Module 11

Objects and Relations

Objectives

- Learn about issues relating to objects and relations

95.305

Topics

- **Programs and Databases: Persistence**
- **Proxies and Adapters**
- **Storing Objects in Relational Tables**
- **Performance Issues**

References

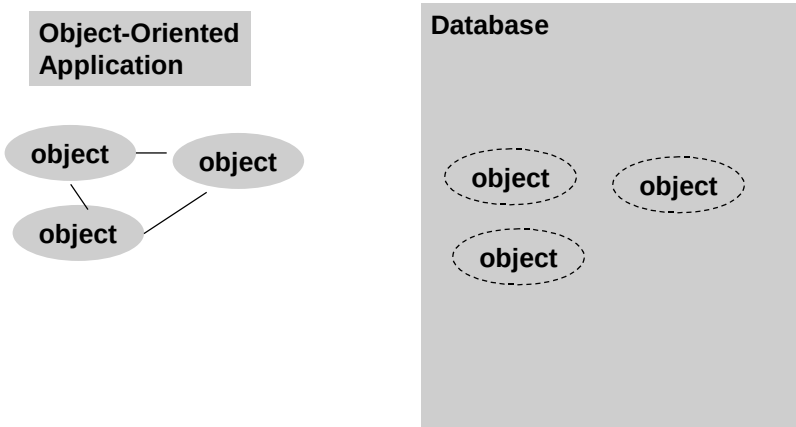
- **White paper on TOPLINK from ObjectPeople**
  `http://www.objectpeople.com`
- **Ivar Jacobson, Object-Oriented Software Engineering: A use case driven approach, Addison Wesley, 1992, Chapter 10**

95.305

Topics

- Programs and Databases: Persistence
- Proxies and Adapters
- Storing Objects in Relational Tables
- Performance Issues

Making Objects Persistent



- Most large applications have persistent data
- An object-oriented application wants to maintain persistent objects
- Using a database to store persistent objects has advantages
  - -database provide integrity
  - -support multi-users, concurrency
  - -may be existing legacy system

- Storing persistent objects in Object-Oriented Database has many advantages
  - -database stores objects (not tables)
  - -translation between persistent entities and local entities is easy (both are objects)
  - -good match between programming language features and data model features (inheritance, data and behaviour, ...)
- But, OODBMS's are still in their infancy and not well established like relational databases
- OODBMS's can be slow

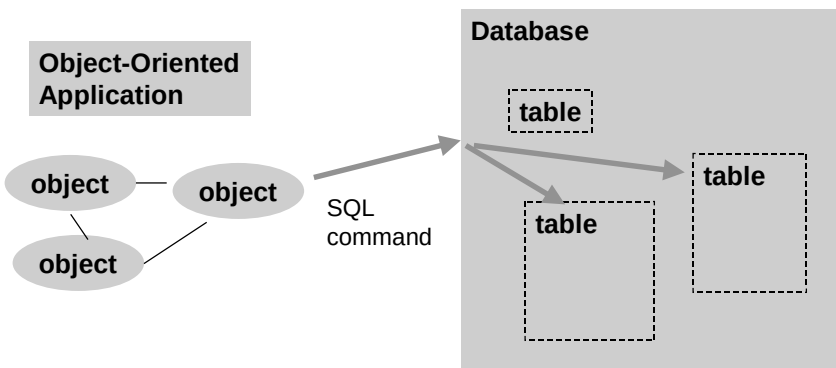
Storing Objects in a Relational Database

- Storing Objects in a relational database causes some problems

  -stores tables not objects
  -"impedance mismatch" between application data model and that of database

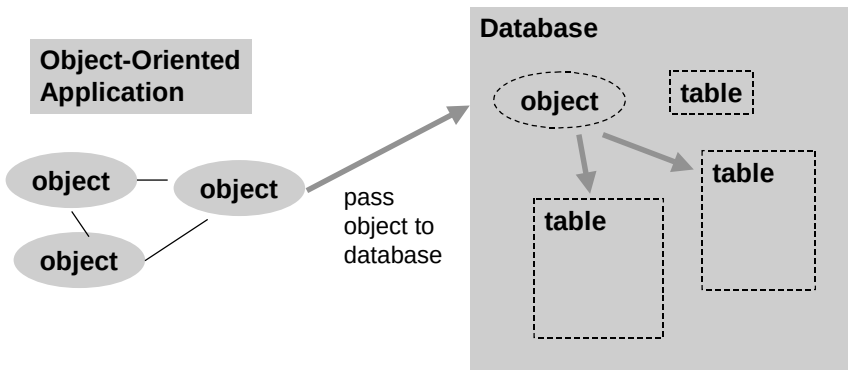
- Still many people are trying to do just that

Approach 1



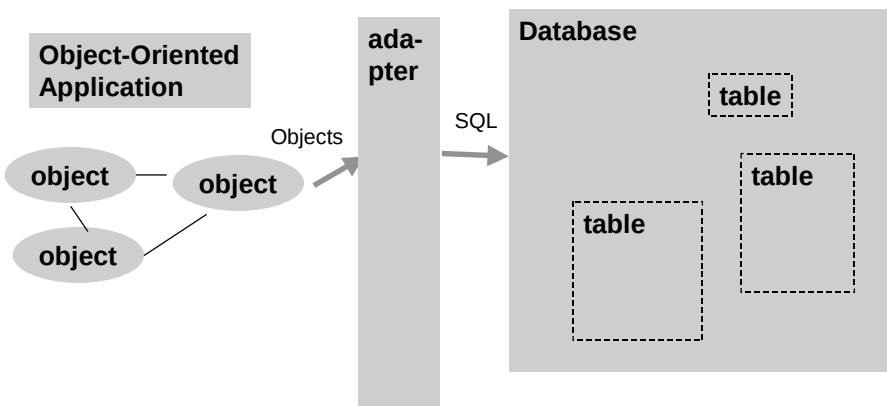
- Objects must know about relational structure and database querying with SQL
- Application objects get polluted with database issues

Approach 2



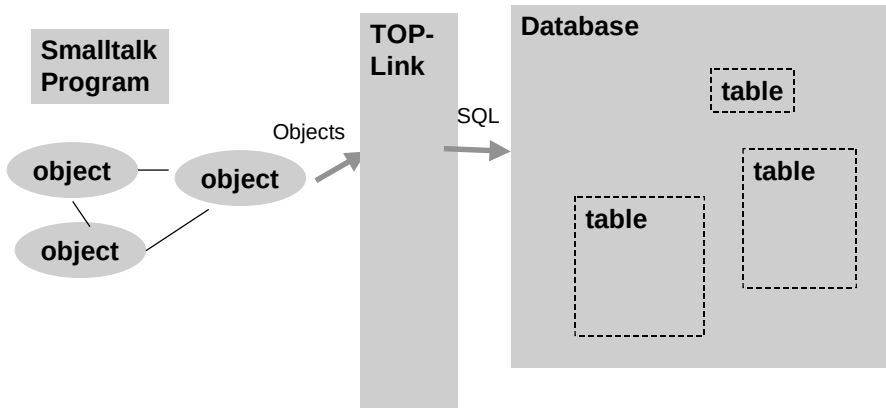
- **Pass objects to database to smash into tables**
- **But, RDBM's don't understand objects; not an industry standard interface like SQL**

Approach 3



- **Use dedicated adaptor between objects and tables**
- **At least application does not have to know about tables, and RDBMS does not know about objects**

TOPLink



- TOPLink, from ObjectPeople, is such an adapter for Smalltalk programs

95.305

Topics

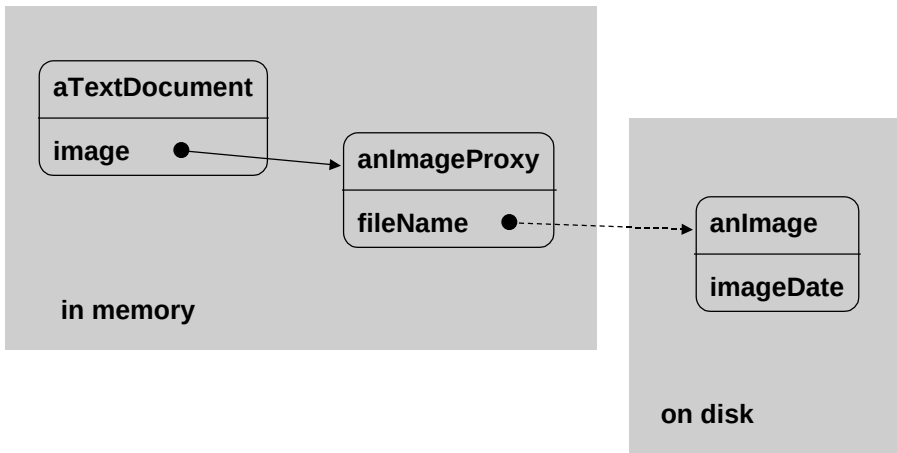
- Programs and Databases: Persistence
- Proxies and Adapters
- Storing Objects in Relational Tables
- Performance Issues

Proxy Pattern

- **Intent:**
  Provide a surrogate or placeholder for another object to control access to it, to introduce side effects for messages sent to the subject.
- **Aka:**
  Surrogate, Encapsulator
- **Other Structural Patterns**
  Adapter, Bridge, Composite, Decorator, Facade, Flyweight

Proxy Pattern -Motivation

- **Example: Deferred creation or initialization**
- **Text document with graphical objects embedded may not want to create the graphical object all when the document is opened, perhaps only those which are in view.**
- **Suggests using a less expensive proxy in place of the real image so the document editor does not know the difference.**
- **The proxy behaves just like the real image, but only creates it when it needs to be drawn (e.g. when draw method is invoked).**



Proxy Pattern: Participants and Collaboration

- **Proxy**
  -maintains reference to real subject
  -provides an interface identical to subjects (client does not know its dealing with a proxy)
  -control access to real subject (and may have to create or destroy it)
- **Proxy forwards request to real subject**
- **Remote proxy maps & encodes request into new address space**
- **Virtual proxy may cache info about real subject to postpone access**
- **Protection proxy ensures client has right to access real subject**

Proxy Pattern: Copy-on-Write example

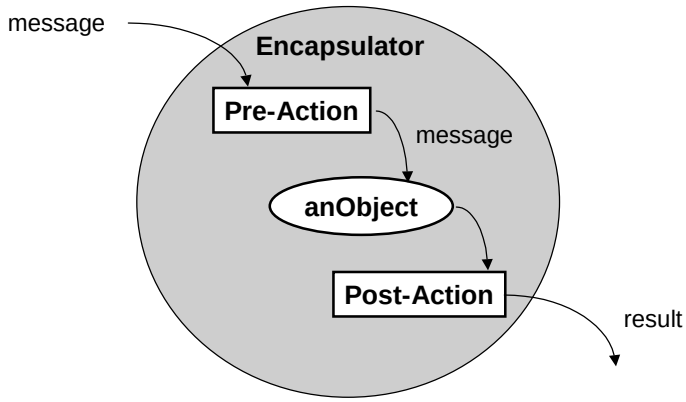
- Copying large objects can be expensive
- Idea: only copy an object if the copy gets modified
- Client thinks they have a copy, but they actually have a proxy to the original
- Only copy the original if client actually tries to modify the proxy (perform a write operation)

Proxy Pattern: Related Patterns

- Adapter
  Provides a different interface to an object.
- Adapts (changes) the interface to an object, proxies offer the same interface
- (Protection proxies though may refuse a certain interface request)
- Decorator
  Similar to proxies but add one or more responsibilities to that of subject
- So, decorators have a different purpose

Related Previous Work: Encapsulators

- **Pascoe, Geoffrey A. “Encapsulators: A New Software Paradigm in Smalltalk-80, OOPSLA ‘86**



Adapter Transparency

- **Adapter should be transparent:**
- **Application objects should not be forced to be a subclass of** `Adapter`
- **Could be implemented with transparent Proxy techniques**

95.305

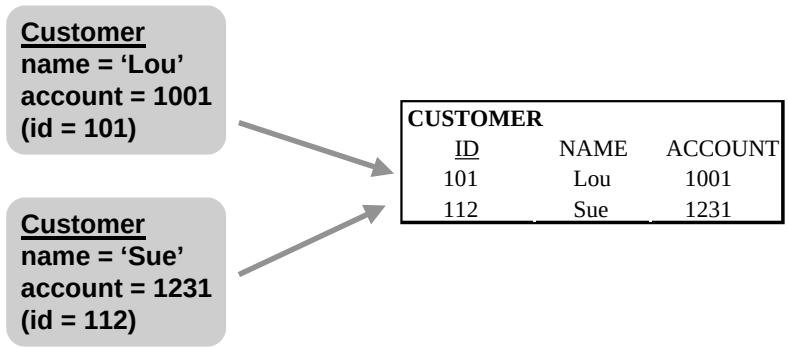
Topics

- Programs and Databases: Persistence
- Proxies and Adapters
- Storing Objects in Relational Tables
- Performance Issues

Issues an Object-RDBMS Adapter must address

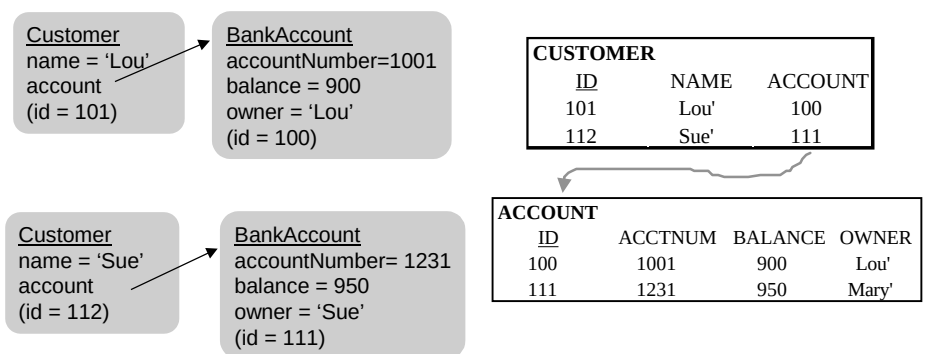
- Object Identity
- In an OOPL objects have identity irrespective of their attribute values -no two objects are identical
- Relational models handles uniqueness through primary keys
- Adapter must match the key in the table with an objects internal identity when in local memory

Storing Objects as Data in Tables



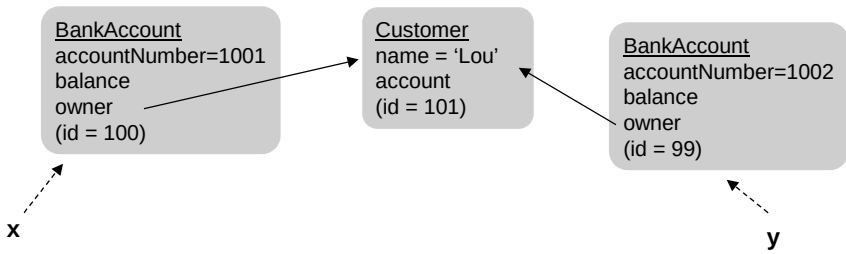
- **Simple idea: Create table for class and use internal (object) id for key**
- **Note: internal object id is not visible to the OO application programmer**

Nested Objects



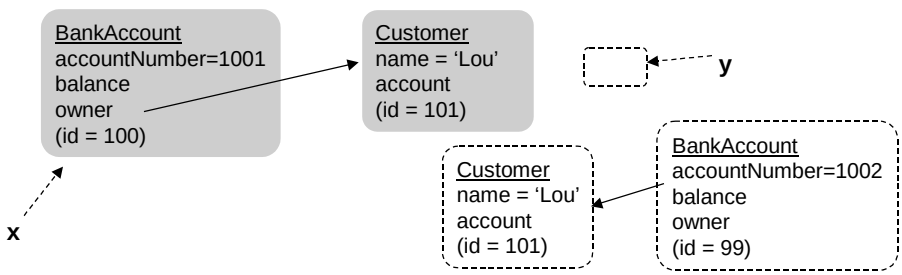
- **Nested objects stored in separate tables with foreign keys**

OO Application Code Example




```
| cust acct |
cust := Customer new name: 'Lou'.
x := BankAccount new owner: cust.
y := BankAccount new owner: cust.
x customer == y customer
```

```
true
```

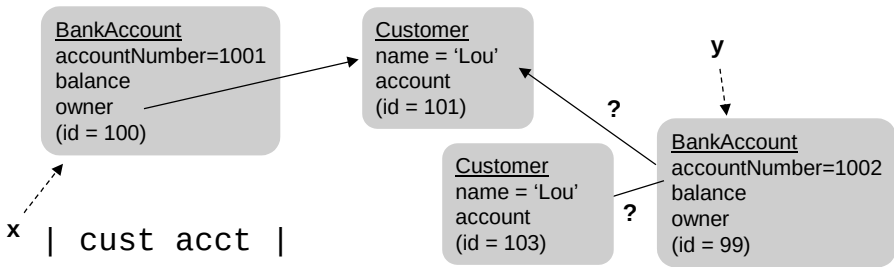

```
| cust acct |
cust := Customer new name: 'Lou'.
x := BankAccount new owner: cust.
y := BankAccount new owner: cust.
y storeOnDatabase
```

```
"Object instance method"
storeOnDatabase
"make receiver persistent and remove
  receiver"
| key |
key := Database insert: self asTuple.
self become:
      (Proxy new class: self class;
                  key: key).
```



```
"Proxy instance method"
doesNotUnderstand
"replace receiver proxy with object created
  from database"
|object tuple|
object := self subject class new.
tuple := Database select: 'all'
                  from: subject class name
                  where: id = subject key.
1 to tuple size do:
 [:i|object instVar at: i put: tuple at: i]
self become: object
```

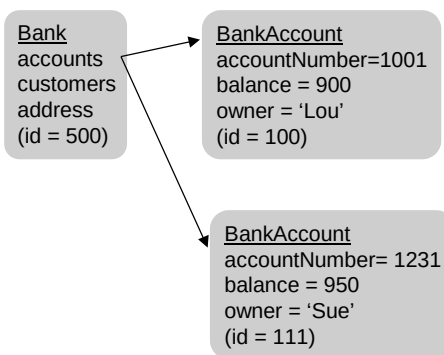
Are object references recovered properly?




```
| cust acct |
cust := Customer new name: 'Lou'.
x := BankAccount new owner: cust.
y := BankAccount new owner: cust.
y storeOnDatabase.
y balance.  "causes y to be recreated"
y customer == x customer
```

????

Collections (1:N Relationships)



BANK

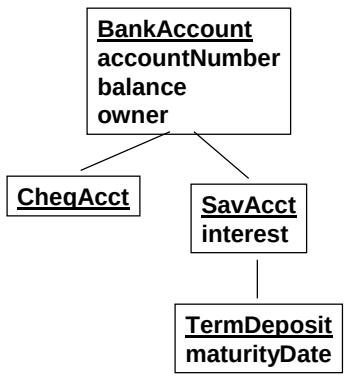
ID	ADDRESS
500	Downtown'
525	Glebe'

DEPOSIT

BANK_ID	ACCOUNT_ID
500	101
500	112

- **1:N Relationships (Collection Attributes) might be stored in separate tables**

What about Inheritance



CHEQACCT

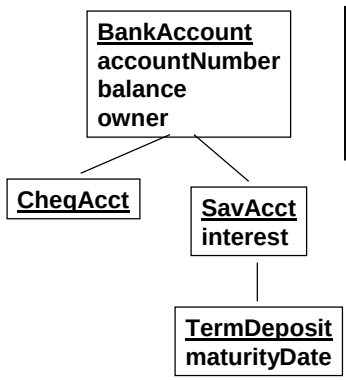
ID	ACCTNUM	BALANCE	OWNER
100	1001	900	Lou'
111	1231	950	Mary'

TERMDEPOSIT

ID	ACCTNUM	BALANCE	OWNER	MATURITY
201	1013	700	Lou'	Feb-96
212	1222	850	Alice'	Jun-97

- **Option1: Represent concrete classes with inherited attributes, but don't represent abstract classes**

...What about Inheritance



BANKACCOUNT

ID	ACCTNUM	BALANCE	OWNER	SUBLCASS
100	1001	900	Lou'	CHEQACCT
111	1231	950	Mary'	CHEQACCT
201	1013	700	Lou'	TERMDEPOSIT
212	1222	850	Alice'	TERMDEPOSIT

CHEQACCT

ID
101
112

TERMDEPOSIT

ID	MATURITY
201	Feb-96
212	Jun-97

- **Option2: Store common attributes in Abstract class table and inherited attributes in subclass tables**

Normalization

- Jacobson suggests:
- A Database designed based on objects will normally end up in 3NF
- “Because table represent objects with unique id and mutually exclusive facts about that object”
- “Because OO models often model reality we tend to identify unique objects and assign attributes where they belong”
- may be a bit naive

95.305

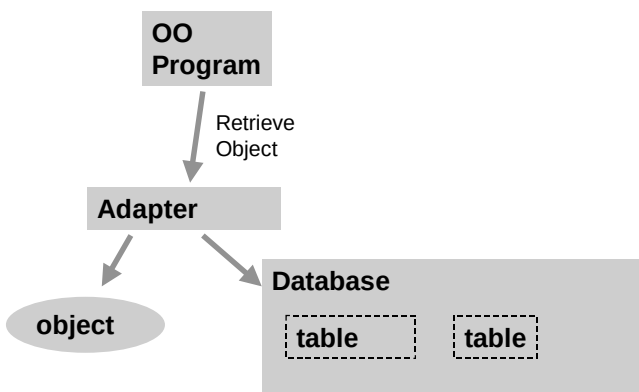
Topics

- Programs and Databases: Persistence
- Proxies and Adapters
- Storing Objects in Relational Tables
- Performance Issues

Adapter Performance Issues

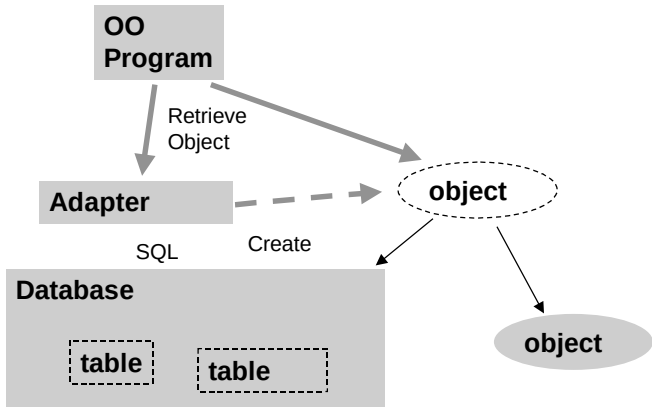
- A Query to the RDBMS might result in many objects.
- The adapter should probably retrieve only a few and provide proxies for others until they are actually needed
- Object proxies can be constructed by transparent wrapper objects
- Adapter should also Cache objects so subsequent references don't go to the database (The application-database interactions are the expensive thing)

Caching



- If the adapter is asked to retrieve an object it first checks to see if it is already in the cache

Caching



- Adapter creates a proxy which the OO program references
- Proxies may have cached objects or else talk to database

Large Results

- Query to database can result in a large number of objects
- Building objects for all rows in a resulting table can be too expensive
- Adapter can use a Stream with cursor position to scroll through “objects” and only create them when needed (amortise over the their use)
- Reduces the perceived delay of fetching objects from the database

Ad hoc Queries from within Objects

- **Occasionally it might be desirable for Objects to know about the database and make direct SQL queries**
- **TOPLink, for example, provides a SQL SELECT-like access using familiar Smalltalk blocks**
- **e.g.**

```
aTopLinkSession
   readAllFor: Employee
   where: [:anEmployee  |
       anEmployee manager address street = 'Elm'
      and: (anEmployee lastName = 'Smith') ]
```

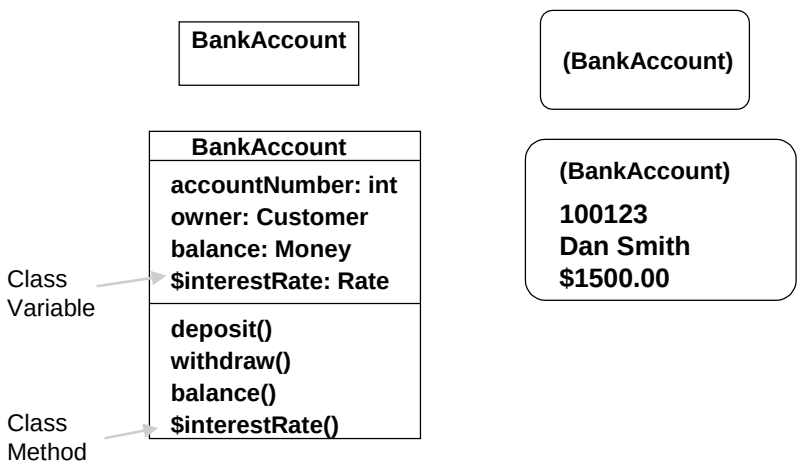

Reading and Writing Data

- **It is one thing to keep your constructed objects consistent with the data in the database,**
- **It is much more difficult to ensure when writing data that database stays consistent with object model**

Summary

- The issues here have been greatly over simplified
- It is a very challenging problem to store objects in relational databases and recover them properly
- Especially tricky with recursive references

Showing Class and Instance Details



OMT Notation Summary

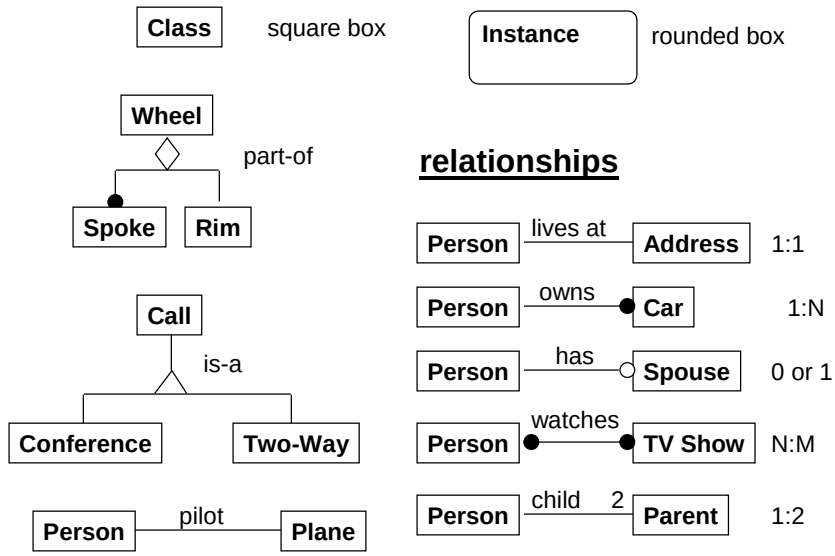


Figure 12. Mapping Classes to Tables

Object Model

Patient
patient name sin address

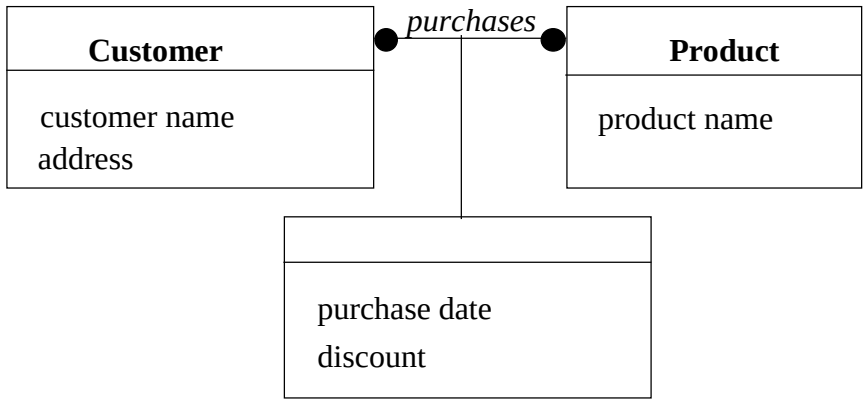
Table Model

Attribute name	Nulls?	Domain
patient-ID	N	long
patient name	N	string
sin	Y	long
address	Y	string

SQL Code

```
CREATE TABLE Patient
  (patient-ID          long          not null,
   patient -name       char(30)  not null,
   social-insurance-number long,
   address             char(30),
   PRIMARY KEY (patient-ID));
```

Figure 5. Link Attributes



Joe Smith	02/10/1997	3%	CD-ROM
Randy McGuire	05/10/1997	2%	Hard Disk
Susan Keiner	06/10/1997	3%	Speakers
Joe Smith	08/10/1997	3%	Hard Disk

Figure 6. Link Versus Object Attributes

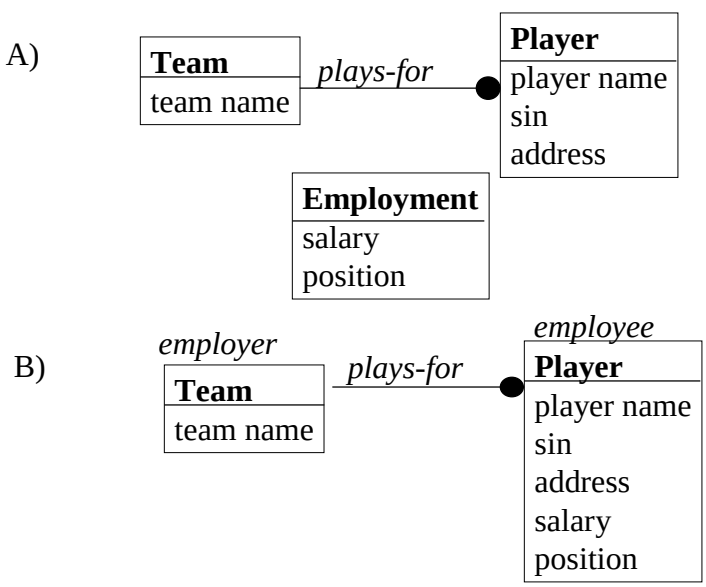
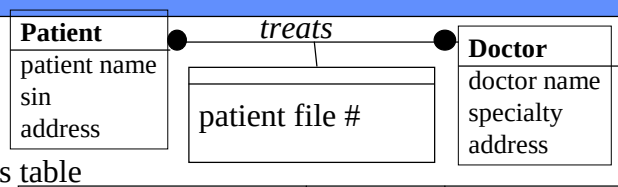


Figure 13. Mapping Many-To-Many Associations to Tables

Object Model



Treats table

Table Model

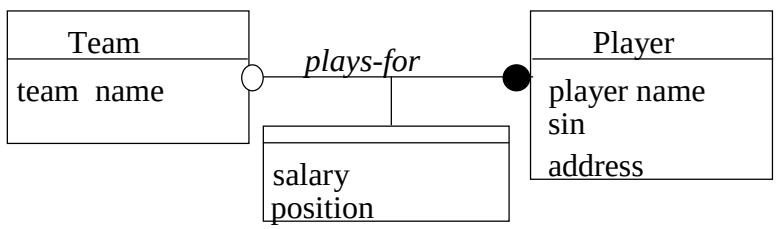
Attribute name	Nulls?	Domain
patient-ID	N	long
doctor-ID	N	long
patient file #	Y	long

SQL Code

```
CREATE TABLE Treats
        (patient-ID          long    not null,
        doctor-ID            long    not null,
        patient file #       long    not null,
        PRIMARY KEY (patient-ID, doctor-ID),
    FOREIGN KEY (patient-ID) REFERENCES Patient,
    FOREIGN KEY (doctor-ID) REFERENCES Doctor);
```


Mapping 1:N Associations to Tables, distinct association table

Object Model



Plays-for table

Table Model

Attribute name	Nulls?	Domain
team-ID	N	long
player-ID	N	long
salary	Y	double
position	Y	string

Mapping 1:N Associations to Tables, buried foreign key

Table Model

Player table

Attribute name	Nulls?	Domain
player-ID	N	long
player name	N	string
sin	Y	long
address	Y	string
team-ID	Y	long
salary	Y	double
position	Y	string

Figure 9. Investment Generalization Relationship

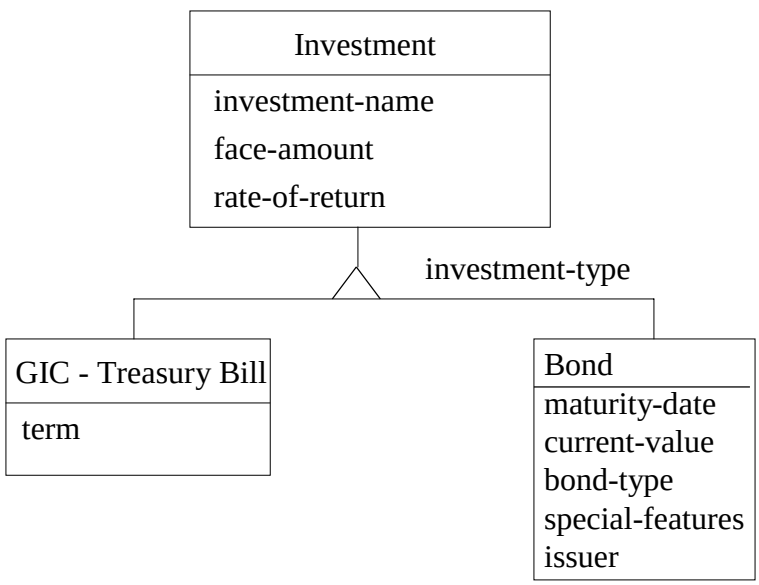


Figure 19. Mapping Generalizations to Tables, superclass & subclass tables

Investment Table

Attribute name	Nulls?	Domain
investment-ID	N	long
investment-name	N	string
face-amount	Y	double
rate-of-return	Y	double
investment-type	N	string

Bond Table

Attribute name	Nulls?	Domain
investment-ID	N	long
maturity-date	Y	date
current-value	Y	double
bond-type	Y	string
special-features	Y	string
issuer	Y	string

GIC - Treasury Bill Table

Attribute name	Nulls?	Domain
investment-ID	N	long
term	Y	long

Figure 20. Mapping Generalizations to Tables, many subclass tables

GIC - Treasury Bill Table

Attribute name	Nulls?	Domain
investment-ID	N	long
investment-name	N	string
face-amount	Y	double
rate-of-return	Y	double
term	Y	long

Bond Table

Attribute name	Nulls?	Domain
investment-ID	N	long
investment-name	N	string
face-amount	Y	double
rate-of-return	Y	double
maturity-date	Y	date
current-value	Y	double
bond-type	Y	string
special-features	Y	string
issuer	Y	string

Figure 21. Mapping Generalizations to Tables, one superclass table

Investment Table

Attribute name	Nulls?	Domain
investment-ID	N	long
investment-name	N	string
face-amount	Y	double
rate-of-return	Y	double
investment-type	N	string
term	Y	long
maturity-date	Y	date
current-value	Y	double
bond-type	Y	string
special-features	Y	string
issuer	Y	string

Figure 8. Aggregation