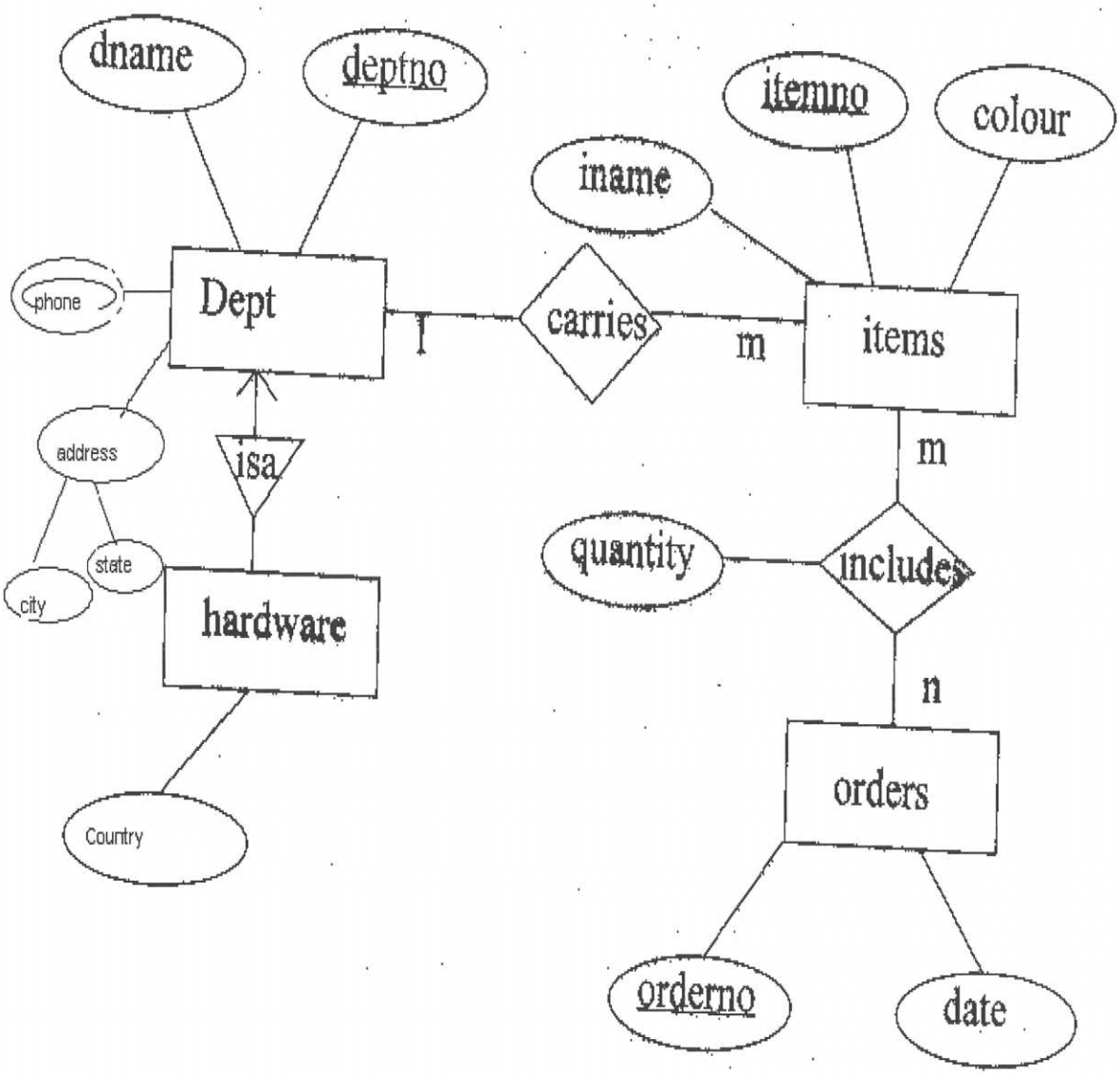
An object-relational database example
CS 5513

E-R Diagram:



Building a Sample Application

- Defining types:
  - *create type address_obj AS object(*
    city varchar2(15),
    state varchar2(2)
    );

  - This statement creates an user defined data type(object) called address_obj

  - Similar to converting a *complex attribute* into a table in RDBMS

  - *Create type phone_typ AS varray(10) OF varchar2(15);*

  - This statement creates a user-defined data type called phone_typ which can store up to 10 phone numbers

  - As opposed to converting a *multi-valued attribute* into a table in RDBMS

- Defining objects for tables:
    - *create type dept_obj AS object (*
      dept_no number(5),
      dname varchar2(20),
      address address_obj
      phone_no phone_typ
) ***NOT FINAL****;*
    - This statement creates an object called dept_obj, with the necessary attributes.
    - Makes use of the user defined data types defined previously
    - The key word *NOT FINAL* suggests that there will be object definitions in future which have the same attributes (*INHERITANCE*).
    - *create type hardware_obj UNDER dept_obj(*
      country varchar2(20)
      );
    - The above statement creates a sub-type of the object dept_obj.
    - hardware_obj therefore inherits all the properties of dept_obj.
    - *alter type dept_obj* ***FINAL;***
    - Can be altered only if dept_obj has no sub-types

- *create type items_obj AS object (*
  item_no number(5),
  iname varchar2(20),
  color varchar2(15),
  dept_no ***REF*** *dept_obj*
  );

- Creates an object called items_obj with a reference attribute

- REF keyword suggests that the dept_no attribute references the primary key attribute of dept_obj

- *create type orders_obj AS object (*
  order_no number(5),
  odate varchar2(15)
  );

- *create type includes_obj AS object (*
  item_no ***REF*** *items_obj,*
  order_no ***REF*** *orders_obj,*
  quantity number(5)
  );

Creating object tables:

- *create table dept_tab OF dept_obj*
  (dept_no primary key)
  object id primary key;

- A table is created based on the object definition of dept_obj.

- The primary key is defined and it is also identified as the object id.

- *create table items_tab OF items_obj*
  (primary key(item_no),
  foreign key(dept_no)
  references dept_tab)
  object id primary key;

- items_tab table is created from the items_obj

- Primary key and foreign key are defined

- The object id for this table is its primary key

- *create table hardware_tab OF hardware_obj;*

- Primary key is the same as that of dept_tab table since hardware_obj is a sub-type of dept_obj

Query1:

→ **Insert a new set of values into dept_tab:**

- *INSERT INTO dept_tab VALUES (10, 'dept1', address_obj('norman', 'OK'));*
- Note how address_obj is used in the insertion process
- This is not possible in Relational DBMS. A separate table is needed for address if it is a complex attribute

Query 2:

→Insert a new set of values into harware_tab

- *INSERT INTO hardware_tab VALUES (10, 'dept1', address_obj('norman', 'OK'), 'USA');*
- Though the attributes were not declared explicitly, the table description would show all the attributes of dept_obj as well.
- It is essential to insert values for all the attributes and not just for "country"

Query 3:
→Insert a new set of items and associate them with the corresponding departments:

- *INSERT INTO items_tab*
  SELECT
  1001,
  'item1',
  'red',
  REF(D)
  FROM dept_tab D
  WHERE D.dept_no = 10;

- Note how the foreign key value is inserted into the table

- Completely different form the traditional way of inserting values

Query 3:
→Display all the departments:

- *Select D.dept_no,*
  D.dname,
  D.address.city, // path query
  D.address.state
  From
  dept_tab D;

- Note how the address attribute is accessed to retrieve the name of the city and state from it