Network Model and Hierarchy Model

Introduction

- In network model the data is stored in the form of graph In the hierarchy model data is stored in the form of Trees

Database Models

- Network Model
- Hierarchical Model
- Relational Model
- Object/Relational Model
- Object-Oriented Model
- Semi structured Model
- Associative Model
- Entity-Attribute-Value (EAV) data model
- Context Model

Hierarchy Model

- The hierarchical data model organizes data in a tree structure. There is a hierarchy of parent and child data segments. This structure implies that a record can have repeating information, generally in the child data segments. Data in a series of records, which have a set of field values attached to it.

continue

- It collects all the instances of a specific record together as a record type. These record types are the equivalent of tables in the relational model, and with the individual records being the equivalent of rows.

Network Model

- Basic Concepts
- Data-Structure Diagrams
- The DBTG(database task group) CODASYL (multiuser compliance database mgmt system)Model
- DBTG Data-Retrieval Facility
- DBTG Update Facility
- DBTG Set-Processing Facility
- Mapping of Networks to Files

Basic Concepts

- Data are represented by collections of *records*.
  - similar to an entity in the E-R model
  - Records and their fields are represented as *record type*

type *customer* = **record** **type** *account* = **record**
  customer-name: string; *account-number:* integer;
  customer-street: string; *balance:* integer;
  customer-city: string;
end **end**

- Relationships among data are represented by *links*
  - similar to a restricted (binary) form of an E-R relationship
  - restrictions on links depend on whether the relationship is many-many, many-to-one, or one-to-one.

Data-Structure Diagrams

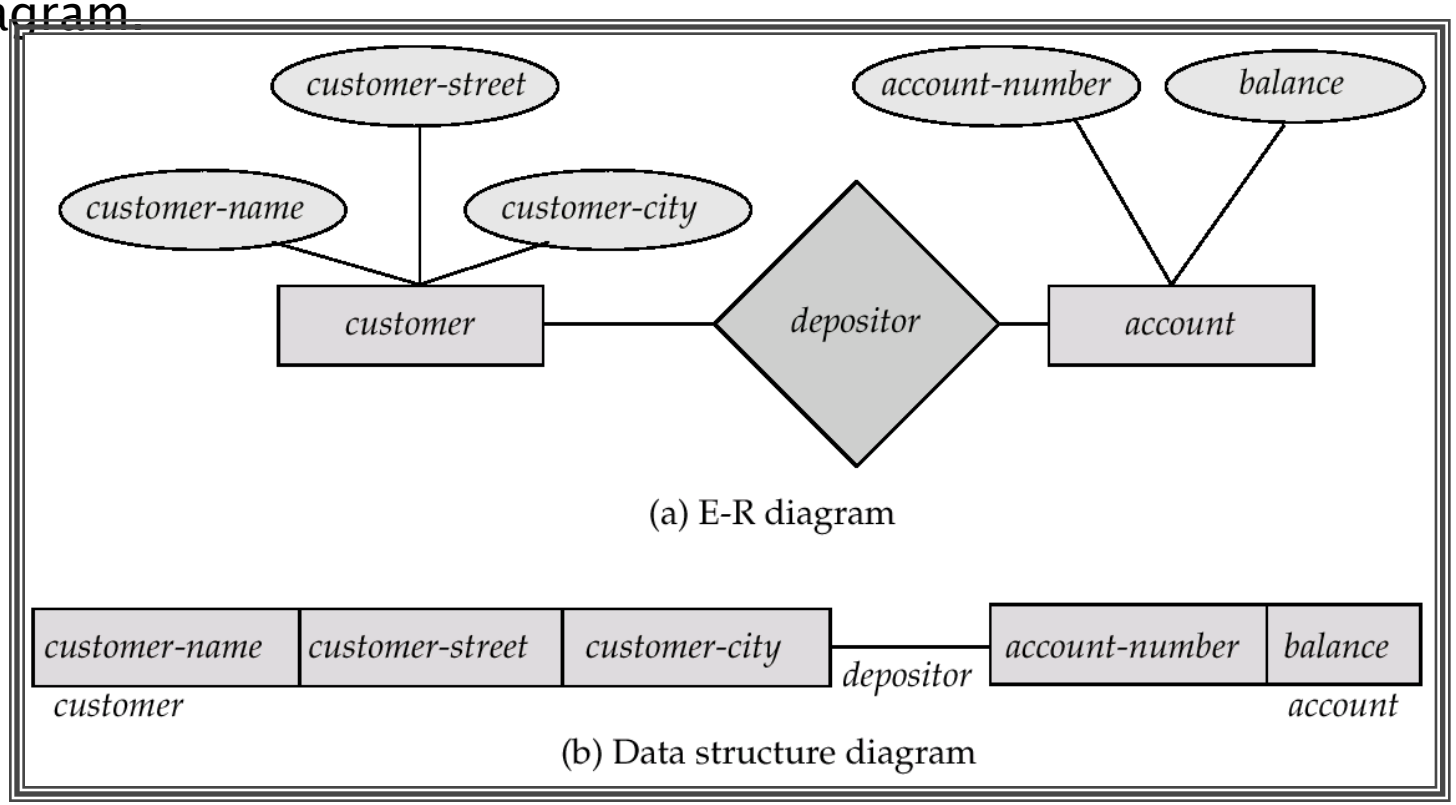
- Schema representing the design of a network database.

- A data-structure diagram consists of two basic components:
  - **Boxes**, which correspond to record types.
  - **Lines**, which correspond to links.

- Specifies the overall logical structure of the database.

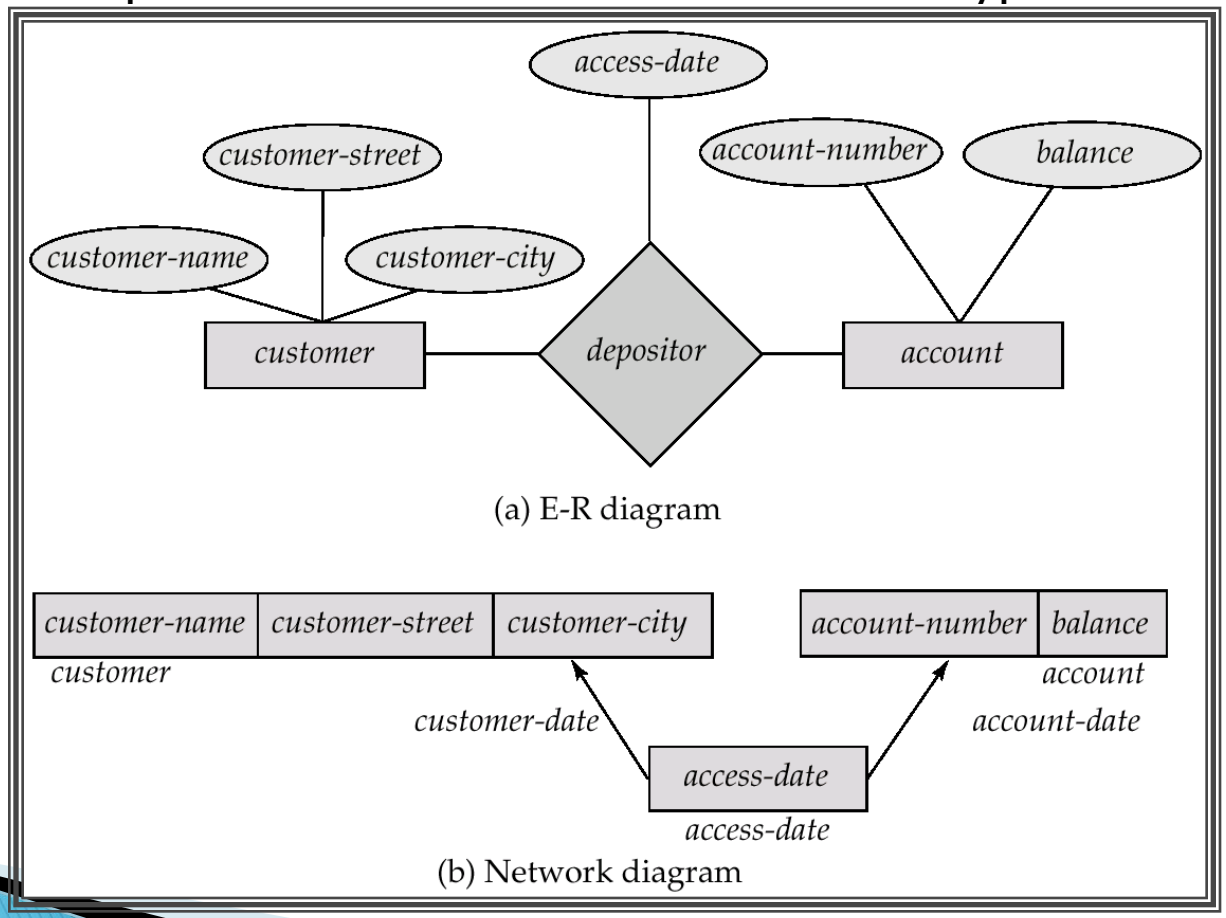
Data-Structure Diagrams (Cont.)

- For every E-R diagram, there is a corresponding data-structure diagram.



Data-Structure Diagrams (Cont.)

- Since a link cannot contain any data value, represent an E-R relationship with attributes with a new record type and links.

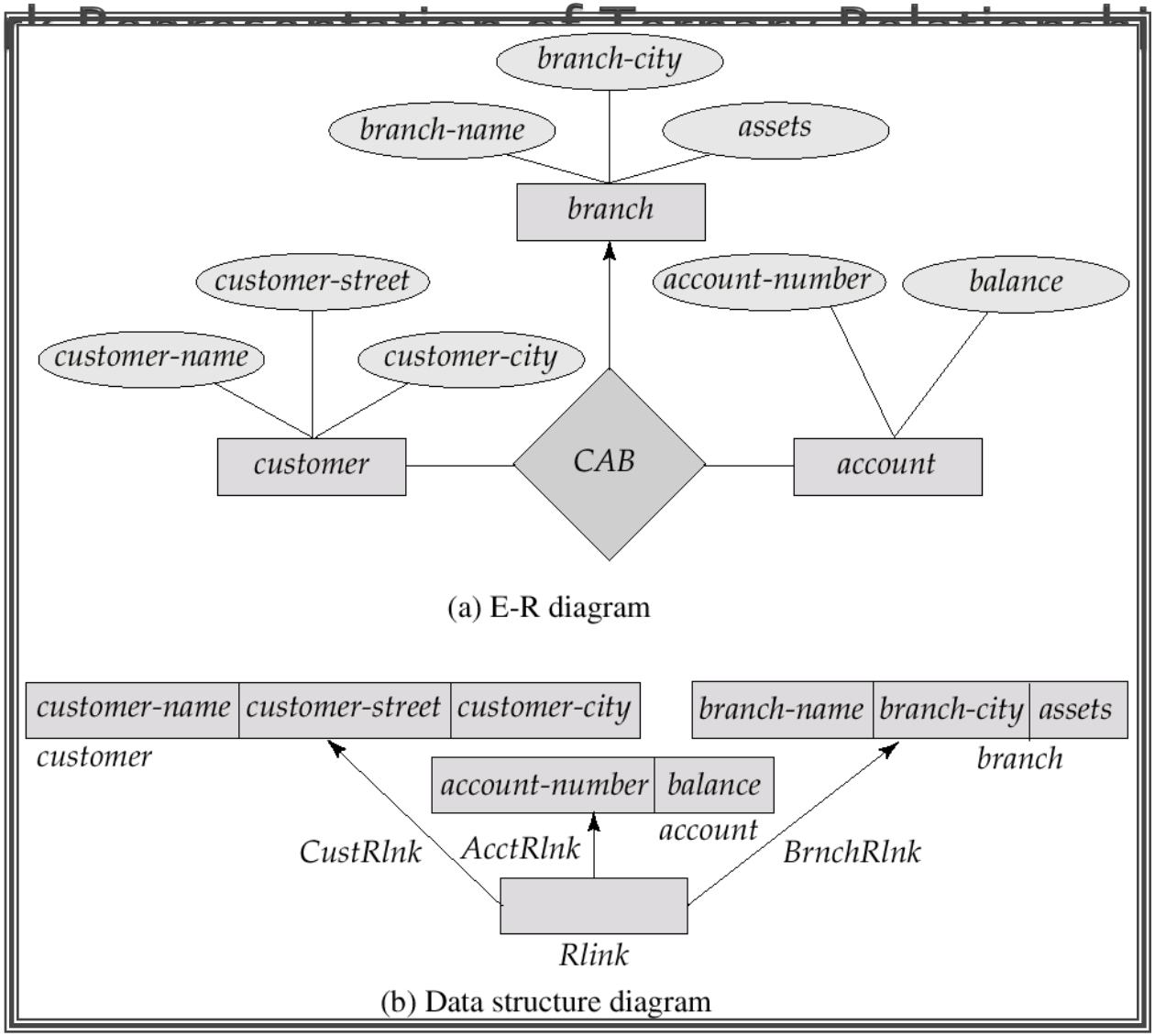


General Relationships

- To represent an E-R relationship of degree 3 or higher, connect the participating record types through a new record type that is linked directly to each of the original record types.

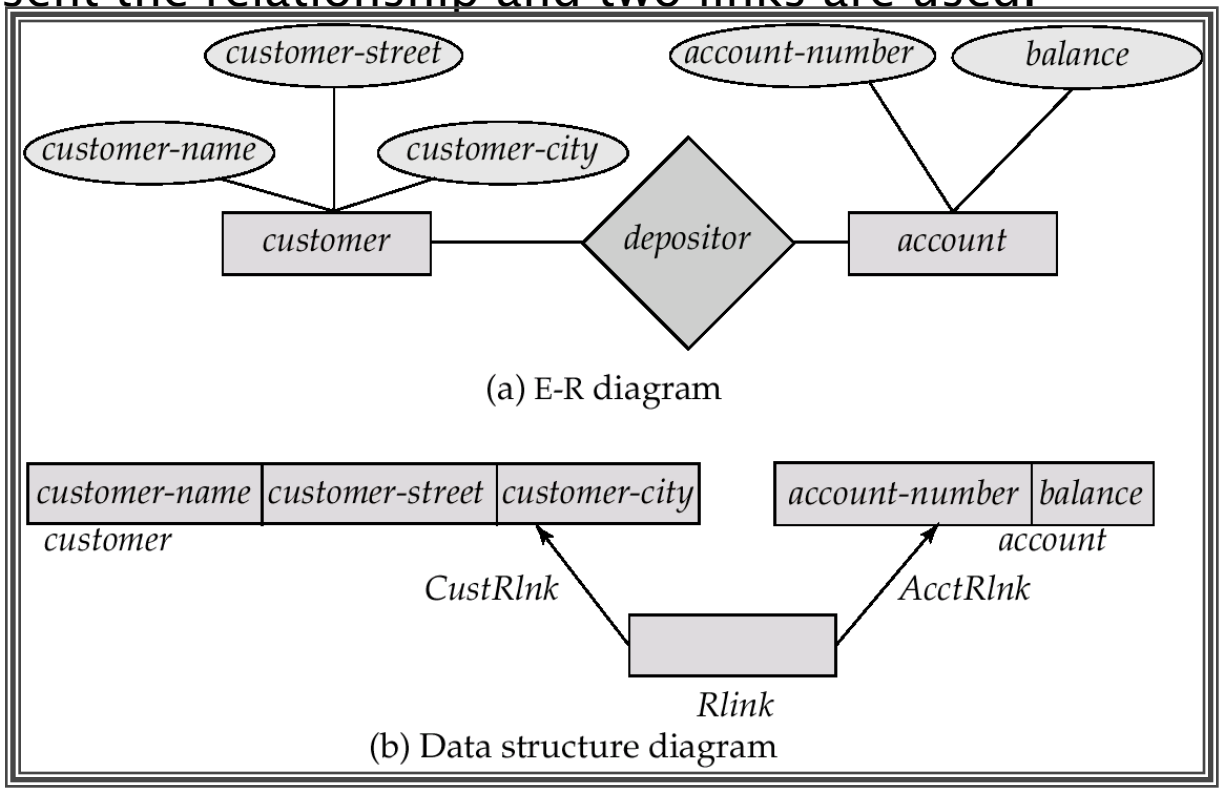
1. Replace entity sets *account, customer,* and *branch* with record types *account, customer,* and *branch,* respectively.
2. Create a new record type *Rlink* (referred to as a *dummy* record type).
3. Create the following many-to-one links:
   - *CustRlink* from *Rlink* record type to *customer* record type
   - *AcctRlnk* from *Rlink* record type to *account* record type
   - *BrncRlnk* from *Rlink* record type to *branch* record type

Netwo[illegible]p


branch-city
branch-name
assets
branch
customer-street
account-number
balance
customer-name
customer-city
customer
CAB
account
(a) E-R diagram
customer-name
customer-street
customer-city
customer
branch-name
branch-city
assets
branch
account-number
balance
account
CustRlnk
AcctRlnk
BrnchRlnk
Rlink
(b) Data structure diagram

The DBTG CODASYL Model

- All links are treated as many-to-one relationships.
- To model many-to-many relationships, a record type is defined to represent the relationship and two links are used.



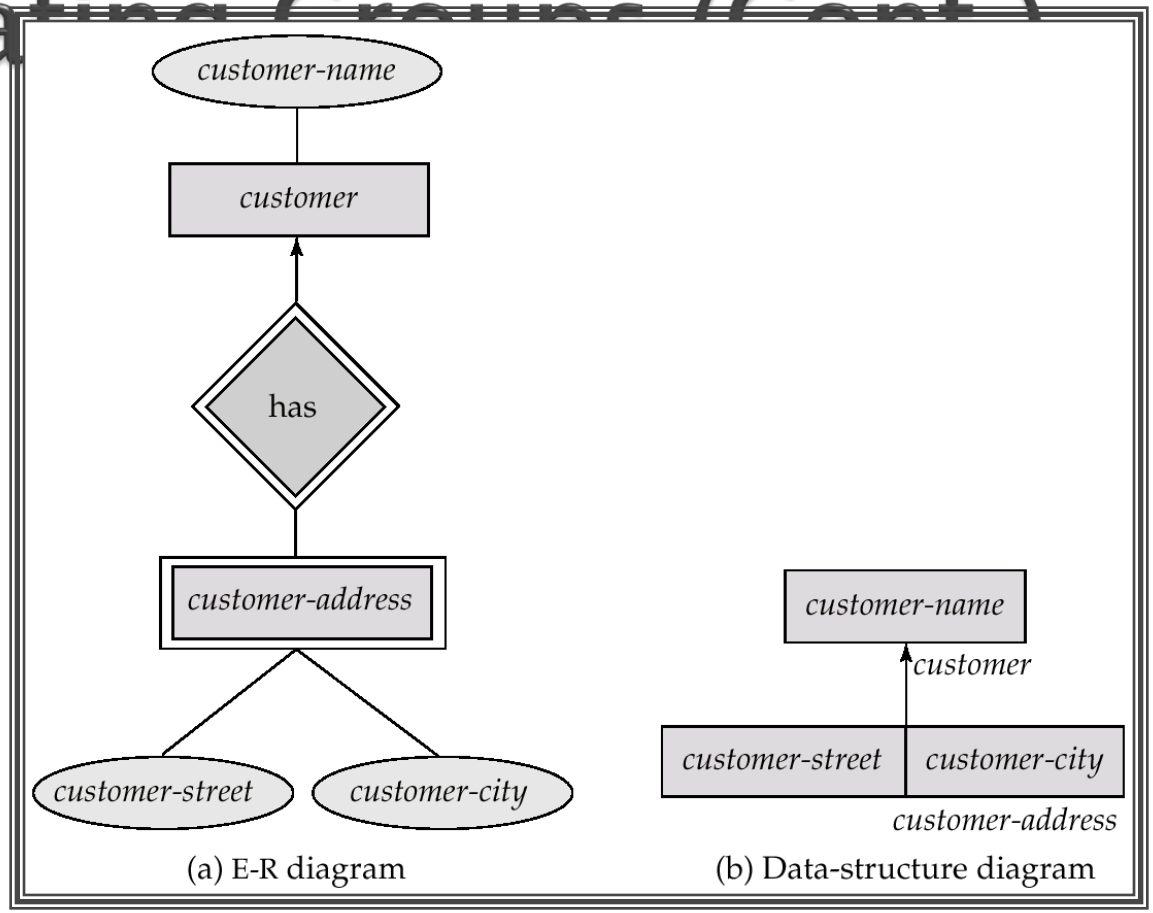
DBTG Sets

- The structure consisting of two record types that are linked together is referred to in the DBTG(database task group) model as a *DBTG set*
- In each DBTG set, one record type is designated as the *owner,* and the other is designated as the *member*, of the set.
- Each DBTG set can have any number of *set occurrences* (actual instances of linked records).
- Since many-to-many links are disallowed, each set occurrence has precisely one owner, and has zero or more member records.
- No member record of a set can participate in more than one occurrence of the set at any point.
- A member record can participate simultaneously in several set occurrences of *different* DBTG sets.

Repeating Groups

- Provide a mechanism for a field to have a set of values rather than a single value.
- Alternative representation of weak entities from the E-R model
- Example: Two sets.
  - *customer (customer-name)*
  - *customer-address (customer-street, customer-city)*
- The following diagrams represent these sets without the repeating-group construct.

Repeating Groups (Cont.)



- With the repeating-group construct, the data-structure diagram consists of the single record type *customer.*

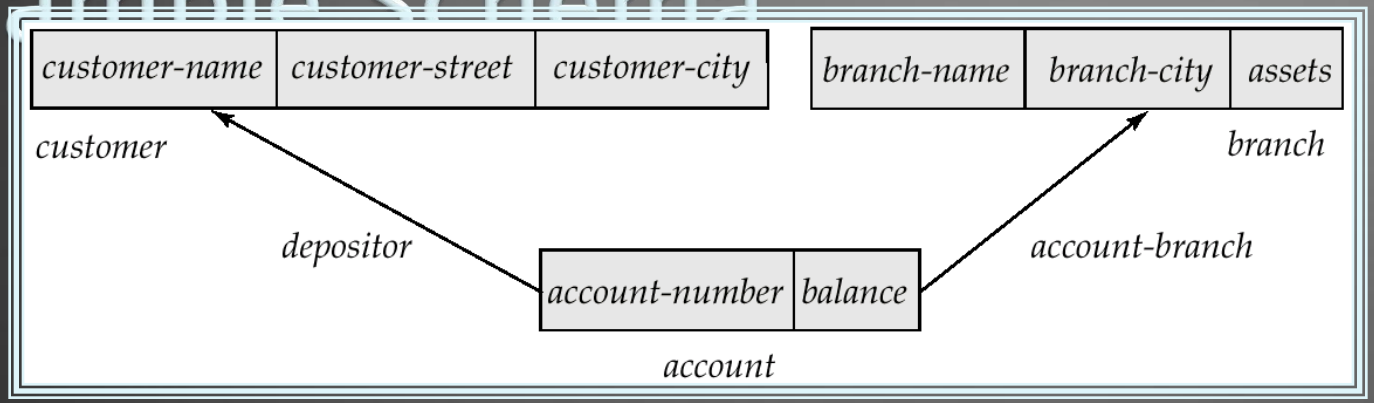
DBTG Data-Retrieval Facility

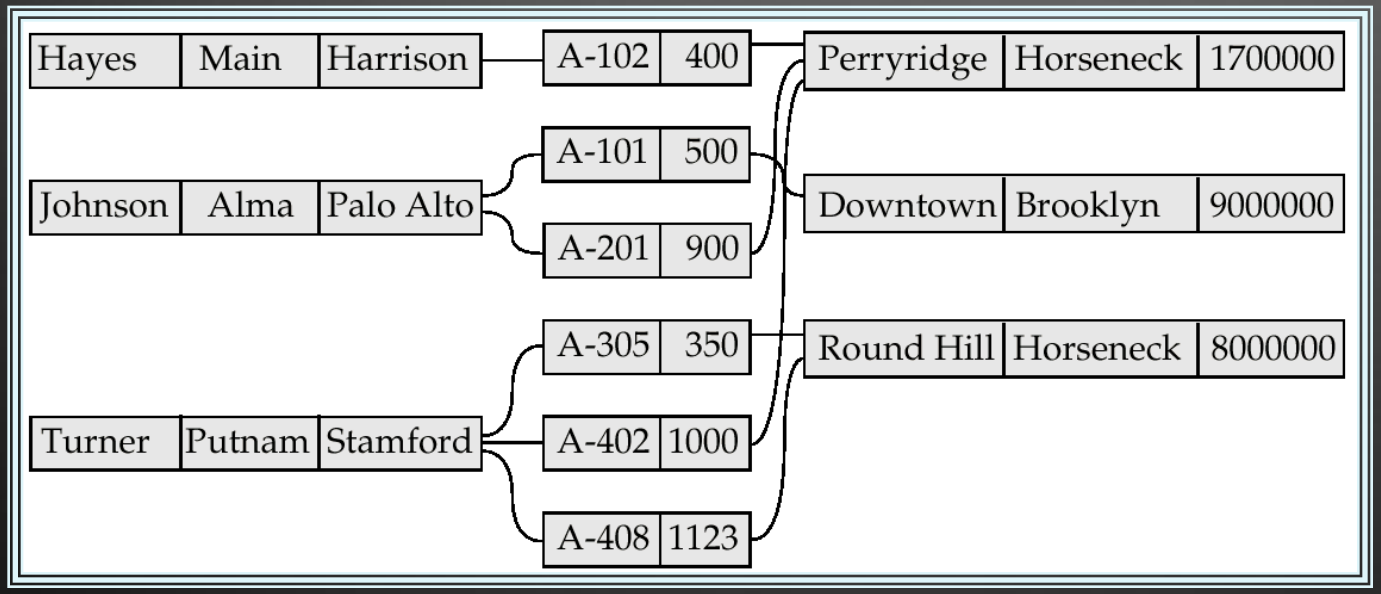
- The DBTG data manipulation language consists of a number of commands that are embedded in a host language.
- *Run unit* — system application program consisting of a sequence of host language and DBTG command statements. Statements access and manipulate database items as well as locally declared variables.
- *Program work-area* (or *user work area*) — a buffer storage area the system maintains for each application program

DBTG Variables

- Record Templates
- Currency pointers
  - Current of record type
  - Current of set type
  - Current of run unit
- Status flags
  - **DB-status** is most frequently used
  - Additional variables: **DB-set-name, DB-record-name,** and **DB-data-name**

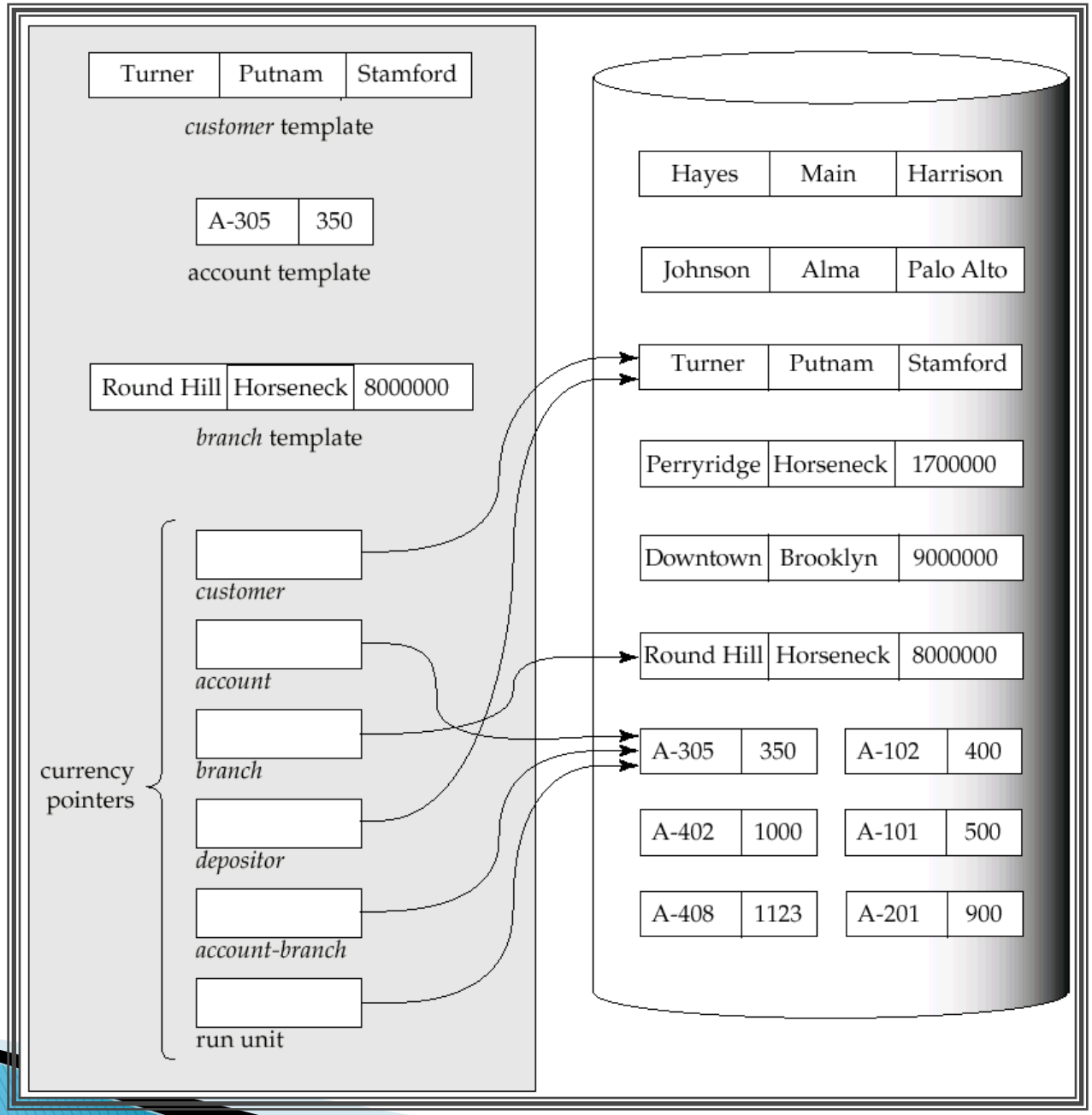
Example Schema





Example Program Work Area

- Templates for three record types: *customer*, *account*, and *branch.*
- Six currency pointers
  - Three pointers for record types: one each tot he most recently accessed *customer*, *account*, and *branch* record
  - Two pointers for set types: one to the most recently accessed record in an occurrence of the set *depositor*, one to the most recently accessed record in an occurrence of the set *account-branch*
  - One run-unit pointer.
- Status flags: four variables defined previously
- Following diagram shows an example program work area state.


Turner
Putnam
Stamford
customer template
A-305
350
account template
Round Hill
Horseneck
8000000
branch template
currency pointers
customer
account
branch
depositor
account-branch
run unit
Hayes
Main
Harrison
Johnson
Alma
Palo Alto
Turner
Putnam
Stamford
Perryridge
Horseneck
1700000
Downtown
Brooklyn
9000000
Round Hill
Horseneck
8000000
A-305
350
A-102
400
A-402
1000
A-101
500
A-408
1123
A-201
900

The Find and Get Commands

- **find** locates a record in the database and sets the appropriate currency pointers
- **get** copies of the record to which the current of run-unit points from the database to the appropriate program work area template
- Example: Executing a **find** command to locate the customer record belonging to Johnson causes the following changes to occur in the state of the program work area.
  - The current of the record type *customer* now points to the record of Johnson.
  - The current of set type *depositor* now points to the set owned by Johnson
  - The current of run unit now points to *customer* record Johnson.

Access of Individual Records

- **find any** <record type> **using** <record-field>
  Locates a record of type <record type> whose <record-field> value is the same as the value of <record-field> in the <record type> template in the program work area.
- Once such a record is found, the following currency pointers are set to point to that record:
  - The current of run-unit pointer
  - The record-type currency pointer for <record type>
  - For each set in which that record belongs, the appropriate set currency pointer
- **find duplicate** <record type> **using** <record-field>
  Locates (according to a system-dependent ordering) the next record that matches the <record-field>

Access of Records Within a Set

- Other **find** commands locate records in the DBTG set that is pointed to by the <set-type> currency pointer.
- **find first** <record type> **within** <set-type>
  Locates the first database record of type <record type> belonging to the current <set-type>.
- To locate the other members of a set,k we use

  find next <record type> **within** <set-type>

  which finds the next element in the set <set-type>.
- **find owner within** <set-type>
  Locates the owner of a particular DBTG set

Predicates

- For queries in which a field value must be matched with a specified range of values, rather than to only one, we need to:
  - **get** the appropriate records into memory
  - examine each one separately for a match
  - determine whether each is the; target of our **find** statement

Example DBTG Query

- Print the total number of accounts in the Perryridge branch with a balance greater than $10,000.

```
count := 0;
branch.branch-name := "Perryridge";
find any branch using  branch-name;
find first account within account-branch;
while DB-status = 0 do
    begin
        get account
        if account.balance > 10000 then count := count  + 1;
        find next account within  account-branch;
    end
print (count);
```

DBTG Update Facility

- DBTG mechanisms are available to update information in the database.
- To create a new record of type <record type>
  - insert the appropriate values in the corresponding <record type> template
  - add this new record to the database by executing

store <record type>

- Can create and add new records only one at a time

DBTG Update Facility (Cont.)

- To modify an existing record of type <record type>
  - find that record in the database
  - get that record into memory
  - change the desired fields in the template of <record type>
  - reflect the changes to the record to which the currency point of <record type> points by executing

modify <record type>

DBTG Update Facility (Cont.)

- To delete an existing record of type <record type>
  - make the currency pointer of that type point to the record in the database to be deleted
  - delete that record by executing

    erase <record type>

- Delete an entire set occurrence by finding the owner of the set and executing

  erase all <record type>

  - Deletes the owner of the set, as well as all the set's members.
  - If a member of the set is an owner of another set, the members of that second set also will be deleted.
  - **erase all** is recursive.

DBTG Set-Processing Facility

- Mechanisms are provided for inserting records into and removing records from a particular set occurrence
- Insert a new record into a set by executing the **connect** statement.

  connect <record type> **to** <set-type>

- Remove a record from a set by executing the **disconnect** statement.

  disconnect <record type> **from** <set-type>