Database Systems 2

Lecture 20

Storing Objects in Databases

Object-Oriented Concepts

Objects and attributes.

Classes, subclasses, superclasses, and inheritance.

Persistent Objects

Mapping to a RDBMS – Three Approaches.

OODBMS

Oracle User Defined Types (UDT).

Object

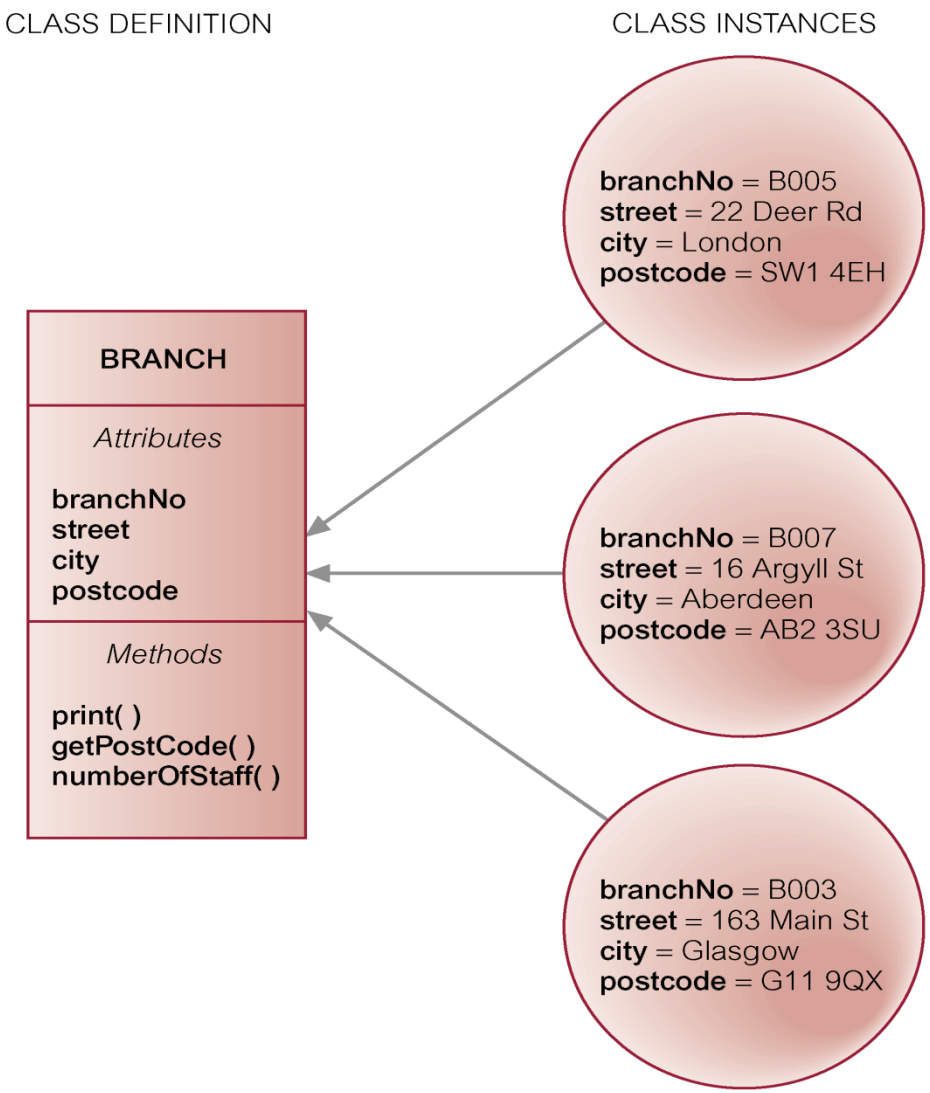
Uniquely identifiable collection of data that contains both the attributes that describe the state of a real-world object and the actions associated with it.

Definition very similar to that of a database entity, however:

- an object encapsulates both attributes and methods.
- a database entity only models attributes.

This means that when trying to store objects in a relational database, there is an **impedance mismatch**.

Objects Contain Attributes and Share Methods



Subclasses, Superclasses, and Inheritance

Inheritance allows one class of objects to be defined as a special case of a more general class.

Special cases are *subclasses* and more general cases are *superclasses*.

Process of forming a superclass is *generalization*; forming a subclass is *specialization*.

Subclass inherits all properties of its superclass and can define its own unique properties.

Subclass can override inherited methods.

What is inheritance?

Inheritance is a mechanism which allows a new class to be defined by reference to an existing ‘parent’ class

The new class can:

Add extra attributes to the ones inherited from the parent

Add extra methods to the ones inherited from the parent

Modify inherited behaviour

Example of Inheritance

Class:	Kettle
Parent:	None
Attributes:	Capacity Amount Held
Methods:	Fill ( ) Pour ( ) Heatwater ( )

Class:	Electric Kettle
Parent:	Kettle
Attributes:	Temperature
Methods:	Heatwater ( ) Turnoff ( )

Class Definitions

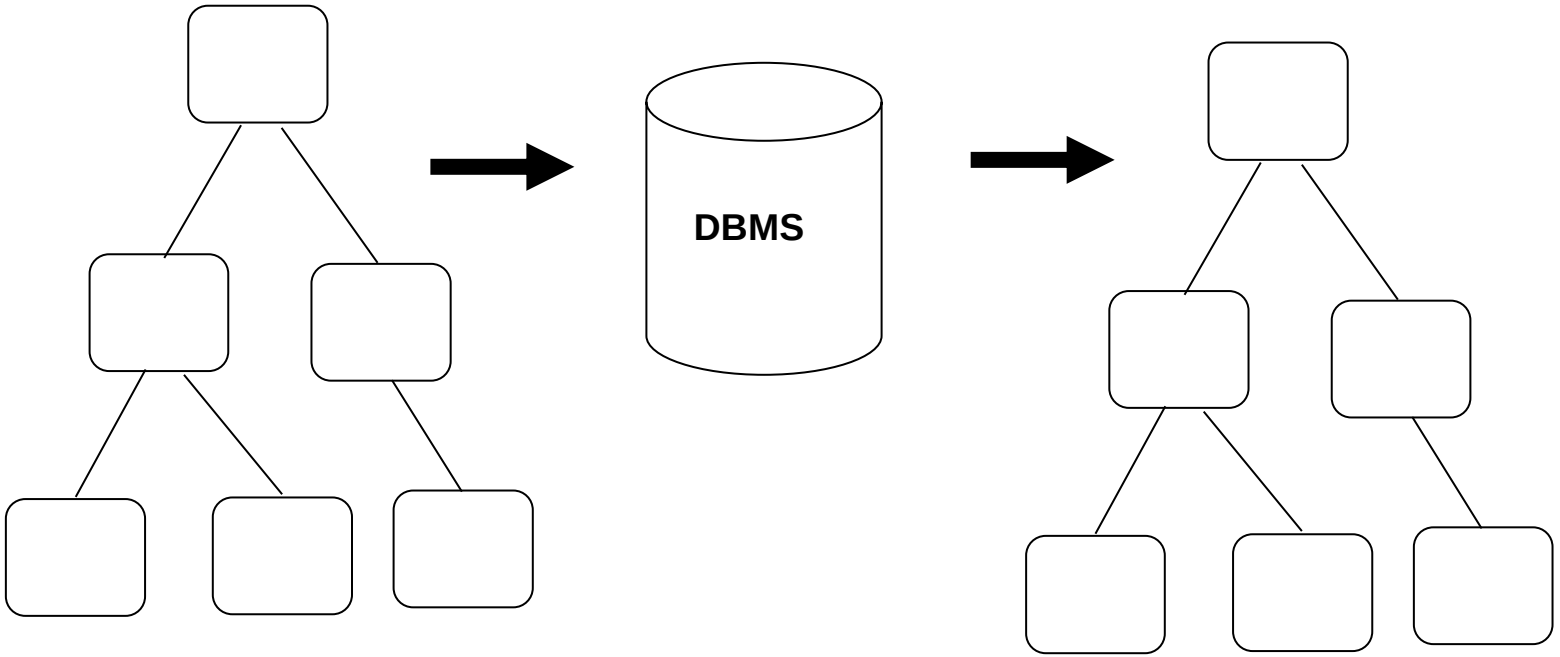
KevsKettle : Kettle
1 Litre 0.3 Litre
Fill( ) Pour( ) Heatwater( )

HelensKettle : Electric Kettle
1.2 Litre 0.5 Litre **40 Degrees C**
Fill( ) Pour( ) **Heatwater( )** **TurnOff( )**

Objects based on classes

Persistent Objects

Nowadays, programs don't just produce data that needs to be stored. They tend to produce hierarchies of objects, which have data inside. If you want to use objects again you must store them!



Storage Mechanisms

Relational Database

table = all objects based on a class,

columns = class attributes,

row = object

Extended relational databases

binary large objects (BLOBS) = could be a collection of objects

Object-oriented databases

class = class, object = object

Mapping to an RDBMS

Need to transform object structure into table-oriented structure

OO model:

Very rich set of types. Each class is a user defined type.

Will contain complex arrangements of attributes and pointers to methods.

Relational model:

Primitive set of types: number, text, date/time.

This disparity is called an **Impedance mismatch**.

Possible Mapping Strategies

Approach 1

Map each class to a table and map each subclass to a table containing just those columns which are additional to those of the superclass.

OR

Approach 2

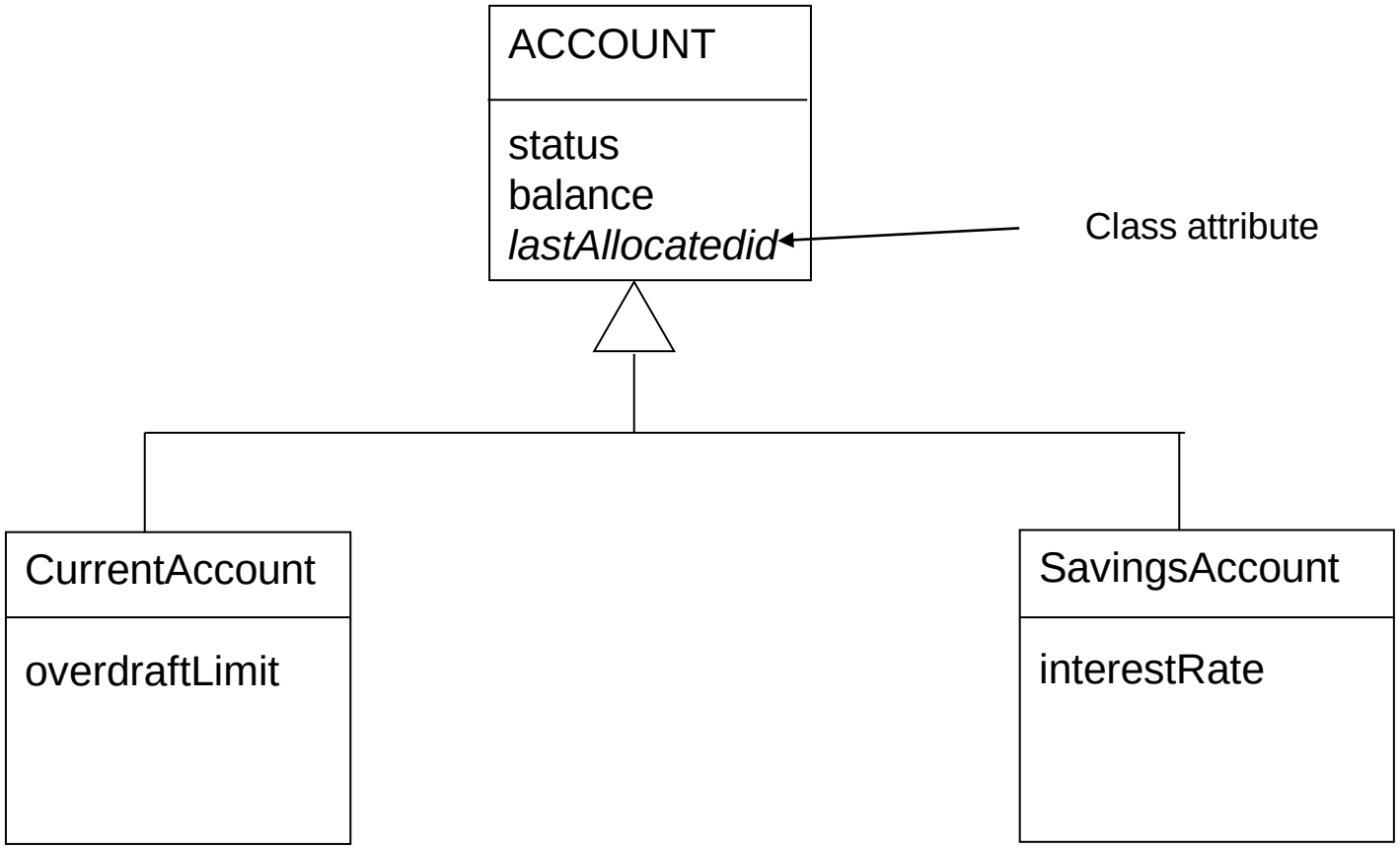
Have each class store its own attributes plus those of each of its superclasses, all the way up to the top of the hierarchy.

OR

Approach 3

Map a complete class hierarchy into a single table.

RDBMS Mapping Example



What attributes does CurrentAccount contain? SavingsAccount?

Relational Tables – Approach 1

Store the data that is common to all classes in a super-class table, and represent each sub-class by a sub-class table

ACCOUNT
lastAllocatedid status balance

id	status	balance	lastAllocatedid
2890	active	234.44	2893
2891	active	55.33	2893
2892	over	-3000.99	2893
2893	active	222.11	2893

CurrentAccount
overdraftLimit

id	overdraftLimit
2890	1000
2891	500

Savings Account
interestRate

id	interestRate
2892	12
2893	14

Relational Tables – Approach 2

Store all data in the sub-class tables. The superclass table only contains Class variables.

ACCOUNT
lastAllocatedid

Key	lastAllocatedid
1	2893

CurrentAccount
overdraftLimit status Balance

id	overdraftLimit	status	balance
2890	1000	active	234.44
2891	500	active	55.33

CurrentAccount
overdraftLimit status Balance

id	interestRate	status	balance
2892	12	over	-3000.99
2893	14	active	222.11

Relational Tables – Approach 3

Store all data in a super-class table.

ACCOUNT
lastAllocatedid overdraftLimit interestRate Status balance

id	overdraftLimit	interestRate	status	balance	lastAllocateid
2890	1000	**NULL**	active	234.44	2893
2891	500	**NULL**	active	55.33	2893
2892	**NULL**	12	over	-3000.99	2893
2893	**NULL**	14	active	222.11	2893

ODBMS

Extended Relational DBMS / Object Relational DBMS
OO features built onto the top of a standard RDBMS

ODBMS
Redesigned from the bottom up. First generation not very robust and quite slow.

Second generation are a lot better and may compete with RDMS

Quite similar to hierarchical and network databases, in that they use pointers.

Can handle nested structures.

In theory, they allow methods (functions) to be stored in the database as well as attributes. They are active, whereas relational databases are passive.

User Defined Types

User Defined Types (also called Oracle Objects). You can also create a collection (array) of objects.

```
SQL> CREATE OR REPLACE TYPE employee AS OBJECT
     (
     emp_name VARCHAR2(50),
     soc_sec VARCHAR2(9),
     address VARCHAR2(100)
     );

Type created.

SQL> CREATE TABLE emps_with_type
     (
     employee_data    employee,
     date_added       DATE,
     note             VARCHAR2(20)
     );

Table created.

SQL> desc employee

Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMP_NAME                                           VARCHAR2(50)
 SOC_SEC                                            VARCHAR2(9)
 ADDRESS                                            VARCHAR2(100)
```

```
SQL> desc emps_with_type

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPLOYEE_DATA                                      EMPLOYEE
 DATE_ADDED                                          DATE
 NOTE                                                VARCHAR2(20)


SQL> INSERT INTO emps_with_type (EMPLOYEE_DATA, date_added, note)
     VALUES (employee('Lewis', '123456789', '666 mockingbird lane'),'20-MAR-14', 'good');

1 row created.


SQL> commit;

Commit complete.


SQL> SELECT * FROM emps_with_type;

EMPLOYEE_DATA(EMP_NAME, SOC_SEC, ADDRESS)                 DATE_ADDED          NOTE
-------------------------------------------------------------------------------
EMPLOYEE('Lewis', '123456789', '666 mockingbird lane')    20-MAR-14           good
```

See:
http://docs.oracle.com/cd/B19306_01/appdev.102/b14260/adobjint.htm