O-R Mapping

What is it and should we be using it?

Object-Relational Mapping

- Object-Relational mapping is a mapping of objects to tables in relational databases.
- In our case, this means mapping Java objects to Oracle SQL tables.
- Two of the four primary alternatives to Java and DB development use O-R mapping:

Alternative 1: Non-OO Java!

1. Use Java in a non-object-oriented way, with static methods. This basically means that classes in our program merely represent independent program modules, and not conceptual objects.

   This is our current development model – it is similar to programming in a procedural language like C.

Alternative 2: Manual O-R

2. Use Java in an O-O way, representing concepts in our applications as objects, for example, **User**, **Contact**, **Calendar**, **Plan**, **Project**, **Group**, etc.

   Here we manually create code to map from the database tables to objects and back again.

   This is unwieldy and creates too many problems trying to keep data consistent. We have tried this and it doesn't work well.

Alternative 3: Object Databases

3. This is where we use databases that store objects directly, rather than using a conventional relational DB model.

   Although at first this seems like a good option, such databases are not currently a scalable option. They cannot handle large volumes of data.

Alternative 4: Automated O-R

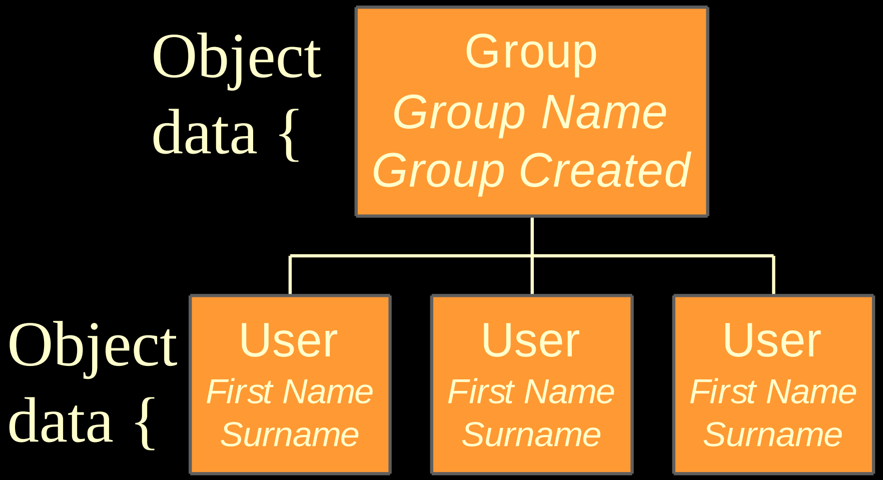
4. Purchase or create (!) O-R mapping tools which enable us to program in an object-oriented way.

   This provides automatic creation of code to map between objects and database tables, removing this concern for the developer.

   Illustrating further:

Example of Automated O-R - Objects

Object Relationships



- This is a snapshot of objects in memory as they might occur in our system.

Example of Automated O-R - Database

- This is an example of the type of database schema we might be using.

User ID	First Name	Surname
1	Fred	Smith
2	George	Jones
3	Mary	Simpson

Group ID	User ID
3	1
3	2
4	3
4	2

Group ID	Group Name	Group Created
3	Group 1	01/01/2000
4	Group 2	01/03/2000

Mapping

- With automated tools, one can create a mapping from the schema to the objects. These tools then generate the Java code to map between the objects and the DB automatically.

Mapping

- Dealing with foreign key constraints is one of the primary problems with Manual O-R code, and one of the main advantages of using Automated O-R tools, because:

- Most mapping tools understand that a foreign key is equivalent to object aggregation in Java, and can take this into account, to ensure correctness of the Java code automatically generated to handle the database.

Java Blend

- A typical Automated O-R tool is Java Blend:

Default DB Mapping

Each table

Each Java Class

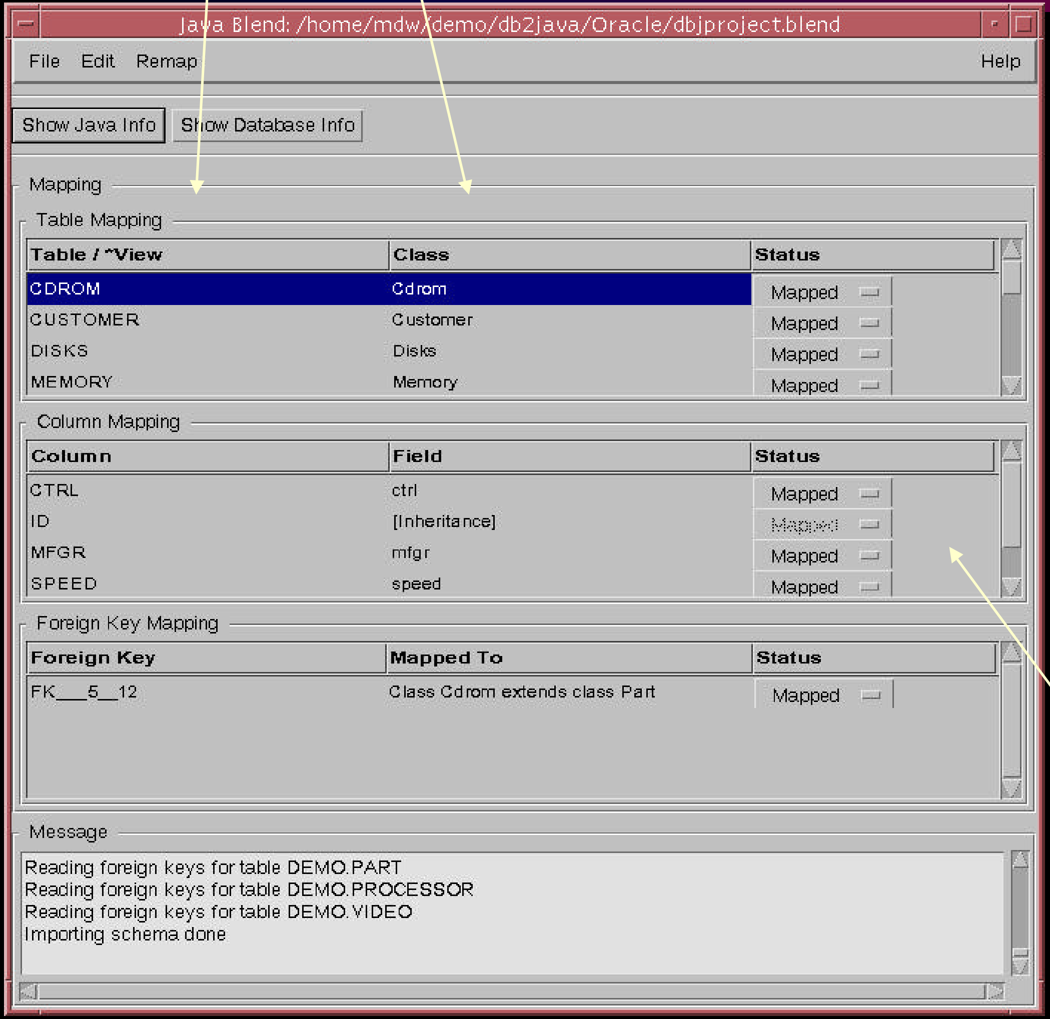


Table / ^View	Class	Status
CDROM	Cdrom	Mapped
CUSTOMER	Customer	Mapped
DISKS	Disks	Mapped
MEMORY	Memory	Mapped

Column	Field	Status
CTRL	ctrl	Mapped
ID	[Inheritance]	Mapped
MFGR	mfgr	Mapped
SPEED	speed	Mapped

Foreign Key	Mapped To	Status
FK___5__12	Class Cdrom extends class Part	Mapped

```
Reading foreign keys for table DEMO.PART
Reading foreign keys for table DEMO.PROCESSOR
Reading foreign keys for table DEMO.VIDEO
Importing schema done
```

Java Blend allows visual manipulation of the mapping between objects and tables before automatic creation of the Java code.

Column and field mapping for each table / class

Recommendation

- My recommendation <u>in the long term</u> is this fourth approach – using O-R Automated tools such as Java Blend.

Advantages of Automated O-R

- It reduces the code each developer has to write. Moreover, this is tedious and repetitive code.
- Reorganisation of Java code or DB schemas is made less painful, because large amounts of code can be re-generated automatically.

Advantages of Automated O-R

- Off-the-shelf tools already exist which have claimed scalability to enterprise size. I have not had the resources to test whether this is the case.
- However, since the tools generate pure Java JDBC code (certainly Java Blend does), they have the <u>potential</u> to be as good as anything we could write.

Places where we could apply O-R

O-R could be useful in two primary locations:

1. **ApoApsis Core** – although this is currently a small set of code, I predict it will grow as we ‘discover’ desired features for applications we develop, which are required again and again.
2. **Applications themselves** – each application will obviously require their own ‘concepts’ which will embodied as classes and tables – the O-R tool will aid rapid development of these applications (RAD!).

Disadvantages

- However, there are disadvantages of using O-R mapping tools, and this is why we are not currently using them:

Disadvantages

- They cost money – e.g. Java Blend, $1000 / developer seat.
- They tie us into a technology. This means our entire architecture relies on a certain tool. We want the architecture to be as system-independent as possible: hence why we are currently using Java and JDBC-only DB access.

Conclusion

- The rollout time for our initial applications and core would be far greater if we were to use Automated O-R tools now. But in the medium to long term, I believe we should aim to migrate to that model, since it allows for far easier and more rapid development.

Conclusions

- Full evaluation of all the Automated O-R tools, and their scalability and practicality would of course be required.

- Also, we would need to evaluate the implications of technology tie-in, and investigate how difficult it would be to migrate away from a tool as well if required.