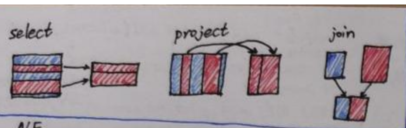


不符合原子性

id	name	color
5	bear	w, y, blue
6	hat	g, y
9	kite	r, blue, g
10	yoyo	w, y

依然非1NF，各列储存了相同分类的数据

id	name	color1	color2	color3
5	bear	w	y	blue
6	hat	g	y	
9	kite	r	blue	g

NF

1st normalization (1NF)

- 数据有原子性（每行要有具有原子性的值，也要有主键）
- 没有重复数据组

2NF

- 所有列都是主键的一部分，对主键起到补充说明作用，且符合1NF

若符合1NF的表只有一列主键，则也符合2NF

3NF

- 符合2NF，且没有传递函数依赖性．每一列只依赖主键．

Data Type

Text data types

char(len) 固定长度
varchar(len) 不定长度
clob 很长文本（text在mysql）

Date types

date YYYY-MM-DD
datetime YYYY-MM-DD HH:mm:ss
timestamp 同上
范围：1970-01-01 00:00:01 UTC
到 2038-01-19 03:14:07 UTC

SQL (Structured Query Language)

DDL、DML、DQL、DCL
↓ Data Control Language

1. Data Definition Language (DDL)

primary key

```
① create table emp(
     id int primary key,
     name varchar(30)
   );

② create table emp(
     id int,
     name varchar(30),
     constraint pk_con primary key(id)
   );

③ create table emp(
     name varchar(30),
     dep varchar(20),
     salary numeric(9,2),
     primary key(name, dep)
   );
```

foreign key

```
① create table emp(
     ...,
     depId int references dep(id)      -- 其他表 的某一列
   );

② create table emp(
     ...,
     constraint fk_dep foreign key(depId) references dep(id)   -- 本表的列
   );
```

not null

```
create table emp(
  ...,
  name varchar(30) not null
);
```

unique

```
① create table emp(
     name varchar(30) unique
   );
② ____ __(
     ...,
     unique(name)
   );
③ ____ __(
     ...,
     unique(a, b)
   );
```

default

```
____ __(
  ...,
  salary numeric(9,2) default 0.0
);
```

constraint

```
① price numeric constraint positive_price check (price > 0)
② ____ __(
     ...,
     check (price >0)
   );
```

2. Data Manipulation Languge (DML)

update

```
insert into table (col1, col2, col3) values (..., ..., ...);
update table set col1 = __ where ____;
delete from table where ____;
```

drop drop table table_name;

alter alter table r add column name varchar(30)
alter table r drop column name
(checks, foreign key, indexes 也可以 drop)

alter (cont.)

加列 alter table emp add column office varchar(30) [check (office <> '')];
删列 alter table emp drop column depId [cascade];
加限制 alter table emp add { primary key(id) / foreign key (depId) references dept(id) / constraint phone_unique unique(phone) }
删限制 alter table emp drop constraint cons_name;
修改默认 alter table emp alter column salary set default 0.00
alter table emp alter column salary drop default
重命名 alter table emp rename column phone to tel_no
alter table emp rename to employee

alter table emp alter column col set not null
alter table emp alter column col drop not null

修改数据类型
alter table emp alter column salary type numeric(9,4);

Numerical

int bigint 整型
float(n) 用户指定精度，共n位 （不是小数点后n位）
real 固定6位有效
numeric(p,d) 共p位，小数点后d位．

Binary data types

raw(max length)
varbinary(max length)
blob binary large object
bytea used in pgsql

3. Data Query Language (DQL)

```
select ¹ as from table
where² ______ (可以用 in (...))   -- 集合/子查询
group by³ ______
having⁴ ______
order by⁵ ______ asc/desc, ______ asc/desc, ...   (默认 asc)
limit __ offset __ (默认 offset 0)
```

<1> ① as 可改名
② 聚合函数：count([distinct] col), max(col), min(col), avg(col)
③ || 拼接字符串
④ case

```
select name, case
  when score > 90 then 'A'
  when score < 60 then 'C'
  else 'B'
  end as grade
from student;
```

<2> ① in, not in
② like, %, _
where name like '张%'
%：匹配0个、1个、多个 _：匹配1个
③ null, not null
where name is null
where name is not null

<3><4>

```
select count(district), district from stations
group by district
having district <> '';
```

Some functions

① upper(__), lower(__): 大/小写
② trim(__): 去除头尾空格
③ substr(__, __, __): 子串
substr('Nanshan', 3, 4) returns 'nsha'
④ replace(__, __, __): 替换
replace('Sheep', 'ee', 'i') returns 'Ship'
⑤ length(__): 长度
⑥ round, trunc:
round(3.141592, 3) --3.142
trunc(3.141592, 3) --3.141
⑦ cast(col as type)

Join

1. Cross Join 笛卡尔积
select * from t1 cross join t2

t1 cross join t2 / t1 inner join t2 on true / t1, t2 } are same

2. Inner Join
select * from { t1 inner join t2 on t1.num = t2.num / t1 inner join t2 using (num) / t1 natural join t2 }
inner 可省略
nature join 会把所有 t1 和 t2 共有的列加入 using(...) 中

3. Left/Right Outer Join
t1 left join t2 on t1.num = t2.num
t1 left join t2 using (num)

4. Full Outer Join

Combining Query

query1 union[all] query2 并集
query1 intersect[all] query2 交集
query1 expect[all] query2 差集
加all不去重，不加all等于加distinct

About NULL

col + null, col − null, col * null, col / null ⇒ null
col > null, col = null, col < null ⇒ null
true and null → null
false and null → false
true or null → true
false or null → null
col is null → true false

Ordering

用case排序

```
order by
  case
    when role='A' then 1
    when role='D' then 2
    else 3
  end
```

排序规则：
string 字典序
number 大小
boolean false < true
dates & times 越晚越大
SQL Server, MySQL, null 最小
Oracle, pgsql null 最大

Window Function

① over() 开窗为全局
② over (partition by ... order by ...)

```
select country, title, runtime, year_release,
       rank() over (order by year_release)
from movie where year_release > 2015;

select ________,
       rank() over (partition by country)
from movie where year_release > 2015;
```

③

```
select country, runtime, year_release,
       avg(runtime) over (partition by country
       order by year_release)
from movie where year_release > 2015;

select country, runtime, year_release,
       avg(runtime) over (partition by country, year_re
from movie where year_release > 2015;
```

country	runtime	y_r	avg
bd	160	2016	160
bd	140	2017	75.5
bd	0	2017	75.5
bd	2	2017	75.5

country	runtime	y_r	avg
bd	160	2016	160
bd	140	2017	47.33...
bd	0	2017	47.33...
bd	2	2017	47.33...

④

rank():	1	2	2	2	5	5	7	8	8	8	11	12	12
dense_rank():	1	2	2	2	3	3	4	5	5	5	6	7	7
row_number():	1	2	3	4	5	6	7	8	9	10	11	12	13

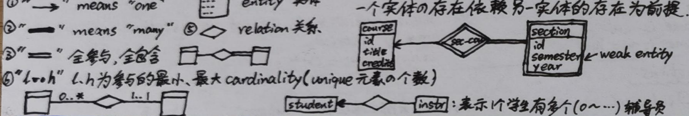
⑤ lag (col, __, __) 填补空白(可无)；偏移量(可无)

⑥ Multi-Window

```
select movieid, title, country, y_r, runtime,
       sum(runtime) over w sum_all,
       max(runtime) over w max,
       avg(runtime) over w avg
from movie where y_r > 2010
window w as (partition by country order by y_r);
```

E-R Diagram

① "→" means "one"
② "—" means "many"
③ "═" 全参与，全包含
④ entity 实体
⑤ ◇ relation 关系
⑥ "l..h" l, h 为参与的最小、最大 cardinality (unique 元素の个数)
⑦ Weak entity (弱实体)：一个实体の存在依赖另一实体的存在为前提．



student ← ◇ — instr：表示1个学生有多个(0~...)辅导员

Relation Algebra

6 basic operator: select σ, project Π, union ∪, set difference −, Cartesian product ×, rename ρ

① $\sigma_p(r)$：从表 r 中根据条件 p 选择
$\sigma_{dept='phd' \wedge salary<90000}(instructor)$
② $\Pi_{A_1, A_2, \cdots A_k}(r)$：从表 r 选出 $A_1, A_2, \cdots A_k$ 列
$\Pi_{id, name, salary}(instructor)$
③ Join: $\sigma_{instr.id = teach.id}(instr \times teach)$
另一种表示：$r \bowtie_\theta s = \sigma_\theta(r \times s)$ θ: predicate (谓词)
④ ∪ union, ∩ intersect, − difference
$r \cup s$ $r \cap s$ $r - s$
⑤ "←" Assignment (赋值)
$Physics \leftarrow \sigma_{dept='phy'}(instr)$
$Music \leftarrow \sigma_{dept='music'}(instr)$
$Physics \cup Music$
⑥ Rename ρ
$\rho_x(E)$：给表达式 E 的结果命名为 x
$\rho_{x(A_1, A_2, \cdots A_n)}(E)$：也可以给列命名为 $A_1, A_2, \cdots, A_n$

Transaction

ACID:
Atomicity: 要么全提交，要么不提交．
Consistency: 要么创建一个新有效数据状态，要么回滚
Isolation: 隔离性
dirty read: 读取其他事务修改未提交的数据(sj)
nonrepeatable read: 重新读取它以前读的sj，发现被已提交の事务修改
phantom read: 重新执行一个查询，发现结果集由于最近另一个事务的提交改变
serialization anomaly: 提交一组事务的结果可能因每个事务提交顺序不同而结果不同．
Durability: 提交の sj 由系统保存，即使故障或重启，sj 也正常可用．

A allowed, but not in pgsql
P possible
N not possible

Isolation level	DR	NR	PR	SA
Read uncommitted	A	P	P	P
Read committed (default in pgsql)	N	P	P	P
Repeatable read	N	N	A	P
Serializable	N	N	N	N

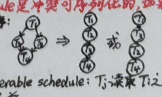
SQL about transaction:

```
begin;          rollback to sp1;
commit;         show transaction_isolation;
rollback;       set default_transaction_isolation = '___';
savepoint sp1;
```

Serializability

一个schedule（并发的）等价于一个串行的schedule，则它可序列化．
简化为只有read, write操作，两个过程都read(R)，不冲突，其它3种冲突．操作对象不同，不冲突．

一个schedule S 可通过交换不冲突操作得到 S'，则 S 与 S' 冲突等价．
S 冲突等价于一个串行 schedule，则 S 冲突可序列化．
Precedence graph：若 Ti, Tj 冲突，Ti 必须在 Tj 前，则画 (Ti)→(Tj)
一个schedule是冲突可序列化的，如果 precedence graph 是无环的．
拓扑排序：



Recoverable schedule: Tj 读取 Ti 之前写入的数据，则 Ti 的提交在 Tj 之前．

Function & Procedure

在 pgsql 里，procedure 是返回值为 void（没返回值）的函数
Function 分类：
- Query Languge func (written in SQL)
- Procedure Language func (pgSQL/Tcl/Perl/Python)
- Internal function
- C-language function

1. SQL Func

```
create [or replace] function add(x integer, y integer)
returns integer as $$
  select x+y;      <- select $1+$2;  (integer, integer)
$$ language sql;
```

有输出，则是：(x int, y int, out sum int)
有默认值：(x int, y int default 2, z int default 3) 有缺省值的后面都必须有缺省

2. PgSQL Func

```
create or replace function func_name(...) returns re_type
as $$
declare
    var_name var_type := value;
    ...
begin
    ...;
end
$$ language plpgsql;
```

4. 返回值类型也可以是表

```
returns table
(
    col1 col1_type,
    col2 col2_type,
    ...
)
```

3. 条件语句

```
① if (...) then
     ...;
   elseif (...) then
     ...;
   else
     ...;
   end if;

② for var_name in start..end loop
     ...;
   end loop;
例：for i in 1..num loop
       res = res * i;
   end loop;

③ while condition loop
     ...;
   end loop;
```

Trigger

Trigger Function 返回值为 trigger，返回 null 或 new, old 等．

```
create trigger trigger_name
before/after insert or update or delete (一个或多个)
on table_name
for each row/statement
execute procedure trigger_func();
```

1. Trigger 可作用于：1. tables 2. views 3. foreign tables
2. 影响：1. row 每改变一行触发一次 2. statement 执行一条语句触发1次，即使语句改变了多行
3. before、after、instead of (少用)

when	event	row-level	statement-level
before	I/U/D	T. F	T. V. F
before	Trun	–	T
after	I/U/D	T. F	T
after	Trun	–	T

T: tables
V: views
F: foreign tables

4.

Variable	Data Type	说明
new	record	new 为新的行，若为 row 的 I/U，new 为 null，若为 stat 或 row 的 D
old	record	old 为旧的行，若为 row 的 U/D，old 为 null，若为 stat 或 row 的 I
TG_name	name	name of trigger
TG_when	text	before/after/instead of 字符串
TG_level	text	row/statement 字符串
TG_op	text	insert/update/delete/truncate

（TG_when、TG_level、TG_op：全大写！）

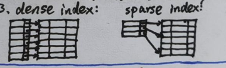
5. Event trigger

```
create event trigger name
on event_name
execute function func_name();
例：create event trigger tr_fun_ev on ddl_command_st
    execute function event_tr();
```

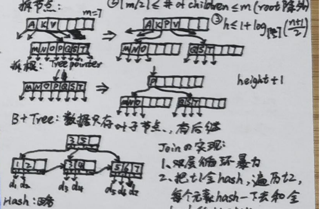
Indexing

```
create index idx_name on table_name (col1, col2, ...);
drop index [if exists] idx_name [cascade];
```

分类：
1. integrated index: PK index in MySQL InnoDB, SQL Server
external index: indexes in PgSQL, MySQL MyISAM
2. clustered index: primary index
non-clustered index: secondary index (have to be dense) ← 有重复，像 hash 一样装篮子
3. dense index: sparse index:
4. multi-key index
single-key index

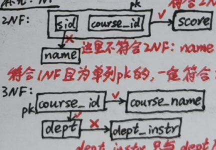


B-Tree 实现
① # of children = # of keys + 1
② ⌈m/2⌉ ≤ # of children ≤ m (root 除外)
③ $h \le 1 + \log_{\lceil m/2 \rceil}\left(\frac{n+1}{2}\right)$
拆节点：
拆根：
height + 1
B+ Tree：数据只存叶子节点，有后继
Hash：略



Join の实现：
1. 双层循环暴力
2. 把 t1 全 hash，遍历 t2，每个元素 hash 一下去和全 hash 的 t1 对比
3. sort-merge join
排序，然后双指针扫

补充：NF



符合2NF：score 与 pk 都有关
这里不符合2NF：name 只与 pk 的一部分 sid 有关
符合1NF且为单列pk的，一定符合2NF
dept_instr 只与 dept 有关，与 pk 没直接关系，只有传递关系

With 用法

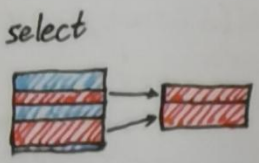
1. 表示一个表，简化子查询

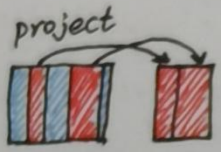
```
with t1 as (
  select ...
)
select * from t1
where ...
```

2. 表示迭代

```
with recursive t(n) as (
    values (1)
    union all
    select n+1 from t where n<100
)
select sum(n) from t limit 100;

with recursive t(n) as (
    select 1
    union all
    select n+1 from t
) select n from t limit 100;
```





不符合原子性

id	name	color
5	bear	w,y,blue
6	hat	g,y
9	kite	r,blue,g
10	yoyo	w,y

依然非1NF，各列储存了相同分类的数据

id	name	color1	color2	color3
5	bear	w	y	blue
6	hat	g	y	
9	kite	r	blue	g

NF

1st normalization (1NF)

- 数据有原子性(每行要有具有原子性的值，也要有主键)
- 没有重复数据组

2NF

- 所有列都是主键的一部分，对主键起到补充说明作用，且符合1NF

若符合1NF的表只有一列主键，则也符合2NF

3NF

- 符合2NF，且没有传递函数依赖性。每一列只依赖主键。

Data Type

Text data types

char(len) 固定长度
varchar(len) 不定长度
clob 很长文本(text在mysql)

Date types

date YYYY-MM-DD
datetime YYYY-MM-DD HH:mm:ss
timestamp 同上
范围: 1970-01-01 00:00:01 UTC
到 2038-01-19 03:14:07 UTC

SQL (Structured Query Language)

DDL、DML、DQL、DCL

1. Data Definition Language (DDL)

Data Control Language

primary key

①
```
create table emp(
    id int primary key,
    name varchar(30)
);
```

②
```
create table emp(
    id int,
    name varchar(30),
    constraint pk_con primary key(id)
);
```

③
```
create table emp(
    name varchar(30),
    dep varchar(20),
    salary numeric(9,2),
    primary key(name, dep)
);
```

foreign key

①
```
create table emp(
    ...,
    depId int references dep(id)
);
```
(dep: 其他表; id: 其他表的某一列)

②
```
create table emp(
    ...,
    constraint fk_dep foreign key(depId) references dep(id)
);
```
(depId: 本表的列)

not null

```
create table emp(
    ...,
    name varchar(30) not null
);
```

unique

①
```
create table emp(
    name varchar(30) unique
);
```

②
```
_____ _(
    ...,
    unique(name)
);
```

③
```
_____ _(
    ...,
    unique(a, b)
);
```

default

```
_____ _(
    ...,
    salary numeric(9,2) default 0.0
);
```

constraint

① price numeric constraint positive_price check (price > 0)

②
```
_____ _(
    ...,
    check(price > 0)
);
```

2. Data Manipulation Languge (DML)

update
```
insert into table (col1, col2, col3) values (--, --, --);
update table set col1 = ___ where _____;
delete from table where _____;
```

drop
```
drop drop table table_name;
```

alter
```
alter table r add column name varchar(20)
alter table r drop column name
```
(checks, foreign key, indexes 也可以 drop)

alter (cont.)

加列 `alter table emp add column office varchar(30) [check (office <> '')];`
删列 `alter table emp drop column depId [cascade];`
加限制 `alter table emp add { primary key(id) / foreign key (depId) references dept(id) / constraint phone_unique unique(phone) }`
`alter table emp alter column col set not null`
删限制 `alter table emp drop constraint cons_name;` `alter table emp alter column col drop not null`
修改默认 `alter table emp alter column salary set default 0.00` / `alter table emp alter column salary drop default`
修改数据类型 `alter table emp alter column salary type numeric(9,4);`
重命名 `alter table emp rename column phone to tel_no` / `alter table emp rename to employee`

Numerical

不是小数点后n位

int bigint 整型
float(n) 用户指定精度，共n位
real 固定6位有数
numeric(p,d) 共p位，小数点后d位.

Binary data types

raw(max length)
varbinary(max length)
blob binary large object
bytea used in pgsql

3. Data Query Language (DQL)

```
select 1 as ... from table
where 2 ____ (可以用 in (...))
group by 3 ____
having 4 ____
order by 5 ____ asc/desc , ____ asc/desc , ...
limit __ offset __ (默认 offset 0)
```

(默认 asc)

<1> ①as 可改名

②聚合函数: count([distinct] col) / max(col) / min(col) / avg(col)

③|| 拼接字符串

④ case

```
select name, case
  when score > 90 then 'A'
  when score < 60 then 'C'
  else 'B'
  end as grade
from student;
```

<2> ① in, not in

② like, %, _

```
where name like '张%'
```

%: 匹配0个、1个、多个 _: 匹配1个

③ null, not null

```
where name is null
where name is not null
```

<3> <4>

```
select count(district), district from stations
group by district
having district <> '';
```

Some functions

①upper(__), lower(__): 大/小写 ②trim(__): 去除头尾空格 ③substr(__,__,__): 子串

substr('Nanshan', 3, 4) returns 'nsha'

④replace(__,__,__): 替换 ⑤length(__): 长度 ⑥round, trunc:

replace('Sheep', 'ee', 'i') returns 'Ship'

round(3.141592, 3) --3.142
trunc(3.141592, 3) --3.141

⑦ cast(col as type)

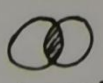
Join

1. Cross Join 笛卡尔积

```
select * from t1 cross join t2
```

t1 cross join t2
t1 inner join t2 on true
} are same
t1, t2

2. Inner Join

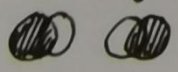


inner 可省略

```
select * from { t1 inner join t2 on t1.num = t2.num
              / t1 inner join t2 using (num)
              / t1 natural join t2
```

nature join会把所有t1和t2都共有的列加入using(...)中

3. Left/Right Outer Join



t1 left join t2 on t1.num = t2.num
t1 left join t2 using (num)

4. Full Outer Join

Combining Query

query1 union [all] query2 并集 加all不去重
query1 intersect [all] query2 交集 不加all等于加distinct
query1 except [all] query2 差集

Ordering

用case排序

```
order by
  case
    when role='A' then 1
    when role='D' then 2
    else 3
  end
```

排序规则:
string 字典序
number 大小
boolean false < true
dates & times 越晚越大
SQL Server, MySQL, null 最小
Oracle, pgsql null 最大

About NULL

col + null
col - null
col * null
col / null
⇒ null

col > null
col = null ⇒ null
col < null

true and null → null
false and null → false
true or null → true
false or null → null

col is null → true false

Window Function

① over() 开窗为全局

② over (partition by ··· order by ···)

```
select country, title, runtime, year_release,
       rank() over (order by year_release)
from movie where year_release > 2015;
```

```
select _____,
       rank() over (partition by country)
from movie where year_release > 2015;
```

③
```
select country, runtime, year_release,
       avg(runtime) over (partition by country
       order by year_release)
from movie where year_release > 2015;
```

```
select country, runtime, year_release,
       avg(runtime) over (partition by country, year_re
from movie where year_release > 2015;
```

country	runtime	y_r	avg
bd	160	2016	160
bd	140	2017	75.5
bd	0	2017	75.5
bd	2	2017	75.5

avg

country	runtime	y_r	avg
bd	160	2016	160
bd	140	2017	47.33···
bd	0	2017	47.33···
bd	2	2017	47.33···

avg

④ rank(): 1 2 2 2 5 5 7 8 8 8 11 12 12

dense_rank(): 1 2 2 2 3 3 4 5 5 5 6 7 7

row_number(): 1 2 3 4 5 6 7 8 9 10 11 12 13

⑤ lag(col, 1, —) 填补空白(可无)
偏移量(可无)

⑥ Multi-Window

```
select movieid, title, country, y_r, runtime,
       sum(runtime) over w sum_all,
       max(runtime) over w max,
       avg(runtime) over w avg
from movie where y_r > 2010
window w as (partition by county order by y_r);
```

E-R Diagram

① "→" means "one"

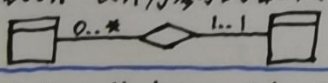
② "▬" means "many"

③ "═" 全参与, 全包含

④ entity 实体

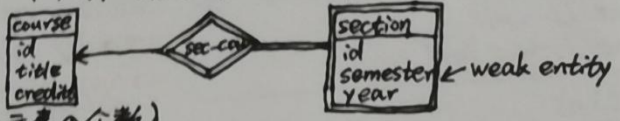
⑤ relation 关系

⑥ "l..h" l..h为参与的最小、最大 cardinality(unique元素的个数)



student ← instr : 表示1个学生有多个(0~···)辅导员

⑦ Weak entity(弱实体):
一个实体的存在依赖另一实体的存在为前提.



Relation Algebra

6 basic operator: select σ, project Π, union ∪, set difference −, Cartesian product ×, rename ρ

① $\sigma_p(r)$: 从表r中根据条件p选择

$\sigma_{dept='phd' \wedge salary<90000}(instructor)$

② $\Pi_{A_1, A_2, \cdots A_k}(r)$: 从表r选出 $A_1, A_2, \cdots A_k$ 列

$\Pi_{id, name, salary}(instructor)$

③ Join: $\sigma_{instr.id = teach.id}(instr \times teach)$

另一种表示: $r \bowtie_\theta S = \sigma_\theta(r \times s)$ θ:predicate(谓词)

④ ∪ union, ∩ intersect, − difference

$r \cup s$ $r \cap s$ $r - s$

⑤ "←" Assignment(赋值)

Physics ← $\sigma_{dept='phy'}(instr)$

Music ← $\sigma_{dept='music'}(instr)$

Physics ∪ Music

⑥ Rename ρ

$\rho_x(E)$: 给表达式E的结果命名为x

$\rho_{x(A_1, A_2, \cdots A_n)}(E)$: 也可以给列命名为 $A_1, A_2, \cdots, A_n$

Transaction

ACID:

Atomicity: 要么全提交, 要么不提交.

Consistency: 要么创建一个新有效数据状态,
要么回滚

Isolation: 隔离性

dirty read: 读取其他事务修改未提交的数据(sj)

nonrepeatable read: 重新读取它以前读的sj,
发现被已提交的事务修改

phantom read: 重新执行一个查询, 发现结果集
由于最近另一个事务的提交改变

serialization anomaly: 提交一组事务的结果可能因
每个事务提交顺序不同而结果不同.

Durability: 提交的sj由系统保存, 即使故障或重启,
sj也正常可用.

A allowed, but not in pgsql
P possible
N not possible

Isolation level	DR	NR	PR	SA
Read uncommitted	A	P	P	P
Read committed	N	P	P	P
Repeatable read	N	N	A	P
Serializable	N	N	N	N

default in pgsql

SQL about transaction:

```
begin;
commit;
rollback;
savepoint sp1;
rollback to sp1;
show transaction_isolation;
set default_transaction_isolation = '____';
```

Serializability

一个schedule(并发的)
等价于一个串行的 sched
则它可序列化.

简化为只有read, write
两个过程都read(B), 不
其它3种
操作对
不冲突

Function & Procedure

在pgsql里，procedure是返回值为void(没返回值)的函数

Function分类：
- Query Language func (written in SQL)
- Procedure Language func (pgSQL/Tcl/Perl/Python)
- Internal function
- C-language function

2. PgSQL Func

```
create or replace function func_name(...) returns re_type
as $$
declare
    var_name var_type := value;
    ...
begin
    ...;
end
$$ language plpgsql;
```

4. 返回值类型也可以是表

```
returns table
(
    col1 col1_type,
    col2 col2_type,
    ...
)
```

1. SQL Func

```
create [or replace] function add(x integer, y integer)
returns integer as $$
    select x+y;        -- select $1+$2;
$$ language sql;
```

(integer, integer)

有输出，则是：(x int, y int, out sum int)

有默认值：(x int, y int default 2, z int default 3)

有默认值的输入都必须有缺省

3. 条件语句

```
① if (...) then
    ...;
elseif (...) then
    ...;
else
    ...;
end if;
```

```
② for var_name in start..end loop
    ...;
end loop;
例: for i in 1..num loop
        res = res * i;
    end loop;
③ while condition loop
    ...;
end loop;
```

Trigger

Trigger Function返回值为trigger，返回null或new, old等.

```
create trigger trigger_name
before/after insert or update or delete (一个或多个)
on table_name
for each row/statement
execute procedure trigger_func();
```

1. Trigger可作用于：1. tables 2. views 3. foreign tables

2. 影响：1. row 每改变一行触发次 2. statement 执行一条语句触发1次，即使语句改变3多行

3. before、after、instead of(作用)

when	event	row-level	statement-level
before	I/U/D	T、F	T、V、F
before	Trun	-	T
after	I/U/D	T、F	T
after	Trun	-	T

T: tables
V: views
F: foreign tables

4. Variable

Variable	DataType	说明
new	record	new为新的行，若为row的I/U，new为null，若为stat或row的D
old	record	old为旧的行，若为row的U/D，old为null，若为stat或row的I
TG_name	name	name of trigger
TG_when	text	before/after/instead of字符串
TG_level	text	row/statement字符串
TG_op	text	insert/update/delete/truncate

全大写!

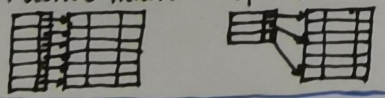
5. Event trigger

```
create event trigger name
on event_name
execute function func_name();
例: create event trigger tr_fun_ev on ddl_command_st
    execute function event_tr();
```

Indexing

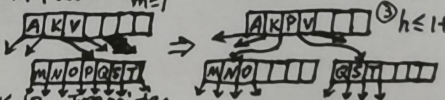
```
create index idx_name on table_name (col1, col2, ...);
drop index [if exists] idx_name [cascade];
```

分类：
1. integrated index: PK index in MySQL InnoDB, SQL Server
   external index: indexes in PgSQL, MySQL MyISAM
2. clustered index: primary index
   non-clustered index: secondary index (have to be dense)
   有重复，像hash一样装篮子
3. dense index: sparse index!
4. multi-key index
   single-key index



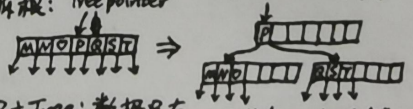
B-Tree实现 ① # of children = # of keys + 1
② ⌈m/2⌉ ≤ # of children ≤ m (root除外)

拆节点：m=7



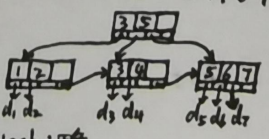
③ $h \le 1 + \log_{\lceil m/2 \rceil}\left(\frac{n+1}{2}\right)$

拆根：Tree pointer



height+1

B+Tree：数据只存叶子节点，有后继

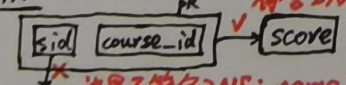


Hash：略

Join的实现：
1. 双层循环暴力
2. 把t1全hash，遍历t2，每个元素hash一下去和全hash的t1对比
3. sort-merge join
   排序，然后双指针扫

补充：NF

2NF：

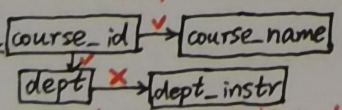


符合2NF：score与pk都有关

这里不符合2NF：name只与pk的一部分sid有关

符合1NF且为单列pk的，一定符合2NF

3NF：



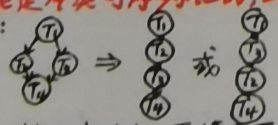
dept_instr只与dept有关，与pk没直接关系，只有传递关系

一个schedule S可通过交换不冲突操作得到S'，则S与S'冲突等价.
S冲突等价于一个串行schedule，则S冲突可序列化.

Precedence graph：若Ti, Tj冲突，Ti必须在Tj前，则画 (Ti)→(Tj)

一个schedule是冲突可序列化的，如果precedence graph是无环的.

拓扑排序：



read, write操作，read(Q)，不冲突，其它3种冲突.
操作对象不同，不冲突.

Recoverable schedule：Tj读取Ti之前写入的数据，则Ti的提交在Tj之前.

With用法

1. 表示一个表，简化子查询

```
with t1 as (
    select ...
)
select * from t1
where ...
```

2. 表示迭代

```
with recursive t(n) as (
    values (9)
    union all
    select n+1 from t where n<100
)
select sum(n) from t limit 100;
```

```
with recursive t(n) as (
    select 1
    union all
    select n+1 from t
) select n from t limit 100;
```