第二章 数据库基础知识

前言

- 不同的数据库产品各有特点，但是在主要的数据库概念上大家都具有一定的共同基础，都实现了各种数据库对象，实现了不同层级的安全保护措施，都强调对数据库性能管理和日常运维管理。
- 本章主要讲述数据库管理的主要职责和内容，并对一些常见的、重要的数据库概念进行了介绍，作为下一阶段学习的基础。

目标

- 学完本课程后，您将能够：
  - 描述数据库管理工作的主要内容；
    - 区分不同的备份方式
    - 列举安全管理的措施
    - 描述性能管理的工作
  - 描述数据库的重要概念，以及各数据库对象的使用方法。

目录

数据库管理 (Database Admin)

- 数据库管理
  - 数据库管理工作就是对数据库管理系统进行管理和维护的工作。
  - 核心目标，保证数据库管理系统的：
    - 稳定性
    - 安全性
    - 数据一致性
    - 系统的高性能
- 数据库管理员(Database Administrator)
  - 从事管理和维护数据库管理系统的相关人员的统称。

数据库管理工作范围 (1)

- 数据库对象管理
  - 物理设计工作；
  - 物理实现工作。
- 数据库安全管理
  - 防止未授权访问，避免受保护的信息泄露；
  - 防止安全漏洞和不当的数据修改；
  - 确保数据只提供给授权用户使用。
- 备份恢复管理
  - 制定合理的备份策略，实现数据定期备份功能；
  - 保证灾难发生时数据库系统能够做到最快恢复和最小损失。

数据库管理工作范围 (2)

- 数据库性能管理
  - 对影响数据库性能的因素进行监控和优化。
  - 对数据库能使用的资源进行优化，从而增加系统吞吐量，并减少竞争，最大可能地处理工作负载。
- 数据库环境管理
  - 数据库的运行和维护管理，包括安装，配置，升级，迁移等；
  - 确保数据库系统在内的IT基础设施的正常运作。

目录

数据库对象

- 什么是数据库对象?
  - 数据库里用来存储和指向数据的各种概念和结构的总称。
  - 对象管理就是使用对象定义语言或者工具创建，修改或删除各种数据库对象的管理过程。
  - 常见的基本数据库对象：

对象	名称	作用
TABLE	表	用于存储数据的基本结构。
VIEW	视图	以不同的侧面反映表的数据，是一种逻辑上的“虚拟表”，视图本身不存储数据。
INDEX	索引	索引提供指向存储在表的指定列中的数据值的指针，如同图书的目录，能够加快表的查询速度。
SEQUENCE	序列	用来产生唯一整数的数据库对象。
STORE PROCEDURE、FUNCTION	存储过程、函数	一组为了完成特定功能的SQL 语句集。存储过程、函数经过编译后，可以被重复调用，从而可以减少数据库开发人员的工作量。

制定数据库对象命名规范

- 良好的设计是良好的开端
  - 数据库产品本身没有严格的限制。
  - 随意的对象命名会导致系统的不可控，不可维护。
- 命名规范的几点建议
  - 统一名称的大小写；
  - 利用前缀标识对象类型，如表名前缀t_，视图前缀v_，函数前缀f_等；
  - 命名尽量采用富有意义、易于记忆、描述性强、简短及具有唯一性的英文词汇，不建议使用汉语拼音；
  - 以项目为单位，采用名称词典，制定一些公共的缩略词，如amt代表amount（数量）。

不建议命名	建议命名	说明
Table_customer	t_customer	Table是数据库保留关键词，不建议使用
t_001	t_customer_orders	命名只有数字和无意义的字母，整体名称无法体现对象的含义
v@orders	v_orders	还有非法字符
shitu_dizhi	v_address	避免使用汉语拼音
special_customer_account_total_amount	acct_amt	名称过长，适当使用缩略词
T_Customer_Orders, v_customer_orders	t_cust_orders, v_cust_orders,	大小写规则要统一

目录

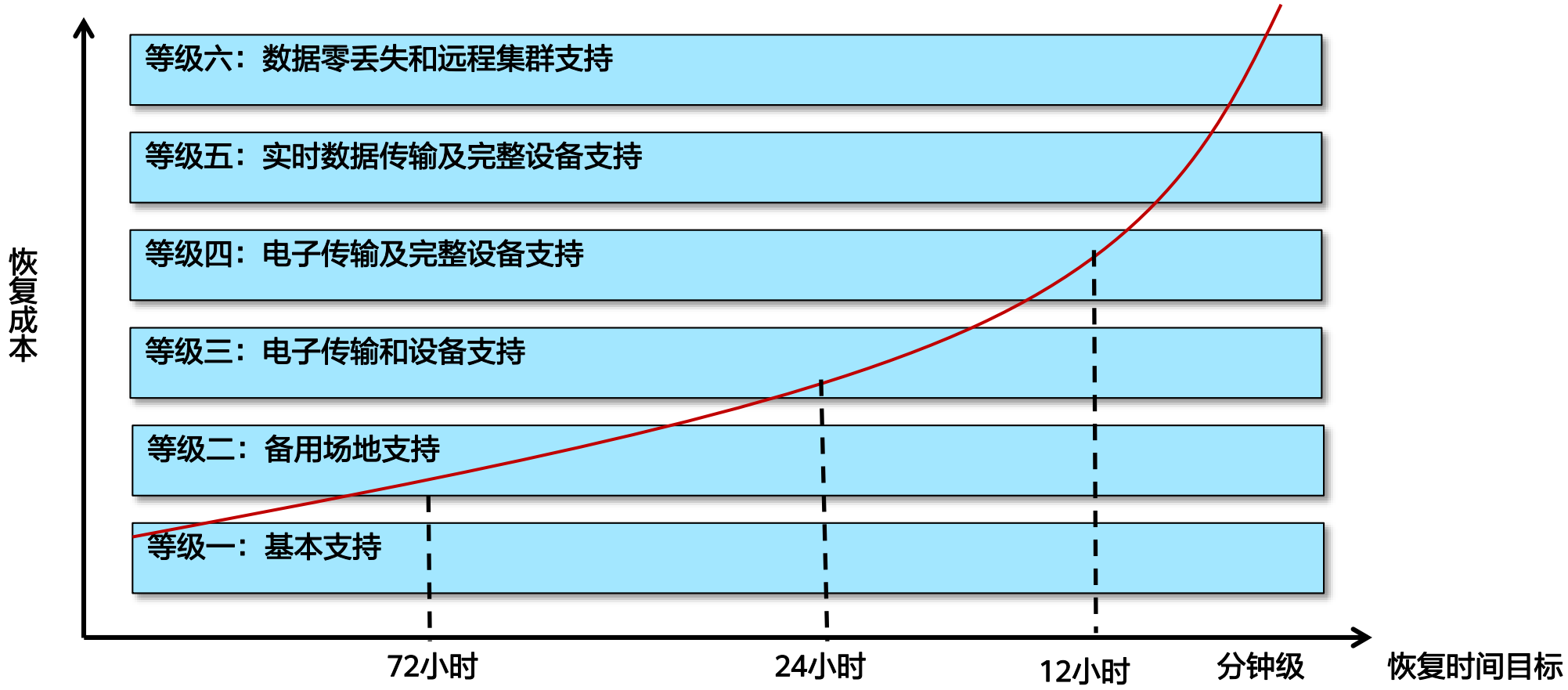
备份和恢复的基本概念

- 数据库备份
  - 备份数据库就是将数据库中的数据，以及保证数据库系统正常运行的有关信息保存起来，以备系统出现故障后恢复数据库时使用。
- 备份对象，包括但不限于：
  - 数据本身；
  - 和数据相关的数据库对象；
  - 用户及权限；
  - 数据库环境，如配置文件，定时任务等。
- 数据库恢复
  - 将数据库系统从故障或者瘫痪状态恢复到可正常运行，并能够将数据恢复到可接受状态的活动。

灾难恢复

- 企业级容灾
  - 对于企业和单位来说，数据库系统和其他应用系统构成更大的信息系统平台，所以数据库备份恢复并不是孤立的功能点，要和其它应用系统一并考虑整个信息系统平台的容灾性能。
- 灾难备份
  - 为了灾难恢复而对数据、数据处理系统、网络系统、基础设施、专业技术能力和运行管理能力进行备份的过程。
  - 恢复时间目标(RTO)
    - 灾难发生后，信息系统或业务功能从停顿到必须恢复的时间要求。
  - 恢复点目标(RPO)
    - 灾难发生后，系统和数据必须恢复到的时间点要求。

灾难恢复等级



灾难恢复等级划分参考国家标准规范：《 GB/T 20988-2007 ： 信息系统灾难恢复规范 》

某行业RTO/RPO与灾难恢复能力等级关系

灾难恢复能力等级	RTO	RPO
1	2天以上	1天至7天
2	24小时以上	1天至7天
3	12小时以上	数小时至1天
4	数小时至2天	数小时至1天
5	数分钟至2天	0至30分钟
6	数分钟	0

备份方式

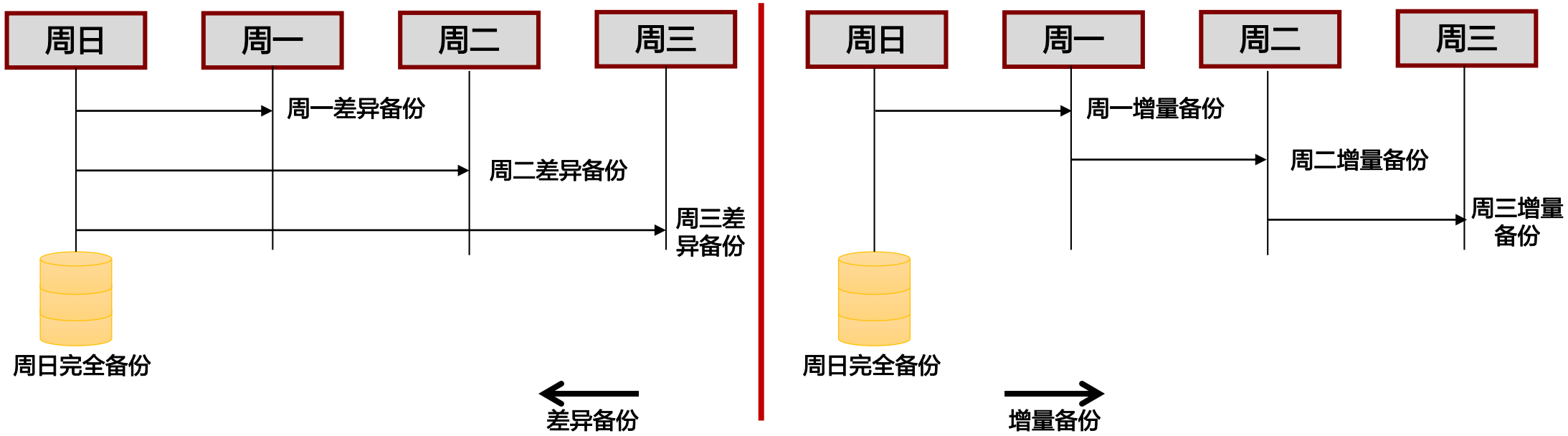
- 根据备份的数据集合的范围：
  - 全量备份
  - 差异备份
  - 增量备份
- 根据是否停用数据库：
  - 热备
  - 温备
  - 冷备
- 根据备份内容：
  - 物理备份
  - 逻辑备份

全量备份

- 全量备份
  - 也称为完全备份。
  - 对某个指定时间点的所有数据和对应的结构进行一个完全的备份。
- 特点
  - 数据最完备；
  - 安全性最高；
  - 备份和恢复时间随着数据的体量而明显增加；
  - 非常重要，是差异备份和增量备份的基础；
  - 备份期间会对系统性能产生一定影响。

差异备份和增量备份

- 差异备份
  - 差异备份是指上一次全量备份之后，对发生变化的数据进行的备份。
- 增量备份
  - 增量备份是指上一次备份之后，对发生变化的数据进行的备份。



热备，温备和冷备

- 热备
  - 在数据库正常运行下进行备份。
  - 备份期间，数据库读写均可正常进行。
- 温备
  - 数据库可用性弱于热备，备份期间，数据库只能进行读操作，不能进行写操作。
- 冷备
  - 在备份期间，应用的读写操作不可进行。
  - 备份出的数据可靠性最高。

物理备份和逻辑备份

- 物理备份
  - 直接备份数据库所对应的数据文件甚至是整个磁盘。
- 逻辑备份
  - 将数据从数据库中导出，并将导出的数据进行存档备份。

类别	物理备份	逻辑备份
备份对象	数据库的物理文件（如数据文件，控制文件，归档日志文件等）	数据库对象（如用户，表，存储过程等）
可移植性	较弱，甚至不可移植	数据库对象级备份，可移植性较强
占用空间	占用空间大	占用空间相对较小
恢复效率	效率高	效率较低
适用场景	大型业务系统或者整系统的容灾恢复、系统级全量备份	主备数据库间的增量数据备份、不同业务系统之间的数据同步、业务不中断升级过程中在线数据迁移

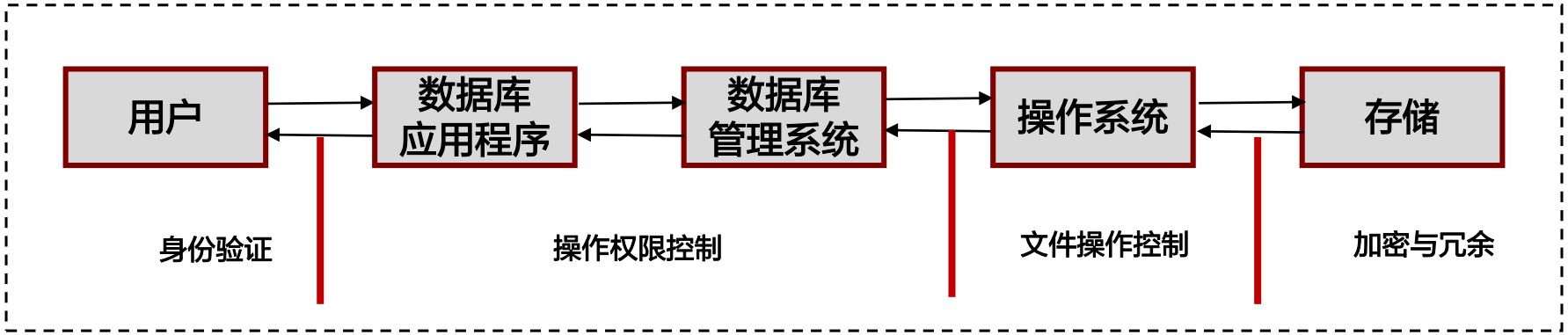
目录

数据库系统安全框架

- 广义范围，数据库安全框架可以分为三个层次：
  - 网络层次安全
    - 从技术角度讲，网络系统层次安全方法技术主要有加密技术，数字签名技术，防火墙技术和入侵检测技术等。
  - 操作系统层次安全
    - 核心是要保证服务器的安全，主要体现在服务器的用户账户，口令，访问权限等方面。
    - 数据安全主要体现在加密技术、数据存储的安全性，数据传输的安全性等方面，如Kerberos，IPsec，SSL和VPN等技术。
  - 数据管理系统层次安全
    - 数据库加密；
    - 数据存取访问控制；
    - 安全审计；
    - 数据备份。

安全控制模型

- 安全控制
  - 在数据库应用系统的不同层次提供对有意和无意损害行为的安全防范，例如：
    - 加密存取数据 -> 有意非法活动
    - 用户身份验证，限制操作权限 -> 有意的非法操作
    - 提高系统可靠性和数据备份 -> 无意的损害行为
- 安全控制模型



身份验证

- 数据库用户的身份验证是DBMS提供的最外层安全保护措施。
  - 阻止未经授权的用户的访问。
  - 对于数据库应用目前普遍采用用户名密码验证模式，所以有必要增强密码强度。
    - 采用长度较长的字符串，如8-20个字符；
    - 混合数字，字母和符号的密码；
    - 定期更换密码；
    - 密码不能重复使用。
  - 在开发的代码或者脚本中，禁止出现数据库用户的密码明文。

访问控制

- 访问控制是数据库安全中最有效的办法也是最容易出问题的地方。
- 基本原则
  - 对于不同用户根据敏感数据的分类要求，给予不同的权限。
    - 最小权限原则
    - 检查关键权限
    - 检查关键数据库对象的权限
- 基于角色的权限管理
  - 对于大型数据库系统或者用户数量多的系统，权限管理主要使用基于角色的访问控制(Role Based Access Control，RBAC)。

开启审计

- 审计可以帮助数据库管理员发现现存架构和使用中的漏洞。
- 数据库审计的层次
  - 访问及身份验证审计，数据库用户登入(logon)，登出(logoff)的相关信息，如登入登出时间，连接方式及参数信息，登入途径等。
  - 用户与管理员审计：针对用户和管理员执行的活动进行分析和报告。
  - 安全活动监控：记录数据库中任何未授权或者可疑的活动生成审计报告。
  - 漏洞与威胁审计：发现数据库可能存在的漏洞，以及想要利用这些漏洞的“用户”。

数据库加密

- 数据库加密的不同层次:
  - DBMS内核层
    - 数据在物理存取之前完成加/解密工作;
    - 对于数据库用户来说是透明的，没有感觉的;
    - 采用加密存储，加密运算在服务器端运行，在一定程度上会加重服务器的负载。
  - DBMS外层加密
    - 开发专门的加解密工具，或者定义加解密方法;
    - 可以控制加密对象粒度，到表或者字段级别进行加解密;
    - 用户只需要关注敏感信息范围。

目录

资源

- 供给类资源
  - 这类资源也称为基础资源，是计算机硬件对应的资源。
  - 操作系统管理的资源。
  - 处理能力：CPU>内存>>磁盘≈网络。
- 并发性控制资源
  - 这类资源包括但不限于：锁，队列，缓存，互斥信号等。
  - 数据库系统管理的资源。
- 性能管理的基本原则
  - 充分利用资源不浪费。

指标	说明	时间(ns)
L1 cache reference	读取CPU的一级缓存	0.5
L2 cache reference	读取CPU的二级缓存	7
Main memory reference	读取内存数据	100
Compress 1K bytes with Zippy	Zippy算法压缩1K字节	10,000
Send 2K bytes over 1 Gbps network	在千兆网网发送2K字节	20,000
Read 1 MB sequentially from memory	从内存顺序读取1MB	250,000
Disk seek	磁盘搜索	10,000,000(10 ms)
Read 1 MB sequentially from network	从网络上顺序读取1兆的数据	10,000,000
Read 1 MB sequentially from disk	从磁盘顺序读出1MB	30,000,000

性能管理的意义

- 资源的高效使用
  - 数据库实际上总是在有限的环境下运行。
  - 对资源的有效管理确保数据库系统在高峰时期能够满足用户对系统的性能要求。
- 侦测系统问题
  - 实时的系统性能监控（通过数据库提供的日志或者工具进行实时监控系统性能）。
  - 系统历史性能数据跟踪（历史性能数据的分析）。
- 容量规划
  - 性能管理所收集到的数据是进行系统容量规划及其他前瞻性规划的基础。
  - 用事实而不是感觉说话。

性能管理的目标

- 数据库系统的基本指标
  - 吞吐量；
  - 响应时间。
- OLTP
  - 在可接受的响应时间基础之上提供尽可能高的吞吐量。
  - 降低单位资源消耗，快速通过并发共享区域，减少瓶颈制约。
- OLAP
  - 在有限的资源内尽可能地缩短响应时间。
  - 一个事务应该充分利用资源来加速处理时间。

性能优化工作的一些场景

- 上线优化或未达到性能期望的性能优化；
- 响应速度逐渐变慢的系统优化；
- 系统运行过程中突然变慢的系统优化（应急处理）；
- 突然变慢，持续一段时间以后又恢复正常；
- 基于降低资源消耗的系统优化；
- 预防性的日常巡检工作。

性能管理需要采集的数据

- 性能管理需要采集的数据范围，包括但不限于：
  - CPU使用数据；
  - 空间使用率；
  - 使用数据库系统的用户和角色；
  - 心跳查询的响应时间；
  - 提交到数据库的SQL为基本单元的性能数据；
  - 数据库工具提交的作业相关的性能数据（如加载，卸载，备份，恢复等）。
- 关注的时间范围：
  - 日常范围：一周高峰时段的时间；月度结束的时间；季节变化数据。
  - 一天范围内：用户集中使用系统的时间段；系统压力比较高的时间段等。

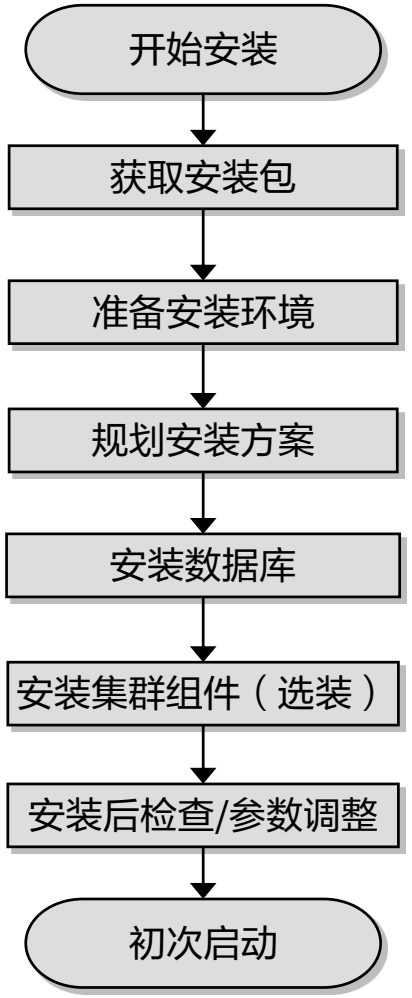
建立性能报表

- 数据库系统内置很多监控报表
  - 提取性能相关的数据建立定期性能报表（日报，周报，月报）。
  - 建立常见指标的性能趋势分析报表，可以对当前系统性能有直观的展现。
  - 特定趋势类型的报表，包括但不限于
    - 基于异常事件的报表；
    - 消耗大量资源的SQL或是作业；
    - 特定用户、用户群的资源消耗报表；
    - 特定应用的资源消耗报表。

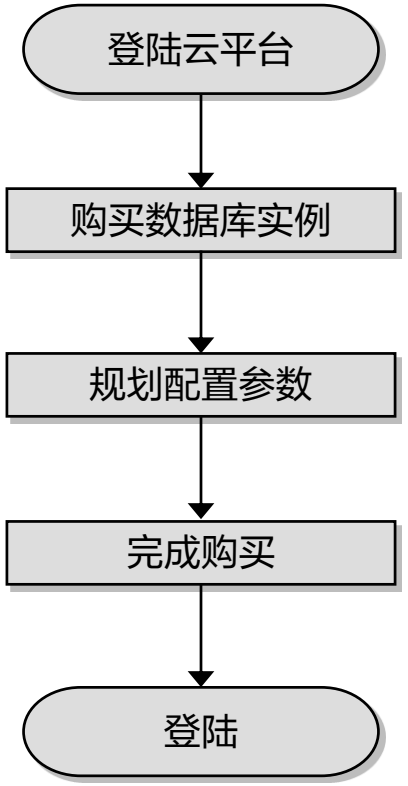
目录

数据库安装

- 知识准备
  - 关系数据库理论
  - 操作系统知识
  - 了解数据库产品的特点
    - 软件架构
    - 网络架构
    - 服务器架构
  - 了解目标数据库的专有名词和特定术语
  - 阅读安装手册，尤其是安装注意事项



传统数据库安装



云数据库安装

数据库卸载

- 在实际场景中，多发生于数据库的版本升级之前，需要对老版本的数据库进行卸载或者清理。
- 基本步骤
  - 传统数据库
    - （可选）对数据库进行一次全备。
    - 停止数据库服务。
    - 卸载数据库。
  - 云数据库
    - （可选）对数据库进行一次全备。
    - 云平台删除数据库实例。
- 不同架构场景下
  - 单机，主备或一主多备的卸载方式都是类似的，需要在每个节点上执行相同的卸载操作。
  - 分布式集群的卸载一般使用专有的卸载工具。
- 卸载后
  - 对于一些客户，数据库卸载后需要对存储介质上的数据再进行销毁处理，保证数据不外泄。

数据库迁移

- 数据库迁移
  - 需要依据不同的迁移场景需求设计迁移方案。
  - 考虑的要素
    - 迁移可用的时间窗口；
    - 迁移可以使用的工具；
    - 迁移过程中数据源系统是否停止写入操作；
    - 迁移过程的数据源系统和目标系统之间的网络情况如何；
    - 根据迁移的数据量估算备份/恢复时间；
    - 迁移后，源和目标数据库系统之间的数据一致性稽核。

数据库扩容

- 任何一个数据库系统的容量都是在某个时间点的基础上对未来一段时间内的数据量进行估算后确定的，容量不仅仅是数据存储量，还需要考虑以下几个方面：
  - 计算能力不足（整个系统CPU日均繁忙程度>90%）；
  - 响应/并发能力不足（QPS，TPS显著下降，无法满足SLA）；
  - 数据容量不足（可用的数据空间低于15%）。
- 扩容方案的选择
  - 垂直扩容
    - 垂直扩容是增加数据库服务器硬件，如增加内存，增大存储，提升网络带宽，提升单机硬件方面性能配置。这种方式相对简单，但是会遭遇单机硬件性能瓶颈。
  - 水平扩容
    - 横向增加服务器数量，利用集群中服务器数量的优势来增加整体系统的性能。
  - 停机扩容
    - 简单，但是时间窗口有限，出现问题会导致扩容失败。而且如果时间过长，不易被客户接受。
  - 平滑扩容
    - 对数据库服务无影响；
    - 技术方案相对复杂，尤其数据库服务器数量增多，扩容复杂程度就急剧上升。

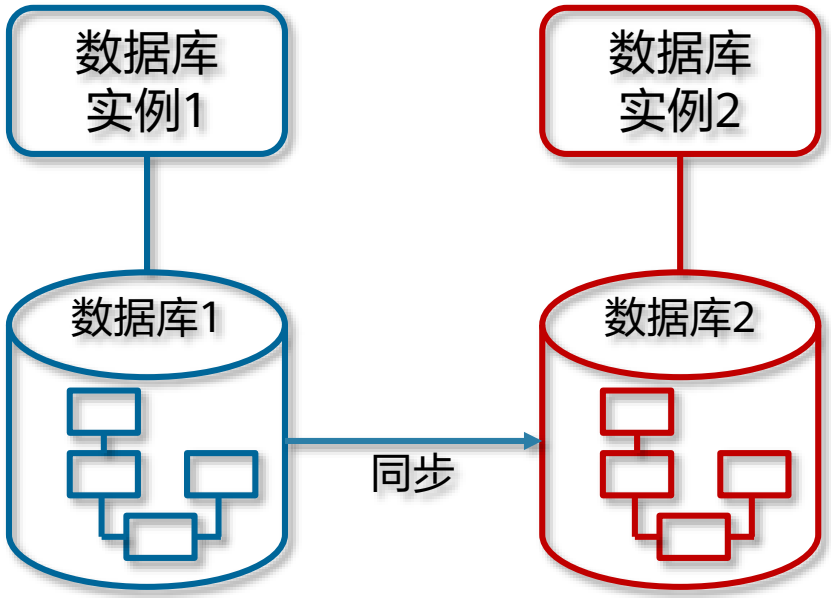
例行维护工作

- 数据库故障处理
  - 配置数据库监控指标和告警阈值；
  - 针对故障事件的等级设置告警通知流程；
  - 接受告警信息后，根据日志进行故障定位；
  - 对于遇到的问题，应详细记录原始信息；
  - 严格遵守操作规程和行业安全规程；
  - 对于重大操作，在操作前要确认操作可行性，做好相应的备份、应急和安全措施后，由有权限的操作人员执行。
- 数据库健康巡检
  - 查看健康检查任务；
  - 管理健康检查报告；
  - 修改健康检查配置。

目录

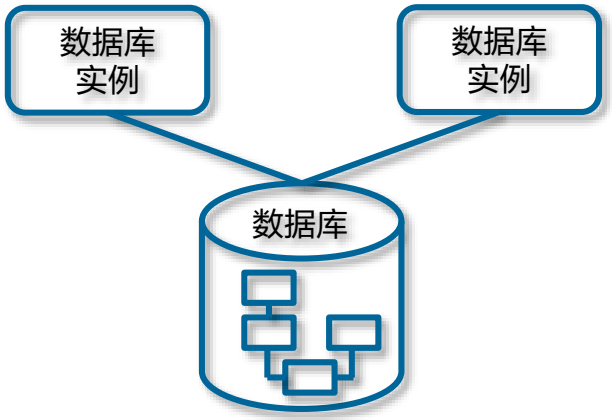
数据库和数据库实例 (1)

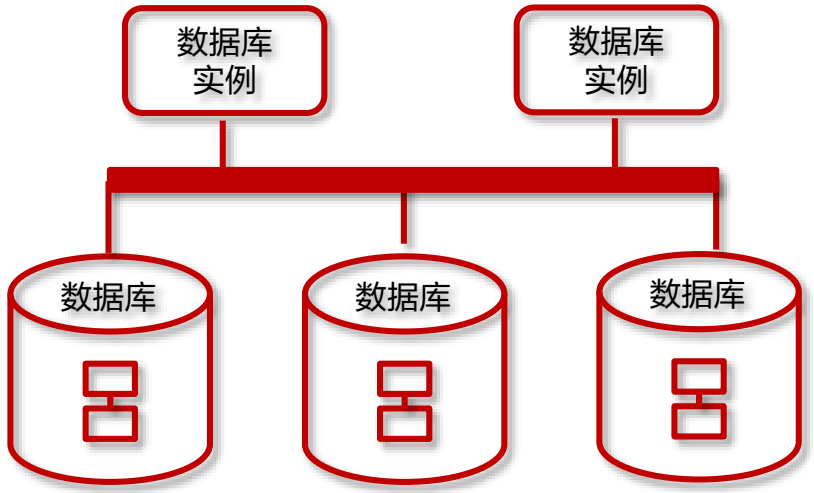
- 数据库(Database)
  - 物理操作系统文件或磁盘数据块的集合
    - 比如数据文件，索引文件，结构文件。
    - 并非所有的数据库系统都是基于文件的，也有直接把数据写入数据存储的形式。
- 数据库实例(Database Instance)
  - 实例指的就是操作系统中一系列的进程以及为这些进程所分配的内存块。
- 数据库实例是访问数据库的通道。
- 通常来说一个数据库实例对应一个数据库。



数据库和数据库实例 (2)

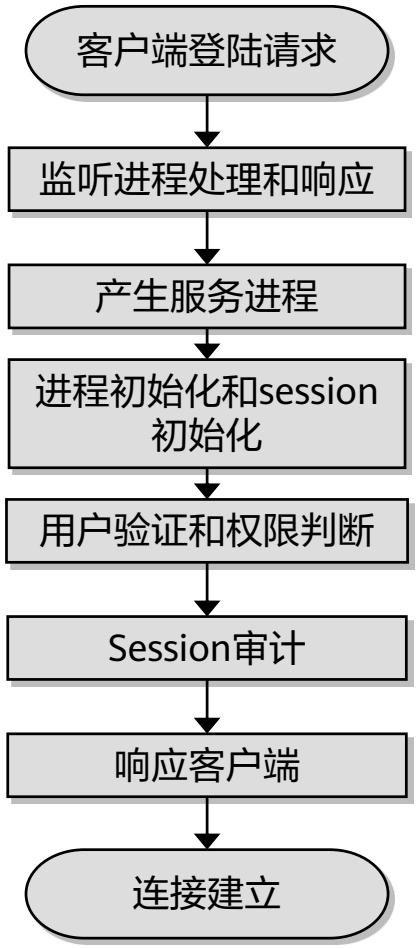
- 多实例
  - 利用多实例操作，可以更充分的利用硬件资源，让服务器性能最大化。
- 分布式集群
  - 集群就是一组相互独立的服务器，通过高速的网络组成一个计算机系统。
  - 分布式集群中，每个服务器都可能有数据库的一份完整副本，或者部分副本，所有服务器通过网络互相连接，共同组成一个完整的、全局的，逻辑上集中、物理上分布的大型数据库。





数据库连接和会话

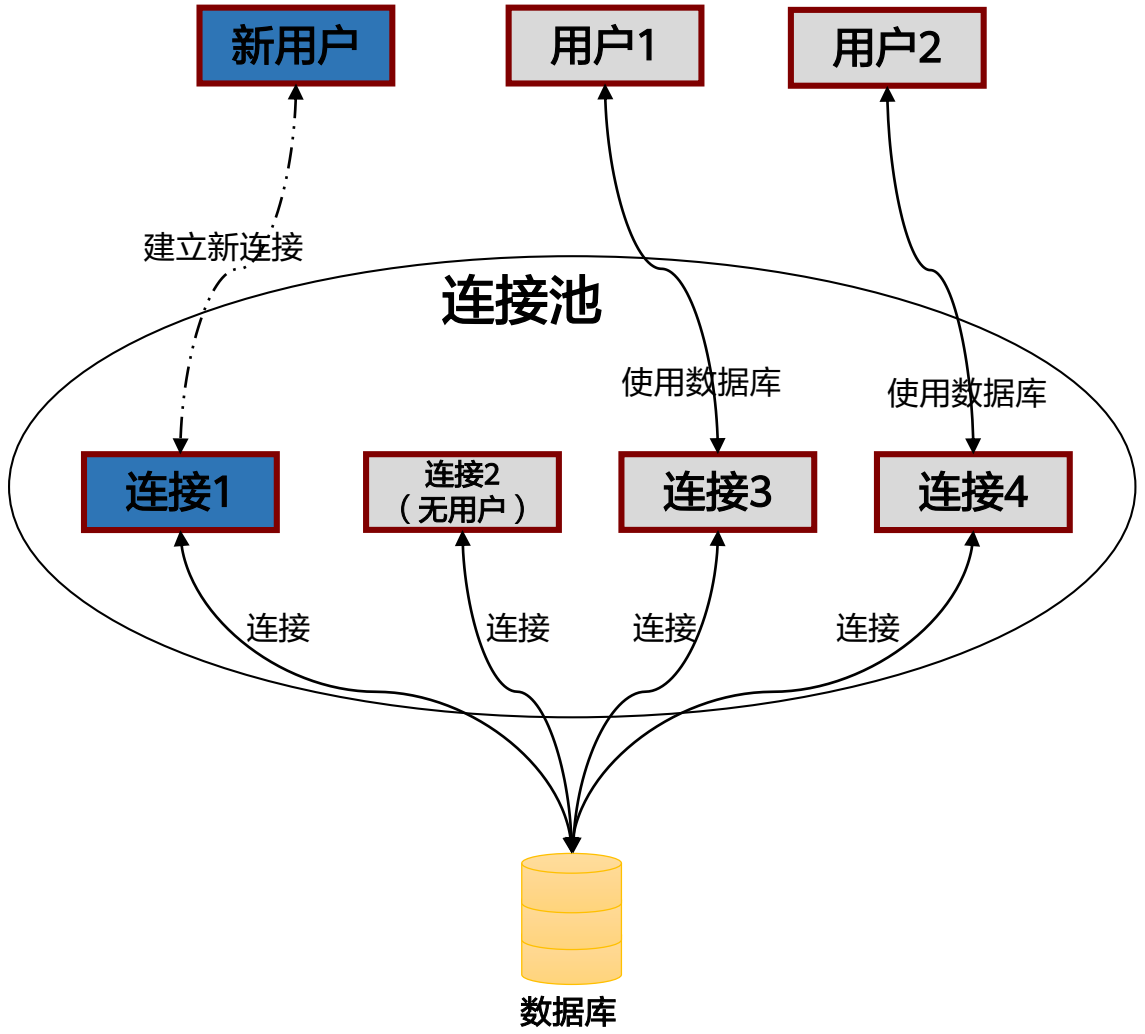
- 数据库连接(Connection)
  - 物理层面的通信连接，指的是一个通过网络建立的客户端和专有服务器(Dedicated Server)或调度器(Shared Server)的一个网络连接。
  - 建立连接时候指定连接参数，如服务器主机名或ip，端口号，连接用户名和口令等。
- 数据库会话(Session)
  - 客户端和数据库之间通信的逻辑概念。
  - 通信双方从开始通信到通信结束期间的一个上下文(Context)。这个上下文是一段位于服务器端的内存：记录了本次连接的客户端机器、对应的应用程序进程号、对应的用户登录等信息。



数据库连接建立流程示意图

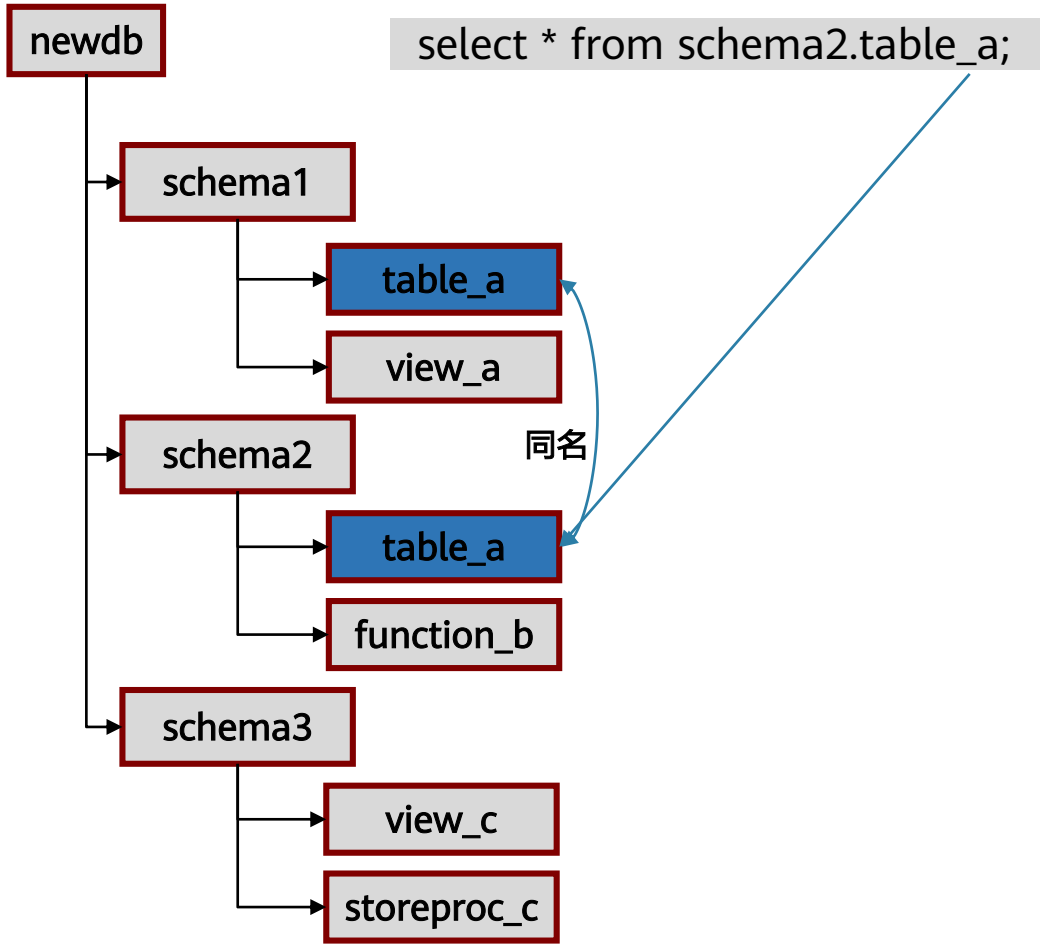
数据库连接池

- 建立数据库连接是有代价的
  - 频繁的建立和关闭数据库连接，会使得对连接资源的分配和释放成为数据库的瓶颈，从而降低数据库系统的性能。
- 连接池：数据库连接的复用
  - 负责分配、管理和释放数据库连接，它允许应用程序重复使用一个现有的数据库连接，而不是再重新建立一个。
  - 数据库连接可以得到高效、安全的复用。



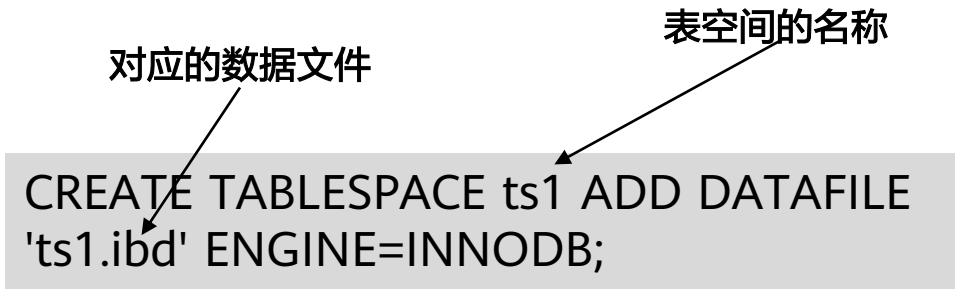
Schema

- Schema是数据库形式语言描述的一种结构，是对象的集合。
  - 允许多个用户使用一个数据库而不干扰其他用户。
  - 把数据库对象组织成逻辑组，让他们更便于管理。
  - 形成命名空间，避免对象的名字冲突。
  - schema包含表及其他数据库对象，数据类型、函数、操作符等。



表空间 (Tablespace)

- 表空间是由一个或者多个数据文件组成的。
  - 通过表空间定义数据库对象文件的存放位置。
  - 数据库中所有对象在逻辑上都存放在表空间中。
  - 在物理上储存在表空间所属的数据文件中。
- 表空间作用
  - 根据数据库对象使用模式安排数据物理存放位置，提高性能。
    - 频繁使用的索引放置在性能稳定且运算速度快的磁盘上。
    - 归档数据，使用频率低，对访问性能要求低的表存放在速度慢的磁盘上。
  - 通过表空间指定数据占用的物理磁盘空间。
    - 通过表空间限制物理空间使用上限，避免磁盘空间被耗尽。



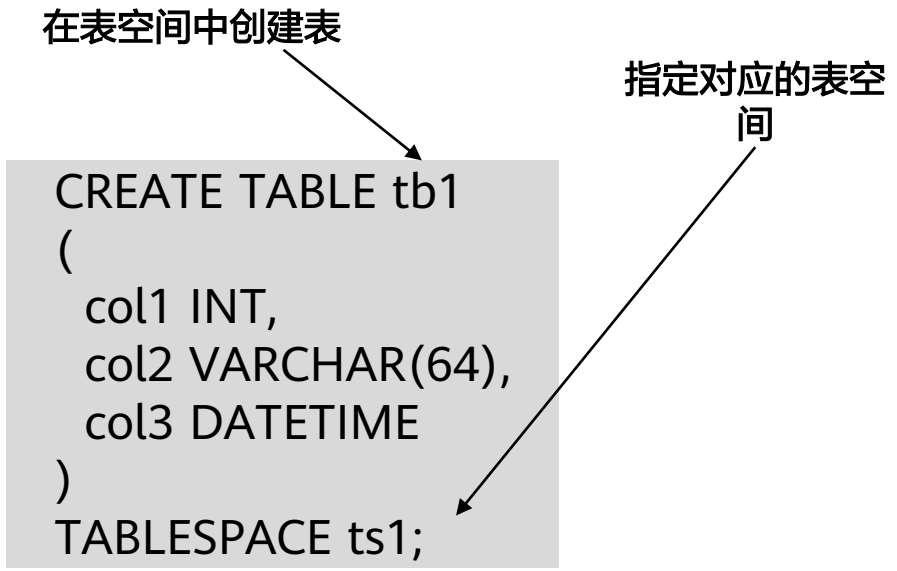


表 (Table)

- 在关系数据库中，数据库表就是一系列二维数组的集合
  - 用来代表和储存数据对象之间的关系。
- 记录
  - 表中的每一行称为一个记录，由若干个字段组成。
- 字段
  - 也称为域，表中的每一列称为一个字段。
  - 每个字段都包含两个属性：列名和数据类型。

	字段1	字段2	字段3	字段4
	作者id	作者姓名	作者年龄	作者联系地址
记录1	001	张三	40	XXX省YYY市
记录2	002	李四	50	AAA省BBB市
…	…	…	…	…

```
CREATE TABLE author_t1
(
author_id integer,
author_name char(60),
author_age integer,
authro_address varchar(255)
);
```

字段名称

字段数据类型

字段数据长度

临时表

- GaussDB(for MySQL)支持创建临时表
  - 临时表用来保存一个会话中需要的数据。当会话退出的时候，临时表的数据自动清空。
  - 临时表中的数据是临时的，过程性的，不需要和普通数据表那样永久保留的。
  - 使用 SHOW TABLES命令无法显示临时表。

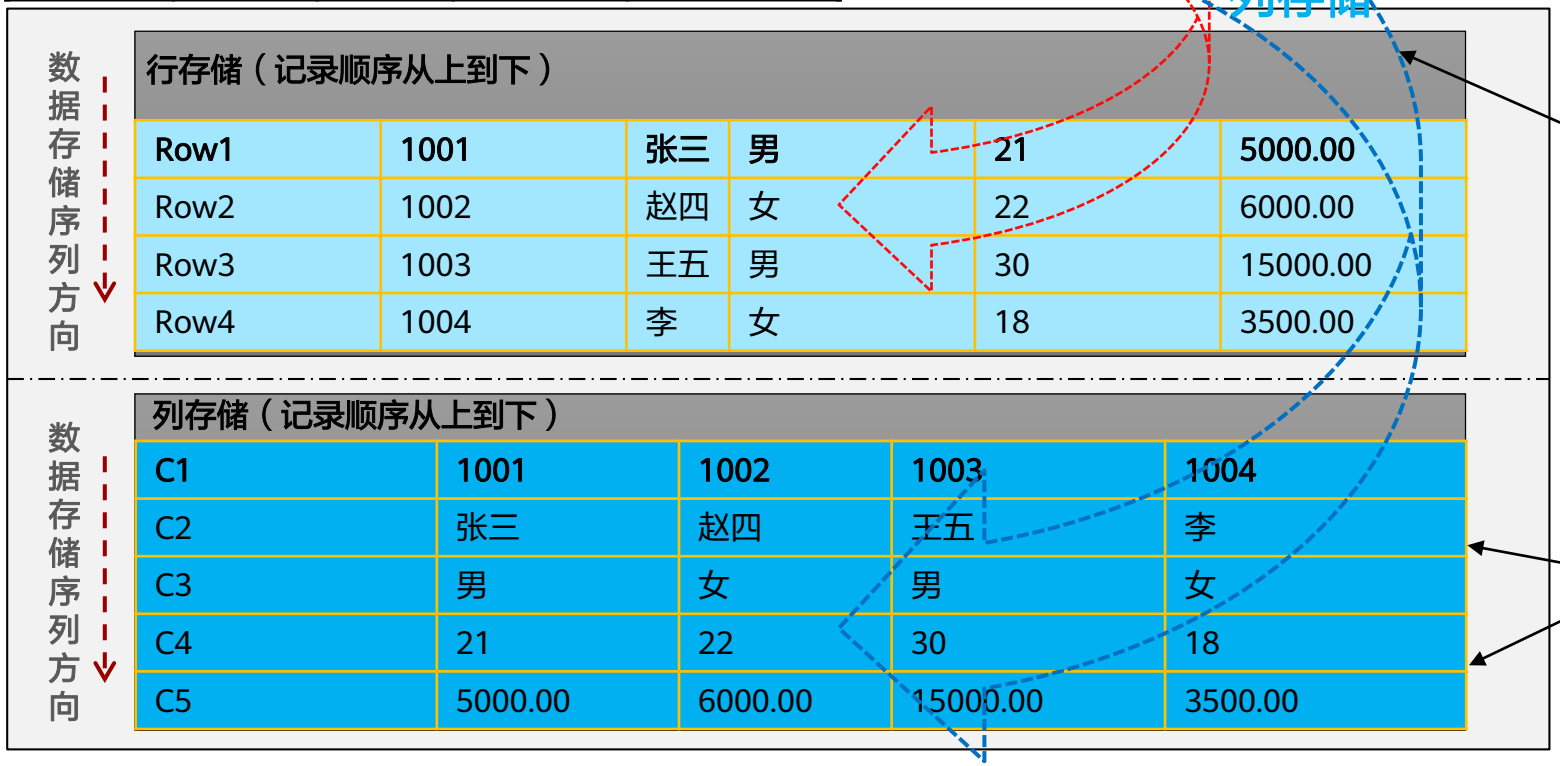
```
CREATE TEMPORARY TABLE staff_history_session
(
startdate DATE,
enddate DATE
);
```

临时表关键词

表的存储方式

工号	姓名	性别	年龄	薪水
1001	张三	男	21	5000.00
1002	赵四	女	22	6000.00
1003	王五	男	30	15000.00
1004	李	女	18	3500.00

- 按照数据的存储方式，表分为两种：
  - 行存储表
  - 列存储表



写入过程要把记录拆开，不同的列数据分别写入不同的存储区域，多次写入过程会导致IO次数增加，效率相对较慢。

select name,age from employee;
对于列存储表的查询，只要扫描少量所需要列对应的存储即可，IO开销比较小。

存储方式的选择

- 列存适合的场景
  - 统计分析类查询（group，join多的场景）；
  - 适合OLAP，数据挖掘等大量查询的应用查询。
- 行存适合的场景
  - 点查询（返回记录少，基于索引的简单查询）；
  - 适合OLTP，这种轻量级事务，大量写操作，数据增删改比较多的场景。

分区 (Partition)

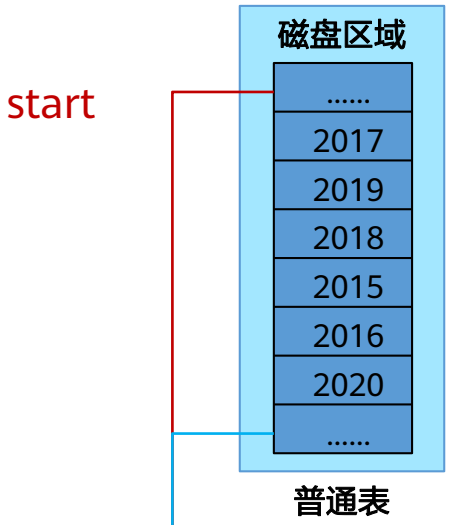
- 分区表是将大表的数据分成许多小的数据子集，称为分区。
  - 范围分区表
  - 列表分区表
  - 哈希分区表
  - 间隔分区表
- 分区表的收益
  - 改善查询性能
  - 增强可用性
  - 方便维护
  - 均衡I/O

```
CREATE TABLE tp
(
id INT,
name VARCHAR(50),
purchased DATE
)
    PARTITION BY RANGE( YEAR(purchased) )
(
    PARTITION p0 VALUES LESS THAN (2015),
    PARTITION p1 VALUES LESS THAN (2016),
    PARTITION p2 VALUES LESS THAN (2017),
    PARTITION p3 VALUES LESS THAN (2018),
    PARTITION p4 VALUES LESS THAN (2019),
    PARTITION p5 VALUES LESS THAN (2020)
    );
```

对日期进行范围分区

分区剪枝的原理

- 分区剪枝
  - 对分区对象的查询可以仅搜索自己关心的分区，提高检索效率。



假设表中所有10年数据，按年份分区的话，分区表执行对应sql， 经过分区剪枝后只需要扫描其中一个分区，扫描数据量为1/10。

计算2019年的所有订单的销售总额

```
SELECT name FROM tp
WHERE purchased BETWEEN date_format('2019-01-01', '%Y%m%d') AND
date_format('2019-12-31', '%Y%m%d');

SELECT * FROM tr PARTITION (p4);
```

分区适用场景

场景描述	收益
当表中访问率较高的行位于一个单独分区或少数几个分区时	大幅减少搜索空间，从而提升访问性能。
向空分区插入数据	空分区插入数据效率提高。
当需要大量加载或者删除的记录位于一个单独分区或少数几个分区时	可直接读取或删除对应分区，从而提升处理性能；同时由于避免大量零散的删除操作，可减少清理碎片工作量。

数据分布

- GaussDB(DWS)分布式数据库的数据表是分散在所有数据节点(DataNode, DN)上的，所以创建表的时候需要指定分布列。

分布方式	说明
Hash	表数据通过Hash方式散列到集群中的所有DN。
Replication	集群中每一个DN都有一份全量表数据。
List	表数据通过List方式分布到指定DN节点上。
Range	表数据通过Range方式分布到指定DN节点上。

```
CREATE TABLE sales_fact
(
region_id   INTEGER,
depart_id   INTEGER,
product_id INTEGER,
sale_amt    NUMERIC(9,2),
sale_qty    INTEGER
)
DISTRIBUTE BY HASH(region_id, depart_id,
product_id);
```

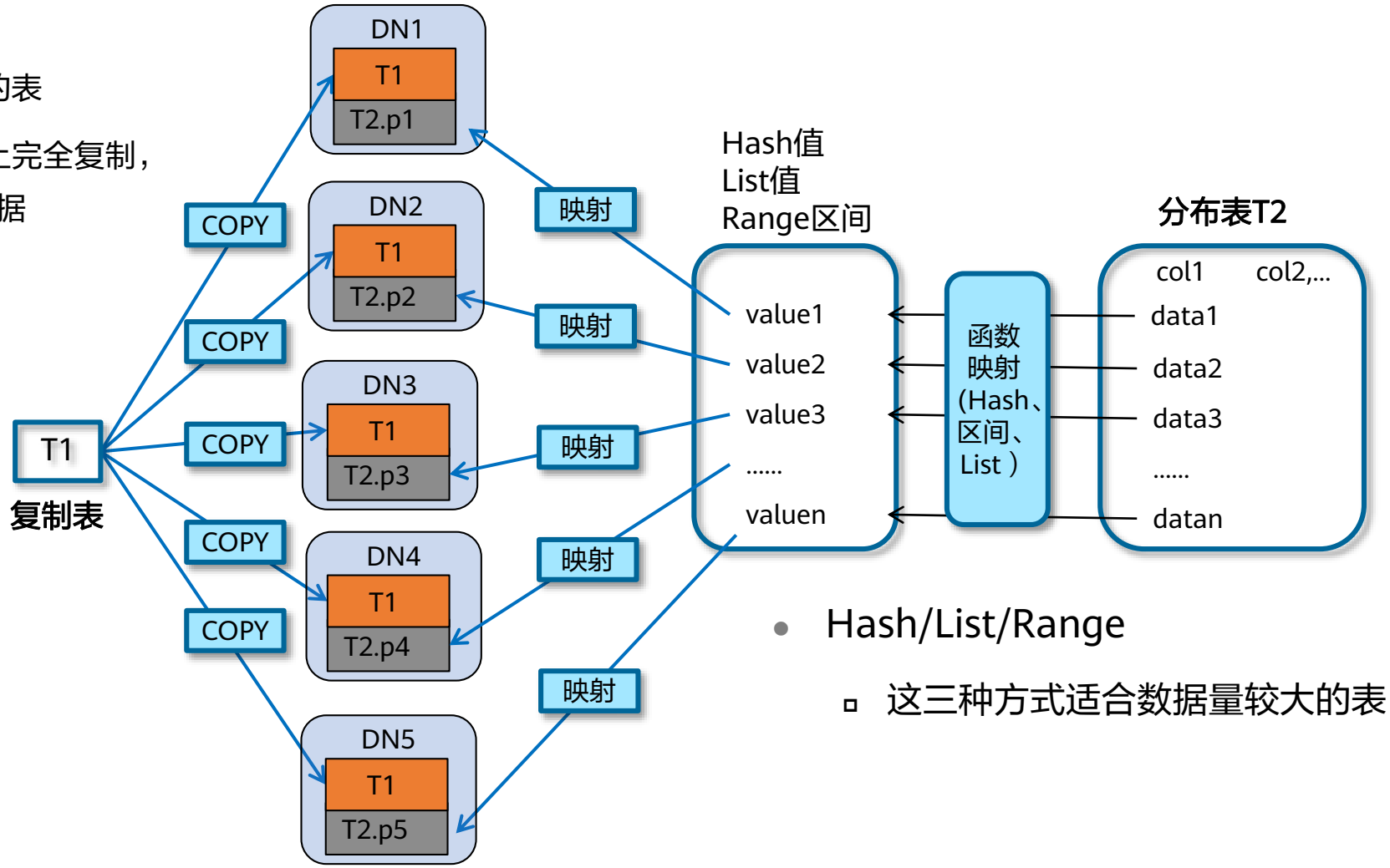
Hash分布方式

```
CREATE TABLE depart_dim
(
depart_id   INTEGER,
depart_name VARHCARH2(60)
)
DISTRIBUTE BY REPLICATION;
```

Replication复制分布方式

数据策略选择

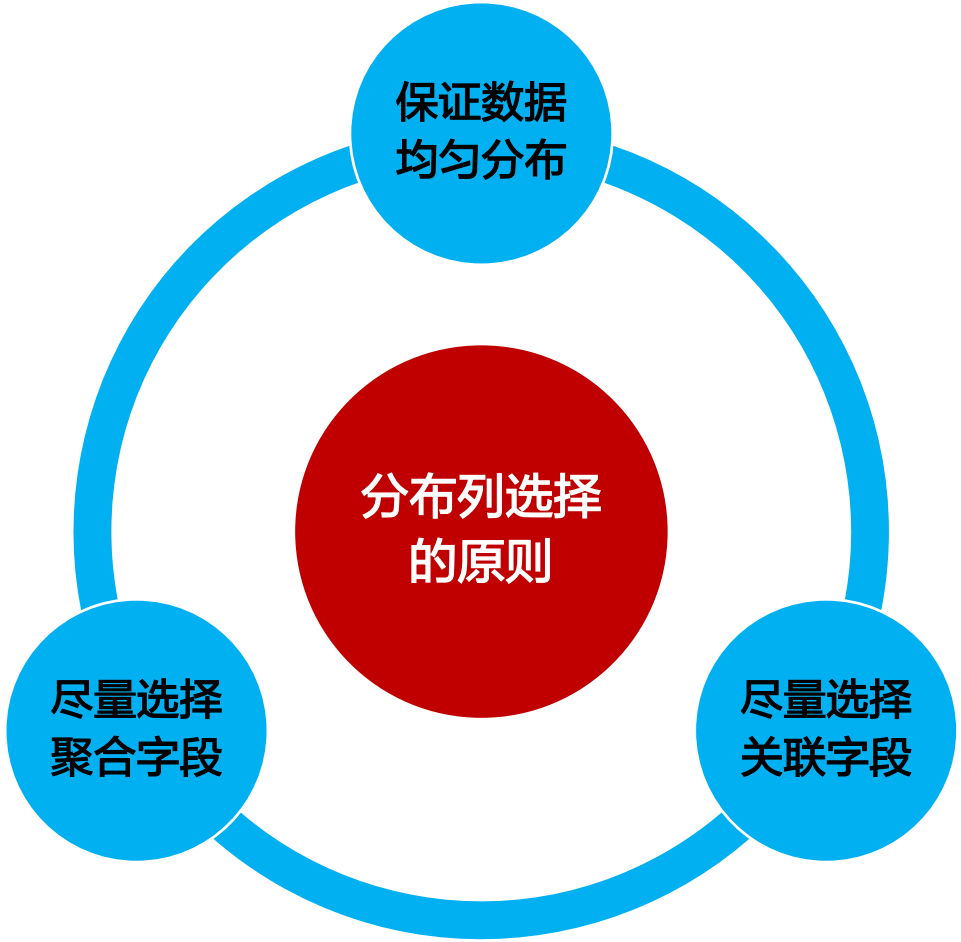
- 复制(Replication)
  - 适合于记录集较小的表
  - 表中数据在各节点上完全复制，各DN都拥有全量数据



- Hash/List/Range
  - 这三种方式适合数据量较大的表

分布列选择原则

- 选择分布列的时候，一般遵循下面三个原则：
  - 尽量选择离散值比较多的列，保证数据均匀分布。分布均匀是为了避免木桶效应，各个DN对等执行。
  - 在满足第一条原则的情况下，不要选择存在常量过滤的列。
  - 满足前两条原则的情况下，尽量选择关联字段或聚合字段做分布列，这种方式是为了避免数据节点之间数据重分布，降低IO的开销从而提升关联操作和聚合操作的性能。



数据类型

- 基本数据类型
  - 数值类型
  - 字符类型
  - 二进制类型
  - 日期/时间类型
  - 布尔类型
  - 枚举类型等
- 序列号类型
- 几何类型

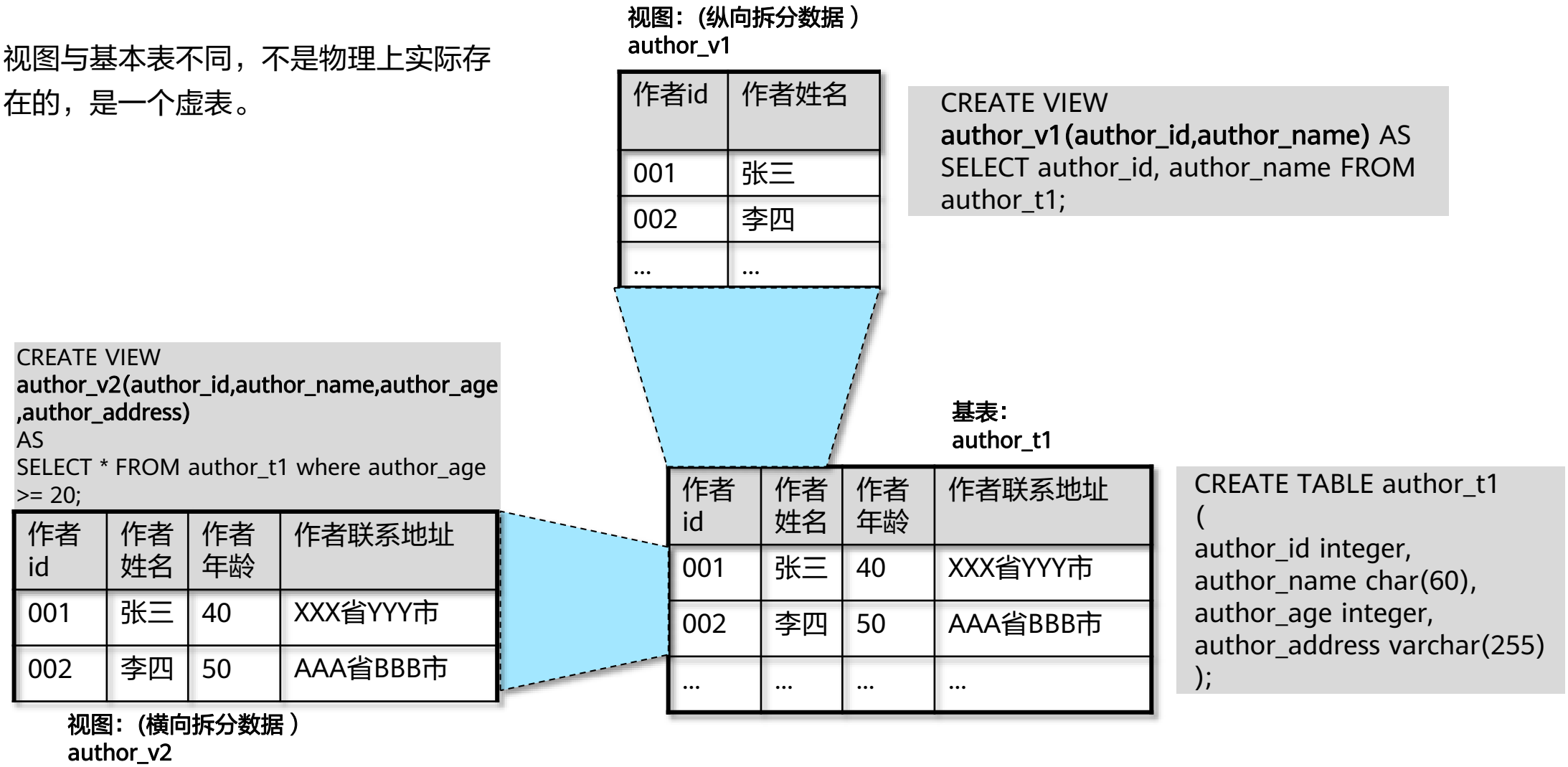
数据类型	说明	数据类型	说明
smallint	2字节常用整数，取值范围是-32768 到 +32767	varchar(n)	变长，有长度限制n
integer	4字节常用整数，取值范围是-2147483648 到 +2147483647	char(n)	定长，不足补空白
bigint	8字节常用整数，取值范围是-9223372036854775808 到 9223372036854775807	text	变长，长文本数据
decimal	精度数字。decimal(m,n)是精确到小数点后n位数字	date	3字节，以YYYY-MM-DD的格式显示，比如：2009-07-19
numeric	精度数字，等同于decimal	time	3字节，以HH:MM:SS的格式显示。比如：11：22：30
float	4字节，单精度浮点型数字	timestamp	4字节，年月日时分秒
double	8字节，双精度浮点型数字	boolean	1字节, TRUE/FALSE

字段设计建议

- 尽量使用高效数据类型
  - 尽量使用执行效率比较高的数据类型
  - 尽量使用短字段的数据类型
  - 使用一致的数据类型
- 当多个表存在逻辑关系时，表示同一含义的字段应该使用相同的数据类型。
- 对于字符串数据，建议使用变长字符串数据类型，并指定最大长度。

视图 (View)

- 视图与基本表不同，不是物理上实际存在的，是一个虚表。



视图的作用

- 视图作用
  - 简化操作，把经常使用的数据定义为视图。
  - 安全性，用户只能查询和修改能看到的数据。
  - 逻辑上的独立性，屏蔽了真实表的结构带来的影响。
- 限制性
  - 性能问题：查询可能很简单，但是封装的视图语句很复杂。
  - 修改限制：对于复杂视图，用户不能通过视图修改基表数据。

通过视图封装较为复杂的逻辑

```
CREATE VIEW stu_class(id,name,class)
AS
select student.s_id,student.name,stu_info.class
from student, stu_info
where student.s_id=stu_info.s_id;
```

```
select * from stu_class where
class='Beijing'
```

用户使用的时候和普通表一样，简化SQL查询语句

索引 (Index)

- 索引提供指向存储在表的指定列中的数据值的指针，如同图书的目录，能够加快表的查询速度，但同时也增加了插入、更新和删除操作的处理时间。
- 在创建索引时，以下建议作为参考：
  - 在经常需要搜索查询的列上创建索引，可以加快搜索的速度。
  - 在作为主键的列上创建索引，强制该列的唯一性和组织表中数据的排列结构。
  - 在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的。
  - 在经常需要排序的列上创建索引，因为索引已经排序，这样查询可以利用索引的排序，加快排序查询时间。
  - 在经常使用WHERE子句的列上创建索引，加快条件的判断速度。
  - 为经常出现在关键字ORDER BY、GROUP BY、DISTINCT后面的字段建立索引。

有效索引

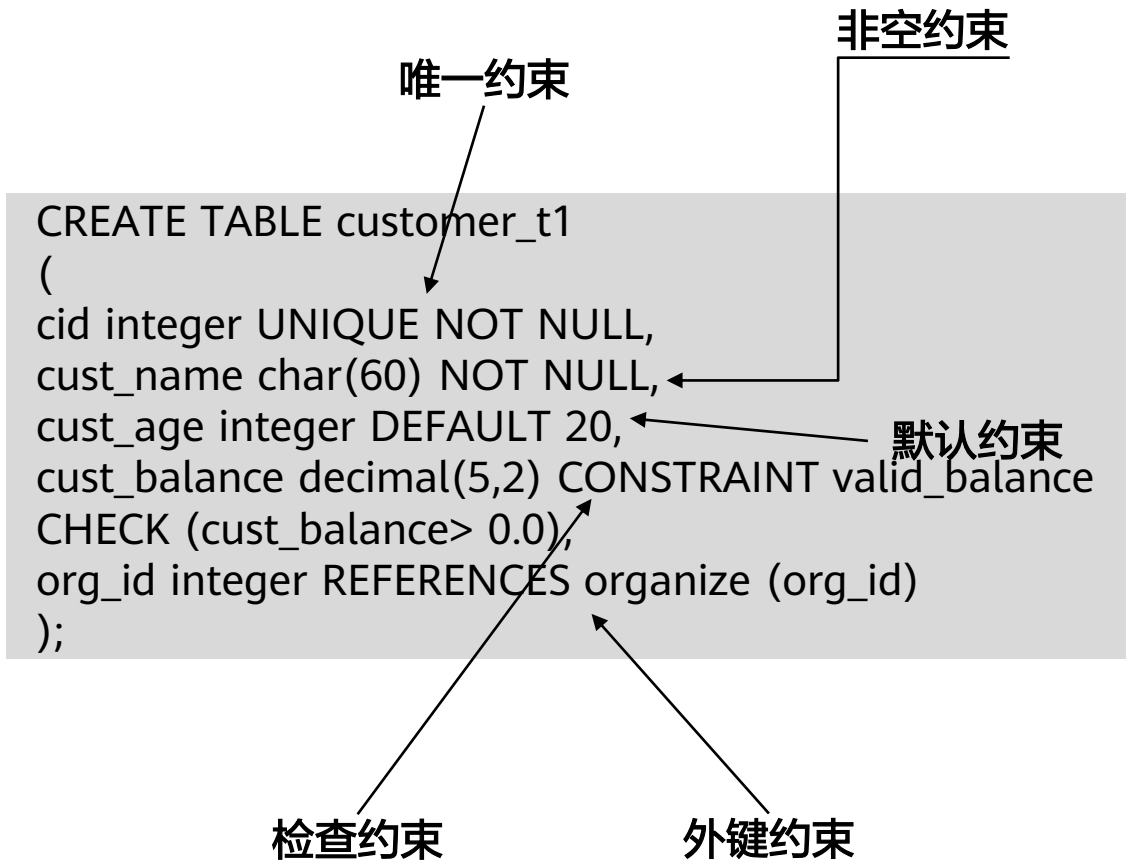
- 创建索引 ≠ 索引一定被使用
  - 索引创建成功后，系统会自动判断何时引用索引。当系统认为使用索引比顺序扫描更快时，就会使用索引。
  - 索引创建成功后，必须和表保持同步以保证能够准确地找到新数据，这样就增加了数据操作的负荷。
  - 需要定期删除无用的索引。
- 判断方法
  - 通过执行explain语句查看执行计划来判断是否使用索引。

索引方式

索引方式	描述
普通索引	基本索引类型，没有什么限制，允许在定义索引的列中插入重复值和空值，只是为了加快查询。
唯一索引	索引列中的值必须是唯一的，但是允许为空值。
主键索引	是一种特殊的唯一索引，不允许有空值。
组合索引	在表中的多个字段组合上创建的索引，只有在查询条件中使用了这些字段的左边字段时，索引才会被使用。
全文索引	主要用来查找文本中的关键字，而不是直接与索引中的值相比较。

约束

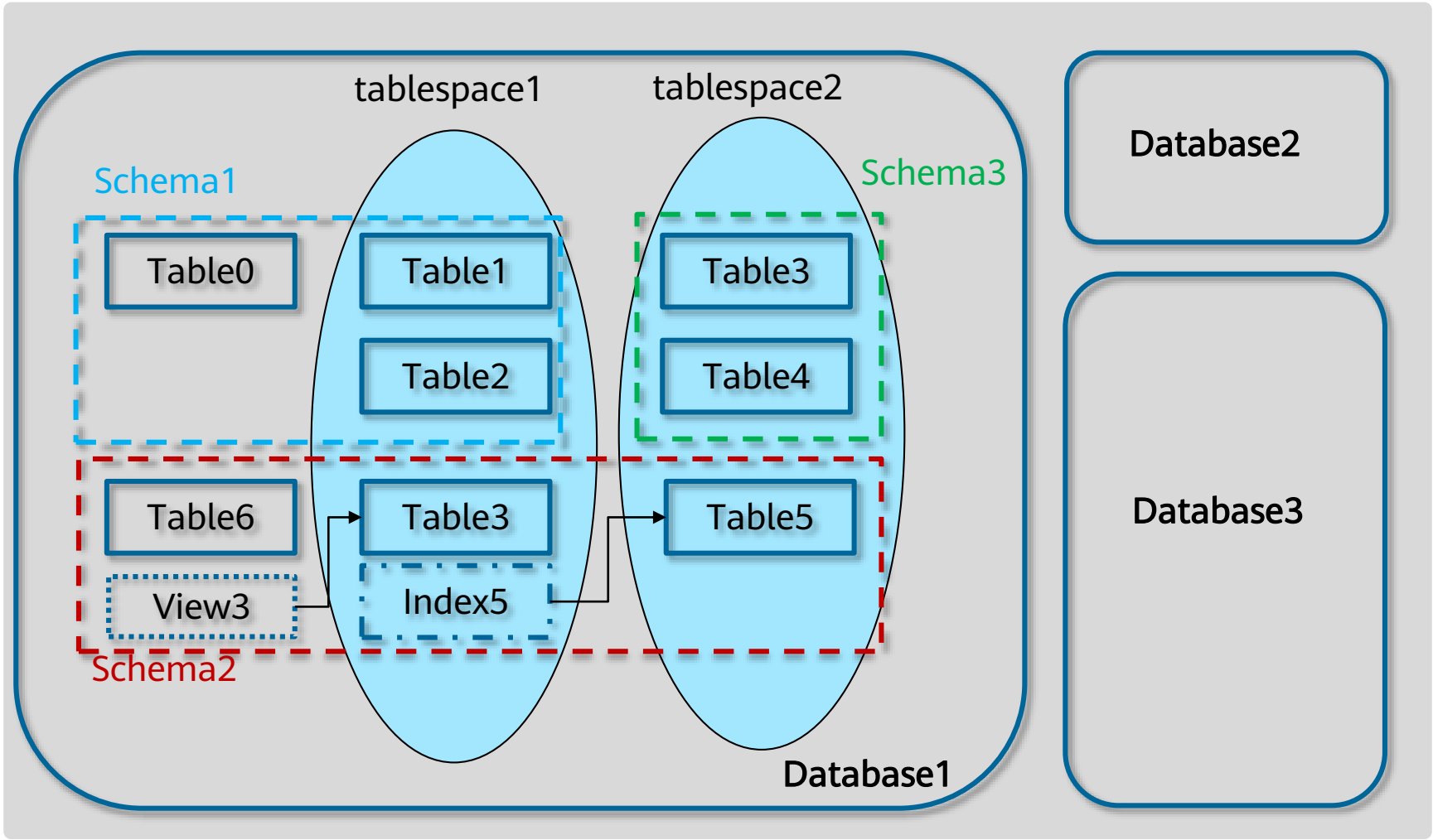
- 数据的完整性是指数据的正确性和一致性，可以通过定义表时定义完整性约束。
  - 完整性约束是一种规则，本身不占用数据库空间；
  - 完整性约束和表结构定义一起保存在数据字典中。
- 常见的约束类型
  - 唯一性和主键约束(UNIQUE/PRIMARY KEY)
  - 外键约束(FOREIGN KEY)
  - 检查约束(CHECK)
  - 非空约束(NOT NULL)
  - 默认约束(DEFAULT)



约束的设计

- 如果能够从业务层面补全字段值，就不建议使用DEFAULT约束，避免数据加载时产生不符合预期的结果。
- 给明确不存在NULL值的字段加上NOT NULL约束，优化器会对其进行自动优化。
- 给可以显式命名的约束显式命名。除了NOT NULL和DEFAULT约束外，其他约束都可以显式命名。

数据库对象间关系



事务 (Transaction)

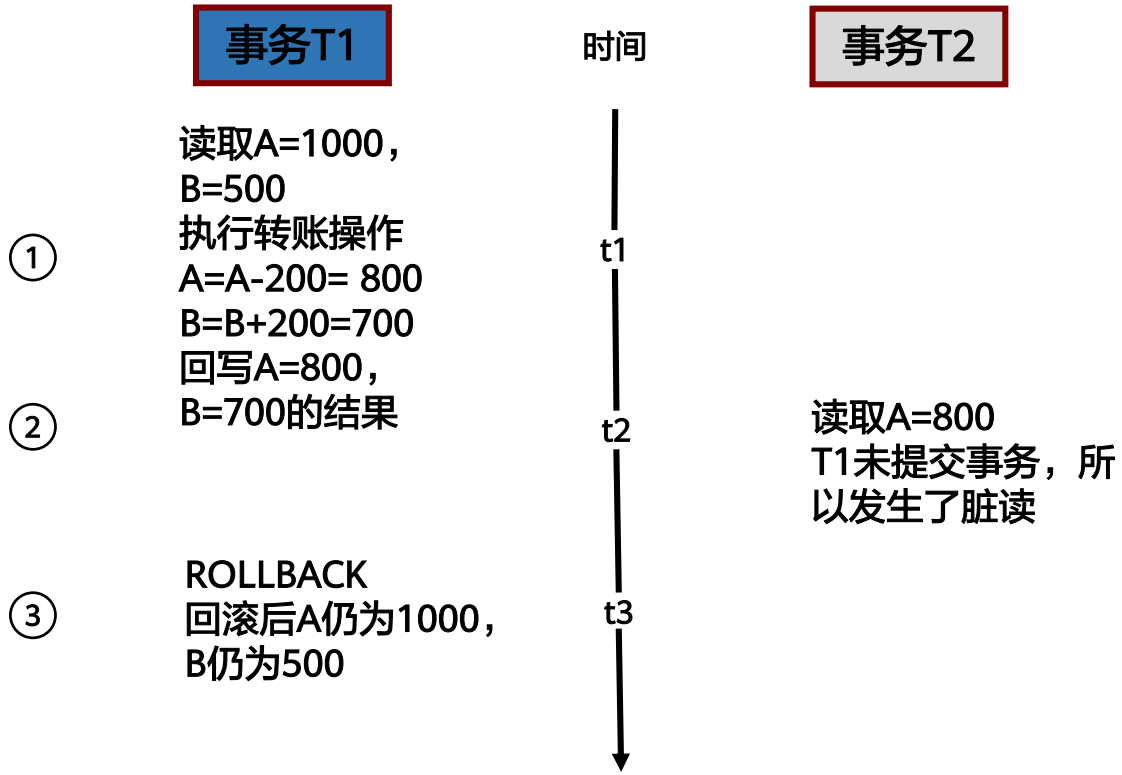
- 事务是用户定义的数据操作系列，这些操作作为一个完整的工作单元执行。
  - 原子性(Atomicity)：事务是数据库的逻辑工作单位，事务中的操作，要么都做，要么都不做。
  - 一致性(Consistency)：事务的执行结果必须是使数据库从一个一致性状态转到另一个一致性状态。
  - 隔离性(Isolation)：数据库中一个事务的执行不能被其他事务干扰。即一个事务的内部操作及使用的数据对其他事务是隔离的，并发执行的各个事务不能相互干扰。
  - 持久性(Durability)：事务一旦提交，对数据库中数据的改变是永久的。提交后的操作或者故障不会对事务的操作结果产生任何影响。
- 事务结束的标记有两个：
  - 正常结束，COMMIT（提交事务）。
  - 异常结束，ROLLBACK（回滚事务）。

事务处理模型

- 事务处理模型分为两类：
  - 隐式事务：每一条数据操作语句都自动地成为一个事务。GaussDB(for MySQL) 默认是隐式提交。
  - 显式事务：事务有显式的开始和结束标记。

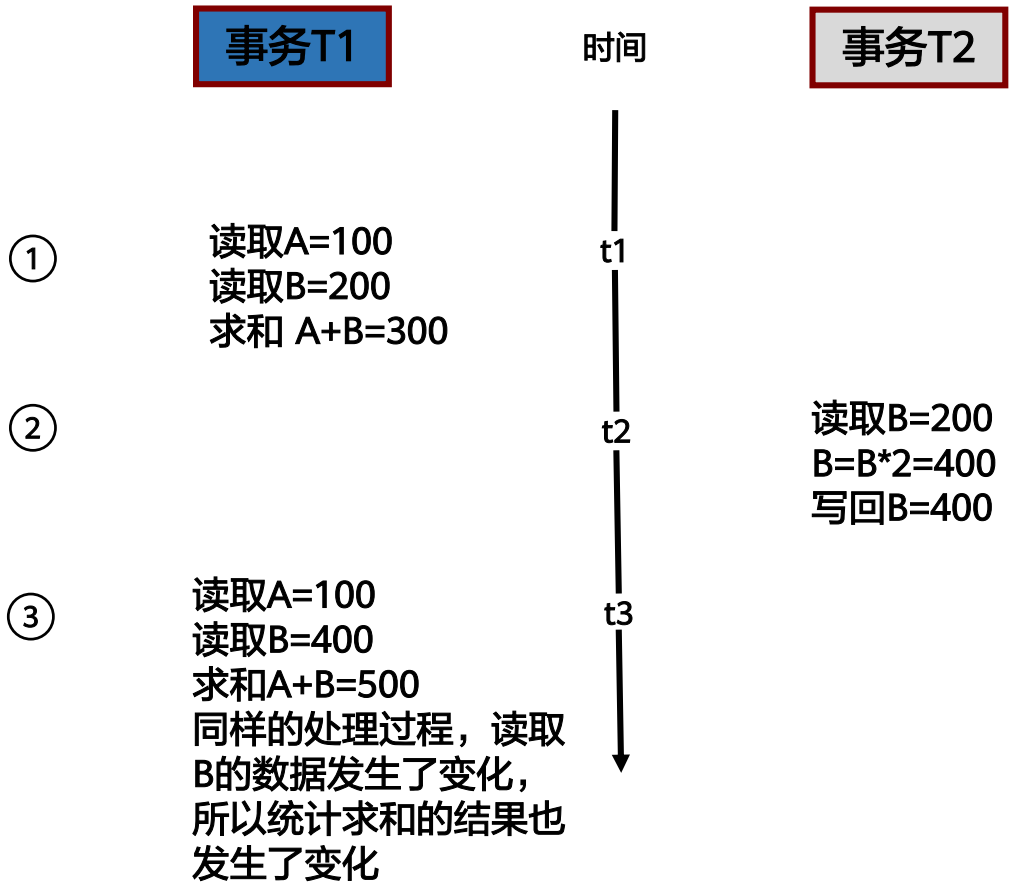
数据不一致情况 - 脏读

- ”Dirty” Reads（脏读）
  - 一个事务读取到了其他事务中还没有提交(Committed)的数据。
  - 因为未提交数据存在回滚的可能，所以被称为“脏”数据。



数据不一致情况 - 不可重复读

- Non-repeatable Reads（不可重复读）
  - 一个事务所获取到的数据是可以被其它事务修改的。
  - 一个事务在处理过程中多次读取同一个数据（重复读），这个数据是可能发生变化的，因此被称为不可重复读。
- Phantom Read（幻影读）
  - 是不可重复读的更为特殊的一个场景。
  - 事务T1按照一定条件读取数据（使用了where条件过滤）后，事务T2删除了部分记录或者插入了一些新的记录，这些变更的数据是满足where条件过滤的。
  - 那么当T1再次按照相同条件读取数据时，就会发现莫名其妙地少了（也可能多了）一些数据。
  - 这些变化的数据就被称为“幻影”数据。



事务隔离级别 (1)

- ANSI SQL标准定义了4种事务隔离级别来避免3种数据不一致的问题。事务等级从高到低，分别为：
  - Serializable（序列化）
    - 系统中所有的事务以串行地方式逐个执行，所以能避免所有数据不一致情况。
    - 但是这种以排他方式来控制并发事务，串行化执行方式会导致事务排队，系统的并发量大幅下降，使用的时候要绝对慎重。
  - Repeatable read（可重复读）
    - 一个事务一旦开始，事务过程中所读取的所有数据不允许被其他事务修改。
    - 这个隔离级别没有办法解决“幻影读”的问题。
    - 因为它只“保护”了它读取的数据不被修改，但是其他数据会被修改。如果其他数据被修改后恰好满足了当前事务的过滤条件（where语句），那么就会发生“幻影读”的情况。

事务隔离级别 (2)

- 其他两种事务隔离等级为：
  - Read Committed（已提交读）
    - 一个事务能读取到其他事务提交过(Committed)的数据。
    - 一个事务在处理过程中如果重复读取某一个数据，而且这个数据恰好被其他事务修改并提交了，那么当前重复读取数据的事务就会出现同一个数据前后不同的情况。
    - 在这个隔离级别会发生“不可重复读”的场景。
  - Read Uncommitted（未提交读）
    - 一个事务能读取到其他事务修改过，但是还没有提交的(Uncommitted)的数据。
    - 数据被其他事务修改过，但还没有提交，就存在着回滚的可能性，这时候读取这些“未提交”数据的情况就是“脏读”。
    - 在这个隔离级别会发生“脏读”场景。

事务隔离级别与问题对应表

事务隔离级别	脏读	不可重复读	幻影读
未提交读	可能	可能	可能
已提交读	不可能	可能	可能
可重复读	不可能	不可能	可能
序列化	不可能	不可能	不可能

本章总结

- 说明了数据库管理的核心目标，并介绍了数据库管理的工作范围：
  - 介绍了数据库对象管理的工作内容；
  - 介绍了备份恢复的基本概念，灾难恢复等级以及相关概念，并介绍了不同备份策略以及之间的差异；
  - 介绍了数据库系统安全框架和控制模型；
  - 介绍了数据库性能管理的意义和目标，以及性能管理工作的一些场景和工作内容。
- 对数据库主要的概念进行了介绍和说明：
  - 针对一些容易混淆的概念进行了对比说明；
  - 并对重要但是不宜理解的概念进行了基于场景的介绍分析。

学习推荐

欢迎关注我们：
华为人才在线微信公众号


谢 谢
www.huawei.com