FOREWORD

This is the twelfth in the series of highly successful international workshops on the Teaching, Learning and Assessment of Databases (TLAD 2014). TLAD 2014 is held on the 4th July at Southampton Solent University, Maritime and Technology Faculty, and hopes to be just as successful as its predecessors.

The teaching of databases continues to be central to all Computing Science, Software Engineering, Information Systems and Information Technology courses. This year's workshop continues the tradition of bringing together database teachers, researchers and practitioners in order to share good learning, teaching and assessment practice and experience, and further the growing community amongst database academics. As well as attracting academics and teachers from the UK community, the workshop has also been successful in attracting academics from the wider international community, through serving on the programme committee, and attending and presenting papers.

This year, the workshop includes six refereed papers which cover a number of themes: NoSQL and Big Data, teaching approaches, and Games and tools for database teaching. With this year's focus on Big Data, we look forward particularly to our keynote presentation which will give us an industrial perspective of big data. Entitled "Big Data in the Third Platform", the keynote will be presented by James T (JT) Lewis, Partner Sales Director Distribution at EMC Computer Systems (UK) Ltd.

We would like to thank members of the programme and steering committees for their reviews and their continuing support of this workshop. Many members have been involved in the workshops since the first TLAD, thus showing the strength of the database teaching community both within the UK and overseas. We would also like to thank the Higher Education Academy, especially Karen Fraser, for their continued support.

Petra Leimich, Alastair Monger
Workshop co-Chairs

STEERING COMMITTEE
Petra Leimich (University of Abertay Dundee) Workshop co-Chair
Alastair Monger (Southampton Solent University) Workshop co-Chair
David Nelson (University of Sunderland)
Bernadette-Marie Byrne (University of Hertfordshire)

PROGRAMME COMMITTEE
Jagdev Bhogal (Birmingham City University)
Bernadette Marie Byrne (University of Hertfordshire)
Fang Fang Cai (London Metropolitan University)
Jackie Campbell (Leeds Metropolitan University)
Karen Davis (University of Cincinnati)
Mary Garvey (University of Wolverhampton)
Petra Leimich (University of Abertay Dundee)
Pirjo Moen (University of Helsinki)
Alastair Monger (Southampton Solent University)
David Nelson (University of Sunderland)
James Paterson (Glasgow Caledonian University)
Clare Stanier (University of Staffordshire)
Tony Valsamidis (University of Greenwich)

HIGHER EDUCATION ACADEMY
Karen Fraser (HEA contact)
<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keynote: Big Data in the Third Platform</td>
<td>7</td>
</tr>
<tr>
<td>James T Lewis (speaker information)</td>
<td></td>
</tr>
<tr>
<td>Teaching NoSQL with RavenDB and Neo4j</td>
<td>9</td>
</tr>
<tr>
<td>James Paterson and Eddie Gray (Glasgow Caledonian University)</td>
<td></td>
</tr>
<tr>
<td>Working with Newer Data Management Technologies</td>
<td>17</td>
</tr>
<tr>
<td>Clare Stanier and Emmanuel Isitor (Staffordshire University)</td>
<td></td>
</tr>
<tr>
<td>Practice makes perfect: Improving success rates in a Database Design module</td>
<td>25</td>
</tr>
<tr>
<td>Peter Stordy and Paul Clough (Sheffield University)</td>
<td></td>
</tr>
<tr>
<td>Contextualised Problem-based Approach for Teaching Second Year Undergraduate Database Module</td>
<td>31</td>
</tr>
<tr>
<td>James Xue (University of Northampton)</td>
<td></td>
</tr>
<tr>
<td>Using Multiplayer Game Development to Integrate HND Modules and Develop Practical Understanding of Database Concepts</td>
<td>41</td>
</tr>
<tr>
<td>Joseph Hanke, Sage Lal and Dilaksha Attanayake (North East Surrey College of Technology (Nescot))</td>
<td></td>
</tr>
<tr>
<td>EDNA, a Software Tool for Verifying Normalisation of Relations during Logical Database Design Process</td>
<td>51</td>
</tr>
<tr>
<td>Adam Floyd, Hongbo Du (University of Buckingham)</td>
<td></td>
</tr>
</tbody>
</table>
Keynote: Big Data in the Third Platform
James T Lewis

James T. Lewis (JT)
Partner Sales Director Distribution, United Kingdom & Ireland
EMC Computer Systems (UK) Ltd
Jt.lewis@emc.com

JT has responsibility for managing the Partner Distribution Supply Chain for the EMC UK&I Division –primarily focused on sales, business development, and operational management. In addition to performing the associated regional tasks for this role, JT also participates on the Partner Sales Managers Leaders Team for the EMEA Distribution Team.

Prior to joining the UKI Division in January 2011 JT performed the role of Regional Practice Manager for the Department of Defense in the EMC Federal Residence Practice beginning in January 2007.

Before joining EMC, JT held technology leadership positions for the United States Army Europe Headquarters, in Heidelberg Germany ranging from Information Technology Operations to Governance and IT Security. In this role JT was responsible for IT Operations in Europe the Horn of Africa and throughout the Middle East.

Having spent over 15 years in the technology industry, JT has a variety of experiences in Multi-Site Technology Operations, Quality Performance Improvement, and Strategic & Operational Planning. He has been innovative in developing solutions for customers and organizations. While in the Federal Division he was instrumental in the growth of the practice helping them to achieve double digit growth for three consecutive years. Since 2012 JT has been on the UK&I Channel Team as the Global Services Lead growing the services revenue to over $35M an improvement by $27M over the two year period. JT is a graduate of Troy State University in Troy, Alabama and various leadership academies with the US Army including, The Basic and Advanced Non-Commissioned Officers Leadership Courses. JT holds several Industry Certifications in Information Technology.
Teaching NoSQL with RavenDB and Neo4j

James H. Paterson
Glasgow Caledonian University
Cowcaddens Road
Glasgow G4 0BA
James.Paterson@gcu.ac.uk
http://www.gcu.ac.uk/ebe/staff/jpaterson

Edwin Gray
Glasgow Caledonian University
Cowcaddens Road
Glasgow G4 0BA
E.Gray@gcu.ac.uk
http://www.gcu.ac.uk/ebe/staff/egray

Abstract
There is a new trend towards polyglot persistence: the use of a variety of different data storage, including relational databases and NoSQL datastores, for different applications. This paper summarises the main categories of NoSQL datastores and describes examples used in teaching NoSQL in a practical context focused on the .NET platform. Specific datastore systems were chosen on the basis of availability of libraries to support .NET integration and of suitable tools for visualisation of the data. RavenDB was used as an example of a document store, while Neo4j was used as an example of a graph database.

Keywords
NoSQL, database curriculum, document store, graph database

1. Introduction
New “Big Data” needs which emerged in the 2000’s led to the emergence of a family of datastores commonly described as NoSQL. This terminology differentiates these datastores from the relational databases which have long been the mainstay of data storage, but which companies such as Google and Amazon found did not scale well to handle the needs of their huge user bases. There is now a trend towards polyglot persistence (Leberknight 2008): the use of a variety of different data stores for different data. There are many different NoSQL datastores available, and these are broadly classified into four types: key-value stores, document stores, wide column stores and graph databases. While NoSQL datastores are often referred to as “schemaless”, in software applications which use them there is usually a well-defined schema which is often defined or constructed by the application itself rather than pre-defined within the database.

In a previous TLAD paper, Stanier (2012) described the challenge that this trend presents for database teaching and made general recommendations for pedagogy and resources. This paper describes the specific approach and the resources used in introducing NoSQL databases in a practical context within a postgraduate module which focuses on the integration between software applications on the .NET platform and data. The aim was to illustrate by example the differences between relational and NoSQL data, and also the diversity in data models and usage across the NoSQL landscape.
2. NoSQL Datastore Types

There is no prescriptive definition of NoSQL, but it is possible to identify a set of characteristics which are common to most datastores which are considered to be NoSQL:

- Support for massive data storage (petabyte+)
- Distribution across multiple servers using sharding and replication
- “Schema-less” data – can dynamically add new attributes to data records
- Weaker consistency model than most relational databases

There is, however, significant diversity in other characteristics among NoSQL datastores, most evidently the data models which they implement, reflecting the diversity in the purposes for which these datastores were developed. There are four main families of NoSQL databases (NoSQL Databases, 2014), as summarised in Table 1. Sadalage and Fowler (2013) further characterise these families depending on whether or not they are aggregate oriented, also indicated in the table.

<table>
<thead>
<tr>
<th>Family</th>
<th>Characteristics</th>
<th>Aggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key/value stores</td>
<td>Manage a simple value indexed by a key</td>
<td>Yes</td>
</tr>
<tr>
<td>Document stores</td>
<td>Multi-field indexable/queryable documents, usually represented as JSON</td>
<td>Yes</td>
</tr>
<tr>
<td>Wide column stores</td>
<td>Manage data in records with an ability to hold large numbers of dynamic columns</td>
<td>Yes</td>
</tr>
<tr>
<td>Graph databases</td>
<td>Represent entities and relationships by storing nodes, edges, and properties</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 1. NoSQL database families

Aggregate orientation, which is a feature common to three of the four main categories of NoSQL datastore, recognises that there is often a requirement to store structures which are more complex than the simple tuple structure supported by relational databases. Sadalage and Fowler (2013) refers to these more complex records, which may include, for example, nested entities and nested lists, as aggregates. Of course, relational databases can store and retrieve arbitrarily complex data through foreign keys and joins. In an aggregate oriented database, however, all the component parts of a single aggregate can be stored together rather than split between multiple tables. This is useful when distributing data across many separate nodes, or shards, which is a technique commonly, used to meet the throughput requirements associated with Big Data. There is a high probability that for a given operation, all of the required data is contained within the same aggregate, and hence will be stored within the same shard and can be retrieved efficiently even in a highly distributed datastore. While key/value, document and wide column stores differ in the form and complexity of their aggregates, they all support this basic scenario.

When working with large volumes of distributed data, MapReduce is a powerful technique for condensing data into useful aggregated results (Dean and Ghemawat, 2008). Map-reduce operations can process data in parallel very efficiently in a sharded environment. The technique is supported by many aggregate-oriented datastores and is an important aspect of learning to use these.

In some data storage scenarios the nature and meaning of the relationships between entities is as important as the entities themselves. For example, recommendation systems in e-commerce applications are based on the connections established between customers and products through customer activity. Relational databases, ironically, are not particularly good at storing relationships.
The foreign key is essentially a mechanism to allow related records to be located, but it doesn’t capture any meaning to the relationship. Aggregate-oriented NoSQL datastores don’t help as they are optimised for data related through belonging to an aggregate, and joining data from different aggregates requires a crude process of matching object identifiers. However, the fourth category of NoSQL datastores listed in Table 1, graph databases, explicitly embrace relationships (Robinson et al., 2014). In a graph database, simple abstractions of nodes and edges, each with their own properties, can be assembled into complex models which map closely to the problem domain in a scenario and which can be queried and explored efficiently through relationships.

3. **Practical NoSQL Examples**

The examples presented here were used in a module on a Masters course on web development with a specific focus on the .NET framework. The module studies the use of data storage solutions within .NET applications. Given the limited time available within the module to cover NoSQL, it was decided to limit the scope to examples of one type of aggregate-oriented data store and a graph datastore to emphasis the diversity with NoSQL. Specific datastore systems were chosen on the basis of availability of libraries to support .NET integration and of suitable tools for visualisation of the data.

3.1 **Document Store**

For the aggregate-oriented datastore, a document store seemed the most suitable choice as the data model can allow exploration of more complex structures and design issues than a key-value store, without the greater level of complexity of wide column stores. RavenDB was used as the main example of a document store, as it is prominent within the .NET community and is closely integrated with .NET. It also provides a simple mechanism to implement and illustrate the use of map-reduce techniques. However, MongoDB is by some margin the most popular document store (DB-Engines, 2014), and some material on it was also included. The example scenario, based on an order system, was designed to illustrate: an aggregate data model; querying against the model; and MapReduce.

The key consideration for data model design is the choice of embedding related entities or using references (MongoDB, 2014). These options are illustrated by the Order document in Figure 1. Each Order is an aggregate, with a reference to a related Customer aggregate. Product entities are embedded in the document. There is scope for students to discuss likely query patterns and alternative versions of this design.

```json
{
    "CustomerId": "CUST2",
    "OrderDate": "2013-03-02T00:00:00.0000000",
    "OrderLines": [
        {
            "Product": {
                "Id": "PROD1",
                "Description": "Baked Beans 250g",
                "UnitPrice": 0.45
            },
            "Quantity": 24
        },
        ...
    ]
}
```

*Figure 1. Document design example*
To create a document in RavenDB from C# code, classes must be created to implement the document “schema”. Note that the datastore is schemaless, in that it will store documents without any schema being predefined within it. However, in a given scenario there is usually a structure defined somewhere, in this case within the C# application. Figure 2 shows code to create an instance of the Customer class and store it.

```csharp
using (IDocumentSession session = store.OpenSession()){
    Customer cust1 = new Customer {
        Id = "CUST1",
        Name = "Fernando",
        Address = "1 First Street"
    };
    session.Store(cust1);
}
```

**Figure 2. Storing a document in RavenDB from a C# application**

Students need to learn that NoSQL datastores typically have their own query mechanisms which must be learned as required. This is offset to some extent for .NET developers, as datastore clients which support .NET often support the LINQ feature of that framework, and can be queried in a consistent way. Figure 3 shows a LINQ query which finds all Order documents which contain a specified Product.

```csharp
var target = "PROD3";
var query1 = session.Query<Order>()
    .Where(o => o.OrderLines.Any( 
        i => i.Product.Id == target))
    .ToList();
```

**Figure 3. A RavenDB LINQ query**

This scenario provides a simple example of a query which can be solved using MapReduce. We want to find the total number ordered of each Product. There is a Quantity associated with each Product in an OrderLine. The MapReduce query runs in two stages.

**Map** - Each Order is examined and each Product ID found is emitted as a key, along with the associated Quantity. A Product ID may exist in many aggregates, but each aggregate is independent, so the datastore can divide the workload into units which run in parallel

**Reduce** – The values for each emitted key from all the documents are combined – all Map results with a given Product ID key are reduced down to a single result which sums their quantities. For large data sets, the datastore may be able to parallelise reduce by partitioning the Map results and combining in multiple reduce cycles

In RavenDB this MapReduce query can be performed by adding an index to the database. Figure 4 shows C# code to do so.
```csharp
public class ProductsOrderedIndex : AbstractIndexCreationTask<Order, ProductsOrderedResult>
{
    public ProductsOrderedIndex()
    {
        Map = orders => from ord in orders
                        from line in ord.OrderLines
                        select new {
                            ProductId = line.Product.Id,
                            Count = line.Quantity
                        };
        Reduce = results => from result in results
                            group result by result.ProductId into g
                            select new {
                                ProductId = g.Key,
                                Count = g.Sum(x => x.Count)
                            };
    }
}
```

Figure 4. Defining a RavenDB Map-reduce index

Students were also encouraged to explore the RavenDB Studio web interface which can be used to view documents and define and execute queries. Figure 5 shows RavenDB Studio being used to view an Order document in the datastore.

Figure 5. RavenDB Studio

3.2 Graph Database

Neo4j was used as an example of a graph database. Neo4j is written in Java, but supports a wide range of client types, including C# and REST clients. The example scenario is based on the idea of a social network which consists of members who may have connections to each other, the places they live, the schools they graduated from and so on. Figure 6 shows an example of the way this data may be represented as a graph. The Graduated From relationship carries an additional piece of information: the year of graduation. These entities are represented as nodes, and may have additional relationships to other nodes in the database. For example, there may be many other members who graduated from the school GCU.
Students were shown how to define entity and relationship data using C# classes. As before, the classes define the “schema” for the scenario, while the database is able to store data without prior knowledge of the structure. Figure 7 shows a C# class which defines the Graduated From relationship and the code to add a relationship between Member and School entities, which are themselves defined by classes. Another class, GraduatedFromData, encapsulates the year of graduation for this relationship.

```csharp
public class GraduatedFromRelationship : Relationship<GraduatedFromData>, IRelationshipAllowingSourceNode<Member>, IRelationshipAllowingTargetNode<School>
{
    public static readonly string TypeKey = "GRADUATEDFROM";
    public GraduatedFromRelationship(NodeReference targetNode, GraduatedFromData data): base(targetNode, data) {}
}

client.CreateRelationship(member1, new GraduatedFromRelationship(sch1, new GraduatedFromData(2002)));
```

Querying graph databases requires a specialised graph query language. Neo4j supports two: Cypher and Gremlin (Holzschuher and Peinl, 2013). Figure 8 shows a query written in a LINQ-like query syntax provided by the Neo4j C# client, which finds all members who graduated from GCU. It starts from a specific node in the database and matches member entities which have a GRADUATEDFROM relationship targeting this node. When the C# application runs this query is translated into a Cypher query and sent to the database.

```csharp
var query = client.Cypher
    .Start("school", myNode.Reference)
    .Match("member-[[:GRADUATEDFROM]->school"
    .Return<Node<Member>>("member");
```

Figure 6. Graph data example

Figure 7. C# class to define a relationship and the code to add a relationship of this type

Figure 8. A graph database query defined in C#
The data can be viewed interactively in the web-based Neo4j interface, as shown in Figure 9. This is a dynamic view and greyed nodes can be clicked to explore their connections.

![Figure 9. Visualising graph data in Neo4j](image)

**4. Conclusions and Recommendations**

This topic introduced students who had significant experience with relational databases to a new diverse data storage landscape. Within the context of the module in which it was delivered it was not possible to do any more than an introduction. A decision was taken to focus on only two of the main NoSQL database categories to show the significant differences in the nature of these. This meant, of course, that students had no exposure to key-value or wide-column stores. The datastores chosen have completely different query mechanisms, and there was not enough time to explore or practice these in any significant depth beyond simple illustrative examples. The module assessment included a practical group assignment where students had a free choice of data storage solutions for the specified scenario, one which might have lent itself to the use of a graph database. However, much of the focus of the module was on working with relational data from within .NET applications, and the students “played safe” with their storage designs.

Even if NoSQL is included as a larger component of a module, some of the same challenges present themselves. The aim of traditional relational database courses is essentially mastery of the relational model and SQL. In the case of NoSQL, however, students must learn about the data models of a diverse range of datastores and be able to adapt their knowledge to deal with different systems. Diversity in models and query mechanisms makes it more difficult to achieve depth without focusing on one system to the exclusion of others.

On the other hand, teaching NoSQL offers opportunities to develop a different set of skills in critically evaluating the needs of an application scenario and making an appropriate selection of datastore from the wide range of available systems, including, of course, relational databases. As Stanier (2012) observes, this will require a greater emphasis in teaching on requirements analysis, and the introduction of example scenarios which reflect the developments in the industry which have led to the NoSQL movement.

In order to make appropriate decisions students need a strong appreciation of the nature and capabilities of different categories of datastore. A broad understanding of the techniques for design and querying which they require can provide a starting point for deeper exploration to solve the problems of a specific scenario. Practical examples and hands-on experience are important in developing this appreciation. Experience of creating and viewing data in document stores and graph databases, and of developing and observing the results of simple map-reduce queries or graph queries such as those shown in this paper can strongly reinforce theoretical explanations. It is difficult, however, in lecture and lab examples to provide hands-on experience related to the issues of scale, and some thought and imagination needs to be given to this.
REFERENCES


Abstract
Data management technologies are changing rapidly and this presents a significant challenge for database teaching. There is a requirement to teach traditional relational database concepts and to ensure that students are equipped with the advanced skills expected by employers. There is also a requirement to prepare students to work with newer data models and NoSQL and to understand and be able to leverage concepts such as Big Data analytics. This paper discusses the experience of working with MongoDB and MapReduce and starting to work with Hadoop in undergraduate and postgraduate teaching at Staffordshire University. It is suggested that while the amount of time that can be given to newer technologies in the undergraduate curriculum is limited, this is a subject area which has the power to capture students’ imaginations and provides a good basis for undergraduate projects and Masters level dissertations.

Keywords
Big Data; NoSQL; Data Curriculum

1. INTRODUCTION
Industry data shows that although systems based on the relational data model still dominate the global database market, data management approaches and technologies designed for what is known as NoSQL and ‘Big Data’ are growing rapidly (Market Research Media, 2013, Gartner, 2014). Figures as to job vacancies are difficult to verify but it is clear that this is an expanding area and one where there is currently a skills shortage (Barnes, 2014). Since the introduction of the relational data model there have been a series of ‘New Waves’ designed to support more efficient and/or more flexible data handling. Over time, these newer approaches have tended either to be incorporated into relational implementations as with object-relational and XML functionality or have remained relatively peripheral to main stream data management (Java Developer 2004, Leavitt, n.d.). The Big Data era represents a different kind of challenge since the driver is not a technology looking for an application, as was arguably the case with object-oriented databases, but the data itself. Boyd and Crawford (2012) described Big Data as a socio-technical phenomenon which...
changes the definition of knowledge. Kambatla et al. (2014) note that data traffic grew 56-fold between 2002-2009 and that data creation and harvesting are both forecast to continue to expand rapidly. Big Data is most often defined by its characteristics (the 3 or 5 ‘V’s (Demchenko et al. 2013) but a more useful definition here is the shorthand version given by Madden (2012) – data that is too big, too fast or too hard for existing tools to process. Data now exists in volumes and in formats that were previously unknown or inaccessible and there has been a corresponding emergence of systems and tools developed to handle the data; this provides new challenges in data handling and data analytics (Kraska, 2013; Kambatla et al. 2014) and for the data curriculum. For the purposes of this paper, we understand the term ‘newer data management technologies’ to include NoSQL databases and the issues involved in storing, processing and analysing Big Data.

2. THE DATA CURRICULUM

The Quality Assurance Agency (QAA) 2007 Computing Benchmark statement (undergraduate) includes databases and related topics. Of 20 BSc (Hons) Computer Science and Computing Science degrees offered by UK universities for 2014/5 (randomly selected, from both Russell Group and post-92 institutions), all but one included some version of databases/data management in their outline online curriculum, sometimes specifying relational databases. There are, however, very few undergraduate specialist awards in database/data management as compared for example with Computer Security or Computer Networking. The UCAS undergraduate course search for 2014/5 found 7 courses that had database in the title but these were HNC/HND or foundation degrees. Two undergraduate Data Science courses were found, one at the University of Warwick, one University of Bedfordshire. There were a large number of Information Systems and Business Information undergraduate honours degrees and these awards typically include a substantial element of database work. At postgraduate level there was a wide range of specialist Data Science or equivalent masters degrees. This appears to be part of a global trend with new courses in data analytics being developed in the States, in Australia, India and Eire (Patel, 2013).

This data suggests a number of conclusions: traditional database topics are a core part of the general undergraduate computing curriculum but are primarily a service subject, in the same way that all computing students study maths and programming. A number of institutions offer specialist/advanced database/data management modules but these tend to be options, often in the final year. Information Systems and related specialist degrees include more data management/database topics than other undergraduate awards. At postgraduate level there is much more scope for specialisation. This has implications for the teaching of NoSQL/Big Data topics. Some areas such as data storage and processing may be covered as part of other specialist awards, as for example in the Staffordshire University Cloud Computing degree but as a generalisation, the database element in the undergraduate core curriculum tends to focus on established relational and object oriented database technologies. This reflects the experience at Staffordshire University where NoSQL/Big Data topics are taught as part of final year advanced database option modules and on postgraduate awards. We make the assumption that students come to these topics with a good understanding of relational and object relational design and implementation and that they have chosen database option modules because they have an interest in data management.
3. INTRODUCING BIG DATA & NOSQL TOPICS TO THE CURRICULUM

3.1 Related Work

Big Data as a topic is starting to appear in subject areas as different as Quality (Antony, 2013), Health (Sherestha 2014, and Engineering (Jelinek & Bergey 2013). In computing, there have been a number of studies on introducing Big Data and NoSQL topics to the undergraduate and postgraduate curriculum. The Portland approach is to introduce Big Data concepts at an early stage, teaching MapReduce, for example, at first year undergraduate level (Grossniklaus & Maier 2012). Anderson et al. (2014) review the experience of providing a specialist data science undergraduate degree at an American university and comment on the difficulty of sustaining a separate Data Science first degree. Three core data science modules are described in detail in the paper; all deal with Data Science core concepts and in each case Big Data issues are presented to the students side by side with relational and other topics. Sattar et al. (2013) discuss the experience of an undergraduate semester long web based project which used both Oracle 11g and MongoDB. The teaching schedule shows that of the 15 week course, 10 weeks were devoted to relational concepts and only 2 to NoSQL; the remaining sessions were devoted to web interface and project work. This reflects our own experience of teaching on an advanced database systems module in which the demands of the curriculum meant that only a small part of the teaching time could be devoted to newer data management approaches and these approaches were taught side by side with traditional database topics.

4. A STRATEGY FOR INTRODUCING BIG DATA & NOSQL TO THE CURRICULUM

The assumptions that underlie our strategy are that students come to these topics with a good grasp of relational concepts and that there is not enough curriculum space in a non-specialist undergraduate award to cover newer data management approaches in depth. The focus therefore is on introducing students to relevant concepts and providing them with the underpinning knowledge to support further study. As discussed in section 5, it is our experience that Big Data/NoSQL are topics which can capture students’ imaginations and that some students would wish to explore these topics further as part of final year projects/Masters dissertations. The approach taken is outlined in Figure 1 overleaf.

The following sections discuss our experience of working with Big Data and NoSQL.

4.1 Introduction to working with a NoSQL Datastore

Our initial experience of introducing NoSQL to the undergraduate curriculum was discussed in a paper presented to TLAD ’12 (Stanier, 2012) which explained our reasons for preferring MongoDB to other open source datastores. A key factor was the ease of download and the availability of supporting documentation and tutorials. MongoDB was locally loaded on to all the machines in our specialist lab and on to a number of machines in other labs for student access outside scheduled sessions. We found that the majority of students on final year undergraduate database modules downloaded MongoDB on to their machines and no students had problems with the download. We chose to work with the JavaScript shell as we felt this would be simultaneously familiar to students but very different from relational GUIs. Our evaluation with students showed that the use of MongoDB was welcomed and that they were enthusiastic about working with a newer data management approach.
Since the initial introduction of MongoDB, we have found that an increasing number of students come to the module with some prior experience of NoSQL, most usually through having worked with MongoDB on placement or having experimented with NoSQL themselves. We have not so far had any students claiming prior experience of any NoSQL database other than MongoDB but this may reflect the fact that MongoDB is now installed on most labs in the School of Computing and hence students have greater exposure to Mongo. This trend meant that in the current academic year (2013/4), we had two groups within the undergraduate cohort taking advanced database modules. The majority of students had no experience of NoSQL and needed introductory material but there was a smaller group of students who had already advanced beyond introductory level and were interested in implementation and performance issues. We expect that in future years, as there is more coverage of NoSQL in Level 5 (second year) modules, students on final year modules will already have a basic understanding of NoSQL and it will be possible to introduce more advanced topics for the whole cohort.

When we first introduced NoSQL in 2011/2, MongoDB was initially used only in one final year undergraduate option module and the focus was on introducing students to the schema later (schemaless) element and design issues. The assessment was based on a data management case study in which some aspects of the case study lent themselves to a relational solution and some to a NoSQL solution. Students were required to develop an enterprise solution in either Oracle or SQL Server and to create and query a customer comments blog in MongoDB. For the MongoDB element, students created collections to hold documents, queried and updated documents and evaluated the differences between relational and NoSQL design. We have since extended the use of NoSQL into other modules and have started to look at a wider range of issues. As discussed in 3.1, the literature suggests that most institutions teach relational and non relational concepts side by side; we found this helpful as well understood relational concepts provided depth and context and gave students a point of reference. As an example, the syllabus for a Level 5 database security module includes the ACID protocol; this year we discussed BASE (Basically Available, Soft state
Eventual consistency) and the CAP Theorem (Consistency, Availability, Partition Tolerance) in a relational and non-relational context, illustrated with reference to MongoDB and developer blogs. This allowed students to relate concepts to real-world data management problems. One student picked ACID/BASE for the research paper which was part of the module assessment. We have recently had a revalidation which has resulted in moving some relational topics into Level 5 modules, leaving space in final year modules for newer data management approaches.

We encountered two problems with the MongoDB installation. A minor problem was that the Data folder sometimes needed to be purged manually (worth investigating if there are unexplained problems with access). A more significant issue was that a student who wished to develop a project which focused on sharding, encountered network permission issues and had to be given extended privileges to complete the project. We are currently discussing installing MongoDB in a virtual environment to prevent permission problems in the future.

4.2 Data Sets
We have worked with large data volumes for some time as students are required to create million+ data sets for relational optimisation exercises, using either free data generators or tutor-provided data generation procedures. This is not, however, Big Data and working with data sets large enough to be described as Big Data has so far been problematic. The issue is not the availability of data sets; we have for example used the Stanford Gowalla data in a number of projects. The difficulty is load time, particularly for students working with unfamiliar functions such as mongoimport. The alternative approach is to preload data. For teaching purposes we wanted students to experience the differences between data manipulation in relational and NoSQL environments and for this reason we chose to work with small JSON datasets which could easily be created, manipulated and most importantly understood by students. As discussed in 4.3, we also propose to provide preloaded data sets for more advanced work.

4.3 MapReduce and Hadoop
Teaching MapReduce, we were limited to the MongoDB platform as we did not have an Hadoop installation for classroom use. This had some drawbacks since although MongoDB provides support for this approach, it is not optimised for MapReduce\(^1\) and recommends the MongoDB aggregation framework instead. As we had previously decided to allow students to create their own data sets, the data used was trivial for a MapReduce operation. The advantage was that students were working with a now familiar interface and the small scale of the data made it easier for them to track what was happening. A basic tutorial was used which enabled students to map the data to Key Value pairs and then reduce the data. Comparing this with a group of students who had run the traditional MapReduce word count program, the small data set approach seemed to support a better understanding of concepts but did not give the students a sense of scale. For the next academic year, we plan to have the Hadoop ecosystem available for classroom use so that students can be introduced to MapReduce using a trivial data set and progress to working with a more realistic data set.

We have been investigating the options for Apache Hadoop both to give students a more realistic Big Data experience – we would like to introduce students to Pig and Hive and allow them to compare results with relational data manipulation – and to ensure that students who wish to use

---
\(^1\) Current release; this will change in future releases
Hadoop for projects/dissertations have easy access to software. We are currently investigating 3 options. Hadoop has been installed on OS-X 10.5. We are not proposing to use this installation for classroom teaching as our data management modules use a wide range of software which will not be installed on the Macs. The OS-X version will be available for project/dissertation use. We considered a dual boot solution but this was rejected for performance and partly for security reasons. The remaining option was to work in a virtual environment. One of the authors of this paper has been working with the HortonWorks Sandbox. This is an open source download which provides a simple interface to the Hadoop ecosystems – Hive and Pig for example are preinstalled. The system requires a virtual environment but the download is straightforward. Hortonworks provides a wide range of tutorials and support material. There are some limitations, for example on the data volumes uploaded but the author worked with data sets which although not meeting the definition of Big Data were large enough to provide a good basis for exploring the functionality provided. We are now testing the sandbox for classroom use.

5. Student Work
As discussed in section 4, NoSQL and Big Data are topics which are capable of capturing students’ imaginations and which students are keen to explore further. A number of students have developed final year projects and masters’ dissertations in this area, producing high quality work and we expect this to continue. To illustrate the work produced, we discuss two example projects, one undergraduate, one masters. The undergraduate project involved the creation of a performance analysis dashboard for use with MongoDB. One of the disadvantages of NoSQL databases, as compared with relational DBMS, is the relative lack of DBA tools. The project created a tool to monitor and analyse performance, integrating with 3rd party applications and using a range of programming languages and environments. A dedicated test environment was created which involved configuring multiple MongoDB servers. The Masters project was Hadoop based and investigated the causes of failure in the Name Node and the recovery process. This required the student to install the Hadoop ecosystem and develop an understanding of the Hadoop architecture as a preliminary to carrying out the project. Both projects were demanding and in both cases the installation/testing element could almost have formed a project in its own right; the challenge of working with a new and unfamiliar technology and the students’ enthusiasm for the topic meant that they set themselves ambitious goals. One of our reasons for moving to use MongoDB in a virtual environment, and installing the Hadoop ecosystem on a number of machines is to support future projects.

6. Conclusion
Computing education is used to seeing topics come in and out of favour; some develop into permanent additions to the curriculum and others are relegated to history. Stonebraker (2013) described ‘Big Data’ as the ‘Buzzword du jour’ but the same article outlined the challenges that Big Data represents and the way it is changing what we can do with data. The tools that we are currently use are already evolving as more functionality is added to NoSQL databases and alternatives strategies for working with Big Data start to develop. Teaching strategies for NoSQL and Big Data will continue to be a work in progress but our view is that these topics are now a core part of the database curriculum. The final comment comes from Michael Rappa of the Institute for Advanced Analytics: “Big data isn’t a new speciality or suite of tools we have to train people into, as much as it’s a new organizational reality that everyone will need to adjust to occupationally” (Rappa, 2013)
REFERENCES

Anderson P., Bowring J., McCauley R., Pothering G. & Starrr C. (2014) An Undergraduate Degree in Data Science: Curriculum and a Decade of Implementation Experience SIGCSE’14, March 5–8, 2014, Atlanta, Georgia, USA.


Leavitt Communications n.d. Whatever Happened to Object Oriented Databases http://www.leavcom.com/db_08_00.htm accessed 17 June 2014

Madden S. (2012) From Databases to Big Data IEEE Internet Computing pp 4-6


Stanier C. (2012) Introducing NoSQL into the Database Curriculum TLAD 2012 pp. 61 -72

Practice makes perfect: Improving success rates in a Database Design module

Peter Stordy
Information School
University of Sheffield
UK
peter.stordy@sheffield.ac.uk
http://www.sheffield.ac.uk/is/staff/stordy

Paul Clough
Information School
University of Sheffield
UK
p.d.clough@sheffield.ac.uk
http://www.sheffield.ac.uk/is/staff/clough

Abstract
The Information School, University of Sheffield attracts postgraduate and undergraduate students from a diverse range of backgrounds and with a wide spread of computer literacy levels. Many take the module ‘Database Design’ because they perceive it will increase their employability and it is central to their degree. However, failure rates have been high and the distributions of grades achieved ‘negative skewed’. A new module team set about identifying potential causes and developing solutions. A series of Oracle database design and implementation laboratories was perceived to be pivotal. These were failing to increase students’ understanding of relational databases before they embarked upon the coursework assignments and were not reinforcing the concepts introduced in the lectures. The solution was radical: attendance at the laboratories was made entirely optional. The demonstrations were replaced with a series of four online tutorials supported by the university’s virtual learning environment, bespoke YouTube videos, informative slide shows, carefully chosen Web links and tailored tasks. The completion of the tasks during the first few weeks of the semester became a mandatory aspect of the module’s assessment. By making these changes, the number of students failing the module significantly reduced, the distributions of grades achieved became ‘positively skewed’, student satisfaction with the module rose and fewer students sought extra tuition from the module team.

Keywords
blended learning, relational databases, pedagogy, online, computer literacy

1. Introduction
At the first ‘Teaching, Learning and Assessment in Databases’ (TLAD) conference, Eaglestone and Nunes (2003) reflected upon the experiences of delivering a Database Design module in the Department of Information Studies (now the Information School) at the University of Sheffield. Their innovative approach attempted to address the tensions implicit in teaching database design to a technologically mixed body of undergraduate and postgraduate students in a research-led, non-computer science department. In particular, they described how they walked, “the tightrope between superficiality and technological impenetrability”. Eleven years on, many of the challenges and solutions remain. Although students’ computer literacy is generally better than it was in 2003, they still present wide variations of experience. However, the more computer literate students
appear less tolerant of technical instruction, whilst the less computer literate students find it does not meet their needs. Consequently, our response has evolved. Whilst the module retains its problem-based and “learning by doing” (Dewey, 1916) pedagogy, it now utilises current online technologies. Central, are a sequence of online tutorials that teach the basics of relational databases and a series of assessed practical tasks. This paper describes the challenges and our pedagogy in more detail.

2. BACKGROUND
During the academic year 2013-14, the Information School, University of Sheffield had 191 postgraduate and 116 undergraduate full time, part-time and distance learning students. 40% of the students were classified as ‘Home’, 57% ‘Overseas’ and 3% ‘European’. Whilst the majority of the overseas students came from mainland China (69%), the Information School attracts students from all over the world, including Nigeria, India, Mexico, South Korea, Saudi Arabia and Brunei. Despite the technical nature of our degrees, only 31% of the postgraduate students have a Computer Science related undergraduate degree and 9% of the undergraduates have an ICT-related Level 3 qualification. The numbers of male and female students is typically equal. 26% of the students are studying dual degrees, predominantly with the University’s Management School.

Over half of the Information School students study Information Management (MSc, BSc or BA) at undergraduate or postgraduate level. To varying degrees, Information Management concepts permeate the other students’ degrees, which include Information Systems, Librarianship and now Data Science (Information School, 2014). Information Management was defined by a former Information School Head of Department as, “the effective management of the information resources (internal and external) of an organization through the proper application of information technology” (Wilson, 1989:204). More recently, conceptions of Information Management have become less outcome-based and more of a cyclic process. For example, Chun Wei Choo found that “intelligent organizations” effectively managed information as a series of 5 basic steps: “identification of information needs”, “information acquisition”, “information organization and storage”, “information distribution” and “information use” (Choo, 1998:24).

Notwithstanding differences between data and information (for example Ackoff, 1989) Database Management Systems (DBMS) are a key information technology that facilitates the acquisition, organisation, storage and distribution of information within many organisations. Topics such as Data Management and Data Analytics are also emerging as key aspects of the ‘data revolution’ whereby effective utilisation of data is empowering organisations to gain better business insight and understanding. Given the central role of DBMS within Information Management, the Information School’s ‘Database Design’ module is a relatively popular option for many of its students who perceive it will increase their employability. For other students, the module is ‘core’ to their degree and mandatory. For a variety of reasons outside the control of the staff teaching the module, the number of students enrolling on this module has fluctuated over the last four years between 36 and 100 students.

The aims of the Database Design module would not look out-of-place thirty years ago, although the content has been updated to reflect the recent focus on topics, such as Data Warehousing and NoSQL. Primarily, the module addresses the role of databases within organisations, provides an appreciation of the rigorous methods that are needed to design, develop and maintain database systems, and highlights current and emerging database technologies. These aims align with QAA
(2014) Librarianship and Information Management subject benchmarks. In particular, students are expected to apply database design techniques to problems to produce conceptual and logical data models. Finally, they are expected to implement a logical data model, and any necessary physical design, using an industry standard relational database and to populate it with test data and query it.

Unsurprisingly, the assessment for this module is aligned with the module's aims and pedagogy. Essentially, students are given a detailed case study related to the Information School's teaching and assessment portfolio. They are required to apply conceptual database design techniques to produce a conceptual data model in the form of UML Class diagrams. Logical database design techniques must also be applied to transform this conceptual data model into a logical relational data structure in form of a pseudo Data Definition Language. Finally, the students are required to conduct a rudimentary physical design in order to implement this logical relational data structure as an Oracle database using Oracle's Apex Web interface, populate the implementation with a given data and produce a set of views to address potential users' information needs.

3. **Issues**

Historically, failure rates on this Database Design module have been relatively high and the distributions of grades achieved have been negatively skewed. To our surprise, it was not necessarily the students with weaker computer literacy who were perform poorly, but often those who would be considered more computer literate; for example, the Computer Science students taking the module. A new module team in 2010 set about identifying the causes and developing potential solutions.

As identified back in 2003 (Eaglestone and Nunes, 2003) students taking the Database Design module come from a diverse range of background and have a wide range of computer literacy levels. Too many students had inflated perceptions of their own computer-related abilities and some poor time management issues. One consequence was that many students underestimated the time it would take to implement the coursework relational database. They struggled to finish it before the deadline. The original practical laboratories that taught students relational database design and implementation skills failed to engage the students, were poorly attended and were plagued by issues. For example, the layout and size of the laboratory meant students struggled to follow demonstrations on the screen. With such a diverse group of students, it was difficult to gauge the pace and level of detail required. More often than not, the demonstrations were laborious for some, too hasty for others and only about right for a handful of students. Furthermore, this lack of engagement or inability to follow the demonstrations meant many students lacked the deeper understandings of relational databases that “learning by doing” or a “practice makes perfect” learning philosophy promotes. That is, the process of implementing a relational database leads students to have a better understanding of relational databases. Practice and increased understanding are complementary. The module team felt this pedagogic dependency was not fully exploited in the module.

4. **Our Solution**

The design/implementation laboratories were recognised as key to addressing the issues identified in the previous section. Our strategy was to redesign the laboratories, reposition them within the module’s teaching sequence and introduce a third assessment component to the module.
The laboratory demonstrations were replaced with a series of four online tutorials supported by the university’s virtual learning environment (BlackBoard), bespoke YouTube videos (Stordy, 2014), informative slide shows, carefully chosen Web links and tailored tasks:

**Tutorial 1:**
- Oracle Login and changing password;
- Create HR database tables;
- Creating primary keys;
- Creating foreign keys;
- Introducing SQL.

**Tutorial 2:**
- Testing table constraints;
- Creating new tables;
- Implementing Oracle SQL.

**Tutorial 3:**
- Creating Oracle views;
- Advanced SQL.

**Tutorial 4:**
- Mock coursework;
- More advanced SQL examples;
- Check constraints and examples.

The online tutorials and tasks were released each week to mimic the weekly laboratory sessions. The completion of the tasks in the first weeks of the module became a mandatory aspect of the module’s assessment. The laboratory sessions continued, but were relabelled ‘drop-in’ sessions. Attendance was made entirely optional. Rather than students watching a lecturer demonstrate using the Oracle Apex interface, the laboratories supported the online tutorials. Students could choose whether to complete the weekly online tutorials and tasks during the laboratories, supported by a member of staff, or do them at another time on their own. Some students attended simply to ask questions. They valued the opportunity to seek immediate support when needed. That is, the laboratories became more student-centred and better able to adapt to students’ preferred learning style. Students could repeat, stop and start the tutorials as many times as they needed. By positioning the online tutorials and tasks at the beginning of the module, the module team envisaged that students’ deeper understanding of relational databases, entity integrity and referential integrity, would support their attempts to conduct conceptual and logical design within the main coursework, and curtail the disproportionate amount of students who submitted late or produced work that lacked understanding.
For reasons which are beyond the scope of this short paper, our students tend to be very assessment-focussed. Unless a learning activity is directly associated with passing the module or gaining a good grade, they are unlikely to engage with it. Hence, completion of the four online tutorials and tasks attracted 5% of the module’s grade. Students were required to pass this assessment component to pass the module overall. Without this incentive, it is unlikely that the students would engage with the online activities. Furthermore, we insisted that students completed all the tutorials and tasks. Rather than assess their work per se, we simply checked that all the work had been done. Mechanisms in our virtual learning environment facilitated this. Simple SQL scripts were also run to check students’ databases were correctly populated. The administrative overhead of this aspect of the module was minimal. Students either received the full 5% or zero.

The remainder of the module’s teaching and assessment evolved from Eaglestone and Nunes (2003) original design. A series of lectures covered the remaining aims of the module and dovetailed into the weekly online tutorials:

**Introductory lecture strand:**
- Introduction to Databases and the Course;
- Database Planning, Design and Administration.

**Relational database lecture strand:**
- Conceptual Database Design;
- Relational Databases and SQL;
- Logical Database Design;
- Physical Database Design;
- Database Normalisation.

**Beyond relational databases:**
- Spatial and Temporal Databases;
- Data Management and Organisational Intelligence;
- Non-Relational Databases – NoSQL.

The other components of the assessment remained mostly unaltered. Students are still required to produce conceptual and logical models of a given case study and set of requirements. However, rather than students producing a physical design of their own logical model and implementing it in Oracle, students were given an exemplary logical model to work with. Whilst not as satisfying as conducting the entire database design process, it ensured students all had the same opportunity to demonstrate their understanding of physical design and implementation. Furthermore, it simplified the assessment process as simple SQL scripts could be run to interrogate their databases and highlight any errors.
5. **CONCLUSION**

In 2007, almost 20% of the Database Design students failed the module and the mean overall grade was 58 (N=56). In the first year that we ran this new assessment structure (2011), the failure rate had dropped to 14% and the mean overall grade had increased to 70 (N=85). However, these statistics only tell part of the story. Anecdotal and student module evaluation comments indicate that student satisfaction had increased. Fewer students sought extra tuition from the module team and fewer students submitted coursework late. The module team would argue that the mandatory online tutorials and tasks at the beginning of the module have contributed to these improvements. Overall, we are confident that the changes have benefitted students’ understanding of relational databases and their module attainment.

However, the module team recognise that students are still failing this module and some students are still not grasping basic relational database design concepts. Recent changes to the Information School’s degree portfolio mean that we have scope to reintroduce aspects of Eaglestone and Nunes (2003) original design. For example, the main coursework could become more collaborative and small group tutor-guided. The online environment which accommodates the tutorials could also facilitate more collaborative features like tutor moderated discussion boards. In other words, the module has the potential to become more ‘blended’, incorporating further opportunities for online discussion, adaptation, interaction and reflection (Laurillard, 2001).

**REFERENCES**


Contextualised Problem-based Approach for Teaching Undergraduate Database Module

James Xue
University of Northampton
St George’s Avenue
Northamptonshire
james.xue@northampton.ac.uk
www.computing.northampton.ac.uk/~james

Abstract
In this paper, a new approach has been used in teaching the second year undergraduate database module. The approach is a combination of contextualisation, problem-based approach, group work and continuous formative assessment. The contextualisation ensures the visibility of teaching/learning activities so that students are aware of the values of activities and how they can fit into a big picture. Problem-based approach gives the students tasks/problems to solve before the relevant lecture takes place, hence can better develop effective reasoning processes, independently learning skills and improve motivation and engagement. Group work is regularly used due to the diversity of student backgrounds and level of prior knowledge of certain topics. By having group work, students can learn from each other and easily clarify confusions among themselves before approaching the lecturer. This gives the lecture more time focusing on common issues. Formative assessment has also been used to support teaching/learning activities and to reinforce their understanding. The work in this paper has been evaluated via an end-of-year online module survey. The results show good effectiveness of the new approach, although there are still spaces for improvement.

Keywords
Contextualisation, problem-based Learning, peer teaching and assessment, formative assessment

1. INTRODUCTION
1.1 Module and Student Backgrounds
The second year undergraduate database module is a core module for the most of the computing courses at the University of Northampton. This module is the pre-requisite of the final year database module and is one of the co-requisites of the second year group project module, which itself is the pre-requisite of the final year dissertation module. Therefore, student performance on the second year database module has direct impact on other modules and likely their final year degree classification. This module develops an understanding of the process in building database applications, with particular focus on the underlying technologies, which make database application development possible and efficient.

Students on the module usually have mixed backgrounds. The majority are home and EU students; there are also some international students who usually have special needs with regard to language
provision and social support. Even for the domestic students, there is a big variation between HND and BSc computing students in terms of entry standards and prior knowledge of certain topics (e.g., from their A level studies or employment). Those students who have prior knowledge are likely to get bored or be absent when being taught basic database concepts, whereas others might be struggling in understanding them. Moreover, there are also different age groups among those students, some of which are mature students who tend to have better motivation, hence better learning attitude. Furthermore, students enrolled on this module are from different courses, perception of importance of the module is different. There is an impression among the computing students that the database module is more relevant to some courses than others, and some students are even not sure why the module is needed for their course. In such a diverse environment, actively engaging students becomes a very challenging task.

1.2 Introduction to Contextualisation and Problem-based Learning

Contextualised teaching is to put individual topics to be taught into a meaningful and real context rather than to treat them as isolated items. Contextualisation allows the learners to see the big picture and how individual topics fit into the big picture and the relationships between them. This will help the learners easily understand the topics being taught, and to quickly recall them during revision period. During university studies, all teaching and learning activities set for students should be seen as having value and as readily performable. Students should be required to build on what they already know, to be relevantly active, to receive formative feedback and to be engaged in monitoring and reflecting on their own learning (Biggs et al 2011). Teaching and learning activities need to be aligned to the intended learning outcomes that are to be facilitated. The alignment should take place in all activities, including content design and delivery, creation of formative assessment and assignments and examinations, marking criteria, feedback, etc. Contextualisation is to make the learning visible, so that students at all learning stages have clear idea about why they are learning the topics and what learning outcomes the activities will lead to.

Contextualisation for the undergraduate database module was not done properly in the past as some students were not sure about the values of certain topics and activities and could not link the topics together. For example, some students often asked the lecturer about the relevance and importance of entity relationship modelling; other questions asked were about the relationships between certain topics; in another word, they could not see the big picture. It can be predicted that when students have such confusion, they could easily get frustrated.

Traditional teaching practice follows the fill-up-the-tanks model of knowledge acquisition by teaching the disciplines first, independent of one another, and armed with all that declarative knowledge and some professionally relevant but atheoretically taught skills (Biggs et al 2011). The problem of traditional model is the misalignment of intended learning outcomes, teaching and assessments. Problem-based learning (PBL) gives students with functioning knowledge so that their induction into real-life professional practice is much quicker. PBL reflects the way people learn in real life; they simply get on with solving the problems life puts before them with whatever resources are to hand (Biggs et al 2011). (Savin-Baden 2000) argues that PBL is often confused with problem-solving learning, which simply means setting problems for students to solve after they have been taught conventionally and then discuss them later. According to (Boud 1985), in PBL the starting point for leaning should be a problem, query or puzzle that the learner wishes to solve.
Formative assessment is powerful teaching/learning activity that uses error detection as the basis for error correction (Biggs et al 2011). It is an ungraded assessment and used to assist on-going learning. When formative assessment is used, it is very likely that students will feel free to admit their errors and learn from them. In contrast, summative assessment is mainly used for grading students and the grades are final. Students are unwilling to admit their mistakes, as they fear the assessment outcome. Error is no longer there to instruct, as in formative assessment; error now results in punishment (Biggs et al 2011). Formative assessment can be in various forms such as questions and answers sessions, quiz, short assignment, ungraded class test, peer assessment, etc. Formative assessment and feedback should be used to empower students as self-regulated learners; more recognition should be given to the role of feedback on learners’ motivational beliefs and self-esteem (Nicol et al 2006).

In this paper, PBL will be used in combination with contextualisation, group work and formative assessment to achieved intended goals of improved motivation, better engagement and ultimately good learning outcomes.

2. INQUIRY-BASED PROJECT

2.1 Project Rationale

The teaching and learning practice for the module of study in the past followed the traditional lecture plus practical session model. Most of the lecture notes were developed from scratch, and several textbooks were used. The previous materials for the lab sessions were not systematic, and were replaced by a well-designed lab guide, which uses examples from a different business scenario. Due to lack of consistent information (examples used from different books), it was difficult for the students to see the big picture of various topics covered in lectures and labs, hence difficult to see the interconnection between them. Because examples were from multiple resources, it was very difficult for students to work on some example and use it for next practical sessions. Instead, students had to implement several examples in order to verify their understanding of certain topics. This discouraged some students from trying out the examples. Contextualisation helps put individual topics into a big picture, therefore the interconnections between the topics can be easily seen by the students, and the usefulness of the topics can be realised.

In each of the practical sessions, the main task was to follow the step-by-step instructions in the lab guide. Most of the students simply followed the instructions to complete the tasks without much thinking. This kind of spoon-feeding practice prevented metacognitive activities and affected functioning intended learning outcomes (Biggs et al 2011). When students were asked to solve problems independently, they struggled to come out with solutions. PBL is one of the active teaching/learning methods that can narrow the gap between students doing higher order cognitive activities, as it requires students to question, to speculate, and to generate solutions. (Biggs et al 2011) classifies problem-based learning as a good teaching method as it gets most students to use the level of cognitive process needed to achieve the intended outcomes that the more academic students use spontaneously. Contextualised problem-based approach helps students learn academic knowledge and develop professional skills during the process of solving real-world problems, which will benefit their future studies and employment.
Due to the big diversity of the student backgrounds and prior knowledge of the certain topics, when teaching some basic database concepts, it was very difficult to get the same level of engagement from all students. Also, some students thought they understood some topics that they learnt before, but actually they were not able to do the related tasks due to different level of difficulty. In PBL, the students are given problems to solve before the lecture takes place, hence likely pay more attention to what confused them; for those who have prior knowledge about the topics, this approach tests their real understanding.

The group discussion has been used for two main reasons: i) based on the author’s past experience, not all students like asking the lecturer questions, particularly in a big class; small groups encourage them to express themselves, therefore enable better engagement; ii) in a lab session, the tutor has limited time on each of the students. Group discussion gives students a sense of involvement and helps them correct most of the mistakes; therefore the tutor could make better use of the limited lab time. This approach results in good efficiency and productivity. Peer teaching and assessment provides a structure and framework for discussions about quality of work, and helps student to become critical about their own work and the discipline-related body of knowledge (Hinett 2002). During the discussion, better students can help others clarify confusion, making themselves feel helpful; teaching others also makes them understand the topics better. With regular formative assessment, students are assessed on how well they meet preset criteria, where they were before, where they are now and what they need to get a high grade. With proper guidance from the lecturer and discussion with their classmates, the learners will eventually be able to solve the problems themselves, and their confidence will be built, and expectation of success will follow. When the learners see the value of what they are learning and the possibility of success, according to the expectancy-value theory (Feather 1982) they will automatically have intrinsic motivation, which drives deep learning and the best academic work (Biggs et al 2011).

2.2 Aims and Objectives

The aims of the work in this paper are to use contextualised problem-based learning to increase motivation, improve engagement, promote metacognitive activities in learning this module, and therefore to achieve better learning outcomes. The main objectives of this paper are as below:

*Increase motivation* - contextualisation makes students believe what they are going to learn is useful; problem-based learning helps them develop useful skills to solve the real-world problems. When students know they can be successful, their motivation will automatically follow.

*Improve student engagement and encourage metacognition* - peer teaching and assessment improves student engagement; PBL encourages more metacognitive activities.

*Change students’ perspective on what they have learnt* - contextualised problem-based learning makes students feel they have learnt something useful from their own perspective. It reflects the phenomeno-graphic approach (Prosser et al 1999), which states that it is important to change the learners’ perspective on how they see the world and how the learners represent knowledge. When the learners’ perspective changes, it will likely lead them to higher order levels of understanding.
The new approach is part of the transformative reflection practice. The multi-stage process of reflect \(\rightarrow\)plan \(\rightarrow\)apply \(\rightarrow\)evaluate will always be applied for continuous improvement in future teaching/learning activities.

![Diagram](image)

**Figure 1**

3. **PROJECT IMPLEMENTATION**

3.1 **Contextualisation**
In the module specification, there are four main topics that need to be taught: entity relationship (ER) modelling, data normalisation, creation of databases, SQL, etc. Building a relational database for a given business scenario and answer some important business queries is a chain of process. From Figure 1, it can be seen that when a scenario is given, a database design can be built using ER modelling, data normalisation, or the combination of them. The model can then be used for database creation (either manually or using existing software tools). The creation of relational database and data input can be done using Data Definition Language (DDL). Finally, the database can be queried based on business requirements using Data Manipulation Language (DML). All these steps are essential, and they are very closely related. In the past, the topics were taught separately, the relationships between them were not emphasised, although significant amount of time was spent, quite a few students were still struggling to understand them and even not sure why they need to learn them. One typical example was that several students asked me why they needed to learn ER modelling. The author was told that they thought the ER diagrams (ERD) were just academic practice, which was of little use in solving real problems. To help them understand the relationship, at the beginning of each of the topics, some time was spent telling how each of them fits in the big picture and how important they are. When the importance of the topics is addressed, motivation will follow.
3.2 Problem-based Learning
Students were given problems (in the form of gobbets) usually before relevant lectures took place. Students had one or two days to read the scenarios and to think about the questions. According to (Johnstone 1976), concentration during a one-hour lecture is only about ten to fifteen minutes; if this short period is used to focus on something confusing, it will be more efficient. With the questions in mind, students usually paid more attention in the following lecture to what confused them. They were given plenty of time to ask questions to clarify the confusion. The lecture gave students necessary knowledge to solve the problem and in that sense became a facilitating session. For each of the problems/topics, a list of common issues from previous years was also given, so that students can learn from others’ mistakes. The common mistakes were also used on blackboard as comment repository for assignment marking and student feedback.

3.3 Group Work
The problem based approach was also used in group work, which was regularly used for two reasons: 1) as mentioned in the Section 1, students on this module have very diverse backgrounds and different prior knowledge of certain topics, therefore, group work seems ideal in such a situation; 2) some of the topics in this module such as ER modelling and data normalisation are, especially at the beginning, confusing, the author has encountered all sorts of mistakes from students. Group work took place during practical sessions, students were asked to form small groups of two to three students randomly as suggested in (Yamane 2006) to avoid gossip or discussion of off-tasks. They were asked to perform the tasks first on their own, when finished, they needed to compare the answers and convince the group members. After short group discussions, most groups could come out with a good answer; occasionally, all groups made a common mistake that was usually due to insufficient explanation of the topic in the lecture. In that case, more time was spent on the particular topic.

3.4 Formative Assessment
Formative assessment was regularly conducted by peer assessment and tutor assessment. One of the main advantages of formative assessment is that students are not afraid of admitting mistakes in front of the lecturer or a big class or in a work to be graded. Peer assessment is very informative, based on the observation, students were usually not afraid of admitting mistakes. In the practical lab session, the tutor usually walked around and checked whether the students understood the topics by asking them questions. Doing the informal conversation, most of the students tended to ask questions, which otherwise might not be asked in a big class. Continuous formative assessment consolidated students’ understanding of different topics and improved their confidence.

4. PROJECT EVALUATION
4.1 The Student Survey
The new approach used for teaching the module was evaluated at the end of the academic year via online module survey. According to experience in the past, the more questions in the survey, the less responses students made. Since the survey was done after all assessments, students were less likely to respond to the survey actively. To encourage more responses, there were only ten questions in the survey. All questions except the last one were multiple choice questions (MCQ) with scale 1 to 5, representing strongly agree to strongly disagree, respectively. The questions were carefully designed to get students’ feedback on different aspects of the module in a way similar to those used in national student survey (NSS).
4.2 The Results

The survey form was active for three days, and during the time twenty-seven out of sixty students responded, which is considered a good sample. In this section, the evaluation results are analysed to see the effectiveness of the new approach. The response to question 1 (refer to Figure 2) shows that about 60% of the students considered the module is intellectually challenging, slightly less than 20% of them did not share the same impression. This response reflects the diverse student backgrounds mentioned in Section 1.1. The second question is about teaching style. From Figure 3 it can be seen that 63% of the students believed the traditional style (lecture plus lab session) was suitable for this module whereas 29% of them had opposite view. The figure suggests some changes in teaching style are required and it is one of the reasons of this paper.

![Figure 2. Responses to survey Q1](image1)

![Figure 3. Response to survey Q2](image2)

As mentioned earlier, the problem-based approach together with group work has been introduced for this module, particular in the lab session. Figure 6 shows that 66% of the students believed that the problems given in the lab sessions were challenging, whereas 19% of them needed more challenging tasks. Due to the diverse backgrounds this is expected, and will be solved by adding more challenging tasks for more competent students. From Figure 4 and 8, it can be seen that some students were quite happy (60%) about the group discussion to solve the problems among themselves in lab, however, the group work resulted in less involvement from the lecturer, which around 30% of the students seek direct help from. These two figures verify each other, and the textual feedback in Figure 11 confirms the conclusion. In the future, perhaps the lecturer should participate more in-group discussion to give students perception of involvement.

![Figure 4. Responses to survey Q3](image3)

![Figure 5. Responses to survey Q4](image4)
Question 6 of the survey asks students if they feel the knowledge from the facilitating lectures is enough to solve the problems. 66% of them felt very positive (refer to Figure 7), but students (26%) felt negatively. The reason for this is unknown, it could be due to insufficient explanation of the topics during lecturing, or some students were poor applying the theory to solve practical problems, therefore more problem solving examples should be given. From the response of the last survey question (refer to Figure 11), the latter is more likely to the case.

![Figure 6. Responses to survey Q5](image)

![Figure 7. Responses to survey Q6](image)

In this module, a comprehensive MySQL lab guide has been used to improve the students’ practical skills, particularly the Structured Query Language (SQL) skills. As the instructions are very detailed, therefore it tends to be more independent work. The lecturer usually checked the students’ progress by completion of the tasks. Figure 7 shows that 30% of the students did not think the guide was very helpful. It might be due to the guide is so detailed (step-by-step), completion of the tasks in the guide did not really help the students solve difficult problems independently. The reason remains to be investigated.

![Figure 8. Responses to survey Q7](image)

![Figure 9. Responses to survey Q8](image)

Questions 8 and 9 are about assignment marking criteria and feedback. The majority (over 60%) of the students (Figures 9 and 10) were happy, but some students (less than 20%) were still not very happy. This needs to be improved in the future.

Overall, for all questions, about 60% of them agreed or strongly agreed on the current practice and about 20% disagreed and the rest had no strong opinions. Since the survey was conducted after the examination (which has high weight (60%) of the module assessment), which, the author believe, had high impact on the survey results. To eliminate such an effect, it might be a good idea to choose more suitable time to conduct surveys in the future.
5. **Discussion**
During the evaluation, the author noticed that although the problem-based learning worked well for some students, it did not work well for others. Barrows in (Barrows 1986) argues that in order for PBL to work well, two things need to be considered: 1) the degree to which the problem is structured. In another word, the level of difficulty should be decided carefully and all the information needed to solve the problem should be provided. If the problem are too difficult or there is insufficient information available (supplied by the tutor or on other resources), it will demotivate the students from trying it; 2) the extent of tutor’s direction towards the solution. The tutor needs to have the right level of involvement/direction, so students will not be leave to solve the problems on their own. The response to the last survey question suggests that the direction from the tutor was insufficient, and this should be improved in the future.

| Figure 10. Responses to survey Q9 | Figure 11. Responses to survey Q10 |

It has also been observed that ownership is essential for PBL to work well. The ownership comes from strong motivation. As described earlier in this article, contextualisation ensures visibility of the values of all teaching/learning activities. According to the motivation theory (Biggs et al 2011), when students see the values of the activities, and can expect success when engaging the learning tasks, motivation will follow. Therefore, contextualisation and PBL is a good combination for good teaching/learning.

6. **Conclusions**
The paper applies a new approach for teaching/learning second year database module. This module is a core module of most of the undergraduate computing courses and serves as the pre-requisite or co-requisite to other modules; therefore performance of this module has big impact on the overall studies. Students on this module usually have very diverse backgrounds in terms of possession of prior knowledge, age groups, social and cultural differences, etc. These factors make it difficult to have a right balance for all students; as a result engagement was usually poor. To promote good engagement and achieve indented learning outcomes, a combination of contextualisation, PBL, group work and regular formative assessment has been used in teaching/learning the module. Contextualisation ensures visibility of values of the tasks, hence can improve the motivation, which is the key success factor for PBL. Group work allows students to learn from each other and clarify confusions among themselves before approaching the lecturer. Good students can learn better by helping the peers and will not be bored. It also spares the lecturer’s time to focus on common issues. Formative assessment has also been regularly used to reinforce understanding of the topics.

The work in this paper has been evaluated via an online module survey. There are some positive results (about 60% of the students agreed or strongly agreed on the new approach), however,
about 20% of the students who responded negatively to the new approach, therefore there are still some spaces for improvement. The new approach is part of the transformative reflection practice. The multi-stage process of \textit{reflect} $\rightarrow$ \textit{plan} $\rightarrow$ \textit{apply} $\rightarrow$ \textit{evaluate} will always be applied for continuous improvement in future teaching/learning activities.

\textbf{References}


Using Multiplayer Game Development to Integrate HND Modules and Develop Practical Understanding of Database Concepts

Joseph Hanke  
North East Surrey College of Technology (Nescot)  
Ewell, Epsom  
Surrey  
KT17 3DS  
JHanke@Nescot.ac.uk

Sage Lal  
North East Surrey College of Technology (Nescot)  
Ewell, Epsom  
Surrey  
KT17 3DS  
SLal@Nescot.ac.uk

Dilaksha Attanayake  
North East Surrey College of Technology (Nescot)  
Ewell, Epsom  
Surrey  
KT17 3DS  
RAttanayake@Nescot.ac.uk

Abstract
In recent years, game development strategies to teach ICT subjects have been employed by many pedagogical practitioners. Of these subjects, Object Oriented Programming (OOP) concepts, software engineering principles, 2D and 3D graphics programming were common. However, some research has been already done to show game development can be used to teach database concepts as well (Wadley, 2007). Nescot College has already used game development to teach OOP concepts and has experienced a significant improvement in student achievement both quantitatively and qualitatively. This paper outlines a proposed database teaching module that will run at Nescot for HND (Higher National Diploma in computing) students and which will be based on multiplayer game development. The proposed curriculum design aims to develop students’ technical skills and understanding of database concepts through a series of practical game design and development activities.

Keywords
Game development based teaching, learn database using game development.

1. INTRODUCTION
Database design and development is the fundamental basis for many careers in IT, and it is vital that student engagement in this area is maintained. Recent entry tests for higher level courses at Nescot College showed that student retention of understanding in this area was extremely weak compared to other areas such as programming and networking. It is generally believed that database concepts (for example, normalisation rules and relational database models) are relatively hard to understand and to teach (Kung, 2006). To solve this problem, many pedagogical practitioners used different teaching strategies (James, 2009; Cai, 1990) to teach databases in a more creative manner than traditional teaching methods.
Games are often used within education as learning tools to teach subjects such as English and Maths. Although games are now being used to teach computer related subjects as well, the authors of this paper believe that students need to be encouraged to move beyond being passive consumers of other people’s software towards being producers of creative gameplay experiences through the development of their own potential in technological skills. An analogy to this argument is the way that English language is developed – moving beyond reading what others have written towards the creative writing process themselves.

Studies in ‘Imaginative Learning’ (Egan, 2005) have shown that learners are more motivated to engage with a subject when given the freedom to develop their skills creatively within a narrative they themselves have constructed. For English language learning, this may involve writing a short story. We suggest that this imaginative learning approach be applied to technical computing subjects where the prevalent narrative story telling structure for this generation can be found in games and game culture (Squire, 2011).

More specifically, the aim of this research project is to teach database design and development by building the module content around the development of multiplayer games. This approach is significantly different from ‘Gamification’ in learning and teaching (Renaud et. al., 2011; Kapp, 2012; Osheim, 2013) in which the learners play a game that has been developed by tutors, rather than developing the game itself. In this paper, we focus on the design of the curriculum for a HND database module using this approach.

The rest of this paper is organised as follows. We first give a brief background of the project. Following this, we briefly describe the current HND database module at Nescot College. We then present our proposed novel game development-based database module that we will run for 2014/15 HND course. We also discuss how we intend to evaluate the impact of this method at the end of the academic year. We conclude by giving some future directions in this area.

2. **BACKGROUND**

Over the last year, we enacted a ‘supported experiment’ (Petty, 2003) to attempt to deliver Object Oriented (OOP) concepts through a Game Design and Development structure. Much of the first part of this module was taught through a conceptual model, with the theory given in relation to example games. A timed assessment was held halfway through the module, with the actual game creation using the concepts learned left to the last four weeks, where students were able to apply the techniques learned to a game of their own design.

We have found this to have had an extremely positive impact on the achievement of students, both in terms of high grades achieved and the attendance and motivation of students on the module. These results were significant considering that the module was delivered by the same teacher as the previous year. Figure 1 illustrates the improvement of the student grades after applying our method for OOP/ Game Design module.
At the end of the 2013/14 OOP / Game Design module, feedback from students was very encouraging. One of the students on the module has identified the benefit of learning technical skills through game development because:

“I had more drive on the assignment due to it being related to a topic which is interesting”

While another went on to explain the student engagement:

“pretty much everyone loves games so it keeps the students attention”

This latter comment reflects the seismic paradigm shift in this medium over the last 10 years. At present, 82% of 8-65 year olds in the UK regularly play games (Gaming Britain, 2014). As a consequence, most of our students now experience reality, social interaction and learning through gaming. This outcome has strongly influenced the aim of the current project explained in this paper.

3. CURRENT DATABASE MODULE

Database module content is currently task and product focused, where students are guided through a variety of practical classroom activities to support them in understanding key concepts relating to database design and development. For example, normalisation may be taught through demonstration of the concepts using an example database, and then supplying the students with a set of data to apply the normalisation techniques themselves. The problem with this method is that there is a dissonance between the work the students complete, and the end result of their work, which is ephemeral and abstract. Students benefit from working towards a goal rooted in reality, where they are able to view and play with the end result of their work.

Students who completed this module felt that the current scenarios used are uninspiring. Among others, the following comment shows that this was indeed the case with one of them.

“you are limited to what you can make as the scenario is asking you to make a set application which can limit you for what you actually want to make”
Although the overall feedback ratings from students (Table 1) were good, we think it can be improved with a suitable change of teaching strategy in place.

<table>
<thead>
<tr>
<th>Response</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Good</td>
<td>42%</td>
</tr>
<tr>
<td>Good</td>
<td>50%</td>
</tr>
<tr>
<td>Acceptable</td>
<td>8%</td>
</tr>
</tbody>
</table>

Table 1: Student feedback for 2014 HND database module

4. PROPOSED DESIGN
The HND database module delivery model we propose has four main stages. These stages are: preparation, controlled practice, free practice and impact assessment. Each stage is designed to help structure the teaching and learning process in order to improve the motivation, learning and understanding of students. The flow of these stages is illustrated in Figure 2.

4.1 Stage 1: Preparation
Teaching staff will develop a networked game to be used over a multiplayer interface, with the use of selected characters, power-ups and inventories which will need to be stored in a database backend. This game will be given to the students in the controlled practice stage in two versions. The first version is an executable to be used as a black-box example (game). The other version (project) is a front-end only version where the code of the game is given but without a database.

4.2 Stage 2: Controlled Practice
Students will start their work in this stage. They will spend the first few weeks learning conceptual data design relating to this module, with particular reference to the data structures required for game development. Following a timed assessment to verify student understanding of the required
knowledge, students will begin to work in small groups to construct a data model for the black-box game example provided by their teacher. Students will carry this out as their first task of the module assignment. The rationale behind the black-box example is two-fold: first, it helps them to understand the final model they are working towards. Secondly, it will keep them engaged in the learning activities in an environment that does not irritate or threaten learning but promotes a sense of collaboration that will be found in the real world.

Once completed, students will proceed to the development of their database and test the connection to the front-end across a network, competing for high scores as part of a group testing project. The module teacher will provide formative feedback to students throughout the process and summative feedback on submission.

4.3 Stage 3: Free Practice
In this stage of the module, students will work individually to design a new game which will be run over a network while connected to a database. This work will contribute to the second part of the module assignment. The further away from the example they are able to work, the higher the grade they will achieve in recognition of their capability for independent work.

The module teacher will assess the design model, and ascertain it's suitability for the development of a back end data structure. Students will submit their data design models before constructing the physical back end for the game they have designed.

This may be implemented and tested over a network, with the end goal of publishing on the college network or on a free game distribution network. This part will require a higher standard of OOP knowledge, which is not a part of this module, however, it is assumed at this stage the students will be proficient with these skills through other modules taught to them.

4.4 Stage 4: Impact Assessment
The evaluation of our proposed method will take into account both qualitative and quantitative assessment methods. Firstly, a survey will be given at the start of the module to assess the current knowledge of the database concepts of the students. An extended questionnaire will then be used at the end of the module to measure the distance travelled by the students. Secondly, the results of the final assignment submissions will be analysed to gain a quantitative measure of the achievement.

The project will be considered a success if the grade profile for the group as a whole rises from a merit to a distinction profile on this unit, which will demonstrate the effectiveness of imaginative learning on the ability of students to apply theoretical knowledge to an immediate practical task with tangible benefits within the creative process. Feedback from the group should also rise from 42% (Table 1) to over 80% 'very good' when feeding back on the quality of teaching and learning activities associated with this module.
5. **Controlled Practice Game Structure**

The black-box example game has been adapted from a game called ‘Space Vader’ developed by a Nescot student, who made use of the Microsoft XNA framework (Carter, 2007). The objective of the game is to try to defeat the alien enemies that are heading towards you as well as avoiding the rocks. The player then has to try to defeat the boss (the super enemy). Figure 3 illustrates a snapshot of this game.

![Space Vader's game screenshot](image)

**Figure 3: Space Vader's game screenshot**

The goals of the game are to move a player around the screen, shoot lasers when a button is pressed, avoid lasers and the randomly generated enemies, collect health and armour packs, and finally, defeat the boss at the end of the game. Users play the game using the arrow keys to move around and the space bar to shoot at enemies.

The database is hosted on a separate MS SQL 2008 server, which can be connected to each instance of the game. It will hold player information such as name, characters, games played, achievements and scores. The purpose of the database connectivity is to show statistics of the game – for example, own/high scores and how long they/others survived the game. Figure 4 is a conceptual entity relationship diagram for this database. The conceptual model students will be arriving at in the initial stage of their first task of the assignment will be similar to this.
6. **Teaching Plan**

As described, the database module is planned to run in four stages (preparation, controlled practice, free practice and impact assessment) aligned with the four stages of the proposed teaching model (see Table 2). Each stage clearly defined in order to monitor student progress over the course.
<table>
<thead>
<tr>
<th>Stage</th>
<th>Student Activity</th>
<th>Teacher Activity</th>
<th>Concepts covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Complete preliminary questionnaire. Learn database concepts and game design. Timed assessment. Attempt task 1 of assignment (group).</td>
<td>Deliver lectures, seminars and tutorials. Improve and develop teaching materials. Formulate and monitor group work progress. Formative assessment of students.</td>
<td>Create an entity relationship diagram by identifying entities, attributes, relationships and constraints from a set of requirements. Normalize the entity relationship diagram to third normal form. Enhance the entity relationship diagram to utilize several data modelling techniques. Create a data flow diagram by identifying processes, external agents, information stores and information flows. Engineer the entity relationship model into an initial relational database design. Optimize the relational database design. Complete the physical model and generate the DDL.</td>
</tr>
<tr>
<td>4</td>
<td>Complete end of module review survey and extended questionnaire.</td>
<td>Assess assignments. Analysis of preliminary and post questionnaires.</td>
<td>Review and share learner journey.</td>
</tr>
</tbody>
</table>

Table 2: Teaching schedule
7. **SUMMARY**

The previous experience of the OOP game delivery has indicated that student engagement and achievement can be improved through the targeted application of game development as part of a module. Students develop their own understanding of theoretical concepts when they can see a direct result of their actions enacted in a graphical environment. To this end, we are keen to apply a similar, but more structured teaching and learning model to the database module, and to measure its impact. If successful, we will be integrating this style of game development based delivery to other modules on the same module.

We have identified potential risks associated with the project. The primary risk is that this style of learning based on game development may not suit all learners, for example, special needs students. As with any other strategy, other real world applications of databases will not be covered by focussing on gaming scenarios.

8. **FUTURE WORK**

A comprehensive study with more controlled classroom sessions could be employed to evaluate our method in the future. The analysis of the results of this pilot study will help us to determine how to carry out such more controlled study. At the end of the proposed study, we hope to introduce this method to our other higher level courses such as the BSc Computing top-up courses (BSc top up courses give students the opportunity to top up their existing skills and qualifications to a full degree). If successful, we would like to utilise this method across other areas such as networking and web development in the future.

**REFERENCES**


Nescot College, Ewell, Epsom, Surrey, UK, www.nescot.ac.uk, [Last Accessed on 02/06/2014]

Osheim, Darcy Elaine, "This Could Be a Game!": Defining Gamification for the Classroom" (2013).Master’s Theses. Paper 4304.


EDNA, a Software Tool for Verifying the Normalisation of Relations during the Logical Database Design Process

Adam Floyd  
Department of Applied Computing  
University of Buckingham  
Buckingham MK18 1EG  
adam.floyd@buckingham.ac.uk

Hongbo Du  
Department of Applied Computing  
University of Buckingham  
Buckingham MK18 1EG  
hongbo.du@buckingham.ac.uk

Abstract
Logical design of relational databases is an important part of a typical undergraduate database module in a range of computing curricula such as software engineering, computer science, computing, business computing, information systems, etc. A crucial part of logical design is normalisation, which is often perceived as a difficult topic to study by students and not so easy to teach by lecturers. This paper presents a software tool named EDNA that aims to automate the normalisation of relation schemes to 2NF, 3NF or BCNF. Upon user request, EDNA can generate standard SQL statements necessary for the creation of tables of the normalised schemes for a number of designated database management systems. EDNA is equipped with a simple interface that allows the user to enter a list of attributes and specify or edit functional dependencies among the attributes with ease. The software can be used as a teaching tool for verifying the validity of manually normalised relation schemes, aiming to enhance the student learning experience of the topic.

Keywords
Relational database, relational schemes, logical database design, normalisation.

1. INTRODUCTION
The relational data model is still the dominant form of data model used in current database systems. The relational database therefore remains a central theme of a typical undergraduate database module at the introductory level, and is covered by most texts on databases, such as Date (1995) and Elmasri & Navathe (2011). One important aspect of relational database theory is database design through normalisation.

Normalisation is a formal process of decomposing loosely organised and often large relation schemes into smaller and more tightly coupled structures according to a set of constraints on data such as functional dependencies between attributes (Elmasri & Navathe 2011). The higher the normal form relation schemes are, the more constraints are imposed. The process of normalisation aims to reduce the amount of data redundancy and avoid the associated update anomalies in the practical use of the final database.

Many students consider normalisation a hard topic to learn, and many lecturers find it not so easy to teach. Although attempts have been made in the past to dispense with normalisation by adopting the Object Role Modelling (ORM) methodology (Byrne et al. 2003) or enhancing “good” conceptual designs (Byrne 2003), such an approach of database...
design without normalisation has not been endorsed by the mainstream (Nelson & Jayakumar 2012). Normalisation remains a fundamental topic within the relational database design that is being taught in classrooms across universities, and a practice exercised widely by many database designers in the industry. A software tool that can assist the process of normalisation and/or the teaching of normalisation would be therefore desirable.

In this paper, we report a software tool known as EDNA (an Enhanced Database Normalisation Automator). The software automates the process of normalisation of relational schemes to second (2NF), third (3NF) or Boyce-Codd (BCNF) normal forms. Although the software was primarily intended for assisting database designers in the field, it can also be used as a tool to validate the result of manual normalisation while students are learning this topic or practising normalisation in their database design projects. The purpose of introducing EDNA to the TLAD community is to enhance the student learning experience of this useful topic.

The rest of the paper is organised as follows. In section two, a brief review of database design approaches and the position of normalisation in the process of database design is given. Existing works in the literature about automatic normalisation are also presented and discussed. Difficulties in teaching and learning this topic are summarised. Section 3 describes the main functions and the interactive user interface of EDNA together with issues relating to the organisation and implementation of the software. Section 4 presents a brief and qualitative evaluation about the effectiveness of the software, its strengths and limitations in practical use for teaching and learning before section 5 concludes the paper with future works.

2. RELATIONAL DATABASE DESIGN AND NORMALISATION

2.1 Normalisation Process and Relational Database Design

Relational database design tends to follow one of two established approaches. In the bottom-up approach (Date 1995), all attributes describing data objects are collected into a single universal relation scheme where various dependencies between the attributes such as functional dependencies are globally studied and specified. A step-by-step stringent normalisation process is then followed to decompose the universal scheme into higher normal form schemes. For instance, 2NF schemes are obtained to ensure total dependencies from non-key attributes to the primary key, followed by 3NF normalisation to ensure no transitive dependencies exist from non-key to key attributes. The process may eventually continue up to 5NF normal form schemes, but 1NF, 2NF and 3NF schemes are arguably the most common. The process of normalisation is well established, having been proposed by Codd as part of his wider work on relational database theory in the late 1960s (Codd 1970).

Alternatively, a top-down approach of design can be followed (Elmasri & Navathe 2011). In this approach, a high level conceptual model such as the Entity-Relationship (ER) model is first used to describe a database in terms of high level concepts like entities (data objects), relationships (associations) and attributes (properties). The conceptual design result, in the form of an Entity-Relationship Diagram (ERD) for example, is then transformed or mapped into a set of relation schemes according to a set of explicit rules (Elmasri & Navathe 2011). However, the ER model has its own limitations: it does not explicitly express the transitive functional dependencies between attributes of an entity, and hence the resulting schemes are in second normal form but not always in fourth normal form as suggested in Byrne (2003), unless the designers explicitly describe the concerned attributes as separate entity
types. It is therefore safe to state that the resulting schemes from the transformation may still not be final and need further normalisation to decompose into higher normal forms.

In summary, it appears that no matter which approach one takes, a normalisation process of relation schemes is normally needed. In fact, Connolly & Begg (2010) suggest using normalisation at least for validating the resulting relational schemes from the process of translating ERD into table structures.

2.2 Practical Problems in Teaching and Learning Normalisation
Based on the first author’s recent learning experience and the second author’s past teaching experiences over many years, students tend to face a number of problems when studying normalisation. First, students often consider the concepts of functional dependency and normalisation to be abstract and have difficulty relating them to real-world problems, an issue that is further compounded by the use in many textbooks of abstract terms such as ‘dependency’ that separate students from the data reality. Second, students often find it non-trivial to effectively apply the concept of the functional dependency in stating facts through non-key attributes about entities and relationships expressed by the keys. This is less of a problem when only a small number of attributes are involved, but the amount of difficulty increases significantly when the number of attributes gets large and the dependencies tend to involve multiple key and non-key attributes. Third and more seriously, despite textbook guidelines on how to normalise, students tend to conduct the normalisation process in a much less disciplined manner with a great chance of trial and error. This may not be entirely the fault of the students as some tutors may be more interested in formally defining normal form schemes accurately than explaining how to achieve them practically. Last but not least, students often find the manual normalisation process boring and time-consuming, and hence the risk of human error is constant. As a result, the conduct of normalisation is often unsatisfactory, causing loss and bad decomposition of schemes and ultimately a collection of poorly organised tables that are difficult to work with.

2.3 Normalisation Algorithms
Algorithms for decomposing relation schemes up to 3NF and BCNF existed in early literature on relational database design in the late 1970s and early 1980s, and can be found in a number of reputable textbooks on databases today, for instance Elmasri & Navathe (2011) and Ullman & Widom (2008). However, these high level abstract algorithms have been used in those texts to demonstrate the correctness of a formal manual decomposition process for normalisation, rather than as a basis for the development of a fully automated normalisation system. In fact, these algorithms in their abstract forms are more useful in understanding the theoretical aspects of relational database design. As indicated in (Ullman & Widom 2008), some of these algorithms are potentially computationally expensive and hence were prohibitive at that time of limited computing resources.

Nevertheless, attempts at automating the normalisation process have been continuing for more than three decades. The earliest work can be traced back to 1983 when a prototype tool based on Ullman’s algorithms (Travis 1983) was developed for the United States Air Force, but little detail regarding implementation and performance of the final software is known. Du and Wery (1999) developed Micro, the first automated system that used efficient data structures to implement their own 2NF algorithm and the 3NF and BCNF algorithms from Elmasri & Navathe (2011). However, not only does the system still lack the
functionality of table creation for serious practical use, but it is also now outdated in terms of software modularity and portability.

Since then, several follow-ups have been reported. More recent is a software tool known as RDBNorma that claims more efficient solutions than Micro using simpler data structures (Dongare et al. 2011), but this claim cannot be substantiated beyond a simple timing comparison that is lacking in technical details of the test environment. Bahmani et al. (2008) proposed graphical representations of functional dependencies known as directed graph dependency matrices to visually illustrate dependencies between attributes and provide a means of manipulating the dependencies to achieve normalised relation schemes. Akehurst et al. (2002) proposed a class-based method for modelling database schemas using the Unified Modelling Language (UML) extension known as Object Constraint Language. Attaran and Hosseinian Far (2011) developed a tree-based visualization tool for producing a normalised database schema by connecting trees and avoiding the use of functional dependencies entirely.

The majority of these systems have however been developed purely as academic research projects, with an emphasis on efficiently delivering accurate results over explaining the process involved. Their authors have made little or no attempt to build in features that are desirable for supporting the teaching and learning of database design, and as such these systems are poorly suited to use in an educational environment. The emphasis of teaching and learning strategies therefore remains very much on the explanation of normalisation as a manual process, with little practical support and the attendant drawbacks as discussed above.

3. EDNA, an Enhanced Database Normalisation Automator

3.1 Overview of the Main Functions

The primary function of EDNA is to take a user-defined set of atomic attributes and functional dependencies, and perform normalisation up to and including Boyce-Codd normal form. The normalisation procedure is carried out in a fully automated fashion using the user-defined dependencies, with no further user intervention required. When complete, the user is then able to specify data types for each attribute of the resulting decomposed relation schemes and generate SQL commands to create the physical table structures in a number of popular database management systems. The latter facility provides an opportunity to introduce a practical element into the teaching of database technologies; in that students can actually create the databases they have designed and enter some sample data, giving them a greater understanding of the differences between normal forms and an appreciation of the problems caused by poor design.

EDNA includes a facility to save design data to disk in the form of a project file at any stage of the process. This allows a set of attributes and functional dependencies to be defined in advance of a lesson and loaded into EDNA on demand. The same set of dependencies can be given to students to normalise manually as a practical task, and then loaded into EDNA for automatic normalisation so the results can be compared to validate their understanding of the subject. It is to be hoped that this comparison will provoke a discussion, especially if EDNA’s results differ from the students’ own, which will help to further their knowledge of the subject.
3.2 The Interactive User Interface
EDNA is designed to be easy to use and presents a Windows Forms-based interface that should be familiar in style to most computer users. The user is guided through the stages of the database design process by the use of tabs that follow the logical order of operations from definition of attributes to database creation.

As shown in figure 1, the first tab presents a simple interface to define the attributes required within the database. New attributes are entered simply by typing them into the text entry field and clicking a button or pressing the enter key. Existing attributes can be renamed or deleted, and changes made to attribute definitions will automatically propagate through to any functional dependencies that exist. EDNA enforces the uniqueness of every attribute name, and as such any attempt to create a duplicate name will be rejected.

Functional dependencies are defined on the second tab, shown in figure 2. The user first selects one or more attributes that form the determinants from the left-hand list, and similarly selects those that form the dependents from the right-hand list, before clicking the ‘Determines’ button. This will create a new functional dependency, which is added to the list of existing dependencies on the right hand side. It is possible to select multiple dependents within a single functional dependency, but these will automatically be split into multiple dependencies meeting the requirement of minimal cover that each dependency must have exactly one dependent (Date 1995) (Elmasri & Navathe 2011). EDNA does permit the user to enter trivial dependencies, but these will be automatically removed as part of the normalisation process.

Once the definition of attributes and functional dependencies has been completed, normalisation is performed using the third tab, as shown in figure 3. The desired normal form is selected from the available options of 2NF, 3NF and BCNF, and the ‘GO’ button clicked to carry out the normalisation process.
No further user intervention is required during the process and the resulting decomposed relation scheme is displayed in a tree view form with primary key attributes highlighted in red. The resulting relations are named using a default system, but can be renamed by editing the default names, as partially seen in figure 3, where the first three relations have been given meaningful names but the remaining ones retain their default names.
A relation scheme defined by EDNA can be created as a physical table structure in one of the popular database management systems by first selecting the desired DBMS from the options. An appropriate data type is then specified for each attribute by selecting a suitable option from the list of data types available in that particular DBMS in the combo box on the right. When multiple instances of an attribute exist, all are given the same data type to prevent join issues caused by mismatches between primary key and foreign key pairs. Finally, clicking the ‘Generate SQL’ button will generate a set of SQL commands that can be executed within the DBMS to create a physical database structure that matches the relation scheme produced by EDNA, an example of which can be seen in figure 4.

3.3 The Implementation Issues

The major challenge of producing an automated normalisation tool lies in the difficulty of converting the high-level abstract algorithms presented in the literature into functioning and fully tested program codes. These algorithms are not intended for this purpose and hence are written in a manner close to natural language, often using a single sentence to describe a task that must be performed by a complex function. Detailed analysis of the intended function and results of these algorithms was therefore necessary in order to translate them into the code written in Visual C#.

```sql
CREATE TABLE `Jobs` (`Date` datetime, `Emp#` int UNSIGNED, `Jobtitle` varchar(20), `Salary` decimal(6,2) UNSIGNED, CONSTRAINT PK_Jobs PRIMARY KEY (`Date`, `Emp#`));
CREATE TABLE `Depts` (`Dept#` tinyint UNSIGNED, `Dbudget` decimal, `Mgr#` tinyint UNSIGNED, CONSTRAINT PK_Depts PRIMARY KEY (`Dept#`));
CREATE TABLE `EmployeeProjects` (`Emp#` int UNSIGNED, `Phone#` text(10), `Proj#` int, CONSTRAINT PK_EmployeeProjects PRIMARY KEY (`Emp#`));
CREATE TABLE `Managers` (`Mgr#` tinyint UNSIGNED, `Dept#` tinyint UNSIGNED, CONSTRAINT PK_Managers PRIMARY KEY (`Mgr#`));
CREATE TABLE `Offices` (`Office#` bigint UNSIGNED, `Area` tinytext, `Dept#` tinyint UNSIGNED, CONSTRAINT PK_Offices PRIMARY KEY (`Office#`));
CREATE TABLE `Phones` (`Phone#` text(10), `Office#` bigint UNSIGNED, CONSTRAINT PK_Phones PRIMARY KEY (`Phone#`));
CREATE TABLE `Projects` (`Proj#` int, `Dept#` tinyint UNSIGNED, `Pbudget` decimal(6,2), CONSTRAINT PK_Projects PRIMARY KEY (`Proj#`));
```

Figure 4. MySQL table creation commands for the schema shown in Figure 3

EDNA has been written using the object-oriented style of programming, as the database design procedure involves a number of structured entities that can be considered “objects” in the programming sense of the word. It was necessary to carefully design data structures capable of storing these objects and preserving their format, while allowing efficient in-memory processing by the computationally-intensive normalisation algorithms. This process was further complicated by the need to compose single objects into sets; for example, each individual functional dependency can itself be represented as a single object, but in order to represent the complete schema it is also necessary to store an ordered set of multiple instances of this object together with implementing methods to manage this set.

The design of EDNA is such that the functionality has been split between two separate but linked components for greater flexibility, but it remains a stand-alone program that requires no third party software to be installed. As shown in figure 5, the user interface is provided by a Windows Forms application, which is linked to a DLL (dynamic link library) file that...
contains the normalisation algorithms themselves, both parts being written in Visual C# to ensure compatibility. This approach widens the appeal of the program as potential users are not limited to using the interface provided in the complete EDNA package, but able to integrate the normalisation library into their own software, for instance to produce an alternative version that provides greater visualisation of the intermediate stages of the process to further enhance the learning experience.

A significant property of the normalisation procedure that adds complexity to any implementation of an automated system is its non-deterministic nature. For any given set of functional dependencies, there is not usually a single resulting schema, but a set of multiple schemas, each of which decomposes the relations differently but is equally valid according to the rules of decomposition. Which of the possible schemas is obtained can depend on the order in which the attributes and dependencies are entered, stored and processed, making it difficult to produce consistent and repeatable output. To overcome this issue, EDNA imposes an alphabetical sort order on the set of functional dependencies, meaning that the same result will always be obtained for a particular input set, but this may differ from those published elsewhere.

A specific concern pertinent to the use of EDNA in an educational context is that the published works on the subject (Date 1995) (Elmasri & Navathe 2011) generally do not present an algorithm for achieving second normal form. The reason for this omission is that this normal form does not fully address the issue of redundancy and is therefore not generally used in real-world databases, and it is possible to normalise directly to the more useful 3NF without passing through 2NF as an intermediate stage, so such an algorithm would be redundant in a commercial system. However, when teaching the theory of normalisation, 2NF is an important concept and it is therefore desirable for EDNA to allow normalisation only to this stage in order to compare the resulting database schemas with those of 3NF. As such, EDNA includes an additional algorithm that was developed in (Du & Wery 1999) to achieve second normal form, alongside improved versions of the 3NF and BCNF algorithms based on those from the same source.

4. EVALUATION AND DISCUSSION
EDNA has already been used in a teaching and learning role by students studying the Principles of Database Systems module at the University of Buckingham, as part of their module project. This involved the design and implementation of a working database to meet a specific need defined by a case study, and the assessed project report was required to include a comparison of the students’ own manual schema designs with those produced by EDNA. In general, EDNA has received positive feedback from the students concerned, who felt it was a useful tool that enhanced their understanding of what is perceived to be a difficult topic.
However, it must be remembered that class sizes at Buckingham are generally much smaller than those of other institutions, and the undergraduate programme is completed in just two years, thus limiting the amount of resources available for testing. The Principles of Database Systems class of 2013 consisted of only 18 students, who carried out the module project in groups of two or three, and this module is taught only once per year; at the time of writing this paper, it has yet to be taught again to the first-year students of the 2014 intake. Hence, although it is intended to continue testing the software as part of this module in future years, it has not yet been possible to increase the test sample size, and with such a small sample detailed quantitative analysis of the test results cannot be performed meaningfully so only a limited qualitative evaluation can be provided. One of the main reasons for presenting EDNA to the TLAD community is the opportunity for obtaining a much larger set of testers, from whom some more meaningful conclusions regarding the effectiveness of the tool in assisting the teaching can be drawn in future.

The first two of the problems detailed in section 2.2 are difficult, if not impossible, for any software product to solve as they are reliant on human powers of explanation and comprehension to ensure that students gain a correct understanding of the subject. However, in this context, EDNA does provide an opportunity for learning by experimentation as it allows different combinations of functional dependencies to be tried and the results observed, hopefully providing a justification to the students of the importance of learning these concepts. EDNA has concentrated on addressing the third and fourth problems as the rigid method of entering attributes and dependencies imposes the discipline that is often lacking in manual normalisation, and the decomposed relation schemes are produced in a matter of seconds rather than hours or days, thus preventing boredom from setting in.

The performance of EDNA appears more than satisfactory in terms of speed, even when dealing with large sets of complex dependencies. The program was developed on a 2007 Fujitsu Siemens laptop running Windows XP with a 1.6GHz processor and 1GB of memory, which is an extremely low specification by current standards, and even on this machine, the decomposed relations were generated virtually instantly, so there should be no significant performance issues on any modern system. It has also been tested on the Windows 7 and 8 operating systems and no issues were encountered.

All three of the desirable properties of a normalised database are fully supported by EDNA. The system performs a complete decomposition, in which no extra attributes are added and none of the user-specified attributes are removed, so the set of all attributes of the decomposed schema remains equal to that of the original universal relation. Wherever possible, all non-redundant functional dependencies are preserved, except in those rare cases where producing a valid BCNF schema makes dependency loss unavoidable as reported in most textbooks. EDNA decomposes the relation scheme using a lossless join method, such that all of the information in the original universal relation can be recovered by using natural join operations on the decomposed relations.

In certain circumstances involving multi-value dependencies considered by 4NF normalisation, a many-to-many relationship exists between multiple relations and they must be connected by a link table with no additional attributes, but there is no valid functional dependency that can describe this relationship. For these cases, a special 'null' dependent is provided, which can be used to specify that two or more attributes of different relations are related but do not determine any other attributes, such that an all-key relation is created to link these relations.
The database schemas generated by EDNA have been validated for correctness up to and including third normal form, and are at least partially compliant with fourth normal form. The resulting schemas occasionally vary from those published as examples in the literature, but this is due to the non-deterministic nature of the decomposition process and they have been verified to be equally compliant with Codd’s rules of decomposition (Codd 1970).

While it provides similar functionality to certain of the existing works discussed in section 2.1, EDNA is distinguished from these by its method of implementation. The object-oriented programming approach is a unique feature of the software and the data structures have been carefully designed for the most efficient storage and processing of large and complex datasets. EDNA’s modularity is another significant difference that lends it far more flexibility than existing solutions, which are typically implemented as homogenous ‘black box’ programs; the separation of the normalisation functionality from the user interface provides many opportunities for independently developing each component to suit specific needs.

5. CONCLUDING REMARKS

This paper presented a software tool EDNA for automatic normalisation for supporting teaching and learning of relational database design. By automating the process of normalisation, the system allows students and teachers to verify their own manual normalisation results, and hence improve their learning and teaching experience of this difficult but practically useful topic. The tool is intended to be freely available for teaching purposes across computing departments in the UK (download site: http://www.buckingham.ac.uk/appliedcomputing/researchstudents#databases). Through the widespread use of the tool, we aim to improve its performance and usability further with future new versions.

In its present form, EDNA is by no means a finished product and the emphasis of its development has been on ensuring correct functionality rather than providing an elegant user interface. It is acknowledged that the interface as it stands, while functional, is somewhat basic and user-unfriendly, so immediate future work involves providing documentation and generally improving the appearance and usability of the user interface. It is also felt desirable to add a facility to print sets of functional dependencies or export them to another program, so they can easily be given to students for normalisation as a manual task. In the longer term, the proposed work naturally includes more algorithms for 4NF and 5NF normalisation so that a complete set of normal forms can be obtained upon user request. Another future work will be to exploit the possible use of touch screen technology with visual gestures such as drag-and-drop, pinching etc. in specifying data dependencies to further improve the usability of the software for teaching and learning purposes.

In its current form, the system still requires the user to understand and correctly apply the concept of functional dependencies in order to manually enter them, but there have been a number of suggestions that it would be desirable to automate this process by providing a first normal form dataset and letting the software automatically determine the functional dependencies within. While this is a sound concept in theory, there is a major practical difficulty to overcome. Accurate results would depend on the provision of potentially large datasets that include all possible domain values for every attribute, as the data contained in the provided subset may imply functional dependencies that hold upon this subset but are violated by legal data entered at a later date; this incorrect specification of functional dependencies could result in the very design issues that EDNA initially set out to solve, and this functionality could easily become a separate research area in its own right.
REFERENCES


