A Brief History of the Creation of a Time Traveling Database

Chris Cumming

Who am I? (i.e. Shameless Self Promotion)

Saturday Morning Productions - Consultant

http://nftb.saturdaymp.com

chris.cumming@satudaymp.com

Edmonton .NET Users Group – Program Director

http://edmug.net

The Problem

Create application to adjudicate claims

Claims can be submitted months or even years later

Bunch of other normal business requirements

What Data Needs History?

Customers, Addresses, Relationships,

Companies, Service Providers, Bargaining Agreements,

Coverage Plans, Member Options, Lines of Coverage,

Eligibility, Fee Guides (Dental, Extended Health, etc.),

Fee Codes, Diagnosis Codes, Claim Evidence,

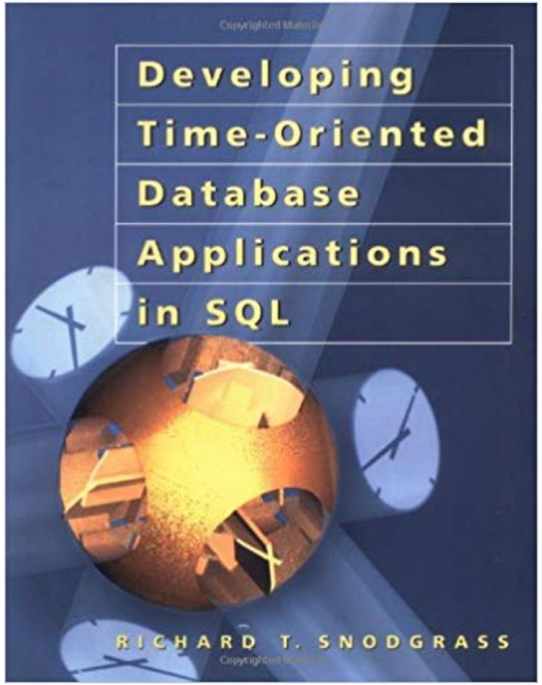
Rule Arguments, Coordination of Benefits

Found the Mystic Tome

Developing Time-Oriented Database Applications in SQL

Richard T. Snodgrass

PDF Version:
https://www2.cs.arizona.edu/people/rts/tdbbook.pdf



History Database Requirements

Fast queries for claim adjudication

Allow gaps

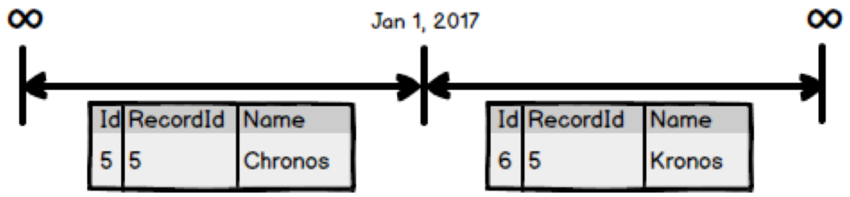
Allow future dating

Day level precision

Work with existing DB (i.e. SQL Server) and Reporting tools

Core Design Rules

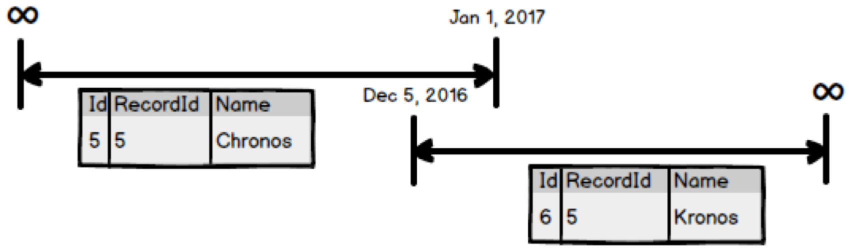
Timelines, records, and segments



Id	RecordId	Name
5	5	Chronos

Id	RecordId	Name
6	5	Kronos

Segments can't overlap

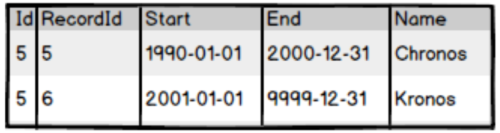


Id	RecordId	Name
5	5	Chronos

Id	RecordId	Name
6	5	Kronos

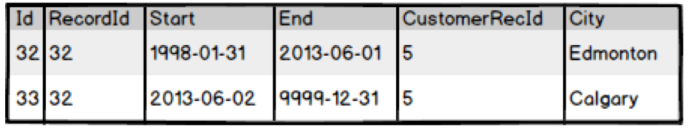
Foreign key integrity

Customers



Id	RecordId	Start	End	Name
5	5	1990-01-01	2000-12-31	Chronos
5	6	2001-01-01	9999-12-31	Kronos

Address



Id	RecordId	Start	End	CustomerRecId	City
32	32	1998-01-31	2013-06-01	5	Edmonton
33	32	2013-06-02	9999-12-31	5	Calgary

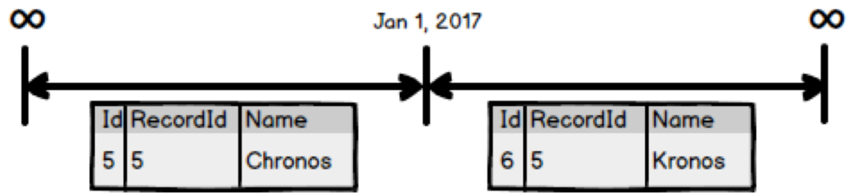
Creating Temporal Tables

Table has Id, RecordId, StartDate, and EndDate fields.

Id is unique, not null, primary key, and auto incremented.

RecordId is shared across segments.

```
CREATE TABLE Customers
(
  Id        INT  NOT NULL IDENTITY PRIMARY KEY,
  RecordId  INT  NULL,
  StartDate DATE NOT NULL,
  EndDate   DATE NOT NULL,
  Name      VARCHAR(100)
)
GO
```



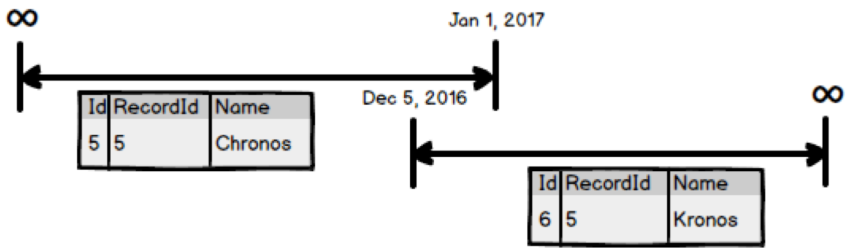
Id	RecordId	Name
5	5	Chronos

Id	RecordId	Name
6	5	Kronos

Overlapping Segments Trigger

Use tool to create triggers for all temporal tables

```
CREATE TRIGGER TR_<Table>_OverlappingSegments ON <Table> FOR UPDATE, INSERT AS
 IF EXISTS(
     SELECT *
     FROM <Table> t
     INNER JOIN inserted i On i.RecordId = t.RecordId
       AND t.Id <> i.Id
       AND t.StartDate <= i.EndDate
       AND t.EndDate >= i.StartDate
   )
   BEGIN
     RAISERROR ('Tried to insert overlapping segments in <Table>  table.', 16, 1);
     ROLLBACK;
   END
GO
```



Id	RecordId	Name
5	5	Chronos

Id	RecordId	Name
6	5	Kronos

Foreign Key Triggers

Two triggers

Delete trigger for parent

Insert/update trigger for child

Use application to create triggers

Delete Trigger

```
CREATE TRIGGER TR_Customers_Addresses_ForeignKey_D ON Customers FOR DELETE AS
 IF NOT EXISTS(
  SELECT *
  FROM Customers
  Where RecordId IN (
   SELECT CustomerRecId
   FROM Addresses
   INNER JOIN deleted On deleted.RecordId = CustomerRecId
   )
  )
  BEGIN
   RAISERROR ('Tried to deleted Customers record that is referenced by Addresses forgien key.', 16, 1);
   ROLLBACK;
  END
```

Insert/Update Trigger

```
CREATE TRIGGER TR_Addresses_Customers_ForeignKey_IU ON Addresses FOR INSERT, UPDATE AS
 IF NOT EXISTS(
  SELECT *
  FROM Customers
  Where RecordId IN (
   SELECT CustomerRecId
   FROM inserted
   )
  )
  BEGIN
   RAISERROR ('Tried to insert/update Addresses record that had a invalid forgien key to the Customers table.', 16, 1);
   ROLLBACK;
  END
```

Writing Queries

Query by RecordId and Query Date

```
Select * From Customers
Where RecordId = #
And StartDate <= ‘2000-03-15’
And EndDate >= ‘2000-03-15’
```

Join by RecordId’s, not by IDs

```
Select * From Customers c
Inner Join Addresses a On a.CustomerRecId = c.RecordId
And c.StartDate <= ‘2000-03-15’
And c.EndDate >= ‘2000-03-15’
And a.StartDate <= ‘2000-03-15’
And a.EndDate >= ‘2000-03-15’
```

Next Steps

Example on GitHub:

https://github.com/saturdaymp-examples/a-brief-history-of-the-creation-of-a-time-traveling-database

Other Items to Consider:

-Not all data need temporality (i.e. financial tables).

-Joining to non-temporal tables.

-Fields that shouldn’t change (i.e. birthdate).

-Effective vs Entered Temporality.