ADDING HISTORY TO CRUD

(Really)

DINO ESPOSITO
Dino Esposito
@despos

Far before #ArtificialIntelligence we need intelligence. Far before #deeplearning we need #software #architecture able to collect #bigdata

9:03 AM - May 12, 2017
Why would you like to study medicine?

I want to defend mankind from diseases!

You have to study computer science then.

All I do as a researcher is analytics on bigdata.
WHERE’s YOURS?
Every time you **update** an existing table record, you automatically **lose track** of the state it had before.

Ever heard about GDPR? Well, they claim every app should be tracking all processing being done on data.
Data is the most valuable **asset** of every company and the input of any **business intelligence** processes.

No, you were not attending the wrong talk!
WHAT’s NEXT?

- Rewriting CRUD?
- Replacing CRUD?
- Evolving CRUD?
CUSTOM API for SOFT UPDATES and DELETES

- A soft update is a standard update operation that preserves in some way the old state.
- A soft delete is an update that marks the state of the record

USE an ad hoc DBMS

- ANSI SQL 2011 standard
- Historical data management
TEMPORAL TABLES in SQL SERVER 2016+
Temporal Tables

- Child history table automatically created
- Read-only for dev code, can’t be deleted
- Historical data can be queried through ad hoc SQL commands
- CASCADE not supported if tables are child tables in a foreign key relationship
- Can’t drop a temporal table
  - Unless you first reset it back to non-temporal state

CREATE TABLE dbo.Employee
(
    [EmployeeID] int NOT NULL PRIMARY KEY CLUSTERED,
    [Name] nvarchar(100) NOT NULL,
    [Position] varchar(100) NOT NULL,
    [Department] varchar(100) NOT NULL,
    [Address] nvarchar(1024) NOT NULL,
    [AnnualSalary] decimal (10,2) NOT NULL,
    [SysStartTime] datetime2 GENERATED ALWAYS AS ROW START,
    [SysEndTime] datetime2 GENERATED ALWAYS AS ROW END,
    PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime)
) WITH (SYSTEM_VERSIONING = ON
    (HISTORY_TABLE = dbo.EmployeeHistory));
DateTime columns can’t be NULL and can be arbitrarily named

Must be marked as GENERATED ALWAYS AS ROW

Reserved to the system

The history table cannot have a primary key and can be created programmatically

Default history tables have a clustered index on datetime columns (PERIOD)

Custom history should also have clustered index for query perf and compression
ALTER TABLE dbo.Employee
SET (SYSTEM_VERSIONING = OFF)
History Table

• Copy of the main table, plus a couple of `datetime2` columns
• Date columns indicate the validity period for that particular state of the record

  • `ValidFrom` indicates when the record got a given state
  • `ValidTo` indicates when the validity of that state ceased

• Updates and deletes are database operations that cause the values in columns to change
### Temporal Data

**SQL Query:**
```
UPDATE bookings SET Hour=13 WHERE id=2
```

**Information:**
```
select * from Bookings where id=2
select * from BookingsHistory where id=2
```

**Table 1:**
<table>
<thead>
<tr>
<th>Id</th>
<th>RoomName</th>
<th>Hour</th>
<th>Length</th>
<th>Owner</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Corner Room</td>
<td>13</td>
<td>1</td>
<td>Dino</td>
<td>NULL</td>
</tr>
</tbody>
</table>

**Table 2:**
<table>
<thead>
<tr>
<th>Id</th>
<th>RoomName</th>
<th>Hour</th>
<th>Length</th>
<th>Owner</th>
<th>Motivation</th>
<th>SysStartTime</th>
<th>SysEndTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Corner Room</td>
<td>9</td>
<td>1</td>
<td>Joe</td>
<td>NULL</td>
<td>2017-09-27 22:04:35</td>
<td>2017-09-27 22:14:33</td>
</tr>
</tbody>
</table>
Entity Framework and Temporal Tables

- What’s not there yet (and won’t for more time to come...)
Fact 1: It’s an all-columns-or-nothing thing.
Fact 2: CASCADE limitation removed in SQL2017.
Fact 3: With BLOB columns can be painful size-wise*. 
Fact 4: Requires direct SQL commands.
Fact 5: In EF Core, can mix with IQueryable.
Fact 6: Basic step towards Event Sourcing.
Fact 7: Still CRUD.
History tables don’t include any of the following:

- Indexes
- Statistics
- Triggers
- Permissions
Memory-optimized Temporal Tables use in-memory tables for storing current data (the temporal table) and disk-based tables for historical data.

MEMORY_OPTIMIZED = ON, 
DURABILITY = SCHEMA_AND_DATA
added to T-SQL definition

Event Sourcing captures all changes to an application state as a sequence of events.

State transitions are an important part of our problem space and should be modelled within our domain.
GREAT INDIAN DEVELOPER SUMMIT 2019
Conference: April 23-26, Bangalore
Register early and get the best discounts!

www.developersummit.com
facebook.com/gids19
@greatindiandev
bit.ly/gidslinkedin
bit.ly/saltmarchyoutube
flickr.com/photos/saltmarch/