Contents at a Glance

Introduction

If you're preparing to take the CompTIA DataSys+ exam, you'll undoubtedly want to find as much information as you can about data and analytics. The more information you have at your disposal at the more hands-on experience you gain, the better off you'll be when attempting the exam. This study guide was written with that in mind. The goal was to provide enough information to prepare you for the test, but not so much that you'll be overloaded with information that's outside of the scope of the exam. We've included review questions at the end of each chapter to give you a taste of what it's like to take the exam. If you're already working in the data field, we recommend that you check out these questions first to gauge your level of expertise. You can then use the book mainly to fill in the gaps in your current knowledge. This study guide will help round out your knowledge base before taking the exam. If you can answer 90 percent or more of the review questions correctly for a given chapter, you can feel safe moving onto the next chapter. If you're unable to answer that many correctly, reread the chapter and try the questions again. Your score should improve.

The DataSys+ Exam

The DataSys+ exam is designed to be a vendor-neutral certification for data systems professionals and those seeking to enter the field. CompTIA recommends this certification for those currently working, or aspiring to work, in data systems and database administration roles.

The exam covers five major domains:

1. **Database Fundamentals**
2. **Database Deployment**
3. **Database Management and Maintenance**
4. **Data and Database Security**
5. **Business Continuity**

These five areas include a range of topics, from Structured Query Language (SQL) to scripting and from data security to business continuity, while focusing heavily on scenario-based learning. That's why CompTIA recommends those attempting the exam have 2-3 years of hands-on work experience, although many individuals pass the exam before moving into their first database administration role.

The DataSys+ exam is conducted in a format that CompTIA calls "performance-based assessment." This means the exam combines standard multiple-choice questions with other, interactive question formats. Your exam may include several types of questions, such as multiple-choice, fill-in-the-blank, multiple-response, drag-and-drop, and image-based problems. The exam costs $358 in the United States, with roughly equivalent prices in other locations around the globe. More details about the DataSys+ exam and how to take it can be found here:

www.comptia.org/certifications/datasys

You'll have 90 minutes to take the exam and will be asked to answer up to 90 questions during that time period. Your exam will be scored on a scale ranging from 100 to 900, with a passing score of 700.

In-Person Exams

CompTIA partners with Pearson VUE's testing centers, so your next step will be to locate a testing center near you. In the United States, you can do this based on your address or your zip code, while non-U.S. test centers may find it easier to enter their city and country. You can search for a test center near you at the Pearson Vue website, where you will need to navigate to "Find a test center".

www.pearsonvue.com/comptia

Now that you know where you'd like to take the exam, simply set up a Pearson VUE testing account and schedule an exam on their site. On the day of the test, take two forms of identification, and make sure to show up with plenty of time before the exam starts. Remember that you will not be able to take your notes, electronic devices (including smartphones and watches), or other materials in with you.

Be sure to review the Candidate identification policy at www.comptia.org/testing/testing-policies-procedures/test-policies/candidate-id-policy to learn what types of ID are acceptable.

After the Data Systems Exam

Once you have taken the exam, you'll be notified of the score immediately, so you'll know if you passed the test right away. You should keep track of your score report with your exam registration records and the email address you used to register for the exam.

Glossary A glossary of key terms from this book is available as a fully searchable PDF.

Exam DS0-001 Exam Objectives

CompTIA goes to great lengths to ensure that its certification programs accurately reflect the IT industry's best practices. They do this by establishing committees for each of its exam programs. Each committee comprises a small group of IT professionals, training providers, and publishers who are responsible for establishing the exam's baseline competency level and who determine the appropriate target-audience level.

Once these factors are determined, CompTIA shares this information with a group of hand-selected subject matter experts (SMEs). These folks are the true brainpower behind the certification program. The SMEs review the committee's findings, refine them, and shape them into the objectives that follow this section. CompTIA calls this process a *job-task analysis (JTA).* Finally, CompTIA conducts a survey to ensure that the weightings and objectives truly reflect job requirements. Only then can the SMEs go to work writing the hundreds of questions needed for the exam. Even so, they have to go back to the drawing board for further refinements in many cases before the exam is ready to go live in its final state. Rest assured that the content you're about to learn will serve you long after you take the exam. CompTIA also publishes relative weightings for each of the exam's objectives. The following table lists the five DataSys+ objective domains and the extent to which they are represented on the exam:

Domain, % of Exam

1.0 Database Fundamentals 24%
2.0 Database Deployment 16%
3.0 Database Management and Maintenance 25%
4.0 Data and Database Security 23%

5.0 Business Continuity 12%

DS0-001 Certification Exam Objective Map

Objective, Chapter

1.0 Database Fundamentals

1.1 Compare and Contrast database structure types - … Chapter 2
1.2 Given a scenario, develop, modify, and run SQL code - … Chapter 3
1.3 Compare and contrast scripting methods and scripting environments - … Chapter 3
1.3 Explain the impact of programming on database operations - … Chapter 3

2.0 Database Deployment

2.1 Compare and contrast aspects of database planning and design - … Chapter 4
2.2 Explain database implementation, testing, and deployment phases - … Chapter 4

3.0 Database Management and Maintenance

3.1 Explain the purpose of monitoring and reporting for database management and performance - … Chapter 5

3.2 Explain common database maintenance processes - … Chapter 5
3.3 Given a scenario, produce documentation and use relevant tools - … Chapter 5
3.4 Given a scenario, implement data management tasks - … Chapter 5

4.0 Data and Database Security

4.1 Explain data security concepts - … Chapter 6
4.2 Explain the purpose of governance and regulatory compliance - … Chapter 6
4.3 Given a scenario, implement policies and best practices related to authentication and authorization - … Chapter 6
4.4 Explain the purpose of database infrastructure security - … Chapter 7
4.5 Describe types of attacks and their effects on data systems - … Chapter 7

5.0 Business Continuity

5.1 Explain the importance of disaster recovery and relevant techniques - … Chapter 8
5.2 Explain backup and restore best practices and processes - … Chapter 8

NOTE Exam objectives are subject to change at any time without prior notice and at CompTIA's discretion. Please visit CompTIA's website (www.comptia.org) for the most current listing of exam objectives.

How to Contact the Publisher

If you believe you've found a mistake in this book, please bring it to our attention. At John Wiley & Sons, we understand how important it is to provide our customers with accurate content, but even with our best efforts an error may occur. To submit your possible errata, please mail it to our Customer Service Team at wileysupport@wiley.com with the subject line "Possible Book Errata Submission."

Assessment Test

1. Kathleen, a data systems analyst at a midsize tech company, is monitoring and configuring the alerts for the company's cloud-based database. The company wants to manage storage resources effectively to prevent unexpected interruptions. Kathleen wants to set up a primary alert related to storage management based on a management-by-exception strategy. Which primary alert should Kathleen set up?

    a. An alert when the database size reaches 70 percent of the total storage capacity.
    b. An alert when there's an unusual surge in active users
    c. An alert when the system encounters a sudden increase in error rates
    d. An alert when the database read/write ratio significantly deviates from the normal range

2. Paige, a database administrator at a multinational company, needs to change the data type of the postal code attribute in the address table to accommodate Canadian addresses, which contain characters and numbers. Which SQL command should she use to achieve this?
    a. ALTER TABLE
    b. DROP TABLE
    c. CREATE TABLE
    d. SELECT

3. Raseel, a database administrator at a growing company, is responsible for designing and deploying a new database system. She must consider various assets and factors to ensure optimal performance and scalability while adhering to budget constraints. Which of the following factors should Rassel prioritize when acquiring hardware assets for the new database system?
    a. Storage, network bandwidth, and decorative server cases
    b. Processing capacity, memory, storage, and network bandwidth
    c. Number of database administrators, processing capacity, and usage
    d. Open-source software, processing capacity, and memory

4. Brian is conducting a disaster recovery test. During the test, he will relocate personnel to the hot site, activate the site, and simulate live operations by processing the same data at the hot site as the organization processes at the primary site. The primary site will be taken offline once the test is underway. What type of test is Brian planning?
    a. Parallel test
    b. Structured walk-through
    c. Full-interruption test
    d. Simulation

5. Tanika is a data systems analyst at an e-commerce startup working on a class diagram using UML. Which of the following best suits her needs?
    a. Microsoft Word
    b. Erwin
    c. Google Sheets
    d. Lucidchart

6. Beth's organization needs to develop policies and procedures to ensure the quality, security, privacy, and regulatory compliance of their data. She would like to identify the appropriate person to lead their data governance activities and work with stakeholders to establish policies and procedures for specific subject area domains. What role in the company would normally be responsible for developing policies and procedures for their data quality, security, privacy, and regulatory compliance, and leading their data governance activities.
    a. Data Owner
    b. Organizational data steward
    c. Subject area data steward
    d. Data custodian

7. Hassan is a data systems analyst at a midsize e-commerce company. He notices an increasing number of deadlocks in the company's central transaction processing database, impacting the overall performance. Which steps should Hassan take to prevent deadlocks from occurring frequently?
    a. Start and end transactions explicitly and minimize transaction size.
    b. Increase the number of concurrent connections to the database.
    c. Increase the size of the connection pool.
    d. Increase the time period for reaping and refreshing dead connections.

8. Jenn is a software developer working on a new flight booking application. She needs to interact with a relational database storing flight data but wants to focus on the application's business logic rather than dealing with complex SQL queries. Which technique would best suit Jennifer's requirements?
    a. Object-oriented programming (OOP)
    b. User experience (UX)
    c. SQL execution plan optimization

d. Object-relationship mapping (ORM)

9. Hamid is a data systems analyst at an established financial services company. His primary responsibility is to ensure the data system's optimal performance. After observing a constant slowdown in the database's responsiveness, he uses Oracle Enterprise Manager to identify poorly performing queries. Exploring the execution plan for a frequently executed query that supports order processing activities, Hamid sees that the query is doing a full table scan. Hamid realizes that this is causing a bottleneck and decides to take steps to improve this query's performance. Which of the following actions would be most appropriate for Hamid?
    a. Rewrite the query to use joins instead of subselects.
    b. Change the code to use bind variables.
    c. Increase the number of cache sequence values.
    d. Create an index to cover the query.

10. Caroline is a senior network administrator at a logistics company and is configuring an internal firewall to allow web servers in the perimeter network to connect to a PostgreSQL database in a private network using its default port. Which network port must she open to allow traffic to reach the database?
    a. 1521
    b. 1433
    c. 5432
    d. 50000

11. Samantha is developing an application prototype that frequently handles credit card information. She would like to deploy a security control that can detect the presence of credit card records across a variety of systems. What detection technology would be best-suited for this task?
    a. MariaDB
    b. Oracle Enterprise Edition
    c. IBM Db2 Standard
    d. Cassandra

12. John Paul works for a merchant that frequently handles credit card information. She would like to deploy a security control that can detect the presence of credit card records across a variety of systems. What detection technology would be best suited for this task?
    a. Watermarking
    b. Pattern matching
    c. Host-based
    d. Network-based

13. Omar is selecting a fire-suppression system for his organization's data center. He would like to use a technology that deploys water only at specific sprinkler heads when a fire is

detected. He hopes that this approach will limit the damage in the facility when water is deployed. What type of system would best meet his needs?

a. Wet pipe sprinkler system
b. Dry pipe sprinkler system
c. Pre-action sprinkler system
d. Deluge sprinkler system

14. Omar is a data systems analyst for a large healthcare organization regulated by the Health Insurance Portability and Accountability Act (HIPAA). The vendor has just released a new patch for the company's database software. The patch is not urgent and addresses minor software defects. Omar is wondering when the best time to apply the patch would be. What should Omar do?
    a. Apply the patch immediately in the production environment without testing.
    b. Ignore the patch since it is not urgent and addresses minor software defects.
    c. Test the patch in a nonproduction environment before scheduling a time to apply it in production.
    d. Apply the patch during peak usage hours to ensure maximum user impact.

15. Sarah is a cybersecurity analyst at an online store. She is tasked with ensuring the security of the store's web applications and database server to prevent SQL injection attacks. Which of the following security measures should Sarah prioritize to protect the web applications and the database server against these attacks?
    a. Implement browser-based input validation.
    b. Create an input allow list on the server side.
    c. Create an input deny list for all user inputs.
    d. Deploy a web application firewall only.

16. Gary is designing a multifactor authentication system to protect a database containing highly sensitive information. The database uses a password authentication approach already. What technology could Gary add to best secure the database?
    a. PIN
    b. Security questions
    c. Fingerprint recognition
    d. Passphrase

17. Patrizia, a database developer at a midsize company, is in the planning phase of designing a new database. She wants to ensure that the database will meet the needs of its users and applications. Which stakeholders should Patrizia consult to gain insights into business trends and patterns using advanced analytics and predictive models?
    a. Executive management
    b. Data scientists
    c. System administrators
    d. Customers

18. The backup administrator configures a system to perform full backups on Sundays at 1 a.m. and incremental backups on Mondays through Saturdays at 1 a.m. The system fails on Wednesday at 4 p.m. What backups must be applied?
    a. Sunday only
    b. Sunday and Wednesday only
    c. Sunday, Monday, and Wednesday only
    d. Sunday, Monday, Tuesday, and Wednesday

19. Fayez manages a team of Java developers and wants a object-relational mapping (ORM) tool to insulate his developers from writing database-specific code. What is his best option?
    a. Hibernate
    b. ActiveRecord
    c. RedBean
    d. Django

20. Zara is a database administrator at a large corporation. She wants to automate monitoring and maintenance activities on her company's database to improve operational consistency and save time for higher-value tasks. Which type of script would be most suitable for Zara's use case?
    a. Client-side script
    b. ETL
    c. ELT
    d. Server-side script

Answers to Assessment Test

1. While all of these alerts are relevant to managing a database, this question asks specifically about storage resources, making a surge in active users, an increase in error rates, or a different read/write ratio incorrect. Since database size correlates to storage use, configuring a size-based alert is the best approach, as described in option A.
2. A. CREATE TABLE creates a new table, DROP TABLE permanently revs a table, and SELECT retrieves data from a table. Paige should use the ALTER TABLE command to modify the data type of an existing column in the address table.

3. B. Decorative server cases are irrelevant to the database system's performance, scalability, or reliability. The number of database administrators is not directly related to hardware acquisition. Open-source software is not a hardware asset. The most appropriate choice is to prioritize processing capability, memory, storage, and network bandwidth when acquiring hardware assets.

4. C. Full-interruption tests activate the alternative processing facility and take the primary site offline. Parallel tests also activate the alternative facility but keep operational

responsibility at the primary site. Structured walk-throughs and simulation tests do not activate the alternate site.

5. D. As a diagramming tool that supports UML, Lucidchart is Tanika's best option, erwin is a data modelling tool, Microsoft Word is a word processor, and Google Sheets is for maintaining spreadsheets.

6. B. The organizational data steward is responsible for developing policies and procedures for an organization's data quality, security, privacy, and regulatory compliance. While data owners work with data stewards to establish policies and procedures for an organization's quality, security, privacy, and regulatory compliance. Subject area data steward is incorrect because while they work on behalf of their data owner to handle daily tasks and are delegated governance activities, their role is specific to their subject area and not responsible for developing policies and procedures for an organization's data quality, security, privacy, and regulatory compliance. Data custodians are incorrect because their role is to implement technical controls that execute data governance policies, such as configuring applications, dashboards, and databases. While important, they are not responsible for developing policies and procedures for an organization's data quality, security, privacy, and regulatory compliance.

7. A. Deadlocks happen when two or more transactions lock a resource the other needs and block each other indefinitely. Starting and ending transactions explicitly and minimizing transaction size are two ways to prevent deadlocks from happening. Since this likely requires code changes, Hassan needs to include developers in this approach. Increasing the size of the connection pool, the time for reaping and refreshing dead connections are ways to address how clients connect to the database and don't directly impact deadlocks.

8. D. Object-oriented programming (OOP) makes software more modular, reusable, and scalable. User experience (UX) design focuses on improving a system's ease of use, efficiency, and usefulness. SQL execution plan optimization refers to understanding and optimizing query performance by examining database engine steps to execute a SQL query. Rewriting the query to use joins instead of subselects is appropriate if the execution plan shows that the query used subselects inefficiently. Changing the code to use bind variables is appropriate if the optimizer needs to parse the query for each execution. Increasing the number of cached sequence values does not impact full table scans.

9. D. Creating an index is the best option, as full table scans suggest an index does not cover the query. Rewriting the query to use joins instead of subselects is appropriate if the execution plan shows that the query used subselects inefficiently. Changing the code to use bind variables is appropriate if the optimizer needs to parse the query for each execution. Increasing the number of cached sequence values does not impact full table scans.

10. C. The default port for Oracle is 1521, the default for Microsoft SQL Server is 1433, and the default for IBM Db2 is 50000. 5432 is the default port for PostgreSQL.

11. A. You have to pay for both Oracle Enterprise Edition and IBM Db2 Standard. While Cassandra is open-source, it is not a relational database. This makes MariaDB the optimal choice.

12. B. John Paul should select a pattern recognition system because that technology can easily recognize the presence of data with regular patterns, such as credit card numbers. Watermarking technology would require that the organization mark every record that contains credit card data and would be difficult to use in this scenario.

13. C. An appropriate fire suppression system for Omar's needs is a pre-action sprinkler system. This system requires two independent events before water is released, providing an additional layer of protection against accidental water discharge. The first event is the detection of smoke or heat, which opens a valve that fills the pipes with water. The second event occurs when a fire is detected by a separate fire detection system, which opens the sprinkler heads in the affected area to release the water. This approach reduces the risk of water damage caused by accidental sprinkler discharge or leaks. A wet pipe sprinkler system has water constantly in the pipes, which means water will flow from all activated sprinkler heads, potentially causing damage to areas that are not affected by the fire. A dry pipe sprinkler system is similar to a wet pipe system, but with air in the pipes until a fire activates the system. A deluge sprinkler system is designed for high hazard areas, where a large volume of water is needed quickly to suppress fires, and is not suitable for most data center environments.

14. C. Testing the patch in a nonproduction environment is the best option. Even though the patch addresses minor defects and isn't time-sensitive, it's still essential to ensure the patch doesn't interfere with the database configuration or client applications before applying it to the production environment. Ignoring the patch could lead to support issues and is not a good choice. Applying the patch in production without testing it first is similarly unwise, as the impact of the patch is unknown.

15. B. Creating an input allow list on the server side is the most effective measure to avoid SQL injection attacks, as it specifies the exact type of input expected from users and ensures only valid and safe input is accepted. Deploying a web application firewall only, while providing an additional layer of defense, should not be the sole security measure, as input list for all user inputs, or input blacklisting, is less effective than input whitelisting, since attackers may still bypass the blacklist. Implementing browser-based input validation should not be relied upon as a security control, as attackers can easily bypass it.

16. C. The system already uses a password, which is a "something you know" factor. Therefore Gary should add a "something you have" or "something you are" factor.

Fingerprint recognition is a “something you are” (or biometric) authentication factor and would constitute multifactor authentication when combined with a password. The answers to security questions, personal identification numbers (PINs), and passphrases are all “something you know” factors and would not create multifactor authentication when combined with a password.

17. B. Executive management is an important stakeholder group that uses data from the database to track operational performance metrics and information strategic decisions. However, they typically rely on reporting tools and do not perform advanced analytics and predictive modelling directly. System administrators operate the virtual or physical servers where the database runs, handling tasks such as server sizing and disk-space allocation. While they play a crucial role in the database infrastructure, they do not directly analyze the data or work on predictive models. Customers typically interact with the database directly, using an application to submit orders or retrieve information. Their input can help shape the database structure to support their analytical work.

18. D. With incremental backups, you must first restore the most recent full backup and then apply all incremental backups that occurred since that full backup. Therefore, the administrator must restore the backups from Sunday, Monday, Tuesday, and Wednesday.

19. A. White all of these are ORM frameworks, Django is for Python, RedBean is for PHP, and ActiveRecord is for Ruby on Rails.

20. D. Client-side scripts run on a client machine and are vulnerable to connectivity issues. ETL and ELT scripts move data between databases. Server-side scripts run on the database server and are ideal for administrative tasks.

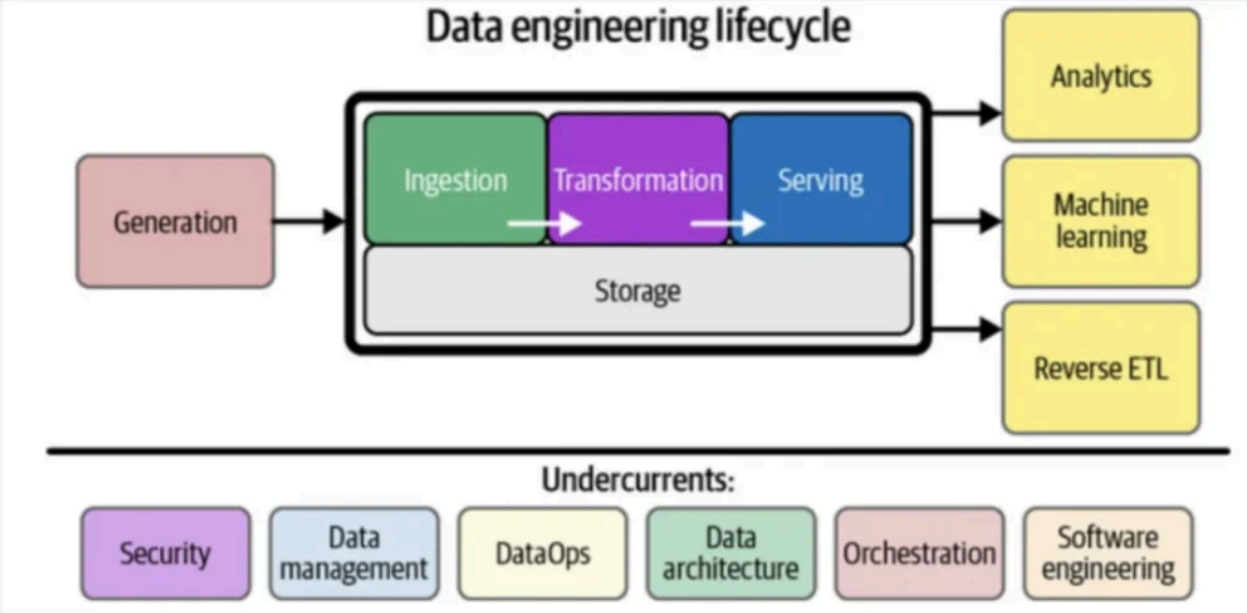
Data drives the modern business. Virtually every organization collects large quantities of data about its customers, products, employees, and service offerings. Managers naturally seek to analyze that data and harness the information it contains to improve the efficiency, effectiveness, and profitability of their work. **Data systems** *provide the ability to store, process, and analyze that data in an effective and efficient manner. Teams of dedicated data systems professionals design and manage these systems to improve their organization’s ability to compete in today’s marketplace. They also understand the importance of maintaining the security of data and databases, as well as the essential nature of ensuring that business continuity programs protect data from loss. These skills allow them to better serve their colleagues throughout the business.*

Data Drives the Modern Business

We are fortunate to live in the golden age of analytics. Businesses around the world recognize the vital nature of data to their work and are investing heavily in analytics program for years,and many of the statistical tools and techniques used in analytics work date back decades. But if

that's the case, why are we just now in the early years of this golden age? Figure 1.1 shows the three major pillars that have come together at this moment to allow analytics programs to thrive: data, storage, and computing power.

FIGURE 1.1 Analytics is made possible by modern data, storage, and computing capabilities.



Data

The amount of data the modern world generates on a daily basis is staggering. From the organized tables of spreadsheets to the storage of photos, video, and audio recordings, modern businesses create an almost overwhelming avalanche of data that is ripe for use in analytics programs.

Let's try to quantify the amount of data that exists in the world. We'll begin with an estimate made by Google's then-CEO Eric Schmidt in 2010. At a technology conference, Schmidt estimated that the sum total of all the stored knowledge created by the world at that point in time was approximately 5 exabytes. To give that a little perspective, the file containing the text of this chapter is around 100 kilobytes. So, Schmidt's estimate is that the world in 2010 had total knowledge that is about the size of 50,000,000,000 (that's 50 million!) copies of this book chapter. That's a staggering number, but that is only the beginning of our journey. Now fast forward just two years to 2012. In that year, researchers estimated that the total Schmidt's estimate of 5 exabytes was made only two years earlier. In just two years, the total number of stored data in the world grew by a factor of 200! But we're still not finished!

In the year 2022, IDC estimates that the world created 94 zettabytes (or 94,000 exabytes) of new information. Compare that to Schmidt's estimate of the world having a total of 5 exabytes of stored information in 2010. If you do the math, you'll discover that on any given day in the

modern era, the world generates an amount of brand-new data that is approximately 32 times the sum total of all information created from the dawn of civilization until 2010! Now, *that* is a staggering amount of data!

From an analytics perspective, this trove of data is a gold mine of untapped potential.

Storage

The second key trend driving the growth of analytics programs is the increased availability of storage at rapidly decreasing costs. Table 1.1 shows the cost of storing a gigabyte of data in different years using magnetic hard drives.

Figure 1: Cloud Storage List Price ($/GB Month) Reductions, 2010-Present

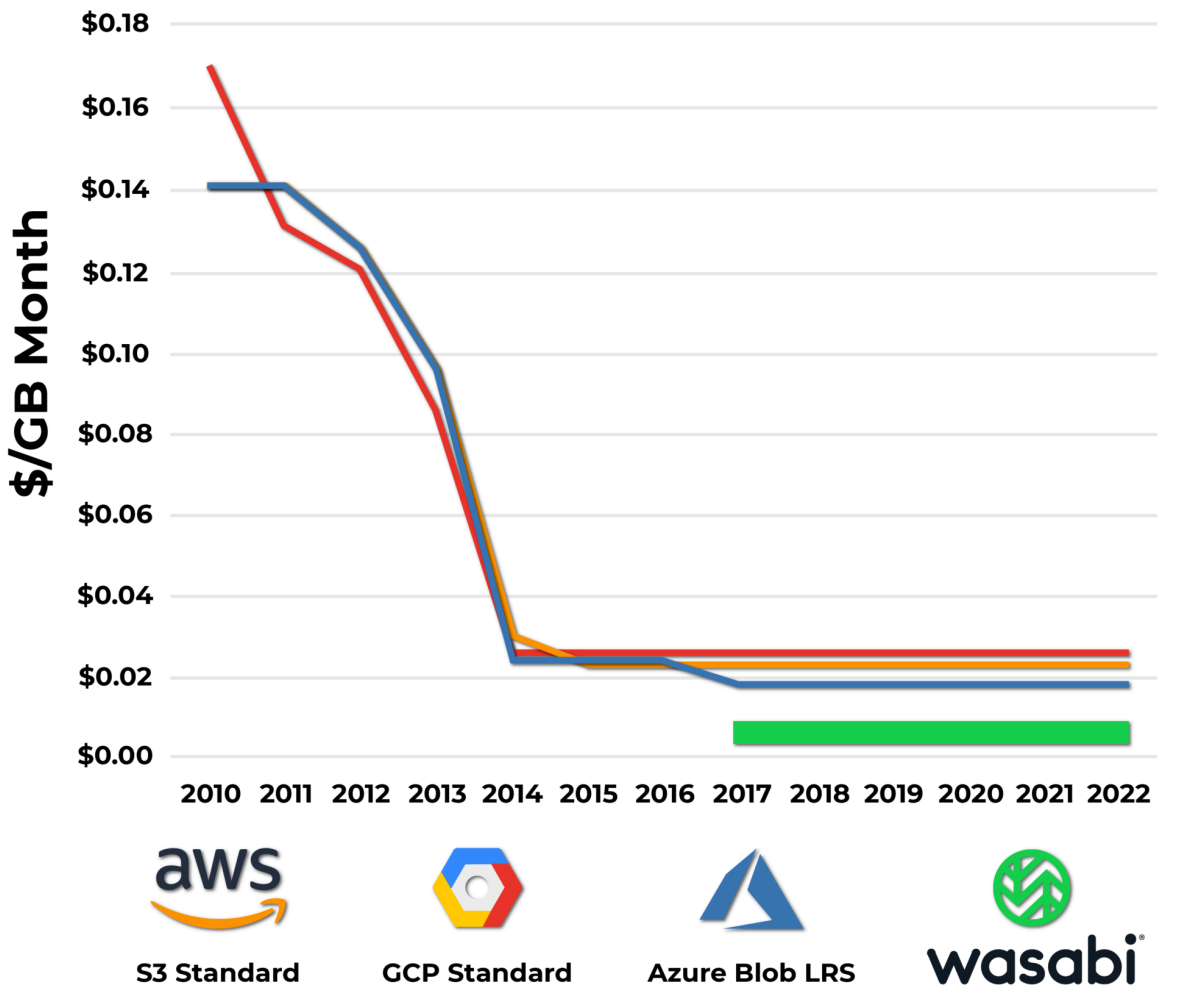


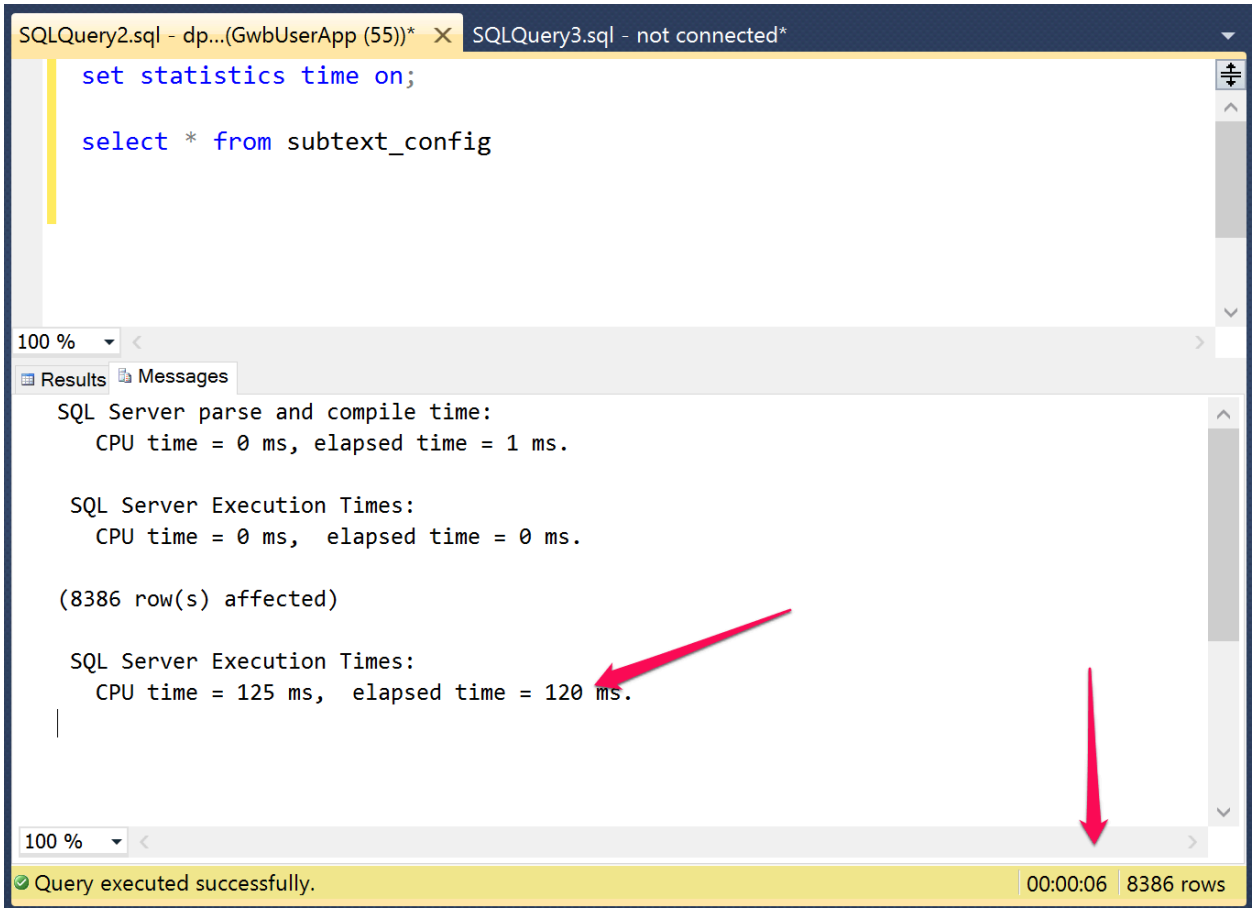
Figure 1.2 shows the same data plotted as a line graph on a logarithmic scale. This visualization clearly demonstrates that storage costs have plummeted to the point where it is almost free, and businesses can afford to retain data for analysis in ways that they have never before.

Computing Power

In 1975, Gordon Moore, one of the cofounders of Intel Corporation, made a prediction that computing technology would continue to advance so quickly that manufacturers would be able to double the number of components placed on an integrated circuit every two years.

Commonly referred to as *Moore's law,* this prediction is often loosely interpreted to mean that we will double the amount of computing power on a single device every two years. That trend has benefitted many technology-enabled fields, among them the world of analytics. In the early days of analytics, computing power was costly and difficult to come by. Organizations with advanced analytics needs purchased massive supercomputers to analyze their data, but those supercomputers were scarce. Analysts fortunate enough to work in an organization that possessed a supercomputer had to justify their requests for small slices of time when they could use the powerful machines. Today, the effects of Moore's law have democratized computing. Most employees in an organization now have enough computing power sitting on their desks to perform a wide variety of analytic tasks. If they require more powerful computing resources,cloud services allow them to rent massive banks of computers at very low cost. Even better, these resources are charged at hourly rates and analysts pay only for the computing time they actually use. These three trends – the massive volume of data generated by our businesses on a daily basis, the availability of inexpensive storage to retain that data, and the cloud's promise of virtually indefinite computing power – come together to create fertile ground for data analytics.

FIGURE 1.3 SQLQuery



Databases are the core technology for storing much of the data we use in our work and personal lives. Whether or not we are aware of it, most people interact with a database daily. At work, you use a database to record sales, manage inventory, process payroll, manage employee performance, and track customer loyalty. At home, you use a database when you check your bank balance or credit card activity, record your latest workout on your favorite fitness tracking application, or create an online photo album. As a data systems analyst, you will help choose a database platform based on an organization's needs. In the first part of this chapter, you will learn about the different types of databases. Just as organizations and individuals have unique data storage needs, there are different types of databases to choose from that best accommodate these diverse needs. You will explore use cases that lend themselves to choosing a specific kind of database. Once you are familiar with database types and how they differ, the second part of this chapter will explore some particular tools that represent the different types of databases.

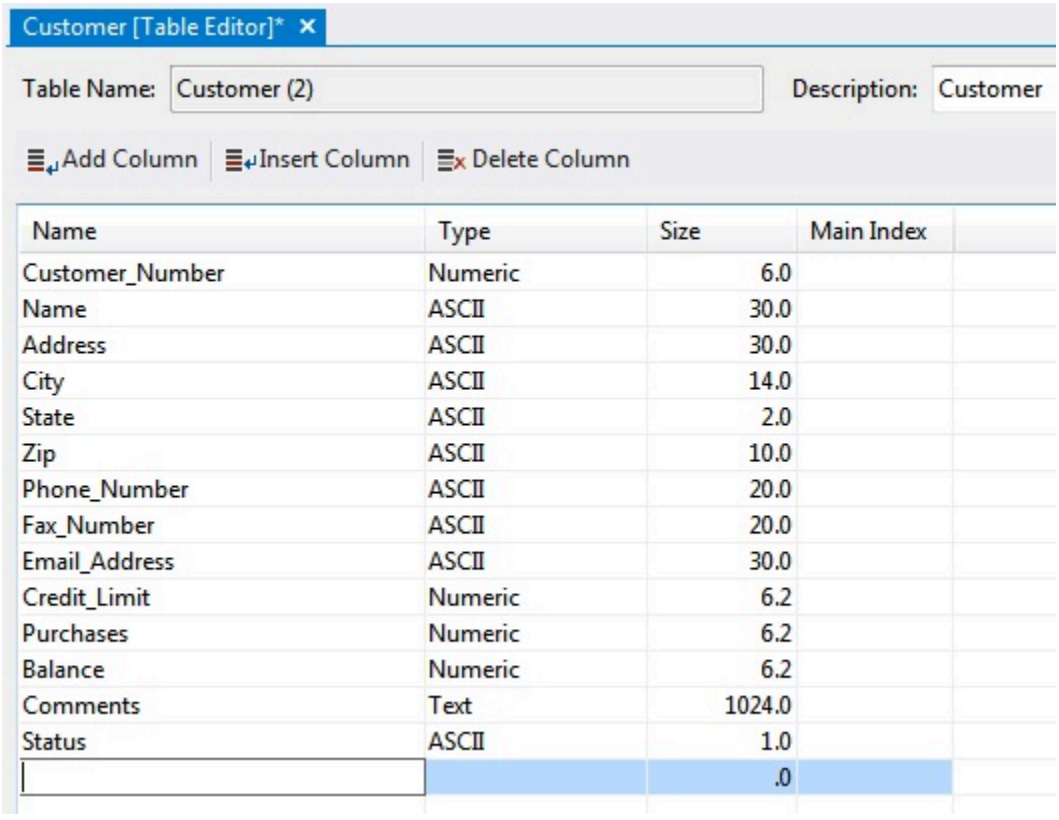
Types of Databases

Modern organizations employ various databases to store, manage, and interact with data. An organization's business requirements, availability of a knowledgeable workforce, and ability to integrate with the rest of an organization's technological ecosystem are all factors that can

influence database selection. There are two broad categories of database engines that an organization can choose from: relational and nonrelational. Let's explore these two types in greater detail.

The Relational Model

Like any piece of software, a database runs on a physical computer. People began using databases in the 1960s when computer and data storage systems were large and expensive. Partially influenced by the hardware constraints of his day, noted computer scientist Edgar F. Codd published a paper called "A Relational Model of Data for Large Shared Data Banks" in 1970. In that paper, Codd describes the term *relational model.* The relational model is an approach for structuring and organizing data. One core concept from Codd's paper is that relations provide the structure for storing data. A *relation,* also known as an *entity* or *table*, is a structure containing a collection of attributes about a data subject. Tables typically store information about people, places, or things. For example, an airline needs a customer table to maintain data.



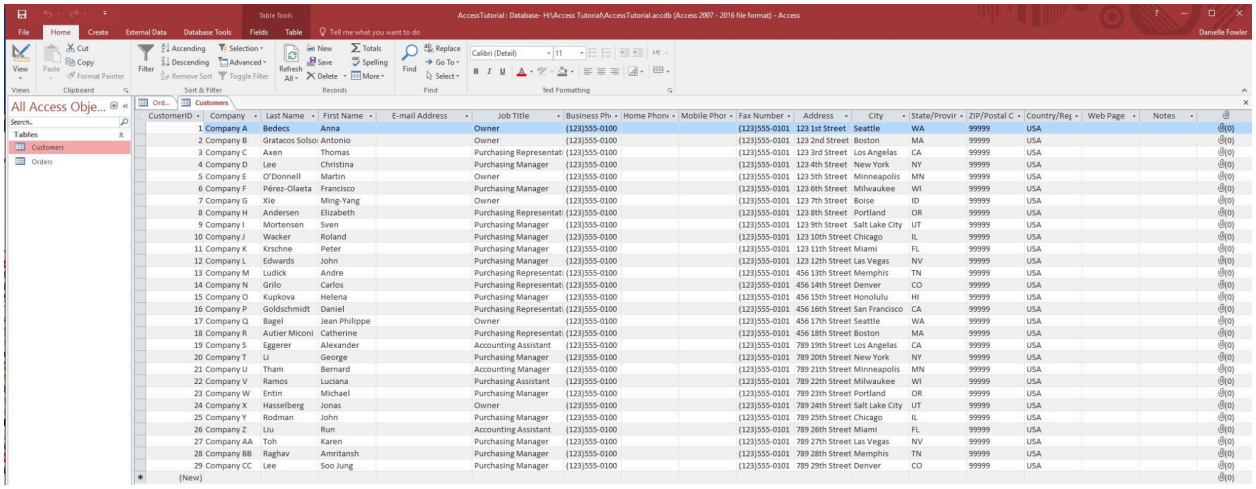
Name	Type	Size	Main Index
Customer_Number	Numeric	6.0	
Name	ASCII	30.0	
Address	ASCII	30.0	
City	ASCII	14.0	
State	ASCII	2.0	
Zip	ASCII	10.0	
Phone_Number	ASCII	20.0	
Fax_Number	ASCII	20.0	
Email_Address	ASCII	30.0	
Credit_Limit	Numeric	6.2	
Purchases	Numeric	6.2	
Balance	Numeric	6.2	
Comments	Text	1024.0	
Status	ASCII	1.0	
		.0	

An *attribute* is an individual characteristic of a table. Building on the idea of storing customer information, sample attributes include title, first name, middle name, last name, and date of birth. Consider Figure 2.1, which illustrates a Customer table with these attributes. Another core concept from Codd's paper is that relations contain tuples. A *tuple*, or *row*, contains specific

pieces of data about a single data subject. Each row needs to have an attribute that uniquely identifies the row. This unique identifier is called a *primary key.* A sequential number is frequently used for primary key values, as the purpose of a primary key is to identify individual rows. In Table 2.1, the first row of data has a primary key of 1, with the remaining columns describing a particular customer, Connor Sampson. It is vital to note that data must be consistent in each column to conform with the table's design. For example, the *First_Name* column in Table 2.1 stores first names. Trying to put a date of birth in the *First_Name* column violates this design.

	A	B	C	D	E	F	G	H	I
1			**Database clients of Jolly Day**						
2	**№**	**Customer**	**Type**	**Country**	**City**	**Contract Number**	**Date**	**Limitation years**	**Contact Manager**
3	1	Intersection	com.network	USA	New York	2314589	12.12.2012	2	Aaron
4	2	Magnet	com.network	USA	New York	2432656	27.08.2014	3	Alex
5	3	Perspective korp.	warehouse	Belarus	Minsk	2456983	31.12.2014	2	Ashley
6	4	Driveway	enterprise	USA	New York	2408570	24.04.2014	5	Aaron
7	5	near	enterprise	USA	Los Angeles	2481553	06.05.2015	2	Ashley
8	6	Nori	warehouse	Japan	Tokyo	2506369	09.09.2015	2	Blake
9	7	Nevsky comp.	com.network	Russia	Moscow	2337735	15.04.2013	1	Caroline
10	8	Perspective korp.	enterprise	Belarus	Minsk	2361112	17.08.2013	2	Daniel
11	9	in touch	warehouse	USA	San Francisco	2384723	20.12.2013	2	Alex
12	10	Nardis	com.network	Japan	Tokyo	2531433	14.01.2016	3	Blake

The concept of structuring data in this manner should feel familiar if you've ever worked with a spreadsheet where columns have names and contain consistent values. The relational model extends this spreadsheet-style organizational model by incorporating relationships between tables. For example, it is common for working professionals to have both a work address and a home address. Table 2.2 illustrates what it looks like to append these columns to the customer table.



Looking at the data within the *Work_Address* and *Home_Address* columns in Table 2.2, the value of each column entry is complex. The concept of an address contains multiple attributes,

including street name, city, state, and zip code. The structure in Table 2.2 makes data manipulation difficult. For example, suppose you want to retrieve all addresses within a specific zip code. You must search through each entry in the *Work_Address* and *Home_Address* columns and isolate the zip code. You can break the address data into a separate table to facilitate data manipulation. While Figure 2.1 illustrates attributes associated with people, Figure 2.2 shows a table design for storing address information.

Figure 2.2 Sample Address table

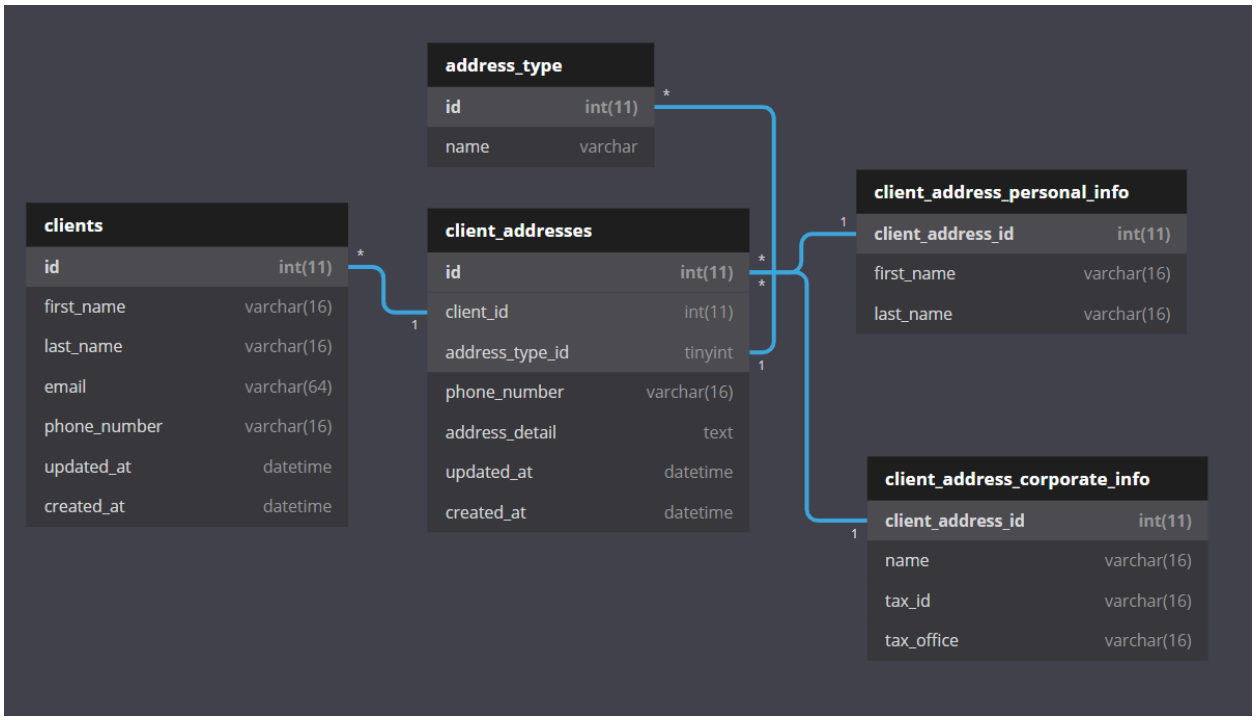


Table 2.3 illustrates what the data from Table 2.2 looks like when using the Address table pictured in Figure 2.2. Note that the values for *Address_ID,* the table's primary key, are arbitrary. The *Address_ID* column could contain any value as long as its unique per row.

Table 2.3 address Data

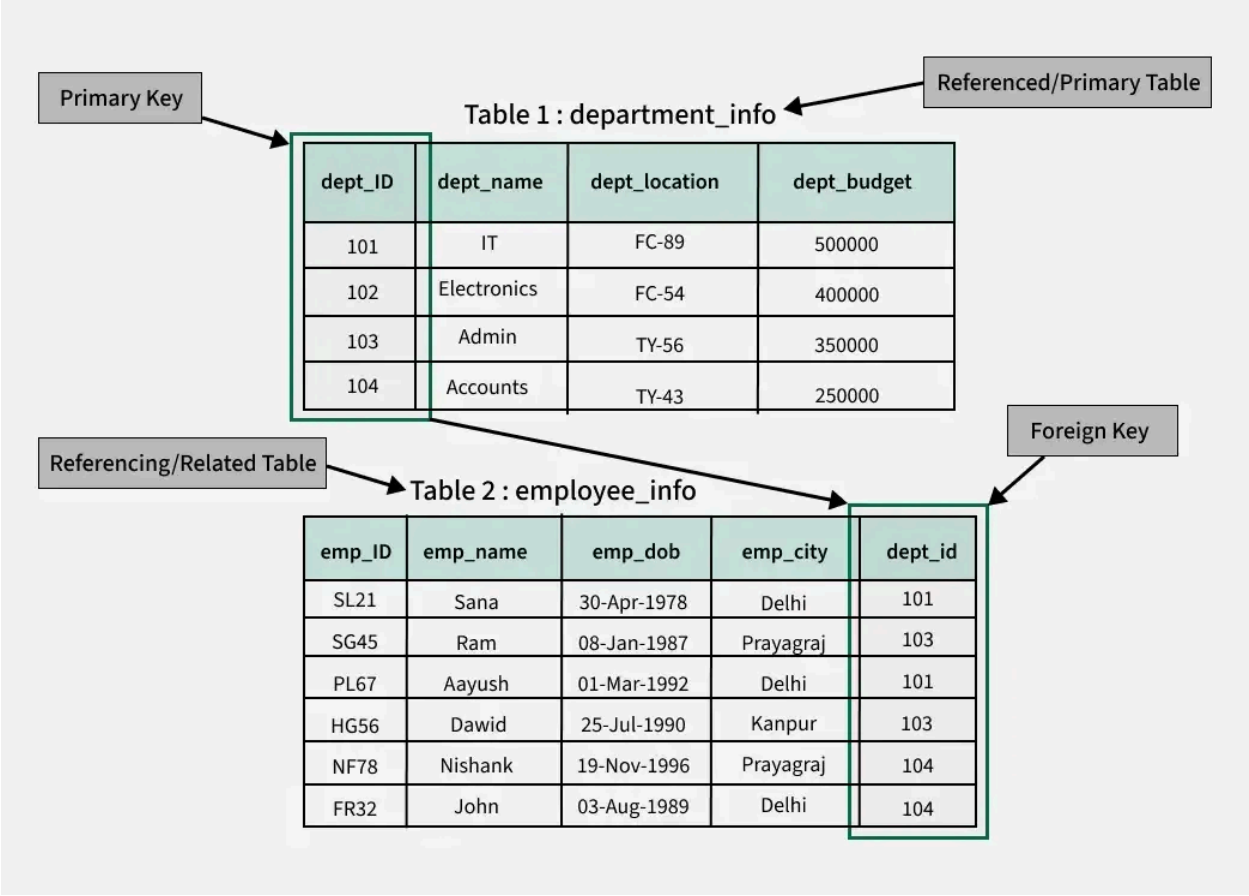
The beauty of the relational model comes from connecting tables that contain related information. Table 2.2 has customer information, including addresses. However, looking at the data in Table 2.1 and Table 2.3, there is no way to associate a specific customer and their addresses. A *foreign key* creates a link between two tables by adding an attribute to a table that references the primary key of another table. In Figure 2.3, adding the *Customer_ID* attribute to the Address table associates a specific address to a particular customer. Figure 2.3 is technically an *entity-relationship diagram*, as it illustrates the relationship between the Customer and Address tables. You will learn more about ERDs in Chapter 5, "Database Management and Maintenance."

Suppose you are working with the data from Table 2.1 and need to retrieve the address information for Laurel Neuhoff. You first need the value for Laurel's primary key from column in Table 2.4, where the *Customer_ID* column is a foreign key referring to the *Customer_ID* column in Table 2.1

Relational Databases

Based on the relational model from Codd's paper, a *relational database* organizes and stores data. A *relational database management system (RDMBS)* is a complex piece of software that lets people create, maintain, and operate relational databases. In 1979, *Oracle* became the first commercially available RDBMS. Oracle remains one of the most popular database platforms to this day. In 1983, International Business Machines (IBM) launched a relational database called DB2, rebranded as Db2 in 2017. The 1980s and 1990s saw continued growth in the number of available RDBMS platforms. Additional proprietary offerings, including *Microsoft SQL Server, Informix*, and Terradata became available. The open-source community created database platforms such as *PostgreSQL* and *MariaDB.*

TABLE 2.4 Address Data with Foreign Key



dept_ID	dept_name	dept_location	dept_budget
101	IT	FC-89	500000
102	Electronics	FC-54	400000
103	Admin	TY-56	350000
104	Accounts	TY-43	250000

emp_ID	emp_name	emp_dob	emp_city	dept_id
SL21	Sana	30-Apr-1978	Delhi	101
SG45	Ram	08-Jan-1987	Prayagraj	103
PL67	Aayush	01-Mar-1992	Delhi	101
HG56	Dawid	25-Jul-1990	Kanpur	103
NF78	Nishank	19-Nov-1996	Prayagraj	104
FR32	John	03-Aug-1989	Delhi	104

The nature of computing shifted drastically with the rise of cloud computing in the 2010s. Cloud computing providers like *Amazon Web Services(AWS), Google Cloud Platform (GCP),* and *Microsoft Azure (Azure)* have their own relational database offerings. *Aurora Azure SQL Datababase, Relational Database Service (RDS)* is another managed relational database offering from AWS that lets you run alternative relational database engines beyond *Aurora*, including *MariaDB, PostgreSQL, Oracle,* and *Microsoft SQL Server.* As cloud providers, Google and Microsoft have similar offerings that let you choose an open-source database, leaving the administration of the database platform to the cloud provider.

Today, there are many relational database management systems from which to choose. Selecting a platform depends on many considerations, including the complexity of your requirements, your technical ecosystem, and the human and capital resources you devote to maintenance and operational tasks. There are many considerations when selecting a relational database management system, including licensing model, scaling, and storage and hosting needs. Let's explore each criterion in greater detail.

Licensing Model

One of the first considerations facing an organization is whether it is worth paying a licensing fee to use a relational database. Oracle, Db2, and SQL Server are all capable RDBMS platforms supported by established companies. These companies also have dedicated organizations to support you if you need platform help. Providing these services is expensive, and you need to pay license fees if you want to use these platforms. There are open-source database platforms alternatives if an organization wants to avoid expensive licensing fees. The most popular open-source platforms include PostgreSQL and MariaDB. Anyone with an Internet connection can freely download and install either of these databases. Keep in mind if that you select an open-source database, there is no formal enhancements. If being entirely responsible for database operations is troubling, rest assured that some companies offer support and hosting options for open-source database platforms. Open-core is similar to open-source in that the core database technology remains freely available for people to use. MySQL is a popular open-core database that Oracle owns. While Oracle offers the Community Edition of MySQL for free, other versions are available for purchase, including a Standard, Enterprise, and Cluster Carrier Grade Edition. The paid versions of MySQL offer advanced scaling, backup, and monitoring capabilities.

Scale

Another consideration is your expected operational scaling capabilities. Suppose you have relatively simple needs and use the Microsoft productivity ecosystem, which includes Microsoft Windows and Microsoft 365 (Microsoft Office). Microsoft Access (Access) is an approachable starter database that integrates nicely with the rest of the Microsoft ecosystem. However, Microsoft Access has limitations that exclude it from consideration for larger organizations. For example, Access was initially designed to support one person at a time. As a legacy of that original design, database performance starts to degrade when more than one person uses an

Access database simultaneously. Another limitation is that Access stores all of its data in a single file, with a size limit of 2 gigabytes (GB). White these limits might be sufficient for individual use, they are not suitable for tasks like keeping track of each package sent by a logistics company such as FedEx, UPS, and DHL. Suppose you operate a database that supports a hotel. Hotels have a fixed number of rooms, so the number of possible reservations for any given date has a known limit. A reservation can come from the hotel's branded website or online travel shopping sites. Regardless of where a reservation comes from, the maximum number of daily reservations is known. The database would not need to scale beyond this maximum unless the hotel added rooms or opened another property in a different location. With known scaling targets, you can choose from open-source and proprietary RDBMS options. For example, Microsoft SQL server is available if you want to stay within the Microsoft technology ecosystem. As a product, SQL Server is designed to support more than 32,000 concurrent connections and can handle databases that can exceed one petabyte (PB) of data. If your technology stack uses an operating system other than Microsoft Windows, platforms including PostgreSQL, Oracle, Db2, and Terradata support thousands of concurrent connections and petabyte-scale databases.

Hosting

Another crucial operational consideration is whether you want to host an RDMBS platform yourself or rely on a platform running in the public cloud. This decision ultimately comes down to the location of other systems that interact with your database and your overall technology strategy. Suppose you have servers in a data center you own or a colocation facility where you rent space. In that case, you host the database platform yourself. When you host, you are responsible for the installation, maintenance, and operations of the database platform and any databases running on that platform. It is possible to outsource the operational aspects of an RDBMS platform to a public cloud provider. For example, AWS offers Aurora, an RDBMS that supports open-source database engines, including MySQL and PostgreSQL. You don't have to worry about installing and configuring the database software when using Aurora. With the hosted model, you also avoid concerning yourself with the underlying hardware. The trade-off is that you must use a version of the database platform supported by the cloud provider.

FIGURE 2.4 *Key-value sample data*

Phone directory

Key	Value
Paul	(091) 923467833667
Greg	(091) 947078234933
Marco	(091) 82094939023

MAC table

Key	Value
10.94.214.172	3c:22:fb:86:c1:b1
10.94.214.173	00:0a:95:9d:68:16
10.94.214.174	3c:1b:fb:45:c4:b1

One reason for choosing a key-value database is that you have a lot of data and can search by a key's value. Imagine an online music streaming service that uses a song's name as a key and the corresponding digital audio file as the value. When subscribers want to listen to music, they search for the song's name. With a known key, the application can quickly retrieve and stream the song to the user.

Document

A document database extends the key-value concept by adding restrictions on the stored values. The value in a key-value database can contain virtually anything. On the other hand, the value in a document database must conform to a specific structured format. For example, Figure 2.5 shows the data for the first two people from Table 2.2 using JSON as the document format.

FIGURE 2.5 JSON document sample data

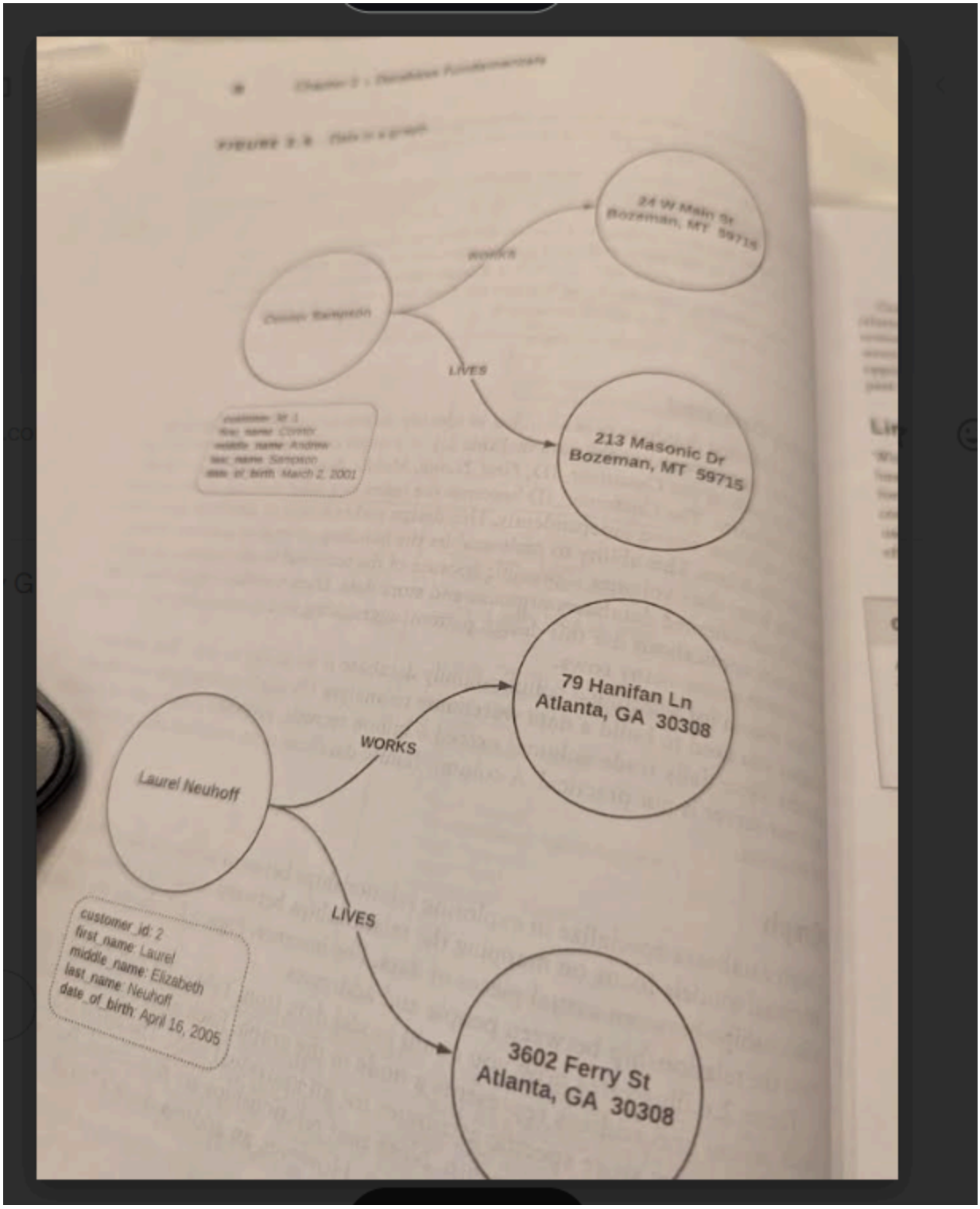
Column-Oriented

A column-oriented database uses an index to identify data in groups of related columns. A relational database stores the data in Table 2.1 in a single table, where each row contains a value for each of the *Customer_ID, First_Name, Last_Name,* and *Date_of_Birth* columns. The *Customer_ID* becomes the index in a column-family database, with the other columns stored independently. This design makes it easy to distribute data across multiple machines. This ability to scale enables the handling of massive amounts of data. Handling large data volumes is possible because of the technical implementation details of how column-oriented databases organize and store data. Data warehousing and business intelligence applications use this design pattern, aggregating and summarizing the contents of a column across many rows. One reason for selecting a column-family database is its ability to scale. For instance, imagine you need to build a data warehouse to analyze US stock transactions over three decades. Since daily trade volumes exceed 6 billion records, processing that data on a single database server is not practical. A column-family database is an excellent choice for this type of situation.

Graph

Graph databases specialize in exploring relationships between pieces of data. While relational models focus on mapping the relationships between abstract entities, graphs map relationships

between actual pieces of data. For instance, Figure 2.3 shows the tables that store the relationship between people and addresses. Figure 2.6 illustrates how you could model data from Tables 2.1 and 2.4 as a graph. Each person and address represents a *node* in the graph. Each node can have multiple *properties*. Properties store specific attributes for an individual node. The arrow that links nodes together represents a *relationship*. Note that relationships are directional. For instance, Connor works at 24 W Main St in Bozeman. However, an address doesn't "work" on a person. Relationships can also have properties.



Graphs are optimal if you need to create a recommendation engine, as graphs explore relationships between individual pieces of data. For example, consider a recommendation system where the nodes in the graph represent users and items and the relationships between them. With this design, you can see which people have purchased similar items. This approach

allows for the generation of personalized recommendations based on the user's past behavior and the behavior of other users in the graph.

A CSV file is a text file that uses a simple format for storing data in a tabular form. In a CSV file, each line of text represents a record. A comma separates the record's values for each field. Figure 2.7 illustrates the data from Table 2.1 in CSV form. This simple format makes CSV files easy to create, read, and write. As a result, people frequently use CSV files to exchange data between systems and relational databases.

Nonlinear data formats are those in which data is organized in a hierarchical or nested structure, allowing for complex relationships between data entities. JSON is an example of a nonlinear data format. Another example is extensible markup language (XML), which represents data in a tree-like structure for nested objects and elements. Nonlinear data formats are well-suited for describing complex data structures.

Exam Tip

Organizational needs change over time. Suppose you work with linear data in a relational database and have a new requirement to incorporate nonlinear data. Think carefully before trying to wedge nonlinear data into a relational database. While it is possible, remember that relational databases are optimal for linear data when selecting a database model. When you need to store nonlinear data, a NoSQL database is a better choice.

Programming and Database Operations

Object-Relational Mapping

Object-relational mapping (ORM) is a technique that presents relational data as objects in an object-oriented programming language. ORM lets developers work with data stored in relational databases using object-oriented programming languages, including Java, Python, and C++. An ORM framework provides a layer of abstraction between the programming objects and the data in the database, making it easier for developers to work with the data more intuitively and naturally. In practical terms, developers can write code that uses objects and classes to represent data rather than writing complex SQL queries that can directly manipulate the data. The ORM framework translates the operations performed on these objects into the appropriate SQL commands. This abstraction frees developers from needing to understand the organization or structure of data in the database.

One of the main benefits of using an ORM framework is that it allows developers to write more reusable and maintainable code. Since ORM frameworks abstract away the data's structural details, the generated code can work with different proprietary and open-source relational databases. The flexibility that ORM frameworks provide lets developers write portable code that can easily migrate to new database platforms. ORMs allow developers to maximize the

percentage of their time toward feature creation instead of worrying about the underlying database structures.

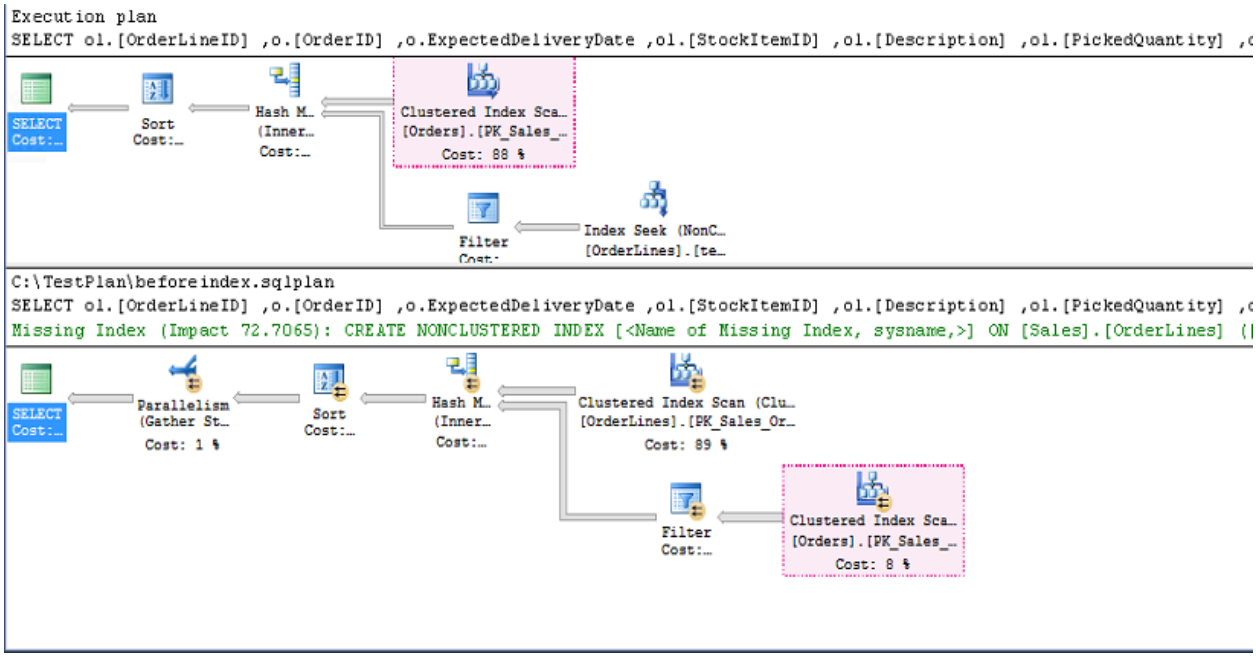
Overall, ORM frameworks are tools that improve the productivity of modern software development. ORM frameworks allow developers to focus on the business logic of their applications rather than the details of how data is stored and retrieved. By using ORM, developers can create robust, scalable, and maintainable applications to meet their users' needs better.

There are a variety of ORM frameworks from which to choose. *Hibernate* and *Ebean* are both open-source ORM frameworks for the Java programming language. As ORM frameworks, Hibernate and Ebean let developers use Java objects directly to manipulate data without writing SQL queries. Ebean is the less complex of the two frameworks, prioritizing ease of use. In addition to abstracting a database's structure, Hibernate provides additional features, including caching and transaction management. With caching, it is possible to improve the performance of an application. However, if a database gets directly modified by a SQL statement and not through the ORM framework, the Hibernate cache could contain stale data. The *Entity Framework* is an open-source ORM framework for the .NET programming language. As with all ORM frameworks, it provides a layer of abstraction over a relational database. One unique feature of the Entity Framework is that it supports Language Integrated Query (LINQ). LINQ lets .NET developers write queries using a syntax similar to the syntax used in the C# programming language. There is at least one ORM framework for every object-oriented programming language. *ActiveRecord* is an ORM framework built into *Ruby on Rails*. *Doctrine* and *RedBean* are among the ORM frameworks for the PHP programming language. *SQLAlchemy, Django,* and *Storm* are among the most broadly adopted options for Python developers.

Process to Gauge Impact

As with any tool, you need a process to gauge the impact of using an ORM framework on a relational database. ORMs generate SQL code as part of abstracting the database to the programmer. One step that can help you assess impact is to review the SQL code generated by the ORM. Specifically, you can compare the SQL code generated by the ORM with the SQL you would write manually for the same operations. Comparing the generated and manual code can help you understand how the ORM translates your code into SQL. In addition, this manual comparison lets you understand whether the ORM generates efficient and effective SQL.

To help confirm the validity of the generated SQL code, you can start by setting up a test environment with a sample database and some test data. Then you can write some code using the ORM to perform database operations consistent with those needed by your application such as querying, inserting, updating, and deleting data. Once you have the ORM-generated SQL, you can compare it to the SQL code you would write manually to perform the same operations. To determine the ORM's impact on the database server, you can use database performance tools to examine the efficiency of the ORM-generated SQL. One thing to pay close attention to is SQL's explain plan, also known as an *execution plan.*



Once you have a representative sample of how the ORM-generated SQL compares with queries you write manually, you can assess the ORM's viability for your needs. You can choose your ORM Framework if its generated queries provide solutions to your data interaction use cases. Some situations will cause you to pursue an alternate approach. For example, the complexity of your decision increases if you are developing a new application against a database that is in use by existing applications. In that case, you might get data inconsistency in the ORM's caching layer because of an update that doesn't go through the ORM. As a result, you might opt against using an ORM for this specific use case.

Summary

A database is a structured collection of data stored and accessed electronically. Databases store and manage large amounts of data and allow for easy access, organization, and manipulation of that data. Database manipulation systems are platforms where you create databases. You interact with databases when you do things online, including buying something from a retailer, managing your finances, or making a service appointment for your car. There are two main database categories. The most dominant type of database is the relational database. A relational database is rooted in Codd's 1970 paper that describes the relational model. The relational model is a way of organizing data in a database. In the relational model, you organize data into tables with well-defined relationships between them. You use primary and foreign keys to establish these relationships. A primary key is a field in a table that uniquely identifies each record in a table. A foreign key is a field in one table that refers to the primary key in another table. Relational database management systems let you create relational databases. Based on the relational model, relational databases easily manipulate and query data and are used in various applications. Relational databases are powerful and flexible ways of organizing and

storing linear data. Proprietary databases include Oracle, SQL Server, and Db2. PostgreSQL and MariaDB are among the open-source options, while MySQL is one of the most popular open-core databases. NoSQL is another database category. While not as prevalent as the traditional database, NoSQL's popularity is rising. NoSQL databases don't use a traditional relational model and handle large amounts of nonlinear data by design. Key-value, document, column-oriented, and graph are all NoSQL databases. The key-value database is the most basic of the NoSQL queries. In a key-value database, a globally unique key identifier identifies a specific value. That value can contain virtually anything, including text, an image, or a video. DynamoDB is one of the leading key-value database options.

Document databases are conceptually similar to the key-value database. A document database uses globally unique keys while enforcing some structure on the values. For example, MongoDB is a document database based on the JSON structure.

Exam Essentials

Describe the difference between relational and nonrelational databases. Relational and nonrelational, or NoSQL, are two different types of databases used to store and manage data. Relational databases use a structured data model, which organizes data into tables with rows and columns. Nonrelational databases use a variety of data models, including key-value, document, column-oriented, and graph. With relational databases, you need to define data structures before using the database. Nonrelational databases often use a more flexible data structure, allowing them to store data in various formats. From an operational standpoint, nonrelational databases are easier to scale than their relational counterparts.

Describe the difference between linear and nonlinear data. Linear and nonlinear data formats are two ways of organizing and storing data. Linear data formats are relatively straightforward to work with, given that the data is in a linear, sequential format. A CSV file is an example of a linear data format. Since relational databases are highly structured, they are well-suited to storing linear data.

Nonlinear data formats store data in a more complex, interconnected manner. Nonlinear data are more complicated to work with than linear data and allow for multiple connections and relationships between different pieces of data. Among the NoSQL databases, graph databases are particularly well-suited for storing and manipulating non-linear data.

Describe common NoSQL database categories. Key-value, document databases, column oriented, and graph are four categories of NoSQL databases that store and manage data differently. Key-value databases stored data in key-value pairs and are ideal for storing audio, image, and video data. Document databases are similar to key-value, restricting the values in a known JSON-like document format. Column-oriented databases store data in columns rather than rows, making them suitable for the aggregations associated with data warehousing and business intelligence applications. Graph databases store data in nodes and edges,

representing the entities and relationships in a dataset. Graph use cases include real-time recommendation engines, fraud detection, and identity and access management.

Describe object-relational mapping. Object-relational mapping is a technique that allows developers to work with databases using object-oriented programming languages rather than writing SQL statements. Available for many object-oriented programming languages, ORM frameworks provide a layer of abstraction between a relational database and an application, allowing developers to work with objects in their code. Meanwhile, the ORM handles the translation of these objects into SQL statements.

ORM frameworks make it easier for developers to work with databases, allowing them to use familiar object-oriented concepts and syntax rather than learning the details of SQL. It can also improve the maintainability and flexibility of an application as it decouples the application's data access logic from the underlying database structure.

Describe the impact of using an object-relational mapping framework. To gauge the impact of using an object-relational mapping framework, you first need to decide how to assess the ORM's impact. You need to select a performance measure, such as response time, throughput, or resource utilization, as the basis for comparison. You can then compare database performance between the ORM's SQL and the SQL a human writes. If you have multiple applications using the same database, be sure to include the effect of the ORM's caching layer in your evaluation, as stale data in the cache can cause inconsistent application behavior. You can also incorporate other factors, including code maintainability, in your assessment. If you determine the OM is having an outsized impact on performance, you decide whether it is viable to evaluate a different ORM framework or use human-generated SQL.

Review Questions

1. Lionel is creating a financial data warehouse. Which type of database should he consider?
    a. Key-value
    b. Document
    c. Column-oriented
    d. Graph
2. Inaya wants to use a graph database to build an identity and access management system. Which of the following is a viable option?
    a. Cassandra
    b. Neo4j
    c. MongoDB
    d. DynamoDB
3. Monique is a database administrator for a company managing customer information. She is optimizing the existing database structure to make it easier to retrieve customer addresses. Which database concept should she use to associate customer information with their addresses while maintaining data integrity?

    a. Primary Key
    b. Tuple
    c. Foreign Key
    d. Relation
4. Howard is a database designer for an e-commerce website working on creating a table to store customer information. He wants to ensure that each customer can be uniquely identified within the table. Which database concept should Jack use to accomplish this goal?
    a. Primary Key
    b. Tuple
    c. Foreign Key
    d. Relation
5. Lisa is a database administrator for a growing online retail company that requires a relational database management system to handle hundreds of concurrent connections. Lisa's company makes extensive use of Linux. Which RDBMS platform should Lisa consider for her company's needs?
    a. Access
    b. PostgreSQL
    c. SQL Server
    d. Aurora
6. Miguel is responsible for database strategy for a growing retail company that requires a relational database to handle what will eventually be a petabyte-scale database. Most of the company's infrastructure runs Linux in the cloud, and Miguel prefers a cloud-based solution that minimizes administrative overhead. Which RDBMS platform should Miguel consider for his company's needs?
    a. Access
    b. PostgreSQL
    c. SQL Server
    d. Aurora
7. Allison is working on building a database for a new online store. She is developing a feature to recommend products based on similar purchases and search history. Which type of database would be best suited for Allison's needs?
    a. Key-value
    b. Document
    c. Column-oriented
    d. Graph

8. Katie works for a social network startup and needs a database to store social network profiles. Which profiles are all JSON documents, each profile may contain different fields. Which type of NOSQL database would be Katie's best choice?
    a. Key-value
    b. Column-oriented
    c. Graph
    d. Document

9. Andy is building an environment to analyze five decades worth of financial transactions. If Andy uses commodity servers and a column-oriented design, which of the following databases best fits Andy's needs?
    a. MySQL
    b. MongoDB
    c. Cassandra
    d. SQLite
10. Sameer is a software developer working on a project that requires storing and managing JSON data. He is considering different approaches to storing the data. Which of the following describes what Sammer should choose for this project.
    a. Relational database with linear data format
    b. NoSQL database with nonlinear data format
    c. Relational database with nonlinear data format
    d. NoSQL database with linear data format

11. Luca is a software developer working on an application that requires interaction with a relational database. He is considering using an ORM framework to simplify his work. What is the primary benefit of using an ORM framework in this context?
    a. Reduces the size of the database
    b. Allows developers to work with data using object-oriented programming languages
    c. Increases the number of supported database platforms
    d. Makes the database more secure

12. Enzo is a software developer working on a project incorporating nonlinear data to support a new initiative at his company. Which type of database would be the most appropriate choice for this situation?
    a. Relational database
    b. NoSQL database
    c. Flat-file database
    d. Spreadsheet

13. Gina is a developer in a growing organization that uses a relational database for its linear data. They have recently identified the need to store and manage highly interconnected data for product recommendations. Which type of database would be the most appropriate for this situation?
    a. Relational
    b. Graph
    c. Key-value
    d. Column-oriented

14. Jane is a software developer working on a file-based interface sending employee data to the company's insurance provider. She needs a format that is easy to create, read, and write. Which file format should she choose?
    a. JSON
    b. XML
    c. CSV
    d. BSON

15. Ron is developing an application that requires storing complex data structures with hierarchical relationships. He needs a file format that can effectively represent these relationships and be easily readable. Which file format should she choose? Select the best answer.
    a. CSV
    b. DOCX
    c. JSON
    d. XML

16. Katie is designing a web application requiring an efficient approach for exchanging structured data between clients and the server. The application must easily parse and generate data in this format. Which file format should she choose? Select the best option.
    a. XML
    b. CSV
    c. JSON
    d. TSV

17. Christa is working on an application that requires storing and managing a list of customer records, each with a fixed number of fields, such as name, email address and phone number. The data will be used for simple data analysis and primarily accessed sequentially. Which type of data format should Christa choose for this task?
    a. Linear data format
    b. Nonlinear data format
    c. Graph data format
    d. Tree data format

18. Swan is a software developer working on a project to improve his company's application. He is enhancing the overall user interaction and ensuring that the application is intuitive and user-friendly. Which aspect of the application should Swan focus on?
    a. UX
    b. Database optimization
    c. APIs
    d. Load balancing

19. Sven is a development manager at a financial services firm. He wants to create a software system that is easy to maintain, scale, and modify. He's considering using a design principle focusing on object interaction and encapsulating data and behavior. What design principle should Sven use for his application?
    a. Functional programming
    b. Object-oriented programming
    c. Procedural programming
    d. Logic programming

20. Monique is a development manager for a group of Java programmers. Which of the following ORM frameworks should she evaluate to abstract away the structural details of a relational database and facilitate more reusable and maintainable code?
    a. ActiveRecord
    b. Django
    c. Hibernate
    d. LINQ

Structured Query Language is the standard programming language for managing and manipulating data stored in relational databases. It is used by organizations of all sizes and in various industries to store, retrieve, and analyze data. SQL is essential for working with relational databases because it allows developers and database administrators to create data structures and manipulate data. Scripting is the process of automating a series of tasks. As you can imagine, a data systems analyst frequently needs to automate routine relational database tasks. In the first part of this chapter, you will learn about the two main types of SQL. You will explore the theory that makes SQL so powerful and the four principles behind a transaction. You will then learn about different ways you can program with SQL. In the latter part of this chapter, you will delve into the nuances of various scripting languages, locations, and approaches.

Flavors of SQL

Structured Query Language (SQL) is the standard programming language for interacting with a relational database. Recognizing the need for standardization, the **American National Standards Institute (ANSI)** adopted SQL as a standard in 1986. The **International Organization for Standards (ISO)** followed suit in 1987, establishing SQL as a worldwide standard for interacting with relational databases. Relational databases support SQL, making it easy for developers, analysts, and administrators to use the same language to work with various relational databases, from PostgreSQL to Oracle. SQL has gone through several significant revisions through the years. The first version defined the basic syntax and data types for SQL and the structure and syntax of SQL statements. The 1992 revision introduced several new features and improvements, including support for ***views, triggers,*** and ***stored procedures***. This revision also included enhancements to SQL's data types and syntax. In 1999, ANSI updated the standard to include support for recursive queries, common table expressions, and embedding SQL in Java. Subsequent releases in 2003, 2006, 2008, 2011, and

most recently, 2016, continued to further enhance and improve the language. As it continues to evolve, SQL remains an essential tool for working with relational databases, playing a critical role in managing and analyzing data in many organizations. While the 2016 version is the most recent version of ANSI SQL as of this writing, database platforms do not uniformly support every feature of ANSI SQL.

What is ANSI SQL?

ANSI SQL (American National Standards Institute Structured Query Language) is a standardized database query language designed to ensure consistent database management and interoperability across various Database Management Systems (DBMS). First established by the American National Standards Institute (ANSI) in 1986, it has evolved through multiple versions to accommodate new features and improvements. The goal of ANSI SQL is to provide a uniform set of syntax and rules for database operations, making it easier for developers to use SQL across different platforms without having to learn proprietary extensions.

Why ANSI SQL Matters?

- **Standardization**: ANSI SQL ensures that SQL code written according to the standard can be executed on different DBMS with little or no modifications. This standardization helps avoid vendor lock-in, making it easier for businesses to switch or use multiple database systems simultaneously.
- **Cross-Platform Interoperability**: By adhering to ANSI SQL, developers can write SQL queries that are compatible with major database systems like MySQL, PostgreSQL, Oracle, and Microsoft SQL Server. This reduces the need to learn the intricacies of each system, facilitating more straightforward migration and integration efforts.

- **Foundation for SQL Learning**: ANSI SQL provides the fundamental constructs that form the basis of SQL. Once you master the standard, extending your knowledge to specific database implementations becomes more manageable.

How ANSI SQL Has Evolved Over Time

The evolution of ANSI SQL (American National Standards Institute Structured Query Language) reflects its development as the standard language for managing and querying relational databases. Here's an overview of the key milestones in the evolution of ANSI SQL:

1. SQL-86 (SQL-1)

- **Year**: 1986
- **Significance**: The first version of SQL standardized by ANSI and ISO.
- **Features**:
    - Basic data definition (CREATE, DROP)
    - Data manipulation (SELECT, INSERT, UPDATE, DELETE)
    - Simple predicates (WHERE clause)
    - Basic set operations (JOINs, UNION)
    - No advanced data types or constraints.

Impact: Standardized SQL across multiple database vendors, making SQL a widely adopted language for relational databases.

2. SQL-89

- **Year**: 1989
- **Significance**: A minor revision to the SQL-86 standard, improving consistency and fixing ambiguities.

- **Features**: Mostly clarifications and minor corrections, no major new features were added.

Impact: It helped address some implementation differences between vendors.

3. SQL-92 (SQL-2)

- **Year**: 1992
- **Significance**: A major update to the SQL standard, expanding it significantly.
- **Features**:
    - Advanced JOIN types (LEFT, RIGHT, FULL OUTER JOIN)
    - Subqueries
    - String and date manipulation functions
    - Transaction control (START TRANSACTION, COMMIT, ROLLBACK)
    - Support for data integrity through constraints (PRIMARY KEY, FOREIGN KEY, UNIQUE)
    - Views (virtual tables)
    - Set operations (INTERSECT, EXCEPT)
    - Support for NULLs.

Impact: SQL-92 became a robust standard adopted by many major database systems, including Oracle, DB2, and SQL Server, making interoperability between systems easier.

4. SQL:1999 (SQL-3)

- **Year**: 1999
- **Significance**: Introduced object-oriented programming features and procedural extensions.
- **Features**:
    - User-defined types (UDTs)
    - Recursive queries (WITH RECURSIVE)
    - Triggers
    - Procedural elements (control-flow statements such as IF, CASE, LOOP, etc.)
    - SQL-based functions and stored procedures
    - Temporary tables.

Impact: The object-oriented capabilities made it possible to model complex data types, and recursive queries enabled handling hierarchical and graph-like data structures.

5. SQL:2003

- **Year**: 2003
- **Significance**: Introduced XML data support and window functions.
- **Features**:
    - XML data type and operations (XML support in SQL queries)
    - Window functions (e.g., ROW_NUMBER(), RANK(), PARTITION BY, OVER)
    - Sequence generators (auto-incrementing values).

Impact: Window functions significantly improved SQL's ability to handle analytical queries, making it much more powerful for reporting and analytics.

6. SQL:2006

- **Year**: 2006
- **Significance**: Focused mainly on XML-related extensions.
- **Features**:
    - More extensive support for working with XML data (XPath, XQuery support).

Impact: XML was still growing as a data exchange format, so this revision enhanced SQL's ability to handle XML documents within databases.

7. SQL:2008

- **Year**: 2008
- **Significance**: Refined many features and added minor enhancements.
- **Features**:
    - New data types (e.g., `BIGINT`)
    - Enhancements to OLAP (online analytical processing) functions.

Impact: Improved SQL for modern business intelligence and analytics use cases.

8. SQL:2011

- **Year**: 2011
- **Significance**: Added time-based data management.
- **Features**:
    - Temporal tables (for time-based data management and history tracking)
    - Period data types.

Impact: Temporal tables allowed databases to maintain historical data, making SQL more useful for tracking changes over time (e.g., in financial or auditing systems).

9. SQL:2016

- **Year**: 2016
- **Significance**: **Added JSON data support and improvements to security features.**
- **Features**:
    - JSON data type and functions (e.g., JSON_TABLE)
    - Row pattern recognition (MATCH_RECOGNIZE for complex event processing).

Impact: Addressed the growing use of semi-structured data by integrating JSON into relational databases, further expanding SQL's reach into NoSQL-like data handling.

10. SQL:2019

- **Year**: 2019
- **Significance**: Introduced extensions for working with multidimensional arrays and enhanced data processing.
- **Features**:
    - Improvements in window functions
    - Support for polymorphic table functions
    - Enhancements to multidimensional arrays.

Impact: Extended SQL's capabilities for complex analytics, enhancing its usage for data science applications and real-time analytics in large-scale systems.

Current Trends

SQL has continued to evolve in response to the growing demands of data management and analytics, integrating with modern data formats (JSON, XML), supporting complex analytical functions (windowing, recursive queries), and expanding to accommodate new paradigms like big data, distributed systems, and real-time processing.

Future developments are likely to focus on:

- Better integration with cloud environments.
- Enhanced machine learning support.
- More seamless interoperability between SQL and non-SQL data systems (e.g., integration with graph databases and NoSQL).
- Further enhancements to support streaming data and event-driven architectures.

Key Components of ANSI SQL

1. Data Definition Language (DDL)

DDL commands are used to define, modify, and remove database objects like tables, indexes, and schemas. The most common DDL statements are `CREATE`, `ALTER`, and `DROP`.

Example: Creating a Table

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    hire_date DATE,
    salary DECIMAL(10, 2),
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES
departments(department_id)
);
```

- **Explanation**:
  - `CREATE TABLE`: Creates a new table called `employees`.
  - `employee_id INT PRIMARY KEY`: Defines a column for the employee's unique identifier, marked as the primary key.
  - `first_name VARCHAR(50) NOT NULL`: Defines a column for the employee's first name that must be filled (`NOT NULL` constraint).
  - `FOREIGN KEY (department_id) REFERENCES departments(department_id)`: Specifies a foreign key relationship with another table, `departments`.

Example: Modifying a Table

```
ALTER TABLE employees ADD email VARCHAR(100);
```

- **Explanation**: Adds a new column `email` of type `VARCHAR(100)` to the existing `employees` table.

Example: Dropping a Table

```
DROP TABLE employees;
```

- **Explanation**: Removes the `employees` table from the database completely.

2. Data Manipulation Language (DML)

DML commands are used to retrieve and manipulate the data within the database. The most common DML commands are `SELECT`, `INSERT`, `UPDATE`, and `DELETE`.

Example: Inserting Data into a Table

```
 INSERT INTO employees (employee_id, first_name, last_name,
hire_date, salary, department_id)

VALUES (1, 'John', 'Doe', '2022-01-01', 50000.00, 101);
```

- **Explanation**:
  - Inserts a new row into the `employees` table with the specified values.
  - The values provided correspond to the `employee_id`, `first_name`, `last_name`, `hire_date`, `salary`, and `department_id` columns.

Example: Selecting Data from a Table

```
SELECT first_name, last_name, salary

FROM employees

WHERE department_id = 101
```

```
ORDER BY last_name;
```

- **Explanation**:
  - Retrieves `first_name`, `last_name`, and `salary` from all employees in department `101`.
  - The result set is ordered alphabetically by `last_name`.

Example: Updating Data in a Table

```
UPDATE employees

SET salary = salary * 1.10

WHERE department_id = 101;
```

- **Explanation**:
  - Increases the salary of all employees in department `101` by 10%.

Example: Deleting Data from a Table

```
DELETE FROM employees

WHERE employee_id = 1;
```

- **Explanation**:
  - Deletes the row where the `employee_id` is `1` from the `employees` table.

3. Data Control Language (DCL)

DCL commands manage permissions and access control for database objects. The most common commands are `GRANT` and `REVOKE`.

Example: Granting Permissions

```
GRANT SELECT, INSERT ON employees TO user1;
```

- **Explanation**:
  - Grants `SELECT` and `INSERT` permissions on the `employees` table to `user1`.

Example: Revoking Permissions

```
REVOKE INSERT ON employees FROM user1;
```

- **Explanation**:
  - Removes the `INSERT` permission on the `employees` table from `user1`, but the user still retains `SELECT` access.

4. Transaction Control Language (TCL)

TCL commands manage transactions to ensure data integrity. Common TCL commands are `BEGIN`, `COMMIT`, and `ROLLBACK`.

Example: Transaction Management

```
BEGIN TRANSACTION;
```

```
UPDATE employees

SET salary = salary * 1.10

WHERE department_id = 101;




COMMIT;
```

- **Explanation**:
  - The `BEGIN TRANSACTION` starts a transaction.
  - The `UPDATE` command increases the salary of employees in department `101` by 10%.
  - The `COMMIT` command finalizes the transaction, ensuring the changes are permanently applied to the database.

Example: Rolling Back a Transaction

```
BEGIN TRANSACTION;




UPDATE employees

SET salary = salary * 1.10

WHERE department_id = 101;
```

```
ROLLBACK; -- Cancel the transaction
```

- **Explanation**:
    - This example begins a transaction, but the `ROLLBACK` command cancels all changes made within the transaction, restoring the data to its previous state.

ANSI vs. Non-ANSI Joins in SQL: Understanding the Difference

When it comes to writing SQL queries that join two or more tables, there are two distinct approaches: the **ANSI standard** and the **Non-ANSI standard**. We'll break down both approaches, explain how they work, and highlight which method is generally considered better for modern SQL development.

ANSI Joins: The Standard Method

ANSI joins are the modern, widely accepted way of writing SQL joins. These joins explicitly use the `JOIN` keyword along with the `ON` clause to define the join condition between tables. This approach allows for clearer, more structured SQL queries, making it easier to distinguish between join conditions and filtering conditions.

Example: ANSI SQL Inner Join

```
SELECT e.employee_name, d.department_name

FROM employees e

INNER JOIN departments d

ON e.department_id = d.department_id;
```

- **Explanation**:
  - `INNER JOIN`: Specifies that only rows with matching `department_id` values from both `employees` and `departments` will be returned.
  - `ON e.department_id = d.department_id`: Defines the condition on which the two tables are joined.

In this example, we are fetching the employee names along with the names of the departments they belong to. The use of the `INNER JOIN` keyword and the `ON` clause makes the query easy to read and understand.

Example: ANSI SQL Left Join

```
SELECT e.employee_name, d.department_name

FROM employees e

LEFT JOIN departments d

ON e.department_id = d.department_id;
```

- **Explanation**: The `LEFT JOIN` returns all employees, even if they do not belong to a department. If no match is found in the `departments` table, the `department_name` will be `NULL`.

Non-ANSI Joins: The Legacy Method

Non-ANSI joins are an older way of writing SQL joins, often referred to as "implicit joins." Before the `JOIN` keyword was introduced, SQL developers would write joins by simply listing the tables in the `FROM` clause, separated by commas, and then specifying the join condition in the `WHERE` clause. This method can still be found in legacy systems or older SQL scripts, but it is generally considered outdated and harder to maintain.

Example: Non-ANSI Inner Join

```
SELECT e.employee_name, d.department_name

FROM employees e, departments d

WHERE e.department_id = d.department_id;
```

- **Explanation**: Instead of using the `JOIN` keyword, the tables are separated by a comma, and the join condition (`e.department_id = d.department_id`) is placed in the `WHERE` clause.

While this query will return the same result as the ANSI inner join, the non-ANSI format is harder to read, especially in more complex queries involving multiple joins.

Example: Non-ANSI Left Join (Oracle Syntax)

Non-ANSI joins are particularly tricky when dealing with outer joins. In systems like Oracle, a special syntax using the `(+)` symbol is required.

```
SELECT e.employee_name, d.department_name

FROM employees e, departments d

WHERE e.department_id = d.department_id(+);
```

- **Explanation**: The `(+)` symbol indicates that a left outer join should be performed. This query will return all employees, even if they don't have a matching department.

However, this syntax is not supported in many other databases like PostgreSQL or MySQL, making it less portable.

ANSI vs. Non-ANSI Joins: Key Differences

1. Syntax and Readability

- **ANSI Joins**: The join condition is explicitly defined using the `JOIN` keyword and the `ON` clause. This separates the join logic from filtering conditions and makes the query more readable.
  Example:
  ```
  SELECT e.employee_name, d.department_name
  FROM employees e
  INNER JOIN departments d
  ON e.department_id = d.department_id
  WHERE d.department_name = 'HR';
  ```
- **Non-ANSI Joins**: The join condition is mixed with filtering conditions in the `WHERE` clause. This can make the query harder to read and understand,

especially as the complexity increases.
Example:

```
SELECT e.employee_name, d.department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id AND
d.department_name = 'HR';
```

2. Portability Across RDBMS

- **ANSI Joins**: ANSI SQL joins are supported by all major RDBMS, making them highly portable. Whether you're working in MySQL, SQL Server, Oracle, or PostgreSQL, ANSI SQL queries will run consistently.
- **Non-ANSI Joins**: Non-ANSI syntax, especially for outer joins (like the Oracle `(+)` symbol), is not supported across all databases. For example, PostgreSQL and MySQL do not support this method, which limits the portability of your SQL code.

3. Error Detection and Prevention

- **ANSI Joins**: The explicit use of the `JOIN` keyword helps prevent accidental cross joins. If you forget to specify a join condition, SQL will throw an error.
  Example:

  ```
  SELECT e.employee_name, d.department_name
  FROM employees e
  INNER JOIN departments d;
  ```

  - This query will throw an error because the `ON` clause is missing.
- **Non-ANSI Joins**: If you forget to specify a join condition in a non-ANSI join, SQL will perform a cross join, which can result in an enormous and unintended dataset. This can be a significant issue in larger databases.
  Example:

  ```
  SELECT e.employee_name, d.department_name
  FROM employees e, departments d;
  ```

  - This query will return the Cartesian product of both tables, combining every row in `employees` with every row in `departments`.

4. Outer Joins Complexity

- **ANSI Joins**: Handling outer joins is simple and consistent in ANSI SQL. You can easily perform `LEFT JOIN`, `RIGHT JOIN`, or `FULL JOIN` with clear syntax.
- **Non-ANSI Joins**: Non-ANSI joins require database-specific syntax (e.g., the Oracle `(+)` symbol for outer joins), making the code less portable and harder to understand.

Advantages of ANSI Joins

- **Readability**: The clear separation of the join condition (`ON` clause) and filtering logic (`WHERE` clause) makes ANSI SQL easier to read, especially for complex queries involving multiple tables and joins.
- **Error Prevention**: ANSI SQL helps prevent cross joins by requiring an explicit join condition. If the join condition is missing, the query will throw an error instead of running incorrectly.
- **Portability**: ANSI SQL is universally supported across all major RDBMS, making your SQL queries more portable and adaptable.
- **Maintainability**: As your queries grow in complexity, ANSI joins provide better structure and are easier to maintain and debug.

While both ANSI and Non-ANSI join syntax will return the same results for basic queries, **ANSI joins** are considered the best practice in modern SQL development. They provide better readability, error prevention, and portability across database systems, making them more suitable for complex queries and long-term maintenance.

Therefore, if you're writing SQL today or maintaining an existing codebase, it's highly recommended to use ANSI SQL for all join operations.

ANSI SQL FAQ

What is the relationship between ANSI SQL and MySQL?

MySQL is a relational database management system (RDBMS) that implements SQL, following the ANSI SQL standard. However, MySQL also includes several proprietary extensions and features that go beyond ANSI SQL, making it a specific implementation of SQL with additional functionalities.

Does MySQL fully comply with ANSI SQL?

While MySQL adheres to the core principles of ANSI SQL, it does not fully comply with the standard. MySQL implements most of the SQL-92 standard and parts of SQL:1999, SQL:2003, and later versions, but it also has unique features and extensions not found in ANSI SQL, such as additional functions and data types.

How does MySQL differ from ANSI SQL?

MySQL differs from ANSI SQL in several ways:

- **Proprietary features**: MySQL introduces proprietary extensions, such as specific functions (e.g., `INET_ATON()`, `FIND_IN_SET()`) and storage engines like InnoDB and MyISAM.
- **Data types**: MySQL supports some data types that aren't part of the ANSI SQL standard (e.g., `TINYINT`, `ENUM`).
- **Handling of NULLs**: MySQL may treat NULL values differently than ANSI SQL in certain contexts, such as indexing or aggregation.
- **Limit and pagination**: MySQL uses the `LIMIT` clause for pagination, while ANSI SQL uses more standardized methods like `FETCH FIRST` or `OFFSET`.

Which databases are ANSI SQL-compliant?

Several popular databases are ANSI SQL-compliant, meaning they implement most of the core SQL functionality based on the ANSI standard. These include:

- **MySQL**
- **PostgreSQL**
- **Oracle Database**
- **Microsoft SQL Server**
- **StarRocks**
- **SQLite**

These databases implement the core SQL functionality defined by the ANSI standard while also offering proprietary extensions.

Is it possible to migrate SQL queries between different ANSI SQL-compliant databases?

Yes, one of the main advantages of ANSI SQL compliance is query portability. Basic SQL queries should work across compliant databases with minimal modification. However, if a query uses database-specific extensions or optimizations, some adjustments might be necessary during migration.

Are there differences in performance between ANSI SQL-compliant databases?

While the SQL syntax might be standardized, performance can vary between ANSI SQL-compliant databases due to differences in query optimization, indexing, storage engines, and hardware architectures. For example, while adhering to the ANSI SQL standard, StarRocks offers significant performance optimizations tailored for complex analytical queries. By combining ANSI SQL compliance with enhanced query execution speed, StarRocks enables businesses to use standardized SQL while benefiting from faster query performance in large-scale data environments. This gives StarRocks an advantage for data-intensive workloads without sacrificing SQL portability.

What happens if a database is not ANSI SQL-compliant?

Non-ANSI SQL-compliant databases often introduce their own query languages or syntax extensions, which can limit the portability of SQL queries. These databases may be optimized for specific use cases but may require additional learning or code adjustments when switching between systems.

How does ANSI SQL differ from NoSQL databases?

ANSI SQL is used with relational databases that follow a structured, schema-based approach. In contrast, NoSQL databases handle unstructured or semi-structured data

without requiring a predefined schema. However, ANSI SQL has evolved to handle semi-structured data types like JSON, narrowing the gap between the two.

Can you use ANSI SQL in non-relational databases?

No, ANSI SQL is designed specifically for relational databases. However, many modern database systems, including some NoSQL databases, provide SQL-like querying capabilities to offer similar functionality.

How does ANSI SQL handle semi-structured data?

Starting with SQL:2016, ANSI SQL provides support for semi-structured data like JSON. This allows for the storage and querying of data that doesn't fit neatly into relational rows and columns, bridging the gap between traditional SQL and NoSQL databases.

Is ANSI SQL still relevant today?

Yes, ANSI SQL remains highly relevant as it is the foundational query language for relational databases. Over time, it has evolved to support modern data types, analytics functions, and new data formats like JSON and XML, ensuring it remains a crucial tool for data management.

Conclusion

Understanding the key components of ANSI SQL (DDL, DML, DCL, TCL) and the differences between ANSI SQL and proprietary joins is essential for database professionals. ANSI SQL provides a standardized, portable foundation for managing relational databases, while proprietary extensions offer additional functionality and performance optimizations tailored to specific systems. By mastering both, developers can write highly efficient, maintainable, and portable SQL code.