Bachelor's Thesis

Information Technology

Software Development

2014

Pauli Kettunen

DATABASES IN WINDOWS PHONE-APPLICATION DEVELOPMENT

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Software Development

June 2014 | 32

Tiina Ferm

Pauli Kettunen

DATABASES IN WINDOWS PHONE -APPLICATION DEVELOPMENT

This thesis stems from the development of DiscGolf for Windows Phone, which is a disc golf score-card application for Windows Phone. The beginning provides a brief look into databases, which are the main means of storage in the application. After that Windows Phone itself, is discussed, including its history and how applications can be developed for it. Finally, the study looks into how a database can be used in a Windows Phone application.

Databases have been used for data storage since the 1970's, and have evolved into an effective means of data storage, especially when dealing with large amounts of structured data. Most developers know at least some SQL, which is the most commonly used query language when dealing with relational databases.

Windows Phone is a mobile operating system developed by Microsoft and originally released in 2010. The initial versions of Windows Phone were based on Silverlight, which developers can use to program applications for the platform. Development can also be done using XNA, which is designed with game development in mind.

The goal of the DiscGolf application project was to get acquainted with Windows Phone programming and as many commonly used techniques as possible, while keeping the application relatively simple. The main focus in the project was data storage, and in particular the local database where the scores are stored.

This thesis can be used as a reference or "cheat sheet" when programming database-driven Windows Phone applications, as it shows how all of the necessary pieces for creating and maintaining a database are made and how they work.

KEYWORDS:

Windows Phone, database, SQL, Silverlight, XNA, Microsoft

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikan koulutusohjelma | Ohjelmistotuotanto

Kesäkuu 2014 | 32

Tiina Ferm

Pauli Kettunen

TIETOKANNAT WINDOWS PHONE -SOVELLUSKEHITYKSESSÄ

Tämän opinnäytetyön perustana toimii DiscGolf for Windows Phone -sovellus, joka on frisbeegolfin pistekorttisovellus Windows Phonelle. Ensimmäisessä luvussa kuvataan tietokantoja, jotka toimivat sovelluksen pääasiallisena tietovarastona. Toisessa luvussa tutustutaan lähemmin Windows Phoneen ja sen historiaan, kehitystyökaluihin ja ohjelmointikieliin. Lopuksi tutustutaan projektityönä tehtyyn DiscGolf for Windows Phone-sovellukseen, ja erityisesti sen käyttämiin tallennusratkaisuihin.

Tietokantoja on käytetty tietojen tallentamiseen jo 1970-luvusta lähtien, ja ne ovat kehittyneet tehokkaiksi työkaluiksi, jotka soveltuvat erityisesti suurien tietomäärien käsittelyyn. Lähes jokainen koodaaja osaa ainakin vähän SQL:ää, joka onkin relaatiotietokantojen yhteydessä yleisimmin käytetty kieli.

Windows Phone on Microsoftin vuonna 2010 julkaisema mobiilikäyttöjärjestelmä. Alkuperäiset Windows Phone -versiot perustuivat Silverlight-tekniikkaan, jonka avulla kehittäjät voivat ohjelmoida sille omia sovelluksiaan. Sovelluksia voi kehittää myös pelikehitystä varten suunitellulla XNA-tekniikalla.

DiscGolf-sovellusprojektin tarkoitus oli tutustua Windows Phone -ohjelmointiin ja mahdollisimman moneen siinä käytettävään tekniikkaan. Sovellus pyrittiin kuitenkin pitämään mahdollisimman yksinkertaisena. Päävastuualueena sovelluksessa oli tietojen tallennus, ja erityisesti paikallinen tietokanta johon tulokset tallentuvat.

Opinnäytetyötä voi käyttää apuna tietokantapohjaisia Windows Phone –sovelluksia ohjelmoidessa, sillä se sisältää ohjeet kaikkien tietokannan luomiseen ja ylläpitämiseen liittyvien komponenttejen tekemiseen ja niiden käyttöön.

ASIASANAT:

Windows Phone, tietokanta, SQL, Silverlight, XNA, Microsoft

CONTENT

LIST OF ABBREVIATIONS (OR) SYMBOLS **6**

1 INTRODUCTION **7**

2 DATABASES **9**
2.1 Database types 9
2.2 Advantages over other types of data storage 10
2.3 SQL 10

3 WINDOWS PHONE **12**
3.1 History 12
3.1.1 Windows Mobile 12
3.1.2 Windows Phone 7 14
3.1.3 Windows Phone 8 and 8.1 16
3.2 Application development 17
3.2.1 C# and XAML 18
3.2.2 Visual Studio and Expression Blend 19
3.2.3 LINQ 19

4 THE APPLICATION PROJECT **20**
4.1 The problem 20
4.2 The solution 21
4.3 Planning and designing the database 22
4.4 Writing the database queries 25
4.5 Version 2.0: Schema update 27

5 FUTURE CONSIDERATIONS **29**

6 CONCLUSIONS **30**

REFERENCES **31**

PICTURES

Picture 1. Pocket PC 2000 Today Screen (Wikipedia 2010) 13

Picture 2. Windows Mobile 6.5 (Wikipedia, 2010) 14

Picture 3. A Windows Phone 7 device produced by Nokia, the Lumia 800 (Wikipedia, 2012) 15

Picture 4. The Nokia Lumia 930, a Windows Phone 8.1 device (Wikipedia 2014) 17

Picture 5. DiscGolf for Windows Phone 22

Picture 6. The relational diagram of the database 23

LIST OF ABBREVIATIONS (OR) SYMBOLS

SQL	Structured Query Language
LINQ	Language Integrated Query
JSON	JavaScript Object Notation
XML	Extensible Markup Language

1 INTRODUCTION

As in all application development, data storage plays an important role in Windows Phone applications. Unless data is stored somehow, the user's actions are forgotten when the application is closed or when the current page in the application is closed.

Data storage can be handled in various ways on Windows Phone, including serializing the data into JSON or XML files, or using local or remotely hosted databases. Essentially, data storage on Windows Phone (and other mobile platforms) works the same way as in any application platform, but on mobile devices certain limitations must be taken into account.

Current mobile devices do not have as much memory as computers, and therefore programmers must consider what to keep in memory and what to store elsewhere, but reading and writing from and to a file or database is more processor intensive and can drain the phone's battery. File reading and writing must also be carefully synchronized to prevent data from being read at the same time as it is being written, which can be difficult.

This thesis explains how data can be stored in a local database on Windows Phone. First the study looks into databases in general. Since databases are a very broad subject, only the basics and the most common types of databases are covered.

After defining a database, the study discusses what Windows Phone is, starting with where it has come from. After the short history, the focus lies on how Windows Phone applications can be programmed and what tools are commonly used.

The final chapter deals with the programming of the local database and queries used in DiscGolf for Windows Phone, which is a disc golf scorecard application for Windows Phone. The chapter provides an example on how a local database can be created and how its contents can be manipulated. It also shows how the initial schema of the database can be updated.

2 DATABASES

A database is a structured collection of data. There are several different database types that suit different requirements. The data contained in a database can be accessed and manipulated using a database management system (DBMS). The terms database, database server, database system, data server, and database management systems are often used interchangeably, even though they have different meanings. (Sharma et al. 2010, 23-24)

2.1 Database types

If data is stored in tabular form, it is called a relational database. When data is organized in a tree structure form, it is called a hierarchical database. Data stored as graphs representing relationships between objects is referred to as a network database. (Sharma et al. 2010. 23)

In 1969, the Committee on Data Systems Languages released its first specification about the network data model. A collection of record types and keys form a CODASYL network or database. A child can have more than one parent, and each record type can point to each other with next, prior and direct pointers. (Sharma et al. 2010, 28)

The hierarchical model organizes its data using a tree structure. The root of the tree is the parent followed by child nodes. A child node can only have one parent node, but a parent can have many child nodes. In a hierarchical model, a collection of named fields with their associated data types is called a record type. Each instance of a record type is forced to obey the data description indicated in the definition of the record type. Some fields in the record type are keys. (Sharma et al. 2010, 29)

The relational model uses a collection of tables to represent both data and the relationship among the data. A table is a collection of rows and columns. Each column has a unique name. Each row in the table represents a collection of related data values. A row is called a tuple, a column is called an attribute and the table is called a relation. (Pallaw 2010, 65)

2.2 Advantages over other types of data storage

Splitting data into a number of related tables brings many advantages over for example serializing data into XML documents. Both formats can be used to store data and they both have established techniques for extracting the data they contain, but databases are better for handling large volumes of data. Databases have mature management systems that can efficiently and reliably maintain large quantities of structured data. This data can be updated using transactions that ensure the integrity of the database and the content can be extracted very quickly when properly configured. (Stephens 2010. 24, 27-28, 34)

Relational databases are the most commonly used kind of database today. A lot of very powerful companies have spent a huge amount of time building them. That means that relational databases have been thoroughly studied and have evolved to the point where they are quite useful and effective. Unless the application has very special needs, relational databases are usually an excellent choice. (Stephens 2010. 27-28)

2.3 SQL

SQL is the main language used in relational database systems, and has its roots in IBM research conducted in the late 1960s. Despite several attempts at standardization, virtually every relational database management system has its own variation of SQL, each being slightly different in syntax and implementation details. (Alex, 2011. 28)

SQL provides language constructs to insert and manipulate data through statements such as INSERT, SELECT, DELETE, and UPDATE. SQL is a declarative language, which means it instructs the database management system about what to do, but leaves the details about how to handle the request to the system itself. (Alex, 2011. 10)

3 WINDOWS PHONE

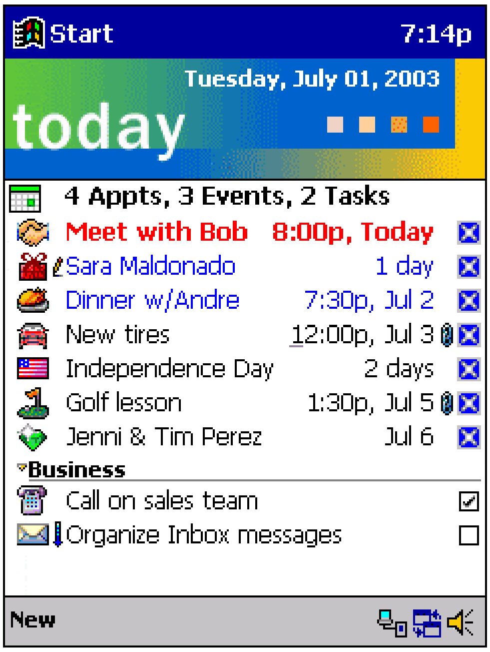
Windows Phone is an operating system for mobile devices developed by Microsoft. It was originally launched in October 2010. Unlike its predecessor Windows Mobile, Windows Phone is aimed at the consumer market instead of the enterprise market. Windows Phone 7 devices were produced by Dell, HTC, LG, Samsung, Acer, Alcatel, Fujitsu, Toshiba, Nokia, and ZTE. Windows Phone 8 devices are currently being produced by HTC, Huawei, Nokia, and Samsung. (Wikipedia 2014b)

3.1 History

Microsoft started its work on portable devices in 1990. Two of their initial mobile operating system projects were scrapped due to hardware limitations, and out of the two disbanded teams the Windows Mobile team was born. Windows Mobile was developed until 2010, when Microsoft unveiled Windows Phone 7. (Wikipedia 2014a)

3.1.1 Windows Mobile

In the year 2000, Microsoft released Pocket PC 2000, which was later renamed to Windows Mobile. At this time, the operating system had similar appearance to Windows 98, Windows ME and Windows 2000, and featured many built-in applications including Windows Media Player and Pocket Internet Explorer. (Amy 2010)



Picture 1. Pocket PC 2000 Today Screen (Wikipedia 2013)

The first version under the Windows Mobile name was released in 2003. Windows Mobile 2003 came in several different editions, some of which contained phone-specific features like SMS-support. Windows Mobile 2003 Second Edition was released in 2004 and contained enhancements like the ability to switch between portrait and landscape display modes and support for new screen resolutions. (Amy 2010)

Windows Mobile 5.0 was released in 2005, and is still supported until October 13, 2015. This new version featured increased battery life by saving most of its data onto flash memory instead of RAM like on the previous versions. On earlier versions up to 50% of battery power was reserved just to maintain data. (Wikipedia 2014a)

In 2007 Microsoft unveiled Windows Mobile 6.0, which is strongly linked to their then new Windows Live and Exchange 2007 products. Windows Mobile 6 was meant to look similar to Windows Vista, even though it works much like Windows Mobile 5. Two upgrades to Windows Mobile 6.0 were released before development was shifted over to the new, fully multi-touch-enabled Windows Phone 7. (Amy 2010)

Picture 2. Windows Mobile 6.5 (Wikipedia, 2010)

3.1.2 Windows Phone 7

The first version of Windows Phone was released near the end of 2010. The desktop-like user interface of Windows Mobile was ditched in favor of Microsoft's new Modern design language based, touch screen optimized interface. The kernel in Windows Phone is based on the same version of Windows CE as the one used in the last versions of Windows Mobile. Windows Phone 7 is not compatible with Windows Mobile applications, which is a result of quick development. (Järvinen 2012. 12-13)

Picture 3. A Windows Phone 7 device produced by Nokia, the Lumia 800 (Wikipedia, 2012)

An updated version of Windows Phone 7, Mango, was released in 2011. The update included a mobile version of Internet Explorer 9 along with improved multitasking capabilities and other improvements. Another minor update known as Tango was released in 2012, and lowered the hardware requirements of Windows Phone to allow devices with 800 MHz processors and 256 MB of RAM to run the operating system. (Wikipedia 2014b)

The last update to Windows Phone 7 was Windows Phone 7.8, which added some features from Windows Phone 8, such as an updated start screen, new color schemes and additional lock screen wallpaper options. The main reason for the release of Windows Phone 7.8 was that older Windows Phone devices wouldn't be upgradeable to Windows Phone 8 due to hardware limitations. (Wikipedia 2014b)

3.1.3 Windows Phone 8 and 8.1

Windows Phone 8 was released in late 2012. Windows Phone 8 uses a kernel based on Windows NT instead of the Windows CE based one previously used. This change was made so that more code could be shared directly between Windows 8 and Windows Phone 8. (Wikipedia 2014b)

Windows Phone 8 brought an updated home screen to Windows Phone, which allows tiles to be resized between the small, medium and large tile sizes. This allows the user to customize his or her phone more than before. Other updated features include new lock screen options, better multi-tasking, Skype-integration, and NFC support among others. (Rubino 2012)

Windows Phone 8.1 takes the customization even further, allowing users to set a background image for their live tiles instead of the flat colors used before. Highlighted features include Cortana, a voice detecting assistant application, and a new action center which allows users to see their notifications and change common settings quickly. Many other smaller improvements are also included, including an updated calendar, a new version of Internet Explorer, and an improved search feature, among many others. (Microsoft 2014)

Picture 4. The Nokia Lumia 930, a Windows Phone 8.1 device (Wikipedia 2014)

3.2 Application development

Windows Phone 7 applications can be developed using either Silverlight or XNA. For Windows Phone 8.1, Silverlight will be replaced by the Windows Runtime, which is also used in Windows 8 Store applications, in order to share more code between PC and Windows Phone applications. Silverlight applications made for Windows Phone 7 and Windows Phone 8 work on Windows Phone 8.1, but Windows Phone 8 and Windows Phone 8.1 applications don't work on Windows Phone 7. Windows Phone 8.1 applications also do not work on Windows Phone 8. (Wikipedia 2014b)

Silverlight is a compact version of Windows Presentation Foundation, and therefore the same techniques used when programming any .NET-based applications can be used in Windows Phone applications. Programming is done in either C# or Visual Basic and the user interface is designed in XAML, which is quite similar to writing a website using HTML. (Järvinen 2012. 19, 97-98)

XNA is mainly used for programming games. The main difference between XNA and Silverlight is that there are no ready-made user interface components in XNA, and the whole application must be programmed in either C# or Visual Basic in a more primitive way. (Järvinen 2012. 188)

3.2.1 C# and XAML

The C# programming language was created by Microsoft to respond to some specific and important needs in the programming community. It is an evolution of the C and C++ languages and is specifically intended to work with the .NET platform. The C# language has been designed to incorporate many of the best features from other languages, while clearing up their problems. (Nagel et al. 2010. 8-9)

Application development using C# is easier than with C++, because the syntax is more simple. C# is still a powerful language, and there are very few things you'd rather do in C++ because they can't be done in C#. Unlike C++, C# is type-safe, which means that once some data has been assigned to a type, it cannot transform itself into another unrelated type, which makes the code more robust and debugging more simple. (Nagel et al. 2010. 8-9)

XAML is a markup language similar to XML used to design user interfaces. It was introduced with the Windows Presentation Foundation (WPF) to replace the old Windows Forms in Windows desktop applications. Now, XAML is used for designing user interfaces in desktop applications, Windows 8 Store applications and Windows Phone applications among other types of .NET applications. (Lecrenski et al. 2012. 50-51)

3.2.2 Visual Studio and Expression Blend

Visual Studio is an integrated development environment from Microsoft. It is the ultimate tool for developing applications for all Microsoft platforms, including Windows 8 and its predecessors, .NET Framework, Windows Phone, Silverlight, and Windows Azure. There are several different versions of Visual Studio for different purposes: Professional, Premium, Ultimate, and Express. The Express version is split into different subversions for different platform, and is free. (Babaian et al. 2012. 91)

Expression Blend is a tool for user interface designers, in which they can produce unique application pages using the XAML or HTML technologies. It's not just a graphical design tool though, it can also be used for designing and creating full applications too. (Babaian et al. 2012. 108)

Visual Studio and Blend are usually used in tandem. Developers work in Visual Studio to create the basic user interface and write the logic behind the application, and designers use Blend to enhance the user interface with rich animation effects, media and nice layouts. (Babaian et al. 2012. 115)

3.2.3 LINQ

LINQ is a programming methodology that transforms the relationship between programs and data. LINQ defines a .NET application programming interface and set of extensions to the Visual Basic and C# languages that enables querying diverse data types with a single syntax similar to SQL. Writing queries using LINQ enables strong typing and productivity enhancing features, such as statement completion and IntelliSense. LINQ can be used to query data from collections of objects which are in memory, tables in SQL servers, XML-files, and various other data sources. LINQ permits custom extensions, and can therefore theoretically support anything, if someone writes a suitable extension for it. (Jennings 2009. 5)

4 THE APPLICATION PROJECT

This thesis is based around the application DiscGolf for Windows Phone. The first version of the application was developed during a Windows Phone beginner's course together with two other software developers. Initially, the goal of the project was to get acquainted with mobile development and Windows Phone in general.

4.1 The problem

During the Windows Phone –course, each team had to make an application. A Disc Golf scorecard seemed simple enough, but would still contain most of the basic components used in any mobile applications. Since similar applications were available for other platforms, but not Windows Phone, it was decided that this application would be the first one in the Windows Phone Store.

As the name implies, disc golf is a game very similar to golf. The main difference is that instead of hitting a ball into a hole, players throw a disc into a basket. The player who reaches the basket with the least throws wins.

The problem with the game is remembering your scores. After a few rounds, you have to start writing them somewhere to keep track of who is winning. Carrying pens and paper around with you is quite inconvenient, not to mention sharing and storing the scores. Besides, pretty much everyone has a smart phone these days, so why not make an application to handle the scorekeeping?

4.2 The solution

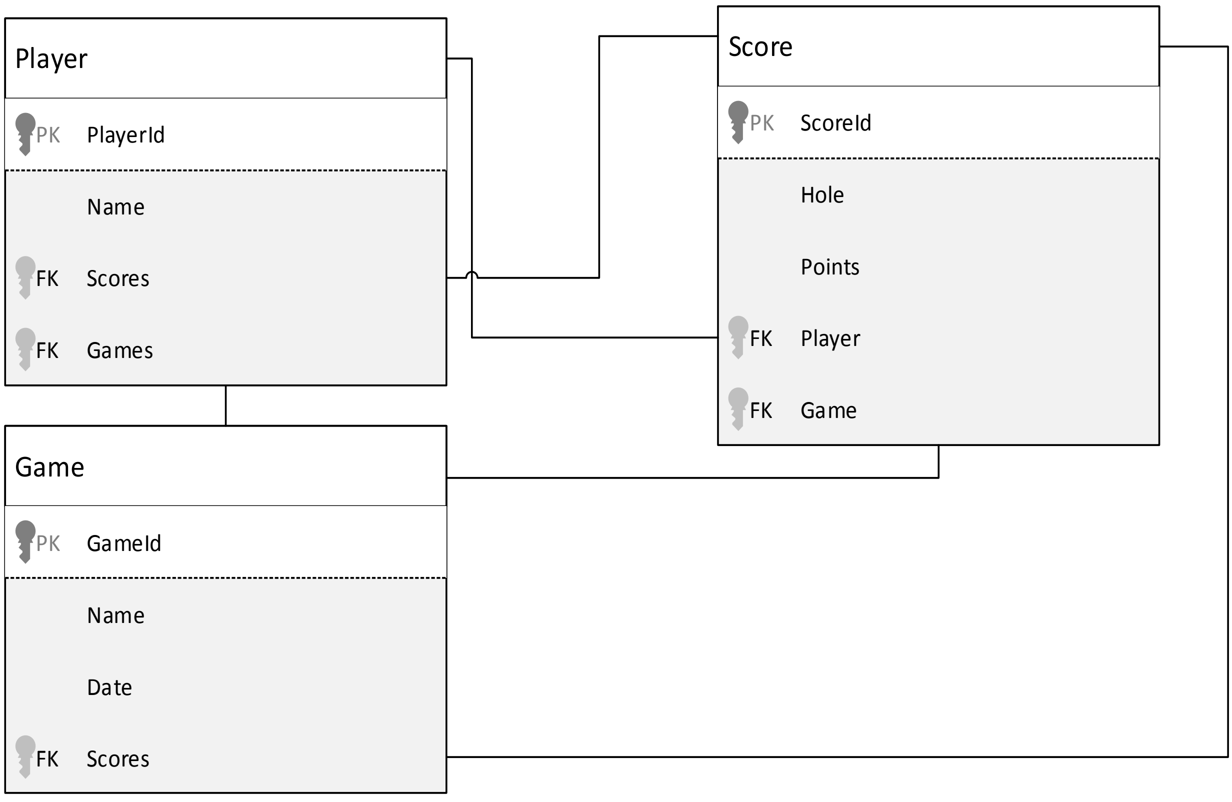
Picture 5. DiscGolf for Windows Phone

Disc Golf is a scorecard application for Windows Phone. In the application, users can add an infinite amount of players who can play an infinite amount of games on an infinite amount of courses. The Finnish version of the application, FrisbeeGolf Pro, also contains the addresses and other details about every course in Finland, and lets the user navigate to them from their current location. The course list can also be updated from the internet. All data except the courses are stored in a database, which is the main topic of this chapter.

4.3 Planning and designing the database

In order to support an infinite amount of players, games, scores, and courses, it was decided that the data will be stored in a database. Courses, however, are stored in an XML-file in order to maintain compatibility with the official course list that gets updated from the internet.

Since the team had no prior experience with databases, this one was designed to be as simple as possible. It contains one table for games, one for players and one for scores, as shown in the picture below.



Picture 6. The relational diagram of the database

On Windows Phone, database tables are written the same way as normal classes. The whole class is prefixed with [Table], and attributes to be written to the database are prefixed with [Column]. The prefixes can also contain options like

CanBeNull and others. A simple database table class without any relations might look like this:

```
[Table]
public class Score
{
    [Column(
        IsPrimaryKey = true,
        CanBeNull = false,
        IsDbGenerated = true,
        AutoSync = AutoSync.OnInsert)]
    public int scoreID { get; set;}
    [Column(CanBeNull = false)]
    public int hole { get; set;}
    [Column(CanBeNull = false)]
    public int points { get; set; }

}
```

Here, the “CanBeNull = false” is actually unnecessary since an integer-type value can’t be null anyway, but it doesn’t break anything, so it was just left in place. The “IsPrimaryKey = true” defines that this column is the primary key of the table and “IsDbGenerated = true” defines that the database engine should automatically generate this value. “AutoSync = AutoSync.OnInsert” tells the database engine to update this value once the object has been inserted into the database. This is useful when a value is generated by the database, because otherwise we would need to query the object from the database to be able to identify it by its ID.

The most complicated part of designing the database are the relations. Luckily the code is the same everywhere, so once all the different parts have been figured out, the previous relations can just be copy-pasted, simply changing the names and data types. Relations to other tables are prefixed with [Association]. The foreign key is usually written to the database as a normal column. In addition to the foreign key, a storage-object is needed for the relation to work. A relation might look like this:

```
[Column]
internal int _playerID;
private EntityRef<Player> _player;
[Association(
    Storage = "_player",
    ThisKey = "_playerID",
    OtherKey = "playerID",
    IsForeignKey = true)]
public Player player
{
    get
    {
        return _player.Entity;
    }
    set
    {
        _player.Entity = value;
        if (value != null)
        {
            _playerID = value.playerID;
        }
    }
}
```

The Storage-option defines which EntityRef<Player>-object the associated object should be stored in. ThisKey is the foreign key to be stored in this table, and OtherKey is the same value in the table from which the associated object is queried. “IsForeignKey = True” defines that a foreign key is used to identify the object.

In this case, a player is linked to a score. In order to be able to query a players’ scores, the relation also needs to be defined in the Player-class. Since a player can have many scores, the Player-class needs to have an EntitySet<Score>-object (which is essentially an array of Score-objects) to store the data. This part of the relations looks like this:

```
private EntitySet<Score> _scores;
[Association(Storage = "_scores",
             OtherKey = "_playerID",
              ThisKey = "playerID")]
public EntitySet<Score> Scores
{
    get
    {
        return this._scores;
    }
    set
    {
        this._scores.Assign(value);
    }
}
```

In addition to defining the list of scores, attach and detach methods need to be defined. In this case, these methods look like this:

```
private void attach_Score(Score score)
{
    score.player = this;
}

private void detach_Score(Score score)
{
    Score.player = null;
}
```

Finally, the methods need to be assigned to the relation. This is done in the classes' constructor:

```
public Player()
{
    _scores = new EntitySet<Score>(
    new Action<Score>(this.attach_Score),
    new Action<Score>(this.detach_Score)
    );
}
```

4.4 Writing the database queries

Since the database schema is fairly simple and not very thought out, some of the queries ended up being quite complicated. For example, in order to get all the players for a certain game, all the scores for the first hole in a game have to be queried in order to get the players from those records. A more logical way to handle this would be to list the players under each game, or to make another table with relations to players and games.

Before querying any data, a database context must be initialized and the database needs to be created if it doesn't exist:

```
FrisbeeDataContext context = new FrisbeeDataContext(@"isostore:/dgdb.sdf");

 if (!context.DatabaseExists())
{
    context.CreateDatabase();
}
```

This initializes a database context and creates a new database named dgdb.sdf in the applications own Isolated Storage folder. Only the application itself can access files stored in its Isolated Storage. After creating a brand new database, it's a good idea to add the PAR-player which is needed in all games:

```
Player player = new Player{ Name = "PAR" };
context.Players.InsertOnSubmit(player);
context.SubmitChanges();
```

Once the player has been added, it can be acquired from the database using the following LINQ-statement:

```
(from p in context.Players
where p.Name == "PAR"
select p).FirstOrDefault();
```

Or this lambda-expression:

```
context.Players.Where(p => p.Name == "PAR").FirstOrDefault();
```

The FirstOrDefault()-extension is what actually queries the data. It returns the first matching object of the given type, or the default value (in this case null) if no objects match the criteria. Without it, a database query object would be returned. FirstOrDefault() can be replaced with any LINQ-extension, for example ToList(), which returns a list of objects and is useful when multiple objects match the given criteria, or Any(), which returns a Boolean value of true if any objects match the criteria and false if none match it.

Getting one player by name is simple, but the application also contains more complicated queries. For example, to get a certain players' score for a certain hole in a certain game, the following query is issued:

```
(from c in context.Scores
where
((c.game != null) ? (c.game.GameID == game) : false)
&&
((c.player != null) ? (c.player.PlayerID == player.PlayerID) : false)
&&
(c.hole == hole)
select c).FirstOrDefault();
```

There are three conditions in this query. The GameID, the PlayerID and the hole must all match in order to be selected. In addition to checking if the values match, checks have been added to make sure that the queried Game and Player objects are not null, as trying to access values inside non-initialized objects would cause NullReferenceExceptions which must be handled to prevent the application from crashing.

4.5 Version 2.0: Schema update

In version 2.0, a new score view which shows the best scores per course was introduced. The scores-table didn't contain a record of which course the game was played on, so this had to be added in order to be able to filter the scores. For this exercise, DatabaseSchemaUpdater is used. The code looks like this:

```
DatabaseSchemaUpdater schemaUpdater =
         context.CreateDatabaseSchemaUpdater();
int version = schemaUpdater.DatabaseSchemaVersion;
if (version == 0)
{
    schemaUpdater.AddColumn<Score>("Course");
    schemaUpdater.DatabaseSchemaVersion = 1;
    schemaUpdater.Execute();
}
```

The course-attribute also needs to be added into the score model-class:

```
[Column]
public string Course { get; set; }
```

The DatabaseSchemaUpdater-code is called when the database is initialized and adds the gameCourse-column to the database. The schema version is zero by default, and after updating it, it is incremented to one in order to prevent trying to do the same update again. Trying to add a column that already exists in the database results in an exception.

If the user launches DiscGolf 2.0 without having used an older version, the database will already be created with the updated schema, and therefore the schema version number needs to be updated to one using the same code, but without adding the column:

```
context.CreateDatabase();
DatabaseSchemaUpdater schemaUpdater =
        context.CreateDatabaseSchemaUpdater();
schemaUpdater.DatabaseSchemaVersion = 1;
schemaUpdater.Execute();
```

This, again, ensures that we don’t try to add a column that already exists.

5 FUTURE CONSIDERATIONS

In the future, Microsoft might drop support for Silverlight altogether. In this case, the application will have to be ported to the Windows Runtime, which brings a set of new hurdles along with new possibilities. For example, an application's Isolated Storage is accessed in a similar way to accessing a file on the desktop version of Windows, instead of using the Isolated Storage classes used in Silverlight.

So far, databases themselves seem to be unaffected, but given the fact that Windows 8 Store-apps don't support local databases at all, it is possible that later Windows Phone releases won't either and the data will have to be serialized into text files instead.

Another thing that has been considered is synchronizing the database into a user's OneDrive. This would allow the user to switch devices while still retaining the data from their previous device. There are many things to consider when doing this though: What if a user downloads an older or completely different database into their phone? Will the data be merged somehow or will the old database be overwritten completely? What if the user puts a broken database into their OneDrive and tries to synchronize that? Those scenarios, probably along with dozens of others, will have to be taken into consideration.

6 CONCLUSIONS

The goal of this thesis was to explain how a locally stored database can be used in a Windows Phone application. As seen above, the task is not as complicated as one might think.

If one is familiar with .NET programming in general, Windows Phone programming should feel quite natural apart from designing the user interfaces and taking a few limitations into consideration. Designing a database for Windows Phone is basically like writing a bunch of model classes that represent the data to be stored, the only complicated part being the relations. The data is then queried from the database using LINQ, which is very similar to SQL. Data is stored into the database using commands that are not much more complicated than ”store data x into database y”.

The author personally learned a lot from this project. A scorecard is a very suitable application to start programming with, as it is relatively simple, but can contain all the typical components used in all kinds of applications. After this project, the author was ready to take on more complicated application projects and eventually start his own software company with a few partners.

REFERENCES

Amy 2010. A Brief History of Windows Mobile. Referenced 01.06.2014 http://notebooks.com/2010/04/12/a-brief-history-of-windows-mobile/

Babaian, Ani Novák, Istvan Arvai, Zoltan 2012. Beginning Windows 8 Application Development. Somerset, NJ, USA: Wiley.

Jennings, Roger 2009. Professional ADO.NET 3.5 with LINQ and the Entity Framework. Hoboken, NJ, USA: Wrox.

Järvinen, Jani 2012. Windows Phone sovelluskehitys. Helsinki: Docendo Oy.

Kriegel, Alex 2011. Discovering SQL : A Hands-on Guide for Beginners. Hoboken, NJ, USA: Wrox.

Lecrenski, Nick Lecrenski, Stephen Ashley, Kevin 2012. Professional Windows 8 Programming : Application Development with C# and XAML. Somerset, NJ, USA: Wiley.

Microsoft 2014. What's new in Windows Phone 8.1. Refenced 02.06.2014 http://www.windowsphone.com/en-us/how-to/wp8/basics/whats-new-in-windows-phone

Pallaw, Viljay Krisha 2010. Database Management Systems. Delhi, India: Global Media.

Rubino, Daniel 2012. Overview and review of Windows Phone 8. Referenced 02.06.2014 http://www.wpcentral.com/overview-and-review-windows-phone-8

Sharma N, Perniu L, Chong R, Iyer A, Nandan C, Mitea A, Nonvinkere M, Danubianu M 2010. Database Fundamentals. Canada: IBM.

Stephens, Rod 2009. Beginning Database Design Solutions. Hoboken, NJ, USA: Wiley.

Watson, Karli Nagel, Christian Pedersen, Jacob Hammer 2010. Beginning Visual C# 2010. Hoboken, NJ, USA: Wrox.

Wikipedia 2014a. Windows Mobile. Referenced 01.06.2014 http://en.wikipedia.org/wiki/Windows_Mobile

Wikipedia 2014b. Windows Phone. Referenced 02.06.2014 http://en.wikipedia.org/wiki/Windows_Phone