# 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 |

# Attribute Types

- Each attribute of a relation has a name
- Set of allowed values: **domain** of the attribute
- Values (normally) required to be atomic;
  - indivisible
  - e.g., *Downtown*, but not *{Downtown, Mianus}*
- Domain is said to be atomic if all its members are atomic
- Special value **null**: member of every domain
  - unknown or non-existent value
- 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

| *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 |

# Domains and Tuples

- Domain $D_1$: set of all account numbers, similarly $D_2$, $D_3$
- Every row: tuple of 3 values ($v_1$, $v_2$, $v_3$)
- Table *account* $\in$ all such tuples

# Relation Schema

- Given domains $D_1$, $D_2$, …. $D_n$ a **relation** *r* is a subset of

  $D_1$ x $D_2$ x … x $D_n$ (*cartesian product*)

  Thus, a relation is a set of tuples (a1, a2, …, an)

  where each ai $\in$ Di

- Schema of a relation consists of
  - attribute definitions
    - name
    - type/domain
  - integrity constraints

**Tuple** ⟶ **Row**

**Relation** ⟶ **Table**

Math General

# Schema and Relations

`Account_schema = (account_number, branch_name, balance)`

Schema

`account(Account_schema)`

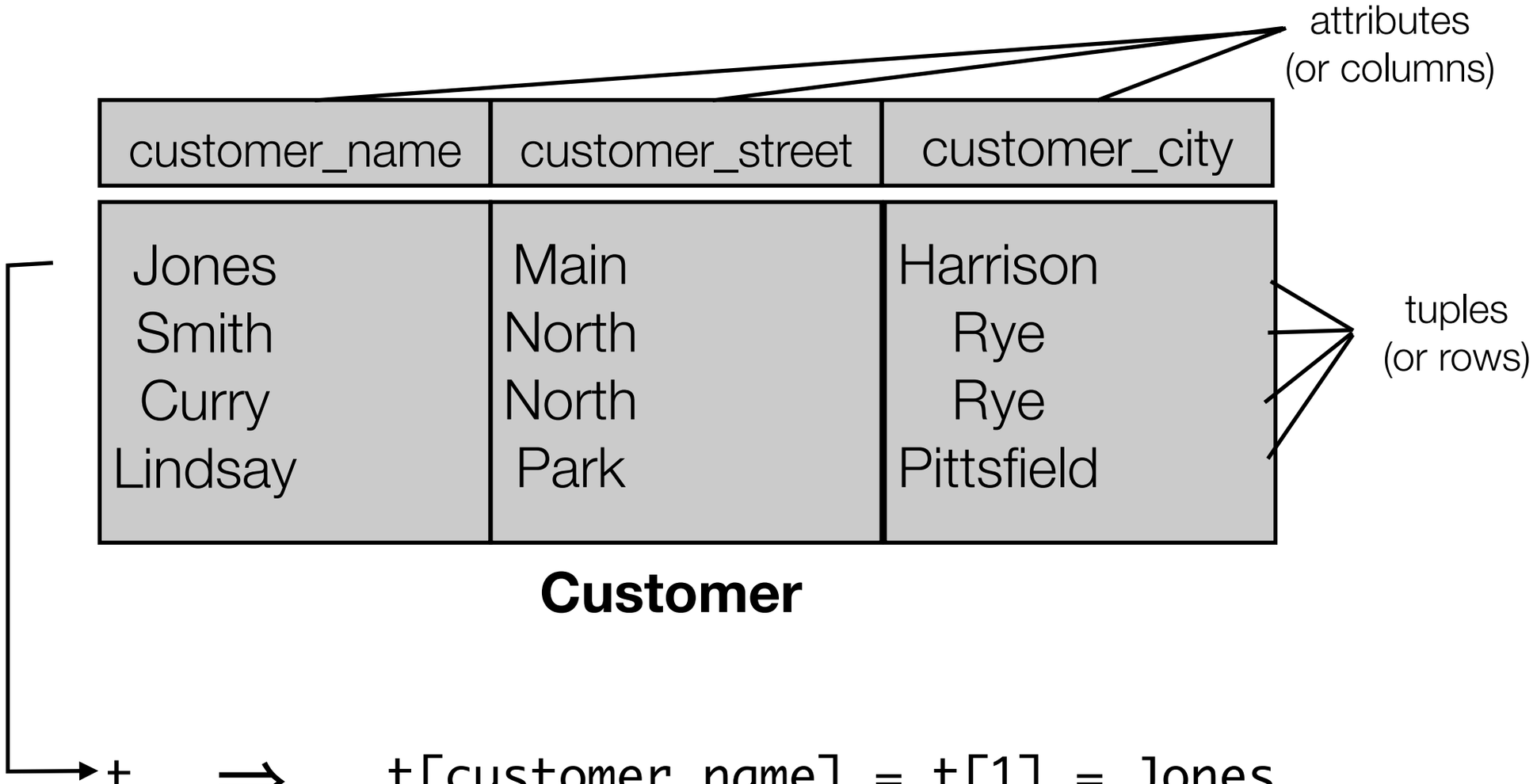
Relation from a schema

| *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 |

Relation **instance**

# 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
- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)




**Customer**

# 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

  `depositor` : which customer owns which account

  `customer` : information about customers

# The `customer` 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_schema = (customer_name, customer_street, customer_city)
```

# The `depositor` Relation

| *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 |

```
Depositor_schema = (customer_name, account_number)
```

# Why Split Information Across Relations?

- Storing all information as a single relation such as

  ```
  Bank_schema = (account_number, balance, customer_name, ..)
  ```

- Results in
  - repetition of information
    - e.g.,if two customers own an account (What gets repeated?)
  - the need for null values
    - e.g., to represent a customer without an account
- Normalization theory (Chapter 7) deals with how to design relational schemas