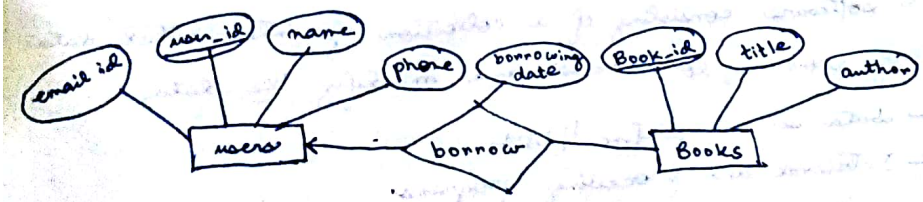
Functionality of DBMS —

1) Provide a mechanism for organized development of applications in an integrated manner.
2) Prevent redundancy and inconsistency.
3) Handle the problems of concurrency.
4) Provide security features.
5) Allow expressing of integrity constraints which can be automatically checked.
6) Backup and recovery
7) Efficiency

Database Models

1) E-R Model :



← arrow-head towards the one side in one-to-many.

→ Relation

Entity: Objects that exists and are distinguishable.

- A relation may or may not have attributes. There will be not primary key.

Relation: Association between two or more entities.

- Attributes are the properties of an entity.

Banking System

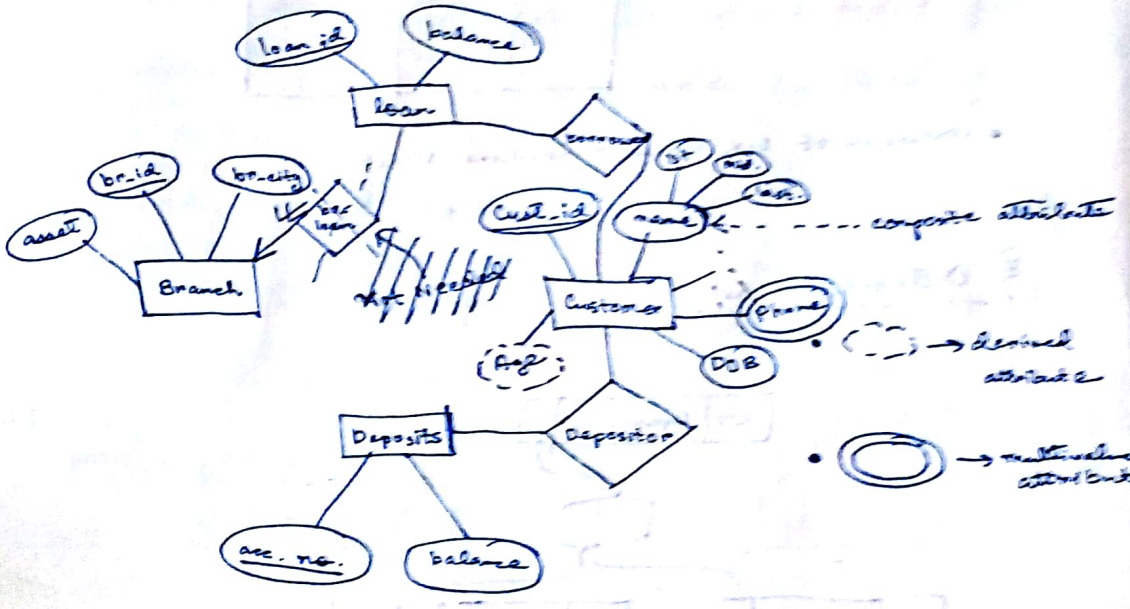
- Branches (br_id, br_city), br_area, assets)
- Customers (cust_id, name, ~~addr~~, phone, ~~email~~)
- ~~Offer~~ loans (loan_id, loan_amt, br_id, ~~loan_date~~)
- Deposits (A/c No, br_id, balance, interests, ...)

- Borrower (cust_id, loan_id)
- Depositor (cust_id, A/C_no)

— Entity
— Attributes
— Relationship

• Collection of entity having the same property → entity set

— Keys



• ( ) → derived attribute

• ◎ → multivalued attribute

Attributes

- Single
- Composite (DOB(date, month, year))
- Single Valued [DOB → cannot have more than 1 DOB]
- Multivalued [Ph no]
- Derived [age, given date of birth]
- Null values

• Every attribute should have a Domain. → Values it can take

- Intersection ($\cap$) : $r \cap s$
- natural join ($\bowtie$)
- Division ($\div$)

Intersection ($\cap$):

$$\Pi_{cust\_name}(borrower) \cap \Pi_{cust\_name}(depositor)$$

Natural Join ($\bowtie$):

$r(A\ B)$

α	1
β	2

$s(A\ D\ E)$

α	10	a
β	10	a
β	20	b
γ	10	b

~~$r \times s$~~ (A)

$r \times s\ (A\ B\ s.A\ D\ E)$

$r \bowtie s\ (A\ B\ D\ E)$

loan (loan_no, br_name, amount)

borrower (loan_no, cust_name)

$r \bowtie s$

$(A$	B	D	$E)$
α	1	10	a
α	1	10	b
β	2	10	a
β	2	20	b

$$\Pi_{\underbrace{A,B,D,E}_{RUS}}(\sigma_{r.A=s.A}(r \times s))$$

1) Loan × borrower → 9 records, 5 attributes

2) loan ⋈ borrower ← inner join

loan_no	br_name	amount	cust_name
L-170	Saltlake	10L	Ram
L-230	Garia	20L	Jadu

If another records need to add along with the above —

L-260 Esplanade 30L null

3) loan ⟕ borrower ← Left outer join

null ↑ record in borrower table is not present

~~L-170 Salt~~

4) loan ⟖ borrower ← right outer join

Along with the 1st 2 records —

L-165 null null Madhu

5) loan ⟗ borrower ← full outer join

Aggregation Function

$$G_{avg(amt)}(branch)$$

$$_{branch}G_{avg(balance)}(account)$$

$$_{G_1, G_2 \ldots G_m}G_{F_1(A_1)\, F_2(A_2) \ldots F_m(A_m)}(\gamma)$$

Some Queries

Loan

Loan_no	br_name	amount
L-170	Garia	10L
L-230	Anandapur	20L
L-260	Ballygunge	8L

Borrower

cust_name	loan_no
Ram	L-170
Jadu	L-230
Madhu	L-155

1) loan natural inner join borrower on

natural inner join → Then loan_no won't appear twice in the output and the condition loan.loan_no = borrower.loan_no will be checked automatically

loan.loan_no = borrower.loan_no as

llb (loan.loan_no, br_name, amount, cust_name, borrower.loan_no)

Output:

L-170	Garia	10L	Ram	L-170
L-230	Anandapur	20L	Jadu	L-230

2) loan natural inner join as

llb (loan_no, br_name, amount, cust_name)

Same Output

Virtual Relation

3) loan left outer join borrower on
loan.loan_no = borrower.loan_no

loan.loan_no	br_name	amount	cust_name	borrower.loan_no
L-170	Garia	10L	Ram	L-170
L-230	Anandapur	20L	Jadu	L-230
L-260	Ballygunge	8L	null	null

4) loan natural left outer join borrower

loan_no	br_name	amount	cust_name
L-170	Garia	10L	Ram
L-230	Anandapur	20L	Jadu
L-260	Ballygunge	8L	null

5) loan full outer join borrower using (loan_no)

* instead of loan on loan.loan_no = borrower.loan_no

L-170	Garia	10L	Ram	L-170
L-230	Anandapur	20L	Jadu	L-230
L-260	Ballygunge	8L	null	null
null	null	null	Madhu	L-155

6) loan natural full outer join borrower

L-170	Garia	10L	Ram
L-230	Anandapur	20L	Jadu
L-260	Ballygunge	8L	null
L-155	null	null	Madhu

Join Type	Join Condition
1) inner join	natural on <predicate> using $(A_1, A_2, \ldots A_n)$
2) left outer join	"
3) right outer join	"
4) full outer join	"

Write Queries:

1) Find the average account balance of those branches where the average account balance is 12L.

~~select br_name~~

2) Find the maximum across all branches of the total balance at each branch.

1.
```
select br_name, avg(balance)
from ~~branch~~ account
group by (br_name)
having (avg(balance) > 12L);
```

```
select br_name, avg_balance
from (select br_name, avg(balance)
      from account
      group by (br_name)
      as br_total (br_name, avg_balance))
where avg_balance > 12L;
```

2.

```
select max(balance) as max_balance
from balance;
```

```
select br_name, max(sum_b)
from (select br_name, sum(balance)
      from account
      group by br_name)
as br_max(br_name, sum_b));
```

Another Method:

```
with max_balance(value)
    as select max(balance)
       from account
```