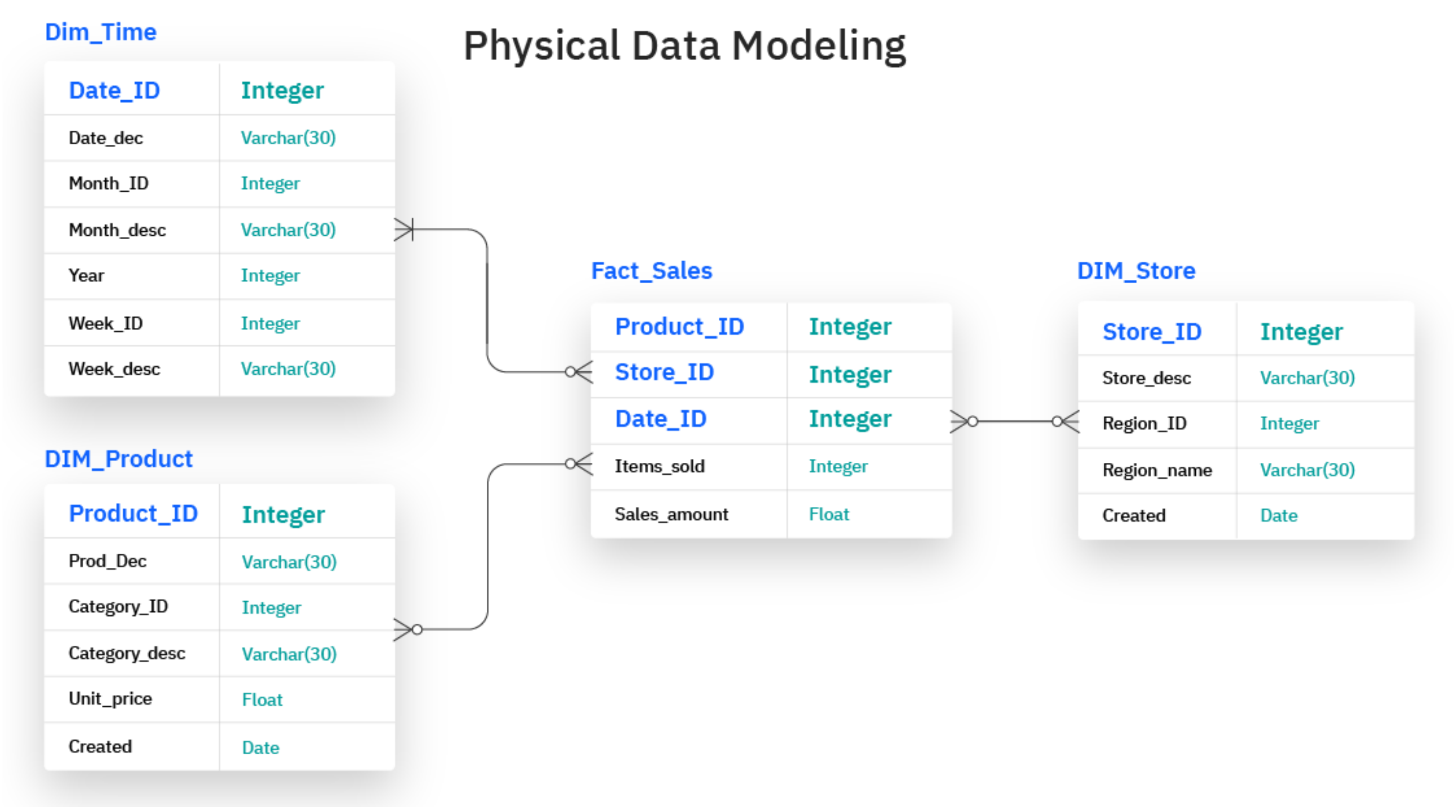
They provide a schema for how the data will be physically stored within a database. As such, they're the least abstract of all. They offer a finalized design that can be implem

a relational database, including associative tables that illustrate the relationships

Your privacy choices

Let's talk

entities as well as the primary keys and foreign keys that will be used to maintain those relationships. Physical data models can include database management system (DBMS)-specific properties, including performance tuning.



Data modeling process

As a discipline, data modeling invites stakeholders to evaluate data processing and storage in painstaking detail. Data modeling techniques have different conventions that dictate which symbols are used to represent the data, how models are laid out, and how business requirements are conveyed. All approaches provide formalized workflows that include a sequence of tasks to be performed in an iterative manner. Those workflows generally look like this:

1. **Identify the entities.** The process of data modeling begins with the identification of the things, events or concepts that are represented in the data set that is to be modeled. Each entity should be cohesive and logically discrete from all others.

Your privacy choices

Let's talk

2. **Identify key properties of each entity.** Each entity type can be differentiated from all others because it has one or more unique properties, called attributes. For instance, an entity called “customer” might possess such attributes as a first name, last name, telephone number and salutation, while an entity called “address” might include a street name and number, a city, state, country and zip code.
3. **Identify relationships among entities.** The earliest draft of a data model will specify the nature of the relationships each entity has with the others. In the above example, each customer “lives at” an address. If that model were expanded to include an entity called “orders,” each order would be shipped to and billed to an address as well. These relationships are usually documented via unified modeling language (UML).
4. **Map attributes to entities completely.** This will ensure the model reflects how the business will use the data. Several formal data modeling patterns are in widespread use. Object-oriented developers often apply analysis patterns or design patterns, while stakeholders from other business domains may turn to other patterns.
5. **Assign keys as needed, and decide on a degree of normalization that balances the need to reduce redundancy with performance requirements.** Normalization is a technique for organizing data models (and the databases they represent) in which numerical identifiers, called keys, are assigned to groups of data to represent relationships between them without repeating the data. For instance, if customers are each assigned a key, that key can be linked to both their address and their order history without having to repeat this information in the table of customer names. Normalization tends to reduce the amount of storage space a database will require, but it can at cost to query performance.
6. **Finalize and validate the data model.** Data modeling is an iterative process that should be repeated and refined as business needs change.

Types of data modeling

Data modeling has evolved alongside database management systems, with model types increasing in complexity as businesses' data storage needs have grown. Here are several model types:

- **Hierarchical data models** represent one-to-many relationships in a treelike format. In this type of model, each record has a single root or parent which maps to one or
tables. This model was implemented in the IBM Information Management Sy

which was introduced in 1966 and rapidly found widespread use, especially in banking. Though this approach is less efficient than more recently developed database models, it's still used in Extensible Markup Language (XML) systems and geographic information systems (GISs).

- **Relational data models** were initially proposed by IBM researcher E.F. Codd in 1970. They are still implemented today in the many different relational databases commonly used in enterprise computing. Relational data modeling doesn't require a detailed understanding of the physical properties of the data storage being used. In it, data segments are explicitly joined through the use of tables, reducing database complexity.

Relational databases frequently employ structured query language (SQL) for data management. These databases work well for maintaining data integrity and minimizing redundancy. They're often used in point-of-sale systems, as well as for other types of transaction processing.

- **Entity-relationship (ER) data models** use formal diagrams to represent the relationships between entities in a database. Several ER modeling tools are used by data architects to create visual maps that convey database design objectives.
- **Object-oriented data models** gained traction as object-oriented programming and it became popular in the mid-1990s. The "objects" involved are abstractions of real-world entities. Objects are grouped in class hierarchies, and have associated features. Object-oriented databases can incorporate tables, but can also support more complex data relationships. This approach is employed in multimedia and hypertext databases as well as other use cases.
- **Dimensional data models** were developed by Ralph Kimball, and they were designed to optimize data retrieval speeds for analytic purposes in a data warehouse. While relational and ER models emphasize efficient storage, dimensional models increase redundancy in order to make it easier to locate information for reporting and retrieval. This modeling is typically used across OLAP systems.

Two popular dimensional data models are the star schema, in which data is organized into facts (measurable items) and dimensions (reference information), where each fact is surrounded by its associated dimensions in a star-like pattern. The other is the snowflake schema, which resembles the star schema but includes additional layers of associated dimensions, making the branching pattern more complex.

Benefits of data modeling

Data modeling makes it easier for developers, data architects, business analysts, and other stakeholders to view and understand relationships among the data in a database or data warehouse. In addition, it can:

- Reduce errors in software and database development.
- Increase consistency in documentation and system design across the enterprise.
- Improve application and database performance.
- Ease data mapping throughout the organization.
- Improve communication between developers and business intelligence teams.
- Ease and speed the process of database design at the conceptual, logical and physical levels.

Data modeling tools

Numerous commercial and open source computer-aided software engineering (CASE) solutions are widely used today, including multiple data modeling, diagramming and visualization tools. Here are several examples:

- **erwin Data Modeler** is a data modeling tool based on the Integration DEFinition for information modeling (IDEF1X) data modeling language that now supports other notation methodologies, including a dimensional approach.
- **Enterprise Architect** is a visual modeling and design tool that supports the modeling of enterprise information systems and architectures as well as software applications and databases. It's based on object-oriented languages and standards.
- **ER/Studio** is database design software that's compatible with several of today's most popular database management systems. It supports both relational and dimensional data modeling.
- **Free data modeling tools** include open source solutions such as Open ModelSphere.

Related solutions