Appendix A – Results

This appendix shows the resulting mappings generated from the actual implementation framework in Chapter 7. kch_cuid_ represents the object’s identifier, kch_ref_ is its reference count, kch_value_ is the associated value if it is a primitive class, and the kch_FK_ prefix indicates foreign keys. These columns are added by the persistence broker for internal use. All four types of persistence mechanisms were used. The sample code used to generate these mappings can be found in Appendix B.10.

A.1 Class JDBC Mechanism

The object was saved using the JDBC-ODBC bridge driver into a Microsoft Access 2000 and Access 97 database.

```java
java_util_LinkedList

<table>
<thead>
<tr>
<th>kch_cuid_</th>
<th>kch_ref_</th>
<th>size</th>
<th>serialVersionUID</th>
<th>modCount_</th>
<th>kch_FK_header_</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.util.LinkedList@comp any1</td>
<td>1</td>
<td>11</td>
<td>876323262645 176354</td>
<td>11</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50_-8000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>kch_cuid_</th>
<th>kch_ref_</th>
<th>kch_FK_element_</th>
<th>kch_FK_next_</th>
<th>kch_FK_previous_</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50_-7fe9</td>
<td>2</td>
<td>employee@457895b8_df642ebf50_-7fe8</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50_-7fe7</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50_-7fe7</td>
</tr>
<tr>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50_-7fe9</td>
<td>2</td>
<td>employee@457895b8_df642ebf50_-7fe6</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50_-7fe5</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50_-7fe9</td>
</tr>
<tr>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50_-7fe9</td>
<td>2</td>
<td>employee@457895b8_df642ebf50_-7fe4</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50_-8000</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50_-7fe7</td>
</tr>
</tbody>
</table>
```

68
<table>
<thead>
<tr>
<th>kch_cuid</th>
<th>kch_ref</th>
<th>kch_FK_element</th>
<th>kch_FK_next</th>
<th>kch_FK_previous</th>
</tr>
</thead>
<tbody>
<tr>
<td>f642ebf50</td>
<td>-7fe5</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
<td>java.util.LinkedList$Entry@457895b8_df642ebf50</td>
</tr>
</tbody>
</table>

69
<table>
<thead>
<tr>
<th>Employee</th>
<th>Position</th>
<th>Salary</th>
<th>Height</th>
<th>Weight</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chew</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A AH</td>
</tr>
<tr>
<td>Tan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B TS</td>
</tr>
<tr>
<td>Lee</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C EL</td>
</tr>
<tr>
<td>Mak</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D BM</td>
</tr>
<tr>
<td>Han</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E HN</td>
</tr>
<tr>
<td>See</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F S</td>
</tr>
<tr>
<td>See</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>G S</td>
</tr>
<tr>
<td>kch_cuid_ref</td>
<td>kch_ref</td>
<td>kch_FK_pos</td>
<td>address_</td>
<td>salary_</td>
<td>weight_</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>------------</td>
<td>----------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>42ebf50_7ff2</td>
<td></td>
<td>7ffd</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>employee</td>
<td>457895b8_dff6e43cc794_8000</td>
<td>1</td>
<td>position</td>
<td>4338e829_dff6e43cc794_8000</td>
<td>12, Jln ABC</td>
</tr>
</tbody>
</table>

**position**

<table>
<thead>
<tr>
<th>kch_cuid_ref</th>
<th>kch_ref</th>
<th>kch_FK_dept</th>
<th>PositionID_</th>
<th>name_</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>457895b8_dff6e43cc794_8000</td>
<td>2</td>
<td>department</td>
<td>457895b8_dff6e43cc794_8000</td>
</tr>
<tr>
<td>position</td>
<td>457895b8_dff6e43cc794_8000</td>
<td>5</td>
<td>department</td>
<td>457895b8_dff6e43cc794_8000</td>
</tr>
<tr>
<td>position</td>
<td>457895b8_dff6e43cc794_8000</td>
<td>1</td>
<td>department</td>
<td>457895b8_dff6e43cc794_8000</td>
</tr>
<tr>
<td>position</td>
<td>457895b8_dff6e43cc794_8000</td>
<td>3</td>
<td>department</td>
<td>457895b8_dff6e43cc794_8000</td>
</tr>
</tbody>
</table>

**department**

<table>
<thead>
<tr>
<th>kch_cuid_ref</th>
<th>kch_ref</th>
<th>DepartmentID_</th>
<th>name_</th>
</tr>
</thead>
<tbody>
<tr>
<td>department</td>
<td>457895b8_dff6e43cc794_8000</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>department</td>
<td>457895b8_dff6e43cc794_8000</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>department</td>
<td>457895b8_dff6e43cc794_8000</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>department</td>
<td>457895b8_dff6e43cc794_8000</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

**A.2 Class FlatFileMechanism**

The object was saved into files in a subdirectory on the local machine's hard disk. Each file represents a table. The table name is shown in bold.

```
java.util.LinkedList

$RECORD
kch_cuid
java.lang.String
java.util.LinkedList@companyl
kch_ref
java.lang.Integer
1
size
java.lang.Integer
11
serialVersionUID
java.lang.Long
876323262645176354
modCount
java.lang.Integer
11
kch_FK_header
```
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-8000
$RECORD
kch_cuid
java.lang.String
java.util.LinkedList$Entry86a82d996_df708126e6_-7fff
kch_ref
java.lang.Integer
1
size
java.lang.Integer
0
serialVersionUID
java.lang.Long
876323262645176354
modCount
java.lang.Integer
0
kch_FK_header
java.lang.String
java.util.LinkedList$Entry86a82d996_df708126e6_-7ffe

java.util.LinkedList$Entry
$RECORD
kch_cuid
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7ff0
kch_ref
java.lang.Integer
2
kch_FK_element
java.lang.String
employee86a9c84d3_df63ecab92_-7ffa
kch_FK_next
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7ff7
kch_FK_previous
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7fff
$RECORD
kch_cuid
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7fe0
kch_ref
java.lang.Integer
2
kch_FK_element
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7fe0
kch_FK_next
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7f00
kch_FK_previous
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7ff1
$RECORD
kch_cuid
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7fe0
kch_ref
java.lang.Integer
2
kch_FK_element
java.lang.String
employee86a9c84d3_df63ecab92_-7f00
kch_FK_next
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7feb
kch_FK_previous
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7fe0
$RECORD
kch_cuid
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7f00
kch_ref
java.lang.Integer
2
kch_FK_element
java.lang.String
employee86a9c84d3_df63ecab92_-7f00
kch_FK_next
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7feb
kch_FK_previous
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7fe0
$RECORD
kch_cuid
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7f00
kch_ref
java.lang.Integer
2
kch_FK_element
java.lang.String
employee86a9c84d3_df63ecab92_-7f00
kch_FK_next
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7feb
kch_FK_previous
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7fe0
$RECORD
kch_cuid
java.lang.String
java.util.LinkedList$Entry86a9c84d3_df63ecab92_-7f00
kch_ref
java.lang.Integer
2
kch_FK_element

kch_FK_element
java.lang.Object
NULL
kch_FK_next
java.lang.String
java.util.LinkedList$Entry@6a82d996_df788126e6_7ffe
kch_FK_previous
java.lang.String
java.util.LinkedList$Entry@6a82d996_df788126e6_7ffe

employee

$RECORD
kch_cuid
java.lang.String
employee@6a9c84d3_df63ecab92_7ffe
kch_ref
java.lang.Integer
1
kch_FK_pos
java.lang.String
position@6a9c84d3_df63ecab92_7fff
d
address
java.lang.String
26, Jin 222
salary
java.lang.Integer
1500
weight
java.lang.Double
65.0
height
java.lang.Double
1.7
EmployeeID
java.lang.Integer
1
name
java.lang.String
Bak Mak
$RECORD
kch_cuid
java.lang.String
employee@6a9c84d3_df63ecab92_7ff9
kch_ref
java.lang.Integer
1
kch_FK_pos
java.lang.String
position@6a9c84d3_df63ecab92_7ff9
d
address
java.lang.String
12, Bugis St.
salary
java.lang.Integer
5000
weight
java.lang.Double
55.0
height
java.lang.Double
1.67
EmployeeID
java.lang.Integer
2
name
java.lang.String
TO Tan
$RECORD
kch_cuid
java.lang.String
employee@6a9c84d3_df63ecab92_7ffe
kch_ref
java.lang.Integer
1
Al Lee
500
20.0
6.7
1
Al Lee
500
20.0
6.7
1
MY Tan
500
20.0
6.7
1
name
java.lang.String
AB See
$RECORD
kch_cuid
java.lang.String
employee6a9c84d3_df63ecab92_-7ff6
kch_ref
java.lang.Integer
1
kch_FK_pos
java.lang.String
position6a9c84d3_df63ecab92_-7ff5
address
java.lang.String
1, Baker St.
salary
java.lang.Integer
500
weight
java.lang.Double
70.0
height
java.lang.Double
1.77
EmployeeID
java.lang.Integer
3
name
java.lang.String
AH Chew
$RECORD
kch_cuid
java.lang.String
employee6a9c84d3_df63ecab92_-7ff2
kch_ref
java.lang.Integer
1
kch_FK_pos
java.lang.String
position6a9c84d3_df63ecab92_-7ffd
address
java.lang.String
16, Jln SS3/16
salary
java.lang.Integer
1500
weight
java.lang.Double
65.0
height
java.lang.Double
1.7
EmployeeID
java.lang.Integer
4
name
java.lang.String
AF Chan
$RECORD
kch_cuid
java.lang.String
employee6a9c84d3_df63ecab92_-7ff0
kch_ref
java.lang.Integer
1
kch_FK_pos
java.lang.String
position6a9c84d3_df63ecab92_-7ffd
address
java.lang.String
12, Jln ABC
salary
java.lang.Integer
| EmployeeID | weight | height | name | UL Lim | $RECORD | kch_cuid | employee86a9c84d3_df63ecab92_-7fe8 | kch_ref | java.lang.Integer | 1 | kch_FK_pos | position86a9c84d3_df63ecab92_-7ff5 | address | java.lang.String | 4, JIn SS4/102 | salary | java.lang.Integer | 1253 | weight | java.lang.Double | 46.0 | height | java.lang.Double | 1.22 | EmployeeID | java.lang.Integer | 9 | name | java.lang.String | RH Bee | $RECORD | kch_cuid | employee86a9c84d3_df63ecab92_-7fe6 | kch_ref | java.lang.Integer | 1 | kch_FK_pos | position86a9c84d3_df63ecab92_-7ff5 | address | java.lang.String | 4, JIn SS4/102 | salary | java.lang.Integer | 10 | weight | java.lang.Double | 46.0 | height | java.lang.Double | 1.22 | EmployeeID | java.lang.Integer | 10 | name | java.lang.String | AH Beng | $RECORD | kch_cuid | employee86a9c84d3_df63ecab92_-7fe4 | kch_ref |
java.lang.Integer
1
kch_FK_pos
java.lang.String
position#6a9c84d3_df63ecab92_-7ff5
address
java.lang.String
5, Jln SS5/103
salary
java.lang.Integer
5
weight
java.lang.Double
46.0
height
java.lang.Double
1.22
EmployeeID
java.lang.Integer
11
name
java.lang.String
AH Lian

position
$RECORD
kch_cid
java.lang.String
position#6a9c84d3_df63ecab92_-7ff9
kch_ref
java.lang.Integer
2
kch_FK_dept
java.lang.String
department#6a9c84d3_df63ecab92_-7ff8
PositionID
java.lang.Integer
2
name
java.lang.String
Manager
$RECORD
kch_cid
java.lang.String
position#6a9c84d3_df63ecab92_-7ff5
kch_ref
java.lang.Integer
5
kch_FK_dept
java.lang.String
department#6a9c84d3_df63ecab92_-7ff4
PositionID
java.lang.Integer
1
name
java.lang.String
Janitor
$RECORD
kch_cid
java.lang.String
position#6a9c84d3_df63ecab92_-7ffd
kch_ref
java.lang.Integer
4
kch_FK_dept
java.lang.String
department#6a9c84d3_df63ecab92_-7ffc
PositionID
java.lang.Integer
3
name
java.lang.String
Programmer

79
A.3 Class EmailMechanism

This mechanism stores the table data in exactly the same form as FlatFileMechanism except that each table is stored as an e-mail on the server with the subject line representing the table name. Refer to Section A.2 for the actual stored data.

A.4 ObjectStoreMechanism

This mechanism saves the TableObj object directly into the database as it supports objects. The object data stored in this mechanism is of its own proprietary format which is inaccessible and thus cannot be shown here.
Appendix B – User Guide

This section contains the user guide for the persistence framework, and describes the usage and programming interface in Java. A complete example program is given in the last section.

B.1 Creating the Broker

To use the services of the persistence framework, the user first needs to create the persistence broker object. Only one instance of the broker is needed. The user just calls the default constructor provided, as follows:

```java
PersistenceBroker pb=new PersistenceBroker();
```

B.2 Connecting to a Data Source

The user then needs to specify the data source to connect to by passing the data source name, username, and password. The data source name is in the form of a Uniform Resource Locator (URL) format as follows:

```
[sourcetype]:[persistence mechanism-specific name]
```

The framework has four built-in persistence mechanism types, with the following URL formats:

**Flat files:**  
`file:[directory name]`

E.g.  
`file:/usr/user/tmp`

`file:c:\windows\temp`

A flat file URL is prefixed by the type `file:` followed by a directory name in a file system. The directory must exist and the user must have read and write permission for it.

**Email:**  
`email:[popserver],[smtpserver]`
E.g. email:siswazah.fsktm.um.edu.my
     email:pop.mail.yahoo.com,smtp.mail.yahoo.com

The user needs to specify the Post Office Protocol (POP) server where mail is to be received. The actual mail is sent to [username]@[popserver] using the Simple Mail Transfer Protocol (SMTP) server. In certain cases where both the POP and SMTP servers are the same, then only one server needs to be specified.

JDBC: jdbc:[source name]

E.g. jdbc:dceming:MyDatabase
     jdbc:odbc:CompanyDB

The Java Database Connectivity (JDBC) mechanism is a native protocol supported under Java. The source name is driver-dependent and may contain subprotocol names, server addresses, and other parameters. Most database vendors provide drivers for JDBC which should contain instructions on connecting to a database.

ObjectStore: objstore:[database name]

E.g. objstore:mydatabase.odb
     objstore:c:\tmp\db\employee.odb

This persistence mechanism connects to Object Design’s ObjectStore object-oriented database. The database name is specified as a filename ending the with extension .odb. If the database does not exist, it will be created.

The actual connection is done by calling the broker’s connect method, passing the data source name, username, and password as follows:

    int handle=pb.connect(“jdbc:odbc:MyEmployee”,”user”,”pass”);

The method will return an integer handle representing the connection which will need to be passed to other broker methods. The broker supports a maximum of 255 simultaneous connections.
B.3 Saving Objects

To store objects in the persistence mechanism the user has to pass the connection handle, the object itself, and an arbitrary name for the object as follows:

```java
int save(int handle, Object obj, String name)
```

E.g.

```java
String myobj="This is my object";
```

```java
pb.save(handle, obj,"myobj");
```

The `save` method returns an integer value of zero indicating success, or a non-zero value indicating failure.

B.4 Restoring Objects

To restore a previously saved object, the `restore` method is used:

```java
Object restore(int handle, String classname, String name)
```

E.g.

```java
String myobj=(String)pb.restore(handle,"java.lang.String","myobj");
```

The user has to pass the handle, class name, and the object name. The class name must be specified as a fully qualified name, preceded by its package names. The restored object must be cast into the appropriate type by the user. Repeated calls to `restore` will not generate duplicate objects, but instead will return the same object overwritten with data from the persistence mechanism. In the event of failure, a null object reference is returned.

B.5 Deleting objects
The delete method has the following parameters:

```java
int delete(int handle, String classname, String name)
```

E.g. `pb.delete(handle,"java.lang.String","myobj");`

The parameters passed are exactly the same as those for the restore method. The specified object will be deleted from the persistence mechanism. The return value is an integer, zero indicating success and non-zero indicating failure.

### B.6 Updating objects

Once the object has been saved, its fields may be updated with new values by calling the update method as below:

```java
int update(int handle, String classname, String setcol, Object val,
            String cmpcol, Object cond)
```

E.g. `pb.update(handle, "employee","salary", new Integer(2800),
                "name","John Doe");`

In this example, the broker will look for all objects of the class “employee”, then finds the first object with the attribute “name”="John Doe" and sets its attribute “salary”=2800. To set or compare Java primitives, it must first be wrapped in its wrapper class, as shown in the example. The method returns zero on success, non-zero on failure.

### B.7 Queries

If the underlying persistence mechanism supports SQL queries, then the query method can be used for simple SQL queries.

```java
Object[] query(int handle, String sql)
```

E.g. `Object[] res=query(handle,"select * from employee;");`
The SQL statement syntax is as follows:

```
select [attribute name],[attribute name],...
from [class name]
where [attribute name][operator][value] and
    [attribute name][operator][value] and ...
```

The attribute names correspond to the object attributes, with nested attributes supported in the where section but not the select section. The from section only allows one class name. The where section specifies the conditions to be imposed, with operator representing the SQL comparison operators, =, ! =, <>, <, >, <=, >=. Only the and boolean operator is supported. The method then returns an array of objects that satisfy the query. Only the selected object attributes are initialized.

Examples of supported SQL statements are as follows:

```
select name,address from employee where salary>2000;

select name,address,salary from employee where position='Manager';

select * from employee where position.dept.name='R&D' and position='Programmer' and salary<=3000;

select position from employee where position!='Manager' and position.dept!='R&D';
```

B.8 Transactions

The user can call the following methods for transaction management:

```
boolean supportsTransactions(int handle)
```

This determines whether the underlying persistence mechanism supports transactions.

```
int getTransactionIsolation(int handle)
```

This returns an integer value indicating the current transaction isolation level. The return value corresponds to equivalent JDBC transaction isolation levels as follows:

```
0  - java.sql.Connection.TRANSACTION_NONE
```
Indicates that transactions are not supported.

1       - java.sql.Connection.TRANSACTION_READ_UNCOMMITTED

Dirty reads, non-repeatable reads and phantom reads can occur. This level allows a row changed by one transaction to be read by another transaction before any changes in that row have been committed (a "dirty read"). If any of the changes are rolled back, the second transaction will have retrieved an invalid row.

2       - java.sql.Connection.TRANSACTION_READ_COMMITTED

Dirty reads are prevented; non-repeatable reads and phantom reads can occur. This level only prohibits a transaction from reading a row with uncommitted changes in it.

3       - java.sql.Connection.TRANSACTION_REPEATABLE_READ

Dirty reads and non-repeatable reads are prevented; phantom reads can occur. This level prohibits a transaction from reading a row with uncommitted changes in it, and it also prohibits the situation where one transaction reads a row, a second transaction alters the row, and the first transaction rereads the row, getting different values the second time (a "non-repeatable read").

4       - java.sql.Connection.TRANSACTION_SERIALIZABLE

Dirty reads, non-repeatable reads and phantom reads are prevented. This level includes the prohibitions in TRANSACTION_REPEATABLE_READ and further prohibits the situation where one transaction reads all rows that satisfy a WHERE condition, a second transaction inserts a row that satisfies that WHERE condition, and the first transaction rereads for the same condition, retrieving the additional "phantom" row in the second read.

```java
int setTransactionIsolation(int handle, int level)
```

This method allows the user to set the desired transaction isolation level (0 to 4) as described above. A return value of zero indicates success, non-zero indicates failure.

```java
boolean supportsTransactionIsolationLevel(int handle, int level)
```

This determines whether the transaction isolation level is supported.
B.9 Disconnecting

When user is done with a particular persistence mechanism, the `disconnect` method is called:

```java
int disconnect(int handle)
```

This will close the connection to the corresponding persistence mechanism. A zero value is returned on success, non-zero on failure.

B.10 Example program

This section gives a sample program to demonstrate the use of the persistence framework.

```java
import java.lang.reflect.*;
import java.rmi.server.*;
import java.util.*;
import java.util.zip.*;
import java.io.*;
import kch.*;

class human {
    public double height;
    public double weight;
}
class person extends human {
    public String name;
    public String address;
}
class employee extends person {
    public int EmployeeID;
    public position pos;
    public int salary;
    public employee(String nam, String addr, double hgt, double wgt, int e, position p, int s)
        {name=nam;address=addr;height=hgt;weight=wgt;EmployeeID=e;pos=p;salary=s;}
    public void println()
        { System.out.println(name+" "+address+" "+height+" "+weight+" "+EmployeeID+" "+pos.toString()+" "+salary);}
}
class position {
    public int PositionID;
    public String name;
    public department dept;
    public position(int p, String n, department d) {PositionID=p;name=n;dept=d;}
    public String toString()
        { return name+" "+dept.toString();}
```
class department {
  public int DepartmentID;
  public String name;
  public department(int d, String n) {DepartmentID=d; name=n;}
  public String toString() {
    return name;
  }
}

class demo {

  public static void main(String args[])
  {
    try {
      PrintStream pis=new PrintStream(new FileOutputStream("err.log"));
      System.setErr(pis);

      department d1=new department(1,"Sanitation");
      department d2=new department(2,"Administration");
      department d3=new department(3,"R&D");
      position p1=new position(1,"Janitor",d1);
      position p2=new position(2,"Manager",d2);
      position p3=new position(3,"Programmer",d3);
      employee[] e=new employee[11];
      e[0]=new employee("BB Mak","26, Jln 222",1.70,65,1,p3,1500);
      e[1]=new employee("TO Tan","12, Bugis St.",1.67,55,2,p2,5000);
      e[2]=new employee("AH Chew","1, Baker St.",1.77,70,3,p1,500);
      e[3]=new employee("AF Chan","16, Jln SS3/16",1.70,65,4,p3,1500);
      e[4]=new employee("UL Lim","12, Jln ABC",1.27,58,5,p3,3400);
      e[5]=new employee("AL Lee","133, Jln 123.",0.77,20,6,p1,600);
      e[6]=new employee("MY Tan","2, Jln SS2/100",2.70,165,7,p3,1200);
      e[7]=new employee("AB See","3, Jln SS3/101",1.27,51,8,p2,8000);
      e[8]=new employee("RH Bee","4, Jln SS4/102",1.22,46,9,p1,1253);
      e[9]=new employee("AH Beng","4, Jln SS4/102",1.22,46,10,p1,10);
      e[10]=new employee("AH Lian","5, Jln SS5/103",1.22,46,11,p1,5);
      LinkedList mycompany=new LinkedList();
      for(int n=0;n<e.length;n++)
        mycompany.add(e[n]);
      PersistenceBroker pb=new PersistenceBroker();
      System.out.println("connecting...");
      int handle=pb.connect("jdbc:odbc:MyEmployee","khor","" ;
      //int handle=pb.connect("file:tmp","khor","" ;
      //int handle=pb.connect("email:kch2","khor","" ;
      if(handle==-1) System.out.println("cannot connect");

      // ***********test save***********
      System.out.println("saving...");
      pb.save(handle,mycompany,"company1");

      // ***********test restore***********
      System.out.println("restoring...");
      LinkedList company1=(LinkedList)pb.restore(handle,"java.util.LinkedList","company1");
      for(int n=0;n<company1.size();n++)
        ((employee)company1.get(n)).println();

      // ***********test query***********
      LineNumberReader lnr=new LineNumberReader(new InputStreamReader(System.in));

      88
String insql;
Object[] res;
while(true)
{
  System.out.print("Enter query: ");
  insql=lnr.readLine();
  if(insql.toLowerCase().equals("quit")) break;
  res=pb.query(handl,insql);
  if(res==null) System.out.println("query fail");
  for(int n=0;n<res.length;n++)
    ((employee)res[n]).println();
}

// ****************test update**************
  department tmpdept=new department(4,"F&B");
  position tmnpos=new position(4,"Tea Lady",tmpdept);
  pb.update(handl,"employee","pos",tmnpos,"name","UL Lim");

// ****************test delete**************
  pb.delete(handl,"java.util.LinkedList","company1");

  pb.disconnect(handl);
  pis.close();
}
  } catch(Exception e) {System.out.println(e);}