Relational Model

Dr. Odelu Vanga

Indian Institute of Information Technology Sri City

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

Tuple Relational Calculus

- A nonprocedural query language, where each query is of the form

$$\{t \mid P(t)\}$$

- It is the set of all tuples t such that predicate P is true for t
- t is a *tuple variable*, $t[A]$ denotes the value of tuple t on attribute A
- $t \in r$ denotes that tuple t is in relation r
- P is a *formula* similar to that of the predicate calculus

Predicate Calculus Formula

1. Set of attributes and constants
2. Set of comparison operators: (e.g., $<, \leq, =, \neq, >, \geq$)
3. Set of connectives: and ($\wedge$), or (v), not ($\neg$)
4. Implication ($\Rightarrow$): x $\Rightarrow$ y, if x true, then y is true

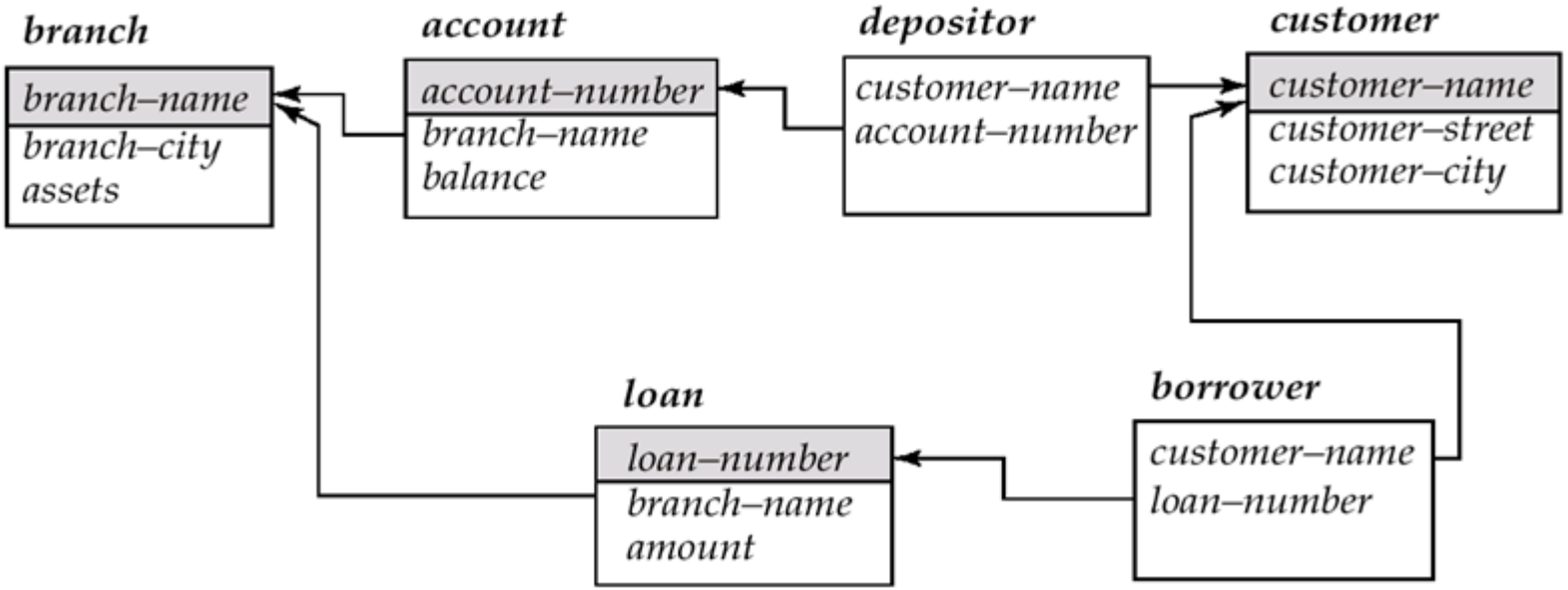
$$x \Rightarrow y \equiv \neg x \text{ v } y$$

for example: **x = 2 implies x + 3 = 5**

5. Set of quantifiers:
   - $\exists\, t \in r\, (Q(t))$ ≡ "there exists" a tuple t in relation r such that predicate $Q(t)$ is true
   - $\forall t \in r\, (Q(t))$ ≡ Q is true "for all" tuples t in relation r such that predicate $Q(t)$ is true

Banking Example

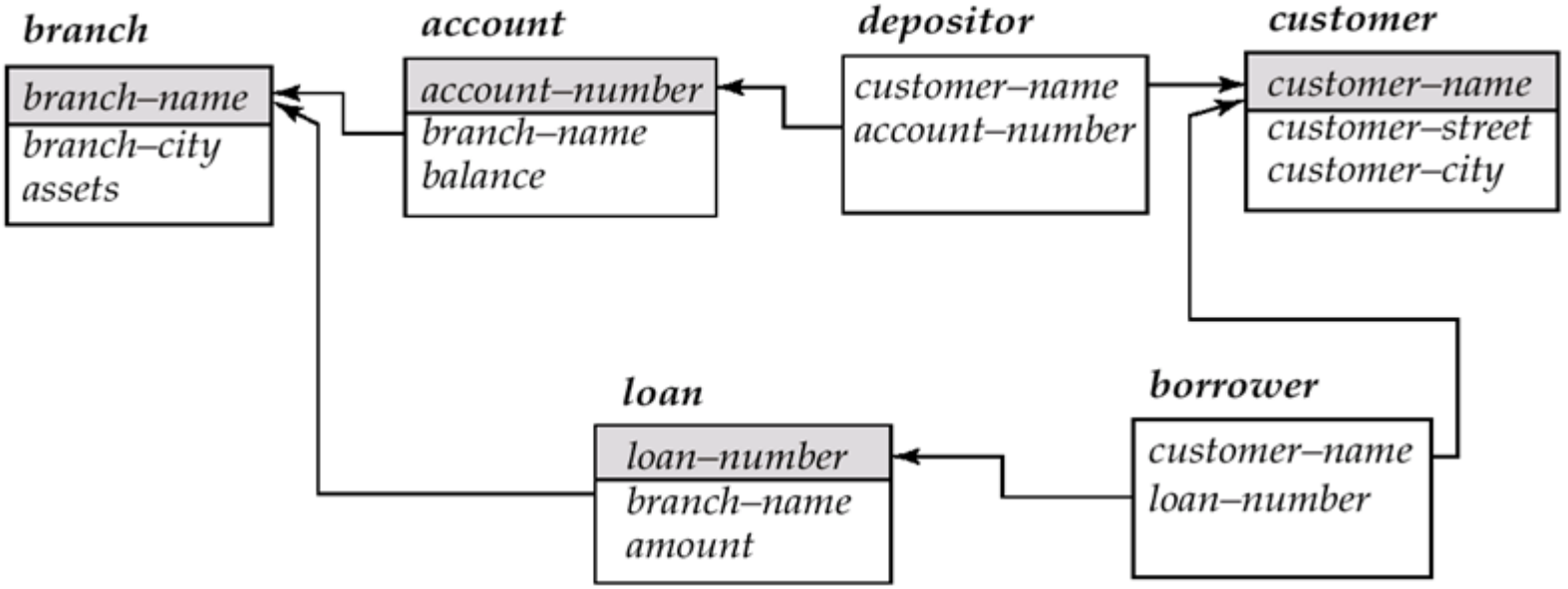
- *branch (branch-name, branch-city, assets)*
- *customer (customer-name, customer-street, customer-city)*
- *account (account-number, branch-name, balance)*
- *loan (loan-number, branch-name, amount)*
- *depositor (customer-name, account-number)*
- *borrower (customer-name, loan-number)*



Example Queries

- Find the *loan-number*, *branch-name*, and *amount* for loans of over $1200

$$\{t \mid t \in loan \wedge t\,[amount] > 1200\}$$

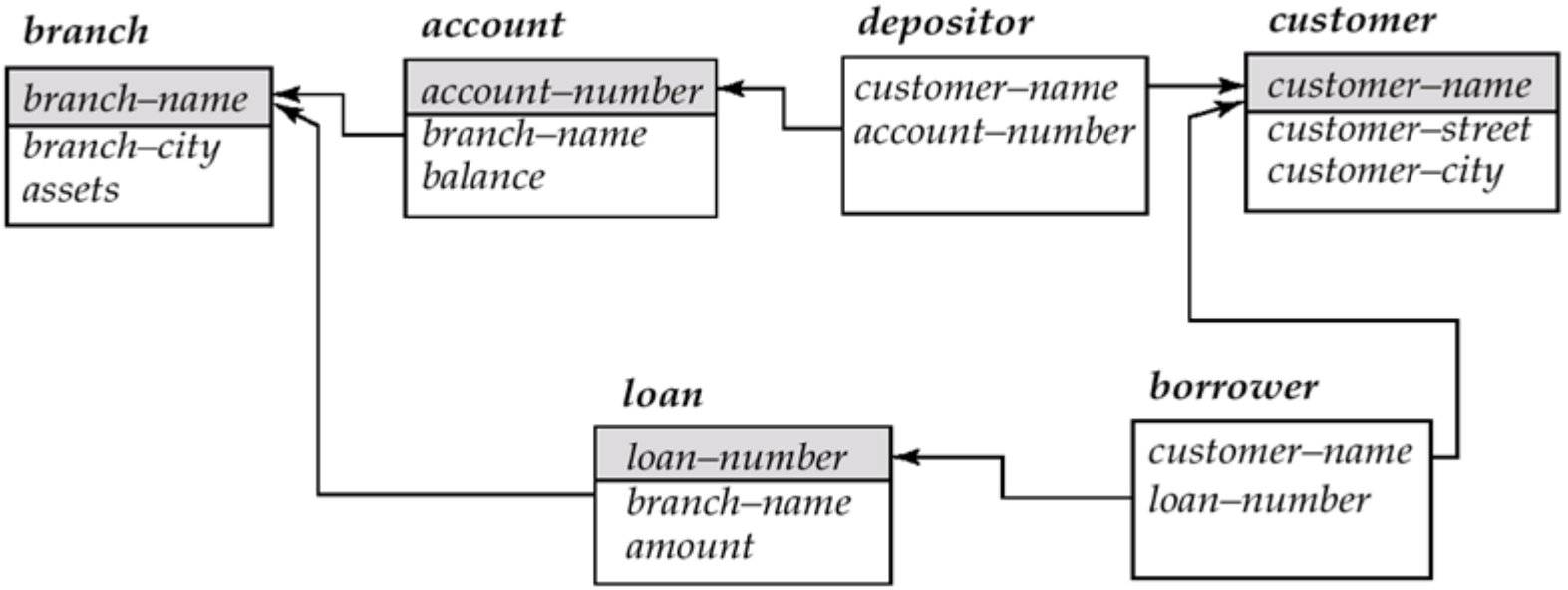


Example Queries

- Find the loan number for each loan of an amount greater than $1200

$$\{t \mid \exists s \in \text{loan}\ (t[\textit{loan-number}] = s[\textit{loan-number}] \wedge s\ [\textit{amount}] > 1200\}$$

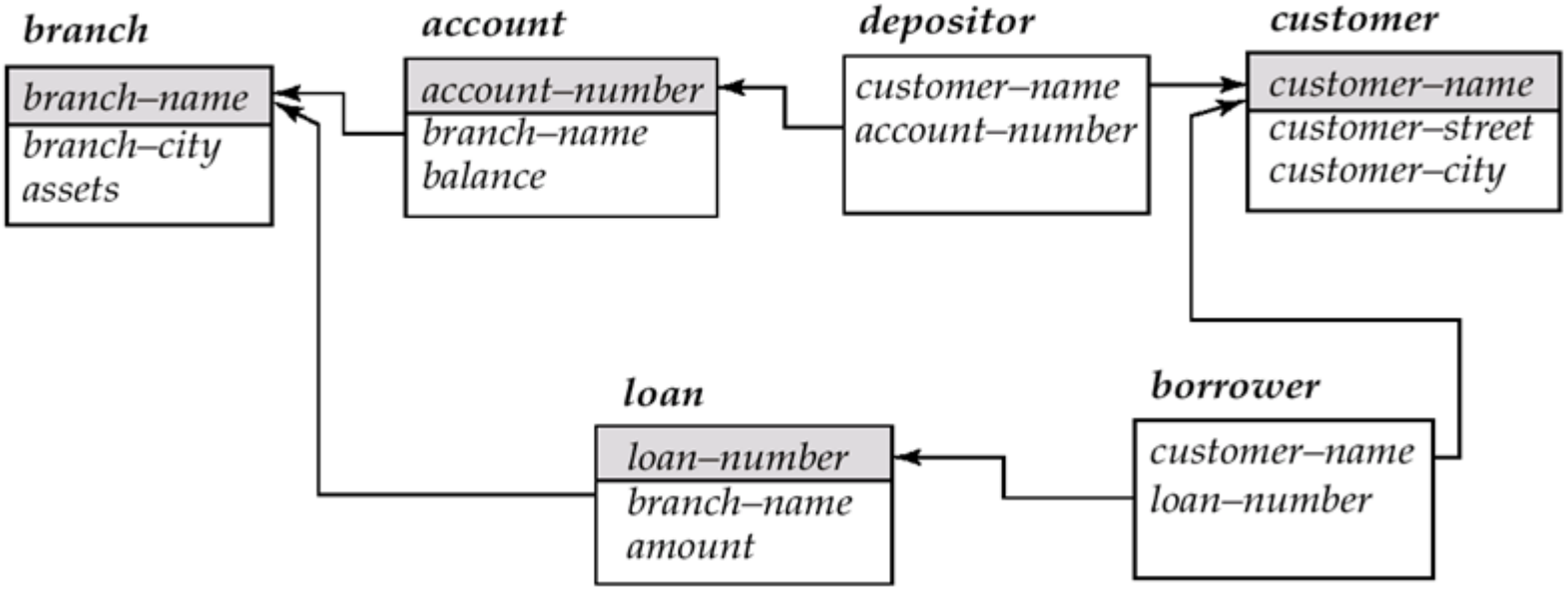
Notice that a relation on schema [*customer-name*] is implicitly defined by the query



Example Queries

- Find the names of all customers having a loan, an account, or both at the bank

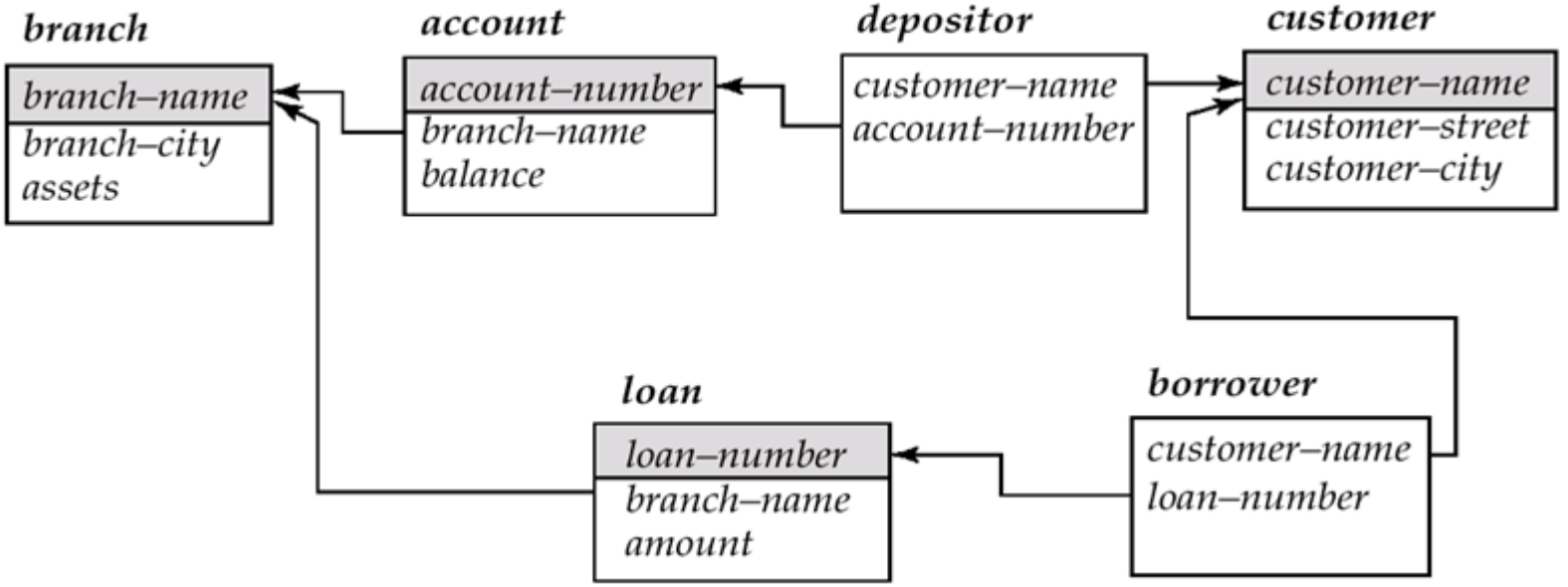
$$\{t \mid \exists s \in borrower(t[customer\text{-}name] = s[customer\text{-}name]) \vee \exists u \in depositor(t[customer\text{-}name] = u[customer\text{-}name])$$



Example Queries

- Find the names of all customers who have a loan and an account at the bank

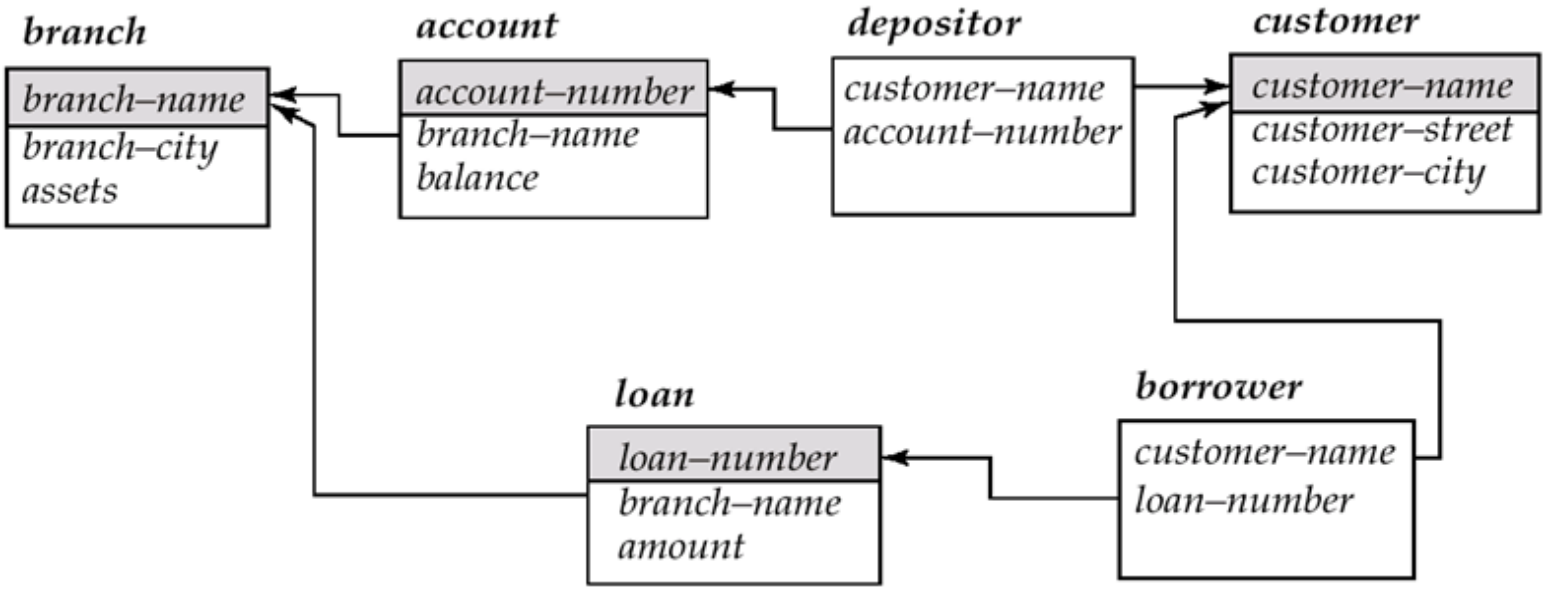
$$\{t \mid \exists s \in borrower(t[customer\text{-}name] = s[customer\text{-}name]) \wedge \exists u \in depositor(t[customer\text{-}name] = u[customer\text{-}name])\}$$



Example Queries

- Find the names of all customers having a loan at the Perryridge branch

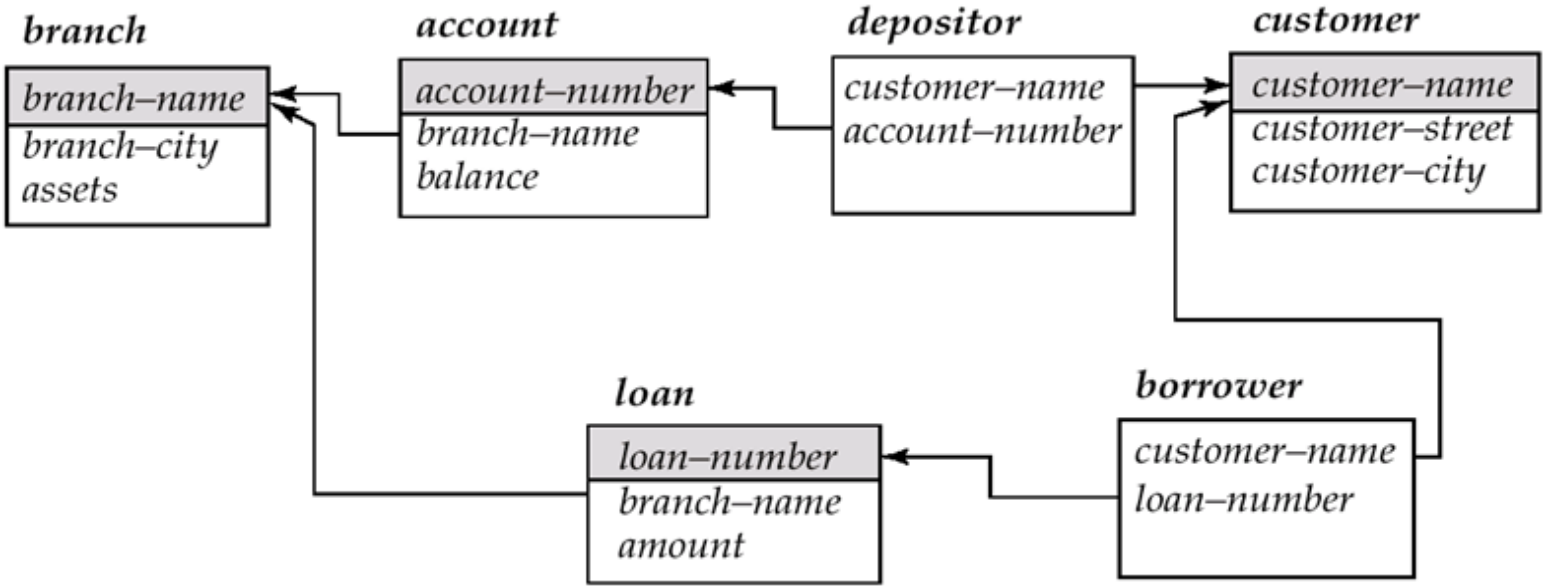
$$\{t \mid \exists s \in borrower(t[customer\text{-}name] = s[customer\text{-}name] \wedge \exists u \in loan(u[branch\text{-}name] = \text{“Perryridge”} \wedge u[loan\text{-}number] = s[loan\text{-}number]))\}$$



Example Queries

- Find the names of all customers who have a loan at the Perryridge branch, but no account at any branch of the bank

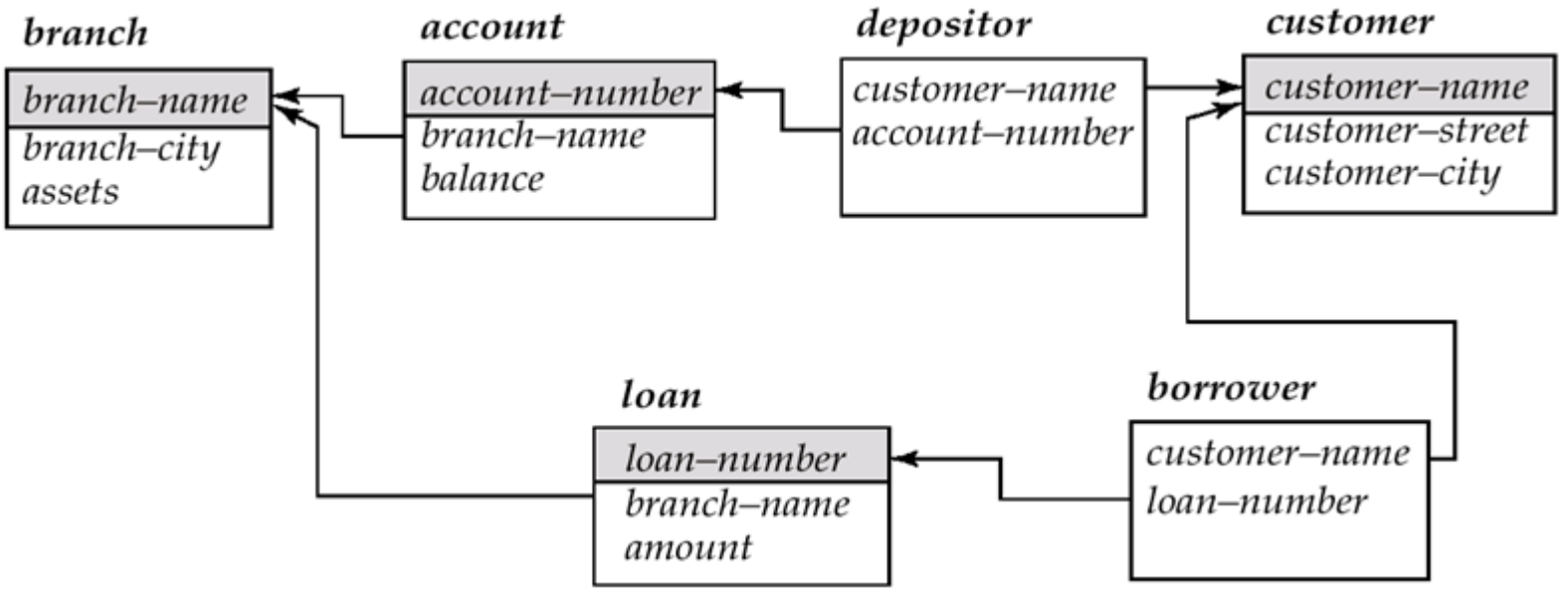
$$\{t \mid \exists s \in borrower(t[customer\text{-}name] = s[\text{customer-name}] \wedge \exists u \in loan(u[branch\text{-}name] = \text{“Perryridge”} \wedge u[loan\text{-}number] = s[\text{loan-}number])) \wedge \textbf{not } \exists v \in depositor\ (v[customer\text{-}name] = t[\text{customer-name}])\}$$



Example Queries

- Find the names of all customers who have an account at all branches located in Brooklyn:

$$\{t \mid \exists\, c \in \text{customer}\ (t[\text{customer-name}] = c[\text{customer-name}]) \wedge$$
$$\forall\, s \in \textit{branch}(s[\textit{branch-city}] = \text{“Brooklyn”} \Rightarrow$$
$$\exists\, u \in \textit{account}\ (\ s[\textit{branch-name}] = u[\text{branch-name}]$$
$$\wedge \exists\, s \in \textit{depositor}\ (\ t[\textit{customer-name}] = s[\text{customer-name}]$$
$$\wedge\ s[\textit{account-number}] = u[\text{account-number}]\ ))\ )\}$$



Safety of Expressions

- It is possible to write tuple calculus expressions that generate infinite relations.
- For example, $\{t \mid \neg\, t \in r\}$ results in an infinite relation if the domain of any attribute of relation r is infinite
- To guard against the problem, we restrict the set of allowable expressions to safe expressions.
- An expression $\{t \mid P(t)\}$ in the tuple relational calculus is *safe* if every component of t appears in one of the relations, tuples, or constants that appear in P

Domain Relational Calculus

- Each query is an expression of the form:

$$\{ < x_1, x_2, \ldots, x_n > \mid P(x_1, x_2, \ldots, x_n) \}$$

  - $x_1, x_2, \ldots, x_n$ represent domain variables
  - P represents a formula similar to that of the predicate calculus

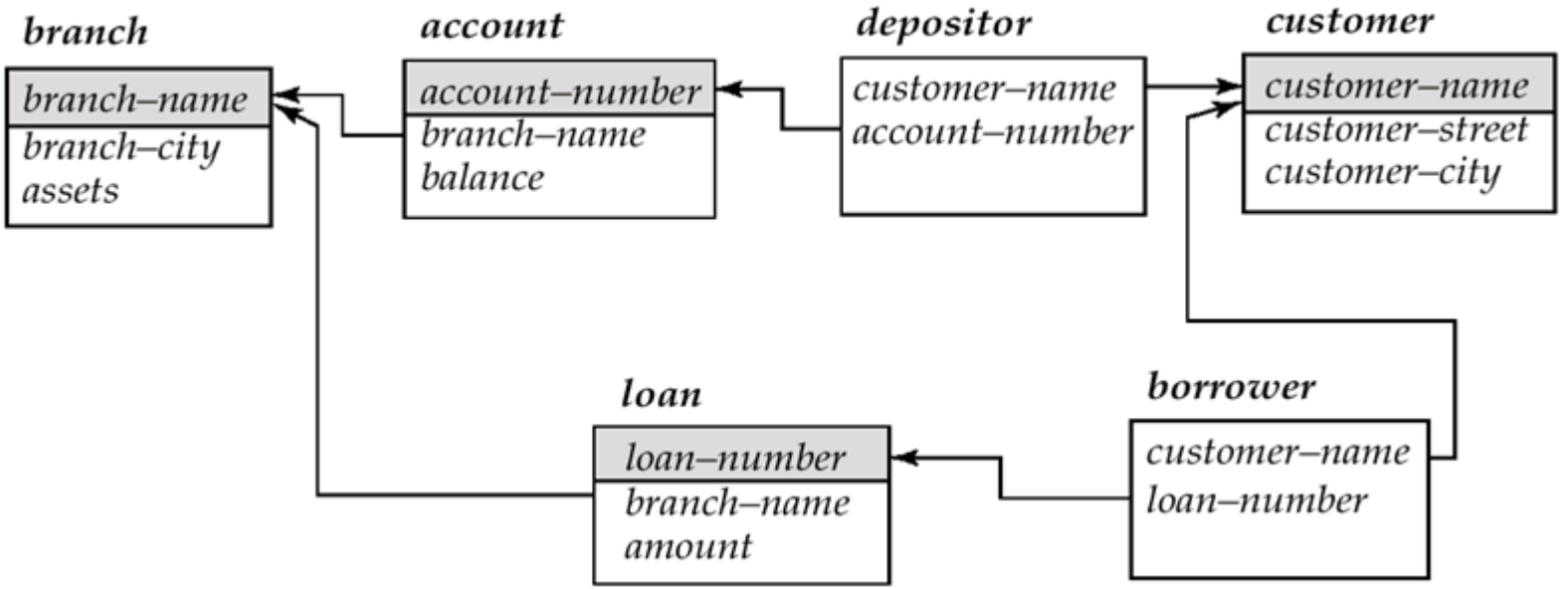
Example Queries

- Find the *branch-name*, *loan-number*, and *amount* for loans of over $1200

  $\{< l, b, a > \mid < l, b, a > \in loan \wedge a > 1200\}$

- Find the names of all customers who have a loan of over $1200

  $\{< c > \mid \exists\, l, b, a\, (< c, l > \in borrower \wedge < l, b, a > \in loan \wedge a > 1200)\}$



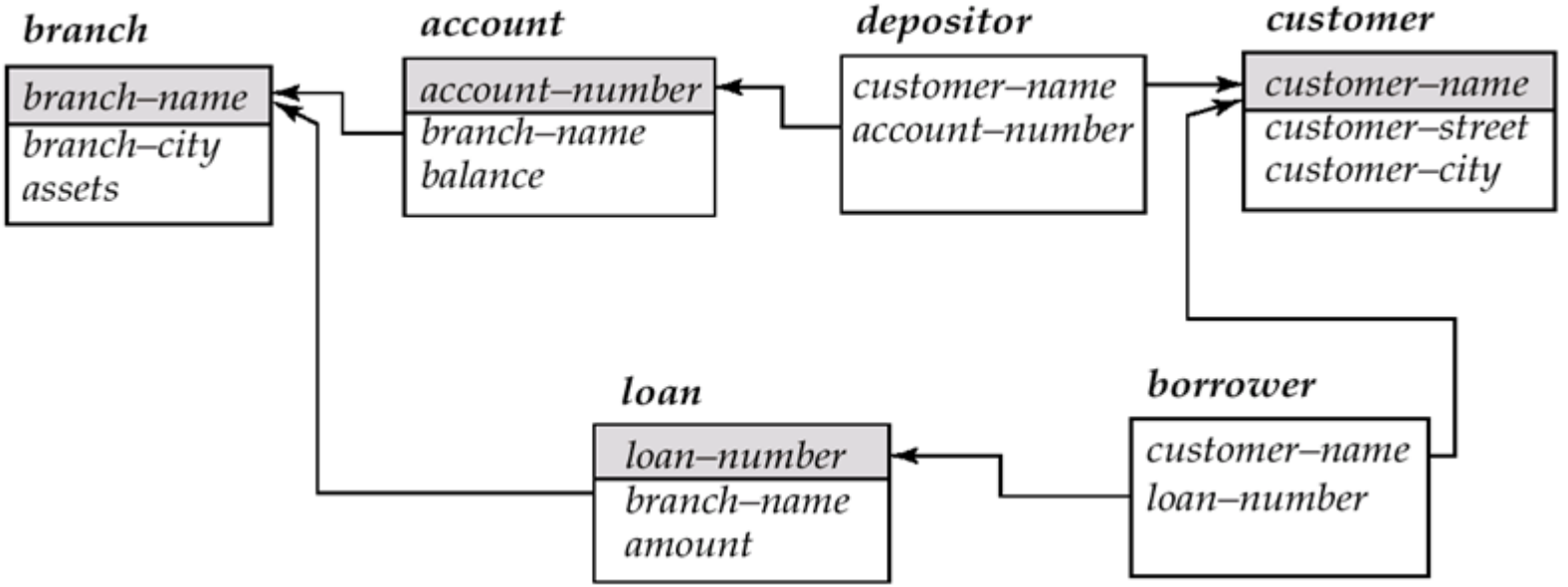
Example Queries

- Find the names of all customers who have a loan from the Perryridge branch and the loan amount:

  $\{< c, a > \mid \exists\, l\, (< c, l > \in borrower \wedge \exists b(< l, b, a > \in loan \wedge b = \text{“Perryridge”}))\}$

  or

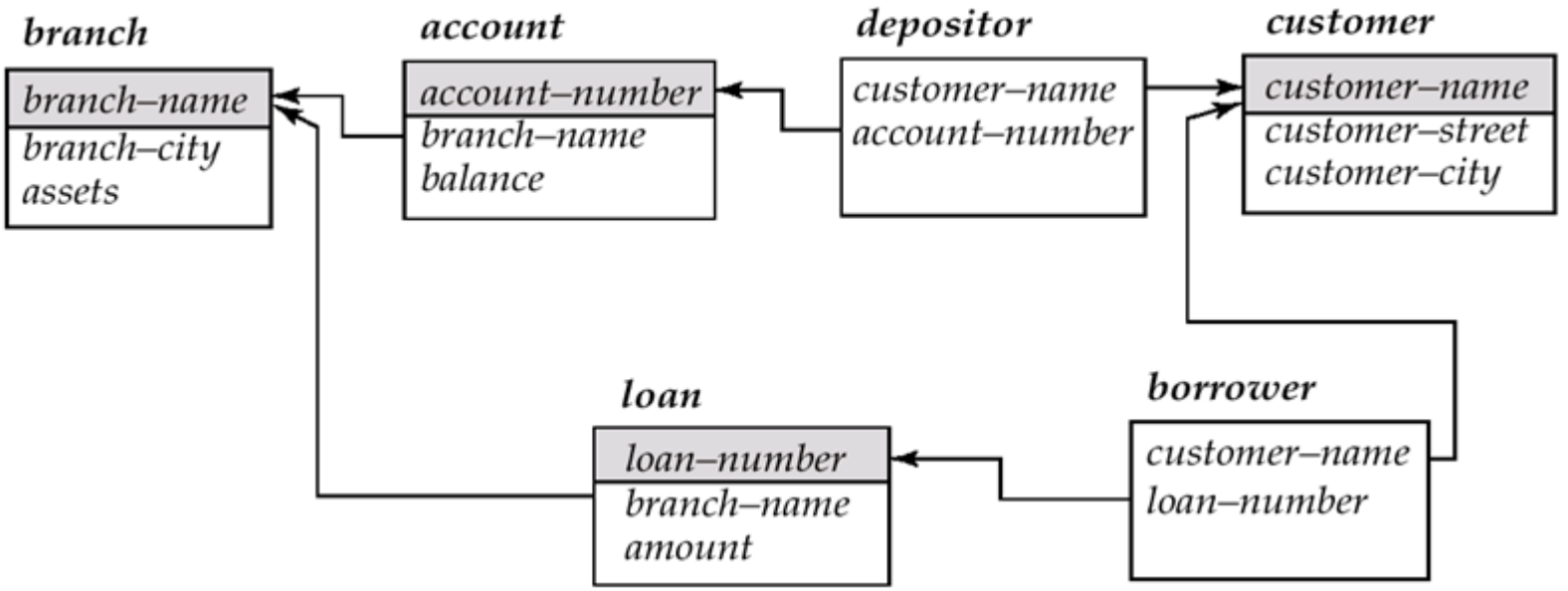
  $\{< c, a > \mid \exists\, l\, (< c, l > \in borrower \wedge < l, \text{“Perryridge”}, a > \in loan)\}$



Example Queries

- Find the names of all customers having a loan, an account, or both at the Perryridge branch:

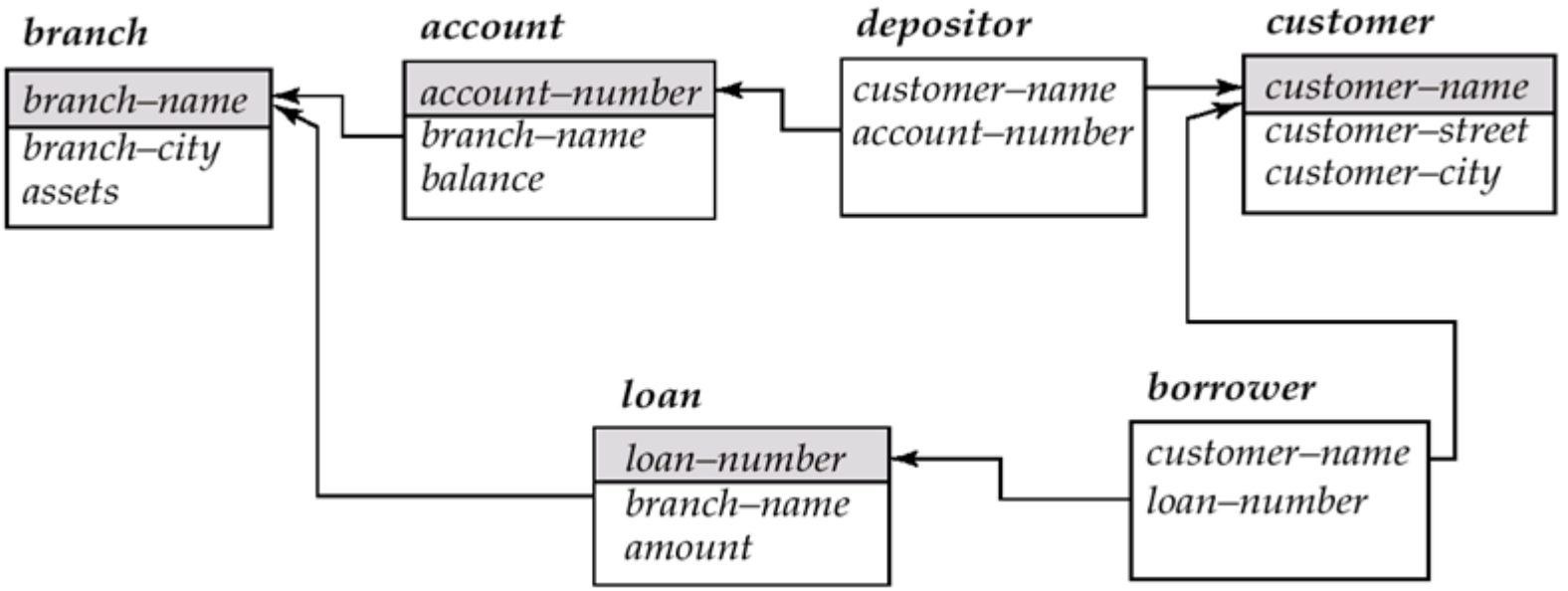
$$\{< c > \mid \exists\, l\, (\{< c, l > \in \textit{borrower} \wedge \exists\, b, a(< l, b, a > \in \textit{loan} \wedge b = \text{“Perryridge”})) \vee \exists\, a(< c, a > \in \textit{depositor} \wedge \exists\, b, n(< a, b, n > \in \textit{account} \wedge b = \text{“Perryridge”}))\}$$



Example Queries

- Find the names of all customers who have an account at all branches located in Brooklyn:

$$\{< c > \mid \exists\, n\, (< c, s, n > \in \text{customer}) \wedge$$
$$\forall\, x,y,z(< x, y, z > \in \textit{branch} \wedge y = \text{“Brooklyn”}) \Rightarrow$$
$$\exists\, a,b\, (< a, y, b > \in \textit{account} \wedge < c,a > \in \textit{depositor})\}$$



Safety of Expressions

$$\{ < x_1, x_2, \ldots, x_n > \mid P(x_1, x_2, \ldots, x_n)\}$$

It is safe if all of the following hold:

1. All values that appear in tuples of the expression are values from *dom*(P) (that is, the values appear either in P or in a tuple of a relation mentioned in P).
2. For every "there exists" sub-formula of the form $\exists\ x\ (P_1(x))$, the sub-formula is true if an only if $P_1(x)$ is true for all values x from *dom*(P_1).
3. For every "for all" sub-formula of the form $\forall$x $(P_1\ (x))$, the sub-formula is true if and only if $P_1(x)$ is true for all values x from *dom* (P_1).

THANK YOU