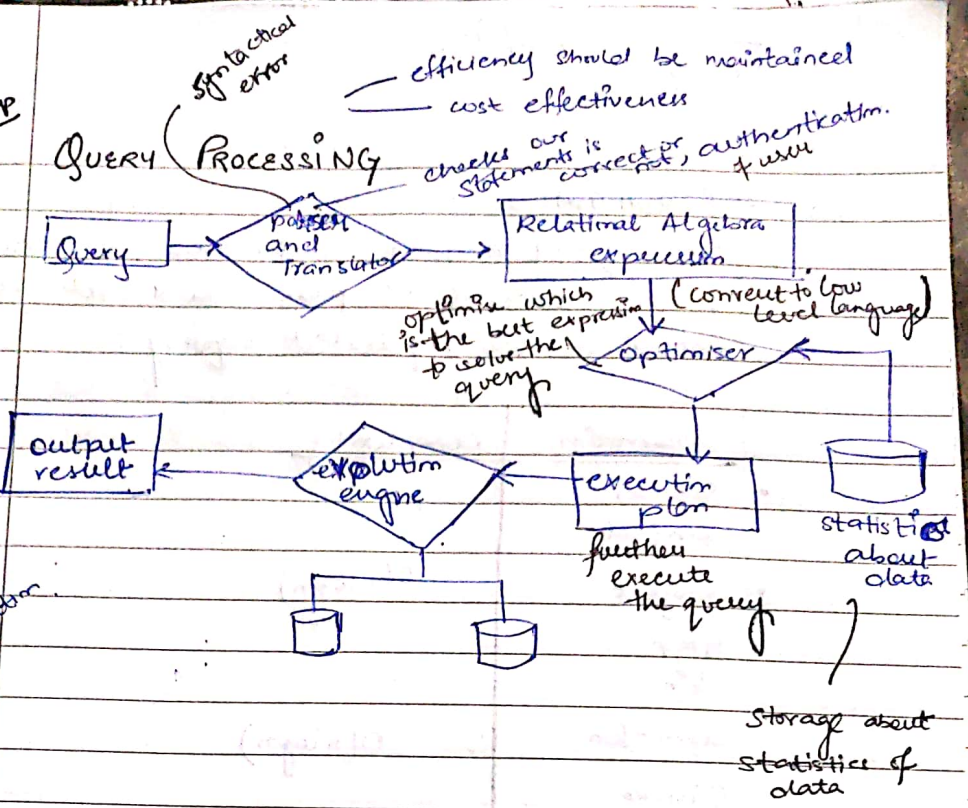
ributes of a
mentation
by columns

ich is created

reating a
ust include
parents
fragments

fragmentation
is achieved
which is

V. Imp




- It is a 3 step process that transforms a high-level query into an equivalent and more efficient lower level query which is relational algebric expression

PARSER & TRANSLATOR

- checks the syntax and verifies relation.
- translate the query into an equivalent relational algebraic expression

OPTIMISATION

generates an optimal evaluation plan with lowest cost for the query plan.

EVALUATION

the query execution engine takes a evaluation plan execute the plan and returns the answer of the requested query.

| operation | complexity |
|---|---|
| • Select, project | O(n) |
| • project | O(n log n) |
| • Group | |
| Join | |
| Semi-Join | O(n log n) |
| Division | |
| Set operators | |
| • cartesian product | $O(n^2)$ |

emp(empno, ename, title) ; asg(eno, pno, resp, dur)

Select ename from emp, asg where emp.eno = asg.eno and dur > 37

Que find the name of employee who are managing a project and duration are more than 37 hours.

$\Pi_{ename}(\sigma_{DUR>37 \wedge emp.eno = asg.eno}(emp \times asg))$

or $\Pi_{ename}(emp \bowtie_{eno}(\sigma_{DUR>37}(asg)))$;

Que emp(eno, ename, title) ; asg(eno, pno, Resp, dur)

site 1 : Asg 1 = $\sigma_{eno \leq E3}$ (asg);

site 2 : Asg 2 = $\sigma_{eno > E3}$ (asg);

site 3 : emp 1 = $\sigma_{eno \leq E3}$ (emp);

site 4 : emp 2 = $\sigma_{eno > E3}$ (emp);

emp ⋈ eno asg

Join — on basis of

* cost applied only in accessing not in comparing

V. imp

10 marks

# QUERY PROCESSING

equal distribution

emp (eno, ename, title);

asg (eno, pno, resp, dur);

emp = 400 records

asg = 1000

- 1 unit cost to access the records
- transfer cost = 10 unit

$\pi_{ename}(emp \bowtie_{eno}(\sigma_{dur>37}(asg)))$

site 5

result = emp1 ∪ emp2

20 records where duration > 37 hours

Transfer cost

site 3

10×10=100

10+10×1 = 20

ac.

10×10=100

10+10×1=20

site 4

emp1 = emp1 ⋈ eno Asg1

already have 10 records

emp2 = emp2 ⋈ eno Asg2

TC = 10×10 = 100

TC = 10×10 = 100

transfer

site 1

site 2

10

Asg1 = σ dur > 37 (asg)

500 records

10

Asg2 = σ dur > 37 (asg)

500

10×1

Ac. = 10

access cost = 10×1 = 10

200 + 40 + 200 + 20

= 460 units.

very efficient as cost is less

site 5

result = (emp1 ∪ emp2 ⋈ eno (σ dur > 37 (asg1 ∪ asg2)))

500×1

20

Ac

1000

total records

500×10

500×10

200×10

20×10

asg1 site1 500

asg2 site2 500

emp1 site3 200

emp2 site4 20

4000

10000

1000

8000

23000

400×1×20×1

# LAYERS Of QUERY PROCESSING

query is allocated globally not locally

Query on Global relations

high level lang converted to algebraic exp

Query decomposition ← Global schema

Algebric query on global relations

control site

Data localization ← fragment schema

algebraic exp is breaked and localised on diff sites

Algebraic query on fragments

global optimization ← Allocation schema

Dist. Query execution

Distributed execution } local site result is generated locally

PHASES

- Normalization
- analysis
- Elimination of redundancy
- rewriting

# NORMALIZATION

- $P_1 \wedge P_2 \iff P_2 \wedge P_1$
- $P_1 \vee P_2 \iff P_2 \vee P_1$
- $P_1 \wedge (P_2 \wedge P_3) \iff (P_1 \wedge P_2) \wedge P_3$
- $P_1 \vee (P_2 \vee P_3) \iff (P_1 \vee P_2) \vee P_3$
- $P_1 \wedge (P_2 \vee P_3) \iff (P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
- $P_1 \vee (P_2 \wedge P_3) \iff (P_1 \vee P_2) \wedge (P_1 \vee P_3)$
- $\neg (P_1 \wedge P_2) \iff \neg P_1 \vee \neg P_2$
- $\neg (P_1 \vee P_2) \iff \neg P_1 \wedge \neg P_2$
- $\neg (\neg P) \iff P$

TYPES OF NORMALISATION

- Conjunctions Nf is conjunction($\wedge$) of disjunction($\vee$)
- Disjunction ($\vee$) Nf

$\Rightarrow (P_{11} \vee P_{12} \vee \dots P_{1N}) \wedge (P_{m1} \vee P_{m2} \vee \dots \vee P_{mn})$

$\Rightarrow (P_{11} \wedge P_{12} \wedge \dots P_{1N}) \vee (P_{m1} \wedge P_{m2} \wedge \dots \wedge P_{mn})$

emp(eno, ename, title) ; asg(eno, pno, resp, dur)

Que find the name of employees who have been working in project $P_1$ for 12 or 24 months.

select ename from emp, asg where emp.eno = asg.eno and asg.pno = 'p1' and dur = 12 or dur = 24

IN CNf -

emp.eno = asg.eno $\wedge$ asg.Pno = 'P1' $\wedge$ (Dur = 12 $\vee$ Dur = 24)

conjunction of disjunctn

IN DNf

(emp.eno = asg.eno $\wedge$ asg.Pno = 'P1' $\wedge$ Dur = 12) $\vee$ (emp.eno = asg.eno $\wedge$ asg.pno = 'P1' $\wedge$ Dur = 24)

disjunction of conjuctin

NORMALIZATION

The query may be complex so to normalize this complex query into an easy form there are two normal forms conjunctive and disjunctive
- one giving priority to ∧ (and) and other to ∨ (or)

ANALYSIS-

checks the query is correct or not. syntax is correct or not

(ii) QUERY GRAPH
(conditions and joins are checked)

(iii) JOIN GRAPH (condition cannot be identified)
(checks where valid joins are applied or not)

- This enables the rejection of normalized queries for which further processing is either impossible or unnecessary. rejection maybe because of type incorrect or semantically wrong data (wrong datatype)

wrong cases→

- attribute or relation name not defined
- attribute having wrong input in such cases query is simply returned to user with an explanation.

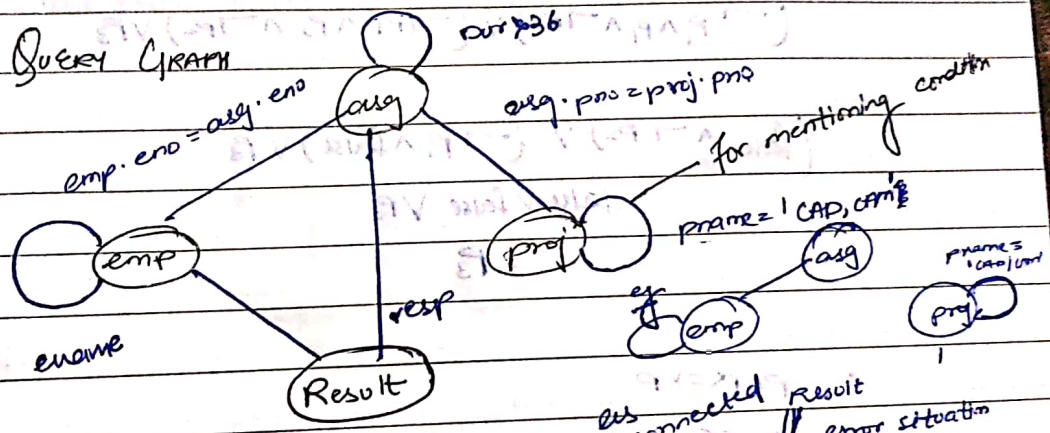
proj (pno, pname)

Que find the name and resp of programmers who have been working on CAD/cam project for more than 3 years

Select ename, resp from emp, asg, proj where emp.eno = asg.eno and asg.pno = proj.pno and pname = 'CAD/CAM' and dur > 36 and title = 'programmer';
(months)

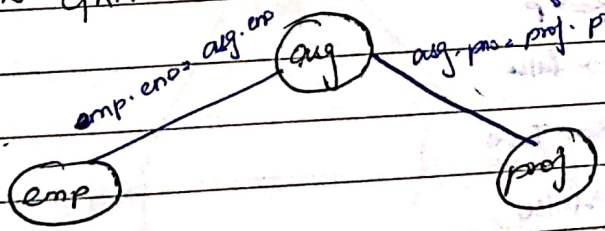
imp

Query Graph




as disconnected Result // error situation

Join Graph




check nodes only.

matrix YOUTH SERIES

# ELIMINATION OF REDUDANCY

Query

select title from emp wher ( NOT (Title = 'programmer') AND (Title = 'programmer' or title = 'electrical engg') and NOT (Title = 'elect engg')) OR ename = 'James';

same.

select title from emp where ename = 'James';

$(\neg P_1 \wedge (P_1 \vee P_2) \wedge \neg P_2) \vee P_3$

$(\neg P_1 \wedge ((P_1 \wedge \neg P_2) \vee (P_2 \wedge \neg P_2)) \vee P_3$

$(\neg P_1 \wedge P_1 \wedge \neg P_2) \vee (\neg P_1 \wedge P_2 \wedge \neg P_2) \vee P_3$

$(false \wedge \neg P_2) \vee (\neg P_1 \wedge false) \vee P_3$

$false \vee false \vee P_3$

$= P_3$

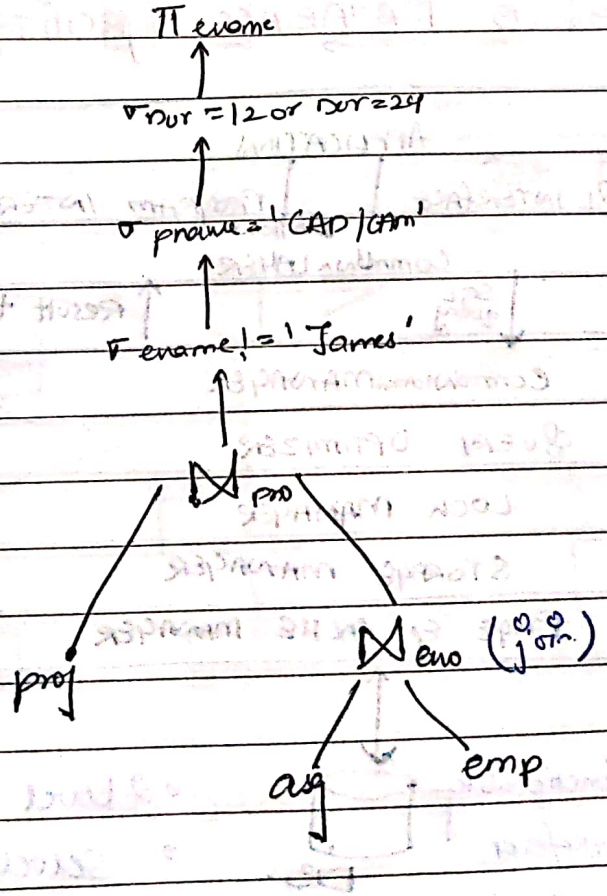
- $P \wedge P \Rightarrow P$
- $P \vee P \Leftrightarrow P$
- $P \wedge True \Leftrightarrow P$
- $P \vee false \Leftrightarrow P$
- $P \wedge false \Leftrightarrow false$
- $P \vee True \Leftrightarrow True$
- $P \wedge \neg P \Leftrightarrow false$
- $P \vee \neg P \Leftrightarrow True$
- $P_1 \wedge (P_1 \vee P_2) \Leftrightarrow P_1$
- $P_1 \vee (P_1 \wedge P_2) \Leftrightarrow P_1$

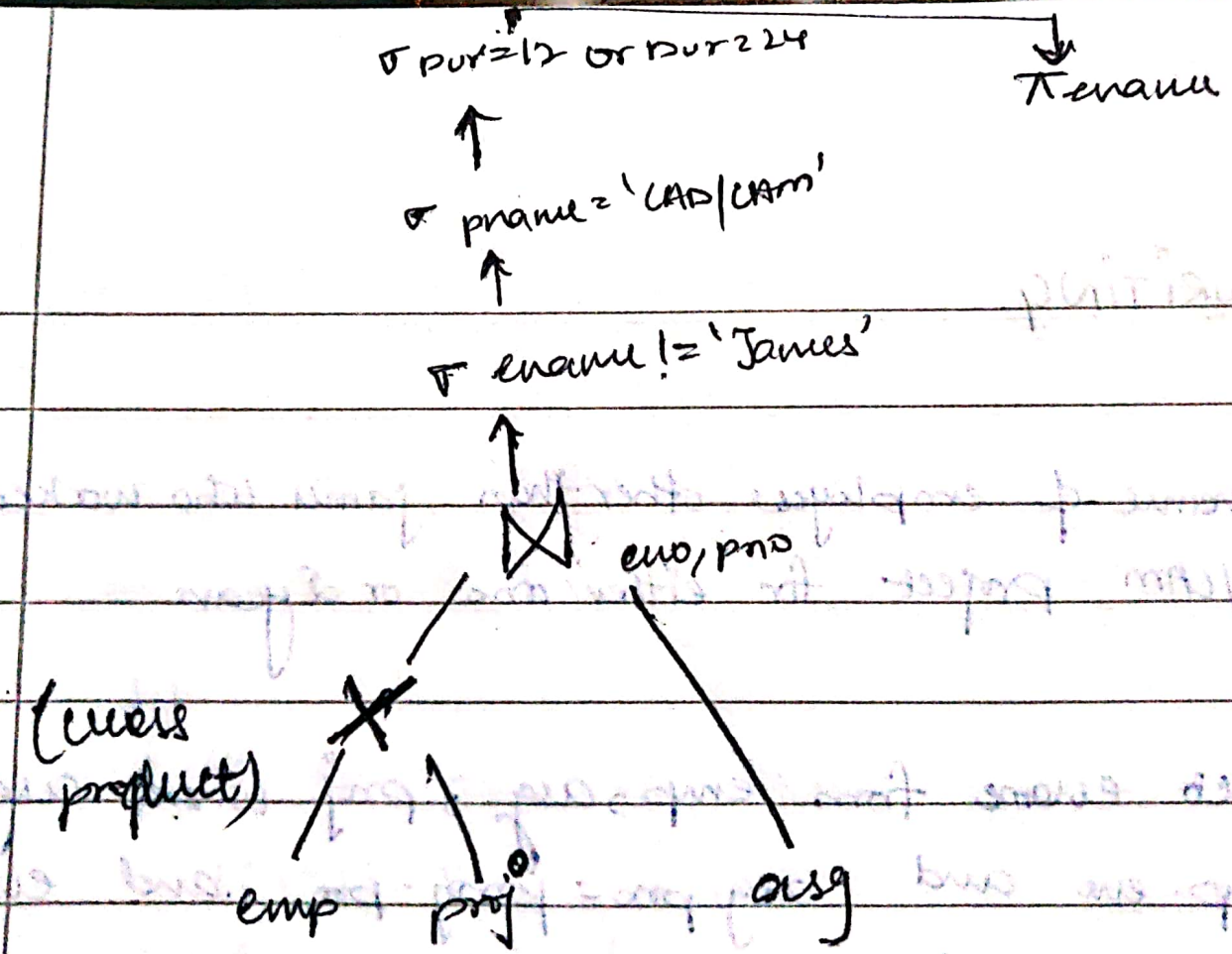
# REWRITING

Query → find name of employees other than James who worked on CAD/CAM project for either one or 2 years

→ select ename from emp, asg, proj where asg.eno = emp.eno and asg.pno = proj.pno and ename != 'James' and pname = 'CAD/CAM' and dur = 12 or dur = 24;

(1)






(a)

they are the temporary storage space

## CURSORS

to display multiple records on output screen

TYPES → Implicit, Explicit

Implicit

through variable displaying a single value (x multiple records)

eg select empno into e
from emp
where ename='SMITH';

Not

%found (if record exists with the table or DB)

%Notfound (if record doesn't exists)

%Rowcount (update sal set sal=sal+500 from emp; (count the no. of records which are updated)

%Isopen (if cursor is not closed (when a cursor is created it needs to be closed) open then value is True

| ATTRIBUTE | REASON | USE |
|---|---|---|
| • %found | • the return value is true if dml operation affect atleast one row. otherwise it is false. | SQL %FOUND |
| • %Notfound | • True if no row was selected updated, inserted or deleted otherwise false | SQL %NOTFOUND |
| • %Rowcount | • returns the no. of rows affected by the DML operation | SQL %ROWCOUNT |
| • %Isopen | • returns True if cursor is open otherwise false | SQL %ISOPEN |

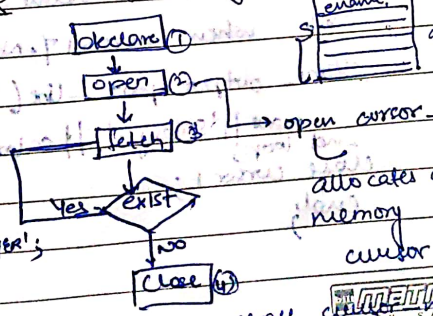
filtered out data and put in cursor is known as ACTIVE SET

- cursors are used to display multiple records on the screen. A cursor act as a pointer to the records in database table or a virtual table represented by the result of a select statement.
- A cursor is a temporary work area created in the system memory when SQL statement is executed a cursor can hold more than one row but can process only one row at a time. The set of rows the cursor hold is called the active set. Imp

- IMPLICIT CURSOR
- When DML statements like insert, update, delete and select statements are executed implicit statements are created to process these statements. Oracle provides few attributes for implicit cursor to check the status of DML operation

- EXPLICIT CURSOR

Declare
cursor cursor_name is
select ename from
emp where job='MANAGER';




only declare cursor name & will not go to cursor.




open cursor_name;
allocates a memory space to cursor.
close cursor_name;

- when a cursor is created there is a pointer which points to a single value at a time

fetch
```
Declare
name emp.ename%type;
fetch cursor_name into name;
```

eg
```
Declare
name emp.ename%type;
salary emp.sal%type;
Begin
fetch cursor_name into name, salary;
```

eg
```
Declare
eid emp.empno%type;
name emp.ename%type;
salary emp.sal%type
cursor cursor1 is select ename, empno, sal from
emp;
Begin
open cursor1;
loop
fetch cursor1 into name, eid, salary;
Exit when cursor1%Notfound;
dbms_output.put_line('eid='||eid||'name='||
name||'salary='||salary);
end loop;
close cursor1;
end;
```

cursor%Rowcount);

matrix

- The main task of cursor is selection

* if we are performing delete operation after cursor is created then the data from the cursor won't be deleted as we are performing delete operation on the table when cursor already contain the data we are not deleting from the cursor

Query: write a prog. to increase sal. of emp: the emp with post clerk will get 8% increment and the post salesman will get 5% increment

Query: write a prog to update the sal of employee who are working as managers.

Declare
cursor c1 is select job, sal from emp;
Begin
open c1

# DATA WAREHOUSE

- collection of different types of data
- a place where products are placed
- inside warehouse it is not necessary to keep data of same type
- data is uniquely identified inside warehouse

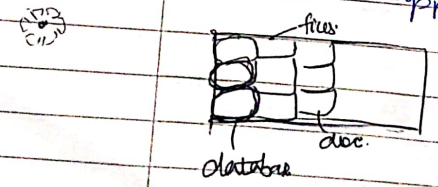
we have historical data.

- a database stores one type of data eg for a college.
- If i delete data from database it doesnot mean it will be deleted from the warehouse.

warehouse can contain data of many long years whereas database will occupy a lot of memory to store it.

- A data warehouse is a subject oriented, integrated, time variant and non volatile collection of data in support of managements decision making process.

- SUBJECT ORIENTED → organised around major subject such as customer, sales and products.
- It focuses on modelling and analysis of data for decision making
- It excludes data that are not useful in the decision support process.



Note - data never gets deleted from data warehouse.

- INTEGRATE - constructed by integrating multiple heterogeneous data sources
  - relational dB, flat files, online transaction records. (files other than dB file like doc, image, excel sheet(s))
- TIME VARIANT - provide information from a historical prespective.
- NON VOLATILE - no updation occur in warehouse.

- OPERATIONAL DBMS - to deal with current data (transaction)
  - ↳ OLTP (Online Transaction processing)
- datawarehouse is also known as OLAP (Online Analytical processing) — deals with historical data

| OLTP | OLAP |
|---|---|
| • professionals are clerks, IT professionals | • These are knowledge workers who perform the analysis data. |
| • deals with day to day transaction | • helps in decision support and deals with historical data |
| • application oriented. (isolated from each other) | • subject oriented. (use like stocks) |
| • In OLTP, data is isolated. (2d model) | • there is historical summarised multidimensional integrated (3d model) data. |

| | |
|---|---|
| • single queries are executed | • complex queries are executed |
| • users are in thousands | • users are in hundreds |
| • DB size is from 100MB–GB | • 100 GB – 100 TB |

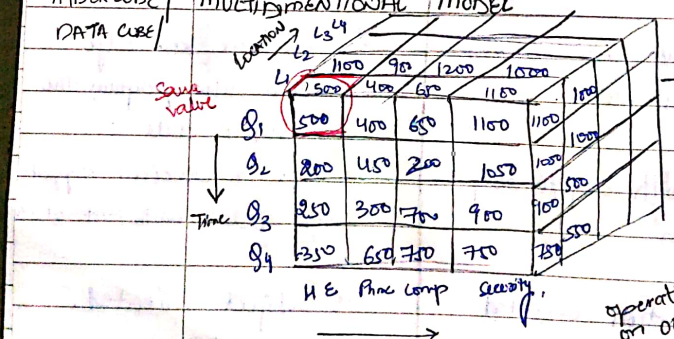
• OPERATION OVER OLAP (Online Analytical processing)

| | L1 | | | | L2 | | | | L3 |
|---|---|---|---|---|---|---|---|---|---|
| Time | HE | Phone | Comp | Security | HE | Phone | Comp | Sec | |
| $Q_1$ | 500 | 400 | 650 | 1100 | 1100 | 900 | 1200 | 1000 | |
| $Q_2$ | 200 | 450 | 700 | 1000 | 500 | 500 | 1100 | 1000 | |
| $Q_3$ | 250 | 300 | 700 | 900 | 600 | 1200 | 1050 | 500 | |
| $Q_4$ | 350 | 650 | 750 | 750 | 700 | 850 | 900 | 500 | |

2D MODEL

developed to make comparison easy.

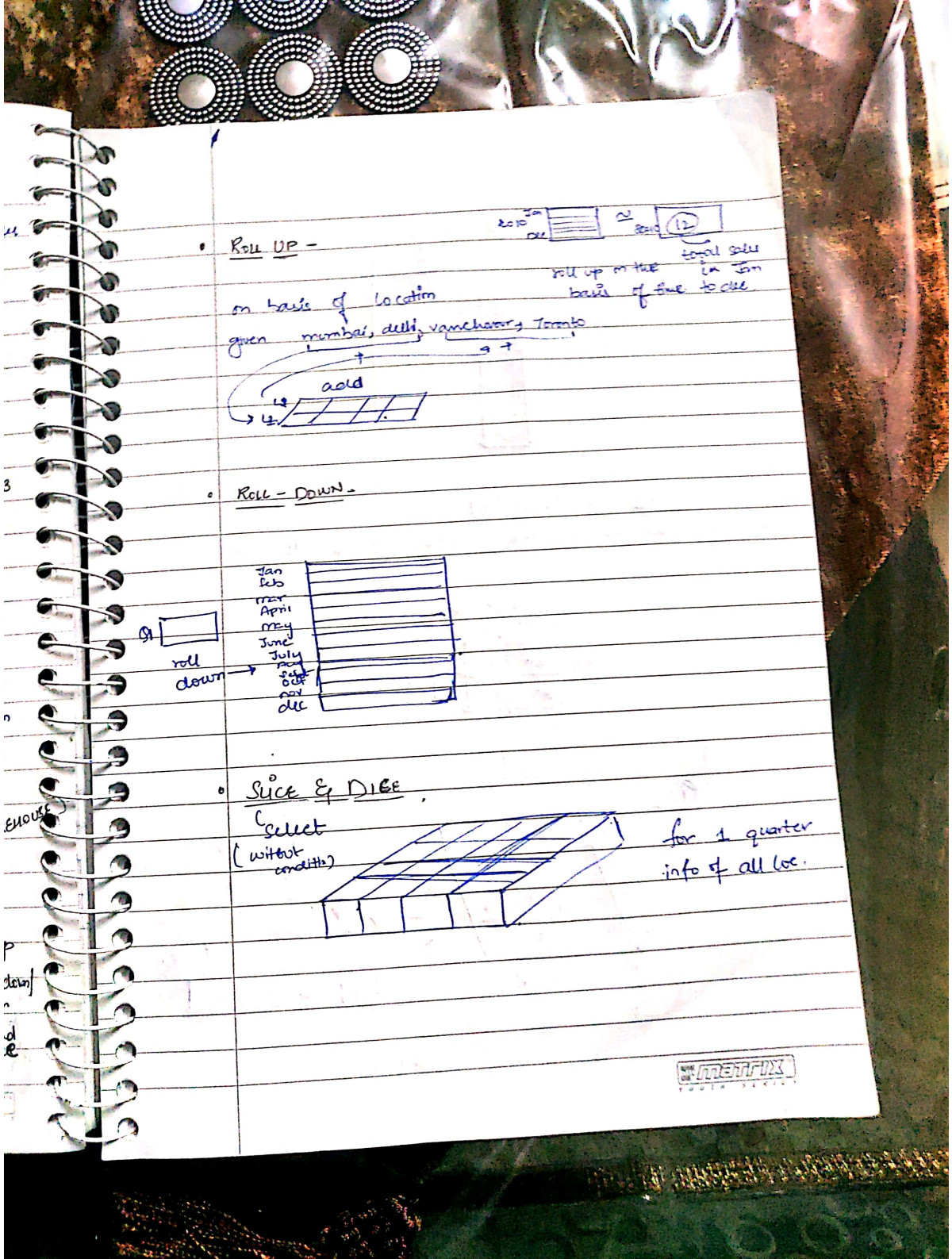
HYPERCUBE / DATA CUBE — MULTIDIMENSIONAL MODEL



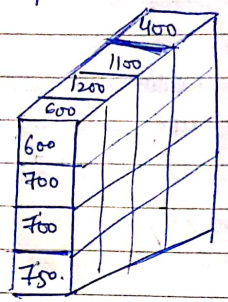

followed in OLAP (DATA WAREHOUSE)

operation on OLAP

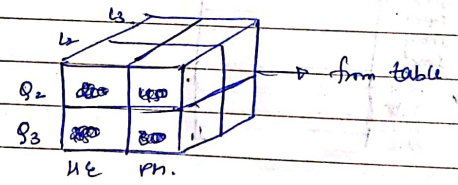
- Roll up / drill up
- Roll-down / drill down
- slice and dice
- pivot



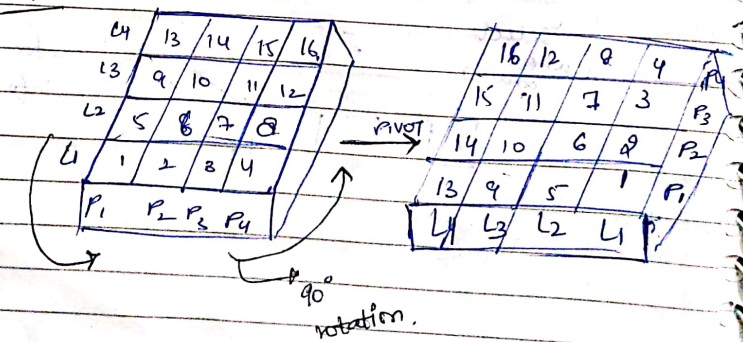
- for comp Ep for all loc.




- DICE (for condition)
for loc 2,3 and Q 2,3. → for ME., Ph.



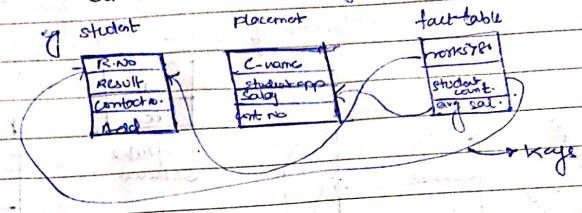

Pivot



multidimensional. one fact table is created only

Tables

dimensions — fact (condition) eg marks > 80%.




DIMENSION

~~FACT~~ TABLE

It consists of tuples of attribute such as student item (Item-id, brand, type)

FACT TABLE

It contains measures such as total count, avg sal, total profit and keys to each of the related dimension table.

v.imp.

imp / Numerical

TYPES OF SCHEMA

- STAR SCHEMA

DATA MODELING FOR DATA WAREHOUSE

A fact table in the middle connected to a set of dimension table

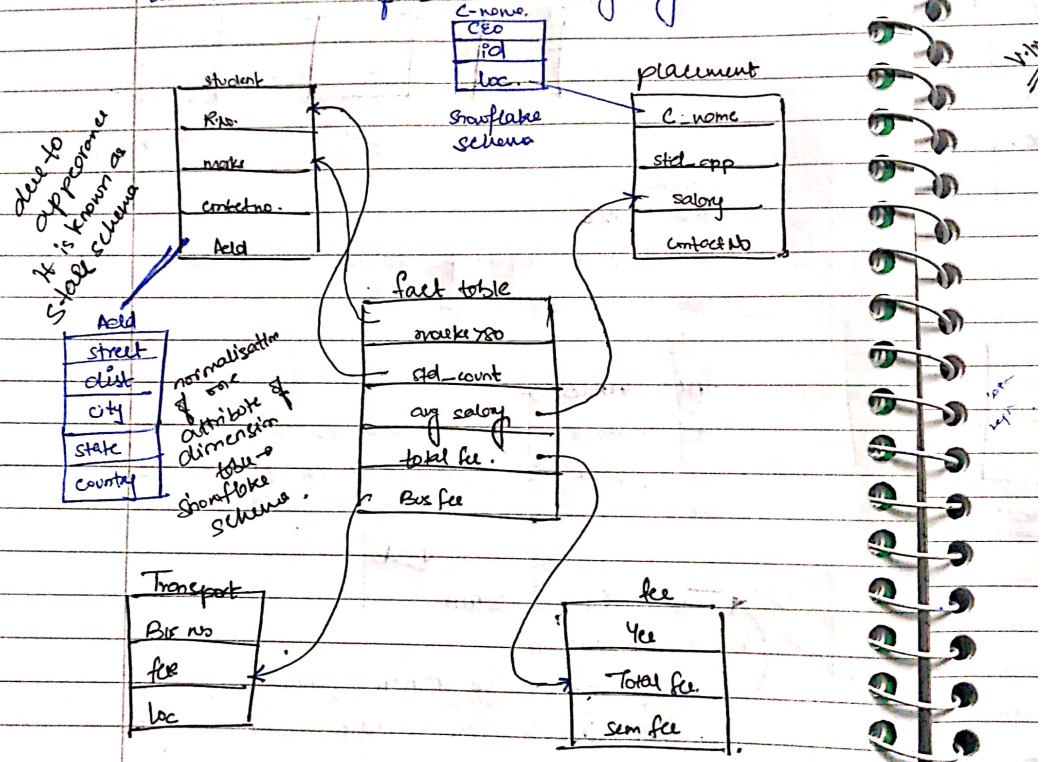
- SNOWFLAKE SCHEMA

refinement of star schema where some dimensional hierarchy is normalised into a set of smaller dimension tables forming a shape similar to snowflake.

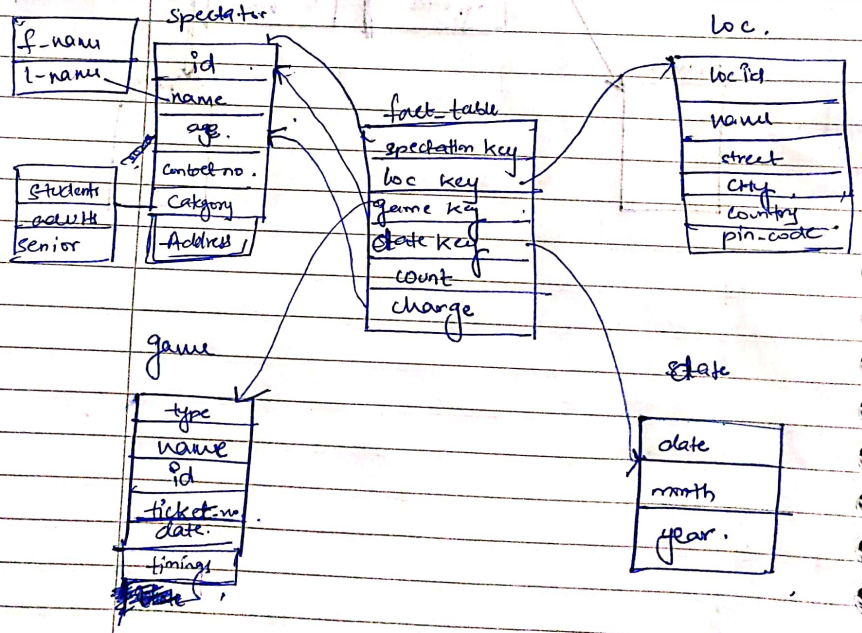
matrix YOUTH SERIES

- FACT CONSTELLATION

multiple fact table share dimension table viewed as a collection of stars called galaxy schema.



Q.e Suppose that a data warehouse consists of the four dimensions spectator, location, date and game, and 2 majors count & charge. where charge is the fees that the spectator pays when watching a game on a given date. spectator may be students, adults, senior with each category having its own charge rate. create star & snowflake scheme



FOR LOOP

- for loop_index (cursor)
  loop
  stmts ;
  end loop;

- Oleclare
  cursor cname is select ename, sal from emp;
  Begin
  for a in cursor1 / (select ename, sal from emp)
  loop
  dBms_output.put_line(a.ename||' '||a.sal);
  end loop;
  end;

PASSING PARAMETERS TO CURSOR

Declare
cursor emp_cur(p_id number, p_name varchar) is select empno, ename from emp where empno=pid and e_name= p_name;
Begin
open emp_cur (7831, 'ARC');
close emp_cur;
end;

matrix YOUTH SERIES

10 marks — Knowledge discovery in database.

# DATA MINING

fetching the data from data warehouse is called mining

selecting the data on the basis of user query and transform it in language understandable by users.

Evaluation & presentation.

analyzer.

Knowledge

patterns (users can easily understand) (bargraphs, documents, report)

eliminating the columns that are not required

Selection & transformation

data mining

data selected from warehouse

cleaning & integration

data warehouse

database

flat files

any doc files other than dB

to get data of the query entered by user from different tables.

market strategies and studies customer requirements and decide how the things are to be placed.

10 marks

# MARKET-BASED ANALYSIS

Imp 5 marks

→ association rules.

- support(s) = P(A ∪ B)
- confidence (c) = P(A ∪ B) / P(A)

| | |
|---|---|
| 1 | Bread, milk |
| 2 | Bread, Butter, Jam, eggs |
| 3 | milk, butter, Jam, cold drink |
| 4 | bread, milk, butter, Jam |
| 5 | Bread, milk, butter, cold drink. |

Support & confidence

A → B

eg {milk, butter} ⇒ {Jam}

$S = \frac{P(A \cup B)}{n} = \frac{2}{5} = 0.4 = 40\%$

A is compulsory, B is optional.

$C = \frac{P(A \cup B)}{P(A)} = \frac{2}{3} = 0.666 = 67\%$

decision depends upon organisation

- {milk, jam} ⇒ {Butter}

  $S = \frac{2}{5} = 40\%$ $\quad C = \frac{2}{2} = 100\%$

- {Butter} ⇒ {milk, jam}

  $S = \frac{2}{5} = 40\%$ $\quad C = \frac{2}{4} = 0.5 = 50\%$

  only butter

10 marks

## APRIORI ALGORITHM

(fetch the items that are mostly sold)

(given) min support = 2 (that only will sold) or more be kept in store)

5 short q transaction

| $T_{ID}$ | List of Items |
|---|---|
| $T_1$ | $I_1, I_2, I_5$ |
| $T_2$ | $I_2, I_4$ |
| $T_3$ | $I_2, I_3$ |
| $T_4$ | $I_1, I_2, I_4$ |
| $T_5$ | $I_1, I_3$ |
| $T_6$ | $I_2, I_3$ |
| $T_7$ | $I_1, I_3$ |
| $T_8$ | $I_1, I_2, I_3, I_5$ |
| $T_9$ | $I_1, I_2, I_3$ |

scan for count →

| Items | count. |
|---|---|
| $I_1$ | 6 |
| $I_2$ | 7 |
| $I_3$ | 6 |
| $I_4$ | 2 |
| $I_5$ | 2 |

$I_1$

compare with min support ≥ 2 otherwise reject

$L_2$

| Item set | Support count |
|---|---|
| $I_1, I_2$ | 4 |
| $I_1, I_3$ | 4 |
| $I_1, I_4$ | 1 |
| $I_1, I_5$ | 2 |
| $I_2, I_3$ | 4 |
| $I_2, I_4$ | 2 |
| $I_2, I_5$ | 2 |
| $I_3, I_4$ | 0 |
| $I_3, I_5$ | 1 |
| $I_4, I_5$ | 0 |

List of candidate from $L_1$

X (Rejected)

$L_3$

| Itemset | count |
|---|---|
| $I_1, I_2$ | 4 |
| $I_1, I_3$ | 4 |
| $I_1, I_5$ | 2 |
| $I_2, I_3$ | 4 |
| $I_2, I_4$ | 2 |
| $I_2, I_5$ | 2 |

List of candidate from $L_2$

X
X
X

Compare with min support

$L_3$

| item set | count |
|---|---|
| $I_1, I_2, I_3$ | 2 |
| $I_1, I_2, I_4$ | 1 |
| $I_1, I_2, I_5$ | 2 |
| $I_1, I_3, I_4$ | 0 |
| $I_1, I_3, I_5$ | 1 |
| $I_1, I_4, I_5$ | 0 |
| $I_2, I_3, I_4$ | 0 |
| $I_2, I_3, I_5$ | 1 |
| $I_2, I_4, I_5$ | 0 |
| $I_3, I_4, I_5$ | 0 |

X
X
X
X
X
X
X

| Item | count |
|---|---|
| $I_1, I_2, I_3$ | 2 |
| $I_1, I_2, I_5$ | 2 |

Scanned by CamScanner

confidence value is given eg c > 50%.

(iii) for x = $I_1, I_2, I_3$.

$\{I_1, I_2\} \rightarrow \{I_3\}$ c = 2/4 = 50%

$\{I_1, I_3\} \rightarrow \{I_2\}$ c = 2/4 = 50%

$\{I_2, I_3\} \rightarrow \{I_1\}$ c = 2/4 = 50%

very unlikely

$\{I_1\} \rightarrow \{I_2, I_3\}$ c = 2/6 = 0.33 ✗

$\{I_2\} \rightarrow \{I_1, I_3\}$ c = 2/7 = 0.29. ✗

$\{I_3\} \rightarrow \{I_1, I_2\}$ c = 2/6 = 0.33 ✗

for x = $I_1, I_2, I_5$

$\{I_1, I_2\} \rightarrow \{I_5\}$

$\{I_1, I_5\} \rightarrow \{I_2\}$