



# **“As Of” Support**

## Temporal Data Processing

# Notes

- All technical content subject to change
- All syntax examples subject to change

# As Of - Background & History

- 1989: ACM SIGMOD - Access Methods for Multiversion Data
- 2008: VLDB - Transaction Time Indexing with Version Compression
- 2011: ISO/IEC 9075:2011 - Temporal Database Support (SQL:2011)
- 2013: ISO/IEC PDTR 19075-2:2014(E) - SQL support for time- related information

# Why do you need historic versions of data?

- Operational
  - Data corruption
  - Bad data fix/patch
- Business / Use Case
  - Audit requires a financial institution to report on changes made to a client's records during the past five years.
  - Lawsuit prompts a hospital to reassess its knowledge of a patient's medical condition just before a new treatment was ordered
  - A client challenges an insurance agency's resolution of a claim involving a car accident. The agency needs to determine the policy's terms in effect when the accident occurred.
  - A client inquiry reveals a data entry error involving the three-month introductory interest rate on a credit card. The bank needs to retroactively correct the error (and compute a new balance, if necessary)
  - Need to understand how a Sales Opportunity has fluctuated over time
  - Need change history support in your API or Application

# “As Of” Project Goals

- Simple Management
  - Automatic versioning of records and foreign-key relationships
  - Automatic tracking of schema evolution (e.g. add column, drop column)
  - Selectively add versioning on a table by table basis
  - Prune/truncate version history as needed
- Eliminate application changes
  - Applications continue to work “as is”
- Standard dialect to support temporal queries
  - Use appropriate standard (i.e. SQL-2011)

# Example 1 - Change history

```
insert into emp (name, salary, dept)
values ("Bill", 1000, 10)
```

```
update emp set salary=2000
where name = "Bill"
```

```
update emp set dept=20
where name = "Bill"
```

name	salary	dept
------	--------	------

name	salary	dept
Bill	1000	10

name	salary	dept
Bill	2000	10

name	salary	dept
Bill	2000	20



```
select * from emp where name = "Bill"
```

```
select *
from emp for system_time as of timestamp @t1
where name = "Bill"
```

```
select dept, sum(salary)
from emp for system_time as of timestamp @t2
group by dept_id
```

```
select *
from emp for system_time as of timestamp @t0
where name = "Bill"
```

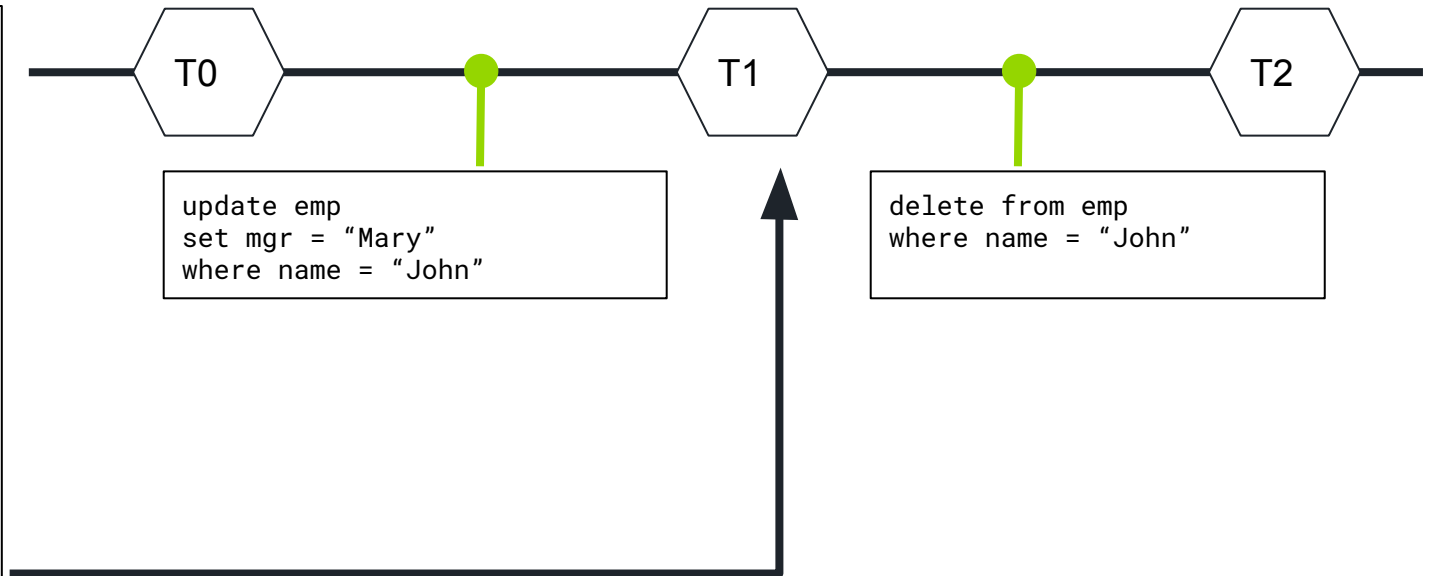
# Example 2 - Relationship navigation

name	salary	mgr
Bill	1000	
Mary	750	Bill
John	500	Bill

name	salary	mgr
Bill	1000	
Mary	750	Bill
John	500	Mary

name	salary	mgr
Bill	1000	
Mary	750	Bill

```
with recursive
orgchart
as
(
  select e.name, e.mgr, e.salary
  from emp as e for system_time as of timestamp @t1
  where e.name = 'Bill'
  union
  select e.name, e.mgr, e.salary
  from emp as e for system_time as of timestamp @t1,
       ancestors as a
  where e.mgr = a.name
)
select * from orgchart;
```



# Example 3 - Time spans

```
insert into emp (name, salary, dept)
values ("Bill", 1000, 10)
```

```
update emp set salary=2000
where name = "Bill"
```

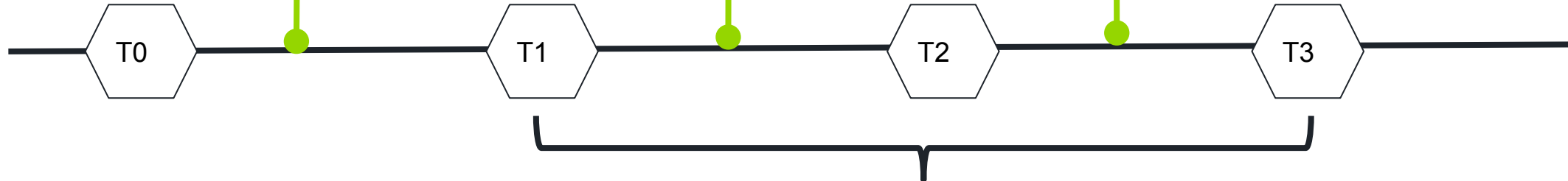
```
update emp set dept=20
where name = "Bill"
```

name	salary	dept
------	--------	------

name	salary	dept
Bill	1000	10

name	salary	dept
Bill	2000	10

name	salary	dept
Bill	2000	20



```
select name, salary, dept
from emp for system_time between timestamp @t1 and timestamp @t3
```

name	salary	dept
Bill	1000	10
Bill	2000	10
Bill	2000	20

```
select dept, sum(salary)
from emp for system_time between timestamp @t1 and timestamp @t3
group by dept_id
```

dept	sum(salary)
10	3000
20	2000



# More “As Of” support

- Tables
  - Versioned tables can use Foreign Key rules
    - ON DELETE
    - ON UPDATE
  - Column support
    - AUTO\_INCREMENT
    - Virtual Columns
    - Expressions
  - Triggers on Versioned Tables
- View definitions
  - Can use “As Of” clause
- Stored procedures
  - Queries can use “As Of” clause

# Adding & Managing Temporal Support

```
alter table emp ... with system versioning
```

```
truncate table emp  
  for system_time before timestamp '2016-02-06 10:03:03';
```

```
truncate table emp  
  for system_time timestamp  
  between '2016-01-01 00:00:01' and '2016-12-31 23:59:59';
```

# Considerations

- Update - creates versioned records
- Delete - does not remove version history!
  - Truncate removes version history
- System Performance
  - More DRAM and IO pressure
  - More Disk space
- Deployment
  - “as of” enabled on a Replica copy of the data
    - Machine can be sized / spec’ed as appropriate
    - Temporal queries directed to specific node