Database : organised collection of data so that it can be easily accessed and managed.

DBMS ( Database Management System)

Database : collection of related data / set of data

Management : storage space / structure for storage of information

mechanism for manipulation → i) Insert, ii) update, iii) delete } program

file is also collection of data.

⊛ Diff between file and database (advantages of DBMS) p-4

1) Redundancy : (duplicate item)

| Ac. no. | Balance | Phn |
|---|---|---|
| | | |

Table 01

| Ac no | Father | city | phn |
|---|---|---|---|
| | | | |

Table 02

DBMS এ redundancy completely remove করা যায় না।

যেমন Table 02 এ Ac no ছাড়া identify করা সম্ভব হয় না। তাই

আবারো Ac. no. add করে Link করা হয়েছে।

কিন্তু file system এ redundancy সবল।

2) Consistancy : (সামঞ্জস্য) / various copies of same data may no longer agree

DBMS এ কোনো element change করলে, ঐ element টি database এ

যত জায়গায় থাকে সব জায়গায় change হবে। কিন্তু file এ তা হয় না।

যেমন file system এ table 1 এ phn no. change করলে এবং table এ change হবে না।

3. Atomicity: (either all or none)

A → B

A থেকে B তে transaction হয় পুরোটা ও হবে অথবা হবে না। partially কখনও হবে না।

4. Accessibility: data easily access/search করা যায়।

5. Security: DBMS এ highlevel এর security assure করা হয়। encryption, password, etc

6. Data integrity: যে data গুলো store করা হবে সেই data গুলোকে কিছু column/row মেনে চলতে হবে। যেমন: যেসব employee এর age 25 এর নিচে তাদের info store করা হবে। যেসব value এই constraints follow করবে না

# View of Data (1.3)

View Level of Abstraction (কোন level পর্যন্ত data access করতে পারবে

lowest level of abstraction → physical level

logical level

view level

**physical level** : (How data stored) অন্যরা access করতে পারবে না।

শুধুমাত্র designer জানবে। sale list কিভাবে create হচ্ছে?

**Logical level**: institute internally (what data and what relationship)

discount in sale (relatively simple structure)

**View level**: (just overall view) salesman

Person

```
create table person
(acc_info integer (20),
        → physical
acc_balance integer (50))
  → logical
```

person

| acc_info | balance | |
|---|---|---|
| a | 500 | |
| b | 2000 | |
| c | 1500 | |
| d | | |

view level overall table টা a, b, c, d এগুলো logical level.

**Entity** is a distinguishable real-world object, thing, place, person or item

**Attribute:** describes the elementary features of an entity.

**Instances** → collection of data (actual content)

**Schema** → overall design of database. (structure)

int a, b; // schema | variable declaration

a=10, b=5; // instances | value of variable

Tools for describing data, data relationships, data semantics, consistancy constraints.

(1.3.3) **Data model:** Tools of designing database

→ E-R model (Entity relationship)

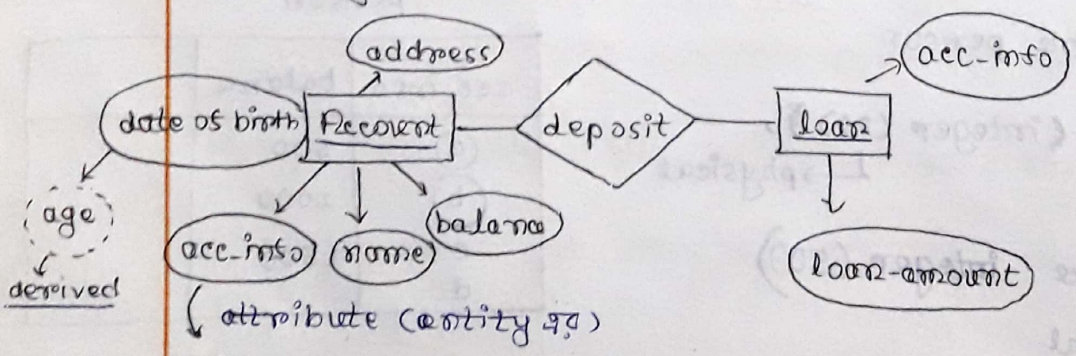
Relational model (table) ✓

## ER MODEL

Entity - R : Entity (object) acc, person) table name

দুইয়ের অধিক object এর মধ্যে relationship

graphical representation




attribute (entity এর)

domain { 101
102

E-R

Acc and loan এর মধ্যে relation deposit

rectangle view → object

◇ view (relation)

Entity → table name (acc, balance etc)

Entity → table name
Attribute → কলাম নাম (name, balance etc)
instances entity → কলামের value (Tasneem, 2000 etc)

Account

| acc-info | name | balance | address |
|---|---|---|---|
| | | | |
| loan | | | address |

RM : collection of table to present both data and the relationships among those data (record based model)

E-R : collection of basic objects (entities) and relat among those objects.

আগে E-R model, then relational model এ transform
flow table

**Attribute** (characteristics কে উনার depend করে) / column name

(i) simple / composite

(ii) single valued / multivalued

(iii) derived

person

| name | roll |
|---|---|
| T | 1 |
| A | 2 |

name (composit)
first middle last

balance (simple)
201

acc-id (single valued) | mobile no. (multivalued)

Account এ date of birth থাকলে, age derive করা possible

attribute এর value unknown হলে NULL বসানো হয়।

primary key

Key (identify করার জন্য) attribute or collection of attributes that uniquely identify an entity

(i) super key (যত possible way তে unique attribute তৈরি করা সম্ভব)

(ii) candidate key (A superkey without redundancy)

(iii) primary key (is a candidate key with no NULL values.

Super key : {acc_id} , {acc_name}

{acc_id, name} → redundent optional way

{name, phone}

{name, address, acc_id}

A set of attributes within a table whose values can be used to uniquely identy a tuple

optimise করে, যেটা একাই কাজ করতে পারে তাকেই নিয়ে নেওয়া সম্ভব।

Candidate key : optimised set (minimum set of attributes used to uniqely differentiate records. {acc_id}, {acc_na

primary key : single যেটা choose করা হয়।

{acc_id} or {acc_name}

Foreign Key : column that creates a relationships between two table.

## Query Language

(i) DDL (Data Definition Language) → allows one to define table, integrity constraint, create, constraint

(ii) DML ( " Manipulation Language) [enable users to access or manipulate data as organized by appropriate data mo

DDL
constraint
acc-info
→ not null

```
creat table ~~data~~ department
  (dept_name   char (20),
   building    char (20),
   budget      numeric (12,2));
```

plsql (for loop use করা যায়)

DML :

insert, delete, update

```
select instructor.name
from instructor where instructor.i_id = '1'
```

PHP  mysql

create Database University
stu , employee, teacher

create table student

```
( student_id {integer (20)},
  column_name
  student_name {varchar (20)});
```

Insert into student table name (column_)
values ((153001, 'aa'),
(153002, 'bb'),
(153003, 'cc'))

University
↓
student

| student_id | student_name |
|---|---|
| 1503001 | 'aa' |
| 1503002 | 'bb' |
| 1503003 | 'cc' |
| 1503004 | 'dd' |

Insert into student (student_id, student_name)
values (153001, 'aa'), (153002, 'bb'),

CREATE TABLE AS (SELECT * FROM old_table)

↳ create a table from existing table by copying existing table's columns.

new table

create table (student_info)

(student_roll, ~~student_result integer(20)~~)

as (select student_id

from school);

create table table_name as (select * from existing_table)

student_info

| student roll | cgpa | address |
|---|---|---|
| 1501 | | |
| 1502 | | |
| 1503 | | |

পুরাতন table এর column এর নাম

After table student_info

add (cgpa number (3,2)) ⟶ এটা value, দশমিকের পর ২টা 3.42

(age float (30),

modify (cgpa integer (10),

. . . . )

After table table_name

Drop column column_name

After table student_info

modify cgpa ~~rate~~ float (10),

" new column add "

student_id, ( 13003, cc ), distinct address.

select student_name from student

where student_id = 15003;

select * from student order by student_id; desc;

all

descending order.

update student

set address = 'z' where

student id = 1503004;

Delete from student; // where না দিলে সবগুলো delete

Delete from student where student_id = 1503001;

Drop student; // table delete

'student'

| student_id | student_name | address |
|---|---|---|
| | | |

* select count (DISTINCT student_name) from student

* course, section, teaches, takes, Prereq

| customer-id | customer-address | balance | interest |
|---|---|---|---|
| 1 | xyz | 500 | |
| 2 | yzz | 5000 | |
| 3 | zxy | 2000 | |
| 5 | bb | | |
| 7 | xyz | | |
| 8 | yzz | | |

main database change না করে শুধু interest part এ vary করবে।

(display হবে just)

select customer_id, interest * 0.5 Increased

from customer where

customer id = 3 AND id = 5;

OR

| customer_id | Increased / interest * 0.5 |
|---|---|

না লিখলেও হবে

কয়েক অধিক customer এর জন্য,

customer_id In {3,5,7,9}

NotIn {3,5,7,9} → এগুলো বাদ

range এর ক্ষেত্রে,

customer_id Not between 3 AND 10

id >= 3 && id <= 10

customer_id > 5 AND

( customer_address = 'X' or

customer_address = 'Y' )

The percent sign (%) represent zero, one or multiple character.

customer_address Like '%b%' ( যে যে জায়গায় b থাকবে তা দেখাবে )

→ Like '__%z' ; ( z এর আগে ২টা Letter থাকবে )

→ Like '%Y__' ; ( Y দিয়ে শুরু যেগুলো, সেগুলো দেখাবে )

→ Like '__z%' ; ( z দিয়ে শুরু )

The underscore sign (__) each represent one, single character.

Aggregate functions ( একসাথে যুক্ত করার জন্য

↳ sum ( balance )

→ MIN

→ MAX

→ AVG

→ count

sum :

select sum (balance)

"~~Total~~" from

customer; where

customer_id in {3, 5, 7, 9};

normally condition এর জন্য where

MIN/MAX : SELECT {MIN}(column_name) FROM table where condition

count : row no.

(All | Distinct) balance/

↳ (same নাম হলে দেখাবে না)

count (Distinct balance) from customer.

group by (প্রতি value তে যোগ করে দেখাবে)

select customer_id, balance from (online shopping

customer group by ~~balance~~ customer_id

| নাম/id | | |
|---|---|---|
| xyz | 1000 | // 500+500 |
| $\boxed{yzz}$ | 5100 | // 2500 + 500 + 200 |
| aa | 1200 | // |
| $\boxed{bb}$ | | |
| cc | | |

having customer_add = 'yzz' AND cus_add = 'bb';

'group by' এর সাথে 'where' use করা যায় না, সেজন্য having use করা হয়।

JOIN is used to combine rows from two or more tables, based on related column between them.

3C

Join (Inner Join)

customer

| c_id | name | address | phn no |
|---|---|---|---|
| 1 | a | aa | 001 |
| 2 | b | bb | 002 |
| 3 | c | cc | 003 |
| 4 | d | dd | 004 |

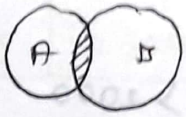
account

| c_id | acc_id | balance |
|---|---|---|
| 1 | 101 | 500 |
| 5 2 | 102 | 1000 |
| 3 | 103 | 1500 |
| 6 4 | 104 | 1000 |

loan

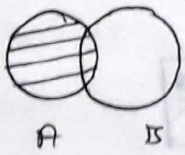
| acc_id | amount |
|---|---|
| 101 | 500 |
| 102 | 2000 |
| 103 | 1500 |
| 104 | 1000 |

**Join**: i) Inner join →



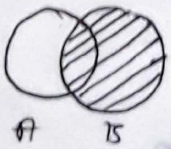

(column name same, data র same display করবে)

ii) Left join →

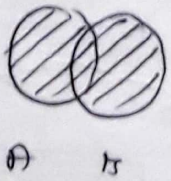



→ right র only এমন rowগুলো দেখাবে যাগুলো match হবে না সেগুলো NULL

iii) Right join →




iv) full join →




→ full join return all records when there is a match in either left (table1) or right table.

Left join

Left outer join

account

| c_id | name | addr | phn | acc_id | balance |
|---|---|---|---|---|---|
| 1 | - | - | - | - | - |
| 2 | - | - | - | Null | Null |
| 3 | - | - | - | - | - |
| 4 | - | - | - | Null | Null |

account র 2, 4 c_id নেই

(কোন table থেকে নেওয়া হবে, সেটাতে ambiguity create হতে পারে, তাই table name....

select customer.name,

account.balance from customer inner join

account on customer_id = account.c_id

| name | balance |
|---|---|
| a | 500 |
| c | 1500 |

where account.balance > 1000

| name | balance |
|---|---|
| c | 1500 |

left join

| name | balance |
|---|---|
| a | Null |
| b | Null |
| c | 1500 |
| d | N |

(1000 এর ছোট)

select customer.name, loan.amount as balance,
account.balance from customer inner join account

on customer.cid = account.c_id

inner join loan on account.acc_id = loan.acc_id

| name | balance | amount |
|---|---|---|
| a | 500 | 5000 |
| c | 1500 | 1500 |

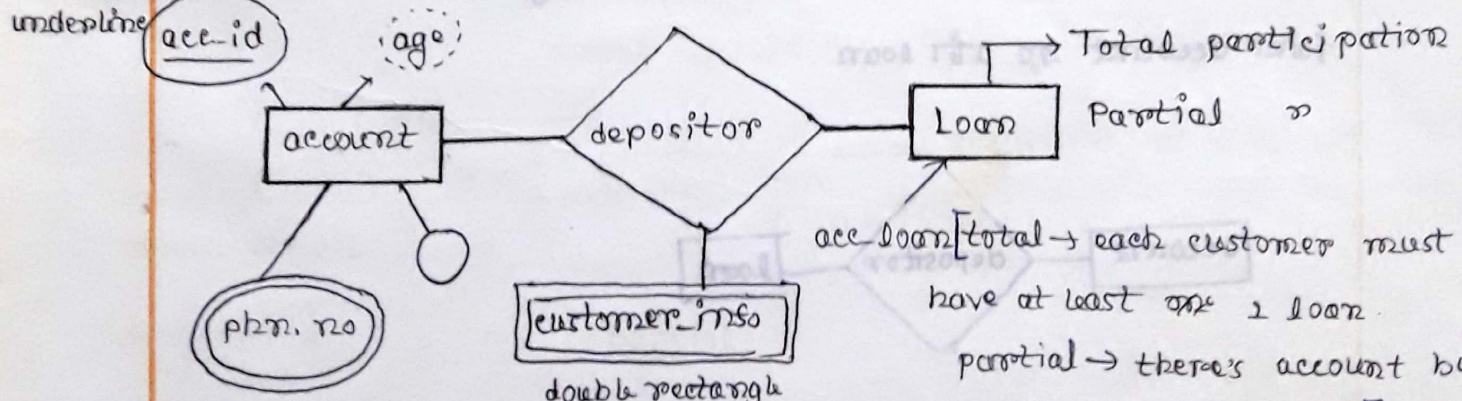
---

defines the conceptual view of a database.

relationship (binary, tarnary)

**Entity Model**

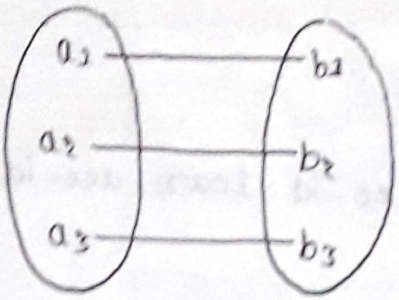
Participation constraint (শর্ত) (সীমাবদ্ধতা)




101 → 'যতজনের loan থাকবে
102 → total'
103
104

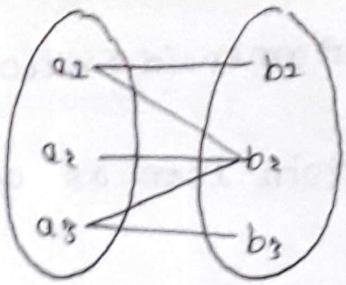
weak Entity set → no primary key
strong " " → primary key থাকবে।

Mapping cardinality (Entity র মধ্যে connection)



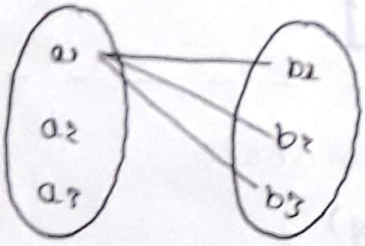

1 to 1 (total)

1টা customer এর 1টা loan



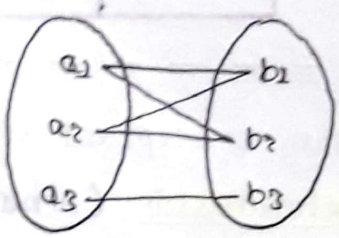

1 to many

1টা customer এর একাধিক loan



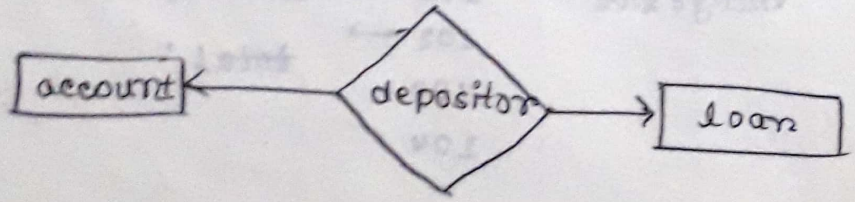

many to 1

joint account এর 1টা loan

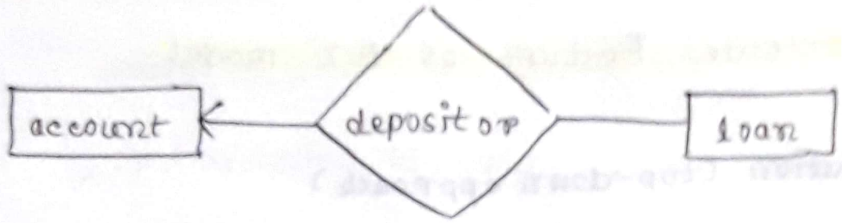



many to many

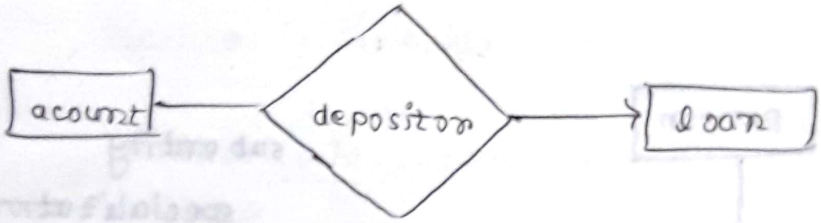
undirected (many to many)



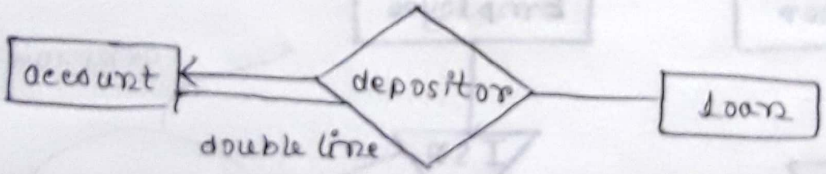

one to one




one to many




many to one




double line
total

Loan

(min) (max) → unlimited

1 to 1 হবে না ber at most নয়

( one to many)

atleast one ← 1 . . . *

zero or more ← 0 . . . * (partial)

1 . . . 1 (one to one)

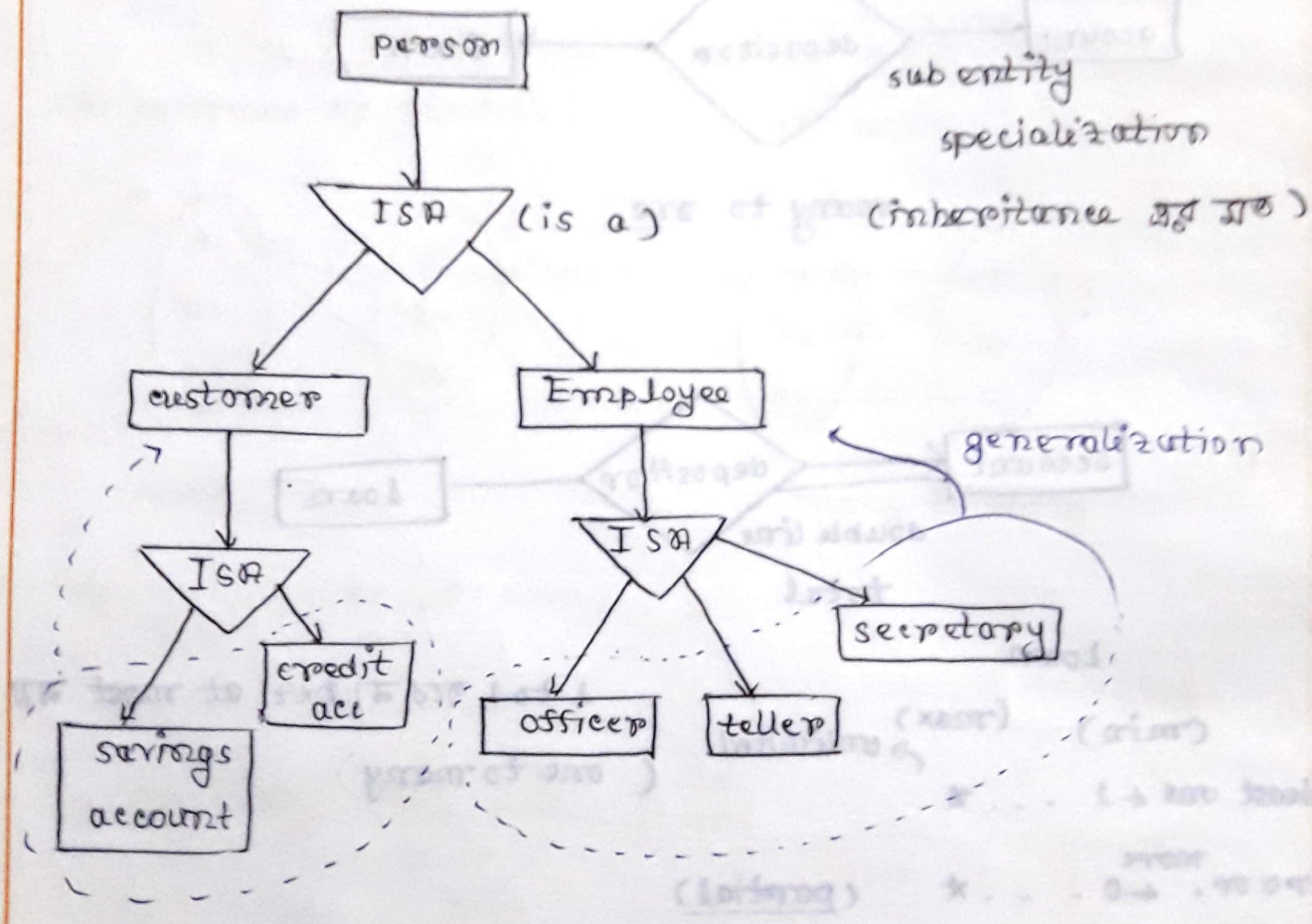
maximum
one

B ☐ entity exercise

# Extended Feature of E-R model

→ specialization (top-down approach)

→ Generalization (bottom up approach)



3D

course (c-id, title, d-name, credit)

Instructor (I-id, name, d-name, salary)

section (c-id, sec_id, semester, year, building, r-no, time-slot)

Teachers (i-id, c-id, sec-id, semester, year)

student (s_id, s-name, d-name, tot-credit, weight)

~~Takes~~ Taught (s-id, c-id, semester, year, grade)
Takes

Advisor (s-id, i-id)

① Find the name of the ~~teachers~~ instructor (teaches) instructors, the course-id and title (course) which they have taught. (course and teaches join)

② Find course name, credits running in semester 4. (section)

③ Find pair of student-id and ins-id where instruction is teacher and advisor of students also retrieve the correspond d-name (Inst and Advisor)

④ Find the title of course in CSE, that have 3 credits, show the student id, who have taken these credits along with

their total credits (taken)

Ans:

1. select ~~instructor, course~~

select ~~name~~,

select Instructor.name, ~~course.c_id~~, Teacher.c_id

course.title from Instructor inner join
on Instructor.I_id = Teacher.I_id
Teacher inner join course on Teacher.c_Id =
course.c_Id

(section.semester)

2. Select course.name, course.credit from course

Inner join semester on course.c_id = semester.c_id

Where section.semester = 4

(3) Select I.i-id, A.s_id, s.d-name from Instructor I inner join Advisor A on I.i-id = A.i-id inner join student s.s-id on

s.s_id = A.s_id

(4) select course.title, course.credits takes.s-id from

course inner join takes on course.c-id = student.c-id

total_credit

1. Select teaches.name

# Self Join

একটা table এর মধ্যেই info merge.

Employee

| e_id | e_name | supervisor id |
|---|---|---|
| 100 | AA | - |
| 101 | BB | 100 (AA) |
| 102 | CC | 101 (BB) |
| 103 | DD | 100 (AA) |
| 104 | EE | 101 (BB) |
| 105 | FF | 101 |

↓

| emp e_id | emp_name | sup_name | sup_id |
|---|---|---|---|
| 101 | BB | AA | 100 |
| 103 | DD | AA | 100 |
| 102 | CC | BB | 101 |
| 104 | EE | BB | 101 |
| 105 | FF | BB | 101 |

```
select e.e_id as 'emp_id', e.e_name as 'emp_name',
       e1.e_name as 'sup_name', e1.supervisor_id
from Employee e,
     Employee e1
where e.e_id = e1.supervisor_id
```

| emp_name | sup_name |
|---|---|
| BB | AA |
| DD | AA |
| CC | BB |
| EE | BB |
| FF | BB |

same city তে যারা থাকে তাদের display কর।

student

| s_id | s_name | s_city | phn no. |
|---|---|---|---|
| 1 | A | AA | |
| 2 | B | BB | |
| 3 | C | AA | |
| 4 | D | XY | |
| 5 | E | XY | |

display

| name | Name2 | city |
|---|---|---|
| A | C | AA |
| D | E | XY |

s1.s_id s2.s_id

where s1. city = s2. city

# Normalization

Redundant data: 150 → (K) একটা edit করলে
150 → (F)

**Primary key:**

Primary key থাকার পরও redundancy তো থাকলে, তা দূর করার জন্য Normalisation use হয়।

## Normalization

উদ্দেশ্য: redundancy মুক্ত করা, insertion, update and Deletion anomaly দূর করা।

(1NF) **1st Normal form**: কোনো repeating column থাকা যাবে না, একটি column এ child এর 200টা ~~column~~ value থাকতে পারবে না।

Personal Info

| RID | Name | Dhan | C1 | C2 | C3 | C4 |
|---|---|---|---|---|---|---|
| 007 | | | Ab | Bc | Ca | |

→ column এর redundancy (mobile no) repeating column থাকলে আলাদা করে ফেলতে হবে আলাদা table.

Rajar Bacha

| RID | serial | child |
|---|---|---|
| 007 | 1 | Ab |
| | 2 | Bc |
| | 3 | Ca |
| 008 | 1 | Abc |
| New ← 007 | 04 | Da |

* each table cell should contain a single value.

সমস্যা ২: composite primary key

Nonprime

Personal Project

| | NID | Project ID | Name | Dham | Contact No. | Project_Name |
|---|---|---|---|---|---|---|
| uniq pair | 1 | 1 | CA | Ab | 172 | ML |
| | 1 | 2 | CB | Ab | 172 | DL |
| | 1 | 3 | CC | Ab | 172 | |
| | 2 | 3 | | | | NW |
| | 3 | 3 | | | | NW |

Name, Dham, contact only NID এর উপর depend করে

2nd Normal form: All nonkey attributes must depend on full p.k.

Personal

| NID | Name | Dham | Contact No |
|---|---|---|---|
| | | | |

Project Info

| Project ID | Project Name |
|---|---|
| | |

Personal project (comp.

| NID | Project ID |
|---|---|

দুইটার উপর depend করে এমন column

## * Normalization:

inconsistant : একজায়গায় update হয়েছে, আরেকজায়গায় update হয় নি

To remove redundancy

To organize database

To illuminate anomaly (abnormal situation)

(insert, Delete, update)

To reduce the need to restructure tables as new applicati requirement arise.

* same person multiple job

employee NOT NULL

| e_id | e_name | e_address | e_dept_name |
|---|---|---|---|
| 1 | A | AA | CSE |
| 2 | B | BB update | EEE |
| 3 | C | CC | MTE (delete) |
| 4 | D | DD | ETE |
| 01 | A | AA | ETE |
| 2 | B | BB | ECE |
| 5 | E | EE | - |

প্রত্যেকটা আলাদা করে update করা

করতে গেলে পুরো row delete হবে এবং anomaly create হবে

new insert (Null value এর জন্য anomaly create হলো)

normalized

| e_name | e_address | e_id |
|---|---|---|
| B | BB | 2 |
| A | AA | 1 |
| C | CC | 3 |
| D | DD | 4 |
| E | EE | 5 |

every table contain single idea

~~table গুলোকে break করে ছোট ছোট table এ নিয়ে আসা~~

① Normal Form (1NF) → No repeating columns, atomic values

② Normal Form (2NF) → fully dependent on primary key

③ Normal Form (3NF) → Non transitivity.

email_id

| stu_id | s_name | s_add | email1 | email2 | email3 | email4 | phn1 | P 2 |
|---|---|---|---|---|---|---|---|---|
| 1 | A | AA | xxx | yyy | zzz | | | |
| 2 | B | BB | xyz | xzy | - | | | |
| 3 | C | CC | - | - | - | | | |

1st Normal Form

| s_id | s_name | s_add |
|---|---|---|
| 1 | A | AA |
| 2 | B | BB |
| 3 | C | CC |

| s_id | seq_no | email | phn no |
|---|---|---|---|
| 1 | 1 | xxx | |
| 1 | 2 | yyy | |
| 1 | 3 | zzz | |
| 2 | 1 | xyz | |
| 2 | 2 | xzy | |
| 91 | 4 | ??? | |

## 2nd Normal Form (fully depend on primary key)

| s_id | s_name | s_add | sub_id | marks | Teacher | Exam name | Total mar |
|---|---|---|---|---|---|---|---|
| 1 | A | AA | 2 | 70 | CSE | | |
| 2 | B | BB | 1 | 80 | EEE | | |
| 3 | C | CC | 4 | 90 | MTE | | |

prime: A

nonprime: B C D E F

(A) B C (D) E F

AD candidate key

nonprime গুলো prime এর উপর fully dependent?

(s_id, s_name, s_add)

CSE তে কজন student কত পেয়েছে? | full table consider

এটার জন্য s_id, sub_id দুইটাই লাগবে।

candidate key

• CSE এর teacher কে? এটা জানার জন্য s_id এর দরকার নেই, তাই s_id এর উপর fully dependent না।

বরং partially dependent.

এটা দূর করার জন্য table কে ভাঙতে হবে।

• New row add করলে s_id = 1 again, then name, s_add repeate হবে অর্থাৎ redundancy create হয়।

primary

| s_id | s_name | s_add |
|---|---|---|
| | | |

prime

| sub_id | Marks | Teacher |
|---|---|---|
| | | |

| sid | sub_id |
|---|---|
| 1 | |
| 2 | |

$T_1 \rightarrow$ candidate key fully dependendence

$T_2 \rightarrow$ এখানে partially dependent হিসেবে

$T_3 \rightarrow$

same primary key যদি অন্য table এ reference হিসেবে use হয় তাহলে তাকে foreign key বলে।

**3NF**

transitivity

$x \rightarrow y$

$y \rightarrow z$

$z \rightarrow x$

$x \rightarrow z$

→ Dept অনুযায়ী আলাদা হতে পারে

| (x) sub_id | marks | Teacher | E. id | (y) Exam_name | (z) Total marks |
|---|---|---|---|---|---|
| 2 | | | | | |
| 1 | | | | | |
| 4 | | | | | |
| 5 | | | | | |

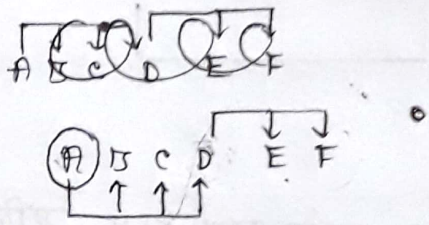
A B C D E F

$y \in x$

$z \in y$

so

$z \in x$




* 1 primary key/candidate key so there is no partial dependence

Exam

| D | E | F |
|---|---|---|
| Exam_id | Exam_name | Total_marks |
| | | |

→ R2 (DEF

R2(ABCD)

R2(DEF)

| A | B | C | D |
|---|---|---|---|
| sub-id | marks | Teachers | Exam-id |
| | | | |

Sub query (direct করা যাবে না, আলাদা করে query দিয়ে বের করে নিতে হবে।

| s_id | Name |
|---|---|
| 1 | A |
| 2 | B |
| 3 | C |
| 4 | D |

| s_id | marks |
|---|---|
| 1 | 50 |
| 2 | 60 |
| 3 | 40 |
| 4 | 20 |

single value/ column

যার marks $\boxed{\text{4 no id}}$ থেকে বেশি তাদের marks identify করতে হবে।

i) 4 no id এর marks বের করতে

(select s_id marks from s1 where s_id = 4) subquery

select s_id, s.name, s1. marks from s .s1 where marks> (subquery)

update s set name = 'E',

where marks < ( subquery
subquery )

ANY

ALL

ANY

select s_id, s.name, s1.marks from s, s1 where

marks > ANY/(subquery)
ALL subquery তে অনেকগুলো value

where s.id = 4 && marks > ALL ( subquery)
20 10, 60
0

1. maximum এর subquery

2. 16 এর id এর salary "

3.

# View

View (main table এর মত perform করবে) / security provide

(i) Logical data (how we want to see)

(ii) Physical data (how it is structured)

original table এর কিছু portion নিয়ে কাজ (original table change হবে না)

student

| id | name | add | phone no. |
|---|---|---|---|
| | | | |

create view temporary_table as select id, name from student where id>4 ;

| | | |
|---|---|---|
| | | |

update view (সব view কে update করা যায় না)

→ updatable view

join এর মাধ্যমে দুইটি table merge করে view হিসেবে রাখা যায়।

~~table~~

constraints ① joining এর মাধ্যমে যে table, তাকে update করা যায় না।

② group By এর update করা যাবে না।

③ Aggregate function এর মাধ্যমে view create হওয়া constraints.

④ view তে main table এর কয়েক কয়েক attribute থাকলে null use হবে

Create view temporary.table as select id.stu, name.stu, result.stu1 from . . .

temporary

| id | name | result |
|---|---|---|
| | | |

create . . . .

stu-id | autoincrement | data type

1 5
2 12
3 19
4 26

(5, 7)
starting
increament value

joing এর কাজ UNION

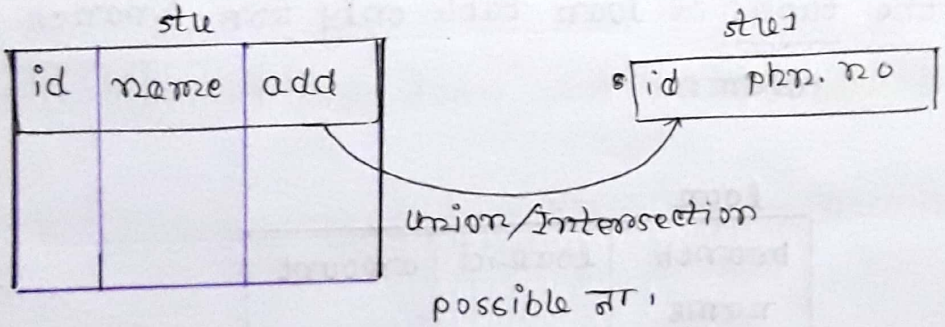
INTERSECTION

select * from stu

Union / Intersection

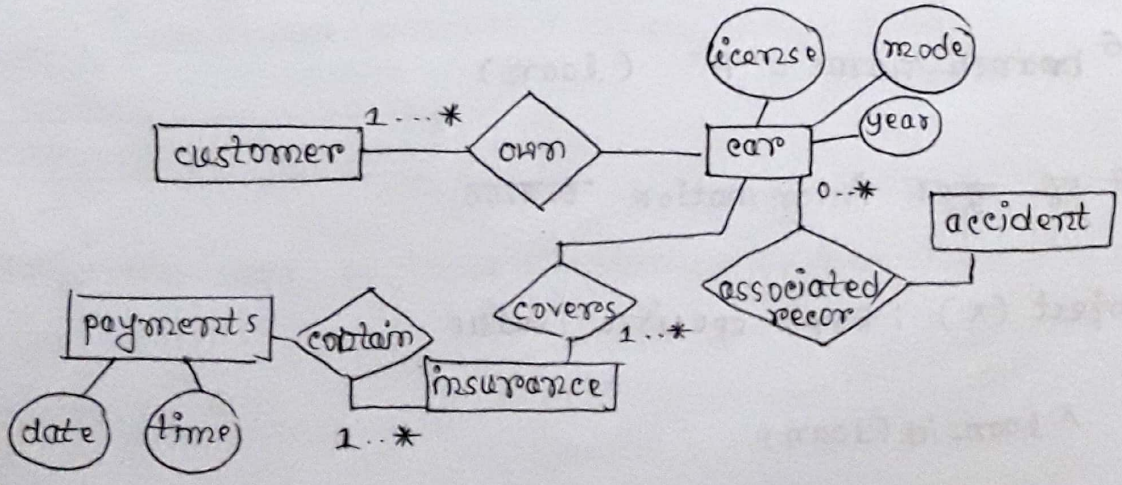
Select * from stu1

Intersect/Union এর constraints : দুই table এ column number, same হবে column name, type




ER

Entity → object , customer, car



8B (c day এর পরিবর্তে)

# Relational Algebra

some operations: $\underbrace{\text{select, project, rename}}_{\text{unary operation}}$, cartesian product, joining, union, intersect, insert, delete, update

Select : σ

* Find the tuples of loan table only for branch → A

column

loan

| branch name | loan-id | amount |
|---|---|---|
| A | | |
| A | | |

σ branch-name = "A" (loan)

A এর সুধা information আসবে

project (π) ; for specific value

π loan-id (loan)

$\pi_{loan\text{-}id}(\sigma_{branch\text{-}name = "A"}(loan))$

**Union:**

Display those ~~course~~ customer-name who have either loan or account or both

$\pi_{custname}(loan) \cup \pi_{cus\text{-}name}(account)$

**Intersection:** यानेकि loan एवं account both

$\pi_{cus\text{-}name}(loan) \cap \pi_{cus\text{-}name}(account)$

**Set difference:** Display those cus-name who have account but no loan

$\pi_{cus\text{-}name}(account) - \pi_{cus\text{-}name}(loan)$

**Cartesian Product:**

Find the name of those customer's who have loan at branch = "A"

loan

| branch-name | loan_id | amount |
|---|---|---|
| A | 1 | |
| A | 2 | |
| B | 3 | |

customer

| cus-name | loan-id |
|---|---|
| A | 1 |
| B | 2 |
| C | 3 |
| D | 4 |

$\pi_{cus\text{-}name}(\sigma_{branch\text{-}name = \text{"A"}}(loan \times customer))$

দুইটা table থেকে information নামানোর জন্য cartesian product)

$\pi_{cus\text{-}name}(\pi_{loan.loan\text{-}id = customer.loan\text{-}id}(\sigma_{branch\text{-}name = \text{'A'}}(loan \times customer)))$

$\pi_{cus\text{-}name}(\sigma_{branch\text{-}name = \text{'A'}}(loan \bowtie customer))$

Natural/inner join sign

Insert

$loan \leftarrow loan \cup \{5, 1700\}$

loan

| id | amount |
|---|---|
| 1 | 400 |
| 2 | 500 |
| 4 | 1500 |
| 5 | 1700 |

Delete

$loan \leftarrow loan - (\sigma_{id=4})$

Update

Increase amount by 5% for id > 2

$\pi_{amount * 0.5} (\sigma_{id > 2} (loan))$

<u>Aggregate function:</u>

→ sum up the total salary of employee

$\mathcal{G}_{sum(salary)} (employee)$

→ group by branch name

$_{branch_name}\mathcal{G}_{sum(salary)} (employee)$

$_{dept_name}\mathcal{G}_{average(salary)} (instructor)$

D day → 8 [Relational Algebra, subquery]

PL/SQL → extention of SQL.

normal SQL এ condition if, else, loop. function, value use করা যায় না, but PL এ করা যায়।

PLSQL এ SQL এর সব criteria থাকবে, তার সাথে extra কিছু function থাকবে।

int i, j,

i=10; variable

number (10)

number (10,2)

char ()

varchar ()

variable
Declaration (optional)

Execution
Begin → PL/SQL → block (প্রত্যেক execution)

END

Error Handling (range cross করে গেলে) (optional)

Emp

| ename | salary |
|---|---|
| S | |
| | |

Declaration

Declare tax varchar (20); current_salary varchar (20)

tax := price * tax_rate; (column name/value)

bonus := current_salary * 0.10;

Begin

select salary into current_salary new variable

from emp where ename = 'S'

bonus := current_salary * .10;

END

Declare :

salary র যে কাজ type , current-salary র same

data type দিতে চাইলে,

current_salary emp.salary % type

new variable

tablname . column

Declare

acc_balance number (11,2)

acc (constant) number (2) := 3;

debit number (5) := 2000;

min_balance (constant) number (5,2) := 500.00 ;

Begin

select. balance into acc_balance from account

where acc_id = acc ;
→ 3

acc_balance := acc_balance - debit ;

= 5000 - 2000

= 3000

ধরি variable

( if acc_balance > = min_balance
3000 > = 500

then

update account set balance = balance - debit where acc_id = acc

end if; )

END;

accounts

| acc_id | balance |
|---|---|
| 1 | |
| 2 | |
| 3 | |

9c

LOOP

i) simple loop

ii) For loop

iii) while loop

iv) go to

```
Declare
    x number := 1;
begin
    Loop
        x := x+1;
    END Loop;
    Exit when x>10;
```

break

or

```
if x>10 then Exit
End if;
```

```
Begin
    Loop
        x := x+1
    Exit when x>10;
    End loop
End;
```

※

```
Begin  Declare
        x number := 1;
        y number := 1;
    Begin
        << outer_loop >>
        Loop
            x := x+1;
            y := y;
            << inner_loop >>
```

```
Loop
    y:= y+1;

    s:= s+x*y;

    inner_loop
Exit when y>15;

    outer_loop
Exit when s>100;

    End Loop  inner_loop

    END Loop  outer_loop
```

x = 2
y = 2
s = 9

```
Declare
  . . .

Begin
  \
    Declare

    Begin
```

Ivan Bay toss

## ii) For LOOP :

```
Declare
    x number := 1
begin
    for i in 1 .. 10 / while (i <= 10)
    loop if mod (i,2) = 0 then
    dbms insert into table values (i,x,' i is even');
    else
        insert into table values (i,x,' i is odd');
    end if;
x:=x+100;
end loop.
dbms_output.putline ('value of x' || x);
End;
```

go to

```
begin
    goto [innero;] label name
    goto outero;
<< innero >>

<< outero >>
```

(9D)

[Transaction] → consider operation as atomic/ single unit

↓

ACID property

PL/SQL এর set of operation successfully complete হলে তাকে transaction বলে।

```
begin
    update
```
] block

transaction management components take care of atomicity

transaction অনেকগুলো instruction নিয়ে হয়। atomicity হলো সবগুলো instruction ই কাজ করবে। partially হলে চলবে না।

## Database এর property

A C I D

- A → Atomicity (either all or none)
- C → consistancy সামঞ্জস্য
- I → Isolation একটা complete না হওয়া পর্যন্ত অন্যটা শুরু হবে না।
- D → Durability (permanent

Balance

Debit account $\xrightarrow{\text{transfer}}$ checking account

A $\xrightarrow{5000}$ B

Decrease update — Increase update

[ Atomicity property থাকতে হবে, A থেকে 5000 decrease হওয়ার পর checking account এ যাওয়ার আগেই transection off হয়ে গেলে consistancy break হয়

**commit:** commits the current transaction; that is, it makes the updates performed by transaction become permanent in the database.

Read : at a time অনেকে access করতে পারে।

Write → update, delete, insert

Write operation এর সময় অন্য কেউ access করতে পারবে না।

Transaction is a sequence of SQL statement that oracle database treats as unit (single)

Declare

Begin

Update account set balance = balance - 5000

where acc_id = 5;

// এখানে commit লেখা যাবে না।

Update account1 set balance = balance + 5000

where acc_id = 6;

commit; → successfull transaction.

END

transaction

Roll back ( commit এর মাধ্যমে যেগুলো save করা হয়েছিলো, সেগুলো undo করার জন্য roll back )

নিজে থেকে

// if you don't explicitly commit the transaction and the program terminates abnormally, then the last uncommited transaction is automatically rolled back.

program crash করলে system roll back করে থাকার situation এ করতে পারে। Implicite, atomicity maintain করার জন্য

```
Begin
    Update account
        where acc_id = 5;
        Delete acc_details from account where
        acc_id > 6;
        savepoint a1;
                  levelname

commit;
    [roll back] to savepoint a1;
```

- Explicit (by user)
- Implicit (by system)

**Rollback:** it undoes all the updates performed by SQL statement in the transaction.

The database state is restored to what it was before the first statement of the transaction was executed.

10c

# cursor

main memory র যে part এ কাজ হচ্ছে, সেই part কে /area indicate করে cursor | cursor is a pointer to contex area

Begin
select * from table 1
→ SQL private area (একে point করে cursor)

[DBMS op] → Delete * from t1 where id = 2;

cursor দুই ধরনের হতে পারে:

- Implicit cursor [attribute]
- Explicit cursor

Attribute of implicit cursor: [no row affected ...
এসব implicite cursor maintain করছে]

→ Not % Found (~~return false~~, return true)

→ % Found (return true, is affected)

→ % row count (returns the number of rows affected

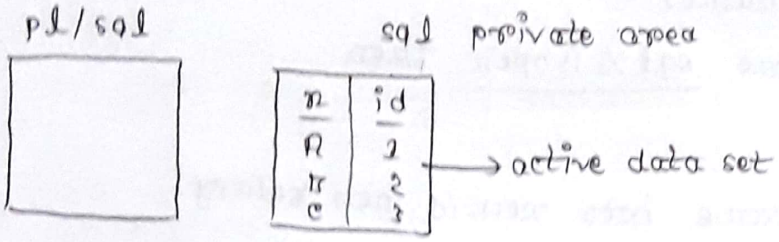
→ % Isopen (always return false, because the database always closes the SQL cursor automatically after executing its associated SQL statement

# implicit cursor

```
Declare
        rows char(4);
Begin
    update ex set salary = salary * 0.25
    where job = 'p';
        If sql % not found then dbms_output.put_line
                        ("not affected");
    rows := (sql % row count)
if sql % rowcount > 0 then
        dbms_out.put_line (rows);
```

isopen (cursor open আছে কিনা)

## Cursor (pointer)




individually access করতে চাইলে explicit cursor use হবে।

Attribute: (Explicite)

→ cursor declare

→ open

→ Fetch

→ close (memory থেকে free করে দিতে হবে)

Declare

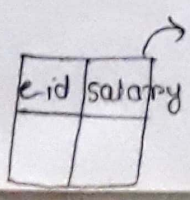
cursor নতুন নাম

cursor c_name is select e_id, salary from employee where dept_no = 3;

table, e_id এর data type

new_id employee.e_id % type;

new_salary employee.salary % type;




| eid | Sal | ad | dept |
|---|---|---|---|
| | | | |

```
Begin
    open c_name;
    if c_name sql% isopen then
loop
    fetch c_name into new_id, new_salary;
    update employee set new_salary = new_salary+100
        where e_id = new_id;
    exit when c_name sql% not found
    end loop;

    commit;

    close c_name;
→ else
    end;
```

Exception handling (optional) : Error

implicit

data not found

Too many rows

Logon denied

c day 3.40 CT

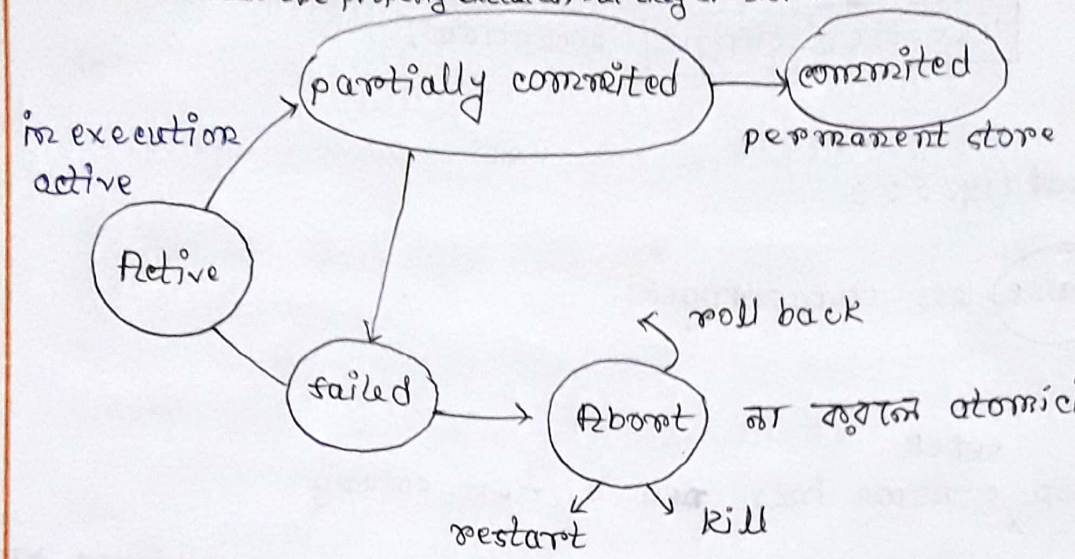
## Explicit:

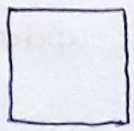
```
Declare
        less_than_target exception;

⇒ if salary<500
go-to <---- raise less_than_target;
else
        setek
  loop c_name into new_id, new_salary
        update employee set new_salary=new_salary +100;
        where e_id=new_id;
        exit when c_name sql%notfound
  End loop;
  commit;
  close c_name;

Exception
        when less_than_target then
                dbms_output ("not possible");
```

# Transaction

নিচের ৫টা state এ থাকতে পারে

whatever the change is the transaction has made, are properly executed, but they are stored in the local batter in main memory



commit লেখার আগ পর্যন্ত partially

commit) commited

A → 1000
B → 2000 | A+B

Transfer 50 from A → B

→ Read (A); (আগে check করতে হবে 50$ আছে কিনা)

A := A − 50;

write (A); (update)

→ Read (B);

B := B + 50;

write (B);

Logical error : kill

system crash → Restart

transaction এর পর A+B এর value preserved (so atomicity maintained)

concurrent Execution/ (একই সাথে execution)

concurrency control

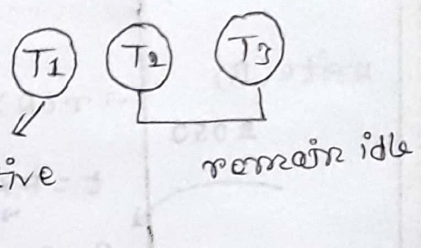
A → B

10% ] দুইটা একসাথে possible না।
50$

একটার পর আরেকটা হবে।

$T_1 \rightarrow T_2$

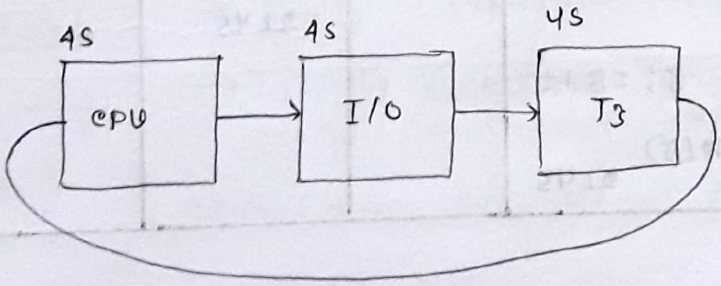
serial transection




CPU, I/O → একসাথে operation

concurrent execution

time to divide throughput




concurrency control component

concurrent, consistency

A → 1000
B → 2000

S1

| $T_1$ | $T_2$ |
|---|---|
| r(A); | |
| A = A - 50; | |
| w(A); 950 | |
| r(B); | |
| B := B + 50; | |
| write(B) 2050 | |
| | → r(A) |
| | t = A * .1 → 95 |
| | A = A - t; |
| | w(A); 855 |
| | r(B); |
| | B := B + t |
| | w(B) 2145 |

S2

| $T_1$ | $T_2$ |
|---|---|
| r(A) | |
| A = A - 50 | |
| w(A) → 950 | r(A) |
| | t = A * .1 |
| | A = A - t |
| | w(A) 855 |
| 2000 r(B) | |
| B = B + 50 | |
| w(B) 2050 | |
| | r(B) |
| | B := B + t |
| | w(B) 2145 |

S3

| $T_1$ | $T_2$ |
|---|---|
| r(A) | |
| A = A - 50 950 | |
| w(A) 950 | |
| r(B) | |
| B = B + 50 | |
| w(B) 2050 | |
| | r(A) → 1000 |
| | t = A * .1 |
| | A = A - t 900 |
| | w(A) → 900 |
| | r(B) → 2000 |
| | B = B + t 2150 |
| | w(B) |

S1 → not concurrent

S2 → concurrent (consistancy maintained)

S3 → concurrent (but consistancy is not maintained)

## Serializability

concurrent perform করলেও যেন serially perform এর মত রেজাল্ট show করে।

→ Concurrent Execution ( $T_1$, $T_2$, $T_1$, $T_2$

→ consistency maintain (serially)

→ conflict

→ view

**Conflict :**

→ $I_i$, $I_j$ (separate operation) — one Read, one write

→ $T_1$, $T_2$ (separate transaction) ($R_1(A)$, $W_2(A)$)

→ Q (same data item) ($W_1(A)$, $W_2(A)$)

এই তিন case এ conflict হবে।

$T_1$ → Read(Q), $T_2$ → Read(Q) [same op, no conflict

$T_1$ → " $T_2$ → write(Q)

$T_1$ → write(Q) $T_2$ → write(Q) [write দুইটা হলে conflict)

$T_1$ → write(Q) $T_2$ → Read(Q)

serial maintain করতে conflict

conflict serializability:

| S<br>schedule 1 | | S'<br>schedule 2 | | | |
|---|---|---|---|---|---|
| $T_1$ | $T_2$ | $T_1$ | $T_2$ | $T_1$ | $T_2$ |
| r(A) | | r(A) | | r(A) | |
| w(A) | | w(A) | | w(A) | |
| | r(A) | | r(A) | r(B) | |
| | w(A) | r(B) | | w(B) | |
| r(B) | | | w(A) | | |
| w(B) | | w(B) | | | r(A) |
| | r(B) | | r(B) | | w(A) |
| | w(B) | | w(B) | | r(B) |
| | | | | | w(B) |

non-conflict

conflict equivalence → serializability

$\boxed{S \rightarrow S'}$

non conflict operation swapping

যেগুলোর conflict নেই, সেগুলো swap করলে 3

same result আসবে।

Q: conflict-S. কিনা check কর।

* বই এর table

View Serializability: (same operation perform

$T_1 \rightarrow T_2$

| $T_1$ | $T_1$ |
|---|---|
| $\boxed{r(A)}$ | $\boxed{r(A)}$ |
| $w(A)$ | $w(A)$ |
| $r(B)$ | $r(A)$ |
| $w(B)$ | $w(A)$ |
| $r(A)$ | $r(B)$ |
| $w(A)$ | $w(B)$ |
| $r(B)$ | $r(B)$ |
| $\boxed{w(B)}$ | $\boxed{w(B)}$ |
| $S_1$ | $S_2$ |

1. initially same value read
2. write op দিয়ে শেষ হবে, same data
3. initial value না হলে, কমপক্ষে same type হতে হবে।

$T_1 \rightarrow T_2$

Locking Protocol (To maintain isolation)

↳ shared → if both read (read only)

→ Exclusive → Rest

| | S | X |
|---|---|---|
| S | T | F |
| X | F | F |

compatibility

Lock → X(A); exclusive lock isolation provide

W(A);

Unlock (A)

Lock → S(B);

# Locking schedule

schedule 1 w(A) perform করার আগেই unlock করলে consistancy বজায় থাকবে না।

'বই এর উদাহরণ'

$\boxed{12C}$

Lock

$T_1$: Lock-X(A)

w(A);

unlock(A);

$T_2$: Lock-S(B)

r(B);

| Exclusive lock, write transaction এর সময় অন্য কেউ যেন access করতে না পারে।

| $T_1$ | $T_2$ |
|---|---|
| lock-X(A) | |
| ~~lock-S(A)~~ | lock-S(B) |
| lock-X(B) | |
| | lock-S(A) |

Deadlock

[continuous waiting situation]

A transaction may be waiting for an X-lock on an item, while a sequence of other transac request and are granted an S-loc on same item.

(starvation)

$T_1 \xrightarrow{\text{held}} S(A)$

$T_2 \xrightarrow{\text{Request}} X(A)$ ||wait

$T_3 \xrightarrow{\text{Request}} S(A)$ ✓

$T_4 \xrightarrow{\text{Request}} S(A)$ ✓

unlock এর জন্য wait করতে হচ্ছে

Two phase Locking: serializability maintain
concurrency

Step :

→ Growing phase : Access to all locks

→ shrinking phase : Release of all locks

$T_1$ : ✓ lock - S(A)

r(A)

locking point * lock - S(B)

r(B)

display (A+B);

✓ unlock (A);

* unlock (B);

growing phase

shrinking

lock-S(B)

r(B)

unlock (B);

display (A+B);

display এর আগে অন্য কেউ access করতে পারে

$T_2$: lock-X(B)

B = B - 50;

W(B);

lock-S(A);

R(A)

Unlock(B)

Unlock(A);

Unlock(B);

Deadlock handling:

1) Deadlock prevention:

i) ordering request for locks

ii) transaction rollback.

2) Detection and recovery: