Windows Azure ile Cloud Computing Uygulamaları – 7

Zaman içerisinde planlı ya da plansız olarak, birçok işlem yapılmaktadır. Günümüzde sosyal ağlar ile paylaşımlar ya da bankalar ile birçok harcamalar yapmaktayız. Yapılan işlemler ile kullanıcılar için önemli olmayan birçok işlem alışkanlıkları, şirketler için CRM bilgisi olarak depolanmaktadır.

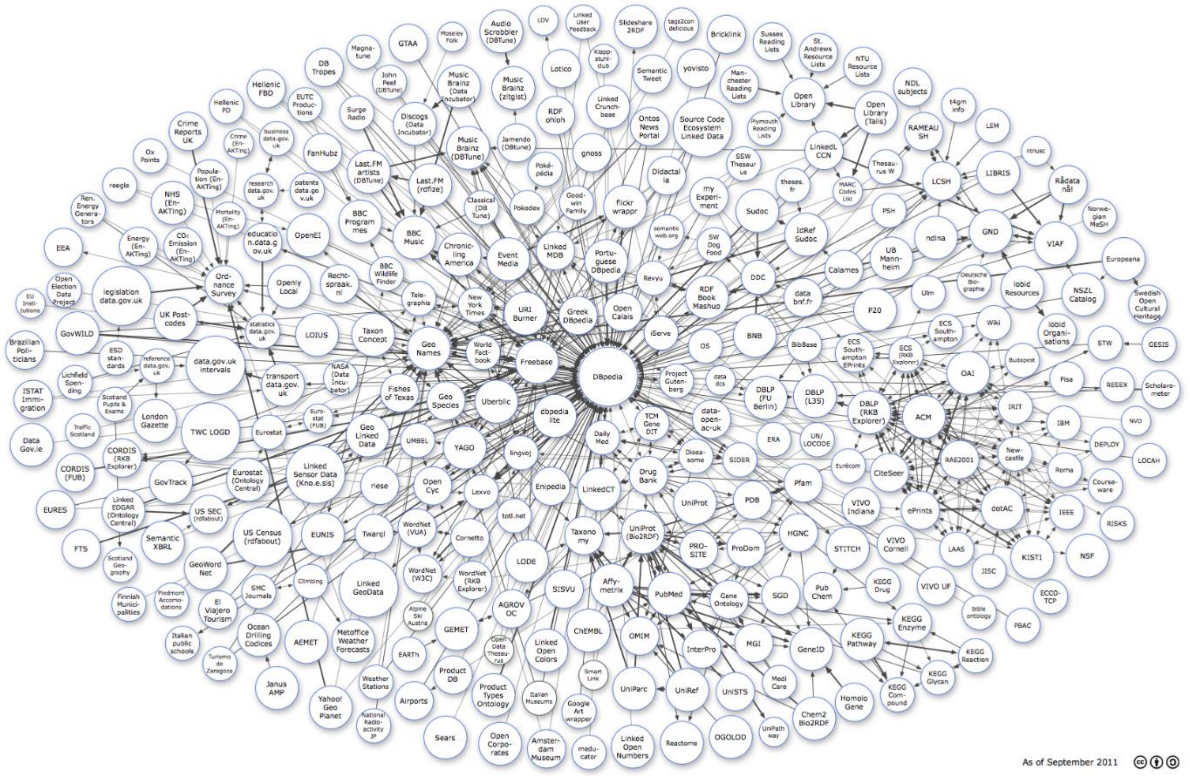
Günümüzde müşteri isteklerini, talep etmeden hazırlayan şirketler kazanmaktadır.

Yeni nesil CRM süreçleri, her yeni gün özellik kazanarak şekillenmektedir. Söz konusu özellikler, CRM sürecin gerçekleştirilmiş olduğu, ülkenin belirtmiş oluğu yasal yaptırımlara göre gerçekleştirilmektedir.

Şirketler, iş planlama ve uygulama senaryolarını, istek ve yasal yaptırımlara uyumluluk sağlama amacı ile güncellemektedir. Geçmiş süreç de gereksinim bulunmayan özellikler, gelecek dönem de istenebilmektedir. Gerçekleşen değişimler, kullanılan iş uygulamalarının şekillendirilmesi ya da yeniden geliştirilmesi gündeme gelebilmektedir.

Türkiye Cumhuriyeti'nde, 1999 yılın sonu itibari ile vatandaşlık numarası uygulaması başlatılmıştır. Uygulama ile isim benzerliklerinde kaynaklanan problemler ve tüm işlemlerin tek numara üzerinde gerçekleştirilmesi amaçlamıştır. Çalışma ile başta sağlık sektörü olmak üzere, insan ile ilişkili tüm sektörlerde kullanılan uygulamalarda, vatandaşlık numarasını temel alma özelliğinin kazandırılması gereksinimi ortaya çıkmıştır. Yapılan işlemler ile tamamlanmış mevcut ya da süren işlem verilerinin düzenlenmesi gereksinimi ortaya çıkmıştır.

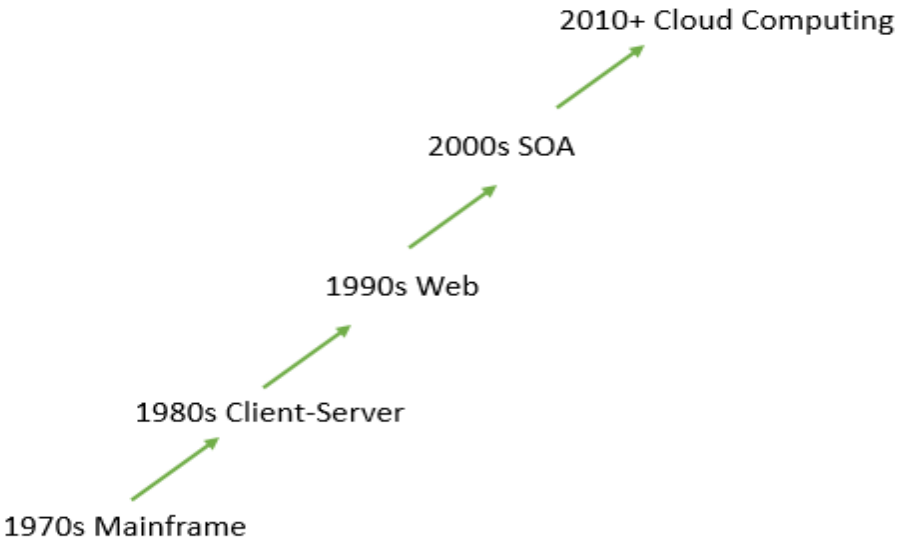
İş süreçleri, iş kural ya da yasal yaptırımların uygulanması sebebi ile işlem girdi ve çıktılarının değişiklik göstermesi gündeme gelebilmektedir. Yapılan işlemler, mevcutta verilerin düzenlenmesi, performans ve depolama süreçlerini de etkilemektedir.



İş uygulamaları, kullanıcılarından yazılı ya da Binary türünde çeşitli veriler almakta ya da üretilmektedir. Meydana gelen veriler, zaman içerisinde veri yığınların oluşmasına neden olmaktadır. Veri yığınları ise, zaman ile büyük veri(**Big Data**) olgusunu gündeme getirmektedir.

Her şirketin çalışmış olduğu ana iş dalı üzerinde çalışması, konu ile ilgili uzmanlaşmasını sağlayacaktır. Gereksinimleri duyduğu diğer yan ihtiyaçları, konunun uzmanlarında sağlaması, iş sürecini ve üretimin kalitesini artıracaktır.

Şirketler sahip oldukları veri yığınlarını, depolama, analiz ve yönetim için çeşitli iş uygulamaları kullanmaktadır. Veri yığınları ile ilgili işlem yapılabilmek için edinilen araçlar, zamanla her şirketlerin birer veri merkezinin oluşmasına neden olmaktadır. Veri merkezlerin kurulması, yönetilmesi, bakımı ya da nitelikli iş gücünün oluşturulması, şirketlerin iş kolları dışında ağır yükler altına girmesine neden olmaktadır.



Günümüzde sektör ve iş kolların çeşitlenmesi, iş ihtiyaçlarının ve gereksinim duyulan verinin artmasına neden olmaktadır. İş gereksinimleri sağlanması ve istenilen sonuçların üretilebilmesi için uygulamalar, **Cloud Computing** saylayıcılara ve özellikle **Microsoft Windows Azure Platform** 'a taşınmaktadır.

İş uygulamalarının, gerçekleştirilmesi istenen süreci en avantajlı gerçekleştirilmesi amacı ile çeşitli teknolojik bağımlıklara yüklenmektedir. Çeşitli nedenlerden dolayı, değişen iş süreçleri, kullanılan iş uygulamalarının da güncellemesi gereksinimini ortaya çıkartmaktadır. İş Uygulamaları, iş süreçleri ile ilgili çeşitli avantajlar sunarken, değişim süreçlerinde handikapların oluşmasına neden olabilmektedir.

Uygulama mimarileri, iş süreç ve yasal bağımlıkları nedeni ile kullanıcı tarafından alınan ve üretilen verilerin güncellenmesi gündeme gelebilmektedir. Uygulama veri yapılarının, zaman ve ihtiyaçların gereğince değişmesi, uygulama performansı ya da veritabanı optimizasyonu gibi problem oluşturabilmektedir. Geçmiş dönemde gereksinim olduğu düşünülen veriler, farklı bir dönemde gereksinim duyulmayabilir. Benzer durum olarak, bazı verilere de gereksinim duyulabilmektedir. Yapılan işlemler, verinin depolanması ya da analiz süreçlerini etkileyebilmektedir.

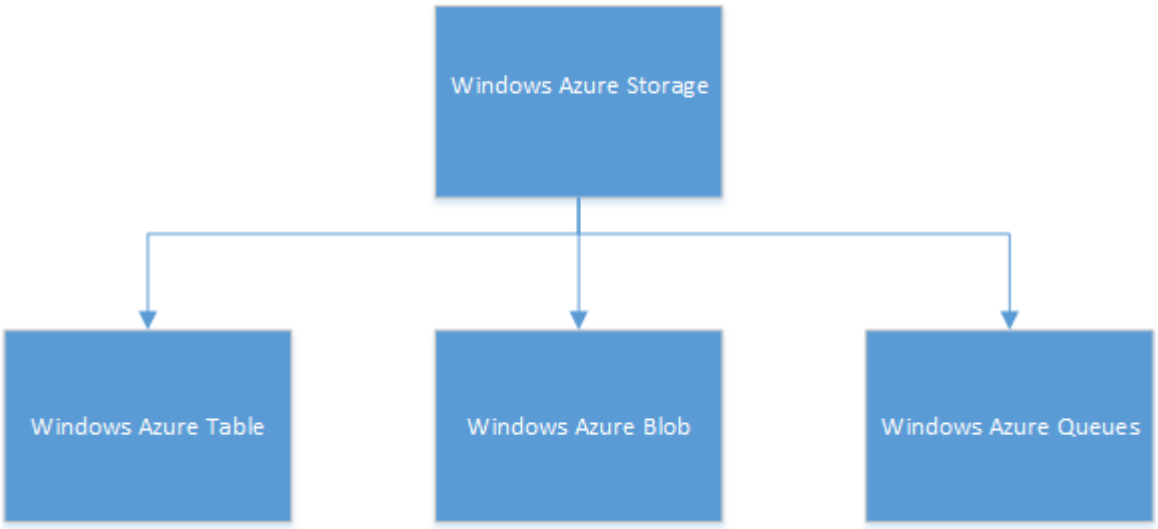
No	Name	SurName	Status	Title	BirthDate	Phone
1	Davolio	Nancy	FALSE	*	*	*
2	Fuller	Andrew	FALSE	*	*	*
3	Leverling	Janet	FALSE	Sales Representative	*	(206) 555-8122
4	Peacock	Margaret	FALSE	Sales Representative	*	(206) 555-1189
5	Buchanan	Steven	TRUE	Sales Manager	4/3/1955	*
6	Suyama	Michael	TRUE	Sales Representative	2/7/1963	*

Gelişen teknoloji, iş süreçlerinin kolay, isteklere hızlı cevap veren iş çözümleri sunmasına olanak sağlamaktadır. Günümüzde iş uygulamalarındaki veri yapıların hızlı sorgulama ve düzenleme gereksinimi ortaya çıkmıştır. Ortaya çıkan gereksinimler, veri yapılarının esnek olma gereksinimini doğurdu. 2009 yılın da "*eğlenceyi seç, ilişkisel=yanlış; olan gerçek Dünyadan faydalan*" sloganı ile yapılan, "*no:sql(east)*" konferansı **NoSQL** konusunda çeşitli düşüncelerin oluşmasına büyük etki oluşturmuştur.

***NoSQL**, 1998'un sonlarına doğru ortaya çıkan bir kavramdır. Klasik ilişkisel veritabanı yapışanda bulunmayan ve sorgulama için SQL dili kullanmayan veritabanı türüdür. Genel olarak xml ya da JSON formatında veri depolama yapmaktadır.*

Teknolojik değişim, veri yapılarının esnek olma gereksiniminden çıkarak, hizmet olarak sunulma sürecini beraberinde getirmiştir. **Microsoft** ve diğer **Cloud Computing** sağlayıcıları çeşitli depolama hizmetleri ile kullanıcılarına **Cloud Storage** hizmetleri sunmaktadır. **Microsoft Windows Azure Platform** ile kullanıcılarına klasik ilişkisel(**SQL Database**) veritabanı hizmeti sunduğu gibi No-Relation(**Windows Azure Storage**) iş çözümü de sunmaktadır.

Windows Azure Platform No-Relation iş çözümü olarak, **Windows Azure Storage** hizmeti sağlamaktadır. Hizmet ile Binary ve diğer veri türleri farklı nesneler içerisinde güveli ve yüksek optimizasyonu değerlerinde, depolanması sağlanmaktadır. Aşağıda **Windows Azure Storage** hizmeti katmanları belirtilmiştir.



Windows Azure Storage hizmeti, üç parçalı ve parçalara özgü özelliklerin olması sebebi ile anlatımın devamında **Windows Azure Storage** 'ın parçalarından olan **Windows Azure Table Storage** ile ilgili bilgiler verilecektir.

Windows Azure Table Storage, **SQL Database** yaklaşımında bulunan **Table** nesnesine benzemektedir. Ama yapısal bazı özgü özelliklerden dolayı, **Windows Azure Table Storage** yapısı **SQL Database**'de bazı farklıklar göstermektedir. Aşağıda **Windows Azure Storage** ve **SQL Database** ile ilgili bilgi ve karşılaştırmalar bulunmaktadır.

Özellik	Windows Azure Table Storage	SQL Database
İlişkisel veri	Desteklemiyor	Destekliyor
Server-side çalışama	Desteklemiyor	Destekliyor
Geo-replication	Destekliyor	Desteklemiyor
Scale-out	Otomatik	Manuel
LINQ desteği	Destekliyor	Destekliyor
Veri erişimi	OData Protokol	ODBC ya da JDBC
Yönetim protokollü	HTTP/HTTPS üzerinde REST	HTTP/HTTPS üzerinde REST ile ODBC/JDBC
En az veri depolama	1MB	2GB
En fazla veri depolama	100TB Tablo başına	150GB Tablo başına
Firewall Güvenliği	Desteklemiyor	Destekliyor

REST protokol ile erişim	Destekliyor	Destekliyor
Transaction	Destekliyor (Limitli)	Destekliyor
Transaction logs tutma	Desteklemiyor	Destekliyor

Windows Azure Table ile çalışmak, veritabanı işlemlerini **Entity Framework** ile yapmak kadar kolay, işlevsel ve hızlıdır. **Windows Azure Table Storage**, veri işlem süreçlerin de **LINQ** sorgularını kullanılabilmektedir. **Entity Framework** ile geliştirilen uygulamalarda **DbContext** nesnesi kullanıldığı gibi **Windows Azure Table Storage** yapısında ise, **TableServiceContext** nesnesi kullanılmaktadır.

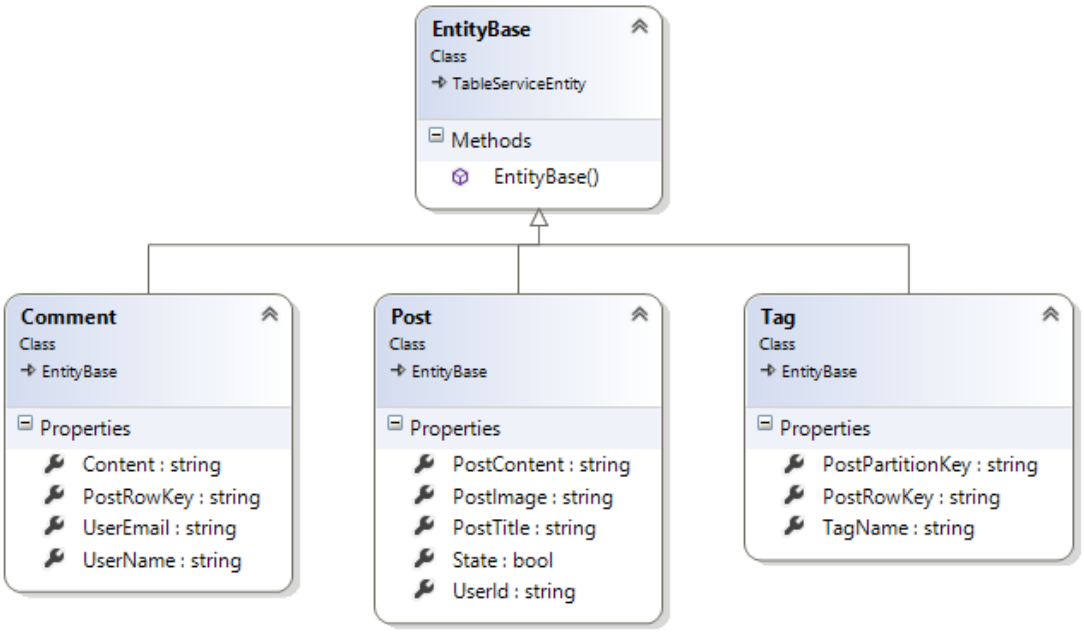
```
public class FunnyAppContext : TableServiceContext
{
    public FunnyAppContext(string baseAddress, StorageCredentials credentials)
        : base(baseAddress, credentials)
    {
    }

    public IQueryable<Tag> Tag
    {
        get { return this.CreateQuery<Tag>("Tag"); }
    }

    public IQueryable<Post> Post
    {
        get { return this.CreateQuery<Post>("Post"); }
    }

    public IQueryable<Comment> Comment
    {
        get { return this.CreateQuery<Comment>("Comment"); }
    }
}
```

Uygulama örneği olan, "*WindowsAzure.FunnyApp*" çalışması nesneye yönelimli olarak geliştirilerek, gerçek yaşamda karşılaşabilecek durumlar ele alınmaya çalışılmıştır. Aşağıda "*WindowsAzure.FunnyApp*" çalışması ile ilgili sınıf ilişkisel şeması bulunmaktadır.



Yukarıdaki şema da görüldüğü gibi **EntityBase** sınıfı **TableServiceEntity** sınıfında türetilmiştir. Yapılan türetme ile **Windows Azure Table Storage** Entity fonksiyonelliği kazandırılmış olacaktır. Temel sınıf olan **EntityBase** sınıfından türeyen sınıflar, "**PartitionKey**" , "**RowKey**" ve "**Timestamp**" özellikleri kazanmaktadır. Söz konusu özellikler, **Windows Azure Table Storage** içerisinde verilerin, benzersiniz("**PartitionKey**" , "**RowKey**") ve işlem zamanı("**Timestamp**") ile ilgili bilgiler taşımaktadır.

```
public class EntityBase : TableServiceEntity
{
    public EntityBase()
    {
        this.PartitionKey = DateTime.UtcNow.ToString("MMddyyyy");
        this.RowKey = string.Format("{0:10}_{1}", DateTime.MaxValue.Ticks -
 DateTime.Now.Ticks, Guid.NewGuid());
    }
}
```

Klasik uygulama veritabanı uygulama geliştirme sürecinde gerçekleştirildiği gibi **Windows Azure Table Storage** ile de GRUD işlemleri yapılabilmektedir. Yapılan işlemlerin bir kalıp üzerinde gidebilmesi amacı ile Repository tasarım deseni kullanılmıştır.

```
public interface IRepository<TEntity>
{
    TEntity Find(string partitionKey, string rowKey);
    TEntity Find(string rowKey);
    void Create(TEntity entity);
    void Delete(TEntity entity);
    void Update(TEntity entityToUpdate);
    void SubmitChange();
    IQueryable<TEntity> Get();
}
```

Windows Azure Table Storage nesnesi ile veri işlemlerinin yönetebilmesi amacı için Repository nesnesinin yönetilebildiği FunnyAppRepository sınıfı hazırlanmıştır. Sınıf **EntityBase** türüne ait, nesnelere hizmet vermek amacı ile sınırlandırılmıştır.

```
public class FunnyAppRepository<TEntity> : IRepository<TEntity>,
    IDisposable where TEntity : EntityBase
{
    // veri işlemleri ile ilgili hesap
    // bilgilerinin alınması
    private static CloudStorageAccount _storageAccount;

    // iş nesnesi ile ilgili tanımlamanın yapılması
    private readonly FunnyAppContext _context;

    // nesne türüne ayit isim alınması
    private readonly string _entitySetName;

    public FunnyAppRepository()
    {
        // veri işlemleri ile ilgili gerekli bilgilerin alınması
        _storageAccount =
            CloudStorageAccount.FromConfigurationSetting(Utils.ConfigurationString
);

        // veri hesabı bilgilerine bağlı olarak,
        // veri nesnelerinin Windows Azure Storage üzerinde oluşturulması
        CloudTableClient.CreateTablesFromModel(
            typeof (FunnyAppContext),
            _storageAccount.TableEndpoint.AbsoluteUri,
            _storageAccount.Credentials);

        // seçili olan nesne türünün isminin alınması
        _entitySetName = typeof (TEntity).Name;

        // seçili olan nesne ile ilgili gerekli tanımlamaların
        // Windows Azure Storage içerisinde olup/olmadığını kontrollünün yapılması
.
        // Söz konsu nesnenin Windows Azure Storage içerisinde olmaması durumunda
        // nesne isimine bağlı olarak, gerekli tanımlamaları yapılması
        _storageAccount.CreateCloudTableClient().CreateTableIfNotExist(_entitySetN
ame);

        // GRUD işlemlerinde sorumlu olan, iş nesnesine erişim ile ilgili
        // gerekli bilgilerin atanması
        this._context = new FunnyAppContext(_storageAccount.TableEndpoint.Absolute
Uri,
            _storageAccount.Credentials);
        this._context.RetryPolicy = RetryPolicies.Retry(3, TimeSpan.FromSeconds(1)
);
    }

    // nesne türüne bağlı olarak
    // rowkey ve partitionkey ile kayıt getirme
    public TEntity Find(string partitionKey, string rowKey)
    {
        return (from g in _context.CreateQuery<TEntity>(_entitySetName)
                where g.PartitionKey == partitionKey && g.RowKey == rowKey
                select g).FirstOrDefault();
    }
```

```
    // nesne türüne bağlı olarak
    // rowkey ile kayıt getirme
    public TEntity Find(string rowKey)
    {
        return (from g in _context.CreateQuery<TEntity>(_entitySetName)
                where g.RowKey == rowKey
                select g).FirstOrDefault();
    }

    // nesne türüne ayit yeni üye oluşturma
    public void Create(TEntity entity)
    {
        this._context.AddObject(_entitySetName, entity);
    }

    // nesne silmek için
    public void Delete(TEntity entity)
    {
        this._context.DeleteObject(entity);
    }

    // nesne güncellemek için
    public void Update(TEntity entityToUpdate)
    {
        this._context.UpdateObject(entityToUpdate);
    }

    // yapılan değişikleri depolama alanına yansıtmak için
    public void SubmitChange()
    {
        this._context.SaveChanges();
    }

    // nesne türüne ayit tüm içerikleri çekebilmek için
    public IQueryable<TEntity> Get()
    {
        return this._context.CreateQuery<TEntity>(_entitySetName);
    }

    public void Dispose()
    {
        GC.SuppressFinalize(this);
    }
}
```

Tanımlaması yapılan iş nesnelerinin, kullanımı ile uygulama senaryosu olan, Kullanıcın tarafında yüklenen fotoğraf içeriklerin görüntülenme süreci gerçekleştirilmektedir. Yapılan işlem ile **LINQ** sorgusu kullanarak, gerekli bilgiler edinilmektedir.

```
Protected void Page_Load(object sender, EventArgs e)
{
    if (IsPostBack) return;
    // Post nesne türüne ayıt, repository iş nesnesinin tanımlanması
    FunnyAppRepository<Post> _postRepository = new FunnyAppRepository<Post>();
```

```
    List<PostViewData> viewDatas = new List<PostViewData>();

    // istenilen öncüllerin oluşturulduğu Linq sorgusu
    _postRepository.Get().Where(post => post.UserId == Membership.GetUser(Page.Use
r.Identity.Name)
        .ProviderUserKey.ToString()).ToList()
                    .Where(post=> post.State)
                    .OrderByDescending(post => post.Timestamp)
                    .Take(20).ToList().ForEach(post => viewDatas.Add(new PostViewDa
ta()
                    {
                        PostContent = post.PostContent,
                        PostImage = post.PostImage,
                        RowKey = post.RowKey,
                        UserId = post.UserId
                    }));

    this.RepeaterImages.DataSource = viewDatas;
    this.RepeaterImages.DataBind();
}
```

Her yeni gün, yeni çalışmalar ve işlemler yapmaktayız. Gerçekleştirilen süreçler, veri yığınları oluşmasına neden olmaktadır. Meydana gelen veri yığınları, güncelleme, analiz ve depolama gibi gereksinimleri ortaya çıkarmaktadır. Yaşanan problemler, ortaya çıkan **NoSQL** yaklaşımları ile amaçlanmaktadır.

Veri yapılarının hatalı analiz ve planlanması nedeni ile veri ile ilgili tüm alanlarda problemlerin ortaya çıkması mümkün olacaktır.

Not: Yapılan anlatımın örneklenmesi amacı ile "*WindowsAzure.FunnyApp*" uygulaması hazırlanmıştır. Aşağıdaki bağlantı kullanılarak, uygulama kaynak kodlarına erişebilirsiniz.

Github / https://github.com/ibrahimatay/WindowsAzure.FunnyApp

Windows Azure Storage yaklaşımı, geliştiricilerine yönetmekte oldukları verileri sınıflandırma ve planlamasına yöneltmektedir. Yönetilmesi amaçlanan verilerin **Windows Azure Storage** yaklaşıma uyumluğunun sağlanması ile esnek ve kolay yönetilebilir veri yapılarına sahip olunmaktadır. Konu ile ilgili sorularınızı info@ibrahimatay.org eposta adresine yöneltebilirsiniz.

İbrahim ATAY

TC Kimlik Numarası

http://www.nvi.gov.tr/Hakkimizda/Projeler,Mernis_Hedef.html

http://tr.wikipedia.org/wiki/T%C3%BCrkiye_Cumhuriyeti_Kimlik_Numaras%C4%B1

no:sql(east)

https://nosqleast.com/2009/

Repository Pattern

http://martinfowler.com/eaaCatalog/repository.html

Windows Azure Table Storage and SQL Database - Compared and Contrasted

http://msdn.microsoft.com/en-us/library/windowsazure/jj553018.aspx

Windows Azure Table Storage

http://www.windowsazure.com/en-us/develop/net/how-to-guides/table-services/