实体集

$r_{DCS} = (r_{DWS}) \cup [(null, \dots, null)] \times (S - \Pi_S(r_{WS}))$

$r_{DCS} = (r_{WS}) \cup (r - \Pi_{R(WS)}(r_{WS})) \times [(null, \dots, null)] \cup [(null, \dots, null)] \times (S - \Pi_S(r_{WS}))$

Chap1 Introduction

database application

↕ API (ODBC, JDBC)

DBMS ↕ OS

database

A	B	C	D	E
a	a	a	a	a
a	a	a	b	b
a	a	b	b	b
b	b	b	b	b
b	b	b	b	b

r

a	b
a	a
b	b

s

A	B	C
a	a	γ
γ	a	γ

$r \div s =$

1-1 purpose of database systems

① data redundancy and inconsistency 数据冗余 不一致

② data isolation 数据孤立 数据孤岛

③ difficulty in accessing data 存取数据困难

④ integrity problem 完整性问题

+完整性约束条件不显式声明

⑤ atomicity problem 原子性问题

⑥ concurrent access anomalies 并发访问异常

⑦ security problems 安全性问题

authentication 认证

privilege 权限

audit 审计

特性: ①数据持久性 ②数据独立性 ③数据完整性 ④多用户并发性 ⑤故障恢复 ⑥安全性

1-2 层次模型

view level 视图层 — 应用程序数据抽象性

logical level 逻辑层 内容结构 logical schema

physical level 物理层 physical schema 物理数据存储方式

1-3 概念 schema - logical structure of the database

relationship 模式间关系

Chap2 The relational model 关系模型

2-1 概念 relation 关系 tuple - 行, attribute - 列, relation instance - 一组特定的行 属性 域 不可分的(原子性)

relation schema ⇒ $R = (A_1, A_2, \dots, A_n)$ attribute

domain 域: 每个attribute可取的值. null 属于任一domain

2-2 键

superkey 超键 — 唯一标识一个元组的属性集

↓ minimal 的 superkey 任一真子集都不是superkey

candidate key 候选键 — 一个 superkey 的子集 primary key 主键

2-3 关系运算 select predicate 关系运算的结果 不同的元组 ↓ 可用集合运算 ∪∩− 不同的元组 ↓ 用 union key 的并集

① 6种基本操作 (join)

select σ 横向选择 | project Π 纵向选择 | union ∪ 并集 相同的元组

set difference − 差 相减 | cartesian product × 笛卡尔积 | rename ρ 改名 重命名

合并 find the largest salary

$\Pi_{instructor.salary}(\sigma_{instructor.salary < d.salary}(instructor \times \rho_d(\Pi_{salary}(instructor))))$

② additional operator 在基本操作上的, 表达力更高或更方便的

∩ set intersection 交 r−(r−s)

⋈ natural join 自然连接 (属性名相同的属性)

$\Pi_{A,B,C,D,E}(r \bowtie s) = \Pi_{A,B,C,D,E}(\sigma_{r.B=s.B \wedge r.D=s.D}(r \times s))$

← assignment 赋值

r⟕s, r⟖s, r⟗s outer join 外连接 未匹配的元组会填充null

÷ division operator 除法. 适用于属性集合 B 的元组 A

* theta join 连接 $r \bowtie_\theta s = \sigma_\theta(r \times s)$

semi join 半连接 $r \ltimes_\theta s$ 保留为 M 关系 r 的元组

② extended operation

3-1 Generalized Operation $\Pi_{F_1, F_2, \dots, F_n}(E)$ — arithmetic functions

3-2 Aggregation 忽略null

function 聚合函数 avg. max. min. sum. count count(*) 只计数不同值 count(distinct col-name)

operation $G_1, G_2, \dots, G_n\ G_{F_1(A_1), F_2(A_2), \dots, F_n(A_n)}(E)$ 分组属性 aggregate function attribute name 可用 as 重命名

2-4 Matrix - 不重要

chap3 introduction to SQL 定义 基本SQL查询

3-1 SQL操作语句 没有不兼容的

DDL - 定义语言 Create. alter. drop

DML - 操作语言 select. insert. delete. update

DCL - 控制语言 Grant. revoke.

3-2 模式设计

from → where → group by → having → select → distinct → order by 聚合函数不能直接用在where语句中. 除非作为函数的属性 或子查询

3-3 null (未知unknown) 与null运算 null return null where ... is null/not null

OR: (uk or T) = T AND: (T and uk) = uk NOT: not uk = uk

(uk or F) = uk (F and uk) = F where 语句中, uk作 F 处理

(uk or uk) = uk (uk and uk) = uk

3-4. DDL

① domain max为n *P位, 后d位 ≥ digits

char(n). varchar(n). int. smallint. numeric(p, d). float(n)

date. time. timestamp. interval

② create table 完整性约束

create table r (A1 D1, A2 D2, ... An Dn, ... attribute name → Data Type

(integrity-constraint1, → ① not null to A1D1 后面

... ② primary key (A1, ... An)

integrity-constraint k)) ③ foreign key (An, ... An) references r

④ default x 设置默认值

③ 对已有的属性处理

on delete cascade | set null | restrict | set default

on update cascade | set null | restrict | set default

级联删改 不允许删改

④ drop / alter 表

- drop table 删除表及其内容 - delete from 删除内容

- alter table r add AD 属性 - alter table r drop A 删属性

3-5. DML 主键

① basic query - select distinct dept-name from instructor

join 操作时使用 using 语句 在其使用的属性

e.g. select distinct student. id from (student natural join takes)

join course using (course_id) where student. dept-name = ...

- Rename: select .. from .. where name like '%dar%' escape '\'

- order by name desc / asc

- limit offset. rel-count 限制元组

- set operations 集合运算, 加 all 全部

union all - m+n

intersect all - min(m, n)

except all - max(m − n, 0)

② nested query 嵌套查询

- in, not in 一个标量 - exists, not exists

- some. all, scalar 子query 必须只返回一个结果, 不用 some. all.

- unique 检查是否有非空结果 → 查询不能加 distinct

③ with clause 构造一个只在查询期间的临时表. 可多次引用

e.g. with max_budget (value) as (select max(budget) from department)

select dept.name from department. max_budget where department. budget = max_budget. value

④ 增删改 delete from ... (where ...) | update ... set ... = ... where ... | insert into ... values (...) select ...

chap4 Intermediate SQL 内连接

① Join e.g. course inner join prereq on course. id = prereq. id

left outer join course natural left outer join prereq

full outer join course

② Course natural left outer join prereq

4-1 types

4-2 conditions natural 连接时会合并列 — on <predicate>

全值不会 using (A1, A2, ... An)

4-2 user-defined 自定义 create type dollars as numeric(12, 2) final

domains create domain degree level char(10)

blob 大数据类型 constraint degree level-test

clob 字符大对象 check (value in ('A', 'B', 'C'));

4-3 integrity constraints

① not null ② primary key ③ unique ④ check P. where P is a predicate

⑤ foreign key 外键 references 表名 ⑥ assertion - create assertion assertion-name check predicate

4-4 views create view v as <query expression>

物理层面的视图 每次调用view都会重新计算一次查询

4-5 indexes create index ... on ...

4-6 transactions - ACID properties

Atomicity 原子性 Consistency - 一致性 Isolation 隔离性 Durability 持久性

4-7 authorization 授权

操作权限. select. insert. update. delete

定义权限. index. resource. alteration. drop

① grant <privilege list> on <relation name or view name> to <user list> from ... on ...

② revoke ...

③ create role role-name; grant role-name to user

with grant option 用户可任意授权 revoke.. cascade 级联删除 restrict 只对直接删除

Chap6 Entity-Relationship Model

6-1 基本概念

E entyset

R relationship set 关系集合

E1 A1 A2 A3 A4 A5 属性

attributes: simple (A1), composite (A2), multivalued (A3), derived (A4)

primary key

属性的属性 属性的attribute

total / partial 部分 / 全部

role

cardinality limits

1:1, 1:n, many-to-many

E1 total / not total

total (disjoint) generalization

disjoint generalization 不相交

overlapping 可重叠

role isolation

E1 E2 A1 A2 A3 A4 A5

多元(>2)关系 n个实体 n个属性 n个基数. 保证

overlapping. 同一E可以属于多个实体. 属性不同

6-2 Relationship 的 primary key.

- 对一 - 唯一实体的key. - 对多 - 多的可识别的key; 多对多 - 两实体key组合

6-3 弱实体集

identifying entity set 标识性实体集 identifying relationship 标识性联系 discriminator 区别符 / partial key

weak entity set.

6-4 E-R 图的设计

① 实体集 - 现实世界中可区别的事物 ② 属性 - 简单 / 复合 / 单值 / 多值 / 派生

③ 关系 - 一对一 / 一对多 / 多对多 / 多元关系

④ 属性 - composite - 复合, multivalued - 多值属性, derived - 派生属性

⑤ 角色 - role

6-5 ER 图的简化

① 弱实体集 - 弱实体集可以通过其标识性实体集的标识性联系集的primary key 唯一确定

② 多元关系 - 多元关系可以通过二元关系来表示

③ 属性 - composite - 复合, multivalued - 多值属性, derived - 派生属性

Chap7 Relational Database Design

7-1. INF: 满足原子性

缺点: ① 信息重复 ② 更新异常 ③ 插入异常

7-1-2 函数依赖 (x) (y)

平凡(trivial) 函数依赖: $\alpha \to \beta$, $\beta \subseteq \alpha$ 都成立

是一般函数依赖都成立的特例, 它表示任意可以决定自身属性 ①

7-3 对码的理解

定义: 能决定其后可以唯一确定的属性集, 并决定所有的所有关系的属性集, 只是并不一定是最小的 key.

7-4 闭包 closure

7-4-1 函数依赖的闭包 函数依赖集合 F 的闭包 F+, 包含所有可以由 F 推导出的函数依赖

推导规则: ① 自反律 $\beta \subseteq \alpha \to \alpha \to \beta$ ② 增广律 $\alpha \to \beta \to \gamma\alpha \to \gamma\beta$ (augmentation)

(Armstrong) (reflexivity)

③ 传递律 (transitivity) $\alpha \to \beta$ and $\beta \to \gamma \to \alpha \to \gamma$

易证 或更强的规则: ① 合并 (Union) $\alpha \to \beta$ and $\alpha \to \gamma \to \alpha \to \beta\gamma$

② 分解 (decomposition) $\alpha \to \beta\gamma \to \alpha \to \beta$ and $\alpha \to \gamma$

③ 伪传递律 (pseudo transitivity) $\alpha \to \beta$ and $\gamma\beta \to \delta \to \alpha\gamma \to \delta$

计算: 直接法 - 直接, 间接法 ① 对 F: 每项 F ② 对 F1, F2 分别 F+

7-4-2 属性闭包 给出 F 以及 α 的属性集合的闭包 α+, 看闭包 α+ 是否包含 α 的 key 可用

计算: 直接法 - 直接, result: = α

while (changes to result) do for each β → γ in F do

if β ⊆ result then result := result ∪ γ

7-5 正则覆盖 ① 在 F 中不存在 α → β 使得 α 或 β 中有一个属性可以从 F 中删去 ② 对 F 中任意 α → β 和 α → γ, 使用 α → βγ 替换

7-5-1 计算 ① 合并 α → β1 和 α → β2 为 α → β1β2 ② 删除 α → β 中的 β 属性 if β ⊆ (α − β)+ ③ 删除 F 中的 α → β if β ⊆ (F − (α → β))+

7-6 依赖保持 dependency preservation 原来关系R上的函数

一个函数依赖, 都可以在拆解后的子表上检验出来保留

7-7 BCNF - 规范化范围 但不一定能保证 ① "good form"

7-7-1 定义 对于F中所有的 α → β 如果满足以下任一条件, ① α → β trivial

② α 是超键 ③ α 是某个模式R的一个候选键 (α+ 包含 R) 则 α → R

7-7-2 计算 对每条违反 BCNF 的 α → β:

result := (result − R) ∪ (R − β) ∪ (α, β) ① ∪ (R − (β − α))

更新规则:

7-8 3NF 一定保证无损连接和依赖保存

7-8-1 定义 在7-7-1基础上加③ 每个 β − α 中的属性 A 都包含在一个R的候选键中

7-8-2 计算 ① 遍历F, 每个 α → β 创建 R; ② 若目前没有任一关系模式包含候选键, 并保证连接的一个 R; ③ 去冗余.

7-9 4NF

7-9-1. 多值依赖 $Y \to\to Z \Leftrightarrow \langle y_1, z_1, w_1 \rangle \in R$ and $\langle y_1, z_2, w_2 \rangle \in R$ then $\langle y_1, z_1, w_2 \rangle \in R$ and $\langle y_1, z_2, w_1 \rangle \in R$.

7-9-2. 4NF 定义 ① $\alpha \to\to \beta$ is trivial ($\beta \subseteq \alpha$ or $\alpha \cup \beta = R$) 或

② α is a superkey for R.

Chap15 Data Storage Structures

12.8

12-1 storage hierarchy

volatile storage 易失存储

primary storage 一级存储: cache ~1ns; main memory ~10^2ns; flash memory ~10^5ns

secondary storage 二级存储 / online storage: magnetic disk ~10^7ns

tertiary storage 三级存储 / offline storage / near storage: optical disk; magnetic tapes

NVM 兼容了内存和存储的特性

non-volatile storage 非易失存储

12-2 磁盘 disk 不连续 tracks many → 扇区 sectors (最小单位)

Access time (访问时间) = Seek time (寻道时间) + Rotation latency

① 顺序访问 sequential access: 只有读第一个块时需要寻道

② 随机访问 random access. 每一次访问都需要一次寻道

IOPS - 每秒IO操作数 MTTF - 平均故障时间

wear-leveling 磨损均衡

	DRAM	NVM	SSD	HDD
Read Latency	1x	2-4x	50x	10^5x
Write Latency	1x	2-8x	500x	10^5x
Persistence 数据持久性	N	Y	Y	Y
数据可恢复性		Y	N	N
Byte-addressable	Y	Y	N	N
Endurance		N	N	Y

12-3 File organization

数据库 → 文件 → 记录 → 不同 fields

block 数据块 最基本单位

① 定长 - free list 存储空闲块

② 变长 - slotted page 存储数据 offset + size (bytes) 不定长记录 空值 null bit map → 0 非空 1 空

Block Header | Free Space | Records

... Free ...

End of Free Space

- Heap: 空间充足时

- Sequential: 插入或删除, 指针维护, 实现重构.

- 面向列的存储

优点: ① 只访问需要的属性列, 减少IO开销 ② 提高CPU缓存性能 ③ 数据压缩 ④ 数据聚合 ⑤ 可以在压缩状态下处理数据 Vector Processing; 缺点: ① 行重构开销大 ② 需要维护更新的索引 ③ 删除所需时间更长

12-2 buffer manager

基本操作: 读, buffer. 没有在disk读到buffer. 满了替换. 在访问 pin. 写到磁盘 pin count. 被钉住 ← nested loop

替换策略: ① LRU 最近最少使用. 在循环访问不佳

② MRU 最近最常使用 (latest upmost). 适合排序的 merge 时所用

Chap14 Index

14-1 聚集索引

页面内, 点查高效. 一个表只有一个

① 主键 (primary index) 聚集索引 ② 唯一索引 (secondary index) 非聚集索引 ③ 顺序索引 (ordered index) 顺序扫描

② 稠密索引 - 每 search key 都要建立索引 dense index

稀疏索引 sparse index 非 - 一定时间段内

③ Hash 散列索引 (指针) / 顺序索引 ④ 多级索引

14-2 B+树索引

性质: $\lceil \frac{n}{2} \rceil \sim n$ children of leaves node (fanout是总结点数) 3级树就索引, 非叶子节点 $\lceil \frac{n}{2} \rceil \sim n-1$ values of leaves - root is leaf. $0 \sim n-1$ 个值

P: 指向具有search key k值对应记录. P: 指向下一node

计算: 1. 高度估计 $\lceil \log_N(M) \rceil \le h \le \lceil \log_{\lceil \frac{n}{2} \rceil}(M) \rceil$

2. 插入时的最坏cost: $O(\log_{\lceil \frac{n}{2} \rceil} M)$ $\lceil \log_{\lceil \frac{n}{2} \rceil}(\frac{M}{2}) \rceil + 1$

average node occupancy: ① random insert $\frac{2}{3}$ ② in sorted order insert $\frac{1}{2}$

3. Oi not unique → a composite unique key (Oi, Ap)

search for Oi=v → search for (v, -∞) to (v, +∞)

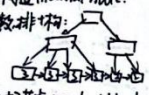
4. 线性存储记录: 一条记录长度为L, block大小为B. 一个block中最多有 $M = \lfloor \frac{B}{L} \rfloor$ 条记录, 共N条记录, 则需 $\lceil \frac{N}{M} \rceil$ 个block.

B+树索引block: 假设block大小是B, 指针大小是a, 被索引的属性值大小是b. fanout(n) $= \lfloor \frac{B-a}{b+a} \rfloor + 1$ primary key.

5. 其他则根据fanout的上下限计算. 注意, root. node. leaf的不同

* B+-Tree File Organization 叶子节点不再放索引页. 就减少底效建立B+索引: 从底向上先排(如按排序的顺序排序). 按顺序插入指针. 按空间自下向上构造 internal node. 并维护相指针. 按空间自下向上构造 internal node.

6. 代价计算 e.g. n=4: ① 插16个数. 排序: 一次seek后全部写出去: 1 seek + 9 block transfers



② 再插9个数 ⇒ 遍历刚用的叶子节点. 1 seek + b blocks

在9个排好序的和合并后写出 1 seek + B blocks

(1+2+3+3)

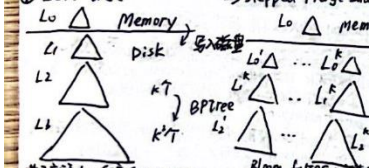
* 内存内的索引: key idea: ① use large nodesize to optimize disk access ② b tree structure loads a node using a M > |bs| ⇒ 不用 recursive partition. (|br+bs|)

如果所有的两部分都能放进内存, n log n 是不用的 nlogn

tree with small nodesize. instead of using an array. to optimize cache access

14-3 Write Optimized Indices to minimize page writes for flash-optimized search trees.

① LSM-tree → Stepped Merge Index



满了就和L1合并向下merge. 优点: 顺序I/O. 对写友好

② Buffer Tree. 中间节点留在作缓存. 满了再往下传. I/O↓. 但 more random I/O. and fanout↓

③ Bitmap Indices. 有用和不用的标. 对低-value. queries on multiple attributes 直接与/或; packed into words use elements storing count of 1s

Chap15 Query Processing

15-1 基本原理: 记录存储→查询优化→执行

15-2 cost的计算 $B \cdot t_T + S \cdot t_S$ 依赖于寻道时间大小

15-3. 选择查询算法

A1 线性搜索 worst: $t_S + b_r * t_T$ avg: $t_S + (\frac{b_r}{2}) * t_T$ ← 在文件中的块数量

A2 B+树主索引 (hi+1)*(ts+tT) 能直接定位到记录的块数

A3 B+树主索引 hi*(ts+tT)+ts+tT*b

A4 B+树辅助索引 (hi+1)*(ts+tT)

A5 B+树辅助索引 hi*(ts+tT)+ts+tT*b

A6 B+树辅助索引 (hi+n+m)*(tT+ts) n pointers may be stored in m blocks

A7 B+树辅助索引 (hi+n+m)*(tT+ts)

15-4. External Sort-Merge

记录在br个block. 内存M个block; 一次读写 b_b 个block:

block transfer: $br(2\lceil \log_{\lfloor M/b_b \rfloor}(b_r/M) \rceil + 1)$

seeks: $2\lceil br/M \rceil + \lceil br/b_b \rceil (2\lceil \log_{\lfloor M/b_b \rfloor}(\lceil b_r/M \rceil) \rceil - 1)$

15-5. Join r-外关系. s-内关系. nr表r记录条数. br表r的块数

① nested loop join

block-transfer: $n_r * b_s + b_r$ seeks: $n_r + b_r$

② block nested loop join 以关系作外循环的关系记录数好

worst: block transfer: $b_r * b_s + b_r$ seeks: $2 * b_r$

best: block transfer: $b_r + b_s$ seeks: 2 内存足够一次情况

M个内存块时: block transfer: $\lceil b_r/(M-2) \rceil * b_s + b_r$

seeks: $2\lceil b_r/(M-2) \rceil$

③ Indexed Nested-Loop Join $b_r(t_T + t_S) + n_r * c$ ← 查询代价

② Merge join 要求关系有序 seeks: 2

block transfer: $b_r + b_s$ seeks: $\lceil b_r/b_b \rceil + \lceil b_s/b_b \rceil$

* 合理分配 X_r 和 X_s: block transfers: $b_r + b_s$

seeks: $\lceil b_r/X_r \rceil + \lceil b_s/X_s \rceil$ $(X_r + X_s = M) \Rightarrow \lceil \frac{br}{M/2} \rceil + \lceil \frac{bs}{M/2} \rceil$

③ Hash-Join 如果所有的两部分都能放进内存, n log n 是不用的 nlogn

block transfer: $3(b_r + b_s) + 4n_h$ seeks: $2(\lceil b_r/b_b \rceil + \lceil b_s/b_b \rceil) + 2n_h$

Hash 桶数量

"ignoring cost of writing partially filled blocks" 就不计 usually ignored

③-2 含recursive partition

block transfers: $2(b_r + b_s)\lceil \log_{\lfloor M/b_b \rfloor - 1}(b_s/M) \rceil + b_r + b_s$

seeks: $2(\lceil b_r/b_b \rceil + \lceil b_s/b_b \rceil) \cdot \lceil \log_{\lfloor M/b_b \rfloor - 1}(b_s/M) \rceil$

Chap16 Query Optimization

16-1 优化目标

① 逻辑优化: 选择代价较低的代数表达式 (代数优化选择, 投影) ② 物理层面: 确定最

选择不同算法

16-2 基价估算原则

① selection $\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$; $\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$ ← 相关属性相同

$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{L_n}(E))\ldots)) = \Pi_{L_1}(E)$; $\sigma_\theta(E_1 \times E_2) = E_1 \bowtie_\theta E_2$;

$\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$ 先选出更小的结果的数

② join $E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$; $(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$

$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$ 选择代价小的先连

$\sigma_{\theta_0}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_0}(E_1)) \bowtie_\theta E_2$; $\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_\theta (\sigma_{\theta_2}(E_2))$

③ projection $\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \Pi_{L_1}(E_1) \bowtie_\theta (\Pi_{L_2}(E_2))$ 0只包含L1L2

$\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \Pi_{L_1 \cup L_3}(\Pi_{L_1 \cup L_2}(E_1)) \bowtie_\theta (\Pi_{L_1 \cup L_4}(E_2))$ 先选出更小的结果

④ set operation. 交换律 $\sigma_\theta(E_1 - E_2) = \sigma_\theta(E_1) - \sigma_\theta(E_2)$; $\Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2))$

① other $\sigma_\theta(\sigma_\theta(E)) \equiv \sigma_{\theta \wedge \theta}(E)$, $E_1 \cap E_2 = E_2 \cap E_1$, $E_1 \cap E_2 = E_1 \bowtie_\theta E_2$

$\sigma_\theta(E_1 \bowtie_\theta E_2) \equiv (\sigma_{\theta_1}(E_1)) \bowtie_\theta E_2$. $\sigma_{\theta_1}(E_1 \bowtie_\theta E_2) \equiv \sigma_{\theta_1}(E_1) \bowtie_\theta E_2$

如果E有null ⇒ θ_1 为false

16-3 代价估算 n_r-r的记录数. b_r-r的块数. l_r-r的记录size. f_r-一个块可以放的记录. $b_r = \lceil \frac{n_r}{f_r} \rceil$. $V(A,r)$ 在r中A属性的distinct value number of distinct values that appear in r for attribute A

① select

默认: $S_i = n_r / V(A,r)$; $\sigma_{A \le v}(r)$ $c = \frac{n_r}{V(A,r)}$ else $c = n_r \cdot \frac{v - \min(A,r)}{\max(A,r) - \min(A,r)}$

前提: 相等属性 $\sigma_{\theta_1 \vee \ldots \vee \theta_n}(r) \Rightarrow n_r * (1 - (1 - \frac{s_1}{n_r}) * (1 - \frac{s_2}{n_r}) \ldots (1 - \frac{s_n}{n_r}))$. $\sigma_{\neg\theta}(r) = n_r - \sigma_\theta(r)$

$\sigma_{\theta_1 \vee \theta_2 \vee \ldots \vee \theta_n}(r) = n_r * (1 - (1 - \frac{s_1}{n_r}) * (1 - \frac{s_2}{n_r}) \ldots (1 - \frac{s_n}{n_r}))$

② join $r \bowtie s \Rightarrow n_r \cdot n_s$ tuples, each tuple $s_r + s_s$ bytes

③ $R \cap S = \emptyset$. 笛卡尔积 ② R∩S is a key for R. s的一个tuple最多join r的一个tuple ③ R∩S is a foreign key in s referencing R. $n_r n_s = n_s$

④ $R \cap S = \{A\}$. A not a key ⇒ $\min\{n_r * \frac{n_s}{V(A,s)}, n_s * \frac{n_r}{V(A,r)}\}$

⑤ other. $T(r) = V(A_1, r) \cdot V(A_2, r) \cdot \ldots \cdot V(A_n, r)$ $q_{E(r)} = V(A_1, r) \cdot V(A_2, r) \cdot \ldots \cdot V(A_n, r) \cdot n_r$ $n_{r \times s} = n_r n_s$

$r - s = n_r$

④ $V(A, r \bowtie s) = \min(V(A1, r) * V(A2 - A1, s), V(A, A2, r) * V(A, A1, s))$

代价估算: 算法: op 通过结构. 时间复杂度 $O(3^n)$. 空间 $O(2^n)$

left-deep join tree (左边可以是中间结果, 右边只能是基表). 时间 $O(n \cdot 2^n)$

Chap17 Transactions

17-1 ACID Properties Atomicity (原子性) Consistency (一致性) Isolation (隔离性) Durability (持久性) 崩溃 log恢复数据不丢失

17-2 transaction state

active → partially committed → committed; active → failed → aborted (restart/kill)

17-3 并发执行调度问题

① lost update 丢失修改 ② dirty read 读脏数据 ③ unrepeatable read 不可重复读 ④ phantom problem 幻读问题

17-4 serializability

① 冲突可串行化: 冲突指令: read(Q) write(Q) write(Q) read(Q) 冲突 read read 不冲突 ② 视图可串行化: S and S' are view equivalent if ① both Ti reads the initial value of Q ② 可视读取到 read (同一事务内) ③ 最终写自同一事务的 write

① 冲突可串行化: read(Q) write(Q) write(Q) read(Q) 冲突 read read 不冲突

② 视图可串行化: S and S' are view equivalent if ① both Ti reads the initial value of Q ② 可视读取到 read (同一事务内) ③ 最终写自同一事务的 write

① both Ti process final write. 只要可串行即是视图可串行. 反过来不一定

17-5. recoverable schedule (事务读取一事务写入数据, 必须先提交后才提交)

17-6. 级联回滚 ① serializable 可串行化 ② recoverable 保证不读脏数据

③ repeated read 不重复读问题 ④ read committed 保证不读脏数据

read→validation→write

⑤ read uncommitted

17-7. 并发控制方法 ① lock ② 时间戳 ③ Validation-Based protocols

Chap18 Concurrency Control

18-1. 2PL (growing phase → shrinking phase) 可以申请lock points 未定义 即是 只能放锁之前的定义

Growing Lock (upgrade) Time 在一段时间内没有新的加锁请求的时候是安全的

只能向所有锁加锁 (STRICT 2PL) S只可读取数据, X表明是写数据 ← 只有读写数据

强制所有锁加锁 (rigorous 2PL) 所有锁在结束时才释放 ← 只有读写数据

两阶段锁 协议是可串行化的必要条件

18-2 lock table

rid 的 lock 表 rid 指向 waiting 的 trans

wait-die scheme older may wait for younger. Younger never wait for older. 被杀的事务以原时戳重启

wound-wait scheme older may wounds (forces rollback) of younger transaction instead of waiting for it. Younger transactions may wait for older ones.

18-3 Graph-Based Protocols 如果以前数据项访问偏序关系 tree protocol 只能加一次锁. 每个结点只能加锁一次. 已解锁的结点不能再加锁. 满足 E 不再加锁. dead lock-free. 可能会读脏数据. 不可恢复

18-4 Multiple Granularity

fine granularity - 细粒度 - lower in tree. 小

coarse granularity - 粗粒度 - higher in tree. 大

database → area → file (table) → record

	IS	IX	S	SIX	X
IS	T	T	T	T	F
IX	T	T	F	F	F
S	T	F	T	F	F
SIX	T	F	F	F	F
X	F	F	F	F	F

下面加锁 上面不加锁

下面加锁 上面可加锁

18-5 插入幻读

加锁: 加锁 X锁. 插入后加锁

不加锁: 不加锁 X锁. 插入后不加锁

只加锁 S锁

不加锁 S锁

不加锁 IX锁

不加锁 IS锁

不加锁 X锁

不加锁 SIX锁

不加锁 X锁

不加锁 X锁

不加锁 X