, create table

- table name } In sensi
  key word } $\underline{\underline{}}$

- create table t_n (
      A1 D1,
      A2 D2,
      A3 D3 $\underline{\underline{}}$)

△ data type
  char (length)
  var ... (max l)
  int
  float (n)
  numeric (p.d)
  date
  date time
  time stamp

△ constraints
  eg. create table r (
        id int primary key,
        f_n varchar (30) not null
        born numeri (4) unique
  eg. unique (f_n, l-n),
  eg check (f_n = upper(s_n))
  constraint con_name check(..)

△ foreign key
  eg coun char(2),
     foreign key (coun) references
     con_list (con-code)

△ insert into lab (a, t, te)
  values ('408', '2-78', 36)

△ constraints
  where country = ~~X~~ 'us'

△ select '437' as FOO
  select time * 10 as time10
                ↓
              can be + - × /

△ logical
  in (a, b, c)
  between a and b [a,b]
  <>
  =
  NOT : eg c not in ('us', 'gh')
  AND
  OR

△ like
  { % : title not like '%A%'
  { _ ONLY 1

△ null
  runtime $\overset{\neq}{\underline{\underline{\text{is}}}}$ null

△ f
- select t || 'is'
           || r_l   move_l
                          ↓
                      column name

- cast (col_n as varchar)
- round(3.14152, 3)   3.142
  trunc (, 3)   3.141
- upper / lower ('...')
  substr ('citizen', 5, 3) : zen
  trim (' o ')   'o'
  replace ('sheep', 'ee', 'i') 'ship'
- case (boolean)
  select when true then '...'
              ... false, ... '...'
              else (...)
          end as status
          { no comma
            as
  no else : null
  when (boolean) then

△ distinct
△ aggregate
- select count (*) num
  from movies
  group by country ;
- select min(...) old
    from movies
    group by country
    HAVING min(..d) < 1860
- ignore Null
- { count (*)   Null in
  { count (col)   Null not in
- select count (distinct col)

△ join
- all col all row
- select ..
  from ..
  join ..
  ON country = code
- = natural join
  = join ... using (id)

case
- when ... is null then 0
  end

△ set operator
  union : remove duplicate
          rows
  union all : no remove

△ subquery
  where country in (
     select ...
     from ...
     where c = 'En'
  );

△ update
  update t-name
  set c-name = new_val,
      c2 = v2,
      ...
  where ...
  eg update people p
      set num = (
         select count (*)
         from credit c
         where c.pid = p.pid
)
      where pid < 500;

△ delete
  delete from t-n
  where ..

○ procedural <-> declarative

△ f
- (create [or replace] function
  fun_name (args)
  returns r_type
  as $$
    Declare
        ... ;
    begin
      return val / val_n
    end;
  $$ language plpgsql;
  eg.
  create function full_name
  (p_fname varchar, ..)
  return varchar
  as $$
  begin return case
            when p_fname is
                 null then ..
                 else ... end
                 ||.
  end;
  $$ language plpgsql;

- use it:

```
select full-name (first-name),
from ..
where sur like '%(von)';
```

- returns con. c-n %type

```
declare   v-n   con.c-n %type
begin   select  c-n
        into    v-n
        from    con
        where ...
        return  v-n
```

△ procedure
- returns void
- if n=0
  then raise exception '.'
  end if;

△ relational
- select $\sigma_p(r)$
  p: selection predicate

$=. \neq \ > \geq < \leq$
$\wedge$ (and) $\vee$. $\neg$
eg $\sigma_{n='1' \wedge s>P00}(ins)$

- $\Pi_{A_1, A_2 ..}(r)$
  return args relation
  eg $\Pi name(\sigma_{n='s'}(ins))$

- $\underset{join}{A \times B} = \{(x,y) \mid x \in A \underset{and}{\wedge} y \in B\}$
  eg $\sigma_{p.id=q.id}(p \times q)$
- $r \bowtie_\theta s = \sigma_\theta(r \times s)$
- $r \cup s$: union
- $r - s$: in r not s
- $P \leftarrow \sigma_{p='m'}(ins)$
  $Q \leftarrow \sigma_{d=''}(ins)$
  $P \cup Q$

△ transaction
- Begin transection,
  update emp set
  s=s+10 where i=5;
  commit,
  rollback,
- conflict serializability
- view serializability

- $T_1 \longrightarrow T_2$
  T_1 must appear before T_2

△ Trigger
- create trigger t-n

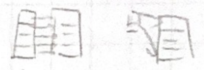
```
  before/after update/...
  on t-n
  for each row
  execute procedure f-n();

create or replace
function f-n()
returns trigger
as
$$
begin   select count(new.id)
        into new.num
        from credit.c
        where c.id=new.id
        return new;
end;
$$ language plpgsql
```

- new/old row before/after changes
- modify input
    before insert/or update
  check rules
    before in/update
  manage redundancy
    after ././.
- auditing

```
begin
  insert into audit (.,.)
  values (new.id,
      case
        when tg_op='update'
          then 'u'
          ..
        else 'x'
        end,
      current_timestamp);
```

- If want update incoming data: before trigger

△ view
- create view v-n as
    select..
- eg create view v-n
    (c1, c2, c3) as
  select   m.ti
           c.c-n
           m.~
  from movie m join con c on.

△ index
create index i_n
on t-n (c_n).
- $\underset{secondary}{(Non-)}$ $\underset{primary}{clustered}$ index
  specify order of r stored
- 
  dense   sparse indx

△ B-tree
- 每节 $[\frac{n}{2}, n]$
  root ≥ 2 子
  leave in same height