数据模型，SQL语言等接口（40/100）

DBMS实现技术

数据库安全与一致性

分布式数据库系统

数据仓库与数据挖掘

实验：

数据库语言，数据库管理系统，查询优化，并发事物管理

引言，**数据模型**，数据库系统接口**SQL**，DBMS内部实现

数据库安全，**数据库设计**（一事一地）

数据库：大的，有机融合的数据集合

DBMS：管理维护数据库的软件包

数据：描述现实世界的符号

数据模型：描述数据的概念与定义（关系模型。。。）

数据模式schema：用给定数据模型描述具体数据库

物理模式，逻辑模式，外模式（虚表）

逻辑独立性：逻辑表变化，通过更改**映射**关系使外模式不变

物理独立性：底层存储结构变化，逻辑与外模式均不受影响

数据库发展：

种类：树型、网状结构、关系型、面向对象、XML等

结构：集中式、并行式、分布/联邦式、CS/BS架构、移动式、虚拟化

应用领域：联机事物处理OLTP、工程、演绎、多媒体、时/空间轨迹、知识管理

数据库系统：数据库+**DBMS**(提供高级数据接口)+数据库管理员+应用程序

数据模型

层次模型：使用树型结构表达现实的层次化世界（记录与域）

PCR：现实中实体一对多关系（一条指针）

包含多个PCR关系、表达一对多类型、每种类型有且只有一个双亲

虚记录：引入虚记录指针，解决多对多问题

虚记录需指向实记录、虚记录不能为根

网状模型：使用“系“来表达现实网状世界（记录与数据项）

一个记录类型可以在多个set中做任意主/属记录，多个set组成复杂的网状结构

LINK记录：对需自领导的数据(1:N)，定义LINK记录替身（1:1）

属记录使用LINK替身连接属记录

关系模型：将现实中实体与其间联系用表、关系表达

关系、属性、元组tuple（表，列，行）

主键：唯一决定其它所有属性而无真子集的键

外键：存在于另一张表且为其主键的属性为此表外键（引用完整性：外键不能为空）

域完整性：表中属性取值在值域内

实体完整性：表中有非空唯一主键

关系代数：

完备操作集：选择、投影（去除属性列）、笛卡尔乘积、集合差、集合并/交

选择：选出元组值、结果仍为表

投影：选出所需列、注意删除重复元组（DBSM一般不做）

并交差：要求schema数目、类型相同，满足并兼容

笛卡尔cross-product：属性两两拼接、属性冲突-重命名

连接：笛卡尔乘积基础（非实际操作）上进行选择、运算效率可以更高

自然连接：对公共属性做等值连接

外部连接：保留找不到匹配项的结果、剩余属性填null（左、右、全）

外并：保留找不到匹配项的所有结果、剩余属性填null

除法：遇到“所有”情况时、寻找A中与B中每个y都有关系的x（A中有<x,y>）、使用否定值否定运算

关系演算：一阶逻辑谓词子集（元组TRC/域DRC关系）

非过程化、使bool公式为真的结果

不安全查询：结果数目无限

ER图：方形（实体）、椭圆（属性）、菱形（联系）

扩展ER模型：弱实体：依赖于其它实体存在（教工家属）、普遍/特殊化、聚集、范畴（构建混杂集合）

SQL语言

查询语言非编程语言（无图灵完备、复杂运算、支持简单/高效大规模数据查询）

SQL：DDL（操作属性）、QL（查询）、DML（增删改）、DCL（权限检查）

WITH 表表达式 SELECT [DISTINCT] 目标列表 FROM 相关表 WHERE 元组选择条件

GROUP BY 属性集合 HAVING 组筛选 ORDER BY 列排序

分组：目标列表、组筛选中属性必须在属性集合中

LIKE：“?“ 表示1个字符，\*表示0到多个字符

且操作：表自链接、集合交、嵌套查询

UNIQUE：判断值是否唯一 <ANY、<ALL（未必都支持）

INTERSECT：集合交、更换查询目标语义问题

表达“所有“除法：否定之否定（集合差、EXISTS）

聚集：COUNT（放在SELECT处）、SUM、MAX、MIN、EVERY

EXCEPT/UNION（ALL）：集合差/并（出现重复元组）

NULL：IS NULL而不用=，空值表示值未知

扩展功能：

类型转换：CAST（表达式/NULL AS 数据类型）

CASE WHEN 条件 THEN 运算 ELSE 相关处理 END AS 名称（避免/0等）

子查询：标量子查询Scalar（结果单值）、

表表达式：多级查询嵌套、WITH（SELECT之前）只定义计算一次公共表

递归查询：多级、查询结果UNION

DML：插、删、改

视图：普通视图（保存）定义、临时视图（WITH、RECURSIVE）

SQL编程：程序接受SQL、数据交换、集合/对象为单位、数据类型

嵌入式源代码、编程API、类库

动态SQL：程序中拼接、占位符输入USING替换

存储过程：方便用户可直接提供参数、扩展DBMS功能、定义高效可维护存储过程

DBMS

核心、运行结构、底层存储、查询优化、事物管理（恢复、并发控制）

核心：词法文法分析、权限检查、语义分析与查询（SQL）、存储管理

访问原语：物理层提供的API、调用操作系统文件管理

多进程：进程间通信、socket

多线程：启动单一监听进程、开启线程并建立通信管道

访问类型：大多记录（>15% 绝大部分物理块）、特定元组、部分记录、范围查询、更新

存储方式：堆文件、hash文件、堆与B+树索引、簇集+索引、动态hash、栅格(条件同时)

RAW disk簇集：申请连续空间、设计DBMS自身的文件系统、物理上顺序存放

B+树：叶子节点双向链表连接、建立堆的索引（维护树需要代价）

查询优化：下压一元操作、寻找合并公共子表达式

重写查询语句：代数、操作优化，投影滤除无用属性、选择连接顺序与算法（利用物理结构与索引 连接可交换、结合 选择可拆分/合并 选择与投影可交换 选择、投影可下压性）

连接：建立内存缓冲区，在内存中执行匹配，减少IO次数（对于嵌套循环）、实现外排序使用归并扫描、利用索引寻找匹配元组（值选择性强重复少）、RS处于相同域散列储存

恢复：冗余必须redundancy、检查发现所有故障

方法：周期备份dumping（完整备份+增量转储）、备份+日志+重演（前像B.I+后像A.I）

最近一致状态：Half down事务：根据B.I undo，UnWrotten事务：使用A.I redo

**事务：原子性A**tomic**、保持一致C**onsistency**、隔离性I**solation**、持久性D**urability

恢复保障：提交、活动列表（TID list）、log（前后像链表）

Commit：事务的A.I在commit前必须写入非易失存储器

Log Ahead：B.I在A.I写入数据库前需先写入log

Check point检查点：系统定时检查恢复

**三种更新方式**：直接写DB（先B.I写入log）、提交后写DB（先A.I写入log）、提交修改并发（A.I B.I写入log，后台进程发现硬盘空闲搬移log修改数据库）

故障**恢复**：检查active、commit列表，按照更新方式恢复（undo ac、redo co&ac、undo ac, redo co&ac 删除相应TID）

并发控制：丢失更新（写写冲突 禁止）、读脏数据、不可重复读（隔离性）

可串行化Serializable：事务可串行运行认为结果正确（n!种结果）

封锁法Lock：X锁（读写排他，相容矩阵分配锁）、封锁表为系统临界资源

两阶段事务：所有锁申请在释放前、申请锁后读（守规矩well-form）、**任何两阶段+well-form事务组可串行2PL**（反证法证明）+解锁在事务结束EOF（可恢复 回滚无影响）

严格两端加锁：2PL+wellformed+所有锁释放在EOT（多粒度封锁时有漏洞）

SUX锁协议：U（更新读取申请，写时升级为X锁）、X（读写排他）、S（读标识锁）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | XL | S | U | X |
| XL | Y | Y | Y | Y |
| S | Y | Y | Y | N |
| U | Y | Y | N | N |
| X | Y | N | N | N |

死锁：等待时间限制、优先级允许等待、构建等待图（周期性、加入新事务时检查）

解决死锁：等待死亡（记录等待时间）、击伤等待（杀死年轻事务）

Security & Integrity安全与一致性

视图、访问控制（普通，资源特权，DBA）、识别授权（角色）、数据加密、审计（记录动作）

CREATE TRIGGER 监控器名

BEFORE/AFTER INSERT ON 监控表名

REFERENCING NEW TABLE 新表名

FOR EACH STATRMENTS (操作名)

完整性约束：关系中每一条合法数据应该满足的条件（固有、隐含、显式/动态） REFERENCING

CHECK：定义表插删改规则、由DBMS而非应用程序检查数据完整性

Triggers触发器（主动数据库）：某种变化发生时自启动的过程（连锁触发）

提供影子表，便于共享数据库维护

Database Design

数据依赖：**函数依赖**FD(多对一)、多值依赖MVD(一对多)、联系依赖JD(无损连接分解）

无损连接分解：投影的子关系join后可以还原

**一事一地**（主键不能为空）

**1NF范式**：每个属性应为原子类型（不允许表中套表）

**2NF范式**：1NF属性对主键无部分函数依赖

**3NF范式**：2NF属性间无传递依赖（信息冗余，一致性难保证，插入、删除、更新异常）

**BCNF范式**：决定子全为主键

**4NF范式**：消除多值依赖

**5NF范式**：消除连接依赖