Chris Cumming
Saturday Morning Productions

Create a Time Traveling Database

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

Saturday Morning Productions

Consultant/Owner

http://www.saturdaymp.com

chris.cumming@saturdaymp.com

Edmonton .NET Users Group

Program Director

http://edmug.net

info@edmug.net

PASS SQLSATURDAY EDMONTON | APR 27 2019

EDMPASS User Group

http://EDMPASS.PASS.Org

Meetings Every 3rd Thursday 5:30 – 7:30

Canadian Western Bank Conference Centre
Lower Concourse (Basement)
10303 Jasper Avenue NW

Thank Our Sponsors

PASS SQLSATURDAY EDMONTON | APR 27 2019

Data Changes Over Time

The only thing that is constant is change.

- Heraclitus

Claim Adjudication

Claims submitted months later.

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

What is a Temporal Database?

Designed for data that changes over time.

Allow past, present, future data and allow gaps.

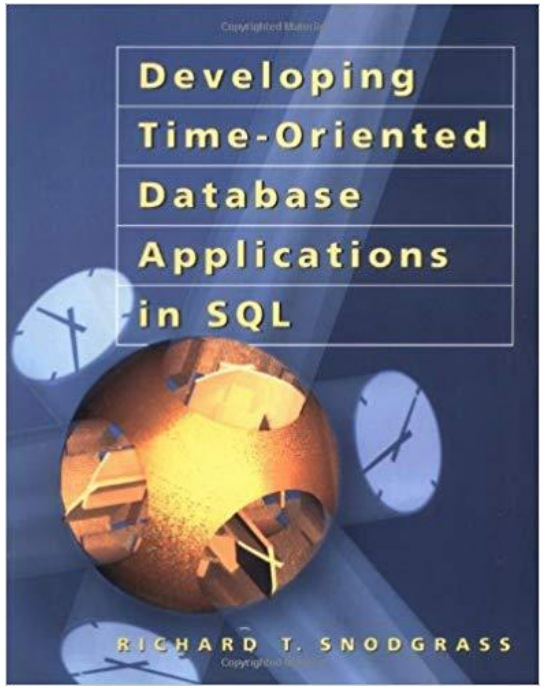
Use existing database (SQL Server 2005) and tools such as reporting.

Main Reference

Developing Time-Oriented Database Applications in SQL

Richard T. Snodgrass

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



Types of Temporalness

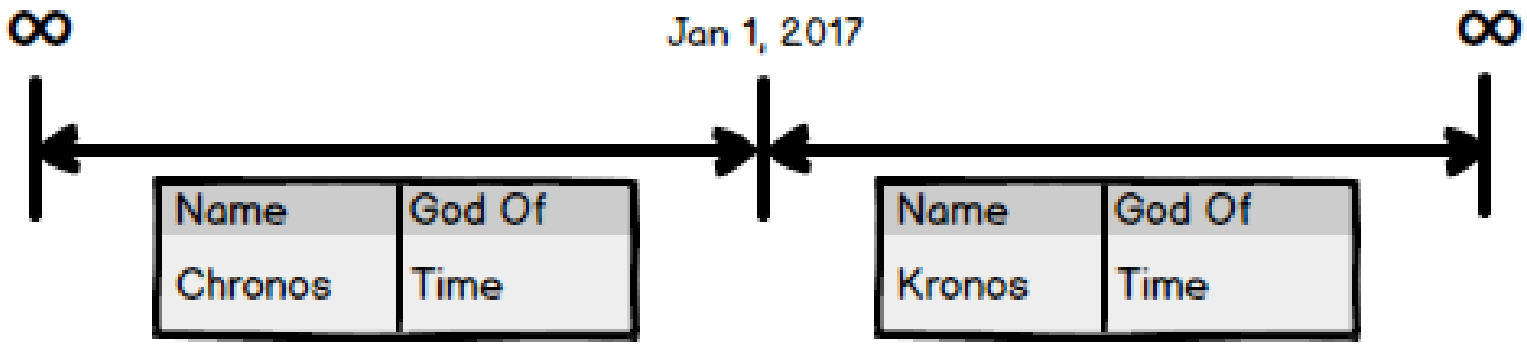
Valid (Effective): When the data is true in the real world.

Transaction (Entered): When was the data entered into the database.

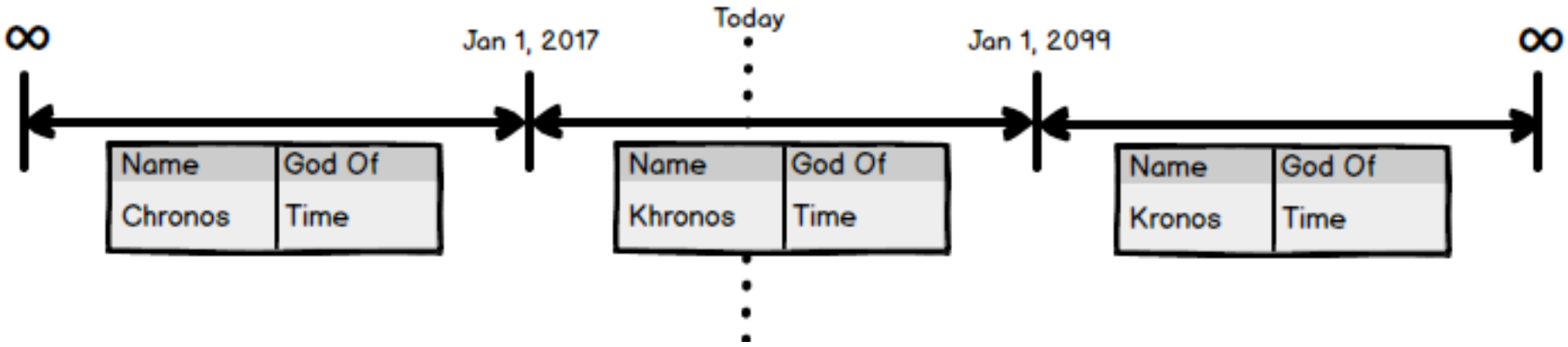
Timelines

Visually represent data changing over time.

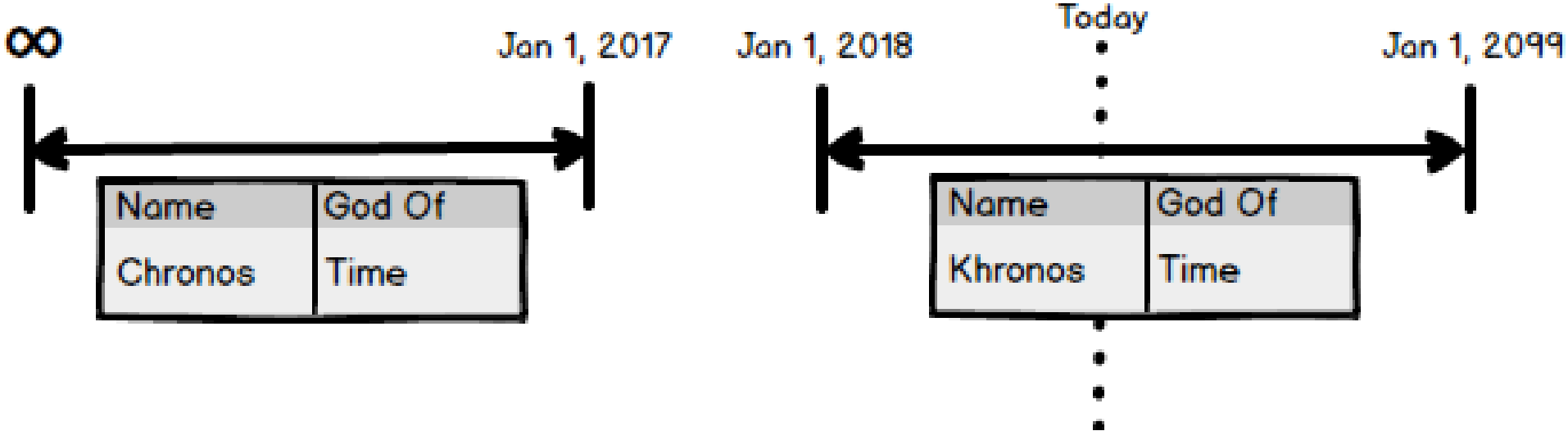
Made up of intervals.



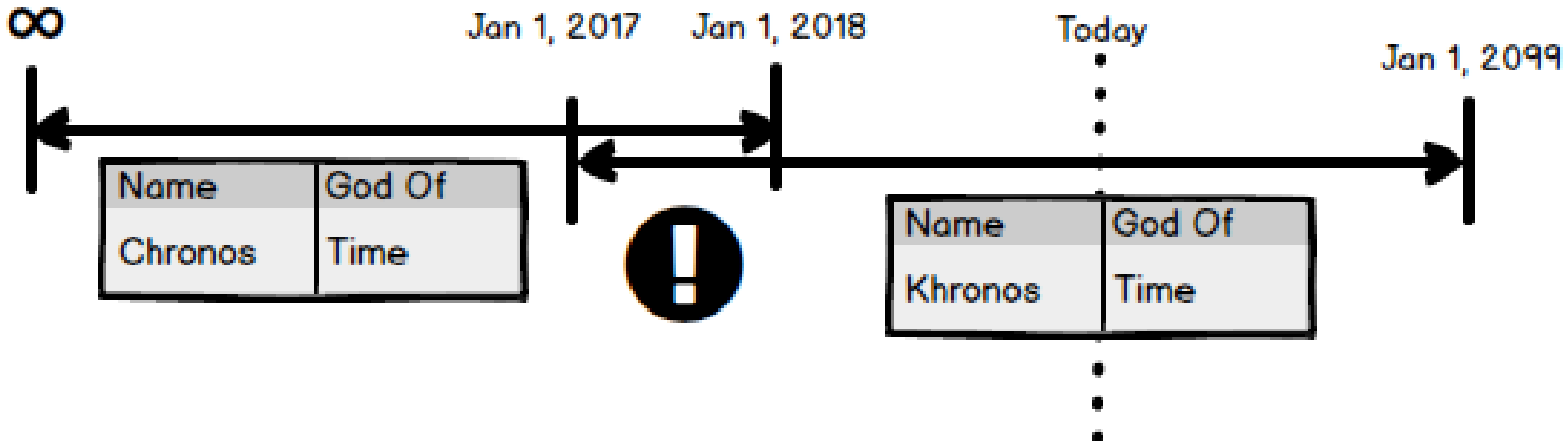
Timeline Properties: Past, Present and Future Intervals



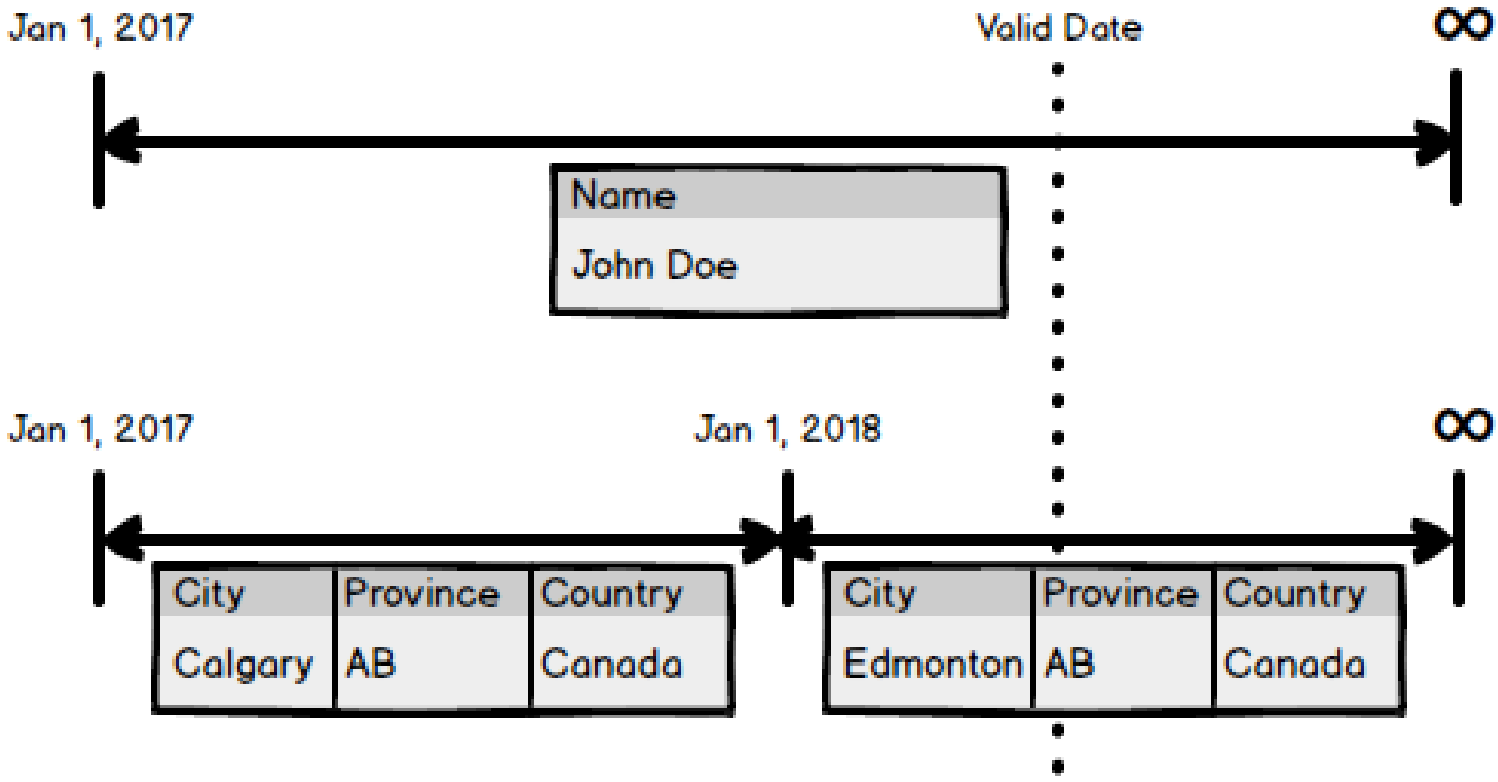
Timeline Properties: Gaps are Allowed



Timeline Properties: Intervals Can’t Overlap!



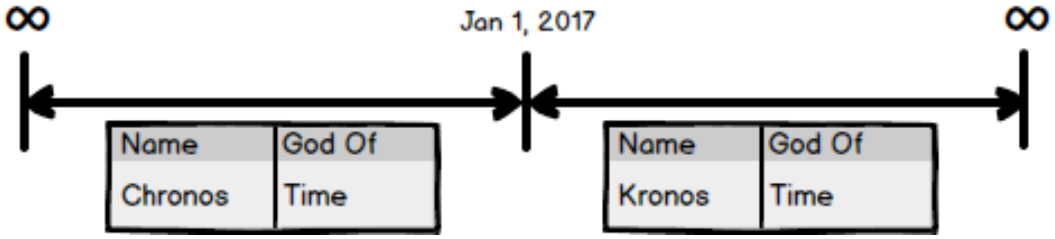
Multiple Timelines



Theory to Practice

Timelines become a record in a table.

Each interval becomes a row in the table.



Id	RecordId	StartDate	EndDate	Name	God Of
5	5	Jan 1 ,1753	Dec 31,2016	Chronos	Time
6	5	Jan 1,2017	Dec 31,9999	Kronos	Time

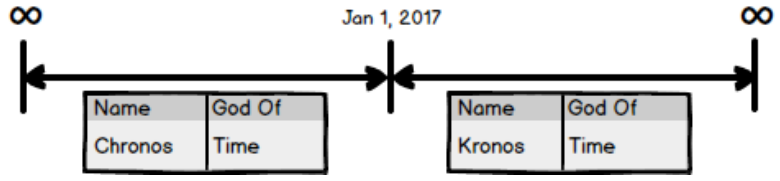
Creating a Temporal Table

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

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

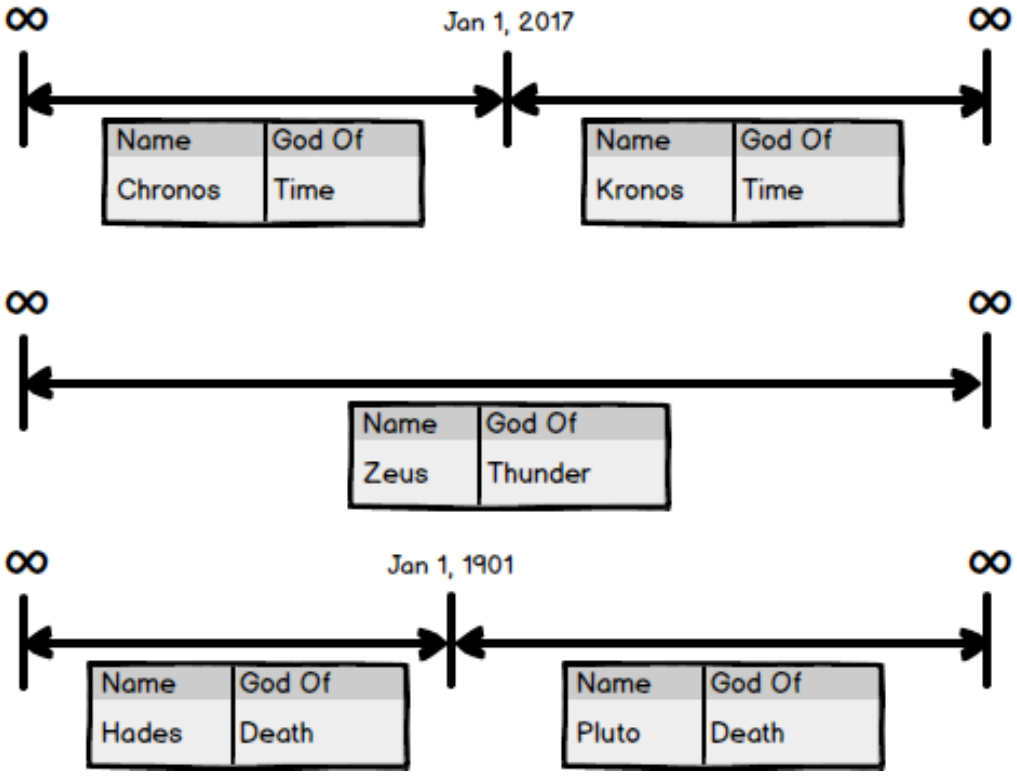
RecordId is shared across invervals.

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



Id	RecordId	StartDate	EndDate	Name	God Of
5	5	Jan 1 ,1753	Dec 31,2016	Chronos	Time
6	5	Jan 1,2017	Dec 31,9999	Kronos	Time

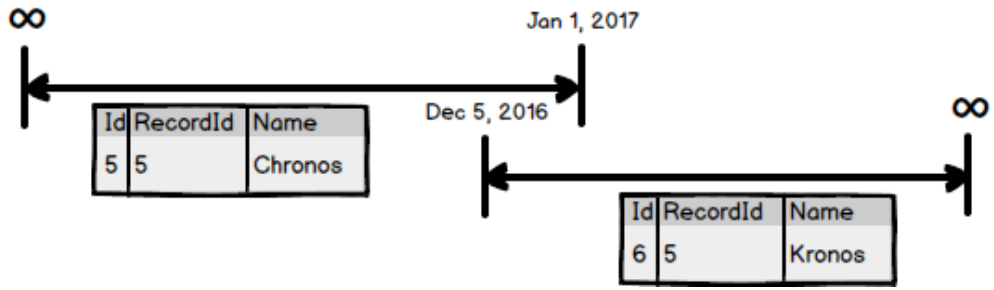
Temporal Table Example



Id	RecordId	StartDate	EndDate	Name	God Of
5	5	Jan 1 ,1753	Dec 31,2016	Chronos	Time
6	5	Jan 1,2017	Dec 31,9999	Kronos	Time
7	7	Jan 1,1753	Dec 31,1900	Hades	Death
8	8	Jan 1,1753	Dec 31,9999	Zeus	Thunder
9	7	Jan 1\1901	Dec 31\9999	Pluto	Death

Detecting Overlapping Segments

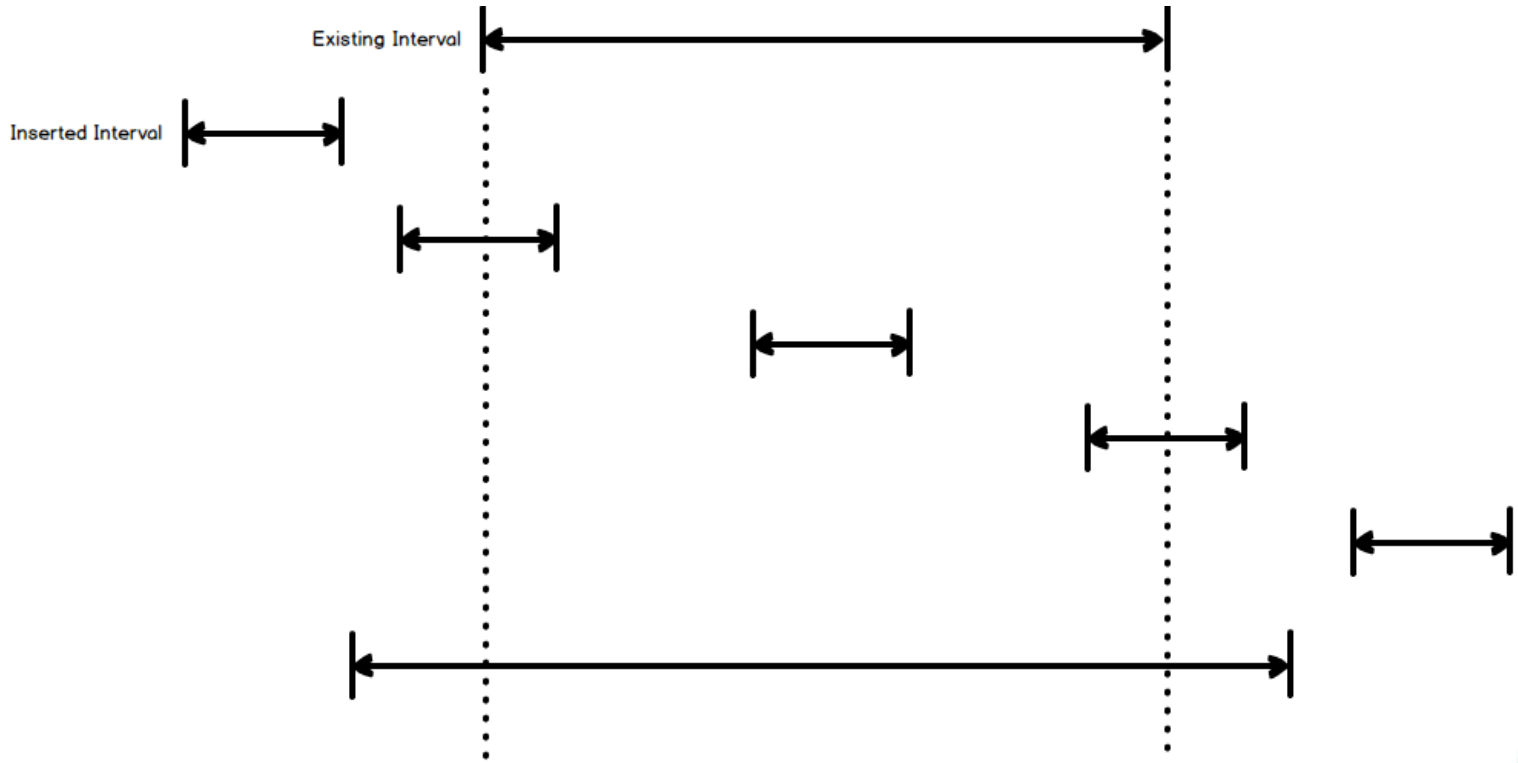
```
CREATE TRIGGER TR_<Table>_OverlappingSegments ON <Table> FOR UPDATE, INSERT AS
  IF EXISTS(
      SELECT *
      FROM <Table> t
      INNER JOIN inserted i On i.RecordId = t.RecordId
        AND t.Id <> i.Id
        AND t.StartDate <= i.EndDate
        AND t.EndDate >= i.StartDate
    )
    BEGIN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



Algorithm to Find Overlapping Segments

Inserted.EndDate >= Existing.StartDate

And Inserted.StartDate <= Existing.EndDate



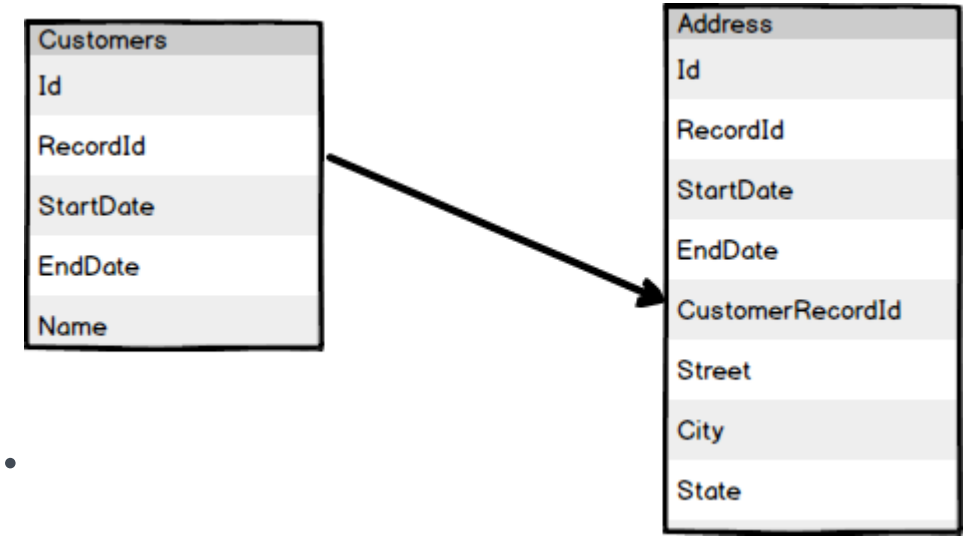
Temporal Foreign Keys

Can’t use SQL Server foreign keys.

Use Triggers instead.

Delete trigger for parent table.

Insert/Update trigger for child table.



Temporal Foreign Keys Example

Customer Table

Id	RecordId	StartDate	EndDate	Name
5	5	Jan 1 ,1990	Dec 31,2016	Jane Doe
6	5	Jan 1,2017	Dec 31,9999	Jane Doe-Deer

Address Table

Id	RecordId	StartDate	EndDate	CustomerRecordId	City
101	101	Jan 1 ,1990	Dec 31,2016	5	Edmonton
102	101	Jan 1,2018	Dec 31,9999	5	

Temporal Foreign Key Delete Trigger

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

Temporal Foreign Key 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

Same as non-temporal queries except:

Filter by valid date:

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

Join by RecordId:

```
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'
```

Not Covered

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

Joining to non-temporal tables.

Fields that shouldn't change (i.e. birthdate).

Learn More

Slides and Example Code

https://github.com/saturdaymp-examples/create-a-time-travelling-database

Contact Me

chris.cumming@saturdaymp.com
@saturdaymp

Thank you attending my presentation. Questions?