

- Coursework -- 70% :
- In-Class Exercises -- 5%
- Homework assignment -- 15%
- Mid-term Quiz -- 20%
- Project -- 30%
- in the form of team-project (each team consists of 4~6 students)
- Students must be in the same Session
- Final examination -- 30%

# Lecture Outline

Week	Date	Topic	Lecturer
1	5 Sep 2024 12:30pm-15:30pm	Introduction to DB Systems	Dr. Zhou
2	12 Sep 2024	ER, Relational Data Model and SQL (I) Tutorial 1	Dr. Zhou
3	19 Sep 2024	ER, Relational Data Model and SQL (II) Tutorial 2	Dr. Zhou
4	26 Sep 2024	Relational Algebra Lab 1	Dr. Zhou
5	3 Oct 2023	Integrity Constraints (ICs) & Normal Forms (I) Lab 2 Deadline for forming Project Groups	Dr. Zhou
6	10 Oct 2023	Integrity Constraints (ICs) & Normal Forms (II) Lab 3 Project Specification Released	Dr. Zhou

# Lecture Outline

Week	Date	Course Activities	Lecturer
7	17 Oct 2024	File Organization and Indexing (I) Midterm Tutorial 3 Assignment Released	Dr. Hua
8	24 Oct 2024	File Organization and Indexing (II) Tutorial 4	Dr. Hua
9	31 Oct 2024	Query Optimization (I) Tutorial 5 Project First Deadline	Dr. Hua
10	7 Nov 2024	Query Optimization (II) Lab 4	Dr. Hua
11	14 Nov 2024	Transactions and Concurrency Control (I) Lab 5 Assignment Deadline	Dr. Hua
12	21 Nov 2024	Transactions and Concurrency Control (II) Tutorial 6	Dr. Hua
13	28 Nov 2024	Database Recovery; Review & Trend Tutorial 7 Project Final Deadline	Dr. Hua

Database: non-redundant, persistent (fixed?), 逻辑相关, 结构化

Database Management System: 创建 储存 更新 访问

## Data Abstraction 简化 隐藏细节

Low -> High

- Physical/Internal 实际咋弄的
- Logical/Conceptual 存的哪些东西
- External/View: partial schema / 移除不想管. 具体 group 的  
Logical Data Independence: 不改变概念层 e 不造成应用程序重写 [- - - Different use right (which part can see by the user, which can not)]  
Physical Data Independence [- - - DB can run different divides (HDD, SSD)]

## Data Model

管理一系列的工具

例子: Entity-Relationship (ER) Model / Relation Model

Data Schema (每一层都是 schema) 逻辑: 结构 / 物理: 格式

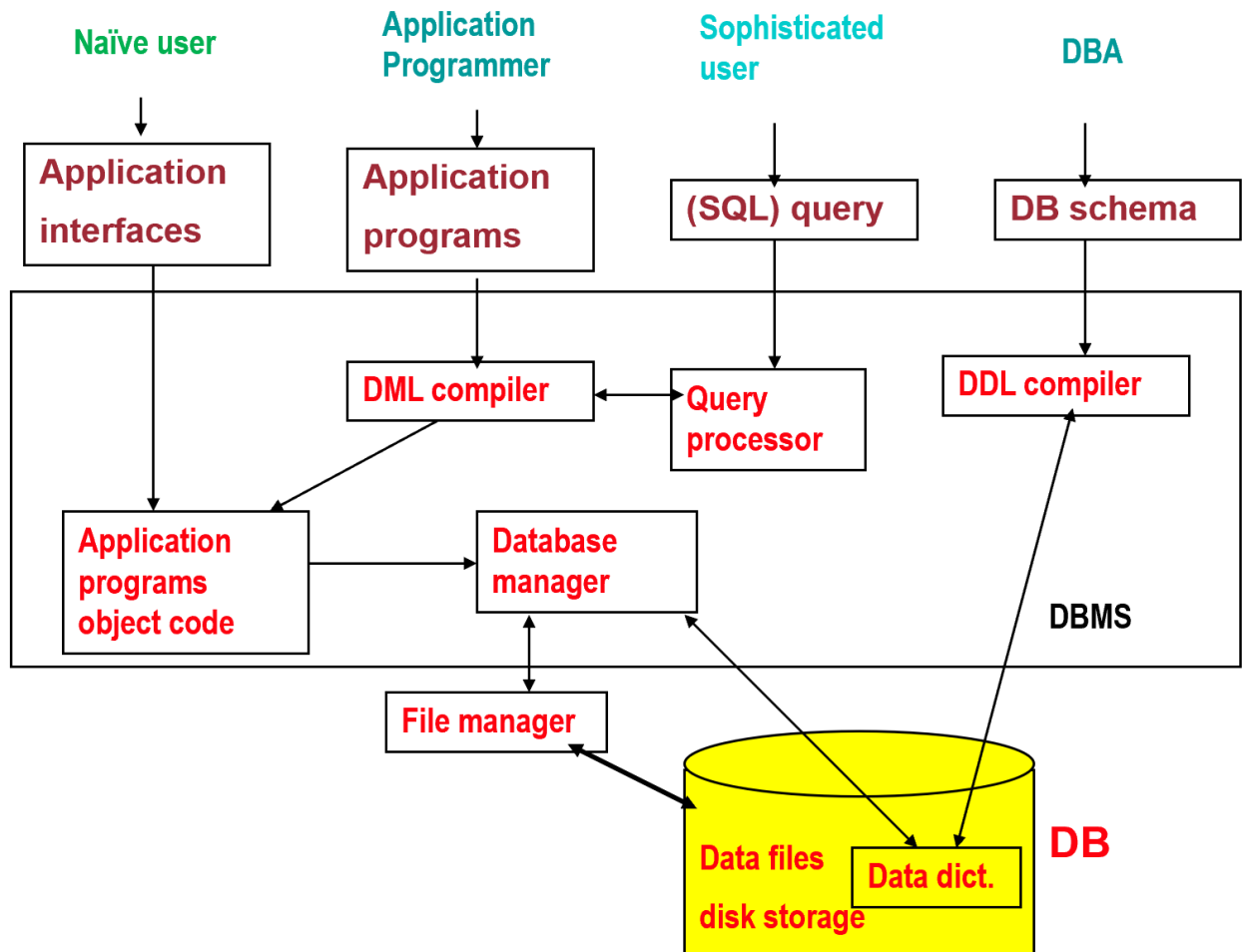
DB Lang

DDL: Data Definition (define schema) / 【CREATE, ALTER, DROP, TRUNCATE】

DML: Manipulation (修改数据) 【INSERT, DELETE, UPDATE, SELECT,】

- Procedural What and how 需要什么数据怎么得到

- Declarative: 不知道在哪



### Database Administrator (DBA):

User of DB:

- Application Programmers
- 交互用户
- 笨笨用户 Naive Users

File Manager

DB Manager:

- conceptual → physical

## Lecture 1

**Cardinality Ratio:** 基数比

ERD:

- Multivalued: 多个圈圈\*
- Derived Attribute: 可以从别的转化曲线 虚线
- 复合: 可以分解

# Lecture 2

Relation Data Model, Record-Based

Table, rows (tuples), columns (attributes)

Key: 从上到下包含关系越来越小


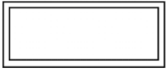









- Super key: 可以唯一确定
- Candidate Key: 极小的可以唯一确定 (多个集合)
- Primary Key: 选的候选 key (通常最小的选择)

Participation 限制:

- Partial 单线
- Total: 双线 (每个至少对应一个那个)

Weak:

- 双线

Symbol	Meaning
	ENTITY TYPE
	WEAK ENTITY TYPE
	RELATIONSHIP TYPE
	IDENTIFYING RELATIONSHIP TYPE
	ATTRIBUTE
	KEY ATTRIBUTE
	MULTIVALUED ATTRIBUTE
	COMPOSITE ATTRIBUTE
	DERIVED ATTRIBUTE <b>DoB -&gt; Age</b>
	TOTAL PARTICIPATION OF <b>E<sub>2</sub></b> IN R
	CARDINALITY RATIO 1:N FOR E <sub>1</sub> :E <sub>2</sub> IN R

## 性质

- 元组没顺序
- 属性有顺序

- value 原子不可分, null 不知道

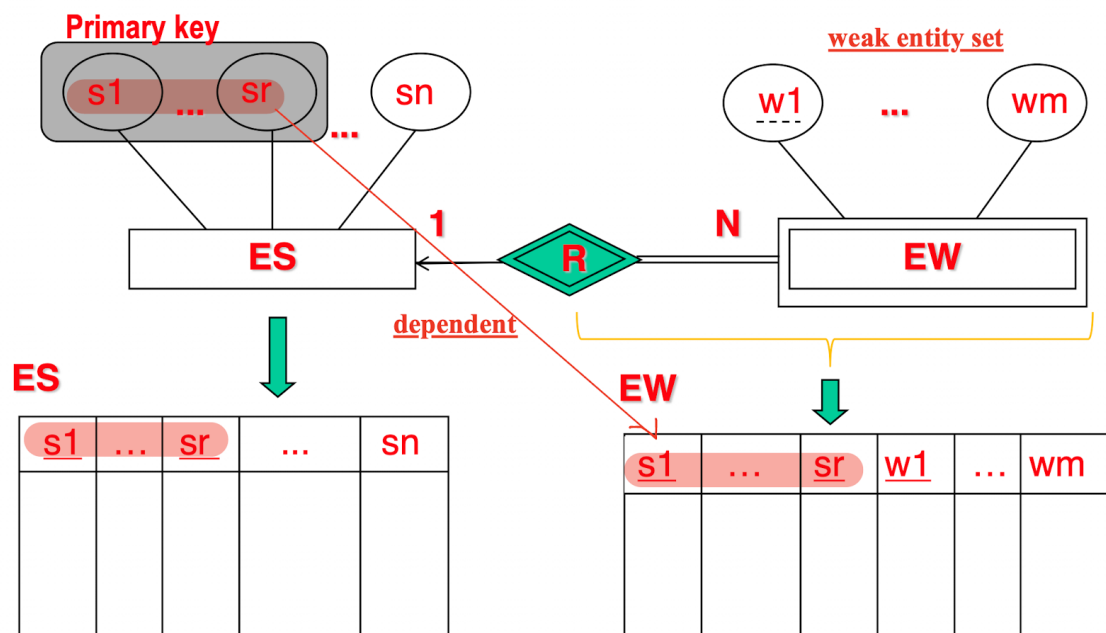
NULL 可不可以为空 (主不行)

外部 key:

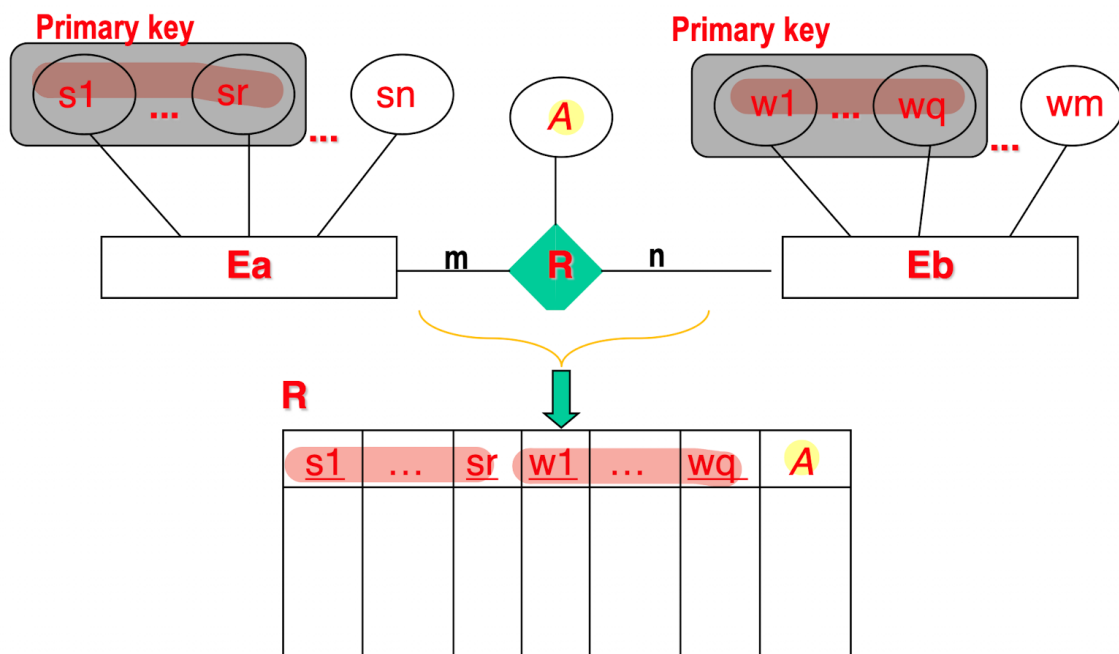
- 可以间接
- 直接写箭头就完了
- 可以复合, 一个季节指过去
- 数量:  $M:N \times 2$  【这个 2 对应这两边的 primary key 的 sum】, 剩下  $\times 1$  【refer 那边的 primary key 数量】

ER 转 RelationModel

- 先放实体, 不要虚线, 忽略双, 不要派生属性
- 弱实体, partial 和 primary 都要下划线, 然后箭头指一下



- multi value (双) 新建一下 (主 [refer], 自己) 都underline
- 1:1 外键放在完全参与的地方? 也可以指代一下创意谷指向主
- $n \rightarrow 1$
- $n:m$  加一个 R 指向俩的 (扩展到  $>2$  个度同样创一个新的指向四周)



- weak 不用 n - 1 的 foreign，指的还是 weak 直接指(变量都要用)就行了  
recursive: 自环

## Lecture 3 SQL

； 结尾； 关键词全大写

<>和!=等价

- 创建：属性和值域

CHAR(n)  
VARCHAR(n)  
DATE: YYYY-MM-DD  
DATETIME: YYYY-MM-DD HH:MM:SS  
DOUBLE(n, d)

小数位数

- NOT NULL

PRIMARY KEY(xxx),  
FOREIGN KEY(xxx) references A(B)

完整性约束： Integrity Constraints

# Data Definition (Language)

```
CREATE TABLE <tablename> (... , ... ,... .)

DROP TABLE <table name>;

INSERT INTO <table name> VALUES (... , .. ,... );

ALTER TABLE <table name>
ADD / MODIFY (varname, type);

TRUNCATE TABLE <table name>;
/* 删除数据不删除 keep table */
ALTER TABLE <table name>

ADD <attribute name> <domain> |

ADD CONSTRAINT <constraint> |

DROP COLUMN <attribute name> |

DROP CONSTRAINT <constraint> |

ALTER COLUMN <attribute name> <new domain> ;
```

## 描述

```
DESCRIBE <table name>
```

## Select

```
SELECT <attribute1>, <attribute3>

FROM <table name>;
```

AND 和 OR

DISTINCT 去重

## Like

\_ 一个字符

% 任意 (可以为空)



IN

IS NULL

BETWEEN A AND B [A, B]

把属性加起来

## Join

合并 (逗号

原名 alias

NOT IN

WHERE EXISTS (语句)

WHERE NOT EXISTS()

ANY (属性): 任意

ALL(一个属性): Every

ORDER BY attr ASC/DESC

..

## Double Negation

C 指向 AB

for A

是不是不存在

for B

是不是不存在 AB 一组的是对的

这样存在一个B找不到里面就有东西, A就false。

啥也没有就全找到 true 就全有

就是A包含的所有B都是对的

里面会返回错的的B, 不存在错的B, A就对了

或者就是  $n$  个 A,  $m$  个 B, 找出 A 对应  $m$  个全有的。

for A

是不是不存在 F(A) (B

F(A)

for B

是不是不存在有的堆上的 AB  
有一个错的，就会返回true  
全对，就会返回false

## Aggregate Queries

```
§AVG(attribute)
§SUM(attribute)
§MIN(attribute)
§MAX(attribute)
COUNT(*) 一般就 * 最好用
```

```
SELECT department, MAX(age) AS OldestAge
/* AS:别名 */
/* GROUP BY */
FROM Employee

GROUP BY department;
```

```
SELECT
FROM
WHERE
GROUP BY
HAVING /* 聚合完筛选 check having*/
ORDER BY /* 默认从高到低 */
```

```
INSERT INTO <table name> VALUES (...), (...);
```

```
INSERT INTO <table name> (SELECT * ... )
```

```
DELETE FROM <table name>
WHERE <condi>;
```

```
UPDATE <table name>
SET <col> = val
WHERE <con>
```

```
CREATE VIEW <vname> AS 自动更新
SELECT
FROM
WHERE;
```

```
CREATE MATERIALIZED VIEW <vname> AS 自动更新
SELECT
FROM
WHERE;
```

## Lecture 4 关系代数

$\pi$  projection 去掉不想要的(不满足关系的), 底下也要的属性

$\sigma$  选择行 (子集) sigma 选择, 底下写关系, 里面写库

$\times$  组合

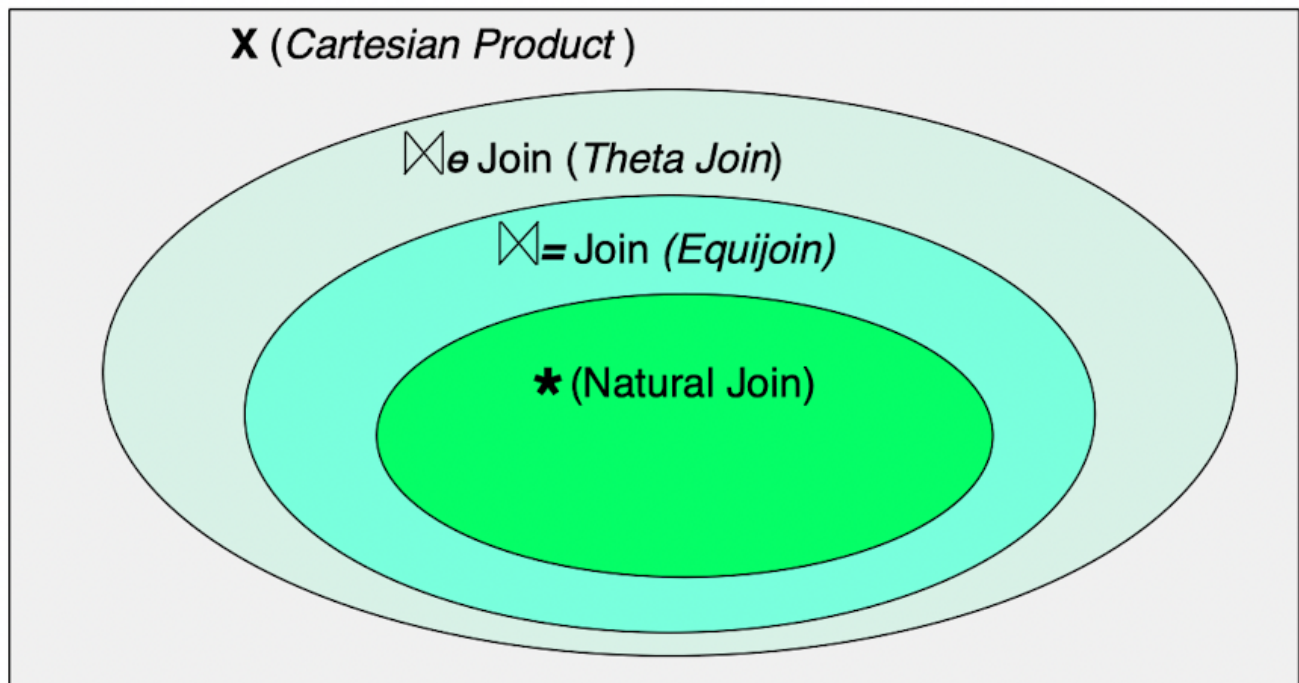
$\rho$  重命名

$join(\bowtie), division(/)$  :

$R$

$\bowtie_{\theta}$  有个条件

### Different Join operations and their relationships:



Equijoin

Natural Join 名字相同有个树形

$R \div S$ 。R里面那些全有 S 全包含然后保留, 然后把 S 有的那些属性去掉。

## Lecture 5 Integrity Constraints

修改 Schema

如果会造成，会不让 chg 必须尊重完整性

## Constraints Violate 类型

- Atom 必须原子化值 (atomic value) 不可再分 (注意机器想的完整，你觉得主观意义上的没意义)
- Unique: 元组必须本质不同
- Key: primary 必须本质不同
- Domain: 定义域也得对
- Not Null

```
CREATE DOMAIN <domain_name> AS <type>  
CHECK <constraint>
```

用 VALUE 代表要 check 的，增强定义域限制让他更合理、

Referential Integrity Constraint Violation 指向的找不到就急了‘

foreign 要比 refer 的远

```
◦ ALTER TABLE Employees ADD CONSTRAINT Department REFERENCES Departments(Code);
```

delete 也得查

FOREIGN KEY: ON DELETE ; ON UPDATE 违反干啥

默认: RESTRICT

```
FOREIGN KEY Employees(Departments)  
REFERENCES Departments(Code)  
ON DELETE RESTRICT ON UPDATE RESTRICT;
```

A 指 B, 指错了

- SET NULL 指向不存在赋值为空
- CASCADE 删掉 (A) 指的不对的 (很危险,
- Restrict (B 不让改)

## 其他限制 Constrains

§ Semantic Constraints:

§ Transition Constraints: 过度约束

We enforce these constraints with TRIGGER and ASSERTION

◦We don't cover this in this course

## Lab 1

```
CREATE TABLE <tablename> as SELECT ...  
UPDATE <tablename> SET xx = yy WHERE u = v;  
DELETE FROM <tablename> WHERE .. ;
```

## Lab 2

ssh [23096511d@csdoor.comp.polyu.edu.hk](ssh://23096511d@csdoor.comp.polyu.edu.hk)

sqlplus

```
source /compsoft/app/oracle/dbms.bashrc  
"23096511d"@dbms  
密码: 123456
```

# Lecture 6 Function Dependency & Normalization

## Keys Refresher

Superkey 唯一定义关系的,可以有很多 包含candidate key

Candidate Key: 极小的,可能有很多

Primary Key: 选的主键

Explicit and Implicit FDs 看成图,有传递性的间接

Armstrong's Axioms 函数依赖的有限公理系统

## Closure

$F^+$ : 能退出来的所有规则

$A^+$ : A 是属性: 能退出来的所有东西

## Anomalies 错错错

- Insertion Anomaly 定义域错了, 或者不让空 NULL 了
- Deletion Anomaly 造成了不是故意的信息损失
- Update Anomaly 比如更新造成了函数依赖的错误

# Normalization

## Normal Form

1/2/3NF, BCNF 包含

- 1NF: 元组互不相同, 不可分
- 2NF: 没有 Partial Dependencies (就Candidate key所有的才能退出他, 不能一部分退出了) 且是 1NF (去除, 把那些部分的拿出来建一个新图, Decomposition, 必须 Lossless Join, 就是乘起来还能还原【Lossy Join, 爆了】)
- 3NF: 是 2NF, 且没有传递的间接关系, 每个转移建个新表
- BCNF [3.5NF]: 任何关系  $x \rightarrow y$ ,  $x$  都是 superkey (这个已经保证是 3NF 了? 想不清楚)

partial - 传递 - 去交叉 CK

删除元素的时候注意:

- 左边有删掉整个
- 右边有就删掉字母

Denormalization

: 加速

## Lecture 7 Storage

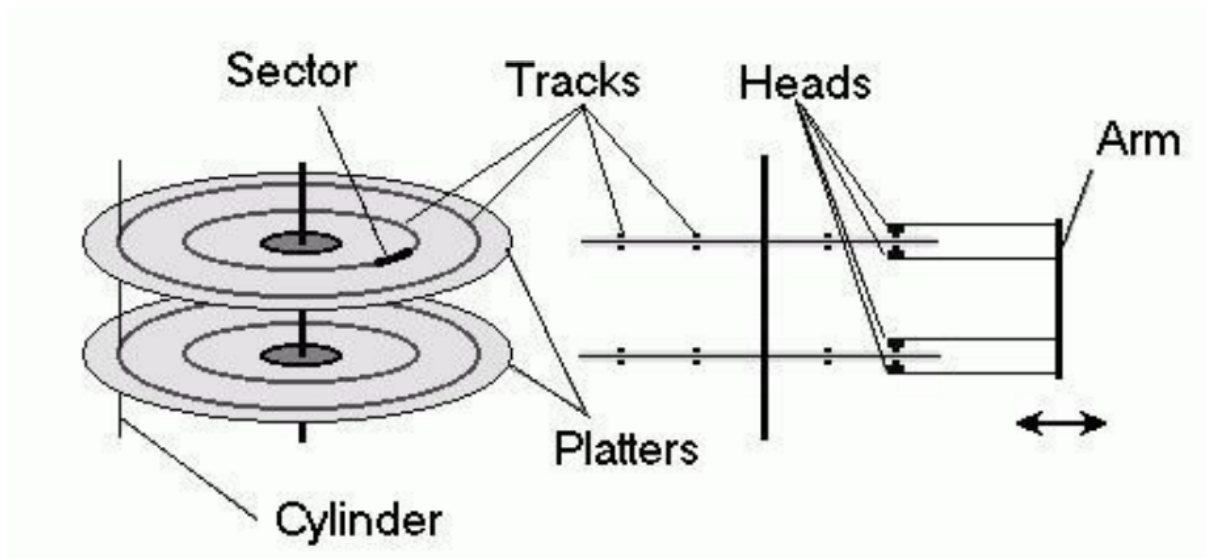
Transactions 逻辑排好序的

Disk Components:

Track (圆环 一个面 Ring): Platter (1?2 单面 / 双面):

Cylinder (相同直径的环叠起来)

Sectors



- Average block access time = 转 Seek Time + Rotational delay + Transfer time

## Questions

Parameter	Value
Sector size	512 bytes
# of sectors per track	50 sectors
# of tracks per surface	2000
# of platters	5 double-sided
Rotational speed	5400 rpm
Average Seek time	10 msec

1. What is the capacity of a track?
2. What is the capacity of a surface?
3. What is the disk capacity?

Since rotational speed is 5400 rpm (revolution per minutes), find:

4. the maximum **rotational delay**,
5. the average **rotational delay**

7. What is the average time to read an entire track from the beginning?

6. Assume one track can be transferred per rotation, what is the **transfer rate**?

rpm: revolution per minute

delay: 转一次要几秒

4 转一圈

avg / 2

6. transfer rate 就是一秒转多少个byte (转的是一个track) ? 就是固定时间 t, 看track能赚多少 (比如就一圈的时间) (总 track的byte / 一圈的时间)

7. block transfer time: 一个 block 多少秒 =  $B / tr$  平均。

8. average rotational delay: = 转一圈的时间 (transfer time) / 2

9. random: 所有都 \*, 连续的只乘transfer时间

## Cylinder-based Organization

Blocks on the same track or cylinder effectively involve only: the first seek time and the first rotational latency (?)

## 谁怎么存储的

Field - Record - Block - File 越来越大的包含

## Blocks & Files

- Blocking factor bfr ( $B / R$ )
- Fix 固定长度记录:
  - 好: 无额外空间, 相同访问速度,
  - 简单; 尾: Block internal fragmentation, Record internal fragmentation (不能扩展, 记录分开了因为用最大的), 回访问更多的 disk (more disk access)
- Var 变量长度记录
  - 好: No RIF
  - 坏: 访问时间和到开始距离成比例; 重新使用不方便; 没有空间如果成长很多

## Block Organization

Unspanned: 无法扩展

Spanned: 扩展 (超过一个块)

## Files

### Unordered Files (Heap, Random)

insert 很快啊

delete: deletion marker

### Ordered Files (Sequential)

有顺序

IO Cost (注意这里考虑的是平均切恰好

Heap:



- Scan B
- Equality Search  $0.5B$
- Range Search B
- Insert 2  
Sorted
- Scan B
- Equality  $\log B$
- Range  $\log B + \text{match}$
- Insert B

## Lecture 8 Index Structure

### Single-level indexes

### Ordered File 有顺序的

index entry = key + pointer

### Primary Index

索引就是每个 block 第一个是啥

Ordered

$\langle K(i), P(i) \rangle$  属性, 指针 分散的存比较好

Index entry: (1st record, block地址)

坟山的比较好

data blocks

一般来说一个 index 回占用很多 block

$$B_{index} \ll B_{data}$$

$bfr_{index}$  就是说一个 block 有多少 index entry

index

search:  $\log$  index block

### Clustering Index

有顺序, 但是在 non-key field (就会重复

存的是每个 key 第一次出现的那个快 (索引数是值域

# 无序文件

Dense

## Secondary Index

每个 point 都指向对应的，虽然很乱但没事，那个 field value 就按顺序放

dense index.

## Key 每个存

## Non-key 有重复

三个 level Block Pointer Blocks of Record Pointers

## Multi-level indexes

(B 树+  
底数 =  $fo = bfr$  index

B+ 树插入就是爆了就分一半，往上推，然后地柜插入父亲

## Lec 9 Query Optimazation

注意 没有 Disjunction 或。。 (

Selectivity: 选择性，有多少符合条件的 (存一下)

平均分布

outer 小

只考虑

- Nested-loop join: 把小的放外面然后/(N-2)
- Single-loop Join:  $B_{外} + N_{外}(L_{找} + 1)$

. General Transformation Rules -And 可拆 / Commutativity 交换律

一系列 II 可以值表留最后一个 (记不住也没意义)

他这个五步记不住也一点用没有，不是系统机械化处理方式，不要了，。

## Lecture 11 - 12 并行控制

## ACID Properties

Atomicity

Consistency

Isolation

Durability

- START (alias BEGIN, SET, DECLARE)
- Rollback (= Abort)

所有放一个 disk (valid) (?? 我什么意思来着)

Serial Executions correct: (存在一种顺序 并行效果要和序列一样 else nonserializable)

## Anomalies 并行的问题 (异常现象)

- Lost Update Problem: 写完没更新
- Dirty Read Problem: 读到脏东西
- Unrepeatable Read Problem 不可重读, (两次读入变了)

schedule = history (调度)

## 本质相同 (结果相同)

必须 read 改 wt commit

## Basic Two Phase Locking (2PL)

Growing 必须要锁, 一旦放锁不能在要了1

可以解决除了脏读的以外的问题 (Strict 能解决脏读)

任何 2PL 都可以调度 真神奇。(懒得想 / 证, 但感觉

可能会死锁

## Conservative 2PL:

没死锁 很保守 之前把锁全要了, 提前声明困难

锁的细粒度: Granularity of Locks 跟结果无关 跟效率有关

Fine 细 Coarse 粗

冲突概率粗高

并行效果和开销都是细高

检测到死锁只能杀了 / 方法 1 定期检查超时, 简单 可能 判错

## Wait-for-graph (WFG)

### Strict 2PL

结束再松开 write lock, 可能还有 deadlock

Conflicting Operations () Conflict equivalent / Conflict serializable (CSR):

Testing CSR  $i$  到  $j$  有边仅当  $i$  有一句话再  $j$  前面并且冲突 (操作变量相同, 至少一个 write

## Lecture 13 Recovery

### ACID

Recovery

- A (原子性, 要么全对 要么全不执行) Undo
- D (持续) (■ Atomicity & Durability) : Redo
- I (并行)
- C (完整性约束)

### Log

logical physical log

BFIM / AFIM 改的值, Back P, Next P (相同 tran 上一条吓一跳)

### Buffer

Buffer page size 和 disk block 一样, 就是相同最小单位\* (

### Cache Flushing

Simple: Force (但高 IO) / No-Steal (但 Cache 大): 这样undo redo都不用操作了。  
steal (之前悄悄 / 可以undo), forces(commit必须? / 不能redo)

### Write-Ahead Logging (WAL)

改前写

### Commit Point of a Transaction

Log 写完, 全执行对。没 commit 就要回滚。

# Checkpoint

周期检查写

刷新所有缓存页面到磁盘