使用 Amazon DynamoDB 对数据建模

AWS 规范性指导

AWS 规范性指导: 使用 Amazon DynamoDB 对数据建模

Table of Contents

使用 Amazon DynamoDB 对数据建模

流程、模板和最佳实践

亚马逊 Web Services (AWS)

2023 年 12 月（文档历史记录）

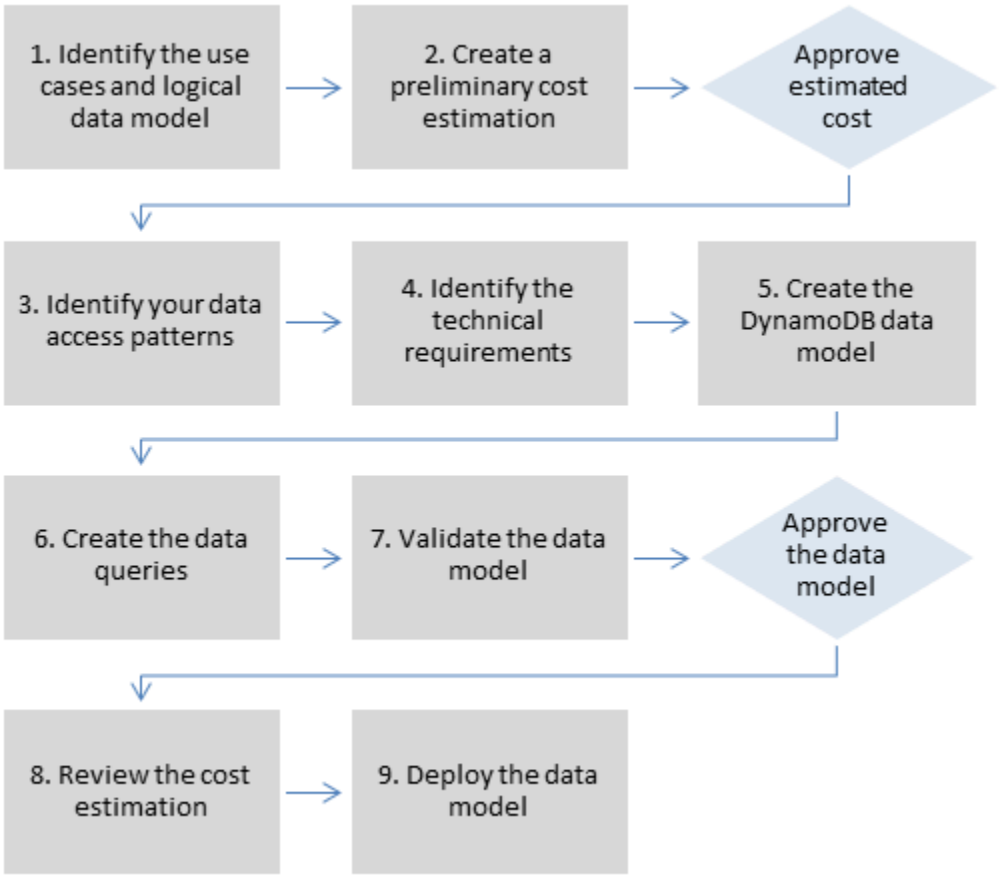
NoSQL 数据库为构建现代应用程序提供了灵活的架构。它们因其易于开发、功能强大和大规模性能而广受认可。Amazon DynamoDB 为 Amazon Web Services (AWS) 云中的 NoSQL 数据库提供快速、可预测的性能，能够实现无缝扩展。作为一项完全托管的数据库服务，DynamoDB 可帮助您减轻操作和扩展分布式数据库的管理负担。您不必担心硬件配置、设置和配置、复制、软件修补或集群扩展。

NoSQL 架构设计需要不同于传统关系数据库管理系统 (RDBMS) 设计的方法。RDBMS 数据模型侧重于数据的结构及其与其他数据的关系。NoSQL 数据建模侧重于访问模式或应用程序将如何使用数据，因此它以支持简单查询操作的方式存储数据。对于像 Microsoft SQL Server 或 IBM Db2 这样的 RDBMS，你可以创建标准化的数据模型，而无需过多考虑访问模式。您可以稍后扩展数据模型以支持您的模式和查询。

本指南介绍了使用 DynamoDB 的数据建模流程，该流程提供了功能要求、性能和有效成本。该指南适用于计划使用 DynamoDB 作为其正在 AWS 运行的应用程序的操作数据库的数据库工程师。AWS专业服务使用推荐的流程来帮助企业公司针对不同的用例和工作负载进行 DynamoDB 数据建模。

数据建模流程

我们建议在使用 Amazon DynamoDB 进行数据建模时采用以下流程。本指南稍后将详细讨论这些步骤。



RACI 矩阵

一些组织使用责任分配矩阵（也称为 RACI 矩阵）来描述一个特定项目或业务流程中涉及的各种角色。本指南提供了一个建议的 RACI 矩阵，可以帮助您的组织为 DynamoDB 数据建模过程确定合适的人员和正确的责任。对于该流程中的每个步骤，它列出了利益相关者及其参与情况：

- R — 负责完成步骤
- A — 负责批准和签署工作
- C — 咨询以提供任务的意见
- I — 了解了进展情况，但不直接参与任务

根据您的组织和项目团队的结构，以下 RACI 矩阵中的角色可以由同一个利益相关者担任。在某些情况下，利益相关方既要对具体步骤负责，又要承担责任。例如，数据库工程师可以负责创建和批准数据模型，因为这是他们的域区域。

过程步骤	业务用户	业务分析师	解决方案架构师	数据库工程师	应用程序开发者	DevOps 工程师
1. 确定用例和逻辑数据模型	C	R/A	I	R		
2. 创建初步的成本估算	C	A	I	R		
3. 识别数据访问模式	C	A	I	R		
4. 确定技术要求	C	C	A	R		
5. 创建 DynamoDB 数据模型	I	I	I	R/A		
6. 创建数据查询	I	I	I	R/A	R	
7. 验证数据模型	A	R	I	C		
8. 查看成本估算	C	A	I	R		
9. 部署 DynamoDB 数据模型	I	I	C	C		R/A

数据建模过程步骤

本节详细介绍了 Amazon DynamoDB 推荐的数据建模过程的每个步骤。

主题

第 1 步。确定用例和逻辑数据模型

目标

- 收集需要 NoSQL 数据库的业务需求和用例。
- 使用实体关系 (ER) 图定义逻辑数据模型。

过程

- 业务分析师会访谈业务用户，以确定用例和预期结果。
- 数据库工程师创建概念数据模型。
- 数据库工程师创建逻辑数据模型。
- 数据库工程师收集有关项目大小、数据量和预期读写吞吐量的信息。

工具和资源

- 业务需求评测（请参阅模板）

- 访问模式矩阵（请参阅模板）
- 您首选的图表创建工具

RACI

业务用户	业务分析师	解决方案架构师	数据库工程师	应用程序开发者	DevOps 工程师
C	R/A	I	R		

输出

- 记录用例和业务需求
- 逻辑数据模型（ER 图）

第 2 步。创建初步的成本估算

目标

- 为 DynamoDB 制定初步的成本估算。

过程

- 数据库工程师使用 DynamoDB 定价页面上提供的可用信息和示例创建初始成本分析。
  - 为按需容量创建成本估算（请参阅示例）。
  - 为预置容量创建成本估算（请参阅示例）。
    - 对于预置容量模型，请从计算器中获取估算成本，然后对预留容量应用折扣。
  - 比较两种容量模型的估计成本。
  - 为所有环境（开发、生产、质量保证）创建估算值。
- 业务分析师审查并批准或拒绝初步成本估算。

工具和资源

- AWS 定价计算器

RACI

业务用户	业务分析师	解决方案架构师	数据库工程师	应用程序开发者	DevOps 工程师
C	A	I	R		

输出

- 初步的成本估算

第 3 步。识别数据访问模式

访问模式或查询模式定义了用户和系统如何访问数据以满足业务需求。

目标

- 记录数据访问模式。

过程

- 数据库工程师和业务分析师会访谈最终用户，以确定如何使用数据访问模式矩阵模板查询数据。
  - 对于新应用程序，他们查看有关活动和目标的用户案例。他们记录用例并分析用例所需的访问模式。
  - 对于现有应用程序，他们分析查询日志以了解人们目前使用该系统的方式，以及确认键访问模式。
- 数据库工程师确定了访问模式的以下属性：
  - 数据大小：了解一次存储和请求的数据量将有助于确定对数据进行分区的最有效方法（请参阅博客帖子）。
  - 数据形状：NoSQL 数据库处理查询时不会改变数据形状（RDBMS 系统会这样做），而是整理数据，使数据库中的数据形状与查询内容对应。这是加快速度并增强可扩展性的一个关键因素。

  - 数据速度：DynamoDB 通过增加可用于处理查询的物理分区数量，并在这些分区高效分布数据来进行扩展。预先了解峰值查询负载可能有助于确定数据分区方式，从而最高效地使用 I/O 容量。
- 企业用户对访问或查询模式进行优先排序。
  - 优先级查询通常是最常用或最相关的查询。识别需要较低响应延迟的查询也很重要。

工具和资源

- 访问模式矩阵（请参阅模板）
- 选择正确的 DynamoDB 分区键（AWS 数据库博客）
- 适用于 DynamoDB 的 NoSQL 设计（DynamoDB 文档）

RACI

业务用户	业务分析师	解决方案架构师	数据库工程师	应用程序开发者	DevOps 工程师
C	A	I	R		

输出

- 数据访问模式矩阵

示例

访问模式	优先级	读或写	描述	类型（单件商品、多件商品或全部）	关键属性	筛选器	结果排序
创建用户个人资料	高	写入	用户创建新的个人资料	单个项目	Username	不适用	不适用

访问模式	优先级	读或写	描述	类型（单件商品、多件商品或全部）	关键属性	筛选器	结果排序
更新用户个人资料	中	写入	用户更新了他们的个人资料	单个项目	Username	用户名 = 当前用户	不适用

第 4 步。确定技术要求

目标

- 收集 DynamoDB 数据库的技术要求。

过程

- 业务分析师使用评估问卷采访业务用户和 DevOps 团队，以收集技术需求。

工具和资源

- 技术需求评估（参见示例问卷）

RACI

业务用户	业务分析师	解决方案架构师	数据库工程师	应用程序开发者	DevOps 工程师
C	C	A	R		

输出

- 技术要求文档

第 5 步。创建 DynamoDB 数据模型

目标

- 创建 DynamoDB 数据模型。

过程

- 数据库工程师确定每个用例需要多少张表。我们建议在 DynamoDB 应用程序中保留尽可能少的表。
- 根据最常见的访问模式，确定可以是以下两种类型之一的主键：带有标识数据的分区键的主键，或带有分区键和排序键的主键。排序键是用于分组和组织数据的辅助索引，因此可以在分区内高效地对其进行查询。您可以利用排序键，在数据中定义层级关系，在任何层级查询（请参阅博客帖子）。
  - 分区键设计
    - 定义分区键并评估其分布。
    - 确定是否需要进行写入分片均匀分配工作负载。
  - 排序键设计
    - 确定排序键。
    - 确定是否需要复合排序键。
    - 确定版本控制的需求。
- 根据访问模式，确定二级索引以满足查询要求。
  - 确定是否需要本地二级索引（LSI）。存在分区键与基表相同、但排序键不同的索引。
    - 对于具有 LSI 的表，每个分区键值的大小限制为 10 GB。只要任何一个分区键值的总大小不超过 10 GB，带有 LSI 的表就可以存储任意数量的项目。
  - 确定是否需要全局二级索引（GSI）。存在分区键和排序键可以与基表上的分区键和排序键不同的索引（请参阅博客帖子）。
  - 定义索引投影。考虑减少投影属性的数量，尽可能减少写入索引的项目大小。在此步骤中，应确定是否要使用以下内容：
    - 稀疏索引
    - 物化聚合查询
    - GSI 重载
    - GSI 分片
    - 使用 GSI 的最终一致的副本

- 数据库工程师决定数据是否包含大型项目。如果是，他们通过使用压缩方法或在 Amazon Simple Storage Service (Amazon S3) 存储数据，设计解决方案。
- 数据库工程师决定是否需要时间序列数据。如果是，他们将使用时间序列设计模式对数据进行建模。
- 数据库工程师确定 ER 模型是否包含 many-to-many 关系。如果是，他们将使用相邻列表设计模式对数据进行建模。

工具和资源

- 适用@@ 于 Amazon DynamoDB 的 NoSQL Workbench — 提供数据建模、数据可视化以及查询开发和测试功能，帮助您设计 Dyn amoDB 数据库
- 适用于 DynamoDB 的 NoSQL 设计（DynamoDB 文档）
- 选择正确的 DynamoDB 分区键（AWS 数据库博客）
- 在 DynamoDB 中使用二级索引的最佳实践（DynamoDB 文档）
- 如何设计 Amazon DynamoDB 全球二级索引（AWS 数据库博客）

RACI

业务用户	业务分析师	解决方案架构师	数据库工程师	应用程序开发者	DevOps 工程师
I	I	I	R/A		

输出

- 满足您的访问模式和要求的 DynamoDB 表架构

示例

以下屏幕截图显示了 NoSQL 工作台。

RetailDatabase GSI: BundleProducts-VendorProducts-CustomerOrdersByOrderId GSI: VendorOrdersByStatusDate-ProductInventory GSI: ProductCatalog-CustomerOrdersByProduct

Primary Key		Attributes					
Partition Key: pk	Sort Key: sk						
P1	B1	GSI1-PK	GSI1-SK	name	desc		
		B1	P1	The Tiki Bundle	Everything you need for an island theme party.		
P4	B2	GSI1-PK	GSI1-SK	name	desc		
		B2	P4	Tiki Bar Set	Be the Mai Tai master with your very own Tiki Bar.		
P2	B1	name	desc	qty	GSI1-PK	GSI1-SK	location
		Tiki Torch	Bamboo tiki torch, 4 ft	6	B1	P2	W1-A9-S10-B52
	B2	name	desc	qty	GSI1-PK	GSI1-SK	location
		Tiki Torch	Bamboo tiki torch, 4 ft	2	B2	P2	W1-A9-S10-B52
	P2	name	desc	qty	location	reorderAt	GSI3-SK
		Tiki Torch	Bamboo tiki torch, 4 ft	656	W1-A9-S10-B52	100	/GardenOutdoor/OutdoorDecor/Lighting/LanternsT
	B1	name	desc	qty	GSI1-PK	GSI1-SK	location
		Tiki Statue - Pele	Tiki of the Hawaiian Fire Goddess Pele, 5 ft.	1	B1	P3	W1-A15-S6-B27

第 6 步。创建数据查询

目标

- 创建主要查询以验证数据模型。

过程

- 数据库工程师在 AWS 区域或他们的计算机（DynamoDB Local）上手动创建 DynamoDB 表。
- 数据库工程师将示例数据添加到 DynamoDB 表中。
- 数据库工程师使用 NoSQL Workbench for Amazon DynamoDB 或适用于 Java 或 Python 的 AWS 构建分面，来构建示例查询（请参阅博客帖子）。

  分面就像 DynamoDB 表的视图。
- 数据库工程师和云开发人员使用首选语言的 AWS Command Line Interface (AWS CLI) 或 AWS 软件开发工具构建示例查询。

工具和资源

- 一个活跃 AWS 账户，用于访问 DynamoDB 控制台
- DynamoDB Local（可选），如果您想在不访问 DynamoDB 网络服务的情况下在计算机上构建数据库
- NoSQL Workbench for Amazon DynamoDB（下载和文档）
- AWS使用你选择的语言（Python JavaScript、PHP、.NET、Ruby、Java、Go、Node.js、C++ 和 SAP ABAP）开发工具包

RACI

业务用户	业务分析师	解决方案架构师	数据库工程师	应用程序开发者	DevOps 工程师
I	I	I	R/A	R	

输出

- 查询 DynamoDB 表的代码

示例

- 使用适用于 Java 的 AWS SDK 的 DynamoDB 示例
- Python 示例
- JavaScript例子

第 7 步。验证数据模型

目标

- 确保数据模型满足您的要求。

过程

- 数据库工程师用示例数据填充到 DynamoDB 表中。
- 数据库工程师运行代码来查询 DynamoDB 表。
- 数据库工程师收集查询结果。
- 数据库工程师收集查询性能指标。
- 企业用户验证查询结果是否满足业务需求。
- 业务分析师验证技术要求。

工具和资源

- 一个活跃 AWS 账户，用于访问 DynamoDB 控制台
- DynamoDB Local（可选），如果您想在不访问 DynamoDB 网络服务的情况下在计算机上构建数据库
- 使用您选择的语言的 AWS SDK

RACI

业务用户	业务分析师	解决方案架构师	数据库工程师	应用程序开发者	DevOps 工程师
A	R	I	C		

输出

- 批准的数据模型

步骤 8：查看成本估算

目标

- 定义容量模型并估计 DynamoDB 成本，以完善第 2 步中的成本估算。
- 获得业务分析师和利益相关者的最终财务批准。

过程

- 数据库工程师确定数据量的估计值。
- 数据库工程师确定数据传输要求。
- 数据库工程师定义所需的读取和写入容量单位。
- 业务分析师在按需容量模式和预置容量模式之间做出决定。
- 数据库工程师确定了 DynamoDB 自动扩缩的需求。
- 数据库工程师在 Simple Monthly Calculator 工具中输入参数。
- 数据库工程师向业务利益相关者提供最终的价格估算。
- 业务分析师和利益相关者批准或拒绝该解决方案。

工具和资源

- AWS 定价计算器

RACI

业务用户	业务分析师	解决方案架构师	数据库工程师	应用程序开发者	DevOps 工程师
C	A	I	R		

输出

- 容量模式
- 订正成本估算

第 9 步。部署数据模型

目标

- 将 DynamoDB 表（或多张表）部署到。AWS 区域

过程

- DevOps 架构师为 DynamoDB 表（或多个表）创建AWS CloudFormation模板或其他基础设施即代码 (IaC) 工具。 AWS CloudFormation提供了一种自动配置和配置表及相关资源的方式。

工具和资源

- AWS CloudFormation

RACI

业务用户	业务分析师	解决方案架构师	数据库工程师	应用程序开发者	DevOps 工程师
I	I	C	C		R/A

输出

- AWS CloudFormation 模板

示例

```
mySecondDDBTable:
   Type: AWS::DynamoDB::
   Table DependsOn: "myFirstDDBTable"
   Properties:
   AttributeDefinitions:
      -  AttributeName: "ArtistId"
         AttributeType: "S"
      -  AttributeName: "Concert"
         AttributeType: "S"
      -  AttributeName: "TicketSales"
         AttributeType: "S"
   KeySchema:
      -  AttributeName: "ArtistId"
         KeyType: "HASH"
```

```
      -  AttributeName: "Concert"
         KeyType: "RANGE"
   ProvisionedThroughput:
      ReadCapacityUnits:
         Ref: "ReadCapacityUnits"
      WriteCapacityUnits:
         Ref: "WriteCapacityUnits"
   GlobalSecondaryIndexes:
      -  IndexName: "myGSI"
         KeySchema:
            - AttributeName: "TicketSales"
            KeyType: "HASH"
         Projection:
            ProjectionType: "KEYS_ONLY"
         ProvisionedThroughput:
         ReadCapacityUnits:
            Ref: "ReadCapacityUnits"
         WriteCapacityUnits:
            Ref: "WriteCapacityUnits"
   Tags:
      -  Key: mykey
         Value: myvalue
```

模板

本节提供的模板基于 AWS 网站上使用 Amazon DynamoDB 对游戏玩家数据进行建模。

> **Note**
>
> 本节中的表格使用 MM 作为百万的缩写，使用 K 作为千的缩写。

主题

业务需求评估模板

提供用例的描述：

> **描述**
>
> 想象一下，您正在开发一款在线多人游戏。在您的游戏中，由 50 名玩家组成的小组加入一个会话来玩游戏，这通常需要大约 30 分钟才能玩完。在游戏过程中，您必须更新特定玩家的记录，以显示该玩家的游戏时间、他们的统计数据或他们是否赢得了比赛。用户希望看到他们之前玩过的游戏，要么查看游戏的获胜者，要么观看每款游戏动作的重播。

提供有关您的用户的信息：

用户	描述	预期数字
游戏玩家	在线游戏玩家。	1 毫米
开发小组	内部团队将使用游戏统计数据来改进 游戏体验。	100

提供有关数据来源和数据摄取方式的信息：

源	描述	用户
在线游戏	游戏玩家将创建个人资料并开始新游戏。	游戏玩家
游戏应用程序	游戏应用程序将自动收集有关游戏的统计信息，例如开始和结束时间、玩家人数、每个玩家的位置以及游戏地图。	

提供有关如何使用数据的信息：

消费者	描述	用户
在线游戏	游戏玩家将查看个人资料并查看他们的游戏统计信息。	游戏玩家
数据分析	游戏开发团队将提取游戏统计数据进行数据分析并改善用户体验。数据将从数据存储中导出并导入到 Amazon S3 中，以支持通过 Spark 应用程序进行分析。	开发小组

提供实体清单及其识别方式：

实体名称	描述	标识符
游戏玩家	存储每个用户（玩家）的身份、地址、人口统计、兴趣等信息。	用户名

游戏实例	提供有关玩过的每个游戏的信息，包括创建者、开始、结束和地图 Yplayed。	游戏编号
游戏用户映射	代表用户和游戏之间的 many-to-many 关系。	游戏 ID 和用户名

为实体创建 ER 模型：

提供有关实体的高级统计数据：

Entity Name	估计的记录数	记录大小	备注
游戏玩家	1 毫米	< 1 KB	该游戏平台有大约100万用户。
游戏实例	6 毫米 (100,000k/天 * 60 天)	< 1 KB	平均而言，每天有10万场游戏。我们需要存储最近 60 天的数据。

游戏用户映射	300 MM (6 个 MM 游戏 * 50 个玩家)	< 1 KB	平均而言，每款游戏有 50 名玩家需要我们存储相关信息。

技术需求评估模板

提供有关数据摄取类型的信息：

数据提取类型	是/否	描述	Frequency
应用程序访问权限	Y		
API 网关	Y		
数据流	否		
Batch 处理	否		
ETL	否		
数据导入	否		
时间序列	否		

提供有关数据使用类型的信息：

数据消耗类型	是/否	描述	Frequency
应用程序访问权限			
API 网关			
数据导出			
数据分析			
数据聚合			

报告

搜索

数据流

ETL

提供数据量估算值：

实体名称	估计的记录数	记录大小	数据量
游戏玩家	1 毫米	< 1 KB	~ 1 GB (1 毫米 * 1 KB)
游戏实例	6 毫米 (100k/天 * 60 天)	< 1 KB	~ 6 GB (6 毫米 * 1 KB)
游戏用户映射	300 MM (6 个 MM 游戏 * 50 个玩家)	< 1 KB	大约 300 GB (300 MM * 1 KB)

> **Note**
>
> 数据留存期 60 天。60 天后，必须将数据存储在 Amazon S3 中进行分析，方法是使用 DynamoDB 生存时间 (TTL) 自动将数据从 DynamoDB 移至 Amazon S3。

回答以下有关时间模式的问题：

- 用户可以在什么时间范围内使用该应用程序（例如，全天候或工作日上午 9 点至下午 5 点）？
- 白天的使用量是否达到高峰？ 多少小时？ 应用程序使用百分比是多少？

指定写入吞吐量要求：

实体名称	每天写入	小时/天	写入/秒
游戏玩家	一万次更新	18	< 1
游戏实例	30万	18	< 5
游戏用户映射	1800,000,000	18	~ 27.777

注意事项

游戏玩家写入操作：1% 的用户每天更新他们的个人资料，因此我们预计 100 万名用户会有 10,000 次更新。
游戏实例写入操作：100,000 个游戏/天。对于每款游戏，我们至少有 3 个写入操作（在创建、开始和结束时），因此总共有 300,000 个写入操作。
游戏用户映射写入操作：每款游戏每天有 100,000 个，每次有 50 个玩家。平均游戏时长为 30 分钟，玩家位置每 5 秒更新一次。我们估计每位玩家平均有 360 次更新，因此总数为 100,000 * 50 * 360 = 18 亿次写入操作。

指定读取吞吐量要求：

实体名称	读数/天	小时/天	读取/秒
游戏玩家	20万	18	~ 3
游戏实例	5,000,000	18	~ 77
游戏用户映射	1800,000,000	18	~ 27.777

注意事项

游戏玩家读取操作：20% 的用户开始游戏，因此 1 MM * 0.2 = 200,000。
游戏实例读取操作：100,000 个游戏/天。对于每款游戏，我们每位玩家至少有 1 次读取操作，每款游戏有 50 名玩家，因此读取操作总数为 500 万次。

游戏用户映射读取操作：每天50 个玩家有 100,000 场游戏。平均游戏时长为 30 分钟，玩家位置每 5 秒更新一次。我们估计每位玩家平均需要 360 次更新，并且每次更新都需要读取操作，因此总数为 100,000 * 50 * 360 = 18 亿次读取操作。

指定数据访问延迟要求：

操作	99 个百分位数	最大延迟
读取	30 毫秒	100 毫秒
写入	10 毫秒	50 毫秒

指定数据可用性要求：

要求	是/否	指标	备注
高可用性	Y	99.9%	
RTO	Y	1 小时	恢复时间目标
RPO	Y	1 小时	恢复点目标
灾难恢复	否		
区域内数据复制	否		
跨区域数据复制	否	3 秒延迟	哪个AWS 区域？

指定安全要求：

要求	是/否	备注
敏感数据存储	否	受保护的健康信息 (PHI)、支付卡行业 (PCI) 信息、个人身份信息 (PII)？
静态加密	Y	

传输中加密	Y
客户端加密	否
任何专有或第三方供应商的加密库	否
数据访问记录	否
数据访问审计	否

访问模式模板

使用以下字段收集和记录有关用例访问模式的信息：

字段	描述
访问模式	提供访问模式的名称。
描述	提供对访问模式的更详细描述。
优先级	定义访问模式的优先级（高、中或低）。这定义了应用程序最相关的访问模式。
读或写	它是读取权限还是写入访问模式？
类型	该模式是访问单个项目、多个项目还是所有项目？
筛选条件	访问模式是否需要任何过滤器？
排序	结果需要任何排序吗？

模板

访问模式	描述	优先级	读或写	类型（单件商品，	关键属性	筛选器	结果排序

				多个 物品，或全部）			
创建用户个人资料	用户创建新的个人资料。	高	写入	单个项目	用户名	不适用	不适用
更新用户个人资料	用户更新其个人资料。	中等	写入	单个项目	用户名	用户名 = 当前用户	不适用
获取用户个人资料	用户查看他们的个人资料。	高	读取	单个项目	用户名	用户名 = 当前用户	不适用
创建游戏	用户创建了一个新游戏。	高	写入	单个项目	GameID	不适用	不适用
查找开放游戏	用户搜索打开的游戏。搜索结果按起始时间戳降序排序。	高	读取	多件商品		GameStatus = 打开	起始时间戳后代
通过地图查找打开的游戏	用户使用按开始时间戳降序排序的特定地图来搜索已打开的游戏 订购。	中等	读取	多件商品		GameStatus = 打开并地图 = XYZ	起始时间戳后代

查看游戏	用户查看游戏的细节。	高	读取	单个项目	GameID	不适用	不适用
查看游戏中的用户	用户会获得游戏中所有用户的列表。	中等	读取	多件商品		GameID = XYZ	不适用
让用户加入游戏	用户加入打开的游戏。	高	写入	单个项目	GameID 和用户名	GameStatus = 打开	不适用
开始一场游戏	用户开始新游戏。	高	写入	单个项目	GameID	不适用	不适用
为用户更新游戏	更新用户在游戏中的位置。	中等	写入	单个项目	GameID 和用户名	不适用	不适用
更新游戏	游戏结束；更新统计数据。	中等	写入	单个项目	GameID	不适用	不适用
为用户查找所有过去的游戏	按游戏开始时间戳顺序列出用户玩过的所有游戏。	低	读取	多件商品	用户名和游戏 ID	用户名 = 当前用户	开始时间戳

导出数据进行数据分析	开发团队将运行批处理作业，将数据导出到 Amazon S3。	低	读取	全部	不适用	不适用	不适用

最佳实践

考虑使用以下 DynamoDB 设计最佳实践：

- 分区键设计 - 使用高基数分区键均匀分配负载。
- 邻接列表设计模式-使用此设计模式进行 one-to-many 管理和 many-to-many 关系。
- 稀疏索引-使用稀疏索引作为全局二级索引 (GSI)。创建 GSI 时，您可以指定分区键和排序键（可选）。只有基表中包含相应 GSI 分区键的项目才会显示在稀疏索引中。这有助于缩小 GSI。
- 索引过载 - 使用相同的 GSI 为各种类型的项目编制索引。
- GSI 写入分片 - 明智地分片，将数据分布到各个分区，以实现更高效、更快速的查询。
- 大项目 - 仅在表中存储元数据，将 blob 保存在 Amazon S3 中，将引用保留在 DynamoDB 中。将大项目分成多个项目，并使用排序键有效地编制索引。

有关更多最佳设计实践，请参阅亚马逊 DynamoDB 文档。

分层数据建模示例

以下各节以汽车公司为例，说明如何使用数据建模过程步骤在 DynamoDB 中设计多级组件管理系统。

主题

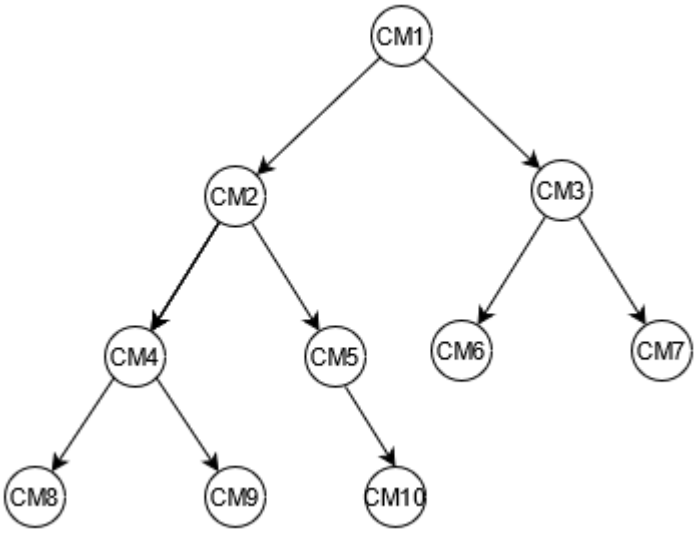
步骤 1：确定用例和逻辑数据模型

一家汽车公司希望建立一个交易组件管理系统来存储和搜索所有可用的汽车零部件，并在不同的部件和零件之间建立关系。例如，一辆汽车包含多个电池，每个电池包含多个高级模块，每个模块包含多个单元，每个单元包含多个低级组件。

通常，对于构建层次关系模型，Amazon Neptune 之类的图形数据库是更好的选择。但是，在某些情况下，由于其灵活性、安全性、性能和可扩展性，Amazon DynamoDB 是分层数据建模的更好替代方案。

例如，您可以构建一个系统，其中 80-90% 的查询是事务性的，那么 DynamoDB 就非常适合。在此示例中，其他 10-20% 的查询是关系型查询，其中像 Neptune 这样的图形数据库更适合。在这种情况下，在架构中添加一个额外的数据库以仅满足 10-20% 的查询可能会增加成本。它还增加了维护多个系统和同步数据的运营负担。您可以在 DynamoDB 中对 10-20% 的关系查询进行建模。

绘制汽车组件的示例树有助于映射它们之间的关系。以下示意图显示了具有四个级别的依赖关系图。CM1 是示例汽车本身的顶部组件。它有两个子组件，用于两个示例电池：CM2 和 CM3。每个电池有两个子组件，即模块。CM2 具有模块 CM4 和 CM5，CM3 具有模块 CM6 和 CM7。每个模块有几个子组件，即单元。CM4 模块有两个单元：CM8 和 CM9。CM5 只有一个单元，即 CM10。CM6 和 CM7 还没有任何关联的单元。



本指南将使用此树及其组件标识符作为参考。顶部组件称为父级，子组件称为子级。例如，顶部组件 CM1 是 CM2 和 CM3 的父级。CM2 是 CM4 和 CM5 的父级。这描绘了父子关系图。

从树状图中，您可以看到组件的完整依赖关系图。例如，CM8 依赖于 CM4，CM4 依赖于 CM2，CM2 依赖于 CM1。树将完整的依赖关系图定义为路径。路径描述了两点：

- 依赖关系图
- 树中的位置

填写业务需求模板：

提供有关您的用户的信息：

用户	描述
员工	需要汽车及其零部件信息的汽车公司内部员工

提供有关数据来源和数据摄取方式的信息：

源	描述	用户
管理系统	该系统将存储与可用汽车零件及其与其他部件和零件的关系有关的所有数据。	员工

提供有关如何使用数据的信息：

消费者	描述	用户
管理系统	检索父组件 ID 的所有直接子组件。	员工
管理系统	检索组件 ID 的所有子组件的递归列表。	员工
管理系统	查看组件的祖先。	员工

第 2 步：创建初步成本估算

计算应用程序所有环境的成本估算值很重要，这样您就可以检查解决方案在财务上是否可行。最佳做法是在继续开发和部署之前，先进行高级估算并获得业务分析师的批准。

- 数据库工程师使用 DynamoDB 定价页面上提供的可用信息和示例创建初始成本分析。
  - 为按需容量创建成本估算（请参阅示例）。
  - 为预置容量创建成本估算（请参阅示例）。
    - 对于预置容量模型，请从计算器中获取估算成本，然后对预留容量应用折扣。
  - 比较两种容量模型的估计成本。
  - 为所有环境（开发、生产、质量保证）创建估算值。
- 业务分析师审查并批准或拒绝初步成本估算。

使用这些参考值，您可以创建预估价格以提交审批。要创建预算，您可以使用 DynamoDB 定价页面和 AWS 定价计算器。

步骤 3：确定您的数据访问模式

此示例用例具有以下访问模式，用于管理不同汽车组件之间的关系。

访问模式	优先级	读或写	描述	类型	筛选器	结果排序
直系子女	高	读取	检索父组件 ID 的所	多个	`Component ID`	不适用

			有直接子组件。			
所有子组件	高	读取	检索组件 ID 的所有子组件的递归列表。	多个	Component ID	不适用
祖先	高	读取	检索组件的祖先。	多个	Component ID	不适用

第 4 步：确定技术要求

此示例没有任何特定的技术要求，这些要求超出了本示例的范围。在实际情况下，最佳做法是在继续开发和部署之前完成此步骤并验证是否满足了所有技术要求。您可以使用示例问卷来完成业务案例中的此步骤。此外，我们建议验证 DynamoDB 服务配额，以确保您的设计解决方案没有硬性限制。

步骤 5：创建 DynamoDB 数据模型

为基表和全局二级索引 (GSI) 定义分区键：

- 遵循密钥设计最佳实践，在本示例中ComponentId用作基表的分区键。因为它是独一无二的，所以ComponentId可以提供粒度。DynamoDB 使用分区键的哈希值来确定物理存储数据的分区。唯一的组件 ID 会生成不同的哈希值，这有助于在表内分布数据。您可以使用ComponentId分区键查询基表。
- 要查找组件的直接子组件，请创建一个 GSI，其中ParentId是分区键，ComponentId也是排序键。您可以使用ParentId作为分区键来查询此 GSI。
- 要查找组件的所有递归子级，请创建一个 GSI，其中 GraphId 是分区键，Path 是排序键。您可以将 GraphId 用作分区键和排序键上的 BEGINS_WITH(Path, "$path") 运算符来查询此 GSI。

	分区键	排序键	映射属性
基表	ComponentId		ParentId, GraphId, Path
GSI1	ParentId	ComponentId	

GSI2	GraphId	Path	ComponentId

在表格中存储组件

下一步是将每个组件存储在 DynamoDB 基表中。插入示例树中的所有组件后，您将获得以下基表。

ComponentId	ParentId	GraphId	路径
CM1		CM1 #1	CM1
CM2	CM1	CM1 #1	CM1|CM2
CM3	CM1	CM1 #1	CM1|CM3
CM4	CM2	CM1 #1	CM1|CM2|CM4
CM5	CM2	CM1 #1	CM1|CM2|CM5
CM6	CM3	CM1 #1	CM1|CM3|CM6
CM7	CM3	CM1 #1	CM1|CM3|CM7
CM8	CM4	CM1 #1	CM1|CM2|CM4|CM8

CM9	CM4	CM1 #1	CM1|CM2|CM4|CM9
CM10	CM5	CM1 #1	CM1|CM2|CM5|CM10

GSI1 索引

要检查组件的所有直接子组件，可以创建一个`ParentId`用作分区键和`ComponentId`排序键的索引。以下数据透视表表示 GSI1 索引。您可以使用此索引通过父组件 ID 检索所有直接子组件。例如，您可以了解汽车中有多少电池可用（CM1）或模块中有哪些单元可用（CM4）。

ParentId	ComponentId
CM1	CM2
	CM3
CM2	CM4
	CM5
CM3	CM6
	CM7
CM4	CM8
	CM9
CM5	CM10

GSI2 索引

以下数据透视表表示 GSI2 索引。它配置为将 `GraphId` 用作分区键，将 `Path` 用作排序键。使用 `GraphI d` 和对排序键 (`Path`) 的`begins_with`操作，你可以在树中找到组件的完整谱系。

GraphId	路径	ComponentId

CM1 #1	CM1	CM1
	CM1|CM2	CM2
	CM1|CM3	CM3
	CM1|CM2|CM4	CM4
	CM1|CM2|CM5	CM5
	CM1|CM2|CM4|CM8	CM8
	CM1|CM2|CM4|CM9	CM9
	CM1|CM2|CM5|CM10	CM10
	CM1|CM3|CM6	CM6
	CM1|CM3|CM7	CM7

步骤 6：创建数据查询

定义访问模式并设计数据模型后，您可以查询 DynamoDB 数据库中的分层数据。作为节省成本和帮助确保性能的最佳实践，以下示例仅使用不带查询操作`Scan`。

- 查找组件的祖先。

  要查找 CM8 组件的原级（父级、祖父级、曾祖父级等），请使用 `ComponentId = "CM8"` 查询基表。查询将返回以下记录。

  要减小结果数据的大小，您可以使用投影表达式仅返回 `Path` 属性。

ComponentId	ParentId	GraphId	路径
CM8	CM4	CM1 #1	CM1|CM2|CM4|CM8

  路径

CM1|CM2|CM4|CM8

现在，使用管道 (“|”) 分割路径，然后取前 N-1 分量来获得祖先。

查询结果：CM8 的原级是 CM1、CM2、CM4。

- 查找组件的直系子组件。

  要获取 CM2 组件的所有直接子组件或下游单级组件，请使用查询 GSI1。`ParentId = "CM2"`查询将返回以下记录。

ParentId	ComponentId
CM2	CM4
	CM5

- 使用顶级组件查找所有下游子组件。

  要获取顶级组件 CM1 的所有子组件或下游组件，请使用和查询 GSI2`begins_with("Path", "CM1|")`，`GraphId = "CM1#1"`并使用带的投影表达式。`ComponentId`它将返回与该树相关的所有组件。

  本例中有一个树，CM1 作为顶部组件。实际上，同一个表中可能有数百万个顶部组件。

GraphId	ComponentId
	CM2
CM1 #1	CM3
	CM4
	CM5
	CM8
	CM9
	CM10

CM6

CM7

- 使用中间级别组件查找所有下游子组件。

  要递归获取组件 CM2 的所有子组件或下游组件，您有两个选择。您可以逐级递归查询，也可以查询 GSI2 索引。

  - 逐级递归查询 GSI1，直至到达子组件的最后一级。

    1. 使用 `ParentId = "CM2"` 查询 GSI1。这将返回以下记录。

ParentId	ComponentId
CM2	CM4
	CM5

    2. 同样，使用 `ParentId = "CM4"` 查询 GSI1。这将返回以下记录。

ParentId	ComponentId
CM4	CM8
	CM9

    3. 同样，使用 `ParentId = "CM5"` 查询 GSI1。这将返回以下记录。

       继续循环：查询每个 `ComponentId`，直至到达最后一级。当使用 `ParentId = "<ComponentId>"` 查询未返回任何结果时，上一个结果来自树的最后一级。

ParentId	ComponentId
CM5	CM10

    4. 合并所有结果。

       result= [cm4，cm5] + [cm8，cm9] + [CM10]

= [CM4、CM5、CM8、CM9、CM10]

- 查询 GSI2，它存储顶部组件（汽车或 CM1）的层次树。
  1. 首先，找到 CM2 的顶部组件或顶部原级和 Path。为此，使用 ComponentId = "CM2" 查询基表，在层次树中找到该组件的路径。选择GraphId和 Pat h 属性。查询将返回以下记录。

GraphId	路径
CM1 #1	CM1|CM2

  2. 使用查询 GSI2。GraphId = "CM1#1" AND BEGINS_WITH("Path", "CM1|CM2|")查询将返回以下结果。

GraphId	路径	ComponentId
CM1 #1	CM1|CM2|CM4	CM4
	CM1|CM2|CM5	CM5
	CM1|CM2|CM4|CM8	CM8
	CM1|CM2|CM4|CM9	CM9
	CM1|CM2|CM5|CM10	CM10

  3. 选择 ComponentId 属性以返回 CM2 的所有子组件。

步骤 7：验证数据模型

在此步骤中，业务用户验证查询结果并检查它们是否满足业务需求。您可以使用下表根据用户的要求检查访问模式。

问题	基本表/GSI	查询
作为用户，我想检索父组件 ID 的所有直接子组件。	GSI1	ParentId = "<Compone ntId>" （查找组件的直接子组件。）

作为用户，我想检索组件 ID 的所有子组件的递归列表。	GSI1 或 GSI2	GSI1：`ParentId = "<ComponentId>"` 或 GSI2：`GraphId = "<TopLevelComponentId>#N" AND BEGINS_WITH("Path", "<PATH_OF_Component>")` （使用顶部组件查找所有下级子组件，使用中间组件查找所有下级子组件。）
作为用户，我想查看原级组件。	基表	`ComponentId = "<ComponentId>"` ，然后选择 “路径” 属性。 （查找组件的原级。）

您也可以使用任何编程语言实现脚本（测试），以直接查询 DynamoDB 并将结果与预期结果进行比较。

第 8 步：查看成本估算

再次审查并完善成本估算。此外，向业务利益相关者进行验证并获得批准以进入下一步是一个不错的做法。

目标

- 定义容量模型，并估计 DynamoDB 成本，以完善步骤 2 中的成本估算。
- 获得业务分析师和利益相关者的最终财务批准。

过程

- 数据库工程师确定数据量的估计值。

- 数据库工程师确定数据传输要求。
- 数据库工程师定义所需的读取和写入容量单位。
- 业务分析师在按需容量模式和预置容量模式之间做出决定。
- 数据库工程师确定了 DynamoDB 自动扩缩的需求。
- 数据库工程师在中输入参数AWS Pricing Calculator。
- 数据库工程师向业务利益相关者提供最终的价格估算。
- 业务分析师和利益相关者批准或拒绝该解决方案。

步骤 9：部署数据模型

在这个具体的示例中，模型的部署是使用 NoSQL Workbench完成的，NoSQL Workbench 是一款用于现代数据库开发和操作的应用程序。使用此工具，您可以选择创建数据模型、上传数据并将其直接部署到您的AWS 账户。如果要实现此示例，则可以使用以下由 NoSQL Workbench 生成的AWS CloudFormation模板。

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Components:
    Type: 'AWS::DynamoDB::Table'
    Properties:
      KeySchema:
        - AttributeName: ComponentId
          KeyType: HASH
      AttributeDefinitions:
        - AttributeName: ComponentId
          AttributeType: S
        - AttributeName: ParentId
          AttributeType: S
        - AttributeName: GraphId
          AttributeType: S
        - AttributeName: Path
          AttributeType: S
      GlobalSecondaryIndexes:
        - IndexName: GS1
          KeySchema:
            - AttributeName: ParentId
              KeyType: HASH
            - AttributeName: ComponentId
              KeyType: RANGE
```

```
      Projection:
        ProjectionType: KEYS_ONLY
    - IndexName: GSI2
      KeySchema:
        - AttributeName: GraphId
          KeyType: HASH
        - AttributeName: Path
          KeyType: RANGE
      Projection:
        ProjectionType: INCLUDE
        NonKeyAttributes:
          - ComponentId
  BillingMode: PAY_PER_REQUEST
  TableName: Components
```

其他资源

有关 DynamoDB 的更多信息

- DynamoDB 定价
- DynamoDB 文档
- 适用于 DynamoDB 的 NoSQL 设计
- 写入分片
- 本地二级索引（LSI）
- 全局二级索引（GSI）
- 重载 GSI
- GSI 分片
- 使用 GSI 创建最终一致的副本
- 稀疏索引
- 物化聚合查询
- 时间序列设计模式
- 相邻列表设计模式
- 按需和预置容量模式
- DynamoDB 自动扩缩
- DynamoDB 生存时间 (TTL)
- 使用 DynamoDB 对游戏玩家数据进行建模（实验室）

AWS 服务

- AWS CloudFormation
- Amazon S3

工具

- AWS Pricing Calculator
- NoSQL Workbench for DynamoDB
- DynamoDB Local

- DynamoDB 和 AWS SDK

最佳实践

- 使用 DynamoDB 进行设计和架构的最佳实践（DynamoDB 文档）
- 使用二级索引的最佳实践（DynamoDB 文档）
- 存储大型项目和属性的最佳实践（DynamoDB 文档）
- 选择正确的 DynamoDB 分区键AWS（数据库博客）
- 如何设计亚马逊 DynamoDBGlobal 二级索引（AWS数据库博客）
- 适用于 Amazon DynamoDB 的 NoSQL Workbench 有哪些方面（Medium 网站）

AWS 一般资源

- AWS Prescriptive Guidance 网站
- AWS 文档
- AWS 一般参考

贡献者

本指南的贡献者包括：

- 卡米洛·冈萨雷斯，高级数据架构师，AWS
- Moinul Al-Mamun，高级大数据架构师，AWS
- 圣地亚哥·塞古拉，专业服务顾问，AWS
- Satheish Kumar Chandraprakasam，云应用程序架构师，AWS

文档历史记录

下表介绍了本指南的一些重要更改。如果您希望收到有关未来更新的通知，可以订阅 RSS 源。

变更	说明	日期
添加了“最佳实践”部分和分层数据建模示例。	我们添加了 DynamoDB 最佳实践摘要以及设计和 step-by-step 验证分层模型的示例。	2023 年 12 月 5 日
初次发布	—	2020 年 10 月 26 日

AWS Prescriptive Guidance 术语表

以下是 AWS Prescriptive Guidance 提供的策略、指南和模式中常用的术语。若要推荐词条，请使用术语表末尾的提供反馈链接。

数字

7 R

将应用程序迁移到云中的 7 种常见迁移策略。这些策略以 Gartner 于 2011 年确定的 5 R 为基础，包括以下内容：

- 重构/重新架构 - 充分利用云原生功能来提高敏捷性、性能和可扩展性，以迁移应用程序并修改其架构。这通常涉及到移植操作系统和数据库。示例：将本地 Oracle 数据库迁移到 Amazon Aurora PostgreSQL 兼容版。
- 更换平台 - 将应用程序迁移到云中，并进行一定程度的优化，以利用云功能。示例：将本地 Oracle 数据库迁移到 AWS 云中的 Amazon Relational Database Service（Amazon RDS）for Oracle。
- 重新购买 - 转换到其他产品，通常是从传统许可转向 SaaS 模式。示例：将客户关系管理（CRM）系统迁移到 Salesforce.com。
- 更换主机（直接迁移）- 将应用程序迁移到云中，无需进行任何更改即可利用云功能。示例：将本地 Oracle 数据库迁移到 AWS 云中 EC2 实例上的 Oracle。
- 重新定位（虚拟机监控器级直接迁移）- 将基础设施迁移到云中，无需购买新硬件、重写应用程序或修改现有操作。此迁移场景特定于 VMware Cloud on AWS，它支持本地环境和 AWS 之间的虚拟机兼容性和工作负载可移植性。在将基础设施迁移到 VMware Cloud on AWS 时，您可以在本地数据中心使用 VMware Cloud Foundation 技术。示例：将托管 Oracle 数据库的管理程序重新定位到 VMware Cloud on AWS。
- 保留（重访）- 将应用程序保留在源环境中。其中可能包括需要进行重大重构的应用程序，并且您希望将工作推迟到以后，以及您希望保留的遗留应用程序，因为迁移它们没有商业上的理由。
- 停用 - 停用或删除源环境中不再需要的应用程序。

A

ABAC

请参阅基于属性的访问控制。

抽象服务

参见托管服务。

酸

参见原子性、一致性、隔离性、耐久性。

主动-主动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步（通过使用双向复制工具或双写操作），两个数据库都在迁移期间处理来自连接应用程序的事务。这种方法支持小批量、可控的迁移，而不需要一次性割接。与主动-被动迁移相比，它更灵活，但需要更多的工作。

主动-被动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步，但在将数据复制到目标数据库时，只有源数据库处理来自连接应用程序的事务。目标数据库在迁移期间不接受任何事务。

聚合函数

一个 SQL 函数，它对一组行进行操作并计算该组的单个返回值。聚合函数的示例包括SUM和MAX。

AI

参见人工智能。

AIOps

参见人工智能运营。

匿名化

永久删除数据集中个人信息的过程。匿名化可以帮助保护个人隐私。匿名化数据不再被视为个人数据。

反模式

一种用于解决反复出现的问题的常用解决方案，而在这类问题中，此解决方案适得其反、无效或不如替代方案有效。

应用程序控制

一种安全方法，仅允许使用经批准的应用程序，以帮助保护系统免受恶意软件的侵害。

应用程序组合

有关组织使用的每个应用程序的详细信息的集合，包括构建和维护该应用程序的成本及其业务价值。这些信息是产品组合发现和分析过程的关键，有助于识别需要进行迁移、现代化和优化的应用程序并确定其优先级。

人工智能（AI）

计算机科学领域致力于使用计算技术执行通常与人类相关的认知功能，例如学习、解决问题和识别模式。有关更多信息，请参阅什么是人工智能？

人工智能运营（AIOps）

使用机器学习技术解决运营问题、减少运营事故和人为干预以及提高服务质量的过程。有关如何在 AWS 迁移策略中使用 AIOps 的更多信息，请参阅运营集成指南。

非对称加密

一种加密算法，使用一对密钥，一个公钥用于加密，一个私钥用于解密。您可以共享公钥，因为它不用于解密，但对私钥的访问应受到严格限制。

原子性、一致性、隔离性、持久性（ACID）

一组软件属性，即使在出现错误、电源故障或其他问题的情况下，也能保证数据库的数据有效性和操作可靠性。

基于属性的访问权限控制（ABAC）

根据用户属性（如部门、工作角色和团队名称）创建精细访问权限的做法。有关更多信息，请参阅 AWS Identity and Access Management（IAM）文档中的有关 AWS 的 ABAC。

权威数据源

存储主要数据版本的位置，被认为是最可靠的信息源。您可以将数据从权威数据源复制到其他位置，以便处理或修改数据，例如对数据进行匿名化、编辑或假名化。

可用区

一个 AWS 区域 中的不同位置，用于与其他可用区的故障隔离，并向同一区域中的其他可用区提供低成本、低延迟的网络连接。

AWS 云采用框架（AWS CAF）

AWS 的指导原则和最佳实践框架，旨在帮助组织制定高效且有效的计划来成功迁移到云。AWSCAF 将指导原则分为六个重点领域（角度）：业务、人员、治理、平台、安全和运营。业务、人员和治理角度侧重于业务技能和流程；平台、安全和运营角度侧重于技术技能和流程。例如，人员角度针对的是负责人力资源（HR）、人员配置职能和人员管理的利益相关者。从这个角度来看，AWS CAF 为人员发展、培训和沟通提供了指导，帮助组织为成功采用云做好准备。有关更多信息，请参阅 AWS CAF 网站和 AWS CAF 白皮书。

AWS Workload Qualification Framework（AWS WQF）

一种评估数据库迁移工作负载、推荐迁移策略并提供工作量估算的工具。AWSWQF 包含在 AWS Schema Conversion Tool（AWS SCT）中。它用来分析数据库架构和代码对象、应用程序代码、依赖关系和性能特征，并提供评测报告。

B

BCP

参见业务连续性计划。

行为图

一段时间内资源行为和交互的统一交互式视图。您可以使用 Amazon Detective 的行为图来检查失败的登录尝试、可疑的 API 调用和类似的操作。有关更多信息，请参阅 Detective 文档中的行为图中的数据。

大端序系统

一个先存储最高有效字节的系统。另请参见字节顺序。

二进制分类

一种预测二进制结果（两个可能的类别之一）的过程。例如，您的 ML 模型可能需要预测诸如“该电子邮件是否为垃圾邮件？”或“这个产品是书还是汽车？”之类的问题

bloom 筛选条件

一种概率性、内存高效的数据结构，用于测试元素是否为集合的成员。

分支

代码存储库的一个包含区域。在存储库中创建的第一个分支是主分支。您可以从现有分支创建新分支，然后在新分支中开发功能或修复错误。为构建功能而创建的分支通常称为功能分支。当功能可以发布时，将功能分支合并回主分支。有关更多信息，请参阅关于分支（GitHub 文档）。

破碎的玻璃通道

在特殊情况下，通过批准的流程，用户AWS 账户可以快速访问他们通常没有访问权限的内容。有关更多信息，请参阅 Well -Architected 指南中的“实施破碎玻璃程序”指示AWS器。

棕地策略

您环境中的现有基础设施。在为系统架构采用棕地策略时，您需要围绕当前系统和基础设施的限制来设计架构。如果您正在扩展现有基础设施，则可以将棕地策略和全新策略混合。

缓冲区缓存

存储最常访问的数据的内存区域。

业务能力

企业如何创造价值（例如，销售、客户服务或营销）。微服务架构和开发决策可以由业务能力驱动。有关更多信息，请参阅在 AWS 上运行容器化微服务白皮书中的围绕业务能力进行组织部分。

业务连续性计划（BCP）

一项计划，旨在应对大规模迁移等破坏性事件对运营的潜在影响，并使企业能够快速恢复运营。

C

CAF

参见AWS云采用框架。

CCoE

参见云卓越中心。

CDC

参见变更数据捕获。

更改数据捕获（CDC）

跟踪数据来源（如数据库表）的更改并记录有关更改的元数据的过程。您可以将 CDC 用于各种目的，例如审计或复制目标系统中的更改以保持同步。

混沌工程

故意引入故障或破坏性事件来测试系统的弹性。您可以使用 AWS Fault Injection Service(AWS FIS) 来执行实验，对您的AWS工作负载施加压力并评估其响应。

CI/CD

查看持续集成和持续交付。

分类

一种有助于生成预测的分类流程。分类问题的 ML 模型预测离散值。离散值始终彼此不同。例如，一个模型可能需要评估图像中是否有汽车。

客户端加密

在目标 AWS 服务 接收数据之前，在本地对数据进行加密。

云卓越中心（CCoE）

一个多学科团队，负责推动整个组织的云采用工作，包括开发云最佳实践、调动资源、制定迁移时间表、领导组织完成大规模转型。有关更多信息，请参阅 AWS 云企业战略博客上的 CCoE 帖子。

云计算

通常用于远程数据存储和 IoT 设备管理的云技术。云计算通常与边缘计算技术相关。

云运营模型

在 IT 组织中，一种用于构建、完善和优化一个或多个云环境的运营模型。有关更多信息，请参阅构建您的云运营模型。

云采用阶段

组织迁移到 AWS 云中时通常会经历四个阶段：

- 项目 - 出于概念验证和学习目的，开展一些与云相关的项目
- 基础 - 进行基础投资以扩大云采用率（例如，创建登录区、定义 CCoE、建立运营模型）
- 迁移 - 迁移单个应用程序
- 重塑 - 优化产品和服务，在云中创新

Stephen Orban 在 AWS 云企业战略博客上发表的博客文章云优先之旅和采用阶段中对这些阶段进行了定义。有关它们与 AWS 迁移策略的关系的信息，请参阅迁移准备指南。

CMDB

参见配置管理数据库。

代码存储库

通过版本控制过程存储和更新源代码和其他资产（如文档、示例和脚本）的位置。常见的云存储库包括 GitHub 或AWS CodeCommit。每个版本的代码都称为一个分支。在微服务结构中，每个存储库都专门用于一个功能。单个 CI/CD 管道可以使用多个存储库。

冷缓存

一种空的、填充不足或包含过时或不相关数据的缓冲区缓存。这会影响性能，因为数据库实例必须从主内存或磁盘读取，这比从缓冲区缓存读取要慢。

冷数据

很少访问的数据，且通常是历史数据。查询此类数据时，通常可以接受慢速查询。将这些数据转移到性能较低且成本更低的存储层或类别可以降低成本。

计算机视觉

机器用来在图像中以比肩或高于人类水平的精度辨识人、地点和事物的 AI 领域。它往往用深度学习模型构建，使得可自动从单幅图像或一系列图像提取、分析、分类和理解有用信息。

配置管理数据库（CMDB）

一种存储库，用于存储和管理有关数据库及其 IT 环境的信息，包括硬件和软件组件及其配置。您通常在迁移的产品组合发现和分析阶段使用来自 CMDB 的数据。

合规性包

一系列 AWS Config 规则和修复操作，您可以将其组合起来以自定义合规性和安全性检查。您可以使用 YAML 模板，将合规性包作为单个实体部署到 AWS 账户 区域中，或者跨组织部署。有关更多信息，请参阅 AWS Config 文档中的合规性包。

持续集成和持续交付（CI/CD）

自动执行软件发布过程的源代码、构建、测试、暂存和生产阶段的过程。CI/CD 通常被描述为管道。CI/CD 可以帮助您实现流程自动化、提高工作效率、改善代码质量并加快交付速度。有关更多信息，请参阅持续交付的优势。CD 也可以表示持续部署。有关更多信息，请参阅持续交付与持续部署。

D

静态数据

网络中静止的数据，例如存储中的数据。

数据分类

根据网络中数据的关键性和敏感性对其进行识别和分类的过程。它是任何网络安全风险管理策略的关键组成部分，因为它可以帮助您确定对数据的适当保护和保留控制。数据分类是 AWS Well-Architected Framework 中安全支柱的一个组成部分。有关详细信息，请参阅数据分类。

数据漂移

生产数据与用来训练机器学习模型的数据之间的有意义差异，或者输入数据随时间推移的有意义变化。数据漂移可能降低机器学习模型预测的整体质量、准确性和公平性。

传输中数据

在网络中主动移动的数据，例如在网络资源之间移动的数据。

数据最少化

仅收集并处理绝对必要数据的原则。在 AWS Cloud 中践行数据最少化可以降低隐私风险、成本和您的分析碳足迹。

数据边界

AWS环境中的一组预防性防护措施，可帮助确保只有可信身份才能访问来自预期网络的可信资源。有关更多信息，请参阅在上构建数据边界。AWS

数据预处理

将原始数据转换为 ML 模型易于解析的格式。预处理数据可能意味着删除某些列或行，并处理缺失、不一致或重复的值。

数据溯源

在数据的整个生命周期跟踪其来源和历史的过程，例如数据如何生成、传输和存储。

数据主体

正在收集和处理其数据的个人。

数据仓库

一种支持商业智能（例如分析）的数据管理系统。数据仓库通常包含大量历史数据，通常用于查询和分析。

数据库定义语言（DDL）

在数据库中创建或修改表和对象结构的语句或命令。

数据库操作语言（DML）

在数据库中修改（插入、更新和删除）信息的语句或命令。

DDL

参见数据库定义语言。

深度融合

组合多个深度学习模型进行预测。您可以使用深度融合来获得更准确的预测或估算预测中的不确定性。

深度学习

一个 ML 子字段使用多层人工神经网络来识别输入数据和感兴趣的目标变量之间的映射。

defense-in-depth

一种信息安全方法，经过深思熟虑，在整个计算机网络中分层实施一系列安全机制和控制措施，以保护网络及其中数据的机密性、完整性和可用性。当您在 AWS 上采用此策略时，您可以在 AWS Organizations 结构的不同层添加多种控制措施，来保护资源。例如，一种 defense-in-depth 方法可以结合多因素身份验证、网络分段和加密。

委托管理员

在 AWS Organizations 中，兼容服务可以注册 AWS 成员账户来管理组织的账户，并管理该服务的权限。此账户被称为该服务的委托管理员。有关更多信息和兼容服务列表，请参阅 AWS Organizations 文档中使用 AWS Organizations 的服务。

部署

使应用程序、新功能或代码修复在目标环境中可用的过程。部署涉及在代码库中实现更改，然后在应用程序的环境中构建和运行该代码库。

开发环境

参见环境。

侦测性控制

一种安全控制，在事件发生后进行检测、记录日志和发出警报。这些控制是第二道防线，提醒您注意绕过现有预防性控制的安全事件。有关更多信息，请参阅在 AWS 上实施安全控制中的侦测性控制。

开发价值流映射 (DVSM)

用于识别对软件开发生命周期中的速度和质量产生不利影响的限制因素并确定其优先级的流程。DVSM 扩展了最初为精益生产实践设计的价值流映射流程。其重点关注在软件开发过程中创造和转移价值所需的步骤和团队。

数字孪生

真实世界系统的虚拟再现，如建筑物、工厂、工业设备或生产线。数字孪生支持预测性维护、远程监控和生产优化。

维度表

在星型架构中，一种较小的表，其中包含事实表中定量数据的数据属性。维度表属性通常是文本字段或行为类似于文本的离散数字。这些属性通常用于查询约束、筛选和结果集标注。

灾难

阻止工作负载或系统在其主要部署位置实现其业务目标的事件。这些事件可能是自然灾害、技术故障或人为操作的结果，例如无意的配置错误或恶意软件攻击。

灾难恢复 (DR)

您用来最大限度地减少灾难造成的停机时间和数据丢失的策略和流程。有关更多信息，请参阅 AWS 上工作负载的灾难恢复：AWS Well-Architected Framework 中的云中恢复。

DML

参见数据库操作语言。

领域驱动设计

一种开发复杂软件系统的方法，通过将其组件连接到每个组件所服务的不断发展的领域或核心业务目标。Eric Evans 在其著作领域驱动设计：软件核心复杂性应对之道（Boston: Addison-Wesley Professional, 2003）中介绍了这一概念。有关如何将领域驱动设计与 strangler fig 模式结合使用的信息，请参阅使用容器和 Amazon API Gateway 逐步将原有的 Microsoft ASP.NET（ASMX）Web 服务现代化。

DR

参见灾难恢复。

漂移检测

跟踪与基准配置的偏差。例如，您可以使用AWS CloudFormation来检测系统资源中的偏差，也可以使用AWS Control Tower来检测着陆区中可能影响监管要求合规性的变化。

DVSM

参见开发价值流映射。

E

EDA

参见探索性数据分析。

边缘计算

该技术可提高位于 IoT 网络边缘的智能设备的计算能力。与云计算相比，边缘计算可以减少通信延迟并缩短响应时间。

加密

一种将人类可读的纯文本数据转换为密文的计算过程。

加密密钥

由加密算法生成的随机位的加密字符串。密钥的长度可能有所不同，而且每个密钥都设计为不可预测且唯一。

字节顺序

字节在计算机内存中的存储顺序。大端序系统先存储最高有效字节。小端序系统先存储最低有效字节。

端点

参见服务端点。

端点服务

一种可以在虚拟私有云（VPC）中托管，与其他用户共享的服务。您可以使用 AWS PrivateLink 创建端点服务，并将权限授予其他 AWS 账户 或 AWS Identity and Access Management（IAM）主体。这些账户或主体可通过创建接口 VPC 端点来私密地连接到您的端点服务。有关更多信息，请参阅 Amazon Virtual Private Cloud（Amazon VPC）文档中的创建端点服务。

信封加密

用另一个加密密钥对加密密钥进行加密的过程。有关更多信息，请参阅 AWS Key Management Service（AWS KMS）文档中的信封加密。

environment

正在运行的应用程序的实例。以下是云计算中常见的环境类型：

- 开发环境 — 正在运行的应用程序的实例，只有负责维护应用程序的核心团队才能使用。开发环境用于测试更改，然后再将其提升到上层环境。这类环境有时称为测试环境。
- 下层环境 — 应用程序的所有开发环境，比如用于初始构建和测试的环境。
- 生产环境 — 最终用户可以访问的正在运行的应用程序的实例。在 CI/CD 管道中，生产环境是最后一个部署环境。
- 上层环境 — 除核心开发团队以外的用户可以访问的所有环境。这可能包括生产环境、预生产环境和用户验收测试环境。

epic

在敏捷方法学中，有助于组织工作和确定优先级的功能类别。epics 提供了对需求和实施任务的总体描述。例如，AWS CAF security epics 包括身份和访问管理、侦测性控制、基础设施安全、数据保护和事件响应。有关 AWS 迁移策略中 epics 的更多信息，请参阅计划实施指南。

探索性数据分析（EDA）

分析数据集以了解其主要特征的过程。您收集或汇总数据，并进行初步调查，以发现模式、检测异常并检查假定情况。EDA 通过计算汇总统计数据和创建数据可视化得以执行。

F

事实表

星形架构中的中心表。它存储有关业务运营的定量数据。通常，事实表包含两种类型的列：包含度量的列和包含维度表外键的列。

失败得很快

一种使用频繁和增量测试来缩短开发生命周期的理念。这是敏捷方法的关键部分。

故障隔离边界

在中AWS Cloud，诸如可用区AWS 区域、控制平面或数据平面之类的边界，它限制了故障的影响并有助于提高工作负载的弹性。有关更多信息，请参阅AWS故障隔离边界。

功能分支

参见分支。

特征

您用来进行预测的输入数据。例如，在制造环境中，特征可能是定期从生产线捕获的图像。

特征重要性

特征对于模型预测的重要性。这通常表示为数值分数，可以通过各种技术进行计算，例如 Shapley 加法解释（SHAP）和积分梯度。有关更多信息，请参阅机器学习模型的可解释性：AWS。

功能转换

为 ML 流程优化数据，包括使用其他来源丰富数据、扩展值或从单个数据字段中提取多组信息。这使得 ML 模型能从数据中获益。例如，如果您将“2021-05-27 00:15:37”日期分解为“2021”、“五月”、“星期四”和“15”，则可以帮助学习与不同数据成分相关的算法学习精细模式。

FGAC

请参阅精细的访问控制。

精细访问控制 (FGAC)

使用多个条件允许或拒绝访问请求。

快闪迁移

一种数据库迁移方法，它使用连续的数据复制，通过更改数据捕获在尽可能短的时间内迁移数据，而不是使用分阶段的方法。目标是将停机时间降至最低。

G

地理封锁

请参阅地理限制。

地理限制（地理阻止）

在 Amazon 中 CloudFront，一种阻止特定国家/地区的用户访问内容分发的选项。您可以使用允许列表或阻止列表来指定已批准和已禁止的国家/地区。有关更多信息，请参阅 CloudFront 文档中的限制内容的地理分布。

GitFlow 工作流程

一种方法，在这种方法中，下层和上层环境在源代码存储库中使用不同的分支。Gitflow 工作流程被认为是传统的，而基于主干的工作流程是现代的首选方法。

全新策略

在新环境中缺少现有基础设施。在对系统架构采用全新策略时，您可以选择所有新技术，而不受对现有基础设施（也称为棕地）兼容性的限制。如果您正在扩展现有基础设施，则可以将棕地策略和全新策略混合。

防护机制

一种高级规则，用于跨组织单位（OU）管理资源、策略和合规性。预防性防护机制会执行策略以确保符合合规性标准。它们是使用服务控制策略和 IAM 权限边界实现的。侦测性防护机制会检测策略违规和合规性问题，并生成警报以进行修复。它们通过使用AWS Config、、Amazon、AWS Security Hub GuardDutyAWS Trusted Advisor、Amazon Inspector 和自定义AWS Lambda支票来实现。

H

HA

参见高可用性。

异构数据库迁移

将源数据库迁移到使用不同数据库引擎的目标数据库（例如，从 Oracle 迁移到 Amazon Aurora）。异构迁移通常是重新架构工作的一部分，而转换架构可能是一项复杂的任务。AWS 提供了 AWS SCT 来帮助实现架构转换。

高可用性 (HA)

在遇到挑战或灾难时，工作负载无需干预即可连续运行的能力。HA 系统旨在自动进行故障转移、持续提供良好性能，并以最小的性能影响处理不同负载和故障。

历史数据库现代化

一种用于实现运营技术 (OT) 系统现代化和升级以更好满足制造业需求的方法。历史数据库是一种用于收集和存储工厂中各种来源数据的数据库。

同构数据库迁移

将源数据库迁移到共享同一数据库引擎的目标数据库（例如，从 Microsoft SQL Server 迁移到 Amazon RDS for SQL Server）。同构迁移通常是更换主机或更换平台工作的一部分。您可以使用本机数据库实用程序来迁移架构。

热数据

经常访问的数据，例如实时数据或近期的转化数据。这些数据通常需要高性能存储层或存储类别才能提供快速的查询响应。

修补程序

针对生产环境中关键问题的紧急修复。由于其紧迫性，修补程序通常是在典型的 DevOps 发布工作流程之外进行的。

hypercare 周期

割接之后，迁移团队立即管理和监控云中迁移的应用程序以解决任何问题的时间段。通常，这个周期持续 1-4 天。在 hypercare 周期结束时，迁移团队通常会将应用程序的责任移交给云运营团队。

I

IaC

参见基础架构即代码。

基于身份的策略

附加到一个或多个 IAM 主体的策略，用于定义它们在 AWS Cloud 环境中的权限。

空闲应用程序

90 天内平均 CPU 和内存使用率在 5% 到 20% 之间的应用程序。在迁移项目中，通常会停用这些应用程序或将其保留在本地。

IIoT

参见工业物联网。

不可变的基础架构

一种为生产工作负载部署新基础架构，而不是更新、修补或修改现有基础架构的模型。不可变基础架构本质上比可变基础架构更一致、更可靠、更可预测。有关更多信息，请参阅 Well-Architected Framework 中的使用不可变基础架构AWS部署最佳实践。

入站（入口）VPC

在 AWS 多账户架构中，一种用于接受、检查和路由来自应用程序外部的网络连接的 VPC。AWS 安全参考架构建议使用入站、出站和检查 VPC 设置网络账户，保护应用程序与广泛的互联网之间的双向接口。

增量迁移

一种割接策略，在这种策略中，您可以将应用程序分成小部分进行迁移，而不是一次性完整割接。例如，您最初可能只将几个微服务或用户迁移到新系统。在确认一切正常后，您可以逐步迁移其他微服务或用户，直到停用遗留系统。这种策略降低了大规模迁移带来的风险。

基础设施

应用程序环境中包含的所有资源和资产。

基础设施即代码（IaC）

通过一组配置文件预置和管理应用程序基础设施的过程。IaC 旨在帮助您集中管理基础设施、实现资源标准化和快速扩展，使新环境具有可重复性、可靠性和一致性。

工业物联网（IIoT）

在工业领域使用联网的传感器和设备，例如制造业、能源、汽车、医疗保健、生命科学和农业。有关更多信息，请参阅制定工业物联网（IIoT）数字化转型策略。

检查 VPC

在 AWS 多账户架构中，一种用于管理 VPC（相同或不同的 AWS 区域）、互联网和本地网络之间的网络流量检查的集中式 VPC。AWS 安全参考架构建议使用入站、出站和检查 VPC 设置网络账户，保护应用程序与广泛的互联网之间的双向接口。

物联网（IoT）

由带有嵌入式传感器或处理器的连接物理对象组成的网络，这些传感器或处理器通过互联网或本地通信网络与其他设备和系统进行通信。有关更多信息，请参阅什么是 IoT？

可解释性

它是机器学习模型的一种特征，描述了人类可以理解模型的预测如何取决于其输入的程度。有关更多信息，请参阅使用 AWS 实现机器学习模型的可解释性。

IoT

参见物联网。

IT 信息库（ITIL）

提供 IT 服务并使这些服务符合业务要求的一套最佳实践。ITIL 是 ITSM 的基础。

IT 服务管理（ITSM）

为组织设计、实施、管理和支持 IT 服务的相关活动。有关将云运营与 ITSM 工具集成的信息，请参阅运营集成指南。

ITIL

请参阅 IT 信息库。

ITSM

请参阅 IT 服务管理。

L

基于标签的访问控制 (LBAC)

强制访问控制 (MAC) 的一种实施方式，其中明确为用户和数据本身分配了安全标签值。用户安全标签和数据安全标签之间的交集决定了用户可以看到哪些行和列。

登录区

登录区是一个架构完善、可扩展且安全的多账户 AWS 环境。这是一个起点，您的组织可以从这里放心地在安全和基础设施环境中快速启动和部署工作负载和应用程序。有关登录区的更多信息，请参阅设置安全且可扩展的多账户 AWS 环境。

大规模迁移

迁移 300 台或更多服务器。

LBAC

请参阅基于标签的访问控制。

最低权限

授予执行任务所需的最低权限的最佳安全实践。有关更多信息，请参阅 IAM 文档中的应用最低权限许可。

直接迁移

见 7 R。

小端序系统

一个先存储最低有效字节的系统。另请参见字节顺序。

下层环境

参见环境。

M

机器学习（ML）

一种使用算法和技术进行模式识别和学习的人工智能。ML 对记录的数据（例如物联网（IoT）数据）进行分析和学习，以生成基于模式的统计模型。有关更多信息，请参阅机器学习。

主分支

参见分支。

托管服务

AWS 服务它AWS运行基础设施层、操作系统和平台，您可以访问端点来存储和检索数据。亚马逊简单存储服务 (Amazon S3) Service 和 Amazon DynamoDB 就是托管服务的示例。这些服务也称为抽象服务。

MAP

参见迁移加速计划。

机制

一个完整的过程，在此过程中，您可以创建工具，推动工具的采用，然后检查结果以进行调整。机制是一种在运行过程中自我增强和改进的循环。有关更多信息，请参阅在 Well-Architect AWS ed 框架中构建机制。

成员账户

除管理账户外，属于 AWS Organizations 中的组织的所有 AWS 账户。一个账户一次只能是一个组织的成员。

微服务

一种小型独立服务，通过明确定义的 API 进行通信，通常由小型独立团队拥有。例如，保险系统可能包括映射到业务能力（如销售或营销）或子域（如购买、理赔或分析）的微服务。微服务的好处包括敏捷、灵活扩展、易于部署、可重复使用的代码和恢复能力。有关更多信息，请参阅使用 AWS 无服务器服务集成微服务。

微服务架构

一种使用独立组件构建应用程序的方法，这些组件将每个应用程序进程作为微服务运行。这些微服务使用轻量级 API 通过明确定义的接口进行通信。该架构中的每个微服务都可以更新、部署和扩展，以满足对应用程序特定功能的需求。有关更多信息，请参阅在 AWS 上实现微服务。

迁移加速计划（MAP）

一项提供咨询支持、培训和服务的 AWS 计划，旨在帮助组织为迁移到云奠定坚实的运营基础，并抵消迁移的初始成本。MAP 提供了一种以系统的方式执行遗留迁移的迁移方法，以及一套用于自动执行和加速常见迁移场景的工具。

大规模迁移

将大部分应用程序组合分波迁移到云中的过程，在每一波中以更快的速度迁移更多应用程序。本阶段使用从早期阶段获得的最佳实践和经验教训，实施由团队、工具和流程组成的迁移工厂，通过自动化和敏捷交付简化工作负载的迁移。这是 AWS 迁移策略的第三阶段。

迁移工厂

跨职能团队，通过自动化、敏捷的方法简化工作负载迁移。迁移工厂团队通常包括运营、业务分析师和所有者、迁移工程师、开发 DevOps 人员和冲刺专业人员。20% 到 50% 的企业应用程序组合由可通过工厂方法优化的重复模式组成。有关更多信息，请参阅本内容集中有关迁移工厂的讨论和云迁移工厂指南。

迁移元数据

有关完成迁移所需的应用程序和服务器的信息。每种迁移模式都需要一套不同的迁移元数据。迁移元数据的示例包括目标子网、安全组和 AWS 账户。

迁移模式

一种可重复的迁移任务，详细列出了迁移策略、迁移目标以及所使用的迁移应用程序或服务。示例：使用 AWS Application Migration Service 将主机迁移到 Amazon EC2。

迁移组合评测（MPA）

一种在线工具，提供了用于验证迁移到 AWS 云的业务用例的信息。MPA 提供了详细的组合评测（服务器规模调整、定价、TCO 比较、迁移成本分析）以及迁移计划（应用程序数据分析和数据收集、应用程序分组、迁移优先级排序和波次规划）。MPA 工具（需要登录）向所有 AWS 顾问和 APN 合作伙伴顾问免费提供。

迁移准备情况评测（MRA）

使用 AWS CAF 深入了解组织的云就绪状态，找出优势和劣势，并制定行动计划来弥补发现的差距。有关更多信息，请参阅迁移准备指南。MRA 是 AWS 迁移策略的第一阶段。

迁移策略

将工作负载迁移到 AWS 云中的方法。有关更多信息，请参阅此词汇表中的 7 R 条目和动员组织以加快大规模迁移。

ML

参见机器学习。

MPA

参见迁移组合评估。

现代化

将过时的（原有的或单体）应用程序及其基础设施转变为云中敏捷、弹性和高度可用的系统，以降低成本、提高效率和利用创新。有关详细信息，请参阅在 AWS 云中实现应用程序现代化的策略。

现代化准备情况评估

一种评估方式，有助于确定组织应用程序的现代化准备情况；确定收益、风险和依赖关系；确定组织能够在多大程度上支持这些应用程序的未来状态。评估结果是目标架构的蓝图、详细说明现代化进程发展阶段和里程碑的路线图以及解决已发现差距的行动计划。有关详细信息，请参阅在 AWS 云中评估应用程序的现代化准备情况。

单体应用程序（单体式）

作为具有紧密耦合进程的单个服务运行的应用程序。单体应用程序有几个缺点。如果某个应用程序功能的需求激增，则必须扩展整个架构。随着代码库的增长，添加或改进单体应用程序的功能也会变得更加复杂。若要解决这些问题，可以使用微服务架构。有关更多信息，请参阅将单体分解为微服务。

多分类器

一种帮助为多个类别生成预测（预测两个以上结果之一）的过程。例如，ML 模型可能会询问“这个产品是书、汽车还是手机？”或“此客户最感兴趣什么类别的产品？”

可变基础架构

一种用于更新和修改现有生产工作负载基础架构的模型。为了提高一致性、可靠性和可预测性，Well-Architect AWS ed Framework 建议使用不可变基础设施作为最佳实践。

O

OAC

请参阅源站访问控制。

OAI

参见源访问身份。

OCM

参见组织变更管理。

离线迁移

一种迁移方法，在这种方法中，源工作负载会在迁移过程中停止运行。这种方法会延长停机时间，通常用于小型非关键工作负载。

OI

参见运营集成。

OLA

参见运营层协议。

在线迁移

一种迁移方法，在这种方法中，源工作负载无需离线即可复制到目标系统。在迁移过程中，连接工作负载的应用程序可以继续运行。这种方法的停机时间为零或最短，通常用于关键生产工作负载。

运营级别协议（OLA）

一项协议，阐明了 IT 职能部门承诺相互交付的内容，以支持服务水平协议（SLA）。

运营准备情况审查 (ORR)

一份问题清单和相关的最佳实践，可帮助您理解、评估、预防或缩小事件和可能的故障的范围。有关更多信息，请参阅 Well-Architecte AWS d Frame work 中的运营准备情况评估 (ORR)。

运营整合（OI）

在云中实现运营现代化的过程，包括就绪计划、自动化和集成。有关更多信息，请参阅运营整合指南。

组织跟踪

由 AWS CloudTrail 创建的跟踪，用于记录 AWS Organizations 中的组织的所有 AWS 账户 事件。该跟踪是在每个 AWS 账户 中创建的，属于组织的一部分，并跟踪每个账户的活动。有关更多信息，请参阅 CloudTrail文档中的为组织创建跟踪。

组织变革管理（OCM）

一个从人员、文化和领导力角度管理重大、颠覆性业务转型的框架。OCM 通过加快变革采用、解决过渡问题以及推动文化和组织变革，帮助组织为新系统和战略做好准备和过渡。在 AWS 迁移策略中，这个框架称为人员加速，因为云采用项目需要快速的变革。有关更多信息，请参阅 OCM 指南。

来源访问控制（OAC）

在中 CloudFront，一个增强的选项，用于限制访问以保护您的亚马逊简单存储服务 (Amazon S3) 内容。OAC 支持所有 AWS 区域 中的 S3 存储桶、使用 AWS KMS 的服务器端加密（SSE-KMS）以及对 S3 存储桶的动态 PUT 和 DELETE 请求。

来源访问身份（OAI）

在中 CloudFront，一个用于限制访问权限以保护您的 Amazon S3 内容的选项。当您使用 OAI 时，CloudFront 会创建一个 Amazon S3 可以对其进行身份验证的委托人。经过身份验证的委托人只能通过特定 CloudFront 分配访问 S3 存储桶中的内容。另请参阅 OAC，其中提供了更精细和增强的访问控制。

或者

参见运营准备情况审查。

出站（出口）VPC

在 AWS 多账户架构中，一种用于处理从应用程序内部启动的网络连接的 VPC。AWS 安全参考架构建议使用入站、出站和检查 VPC 设置网络账户，保护应用程序与广泛的互联网之间的双向接口。

P

权限边界

附加到 IAM 主体的 IAM 管理策略，用于设置用户或角色可以拥有的最大权限。有关更多信息，请参阅 IAM 文档中的权限边界。

个人身份信息 (PII)

直接查看其他相关数据或与之配对时可用于合理推断个人身份的信息。PII 的示例包括姓名、地址和联系信息。

PII

查看个人身份信息。

playbook

一套预定义的步骤，用于捕获与迁移相关的工作，例如在云中交付核心运营功能。playbook 可以采用脚本、自动化运行手册的形式，也可以是操作现代化环境所需的流程或步骤的摘要。

策略

一个对象，可以在中定义权限（参见基于身份的策略）、指定访问条件（参见基于资源的策略）或定义组织中所有账户的最大权限AWS Organizations（参见服务控制策略）。

多语言持久性

根据数据访问模式和其他要求，独立选择微服务的数据存储技术。如果您的微服务采用相同的数据存储技术，它们可能会遇到实现难题或性能不佳。如果微服务使用最适合其需求的数据存储，则可以更轻松地实现微服务，并获得更好的性能和可扩展性。有关更多信息，请参阅在微服务中实现数据持久性。

组合评测

一个发现、分析和确定应用程序组合优先级以规划迁移的过程。有关更多信息，请参阅评估迁移准备情况。

谓词

返回`true`或的查询条件`false`，通常位于子`WHERE`句中。

谓词下推

一种数据库查询优化技术，可在传输前筛选查询中的数据。这减少了必须从关系数据库检索和处理的数据量，并提高了查询性能。

预防性控制

一种安全控制，旨在防止事件发生。这些控制是第一道防线，帮助防止未经授权的访问或对网络的意外更改。有关更多信息，请参阅在 AWS 上实施安全控制中的预防性控制。

主体

AWS 中可执行操作并访问资源的实体。该实体通常是 AWS 账户、IAM 角色或用户的根用户。有关更多信息，请参阅 IAM 文档中角色术语和概念中的主体。

隐私设计

一种贯穿整个工程化过程考虑隐私的系统工程方法。

私有托管区

私有托管区就是一个容器，其中包含的信息说明您希望 Amazon Route 53 如何响应一个或多个 VPC 中的某个域及其子域的 DNS 查询。有关更多信息，请参阅 Route 53 文档中的私有托管区的使用。

主动控制

一种安全控制措施，旨在防止部署不合规的资源。这些控件会在资源置备之前对其进行扫描。如果资源与控件不兼容，则不会对其进行配置。有关更多信息，请参阅AWS Control Tower文档中的控制参考指南，并参见在上实施安全控制中的主动控制AWS。

生产环境

参见环境。

假名化

用占位符值替换数据集中个人标识符的过程。假名化可以帮助保护个人隐私。假名化数据仍被视为个人数据。

Q

查询计划

一系列步骤，例如指令，用于访问 SQL 关系数据库系统中的数据。

查询计划回归

当数据库服务优化程序选择的最佳计划不如数据库环境发生特定变化之前时。这可能是由统计数据、约束、环境设置、查询参数绑定更改和数据库引擎更新造成的。

R

RACI 矩阵

参见 “负责任、负责、咨询、知情”（RACI）。

勒索软件

一种恶意软件，旨在阻止对计算机系统或数据的访问，直到付款为止。

RASCI 矩阵

参见 “负责任、负责、咨询、知情”（RACI）。

RCAC

请参阅行和列访问控制。

只读副本

用于只读目的的数据库副本。您可以将查询路由到只读副本，以减轻主数据库的负载。

重新设计架构

见 7 R。

恢复点目标 (RPO)

自上一个数据恢复点以来可接受的最长时间。这决定了从上一个恢复点到服务中断之间可接受的数据丢失情况。

恢复时间目标 (RTO)

服务中断和服务恢复之间可接受的最大延迟。

重构

见 7 R。

区域

地理区域中的 AWS 资源集合。每个 AWS 区域 是孤立的，独立于其他的区域，以提供容错能力、稳定性和弹性。有关更多信息，请参阅在 AWS 一般参考 中管理 AWS 区域。

回归

一种预测数值的 ML 技术。例如，要解决“这套房子的售价是多少？”的问题 ML 模型可以使用线性回归模型，根据房屋的已知事实（如建筑面积）来预测房屋的销售价格。

重新托管

见 7 R。

版本

在部署过程中，推动生产环境变更的行为。

搬迁

见 7 R。

更换平台

见 7 R。

回购

见 7 R。

基于资源的策略

一种附加到资源的策略，例如 AmazonS3 存储桶、端点或加密密钥。此类策略指定了允许哪些主体访问、支持的操作以及必须满足的任何其他条件。

责任、问责、咨询和知情（RACI）矩阵

定义参与迁移活动和云运营的所有各方的角色和责任的矩阵。矩阵名称源自矩阵中定义的责任类型：负责 (R)、问责 (A)、咨询 (C) 和知情 (I)。支持 (S) 类型是可选的。如果包括支持，则该矩阵称为 RASCI 矩阵，如果将其排除在外，则称为 RACI 矩阵。

响应性控制

一种安全控制，旨在推动对不良事件或偏离安全基线的情况进行修复。有关更多信息，请参阅在 AWS 上实施安全控制中的响应性控制。

保留

见 7 R。

退休

见 7 R。

旋转

定期更新密钥以使攻击者更难访问凭据的过程。

行列访问控制 (RCAC)

使用已定义访问规则的基本、灵活的 SQL 表达式。RCAC 由行权限和列掩码组成。

RPO

参见恢复点目标。

RTO

参见恢复时间目标。

运行手册

执行特定任务所需的一套手动或自动程序。它们通常是为了简化重复性操作或高错误率的程序而设计的。

S

SAML 2.0

许多身份提供商 (IdPs) 使用的开放标准。此功能可实现联合单点登录（SSO），因此用户可以登录 AWS Management Console 或调用 AWS API 操作，而无需在 IAM 中为组织中的每个人都创建用户。有关基于 SAML 2.0 的联合身份验证的更多信息，请参阅 IAM 文档中的关于基于 SAML 2.0 的联合身份验证。

SCP

参见服务控制政策。

secret

在中AWS Secrets Manager，您以加密形式存储的机密或受限信息，例如密码或用户凭证。它由密钥值及其元数据组成。密钥值可以是二进制、单个字符串或多个字符串。有关更多信息，请参阅 Sec rets Manager 文档中的密钥。

安全控制

一种技术或管理防护机制，可防止、检测或降低威胁行为体利用安全漏洞的能力。安全控制主要有四种类型：预防性、侦测、响应式和主动式。

安全加固

缩小攻击面，使其更能抵御攻击的过程。这可能包括删除不再需要的资源、实施授予最低权限的最佳安全实践或停用配置文件中不必要的功能等操作。

安全信息和事件管理（SIEM）系统

结合了安全信息管理（SIM）和安全事件管理（SEM）系统的工具和服务。SIEM 系统会收集、监控和分析来自服务器、网络、设备和其他来源的数据，以检测威胁和安全漏洞，并生成警报。

安全响应自动化

一种预定义和编程的操作，旨在自动响应或修复安全事件。这些自动化可作为侦探或响应式安全控制措施，帮助您实施AWS安全最佳实践。自动响应操作的示例包括修改 VPC 安全组、修补 Amazon EC2 实例或轮换证书。

服务器端加密

由接收数据的 AWS 服务 在目的地对数据进行加密。

服务控制策略（SCP）

一种策略，用于集中控制 AWS Organizations 的组织中所有账户的权限。SCP 为管理员可以委托给用户或角色的操作定义了防护机制或设定了限制。您可以将 SCP 用作允许列表或拒绝列表，指定允许或禁止哪些服务或操作。有关更多信息，请参阅 AWS Organizations 文档中的服务控制策略。

服务端点

AWS 服务 的入口点的 URL。您可以使用端点，通过编程方式连接到目标服务。有关更多信息，请参阅 AWS 一般参考 中的 AWS 服务 端点。

服务水平协议（SLA）

一份协议，阐明了 IT 团队承诺向客户交付的内容，比如服务正常运行时间和性能。

服务级别指示器 (SLI)

对服务性能方面的衡量，例如其错误率、可用性或吞吐量。

服务级别目标 (SLO)

代表服务运行状况的目标指标，由服务级别指标衡量。

责任共担模式

一种描述您在云安全性和合规性方面与 AWS 共担的责任模型。AWS 负责云的安全，而您则负责云中的安全。有关更多信息，请参阅责任共担模式。

暹粒

参见安全信息和事件管理系统。

单点故障 (SPOF)

应用程序的单个关键组件出现故障，可能会中断系统。

SLA

参见服务级别协议。

SLI

参见服务级别指标。

SLO

参见服务级别目标。

split-and-seed 模型

一种扩展和加速现代化项目的模式。随着新功能和产品发布的定义，核心团队会拆分以创建新的产品团队。这有助于扩展组织的能力和服务，提高开发人员的工作效率，支持快速创新。有关更多信息，请参阅中的分阶段实现应用程序现代化的方法。AWS Cloud

恶作剧

参见单点故障。

星型架构

一种数据库组织结构，它使用一个大型事实表来存储交易数据或测量数据，并使用一个或多个较小的维度表来存储数据属性。此结构专为在数据仓库中使用或用于商业智能目的而设计。

strangler fig 模式

一种通过逐步重写和替换系统功能直至可以停用原有的系统来实现单体系统现代化的方法。这种模式用无花果藤作为类比，这种藤蔓成长为一棵树，最终战胜并取代了宿主。该模式是由 Martin Fowler 提出的，作为重写单体系统时管理风险的一种方法。有关如何应用此模式的示例，请参阅使用容器和 Amazon API Gateway 逐步将原有的 Microsoft ASP.NET（ASMX）Web 服务现代化。

子网

您的 VPC 内的一个 IP 地址范围。子网必须位于单个可用区中。

对称加密

一种加密算法，它使用相同的密钥来加密和解密数据。

综合测试

以模拟用户交互的方式测试系统，以检测潜在问题或监控性能。你可以使用 Amazon S CloudWatch ynthetics 来创建这些测试。

T

tags

充当元数据的键值对，用于组织 AWS 资源。标签可帮助您管理、识别、组织、搜索和筛选资源。有关更多信息，请参阅标记您的 AWS 资源。

目标变量

您在监督式 ML 中尝试预测的值。这也被称为结果变量。例如，在制造环境中，目标变量可能是产品缺陷。

任务列表

一种通过运行手册用于跟踪进度的工具。任务列表包含运行手册的概述和要完成的常规任务列表。对于每项常规任务，它包括预计所需时间、所有者和进度。

测试环境

参见环境。

训练

为您的 ML 模型提供学习数据。训练数据必须包含正确答案。学习算法在训练数据中查找将输入数据属性映射到目标（您希望预测的答案）的模式。然后输出捕获这些模式的 ML 模型。然后，您可以使用 ML 模型对不知道目标的新数据进行预测。

中转网关

中转网关是网络中转中心，您可用它来互连 VPC 和本地网络。有关更多信息，请参阅 AWS Transit Gateway 文档中的什么是中转网关。

基于中继的工作流程

一种方法，开发人员在功能分支中本地构建和测试功能，然后将这些更改合并到主分支中。然后，按顺序将主分支构建到开发、预生产和生产环境。

可信访问权限

为您指定的服务授予权限，让其代表您在 AWS Organizations 的组织中及其账户中执行任务。当需要服务相关的角色时，受信任的服务会在每个账户中创建一个角色，为您执行管理任务。有关更多信息，请参阅 AWS Organizations 文档中的将 AWS Organizations 与其他 AWS 服务一起使用。

优化

更改训练过程的各个方面，以提高 ML 模型的准确性。例如，您可以通过生成标签集、添加标签，并在不同的设置下多次重复这些步骤来优化模型，从而训练 ML 模型。

双披萨团队

一个小 DevOps 团队，你可以用两个披萨来喂食。双披萨团队的规模可确保在软件开发过程中充分协作。

U

不确定性

这一概念指的是不精确、不完整或未知的信息，这些信息可能会破坏预测式 ML 模型的可靠性。不确定性有两种类型：认知不确定性是由有限的、不完整的数据造成的，而偶然不确定性是由数据中固有的噪声和随机性导致的。有关更多信息，请参阅量化深度学习系统中的不确定性指南。

无差别任务

也称为繁重工作，即创建和运行应用程序所必需的工作，但不能为最终用户提供直接价值或竞争优势。无差别任务的示例包括采购、维护和容量规划。

上层环境

参见环境。

V

vacuum 操作

一种数据库维护操作，包括在增量更新后进行清理，以回收存储空间并提高性能。

版本控制

跟踪更改的过程和工具，例如存储库中源代码的更改。

VPC 对等连接

两个 VPC 之间的连接，允许您使用私有 IP 地址路由流量。有关更多信息，请参阅 Amazon VPC 文档中的什么是 VPC 对等连接。

漏洞

损害系统安全的软件缺陷或硬件缺陷。

W

热缓存

一种包含经常访问的当前相关数据的缓冲区缓存。数据库实例可以从缓冲区缓存读取，这比从主内存或磁盘读取要快。

暖数据

不常访问的数据。查询此类数据时，通常可以接受中速查询。

窗口函数

一个 SQL 函数，用于对一组以某种方式与当前记录相关的行进行计算。窗口函数对于处理任务很有用，例如计算移动平均线或根据当前行的相对位置访问行的值。

工作负载

一系列资源和代码，它们可以提供商业价值，如面向客户的应用程序或后端过程。

工作流

迁移项目中负责一组特定任务的职能小组。每个工作流都是独立的，但支持项目中的其他工作流。例如，组合工作流负责确定应用程序的优先级、波次规划和收集迁移元数据。组合工作流将这些资产交付给迁移工作流，然后迁移服务器和应用程序。

蠕虫

参见一次写入，多读。

WQF

请参阅 AWS 工作负载资格框架。

一次写入，多次读取 (WORM)

一种存储模型，它可以一次写入数据并防止数据被删除或修改。授权用户可以根据需要多次读取数据，但他们无法对其进行更改。这种数据存储基础架构被认为是不可变的。

Z

零日漏洞利用

一种利用未修补漏洞的攻击，通常是恶意软件。

零日漏洞

生产系统中不可避免的缺陷或漏洞。威胁主体可能利用这种类型的漏洞攻击系统。开发人员经常因攻击而意识到该漏洞。

僵尸应用程序

平均 CPU 和内存使用率低于 5% 的应用程序。在迁移项目中，通常会停用这些应用程序。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。