

The Microsoft Data Platform

David Campbell and Anil Nori

Microsoft Corporation
One Microsoft Way, Redmond, WA 98052-6399
{davidc, anilnori}@microsoft.com

ABSTRACT

Advances in hardware, storage, devices, connectivity, and web technology are changing the way applications are designed, deployed, and managed. Applications are increasingly becoming data-centric and data is everywhere, and in all tiers (from the client to the cloud). Data across multiple tiers requires data access and management capabilities across these tiers.

The Microsoft Data Platform presents a vision for an end-to-end data platform that offers data services across all tiers.

Categories and Subject Descriptors:

H.2 [Database Management], H.3 [Information Storage and Retrieval], H.4 [Information Systems Applications]

General Terms: Design, Management

Keywords: Client, Cloud, Data Platform, Device, Model-centric Data and Services, Synchronization, Tiers

1. INTRODUCTION

Most business, consumer, and entertainment media is now born and delivered in digital form. Dramatic advances in storage technology allow us to capture and store this data relatively inexpensively. Managing this data – organizing and accessing it however is becoming more of a challenge. The advances in processors, memory, storage, and connectivity have paved the way for next-generation applications that are data-driven, whose data can reside anywhere (i.e. on the server, desktop, devices, embedded in applications) and that support access from anywhere (i.e. local, remote, over the network, from devices, in connected and disconnected mode). This naturally leads to distributed, multi-tiered, and composite application architectures. Application components can run on different tiers, in different service boundaries, and on different platforms (e.g. server, desktop, devices). For example, in a typical web application, most of the application resides on a server supporting a large number of users; however, some client components of the application may run on desktops, mobile devices, and web browsers. The advances in connectivity and cheap storage combined with the complexity of software management are making way for on-line services and software-as-a-service. In such services models, applications (and

their data) are hosted in central data centers (also sometimes referred to as the ‘cloud’) and are accessible and shared over the web.

As mentioned above, data and applications can reside in different tiers with different semantics and access patterns. For example, data in back-end servers/clusters or in the cloud tends to be authoritative; data on the wire is message-oriented; data in the mid-tier is either cached data for performance or application session data; data on the devices could be local data or data off-lined from back-end sources. The data access and management needs in these tiers are different from what commercial DBMSs support. To allow cheap fault-tolerant and self-managing clusters, databases need to be built on very inexpensive, commodity hardware. In addition, in a world where hundreds of gigabytes of storage is the norm, the ability to work with most data in memory and easily shift from tables to trees to graphs of objects is the key to programmer productivity for next generation applications.

In this paper, we describe the Microsoft Data Platform vision to address data and application needs across all tiers – devices, desktops, servers/clusters, and the cloud. We describe the trends and the challenges that drive the Data Platform vision followed by the Microsoft Data Platform overview.

The rest of this paper is organized as follows. Section 2 provides a description of the trends that define the new environment for data-centric applications. Section 3 describes the Microsoft Data Platform. Section 4 provides a summary of this paper.

2. THE NEW ENVIRONMENT

The vision for the Microsoft Data Platform is driven by mega trends in hardware, storage, devices, and in web applications. In this section we highlight some of the compelling trends that have significant influence on the thinking for the Microsoft Data Platform.

2.1 Hardware Trends

64-bit CPUs and 64-bit OSs

With the maturity of 64-bit hardware, 64-bit CPUs are becoming the norm for client and server machines. True 64-bit architectures support 64-bit CPUs, 64-bit data or address buses, virtual addressability etc, and dramatically increase memory limits (to 2⁶⁴ bytes). Operating systems (e.g. Windows, Linux) have also been upgraded to support and take advantage of 64 bit address-space and large memories.

Multi-Core CPUs

Next generation CPUs are going to be multi-core processors. Dual-cores are already very common; quad-cores are expected to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGMOD '07, June 12–14, 2007, Beijing, China.
Copyright 2007 ACM 978-1-59593-686-8/07/0006...\$5.00.

be common by 2008; and by 2012, CPUs with tens of cores (if not hundreds) are likely to be the norm.

Main Memory Sizes are growing...

With the costs of memory going down, configurable memory size on desktop and server machines is increasing. For example, desktops can be configured with 16GB RAM, and servers can be configured with up to 2 TB of RAM. Large RAM does provide significant performance benefits to applications. While very large memory configurations (100s GBs) are possible, they are still prohibitively expensive compared to equivalent disk capacities. However,

NV RAMs, Flash etc.

Non-volatile memories are becoming more common – the canonical example being Flash. Flash memory is already being used instead of disks in many scenarios (cell phones and other mobile devices). Given Flash memory's attractive read-write characteristics, power consumption, and their decreasing cost, we believe it will become a key storage media.

2.2 Storage Trends

Low Cost Commodity Storage

The storage technology advances have caused significant increase in densities and decrease in cost. Large capacity disks have become a commodity. Terabyte (1 TB) disk units are available for <\$300; 10 TB disk units are expected in 5 years and the unit costs are expected to go below \$100. However, these commodity disks are considered less reliable.

Commercial databases have traditionally depended on high cost SCSI disks for high performance and reliability. As commodity disks gain more widespread usage, database software must evolve to produce low cost, reliable and scalable solutions from off-the-shelf components

Cost differential between raw and delivered storage

The operational costs required to manage data, whether in a database or a file system, vastly exceeds the cost of the hardware required to store it. Furthermore, there is a significant difference in hardware costs between “large scale enterprise class” hardware and “off the shelf” high volume, industry standard components.

Blurring structured and unstructured worlds

New technologies like XML, especially in the Document management world, are blurring the line between the worlds of structured and unstructured data. These allow for modeling and management of complex items such as an insurance claim that contains structured data (date, location, policy, etc.) and unstructured data (a sketch of the incident and photographs).

2.3 Device and Client Trends

More capable Devices

Devices are improving dramatically. Tomorrow's devices are likely to be as powerful and as capable as desktops of today. Devices, with all the different form factors, will become the environment of choice for most applications.

Mobility

As more users adopt Wi-Fi enabled laptops, increasingly capable PDAs, and Smartphones, the need for mobile applications is increasing. Applications like Email, Calendaring, CRM (Customer Relationship Management) already target mobile devices. Middleware infrastructures like application servers and workflow services are becoming mobile-aware. Some reasons for such mobility trends are:

- More employees are becoming mobile
- Mobile usage is broadening and becoming mainstream. It is already prevalent in certain vertical domains like Healthcare, Insurance, and Field Services.
- Email and offline access is becoming pervasive
- Mobile applications are more than just browser windows – more and more applications now run natively on mobile devices.

Data access and management on mobile devices is central to mobile applications. As these applications grow more disconnected and sophisticated, data sizes increase and the need for rich data processing capabilities increases. Mobile and embedded DBMSs – which are needed to support such applications - become an important part of the IT infrastructure

Streaming

In certain scenarios, data is simply streamed intelligently through application logic. There is no additional need to store the data – except transiently, for the duration of the operation. Conventional DBMSs, however, require data to be first loaded into the database; the operation is then performed, and the data may be later removed from the database. All of this adds significant complexity to the application, and dramatically reduces its performance and throughput.

Consider, for example, RFID data in applications. RFID tags are portable sensors that communicate over specialized protocols with RFID reader devices. An increasing number of applications have begun to utilize RFID technology. In these applications, RFID readers (devices) generate events when they identify RFID tags. The RFID event streams are filtered, aggregated, transformed, and correlated so that the events can be monitored in real-time. The event processing is data-centric and typically requires an in-memory rule engine and a query processor for real-time processing.

Occasionally Connected

Distributed and occasionally connected (or disconnected) application architectures fundamentally change the way applications access and manage data. Instead of locking data in a central location, data is moved closer to the applications, and this enables data processing to be performed in an autonomous and efficient fashion. These applications may run in the mid-tier, on desktops, or on mobile devices. Such an environment is inherently disconnected – there is no need for continuous connectivity between the data sources. Data may be fetched from its original sources, using data synchronization mechanisms, transformed and cached (or stored) close to the applications. Offline email is most pervasive applications that most of us are familiar with; applications like Aggregated Catalog is another example where

product information across multiple backend applications and data sources is aggregated and cached decoupled (actually occasionally connected to get changes) from its original data sources. Operations on these offline caches are data-centric and often require rich search, query, and other database capabilities.

2.4 Data and Applications Trends

Data Everywhere

As described earlier, data can reside in different tiers (in different service boundaries) with different semantics. For example, data in back-end data sources tends to be authoritative; data on the wire is message-oriented; data in mid-tier is cached or application session data. [5] refers to such data as Resource-Oriented Request Response Data and Activity-Oriented Reference Data. Resource Data is authoritative data – it is transactionally consistent and up to date; Reference data is typically a cached copy (in some other service boundary) of Resource Data. Reference Data may be stale; however, it is consistent enough for many applications (e.g. Product catalog data cached as Reference Data).

Data Proximity

Different forms of data have different processing requirements. For example, Activity Data (e.g. Shopping Cart) and Reference Data (e.g. Catalog data) can be cached close to the application and have minimal update requirements. In fact, in most applications, such data is either created (session data) or cached in memory by the application for performance and throughput reasons, within the application process. In-memory DBMS capabilities like query and transactions are required on the cached data. Also, on devices, typically the applications are simple and single user apps; they require data to be embedded in the application.

Desktop applications like Access require query and expression evaluation capabilities.

Embedded

Many applications perform simple to moderate manipulation of data. They need a way of storing, retrieving and manipulating the data within the application. These applications themselves are not complex in nature, and are designed to meet a specific user need. Typically, these applications are developed by application vendors specializing in industry verticals (domains) – e.g. Healthcare, Finance etc. These vendors are domain experts but not database specialists. They would rather not spend their time in database installation, deployment, and management. Their focus is the application and everything is embedded within the application. The databases are typically single application databases; the applications do not share (or coordinate) their database with others. These applications could run on devices, tablets, desktops, or servers. The ideal way of deployment is deploying the database components along with the application as a single install.

Enterprise application and data services have moved beyond the traditional relational database model

Ten years ago, the services provided by relational database systems such as: Insert, Delete, Query and Load were defined in terms of the logical relational model itself – namely relations (tables) and tuples (rows). Client/server applications developed a decade ago were also constructed by operating directly over the relational model. The code behind the “New Order” form knew to

start a new transaction and perform a query to get the customer ID; insert a new row in the “OrderHeaders” table, and then insert a series of rows in the “OrderDetails” table. Committing the transaction brought the new order to life. (A case of function literally following form...) See Figure 1 for an illustration.

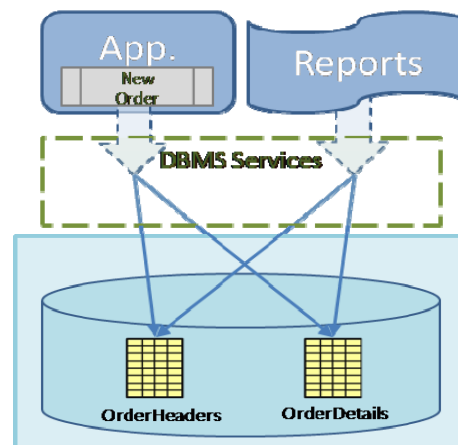


Figure 1: Traditional Client/Server Application Pattern

Today’s database services as well as current and emerging application patterns are no longer directly built upon the relational model. We’ll include several examples to briefly illustrate the shift.

SQL Server 7.0 introduced a feature known as “Merge Replication” that allowed an application to update rows in a SQL table configured in a replication relationship with another server and then merge the changes back into the publishing table. Once shipped, customers ran into logical consistency errors. Using the same “New Order” example from above, imagine a salesman used an application that created a new order with 5 line items (and thus 5 new rows in the OrderDetails table). Now assume the salesman is to synchronize the local changes with the publishing OrderHeaders and OrderDetails table and during the operation, replication is interrupted after the OrderHeader row is inserted and 3 of the 5 OrderDetails rows have been transferred. The orderprocessing logic, when invoked, will process an incomplete order. This problem was fixed in SQL Server 2005 with the addition of a new replication feature known as “Logical Records” in which metadata describing the logical transfer units is described in the SQL catalog and the replication subsystem has been enhanced to only process complete logical records. Merge Replication now represents a SQL Server data service that is defined over a logical “entity” rather than the underlying storage representation in a relational schema.

SQL Server 2005 adds several other new data services over higher level entities such as the end user reporting tool: “Report Builder”. Analysts want to author reports directly over Customers and Orders rather than expressing joins over the 5-10 tables that may represent Customers and Orders. Analysts would also want to build information in terms of Customers and Orders instead of having to express the underlying joins to create the cube. See Figure 2.

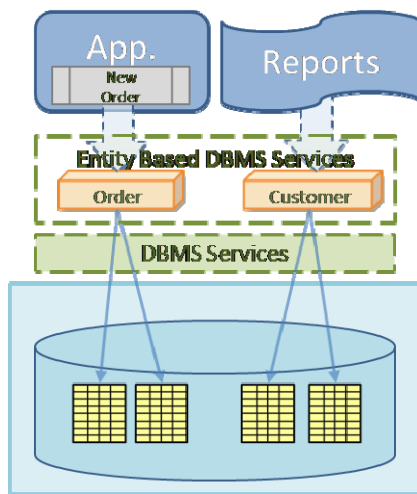


Figure 2: Entity Based Database Services

The movement towards assembling the data associated with a logical business “Entity” has been underway for some time. Application designers have been creating “Business Objects” for some time by constructing these entities with database queries and then adding other services, such as business behavior on top. Emerging information products, such as Enterprise Information Integration (EII) systems, Enterprise Search and Business Data Catalogs are services based upon higher level logical entities such as Customers, SalesPeople, and FixedAssets as well¹.

2.5 Services Trends

Data in the Cloud

With the increased connectivity and low cost storage, more data is being stored in web, especially in the consumer markets. The explosion of on-line search (e.g. Google, MSN), collaboration (e.g. Yahoo), and shopping (e.g. Amazon, EBAY), since the early 2000s, is driving a shift in how users want to store and share their information. The convenience of browser-based access to data from anywhere and at any time is causing more data that was traditionally stored on desktops, to migrate to the web (cloud). To facilitate this trend, the cloud tier has to provide reliable storage at very low cost to the users; the data service providers have to build the cloud infrastructure with low capital and operational expenditure. Also, as users trust more data storage to the cloud, the cloud tier must provide trustworthy storage with security and privacy.

In the cloud-centric applications, the client also plays an important role. While the thin browser based client can provide access from anywhere, the access is dependent on continuous connectivity and bandwidth. Also the user experience is limited by the browser experience. The desktops have traditionally provided rich client experience but they are not suitable for the mobile, anywhere access. Mobile devices with local storage (caching) can provide

¹ Note that these new data services are generally not built upon Object/Relational mapping technologies as they generally employ standard database technologies for data integration across potentially dissimilar data sources and often employ data transformations for tasks such as domain mapping.

such anywhere access to the cloud, in a connected and disconnected fashion.

3. THE MICROSOFT DATA PLATFORM

In this section we describe the Microsoft Data Platform. The Microsoft Data Platform is not monolithic piece of software; instead it is collection of product components and provides mechanisms to address the changing environment, as described in the previous section.

This section is organized as two sub sections – the Key Tenets that drive the Microsoft Data Platform vision, and an Architecture overview that describes the different tiers in the Data Platform and a product map that support these tiers.

3.1 Key Tenets

Platform for all Data

Technology trends have pushed us to a point where the data organizational constructs of traditional file systems are no longer sufficient. Many customers are collecting and storing petabytes of event data such as web click streams or access control audit logs. Personal digital photograph collections of tens of thousands of photographs are not uncommon. And customers want to search, query, and rate their photo collections. Windows media player uses a small relational database engine [8] at its core to manage music.

By *all* data we mean:

- Traditional structured data such as that stored in relational databases today
- Byte streams of data (audio, video, images etc.) traditionally stored in a file system today
- Semi-structured streams such as XML documents
- Massive streams of data such as web click stream, feeds from sensor networks, audit logs, etc.

In the Microsoft Data Platform we envision the database system and the file system to be better integrated to exploit each others’ strengths. For example, the file system can store its properties in the database for queryability; and the database system can store unstructured data (BLOBs) in the file system for performance and functionality. Currently the transactional capabilities in the Windows file system, the filestream based BLOBs in SQL Server, and full-text search are on the path to enabling the platform for all data. We also envision a common architecture that provides uniform data access (e.g. search, query) and management across all formats of data.

Low cost, Scalable, Reliable Storage

There is a significant opportunity to use high volume off the shelf hardware and smart software to build infinitely scalable and adaptive clusters for both structured and unstructured storage. Our aim is to reduce the total cost (both capital and operational expenditures) of delivered storage significantly in both enterprise and mega-scale scenarios.

Several MSN [6] properties or affiliates (like Hotmail, Messenger) use combination of SQL Server and file system products for supporting mega-scale services at low cost. However, there are significant opportunities for the database and file systems to

leverage commodity storage to deliver reliable storage at significantly lower cost.

Key goals are:

- Deliver significant improvements in acquisition and operations (maintenance and management) costs for structured and unstructured storage.
- Enable friction free addition of necessary resources such as processing and disk with zero-touch provisioning and resource balancing, (including dynamic data repartitioning).

Enable Reach and Rich

In 2006 the volume of storage delivered in high volume off-the-shelf configurations (e.g. SATA and ATA drives) exceeded the volume of storage delivered via enterprise interconnects (SCSI and Fiber Channel) by a 5:1 margin. This provides a measure for the amount of data capacity being placed “at the edge” in consumer PCs, Phones, Cameras, and personal entertainment devices.

The SQL Server Compact Edition [8], as part of the Microsoft Data Platform, is targeted to provide compelling experience by allowing users to store data locally and to synchronize and access their data across all of their personal devices whether they are Phones, PCs, USB Flash Drives, PDAs, etc.

Another compelling capability of the Data Platform is the ability to provide a much better offline experience with zero-friction deployment web applications. Simple web applications such as time and attendance, expense report generation and approval could all benefit from a rich offline data experience. It should be possible for a developer to develop a rich offline capable web application as easily as they can create an online version today.

Model-centric Data and Services

Data centric applications have evolved significantly over the last decade. A decade ago the most common enterprise application model was a 2 or 3 tier client/server topology where all durable state was typically stored in a single relational database and most application logic interacted with the data in terms of the relational logical schema. Today’s applications models and patterns are much richer involving composition and integration of applications with vastly different data models and the data is no longer typically stored in a single database. This shift presents a number of challenges and opportunities.

One implication of the change is that platforms and applications must deal with a “dual schema” tension. Part of the application must work with the relational schema and other parts of the application and platform services must deal with alternative representations such as business objects or entities. By creating a concrete model to represent more complex entities such as “Customer”, “Order”, or “Asset” and then re-casting many of our services such as workflow processing, web services, synchronization in terms of these entities, will allow us to cleanly move many services which must be built in the application tier today to the platform level. Ultimately, we will evolve the SQL Server Data Engine towards natively understanding and processing “Entities” and “Entity Sets” rather than today’s rows and tables.

The Entity Data Model

In this section we describe a conceptual model, called the Entity Data Model (EDM) that provides a higher level of abstraction closer to applications. The EDM is intended for developing rich data-centric applications. It extends the classic relational model with concepts from the E-R domain. The central concepts in the EDM are entities and associations. *Entities* represent top-level items with identity, while *Associations* are used to *relate* (or, describe relationships between) two or more entities.

The EDM and the associated query language (Entity SQL) are implemented by the ADO.NET Entity Framework [2]. Several object programming models can be easily layered on top of the EDM. Similarly, the EDM can map to one or more DBMS implementations for persistence.

The EDM and Entity SQL represent a richer data model and data manipulation language for a data platform and are intended to enable applications such as CRM and ERP, data-intensive services such as Reporting, Business Intelligence, Replication and Synchronization, and data-intensive applications to model and manipulate data at a level of structure and semantics that is closer to their needs. EDM details are available in [3].

Future versions of the EDM will provide support for XML, support for open properties, URIs, nested collections and other capabilities. These capabilities are becoming more and more important, especially in web applications,

It should be mentioned that many existing EII and MDM solutions create and present logical entities as described here by composing and coordinating data from multiple sources. Once formed, these entities are often retained in an in-memory cache and, in some cases; updates may be made to the cache and propagated back to the appropriate system of record. The key point is the “top end” of the entity vision which includes model development and mapping between various transformations is equally viable for these EII and MDM like scenarios.

Data Integration and Rich BI

We are rapidly moving from an era where most of our focus was on data *management* such as loading, querying, backing up, data to an era where much of the focus is on deriving value from data. As the cost of data acquisition and storage has dropped, we have moved from designing information systems to answer specific questions to a design where integration and mining of disparate data sources reveals questions and answers we never anticipated. Allowing users and analysts to access, enrich, and report on the data they need to do their job, while allowing IT to control the source data quality, will significantly improve productivity. Today, complex BI solutions require too much up front planning and suffer from poor time-to-solution and poor return of business value. Microsoft has been a leader in providing “BI to the masses”, and we envision BI solutions across all tiers of the Data Platform in the future.

Cloud scaled data services

Today’s Web 2.0 mashups must often “scrape” results to create data for the mashup. Part of the problem has been that there has been no good way to present data models and query in a consistent way for data services [4].

A rough requirements framing for cloud data services could be:

1. Provide a reference data service with checkpoint restart and incremental update capabilities with no server side state. RSS feeds are getting closer to meeting this requirement. Here is an example service: a ZIP Code to City and State reference service. A subscriber could download the entire service database to start with and then request incremental updates as appropriate.
2. Provide a means to update shared data. A typical pattern is based upon optimistic concurrency control with a variety of conflict resolution schemes. An example data service would be a shared list service [9] with offline synch and edit capabilities.

As part of the Microsoft Data Platform, we envision a data service based on the EDM that allows web clients to navigate entities via HTTP (using REST based APIs) and receive results in an encoded XML or JSON [7].

Powerful Management

We must move to a model where IT manages information and applications rather than servers and disks. This is a global imperative that involves many teams but it has unique requirements for data management and leads us towards a world of “intelligent data management” where policies placed on content control its management rather than the container or store in which it’s placed. Given the amount of intelligence being placed on inexpensive storage devices, we can imagine incredible policy driven data protection schemes. For example, removable USB media devices such as flash drives or hard disks that implement a trusted platform module could allow us to implement a policy whereby certain files stored on the device could *only* be read in specific authorization contexts. That is, I could copy files from my Windows domain onto a flash drive but only access those files while logged in with my domain credentials or smart card.

Hosted SaaS (Software as a Service) Platform

There is a blizzard of activity around loose data integration and providing data services on the web. There are many places in the value chain where we could offer Storage Services from low level BLOB storage much like Amazon’s S3 to allowing third parties to host application logic which integrates with existing storage and services extending them in interesting ways. Fully realizing such hosted platform requires innovations in the areas described above.

3.2 Architecture

The Microsoft Data Platform vision is to provide services for data across all tiers. We sometimes refer to such a platform as End-to-End Data Platform. The Microsoft Data Platform provides data management and data access capabilities everywhere. In the client-server world, “everywhere” includes data services on the client and the (data) server; in the enterprise world, “everywhere” includes the data (server) tier, the app server (mid) tier, and the client tier; the mobile world includes the mobile device tier also; and the next generation web (Cloud) world includes data in the shared web space. Figure 3 illustrates the end-to-end data platform stack.

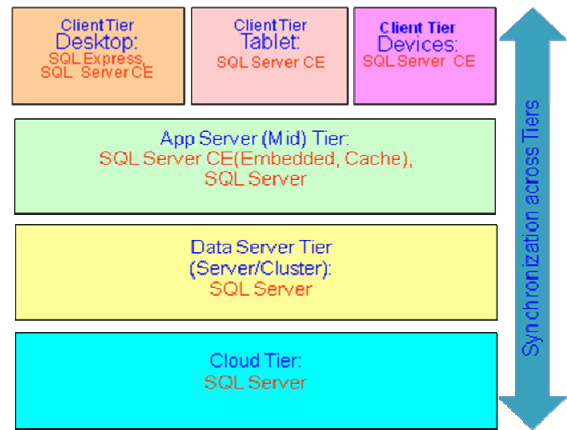


Figure 3: The Microsoft Data Platform

The data management requirements at each of the data tiers are different:

- **Cloud tier:** Data sizes are very large, typically at web scale. Data tends to be highly partitioned; there is lots of unstructured and semi-structured data; and these require lesser degrees of transactional integrity. Hands-off administration is typically a must, yet explicit control on provisioning, monitoring, and maintenance is essential; and simplicity and low cost (cheap) of application development, deployment, and management is critical.
- **Data Server tier:** Typically deals with enterprise data and enterprise scale. Data tends to be mostly structured, and highly transactional; data integrity is absolutely critical. Customization and manageability is important as well. While TCO is an important factor, these systems tend to be somewhat expensive.
- **Application Server Tier:** Typically deals with reference (or cached) data, and tends to be very application-centric - data is usually part of the application. Hosting machines tend to be cheap, off-the-shelf boxes; cost and resource consumption is therefore an important factor. Databases at this tier must be self-managed
- **Client tier:** On devices, small footprint and embedded-ness are critical. While the core relational functionality must be available, advanced functionality is not required. Data sizes are not very large. Zero admin (or self management) is critical for such tiers.

On desktops, applications deal with all types of data – structured and unstructured; PIM applications deal with file and structured data, and easy to manage DBMSs (SQL Express or SQL Server Compact Edition) are more suitable; LOB applications on the desktops are client-side enterprise apps and use stand-alone DBMSs. However, in the low-end developer environments – tablets, devices etc. - most applications (including PIM and LOB applications) are more like the embedded applications described previously and require simple, embedded DBMSs.

As described above and in previous sections, data in applications and in different tiers requires different styles of DBMSs (e.g.

Server DBMS, Mobile DBMS, and Embedded DBMS). These different DBMSs have different functional and physical characteristics (e.g. footprint, process boundaries). It is not practical to have a single DBMS that satisfies all the requirements of all the applications in all the tiers. We believe it is more pragmatic to envision a small number (two or three) of DBMSs, with some commonality, to address application needs across all tiers. Figure 3 illustrates the different tiers and the SQL products that support the data processing needs in these tiers.

3.2.1 Uniform Application Development

While applications in different tiers require different styles of DBMSs with varying levels of data management, they all prefer (require) significant uniformity in application development environment (programming and tools). Often, the same application may be deployed across multiple tiers (e.g. on the devices and on the desktops) and it is highly desirable to develop once and deploy on different tiers. In addition, as application scale increases and the application moves up tiers, it must be possible to upsize the database (e.g. from SQL Server CE to SQL Server), without requiring (significant) application changes.

Support for uniform application development requires the following:

- Entity Model across tiers
 - The entity model, which supports row, XML, and BLOB data, is desired across all the tiers
- Consistent programming model across tiers
 - Same (or subset) APIs; for example, ADO.NET, REST based APIs, etc.
 - Same (or subset) query and data definition language; for example, TSQL, Entity SQL, XQuery
- Same tool environment
 - It must be possible to use the same Visual Studio tools to develop applications against DBMSs in different tiers
 - It must be possible to develop once and deploy applications in different DBMSs in different tiers
- Consistent management across tiers
 - While it seems like a desirable goal, the difference in data scale across tiers makes the management experience different on different tiers. For example, on the client tier, it is highly desirable to have zero (or hands free) administration; in the cloud tier, it is necessary to have control on provisioning, monitoring etc. We believe declarative or policy-based management technology is critical for achieving consistent and yet customizable management across tiers

3.2.2 Synchronization across Tiers

In the distributed environment, where data is everywhere across the tiers, synchronization is a critical mechanism to move the data and keep it consistent across the tiers. Synchronization manifests in the Microsoft Data Platform in different ways. In the data server tier, the SQL Server transaction and merge replications are examples of synchronization mechanisms that replicate enterprise data for scalability and availability. Transaction replication can be used between servers for availability purposes. merge replication is typically used in client-server scenarios where data from the server is replicated across large numbers of clients.

Synchronization mechanisms are also used to replicate data from the data server tier to the mid-tier and application server caches. Data access to data server is offloaded to multiple middle-tier caches, thereby increasing the overall throughput (and scalability) of the application. In cases of mostly read data, these mid-tier caches also provide significant performance improvements for applications. It is critical that the synchronization mechanisms provide support for keeping the caches and the original data sources mutually consistent.

As described in section 2.3, synchronization is critical for supporting occasionally connected and offline applications. In these scenarios, synchronization works across tiers between clients and servers (mid-tier, data server tier, cloud tier). Microsoft Outlook's cached mode operation is an example of such offline application. The mail user always interacts with the local Outlook application; however, Outlook is synchronizing its mail items with the backend Exchange server in the background. Whether Outlook is connected to the backend Exchange server or disconnected from the server, the mail user experience remains the same.

The Microsoft Data Platform provides comprehensive synchronization services:

- Transactional and net-change based synchronization
- Master-slave and Peer-to-Peer configurations
- Synchronization over different communication protocols and media – e.g. over connected networks, wireless networks, using HTTP, etc.
- Synchronization of data at different abstractions – e.g. synchronization of files, rows, logical collection of rows, entities, etc.
- Synchronization from and to all tiers

3.2.3 Other Services

Other services in the Microsoft Data Platform include search, caching, notifications and subscriptions, security at different levels of abstraction, etc.

4. SUMMARY

This paper presented an overview of the Microsoft Data Platform. The Microsoft Data Platform is driven by the changing trends in hardware, software, storage, and in applications. These trends are having a huge impact on the way data-centric applications are being developed and deployed across the various data tiers. This paper presented key tenets and a high level product map to address the data processing requirements across tiers.

5. REFERENCES

- [1] ADO.NET, <http://msdn.microsoft.com/data/ref/adonetnext/>
- [2] Atul Adya, Blakeley, J.A., Melnik Sergey, S. Muralidhar. Anatomy of the ADO.NET Entity Framework, SIGMOD 2007
- [3] Blakeley, J.A., S. Muralidhar, Nori, A. The ADO.NET Entity Framework: Making the Conceptual Level Real, *ER* 2006.
- [4] GDATA, <http://code.google.com/apis/gdata/index.html>
- [5] Helland, Pat., Thoughts on Data and Process, <http://www.pathelland.com/presentations/2003/powerpoint/031218-ThoughtsOnDataandProcess-75mins.ppt>
- [6] Microsoft Network (MSN)
- [7] Pablo Castro, Accessing Data Services in Cloud, MIX07, <http://www.visitmix.com/>
- [8] SQL Server Compact Edition, <http://www.microsoft.com/sql/editions/compact/default.mspix>
- [9] Ta-da Lists, <http://www.tadalist.com/>