ch00-07（文档11、21）

核心内容：数据库基础入门知识，为后续章节铺垫核心概念。

- 数据库系统概述：介绍数据库的定义、发展历程、核心特点及应用场景。
- 数据模型：讲解概念模型、逻辑模型（层次、网状、关系模型）的核心思想与区别。
- 关系数据库基础：初步介绍关系、属性、元组、码等基本概念，为SQL学习奠定理论基础-。

---

ch08-10（文档12、22）

核心内容：关系数据库核心理论，衔接入门知识与SQL实操。

- 关系代数：详细讲解并、差、笛卡尔积、选择、投影、连接等基本运算及组合应用。
- 关系演算：介绍元组关系演算和域关系演算的基本规则与表达方法。
- 数据库系统架构：说明数据库系统的三级模式（外模式、模式、内模式）和两级映像（外-模式-模式、模式-内模式），解释数据独立性原理。

---

ch11-SQL概述（文档1、23）

核心内容：SQL语言的基础概念与核心特性。

- 定义与定位：SQL（Structured Query Language）是关系数据库的标准语言，功能通用-且强大。
- 核心特点：包括综合统一（集DDL、DML、DCL于一体）、高度非过程化（只需明确“做-什么”）、面向集合的操作方式、多使用场景（独立交互+嵌入式）、语言简洁（核心功能-仅9个动词）。
- 与三级模式的对应：基本表对应模式、视图对应外模式、存储文件对应内模式，明确各层-级的关系与作用。

ch12-SQL数据定义（文档1、23）

核心内容：SQL中数据库对象的定义、修改与删除操作。

- 命名机制：采用层次化命名，实例→数据库→模式→表/视图/索引的层级结构。
- 模式操作：CREATE SCHEMA定义命名空间，DROP SCHEMA（支持CASCADE/-RESTRICT）删除模式及下属对象。
- 基本表操作：CREATE TABLE定义表结构与完整性约束，ALTER TABLE修改列定义或约-束，DROP TABLE删除表（级联删除依赖对象）。
- 数据类型：涵盖字符型、数值型、日期时间型等常用类型及适用场景。
- 索引操作：CREATE INDEX建立索引（支持UNIQUE、CLUSTER属性），ALTER INDEX重命名，DROP INDEX删除索引。

---

ch13-SQL数据查询（单表）（文档1、23）

核心内容：单表查询的语法与各类查询场景实现。

- 基本语法：SELECT子句指定目标列，FROM子句指定查询对象，搭配WHERE、GROUP BY、HAVING、ORDER BY子句实现复杂查询。
- 列选择：查询指定列、全部列，支持计算表达式作为目标列，可通过别名优化结果展示。
- 元组选择：通过DISTINCT消除重复行，支持比较大小、范围匹配（BETWEEN AND）、集合匹配（IN）、字符匹配（LIKE）、空值判断（IS NULL）及多重条件组合（AND/OR/NOT）。
- 结果处理：ORDER BY排序（ASC/DESC），聚集函数（COUNT、SUM、AVG、MA-X、MIN）统计数据，GROUP BY分组结合HAVING筛选分组结果。

---

ch14-SQL数据查询（连接）（文档1、23）

核心内容：多表连接查询的实现方式与优化。

- 连接基础：连接查询涉及多个表，通过连接谓词指定关联条件，连接字段类型需可比。
- 连接类型：包括等值连接、自然连接、自身连接（表别名区分）、外连接（左外/右外，保-留主表未匹配元组）、多表连接（多个表联合查询）。
- 执行机制：讲解嵌套循环法、排序合并法、索引连接三种连接算法的执行逻辑。

---

ch15-SQL数据查询（嵌套）（文档1、23）

核心内容：嵌套查询的语法、分类与应用场景。

- 基本概念：外层查询（父查询）与内层查询（子查询）嵌套，支持多层嵌套，子查询不可-用ORDER BY。
- 分类：不相关子查询（子查询结果独立于父查询）与相关子查询（子查询依赖父查询参数-）。
- 常用谓词：IN（匹配子查询结果集）、比较运算符（适用于子查询返回单值）、-ANY/ALL（与比较运算符搭配，匹配部分/全部结果）、EXISTS（判断子查询结果是否非-空，支持NOT EXISTS）。

---

ch16-SQL数据查询（集合）（文档1、23）

核心内容：集合运算在查询中的应用。

- 运算类型：UNION（合并结果并去重）、UNION ALL（合并结果保留重复）、-INTERSECT（取结果交集）、EXCEPT（取结果差集）。
- 应用规则：参与运算的查询结果需列数相同、对应列数据类型一致。
- 实例场景：如查询选修课程1或课程2的学生（UNION）、查询既选修课程1又选修课程2的-学生（INTERSECT）。

---

ch17-SQL数据查询（基于派生表）（文档1、23）

核心内容：利用派生表优化复杂查询。

- 定义：子查询作为FROM子句的查询对象，生成临时派生表。
- 应用场景：简化复杂嵌套查询，如找出学生超过自身选修课程平均成绩的课程号，通过派-生表存储平均成绩后关联查询。
- 注意事项：若子查询无聚集函数，可省略属性列定义，默认沿用子查询目标列名。

---

ch18-SQL数据更新（插入）（文档1、23）

核心内容：向表中插入数据的两种方式。

- 插入元组：INSERT INTO…VALUES语句，可指定部分列（其余列取空值）或全部列。
- 插入子查询结果：INSERT INTO…子查询，将查询结果批量插入目标表，需保证列数与-数据类型匹配。
- 实例：如创建部门平均年龄表并插入各系平均年龄数据。

---

ch19-SQL数据更新（修改）（文档1、23）

核心内容：修改表中已有数据的操作。

- 基本语法：UPDATE…SET…WHERE，SET子句指定修改表达式，WHERE子句限定修改-范围（省略则修改全表）。
- 应用场景：修改单个元组、批量修改（如所有学生年龄加1）、带子查询的修改（如计算机-系学生成绩置零）。
- 完整性检查：修改操作需满足表定义的实体完整性、参照完整性及用户定义完整性。

---

ch20-SQL数据更新（删除）（文档1、23）

核心内容：删除表中数据的操作。

- 基本语法：DELETE FROM…WHERE，WHERE子句限定删除范围（省略则删除全表数据，表结构保留）。
- 应用场景：删除单个元组、批量删除（如所有选课记录）、带子查询的删除（如计算机系学生的选课记录）。
- 依赖处理：删除操作需考虑参照完整性，避免破坏表间关联。

---

ch21-SQL中的空值（文档1、23）

核心内容：空值的产生、判断与运算规则。

- 空值定义：表示“不知道”“不存在”或“无意义”，并非空字符串或0。
- 产生场景：插入时未赋值、更新时设为NULL。
- 判断方法：使用IS NULL或IS NOT NULL（不可用“=”判断）。
- 约束限制：NOT NULL、UNIQUE约束列及码属性不可取空值。
- 运算规则：空值参与算术、比较运算结果为UNKNOWN，逻辑运算需遵循三值逻辑（TRUE/FALSE/UNKNOWN）。

---

ch22-视图（文档1、23）

核心内容：视图的定义、操作与应用价值。

- 基本特性：虚表，仅存储定义不存储数据，数据随基表动态变化。
- 定义与删除：CREATE VIEW（支持WITH CHECK OPTION约束更新条件），DROP VIEW（支持CASCADE级联删除派生视图）。
-

视图操作：查询与基本表一致（通过视图消解转换为基表查询），更新操作受限制（行列-子集视图通常可更新，含聚集函数、分组的视图不可更新）。

- 核心作用：简化用户操作、多角度看待数据、提供逻辑独立性、保护机密数据、清晰表达-复杂查询。

---

ch23-数据库安全性（文档2、24）

核心内容：数据库安全性的定义、风险与安全级别。

- 安全定义：保护数据库免受非合法使用导致的数据泄露、修改或破坏。
- 不安全因素：非授权存取、数据泄露、安全环境脆弱性。
- 安全级别：TCSEC/TDI分为D、C1、C2、B1、B2、B3、A1七级，CC标准的EAL1-EAL7-与TCSEC对应。
- 安全模型：系统级（用户身份鉴别）、数据库级（存取控制）、数据级（加密）的分层保-护机制。

---

ch24-存取控制（文档2、24）

核心内容：自主存取控制与强制存取控制的实现。

- 自主存取控制（DAC）：基于用户权限的控制，通过GRANT授权、REVOKE回收权限。
- 权限管理：支持数据库对象（模式、表、视图等）的精细化权限分配，可通过WITH GRANT OPTION传递授权。
- 数据库角色：将权限集合命名为角色，简化批量授权与回收（CREATE ROLE、GRANT-角色、REVOKE角色）。
- 强制存取控制（MAC）：基于主体（用户/进程）与客体（数据对象）的敏感度标记（密级-），仅当主体许可证级别满足客体密级要求时允许存取。

---

ch25-视图机制、审计、数据加密及其他（文档2、24）

核心内容：数据库安全的补充机制。

- 视图安全：通过视图隐藏敏感数据，间接实现权限控制。
- 审计功能：通过AUDIT语句记录用户操作（如ALTER、UPDATE），NOAUDIT取消审计-，审计日志用于安全监控。
- 数据加密：存储加密（透明/非透明）与传输加密（链路/端到端），防止数据存储或传输时-失密。
- 其他措施：推理控制（避免通过低密级数据推导高密级数据）、数据隐私保护（覆盖数据-全生命周期）。

---

ch26-数据库完整性（文档3）

核心内容：数据库完整性的定义、机制与核心规则。

- 完整性定义：包括数据正确性（符合现实语义）与相容性（不同表中同一对象数据逻辑一-致）。
- 与安全性的区别：完整性防范不合语义的数据，安全性防范非法用户与操作。
- 完整性机制：提供完整性约束定义、检查（INSERT/UPDATE/DELETE后或事务提交时）-、违约处理（拒绝执行/级联操作）。
- 核心规则：实体完整性、参照完整性、用户定义完整性。

---

ch27-实体完整性（文档3）

核心内容：实体完整性的定义与实现。

- 定义方式：CREATE TABLE中通过PRIMARY KEY定义，单属性码可列级或表级定义，-多属性码仅表级定义。
-

检查逻辑：插入或更新主码时，检查主码值唯一性（通过索引优化检查效率）与非空性，-违反则拒绝操作。

---

ch28-参照完整性（文档3）

核心内容：参照完整性的定义、检查与违约处理。

- 定义方式：通过FOREIGN KEY指定外码，REFERENCES指明参照表的主码。
- 检查场景：对参照表（如SC）和被参照表（如Student）的增删改操作可能破坏完整性，-需针对性检查。
- 违约处理：支持拒绝执行（默认）、级联操作（CASCADE，删除/修改被参照表元组时同-步处理参照表）、设置为空值（SET-NULL）。

---

ch29-用户定义的完整性（文档3）

核心内容：用户根据业务需求自定义的完整性约束。

- 属性级约束：包括非空（NOT NULL）、唯一（UNIQUE）、列值条件（CHECK短语，-如性别只能为“男”或“女”）。
- 元组级约束：通过CHECK短语定义不同属性间的逻辑关系（如男性姓名不以“Ms.”开头-）。
- 约束管理：通过CONSTRAINT子句命名约束，ALTER TABLE可删除或新增约束。

---

ch30-断言（文档3）

核心内容：复杂完整性约束的声明式定义。

- 定义与删除：CREATE ASSERTION指定约束条件，DROP ASSERTION删除断言。
- 应用场景：适用于涉及多个表或聚集操作的约束（如限制某课程选修人数不超过60）。

- 注意事项：复杂断言可能增加系统检测与维护开销。

---

ch31-触发器（文档3）

核心内容：事件驱动的复杂数据控制机制。

- 定义语法：CREATE TRIGGER指定触发事件（INSERT/DELETE/UPDATE）、时机（-BEFORE/AFTER）、表名、触发类型（行级/语句级）、触发条件与动作体。
- 核心特性：由服务器自动激活，支持NEW/OLD引用（行级触发器），可实现复杂检查与-操作。
- 应用实例：修改成绩时记录变动、统计插入学生数量、强制教授工资不低于4000元。
- 管理操作：DROP TRIGGER删除触发器。

---

ch32-关系模式及范式（文档4）

核心内容：关系模式的构成与规范化基础。

- 关系模式定义：五元组R(U,D,DOM,F)，简化为三元组R<U,F>（U为属性集，F为数据依-赖）。
- 第一范式（1NF）：最基本要求，属性值为不可分割的基本数据项。
- 1NF的问题：存在数据冗余、更新异常、插入异常、删除异常，根源为不合理的数据依赖-。
- 范式演进：1NF→2NF→3NF→BCNF→4NF→5NF，通过模式分解消除不合理数据依赖。

---

ch33-函数依赖与码（文档4）

核心内容：数据依赖的核心概念与码的定义。

- 函数依赖：X→Y表示X确定Y，包括平凡/非平凡、完全/部分、传递依赖等分类。

- 码的定义：候选码（K→U且无真子集X→U）、主码（选定的候选码）、主属性（含于候-选码的属性）、外码（非本关系码但为其他关系码）。
- 全码：整个属性组为候选码（如演奏者、作品、听众组成的关系）。

---

ch34-1NF，2NF，3NF（文档4）

核心内容：前三范式的定义与模式分解。

- 2NF：在1NF基础上，消除非主属性对候选码的部分依赖。
- 3NF：在2NF基础上，消除非主属性对候选码的传递依赖。
- 模式分解：将不满足高范式的关系模式分解为多个高范式关系，解决数据冗余与异常问题-（如S-L-C分解为SC和S-L）。

---

ch35-BCNF（文档4）

核心内容：修正的第三范式，更高的规范化程度。

- 定义：若X→Y且Y⊈X时X必含码，则R∈BCNF。
- 特性：消除主属性对候选码的部分依赖与传递依赖，在函数依赖范畴内实现彻底分解。
- 分解示例：非BCNF关系模式（如STJ）可分解为多个BCNF关系（ST、TJ）。

---

ch36-多值依赖与4NF（文档4）

核心内容：多值依赖与第四范式的定义。

- 多值依赖：X→→Y表示X确定一组Y值，与Z（U-X-Y）无关，具有对称性、传递性等特性-。
- 4NF：在BCNF基础上，消除非平凡且非函数依赖的多值依赖。

- 应用场景：解决多值依赖导致的数据冗余与操作复杂问题（如Teaching表分解为CT和-CB）。

---

ch37-数据库设计概述（文档5）

核心内容：数据库设计的目标、特点与基本步骤。

- 设计目标：构建优化的逻辑与物理结构，提供高效的数据存储与管理环境。
- 核心特点：结构设计与行为设计相结合，遵循“三分技术、七分管理、十二分基础数据”-原则。
- 设计方法：包括规范设计法（如New Orleans方法）、E-R模型法、3NF设计法等。
- 基本步骤：需求分析→概念结构设计→逻辑结构设计→物理结构设计→数据库实施→数据-库运行与维护。

---

ch38 需求分析

核心内容：需求分析是数据库设计的起点，核心是明确用户对数据存储、处理、安全及完整性的-要求。

- 核心任务：调查现实世界处理对象，了解原系统工作概况，明确信息要求（需存储的数据-）、处理要求（操作类型与性能）、安全性与完整性要求。
- 调查方法：包括跟班作业、开调查会、专人介绍、询问、填写调查表、查阅记录等。
- 分析工具：采用结构化分析（SA）方法，自顶向下、逐层分解系统。
- 核心成果：建立数据字典（记录数据项、数据结构、数据流、数据存储、处理过程）和需-求规格说明书，为后续设计提供依据。

---

ch39 概念模型和ER模型

核心内容：将用户需求抽象为概念模型，核心工具是E-R模型，重点解决实体、属性及联系的定义-与表示。

- 概念模型特点：真实反映现实世界、易于理解与修改、易于向各类数据模型转换。
- 实体间联系类型：包括两个实体型的1：1（一对一）、1：n（一对多）、m：n（多对多）-联系，以及多个实体型间的多元联系、单个实体型内的联系。
- E-R图表示方法：矩形表示实体型、椭圆形表示属性、菱形表示联系，联系可带属性；支持-ISA联系（父类-子类）、基数约束（实体参与联系的次数限制）、Part-of联系（实体间的-整体-部分关系）。
- 弱实体型：依赖其他实体存在的实体型，用双矩形表示，搭配双菱形的识别联系。

---

ch40 概念结构设计（1）

核心内容：采用自顶向下、自底向上、逐步扩张或混合策略，将需求转化为局部E-R图（分E-R图-）。

- 设计策略：常用混合策略，即自顶向下分析需求，自底向上设计局部概念结构。
- 设计步骤：先抽象数据并设计局部视图，再集成局部视图得到全局概念结构。
- 实体与属性划分原则：属性需是不可分的数据项，且不能与其他实体有联系；若需进一步-描述或存在联系，应升级为实体。
- 局部E-R图设计示例：以销售管理子系统为例，明确顾客、订单、订单细则、产品等实体及-属性，梳理实体间的1：n或m：n联系。

---

ch41 概念结构设计（2）

核心内容：将多个分E-R图合并为全局E-R图，消除冲突与冗余，形成基本E-R图。

- 合并步骤：先合并分E-R图生成初步E-R图（解决冲突），再修改重构消除冗余（生成基本-E-R图）。
-

冲突类型及解决：属性冲突（统一数据类型、取值范围）、命名冲突（协商统一名称）、-结构冲突（调整实体抽象、属性集或联系类型）。

- 冗余消除：通过分析数据依赖（如导出数据、导出联系）消除不必要的冗余，也可结合规-范化理论优化，但若需提升效率可保留必要冗余。

- 集成案例：以工厂管理信息系统为例，合并物资管理、销售管理、劳动人事管理等子系统-的E-R图，统一实体命名，消除冗余联系。

---

ch42 逻辑结构设计

核心内容：将基本E-R图转换为具体数据库管理系统支持的数据模型（重点为关系模型），并优化-模型结构。

- 转换原则：1. 实体型：一个实体型转换为一个关系模式，属性为实体属性，码为实体码。2. 联系：1：1联系可独立成表或合并至任意一端；1：n联系可独立成表或合并至n端；-m：n联系及多元联系需独立成表，属性含各实体码及联系自身属性。

- 数据模型优化：基于规范化理论，消除部分函数依赖、传递函数依赖等，确定关系模式范-式；结合应用需求，通过水平分解（按元组分组）或垂直分解（按属性分组）优化性能。

- 用户子模式设计：定义视图，优化属性别名、保障数据安全（不同用户对应不同视图）、-简化复杂查询。

---

ch43 物理结构设计

核心内容：为逻辑数据模型选择最优物理存储结构与存取方法，适配硬件环境与应用需求。

- 设计步骤：确定存取方法与存储结构，评价时间、空间效率，若不满足需求则迭代调整。

- 存取方法选择：B+树索引适用于查询条件、聚集函数、连接条件中的属性；Hash索引适用-于等值查询；聚簇存储适用于需集中存放相同属性值元组的场景（如同一部门学生）。

- 存储结构确定：分离易变与稳定数据、高频与低频存取数据；合理分配磁盘存储（如日志-与数据库对象分盘存放）；调整系统配置参数（如缓冲区大小、物理块大小）。

- 评价标准：权衡存取时间、存储空间利用率、维护代价，选择折中方案。

ch44 数据库的实施和维护

核心内容：完成数据库构建、数据载入、试运行，并进行长期运行维护，保障系统稳定高效。

- 数据库实施：1. 数据载入：通过人工或计算机辅助方式将数据入库，可分期输入小批量数据用于调试。
  2. 应用程序调试：与数据设计并行进行，调试后联合数据库开展功能与性能测试。
  3. 试运行：测试应用功能是否达标，测量性能指标，若不满足设计目标则返回物理或逻辑设计阶段调整。

- 数据库维护（由DBA负责）：1. 转储与恢复：制定定期备份计划，故障后利用后备副本与日志文件恢复数据。
  2. 安全性与完整性控制：根据应用变化调整用户权限与完整性约束。
  3. 性能优化：监测性能参数，分析后调整配置或结构。
  4. 重组织与重构造：重组织（优化物理存储，不改变逻辑结构）、重构造（调整模式与内模式，适配需求变化）。

---

ch45 面向驱动的数据库编程

核心内容：介绍不同编程语言的数据库编程方式，以JDBC为例详解编程流程。

- 主流编程方案：Java用JDBC，微软技术栈用ODBC/OLE DB，Python用DB-API+专用模块（如PyMySQL）。

- JDBC编程步骤：1. 加载数据库驱动（如`Class.forName("com.mysql.jdbc.Driver")`）。
  2. 建立数据库连接（通过`DriverManager.getConnection()`指定URL、用户名、密码）。
  3. 创建Statement对象，发送SQL语句。
  4. 获取ResultSet结果集，读取数据。
  5. 释放资源（关闭结果集、Statement、连接）。

---

ch46 过程化SQL

核心内容：SQL的过程化扩展，支持变量定义、流程控制与异常处理，以块结构为基本单位。

- 块结构组成：1. 定义部分（DECLARE）：声明变量、常量、游标、异常，作用域仅限当-前块。
  2. 执行部分（BEGIN-END）：包含SQL语句与过程化控制语句。
  3. 异常处理部分（EXCEPTION）：捕获并处理执行过程中的异常。

- 核心语法：1. 变量与常量：变量定义需指定数据类型，常量用`CONSTANT`修饰且必须-初始化。
  2. 流程控制：条件控制（IF-THEN-ELSE）、循环控制（LOOP、WHILE-LOOP、-FOR-LOOP）。
  3. 异常处理：捕获系统或自定义异常，执行相应处理逻辑。

---

ch47 存储过程和函数

核心内容：预编译并存储在数据库中的过程化SQL块，支持重复调用，提升执行效率与代码复用性-。

- 存储过程：1. 特点：运行效率高（预编译）、减少网络通信量（客户端仅传调用指令）、-便于统一实施企业规则。
  2. 操作语法：`CREATE OR REPLACE PROCEDURE`创建，`CALL/PERFORM`执行，-`ALTER PROCEDURE`修改，`DROP PROCEDURE`删除；支持输入、输出、输入/输出-参数。
  3. 示例：转账存储过程（检查转出账户余额、验证转入账户、更新余额并提交事务，异常-时回滚）。

- 函数：1. 语法：`CREATE OR REPLACE FUNCTION`创建，需指定返回类型，通过`-CALL/SELECT`调用。
  2. 与存储过程区别：函数需返回值，可嵌入SQL语句中使用，存储过程无强制返回要求，-需单独调用。

---

ch48 事务

核心内容：定义事务的概念、特性与操作，是数据库恢复与并发控制的基本单位。

- 事务定义：用户指定的不可分割的数据库操作序列，要么全做要么全不做。
- 事务操作：`BEGIN TRANSACTION`启动，`COMMIT`提交（永久保存更新），`ROLLBACK`回滚（撤销所有操作）。
- ACID特性：1. 原子性（Atomicity）：操作不可分割，要么全执行要么全不执行。
  2. 一致性（Consistency）：执行后数据库从一个一致性状态转换到另一个一致性状态。
  3. 隔离性（Isolation）：并发执行的事务互不干扰，内部操作对其他事务隔离。
  4. 持续性（Durability）：事务提交后，修改永久有效，不受后续故障影响。
- 破坏因素：并发事务操作交叉、事务被强行终止，需通过恢复与并发控制机制保障ACID。

---

ch49 故障和数据库恢复

核心内容：分析数据库故障类型，明确恢复目标（恢复至一致状态），阐述恢复的基本原理与策略。

- 故障类型及影响：1. 事务内部故障（非预期中断，如溢出、死锁）：导致事务未完成，数据库可能不一致，需撤销（UNDO）事务。
  2. 系统故障（软故障，如CPU故障、断电）：所有运行事务终止，内存缓冲区数据丢失，需UNDO未完成事务、重做（REDO）已提交事务。
  3. 介质故障（硬故障，如磁盘损坏）：破坏数据库存储，需结合后备副本与日志文件恢复。
  4. 计算机病毒：人为破坏，需通过恢复技术修复数据库。
- 恢复原理：利用冗余数据（后备副本、日志文件）重建受损数据。
- 恢复策略：针对不同故障，组合UNDO（撤销错误操作）、REDO（重复正确操作）、重装后备副本等方式。

---

ch50 数据转储和日志文件

核心内容：介绍两种核心冗余数据形式（数据转储、日志文件），详解其类型、作用与应用场景。

- 数据转储：1. 定义：定期将数据库复制到存储介质，生成后备副本。
  2. 类型：静态转储（无运行事务时进行，副本一致但可用性低）与动态转储（并发事务，-需搭配日志文件）；海量转储（全量复制）与增量转储（仅复制更新数据）。
  3. 恢复应用：重装后备副本后，重新运行转储后的所有更新事务，恢复至故障前状态。

- 日志文件：1. 定义：记录事务对数据库的更新操作，是恢复的关键依据。
  2. 格式：以记录为单位（含事务开始、更新操作、事务提交/回滚标记）或以数据块为单-位。
  3. 用途：支持事务故障与系统故障恢复，协助介质故障恢复（搭配后备副本）。

ch51 日志文件的登记与恢复应用

核心内容：详解日志文件的登记规则、恢复原理及具体应用场景，是数据库恢复的核心依据。

- 登记规则：遵循“先写日志，后写数据库”原则，确保日志记录与数据更新的一致性。

- 记录内容：以记录为单位时，包含事务标识、操作类型、操作对象、旧值与新值；以数据-块为单位时，记录数据块的修改前后状态。

- 恢复应用：事务故障时，反向扫描日志执行UNDO（撤销未提交操作）；系统故障时，先-UNDO未提交事务，再REDO已提交但未写入数据库的事务；介质故障时，结合后备副本-与日志文件，先重装副本，再通过日志REDO后续事务。

---

ch52 恢复策略的选择与优化

核心内容：根据故障类型、数据库规模等因素选择合适的恢复策略，平衡恢复效率与系统开销。

- 策略分类：1. 完全恢复：依赖完整后备副本+全程日志，可恢复至故障前任意时刻，适用-于关键业务数据库。
  2. 不完全恢复：仅恢复至指定时间点，适用于日志文件损坏或部分事务无需恢复的场景。

- 优化方向：1. 日志文件分区存储，减少读写冲突。
  2. 增量转储与日志结合，降低转储开销。
  3. 并行恢复技术，提升大规模数据库恢复速度。

---

ch53 并发控制概述

核心内容：介绍并发操作的必要性、潜在问题及并发控制的核心目标。

- 并发操作优势：提高数据库资源利用率，支持多用户同时访问。

- 潜在问题（并发一致性问题）：1. 丢失修改：两个事务同时修改同一数据，后提交的事务-覆盖先提交的修改。
  2. 不可重复读：同一事务内多次读取同一数据，结果不一致。
  3. 脏读：一个事务读取了另一个事务未提交的修改数据，后该事务回滚导致数据无效。

- 控制目标：保证并发事务的隔离性，避免一致性问题，同时最大化并发度。

---

ch54 封锁机制

核心内容：封锁是并发控制的核心技术，详解封锁类型、粒度及封锁协议。

- 封锁类型：1. 共享锁（S锁）：允许其他事务加S锁，禁止加排他锁，适用于读操作。
  2. 排他锁（X锁）：禁止其他事务加任何锁，适用于写操作。

- 封锁粒度：从数据项、元组、关系到数据库，粒度越小并发度越高，开销越大；反之则并-发度低，开销小。

- 封锁协议：1. 一级封锁协议：写操作加X锁，事务结束释放，解决丢失修改问题。
  2. 二级封锁协议：一级基础上，读操作加S锁，读完释放，解决丢失修改和脏读问题。
  3. 三级封锁协议：一级基础上，读操作加S锁，事务结束释放，解决所有并发一致性问题。

---

ch55 活锁与死锁

核心内容：分析活锁与死锁的产生原因、检测方法及解决策略。

- 活锁：事务长期等待资源，因优先级问题始终无法获得锁，可通过“先来先服务”调度策-略避免。

- 死锁：两个或多个事务互相等待对方释放锁，导致事务永久阻塞。1. 产生条件：互斥条件-、请求与保持条件、不剥夺条件、循环等待条件。

2. 检测方法：超时法（超过指定时间判定死锁）、等待图法（检测图中是否存在回路）。
3. 解决策略：死锁预防（破坏产生条件）、死锁检测与解除（终止部分事务释放锁）。

---

ch56 并发调度的可串行性

核心内容：判断并发调度是否正确的核心标准，详解可串行化调度的定义与判定方法。

- 可串行化调度：多个事务的并发调度结果与某一串行调度结果一致，即满足隔离性要求。
- 判定方法：1. 冲突可串行化：通过交换非冲突操作，将调度转化为串行调度，冲突操作指-“同一数据+不同事务+一写一读/两写”。
  2. 视图可串行化：更宽松的判定标准，确保事务对数据的读写视图一致，适用于非冲突操-作无法交换的场景。
- 调度器：数据库管理系统通过调度器生成可串行化调度，保证并发操作的正确性。

---

ch57 两段锁协议与多粒度封锁

核心内容：两段锁协议是实现可串行化的关键，多粒度封锁优化封锁粒度选择。

- 两段锁协议：事务分为加锁段（仅申请锁，不释放锁）和解锁段（仅释放锁，不申请锁）-，遵循该协议的调度一定是冲突可串行化的。
- 多粒度封锁：结合不同封锁粒度，通过意向锁（意向共享锁IS、意向排他锁IX、共享意向-排他锁SIX）协调不同粒度的锁，减少封锁冲突，提升并发效率。

---

ch58 数据库镜像（跳过章节补充）

核心内容：数据库镜像作为介质故障的冗余手段，详解其实现与应用。

- 定义：将数据库副本实时同步到另一个磁盘或服务器，形成镜像数据库。
- 作用：介质故障时，可快速切换至镜像数据库，无需重装后备副本和日志恢复，提升可用-性。

- 实现：同步镜像（主库更新后立即同步至镜像库）、异步镜像（主库更新后批量同步），-平衡一致性与性能。

---

ch59 分布式数据库概述（跳过章节补充）

核心内容：介绍分布式数据库的定义、特点与体系结构。

- 定义：数据分散存储在多个地理位置不同的节点，节点间通过网络连接，逻辑上为统一数-据库。
- 特点：数据分布性、逻辑统一性、节点自治性、数据冗余性（提升可用性）。
- 体系结构：分为全局外层（用户视图）、全局概念层（全局模式）、局部概念层（局部模-式）、局部内层（存储结构），通过分布式透明性简化用户操作。

---

ch60 分布式数据库的分布透明性

核心内容：分布透明性是分布式数据库的核心优势，详解其分级与实现。

- 透明性分级：1. 位置透明性：用户无需知道数据存储节点，访问时无需指定位置。
  2. 复制透明性：用户无需知道数据是否复制，系统自动处理副本同步与访问。
  3. 分片透明性：用户无需知道数据如何分片（水平/垂直分片），系统自动重组数据。
- 实现基础：分布式数据字典，记录数据分布、复制、分片等信息，供系统查询与调度。

---

ch61 分布式查询处理（跳过章节补充）

核心内容：分布式环境下的查询优化，目标是减少数据传输量、提升查询效率。

- 查询处理步骤：查询分解（将全局查询分解为局部查询）、查询优化（选择最优数据传输-与连接策略）、查询执行（协调各节点执行局部查询并汇总结果）。
- 优化策略：1. 数据本地化：优先在数据所在节点执行过滤、聚集等操作，减少传输数据量-。
  2. 连接操作优化：选择传输数据量小的连接方式（如半连接），避免全量数据传输。

ch62 分布式事务管理（跳过章节补充）

核心内容：分布式事务需保证跨节点的ACID特性，详解其协议与实现。

- 分布式事务特点：事务操作涉及多个节点，需协调各节点的提交与回滚。

- 核心协议：1. 两阶段提交协议（2PC）：分为准备阶段（协调者询问各参与者是否就绪）-和提交阶段（所有参与者就绪则提交，否则回滚），保证原子性。
  2. 三阶段提交协议（3PC）：在2PC基础上增加预提交阶段，减少死锁概率，提升容错性-。

---

ch63 数据仓库概述（跳过章节补充）

核心内容：数据仓库是面向分析的结构化数据存储，详解其定义、特点与架构。

- 定义：面向主题、集成、稳定、随时间变化的数据集合，用于支持管理决策分析。

- 核心特点：1. 面向主题：围绕业务主题（如销售、客户）组织数据，而非应用。
  2. 数据集成：整合多个数据源（业务数据库、文件等），消除数据冗余与不一致。
  3. 数据稳定：数据一旦载入，通常不修改，仅追加新数据。
  4. 时间序列：存储历史数据，支持趋势分析。

- 架构：分为数据源、数据存储与管理（ETL过程：抽取、转换、加载）、数据访问与分析-（OLAP工具、报表工具）三层。

---

ch64 数据仓库的数据模型

核心内容：数据仓库常用数据模型，适配分析场景的需求。

- 星型模型：以事实表为核心，周围围绕多个维度表（如时间、产品、客户），结构简单，-查询效率高，适用于常规分析。

- 雪花模型：星型模型的扩展，维度表可进一步分层（如产品维度表下分产品类别表），减-少数据冗余，适用于复杂维度分析。

- 事实星座模型：多个事实表共享维度表，适用于多主题联合分析（如同时分析销售与库存-）。

---

ch65 OLAP技术

核心内容：联机分析处理（OLAP）是数据仓库的核心分析工具，详解其定义与操作。

- 定义：基于数据仓库，支持用户多维度、交互式分析数据，快速响应复杂查询。

- 核心操作：1. 切片与切块：在某一/多个维度上固定取值，分析剩余维度的数据。
  2. 钻取：从汇总数据向下钻取至明细数据（钻取），或从明细数据向上汇总（钻取）。
  3. 旋转：改变维度展示顺序，从不同角度分析数据。

- 实现方式：多维数据库OLAP（MOLAP，预计算汇总数据，查询快）、关系型数据库-OLAP（ROLAP，基于关系表动态计算，灵活度高）。

---

ch66 数据挖掘基础（跳过章节补充）

核心内容：从海量数据中挖掘隐藏的模式与知识，详解其定义、任务与常用算法。

- 定义：基于人工智能、统计学、数据库技术，自动识别数据中的规律，为决策提供支持。

- 核心任务：1. 关联分析：发现数据间的关联规则（如“购买面包的客户80%会购买牛奶”-）。
  2. 分类与预测：构建模型（如决策树、神经网络），对数据分类或预测未来趋势。
  3. 聚类分析：将相似数据自动分组（如客户分群），无需预先定义类别。

- 常用算法：Apriori算法（关联分析）、K-Means算法（聚类）、C4.5算法（决策树分类-）。

---

ch67 数据库性能优化概述

核心内容：数据库性能优化的目标与整体思路，覆盖从设计到运行的全流程。

- 优化目标：提升查询响应速度、降低系统资源占用、支持更多并发用户。
- 优化维度：1. schema优化：合理设计表结构（如范式优化、字段类型选择）、索引优化-
  。
  2. 查询优化：优化SQL语句（避免全表扫描、减少嵌套查询）、利用执行计划调整查询逻-
  辑。
  3. 硬件与配置优化：升级硬件（CPU、内存、磁盘）、调整数据库参数（缓冲区大小、连-
  接数）。
  4. 运行维护优化：定期数据重组织、清理冗余数据、监控性能瓶颈。

---

ch68 索引优化

核心内容：索引是提升查询性能的关键，详解索引设计、选择与维护。

- 索引设计原则：1. 优先为查询条件（WHERE）、连接条件（JOIN）、排序字段（-
  ORDER BY）建立索引。
  2. 避免过度索引：索引会降低插入/更新/删除效率，仅为高频查询字段建立索引。
  3. 合理选择索引类型：B+树索引适用于范围查询，Hash索引适用于等值查询，全文索引-
  适用于文本检索。
- 索引维护：定期重建碎片化索引、删除无用索引、监控索引使用效率。

---

ch69 SQL语句优化

核心内容：通过优化SQL语句逻辑与语法，提升执行效率。

- 优化技巧：1. 避免全表扫描：不用SELECT *、合理使用WHERE条件、为关键字段建立-索引。
  2. 优化连接查询：优先使用内连接、小表驱动大表、避免多表复杂嵌套。
  3. 减少函数使用：避免在WHERE条件中对字段使用函数（如WHERE SUBSTR(S-no,1,4)='2020'），会导致索引失效。
  4. 合理使用分页：避免一次性查询大量数据，使用LIMIT等关键字分页。

---

ch70 数据库配置优化

核心内容：调整数据库系统参数，适配硬件环境与应用需求。

- 关键参数优化：1. 内存参数：增大缓冲区大小（如InnoDB缓冲池），减少磁盘I/O。
  2. 连接参数：调整最大连接数，避免连接耗尽；设置连接超时时间，释放闲置连接。
  3. 日志参数：调整日志刷盘策略（如innodb_flush_log_at_trx_commit），平衡性能与-数据安全性。
  4. 存储参数：优化磁盘I/O（如使用SSD、RAID阵列）、调整数据文件存储路径。

---

ch71 数据库监控与性能诊断

核心内容：通过监控工具与指标，定位性能瓶颈，为优化提供依据。

- 监控指标：1. 资源指标：CPU使用率、内存使用率、磁盘I/O吞吐量、网络带宽。
  2. 数据库指标：查询响应时间、并发连接数、锁等待时间、事务提交成功率。

- 常用工具：数据库自带工具（如MySQL的SHOW PROCESSLIST、SQL Server的SQL Server Profiler）、第三方工具（如Prometheus+Grafana、Navicat监控）。

- 诊断流程：收集监控数据→分析异常指标→定位瓶颈（如慢查询、锁冲突、资源不足）→-制定优化方案。

---

ch72 云数据库概述（跳过章节补充）

核心内容：云数据库是部署在云端的数据库服务，详解其特点、优势与常见类型。

- 核心特点：按需付费、弹性伸缩、高可用性、免运维（厂商负责部署、备份、升级）。
- 优势：降低企业硬件与运维成本、快速扩容应对业务增长、多区域部署提升可用性。
- 常见类型：1. 关系型云数据库（如阿里云RDS、AWS RDS）：兼容传统关系型数据库（-MySQL、Oracle），支持SQL。
  2. 非关系型云数据库（如阿里云MongoDB、AWS DynamoDB）：适用于非结构化/半结-构化数据，支持高并发写入。

---

ch73 数据库安全进阶（补充章节）

核心内容：在基础安全机制上，详解高级安全防护手段。

- 高级安全措施：1. 数据脱敏：对敏感数据（手机号、身份证号）进行部分隐藏（如138***-*5678），避免数据泄露。
  2. 数据库审计：记录所有数据库操作，包括登录、查询、修改，支持安全审计与故障追溯-。
  3. 入侵检测与防御：通过工具监测异常操作（如暴力破解、批量导出数据），及时阻断攻-击。
  4. 隐私计算：在不泄露原始数据的前提下进行数据分析（如联邦学习），保护数据隐私。

---

ch74 数据库技术发展趋势（补充章节）

核心内容：总结数据库技术的前沿方向与未来趋势。

- 主要趋势：1. 多模数据库：支持关系型、非关系型、时序型等多种数据类型，适配复杂应-用场景。
  2. 云原生数据库：专为云环境设计，支持弹性伸缩、分布式架构，优化云端性能。
  3. 智能化：融合AI技术，实现自动索引优化、查询优化、故障自愈。
  4. 边缘数据库：部署在边缘设备（如物联网终端），减少数据传输延迟，支持离线操作。