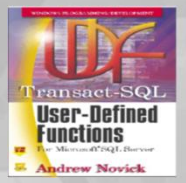
Entity-Attribute-Value (EAV)
The Antipattern Too Great to Give-up

Andy Novick

- SQL Server Consultant
- SQL Server MVP since 2010
- Author of 2 books on SQL Server
- anovick@NovickSoftware.com
- www.NovickSoftware.com



Microsoft® Most Valuable Professional

Novick Software

How long does it take for an enhancement request to get from the end user's request into production?

A Day?

A Week?

2 Weeks?

A Month?

6 Months?

A Year?

What do your users think of that?

What if?.....

- Users could add attributes (columns) at any time without you being involved?
- How about 300 attributes at a time?
- Without adding a new ETL program or changing an existing one?

What do you do when?

- Users need more than 1024 columns?
- More than 30,000?
- And they want to fill them all?

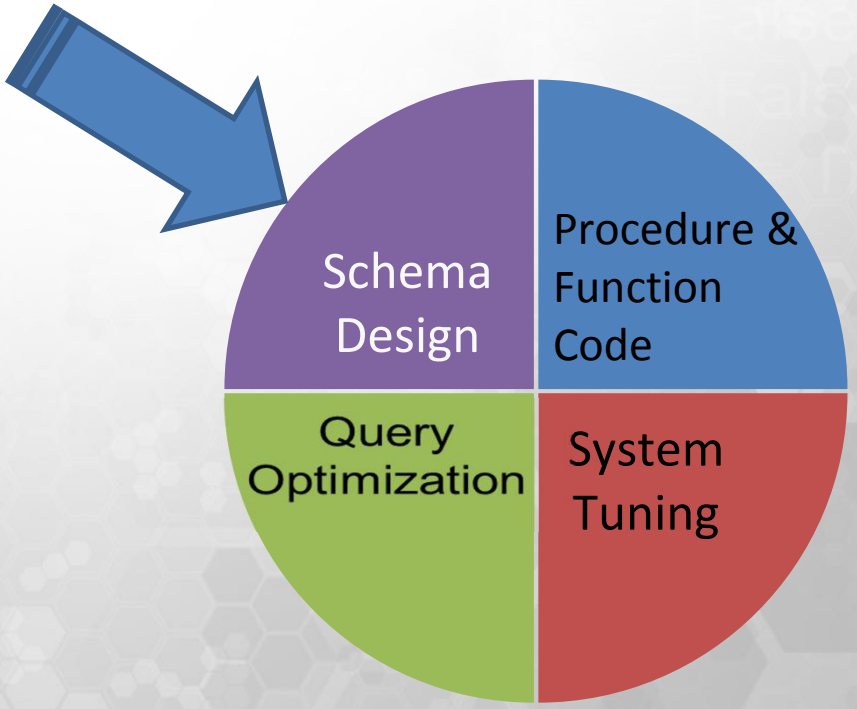
Have you tried an Entity-Attribute-Value schema?

E-A-V

Agenda:

- What is Entity-Attribute-Value
- Why use it
- Why is an Antipattern
- Handing the EAV problems

Focus of this Presentation



WHAT IS ENTITY-ATTRIBUTE-VALUE

Minimal Entity Attribute Value Table

```
CREATE Table eav (

   [entity_id]       int not null
 , attribute_id      smallint not null
 , value             varchar(255) not null


 , CONSTRAINT pk_eav PRIMARY KEY CLUSTERED
       (attribute_id, [entity_id])

   )
```

Usual Entity Attribute Value Table

```
CREATE Table eav (

   [entity_id]       int not null
 , attribute_id      smallint not null
 , value             varchar(255) not null

  TIME goes here!

, CONSTRAINT pk_eav PRIMARY KEY CLUSTERED
      (attribute_id, [entity_id])

  )
```

Where do you find EAV schemas

- Clinical Data — patient – heart rate - beats
- Financial Data — stock - market cap - amount
- E-Commerce — Customer – cell – phone#
- Survey systems — Survey – question - answer

Other Names for EAV

- OTLT – One true lookup table
- Open Schema
- Diabolically Enticing Method Of Data Storage (DEMONS)

EAV is a subset of Sixth Normal Form

Relvar R is in sixth normal form, 6NF, if an only if it can't be nonloss decomposed at all, other than triviality.

Observe, therefore that that (a) 6NF is the ultimate normal form with respect to normalization as conventionally understood;

*C. J. Date**

* The New Relational Database Dictionary - C. J. Date - 2016

Entity

An existing or real thing

- Person
- Company
- Stock
- Car
- Loan

Representing Entity

```
[entity_id]        int not null
```

Recommended

```
[entity_id]        varchar(255) not null
```

This can get you into trouble

Recommended

```
CREATE Table entity (
   [entity_id]        int not null
  , description       nvarchar(4000)
  , CONSTRAINT pk_entity PRIMARY KEY CLUSTERED      (  [entity_id] )
)
```

Attribute

A property or characteristic

- Color
- Price
- Blood Pressure
- Width
- Favorite food

Representing Attribute

attribute_id smallint not null — Recommended

attribute_name varchar(255) not null — This can get you into trouble

Recommended

```
CREATE Table attribute (
    attribute_id          smallint not null
  , attribute_name        varchar(255) not null
  , data_type             char(1) CHECK (data_type in ('C', 'D', 'F'))
  ,  unit_id              small_int NULL FOREIGN KEY REFERENCES (Unit_id)
  , description           varchar(4000)
  , CONSTRAINT pk_attribute PRIMARY KEY CLUSTERED     (  [attribute_id])   )
```

Value

Magnitude or choice-from-a-list of an attribute for an entity

- 37.454
- Red
- VK98312B8
- 2016-03-19 13:35:01.912943

Representing Value

```
value          varchar(255) not null
value          float not null
```

```
, val_type      tinyint not null
, val_number    float       null
, val_string    varchar(255) null
, val_datetime  datetime2(7) null
, value as case val_type WHEN 1 THEN val_string
                         WHEN 2 THEN CONVERT(varchar(30), val_number, 128)
                         WHEN 3 THEN CONVERT(varchar(30), val_datetime, 121)
                         ELSE NULL END
```

Time

- Single Date
- Date Implied Range
- Start Date-time (IR)
- Date Range
- Date-time Range

```
, asof_date          date

, start_date         date

, start_datetime     datetime

, begin_date         date
, end_date           date

, begin_datetime     datetime
, end_datetime       datetime
```

Multiple Times

- When did it happen?
- When did we know it? - Belief Date
- When did that change?

Time Alternatives Demo

The PIVOT issue

- The data looks like this:

	entity...	attribute...	name	value
1	1	192	Incpetion_Year	2005
2	1	222	Color	Magenta
3	1	394	Elivation	1
4	1	999	Budget	120000

- The users want to see this:

	entity...	Color	Inception_Y...	Budget	Elevation
1	1	Magenta	2005-01-01	120000.000000	1.000000
2	2	Black	1956-01-01	39212.000000	7.000000

WHY USE EAV?

Why use Entity-Attribute-Value

- Adding attributes without schema change
- More attributes than allowable columns
- Use more attributes than usable columns in a sparse table

More Good Reasons to use EAV

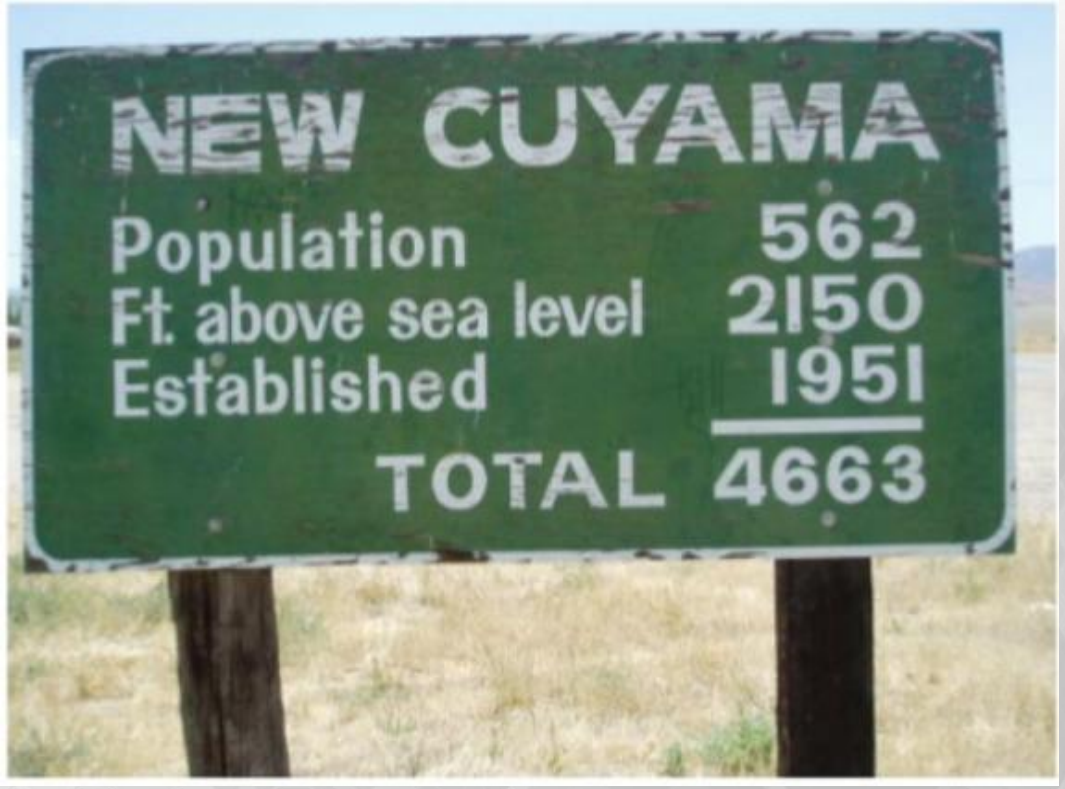
- One or a few ETL programs is enough
- Efficient storage of Temporal data
- Efficient storage of sparse data

EAV IS AN ANTIPATTERN!

- My experience is that the promised flexibility of such models is illusive and more than offset by the penalties and inconveniences they incur.

- “Bad Solution” Bill Karwin in “SQL Anti-Patterns Strike Back”

- In other words, EAV gives you enough rope to hang yourself and in this industry, things should be designed to the lowest level of complexity because the guy replacing you on the project will likely be an idiot.



Yes, Responsibility is placed on the SQL layer and apps to enforce proper data use

Referential Integrity Doesn't Work on Value

- Foreign Key
- Unique Key
- Data Types – If you use a VARCHAR
- CHECK constraints
- Default constraints

Yes, the responsibility is placed on the SQL layer to enforce constraints.
The ETL is the best place to put most of it. Supply Procs for the rest.

Typed Value Columns don't work

```
, val_type      tinyint not null
, val_number    float        null
, val_string    varchar(255) null
, val_datetime  datetime2(7) null
, value as case val_type WHEN 1 THEN val_string
                         WHEN 2 THEN CONVERT(varchar(30), val_number, 128)
                         WHEN 3 THEN CONVERT(varchar(30), val_datetime, 121)
                         ELSE NULL END
```

It puts the burden of knowing the attribute's data type on the application

Yes, the responsibility can be placed on the SQL layer and meta-data.

Representing NULL values is difficult

Using NOT NULL isn't a good idea

It works if you use datetime ranges. NULL is indicated by not having a row

Searches Don't Scale

- You must hard-code each attribute name
  - One JOIN per attribute

```
SELECT a.entity_id
      ,  a.value as title
      ,  y.value as production_year
      , r.value as rating
      , b.value as budget
FROM EAV as a
  JOIN EAV  y on a.entity_id = y.entity_id
   JOIN EAV r  on a.entity_id = r.entity_id
   JOIN EAV b on  a.entity_id = r.entity_id
```

Using dynamic SQL and a CASE based PIVOT, nothing is hard coded and there are no self joins.

Searches Don't Scale

- Alternatively, you can query all attributes, but the result is one attribute per row:
  … and sort it out in your application

Using dynamic SQL and a CASE based PIVOT the application can receive a table very much like a traditional database model. The SQL layer is responsible for the heavy lifting.

Using EAV for OLTP

- Sufficient throughput can't be sustained with an EAV schema

I agree. EAV is better suited to Data Warehouse situations.

Why EAV doesn't work for OLTP….. Yet

- Locking
- The simultaneous insert/update of a few dozen rows locks a similar number of places in the EAV table, including the index pages.
- In-Memory OLTP (Hekaton) might fix that

Additional Negatives

- SQL operations are complex — Yes, they are more complex
- Application code required to reinvent features that the RDBMS provides — Yes, put it in the SQL layer
- PIVOTS required — Yes, PIVOTS are required

It promotes really bad practices

- Storing multiple entities in the same table

Be careful! Use multiple EAV tables or add an entity_type column

- Multiple names for the same attribute

attribute_name varchar(255) not null

```
INSERT EAV ('ANOVICK', 'TEMPRATURE', '98.6')
INSERT EAV ('ANOVICK', 'TEMP', '99.1')
```

Don't do that! Use integer attribute_id and a metadata table with foreign key.

OVERCOMING EAV ISSUES

Issues to overcome

- Query Complexity for Application Developers
- The database can't enforce rules
- ETL speed
  - UNPIVOT required
  - lots of rows
- Query Speed
  - PIVOT required
- Efficient Storage

SQL Application Layer for EAV

- Procs for all Updates/Inserts/Deletes
  - Enforce RI rules
- Provide ETL as part of the “database”
  - More RI rules
- Provide a query API, Procs or Functions
  - Make it easy to get what Apps need

Schema Design That I Use for EAV

- Partitioning
  - Recent Data Partitioned by attribute_id
  - Historical data partitioned by end_datetime
  - Partitioned View to bring them together
- Second Index (Sometimes)
  - Add index starting with entity_id

SQL Code Techniques

- PIVOT code
  - Parallel queries
  - CASE based PIVOT
  - Dynamic SQL
- ETL code
  - SQLCLR
  - Service Broker

EAV Schema

- Starting Point:

```
CREATE Table eav (
    entity_id       int not null
  , attribute_id    smallint not null
  , value           varchar(255) not null
  , begin_datetime  datetime not null
  , end_datetime    datetime not null
    )
```

- This becomes 2 tables and a View

History Table Partitioned on end_datetime

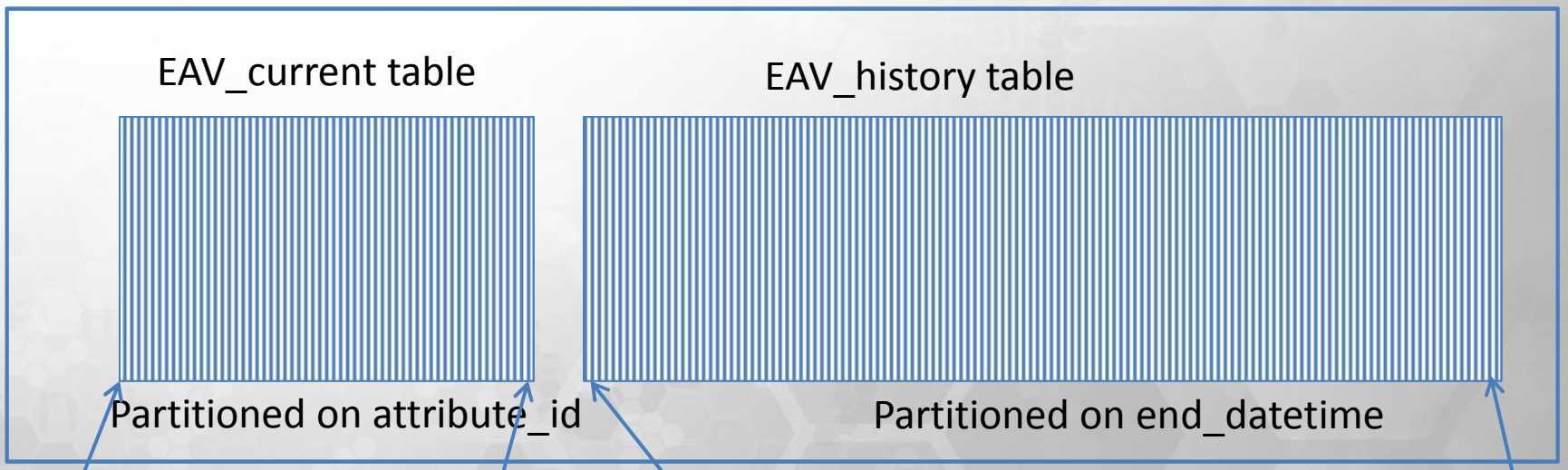
```
CREATE Table eav_history (
    [entity_id]    int not null
  , attribute_id   smallint not null
  , value          varchar(255) not null
  , begin_datetime datetime not null
  , end_datetime   datetime not null
  , CONSTRAINT pk_eav_history PRIMARY KEY CLUSTERED
        (attribute_id, begin_datetime, [entity_id], end_datetime)
    ) ON ps_eav_history_on_history_filegroups (end_datetime)
```

Current Table Partitioned on attribute_id

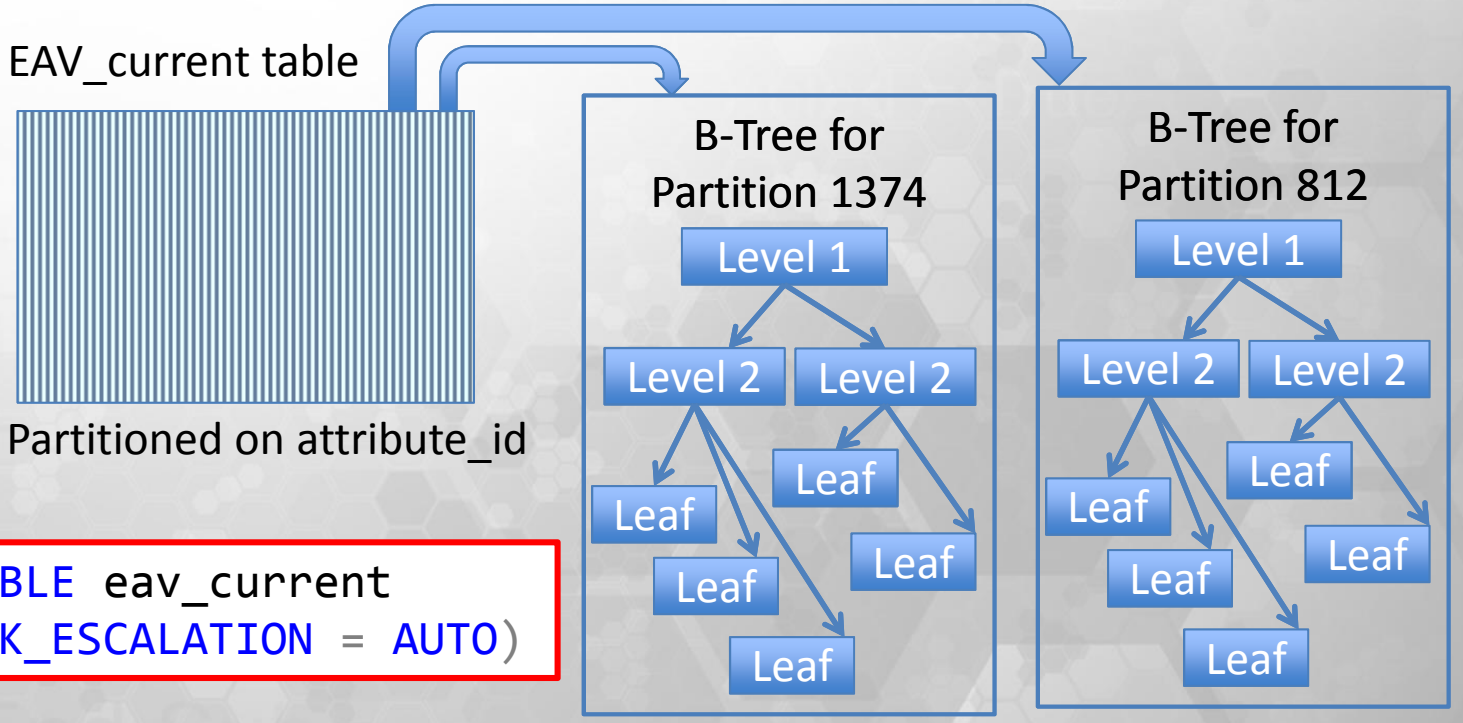
```
CREATE Table eav_current (
    [entity_id]      int not null
  , attribute_id   smallint not null
  , value          varchar(255) not null
  , begin_datetime datetime not null
  , end_datetime   datetime not null
  , CONSTRAINT pk_eav PRIMARY KEY CLUSTERED
               (end_datetime, [entity_id], attribute_id)
    ) ON ps_attribute_id_on_user_tables(attribute_id)
```

EAV Table Partitioning

CREATE View EAV AS SELECT * from EAV_current UNION ALL SELECT * from EAV_history



Partitioned Table Details




```
ALTER TABLE eav_current
SET (LOCK_ESCALATION = AUTO)
```

The PIVOT issue

- The data looks like this:

	entity...	attribute...	name	value
1	1	192	Incpetion_Year	2005
2	1	222	Color	Magenta
3	1	394	Elivation	1
4	1	999	Budget	120000

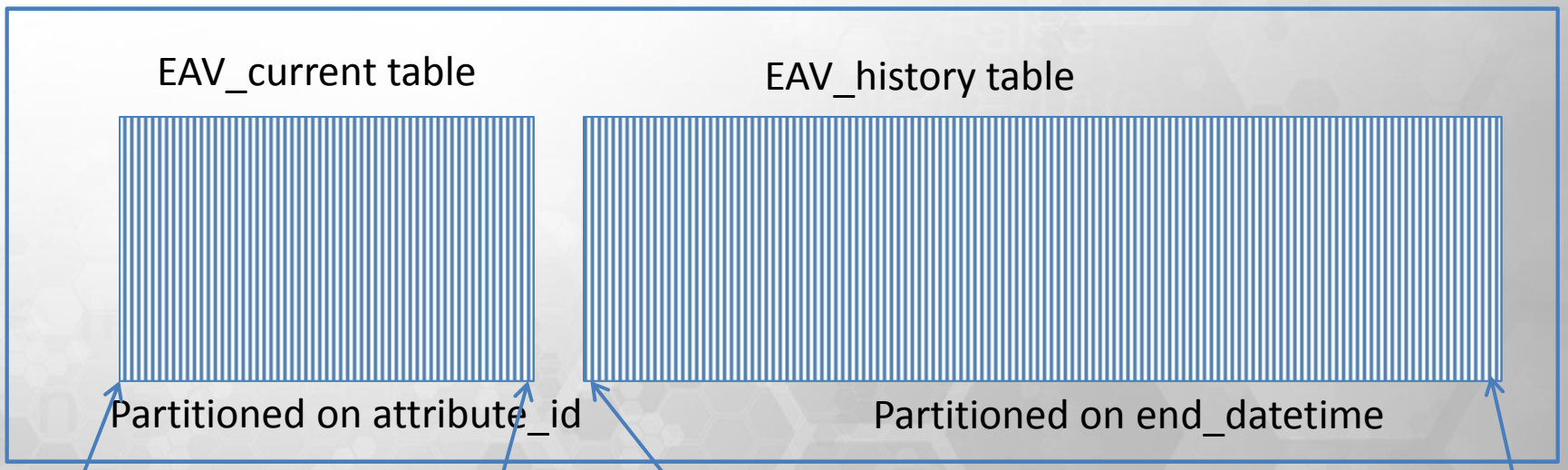
- The users want to see this:

	entity...	Color	Inception_Y...	Budget	Elevation
1	1	Magenta	2005-01-01	120000.000000	1.000000
2	2	Black	1956-01-01	39212.000000	7.000000

PIVOT Demo

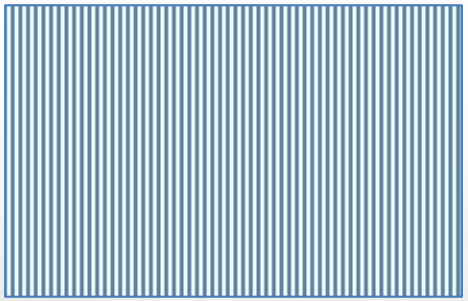
EAV Table Partitioning

CREATE View EAV AS SELECT * from EAV_current UNION ALL SELECT * from EAV_history



Querying EAV_current

EAV_current table



Partitioned on attribute_id

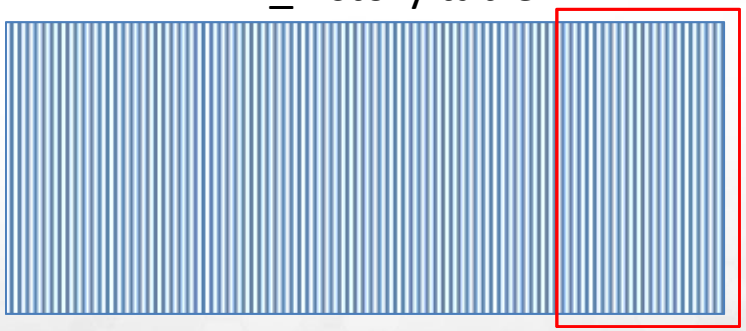
```
SELECT * FROM eav
WHERE a.attribute_id = eav.attribute_id
  and eav.entity_id = a.entity_id
  and begin_datetime <= @target_datetime
  and end_datetime  > @target_datetime
```

```
, CONSTRAINT pk_eav_current PRIMARY KEY CLUSTERED
    (end_datetime, [entity_id], attribute_id)
) ON ps_attribute_id(attribute_id)
```

- Partition elimination targets one attribute_id's HOBT
- Seek on end_datetime starting at @Target_datetime

Querying EAV_History

EAV_history table



Partitioned on end_datetime

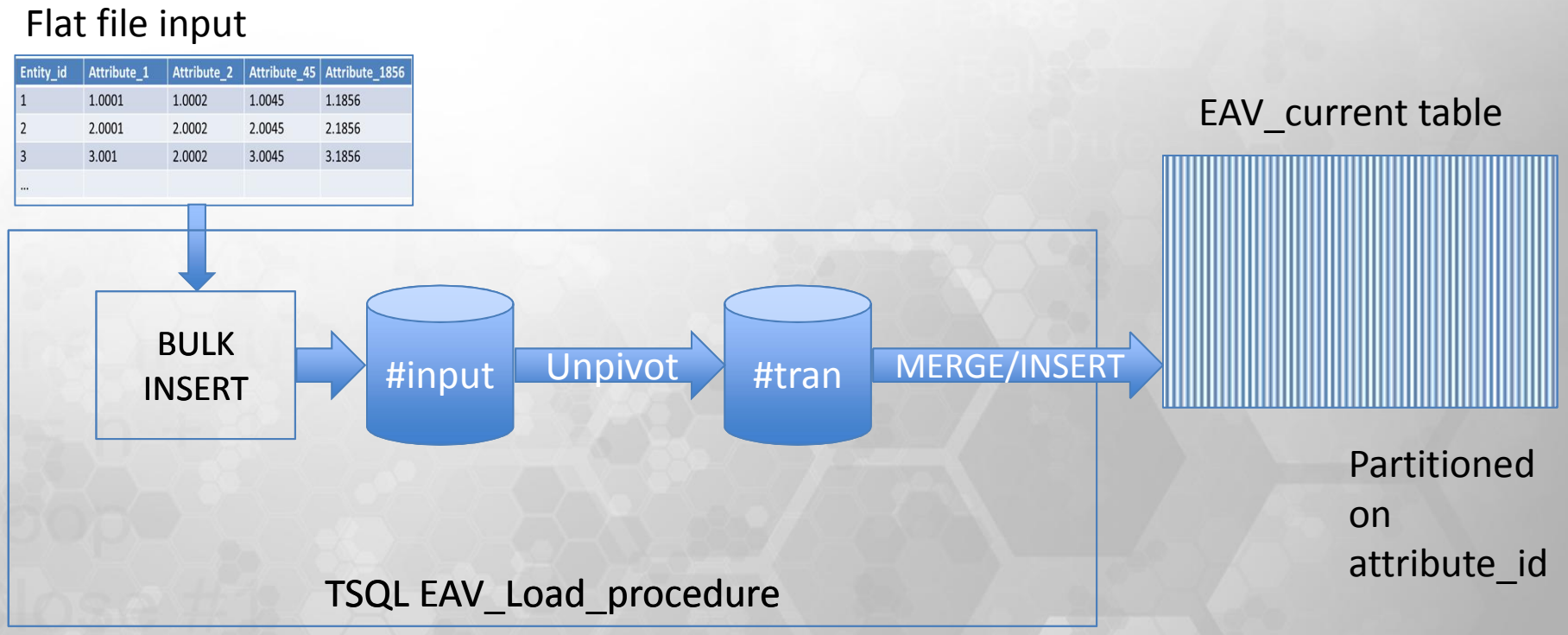
```
SELECT * FROM eav
WHERE a.attribute_id = eav.attribute_id
  and eav.entity_id = a.entity_id
  and begin_datetime <= @target_datetime
  and end_datetime  > @target_datetime
```

```
, CONSTRAINT pk_eav_history PRIMARY KEY CLUSTERED
        (attribute_id, begin_datetime, [entity_id], end_datetime)
    ) ON ps_eav_history_on_history_filegroups (end_datetime)
```

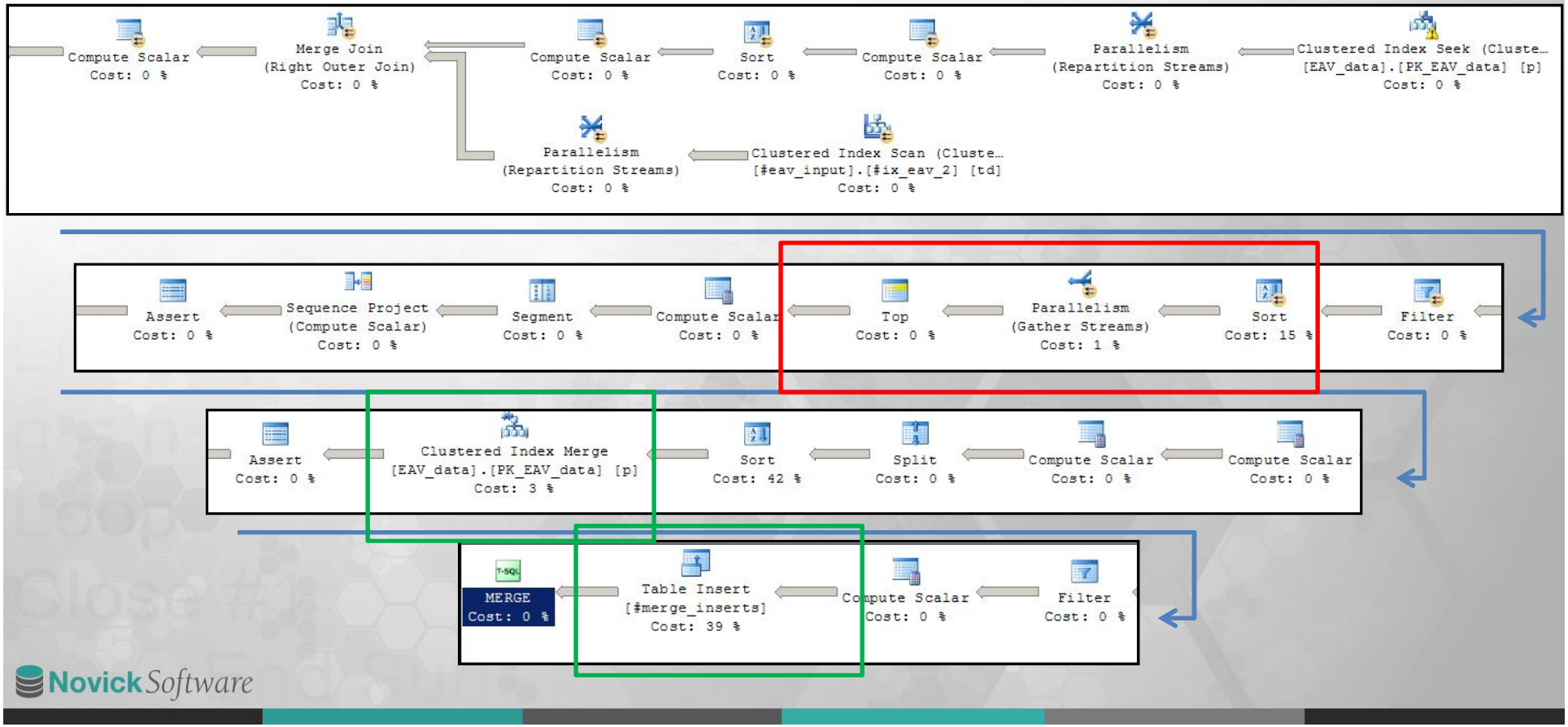
- Partition elimination selects all partitions with end_datetime boundaries > @target_datetime
- Within each partition start by seeking on attribute_id from the start of that attribute_id to the point where begin_datetime <= @target_datetime

- ETL Technique

ETL Process Block Diagram



But it's slow! This is Why



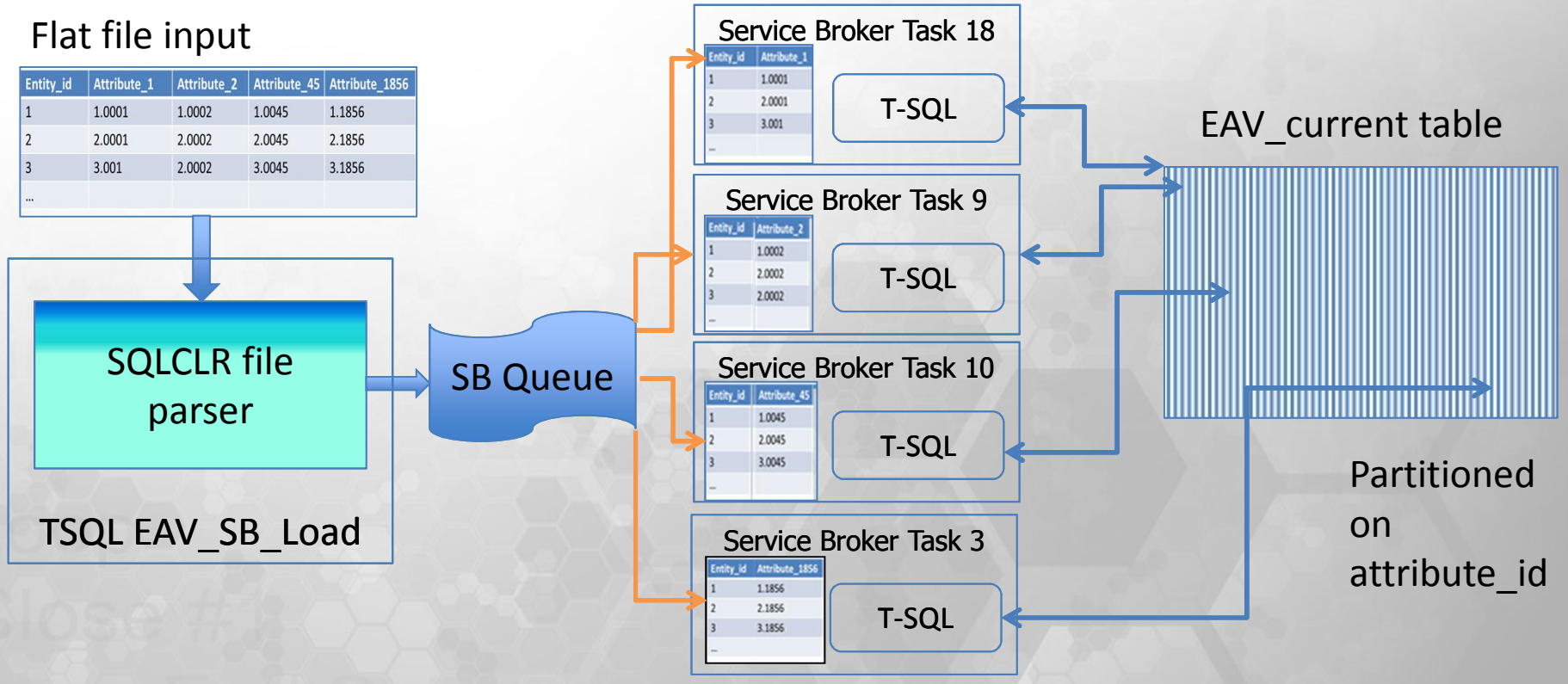
Novick Software

And then… Light Dawns Over Marble Head

Break it down to single partition operations

- SQLCLR proc breaks the file by attribute_id
- SEND attribute_id's data to a Service Broker QUEUE
- Each task is working on ONE attribute_id
  - That's one HOBT / Partition
- Run 1-2 tasks per core

What does that look like?



EAV Load Results

- Before: 26 minutes
- After: 4 to 5 minutes
- More Cores = Faster Loads
- Until we hit the next bottleneck

Resources

- HHS Report on EAV in Clinical Context
  http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2110957/
- SQL Anti-Patterns Strike Back – Bill Karwin
  http://www.slideshare.net/billkarwin/sql-antipatterns-strike-back
- SQL Antipatterns – Avoiding the Pitfalls of Database Programming
  - Book by Bill Karwin

Conclusions

- EAV is great in the right situation
- Plenty of tradeoffs to be made
- Plenty of land-mines to avoid
- The problems can be overcome

Andy Novick
anovick@NovickSoftware.com

New England Microsoft Developers

- First Thursday of the Month 6:30 to 8:30
  - Foliage: 20 North Ave. Burlington, MA
  - http://www.meetup.com/NE-MSFT-Devs/
- April 7th - Introduction to R
- May 5th – Microsoft Evangalist
- June 2nd – Microsoft Evangalist

anovick@NovickSoftware.com
http://www.NovickSoftware.com

Thank you for coming!