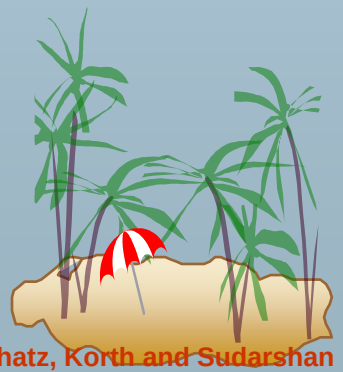
Chapter 3: Relational Model

- Structure of Relational Databases



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

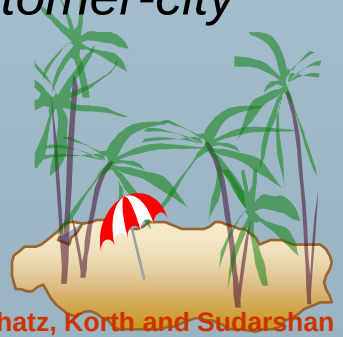
Basic Structure

- 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
  Thus a relation is a set of n-tuples (a_1, a_2, …, a_n) where each $a_i \in D_i$
- Example: if

  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

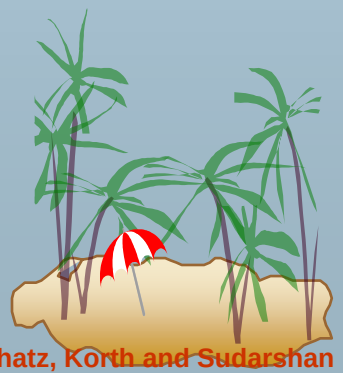
- Each attribute of a relation has a name
- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**, that is, indivisible
  - E.g. multivalued attribute values are not atomic
  - E.g. composite attribute values 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*

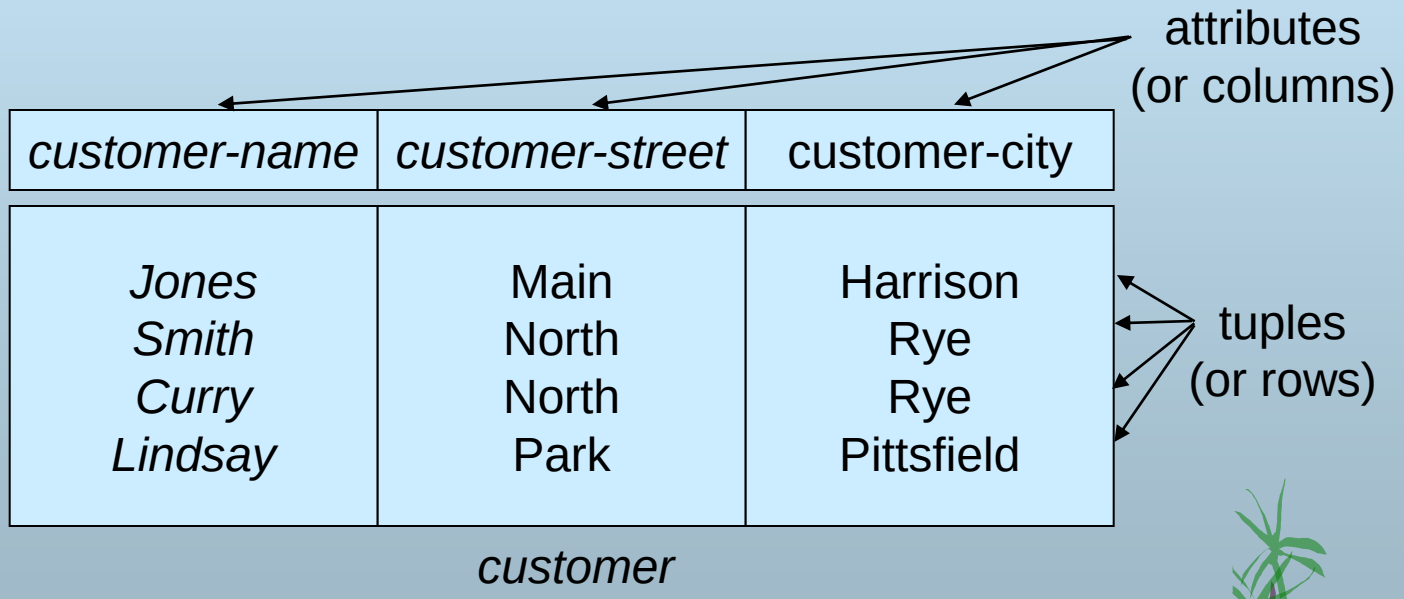
  E.g. *Customer-schema* = (*customer-name, customer-street, customer-city*)
- *r*(*R*) is a *relation* on the *relation schema R*

  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

Database

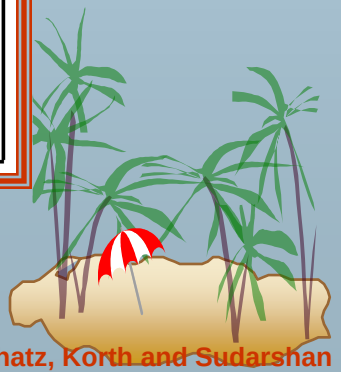
- A database consists of multiple relations
- Information about an enterprise is broken up into parts, with each relation storing one part of the information

  E.g.: *account* : stores information about accounts
  depositor : stores information about which customer owns which account
  customer : stores information about customers

- Storing all information as a single relation such as
  bank(*account-number, balance, customer-name*, ..)
  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)
- Normalization theory (Chapter 7) deals with how to design relational schemas

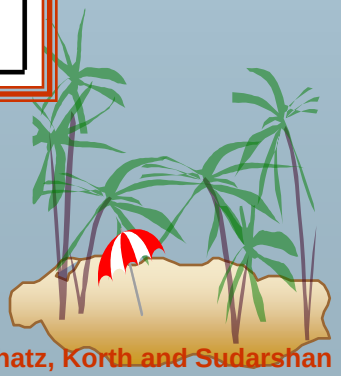
The *customer* Relation

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

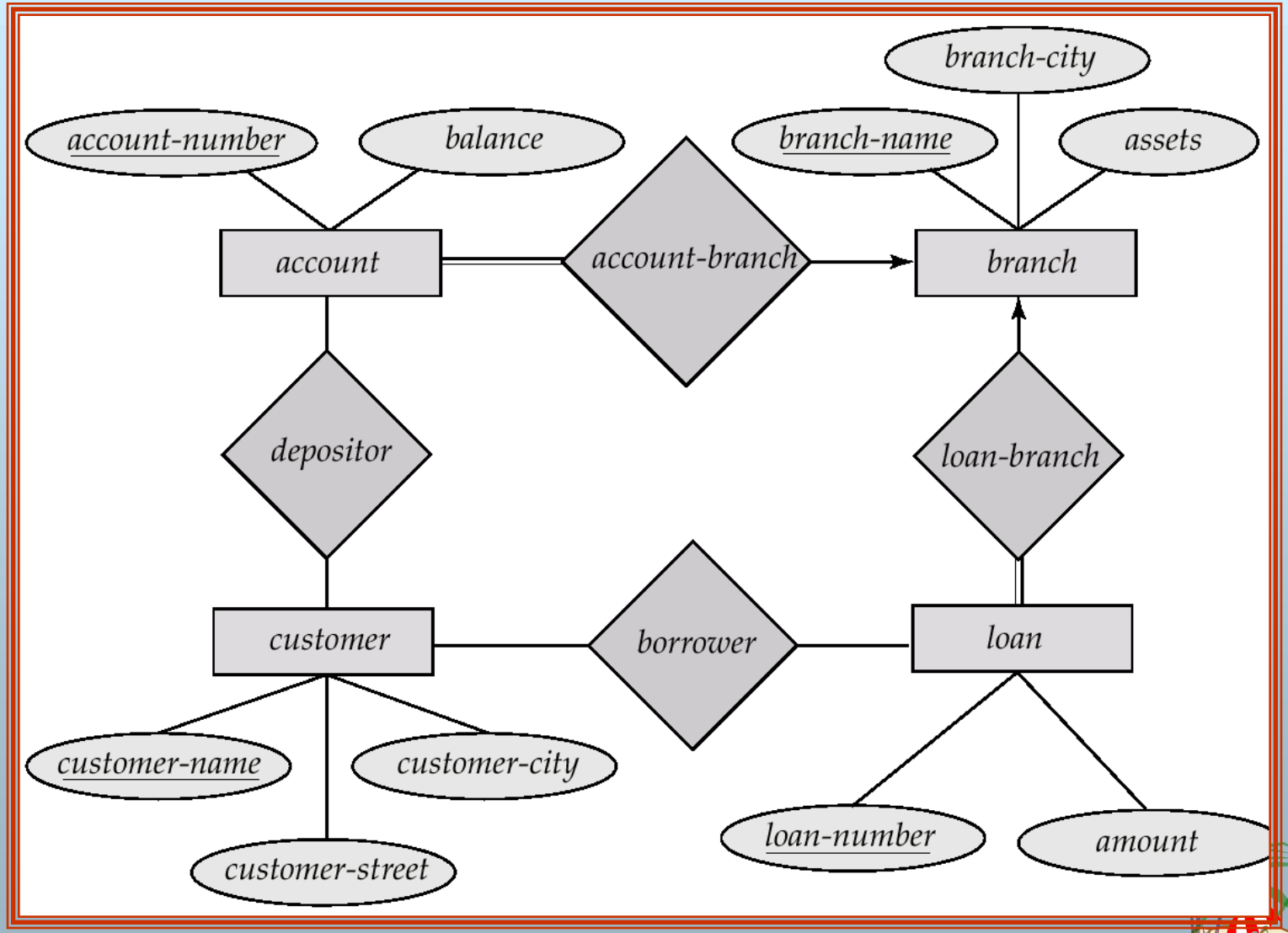


The *depositor* Relation

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

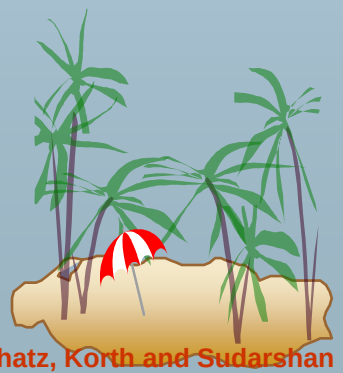


E-R Diagram for the Banking Enterprise



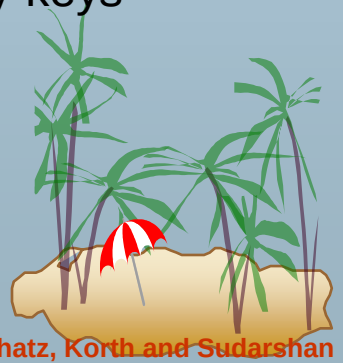
Keys

- 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 (assuming no two customers can possibly have the same name), and no subset of it is a superkey.



Determining Keys from E-R Sets

- **Strong entity set**. The primary key of the entity set becomes the primary key of the relation.
- **Weak entity set**. The primary key of the relation consists of the union of the primary key of the strong entity set and the discriminator of the weak entity set.
- **Relationship set**. The union of the primary keys of the related entity sets becomes a super key of the relation.
  - For binary many-to-one relationship sets, the primary key of the "many" entity set becomes the relation's primary key.
  - For one-to-one relationship sets, the relation's primary key can be that of either entity set.
  - For many-to-many relationship sets, the union of the primary keys becomes the relation's primary key



Schema Diagram for the Banking Enterprise