Unit 5 Syllabus

- Graph Database: Introduction to graphs, Data modelling process, understanding the problem, developing the whiteboard model, constructing the logical data model, checking our model, Mutating a graph, creating vertices and edges, adding edges, removing a vertex, removing an edge

Introduction to graphs

- When you look at a road map, examine an organizational chart, or use social networks such as Facebook, LinkedIn, or Twitter, you use a graph.
- Graphs are a nearly ubiquitous way to think about real-world scenarios as these abstract out the items and the relationships being represented, and this abstraction allows for quick and efficient processing of the connections within the data.

Example of Graph Databases

What is a graph?

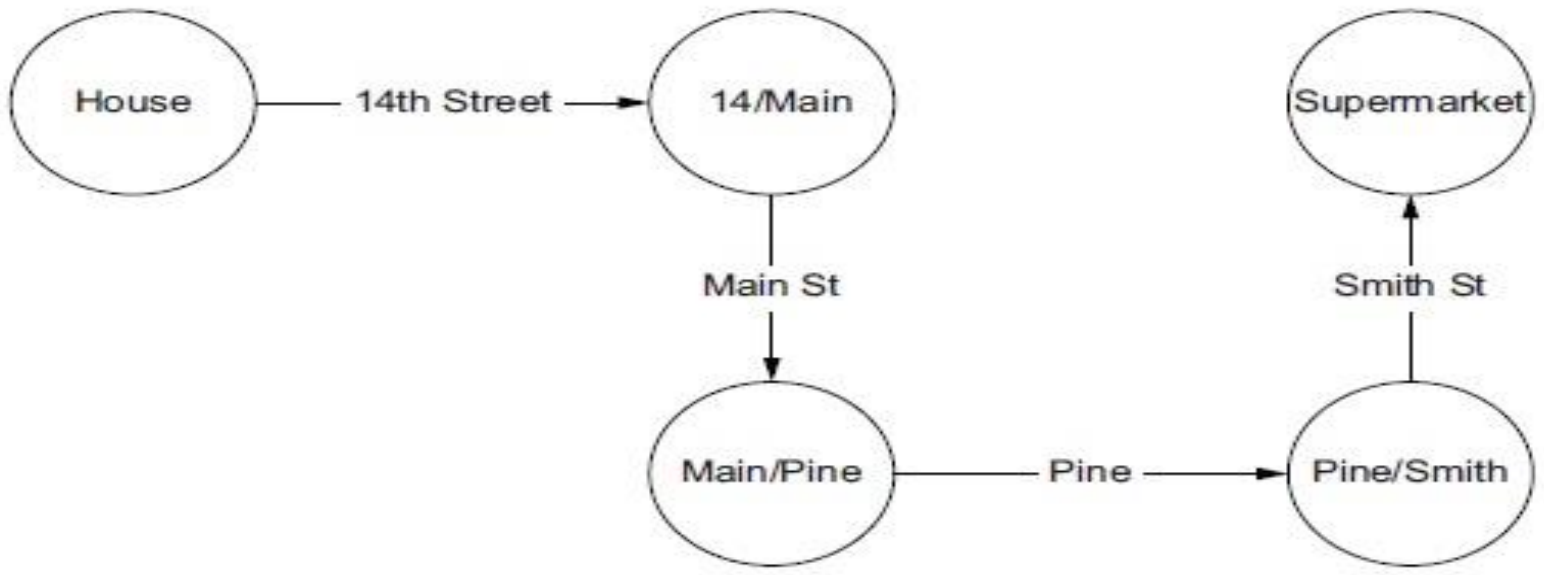

Figure 1.1 A graph representing directions to the supermarket

Terminology

- *Graph*—A set of vertices (singular, vertex) and edges
- *Vertex*—A point in a graph where zero or more edges meet, also referred to as a node or an entity
- *Edge*—A relationship between two vertices within a graph, sometimes called a relationship, link, or connection

What is a graph database?

- A graph database is a data-storage engine that combines the basic graph structures of vertices and edges with a persistence technology and a traversal (query) language to create a database optimized for storage and fast retrieval of highly connected data.
- Unlike other database technologies, graph databases are built on the concept that the relationships between entities are as or more important than the entities within the data.
- Because entities and relationships are treated with equal importance in a graph database.

What is a graph database?

- Graph databases provide excellent tools for moving through relationships in our data.
- By making the connections (edges) as important as the items, the edges connect to (vertices), graph databases represent these associations as full-fledged constructs of the database that can be easily observed and manipulated.
- This ability to store rich relationships is one of the main reasons that graph databases are better suited to handling complex linked-data use cases. In developer parlance, we might say that edges are “first-class citizens” just like the vertices

Comparison with other types of databases

- *Key-value*—Represents all data by a unique identifier (a key) and an associated data object (the value). Examples include Berkeley DB, RocksDB, Redis, and Memcached.
- *Wide-column (or column-oriented)*—Stores data in rows with a potentially large number or possibly varying numbers of columns in each row. Examples include Apache HBase, Azure Table Storage, Apache Cassandra, and Google Cloud Bigtable.

Comparison with other types of databases

- *Document*—Stores data in a uniquely keyed document that can have varying schema and that can contain nested data. Examples include MongoDB and Apache CouchDB.
- *Relational*—Stores data in tables containing rows with strict schema. Relationships can be established between tables allowing the joining of rows. Examples include PostgreSQL, Oracle Database, and Microsoft SQL Server.
- *Graph*—Stores data as vertices (nodes, components) and edges (relationships). Examples include Neo4j, Apache TinkerPop's Gremlin Server, JanusGraph, and TigerGraph.

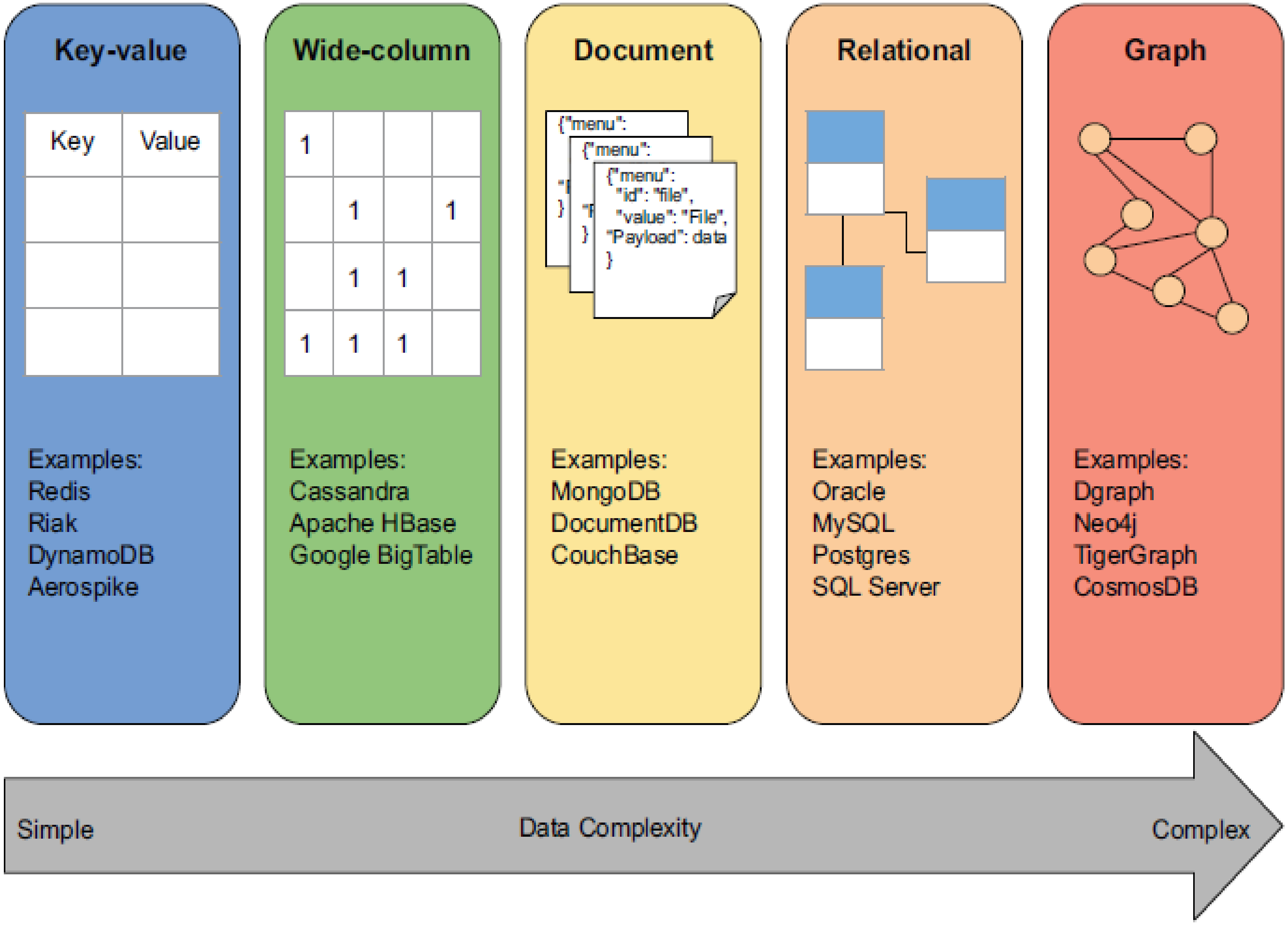

Figure 1.3 **Database engine types ordered by data complexity**

Data modelling process

- Data modeling is the process of translating real-world entities and relationships into equivalent software representations. The extent to which we achieve accurate software representations of these real-world items dictates how well we address the intended problem.
- In relational database applications, the process of data modeling is about translating certain real-world problems, understandings, and questions into software, usually focusing on creating a technical implementation involving a database

Data modelling process

- This includes identifying and understanding the problem, determining the entities and relationships in that problem, and then creating a representation of that problem in the database.
- The graph data modeling process is largely the same.
- The main difference is that we must shift from an “entity first” mindset (or perhaps more accurately, an “entity-only” mindset) to an “entity and relationship” mindset

Data Modelling Terms

- *Entity*—Commonly represented by nouns, entities describe the things or the type of things in the domain (for example, vehicles, users, or geographic locations).
- As we move from problem definition and conceptual modeling, entities often become vertices in the logical model and technical implementation.
- *Relationship*—Often represented by verbs (or verbal phrases), relationships describe how entities interact with one another. It could be something like *moves* as in “a vehicle *moves* to a location,” or *friends* as in the Facebook sense of this word as a verb (for example, “a person *friends* another person”).
- As we move from problem definition and conceptual modeling, relationships often become edges in the logical model and technical implementation

Data Modelling Terms

- *Attribute*—Like entities, also represented by a noun, but always in the context of an entity or relationship. Attributes describe a quality about the entity or relationship.
- We limit our use of attributes as we feel that these can distract from the more critical parts of the model development process.
- *Access pattern*—Describes either questions or methods of interaction in the domain.
- Examples can be questions like, Where is this vehicle going? or Who are this person's friends? As we move from problem definition and conceptual modeling, access patterns often become queries in the logical model and technical implementation.

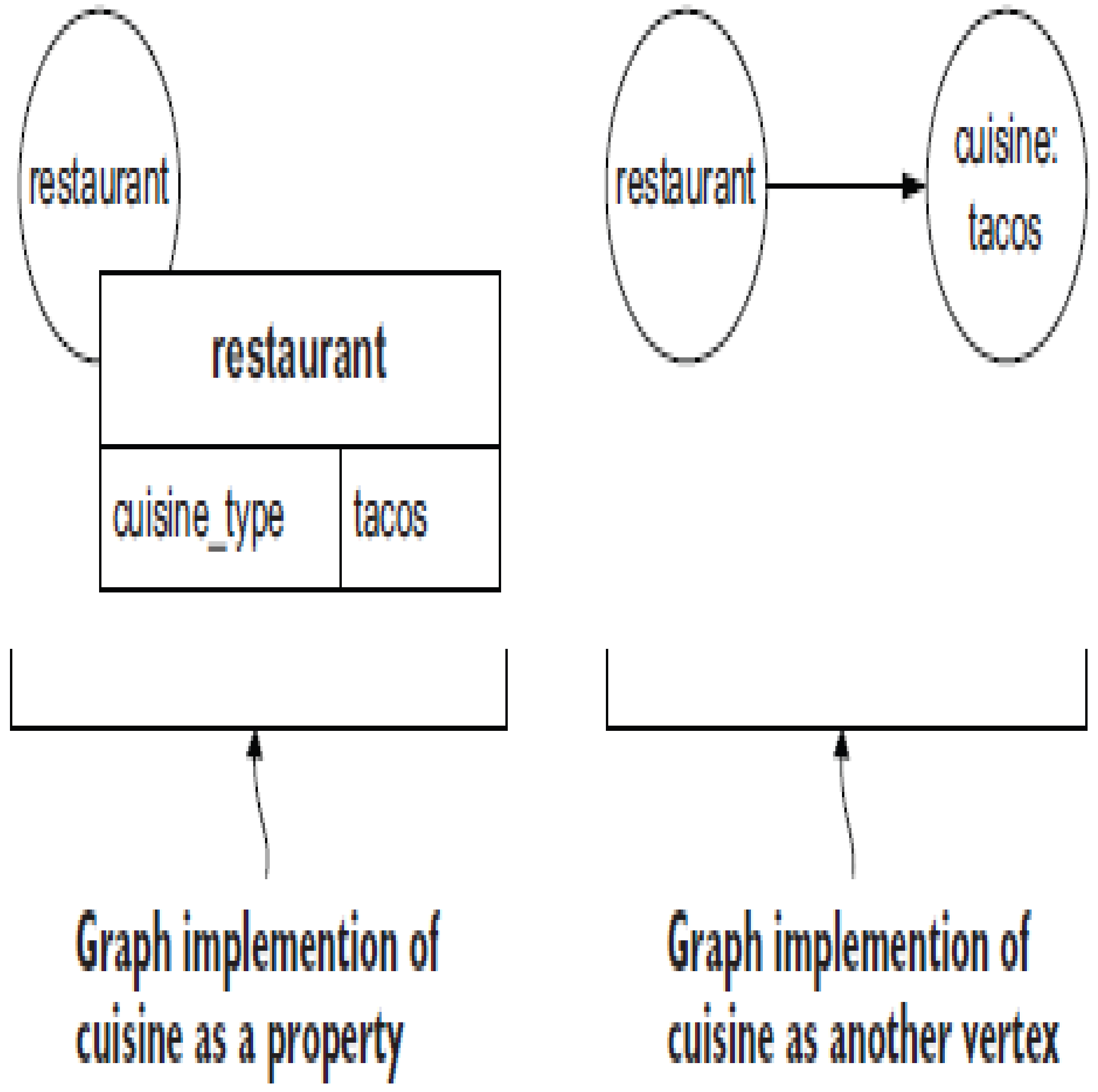
restaurant
restaurant
cuisine_type
tacos
restaurant
cuisine:
tacos
Graph implemention of cuisine as a property
Graph implemention of cuisine as another vertex

Figure 2.1 Two possible graph implementations for a restaurant's cuisine

Four-step process for data modeling

- *Understand the problem*. In domain terms and language to ensure a clear understanding of the end user's perspective. We'll explore project goals to make certain that the domain and scope of the problem is clear. At the end of this step, we'll have specified the common terms and the core access patterns of users.
- *Create a whiteboard or conceptual model*. After understanding the problem and the language used to describe it, from text to a picture, focusing on drawing a diagram that makes sense to the business users and one that's useful to the technical developers.
- We'll define the conceptual model, which includes codifying the main entities and relationships between those entities.
- After completing this step, we'll have a high-level picture of the problem domain from the business perspective.

Four-step process for data modeling

- *Create a logical data model*. In this step, combine the domain defined in the first step with the conceptual model from the second step to create the physical description of the graph data model.
- This includes defining the vertices and the edges, as well as specifying the properties on those
- Most graph databases are schemaless, so once we define this logical model, we’re ready to begin working on our queries. If your chosen graph database requires explicit schema definitions

Four-step process for data modeling

- *Test the model*. In defined problem, that the entities and relationships needed to answer our user's questions exist, and that these are properly named.
- This step focuses on validating coherence in the three previous steps, where we moved from a textual description of the domain to a simple picture of the entities and relationships and, finally, constructed our model with vertices and edges.
- This is largely a matter of asking, "Does this make sense given the other steps?" and "Did we leave anything out which we established before?"

Data modeling step	Participants	Tools	Output
1. Problem definition	Business, Developers	Domain, scope, business entities, functionality	Textual description of problem
2. Conceptual data model	Business, Developers	Entities, relationships, access patterns	Picture of entities and their relationships
3. Logical data model	Developers only	Vertices, edges, proper-ties	Diagram of graph elements
4. Test the model	Developers only	Preceding steps	Coherence of the out-puts of prior steps

Understand the problem

- Whether working in a large enterprise, a small company, or just on a side project, the first step in data modeling is understanding the problem, the domain, and the scope of the work we are addressing.
- In a large enterprise, this work may have already been done for us through some requirements document. In a small company or with personal projects, that is unlikely to be the case. In the end, it doesn't matter if our project has a requirements document or not; it is up to us to have a sufficient understanding of the problem before beginning work on our data model

Understand the problem

- While the questions vary by project, there are different categories of questions that help us gain a clear view of the problem. These categories include the following:
- Domain and scope
- Business entity
- Functionality