Database Development and Design (CPT201)

Lecture 4a: Relational Algebra

Dr. Wei Wang

Department of Computing

Learning Outcomes

- Six basic operators
  - select: σ
  - project: $\prod$
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ
- Additional Operations
  - Set intersection
  - Natural join
  - Division
  - Assignment

Relational Algebra, What and Why?

- Similar to normal algebra (as in 2+3*x-y), except we use relations as values instead of numbers, and the operations and operators are different.
- Not used as a query language in actual DBMSs. (SQL instead.)
- The inner, lower-level operations of a relational DBMS are, or are similar to, relational algebra operations. We need to know about relational algebra to understand query execution and optimisation in a relational DBMS.
- Some advanced SQL queries requires explicit relational algebra operations, most commonly *outer join*.
- Relations are seen as *sets of tuples*, which means that no duplicates are allowed. SQL behaves differently in some cases. Remember the SQL keyword distinct.
- SQL is *declarative*, which means that you tell the DBMS *what* you want, but not *how* it is to be calculated. A C++ or Java program is *procedural*, which means that you have to state, step by step, exactly how the result should be calculated. Relational algebra is (more) procedural than SQL. (Actually, relational algebra is mathematical expressions.)

Concepts and operations from set theory

- Relations in relational algebra are seen as sets of tuples, so we can use basic set operations.
  - set
  - element
  - no duplicate elements (but: multiset = bag)
  - no order among the elements (but: ordered set)
  - subset
  - proper subset (with fewer elements)
  - superset
  - union
  - intersection
  - set difference
  - Cartesian product (cross-product)

Formal Definition

- A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, P is a predicate on attributes in E_1
  - $\prod_S(E_1)$, S is a list consisting of some of the attributes in E_1
  - $\rho_x (E_1)$, x is the new name for the result of E_1

Select Operation– Example

Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Select Operation

- Notation: $\sigma_{p}(r)$
- p is called the **selection predicate**
- Defined as:

$$\sigma_p(\mathbf{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of **terms** connected by : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
Each **term** is one of:

<attribute> *op* <attribute> or <constant>

where *op* is one of: $=, \neq, >, \geq, <, \leq$

- Example of selection:

$$\sigma_{branch\_name=\text{"Perryridge"}}(account)$$

Project Operation – Example

Relation *r*

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\prod_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Project Operation

- Notation:

$$\prod_{A_1, A_2, \ldots, A_k} (r)$$

  where *A_1, A_2* are attribute names and *r* is a relation name.

- The result is defined as the relation of *k* columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: to eliminate the *branch_name* attribute of *account* (branch_name, account_name, balance)

$$\prod_{account\_number,\ balance} (account)$$

Union Operation – Example

Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

r ∪ s:

A	B
α	1
α	2
β	1
β	3

Union Operation

- Notation: $r \cup s$
- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid.

  1. r, s must have the *same* **arity** (same number of attributes)
  2. The attribute domains must be **compatible** (example: 2nd column of r deals with the same type of values as does the 2nd column of s)

- Example: to find all customers with either an account or a loan

$$\Pi_{customer\_name}(depositor) \cup \Pi_{customer\_name}(borrower)$$

Set Difference Operation – Example

Relations r, s

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r - s$

A	B
α	1
β	1

Set Difference Operation

- Notation $r - s$
- Defined as:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

- Set differences must be taken between **compatible** relations.
  - r and s must have the same arity
  - attribute domains of r and s must be compatible

Cartesian-Product Operation – Example

Relations *r, s*

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

r x *s*

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Cartesian-Product Operation

- Notation $r \times s$
- Defined as:

  $r \times s = \{t\, q \mid t \in r \textbf{ and } q \in s\}$
- Assume that attributes of r(R) and s(S) are disjoint, that is, $R \cap S = \emptyset$.
- If attributes of *r(R)* and *s(S)* are not disjoint, then renaming must be used.

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$

$r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

$\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- Example:

$$\rho_X(E)$$

returns the expression *E* under the name *X*

- If a relational-algebra expression *E* has arity *n*, then

$$\rho_{x(A_1, A_2, \ldots, A_n)}(E)$$

returns the result of expression *E* under the name *X*, and with the attributes renamed to A_1, A_2, …., A_n.

Banking Example

branch (branch_name, branch_city, assets)

customer (customer_name, customer_street, customer_city)

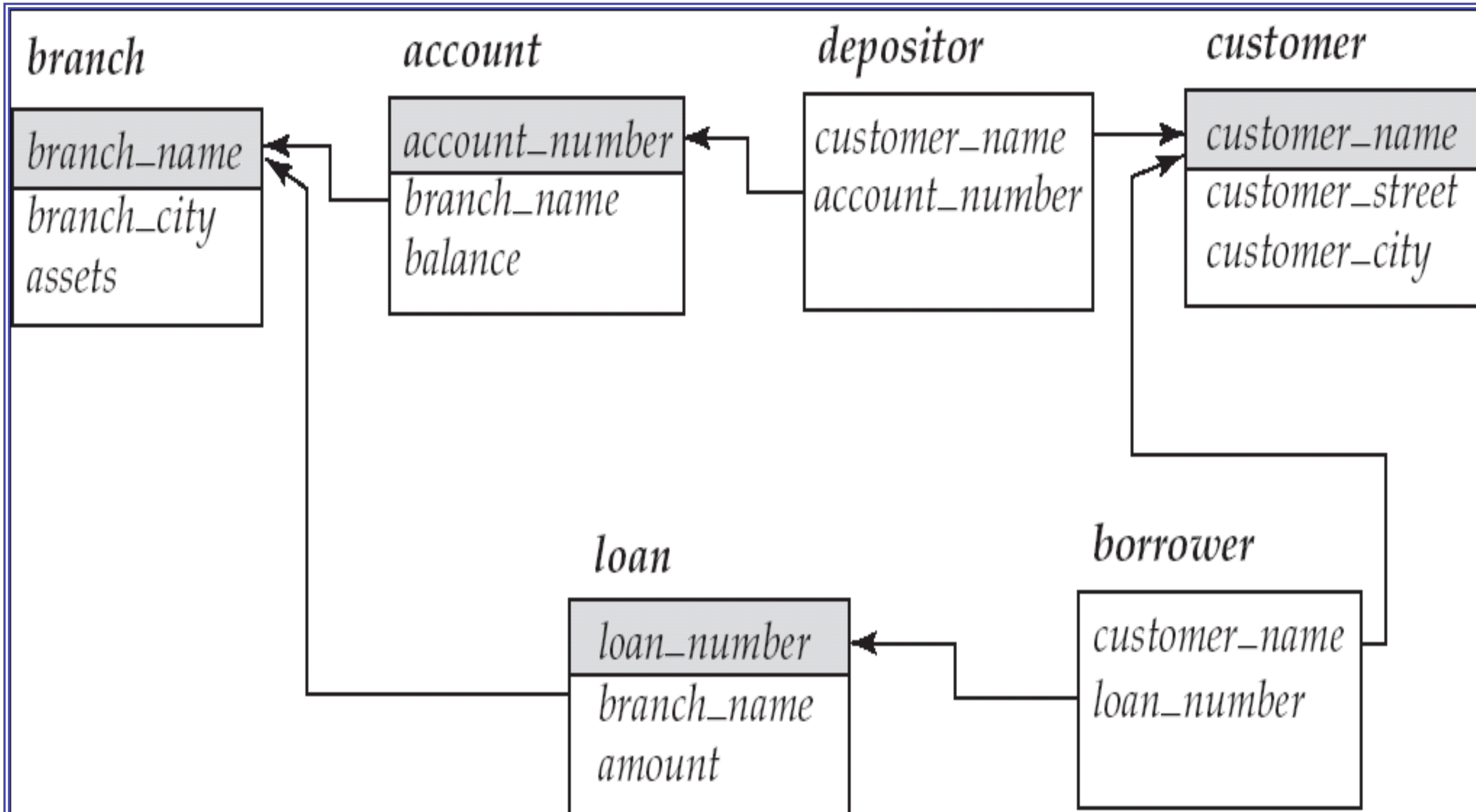
account (account_number, branch_name, balance)

loan (loan_number, branch_name, amount)

depositor (customer_name, account_number)

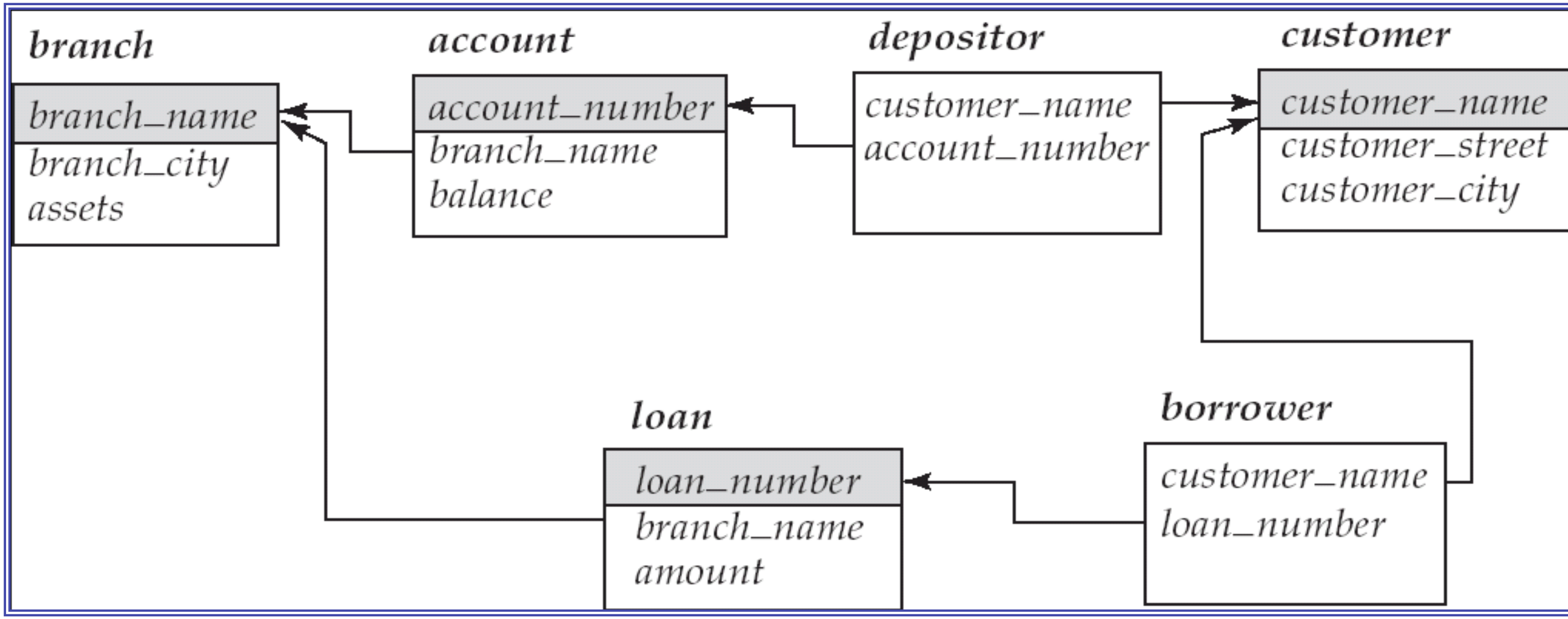
borrower (customer_name, loan_number)

Banking Example



Xi'an Jiaotong-Liverpool University 西交利物浦大學

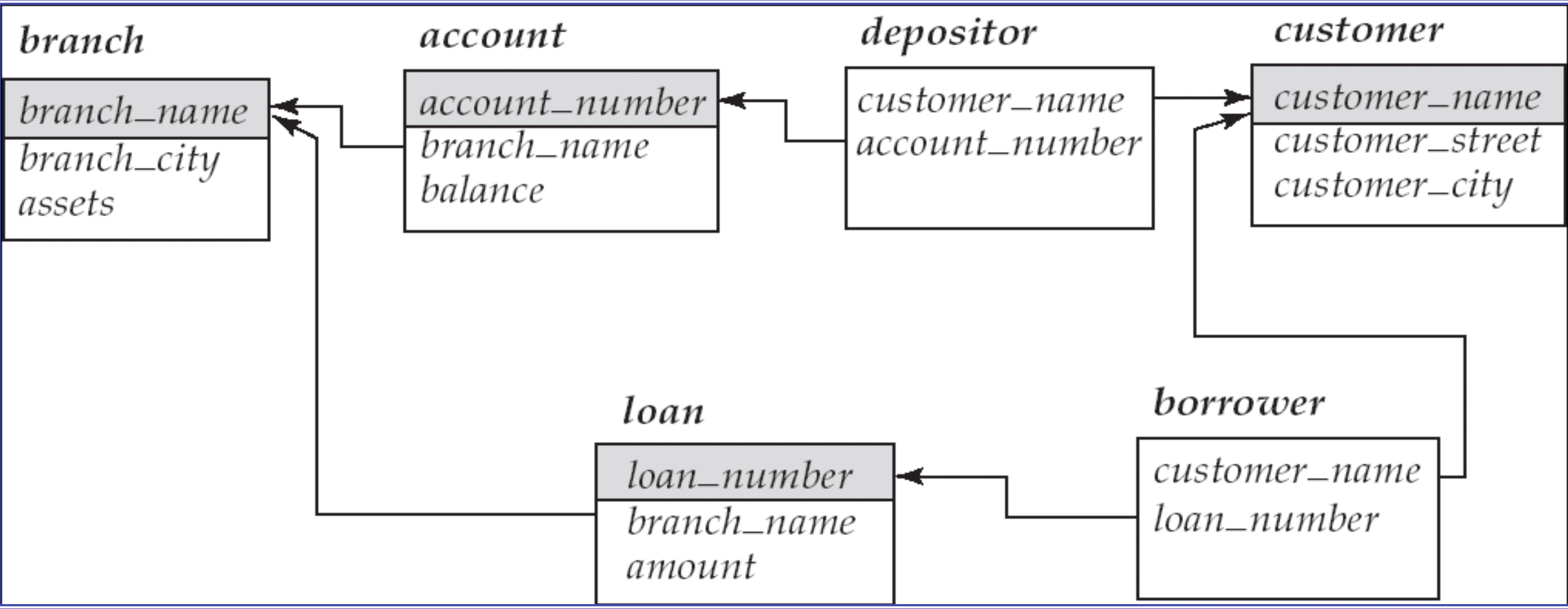
Example Queries



Find all loans of over $1,200

$$\sigma_{amount > 1,200}\ (loan)$$

Example Queries cont'd



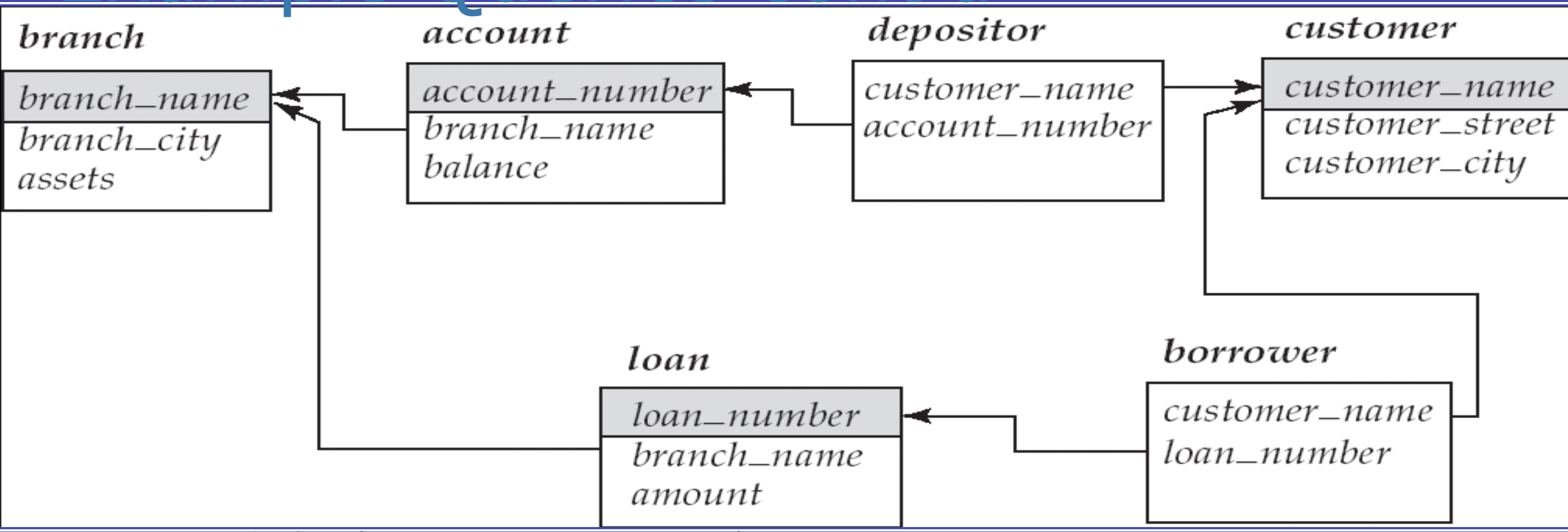
- Find the names of all customers who have a loan at the Perryridge branch.

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{"Perryridge"}}(\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$$

OR

$$\Pi_{customer\_name}(\sigma_{loan.loan\_number = borrower.loan\_number}((\sigma_{branch\_name = \text{"Perryridge"}}(loan)) \times borrower))$$

Example Queries cont'd



- Find the largest account balance
  - Strategy:
    - Find those balances that are *not* the largest
      - Rename *account* relation as *d* so that we can compare each account balance with all others
    - Use set difference to find those account balances that were *not* found in the earlier step.

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

Additional Operations

- We define additional operations that do not add any power to the relational algebra, but that simplify common queries.
  - Set intersection
  - Natural join
  - Division
  - Assignment

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$
- Assume:
  - r, s have the *same arity*
  - attributes of r and s are compatible
- Note: $r \cap s = r - (r - s)$

Set-Intersection Operation – Example

Relation *r, s*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r \cap s$

A	B
α	2

Natural-Join Operation

- Notation: r $\bowtie$ s
- Let *r* and *s* be relations on schemas *R* and *S* respectively. Then, r $\bowtie$ s is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from *r* and t_s from *s*.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple *t* to the result, where
    - *t* has the same value as t_r on *r*
    - *t* has the same value as t_s on *s*
- Example:

  R = (*A, B, C, D*)

  S = (*E, B, D*)
  - Result schema = (*A, B, C, D, E*)
  - $r \bowtie s$ is defined as:

$$\Pi_{r.A,\ r.B,\ r.C,\ r.D,\ s.E}(\sigma_{r.B = s.B \wedge r.D = s.D}(r \times s))$$

Natural Join Operation – Example

Relations r, s

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	$\in$

s

$r \bowtie s$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Division Operation

- Notation: $r \div s$
- Often suited to queries that include the phrase "for all".
- Let r and s be relations on schemas R and S respectively where
  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

  The result of $r \div s$ is a relation on schema

  $R - S = (A_1, \ldots, A_m)$

  $$r \div s = \{ t \mid t \in \prod_{R-S}(r) \wedge \forall u \in s (tu \in r) \}$$

  Where tu means the concatenation of tuples t and u to produce a single tuple

Division Operation – Example

Relations *r, s*

r

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
$\in$	6
$\in$	1
β	2

s

B
1
2

$r \div s$:

A
α
β

Another Division Example

Relations *r, s*

r

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

s

D	E
a	1
b	1

$r \div s$

A	B	C
α	a	γ
γ	a	γ

Division Operation cont'd

- Property
  - Let $q = r \div s$
  - Then q is the largest relation satisfying $q \times s \subseteq r$
- Definition in terms of the basic algebra operation
  Let $r(R)$ and $s(S)$ be relations, and let $S \subseteq R$

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$$

To see why

- $\Pi_{R-S,S}(r)$ simply reorders attributes of r
- $\Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$ gives those tuples t in

  $\Pi_{R-S}(r)$ such that for some tuple $u \in s$, $tu \notin r$.

Assignment Operation

- The assignment operation (←) provides a convenient way to express complex queries.
  - Write query as a sequential program consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as a result of the query.
  - Assignment must always be made to a temporary relation variable.
- Example: Write $r \div s$ as

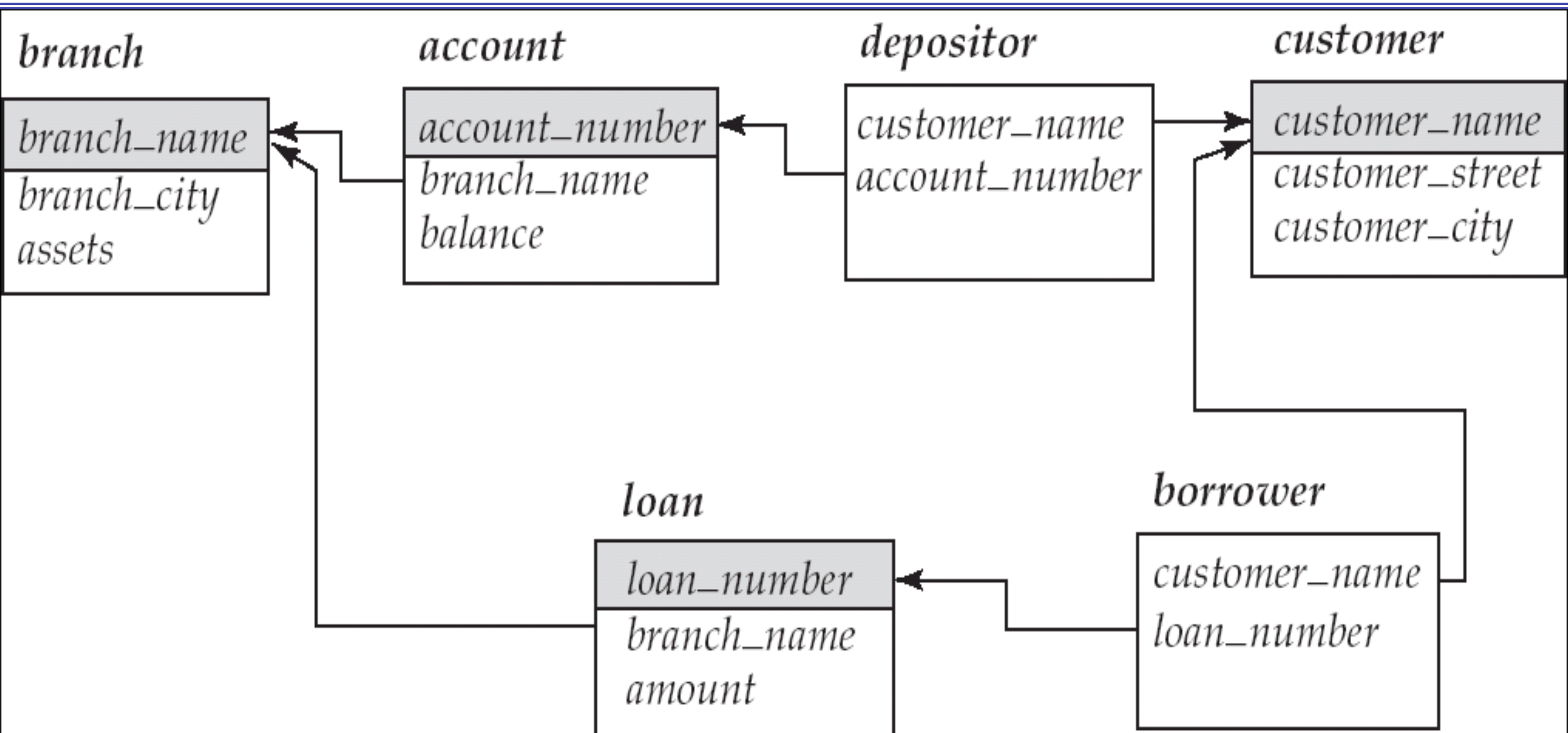
$$temp1 \leftarrow \prod_{R-S}(r)$$

$$temp2 \leftarrow \prod_{R-S}((temp1 \times s) - \prod_{R-S,S}(r))$$

$$result = temp1 - temp2$$

  - The result to the right of the ← is assigned to the relation variable on the left of the ←.
  - May use variable in subsequent expressions.

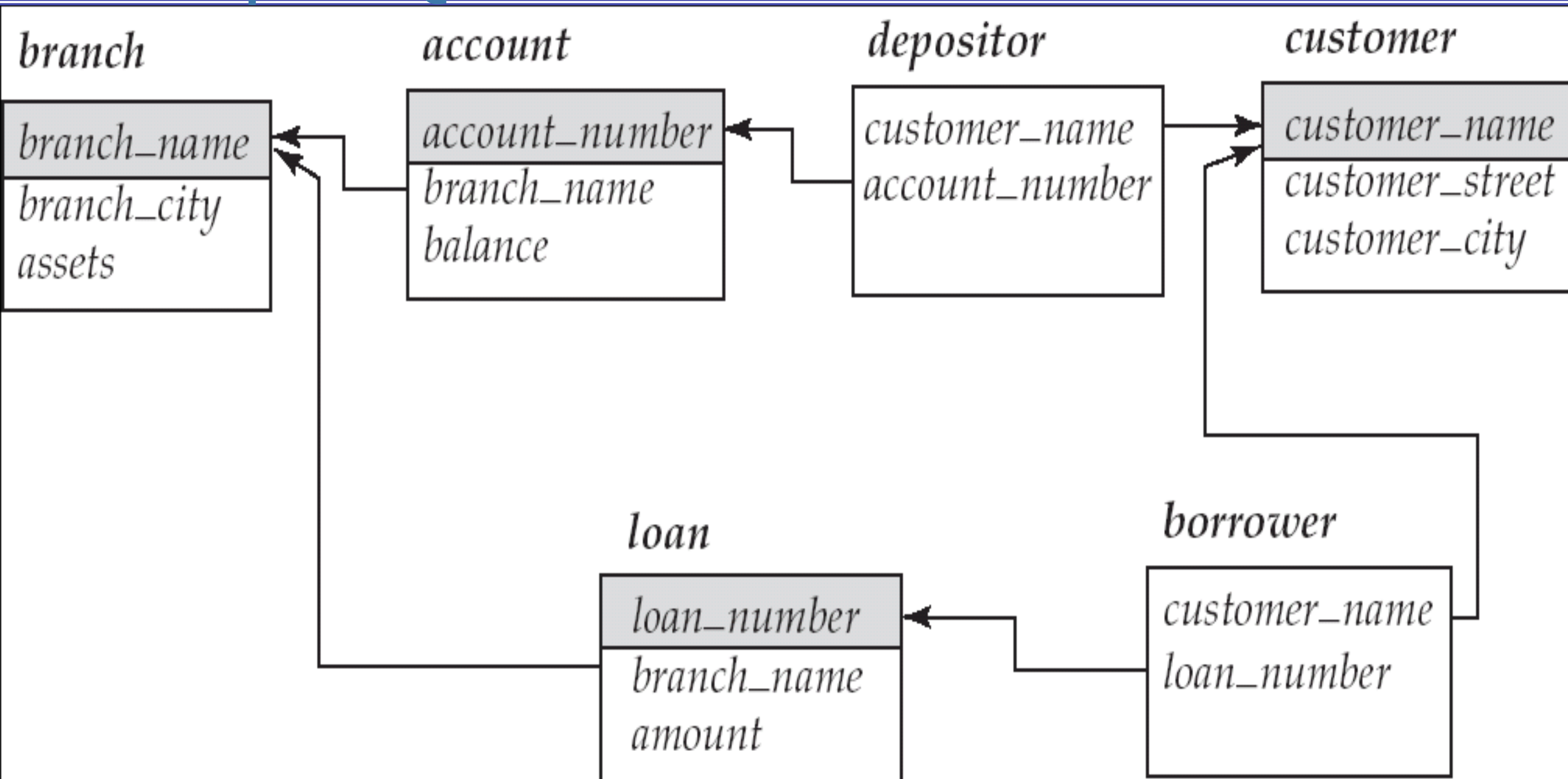
Example Queries cont'd



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

$$\Pi_{customer\_name}\,(borrower) \cap \Pi_{customer\_name}\,(depositor)$$

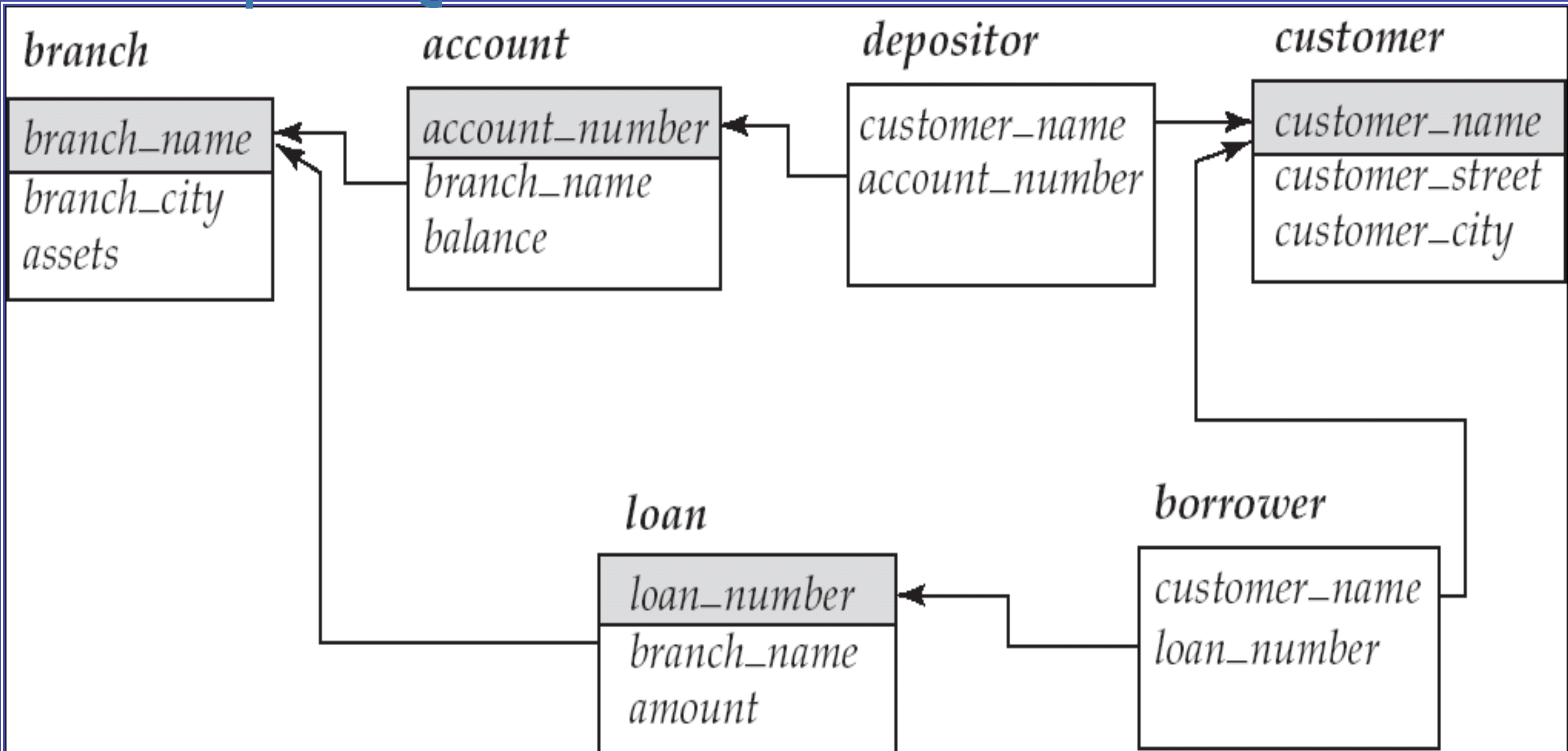
Example Queries cont'd



Find the name of all customers names, their load numbers and loan amount

$$\Pi_{customer\_name,\ loan\_number,\ amount}\ (borrower \bowtie loan)$$

Example Queries cont'd

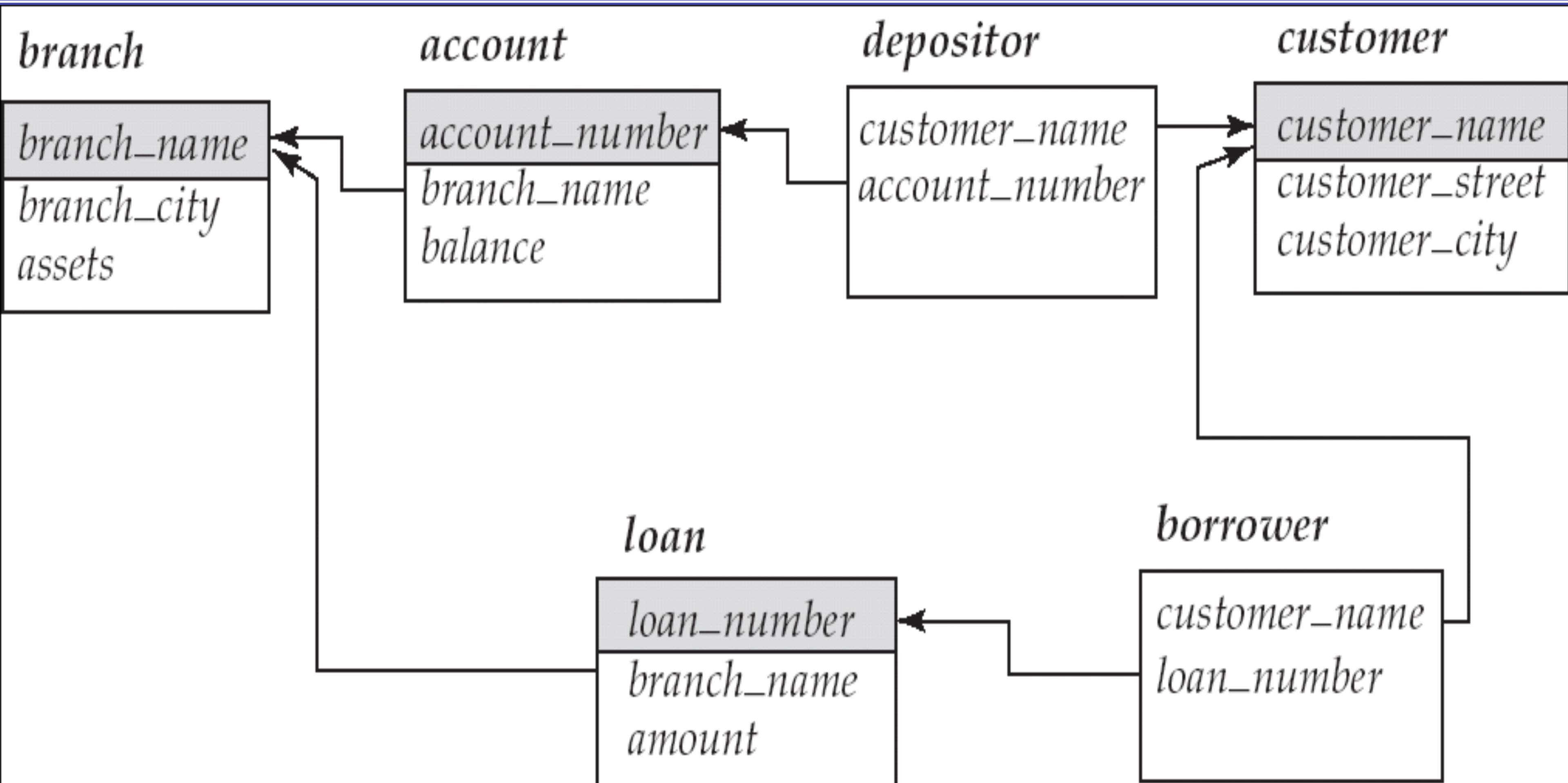


Find names of all customers who have an account from both the "Downtown" and the "Uptown" branches.

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{"Downtown"}}(depositor \bowtie account)) \cap$$

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{"Uptown"}}(depositor \bowtie account))$$

Example Queries cont'd



Find names of all customers who have an account at all branches located in Brooklyn city.

$$\Pi_{customer\_name,\ branch\_name}(depositor \bowtie account) \div \Pi_{branch\_name}(\sigma_{branch\_city = \text{"Brooklyn"}}(branch))$$

End of Lecture

- Summary
  - Basic relational algebra operators
  - Additional relational algebra operators
  - Example queries

- Reading
  - Textbook 6^{th} edition, chapter 2.6, 6.1
  - Textbook 7^{th} edition, chapter 2.6