CHAPTER 2
RELATIONAL MODEL

Chapter 2: Relational Model

- Structure of Relational Databases
- Fundamental Relational Algebra Operations
- Additional Relational Algebra Operations
- Extended Relational-Algebra-Operations
- Null Values
- Modification of the Database

Example of a Relation

account-number	*branch-name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Basic Structure

- Formally,

  given sets D_1, D_2, …. D_n a *relation r* is a subset of D_1 x D_2 x … x D_n

  $R \subseteq D_1$, X ….. X D_n (n: degree of R)

  or

  $R = \{ <d_1, ..., d_n> \mid d_1 \in D_1, ..., d_n \in D_n \}$ (set of tuples)

- Example: *customer-name* = {Jones, Smith, Curry, Lindsay}

  customer-street = {Main, North, Park}

  customer-city = {Harrison, Rye, Pittsfield}

  Then r = { (Jones, Main, Harrison), (Smith, North, Rye),

  (Curry, North, Rye), (Lindsay, Park, Pittsfield)}

  is a relation over *customer-name* x *customer-street* x *customer-city*

Attribute Types

- Each attribute of a relation has a name
- The set of allowed values for each attribute is called the domain of the attribute
- Attribute values are (normally) required to be atomic, that is, indivisible
  - E.g. multivalued attribute values are not atomic
  - E.g. composite attribute values are not atomic
- The special value *null* is a member of every domain
- The null value causes complications in the definition of many operations
  - we shall ignore the effect of null values in our main presentation and consider their effect later

Relation Schema

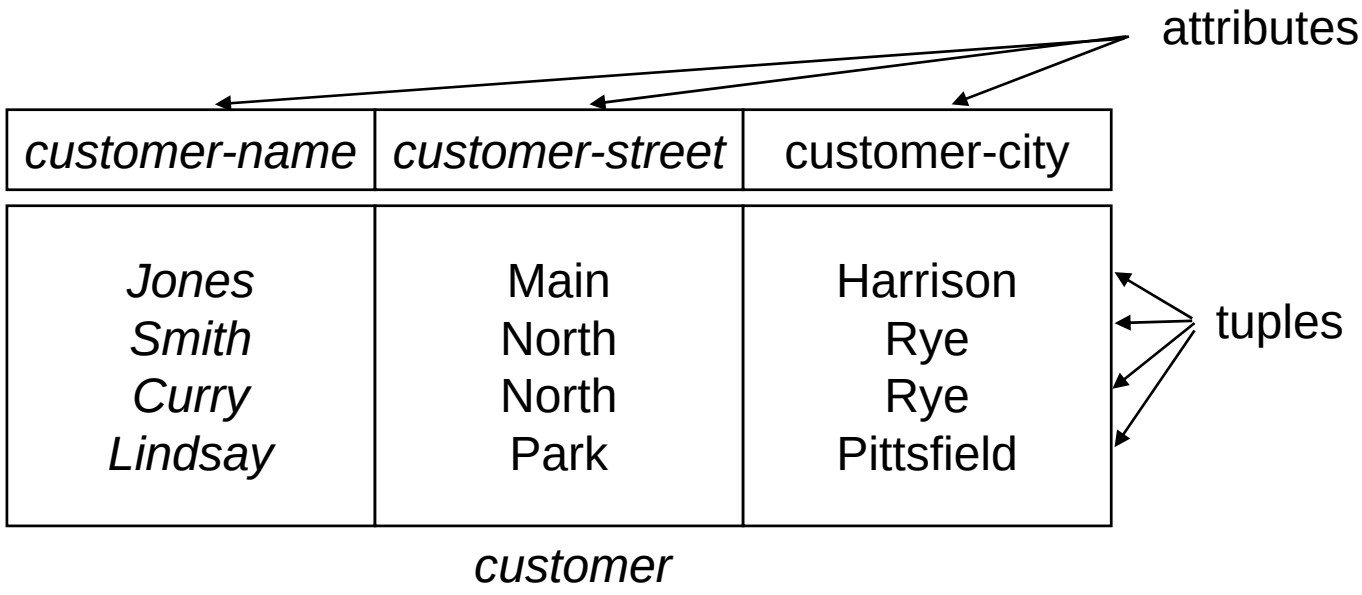
- $A_1, A_2, \ldots, A_n$ are *attributes*
- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

  E.g. *Customer-schema* = (*customer-name, customer-street, customer-city*)
- *r*(R) is a *relation* (*variable*) on the *relation schema* R

  E.g. *customer(Customer-schema)*

Relation Instance

- The current values (*relation instance*) of a relation are specified by a table
- An element *t* of *r* is a *tuple*, represented by a *row* in a table



Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- E.g. *account* relation with unordered tuples

account-number	*branch-name*	*balance*
A-101	Downtown	500
A-215	Mianus	700
A-102	Perryridge	400
A-305	Round Hill	350
A-201	Brighton	900
A-222	Redwood	700
A-217	Brighton	750

Relational Database

- A database consists of multiple relations
- Information about an enterprise is broken up into parts, with each relation storing one part of the information

  E.g.: *account* : information about accounts

  customer : information about customers

  depositor : information about which customer owns which account

- Storing all information as a single relation such as results in
  - repetition of information (e.g. two customers own an account)
  - the need for null values (e.g. represent a customer without an account)

  bank(*account-number, balance, customer-name*, ...)

- Relational database design (Chapter 6 & 7) deals with how to design relational schemas

The *customer* relation & *depositor* relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

customer-name	*account-number*
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

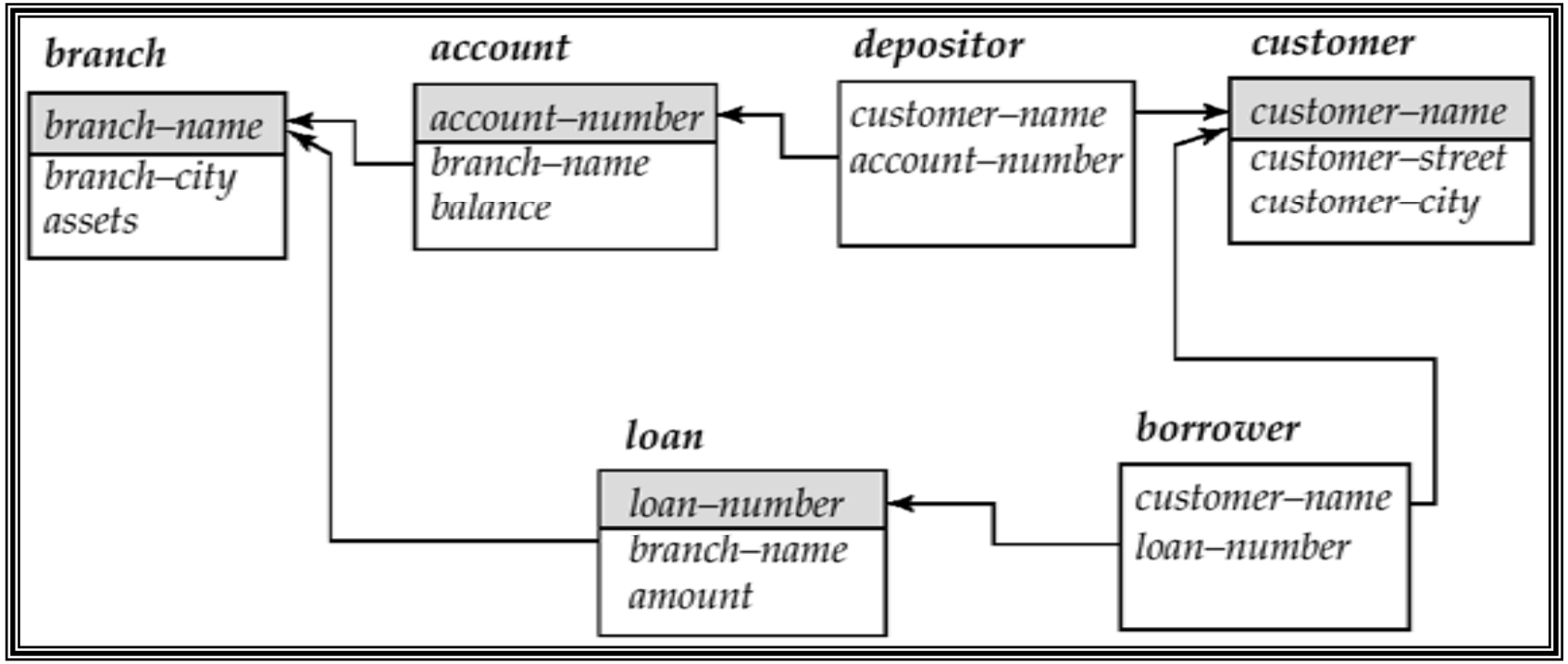
Keys

- Let $K \subseteq R$
- *K* is a *superkey* of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*.
- By "possible *r*" we mean a relation *r* that could exist in the enterprise we are modeling.
  - Example: {*customer-name, customer-street*} and
    {*customer-name*}

    are both superkeys of *Customer*, if no two customers can possibly have the same name.
- *K* is a *candidate key* if *K* is minimal
  - Example: {*customer-name*} is a candidate key for *Customer*,

    since it is a superkey, and no subset of it is a superkey.

Schema Diagram for the Banking Enterprise



Query Languages

- Language in which user requests information from the database.
- Categories of languages
  - procedural
    - specify what data are needed and how to get those data
  - nonprocedural
    - declarative
    - specify only what data are needed
- "Pure" languages:
  - Relational Algebra
  - Tuple Relational Calculus
  - Domain Relational Calculus
- Pure languages form underlying basis of query languages that people use.

Relational Algebra

- Algebra : operators and operands
  - Relational algebra
    - operands : relations
    - operators : basic operators (+ additional operations)
      - take two or more relations as inputs and give a new relation as a result.
- Procedural language
- 6 Fundamental Operators
  - select
  - project
  - union
  - set difference
  - Cartesian product
  - rename

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:

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

  - 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:

$$\sigma_{\textit{branch-name}=\text{“Perryridge”}}(\textit{account})$$

Project Operation – Example

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}$ (*r*)

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Project Operation

- Notation:

$$\Pi_{A1, A2, \ldots, Ak}(r)$$

  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*

- $$\Pi_{account\text{-}number,\ balance}(account)$$

Union Operation – Example

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cup s$:

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 (Union compatible)
  1. r, s must have the *same arity* (same number of attributes)
  2. The attribute domains must be *compatible* (e.g., 2nd column of r deals with the same type of values as does the 2nd column of s)
     - E.g.: to find all customers with either an account or a loan

$$\Pi_{customer\text{-}name}(depositor) \cup \Pi_{customer\text{-}name}(borrower)$$

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:

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

- 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	19	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:

$$r \times s = \{ t\, q \mid t \in r \text{ and } q \in s \}$$

- Assume that attributes of $r(R)$ and $s(S)$ are disjoint; i.e., $R \cap S = \varnothing$.
  - If not, renaming of attributes is needed.

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	19	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	20	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.

- $\rho_N (E)$ returns the expression E under the name N

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

$$\rho_{N\ (A1,\ A2,\ \ldots,\ An)} (E)$$

returns the result of expression E under the name N, and with the attributes renamed to $A1, A2, \ldots, An$.

Banking Example

branch (branch-name, branch-city, assets)

customer (customer-name, customer-street, customer-only)

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 all loans of over \$1200

$$\sigma_{amount > 1200}\ (loan)$$

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

$$\Pi_{loan\text{-}number}\ (\sigma_{amount > 1200}\ (loan))$$

Example Queries

- Find the names of all customers who have a loan, an account, or both, from the bank

$$\Pi_{customer\text{-}name}(borrower) \cup \Pi_{customer\text{-}name}(depositor)$$

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

$$\Pi_{customer\text{-}name}(borrower) \cap \Pi_{customer\text{-}name}(depositor)$$

Example Queries

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

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

- 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.

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

  - $\Pi_{customer\text{-}name}(depositor)$

Example Queries

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

Query 1

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

Query 2

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

Example Queries

- Find the largest account balance
  - Rename *account* relation as *d*
  - The query is:

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

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
  - $\rho_N (E_1)$, N is the new name for the result of E_1

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 union compatibility:
  - 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. The result is a relation on schema $R \cup S$ which is obtained by considering 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 $R \cap S$, a tuple t is added 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:

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

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

Properties

- $\Pi_{A_1,\ldots,A_k}(r) \cap \Pi_{A_1,\ldots,A_k}(s) = \Pi_{A_1,\ldots,A_k}(r \bowtie s)$
- $(r \bowtie s) \bowtie t = r \bowtie (s \bowtie t)$
- *If* $R \cap S = \varnothing$ *then* $r \bowtie s = r \times s$ (*important*)
- *If* $R = S$ *then* $r \bowtie s = r \cap s$

- Theta Join
  - combine selection with Cartesian product

$$r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$$

where θ is a predicate on $R \times S$

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 the variable in subsequent expressions.

Example Queries

- Find all customers who have an account from at least the "Downtown" and the "Uptown" branches.
  - Query 1

$$\Pi_{customer\text{-}name}(\sigma_{branch\text{-}name=\text{“Downtown”}}(depositor \bowtie account)) \cap \Pi_{customer\text{-}name}(\sigma_{branch\text{-}name=\text{“Uptown”}}(depositor \bowtie account))$$

  - Query 2

$$\Pi_{customer\text{-}name,\ branch\text{-}name}(depositor \bowtie account) \div \rho_{temp(branch\text{-}name)}(\{(\text{“Downtown”}), (\text{“Uptown”})\})$$

Example Queries

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

$$\Pi_{customer\text{-}name,\ branch\text{-}name}\ (depositor \bowtie account) \div \Pi_{branch\text{-}name}\ (\sigma_{branch\text{-}city = \text{“Brooklyn”}}\ (branch))$$

END OF CHAPTER 2