

# SQL Server 2014 In-Memory Tables (Hekaton)

Andrew Novick  
[www.NovickSoftware.com](http://www.NovickSoftware.com)

## Andy Novick

- SQL Server Consultant
- SQL Server MVP

Did you ever work on a software project that was late?

### Xanadu, The World's Most Delayed Software, Is Finally Released After 54 Years In The Making

The image shows a screenshot of a document titled "Xanadu". On the left side, there is a vertical timeline with several colored bars (green, red, blue, purple) extending downwards, representing a long history of development. The main content area contains text discussing the origins of the universe and the creation of the Xanadu system. The text is partially obscured by a blue highlight.

**Xanadu**  
Source: SampleContent/Xanadu/Mojito/2.2xv9.xanadoc

**ORIGINS**  
How did all this get here? How did we? Where did the earth and the heavens come from? And people?  
There must be obvious questions, because they have been answered over and over through the ages, with answers that have been many and varied, some of the answers are called "religions," because they involve gods and myths, and some of the answers have been called "science," because they have been advanced by academics. But they have a lot in common.  
The number of possible answers continues to grow with scientific progress and with the growth of new religions and their variants. While the number of sciences is greater than ever, the numbers of religions, and of the poor and ignorant, are also greater than ever, so the scientific and religious approaches continue to grow side by side.  
The origin of the universe is one thing, the origin of people is another. Somehow there came to be people on the earth, and most agree that the universe came first, before the people, but the Creation stories are very different.

**THE FAST CREATION OF THE UNIVERSE AND PEOPLE.**  
Western accounts begin with Bibles, the religious books begun by the Hebrews and extended by Christians. All the Bibles have a common account of a busy week when the universe, and the human race, were created by a God who somehow already existed.  
The creation of the universe is described in the King James Bible like this:  
1:1 In the beginning God created the heavens and the earth.  
1:2 And the earth was without form, and void, and darkness was upon the face of the deep. And the Spirit of God moved upon the face of the waters.  
1:3 And God said, Let there be light: and there was light.

## What's a Heketon?

- 100 Times Faster!
- They didn't get there ;(

## What makes it faster?

- No locks, latches, spinlocks, etc.
- Data is written sequentially, only!
- No indexes are stored or logged. They're in memory only.
- Non-durable tables. No I/O
  - Cache
  - Replacement for TempDB in some cases.
- Checkpoints are background sequential I/O
- Compiled Native stored procedures

## What's not faster

- Chatty applications may not see any gain.
- Logging is still done (except for indexes)
  - So Low Log Latency is still important
- DML on many rows????

## Compiled Sproc - Requirements

- Native Compilation
- SCHEMABINDING
- EXECUTE AS OWNER
- BEGIN Atomic – Begin a Tran or Savepoint
  - Isolation Level Must be declared
  - LANGUAGE – Fixed at compile time

## Compiled SPROC Surface Area Limitations

- Memory optimized tables only
- INNER JOIN supported
- OUTER JOIN not supported
  
- Also no: MERGE, OUTPUT,

## Estimating Memory Consumption

- $\text{Memory Size} = \text{Table Size} + \text{SUM}(\text{Index Size})$
  
- $\text{TABLE Size} = \text{Row Size} * \text{Row Count}$
- $\text{Row Size} = \text{SUM}(\text{Column Size}) + \text{Header}$
- $\text{Header} = 24 + 8 * \text{Index Count}$
  
- $\text{Hash Index size} = \text{Bucket\_Count} * 8 \text{ bytes}$
  
- The multiply it all by 2

## Can it Crash the system

- YES
- Integrated with Resource Governor

## Durability

- Data file
- Delta file

## Merge Process

- DBA job
- Processes the changes and produces a new starting data file
- `sys.sp_xtp_merge_checkpoint_files`
  - `@database_name`
  - `, @transaction_lower_bound`
  - `, @transaction_upper_bound`

## Indexes

- Inherently covering
  - NO INCLUDE needed
- HASH
- RANGE

## Logging

- Uses the traditional Log
- Only logical changes is logged.
- All 3 recovery models are supported

## Recovery

- Recovery at the speed of I/O
- This is a function of database size!



## Isolation Levels

- Snapshot
  - Reads are consistent as of the start of the TXN
  - Writes are always consistent
- Repeatable Read
  - Reads give the same row version a commit time as earlier
- Serializable
  - As if there were no concurrent transactions
  - All actions happen at a single commit time

## Optimistic Locking

- There are no traditional:
  - Locks
  - Latches
  - Spin-locks
- Multi-version Optimistic Locking
- You MUST code for failure.

## Interop vs Compiled SProc

### **Interop**

- Access both memory and disk tables
- Most of T-SQL
  
- Best for
  - Ad hoc queries
  - Reporting Queries
  - Migration

### **Memory with Compiled SProc**

- Only Memory tables
- Limited T-SQL
  
- Best for
  - OLTP
  - Optimized performance

## Not Supported - Tables

- LOB – Varchar(MAX), NVARCHAR(MAX)
  - Text, ntext, image, varbinary(max)
- Foreign Key
- Trigger
- Replication
- Collations other than Bin2! ('A' != 'a')

## Not Supported – Interpreted TSQL

- TRUNCATE TABLE
- MERGE (memory-optimized table as target)
- Dynamic and keyset cursors (these automatically degrade to static).
- Access from CLR modules, using the context connection.
- Referencing a memory-optimized table from an indexed view.
- Cross-database queries
- Cross-database transactions
- Linked servers

## No Supported in Compiled Sprocs

- Merge
- OUTER JOINS
- Lots of other stuff
  - OUTPUT
  - CURSOR

**DEMO**

## OLTP Demo

	OnDisk	InMem
1,000,000	133	14

## EAV

- Entity-Attribute-Value
- Entity – a thing that has characteristics
  - House
  - Person
  - Stock
- Attribute – Characteristic of an entity
  - Number of bedrooms
  - Weight
  - Market price
- Value the value of the attribute for the entity

## Point-in-Time Data

- Each row represents the value of the attribute for the entity for a range of time
- Ways to represent the time range
  - begin\_datetime
  - begin\_datetime - end\_datetime

## Typical EAV table – begin\_datetime

```
CREATE TABLE eav_pit_only_begin_datetime (
    attribute_id int not null
    , entity_id int not null
    , begin_datetime datetime not null
    , value varchar(255) not null

    , constraint pk_eav_pit_only_begin_datetime
    primary key clustered
    (attribute_id, entity_id, begin_datetime)
)
```

## EAV Multi-table with View

- Separate into 3 tables
  - eav\_current
    - begin\_datetime
    - No end\_datetime it is INFERRED as 9999-12-31
    - Partitioned by attribute\_id
  - eav\_recent
    - begin\_datetime/end\_datetime
    - Partitioned\_by\_attribute\_id
  - eav\_history
    - Begin\_datetime/end\_datetime
    - Partitioned by end\_datetime
- ETL loads to eav\_current saves history to eav\_recent
- Archive moves rows from eav\_recent to eav\_history

## Typical EAV table – begin/end datetime

```
CREATE TABLE eav_pit_only_begin_datetime (
  attribute_id int not null
  , entity_id int not null
  , begin_datetime datetime not null
  , end_datetime datetime default ('9999-12-31')
  , value varchar(255) not null

  , constraint pk_eav_pit_only_begin_datetime
    primary key clustered
    (attribute_id, entity_id, end_datetime)
)
```

## Typical EAV table – begin/end + Types

```
CREATE TABLE eav_pit_begin_end_datetime_types (
  attribute_id int not null
  , entity_id int not null
  , begin_datetime datetime not null
  , end_datetime datetime default ('9999-12-31')
  , data_type char(1) not null -- C, D, F, I
  , value_c varchar(255) null
  , value_d (255) null
  , value_f (255) null
  , value_i (255) null

  , constraint pk_eav_pit_begin_end_datetime_types
    primary key clustered
    (attribute_id, entity_id, end_datetime)
)
```

## References

- **SQL Server 2014 In-Mem Sample**  
[http://msdn.microsoft.com/en-us/library/dn511655\(v=sql.120\).aspx?WT.mc\\_id=Blog\\_SQL\\_SQL2014\\_DI](http://msdn.microsoft.com/en-us/library/dn511655(v=sql.120).aspx?WT.mc_id=Blog_SQL_SQL2014_DI)
- **Why Hekaton is Truly Revolutionary**  
[http://sqlblog.com/blogs/merrill\\_aldrich/archive/2013/10/21/why-hekaton-in-memory-oltp-truly-is-revolutionary.aspx](http://sqlblog.com/blogs/merrill_aldrich/archive/2013/10/21/why-hekaton-in-memory-oltp-truly-is-revolutionary.aspx)

Thank you for coming

Andy Novick

[anovick@novicksoftware.com](mailto:anovick@novicksoftware.com)