PART Ⅰ 基础篇

CHAPTER 1 数据库系统引论

What

1. 数据密集型应用（data intensive application）
2. 数据管理（data management）
3. 数据库、DBMS&数据库系统（database, DBMS & database system）
4. 传统数据库&后关系数据库（traditional database & post-relational database）
5. 数据的语法/语义（data syntax/semantics）
6. **数据模型/模式（data model/schema）**
7. **多级数据模型/模式（multilevel data model/schema）**

Why

1. 作为数据管理技术，为什么数据库系统比文件系统优越？
2. **为什么数据库中要采用多级数据模型/模式？**

CHAPTER 2 数据模型

What

1. **关系（数据）模型/模式（relational data model/schema）**
2. 关系/表、元组/行、属性/列、域/数据类型（relation/table, tuple/row, attribute/column, domain/data type）
3. **完整性约束及其类型与作用（integrity constraints and their types）**
4. **键、超键、主键、外键（key, super-key, primary key, foreign key）**
5. 关系代数操作&关系演算（relational algebra operation & relational calculus）
6. **选择、投影、连接/笛卡尔积、并、差（select, project, join/Cartesian product, union, difference）**
7. **关系代数表达式（relational algebra expression）**
8. 关系完备操作集&关系完备系统（relationally complete operation set & relationally complete system）
9. **E/R（数据）模型/模式，E/R 图（Entity-Relationship data model/schema, E/R graph）**

10. 实体、弱实体、属性、联系&联系的语义（entity, weak entity, attribute, relationship & relationship’s semantics）

11. 传统数据模型&后关系（数据）模型（traditional data model & post-relational data model）

Why

1. 为什么关系数据模型会取代层次和网状数据模型，成为数据库的主流数据模型？
2. 为什么要发展后关系数据模型？
3. 为什么 E/R 模型能成为数据库概念设计的有力工具？

How

1. 如何计算一个关系代数表达式？
2. **一个关系代数表达式如何表示成一棵语法树？**
3. **如何求一个关系模式的键、超键？如何确定主健、外键？**
4. 如何评价传统数据模型？

CHAPTER 3 关系数据库语言 SQL

What

1. 数据库操作&数据库语言（DDL，QL，DML，DCL）（database operations, database language: Data Definition Language, Query Language, Data Manipulation Language, Data Control Language）
2. 面向记录 vs 面向集合的数据库语言（record-oriented / set-oriented database language）
3. 导航式访问 vs 联想式访问（navigational access / associative access）
4. 交互式 vs 嵌入式数据库语言（interactive/embedded database language）
5. 过程性 vs 说明性数据库语言（procedural/declarative database language）
6. 计算完备 vs 不完备的语言（computing complete/incomplete language）
7. 数据库用户接口&前端开发工具（database user interface & front development tool）
8. **基表&视图（base-table & view）**

9. SQL 语言标准（SQL-89, SQL2, SQL3/SQL:1999）/ 实现（SQL standard/implementation）
10. **SQL DDL：CREATE TABLE, CREATE VIEW**
11. **SQL QL：SELECT**
12. **SQL DML：INSERT, DELETE, UPDATE**
13. 嵌入式 SQL（embedded SQL）
14. SQL 过程化扩充（SQL procedural extension）

Why

1. 为什么 SQL 会成为关系数据库的标准语言？
2. SQL 为什么要进行过程化扩充？

How

1. 如何使用各种（交互式）SQL 语句？

PART Ⅱ 系统篇

CHAPTER 4 数据库管理系统引论

What

1. DBMS 的组成结构&功能（DBMS' architecture & functionality）
2. DBMS 的进程结构&多线程 DBMS（DBMS' process structure & Multithreading DBMS）
3. 数据库系统的体系结构&多层结构（C/S, 三层结构）（Database system architecture & multi-tier architecture）
4. **事务&ACID 性质（transaction & ACID properties）**
5. 显式/隐式的事务提交与回滚（explicit/implicit transaction commit & rollback）
6. 元数据&数据字典（metadata & date directory, catalog or dictionary）

Why

1. **事务为什么要具有 ACID 性质？**
2. 数据库系统为什么要向多层结构演变？
3. 数据库系统中为什么要建立数据字典？

How

1. SQL 中如何实现事务的提交&回滚？

CHAPTER 5 数据库的存储结构

What

1. 数据库的多级存储（database multilevel storage）
2. 数据库物理结构&数据文件、日志文件、控制文件（database physical structure & data file, log file, control file）
3. **基表的典型存储结构：表、索引表、索引簇表、散列簇表（typical base-table storage structures: table, indexed table, indexed cluster, and hash cluster）**
4. B 树索引（B-tree index）
5. 数据库的逻辑存储结构：逻辑存储空间&表空间，用户模式&模式对象（database logical storage structure: logical storage space & table space, user's schema & schema object）

Why

1. 为什么数据库中要引入逻辑结构的概念？
2. **为什么基表数据的存储要使用索引、簇集等机制？**

CHAPTER 6 查询处理和优化

What

1. 查询（query）
2. 查询处理&求值，查询引擎（query processing & evaluation, query engine）
3. 查询执行计划＆优化，优化器（query execution plan & optimization, optimizer）
4. 查询优化的方法：层次、目标&策略（optimization approach: level, objective & strategy）
5. **代数优化（algebraic optimization）**
6. **依赖于存取路径的规则优化（access-path-dependent rule-based optimization）**
7. 代价估算优化（cost-evaluation-based optimization）

Why

1. 关系系统中为何要进行查询优化？

How

1. **如何用查询语法树表示代数优化的过程？**
2. 连接操作如何实现？

CHAPTER 7 事务管理

What

1. 事务管理（transaction management）
2. 数据库恢复（database recovery）
3. 后备副本（backup）
4. 日志及其结构（log and its structure）
5. **前像/后像&撤消/重做（before image/after image & undo/redo）**
6. 向前恢复/向后恢复（forward recovery/backward recovery）
7. **提交规则&先记后写规则（commit rule & log ahead rule）**
8. **三类数据库故障及其恢复对策（3 types of failure and their recovery strategies）**
9. 并发访问&并发控制（concurrent access & concurrency control）
10. **并发控制的正确性准则（correctness guideline of concurrency control）**
11. **合式事务&两段事务（well-formed transaction & two-phase transaction）**
12. **加锁协议&两段封锁协议（locking protocol & two-phase licking protocol, 2PL）**
13. 各类锁：排它锁，共享锁，共享更新锁（exclusive lock, sharing lock sharing update lock）
14. 封锁粒度、多粒度封锁&意向锁（locking granularity, multiple-granularity locking & intent lock）
15. 死锁 vs 活锁 （dead lock & live lock）

Why

1. 为什么数据库要定期备份？不停地建日志？
2. **为什么更新事务执行要遵循提交规则&先记后写规则？**
3. 为什么要进行并发控制？
4. 事务为什么要遵守两段封锁协议？

How

1. 更新事务如何执行？
2. **各类数据库故障如何恢复？**
3. 如何保证并发控制的正确性？
4. **数据库系统中如何防止死锁发生？**

CHAPTER 8 数据库的安全与完整性约束

What

1. 数据库安全（database security）
2. 数据库用户名&口令（database user name & password）
3. 访问控制（access control）
4. 授权：集中式 vs.分散式（authorization: centralized vs. decentralized）
5. 特权&角色（privilege & role）
6. 数据库审计&审计痕迹（database audit & audit trail）
7. 数据加密（data encryption）
8. **完整性约束及其类型（integrity constraints and their types）**
9. **完整性约束的 SQL 实现机制：非空、唯一列、主键、外键、检验、断言、触发器（SQL mechanisms of integrity constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, ASSERTION, TRIGGER）**
10. 用过程说明约束（declaring constraints with procedure）

Why

1. 数据库中为何要引入安全机制？
2. 商用数据库产品为何不支持数据加密？
3. **数据库中为什么要引入完整性约束机制？**
4. **为什么用过程说明完整性约束不好？**

How

1. 如何用 SQL DDL 创建/删除角色？
2. 如何用 SQL DCL 授权/收权？
3. **如何用 SQL 实现关系数据库的各种完整性约束?**

CHAPTER 9 触发子与主动数据库系统

What

1. 被动数据库系统（passive database system）与主动数据库系统（active database system）
2. **ECA 规则/触发器（Trigger）及其各种类型**
3. **触发器有哪些典型的用途?**

Why

1. 数据库中为何要引入主动机制？

How

1. **如何用 SQL's CREATE TRIGGER 语句定义各类触发器？**

PART Ⅲ 应用篇

CHAPTER 10 数据依赖与关系模式规范化

What

1. 冗余&更新异常（redundancy & update anomaly）
2. **模式的规范化&模式分解（schema normalization & decomposition）**
3. **函数依赖&决定子（functional dependency& determinant）**
4. **平凡/非平凡/完全非平凡函数依赖（trivial/nontrivial/full nontrivial functional dependency）**
5. **完全/部分函数依赖（full/part functional dependency）**
6. **传递函数依赖（transitive functional dependency）**
7. **分裂/合并规则，平凡依赖规则，传递规则（splitting/combining Rule, trivial-dependency rule, and transitive rule）**
8. **范式，1NF/非第一范式条件，2NF，3NF，BCNF（normal form, non-first normal form, NF2）**
9. **全键关系模式（all-key relational schema）**
10. 函数依赖集及其闭包（functional dependency set and its closure）
11. **逻辑蕴涵（logically implicating）**
12. **关系模式的一个分解，无损/保持依赖分解（decomposition, lossless/**

preserve-dependency decomposition）

Why

1. 关系模式为何要规范化？

2. 为什么规范化程度并非越高越好？具体策略是什么？

How

1. 如何判定一个关系模式是否属于某个范式？

2. 如何将一个关系模式无损分解到 BCNF/3NF?

CHAPTER 11 数据库设计

What

1. DB 的生命周期（database's life cycle）
2. **DB 设计的任务、目标、方法、特点、步骤/阶段（task, objective, approach, characteristic, stages/phases of database design）**
3. 需求分析&需求规格说明（requirements analysis & specification of requirements）
4. DB 概念设计&视图集成（DB conceptual design & view integration）
5. **DB 逻辑设计&模式转换（DB logical design）**
6. DB 物理设计&存储结构选择（DB physical design & storage structure selection）

Why

1. 为什么数据库设计要采用面向数据的方法？
2. 为什么数据库设计要分阶段进行？
3. **为什么数据库概念设计时要进行概念（e.g. E/R）建模？**
4. 为什么数据库逻辑设计时要进行规范化与逆规范化？
5. 为什么数据库物理设计时要精心选择基表的存储结构？

How

1. 如何进行视图集成？
2. **如何将 E/R 数据模式转换为关系数据库模式？**
3. 如何为基表选择合适的存储结构？

CHAPTER 12 数据库管理

What

1. 数据库管理&数据库管理员（database management & Database Administrator, DBA）

2. DBA 的职责（responsibilities of DBA）

3. DB 调整、重组与重构（DB adjustment，reorganization & restructuring）

Why

1. 为什么数据库系统在运行过程中要由 DBA 实施管理？