FECHA 2 ~~of~~ Aug

- Database design:
  - Logical design → Designing schema
  - Physical design → -"- Physical layout of database

Design Approaches:
Two ways:
- Entity Relationship model (ER)
- Normalisation Theory

- Database Engine
  - Storage manager
  - Query Processing
  - Transaction manager

① Storage man.

- Storage manager - provides interface between the low level data stored in database & the application programs & queries submitted to system.

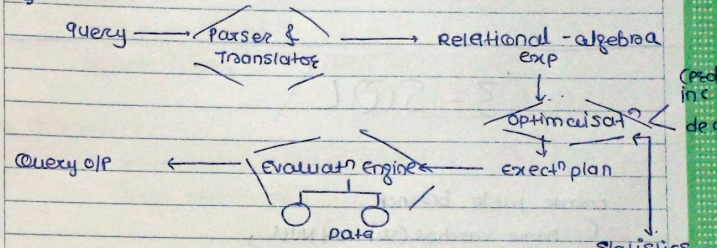
Tasks:
ⓐ Interact with OS file man
ⓑ Eff storing, retrieval

Issue:
storage
file, Indexing

FECHA

② Query
1) parsing & Translation
2) Optimisation
3) Evaluation

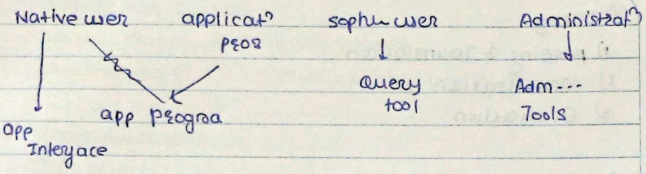
ⓐ



Alternative way in eval:
① Equivalent expr.
② Diff algo for each

- Need to estimate the cost
  - Depends on statistical info
  - Needs to estimate statistics

③ Transaction Management:
- collect° of operat° that perform single logical Funct° in database applicat°



3 - SQL

```
create table branch
(  name varchar(50) not NULL ,
 ---);
```

Constraints:

```
create table branch
( branch_name  char(15)
  primary key (branch_nam))
```

Rename operation:

```
select customers-name , borrower.loan no as load_id , amt
from brower , loan
where borrower.loan_no = loan.loan_no
```

* It will be not stored in mem.
  It exists only for query (scope = query), Hence not for others

Tupples Variables:

Tuple var are defined in the form clause via use of 'as' clause.

```
Select ---
from borrower as T, loan as S
where T.loan_no = S.loan_no
```

① Distinct:

```
select distinct T.branch_name
from branch as T --
```

String Operations:

% - for substring (_) The _ char matches on char

• Find name of customer whose street include subs ma

```
select customer_name
  from customer
where customer_street like '% Main %'
```

• Supports variety of string op:
  Concate '||'

FECHA

Ordering & display of Tuples

```
select *
 from customer
 where borrower-loanN = loan.loan-N and
   branch_name = 'per'
   orderby customer-name   or desc
```

*Default Ascending

Set operations : (performs on tupples)

Union, Intersect, except (Not)

① (slect cust-name
   from depositor)
   union
 (select cus-n from borrower)

② -- " --
   Intersect
   -- " --

② -- " -- except -- " --

Aggregate funct^n:

① Avg - Average
② min - minimum value
③ max - maximum
④ Sum - sum of values
⑤ Count - number of values

FECHA

[Note - funct^n & procedure diff)
funct^n always return procedure need not be]

① select avg (balance)
  from --
  where --

② select count (*)
  from customer

③ select count (distinct name)

④

Group by -

```
select bN, count (distinct name)
 from dipositors, acc
 where depo.accno = acc.accNO
 group by branch_name;
```

* Along with groupby we use having *

```
select branch-name, avg (balance)
 from account
 group by branch-name
 having avg (balance) >1200
```

3-SQL

FECHA 9 Aug 2024

Null values : ≠ 0

will no be used in any numeric value.

N OR T = T	N And T = ~~U~~ N
N OR F = N	F U = F
N OR N = N	U U = U

not U = U

Nested subqueries :

Customers having both an acc & loan at bank

```
select di name
from borrower
where customer name (in) (select name from depositor)
```

in → contains

-'-

not in

All can be done by set theory also.

FECHA

cust both loan & P —

```
select name
from b, l
wher b.loan.no = l.loan.number
and branch = 'P' and
(branch.name) in (select branch.name)
—
```

Q. All branch having greater asset than some branch located in B (self comparison Hence virtual Table)

```
select T.branch_name
from branch as T , branch as S
where T.assets > S.assets and
S.branch_city = 'B'
```

Same using > some clause

```
select branch-name
from branch
wher assets > some
(select assets
from branch
where branch_city = 'B')
```

some ⇒ atleast one / more than one

Q name of all branches having greater asset

—
where assets > all
—

exist → returns true if non empty
not exists

Q cust have an acc at all branch in B

```
select S.name
from depo. as S
wher not exists
( select name
   from branch
   where city = 'B')
   except
select R.branch
from depo. as T , acc as R
wher T.no = R.no &
   S.name = R.name)
```

Test for Absence of Duplicate (unique)

cust at least two acc at P branch

```
select T.name
from depositor as T
wher not unique
(
   select R.name
   from acc, der as R
   wher T.name = R.name and
   R.accNO = acc.accNO and
   acc.branch = 'P')
```

variable outer level ⇒ correlation variable.

Derived Relations :

Q Find avg acc bala > 1200

```
select name, avg
from ( select name, avg(balance)
   from acc
   group by name)
   as branch-avg (name, avg)
   where avg > 1200
```

With clause:

All acc with max balance

variable
with max_balance (value) as
select max(balance)
from acc
select acc_no.
from acc, maxbalance
where acc.balance = max_balance.value

Q All branch where Total deposite is > avg of all total acc deposits at all branch

with branch_Total (name, value) as
select name, sum(balance)
from account
group by name
with branch_total_avg (value) as
select avg (value)
from branch_total
selet name
from branch_total, branch_total_avg
where branch_total.value >= branch_total_avg.value

Views: creates Temp table.
provide mechanism to hide certain data from views of certain user

create view V as < query expression >

A view of bram & their cust

create view all_cust as
(select name, cust
from
where acc.no = ~)
union
(select branch, cust
frm
~)

View defined by another view.

Modificat° - Delete:
Insert table to table.

all cust from perisidge branch:
select cust
from all_custmer
where branch_name = "perrysidge"

Delete all acc at every branch located in city 'Needham'

```
delete from acc
where branch_name in (select branch_name
                      from branch
                      where city = 'Needham')
```

Q Delete the record of all accounts with balance below the avg ath bank.

```
select delete from acc
    where balance < (select avg(balance)
                     from acccoun )
```

after modification aggregate functn gets affected.

Q provide as gift for all loan cust

with update → set & comes → order matters alot
case statement

If you insert in veiew, will not get inserted into original

⋈

JOINED RELATION

- Key:

Let $K \subseteq R$

K is superkey of R if value of K are sufficient to identify a unique tuple of each possible re~

K is candidate key if K is minimal.

Foreign key

Cartisian product:

$r \times s$

$r \times s = \{ tq \mid t \in r \text{ and } q \in s \}$

- Natural JOIN

$r \bowtie s$

$R = (A, B, C, D)$

$S = (E, B, D)$

Res schema = (A, B, C, D, E)