W9. Object-based database

· 复杂数据类型和面向对象
· 对象关系系统
· SQL中的结构化数据类型和继承
· 表继承
· SQL中的数组和多重集类型
· 对象关系映射系统
· 数据库比较

1. 复杂数据类型基础

1.1 动机与定义

- 非原子域(Non-atomic Domains)

  :

  - 传统关系型数据库只允许原子(不可分)值
  - 复杂数据类型允许非原子值，如整数集合、元组集合等
  - 这种扩展违反了第一范式(1NF)，但保留了关系模型的数学基础

1.2 三种集成方式

1. 对象关系数据库系统:
   - 向现有关系数据库添加面向对象特性
   - 保持关系型数据库核心功能
2. 对象关系映射(ORM):
   - 自动转换对象和关系数据
   - 在程序语言对象系统和关系存储之间建立桥梁
3. 纯面向对象数据库:
   - 直接支持对象存储
   - 优点是更好的语言集成
   - 缺点是缺乏声明式查询能力

2. 对象关系数据模型

2.1 主要特点

- 扩展了关系数据模型，包含面向对象构造
- 允许元组属性具有复杂类型，如嵌套关系
- 保持关系基础，特别是声明式数据访问
- 向上兼容现有关系语言

2.2 结构化类型(Structured Types)

- 用户可自定义复杂数据类型
- 支持类型继承(Type Inheritance)
- 支持方法(Methods)定义
- 支持构造函数(Constructor Functions)

2.3 集合类型

- 数组(Array)：有序集合
- 多重集(Multiset)：无序集合
- 支持嵌套关系(Nested Relations)

3. 对象身份与引用类型

3.1 基本概念

- 对象标识符(Object Identifier)
- 引用类型(Reference Types)
- 支持系统生成和用户生成的标识符

3.2 路径表达式(Path Expressions)

- 用于简化对象导航
- 避免显式连接操作
- 使查询表达更简洁直观

4. 对象关系映射(ORM)

4.1 主要特征

- 建立在传统关系数据库之上
- 提供对象到关系的自动映射
- 对象为临时性，无永久对象标识

4.2 应用架构

- 采用MVC(Model-View-Controller)模式
- 包含数据访问层(Data Access Layer)
- 典型实现：Hibernate ORM系统

5. 数据库比较

5.1 关系系统

- 简单数据类型
- 强大查询语言
- 高度保护

5.2 持久化编程语言型面向对象数据库

- 复杂数据类型
- 与编程语言集成
- 高性能

5.3 对象关系系统

- 复杂数据类型
- 强大查询语言
- 高度保护

5.4 对象关系映射系统

- 复杂数据类型
- 与编程语言集成
- 建立在关系数据库之上

W9.b Distributed Database

1.1 分布式系统定义（Definition）

- A distributed database system consists of loosely coupled sites that share no physical component. Each site is autonomous and can operate independently.
- 分布式数据库系统由松散耦合的多个站点组成，各站点之间没有共享的物理组件。

1.2 分布式特性（Key Features）

1. 分布式数据独立性（Distributed Data Independence）：
    - 用户不需要知道数据存储的位置，引用关系或片段即可访问数据。
2. 分布式事务原子性（Distributed Transaction Atomicity）：
    - 事务可以跨多个站点访问和更新数据，确保原子性和一致性。

2. 分布式数据库的类型（Types of Distributed Databases）

1. 同构分布式数据库（**Homogeneous** DDBMS）：
    - 所有站点运行相同的 DBMS 软件。
2. 异构分布式数据库（**Heterogeneous** DDBMS）：
    - 不同站点运行不同的 DBMS 软件，通过网关协议（如 ODBC、JDBC）进行通信。

---

3. 分布式数据库的体系结构（Architectures）

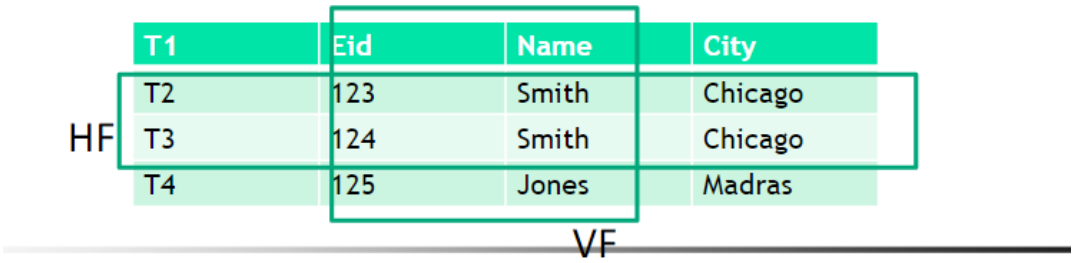
1. 客户端/服务器（Client/Server）：
    - 客户端发送查询，服务器处理并返回结果。
2. 协作服务器（Collaborating Server）：
    - 多个服务器协作完成跨站点事务。
3. 中间件（Middleware）：
    - 中间层管理分布式查询和事务，但不存储数据。

---

4. 数据存储方式（Data Storage Methods）

4.1 数据分片（Data Fragmentation）

- 水平分片（Horizontal Fragmentation）：
    - 按行分割数据，例如按城市分片。
- 垂直分片（Vertical Fragmentation）：
    - 按列分割数据，例如选择部分列。



4.2 数据复制（Data Replication）

对于同一个数据存多个副本，可以提高复用性和查询效率

- 同步复制（Synchronous Replication）：
  - 在事务提交前更新所有副本，确保一致性。
- 异步复制（Asynchronous Replication）：
  - 副本更新可以延迟，适合高性能需求。

5. 分布式事务（Distributed Transactions）

5.1 事务协调器（Transaction Coordinator）

- 功能：
  1. 启动事务并分解为子事务。
  2. 分发子事务到相关站点。
  3. 统一协调事务的提交或回滚。

5.2 分布式锁定协议（Distributed Locking Protocols）

1. 集中式锁管理（Centralised Locking）：
   - 单一站点管理所有锁。
2. 主副本锁管理（Primary Copy Locking）：
   - 只允许主副本站点管理锁请求。
3. 分布式锁管理（Distributed Locking）：
   - 每个站点管理其本地数据的锁请求。

6. 分布式死锁（Distributed Deadlock）

6.1 检测方法

1. 集中式检测：
   - 所有站点的等待图发送到一个站点，由该站点检测死锁。
2. 分层检测：
   - 按层级结构发送等待图，逐级检测。
3. 超时机制：
   - 如果事务等待时间过长，则自动回滚。

7. 故障恢复（Failure Recovery）

7.1 恢复过程

- 协调者故障（Coordinator Failure）：
  - 如果协调者故障，事务的状态可能“不确定（in-doubt）”，恢复后需要重新协调。
- 子事务故障（Subordinate Failure）：
  - 检查日志决定事务是提交还是回滚。

8. 提交协议（Commit Protocols）

8.1 两阶段提交协议（Two-Phase Commit, 2PC）

1. 第一阶段：准备阶段（Prepare Phase）：
   - 协调者发送“准备”消息给所有子事务。
   - 子事务根据其本地状态返回“同意”或“中止”消息。
2. 第二阶段：提交阶段（Commit Phase）：
   - 如果所有子事务返回“同意”，协调者发送“提交”消息；否则，发送“回滚”消息。

9. 分布式查询处理（Distributed Query Processing）

9.1 查询优化

- 考虑通信成本（如数据传输）。
- 利用分布式连接算法（如半连接和布隆连接）。