= WinFS =

WinFS (short for Windows Future Storage) is the code name for a canceled data storage and management system project based on relational databases , developed by Microsoft and first demonstrated in 2003 as an advanced storage subsystem for the Microsoft Windows operating system , designed for persistence and management of structured , semi @-@ structured as well as unstructured data .

WinFS includes a relational database for storage of information , and allows any type of information to be stored in it , provided there is a well defined schema for the type . Individual data items could then be related together by relationships , which are either inferred by the system based on certain attributes or explicitly stated by the user . As the data has a well defined schema , any application can reuse the data ; and using the relationships , related data can be effectively organized as well as retrieved . Because the system knows the structure and intent of the information , it can be used to make complex queries that enable advanced searching through the data and aggregating various data items by exploiting the relationships between them .

While WinFS and its shared type schema make it possible for an application to recognize the different data types , the application still has to be coded to render the different data types . Consequently , it would not allow development of a single application that can view or edit all data types ; rather what WinFS enables applications to understand is the structure of all data and extract the information that it can use further . When WinFS was introduced at the 2003 Professional Developers Conference , Microsoft also released a video presentation , named IWish , showing mockup interfaces that showed how applications would expose interfaces that take advantage of a unified type system . The concepts shown in the video ranged from applications using the relationships of items to dynamically offer filtering options to applications grouping multiple related data types and rendering them in a unified presentation .

WinFS was billed as one of the pillars of the " Longhorn " wave of technologies , and would ship as part of the next version of Windows . It was subsequently decided that WinFS would ship after the release of Windows Vista , but those plans were shelved in June 2006 , with some of its component technologies being integrated into upcoming releases of ADO.NET and Microsoft SQL Server . While it was then assumed by observers that WinFS was finished as a project , in November 2006 Steve Ballmer announced that WinFS was still in development , though it was not clear how the technology was to be delivered . Several components of the last Integrated Storage Initiative project , Microsoft Semantic Engine , presented at Microsoft PDC 2009 , have been integrated back into the SQL Server " Denali " . At the 2010 SQL Server PASS Community Summit , the forthcoming version of SQL Server (" Denali ") was shown , which seems to incorporate many of the WinFS ideas .

= = Motivation = =

Many filesystems found on common operating systems , including the NTFS filesystem which is used in modern versions of Microsoft Windows , store files and other objects only as a stream of bytes , and have little or no information about the data stored in the files . Such file systems also provide only a single way of organizing the files , namely via directories and file names .

Because a file system has no knowledge about the data it stores , applications tend to use their own , often proprietary , file formats . This hampers sharing of data between multiple applications . It becomes difficult to create an application which processes information from multiple file types , because the programmers have to understand the structure and semantics of all the files . Using common file formats is a workaround to this problem but not a universal solution ; there is no guarantee that all applications will use the format . Data with standardized schema , such as XML documents and relational data fare better , as they have a standardized structure and run @-@ time requirements .

Also , a traditional file system can retrieve and search data based only on the filename , because the only knowledge it has about the data is the name of the files that store the data . A better solution is to tag files with attributes that describe them . Attributes are metadata about the files such

as the type of file (such as document , picture , music , creator , etc .) . This allows files to be searched for by their attributes , in ways not possible using a folder hierarchy , such as finding " pictures which have person X " . The attributes can be recognizable by either the file system natively , or via some extension . Desktop search applications take this concept a step further . They extract data , including attributes , from files and index it . To extract the data , they use a filter for each file format . This allows for searching based on both the file 's attributes and the data in it .

However , this still does not help in managing related data , as disparate items do not have any relationships defined . For example , it is impossible to search for " the phone numbers of all persons who live in Acapulco and each have more than 100 appearances in my photo collection and with whom I have had e @-@ mail within last month " . Such a search could not be done unless it is based on a data model which has both the semantics as well as relationships of data defined . WinFS aims to provide such a data model and the runtime infrastructure that can be used to store the data , as well as the relationships between data items according to the data model , doing so at a satisfactory level of performance .

= = Overview = =

WinFS natively recognizes different types of data , such as picture , e @-@ mail , document , audio , video , calendar , contact , rather than just leaving them as raw unanalyzed bytestreams (as most file systems do) . Data stored and managed by the system are instances of the data type recognized by the WinFS runtime . The data are structured by means of properties . For example , an instance of a résumé type will surface the data by exposing properties , such as Name , Educational Qualification , Experience . Each property may be a simple type (strings , integers , dates) or complex types (contacts) . Different data types expose different properties . Besides that , WinFS also allows different data instances to be related together ; such as a document and a contact can be related by an Authored By relationship . Relationships are also exposed as properties ; for example if a document is related to a contact by a Created By relationship , then the document will have a Created By property . When it is accessed , the relationship is traversed and the related data returned . By following the relations , all related data can be reached . WinFS promotes sharing of data between applications by making the data types accessible to all applications , along with their schemas . When an application wants to use a WinFS type , it can use the schema to find the data structure and can use the information . So , an application has access to all data on the system even though the developer did not have to write parsers to recognize the different data formats . It can also use relationships and related data to create dynamic filters to present the information the application deals with . The WinFS API further abstracts the task of accessing data . All WinFS types are exposed as .NET objects with the properties of the object directly mapping to the properties of the data type . Also , by letting different applications which deal with the same data share the same WinFS data instance rather than storing the same data in different files , the hassles of synchronizing the different stores when the data change are removed . Thus WinFS can reduce redundancies .

Access to all the data in the system allows complex searches for data across all the data items managed by WinFS . In the example used above (" the phone numbers of all persons who live in Acapulco and each have more than 100 appearances in my photo collection and with whom I have had e @-@ mail within last month ") , WinFS can traverse the subject relationship of all the photos to find the contact items . Similarly , it can filter all emails in last month and access the communicated with relation to reach the contacts . The common contacts can then be figured out from the two sets of results and their phone number retrieved by accessing the suitable property of the contact items .

In addition to fully schematized data (like XML and relational data) , WinFS supports semi @-@ structured data (such as images , which have an unstructured bitstream plus structured metadata) as well as unstructured data (such as files) as well . It stores the unstructured components as files while storing the structured metadata in the structured store . Internally , WinFS uses a relational database to manage data . It does not limit the data to belonging to any particular data model . The

WinFS runtime maps the schema to a relational modality , by defining the tables it will store the types in and the primary keys and foreign keys that would be required to represent the relationships . WinFS includes mappings for object and XML schemas by default . Mappings for other schemas must be specified . Object schemas are specified in XML ; WinFS generates code to surface the schemas as .NET classes . ADO.NET can be used to directly specify the relational schema , though a mapping to the object schema must be provided to surface it as classes . Relationship traversals are performed as joins on these tables . WinFS also automatically creates indexes on these tables , to enable fast access to the information . Indexing speeds up joins significantly , and traversing relationships to retrieve related data is performed very fast . Indexes are also used during information search ; searching and querying use the indexes to quickly complete the operations , much like desktop search systems .

= = Development = =

The development of WinFS is an extension to a feature which was initially planned in the early 1990s . Dubbed Object File System , it was supposed to be included as part of Cairo . OFS was supposed to have powerful data aggregation features , but the Cairo project was shelved , and with it OFS . However , later during the development of COM , a storage system , called Storage + , based on then @-@ upcoming SQL Server 8 @.@ 0 , was planned , which was slated to offer similar aggregation features . This , too , never materialized , and a similar technology , Relational File System (RFS) , was conceived to be launched with SQL Server 2000 . However , SQL Server 2000 ended up being a minor upgrade to SQL Server 7 @.@ 0 and RFS was not implemented .

But the concept was not scrapped . It just morphed into WinFS . WinFS was initially planned for inclusion in Windows Vista , and build 4051 of Windows Vista , then called by its codename " Longhorn " , given to developers at the Microsoft Professional Developers Conference in 2003 , included WinFS , but it suffered from significant performance issues . In August 2004 , Microsoft announced that WinFS would not ship with Windows Vista ; it would instead be available as a downloadable update after Vista 's release .

On August 29 , 2005 , Microsoft quietly made Beta 1 of WinFS available to MSDN subscribers . It worked on Windows XP , and required the .NET Framework to run . The WinFS API was included in the System.Storage namespace . The beta was refreshed on December 1 , 2005 to be compatible with version 2 @.@ 0 of the .NET Framework . WinFS Beta 2 was planned for some time later in 2006 , and was supposed to include integration with Windows Desktop Search , so that search results include results from both regular files and WinFS stores , as well as allow access of WinFS data using ADO.NET.

However , on June 23 , 2006 , the WinFS team at Microsoft announced that WinFS would no longer be delivered as a separate product , and some components would be brought under the umbrella of other technologies - like the object @-@ relational mapping components into ADO.NET Entity Framework ; support for unstructured data , adminless mode of operation , support for file system objects via the FILESTREAM data type , and hierarchical data in SQL Server 2008 , then codenamed Katmai , as well as integration with Win32 APIs and Windows Shell and support for traversal of hierarchies by traversing relationships into later releases of Microsoft SQL Server ; and the synchronization components into Microsoft Sync Framework . However , having a shared @-@ schema storage system built into a future iteration of Microsoft Windows has not yet been ruled out .

With that announcement , most analysts assumed that the WinFS project was being killed off . But in November 2006 , Steve Ballmer said in an interview that WinFS is being actively developed but integration into the Windows codebase will come only after the technology has fully incubated . It was subsequently confirmed in an interview with Bill Gates and that Microsoft plans to migrate applications like Windows Media Player , Windows Photo Gallery , Microsoft Office Outlook etc. to use WinFS as the data storage back @-@ end .

In 2013 Bill Gates cited WinFS as his greatest disappointment at Microsoft and that the idea of WinFS was ahead of its time , which will re @-@ emerge .

= = Data storage = =

= = = Architecture = = =

WinFS uses a relational engine , which is derived from SQL Server 2005 , to provide the data relations mechanism . WinFS stores are simply SQL Server database (.MDF) files with the FILESTREAM attribute set . These files are stored in access @-@ restricted folder named " System Volume Information " placed into the volume root , in folders under the folder " WinFS " with names of GUIDs of these stores .

At the bottom of the WinFS stack lies WinFS Core which interacts with the filesystem and provides file access and addressing capabilities . The relational engine leverages the WinFS core services to present a structured store and other services such as locking which the WinFS runtime uses to implement the functionality . The WinFS runtime expose Services such as Synchronization and Rules which can be used to synchronize WinFS stores or perform certain actions on the occurrence of certain events .

WinFS runs as a service which runs three processes - WinFS.exe , which hosts relational datastore , WinFSSearch.exe , which hosts the indexing and querying engine , and WinFPM.exe (WinFS File Promotion Manager) , which interfaces with the underlying file system . It allows programmatic access to its features , via a set of .NET Framework APIs , that enables applications to define custom made data types , define relationships among data , store and retrieve information , and allow advanced searches . The applications can then aggregate the data and present the aggregated data to the user .

= = = = Data store = = = =

WinFS stores data in relational stores , which are exposed as virtual locations called stores . A WinFS store is a common repository where any application can store data along with its metadata , relationships and schema . WinFS runtime can apply certain relationships itself ; for example , if the values of the subject property of a picture and the name property of a contact are same , then WinFS can relate the contact with the picture . Relations can also be specified by other applications or the user .

WinFS provides a unified storage , but stops short of defining the format that is to be stored in the data stores . Instead it supports data to be written in application specific formats . But applications must provide a schema that defines how the file format should be interpreted . For example , a schema could be added to allow WinFS to understand how to read and thus be able to search and analyze , say , a PDF file . By using the schema , any application can read data from any other application , and also allows different applications to write in each other ? s format by sharing the schema .

Multiple WinFS stores can be created on a single machine . This allows different classes of data to be kept segregated ; for example , official documents and personal documents can be kept in different stores . WinFS , by default , provides only one store , named " DefaultStore " . WinFS stores are exposed as shell objects , akin to Virtual folders , which dynamically generates a list of all items present in the store and presents them in a folder view . The shell object also allows searching information in the datastore .

A data unit that has to be stored in a WinFS store is called a WinFS Item . A WinFS item , along with the core data item , also contains information on how the data item is related to other data . This Relationship is stored in terms of logical links . Links specify which other data items the current item is related with . Put in other words , links specify the relationship of the data with other data items . Links are physically stored using a link identifier , which specifies the name and intent of the relationship , such as type of or consists of . The link identifier is stored as an attribute of the data item . All the objects which have the same link id are considered to be related . An XML schema ,

defining the structure of the data items that will be stored in WinFS , must be supplied to the WinFS runtime beforehand . In Beta 1 of WinFS , the schema assembly had to be added to the GAC before it could be used .

= = = Data model = = =

WinFS models data using the data items , along with their relationships , extensions and rules governing its usage . WinFS needs to understand the type and structure of the data items , so that the information stored in the data item can be made available to any application that requests it . This is done by the use of schemas . For every type of data item that is to be stored in WinFS , a corresponding schema needs to be provided to define the type , structure and associations of the data . These schemas are defined using XML .

Predefined WinFS schemas include schemas for documents , e @-@ mail , appointments , tasks , media , audio , video , and also includes system schemas that include configuration , programs , and other system @-@ related data . Custom schemas can be defined on a per @-@ application basis , in situations where an application wants to store its data in WinFS , but not share the structure of that data with other applications , or they can be made available across the system .

= = = = Type system = = = =

The most important difference between a file system and WinFS is that WinFS knows the type of each data item that it stores . And the type specifies the properties of the data item . The WinFS type system is closely associated with the .NET framework ? s concept of classes and inheritance . A new type can be created by extending and nesting any predefined types .

WinFS provides four predefined base types ? Items , Relationships , ScalarTypes and NestedTypes . An Item is the fundamental data object which can be stored , and a Relationship is the relation or link between two data items . Since all WinFS items must have a type , the type of item stored defines its properties . The properties of an Item may be a ScalarType , which defines the smallest unit of information a property can have , or a NestedType , which is a collection of more than one ScalarTypes and / or NestedTypes . All WinFS types are made available as .NET CLR classes .

Any object represented as a data unit , such as contact , image , video , document etc . , can be stored in a WinFS store as a specialization of the Item type . By default , WinFS provides Item types for Files , Contact , Documents , Pictures , Audio , Video , Calendar , and Messages . The File Item can store any generic data , which is stored in file systems as files . But unless an advanced schema is provided for the file , by defining it to be a specialized Item , WinFS will not be able to access its data . Such a file Item can only support being related to other Items .

A developer can extend any of these types , or the base type Item , to provide a type for his custom data . The data contained in an Item is defined in terms of properties , or fields which hold the actual data . For example , an Item Contact may have a field Name which is a ScalarType , and one field Address , a NestedType , which is further composed of two ScalarTypes . To define this type , the base class Item is extended and the necessary fields are added to the class . A NestedType field can be defined as another class which contains the two ScalarType fields . Once the type is defined , a schema has to be defined , which denotes the primitive type of each field , for example , the Name field is a String , the Address field is a custom defined Address class , both the fields of which are Strings . Other primitive types that WinFS supports are Integer , Byte , Decimal , Float , Double , Boolean and DateTime , among others . The schema will also define which fields are mandatory and which are optional . The Contact Item defined in this way will be used to store information regarding the Contact , by populating the properties field and storing it . Only those fields marked as mandatory needs to be filled up during initial save . Other fields may be populated later by the user , or not populated at all . If more properties fields , such as last conversed date , need to be added , this type can be extended to accommodate them . Item types for other data can be defined similarly .

WinFS creates tables for all defined Items . All the fields defined for the Item form the columns of

the table and all instances of the Item are stored as rows in the table for the respective Items . Whenever some field in the table refers to data in some other table , it is considered a relationship . The schema of the relationship specifies which tables are involved and what the kind and name of the relationship is . The WinFS runtime manages the relationship schemas . All Items are exposed as .NET CLR objects , with a uniform interface providing access to the data stored in the fields . Thus any application can retrieve object of any Item type and can use the data in the object , without being aware of the physical structure the data was stored in .

WinFS types are exposed as .NET classes , which can be instantiated as .NET objects . Data are stored in these type instances by setting their properties . Once done , they are persisted into the WinFS store . A WinFS store is accessed using an ItemContext class (see Data retrieval section for details) . ItemContext allows transactional access to the WinFS store ; i.e. all the operations since binding an ItemContext object to a store till it is closed either all succeed or are all rolled back . As changes are made to the data , they are not written to the disc ; rather they are written to an in @-@ memory log . Only when the connection is closed are the changes written to the disc in a batch . This helps to optimize disc I / O. The following code snippet , written in C # , creates a contact and stores it in a WinFS store .

= = = = Relationships = = = =

A datum can be related to one more item , giving rise to a one @-@ to @-@ one relationship , or with more than one items , resulting in a one @-@ to @-@ many relationship . The related items , in turn , may be related to other data items as well , resulting in a network of relationships , which is called a many @-@ to @-@ many relationship . Creating a relationship between two Items create another field in the data of the Items concerned which refer the row in the other Item ? s table where the related object is stored .

In WinFS , a Relationship is an instance of the base type Relationship , which is extended to signify a specialization of a relation . A Relationship is a mapping between two items , a Source and a Target . The source has an Outgoing Relationship , whereas the target gets an Incoming Relationship . WinFS provides three types of primitive relationships ? Holding Relationship , Reference Relationship and Embedding Relationship . Any custom relationship between two data types are instances of these relationship types .

Holding Relationships specifies ownership and lifetime (which defines how long the relationship is valid) of the Target Item . For example , the Relationship between a folder and a file , and between an Employee and his Salary record , is a Holding Relationship ? the latter is to be removed when the former is removed . A Target Item can be a part of more than one Holding Relationships . In such a case , it is to be removed when all the Source Items are removed .

Reference Relationships provide linkage between two Items , but do not have any lifetime associated , i.e. , each Item will continue to be stored even without the other .

Embedding Relationships give order to the two Items which are linked by the Relationship , such as the Relationship between a Parent Item and a Child Item .

Relationships between two Items can either be set programmatically by the application creating the data , or the user can use the WinFS Item Browser to manually relate the Items . A WinFS item browser can also graphically display the items and how they are related , to enable the user to know how their data are organized .

= = = = Rules = = = =

WinFS includes Rules , which are executed when a certain condition is met . WinFS rules work on data and data relationships . For example , a rule can be created which states that whenever an Item is created which contains field " Name " and if the value of that field is some particular name , a relationship should be created which relates the Item with some other Item . WinFS rules can also access any external application . For example , a rule can be built which launches a Notify application whenever a mail is received from a particular contact . WinFS rules can also be used to

add new properties fields to existing data Items .
WinFS rules are also exposed as .NET CLR objects . As such any rule can be used for any purpose . A rule can even be extended by inheriting from it to form a new rule which consists of the condition and action of the parent rule plus something more .

= = = RAV = = =

WinFS supports creating Rich Application Views (RAV) by aggregating different data in a virtual table format . Unlike database view , where each individual element can only be a scalar value , RAVs can have complex Items or even collections of Items . The actual data can be across multiple data types or instances and can even be retrieved by traversing relationships . RAVs are intrinsically paged (dividing the entire set of data into smaller pages containing disconnected subsets of the data) by the WinFS runtime . The page size is defined during creation of the view and the WinFS API exposes methods to iterate over the pages . RAVs also supports modification of the view according to different grouping parameters . Views can also be queried against .

= = = Access control = = =

Even though all data are shared , everything is not equally accessible . WinFS uses the Windows authentication system to provide two data protection mechanisms . First , there is share @-@ level security that controls access to your WinFS share . Second , there is item level security that supports NT compatible security descriptors . The process accessing the item must have enough privileges to access it . Also in Vista there is the concept of " integrity level " for an application . Higher integrity data cannot be accessed by a lower integrity process .

= = Data retrieval = =

The primary mode of data retrieval from a WinFS store is querying the WinFS store according to some criteria , which returns an enumerable set of items matching the criteria . The criteria for the query is specified using the OPath query language . The returned data are made available as instances of the type schemas , conforming to the .NET object model . The data in them can be accessed by accessing the properties of individual objects .
Relations are also exposed as properties . Each WinFS Item has two properties , named IncomingRelationships and OutgoingRelationships , which provides access to the set of relationship instances the item participates in . The other item which participates in one relationship instance can be reached through the proper relationship instance .
The fact that the data can be accessed using its description , rather than location , can be used to provide end @-@ user organizational capabilities without limiting to the hierarchical organization as used in file @-@ systems . In a file system , each file or folder is contained in only one folder . But WinFS Items can participate in any number of holding relationships , that too with any other items . As such , end users are not limited to only file / folder organization . Rather , a contact can become a container for documents ; a picture a container for contacts and so on . For legacy compatibility , WinFS includes a pseudo @-@ type called Folder which is present only to participate in holding relationships and emulate file / folder organization . Since any WinFS Item can be related with more than one Folder item , from an end user perspective , an item can reside in multiple folders without duplicating the actual data . Applications can also analyze the relationship graphs to present various filters . For example , an email application can analyze the related contacts and the relationships of the contacts with restaurant bills and dynamically generate filters like " Emails sent to people I had lunch with " .

= = = Searches = = =

The WinFS API provides a class called the ItemContext class , which is bound to a WinFS store .

The ItemContext object can be used to scope the search to the entire store or a subset of it . It also provides transactional access to the store . An object of this class can then spawn an ItemSearcher object which then takes the type (an object representing the type) of the item to be retrieved or the relationship and the OPath query string representing the criteria for the search . A set of all matches is returned , which can then be bound to a UI widget for displaying en masse or enumerating individually . The properties items can also be modified and then stored back to the data store to update the data . The ItemContext object is closed (which marks the end of association of the object with the store) when the queries are made or changes merged into the store .

Related items can also be accessed through the items . The IncomingRelationships and OutgoingRelationships properties give access to all the set of relationship instances , typed to the name of the relationship . These relationship objects expose the other item via a property . So , for example , if a picture is related to a picture , it can be accessed by traversing the relationship as :