Microsoft Jet Database Engine

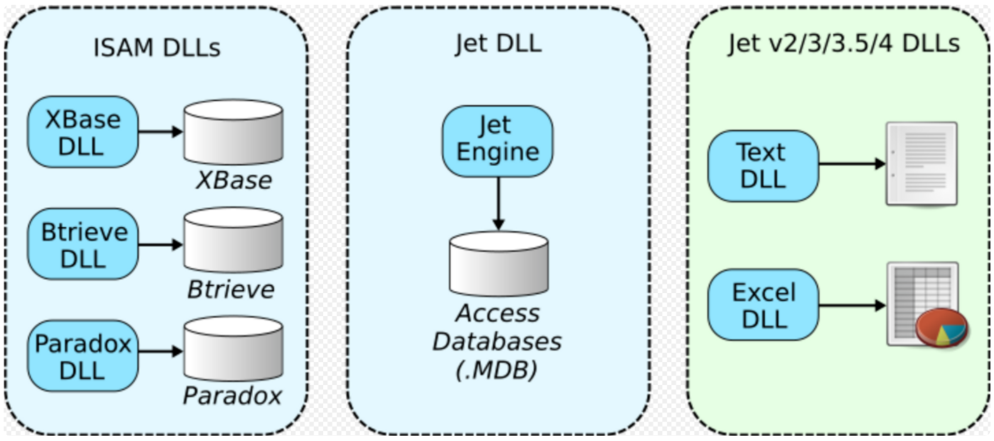
Subjects: Computer Science, Software Engineering
Contributor: HandWiki Li

The Microsoft Jet Database Engine (also Microsoft JET Engine or simply Jet) is a database engine on which several Microsoft products have been built. The first version of Jet was developed in 1992, consisting of three modules which could be used to manipulate a database. JET stands for Joint Engine Technology. Microsoft Access and Visual Basic use or have used Jet as their underlying database engine. However, it has been superseded for general use, first by Microsoft Desktop Engine (MSDE), then later by SQL Server Express. For larger database needs, Jet databases can be upgraded (or, in Microsoft parlance, "up-sized") to Microsoft's flagship SQL Server database product. A five billion record MS Jet (Red) database with compression and encryption turned on requires about one terabyte of disk storage space. It comprises typically hundreds of *.mdb files.

Keywords: msde ; jet ; encryption

1. Architecture

Jet, being part of a relational database management system (RDBMS), allows the manipulation of relational databases. It offers a single interface that other software can use to access Microsoft databases and provides support for security, referential integrity, transaction processing, indexing, record and page locking, and data replication. In later versions, the engine has been extended to run SQL queries, store character data in Unicode format, create database views and allow bi-directional replication with Microsoft SQL Server.



Jet DLLs. https://handwiki.org/wiki/index.php?curid=2041956

There are three modules to Jet: One is the *Native Jet ISAM Driver*, a dynamic link library (DLL) that can directly manipulate Microsoft Access database files (MDB) using a (random access) file system API. Another one of the modules contains the *ISAM Drivers*, DLLs that allow access to a variety of Indexed Sequential Access Method ISAM databases, among them xBase, Paradox, Btrieve and FoxPro, depending on the version of Jet. The final module is the *Data Access Objects* (DAO) DLL. DAO provides an API that allows programmers to access JET databases using any programming language.

1.1. Locking

Jet allows multiple users to access the database concurrently. To prevent that data from being corrupted or invalidated when multiple users try to edit the same record or page of the database, Jet employs a locking policy. Any single user can modify only those database records (that is, items in the database) to which the user has applied a lock, which gives exclusive access to the record until the lock is released. In Jet versions before version 4, a page locking model is used, and in Jet 4, a record locking model is employed. Microsoft databases are organized into data "pages", which are fixed-length (2 kB before Jet 4, 4 kB in Jet 4) data structures. Data is stored in "records" of variable length that may take up less or more than one page. The page locking model works by locking the pages, instead of individual records, which though less resource-intensive also means that when a user locks one record, all other records on the same page are collaterally locked. As a result, no other user can access the collaterally locked records, even though no user is accessing them and there is no need for them to be locked. In Jet 4, the record locking model eliminates collateral locks, so that every record that is not in use is available.

There are two mechanisms that Microsoft uses for locking: *pessimistic locking*, and *optimistic locking*. With pessimistic locking, the record or page is locked immediately when the lock is requested, while with optimistic locking, the locking is delayed until the edited record is saved. Conflicts are less likely to occur with optimistic locking, since the record is locked only for a short period of time. However, with optimistic locking one cannot be certain that the update will succeed because another user could lock the record first. With pessimistic locking, the update is guaranteed to succeed once the lock is obtained. Other users must wait until the lock is released in order to make their changes. Lock conflicts, which either require the user to wait, or cause the request to fail (usually after a timeout) are more common with pessimistic locking.

1.2. Transaction Processing

Jet supports transaction processing for database systems that have this capability. (ODBC systems have one-level transaction processing, while several ISAM systems like Paradox do not support transaction processing.) A transaction is a series of operations performed on a database that must be done together — this is known as atomicity and is one of the ACID (Atomicity, Consistency, Isolation, and Durability), concepts considered to be the key transaction processing features of a database management system. For transaction processing to work (until Jet 3.0), the programmer needed to begin the transaction manually, perform the operations needed to be performed in the transaction, and then commit (save) the transaction. Until the transaction is committed, changes are made only in memory and not actually written to disk.[1] Transactions have a number of advantages over independent database updates. One of the main advantages is that transactions can be abandoned if a problem occurs during the transaction. This is called rolling back the transaction, or just rollback, and it restores the state of the database records to precisely the state before the transaction began. Transactions also permit the state of the database to remain consistent if a system failure occurs in the middle of a sequence of updates required to be atomic. There is no chance that only some of the updates will end up written to the database; either all will succeed, or the changes will be discarded when the database system restarts. With ODBC's in-memory policy, transactions also allow for many updates to a record to occur entirely within memory, with only one expensive disk write at the end.

Implicit transactions were supported in Jet 3.0. These are transactions that are started automatically after the last transaction was committed to the database. Implicit transactions in Jet occurred when an SQL DML statement was issued. However, it was found that this had a negative performance impact in 32-bit Windows (Windows 95, Windows 98), so in Jet 3.5 Microsoft removed implicit transactions when SQL DML statements were made.

1.3. Data Integrity

Jet enforces entity integrity and referential integrity. Jet will by default prevent any change to a record that breaks referential integrity, but Jet databases can instead use propagation constraints (cascading updates and cascading deletes) to maintain referential integrity.

Jet also supports "business rules" (also known as "constraints"), or rules that apply to any column to enforce what data might be placed into the table or column. For example, a rule might be applied that does not allow a date to be entered into a date_logged column that is earlier than the current date and time, or a rule might be applied that forces people to enter a positive value into a numeric only field.

1.4. Security

Access to Jet databases is done on a per user-level. The user information is kept in a separate system database, and access is controlled on each object in the system (for instance by table or by query). In Jet 4, Microsoft implemented functionality that allows database administrators to set security via the SQL commands CREATE, ADD, ALTER, DROP USER and DROP GROUP. These commands are a subset of ANSI SQL 92 standard, and they also apply to the GRANT/REVOKE commands.[1] When Jet 2 was released, security could also be set programmatically through DAO.

1.5. Queries

Queries are the mechanisms that Jet uses to retrieve data from the database. They can be defined in Microsoft QBE (Query By Example), through the Microsoft Access SQL Window or through Access Basic's Data Access Objects (DAO) language. These are then converted to an SQL SELECT statement. The query is then compiled — this involves parsing the query (involves syntax checking and determining the columns to query in the database table), then converted into an internal Jet query object format, which is then tokenized and organised into a tree like structure. In Jet 3.0 onwards these are then optimised using the Microsoft Rushmore query optimisation technology. The query is then executed and the results passed back to the application or user who requested the data.

Jet passes the data retrieved for the query in a dynaset. This is a set of data that is dynamically linked back to the database. Instead of having the query result stored in a temporary table, where the data cannot be updated directly by the user, the dynaset allows the user to view and update the data contained in the dynaset. Thus, if a university lecturer queries all students who received a distinction in their assignment and finds an error in that student's record, they would only need to update the data in the dynaset, which would automatically update the student's database record without the need for them to send a specific update query after storing the query results in a temporary table.

2. History

Jet originally started in 1992 as an underlying data access technology that came from a Microsoft internal database product development project, code named Cirrus. Cirrus was developed from a pre-release version of Visual Basic code and was used as the database engine of Microsoft Access. Tony Goodhew, who worked for Microsoft at the time, says

> "It would be reasonably accurate to say that up until that stage Jet was more the name of the team that was assigned to work on the DB engine modules of Access rather than a component team. For VB [Visual Basic] 3.0 they basically had to tear it out of Access and graft it onto VB. That's why they've had all those Jet/ODBC problems in VB 3.0."

Jet became more componentised when Access 2.0 was released because the Access ODBC developers used parts of the Jet code to produce the ODBC driver. A retrofit was provided that allowed Visual Basic 3.0 users to use the updated Jet issued in Access 2.0.[2]

Jet 2.0 was released as several dynamic linked libraries (DLLs) that were utilised by application software, such as Microsoft's Access database. DLLs in Windows are "libraries" of common code that can be used by more than one application—by keeping code that more than one application uses under a common library which each of these applications can use independently code maintenance is reduced and the functionality of applications increases, with less development effort. Jet 2.0 comprised three DLLs: the Jet DLL, the Data Access Objects (DAO) DLL and several external ISAM DLLs. The Jet DLL determined what sort of database it was accessing, and how to perform what was requested of it. If the data source was an MDB file (a Microsoft Access format) then it would directly read and write the data to the file. If the data source was external, then it would call on the correct ODBC driver to perform its request. The DAO DLL was a component that programmers could use to interface with the Jet engine, and was mainly used by Visual Basic and Access Basic programmers. The ISAM DLLs were a set of modules that allowed Jet to access three ISAM based databases: xBase, Paradox and Btrieve.[2] Jet 2.0 was replaced with Jet 2.1, which used the same database structure but different locking strategies, making it incompatible with Jet 2.0.

Jet 3.0 included many enhancements, including a new index structure that reduced storage size and the time that was taken to create indices that were highly duplicated, the removal of read locks on index pages, a new mechanism for page reuse, a new compacting method for which compacting the database resulted in the indices being stored in a clustered-index format, a new page allocation mechanism to improve Jet's read-ahead capabilities, improved delete operations that speeded processing, multithreading (three threads were used to perform read ahead, write behind, and cache maintenance), implicit transactions (users did not have to instruct the engine to start manually and commit transactions to the database), a new sort engine, long values (such as memos or binary data types) were stored in separate tables, and dynamic buffering (whereby Jet's cache was dynamically allocated at start up and had no limit and which changed from a first in, first out (FIFO) buffer replacement policy to a least recently used (LRU) buffer replacement policy).[3] Jet 3.0 also allowed for database replication. Jet 3.0 was replaced by Jet 3.5, which uses the same database structure, but different locking strategies, making it incompatible with Jet 3.0.

Jet 4.0 gained numerous additional features and enhancements.[1]

- Unicode character storage support, along with an NT sorting method that was also implemented in the Windows 95 version;
- Changes to data types to be more like SQL Server's (LongText or Memo; Binary; LongBinary; Date/Time; Real; Float4; IEEESingle; Double; Byte or Tinyint; Integer or Integer synonyms Smallint, Integer2, and Short; LongInteger or LongInteger synonyms Int, Integer, and Counter; Currency or Money; Boolean and GUID); a new decimal data type
- Memo fields could now be indexed
- Compressible data types
- SQL enhancements to make Jet conform more closely to ANSI SQL-92
- Finer grained security; views support; procedure support
- Invocation and termination (committing or rolling back) of transactions
- Enhanced table creation and modification
- Referential integrity support
- Connection control (connected users remain connected, but once disconnected they cannot reconnect, and new connections cannot be made. This is useful for database administrators to gain control of the database)
- A user list, which allows administrators to determine who is connected to the database
- Record-level locking (previous versions only supported page-locking)
- Bi-directional replication with MS SQL Server.

Microsoft Access versions from Access 2000 to Access 2010 included an "Upsizing Wizard" which could "upsize" (upgrade) a Jet database to "an equivalent database on SQL Server with the same table structure, data, and many other attributes of the original database". Reports, queries, macros and security were not handled by this tool, meaning that some manual modifications might have been needed if the application was heavily reliant on these Jet features.[4]

A standalone version of the Jet 4 database engine was a component of Microsoft Data Access Components (MDAC), and was included in every version of Windows from Windows 2000 on.[5] The Jet database engine was only 32-bit and did not run natively under 64-bit versions of Windows. This meant that native 64-bit applications (such as the 64-bit versions of SQL Server) could not access data stored in MDB files through ODBC, OLE DB, or any other means, except through intermediate 32-bit software (running in WoW64) that acted as a proxy for the 64-bit client.[6]

With version 2007 onwards, Access includes an Office-specific version of Jet, initially called the *Office Access Connectivity Engine* (ACE), but which is now called the *Access Database Engine* (However MS-Access consultants and VBA developers who specialize in MS-Access are more likely to refer to it as "the ACE Database Engine"). This engine was backward-compatible with previous versions of the Jet engine, so it could read and write (.mdb) files from earlier Access versions. It introduced a new default file format, (.accdb), that brought several improvements to Access, including complex data types such as multivalue fields, the attachment data type and history tracking in memo fields. It also brought security changes and encryption improvements and enabled integration with Microsoft Windows SharePoint Services 3.0 and Microsoft Office Outlook 2007.[7][8][9]

The engine in Microsoft Access 2010 discontinued support for Access 1.0, Access 2.0, Lotus 1-2-3 and Paradox files.[10] A 64-bit version of Access 2010 and its ACE Driver/Provider was introduced, which in essence provides a 64-bit version of Jet. The driver is not part of the Windows operating system, but is available as a redistributable.[11]

The engine in Microsoft Access 2013 discontinued support for Access 95, Access 97 and xBase files, and it also discontinued support for replication.[12]

Version 1608 of Microsoft Access 2016 restored support for xBase files,[13] and Version 1703 introduced a Large Number data type.[14]

From a data access technology standpoint, Jet is considered a deprecated technology by Microsoft,[15] but Microsoft continues to support ACE as part of Microsoft Access.

3. Compatibility

Microsoft provides the JET drivers only for Microsoft Windows. Therefore, third party software support for JET databases is almost exclusively found on Windows. There is an open source project that attempts to enable working with JET databases on other platforms, MDB Tools and its much extended Java port named Jackcess.

References

1. MS KB article 275561 (2007-01-29). "Description of the new features that are included in Microsoft Jet 4.0". Microsoft. http://support.microsoft.com/kb/275561/en.
2. Goodhew, Tony (November 1996). "Jet Engine: History". http://www.avdf.com/nov96/acc_jet.html.
3. MS KB article 137039 (2003-12-03). "New Features in Microsoft Jet Version 3.0". Microsoft. http://support.microsoft.com/kb/137039/en.
4. Microsoft, "Microsoft Access 2000 Data Engine Options", white paper.
5. MS KB article 239114 (2008-05-29). "How to obtain the latest service pack for the Microsoft Jet 4.0 Database Engine". Microsoft. http://support.microsoft.com/kb/239114/en.
6. Gorm Braarvig. "Access database from SQL 2005/64". http://gorm-braarvig.blogspot.com/2005/11/access-database-from-sql-200564.html.
7. Jakšić, Aleksandar (August 2008). "Developing Access 2007 Solutions with Native C or C++". Microsoft Corporation. http://msdn.microsoft.com/en-us/library/cc811599.aspx.
8. Andy Baron, Optimizing Microsoft Office Access Applications Linked to SQL Server, November 2006. http://msdn.microsoft.com/en-us/library/bb188204(SQL.90).aspx
9. Microsoft, New features of the Access 2007 file format . http://office.microsoft.com/en-us/access/HA100678311033.aspx
10. Microsoft, Discontinued features and modified functionality in Access 2010. http://office.microsoft.com/en-gb/access-help/discontinued-features-and-modified-functionality-in-access-2010-HA101806473.aspx
11. Adam W. Saxton, Microsoft SQL Server Escalation Services (2010-01-21). "How to get a x64 version of Jet?". https://docs.microsoft.com/en-us/archive/blogs/psssql/how-to-get-a-x64-version-of-jet.
12. Microsoft, Discontinued features and modified functionality in Access 2013. http://office.microsoft.com/en-us/access-help/discontinued-features-and-modified-functionality-in-access-2013-HA102749226.aspx
13. Microsoft, Back by popular demand—dBASE file support in Access https://blogs.office.com/en-us/2016/09/07/back-by-popular-demand-dbase-file-support-in-access/
14. Microsoft, What's New in Access 2016 https://support.office.com/en-gb/article/What-s-new-in-Access-2016-76454345-f85d-47af-ace1-98a456cb3496
15. Shirolkar, Prash; Henry, Alyssa; Pepitone, Stephen; Bunch, Acey J. (January 2008). "Data Access Technologies Road Map". Microsoft Corporation. http://msdn.microsoft.com/en-us/library/ms810810.aspx.