

雪花

权威指南

在 Snowflake Data Cloud 上构
建、设计和部署

乔伊斯·凯·阿维拉

Snowflake：权威指南

Snowflake 能够消除数据孤岛并从单一平台运行工作负载，这为数据分析的大众化创造了机会，使组织内各级的用户都能做出数据驱动的决策。无论您是从事数据仓库或数据科学工作的 IT 专业人员、业务分析师或技术经理，还是希望获得更多 Snowflake 平台实践经验的有抱负的数据专业人员，这本书都适合您。

您将了解 Snowflake 用户如何构建现代集成数据应用程序并基于数据开发新的收入来源。通过动手实践 SQL 示例，您还将了解 Snowflake Data Cloud 如何通过避免不必要的重新构建平台或迁移数据来帮助您加速数据科学。

您将能够：

- 以惊人的速度高效捕获、存储和处理大量数据
- 以结构化和半结构化格式摄取和转换实时数据馈送，并在几分钟内提供有意义的数据洞察
- 使用 Snowflake Time Travel 和零拷贝克隆来制定合理的数据恢复策略，以平衡系统弹性与持续的存储成本
- 通过访问 Snowflake Marketplace 中提供的随时可查询的数据集，安全地共享数据并降低或消除数据集成成本

Joyce Kay Avila 是 Snowflake 数据超级英雄，也是 Snowflake 合作伙伴咨询公司的 Snowflake 实践经理。她是一名注册会计师，完成了会计信息系统的博士课程；她还拥有 MBA 学位以及会计和计算机科学学士学位。

DATA

US \$79.99 CAN\$99.99 ISBN: 978-1-098-10382-8



“乔伊斯的书增加了超越其他 Snowflake 书籍或公共资源的见解和实践知识，为理解并探索 Snowflake 的可能性艺术开辟了新的场所。”

——康斯坦丁·斯坦卡
首席解决方案工程师
在 Snowflake

“这确实是一个权威的指南；我喜欢它的每一点，几乎每一页都学到了新东西。这是 Snowflake 上的 SELECT *，是每个人的必读之作。”

——杰诺·亚马
Analytics 工程师和 Snowflake
数据超级英雄，Canva

推特: @oreillymedia
[linkedin.com/company/oreilly-media/](https://www.linkedin.com/company/oreilly-media/) [youtube.com/oreillymedia](https://www.youtube.com/oreillymedia)

对 Snowflake: The Definitive Guide 的赞誉

乔伊斯的书增加了超越其他 Snowflake 书籍或公共资源的见解和实践知识，为理解
和探索 Snowflake 的可能性艺术开辟了新的场所。

—Constantin Stanca, Snowflake 首席解决方案工程师

这确实是一本权威的指南，我喜欢它的每一点，几乎每一页都学到了一些新东西。它
确实是 Snowflake 上的 SELECT *, 是每个人的必读之作。

—Jeno Yamma, Canva 分析工程师兼
Snowflake 数据超级英雄

学习和做得更好的最好方法是通过实践。这本书是一本写得很好且易于遵循的 Snowflake 指
南。但真正让它与众不同的是所有实际的代码示例和知识检查。乔伊斯教我们理论，鼓励我
们实践，然后测试我们的知识。无论您是刚刚开始探索 Snowflake 还是有经验的人，我保证
您都会从这本书中学到一些东西。

—Veronika Durgin, 数据和分析主管

您进入 Snowflake Data Cloud 的第一站。开始使用 Snowflake? 让 Snowflake Data
超级英雄 Joyce Kay Avila 引导您进入 Data Cloud。

—Daan Bakboord, 数据和分析顾问

如果您刚刚开始 Snowflake Data Cloud 的学习之旅，我强烈推荐这本可读性强、引人入胜且全面的书。它通过清晰的解释和知识检查来教授核心概念，并通过分步说明和代码帮助您获得实践经验。

—Slim Baltagi, Snoable, LLC 创始人

终于，这本书包含了您想知道的有关 Snowflake Data Cloud 的所有信息。

乔伊斯（Joyce）以简单的对话风格撰写了《雪花：权威指南》
(Snowflake: The Definitive Guide)，将吸引广泛的读者。

—In516ht 高级顾问 Maja Ferle

Snowflake：权威指南

架构、设计和部署

在 Snowflake Data Cloud 上

乔伊斯·凯·阿维拉

北京 波士顿 塞瓦斯托波尔 法纳姆酒店 | 东京 | 北京

O'REILLY®

Snowflake: 权威指南

作者: 乔伊斯·凯·阿维拉

版权所有 © 2022 乔伊斯·凯·阿维拉。保留所有权利。

在美国印刷。

由 O'Reilly Media, Inc. 出版, 地址为 1005 Gravenstein Highway North, Sebastopol, CA 95472。

购买 O'Reilly 书籍可用于教育、商业或促销用途。大多数标题 (<http://oreilly.com>) 也提供在线版本。如需更多信息, 请联系我们的企业/机构销售部门: 800-998-9938 或 corporate@oreilly.com。

收购编辑: Jessica Haberman
开发编辑: Michele Cronin 制作
编辑: Clare Laylock 文案编辑:
校对: Audrey Doyle
Justin Billing

索引器: Potomac Indexing, LLC
室内设计师: David Futato
封面设计: Karen Montgomery
插画师: Kate Dullea

2022 年 8 月: 第一版

第一版的修订历史

2022-08-10: 首次发布 2022-12-16: 第二次发布

有关版本详细信息, 请参阅 <http://oreilly.com/catalog/errata.csp?isbn=9781098103828>。

O'Reilly 徽标是 O'Reilly Media, Inc. 的注册商标。Snowflake: The Definitive Guide、封面图片和相关商业外观是 O'Reilly Media, Inc. 的商标。

本作品中表达的观点是作者的观点, 不代表出版商的观点。虽然出版商、作者和 Snowflake 已尽善意努力确保本作品中包含的信息和说明准确和最新, 但出版商、作者和 Snowflake 不对错误或遗漏承担任何责任, 包括但不限于对因使用或依赖本作品而造成的损害的责任。使用本作品中包含的信息和说明的风险由您自行承担。Snowflake 建议您利用此处托管的文档来获取基于 Snowflake 的最新信息。如果本作品包含或描述的任何代码示例或其他技术受开源许可或他人的知识产权的约束, 您有责任确保您对它们的使用符合此类许可和/或权利。

这项工作是 O'Reilly 和 Snowflake 合作的一部分。请参阅我们的编辑独立性声明。

978-1-098-10382-8

[LSI]

目录

前言 xiii

1. 入门 1 Snowflake Web 用户界面 3

准备工作	3
Snowsight 方向 6 Snowsight 首选项 6 导航 Snowsight 工作表 8 上下文设置	

提高生产力 14 Snowflake 社区 17 Snowflake 认证 19 Snowday 和 Snowflake 峰会活动 19 关于书中代码示例的重要注意事项 19 代码清理	8
--	---

总结	21
知识检查	22

2. 创建和管理 Snowflake 架构

准备工作	23
传统数据平台架构 24 共享磁盘（可扩展）架构 24 无共享（可扩展）架构 25 NoSQL 替代方案 25 雪花架构 26 云服务层 27 管理云服务层 28 云服务计费层 28 查询处理（虚拟仓库）计算层 29 v	

虚拟仓库大小	30	纵向扩展虚拟仓库以处理大量数据和复杂查询	31	使用多集群	
虚拟仓库进行横向扩展以最大限度地提高并发性					
				35	
创建和使用虚拟仓库	38	工作负载分离和工作负载管理	42	虚拟仓库层的计费	44
集中式（混合列式）数据库存储层	45	零副本克隆简介	46	时间旅行简介	46
存储层的计费	46	Snowflake 缓存			
					46
查询结果缓存	47	元数据缓存			48
Virtual Warehouse Local Disk Cache	49	代码清理			50
总结					50
知识检查					51
3. 创建和管理 Snowflake 安全数据库对象	53				
准备工作					54
创建和管理 Snowflake 数据库	54	创建和管理 Snowflake 架构	63		
INFORMATION_SCHEMA	65	ACCOUNT_USAGE 架构	69	架构对象层次结构	70
Snowflake 表简介	70	创建和管理视图	76	Snowflake 阶段简介：包含的文件格式	
79 使用存储过程和 UDF 扩展 SQL	82	用户定义的函数（UDF）：包含的任务	84		
返回表格值的安全 SQL UDTF（购物篮分析示例）					
					87
存储过程					89
管道、流和序列简介	95	个 Snowflake 流（深入探讨）	96	个 Snowflake 任务	
（深入探讨）	102	代码清理			
					108
总结					108
知识检查					109

4. 探索 Snowflake SQL 命令、数据类型和函数	111
准备工作	112
在 Snowflake DDL 命令中使用 SQL 命令	113
DCL 命令	113
DML 命令	114
TCL 命令	114
DQL 命令	114
Snowflake 中的 SQL 查询开发、语法和运算符	115
SQL 开发和管理	115
查询语法	117
查询运算符	124
长时间运行的查询以及查询性能和优化	125
Snowflake 查询限制	126
Snowflake 支持的数据类型简介	126
数值数据类型	127
字符串和二进制数据类型	129
日期和时间输入/输出数据类型	130
半结构化数据类型	131
非结构化数据类型	137
Snowflake 如何支持非结构化数据的使用	138
Snowflake SQL 函数和会话变量	141
使用系统定义的（内置）函数	141
创建 SQL 和 JavaScript UDF 和使用会话变量	141
外部函数	144
代码清理	144
总结	145
知识检查	145
5. 利用 Snowflake 访问控制	147
准备工作	148
创建 Snowflake 对象	150
Snowflake 系统定义的角色	154
创建自定义角色	155
功能级业务和 IT 角色	157
系统级服务帐户和对象访问角色	158
角色层次结构分配：为其他角色分配角色	159
为角色授予权限	162
为用户分配角色	165
测试和验证我们的工作	166
用户管理	166
角色管理	169
Snowflake 多账户策略	176
	178

使用 SCIM	178	代码清理管理用户和组	179
总结	180		
知识检查	180		
6. 数据加载和卸载	183		
准备工作	184		
数据加载和卸载基础知识	184	数据类型	185
文件格式	185		
数据文件压缩	189	数据处理频率	189
		雪花阶段参考	190
		数据源	
			191
数据加载工具	192	Snowflake 工作表 SQL 使用 INSERT INTO 和 INSERT ALL 命令	
			192
Web UI 加载数据向导	204	SnowSQL CLI SQL PUT 和 COPY INTO 命令	210
		数据管道	
			212
第三方 ETL 和 ELT 工具	221	加载数据的替代方案	222
		卸载数据的工具	222
Snowflake 数据工程师的数据加载最佳实践	223	选择正确的数据加载工具并考虑适当的数据类型选项	
			224
避免逐行数据处理	224	选择合适的 Snowflake 虚拟仓库大小并根据需要拆分文件	
			225
分步转换数据并使用临时表进行中间结果	225	代码清理	
			225
总结	225		
知识检查	226		
7. 实施数据治理、账户安全以及数据保护和恢复。	.227		
准备工作	228		
Snowflake 安全性	230		
使用 Snowflake ACCESS_HISTORY Account Usage 视图控制账户访问	231	监控活动	
			235
数据保护和恢复	237	复制和故障转移	243
		利用数据管理控制实现数据大众化	244

INFORMATION_SCHEMA Data Dictionary	245
对象标记	245
分类	249
数据掩码	251
行访问策略和行级安全性	253
外部标记化	256
安全视图和 UDF	257
对象依赖关系	257
代码清理	
总结	258
知识检查	258
8. 管理 Snowflake 帐户成本	261
准备工作	262
Snowflake 月度账单	262
存储费	264
数据传输成本	264
消耗的计算积分	265
创建资源监视器以管理虚拟仓库使用情况并降低成本	
资源监控器积分配额	268
资源监控器积分使用情况	268
资源监控器通知和其他操作	268
分配的资源监控规则	269
用于创建和管理资源监控器的 DDL 命令	271
对成本中心使用对象标记	276
查询ACCOUNT_USAGE 视图	276
使用 BI 合作伙伴控制面板监控 Snowflake 的使用情况和成本	277
Snowflake 敏捷软件交付	278
为什么需要 DevOps ?	278
持续数据集成、持续交付和持续部署	279
什么是数据库变更管理?	279
如何使用零拷贝克隆来支持开发/测试环境	281
代码清理	286
总结	284
知识检查	285
9. 分析和改进 Snowflake 查询性能	287
准备工作	287
分析查询性能	287
QUERY_HISTORY 分析	288
HASH () 函数	288

Web UI 历史记录	289
了解 Snowflake 微分区和数据聚类分析	291
分区解释	291
Snowflake 微分区解释	294
Snowflake 数据聚类分析	299
聚类宽度和深度	299
选择聚类键	303
创建聚类键	305
重新聚类	
具体化视图的性能优势	306
探索其他查询优化技术	308
搜索优化服务	308
查询优化技术比较	309
总结	
代码清理	310
知识检查	310
10. 配置和管理安全数据共享 313	
Snowflake 架构数据共享支持	314
Snowgrid 的力量	314
数据共享使用案例	314
Snowflake 对统一 ID 2.0 的支持	315
Snowflake 安全数据共享方法	316
准备工作	
Snowflake 的直接安全数据共享方法	318
创建出站共享	318
Snowflake 数据使用者如何使用入站共享	330
如何在公共 Snowflake 市场上列出和购物	333
提供商的 Snowflake 市场	335
标准与个性化数据列表	337
利用 Snowflake 私有数据交换的力量	340
Snowflake 数据洁净室	341
重要设计，安全性和性能注意事项	
共享设计注意事项	342
共享安全注意事项	343
共享性能注意事项	343
数据库共享和数据库克隆之间的区别	343
数据共享和时间旅行注意事项	344
数据共享共享	344
摘要	
代码清理	344
知识检查	345

11. 在 Snowsight 347 准备工作中可视化数据

	347
Snowsight 中的数据采样 348 基于特定行数的固定大小采样 349 基于概率的基于分数的采样 349 预览字段和数据 349 采样示例 352 使用自动统计和交互式结果 353 Snowsight 控制面板可视化 358 创建控制面板和磁贴 358 使用图表可视化 361 聚合和分桶数据 363 编辑和删除磁贴 366 协作	

12. Snowflake Data Cloud 371 的工作负载

准备工作	372
数据工程	372
数据仓库	374
Data Vault 2.0 建模 374 在 Snowflake 377 数据湖中转换数据	

数据协作 378 数据货币化 379 数据共享的法规和合规性要求 379 数据分析	377
---	-----

金融行业的高级分析 380 医疗保健行业的高级分析 381 制造业和物流服务的高级分析 382 零售垂直行业以及通信和媒体行业的营销分析 382 数据应用程序	380
---	-----

数据科学 383	
雪地公园 385	
Streamlit (流式) 388	
使用 Snowflake 作为安全数据湖的网络安全 388 克服纯 SIEM 架构的挑战 389 搜索优化服务与集群 392	

Unistore 联卖店	401
事务工作负载与分析工作负载 401 混合表	402
总结	403
代码清理	403
知识检查	404
A. 知识检查问题 405 的答案	
B. Snowflake 对象命名最佳实践	423
C. 设置 Snowflake 试用账户	427
索引	431

前言

这本书的由来

早在 2020 年 4 月，我在 YouTube 上发布了一个关于 Snowflake 的视频系列，以帮助人们获得 Snowflake 认证。我从该系列中收到的回应证实了对此类知识的需求远远超出了视频系列所能满足的需求。然而，Snowflake 的旅程早在本视频系列之前就开始了。

Snowflake 在以隐身模式运行三年后，于 2015 年突然出现在公众视野中。

Snowflake 的客户涵盖多个不同的行业，包括公共部门、医疗保健、零售、金融服务等。2017 年，Capital One 成为 Snowflake 的第一个虚拟专用 Snowflake (VPS) 客户，并向 Snowflake 投资了 5 百万美元，以帮助 Snowflake 进一步提高向金融服务行业提供创新技术的能力。2018 年，Snowflake 的客户群增长了 300%，新增的客户群包括几家金融服务公司。

大约在这个时候，也就是 2018 年，我第一次被介绍给 Snowflake，当时我受雇于一家金融服务公司。作为 Salesforce 开发人员，我在 Salesforce 中设计并实施了数据模型，然后团队的其他成员在 Snowflake 中构建了这些数据模型。

我一直对数据充满热情，精通数据建模，热爱学习，尤其是学习令人兴奋的新技术。所以很自然地，我对 Snowflake 了解得越多，我就越想知道。在整个 2018 年和 2019 年，我扩展了我对 Snowflake 的了解，同时继续获得更多的 Salesforce 经验。

2019 年底，我对 Snowflake 的热情如火如荼，我决心获得 Snowflake 认证。

当时，Snowflake 学习指南本身就不存在，也没有足够的资源来准备认证考试。因此，我完成了 Snowflake 大学的每个培训模块，并一页又一页地阅读了 Snowflake 文档。为了帮助我更好地准备，我制作了一组学习笔记，将内容分为特定的主题。这些笔记最终帮助我在 4 月份获得了 Snowflake 认证。

2020，使用这些笔记创建一系列视频是有意义的，这样其他人也可以从中受益。我做梦也没想到这会是一段令人惊奇的 Snowflake 之旅的开始。

在 2020 年创建 Snowflake YouTube 视频系列使我成为了 Snowflake Data 超级英雄。从那时起，我被邀请在 Snowflake 活动中发表演讲，并更多地参与 Snowflake 社区。我对该平台的了解不断增长。与此同时，许多观看我的 YouTube 视频的人伸出援手，让我知道他们有多欣赏这些内容。

当 2021 年初作者 Snowflake : The Definitive Guide for O'Reilly 的机会出现时，我全身心地投入到写这本书的漫长旅程中。我对 O'Reilly 赋予作者如此多的自由来塑造这本书的内容感到惊讶。伴随着这种令人难以置信的特权而来的是很多责任，要选择正确的主题并决定深入研究任何一个特定领域。

这本书旨在成为一本权威的指南，这意味着我在编写时需要涵盖大量信息。在我花在写作和重写上的整个时间里，我不得不依靠其他人来填补我的知识空白，以便我能够提供最好的内容。这些人以及许多其他人是使这本书成为现实不可或缺的一部分。我在下面的致谢不言自明；如果我没有利用这个机会单独感谢那些真正对这本书的结果产生影响的人，那将是我的失职。

这本书适合谁？

这本书是数据仓库、数据科学和数据分析领域的技术从业者以及任何希望成为 Snowflake 认证专业人士的人的必读之作。对于数据利益相关者（如 IT 团队负责人、技术经理和技术团队的负责人）或希望与现代数据技术和趋势保持同步的人，本书也有很多价值。

虽然不需要了解非关系数据库和其他数据相关工具，但您至少应该熟悉关系数据库和 SQL。如果您在以下任何数据相关角色中至少有一到两年的经验，您将充分利用本书：

- 数据架构师 • 数据工程师 • 数据分析师

- 数据科学家 • 数据库管理员 • 技术经理

本书的目标

这本书包含大量动手示例，可帮助您建立扎实的 Snowflake 基础知识。

在本书的结尾，您将了解：

- 独特的 Snowflake 架构如何以惊人的速度有效地捕获、存储和处理大量数据 • 如何快速摄取和转换结构化和半结构化格式的实时数据源，并在几分钟内提供有意义的数据见解 • 如何使用时间旅行和零拷贝克隆来产生合理的数据接收策略，以平衡系统弹性需求和持续的存储成本 • 如何安全地共享数据以及如何通过访问 Snowflake Data Marketplace 中提供的最新的、随时可查询的数据集，降低或消除数据集成成本

您将能够：

- 部署、调整和监控虚拟仓库，以最大限度地提高查询性能和吞吐量，同时控制成本
- 实施基于角色的访问控制系统，使用动态数据掩码，并利用自主访问控制和安全视图来保护和限制数据访问 • 使 Snowsight 能够支持数据分析师在新的 Snowflake Web 界面中更高效地工作 • 遵循 Snowflake 最佳实践并避免常见陷阱，以实现世界级结果

浏览本书

章节的顺序旨在确保前面的章节提供必要的基础知识，以便您可以从后面的章节中获得最大的价值。也就是说，每一章的例子都是独立的。这些章节从该章所需的任何准备工作开始，然后引导您逐步完成动手实践

例子。在每章的末尾，都有清理说明。在开始新章节之前，您无需完成前几章中的任何示例。

前七章是基础章节。接下来的两章将带您了解管理 Snowflake 成本和提高性能的详细信息。第 10 章深入探讨了安全数据共享，这是 Snowflake 的主要区别之一。第 11 章重点介绍在 Snowsight 中可视化数据。您会注意到，除了少数例外，本书中的示例都是在新的 Snowsight Web 用户界面中完成的，尽管所有代码都可以在经典控制台中成功运行。第 12 章介绍了所有不同的 Snowflake 工作负载，包括新的 Unistore 工作负载。

每章的末尾都有一个知识检查部分。这些问题的答案可以在附录 A 中找到。

Snowflake 对象命名最佳实践包含在附录 B 中。最后，您可以在附录 C 中找到有关设置 Snowflake 试用账户的说明。

使用代码示例

补充材料（代码示例、练习等）可在

<https://github.com/SnowflakeDefinitiveGuide> 下载。

如果您有技术问题或使用代码示例时遇到问题，请发送电子邮件至
bookquestions@oreilly.com。

这本书旨在帮助您完成工作。一般来说，如果本书提供了示例代码，您可以在您的程序和文档中使用它。除非您要复制代码的很大一部分，否则您无需联系我们以获得许可。例如，编写使用本书中的多个代码块的程序不需要许可。销售或分发 O'Reilly 书籍中的示例确实需要获得许可。通过引用本书和引用示例代码来回答问题不需要许可。将本书中的大量示例代码合并到您的产品文档中确实需要许可。

我们赞赏，但通常不要求署名。归属通常包括标题、作者、出版商和 ISBN。例如：“雪花：乔伊斯·凯·阿维拉（O'Reilly）的定义指南。版权所有 2022 乔伊斯·凯·阿维拉，978-1-098-10382-8。”

如果您认为您对代码示例的使用超出了合理使用或上述许可的范围，请随时通过
permissions@oreilly.com 与我们联系。

本书中使用的约定

本书使用以下排版约定：

斜体的

指示新术语、URL、电子邮件地址、文件名和文件扩展名。

恒定宽度

用于程序列表，以及在段落中引用程序元素，例如变量或函数名称、数据库、数据类型、环境变量、语句和关键字。

恒定宽度粗体

显示应由用户按字面键入的命令或其他文本。

Constant width 斜体

显示应替换为用户提供的值或由上下文确定的值的文本。



此元素表示提示或建议。



此元素表示一般说明。



此元素表示警告或注意。

O'Reilly 在线学习

O'REILLY® 40 多年来，O'Reilly Media 一直提供技术和商业培训、知识和洞察力，以帮助公司取得成功。

我们独特的专家和创新者网络通过书籍、文章和我们的在线学习平台分享他们的知识和专业技能。O'Reilly 的在线学习平台让您按需访问实时培训课程、深入学习路径、交互式编码环境，以及来自 O'Reilly 和其他 200+ 其他出版商的大量文本和视频。有关更多信息，请访问 <https://oreilly.com>。

如何联系我们

请将有关本书的意见和问题提交给出版商：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472 800-998-9938 (美国
或加拿大) 707-829-0515 (国际或本地)
707-829-0104 (传真)

我们有一个本书的网页，其中列出了勘误表、示例和任何其他信息。您可以通过 <https://oreil.ly/snowflake-the-definitive-guide> 访问此页面。

向 bookquestions@oreilly.com 发送电子邮件，以评论或询问有关本书的技术问题。

有关我们的书籍和课程的新闻和信息，请访问 <https://oreilly.com>。

在 LinkedIn 上找到我们：<https://linkedin.com/company/oreilly-media>

在 Twitter 上关注我们：<https://twitter.com/oreillymedia>

在 YouTube 上观看我们：<https://www.youtube.com/oreillymedia>

确认

写一本 O'Reilly 书的旅程是我一生中最具挑战性但最有益的经历。在这段令人难以置信的旅程中，我从未孤单。我很幸运能有一群人，他们每个人都在向世界传播这本书的过程中发挥了重要作用。

我承认，一开始，我对这个过程的起伏不是很有信心。分享那些早期版本的手稿很不舒服，我真的不想这样做。我的开发编辑 Michele Cronin 说服我相信这个过程。我最衷心的感谢和赞赏 Michele 如此支持和耐心，有时在需要时温柔地推动我前进。

Michele 只是与 O'Reilly Media 相关的一群了不起的人之一，他们帮助这本书成为现实。

我感谢文案编辑 Audrey Doyle 和校对员 Justin Billing，他们花了无数时间不止一次地阅读了这本书的许多页面。我感谢 Kristen Brown 在 O'Reilly 平台上管理早期发布流程，以尽快向用户提供内容。我还要感谢 Karen Montgomery 令人惊叹的封面设计、David Futato 的室内设计以及 Kate Dullea 的插图，它们真正为这本书带来了美感和支持职业主义。

O'Reilly 书籍所经历的质量控制水平是首屈一指的。O'Reilly 团队花费了无数小时梳理所有内容，即使是最小的细节，也使这本书完美无缺。因此，特别感谢 Clare Jensen 付出的非凡努力。

当然，如果没有 Jess Haberman，我作为作家的旅程永远不会开始，她相信我可以利用我对 Snowflake 的热情，在漫长的白天、黑夜和周末保持动力，使这本书栩栩如生。我还要感谢 Andy Kwan 在撰写本书的同时对 O'Reilly 机会的支持，以及 Suzanne Huston 为帮助营销和支持这本书所做的努力。其他来自 O'Reilly 的人曾参与过我的旅程，包括 Cassandra Furtado、Charlotte Ames、Elizabeth Kelly、Joan Baker 和 Shannon Cutt。

我也有一群了不起的审稿人，他们经历了那些非常早期的草稿。他们在早期阶段的反馈帮助塑造了这本书的内容。早期的评论者包括 Jacob Thomas 和 Randy Pitcher II。特别感谢审稿人 Veronika Durgin 和 Daan Bakboord，他们提供了详细的反馈，并花时间实际运行所有代码，并防止一些错误进入本书的最终版本。在两轮单独的审核中，我收到了 Snowflake 首席解决方案架构师 Michael Rainey 的宝贵反馈。

我感谢许多 Snowflake 数据超级英雄，包括 Slim Baltagi、Maja Ferle 和 Jeno Yamma；在很短的时间内，他们浏览了这本书的完整预览版并提供了反馈。我很荣幸能成为这样一个了不起的 Snowflake 数据超级英雄团队的一员，并且非常感谢每一位给予支持的 Snowflake 数据超级英雄。由 Elsa Mayer 和 Howard Lio 领导的 Snowflake 数据英雄社区确实具有一些神奇之处。终于在 2022 年雪花峰会上第一次见到这么多人，真是太棒了。感谢 Jena Donlin 真诚的衷心支持。还有 Dash Desai，你知道我喜欢看到你穿那双鞋！

任何人都可以看到，Snowflake 数据英雄社区的价值观直接反映了公司的核心价值观。“包容和协作，将人和想法聚集在一起”是 Snowflake 核心价值观的一个例子，在我与 Snowflake 员工的所有互动中，我都亲身证明了这一点，而不仅仅是从支持 Snowflake 的数据超级英雄的角度来看。作为一名与 Snowflake 产品团队合作以帮助补充知识的作者，我一直在

被介绍给许多非常聪明和有才华的人。在将近一年的时间里，Aleks Todorova 孜孜不倦地工作，将我与公司内部许多不同的人联系起来，他们可以帮助我扩展我的 Snowflake 知识，以便我能够为这本书提供更好的内容。作为与 Snowflake 客户经理合作为客户提供服务的 Snowflake 咨询合作伙伴，我也有幸结识了更多的 Snowflake 员工。

对于他们一直以来的支持，我要感谢几位现任 Snowflake 员工，以及我已经提到的 Michael、Elsa、Howard、Jena 和 Dash。名单很长！这里按名字的字母顺序排列，只是我想提到的一些 Snowflake 员工：Alex Gutow、Carl Perry、Cassie AgenoWallgren、Chris Keithley、Constantin “Cristi” Stanca、Daniel Myers、Danny Shea、Diane Elinski、Emily Dillon、Emily Lin、Eric Feng、Felicia Dorng、Felipe Hoffa、Fran-cis 毛、Ganesh Subramanian、Ines Marjanovic、Julian Forero、贾斯汀·朗塞斯、凯特·贝斯佩尔、凯蒂·埃克伦德、凯莉·黄、利斯·达拉舍、丽莎·卢斯卡普、玛丽莲·谭、迈克·米勒、尼克·阿金西拉、奥马尔·辛格、帕特里克·古巴、菲利普·科莱蒂、菲利普·欧文、拉南·萨亚格、瑞恩·奥尔德里奇、桑杰·卡蒂马尼、索林·沙阿、斯科特·蒂尔、张思玲、顾诗仪、Shravan Narayen、汤姆·米查姆、特拉维斯·考夫曼、威廉·富恩特斯和查泽尔。

我还要感谢我的雇主 SpringML。在他们的支持下，我能够有效地平衡一份要求苛刻的全职工作和我为世界上最好的技术书籍出版商写作的愿望。我非常感谢 Robert Anderson，他招募我加入 SpringML，是我最大的支持者。帮助发展 SpringML 的 Snowflake 实践也让我有机会花时间了解我们的首席技术官 Girish Reddy，他似乎从不厌倦地扩展他对 Snowflake 等创新技术的了解。Janeesh Jayaraj、Sriram Manda - dapu、Tamera Fall、Vineesh Sriramoju 和 Vishal Deo 只是我有幸在 Snowflake 项目中合作过的另外几个了不起的团队成员。SpringML 团队中有很多成员，他们都对 Snowflake 感到兴奋，并且非常支持我成为 O'Reilly 出版商的个人旅程。感谢 SpringML 团队让我与您分享这段旅程。我感谢你们所有人。

可以想象，如果没有家人的支持，完成这本书是不可能的。十五个多月来，他们通过每天承担更多的家务来弥补空缺，他们鼓励我在认为自己做不到的时候继续前进。一年多来，我的旅程就是他们的旅程。

对我不可思议的孩子 Alanna 来说，你是天使。没有你，我的生活就不会有阳光。我的心充满了希望和乐观，因为你对我坚定不移的信心。你不断看到我最好的一面，这让我想成为一个更好的人，一个更好的母亲。我希望你永远认为我是有史以来最酷的妈妈。

我爱你，Alanna Kay Avila。

对我丈夫罗伯特来说，因为有你，我们家就是家。感谢你读了这本书的每一个字，通常是很遍，并在家里承担了大部分责任，使我能够专注于写作。从书的初稿到初印，您始终是第一个阅读每一章的人。对于每一章，我相信您会给我诚实的反馈，并在我与他人分享之前帮助我改进它。我们所做的一切都是一支出色的团队。你是我最好的朋友，罗伯特，我爱你。

对于我们了不起的狗 Zelda，感谢您提醒我暂停一下。在过去的一年里，有很多次你希望我停下来玩，但我做不到。我承诺在来年弥补你。

感谢我美丽的母亲 Carolyn Kay Hare，感谢你在很久以前播下了种子，让我变得坚强和无所畏惧，即使我有时担心自己会达不到要求。你帮我从字典里删掉了“can't”这个词，这让一切变得不同了。

对于读者，感谢您选择这本书。让我们一起明确 #MakeItSnow！

第一章

开始

云计算的特点是数据存储和计算能力的按需可用性。云计算的一个主要好处是它不需要用户直接或积极参与这些计算机系统资源的管理。其他好处包括访问无限的存储容量、自动软件更新、即时可扩展性、高速和降低成本。正如预期的那样，最近由 AWS Redshift、Google BigQuery 和 Microsoft Azure Data Warehouse 引领的云计算爆炸式增长导致本地数据中心的衰落。

许多主要的数据仓库提供商，例如 Oracle 和 IBM，都是作为传统托管解决方案创建的，后来适应了云环境。与那些传统解决方案不同，Snowflake 是从头开始为云原生构建的。虽然 Snowflake 最初是一个颠覆性的云数据仓库，但它随着时间的推移而发展，如今它已不仅仅是一个创新的现代数据仓库。

一路走来，Snowflake 赢得了一些令人印象深刻的认可。Snowflake 在 2015 年 Strata + Hadoop World 创业大赛中获得第一名，并在 Gartner 的 2015 年 DBMS 魔力象限报告中被评为“最酷供应商”。2019 年，Snowflake 在福布斯杂志的 Cloud 100 榜单中排名第二，在 LinkedIn 的美国顶级初创公司榜单中排名第一。2020 年 9 月 16 日，Snowflake 成为历史上最大的软件首次公开募股（IPO）。

如今，Snowflake Data Cloud Platform 打破了孤岛，并支持许多不同的工作负载。除了传统的数据工程和数据仓库工作负载外，Snowflake 还支持数据湖、数据协作、数据分析、数据应用程序、数据科学、网络安全和 Unistore 工作负载。Snowflake 的“多个数据工作负载，一个平台”方法为组织提供了一种方法，以安全且受监管的方式从快速增长的数据集中快速获取价值，从而支持

“一个平台”方法为组织提供了一种方法，以安全且受监管的方式从快速增长的数据集中快速获取价值，使公司能够满足合规性要求。自 10 年前成立以来，Snowflake 一直在数据云中保持快速创新。

Snowflake 的创始人于 2012 年首次聚集在一起，其愿景是从头开始构建一个云数据仓库，从大量不同类型的数据中释放无限见解的真正潜力。他们的目标是构建此解决方案，使其安全、强大，但经济高效且易于维护。仅仅三年后的 2015 年，Snowflake 的云构建数据仓库开始商用。Snowflake 立即以其独特的架构和与云无关的方法颠覆了数据仓库市场。颠覆性的 Snowflake 平台表单还使数据工程更加面向业务、技术含量更低、耗时更少，这通过允许组织内各级用户做出数据驱动的决策，为数据分析的大众化创造了更多机会。

要了解 Snowflake 的独特品质和方法，了解 Snowflake 的底层架构非常重要。从第 2 章开始，一直到整本书，您将发现 Snowflake 提供近乎零的管理功能的多种方式，以消除与传统数据仓库相关的大部分管理和管理开销。您将亲身体验 Snowflake 的工作原理，因为每章都包含您可以亲自尝试的 SQL 示例。每章末尾还有知识检查。

一些最具创新性的 Snowflake 工作负载依赖于 Snowflake 于 2018 年推出的安全数据共享功能。Snowflake Secure Data Sharing 支持在您的业务生态系统中几乎即时安全地共享受管控数据。它还为数据资产的货币化开辟了许多可能性。第 10 章完全介绍了这种创新的 Snowflake 工作负载。

2021 年，Snowflake 推出了使用 Snowflake Organizations 轻松管理多个账户的功能。管理多个帐户可以独立维护不同的环境，例如开发和生产环境，并采用多云战略。这也意味着您可以更好地管理成本，因为您可以为每个单独的账户选择所需的功能。我们将在第 5 章中探讨 Snowflake 组织管理。

Snowflake 还在 2021 年 6 月推出了 Snowpark，扩展了数据云中的可能性范围。在整个行业中，Snowpark 被公认为数据工程和机器学习领域的游戏规则改变者。

Snowpark 是一个开发人员框架，它为云带来了新的数据可编程性，并使开发人员、数据科学家和数据工程师能够使用 Java、Scala 或 Python 以无服务器方式部署代码。

Snowflake 于 2022 年推出的安全数据湖是一种创新的工作负载，它使网络安全和合规团队能够全面了解大规模的安全日志，同时降低安全信息和事件管理（SIEM）系统的成本。有趣的是，这种网络安全工作负载可以通过 Snowflake Data Exchange 上的网络安全合作伙伴来增强，他们可以提供威胁检测、威胁搜寻、异常检测等。我们将在第 12 章中深入探讨 Snowpark 和安全数据湖工作负载。我们还将在第 12 章中讨论最新的 Snowflake 工作负载 Unistore（用于事务和分析数据的工作负载）。

第一章将向您介绍 Snowflake，并让您在新的 Snowflake Web UI Snowsight 中轻松导航。此外，本章还包括有关 Snowflake 社区、认证和 Snowflake 活动的信息。书中还有一个部分描述了有关代码示例的注意事项。花时间适应本章将使您在浏览成功的章节时取得成功。

Snowflake Web 用户界面

有两种不同的 Snowflake Web 用户界面可用：经典控制台和 Snowsight，即新的 Snowflake Web UI。Snowsight 于 2021 年首次推出，现在是新创建的 Snowflake 账户的默认用户界面。Snowsight 预计将于 2023 年初成为所有账户的默认用户界面，此后不久将弃用经典控制台。除非另有说明，否则我们所有的动手示例都将在 Snowsight 中完成。

与所有其他 Snowflake 特性和功能一样，Snowsight 也在不断改进。因此，有时章节中的屏幕截图可能会与您正在使用的 Snowsight Web UI 中显示的屏幕截图略有不同。



为了支持快速创新，Snowflake 每周部署两个计划版本，每月部署一个行为更改版本。您可以在 Snowflake 文档中找到有关 Snowflake 版本的更多信息。

准备工作

在本书每章的准备部分，我们将创建该章的动手示例所需的任何文件夹、工作表和 Snowflake 对象。

您需要访问 Snowflake 实例，以便在阅读各章时跟随并完成动手示例。如果您还没有 Snowflake 实例的访问权限，则可以设置免费试用 Snowflake 帐户。如果你

需要有关如何创建免费试用 Snowflake 帐户的信息，请参阅附录 C。

如果您有权访问默认使用经典控制台的 Snowflake 组织，则可以通过以下两种方式之一访问 Snowsight。在经典控制台 Web 界面中，您可以单击屏幕右上角的 Snowsight 按钮（如图 1-1 所示）。或者，您可以直接登录到 Snowsight。



图 1-1. 显示 Snowsight 按钮的经典控制台 Web 界面

进入 Snowsight 后，Worksheets（工作表）是默认选项卡（如图 1-2 所示）。您还可以单击一些不同的选项卡（包括 Data 选项卡和 Compute 选项卡），以查看一些可用的菜单选项。正如我们稍后将看到的，Databases 子选项卡将显示您的访问权限中可用的数据库。

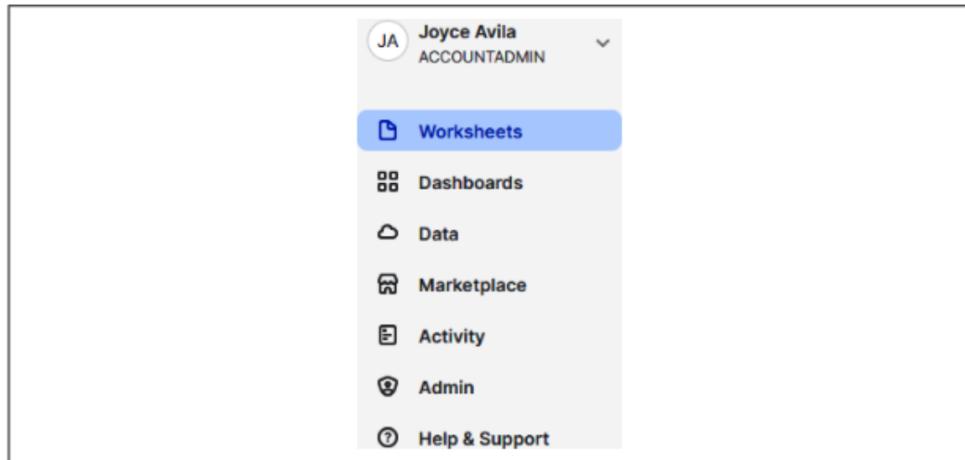


图 1-2. Snowsight UI 选项卡，其中 Worksheets（工作表）选项卡为默认选项卡

如果您之前一直在 Classic Console Web 界面中工作，或者这是您第一次登录，则当您首次进入 Snowsight 时，系统将为您提供导入工作表的选项（如图 1-3 所示）。

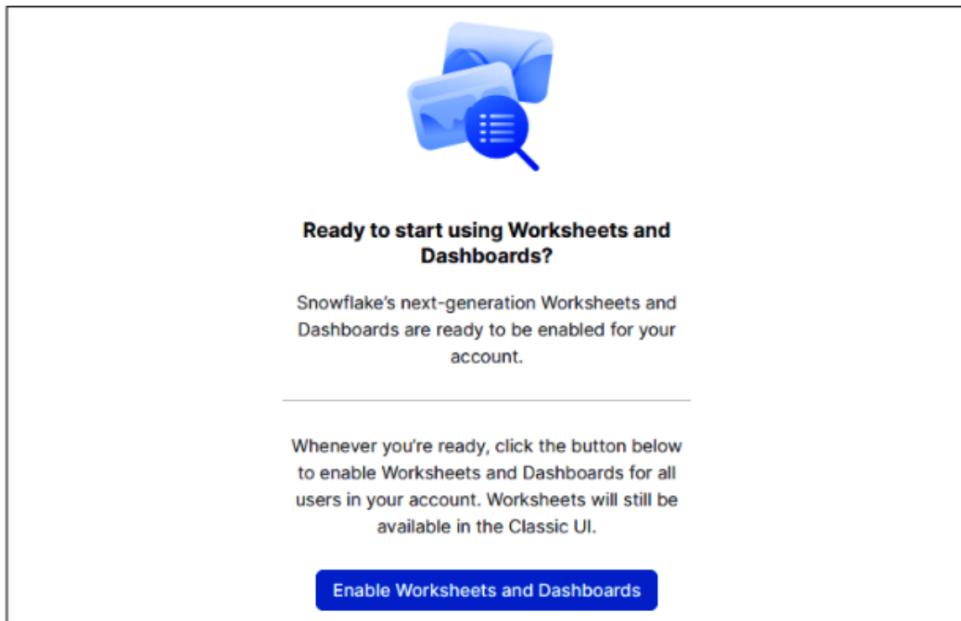


图 1-3。首次使用 Snowsight 时，将向您显示一个用于导入工作表的选项

如果从 Classic Console UI 导入工作表，将创建一个新的带时间戳的文件夹（如图 1-4 所示）。



图 1-4. 文件夹名称默认为导入工作表的日期和时间

您可以使用最新版本的 Google Chrome、Mozilla Firefox 或适用于 macOS 的 Apple Safari 访问 Snowsight。登录后，客户端会话将无限期地维护，并继续进行用户活动。处于非活动状态 4 小时后，当前会话将终止，您必须重新登录。可以更改 4 小时的默认会话超时策略；SES-Sion 策略的最小可配置空闲超时值为 5 分钟。

Snowsight 方向

登录 Snowsight 时，您将可以访问 User（用户）菜单和主菜单。在 User（用户）菜单中，您可以查看您的用户名和当前角色。默认情况下，当您首次注册试用帐户时，将为您分配 SYSADMIN 角色。

您还会注意到，Main（主菜单）默认为 Worksheets（工作表）选项。主菜单右侧的区域是主菜单选择更多选项的地方。在图 1-5 中，您可以看到工作表主菜单右侧的菜单。Worksheets（工作表）菜单下面是特定于 Main menu（主菜单）选择的子菜单。

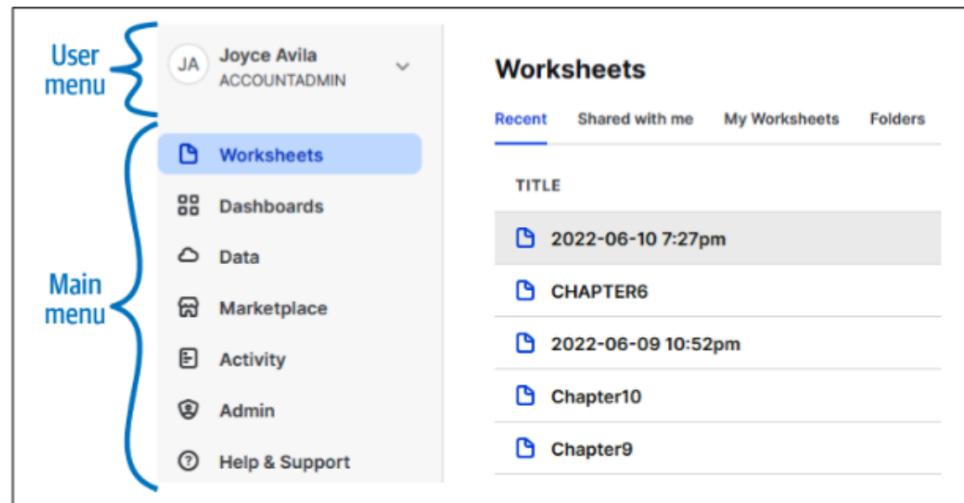


图 1-5. Snowsight 菜单选项

接下来，我们将设置 Snowsight 首选项，并花几分钟时间浏览 Snowsight 工作表区域。

Snowsight 首选项

要设置您的首选项，请直接单击您的姓名，或使用您姓名旁边的下拉箭头访问子菜单（如图 1-6 所示）。您会注意到，可以在此处切换角色。尝试将您的角色切换到其他角色，然后将其切换回 SYSADMIN。这在整个章节练习中都很重要，因为我们需要定期在角色之间切换。

您还会注意到 Snowflake Support（Snowflake 支持）可从 User（用户）菜单中访问。每当您需要提交案例时，只需单击 Support（支持）即可。请务必注意，使用免费试用账户时，您将无法创建支持案例。

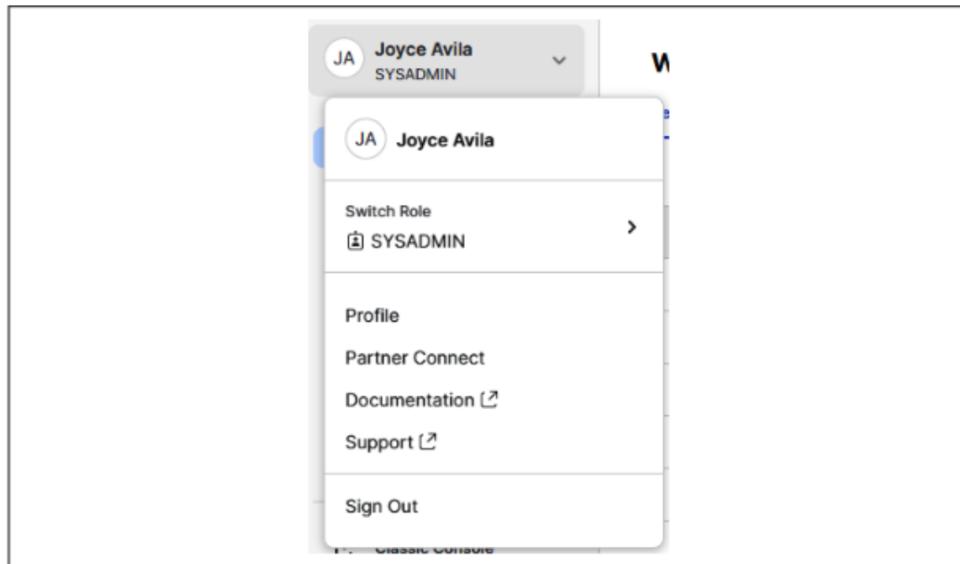


图 1-6. 用户菜单选择

点击 本人简介 选项，然后 本人简介 子菜单将可供您使用（如图 1-7 所示）。此外，您还可以在此处更改首选语言并注册 Multi-Factor Authentication（MFA）。如果您想更改密码，这也是您要去的地方。

如果您对配置文件进行了更改，请单击 Save 按钮。否则，请单击 Close 按钮。

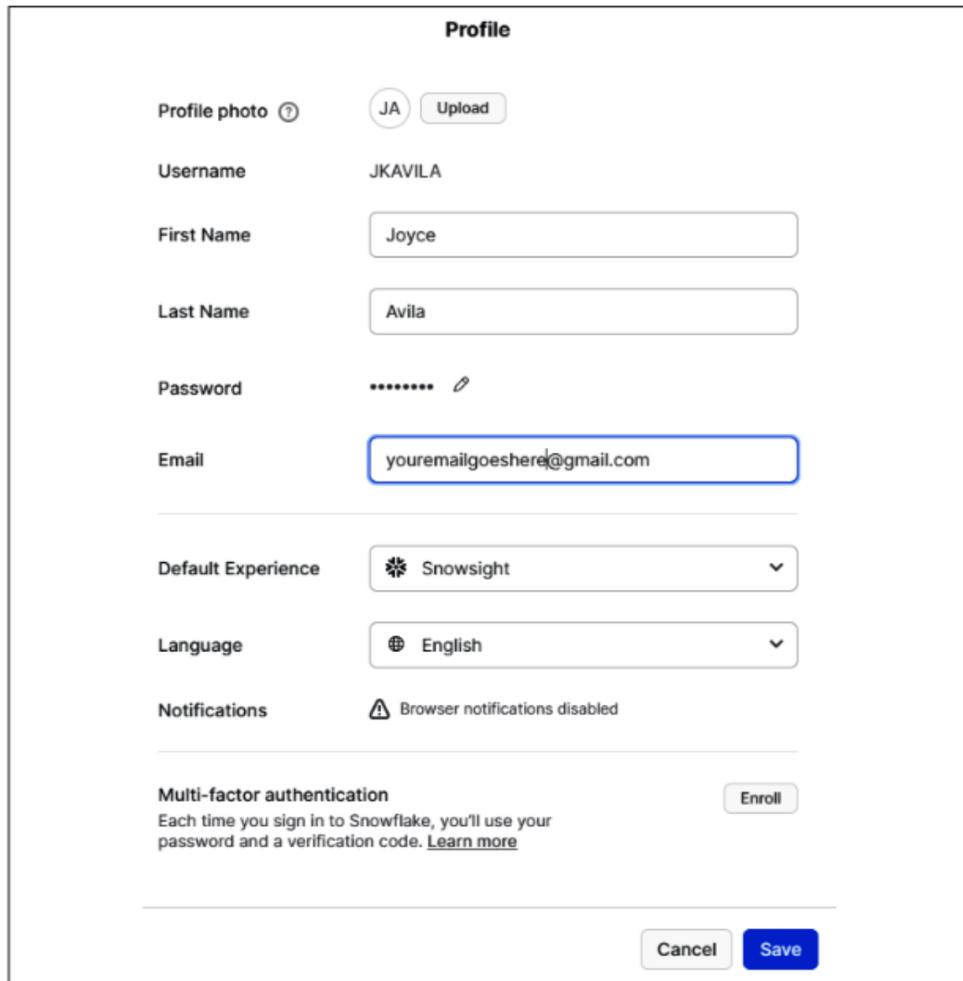


图 1-7. 配置文件 (Profile) 子菜单

导航 Snowsight 工作表

当您浏览 Snowsight 并执行书中的练习时，您会注意到 Snowsight 中启用了自动保存。按照这些思路，Snowsight 的一大特色是版本化历史记录；如有必要，您可以恢复到以前的工作表版本。

上下文设置

设置查询的上下文是我们将在整个示例中执行的操作。要设置上下文，我们需要选择我们将使用的虚拟仓库以及

我们要用于执行查询的角色。我们还需要选择要用于查询的数据库和 schema。
虚拟仓库、角色和其他 Snowflake 对象将在后续章节中介绍。现在，只需知道您当前的角色决定了您可以看到哪些数据以及您可以在 Snowflake 中执行哪些操作。您当前使用的虚拟仓库是选择用于运行查询的计算能力。大多数（但不是全部）查询都需要虚拟仓库来执行查询。如果您使用的是 Snowflake 试用组织，则您的默认角色为 SYSADMIN，默认虚拟仓库为 COMPUTE_WH。

在为即将到来的查询设置上下文之前，我们将了解我们当前使用的角色和虚拟仓库。为此，我们需要在工作表中执行一些 SQL 查询。让我们创建一个新文件夹和新工作表，我们可以在其中运行查询。在右上角，单击省略号，然后单击“新建文件夹”（如图 1-8 所示）。

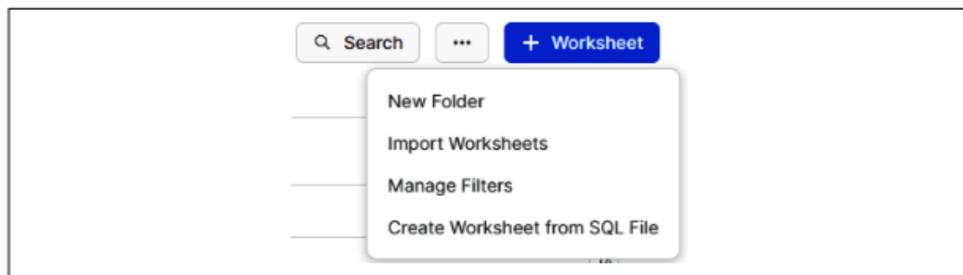


图 1-8. 单击省略号可为用户提供选项，包括创建新文件夹

将新文件夹命名为 Chapter1 并单击 Create Folder 按钮（如图 1-9 所示）。

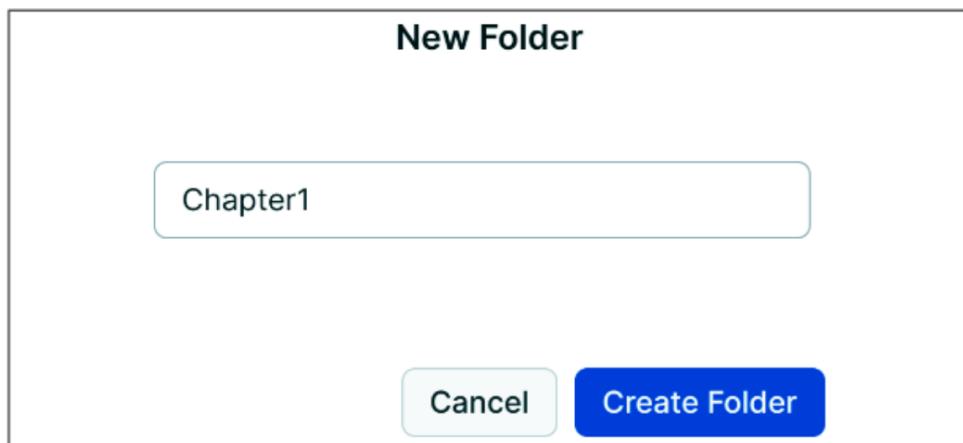


图 1-9. 用于创建新文件夹的菜单

点击 + 工作表 按钮创建一个新的工作表（如图 1-10 所示）。



图 1-10. 通过单击 + Worksheet 按钮在 Snowsight 中创建新工作表

单击 + Worksheet 按钮后，将在 Chapter1 文件夹中创建一个新工作表。您会注意到，工作表的名称默认为当前日期和时间（如图 1-11 所示），并且光标位于工作表中等待 SQL 语句。需要注意的另一件重要事情是，尚未选择任何数据库。

我们稍后会开始为数据库设置上下文。



图 1-11。在 Chapter1 文件夹中创建的新工作表

不过，在执行任何查询之前，让我们更改工作表的名称。单击工作表名称旁边的下拉箭头，然后输入新的工作表名称 Chapter1 Getting Started（如图 1-12 所示）。

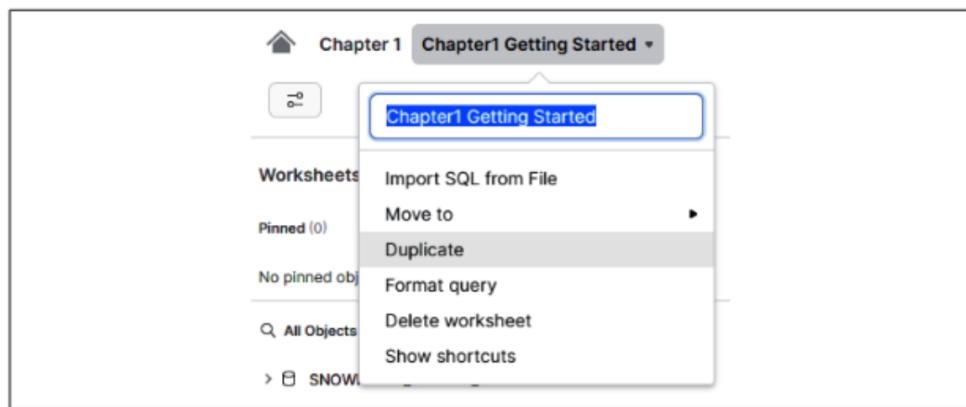
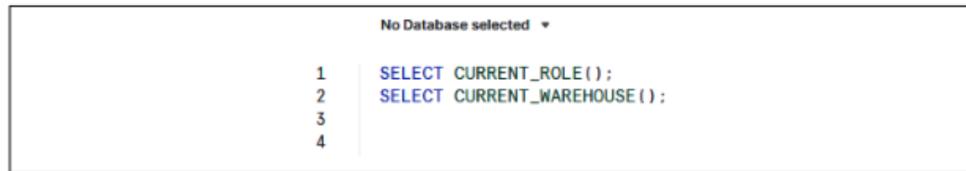


图 1-12. 重命名工作表

现在，您可以在 Snowsight 工作表中输入您的第一个查询。输入以下语句：

```
选择 CURRENT_ROLE();  
选择 CURRENT_WAREHOUSE();
```

您的工作表应如图 1-13 所示，其中两个语句位于两个单独的行上，光标位于第 3 行的开头。这些语句只返回信息；它们不会更改您当前的角色或当前的虚拟仓库。



```
No Database selected ▾  
1 | SELECT CURRENT_ROLE();  
2 | SELECT CURRENT_WAREHOUSE();  
3 |  
4 |
```

图 1-13。光标位于工作表的第 3 行

屏幕右上角是一个带有蓝色箭头的按钮（如图 1-14 所示）。此按钮（从现在开始，我们将称为 Run 按钮），您将单击该按钮以在工作表中运行查询。



图 1-14。蓝色箭头按钮，即 Snowflake 中的 Run 按钮

确保您的光标位于第 3 行的开头，然后单击 Run 按钮。您应该可以看到当前的虚拟仓库，如图 1-15 所示。



	CURRENT_WAREHOUSE()	...
1	COMPUTE_WH	

图 1-15. 从工作表查询结果

现在让我们看看如果我们将光标放在工作表的开头，即第 1 行，然后单击 Run 按钮会发生什么。您将看到如图 1-16 所示的结果。

CURRENT_ROLE()	
1	SYSADMIN

图 1-16. 显示当前角色的查询结果

您将注意到，仅执行了第一个查询。现在，让我们尝试另一种方法来执行这些查询。如图 1-17 所示，突出显示这两个语句，这次按 Ctrl + Enter 而不是 Run 按钮。Ctrl + Enter 是 Run（运行）按钮的快捷方式。您会注意到这两个语句都已执行。

```

No Database selected ▾

1 | SELECT CURRENT_ROLE();
2 | SELECT CURRENT_WAREHOUSE();
3 |
4 |

```

图 1-17。在执行代码之前，所有 Snowflake 语句都会高亮显示

接下来，让我们看看我们当前的数据库是什么。执行以下代码：

```
选择 CURRENT_DATABASE();
```

当我们运行上述语句以选择当前数据库时，将返回 null 值。这是因为我们还没有为数据库设置上下文。我们可以通过以下两种方式之一为数据库设置上下文：使用下拉菜单或使用命令执行查询。如果您不确定可以使用哪些数据库，可以使用下拉菜单。下拉菜单将为您列出您的角色可以访问的所有数据库。这次我们将使用查询。当我们开始输入语句时，Snowflake 使用其智能自动完成功能来帮助我们（如图 1-18 所示）。

```

No Database selected ▾

1 | USE DATABASE SNOW;
2 |
3 |
4 |
5 |

```

图 1-18。智能自动完成的实际应用

继续执行该语句以使用 Snowflake 示例数据库：

```
使用数据库SNOWFLAKE_SAMPLE_DATA;
```

执行该查询时，数据库上下文将发生更改。现在，您可以直接在工作表上看到更改的上下文，单击下拉菜单时，下拉菜单会显示当前数据库（如图 1-19 所示）。它还显示该数据库中可访问的架构。没有任何架构会突出显示或旁边有复选标记，这意味着尚未选择任何架构。

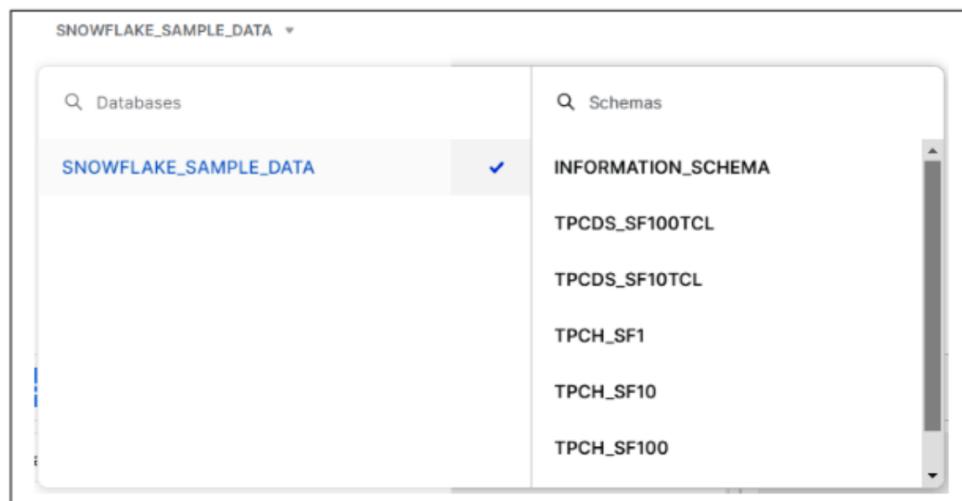


图 1-19. 当前选择的数据库为 SNOWFLAKE_SAMPLE_DATA

这一次，我们将从菜单中进行选择，而不是执行命令来设置当前架构。继续并选择 schema TPCDS_SF100TCL（如图 1-20 所示）。

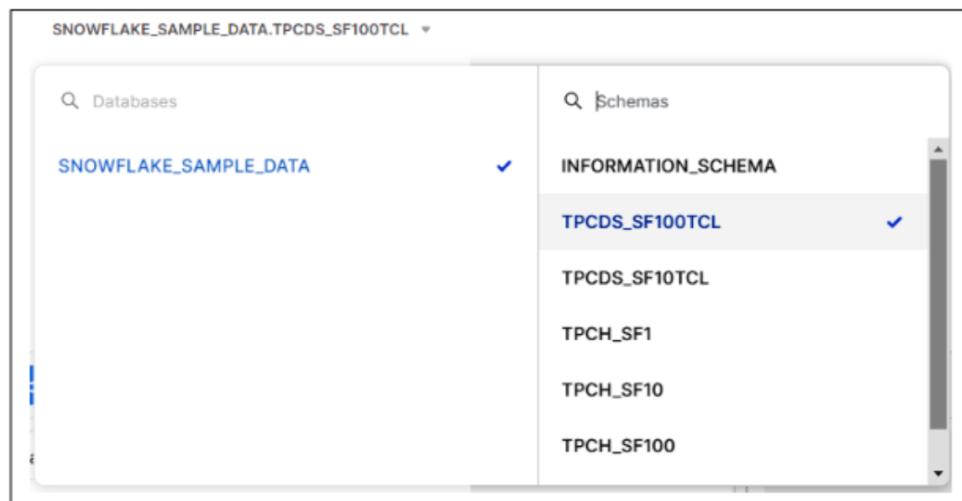


图 1-20。数据库和架构选择

您会注意到的关于 Snowsight 的一点是，查询下方有四个按钮，就在对象列表的右侧。目前，图 1-21 中突出显示了其中三个按钮。不能同时选择所有四个按钮，因为只有在取消选择 Chart 时才能选择 Results。



图 1-21. Snowsight 中的四个按钮，位于查询的正下方

选择 Objects（对象）按钮。现在取消选择，然后重新选择其他三个 buttons 中的每一个，一次一个，看看您的工作区会发生什么。Objects（对象）按钮确定您是否看到屏幕左侧的对象。您可以取消选择 Query（查询）按钮以获得结果的全屏视图，也可以取消选择 Results（结果）按钮以获得工作表查询的全屏视图。如果选择 Chart button，则取消选择 Results 按钮。通过更改 Snowsight 查看区域，您可以自定义 Snowflake UI。

在下一节中，我们将介绍提高用户工作效率的 Snowsight 新功能。在第 11 章中，我们将介绍 Snowsight 图表和控制面板以及一些新的 Snowsight 协作功能。

提高生产力

如果您一直在 Snowflake Classic 控制台 Web 界面中工作，您将会喜欢 Snowsight 带来的许多改进。Snowsight 提供自动 Matic 上下文建议、脚本版本历史记录、工作表文件夹结构、快捷方式等。让我们深入研究一下其中的一些功能。

使用智能自动完成中的上下文建议

当我们编写查询以设置数据库的上下文时，Snowflake 智能自动完成功能为我们提供了上下文建议。事实上，Snowflake 在我们完成数据库名称之前向我们展示了我们的选择（如图 1-22 所示）。如果您要开始输入函数名称，Snowflake 将为您提供有关所需参数的信息以及指向 Snowflake 文档的链接。

The screenshot shows a SQL query editor window titled "SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL". The code being typed is "SELECT COUNT|". A tooltip appears over the cursor, providing information about the COUNT function:

count([DISTINCT] <expr1> [, <expr2> ...])
function Returns either the number of non-NULL records for
the specified columns, or the total number of
records Docs

Below the tooltip, another part of the documentation is visible:

count_if(<condition>)
function Returns the number of records that satisfy a cond...

图 1-22. 智能自动完成提供的 Snowsight 上下文建议

格式化 SQL

Snowsight 使您能够整齐地格式化 SQL 代码。使用下拉菜单选择 “Format query” 以重新格式化您的 SQL 查询，如图 1-23 所示。

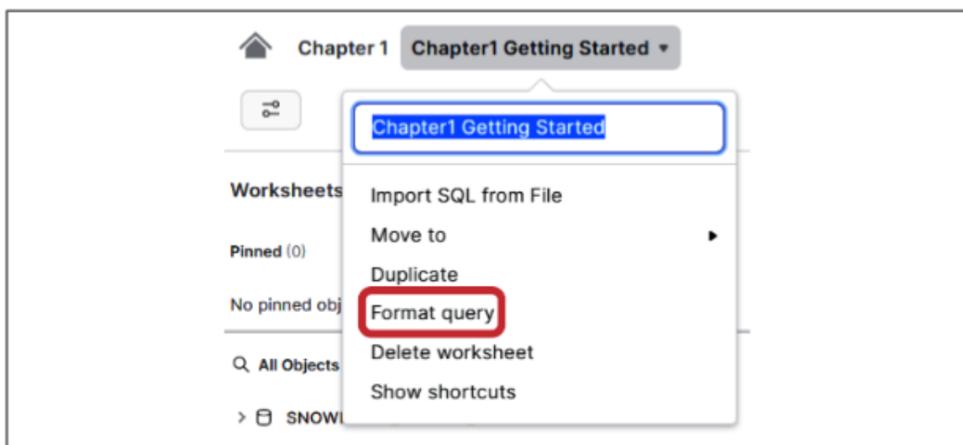


图 1-23. 用于设置查询格式的 Snowsight 选项

在与他人共享或协作处理仪表板之前，不再需要花时间格式化凌乱的代码。“Format query” 选项可以为您处理这个问题。在共享 SQL 代码或与他人协作进行可视化之前，您可以使用 “Format query” 选项来清理任何杂乱的格式。

使用快捷方式

您会发现 Snowsight 命令中有许多快捷命令。单击工作表名称旁边的下拉箭头，然后单击“显示快捷方式”以访问这些命令（如图 1-24 所示）。

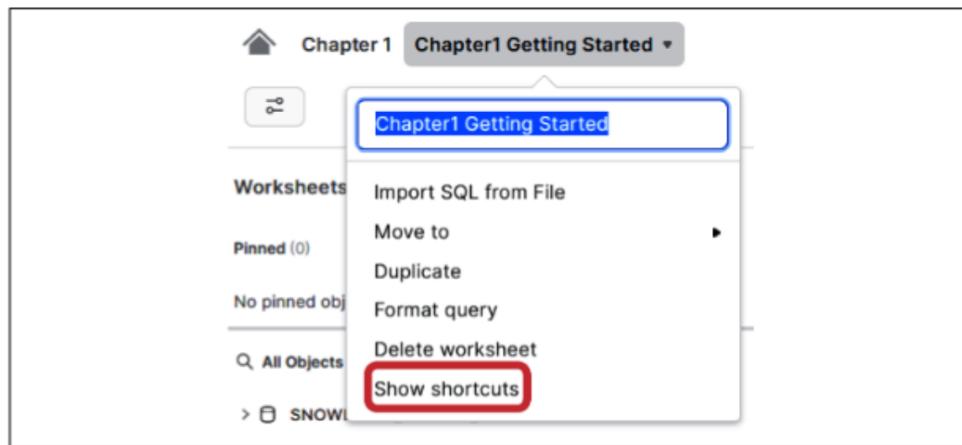


图 1-24. Snowsight 中提供了显示快捷方式的选项

访问版本历史记录

每次在 Snowsight 工作表中运行 SQL 语句时，都会自动保存当前工作表。如果多次执行语句或脚本，则版本历史记录将可用。可以通过访问屏幕右侧的下拉菜单来访问各种版本，该菜单显示上次更新工作表的时间（如图 1-25 所示）。

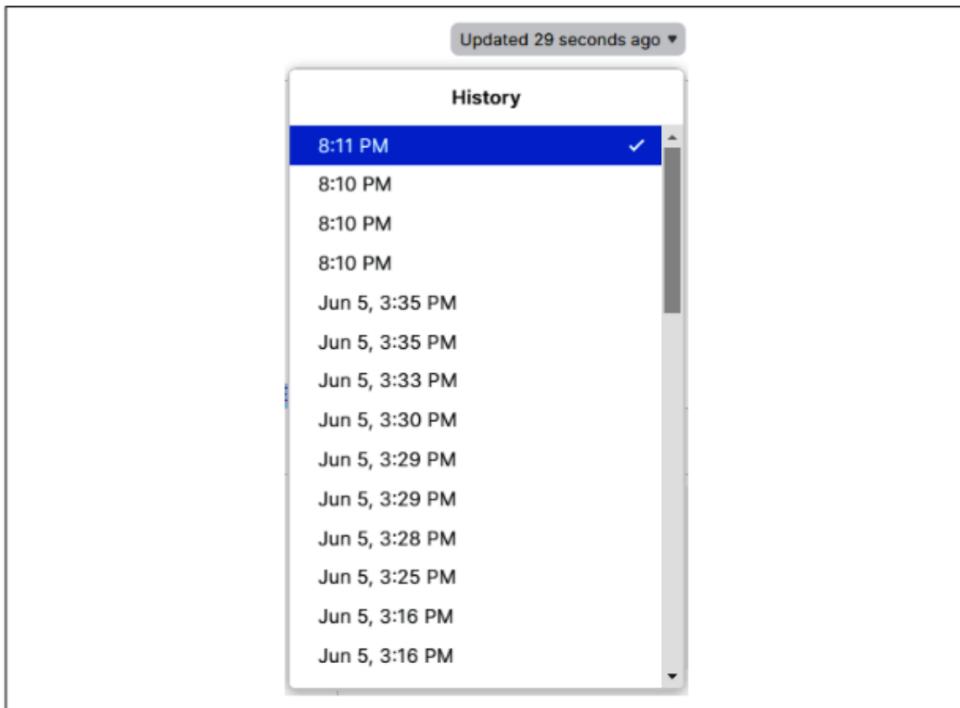


图 1-25. 显示最近更新的查询历史记录

在接下来的章节中，当我们浏览 Snowsight 时，我将指出其他 Snowsight 功能。准备好数字返回主菜单后，只需单击 房子 屏幕左上角的图标（如图 1-26 所示）。

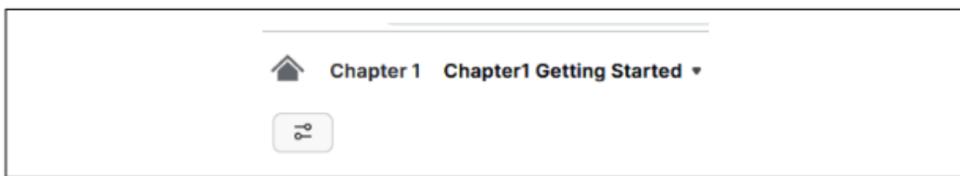


图 1-26。House 图标，将您返回到主菜单

Snowflake 社区

本书将教您 Snowflake 架构以及如何使用 Snowflake。书中的信息和教训是全面的；尽管如此，您可能会发现您对特定主题有疑问。因此，我鼓励您加入并参与 Snowflake 社区。

如果您想与对 Snowflake 有类似热情的其他人建立联系，您可能有兴趣加入一些用户组。部分用户组持有

在北美、欧洲、中东和亚太地区举办现场活动。您可以在 Snowflake 用户组的网站上找到有关他们的更多信息。除了区域用户组之外，还有虚拟特殊兴趣组（参见图 1-27 的列表）。我帮助管理 Data Collaboration 特别内部小组。



图 1-27. Snowflake 特别兴趣小组（虚拟）

Snowflake 用户组只是您参与 Snowflake 社区的方式之一。此外，还有社区组、资源链接以及 Snowflake 社区登录页面提供的讨论访问权限。要访问这些组和资源，请单击页面右上角的 Log In（登录），然后单击“Not a member？（不是成员？）”链接以创建您的免费 Snowflake 社区成员帐户。

Snowflake 为在社区中非常活跃的 Snowflake 专家提供了一个特殊的 Data Superhero 计划。每个被认定为数据超级英雄的人都会收到一个由 Snowflake 公司活动总监 Francis 毛 创建的自定义数据超级英雄角色。图 1-28 显示了一些 Snowflake Data 超级英雄的聚会，包括我！



图 1-28. Snowflake Data 超级英雄

Snowflake 认证

随着你阅读本书中的章节，你会发现你的 Snowflake 知识会迅速增长。在某些时候，您可能需要考虑获得 Snowflake 认证，以向社区展示这些知识。

所有 Snowflake 认证路径都从 SnowPro Core 认证考试开始。通过 SnowPro 考试后，您可以参加五项基于角色的高级认证中的任意或全部：管理员、架构师、数据分析师、数据工程师和数据科学家。Snowflake 证书的有效期为两年，之后必须通过认证考试。通过任何高级 Snowflake 认证都会重置 SnowPro Core 认证的时钟，因此您还有两年时间需要重新认证。有关 Snowflake 认证的更多信息，请访问他们的网站。

Snowday 和 Snowflake 峰会活动

Snowflake 全年举办许多活动，包括 Data for Breakfast 活动。新产品发布信息的两个 Snowflake 主要活动是 Snowday 和 Snowflake Summit。Snowday 是每年 11 月举行的虚拟活动。Snowflake Summit 是每年 6 月举行的现场活动。Snowflake Summit 设有主题演讲、分组会议、动手实验、展览楼层等。

书中有关代码示例的重要注意事项

为本书中的每一章创建的代码示例都经过设计，以便它们可以独立于其他章节完成。这使您能够重新访问任何特定章节的示例，并在每章的末尾执行代码清理。

偶尔，我会指出免费试用帐户中事物的外观或功能，以及它们在付费帐户中的外观或功能。我们会看到很少的差异，但重要的是要认识到两者在某些方面有所不同。

各章中演示的功能假定您拥有 Enterprise Edition Snowflake 实例或更高版本。如果您当前在 Snowflake Standard Edition 组织中工作，则需要设置免费试用帐户并选择 Enterprise Edition。否则，如果您使用 Standard Edition Snowflake 组织，将有一些示例将无法完成。有关不同 Snowflake 版本的更多信息，请参阅附录 C。

在本章的前面部分，您了解了如何使用 Snowsight 格式化 SQL 语句。本书中为您提供的陈述将尽可能遵循该格式，但由于空间限制，格式有时可能会略有不同。此外，在某些章节中，我们将使用随 Snowflake 帐户提供的 Snowflake 示例数据库。由于该 Snowflake 示例数据集中的基础数据有时会发生变化，因此示例章节中包含的结果集可能与您的结果略有不同。<https://github.com/SnowflakeDefinitiveGuide> 将为 Snowflake 示例数据库中稍后变得不可用的表或架构提供替代练习。

在本章中，我们执行了命令来设置数据库的上下文。在接下来的章节示例中，我们将包含用于设置 context 的命令。您可以选择执行命令或从下拉菜单中选择上下文。

对于整本书的编码示例，每当我们创建自己的 Snowflake 对象时，我们都将遵循附录 B 中的命名标准最佳实践。遵循最佳实践有助于使代码更简单、更具可读性且更易于维护。它还支持创建可重用的代码，并使其更容易检测错误。在本节中，我将介绍一些最重要的最佳实践。

对于对象名称，我们将使用全部大写字母。我们可以通过使用所有小写字母或混合大小写来获得相同的结果，因为 Snowflake 会将对象名称转换为大写。例如，如果要使用大小写混合或全部小写字母的对象名称，则需要将对象名称括在引号中。当您使用连接到 Snowflake 的第三方工具时，牢记此功能也很重要。某些第三方工具不接受空格或特殊字符，因此在命名 Snowflake 对象时最好避免这两种情况。

另一个重要的最佳实践是避免使用不同的对象使用相同的名称。在后面的章节中，您将了解不同的 table 类型，包括临时 table 和永久 table。您将发现临时表是基于 session 的；因此，它们不受唯一性要求的约束。即使您可以创建与另一个表同名的临时表，这样做也不是一个好的做法。这将在第 3 章中更详细地讨论。

Snowflake 通过角色控制访问，特定角色负责创建某些类型的对象。例如，SYSADMIN 角色用于创建数据库和虚拟仓库。我们将在第 5 章中了解有关角色的更多信息，并且只要有可能，我们将遵循示例中的最佳实践。有时，我们需要偏离最佳实践以保持示例简单。发生这种情况时，我要提到我们使用的是与推荐角色不同的角色。

创建新对象时，可以单独使用 statement，也可以选择将 the 或语法添加到状态中。对于我们的示例，我们将使用语法，以便您始终可以返回到本章练习的开头重新开始，而不必先删除对象。但是，在实践中，请务必使用 CREATE IF NOT EXISTS 语法，尤其是在生产环境中。如果您在生产中错误地使用了 REPLACE 语法，则可以选择使用 Snowflake Time Travel 功能将对象返回到其原始状态。时间旅行在第 7 章中演示。

书中的代码示例已经过全面测试；然而，总是有可能出现错误。如果您在代码或文本中发现错误，请将您的勘误表提交给 O'Reilly，地址为

<https://oreil.ly/snowflake-the-definitive-guide>。

代码清理

在本章中，我们没有创建任何新的 Snowflake 对象，因此不需要进行清理。可以保留我们创建的文件夹和工作表，因为它们不会占用任何存储空间或产生任何计算费用。

总结

Snowflake 成为历史上最大的软件 IPO 是有原因的，也是 Snowflake 不断创新并推动行业颠覆的原因。Snowflake Data Cloud 安全、可扩展且易于管理；它能够消除数据孤岛并从单一平台运行工作负载，这为数据分析的大众化创造了机会。当您阅读本书的每一章时，您将亲眼目睹为什么 Snowflake 是新数据经济的领导者。

在本介绍性章节中，您已经能够熟悉 Snowsight，即 Snowflake Web 用户界面。您还有机会了解 Snowflake 社区活动以及 Snowflake 认证。重要的是，您了解了本书中代码示例的注意事项。最值得注意的是，除了少数例外，我们将遵循附录 B 中概述的 Snowflake 最佳实践。

下一章将把传统的数据平台架构与 Snowflake 进行比较。然后，我们将深入研究 Snowflake 的三个不同图层。这是重要的基础章节，可帮助您更好地了解 Snowflake 的一些技术性方面。您还将获得创建称为虚拟仓库的新计算资源集群的实践经验。

知识检查

以下问题基于本章提供的信息：

1. Snowflake 免费试用帐户允许您使用付费帐户的几乎所有功能。但是，使用 Snow - flake 试用帐户时有哪些区别？
2. 如果您无法在 Snowflake 中创建支持案例，您有什么选择？

3. 解释为 Snowflake 工作表设置上下文意味着什么。
4. “Format query” 有什么作用？具体来说，“Format query” 是否更正了命令或表名的拼写？
5. Snowflake 认证的有效期是多久？您可以通过哪两种方式延长 Snowflake 认证的日期？
6. 在 Snowflake 工作表中执行 SQL 查询的两种不同方式是什么？

7. 在 Snowflake 工作表中，如何返回主菜单？
8. 在命名 Snow - flake 对象时，我们使用全部大写字母的原因是什么？我们可以改用混合大小写吗？如果是这样，请说明我们将如何实现这一目标。

9. ~~请勿编辑新的~~ 创建或替换对象。为什么最好不要在生产环境中执行此操作？
10. 当您创建新的 Snowflake 工作表时，工作表的默认名称是什么？

这些问题的答案可在附录 A 中找到。

创建和管理 Snowflake 架构

十年前，数据平台架构缺乏必要的可扩展性，无法使数据驱动型团队更容易同时共享相同的数据，无论团队规模或与数据的接近程度如何。对可扩展性的需求不断增长，对数据进行受监管访问以生成可操作见解的需求也随之增加。为了满足这一需求，对现有的数据平台架构进行了修改。然而，考虑到平台的数量和复杂性以及应用程序的数据密集型性质，这并没有解决问题，直到 Snowflake 以独特的架构登上舞台。

Snowflake 是一个进化的现代数据平台，它解决了可扩展性问题。与传统的云数据平台架构相比，Snowflake 的数据存储和处理速度明显更快、更易于使用且更实惠。Snowflake 的数据云将新的 SQL 查询引擎与专为云从头开始设计和构建的创新架构相结合，为用户提供独特的体验。

准备工作

创建标题为 Chapter2 Creating and Managing Snowflake Architecture 的新工作表。如果您在创建新工作表时需要帮助，请参阅第 8 页上的“导航 Snowsight 工作表”。要设置工作表上下文，请确保您使用的是 SYSADMIN 角色和 COMPUTE_WH 虚拟仓库。

传统数据平台架构

在本节中，我们将简要回顾一些传统的数据平台架构，以及它们是如何设计以提高可扩展性的。可伸缩性是系统处理越来越多的工作的能力。我们还将讨论这些架构的局限性，并发现 Snowflake Data Cloud 架构如此独特的原因。之后，我们将详细了解三个不同的 Snowflake 架构层中的每一个：云服务层、查询处理（虚拟仓库）计算层和集中式（混合列式）数据库存储层。

共享磁盘（可扩展）架构

共享磁盘体系结构是一种早期的扩展方法，旨在将数据存储在中央存储位置，并可从多个数据库群集节点访问（如图 2-1 所示）。每个集群节点访问的数据始终可用，因为所有数据修改都已写入共享磁盘。此架构是一种传统的数据库设计，以其数据管理的简单性而闻名。虽然这种方法在理论上很简单，但它需要复杂的磁盘上锁定机制来确保数据一致性，这反过来又会导致瓶颈。数据交互性（允许许多用户影响数据库中的多个事务）也是一个主要问题，添加更多计算节点只会使共享磁盘体系结构中的问题复杂化。因此，此体系结构的真正可伸缩性是有限的。

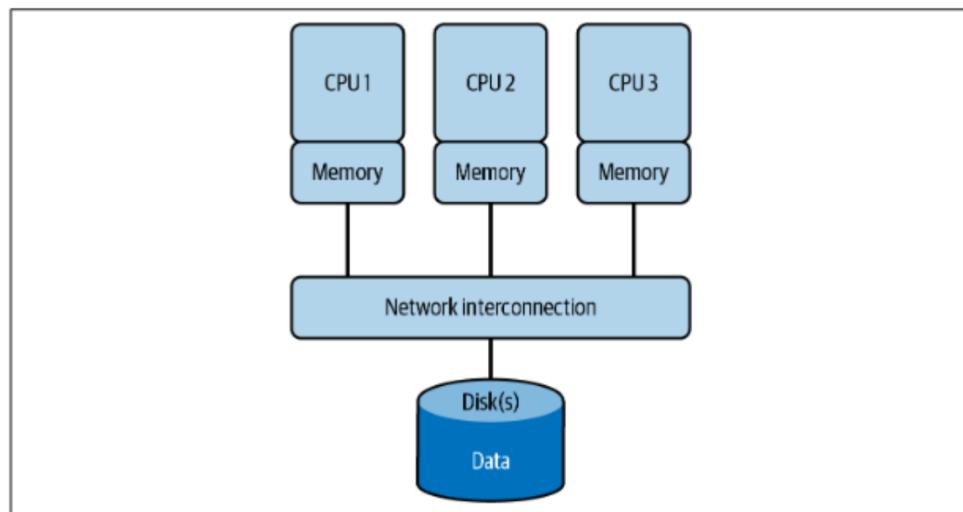


图 2-1. 共享磁盘体系结构，受磁盘性能的限制

Oracle RAC 是共享磁盘体系结构的一个示例。

无共享（可扩展）架构

无共享架构，其中存储和计算一起扩展（如图 2-2 所示），是为了应对共享磁盘架构造成的瓶颈而设计的。架构的这种演变之所以成为可能，是因为存储变得相对便宜。但是，分布式集群节点以及相关的磁盘存储、CPU 和内存需要在节点之间随机排列数据，这会增加开销。数据在节点之间的分配方式将决定额外开销的程度。在存储和计算之间取得适当的平衡尤其困难。即使可以调整集群大小，也需要花费时间。因此，组织通常会过度预置 sharednothing 资源，从而导致未使用、不需要的资源。无共享架构的示例包括 IBM DB2、Vertica 和 Pivotal Greenplum。

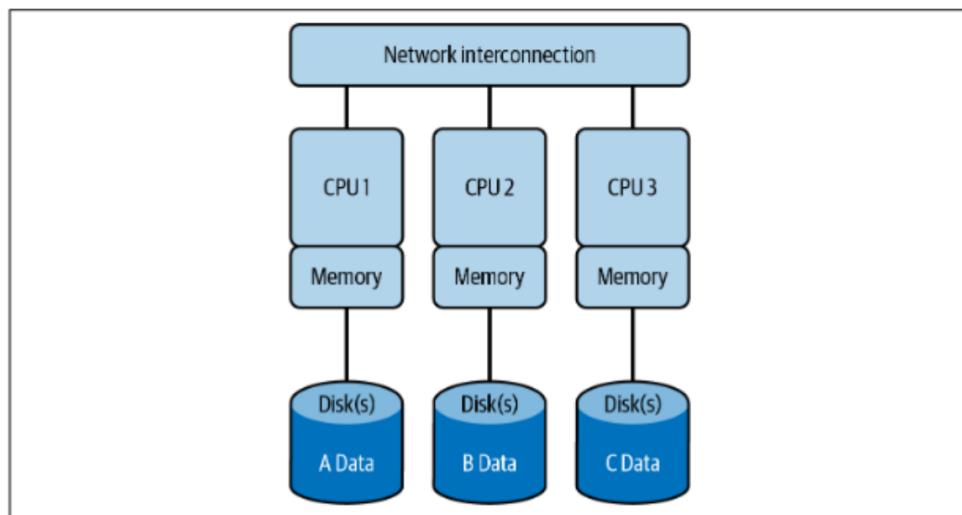


图 2-2. 无共享架构，由于需要跨节点分发和查询数据而受到限制

NoSQL 替代方案

大多数 NoSQL 解决方案都依赖于无共享架构；因此，它们具有许多相同的限制。但是，NoSQL 解决方案的好处是它们可以存储非关系数据，而无需首先转换数据。此外，大多数 NoSQL 系统不需要架构。NoSQL 是一个术语，意思是“不仅是 SQL”而不是“对 SQL 说不”，是存储电子邮件、Web 链接、社交媒体帖子和推文、路线图和空间数据的不错选择。

NoSQL 数据库有四种类型：文档存储、键值（KV）存储、列系列数据存储或宽列数据存储以及图形数据库。

基于文档的 NoSQL 数据库（如 MongoDB）将数据存储在 JSON 对象中，其中每个文档都具有类似键值对的结构。DynamoDB 等键值数据库对于捕获特定环境中的客户行为特别有用。Cassandra 是基于列的数据库的一个示例，其中大量动态列在逻辑上分组到列系列中。基于图形的数据库（如 Neo4j 和 Amazon Neptune）非常适合推荐引擎和社交网络，它们能够帮助查找数据点之间的模式或关系。

NoSQL 存储的一个主要限制是，在执行涉及许多记录的计算时，例如聚合、窗口函数和任意排序，它们的性能很差。因此，当您需要快速创建、读取、更新和删除（CRUD）表中的单个条目时，NoSQL 存储可能非常有用，但不建议将其用于临时分析。此外，NoSQL 替代解决方案需要专业技能，并且它们与大多数基于 SQL 的工具不兼容。

请注意，NoSQL 解决方案不是数据库仓库的替代品。虽然 NoSQL 替代方案对数据科学家很有用，但它们在分析方面表现不佳。

Snowflake 架构

即使是改进的传统数据平台，尤其是那些在本地实施的数据平台，也无法充分解决现代数据问题或解决长期存在的可扩展性问题。Snowflake 团队决定采用一种独特的方法。他们没有试图逐步改进或改造现有的软件架构，而是构建了一个全新的现代数据平台，专门用于云，允许多个用户同时共享实时数据。

独特的 Snowflake 设计在物理上分离，但在逻辑上集成了存储和计算，同时提供安全和管理等服务。当我们在接下来的章节中探索许多独特的 Snowflake 功能时，您将能够亲眼看到为什么 Snowflake 架构是唯一可以启用数据云的架构。

Snowflake 混合模型架构由三层组成，如图 2-3 所示：云服务层、计算层和数据存储层。以下各节将更详细地讨论每个层以及三个 Snowflake 缓存。

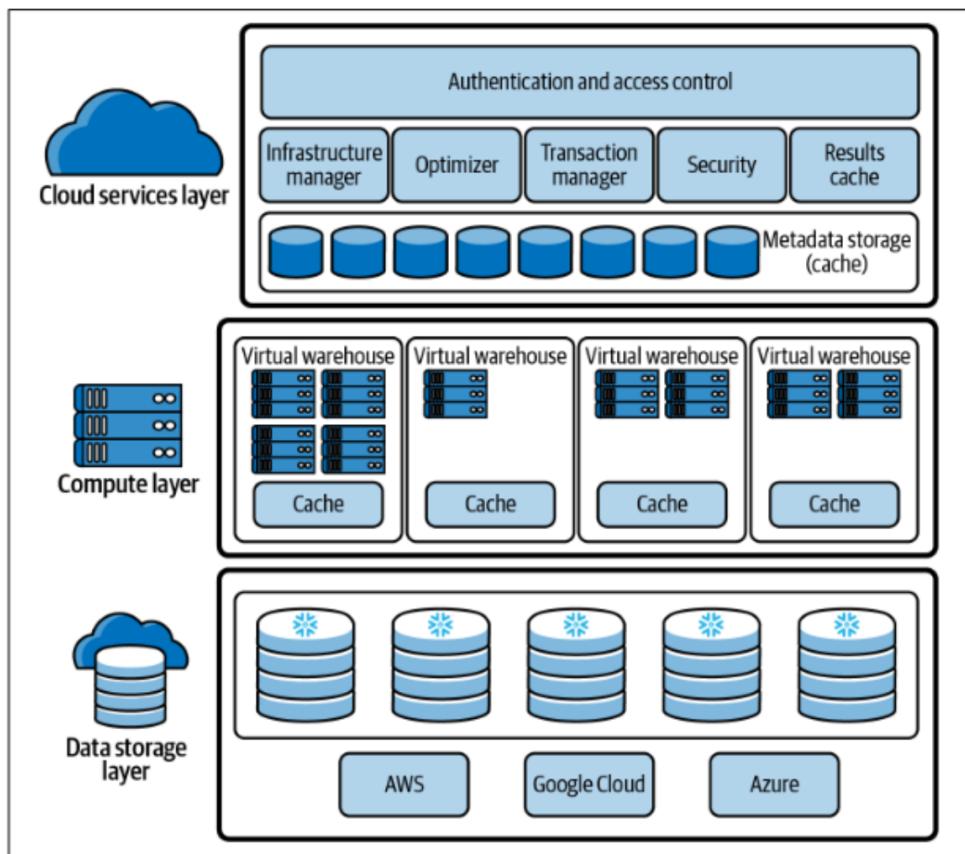


图 2-3. Snowflake 的混合柱式架构

Snowflake 的处理引擎是原生 SQL，正如我们将在后面的章节中看到的那样，Snowflake 还能够处理半结构化和非结构化数据。

云服务层

与 Snowflake 实例中数据的所有交互都从云服务层开始，也称为全局服务层（如图 2-4 所示）。Snowflake 云服务层是协调身份验证、访问控制和加密等活动的服务的集合。它还包括用于基础设施和元数据的管理功能，以及执行查询解析和优化等功能。云服务层有时被称为 Snowflake 大脑，因为所有不同的服务层组件协同工作，以处理从用户请求登录时开始的用户请求。

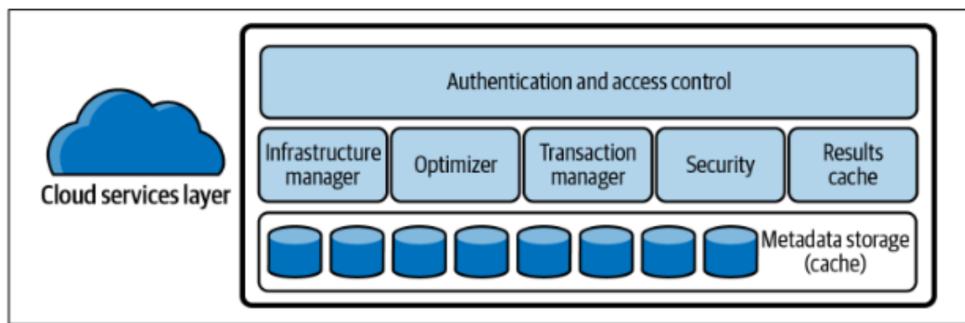


图 2-4. Snowflake 的云服务层

每次用户请求登录时，该请求都由云服务层处理。当用户提交 Snowflake 查询时，SQL 查询将被发送到云服务层优化器，然后再发送到计算层进行处理。云服务层使 SQL 客户端接口能够对数据执行数据定义语言（DDL）和数据操作语言（DML）。

管理云服务层

云服务层管理数据安全，包括数据共享的安全性。Snowflake 云服务层跨每个云提供商区域中的多个可用区运行，并保存结果缓存，即已执行查询结果的缓存副本。查询优化或数据筛选所需的元数据也存储在云服务层中。



与其他 Snowflake 层一样，云服务层将独立于其他层进行扩展。云服务层的扩展是一个自动化过程，不需要 Snowflake 最终用户直接操作。

云服务层的计费

大多数云服务消耗已包含在 Snowflake 定价中。但是，当客户偶尔超过其每日计算积分使用量的 10% 时，他们需要为超额付费。请注意，每日计算积分使用量以 UTC 时区计算。

所有查询都使用少量的云服务资源。DDL 操作是元数据操作，因此，它们仅使用云服务。牢记这两个事实，我们应该评估一些我们知道云服务成本会更高的情况，以考虑增加的成本是否值得这些好处。

使用多个简单查询时，尤其是访问会话信息或使用会话变量的查询时，云服务层的使用量可能会增加。当使用具有许多联接的大型复杂查询时，使用量也会增加。单行插入，而不是批量或批量加载，也会导致更高的云服务消耗。最后，如果您在 INFORMATION_SCHEMA 表上使用命令或某些仅限元数据的命令（例如命令），则只会消耗云服务资源。如果您的云服务成本高于预期，则可能需要调查这些情况。此外，请务必仔细检查任何合作伙伴工具，包括使用 JDBC 驱动程序的工具，因为这些第三方工具可能有改进的机会。

尽管特定使用案例的云服务成本很高，但有时从经济和/或战略角度来看，承担成本是有意义的。例如，利用查询的结果缓存（尤其是大型或复杂查询）意味着该查询的计算成本为零。对 DDL 命令的正确频率和粒度进行深思熟虑的选择，特别是对于克隆等使用案例，有助于更好地平衡云服务消耗成本和虚拟仓库成本，从而降低总体成本。

查询处理 (Virtual Warehouse) 计算层

Snowflake 计算集群通常简称为虚拟仓库，是由 CPU 内存和临时存储组成的动态计算资源集群。在 Snowflake 中创建虚拟仓库利用计算集群，即云中在后台配置的虚拟机。Snowflake 不会发布在任何给定时间使用的确切服务器；随着云提供商修改其服务，它可能会发生变化。Snowflake 计算资源按需创建并部署到 Snowflake 用户，该过程对用户是透明的。

大多数 SQL 查询和所有 DML 操作都需要一个正在运行的虚拟仓库，包括将数据加载和卸载到表中，以及更新表中的行。一些 SQL 查询可以在不需要虚拟仓库的情况下执行，我们很快就会在本章后面讨论查询结果缓存时看到这方面的例子。

Snowflake 的独特架构允许存储和计算分离，这意味着任何虚拟仓库都可以访问与另一个虚拟仓库相同的数据，而不会对其他仓库的性能产生任何影响或影响。这是因为每个 Snowflake 虚拟仓库都独立运行，不与其他虚拟仓库共享计算资源（如图 2-5 所示）。

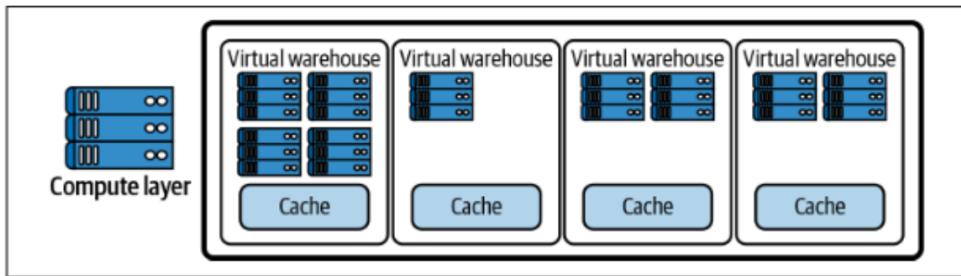


图 2-5. Snowflake 的计算（虚拟仓库）层

虚拟仓库在会话中运行时始终消耗积分。但是，Snowflake 虚拟仓库可以随时启动和停止，并且可以随时调整大小，即使在运行时也是如此。Snowflake 支持两种不同的仓库扩展方式。虚拟仓库可以通过调整仓库的大小来扩展，也可以通过向仓库添加集群来扩展。可以同时使用一种或两种缩放方法。



与 Snowflake 云服务层和数据存储层不同，Snowflake 虚拟仓库层（如图 2-5 所示）不是多租户架构。Snowflake 预先确定虚拟仓库中每个节点的 CPU、内存和固态硬盘（SSD）配置（参见表 2-1）。虽然这些定义可能会发生变化，但它们在所有三个云提供商的定义上是一致的。

虚拟仓库大小

Snowflake 计算集群由其大小定义，其大小对应于虚拟仓库集群中的服务器数量。对于虚拟仓库大小每增加一次，计算资源的数量平均容量就会增加一倍（请参阅表 2-1）。除了 4X-Large 之外，还使用了不同的方法来确定每个集群的服务器数量。但是，对于这些超大型虚拟仓库，每小时的积分仍然增加了 2 倍。

表 2-1. Snowflake 虚拟仓库大小和每个集群的相关服务器数量

X-小	小	中	大	1X-大	2X-大	3X-大	4X-大	
1	2	4	8	16	32	64	128	

将虚拟仓库大小调整为更大的大小，也称为纵向扩展，通常是为了提高查询性能和处理大型工作负载。这将在下一节中更详细地讨论。



由于 Snowflake 采用按秒计费，因此运行更大的虚拟仓库通常具有成本效益，因为您可以在虚拟仓库不使用时暂停它们。例外情况是，当您在大型虚拟仓库上运行大量小型或非常基本的查询时。无论并发查询的数量如何，添加额外的资源都不太可能有任何好处。

纵向扩展虚拟仓库以处理大量数据和复杂查询

在理想情况下（即简单的工作负载，每次测试完全相同），使用 X-Small 虚拟仓库与使用 4X-Large 虚拟仓库支付的总成本相同。唯一的区别是完成时间的缩短。但实际上，事情并没有那么简单。许多因素会影响虚拟仓库的性能。在调整 Snowflake 虚拟仓库的大小时，应考虑并发查询的数量、查询的表数量以及数据的大小和组成。

适当调整大小很重要。由于虚拟仓库太小而缺乏资源，可能会导致完成查询的时间过长。如果查询太小而虚拟仓库太大，则可能会对成本产生负面影响。

调整 Snowflake 虚拟仓库的大小是一个手动过程，即使在查询正在运行时也可以完成，因为无需停止或暂停虚拟仓库即可调整大小。但是，在调整 Snowflake 虚拟仓库的大小时，只有后续查询才会使用新大小。任何已经运行的查询都将完成运行，而任何排队的查询都将在新大小的虚拟仓库上运行。纵向扩展虚拟仓库将增加服务器的数量（如图 2-6 所示）。例如，从 Medium 到 Large。缩小虚拟仓库的规模将减少服务器的数量。



建议您尝试不同类型的查询和不同的虚拟仓库大小，以确定有效且高效地管理虚拟仓库的最佳方式。查询应具有一定的大小和复杂性，您通常希望在不超过 5 到 10 分钟内完成。此外，建议您从小处着手，然后在试验时增加大小。识别规模不足的虚拟仓库比识别未充分利用的虚拟仓库更容易。

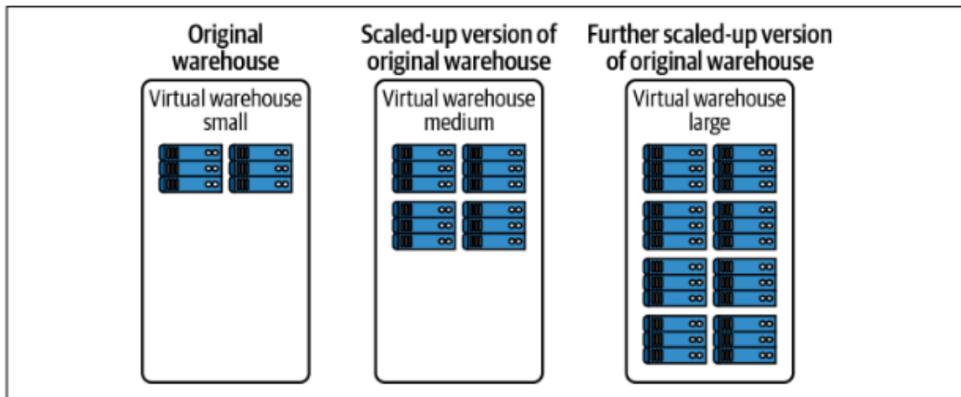


图 2-6. 纵向扩展 Snowflake 虚拟仓库会增加集群的大小

在 Snowflake 中，我们可以通过用户界面或 SQL 创建虚拟仓库。稍后，我们将使用这两种方法创建一些虚拟仓库。但首先，在我们深入研究之前，让我们回顾一下有关虚拟仓库的一些事情。

当我们使用 Snowsight 用户界面创建新的虚拟仓库时，如图 2-7 所示，我们需要知道虚拟仓库的名称以及我们希望虚拟仓库的大小。我们还需要决定是否希望虚拟仓库成为多集群虚拟仓库。多集群虚拟仓库允许 Snowflake 自动扩展和扩展。您将在下一节中了解有关多集群虚拟仓库的更多信息。

New Warehouse

Creating as SYSADMIN

Name	Size
Original_WH	Small 2 credits/hour

Comment (optional)

Multi-cluster Warehouse Scale compute resources as query needs change

Create Warehouse

图 2-7. 在 Snowflake Web UI 中创建新的虚拟仓库

创建虚拟仓库后，我们始终可以返回并调整其大小。调整虚拟仓库的大小意味着我们正在扩大或缩小它。通过编辑现有虚拟仓库来手动完成扩展和缩减。



虚拟仓库可以随时调整大小，包括在运行和处理语句时。调整虚拟仓库的大小不会对虚拟仓库当前正在执行的语句产生任何影响。调整为更大的大小将使新的、更大的虚拟仓库可用于队列中的任何查询以及任何未来的语句。

可以在用户界面中或使用工作表中的 SQL 语句来扩展或缩减虚拟仓库。您会注意到的另一件事是，您可以选择使用一些高级虚拟仓库选项，例如 Auto Suspend 和 Auto Resume（如图 2-8 所示）。默认情况下，自动暂停和自动恢复处于启用状态。启用自动暂停后，如果虚拟仓库在指定时间段内处于非活动状态，Snowflake 将自动挂起虚拟仓库。Auto Suspend 功能可确保您在没有传入查询时不会让虚拟仓库运行。这一点很重要，因为您仅在虚拟仓库运行时付费，因此您希望在虚拟仓库不使用时暂停虚拟仓库。



Auto Resume（自动恢复）和 Auto Suspend（自动暂停）时间的值应等于或超过查询工作负载中的任何常规间隔。例如，如果查询之间有 4 分钟的常规间隔，则不建议将自动暂停设置为少于 4 分钟。如果您这样做，您的虚拟仓库将持续暂停和恢复，这可能会导致更高的成本，因为计费的最低积分使用时间为 60 秒。因此，除非您有充分的理由更改默认的 Auto Suspend 和 Auto Resume 时间，否则建议将默认值保留为 10 分钟。

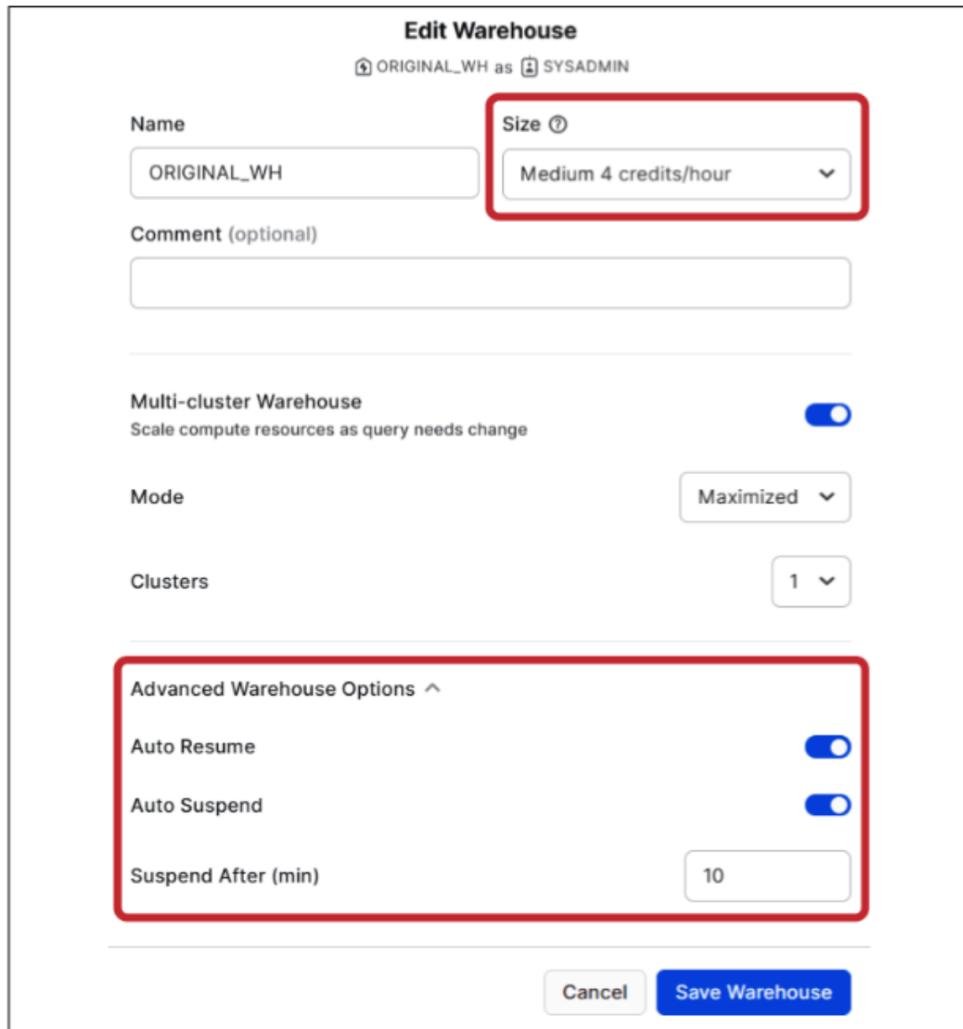


图 2-8. 在 Snowflake Web UI 中将集群大小增加到 Medium（手动扩展）



较大的虚拟仓库不一定会为查询处理或数据加载带来更好的性能。作为查询处理的一部分，查询复杂性应该是选择虚拟仓库大小的一个考虑因素，因为服务器执行复杂查询所需的时间可能比运行简单查询所需的时间要长。要加载或卸载的数据量也会极大地影响性能。具体来说，正在加载的文件数量和每个文件的大小是影响数据加载性能的重要因素。因此，在选择任何大于 Large 的虚拟仓库之前，您需要仔细考虑您的使用案例。该一般规则的一个例外是，如果您同时批量加载数百或数千个文件。

使用多集群虚拟仓库进行横向扩展以最大限度地提高并发性

多集群虚拟仓库的运行方式与单集群虚拟仓库大致相同。目标是在集群的大小和数量方面优化 Snowflake 系统的性能。在上一节中，我们了解到，当由于运行时间非常长的 SQL 查询或要加载或卸载的大量数据而导致排队问题时，扩展可能会导致性能增加，因为查询可以运行得更快。

如果并发问题是由于许多用户或连接引起的，则纵向扩展将无法充分解决问题。相反，我们需要通过添加群集来横向扩展（如图 2-9 所示），例如，从 Min Clusters 值 1 到 Max Clusters 值 3。可以将多集群虚拟仓库设置为在用户和/或查询数量趋于波动时自动扩展。



多集群虚拟仓库可用于 Enterprise、Business Critical 和 Virtual Private Snowflake 版本。

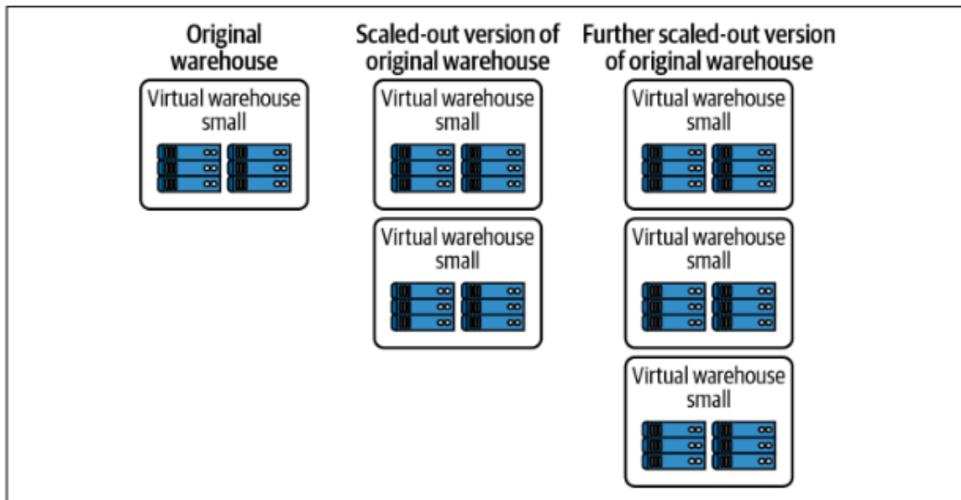


图 2-9. 横向扩展 Snowflake 虚拟仓库会增加 Snowflake 计算集群的数量

与单集群虚拟仓库一样，多集群虚拟仓库可以通过 Web 界面或使用 SQL for Snowflake 实例来创建。与单集群虚拟仓库不同，在单集群虚拟仓库中，大小调整是一个手动过程，而多集群虚拟仓库的横向扩展或横向扩展是一个自动化过程。您只需让 Snowflake 知道您希望多集群虚拟仓库横向扩展的程度。例如，您可以将 Snowflake 虚拟仓库编辑为至少具有一个小型集群和最多三个小型集群，如图 2-10 所示。

可以为多集群虚拟仓库选择两种模式：AutoScale 和 Maximized。Snowflake 扩展策略旨在帮助控制 Auto-scale 模式下的使用积分，可以设置为 Standard 或 Economy。

每当多集群虚拟仓库配置为将扩展策略设置为 Standard 时，第一个虚拟仓库会在查询排队时立即启动，或者如果 Snowflake 系统检测到查询多一个查询超过当前正在运行的集群可以执行的查询。每个连续的虚拟仓库在前一个虚拟仓库启动 20 秒后启动。

如果多集群虚拟仓库配置了扩展策略设置为 Economy，则仅当 Snowflake 系统估计查询负载可以使虚拟仓库繁忙至少六分钟时，虚拟仓库才会启动。Economy 扩展策略的目标是通过保持虚拟仓库满载来节省积分。因此，查询最终可能会排队，并且可能需要更长的时间才能完成。

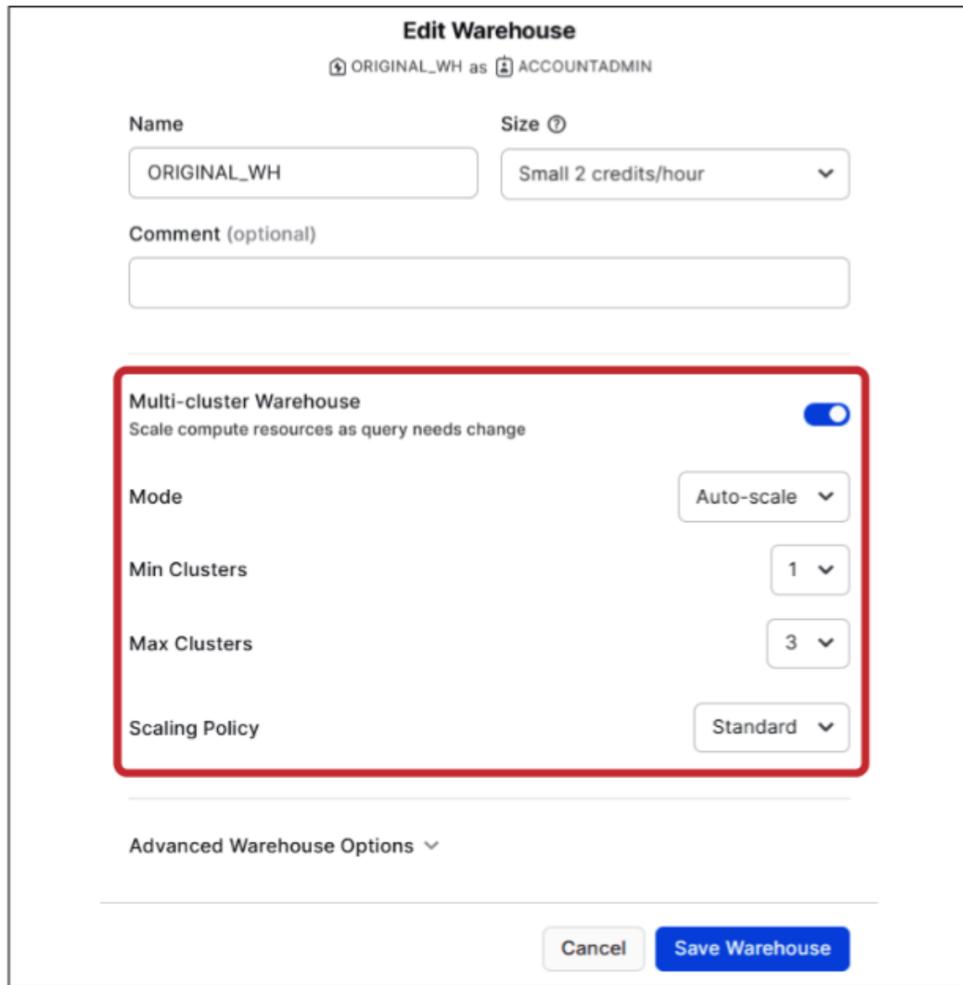


图 2-10. Snowflake Web UI 中的多集群虚拟仓库（自动扩展/缩减）

建议将 Max Clusters 值设置得尽可能高，同时注意相关成本。例如，如果您将 Max Clusters（最大集群数）设置为 10，请记住，在所有 10 个集群运行的时间长度内，您可能会遇到 10 倍的计算成本。当 Min Clusters 值大于 1 且 Min Clusters 和 Max Clusters 值相等时，多集群虚拟仓库将最大化。我们将在下一节中看到一个示例。



计算可以纵向扩展、缩减、缩减或缩减。在所有情况下，对使用的存储空间都没有影响。

创建和使用 Virtual Warehouse

虚拟仓库的命令可以在 Web UI 中执行，也可以使用 SQL 在工作表中执行。我们首先要看一下如何使用 SQL 创建和管理虚拟仓库。然后，我们将了解虚拟仓库的 Web UI 功能。

Auto Suspend（自动暂停）和 Auto Resume（自动恢复）是创建 Snowflake 虚拟仓库时可用的两个选项。Auto Suspend 是虚拟仓库在下线之前在不需要执行查询的情况下将等待的秒数。Auto Resume 将在存在需要计算资源的操作时重新启动虚拟仓库。

以下 SQL 脚本将创建一个具有四个集群的中型虚拟仓库，该仓库将在 5 分钟后自动暂停。您会注意到的一件事是，当您使用 SQL 创建虚拟仓库时，您需要以总秒为单位说明暂停时间。在用户界面中，以分钟为单位输入时间。导航到第 2 章 创建和管理 Snowflake 架构工作表，并执行以下 SQL 语句以创建新的虚拟仓库：

```
使用角色 SYSADMIN; 使用 WAREHOUSE_SIZE = MEDIUM 创建仓库  
CH2_WH  
    AUTO_SUSPEND = 300 AUTO_RESUME = 真 INITIALLY_SUSPENDED = 真;
```



最佳实践是创建处于悬而未决状态的新虚拟仓库。除非 Snowflake 虚拟仓库最初以暂停状态创建，否则 Snowflake 虚拟仓库的初始创建可能需要一些时间来预置计算资源。

之前我们讨论了如何通过扩大或缩小虚拟仓库来改变虚拟仓库的大小，并且这样做是一个手动过程。为了扩大或缩小规模，我们将使用以下命令：

```
使用角色 SYSADMIN; ALTER  
WAREHOUSE CH2_WH SET  
WAREHOUSE_SIZE = LARGE;
```

创建此虚拟仓库后在此工作簿中执行的任何 SQL 语句都将在该虚拟仓库上运行。如果您更喜欢使用某个虚拟仓库来执行脚本，则可以在工作表中指定该仓库：

使用 WAREHOUSE CH2_WH;

或者，您可以在位于 Web UI 右上角的上下文菜单中更新虚拟仓库（如图 2-11 所示）。

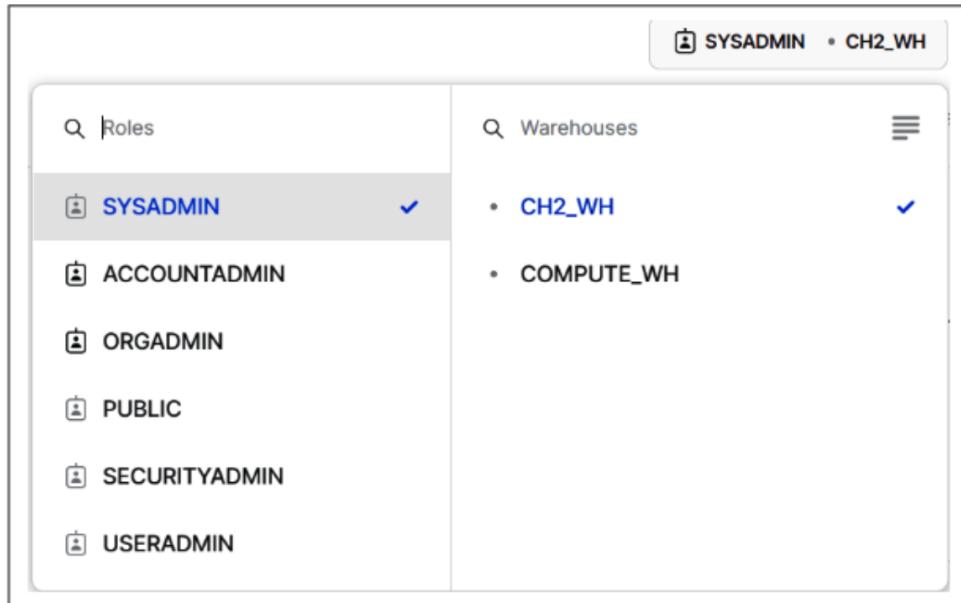


图 2-11. Snowflake Web UI 虚拟仓库选择

我们使用 SQL 命令创建了虚拟仓库，但我们也就可以使用 Snowsight Web UI 来创建和/或编辑 Snowflake 虚拟仓库。现在让我们导航到那里。单击左上角的 主页 图标，然后选择 管理员 → Ware - 房屋。您将看到我们刚刚创建的虚拟仓库（参见图 2-12 中的 CH2_WH）。如果您没有看到这两个虚拟仓库，请尝试刷新页面。

The screenshot shows the Snowflake Web UI interface. On the left, there is a sidebar with a user profile for 'Joyce Avila' (SYSADMIN) and a navigation menu with options like Worksheets, Dashboards, Data, Marketplace, Activity, Admin, Usage, and Warehouses. The 'Warehouses' option is highlighted with a blue background. The main content area is titled 'Warehouses' and shows a table with two rows:

NAME ↑	STATUS	SIZE	CLUSTERS
CH2_WH	Suspended	Large	1 - 1
COMPUTE_WH	Started	X-Small	1 - 1 (1 active)

图 2-12. 可用于当前角色的虚拟仓库

查看最右侧，单击省略号，然后选择 编辑 选项（如图 2-13 所示）。请注意，您可以通过选择 + Warehouse (+ 仓库) 按钮来选择创建新的虚拟仓库，而不是选择编辑现有虚拟仓库。

The screenshot shows the 'Warehouses' page with a focus on a specific warehouse entry. The entry for 'CH2_WH' is selected, showing it was created '13 minutes ago'. To the right of the table, there is a context menu with the following options: Edit, Resume, Drop, and Transfer Ownership.

图 2-13. Snowflake Web UI 的“编辑”选项

选择后 Edit 选项，您现在应该会看到 Edit Warehouse 屏幕，如图 2-14 所示。

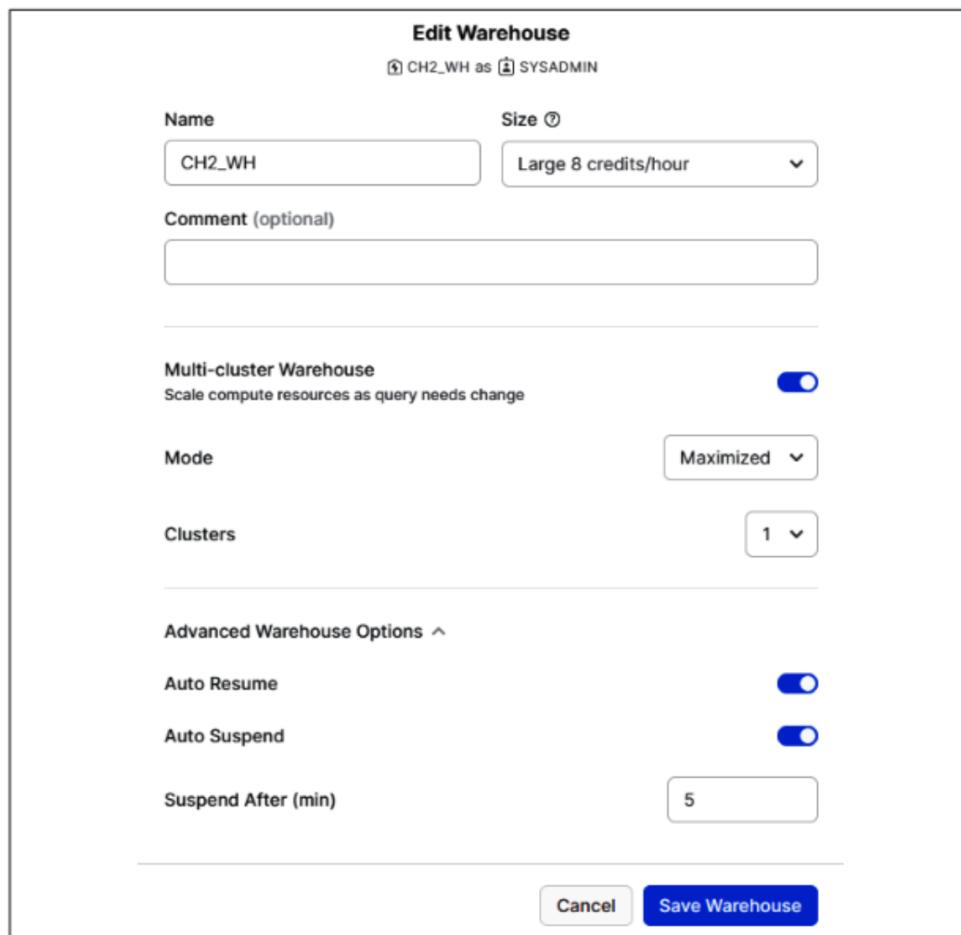


图 2-14。Snowflake Web UI 的 Edit Warehouse 屏幕

如果您在 Snowflake Standard Edition 组织中工作，则不会看到 `multiclus-ter` 虚拟仓库选项。对于企业版、业务关键版和虚拟专用 Snowflake 版，启用了多集群虚拟仓库。即使在选择多集群虚拟仓库时打开了多集群虚拟仓库，也无法横向扩展到单个集群之外。要横向扩展，集群数量需要为 2 个或更大。当您将集群数量增加到 2 个或更多时，请务必评估该模式。您很可能希望将模式从默认的 `Maximized` 模式更改为 `Auto-scale`（自动缩放）。这样，您的多集群虚拟仓库将根据需要进行扩展。

要通过 SQL 创建 Snowflake 多集群虚拟仓库，您需要指定扩展策略以及集群的最小和最大数量。如前所述，扩展策略（仅在虚拟仓库在自动扩展模式下运行时适用）可以是 `Economy` 或 `Standard`。



当最小集群数和最大集群数相同时，称为 `Maximized`（最大化）多集群虚拟仓库。此外，值必须多个。最大化多集群虚拟仓库的一个示例是 `MIN_CLUSTER_COUNT = 3` 且 `MAX_CLUSTER_COUNT = 3`。

工作负载和工作负载管理的分离

当传统数据库系统上的工作负载达到满容量时，查询处理往往会变慢。相比之下，Snowflake 会估计每个查询所需的资源，当工作负载接近 100% 时，每个新查询都会在队列中暂停，直到有足够的资源来执行它们。存在多种选项来有效处理不同大小的工作负载。一种方法是通过将不同的虚拟仓库分配给不同的用户或用户组来分离工作负载（如图 2-15 所示）。图 2-15 中描述的虚拟仓库大小相同，但在实践中，用于不同工作负载的虚拟仓库可能具有不同的大小。

可以将不同的用户组分配到不同大小的不同 Snowflake 虚拟仓库。因此，查询数据的用户将体验到一致的平均年龄查询时间。营销和销售团队可以创建和评估营销活动，同时还可以捕获销售活动。会计和财务团队可以访问他们的报告并深入研究基础数据的详细信息。数据科学家可以对大量数据运行大型 complex 查询。ETL 流程可以持续加载数据。

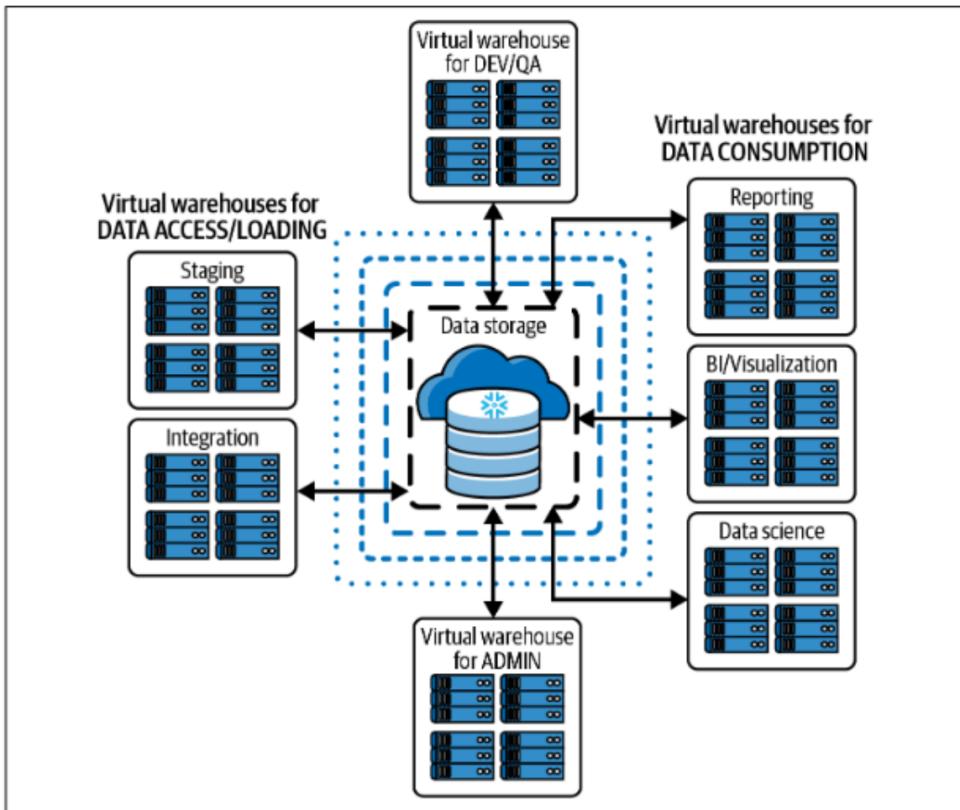


图 2-15. 通过将不同的虚拟仓库分配给用户组来分离 Snowflake 工作负载

我们在本章前面了解到，可以将多集群虚拟仓库设置为自动扩展以避免并发问题。对于自动扩展的多集群虚拟仓库，我们仍然需要定义虚拟仓库大小以及集群的最小和最大数量。之前，我们了解了如何通过 Snowflake UI 创建新的虚拟仓库。现在，让我们使用 SQL 创建用于会计和财务的多集群虚拟仓库，然后看一个示例，了解该虚拟仓库的自动扩展是如何工作的。导航回第 2 章创建和管理 Snowflake 架构工作表：

使用角色 SYSADMIN; 创建或替换仓库ACCOUNTING_WH

```
其中 Warehouse_Size = 中等 MIN_CLUSTER_COUNT = 1  
MAX_CLUSTER_COUNT = 6 SCALING_POLICY = '标准';
```

配置多集群虚拟仓库后，扩展过程会自动进行。图 2-16 说明了当并发 SQL 语句数量增加时自动扩展的工作原理。

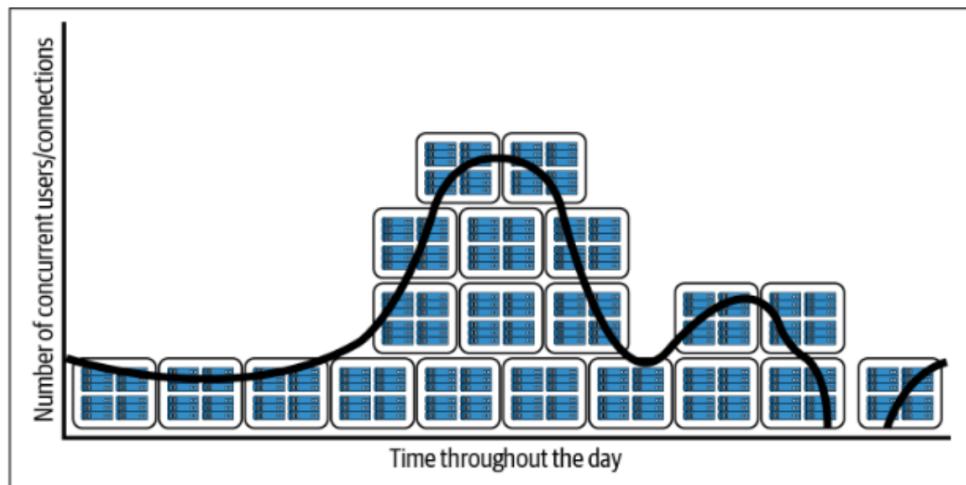


图 2-16。通过使用多集群虚拟仓库进行横向缩减和横向扩展来管理 Snowflake 工作负载

您可以看到，以小时为单位，员工的核心工作时间之间的工作量更大。我们可能还想进行调查，以确认用于报告的消费虚拟仓库在月初的日常工作量总体上较重，因为会计部门正在努力准备和审查上个月的会计报表。

Virtual Warehouse 层的计费

Snowflake 虚拟仓库的使用费用是根据仓库大小计算的，仓库大小由每个集群的服务器数量、集群数量（如果存在多集群虚拟仓库）以及每个集群服务器的运行时间决定。

Snowflake 采用按秒计费，每次启动虚拟仓库或调整虚拟仓库大小时，最短计费时间为 60 秒。当虚拟仓库扩展时，积分将按预置的额外资源的 1 分钟计费。

所有计费，即使以秒为单位计算，也以小时为单位进行报告。

使用 ACCOUNTADMIN 角色时，您可以通过单击 UI 中的 Account → Usage 来查看账户的虚拟仓库积分使用情况。您还可以在 SNOWFLAKE 共享数据库中查询 Account Usage 视图以获取信息。建议您选择 X-Small 虚拟仓库来执行此操作，因为数据集较小且查询简单。

集中式（混合列式）数据库存储层

Snowflake 的集中式数据库存储层保存所有数据，包括结构化和半结构化数据。当数据加载到 Snowflake 中时，它会以最佳方式重组为压缩的列式格式，并在 Snowflake 数据库中存储和维护。每个 Snowflake 数据库都由一个或多个架构组成，这些架构是数据库对象（如表和视图）的逻辑分组。第 3 章专门向您展示如何创建和管理数据库和数据库对象。在第 9 章中，我们将深入了解 Snowflake 的物理数据存储，以更深入地了解微分区以更好地了解数据集群。

存储在 Snowflake 数据库中的数据始终经过压缩和加密。Snowflake 负责管理数据存储方式的各个方面。Snowflake 自动将存储的数据组织成微分区，这是一种优化的、不可变的、压缩的列式格式，使用 AES-256 加密进行加密。Snowflake 优化和压缩数据，使元数据提取和查询处理更轻松、更高效。我们在本章前面了解到，每当用户对 Snowflake 查询进行子处理时，该查询将被发送到云服务优化器，然后再发送到计算层进行处理。

Snowflake 的数据存储层有时称为远程磁盘层。底层文件系统在 Amazon、Microsoft 或 Google Cloud 上实现（如图 2-17 所示）。用于数据存储的特定提供商是您在创建 Snowflake 帐户时选择的提供商。Snowflake 不会限制您可以存储的数据量或可以创建的数据库或数据库对象的数量。Snowflake 表可以轻松存储 PB 级数据。当 Snowflake 账户中的存储增加或减少时，对虚拟仓库大小没有影响。

两者彼此独立扩展，也独立于云服务层进行扩展。

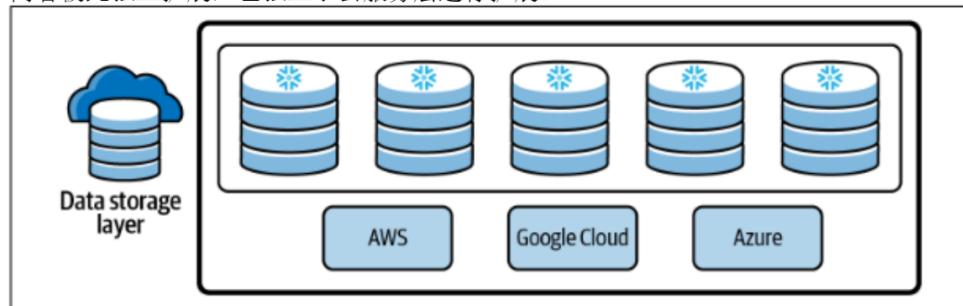


图 2-17. Snowflake 的数据存储层

存储层架构中有两个独特的功能：时间旅行和零拷贝克隆。这两个非常强大的 Snowflake 功能将在本章中介绍，并在后面的章节中更详细地介绍。为了准备后面的章节，您需要对这两个功能有透彻的了解。

零拷贝克隆简介

零副本克隆为用户提供了一种对 Snowflake 数据库、架构或表及其关联数据进行快照的方法。在对克隆对象进行更改之前，不会产生额外的存储费用，因为零副本数据克隆是仅限元数据的操作。例如，如果您克隆一个数据库，然后添加新表或从克隆的表中删除一些行，则此时将评估存储费用。除了创建备份之外，零副本克隆还有许多用途。大多数情况下，零拷贝克隆将用于支持开发和测试环境。我们将在第 8 章看到这样的例子。

时间旅行简介

Time Travel 允许您还原数据库、表或架构的先前版本。这是一个非常有用的功能，让您有机会修复以前错误完成的编辑或恢复错误删除的项目。使用 Time Travel，您还可以通过将 Time Travel 功能与克隆功能相结合来备份过去不同时间点的数据，或者您可以对不再存在的数据库对象执行简单查询。您可以追溯到过去多长时间取决于几个不同的因素。时间旅行将在第 7 章中详细讨论。就本章的目的而言，请务必注意，对于已删除但仍可恢复的任何数据，将评估数据存储费用。

存储层计费

Snowflake 数据存储成本是根据压缩数据（而不是未压缩数据）的每日平均大小计算的。存储成本包括存储在永久表中的持久性数据以及暂存用于批量数据加载和卸载的文件的成本。在计算数据存储成本时，故障安全数据和使用 Time Travel 为数据恢复而保留的数据也被考虑在内。引用已删除数据的表的克隆也同样考虑。

Snowflake 缓存

当您提交查询时，Snowflake 会检查该查询是否已预先运行，如果是，则结果是否仍处于缓存状态。如果缓存的结果集仍然可用，Snowflake 将使用缓存的结果集，而不是执行您刚刚提交的查询。除了从缓存中检索以前的查询结果外，Snowflake 还支持其他缓存技术。有三种 Snowflake 缓存类型：查询结果缓存、虚拟仓库缓存和元数据缓存。

查询结果缓存

从 Snowflake 检索数据的最快方法是使用查询结果缓存。Snowflake 查询的结果将缓存或保留 24 小时，然后清除。这与虚拟仓库缓存和元数据缓存的工作方式形成鲜明对比。这两个缓存都不会根据时间线进行清除。即使结果缓存仅保留 24 小时，每次重新执行查询时也会重置时钟，最长为自首次执行查询之日起的 31 天。31 天后（如果基础数据发生更改，则更早），在再次提交查询时，将生成并缓存新结果。

结果缓存完全由 Snowflake 全球云服务（GCS）层管理，如图 2-18 所示，并且可用于所有虚拟仓库，因为虚拟仓库可以访问所有数据。检索缓存结果的过程由 GCS 手动处理。但是，一旦结果的大小超过某个阈值，结果就会存储在云存储中并从云存储中检索。

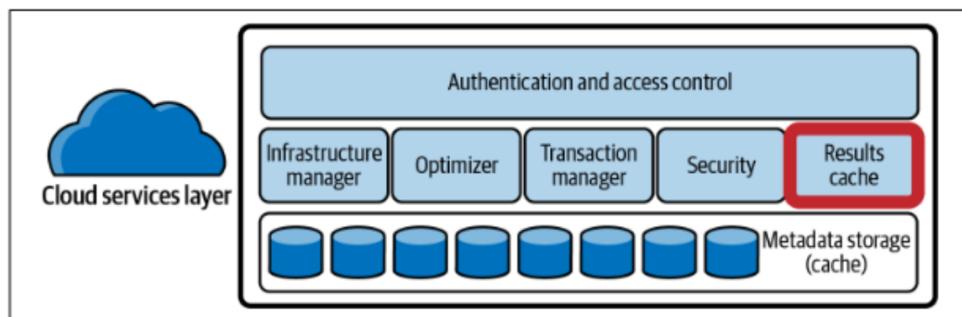


图 2-18. Snowflake 云服务层中的结果缓存

返回给一个用户的查询结果也适用于具有 necessary 访问权限并执行相同查询的任何用户。因此，任何用户都可以对结果缓存运行查询，而无需运行虚拟仓库，前提是查询已缓存且底层数据未更改。

查询结果缓存的另一个独特功能是它是唯一可以被参数禁用的缓存。导航到 Chapter2 工作表并执行以下 SQL 语句：

```
更改会话集 USE_CACHED_RESULT=FALSE;
```

在执行 A/B 测试之前，必须禁用结果缓存，并且在测试完成后启用查询结果缓存非常重要。

元数据缓存

元数据缓存在全局服务层中完全托管（如图 2-19 所示），其中用户确实对元数据有一定的控制权，但对缓存没有控制权。Snowflake 收集和管理有关表、微分区甚至集群的元数据。对于表，Snowflake 存储行数、表大小（以字节为单位）、文件引用和表版本。因此，不需要正在运行的虚拟仓库，因为在表上运行 SELECT COUNT (*) 时，计数统计信息会保留在元数据缓存中。

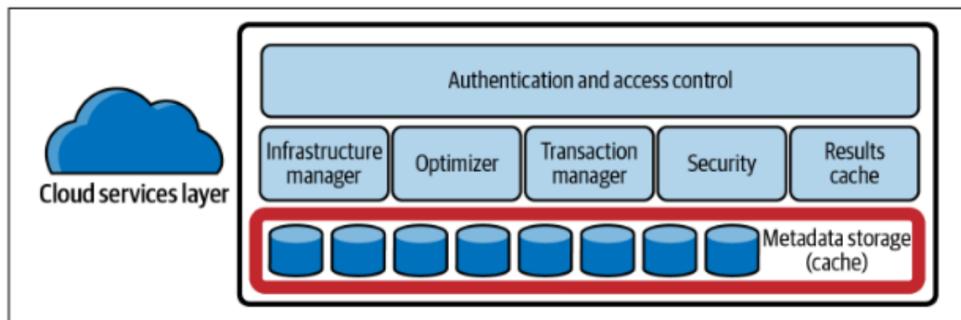


图 2-19. Snowflake 云服务层中的元数据存储缓存

Snowflake 元数据存储库包括表定义和对该表的微分区文件的引用。MIN 和 MAX 方面的值范围、NULL 计数和非重复值的数量从微分区中捕获并存储在 Snowflake 中。因此，某些查询不需要正在运行的虚拟仓库即可返回结果。例如，邮政编码的 MIN（整数数据类型列）不需要仅限虚拟计算的云服务。Snowflake 还存储微分区的总数和重叠微分区的深度，以提供有关集群的信息。



存储在元数据缓存中的信息用于构建查询执行计划。

虚拟仓库本地磁盘缓存

传统的 Snowflake 数据缓存特定于用于处理查询的虚拟仓库。正在运行的虚拟仓库使用 SSD 存储来存储处理查询时从集中式数据库存储层提取的微分区。每当执行查询时，这对于完成请求的查询都是必要的。虚拟仓库的 SSD 缓存的大小由虚拟仓库的计算资源的大小决定（如图 2-20 所示）。



虚拟仓库数据缓存的大小有限，并使用 LRU（Least Recently Used）算法。

每当虚拟仓库收到要执行的查询时，该仓库都会先扫描 SSD 缓存，然后再访问 Snowflake 远程磁盘存储。从 SSD 读取比从数据库存储层读取更快，但仍需要使用正在运行的虚拟仓库。

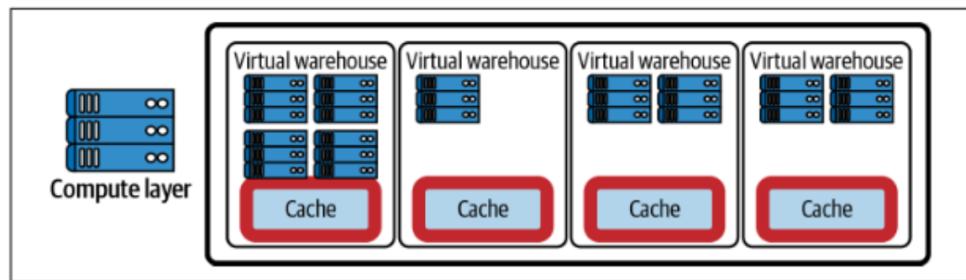


图 2-20. Snowflake 云服务层中的虚拟仓库缓存

尽管虚拟仓库缓存是在每个虚拟仓库独立运行的虚拟仓库层中实现的，但全局服务层处理整体系统数据新鲜度。它通过查询优化器来实现此目的，该优化器检查分配的虚拟仓库的每个数据段的新鲜度，然后构建查询计划，通过将任何段替换为来自远程磁盘存储的数据来更新任何段。

请注意，虚拟仓库缓存有时称为原始数据缓存、SSD 缓存或数据缓存。一旦虚拟仓库暂停，此缓存就会被丢弃，因此您需要考虑保持虚拟仓库运行所消耗的积分与维护先前查询的数据缓存以提高性能的价值之间的权衡。默认情况下，Snowflake 将在空闲时间 10 分钟后自动暂停虚拟仓库，但可以更改此设置。



只要有可能，在合理的情况下，将相同的虚拟仓库分配给将访问相同数据进行查询的用户。这增加了他们从 Virtual Warehouse 本地磁盘缓存中受益的可能性。

代码清理

在 Chapter2 工作表中，确保将缓存的结果设置回：

```
使用角色 SYSADMIN;  
更改会话 USE_CACHED_RESULT=TRUE;
```

继续并删除虚拟仓库：

```
使用角色 SYSADMIN;DROP WAREHOUSE  
CH2_WH;DROP WAREHOUSE  
ACCOUNTING_WH;
```

总结

希望前两章能让您了解 Snowflake Data Cloud 的强大功能及其易用性。在本章中，我们执行了一些 SQL 语句，但在接下来的章节中，我们将通过动手学习示例深入研究核心概念来演示 Snowflake 的许多功能。如果您还没有这样做，现在是注册 Snowflake 免费试用帐户的好时机，以便您充分利用这些章节。请参阅附录 C，了解有关在试用账户中设置的更多详细信息。此外，如果您还没有阅读第 1 章中代码示例的重要注意事项，请务必阅读。

知识检查

以下问题基于本章提供的信息：

1. 命名三个 Snowflake 架构层。
2. 三个 Snowflake 图层中哪一个是多租户图层？
3. 您将在三个 Snowflake 架构层中的哪个层中找到虚拟仓库缓存？结果缓存？
4. 如果您的 Snowflake 云服务成本高于预期，您可能需要调查哪些类型的事情？

5. 解释纵向扩展和横向扩展之间的区别。
6. 纵向扩展或横向扩展对 Snowflake 中使用的存储有什么影响？
7. 无共享架构是从共享磁盘架构演变而来的。NoSQL 替代方案也已创建。他们都试图解决什么主要问题？

8. 在 Snowflake 多集群环境中，可以选择哪些扩展策略？
9. 您需要专门为多集群虚拟仓库配置哪些组件？

10. 更改将用于在特定工作表中运行 SQL 命令的虚拟仓库的两个选项是什么？

这些问题的答案可在附录 A 中找到。

创建和管理 Snowflake 安全数据库对象

在 Snowflake 中，所有数据都存储在数据库表中。Snowflake 数据库表在逻辑上构造为行和列的集合。本章重点介绍数据库和数据库对象（如表和视图）的逻辑结构。

Snowflake 数据存储的物理结构，包括 Snowflake 微分区，将在第 9 章中解释。

在本章中，我们将按特定顺序介绍主题，因为这一系列示例是相互构建的。我们将从创建数据库和架构开始，查看 INFORMATION_SCHEMA 视图和 ACCOUNT_USAGE 视图，然后创建表和视图。接下来，我们将了解阶段、存储过程和用户定义的函数（UDF）。我们将用管道、序列、流和任务来结束本章。本章讨论的所有对象都是安全的数据库对象（如图 3-1 所示）。

Snowflake 安全对象是您为其授予特定角色访问权限的实体。已授予访问权限的角色将分配给用户。Snowflake 用户可以是个人，也可以是应用程序。用户和角色将在第 5 章中讨论。

在上一章中，我们完成了 Snowflake 虚拟仓库的动手练习。用于帮助管理虚拟仓库的 Snowflake 资源监视器将在第 8 章中详细解释。

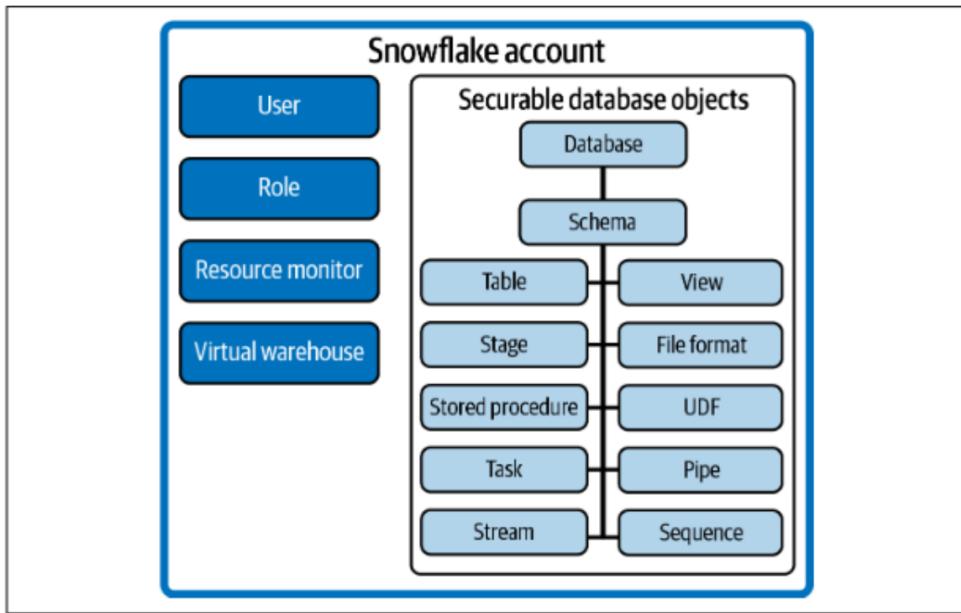


图 3-1. Snowflake 客户实体

准备工作

创建一个标题为 Chapter3 Creating Database Objects 的新工作表。如果您在创建工作表时需要帮助，请参阅第 8 页上的“导航 Snowsight 工作表”。要设置工作表上下文，请确保您使用的是 SYSADMIN 角色和 COMPUTE_WH 虚拟仓库。

创建和管理 Snowflake 数据库

在关系环境中，数据库对象（如表和视图）在数据库中维护。在 Snowflake 中，数据库对数据进行逻辑分组，而架构则对数据进行组织。数据库和架构共同构成了命名空间。在本章的示例中，每当我们使用数据库对象时，都需要指定命名空间，除非我们要使用的架构和数据库是工作区中的活动上下文。如果需要指定数据库或架构，我们将包含该命令。这样，Snowflake 就可以清楚地知道要创建对象的位置以及命令中引用的特定对象。

我们可以创建两种主要类型的数据库：永久（持久）和瞬态。在创建数据库时，如果不指定要创建两种类型中的哪一种，则默认数据库将是永久数据库。



Snowflake 旨在让您的数据在数据生命周期的每个阶段都可以访问和恢复。这是通过持续数据保护（CDP）实现的，CDP 是 Snowflake 的一整套功能，可帮助保护存储在 Snowflake 中的数据免受人为错误、恶意行为以及软件或硬件故障的影响。本章中介绍的重要 Snowflake CDP 功能是时间旅行和故障安全。

临时数据库具有最长一天的数据保留期，即 Time Travel 期，并且没有故障安全期。Snowflake Time Travel 时间段是在历史时间点查询数据库中的表数据的时间。这还允许克隆或取消删除数据库和数据库对象，并恢复历史数据。默认时间旅行周期为 1 天，但对于永久数据库，最长可达 90 天；或者，如果不需要 Time Travel period（时间旅行），用户可以将 Time Travel period（时间旅行）周期设置为零天。请注意，需要 Enterprise Edition 或更高版本才能利用 90 天的时间旅行期。

Snowflake 的故障安全数据恢复服务提供 7 天的时间，在此期间，Snowflake 可以恢复永久数据库和数据库对象中的数据。故障安全数据恢复期是数据保留期结束后的 7 天期限。与 Snowflake 用户可访问的时间旅行数据不同，故障安全数据只能由 Snowflake 员工恢复。

请务必注意，这 7 天会产生数据存储费用。这是决定要创建的数据库类型时的一个考虑因素。另一个考虑因素是，如果存储在数据库中的数据在单个数据 Time Travel 期结束后丢失，则能够从其他源恢复数据。

以下是我们将在本节中介绍的 Snowflake 数据库的基本 SQL 命令：

- 创建数据库
- 更改数据库
- 删除数据库
- 显示数据库

`CREATE DATABASE` 是用于创建新数据库、克隆现有数据库、从另一个 Snowflake 帐户提供的共享创建数据库或创建现有主数据库（即辅助数据库）的副本的命令。

我们可以从 UI 或使用 Snowflake 工作表中的 SQL 代码创建数据库。我们首先演示如何使用 UI 创建一个数据库。本章中的其余对象将使用 Snowflake 工作表中的 SQL 命令创建。



对于本章中的所有练习，请确保将角色设置为 SYSADMIN，除非另有指示。正如我们将在第 5 章中学到的那样，使用 SYSADMIN 角色而不是 ACCOUNTADMIN 或 SECURITYADMIN 角色来创建大多数 Snowflake 对象是一种最佳实践。

让我们开始吧。我们将在工作表中使用 Snowflake Web UI 创建一个永久数据库和一个使用 SQL 代码的临时数据库。导航到 Data → Databases 菜单选项，如图 3-2 所示。确认您的角色设置为 SYSADMIN 非常重要。

The screenshot shows the Snowflake Web UI interface. On the left, there's a sidebar with a user profile (Joyce Avila, SYSADMIN), navigation links for Worksheets, Dashboards, Data, and Databases (which is highlighted with a blue background), and a 'Search' bar. The main content area shows a list of databases under the heading 'Databases'. It indicates '1 Database' and lists 'SNOWFLAKE_SAMPLE_DATA'.

图 3-2. 显示 Data → Databases 选项的主菜单

在屏幕的右上角，您应该会看到一个 + 数据库 按钮（如图 3-3 所示）。

This screenshot shows the 'Databases' page again. At the top right, there's a prominent blue button with a white plus sign and the text '+ Database'. Below this button are three smaller input fields: a search bar labeled 'Search Name', a 'Source All' dropdown, and a refresh/clear button labeled 'C'.

图 3-3. 使用 + Database 按钮创建新的 Snowflake 数据库

点击 + 数据库 按钮，当 新建数据库 出现菜单，输入信息以创建新的 DEM03A_DB 数据库（如图 3-4 所示）。然后点击 创建 按钮。

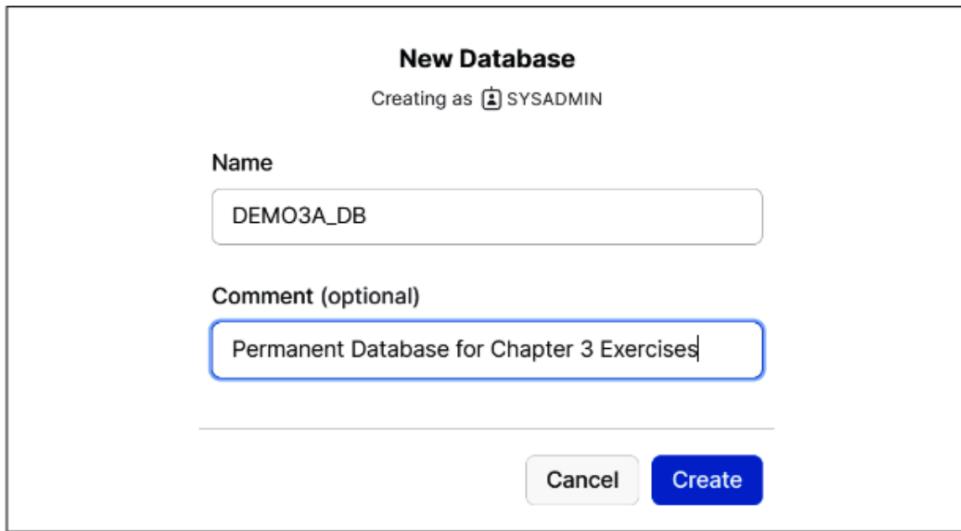


图 3-4. Snowsight 用户界面的 New Database (新建数据库) 屏幕

现在，您将看到新的 DEMO3A_DB 数据库已创建，并且是 SYSADMIN 角色可访问的两个数据库之一（如图 3-5 所示）。

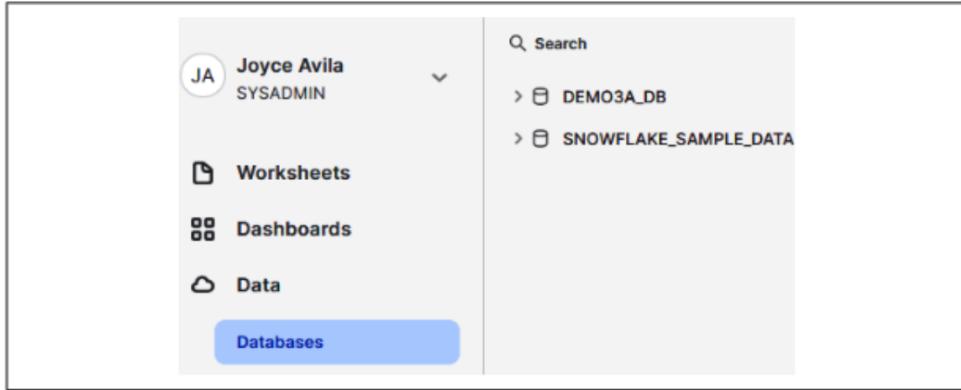


图 3-5. 可用于（当前）SYSADMIN 角色的 Snowflake 数据库列表

现在，我们将创建另一个数据库，但这次我们将使用 Snowflake 工作表来执行此操作。导航到 工作表 菜单，然后单击您之前在 准备工作 部分创建的 创建数据库对象 工作表（如图 3-6 所示）。

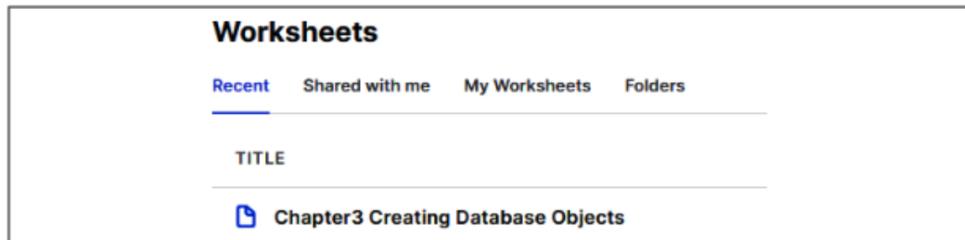


图 3-6. 显示当前工作表的 Snowsight 用户界面

确保为 SYSADMIN 角色和 COMPUTE_WH 虚拟仓库设置了上下文。执行以下语句以创建新的临时数据库：

```
使用角色 SYSADMIN; 使用 WAREHOUSE COMPUTE_WH; CREATE OR REPLACE  
TRANSIENT DATABASE DEM03B_DB comment = '第 3 章练习的临时数据  
库';
```

请注意，在前面的代码中，我们在命令中使用了可选关键字。这样，如果数据库已存在，则不会返回错误，但现有数据库将被完全覆盖。



谨慎，以避免覆盖现有数据库。作为替代方法，如果您不想出现错误，但也不想覆盖现有数据库，请使用 EXISTS 语句。和 optional 关键字不能在同一语句中同时使用。如第 1 章所述，选择在整本书中使用可选关键字，以便您可以轻松地返回本章中的任何位置来修改某个部分或进行更正。但是，这种方法在您的生产环境中并不是最佳实践。有关最佳实践的更多信息，请参阅附录 B。

每当创建新数据库时，该数据库都会自动设置为当前会话的活动数据库。这相当于使用 command。如果我们需要或想要使用我们刚刚创建的数据库以外的数据库，则必须在工作表中包含该命令以选择合适的数据库。或者，我们可以使用下拉菜单来选择不同的数据库。

如果导航到 Databases UI (数据库 UI)，如图 3-7 所示，您将看到它显示了我们刚刚创建的两个数据库以及 Snowflake 帐户自动附带的 Snowflake 示例数据库。

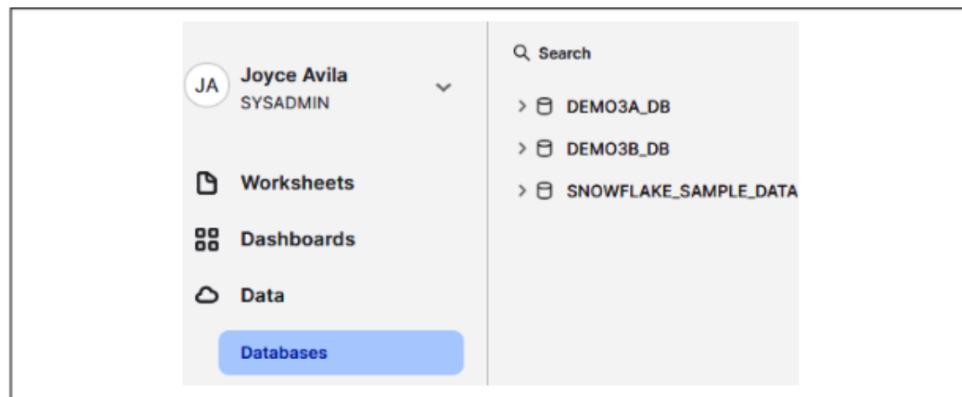


图 3-7. (当前) SYSADMIN 角色的可用数据库列表

导航回 Creating Database Objects 工作表，然后使用右上角的下拉菜单将您的角色更改为 ACCOUNTADMIN。打开工作表中数据库的下拉菜单，您现在将看到另一个数据库，即 SNOWFLAKE 数据库，作为您可以选择的数据库之一包含在内（如图 3-8 所示）。

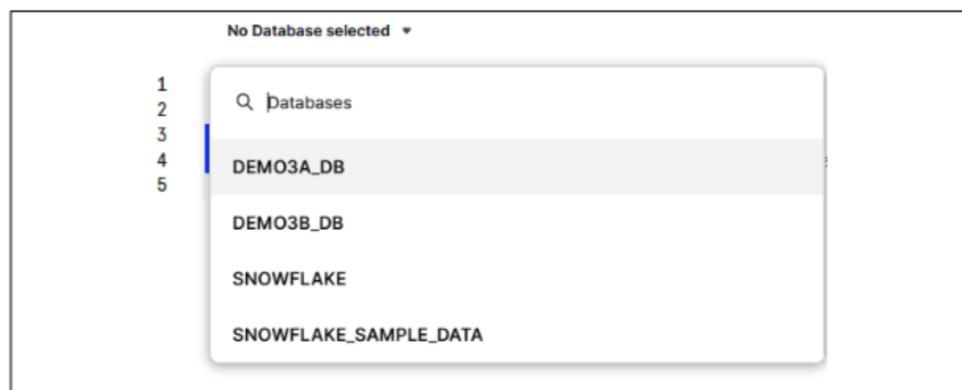


图 3-8. 当前 Snowsight 工作表中的可用数据库列表

在工作表中使用 ACCOUNTADMIN 角色时，执行以下命令：

```
使用角色 ACCOUNTADMIN;显示  
数据库;
```

请注意，在图 3-9 中，所有数据库（包括我们刚刚创建的数据库）都有一天的保留时间。数据保留时间（以天为单位）与 Time Travel 天数相同，并指定删除后保留基础数据的天数，并且可以对数据库执行 `and` 命令。

	name	is_default	is_current	origin	owner	comment	options	retention_time
1	DEMO3A_DB	N	N		SYSADMIN	Permanent		1
2	DEMO3B_DB	N	Y		SYSADMIN	Transient	TRANSIENT	1
3	SNOWFLAKE	N	N	SNOWFLAKE				1
4	SNOWFLAKE_SAMPLE_DATA	N	N	SFC_SAMPLE	ACCOUNTADMIN	Provided by		1

图 3-9. 命令的结果

我们可以更改永久数据库的数据保留时间，但不能更改临时数据库的数据保留时间。我们可以将永久数据库的保留时间更改为最多 90 天，前提是我们在 Enterprise 或更高版本的 Snowflake 版本。我们将继续使用 `DATABASE` 命令将永久数据库的保留时间更改为 10 天。在发出以下命令之前，请务必更改您的角色回 `SYSADMIN`：

```
使用角色 SYSADMIN;
ALTER DATABASE DEMO3A_DB
设置 DATA_RETENTION_TIME_IN_DAYS=10;
```

我们刚刚更改了 `DEMO3A_DB` 数据库的数据保留时间。让我们看看如果我们尝试更改临时数据库的数据保留时间会发生什么情况：

```
使用角色 SYSADMIN;
ALTER DATABASE DEMO3B_DB
设置 DATA_RETENTION_TIME_IN_DAYS=10;
```

如果运行前面的语句，您将收到一条错误消息，告知您值 10 是 `DATA_RETENTION_TIME_IN_DAYS` 参数的无效值。这是因为临时数据库的最长数据保留时间为一天。您可以将保留时间更改为零天，但这样您将无法或取消删除该数据库。

我们将在后面的部分中更详细地介绍表，但现在，重要的是要提及有关表的一些事项，因为它们与永久数据库和临时数据库有关。

Snowflake 在处理永久数据库时采用混合方法，但不适用于临时数据库。永久数据库不限于可以存储在其中的不同类型的对象。例如，您可以将临时表存储在永久数据库中，但不能将永久表存储在临时数据库中。提醒一下，临时表旨在保存

不需要与永久表相同级别的保护和恢复，但需要在会话之后进行维护。

以下是在我们的 `transient` 数据库中创建一个表的示例：

```
使用角色 SYSADMIN;
创建或替换表 DEMO3B_DB。公共。总结
(CASH_AMT号、
RECEIVABLES_AMT号、
CUSTOMER_AMT号) .
```

请注意，我们没有将表的类型指定为 `permanent` 或 `transient`。默认情况下，Snowflake 会在数据库中创建一个永久表，除非您在创建表时指示其他方面。当您在临时数据库中创建表时，情况会例外。在这种情况下，表也必须是 `transient`。

默认情况下，在临时架构中创建的所有表都是临时的。我们可以通过使用命令来看到情况，它为我们提供了图 3-10 中我们刚刚创建的表的结果。

	name	database_name	schema_name	kind
1	SUMMARY	DEMO3B_DB	PUBLIC	TRANSIENT

图 3-10. 使用命令的结果



可以在 Snowflake 帐户中创建的数据库对象、架构和数据库的数量没有限制。

每个 Snowflake 帐户还附带已包含的某些数据库、架构和表，如图 3-11 所示。以下两个数据库最初附带一个 Snowflake 帐户：

- SNOWFLAKE 数据库
- SNOWFLAKE_SAMPLE_DATA 数据库

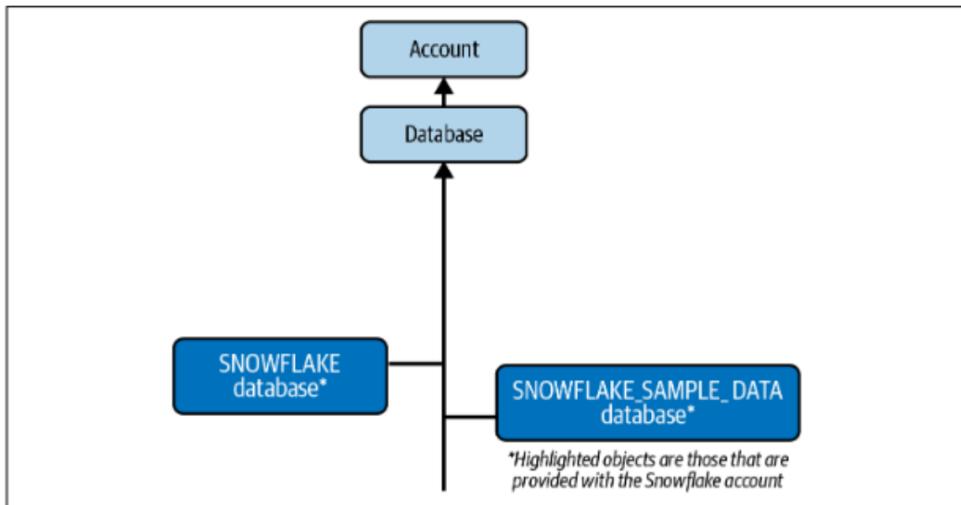


图 3-11. Snowflake 数据库的对象层次结构

SNOWFLAKE 数据库归 Snowflake Inc. 所有，是一个系统定义的只读共享数据库，可提供有关您账户的对象元数据和使用情况指标。与设置时导入您账户的 SNOWFLAKE_SAMPLE_DATA 数据库不同，无法从您的账户中删除 SNOWFLAKE 数据库。



使用 SYSADMIN 角色时，我们无法看到 SNOWFLAKE 数据库，因为默认情况下，SNOWFLAKE 数据库仅对使用 ACCOUNTADMIN 角色的用户显示。

但是，可以将该权限授予其他角色。

如果您在 2022 年之前注册了 Snowflake 试用账户，则您的账户可能还包含 UTIL_DB 数据库和 DEMO_DB 数据库，这些数据库在设置时也导入到您的账户中。如果需要，可以从 Snowflake 帐户中删除这两个数据库。

乍一看，SNOWFLAKE_SAMPLE_DATA 数据库似乎与您在 Snowflake 帐户中创建的数据库相似。但是，sample 数据库实际上是从 Snowflake SFC_SAMPLES 帐户共享的数据库，并且该数据库在您的帐户中是只读的，这意味着无法对该数据库执行任何数据定义语言（DDL）命令。换句话说，不能在样本数据库中添加、删除或更改数据库对象。此外，不能对表执行用于克隆等操作的数据操作语言（DML）命令。但是，您可以查看示例数据库并对表执行查询。

我们将在本章的一些示例中使用 SNOWFLAKE_SAMPLE_DATA 数据库。在第 10 章中，我们将了解共享数据库，但现在需要了解的是，虽然共享示例数据库不会产生任何存储成本，但我们确实需要一个正在运行的虚拟仓库来运行查询，因此在 Snowflake 示例数据库上运行这些查询将产生相关的计算成本。

尽管这很明显，但关于如何构建解决方案以及是否选择永久或临时对象来存储数据，一个重要的考虑因素是在 Snowflake 中存储数据会产生货币成本，并且会对性能产生影响。我们将在后面的章节中更详细地讨论这两者。



Snowflake 账户自动附带的数据库（包括 SNOWFLAKE 账户）都不会产生数据存储费用。

创建和管理 Snowflake 架构

当我们创建数据库时，我们不必指定账户，因为我们一次只能在一个账户中操作。但是当我们创建 Schema 时，我们需要让 Snowflake 知道我们想使用哪个数据库。如果我们没有指定特定的数据库，Snowflake 将使用处于活动状态的数据库。

有多种方法可以创建可实现相同最终结果的架构。以下是完成相同任务的两个示例。在此示例中，该命令让 Snowflake 知道将为哪个数据库创建架构：

使用角色 SYSADMIN; 使用数据库 DEMO3A_DB;
创建或替换 SCHEMA BANK;

在此示例中，我们只使用完全限定的 schema name：

使用角色 SYSADMIN;
创建或替换 SCHEMA DEMO3A_DB。银行业；

如果我们使用如图 3-12 所示的命令，我们会注意到新架构的保留时间也有 10 天的保留时间，就像创建它的数据库一样，而不是默认的 1 天保留。

	created_on	name	is_default	is_current	database_name	owner	comment	options	retention_time
1	1.408 -0700	RANKING	N	Y	DEMO3A_DB	SYSADMIN			10
2	8.320 -0700	INFORMATION_SCHEMA	N	N	DEMO3A_DB	SYSADMIN	Views describing the contents of schemas in this database		10
3	8.789 -0700	PUBLIC	N	N	DEMO3A_DB	SYSADMIN			10

图 3-12. 命令的结果

但是，我们始终可以将架构的保留时间更改为一天：

```
使用角色 SYSADMIN;ALTER SCHEMA  
DEMO3A_DB.BANKING SET  
DATA RETENTION TIME IN DAYS=1;
```

现在再次运行该命令，您将看到 BANKING 架构的保留时间已更改。

在上一节中，我们在 DEMO3B_DB 中创建了一个 SUMMARY 表。PUBLIC 架构。假设我们现在已决定希望该表存在于不同的架构中。我们如何实现这一目标？首先，我们将创建新架构（如果它不存在），然后我们将使用新创建的架构重命名表：

```
使用角色 SYSADMIN;  
创建或替换 SCHEMA DEMO3B_DB。银行业;  
ALTER TABLE DEMO3B_DB. 公共。摘要重命名为  
DEMO3B_DB。银行业。总结;
```



与数据库一样，架构可以是永久的，也可以是暂时的，默认值是永久的。就像数据库一样，我们也可以使用相同的 SQL 命令。但是，对于架构，我们有一些独特的东西，称为托管访问架构。在托管访问架构中，架构所有者管理对架构中对象（如表和视图）的授权，但对对象没有任何、或权限。

以下代码演示如何创建具有托管访问权限的架构：

```
使用角色 SYSADMIN; 使用数据库DEMO3A_DB;  
创建架构 MSCHEMA 或将其替换为托管访问权限;
```

现在，当您运行该命令时，您会注意到 MANAGED ACCESS 将显示在名为 MSCHEMA 的架构的 options 列下（如图 3-13 所示）。

	name	...	is_default	is_current	database_name	owner	comment	options
1	BANKING	N	N		DEMO3A_DB	SYSADMIN		
2	INFORMATION_SCHEMA	N	N		DEMO3A_DB		Views described	
3	MSCHEMA	N	Y		DEMO3A_DB	SYSADMIN		MANAGED ACCESS
4	PUBLIC	N	N		DEMO3A_DB	SYSADMIN		

图 3-13. 对具有托管访问权限的架构的命令结果

正如我们将在第 5 章中看到的那样，对于常规模式，对象所有者角色可以授予其他角色对象访问权限，也可以授予这些角色管理对象授权的能力。但是，在托管访问架构中，
对象所有者无法

颁发授予权限。相反，只有 schema owner 或拨款分配给它的权限可以管理授予权限。



SECURITYADMIN 和 ACCOUNTADMIN 角色本身具有特权。因此，这两个角色都可以管理所有托管架构的授权特权。

如图 3-14 所示，创建的每个数据库中都包含两个数据库架构：INFORMATION_SCHEMA 和 PUBLIC。PUBLIC 模式是默认模式，可用于创建任何其他对象，而 INFORMATION_SCHEMA 是系统的特殊模式，其中包含视图和表函数，这些函数提供对数据库和帐户元数据的访问。INFORMATION_SCHEMA 将在下一节中讨论。

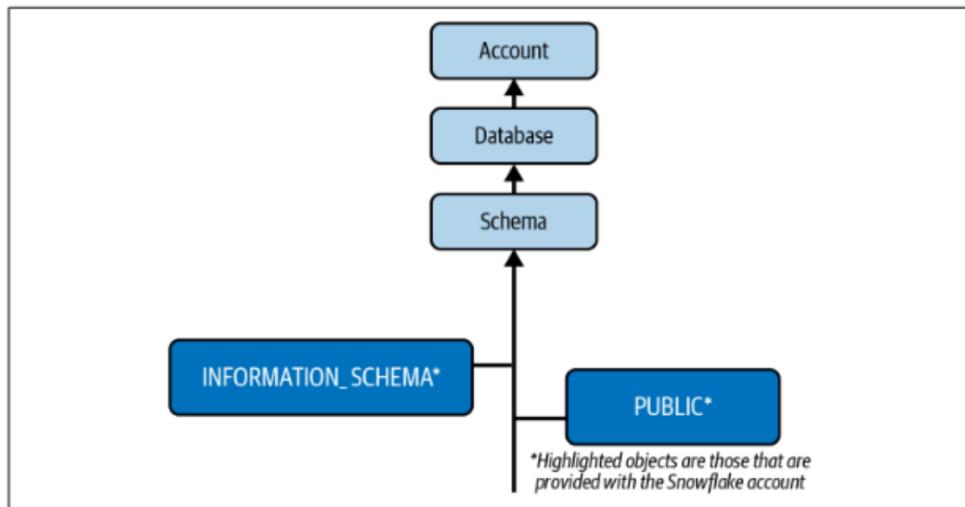


图 3-14. Snowflake 架构的对象层次结构

INFORMATION_SCHEMA

正如我们刚刚了解到的，Snowflake INFORMATION_SCHEMA 包含在 Snowflake 中创建的每个数据库中。INFORMATION_SCHEMA 也称为数据字典，包括有关数据库中对象的元数据信息以及账户级对象（如角色）。在第 8 章中，我们将探讨 INFORMATION_SCHEMA table 函数，这些函数可用于返回历史信息和账户使用信息。现在，我们将更详细地探索 INFORMATION_SCHEMA 数据库视图。

每个 INFORMATION_SCHEMA 中都包含 20 多个系统定义的视图。这些视图可以分为两类：账户视图和数据库视图。

INFORMATION_SCHEMA 账户视图包括以下内容：

APPLICABLE_ROLES

为每个角色授予显示一行

数据库

为账户中定义的每个数据库显示一行

ENABLED_ROLES

为会话中当前启用的每个角色显示一行

INFORMATION_SCHEMA_CATALOG_NAME

显示 INFORMATION_SCHEMA 所在的数据库的名称

LOAD_HISTORY

为使用该命令加载到表中的每个文件显示一行，并返回过去 14 天内加载的所有数据的历史记录，但使用 Snowpipe 加载的数据除外

REPLICATION_DATABASES

为组织中的每个主数据库和辅助数据库（即已启用复制的数据库）显示一行

您可能希望查看每个视图中的内容。您会注意到，对于其中一些数据视图，帐户中的所有数据视图都包含相同的信息。尝试以下两个 SQL 语句，一次一个：

```
SELECT * FROM SNOWFLAKE_SAMPLE_DATA.INFORMATION_SCHEMA.DATABASES;从  
DEMO3A_DB中选择 *.INFORMATION_SCHEMA.DATABASES;
```

您会注意到每个语句的结果是相同的，如图 3-15 所示。

DATABASE_NAME	DATABASE_OWNER	IS_TRANSIENT	COMMENT
DEMO3A_DB	SYSADMIN	NO	Permanent Database for Chapter 3 Exercises
DEMO3B_DB	SYSADMIN	YES	Transient Database for Chapter 3 Exercises
SNOWFLAKE_SAMPLE_DATA	ACCOUNTADMIN	NO	Provided by Snowflake during account provisioning

图 3-15. INFORMATION_SCHEMA 帐户视图返回相同的结果，但所选数据库较少

现在尝试以下两个 SQL 语句，一次一个：

```
SELECT * FROM SNOWFLAKE_SAMPLE_DATA.INFORMATION_SCHEMA.APPLICABLE_ROLES;从  
DEMO3A_DB中选择 *.INFORMATION_SCHEMA.APPLICABLE_ROLES;
```

同样，您会注意到每个语句的结果是相同的。对于所有 INFORMATION_SCHEMA 账户视图，无论您查询哪个数据库INFORMATION_SCHEMA，您都将收到相同的信息。

现在，我们来看一下 INFORMATION_SCHEMA 数据库视图。我们将在查询结果中看到，INFORMATION_SCHEMA 数据库视图将返回特定于数据库的结果。

INFORMATION_SCHEMA 数据库视图包括以下视图：

列

为指定（或当前租赁）数据库中定义的表中的每一列显示一行。

EXTERNAL_TABLES

为指定（或当前）数据库中的每个外部表显示一行。

FILE_FORMATS

为指定（或当前）数据库中定义的每种文件格式显示一行。

功能

为指定（或当前）数据库中定义的每个 UDF 或外部函数显示一行。

OBJECT_PRIVILEGES

为您的账户中定义的所有对象授予的每个访问权限显示一行。

管道

为指定（或当前）数据库中定义的每个管道显示一行。

程序

为指定（或当前）数据库定义的每个存储过程显示一行。

REFERENTIAL_CONSTRAINTS

为指定（或当前）数据库中定义的每个引用完整性约束显示一行。

图式

为指定（或当前）数据库中的每个架构显示一行。

序列

为指定（或当前）数据库中定义的每个序列显示一行。

阶段

为指定（或当前）数据库中定义的每个阶段显示一行。

TABLE_CONSTRAINTS

为指定（或当前）数据库中的表定义的每个引用完整性约束显示一行。

TABLE_PRIVILEGES

为已授予指定（或当前）数据库中的每个角色的每个表特权显示一行。

TABLE_STORAGE_METRICS

显示表级存储利用率信息，包括表元数据，并显示为每个表计费的存储类型数。行将在此视图中保留，直到相应的表不再为任何存储计费，无论表中的数据可能处于各种状态（即活动、时间旅行、故障安全或克隆保留）。

表

为指定（或当前）数据库中的每个表和视图显示一行。

USAGE_PRIVILEGES

为指定（或当前）数据库中的序列定义的每个权限显示一行。

视图

为指定（或当前）数据库中的每个视图显示一行。

在 Snowflake 中，有多种方法可以查看特定于数据库的元数据，其中一些方法使用 INFORMATION_SCHEMA 数据库视图。如果您尝试以下两个命令，您将看到有两种不同的方法可以获取有关 Snowflake 示例数据库中的架构的信息：

```
SELECT * FROM SNOWFLAKE_SAMPLE_DATA.INFORMATION_SCHEMA.图式;在数据库  
SNOWFLAKE_SAMPLE_DATA 中显示架构;
```

您会注意到的一点是，与仅使用命令时相比，INFORMATION_SCHEMA 中包含的元数据要完整得多，信息列也多了几列。

让我们进一步深入研究，看看我们之前创建的两个数据库中的表权限。提醒一下，我们在第二个数据库中创建了一个表，但尚未在第一个数据库中创建任何表。那么，如果我们尝试获取有关每个数据库中的表的信息，会发生什么呢？分别运行这两个语句中的每一个以找出：

```
从 DEMO3A_DB中选择 *.INFORMATION_SCHEMA.TABLE_PRIVILEGES;从  
DEMO3B_DB中选择 *.INFORMATION_SCHEMA.TABLE_PRIVILEGES;
```

您会注意到，第二个查询返回了一行，而第一个查询没有返回任何行。原因是第一个数据库中没有表；因此，将没有 table 权限。

INFORMATION_SCHEMA 是每个 Snowflake 数据库附带的两个架构之一，具有多种用途。此外，INFORMATION_SCHEMA 还提供了有关账户的对象元数据和使用情况指标的大量信息。Snowflake 中还有另一个存储对象元数据和使用情况指标的位置，我们将在下一节中介绍。

ACCOUNT_USAGE 架构

默认情况下，SNOWFLAKE 数据库可由 ACCOUNTADMIN 查看，它包括一个 ACCOUNT_USAGE 架构，该架构与 INFORMATION_SCHEMA 非常相似，但有三个不同之处。

- SNOWFLAKE 数据库ACCOUNT_USAGE 架构包含已删除对象的记录，而 INFORMATION_SCHEMA 则不包含。
- ACCOUNT_USAGE 架构对历史使用情况数据的保留时间较长。INFORMATION_SCHEMA 的可用数据从 7 天到 6 个月不等，而 ACCOUNT_USAGE 视图将历史数据保留一年。
- INFORMATION_SCHEMA 中的大多数视图都没有延迟，但ACCOUNT_USAGE 的延迟时间可能从 45 分钟到 3 小时不等。具体而言，对于INFORMATION_SCHEMA，更新 ACTIVE_BYTES、TIME_TRAVEL_BYTES、FAILSAFE_BYTES 和 RETAINED_FOR_CLONE_BYTES 的存储相关统计信息可能会延迟 1 到 2 小时。

ACCOUNT_USAGE 架构的常见用途之一是跟踪您账户中每个虚拟仓库在一段时间内（月初至今）使用的积分。将您的角色更改为 ACCOUNTADMIN 并执行以下语句：

```
使用角色 ACCOUNTADMIN; 使用数据库 SNOWFLAKE; 使用 SCHEMA ACCOUNT_USAGE; 使用
WAREHOUSE COMPUTE_WH; 选择 start_time: :date AS USAGE_DATE、WAREHOUSE_NAME、

SUM (credits_used) AS TOTAL_CREDITS_CONSUMED FROM
warehouse_metering_history 其中 start_time >= date_trunc
(Month, current_date) GROUP BY 1,2 ORDER BY 2,1;
```

SNOWFLAKE 数据库（包括 ACCOUNT_USAGE 架构）仅可用于 ACCOUNTADMIN 角色，除非 ACCOUNTADMIN 将从基础共享资源导入的权限授予其他角色。我们将在后面的章节中更详细地探讨 ACCOUNT_USAGE 模式，尤其是在

第 8 章和第 9 章将进一步了解如何提高性能和降低成本。不过，现在，您刚刚执行的代码的结果应该类似于图 3-16 中所示。您会注意到，COMPUTE_WH 虚拟仓库和云服务会占用积分。

WAREHOUSE_NAME	TOTAL_CREDITS_CONSUMED
CLOUD_SERVICES_ONLY	0.105878667
COMPUTE_WH	0.278027222

图 3-16. COMPUTE_WH 虚拟仓库和云服务使用的积分



在继续下一部分之前，请务必将您的角色更改回 SYSADMIN。

架构对象层次结构

到目前为止，我们已经了解了 schema 对象，并探索了每个 Snowflake 数据库附带的两种 schema。回顾一下图 3-14，我们可以看到存在于 Schema 对象之上的 schema 对象层次结构，包括 database 对象和 account。接下来，我们要探索层次结构中架构下的 Snowflake 对象。

Snowflake 架构对象中存在许多对象，包括表、视图、阶段、策略、存储过程、UDF 等。在接下来的部分中，我们将仔细研究其中的几个 Snowflake 对象。虽然我们将详细探讨其中一些雪花天体，但本章中的解释是基础性的；我们将在后续的许多章节中更深入地探讨这些对象。

Snowflake 表简介

如前所述，所有 Snowflake 数据都存储在表中。除了 permanent 和 transient 表之外，还可以创建混合表、临时表和外部表，如图 3-17 所示。Snowflake 混合表支持新的 Unistore 工作负载，第 12 章将对此进行更详细的介绍。Snowflake 临时表仅存在于创建它们的会话中，并且经常用于存储临时数据，例如 ETL 数据。Snowflake 外部表使您能够直接处理或查询存在于其他位置的数据，而无需将其提取到 Snowflake 中，包括位于数据湖中的数据。

处理外部数据的一种新方法是将 Apache Hive 元存储与 Snowflake 集成。您可以用新的 Hive Metastore 连接器连接到您的 Hadoop 环境。如果您使用的是较新的技术，例如 Delta Lake 或 Apache Iceberg，也可以使用 Snowflake 支持。

Delta Lake 是基于数据湖 Spark 的平台上的一种表格式。可以创建一个 Snowflake 外部表，该表将引用您的 Delta Lake 云存储位置。

Apache Iceberg 表解决了与对象存储相关的许多挑战，这使得它成为数据湖的热门选择。Hive 在文件夹级别跟踪数据，而 Iceberg 使用持久树结构跟踪表中所有文件的完整列表。在文件夹级别跟踪数据可能会导致性能问题，并且在文件夹级别执行文件列表操作时，数据可能会看起来好像丢失了一样。Apache Iceberg 表格式被许多领先的技术公司使用，包括 Apple、AWS、Expedia、LinkedIn 和 Netflix。

2022 年 6 月，Snowflake 宣布推出一种新的表类型，即动态表（以前称为物化表）。动态表允许用户以声明方式指定可以进行转换的管道。然后，Snowflake 处理递增的 mental refresh 以具体化数据。通过这种方式，Snowflake 动态表会随着新数据的流入而自动刷新。动态表具有具体化视图和任务的一些特征，可以将其视为流的逻辑进程。这种新的动态表类型使 Snowflake 用户能够显著改善其增量具体化体验。与数据库和模式一样，我们可以使用 、 和 命令。此外，我们还需要使用 或 将数据放入表中。对于我们创建的 Snowflake 表，我们还可以使用 or 命令从表中删除数据，但不能删除表对象本身。



TRUNCATE 和 的不同之处在于，它还会清除表加载历史元数据，同时保留元数据。

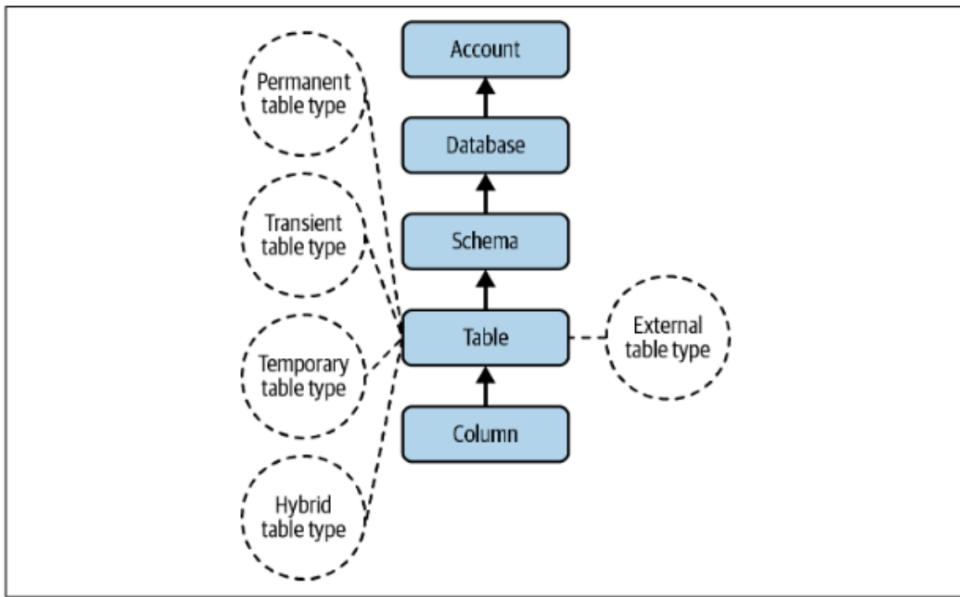


图 3-17. Snowflake 表的对象层次结构

正如我们在第 54 页的“创建和管理 Snowflake 数据库”中所看到的，如果未指定表类型，Snowflake 假定它应该创建一个永久表，除非该表是在临时数据库中创建的。

临时表是 Snowflake 独有的，并且具有永久表和临时表的特征。临时表专为需要在会话之外维护的临时数据而设计，但不需要与永久表相同级别的数据恢复。因此，临时表的数据存储成本将低于永久表的数据存储成本。临时表和永久表之间的最大区别之一是，没有为临时表提供故障安全服务。

无法使用该命令将永久表更改为临时表，因为该属性是在创建表时设置的。同样，临时表不能转换为永久表。如果要更改临时或永久表类型，则需要创建一个新表，使用子句，然后复制数据。使用子句将确保 table 将继承任何显式访问权限。

前面提到过，除非另有指定，否则创建 table 的默认值是 create a permanent table。如果默认情况下将新表自动创建为瞬态类型是有意义的，则可以首先创建瞬态数据库或架构。正如我们在第 54 页的“创建和管理 Snowflake 数据库”中所看到的，之后创建的所有表都将是临时的，而不是永久的。

具有必要权限的其他用户可以访问临时表。但是，临时表仅存在于创建它们的会话中。这意味着其他用户无法使用它们，也无法克隆它们。临时表有许多用途，包括用于 ETL 数据和特定于会话的数据需求。



会话结束后，临时表及其中的数据将无法再访问。在临时表存在期间，它确实计入存储成本；因此，一旦您不再需要临时表，最好将其删除。

表 3-1 总结了不同 Snowflake 表的一些特征。

表 3-1. Snowflake 表特征

特性	永久表	临时表	临时表	外部表	坚持	Until 显式删除	Until 显式删除	会话的剩余部分	Until 明确	下降	下降
时间旅行保留期（天）	0-90	0 或 1.0 或 1.0	故障安全期（天）	7							
可以克隆吗？	是	是	是否	是否	可以创建视图？	是	是	是	是		
^	在企业版及更高版本中，为 0-90 天。在 Standard Edition 中，为 0 或 1 天。										



有趣的是，您可以创建一个与同一架构中的现有表同名的临时表，因为该临时表是基于会话的。不会给出任何错误或警告。因此，最佳做法是为临时表提供唯一的名称，以避免在临时表优先的情况下出现意外问题。

现在，我们将创建一些表，我们将在本章后面使用。确保您使用 SYSADMIN 角色创建表：

```
使用角色 SYSADMIN; 使用数据库 DEM03A_DB;
创建或替换 SCHEMA BANK; 创建或替换表
CUSTOMER_ACCT
(Customer_Account int、金额 int、transaction_ts 时间戳); 创建或替换
TABLE CASH
(Customer_Account int、金额 int、transaction_ts 时间戳); 创建或替换
表应归档
(Customer_Account int、金额 int、transaction_ts 时间戳);
```

创建此表后，活动角色为 SYSADMIN 角色，活动数据库为 DEMO3A_DB，活动架构为 BANKING。我们使用命令指定了数据库。然后，我们创建了 BANKING 架构，这使得 DEMO3A_DB。BANKING 默认架构。因此，如果您创建新表而不专门使用不同的命名空间，则新创建的表将位于 BANKING 架构中。现在让我们试试：

```
使用角色 SYSADMIN;
创建或替换表 NEWTABLE
    (Customer_Account int、金额
     int、transaction_ts 时间戳) ;
```

现在，单击左上角的 Home 图标，然后从菜单中选择 Data → Databases。使用右侧的下拉菜单展开 BANKING 架构和 Tables 选项，以查看 DEMO3A_DB 数据库中 BANKING 架构中所有表的列表。在图 3-18 中，我们可以看到 NEWTABLE 是在 active 命名空间中创建的。

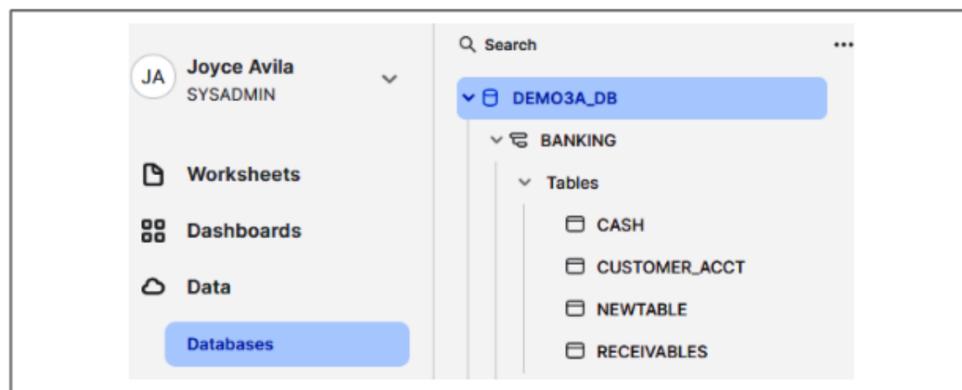


图 3-18. DEMO3A_DB 数据库的 BANKING 架构中的表的列表

现在，让我们删除刚刚使用完全限定的表名创建的新表。导航回 Chapter3 Creating Database Objects 工作表。确保 SYSADMIN 角色是当前角色。在左上角，您将看到您当前位于 Worksheets（工作表）选项卡上。点击 数据库 选项卡（在图 3-19 中圈出）。

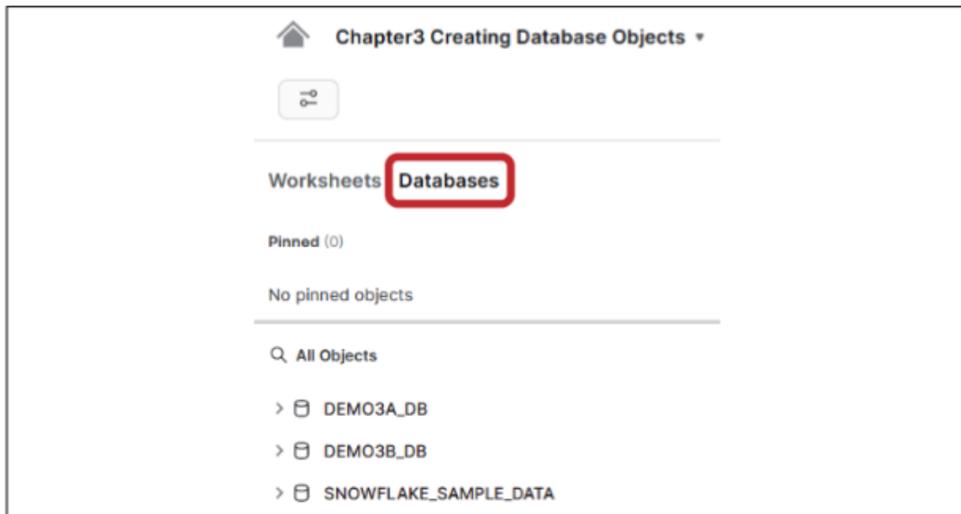


图 3-19. Chapter3 Creating Objects 工作表中的 Databases 选项卡

单击“数据库”选项卡后，可以展开对象，直到能够在 DEMO3A_DB 数据库的 BANKING 架构中看到 NEWTABLE 表（如图 3-20 所示）。

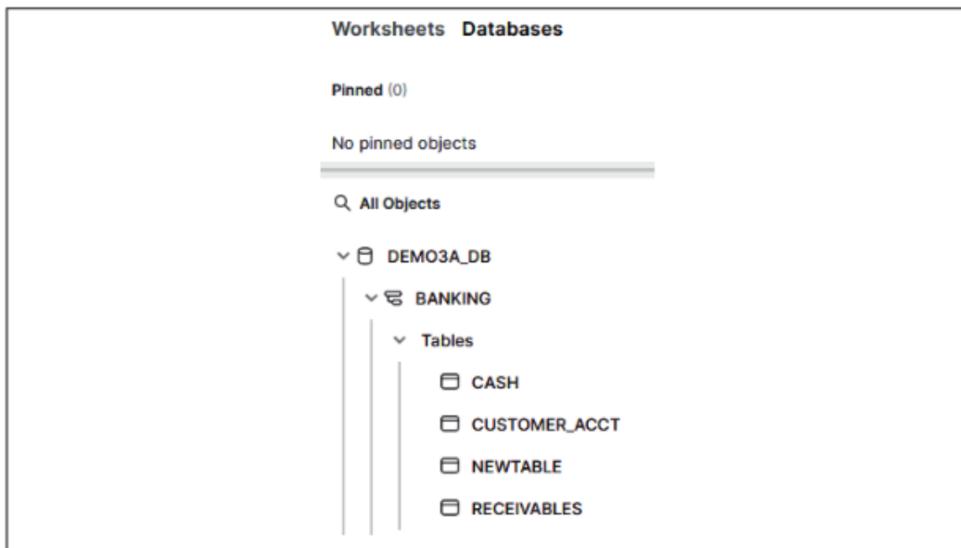
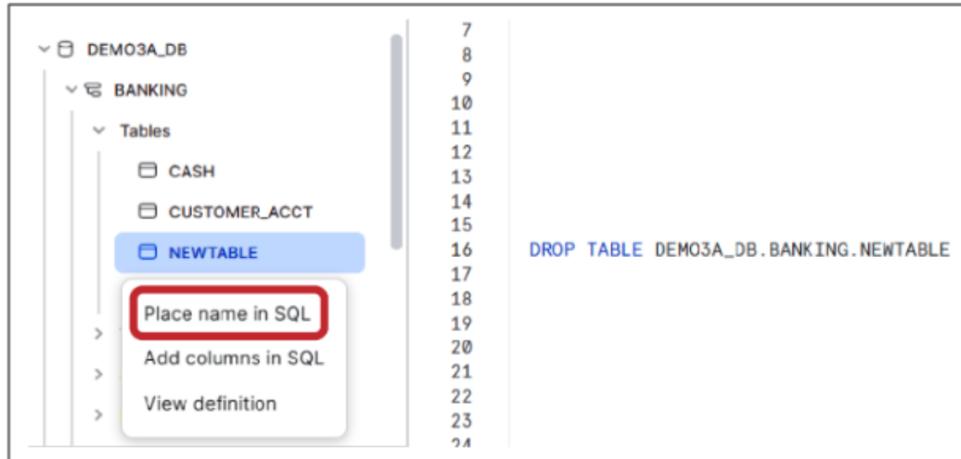


图 3-20. DEMO3A_DB 数据库的 BANKING 架构中的 NEWTABLE 表

在工作表中，输入（并在其后带一个空格）。然后单击左侧的 NEWTABLE 表并选择“Place name in SQL”（SQL 中的地名）选项。您将注意到，表的完全限定名称已添加到 SQL 语句中（如图 3-21 所示）。

如果需要，请在末尾放置一个分号，然后执行该语句。



The screenshot shows the Snowflake interface. On the left, there's a tree view of database objects under 'DEMO3A_DB.BANKING'. The 'NEWTABLE' node is selected and highlighted with a blue background. A context menu is open over it, with the 'Place name in SQL' option highlighted by a red rectangle. Other options in the menu include 'Add columns in SQL', 'View definition', and '>'. To the right of the tree view is a code editor window containing the following SQL code:

```
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
DROP TABLE DEMO3A_DB.BANKING.NEWTABLE
```

图 3-21. 选择完全限定的表名

我们不需要使用完全限定的表名，因为我们位于要删除表的活动空间中。我们本来可以只使用 command DROP TABLE NEWTABLE；而不是使用完全限定名称。但是，最佳实践是使用表的完全限定名称或包含命令，这样可以实现相同的目标。

创建和管理视图

与表一样，Snowflake 视图是数据库架构中维护的主要对象，如图 3-22 所示。视图有两种类型：物化视图和非配对 rialized。每当提到术语 view 但未指定类型时，都可以理解为它是一个非物化视图。



我已经提到了对表使用完全限定名称的重要性。在创建视图时，为表使用完全限定名称更为重要，因为如果未使用基表的命名空间，并且稍后将此表或视图移动到其他架构或数据库，则连接的引用将无效。

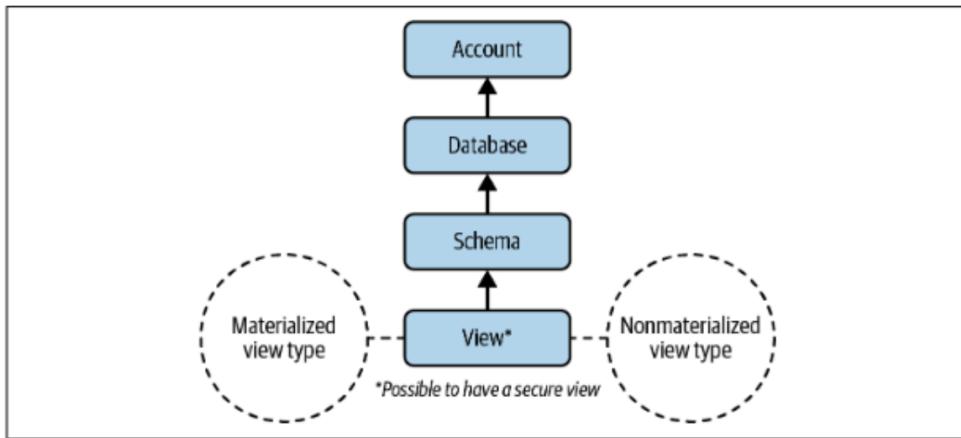


图 3-22. Snowflake 视图的对象层次结构

视图被视为由查询表达式创建的虚拟表；类似于数据库的窗口。让我们通过从 Snowflake 示例数据中选择一列来创建新视图：

```

使用角色 SYSADMIN;
创建或替换视图DEMO3B_DB。公共。新景 AS
SELECT CC_NAME FROM (SELECT * FROM SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL.
CALL_CENTER) ;

```

视图的一个用途是显示一个或多个表中的选定行和列。这是一种通过仅向特定用户公开某些数据来提供一定安全性的方法。视图可以通过创建非具体化视图或具体化视图的特定安全视图来提供更高的安全性。当我们在第 7 章讨论实现账户安全和保护时，我们将更深入地探讨安全视图。我们还将在第 10 章中了解有关安全视图的更多信息。

请务必记住，创建具体化视图需要 Snowflake Enterprise Edition。让我们使用与之前相同的查询创建一个具体化视图：

```

使用角色 SYSADMIN;创建或替换 MATERIALIZED VIEW DEMO3B_DB。公共。NEWVIEW_MVW AS
SELECT CC_NAME FROM (SELECT * FROM SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL.
CALL_CENTER) ;

```

您可以运行命令，结果中将返回两个视图。执行命令时，只会返回物化视图结果。您需要确保从下拉菜单中选择正确的架构，或在语句中包含命令：

```

使用 SCHEMA DEMO3B_DB。公共;显示
视图;

```

如果为每个视图运行命令，您会注意到结果是一致的。那是因为我们还没有真正使用物化视图来实现其预期的目的姿势。与常规视图不同，物化视图对象会定期使用基表中的数据刷新。因此，考虑对 Snowflake 示例数据库使用具体化视图是不合逻辑的，因为 Snowflake 示例数据库无法使用新数据进行更新。

此外，用于检索单个列的查询通常不用于具体化视图。物化视图通常用于聚合和过滤数据，以便将资源密集型操作的结果存储在物化视图中，从而提高数据性能。当经常使用相同的查询时，性能改进尤其好。

Snowflake 使用后台服务自动更新具体化视图。因此，通过具体化视图访问的数据始终是最新的，而不管对基表执行的 DML 量是多少。如果在具体化视图最新之前运行查询，Snowflake 将更新具体化视图或使用具体化视图的最新部分，并从基表中检索较新的数据。

为了实际查看这一点，我们将创建一个具体化视图。我们将在本章后面重新访问具体化视图，以亲自查看它是否包含最新信息：

```
CREATE OR REPLACE MATERIALIZED VIEW DEMO3B_DB.银行业.SUMMARY_MVW AS SELECT * FROM
(SELECT * FROM DEMO3B_DB.银行业.摘要);
```

我们还创建一个非物化视图，以进行比较：

```
CREATE OR REPLACE VIEW DEMO3B_DB.银行业.SUMMARY_VW AS SELECT * FROM
(SELECT * FROM DEMO3B_DB.银行业.摘要);
```

如您所料，视图是只读的。因此，不能使用，

UPDATE 或视图上的命令。此外，Snowflake 不允许用户截断视图。虽然无法在视图上执行 DML 命令，但您可以在 DML 语句中使用子查询来更新基础基表。示例可能如下所示：

```
DELETE FROM WHERE >
(选择平均值);
```

需要注意的另一件事是，视图定义不能使用

ALTER VIEW 命令。但是，该命令可用于重命名具体化视图、添加或替换注释、将视图修改为安全视图等等。和 命令也可用于视图。

如果要更改视图的结构性内容，则必须使用新定义重新创建它。



对源表结构的更改不会自动将门传播到视图。例如，删除表列不会删除视图中的列。

在常规视图和材料化视图之间做出决定时，有许多考虑因素。除了创建常规视图还是材料化视图之外，一个考虑因素是是否使用 ETL 来具体化表中的数据集。

作为一般规则，当视图的结果经常更改并且查询不是那么复杂且重新运行成本不高时，最好使用非具体化视图。常规视图会产生计算成本，但不会产生存储成本。当视图的结果经常变化时，需要权衡刷新视图的计算成本和存储成本与材料化视图的好处。



通常，当查询消耗大量资源时，以及当视图的结果被频繁使用且底层表不经常更改时，使用具体化视图是有益的。此外，如果需要以多种方式对表进行集群化，则可以将材料化视图与不同于基表的集群键的集群键一起使用。

具体化视图存在一些限制。例如，物化视图只能查询单个表，并且不支持联接。建议您查阅 Snowflake 文档，以了解有关具体化视图限制的更多详细信息。



要记住的一点是，我们当前使用的是 SYSADMIN 角色，并且我们正在使用该角色创建视图。没有 SYSDADMIN 角色的用户需要为其分配架构、基础表中的数据库对象以及视图本身（如果他们要使用具体化视图）的权限。

Snowflake 阶段简介：包含文件格式

Snowflake 阶段有两种类型：内部和外部。阶段是指向存储位置的 Snowflake 对象，该存储位置可以是 Snowflake 内部的存储位置，也可以是外部云存储上的存储位置。内部阶段对象可以是命名阶段，也可以是用户或

`table` 舞台。该关键字可用于创建基于会话的命名 `stage` 对象。

在大多数情况下，存储是永久的，而 `stage` 对象（指向存储位置的指针）可能是临时的，也可能随时被丢弃。Snowflake 阶段通常用作将文件加载到 Snowflake 表或将数据从 Snowflake 表卸载到文件中的中间步骤。

Snowflake 永久和内部临时阶段用于在 Snowflake 管理的云存储上存储数据文件，而外部阶段引用存储在 Snowflake 外部位置的数据文件。Snowflake 支持外部位置，无论是私有/受保护还是公共，例如 Amazon S3 存储桶、Google Cloud Storage 存储桶和 Microsoft Azure 容器，都可以在外部阶段使用。

每个 Snowflake 用户都有一个用于存储文件的阶段，该阶段只能由该用户访问。用户阶段可由 `引用`。同样，每个 Snowflake 表都有一个分配用于存储文件的阶段，并且每个表阶段都可以使用 `table>` 引用。



如果多个用户需要访问文件，并且这些文件只需要复制到单个表中，则表阶段非常有用，而当文件只需要由一个用户访问，但需要复制到多个表中时，用户阶段是最佳选择。

用户和表阶段不能更改或删除，并且这两个阶段都不支持设置文件格式。但是，您可以在发出命令时指定 `format` 和 `copy` 选项。此外，表阶段不支持在加载数据时转换数据。表阶段与表本身相关联，而不是单独的数据库对象。要在表阶段执行操作，您必须已被授予表所有权角色。

列出用户阶段的命令是 `OR` 我们将在第 6 章讨论数据加载和卸载时更深入地探讨阶段。

用户阶段和表阶段都是内部阶段的类型，是为每个 Snowflake 帐户自动提供的。除了 `user` 和 `table` 阶段之外，还可以创建内部命名的阶段（参见图 3-23）。内部命名阶段是数据库对象，这意味着它们不仅可以由一个用户使用，而且可以由已被授予具有适当权限的角色的任何用户使用。

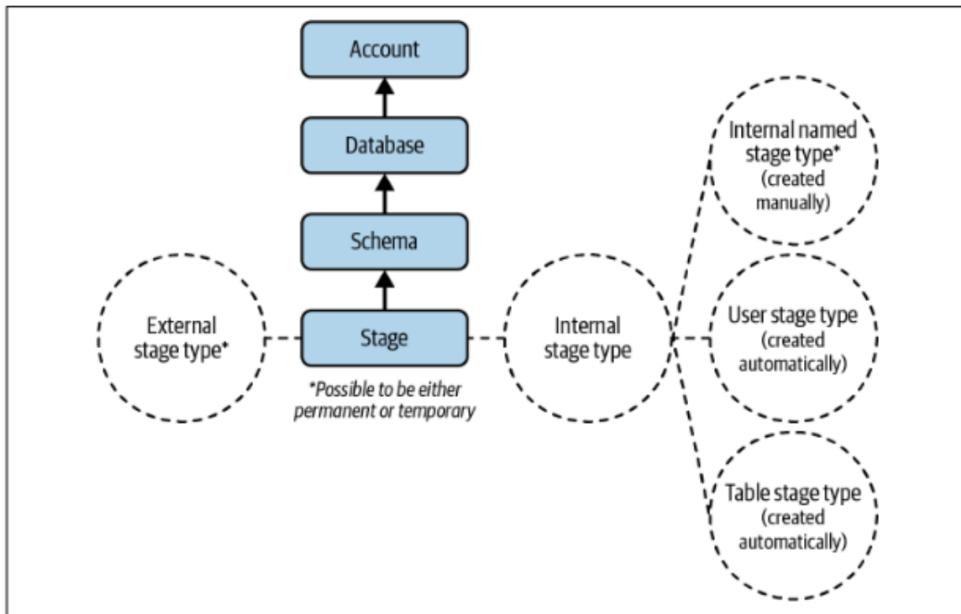


图 3-23. Snowflake 阶段的对象层次结构

内部命名阶段和外部阶段可以创建为永久阶段或临时阶段。删除临时外部舞台时，不会删除任何数据文件，因为这些文件存储在 Snowflake 外部。仅放置 stage 对象。但是，对于临时内部阶段，数据和阶段都会被丢弃，并且文件不可恢复。请务必注意，刚才描述的行为不限于临时舞台对象。临时和永久阶段，内部或外部，都具有相同的特征。

使用阶段时，我们可以使用文件格式来存储将数据从文件加载到表所需的所有格式信息。默认文件格式为 CSV。但是，您可以为其他格式创建文件格式，例如 JSON、Avro、ORC、Parquet 和 XML。在创建文件格式时，还可以包含一些可选参数。在这里，我们将创建一个非常基本的文件格式，没有任何可选的参数，用于将 JSON 数据加载到 stage 中：

```
使用角色 SYSADMIN; 使用数据库DEMO3B_DB; 创建或替换文件格式
FF_JSON TYPE = JSON;
```

创建文件格式后，我们可以在创建内部命名 stage 时使用它：

```
使用数据库DEMO3B_DB; 使用 SCHEMA BANKING; 创建或替换临时舞台 BANKING_STG  
FILE_FORMAT = FF_JSON;
```

我们通过在 Snowflake 工作表中使用 SQL 创建文件格式，在此处简要介绍了文件格式。我们将在第 6 章中详细探讨文件格式，包括使用 Snowsight Web UI 创建文件格式。

表 3-2 总结了不同 Snowflake 阶段的一些特征。

表 3-2. Snowflake 阶段特征

特征	命名阶段	用户阶段	表阶段	外部阶段
Stage 是数据库对象？	是	否	是	默认类型 永久 永久 永久 临时
类型是否可能？	是	否	否	可能？
创建方法	是	否	用户创建	自动
引用@~	如何	引用	@~	
-				
-				
@%<table_name> <stage_name>				
可以丢弃/更改阶段？	是	否	是	数据实际存储在舞台中？
持设置文件格式？	是	否	否	是否 支持
是否可以使用目录表？				
是否 是				
-				
-				
-				
^ 内部舞台				



数据始终处于加密状态，无论是在传输到内部命名阶段的动态中，还是在存储在 Snowflake 数据库表中时处于静态状态。

使用存储过程和 UDF 扩展 SQL

要扩展 Snowflake 中的 SQL 功能，您可以创建存储过程和 UDF 来实现 Snowflake 内置函数无法实现的功能。存储过程和 UDF 都封装并返回单个值（标量）。用户定义的表函数（UDTF）可以返回多个值（表格），而存储过程只能返回一个值。您可以用 JavaScript 或 Snowflake 脚本以本机方式创建 Snowflake 存储过程。此外，通过使用 Snowpark，您可以使用 Java、Python 或 Scala 创建 Snowflake 存储过程。可以使用 SQL、JavaScript、Python 和 Java 语言在本地创建 UDF。可以创建安全的 UDF（请参见图 3-24）。



存储过程的返回值是标量，但如果返回类型是 variant，则过程可以返回多个值。

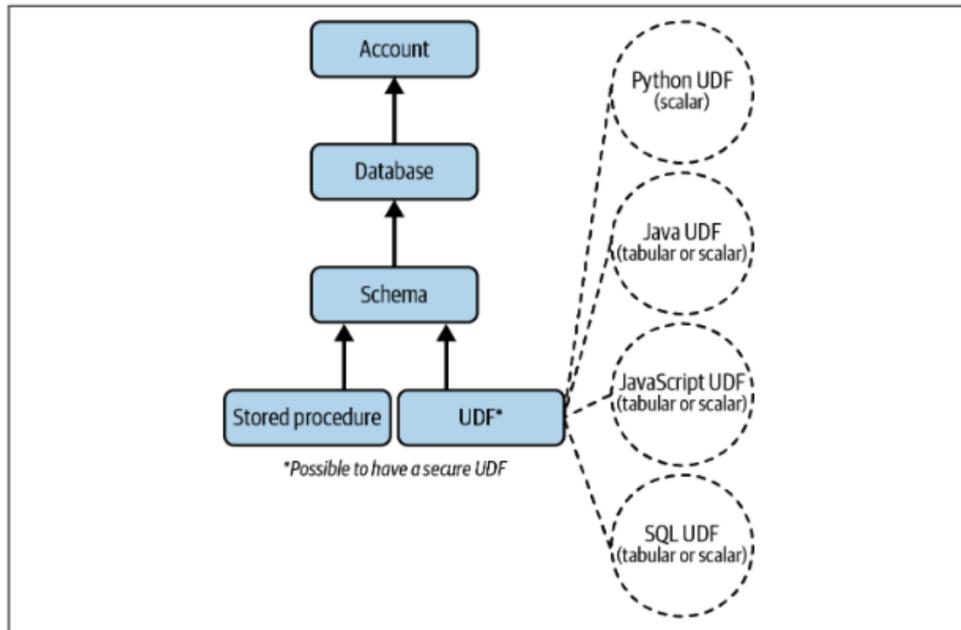


图 3-24. Snowflake 存储过程和 UDF 的对象层次结构

存储过程和 UDF 都扩展了 SQL 功能，但两者之间存在许多差异。最重要的区别之一是它们的使用方式。



如果您需要执行数据库操作，例如，
DELETE 或 ，您需要使用存储过程。如果要将函数用作 SQL 语句或表达式的一部分，或者输出需要包含每个输入行的值，则需要使用 UDF。

UDF 作为 SQL 语句的一部分调用，但不能在 SQL 语句中调用存储过程。相反，使用语句将存储过程称为独立状态。该语句只能为每个语句调用一个存储的进程。

需要 UDF 才能返回值，您可以在 SQL 语句中使用 UDF 返回值。虽然不是必需的，但允许存储过程返回值，但该命令不提供存储返回值的位置。

存储过程也不提供将其传递给其他操作的方法。

接下来的两节总共提供了五个示例。第一个示例是返回标量值的简单 JavaScript UDF，第二个示例是返回表格结果的安全 SQL UDF。接下来是三个存储过程检查：第一个例子是直接传入参数的地方；下一个示例更长，演示了会计信息系统的基础知识，其中每笔银行交易都有一个借方和贷方分录；最后一个示例更高级，它将存储过程与任务组合在一起。

用户定义函数 (UDF)：包含任务

UDF 允许您执行一些通过 Snowflake 提供的内置系统定义函数无法执行的操作。Snowflake 支持 UDF 的四个语言规范。Python UDF 仅限于返回标量结果。SQL、JavaScript 和 Java UDF 可以返回标量或表格结果。

SQL UDF 评估 SQL 语句，并且可以引用其他 UDF，但 SQL UDF 不能直接或间接引用自身。

JavaScript UDF 可用于处理变体和 JSON 数据。JavaScript UDF 表达式可以递归方式引用自身，尽管它不能引用其他 UDF。JavaScript UDF 还具有不适用于 SQL UDF 的大小和深度限制。

我们将再我们的一个示例中看到这一点。

JavaScript UDF 可以访问创建数组、变量和简单对象所需的基本标准 JavaScript 库。您不能使用数学函数或使用 error handling，因为 Snowflake 不允许导入外部库。可以使用以下命令找到可用于 JavaScript UDF 和 JavaScript 过程的属性：

```
使用角色 SYSADMIN;
创建或替换数据库DEMO3C_DB;创建或替换函数
JS_PROPERTIES () 返回字符串语言 JAVASCRIPT 作为
```

```
 $$ return Object.getOwnPropertyNames (this); $$;
```

现在让我们使用命令来查看 JavaScript UDF 属性中包含的内容：

```
选择 JS_PROPERTIES ();
```

上一条命令的结果如图 3-25 所示。

	JS_PROPERTIES()
1	Object,Function,Array,Number,pparseFloat,pparseInt,Infinity,NaN,undefined,Boolean,String,Symbol

图 3-25. 对用户定义的函数使用命令的结果

对于我们的第一个 UDF 示例，我们将创建一个简单的 JavaScript UDF，它返回一个标量结果。我们提到 JavaScript UDF 具有大小和深度限制，我们将能够在此示例中演示。

```
使用角色 SYSADMIN; 使用数据库 DEM03C_DB;
CREATE OR REPLACE FUNCTION FACTORIAL (n variant) 返回
变体 LANGUAGE JAVASCRIPT AS
'var f=n;for (i=n-1; i>0;
i--) {
    f=f*i}
返回 f';
```

当数字 5 用于阶乘函数时，您会期望返回结果 120。立即试用，看看：

选择 FACTORIAL (5)；



如果您使用大于 33 的数字，您将收到一条错误消息。

尝试找到 的结果，看看会发生什么。如果你在阶乘函数中使用数字 50，你将收到一条错误消息，告诉你数值超出范围（如图 3-26 所示）。



图 3-26. 收到错误消息，因为数字 50 大于函数可以使用的值

您会注意到的一件事是，当我们创建 and 函数时，我们没有指定要使用的架构。您认为 Snow - flake 使用什么架构来创建函数？默认情况下，如果您未指定 Schema，则将使用 PUBLIC Schema。您可以展开 DEM03C_DB 数据库的 PUBLIC 结构进行确认（如图 3-27 所示）。

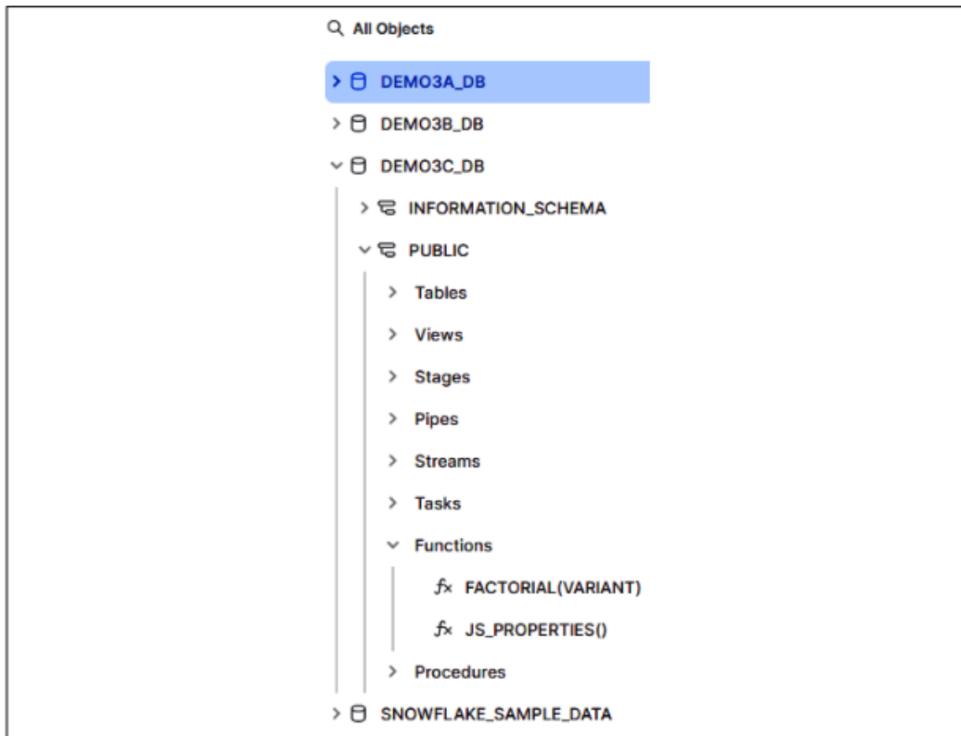


图 3-27. PUBLIC 架构是创建 DEMO3C_DB 时使用的默认架构

我们之前提到过可以创建安全的 UDF。安全 UDF 与不安全 UDF 相同，不同之处在于它们对 UDF 的使用者隐藏了 DDL。由于绕过了一些优化，安全 UDF 确实对性能功能存在限制。因此，数据隐私与性能是一个考虑因素，因为只有安全的 UDF 才能共享。安全 SQL UDF 和安全 Java 脚本 UDF 都是可共享的，但操作方式不同，通常用于不同的目的。安全 JavaScript UDF 通常用于数据清理、地址匹配或其他数据操作。

与安全的 JavaScript UDF 不同，安全的 SQL UDF 可以运行查询。对于安全共享 UDF，只能针对提供程序的数据运行查询。当提供程序与客户共享安全 UDF 时，函数访问的任何表的数据存储费用由提供程序支付，计算费用由使用者支付。可能存在不会产生存储成本的共享函数。

在为您创建的示例中，由于我们使用的是 Snowflake 示例数据集，因此不会产生数据存储成本。

返回表格值的安全 SQL UDTF（购物篮分析示例）

市场篮分析是安全 SQL 用户定义的表函数（UDTF）的常见用途，这就是我们将要演示的内容。在我们的示例中，输出是聚合数据，其中消费者希望查看其他商品与其特定商品一起售出的次数。我们不希望消费者账户访问我们的原始销售数据，因此我们将 SQL 语句包装在安全的 UDF 中并创建一个输入参数。当安全 UDF 与输入参数一起使用时，使用者仍会获得与直接在底层原始数据上运行 SQL 语句相同的结果。

我们希望使用 Snowflake 示例数据库中已提供给我们的数据，但记录数超过了本演示所需的记录数。让我们创建一个新表并从示例数据库中选择 100,000 行，这将绰绰有余：

```
使用角色 SYSADMIN;
创建或替换数据库DEMO3D_DB;创建或替换表DEMO3D_DB。公共。销
售代理
(从 SNOWFLAKE_SAMPLE_DATA中选择 *。TPCDS_SF100TCL。WEB_SALES) 限
100000;
```

接下来，我们将直接在新表上运行查询。在本例中，我们对 SK 为 1 的产品感兴趣。我们想知道在同一笔交易中与此商品一起销售的不同商品。我们可以使用此查询来查找该信息：

```
选择 1 作为 INPUT_ITEM, WS_WEB_SITE_SK 作为 BASKET_ITEM,
对 DEMO3D_DB 中的篮子进行计数（非重复
WS_ORDER_NUMBER）。公共。销售WS_ORDER_NUMBER

(从DEMO3D_DB中选择 WS_ORDER_NUMBER。
公共。销售额 其中 WS_WEB_SITE_SK =
1) 按WS_WEB_SITE_SK顺序按 3 描述分
组, 2;
```



由于几个不同的原因，您的查询结果可能与此处显示的结果略有不同。首先，底层 sample 数据集可能会发生变化。其次，由于我们将之前的结果限制为 10,000 条记录，因此即使底层数据集仍然相同，您的有限数据集也可能有所不同。

此查询的结果显示，SK 为 1 的产品在我们的记录中以 5,050 笔唯一交易售出。SK 为 31 的商品与 SK 为 1 的商品在同一笔交易中售出，共售出四次。带有

SK 为 16、19、40、58、70 和 80 的商品与 SK 为 1 的商品在同一笔交易中售出，每人共售出两次（如图 3-28 所示）。不过，那不是日志。相反，如果有一个 SK 与 1 的 SK 一起大量销售，这对商品制造商来说可能是有价值的信息。

	INPUT_ITEM	BASKET_ITEM	BASKETS
1	1	1	5,050
2	1	31	4
3	1	16	2
4	1	19	2
5	1	40	2
6	1	58	2
7	1	70	2
8	1	80	2

图 3-28。结果显示 SK 为 1 的商品在购物篮中使用了多少次

我们可能愿意与制造商共享这些类型的相关销售详细信息，但我们不希望允许访问基础销售数据。我们可以通过创建一个安全的 SQL UDF 函数来实现我们的目标：

```
使用角色 SYSADMIN;
创建或替换安全功能
DEMO3D_DB。公共。GET_MKTBASKET (INPUT_WEB_SITE_SK number (38) ) 返回
TABLE (INPUT_ITEM NUMBER (38, 0) , BASKET_ITEM NUMBER (38, 0) ,
BASKETS NUMBER (38, 0) ) AS 'SELECT
input_web_site_sk, WS_WEB_SITE_SK as
BASKETITEM;DISTINCT WS_ORDER_NUMBER) 篮子DEMO3D_DB。
公共。销售WS_ORDER_NUMBER

(选择WS_ORDER_NUMBER
从 DEMO3D_DB。公共。销售额 WS_WEB_SITE_SK =
input_web_site_sk) 按ws_web_site_sk顺序分组 3 描述, 2' ;
```

如果我们通过 Snowflake 使用者帐户与其他用户共享此安全 UDF，则该用户可以运行安全 UDF，而无需看到任何底层数据、表结构或 SQL 代码。这是 consumer 的所有者的命令

account 将用于获取结果。你可以用我们之前直接查询表时使用的数字 1 来尝试这个命令，你会得到与 UDF 函数完全相同的结果。您还可以尝试其他产品 SK：

```
选择 * FROM TABLE (DEMO3D_DB. 公共。GET_MKTBASKET (1) ) ;
```



安全 UDF 应用于需要关注数据隐私且您希望限制对敏感数据的访问的情况。重要的是要考虑创建安全 UDF 的目的和必要性，并将其与使用安全 UDF 可能导致的查询性能降低进行权衡。

存储过程

存储过程与函数类似，因为它们只创建一次，并且可以多次执行。存储过程允许您通过将 Snowflake SQL 与 JavaScript 相结合来扩展 Snowflake SQL，以包括分支和循环以及错误处理。存储过程（必须用 JavaScript 编写）可用于自动执行需要频繁执行多个 SQL 语句的任务。虽然存储过程目前只能用 JavaScript 编写，但 SQL 存储过程现在提供个人预览版。

虽然可以在存储过程中使用语句，但必须在存储过程中使用结果。否则，只能返回单个值结果。存储过程非常适合批处理操作，因为存储过程自行运行，并且与触发器类似，可以有条件地绑定到数据库事件。

在第一个存储过程示例中，我们将向存储过程传递一个参数：

```
使用角色 SYSADMIN;
创建或替换数据库DEMO3E_DB;
创建或替换过程 STOREDPROC1 (ARGUMENT1 VARCHAR)
返回 string not null language
javascript AS $$ var INPUT_ARGUMENT1 =
ARGUMENT1;var result =
'$ {INPUT_ARGUMENT1}' 返回结果;$$;
```

现在我们调用存储过程：

```
CALL STOREDPROC1 ('我真的很喜欢 Snowflake *') ;
```



JavaScript 区分大小写，因此请务必在创建存储过程时密切注意。

图 3-29 显示了将参数传递给存储过程的结果。

STOREDPROC1	
1	I really love Snowflake *

图 3-29。将参数传递给存储过程的结果

您可以使用INFORMATION_SCHEMA 查看数据库中 Snowflake 过程的信息：

从 DEMO3E_DB中选择 *。INFORMATION_SCHEMA。程序；

在下一个存储过程示例中，我们将演示会计信息系统的基础知识，其中为记录借方和贷方的银行交易返回表格结果。如果您还记得，在本章的前面，我们创建了此示例所需的表，因此我们可以直接深入研究创建存储过程。

我们正在记录一家银行的会计交易，该银行与客户向银行提供现金并从银行获取现金打交道。我们将示例的会计部分限制为三种类型的交易：存款、取款和贷款支付。

首先，我们要为存款交易创建一个存储过程，其中银行的现金账户被借记，每当客户向银行提供现金以添加到他们的账户时，银行资产负债表上的客户账户就会被贷记：

```
使用角色 SYSADMIN; 使用数据库DEMO3A_DB; 使用 SCHEMA BANKING;CREATE OR REPLACE
PROCEDURE deposit (PARAM_ACCT FLOAT,  PARAM_AMT FLOAT) 返回 STRING LANGUAGE
javascript AS
$$ var ret_val = “” ;var cmd_debit = “” ;var cmd_credit = “” ;
将数据 INSERT 到表中
cmd_debit = “插入DEMO3A_DB_银行业。现金价值 (”
+ PARAM_ACCT + “, ” + PARAM_AMT + “, current_timestamp () ) ;” ;
cmd_credit = “插入DEMO3A_DB_银行业。CUSTOMER_ACCT值 (”
+ PARAM_ACCT + “, ” + PARAM_AMT + “, current_timestamp
() ) ;” ;BEGIN 事务 snowflake.execute ({sqlText:
cmd_debit}) ;snowflake.execute ({sqlText: cmd_credit}) ;

ret_val = “存款交易成功” ;返回ret_val $$;
```

接下来，我们将创建提款交易。在这种情况下，它与存款的情况正好相反：

```
使用角色 SYSADMIN;使用数据库DEMO3A_DB;使用 SCHEMA BANKING;CREATE OR REPLACE
PROCEDURE withdrawal (PARAM_ACCT FLOAT, PARAM_AMT FLOAT) 返回 STRING LANGUAGE
javascrip AS
$$ var ret_val = “” ;var cmd_debit = “” ;var cmd_credit = “” ;
将数据 INSERT 到表中
cmd_debit = “插入DEMO3A_DB。银行业。CUSTOMER_ACCT值（”
+ PARAM_ACCT + “, ” + (-PARAM_AMT) + “, current_timestamp
() ) ;” ;cmd_credit = “插入DEMO3A_DB。银行业。现金价值（”
+ PARAM_ACCT + “, ” + (-PARAM_AMT) + “, current_timestamp() ) ;” ;BEGIN
事务 snowflake.execute ({sqlText: cmd_debit}) ;snowflake.execute
({sqlText: cmd_credit}) ;

ret_val = “提款交易成功” ;返回ret_val $$;
```

最后，贷款支付的交易是借记银行的现金账户并贷记应收账款账户的交易：

```
使用角色 SYSADMIN;使用数据库DEMO3A_DB;使用 SCHEMA BANKING;创建或替换过程 loan_payment
(PARAM_ACCT FLOAT, PARAM_AMT FLOAT) 返回字符串语言 JAVASCRIPT AS
$$ var ret_val = “” ;var cmd_debit = “” ;var cmd_credit = “” ;
将数据 INSERT 到表中
cmd_debit = “插入DEMO3A_DB。银行业。现金价值（”
+ PARAM_ACCT + “, ” + PARAM_AMT + “, current_timestamp
() ) ;” ;cmd_credit = “插入DEMO3A_DB。银行业。应收账款值（”
+ PARAM_ACCT + “, ” + (-PARAM_AMT) + “, current_timestamp() ) ;” ;BEGIN
事务 snowflake.execute ({sqlText: cmd_debit}) ;snowflake.execute
({sqlText: cmd_credit}) ;

ret_val = “贷款支付交易成功” ;返回ret_val $$;
```

现在让我们运行一些快速测试，看看这些过程是否有效。分别运行每个语句：

```
CALL 取款 (21, 100) ;致电
loan_payment (21, 100) ;CALL 存
款 (21, 100) .
```

我们可以通过执行 execute 来查看调用过程时发生的情况的详细信息：

```
选择 CUSTOMER_ACCOUNT、金额 DEMO3A_DB。银行业。现金；
```

在使用命令后跟命令测试一些反式操作后，我们确信该程序正在按预期工作。

所以，现在我们可以截断表，保持表不变，但删除数据：

```
使用角色 SYSADMIN; 使用数据库DEMO3A_DB; 使用 SCHEMA BANKING; 截断表  
DEMO3A_DB。银行业。CUSTOMER_ACCT; 截断表 DEMO3A_DB。银行业。现金; 截断表  
DEMO3A_DB。银行业。应收帐款;
```

请注意，如果我们现在重新运行前面的语句，我们将看到表中没有数据。我们准备通过调用过程并提供输入来输入一些交易：

```
使用角色 SYSADMIN; 看涨保证金  
(21, 10000) ;CALL 存款 (21,  
400) ;致电 loan_payment (14,  
1000) ;CALL 取款 (21, 500) ;看涨  
保证金 (72, 4000) ;CALL 取款  
(21, 250) ;
```

我们希望看到事务摘要，因此我们将创建一个最终的存储过程：

```
使用角色 SYSADMIN; 使用数据库DEMO3B_DB; 使用 SCHEMA BANKING;  
CREATE OR REPLACE PROCEDURE Transactions_Summary () 返回  
字符串语言 JAVASCIPT AS  
$$ var cmd_truncate = '如果存在，则截断表 DEMO3B_DB。银行业。摘要'; var sql =  
snowflake.createStatement ({sqlText: cmd_truncate}) ;Summarize Cash Amount var  
cmd_cash = '插入DEMO3B_DB。银行业。SUMMARY (CASH_AMT) 从 DEMO3A_DB中选择 sum  
(AMOUNT)。银行业。' var sql = snowflake.createStatement ({sqlText: cmd_cash}) ;汇  
总应收帐款金额 var cmd_receivables = '插入DEMO3B_DB。银行业。摘要  
(RECEIVABLES_AMT) 从 DEMO3A_DB中选择 sum (AMOUNT)。银行业。' var sql =  
snowflake.createStatement ({sqlText: cmd_receivables}) ;汇总客户帐户金额 var  
cmd_customer = '插入DEMO3B_DB。银行业。SUMMARY (CUSTOMER_AMT) 从 DEMO3A_DB中选择  
sum (AMOUNT)。银行业。CUSTOMER_ACCT;' var sql = snowflake.createStatement  
( {sqlText: cmd_customer} ) ;BEGIN 事务 snowflake.execute ({sqlText:  
cmd_truncate}) ;snowflake.execute ({sqlText: cmd_cash}) ;snowflake.execute  
( {sqlText: cmd_receivables}) ;snowflake.execute ({sqlText:  
cmd_customer}) ;ret_val = "成功汇总的事务";返回ret_val; $$;
```

现在我们可以看到交易摘要。我们将调用存储过程，以便汇总所有借方和贷方交易，然后我们看一下表：

调用 Transactions_Summary () ;

从 DEMO3B_DB 中选择 *。银行业。总结；

我们还看一下我们之前创建的具体化视图：

使用角色 SYSADMIN; 使用数据库 DEMO3B_DB; 使用 SCHEMA BANKING; 从 DEMO3B_DB 中选择 *。银行业。SUMMARY_MVW;

有趣的是，物化视图一直使用 SUMMARY 基表中的信息进行更新（如图 3-30 所示）。

	CASH_AMT	RECEIVABLES_AMT	CUSTOMER_AMT
1	14,650	null	null
2	null	-1,000	null
3	null	null	13,650

图 3-30. 具体化视图的结果

为了进行比较，让我们也看一下非物化视图：

使用角色 SYSADMIN; 使用数据库 DEMO3B_DB; 使用 SCHEMA BANKING; 从 DEMO3B_DB 中选择 *。银行业。SUMMARY_VW;

现在我们有了最后一个存储过程示例。这是一个高级示例，因为我们要添加一个任务来执行这个存储过程。

顺便说一句，任务非常强大，是一个重要的话题。在本章后面，我们将更详细地介绍任务，届时我们将了解如何将任务与表流组合在一起。现在，了解任务可以调用存储过程（如本例中所示）或者它可以执行单个 SQL 语句就足够了。

我们将创建一个将删除数据库的存储过程。因此，我们必须在不同的数据库中创建存储过程，而不是使用存储过程删除的数据库：

```
使用角色 SYSADMIN; 使用数据库 DEMO3E_DB;
CREATE OR REPLACE PROCEDURE drop_db () 返回 STRING
NOT NULL LANGUAGE JAVASCRIPT AS
$$ var cmd = 'DROP DATABASE DEMO3A_DB;' var sql =
snowflake.createStatement ({sqlText: cmd}) ;var 结果 =
sql.execute () ;
```

```
返回 'Database has been successfully dropped' ;$$;
```

现在我们可以调用存储过程，数据库将被删除。这是我们本章清理的一部分：

调用 drop_db ()；

刷新工作表，您将看到 DEMO3A_DB 数据库已被删除。现在，您已经了解了此存储过程的工作原理，我们将对其进行修改，以便它删除不同的数据库，并添加一个任务，以便在 15 分钟后删除该数据库。这样，您就可以了解任务的工作原理。

让我们继续更改我们的过程，以便我们将删除 DEMO3B_DB 数据库。请注意，我们需要替换该过程，因为无法更改存储过程的内容。但是，可以使用该命令重命名过程、设置和取消设置组件，或更改是否使用所有者或调用者的权限执行过程：

```
使用角色 SYSADMIN;
创建或替换过程 drop_db () 返回 STRING NOT NULL
    语言 JAVASCRIPT AS
    $$ var cmd = 'DROP DATABASE "DEMO3B_DB";' var sql =
snowflake.createStatement ({sqlText: cmd});var 结果 =
sql.execute ();返回 'Database has been successfully
dropped';$$;
```

接下来，我们要创建将存储过程延迟 15 分钟的任务：

```
使用角色 SYSADMIN; 使用数据库DEMO3E_DB; 创建或替换任务
tsk_wait_15 WAREHOUSE = COMPUTE_WH SCHEDULE = '15
MINUTE' AS CALL drop_db () ;
```

SYSADMIN 角色需要一些权限才能执行任务，因此请务必为此命令使用 ACCOUNTADMIN 角色。即使 SYSADMIN 角色是任务所有者，也需要提升的账户级权限才能执行任务。确保将角色设置为 ACCOUNTADMIN：

```
使用角色 ACCOUNTADMIN;
将 EXECUTE TASK ON ACCOUNT 授予角色 SYSADMIN;
```

您现在可以将角色设置为 SYSADMIN。由于任务始终以悬而未决的状态创建，因此需要恢复它们：

```
使用角色 SYSADMIN; 如果存在，则更改任务
tsk_wait_15 RESUME;
```

现在，我们的任务处于 scheduled 状态。我们将能够通过对 INFORMATION_SCHEMA 使用此查询来看到这一点。TASK_HISTORY table 函数：

```
SELECT * FROM table (information_schema.task_history
  (task_name => 'tsk_wait_15',
  scheduled_time_range_start => dateadd ('hour',
  -1, current_timestamp ( ) ) ) ) .
```

图 3-31 显示了查询结果。

	SCHEDULED_TIME	QUERY_START_TIME	...	NEXT_SCHEDULED_TIME
1	2022-05-30 17:07:16.846 -0700	null		2022-05-30 17:22:16.846 -0700

图 3-31. 显示任务处于计划状态的结果

您可以利用接下来的 15 分钟回答本章末尾的问题以测试您的知识，然后再回来。看到任务已完成且数据库已删除（您可以刷新屏幕，也可以再次运行查询以查看任务的状态）后，您可以暂停任务：

```
使用角色 SYSADMIN;如果存在，则更改任务
tsk_15 SUSPEND;
```

暂停任务后，您可以最后一次运行查询，您将看到任务成功，并且没有计划任何任务。

管道、流和序列简介

管道、流和序列是我们尚未介绍的 Snowflake 对象。管道是包含 Snowpipe 使用的命令的对象。Snowpipe 用于将数据持续、无服务器地加载到 Snowflake 目标表中。Snowflake 流也称为变更数据捕获（CDC），用于跟踪对表所做的某些更改，包括插入、更新和删除。Snowflake 流有许多有用的用途，包括记录在临时表中所做的更改，这些更改用于更新另一个表。

序列对象用于生成唯一编号。通常，序列用作主键值的代理键。以下是生成以数字 1 开头并以 1 递增的序列的方法：

```
使用角色 SYSADMIN;使用数据库DEMO3E_DB;
创建或替换序列 SEQ_01 START = 1 增量 = 1;
创建或替换表 SEQUENCE_TEST (i integer);
```

您可以使用该命令 3 次或 4 次来查看增加情况。每次执行该语句时，您都会注意到增加 1：

```
选择 SEQ_01.NEXTVAL;
```

现在让我们创建一个具有不同增量的新序列：

```
使用角色 SYSADMIN; 使用数据库DEMO3E_DB;  
创建或替换序列 SEQ_02 START = 1 增量 = 2;  
创建或替换表 SEQUENCE_TEST (i integer) ;
```

看看当你尝试这个时会发生什么：

```
选择 SEQ_02.NEXTVAL a, SEQ_02.NEXTVAL b, SEQ_02.NEXTVAL c, SEQ_02.NEXTVAL d;
```

您应该看到 A 的值为 1, B 的值为 3, C 的值为 5, D 的值为 7。

需要记住的一些重要事项是，在创建序列后，无法更改序列中的第一个值，并且序列值虽然唯一，但不一定没有间隙。这是因为序列是用户定义的数据库对象；因此，序列值可以由多个表共享，因为序列不绑定到任何特定表。创建序列的另一种方法是使用标识 col - umns，您将在一个特定表中生成自动递增的数字，通常用作主键。



使用序列时的一个注意事项是，它们可能不适用于安全 UDF 情况等情况。在某些情况下，使用者可能能够使用序列号之间的差异来推断有关记录数的信息。在这种情况下，您可以从使用者结果中排除序列列，或使用唯一的字符串 ID 而不是序列。

我们将在第 6 章中探讨 Snowpipe。下一节将深入讨论流，我们将发现如何将它们用于 CDC。我们还将了解如何利用流和任务的强大功能。

雪花溪流（深入探讨）

Snowflake 流的工作原理类似于指针，用于跟踪已定义表上的 DML 操作的状态。Snowflake 表流创建一个更改表，该表显示两个事务时间点之间在行级别发生的变化。流类似于处理队列，可以像表一样进行查询。Table Streams 使获取表中的新数据变得容易，以便进行更高效的处理。Streams 通过在某个时间点拍摄表中所有行的快照，并且仅存储源表的偏移量来实现这一点。这样，流就可以利用版本控制历史记录返回 CDC 记录。

为了深入了解 Snowflake 流，我们来看一个示例。我们将为银行分行创建一个表，用于存储特定日期的分行 ID、城市和关联的美元存款金额。

为了准备该示例，我们将创建一个新的数据库、架构和表。我们将插入一些值以开始使用。我们将在示例中使用该命令，但通常您不会使用 insert 语句将数据加载到 Snowflake 实例中。相反，您可能会使用类似 Snowpipe 的工具。为简单起见，我们一次只插入几条记录，因此我们将使用该命令。以下是为 Snowflake 流示例进行设置的语句：

```
使用角色 SYSADMIN;
创建或替换数据库DEMO3F_DB;CREATE OR REPLACE TABLE BRANCH (ID
varchar, City varchar, 金额数 (20,2) );INSERT INTO BRANCH (ID, City, Amount) 值
```

```
(12001, '阿比林', 5387.97),
(12002, '巴斯托', 34478.10),
(12003, '吉百利', 8994.63).
```

在继续之前，您可以使用 以下语句暂停并查看表中的记录：

```
SELECT * FROM BRANCH;
```

我们将创建两个流，然后使用命令查看创建的流：

```
在 TABLE BRANCH 上创建或替换 STREAM STREAM_A;在 TABLE 分支
上创建或替换流STREAM_B;显示流;
```

如果您在这些流上运行语句，您将看到两者都是空的。如果我们在表中插入一些新记录会发生什么情况？我们预计 Stream 将具有记录：

```
INSERT INTO BRANCH (ID, City, 金额)
值
(12004, '丹顿', 41242.93),
(12005, '埃弗雷特',
6175.22), (12006, '法戈',
```

现在，对 BRANCH 表和两个流执行每个语句，一次一个：

```
SELECT * FROM BRANCH;SELECT
* FROM STREAM_A;SELECT *
FROM STREAM_B;
```

您应该看到 BRANCH 表中现在有 6 条记录，流中各有 3 条记录。如果我们添加另一个流 STREAM_C，我们将预期该流中不会有任何记录：

在 TABLE BRANCH 上创建或替换流STREAM_C；

现在让我们继续第三次插入一些记录：

```
INSERT INTO BRANCH (ID, City, 金额)
值
(12007, '加尔维斯顿',
351247.79), (12008, '休斯顿',
917011.27);
```

如果对表和 3 个流运行语句，您将看到该表有 8 条记录，STREAM_A 和 STREAM_B 各有 5 条记录，STREAM_C 有 2 条记录。

让我们再做一件事。让我们重新创建 STREAM_B 以查看下一部分会发生什么：

在 TABLE 分支上创建或替换流STREAM_B；

此时，STREAM_B 将有零条记录。

要查看删除记录的影响，让我们继续从前面的每个插入中删除第一条记录：

```
从 ID = 12001 的分支中删除;从 ID = 12004
的分支中删除;从 ID = 12007 的分支中删除;
```

如果我们先在表上运行，然后在所有三个流上运行，我们应该在 BRANCH 表中看到 5 条记录（如图 3-32 所示），在 STREAM_A 表中看到 4 条记录，在 STREAM_B 表中看到 3 条记录，在 STREAM_C 表中看到 3 条记录。

	ID	CITY	AMOUNT
1	12002	Barstow	34,478.1
2	12003	Cadbury	8,994.63
3	12005	Everett	6,175.22
4	12006	Fargo	443,689.75
5	12008	Houston	917,011.27

图 3-32。对 BRANCH 表的查询结果

创建 STREAM_A 后，输入了三条记录：12004、12005 和 12006。然后又输入了两条记录：12007 和 12008。删除记录 12004 和 12007 时，它们将从 STREAM_A 中删除。删除记录 12001 时，记录 12001 在 STREAM_A 中显示为新条目，因为记录 12001 没有

以前存在。现在，STREAM_A 中总共应该有 4 条记录（如图 3-33 所示）。

	ID	CITY	AMOUNT	METADATA\$ACTION	METADATA\$ISUPDATE
1	12005	Everett	6,175.22	INSERT	FALSE
2	12006	Fargo	443,689.75	INSERT	FALSE
3	12008	Houston	917,011.27	INSERT	FALSE
4	12001	Abilene	5,387.97	DELETE	FALSE

图 3-33. STREAM_A 上的查询结果

提醒一下，STREAM_B 已重新创建。从重新创建STREAM_B 开始，没有插入任何新记录。当记录 12001、12004 和 12007 被删除时，它们都出现在 STREAM_B 中，因为它们之前没有出现在那里。因此，现在 STREAM_B 中应该有 3 条记录（如图 3-34 所示）。

	ID	CITY	AMOUNT	METADATA\$ACTION	METADATA\$ISUPDATE
1	12004	Denton	41,242.93	DELETE	FALSE
2	12007	Galveston	351,247.79	DELETE	FALSE
3	12001	Abilene	5,387.97	DELETE	FALSE

图 3-34。STREAM_B 上的查询结果

创建 STREAM_C 后，输入了两条记录：12007 和 12008。删除记录 12007 时，该记录将从 STREAM_C 中删除。删除记录 12001 和 12004 时，它们在 STREAM_C 中显示为新条目，因为这些记录以前不存在。现在，STREAM_C 中总共应该有 3 条记录（如图 3-35 所示）。

	ID	CITY	AMOUNT	METADATA\$ACTION	METADATA\$ISUPDATE
1	12008	Houston	917,011.27	INSERT	FALSE
2	12004	Denton	41,242.93	DELETE	FALSE
3	12001	Abilene	5,387.97	DELETE	FALSE

图 3-35. STREAM_C 上的查询结果

我们的示例演示了插入和删除记录的影响，我们可以在 METADATA \$ACTION 列中看到结果。但是，当我们更新记录时会发生什么？让我们将城市更新为 FAYETTEVILLE，其中 BRANCH ID 等于 12006：

```
更新分支  
SET City = '费耶特维尔' WHERE ID = 12006;  
SELECT * FROM BRANCH;
```

正如预期的那样，BRANCH 表显示了更新后的城市（如图 3-36 所示）。

	ID	CITY	AMOUNT
1	12002	Barstow	34,478.1
2	12003	Cadbury	8,994.63
3	12005	Everett	6,175.22
4	12006	Fayetteville	443,689.75
5	12008	Houston	917,011.27

图 3-36。城市设置为 Fayetteville 的 BRANCH 表上的查询结果

记录 12006 在 STREAM_A 中已经存在，因此不需要新的条目。
该值只是在流中更新（如图 3-37 所示）。

	ID	...	CITY	AMOUNT	METADATA\$ACTION	METADATA\$ISUPDATE
1	12005		Everett	6,175.22	INSERT	FALSE
2	12006		Fayetteville	443,689.75	INSERT	FALSE
3	12008		Houston	917,011.27	INSERT	FALSE
4	12001		Abilene	5,387.97	DELETE	FALSE

图 3-37. STREAM_A 上的查询结果

记录 12006 以前在 STREAM_B 中不存在。因此，我们看到有一个条目用于删除法戈市的记录 12006，然后有一个条目用于插入值为 Fayetteville 的新 12006 记录。您可以看到，这两个新条目在 METADATA \$ISUPDATE 列中显示为具有 TRUE 值。添加这两个新条目后，STREAM_B 现在有 5 条记录（如图 3-38 所示）。

	ID	CITY	AMOUNT	METADATA\$ACTION	METADATA\$ISUPDATE
1	12006	Fayetteville	443,689.75	INSERT	TRUE
2	12006	Fargo	443,689.75	DELETE	TRUE
3	12004	Denton	41,242.93	DELETE	FALSE
4	12007	Galveston	351,247.79	DELETE	FALSE
5	12001	Abilene	5,387.97	DELETE	FALSE

图 3-38。STREAM_B 上的查询结果

记录 12006 以前在 STREAM_C 中不存在。因此，我们看到有一个条目用于删除法戈市的记录 12006，然后有一个条目用于插入值为 Fayetteville 的新 12006 记录。您可以看到，这两个条目在 METADATA \$ISUPDATE 列中显示为具有 TRUE 值。添加这两个新条目后，STREAM_C 现在有 5 条记录（如图 3-39 所示）。有趣的是，您会注意到 STREAM_B 和 STREAM_C 此时都有 5 条记录，但它们没有相同的 5 个条目。

	ID	CITY	AMOUNT	METADATA\$ACTION	METADATA\$ISUPDATE
1	12006	Fayetteville	443,689.75	INSERT	TRUE
2	12008	Houston	917,011.27	INSERT	FALSE
3	12006	Fargo	443,689.75	DELETE	TRUE
4	12004	Denton	41,242.93	DELETE	FALSE
5	12001	Abilene	5,387.97	DELETE	FALSE

图 3-39. STREAM_C 上的查询结果

为了总结元数据，让我们回顾一元数据列。META - DATA\$ACTION 列告诉我们该行是插入还是删除的。如果该行已更新，则 METADATA \$ISUPDATE 列将为 TRUE。最后，METADATA \$ROW_ID 列有一个唯一的哈希键。

正如您在提供的示例中看到的那样，Snowflake 流是处理不断变化的数据集的强大方法。在 Snowflake 中，使用表流的最重要原因之一是保持暂存表和生产表同步。将暂存表与流一起使用有助于防止对 production 表进行不需要的更改。Snowflake 表流还经常与其他功能结合使用，例如 Snowflake 管道或 Snowflake 任务。

Snowflake 任务（深入探讨）

Snowflake 任务经常与 Snowflake 流一起使用，如下例所示。它们在不需要使用流的方案中也很有用。您可能还记得，Snowflake 任务在本章的前面部分介绍过，当时我们创建了一个任务，该任务调用存储过程来删除数据库，作为我们清理工作的一部分。

在本节中，我们将演示如何将 Snowflake 任务与流一起使用。在处理一些动手示例之前，我们先回顾一下有关 Snowflake 任务的一些重要细节。

完成 Snowflake 任务的一种方法是使用由用户管理的计算资源。我们之前使用了这种方法，我们指定了一个虚拟仓库供任务使用。或者，任务的完成情况可以由 Snowflake 通过无服务器计算模型进行管理。在无服务器计算模型中，随着工作负载要求的变化，Snowflake 会自动调整资源的大小并扩大或缩小。



在创建任务时，必须指定启用无服务器计算模型的选项。

任务按创建任务时定义的计划运行。或者，您可以建立任务依赖关系，以便任务可以由前置任务触发。没有可以触发任务的事件源；不是由前置任务触发的任务必须按计划运行。

如图 3-40 所示（包括每种任务类型的语句），您可以由前置任务计划或触发任务。您会注意到，我们选择包括一个小型虚拟仓库作为所有任务的初始仓库大小，但我们可以为不同的任务选择不同的大小。

每当存在任务依赖关系时，就会有一个根任务在计划上运行。然后，完成的每个任务都会触发关联的子任务。每个子任务都由一个前置任务触发。每个单独的任务仅限于一个 `precedes - sor` 任务，但最多可以有 100 个子任务。任务树总共限制为 1000 个任务，包括根任务。

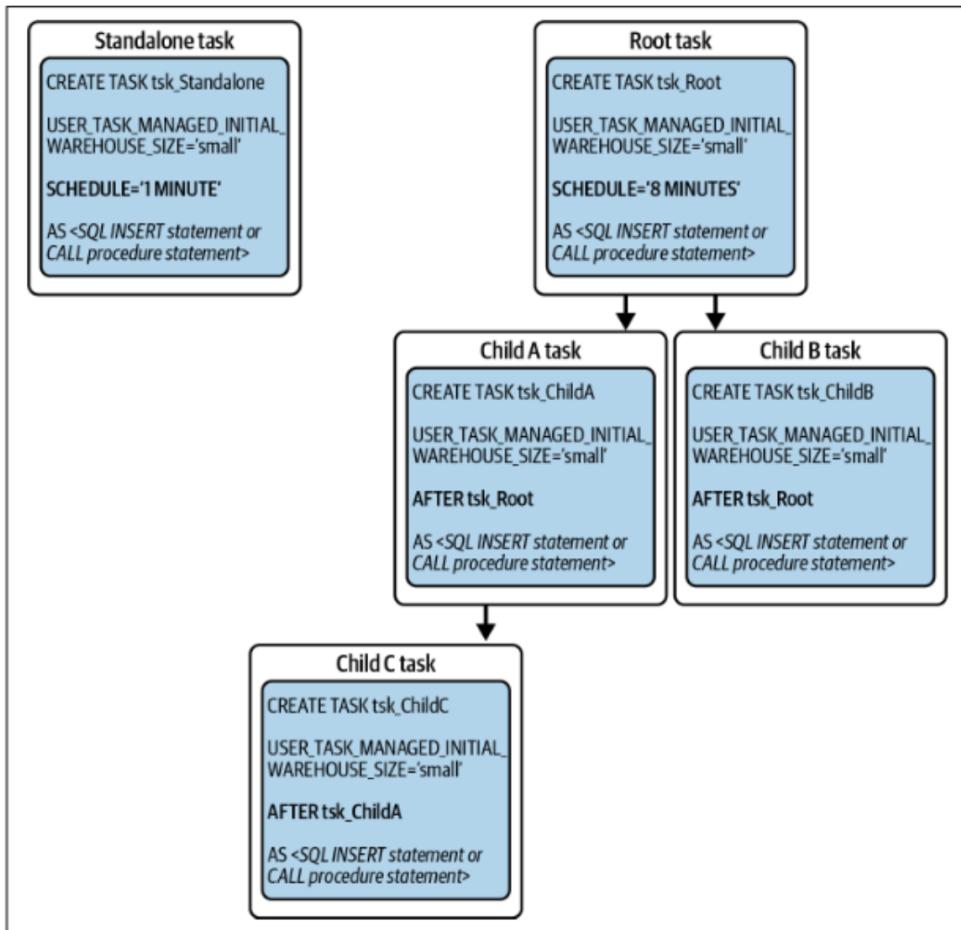


图 3-40. 用于创建独立任务、根任务和子任务的任务示例



您可以在任务定义中使用 cron 表达式，而不是定义分钟数，以支持在计划任务时指定时区。为避免由于夏令时而意外执行任务，请考虑将任务安排在每天凌晨 1 点到 3 点以外的时间，或者安排在一周中包括星期日的日期。

该命令会手动触发计划任务的单次运行，该任务可以是独立任务，也可以是任务树中的根任务。如果需要递归恢复与根任务关联的所有依赖任务，请务必使用 SYSTEM\$TASK_DEPENDENTS_ENABLE 函数，而不是使用 ALTER TASK RESUME 命令单独恢复每个任务。



默认情况下，Snowflake 确保一次只允许运行特定任务树的一个实例。因此，如果运行任务树中所有任务所需的累积时间超过根任务定义中设置的显式计划时间，则至少将跳过任务树的一次运行。当任务树的重叠运行执行的读/写 SQL 操作不会产生不正确或重复的数据时，重叠运行可能不会有问题。在这种情况下，如果这样做有意义，可以通过将根任务中的 `ALLOW_OVERLAPPING_EXECUTION` 参数设置为 来更改默认行为。

如果任务树的重叠运行会带来问题，则有多种选择。您可以选择更合适的仓库大小或使用 Snowflake 托管计算资源，以便任务树在根任务的下一次计划运行之前完成。或者，您可以分析每个任务执行的 SQL 语句或存储过程，以确定是否可以优化代码。另一种选择是，您可以增加根任务运行之间的计划时间。

三个 Snowflake 任务函数可用于检索有关任务的信息：

`系统$CURRENT_USER_TASK_NAME`

此函数返回当前正在执行的任务的名称，当从任务定义的语句或存储过程中调用时。

`TASK_HISTORY`

此函数返回过去 7 天内的任务活动或未来 8 天内的下一次计划执行。

`TASK_DEPENDENTS`

这是一个表函数，用于返回给定根任务的子任务列表。

由于创建和管理任务需要许多访问控制权限，因此建议创建自定义 `TASKADMIN` 角色，以便角色可以更改自己的任务。拥有自定义 `TASKADMIN` 角色还允许 `SECURITYADMIN` 角色或任何具有权限的角色来管理 `TASKADMIN` 角色。否则，只有 `ACCOUNTADMIN` 可以撤销分配给任务所有者的任何任务权限。

以下示例演示如何创建 `TASKADMIN` 角色：

```
使用角色 SECURITYADMIN;  
创建角色 TASKADMIN;
```

```
使用角色 ACCOUNTADMIN;授予 EXECUTE TASK、EXECUTE MANAGED TASK ON ACCOUNT 角色  
TASKADMIN;
```

```
使用角色 SECURITYADMIN;
将角色 TASKADMIN 授予角色 ;
将 TASK ADMIN 角色授予 SYSADMIN 角色;
```

在实践中，您需要确保向其分配 TASKADMIN 角色的USER_ROLE 具有对数据库、架构和表的所有必要访问权限。在本章中，我们不会执行所有这些语句。有关创建角色、授予对象访问权限以及特定角色（如 SECURITYADMIN 和 ACCOUNTADMIN ）的最佳实践的更多信息，请参阅第 5 章。对于此任务的动手示例，我们将使用 ACCOUNTADMIN 角色，而不是 TASKADMIN 角色：

```
使用角色 ACCOUNTADMIN; 使用 WAREHOUSE
COMPUTE_WH; 使用数据库 DEMO3F_DB; 创建或替换
架构任务演示;
```

我们需要创建一个产品表，其中列出了产品 ID 和描述、类别和细分以及制造商 ID 和名称：

```
创建或替换表 DEMO3F_DB.TASKSDEMO 的 TASKS 演示。产品
(Prod_ID int, Prod_Desc
varchar (), 类别 varchar
(30), 段 varchar
(20), Mfg_ID int,
Mfg_Name varchar (50));
```

让我们在 product 表中插入一些值：

```
INSERT INTO DEMO3F_DB 中。TASKSDEMO 的 TASKS 演示。商品值 (1201, '商品 1201', '类别
1201', '细分 1201', '1201', '制造 1201'); INSERT INTO DEMO3F_DB 中。TASKSDEMO 的 TASKS 演示。
产品值 (1202, '产品 1202', '类别 1202', '细分 1202', '1202', '制造 1202'); INSERT
INTO DEMO3F_DB 中。TASKSDEMO 的 TASKS 演示。PRODUCT 值 (1203, 'Product 1203', 'Category
1203', 'Segment 1203', '1203', 'Mfg 1203'); INSERT INTO DEMO3F_DB 中。TASKSDEMO 的 TASKS 演示。
产品值 (1204, '产品 1204', '类别 1204', '细分 1204', '1204', '制造 1204'); INSERT
INTO DEMO3F_DB 中。TASKSDEMO 的 TASKS 演示。产品值 (1205, '产品 1205', '类别 1205', '细分
1205', '1205', '制造 1205'); INSERT INTO DEMO3F_DB 中。TASKSDEMO 的 TASKS 演示。商品值
(1206, '商品 1206', '类别 1206', '细分 1206', '1206', '制造 1206');
```

接下来，我们将创建一个 sales 表，然后在 sales 表上创建一个流：

```
创建或替换表 DEMO3F_DB.TASKSDEMO 的 TASKS 演示。SALES
(Prod_ID int, Customer varchar (), Zip varchar (),
Qty int, 收入十进制 (10, 2));
```

我们希望我们的 table 流在 sales table 上记录仅插入的值:

```
创建或替换 STREAM DEMO3F_DB。TASKSDEMO的 TASKS演示。SALES_STREAM  
ON TABLE DEMO3F_DB。TASKSDEMO的 TASKS演示。销售 APPEND_ONLY =  
TRUE;
```

在创建任务之前，我们希望确保我们的 stream 和其他语句正常工作。让我们继续在 sales 表中插入一些值:

```
INSERT INTO DEMO3F_DB 中。TASKSDEMO的 TASKS演示。销售  
值 (1201, 'Amy Johnson', 45466, 45, 2345.67) ;INSERT  
INTO DEMO3F_DB 中。TASKSDEMO的 TASKS演示。销售值  
(1201, 'Harold Robinson', 89701, 45, 2345.67) ;INSERT  
INTO DEMO3F_DB 中。TASKSDEMO的 TASKS演示。销售值  
(1203, '乍得·诺顿', 33236, 45, 2345.67) ;INSERT INTO  
DEMO3F_DB 中。TASKSDEMO的 TASKS演示。销售值  
(1206, 'Horatio Simon', 75148, 45, 2345.67) ;INSERT  
INTO DEMO3F_DB 中。TASKSDEMO的 TASKS演示。销售值  
(1205, 'Misty Crawford', 10001, 45, 2345.67) ;
```

当我们运行命令时，我们将能够看到我们在 sales 表中输入的值随后被捕获到 SALES_STREAM 中:

```
从 DEMO3F_DB中选择 *。TASKSDEMO的 TASKS演示。SALES_STREAM;
```

现在是时候创建我们的销售交易表了，该表将 SALES_STREAM 与 product 表组合在一起，并添加了时间戳，以便我们可以知道销售数据的记录时间:

```
创建或替换表 DEMO3F_DB。TASKSDEMO的 TASKS演示。SALES_TRANSACT
```

```
(Prod_ID int、Prod_Desc  
varchar ()、类别 varchar  
(30)、区段 varchar (20)、  
Mfg_ID int、Mfg_Name varchar  
(50)、客户 varchar ()、邮  
政编码 varchar ()、数量  
int、收入小数 (10, 2)、
```

```
TS 时间戳)；
```

我们最终希望自动创建 sales transaction 表，但首先我们需要查看手动插入数据时会发生什么。如果一切按预期进行，我们将继续为插入创建一个任务:

插入到

```
DEMO3F_DB。TASKSDEMO的 TASKS演示。SALES_TRANSACT  
(Prod_ID, Prod_Desc, 类别, 区段, Mfg_Id,  
Mfg_Name, 客户, 邮编, 数量, 收入, TS)
```

选择

```
s. Prod_ID, p. Prod_Desc, p. 类别, p. 细分, p. Mfg_ID, p. Mfg_Name, s. 客户,  
s. 邮编, s. 数量, s. 收入, current_timestamp
```

```
FROM  
DEMO3F_DB. TASKSDEMO的 TASKS演示。SALES_STREAM s JOIN  
DEMO3F_DB. TASKSDEMO的 TASKS演示。s.Prod_ID 上的乘积 p = p.Prod_ID;
```

如果我们使用语句，我们将确认记录实际上已插入到交易表中：

```
从 DEMO3F_DB中选择 *。TASKSDEMO的 TASKS演示。SALES_TRANSACT;
```

成功！我们已经证明了这个声明是有效的。现在是时候通过创建一个任务来自动插接插入了：

```
创建或替换任务DEMO3F_DB。TASKSDEMO的 TASKS演示。SALES_TASK仓库 = compute_wh 计划 = '1 分钟'
```

```
当 system$stream_has_data ('DEMO3F_DB. TASKSDEMO的 TASKS演示。SALES_STREAM')
```

```
AS
```

```
插入到
```

```
DEMO3F_DB. TASKSDEMO的 TASKS演示。SALES_TRANSACT
```

```
(Prod_ID, Prod_Desc, 类别, 区段, Mfg_Id,  
Mfg_Name, 客户, 邮编, 数量, 收入, TS)
```

```
选择
```

```
s.Prod_ID, p.Prod_Desc, p.类别, p.细分市场, p.Mfg_ID, p.Mfg_Name, s.客  
户, s.邮编, s.数量, s.收入, current_timestamp FROM
```

```
DEMO3F_DB. TASKSDEMO的 TASKS演示。SALES_STREAM s JOIN
```

```
DEMO3F_DB. TASKSDEMO的 TASKS演示。s.Prod_ID 上的乘积 p = p.Prod_ID;更  
改任务 DEMO3F_DB. TASKSDEMO的 TASKS演示。SALES_TASK简称:
```



默认情况下，已创建的任务将暂停。因此，您需要执行 ALTER
TASK...RESUME 语句以允许任务运行。

或者，对于非根树任务，请使用
功能。

系统\$TASK_DEPENDENTS_ENABLE

请注意，根任务可能会对 stream 事件执行某些操作。然而，它总是有限制，可以将其从最低的制粒时间安排到 1 分钟。前面的示例用于演示如何使用 \$STREAM_HAS_DATA 来实现此目的。

现在让我们看看当我们将值插入到 sales 表中时会发生什么。将值插入 sales 表后，SALES_STREAM 应反映新插入的记录。然后，该任务应在与 prod - uct 表联接并生成时间戳后插入新的销售记录。让我们看看会发生什么：

```
INSERT INTO DEMO3F_DB 中。TASKSDEMO的 TASKS演示。销  
售值 (1201, 'Edward Jameson', 45466, 45,  
2345.67) :INSERT INTO DEMO3F_DB 中。TASKSDEMO的
```

```
(1201, '玛格丽特·伏特', 89701, 45, 2345.67) ; INSERT  
INTO DEM03F_DB 中。TASKSDEMO 的 TASKS 演示。销售值  
(1203, 'Antoine Lancaster', 33236, 45, 2345.67) ; INSERT  
INTO DEM03F_DB 中。TASKSDEMO 的 TASKS 演示。销售值  
(1204, 'Esther Baker', 75148, 45, 2345.67) ;  
INSERT INTO DEM03F_DB 中。TASKSDEMO 的 TASKS 演示。销售  
值 (1206, 'Quintin Anderson', 10001, 45,  
2345.67) ;
```

我们可以确认这些值是由 SALES_STREAM 捕获的：

```
从 DEM03F_DB 中选择 *。TASKSDEMO 的 TASKS 演示。SALES_STREAM;
```

我们需要等待一分钟，以便给任务执行时间。如果您的下一个查询没有结果，请稍等片刻，然后重试。我们只需要等待一分钟，因为那是我们为任务设置的时间。如果我们愿意，我们可以增加时间：

```
从 DEM03F_DB 中选择 *。TASKSDEMO 的 TASKS 演示。SALES_TRANSACT;
```

我们已经确认我们的任务有效！我们将在本章末尾进行清理，但现在让我们继续暂停任务。我们不需要 sales 任务每分钟检查一次 SALES_STREAM 中的新记录：

```
更改任务 DEM03F_DB。TASKSDEMO 的 TASKS 演示。SALES_TASK 暂停;
```

与使用任务相关的成本取决于您是选择专用于该任务的虚拟仓库，还是选择使用 USER_TASK_MANAGED_INI TIAL_WAREHOUSE_SIZE 命令。后者是一项无服务器功能。您可以参考第 8 章，深入了解 Snowflake 计费以及监控虚拟仓库使用情况和降低成本的方法。重要的是，Snowflake 的计费基于使用情况，因此您最终只需为使用的资源付费。

代码清理

让我们执行代码清理以删除 Snowflake 账户中的对象，以便为处理另一个章节示例做准备。

请注意，在删除数据库之前，无需删除数据库下层次结构中的对象。如果您一直在关注，那么您已经删除了两个数据库。现在让我们删除剩余的数据库：

```
DROP DATABASE DEM03C_DB; DROP DATABASE DEM03D_DB; DROP  
DATABASE DEM03E_DB; DROP DATABASE DEM03F_DB;
```

总结

本章提供了几个创建和管理 Snowflake 对象（如数据库、架构、表、视图、阶段、存储过程和 UDF）的动手示例。此外，INFORMATION_SCHEMA 和 ACCOUNT_USAGE 视图

进行了审查。为了巩固我们对 Snowflake 架构和对象的基础知识，我们将在下一章中探索不同的 Snowflake 数据类型，并学习如何使用更多 SQL 命令和函数。

知识检查

以下问题基于本章提供的信息：

1. 可以创建哪些不同类型的数据库、架构和表？如果在创建时没有明确说明特定类型，则每个类型的默认类型是什么？
2. 标量 UDF 和表格 UDF 有什么区别？
3. 您可以使用存储过程执行哪些 UDF 无法执行的操作？
4. ~~如果我们要创建的数据库已经存在呢？~~ 如果我们使用 CREATE OR REPLACE DATA BASE 命令会怎样？
5. 数据库的默认保留时间是多少？是否可以更改数据库保留时间？是否可以更改默认保留时间？
6. 为什么选择使用命令而不是 DROP TABLE 命令？
7. 视图是否有任何相关的存储或计算成本？
8. 完全限定的对象名称和部分限定的对象名称之间有什么区别？
9. 使用 stages 时，默认文件格式是什么？Snowflake 还支持哪些其他文件格式？
10. 每个 Snowflake 帐户附带的 SNOWFLAKE 数据库有什么独特之处？
11. Snowflake 任务可以通过哪些方式触发？
12. 描述 Snowflake 流中的 METADATA \$ACTION、METADATA \$ISUPDATE 和 METADATA \$ROW_ID 列。
13. 举例说明您可能想要使用 Snowflake 流的原因。

这些问题的答案可在附录 A 中找到。

探索 Snowflake SQL 命令、 数据类型和函数

正如我们在前几章中学到的那样，Snowflake 旨在以优化、压缩的列式格式在关系数据库中存储数据。Snowflake 的数据最终用户需要访问存储的数据，并能够给出指令来执行任务、调用函数和对数据执行查询。实现此目的的方法是使用关系数据库的标准编程语言 Structured Query Lan - guage (SQL)。Snowflake 支持 SQL: ANSI，这是最常见的 SQL 标准化版本。除了对结构化数据的 SQL 支持外，Snowflake 还为半结构化数据格式（如 JSON 和 XML）提供本机支持端口。Snowflake 还支持非结构化数据。

本章的主要重点是学习使用 Snowflake 工作表执行使用不同数据类型和函数的各种 SQL 命令的基础知识。除了在 Snowflake Web UI 中使用工作表之外，还可以使用 Snowflake 原生命令行客户端（称为 SnowSQL）来创建和执行 SQL 命令。第 6 章将提供有关 SnowSQL 的更多详细信息。

除了通过 Web UI 或 SnowSQL 连接到 Snowflake 之外，您还可以使用 ODBC 和 JDBC 驱动程序通过外部应用程序（如 Tableau 和 Looker）访问 Snowflake 数据。我们将在第 12 章中探讨与 Tableau 和 Looker 的联系。Python 和 Spark 等本机连接器也可用于开发用于连接到 Snowflake 的应用程序。

为了帮助您为掌握接下来章节中的高级主题做好准备，我们首先要重点介绍 Snowflake SQL 命令、数据类型和 Snowflake 函数的基础知识。

准备工作

创建一个标题为 Chapter4 Syntax Examples, Data Types, and Functions 的新工作表。如果您在创建新工作表时需要帮助, 请参阅第 8 页上的“导航 Snowsight 工作表”。

要设置工作表上下文, 请确保您位于 Syntax Examples 工作表中, 并使用 SYSADMIN 角色和 COMPUTE_WH 虚拟仓库。

在 Snowflake 中使用 SQL 命令

SQL 可以分为五种不同的语言命令类型。要创建 Snowflake 对象, 您需要使用数据定义语言 (DDL) 命令。授予对这些对象的访问权限需要数据控制语言 (DCL)。接下来, 您将使用数据操作语言 (DML) 命令将数据移入和移出 Snowflake。事务控制语言 (TCL) 命令使您能够管理事务块。然后, 使用数据查询语言 (DQL) 语句实际查询数据。以下是按类型组织的常见 SQL 命令列表:

DDL 命令:

- 创造
- 改变
- 截断
- 重命名
- DROP
- 描述
- SHOW
- USE
- SET/UNSET (设置/取消设置)
- 评论

DCL 命令:

- 授予
- 撤回

DML 命令:

- 插入
- 合并
- 更新
- 删除

- PUT
- GET
- LIST
- 驗證
- 刪除

TCL 命令:

- 开始
- 犯
- 反转
- 创造

DQL 命令:

- 选择

这五种不同的命令语言类型中的每一种及其相关的命令都将在以下部分中简要讨论。可以在 Snowflake 在线文档中找到所有 Snowflake SQL 命令的完整列表。

DDL 命令

DDL 命令是用于定义数据库架构的 SQL 命令。这些命令用于创建、修改和删除数据库结构。此外，DDL commands 可用于执行账户级会话操作，例如设置参数，我们将在本章后面讨论 and commands 时看到。DDL 命令包括 CREATE、ALTER、DROP 和 SHOW。除命令外，每个 DDL 命令都采用对象类型和标识符。

Snowflake DDL 命令可操作数据库、虚拟仓库、架构、表和视图等对象；但是，它们不会处理数据。请参阅第 3 章，该章专门演示 Snowflake DDL 命令，以获取深入的解释和许多动手示例。

DCL 命令

DCL 命令是用于启用访问控制的 SQL 命令。DCL 命令的示例包括 GRANT 和 REVOKE。第 5 章将带您完成一系列使用 DCL 命令的完整而详细的示例，向您展示如何保护 Snowflake 对象。

DML 命令

DML 命令是用于处理数据的 SQL 命令。传统的 DML 命令（如 `INSERT`、`UPDATE` 和 `DELETE`）用于一般数据操作。对于数据加载和卸载，Snowflake 提供了 `INTO` 和命令。此外，Snowflake 的 DML 命令包括一些不执行任何实际数据操作但用于暂存和管理存储在 Snowflake 位置的文件的命令。一些示例包括 `CREATE TABLE AS SELECT` 和 `ALTER TABLE`。第 6 章将探讨 Snowflake 的许多 DML 命令。

TCL 命令

TCL 命令是用于管理 Snowflake 中的事务块的 SQL 命令。命令（如 `BEGIN TRANSACTION`、`COMMIT` 和 `ROLLBACK`）可用于会话中的多语句交易。Snowflake 事务是一组读取和写入 SQL 语句，它们作为一个单元一起处理。默认情况下，在查询成功时，如果查询失败，则单独运行的 DML 语句将被单独提交或在语句末尾回滚。

DQL 命令

DQL 命令是用作语句或子句的 SQL 命令，用于检索满足命令中指定条件的数据。请注意，`SELECT` 命令是唯一的 DQL 命令；它用于检索数据，方法是指定位置，然后使用该语句包含包含数据选择所需的属性。

Snowflake 命令适用于外部表，可用于查询历史数据。在某些情况下，使用该语句不需要正在运行的虚拟仓库返回结果；这是因为 Snowflake 缓存，如第 2 章所述。该语句的示例是最常见的 SQL 语句，可以在本书的大部分章节中找到。以下部分提供了有关如何充分利用该命令的详细信息。

SQL 查询开发、语法和运算符

雪花

在 Snowflake 中，可以使用 Snowflake UI 工作表或 SnowSQL 以及通过许多可用的第三方 SQL 工具在本地进行 SQL 开发。

查询语法是 Snowflake SQL 查询的结构或构建方式。通常有许多不同的方法可以编写 SQL 查询来产生所需的结果。请务必考虑如何优化查询以实现最佳数据库性能和最低成本。第 9 章包括一个专门讨论分析解析查询性能和优化技术主题的部分。

查询运算符包括保留的术语，用于指定 SQL 查询状态中的条件，最常用于子句中。它们还可以用作语句中多个条件的连词。我们将在本节后面探讨查询运算符。

SQL 开发和管理

有两个本机 Snowflake 选项可用于开发和查询数据。使用 Snowflake 界面中基于 Worksheets 浏览器的 SQL 编辑器，可以轻松开始 Snowflake SQL 开发。使用 Snowflake 工作表不需要安装或配置。到目前为止，我们只使用了 Snowflake Worksheets 来创建对象和查询数据。

SnowSQL 是 Sheetss 的替代方案，它是一个基于 Python 的客户端，可以从 Snowflake 客户端存储库下载，用于执行 Snowflake 任务，例如查询或执行 DDL 和 DML 命令。SnowSQL 经常用于加载和卸载数据。我们将在第 6 章中获得一些 SnowSQL 的实践经验。

Snowflake 提供适用于 Linux、macOS 和 Microsoft Windows 的 SnowSQL 版本。可执行 SnowSQL 可以作为交互式 shell 运行，也可以以批处理模式运行。Snowflake 提供了有关如何为所有支持移植的平台下载和安装 SnowSQL 的完整说明。

您可以通过查询 Snowflake 的查询历史记录，在 Snowflake 账户中查看最近使用的客户端版本，包括 SnowSQL 版本。要查看该信息，如果您使用的是新的 Snowsight Web 界面，请单击 Activity → Query History。如果您没有立即看到客户端驱动程序信息，请单击 列 按钮并选择 客户端驱动程序（如图 4-1 所示）。

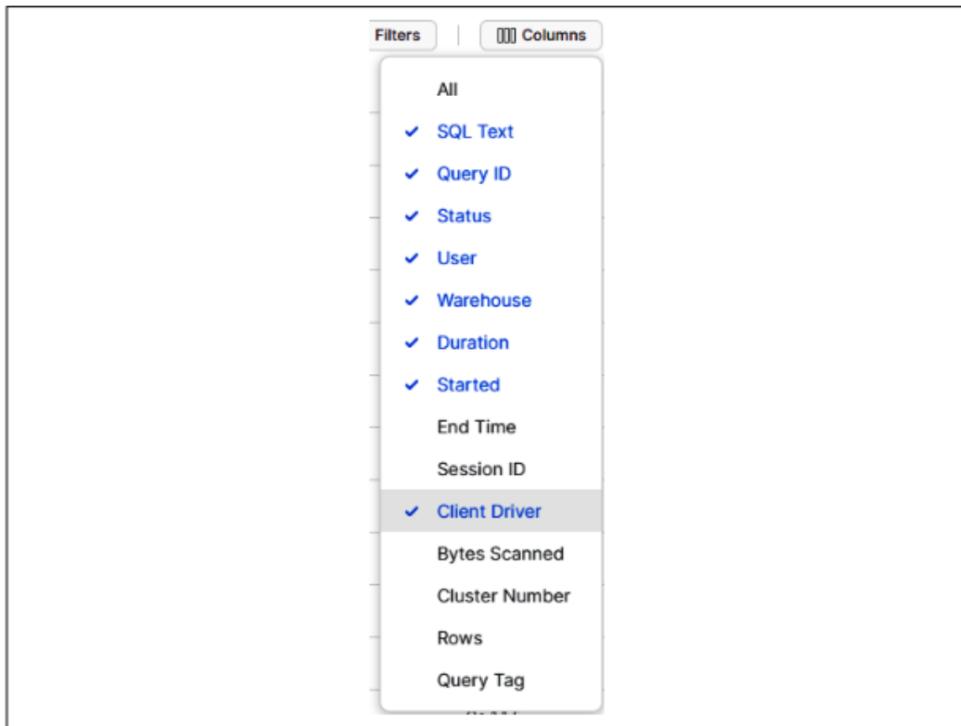


图 4-1. 用于在 Snowsight 中查看查询历史记录活动的可用列

或者，单击 Classic Console Web 界面中的 History 选项卡。从那里，您可以通过单击右上角的 Column 按钮并选择 Client Driver 来查看 Client Info 列。有趣的是，Classic Console Web 界面中的 Client Info（客户端信息）列包含一个图标，用于指示客户端版本是支持端口、不受支持还是即将终止支持。我们一直在使用 Snowsight Web UI，因此我们将看到 Go 客户端驱动程序已被使用并受到支持，如复选标记所示（如图 4-2 所示）。



图 4-2. History（历史记录）选项卡（Classic Console Web 界面）中的 Client Info（客户端信息）列

除了原生 Snowflake 工具之外，还有各种第三方 SQL 工具可用于在 Snowflake 应用程序中建模、开发和部署 SQL 代码。其中一些第三方工具（如 DataOps.live 和 SQLDRM）是

可通过使用 Snowflake Partner Connect 免费试用。您可以访问 Snowflake 在线文档，以获取可用于 Snowflake 的第三方 SQL 工具的更完整列表。



对于支持在执行前发送 SQL 语句进行准备的驱动程序和连接器，Snowflake 将准备 DML 命令，并执行从这些驱动程序和连接器接收的 SQL 语句。从驱动程序和连接器收到的其他类型的 SQL 语句将由 Snowflake 执行，无需准备。

查询语法

Snowflake SQL 查询以子句或 command 开头。子句是语句前面的可选子句，用于定义子句中引用的公共表表达式（CTE）。但是，大多数查询都以 command 和后面出现的其他语法开头。表 4-1 中描述的另一种语法按以下顺序进行评估：

- FROM
- 哪里
- 分组依据
- 拥有
- 窗
- 资格
- 排序依据
- 限制

表 4-1. Snowflake 查询语法

查询语法	查询子句	评论
WITH		位于 SELECT 语句正文前面的可选子句
返回页首		包含返回的最大行数，建议为 包括 ORDER BY
从 AT 之前、变化、 连接依据、联接、 MATCH RECOGNIZE、透视 或取消透视、样本或 TABLESAMPLE_VALUE		指定要在 SELECT 语句中使用的表、视图或表函数

查询
语法

查询子句

评论

哪里

指定与行子集匹配的条件;可以筛选 FROM 子句的结果;
可以指定要在 UPDATE、MERGE 或 DELETE 中操作的
行

按 GROUP BY CUBE、GROUP BY CUBE 和 GROUP BY 分组集,
按 ROLLUP 分组,
具有

使用相同的 group-by-item 表达式对行进行分组，并计算结果组的聚合函数;可以是列名、引用 SELECT 列表中某个位置的数字或通用表达式

资格

筛选窗口函数的结果

排序依据

指定 SELECT 列表中结果表的行的顺序

限制/
获取

限制返回的最大行数;建议至
包括 ORDER BY

请注意，在 window 函数之后计算;使用窗口函数的方式与使用聚合函数和 GROUP BY 子句的方式大致相同。有关窗口函数的更多信息，请参阅本章后面的内容。

子查询、派生列和 CTE

子查询是另一个查询中的查询，可用于计算在列表中返回的值、在子句中分组的值或与 or 子句中的其他表达式进行比较的值。

Snowflake 子查询是支持在下一个或多个 Snowflake SQL 语句中作为块的嵌套语句：

- CREATE TABLE AS
- 选择
- 插入
- 插入到
- 更新
- 删除

为了准备子查询和派生列的动手练习，我们需要创建一些简单的表并在这些表中插入一些值。我们将为本章创建一个数据库。我们还将为子查询和派生列示例创建架构和表。导航到 Snowsight 中的 Chapter4 工作表，以执行以下语句：

使用角色 SYSADMIN; 使用 WAREHOUSE COMPUTE_WH; 创建或替换数据
库DEMO4_DB; 创建或替换架构子查询; 创建或替换表 DEMO4_DB。
SUBQUERIES 的 SUBQUERIES 中。派生

(ID 整数、AMT 整数、总计整数) ;插入派生 (ID, AMT, 总计) 值
(1,1000,4000), (2,2000,3500), (3,3000, 9900), (4,4000,3000),
(5,5000,3700), (6,6000,2222);从 DEMO4_DB中选
择 *。SUBQUERIES 的 SUBQUERIES 中。派生;

您的结果应与图 4-3 中所示的结果相匹配。

	ID	AMT	TOTAL
1	1	1,000	4,000
2	2	2,000	3,500
3	3	3,000	9,900
4	4	4,000	3,000
5	5	5,000	3,700
6	6	6,000	2,222

图 4-3. 将值插入新的 Snowflake DERIVED 表的结果

我们需要 SUBQUERIES 架构中的第二个表;添加表格后, 我们会看到如图 4-4 所示的结
果:

创建或替换表 DEMO4_DB. SUBQUERIES 的 SUBQUERIES 中。表2
(ID 整数、AMT 整数、总计整数) ;插入表 2 (ID, AMT, Total) 值
(1,1000,8300), (2,1001,1900), (3,3000,4400), (4,1010,3535),
(5,1200,3232), (6,1000,2222);从 DEMO4_DB中选
择 *。SUBQUERIES 的 SUBQUERIES 中。表 2;

	ID	AMT	TOTAL
1	1	1,000	8,300
2	2	1,001	1,900
3	3	3,000	4,400
4	4	1,010	3,535
5	5	1,200	3,232
6	6	1,000	2,222

图 4-4. 将值插入新的 Snowflake TABLE2 表的结果

现在创建了两个表，我们可以编写一个不相关的子查询：

```
选择 ID、AMT from DEMO4_DB。  
SUBQUERIES 的 SUBQUERIES 中。派生，其  
中 AMT = (SELECT MAX (AMT)  
从 DEMO4_DB. SUBQUERIES 的 SUBQUERIES 中。表 2)；
```

您会注意到，不相关的子查询是独立的查询，其中返回的值不依赖于外部查询的任何列。不相关的子查询返回一个结果，该结果仅被外部查询使用一次。另一方面，相关子查询引用一个或多个外部列。在外部查询表的每一行上计算相应的子查询，并为每个计算的行返回一个结果。

现在让我们尝试执行一个 correlated subquery：

```
选择 ID、AMT from DEMO4_DB。  
SUBQUERIES 的 SUBQUERIES 中。派生，其  
中 AMT = (选择 AMT  
从 DEMO4_DB. SUBQUERIES 的 SUBQUERIES  
中。表 2 其中 ID = ID)；
```

我们收到一条错误消息，告诉我们单行子查询返回多行（如图 4-5 所示）。这可能不是你所期望的。

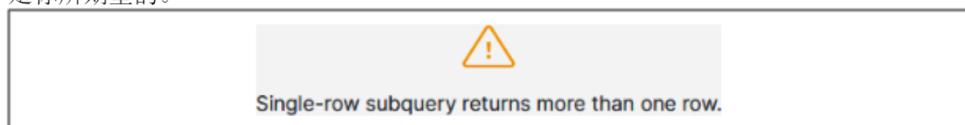


图 4-5. 在没有聚合的情况下执行相关子查询时收到的错误消息

从逻辑上讲，我们知道每个 ID 只有一行；因此，子查询不会在结果集中返回多行。但是，服务器无法知道这一点。我们必须使用 `,`，`,` 或 `函数`，以便服务器可以确定每次执行子查询时只会返回一行。

让我们继续添加到语句中，亲自看看它是如何工作的：

```
选择 ID、AMT from DEMO4_DB。  
SUBQUERIES 的 SUBQUERIES 中。派生，其  
中 AMT = (SELECT MAX (AMT)  
从 DEMO4_DB. SUBQUERIES 的 SUBQUERIES  
中。表 2 其中 ID = ID)；
```

成功！我们得到一个 ID 等于值 3 的一行的结果集。让我们看看如果我们将等号更改为大于号会发生什么：

```
选择 ID、AMT from DEMO4_DB。  
SUBQUERIES 的 SUBQUERIES 中。派生 其  
中 AMT > (SELECT MAX (AMT)
```

从 DEMO4_DB.SUBQUERIES 的 SUBQUERIES 中。表 2 其中 ID = ID) ;

现在我们得到一个包含三个值的结果集（如图 4-6 所示）。

	ID	AMT
1	4	4,000
2	5	5,000
3	6	6,000

图 4-6. 具有三个值的相关查询结果集

让我们看看如果我们将 :

```
选择 ID、AMT from DEMO4_DB.  
SUBQUERIES 的 SUBQUERIES 中。派生位置  
> AMT (SELECT AVG(AMT))  
从 DEMO4_DB.SUBQUERIES 的 SUBQUERIES  
中。表 2 其中 ID = ID) ;
```

结果集中有 5 条记录。您可能希望在

WHERE 子句和子句中的不同聚合器，以亲自了解相关子查询的实际工作原理。

相关子查询不经常使用，因为它们会导致每行一个查询，对于大多数使用案例来说，这可能不是最佳的可扩展方法。

子查询可用于多种用途，其中之一是计算或派生值，然后以各种不同的方式使用。派生列还可以在 Snowflake 中用于计算另一个派生列，可以由外部查询使用，也可以用作子句的一部分。这些派生列值（有时称为计算列值或虚拟列值）并不以物理方式存储在表中，而是在每次查询中引用它们时都会重新计算。

我们的下一个示例演示如何在 Snowflake 中使用派生列来计算另一个派生列。我们还将了解如何在一个查询、子查询和 CTE 中使用派生的 col - umn。

让我们从 AMT 列创建一个派生列 AMT1，然后直接使用第一个派生列创建第二个派生列 AMT2：

从 DEMO4_DB 中选择 ID、AMT、AMT * 10 作为 AMT1，AMT1 + 20 作为 AMT2。SUBQUERIES 的 SUBQUERIES 中。派生；

运行该查询的结果如图 4-7 所示。

	ID	AMT	AMT1	AMT2
1	1	1,000	10,000	10,020
2	2	2,000	20,000	20,020
3	3	3,000	30,000	30,020
4	4	4,000	40,000	40,020
5	5	5,000	50,000	50,020
6	6	6,000	60,000	60,020

图 4-7. 使用两个派生列运行查询的结果

我们可以通过创建一个派生列 AMT1 来实现相同的结果，然后该列可以由外部查询使用。我们示例中的子查询是 Snowflake 不相关的标量子查询。提醒一下，子查询被视为不相关的子查询，因为返回的值不依赖于任何外部查询列：

```
选择 sub.ID, sub.AMT, 子。AMT1 + 20 作为 AMT2 FROM
  (SELECT ID, AMT, AMT * 10 作为 AMT1
   从 DEMO4_DB.SUBQUERIES 的 SUBQUERIES 中。DERIVED) AS sub;
```

最后，我们通过使用派生列作为子句的一部分来获得相同的结果。您会注意到，我们包含一个 CTE 子查询，它可以帮助提高模块化并简化维护。CTE 定义了一个临时视图名称，在我们的示例中为 CTE1。CTE 中包括列名和查询表达式，其结果基本上是一个表：

```
CTE1 作为 (选择 ID、AMT、AMT * 10 作为 AMT2
  从 DEMO4_DB.SUBQUERIES 的 SUBQUERIES 中。派
  生) 从 DEMO4_DB 中选择 a.ID、b.AMT、b.AMT2 +
  20 作为 AMT2。SUBQUERTES 的 SUBQUERTES 中。派
  加入 CTE1 b ON (a.ID = b.ID) ;
```

使用 CTE 的一个主要好处是它可以使您的代码更具可读性。使用 CTE，您可以定义一次临时表，并在需要时引用它，而不必在每个需要的地方都声明相同的子查询。虽然此处未演示，但 CTE 也可以是递归的。递归 CTE 可以将表多次联接到自身以处理分层数据。



只要 CTE 与表或视图存在相同的名称，则 CTE 将优先。因此，建议始终为您的 CTE 选择一个唯一的名称。

关于多行插入的注意事项

现在是短暂暂停以了解有关多行插入的更多信息的好时机。可以使用 `select` 查询插入一行或多行数据，也可以将其作为明确声明的值插入到逗号分隔的列表中。为了简单起见，在本章中，我们在逗号分隔的列表中插入了值。

关于多行插入，有一件重要的事情需要注意。将多行数据插入数据类型时，要插入 `VARCHAR` 列的每种数据类型必须相同，否则插入将失败。数据类型可以接受数据值，例如单词 1 或数字 1，但不能在同一语句中接受两种类型的值。我们可以通过一些例子最好地看到这一点。

我们首先创建一个新的架构和表来执行一些多行插入测试。在第一个示例中，我们将值 1 插入到 `VARCHAR DEPT` 列中：

```
使用角色 SYSADMIN;
创建或替换 SCHEMA DEMO4_DB。测试;
创建或替换表 DEMO4_DB。测试。TEST1 (ID 整数, DEPT Varchar) ;INSERT INTO TEST1
(ID, DEPT) 值 (1, 'one') ;从 DEMO4_DB中选择 *. 测试。测试1;
```

正如预期的那样，该值已成功输入。让我们看看如果我们在 `VARCHAR` 列中插入一个数值会发生什么：

```
使用角色 SYSADMIN;
创建或替换 SCHEMA DEMO4_DB。测试;
创建或替换表 DEMO4_DB。测试。TEST1 (ID 整数, DEPT Varchar) ;INSERT INTO TEST1
(ID, DEPT) 值 (1,1) ;从 DEMO4_DB中选择 *. 测试。测试1;
```

同样，该值已成功输入。现在让我们尝试将这两种类型插入到同一语句中的列中：

```
使用角色 SYSADMIN;
创建或替换 SCHEMA DEMO4_DB。测试;
创建或替换表 DEMO4_DB。测试。TEST1 (ID 整数, DEPT Varchar) ;INSERT 到 TEST1
(ID, DEPT) 中 值 (1, 'one') , (2,2) ;从 DEMO4_DB中选择 *. 测试。测试1;
```

当我们尝试同时将两种不同的数据类型插入 `VARCHAR` 列时，会遇到错误，如图 4-8 所示。



图 4-8. 尝试将两种不同的数据类型插入到一个多行语句的 VARCHAR 列中时收到的错误消息

让我们再试一次，但这次我们将插入两个具有相同数据类型的值：

```
使用角色 SYSADMIN;  
创建或替换 SCHEMA DEMO4_DB。测试；  
创建或替换表 DEMO4_DB。测试。TEST1 (ID 整数, DEPT Varchar) ;INSERT INTO TEST1  
(ID, DEPT) 值 (1, 'one') , (2, 'two') ;  
  
从 DEMO4_DB中选择 *。测试。测试1；
```

如果我们将两个数值插入 VARCHAR 列，我们也成功了：

```
使用角色 SYSADMIN;  
创建或替换 SCHEMA DEMO4_DB。测试；  
创建或替换表 DEMO4_DB。测试。TEST1 (ID 整数, DEPT Varchar) ;INSERT 到 TEST1  
(ID, DEPT) 值 (1,1)、(2,2) 中;从 DEMO4_DB中选择 *。测试。测试1；
```

您会注意到，我们能够成功地将两种不同的数据类型加载到 VARCHAR 列中，但不能同时加载。一旦 VARCHAR 列中有两种不同的数据类型，我们仍然可以添加其他值：

```
INSERT INTO TEST1 (ID, DEPT) 值 (5,  
'5') ;从 DEMO4_DB中选择 *。测试。测试1；
```

多行插入是将数据导入 Snowflake 的一种方法。第 6 章专门介绍数据加载和卸载，并包括对批量数据加载选项和连续数据加载选项的深入讨论。

查询运算符

有几种不同类型的查询运算符，包括 arithmetic、comparison、logical、subquery 和 set 运算符。

算术运算符（包括 +、-、*、/ 和 %）从一个或多个输入生成数值输出。输出的 scale 和 precision 取决于 inputs 的 scale 和 precision。请注意，减法是 DATE 表达式上唯一允许的算术运算。

比较运算符（通常出现在子句中）用于测试两个输入的相等性。比较运算符包括以下内容：

- 等于 (=)
- 不等于 (或)
- 小于 (<)
- 小于或等于 ()
- 大于 (>)
- 大于或等于 () 请记住，值是根据其 UTC 时间进行比较的，这不考虑夏令时。



这很重要，因为在创造的那一刻，
TIMESTAMP_TZ 存储给定时区的偏移量，而不是实际时区。

逻辑运算符只能在子句中使用。这些运算符的优先顺序为 。子查询运算符包括 、 、 ANY 或 和 。使用 INTERSECT 或 、 、 和 等集合运算符时，可以组合查询。

默认设置的运算符优先顺序是作为最高优先级，然后是 、 和 ，最后是最低的优先级。当然，您始终可以使用括号来覆盖默认值。请注意，set 操作的成本很高，因为它需要对记录进行排序以消除重复的行。



使用 set 运算符时，请确保每个查询选择相同数量的列，并且每列的数据类型一致，但如果数据类型不一致，可以使用显式类型转换。

长时间运行的查询以及查询性能和优化

Snowflake 系统将取消长时间运行的查询。长时间运行的查询的默认持续时间为两天，但始终可以在账户、会话、对象或虚拟仓库级别设置

STATEMENT_TIMEOUT_IN_SECONDS 持续时间值。

在 Snowflake SQL 查询过程中，发生的一件事是 optimization 引擎为特定查询找到最有效的执行计划。在第 9 章中，我们将了解有关分析查询性能和优化技术以及如何使用 Snowflake 的查询分析器的更多信息。

Snowflake 查询限制

通过 Snowflake 客户端提交的 SQL 语句的查询文本大小限制为 1 MB。该限制中包括文本，包括字符串和二进制文本。查询文本大小限制适用于查询的压缩大小。但是，由于数据的压缩率差异很大，因此建议将未压缩的查询文本大小保持在 1 MB 以下。

此外，Snowflake 将查询中允许的表达式数限制为 16,384。有一些方法可以解决此类错误，具体取决于您尝试对 SQL 查询语句执行的操作。如果您在收到错误时尝试插入数据，请尝试将语句拆分为更小的查询。但是，更好的选择可能是使用 command 而不是 INSERT 命令。

当使用带有超过 16,384 个值的子句的语句时，会出现另一种类型的查询限制错误。

以下是该代码可能如下所示的示例：

```
<column_1> 从中选择<table_1></table_1></column_1>
其中 <column_2> (1, 2, 3, 4, 5,...) ;</column_2>
```

一种解决方案是在将这些值放入第二个 table 后使用 or 命令。SQL 代码可能如下所示：

```
选择 <column_1></column_1>
<table_1> 从</table_1>
在 <table_2> a.<column_2> = b. 上加入 b.<column_2></column_2></column_2></table_2>
```

Snowflake 支持的数据类型简介

Snowflake 支持基本的 SQL 数据类型，包括地理空间数据类型，以及提供三元逻辑的布尔逻辑数据类型。Snowflake 的数据类型可以具有未知值，也可以是 a 或 值。如果在表达式（如语句）中使用布尔值，则未知值将返回 . 如果将 Boolean 用作谓词（例如在子句中），则未知结果的计算结果将为 。Snowflake 不支持一些数据类型，例如大型对象（LOB），包括 和 ，以及用户定义的数据类型。

Snowflake 为地球表面的点、线和多边形等地理空间功能提供本机支持。Snowflake 数据类型遵循 WGS 标准。地球上的点以经度和纬度表示。

目前不支持 Altitude。



如果您有地理空间数据，例如经度和纬度、WKT、WKB 或 GeoJSON，建议您将此数据转换并存储在 GEOGRAPHY 列中，而不是将数据以其原始格式保留在 VARCHAR、VARIANT 或 NUMBER 列 umns 中。这可以显著提高使用地理空间功能的查询的性能。

在本节中，我们将更深入地研究几种 Snowflake 数据类型，包括数字、字符串和二进制、日期和时间、半结构化和非结构化。

数值数据类型

Snowflake 的数值数据类型包括定点数和浮点数，如表 4-2 中所述。表中包括有关每种数值数据类型的精度和小数位数的信息。精度（总位数）会影响存储，而小数位数（小数点后的数字数）则不会影响存储。但是，处理比例较大的数值数据值可能会导致处理速度变慢。

表 4-2. Snowflake 数值数据类型



一个已知问题是 DOUBLE、DOUBLE PRECISION 和 REAL 列存储为 DOUBLE，但显示为 FLOAT。

定点数是精确的数值，因此，通常用于自然数和精确的十进制值，例如货币金额。相比之下，浮点数据类型最常用于数学和科学。

您可以看到定点数如何根据数据类型而变化。请务必导航到 Chapter4 工作表，然后尝试以下示例：

```
使用角色 SYSADMIN;
创建或替换 SCHEMA DEMO4_DB。数据类型; 创建或替换表
NUMFIXED (
    数字数字, 数字 12 数字
    (12, 0), 十进制十进制
    (10, 2), 整数 整数 );
```

要查看创建的内容，您可以运行
结果如图 4-9 所示。

DESC 表 NUMFIXED 语句获取

	name	type	kind	null?
1	NUM	NUMBER(38,0)	COLUMN	Y
2	NUM12	NUMBER(12,0)	COLUMN	Y
3	DECIMAL	NUMBER(10,2)	COLUMN	Y
4	INT	NUMBER(38,0)	COLUMN	Y
5	INTEGER	NUMBER(38,0)	COLUMN	Y

图 4-9。显示 Snowflake 定点数数据类型的结果

现在，您可以使用下一个示例将定点数与浮点数进行比较：

```
使用角色 SYSADMIN; 使用 SCHEMA DEMO4_DB。数据类型;
创建或替换表 NUMFLOAT (
    FLOAT 浮点数,
    双倍、DP 双精度、实实数
);
```

再次使用命令查看结果，如图 4-10 所示：

DESC 表 NUMFLOAT;

	name	type	kind	null?
1	FLOAT	FLOAT	COLUMN	Y
2	DOUBLE	FLOAT	COLUMN	Y
3	DP	FLOAT	COLUMN	Y
4	REAL	FLOAT	COLUMN	Y

图 4-10. 显示 Snowflake 浮点数数据类型的结果

在传统计算中，众所周知，浮点数据类型的计算速度更快。但是，这仍然是关于现代数据平台中浮点数据类型的准确陈述吗

比如 Snowflake？不一定。请务必考虑到，整数值可以压缩格式存储在 Snowflake 中，而 float 数据类型则不能。这样可以减少整数的存储空间和成本。查询整数表类型的行所需的时间也要少得多。



由于浮点数据类型的不精确性，浮点运算可能会有小的舍入误差，并且这些误差可能会累积，尤其是在使用聚合函数处理大量行时。

数字常量支持 Snowflake 的数值数据类型。常量（也称为文本）表示固定数据值。数字 0 到 9 可以以正号或负号开头。由 e 或 E 表示的指数也由 Snowflake 数值常数支持。

字符串和二进制数据类型

Snowflake 支持文本和二进制字符串数据类型，其详细信息可在表 4-3 中看到。

表 4-3. Snowflake 文本和二进制字符串数据类型

文本字符串数据类型	参数	注释
瓦查尔	可选参数 (N), 最大数量 字符	保存 Unicode 字符; 使用全长 VARCHAR (16,777,216) 或更小的长度之间没有性能差异
字符, 文字		与 VARCHAR 同义; 如果未指定，则长度为 CHAR (1)
字符串, 文本		与 VARCHAR 同义
二进制字符串 类型	评论	
二元的		没有 Unicode 字符的概念，因此长度始终以字节为单位; 如果未指定 length，则默认值为 8 MB (最大长度)
VARBINARY		BINARY 的同义词

您可以通过尝试以下示例来了解文本字符串数据类型如何变化，该示例将创建文本字符串字段，然后描述表：

```
使用角色 SYSADMIN; 使用 SCHEMA DEM04_DB。数据类型;  
创建或替换表 TEXTSTRING (  
    瓦尔查尔 · 瓦尔查尔,  
    V100 VARCHAR (100)、  
    焦炭,  
    C100 字符 (100)、  
    字符串 STRING,  
    S100 字符串 (100)、
```

```
文本文本、  
T100 文本 (100)  
);  
  
DESC 表 TEXTSTRING;
```

如果您按照示例进行操作，您应该会看到如图 4-11 所示的输出。

	name	type	kind	null?
1	VARCHAR	VARCHAR(16777216)	COLUMN	Y
2	V100	VARCHAR(100)	COLUMN	Y
3	CHAR	VARCHAR(1)	COLUMN	Y
4	C100	VARCHAR(100)	COLUMN	Y
5	STRING	VARCHAR(16777216)	COLUMN	Y
6	S100	VARCHAR(100)	COLUMN	Y
7	TEXT	VARCHAR(16777216)	COLUMN	Y
8	T100	VARCHAR(100)	COLUMN	Y

图 4-11. 创建表的结果

字符串常量支持 Snowflake 的字符串数据类型，这些常量始终括在分隔符之间，可以是单引号，也可以是美元符号。当字符串包含许多引号字符时，使用美元符号作为分隔符特别有用。

日期和时间输入/输出数据类型

Snowflake 对所有日期和时间戳都使用公历，而不是儒略历。表 4-4 中汇总了 Snowflake 日期和时间数据类型。

表 4-4. Snowflake 日期和时间数据类型

日期和时间数据 类型	默认映射注释
DATE	单个 DATE 类型;接受最常见的日期表格;所有接受的时间戳都是 TIME 截断的有效输入;关联的时间假定为午夜
日期时间 TIME	TIMESTAMP_NTZ 的别名 格式为 HH: MI: SS 的单个 TIME 类型，内部存储为挂钟时间;未考虑时区
TIMESTAMP 默认值为 TIMESTAMP_NTZ	三种 TIMESTAMP_ 变体之一的用户指定别名

日期和时间数据	默认映射注释	
类型		
TIMESTAMP_LTZ		内部 UTC 时间, 具有指定的精度;具有本地时区的 TIMESTAMP
TIMESTAMP_NTZ	内部挂钟时间;不带时区的	TIMESTAMP
TIMESTAMP_TZ		内部 UTC 时间, 带时区偏移量;带时区的 TIMESTAMP

间隔常量以及日期和时间常量支持 Snowflake 的数据和时间数据类型。间隔常量可用于在日期、时间或时间戳中增加或减去特定时间段。间隔不是数据类型;它只能用于 Date、Time 或 Timestamp 算术, 如果未指定 Date 或 Time 部分, 则表示秒。



间隔增量的顺序很重要, 因为增量是按照其列出顺序添加或减去的。这对于受闰年影响的计算可能很重要。

半结构化数据类型

结构化数据 (称为定量数据) 可以很容易地以行和列的形式存储在数据库表中, 而半结构化数据 (如 XML 数据) 不依赖于架构, 这使得它更难存储在数据库中。但是, 在某些情况下, 半结构化数据可以存储在关系数据库中。

Snowflake 支持用于导入和操作半结构化数据 (如 JSON、Avro、ORC、Parquet 和 XML 数据) 的数据类型。Snowflake 通过其通用数据类型 VARIANT 来实现这一点, VARIANT 是一种特殊的列类型, 允许您存储半结构化数据。表 4-5 提供了有关 Snowflake 半结构化数据类型的更多信息。请注意, 可能会缺少一个值, 这被认为与 true null 值不同。

表 4-5. Snowflake 半结构化数据类型

半结构化 数据类型	特性注释	
变体	可以存储 OBJECT 和 ARRAY	存储任何其他类型的值, 未压缩时最大为 16 MB;内部以压缩的列式二进制表示形式存储
对象		表示键值对的集合, 其中键为非空字符串, 值为 VARIANT 类型
数组		表示任意大小的数组, 其索引为非负整数, 值具有 VARIANT 类型



加载到 VARIANT 列时，非本机值（如日期和时间戳）将存储为字符串。与将日期和时间戳值存储在具有相应数据类型的相关列中相比，以这种方式存储值可能会导致操作速度变慢并占用更多空间。

半结构化数据的动手练习将使用 Snowflake 示例天气数据，该数据以本机 JSON 格式存储。我们将花一些时间了解现有的数据，然后我们将学习如何使用该函数生成半结构化数据的横向视图。



在撰写本文时，天气数据集在 Snowflake 免费试用账户中可用。但是，随着时间的推移，Snowflake 可能会弃用此数据集。有关详细信息 <https://github.com/> 请参阅 SnowflakeDefinitiveGuide。

首先，让我们快速浏览一下几行数据：

```
使用角色 SYSADMIN; 使用 SCHEMA  
SNOWFLAKE_SAMPLE_DATA。天气;SELECT * 从  
DAILY_16_TOTAL LIMIT 5;
```

您应该看到有两列：一列 VARIANT 列 (V) 和一列时间戳列 (T)，如图 4-12 所示。

V	T
1 { "city": { "coord": { "lat": 47.616669, "lon": 9.78333 }, "country": "DE", "id": 2806110, "name": "Hergensweiler" }, "data": [{	2016-09-07 00:38:01.000
2 { "city": { "coord": { "lat": 50.166672, "lon": 8.56667 }, "country": "DE", "id": 2828737, "name": "Steinbach am Taunus" }, "data": [2016-09-07 00:38:01.000
3 { "city": { "coord": { "lat": 52.316669, "lon": 7.58333 }, "country": "DE", "id": 2898603, "name": "Horstel" }, "data": [{	2016-09-07 00:38:01.000
4 { "city": { "coord": { "lat": 50.816669, "lon": 8.96667 }, "country": "DE", "id": 2890504, "name": "Kirchhain" }, "data": [{	2016-09-07 00:38:02.000
5 { "city": { "coord": { "lat": 51.799999, "lon": 10.33333 }, "country": "DE", "id": 2939995, "name": "Clausthal-Zellerfeld" }, "data": [2016-09-07 00:38:02.000

图 4-12。Snowflake weather sample 数据表中的两列

让我们关注 VARIANT 列中的数据：

```
选择 v: city  
从 SNOWFLAKE_SAMPLE_DATA。天气。DAILY_16_TOTAL  
限制 10;
```

返回结果后，单击列顶部的 V: CITY。这将使该列高亮，并为您提供所需的详细信息，以查看此列中有四个不同的对象键（如图 4-13 所示）。与 V: CITY 相关的对象键依次为 coordinates、country、ID 和 name。

V:CITY				
1	{ "coord": { "lat": 51.050121, "lon": -112.685173 }, "country": "CA", "id": 5978906, "name": "Hussar" }			...
2	{ "coord": { "lat": 52.716709, "lon": -103.65097 }, "country": "CA", "id": 5902121, "name": "Bjorkdale" }			
3	{ "coord": { "lat": 46.683449, "lon": -64.165443 }, "country": "CA", "id": 5925150, "name": "Coleman" }			
4	{ "coord": { "lat": 44.750111, "lon": -79.699654 }, "country": "CA", "id": 6177107, "name": "Waubaushene" }			
5	{ "coord": { "lat": 52.250111, "lon": -107.501289 }, "country": "CA", "id": 5886799, "name": "Arelee" }			
6	{ "coord": { "lat": 45.55011, "lon": -78.58287 }, "country": "CA", "id": 5883647, "name": "Algonquin Park" }			
7	{ "coord": { "lat": 45.96677, "lon": -71.815804 }, "country": "CA", "id": 5921064, "name": "Chesterville" }			
8	{ "coord": { "lat": 52.500061, "lon": -105.734421 }, "country": "CA", "id": 5933765, "name": "Cudworth" }			
9	{ "coord": { "lat": 47.533249, "lon": -55.931641 }, "country": "CA", "id": 5974193, "name": "Hermitage" }			
10	{ "coord": { "lat": 49.340729, "lon": -55.083961 }, "country": "CA", "id": 6055114, "name": "Little Burnt Bay" }			

图 4-13. VARIANT 列中 CITY 数据的四个不同对象键

现在让我们手动分解一些 CITY 数据，并以更合乎逻辑的顺序列出它们（如图 4-14 所示）：

从 SNOWFLAKE_SAMPLE_DATA 中选择 v: city: id、v: city: name、v: city: country、v: city: coord。天气。DAILY_16_TOTAL 限制 10;

	V:CITY:ID	V:CITY:NAME	V:CITY: COUNTRY	V:CITY:COORD
1	5978906	"Hussar"	"CA"	{ "lat": 51.050121, "lon": -112.685173 }
2	5902121	"Bjorkdale"	"CA"	{ "lat": 52.716709, "lon": -103.65097 }
3	5925150	"Coleman"	"CA"	{ "lat": 46.683449, "lon": -64.165443 }
4	6177107	"Waubaushene"	"CA"	{ "lat": 44.750111, "lon": -79.699654 }
5	5886799	"Arelee"	"CA"	{ "lat": 52.250111, "lon": -107.501289 }
6	5883647	"Algonquin Park"	"CA"	{ "lat": 45.55011, "lon": -78.58287 }
7	5921064	"Chesterville"	"CA"	{ "lat": 45.96677, "lon": -71.815804 }
8	5933765	"Cudworth"	"CA"	{ "lat": 52.500061, "lon": -105.734421 }
9	5974193	"Hermitage"	"CA"	{ "lat": 47.533249, "lon": -55.931641 }
10	6055114	"Little Burnt Bay"	"CA"	{ "lat": 49.340729, "lon": -55.083961 }

图 4-14. VARIANT 列中 CITY 数据的详细信息

纬度和经度详细信息嵌套在坐标信息中。让我们把它们分开，并为这些列起一些合适的名称：

选择 v: city: id 作为 ID, v: city: name 作为 CITY,
 v: city: country 作为 COUNTRY, v: city: coord: lat 作为纬度, v:
 city: coord: lon 作为 SNOWFLAKE_SAMPLE_DATA 的经度。天气。
 DAILY_16_TOTAL 限制 10;

我们可以将 variant 数据类型转换为另一种数据类型。在下一个示例中，我们将 city 和 country 数据转换为数据类型，并将 mean - ingful 标签分配给列：

```
选择 v: city: id AS ID, v: city: name: : varchar AS city,
v: city.country: : varchar AS 国家/地区, v: city: coord: lon AS
经度, v: city: coord: lat AS 纬度 SNOWFLAKE_SAMPLE_DATA。天气。
DAILY_16_TOTAL 限制 10;
```

结果如图 4-15 所示。

	ID	CITY	COUNTRY	LONGITUDE	LATITUDE
1	2806746	Wolfsbehringen	DE	10.48333	51
2	2840860	Schaderode	DE	10.91667	51.01667
3	2807471	Wischuer	DE	11.7	54.099998
4	6549273	Giersleben	DE	11.56667	51.7666701
5	2920582	Giersleben	DE	11.56667	51.76667
6	2820258	Ullrichsberg	DE	13.15	51.083328
7	2857048	Orsberg	DE	7.25	50.599998
8	2863104	Niedermörsbach	DE	7.78333	50.716671
9	6551468	Krukow	DE	10.4833	53.416698
10	2883432	Krukow	DE	10.48333	53.416672

图 4-15. 将 city 和 country 数据转换为数据类型

我们可以通过要求 Snowflake 描述上一个查询的结果来确认我们已成功转换这两个列：

```
描述 结果 LAST_QUERY_ID();
```

接下来，让我们看看 VARIANT 列中的更多数据：

```
SELECT v: data
从 SNOWFLAKE_SAMPLE_DATA。天气。DAILY_16_TOTAL
限制 10;
```

返回结果后，单击列顶部的 V: DATA。这将对列进行高光照射，并提供您将在右侧看到的列详细信息（如图 4-16 所示）。您将注意到，此列中有一个与 DATA 信息相关的数组。

	V:DATA	Column
1	[{ "clouds": 0, "deg": 132, "dt": 1473246000, "humidity": 78, "pressure": 959.79, "sp": 1000.0, "temp": 13.0, "temp_min": 12.8, "temp_max": 13.2, "weather": "Cloudy", "wind_deg": 132, "wind_gust": 14.0, "wind_speed": 12.0 }]	[[V:DATA]]
2	[{ "clouds": 0, "deg": 86, "dt": 1473246000, "humidity": 63, "pressure": 1004.69, "sp": 1000.0, "temp": 15.0, "temp_min": 14.8, "temp_max": 15.2, "weather": "Cloudy", "wind_deg": 86, "wind_gust": 16.0, "wind_speed": 14.0 }]	100% filled
3	[{ "clouds": 12, "deg": 169, "dt": 1473246000, "humidity": 80, "pressure": 1027.23, "sp": 1000.0, "temp": 17.0, "temp_min": 16.8, "temp_max": 17.2, "weather": "Cloudy", "wind_deg": 169, "wind_gust": 18.0, "wind_speed": 16.0 }]	1 distinct type
4	[{ "clouds": 0, "deg": 101, "dt": 1473246000, "humidity": 70, "pressure": 996.1, "sp": 1000.0, "temp": 19.0, "temp_min": 18.8, "temp_max": 19.2, "weather": "Cloudy", "wind_deg": 101, "wind_gust": 20.0, "wind_speed": 18.0 }]	array
5	[{ "clouds": 8, "deg": 124, "dt": 1473246000, "humidity": 69, "pressure": 1022.45, "sp": 1000.0, "temp": 21.0, "temp_min": 20.8, "temp_max": 21.2, "weather": "Cloudy", "wind_deg": 124, "wind_gust": 22.0, "wind_speed": 20.0 }]	10
6	[{ "clouds": 0, "deg": 90, "dt": 1473246000, "humidity": 52, "pressure": 998.29, "sp": 1000.0, "temp": 23.0, "temp_min": 22.8, "temp_max": 23.2, "weather": "Cloudy", "wind_deg": 90, "wind_gust": 24.0, "wind_speed": 22.0 }]	
7	[{ "clouds": 20, "deg": 172, "dt": 1473246000, "humidity": 63, "pressure": 1029.26, "sp": 1000.0, "temp": 25.0, "temp_min": 24.8, "temp_max": 25.2, "weather": "Cloudy", "wind_deg": 172, "wind_gust": 26.0, "wind_speed": 24.0 }]	
8	[{ "clouds": 0, "deg": 133, "dt": 1473246000, "humidity": 65, "pressure": 1013.45, "sp": 1000.0, "temp": 27.0, "temp_min": 26.8, "temp_max": 27.2, "weather": "Cloudy", "wind_deg": 133, "wind_gust": 28.0, "wind_speed": 26.0 }]	
9	[{ "clouds": 8, "deg": 124, "dt": 1473246000, "humidity": 69, "pressure": 1022.45, "sp": 1000.0, "temp": 29.0, "temp_min": 28.8, "temp_max": 29.2, "weather": "Cloudy", "wind_deg": 124, "wind_gust": 30.0, "wind_speed": 28.0 }]	
10	[{ "clouds": 0, "deg": 92, "dt": 1473246000, "humidity": 73, "pressure": 967.89, "sp": 1000.0, "temp": 31.0, "temp_min": 30.8, "temp_max": 31.2, "weather": "Cloudy", "wind_deg": 92, "wind_gust": 32.0, "wind_speed": 30.0 }]	

图 4-16. VARIANT 列中的 weather 数据数组

因为 DATA 信息是以数组的形式存储的，所以我们可以查看数组中的特定元素。请务必单击每个结果行，以查看每行只选择一个元素：

```
选择 v: data[5]
从 SNOWFLAKE_SAMPLE_DATA。天气。DAILY_16_TOTAL
限制 10;
```

我们可以通过查看特定城市和国家/地区特定日期的湿度值来进一步限制返回的信息：

```
选择 v: city: name AS city, v: city: country AS country,
v: data[0]: 湿度 作为 SNOWFLAKE_SAMPLE_DATA 的湿度。天
气。DAILY_16_TOTAL 限制 10;
```

现在让我们快速回顾一下。当我们查看 DATA 数组时，在下面的语句中，我们注意到每一行都包含一个完整的数据数组。在每个数据数组中，每个不同的数据都有 16 个元素（如图 4-17 所示）。在我们的 SQL 查询中，我们将包含湿度和日温的前两个数据元素：

```
SELECT v: data[0]:d t: : timestamp 作为时间,
v: data[0]: humidity AS HUMIDITY0, v: data[0]: temp: day AS DAY_TEMPO,
v: data[1]: humidity AS HUMIDITY1, v: data[1]: temp: day AS DAY_TEMP1,
v: DATA 作为SNOWFLAKE_SAMPLE_DATA的数据。天气。DAILY_16_TOTAL

限制 100;
```

	TIME	HUMIDITY0	DAY_TEMP0	HUMIDITY1	DAY_TEMP1	DATA
1	I-07 11:00:00.000	78	295.03	65	297.5	[{"clouds": 0, "deg": 132, "dt": 1473246000, "humidity": 78, "pressure": 101.32, "temp": 295.03, "wind": 1473246000, "windchill": 65, "winddir": 132, "winddeg": 132, "windgust": 1473246000, "windhumidity": 78, "windpress": 1473246000, "windtemp": 295.03}, {"clouds": 0, "deg": 86, "dt": 1473246000, "humidity": 63, "pressure": 101.32, "temp": 295.03, "wind": 1473246000, "windchill": 63, "winddir": 86, "winddeg": 86, "windgust": 1473246000, "windhumidity": 63, "windpress": 1473246000, "windtemp": 295.03}, {"clouds": 0, "deg": 12, "dt": 1473246000, "humidity": 80, "pressure": 101.32, "temp": 295.03, "wind": 1473246000, "windchill": 80, "winddir": 12, "winddeg": 12, "windgust": 1473246000, "windhumidity": 80, "windpress": 1473246000, "windtemp": 295.03}, {"clouds": 0, "deg": 101, "dt": 1473246000, "humidity": 70, "pressure": 101.32, "temp": 295.03, "wind": 1473246000, "windchill": 70, "winddir": 101, "winddeg": 101, "windgust": 1473246000, "windhumidity": 70, "windpress": 1473246000, "windtemp": 295.03}, {"clouds": 0, "deg": 124, "dt": 1473246000, "humidity": 66, "pressure": 101.32, "temp": 295.03, "wind": 1473246000, "windchill": 66, "winddir": 124, "winddeg": 124, "windgust": 1473246000, "windhumidity": 66, "windpress": 1473246000, "windtemp": 295.03}, {"clouds": 0, "deg": 80, "dt": 1473246000, "humidity": 52, "pressure": 101.32, "temp": 295.03, "wind": 1473246000, "windchill": 52, "winddir": 80, "winddeg": 80, "windgust": 1473246000, "windhumidity": 52, "windpress": 1473246000, "windtemp": 295.03}, {"clouds": 0, "deg": 92, "dt": 1473246000, "humidity": 65, "pressure": 101.32, "temp": 295.03, "wind": 1473246000, "windchill": 65, "winddir": 92, "winddeg": 92, "windgust": 1473246000, "windhumidity": 65, "windpress": 1473246000, "windtemp": 295.03}, {"clouds": 0, "deg": 124, "dt": 1473246000, "humidity": 66, "pressure": 101.32, "temp": 295.03, "wind": 1473246000, "windchill": 66, "winddir": 124, "winddeg": 124, "windgust": 1473246000, "windhumidity": 66, "windpress": 1473246000, "windtemp": 295.03}, {"clouds": 0, "deg": 101, "dt": 1473246000, "humidity": 73, "pressure": 101.32, "temp": 295.03, "wind": 1473246000, "windchill": 73, "winddir": 101, "winddeg": 101, "windgust": 1473246000, "windhumidity": 73, "windpress": 1473246000, "windtemp": 295.03}, {"clouds": 0, "deg": 127, "dt": 1473246000, "humidity": 78, "pressure": 101.32, "temp": 295.03, "wind": 1473246000, "windchill": 78, "winddir": 127, "winddeg": 127, "windgust": 1473246000, "windhumidity": 78, "windpress": 1473246000, "windtemp": 295.03}, {"clouds": 0, "deg": 86, "dt": 1473246000, "humidity": 54, "pressure": 101.32, "temp": 295.03, "wind": 1473246000, "windchill": 54, "winddir": 86, "winddeg": 86, "windgust": 1473246000, "windhumidity": 54, "windpress": 1473246000, "windtemp": 295.03}], [{"day": 295.03, "eve": 295.03, "max": 295.03, "min": 285.11, "morn": 287.37, "night": 285.11}, {"uv": 0, "weather": [{"description": "clear"}]}]

图 4-17。每行一个 DATA 数组元素（路径）

让我们看看如何利用 table 函数。该函数生成 VARIANT、OBJECT 或 ARRAY 列的横向视图。我们将演示示例天气数据表中 DATA 数组的工作原理：

```
选择 d.value: dt : timestamp 作为时间,
v: city: name AS CITY, v: city: country AS COUNTRY,
d.path AS PATH, d.value: humidity AS HUMIDITY,
d.value: temp: day 作为 DAY_TEMP, v: DATA 作为
SNOWFLAKE_SAMPLE_DATA 的数据。天气。DAILY_16_TOTAL,
LATERAL FLATTEN (input => daily_16_total.v: data) d LIMIT
100;
```

您将注意到，16 行平展的每一行都显示了相同的 DATA 数组，但每行中的 和 都与数组的特定 PATH 相关联（如图 4-18 所示）。

	TIME	CITY	COUNTRY	PATH	HUMIDITY	DAY_TEMP	DATA
1	I-05:00:00	"Secuelame"	"ES"	[0]	83	289.78	[{"clouds": 0, "deg": 282, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 282, "winddeg": 282, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 292, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 292, "winddeg": 292, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 282, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 282, "winddeg": 282, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 292, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 292, "winddeg": 292, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 282, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 282, "winddeg": 282, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 292, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 292, "winddeg": 292, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 282, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 282, "winddeg": 282, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 292, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 292, "winddeg": 292, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 282, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 282, "winddeg": 282, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 292, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 292, "winddeg": 292, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 282, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 282, "winddeg": 282, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 292, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 292, "winddeg": 292, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 282, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 282, "winddeg": 282, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}, {"clouds": 0, "deg": 292, "dt": 1516536000, "humidity": 83, "pressure": 101.32, "temp": 289.78, "wind": 1516536000, "windchill": 83, "winddir": 292, "winddeg": 292, "windgust": 1516536000, "windhumidity": 83, "windpress": 1516536000, "windtemp": 289.78}], [{"day": 289.78, "eve": 289.78, "max": 289.78, "min": 276.53, "morn": 280.78, "night": 271.91}, {"uv": 1.673, "weather": [{"description": "clear"}]}]

图 4-18. 扁平化的 DATA 数组

DATA 数组中的温度信息有六个嵌套值： 、 、 、 、 、 。

min 、 和 。我们可以使用 nested 来进一步展平 DATA 数组。当我们这样做时，每个 DATA 行出现 96 次，16 个 PATH 值中的每个值出现 6 次（如图 4-19 所示）。

选择 d.value: dt: : timestamp 作为时间,
t.key, v: city: name AS CITY, v: city: country AS
COUNTRY, d.path AS PATH, d.value: humidity AS
HUMIDITY, d.value: temp: day AS DAY_TEMP,

d.value: temp: night AS NIGHT_TEMP,
v: data AS 数据
从 SNOWFLAKE_SAMPLE_DATA。天气。DAILY_16_TOTAL, 横向平度
(输入 => daily_16_total.v: 数据) d, 横向平度 (输入 =>
d 值, 温度) +

其中 v: city: id = 1274693

限制 100;

TIME	KEY	CITY	COUNTRY	PATH	HUMIDITY	DAY_TEMP	NIGHT_TEMP	DATA
1 0:00:00	day	"Chandrapur"	"IN"	[0]	44	308.46	288.06	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 308.46, "weather": [{"day": 308.46, "eve": 308.73, "max": 308.46, "min": 288.06, "morn": 300.42, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "
2 0:00:00	eve	"Chandrapur"	"IN"	[0]	44	308.46	288.06	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 308.46, "weather": [{"day": 308.46, "eve": 308.73, "max": 308.46, "min": 288.06, "morn": 300.42, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "
3 0:00:00	max	"Chandrapur"	"IN"	[0]	44	308.46	288.06	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 308.46, "weather": [{"day": 308.46, "eve": 308.73, "max": 308.46, "min": 288.06, "morn": 300.42, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "
4 0:00:00	min	"Chandrapur"	"IN"	[0]	44	308.46	288.06	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 308.46, "weather": [{"day": 308.46, "eve": 308.73, "max": 308.46, "min": 288.06, "morn": 300.42, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "
5 0:00:00	morn	"Chandrapur"	"IN"	[0]	44	308.46	288.06	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 308.46, "weather": [{"day": 308.46, "eve": 308.73, "max": 308.46, "min": 288.06, "morn": 300.42, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "
6 0:00:00	night	"Chandrapur"	"IN"	[0]	44	308.46	288.06	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 308.46, "weather": [{"day": 308.46, "eve": 308.73, "max": 308.46, "min": 288.06, "morn": 300.42, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "
7 0:00:00	day	"Chandrapur"	"IN"	[1]	51	300.83	283.7	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 300.83, "weather": [{"day": 300.83, "eve": 303.73, "max": 300.83, "min": 283.7, "morn": 298.49, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "
8 0:00:00	eve	"Chandrapur"	"IN"	[1]	51	300.83	283.7	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 300.83, "weather": [{"day": 300.83, "eve": 303.73, "max": 300.83, "min": 283.7, "morn": 298.49, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "
9 0:00:00	max	"Chandrapur"	"IN"	[1]	51	300.83	283.7	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 300.83, "weather": [{"day": 300.83, "eve": 303.73, "max": 300.83, "min": 283.7, "morn": 298.49, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "
10 0:00:00	min	"Chandrapur"	"IN"	[1]	51	300.83	283.7	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 300.83, "weather": [{"day": 300.83, "eve": 303.73, "max": 300.83, "min": 283.7, "morn": 298.49, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "
11 0:00:00	morn	"Chandrapur"	"IN"	[1]	51	300.83	283.7	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 300.83, "weather": [{"day": 300.83, "eve": 303.73, "max": 300.83, "min": 283.7, "morn": 298.49, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "
12 0:00:00	night	"Chandrapur"	"IN"	[1]	51	300.83	283.7	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 300.83, "weather": [{"day": 300.83, "eve": 303.73, "max": 300.83, "min": 283.7, "morn": 298.49, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "
13 0:00:00	day	"Chandrapur"	"IN"	[2]	48	298.71	288.55	[{"clouds": 0, "deg": 127, "dt": 15166003800, "humidity": 44, "pressure": 998.73, "speed": 2.78, "temp": 298.71, "weather": [{"day": 298.71, "eve": 303.73, "max": 300.83, "min": 283.7, "morn": 298.49, "night": 288.06}, {"uv": 7.674, "weather": [{"description": "

```
{
  "clouds": 0,
  "deg": 127,
  "dt": 15164000000,
  "humidity": 44,
  "pressure": 998.73,
  "speed": 2.78,
  "temp": 308.46,
  "weather": [
    {
      "day": 308.46,
      "eve": 308.73,
      "max": 308.46,
      "min": 288.06,
      "morn": 300.42,
      "night": 288.06
    },
    {
      "uv": 7.674,
      "weather": [
        {
          "description": "
```

图 4-19. 扁平化的嵌套 DATA 数组

正如我们刚才看到的，Snowflake 函数用于将半结构化数据转换为关系表示形式。



可以将 `WITH` 函数结合使用，以将事件分离到单个 JSON 对象中，同时保留全局数据。

非结构化数据类型

使用非结构化数据获得洞察有很多优势。非结构化数据通常是定性的，但它也可能是缺少行、列或分隔符的定量数据，就像包含定量数据的 PDF 文件一样。媒体日志、医学图像、呼叫中心录音的音频文件、文档图像和许多其他类型的非结构化数据可用于分析目的和情绪分析目的。存储和管理非结构化数据不容易。非结构化数据不是以预定义的方式组织的，这意味着它不太适合关系数据库。通常，非结构化数据存储在

Blob 存储位置，这有几个固有的缺点，使得搜索文件变得困难和耗时。

为了提高非结构化数据的可搜索性，Snowflake 最近推出了内置目录表。现在，使用表格文件目录搜索非结构化数据就像在 `directory` 表上使用命令一样简单。用户还可以在目录表之上构建表流，从而可以创建用于处理非结构化数据的管道。此外，Snowflake 用户可以在目录表上创建安全视图，因此还可以与他人共享这些安全视图。

Snowflake 如何支持非结构化数据使用

非结构化数据在当今生成的数据中所占的比例越来越大。非结构化数据类型的示例包括视频、音频或图像文件、日志文件、传感器数据和社交媒体帖子。非结构化数据可以是人工生成的，也可以是机器生成的，具有内部结构，但不能以结构化数据库格式存储。您想要利用所有这些非结构化数据的原因有很多。此类使用案例可能包括从呼叫中心录音中获取情绪分析等见解；通过使用保险卡或处方药上的光学字符识别程序提取文本进行分析；在 DICOM 医学图像上使用机器学习；或从存储的 PDF 文档中提取键值对。

非结构化数据很复杂，这已经不是什么秘密了。因此，存储、搜索和分析它存在许多挑战。传统的数据仓库和数据湖无法充分支持当今数据格式的工作负载需求，尤其是非结构化数据。但是，Snowflake 不是传统的数据仓库或数据湖。相反，它是一个为云从头开始构建的数据平台；因此，它消除了与存储、搜索和分析或处理非结构化数据相关的许多困难。

使用非结构化数据时，首先要考虑的是非结构化文件的存储方式和位置。使用 Snowflake 时，有两种方法可以执行此操作：内部阶段和外部阶段。如果我们想在 Snowflake 上内部存储数据，我们会使用内部阶段；特别是如果我们正在寻找一种简单、易于管理的解决方案。这是因为 Snowflake 会自动管理存储的可扩展性、加密、数据压缩和其他方面。如果我们有旧数据存储在云中的其他位置，我们也可以使用称为自带存储的外部阶段，因为无需将所有数据移动到 Snowflake 中。

虽然可以使用 VARIANT 列类型在 Snowflake 表中内部存储非结构化数据，但通常不建议这样做，因为文件存储限制为 16 MB。如果我们改用 stage，则没有大小限制

除了构建 Snowflake 实例的主要云提供商施加的限制外：AWS 和 GCP 为 5 TB 数据，Azure 为 256 GB 数据。

无论您使用内部还是外部 Snowflake 阶段，都可以通过基于角色的访问控制轻松实现对数据的控制访问。通过使用 `and` 语句，可以通过向角色授予权限来授予 Snowflake 资源（如阶段）权限，然后将这些权限授予个人。它很容易理解和学习如何提供对内部或外部阶段中的数据的精细访问，或者对存储在视图中的数据子集的访问，这些视图是在阶段之上创建的 Snowflake 对象。有关 Snowflake 访问控制的复习，请参阅第 5 章。

使用 Snowflake，可以通过三种不同的方式存储和授予对非结构化数据的访问权限：
暂存文件 URL、范围限定 URL 或预签名 URL。

Stage 文件 URL 访问

阶段文件 URL 用于创建指向 Snowflake 阶段上的文件的永久 URL，最常用于自定义应用程序。通过向 REST API 端点发出请求以及授权令牌来访问文件 URL。请注意，用户必须具有舞台上的读取权限。阶段文件 URL 具有一个独特的功能，即它们可以在 Snowflake 目录表中列出。

创建目录表（如文件目录）的能力是 Snowflake 为非结构化数据提供的独特功能，您可以轻松搜索以检索文件 URL 以访问暂存文件以及其他元数据。已被授予权限的 Snowflake 角色可以查询目录表以检索 URL 以访问暂存文件。

例如，无论您是想按文件大小还是按上次修改日期排序，还是只取前 100 个文件或最大的文件，都可以使用 Snowflake 直接表格来实现。您还可以将 Snowflake 流和任务与目录表结合使用，以实现强大的组合。例如，使用表流，您可以轻松找到最近添加的所有新文件。由于目录表是一个表，因此您可以逐个表单进行精细的选择和搜索操作。常规 Blob 存储中的搜索操作非常困难，因为它们没有表格格式的目录信息。

Snowflake 目录表是内置的只读表。因此，您无法在 `directory` 表中添加更多列或修改列。您可以做的是使用 Snowflake 流和任务来计算值，并将它们放入一个新表中，该表的列包含计算结果。然后，您将能够通过创建视图将该表与 `directory` 表连接。如果需要，您还可以添加标记。

范围限定的 URL 访问

范围化 URL 经常用于自定义应用程序；尤其是在使用数据共享的其他账户将获得对数据的访问权限的情况下

功能，或者使用 Snowsight 在内部执行临时分析时。使用 Snowflake 可以轻松地在云中安全地共享非结构化数据。在舞台上不需要任何权限。相反，您将创建一个安全视图，并使用范围限定的 URL 共享安全视图的内容。范围限定的 URL 是编码的，因此无法从 URL 中确定帐户、数据库、架构或其他存储详细信息。

使用范围限定 URL 访问访问阶段中的文件可通过以下两种方式之一实现。一种方法是让 Snowflake 用户单击 Snowsight 结果表中的作用域 URL。另一种方法是在请求中发送限定范围的 URL，这会导致 Snowflake 对用户进行身份验证，验证限定范围的 URL 尚未过期，然后将用户重定向到云存储服务中的暂存文件。请记住，暂存文件在云存储中的位置是编码的，因此用户无法确定位置。API 调用输出中的范围 URL 在 24 小时内有效，即结果缓存存在的当前时间长度。



出于安全原因，无法共享已与您共享的范围 URL。如果您要与没有授予他们类似访问权限的其他人共享链接，则会显示消息 access denied。

预签名 URL 访问

预签名 URL 最常用于需要为打开的文件显示非结构化文件内容的商业智能应用程序或报告工具。由于预签名 URL 已经过身份验证，因此用户或应用程序可以直接访问或下载文件，而无需传递授权令牌。

`GET_PRESIGNED_URL` 函数使用阶段名称和相对文件路径作为输入来生成阶段文件的预签名 URL。使用预签名 URL 访问阶段中的文件可以通过三种不同的方式完成：在 Web 浏览器中使用预签名 URL 直接导航到文件，单击 Snowsight 结果表中的预签名 URL，或在 REST API 调用请求中发送预签名 URL。

使用 Java 函数和外部函数处理非结构化数据

对文件内部的非结构化数据运行进程的能力是 Snowflake 提供的最令人兴奋的功能之一。目前，使用 Snowflake 处理非结构化数据有两种方法：Java 函数和外部函数。将来，Snowflake 计划增加使用 Python 函数处理非结构化数据的功能。

如果您已经有编写用于非结构化数据的 Java 代码，那么使用 Java 用户定义函数（UDF）是有意义的。请注意，Java UDF 是使用 Snowflake 虚拟仓库直接在 Snowflake 中执行切割的。因此，Java UDF

请勿在 Snowflake 边界之外进行任何 API 调用。在 Snowflake 环境中，一切都受到严密保护和管理。

如果有您想要使用的外部 API 服务（例如机器学习模型、地理编码器或其他自定义代码），则可以使用外部函数。外部函数使使用现有的机器学习服务从图像中提取文本，或处理 PDF 文件以提取键值对成为可能。在外部功能中，您可以使用任何 AWS、Azure 或 GCP 功能，包括 AWS Rekognition 或 Azure Cognitive Services。对未构建的数据执行的外部功能，无论是存储在内部阶段还是外部阶段，都可用于消除导出和重新导入数据的需要。

Snowflake SQL 函数和会话变量

Snowflake 使用用户能够创建 UDF 和使用外部函数，以及访问许多不同的内置函数。会话变量还扩展了 Snowflake SQL 功能。

使用系统定义（内置）函数

Snowflake 内置函数的示例包括标量、聚合、窗口、表和系统函数。

标量函数接受单个行或值作为输入，然后返回一个值作为结果，而聚合函数也返回单个值，但接受多个行或值作为输入。

标量函数

某些标量函数对字符串或二进制输入值进行操作。示例包括

CONCAT 、 、 、 和 大小写转换，以及 . 其他标量文件函数（如 GET_STAGE_LOCATION ）使您能够访问 Snowflake 云存储中暂存的文件。

此外，您可以在 Snowflake 中使用日期和时间数据类型执行许多操作。标量日期和时间函数以及数据生成函数的一些示例包括：

- 使用月、日和年组件构造/解构（提取）。
- 将日期截断或“四舍五入”到更高的级别。
- 使用字符串解析日期并设置日期格式。
- 加/减以查找和使用日期差异。
- 生成系统日期或日期表。

聚合函数

Snowflake 聚合函数将始终返回一行，即使输入不包含任何行。从输入包含零行的聚合函数返回的行可以是零、空字符串或其他值。聚合函数可以是通用性质的，例如 `+` 和 `*`。聚合功能还包括线性回归、统计和概率、频率估计、百分位数估计等等。

Snowflake 窗口函数是一种特殊类型的聚合函数，可以对行的子集进行操作。此相关行的子集称为窗口。与为一组行返回单个值的聚合函数不同，窗口函数将为每个输入行返回一个输出行。输出不仅取决于传递给函数的 `individual` 行，还取决于传递给函数的 `window` 中其他行的值。

窗口函数通常用于查找同比百分比变化、移动平均值和运行或累积总计，以及按分组或自定义标准对行进行排名。

让我们将聚合函数与窗口函数进行比较。在第一个例子中，我们将使用字母表中的元音及其相应的位置来创建一个聚合函数：

```
SELECT LETTER, SUM(LOCATION) as AGGREGATE
FROM (SELECT 'A' as LETTER, 1 as LOCATION
      UNION ALL (SELECT 'A' 作为 LETTER, 1 作为 LOCATION)
      UNION ALL (选择 'e' 作为 LETTER, 5 作为 LOCATION)
      ) 作为 AGG TABLE
```

按字母分组；

此查询的结果如图 4-20 所示。

	LETTER	AGGREGATE
1	A	2
2	E	5

图 4-20. 聚合函数查询的结果

接下来，我们将使用相同的逻辑创建一个窗口函数：

```
SELECT LETTER, SUM(LOCATION) OVER (PARTITION BY LETTER) 作为 WINDOW_FUNCTION
FROM (SELECT 'A' as LETTER, 1 as LOCATION
      UNION ALL (选择 'A' 作为 LETTER, 1 作为 LOCATION)
      UNION ALL (选择 'e' 作为 LETTER, 5 作为 LOCATION) )
      作为 WINDOW TABLE;
```

请注意，在图 4-21 中，字母 A 在窗口函数中与聚合函数中的和值相同，但在结果中重复，因为输入有两个单独的 A 列表。

	LETTER	WINDOW_FUNCTION
1	A	2
2	A	2
3	E	5

图 4-21. 窗口函数的结果

表函数

表函数（通常称为表格函数）以表格格式返回结果，其中包含一列或多列，并且没有、一行或多行。大多数 Snowflake 表函数都是 1 对 N 函数，其中每个输入行生成 N 个输出行，但也存在一些 M 到 N 表函数，其中一组 M 个输入行生成一组 N 个输出行。表函数可以是系统定义的，也可以是用户定义的。系统定义的表函数的一些示例包括 ‘RESULT_SCAN’ 和 ‘’。

系统功能

内置系统函数返回系统级信息或查询信息，或执行控制操作。

选择 SYSTEM\$CLUSTERING_INFORMATION；

系统控制功能允许您在系统中执行操作。控制函数的一个示例是

SYSTEM\$CANCEL_ALL_QUERIES 并且需要会话 ID。您可以通过 ACCOUNTADMIN 身份登录来获取会话 ID。从 Snowsight 的主菜单中，转到 Activity → Query History，然后使用 Column 按钮选择会话 ID，以便显示会话 ID。或者，转到 Classic Console 界面中的 Account → Sessions：

选择 SYSTEM\$CANCEL_ALL_QUERIES();

如果您需要取消对特定虚拟仓库或用户的查询，而不是会话，则需要使用命令和命令，而不是系统控制功能。

创建 SQL 和 JavaScript UDF 以及使用会话变量

SQL 功能可以通过 SQL UDF、Java UDF、Python UDF 和 session 变量进行扩展。我们在第 3 章中深入研究了 SQL 和 JavaScript UDF，因此我们将在本节中专注于了解有关会话变量的更多信息。

Snowflake 支持用户使用命令声明的 SQL 变量。当 Snowflake 会话处于活动状态时，这些会话变量存在。变量在 Snowflake SQL 语句中由前缀区分，并且在与对象一起使用时也可以包含标识符名称。您必须将变量包装在标识符中，例如 IDENTIFIER (\$Variable)，才能将变量用作标识符。或者，您也可以将变量包装在子句上下文中的对象内。

要查看当前会话中定义的所有变量，请使用命令。

会话变量函数的一些示例包括：

- SYS_CONTEXT and
- SESSION_CONTEXT and SET_SESSION_CONTEXT
- GETVARIABLE and

关闭 Snowflake 会话时，将删除在会话期间创建的所有变量。

如果要在会话期间销毁变量，可以使用命令。

外部功能

外部函数是一种 UDF，它调用在 Snowflake 外部存储和执行的代码。Snowflake 支持标量外部函数，这意味着远程服务必须为接收到的每一行返回一行。在 Snowflake 中，外部函数存储为 Snowflake 用于调用远程服务的数据库对象。

请务必注意，Snowflake 通常不是直接调用远程服务，而是调用代理服务将数据中继到远程服务。Amazon API Gateway 和 Microsoft Azure API 管理服务是可以使用的代理服务的两个示例。远程服务可以实现为 AWS Lambda 函数、Microsoft Azure 函数或在 EC2 实例上运行的 HTTPS 服务器（例如 Node.js）。

远程服务提供商的任何费用将单独计费。使用外部功能时，Snowflake 会收取与数据传输和虚拟仓库使用相关的正常费用。

使用外部函数有很多优点。除了从 Snowflake 中调用之外，还可以创建外部函数以从其他软件程序中调用。此外，远程服务的代码可以用 Go 或 C# 等语言编写，这些语言不能在 Snowflake 支持的其他 UDF 中使用。最大的优势之一是 Snowflake 外部函数的远程服务可以与市售的第三方库（如机器学习评分库）连接。

代码清理

本章的代码清理很简单。您可以使用以下命令删除我们之前创建的数据库：

```
DROP DATABASE DEMO4_DB;
```

请注意，我们不必先删除所有表，因为删除数据库将自动删除关联的表。

总结

在本章中，我们使用 SYSADMIN 角色创建并执行了所有 Snowflake 查询。这是有意为之的，这样我们就可以专注于学习 Snowflake SQL 命令、函数、语句和数据类型的基础知识，而不会增加需要导航 Snowflake 访问控制的复杂性。现在是时候基于这些基础知识，以及我们在第 3 章中学到的有关创建和管理架构对象的知识。

在下一章中，我们将深入探讨如何利用 Snowflake 访问控制。如果您希望被分配为其中一个核心管理员角色的管理员职责，那么下一章可能是您在 Snowflake 学习之旅中最重要的章节之一。即使您从未期望履行管理职责，您仍然需要知道如何在分配给您的权限内利用 Snowflake 的全部功能。此外，即使您没有被分配 Snowflake 管理员角色，您仍有可能获得执行某些曾经仅为管理员保留的功能的权限。

Snowflake 非常小心地设计和构建访问控制，以解决其他平台的一些弱点。这方面的例子是 Snowflake 专门设计了一个访问控制系统，该系统消除了超级用户的概念，这是许多平台的主要风险。也就是说，重要的是要认识到，即使您有为其他平台构建的访问控制的经验，您仍然可以了解有关 Snowflake 独特访问控制的很多知识。

知识检查

以下问题基于本章中包含的信息：

1. 您可以使用什么来确保一行文本是注释，而不是将其视为代码？
2. 字符串常量支持 Snowflake 的字符串数据类型。可以使用哪些分隔符来括起字符串？
3. 使用外部功能有哪些优势？
4. Snowflake 用于确定何时取消长时间运行的查询的默认持续时间是多少？您能否更改该持续时间，如果可以，您将如何更改？
5. 使用浮点数数据类型有哪些风险？
6. 窗口函数与聚合函数有何不同？
7. Snowflake 是否支持非结构化数据类型？
8. Snowflake 支持哪些半结构化数据类型？
9. Snowflake 的数据类型是否支持本地时区和夏令时？解释。
10. 什么是派生列，如何在 Snowflake 中使用它们？
11. 您可以通过哪三种方式访问 Snowflake 中的非结构化数据文件？
12. 列出一些非结构化数据的示例。
13. 目录表是什么类型的表？

这些问题的答案可在附录 A 中找到。

利用 Snowflake 访问控制

一个组织的数据存储库可能充满了有价值的机密信息。因此，保护这些数据是必不可少的，并且通常是法规或法定要求。尽管数据安全需求至关重要，但必须合理平衡数据安全需求与高效数据访问的业务需求。换句话说，数据访问不应过于严格，以至于为了安全起见而阻碍企业有效和高效运营的能力。相反，必须在适当的时间向适当的用户提供必要的数据，而不会牺牲安全性。为了实现这种平衡，开发良好的安全模型必须包括规划并与正确的利益相关者合作。Snowflake 使实现这种平衡变得容易。

利用 Snowflake 的内置安全选项创建多层次安全性是管理安全性的最佳实践之一。外部安全层（图 5-1 中最外层的同心正方形）依赖于网络策略、密钥对身份验证、多重身份验证（MFA）和安全专用网络作为选项，以确保只有经过身份验证的用户才能访问 Snowflake 帐户。Snowflake 通过数据加密提供数据安全保护。第 7 章将更详细地讨论这两个安全外层。

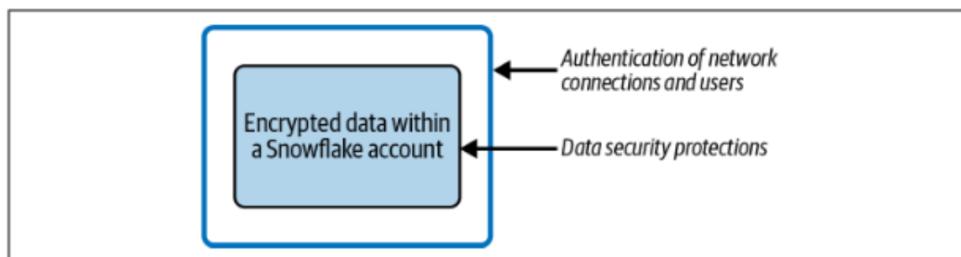


图 5-1。Snowflake 帐户中的所有数据都经过加密，并受到身份验证的保护

在本章中，我们将了解下一层，即用户管理和对象安全。此安全类型使用基于角色的访问控制和自主访问控制来为用户打开对某些数据的访问权限（请参见图 5-2）。除此以外，第 7 章中也介绍了数据隐私访问限制，这些限制是限制访问的内容，例如动态数据掩码、行级安全性以及使用安全视图和安全用户定义函数（UDF），以提供对敏感和私有数据的有限访问。

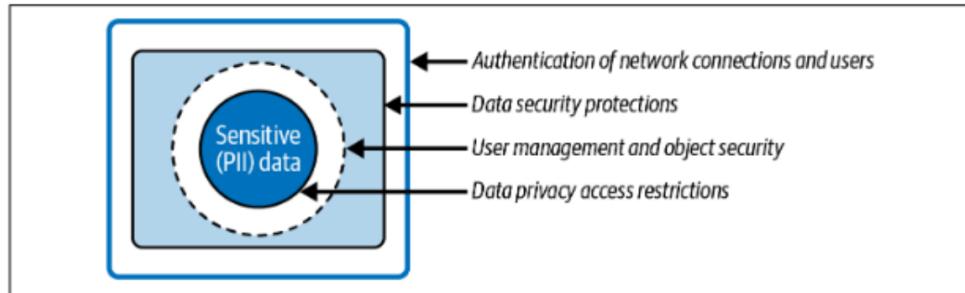


图 5-2. Snowflake 帐户中 Snowflake 数据的分层安全性

在本章中，您将通过一系列相互构建的示例来了解如何使用访问控制来保护 Snowflake 对象。以下是您将完成的步骤：

- 了解 Snowflake 的访问控制模型
- 创建安全对象
- 创建自定义角色
- 分配角色层次结构
- 向角色授予权限
- 为用户分配角色
- 测试和验证工作
- 执行用户管理
- 代码清理
- 知识检查

准备工作

创建一个名为 Chapter5 Snowflake Access Controls 的新工作表。如果您在创建工作表时需要帮助，请参阅第 8 页上的“导航 Snowsight 工作表”。要设置工作表上下文，请确保您使用的是 SYSADMIN 角色和 COMPUTE_WH 虚拟仓库。

Snowflake 的混合访问控制提供精细级别的访问。自主访问控制和基于角色的访问控制方法的组合决定了谁可以访问什么对象并对这些对象执行操作，以及谁可以自己创建或更改访问控制策略。自主访问控制（DAC）是一种安全模型，其中每个对象都有一个对该对象具有控制权的所有者。基于角色的访问控制（RBAC）是一种将访问权限分配给角色，然后将角色分配给一个或多个用户的方法。

在 Snowflake 混合方法中，所有安全对象都由角色而不是用户拥有。此外，每个安全对象仅由一个角色拥有，该角色通常是用于创建对象的角色。请注意，由于可以将一个角色分配给多个用户，因此被授予相同角色的每个用户也都以共享的受控方式具有固有权限。对象所有权可以从一个角色转移到另一个角色。Snowflake 访问控制关键概念由以下定义和图 5-3 描述：

安全对象

实体，例如数据库或表。除非特别授予，否则将拒绝对安全对象的访问。

Role

获得访问对象和对对象执行操作的权限，或者创建或更改访问控制策略本身的权限。将角色分配给用户。还可以将角色分配给其他角色，从而创建角色层次结构。

特权

对对象的固有、已分配或继承的访问权限。

User

Snowflake 可识别的人员、服务帐户或程序。

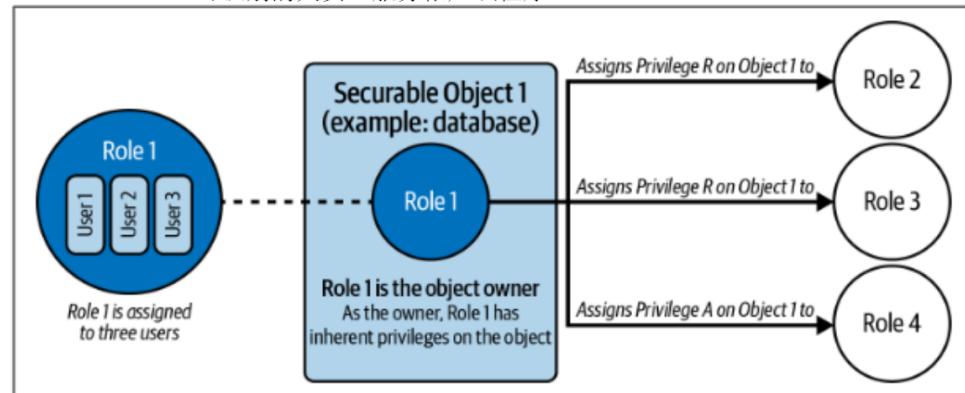


图 5-3。访问控制关键概念

如第 3 章所述，每个安全数据库对象都驻留在包括架构和数据库的层次结构中的逻辑容器中，其中帐户是所有安全对象的顶部容器。图 5-4 说明了 Snowflake 对象以及系统定义的角色，该角色本身拥有每种类型的安全对象。

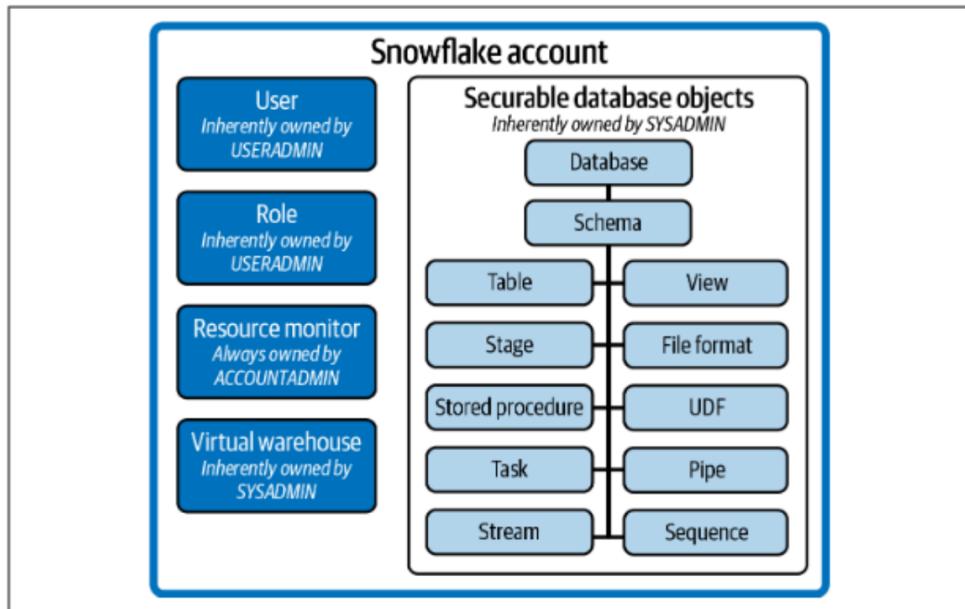


图 5-4. Snowflake 帐户对象



资源监视器始终由 ACCOUNTADMIN 角色拥有。只有 ACCOUNTADMIN 可以创建或删除此对象。ACCOUNTADMIN 不能将此权限授予任何其他角色，但可以授予另一个角色修改已创建的资源监视器的能力。

创建 Snowflake 对象

为了说明 Snowflake 访问控制的工作原理，您的 Snowflake 帐户中必须有 Snowflake 对象进行练习。让我们从创建一些虚拟仓库开始，这些虚拟仓库是执行语句和数据操作语言（DML）操作所必需的。首先，SYSADMIN 角色用于创建新的虚拟仓库。然后，可以使用如图 5-5 所示的命令来确认已创建虚拟仓库：

使用角色 SYSADMIN; 使用 WAREHOUSE COMPUTE_WH; 创建或替换仓库VW1_WH
WAREHOUSE_SIZE='X-SMALL' INITIALLY_SUSPENDED=TRUE;

创建或替换仓库VW2_WH WAREHOUSE_SIZE='X-SMALL' INITIALLY_SUSPENDED=TRUE; 创建或替换仓库VW3_WH WAREHOUSE_SIZE='X-SMALL' INITIALLY_SUSPENDED=TRUE; 展示仓库;

	name	state	type	size	min_cluster_count	max_cluster_count	started_clusters	running	queued	is_default	is_current	auto_suspend
1	COMPUTE_WH	SUSPENDED	STANDARD	X-Small	1	1	0	0	0	Y	N	600
2	VW1_WH	SUSPENDED	STANDARD	X-Small	1	1	0	0	0	N	N	600
3	VW2_WH	SUSPENDED	STANDARD	X-Small	1	1	0	0	0	N	N	600
4	VW3_WH	SUSPENDED	STANDARD	X-Small	1	1	0	0	0	N	Y	600

图 5-5. 新创建的虚拟仓库

接下来，再次使用 SYSADMIN 角色创建两个数据库和三个架构。在创建这些对象之前，请了解该命令如何根据用于发出命令的角色返回不同的结果。之所以出现差异，是因为结果旨在仅返回角色有权访问的数据库。

在创建 Snowflake 账户时，包括三个数据库，每个角色（包括 PUBLIC 角色）都可以使用。因此，使用角色 SYSADMIN 或角色 PUBLIC 查看数据库列表将返回相同的结果，如图 5-6 所示：

使用角色 SYSADMIN;
显示数据库；

现在尝试使用 PUBLIC 角色：

使用 PUBLIC 角色；
显示数据库；

	name	...	is_default	is_current	origin	owner
1	SNOWFLAKE_SAMPLE_DATA	N	N	SFC_SAMPLES.SAMPLE_DATA	ACCOUNTADMIN	

图 5-6. 可用于 PUBLIC 角色的现有 Snowflake 数据库

但是，分配了 ACCOUNTADMIN 角色的用户将看到列出的第四个数据库，如图 5-7 所示。附加数据库是 Snowflake 提供的系统定义的只读共享数据库：

使用角色 ACCOUNTADMIN; 显示
数据库；

	name	...	is_default	is_current	origin	owner
1	SNOWFLAKE		N	N	SNOWFLAKE.ACCOUNT_USAGE	
2	SNOWFLAKE_SAMPLE_DATA		N	N	SFC_SAMPLES.SAMPLE_DATA	ACCOUNTADMIN

图 5-7。可用于 ACCOUNTADMIN 角色的现有 Snowflake 数据库

让我们使用 SYSADMIN 角色来创建数据库和架构。



编写 SQL 命令以创建数据库和架构的顺序很重要。您需要在创建数据库后立即创建架构，否则请使用完全合格的对象名称来创建架构。

对于 DB1，我们将在最后使用完全限定的对象名称创建模式，对于 DB2，我们将在创建数据库后立即创建模式：

```
使用角色 SYSADMIN;
创建或替换数据库 DB1;创建或替换数据库 DB2;创建或
替换 SCHEMA DB2_SCHEMA1;创建或替换 SCHEMA
DB2_SCHEMA2;创建或替换架构 DB1。DB1_SCHEMA1;显示
数据库;
```

结果如图 5-8 所示。

	name	is_default	is_current	origin	owner	...
1	DB1	N	Y		SYSADMIN	
2	DB2	N	N		SYSADMIN	
3	SNOWFLAKE_SAMPLE_DATA	N	N	SFC_SAMPLES.SAMPLE_DATA	ACCOUNTADMIN	

图 5-8。SYSADMIN 角色可用的现有 Snowflake 数据库

资源监视器将在第 8 章中更详细地描述。只有账户管理员才能创建资源监控器。以下代码用于创建资源监视器：

```
使用角色 ACCOUNTADMIN;在 75% 的 DO NOTIFY 上创建或替换为 CREDIT_QUOTA=10000 触发
器的资源监视器MONITOR1_RM
```

```
98% 的人会暂停 105% 的人会
SUSPEND_IMMEDIATE;显示资源监视器;
```

上述代码的结果如图 5-9 所示。

	name	credit_quota	used_credits	remaining_credits	level	frequency
1	MONITOR1_RM	10000.00	0.00	10000.00	null	MONTHLY

图 5-9. 由 ACCOUNTADMIN 角色创建的资源监视器

在本章的后面，我们将详细探讨 Snowflake 用户管理。现在，让我们创建一些信息最少的用户：

```
使用角色 USERADMIN; 创建或替换用户 USER1
LOGIN_NAME=ARNOLD; 创建或替换用户 USER2
LOGIN_NAME=BEATRICE; 创建或替换用户 USER3
LOGIN_NAME=COLLIN; 创建或替换用户 USER4 LOGIN_NAME=DIEDRE;
```

我们能够成功创建用户。让我们看看当我们尝试使用 USERADMIN 角色来显示用户列表时会发生什么情况：

```
使用角色 USERADMIN;
显示用户;
```

结果 5-10 如图 5-10 所示。

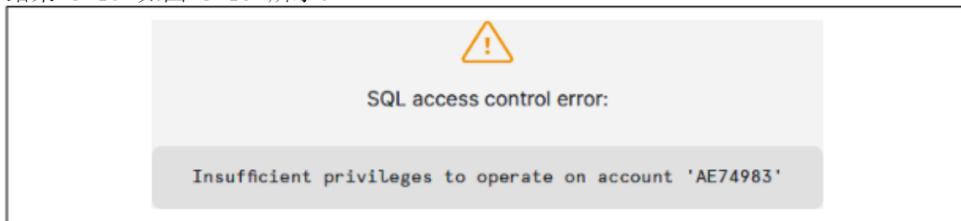


图 5-10. 由于 USERADMIN 的权限不足，因此显示错误

可以使用 USERADMIN 角色上方的层次结构中的角色来查看所有用户的列表，如图 5-11 所示：

```
使用角色 SECURITYADMIN; 显示
用户;
```

	name	created_on	login_name	display_name
1	JKA2022	.945 -0700	JKA2022	JKA2022
2	SNOWFLAKE	.959 -0700	SNOWFLAKE	SNOWFLAKE
3	USER1	.974 -0700	ARNOLD	USER1
4	USER2	.295 -0700	BEATRICE	USER2
5	USER3	.596 -0700	COLLIN	USER3
6	USER4	.917 -0700	DIEDRE	USER4

图 5-11. Snowflake 用户列表

Snowflake 系统定义的角色

分配给用户的 Snowflake 角色是可以为其分配权限的实体。除非特别分配，否则分配的权限不附带将分配的权限授予其他角色的能力。还可以将角色授予其他角色，从而创建角色层次结构。通过这种角色层次结构，特权也被层次结构中特定角色之上的所有角色继承。用户可以拥有多个角色，并且可以在角色之间切换，但在当前 Snowflake 会话中只能有一个角色处于活动状态。如果在活动会话中使用 USE SECONDARY ROLES 选项，则例外。如图 5-12 所示，Snowflake 账户中有少量系统定义的角色。

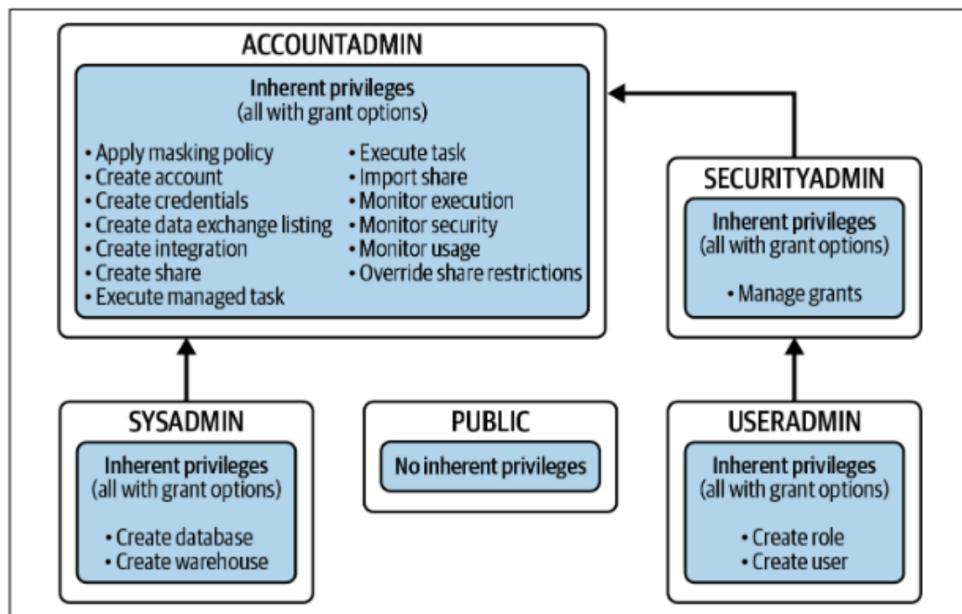


图 5-12. 系统定义的账户角色

如图 5-12 所示，账号管理员（ACCOUNTADMIN）位于系统定义的账号角色的顶层，可以查看和操作账号中的所有对象，但有一个例外。如果对象是由自定义角色创建的，而没有分配的 systemdefined 角色，则账户管理员将无权访问该对象。

ACCOUNTADMIN 角色可以停止任何正在运行的 SQL 语句，并且可以查看和管理 Snowflake 计费。资源监视器的权限对于 ACCOUNTADMIN 角色是唯一的。资源监视器权限附带的任何固有权限均不附带该选项，但 ACCOUNTADMIN 可以将 ALTER RESOURCE MONITOR 权限分配给另一个角色。最佳做法是

将 ACCOUNTADMIN 角色限制为保持对 Snowflake 帐户的控制所需的最小用户数，但不少于两个用户。



Snowflake 没有超级用户或超级角色的概念。对安全对象的所有访问，即使是由帐户管理员访问，都需要在 ACCOUNTADMIN 创建对象本身时显式授予访问权限，或者通过处于更高层次结构的角色中隐式授予访问权限。因此，如果创建自定义角色时未分配给最终导致 ACCOUNTADMIN 角色的层次结构中的其他角色，则 ACCOUNTADMIN 角色将无法访问该角色创建的任何安全对象。

除了 ACCOUNTADMIN 之外，其他系统定义的角色还可以使用分配给它们的固有权限来管理系统中对象的安全性。可以将这些权限授予 Snowflake 中的其他角色（包括自定义角色），以分配管理系统安全性的责任。本章后面将演示 ACCOUNTADMIN 将 Apply Masking Policy 权限分配给自定义角色的示例。

安全管理员（SECURITYADMIN）天生就被赋予了

GRANTS 权限，并且还继承了 USERADMIN 角色的所有权限。用户管理员（USERADMIN）负责创建和管理用户和角色。

系统管理员（SYSADMIN）角色是系统定义的角色，具有在 Snowflake 账户中创建虚拟仓库、数据库和其他对象的权限。由 USERADMIN 创建的最常见角色被分配给 SYSADMIN 角色，从而使系统或账户管理员能够管理由自定义角色创建的任何对象。

PUBLIC 角色会自动授予 Snowflake 账户中的每个角色和每个用户。与任何其他角色一样，PUBLIC 角色可以拥有安全对象。



请务必记住，提供给 PUBLIC 角色的任何权限或对象访问权限都可用于账户中的所有角色和所有用户。

创建自定义角色

创建和管理自定义角色是 Snowflake 中最重要的功能之一。在创建自定义角色之前，应进行规划以设计一个 custom 角色架构，该架构将保护敏感数据，但不会受到不必要的限制。参与规划讨论的应该是数据

管家、治理委员会、了解业务需求的员工以及 IT 专业人员。本章中提供了一个创建自定义角色的方法示例。我采用的方法是将自定义角色划分为功能级自定义角色（包括业务和 IT 角色）和系统级自定义角色（包括服务帐户角色和对象访问角色）。

让我们看看我们为准备创建新的自定义角色而创建的 Snowflake 账户中存在的角色。显示角色的命令将返回角色名称列表以及一些其他信息：

```
使用角色 USERADMIN;
显示角色;
```

在图 5-13 中，对于 USERADMIN 角色，我们看到 PUBLIC 角色的 `is_current` 状态为 Y，`is_inherited` 状态为 Y。这意味着 USERADMIN 角色是当前正在使用的角色，位于层次结构中的 public 角色之上；因此，它将继承分配给 PUBLIC 角色的任何权限。

	name	is_default	is_current	is_inherited
1	ACCOUNTADMIN	Y	N	N
2	ORGADMIN	N	N	N
3	PUBLIC	N	N	Y
4	SECURITYADMIN	N	N	N
5	SYSADMIN	N	N	N
6	USERADMIN	N	Y	N

图 5-13. 将 USERADMIN 角色显示为当前用户

如图 5-14 所示，安全管理员角色继承了 USERADMIN 和 PUBLIC 角色的权限。以下是显示角色的命令：

```
使用角色 SECURITYADMIN;显示
角色;
```

	name	is_default	is_current	is_inherited
1	ACCOUNTADMIN	Y	N	N
2	ORGADMIN	N	N	N
3	PUBLIC	N	N	Y
4	SECURITYADMIN	N	Y	N
5	SYSADMIN	N	N	N
6	USERADMIN	N	N	Y

图 5-14. 将 SECURITYADMIN 显示为当前用户

职能级业务和 IT 角色

您会注意到，在图 5-15 中，决定使用后缀来区分级别，例如 Sr 表示高级分析师，Jr 表示初级分析师。此外，还决定为可能需要不同环境的角色使用前缀，例如用于沙盒的 SBX、用于开发的 DEV 和用于生产的 PRD。

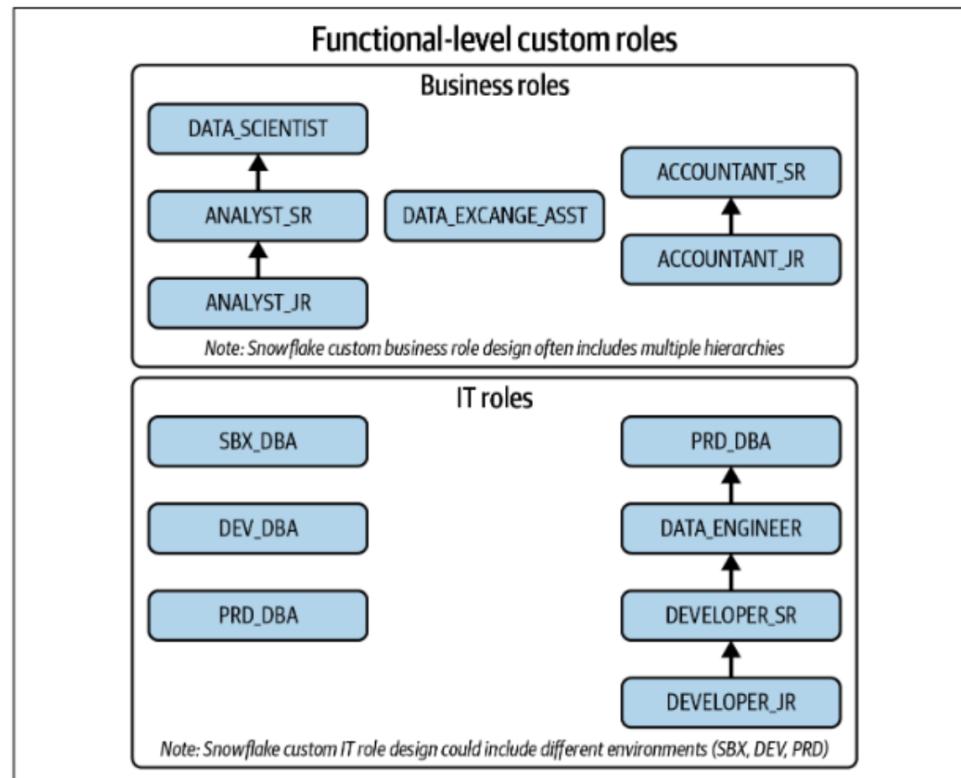


图 5-15. 由 USERADMIN 角色创建的功能级自定义角色示例

我们使用 USERADMIN 角色来创建 10 个功能性自定义角色：

使用角色 USERADMIN；
创建或替换角色 DATA_SCIENTIST；
创建或替换角色 ANALYST_SR；
创建或替换角色 ANALYST_JR；
创建或替换角色 DATA_EXCHANGE_ASST；
创建或替换角色 ACCOUNTANT_SR；
创建或替换角色 ACCOUNTANT_JR；
创建或替换角色 PRD_DBA；
创建或替换角色 DATA_ENGINEER；
创建或替换角色 DEVELOPER_SR；

创建或替换角色 DEVELOPER_SR；

```
创建或替换角色DEVELOPER_JR;  
显示角色;
```

通过使用该命令，您将发现没有一个自定义角色被分配给其他角色。否则，将所有自定义角色分配给层次结构中的另一个角色，并将顶级自定义角色分配给 SYSADMIN 角色或 ACCOUNTADMIN 角色，除非有业务需要隔离自定义角色，否则请务必将所有自定义角色分配给层次结构中的另一个角色。在本章的后面部分，我们将通过将自定义角色分配给系统定义的角色来完成自定义角色的层次结构。

系统级服务账户和对象访问角色

对于系统级自定义角色（参见图 5-16），我创建了两种不同类型的角色。服务账户角色通常是用于加载数据或连接到可视化工具的角色。对象访问角色是将被授予数据访问权限的角色，例如查看特定架构中的数据或将数据插入表的能力。然后，这些对象访问角色将被分配给层次结构中更高级别的其他角色。

首先，让我们使用 USERADMIN 角色创建系统服务账户角色：

```
使用角色 USERADMIN;  
创建或替换 ROLE LOADER; 创建或替换 ROLE  
VISUALIZER; 创建或替换角色报告; 创建或替换  
角色监控;
```

接下来，我们将创建系统对象访问角色：

```
使用角色 USERADMIN; 创建或替换角色  
DB1_SCHEMA1_READONLY; 创建或替换角色DB1_SCHEMA1_ALL;  
创建或替换角色DB2_SCHEMA1_READONLY; 创建或替换角色  
DB2_SCHEMA1_ALL; 创建或替换角色DB2_SCHEMA2_READONLY;  
创建或替换角色DB2_SCHEMA2_ALL;
```

```
创建或替换角色RM1 MODIFY;
```

```
创建或替换角色WH1_USAGE; 创建或替换角色  
WH2_USAGE; 创建或替换角色WH3_USAGE;
```

```
创建或替换角色DB1_MONITOR; 创建或替换角色  
DB2_MONITOR; 创建或替换角色WH1_MONITOR; 创建或替换角色  
WH2_MONITOR; 创建或替换角色WH3_MONITOR; 创建或替换角色RM1_MONITOR;
```

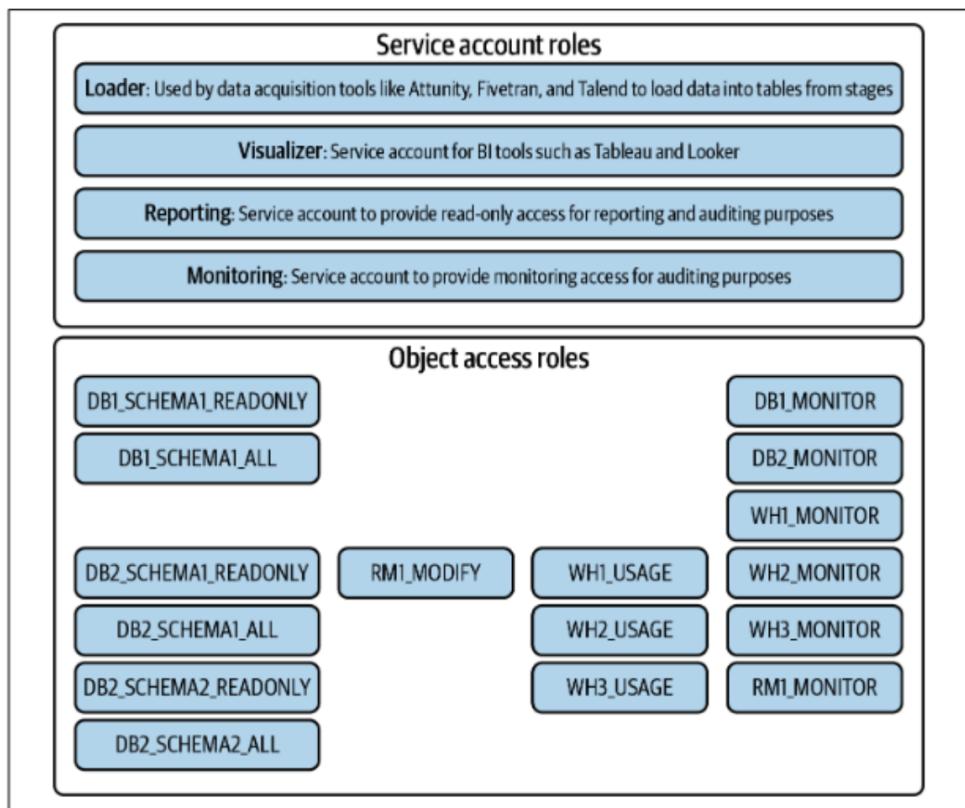


图 5-16. 由 USERADMIN 角色创建的系统级自定义角色示例

角色层次结构分配：将角色分配给其他角色

如前所述，我们希望将对象访问角色分配给角色层次结构中的其他角色。作为图 5-17 的一部分，您将看到还需要将虚拟仓库分配给我们预计需要运行查询能力的任何角色。

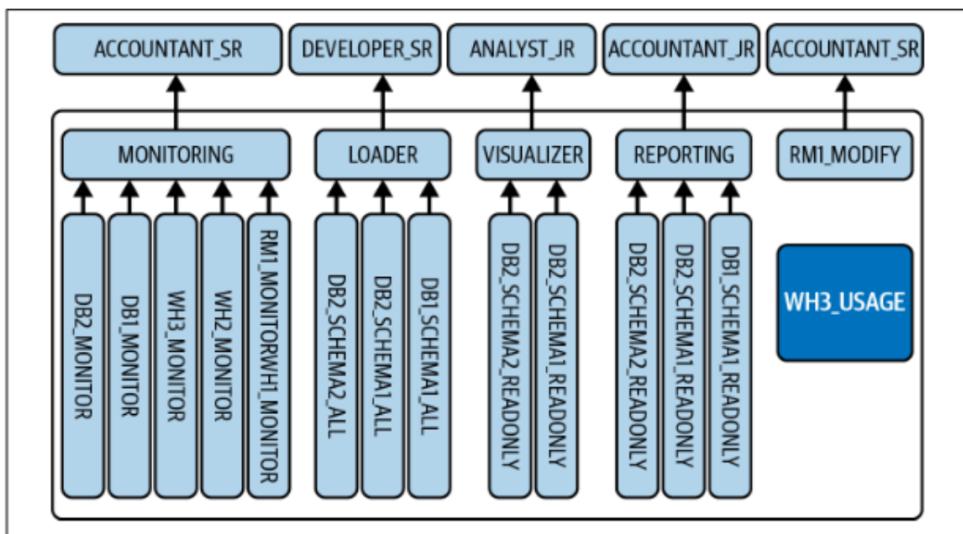


图 5-17. 分配 USERADMIN 角色要授予其他角色的系统级自定义角色

现在，我们将完成系统级角色层次结构分配：

使用角色 USERADMIN;将 ROLE RM1_MONITOR 授予
ROLE MONITORING;将 ROLE WH1_MONITOR 授予 ROLE
MONITORING;将角色WH2_MONITOR授予角色监控;将
ROLE WH3_MONITOR 授予 ROLE MONITORING;将 ROLE
DB1_MONITOR 授予 ROLE MONITORING;将 ROLE
DB2_MONITOR 授予 ROLE MONITORING;将角色
WH3_USAGE授予角色监视；

将 ROLE DB1_SCHEMA1_ALL 授予 ROLE LOADER;将
ROLE DB2_SCHEMA1_ALL 授予 ROLE LOADER;将 ROLE
DB2_SCHEMA2_ALL 授予 ROLE LOADER;将 ROLE
WH3_USAGE 授予 ROLE LOADER;

将 ROLE DB2_SCHEMA1_READONLY 授予 ROLE VISUALIZER;将 ROLE
DB2_SCHEMA2_READONLY 授予 ROLE VISUALIZER;将 ROLE WH3_USAGE
授予 ROLE VISUALIZER.

将角色DB1_SCHEMA1_READONLY授予角色报告;将角色
DB2_SCHEMA1_READONLY授予角色报告;将角色
DB2_SCHEMA2_READONLY授予角色报告。

将角色WH3_USAGE授予角色报告;将角色监控授予角色
ACCOUNTANT_SR;将 ROLE LOADER 授予 ROLE
DEVELOPER_SR;将 ROLE VISUALIZER 授予角色
ANALYST_JR;

将角色报告授予角色 ACCOUNTANT_JR;将 ROLE RM1 MODIFY
授予 ROLE ACCOUNTANT_SR;

完成功能级角色层次结构分配意味着我们还需要将顶级自定义角色分配给 SYSADMIN 或 ACCOUNTADMIN 角色，作为层次结构分配过程的最后一步。和以前一样，我们还希望将虚拟仓库分配给我们预计需要运行查询能力的任何角色。

如图 5-18 所示。

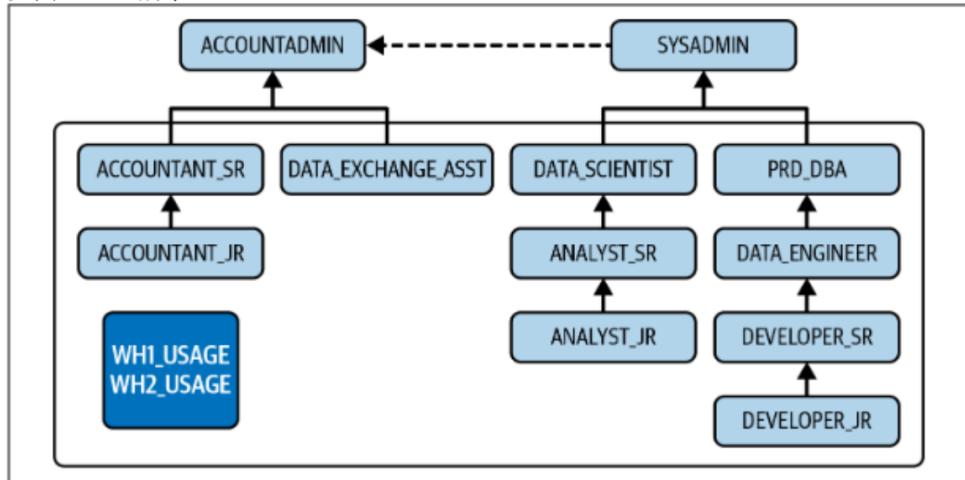


图 5-18. 分配功能级自定义角色，以由 USERADMIN 角色授予其他角色

USERADMIN 角色用于完成功能角色层次结构分配：

使用角色 USERADMIN;将 ROLE ACCOUNTANT_JR 授予 ROLE
ACCOUNTANT_SR;将 ROLE ANALYST_JR 授予 ROLE ANALYST_SR;
将 ROLE ANALYST_SR 授予 ROLE DATA_SCIENTIST;将 ROLE
DEVELOPER_JR 授予 ROLE DEVELOPER_SR;将 ROLE
DEVELOPER_SR 授予 ROLE DATA_ENGINEER;将 ROLE
DATA_ENGINEER 授予 ROLE PRD_DBA;

将角色ACCOUNTANT_SR授予角色 ACCOUNTADMIN;将角色
DATA_EXCHANGE_ASST授予角色 ACCOUNTADMIN;将角色
DATA_SCIENTIST授予角色 SYSADMIN;将角色 PRD_DBA 授予角色
SYSADMIN;

接下来，请务必直接授予 IT 角色，将虚拟仓库VW2_WH 的使用权授予业务角色：

将 ROLE WH1_USAGE 授予 ROLE DEVELOPER_JR;将 ROLE
WH1_USAGE 授予 ROLE DEVELOPER_SR;

将 ROLE WH1_USAGE 授予 ROLE DATA_ENGINEER;将 ROLE
WH1_USAGE 授予 ROLE PRD_DBA;

将 ROLE WH2_USAGE 授予 ROLE ACCOUNTANT_JR; 将 ROLE WH2_USAGE 授予 ROLE ACCOUNTANT_SR; 将 ROLE WH2_USAGE 授予 ROLE DATA_EXCHANGE_ASST; 将 ROLE WH2_USAGE 授予 ROLE ANALYST_JR;

将 ROLE WH2_USAGE 授予 ROLE ANALYST_SR; 将 ROLE WH2_USAGE 授予 ROLE DATA_SCIENTIST;

向角色授予权限

Snowflake 权限是对安全对象定义的访问级别。可以通过使用不同的不同权限来授予访问权限的粒度。如有必要，还可以撤销权限。每个安全对象都有一组特定的特权，可以授予这些特权。对于现有对象，必须对单个对象授予这些权限。为了使授权管理更灵活、更简单，可以在架构中创建的对象上分配未来的授权。我选择在此处为图 5-19 中所示的角色分配以下权限。这些特定权限是只有 ACCOUNTADMIN 才能分配的权限。

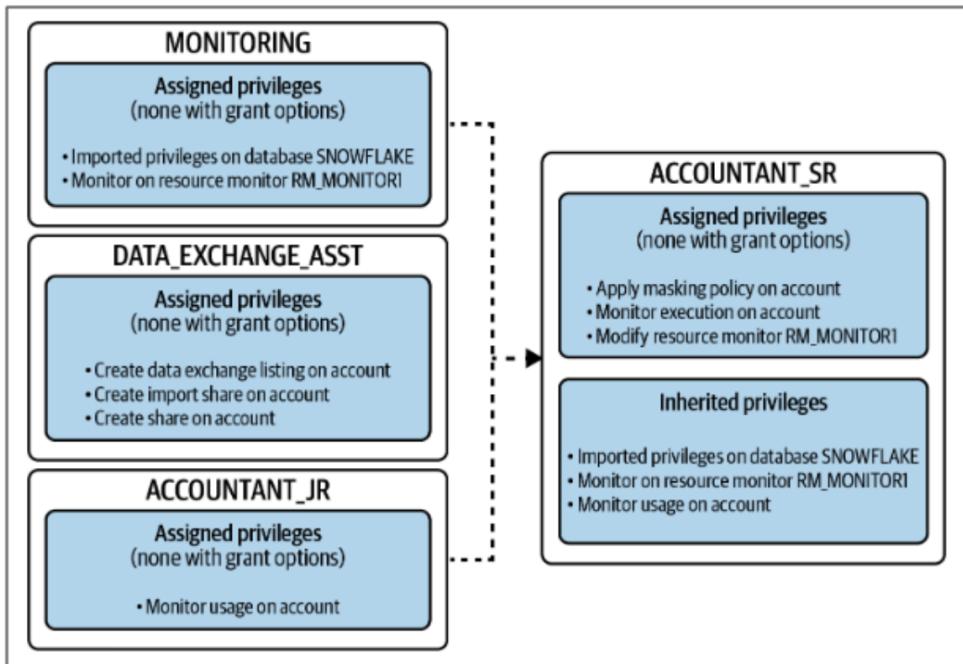


图 5-19. 由 ACCOUNTADMIN 角色授予的 Snowflake 分配的权限

需要 ACCOUNTADMIN 角色才能向功能角色授予直接全局权限。我们还需要使用 ACCOUNTADMIN 角色向只有 ACCOUNTADMIN 才能授予的自定义角色授予权限：

使用角色 ACCOUNTADMIN;将 CREATE DATA EXCHANGE LISTING ON ACCOUNT 授予角色 DATA_EXCHANGE_ASST;将 IMPORT SHARE ON ACCOUNT 授予角色 DATA_EXCHANGE_ASST;将 CREATE SHARE ON ACCOUNT 授予角色 DATA_EXCHANGE_ASST;

将数据库 SNOWFLAKE 的导入权限授予角色 MONITORING;
GRANT MONITOR 对 RESOURCE MONITOR MONITOR1_RM 角色监视;将 MONITOR USAGE ON ACCOUNT 授予角色 ACCOUNTANT_JR;将 APPLY MASKING POLICY ON ACCOUNT 授予角色 ACCOUNTANT_SR;将 MONITOR 执行授予 ACCOUNT 角色 ACCOUNTANT_SR;将 EDIT ON 资源监视器 MONITOR1_RM 授予角色 ACCOUNTANT_SR;

许多不同的自定义角色需要权限才能与对象中的数据进行交互，并且需要能够使用虚拟仓库进行交互。为了能够查看表中的数据，角色需要具有使用表所在的数据库和架构的权限，以及能够对表使用命令。授予这些权限时，将为架构中的任何现有对象分配权限。我们还需要考虑分配权限，以便该角色可以访问将来创建的表。未来的授权只能由 ACCOUNTADMIN 分配;因此，我们必须在后续步骤中分配 Future Grant 访问权限。

以下是已分配权限的汇总列表。所有权限都没有

GRANT 选项。您会注意到，对象监视权限是在数据库级别设置的;因此，该角色将能够监控我们创建的数据库以及层次结构中数据库下的所有对象：

DB1_SCHEMA1_READONLY

- 用法 在数据库 DB1 上
- 架构 DB1 上的 USAGE 方法。DB1_SCHEMA1
- 选择 在架构 DB1 中的所有表上。DB1_SCHEMA1
- 选择 在架构 DB1 中的未来表中。DB1_SCHEMA1

DB2_SCHEMA1_READONLY

- 用法 在数据库 DB2 上
- USAGE 在模式 DB2 上。DB2_SCHEMA1
- 选择 在架构 DB2 中的所有表上。DB2_SCHEMA1

DB2_SCHEMA2_READONLY

- 用法 在数据库 DB2 上
- 用法 在架构 DB2 上。DB2_SCHEMA2
- 对架构 DB2 中的所有表执行 SELECT 操作。DB2_SCHEMA2
- 选择 在模式 DB2 中的未来表中。DB2_SCHEMA1

DB1_SCHEMA1_ALL

- ALL 在架构 DB1 上。DB1_SCHEMA1
- 选择 在架构 DB1 中的未来表中。DB1_SCHEMA1

DB2_SCHEMA1_ALL

- ALL 模式的 DB2 上。DB1_SCHEMA1
- 选择 在模式 DB2 中的未来表中。DB1_SCHEMA1

DB2_SCHEMA2_ALL

- ALL 在架构 DB2 上。DB2_SCHEMA2
- 选择 在模式 DB2 中的未来表中。DB2_SCHEMA2

DB1_MONITOR

- 监控 在数据库 DB1 上

DB2_MONITOR

- 监控 在数据库 DB2 上

WH1_MONITOR

- 监控 在 Virtual Warehouse VW1_WH

WH2_MONITOR

- 监控 在 Virtual Warehouse VW2_WH

WH3_MONITOR

- 监控 在 Virtual Warehouse VW3_WH

WH1_USAGE

- 用法 在 Virtual Warehouse VW1_WH

WH2_USAGE

- 用法 在 Virtual Warehouse VW2_WH

WH3_USAGE

- 用法 在 Virtual Warehouse VW3_WH

请注意，授予权限需要 SYSADMIN 角色，而不是 USERADMIN 角色。使用 SYSADMIN 角色将直接分配的权限授予系统级对象访问角色：

使用角色 SYSADMIN；

将数据库 DB1 的使用权授予角色 DB1_SCHEMA1_READONLY; 将数据库 DB2 的使用权授予

角色 DB2_SCHEMA1_READONLY; 将数据库 DB2 上的使用权授予角色

DB2_SCHEMA2_READONLY; GRANT 对架构 DB1 的使用。DB1_SCHEMA1 角色

DB1_SCHEMA1_READONLY; GRANT 对架构 DB2 的使用。DB2_SCHEMA1 角色

DB2_SCHEMA1_READONLY; GRANT 对架构 DB2 的使用。DB2_SCHEMA2 角色

DB2_SCHEMA2_READONLY;

```
对架构 DB1 中的所有表授予 SELECT 权限。DB1_SCHEMA1
    角色 DB1_SCHEMA1_READONLY;
对架构 DB2 中的所有表授予 SELECT 权限。DB2_SCHEMA1
    角色 DB2_SCHEMA1_READONLY;
对架构 DB2 中的所有表授予 SELECT 权限。DB2_SCHEMA2
    角色 DB1_SCHEMA1_READONLY;

在架构 DB1 上授予 ALL。DB1_SCHEMA1 角色 DB1_SCHEMA1_ALL;在架构 DB2 上
授予 ALL。DB2_SCHEMA1 角色 DB2_SCHEMA1_ALL;在架构 DB2 上授予 ALL。
DB2_SCHEMA2 角色 DB2_SCHEMA2_ALL.
```

将数据库 DB1 上的 MONITOR 授予角色 DB1_MONITOR;将数据库 DB2 上
的 MONITOR 授予角色 DB2_MONITOR;将仓库VW1_WH上的监控器授予角色
WH1_MONITOR;将仓库VW2_WH上的监控器授予角色 WH2_MONITOR;将仓库
VW3_WH上的监视器授予角色 WH3_MONITOR;

将仓库VW1_WH的使用权授予 WH1_USAGE;将仓库VW2_WH的使
用权授予 WH2_USAGE;将仓库VW3_WH的使用权授予
WH3_USAGE;

使用 ACCOUNTADMIN 角色授予直接分配的权限:

```
使用角色 ACCOUNTADMIN;在架构 DB1 中对 FUTURE TABLES 授予 SELECT
权限。DB1_SCHEMA1
    角色 DB1_SCHEMA1_READONLY;
在架构 DB2 中对 FUTURE TABLES 授予 SELECT 权限。DB2_SCHEMA1
    角色 DB2_SCHEMA1_READONLY;
在架构 DB2 中对 FUTURE TABLES 授予 SELECT 权限。DB2_SCHEMA2
角色 DB2_SCHEMA2_READONLY;在架构 DB1 中对 FUTURE TABLES 授予 SELECT 权限。DB1_SCHEMA1 角色
DB1_SCHEMA1_ALL;在架构 DB2 中对 FUTURE TABLES 授予 SELECT 权限。DB2_SCHEMA1 角色
DB2_SCHEMA1_ALL;在架构 DB2 中对 FUTURE TABLES 授予 SELECT 权限。DB2_SCHEMA2 角色
DB2_SCHEMA2_ALL.
```

为用户分配角色

我们在本章的前面创建了四个用户。现在，让我们为这四个用户中的每个用户分配角
色。可以为每个用户分配多个角色，但在任何给定时间只能使用一个角色（即，分配的
角色不能“分层”或组合）。请记住，对于为其分配的任何角色，如果层次结构中其下
有角色，则用户已经继承了该角色。例如，Data Scientist 角色继承了 Analyst Sr
和 Analyst Jr 角色，并将在其账户中看到这些角色。因此，将这两个角色中的任何一个
分配给分配了 Data Scientist 角色的用户都是多余的：

```
使用角色 USERADMIN;将角色 DATA_EXCHANGE_ASST 授予用
户 USER1;将角色 DATA_SCIENTIST 授予用户 USER2;将角
色 ACCOUNTANT_SR 授予用户 USER3;将角色 PRD_DBA 授予
用户 USER4;
```

测试和验证我们的工作

现在是测试和验证我们为建立访问控制安全性而完成的工作的时候了。



使用任何自定义角色运行任何查询时，您需要有一个正在运行的虚拟仓库才能完成查询。如果您在任何时候收到一条错误消息，指出没有正在运行的虚拟仓库，您始终可以使用 `com - mand` 查找该角色的可用虚拟仓库列表，并使用命令来运行虚拟仓库。

例如，角色 `PRD_DBA` 具有分配给该角色的 `VW1_WH` 虚拟仓库和继承的 `VW3_WH` 虚拟仓库。这在图 5-20 中得到了证明。

The screenshot shows two panels of the Snowflake web interface. On the left, the 'Roles' page lists several roles: `PRD_DBA` (selected), `ACCOUNTADMIN`, `ORGADMIN`, `ACCOUNTANT_JR`, `ACCOUNTANT_SR`, and `ANALYST_JR`. On the right, the 'Warehouses' page lists two warehouses: `VW1_WH` and `VW3_WH`.

图 5-20. `PRD_DBA` 角色可用的当前 Snowflake 虚拟仓库

现在让我们测试一下，看看访问是否按预期工作。作为最佳实践，我们只向少数人授予了对 `SNOWFLAKE` 数据库的访问权限。`ACCOUNTADMIN` 角色有权访问 `SNOWFLAKE` 数据库，之前我们使用 `ACCOUNTADMIN` 角色向高级会计师授予访问权限。因此，如果初级会计师角色尝试访问该表，我们预计会返回错误：

```
使用 ROLE ACCOUNTANT_JR; 选择 * FROM SNOWFLAKE. ACCOUNT_USAGE. QUERY_HISTORY 其中  
QUERY_TYPE = 'GRANT';
```

这在图 5-21 中得到了证实。

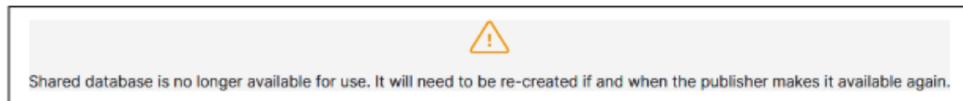


图 5-21. 由于尚未向初级会计角色授予访问权限，因此返回错误



在尝试测试自定义角色之前，您可能需要刷新 Snowsight Web UI 屏幕或注销并重新登录。

如果具有高级会计角色的用户运行相同的查询，如以下代码所示，他们将收到如图 5-22 所示的查询结果：

```
使用 ROLE ACCOUNTANT_SR; 使用 WAREHOUSE VW2_WH; 选择 * FROM SNOWFLAKE. ACCOUNT_USAGE。  
QUERY_HISTORY 其中 QUERY_TYPE =' GRANT' ;
```

	QUERY_TEXT	... DATABASE_ID	DATABASE_NAME	SCHEMA_ID	SCHEMA_NAME	QUERY_TYPE
1	GRANT MONITOR ON DATABASE DB2 TO ROLE DB2_MONITOR;	95	DB1	223	DB1_SCHEMA1	GRANT
2	GRANT SELECT ON FUTURE TABLES IN SCHEMA DB2.DB2_SCHEMA1 TO ROLE DB2_SCHEMA1;	95	DB1	223	DB1_SCHEMA1	GRANT
3	GRANT SELECT ON FUTURE TABLES IN SCHEMA DB1.DB1_SCHEMA1 TO ROLE DB1_SCHEMA1;	95	DB1	223	DB1_SCHEMA1	GRANT
4	GRANT SELECT ON FUTURE TABLES IN SCHEMA DB2.DB2_SCHEMA1 TO ROLE DB2_SCHEMA1;	95	DB1	223	DB1_SCHEMA1	GRANT

图 5-22. 显示结果，因为高级会计师已被授予访问权限

目前，我们的 Snowflake 帐户没有我们创建的表。每当 SYSAD - MIN 角色创建新表时，它必须为其他角色分配该表的必要权限，否则他们将无法访问该表。但是，我们之前使用了 future grants 选项来授予对我们在三个架构中创建的任何未来对象（如表）的访问权限。因此，无需对新创建的表分配权限。

让我们测试一下，看看我们分配的 future grants 权限是否会按预期工作。

首先，您可以看到 SYSADMIN 角色可以访问两个数据库：

```
使用角色 SYSADMIN;  
显示数据库;
```

现在，您可以看到 DB1 模式中当前不存在任何表：

```
使用 SCHEMA DB1_SCHEMA1; 显  
示表格;
```

让我们创建一个简单的表，并确认该表已创建，如图 5-23 所示：

```
创建或替换表 DB1. DB1_SCHEMA1. TABLE1 (一个 varchar) ;  
INSERT INTO TABLE1 值 ('A') ;  
显示表格;
```

	name	database_name	schema_name	kind
1	TABLE1	DB1	DB1_SCHEMA1	TABLE

图 5-23。确认表已创建

接下来，我们将测试 REPORTING 角色是否可以访问我们刚刚创建的表：

使用角色报告；使用 WAREHOUSE VW3_WH；从 DB1 中选择 *。DB1_SCHEMA1。表 1；

根据我们分配给角色的未来授予权限，然后分配给 REPORTING 角色，我们预计 REPORTING 角色应该能够访问该表。这在图 5-24 中得到了证实。

	A	...
1	A	

图 5-24. 确认 REPORTING 角色可以访问新创建的表

我们没有授予 DB1 对 VISUALIZER 角色的访问权限，因此该角色将无法查看表，如图 5-25 所示：

使用 ROLE VISUALIZER；从 DB1 中选择 *。DB1_SCHEMA1。表 1；



图 5-25。VISUALIZER 角色无权访问新创建的表

以下是您可以自行尝试的一些其他查询，它们将为您提供一些有用的信息：

使用 ROLE ACCOUNTANT_SR；使用 WAREHOUSE VW3_WH；选择 * FROM SNOWFLAKE。ACCOUNT_USAGE。GRANTS_TO_USERS；SHOW GRANTS ON ACCOUNT (显示账户赠款)；在数据库 DB1 上显示授权；显示角色 ANALYST_SR 的授予；在数据库 DB1 中显示未来的授权；在架构 DB1 中显示 FUTURE GRANTS。DB1_SCHEMA1；

任何全局权限、帐户对象的权限和架构的权限都可以从角色中撤销。例如，我们将让账户管理员向初级分析师授予一个角色，然后 USERADMIN 将撤销该角色：

```
使用角色 ACCOUNTADMIN;将 MONITOR USAGE ON ACCOUNT 授予角色  
ANALYST_JR;使用角色 USERADMIN;从角色 ANALYST_JR 撤销 MONITOR  
对帐户的使用;
```

在会话期间，如果为用户分配了多个角色，则可以更改其角色。当用户尝试对对象执行操作时，Snowflake 会将完成操作所需的权限与当前角色继承的任何权限或本身授予或分配的任何权限进行比较。如果会话中的角色具有必要的权限，则允许该操作。

用户管理

Snowflake 中的 User 对象存储有关用户的所有信息，包括其登录名、密码和默认值。Snowflake 用户可以是个人或程序。从前面的讨论中，我们知道 Snowflake 用户是由 USERADMIN 系统定义的角色创建和管理的。用户名是必需的，并且在创建用户时应该是唯一的。尽管所有其他属性都是可选的，但最佳做法是包含其中的许多属性。至少，您应该包含一些基本详细信息并分配一个初始密码，您要求用户在下次登录时更改该密码。下面是一个示例：

```
使用角色 USERADMIN;  
创建或替换用户 USER10  
密码='123' LOGIN_NAME = ABARNETT  
DISPLAY_NAME = AMY FIRST_NAME = AMY  
LAST_NAME = BARNETT 电子邮件 =  
'ABARNETT@COMPANY.COM'  
MUST_CHANGE_PASSWORD=TRUE;
```

您可以使用该命令添加新属性或更改现有用户属性。例如，您可能有一个用户，您希望向其授予对 Snowflake 账户的临时访问权限。这可以通过为用户设置过期时间来实现：

```
使用角色 USERADMIN;更改用户 USER10 设置  
DAYS_TO_EXPIRY = 30;
```

为用户添加默认值会让用户更轻松。例如，您可以使用户具有默认角色或默认虚拟仓库。



为用户添加默认虚拟仓库不会验证虚拟仓库是否存在。我们从未创建过WAREHOUSE52，但将该虚拟仓库分配为默认仓库的代码将成功执行，而不会出现错误或警告：

```
使用角色 USERADMIN;更改用户 USER10 SET  
DEFAULT_WAREHOUSE=WAREHOUSE52_WH;
```

我们还可以尝试为用户分配一个确实存在的默认角色：

```
使用角色 USERADMIN;更改用户 USER10 SET  
DEFAULT_ROLE=IMAGINARY_ROLE;
```

但是，如果我们之前没有向用户授予该角色，则无法将其设为默认角色。换句话说，设置默认角色不会将角色分配给用户。同样的事情也适用于对数据库的访问。当我们尝试分配以前未分配给用户的默认值时，不会发出警告，但命令实际上并未成功执行。

要亲自查看，请注销您的 Snowflake 帐户，然后以 USER10 身份登录。请记住，当我们创建用户 ABARNETT 时，我们要求用户重置密码。因此，您需要为 USER10 创建一个新的可接受密码（图 5-26 显示了创建新密码的 UI）。

The screenshot shows a password creation interface. At the top is a blue flower-like logo. Below it, the text "Password expired" is displayed, followed by "Your password for **ABARNETT** has expired". A large input field is labeled "New password". Below the input field is a note: "Your password must be 8 - 256 characters and contain at least 1 number(s), 0 special character(s), 1 uppercase and 1 lowercase letter(s)". Another input field is labeled "Confirm password". At the bottom is a large blue "Submit" button.

图 5-26。首次登录时，USER10 需要新密码

当您创建新密码并尝试登录时，您将收到错误 message，因为 ABARNETT 的默认角色集尚未授予用户。因此，登录将失败。请继续以您自己的身份重新登录，我们将进行必要的更正。

让我们为 USER10 设置一些正确的默认值，如图 5-27 所示。完成后，我们将注销我们的 admin 帐户，并以 USER10 身份重新登录：

```
使用角色 USERADMIN;将角色ACCOUNTANT_SR授予用户 USER10;ALTER USER USER10 设  
置 DEFAULT_NAMESPACE=SNOWFLAKE。ACCOUNT_USAGE;更改用户 USER10 SET  
DEFAULT_WAREHOUSE=VW2_WH;更改用户 USER10 SET DEFAULT_ROLE =  
ACCOUNTANT_SR;ALTER USER USER10 UNSET DEFAULT_WAREHOUSE;
```

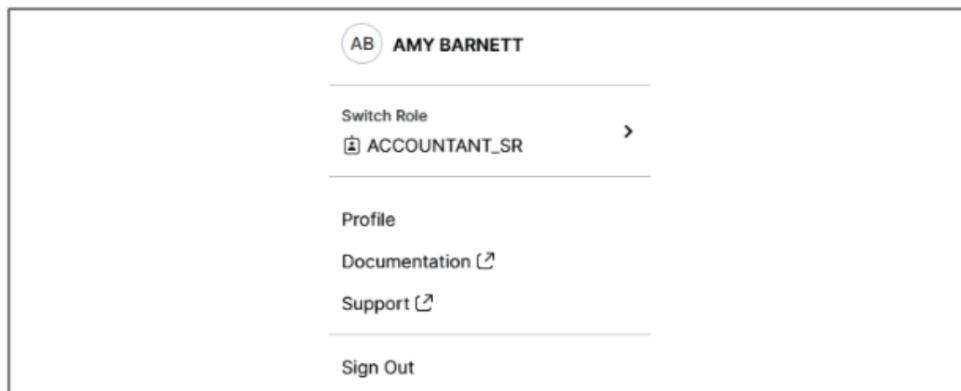


图 5-27. USER10 已被授予 ACCOUNTANT_SR 的默认角色

一个常见的用户管理问题是用户无法登录其账户。Snowflake 系统将锁定连续 5 次尝试后未能成功登录的用户。Snowflake 系统将在 15 分钟后自动清除锁。如果用户无法等待，则可以立即重置计时器：

```
使用角色 USERADMIN;更改用户 USER10 设置  
MINS_TO_UNLOCK=0;
```

请注意，我们用来创建密码的原始方法将不起作用，因为 123 不是一个可接受的密码：USE ROLE USERADMIN ;

```
ALTER USER USER10 SET PASSWORD = '123'  
MUST_CHANGE_PASSWORD = TRUE;
```

如果我们尝试使用 123 密码，会返回一个错误，如图 5-28 所示。



图 5-28。密码被拒绝，因为未达到最小长度

USERADMIN 提交的用于重置忘记密码的密码长度必须至少为 8 个字符，并且至少包含一个数字、一个大写字母和一个小写字母：

```
使用角色 USERADMIN;ALTER USER USER10 SET PASSWORD  
= '123456Aa' MUST_CHANGE_PASSWORD = TRUE;
```

请记住，由于 USERADMIN 角色汇总到 SECURITYADMIN 角色，而 SECURITYADMIN 角色本身又汇总到 ACCOUNTADMIN 角色，因此两个更高的角色也可以运行 USERADMIN 角色可以运行的任何命令。下面是一个示例。

```
使用角色 SECURITYADMIN;更改用户 USER10 设置密码 =  
'123456Bb' MUST_CHANGE_PASSWORD = TRUE;
```

有时可能需要中止用户当前正在运行的查询并阻止该用户运行任何新查询。要完成此操作并立即将用户锁定在 Snowflake 之外，请使用以下命令：

```
使用角色 USERADMIN;  
更改用户 USER10 设置 DISABLED = TRUE;
```

要描述单个用户并获取所有用户的属性值和默认值的列表，请使用以下命令：

```
使用角色 USERADMIN;  
描述 USER USER10;
```

结果如图 5-29 所示。

property	value	...	default	description
1 NAME	USER10		null	Name
2 COMMENT	null		null	user comment associated to an object in the dictionary
3 DISPLAY_NAME	AMY		null	Display name of the associated object
4 LOGIN_NAME	ABARNETT		null	Login name of the user
5 FIRST_NAME	AMY		null	First name of the user
6 MIDDLE_NAME	null		null	Middle name of the user
7 LAST_NAME	BARNETT		null	Last name of the user
8 EMAIL	ABARNETT@COMPANY.COM		null	Email address of the user
9 PASSWORD	*****		null	Password of the user
10 MUST_CHANGE_PASSWORD	true		false	User must change the password
11 DISABLED	true		false	Whether the user is disabled
12 SNOWFLAKE_LOCK	false		false	Whether the user or account is locked by Snowflake
13 SNOWFLAKE_SUPPORT	false		false	Snowflake Support is allowed to use the user or account
14 DAYS_TO_EXPIRY	29.990625		null	User record will be treated as expired after specified number of days

图 5-29. USER10 的描述

要将用户的某个属性值重置回默认值，可以使用以下命令：

```
使用角色 USERADMIN;ALTER USER USER10 设置  
DEFAULT_WAREHOUSE = DEFAULT;
```

```
使用角色 USERADMIN;  
描述 USER USER10;
```

我们之前发现，SECURITYADMIN 角色（而不是 USERADMIN 角色）具有查看所有用户列表的固有权限。相关代码如下，结果如图 5-30 所示：

```
使用角色 SECURITYADMIN;显示  
用户;
```

	name	created_on	login_name	display_name
1	JKA2022	.945 -0700	JKA2022	JKA2022
2	SNOWFLAKE	.959 -0700	SNOWFLAKE	SNOWFLAKE
3	USER1	.092 -0700	ARNOLD	USER1
4	USER10	.785 -0700	ABARNETT	AMY
5	USER2	.426 -0700	BEATRICE	USER2
6	USER3	.914 -0700	COLLIN	USER3
7	USER4	.244 -0700	DIEDRE	USER4

图 5-30. Snowflake 账户中的用户列表

您会注意到，用户列表包括设置账户的初始用户和您创建的所有用户。此外，该帐户还附带了一个用户。SNOWFLAKE 用户是一个特殊用户，只有在需要解决帐户问题时，只有在 ACCOUNTADMIN 许可下，Snowflake 备份端口才会使用该用户。



可以删除 SNOWFLAKE 用户，但删除后，您不能简单地创建另一个可用于支持的 SNOWFLAKE 用户。因此，强烈建议您不要删除此用户。

使用该命令时支持通配符。您可以使用带有下划线的 com - mand 来匹配任何单个字符，使用带有百分号来匹配任何零个或多个字符的序列。下面是一个示例：

```
使用角色 SECURITYADMIN;显示  
用户，如 'USER%';
```

正如可以创建用户一样，也可以删除用户。请注意，在删除用户之前，无需对任何操作执行任何操作，例如撤销其角色。只需发出 DROP 命令：

```
使用角色 USERADMIN;  
删除用户 USER10;
```



ACCOUNTADMIN 可以通过查询 SNOWFLAKE 来访问所有用户的列表。ACCOUNT_USAGE。USERS 表，包括已删除的用户。

到目前为止，我们一直在使用 Snowflake 工作表进行用户管理。我们还有另一个地方来管理用户。导航到 主 菜单 通过单击 主页 图标。一旦进入主菜单，将您的角色更改为SECURITYADMIN，然后点击管理员→用户和角色。您将看到一个包含 6 个用户的列表，包括您的用户账户、SNOWFLAKE 用户和用户 1 到 4。如果单击最右侧的省略号，对于 USER4，您将看到一个选项列表（如图 5-31 所示）。



图 5-31. 用于管理用户的菜单选项

您会注意到向用户授予角色或者禁用或删除用户是多么容易。让我们点击 Edit 查看 Edit User（编辑用户）屏幕（如图 5-32 所示）。

Edit User

USER4 as SECURITYADMIN

User Name	Email
<input type="text" value="USER4"/>	<input type="text"/>
Comment (optional)	
<input type="text" value=" "/>	
Advanced User Options ^	
Login Name	Display Name
<input type="text" value="DIEDRE"/>	<input type="text" value="USER4"/>
First Name	Last Name
<input type="text"/>	<input type="text"/>
Default Role	Default Warehouse
<input type="button" value="Select a role"/>	<input type="button" value="Select a warehouse"/>
Default Namespace	
<input type="text" value="<db_name>.<schema_name>"/>	

图 5-32。Edit User（编辑用户）屏幕

在此屏幕中，您可以分配默认角色和/或默认虚拟仓库。设置默认角色和虚拟仓库将确定在用户启动 Snowflake 会话时与用户关联的角色和虚拟仓库。但是，用户可以随时更改角色或选择不同的虚拟仓库。

角色管理

除了从 管理员 菜单管理用户外，您还可以管理角色（如图 5-33 所示）。

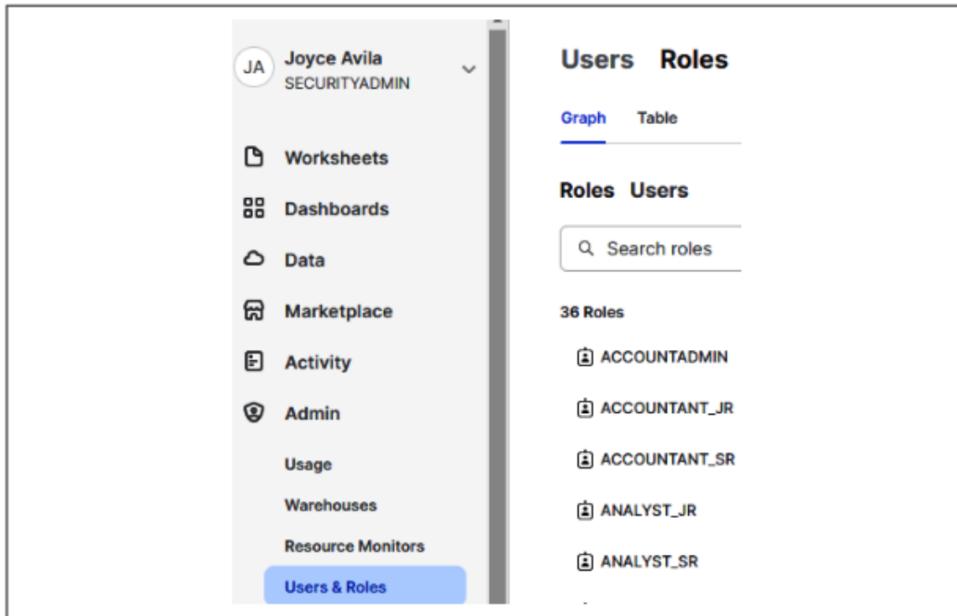


图 5-33。Admin 菜单中的角色管理

在 Roles 子菜单中，您可以选择通过图形或表格管理角色。

Graph（图形）视图特别有用（如图 5-34 所示）。

花一些时间浏览 Roles 子菜单的 Graph（图形）和 Table（表）部分。您会注意到，您可以编辑已创建的角色。此外，屏幕右上角还有一个 + 角色按钮，您可以在其中创建新角色。

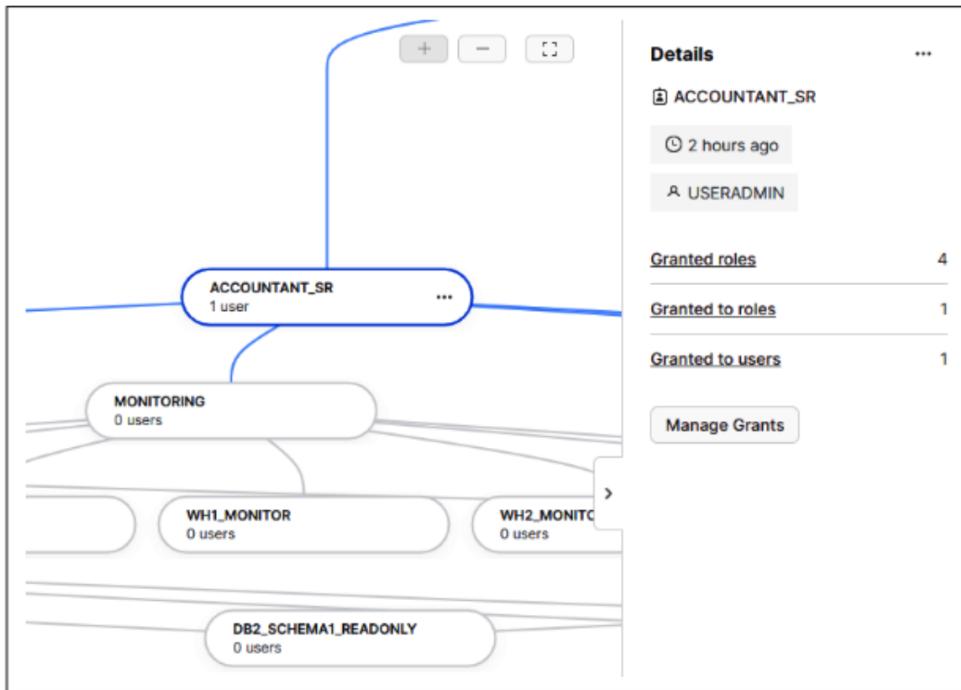


图 5-34。ACCOUNTANT_SR 角色的 Graph 视图和详细信息

前面提到过，可以为任何用户指定默认角色。当用户首次登录时，其主要角色是分配给他们的默认角色。用户在会话中的当前角色被视为其主要角色。

如果为 Snowflake 用户分配了多个角色，他们可以在角色之间切换。只有主要角色和次要角色：任何未用作主要角色的角色都是次要角色。

辅助角色可以以不同的方式处理。一种方法是保持所有角色的 separate，以便用户一次只能使用一个角色。另一种方法是将 secondary 角色授予 primary 角色，实际上是为使用 primary 角色的每个人创建一个特定 secondary 角色到 primary 角色的持久访问分层。我们将在第 7 章中看到这方面的一个例子。相反，如果某些辅助角色应仅授予选定的用户，则可以通过使用 USE SECONDARY ROLES 语句来实现所有 secondary 角色的基于会话的访问分层。

Snowflake 多账户策略

在本章中，我们探讨了如何管理单个 Snowflake 帐户。然而，在很多时候，制定多账户策略来利用账户分离是有意义的。需要多个账户的一个用例是针对不同的环境（例如开发、测试和生产）使用不同的账户。虽然当然可以使用单个账户来完成环境分离，但选择多账户方法意味着不同的环境可以选择不同的安全功能。这样就可以以不同的方式平衡每个账户的生产力与安全风险的处理方式。还可以通过仅将更高级的功能集用于生产而不是开发或测试来节省资金。这在使用多账户策略时是可能的。

有时，公司面临监管限制，无法在一个子公司中使用特定的云提供商或地区。在这种情况下，可以为该特定子公司创建一个新的 Snowflake 帐户，从而可以选择可接受的云提供商和区域。其他时候，出于许多不同的战略原因，会选择多账户。当今的许多组织都是全球性的，需要将其数据位于特定地区的地理位置。有时，组织会选择多云方法来减少对单个供应商的依赖，或者更容易与在不同云提供商上运营的被收购公司合并。

Snowflake 使公司能够通过 Snowflake 组织对象和 ORGADMIN 角色轻松采用多账户策略。Snowflake 组织简化了账户管理和计费、数据库复制和故障转移/故障恢复、安全数据共享以及其他管理任务。



ORGADMIN 可以创建新的 Snowflake 帐户，并可以查看帐户属性。
但是，默认情况下，ORGADMIN 无权访问帐户数据。

DNS 更改大约需要 30 秒才能传播，然后您才能访问新创建的 Snowflake 帐户。默认情况下，Snowflake 组织中的账户数量上限不能超过 25 个；但是，您可以联系 Snowflake 支持来提高限制。

使用 SCIM 管理用户和组

Snowflake 支持 SCIM 2.0，可与身份提供商 Okta 和 Microsoft Azure AD 以及其他需要自定义的身份提供商集成。



特定的 Snowflake SCIM 角色必须拥有从身份提供商导入的任何用户和角色。如果 Snowflake SCIM 角色不拥有导入的用户或角色，则身份提供商中的更新将不会同步到 Snowflake。此外，如果使用身份提供商，请确保不要在 Snowflake 中更改用户和组，因为直接在 Snowflake 中进行的这些更改不会同步回身份提供商。

代码清理

让我们执行代码清理，以便您可以删除 Snowflake 账户中的对象，为编写另一个章节示例做准备。

请注意，在删除数据库之前，无需删除数据库下层次结构中的对象，也无需在删除虚拟仓库之前撤销虚拟仓库使用权限：

```
使用角色 SYSADMIN;
删除数据库 DB1;
删除数据库 DB2;
显示数据库;DROP WAREHOUSE VW1_WH;DROP WAREHOUSE VW2_WH;DROP WAREHOUSE VW3_WH;
展示仓库;
```

正如 ACCOUNTADMIN 必须创建资源监视器一样，必须使用相同的角色来删除资源监视器。删除资源监控器并使用 SHOW 命令后，您将看到该账户上不再有资源监控器：

```
使用角色 ACCOUNTADMIN;删除资源监视器
MONITOR1_RM;显示资源监视器;
```

接下来，USERADMIN 角色将需要删除已创建的所有自定义角色：

```
使用角色 USERADMIN;DROP ROLE DATA_SCIENTIST;DROP ROLE ANALYST_SR;DROP ROLE ANALYST_JR;DROP
ROLE DATA_EXCHANGE_ASST;DROP ROLE ACCOUNTANT_SR;DROP ROLE ACCOUNTANT_JR;DROP ROLE
PRD_DBA;DROP ROLE DATA_ENGINEER;DROP ROLE DEVELOPER_SR;DROP ROLE DEVELOPER_JR;DROP 角色加载器;DROP 角色可视化器;DROP 角色报告;DROP 角色监控;DROP ROLE RM1 MODIFY;DROP ROLE
WH1_USAGE;DROP ROLE WH2_USAGE;DROP ROLE WH3_USAGE;DROP ROLE DB1_MONITOR;DROP ROLE
DB2_MONITOR;DROP ROLE WH1_MONITOR;DROP ROLE WH2_MONITOR;DROP ROLE WH3_MONITOR;DROP ROLE
RM1_MONITOR;DROP ROLE DB1_SCHEMA1_READONLY;DROP ROLE DB1_SCHEMA1_ALL;DROP ROLE
DB2_SCHEMA1_READONLY;DROP ROLE DB2_SCHEMA1_ALL;DROP ROLE DB2_SCHEMA2_READONLY;DROP ROLE
DB2_SCHEMA2_ALL;显示角色;
```

现在我们可以删除我们创建的用户：

```
使用角色 USERADMIN;
删除用户 USER1;丢弃用户 USER2;删除 用户 USER3;删除用户 USER4;
```

我们可以确认所有用户现在都已被删除：

```
使用角色 SECURITYADMIN;显示  
用户;
```

总结

在本章中，我们了解了 Snowflake 的访问控制模型。通过许多实践示例，我们创建了安全对象和自定义角色，分配了角色层次结构，并授予了角色特权。之后，我们将角色分配给用户，然后测试和验证我们的工作，然后执行用户管理并查看我们可以在 Snowsight Web UI 中的哪些位置执行角色管理。

在整个过程中，我们还了解了使用基于角色的访问控制（RBAC）和自主访问控制（DAC）时的一些最佳实践。在以后的章节示例中，您将注意到我们如何使用其中一些最佳实践，例如确保使用标准 SYSADMIN 角色创建新的安全对象，除非为此目的创建了自定义角色并用于此目的。

在下一章中，我们将首先使用 SYSADMIN 角色在 Snowsight 中创建一些对象，作为准备工作的一部分。我们还将介绍 SnowSQL，即 Snowflake 的命令行界面（CLI）。使用 SnowSQL CLI，我们必须使用 SQL 为角色设置上下文，因为 SnowSQL 中没有像 Snowsight 中那样带有下拉菜单的直观 Web 界面。简单直观的 Snowsight Web 用户界面是 Snowflake 用户可能花费大部分时间的地方，但有时需要 SnowSQL，通常是当您专注于加载和卸载数据时。

知识检查

以下问题基于本章中包含的信息：

1. 您如何定义固有特权？请举几个例子。
2. 您能否命名所有 Snowflake 系统定义的账户角色？在角色层次结构中，大多数自定义角色应分配给哪个系统定义的角色？
3. 为用户分配默认值（例如默认角色或默认虚拟仓库）时，需要牢记哪些重要注意事项？
4. ACCOUNTADMIN 角色可以做而其他角色永远不能做的一件事是什么？

5. 您能否说出角色查看表内容所需的权限？
6. 帐户中包含的 SNOWFLAKE 数据库有什么独特之处？

7. 一个实体可以创建多少个 Snowflake 帐户？

8. 特权附带选项是什么意思？
9. 在什么情况下，用户执行 或 SHOW DATABASES 命令会得到不同的结果？
10. Snowflake 有超级用户或超级角色的概念吗？解释。

这些问题的答案可在附录 A 中找到。

数据加载和卸载

数据工程师负责管理来自各种不同来源的原始数据集的摄取和转换，以最终用户所需的状态提供数据，从而获得可操作的见解。本章提供了任何人都可以使用的基础知识，以了解有关如何在 Snowflake 云数据平台上实现最佳数据工程结果的更多信息。

本章首先总结了 Snowflake 数据加载和卸载的基本概念，包括数据类型、压缩方法、文件格式类型和 Snowflake 阶段。在数据加载工具部分，我们将深入学习如何在 Snowflake 工作表中使用 SQL 插入语句来加载结构化和半结构化数据。我们还将学习如何使用 Web UI 加载数据向导和 SnowSQL 命令行界面（CLI）加载数据。对于更自动化的数据加载方法，我们可以使用数据管道、连接器和第三方工具。我们将在本章后面探讨其中一些自动数据加载选项。此外，我们还将研究数据加载的一些替代方案，例如在外部舞台上创建材料化视图和访问共享数据。最后，我们将讨论从 Snowflake 卸载数据文件。

对于我们的动手示例，我们将在 Snowsight（默认 Web UI）中完成大部分工作。但是，当我们需要下载 SnowSQL 时，我们需要短暂切换到 Snowflake Classic 控制台才能使用 Load Data 向导。在深入研究之前，我们首先要了解本章中动手示例所需的准备工作。

准备工作

创建一个标题为 Chapter6 Data Loading and Unloading 的新工作表。如果您在创建新工作表时需要帮助，请参阅第 8 页上的“导航门控 Snowsight 工作表”。要设置工作表上下文，请确保您使用的是 SYSADMIN 角色和 COMPUTE_WH 虚拟仓库。

在本章中，我们所有的工作都将在一个数据库中完成。对于每种上传类型，我们将创建一个单独的 Schema。此外，我们将在所有声明中包括评论。从 Snowsight Web UI 工作表中执行以下 SQL 语句：

```
使用角色 SYSADMIN; 使用 WAREHOUSE  
COMPUTE_WH; 创建或替换数据库 DEMO6_DB
```

```
COMMENT = “所有第 6 章示例的数据库”;CREATE OR REPLACE SCHEMA WS COMMENT = “工作表插入  
示例的架构”;CREATE OR REPLACE SCHEMA UI COMMENT = “用于 Web UI 上传的 Schema”;CREATE  
OR REPLACE SCHEMA SNOW COMMENT = “Schema for SnowSQL Loads”;创建或替换仓库 LOAD_WH
```

```
COMMENT = “CH 6 仓库加载示例”;
```

您还需要一个 CSV 文件，以便使用 Web UI Load Data 向导和 SnowSQL CLI 上传。GitHub 上提供了本章的文件，包括 CSV 文件，您可以下载和使用。或者，您也可以使用表 6-1 中的数据创建自己的文件。您可以将表另存为名为 TABLE20 的 CSV 逗号分隔文件。

表 6-1. 用于本章动手练习的 CSV 文件的数据

ID	NAME	_NAME	城市
-1	安东尼·罗宾逊	亚特兰大	——
2	佩吉·马蒂森	伯明翰	——
3	Marshall Baker	芝加哥	——
4	凯文·克莱恩	丹佛	——

数据加载和卸载的基础知识

在介绍数据加载和卸载的动手示例之前，我们需要回顾几个重要概念。在本节中，我们将花一些时间了解半结构化数据的数据类型。我们还将了解 Snowflake 支持的文件格式类型和数据文件压缩方法。此外，我们还将讨论数据加载选项频率的差异，并回顾 Snowflake 阶段以及我们如何在 SQL 语句中引用不同类型的阶段。最后，我们将记下一些适用于数据加载的不同数据源。

数据类型

第 4 章介绍了 Snowflake 数据类型，其中我们深入探讨了支持结构化数据的 Snowflake 数据类型。在本章中，我们将对结构化数据使用多种数据类型，并将更深入地研究 和 的半结构化数据类型。

半结构化数据类型

Snowflake 为加载半结构化数据提供了灵活的架构数据类型。在加载半结构化数据之前不需要转换，因为 Snowflake 会自动转换半结构化数据，以允许以完全关系的方式对数据进行 SQL 查询。

三种 Snowflake 数据类型用于支持半结构化数据类型：、

OBJECT 和 。有趣的是，该数据类型可以存储任何其他 Snowflake 数据类型的值，包括 和 。因此，被视为通用数据类型。



数据类型的大小限制为 16 MB 的压缩数据。因此，如果存储在数据类型列中的半结构化数据包括数组、带字符串的 num 或日期和时间戳，建议您将对象和键数据展平为单独的关系列。这对于加载到数据类型中的日期和时间戳尤其重要，因为这些数据类型最终会作为字符串存储在 VARIANT 列中。

如果您尚不确定要对加载到 Snowflake 中的半结构化数据执行哪些类型的操作，则 Snowflake 数据类型是一个不错的选择。

Snowflake 数据类型表示键值对的集合，其中键是非空字符串，值是数据类型。Snowflake ARRAY 数据类型表示任意大小的密集或稀疏数组，其中索引是非负整数，值是数据类型。

文件格式

文件格式是可与命令一起使用的 Snowflake 对象。文件 for - mat 使用压缩和文件类型等参数定义数据文件。它还定义了格式选项，例如 CSV 文件的 trim space 和字段分隔符，以及 JSON 文件的 strip 外部数组。本节稍后将介绍 CSV 和 JSON 文件的不同文件格式参数列表。

Snowflake 支持的数据卸载文件格式类型包括用于半结构化数据的 JSON 和 Parquet，以及用于结构化数据的分隔文件格式，例如 CSV 和 TSV。Snowflake 支持的数据加载文件格式类型包括用于卸载的相同格式类型，以及 XML、Avro 和 ORC。

将数据从带分隔符的文件（如 CSV 或 TSV 文件）加载到 Snowflake 时，默认字符集为 UTF-8；分隔文件接受其他字符集，但它们在存储在表中之前会被 Snowflake 转换为 UTF-8。对于所有其他支持的文件格式，UTF-8 是加载数据时唯一支持的字符集。对于卸载数据，无论文件格式类型如何，UTF-8 都是唯一受支持的字符集。



Snowflake 支持换行符分隔的 JSON (NDJSON) 标准格式，以及用于将数据加载到 Snowflake 表中的逗号分隔 JSON 格式。卸载到文件时，Snowflake 仅输出为 NDJSON 格式。

将数据加载到 Snowflake 中和从 Snowflake 中卸载数据时，可以使用许多格式选项。可用的格式选项取决于文件格式的类型。表 6-2 包括 CSV 文件格式类型的格式选项。

表 6-2. CSV 文件格式类型的格式选项

格式选项	参数	装载/ 卸载	描述	违约
编码 =	'<string>' UTF8	加载指定 源数据的字符集	UTF8	
ERROR_ON_COLUMN_COUNT_MISMATCH =	真 FALSE 加载布尔值，指定是否		如果输入文件中的分隔列数与相应表中的列数不匹配，则生成解析错误	TRUE
REPLACE_INVALID_CHARACTERS =	真 FALSE 加载布尔值，指定是否		将无效的 UTF-8 字符替换为 Unicode 替换字符串	假
SKIP_BLANK_LINES = 真 FALSE	加载指定是否跳过空白行			假
SKIP_BYTE_ORDER_MARK = 真 FALSE	加载布尔值，指定是否		跳过字节顺序标记	TRUE
VALIDATE_UTF8 = 真 FALSE	加载布尔值，指定是否		验证 UTF-8 字符编码	TRUE

格式选项	参数	装载/ 卸载	描述	违约
SKIP_HEADER=	<integer>	加载 开头的行数	0 要跳过的文件	
TRIM_SPACE =	真 FALSE 加载布尔值, 指定是否		假 删除 领域	
BINARY_FORMAT =	十六进制 以 64 为基地 UTF8	装载 卸载	定义二进制输入和输出 的编码格式	HEX
DATE_FORMAT =	'<string>' AUTO	装载 卸载	定义日期值的格式	AUTO
EMPTY_FIELD_AS_NULL =	真 FALSE 加载 / 卸载		指定是否为文件中的空字 段插入 SQL NULL	TRUE
逃脱 =	'<character>' NONE	装载 卸载	用作封闭和非封闭值的转 义字符的单个字符串	NONE
ESCAPE_UNENCLOSED _FIELD =	'<character>' NONE	装载 卸载	仅用作非封闭值的转义字 符的单个字符串 / (反斜杠)	
FIELD_DELIMITER =	'<character>' NONE	装载 卸载	分隔文件中字段的一个 或多个字符	、 (逗号)
FIELD_OPTIONALLY _ENCLOSED_BY =	'<character>' NONE	装载 卸载	用于将字符串括起来的字 符, 通常与 Escape 一起	NONE
NULL_IF =	(' [, , ... })	装载 卸载	使用与 SQL NULL 之间 相互转换的字符串	NULL (\\N)
RECORD_DELIMITER =	'<character>' NONE	装载 卸载	分隔文件中记录的一个 或多个字符	换行符 字符
TIME_FORMAT =	'<string>' AUTO	装载 卸载	定义时间值的格式	AUTO
TIMESTAMP_FORMAT =	'<string>' AUTO	装载 卸载	定义时间戳值的格 式	AUTO
FILE_EXTENSION =	'<string>'	unloading	指定文件的扩展名 卸载到阶段	NULL

表 6-3 包括 JSON 文件格式类型的格式选项。JSON 文件格式类型是 Snowflake 支持的几种半结构化数据类型之一。其他半结构化数据类型（如 Avro、ORC、Parquet 和 XML）的格式类型选项可在 Snowflake 在线文档中找到。

表 6-3. JSON 文件格式类型的格式选项

格式选项	参数	装载/ 卸载	描述	违约
ALLOW_DUPLICATE	真 FALSE 加载布尔值, 指定是否		假 允许重复的对象字段名 称 (仅保留最后一个)	
BINARY_FORMAT =	十六进制 以 64 为基地 Loading 定义编码格式 UTF8	对于二进制字符串值	HEX	
DATE_FORMAT =	" 自动加载 定义日期的格式	值	AUTO	
ENABLE_OCTAL =	真 FALSE 加载布尔值, 启用	八进制数	假	
忽略 UTF8_ERRORS = TRUE FALSE 加载布尔值, 指定是否		UTF-8 编码错误会产生条 件	假	
NULL_IF =	([, , ... })	加载用于转换为 和 的 String 从 SQL NULL	NULL (\N)	
REPLACE_INVALID _CHARACTERS =	真 FALSE 加载布尔值, 指定是否	将无效的 UTF-8 字 符替换为 Unicode 替换字符	假	
SKIP_BYTE_ORDER_MARK = 真 FALSE 加载布尔值, 指定是否		跳过字节顺序标记	TRUE	
STRIP_NULL_VALUES = 真 FALSE 加载指示 JSON 的布尔值		parser 删除包含 null 值 的对象字段或数组元素	假	
STRIP_OUTER_ARRAY = 真 FALSE 加载指示 JSON 的布尔值		parser 删除外括号	假	
TIME_FORMAT =	" AUTO Loading 定义时间格式	值	AUTO	
TIMESTAMP_FORMAT = " AUTO Loading 定义		时间戳值	AUTO	
TRIM_SPACE =	真 FALSE 加载布尔值, 指定是否	从字符串中删除空格	假	
FILE_EXTENSION =	'<string>'	unloading 指定文件的扩展名 卸载到阶段	NULL	

为 Snowflake 数据加载创建文件格式是可选的。如果您预见到能够通过命令重用文件格式对象来加载结构相似的文件，建议您创建文件格式对象。

数据文件压缩

您可以将压缩和未压缩的数据加载到 Snowflake 中。默认情况下，存储在 Snowflake 中的所有数据都使用 gzip 进行压缩，无需选择 columns 或选择压缩算法。您可以在将数据加载到 Snowflake 之前压缩数据文件；如果文件较大，建议您压缩文件。Snowflake 支持 GZIP、BZIP2、DEFLATE、RAW_DEFLATE、BROTLI 和 ZSTANDARD (ZSTD) 压缩方法，并且可以自动检测除 BROTLI 和 ZSTANDARD (ZSTD) 之外的所有方法。对于这些压缩方法，您需要在数据加载时指定压缩方法。



Snowflake 还支持 Parquet 文件格式类型的 Lempe-Ziv-Oberhummer (LZO) 和 SNAPPY 压缩方法。

数据处理频率

数据处理从准备、处理和存储的原始数据开始。可以将实时数据流式传输到存储中以实现即时可用性，但更常见的是，数据是近乎实时地处理的。有时，数据是批量处理的。

让我们来看看数据处理类型之间的差异。

批处理

批处理涉及自动处理在指定时间段内累积的大量数据。工资单系统是批处理的一个很好的例子，因为数据可能只需要每两周处理一次。您只能批量访问的任何数据、数据新鲜度不是任务关键型的情况，以及您在大型数据集上使用复杂算法的任何时候，都可能是批处理的良好用例。批处理最重要的好处是它通常比其他类型的数据处理更便宜。它的成本更低，因为仅在执行处理时使用计算资源，并且由于批处理的执行频率低于近乎实时的处理，因此总体成本更低。不过，权衡的是数据新鲜度。

流式处理、连续加载和微批处理

流、流处理、连续加载、近乎实时处理和微批处理这些术语经常互换使用，因为在实践中它们会获得类似的结果。连续加载、近乎实时的处理和微批处理是同义词；但是，这三个术语与术语 Streaming 和 Stream Processing 之间存在差异。

对于微批处理实施，批处理对少量数据执行，这些数据可以在 60 秒内完成处理。如果数据加载时间通常超过 60 秒，则可能需要考虑减小每个批次的大小。加载每个微批次之间的时间可能大于 60 秒，但加载之间的间隔通常不超过几分钟。这实现了近乎实时的结果。

相比之下，在实际实时数据至关重要的情况下，纯流处理解决方案将使用 Kafka Streams API 或 Confluent 的 KSQL 等工具。实时数据很重要的示例包括数据应用程序提供即时欺诈检测功能的安全情况，以及必须实时查看电子商务数据或物联网（IoT）数据（例如安全摄像头或患者医疗设备）的运营情况。

在下一节中，我们将讨论使用所有方法加载数据的方法，但我们将主要关注持续加载实现。与批处理不同，连续加载要么具有小状态，要么根本没有状态，并且通常涉及相对简单的转换。当我们需要最新的近乎实时的数据，而不需要知道最近两三秒内发生了什么时，最常使用连续加载。我们可能希望使用持续加载的某些情况用于 Web 分析和人力资源系统。

Snowflake Stage 参考资料

Snowflake 阶段在第 3 章中介绍。我们将在这里简要回顾一下有关 stages 的一些重要内容，因为它们是 Snowflake 数据加载和卸载的重要组成部分。

Snowflake 阶段是临时存储空间，用作将文件引导至 Snowflake 表或将数据从 Snowflake 表卸载到文件中的中间步骤。阶段主要有两种类型：内部和外部。对于外部阶段，文件存储在外部位置（如 S3 存储桶）中，并由外部阶段引用。以前，对这些外部位置的访问是通过 Cloud Identity 和 Access Management（IAM）角色以及访问控制列表（ACL）来管理的。今天，最佳实践是创建存储集成。存储集成是一个 Snowflake 对象，用于存储为外部云存储生成的 IAM 实体，并可选择包括 Amazon S3、Google Cloud Storage 或 Microsoft Azure 允许或阻止的存储位置。内部阶段类型包括内部命名阶段、用户阶段和表阶段。请注意，Snowflake 中的所有阶段类型都使用 @ 符号表示。

命名阶段

内部命名阶段是数据库对象；因此，它们可由已被授予具有适当权限的角色的任何用户使用。引用命名阶段的 SQL 语句将需要 @ 符号以及阶段的名称。

要列出命名阶段，您可以运行该语句。

用户阶段

每个 Snowflake 用户都有一个用于存储文件的阶段，该阶段只能由该用户访问。用户阶段不是单独的数据库对象，不能更改或删除它。用户阶段中的数据只能由执行 SQL 命令的特定用户访问。引用用户阶段的 SQL 语句将需要 @^ sym - bols。SQL 语句可用于列出用户阶段。对用户阶段运行 list 命令后，结果应类似于图 6-1 中所示的结果。

	name	size	md5	...
1	worksheet_data/c1bccd70-c8cc-4c30-9f2e-b0559dc641af	720	f3112daba61c8171102aa2376cd41ee3	
2	worksheet_data/metadata	400	5dd847674966f16131fafae3ad696971	

图 6-1. 用户阶段的 list 命令的结果

表阶段

表阶段不是一个单独的数据库对象；相反，它是一个与桌子本身相关的隐式阶段。与用户阶段一样，不能更改或删除表阶段。此外，表阶段中的数据只能由那些被授予从表中读取权限的 Snowflake 角色访问。引用表阶段的 SQL 语句将需要符号以及表的名称。是可用于列出表阶段的语句。

数据源

数据可以从本地文件系统、Amazon S3 存储桶、Azure 容器或 GCP 存储桶加载到 Snowflake 中。从不同来源加载数据不受托管 Snowflake 帐户的云平台的限制。除了某些自动摄取 Snowpipe 使用案例外，您可以从前面对提到的任何来源加载数据，无论您的 Snowflake 账户是托管在 Amazon、Azure 还是 GCP 上。

使用 Snowflake 或第三方工具，您几乎可以从任何数据源（包括文件、API、企业应用程序和数据库）将数据加载到 Snowflake 中。

在本章中，我们将获得插入和加载结构化和半结构化数据的实践经验，并将发现使用 Kafka Connector、Snowpipe 和/或第三方工具提取其他数据源（如 IoT 和 Web/日志数据）的方法。

数据加载工具

在本节中，我们将探讨将数据加载到 Snowflake 中的五种不同方法。第一种方法（到目前为止，我们在整个章节中一直在使用）是通过 Snowflake 工作表中的 SQL 命令语句将数据插入表中。接下来，我们将学习如何直接在 Web UI 中上传文件。我们还将探索 SnowSQL CLI，我们将在其中使用 and commands。在有关数据加载的这一部分的最后，我们将了解数据管道以及第三方 ETL 和 ELT 工具。

使用 INSERT INTO 和 INSERT ALL 命令的 Snowflake 工作表 SQL

我们将使用 Snowflake 工作表中的 SQL 命令开始数据加载示例。我们在章节示例中经常使用该命令，因为当您只有几行数据时，它是将数据插入表中的最快方法。因此，您应该熟悉本节中的示例。

对于本节中的所有示例，我们将使用一个架构，我们将在该架构中创建所有必要的表。在准备过程中，我们将执行以下两个命令。如果您在完成本节中的示例时创建新工作表，请务必再次执行这些命令：

```
使用 WAREHOUSE LOAD_WH; 使用
数据库DEMO6_DB; 使用
SCHEMA WS;
```



我们将在本章中演示的所有命令都是数据操作语言（DML）命令，不要与 Snowflake 字符串和二进制函数融合。

用于结构化和半结构化数据的单行插入

我们首先为结构化数据的单行插入示例创建 TABLE1 表。然后，我们将在命令中使用显式指定的值插入一行：

```

创建或替换表 TABLE1
(id 整数、f_name 字符串、l_name 字符串、城市字符串) COMMENT =
“结构化数据的单行插入
using Explicitly Specified Values “;

INSERT INTO TABLE1 (id, f_name, l_name, city)
VALUES (1, 'Anthony', 'Robinson',
'Atlanta') ·SELECT * FROM TABLE1·

```

您现在应该看到已将一条记录输入到表中。让我们继续插入另一行：

```

INSERT INTO TABLE1 (id, f_name, l_name, city)
VALUES (2, 'Peggy', 'Mathison', '伯明翰') ;SELECT
* FROM TABLE1·

```

现在，让我们将注意力转向学习如何在 Snowflake 中插入半结构化数据。对于本章中的半结构化数据示例，我们将使用 JSON 数据。

对于半结构化数据的单行数据插入，我们不能像对结构化数据那样使用 clause。相反，我们将使用 query 子句作为替代项。请注意，这是我们的数据类型：

```

创建或替换表 TABLE2
(id integer, variant1 变体) COMMENT = “半结构化 JSON 数据的单
行插入”；

INSERT INTO TABLE2 (id, variant1) SELECT 1, parse_json (' { "f_name" :
“安东尼”， “l_name”：“罗宾逊”，
“city”：“亚特兰大” } ') ;
从表 2 中选择 *;

```

现在，我们的 TABLE2 表中有一行半结构化数据。让我们继续再插入一条 JSON 记录：

```

INSERT INTO TABLE2 (id, variant1) SELECT 2, parse_json (
{ "f_name" : "Peggy", "l_name" : "Mathison",
"city" : "伯明翰" } ) ;
从表 2 中选择 *;

```

语句的结果应显示两行，如图 6-2 所示。

	ID	VARIANT1	...
	1	{ "city": "Atlanta", "f_name": "Anthony", "l_name": "Robinson" }	
	2	{ "city": "Birmingham", "f_name": "Peggy", "l_name": "Mathison" }	

图 6-2. 在 TABLE2 表中插入两行半结构化数据的结果

到目前为止，我们的示例已经演示了如何向 Snowflake 表添加单行数据（结构化或半结构化）。接下来，我们将看看如何使用一个 SQL 语句一次添加多行数据。

结构化和半结构化数据的多行插入

首先，我们将创建一个新表，然后使用一个 SQL 语句插入两行：

创建或替换表 TABLE3

(id 整数、f_name 字符串、l_name 字符串、城市字符串) COMMENT = “使用显式声明值的结构化数据的多行插入” ;INSERT INTO TABLE3 (id, f_name, l_name, city) 值

(1, '安东尼', '罗宾逊', '亚特兰大'), (2, '佩吉', '马蒂森', '伯明翰');

从表 3 中选择 *;

您的结果应如图 6-3 所示。

	ID	F_NAME	L_NAME	CITY	...
1	1	Anthony	Robinson	Atlanta	
2	2	Peggy	Mathison	Birmingham	

图 6-3. 使用一条记录插入两条具有结构化数据类型的记录的结果
插入命令

现在，让我们尝试将数据从现有表插入到新表中。我们将创建与现有表完全相同的列的新表。请注意，我们将应用一个条件，以便仅将现有表中的特定记录插入到新表中：

创建或替换表 TABLE4

(id 整数、f_name 字符串、l_name 字符串、城市字符串) COMMENT = “使用查询对结构化数据进行多行插入，所有列都相同” ;插入表 4 (id, f_name, l_name, city)

SELECT * FROM TABLE3 WHERE CONTAINS (city, 'Atlanta') ;从表 4 中选择 *;

您可以在图 6-4 中看到命令的结果。新表中只插入了一行，因为只有该行符合条件。如果现有表中的多行满足选择条件，则将插入多行。

	...	ID	F_NAME	L_NAME	CITY
1		1	Anthony	Robinson	Atlanta

图 6-4. 将数据从应用了条件的现有表插入新表的结果

在下一个示例中，我们将创建另一个表，其行数与现有表相同，但我们在新表中插入的列值少于现有表中可用的列值：

```
CREATE OR REPLACE TABLE TABLE5
(id 整数、f_name 字符串、l_name 字符串、城市字符串) COMMENT =
“使用查询对结构化数据进行多行插入，列数较少”;
```

如果我们尝试使用命令，就像我们在前面的例子中所做的那样，将返回一个错误：

```
INSERT INTO TABLE5
SELECT * FROM TABLE3 WHERE CONTAINS (city, 'Atlanta');
```

如果我们在命令中指定要插入的列，我们的命令就会成功：

```
INSERT INTO TABLE5 (id, f_name, l_name)
SELECT id, f_name, l_name FROM TABLE3 WHERE CONTAINS (city, 'Atlanta') ;从
TABLE5 中选择 *;
```

TABLE5 表的命令结果（如图 6-5 所示）包括 CITY 列的 null 值。这是因为我们没有从现有表中插入该 column 值。

	ID	F_NAME	...	L_NAME	CITY
1	1	Anthony		Robinson	

图 6-5. 将现有源表中的较少列插入到新目标表中的结果

我们的下一个示例将演示如何使用公共表表达式（CTE）（子句中定义的命名子查询）将结构化数据的多行插入到表中。CTE 在语句中用作临时视图，但它不将定义存储在元数据中。为了演示此示例，我们首先需要创建一个表，该表将用作下一个演示的一部分：

```
CREATE OR REPLACE TABLE TABLE6
(id 整数、first_name 字符串、last_name 字符串、city_name 字符串) COMMENT =
“用作下一个演示一部分的表格”;

INSERT INTO TABLE6 (id, first_name, last_name, city_name) 值
(1, '安东尼', '罗宾逊', '亚特兰大'),
(2, '佩吉', '马蒂森', '伯明翰');
```

现在，我们将创建一个新表 TABLE7，我们将使用从 TABLE6 表中获取数据的 CTE 将数据插入其中：

```
创建或替换表 TABLE7
(id 整数、f_name 字符串、l_name 字符串、城市字符串) COMMENT =
“使用 CTE 对结构化数据进行多行插入”;

插入表 7 (id, f_name, l_name, city)
使用 CTE AS
(SELECT id, first_name as f_name, last_name as l_name,
city_name AS city FROM TABLE6) 选择
id、f_name、l_name、city FROM
CTE;SELECT * FROM TABLE7;
```

图 6-6 显示了 CTE 示例中的命令结果。CTE 用于提高代码可读性，使代码更易于维护，并利用递归编程。

	ID	F_NAME	L_NAME	...	CITY
1	1	Anthony	Robinson		Atlanta
2	2	Peggy	Mathison		Birmingham

图 6-6. CTE 示例中的命令结果

接下来，我们将探讨如何通过使用邮政编码列在两个表上创建内部联接来将记录插入到新表中。我们首先需要创建两个表，我们将从这两个表创建内部联接。第一个表将包括 identifier 列、名字和姓氏以及邮政编码列：

```
创建或替换表 TABLE8
(id 整数、f_name 字符串、l_name 字符串、zip_code 字符串) COMMENT = “用作下一个演示一部分的表格”;
INSERT INTO TABLE8 (id, f_name, l_name, zip_code) VALUES (1,
'Anthony', 'Robinson', '30301'), (2, 'Peggy', 'Mathison', '35005');
```

下一个表将包括 identifier、zip code、city 和 state 列：

```
创建或替换表 TABLE9
(id 整数、zip_code 字符串、城市字符串、州/省/市/自治区字符串)
COMMENT = “用作下一个演示一部分的表格”;
INSERT INTO TABLE9
(id, zip_code, city, state) 值
(1, '30301', '亚特兰大', '格鲁吉亚'),
(2, '35005', '伯明翰', '阿拉巴马州');
```

现在，我们将创建一个新表，我们将使用刚刚创建的两个新表中的内部联接来插入记录：

```
创建或替换表 TABLE10
(id 整数、f_name字符串、l_name字符串、城市字符串、州字符串zip_code字符串) COMMENT =
“在zip_code上使用内部 JOIN 从两个表中插入多行”;
```

```
插入到TABLE10 (id, f_name, l_name, city, state, zip_code) 从表 8
a 中选择 a.id, a.f_name、a.l_name、b.city、b.state、a.zip_code
```

```
INNER JOIN TABLE9 b 在 a.zip_code = b.zip_code;SELECT
*FROM TABLE10;
```

图 6-7 显示了使用内部联接的命令的结果。

	ID	F_NAME	L_NAME	...	CITY	STATE	ZIP_CODE
1	1	Anthony	Robinson		Atlanta	Georgia	30301
2	2	Peggy	Mathison		Birmingham	Alabama	35005

图 6-7。在两个现有表上使用内部联接的新表中的命令结果

到目前为止，我们的多行插入示例都是使用结构化数据。让我们看一个包含半结构化数据的多行插入示例：

```
创建或替换表TABLE11
```

```
(变体 1 变体) COMMENT = “半结构化 JSON 数据的多行插入”;
```

您会注意到，对于我们的下一个命令，我们将在第一个值的 ID 列前面包含一个下划线。对于第二个值，我们不会在 ID 列前面放置低于分数线：

```
插入到TABLE11
选择 parse_json (column1)
FROM 值
'{"_id": "1", "name": { "first": "Anthony", "last": "Robinson" }, "company": "Pascal", "email": "anthony@pascal.com", "phone": "+1 (999) 444-2222"}, {"_id": "2", "name": { "first": "Peggy", "last": "Mathison" }, "company": "Ada", "email": "Peggy@ada.com", "phone": "+1 (999) 555-3333"}';SELECT * FROM TABLE11;
```

命令的结果如图 6-8 所示。您会注意到，这些列按字母顺序显示，而不是按其列出顺序显示

语句。由于第一条记录的 ID 列前面有下划线，因此 ID 列是列出的第一列。

	VARIANT1	...
1	{ "_id": "1", "company": "Pascal", "email": "anthony@pascal.com", "name": { "first": "Anthony", "last": "Robinson" } }, "	
2	{ "company": "Ada", "email": "Peggy@ada.com", "id": "2", "name": { "first": "Peggy", "last": "Mathison" }, "phone": "+" }	

图 6-8. 半结构化 JSON 数据的多行插入结果

在前面的示例中，我们一直在将数据插入到单个表中。在下一部分中，我们将学习如何一次将数据插入多个表中。

多表插入

通过使用 query 语句将一行或多行插入表中，可以一次更新多个 table。插入可以是无条件的，也可以是有条件的。条件多表插入通过使用子句来确定每行将插入的表来创建插入。

第一个例子将演示无条件多表插入是如何工作的。首先，我们将创建源表：

```
CREATE TABLE TABLE12
  (id 整数、first_name 字符串、last_name 字符串、city_name 字符串) COMMENT = "源表用作无条件表插入的下一个演示的一部分";
INSERT INTO TABLE12 (id, first_name, last_name, city_name)
VALUES (1, 'Anthony', 'Robinson', 'Atlanta'), (2, 'Peggy', 'Mathison', 'Birmingham');
```

让我们创建两个目标表：

```
CREATE TABLE TABLE13
  (id 整数、f_name 字符串、l_name 字符串、城市字符串) COMMENT = "无条件表格插入 - 无条件的目标表格 1";
CREATE TABLE TABLE14
  (id 整数、f_name 字符串、l_name 字符串、城市字符串) COMMENT = "无条件表格插入 - 无条件的目标表格 2";
```

```
多表插入 ":
```

```
CREATE TABLE TABLE12
  (id 整数、first_name 字符串、last_name 字符串、city_name 字符串) COMMENT = "源表用作无条件表插入的下一个演示的一部分";
INSERT INTO TABLE13 (id, first_name, last_name, city_name)
VALUES (1, 'Anthony', 'Robinson', 'Atlanta'), (2, 'Peggy', 'Mathison', 'Birmingham');
```

现在，我们可以使用 TABLE12 表中的数据插入到两个新表中。您会注意到，我们有几个命令可以将数据插入到表中。首先，我们将 TABLE12 表中的所有数据插入到 TABLE13 表中。然后，我们将 select 值插入 TABLE13 和 TABLE14 表中。请特别注意对账单行：

```

全部插入
    INTO TABLE13 INTO TABLE13 (id, f_name,
        l_name, city)
VALUES (id, last_name, first_name, default) INTO TABLE14
(id, f_name, l_name, city) INTO TABLE14 VALUES (id,
city_name, last_name, first_name) SELECT id, first_name,
last_name, city_name FROM TABLE12

```

让我们看看我们的声明取得了什么成就。首先，使用 `SELECT * FROM TABLE12` 查看原始表中的数据；陈述。

接下来 TABLE13，使用

从 TABLE13；陈述。在图 6-9 中，您将看到 4 条记录入到 TABLE13 表中。请注意，第三条和第四条记录在 CITY 列中没有值。

	ID	F_NAME	L_NAME	CITY
1	1	Anthony	Robinson	Atlanta
2	2	Peggy	Mathison	Birmingham
3	1	Robinson	Anthony	
4	2	Mathison	Peggy	

图 6-9. multititable 语句的两个目标表之一

TABLE14 表呢？使用

`SELECT * FROM TABLE14: statement` 来查看

该表中存在的内容（如图 6-10 所示）。

	ID	F_NAME	L_NAME	CITY	...
1	1	Anthony	Robinson	Atlanta	
2	2	Peggy	Mathison	Birmingham	
3	1	Atlanta	Robinson	Anthony	
4	2	Birmingham	Mathison	Peggy	

图 6-10. multititable 语句的两个目标表中的第二个

我们可以看到 TABLE14 表中也有四行。前两条记录与 TABLE12 和 TABLE13 表中的记录相同。但是，第 3 行和第 4 行是不同的。查看前面的语句，了解如何将不同的值插入到各个行中。

我们前面的示例演示了一个无条件多表插入示例。现在我们来看一个条件多表插入示例。让我们为此示例创建源表：

创建或替换表 TABLE15

(id 整数、first_name 字符串、last_name 字符串、city_name 字符串)

```
COMMENT = “源表将用作下一个演示的一部分  
条件多表插入“插入到TABLE15 (id, first_name, last_name, city_name) 值  
(1, ‘Anthony’, ‘Robinson’, ‘Atlanta’), (2, ‘Peggy’, ‘Mathison’, ‘伯  
明翰’), (3, ‘Marshall’, ‘Baker’, ‘芝加哥’), (4, ‘Kevin’, ‘Cline’,  
‘丹佛’), (5, ‘艾米’, ‘Ranger’, ‘埃弗利’), (6, ‘安迪’, ‘穆雷’, ‘弗雷  
斯诺’) ;
```

接下来，我们将创建两个目标表：

```
CREATE OR REPLACE TABLE TABLE16  
(id 整数、f_name 字符串、l_name 字符串、城市字符串) COMMENT = “条件多  
表插入的目标表 1”;
```

```
CREATE OR REPLACE TABLE TABLE17  
(id 整数、f_name 字符串、l_name 字符串、城市字符串) COMMENT = “条件多  
表插入的目标表 2”;
```

现在，我们准备演示条件多表插入：

```
全部插入  
当 id <5 时  
    走进TABLE16  
当 id <3 时  
    走进TABLE16  
    走进TABLE17  
当 id = 1 时 THEN  
    INTO TABLE16 (id, f_name) VALUES (id, first_name)  
    ELSE  
    INTO TABLE17 SELECT id, first_name, last_name,  
    city_name FROM TABLE15;
```

使用中，我们可以看到表中插入了 7 行（如图 6-11 所示）。

	ID	F_NAME	L_NAME	CITY	...
1	1	Anthony	Robinson	Atlanta	
2	2	Peggy	Mathison	Birmingham	
3	3	Marshall	Baker	Chicago	
4	4	Kevin	Cline	Denver	
5	1	Anthony	Robinson	Atlanta	
6	2	Peggy	Mathison	Birmingham	
7	1	Anthony			

图 6-11. 将条件多表插入到第一个目标表中的结果

在前面的语句中，您会注意到 TABLE16 表包含在三个不同的条件语句中。这些条件之

语句将值插入到四列中的两列中，这就是最后一条记录有两个 null 值的原因。

使用该语句，我们可以在图 6-12 中看到这四行已插入到表中。

	ID	F_NAME	...	L_NAME	CITY
1	1	Anthony		Robinson	Atlanta
2	2	Peggy		Mathison	Birmingham
3	5	Amy		Ranger	Everly
4	6	Andy		Murray	Fresno

图 6-12. 将条件多表插入到第二个目标表中的结果

您会注意到，ID 为 5 和 6 的记录仅显示在第二个 Target 表中。这两条记录是作为语句部分的结果插入的。

在到目前为止的示例中，我们已经加载了结构化数据和半结构化数据。我们加载的半结构化数据是 JSON 文件格式类型，我们使用 Snowflake 数据类型来加载数据。该类型只是半结构化数据的一种数据类型。半结构化数据还有另外两种 Snowflake 数据类型。我们首先看数据类型的示例，然后看 OBJECT 数据类型；我们将使用 Universal Data Type 加载 AN 和 AN 的数据。

ARRAY_INSERT

Snowflake 函数为我们提供了一种将数据直接插入到表中的方法。我们将创建一个新表，用于演示 ARRAY_INSERT data 函数：

```
CREATE OR REPLACE TABLE TABLE18  
  (数组变体) COMMENT =  
  "插入数组" ;
```

该函数的语法如下：

```
ARRAY_INSERT (, , ) 其中包括 ARRAY_CONSTRUCT ()
```

如你所见，该函数在数组中的特定位置插入一个数组，后跟一个新的数组值。我们将使用两个不同的示例来了解它是如何工作的。首先，让我们插入一个具有新值的数组，该值自然会出现在数组中的下一个位置：

```
插入到TABLE18  
选择 ARRAY_INSERT (array_construct (0, 1, 2, 3), 4, 4) ;
```

在 TABLE18 表上运行一条语句，您将获得如图 6-13 所示的结果。

	ARRAY
1	[0, 1, 2, 3, 4]

图 6-13。使用命令的结果

现在让我们看看如果我们插入数组中自然会排在后面的值，但将其插入到不同的位置，会发生什么：

```
插入到TABLE18  
选择 ARRAY_INSERT (array_construct (0, 1, 2, 3), 7, 4);
```

如果我们再次运行一条语句，我们将看到如图 6-14 所示的结果。

	ARRAY
1	[0, 1, 2, 3, 4]
2	[0, 1, 2, 3, undefined, undefined, undefined, 4]

图 6-14. 第二次使用该命令的结果

如图 6-14 所示，数字 4 到第七个位置。对于位置 4 到 6，存在未定义的值，因为没有为这些位置提供值。

OBJECT_INSERT

OBJECT_INSERT 是一个 Snowflake 半结构化数据函数。为了演示该函数的工作原理，让我们首先创建一个新表：

```
创建或替换表 TABLE19  
(对象变量) COMMENT = “插  
入对象”;
```

该函数的语法如下：

```
OBJECT_INSERT (, , ) WHERE INCLUDES OBJECT_CONSTRUCT (<键值对>)
```

现在让我们使用函数将一些键值对插入到新表中：

```
插入 TABLE19  
选择 OBJECT_INSERT (OBJECT_CONSTRUCT ('a', 1, 'b', 2, 'c', 3), 'd',  
4);SELECT * FROM TABLE19;
```

如图 6-15 所示，键值对的插入符合预期。

	OBJECT
1	{ "a": 1, "b": 2, "c": 3, "d": 4 }

图 6-15. 使用函数的结果

接下来，我们将了解如何处理 null 值。首先，我们将插入一个空值、一个用引号括起来的单词 null 值的 null，然后插入一个不带引号的 null 值：

```
插入到TABLE19选择
OBJECT_INSERT (object_construct ('a', 1, 'b', 2, 'c', 3), 'd',
') :插入到TABLE19选择
OBJECT_INSERT (object_construct ('a', 1, 'b', 2, 'c', 3), 'd',
'null') :插入到TABLE19选择
OBJECT_INSERT (object_construct ('a', 1, 'b', 2, 'c', 3), 'd',
null) :插入到TABLE19选择
OBJECT_INSERT (object_construct ('a', 1, 'b', 2, 'c', 3), null,
')
```

所有命令的结果如图 6-16 所示。您会从最后两条插入的记录中注意到，只要键值对包含 null 值，就不会插入键值对。

	OBJECT
1	{ "a": 1, "b": 2, "c": 3, "d": 4 }
2	{ "a": 1, "b": 2, "c": 3, "d": "" }
3	{ "a": 1, "b": 2, "c": 3, "d": "null" }
4	{ "a": 1, "b": 2, "c": 3 }
5	{ "a": 1, "b": 2, "c": 3, "d": "" }
6	{ "a": 1, "b": 2, "c": 3, "d": "null" }
7	{ "a": 1, "b": 2, "c": 3 }
8	{ "a": 1, "b": 2, "c": 3 }

图 6-16. 使用函数插入行的结果

我们已经了解了使用 Snowflake 工作表中的 SQL 语句将数据插入 Snowflake 表的几种不同方法。我们使用 Snowflake 工作表来演示许多概念，因为它易于使用。我们在本节中学到的许多概念也适用于我们使用 Snowflake Web UI 加载数据文件或使用 SnowSQL CLI 时的情况。

在下一节中，我们将使用 Web UI Load Data 向导加载结构化数据文件和半结构化数据文件。该向导仅用于一次手动加载几个小文件。

在继续不同的架构之前，让我们回顾一下此架构中我们一直在使用的所有表的表注释：

显示 ‘%TABLE%’ 等表：

我们一直在为本章中创建的所有表使用注释。如果需要，我们还可以为我们在表中创建的任何单个字段使用注释。

Web UI 加载数据向导

在前面将数据插入 Snowflake 表中的示例中，我们使用 SQL 语句将数据直接插入到表中。我们现在将探索一种加载数据文件的方法。在本节中，我们将加载一些结构化数据和半结构化数据文件，每个文件的大小都小于 50 MB。



在使用 Snowflake Web UI Load Data 向导时，请务必遵循小文件大小建议。这样做将有助于确保更好的性能。它还可以防止您的浏览器崩溃，因为随着文件大小的增加，将需要更多的内存消耗来加密文件。如果您需要加载更大的文件，请参阅下一节，其中介绍了如何使用 Snowflake SnowSQL CLI。

以前我们使用命令将数据直接加载到 Snowflake 表中，而向导通过无缝组合两个阶段来间接将数据加载到 Snowflake 表中：一个阶段用于暂存文件，下一个阶段用于将暂存文件加载到表中（如图 6-17 所示）。

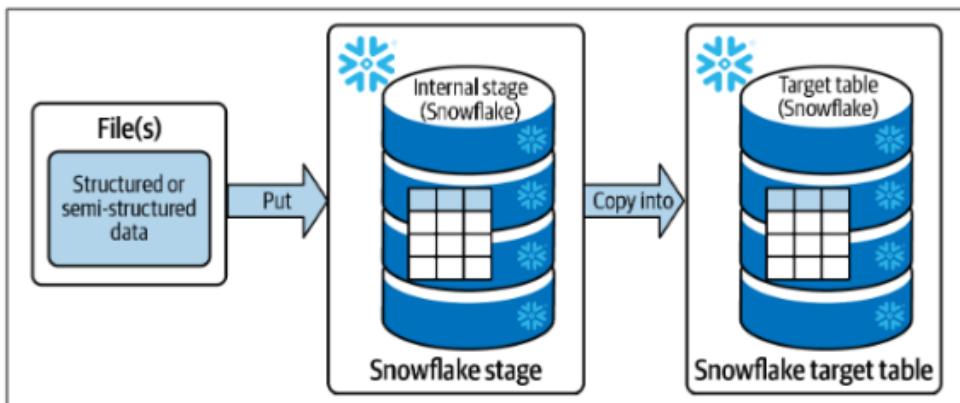


图 6-17. 通过 Snowflake 阶段将文件加载到 Snowflake 目标表中

为此，向导在后台使用 `and` 命令。该命令将文件放置在一个阶段中，并且该命令将数据从 Snowflake 阶段移动到 Snowflake 表。将数据从暂存加载到 Snowflake 表中后，向导将删除所有暂存文件。

目前，Snowflake Web UI 加载数据向导仅在经典控制台中可用。预计该向导将在 2023 年初的某个时候可用于 Snowsight。

当我们仍在 Snowsight 中时，让我们继续创建动手示例所需的表。请务必使用此示例的 UI 架构：

```
使用 SCHEMA UI;  
创建或替换表 TABLE20  
(id 整数、f_name 字符串、l_name 字符串、城市字符串) COMMENT =  
“通过 Web UI 向导加载结构化数据文件”;
```

现在，我们需要导航到 Classic Console 以完成本节中的示例。在屏幕的左上角，单击 首页 图标，然后单击 经典控制台 按钮（如图 6-18 所示）。

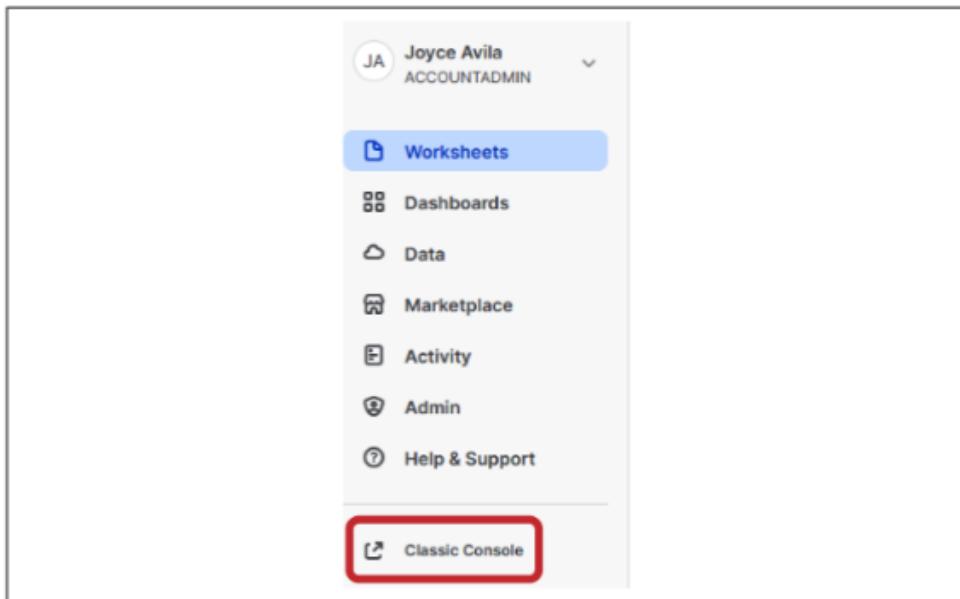


图 6-18. 单击 Home 图标和 Classic Console 按钮从 Snowsight 导航到 Classic Console

进入 Classic Console 后，单击 Databases 选项卡，然后向下钻取到 DEMO6_DB 数据库。接下来，单击 TABLE20 表名称，我们将使用它来插入包含结构化数据的文件。您的屏幕现在应该如图 6-19 所示。

Column Name	Ordinal ▲	Type	Nullable	Default
ID	1	NUMBER(38,0)	true	NULL
F_NAME	2	VARCHAR(16777216)	true	NULL
L_NAME	3	VARCHAR(16777216)	true	NULL
CITY	4	VARCHAR(16777216)	true	NULL

图 6-19。单击 Databases 选项卡后，DEMO6_DB数据库和 TABLE20 表

您会注意到 Load Table upload 图标。这就是我们将用于将包含结构化数据的文件上传到 TABLE20 Snowflake 表的方法。

结构化数据示例

我们已准备好继续我们的结构化数据示例。单击 Load Table 图标后，系统会要求您选择要用于加载文件的虚拟仓库（如图 6-20 所示）。

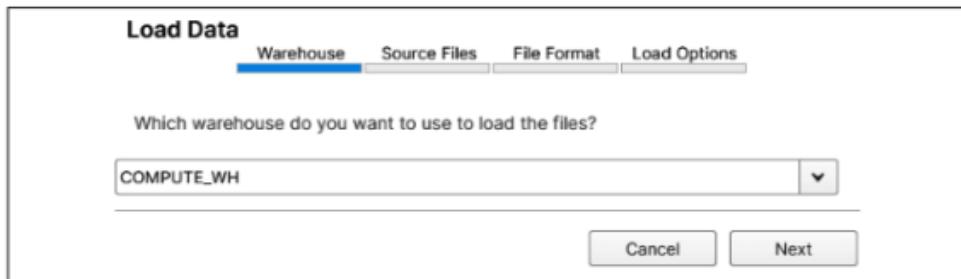


图 6-20. 用于使用向导加载文件的虚拟仓库选择

选择我们在本章开头创建的 LOAD_WH 虚拟仓库，然后单击 Next（下一步）按钮。选择 CSV file 通过单击从您的计算机中选择 Files 按钮，然后选择保存在您的计算机上的 TABLE20 CSV file 如图 6-21 所示。单击 Next 按钮。

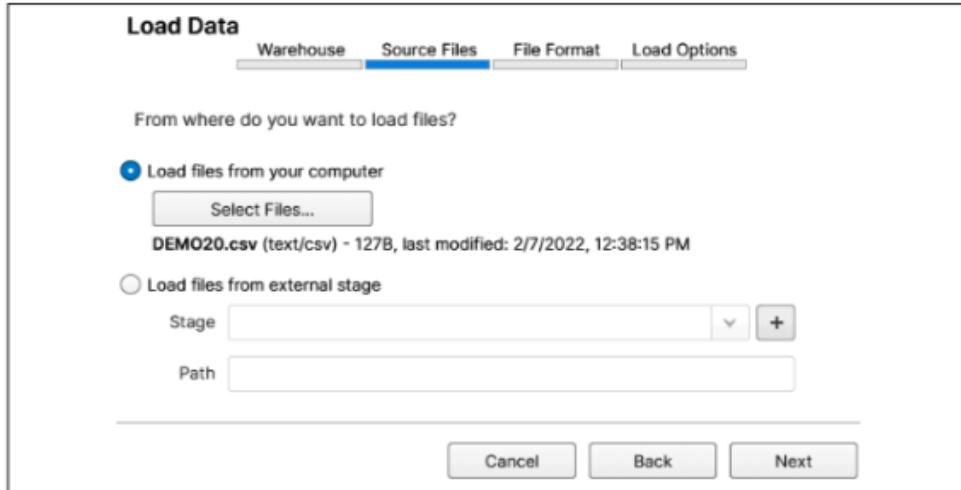


图 6-21. 使用向导从计算机中选择 CSV 文件

现在是我们必须选择文件格式的时候，如果我们已经创建了文件格式。否则，我们将创建一个新的文件格式。在我们的示例中，到目前为止我们还没有创建任何文件格式，因此我们需要创建一个新的文件格式。要为 mat 创建新文件，请单击 +（加号）按钮（如图 6-22 所示）。

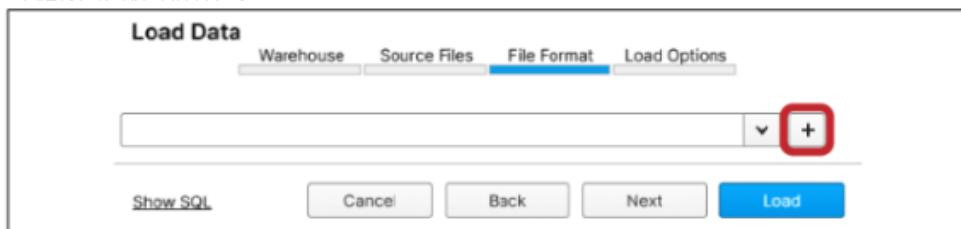


图 6-22。在向导中创建新文件格式

单击 + 按钮后，我们将可以选择要创建的 mat 文件。您应该熟悉图 6-23 中的文件格式选项。请参阅表 6-2 以查看这些相同的选项，如果您选择在工作表中使用 SQL 创建新文件格式，则可以选择这些选项。

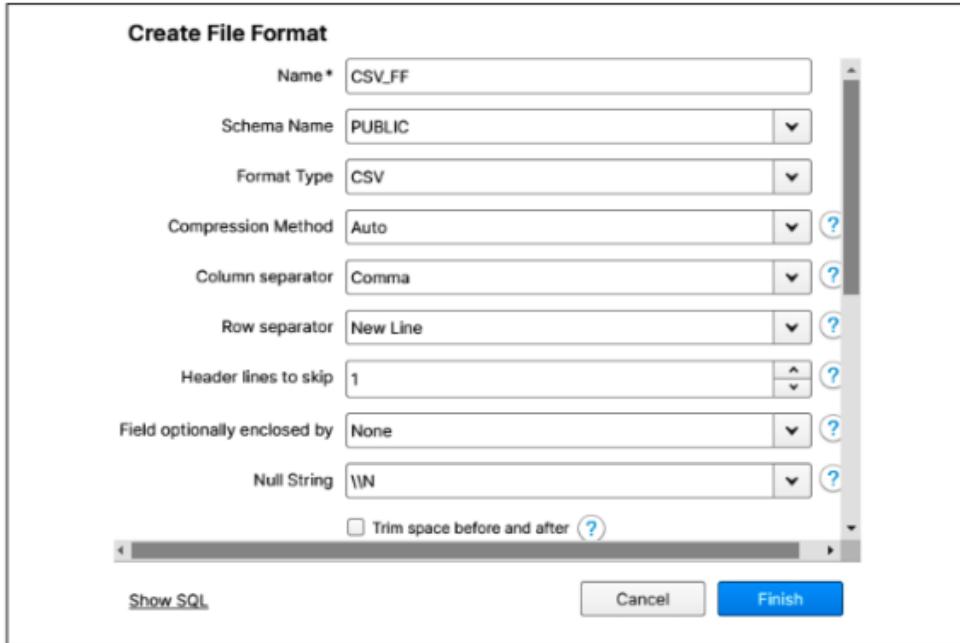


图 6-23。从 Snowflake Load Data 向导中创建文件格式

因为我们的 CSV 文件有一个标题，所以我们想将“Header lines to skip”选项从 0 更改为 1。如果您单击 Show SQL 链接，将显示图 6-24 中所示的 SQL 代码。

The screenshot shows a SQL code block with the title "SQL". The code is:

```
1 CREATE FILE FORMAT "DEMO6_DB"."PUBLIC".CSV_FF TYPE = 'CSV' COMPRESSION =  
  'AUTO' FIELD_DELIMITER = ',' RECORD_DELIMITER = '\n' SKIP_HEADER = 1  
  FIELD_OPTIONALLY_ENCLOSED_BY = 'NONE' TRIM_SPACE = FALSE  
  ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE ESCAPE = 'NONE' ESCAPE_UNENCLOSED_FIELD  
  = '\134' DATE_FORMAT = 'AUTO' TIMESTAMP_FORMAT = 'AUTO' NULL_IF = ('\\N');
```

Below the code are two buttons: "Select SQL" and "Close".

图 6-24。使用向导中的默认选项创建 Snowflake 文件格式的 SQL 代码

如果您查看了 SQL，请单击 Close 按钮，然后单击 Finish 按钮，然后单击 Next 按钮。此时，您将看到可用于错误处理的 Load 选项（如图 6-25 所示）。

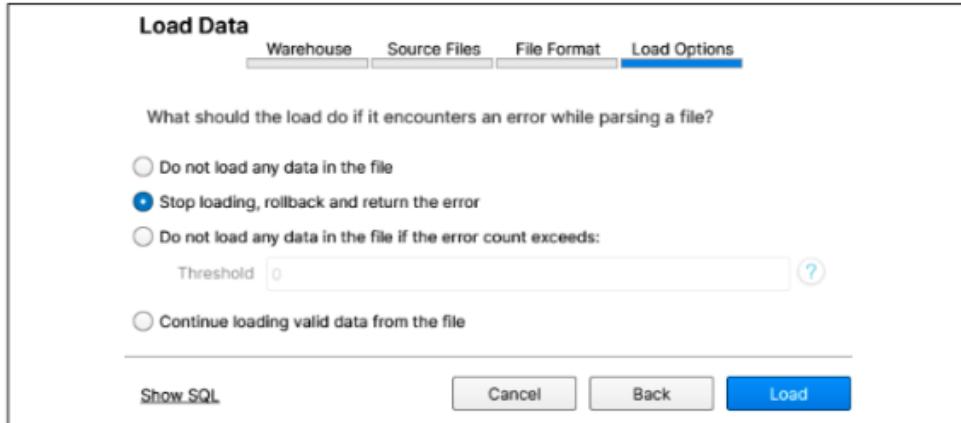


图 6-25. Snowflake Load Data 向导中的错误处理选项

我建议您单击每个选项，然后选择 Show SQL（显示 SQL）选项，以便您可以查看如何编写 SQL 代码（如果需要）。



如果您正在 SnowSQL CLI 中工作，并且不确定如何编写创建特定文件格式所需的 SQL，或者如果您需要有关创建错误处理的更多帮助，则可以返回到 Snowflake Load Data 向导，方法是进行选择，然后单击 Show SQL 选项。

现在单击 Load 按钮，您将看到消息，指出 Snowflake 正在暂存文件（如图 6-26 所示）。



图 6-26. 指示 Snowflake 已加密文件并正在将其放入某个阶段的消息

Snowflake 完成数据加载后，您将看到加载结果，指示已解析和加载四行。我们继续通过单击屏幕右上角的 Snowsight 选项卡返回到 Snowsight（如图 6-27 所示）。

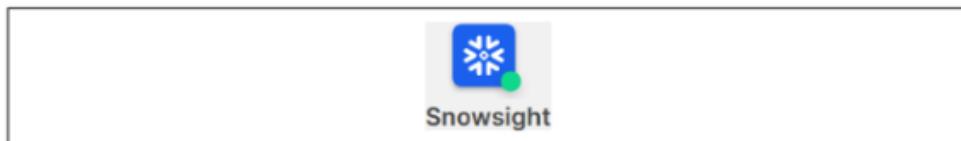


图 6-27. Snowflake Classic 控制台中的 Snowsight 选项卡

导航回工作表并执行以下命令：

```
使用数据库DEMO6_DB;使用
SCHEMA UI;SELECT * FROM
TABLE20;
```

通过使用 DEMO6_DB 数据库和 UI 架构运行命令，您会注意到 4 条记录已成功插入到 TABLE20 表中。

在下一节中，我们将使用相同的 CSV 数据文件，但这次我们将使用 SnowSQL CLI 上传文件。

SnowSQL、CLI、SQL PUT 和 COPY INTO 命令

SnowSQL 是用于连接到 Snowflake 的 CLI。它可用于执行 SQL 查询并执行所有数据定义语言（DDL）和 DML 操作。可以从 Snowflake Classic 控制台下载 SnowSQL，方法是单击 Help（帮助）选项卡，然后选择 Download（下载）选项。然后，您可以选择下载适用于 Linux、Windows 或 macOS 的 SnowSQL CLI。有关如何为每个操作系统安装 SnowSQL 的说明，请参阅 Snowflake 在线文档。

下载并安装 SnowSQL 后，您可以使用以下示例从命令提示符登录到 SnowSQL。请注意，JKAVILA2022 是我的用户名，dx58224.us-central1.gcp 是我的 Snowflake 帐户名称。该选项用于指示您的 Snowflake 帐户，并用于标识要连接到 Snowflake 的用户名。此处显示的帐户名和用户名仅用于考试。它们不是实际可用的信息，而是我之前创建试用 Snowflake 帐户时提供的信息：

```
c:\>snowsql -a dx58224.us-central1.gcp -u JKAVILA2022 密码: *****
```

让我们通过指定要用于此示例的角色、虚拟仓库、数据库和架构来开始使用 SnowSQL：

```
使用角色 ACCOUNTADMIN;使用
WAREHOUSE LOAD_WH;使用数据
库DEMO6_DB;使用 SCHEMA
SNOW;
```

您可以看到最后一个 SQL 语句和所有先前 SQL 语句的结果，其中显示命令行提示符 house>@ 的（图 6-28）。

```

OKAVILA2022#LOAD WHCH6 DB.PUBLIC>USE SCHEMA SNOW;
+-----+
| status |
+-----+
| Statement executed successfully. |
+-----+
1 Row(s) produced. Time Elapsed: 0.076s
OKAVILA2022#LOAD WHCH6 DB.SNOW;

```

图 6-28. 使用所述角色、虚拟仓库、数据库和架构的结果

现在，让我们使用以下 SQL 语句创建一个新表：

```
CREATE OR REPLACE TABLE TABLE20 (id integer, f_name string,
    l_name字符串、城市字符串);
```

创建表后，您就可以使用命令将 CSV 文件加载到表阶段了：

```
file:///users/joyce/documents/TABLE20.csv @ "DEMO6_DB".$TABLE20";
```

替换为自己的文件路径。此文件请确保在计算机上的文件路径和 @ 符号之间放置一个空格。图 6-29 显示了使用命令将 CSV 文件加载到表阶段的结果。

```

OKAVILA2022#LOAD WHCH6 DB.SNOW>Put file:///users/joyce/documents/DEMO20.csv @ "CH6_DB"."SNOW".%$DEMO20";
+-----+
| source | target | source_size | target_size | source_compression | target_compression | status |
+-----+
| DEMO20.csv | DEMO20.csv.gz | 128 | 144 | NONE | GZIP | UPLOADED |
+-----+

```

图 6-29. 使用命令将 CSV 文件加载到表阶段的结果

接下来，我们将从 table 阶段复制文件，结果如图 6-30 所示：

```
从 @ "DEMO6_DB" 复制到 "TABLE20"。雪 "%$TABLE20"
file_format=( type=csv SKIP_HEADER=1 );
```

```

OKAVILA2022#LOAD WHCH6 DB.SNOW>COPY @ "DEMO6_DB".$TABLE20 FROM file:///users/joyce/documents/DEMO20.csv FILE_FORMAT=( type=csv SKIP_HEADER=1 );
+-----+
| File | status | rows_parsed | rows_loaded | error_start | errors_total | first_error | first_error_line | first_error_character | first_error_code |
+-----+
| DEMO20.csv.gz | LOADED | 4 | 4 | 0 | NULL | NULL | NULL | NULL | NULL |
+-----+
1 Row(s) produced. Time Elapsed: 2.497s
OKAVILA2022#LOAD WHCH6 DB.SNOW;

```

图 6-30. 使用命令将数据从阶段加载到目标表的结果

在此示例中，我们将文件放入 table 阶段，因为我们要加载到单个表中。我们可以很容易地创建一个内部命名的 stage 并将文件放在那里。如果您与多个用户共享文件，或者将文件加载到多个表中，建议使用内部命名阶段。

使用该命令时，可以执行基本转换，例如使用命令对列重新排序或执行强制转换。有关在加载期间转换的更多信息，请参阅 Snowflake 文档。

我们已经了解了如何使用 Web UI Load Data 向导和 SnowSQL 来获取文件并手动将它们加载到 Snowflake 表中，方法是首先将文件放入一个阶段，然后将数据从该阶段复制到 Snowflake 表中。在下一节中，我们将了解如何将数据自动加载到 Snowflake 中。

数据管道

数据管道是一种数据传输渠道，它使用自动化处理步骤将数据移动到目标存储库。管道的数据来源（有时称为数据管道定义）是数据通过管道移动之前数据收集的终点。数据在管道中的移动称为数据流。管道架构可以是批处理或流处理类型，连续加载架构介于两者之间。

一种数据收集方法称为提取、转换、加载（ETL）方法，其中大批量数据在特定时间移动，不同的存储区域在数据通过管道时保留数据。ETL 工具需要处理引擎来完成转换，然后才能将数据加载到目标存储库中。当目标需要与管道定义不同的特定数据时，ETL 可能是一个理想的解决方案。

作为 ETL 的替代方案，有一个称为提取、加载、转换（ELT）的数据管道过程，该流程在数据进入存储库后使用目标的处理引擎来转换数据。这是一种更简化的方法，因为它消除了中间步骤，并允许组织仅在必要时转换其原始数据。当目标是云原生数据仓库时，ELT 方法通常是首选方法。

在 ELT 方法中，Snowflake Snowpipe 是提取部分。可以执行某些转换，例如更改列名称或更改列的顺序，然后命令将数据加载到目标目标中。



Snowpipe 支持半结构化数据类型，例如 JSON 和 Avro。

要使用 Snowpipe 完成 ELT 数据管道，我们还需要使用 Snowflake 对象（如流和任务）来转换需要字符串连接或数字计算等内容的数据。

使用 Snowflake 的 Snowpipe 时，有两种不同的机制可以检测暂存文件何时可用：使用云消息收发自动化 Snowpipe 和调用 Snowpipe REST 终端节点。

需要注意的一点是，您可以通过提取较小的数据文件来加速 Snowpipe。将大文件分块为较小的文件不仅可以让 Snowflake 更快地处理数据，还可以更快地提供数据。



根据经验，加载到 Snowpipe 中的最佳文件大小是 100 到 250 MB 的压缩数据。如果数据连续到达，请尝试在 60 秒的间隔内暂存数据。

请记住，可以以这样一种方式创建 Snowpipe，从而减少延迟并显著提高吞吐量，但需要权衡架构设计与随着触发更频繁的文件提取而增加的 Snowpipe 成本。

虽然 Snowpipe 延迟的减少对于大多数近乎实时的坐姿来说可能已经足够好了，但有时 Snowpipe 的速度可能不够快。电子商务和 IoT 是可能需要实际实时数据处理的两个示例。在这些情况下，我们需要使用纯流处理工具，例如 Confluent 的 KSQL（直接在 Kafka Stream 中处理数据），或者 Apache Flink 和 Apache Flume。

接下来的部分将提供有关 Kafka 和 Snowpipe 的更多详细信息，它们是持续加载数据管道的最常见选择。



在自动摄取或 REST API 这两种 Snowpipe 方法之间做出决定时，需要考虑一些不同的因素。在文件不断到达的情况下，您可以使用 Snowflake 的自动摄取功能创建事件通知。Snowpipe 自动提取是更具可扩展性的方法。对于数据随机到达和/或预处理需求需要使用 ETL 或 ELT 工具的使用案例，或者外部阶段不可用的情况，REST API 是更好的选择。

使用 Apache Kafka

Apache Kafka 是用于将 Kafka 连接到外部系统的框架。Kafka 最初创建的目的是启用消息队列工作负载，其中包括发布者和订阅者模型中的创建者和使用者（如图 6-31 所示）。这

Kafka 生产者和使用者 API 以其简单性而闻名。数据以异步方式发送，这会导致回调。尽管 Kafka 生产者 API 可以扩展和构建，但不建议将 Kafka 生产者 API 或 Consumer API 用于 ETL 方法。相反，您应该考虑使用 Kafka Connect 源 API。

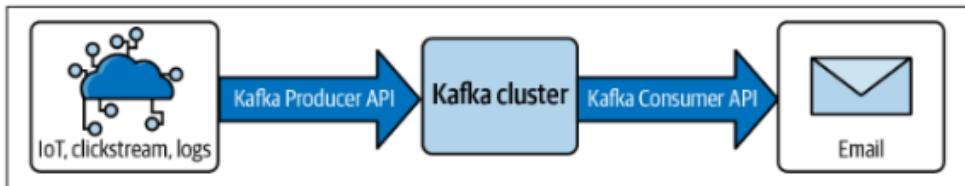


图 6-31。Kafka 生产者和使用者工作负载

如今，Kafka 架构包括其他工作负载。Kafka Streams API 和 KSQL 可用于希望从 Kafka 消费并生成回 Kafka 的应用程序（如图 6-32 所示）。Kafka Streams API 是一个用 Java 语言编写的流处理库，如果您希望编写复杂的逻辑，建议使用它。或者，如果您想编写类似 SQL 的实时作业，则可以使用 KSQL。

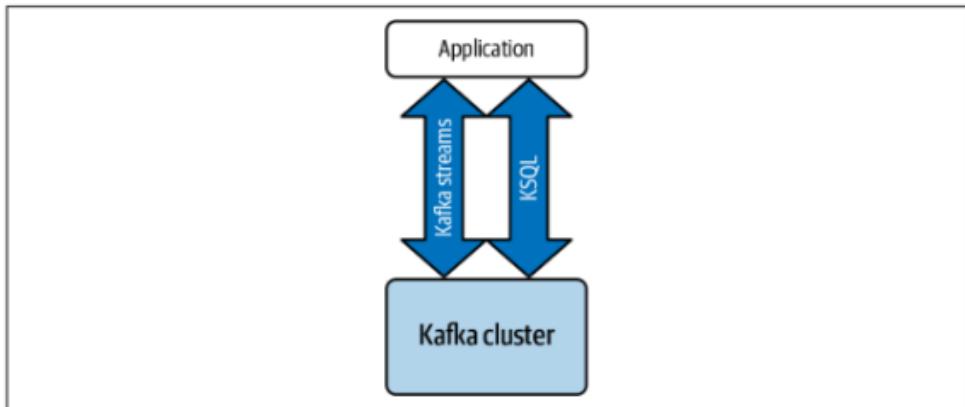


图 6-32. 用于实时作业的 Kafka Streams 和 KSQL

Kafka Streams 要求进行自定义编码，并且它确实带有联接和聚合。与 Kafka 连接器不同，Kafka Streams 包括 Exactly-once 处理本机功能。Apache Kafka 还可以与 Apache Flink、Apache Spark 和 Apache NiFi 等外部处理系统配合使用进行流处理。

另一种 Kafka 方法是 Kafka 连接器源，通常用于在 Kafka 和数据存储（如 Twitter）和 Kafka 连接器接收器之间架起桥梁。图 6-33 中所示的 Kafka 连接器是我们在本节中重点介绍的内容。

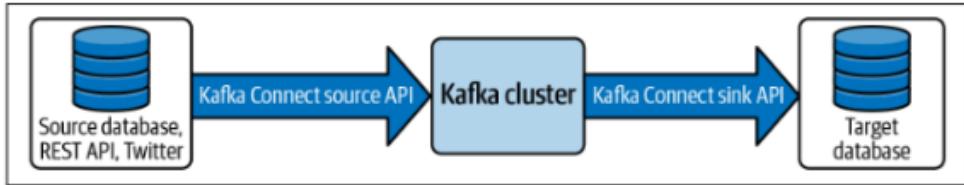


图 6-33。Kafka 连接器源 API 和 Kafka 连接器接收器 API

Kafka 在由一个或多个服务器（称为代理）组成的集群上运行，并将所有 Kafka 主题分区以分布在集群节点之间。Kafka 主题是记录行存储到的类别或源名称。

一个 Kafka 主题处理一个消息流，该消息流由行组成，然后插入到 Snowflake 表中。在 Kafka 配置中，可以将主题映射到现有的 Snowflake 表（如图 6-34 所示）。对于任何未映射的主题，Kafka 连接器使用主题名称创建一个新的 Snowflake 表。

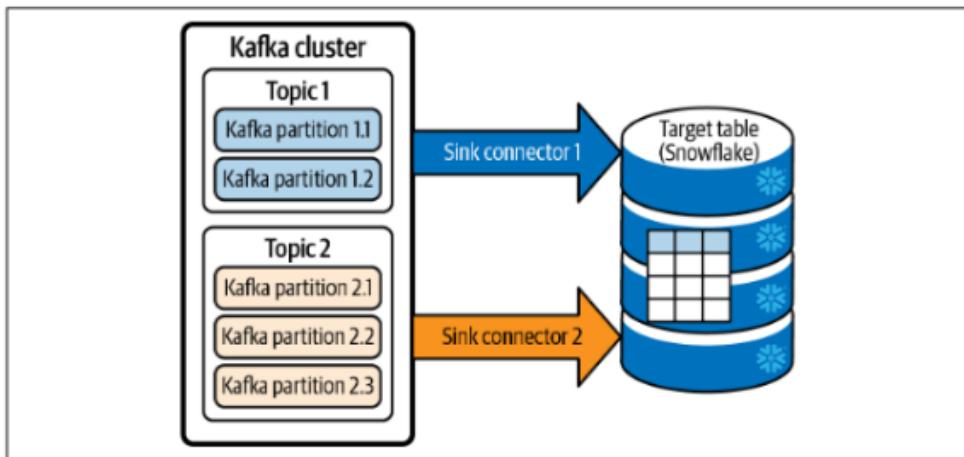


图 6-34。Kafka 主题 1 映射到现有的 Snowflake 表，而未映射的主题 2 将导致 Kafka 连接器创建一个新表

创建新表时，Kafka 连接器将使用以下规则将主题名称转换为 Snowflake 表名称：

- 小写主题名称将转换为大写表名称。
- 连接器会在表名称前面加上下划线，除非主题中的第一个字符是大写或小写字母或下划线字符。
- 下划线字符将替换不是 Snowflake 表名称合法字符的任何主题名称字符。



建议您创建的任何 Kafka 主题名称都应遵循 Snowflake 标识符名称的规则。这样，就不需要低于分数的字符来替换 Snowflake 表名非法字符。

现在我们已经对 Kafka Connect 有了基本的了解，让我们看看使用 Kafka 连接器将 Kafka 摄取流引入 Snowflake 表（如图 6-35 所示）。请务必注意，您可以通过为 Snowflake 登录凭证、主题名称、Snowflake 表名称等内容创建具有指定参数的文件来配置 Kafka 连接器。



Kafka 连接器可以从多个主题中提取消息，但一个配置文件中列出的连接器的相应表必须全部存储在单个数据库和架构中。

使用 Kafka 连接器本身不会产生任何费用。但是，会产生 Snowpipe 费用，因为 Snowpipe 用于加载从 Kafka Connector 读取的数据。



关于 Snowpipe 计费，需要记住的一些重要事项是 Snowpipe 不需要虚拟仓库，因为它是一个无验证模型。在无服务器模型中，Snowflake 根据负载提供并自动管理计算资源。此外，每 1000 个通知的文件需要 0.06 个积分的使用成本。使用成本适用于 Snowpipe REST 和 Snowpipe AUTO_INGEST。

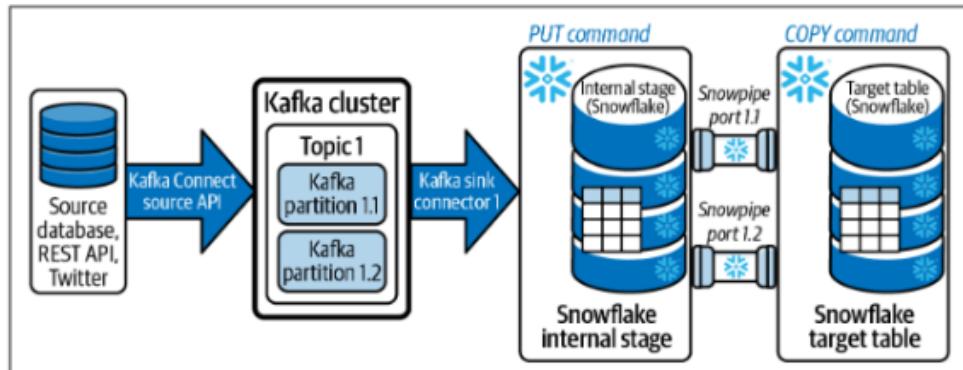


图 6-35。Kafka 提取流向使用 Kafka 连接器的 Snowflake 表

从图 6-35 中，我们可以看到 Kafka Connect 源 API 从源数据库、REST API 或 Twitter 等应用程序接收记录。Kafka 集群接收这些记录，并将每个主题拆分为一个或多个 Kafka 分区。需要注意的是，在我们的示例中，我们只在图 6-35 中包含一个主题和接收器连接器，但 Kafka 集群可以有多个主题，因此会使用多个接收器连接器。

经过一定时间或收到一定数量的记录后，Kafka 接收器连接器会使用命令将消息写入 Snowflake 内部命名阶段。然后，接收器连接器触发 Snowpipe 以提取暂存的文件。您会在图 6-35 中注意到，有两个 Snowpipes，因为有两个 Kafka 主题分区。

Snowflake Snowpipe 无服务器计算集群将使用该命令将数据从内部 Snowflake 阶段加载到 Snowflake 表中。确认数据加载成功后，将删除 internal stage 中的文件。

当 Kafka 连接器创建 Snowflake 表时，

VARIANT 数据类型。第一列是包含 Kafka 消息的 RECORD_CONTENT col – umn。



Kafka 消息以 JSON 或 Avro 格式传递给 Snowflake，每条消息都存储在一个 VARIANT 列中，无需解析数据。消息中的字段名称和值区分大小写。

创建的第二个 VARIANT 列是 RECORD_METADATA 列，其中包含有关消息的元数据。此外，元数据包括主题和主题中的 Kafka 分区数。



可以创建自定义 Kafka 代码来执行正在进行的转换或更快地提供数据。在这些情况下，sink 连接器将被使用 JDBC 批量插入语句的自定义代码替换。

Kafka 连接器只是可用于数据集成的众多连接器之一。有关 Snowflake 连接器和驱动程序的完整信息，您可以查看 Snowflake 开发人员文档。

使用云消息收发和可选的本地 Kafka 集群实现 Snowpipe 自动化

Snowflake Snowpipe 可以使用来自 AWS、Azure 或 GCP 的云消息收发实现自动化。自动引入方法使用云消息事件通知和 Snowflake 外部舞台。外部舞台很重要，因为自动 Snowpipe 仅适用于 Snowflake 外部舞台对象。



或者，结合使用云消息收发的自动化 Snowpipe，可以利用本地 Kafka 集群将主题流式传输到其中一个云提供商，在那里收集和存储消息，然后使用事件通知自动提取消息。

对于构建 Snowflake 的三个云提供商中的每一个，都有一个可用于云消息传递的关联引用事件通知类型（如图 6-36 所示）。

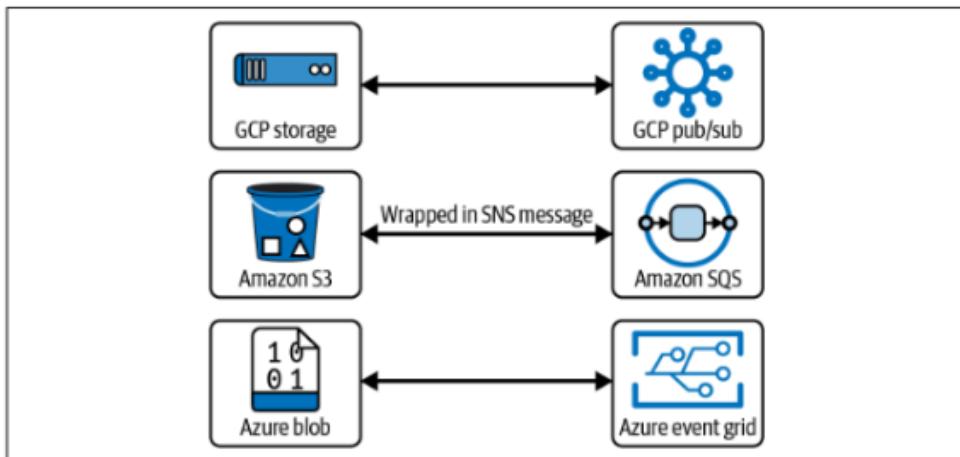


图 6-36。GCP、AWS 和 Microsoft 的事件通知类型

云消息收发服务会通知 Snowpipe 新数据文件的到来。以连续无服务器方式，Snowpipe 将数据加载到目标表中（如图 6-37 所示）。

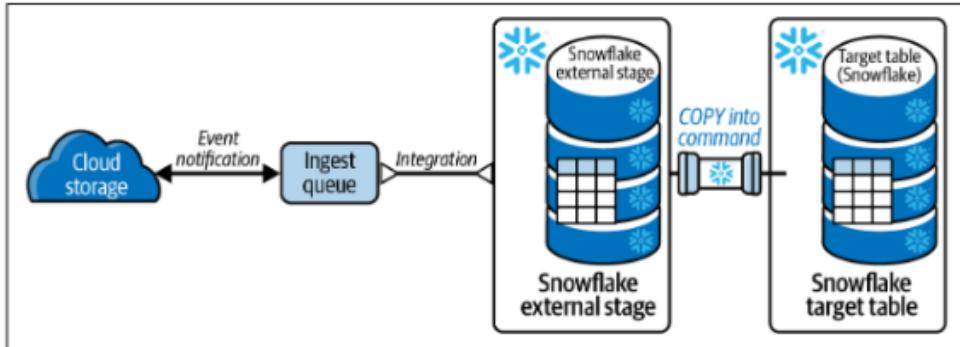


图 6-37。使用云消息收发的 Snowpipe 自动摄取

让我们考虑一个如何使用 Snowpipe 自动摄取将数据加载到 Snowflake 中的示例。我们将以 Microsoft Azure 为例，从高层次上考虑我们需要的信息，然后我们看看所需步骤。

首先，您需要从 Azure 分配的几条信息：

- 租户 ID • 存储队列通知 URL，其格式类似于
`https://sflakesnowpipe.queue.core.windows.net/snowdata-queue` • 应用程序名称，其格式类似于 SnowflakePACInt1025 • 主 Azure Blob 存储终结点，其格式类似于 `https://sflakesnowpipe.blob.core.windows.net/`

- Blob 服务共享访问签名，其形式类似于
`https://sflakesnowpipe.blob.core.windows.net/xxxx`

在 Azure 中，您需要创建以下项：资源组、存储帐户名称、容器名称、队列名称、事件订阅名称和系统主题名称。

您还需要获取在 Snowflake 中分配的 Azure 同意 URL。它将采用类似于
`https://login.microsoftonline.com/xxxx` 的形式。

在 Snowflake 中，您需要确保已创建数据库、架构、包含字段的表和阶段。您还需要在 Snowflake 中创建集成和管道。



指向 Snowflake 集成的阶段链接是使用隐藏的 ID 而不是集成的名称创建的。因此，如果重新创建集成，则 Snowflake 集成与引用它的任何阶段之间的关联将被打破，即使使用相同的名称也是如此。

以下是创建用于从 Microsoft Azure 加载数据的 Snowpipe 自动摄取方法所需采取的步骤：

1. 登录您的 Azure 帐户并获取您的租户 ID。
2. 在 Azure 中，创建以下项（如果尚不存在）：资源组、资源组中的存储帐户、容器、队列和事件订阅。
3. 在 Azure 中，获取队列 URI。
4. 在 Snowflake 中，创建以下项（如果尚不存在）：database、schema 和 table。
5. 在 Snowflake 中，将你的角色设置为 `INTEGRATOR` 并创建通知集成：
 创建通知集成 <集成名称>
 启用 = TRUE
 类型= 队列
 NOTIFICATION_PROVIDER=AZURE_STORAGE_QUEUE
 AZURE_STORAGE_QUEUE_PRIMARY_URI = ''
 AZURE_TENANT_ID = '74B1C3A'

6. 在 Snowflake 中，获取 AZURE 同意 URL：

```
DESC 通知集成 <集成名称>;
```

7. 在 Azure 中，添加角色分配并获取终端节点。

8. 在 Azure 中，生成共享访问签名。

9. 在 Snowflake 中，创建一个外部舞台。

10. 在 Snowflake 中，创建一个 Snowpipe: CREATE PIPE

```
AUTO_INGEST=TRUE INTEGRATION=<集成名称>作为 COPY  
INTO FROM @ FILE_FORMAT= (TYPE='JSON') ;
```



要解决与数据问题相关的数据加载错误，强烈建议您在后加载验证过程中使用该函数，以确保所有文件都已成功完全加载到目标表中。

本节提供了有关自动提取 Snowpipe 的基本信息。您可以在 Snowflake 网站上找到更多详细信息，在那里您可以

了解有关自动化 Snowpipe 的云存储服务支持的更多信息。值得注意的是，跨云支持目前仅适用于托管在 Amazon Web Services 上的账户。

除了自动摄取 Snowpipe 之外，Snowflake 还提供了一个 REST API 选项，用于将数据触发到 Snowpipe。一个重要的注意事项是，Snowpipe REST API 可与内部和外部 Snowflake 阶段配合使用。

调用 Snowpipe REST 终端节点

触发数据加载事件的另一种方法是使用 Snowflake 的 Snowpipe REST API 方法。使用此方法，将调用具有管道名称和文件名列表的公共 REST 终端节点。如果已识别与文件名列表匹配的新数据文件，则它们将排队等待加载。然后，Snowpipe 无服务器计算资源会将排队的文件加载到 Snowflake 目标表中。

云存储服务支持来自 Snowflake 账户的 Snowpipe REST API 调用，允许对 Amazon Web Services、Google Cloud Platform 或 Microsoft Azure 上托管的所有账户提供跨云支持。

表 6-4 介绍了使用 Snowpipe 的两种方法之间的差异。

表 6-4 Snowpipe REST 和 Snowpipe AUTO_INGEST 之间的区别

Snowpipe REST 和 Snowpipe AUTO_INGEST	可用于内部和外部阶段	仅适用于外部阶段	手动调用 Snowpipe REST API 终端节点，使用管道名称和文件名列表	当新文件到达时，会收到来自云提供商的通知
-	-	-	- 在舞台位置传递文件列表	- 唤醒时处理新文件

如果您想了解有关 Snowpipe 的更多信息，Snowflake 提供了 Snowpipe 快速入门。正如我们所了解的，Snowpipe 可能会增加流数据的延迟。它还要求您在将数据加载到表中之前将数据存储在一个阶段中，这会增加存储成本。为了帮助解决这些问题，Snowflake 创建了一个新的流式处理 API。这个新的 API 允许您使用 Streaming Ingest SDK，该 SDK 可以使用 Java 实现。然后可以创建要插入的值和行的映射。如果您目前正在使用 Kafka 作为解决方案的一部分，则更改属性并开始利用新的 Snowpipe Streaming 功能相对容易。

第三方 ETL 和 ELT 工具

Snowflake 支持在加载期间转换数据（ETL）的工具，以及在加载后转换数据（ELT）的工具。许多不同的第三方集成工具

由 Snowflake 原生支持，包括 Fivetran、Informatica、Matillion、Talend 等。您可以在 Snowflake 开发人员文档页面中查看 Snowflake 支持的集成工具列表。

加载数据的替代方法

可以对未加载到 Snowflake 中的数据执行查询。一个例子是针对数据湖中的数据创建的外部表。使用 Snowflake 外部表，您可以查询存储在外部云存储中的现有数据，其中数据仍作为事实来源，而无需将数据加载到 Snowflake 中。当外部云存储中存在大量数据，但只需要一部分数据进行分析时，通常会使用此解决方案。在这种情况下，我们可以在外部表上创建一个物化视图，以便只使用数据的子集；这可以提高查询性能。

外部表和物化视图在 Chapter 4 中介绍。有关物化视图的更多详细信息也包含在第 9 章中。

将数据加载到表中的另一种替代方法是使用 Snowflake 数据共享来访问其他人拥有并与您共享的数据。Snowflake Marketplace 包括许多向公众提供的数据产品。第 10 章包括所有 Snowflake 安全数据共享选项的详细信息。

数据加载的另一种替代方法是克隆表。使用该命令克隆表会导致使用与现有表相同的数据创建新表。还可以复制包含现有数据的表。用于克隆表的语句示例如下：

```
使用角色 SYSADMIN;使用 SCHEMA WS;CREATE  
TABLE DEMO_CLONE CLONE TABLE1;
```

最后，加载数据的一种较新的替代方法是使用 Snowflake 的 Snowpark 功能。Snowpark 提供了在数据管道中查询和处理数据的能力，而无需将数据移动到运行应用程序代码的系统。Snowpark 在第 12 章中有更详细的描述。

卸载数据的工具

从 Snowflake 卸载数据的过程与数据加载过程相同，只是相反。当目标位置是本地文件系统时，首先使用命令将数据卸载到 Snowflake 内部阶段。然后使用 command 从内部阶段下载到本地文件系统（如图 6-38 所示）。

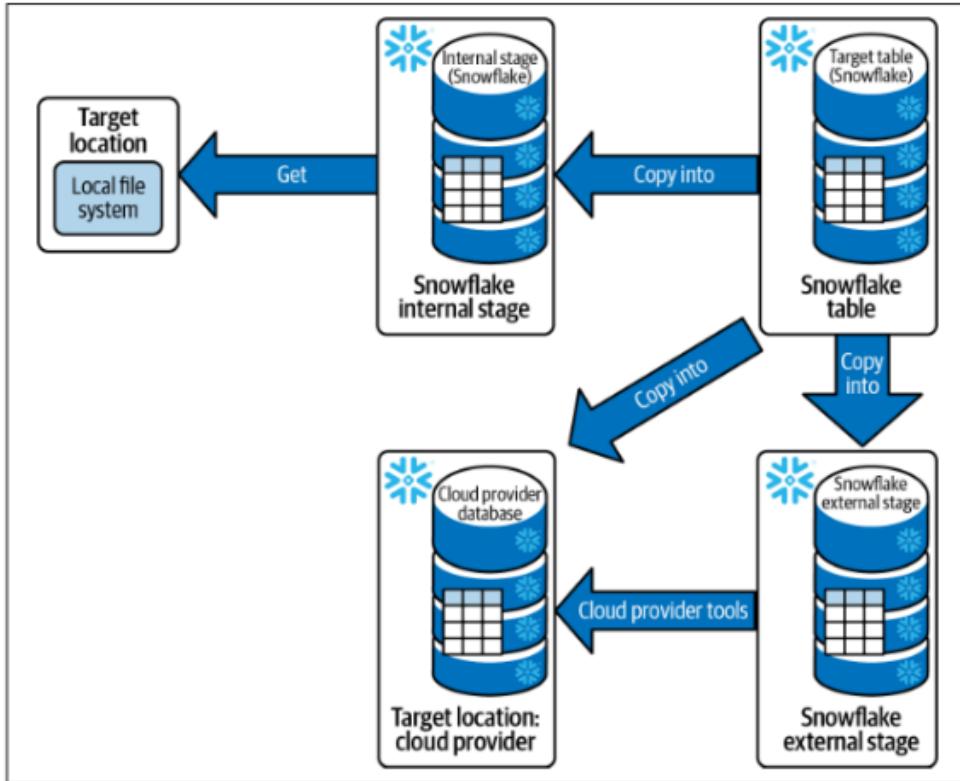


图 6-38。将数据从 Snowflake 表下载到目标位置的多种方式

对于卸载到三个主要云提供商之一的数据，有两个选项。数据文件可以直接卸载到云专业人士的存储位置，Snowflake 帐户就是基于该帐户构建的。例如，如果您选择 Google Cloud 作为 Snowflake 提供商，则可以将文件直接卸载到 GCP 容器，然后使用 Google Cloud Storage 实用程序在本地下载。或者，您可以使用该命令将文件从 Snowflake 表提取到 Snowflake 外部舞台中，然后使用云提供商工具下载到存储中。

面向 Snowflake 数据工程师的数据加载最佳实践

本章提供了有关将数据加载到 Snowflake 中的详细信息。本章中提供了一些建议和技巧，但在本部分中为 Snowflake 数据工程师总结一些最佳实践是有意义的。

选择正确的数据加载工具并考虑适当的数据类型选项

数据加载工具的选择受许多因素影响：加载是重复过程还是一次性加载，最终用户是实时需要数据还是近乎实时地需要数据，要加载的数据的数据类型，团队的现有技能集等等。牢记这些因素，我们仍然可以提出一般的最佳实践建议。

在 ETL 方法中，可以使用第三方工具在将数据加载到 Snowflake 之前对其进行转换。不过，由于 Snowflake 具有横向扩展能力，因此采用 ELT 方法而不是 ETL 方法进行数据加载通常更有意义。在 ELT 方法中，可以完成一些转换，例如更改列名称或更改列的顺序。引入数据后，可以使用 Snowflake 流和任务或存储过程完成更复杂的数据转换，例如字符串连接或数字计算。还有使用 Snowpark API 和 Java UDF 转换数据的强大选项。

JDBC 和 ODBC 连接器在某些情况下是不错的选择，但它们不应该是大型常规数据加载的首选工具。

在考虑正在加载的数据时，在创建 Snowflake 表时选择合适的数据类型非常重要。例如，日期和时间戳数据类型在 Snowflake 中的存储效率高于数据类型。

应遵守数据文件大小限制和建议，尤其是对于半结构化数据。大于 3 GB 的 Parquet 文件可能会超时，因此建议将它们拆分为 1 GB 或更小的大小。

了解将数据原生加载到 Snowflake 中的一些独特方法非常重要，因为它为您可以考虑的工具提供了可能性。您应该考虑使用的独特 Snowflake 工具的两个示例是 Snowpipe 和 Snowflake 支持的许多半结构化数据类型的文件格式。

避免逐行数据处理

在涉及逐行数据处理的情况下，查询性能会受到很大影响。鉴于 Snowflake 旨在轻松提取和处理数十亿行，因此您设计的数据处理应侧重于处理整个数据集，而不是一次处理一行。逐行数据处理的负面影响怎么强调都不为过。

选择合适的 Snowflake 虚拟仓库大小并根据需要拆分文件

为防止资源争用，请务必通过为每个作业指定单独的虚拟仓库来将数据加载作业与查询隔离开来。与其假设较大的虚拟仓库加载大量数据文件的速度一定比较小的虚拟仓库快（很可能不会），不如尝试将大文件拆分为大小约为 100 到 250 MB 的较小文件。请记住，正在加载的文件数量和每个文件的大小对性能的影响比对虚拟仓库大小的影响更大。

也就是说，请务必注意，在使用命令时，必须选择正确的 Snowflake 虚拟仓库大小。但是，根据使用案例，无服务器方法可能更理想。如果需要无服务器方法，可以使用 Snowpipe。在这种情况下，无需担心虚拟仓库的规模。

分步骤转换数据，并使用瞬态表获得中间结果

在适当的时候，分多个步骤转换数据，而不是使用大量的 SQL 语句。这应该会产生更简单的代码，并可以测试中介间结果。对于中间结果，建议您使用临时表，并在下次数据加载之前将其截断，以便降低按时间旅行存储费用。

代码清理

DEMO6_DB，因为我们在一个数据库中创建了所有表，所以我们的清理很简单。我们不能放弃我们目前正在使用的虚拟仓库；因此，我们需要先更改 Virtual Warehouse：

```
USE WAREHOUSE COMPUTE_WH;
DROP DATABASE DEMO6_DB;
DROP WAREHOUSE LOAD_WH;
```

总结

在本章中，我们深入研究了几种不同类型的数据加载工具，这些工具既可以手动将几行数据插入表中，也可以创建可用于加载大量数据文件的自动化流程。我们还考虑了 Snowflake 数据工程师的一些数据加载最佳实践。

但是，仅仅知道如何以最佳方式将数据加载到 Snowflake 中是不够的。我们还需要确保我们对我们能做到的方式有很好的理解和欣赏

保护我们的数据。下一章提供了更广泛的理解和欣赏数据治理和数据安全所需的重要后续步骤。

知识检查

以下问题基于本章中包含的信息：

1. 半结构化数据使用的三种 Snowflake 数据类型是什么？这三者中的哪一个通用数据类型？
2. Snowflake 舞台有哪些类型？阶段是 Snowflake 对象吗？解释。
3. SQL 语句在什么情况下使用？
4. 如果键值对在插入时包含 null 值，会发生什么情况？
5. 使用“加载数据”向导时，有哪些错误处理选项，哪一个是默认选项？

6. 是否可以在使用命令时执行转换？
7. Snowflake 支持哪些半结构化数据类型进行加载和卸载？
8. 在两种 Snowpipe 方法中，哪种方法的可扩展性更强？何时最好使用每种方法？
9. 数据加载的最佳实践示例是什么，因为它与虚拟仓库特别相关？

10. 什么是将数据加载到阶段的 Snowflake 命令和用于从阶段卸载数据的命令？

这些问题的答案可在附录 A 中找到。

实施数据治理， 帐户安全以及数 据保护和恢复

关于数据管理和账户安全这两个非常广泛的主题，已经写了很多文章。虽然本章并非旨在让您成为这些主题的专家，但它将向您介绍许多 Snowflake 数据安全功能和管理工具，这些工具将使您对在 Snowflake 中实施数据治理和安全策略的多种方式有更广泛的了解和欣赏。

传统上，数据治理的重点是让合适的人员在合适的时间使用合适的技术使用合适的数据执行合适的流程。Data Governance 2.0 框架（如图 7-1 所示）通过强调业务价值主张来扩展该重点。具体来说，与数据可发现性相关的数据共享领域提供了如此多的商业价值，以至于第 11 章专门讨论了这个主题。

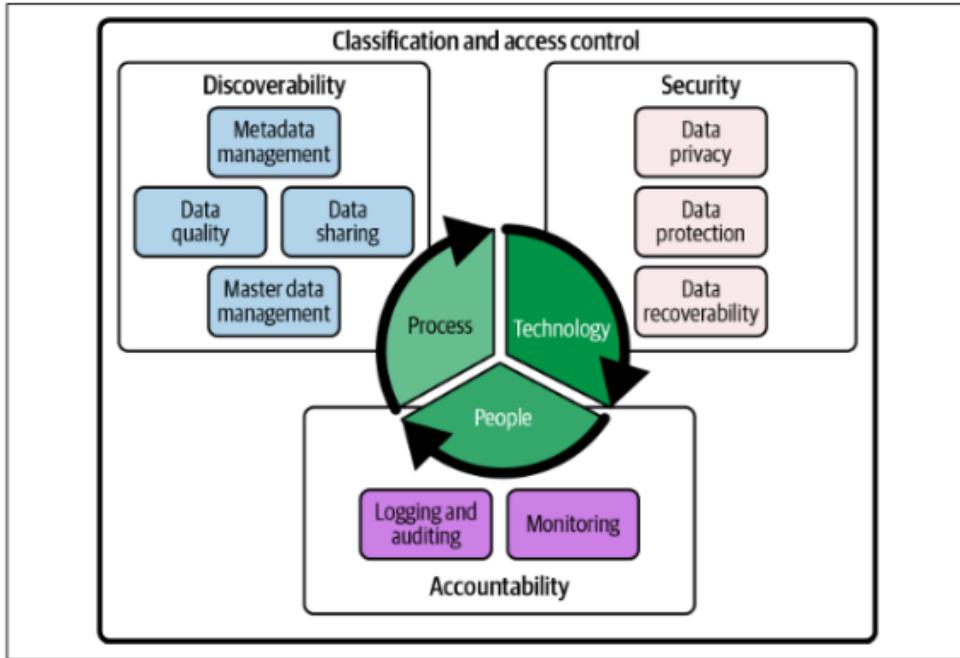


图 7-1. 数据治理 2.0 框架

本章的重点是通过使用功能强大但易于使用的 Snowflake 功能，在 Snowflake 中实施数据管理 2.0 框架中的概念。例如，您将学习如何管理网络安全策略和防火墙访问。您还将了解有关 Time Travel 和故障保护、数据库复制和故障转移/故障恢复的更多信息。此外，本章还包括许多示例，说明您可以通过数据管理控制（例如对象标记、分类、数据屏蔽、行访问策略、外部标记化和匿名化）实现数据民主化。本章中的几个动手示例旨在加深您对 Snowflake 中的数据治理、账户安全以及数据保护和恢复的知识和理解。

准备工作

我们将通过一些示例来演示 Time Travel、对象标记、动态数据掩码和行级安全性在 Snowflake 中的工作原理。在开始每个部分之前，我们需要做一些设置工作来创建一些角色和表并分配一些权限。这也将锻炼您在前面章节中学到的一些技能。

创建一个标题为第 7 章安全、保护、数据恢复的新文件夹。如果您需要帮助创建新文件夹，请参阅第 8 页上的“导航 Snowsight 工作表”

和 worksheet 的要设置工作表上下文，请确保您使用的是 SYSADMIN 角色和 COMPUTE_WH 虚拟仓库。

在本章的示例中，我们跟踪一位名叫 Adam 的新用户，并使用他的 credentials 来确认已授予适当的访问权限。



警告
您的实际电子邮件地址将无法按预期工作。请务必在您的电子邮件地址周围留下单引号。

现在让我们为 Adam 创建一个新的用户帐户：

```
使用角色 USERADMIN;
创建或替换用户 ADAM
密码 = '123'
LOGIN_NAME=亚当
DISPLAY_NAME=亚当
电子邮件 = 'YOUREMAIL@EMAIL.COM'
MUST_CHANGE_PASSWORD=真;
```

我们还需要一些新角色。我们将使用 Human Resources 角色 HR_ROLE 来测试我们的动态数据掩码。稍后将使用这两个 AREA 角色来演示行级安全性：

```
使用角色 USERADMIN;
创建或替换角色HR_ROLE;创建或替换角色
AREA1_ROLE;创建或替换角色AREA2_ROLE;
```

SYSADMIN 角色将用于创建我们将使用的一些对象：

```
使用角色 SYSADMIN;使用 WAREHOUSE COMPUTE_WH;创建或替换数据库DEMO7_DB;创建
或替换 SCHEMA TAG_LIBRARY;创建或替换架构 HRDATA;创建或替换 SCHEMA CH7DATA;
创建或替换表 DEMO7_DB.CH7DATA.RATINGS (EMP_ID整数、RATING 整数、DEPT_ID
varchar、AREA 整数);
```

我们还需要插入一些任意数据：

```
INSERT INTO DEMO7_DB 中。CH7DATA。评分值 (1,
77, '100', 1), (2, 80, '100', 1),
(3, 72, '101', 1),
(4, 94, '200', 2),
(5, 88, '300', 3),
(6, 91, '400', 3);
```

接下来，我们将向角色授予一些权限，然后将这些角色分配给我们的新用户。我们将使用 SECURITYADMIN 角色向三个新角色授予虚拟仓库使用权，以便它们能够执行查询操作。

使用角色 SECURITYADMIN: 将仓库 COMPUTE_WH 的使用权限授予角色
HR_ROLE; 将仓库 COMPUTE_WH 的使用权限授予角色 AREA1_ROLE; 将
WAREHOUSE COMPUTE_WH 的使用权限授予角色 AREA2_ROLE;

我们还需要将这些角色授予必要的对象使用权：

将数据库 DEM07_DB 的使用权限授予角色 HR_ROLE; 将数据库 DEM07_DB 的使用权限授予角色
AREA1_ROLE; 将数据库 DEM07_DB 的使用权限授予角色 AREA2_ROLE; 授予对 SCHEMA DEM07_DB 的使
用权。CH7DATA 角色 HR_ROLE; 授予对 SCHEMA DEM07_DB 的使用权。HRDATA 转换为角色 HR_ROLE;
授予对 SCHEMA DEM07_DB 的使用权。CH7DATA 角色 AREA1_ROLE; 授予对 SCHEMA DEM07_DB 的使
用权。CH7DATA 角色 AREA2_ROLE; 对架构 DEM07_DB 中的所有表授予 SELECT 权限。CH7DATA 角色
HR_ROLE; 对架构 DEM07_DB 中的所有表授予 SELECT 权限。CH7DATA 角色 AREA1_ROLE; 对架
构 DEM07_DB 中的所有表授予 SELECT 权限。CH7DATA 角色 AREA2_ROLE;

最后，我们将要将这三个角色分配给我们的新用户。其中两个角色也将被分配回
SYSADMIN 角色。我们选择不将 HR_ROLE 分配回 SYSADMIN 角色：

将角色 HR_ROLE 授予用户 ADAM; 将角色 AREA1_ROLE 授
予用户 ADAM; 将角色 AREA2_ROLE 授予用户 ADAM; 将
角色 AREA1_ROLE 授予角色 SYSADMIN; 将角色
AREA2_ROLE 授予角色 SYSADMIN;

我们现在需要使用 ACCOUNTADMIN 角色向 HR_ROLE 授予将来的 select 和 insert 权
限。之后，让我们将角色重置回 SYSADMIN 角色：

使用角色 ACCOUNTADMIN; 在 SCHEMA DEM07_DB 中对 FUTURE TABLES 授予 SELECT 权限。
HRDATA 转换为角色 HR_ROLE; 在 SCHEMA DEM07_DB 中对 FUTURE TABLES 授予 INSERT。HRDATA
转换为角色 HR_ROLE; 使用角色 SYSADMIN;

Snowflake 安全性

确保您的 Snowflake 帐户和数据安全是 Snowflake 和您共同努力的结果。Snowflake 负责 Data Cloud 中传输的数据的安全性，以及存储在表中的静态数据的安全性。这是通过使用强大的加密方法来保护数据免受外部 Par-Ties 的影响来实现的。Snowflake 为所有 Snowflake 账户提供 Soc 1 Type II 和 Soc 2 Type II 合规性安全验证。此外，Snowflake 每年都会执行许多渗透测试，并提供对 HIPAA 合规性、PCI DSS 合规性和

FedRAMP 在某些区域，Snowflake 的业务关键版及更高版本具有中等合规性。Snowflake 已获得 FedRAMP 的中等授权，与医疗保险和医疗补助服务中心、CSA 以及卫生与公众服务部等机构合作。Snowflake 支持由第三方组织审核的 ITAR 合规性。Snowflake 正在获得 FedRAMP High 授权，预计将于 2022 年冬季获得，具体取决于机构和 PMO 的可用性。与此同时，Snowflake 正在寻求 IL4，并打算在 2023 年成为 IL4 作者。Snowflake 也是 HITRUST 责任共担和继承计划（SRM）的一部分，并已获得 HITRUST 认证。HITRUST CSF 是医疗保健行业的领先信息安全框架。Snowflake 还提供了确保网站访问安全、保护数据、监控用户活动以及在必要时恢复数据所需的工具。在本节中，我们将探讨其中的许多工具，包括原生 Snowflake 身份验证方法、网络安全策略、通过访问历史记录进行 Snowflake 查询监控、时间旅行、故障安全以及复制和故障转移。

控制账户访问

想要访问 Snowflake 的用户需要识别和验证自己，以证明自己的身份。该证明用于验证或验证用户确实是他们声称的身份。除了对用户进行身份验证以便仅向允许的用户授予访问权限外，Snowflake 还使管理员能够创建安全策略，以限制用户可以登录的位置。

当然，访问控制太少会导致风险增加。但是，要求用户跳过太多的障碍也会带来风险。当用户遇到过多的用户身份验证摩擦或其他过于严格的访问控制时，人们倾向于通过参与影子 IT 活动来避免复杂性，这通常会导致信息孤岛。虽然这些影子 IT 活动很可能在某个时候被发现，但最好考虑如何在实施过多和不足的控制之间取得适当的平衡。

身份验证和用户管理

Snowflake 支持的身份验证方法可以轻松适应用户登录 Snowflake 的不同方式。到目前为止，我们已经演示了用户如何尝试直接访问 Snowflake 数据。或者，用户可以间接访问 Snowflake。通过商业智能（BI）工具间接访问 Snowflake 就是一个例子。还可以建立集成用户，以提供一种更安全、更可审计的方式将数据移入和移出 Snowflake，而无需依赖现有用户帐户。例如，为 Snowpipe 创建集成用户是最好的实践方法。对于集成用户，密钥对是首选的身份验证方法。

对于通过第三方应用程序访问 Snowflake 数据的用户，OAuth 集成是首选，因为它允许在不共享用户密码的情况下访问数据，并且将身份验证和授权分离。不需要存储凭证，因为会使用令牌并将其绑定到 Snowflake 角色。

当用户需要直接访问才能在 Snowflake 中工作时，他们可以使用联合身份验证进行身份验证。对于联合身份验证，使用身份提供商（IdP）（如 Okta 或 Microsoft Active Directory 联合身份验证服务（AD FS））通过单点登录（SSO）对用户进行身份验证。

对于不使用 SSO 的 Snowflake 帐户，Snowflake 提供了多重身份验证（MFA）本机解决方案，该解决方案支持通过 Web UI 进行连接、使用 SnowSQL 命令行界面（CLI）进行连接以及 JDBC 和 ODBC 客户端连接。MFA 旨在与强密码一起使用。Snowflake 为所有账户级别提供自助式 MFA，以便用户可以自行注册。

要从 Web UI 注册 MFA，请使用 Home（主页）图标转到 Main（主菜单）。在此处的左上角，单击您姓名右侧的向下箭头。选择 Profile（个人资料），然后单击“Multi-factor authentication”（多重身份验证）旁边的 Enroll（注册）按钮。接下来，单击“开始设置”按钮（如图 7-2 所示）。

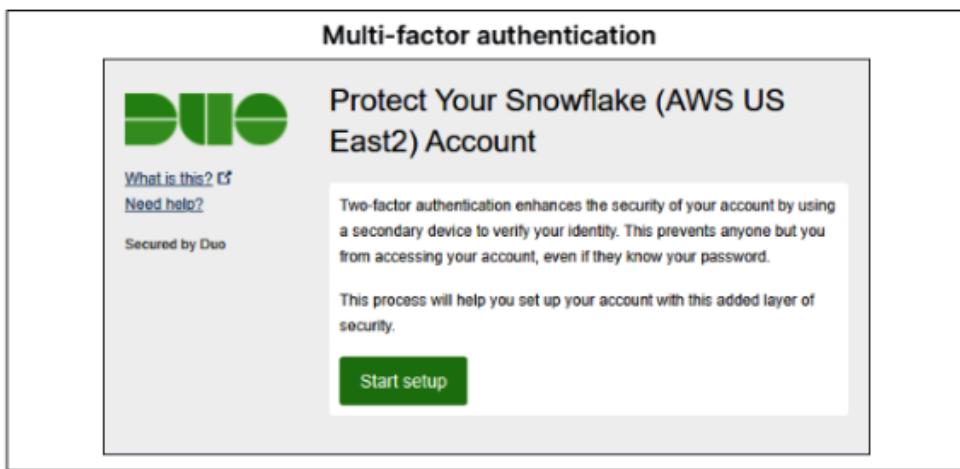


图 7-2. Multifactor Authentication 设置屏幕

从那里，您可以指定要添加的设备类型。在完成注册过程时，您需要安装 Duo Mobile 应用程序（见图 7-3）并扫描 QR 码。

身份验证有助于控制帐户访问，但只是确保数据安全所需的一部分。用户通过身份验证后，他们可以看到的数据以及他们可以在 Snowflake 中执行的活动（例如创建表和使用命令查看数据）也需要受到控制。在 Snowflake 中，我们可以

通过使用基于角色的访问控制（RBAC）来实现这一点。第 5 章专门讨论 RBAC，因为限制数据访问很重要。

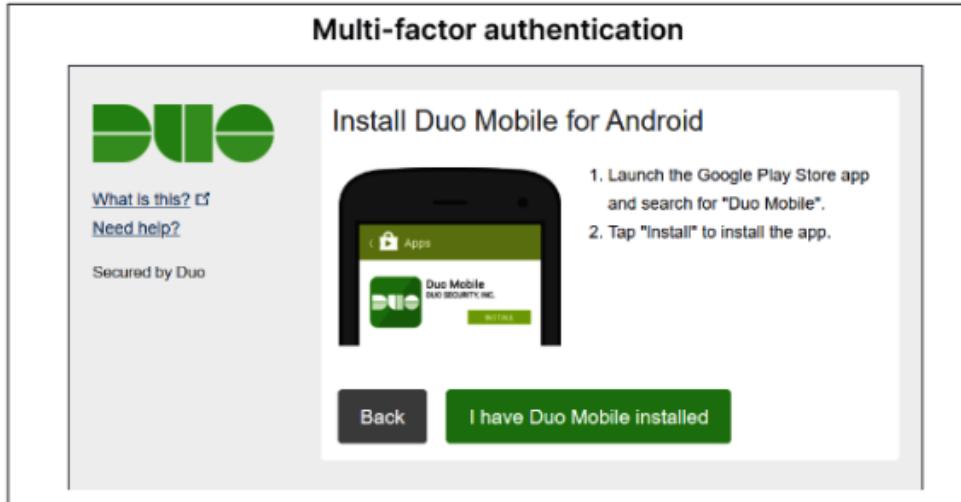


图 7-3. 用于多重身份验证的 Duo Mobile 设置屏幕

在第 5 章中，我们讨论了最佳实践。我们建议的最佳实践的一个示例是将自定义角色重新授予系统管理员角色。但是，对于用于访问敏感数据（如 HR、财务或医疗数据）的角色，您可能需要考虑该做法的一些例外情况。在本章的前面部分，我们创建了三个新的自定义角色，并将其中两个角色分配回给系统管理员。但是，您会注意到我们没有向管理员授予 HR 角色。这不是疏忽，而是故意的，因为可能存在敏感的 HR 数据，即使是 Snowflake 管理员也不应被允许访问。提醒一下，您可以使用第 5 章中介绍的角色层次结构可视化资源管理器来查看角色关系。

管理网络安全策略和防火墙访问

默认情况下，用户可以从任何位置连接到 Snowflake。但是，为了降低安全风险或满足合规性要求，您可能需要限制用户连接到 Snowflake 的方式和位置。这可以通过创建网络策略 `icy` 对象来实现。例如，可以创建网络策略来允许来自某些 IP 地址或组织的虚拟专用网络（VPN）和虚拟私有云（VPC）的流量。您可以指定要允许或禁止的 IP 范围列表。



任何定义的阻止 IP 范围都应该是允许的 IP 地址列表的子集，因为无论如何，允许的 IP 范围列表之外的任何内容都将被自动禁止。

网络策略有两种类型：账户级网络策略和用户级网络策略。可以通过 Web 界面或使用 SQL 创建账户级网络安全策略。相比之下，用户级网络策略只能使用 SQL 形成。

您需要使用账户管理员角色来设置网络策略。作为账户管理员，您可以单击主菜单中的 Admin → Security。确保您的角色设置为 ACCOUNTADMIN。在右上角，单击 + Network Policy 按钮。这将提供一个屏幕，如图 7-4 所示，您可以在其中创建新的网络策略。

New network policy

Creating as ACCOUNTADMIN

Enter a valid IPv4 address and optional CIDR or multiple addresses in a comma-separated list.

Policy Name

Allowed IP Addresses

Blocked IP Addresses (optional)

图 7-4. 网络策略创建屏幕

创建账户级网络策略后，需要通过将策略关联到您的 Snowflake 账户来激活该策略。屏幕右侧是 Activate Policy（激活策略）按钮。

SECURITYADMIN 或 ACCOUNTADMIN 角色可以创建账户级别的网络工作策略，但一次只能激活一个网络策略。同样，在 Snowflake 强制实施该策略之前，还需要激活用户级网络策略，并且每个用户一次只能激活一个用户级策略。对于同时具有账户级和用户级网络策略的任何用户，用户级策略将优先。

您可以查看 [Snowflake 网络策略](#) 中所有网络策略的列表。确保使用 ACCOUNTADMIN 角色来执行该语句。执行该命令不需要正在运行的仓库。

对于防火墙出口流量，如果您的网络具有用于出口流量的防火墙，则可以允许组织的防火墙将客户端应用程序连接到 Snowflake。如果允许公共端点，则可以执行 `SELECT SYSTEM$WHITELIST()` 函数。如果您有可访问的私有终端节点，则可以执行 `$WHITELIST_PRIVATERLINK()` 函数。请注意，如果您使用网络代理来检查出口流量，则可以将其设置为 SSL 直通。



Snowflake 不支持 SSL 终止代理。

可以使用 Snowflake 连接诊断工具 SnowCD 完成网络连接问题的诊断和故障排除。SnowCD 利用前面的系统功能列出的 Snowflake 主机名 IP 地址和端口来运行一系列连接检查，以评估和帮助排查与 Snowflake 的网络连接问题。有关如何下载和安装 SnowCD 的说明，请查看 [Snowflake 用户指南](#) 文档。

使用 Snowflake ACCESS_HISTORY 账户使用情况视图监控活动

设计和实施访问控制以限制对某些数据的访问是账户安全的关键部分。此外，还存在可用于监控访问的 Snowflake 工具。Snowflake ACCESS_HISTORY 视图监控访问，使公司能够满足其合规性审计并遵守法规要求。在出于安全原因执行监控时，建议使用 ACCESS_HISTORY 视图来识别任何未使用的表或列，这些表或列可以删除以优化存储成本。

Snowflake ACCESS_HISTORY 视图用于查询 Snowflake 对象在过去 365 天内的访问历史记录。Snowflake 可以进入

ACCOUNTADMIN 角色。ACCESS_HISTORY 视图支持 SQL 读写操作。此视图中的列包括 QUERY_ID、运行查询的用户的名称、查询的开始时间以及用户访问或修改的对象。请注意，访问的对象的输出以 JSON 格式给出，如图 7-5 所示。因此，我们需要将输出展开。我们首先在第 4 章中了解了该函数。

DIRECT_OBJECTS_ACCESSED	BASE_OBJECTS_ACCESSED
[{"columns": [{"columnId": 9225, "columnName": "DEPT_ID"}], [{"columns": [{"columnId": 10242, "columnName": "RATING"}]}]}	[{"columns": [{"columnId": 10242, "columnName": "RATING"}]}]
[{"columns": [{"columnId": 9225, "columnName": "DEPT_ID"}], [{"columns": [{"columnId": 10242, "columnName": "RATING"}]}]}	[{"columns": [{"columnId": 10242, "columnName": "RATING"}]}]
[{"columns": [{"columnId": 9225, "columnName": "DEPT_ID"}], [{"columns": [{"columnId": 10242, "columnName": "RATING"}]}]}	[{"columns": [{"columnId": 10242, "columnName": "RATING"}]}]
[{"columns": [{"columnId": 9225, "columnName": "DEPT_ID"}], [{"columns": [{"columnId": 10242, "columnName": "RATING"}]}]}	[{"columns": [{"columnId": 10242, "columnName": "RATING"}]}]
[{"columns": [{"columnId": 10242, "columnName": "RATING"}], [{"columns": [{"columnId": 10242, "columnName": "RATING"}]}]}	[{"columns": [{"columnId": 10242, "columnName": "RATING"}]}]
[{"columns": [{"columnId": 10242, "columnName": "RATING"}], [{"columns": [{"columnId": 10242, "columnName": "RATING"}]}]}	[{"columns": [{"columnId": 10242, "columnName": "RATING"}]}]
[{"columns": [{"columnId": 10258, "columnName": "DEPT"}], [{"columns": [{"columnId": 10258, "columnName": "DEPT"}]}]}	[{"columns": [{"columnId": 10258, "columnName": "DEPT"}]}]
[{"columns": [{"columnId": 10258, "columnName": "DEPT"}], [{"columns": [{"columnId": 10258, "columnName": "DEPT"}]}]}	[{"columns": [{"columnId": 10258, "columnName": "DEPT"}]}]
[{"columns": [{"columnId": 10259, "columnName": "ROLE_NAME"}], [{"columns": [{"columnId": 10259, "columnName": "ROLE_NAME"}]}]}	[{"columns": [{"columnId": 10259, "columnName": "ROLE_NAME"}]}]
[{"columns": [{"columnId": 10261, "columnName": "ROLE_NAME"}], [{"columns": [{"columnId": 10261, "columnName": "ROLE_NAME"}]}]}	[{"columns": [{"columnId": 10261, "columnName": "ROLE_NAME"}]}]

图 7-5. 访问对象的输出为 JSON 格式

Snowflake ACCESS_HISTORY 视图支持大多数写入操作命令，包括`CREATE`、`ALTER` 和`DELETE`。填充视图或流的操作以及复制导致的数据移动不包括在 ACCESS_HISTORY 视图中。

让我们考虑一下 ACCESS_HISTORY 视图可以为我们提供哪些具体信息。我们可以运行查询来查看每个用户运行的查询数量。我们将首先列出查询次数最频繁的用户：

```
使用角色 ACCOUNTADMIN;选择 USER_NAME, COUNT(*) USES FROM SNOWFLAKE.ACCOUNT_USAGE。
ACCESS_HISTORY GROUP BY USER_NAME ORDER BY 使用 DESC;
```

我们还可以查询 Snowflake ACCESS_HISTORY 视图，以找出最常用的表：

```
选择 OBJ.VALUE: objectName: : STRING TABLENAME, COUNT(*) 使用 FROM SNOWFLAKE。
ACCOUNT_USAGE.ACCESS_HISTORY, TABLE (FLATTEN (BASE_OBJECTS_ACCESSED) ) OBJ 按 TABLENAME
排序 BY 使用 DESC.
```

虽然这两条信息都很有趣，但如果有一些额外的上下文，这些信息并不是特别有用。我们真正想知道的是哪些表正在被访问、由谁访问以及访问频率如何。下一个语句将为我们提供该信息：

```
选择 OBJ.VALUE: objectName: : string TABLENAME, USER_NAME, COUNT(*) 使用 FROM
SNOWFLAKE.ACCOUNT_USAGE.ACCESS_HISTORY, TABLE (FLATTEN (BASE_OBJECTS_ACCESSED) )
排序 BY 1, 2 排序 BY 使用 DESC.
```

`SNOWFLAKE ACCOUNT_USAGE` 架构包括其他可用于监控的视图。`LOGIN_HISTORY` 视图是使用 Snowflake 建立的每个连接的日志，以便您可以确定谁从何处登录以及使用何种身份验证方法。`QUERY_HISTORY` 视图提供在 Snowflake 中运行的每个查询的日志，包括针对元数据（如用户或角色）的查询以及针对数据的查询。

数据保护和恢复

Snowflake 提供了多种原生数据保护和高可用性功能。其中许多功能（例如加密和密钥管理）都是自动的，无需您参与即可完全运行。其他数据保护和恢复功能（例如时间旅行和故障安全）会自动存在于 Snowflake 中，您可以在必要时随时利用它们。最后，还有一些 Snowflake 数据保护和恢复本机功能，例如复制和故障转移，需要您进行一些规划和实施才能利用它们。也就是说，让我们更深入地了解刚才提到的三类数据保护和恢复功能中的每类。

加密和密钥管理

Snowflake 会自动为所有 Snowflake 版本提供端到端加密（E2EE），无论版本类型如何，因此，即使是部署了 Snowflake 帐户的云提供商也无法访问您的数据。当您存储 Snowflake 表时，将使用云提供商的存储服务创建和存储一个或多个文件。

Snowflake 使用不同的密钥加密每个文件，然后通过使用更高级别的表主密钥加密每个文件和文件密钥，将它们“包装”在一起。这样做的一个原因是避免在安全位置存储许多单独的文件密钥。用于加密和解密文件密钥的表主密钥本身使用更高级别的账户主密钥进行加密，并存储在表的元数据中。同样，用于加密和解密表主密钥的账户主密钥也使用根密钥进行加密。根密钥是以明文形式存储的唯一密钥，用于加密和解密账户主密钥。

总而言之，Snowflake 的分层密钥模型由根密钥、账户主密钥、表主密钥和文件密钥组成。当账户键或表键的存在时间超过 30 天时，Snowflake 会轮换该账户键或表键。请注意，结果主键和阶段主键也存在于表主键级别。

Snowflake 的根密钥位于硬件安全模块（HSM）中。Snowflake 为需要对密钥进行更多控制并拥有 Snowflake 业务关键版或更高版本的组织提供 Tri-Secret Secure 功能。此功能允许您使用云提供商的密钥管理服务生成自己的根密钥，以便使用两个账户主密钥，而不仅仅是一个。每个主密钥都由

根键，您的根密钥包装一个账户主密钥，而 Snowflake 的根密钥包装另一个账户主密钥。如图 7-6 所示，解密 table、result 或 stage master key 需要两个 master keys。

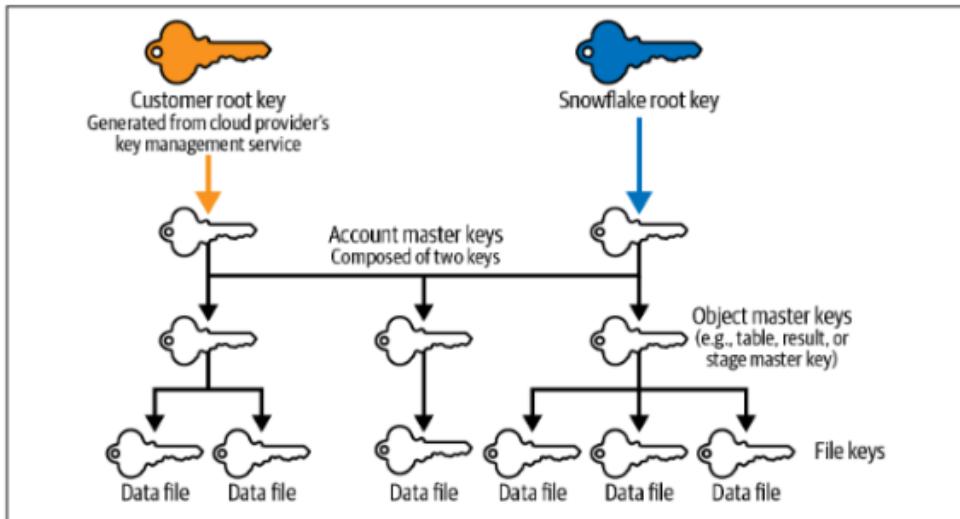


图 7-6. 密钥的 Tri-Secret Secure 层次结构

使用业务关键版本及更高版本，Snowflake 可以加密 VPC 内通过网络传输的所有数据。Enterprise for Sensitive Data (ESD) 账户支持查询语句加密。

Time Travel 和 fail-safe 保留期（如下一节所述）不受重新生成密钥的影响。重新生成密钥对这两个功能都是透明的。但是，一些额外的存储费用与故障安全中的数据重新生成密钥有关。

时间旅行和故障安全

Snowflake 提供本机变更数据捕获 (CDC) 功能，以确保在特定时间段内，已更改或删除的重要数据持续可用。永久数据库和数据库对象中的历史数据支持查询、克隆和恢复，最长可达 90 天。临时和临时对象中的数据最多可以访问 24 小时。在这些时间旅行访问期之后，Snowflake 员工最多可以在 7 天后恢复永久数据库和数据库对象中的历史数据。这 7 天的可恢复性称为故障安全期。

使用 Time Travel 功能，Snowflake 管理员可以查询过去更新或删除的数据，并且可以在过去特定时间点或之前创建整个表、数据库和数据库的克隆。拥有 Time Travel 优惠

防止意外的数据操作，例如错误地删除列或表。

时间旅行可用于永久、临时或临时表，但不适用于外部表。以下命令可用于时间旅行：SELECT At/Before、CLONE At/Before 和 。通过 Time Travel，用户可以通过以下三种方式之一使用 command 查询对象：

- 可以使用时间戳来完成查询，以查看对象在给定时间点之前的外观。
- 可以使用时间偏移来完成查询，以便根据经过的一定时间量查看对象之前的外观。
- 可以使用时间旅行在运行特定查询之前查看对象的外观。要获取任何查询 ID，您可以单击 活动 → 查询他的菜单选项，然后您将能够看到所有查询的列表，包括查询 ID 和状态，如图 7-7 所示。

The screenshot shows the 'Query History' section of the Snowflake interface. On the left, there is a sidebar with user information (Joyce Avila, ACCOUNTADMIN) and navigation links: Worksheets, Dashboards, Data, Marketplace, Activity (which is selected and highlighted in blue), Copy History, Admin, and Help & Support. The main area is titled 'Query History' and displays '250+ Queries'. It has three columns: SQL TEXT, QUERY ID, and STATUS. Below is a table with sample query logs:

SQL TEXT	QUERY ID	STATUS
SELECT OBJ.VALUE:objectName::string TABLEN...	01a4a566-0000-9c25-00...	Success
SELECT OBJ.VALUE:objectName::STRING TABLEN...	01a4a566-0000-9c25-00...	Success
SELECT USER_NAME, COUNT(*) USES FROM SNOWFL...	01a4a566-0000-9bb7-00...	Success
USE ROLE ACCOUNTADMIN;	01a4a566-0000-9c25-00...	Success
SELECT * FROM SNOWFLAKE.ACCOUNT_USAGE.ACCE...	01a4a561-0000-9bb7-00...	Success
with active_contracts as (select contract...	01a4a55e-0000-9c25-00...	Success
USE ROLE SYSADMIN;	01a4a55a-0000-9c25-00...	Success
GRANT INSERT ON FUTURE TABLES IN SCHEMA DE...	01a4a55a-0000-9bb7-00...	Success

图 7-7. 从 Query History 子菜单中看到的 Query history

自动为所有 Snowflake 账户启用按时间旅行保留期，默认值为 24 小时或 1 天；但是，可以在 Account 和 Object 级别将其设置为零。对于 Snowflake Enterprise Edition 和更高级别的组织，永久数据库、架构和表的保留期最多可以设置为 90 天。让我们继续将数据库的保留期设置为 90 天。导航回工作表并执行以下命令：

```
使用角色 SYSADMIN;ALTER DATABASE DEM07_DB 设置  
DATA_RETENTION_TIME_IN_DAYS = 90;
```

通过将数据库的保留时间设置为 90 天，所有数据库对象也将具有 90 天的保留期限。我们可以通过查看数据库的 INFORMATION_SCHEMA TABLES 视图来确认这一点：

使用角色 SYSADMIN:选择 TABLE_CATALOG、TABLE_SCHEMA、TABLE_NAME、RETENTION_TIME DEM07_DB。INFORMATION_SCHEMA。表：

然后，您将能够看到 RATINGS 表具有 90 天的保留时间，如图 7-8 所示。

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	RETENTION_TIME
1	DEMO7_DB	CH7DATA	RATINGS	90

图 7-8. 显示 RATINGS 表的 90 天保留时间的信息架构的详细信息

虽然无法为帐户禁用时间旅行，但您可以通过将默认数据保留时间从 1 天更改为 0 天，有效地禁用特定数据库、架构或表的时间旅行。有关已删除对象的信息仍可在 Snowflake 账户使用情况中看到，但在信息架构中看不到。

如果您更改保留期，则需要注意一些细微差别。在架构级别更改数据保留时间将导致架构中的所有表都继承架构的保留期，除非明确为表指定了不同的保留期。您应该记住的另一件事是，如果存在差异，您删除对象的顺序确实会影响保留期。删除架构时，所有现有表都将在与架构相同的时间段内可用。如果要确保子对象的数据保留期为 hon- or，则需要在删除父对象之前删除子对象。

现在让我们看一些示例，了解 Time Travel 的工作原理。假设我们犯了一个错误并更新了列中的所有值，而不仅仅是更新一个值。这可能类似于以下语句：

使用角色 SYSADMIN:更新DEMO7_DB。CH7DATA。评分区域 SET
AREA=4;

当我们查看上一个语句之后的表格时，我们发现我们确实犯了一个错误。使用该语句，我们看到所有 area 值都已更改为值 4，而不仅仅是一个 employee 的值：

SELECT * FROM DEM07_DB。CH7DATA。评级；

我们有几种方法可以将 AREA 列恢复为以前的值。我们可以使用基于时间值的两种方法中的一种，要么使用过去的特定时间，要么从现在开始倒退一定时间，或者我们可以使用

查询 ID。由于我们立即意识到了我们的错误，因此这三者中的任何一个都很容易实现。对于我们的示例，我们将继续将时间倒流 5 分钟。我们首先确认一下，回到过去 5 分钟将产生我们想要的结果。

```
SELECT * FROM DEMO7_DB.CH7DATA.RATINGS at (offset => -60*5);
```

如果表格看起来正确，请继续执行下一个语句。如果没有，请调整分钟数，直到找到所需的正确时间，然后继续执行下一个语句：

```
CREATE OR REPLACE TABLE DEMO7_DB.CH7DATA.RATING AS SELECT * FROM DEMO7_DB.CH7DATA.RATINGS at (offset => -60*5);
```

如果我们现在检查，我们将看到该表具有我们进行更新之前的先前值：

```
SELECT * FROM DEMO7_DB.CH7DATA.RATING;
```

我们本可以通过使用历史记录中的查询 ID 来完成相同的操作，而不是使用时间偏移量回到过去。该语句将如下所示：

```
SELECT * from DEMO7_DB.CH7DATA.RATINGS before (statement =>
    '<来自历史记录的查询 ID>');
```

可能发生的其他情况是，我们可能会错误地删除了一个表，然后在意识到我们的错误后，我们会想要回到表被删除之前。让我们看看它会是什么样子。我们首先删除表格：

```
DROP TABLE DEMO7_DB.CH7DATA.RATING;
```

尝试运行命令，您将看到该表不再存在。现在取消删除表并再次尝试运行该命令：

```
CANCEL STATEMENT DEMO7_DB.CH7DATA.RATING;
```

请务必记住，时间旅行适用于永久、暂时和临时对象，例如数据库和数据库对象。它不适用于用户或仓库。例如，如果您删除了一个用户，则必须重新创建该用户。

虽然时间旅行允许 Snowflake 用户在一段时间内访问数据，但只有 Snowflake 员工才能从故障安全存储中恢复数据。故障安全存储是一种尽力而为的紧急措施，它可能能够保护由于灾难性事件（如安全漏洞或系统故障）而导致的永久表中的数据丢失。

接下来，我们将总结时间旅行和故障安全如何协同工作于永久表，以及这与时间旅行如何工作于临时表和临时表。在图 7-9 中，我们可以可视化永久对象的 Time Travel 和 fail-safe 时间线。

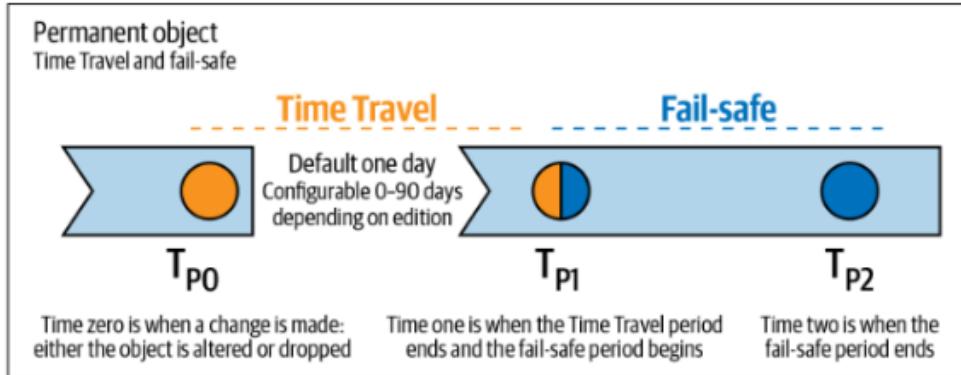


图 7-9。永久对象 Time Travel 和 fail-safe 时间轴

可以增加永久对象的 Time Travel 保留期。完成后，当前 Time Travel 保留期内存在的数据现在将保留更长时间，然后再移动到故障安全。但是，之前已超过当前保留期且已进入故障安全 7 天期的数据将不受影响。数据无法从 Failsafe 期间移回 Time Travel 期间。

当永久对象的 Time Travel 保留期缩短时，故障安全中的数据不会受到影响。Time Travel 中的数据将受到影响。任何时间超过新保留期的 Time Travel 数据都将被移至故障安全状态，并在那里存储 7 天。

Snowflake 瞬态对象类似于永久对象，不同之处在于瞬态对象没有故障安全期。此外，瞬态对象的最长默认时间旅行周期仅为一天，远小于永久对象可能的 90 天时间旅行周期。有关时间轴的可视化效果，请参见图 7-10。

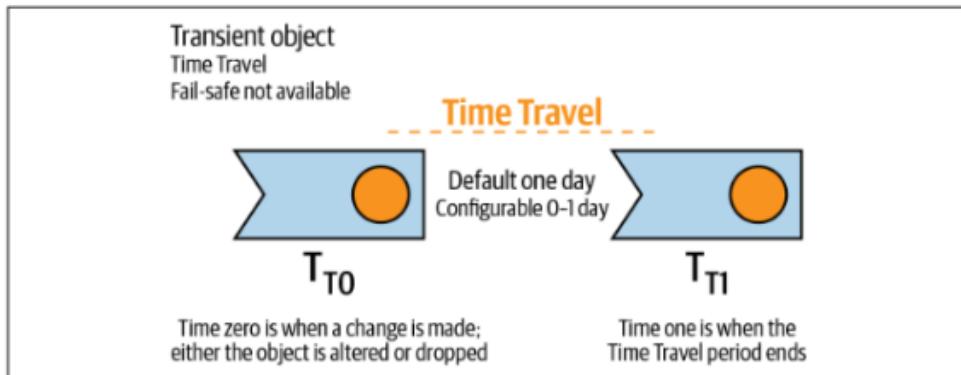


图 7-10. 瞬态对象 Time Travel 时间线

临时对象与瞬态对象具有相同的时间旅行默认值和最大值。就像 transient objects 一样，临时 objects 没有 fail-safe。临时对象和临时对象之间的区别在于，当会话结束时，临时对象中的数据无法再通过时间旅行功能访问。有关临时 Snowflake 对象的时间线可视化，请参见图 7-11。

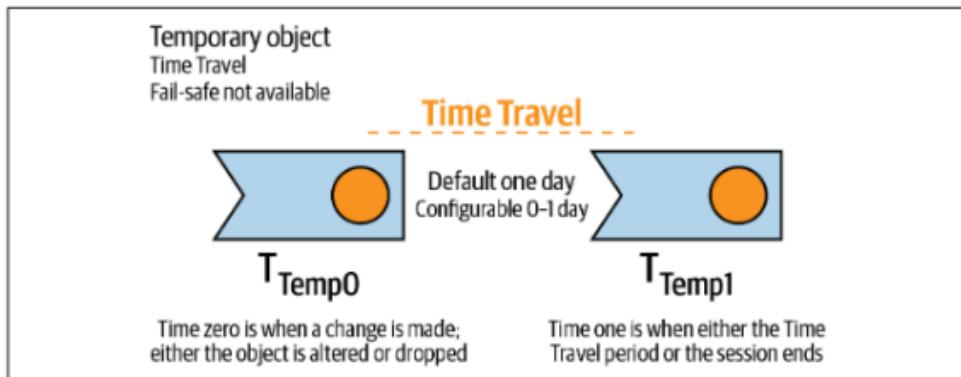


图 7-11. 临时对象 Time Travel 时间线

Time Travel 和 fail-safe 以及 active storage 都是产生存储成本的总计算存储的一部分。您可以使用以下语句查看三个部分中每个部分每个表的存储量（以字节为单位）：

使用角色 ACCOUNTADMIN：从 SNOWFLAKE 中选择 TABLE_NAME, ACTIVE_BYTES, TIME_TRAVEL_BYTES, FAILSAFE_BYTES。ACCOUNT_USAGE。TABLE_STORAGE_METRICS；

复制和故障转移

医疗保健、金融和零售企业只是不可接受停机的几种类型的组织。然而，这些组织在维护 24/7 系统和数据可用性方面面临着各种挑战，包括自然灾害、网络问题和病毒。这就是 Snowflake 复制和故障转移可以提供帮助的地方。

Snowflake 复制支持跨区域和跨云平台，使组织能够在 Snowflake 账户之间复制数据库。因此，复制和故障转移几乎可以立即同步 Snowflake 数据，并且由于跨云和跨区域复制是异步的，因此对主数据库的性能影响很小或没有影响。



Snowflake 使用随机唯一密钥加密要复制的文件。

请务必注意，克隆的 Snowflake 对象是物理复制的，而不是逻辑复制到辅助数据库的。这意味着任何克隆的对象都将产生额外的数据存储成本。复制费用还包括数据传输成本和计算资源成本。复制支持增量刷新，客户可以确定复制的运行频率。

请务必了解，这些费用将在目标账户上计费。

永久和临时主数据库都可以复制。当复制这些主数据库类型之一时，以下项目也会复制到辅助数据库：集群表的自动集群、视图（包括材料化视图）、存储过程、掩码和行访问策略、标签、用户定义函数（UDF）、文件格式和序列。

数据库支持 Snowflake 复制，但如果主数据库中存在外部表，则无法使用外部表创建或刷新辅助数据库。此外，无法复制以下单个对象：仓库、资源监视器、角色、用户、管道和流。请注意，可以使用账户级复制和创建故障转移组的功能。



在将数据库复制到其他地理区域或国家/地区之前，建议您研究一下您的组织是否对数据的托管位置有任何法律或法规限制。

通过数据治理控制实现数据大众化

数据治理的一个目标是建立有效的控制措施，通过识别敏感数据并保护敏感数据来更好地管理风险，同时持续监控控制措施的有效性。政府法规和监管合规性要求的提高使得关注数据治理比以往任何时候都更加重要。正如您所料，数据管理会带来相关成本。虽然人们很容易将数据治理视为有成本的东西，但不会提供任何真正的积极好处，但事实并非如此。事实上，设计良好的数据治理控制措施提供了支持数据民主化的必要框架，这是组织可以实现的最具竞争优势之一。

保护敏感数据通常是通过简单地禁止访问敏感数据所属的所有数据或通过在没有敏感数据的情况下创建单独的数据存储来实现的。这造成了数据孤岛，这些孤岛通常由陈旧数据组成，并且实施和维护成本很高。在许多情况下，将非敏感数据提供给组织中的其他人太耗时或成本太高。因此，组织组创建自己的非 IT 部门并不罕见。

批准的数据解决方案。不仅存在基于陈旧、不完整数据的糟糕决策，而且还增加了组织通常没有意识到的风险。人们如此消极地看待数据治理也就不足为奇了。Snowflake 提供了帮助解决这些问题的工具。

随着 Snowflake 的改进工具和方法的推出，复制数据以实现数据管理目标既不是必需的，也不推荐。借助 Snowflake 的许多治理相关功能，可以快速轻松地保护数据，从而允许对实时数据的开放访问。通过提供对最新完整数据的访问，Snowflake 使数据民主化在组织内存在并蓬勃发展成为可能。由于有效的数据治理，数据民主化可以增加组织内所有知识工作者对数据的访问。如果数据中存在敏感数据，我们可以简单地识别它并限制对实际数据的访问，而不是对元数据的访问。

数据驱动型组织中的所有利益相关者都能够实时做出有洞察力的数据驱动型决策，其切实的好处怎么强调都不为过。这一切都始于将数据治理作为数据民主化的框架，以及数据安全、确保隐私以及遵守法规和合规性要求的必要工具。

之前，我们讨论了 Snowflake ACCESS_HISTORY，这是主要的治理功能之一。与 Snowflake ACCESS_HISTORY 一样，本节中描述的管理功能在 Enterprise Edition 和更高级别的 Snowflake 组织中可用。在本节中，我们将介绍更多治理功能，包括 INFORMATION_SCHEMA Data Dictionary、对象标记、数据掩码、行访问策略、行级安全性、外部标记化和安全视图。

INFORMATION_SCHEMA 数据字典

INFORMATION_SCHEMA 也称为 Snowflake 数据字典，包括使用指标和元数据信息。INFORMATION_SCHEMA 中的信息包括账户和数据库级别的视图。第 3 章详细介绍了 INFORMATION_SCHEMA。

对象标记

在 Snowflake 中保护数据首先要了解对象中存储的数据类型并注释该信息，以便采取必要的步骤来保护数据。为了帮助完成注释过程，Snowflake pro 提供了一个原生功能，可用于创建自定义标签库。然后，可以将这些标签与 Snowflake 对象相关联。

Snowflake 标签是架构级对象，可以与不同的对象类型相关联，包括列以及架构、数据库和账户级对象。

标签存储为键值对，其中标签是键，每个标签可以有许多不同的字符串值。标记标识符在每个架构中必须是唯一的。

Snowflake 将每个账户的标签数量限制为 10000 个。



如果将标识符括在双引号中，则区分大小写，并且可以包含空格。如果标识符未加引号，则它不能区分大小写，不能包含空格，并且必须以字母或下划线开头。

为了演示 Snowflake 对象标记的工作原理，我们将首先创建我们打算使用的标签的分类目录：

```
使用角色 SYSADMIN;  
创建或替换 SCHEMA DEMO7_DB。TAG_LIBRARY;
```

我们要设置的第一个标签是在表级别关联的标签，用于描述表是否将被分类为机密、受限、内部或公共。请注意，我们确保包含注释以列出我们想要用于该标签的可能值：

```
CREATE OR REPLACE TAG 分类:ALTER TAG 分类集  
注释 =  
“使用以下分类值之一标记表或视图：‘机密’、‘受限’、‘内部’、‘公共’”;
```

接下来，我们将创建一个在列级别关联的标签，该标签将标识包含个人身份信息(PII)的字段：

```
创建或替换标签 PII:  
ALTER TAG PII set comment = “带有一个或多个 PII 的标记表或视图  
以下值：‘Phone’，‘Email’，‘Address’”;
```

最后，我们将创建一个在列级别关联的标签，该标签将标识包含敏感 PII 的字段，例如社会保险号、护照信息和医疗信息：

```
创建或替换标签 SENSITIVE_PII;ALTER TAG SENSITIVE_PII set comment = “使用敏感 PII 标记表  
或视图  
具有以下一个或多个值：‘SSN’、‘DL’、‘Passport’、  
‘财务’，‘医疗’”;
```

我们将继续使用管理员角色创建新的 EMPLOYEES 表。请记住，之前我们授予了 HR 角色在 HRDATA 架构中对新表使用语句的能力，以便该角色可以访问我们正在创建的新表：

```
使用角色 SYSADMIN 创建或替换表 DEM07_DB。HRDATA 的EMPLOYEES (emp_id 整数,
fname varchar(50), lname varchar(50), ssn varchar(50), 电子邮件 varchar
(50), dept_id 整数, 部门 varchar(50)) ;INSERT INTO DEM07_DB 中。HRDATA 的员工
VALUES (1, 'Fiona', 'Lance', '123-4567-8901', 'dept_1', 'First', 'Last', '000-0000',
'fiona@email.com', '100', 'IT') ;
```

创建新的 EMPLOYEES 表后，我们将使用 `com-mand` 分配标签。首先，我们将值为 的 classification 标记分配给整个表：

```
ALTER TABLE DEM07_DB。HRDATA 的员工
set tag DEM07_DB。TAG_LIBRARY。分类=“机密”；
```

接下来，我们将两个标签分配给两个单独的列：

```
ALTER TABLE DEM07_DB。HRDATA 的员工修改电子邮件
set tag DEM07_DB。TAG_LIBRARY.PII = “电子邮件”；ALTER TABLE DEM07_DB。HRDATA 的员工修改 SSN
设置 TAG DEM07_DB。TAG_LIBRARY.SENSITIVE_PII = “SSN”；
```



除了直接分配给对象的标签外，对象还可以具有基于 Snowflake 对象层次结构的继承标签。例如，应用于表的任何标记也会应用于该表中的 columns。

您可以使用该语句查看所有三个标签均已创建。

结果 7-12 如图 12 所示。

	created_on	name	database_name	schema_name	owner
1	.004 -0700	CLASSIFICATION	DEMO7_DB	TAG_LIBRARY	SYSADMIN
2	.596 -0700	PII	DEMO7_DB	TAG_LIBRARY	SYSADMIN
3	.686 -0700	SENSITIVE_PII	DEMO7_DB	TAG_LIBRARY	SYSADMIN

图 7-12. 声明结果

您还可以查询表或列，以查看特定标签值是否与表或列关联。首先，让我们看看是否有任何标签值与我们的 EMPLOYEES 表相关联：

```
SELECT SYSTEM$GET_TAG('分类', 'DEMO7_DB.HRDATA 的EMPLOYEES', 'table') ;
```

我们看到 tag 值是关联的。现在让我们看看是否有 SENSITIVE_PII tag 都与列相关联。你会注意到，我们必须指定要检查的列：

```
选择 SYSTEM$GET_TAG ('SENSITIVE_PII', 'DEMO7_DB.HRDATA 的员工。SSN'、  
'column');
```

正如预期的那样，tag 值与 SSN 列相关联。让我们看看如果我们尝试检查列中不存在的标签值会发生什么情况：

```
选择 SYSTEM$GET_TAG ('SENSITIVE_PII', 'DEMO7_DB.HRDATA 的员工。电子附件'、  
'column');
```

当与我们正在查询的特定标签关联的列中没有标签值时，我们将返回一个值。在我们的示例中，我们得到了该值，因为 EMAIL 列中没有标签。但是，EMAIL 列中有一个标签，您可以通过更改查询来确认该标签。

适当地为 table、view 和 columns 设置 tag，我们可以根据 tag 进行查询，发现哪些数据库对象和列包含敏感信息。然后，数据管理员可以根据需要实施数据屏蔽策略或行访问策略。

让我们创建一个 SQL 语句来查找已使用 or 标签标记但没有屏蔽策略的任何对象或列。让我们看看当我们运行以下语句时会发生什么：

```
在没有屏蔽策略的情况下审核所有带有敏感标签的列 //请注意，tag_references 使用  
角色 ACCOUNTADMIN 的延迟可能长达 2 小时;column_with_tag AS (SELECT
```

```
object_name table_name,
```

```
column_name column_name,  
object_database db_name, object_schema schema_name FROM  
snowflake.account_usage.tag_references 中，  
tag_schema='TAG_LIBRARY' AND (tag_name='SENSITIVE_PII' OR  
tag_name = 'PII') 且 column_name 不为空)，
```

```
column_with_policy  
AS (选择 ref_entity_name table_name、  
ref_column_name column_name、ref_database_name  
db_name ref_schema_name schema_name FROM  
snowflake.account_usage.policy_references where  
policy_kind='掩码策略' ) SELECT * FROM  
column_with_tag MINUS SELECT * FROM  
column_with_policy;
```

最有可能的是，查询已成功运行，但未返回任何结果。这是因为在您能够查询 Snowflake 数据库以获取标签引用之前，存在长达两个小时的延迟。因此，您需要继续阅读本章的其余部分，但请记住稍后返回并重新运行此语句以查看结果，如图 7-13 所示。

	TABLE_NAME	COLUMN_NAME	DB_NAME	...	SCHEMA_NAME
1	EMPLOYEES	EMAIL	DEMO7_DB		HRDATA
2	EMPLOYEES	SSN	DEMO7_DB		HRDATA

图 7-13. 结果显示哪些标记列没有掩码策略

Snowflake 标签的设计方式允许采用不同的管理方法。在集中式方法中，管理员负责创建标签并将其应用于 Snowflake 对象。管理员可以创建一个 TAG_ADMIN 角色来委派该责任的处理。或者，可以以分散的方式使用 TAG_ADMIN 角色，以便各个团队将标签应用于 Snowflake 对象。

在本章的后面部分，我们将讨论复制。现在，我们只说明在复制 Snowflake 主数据库时，标签也会复制到辅助数据库。

Snowflake 分类是一项新功能，可自动检测表、视图或列中的 PII 并对其进行标记。分配分类生成的标签后，它们可用于数据管理、数据共享和数据隐私目的。

在本节中，我们重点介绍了对象标记示例的安全问题。对象标记还经常用于跟踪数据和仓库消耗，以便进行成本报告。有关该对象标记示例，请参见第 8 章。

分类

Snowflake 的数据分类会分析结构化数据中的列，以查找可能被视为敏感信息的个人信息。分类功能内置于 Snowflake 平台上，包括预定义的系统标签，以帮助对数据进行分类。数据分类后，可通过查询 INFORMATION_SCHEMA 来发现数据。然后，可以通过 ACCESS_HISTORY 视图审核分类数据访问，以确保已实施适当的基于角色的策略和匿名化。

分类类别类型

有两种类别分类类型：语义和隐私。语义分类标识存储个人属性的列，例如姓名、地址、

年龄或性别。标识了语义类别的任何列都将进一步分类为隐私类别。



分类适用于所有 Enterprise Edition 帐户及更高版本。要对表或视图中的数据进行分类，用于对数据进行分类的角色必须具有对 SNOWFLAKE 数据库的 IMPORTED PRIVILEGES 特权。此外，还需要对数据库和架构的 USAGE 特权，以及至少对表或视图的 SELECT 特权。由于分类功能使用标签，因此还需要具有 APPLY TAG 全局权限。

Snowflake 分类支持以下三种隐私类别：

标识符

也称为直接标识符。直接标识符标识个人。直接标识符的示例包括姓名、社会安全号码和电话号码。

准标识符

也称为间接标识符。当与其他属性结合使用时，准标识符可用于唯一标识个人。使用准标识符可以唯一标识个人的一个示例是，如果一个人的年龄、性别和邮政编码可用。

敏感信息

包括不直接或间接识别的个人属性，但这些属性是私有的，因此不应披露。示例包括工资详情和医疗状况。



目前，Snowflake 中的数据只能归类为敏感标识符或直接/间接标识符，而不能同时归类为两者。了解这一点可能会影响您为受监管的访问计划所做的工作。

分析 Snowflake 表或视图后，将返回系统标记的值。要分析表或视图中的列，请使用 Snowflake 的 EXTRACT_SEMANTIC_CATEGORIES 函数生成语义和隐私分类。接下来，需要自动或手动标记具有语义或隐私类别的列。我们在上一节中了解了手动标记。要自动标记具有关联语义或隐私类别的 column，可以使用 ASSOCIATE_SEMANTIC_CATEGORY_TAGS 存储过程。

可以对所有表和视图类型执行分类，包括外部表、具体化视图和安全视图。Snowflake 支持对除`CHAR`、`VARCHAR` 和`BIGINT` 数据类型之外的所有数据类型的数据进行分类。

数据掩码

向所有用户透露某些数据可能会产生后果，尤其是在有法律和合规性要求来保持数据的私密性和安全性的情况下。限制内部对私有或敏感数据的访问可降低数据有意或无意泄露的风险。使用对象标记是识别需要保护的数据的必要第一步。然后，我们可以使用 Snowflake 的原生治理控制，例如数据掩码、列级安全性和行级策略。行级访问控制将在下一节中讨论。

Snowflake 掩码策略是应用于表或视图中各个列的架构级对象。因此，在将掩码策略应用于列之前，必须存在数据库和架构。此列级安全功能使用掩码 policy 选择性地使用部分掩码值或完全掩码值屏蔽纯文本数据。

动态数据掩码

使用动态数据掩码，掩码策略将在查询运行时应用于列。掩码策略可以包含角色层次结构和/或基于网络安全策略等内容。此外，动态数据掩码可以与数据共享一起使用，而外部标记化则不能。

在我们的第一个数据掩码示例中，我们将创建一个基于角色的掩码策略。我们希望允许`HR` 角色查看纯文本数据，而所有其他角色将看到完全掩码的值。请记住，我们需要使用账户管理员角色来创建掩码策略：

```
使用角色 ACCOUNTADMIN:CREATE OR REPLACE 掩码策略DEMO7_DB。HRDATA.电子邮件部件掩码
AS (val string) 返回字符串 ->
CASE 当 current_role() in ('HR_ROLE') THEN val
ELSE '**MASKED**' 结束;

ALTER TABLE DEMO7_DB。HRDATA 的EMPLOYEES modify 列 EMAIL
DEMO7_DB设置掩码策略。HRDATA.电子邮件掩码;
```

现在我们已经创建了电子邮件掩码策略，我们需要为社会安全号码创建第二个掩码策略：

```
CREATE OR REPLACE 捞码策略 DEM07_DB。HRDATA 的SSN 捞码
AS (val string) 返回字符串 ->
CASE 当 current_role () in ('HR_ROLE') THEN val
ELSE '**MASKED**' 结束;
```

```
ALTER TABLE DEM07_DB。HRDATA 的EMPLOYEES 修改列 SSN
DEM07_DB设置掩码策略。HRDATA 的SSN 捞码:
```

如果我们使用 ACCOUNTADMIN 角色查看数据，而该角色尚未被授予HR_ROLE，我们将看到掩码数据。这是有意为之的，因为 Snowflake 已删除有权访问所有数据的 Snowflake 超级用户的概念。这一独特功能大大降低了风险，因为它允许深思熟虑地考虑谁应该被授予访问不同敏感信息的权限：

使用角色 ACCOUNTADMIN;SELECT * FROM DEM07_DB。
HRDATA 的员工:

让我们确认一下 HR 角色实际上可以看到数据。在发出以下 SQL 语句之前，请务必以新用户 Adam 的身份登录。如果您未以 Adam 身份登录，则在尝试执行该命令时会收到一条错误消息。这是因为您登录的用户未被授予HR_ROLE，因此您无法使用该角色：

使用 ROLE HR_ROLE;使用 WAREHOUSE
COMPUTE_WH;SELECT * FROM DEM07_DB。HRDATA 的员
工。

现在，我们确信 HR 数据将被适当地屏蔽，让我们在表中插入一些数据。您仍应以用户 Adam 的身份登录：

```
INSERT INTO DEM07_DB 中。HRDATA 的员工 (emp_id、fname、lname、ssn、电子邮件、
dept_id、部门) 值 (1, 'Harry', 'Smith', '111-1111', 'harry@coemail.com', '100',
'IT'), (2, 'Marsha', 'Addison', '222-2222', 'marsha@coemail.com',
'100', 'IT'), (3, 'Javier', 'Sanchez', '333-3333', 'javier@coemail.com',
'101',
'营销'), (4, '艾丽西亚', '罗德里格斯', '444-44-4444',
'elicia@coemail.com', '200',
'财务'), (5, 'Marco', 'Anderson', '555-5555', 'marco@coemail.com', '300',
'HR'), (6, 'Barbara', 'Francis', '666-66-6666', 'barbara@coemail.com', '400', '执
行');
SELECT * FROM DEM07_DB。HRDATA 的员工;
```

因为 Adam 被授予了 HR 角色，所以他可以看到所有数据。现在，让我们回到 Snowflake 主账户，以便我们准备好继续我们的动手示例。



可以查看未屏蔽数据列的角色能够将未屏蔽的数据插入到可能不受数
据屏蔽保护的另一列中。

条件掩码

条件掩码指定两个参数，使用一列来确定是否应掩码另一列中的数据。第一个参数指示将屏蔽哪个列，而第二个和后续参数用作条件列。

在我们的示例中，我们希望 HR 角色始终可以看到员工 ID，但 IT 人员的员工 ID 应该是每个人都可用的唯一员工 ID。除 IT 部门以外的所有员工 ID 号都应向除 HR 以外的所有人屏蔽：

```
使用角色 ACCOUNTADMIN;CREATE OR REPLACE 掩码策略DEMO7_DB。
HRDATA.namemask 中文版
AS (EMP_ID整数, DEPT_ID整数) 返回整数 ->
当 current_role () = 'HR_ROLE' 时的情况
    则 EMP_ID WHEN DEPT_ID = '100' THEN
        EMP_ID ELSE '**MASKED**' 结束;
```

在 Snowflake 主账户和用户 Adam 中使用该命令，查看员工 ID 号的查看方式的差异。

静态遮罩

静态掩码是掩码 Snowflake 数据的另一种方法。我们使用 ETL 工具转换一组数据，以创建一个或多个始终屏蔽某些列的表，而不是动态创建掩码。然后，我们能够根据特定角色授予对每个表的访问权限，这些表基于同一组基础数据构建。静态掩码允许明确分离角色，但由于额外的表维护和需要创建多个不同的表，因此在时间和金钱方面的成本会增加。

行访问策略和行级安全性

数据掩码是基于列的安全性，而行访问策略提供动态的行级安全性。串联设计这些策略很重要，因为 column 不能同时具有掩码策略和行访问策略。这就是为什么我们想通过使用两个不同的架构和表来演示掩码策略和行访问策略来将示例分开的原因。

在创建对象时或创建对象后添加到表或视图中，行访问策略根据角色和/或地理区域等信息限制查询结果中返回的行。虽然行访问策略可以防止在查询中返回行，但它们不能阻止更新现有行或插入新行。

可以在不同的表和视图上设置一个策略，并且行访问策略的范围可以从简单到复杂。复杂行访问策略的一个示例是 map - ping 表。为了说明行策略的工作原理，我们将根据如何向两个不同的角色授予对相同数据的访问权限来创建一个行级安全检查。首先，我们将创建一个映射表：

```
使用角色 SYSADMIN;使用数据库DEMO7_DB;使用 SCHEMA CH7DATA;CREATE OR REPLACE TABLE
AreaFiltering (role_name文本, area_id整数);INSERT INTO AreaFiltering (role_name,
area_id) 值
('AREA1_ROLE', 1), ('AREA2_ROLE',
2);SELECT * FROM AreaFiltering;
```

接下来，我们将创建一个安全视图来保存 RATINGS 表中的数据，以便该角色可以访问允许其访问的特定角色：

```
使用角色 SYSADMIN;创建或替换安全视图V_RATINGS_SUMMARY AS
SELECT emp_id, rating, dept_id, area FROM RATINGS WHERE
area= (SELECT area_id FROM AreaFiltering WHERE
role_name=CURRENT_ROLE ()) ;
```

我们仍在使用 SYSADMIN 角色，因此我们不会在安全视图中看到数据。要确认，请务必运行以下语句：

```
SELECT * FROM v_ratings_summary;
```

现在，让我们让 AREA1_ROLE 能够在安全视图中查看数据。当 area 列中的值从 DEMO7_DB.CH7DATA。RATINGS 表等于 AREA1：

```
将 SELECT 应用于 ROLE AREA1_ROLE IN SCHEMA CH7DATA 中的所有表;
GRANT SELECT ON AreaFiltering TO ROLE AREA1_ROLE;将 SELECT
v_ratings_summary 授予角色 AREA1_ROLE;
```

作为新用户 Adam，让我们尝试使用 AREA1 角色查看数据：

```
使用 ROLE AREA1_ROLE;使用数据库
DEMO7_DB;使用 SCHEMA CH7DATA;SELECT *
FROM v_ratings_summary;
```

因为有三行数据与 AREA1 相关联，所以我们期望 Adam 能够使用 AREA1_ROLE 看到三行数据，而这正是我们得到的结果（参见图 7-14）。

	EMP_ID	RATING	DEPT_ID	...	AREA
1	1	77	100		1
2	2	80	100		1
3	3	72	101		1

图 7-14. 访问安全视图时 AREA1 角色的结果

现在，让我们切换回主 Snowflake 用户，并向 AREA2 角色授予权限：

```
使用角色 SYSADMIN;
将 SELECT 应用于 SCHEMA CH7DATA 中的所有表，以授予角色 AREA2_ROLE;
GRANT SELECT ON AreaFiltering TO 角色 AREA2_ROLE; 将 SELECT
v_ratings_summary 授予角色 AREA2_ROLE;
```

我们可以再次切换回 Adam 的用户帐户，并确认我们可以使用 AREA2 角色查看与 AREA2 关联的数据：

```
使用 ROLE AREA2_ROLE; 使用数据库
DEMO7_DB; 使用 SCHEMA CH7DATA; SELECT *
FROM v_ratings_summary;
```

虽然 Adam 可以使用这两个区域角色在安全视图中查看数据，但如果 Adam 可以同时看到他应该能够看到的所有数据，那就更好了。幸运的是，我们在 Snowflake 中有办法实现这一目标。我们将用允许多条件查询的新安全视图替换原来的安全视图。确保您位于 Snowflake 主用户账户中，并运行以下语句。

我们需要先删除 secure 视图：

```
使用角色 SYSADMIN;DROP VIEW
v_ratings_summary;
```

接下来，我们将创建一个使用 multicondition 查询的新安全视图：

```
使用角色 SYSADMIN; 使用数据库 DEMO7_DB; 使用
SCHEMA CH7DATA; 创建 SECURE VIEW
v_ratings_summary AS

SELECT emp_id, rating, dept_id, area FROM RATINGS
WHERE area IN (SELECT area_id FROM AreaFiltering WHERE
role_name IN

(SELECT value FROM TABLE (flatten (input =>
parse_json (CURRENT_AVAILABLE_ROLES ())))) ) ;
```

最后，我们将为角色提供必要的权限：

```
GRANT SELECT ON AreaFiltering TO ROLE AREA1_ROLE;将 SELECT  
v_ratings_summary 授予角色 AREA1_ROLE;GRANT SELECT ON  
AreaFiltering TO 角色 AREA2_ROLE;将 SELECT v_ratings_summary  
授予角色 AREA2_ROLE.
```

现在是有趣的部分！转到 Adam 的用户帐户，看看如果我们使用任一 AREA 角色会发生什么：

```
使用 ROLE AREA1_ROLE;使用数据库  
DEMO7_DB;使用 SCHEMA CH7DATA;SELECT *  
FROM v_ratings_summary;
```

使用任一 AREA 角色，Adam 将能够看到 AREA1 和 AREA2 查询结果的组合结果（如图 7-15 所示）。

	EMP_ID	RATING	DEPT_ID	...	AREA
1	1	77	100		1
2	2	80	100		1
3	3	72	101		1
4	4	94	200		2

图 7-15. AREA1 和 AREA2 查询结果的组合结果

让我们看看如果我们使用 Adam 的另一个角色来查询安全视图会发生什么：

```
使用 ROLE HR_ROLE;使用数据库DEMO7_DB;  
使用 SCHEMA CH7DATA;SELECT * FROM  
v_ratings_summary;
```

Adam 收到消息，指出该视图不存在，或者他未获得授权。这向我们证实，如果 Adam 使用有权访问视图的角色，则可以获得对结果的完全访问权限。相反，如果 Adam 使用的角色无权访问视图，则他将无法看到任何结果。

外部分词

使用外部函数的外部标记化功能在 Snowflake Standard Edition 及更高版本中可用。如果您想将令牌化提供商与 Snowflake 外部令牌化集成，则需要拥有 Snowflake Enterprise Edition 或更高版本。

安全视图和 UDF

如果数据库中存在敏感数据，最佳实践是使用安全视图和安全 UDF，而不是直接共享 Snowflake 表。在第 3 章中，我们使用安全的 SQL UDF 完成了一个市场篮分析示例。第 10 章详细介绍了安全的数据共享方法，我们将学习如何使用映射表与许多不同的账户共享基表中的数据，我们希望与特定账户共享表中的特定行。

我们之前已经介绍了所有 Snowflake 数据库中可用的 PUBLIC 架构。最佳实践是将公有 Schema 用于安全视图，将私有 Schema 用于基表。

对象依赖关系

Snowflake 对象依赖关系功能允许您识别 Snowflake 对象之间的依赖关系，并经常用于影响分析、数据完整性保证和合规性目的。

当现有对象必须代表其或至少一个其他对象引用某些元数据时，将建立对象依赖关系。依赖项可以由对象的名称和/或其 ID 值触发。对象依赖关系的一个示例是 Snowflake 外部阶段，其中 ID 用作对外部存储位置的引用。

数据工程师在更改 Snowflake 架构之前了解对象依赖关系非常重要。例如，在工程师重命名表或删除表之前，他们可以通知用户依赖于这些表的下游视图。通过这种方式，工程师可以在实际进行更改之前获得必要的反馈来评估修改的影响。

此外，数据分析师和数据科学家可以向上游查看视图的沿袭。现在能够使用对象依赖关系功能做到这一点，让他们确信视图由可信的对象源提供支持。最后，新的对象依赖关系功能对于需要能够将数据从给定对象跟踪到其原始数据源以满足合规性和法规要求的合规性官员和审计员来说非常重要。

对象依赖项在 Snowflake Account Usage 架构中显示为视图。Snowflake 帐户管理员可以查询 OBJECT_DEPENDENCIES 视图，输出可以包括对象名称路径、引用对象名称和域，以及引用的对象名称和域。

Snowflake 的对象依赖关系功能是 Snowflake 提供的众多全面的数据管理原生功能之一。这套功能可以轻松有效地了解和保护您的数据。这很重要，因为管理您的数据

对于确保您的数据安全、值得信赖并符合法规要求至关重要。

代码清理

我们可以通过删除第 7 章数据库和我们创建的新用户来快速轻松地清理我们的 Snowflake 组织：

```
使用角色 ACCOUNTADMIN;
DROP DATABASE DEMO7_DB;DROP
用户 ADAM;
```

总结

本章提供的信息可帮助您更有效地了解 Snowflake 数据并更好地保护它。我们了解了分类和对象标记如何帮助我们了解 Snowflake 帐户中存在哪些数据、数据的存储位置以及不同角色如何访问这些数据。我们还学习了如何管理网络工作安全策略和防火墙访问，以及如何使用 Snowflake 的 ACCOUNT_USAGE 视图监控用户活动。此外，还介绍了许多保护数据的方法：时间旅行、故障保护、复制、故障转移/故障恢复、数据掩码、行访问策略、外部标记化以及安全视图和 UDF。

更好地了解 Snowflake 中拥有哪些数据，并更有效地保护这些数据，不仅可以降低风险，还可以减少在需要时查找数据所需的时间。降低风险、节省时间和提高服务级别最终会带来更低的总体成本和更好的性能。第 8 章将揭示更多帮助管理成本的技术。

更具体地说，在该章中，您将了解帮助您降低 Snowflake 账户成本的工具和技术。本章中讨论的一些新工具包括资源监控器，用于帮助控制虚拟仓库成本并避免意外的信用使用，以及用于监控 Snowflake 使用情况和成本的商业智能仪表板。此外，您将发现，本章中出于安全和保护原因使用的 ACCOUNT_USAGE 视图和对象标记也是可用于帮助管理成本的工具。

知识检查

以下问题基于本章中包含的信息：

1. 何时应使用多重身份验证，用户如何注册？
2. 可以存在的网络策略数量是否有限制？
3. 根加密密钥有哪些独特功能？

4. 描述不同类型的数据屏蔽以及数据屏蔽策略和行访问策略之间的主要区别。使用这两个策略时，最重要的限制是什么？
 5. 可以增加或减少 Time Travel 保留期吗？如果是这样，结果会发什么什么？
 6. 对象标签需要用引号、双引号还是不用引号括起来？解释。Snowflake 现在提供哪两项自动数据标记功能？
 7. 为什么我们需要扁平化查询输出？在本章中描述我们如何使用 FLATTEN 命令。
-
8. 在数据库复制中如何处理克隆的对象？
 9. 我们可以通过哪些不同的方式使用时间旅行来访问历史数据？
 10. 时间旅行需要考虑哪些相关费用？对于重复和故障转移呢？

这些问题的答案可在附录 A 中找到。

管理 Snowflake 账户成本

组织使用最新和最准确的信息创建业务计划和预算。了解本地虚拟仓库的预算金额相对容易，因为在大多数情况下，成本是固定的。基于使用量的成本方法（例如 Snowflake 当前的定价模型）最初可能看起来令人生畏。事实上，要了解任何基于使用量的成本模型，都需要提前清楚地了解成本是如何计算的，以及可以添加哪些护栏来控制以获得更大的弹性，这样您就不会遇到成本失控的情况。Snowflake 包括资源监视器等工具，这些工具允许您根据初始计划限制实际成本。超越对基于使用量的定价的基本理解，并使您的组织能够利用它，可以产生许多积极的结果。

基于使用量的定价的最大好处之一是，它为您提供了一个机会，可以透明地查看和了解您的组织如何为存储和使用数据付费。这通常会导致变化，这些变化将立即开始降低成本。Snowflake 基于使用量的计费和 Snowflake 对象标记使您可以轻松地将适当的成本分配给适当的账户或部门。这意味着每个团队和工作负载都有更多的成本责任。随着细粒度层面的问责制增加，人们可能会更多地考虑资源的使用方式，因此，将采取更多行动来提高效率。

Snowflake 的定价模型简单明了、透明且易于理解。除了任何基于云技术使用的成本方法可能带来的许多好处外，Snowflake 还具有独特的功能，可以降低成本。一个例子是零拷贝克隆，它提供了在不实际创建物理拷贝的情况下复制对象的能力。这意味着不会为克隆评估存储成本。此外，Snowflake 的原生功能允许您设置积分使用成本的限制。这些 Snowflake 资源监视器可以

这些 Snowflake 资源监控器可以设置为执行通知或暂停消耗等操作。此外，Snowflake 为公司提供了大量使用数据来了解他们的成本，这使得 Snowflake 上的客户更具成本效益。

在利用 Snowflake 的基于使用量的成本方法功能时，应从一开始就考虑软件交付操作。幸运的是，Snowflake 有几种方法可以缓解数据库变更管理的独特挑战并降低软件开发成本。

准备工作

创建标题为 Chapter8 Managing Costs 的新工作表。如果您在创建新工作表时需要帮助，请参阅第 8 页的“导航 Snow-sight 工作表”。要设置工作表上下文，请确保您使用的是 SYSADMIN 角色和 COMPUTE_WH 虚拟仓库。

在本章的后面，我们将创建一些 Snowflake 资源监视器。资源监视器有助于监控 Snowflake 虚拟仓库的使用情况。要准备资源监视器部分，我们需要创建一些 Snowflake 虚拟仓库。最佳做法是使用 SYSADMIN 角色创建新的虚拟仓库：

使用角色 SYSADMIN 使用 WAREHOUSE COMPUTE_WH 创建仓库 VW2_WH 或将仓库替换为 WAREHOUSE_SIZE = MEDIUM

AUTO_SUSPEND = 300 AUTO_RESUME = 真, INITIALLY_SUSPENDED=真: 创建仓库 VW3_WH 或将仓库替换为 WAREHOUSE_SIZE = SMALL

AUTO_SUSPEND = 300 AUTO_RESUME = 真, INITIALLY_SUSPENDED=真: 创建仓库 VW4_WH 或将仓库替换为 WAREHOUSE_SIZE = MEDIUM

AUTO_SUSPEND = 300 AUTO_RESUME = 真, INITIALLY_SUSPENDED=真: 创建仓库 VW5_WH 或将其替换为 WAREHOUSE_SIZE = SMALL

AUTO_SUSPEND = 300 AUTO_RESUME = 真, INITIALLY_SUSPENDED=真: 创建仓库 VW6_WH 或将其替换为 WAREHOUSE_SIZE = MEDIUM

Snowflake 月度账单

组织可以创建和管理多个 Snowflake 账户。每个 Snowflake 账户都会产生费用，这些费用汇总在组织的月度账单中。您的 Snowflake 月度账单总额包括每个账户的三项不同费用：存储费用、数据传输费用和消耗的积分。我们将审查每种类型的收费；您还可以通过查看您的 Snowflake 组织来查看您当前的费用。

使用 Home 图标导航到 Main 菜单。确保您的角色设置为 ACCOUNTADMIN。单击 Admin → Usage，如图 8-1 所示。

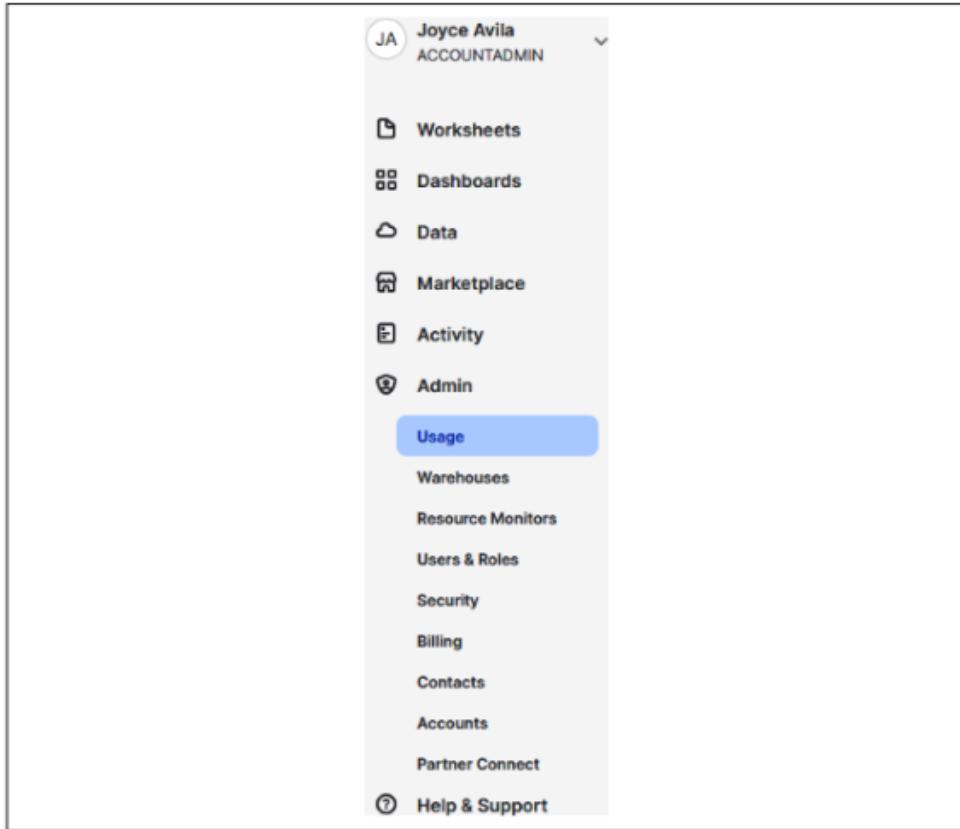


图 8-1. Snowflake Account → Usage 菜单

在 Usage 部分，您可以选择 All Usage Types、Compute、Storage 或 Data Transfer（如图 8-2 所示）。



图 8-2。帐户使用信息

在本节中，我们将重点介绍账户级别的计费。对于账户级别的账单，我们将开始了解仓储费，然后查看转账费用。最后，我们将深入了解 Snowflake 积分。Snowflake 网站上提供了当前租金最高的 Snowflake 服务消耗表定价。

仓储费

Snowflake 月度存储费用是根据每月使用的平均 TB 数计算的，基于账户中存储的所有数据的定期快照。您会注意到，容量定价为预付款提供了很大的折扣。

请注意，虽然存储定价是为您的 Snowflake 账户选择哪个区域的考虑因素，但您还需要考虑哪个区域可以最大限度地减少延迟并让您访问可能需要的任何功能。请务必注意，如果您稍后决定将数据移动到其他区域，则会产生数据传输费用。



尝试更多地使用临时表，因为它们不在历史表中维护，这反过来又降低了历史表的数据存储成本。有关 `transient table` 的更多信息，请参见第 3 章。

数据传输成本

当存储在 Snowflake 中的数据从一个区域传输到另一个区域、在同一云平台内或将数据传出云时，将评估数据传输成本。



Snowflake 不收取数据入口费用。您支付的数据出站费用是向 Snowflake 支付的补偿，这三家云数据提供商在传输数据时收取数据出站费用。

仅当从 Snowflake 卸载数据、使用外部函数或将数据复制到与主 Snowflake 账户托管位置不同的区域或云平台中的 Snowflake 账户时，才收取数据出口费用。

消耗的计算积分

Snowflake 上的资源消耗使用 Snowflake 积分支付，积分是一种计量单位。该度量相当于单个计算节点使用一小时。



虽然 1 积分相当于 1 小时，但虚拟仓库的实际计费是按秒计费的，最少计费 1 分钟。

您可以按需购买积分。按需积分的价格取决于所选的云提供商、区域和服务版本。或者，您可以以协商的折扣率预购积分。

Snowflake 积分仅在您使用资源时消耗。Snowflake 资源的使用或使用方式有三种：运行虚拟仓库、云服务层执行工作时，或者使用无服务器功能（如自动聚类或维护具体化视图）。

Snowflake 运行虚拟仓库所消耗的积分取决于仓库的大小。例如，X-Small 虚拟仓库的积分消耗按每小时 1 个 Snowflake 积分计费。



使用自动暂停和自动恢复等功能以最佳方式管理计算指标的运行状态，有助于管理计费的 Snowflake 消费积分。有关创建和管理虚拟仓库的更多信息，请参阅第 2 章。

云服务层使用的积分最多可免费用于每日消耗的虚拟仓库计算的 10%。任何超过云服务层每日使用阈值 10% 的内容都会计费，并作为 CLOUD_SERVICES_ONLY 虚拟仓库的费用显示在您的 Snowflake 账户中。

使用无服务器功能所消耗的积分的计费费率明显不同，具体取决于无服务器功能。

Snowflake 的许多原生功能（例如微分区和自动计算集群）减少了对积分消耗的需求，从而降低了总体成本。Snowflake 的高度弹性计算和每秒计费模型还有助于保持较低的积分使用费用。这些功能来自 Snowflake 开箱即用；但是，您也可以采取一些措施来更主动地了解和管理您的 Snowflake 消费积分使用情况。

创建资源监控器以管理虚拟仓库使用情况并降低成本

Snowflake 基于使用量的定价很容易理解。这很简单，您只需按实际使用量付费。因此，一个目标是监控您使用的数量。

Snowflake 虚拟仓库在使用时会消耗积分。虚拟仓库消耗的 Snowflake 积分数量取决于两个因素：虚拟仓库的大小和运行时间。

我们在第 2 章中了解到，虚拟仓库的大小增加了两倍，最高可达 4 倍大。因此，X-Small 虚拟仓库每小时消耗 1 个 Snowflake 积分，而 4X-Large 虚拟仓库每小时运行一个 128 积分。



虚拟仓库使用量按秒计费，最短 60 秒。

监控积分消耗将有助于防止意外的积分使用，因此，提供了一种控制计算成本的方法。Snowflake 提供了必要的工具（称为资源监控器），以帮助自动执行监控积分消耗的任务，并根据预设阈值触发操作。

资源监控器是一流的 Snowflake 对象，可应用于用户管理的虚拟仓库和云服务使用的虚拟仓库。

Snowflake 资源监视器具有以下属性：

- 积分配额（Snowflake 积分数）
- 积分使用情况（每个计费周期内消耗的 Snowflake 积分总数）
 - 监控级别：账户或特定虚拟仓库
 - 计划
- 频率：每天、每周、每月、每年、从不一 开始和结束，尽管有结束值并不常见

- 触发器（操作） - 通知 - 通知和暂停

— 立即通知并暂停

从概念上讲，资源监视器规划如图 8-3 所示。

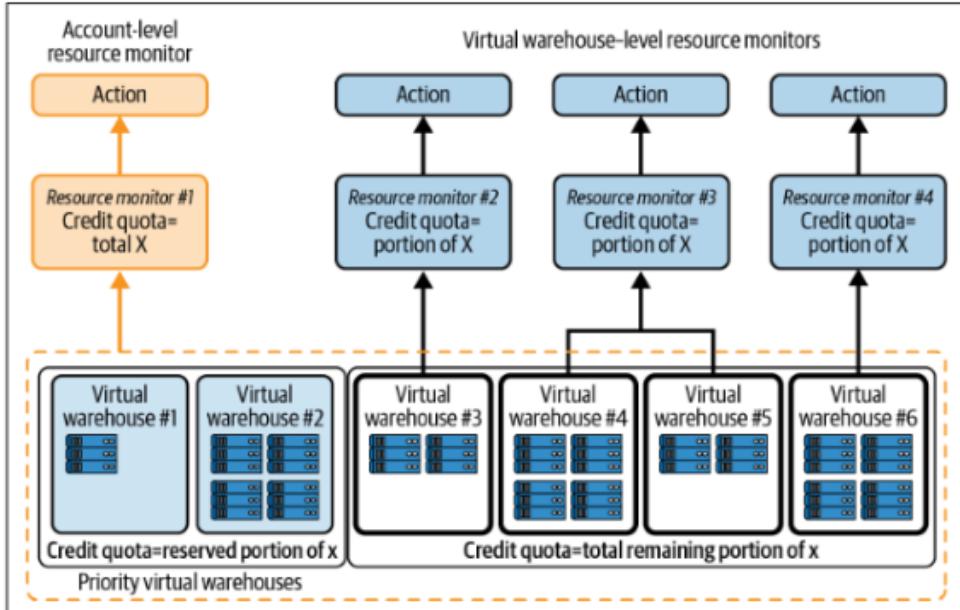


图 8-3。资源监控规划

资源监控规划的建议步骤如下：

- 确定将用于建立账户级资源监控器的总体账户级积分预算。在图 8-3 中，这是 Total X 值。
- 定义优先虚拟仓库，并为所有优先虚拟仓库分配一部分积分预算，在图 8-3 中称为 x 的预留部分。优先虚拟仓库是您不希望暂停的那些虚拟仓库。
- 通过创建虚拟仓库级别的资源监控器，将预算的剩余部分分配给所有其他非优先级虚拟仓库。这样做将限制他们的个人积分使用，以确保为优先虚拟仓库预留足够的预算。请注意，您可以创建的资源监控器数量没有限制。如果需要，您可以为每个单独的虚拟仓库分配一个资源监控器。

4. 如果您不打算使用默认设置，请确定计划。
5. 确定每个资源监视器的操作。最有可能的是，您需要使用“Notify-only”操作建立账户级资源监控器，以便优先虚拟仓库不会被暂停。

资源监控器积分配额

账户级资源监控服务抵扣金配额（以 Snowflake 服务抵扣金表示）基于云服务使用量的总量。但是，仓库级别的积分配额不考虑云服务的每日 10% 调整。只有账户级别的资源监控器才能监控提供云服务的虚拟仓库的积分使用情况。在虚拟仓库级别，只能为用户创建和用户管理的虚拟仓库创建资源监控器。



当资源监控操作设置为 Suspend Immediately（立即暂停）的虚拟仓库达到积分配额阈值时，可能仍会消耗额外的 Snowflake 积分。因此，如果您想严格执行引号，则可能需要考虑在触发器操作的配额阈值中使用缓冲区。例如，将阈值设置为 95% 而不是 100%，以确保您的服务抵扣金额使用量不会超过实际配额。

资源监视器积分使用情况

如前所述，资源监控器可以在账户级别或虚拟仓库级别创建。



账户级资源监控器仅控制在您的账户中创建的虚拟仓库的积分使用情况；Snowflake 为无服务器功能提供了虚拟仓库，例如用于 Snowpipe、自动集群或具体化视图的功能。

可以创建 Snowflake 资源监视器，以对指定的时间间隔或日期范围消耗的 Snowflake 积分数量施加限制。

对于默认计划，资源监控器在创建的虚拟仓库后立即开始跟踪它，并且积分使用情况跟踪在每个日历月开始时重置为 0。

要自定义计划以使资源监控器按特定间隔（例如每天或每周）重置，需要您同时选择频率和开始时间。

选择结束时间是可选的。

资源监视器通知和其他操作

可以创建资源监视器来触发特定操作，例如发送警报通知和/或暂停虚拟仓库。



每个资源监视器最多只能执行 5 个 Notify 操作、1 个 Suspend 操作和 1 个 Suspend Immediately 操作。

默认情况下，资源监视器通知未启用。ACCOUNTADMIN 角色用户只有在提供经过验证的有效电子邮件地址并启用通知后才能收到通知。

要通过 Web 界面启用通知，请确保您的角色设置为 ACCOUNTADMIN。您可以选择 首选项，然后选择 通知。您需要选择通知首选项作为 Web、电子邮件或全部。如果您选择 Email（电子邮件）或 All（全部），系统将提示您输入有效的电子邮件地址。Suspend 操作和 Suspend Immediately 操作之间的区别与是否允许虚拟仓库执行的语句有关。

当触发器导致虚拟仓库暂停时，暂停的虚拟仓库无法恢复，直到出现以下情况之一：

- 资源监控器不再分配给虚拟仓库或已删除。
 - 增加监视器的积分配额或触发器的积分阈值。
-
- 新的月度计费周期开始。

工作分配的资源监视器规则

分配的资源监控规则可以总结如下：

- 可以在账户级别设置账户级资源监控器，以控制账户中所有虚拟仓库的积分使用情况；但是，只能有一个账户级资源监控器。
- 虚拟仓库和虚拟仓库级别的资源监控器之间存在一对多的关系。每个虚拟仓库只能分配给一个虚拟仓库级别的资源监控器；可以将一个资源监视器分配给一个或多个虚拟仓库。

- 创建资源监控器时，都需要 Credit quota 和 Credit usage 属性。如果未明确说明计划，则默认计划适用。默认情况下，服务抵扣金额配额在每个日历月开始时重置。设置 triggers 是可选的。
- 在创建虚拟仓库或账户后，必须先将资源监控器分配给该监控器，然后才能执行监控操作。

让我们使用到目前为止学到的有关资源监视器的知识，为拥有大约 10,000 名员工的大型企业创建资源监视计划（如图 8-4 所示）。我们将在下一节中实施资源监视计划。

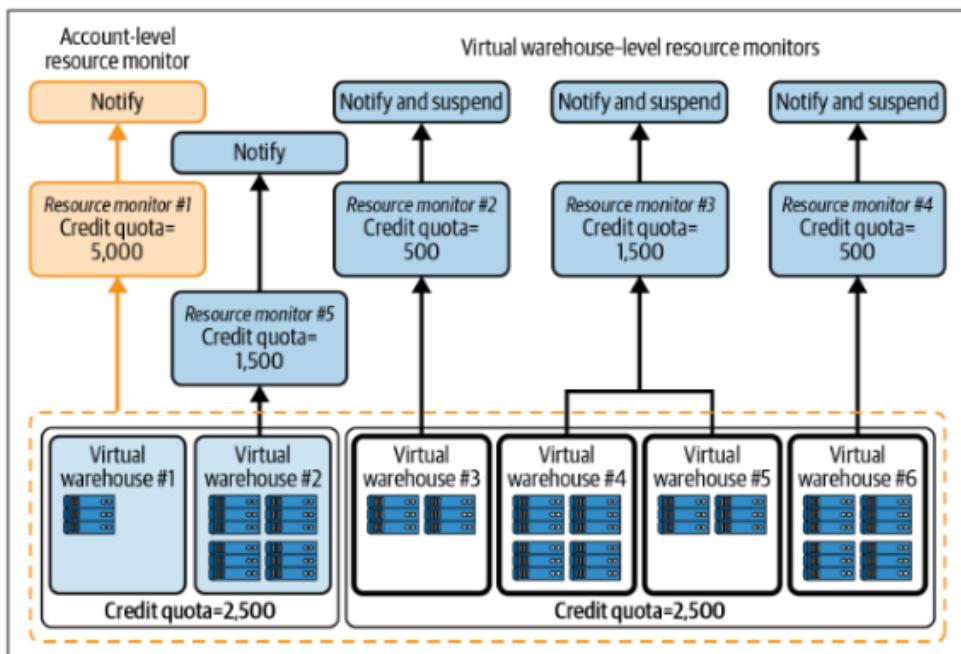


图 8-4。资源监控实施规划示例

假设我们希望将积分消耗限制为 5,000 个积分。我们需要创建一个账户级资源监控器，其中积分配额等于 5000，并且我们希望在达到该积分配额时收到通知。我们将选择使用默认计划。

在我们账户的 6 个虚拟仓库中，我们确定了两个优先的虚拟仓库，我们希望为它们分配一半的信用预算。这意味着从 5000 个总积分中预留了 2500 个积分。其他四个虚拟仓库将需要拆分剩余的 2,500 个积分，我们将创建虚拟仓库级别的资源监控器，以便在积分配额达到时通知和暂停。

已联系到。此外，我们决定在第二个 Priority 虚拟仓库达到 1,500 个信用额度时收到通知。

如果适用，如果账户级资源监控器或虚拟仓库级资源监控器达到其阈值，并且存在与资源监控器关联的暂停操作，则虚拟仓库将被暂停。不过，在我们的示例中，我们为账户级资源监控器建立了“Notify-only”操作。我们还创建了一个与虚拟仓库 #2 关联的资源监视器，该仓库是优先虚拟仓库之一，具有“仅通知”操作。因此，前两个 Virtual Warehouse 永远不会因为任何一个资源监视器达到其阈值而暂停。如果/当其关联的 Resource Monitor 达到其阈值时，其余的 Virtual Warehouse 将暂停。



共享同一资源监控器的虚拟仓库也共享相同的阈值。图 8-4 中的虚拟仓库 #3 和 #4 就是一个示例。这可能会导致一个虚拟仓库的积分使用影响其他分配的虚拟仓库。

用于创建和管理资源监视器的 DDL 命令

默认情况下，Snowflake 资源监视器只能由 ACCOUNTADMIN 角色创建、查看和维护。尽管 Snowflake 资源监视器的创建是 ACCOUNTADMIN 角色独有的，但 Snowflake 帐户管理员可以启用其他角色来查看和修改资源监视器。

如果您使用的是 ACCOUNTADMIN 角色以外的角色，则该角色需要具有两个权限才能查看和编辑资源监视器：

- 修改 • 监控

可以在 Web 界面中创建 Snowflake 资源监视器，也可以在工作表中使用 SQL 创建。对于本章中的示例，我们将使用 SQL 选项来创建和管理资源监视器。

用于创建和管理资源监视器的 Snowflake 数据定义语言（DDL）命令包括：

创建资源监视器

将虚拟仓库分配给资源监视器

更改资源监视器

修改现有资源监视器

显示资源监视器

查看现有资源监视器

删除资源监视器

删除现有资源监视器

要创建资源监控器，您需要为资源监控器分配至少一个虚拟仓库，除非您在账户级别设置监控器。如果您在 Web 界面中创建资源监控器，则需要在创建时分配虚拟仓库。如果您使用 SQL 创建资源监控器，则必须先创建资源监控器，然后使用命令将一个或多个虚拟仓库分配给资源监控器。

让我们首先创建一个账户级资源监控器。导航回 Chapter8 Managing Costs 工作表并执行以下语句：

使用角色 ACCOUNTADMIN: 创建资源监视器 MONITOR1_RM 或将其替换为 CREDIT_QUOTA = 5000

```
TRIGGERS on 50% 会通知  
    75% 的人会通知  
    100% 的人通知 110% 的人通知  
    125% 的人通知:
```

您会注意到，我们创建了五个不同的通知操作，但没有暂停操作，遵循我们之前创建的计划。创建资源监视器后，我们需要将其分配给我们的 Snowflake 帐户：

使用角色 ACCOUNTADMIN: 更改账户集 RESOURCE_MONITOR =
MONITOR1_RM;

接下来，我们将创建一个 Virtual Warehouse 级别的监控器。我们希望为我们的一个优先虚拟仓库创建资源监控器，并且我们将对优先虚拟仓库使用与账户级资源监控器相同的通知操作。创建资源监视器后，我们会将其分配给我们的优先级 Virtual Warehouse：

使用角色 ACCOUNTADMIN: 创建资源监视器 MONITOR5_RM 或将其替换为 CREDIT_QUOTA = 1500

```
TRIGGERS on 50% 会通知  
    75% 的人会通知  
    100% 的人通知 110% 的人通知 125% 的人通知:更改仓库  
    VW2_WH 设置 RESOURCE_MONITOR = MONITOR5_RM;
```

现在，让我们使用显示资源监视器
到目前为止，我们已经创建了：

命令查看资源监视器

使用角色 ACCOUNTADMIN: 显示
资源监视器:

在图 8-5 中，您会注意到我们有一个帐户级和一个虚拟仓库级资源监视器。正如预期的那样，它们都规定了每月的频率，因为我们没有明确说明不同的时间表。

name	credit_quota	used_credits	remaining_credits	level	frequency	start_time	end_time	notify_at
1 MONITOR1_RM	5000.00	0.00	5000.00	ACCOUNT	MONTHLY	2022-05-31 17:00:00.000 -0700		100%, 110%, 125%, 50%, 75%
2 MONITOR2_RM	1500.00	0.00	1500.00	WAREHOUSE	MONTHLY	2022-05-31 17:00:00.000 -0700		100%, 110%, 125%, 50%, 75%

图 8-5. 创建两个资源监视器后的结果

我们已准备好创建另一个 Virtual Warehouse 级别的资源监控器：

使用角色 ACCOUNTADMIN: 创建资源监视器 MONITOR2_RM 或将其替换为 CREDIT_QUOTA = 500

TRIGGERS on 50% 会通知
75% 的人会通知

对 100% 的人做通知，对 100% 的人做暂停，对 110% 的人

做通知，对 110% 的人做suspend_immediate:更改仓库

VW3_WH 设置 RESOURCE_MONITOR = MONITOR2_RM;

到目前为止，一切都很好。现在让我们故意犯一个错误，看看会发生什么。我们将创建一个资源监视器并将其分配给一个虚拟仓库，然后我们将创建另一个资源监视器并将其分配给同一个虚拟仓库。我们知道一个虚拟仓库只能分配给一个资源监视器，因此我们预计，如果第二次尝试将资源监视器分配给同一个虚拟仓库，则会导致命令失败或第二个资源监视器分配覆盖第一个分配。让我们看看会发生什么：

使用角色 ACCOUNTADMIN: 创建资源监视器 MONITOR6_RM 或将其替换为 CREDIT_QUOTA = 500

TRIGGERS on 50% 会通知
75% 的人会通知

对 100% 的人做通知，对 100% 的人做暂停，对 110% 的人

做通知，对 110% 的人做suspend_immediate:更改仓库

VW6_WH 设置 RESOURCE_MONITOR = MONITOR6_RM;

从我们的规划图中您会注意到，我们没有计划第六个资源监视器，因此创建第六个资源监视器是一个错误。不过不用担心。我们稍后将删除不需要的资源监视器。现在，让我们创建应分配给虚拟仓库的正确资源监视器：

使用角色 ACCOUNTADMIN: 创建资源监视器 MONITOR4_RM 或将其替换为 CREDIT_QUOTA = 500

TRIGGERS on 50% 会通知
75% 的人会通知
在 100% 上发出通知，在 100%
上暂停

110% 的人通知 110% 的人 suspend_immediate; 更改仓库
VW6_WH 设置 RESOURCE_MONITOR = MONITOR4_RM;

该语句已成功执行。因此，让我们运行一个命令并查看结果，如图 8-6 所示：

使用角色 ACCOUNTADMIN 显示
资源监视器：

	name	credit_quota	used_credits	remaining_credits	level	frequency
1	MONITOR1_RM	5000.00	0.00	5000.00	ACCOUNT	MONTHLY
2	MONITOR2_RM	500.00	0.00	500.00	WAREHOUSE	MONTHLY
3	MONITOR4_RM	500.00	0.00	500.00	WAREHOUSE	MONTHLY
4	MONITOR5_RM	1500.00	0.00	1500.00	WAREHOUSE	MONTHLY
5	MONITOR6_RM	500.00	0.00	500.00		MONTHLY

图 8-6. 结果 显示资源监视器 命令

我们可以看到，当我们将第四个资源监控器分配给虚拟仓库时，第六个资源监控器似乎已失效。我们可以使用 SHOW WAREHOUSES 命令进行确认：

使用角色 ACCOUNTADMIN 展示
仓库：

图 8-7 显示了我们的语句的结果。正如我们所怀疑的那样，将资源监视器分配给虚拟仓库覆盖了之前的分配。

	name	resource_monitor
1	COMPUTE_WH	null
2	VW2_WH	MONITOR5_RM
3	VW3_WH	MONITOR2_RM
4	VW4_WH	null
5	VW5_WH	null
6	VW6_WH	MONITOR4_RM

图 8-7. 结果 显示仓库 命令

现在，我们可以删除错误创建的资源监视器：

删除资源监视器 MONITOR6_RM；

让我们从规划图中创建最后一个资源监视器。我们将资源监视器分配给两个虚拟仓库：

使用角色 ACCOUNTADMIN 创建资源监视器 MONITOR3_RM 或将其替换为 CREDIT_QUOTA = 1500

TRIGGERS on 50% 会通知

75% 的人会通知

对 100% 的人做通知，对 100% 的人做暂停，对 110% 的人做通知，对 110%
的人做 suspend_immediate;ALTER WAREHOUSE VW4_WH 设置
RESOURCE_MONITOR = MONITOR3_RM;更改仓库 VW5_WH 设置
RESOURCE_MONITOR = MONITOR3_RM;

您可以再次查看 Snowflake 账户中的所有资源监控器：

使用角色 ACCOUNTADMIN 显示
资源监视器：

您会注意到，已删除的资源监视器未显示在资源监视器列表中。此时，我们可以运行另一个语句。我们还可以更进一步，获取尚未分配给虚拟仓库级资源监控器的任何虚拟仓库的列表。但请注意，为了使以下语句正常工作，您只需最近运行 SHOW WAREHOUSES 语句：

```
SELECT "name", "size" FROM TABLE
(RESULT_SCAN(LAST_QUERY_ID()) ) WHERE
"----- monitor" = '#1'.
```

我们可以看到，COMPUTE_WH，即我们的 Snowflake 帐户附带的虚拟仓库以及我们所说的虚拟仓库 #1，尚未分配给资源监视器（如图 8-8 所示）。如果我们查看我们的规划图，我们会发现这与我们所期望的一样。

	name	...	size
1	COMPUTE_WH		X-Small

图 8-8. 结果显示虚拟仓库未分配给资源监视器

我们在工作表中使用 SQL 创建和管理了我们的资源监视器。Alter-native，我们可以在 Web UI 中创建和管理我们的资源。我们需要确保我们使用的是 ACCOUNTADMIN 角色。然后，我们将能够查看刚刚完成的资源监视器，并且我们将能够从 Web UI 创建任何新资源，如图 8-9 所示。

The screenshot shows the Snowflake Web UI with a sidebar on the left and a main content area on the right.

Sidebar (Left):

- User profile: Joyce Avila, ACCOUNTADMIN
- Navigation menu:
 - Worksheets
 - Dashboards
 - Data
 - Marketplace
 - Activity
 - Admin
 - Usage
 - Warehouses
 - Resource Monitors

Main Content Area (Right):

Resource Monitors

5 Resource Monitors

NAME ↑	QUOTA USED	LEVEL	WAREHOUSES
MONITOR1_RM	0.02%	Account	—
MONITOR2_RM	0.00%	Warehouse	VW3_WH
MONITOR3_RM	0.00%	Warehouse	VW4_WH +1
MONITOR4_RM	0.13%	Warehouse	VW6_WH
MONITOR5_RM	0.00%	Warehouse	VW2_WH

图 8-9。显示 ACCOUNTADMIN 角色的资源监视器的 Snowflake Web UI

关于资源监视器的最后一点说明是，对于已创建读取器账户的 Snowflake 提供程序账户，存在一个RESOURCE_MONITORS视图，可用于查询读取器账户的资源监视器使用情况。您可以在第 10 章中找到有关 Snowflake 提供商和读者账户的更多信息。

对成本中心使用对象标记

对象标记是在第 7 章中介绍的，当时我们为数据管理目的创建了标签。在该章的示例中，我们了解到可以将 Snowflake 对象标签分配给表、视图或列，以帮助跟踪正在存储的数据类型以及该数据是否包含敏感信息。

除了出于安全原因使用对象标签外，还可以为对象创建 Snowflake 标签，以帮助跟踪资源使用情况。例如，成本中心标签可以具有 Accounting、Finance、Marketing 和 Engineering 等值。在这种情况下，成本中心对象标记将允许在部门级别提供更精细的洞察。此外，标签可用于跟踪和分析特定短期或长期项目的资源使用情况。

查询 ACCOUNT_USAGE 视图

要获取有关虚拟仓库成本的更精细详细信息，我们可以利用 Snowflake ACCOUNT_USAGE 视图上的查询。



默认情况下，仅向 ACCOUNTADMIN 角色授予对 ACCOUNT_USAGE 视图的访问权限。因此，我们将在示例中使用该角色来查询视图。

此查询根据虚拟仓库的启动时间并假设积分价格为 3.00 美元，提供成本的各个详细信息：

```
使用角色 ACCOUNTADMIN;设置 CREDIT_PRICE = 3.00;使用数据库 SNOWFLAKE;
使用 SCHEMA ACCOUNT_USAGE;选择 WAREHOUSE_NAME、START_TIME、
END_TIME、CREDITS_USED、
```

```
($CREDIT_价格*CREDITS_USED) 正如 SNOWFLAKE_DOLLARS_USED。
ACCOUNT_USAGE。WAREHOUSE_METERING_HISTORY ORDER 按 START_TIME
DESC.
```

此查询汇总了过去 30 天内每个虚拟仓库的成本，假设上一个查询中设置的积分价格为 3.00 美元：

```
选择 WAREHOUSE_NAME, SUM (CREDITS_USED_COMPUTE)
AS CREDITS_USED_COMPUTE_30DAYS, ($CREDIT_PRICE*CREDITS_USED_COMPUTE_30DAYS) AS
DOLLARS_USED_30DAYS FROM ACCOUNT_USAGE。WAREHOUSE_METERING_HISTORY 其中
START_TIME >= DATEADD (DAY, -30, CURRENT_TIMESTAMP ()) 按 1 个 ORDER BY 2
DESC;
```

前面的代码示例提供了一个示例，说明如何查询 ACCOUNT_USAGE 视图以监控虚拟仓库计量，以了解消耗的计算仓库积分。可以创建类似的查询来监控从 Snowpipe、集群、具体化视图、搜索优化和复制消耗的计算积分。Snowflake 教程中的“Snowflake 资源优化：计费指标快速入门”中也提供了一些很好的代码示例。

使用 BI 合作伙伴控制面板监控 Snowflake 使用情况和成本

如果您有权访问 Snowflake 商业智能（BI）和分析合作伙伴工具，则可以访问一个或多个即插即用控制面板，以帮助您监控 Snowflake 的使用情况。以下是当前可用的一些预构建控制面板的列表：

Tableau 仪表板
计算成本概述

西格马
计算成本、存储成本和 Snowflake 成本（读取器账户）

美人
Snowflake 成本和使用情况分析

Microsoft Power BI
Snowflake 使用情况报告

Qlik
Snowflake 使用情况控制面板

Snowflake 敏捷软件交付

管理 Snowflake 成本不仅限于查看 Snowflake 月度账单。当前的 Snowflake 费用是过去决策的结果，包括架构选择、要存储的数据以及如何构建某些查询来访问数据。但组织及其需求总是在变化。因此，预计组织的 Snowflake 环境也会发生变化。

更改任何软件都会产生相关费用，包括更改 Snowflake 软件即服务（SaaS）帐户环境。员工需要时间来架构和构建新软件和/或增强现有软件。对业务的任何破坏也会产生与之相关的成本。最大限度地减少软件开发成本和业务中断，同时尽快进行改进，这需要一套既定的实践和敏捷的软件交付思维方式。

DevOps 是一个术语，用于描述为实现高软件质量而建立的一组实践，这些实践旨在帮助软件开发人员和 IT 运营人员缩短系统开发生命周期并提供持续交付。数据库变更管理（DCM）是 DevOps 的一个子集，专门关注对数据库所做的更改。DCM 面临独特的挑战，但好消息是，Snowflake 的功能（例如零副本克隆和时间旅行）使团队可以更轻松地使用敏捷的 DevOps 方法。

为什么需要 DevOps？

由于适用于组织软件系统和数据的法规和合规性要求，因此维护软件和应用程序更改的审计跟踪至关重要。出于多种原因，快速解决任何 bug 修复也很重要。还应经常发布新功能和请求的增强功能，以便组织在当今的

快节奏的环境，特别是考虑到大量强大的软件应用程序开发工具可用。实现这些目标需要仔细考虑。最重要的是，它需要自动化。

软件开发、测试和发布的自动化是加快创新速度、交付更多改进的应用程序、遵守法规要求并获得更大投资回报的最可能方式。DevOps 的目标是最大限度地减少软件开发、测试和部署的手动方面。

持续数据集成、持续交付和持续部署

通过开发、测试和部署的持续自动化实现持续集成和持续交付（CI/CD）是通过 CI/CD 管道以连接的方式进行的（如图 8-10 所示）。

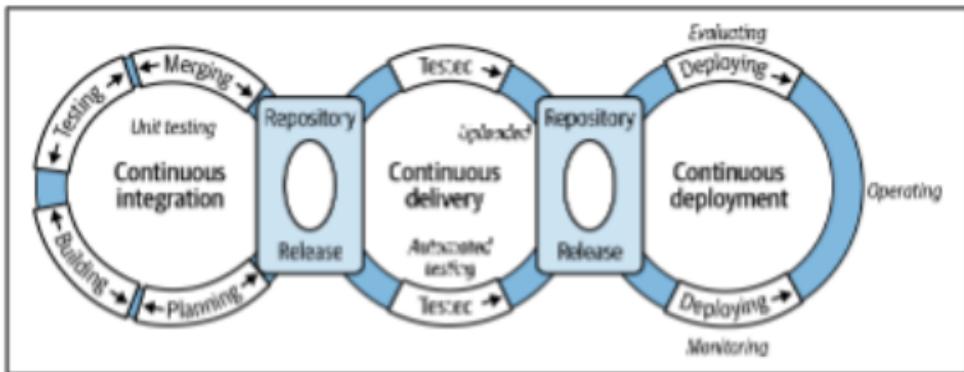


图 8-10 CI/CD 管道

一般来说，持续集成是一种 DevOps 软件开发实践，开发人员通过这种实践定期将其代码合并到一个中央存储库中。当应用程序更改经过自动测试并上传到源代码控制存储库（如 GitHub）时，就会发生持续交付，然后运营团队可以在其中将其部署到生产环境中。持续部署通过自动将更改从存储库发布到生产环境，自动执行管道中的下一阶段。

持续集成、持续交付和持续部署这些术语对每个特定组织的含义取决于组织的 CI/CD 管道中内置了多少自动化。

什么是 Database Change Management?

数据库 DevOps 通常称为数据库变更管理（DCM）。与 DevOps 对软件应用程序的困难相比，基于数据的 DevOps，尤其是对于虚拟仓库的 DevOps，面临着一些独特的挑战。Snowflake 的独特架构解决了其中的许多问题。此外，还提供了特定的 DCM 工具，这些工具可以轻松地与 Snowflake 配合使用。

克服数据库更改的独特挑战

对数据库进行更改会带来一系列独特的挑战，尤其是对于传统的本地数据库，这些数据库必须处理硬件升级的停机问题，并且必须明确地提前备份数据库，以便在需要时可以恢复它们。如果需要回滚，则操作会冻结，并且在恢复过程中不需要执行任何工作。此外，还存在漂移问题，即开发人员可能正在使用与生产版本不同的数据库版本，并且还存在负载平衡问题。

有了 Snowflake，用户不再面临这些主要挑战。硬件修复和升级由 Snowflake 团队处理，对用户没有影响。对于回滚和版本控制，Snowflake 具有 Time Travel 和 Fail Safe，无需进行传统的数据备份。时间旅行和故障安全是 Snowflake 持续数据保护生命周期的一部分。为了缓解漂移问题，Snowflake 用户可以利用上一节中描述的零拷贝克隆来简化不同环境中数据库的加载和种子设定。

鉴于这一切，并且由于 Snowflake 中的大多数内容都可以通过 SQL 实现自动化，因此可以通过为小型部署创建轻量级 DevOps 框架，在 Snowflake 中非常简单地处理 DCM 流程。但是，对于更重量级的 Snowflake DevOps 框架，可以使用 DCM 工具。

适用于重量级 Snowflake DevOps 框架的 DCM 工具

DCM 是指用于管理数据库中对象的一组进程和工具。对于 Snowflake DCM，要考虑的最重要的工具是源代码控制存储库和管道工具。DCM 的托管选项示例包括 AWS CodeCommit 和 CodePipeline、Azure DevOps、GitHub Actions 和 Bitbucket。

除了源代码控制存储库和管道工具之外，您还需要数据库迁移工具来将更改部署到数据库。Snowflake 可以使用多种 DCM 工具。有些主要是 SaaS 工具或数据转换工具，其中许多是社区开发的。可用作数据库迁移工具的工具示例包括 Sqitch、Flyway、snowchange、schemachange、SqlDBM、Jenkins 和 dbt。

最后，需要一个测试框架，例如 JUnit、pytest 或 DbFit。

使用自动化数据集成和交付管道的 DCM 工具可以充分利用前面提到的所有 DevOps 优势，并使您能够实现持续的交付和部署。

如何使用零拷贝克隆来支持开发/测试环境

第 2 章介绍了零副本克隆。由于零副本克隆是仅元数据的操作，因此除非进行更改，否则 Snowflake 克隆的对象不会产生额外的存储费用。

零副本克隆经常用于支持开发生命周期中的开发和测试环境。在本节中，我将演示一些概念，这些概念展示了如何在 Snowflake 敏捷开发中使用零拷贝克隆。对 Snowflake 生产环境的更改可以采用新开发的形式。创建新表就是一个例子。更改也可能以增强 production 中现有对象的形式。我们将深入研究如何将零拷贝克隆用于新的开发。

在我们的示例中，我们将使用我们在第 5 章中学到的一些基于角色的访问控制（RBAC）最佳实践。但是，在实际环境中，您可能需要设置不同的角色和权限才能正确管理 CI/CD 流程。本章的目的是，因为它与基于角色的访问有关，因此为您提供更多的动手实践。

在我们的示例中，假设我们的生产环境中已经存在表 A，并且我们希望在 production 中添加一个额外的表。我们首先要做的是创建生产数据库的克隆，我们可以在其中进行开发。这就是我们将创建新表的位置。我们还希望创建开发数据库的克隆，我们可以在此执行测试或质量保证活动。图 8-11 说明了我们的第一步。

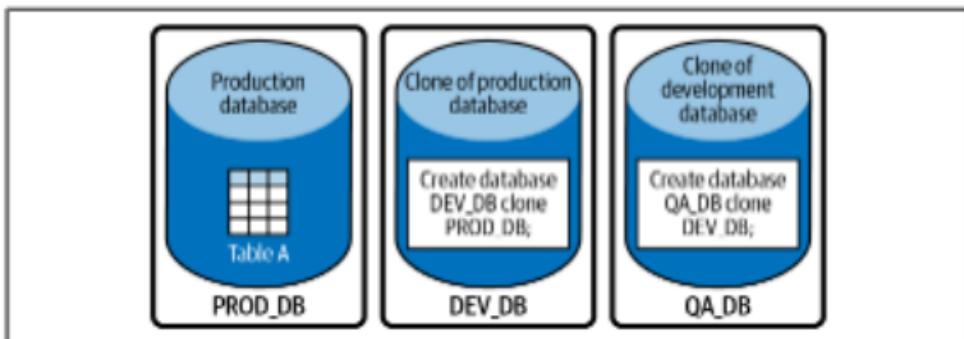


图 8-11. 使用零拷贝克隆进行敏捷开发

让我们首先为我们的生产、开发和 QA 环境创建一些特定的角色。作为最佳实践，我们会将新的自定义角色分配回 SYSADMIN 角色：

```
使用角色 SECURITYADMIN;
创建或替换角色 PROD_ADMIN;创建或替换角色
DEV_ADMIN;创建或替换角色 QA_ADMIN;将角色
PROD_ADMIN 授予角色 SYSADMIN;将角色 DEV_ADMIN
授予角色 SYSADMIN;将角色 QA_ADMIN 授予角色
SYSADMIN;
```

接下来，我们希望为新角色提供一些权限，以便他们可以在账户上创建基于数据：

```
使用角色 SYSADMIN;
将 CREATE DATABASE ON ACCOUNT 授予角色 PROD_ADMIN;将 CREATE
DATABASE ON ACCOUNT 授予角色 DEV_ADMIN;将 CREATE DATABASE ON
ACCOUNT 授予角色 QA_ADMIN.
```

假设我们想为每个管理员分配一个单独的虚拟仓库。我们可以重命名之前创建的三个虚拟仓库：

```
使用角色 SYSADMIN;ALTER WAREHOUSE 如果存在 VW2_WH 则重命名为
WH_PROD;ALTER WAREHOUSE 如果存在 VW3_WH 则重命名为
WH_DEV;ALTER WAREHOUSE 如果存在 VW4_WH 则重命名为 WH_QA;展示
仓库:
```

从命令中可以看到，虚拟仓库现在已重命名。让我们将这些虚拟仓库的使用权授予每个关联的角色：

```
使用角色 ACCOUNTADMIN;使用 WAREHOUSE COMPUTE_WH;将 WAREHOUSE
WH_PROD 的使用权限授予角色 PROD_ADMIN;将仓库 WH_DEV 的使用权限
授予角色 DEV_ADMIN;将仓库 WH_QA 的使用权限授予角色 QA_ADMIN;
```

我们需要创建一个生产环境数据库、架构和表。我们还将向开发管理员授予对数据库的使用权限：

```
使用 ROLE PROD_ADMIN;使用 WAREHOUSE WH_PROD;
创建或替换数据库 PROD_DB;创建或替换 SCHEMA
CHS_SCHEMA;创建或替换表 TABLE_A

(Customer_Account int、金额 int、transaction_ts 时间戳);将数据库
PROD_DB 的使用权授予角色 DEV_ADMIN;
```

现在，开发管理员可以创建生产数据库的克隆，并将克隆数据库的使用权授予 QA 管理员角色：

```
使用 ROLE DEV_ADMIN;使用
WAREHOUSE WH_DEV;
```

创建或替换数据库 DEV_DB 克隆 PROD_DB;将数据库DEV_DB的使用权授予角色 QA_ADMIN;

QA 管理员现在创建开发数据库的克隆，并授予生产管理员角色使用权：

使用 ROLE QA_ADMIN;使用 WAREHOUSE WH_QA;创建或替换数据库 QA_DB克隆DEV_DB;将数据库QA_DB的使用权授予角色 PROD_ADMIN;

现在我们已经准备好进行开发工作了。回到开发数据库，我们使用 DEV_ADMIN 角色创建一个新的开发架构和一个新表：

使用 ROLE DEV_ADMIN;使用 WAREHOUSE WH_DEV;使用数据库DEV_DB;创建或替换架构开发;创建或替换表TABLE_B

(Vendor_Account int, 金额 int, transaction_ts 时间戳);将 SCHEMA DEVELOPMENT 的使用权限授予角色 QA_ADMIN;将 SCHEMA DEVELOPMENT 中所有表的所有权限授予角色 QA_ADMIN;

开发完成后，QA 团队可以通过在测试环境中从开发环境中新创建的表创建新表来执行测试：

使用 ROLE QA_ADMIN;使用 WAREHOUSE WH_QA;使用数据库QA_DB;创建或替换架构测试;创建或替换表 QA_DB。测试。

TABLE_B

如 SELECT * FROM DEV_DB.发展。TABLE_B;将 SCHEMA TEST 的使用权限授予角色 PROD_ADMIN;将 SCHEMA TEST 中所有表的所有权限授予角色 PROD_ADMIN;



可以使用 CREATE TABLE AS SELECT (也称为 CTAS) 命令或 CREATE TABLE LIKE 命令创建 QA 环境中的新表。两者之间的区别在于，后者创建现有表的空副本，而前者创建填充的表：

当 QA 团队完成测试后，生产管理员可以将表复制到生产环境中：

使用 ROLE PROD_ADMIN;使用 WAREHOUSE WH_PROD;使用数据库PROD_DB;使用 SCHEMA CHS_SCHEMA;创建或替换 TABLE TABLE_B SELECT * FROM QA_DB。测试。TABLE_B;

如果您想全面了解创建的内容，可以使用 SYSADMIN 或 ACCOUNTADMIN 角色。使用该角色，您可以看到在开发、生产和测试环境中创建的内容（如图 8-12 所示）。



图 8-12. 本节中创建数据库、架构和表的结果

对于涉及增强现有对象的开发，您很可能需要结合使用零副本克隆和 Time Travel。您可以克隆生产数据库，因为它是特定天数前的数据库，使用脚本对表进行增强，然后重新运行前几天的数据加载。在比较结果时，两者之间的唯一区别应该是增强功能。

如果您想查看此示例中的练习使用了多少计算，可以重新运行我们在本章前面看到的代码，该代码总结了过去 30 天内每个虚拟仓库的成本。

代码清理

我们只需几个快速步骤即可清理 Snowflake 环境。我们可以删除我们的数据库和我们创建的新自定义角色：

```

使用角色 ACCOUNTADMIN;
删除数据库 DEV_DB;删除数据库 PROD_DB;删除数据库 QA_DB;
DROP ROLE PROD_ADMIN;DROP ROLE DEV_ADMIN;DROP ROLE QA_ADMIN;

```

接下来，让我们放下资源监视器：

```

删除资源监视器 MONITOR1_RM;删除资源监视器 MONITOR2_RM;删除资源监视器
MONITOR3_RM;删除资源监视器 MONITOR4_RM;删除资源监视器 MONITOR5_RM;

```

最后，让我们放弃我们的虚拟仓库。请注意，我们将删除我们创建的虚拟仓库，但不会尝试删除 COMPUTE_WH 仓库：

```

DROP WAREHOUSE WH_PROD;DROP WAREHOUSE WH_DEV;DROP WAREHOUSE WH_QA;DROP
WAREHOUSE VW5_WH;DROP WAREHOUSE VW6_WH;

```

总结

在本章中，我们重点介绍了 Snowflake 成本以及如何管理这些成本。通过了解 Snowflake 基于使用量的定价、使用 Snowflake 资源监视器和成本中心的对象标记、查询 ACCOUNT_USAGE 视图以及使用 BI 合作伙伴控制面板，可以实现成本管理。基于使用情况

定价正在改变团队的预算和计划方式。使用本地部署后，如果没有另一个规划周期，他们就无法扩展到超出原始计划。在 Snowflake 上，弹性要大得多，因此，扩展很容易，无需添加更多硬件。

我们还考虑了在 Snowflake 环境中进行更改的成本。人类衰老成本很重要。与管理成本同样重要的是拥有一个运行良好的解决方案。有效且高效的 Snowflake 性能对于成功至关重要。在下一章中，我们将了解 Snowflake 的独特微分区、如何利用集群以及如何提高查询性能。

知识检查

以下问题基于本章中包含的信息：

1. Snowflake 月度账单的三大类是什么？
2. 选择的区域是决定存储成本的重要因素。除了存储成本考虑外，在选择区域时还需要考虑哪些其他重要事项？
3. 按需 Snowflake 积分的价格取决于哪三件事？
4. Snowflake 资源监视器的三个属性是什么？要创建资源监视器，需要在 SQL 语句中包含哪个属性？
5. 可以创建的 Snowflake 资源监视器的数量是否有限制？

6. Snowflake 资源监控器操作的数量是否有限制？
7. 与数据库更改相关的独特挑战有哪些？
8. Snowflake 的零拷贝克隆有哪些独特之处？
9. （也称为 CTAS）命令和 CREATE TABLE LIKE 命令有什么区别？

10. 如果您将虚拟仓库资源监控器分配给已分配给资源监控器的虚拟仓库，会发生什么情况？

这些问题的答案可在附录 A 中找到。

分析和提高 Snowflake 查询性 能

Snowflake 是为来自地面的云而建造的。此外，它的构建是为了消除用户在云中管理数据时通常面临的许多复杂性。微分区、搜索优化服务和材料化视图等功能是 Snowflake 在后台工作以提高性能的独特方式的示例。在本章中，我们将了解这些独特的功能。

Snowflake 还可以通过各种不同的方法轻松分析查询性能。我们将了解分析 Snowflake 查询性能的一些更常见的方法，例如查询历史记录分析、哈希函数和 Query Profile 工具。

准备工作

创建一个标题为 Chapter9 Improving Queries（第 9 章改进查询）的新工作表。如果您在创建新文件夹和工作表时需要帮助，请参阅第 8 页的“导航 Snowsight 工作表”。要设置工作表上下文，请确保您使用的是 SYSADMIN 角色和 COMPUTE_WH 虚拟仓库。我们将使用 SNOWFLAKE 示例数据库；因此，本章不需要额外的准备工作或清理工作。

分析查询性能

查询性能分析有助于识别可能消耗超额积分的性能不佳的查询。有许多不同的方法可以分析 Snowflake 查询性能。在本节中，我们将介绍其中的三个：分析、HASH() 函数和使用 Web UI 的历史记录。

QUERY_HISTORY 分析

我们首先了解INFORMATION_SCHEMA是在第3章。提醒一下，SNOWFLAKE INFORMATION_SCHEMA（通常称为表数据的数据字典）是为每个Snowflake数据库自动创建的。INFORMATION_SCHEMA上使用的QUERY_HISTORY表函数提供了有关对特定数据库执行的查询的非常有用的信息。

您可以执行以下查询来获取有关当前租赁用户在过去一天运行的所有查询的详细信息，并按总运行时间的降序返回记录：

```
使用角色 ACCOUNTADMIN;
使用 DATABASE <数据库名称>;从 TABLE
  (INFORMATION_SCHEMA 中选择 *。QUERY_HISTORY (
    dateadd ('days', -1, current_timestamp () ) ,
    current_timestamp () )
  按 TOTAL_ELAPSED_TIME DESC 排序;
```

此查询的结果为您提供了发现频繁运行的长时间运行的查询所需的信息。通过这种方式，您可以先分析这些查询，以提高性能，从而为您的用户带来最大的改变。

可以通过选择要详细了解的特定用户、会话或虚拟仓库来缩小查询历史记录的关注范围。这些功能如下：

- QUERY_HISTORY_BY_USER
- QUERY_HISTORY_BY_SESSION
- QUERY_HISTORY_BY_WAREHOUSE

HASH() 函数

Snowflake的哈希函数是一个实用函数，用于返回信息，例如查询描述。它不是加密哈希函数。

编写以下查询是为了返回为特定数据库执行的查询（按频率和平均编译时间的顺序）。除了平均编译时间外，还包括平均执行时间以比较这两个时间：

```
使用角色 ACCOUNTADMIN;
使用 DATABASE <数据库名称>;
选择 HASH (query_text)、QUERY_TEXT、COUNT (*) 、
  AVG (compilation_time) , AVG (execution_time) 来自 TABLE
  (INFORMATION_SCHEMA.QUERY_HISTORY (dateadd ('days', -1,
    current_timestamp () ) , current_timestamp () ) )
```

按 HASH (query_text) 分组, QUERY_TEXT
按计数排序 (*) DESC, AVG (compilation_time) DESC :

Web UI 历史记录

我们可以使用 Snowflake Web 界面轻松查看单个查询。Snowflake 提供的图形表示允许我们查看查询处理计划的主要组件以及每个组件的统计信息。我们还可以看到查询的总体统计信息。

要使用 Snowflake Web 界面查看查询, 请单击主菜单中的 Activity, 然后单击 Query History, 如图 9-1 所示。

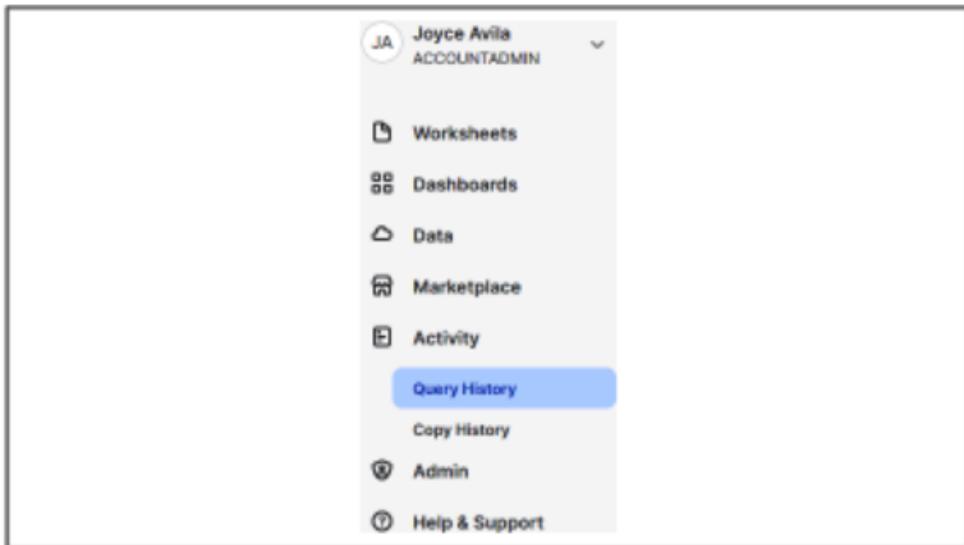


图 9-1. Snowflake 菜单

您应该会看到已执行的所有查询的列表, 最近的查询列在顶部 (如图 9-2 所示)。

Query History						
SQL TEXT	CREATED	LAST RUN	STATE	COMPILE TIME	EXECUTION TIME	ROWS READ
WITH A AS (SELECT * FROM ...)	2023-01-01 10:00:00	2023-01-01 10:00:00	RUNNING	0.000 ms	0.000 ms	0 rows
WITH B AS (SELECT * FROM ...)	2023-01-01 10:00:00	2023-01-01 10:00:00	RUNNING	0.000 ms	0.000 ms	0 rows
WITH C AS (SELECT * FROM ...)	2023-01-01 10:00:00	2023-01-01 10:00:00	RUNNING	0.000 ms	0.000 ms	0 rows
WITH D AS (SELECT * FROM ...)	2023-01-01 10:00:00	2023-01-01 10:00:00	RUNNING	0.000 ms	0.000 ms	0 rows
SELECT * FROM A B C D	2023-01-01 10:00:00	2023-01-01 10:00:00	FINISHED	0.000 ms	0.000 ms	0 rows

图 9-2. 所有查询的历史记录, 按降序排列

使用 Snowflake 的 Query Profile 工具

Snowflake 的 Query Profile 工具提供查询的执行详细信息，以帮助您发现 SQL 查询表达式中的常见错误。您可以使用 Query Profile 了解有关查询性能或行为的更多信息，这将帮助您找到潜在的性能瓶颈并确定改进机会。

从图 9-2 所示的查询列表中，我们可以单击 SQL 文本或 Query ID 列中的特定查询，以查看查询详细信息的图形表示形式。默认选项卡为 Query Details，因此请务必单击 Query Profile 选项卡（如图 9-3 所示）。

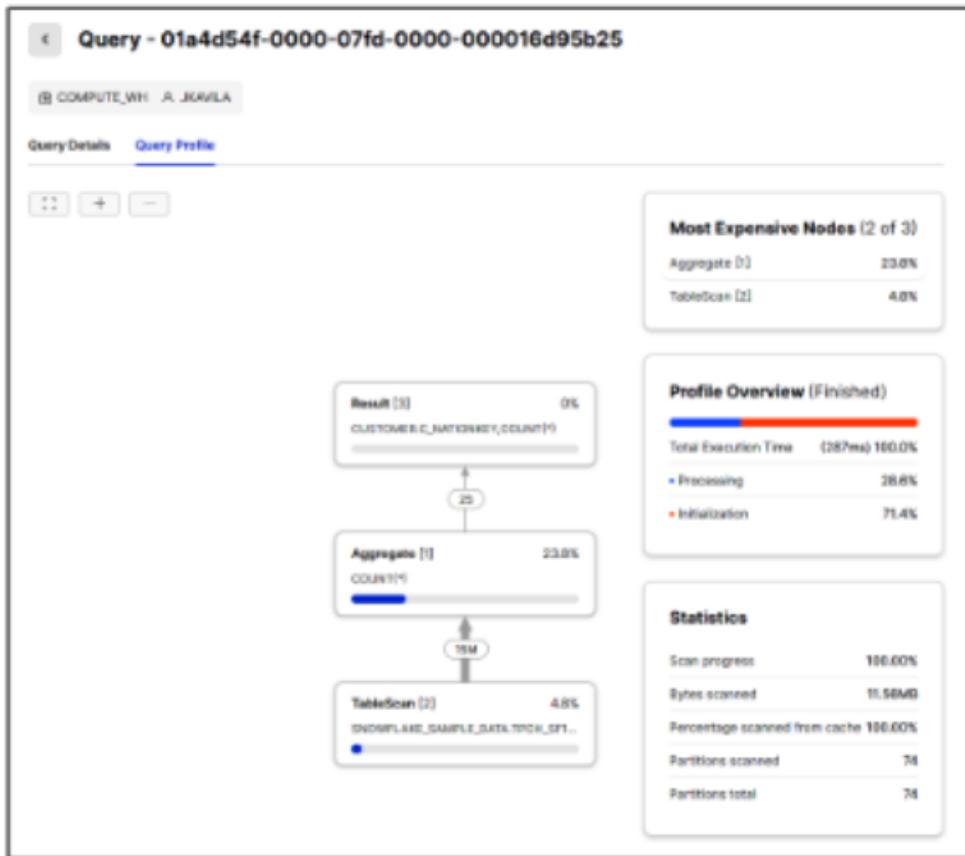


图 9-3。查询用户档案详细信息

您将注意到的一件事是，诸如 and 之类的语句没有查询配置文件（如图 9-4 所示）。

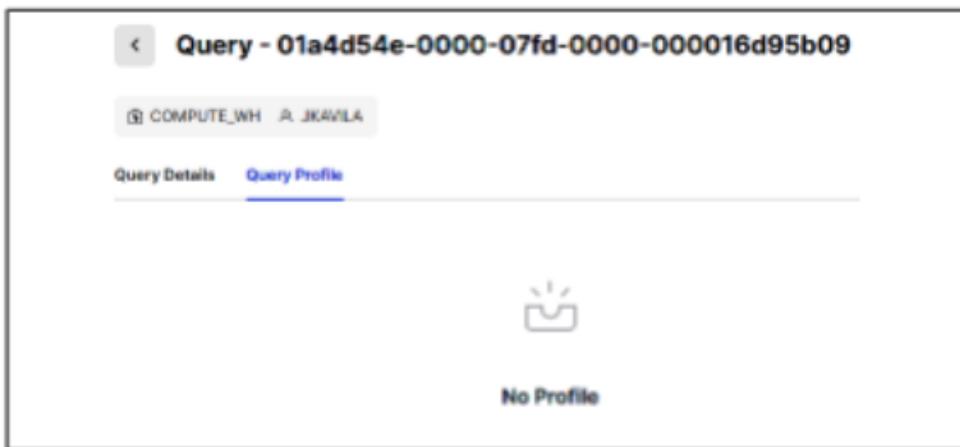


图 9-4. 和语句没有 Query Profile

Query Profile 提供有关单个查询的大量信息。有关如何使用 Query Profile 分析查询的完整详细信息，请访问 Snowflake 在线文档。

了解 Snowflake 微分区和数据集群

存储在 Snowflake 表中的数据动态划分为微分区。无需像传统分割那样手动定义或维护分区。Snowflake 会自动为您处理这些。在将数据加载到 Snowflake 表中时，将存储有关每个微分区的元数据，以便可以利用它来避免在查询数据时对微分区进行不必要的扫描。

为了更好地理解和欣赏 Snowflake 微分区的创新性，我们将首先深入研究分区的主题，以便更好地理解传统分区的工作原理。一旦我们将其与 Snowflake 分区进行比较，Snowflake 微分区的好处将变得更加明显。

分区说明

对关系数据库进行分区将导致大型表被分成许多较小的部分或分区。数据库通常进行分区，以提高可扩展性、协助备份和恢复、提高安全性以及提高查询性能。对于分区数据库，查询只需扫描相关分区，而不是所有数据。

这减少了读取和加载 SQL 操作数据所需的响应时间。

有两种传统的方法可以对关系数据库进行分区：水平和垂直。我们将通过一些示例来演示这两种方法。让我们从一个水平分区示例开始。我们将从一个原始表开始，该表由 8 列和 8 行组成，代表客户从我们的美国网站、零售店和直接渠道购买的商品（如图 9-5 所示）。

CID	Fname	Lname	Reward	Zip	Channel	Amount	Date
1	Arnold	Johnson	Gold	94015	Web	\$2,500	8/7/2022
2	Janice	Switzer	Bronze	76012	Retail	\$450	8/7/2022
3	Amy	Majors	Gold	76015	Web	\$1,220	8/7/2022
6	Marty	Anders	Silver	45232	Direct	\$3,315	8/7/2022
3	Amy	Majors	Gold	76012	Web	\$795	8/7/2022
7	Harold	Webb	Silver	54011	Web	\$490	8/8/2022
5	Nathan	Harris	Silver	76210	Retail	\$1,115	8/8/2022
4	Cathy	Ames	Silver	45615	Web	\$635	8/8/2022

图 9-5. 原始客户交易表

水平分区（也称为分片）将表行存储在不同的数据基础集群中。每个分区具有相同的 schema 和列，但包含不同的数据行（如图 9-6 所示）。分片最重要的好处是，它允许原本太大的数据库能够水平扩展。

Horizontal partition #1 (Last name A-J)							
CID	Fname	Lname	Reward	Zip	Channel	Amount	Date
1	Arnold	Johnson	Gold	94015	Web	\$2,500	8/7/2022
6	Marty	Anders	Silver	45232	Direct	\$3,315	8/7/2022
5	Nathan	Harris	Silver	76210	Retail	\$1,115	8/8/2022
4	Cathy	Ames	Silver	45615	Web	\$635	8/8/2022

Horizontal partition #2 (Last name K-Z)							
CID	Fname	Lname	Reward	Zip	Channel	Amount	Date
2	Janice	Switzer	Bronze	76012	Retail	\$450	8/7/2022
3	Amy	Majors	Gold	76015	Web	\$1,220	8/7/2022
3	Amy	Majors	Gold	76012	Web	\$795	8/7/2022
7	Harold	Webb	Silver	54011	Web	\$490	8/8/2022

图 9-6. 客户交易表中的水平分区

需要注意的是，Snowflake 用户无需担心如何解决横向扩展的问题。Snowflake 是基于列的水平分区，而 Snowflake 多集群虚拟仓库功能意味着不需要分片来横向扩展数据库。第 2 章更详细地介绍了 Snowflake 多集群虚拟仓库功能。在下一节中，我将解释基于 column（基于列）和 horizontally partitioned（水平分区）对于 Snowflake 的含义。

第二种类型的分区是垂直分区，也称为基于列的分区。垂直分区是通过创建具有较少 column 的 table 并将剩余列存储在其他 table 中来实现的；这种做法通常称为行拆分。

垂直分区是一种物理优化技术，但它可以与称为归一化的概念优化技术结合使用。规范化从表中删除冗余列并将其放入另一个表中，同时将它们与外键关系链接。在我们的示例中，外键将是客户 ID（CID）和产品 ID（PID）字段。

继续相同的示例，我们将创建两个垂直分区：一个用于客户数据，另一个用于交易数据（如图 9-7 所示）。

Vertical partition #1 (customer information)					Vertical partition #2 (transaction information)			
CID	Fname	Lname	Reward	Zip	ID	Channel	Amount	Date
1	Arnold	Johnson	Gold	94015	1	Web	\$2,500	8/7/2022
2	Janice	Switzer	Bronze	76012	2	Retail	\$450	8/7/2022
3	Amy	Majors	Gold	76015	3	Web	\$1,220	8/7/2022
6	Marty	Anders	Silver	45032	6	Direct	\$3,315	8/7/2022
7	Harold	Webb	Silver	94011	3	Web	\$795	8/7/2022
5	Nathan	Harris	Silver	76010	7	Web	\$490	8/8/2022
4	Cathy	Amos	Silver	45615	5	Retail	\$1,115	8/8/2022
					4	Web	\$635	8/8/2022

图 9-7。客户交易表中的垂直分区

数据规范化有很多好处，包括能够更轻松地更新数据并消除重复项。数据规范化也是构建机器学习模型的一个重要部分，因为可以在需要时更改数据的分布形状。虽然规范化数据确实会产生成本，但使用 Snowflake 的好处之一是，您可以存储大量数据，然后仅根据需要对数据进行规范化。

最常见的垂直分区方法可能是将静态数据与动态数据分开。目的是根据

数据访问以及表更改的频率。使用产品库存表的相关示例（如图 9-8 所示）将用于演示目的。

PID	Product	Price	Location	Qty
1001	Laptop	\$2,500	L2	296
1002	Printer	\$450	R3	1468
1010	Desk	\$1,220	A3	32
1007	Desktop	\$3,315	L2	1047
1013	Monitor	\$795	A4	997
1012	Scanner	\$490	R3	354

图 9-8. 原始产品库存表

对数据进行建模的一种方法是将静态或缓慢变化的信息存储在一个表中，将动态或频繁变化的信息存储在另一个表中（如图 9-9 所示）。在使用 Snowflake 时，以这种方式对数据进行建模尤其有用，因为它将能够利用 Snowflake 功能（如缓存）来帮助提高性能并降低成本。

Vertical partition #1 (static product information)			Vertical partition #2 (dynamic product information)		
PID	Product	Price	PID	Location	Qty
1001	Laptop	\$2,500	1001	L2	296
1002	Printer	\$450	1002	R3	1468
1010	Desk	\$1,220	1010	A3	32
1007	Desktop	\$3,315	1007	L2	1047
1013	Monitor	\$795	1013	A4	997
1012	Scanner	\$490	1012	R3	354

图 9-9。产品库存表中的垂直分区

总而言之，水平分区按行分解表。水平分区包括所有字段并保存数据的特定子集。相反，垂直分区按列分解表。垂直分区包含与字段子集相关的所有相关数据。包含外键的垂直分区（将它们链接到另一个分区）正在利用规范化。

Snowflake 微分区说明

所有 Snowflake 数据都存储在数据库表中。从逻辑上讲，表格的结构是行和列的列。这种行和列的逻辑结构映射到 Snowflake 物理表结构（称为微分区）。Snowflake 的

独特的微分区方法对用户是透明的，这与传统的数据仓库分区方法不同，在传统的数据仓库分区方法中，用户必须使用数据定义语言（DDL）命令独立处理后期分区。



Snowflake 用户无法对自己的表进行分区。相反，所有 Snowflake 表都会在插入或加载数据时根据数据的顺序自动划分为微分区。

Snowflake 微分区是一个小型物理云存储块，可存储 50 MB 到 500 MB 的未压缩数据。但是，实际的存储块大小甚至更小，因为 Snowflake 数据始终以压缩方式存储，并且 Snowflake 会自动确定每个微分区中列的最有效压缩算法。因此，Snowflake 微分区包含大约 16 MB 的压缩数据。

具有较小的微分区大小有很多好处，其中每个块表示表中的一组行。Snowflake 的微分区允许对列进行高效的扫描，对大型表进行有效的更细粒度的查询修剪，并防止偏差。此外，Snowflake 还存储有关存储在微分区中的所有行的元数据。此外，此元数据包括值的范围和每列的不同值的数量。

Snowflake 专为批量插入和更新而设计。如前所述，Snowflake 中的所有数据都存储在表中，并在插入或加载数据时进行透明分区。用于这些微区块的数据操作语言（DML）操作更改不会导致每个微分区文件中的更新。相反，DML 操作将添加新的分区块和/或删除现有的分区块。这些微分区的添加和删除在 Snowflake 元数据中进行跟踪。

现在让我们来看一个例子。上一节中的 `customer` 和 `transaction` 表将用于演示微分区的工作原理。在我们的示例中，我们假设每个微分区包含 6 条记录。请注意，这是我选择的任意数量，以便于演示我们的示例。在实践中，Snowflake 会自动确定每个微分区的记录数。

假设每个微分区有 6 条记录，图 9-10 演示了如何物理存储 `transaction` 表中的数据。您可以看到，最初加载的 8 条记录最终位于两个微分区中，第二个微分区只有 2 条记录。您还会注意到，数据在微分区中按列物理地组合在一起。

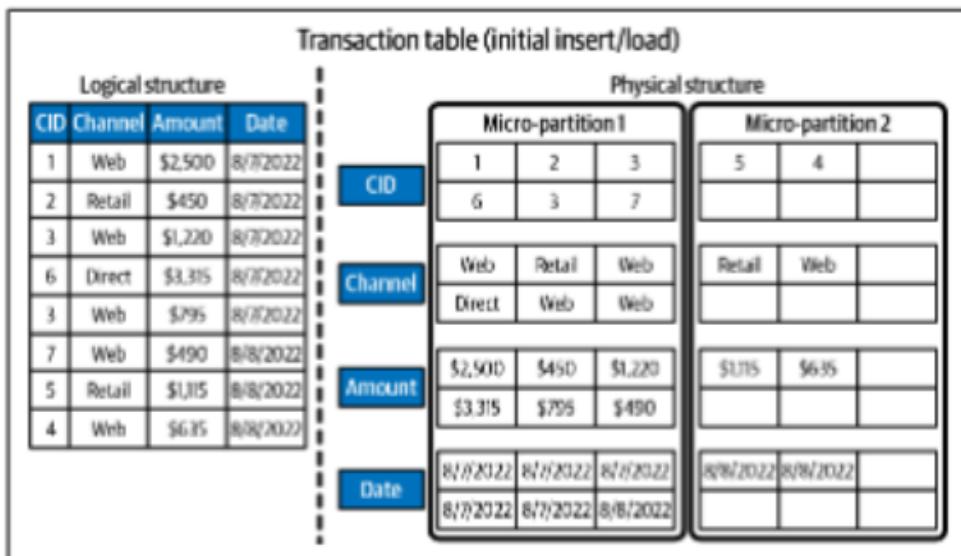


图 9-10. 事务表中的逻辑与 Snowflake 物理结构，如图 9-7 所示

如果第二天添加了三个新记录，则不会将它们添加到第二个微分区中；相反，将创建第三个微分区（如图 9-11 所示）。

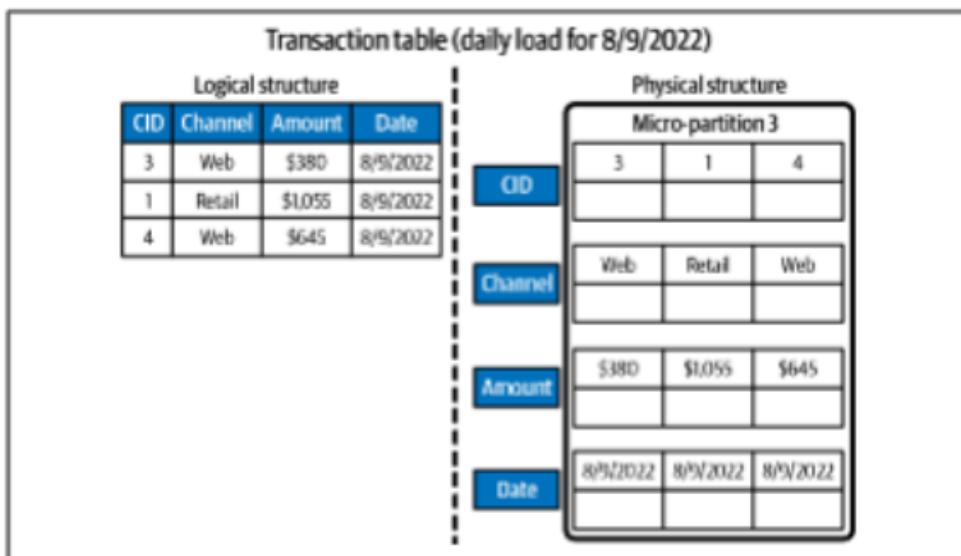


图 9-11. 新记录将创建额外的 Snowflake 微分区

现在，我们来看一下 `customer` 表以及如何存储初始数据加载。您会注意到，虽然我们有 8 笔交易，但我们有 7 个客户，因此将加载 7 个客户记录。这 7 个 `customer` 将存储在两个 micro-partition 中（如图 9-12 所示）。

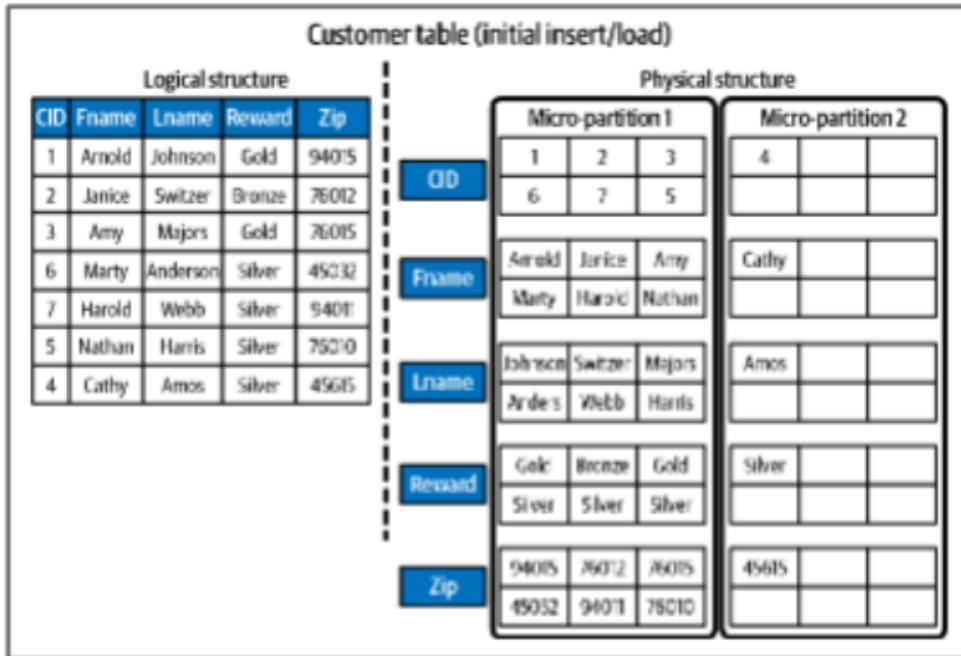


图 9-12。图 9-7 中 `customer` 表的逻辑结构与 Snowflake 物理结构

对于我们的客户表，我们预计会有新的添加，但可能不会有 `delete`。我们还预计表可能会有更新，因为我们正在存储客户的邮政编码，如果客户搬家，则需要更新该邮政编码。

当记录有更新时，存储的数据会发生什么情况？

如图 9-13 所示，Snowflake 在执行更新或删除操作时删除现有分区文件并将其替换为新文件。

Customer table (updated zip code for Nathan Harris)

Logical structure

CID	Fname	Lname	Reward	Zip
1	Arnold	Johnson	Gold	94015
2	Janice	Switzer	Bronze	76012
3	Amy	Majors	Gold	76015
6	Marty	Anderson	Silver	45032
7	Harold	Webb	Silver	94011
5	Nathan	Harris	Silver	94012
4	Cathy	Amos	Silver	45615

Physical structure

Micro-partition 1			Micro-partition 2			Micro-partition 3			
CID	1	2	3	4		5	6	7	
Fname	Arnold	Janice	Amy	Cathy			Arnold	Janice	Amy
	Marty	Harold	Nathan				Marty	Harold	Nathan
Lname	Johnson	Switzer	Majors	Amos			Johnson	Switzer	Majors
	Anders	Webb	Harris				Anders	Webb	Harris
Reward	Gold	Bronze	Gold	Silver			Gold	Bronze	Gold
	Silver	Silver	Silver				Silver	Silver	Silver
Zip	94015	76012	76015	45615			94015	76012	76015
	45032	94011	/6010				45032	94011	94012

图 9-13. 更新记录时对 Snowflake 微分区的更改

事实上，微分区本质上很小，并且存储为列化的行集。以这种方式存储数据，其中列独立存储在微分区中，允许高效的列扫描和筛选。这对于可能具有数百万个微分区的非常大的表尤其有益。使用微分区可以对大型表进行精细修剪。有效和高效的修剪很重要，因为扫描整个表对于大多数查询来说并不是最佳选择。

Snowflake 独特的微分区以及有关每个微分区中行的元数据（包括每列的范围和不同值）的使用为优化查询和高效查询处理奠定了基础。此外，Snowflake 表中的数据按其自然摄取顺序存储，这适用于事务数据等数据，这些数据很少更新或删除，并且通常按日期查询。对于其他情况，自然摄取可能不是最佳物理顺序，尤其是在表数据频繁更新的情况下。对于这些使用案例，我们可以使用集群。

Snowflake 数据聚类分析说明

在上一节中，我们了解了如何创建 Snowflake 微分区。将数据加载到 Snowflake 中时，Snowflake 会根据数据的加载顺序自动创建微分区。这在大多数情况下效果很好，因为 Snowflake 通常使用微分区方法在表中生成聚集良好的数据。但是，有时 Snowflake 表并不是最佳选择，尤其是当 DML 操作频繁发生在包含数 TB 数据的非常大的表上时。发生这种情况时，我们应该考虑定义自己的集群键。

集群键是表中列的子集或表中的表达式。创建集群键的目的是将数据放在表中的相同微分区中。



并非要为所有表创建集群键。在决定是否对表进行聚类之前，建议您对表测试一组具有代表性的查询，以建立按性能基线的查询。表的查询性能以及表的大小应确定是否定义 `clustering key`。通常，当查询运行速度慢于预期和/或随着时间的推移明显降级时，可能是时候创建集群键了。较大的聚类深度是表需要聚类的另一个指示，但聚类深度不是精确的度量。聚类良好的表的最佳指标是其查询性能。

聚类宽度和深度

Snowflake 可以通过考虑重叠的 `partition` 来确定表的聚集程度。微分区可以在分区中彼此重叠；重叠的数量称为聚类宽度。微分区可以在分区中的特定点重叠，该特定点的重叠数决定了聚类深度。让我们看几个例子。

在第一个例子中，我们在微分区中有 6 个分区，所有这些分区都在一个特定值上重叠，并且彼此重叠（如图 9-14 所示）。这是集群最少的微分区列的示例。

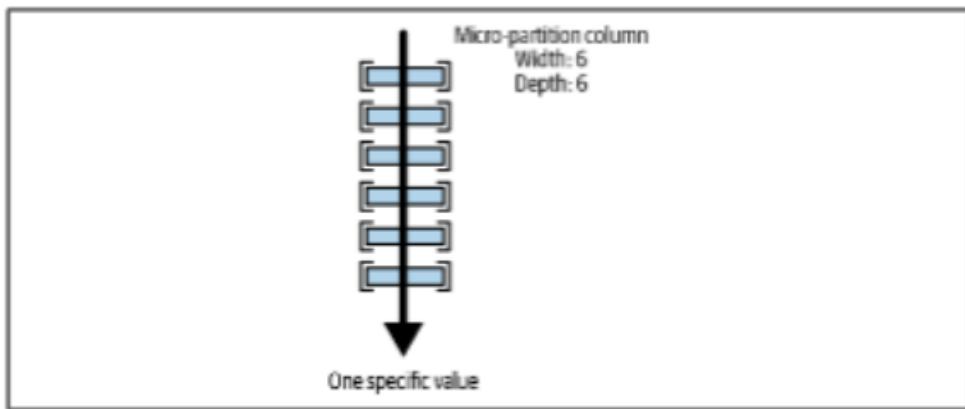


图 9-14. 聚类最少的微分区列

下一个示例是完美群集的微分区。一个分区重叠特定值，没有分区相互重叠（如图 9-15 所示）。

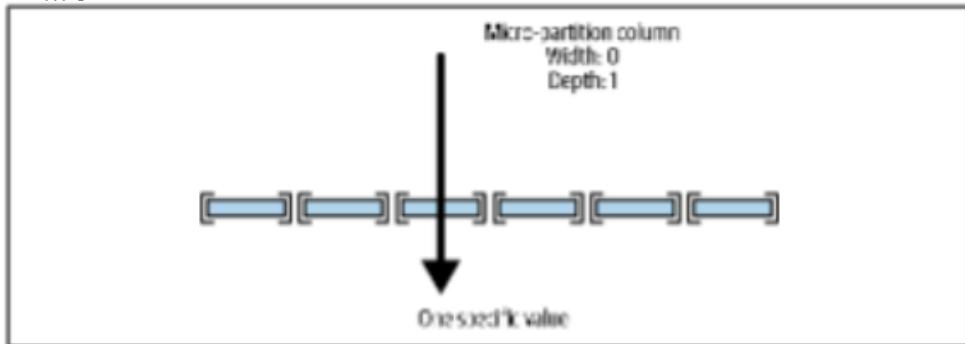


图 9-15. 完美聚集的微分区柱

在第三个示例中，3 个 partition 重叠了特定值，4 个 partition 相互重叠（如图 9-16 所示）。



深度越小，表的聚集性越好。没有微分区的表的聚类深度为零，这表示该表尚未填充数据。

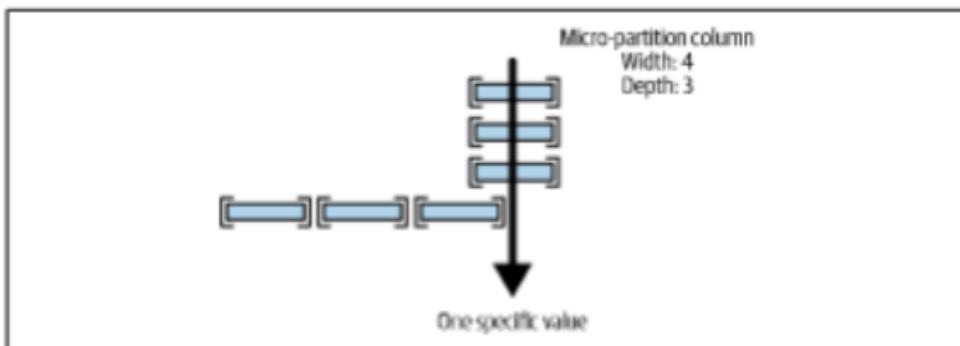


图 9-16. 宽度为 4 且深度为 3 的微分区列

更改我们查询的特定值可能会更改聚类深度，而聚类宽度不会更改（如图 9-17 所示）。

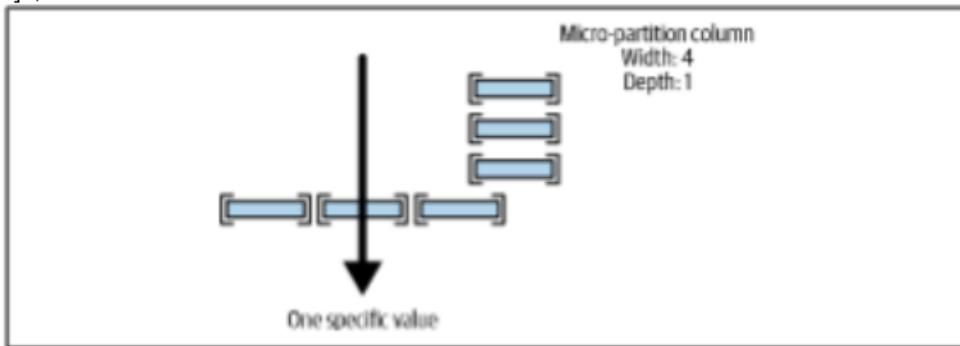


图 9-17. 当特定值发生更改时，微分区列的深度会发生变化

如果我们要查看所有三个微分区，假设它们是表中仅有的三个微分区，我们将能够看到表中的全部值范围（如图 9-18 所示）。

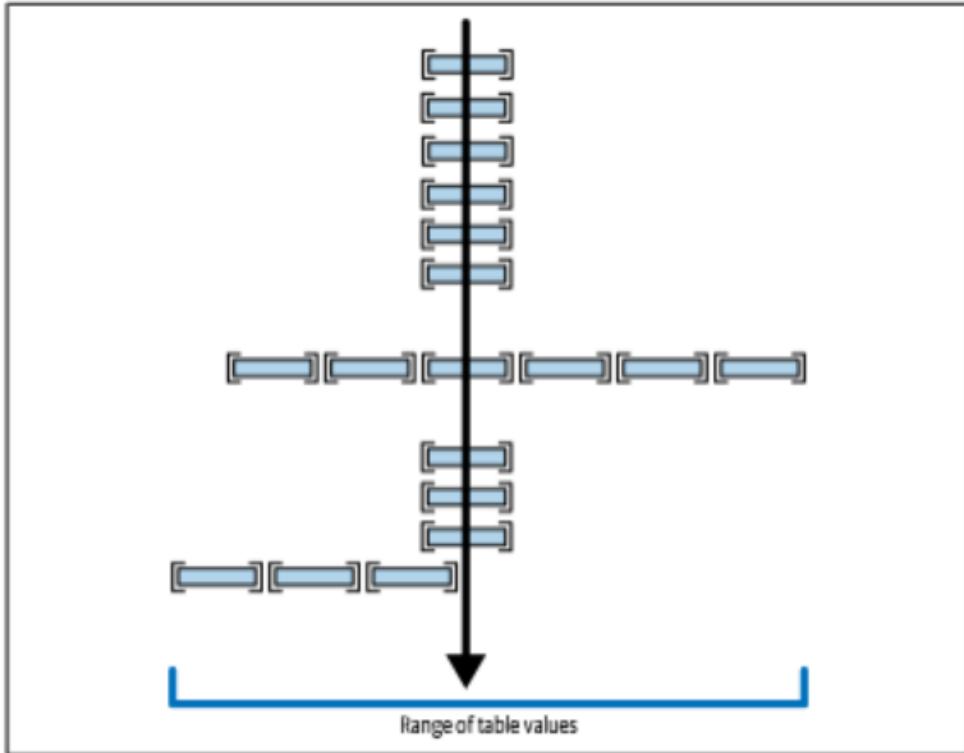


图 9-18. 微分区的总宽度等于表中的值范围

从示例中我们可以看到，表格的宽度为我们提供了值的范围。例如，如果表值的范围跨越 40 个值，我们可以通过计算所有 40 个深度的平均值来获得平均聚类深度。

在 Snowflake 中的表上，您可以使用 SYSTEM\$CLUSTERING_INFORMATION 函数查看集群深度以及其他详细信息。如果表已聚集，则无需指定列。如果表尚未进行聚类，则需要在语句中包含列。下面是一个 col = umn 的示例，您可以查看其详细信息。导航回工作表并执行以下语句：

```
使用角色 ACCOUNTADMIN; 使用数据库 SNOWFLAKE_SAMPLE_DATA; 使用 SCHEMA TPCH_SF100; 选
择系统$CLUSTERING_INFORMATION('客户', '(C_NATIONKEY)');
```

从本节中的示例中，我们可以看到，将数据加载到 Snowflake 表中会根据记录的加载顺序创建微分区。当我们使用 SQL 查询数据时，该子句用于修剪需要扫描的分区的搜索空间。如果当前将数据加载到

table 的搜索效率不高，我们可以创建一个聚类键，允许我们重新排序记录，以便数据与同一个微分区位于同一位置。这样做通常会显著提高性能，尤其是在查询使用集群键作为子句中的筛选条件时。

选择集群键

在决定是否对表进行聚类时，请务必记住，对表进行聚类的主要目标是帮助查询优化器实现更好的修剪效率。因此，建议在默认情况下，在大于 1 TB 的表上定义集群键。

集群键可以包含表中的一个或多个列或表达式。在决定将哪些列用于聚类键时，应首先评估表的数据特征。请务必记住，实现修剪效率的程度在一定程度上取决于用户工作负载中的特定查询。

表数据特征和工作负载注意事项

在决定集群键时，一个好的经验法则是选择一个键，该键具有足够的非重复值以进行有效修剪，但又不能有太多值，以至于 Snowflake 无法有效地对同一微分区中的行进行分组。表中不同值的数量是其基数。表的基数是用于计算选择性的公式的一部分；衡量 table 列的值中有多少变化的度量。选择性是介于 0 和 1 之间的值。值越接近 0 表示列值的种类越少。值 1 表示表 col = ‘mn’ 中的每个值都是唯一的。

我们将使用 Snowflake 示例数据表来演示基数和选择性。让我们首先运行几个查询来获取值的不同计数以及记录总数：

```
使用角色 ACCOUNTADMIN; 使用数据库 SNOWFLAKE_SAMPLE_DATA; 使用
SCHEMA TPCH_SF100; 从 CUSTOMER 中选择 COUNT(DISTINCT
C_NATIONKEY); SELECT COUNT(C_NATIONKEY) FROM CUSTOMER
(客户);
```

结果表明，CUSTOMER 表中的 C_NATIONKEY 列有 25 个不同的值，该表有 1500 万条记录。基数为 25。为了计算选择性，我们可以使用以下语句，它给我们的值为 0.000002；极低的选择性值：

```
使用角色 ACCOUNTADMIN; 使用数据库 SNOWFLAKE_SAMPLE_DATA; 使用 SCHEMA TPCH_SF100; 选
择 COUNT(DISTINCT C_NATIONKEY) / Count(C_NATIONKEY) FROM CUSTOMER;
```

如果我们认为此列可能是一个很好的聚类键候选者，那么在决定之前，我们需要仔细查看数据分布：

```
SELECT C_NATIONKEY, count(*) FROM CUSTOMER group by C_NATIONKEY;
```

查询结果（如图 9-19 所示）表明记录在 s 之间分布得相当均匀。

	C_NATIONKEY	COUNT(*)
1	10	600,195
2	23	599,045
3	1	600,231
4	5	600,226
5	19	600,502
6	7	599,804
7	14	599,406
8	9	601,133
9	4	601,008
10	15	599,480
11	20	600,300
12	13	600,006
13	16	599,835
14	0	599,274
15	21	600,098
16	17	600,098
17	8	599,202
18	12	599,713
19	22	598,913
20	6	600,335
21	2	600,381
22	3	601,469
23	11	600,007
24	18	599,613
25	24	599,726

图 9-19。CUSTOMER 表示例数据中 C_NATIONKEY 列的分布

为了进行比较，让我们看看 `cus = tomer` 表中 `C_NAME` 列的选择性值：

```
使用角色 ACCOUNTADMIN; 使用数据库 SNOWFLAKE_SAMPLE_DATA; 使用 SCHEMA  
TPCH_SF100; SELECT COUNT(DISTINCT C_NAME) / Count(C_NAME) FROM  
CUSTOMER;
```

结果恰好是 1，这意味着 1500 万条记录中的每条记录都有一个唯一的名称。

具有非常高基数的列（例如包含客户名称或纳秒时间戳值的列）具有如此高的选择性值，以至于可能会产生较差的修剪结果。在另一个极端，基数非常低的字段（如 `gender` 或具有 Yes/No 值的二进制列）可能不适合用作聚类键。



如果将具有非常高基数的列定义为列上的表达式，而不是直接在列上定义，则可以更有效地将其用作聚类键。这样，表达式可以保留列的原始顺序。例如，通过将值强制转换为日期而不是时间戳，将时间戳列用作聚类键。如果列名称为 `TIME - STAMP_NTZ`，则语句将类似于以下内容：

```
CREATE OR REPLACE TABLE <表名>  
聚类依据 (to_date(TIMESTAMP_NTZ)) ;
```

效果是将非重复值的数量减少到天数，而不是时间戳的数量。这可能会产生更好的修剪结果。

创建集群密钥

可以在创建表时通过添加命令

`CLUSTER BY (column name(s))` 添加到语句中：

```
ALTER TABLE <表名> CLUSTER BY (列名) ;
```

可以在创建表后添加或更改集群键，并且可以随时删除。



在创建集群键时，建议您选择不超过 3 或 4 列或表达式。虽然列数是一个重要的考虑因素，但最重要的是专注于为聚类键选择正确的列或表达式，并将它们按最佳顺序排列。

如果使用多个列或表达式对表进行聚类，则建议优先考虑选择性过滤器中最积极使用的列。对于使用基于日期的查询的表，为集群键选择日期列是有意义的。



如果选择前导字符相同的字符串数据类型列，则可以使用表达式截断常见的前导字符。

如果在考虑其他优先级较高的列后仍有空间容纳其他群集键，则最好包含联接谓词中经常使用的列。在为集群键选择列时，应仔细充分考虑查询工作负载和 DML 工作负载之间的平衡。

也就是说，作为一般规则，Snowflake 建议假设数据均匀分布，从最低到最高基数对所选列进行排序。这样做将允许具有较少 distinct 值的列在聚类键顺序中排在前面。当然，此一般规则可能有例外，因此不要让它覆盖本来更好的选择顺序。

重新聚集

随着时间的推移，表中的数据可能会变得不那么聚集，因为 DML 操作是在表上按格式进行的。为了保持最佳性能，可能需要定期对表进行重新聚集。重新聚集不需要手动干预或维护；Snowflake 会根据需要自动对表进行重新聚类。



重新聚类会消耗积分，还会产生存储成本。数据量和表的大小决定了执行重新集群将消耗的积分数。额外的存储成本是由于从重新聚类中删除的数据在 Time Travel 保留期和随后的故障安全期内仍然可用。

集群只是我们可以用来提高查询性能的一种工具。另一个选项是创建具体化视图。

具体化视图的性能优势

物化视图首次在第 3 章中介绍。具体化视图包含表中数据子集的最新副本。因此，物化视图可用于提高查询性能。

请务必记住，与其他 Snowflake 安全对象类似，材料化视图由角色拥有。但是，在创建实例化视图时，它们不会自动继承基表的权限。



更改基表不会更改具体化视图；相反，需要创建一个新的 materialized 视图。更具体地说，向基表添加新列不会导致将新列添加到具体化视图中。如果对基表中已有的列进行了更改，例如更改标签、删除列，则该基表上的所有物化视图都将被挂起。

正如我们在第 3 章中看到的，物化视图实现了安全优势。他们还在几个不同的使用案例中实现了性能优势：

- Snowflake 具体化视图可以提高重复使用相同子查询结果的查询的性能。
- 查询具体化视图通常比使用缓存结果慢，但比查询表快。
- 当查询用于外部表数据时，具体化视图特别有用，其中数据集存储在外部阶段的文件中。这是因为与查询外部表相比，查询内部表时的查询性能通常更快。

具体化视图支持聚类分析。重要的是，您可以将具体化视图聚集在不同的列上，而不是用于聚集定义具体化视图的基表的列。



作为一般经验法则，当查询结果经常使用且结果不经常更改时，尤其是在查询消耗大量资源时，建议使用具体化视图而不是常规视图。

需要权衡创建具体化视图的性能优势与成本考虑。与不缓存数据的常规视图不同，物化视图确实使用存储。因此，存在与具体化视图相关的存储费用。配对视图也会消耗配额进行维护。

虽然具体化视图有很多好处，但也有一些限制。`materialized` 视图不支持 DML 操作，只能单独查询一个表，不能查询另一个视图或用户定义的函数。此外，具体化视图不支持联接，并且不能包含以下任何子句：`,`，`,`，`LIMIT` 或`.`

探索其他查询优化技术

除了对表进行聚类和创建具体化视图之外，Snowflake 的搜索优化服务是优化查询性能的另一种方法。Snowflake 的搜索优化服务与其他两种查询优化技术之间有一个重要的区别。具体化视图和簇状表可以加快范围搜索和相等搜索的速度；搜索优化服务只能加快相等搜索的速度。

相等搜索使用相等谓词，例如

`value` 而 `range` 搜索查询时间范围。

除了相等搜索之外，搜索优化服务还支持使用以下数据类型的条件表达式在列表中进行谓词搜索：

- 定点数，例如`和`
- `DATE` 和
- 瓦查尔
- 二元的

搜索优化服务

Snowflake 的搜索优化服务的目标是提高选择性点查找查询的性能，选择性点查找查询是仅返回一个或少量非重复行的查询。在 Snowflake 中，我们很可能会运行更多的聚合查询，而不是在表中大量行中仅返回几行的查询。因此，搜索优化服务可能是最佳选择的用例可能并不多。但是，对于可以从 Snowflake 的搜索优化服务中受益的非常具体的使用案例，查询性能的改进可能会非常显著。



Snowflake 搜索优化服务在 Snowflake Enterprise 及更高版本中可用。

要利用搜索优化服务，您需要向该服务注册表。将搜索优化添加到 Snowflake 表时，维护服务将在后台运行。我们知道搜索优化服务用于根据子句中的 using 查找一条或少量记录，但幸运的是，我们不必决定何时使用该服务有意义。执行查询时，Snowflake 优化器会自动决定何时使用搜索优化服务。



在决定是否向该服务注册表之前，您可能需要考虑估算向表添加搜索优化的成本。您可以使用函数 SYSTEM\$ESTIMATE_SEARCH_OPTIMIZATION_COSTS 来执行此操作。

确定向表添加搜索优化是有意义的后，可以使用以下命令执行此操作：

```
ALTER TABLE [IF EXISTS] <表名> 添加搜索优化;
```

搜索优化服务是一项会产生计算成本的无服务器功能。由于搜索优化服务维护搜索访问路径，因此还会产生与该服务相关的存储成本。成本与启用该功能的表数、这些表中非重复值的数量以及表中更改的数据量成正比。

查询优化技术比较

到目前为止，我们已经深入研究了三种最常用的 Snowflake 查询优化技术。表 9-1 总结了不同优化技术之间的一些差异。

表 9-1 Snowflake 查询优化技术比较

表聚类	物化视图	搜索优化服务
存储成本 X X		
计算成本 X X 无服务器功能 X X 用例 用于大型表:		
- 通常适用于除全表读取之外的所有工作负载	在重复使用同一子查询或外部表查询时最有利	当您需要根据大型表中的选择性点查找查询访问特定行时



在大多数情况下，在集群表上使用 Snowflake 的搜索优化服务是多余的，除非查询位于主集群键以外的 `columns` 上。

总结

在本章中，我们学习了分析 Snowflake 查询性能的几种不同方法。我们发现，`QUERY_HISTORY` 分析使我们能够按用户、会话或虚拟仓库评估查询。我们使用哈希函数创建了一个语句，该语句返回按频率和平均编译时间顺序执行的数据库查询列表。我们查看了如何从 Snowflake Web UI 访问 Snowflake 查询配置文件工具。

我们了解了三种 Snowflake 优化技术：聚类、具体化视图和搜索优化服务。我们还深入研究了分区，以更好地了解为什么 Snowflake 独特的微分区如此创新和有益。

到目前为止，我们一直专注于了解 Snowflake 架构和功能的所有不同部分。我们精心奠定了 Snowflake 的基础地位以及它的独特之处。展望未来，我们将能够利用我们对 Snowflake 的深入了解来探讨一些真正有趣且具有重要意义的主题。

下一章将深入探讨 Snowflake 的数据共享功能和产品/服务。Snowflake 创建了其安全数据共享功能，功能非常强大，但易于设置和维护。

代码清理

不需要代码清理。

知识检查

以下问题基于本章中包含的信息：

1. 函数和 `QUERY_HISTORY_BY_USER` 函数有什么区别？
2. 您将使用 Snowflake 中的函数来做什么？
3. 关系数据库的分区传统方法是什么？
4. 什么是 Snowflake 微分区，为什么它如此重要？

5. 当记录有更新时，存储在 Snowflake 表中的物理数据会发生什么情况？
 6. Snowflake 如何确定表的聚集程度？
 7. 您可以使用哪个 Snowflake 函数来查看表中的聚类深度？
 8. 在决定集群键时，什么是好的经验法则？
 9. 您应该为集群键选择的最大列数或表达式数是多少？
10. 以下哪种查询优化技术是无服务器的？
- 表聚类
 - 具体化视图
 - 搜索优化服务

这些问题的答案可在附录 A 中找到。

配置和管理 安全数据共享

数据共享支持业务合作伙伴之间的协作，提供通知客户的机会，并为您提供了一种从客户和供应商处获取实时信息的方法。数据共享还为您提供了通过数据获利的机会。但是，使用传统数据共享选项时，存在有效和高效的数据共享障碍。首先，传统的数据共享选项（如 FTP 传输、API 调用、发送和接收 CSV 文件以及 ETL 工具和流程）通常需要构建复杂的基础设施。

同样，传统的数据共享方法（涉及传输数据副本以及重建和存储重复数据）成本高昂。传统的数据共享选项通常没有单一的事实来源，并且由于延迟访问过时且有时不完整的数据而产生的可操作见解较少。然而，通过使用 Snowflake 的安全数据共享技术，可以克服其中的许多共享挑战。

安全数据共享使数据可以通过实时连接访问，以便数据使用者可以自动实时使用更新的数据。业务逻辑和数据也可以共享。除了安全、高效和将数据货币化的好方法之外，Snowflake 安全数据共享还让您高枕无忧，因为您可以随时撤销访问权限。

由于 Snowflake 独特的架构，这些强大的功能成为可能。

Snowflake 架构数据共享支持

在第 2 章中，我们了解了传统的数据平台架构如何具有固定的计算和存储比率。这与 Snowflake 的多集群共享数据架构形成鲜明对比，后者具有可以自动扩展和收缩的存储功能，并且具有可以同时读取和写入以及即时调整大小的独立计算集群。

当您将计算与存储分离并共享数据访问控制时，就像 Snowflake 一样，您可以让多个虚拟仓库同时处理相同的数据。因此，Snowflake 的架构支持拥有单一事实来源的能力。

如果您在同一云提供商和区域内共享数据，则无需复制数据即可共享数据。如果您将数据共享到其他区域或不同的云提供商，Snowflake 会复制数据。但是，它是通过自动履行来实现的，因此您无需执行任何手动操作。Snowgrid 是一个术语，用于描述数据复制，以便于数据共享。

Snowgrid 的力量

Snowgrid 是全球性的，可以无缝连接可能按区域或云分隔的 Snowflake 用户。

Snowgrid 通过自动执行进行复制来实现这一点。即使在跨云和区域共享数据时，共享在事务上也是一致的，这意味着事实来源仍然得到维护。Snowgrid 中包括所有原生跨云治理控制，它们作为实现联合治理的基础构建块。有关数据库复制细节的更多详细信息，请参阅第 7 章。

借助安全数据共享和 Snowgrid 的强大功能，Snowflake 提供商可以与全球数据使用者创建和共享任意数量的数据共享。每个共享都封装了必要的访问控制，因此它是安全的。一旦提供商的出站共享在数据使用者的账户中显示为入站共享，使用者就可以从该共享创建数据库。元数据指向原始数据源，因此数据继续仅存在于提供商账户中。

数据共享使用案例

许多行业已经开始采用 Snowflake 数据共享功能的功能。例如，金融服务和保险公司利用 Snowflake 的安全存储和数据共享功能来高效摄取物联网（IoT）数据以及交易和点击流数据。使用第三方 Snowflake 数据（例如 Equifax 数据）丰富该数据，可以提供更完整的客户视图。拥有完整的 360 度客户视图为组织提供了

有机会识别高价值客户并确保他们在每个接触点都有良好的体验。

医疗保健和生命科学组织可以利用 Snowflake 数据共享功能来取代几十年来的数据共享方法，例如 FTP、DVD 和其他物理介质。事实上，存储 PB 级受保护健康信息（PHI）和个人身份信息（PII）的付款人需要与保险经纪人、提供商和其他供应商共享，他们正在转向 Snowflake 作为解决方案，因为医疗保健行业的数据共享是强制性的和受监管的。拥有大量数据的医疗保健提供商通常与医疗保健联盟合作或隶属于大学；因此，实时共享数据而不是批量共享数据是绝对必要的。

通过使用 IoT 数据优化车队路线的卡车运输公司和能够通过与物流合作伙伴共享数据来改善供应链的零售商是结合 Snowflake 实时数据共享能力的另外两个行业示例。此外，一些企业公司使用不同的 Snowflake 帐户进行开发、测试和生产目的。数据共享提供了整合强大的审计控制并以受控方式移动数据的能力，而不必准备平面文件并将其重新加载到不同的环境中。

出版商、广告代理商和品牌需要快速开发一种新的数据共享辅助数字，因为我们正在快速接近一个不存在第三方 cookie 的世界。随着第三方 Cookie 被弃用，营销人员将需要找到新的方法来识别在线人员，以便他们能够优化营销活动并继续个性化消息。

Snowflake 对统一 ID 2.0 的支持

Cookie 于 1994 年开发，是在您访问网站时安装在您的网络浏览器上的一小段软件代码，以方便再次访问和跟踪，是一种过时的技术。如今，互联网使用更频繁地发生在移动应用程序和连接电视设备上，在这些设备上，cookie 大多无关紧要。Cookie 通常包含大量个人数据，这些数据可能会在未经您同意的情况下识别您的身份；因此，它们受欧盟《通用数据保护条例》（GDPR）和《电子隐私指令》（EPD）的约束。EPD 最终将被《电子隐私条例》（EPR）取代。Apple、Firefox 和 Google 等主要平台已经开始限制第三方 cookie 的使用。在不久的将来，互联网平台隐私政策的变化甚至可能导致 cookie（互联网的第一个通用标识符）过时。如果您有兴趣了解有关 ePrivacy 的更多信息，可以访问 <https://cms.law/en/deu/insight/e-privacy>。

对用户隐私和更多监管的担忧使考虑新方法成为可能。Unified ID 2.0 由 The Trade Desk 开发，是一个开源的、

行业监管的身份解决方案，为用户提供匿名化、更高的透明度和更好的控制。一个人的 UID 2.0 是从电子邮件地址生成的一串随机的数字和字母，无法逆向工程为电子邮件地址或任何其他形式的身份证明。目前，Unified 2.0 已被 The Washington Post、Oracle、Nielsen 和 Tubi TV 等接受。

Snowflake 支持将 Unified ID 2.0 与安全数据共享一起使用，尤其是与数据净室一起使用，本章稍后将对此进行更详细的讨论。因此，Snowflake 客户可以以更注重隐私的方式直接在 Snowflake 平台中加入第一方和第三方数据。Snowflake Data Cloud 和 Unified ID 2.0 的强大功能提供了客户的单一视图。

Snowflake 安全数据共享方法

有四种不同的方法可以实现 Snowflake 的安全数据共享。最简单的方法是进行账户到账户的数据共享，这样您就可以直接与另一个账户共享数据，并让您的数据显示在他们的 Snowflake 账户中，而无需移动或复制它。例如，另一个帐户可能属于不同的内部业务部门，或者完全属于不同的组织。

您可以使用 Snowflake 的 Marketplace，这是一种公共安全数据共享方法，可将全球数据提供商与世界各地的消费者连接起来。

您甚至可以创建自己的 Snowflake Private Data Exchange 以与他人协作；您可以控制谁可以加入数据交换，哪些成员可以提供数据、使用数据或两者兼而有之。

或者，您可以使用 Data Clean Room，这是一个在两方或多方之间共享数据的框架；数据根据特定准则进行汇总，以便对 PII 进行匿名化和处理，并且可以允许符合隐私法规的方式进行存储。图 10-1 总结了四种主要的 Snowflake 安全数据共享方法。



图 10-1. 四种 Snowflake 安全数据共享方法

现在，我们将进行一些准备工作，然后我们将探索每种 Snowflake 安全数据共享方法。

准备工作

我们已准备好创建一个名为 Chapter10 Data Sharing 的新工作表。如果您在创建工作表时需要帮助，请参阅第 8 页上的“导航门控 Snowsight 工作表”。

如果您使用的是 Snowflake 免费试用账户，您会注意到，当您首次登录时，您会看到您被分配了 SYSADMIN 角色。使用数据共享需要为您的角色分配某些权限，除非您使用的是 ACCOUNTADMIN 角色。在本章的课程中，我们将使用 ACCOUNTADMIN 角色，因此请确保正确设置您的角色。提醒一下，您可以使用下拉菜单或工作表中的 SQL 状态来更改您的角色。

接下来，我们将为本章中的动手示例创建一个新的数据库和表。在新的 Data Sharing 工作表中，执行以下命令：

```
使用角色 ACCOUNTADMIN; 使用 WAREHOUSE COMPUTE_WH; 创建或  
替换数据库 DEMO10_DB; 使用 SCHEMA DEMO10_DB。公共; CREATE  
OR REPLACE TABLE SHARINGDATA (i 整数);
```

让我们继续看一下与我们共享了哪些数据。通过单击主页图标导航回主菜单，然后单击菜单中的 Data → Private Sharing 选项。在图 10-2 中可以看到，我当前可以访问来自两个直接数据共享的数据。

The screenshot shows the Snowflake Data Sharing interface. On the left, there is a sidebar with a user profile for 'Joyce Avila' (ACCOUNTADMIN) and a navigation menu with options like Worksheets, Dashboards, Data, Databases, Private Sharing (which is selected and highlighted in blue), Marketplace, Activity, Admin, Help & Support, and Classic Console. The main content area is titled 'Shared With Me' and shows a search bar. Below it, under 'Direct Shares', there are two items:

- SNOWFLAKE** - ACCOUNT_USAGE: Shared 2 months ago.
- SFC_SAMPLES** - SAMPLE_DATA: Shared 2 months ago.

图 10-2. 与我（账户管理员）共享的直接数据共享

在本书的整个章节中，我们使用了 SAMPLE_DATA 份额作为示例。我们还讨论了 Snowflake ACCOUNT_USAGE 份额。我们将在后面的部分中查看有关此特殊 ACCOUNT_USAGE 共享的详细信息。

Shared Data（共享数据）菜单选项的默认选项是 Shared With Me（与我共享），您可以在屏幕的顶部中间看到该选项。要查看您的任何出站数据共享，您可以单击 Shared By My Account，如图 10-3 所示。



图 10-3. 我与他人共享的数据

Snowflake 的直接安全数据共享方法

直接共享是 Snowflake 最简单的数据共享形式，当 Snowflake 提供商创建出站数据共享，然后成为使用者的 Snowflake 账户中的入站共享时，就会产生直接共享。提供程序可以通过共享对象共享 Snowflake 数据库表、安全视图或安全的用户定义函数（UDF）。需要注意的是，共享不是从对象创建的，而是被授予对数据库、架构、表、视图或 UDF 的特权。

创建出站共享

当您创建出站共享时，您的账户是 Snowflake 提供商账户，而您要与之共享数据的账户是 Snowflake 使用者账户。请务必知道，共享只能包含单个数据库。



要在共享中包含多个数据库，您可以创建可共享的安全视图。如果共享中引用的对象驻留在同一个 Snowflake 帐户中，这是可能的。

要创建共享对象，请单击屏幕右上角的 Share Data 按钮，然后单击 Select Data 按钮。请务必选择 DEMO10_DB 数据库、PUBLIC 架构和 SHARINGDATA 表。请务必选中表旁边的框，以便选择要包含在共享中的表（如图 10-4 所示）。完成后，单击 Done（完成）。



图 10-4 在创建共享对象时选择数据

您将看到 Secure Share Identifier 字段现在已填充（如图 10-5 所示），但您可以更改标识符。请务必注意，安全共享标识 fier 对于创建共享的帐户必须是唯一的，并且标识符区分大小写。另请注意，您可以通过按名称输入您所在区域中的账户来与特定使用者共享对象。

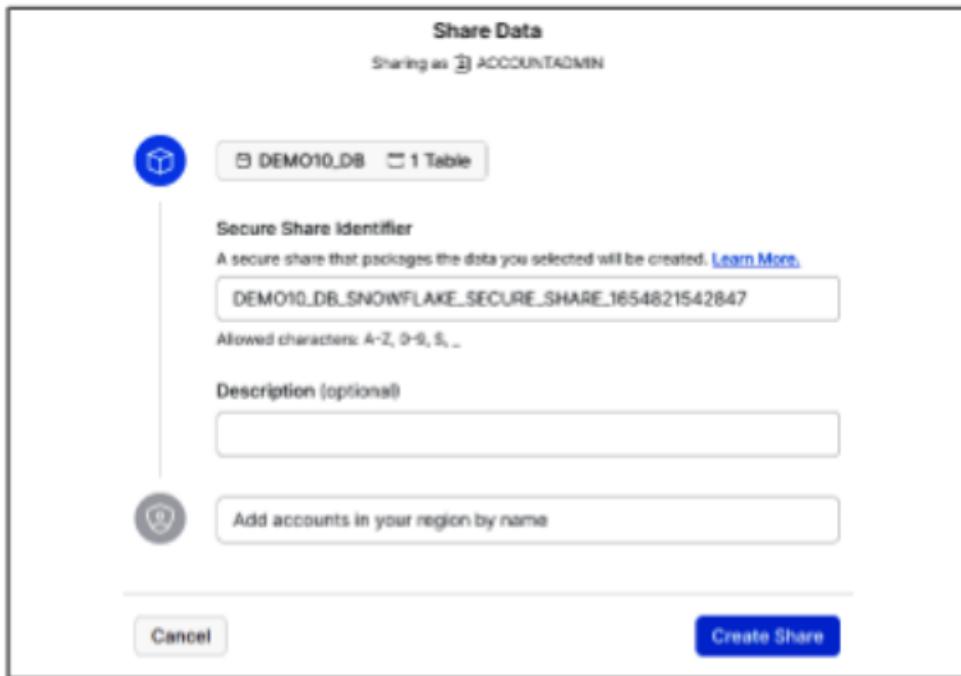


图 10-5. 命名安全共享并将使用者添加到共享对象

让我们更改自动生成的标识符，此时让我们跳过添加新的使用者。在 Secure Share Identifier 字段中，输入 并单击 Create Share 按钮。您将收到共享已创建的确认（如图 10-6 所示）。

The screenshot shows the Snowflake web interface for managing shares. At the top, the title bar displays 'DEMO10_SHARE'. Below it, a header row includes account information ('ACCOUNTADMIN'), share type ('Secure Share'), creation time ('Created: just now'), and location ('Azure - South Central US').

The main content area is divided into three sections:

- Shared With**: A section for adding consumers, featuring a button labeled 'Add Consumers'.
- Description**: A section for adding a summary, featuring a button labeled 'Add'.
- Data**: A section listing the shared data, featuring a button labeled 'Edit'. It shows one table named 'SHARINGDATA' from the 'DEMO10_DB' database, categorized under 'Table' and 'PUBLIC' schema.

图 10-6. 共享对象已成功创建

创建出站共享的另一种方法是在工作表中使用 SQL 命令。导航回 Data Sharing 工作表并执行以下语句：

```
使用角色 ACCOUNTADMIN;
CREATE OR REPLACE SHARED DEMO10_SHARE2; 授予对数据库 DEMO10_DB 的使用权以共享 DEMO10_SHARE2; 授予对
SCHEMA DEMO10_DB 的使用权。PUBLIC 用于共享 DEMO10_SHARE2; 授予 SELECT ON 表 DEMO10_DB。
公共。SHARINGDATA 用于共享 DEMO10_SHARE2;
```

您将注意到的一件事是，您正在为数据使用者将使用的共享分配访问权限。这应该看起来很熟悉。您为数据共享授予对对象的访问权限的方式是在第 5 章中共同介绍的基于角色的访问控制示例的一部分。

现在，如果您返回菜单选项 数据 → 私有共享 → 由我的帐户共享，您将看到我们刚刚创建的两个共享（如图 10-7 所示）。如果两个共享都没有立即显示，请尝试刷新页面。

The screenshot shows a table with three columns: 'TITLE', 'SHARED WITH', and 'CREATED'. There are two rows:

TITLE	SHARED WITH	CREATED
DEMO10_SHARE2	—	Just now
DEMO10_SHARE	—	1 minute ago

图 10-7. 账户中两个新创建的共享的列表

数据提供者在创建和管理共享中的角色

Snowflake 数据提供商创建出站共享对象，并使用其账户中存储的数据分配对一个或多个数据库表、安全视图或安全 UDF 的权限。因此，数据提供商负责承担数据存储成本。使用者无需支付任何数据存储费用，因为数据共享和通过共享访问的对象不会复制或移动到使用者的账户。



Snowflake 提供商可以与几乎无限数量的账户共享出站共享，并且在同一云提供商区域内的账户之间共享数据的行为永远不会产生任何费用。Snowflake 数据提供商还可以通过合同设置条款，以限制数据的重新共享。

如果您已准备好让 Consumer 访问 DEMO10_SHARE，则可以单击共享列表，然后单击右侧的 Add Consumers 按钮（如图 10-8 所示）。

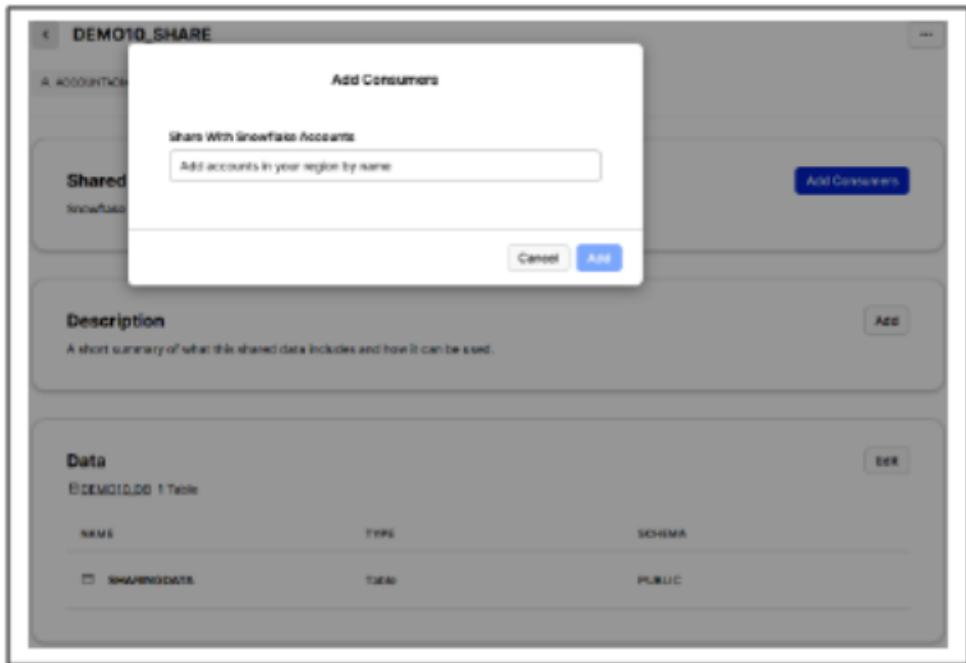


图 10-8. 将数据使用者添加到现有共享

我们不会在 UI 中添加任何使用者，因此请继续点击 Cancel 按钮。除了使用 UI 之外，Snowflake 提供程序还可以使用工作表中的 SQL 命令来添加一个或多个拥有自己的 Snowflake 帐户的使用者。该命令将如下所示：

```
ALTER SHARE <name_of_share> ADD ACCOUNTS = <name_of_consumer_account>;
```



您可以通过用逗号分隔名称来添加多个使用者账户。

当然，请记住，您可以通过完全排除数据或屏蔽可以看到的数据来限制某些数据的可访问性。一些测试是动态数据掩码和行级安全性，我们在第 7 章中讨论过。

如果您想与不同的使用者共享数据共享中的特定记录，该怎么办？实现该目标的一种方法是分离这些记录的存储，然后创建许多不同的数据共享，但这非常耗时且容易出错。幸运的是，有一种更简单、更好的方法。提供商可以维护一个数据共享，并使用 Snowflake 的 Current Account 功能向 Current Account 提供对自定义数据切片的访问权限。还可以向当前账户外部的每个使用者账户提供特定的数据切片。

让我们看看它的实际效果。在此示例中，我们将使用不同的国家/地区提供数据，并将这些国家/地区关联到特定区域。我们希望不同的账户可以访问这些区域数据。我们需要创建表并在 Snowflake 实例中插入该基表的值。我们将 ID 指定为区域。导航返回 Data Sharing 工作表并执行以下语句：

```
使用角色 ACCOUNTADMIN; 使用数据库
DEMO10_DB; 创建或替换 SCHEMA PRIVATE;

创建或替换表 DEMO10_DB。私人。SENSITIVE_DATA
(国家字符串,
 价格浮动,
 大小 int,
 id 字符串);

INSERT INTO DEMO10_DB。私人。SENSITIVE_DATA
values ('USA', 123.5, 10, 'REGION1'),
       ('美国', 89.2, 14,
        'REGION1'), ('加拿大', 99.0,
        35, 'REGION2'), ('加拿大',
        58.0, 22, 'REGION2'), ('墨西哥',
        112.6, 18, 'REGION2'),
       ('墨西哥', 144.2, 15,
        'REGION2'), ('爱尔兰', 96.8,
```

接下来，我们将创建一个表，该表将保存到各个账户的映射：

```
创建或替换表 DEMO10_DB。私人。SHARING_ACCESS
(id 字符串, snowflake_account 字符串);
```

让我们通过将这些详细信息插入到 SHARING_ACCESS 映射表中，为我们的 current account 提供对 REGION1 中数据（包括 USA 数据）的访问权限：

```
INSERT INTO SHARING_ACCESS values ('REGION1', current_account());
```

现在，我们将分配 REGION2 并REGION3与其各自账户关联的值，然后查看 SHARING_ACCESS 表的内容：

```
INSERT INTO SHARING_ACCESS values ('REGION2',
'ACCT2') ; INSERT INTO SHARING_ACCESS values ('REGION3',
'ACCT3') ; SELECT * FROM SHARING_ACCESS;
```

您会注意到，在图 10-9 中，我们将每个区域与一个 Snowflake 账户相关联。您可以在结果中看到我的 Snowflake 试用账户标识符为 VL35342；您的 Snowflake 账户不同，并且对您的组织是唯一的。

ID	SNOWFLAKE_ACCOUNT
1	REGION1
2	REGION2
3	REGION3

图 10-9. SHARING_ACCESS 表的结果

在我们的示例中，我们使用了 ACCT2 和 ACCT3，但如果要在生产 Snowflake 组织中创建此 SHARING_ACCESS 表，则需要将实际的 Snowflake 帐户标识符替换为 ACCT2 和 ACCT3。

我们的下一步操作是创建一个安全视图，我们将在其中将 SENSITIVE_DATA 基表中的所有数据与 SHARING_ACCESS 映射表联接起来：

```
CREATE OR REPLACE SECURE VIEW DEMO10_DB.公共.PAID_SENSITIVE_DATA AS
SELECT *
FROM DEMO10_DB.公共.SENSITIVE_DATA sd
JOIN DEMO10_DB.私人.SHARING_ACCESS sa
ON sd.id = sa.id AND
sa.snowflake_account = current_account();
```

我们希望为所有角色授予安全视图的权限，我们可以通过将此权限授予 PUBLIC 角色来实现：

```
GRANT SELECT ON DEMO10_DB.公共.PAID_SENSITIVE_DATA TO PUBLIC;
```

让我们看一下 SENSITIVE_DATA 基表。请记住，此表包含所有 8 条记录：

```
SELECT * FROM DEMO10_DB.私人.SENSITIVE_DATA;
```

正如预期的那样，图 10-10 显示我们能够查看所有 8 条记录。

	NATION	PRICE	SIZE	ID	...
1	USA	123.5	10	REGION1	
2	USA	89.2	14	REGION1	
3	CAN	99	35	REGION2	
4	CAN	58	22	REGION2	
5	MEX	112.6	18	REGION2	
6	MEX	144.2	15	REGION2	
7	IRE	96.8	22	REGION3	
8	IRE	107.4	19	REGION3	

图 10-10. SENSITIVE_DATA 表中的所有记录

现在让我们看看我们能够在安全视图中看到哪些记录：

```
SELECT * FROM DEMO10_DB.公共.PAID_SENITITVE_DATA;
```

我们当前的 Snowflake 账户映射到 REGION1，其中包括美国国家/地区。正如预期的那样，我们看到了两条 USA 记录（如图 10-11 所示）。

	NATION	PRICE	SIZE	...
1	USA	123.5	10	
2	USA	89.2	14	

图 10-11. 我们的帐户可以访问PAID_SENITITVE_DATA安全视图中的记录

让我们使用会话变量来模拟 Snowflake 帐户为 ACCT2，并看看我们能够在安全视图中看到的内容：

```
更改会话集 simulated_data_sharing_consumer='ACCT2';SELECT * FROM  
DEMO10_DB.公共.PAID_SENITITVE_DATA;
```

作为 ACCT2，我们可以看到所有 REGION2 数据，其中包括加拿大和墨西哥国家（如图 10-12 所示）。

	NATION	PRICE	---	SIZE
1	CAN	99		35
2	CAN	58		22
3	MEX	112.6		18
4	MEX	144.2		15

图 10-12. ACCT2 可访问的 PAID_SENSITIVE_DATA 安全视图中的记录

最后，让我们看看如果我们将 Snowflake 账户设置为 ACCT3，我们会看到什么：

```
更改会话集 simulated_data_sharing_consumer='ACCT3';SELECT * FROM
DEMO10_DB.公共.PAID_SENSITIVE_DATA;
```

如图 10-13 所示，我们可以看到爱尔兰国家，它是 REGION3 的一部分。

	NATION	---	PRICE	SIZE
1	IRE		96.8	22
2	IRE		107.4	19

图 10-13. PAID_SENSITIVE_DATA 安全视图中的记录可供 ACCT3 访问

我们想将 Snowflake 账户恢复到原始账户：

```
ALTER SESSION UNSET simulated_data_sharing_consumer;
```

现在我们已经成功测试了如何与不同账户共享数据，我们准备创建一个新的共享：

使用角色 ACCOUNTADMIN：使用数据库 DEMO10_DB；使
用 SCHEMA DEMO10_DB. 公共 创建或替换共享
NATIONS_SHARED；显示股票：

您会注意到，此新共享将包含在出站共享列表中。但我们还没有结束。我们需要向新共
享授予权限：

授予对数据库 DEMO10_DB 的使用权以共享 NATIONS_SHARED；授予对 SCHEMA DEMO10_DB 的使用权。
PUBLIC 用于共享 NATIONS_SHARED；在 DEMO10_DB 上授予 SELECT。公共。PAID_SENSITIVE_DATA 分享
NATIONS_SHARED。

您可以使用该语句确认共享的内容：

```
显示 GRANT 以共享 NATIONS_SHARED；
```

同样，您会注意到 grantee_name 包含您当前的帐户信息（如图 10-14 所示）。

privilege	granted_on	name	granted_to	grantee_name	grant_option —	granted_by
USAGE	DATABASE	DEM010.DB	SHARE	V135142.NATIONS.SHARED	false	ACCOUNTOWNER
USAGE	SCHEMA	DEM010_DB.PUBLIC	SHARE	V135142.NATIONS.SHARED	false	ACCOUNTOWNER
SELECT	VIEW	DEM010.DB.PUBLIC.HOLDSENITIIVE_DATA	SHARE	V135142.NATIONS.SHARED	false	ACCOUNTOWNER

图 10-14 授予 NATIONS_SHARED 共享的权限

现在，从提供商的角度来看，剩下要做的就是将账户添加到共享中：

这是伪代码：您需要替换为实际的 Snowflake 帐户
ALTER SHARE NATIONS_SHARED ADD
ACCOUNTS = ACCT2, ACCT3;

如果要检索具有从共享创建数据库的所有 Snowflake 使用者帐户的列表，可以使用以下命令：

显示股份授予 NATIONS_SHARED:

重要的是要记住，提供商账户支付共享数据的数据存储成本，而消费者账户支付查询数据的虚拟仓库成本。这假定数据提供商和数据客户都有自己的 Snowflake 帐户。

可能会出现数据提供商希望与当前没有 Snowflake 帐户的消费者共享数据的情况。在这种情况下，提供商可以建立和管理 Snowflake 读取器账户。当提供商为使用者建立读取器账户时，读取器账户将为使用者提供只读功能。

设置读取者帐户

在上一节中，我们导航到 Data 主菜单的 Private Sharing 部分，然后单击 Shared By My Account。然后，我们单击要为其添加使用者的特定共享。相反，如果我们想要创建读者帐户，请单击屏幕顶部的 Reader Accounts 部分，然后单击 + New 按钮。然后，我们将得到一个对话框页面，我们可以在其中创建一个读者帐户（如图 10-15 所示）。

对话框页面要求您提供读者帐户名称，以及将被授予管理员权限的人员的用户名和密码。



创建的读者帐户将与您自己的 Snowflake 帐户具有相同的版本、区域和云，并且您将使用管理员登录信息来管理读者帐户。

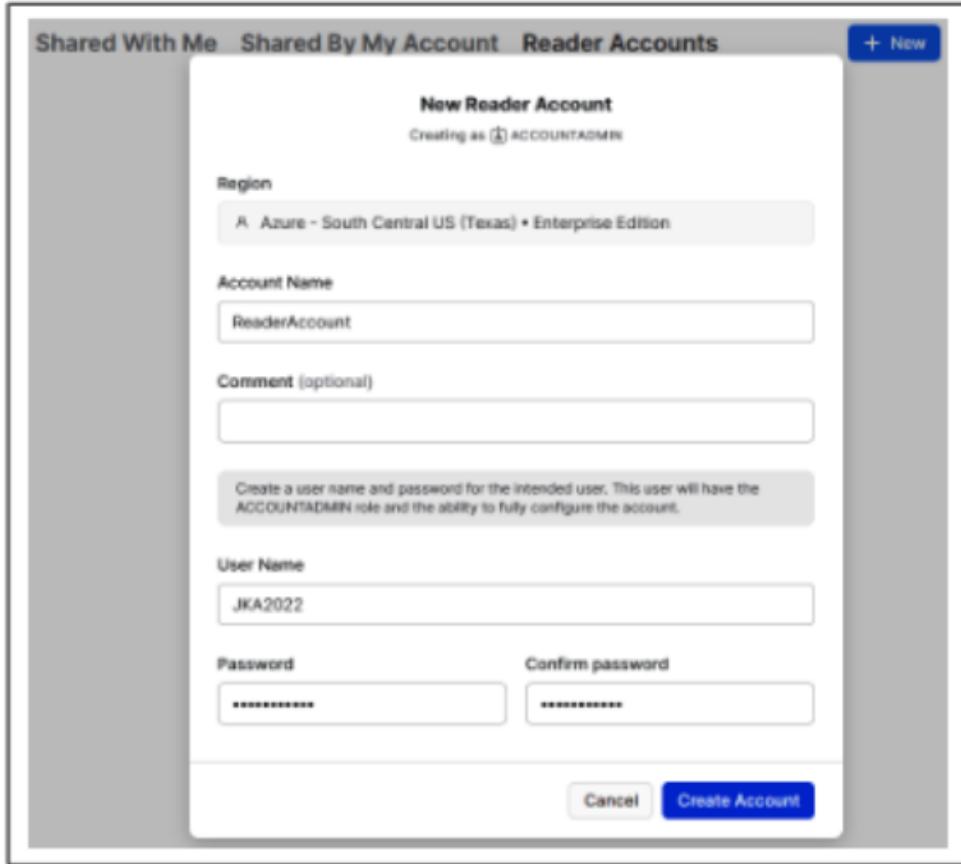


图 10-15. 创建新的读取者帐户

创建读取器帐户后，单击页面右侧的省略号，将出现“删除读取器帐户”的选项（如图 10-16 所示）。

Shared With Me Shared By My Account Reader Accounts						
Search		C				
1 Reader Account ⓘ						
Name	Cloud	Region	Locator	Locator URL	Created	Actions
READERACCOUNT Pending	Azure	South Central US (Texas)	OP3983	...	Just now	...

图 10-16. 删除读者帐户

单击 Drop 按钮后，您会注意到读者帐户的定位符 URL 会将 URL 复制到剪贴板。然后，您可以使用该 URL 和您创建的登录信息登录到读取器帐户。

以数据提供程序管理员身份登录到读取器帐户后，您需要从入站共享创建数据库，并设置一个数据使用者用户帐户，以分配给您希望有权访问 Snowflake 读取器帐户的人员。



作为数据提供商，您需要承担查询数据时产生的计算费用，因此您可能需要考虑设置资源监视器以限制总成本或在消耗指定数量的积分时收到通知。

Snowflake 数据使用者如何使用入站共享

使用 ACCOUNTADMIN 角色，作为 Snowflake 帐户所有者的数据使用者可以从提供者的出站共享（现在是使用者的入站共享）创建数据库。

在 Snowflake 工作表中，以下是使用者用于创建共享数据库的语句。请注意，要执行此语句，您需要包含数据库名称和入站共享名称：

从共享创建数据库：

如前所述，无法编辑共享数据。共享数据库是只读的；因此，数据无法更新，数据使用者也无法在数据库中创建新对象。这个新创建的数据库还有一些其他限制。共享数据库的一个独特属性是无法编辑组件。

数据使用者无法克隆共享数据库或其中的对象。无论如何，可以将共享数据复制到新表中。虽然从技术上讲，可以复制共享数据库，但如果双方签订此类协议，则可能违反合同条款。对于作为数据库所在 Snowflake 账户所有者的数据使用者，可以查询数据的次数没有限制。但是，请务必记住，数据使用者承担查询数据的计算成本。

了解使用者账户：读取者账户与完整账户

读取者账户和完整账户都是 Snowflake 数据共享中使用的使用者账户类型。Reader 账户无需支付查询数据的任何计算费用。这些费用由创建 reader 账户的提供商账户支付。因此，提供商可以选择使用资源监控器来限制读取器账户的查询。读者帐户可以作为合作伙伴的中间步骤。

尚未准备好从当前的数据流程过渡。设置和维护读者账户供其合作伙伴使用确实需要提供商付出额外的努力。



完整使用者账户还可以直接在 Snowflake 中将其数据与共享数据连接，而读取者账户只能在 Snowflake 中查看与他们共享的数据。

ACCOUNT_USAGE 共享与所有其他入站共享有何不同

ACCOUNT_USAGE 共享是一种入站共享，在第 3 章中引入，当时我们将 SNOWFLAKE 数据库中的 ACCOUNT_USAGE 共享与 INFORMATION_SCHEMA（为每个数据库提供的架构）进行比较。默认情况下，SNOWFLAKE 数据库只能由 ACCOUNTADMIN 查看，它与 INFORMATION_SCHEMA 数据库类似，因为它们都提供有关账户的对象元数据和使用情况指标的信息。但是，两者之间有三个主要区别：

- 具有多个视图的 ACCOUNT_USAGE 共享包括放置对象的记录。
- ACCOUNT_USAGE 份额的保留时间也比 INFORMATION_SCHEMA 长。
- ACCOUNT_USAGE 共享的平均延迟约为 2 小时，而查询 INFORMATION_SCHEMA 时没有延迟。

共享权限是导入的，大多数入站共享都允许您创建和重命名数据库，以及删除从中创建入站共享的数据库。但是，ACCOUNT_USAGE 份额是不同的。您无法创建、重命名或删除与 ACCOUNT_USAGE 共享关联的数据库。您也无法添加注释。ACCOUNT_USAGE 共享的管理方式不同，因为这是 Snowflake 就您的账户与您进行沟通的一种方式。

在本章中，当我们提到入站共享时，我们指的是除 ACCOUNT_USAGE 共享之外的所有入站共享，这是唯一的。与 ACCOUNT_USAGE 共享不同，可以删除入站共享，也可以使用与入站共享名称不同的名称创建新数据库。

入站共享上的数据库与常规数据库之间的比较

正如我们所看到的，共享权限是导入的，而共享数据是只读的。如表 10-1 所示，入站共享允许您为入站共享创建和删除数据库，但不能更改数据库的结构。

表 10-1 入站共享数据库和常规数据库之间的差异

CREATE、RENAME DROP 数据库	现有数据库中的 CREATE、 ALTER 或 DROP 架构、表、 阶段和/或视图	现有数据库中的 GRANT/REVOKE ALL、 MONITOR
入站共享 是 否	否	是
常规数据库 是	是	是

^a ACCOUNT_USAGE 份额除外

需要注意的是，与大多数传统数据库不同，Snowflake 支持跨数据库联接，包括与从入站共享构建的数据库的联接（请参见图 10-17）。除了具有读取者帐户的使用者之外，使用者还可以在自己的数据库中创建视图，该视图可以将自己的数据与共享数据库中的数据组合在一起。这是非常有益的，因为消费者可以丰富自己的数据，使其更有价值。

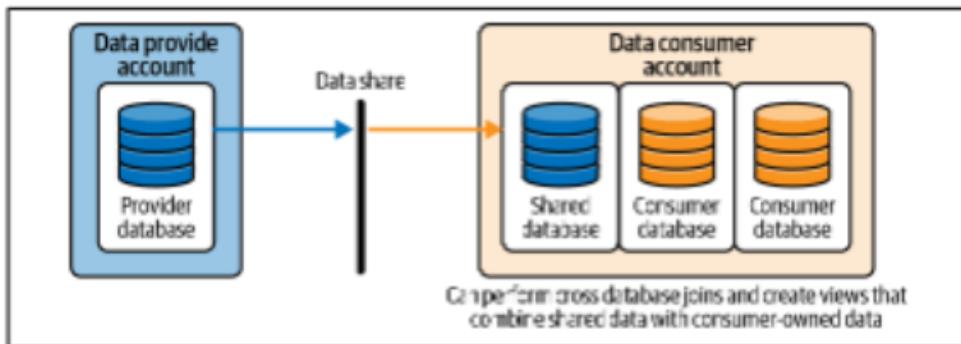


图 10-17. 共享数据可以与消费者拥有的数据结合使用

当在提供商的 Snowflake 账户中创建新记录作为共享数据库的一部分时，这些记录几乎可以立即在数据使用者的入站共享数据库中使用。但是，当提供程序在共享数据库中创建新对象时，这些对象不会自动与数据使用者共享。提供者必须先向数据共享授予授权，然后使用者才能查看这些对象中的记录。

如何在公共 Snowflake Marketplace 上发布商品和购物

数据提供商可以通过 Snowflake Marketplace 与全球数据消费者公开连接。作为第三方数据的潜在使用者，您可以直接从 Snowflake 用户界面访问 Marketplace。

首先，确保您已将角色设置为 ACCOUNTADMIN，然后单击 市场 选项，如图 10-18 所示。如果您的角色当前未设置为 ACCOUNTADMIN，请务必单击您的姓名旁边的下拉箭头并更改您的角色。

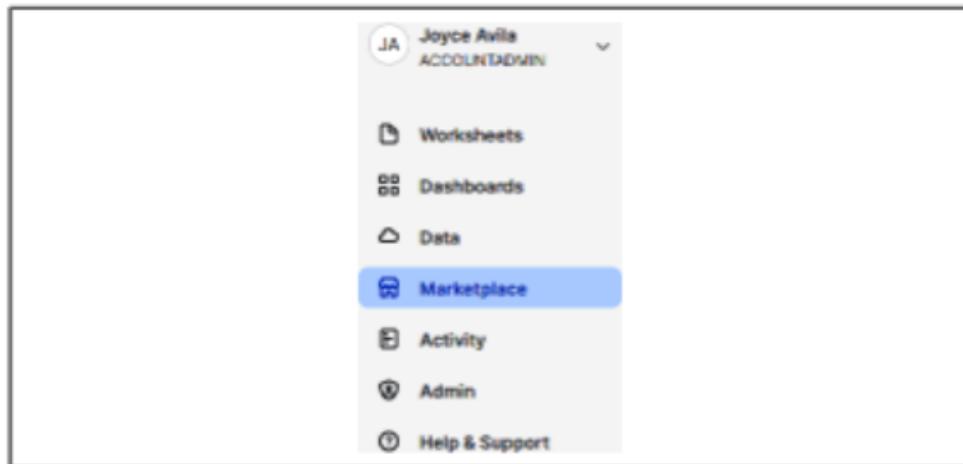


图 10-18. 直接从用户界面访问 Marketplace

现在，您可以在 Snowflake UI 中搜索 Marketplace（参见图 10-19）。当您搜索 Marketplace 时，您将找到可供您立即访问的标准列表，以及您可以发送访问请求的个性化列表。

The screenshot shows the Snowflake Marketplace homepage. At the top, there is a search bar labeled "Search Snowflake Marketplace", followed by navigation links for "Categories", "Business Needs", "Providers", and "My Requests". Below the header, there is a section titled "Featured Providers" featuring icons for iPatio, Heap, and Cloctrack. The main content area is divided into several sections: "Most Recent" (with four provider cards: GoldGraph, Hubu, Hubu, and Hubu), "Financial" (with four provider cards: DataHub, Tema AI, Prepper Inc., and Valuus Alpha), and other sections like "Data Quality", "Machine Learning", and "Analytics". Each provider card includes a thumbnail, the provider's name, a brief description, and a "Personalized" button.

图 10-19. Snowflake 市场

一些第三方数据提供商免费提供数据访问，有时也会向您收取访问其数据的费用。他们可能还需要签订合同以确保您不会重新共享他们的数据。无论访问数据是免费的还是提供商向您收取费用，您与数据提供商之间的安排都与您与 Snowflake 的安排是分开的。Snowflake 仅针对您用于查询共享数据的计算向您收费。数据存储费用由数据提供商负责。

面向提供商的 Snowflake Marketplace

对于数据提供商来说，Snowflake Marketplace 提供了一个以前所未有的方式将数据货币化的机会。您可以使用自己的自定义业务模型提供安全、个性化的数据视图。Snowflake 不是数据提供商和数据使用者之间交易的一方，在 Data Cloud 上共享数据的行为不会产生任何费用。相反，Snowflake 在其平台上支持数据共享，并在数据使用者使用虚拟仓库查询数据时赚取收入。

要成为获得批准的 Snowflake 数据提供商，您需要提交请求，签署 Snowflake 提供商协议，并同意在 Snowflake 提供商策略范围内运营。重要的是，您的数据必须满足某些要求。

一个要求是您的数据必须是最新的和非静态的，这意味着您将提供的数据必须是相对较新的，并且您有计划继续更新数据。

另一个要求是数据不能是样本、模拟数据。需要明确的是，可以提供示例数据来帮助消费者确定他们是否要使用完整的数据集。但是，完整数据集本身不能只是样本数据或模拟数据。

最后，您必须拥有使数据可用并确保数据不包含敏感私人信息的权利。对于每个列表，表、列和共享名称的对象标识必须全部使用大写字母书写，并且只能使用字母数字字符。您还需要为每个列表提供至少一个使用示例。ZoomInfo 的使用示例如图 10-20 所示。

ZoomInfo SQL Query
Select all companies and their core firmographics + advanced attributes including employee range, revenue range, primary industry

```
1   SELECT "ZoomInfo Company ID",
2   "Company Name",
3   "Website",
4   "Company HQ Phone",
5   "Fax",
6   "Ticker",
7   "Revenue (in $000s)",
8   "Revenue Range",
9   "Employees",
10  "Employee Range",
11  "SIC Code 1",
12  "SIC Code 2",
13  "SIC Codes",
14  "NAICS Code 1",
15  "NAICS Code 2",
16  "NAICS Codes",
17  "Primary Industry",
18  "Primary Sub-Industry",
19  "All Industries",
20  "All Sub-Industries",
21  "Industry Hierarchical Category",
22  "Secondary Industry Hierarchical Category",
23  "ZoomInfo Company Profile URL",
24  "Ownership Type",
25  "Business Model",
26  "Certified Active Company",
27  "Company Street Address",
28  "Company City",
29  "Company State",
30  "Company Zip Code",
31  "Company Country",
32  "Company Is Acquired",
33  "Company ID (Ultimate Parent)",
34  "Entity Name (Ultimate Parent)",
35  "Company ID (Immediate Parent)",
36  "Entity Name (Immediate Parent)",
37  "Relationship (Immediate Parent)"
38  FROM "PROD_MASTER_DB"."COMPANY","COMPANY_DATABRICK_TABLE"
```

Show less ^

图 10-20. Snowflake Marketplace 的 ZoomInfo 使用示例

Provider 工作室

Provider Studio 提供了执行所需操作以获得批准列表和查看来自消费者的数据请求的功能。此外，Provider Studio 还使 Snowflake 提供商能够通过选择 Analytics 选项卡来查看与 Marketplace 列表相关的关键分析。只有 Snowflake 数据提供商才能访问 Provider Studio。

标准数据列表与个性化数据列表

所有 Snowflake Marketplace 列表都包含有关共享数据的详细信息以及示例查询和使用示例。列表还包括有关数据提供商的信息。标准和个性化两种类型的列表之间的区别在于共享数据是否可以立即访问。

Snowflake Marketplace 标准列表提供对已发布数据集的即时访问，最适合用于通用、聚合或非客户特定的数据。个性化数据列表是必须发出访问数据集的请求的列表。发出请求后，数据提供商将收到通知，然后联系使用者。

标准 listing 的示例如图 10-21 所示。

The screenshot shows a listing for 'Free Transaction Data Sample' by SafeGraph. The page is divided into two main sections: a left sidebar and a right panel.

Left Sidebar (Product Details):

- Icon:** A blue square icon with a white graph symbol.
- Title:** Free Transaction Data Sample
- Category:** SafeGraph - Financial - Never Static Data
- Description:** This free dataset includes spending behavior data at individual POs in the US based on aggregated debit/credit card transactions. Dataset contains sample of major brands and chains across the US.
- Sampled Tables Included:**
 - Location name
 - Address
 - Category
 - Total spend
 - Number of Transactions
 - Number of unique customers
 - Median spend per transaction
- Additional Information:** All SafeGraph POI-based datasets utilize Placekey as the primary key and are formatted as delimited CSVs. SafeGraph updates the Spend dataset every month with the past month's openings and closings and maintains a persistent Placekey across releases. Our detailed SafeGraph Spend Schema[2] is available online. Additionally please refer to Places Data Manual[3] for more detailed field definitions and methodologies. Our Places Summary Statistics[4] is updated with every monthly release to reflect data coverage.
- Data Dictionary:** Our documentation site includes detailed information for all of our products, but this product listing specifically only includes the Spend data set and does not include Places, Geometry, or Patterns data.
 - (1) Spend Schema: <https://docs.safegraph.com/docs/spend>
 - (2) Places Data Manual: <https://docs.safegraph.com/docs/places-data-manual>
 - (3) Places Summary Statistics: <https://docs.safegraph.com/docs/places-summary-statistics>
- Show Less ▾**

Right Panel (Actions and Information):

- Free** (Subscription level)
- Unlimited queries**
- Get Data** (Large blue button)
- SafeGraph** (Logo)
- SafeGraph is a global geospatial company that offers any data in any place in the world. Customers like Dow, Fiserv, Mapbox, and Syntex use SafeGraph data to better understand their...**
- Show More ↴**
- Contact**
- Documentation**
- Support**
- Terms of Service**

图 10-21. Snowflake Marketplace 中的常设列表示例

您可以从几个方面看出图 10-21 中的列表是一个标准列表。首先，列表的主页上没有包含个性化一词。此外，当您单击列表时，您将看到 Get Data 按钮，单击该按钮将显示有关列表的详细信息，以及您需要单击的最后一个按钮，即 Get Data 按钮（如图 10-22 所示）。请注意，除了 ACCOUNTADMIN 角色之外，您还可以添加更多角色，以授予更多角色对共享数据库的访问权限。

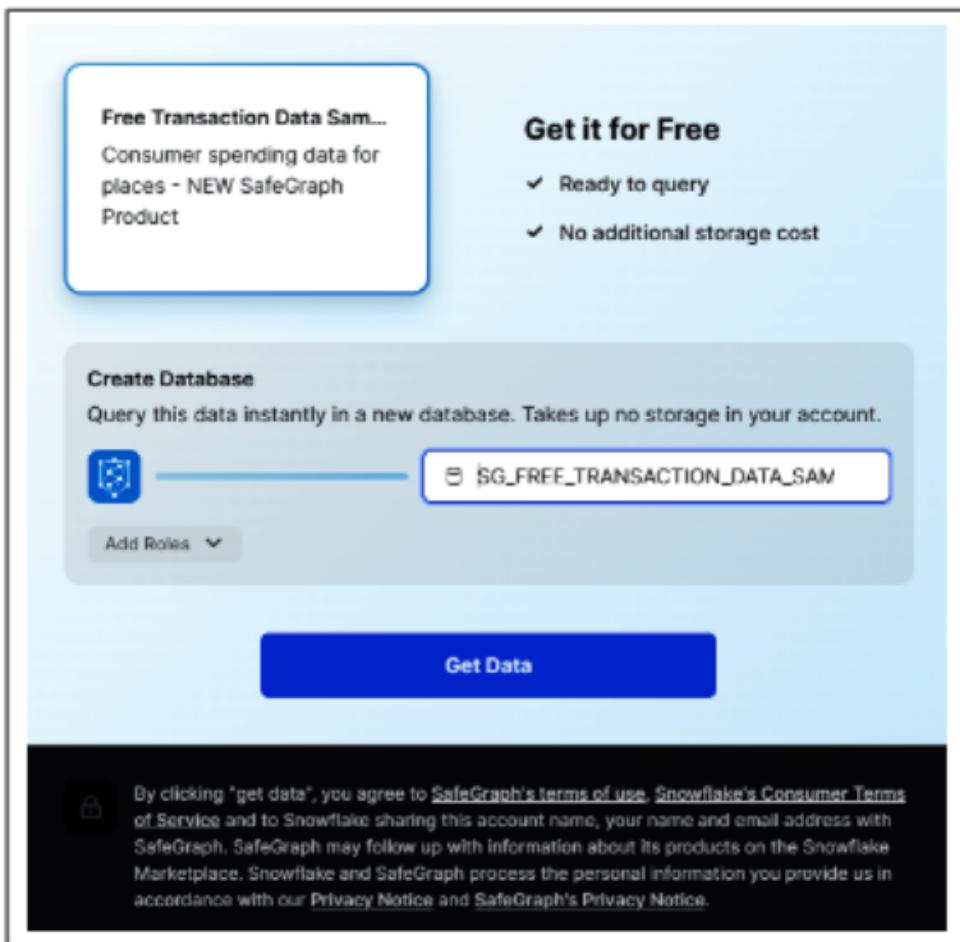


图 10-22 在从 Marketplace 标准列表中获取数据之前添加角色

单击 Get Data 按钮时，您将看到数据现在已准备好进行查询（如图 10-23 所示）。

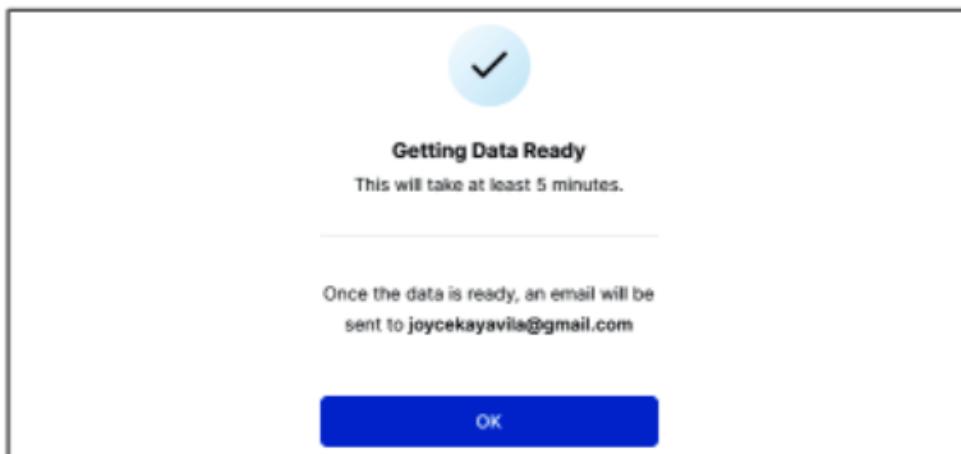


图 10-23 确认 Marketplace 标准产品信息中的数据已准备好查询

图 10-24 显示了 Knoema 在 Snowflake Marketplace 中的个性化列表示例。您可以单击 Request Data (请求数据) 按钮以向 Knoema 请求访问权限。

A screenshot of a product page for the "Global Labor Data Pack" by Knoema in the Snowflake Marketplace. The page has a white background with a left sidebar and a main content area. The sidebar on the left contains the Knoema logo, the title "Global Labor Data Pack", and a brief description: "The Knoema Global Labor Data Pack provides access to labor data for enterprises seeking to streamline the ingestion, maintenance, and delivery of public labor related indicators. It includes 343M time series on labor force participation, migration, unemployment, environment by industry, occupation and education attainment and other labor related indicators." Below this is another paragraph: "All the global labor data is ingested and curated by Knoema. The Global Labor Data for Nations States Data Pack comes from public sources, including international organizations, national sources and research institutions." A list of major topics follows: "The list of time series is organized into five major topics that enable quick and easy access to the indicator of interest:

- Labor Force and Migration
- Employment
- Productivity
- Unemployment and labor statistics
- Institutional Environment

Datasets include:

- U.S. Weekly Unemployment Insurance Data
- American Community Survey & their integrated economic characteristics
- Youth employment-to-population ratio
- Employment distribution by economic activity
- Population estimates and projections
- Proportion of citizens engaged in economic activity
- 1700+ others (please see reference table DATASETS for the list of all the datasets included in this listing, including key details on each dataset such as name, source, frequency of refresh, documentation on data, etc.)

" To the right of the sidebar is a large "Personalized Data" card with a blue header. It contains the text "Available on request" and a large blue "Request Data" button. Below this is another Knoema card with the Knoema logo, a brief description: "Knoema is the most comprehensive source of global data for making data in the world. Our data technology solutions encompass the complete data lifecycle to insight workflow from raw... More >" and three buttons: "Contact", "Documentation", and "Support".

图 10-24 Knoema 在 Snowflake Marketplace 中的个性化列表

如今，有数百家提供商参与了 Snowflake Marketplace，并且有数百个列表，其中包括许多知名数据提供商。可以在 UI 中找到所有 Snowflake Marketplace 数据集的最新计数和详细信息。Snowflake 的网站上也介绍了 Snowflake Marketplace。

利用 Snowflake 私有数据交换的力量

Snowflake Private Data Exchange 使组织能够与一组选定的参与者私下共享数据集。通过这种方式，组织可以创建自己的 Snowflake Marketplace 的原始版本，但可以控制哪些数据可以显示以及谁可以访问列表。Snowflake Private Data Exchange 对于拥有不同部门的公司来说是一个不错的选择，每个部门都有自己的 Snowflake 帐户，并希望控制在整个组织中共享哪些数据。希望与其供应商共享数据的零售商也可以使用 Snowflake Private Data Exchange。

设置 Snowflake Private Data Exchange 是一个相当简单的过程，通常只需要一两个工作日。您需要导航到 Snowflake 支持页面并提交案例。您需要提供业务案例理由、新数据交换的唯一名称以及数据交换显示名称和您的帐户 URL。如果您使用的是组织 Snowflake 功能，您还需要提供组织 ID。

设置 Private Data Exchange 后，您可以通过选择 Data → Shared Data → Manage Exchanges 来访问它。然后，您将看到 Private Data Exchange。在此示例中，我创建了一个名为 SpringML 的 Private Data Exchange（如图 10-25 所示）。



图 10-25. 管理私有数据交换

单击 Private Data Exchange 的名称后，您将看到其详细信息并能够向其添加成员（如图 10-26 所示）。



图 10-26. 私有数据交换的详细信息

如果单击 Add Member 按钮，则可以选择添加新帐户，并允许该帐户作为使用者和/或提供者参与 Private Data Exchange（如图 10-27 所示）。



图 10-27. 向 Private Data Exchange 添加成员

Snowflake 数据交换是通过邀请合作伙伴、客户、供应商和其他人参与来扩展业务的好方法。

Snowflake Data Clean Rooms

Snowflake 上的数据净室是一种设计模式，用于以保护隐私的方式在两方或多边之间共享数据。数据净室的一个常见用例是链接来自多个参与者的匿名营销和广告数据。这为参与者提供了一种遵守隐私法的方法，例如 GDPR 和加州消费者隐私法案（CCPA），因为数据净室不允许可以追溯到特定用户的数据点离开环境。

传统的数据净室要求所有数据存储在一个物理位置。相比之下，Snowflake 上的数据净室不需要参与者将其数据从现有位置移动，以便在 Snowflake Data Exchange 中列出其数据。相反，参与者使用安全数据共享，其中 Clean Room 中包含的 PII 受到保护和加密。

使用 Snowflake 创建分布式数据净室功能的独特之处在于，Snowflake 使每个数据净室参与者都能够控制自己的数据，而不必信任拥有所有数据的单一方。除了

限制谁可以访问数据，Private Data Exchange 参与者可以使用安全功能和访问控制来保护他们的数据，同时仍然允许进行 Join Data Analy-SIS。

Snowflake 上的数据净室从设置私有数据交换开始，并使用当前的 Snowflake 功能，允许相关方将数据联接到一个查询中，从而产生聚合结果，而不允许访问底层的低级数据。安全功能和联接可用于建立指向人员、设备或 Cookie 的单个链接，而无需交换或显示任何 PII。

除了 Snowflake 安全数据共享技术（可在多个 Snowflake 账户之间自动安全地共享表）之外，还有其他技术功能可用于数据净室。行访问策略匹配客户数据而不暴露 PII，存储过程验证查询请求。Snowflake 流和任务可监控数据洁净室请求并自动响应。建立了准则来确定允许哪些数据进入以及哪些数据（如果有）可以从 Snowflake 上的分布式数据净室离开。

重要的设计、安全性和性能注意事项

Snowflake 股票对于提供商来说相对快速和容易地设置并可供消费者使用。但是，在创建新共享之前，最好考虑一下共享的设计、安全性和性能影响。

共享设计注意事项

当记录被添加到与使用者共享的提供者对象时，它们立即可供使用者使用。但是，如果将对象（如新表）添加到数据库中，则不会自动与使用者共享新表。

创建者必须采取措施来共享该新对象。

从技术角度来看，不同的共享资源可以使用相同的架构，但即使它们不是，也会显得好像所有对象都是共享资源的一部分。这使得将新对象添加到属于同一架构的现有共享资源变得特别困难。



不需要为每个列表提供单独的架构。但是，从设计角度来看，计划为每个列表创建单独的架构可以减少混淆和意外共享对象的可能性。

入站共享只能包含一个数据库，但生产者可以使用安全视图共享来自多个数据库的数据。Snowflake 提供程序可以创建一个安全视图，该视图引用多个数据库中的架构、表和其他视图，只要这些数据库属于同一帐户即可。

共享安全注意事项

共享安全对象的工作方式与共享表的方式相同。您可以轻松地设计一种安全地共享对象的方法，而不是共享表。一种选择是创建基表所在的内部可用架构，然后为安全对象创建外部可用的架构。这样，您就可以共享外部架构和安全对象，而无需公开基表。



可以共享以下 Snowflake 对象：表、外部表、安全视图、安全具体化视图和安全 UDF。但是，为了确保共享数据库中的敏感数据不会暴露给使用者账户中的用户，强烈建议您共享安全视图或安全 UDF，而不是直接共享表。

通过使用动态数据掩码或行访问策略，可以与不同的使用者账户共享基表中的一部分数据，而不必为每个使用者创建单独的视图。



也可以使用 CURRENT_ACCOUNT() 函数。请注意，它只能使用 CURRENT_ACCOUNT() 用于在使用安全视图和安全具体化视图时进行

共享性能注意事项

在共享来自大型表的数据时，建议您作为提供商在表上定义集群键。这将防止您的使用者在查询数据库共享时受到性能下降的负面影响。

数据库共享和数据库克隆之间的区别

通常，人们会问克隆数据库和使用 Snowflake 安全数据共享功能有什么区别。答案是，数据库克隆可以有效地拍摄数据库的快照，当原始数据库继续更改时，克隆的数据库不会。请务必注意，元数据不是实际创建数据的物理副本，而是用于显示

原始数据库。原始数据库和克隆的数据库都存在于同一个账户中。相反，数据共享发生在不同的账户之间，共享数据是数据的实时视图，每当原始数据库中发生更改时，该视图都会发生变化。

数据共享和时间旅行注意事项

Snowflake Time Travel 可以在指定时间点访问历史数据。出于安全原因，没有为数据共享启用 Time Travel。否则，使用者可能会回到他们可以访问现在不应允许访问的视图的时代。

共享数据共享

使用者无法将共享对象重新共享给其他账户。使用者也无法克隆共享对象。但是，使用者可以将数据复制到表中，以创建技术上可以共享的数据副本。不过，这种风险不仅限于数据共享。例如，通过 API 或 FTP 等传统方法与他人共享的任何数据都存在进一步共享的风险。确实，任何拥有手机并可以访问您的数据的人都可以复制和共享它。因此，任何限制（例如不允许与组织外部的任何人共享共享数据）最好使用合同协议来解决。

总结

在本章中，我们了解了 Snowflake 的架构如何支持安全数据共享，并考虑了特定的用例，以强调数据共享的重要性。我们深入研究了 Snowflake 安全数据共享的四种方法：直接安全数据共享、私有数据交换、公共 Snowflake 数据交换和数据净室。随着我们将注意力集中在学习如何创建出站共享和了解 Snowflake 数据使用者如何使用入站共享上，安全数据共享的提供者和使用者角色之间的区别变得更加明显。

拥有足够的数据，深入和广泛，以及拥有正确的数据类型，通常意味着我们需要整合外部或第三方数据。Snowflake 的安全数据共享功能使我们能够做到这一点。确保我们自己的数据安全并访问我们需要的共享数据是能够提取有价值的数据洞察的必要第一步。我们获得更好见解的一种重要方法是可视化数据。在下一章中，我们将获得一些实践经验来了解 Snowsight 的可视化功能，并且我们将了解提供 Snowflake 原生强大数据可视化工具的 Snowflake 合作伙伴。

提醒一下，我们在本章中都使用了 ACCOUNTADMIN 角色以实现模拟，但最佳实践是使用 SYSADMIN 角色创建新的 Snowflake 对象并根据需要向自定义角色授予权限。

代码清理

本章的代码清理所需要做的就是删除我们创建的数据库。但是，让我们看看当我们尝试这样做时会发生什么：

```
使用角色 ACCOUNTADMIN;  
DROP DATABASE DEMO10_DB;
```

我们收到一个错误，因为当有与数据库关联的活动共享时，我们无法删除数据库（如图 10-28 所示）。因此，如果我们想删除数据库，则需要撤销对数据库的共享访问权限。



图 10-28. 尝试删除具有活动共享的数据库时收到错误消息

让我们撤销权限，然后看看我们是否能够删除数据库：

```
从共享NATIONS_SHARED撤销对数据库DEMO10_DB的使用:撤销共享DEMO10_SHARE对  
数据库DEMO10_DB的使用:撤销共享DEMO10_SHARE2对数据库DEMO10_DB的使  
用:DROP DATABASE DEMO10_DB;
```

您会注意到，我们只需要在数据库级别撤销访问权限，因为一旦删除了该数据库访问权限，架构或表级别就不再有访问权限。此外，我们只需要删除数据库，这将导致数据库中的任何 schemas、tables 和 views 被删除。

知识检查

以下问题基于本章中包含的数据共享信息：

1. Snowflake 安全数据共享的四种方法是什么？
2. Snowflake 是否支持跨数据库联接？解释与安全数据共享相关的相关性。
3. 数据共享是否可以包含多个数据库？
4. Snowflake Marketplace 和 Snowflake Data Exchange 有什么区别？

5. 哪些角色可用于创建和维护数据共享？
6. 哪个 Snowflake 账户需要支付与数据共享相关的数据存储费用？
7. 谁来支付查询数据共享时产生的虚拟仓库计算费用？

8. 解释术语 Snowgrid。
9. 哪些 Snowflake 对象可用于创建数据共享？是否有任何与选择对象以创建数据共享相关的最佳实践？
10. 获得批准在 Snowflake Marketplace 上发布商品需要满足哪些要求？

这些问题的答案可在附录 A 中找到。

在 Snowsight 中可视化数据

数据可视化可以更轻松地识别趋势、模式和异常值，并在复杂数据中发现见解。在视觉上下文（例如图形或地图）中显示的信息是人类大脑理解大量数据的一种更自然的方式。Domo、Power BI、Sisense、Sigma、Tableau 和 ThoughtSpot 等商业智能（BI）工具以这种方式用于公司报告和构建复杂的仪表板，其中许多用户是特定仪表板的使用者。

有时，组织中的个人或小组用户只需要进行临时数据分析，在加载数据时执行数据验证，或者快速创建可以作为一个团队一起共享和探索的次要图表和仪表板。对于这些使用案例，Snowflake 的 Snowsight 可视化工具是不错的选择。Snowsight 可视化工具包括带有交互式结果的自动统计数据和带有图表图块的控制面板。

在深入研究 Snowsight 的可视化工具之前，我们需要讨论采样。这是一个考虑因素，因为我们经常处理的数据集太大，Snowsight 统计数据和可视化无法管理。Snowsight 在帮助识别初始数据加载的异常值和质量问题方面特别有用，但其功能仅适用于少量行。我们可以通过 clause 或 clause 将大型数据集缩减为较小的子集，以便使用 Snowsight 可视化工具。在本章中，我们将比较 and 子句。

准备工作

为了准备我们的第一个练习，让我们创建一个新文件夹。在右上角，单击省略号，然后单击“新建文件夹”（如图 11-1 所示）。

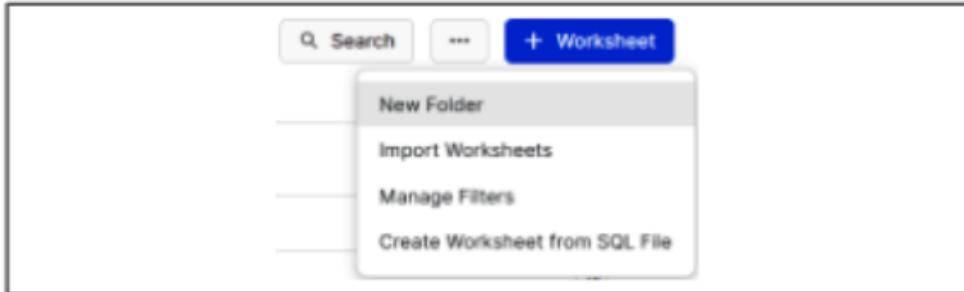


图 11-1. 创建新文件夹

将新文件夹命名为 Chapter11，然后单击 + 工作表 按钮创建一个新工作表。将工作表命名为 Visualization（可视化）（如图 11-2 所示）。

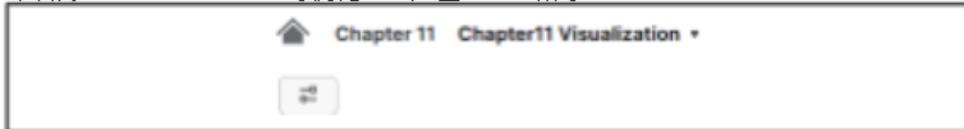


图 11-2. 使用 Chapter11 文件夹和 Visualization 工作表进行环境设置

我们将使用 Snowflake 的示例数据库；因此，本章不需要额外的准备工作。

Snowsight 中的数据采样

对数据进行采样是效率和可视化目的的重要主题。加载大型数据集后，负责加载数据的 ETL 开发人员或软件工程师可以扫描数据样本，以确保没有异常情况。同样，业务分析师可以将不熟悉数据的示例作为探索数据的第一步，而审计员可以获取用于执行分析的数据样本。



可用于 Snowsight 可视化的记录数量存在限制。对于自动统计可视化以及图表和仪表板，必须少于 1 百万个狮子行。因此，通常无法在 Snowsight 中可视化完整的数据集。在这些情况下，获取要可视化的数据样本非常重要。

通过采样，您可以创建一个控制面板，其中包含与完整数据集非常相似的有限数据子集。Sampling 返回指定表中的行子集。对 Snowflake 表进行采样返回的行数取决于您选择的两种不同方法中的哪一种。您可以选择根据您指定的确切数量返回行，也可以让 Snowflake 返回一定百分比的表行。

基于特定行数的固定大小采样

可以对查询中指定的固定行数进行采样。如果您希望返回数据样本而不是所有数据行，请确保指定要采样的行数小于表中的总行数。行数可以是介于 0 和 1,000,000 之间的任何整数。请注意，当您按行数进行采样时，将使用 Bernoulli 采样方法，在这种情况下，不支持 System/Block 和 Seed 方法。伯努利抽样是一个抽样过程，其中每一行都有相等的机会被选中。

基于概率的基于分数的抽样

您可以对表的一部分进行采样，并根据表的大小返回确切的行数。实现此目的的一种方法是指定种子，然后声明样本中包含特定行的指定概率。

在 Snowsight 中，可以使用 Bernoulli 或 System 采样方法。系统采样通常比伯努利采样更快，但样本可能会有偏差，尤其是对于小表格。如果未指定，则 Bernoulli 是默认采样方法。



伯努利抽样是等概率，无替换抽样设计。每行都有相等的被选中机会，因此，伯努利采样可以称为行采样。使用系统采样时，可能会选择特定的行块。系统采样有时称为块采样。对于大型表，这两种采样方法之间的差异可以忽略不计。

预览字段和数据

对于这些示例，我们需要将角色设置为 SYSADMIN，并使用 SNOWFLAKE_SAMPLE_DATA 数据库中的 schemas 之一：

```
使用角色 SYSADMIN;使用数据库  
SNOWFLAKE_SAMPLE_DATA;使用 SCHEMA  
TPCDS_SF100TCL.
```

在屏幕左侧，展开 TPCDS_SF100TCL 架构下的表（如图 11-3 所示）。



图 11-3. TPCDS_SF100TCL 架构中的表列表

接下来，将光标悬停在 `STORE_SALES` 表上，或查看底部的专用部分，您将看到一个预览，显示表中有超过 288 个 billion 行。除了查看 `STORE_SALES` 表中数据表的大致行数外，您还可以检查字段列表。数字字段以 123 开头，文本字段以 Aa 开头（如图 11-4 所示）。

The screenshot shows the Snowsight interface with the search bar at the top containing 'All Objects'. Below the search bar is a sidebar with icons for various dimensions: PROMOTION, REASON, SHIP_MODE, STORE, STORE RETURNS, and STORE SALES. The 'STORE SALES' icon is highlighted with a blue background. The main area displays the 'STORE_SALES' table with 288,08 Rows. The table contains the following columns:

	STORE_SALES	288,08 Rows	...
123	SS_SOLD_DATE_SK		
123	SS_SOLD_TIME_SK		
123	SS_ITEM_SK		
123	SS_CUSTOMER_SK		
123	SS_CDEMO_SK		
123	SS_HDEMO_SK		
123	SS_ADDR_SK		
123	SS_STORE_SK		
123	SS_PROMO_SK		
123	SS_TICKET_NUMBER		

图 11-4. STORE_SALES 表中的字段列表

表格行数的右侧是一个放大镜图标; 单击该图标, Snowflake 将返回前 100 行供您查看 (如图 11-5 所示)。您可以使用水平和垂直滚动条滚动来浏览行, 以了解存在的数据。

	SI_SKU_ID	SI_SKU_ID_VW	SI_SKU_ID_H	-	SI_SKU_ID_VW_H	SI_SKU_ID_H	SI_SKU_ID_H	SI_SKU_ID_H	SI_SKU_ID_H
1	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
2	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
3	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
4	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
5	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
6	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
7	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
8	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
9	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
10	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
11	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
12	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
13	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
14	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
15	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
16	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
17	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
18	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
19	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
20	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
21	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
22	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
23	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
24	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
25	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
26	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
27	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
28	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
29	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
30	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
31	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
32	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
33	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
34	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
35	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
36	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
37	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
38	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
39	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
40	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
41	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
42	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
43	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
44	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
45	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
46	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
47	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
48	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
49	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
50	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
51	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
52	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
53	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
54	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
55	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
56	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
57	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
58	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
59	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
60	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
61	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
62	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
63	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
64	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
65	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
66	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
67	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
68	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
69	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
70	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
71	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
72	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
73	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
74	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
75	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
76	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
77	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
78	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
79	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
80	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
81	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
82	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
83	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
84	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
85	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
86	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
87	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
88	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
89	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
90	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
91	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
92	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
93	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
94	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
95	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
96	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
97	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
98	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
99	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	
100	3_4011_229	41_039	213_567	-	7_074_212	008_179	3_201	10_086_039	

图 11-5. 表中的前 100 行，单击放大镜图标后可以看到



需要活动的虚拟仓库才能预览数据。

接下来，单击 Run 按钮，该按钮是位于屏幕右上角圆圈内的蓝色箭头。或者，在 macOS 上按 Cmd + Return 或在 Windows 上按 Ctrl + Enter。

当您向下滚动结果时，您将看到恰好返回了 100 行。它们与您预览的 100 行不同。这可能是您所期望的。该子句是不确定的，随机选择要返回的行，因此它与显示前 100 行的预览不同是有道理的。

再次运行查询，您会注意到返回了相同的 100 条记录。这是您所期望的吗？鉴于该子句是不确定的，您可能希望看到一组不同的 100 条记录。返回相同 100 行的原因是 Snowflake 缓存，如第 2 章所述。查询结果缓存保存过去 24 小时内执行的每个查询的结果。因此，

当您重新运行同一查询时，将返回使用子句返回的相同 100 条记录。

当我们使用采样时，也会发生同样的事情吗？我看看。查询同一个表以返回 100 行，但这次使用而不是：

```
SELECT * FROM STORE_SALES SAMPLE (100 ROWS);
```

结果是固定大小采样方法的一个示例。如果再次运行查询，则不会收到相同的结果。这可能不是你所期望的。当我们查看基于分数的采样示例时，您认为会发生什么？

对于基于分数的采样示例，我们将使用而不是。它们是可互换的。我们还将抽样指定为 System sampling（系统抽样），并将百分比设置为 0.015%。我们选择了一个非常低的百分比，因为我们有超过 2880 亿行：

```
SELECT * FROM STORE_SALES TABLESAMPLE 系统 (0.015);
```

您会注意到，结果为样本返回了超过 3000 万条记录。如果我们需要并且想要这么大的样本量，那么对于样本量来说，这是可以的，但由于样本量超过 999,999 行，因此 Snowsight 无法生成统计指标。我们将在本章后面介绍统计数据。



示例查询结果是唯一的，因为它们不会缓存。您在第二次运行 fixed-size sampling 查询时看到了这一点。现在尝试再次执行基于分数的采样查询，看看会发生什么。和以前一样，如果您一次又一次地运行相同的采样查询，您将收到不同的记录子集。

使用自动统计和交互式结果

在 Snowsight 中可视化数据的一种方法是使用具有活动间结果的自动统计数据。

Snowsight 主动使用元数据来快速提供交互式结果，无论返回多少行。当您加载和查询数据时，Snowflake 会将存储在后台的所有丰富元数据放入 GUI 中，以便您可以开始了解数据中的一些趋势和/或及早发现任何错误。例如，您可以立即判断字段中是否有很多 null 值。这对 ETL 工程师尤其有利，他们将能够检测接收到的数据中的加载错误或间隙。

除了 Snowsight 自动上下文统计数据（告诉您填充了多少行）之外，您还将看到所有日期、时间和数字列的直方图，以及分类列的频率分布。此外，Snowflake

提供电子邮件列的电子邮件域分配和 JSON 对象的密钥分配。

我们来看看 Snowsight 的一些自动统计数据和交互式结果。对 CATALOG RETURNS 表运行语句，并将结果限制为 500 万。您将在结果右侧看到查询详细信息，但没有统计数据（如图 11-6 所示）：

```
SELECT * FROM CATALOG RETURNS LIMIT 5000000;
```

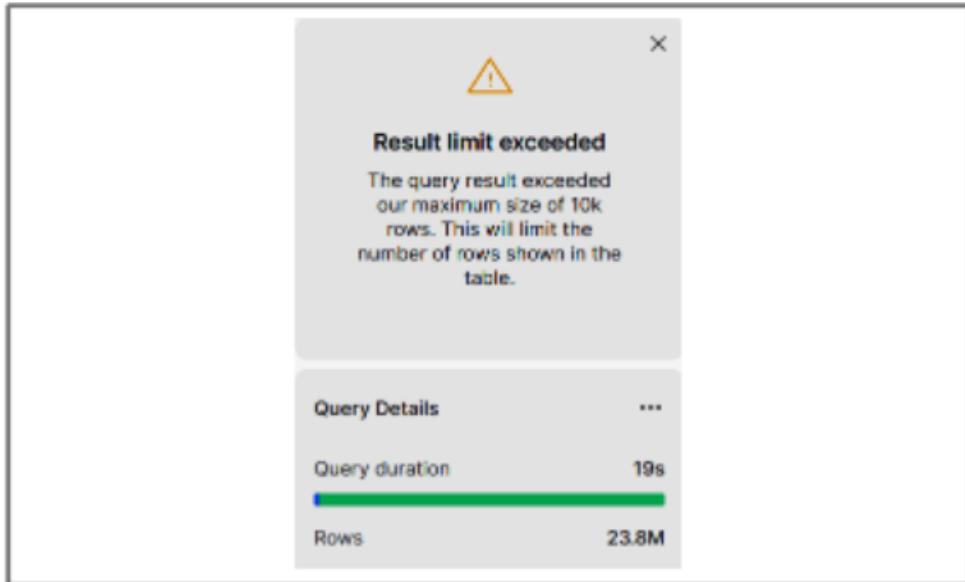


图 11-6. 统计数据不适用于包含超过 9,999 行的数据集

请记住，我们看不到统计数据的原因是结果集中的行数大于允许的行数。查看统计数据时，返回的行数必须小于 10000 行（如图 11-7 所示）。让我们用一个较小的数字再试一次：

```
SELECT * FROM CATALOG RETURNS LIMIT 9999;
```



如果您暂时没有看到统计数据，请尝试刷新屏幕。



图 11-7. 查询详情和统计信息

统计结果是交互式的。如果单击第 13 个图表（向下的第 13 个图表）的 CR_CATALOG_PAGE_SK 列的统计信息，您将看到 98% 的记录都有值（如图 11-8 所示）。

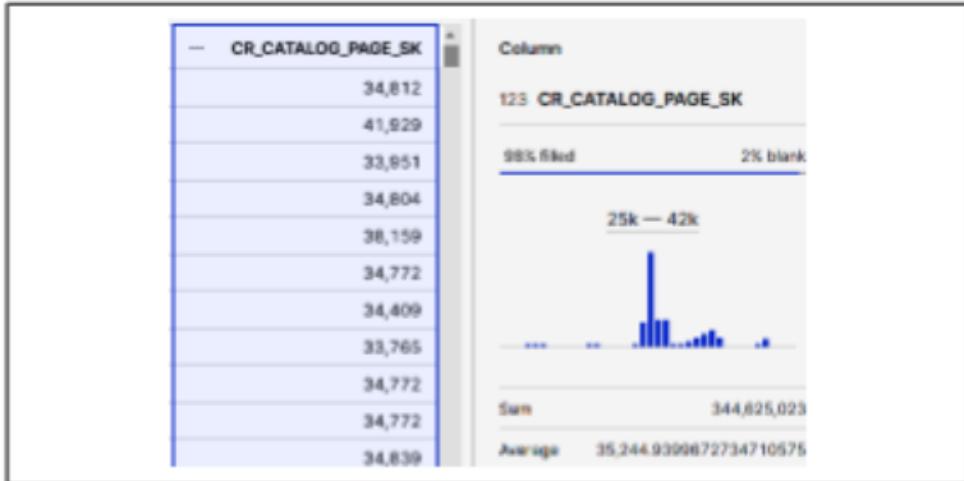


图 11-8. 单个列的查询结果统计信息

我们可能有兴趣了解那些没有 PAGE_SK 值。要查看这些记录，请单击“2% 无效”文本。您将注意到，该列已排序，因此现在所有具有 null 值的行都首先列出（如图 11-9 所示）。

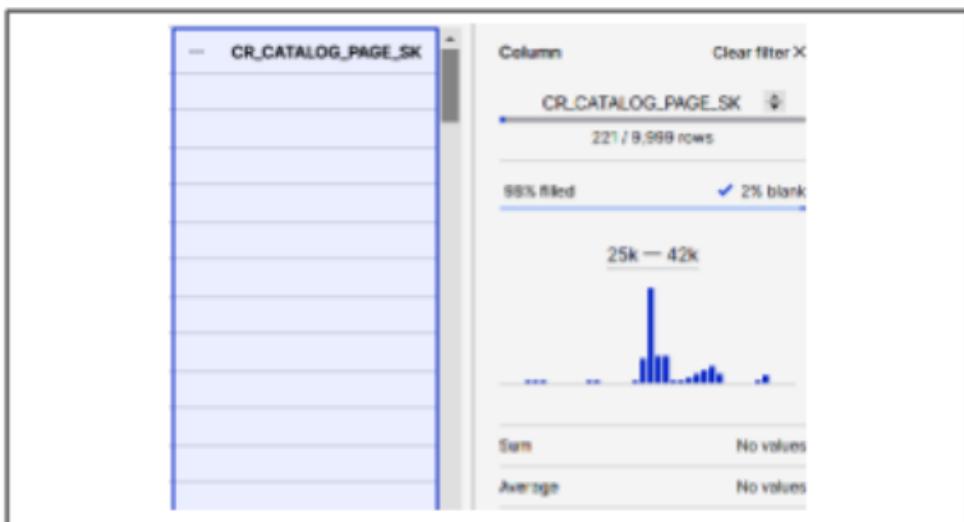


图 11-9. 单击“2% 无效”会显示具有 null 值的行

如果清除过滤器，将显示完整的统计信息。您可以使用滚动条向下滚动并查看有关每列的信息。我们将一起看另一个交互式图表。单击 CR_RETURN_QUANTITY 图（如图 11-10 所示）。

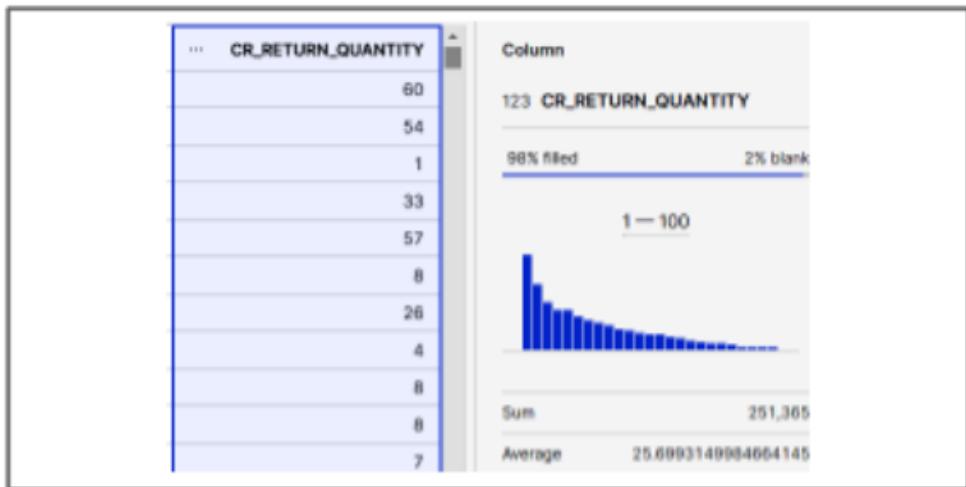


图 11-10. CR_RETURN_QUANTITY 详细信息和统计信息

现在，将光标悬停在图表上，然后单击条形图中的一个条形。我选择了数字 21（如图 11-11 所示）。在图表中选择一个条形后，您将能够在左侧的列中看到与该值关联的记录。

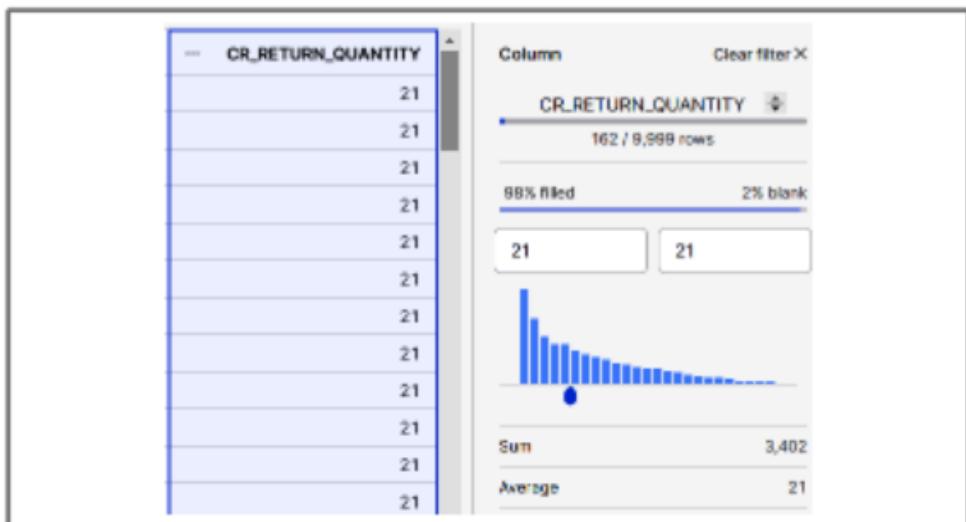


图 11-11. 在 Statistics 部分中选择的单个条形

您甚至可以在图表中选择一系列值（如图 11-12 所示）。

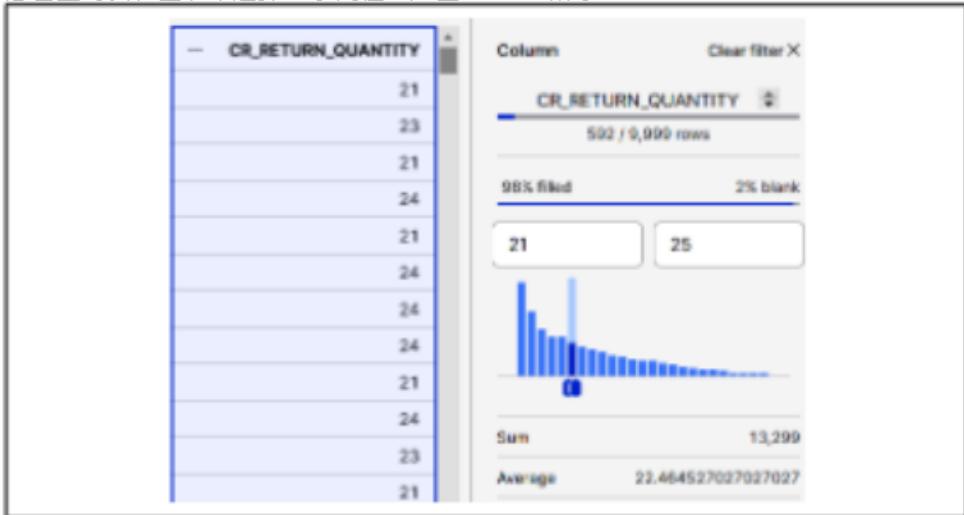


图 11-12. 在 Statistics 部分中选择的多个柱形

用户现在无需编写单独的查询或切换到数据科学或可视化工具来了解数据的完整性以及每列的不同值或日期范围，而是可以使用 Snowsight 的自动统计数据和交互式结果。

Snowsight 控制面板可视化

能够预览 100 行数据非常有用，尤其是能够使用自动统计和交互式结果来快速了解数据。但是，查看元数据通常是不够的。我们希望使用新的 Snowsight 可视化工具来更全面地了解数据。

创建控制面板和磁贴

导航回 Visualizations 文件夹，然后单击 Dashboards 菜单选项（如图 11-13 所示）。

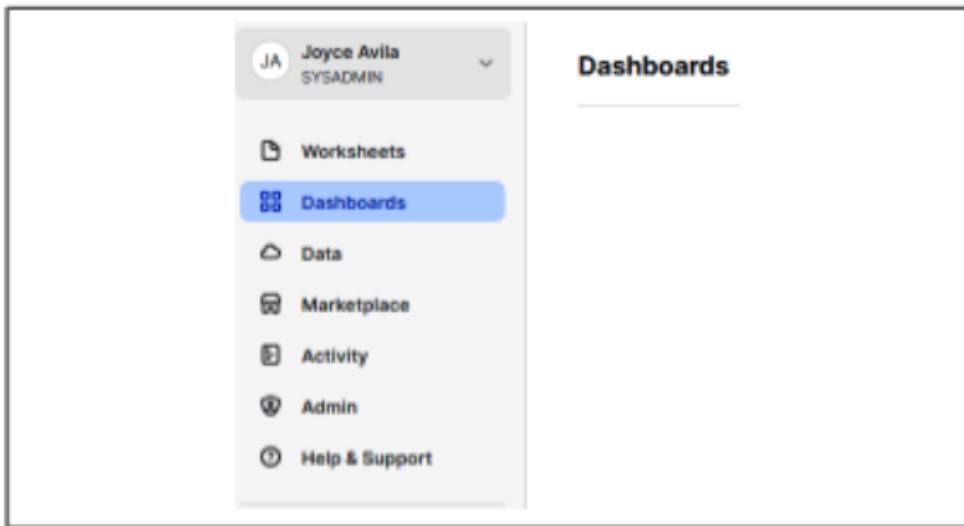


图 11-13. Dashboards（控制面板）菜单选项

在右侧，单击 + Dashboard 按钮创建一个新的 Dashboard（如图 11-14 所示）。



图 11-14. 在 Snowsight 中创建新控制面板

将仪表板命名为 Chapter11，然后单击 Create Dashboard 按钮（如图 11-15 所示）。



图 11-15. 命名新仪表板

接下来，单击 + New Tile 按钮添加新的仪表板瓦片（如图 11-16 所示）。

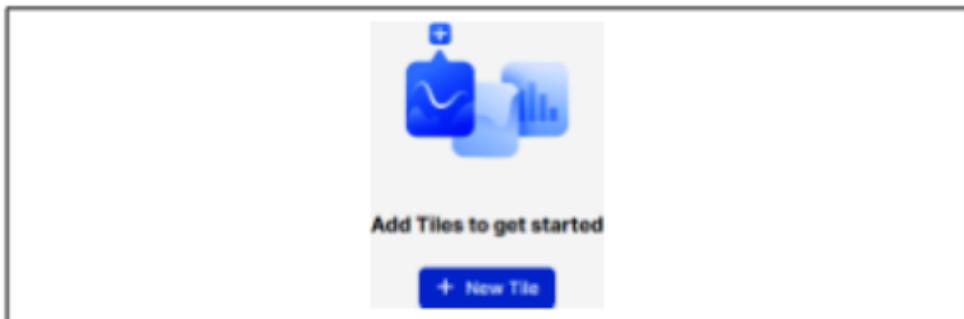


图 11-16. 添加新的仪表板磁贴

单击 + New Tile 按钮后，将打开一个工作表样式的屏幕。使用屏幕中间的下拉菜单，选择 database 和 schema，如图 11-17 所示。



图 11-17. 为新的 Chapter11 控制面板选择数据库和架构

我们之前花时间查看了 Snowsight 中的交互式统计数据，这些统计数据对于动态查看数据非常有帮助。现在，我们来探索一些可用于构建控制面板的图表可视化效果。

使用图表可视化

我们将从一个返回 10,000 行的基本查询开始：

```
SELECT * FROM TPCH_SF1.LINEITEM 限制 10000;
```

单击底部的 Chart 按钮查看图表。您会注意到它默认为折线图（如图 11-18 所示）。

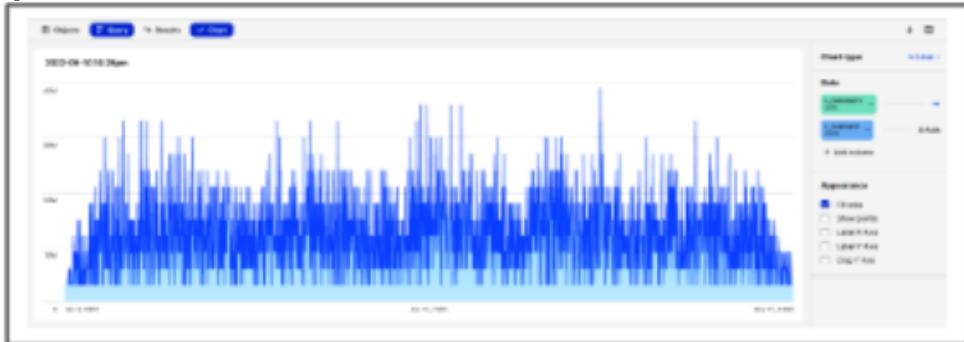


图 11-18. 默认折线图的结果

右侧有一个图表下拉菜单（如图 11-19 所示）。选择 Bar 并查看会发生什么。

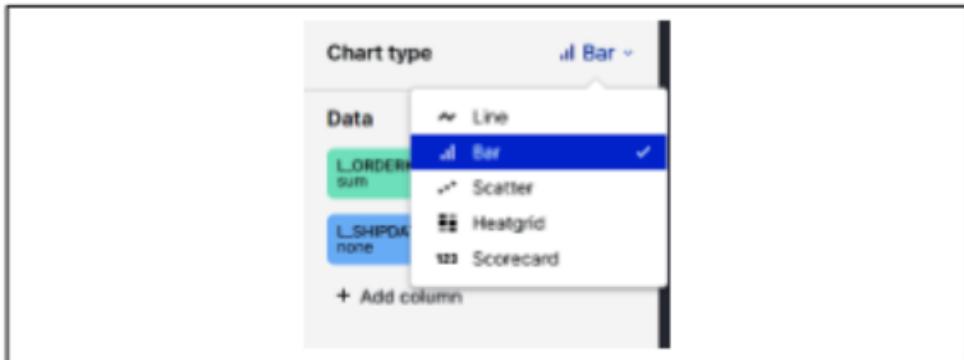


图 11-19. 可以选择不同的图表类型

选择条形图后，更改方向。请注意，方向位于 Appearance（外观）部分标签的正下方。确保 Descending 被选中作为顺序方向（如图 11-20 所示）。



图 11-20. 图表的外观方向和订单方向选择

做出选择后，您的图表应如图 11-21 所示。



图 11-21. 条形图结果

聚合和分桶数据

将图表结果按每日级别和降序排列并不能提供我们认为有帮助的视觉图片。让我们看看我们是否可以做出一些改进。

单击右侧的 SHIPDATE 按钮。您可以将 Bucketing（分桶）从 None（无）更改为您认为最好的其他值。尝试选择 Month 并将方向更改为 Ascending，以便时间从左向右移动。此外，选择 标签 X 轴 和 标签 Y 轴 选项并将方向返回到其原始选择（参见图 11-22）。

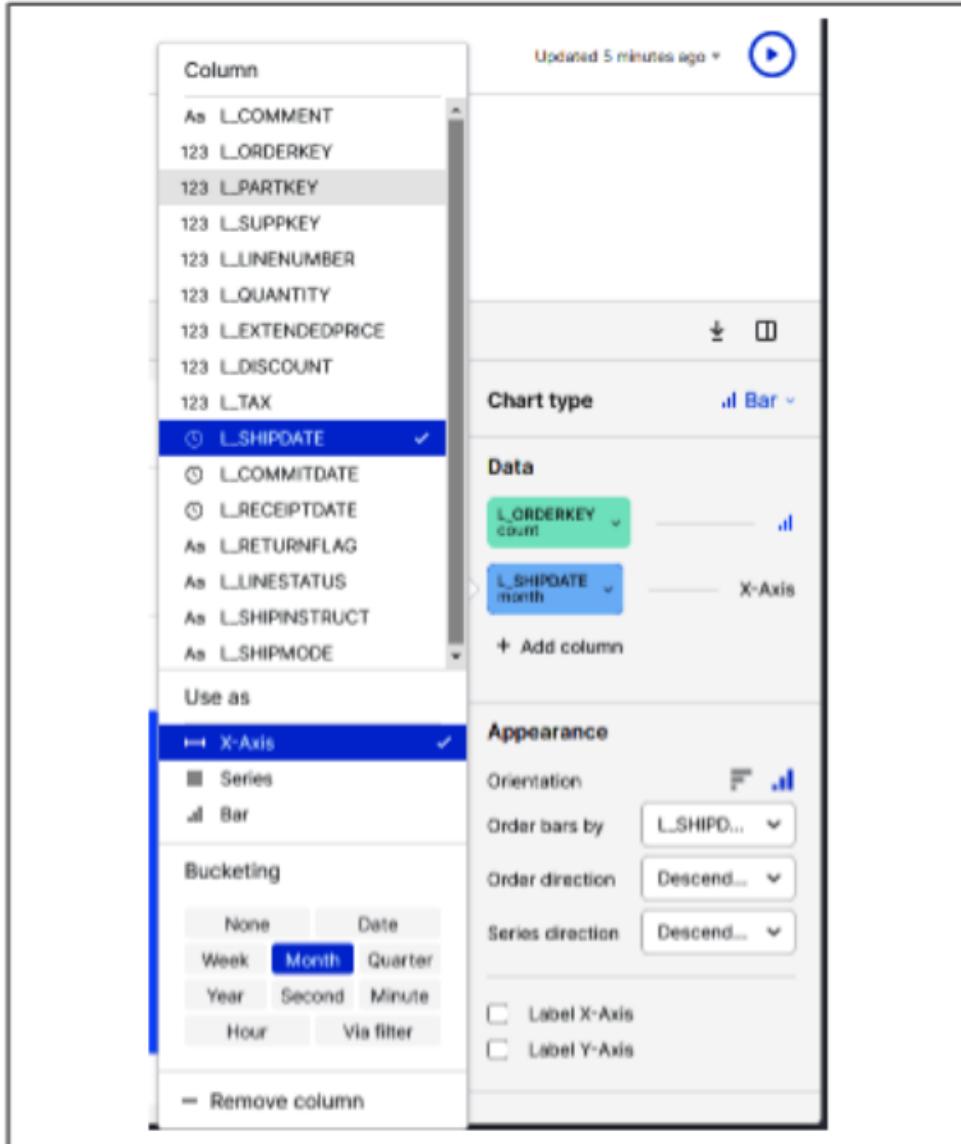


图 11-22. 对 SHIPDATE 列进行分桶

我所做的更改使我提供了一个更具吸引力的可视化效果，我可以从中获得一些见解（如图 11-23 所示）。

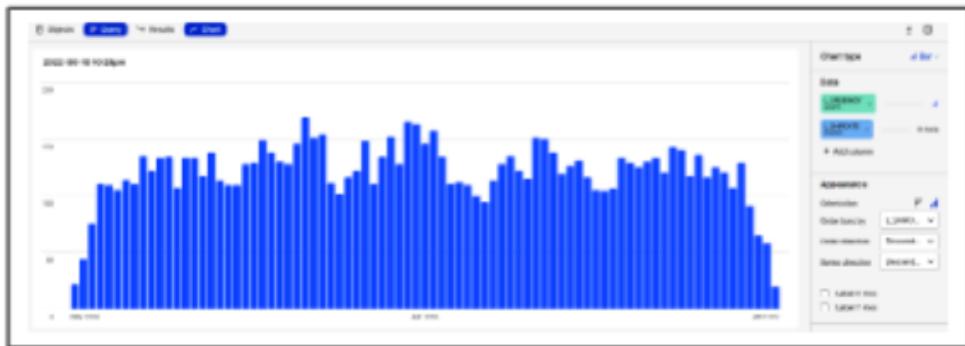


图 11-23. 对 SHIPDATE 列进行分桶后的数据可视化

请注意，如果您将光标悬停在任何条形的顶部，您将看到月份和年份以及该特定月份的订单数量。

在左上角，您将看到 Return to Chapter11 选项，其中 Chapter11 是仪表板的名称（如图 11-24 所示）。

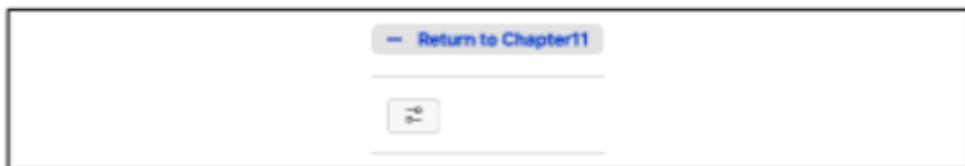


图 11-24. 屏幕左上角提供了返回控制面板的功能

继续并单击 返回第 11 章 选项。您会在仪表板上看到该图块。您还可以选择通过单击 Chapter11 仪表板标签下方的加号 (+) 来创建另一个磁贴（如图 11-25 所示）。



图 11-25. 在仪表板中添加另一个磁贴

通过单击左上角的 + 符号创建新工作表。现在点击 从工作表新建平铺 按钮在新工作表中创建新平铺（如图 11-26 所示）。

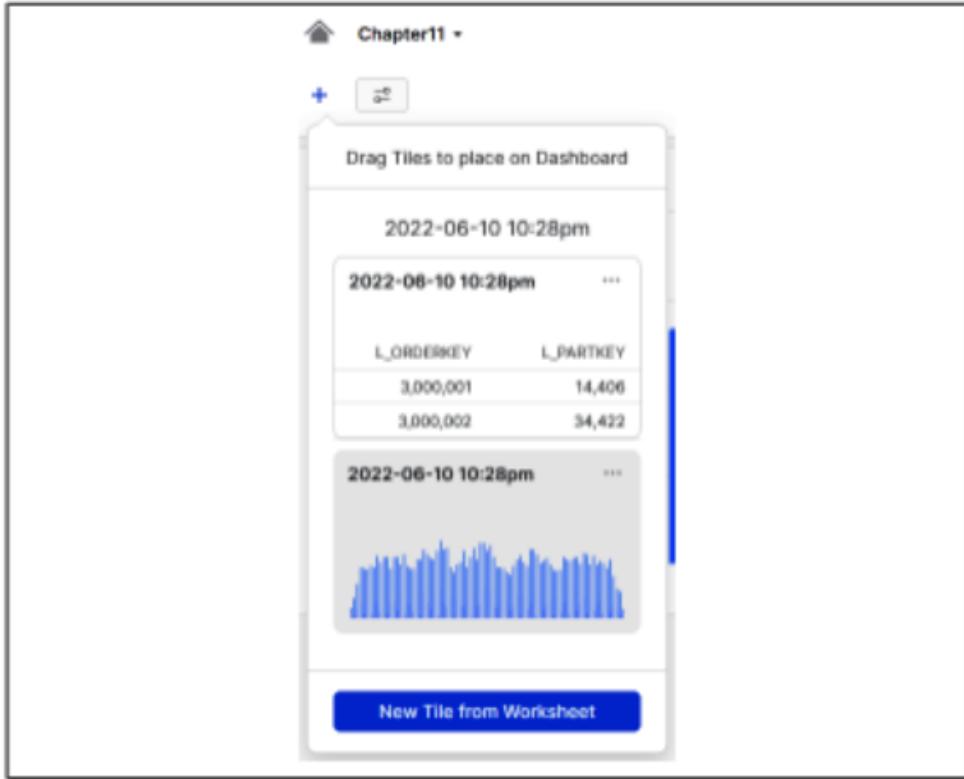


图 11-26. 在新工作表中创建新磁贴

如果需要，请花一些时间为仪表板构建其他磁贴。

编辑和删除瓦片

即使放置在仪表板上，也可以编辑单个磁贴。在图表的右上角（如图 11-27 所示），单击省略号并从选项中进行选择。当然，您可以通过编辑基础查询来编辑图表，一个方便的功能是您可以复制磁贴。复制磁贴会复制 under - ying 工作表和查询。

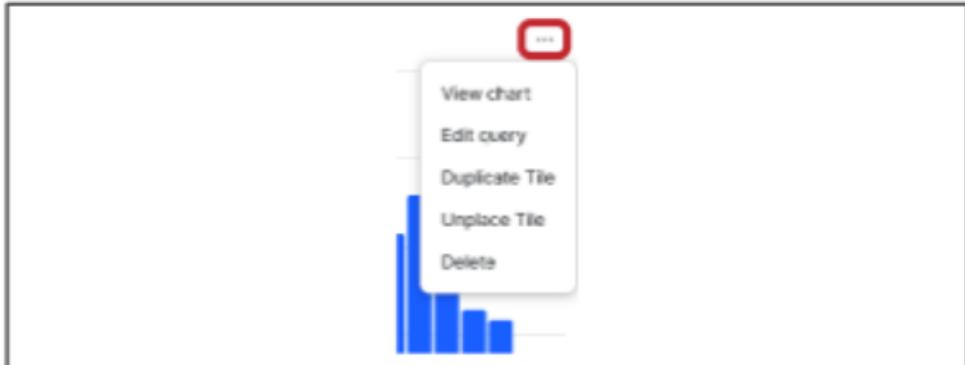


图 11-27. 单击省略号后可访问的磁贴选项



从仪表板中删除磁贴也会永久删除基础工作表。执行删除操作后，您将无法检索磁贴或工作表。

建议不要删除磁贴，除非您确定不需要构建图表的工作表。



删除磁贴的一种替代方法是直接选择 Unplace Tile，使其仍然存在，但无法在仪表板上查看。

协作

共享工作成果和协作的目标是让小组从个人的工作中受益，而无需每个人或业务部门重新创建工作。它还为更多创新和改进提供了机会，因为每个人都可能为知识库做出贡献并更快地推动它向前发展。

共享查询结果

要共享任何查询的结果，请首先确保导航回工作表。然后，只需单击向下箭头即可下载查询结果的 CSV 文件。然后可以与他人共享结果（如图 11-28 所示）。

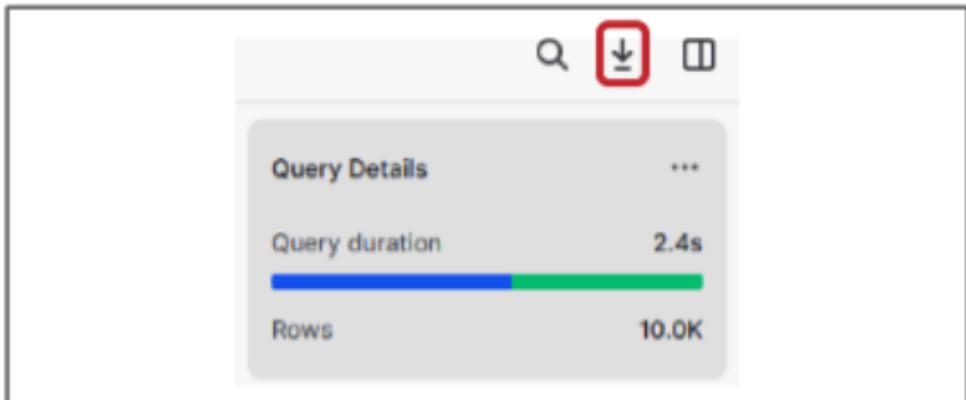


图 11-28. 使用向下箭头开始与他人共享查询结果的过程

使用私有链接协作处理控制面板

Snowsight 最重要的功能之一是个人能够在控制面板上协同工作。这很容易做到。导航回 Chapter11 控制面板。右上角有一个 Share 按钮（如图 11-29 所示）。

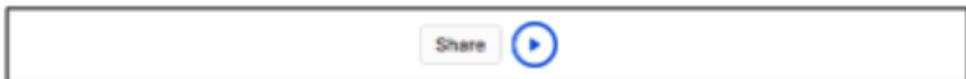


图 11-29. Share（共享）按钮允许与他人共享以进行仪表板协作

单击 Share 按钮，您将看到一个选项，用于邀请用户或允许拥有链接的每个人都可以访问仪表板。您可以通过 dashboard 链接为人员授予一些不同的权限，包括仅查看结果的能力，或运行 dashboard，这使他们能够刷新 dashboard 以获取最新结果（如图 11-30 所示）。

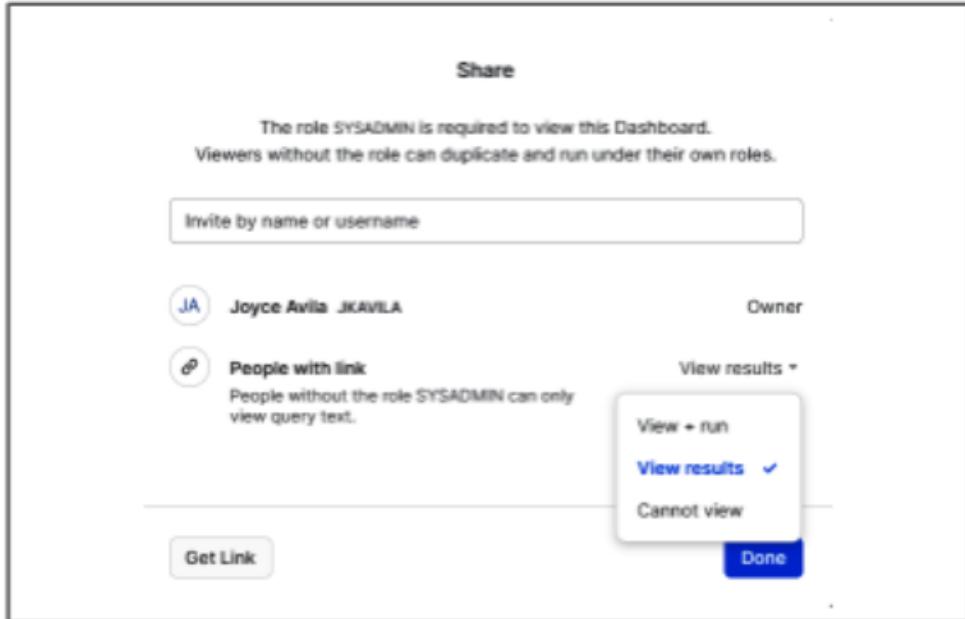


图 11-30. 为具有仪表板链接的人员分配权限

用户可以筛选结果并绘制数据图表以创建控制面板。然后，他们可以在几分钟内与同事共享控制面板，而无需离开 Snowflake 并且仅使用 SQL。

总结

如本章所述，Snowsight 以多种方式支持数据分析师活动。此外，还可以共享查询和控制面板，从而轻松与他人协作。

需要记住的一件重要事情是，Snowsight 可视化不是旨在处理非常大的数据集的 Power BI 工具的替代品。在下一章中，我们将介绍的主题之一是分析工作负载。

代码清理

由于我们使用了 Snowflake 的示例数据库，因此本章不需要进行代码清理。

知识检查

以下问题基于本章中涵盖的信息：

1. 您可以通过哪些不同的方式指示 Snowflake 通过采样技术返回行的子集？为什么在创建 Snowsight 可视化时需要使用采样？
2. 要使用预览选项，您是否编写 SQL 查询？使用预览选项时返回多少行？
3. 当您重新运行采样查询时，第二次返回的行是否与第一次返回的行相同？
4. 运行抽样查询是否需要活动虚拟仓库？是否需要活动的虚拟仓库来预览数据？

5. 您可以通过哪些不同的方式在 Snowsight 中存储数据？
6. 从仪表板中删除磁贴时会发生什么情况？
7. 使用 Snowsight 可视化有哪些使用案例？
8. 除了从仪表板中删除磁贴之外，还有哪些替代方法？
9. 您可以从一个工作表中创建多少个图表？
10. Snowflake 是否允许从内部数据源和/或外部数据源进行可视化？

这些问题的答案可在附录 A 中找到。

Snowflake Data Cloud 的工作负载

数据工作负载是一种能力、服务或流程，可以协助数据获取和数据访问，有助于从数据中提取价值，或作为开发其他人可以使用的东西的源泉。在前面的章节中，我们看到了 Snowflake 从头开始构建数据工作负载方法的许多示例，同时我们花时间掌握 Snowflake 构建块并学习如何利用 Snowflake 的强大功能。

在本章中，我们将重点介绍如何应用许多 Snowflake 工作负载来从 Snowflake 数据平台提供业务价值。我们将总结所有 Snowflake 工作负载，包括数据工程、数据仓库、数据湖、数据分析和数据科学工作负载（如图 12-1 所示）。

此外，我们将介绍数据协作工作负载，以讨论如何利用 Snowflake 的安全数据共享功能实现数据货币化。我们还将介绍 Snowflake 数据应用程序工作负载的几个示例，该工作负载用于开发大规模数据密集型应用程序。我们还将了解新的 Snowflake 工作负载、网络安全以及将 Snowflake 用作安全数据湖。最后，我们将了解 Snowflake 的最新工作负载 Unistore，它可用于统一分析和事务数据。

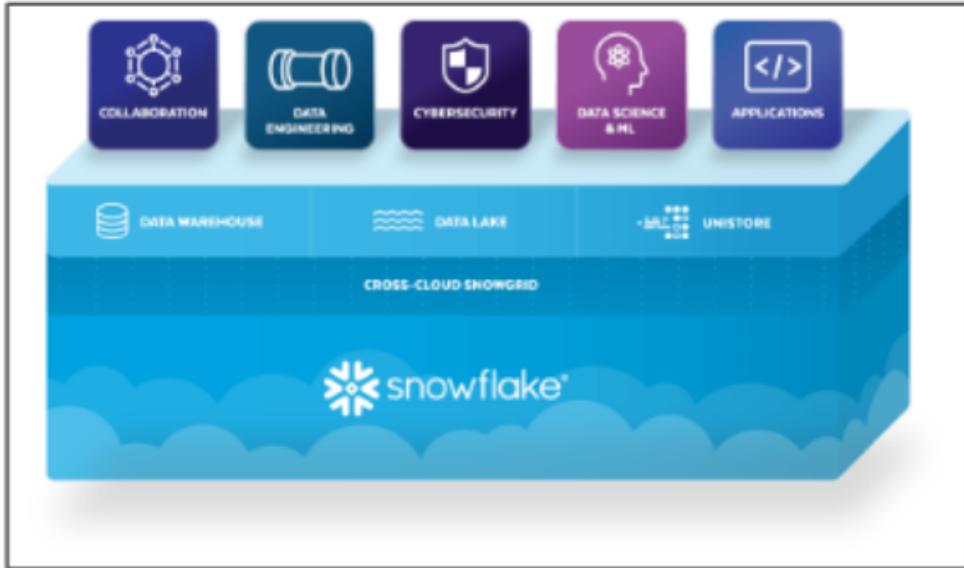


图 12-1. Snowflake 工作负载

准备工作

创建一个名为 Chapter12 Workloads 的新文件夹。如果您在创建新文件夹和工作表时需要帮助，请参阅第 8 页的“导航 Snowsight 工作表”。要设置工作表上下文，请确保您使用的是 SYSADMIN 角色和 COMPUTE_WH 虚拟仓库。

您会注意到，在下面的代码中，我们正在更改会话，以便我们不使用缓存的结果。这样做的原因是，我们想看看某些查询运行需要多长时间，并且我们不希望缓存的结果在第二次运行查询时影响性能时间：

```
使用角色 SYSADMIN;使用 WAREHOUSE COMPUTE_WH;创建或  
替换数据库DEMO12_DB;创建或替换 SCHEMA  
CYBERSECURITY;更改会话集 USE_CACHED_RESULT = FALSE;
```

数据工程

数据工程的目标是提取、转换、加载、聚合、存储和验证数据。为了实现这些目标，数据工程师需要构建高效的数据管道。数据管道是现代企业的命脉，高效的数据管道可能是为业务提供真正价值的架构与成为负担的架构之间的区别。

数据工程管道的构建方式也需要使数据符合数据治理规则和法规。尽管数据工程师不负责创建这些监管规则和法规，但他们的任务是实施这些规则和法规。第 7 章介绍了许多数据管理控制的示例。我们在该章中完成的一些动手示例是对象标记、分类、数据掩码和行访问策略。

如第 6 章所述，Snowflake 支持广泛的数据集成和处理工具，这为 Snowflake 数据工程师提供了更多的架构选择。例如，由于 Snowflake 的许多不同的集成工具及其几乎立即部署和扩展虚拟仓库的能力，因此 ELT 风格的数据工程方法成为可能。

Snowflake 的数据摄取过程得到了简化。无需转换即可轻松提取 JSON、XML、Avro、Parquet 和其他数据。当 schema 更改时，无需调整管道。在 Snowflake 中，使用 SQL 查询结构和半结构化数据也很容易。而且，正如我们在第 4 章中所看到的，Snowflake 还支持非结构化数据。

在第 6 章中，我们了解了 Snowflake 的数据加载工具以及如何卸载数据。我们发现了如何在 Snowflake 工作表中使用 SQL 加载数据，以及如何使用 Web UI 数据加载功能。我们探索了 SnowSQL 命令行界面（CLI），在那里我们熟悉了 和 commands。我们还了解了用于持续加载和流式处理的数据管道，并了解了第三方 ELT 工具。在第 8 章中，我们了解了敏捷软件交付，包括持续数据集成、持续交付和持续部署。

在前面的章节中，我们了解到，借助 Snowflake 独特的数据云平台，组织可以收集比以往更多的数据。此外，数据工程师可以使用 Snowflake 数据平台以他们选择的语言构建简单、可靠的数据管道，从而使他们能够提供现代企业所需的性能和可扩展性，以支持数千个并发用户，而不会发生资源争用。

除了能够近乎实时地摄取、转换和查询数据之外，Snowflake 还是一个维护成本极低的平台。正如我们在 第 9 章 中看到的那样，Snowflake 负责为您定义和维护分区。其自动集群功能无缝且持续地管理所有重新集群；仅在需要时对表进行重新聚类。Snowflake 还负责查询优化、元数据和统计信息管理、索引管理和计算扩展。就此而言，Snowflake 平台的管理很简单；您可以轻松设置用户、创建对象和实施成本监控解决方案。

数据仓库

组织构建数据仓库来存储和集中整个组织生成的数据，目的是执行快速数据分析和报告。当然，组织希望通过低维护租期和低成本的数据仓库来实现这些目标。Snowflake 的数据仓库工作负载可以轻松满足这些要求。

正如我们所看到的，Snowflake 通过自动缓存和查询优化自动处理大规模性能优化。Snowflake 还处理补救数据库管理员活动，例如存储分配、容量规划、资源扩展、加密以及无缝备份和升级。Snowflake 近乎无限的存储、规模和速度意味着分析师可以随时运行任何查询并快速获得答案。

Snowflake 的数据云平台架构可以满足组织的数据仓库需求。然而，要确保成功实施数据管理计划，还需要做更多工作。开发一个数据模型，以最好地组织数据仓库中的信息也很重要。此外，考虑最适合组织需求的数据建模方法也很重要，因为有效且可持续的数据建模方法为数据仓库提供了坚实的基础。

Snowflake 支持所有标准数据模型，包括 Irmon 第三范式方法、Kimball 维度模型、新的 Data Vault 模型等。如果您还记得前面的章节，除了 NOT NULL 之外，引用完整性约束在 Snowflake 中是信息性的，这允许在方法上具有更大的灵活性。

您可能熟悉数据库架构设计方法，包括 1971 年定义的关系数据库的第三范式。第三范式使用归一化原则来减少数据的重复，避免数据异常，保证引用完整性，简化数据管理。Irmon 数据建模方法建立在第三范式之上，用于识别关键主题领域和关键业务实体。Kimball 模型于 1996 年推出，它允许用户构建多个星型架构以满足各种报告需求。Irmon 和 Kimball 数据仓库设计是讨论最广泛的两种方法；但是，Data Vault 数据模型方法已变得非常流行。

Data Vault 2.0 建模

商业智能（BI）建模的 Data Vault 系统以一种将结构信息（如表的唯一标识符或外键关系）与其属性分开的方式组织数据。专为满足现代需求而设计

数据仓库中，Data Vault 方法是一种混合方法，它结合了第三范式和星型模式建模方法的优点。

Data Vault 是由 Dan Linstedt 在为 Lockheed Martin 工作时发明的。它于 2000 年初向公众推出。最初，Data Vault 专注于数据仓库的建模技术和数据加载流程。后来，Data Vault 的范围扩展到包括参考架构、敏捷项目交付、自动化和持续改进，Data Vault 被标记为 Data Vault 2.0。

Snowflake 支持 Data Vault 2.0，使用 Data Vault 建模方法，您可能会在 Snowflake 上获得更好的结果。这是因为 Data Vault 2.0 可以利用 Snowflake 的大规模并行处理计算集群和优化的列式存储格式。此外，使用 Snowflake，您无需预先规划分区或分配键。您也不需要构建索引即可获得出色的性能。

数据保险库模型由以下基本表类型组成（如图 12-2 所示）：

Hub 表

Hub 表包含主题的所有唯一业务键。集线器还包括描述业务密钥来源的元数据，称为记录源，用于跟踪数据的来源和时间。例如，HUB_EMPLOYEE 可以使用员工编号来标识唯一的员工。

链接表

链接表跟踪中心之间的所有关系，实质上是描述多对多关系。链接可以链接到其他链接。链接通常用于处理数据粒度的变化，这减少了向链接的中心添加新的业务密钥的影响。例如，LINK_EMPLOYEE_STORE 将跟踪员工与他们工作的商店位置之间的关系。

卫星表

卫星表包含与链接或中心相关的任何属性，并在它们更改时更新这些属性。这类似于 Kimball II 型缓慢变化的维度。中心和链接构成了数据模型的结构，而卫星则包含时间和描述性属性。这些元数据属性包含记录生效的日期和过期日期。因此，卫星提供了强大的历史功能。例如，SAT_EMPLOYEE 可能包括员工的名字和姓氏、职称或雇用日期等属性。

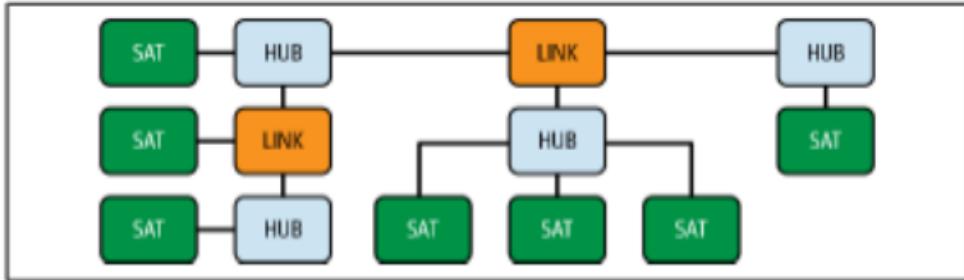


图 12-2. 集线器、链接和卫星的 Data Vault 关系示例

Data Vault 模型在某些情况下特别有用。Data Vault 使添加属性变得简单；因此，当存在许多数据源系统或存在不断变化的关系时，它提供最大的好处。Data Vault 方法还可以更快地加载数据，因为可以同时并行加载表，这意味着您可以从多个系统快速加载数据。Data Vault 可以做到这一点，因为在加载过程中，表之间的依赖关系较少。此外，由于只有插入，因此简化了摄取过程，这比 merges 或 upsert 加载得更快。当您需要轻松跟踪和审计数据时，Data Vault 也是一个不错的选择，因为 Data Vault 本身就具有审计功能。为每一行输入记录源，并且加载时间作为 satellites 中主键的一部分包含在内。

如前所述，如果您需要从多个源系统快速加载数据、数据关系经常更改，或者您需要轻松跟踪和审计数据，则 Data Vault 最有用。这些场景是 Data Vault 方法的亮点。但是，如果您的数据更简单，则 Kimball 维度模型可能是更好的选择，因为在不需要的地方实施 Data Vault 需要增加业务逻辑的数量。

另一个考虑因素是 Data Vault 比其他建模方法需要更多的存储空间。将主题区域划分为三个不同的表格将增加表格的数量。因此，如果您只有一个源系统，或者数据相对静态，那么使用 Data Vault 可能没有意义。

总结一下复杂性和实施时间方面的差异，Data Vault 方法和 Kimball 方法相对简单，与 Inmon 方法相比，实施所需的时间更少。Kimball 方法基于四步规范化过程，而 Data Vault 使用敏捷方法。一般来说，实施 Data Vault 建模方法的成本是三种方法中最低的，但 Data Vault 确实需要更多存储空间，并且当数据简单且不需要数据审计时，它不是最佳选择。

在 Snowflake 中转换数据

一个经常被忽视的话题是如何在 Snowflake 中转换数据。摄取到 Snowflake 中的原始数据通常来自不同的来源，这些来源通常是不同的。因此，您可能需要对齐这些数据类型或为空列分配默认值。可能需要将数字转换为字符串或将字符串转换为数字。您可能需要平展半结构化数据或重命名列。有出色的第三方工具，如 dbt，可以帮助您转换数据。除了转换数据之外，还可以在简化和清理数据时使用其他数据转换工具，例如 dbt Labs 的数据构建工具（dbt），数据科学家可能需要这些工具。您还可以采用一些本机方法进行数据转换。

在 Snowflake 中持续转换数据的一种简单方法是使用任务自动执行语句。任务（在第 3 章中首次介绍）能够组合来自多个源表的数据。使用任务进行数据转换还有其他优点：它们可用于以高性能方式逐个构建更复杂的转换，并且可以使用资源监视器跟踪与使用任务相关的计算成本。

具体化视图是另一种经常用于实时简单转换的本机数据转换方法。使用物化视图的优点是它们易于创建并实时提供转换的数据。

数据湖

组织构建数据湖以利用来自各种数据源的大量数据，最终整合所有数据并提供单一事实来源。最终，目标是向目标用户和适当的用例提供所有数据。

迁移到云在许多方面简化了数据基础架构和数据管理，因此用户可以利用数据湖工作负载来实现他们的许多目标。然而，直到最近，在数据湖环境中构建数据应用程序还是不可行的。数据分析工作负载也是数据湖工作负载中缺失的部分。但是，Snowflake 的架构消除了对数据存储和企业数据仓库环境进行独立维护的需要。数据湖和仓库之间的区别通过 Snowflake 的可扩展数据架构被消除。

使用 Snowflake，您可以混合和匹配数据湖模式的组件，以释放数据的全部潜力。分析师可以直接通过数据湖查询数据，具有无限的弹性可扩展性，并且没有资源争用或并发问题。数据工程师可以使用简化的架构来运行可靠且高性能的数据管道。借助内置的治理和安全性，所有用户都可以访问

数据湖中的所有数据都具有精细级访问和隐私控制。Snowflake 可以用作所有数据和所有用户的完整数据湖。

在第 374 页的 “Data Warehousing” 中，我们了解了数据建模技术。具体来说，我们了解了 Data Vault 2.0 方法。重要的是要了解 Data Vault 和数据湖是互补的，可以满足组织分析要求的不同部分。数据湖提供了一个持久的暂存区域，旨在捕获数据超集，这些数据的体积太大、速度太快，或者具有太多结构变化，传统关系系统无法处理。数据湖是数据科学活动的绝佳选择。为了集成对业务用户最有用的数据，可以将数据湖中的数据子集馈送到 Data Vault。

我们提到过，数据湖架构对于数据科学工作负载非常有用。在接下来的部分中，我们还将了解用户可以从 Snowflake 的数据湖工作负载中受益的其他方式。在第 383 页的 “数据应用程序” 中，我们将看到如何在 Snowflake 上构建大规模数据密集型应用程序的示例。我们还将了解 Snowflake 作为安全数据湖如何支持新的 Snowflake 网络安全工作负载。

数据协作

Secure Data Sharing 支持数据协作。Snowflake Secure Data Sharing 工作负载的重要性怎么强调都不为过。第 10 章专门解释了四种 Snowflake 安全数据共享方法（如图 12-3 所示）。



图 12-3. Snowflake 安全的四种数据共享方法

了解 Snowflake 安全数据共享的工作原理、如何利用 Snowflake Data Exchange 的强大功能以及如何在 Snowflake Marketplace 上发布和购物非常重要。随着 Snowflake Marketplace 的兴起，数据网络效应随之而来，驻留在 Marketplace 中的数据变得更有价值，因为它被更频繁地请求，并且额外的数据提供者也提供了他们的数据。

Snowflake Marketplace 使组织能够利用 Snowflake 安全数据共享的无副本共享和安全功能来提供其专有数据和见解。数据货币化从未如此简单。

数据货币化

如果您的组织愿意通过其数据获利，您将需要从创建可共享数据清单开始。任何可能危及您的竞争力的信息（例如商业秘密）或违反隐私法的信息都不应被考虑。操作数据（例如交易记录和传感器日志）是一种选择。您可能还需要考虑将聚合或去标识化的营销数据（例如客户信息、偏好或 Web 流量）货币化。或者，您可能拥有其他人感兴趣的开源数据，如社交网络帖子或政府统计数据。

确定要考虑变现的数据后，您需要确定原始数据是否足够有价值，或者是否希望使用其他数据集来扩充数据。例如，如果零售商的商店级数据与人口统计数据相结合，它将更有价值。

接下来你要考虑的是如何为数据定价。两种常见的定价策略是成本定价和价值定价。在成本定价方法中，您首先需要计算采购、打包和共享数据所涉及的成本，然后加上高于该成本的金额，即您的利润率。对于价值定价方法，您需要考虑客户每次数据的价值。对客户的价值将基于数据的唯一性、获取数据所需的难度级别以及已经提供相同数据的竞争对手数量等因素。数据的价值还取决于数据更新的频率以及您将提供的历史数据量。此外，如果您的数据范围是全球性的，而不是本地的，则您的数据可能会更有价值。作为定价策略的一部分，您还需要决定是将数据作为集合还是作为订阅出售。或者，您可以决定根据数据使用情况收费。



考虑免费增值定价结构，该结构具有免费有限数量的数据、标准访问的基本费用，然后是其他服务功能的额外费用。免费增值定价结构是获得新客户的好方法。

数据共享的法规和合规性要求

除了将数据货币化（这更像是拉动数据市场）之外，由于监管和合规性要求，预计还会有多人推动集中式数据市场。最近的一个例子是美国国家卫生研究所（NIH），它是世界上最大的生物医学研究资助者。2023 年 1 月，NIH 将开始要求其每年资助的 2,500 个机构和 300,000 名研究人员中的大部分公开他们的数据。

NIH 的任务旨在应对科学研究中的可重复性危机，这是一种持续的方法论危机，其中科学结果越来越难以或不可能复制。可重复性危机浪费了纳税人的钱，破坏了公众对科学的信任。一旦 NIH 研究数据明年能够公开使用，看看科学界和其他人的反应将很有趣。能够访问可以独立执行分析的原始数据将为研究结果提供可信度。

Snowflake 独特的安全数据共享功能使充分利用 Snowflake 强大的数据分析和数据科学工作负载成为可能。正如您将在下一节中看到的那样，安全数据共享通常在数据分析工作负载中发挥着重要作用。

数据分析

数据分析可帮助用户理解数据。选择正确的数据分析工具很重要，尤其是在试图为数据分析民主化创建清晰的路径时。数据分析民主化使组织中的每个人都可以轻松访问和处理组织的数据。这样，人们就可以做出基于数据的决策，并利用数据提供卓越的客户体验。

Snowflake 合作伙伴提供了大量不同类型的原生商业智能（BI）工具，以帮助实现数据民主化。Snowflake 原生 BI 工具包括熟悉的名字，例如 Domo、Power BI、Sisense、Sigma、Tableau 和 ThoughtSpot。有关 Snowflake BI 合作伙伴的完整列表，请参阅 [Snowflake 在线文档](#)。

让我们集中精力更多地了解这些分析工具在各个行业中的使用方式。在浏览以下示例时，请注意您将数据共享功能视为整体分析解决方案一部分的频率。

几年前，大多数组织无法想象能够利用情绪分析、集群分析、预测建模、机器学习和人工智能等高级分析。不过，高级分析的采用一直在增长，尤其是随着云平台的激增。特定于行业的高级分析解决方案组合一直在发展，在某些行业（如金融、医疗保健、制造、零售和通信）中尤为明显。让我们探索一些适用于这些行业的专业高级分析解决方案。

金融行业的高级分析

金融机构包括商业和投资银行、信用合作社、信用卡公司、股票经纪、外汇服务和保险

公司。金融服务公司的运营环境承受着来自新兴公司的巨大竞争压力，并且需要满足不断变化的监管要求。

为了在这种充满挑战的环境中取得成功，金融机构还必须找到降低与欺诈交易和信用违约相关的风险的方法。正是在降低风险方面，金融机构通过使用数据科学技术取得了最大的进步。金融服务公司能够使用基于机器学习的高级预警系统实时检测欺诈活动，该系统能够在交易发生时挖掘数据。通过部署分析驱动的数字信用评估和信用收集分析，还可以更快、更彻底地审查潜在借款人。

许多世界领先的金融服务公司已经意识到无缝共享信息以帮助管理和降低风险的价值。因此，Snowflake 数据平台上原生提供了越来越多的财务相关数据。现在，可以更轻松地与监管机构共享这些数据，例如，通过使用 Snowflake 安全数据共享功能。它也可以用作第三方数据，以增强组织的高级分析模型。Equi-fax、First American Data & Analytics 和 S&P 全球市场情报在 Snowflake Marketplace 上提供他们的数据，以便他们的客户可以简单快速地实时访问数据，而无需存储数据的副本。

面向医疗保健行业的高级分析

医疗保健分析包括 360 度患者视图、临床试验数据管理以及互联医疗设备和应用程序。通过设施利用率和人员配备来提高运营效率也是医疗保健行业的优先事项，因为该行业可以节省大量成本。

此外，由于基于结果的治疗的经济激励，共享患者信息的需求急剧增加，这在美国是一个相对较新的概念。这种基于价值的护理模式，而不是传统的按服务收费模式，导致医疗保健行业从根本上转向数据共享，尤其是在美国。

集中式电子健康记录（EHR）系统正在迅速取代患者图表中归档的传统手写笔记。通过 Snowflake 的安全数据共享功能，无需传输这些敏感信息，即可更轻松地安全地共享患者信息。来自医疗物联网（IoMT）的数据，包括消费者健康可穿戴设备、支持传感器的医院病床和药物跟踪系统，可以使用当今复杂的数据科学工具轻松摄取、集成和分析。

从 EHR 和 IoT 数据中访问更完整的数据提供了无限的机会。借助高级分析和更完整的数据，医疗保健提供商可以更快、更准确地诊断患者，发现可能导致突破性治疗方法的趋势，并通过可能带来新收入来源的新业务用例进行创新。

面向制造业和物流服务的高级分析

制造商使用高级分析技术来提高产品质量并提高制造运营效率。来自制造运营传感器的实时馈送可用于在质量控制问题变得过度之前识别它们。通过预先决定生产需求的起伏，可以更好地预测和管理劳动力需求的波动。

通过与合作伙伴实施安全数据共享，制造商可以打破传统供应变更管理系统、企业资源规划平台、订单履行系统和物联网（IoT）设备中经常出现的数据孤岛。将安全数据共享与高级分析相结合，可以通过实时跟踪货物和库存来减少瓶颈和中断，从而帮助提高供应链的效率。

零售垂直行业以及通信和媒体行业的营销分析

为您的品牌、商品和业务创造和维护良好的声誉非常重要。潜在客户对您的产品或服务的最初反应通常取决于他们在社交媒体上看到的评论或回复。这就是为什么利用在线品牌的情绪来计算分数很重要的原因，该分数表明社交媒体帖子或产品评论是负面的、中立的还是积极的。通过使用文本转语音将评分应用于电子邮件和电话对话也很重要。在此过程中，您可以确定特定的产品和服务或社交媒体营销工作，以引起您想要的积极反应。它还允许您识别客户服务中的任何故障。情绪分析是一种可用于自动执行为单词分配分数并确定您的声誉总体上是否积极的过程的方法。

不仅您的客户用来描述您或您的产品的词语很重要，您用来向他们推销的词语也很重要。内容策略可以采用串行测试，并使用无监督学习算法来发现哪种单词选择或颜色在您的数字广告活动中最有效。内容策略也可能受到第三方数据的影响，例如天气或 COVID-19 数据，Snowflake Marketplace 上提供的数据。内容策略可以与

渠道优化，它使用亲和力分析和市场篮分析等工具。例如，渠道优化有助于突出特定社交媒体平台（如 Instagram、Pinterest 和 YouTube）上错过的机会联系。此外，一些渠道可以使用推荐引擎来推荐类似产品或追加销售商品。

数据科学和机器学习模型还可以帮助识别可能提高客户忠诚度的因素，因为与向新潜在客户进行营销相比，向现有客户进行营销更容易且成本更低。为了提供帮助，可以从 Snowflake Marketplace 上的技术公司 Ibotta 获取第三方忠诚度数据。在向新的潜在客户进行营销时，通过潜在客户评分可以显著提高成功的可能性。使用增强的第三方数据以及您自己的营销数据时，潜在客户的评分最准确。除了用于潜在客户评分外，预测分析还可用于帮助识别具有较高潜在生命周期价值的客户。总而言之，营销分析和数据科学旨在提供有关客户偏好和行为的宝贵见解，以实现营销预算优化和利润最大化。

在这些营销分析示例中，您可能已经注意到的一件事是，营销分析通常采用更具战术性的方法来进行个人层面的分析。医疗保健行业的许多高级分析解决方案也是如此，它们可以在最精细的级别上使用。

无论您在哪个行业或垂直领域运营，无论您选择什么 BI 工具，您都需要访问分析所需的数据。对于许多组织来说，最重要的数据源之一是客户自己的内部数据应用程序捕获的客户数据。正如我们接下来将看到的，Snowflake 数据应用程序工作负载是自定义构建的数据密集型应用程序最理想的选择之一。

数据应用

数据应用程序开发人员的目标是构建能够为公司带来价值的应用程序，同时满足最终用户的实际和定性需求。实现这一目标，同时始终如一地提供高质量的用户体验是一项艰巨的任务，尤其是在开发大规模数据密集型应用程序时。

数据密集型应用程序以数十亿行形式处理数 TB 的快速变化的数据，并利用这些数据为用户提供价值。应用包括销售和零售跟踪系统、物联网、健康和 Customer 360 应用程序，后者用于有针对性的电子邮件活动和生成个性化优惠。嵌入式分析和机器学习应用程序是数据密集型用例的进一步示例。

为了快速构建可提供高价值的应用程序，许多开发人员在平台基础上构建应用程序，并内置了所需的基础设施元素，例如数据存储、处理、管理和安全性。使用此策略，

这些开发人员寻找易于使用的云平台;是安全的;支持可扩展性、并发性和半结构化数据的使用;并包括 Intelligent Query Execution。Snowflake 是一个包含所有这些基础元素的基于云的平台,凭借其几乎为零的管理能力,开发人员可以专注于应用程序开发,而不是数据平台管理和数据平台安全。

使用 Snowflake 云平台的面向公众的流行数据密集型应用程序示例包括自行车共享应用程序 Lime、杂货配送应用程序 Instacart、在线旅游应用程序 Hotel Tonight 和数字学习应用程序 Blackboard。另一个基于 Snowflake 构建的高性能应用程序是 Twilio SendGrid,这是一款营销活动应用程序,每月代表数千名客户发送数十亿封电子邮件。

Snowflake 数据平台的一个关键功能是它对开发人员友好。它允许应用程序开发人员在多种编程语言中进行选择,包括 Node.js、.NET、Go、Java、Python 和 SQL。例如,Snowflake SQL REST API 可用于构建应用程序和集成自定义插件。使用 Snowflake 的 SQL API Playground,您可以获得使用 SQL REST API 开发数据应用程序和集成的实践经验。在 Playground 中,您可以浏览预定义的查询和响应,或者使用它在 Snowflake 环境中试验和执行真正的查询。您还可以下载 API 的 Postman Collection 或浏览 OpenAPI 规范文件。Snowflake SQL API 游戏场地可在 <https://api.developers.snowflake.com> 访问。

2021 年, Snowflake 推出了名为 Powered by Snowflake 的互联应用程序部署模型,该计划旨在帮助公司在数据云中构建、运营和发展应用程序。许多 Powered by Snowflake 合作伙伴,如 AuditBoard、Hunters、MessageGears、Securonix 和 Supergrain,已经在其应用程序中采用了连接的应用程序模型。

为了加快您的 Snowflake 应用程序之旅,可以利用 Powered By Snowflake 计划参加研讨会并请求访问专家,以帮助您为面向客户的应用程序设计正确的数据架构。

要查看 Powered By Snowflake 应用程序的示例,请访问 Snowflake 网站。

此外,还可以使用 Snowflake Snowpark 开发人员框架构建数据密集型应用程序,供内部业务用户使用。Snowpark 经常用于构建用于创建机器学习应用程序的复杂管道。应用程序开发人员可以使用 Snowpark 框架,以他们熟悉的编码语言和结构协作处理数据项目。最重要的是,开发人员可以构建在 Snowflake 中处理数据的应用程序,而无需将数据移动到存在应用程序代码的其他系统。

无论您是需要数据应用程序工作负载来为客户提供、员工还是内部业务用户构建应用程序，Snowflake 平台都允许您构建大规模数据应用程序，而不会造成运营负担。

新的 Snowflake 原生应用程序框架允许提供商通过 Snowflake Marketplace 在 Data Cloud 中轻松构建、销售和部署应用程序，从而将互联应用程序提升到一个新的水平。该框架包括遥测工具，使提供商能够轻松监控和支持其应用程序。本机应用程序部署模型是对连接和托管应用程序部署模型的补充。

数据科学

Snowflake 数据工程、数据仓库和数据湖工作负载的许多优势已在前几章中详细介绍，并在本章的前面部分进行了总结。这些特定的 Snowflake 工作负载为 Snowflake 平台上强大的数据科学工作负载奠定了基础，而 Snowflake 安全数据共享工作负载的许多独特功能增强了数据科学工作负载。因此，Snowflake 的数据云可用于加速数据科学工作负载的内部和第三方数据的收集和处理。

多个 Snowflake 数据科学合作伙伴提供了强大的工具，包括 Alteryx、Amazon (SageMaker)、Anaconda、Dataiku、DataRobot、H2O.ai 和 Microsoft (Azure ML)。Snowflake 的网站包含所有 Snowflake 技术合作伙伴（包括数据科学合作伙伴）的完整列表。除了通过 Snowflake 合作伙伴网络提供的各种数据科学工具和平台外，还有 Snowflake 的两项新功能 Snowpark 和 Streamlit。

除了通过 Snowflake 合作伙伴网络提供的各种数据科学工具和平台外，还有一项名为 Snowpark 的 Snowflake 新功能。

雪地公园

Snowpark 是一个开发人员框架，它通过扩展 Snowflake 的功能为数据云带来新的数据可编程性。Snowflake API 使开发人员、数据科学家和数据工程师能够使用他们选择的语言以无服务器方式部署代码。Snowpark 目前支持 Java、Scala 和 Python。Snowpark 的未来增强功能包括 Python 工作表，它将允许您在 Snowflake UI 中编写和运行 Python 代码，就像记事本环境一样。适用于 Python 的 Snowflake 工作表目前为个人预览版。

Snowpark 大约在一年前首次向客户发布，代表了 Snowflake 中对新可编程性选项的重大尝试。正如 Snowflake 平台所具有的

发展到 Data Cloud 后，自然而然地扩展了开发人员与平台交互的方式，因为并非每个开发人员都希望用 SQL 编写代码，也不是每个数据问题都非常适合 SQL 解决方案。

Snowpark 使数据工程师和数据科学家能够轻松创建复杂的数据管道，无需将数据移动到存在应用程序代码的系统。相反，应用程序代码的处理将直接带到数据所在的位置。Snowpark 提供了一个访问数据的单一层，并开放了 DataFrame API 的使用，该 API 提供了更高效的编码功能，尤其是大型数据集。

Snowpark 利用 Snowflake 现成的虚拟仓库，因此访问资源没有时间延迟，这意味着可以近乎实时地提供来自机器学习模型或 Web 会话分析的统计数据和计算。事实上，不需要 Snowflake 之外的集群，因为所有计算都是在 Snowflake 中完成的。由于 Snowpark 是 Snowflake 的原生版本，因此无需人工分区，这是基于文件的数据湖中经常出现的常见且困难的问题。

Snowpark 的最佳功能之一是您可以在代码中创建用户定义的函数（UDF），然后 Snowpark 可以将其推送到服务器，在那里代码将直接对数据进行操作。Snowflake 允许您在 UDF 中构建复杂的逻辑，并利用强大的外部功能。目前，Snowflake 支持 Java、JavaScript 和 Python UDF 以及外部函数。

Snowpark 库通过 Maven 作为 JAR 文件分发，可用于在许多不同的环境中开发应用程序，包括 Jupyter Notebook，以及 IntelliJ IDEA 和 Visual Studio 等应用程序开发工具。为 Snowpark 设置开发环境后，您需要使用 Snowflake 数据库创建一个会话才能使用 Snowpark 库。完成后，请务必关闭会话。

具体来说，创建会话后 Snowpark 如何工作？在开发环境中，您将编排转换并触发要在 Snowflake 计算环境中发生的操作。常见的 Snowpark 转换操作包括 `CREATE`、`ALTER` 和 `DROP`，常见操作包括 `WRITE`、`APPEND` 和 `REFRESH`。



调用某些操作（如 `APPEND` 和 `REFRESH`）会将数据下载到正在运行的客户端或驱动程序应用程序的内存中，并可能导致内存不足错误。

使用 Snowpark 库，您可以将每个 Snowflake 表作为 DataFrame 读取，该 DataFrame 会创建对该表的内存引用。可以对内存引用执行转换，但只有在执行操作后才会处理所有转换。换句话说，Snowpark DataFrame 是惰性计算的。在执行操作时，Snowpark 库会汇总所有提交的转换，同时生成等效的 SQL 查询语句。然后，此 SQL 状态将传递到 Snowflake 计算资源，并在其中执行（如图 12-4 所示）。

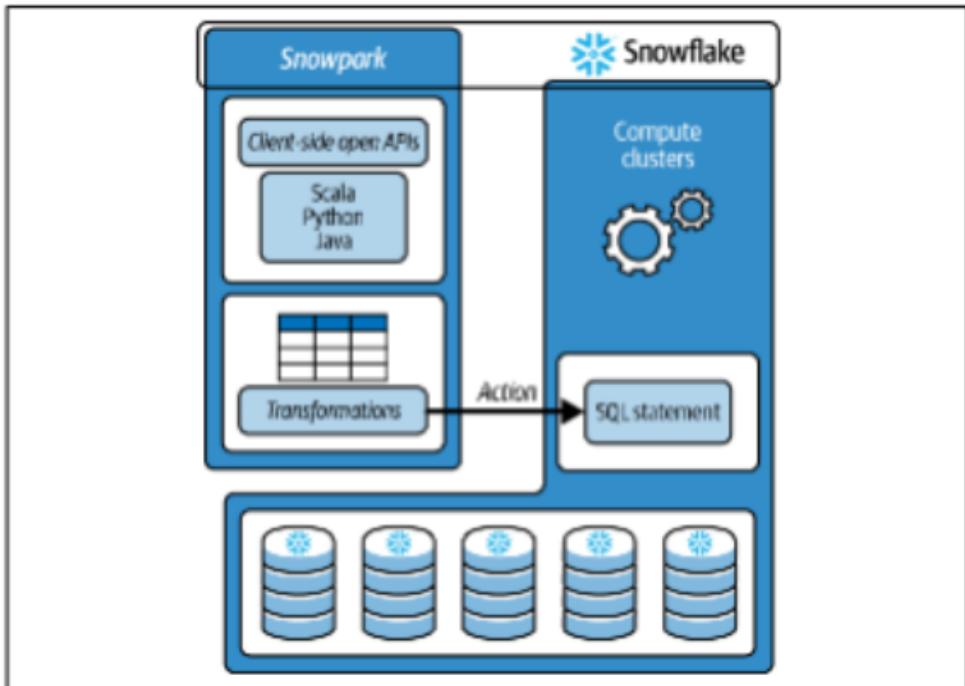


图 12-4. 一旦从 Snowpark 触发操作以将转换聚合到 SQL 查询中，就会在 Snowflake 中执行 SQL 语句



可以在 Snowflake UI 中手动测试或调试 SQL 查询。您可以在客户端或驱动程序应用程序日志中找到 SQL 语句。

前面提到过，如果您的转换代码太复杂而无法表示为 DataFrame 操作，则可以利用 Snowflake 的 UDF。UDF 将一次处理一条记录，并返回一个可以附加到 DataFrame 的新列值。



使用 Snowpark，您可以创建没有参数的 UDF。最多可以将 10 个参数作为 DataFrame 列传递。

Snowpark 可用的 UDF 参数的一个限制是，有一些数据类型（尤其是数组）无法通过 Snowpark API 获得原生支持。使用 Snowflake 变体数据类型有一些解决方法，但它们不是最佳解决方案。但是，如果您发现需要 Snowflake 的 UDF 无法轻松处理的复合编码，则还有另一种选择。正如您可以利用 Snowflake 的 UDF 一样，也可以将外部功能整合到您的 Snowpark 解决方案中。外部函数是存储在 Snowflake 外部的 UDF。

如果您想获得 Snowpark 或一些数据科学合作伙伴工具的实践经验，您可以参加 Snowflake Data Cloud Academy 的指导数据科学家学校。

Streamlit（流式）

Snowflake 于 2022 年 3 月收购了 Streamlit。Streamlit 有许多机器学习和数据科学使用案例，可用于构建复杂和动态的应用程序。例如，在 Streamlit 框架中，用户可以使用 Python 代码创建丰富的交互式可视化效果。Streamlit 的其他一些巨大好处是它结构化且简单。

Streamlit 的另一个令人兴奋的功能是，当与 Python 工作表结合使用时，Streamlit 应用程序可以直接在 Snowflake 用户界面中呈现。一旦这些 Streamlit 应用程序作为本机应用程序部署在 Snowflake 上，用户将能够通过基于角色的身份验证将这些应用程序提供给其他 Snowflake 用户，其方式与目前处理数据的方式大致相同。

使用 Snowflake 作为安全数据湖的网络安全

安全漏洞的高昂成本使网络安全成为公司高管的首要任务。其中一些成本包括收入损失、客户信任损失和知识产权损失。保持强大的安全态势至关重要且具有挑战性。首先，随着数据的指数级增长，识别和遏制违规行为需要付出巨大的努力和时间。解决此问题的传统方法大量依赖于传统的安全信息和事件管理（SIEM）工具，这些工具并非为当今的需求而设计。

仅 SIEM 架构存在许多挑战。首先，SIEM 按数量收费，这使得摄取和处理所有日志的成本高得令人望而去。不

它们不仅价格昂贵，而且 SIEM 是运行起来耗时的解决方案；因此，标识为非关键的事件通常会被排除在 SIEM 处理之外。

其次，数据孤岛给调查人员留下了不可见的差距、不完整的事件图景和失败的审计（如图 12-5 所示）。团队一次使用 10 个或更多不同的安全产品并不罕见。许多用于收集安全日志和安全数据的工具是分开和孤立的。如果没有统一的方法来管理这些工具，安全分析师经常发现自己以转椅式的方式从一个工具跳到另一个工具。这还需要手动操作，例如下载数据和创建组合报告，以及花费时间不必要的调查过多的误报。这种方法会导致可见性、速度和效率的损失。

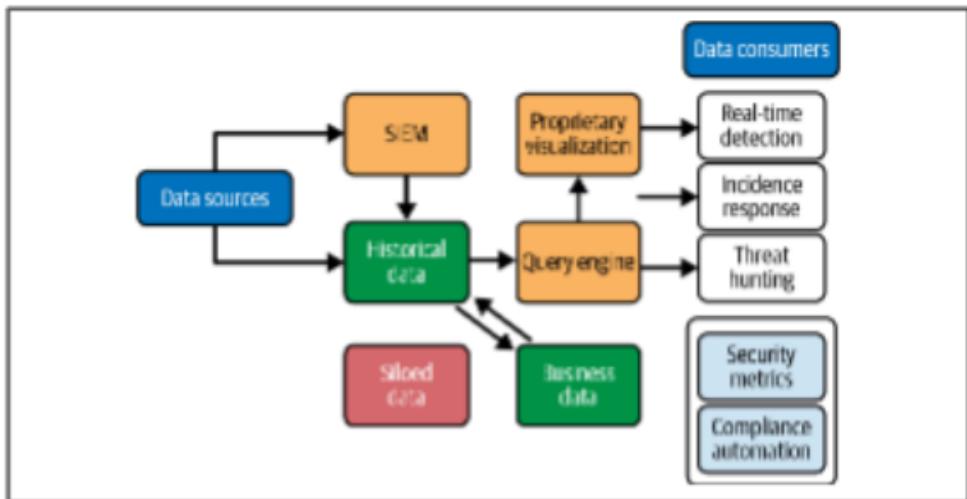


图 12-5. 传统网络安全架构

最后，SIEM 仅支持专有语言；因此，用户无法跨不同来源联接数据或使用常用语言来调查日志。因此，对网络安全分析师的需求很高，但并没有大量具备安全团队所需特定技能的人才。

克服纯 SIEM 架构的挑战

为了克服这些挑战，安全团队通过安全编排、自动化和响应（SOAR）以及扩展检测和响应（XDR）产品增强了传统的纯 SIEM 方法。SOAR 工具（有时内置于 SIEM 工具中）可简化手动修复工作。尽管如此，这些 SOAR 工具通常是需要配置和调整的附加组件和插件，SIEM 和 SOAR 工具的组合并不能解决数据孤立的问题。

最新一代的 SIEM 工具可能提供 XDR 功能，试图通过集中和规范来自多个安全层的数据来缩短威胁检测和响应时间。XDR 解决方案的目标是更好地验证警报，从而减少误报并提高可靠性。SIEM 更积极，而 XDR 是一个被动系统。SIEM 解决方案从许多来源收集浅层数据，而 XDR 从目标来源收集更深层数据。XDR 还可以为事件提供更好的上下文，并且无需手动调整或集成数据。

XDR 改善了网络安全，但仍然需要一种能够经常接受、存储和处理大量非结构化和结构化数据的解决方案。应对这些挑战的一种解决方案是数据湖方法，该方法可用于管理、共享和使用来自不同来源的不同类型的数据。XDR 数据湖进程从所有层收集数据，将数据馈送到数据湖中，对其进行消毒，然后将数据与任何渗透的攻击表面相关联（如图 12-6 所示）。这样一来，XDR-数据湖流程就能够挖掘攻击者利用安全孤岛中的漏洞时出现的模式或见解，这使他们能够在孤立的产品之间滑动，然后随着时间的推移，随着他们对网络防御的了解越来越多，并为未来的攻击做准备。

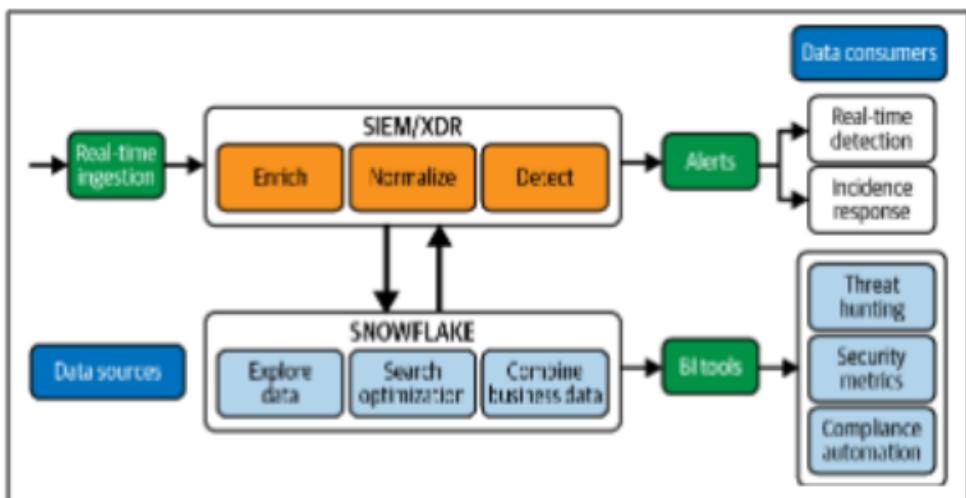


图 12-6. 现代网络安全架构

如今，许多组织正在采用安全数据湖作为其网络安全战略的基础。安全数据湖使公司能够捕获并有效地存储来自所有日志源的数据，访问所有日志源以响应事件并搜寻威胁。借助安全数据湖，可以经济高效地将多年数据存储在一个位置，与其他企业一起对安全日志运行分析。

数据源，并为合作伙伴提供对数据的无缝、受控访问。

Snowflake 是安全数据湖的自然选择。

Snowflake 独特的架构提供了许多价值主张。通过将计算与存储分离，安全分析师可以比使用 SIEM 更快地执行查询搜索和历史分析，因为 SIEM 通常需要 30 分钟或更长时间才能返回结果。使用 Snowflake 作为安全数据湖来存储无限数据，您可以打破这些数据孤岛，并且比使用 SIEM 更具可扩展性。此外，您可以在 Snowflake 中单独存储数据的时间没有限制。可以以每 TB 23 USD 的价格在单个热层中存储无限数量的日志，且保留期不受限制。

关于用户可用的语言，Snowflake 支持多种语言，包括 SQL 和 Python 等通用语言。这种支持允许使用 Snowflake 作为安全数据湖的组织利用他们已经拥有的拥有这些技能的人员来帮助完成安全任务。

另一个价值主张是，通过将 Snowflake 用作安全数据湖，可以组合传统 SIEM 或遗留系统无法组合的不同类型的数据集进行分析。传统的 SIEM 只能提取某些类型的安全日志，而 Snowflake 可以支持结构化、半结构化和非结构化数据类型。此外，Snowflake 的生态系统包括企业 BI 工具选项，这使得构建动态控制面板变得容易。相比之下，传统的 SIEM 在仅用几个不同的模板呈现结果的方式上非常有限。

需要明确的是，使用 Snowflake 作为安全数据湖的价值不仅仅是廉价的云存储和即时计算可扩展性。有了 Snowflake，就有了丰富的合作伙伴生态系统，他们提供与 Snowflake 相关的应用程序，以提供更高水平的安全功能。例如，可以通过 Hunters、Lacework 和 Panthers Lab 等 Snowflake 合作伙伴提供开箱即用的检测服务或工作流程、事件时间表等。从本质上讲，这些合作伙伴提供直观的前端用户界面，将 Snowflake 作为其网络安全战略的支柱。

使用 Snowflake 作为安全数据湖，公司可以在单个系统中全面了解其所有日志数据源，降低风险，并改善整体安全状况。Snowflake 消除了数据孤岛，并支持自动分析 IC，以帮助团队使用相关指标和动态仪表板做出更明智的决策。此外，Snowflake 不断创新，并寻找通过其搜索优化服务等功能来支持网络安全工作负载的方法。

搜索优化服务与聚类分析

Snowflake 的搜索优化服务提高了点查找查询的性能。点查找查询返回一个或几个不同的行。需要对具有高度选择性筛选器的关键仪表板进行快速响应的业务用户，以及需要大型数据集的特定子集的数据科学家经常使用点查找查询。

Snowflake 的搜索优化服务在后台运行，以收集有关用于填充搜索访问路径的列的元数据信息。搜索优化可能看起来类似于聚类分析，但存在一些关键差异。首先，为所有列启用搜索优化，而对一个键或一组键进行聚类分析。另一个区别是，搜索优化仅加快相等搜索的速度，而聚类分析同时加快范围搜索和相等搜索的速度。相等性搜索使用筛选器来识别包含给定属性的特定值的记录，而范围查询检索某个值介于上限和下限之间的所有记录。另一个重要区别是，筛选只增加了运行后台进程进行重新聚类的计算成本，而搜索优化服务则同时产生计算和存储成本。存储搜索访问路径会产生存储成本。

以下示例可帮助演示聚类分析和搜索优化之间的区别。我们将使用 Snowflake 示例数据创建表。由于我们无法克隆 Snowflake 示例数据，因此我们需要使用 Snowflake 示例数据创建一个基表，以便我们可以多次克隆该表。该表有 6 亿行，因此创建克隆需要几分钟时间：

```
创建或替换 TABLE BASETABLE 为  
SELECT * FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF100.LINEITEM;
```

首先，克隆表以用于集群示例并启用集群：

```
创建或替换表 CLUSTEREXAMPLE 克隆 BASETABLE;ALTER TABLE  
CLUSTEREXAMPLE CLUSTER BY (L_SHIPDATE);
```

首先查看聚集表，验证集群是否完整：

```
选择 SYSTEM$CLUSTERING_INFORMATION ('CLUSTEREXAMPLE', '(L_SHIPDATE)');
```

图 12-7 显示了 CLUSTEREXAMPLE 表的聚类信息。

```
As SYSTEM$CLUSTERING_INFORMATION(CLUSTEREXAMPLE,'L_SHIPDATE')

{
    "cluster_by_keys" : "LINEAR(L_SHIPDATE)",
    "total_partition_count" : 919,
    "total_constant_partition_count" : 6,
    "average_overlaps" : 8.2802,
    "average_depth" : 5.4875,
    "partition_depth_histogram" : {
        "00000" : 8,
        "00001" : 3,
        "00002" : 14,
        "00003" : 122,
        "00004" : 166,
        "00005" : 158,
        "00006" : 170,
        "00007" : 147,
        "00008" : 188,
        "00009" : 31,
        "00010" : 8,
        "00011" : 8,
        "00012" : 8,
        "00013" : 8,
        "00014" : 8,
        "00015" : 8,
        "00016" : 8
    }
}
```

图 12-7. CLUSTEREXAMPLE 表的集群信息

现在，让我们将这些结果与非聚集基表进行比较：

```
选择 SYSTEM$CLUSTERING_INFORMATION ('BASETABLE', '(L_SHIPDATE') :
```

结果 12-8 如图 8 所示。

```
As SYSTEM$CLUSTERING_INFORMATION(BASETABLE:(L_SHIPDATE))

{
    "cluster_by_keys": "LINEAR(L_SHIPDATE)",
    "total_partition_count": 986,
    "total_constant_partition_count": 2,
    "average_overlaps": 263.4746,
    "average_depth": 159.9106,
    "partition_depth_histogram": {
        "00000": 0,
        "00001": 1,
        "00002": 0,
        "00003": 0,
        "00004": 0,
        "00005": 0,
        "00006": 0,
        "00007": 0,
        "00008": 0,
        "00009": 0,
        "00010": 0,
        "00011": 0,
        "00012": 0,
        "00013": 0,
        "00014": 0,
        "00015": 0,
        "00016": 0,
        "00032": 5,
        "00064": 41,
        "00128": 234,
        "00256": 626
    }
}
```

图 12-8. BASETABLE 表的聚类信息

您会注意到，聚集表的分区数略高，但除此之外，目前聚集表和非聚集表之间没有太大差异。这是因为在后台运行的集群需要一些时间才能完成，尤其是在大型表（例如我们正在使用的表）上。稍后我们将返回并再次查看集群表。现在，让我们继续克隆表以用于搜索优化示例：

```
CREATE OR REPLACE TABLE OPTIMIZEEXAMPLE CLONE BASETABLE; ALTER TABLE
OPTIMIZEEXAMPLE ADD SEARCH OPTIMIZATION;
```

使用以下命令验证搜索优化是否已完成：

显示 '%EXAMPLE%' 之类的表：

您应该看到 OPTIMIZEEXAMPLE 表的 SEARCH_OPTIMIZATION_PROGRESS col - umn 的值现在为 100，这表示搜索优化已完成。您还会注意到，CLUSTEREXAMPLE 表的 AUTOMATIC_CLUSTERING 列值为 ON。

接下来，我们需要对原始表运行点查找查询，该表既未启用聚类分析，也未启用优化。让我们使用以下命令来获取稍后可用于点查找查询的记录示例：

```
从 BASETABLE LIMIT 1 中选择
*;
```

记下返回的记录的值，并在下一个查询中使用它：

```
SELECT * FROM BASETABLE WHERE L_ORDERKEY = '363617027' ;
```

结果 12-9 如图 9 所示。

L_ORDERKEY	L_PARTKEY	L_SUPPKEY	L_LINENUMBER	L_QUANTITY	L_EXTENDEDPRICE
1 363,617,027	5,916,396	416,407	5	40	36,484
2 363,617,027	13,148,870	146,671	2	7	13,427.54
3 363,617,027	3,846,789	96,803	1	66	80,297.6
4 363,617,027	13,046,496	46,497	3	31	44,897.04
5 363,617,027	12,441,597	941,622	4	27	41,325.19

图 12-9. 对 BASETABLE 表的点查找查询的结果

您会注意到，如图 12-10 所示，查询探查器花了 4.0 秒才完成。



图 12-10. BASETABLE 表上的点查找查询的 Query Details 视图

单击 Home 图标导航回 Main 菜单，然后单击 Activity → Query History。单击您刚刚运行的查询。您可以看到，查询必须扫描所有 906 个微分区才能获得结果（如图 12-11 所示）。使用 Snowflake 作为安全数据湖实现网络安全 |395

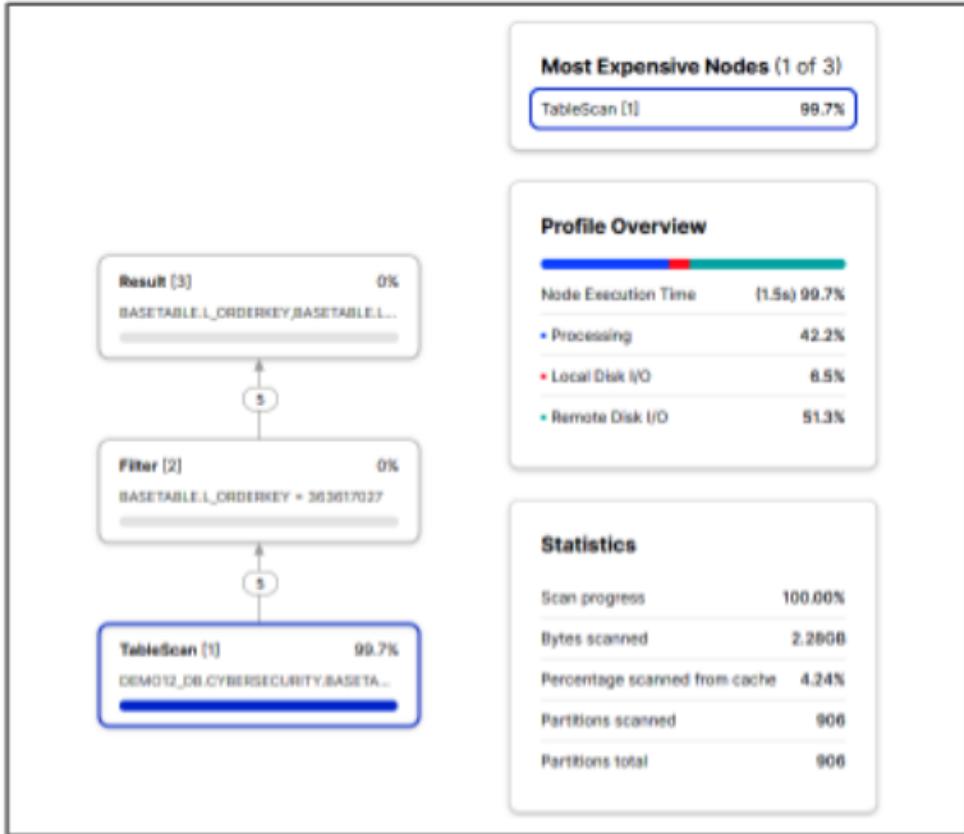


图 12-11. `BASETABLE` 表上的点查找查询的 Query Profile 视图

尝试对聚集表执行相同的查询时，我们注意到返回结果花费的时间甚至更长（如图 12-12 所示）：

```
SELECT * FROM CLUSTEREXAMPLE 其中 L_ORDERKEY = 363617027;
```



图 12-12. `CLUSTEREXAMPLE` 表上的点查找查询的 Query Details（查询详细信息）视图

这次扫描的分区也更多（如图 12-13 所示）。



图 12-13. CLUSTEREXAMPLE 表上的点查找查询的 Query Profile 视图

我们应该能够以这样一种方式重写查询，以便我们可以演示如何利用集群。让我们看看我们是否可以通过扫描存在 clustering key 的特定微分区来减少运行时间（如图 12-14 所示）。如果您查看之前收到的结果，您会注意到有一个 L_SHIPDATE 列。请务必使用搜索中返回的记录中的日期范围：

```
SELECT * FROM CLUSTEREXAMPLE  
其中 L_SHIPDATE >= '1992-12-05' 和 L_SHIPDATE <='1993-02-20' 和  
L_ORDERKEY = '363617027';
```

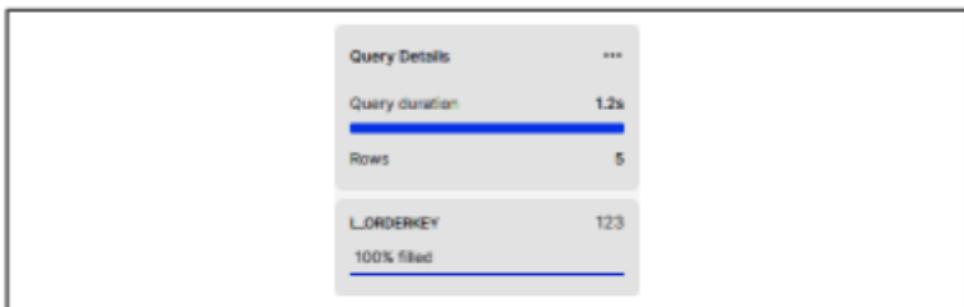


图 12-14. 使用聚类键上的特定范围对 CLUSTEREXAMPLE 表进行点查找查询的 Query Details 视图

当我们查看 query profile (查询配置文件) 时, 我们注意到只需要扫描 34 个分区即可返回结果 (如图 12-15 所示)。

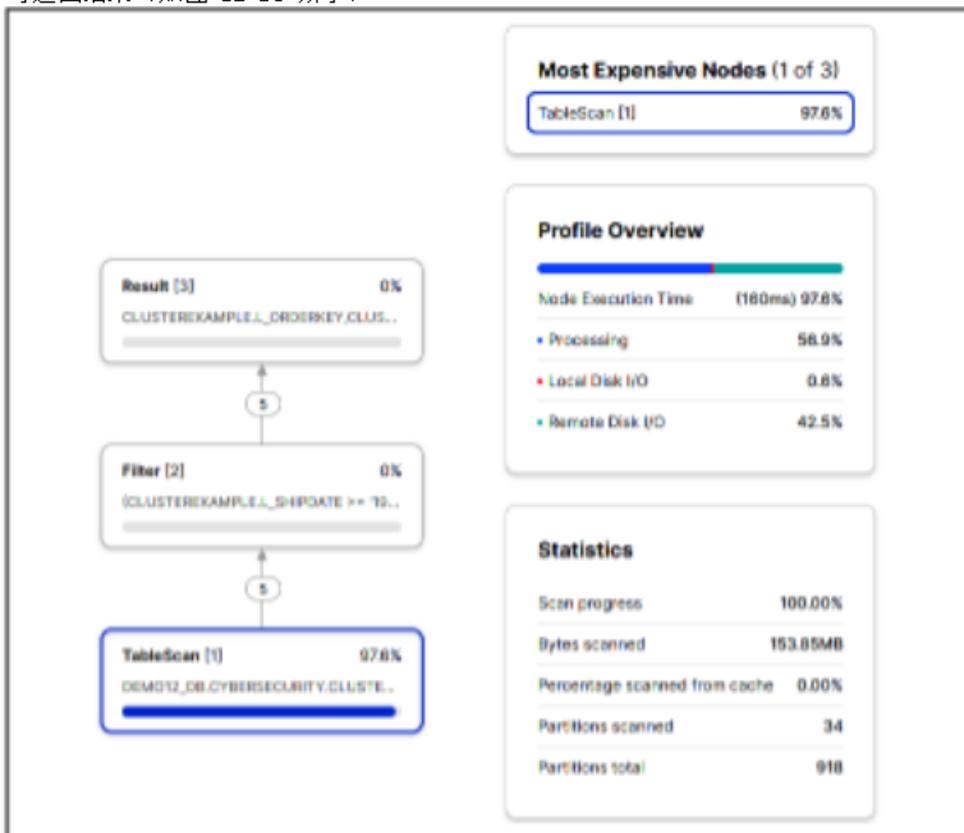


图 12-15. Query Profile (查询配置文件) 视图, 用于对 CLUSTEREXAMPLE 表使用聚类键上的特定范围的点查找查询

到目前为止，我们学到了什么？我们发现，聚类并没有提高点查找查询的性能。例外情况是，当我们事先知道聚类键的值范围时，这将使我们能够缩小扫描的 partition 的数量。实际上，我们在实践中很少可能看到异常。因此，我们可以说，作为一般规则，聚类不会提高点查找查询的性能。优化呢？

当我们尝试对启用了搜索优化功能的表进行查询时，我们将看到，启用搜索优化后，由于分析器能够使用搜索优化，因此返回结果的速度会快得多（如图 12-16 所示）：

```
SELECT * FROM OPTIMIZEEXAMPLE WHERE L_ORDERKEY = 363617027 ;
```

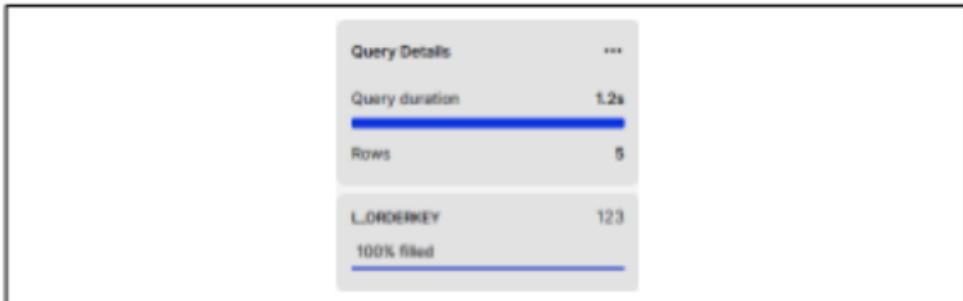


图 12-16. OPTIMIZEEXAMPLE 表上的点查找查询的 Query Details 视图

有趣的是，只需要扫描 6 个微分区即可返回结果（如图 12-17 所示）。



图 12-17. CLUSTEREXAMPLE 表上的点查找查询的 Query Profile 视图

当用于安全数据湖中的点查找查询时，Snowflake 的搜索优化服务可以通过加快工作负载来产生重大影响。此外，使用搜索优化服务可以让您在现有工作负载上使用较小的虚拟仓库，从而降低总体成本。

还记得之前我们查看 CLUSTERINGEXAMPLE 表时吗？结果请参考图 12-7。现在已经过去了一段时间，让我们再次看一下表格（如图 12-18 所示）：

```
选择 SYSTEM$CLUSTERING_INFORMATION ('CLUSTEREXAMPLE', '(L_SHIPDATE)');
```

```
As SYSTEM$CLUSTERING_INFORMATION(CLUSTEREXAMPLE@L_SHIPDATE)

{
    "cluster_by_keys" : "LINEAR(L_SHIPDATE)",
    "total_partition_count" : 918,
    "total_constant_partition_count" : 6,
    "average_overlaps" : 5.7211,
    "average_depth" : 4.1383,
    "partition_depth_histogram" : {
        "00000" : 8,
        "00001" : 3,
        "00002" : 32,
        "00003" : 290,
        "00004" : 277,
        "00005" : 180,
        "00006" : 186,
        "00007" : 22,
        "00008" : 8,
        "00009" : 8,
        "00010" : 8,
        "00011" : 8,
        "00012" : 8,
        "00013" : 8,
        "00014" : 8,
        "00015" : 8,
        "00016" : 8
    }
}
```

图 12-18. CLUSTEREXAMPLE 表的聚类信息（与图 12-7 比较）

现在，您应该会看到表的聚集方式有很大的不同。

Unistore 联卖店

Unistore 是 Snowflake 的最新工作负载，它扩展了 Snowflake 的功能，以实现一种现代方法，以便在一个单一的平台中同时处理事务和分析数据。传统上，事务和分析数据是孤立的，但借助 Unistore，Snowflake 用户现在可以包含事务性使用案例，例如应用程序状态和数据服务。Unistore 工作负载是通过 Snowflake 混合表实现的。为了更好地了解新的 Unistore 工作负载，我们首先描述一下问题。

事务性工作负载与分析工作负载

事务性工作负载和分析工作负载之间存在一些差异，如表 12-1 中所示。

表 12-1 事务性工作负载和分析工作负载之间的差异

	事务	分析
用途	运营分析 并发用户数高 低到中	响应时间 亚秒到小时 线性缩放 非线性 (复杂查询)
数据模型	规范化 列式存储	新鲜度 实时 历史 查询 表单
INSERT, DELETE, UPDATE	SELECT	
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-

缩小事务性工作负载和分析工作负载之间差距的需求越来越大。例如，零售商可以使用分析结果来识别交叉销售和追加销售机会，但前提是能够足够快地获得结果以采取行动。该问题的一种常见实施解决方案是运营数据存储（ODS），这是一种通常用作数据仓库的临时本地存储区域的数据库。虽然 ODS 确实部分解决了延迟问题，但它也增加了更多的复杂性并增加了成本。

在前面的章节中，我们了解了 Snowflake 是如何从头开始构建的，以便为分析工作负载提供可扩展的解决方案，但现在，随着新的 Unistore 工作负载的推出，该解决方案扩展到了事务性工作负载。Unistore 工作负载强制执行主键约束，而引用完整性通过 for 键约束强制执行。行级锁定、低延迟查找以及唯一和非唯一索引也是基于现有 Snowflake 框架构建的功能，用于支持 Unistore 工作负载。Snowflake 通过使用其新的混合表实现了这一新功能。

混合表

混合表使客户能够对事务数据执行快速分析，以获取即时上下文。Snowflake 混合表与其他类型的 Snowflake 表非常相似。可以将混合表与现有 Snowflake 表联接，以获得所有数据的整体视图，并且许多常见的 Snowflake 功能（如流、克隆和 UDF）都可用于混合表。由于混合表只是 Snowflake 中的另一种表类型，因此您还可以体验到与其他 Snowflake 表类型相同的强大语法以及一致的治理和安全性。

混合表的不同之处在于，它们旨在支持快速的单行操作，并为您提供出色的分析查询性能。创建 Snowflake 混合表时，您需要确保标识主键，因为主键是必需的。由于具有必需的主键，Snowflake 能够防止重复记录。重复记录预防是以前不支持的功能。



对于事务性工作负载，通常使用 Order ID 字段作为主键，并在 order date 列上创建索引，以便更快地访问最近的订单。

利用 Snowflake 的混合表，您可以将事务和分析数据整合到一个统一的平台中，而无需使用多个不同的产品。这简化并精简了企业应用程序开发，并消除了在系统之间移动数据的需要。

总结

本章介绍了 Snowflake 的工作负载，包括数据工程、数据仓库存储、数据湖、数据协作、数据分析、数据应用程序、数据科学、使用 Snowflake 作为安全数据湖的网络安全以及 Unistore。当我们逐一完成这些测试时，我们被提醒管理工作负载是多么容易，以及 Snowflake 平台如何成为当今许多数据密集型应用程序的不错选择。

我们了解了 Snowflake 独特的安全数据共享方法如何为我们带来私有和公共数据交换。我们讨论了如何通过 Snowflake Marketplace 轻松实现数据货币化，并满足不断增长的合规性和法规要求，以维护对数据的公共访问。

对于数据分析工作负载，我们认识了 Snowflake 的 BI 技术合作伙伴，并考虑了一些使用高级分析技术的特定行业解决方案。对于数据科学工作负载，我们深入研究了新的 Snowpark 开发框架。

我们还了解了 Snowflake 的新网络安全工作负载，其中还包括一些动手实践示例，以了解有关 Snowflake 搜索优化服务的更多信息以及它与集群的不同之处。

在整本书中，我们一起踏上了 Snowflake 的旅程。在此过程中，我们进行了许多动手练习，我提供了许多指导技巧和警告，以帮助指导您进一步了解 Snowflake。我希望你能超越在本书中学到的经验教训，在你的 Snowflake 之旅中继续创造令人惊叹的东西。

代码清理

本章的代码清理只需要以下命令：

```
DROP DATABASE DEMO12_DB;更改会话集  
USE_CACHED_RESULT = TRUE;
```

我们只需要删除数据库，这将导致数据库中的任何架构和表都被删除。



谨防必应您将竭尽在低期间LT
利用缓存。

知识检查

以下问题基于本章中涵盖的信息：

1. Snowflake 多久发布一次新版本？
2. Snowpark 支持哪些客户端开放 API 语言？
3. Snowflake 数据平台的一个关键特点是它对开发人员友好。
解释。
4. 设置的原因是什么 `USE_CACHED_RESULT` to ?
5. 何时适合使用聚类分析与搜索优化？
6. 搜索优化服务如何支持 Snowflake 网络安全工作负载？
7. 为什么 Data Vault 是 Snowflake 平台上常用的建模技术？

8. Data Vault 与其他数据建模方法相比如何？
9. 数据工程师是否负责数据治理？解释。
10. 数据货币化的两种常见定价策略是什么？

这些问题的答案可在附录 A 中找到。

知识检查问题的答案

第一章

1. 从 Snowflake 中创建支持案例的功能不适用于试用账户或读者账户。但是，仍然有很多方法可以获得帮助（请参阅以下答案）。

2. 如果您无法在 Snowflake 中创建支持案例，则您有几个

选项：

- 查阅 Snowflake 文档。
- 查阅 Snowflake 知识库。
- 向 Snowflake 社区寻求支持。
- 查看 Stack Overflow 问答。
- 获得直接支持（激活、密码、计费和取消）
仅限请求）从 Snowflake 社区中：

1. 登录 Snowflake Data Heroes 社区。
2. 单击“提交案例”快速链接。
3. 选择账户类型“我是 30 天试用用户”。

3. 为 Snowflake 工作表设置上下文意味着您需要让 Snowflake 知道您要在工作表中使用哪个角色、虚拟仓库和数据库。您可以通过下拉菜单或工作表中的语句（即，）设置上下文。

4. 下拉菜单中的“格式化查询”选项使您可以整齐地格式化 SQL 代码。它不会更正拼写错误。

5. Snowflake 认证的有效期为两年，之后必须通过资格重新认证考试。请注意，通过任何高级 Snowflake 认证都会重置 SnowPro 核心认证的时钟，因此您还有两年时间需要重新认证。
6. 您可以通过两种方式在 Snowflake 工作表中执行 SQL 查询：
 - 将光标置于查询的开头，然后单击蓝色箭头（“运行”按钮）。
 - 突出显示所有语句，然后按 `Ctrl + Enter` 键，而不是单击“运行”按钮。
7. 屏幕左上角的 House 图标将带您返回主菜单。
8. 我们将使用所有大写字母来命名 Snowflake 对象，因为 Snowflake 无论如何都会将这些名称转换为大写，即使它们最初以小写或混合大小写形式给出。例外情况是，如果将对象名称括在引号中，在这种情况下，对象名称不会自动转换为大写。相反，它将完全按照您在引号之间输入的方式保存。请注意，如果您在创建对象时选择将对象名称括在引号中，则还需要在将来的 SQL 语句中使用引号引用对象名称。
9. 在本书中，我们使用语法，以便您始终可以返回到本章中的任何一点并从该点开始，而不必先删除对象。但是请注意，这是您不想在生产环境中使用的做法，因为使用该语句将导致您最终覆盖对象。如果您在生产中错误地使用了语法，则可以使用 Time Travel 功能来恢复对象。
10. 首次创建工作表时，名称默认为当前日期和时间。

第 2 章

1. 三个 Snowflake 架构层是云服务层、查询处理（虚拟仓库）计算层和集中式（混合列式）数据库存储层。
2. 云服务层和集中式（混合列式）数据库存储层是多租户的。查询处理（虚拟仓库）计算层不是多租户的。请注意，Snowflake 是一项多租户服务，而云对象存储是一项多租户服务。因此，数据在公共云级别并没有真正隔离。正是加密创建了隔离。Snowflake 的混合动力车

在公有云级别，租户策略使用多租户表来整合存储，但为每个租户分配专用计算资源。

3. 虚拟仓库缓存位于计算层中。结果缓存位于云服务层中。元数据存储缓存层位于云服务层中。

4. 如果您的

Snowflake 云服务的预期成本：

- 访问会话信息或使用会话变量的简单查询
- 具有许多联接的大型复杂查询
- 单行插件，与批量或批量加载相比
- 频繁使用 `INFORMATION_SCHEMA` 命令
- 频繁使用仅元数据命令，例如命令 5。纵向扩展是将虚拟仓库的大小手动调整为更大或更小的大小，通常是为了提高查询性能和处理大型工作负载。横向扩展是增加和减少计算集群数量的自动过程，更常用于最大化并发性。

扩展是使用多集群虚拟仓库实现的，如果用户和/或查询的数量趋于波动，它可以自动扩展。

6. 计算可以扩展、缩减、缩减或扩展。在所有情况下，对使用的存储空间都没有影响。

7. 可扩展性问题是建筑学难以解决的主要挑战。平台架构需要可扩展，以支持与数据驱动型团队（无论大小、靠近或远离数据）共享相同的数据。

8. 选择 `Auto-scale` 模式时，扩展策略选项为 `Standard`

或经济性：

- 使用标准扩展策略时，当查询排队时，或者 Snowflake 系统检测到有多个查询超过当前正在运行的集群可以执行的查询，第一个虚拟仓库会立即启动。

- 使用经济扩展策略时，仅当 Snowflake 系统估计查询负载可以使虚拟仓库繁忙至少 6 分钟时，虚拟仓库才会启动。Economy 扩展策略的目标是通过保持虚拟仓库满载来节省积分。

9. 您需要为多集群虚拟配置以下组件
仓库：

- Mode - 自动缩放; 可以设置为 Standard 或 Economy
- 最大化; 当 Min Clusters 值大于 1 且 Min Clusters 和 Max Clusters 值相等时最大化 • Min Clusters • Max Clusters 10。您可以通过下拉菜单更改虚拟仓库。您可以在工作表中使用 USE WAREHOUSE SQL 命令。

第 3 章

1. 我们可以创建两种主要类型的数据库：永久（持久）数据库（默认）和临时数据库。我们可以创建两种主要类型的模式：个人（持久）模式（如果在永久数据库中，则为默认）和瞬态模式（如果在临时数据库中，则为默认）。我们可以创建四种主要类型的表：永久（持久）表（如果在永久架构中，则为默认）、瞬态表（如果在瞬态架构中，则为默认）、临时表和外部表。
2. 标量 UDF 返回单个值，而表格 UDF 返回多个值。
3. 与用户定义的函数（UDF）不同，存储过程可用于按表单数据库操作，例如 和 。
4. 如果我们使用该命令并且要创建的数据库已经存在，我们将收到一条错误消息。如果数据库已存在并且我们使用 optional keywords，则不会返回错误。相反，现有数据库将被完全覆盖。如果您不想出现错误，但也不想覆盖现有数据库，可以使用

NOT EXISTS 可选语句。

5. 数据库的默认数据保留时间为 1 天。我们可以将永久数据库的保留时间更改为最多 90 天，前提是我们在 Enterprise 或更高版本的 Snowflake 版本。但是，我们无法更改默认保留时间；默认保留时间将始终为 1 天。
6. 该命令从表中删除数据，而 TABLE 命令删除实际表本身。请注意， 和

DELETE 命令从表中删除数据，但不会删除表对象本身。该命令清除表加载历史记录元数据，而 DELETE 命令保留元数据。因此，使用该命令允许将外部文件或阶段中的文件再次加载到表中。

7. 常规视图会产生计算成本，但不会产生存储成本。具体化视图会产生计算成本和存储成本。具体化视图是从查询规范派生并存储以供以后使用的预计算数据集，因此会产生存储成本。具体化视图是一项无服务器功能，利用 Snowflake 托管的计算服务。
8. 完全限定的对象名称包括完全标识对象所需的所有对象，所有对象都用句点分隔。我们希望对 table 使用完全限定名称的一个示例是

`name>` 的。。允许使用部分限定的对象名称，而不标识所有限定符，但您应该了解如何解析部分限定对象。省略数据库名称时，将使用当前数据库扩充对象。使用单个标识符时，非限定对象的解析方式取决于对象是出现在数据定义语言（DDL）或数据操作语言（DML）语句中，还是出现在查询中。更多详细信息可以在 Snowflake 文档中找到。

9. 阶段的默认文件格式为 CSV。但是，您可以创建其他格式的文件格式，例如 JSON、Avro、ORC、Parquet 和 XML。
10. SNOWFLAKE 数据库归 Snowflake Inc. 所有，是一个系统定义的只读共享数据库，可提供有关您账户的对象元数据和使用情况指标。与设置时 SNOWFLAKE_SAMPLE_DATA 数据库不同，SNOWFLAKE 数据库无法从您的帐户中删除。
11. 有两种方法可以触发 Snowflake 任务：任务可以按创建任务时定义的计划运行，或者可以建立任务依赖关系，从而可以由前置任务触发任务。请注意，没有可以触发任务的事件源。不是由前置任务触发的任务必须按计划运行。
12. METADATA\$ACTION 指定 or 操作。METADATA \$ISUPDATE 指定 or 操作是否是应用于源表或视图中的行的一部分。METADATA\$ROW_ID 指定行的唯一且不可变的 ID，该 ID 可用于跟踪特定行随时间的变化。
13. Snowflake 流是处理不断变化的数据集的有效方法。在 Snowflake 中，使用表流的最重要原因之一是保持暂存表和生产表同步。将暂存表与流一起使用有助于防止对 production 表进行不需要的更改。Snowflake 表流还经常与其他功能结合使用，例如 Snowflake 管道和 Snowflake 任务。

第 4 章

- 以下内容可用于使一行文本成为注释，而不是视为代码：

- 命令
- /* <推荐出代码> */
- <注释掉了代码>

请注意，您可以通过高亮显示多行，然后按 Cmd + / (macOS) 或 Ctrl + / (Windows) 来注释或取消注释多行。突出显示的行被注释掉。

- Snowflake 的字符串数据类型由字符串常量支持，这些常量始终括在分隔符之间，可以是单引号，也可以是美元符号。当字符串包含许多引号字符时，使用美元符号作为分隔符特别有用。
- 外部函数使使用现有的机器学习服务从图像中提取文本，或处理 PDF 文件以提取键值对成为可能。在外部函数中，您可以使用任何 AWS、Azure 或 GCP 功能，例如 AWS Rekognition 或 Azure Cognitive Services。对非结构化数据执行的外部函数，无论是存储在内部阶段还是外部阶段，都可用于消除导出和重新导入数据的需要。
- 长时间运行的查询的默认持续时间为两天。您可以在账户、会话、对象或虚拟仓库级别使用 STATEMENT_TIMEOUT_IN_SECONDS duration 值。
- 由于浮点数据类型的不精确性，浮点运算可能会有小的舍入误差，并且这些误差可能会累积，尤其是在使用聚合函数处理大量行时。
- Snowflake 窗口函数是一种特殊类型的聚合函数，可以对行的子集进行操作。此相关行的子集称为窗口。与一组行返回单个值的聚合函数不同，窗口函数将为每个输入行返回一个输出行。输出不仅取决于传递给函数的单个行，还取决于传递给函数的窗口中其他行的值。窗口函数通常仅用于查找同比百分比变化、移动平均值、运行或累积总计，并按分组或自定义标准对行进行排名。
- 是的，Snowflake 支持非结构化数据类型。

8. Snowflake 支持以下半结构化数据类型：
 - 变体
 - 对象
 - 数组
9. Snowflake 的数据类型支持本地时区。TIMESTAMP_LTZ 是具有指定精度的内部 UTC 时间;它是具有本地时区的。不支持夏令时。TIMESTAMP_TZ 值是根据 UTC 时间进行对比的，UTC 时间不考虑夏令时。这一点很重要，因为在创建时，TIMESTAMP_TZ 存储的是给定时区的偏移量，而不是实际时区。
10. 派生列（有时称为计算列值或虚拟列值）并不以物理方式存储在表中，而是在每次查询中引用它们时都会重新计算。派生列可用于计算另一个派生列，可由外部查询使用，或可用作子句的一部分。
11. 以下是您可以访问 非结构化数据文件的三种方式

雪花：

- 舞台文件 URL
 - 范围限定的 URL
 - 预签名 URL
12. 非结构化数据类型的示例包括视频、音频或图像文件、日志文件、传感器数据和社交媒体帖子。非结构化数据可以是人类生成的，也可以是机器生成的，具有内部结构，但不能以结构化数据库格式存储。
 13. Snowflake 目录表是内置的只读表。

第 5 章

1. 固有权限是授予已分配角色的每个用户的权限。更具体地说，某些系统定义的角色具有 `inheritent` 权限。例如，资源监视器的权限是 `ACCOUNTADMIN` 角色所固有的。请注意，除了角色固有的权限外，还可以分配和嵌入权限。
2. 以下是 Snowflake 系统定义的账户角色：
 - `ORGADMIN`
 - 帐户管理员
 - 系统管理员

- SECURITYADMIN • 用户管理员 • 公共

大多数自定义角色应分配给 SYSADMIN 角色。

3. 为用户添加默认值会让用户更轻松，但添加默认值并不会验证是否已实际向用户授予该权限。
4. 只有 ACCOUNTADMIN 可以创建新的资源监视器。
5. 要能够查看表中的数据，角色需要具有使用表所在的数据库和架构的权限，以及能够对表使用命令的权限。
6. 默认情况下，只有 ACCOUNTADMIN 角色有权访问 SNOWFLAKE 数据库。
7. 默认情况下，Snowflake 组织中的账户数量上限不能超过 25 个；但是，您可以联系 Snowflake 支持来提高限制。
8. 可以将具有授权选项的权限授予其他系统定义或自定义角色。
9. 该命令根据执行命令的用户的角色返回结果，因此，例如，具有不同角色的用户如果可以访问不同的虚拟仓库或不同的数据库，则可能会获得不同的结果。
10. Snowflake 没有超级用户或超级角色的概念。对安全对象的所有访问，即使是由帐户管理员访问，都需要通过处于更高层次结构的角色中明确或隐含地授予访问权限。

第 6 章

1. 以下是用于半结构化数据的三种 Snowflake 数据类型：

- 变体
- 对象
- 数组

VARIANT 是通用数据类型。

2. Snowflake 阶段的类型如下：
 - 内部阶段：用户阶段、表阶段和内部命名阶段
 - 外部阶段

内部命名阶段和外部阶段是数据库对象。用户阶段和表阶段不能更改或删除，因此不是单独的数据库对象。

3. 该命令可用于无条件或有条件多表插入。
4. 当键值对包含 null 值时，不会插入键值对。
5. 以下是使用“加载数据”向导时的错误处理选项：
 - 不要在文件中加载任何数据。
 - 停止加载、回滚并返回错误（默认）。
 - 如果错误计数超过，则不要在文件中加载任何数据。
 - 继续从文件中加载有效数据。
6. 使用该命令时，可以执行基本转换，例如使用命令对列重新排序或执行强制转换。
7. Snowflake 支持以下半结构化数据类型：
 - 加载数据类型：JSON、Parquet、XML、Avro、ORC（半结构化）和 CSV/TSV（结构化）
 - 卸载数据类型：JSON 和 Parquet（半结构化）以及 CSV/TSV（结构化）
8. 以下有关 Snowpipe REST 的详细信息：
 - 可用于内部和外部阶段
 - 使用管道名称和文件名列表手动调用 Snowpipe REST API 终端节点
 - 在阶段位置传递文件列表
 - 数据随机到达和/或预处理需要使用 ETL 或 ELT 工具的使用案例的最佳选择，或者在外部

阶段不可用以下是有关 Snowpipe AUTO_INGEST（更具可扩展性的方法）的详细信息：

- 仅适用于外部载物台
- 当新文件到达时，会收到来自云提供商的通知
- 唤醒时处理新文件
- 文件连续到达的用例的最佳选择
9. 为防止资源争用，请务必通过为每个作业指定单独的虚拟仓库来将数据加载作业与查询隔离开来。而不是假设一个大型的

虚拟仓库加载大量数据文件的速度必然会比小型虚拟仓库快（可能不会），请务必尝试将大文件拆分为大小约为 100 到 250 MB 的小文件。请记住，正在加载的文件数和每个文件的大小对性能的影响大于对虚拟仓库大小的影响。

10. 该命令将数据加载到一个阶段中（然后该命令将数据从阶段加载到表中）。该命令从阶段中卸载数据（在命令将数据从表卸载到阶段之后）。

第 7 章

1. 多重身份验证（MFA）旨在与强密码一起使用。Snowflake 为所有账户级别提供自助式 MFA，以便用户可以自行注册。
2. 可以存在的账户网络策略数量没有限制，但一次只能激活一个账户级网络策略。可以存在的用户网络策略数量也没有限制，但每个用户一次只能激活一个用户级网络策略。
3. Snowflake 的根密钥位于硬件安全模块中，是唯一以明文形式存储的密钥。
4. 数据脱敏策略包括动态脱敏、条件脱敏和静态脱敏。数据掩码是基于列的，而行访问策略提供动态的行级安全性。请注意，一个列不能同时具有掩码 policy 和行访问策略。
5. 系统会自动为所有 Snowflake 账户启用按时间旅行保留期，默认值为 24 小时，但可以在账户和对象级别设置为零。对于 Snowflake Enterprise Edition 和更高级别的组织，永久数据库、架构和表的保留期最多可以设置为 90 天。

通过将数据库的保留时间设置为 90 天，所有数据库对象也将具有 90 天的保留期限。

在架构级别更改数据保留时间将导致架构中的所有表都继承架构的保留期，除非明确为表指定了不同的保留期。您应该记住的另一件事是，如果存在差异，您删除对象的顺序确实会影响保留期。删除架构时，所有现有表都将在与架构相同的时间段内可用。如果要确保子对象的数据保留期得到遵守，则需要在删除父对象之前删除它们。

6. 标签不需要用引号括起来。如果将标记标识符括在双引号中，则它区分大小写，并且可以包含空格。如果标识符未加引号，则它不能区分大小写，不能包含空格，并且必须以字母或下划线开头。

Snowflake 的自动数据标记可以有两种不同的类型：

- 语义
 - 隐私
7. 每当有嵌套的数据层对象时，我们都需要展平查询输出。这将允许我们将嵌套的数据层对象转换为只有一层键值对的新对象。

在本章中，我们使用该函数展平 Snowflake ACCESS_HISTORY 视图中的 DIRECT_OBJECTS_ACCESSED 和 BASE_OBJECTS_ACCESSED 列。

8. 克隆的 Snowflake 对象以物理方式（而不是逻辑方式）复制到第二个数据库。这意味着复制的任何克隆对象都将产生额外的数据存储成本。
9. 我们可以使用 Time Travel 访问历史数据的具体方式是 /

以前，/以前 和。

10. 时间旅行和故障安全以及主动存储是产生成本的总计算存储的一部分。复制和故障转移/故障恢复费用包括数据传输成本和计算资源成本。对于克隆对象的复制，还将产生额外的数据存储成本。

第八章

1. Snowflake 月度账单的三大类别是存储费用、数据传输成本和消耗的积分。
2. 虽然存储定价是为您的 Snowflake 账户选择哪个区域的考虑因素，但您还需要考虑哪个区域可以最大限度地减少延迟，并可以让您访问可能需要的任何必需功能。请务必注意，如果您稍后决定将数据移动到其他区域，则会产生数据传输成本。
3. 按需 Snowflake 积分的价格取决于云提供商、区域和服务版本。或者，您可以以协商的折扣率预购积分。

4. Snowflake 资源监视器的三个属性是积分配额、积分使用情况和触发器。创建资源监控器都需要 Credit quota 和 Credit usage 属性。如果未明确说明计划，则默认计划适用。默认情况下，积分配额在每个日历月开始时重置。设置 triggers 是可选的。
5. 可以创建的虚拟仓库级资源监控器的数量没有限制，但只能有一个账户级资源监控器。
6. 每个资源监控器最多只能执行 5 个 Notify 操作、1 个 Suspend 操作和 1 个 Suspend Immediately 操作。
7. 对数据库进行更改会带来一系列独特的挑战，尤其是对于传统的本地数据库，这些数据库必须处理硬件停机时间的问题，并且必须明确地提前备份数据库，以便在需要时可以恢复它们。如果需要回滚，则操作会冻结，并且在恢复过程中不需要执行任何工作。此外，还存在漂移问题，即开发人员可能正在使用与生产版本不同的数据库版本，并且还存在负载平衡问题。有了 Snowflake，这些顶级挑战对用户来说不再存在。通过为小型部署创建轻量级 DevOps 框架，可以在 Snowflake 中非常简单地处理数据库变更管理流程。对于更重量级的 Snowflake DevOps 框架，可以使用数据库更改管理工具。
8. 零拷贝克隆是仅元数据的操作；因此，零副本克隆提供了复制对象而无需实际创建物理副本的功能。因此，除非进行更改，否则 Snowflake 克隆对象不会产生额外的存储费用。

零拷贝克隆经常用于支持在开发和测试环境中工作，作为开发生命周期的一部分。

9. 使用 CREATE TABLE LIKE 命令，而 CREATE TABLE AS SELECT 命令将创建一个填充的表。
10. 将虚拟仓库资源监控器分配给已分配给资源监控器的虚拟仓库将覆盖之前的分配。

第 9 章

1. 这 QUERY_HISTORY_BY_USER 函数缩小了查询历史记录的范围。QUERY_HISTORY_BY_SESSION and QUERY_HISTORY_BY_WAREHOUSE 也可用能够使用。
2. Snowflake 函数不是加密哈希函数。相反，它是一个实用程序函数，用于返回查询描述等信息。

3. 关系数据库的两种传统分区方法是水平和垂直。
4. 所有 Snowflake 数据都存储在数据库表中。从逻辑上讲，表的结构为行和列的集合。这种行和列的逻辑结构映射到 Snowflake 物理表结构（称为微分区）。

Snowflake 独特的微分区方法对用户是透明的，这与传统的数据仓库分区方法不同，在传统的数据仓库分区方法中，用户必须使用 DDL 命令独立操作分区。

Snowflake 微分区本身就很小，存储为一组列叙的行，其中包含大约 16 MB 的压缩数据。以这种方式存储数据，其中列独立存储在微分区中，允许高效的列扫描和筛选。这对于非常大的表尤其有益。使用微分区可以对大型表进行精细修剪。有效且高效的修剪非常重要，因为扫描整个表对于大多数查询来说并不是最佳选择。

5. DML 操作（如更新）将添加新的分区块并删除现有的分区块。在 Snowflake 元数据中跟踪微分区的添加和删除。
6. Snowflake 可以通过考虑重叠的分区数来确定表的聚集程度。微分区可以在分区中彼此重叠；重叠的数量称为聚类宽度。微分区可以在分区中的特定点重叠，该特定点的重叠数决定了聚类深度。

归根结底，聚类良好的表的最佳指标是其查询性能。

7. 要查看表中的聚类深度，请使用 `SYSTEM$CLUSTERING_INFORMATION` 功能。如果表已聚集，则无需指定列。如果尚未进行聚类，则需要确保在语句中包含列。
8. 在决定聚类键时，一个好的经验法则是选择一个键，该键具有足够的非重复值以进行有效修剪，但又不能选择太多值，以便 Snowflake 仍然能够有效地对同一微分区中的行进行分组。
9. 创建集群键时，建议您选择不超过 3 或 4 列或表达式。虽然列数是一个重要的考虑因素，但最重要的是专注于为聚类键选择正确的列 `unn` 或表达式，并将它们按最佳顺序排列。
10. 物化视图和搜索优化服务是无服务器的。表显示会产生计算成本，并且不是一项无服务器功能。

第十章

1. 直接共享、数据交换、全球市场和数据净室 2. 与大多数传统数据库不同，Snowflake 支持跨数据库联接，包括与从入站共享构建的数据库的联接。这意味着除了拥有读者帐户的消费者之外，消费者还可以在自己的数据库中创建视图，该视图可以将自己的数据与共享数据库中的数据相结合。

这提供了巨大的好处，因为消费者可以丰富自己的数据以使其更有价值。

3. 入站共享只能包含一个数据库，但创建者可以使用安全视图共享来自多个数据库的数据。Snowflake 支持者可以创建一个安全视图，该视图引用多个数据库中的架构、表和其他视图，只要这些数据库属于同一账户即可。

4. Snowflake Marketplace 和 Data Exchange 都利用 Snowflake 安全数据共享将数据提供商与消费者联系起来。Marketplace 是一个公共的全球市场，可促进第三方之间的共享。Data Exchange 是您自己的私有中心，用于与您邀请的选定成员组安全地围绕数据进行协作。

5. 默认情况下，ACCOUNTADMIN 具有创建和维护数据共享的固有权限。

6. 仅评估数据提供商与数据共享相关的数据存储费用。

7. 数据使用者支付查询数据共享所产生的费用。每当使用 reader 账户查询数据共享时，费用都由数据提供商支付。

8. Snowgrid 是一个术语，用于描述数据复制，以便于数据共享。借助安全数据共享和 Snowgrid 的强大功能，Snowflake 提供商可以创建任意数量的数据共享，并与全球数据使用者共享。Snowgrid 遍布全球，无缝连接可能按区域或云分隔的 Snowflake 用户。Snowgrid 通过自动执行进行复制来实现这一点。即使数据在云和区域之间共享，共享在事务上也是一致的，这意味着事实来源仍然得到维护。

9. Snowflake 允许您共享以下对象：

- 数据库
- 表
- 外部表

- 安全视图 • 安全具体化视图 • 安全 UDF

最佳实践包括以下内容：

- 建议为每个列表创建单独的架构，但这不是必需的。创建单独架构的原因是，这可以减少混淆和意外共享对象的可能性。

- 为确保共享数据库中的敏感数据不会暴露给使用者帐户中的用户，强烈建议您共享安全视图或安全 UDF，而不是直接共享表。
- 共享来自大型表的数据时，建议您作为提供者在表上定义聚类键。

10. 必须满足以下要求才能获得批准上市 Snowflake Marketplace 的 Marketplace：
 - 数据必须是最新的和非静态的。
 - 数据不能仅包含样本的模拟数据。
 - 数据不能包含私有数据。
 - 您必须拥有提供数据的权利。

第十一章

1. 您可以指示 Snowflake 通过以下方法：
 - 基于特定行数的固定大小采样（伯努利采样）
 - 基于表行百分比的基于分数的采样：
 - 伯努利采样（默认）
 - 系统采样 可用于 Snowsight 可视化的记录数存在限制。自动统计可视化以及图表和功能板的行数必须小于 100 万。因此，通常无法在 Snowsight 中可视化完整的数据集。在这些情况下，获取要可视化的数据样本非常重要。
2. 不需要 SQL 查询来预览数据。您只需单击 Magnifying Glass 图标即可预览数据。预览数据时，我们会看到前 100 行。我们不需要活动的虚拟仓库来预览数据。

3. 当您重新运行使用该子句的查询时，您将得到第二次返回的相同行。但是，当您重新运行采样查询时，第二次返回的行数不会与第一次返回的行数相同。

示例查询结果是唯一的，因为结果不会缓存。

4. 运行抽样查询需要活动虚拟仓库，但使用放大镜预览数据不需要。

5. 您可以通过以下方式在 Snowsight 中存储数据：

- 没有
- 日期
- 第二
- 分钟
- 小时
- 周
- 月
- 季度
- 年
- 通过过滤器

6. 从仪表板中删除磁贴将永久删除基础工作表。执行删除操作后，您将无法检索磁贴或工作表。

7. 使用 Snowsight 可视化的一些使用案例包括：

- 当个人或小组用户需要进行临时数据分析时
- 在加载数据时执行数据验证
- 快速创建可以作为团队共享的次要图表和控制面板

8. 删除磁贴的一种替代方法是简单地取消放置磁贴，使其仍然存在，但无法在仪表板上查看。

9. 您可以从一个工作表创建无限数量的图表。

Knowledge Check 问题的答案 | 419 10. Snowflake 允许从内部数据源和外部数据源进行可视化。

第十二章

1. 每周，Snowflake 都会部署两个计划/预定的版本，它们可能会包括以下内容：
 - 完整版本（在一周中的任何一天部署，周五除外）— 新功能 — 功能增强或新
 - 修复 • 补丁发布 每个月，Snowflake 都会部署一个行为更改版本。
2. Snowpark 支持以下客户端开放 API 语言：
 - Node.js
 - 网
 - Go
 - 爪哇岛
 - 蟒
 - SQL
3. 据说 Snowflake 数据平台对开发人员友好，因为它允许应用程序开发人员在多种编程语言中进行选择。
4. 我们~~设置~~我们不想将缓存结果用于我们的示例，因为我们正在测试集群和搜索优化服务的查询性能。如果我们没有将值更改为 false，那么如果我们重新运行查询两次或更多次，我们将获得准确的性能测试结果。
5. 对于点查找查询，使用搜索优化而不是聚类更合适。关于聚类，Snowflake 通常会在表中生成聚类良好的数据。但是，随着时间的推移，特别是当 DML 发生在非常大的表上时，某些表行中的数据可能不再在所需的维度上以最佳方式聚集。此外，对于排序不理想的非常大的表，可能有更好的方法可以将数据放在相同的微分区中。在这两种情况下，Snowflake 的集群都可以提供帮助。
6. 网络安全工作负载通常涉及需要快速响应关键仪表板的团队，这些仪表板具有高度选择性的过滤器，这通常需要点查找查询。因此，搜索优化服务（对于点查找查询非常有用）可用于支持 Snowflake 网络安全工作负载。

7. Data Vault 建模方法可以利用 Snowflake 的大规模并行处理计算集群和优化的列式存储格式。
 8. Data Vault 方法和 Kimball 方法相对简单，与 Inmon 方法相比，实施时间更少。Kimball 方法基于四步规范化过程，而 Data Vault 使用敏捷方法。一般来说，实施 Data Vault 建模方法的成本最低，但 Data Vault 确实需要更多存储空间，当数据简单明了且不需要审计数据时，Data Vault 并不是最佳选择。
 9. 数据工程师不负责制定治理规则和法规，但他们的任务是实施它们。
-
10. 数据货币化的两种常见定价策略是成本定价和价值定价。

Snowflake 对象命名最佳实践

一般（角色相关）

- 创建简短而有意义的名称。尽可能使用缩写，尤其是在长度超过 8 个字符时。尽管名称最多可以包含 128 个字符，但最好不要超过 24 个字符。与其在名称中使用更多字符，不如考虑使用注释或标签作为描述符。
- 避免混合大小写的字母。如果您选择使用全大写或全小写字母以外的其他字母，则需要在名称周围加上引号，因为如果没有引号，Snowflake 会将名称转换为全大写。
- 避免空白。请改用下划线（_）。否则，这可能会导致某些第三方应用程序出现问题。
- 对表和字段使用单数术语。例如，Customer 表可以具有 CustomerID 字段。
- 避免使用特殊字符，如 #、-、@ 和！。尽管 Snowflake 允许使用某些应用程序，但它们可能会导致某些第三方应用程序出现问题，因此最好避免使用它们。

一般（与角色无关）

- 使用建议的 Snowflake 系统定义角色创建对象。例如，使用 SYSADMIN 角色创建数据库和虚拟仓库。有关详细信息，请参阅第 5 章中的图 12-5。

¹ 为简单起见，有时我们不会使用推荐的 Snowflake 系统定义角色来创建对象。在这些示例中，我将提到我们使用的是与推荐角色不同的角色。

- 在创建新对象时使用语法。
 - 对语言进行标准化，选择地区或方言，例如美式英语。
例如，使用 COLOR 而不是 COLOUR。
 - 避免介词。例如，使用 ERROR_CODE 而不是 CODE_FOR_ERROR。
 - 避免使用后置形容词。例如，使用 APPROVED_ARTICLES 而不是 ARTICLES_APPROVED。
-
- 避免使用通用短语，例如 OTHER 和 MISC。
 - 避免使用表和列的前缀。请谨慎使用下划线。
 - 避免使用与其他对象类型使用的名称相同的名称。例如，不要为同一架构中的临时表和永久表指定相同的名称。临时表是基于会话的；因此，他们不受唯一性要求的约束。即使您可以创建一个与另一个表同名的临时表，但这样做并不是一个好的做法，因为它可能会导致不必要的后果。
 - 避免保留字。Snowflake 文档中提供了保留关键字的列表。

标准标签缩写

DEV

用于数据库和角色的开发环境（前缀）

EXT

用于阶段和表的外部（前缀）

FB

用于数据库和角色的功能分支环境（前缀）

FF

文件格式（前缀）

POC

用于数据库和角色的概念验证环境（前缀）

2 对于章节练习，我们使用语法 CREATE OR REPLACE 而不是 CREATE IF NOT EXISTS。我们使用 OR REPLACE 语法，以便您始终可以返回到本章练习的开头重新开始，而不必先删除对象。但是，在实践中，请务必使用 IF NOT EXISTS 语法，尤其是在生产环境中。如果您在生产中错误地使用了语法，则可以选择使用 Snowflake Time Travel 功能将对象返回到其原始状态。

PROD

用于数据库和角色的生产环境（前缀）

QA

用于数据库和角色的测试/质量保证环境（前缀）

SP

存储过程（前缀）

TSK

任务（前缀）

DB

数据库

INT

集成

MVW

物化视图

RM

资源监视器

STG

阶段

SVC

服务账户

TBL

表（可选）

VW

View

WH

仓库

设置 Snowflake 试用账户

大多数公共云提供商解决方案要求您登录到管理门户才能配置数据仓库或数据库服务的实例。Snowflake 的方法则不同。使用 Snowflake，用户有一个单一且一致的入口点，从而降低了复杂性和管理开销。

注册 Snowflake 帐户后，您将获得一个以 snowflakecomputing.com 结尾的唯一 URL，创建实例后，您真的不需要了解底层云平台。地图由 Snowflake 人工绘制。此外，Snowflake 还提供跨多个云和区域的单一数据体验。

访问 Snowflake 注册页面，开始设置新的 Snowflake 试用账户（如图 C-1 所示）。

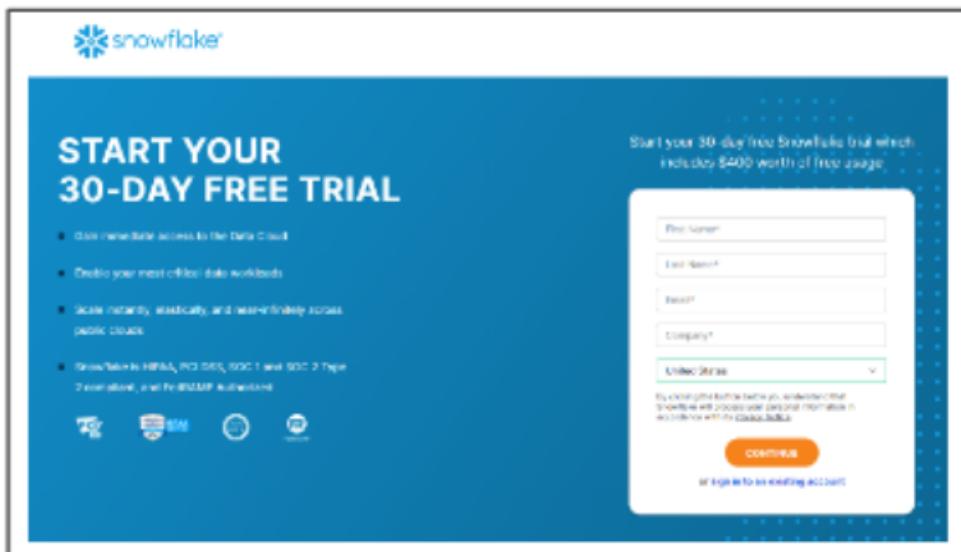


图 C-1. Snowflake 提供 30 天试用期

在第一个屏幕上填写信息后，您需要选择 Snowflake 版本的类型、云提供商和区域。在做出这些选择时，需要考虑一些因素。您可能希望选择大多数数据应用程序所在的同一公有云和区域，因为这将使您需要时管理数据传输变得更加容易且成本更低。

表 C-1 总结了 Snowflake 版本的主要差异。如果您要注册 Snowflake 试用帐户以完成本书中的练习，则企业版可能是您的最佳选择。Snowflake 企业版将让您访问完成书中所有练习所需的所有功能。

表 C-1 Snowflake 版本的比较

	标准企业	业务	危急	虚拟专用	雪花 (VPS)
完整的 SQL 数据仓库	X X X X	跨区域/云的安全数据共享	X X X	传输中和静态加密	X X X
时间旅行 (默认)	X X X X	客户专用的虚拟仓库	X X X X	联合身份验证	X X X X
外部功能	X X X X	Snowsight	X X X X X	数据库复制	X X X X
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-

	标准企业业务	危急	虚拟专用 雪花 (VPS)
创建您自己的数据交换 X XXX 市场访问 XXXX 多集群仓库 XX 长达 90 天的时间旅行 X			
X X 加密数据的年度重新密钥 XXX 物化视图 XX 搜索优化服务 XX 动态数据掩码 XX 外部数 据令牌化 XX HIPAA 支持 XX PCI 合规性 XX 数据加密无处不在 XX 三重机密使用客户管理			
“的密钥进行安全保护 XX AWS 和 Azure 专用链接支持 XX 数据库故障转移/故障恢复以实现业 务连续性 XX 外部功能 - AWS API Gateway 专用端点支持			
-			
-			
-			
-			
-			
-			
-			
-			
-			
-			
-			
-			
-			
-			
-			
-		X	X
客户专用的元数据存储			X
客户专用的虚拟服务器，无论加密密钥位于何处			X
-			

指数

符号

\$ (变量前缀) 、 144 @% (文件存储阶段引用) 、 80 @~ (用户阶段引用) 、 80

A

访问控制, 147–180 帐户访问, 231–235 分配用户角色, 165 创建对象, 150–153 自定义角色, 155–161, 163, 166 直接访问, 231 向角色授予权限, 64, 162–165, 271 间接访问, 231 多账户策略, 178 角色层次结构分配, 154–155,

159–161

角色、20、151、176 行访问策略和行级安全性,

253–256

次要角色, 177 系统定义的角色, 154–155 测试和验证安全性, 166–169 用户管理, 169–175, 178 ACCESS_HISTORY 帐户使用情况视图, 235–237, 245, 249 帐户成本, 管理, 261–285 敏捷软件交付, 278–284 BI 合作伙伴仪表板, 277 云服务层, 28, 265 数据共享, 322, 330, 334 数据传输成本, 264 数据库存储层, 46, 63 故障安全数据恢复, 55

Kafka 连接器相关, 216 物化视图, 307 成本中心对象标记, 276 重新集群, 306 远程服务, 144 复制, 244 搜索优化服务, 309 无服务器功能, 265 和虚拟仓库的大小, 31 个 Snowflake 月度账单, 262–265 个存储费用, 264 个任务, 108 基于使用量的定价, 261 虚拟仓库层, 44, 266–276 个账户级网络策略, 234–235 个账户到账户数据共享 (请参阅直接共享) ACCOUNTADMIN 角色, 59, 69, 151, 154, 161,

165

帐户, 单个与多个, 178 ACCOUNT_USAGE 架构, 69–70 ACCOUNT_USAGE 共享, 331 ACCOUNT_USAGE 视图, 查询, 276 Active Directory 联合身份验证服务 (AD FS),

232

广告代理商, 315 个聚合函数, 142–143 聚合和分桶数据, 363–366 个聚合查询, 308 敏捷软件交付, 278–284 CI/CD 管道, 279 数据库变更管理, 280–281 DevOps, 278 零副本克隆, 281–284

55 ALLOW_OVERLAPPING_EXECUTION参数, 104
ALTER 命令, 38 ALTER DATABASE 命令, 55、60
ALTER MATERIALIZED VIEW 命令, 78 ALTER TASK
RESUME 命令, 103 ALTER WAREHOUSE 命令, 272
分析与事务工作负载, 401 Apache Hive 元存
储, 71 Apache Iceberg, 71 Apache Kafka,
213–220 APPLICABLE_ROLES帐户视图, 66 应用程
序, 工作负载, 383 架构, 23–50 缓存, 46–50
云服务层, 27–29 数据库存储层, 45–46、63 查
询处理层, 29–44 传统数据平台架构,

间接访问 Snowflake, 231 预签名 URL 访
问, 140

C

缓存, 28、46–50、353 CALL 语句、数据库表
的 83 基数、303、305 CDP (持续数据保
护)、55 变更数据捕获 (CDC)、95、238
字符集、分隔文件、186 图表可视化、361–
362 子任务、102 CI/CD (持续集成、持续交
付/部署) 管道, 279 经典控制台、3 数据分
类、类别、249–251 CLONE 命令、60 克隆与
数据库共享、343 个表, 222 个零拷贝克
隆, 46、261、281–284 个云计算, 1 个用于
自动化 Snowpipe 的云消息收发,

24–26

虚拟仓库 (请参阅虚拟仓库) 算术运算符、124
ARRAY 数据类型, 131、185 ARRAY_INSERT函数、
201 分配规则、资源监视器、269–271
ASSOCIATE_SEMANTIC_CATE GORY_TAGS存储过程、
250 身份验证, 169–172、231–233 自动恢复、
33、38 自动暂停, 33、38 自动摄取
(Snowpipe)、213、218–221 自动扩展多集群虚
拟仓库、36、42、43 自动统计、353–358
Azure, 219–219

218–221

云服务层, 27–29 个集群密钥, 79、303–306 个
自动化 Snowpipe 的集群, 218–221 Kafka 连接
器, 214–217 最大化的多集群虚拟仓库,

42

多集群虚拟仓库、35–37、42 查询处理计算
层、29 横向扩展、30、35–37、43、293 协
作、数据、367–369 基于列的数据库、26、45–
46 数据掩码、251–253 派生列、118、121 和
垂直分区、293 COLUMNS 数据库视图、67 个公
共表达式 (CTE)、117、121–122、195 通
信和媒体行业、382 个比较运算符、124 个复
杂查询处理、31–35 压缩, 数据, 45、189、
295 计算列值, 121 COMPUTE_WH, 9 并发, 数
据, 24、35–37

B

伯努利采样, 349 BI (商业智能) (参见商业智
能 (BI)) 计费注意事项 (参见账户成本、人
力老龄化) 二进制数据类型, 129–130 BOOLEAN
数据类型, 126 个代理, Kafka, 215 个用于监控
使用情况和成本的商业智能 (BI) 仪表板, 277
和数据民主化, 380 Data Vault 2.0, 374–376

条件数据掩码, 253 个常量, 数字数据类型, 129 个直接数据共享中的消费者账户
322–330
上下文设置、Snowsight 工作表、8–14 持续数据保护 (CDP)、55 持续集成、持续交付/部署 (CI/CD) 管道、279 持续加载、189 Cookie 弃用、315 COPY 命令、185、205、225 COPY GRANTS 子句、72 COPY INTO 命令、71、80、212、222 相关子查询与不相关子查询、

120–122

成本, 账户 (请参见账户成本, 管理)
CREATE CLUSTER BY () 语句, 305 CREATE 命令, 222 CREATE DATABASE 命令, 55
CREATE IF NOT EXISTS 语句, 21, 58
CREATE OR REPLACE 语句, 21, 58 CREATE 语句, 21 CREATE TABLE AS SELECT (CTAS) com-mand, 283 CREATE TABLE LIKE 命令, 283 积分配额, 资源监视器, 268 积分使用情况, 资源监视器, 消耗计算的 268 个积分, 265、266 个 CSV 文件格式选项, 186–187 个 CTE (公共表表达式), 117、121–122、195 CURRENT_ACCOUNT () 函数, 343 个自定义角色, 155–161 个功能级业务和 IT 角色,

157–158、161 授予权限、163
系统级、158–161 虚拟仓库、
163、166 网络安全工作负载、
388–401

D

仪表板可视化、358–367 聚合和分桶数据、
363–366 图表可视化、361–362 创建仪表板和
磁贴、358–360 编辑和删除磁贴、366 用于协
作的专用链接、368 数据分析、工作负载、
380–383 数据应用程序、工作负载、383

数据缓存, 28, 46–50, 353 数据分类, 249–
251 数据净室, 316, 341 数据聚类, 299–
306, 309 聚类键, 79, 303–306 和具体化视
图, 307 和微分区, 45, 291–299 重新聚类,
306 与搜索优化服务, 392–401 宽度和深度,
299–303 数据协作, 367–369, 378–380 存储数
据压缩, 45, 189, 295 数据并发, 24, 35–
37 数据控制语言 (DCL), 112, 113 数据定
义语言 (DDL), 28, 112, 113

271–276

数据民主化, 244、380 数据字典 (参见
INFORMATION_SCHEMA) 数据出口费用, 264 数
据工程工作负载, 372 数据治理控制, 244–258
数据分类, 249–251 外部标记化, 256
INFORMATION_SCHEMA, 245 数据掩码, 251–253
对象依赖关系, 257 对象标记, 245–249 行访问
策略和行级安全性,

253–256

安全视图和安全 UDF, 257 数据入口费用,
Snowflake 的缺乏, 264 数据湖, 3, 377, 390–
391 数据加载和卸载, 183–226 替代方案, 222
数据工程师的最佳实践, 223–225 数据源, 191
数据类型, 185–224 文件压缩, 189 文件格
式, 80、185–188 数据处理频率, 189–190 加
载工具, 192–222

数据管道, 212–221 SnowSQL CLI, 210–212
工作表中的 SQL 命令, 192–204 UI 加载数据
向导, 204–210 阶段参考, 80、190–191 卸载
工具, 186、222

数据操作语言 (DML), 28、112、114、192、295 数据掩码, 251–253 数据货币化, 379 数据规范化, 293、374 数据管道, 212–221 Apache Kafka, 213–220 自动化 Snowpipe, 218–220 CI/CD 管道, 279 数据工程的重要性, 372 REST 终端节点, 调用 Snowpipe, 221 数据保护和恢复, 237–243 数据查询语言 (DQL), 112、114 数据采样 (Snowsight), 348–353 数据科学, 378、385 数据共享帐户成本, 322、330、334 作为数据加载的替代方案, 222 协作查询结果, 368 法规和合规性要求

对于, 379 个安全注意事项, 343 个数据源, 用于加载和卸载, 191 个数据超级英雄计划, 18 个数据传输成本, 264 个数据转换, 183 个连续加载, 190 个 COPY INTO 命令, 212 个自定义 Kafka 代码, 217 个用于数据管道, 212 个 DCM 工具, 280 个步骤, 225 个在存储前不需要的 NoSQL, 25 个在加载到 Snowflake 之前, 224 和半结构化数据类型, 185 在 Snowflake 中, 377 Snowflake 的数据摄取过程, 373 Snowpark 的角色, 386–387 数据类型, 126–141 二进制, 129–130 数据加载和卸载, 185、224 日期和时间输入/输出, 130 数字, 127–129 与搜索优化服务, 308 半结构化, 131–137, 185, 224 字符串, 129–130 非结构化, 137–141 Data Vault 2.0 建模, 374–376 数据仓库, 1

数据工作负载 (请参阅工作负载) 数据库变更管理 (DCM), 278, 280–281

数据库存储层, 45–46, 63 数据库克隆与共享, 343 基于列, 26, 45–46, 118, 121, 251–253 创建和管理, 54–63 使用存储过程删除, 93 数据加密和压缩, 45 按数据库对数据进行分组, 54 在入站共享与常规数据库上,

331

迁移工具, 280 永久与临时, 54, 72 复制注意事项, 244 在 Snowsight 中选择, 12–13 共享多个数据库, 319 非结构化数据存储中的表, 138 数据库账户视图, 66 数据流, 212 日期和时间输入/输出数据类型, 130 DCL 命令, 112、113 DCM (数据库变更管理), 278,

280–281

DDL 命令, 28、112、113、271–276 DELETE 命令, 71 Delta Lake, 71 数据访问和分析的民主化, 245、380 DEMO_DB 数据库, 62 派生列, 118、121 DESCRIBE 命令, 78 DevOps 框架, 278、280–281 直接标识符隐私类别, 250 直接共享, 316、318–332 入站共享, 消费者使用, 330–332,

343

出站共享, 创建, 318–330 非结构化数据的目录表, 138, 139 自主访问控制, 148 DML 命令, 28, 112, 114, 192, 295 基于文档的 NoSQL 数据库, 26 DQL 命令, 113, 114 管理数据中的漂移挑战, 280 DROP 命令, 174 DROP DATABASE 命令, 55 DROP TABLE 命令, 76

动态数据和静态数据，拆分，293 动态数据
掩码，251 动态表，71

E

E2EE（端到端加密），237 经济扩展方法，多集群虚拟仓库，36 ELT（提取加载转换）工具，
212, 221,

224

ENABLED_ROLES 帐户视图，66 种加密，45、237–
238、243 端到端加密（E2EE）、237 个 ETL
（提取转换加载）工具，212、221、224、253 事
件通知类型、云消息传递，218 EXECUTE_TASK 命
令，103 扩展检测和响应（XDR）、390 外部云
存储、无负载访问，222 外部功能，140、144 外
部阶段，79、81、82、138 外部表，70、73、
244 外部分词，256 EXTERNAL_TABLES 数据库视
图，67 提取加载转换（ELT）工具，212、
221、

224

提取转换加载（ETL）工具，212、221、224、
253

F

故障安全数据保护和恢复，46，55，72，
238，241–243 故障转移，243 联合身份验
证，232 文件压缩，189 文件格式，81，
185–188 FILE_FORMATS 数据库视图，67 金融行
业，314，380 防火墙访问管理，233–235 定
点数字数据类型，127–129 基于行数的固定大
小采样，

349

FLATTEN 函数，132、136–137 浮点数数据类
型，127–129 格式化 SQL，Snowsight 工作表，
15 基于概率的基于分数的采样，349、353 数据
处理频率，189–190

FROM 子句，117, 118 表的完全限定名称，76 功
能级自定义角色，157–158, 161 函数聚合，142–
143 外部，140, 144 Java，140 QUERY_HISTORY
分析，288 标量，141 SQL，141–143 系统，143
表，143 任务，104 UDF（请参阅用户定义的函
数（UDF））窗口，142 FUNCTIONS 数据库视
图，67

G

GEOGRAPHY 数据类型，126 地理空间数据，126
GET 命令，222 GET_PRESIGNED_URL 函数，140 全
球云服务（GCS）层，27–29, 47 治理，数据（请
参阅数据管理控制）GRANT 语句，139 授予权限，
64, 162–165, 271 基于图形的数据库，26 GROUP BY
子句，118

H

Hadoop 环境，71 硬件安全模块（HSM），
237 HASH() 函数，288 医疗保健行业，
315、381 用于加密的分层密钥模型，237 水
平数据库分区，292 HSM（硬件安全模块），
237 个中心表，375 混合列式架构，27、45–
46 混合表（Unistore），70、402

Identifier 隐私类别，250 个标识符、语法规
则，246 个身份和访问管理（IAM）角色、

190

身份和访问管理角色，190 身份提供商
(IdP)，179, 232

入站份额, 330–332, 343 独立查询, 不相关子查询 *as*, 120
间接标识符隐私类别, 250 INFORMATION_SCHEMA, 29, 65–69 帐户视图, 65–67 与 ACCOUNT_USAGE 共享, 331 数据库视图, 67–69, 245 个存储过程, 90 个 INFORMATION_SCHEMA_CATALOG_NAME 帐户视图, 66 个继承的对象标记, 247 个 Inmon 数据仓库设计, 374 个 INSERT ALL 命令, 198 个 INSERT 命令, 97、192 个 INSERT INTO 命令, 71、192 个集成用户, 231 个交互式结果 (Snowsight), 353–358 个内部阶段, 79、80、138、190–191 个间隔常量, 数据和时间数据类型, 131 个 IP 地址和安全措施, 233 个 IT 和职能级业务角色, 157–158

J
Java 函数, 处理数据, 140 个
JavaScript UDF, 84–86, 224
JavaScript, 存储过程, 89–95 JSON (NDJSON) 标准格式, 186 个 JSON 数据, 加载到阶段, 81 个 JSON 文件格式选项, 187–188

K
Kafka, 213–220 Kafka 连接器, 214–217 Kafka Streams API, 214 个键值数据库, 26 Kimball 维度模型, 374, 376 KSQL, 214

L
数据量大, 查询过程复杂, 31–35 LATERAL JOIN 函数, 137 LIMIT 子句, 347, 352 LIMIT/FETCH 子句, 118 个链接表, 375 Linstedt, Dan, 375 LIST @~; 或 LS@~;, 80, 191

文字, 数字数据类型, 129 数据加载 (参见数据加载和卸载) LOAD_HISTORY 账户视图, 66 逻辑运算符, 125 物流服务, 315, 382 长时间运行的查询, 35, 125, 288 Looker 仪表板, 277

M

机器学习模型, 383 管理授权权限, 64, 104 托管访问架构, 64 制造业, 382 毛, 弗朗西斯, 18 营销分析, 382 市场, 316, 333–340 数据掩码, 251–253 物化表, 71 物化视图, 76, 77–79, 306–308, 309 最大集群值, 设置, 37 最大化多集群虚拟仓库, 36,

42

微分区中的元数据, 299 和安全数据共享, 314 Snowsight 结果, 353 元数据缓存, 28、47、48 MFA (多因素身份验证), 7、232 微批处理, 190 个微分区, 45、291–299 Microsoft Azure, 219–219 Microsoft Power BI 仪表板, 277 多账户策略, 访问控制, 2, 178 多集群虚拟仓库, 35–37、42、43 多因素身份验证 (MFA), 7, 232 多个数据库, 共享, 319 多行插入 注意, 查询语法, 123–124 多行 SQL 命令插入, 194–198 多表 SQL 命令插入, 198–201

N

命名阶段, 79、80、82、191 命名空间, 54 命名最佳实践 Kafka 主题名称, 215 个对象, 20、423 本机应用程序, 388 NDJSON 标准格式, 186

近乎实时的处理, 190 个网络连接, 故障排除, 235 个网络安全策略, 233–235 换行分隔的 JSON 标准格式, 186 个非具体化视图, 76, 79 规范化优化技术, 293, 374 个 NoSQL 替代方案, 25 个通知和其他操作, 资源监控器, 269 种数字数据类型, 127–129

永久数据库与临时数据库, 54, 61, 72
个性化与标准数据列表, Marketplace, 337–340
个管道 (参见数据管道) 管道, 95 个 PIPES 数据库视图, 67 个 Powered by Snowflake, 384 精度, 数字数据类型, 127 个预签名 URL 访问, 140 个预览字段和数据, 349–352 个隐私考虑访问限制类型, 148 个数据洁净室, 341 个医疗隐私功能, 230 个用于协作的私人仪表板链接,

O

OAuth 集成, 232 OBJECT 数据类型, 131, 185 对象依赖关系, 257 对象层次结构, 65, 70, 80 对象访问控制, 150–153 自定义访问角色, 158 依赖关系、数据管理控制,

257

授予权限, 162 命名最佳实践, 20, 423 组织, 178 永久, 244 管道, 95 复制, 244 资源监视器, 266 架构对象层次结构, 70 序列, 95 可共享, 343 流 (请参阅流) 标记, 245–249, 276

368

标签分类, 250 和统一 ID 2.0, 315 私有数据交换, 316, 340, 341 权限, 授予, 64, 162–165, 271 概率, 基于分数基础采样,

349

PROCEDURES 数据库视图, 67 Provider Studio, 336 个数据共享提供者, 210, 229–230

335–336

用于远程服务调用的代理服务, 144 PUBLIC 角色, 151, 155 PUBLIC 架构, 65, 85 出版行业, 315 PUT 命令, 205, 211 Python 工作表, 385, 388

临时的, 243

瞬时, 243, 244

用户, 169 OBJECT_INSERT 函数, 202–204 OBJECT_PRIVILEGES 数据库视图, 67 操作数据存储 (ODS), 402 运算符、查询, 115, 124 ORDER BY 子句, 118 ORGADMIN 角色, 178 出站共享, 318–330

Q

QUALIFY 子句, 118 准标识符隐私类别, 250 个查询、ACCOUNT_USAGE 视图的 115–126, 276 聚合与搜索优化, 308 上下文设置, 8–14 开发和管理、Snowsight 中的 115–117 格式设置, 15 个限制, 126 个长时间运行, 35, 125, 288 运算符, 115, 124 性能和优化 (请参阅查询)

P

Parquet 文件, 189, 224 个分区, 45, 291–299 密码, 171 个永久对象, 复制, 244

performance) 查询处理层, 29–44

协作共享查询结果，368 子查询，118–122, 125
语法，115, 117–124 时间旅行，238 查询复杂性，作为虚拟仓库大小的标准，35 查询执行计划，48 查询优化器，49 查询性能，125, 287–310 分析，287–291 比较技术，309 数据聚类，299–306 物化视图，306–308 微分区，291–299 搜索优化服务，308 查询配置文件工具，290 查询结果缓存，47 `QUERY_HISTORY`分析，288

和访问控制（请参阅访问控制）管理、176、232 角色切换、6 辅助、177 超级用户或超级角色、155 用户管理、53、153 根任务、102 行访问策略和行级安全性，

253–256

逐行数据处理，避免，224

S

`SAMPLE` 子句，347、353 数据采样（`Snowsight`），348–353 卫星表，375 数据平台架构的可扩展性，23、24–25、28、30 标量函数，141 比例，数字数据类型，127 使用集群进行扩展，35–37、43–293 虚拟仓库的扩展过程，31–35 架构，45、63–70 `ACCOUNT_USAGE`架构，69–70 数据共享设计考虑，342 授予权限，64 `INFORMATION_SCHEMA`，65–69 对象层次结构，70 数据组织方式，54 永久与临时，64 保留时间，63 架构对象层次结构，65 `SCHEMATA` 数据库视图，67 SCIM 2.0，身份提供程序集成，178 范围 URL 访问，139 搜索优化服务，308、392–401 辅助角色，访问控制，177 安全数据库对象，53–109 创建和管理数据库，54–63 扩展 SQL，82–95 层次结构，150 个管道，95 个架构，45、54、63–70、342 个序列，95 个阶段，79–82 个存储过程，82–84, 89–95 个流，95–101 个表，70–76 个任务，102–108 个 UDF，82–95, 257

R

原始数据缓存，49–50 RBAC（基于角色的访问控制），148, 232, 281 读取器帐户，328–330, 330 重新聚集，306 作为信息引用完整性约束，374 `REFERENTIAL_CONSTRAINTS`数据库视图，67 远程磁盘层（参见数据库存储层）通过代理服务进行远程服务调用，144 复制，数据，243, 314 `REPLICATION_DATABASES` 帐户视图，66 数据共享的重新共享，禁止、344 资源监视器，266–276 `ACCOUNTADMIN` 作为所有者、150、152 积分配额、268、269 积分使用、268、269 数据共享、330 DDL 命令、271–276 通知和其他操作、269 分配规则、269–271 REST API（`Snowpipe`），213、221 REST 终端节点、调用 `Snowpipe`、221 恢复数据（请参阅按时间旅行）结果缓存、28 个零售垂直领域、数据分析、382 个 `REVOKE` 语句，139 个基于角色的访问控制（RBAC），148、232、281 个角色

视图, 76-79 安全数据共享, 2, 313-345 克隆与共享, 343 数据净室, 316, 341 数据协作, 378-380 设计注意事项, 342 直接共享, 316, 318-332

入站共享、330-332 出站共享、318-330 性能注意事项、343 私有数据交换、316、340、341 安全注意事项、343 数据共享、344 Snowflake Marketplace、316、333-340 Snowgrid、314 时间旅行注意事项、344 统一 ID 2.0、315 用例、314 安全套接字层 (SSL)、235 返回表格值的安全 SQL UDTF、

87-89

安全 UDF, 82, 87-89, 96, 257 安全视图, 77, 257 安全注意事项, 230-244 访问控制 (参见访问控制) 与访问效率, 147 ACCESS_HISTORY 帐户使用情况视图,

235-237、245、249 帐户访问控制、231-235 云服务层管理、28 数据治理 (请参阅数据治理控制) 数据保护和恢复、237-243 数据共享 (请参阅安全数据共享) 复制和故障转移、243 安全对象 (请参阅安全对象) Snowflake 功能、230 安全数据湖、3、390-391 安全信息和事件管理 (SIEM)、3、388-391 安全编排、自动化, 响应 (SOAR), 389 SECURITYADMIN 角色, 104, 155, 172 SELECT 命令, 114, 117, 195 SELECT 语句, 11, 89 SELECT SYSTEM\$WHITELIST () 函数,

235

SELECT SYSTEM\$WHITELIST_PRIVATE - LINK () 函数, 235

数据库表的选择性, 303-306 半结构化数据类型, 131-137、185、224 敏感信息隐私类别, 250 个 SEQUENCE 对象, 95 个 SEQUENCES 数据库视图, 67 个无服务器计算模型, 102、225 个会话变量, 创建 UDF, 144 个 SET 命令, 144 个 SET 运算符, 125 个影子 IT, 231 分片 (水平分区), 292 个共享磁盘架构, 24 个无共享架构, 25 个共享数据 (请参阅安全数据共享) 快捷方式, Snowsight, 16 SHOW 命令, 78、173 SHOW DATABASES 命令, 55、60、151 SHOW MATERIALIZED VIEWS 命令,

77

SHOW NETWORK POLICIES 语句, 235 SHOW SCHEMAS 命令, 63、68 SHOW TABLES 命令, 61 SHOW TAGS 语句, 247 SHOW VARIABLES 命令, 144 SHOW VIEWS 命令, 77 SIEM (参见安全信息和事件管理) Sigma 仪表板, 277 单点登录 (SSO), 232 智能自动完成, 14 数据库快照, 46 SnowCD, 235 Snowday 事件, 19 Snowflake Data Cloud Platform 架构 (参见架构) 优势和创新, 1-3 个认证、19 个代码示例注意事项、19-20 个社区、17-18 版本比较、428 个事件、19 个月度账单、262-265 个命名标准、20 个虚拟仓库 (请参阅虚拟仓库) Snowflake Data Exchange、3 个 SNOWFLAKE 数据库、62、69 个 Snowflake Marketplace、316、333-340 Snowflake 本机应用程序框架、385 个 Snowflake 组织、2 个

Snowflake Summit、19 SNOWFLAKE 用户、173
SNOWFLAKE_SAMPLE_DATA 数据库、62 Snowgrid、
314 Snowpark API、224 Snowpark 开发人员框架、2、222、384、

385–388

Snowpipe, 95 岁

自动化, 218–220

计费, 216 集成用户, 231 最佳文件大小、213

REST 终端节点、调用, 221 流数据、221

SnowPro Core 认证考试、19 Snowsight、3、
347–369 自动统计、353–358 协作、367–369 创
建数据库、56–59 仪表板可视化、358–367 数据
采样、348–353 交互式结果、353–358 记录数限
制、348 方向、4–5 首选项、6–7 配置文件、7
角色切换、6 支持, 7 版本历史记录, 16–
17, 280 工作表导航, 8–17 SnowSQL CLI,
111, 183 多因素身份验证, 232 PUT 和 COPY
INTO 命令, 210–212 查询开发, 115–117

Snowflake Load Data 向导, 209 SQL (结构化查
询语言), 111 SQL 命令, 112–114 数据类型
(请参阅数据类型) 函数 (请参阅函数) 顺序的
重要性, 152 插入工作表中, Snowsight 中的
192–204 个查询快捷方式、16 个 UDF 与存储过
程、83 个 SQL UDF、84、87–89 SSD 缓存、49
个 SSL (安全套接字层)、235 个 SSO (单点登
录)、232 个阶段文件 URL 访问、139、140 个
阶段、79–82

外部, 79, 81, 82, 138 内部, 79, 80,
138, 190–191 阶段数据库视图, 68 独立任务,
102 标准多集群虚拟仓库, 36 标准与个性化数据
列表, Marketplace, 337–340
STATEMENT_TIMEOUT_IN_SECONDS 持续时间值,
125 静态数据和动态数据, 拆分, 293 静态数据
掩码, 253 存储, 数据, 45–46, 63 费用,
264 集成, 190 个存储过程, 82–84、89–95
Streamlit, 388 个流, 95、96–101、139、190、
213 字符串数据类型, 129–130 结构化数据,
111、192–198、206–210 结构化查询语言
(SQL), 111 个子查询, 118–122, 125 超级用户
或超级角色, 155 SYSADMIN 角色, 56、60、79、
150、155、164 系统采样 (Snowsight), 349
SYSTEM\$CLUSTERING_INFORMATION 函数, 143、302
SYSTEM\$CURRENT_USER_TASK_NAME 函数, 104
SYSTEM\$ESTIMATE_SEARCH_OPTIMIZATION_COSTS
函数, 309 SYSTEM\$TASK_DEPENDENTS_ENABLE 函数,
103, 107 系统定义角色, 访问控制,
154–155 系统级自定义角色, 158–161

T

表权限, 68 个 Tableau 仪表板, 277 个
表, 70–76 个特征和工作负载, 303–305 动
态, 71 个外部, 70、73、244 个函数, 143
个使用流进行更改, 96–101 物化, 71 个永
久数据库, 54、61、72 个阶段, 79–82, 191
个瞬态数据库, 54、72、225、264 TABLES
数据库视图, 68

TABLESAMPLE 和 SAMPLE、353

TABLE_CONSTRAINTS 数据库视图、68

TABLE_PRIVILEGES 数据库视图、68

TABLE_STORAGE_METRICS 数据库视图、

68

对象标记，245–249，276 TASKADMIN 角色，

104 任务，93–95，102–108，139

TASK_DEPENDENTS 函数，104 TASK_HISTORY 函

数，104 TCL 命令，113，114 临时内部阶段，

81 临时关键字，80 临时对象，243 临时表，

70，73 测试和验证访问控制，166–169 测试框

架工具，281 第三范式方法，374 阈值，资源

监视器，271 个磁贴，仪表板可视化，358–

360，366 时间旅行，46 个优势，280 个命令，

239 个数据保护和恢复，238–243 个数据共享，

344 个默认期，55 个保留期，239 个

TIMESTAMP_TZ 值，125 个 TOP 子句，117 个传统

数据平台架构，24–26 个 NoSQL 替代方案，25

个可扩展架构，24 个共享磁盘架构，24 个事务

控制语言（TCL），112、114 个事务与分析工作

负载（Uni store），401 转换数据（请参阅数

据转换）临时数据库表，54，72，225，264

临时对象，243，244 试用帐户设置，427 触

发器，资源监视器，269 卡车运输公司，315

TRUNCATE 命令，71

UNDROP 命令，60 统一 ID 2.0/隐私/cookie，

315 Uni store，401–403 卸载工具，数据，

186,222 UNSET 命令，144 非结构化数据类型，

137–141 USAGE_PRIVILEGES 数据库视图，68 用

例，138、314 USE 命令，20、54 USE DATABASE

命令，58 USE SECONDARY ROLES 语句，177 用户

管理，231–233 访问控制，169–175 分配角

色，53、165 删除用户，174 个查询结果缓存

访问，47 个和角色，153 个使用 SCIM，178 个

用于单独的虚拟仓库，42 个用户阶段，79–

82,191 个用户定义的函数（UDF），82–95 个

JAVA UDF，140 个 JavaScript，84–86,224 个

安全，257 个会话变量，创建方式，144 个

Snowpark，386,387 个与存储过程，83 个任务

包括，84–86 个用户级网络策略，234–235 个

USERADMIN 角色，153,155，172 USER_ROLE，

105 USER_TASK_MANAGED_INITIAL_WARE

HOUSE_SIZE 命令，108 UTIL_DB 数据库，62

V

VALIDATE 函数，220 VARCHAR 数据类型，

134 VARCHAR 数据类型，多行插入，123 变

量，语法，144 VARIANT 数据类型，131–

134，185，217 版本历史记录，16–17，

280 垂直数据库分区，293 视图，76–79 帐

户，65–67 数据库，67–69，245 物化，

76，77–79，306–308，309 非物化，76，

79 安全，77，257

U

UDF（请参阅用户定义的函数）UI LOAD 数据

向导，204–210 个不相关子查询与相关子查

询

120–122

视图数据库视图，68 个虚拟列值，121 个虚拟仓库缓存，49–50 个虚拟仓库层，44、266–276 个虚拟仓库，1–2、21、29–44 计费，44 选择最佳大小，225 创建和使用，38–42 个自定义角色，163、166 默认 COMPUTE_WH，9 个默认用户，170 个本地磁盘缓存，47、49 管理文件大小，225 监控使用情况以降低成本，266–276 使用集群进行横向扩展，35–37、43、293 纵向扩展，31–35 选择仓库，11 存储和计算分离，29 大小，30、266 工作负载管理，42–44 可视化数据（请参阅 *Snowsight*）

WITH 子句，117
工作负载，371–403 平衡查询和用于集群的 DML，
306
网络安全，388–401 数据分析、380–383 数据应用程序、383 数据协作、378–380 数据工程、372 数据湖、377、390–391 数据科学、378、385 数据仓库、374–376 分离工作负载、42–44 Unistore、401–403 工作表导航（*Snowsight*）、4–5、8–17、115 上下文设置、8–14 格式化 SQL、15 快捷方式、16 智能自动完成、14 版本历史记录、16–17

W

Web UI 历史记录，289–291 Web UI 加载数据向导，204–210 Web 用户界面，3–5 WHEN-ELSE 子句，198 WHERE 子句，118，302 窗口函数，142

X

XDR（扩展检测和响应），390

Z

零拷贝克隆，46, 261, 281–284

关于作者

Joyce Kay Avila 作为业务和技术领导者拥有超过 25 年的经验。她是德克萨斯州注册会计师，目前持有 Snowflake 和 Amazon Web Services 的认证。她还拥有 12 项 Salesforce 认证，包括 Tableau CRM 和 Einstein Discovery 认证，以及多项 Salesforce 架构师级认证。

Joyce 获得了计算机科学和会计工商管理学士学位，获得了工商管理硕士学位，并完成了会计信息系统的博士课程。

为了支持 Salesforce 和 Snowflake 社区，Joyce 在她的 YouTube 频道上制作了一系列操作视频。2022 年，她被 Snowflake 选为全球 48 位 Snowflake Data 超级英雄之一。

跋

Snowflake: The Definitive Guide 封面上的动物是 junco（Junco），一种新世界麻雀。该属有多个物种，但确切的数量存在争议。

黑眼 junco 是北美最常见的物种之一。它们喜欢在地面上进行许多活动，例如觅食昆虫和种子，以及筑巢。它们倾向于在一个窝中产下 3 到 5 个蛋，孵化期为 11 到 13 天。Juncos 通常生活在干燥和森林地区，而那些生活在北方的 Juncos 则向南迁徙过冬。

O'Reilly 封面上的许多动物都濒临灭绝；他们都对世界很重要。

封面插图由 Karen Montgomery 绘制。封面字体为 Gilroy Semibold 和 Guardian Sans。文本字体为 Adobe Minion Pro；标题字体为 Adobe Myriad Condensed；代码字体是 Dalton Maag 的 Ubuntu Mono。

O'REILLY®

向专家学习。
自己也成为其中一员。

书籍 | 实时在线课程 Instant
Answers | 虚拟活动 视频 | 互动学
习

从 oreilly.com 开始。