# Relational Model

Dr. Odelu Vanga

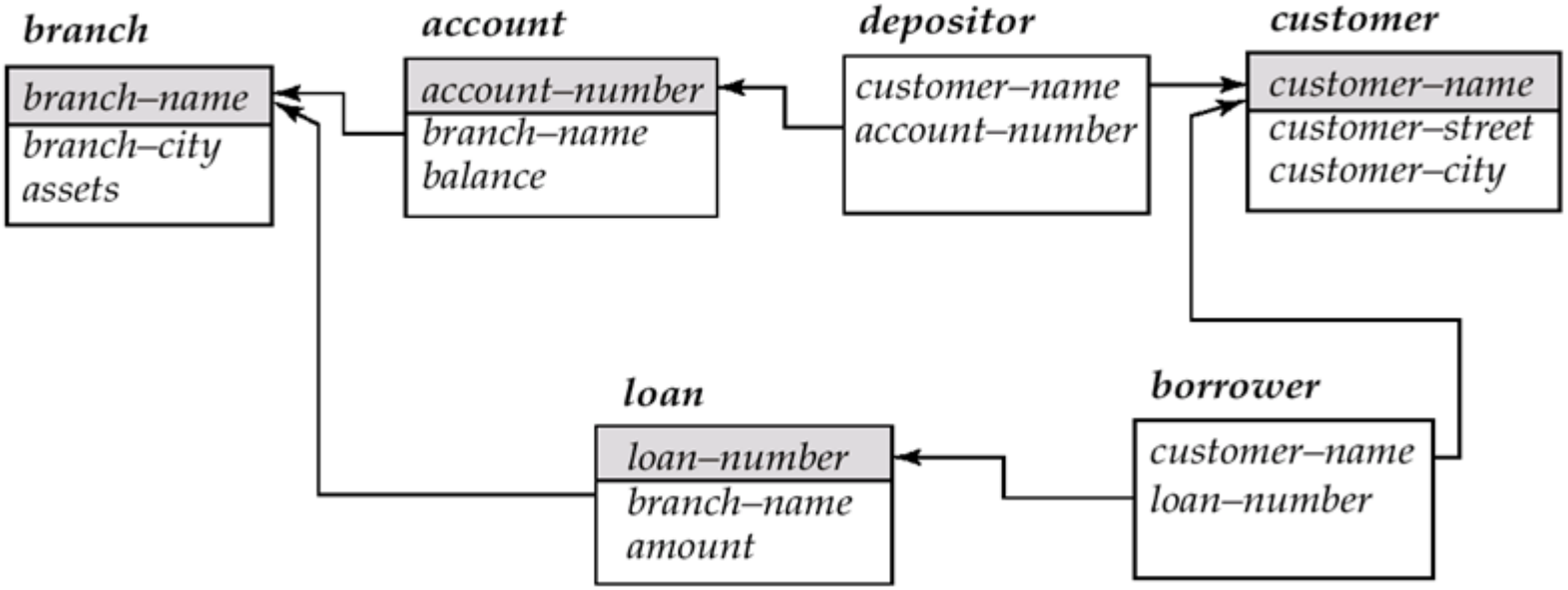
Indian Institute of Information Technology Sri City

http://www.iiits.ac.in/people/regular-faculty/dr-odelu-vanga/

# Nested Subqueries

- SQL provides a mechanism for the nesting of subqueries.
- A subquery is a **select-from-where** expression that is nested within another query.
- A common use of subqueries is to perform tests for
  1. set membership
  2. set comparisons
  3. set cardinality

# set membership




- Find all customers who have both an account and a loan at the bank.

  **select distinct** *customer-name*
  **from** *borrower*
  **where** *customer-name* **in (select** *customer-name*
  **from** depositor)

- Find all customers who have a loan at the bank but do not have an account at the bank

  **select distinct** *customer-name*
  **from** *borrower*
  **where** *customer-name* **not in (select** *customer-name*
  **from** *depositor)*

# Set Comparison

- Find all branches that have greater assets than some branch located in Brooklyn.

```
select distinct  T.branch-name
from branch as T, branch as S
where  T.assets > S.assets and
          S.branch-city = 'Brooklyn'
```

- Same query using > **some** clause

```
select branch-name
from branch
where assets > some
                (select assets
                 from branch
                where branch-city = 'Brooklyn')
```

# Definition of Some Clause

- F <comp> **some** $r \Leftrightarrow \exists\ t \in r\ \ s.t.$ (F <comp> $t$)
  Where <comp> can be: $<, \leq, >, =, \neq$

(5 < **some** | 0 / 5 / 6 |) = true (read: 5 < some tuple in the relation)

(5 < **some** | 0 / 5 |) = false

(5 = **some** | 0 / 5 |) = true

(5 ≠ **some** | 0 / 5 |) = true (since 0 ≠ 5)

(= **some**) ≡ **in**
However, (≠ **some**) ≢ **not in**

# Definition of all Clause

- F <comp> **all** $r \Leftrightarrow \forall\ t \in r$ (F <comp> $t$)

(5< **all** | 0 / 5 / 6 ) = false

(5< **all** | 6 / 10 ) = true

(5 = **all** | 4 / 5 ) = false

(5 ≠ **all** | 4 / 6 ) = true (since 5 ≠ 4 and 5 ≠ 6)

(≠ **all**) ≡ **not in**

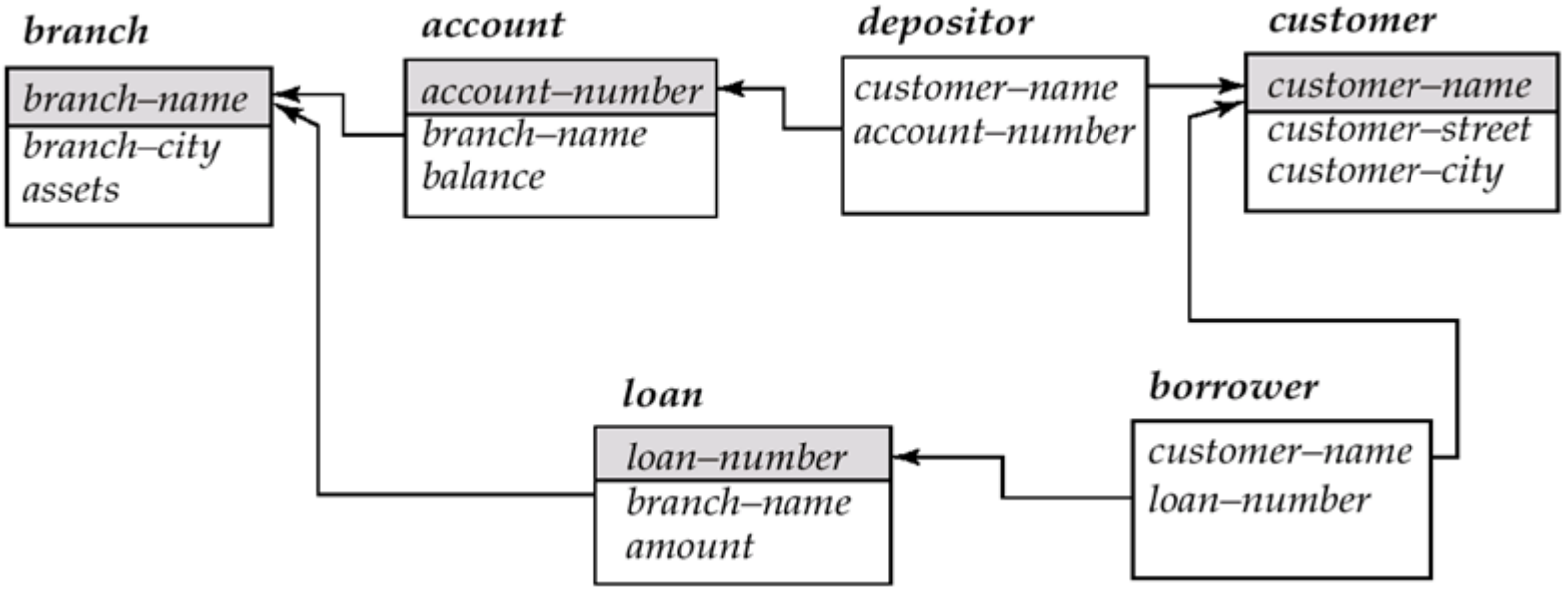
However, (= **all**) ≠ **in**

# Example Query

- Find the names of all branches that have greater assets than all branches located in Brooklyn.

```
select branch-name
from branch
where assets > all
    (select assets
    from branch
    where branch-city = 'Brooklyn')
```

# Derived Relations

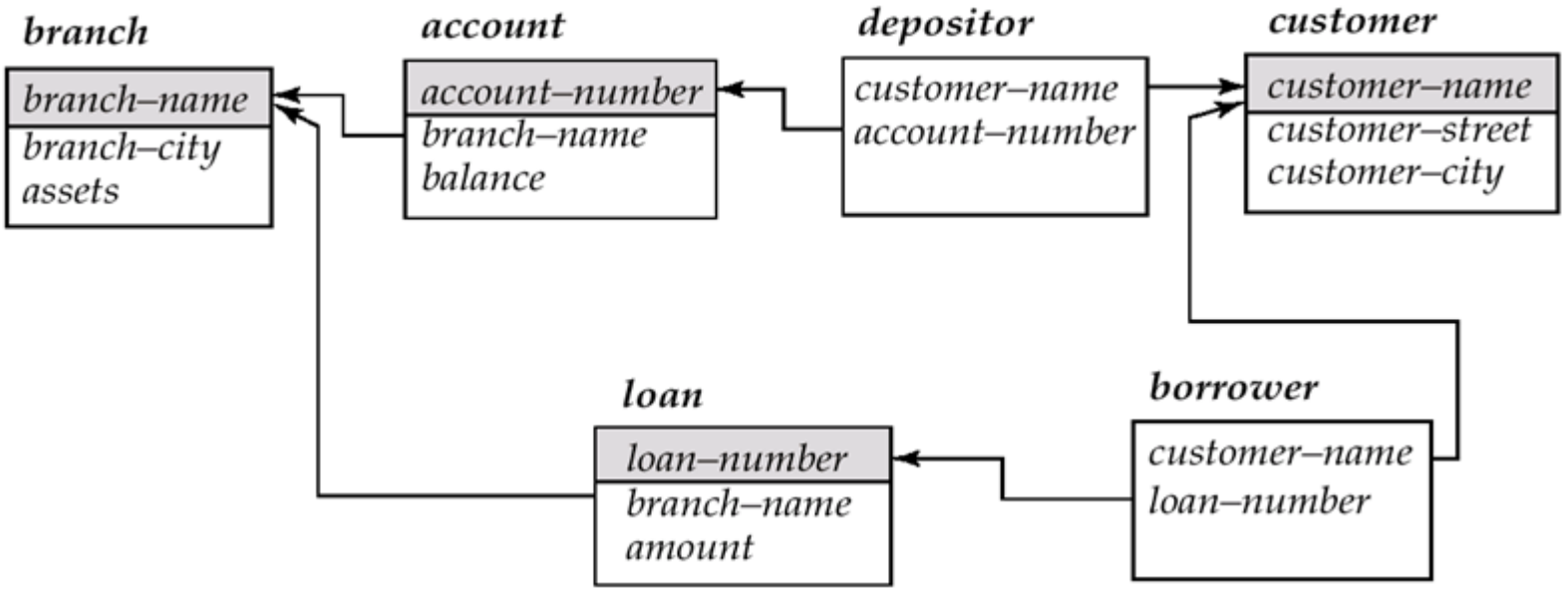



Find the names of all customers who have a loan at the Perryridge branch.

- Query 1

$$\Pi_{\text{customer-name}}(\sigma_{\text{branch-name} = \text{“Perryridge”}}(\sigma_{\text{borrower.loan-number} = \text{loan.loan-number}}(\text{borrower x loan})))$$

# Derived Relations




Find the names of all customers who have a loan at the Perryridge branch.

– Query 2

$$\Pi_{\text{customer-name}}(\sigma_{\text{loan.loan-number} = \text{borrower.loan-number}}( (\sigma_{\text{branch-name} = \text{“Perryridge”}}(\text{loan})) \times \text{borrower}))$$

# Example Queries

- Find the largest account balance and rename *account* relation as *d*

Expression:

$$\Pi_{balance}(account) - \Pi_{account.balance}(\sigma_{account.balance < d.balance}(account \times \rho_{d}(account)))$$

SQL query:

```
select balance
from account
where balance not in
        ( select account.balance
          from account, account as d
          where accounnt.balance < d.balance);
```

# Example Queries

- **(Try Own)** Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank.

$$\Pi_{customer\text{-}name}(\sigma_{branch\text{-}name = \text{“Perryridge”}}(\sigma_{borrower.loan\text{-}number = loan.loan\text{-}number}(\text{borrower x loan}))) - \Pi_{customer\text{-}name}(\text{depositor})$$

# THANK YOU