對本書的讚譽

資料世界發展到現在已經有一段時間。首先是設計師，然後是資料庫管理員，接著是首席資訊官，然後是資料架構師。這本書將促使該行業朝著更加成熟和先進的方向發展。對於任何認真對待自己的專業和職業的人來說，這本書是一本必讀之作。

—比爾．英蒙（Bill Inmon），資料倉儲的創建者

本書是一本絕佳的入門書籍，它涵蓋了資料遷移、處理和管理的相關業務。它解釋了資料概念的分類，而不過於關注個別工具或供應商，因此這些技術和想法應該能夠比任何個別的趨勢或產品更持久。這本書非常適合任何想要快速瞭解資料工程或分析的人，或者想要填補自己知識漏洞的現有從業人員。

—喬丹．蒂加尼（Jordan Tigani），MotherDuck 的創始人和首席執行官，以及 BigQuery 的創始工程師和共同創始人

如果想在行業中取得領先地位，你必須有能力提供優質的客戶和員工體驗。這不僅僅是技術問題，更是一個發掘和培養人才的機遇。這種轉變將影響你的業務。而在這個轉變的過程中，資料工程師處於核心地位，扮演著關鍵角色。但如今這個領域往往被誤解。這本書將揭開資料工程的神祕面紗，並成為你成功利用資料的終極指南。

—布魯諾．阿齊扎（Bruno Aziza），Google Cloud 資料分析主管

《資料工程基礎》是第一本深入而全面探討當今資料工程師需求的書籍。正如你所看到的，這本書深入探討了資料工程的關鍵領域，包括在當今複雜的技術環境中管理、移動、整理資料所使用的技能集、工具和架構。

更重要的是，Joe 和 Matt 展現了他們對資料工程的深入瞭解，並花時間深入探討資料工程更細微的領域，使其與讀者產生共鳴。無論你是經理、經驗豐富的資料工程師，還是想要進入這個領域的人，這本書都提供了對當今資料工程領域的實用見解。

—金喬恩（Jon King），首席資料架構師

有兩件事到了 2042 年仍將與資料工程師息息相關：SQL 和這本書。Joe 和 Matt 避免了被工具的誇大宣傳所影響，專注於揭示該領域中重要且持續演變的要素。無論你是剛踏上資料工程之旅，還是想進一步提升自己的專業水準，本書都將為你奠定精通資料工程的基礎。

—胡凱文（Kevin Hu），Metaplane 首席執行官

在這個瞬息萬變的領域中，不斷湧現新的技術解決方案，Joe 和 Matt 提供了清晰而永恆的指導，專注於作為一名資料工程師所需具備的核心概念和基礎知識。這本書包含了大量資訊，使你能夠在設計資料架構和在整個資料工程生命週期中實施解決方案時，提出正確的問題、瞭解權衡並做出最佳決策。無論你是正在考慮成為一名資料工程師，還是已經在這個領域工作許多年，我保證你會從這本書中學到一些東西！

—朱莉．普萊斯（Julie Price），SingleStore 高階產品經理

《資料工程基礎》不僅僅是一本指導手冊，它還教你如何像資料工程師一樣思考。這本書一部分是歷史課程，一部分是理論，一部分是從 Joe 和 Matt 數十年的經驗中所獲得的知識，絕對應該在每位資料專業人士的書架上佔有一席之地。

—斯科特．布萊特諾爾（Scott Breitenother），Brooklyn Data Co. 創始人兼首席執行官

前言

這本書是怎麼來的？它的源頭深植於我們從資料科學轉向資料工程的旅程中。我們常開玩笑地稱自己為「康復中的資料科學家」。我們兩人都有過被指派參與資料科學專案的經驗，但由於缺乏適當的基礎而無法順利進行這些專案。我們的資料工程之旅始於我們承擔資料工程任務以建構基礎架構和基礎設施。

隨著資料科學的興起，企業大肆投資於資料科學人才，希望獲得豐厚的回報。很多時候，資料科學家都在努力解決他們的背景和培訓無法解決的基本問題——資料蒐集、資料清理、資料存取、資料轉換和資料基礎架構。而這些都是資料工程所要解決的問題。

這本書不是什麼

在我們介紹這本書的內容以及你會從中得到什麼之前，讓我們來快速瞭解一下這本書不是什麼。關於資料工程，本書不會探討特定的工具、技術或平台。儘管有許多優秀的書籍會從這個角度來介紹資料工程技術，但這些書籍的保存期很短。相反地，我們只會專注在資料工程背後的基本概念。

這本書是什麼

本書的目標是填補當前資料工程相關內容和材料中的空白。雖然有許多技術資源涉及特定的資料工程工具和技術，但人們很難理解如何將這些元件組合成一個適用於現實世界的連貫整體。這本書將資料生命週期的各個環節聯繫了起來。向你展示如何將各種技術組合起來，以滿足分析師、資料科學家和機器學習工程師等下游資料消費者的需求。這本書是 O'Reilly「專門探討特定技術、平台和程式設計語言細節」之其他書籍的補充。

本書的重要概念是**資料工程生命週期**（*data engineering lifecycle*）：資料的產生、儲存、攝取、轉換和提供。自從資料開始被廣泛應用以來，我們見證了無數特定技術和供應商產品的興衰，但資料工程生命週期的各個階段基本保持不變。透過這個框架，讀者將會對如何把技術應用於現實世界中的商業問題有一個很好的瞭解。

我們的目標是制定一套跨越兩個維度的原則。首先，我們希望將資料工程提煉成可以包含**任何相關技術**（*any relevant technology*）的原則。其次，我們希望提出經得起**時間**考驗的原則。我們希望這些想法能夠反映過去二十年來從資料技術變革中所學到的教訓，並且我們的思維框架在未來十年或更長時間內仍然有用。

需要注意的一點是：我們毫不後悔地採用了雲端優先的作法。我們將雲端視為一個根本性的變革，而且將會持續數十年；大多數的本地資料系統和工作負載，最終將遷移到雲端託管。我們假設基礎架構和系統是**短暫的**（*ephemeral*）且**可擴展的**（*scalable*），並且資料工程師將傾向於在雲端部署託管服務。儘管如此，本書中的大多數概念也適用於非雲端環境。

誰應該閱讀這本書

本書的主要目標讀者包括技術從業者、中高級軟體工程師、資料科學家或有興趣進入資料工程領域的分析師；或者在特定技術領域內從事工作的資料工程師，但希望發展更全面的視角。我們的次要目標讀者是與技術從業人員相關的資料利益相關者，例如，具有技術背景並負責監督資料工程師團隊的資料團隊負責人，或者希望從本地技術（on-premises technology）遷移到基於雲端之解決方案（cloud-based solution）的資料倉儲主管。

理想情況下，你是一個好奇且渴望學習的人——否則你為什麼會閱讀這本書呢？你會透過閱讀有關資料倉儲 / 資料湖泊、批次和串流系統、編排、建模、管理、分析、雲端技術開發等方面的書籍和文章，維持對資料技術和趨勢的最新瞭解。這本書將幫助你把所閱讀到的內容，編織成一幅跨越技術和範式的資料工程完整圖像。

先備知識和技能

我們假設讀者對於企業環境（corporate setting）中的各種資料系統有相當的熟悉度。此外，我們假設讀者對 SQL 和 Python（或其他程式設計語言）有一定的熟悉度，並且有使用雲端服務的經驗。

對於有志於成為資料工程師的人來說，有大量的資源可供練習 Python 和 SQL。免費的線上資源（部落格文章、教學網站、YouTube 影片）比比皆是，並且每年還會有許多新的 Python 書籍出版。

雲端提供了前所未有的機會，讓人們能夠親身體驗資料工具。我們建議有志成為資料工程師的人在 AWS、Azure、Google Cloud Platform、Snowflake、Databricks 等雲端服務上設置帳戶。需要注意的是，其中許多平台都有提供免費的選項，但讀者在學習時應密切關注成本，並僅使用少量資料和單節點叢集來進行工作。

企業環境之外的人，要熟悉企業資料系統仍然很困難，這為那些尚未找到第一份資料工程工作的有志之士帶來了一定的障礙。本書可以幫上他們的忙。我們建議資料工程新手，先瞭解資料工程的基本概念和原則，然後查閱每章末尾「其他資源」中所列的資料。在第二次閱讀時，注意任何不熟悉的術語和技術。你可以利用 Google、維基百科、部落格文章、YouTube 影片和供應商網站來熟悉新術語，填補你對相關知識理解上的空白。

你將學到什麼以及它將如何提升你的能力

本書的目的是在幫助你建構解決真實世界之資料工程問題的堅實基礎。

閱讀完本書後，你將能夠瞭解：

- 資料工程如何影響你目前的角色（資料科學家、軟體工程師或資料團隊負責人）
- 如何避免被行銷宣傳所迷惑，選擇正確的技術、資料架構和流程
- 如何使用資料工程生命週期來設計和建構堅實的架構
- 資料生命週期每個階段的最佳作法

而且你將能夠：

- 在你目前的角色（資料科學家、分析師、軟體工程師、資料團隊負責人等）中融入資料工程原則
- 將多種雲端技術整合在一起，以滿足下游資料消費者的需求
- 使用端到端的最佳實踐框架來評估資料工程問題
- 在整個資料工程生命週期中融入資料治理和安全性

本書導覽

本書分為四個部分：

- 第一篇，「基本概念和構成要素」
- 第二篇，「資料工程生命週期深入解析」
- 第三篇，「安全性、隱私以及資料工程的未來」
- 附錄 A 和 B：分別涵蓋序列化和壓縮以及雲端網路

第一篇中，我們首先在第 1 章中定義資料工程，然後在第 2 章中描繪資料工程生命週期。在第 3 章中，我們討論了**良好的架構**（*good architecture*）。在第 4 章中，我們介紹了一個選擇合適技術的框架，雖然我們經常看到技術和架構被混為一談，但實際上它們是非常不同的主題。

第二篇以第 2 章為基礎，深入探討了資料工程生命週期；每個生命週期階段（資料產生、儲存、攝取、轉換和提供）都在獨立的章節中進行了介紹。第二篇可以說是本書的核心，其他章節存在的目的是為了支持這裡介紹的核心觀點。

第三篇涵蓋了其他主題。在第 10 章中，我們討論了**安全性**（*security*）和**隱私**（*privacy*）。雖然安全性一直是資料工程專業的重要組成部分，但隨著有利可圖之黑客活動（profit hacking）和國家級之網路攻擊（state sponsored cyber attacks）的興起，它變得更加重要。那麼隱私呢？企業隱私虛無主義的時代已經結束——沒有公司希望看到自己的名字出現在有關隨意處理隱私（sloppy privacy practices）的文章標題中。隨著 GDPR、CCPA 和其他法規的出現，對個人資料的草率處理也可能產生重大的法律後果。簡而言之，在任何資料工程工作中，安全性和隱私必須是首要的任務。

在從事資料工程工作、為本書進行研究並採訪眾多專家的過程中，我們思考了該領域近期和長期的發展方向。第 11 章概述了我們對資料工程未來的高度推測性想法。就其本質而言，未來是一個棘手的問題。時間會證明我們的一些想法是否正確。我們很想聽聽讀者對未來的看法與我們的看法有何一致或不同之處。

在附錄中，我們介紹了一些與資料工程的日常工作極其相關但不適合放入正文的主題。具體來說，工程師需要瞭解序列化和壓縮（見附錄 A），這樣既可以直接處理資料檔，也可以評估資料系統中的性能因素，而隨著資料工程轉向雲端，雲端網路（見附錄 B）成為了一個關鍵主題。

本書編排慣例

本書使用了以下的排版慣例：

斜體字

用於新術語、網址、電子郵件地址、檔名和副檔名。中文以楷體表示。

定寬字

用於程式列表，以及在段落內引用的程式元素，例如變數或函式名稱、資料庫、資料類型、環境變數、陳述句和關鍵字。

此圖示用於提示或建議。

第二章

資料工程生命週期

本書的主要目標是鼓勵你不要把資料工程視為資料技術的特定集合。資料領域正在經歷新資料技術和實踐方法的爆炸式成長，並且不斷提高抽象程度和易用性。由於技術抽象程度的提高，資料工程師將越來越成為**資料生命週期工程師**（*data lifecycle engineers*），根據資料生命週期管理的**原則**（*principles*）進行思考和操作。

在本章中，你將瞭解**資料工程生命週期**（*data engineering lifecycle*）的相關知識，這是本書的核心主題。資料工程生命週期是一個框架，用於描述資料工程「從開始到結束」（cradle to grave）的過程。你還將瞭解資料工程生命週期的潛在因素（undercurrents），它們是支援所有資料工程工作的關鍵基礎。

資料工程生命週期是什麼？

資料工程生命週期包括將原始資料轉化為有用之最終產品的各個階段，以供分析師、資料科學家、機器學習（ML）工程師和其他人員使用。本章將介紹資料工程生命週期的主要階段，重點放在每個階段的核心概念，並把細節保留到後面的章節。

我們將資料工程生命週期分為五個階段（圖 2-1，頂部）：

- 產生（Generation）
- 儲存（Storage）
- 攝取（Ingestion）
- 轉換（Transformation）
- 提供資料（Serving data）

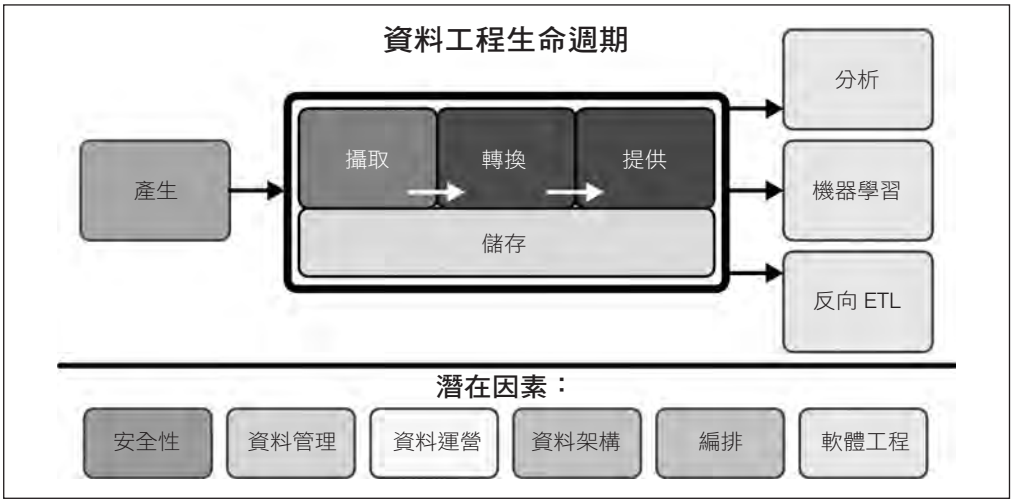

圖 2-1　資料工程生命週期的組成部分和潛在因素

我們透過從來源系統中獲取資料並保存它來開始資料工程生命週期。接下來，我們會對資料進行轉換，然後進行我們的核心目標，將資料提供給分析師、資料科學家、機器學習工程師和其他人員。實際上，因為資料從開始到結束都在流動，儲存在整個生命週期中都會發生——因此，如該圖所示，儲存「階段」（storage "stage"）是支撐其他階段的基礎。

通常，中間階段——儲存、攝取、轉換——可能有些混亂。沒關係。儘管我們將資料工程生命週期拆分成不同部分，但它並不總是一個整潔、連續的流程。生命週期的各個階段可能會重複出現、順序混亂、重疊或以有趣而意想不到的方式交織在一起。

作為基石的是貫穿資料工程生命週期多個階段的*潛在因素*（*undercurrents*）（圖 2-1，底部）：安全性、資料管理、資料運營（DataOps）、資料架構、編排和軟體工程。沒有這些潛在因素，資料工程生命週期的任何部分都無法充分發揮作用。

資料生命週期與資料工程生命週期的區別

你可能想知道整個資料生命週期和資料工程生命週期之間的區別。此二者之間存在一個微妙的區別。資料工程生命週期是整個資料生命週期的子集（圖 2-2）。完整的資料生命週期涵蓋了資料在整個生命週期中的各個階段，而資料工程生命週期則專注於資料工程師所控制的階段。

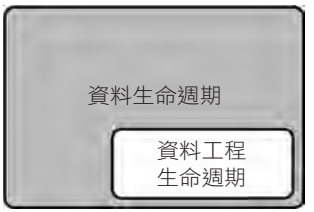

圖 2-2　資料工程生命週期是整個資料生命週期的子集

產生：來源系統

來源系統（*source system*）是資料工程生命週期中所使用資料的來源。例如，來源系統可以是 IoT 設備、應用程式訊息佇列或交易資料庫。資料工程師會從來源系統中提取資料，但通常並不擁有或控制來源系統本身。資料工程師需要對來源系統的工作方式、資料的產生方式、資料的產生頻率和速度以及所產生資料的種類有一定的瞭解。

工程師還需要與來源系統的擁有者保持良好的溝通，以瞭解可能會破壞管道（pipelines）和分析的任何變化。應用程式之程式碼可能會更改欄位中資料的結構，或者應用程式團隊甚至可能選擇將後端（backend）遷移到全新的資料庫技術。

資料工程中的一個主要挑戰是，工程師必須使用和瞭解令人眼花繚亂的資料來源系統。為了舉例說明，讓我們看一下兩個常見的來源系統，一個是非常傳統的（應用程式資料庫），另一個是比較新的（IoT swarms（物聯網群集））。

圖 2-3 展示了由一個資料庫支援多個應用程式伺服器的傳統來源系統。這種來源系統模式在 1980 年代隨著關聯式資料庫管理系統（RDBMS）的爆炸性成功而流行起來。「應用程式 + 資料庫」（application + database）模式今日在軟體開發實踐的各種現代演變中仍然很受歡迎。例如，應用程式通常由許多小型的服務 / 資料庫對（service/database pairs）與微服務（microservices）組成，而不是一個單體式應用程式。

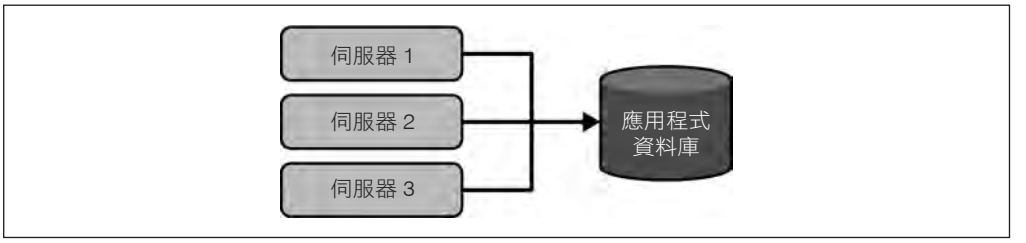

圖 2-3　來源系統範例：應用程式資料庫

讓我們看另一個來源系統的例子。圖 2-4 展示了一個物聯網群集（IoT swarm）：一個由許多設備（圓形）組成的系統，這些設備會向中央收集系統發送資料訊息（矩形）。隨著感測器（sensors）、智慧設備（smart devices）等物聯網設備（IoT devices）在各個領域的應用越來越廣泛，這種物聯網來源系統變得越來越普遍。

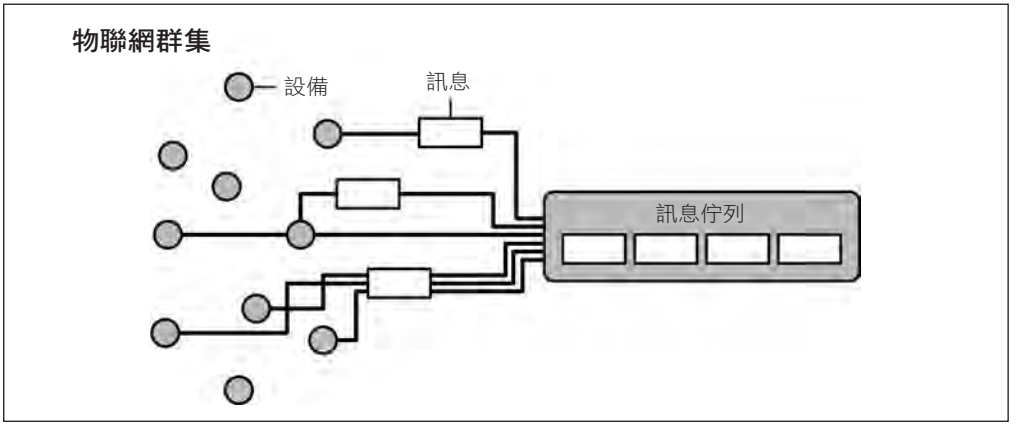

圖 2-4　來源系統範例：物聯網群集和訊息佇列

評估來源系統：關鍵的工程考慮因素

評估來源系統時，需要考慮許多因素，包括系統如何處理資料攝取、狀態和資料產生。以下是資料工程師必須考慮的一組來源系統評估問題：

- 資料來源的基本特徵是什麼？它是一個應用程式，還是一大群物聯網設備？
- 資料如何持久保存在來源系統中？資料是長期保留，還是臨時性的並且會很快被刪除？
- 資料產生的速率是多少？每秒有多少個事件？每小時有多少 GB ？
- 資料工程師期望從輸出資料中獲得什麼程度的一致性？如果你對輸出資料執行資料品質檢查，則資料不一致（例如，意外的空值、糟糕的格式設定…等等）的發生頻率如何？
- 錯誤多久發生一次？
- 資料是否包含重複內容？
- 某些資料值是否會延遲到達，是否比同時產生的其他訊息晚得多？
- 資料攝取的綱要（schema）是什麼？資料工程師是否需要跨多個資料表或甚至多個系統進行聯接（join），以獲得資料的完整圖像？
- 如果綱要發生變化（例如，添加了新欄位），該如何處理並將其傳達給下游利益相關者？
- 從來源系統提取資料應該多久一次？
- 對於有狀態系統（例如，追蹤客戶帳戶資訊的資料庫），資料是以定期快照（periodic snapshots）的形式提供，還是以異動資料擷取（change data capture，CDC）的更新事件（update events）提供？變更的執行邏輯是什麼，這些變更要如何在來源資料庫中進行追蹤？
- 將資料傳輸給下游消費者的資料提供者是誰 / 什麼？
- 從資料來源讀取資料是否會影響其性能？
- 來源系統是否有上游資料的依賴性？這些上游系統的特徵是什麼？
- 是否有資料品質檢查過程來檢測延遲或遺漏的資料？

來源產生的資料可供下游系統消費，包括人工產生的電子試算表、物聯網感測器以及 Web 和行動應用程式。每個來源都有其獨特的資料產生量和節奏。資料工程師應該知道來源如何產生資料，包括相關的特徵或細微差別。資料工程師還需要瞭解與他們互動的來源系統之局限性。例如，針對來源應用程式資料庫的分析查詢，是否會導致資源爭用和性能問題？

來源資料（source data）中最具挑戰性的細微差別之一是綱要。**綱要**（*schema*）定義了資料的層次組織。從邏輯上說，我們可以把資料的層次組織分為整個來源系統、個別的資料表以及各個欄位的結構。來源系統發送的資料的綱要可以透過不同的方式進行各種處理。有兩個常見的選項：無綱要（schemaless）和固定綱要（fixed schema）。

無綱要（*schemaless*）並不表示沒有綱要。相反地，它意味著應用程式在資料被寫入時定義綱要（schema），不論是寫入訊息佇列（message queue）、平面檔案（flat file）、二進位大型物件（Binary Large Object，blob）還是 MongoDB 之類的文件資料庫（document database）。建構在關聯式資料庫儲存（relational database storage）之上的更傳統模型，則使用在資料庫中強制實施的**固定綱要**（*fixed schema*），應用程式的資料寫入必須符合該綱要。

這兩種模型都給資料工程師帶來了挑戰。資料庫的綱要會隨時間而變化；事實上，在敏捷軟體開發方法中，鼓勵對資料庫的綱要進行修改和擴充。資料工程師職務的一個關鍵部分，是把來源系統綱要中的原始資料輸入轉換為有價值的分析輸出。隨著來源系統綱要的發展，此工作將變得更具挑戰性。

我們將在第 5 章更詳細地介紹來源系統；我們還將分別在第 6 章和第 8 章介紹綱要（schema）和資料建模（data modeling）。

儲存

你需要一個保存資料的地方。選擇儲存解決方案（storage solution）是成功完成資料生命週期（data lifecycle）其餘部分的關鍵，它也是資料生命週期中最複雜的階段之一，原因有很多。首先，雲端中的資料架構（data architectures）通常會利用多種儲存解決方案。其次，很少有資料儲存解決方案僅作為儲存之用，其中許多解決方案支援複雜的轉換查詢；即使是物件儲存解決方案，也可能支援強大的查詢功能，例如 Amazon S3 Select（*https://oreil.ly/XzcKh*）。第三，雖然儲

存是資料工程生命週期的一個階段，但它經常涉及其他階段，例如攝取、轉換和提供。

儲存（storage）涵蓋整個資料工程生命週期，通常出現在資料管道（data pipeline）中的多個地方，儲存系統與來源系統，攝取、轉換和提供（serving）相互交疊。在許多方面，資料的保存方式會影響資料工程生命週期中所有階段的使用方式。例如，雲端資料倉儲（cloud data warehouses）可以保存資料，在管道中處理資料，並將其提供給分析師。Apache Kafka 和 Pulsar 等串流框架（streaming frameworks）可以同時作為訊息的攝取、儲存和查詢系統，而物件儲存則是資料傳輸的標準層。

評估儲存系統：關鍵的工程考慮因素

以下是選擇資料倉儲（data warehouse）、資料湖泊（data lakehouse）、資料庫（database）或物件儲存（object storage）等儲存系統時，需要考慮的幾個關鍵的工程問題：

- 這個儲存解決方案是否與架構所需的讀寫速度相容？
- 儲存是否成為下游流程的瓶頸？
- 你是否瞭解這種儲存技術的工作原理？你是否以最佳方式利用儲存系統，還是在進行不自然的操作？例如，你是否在物件儲存系統中應用了高速率的隨機存取更新操作？（這是一種反模式，會帶來顯著的性能開銷。）
- 這個儲存系統是否能夠應付預期的未來規模？你應該考慮儲存系統的所有容量限制：總可用儲存空間、讀取操作速率、寫入量…等等。
- 下游用戶和流程是否按照所需的「服務水準協議」（service-level agreement，SLA）檢索資料？
- 你是否捕獲了有關綱要演進（schema evolution）、資料流（data flow）、資料沿襲（data lineage）等的中介資料（metadata）？中介資料代表著對未來的投資，能顯著提高資料的可發現性（discoverability）和機構知識（institutional knowledge），進而簡化未來的專案和架構變更。
- 這是一個純粹的儲存解決方案（物件儲存），還是支援複雜的查詢模式（例如，雲端資料倉儲）？