oddgen – Bi-Temporal Table API in Action

Philipp Salvisberg

@phsalvisberg

Philipp Salvisberg

- Trivadian since April 2000
  - Senior Principal Consultant, Partner
  - Member of the Board of Directors
  - www.salvis.com/blog
  - @phsalvisberg
- Database centric development with Oracle database
- Fond of DSLs to build full stack solutions efficiently and keep them manageable
- Author of free SQL Developer Extensions PL/SQL Cop, PL/SQL Unwrapper, oddgen and Bitemp Remodeler

trivadis makes IT easier.

Agenda

1. Introduction
2. What You Should Know About Flashback Data Archive
3. Bitemp Remodeler Data Modeling Principles
4. Temporal DML
5. Switching Models
6. Bulk Processing
7. Core Messages

Introduction

What is oddgen?

our **d**ictionary-**d**riven **gen**erator

/odˈdʒen/

Oracle community project providing
tooling for dictionary-driven code generation

What is a Model?

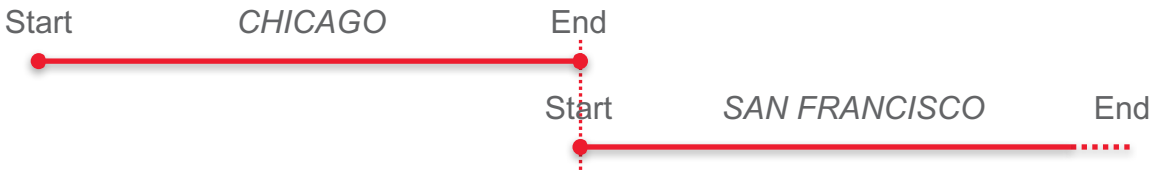
- A simplification
- With a purpose
- To be used instead of the "real thing"

Model-Driven Software Development

Reduce the problem-solution gap to
improve the efficency of software development.

Problem	Domain Specific Language	General Purpose Modeling Language	oddgen	High-level Programming Language	Low-level Programming Language	Machine Code
	DSL = Model	UML		Java C# Xtend Scala	Assembler C	

Temporal Periods – Semantics and Granularity



1981-02-20	2017-01-01	NULL (∞)
1981-02-20 00:00:00	2017-01-01 00:00:00	NULL (∞)
1981-02-20 00:00:00.000000	2017-01-01 00:00:00.000000	NULL (∞)

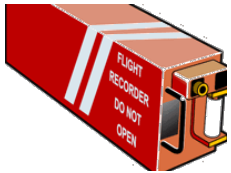
Use of Temporal Periods

Modeling Method	Data Category	Appropriate?
Entity Relationship	Master Data	Yes
	Reference Data	Yes
	Position Data	-
	Transaction Data	-
Dimensional	Dimension (SCD2)	Yes
	Fact	-
Data Vault	Hub	-
	Links	-
	Satellites (Load Start/End)	Yes

Historization Types

- Non-temporal model
  - E.g. EMP and DEPT in schema SCOTT
- Uni-temporal model
  - Typically transaction-time or valid-time
- Bi-temporal model
  - Typically transaction-time and valid-time

Transaction Time (TT)



Aka	TT, System Time, System Change Time, SCN
Start	System timestamp when the data has been entered.
End	System timestamp when the data has been outdated.
Example	Salary increase. Entered into the system on 2016-09-29 15:42:42. Preceding salary is outdated at the same time.
Charac-teristics	Data is available about previous and current states. Backdated changes are not possible (typically not wanted). Future changes are not possible. No data model changes required (if you use FBA). Transaction awareness required for a consistent picture of the past.

Consistent Picture of the Past with FBA?

SCN	Session A	Session B
1	INSERT INTO emp (empno, ename, job, sal, deptno) VALUES (4242, 'CARTER', 'CLERK', '2400', 20);	
2	SELECT COUNT(*) FROM emp; -- 15 rows	
3		SELECT COUNT(*) FROM emp; -- 14 rows
4	COMMIT; -- delayed FBA population	

SELECT COUNT(*) FROM emp AS OF SCN 3
will retrieve 14 rows

Valid Time (VT)

Aka	VT, Business Time, validity in real-world
Start	Start of validity, typically different to TT
End	End of validity, typically different to TT
Example	Salary increase. Becomes valid on 2017-01-01. Preceding salary becomes invalid on the same date.
Charac-teristics	Data is available about previous, current and future states. Backdated changes are possible. Future changes are possible. Requires a valid-time-aware data model.

What You Should Know About Flashback Data Archive

Read Consistency

ORA-01555: snapshot too old:
rollback segment number … with name "…" too small

History

- Read consistency through MVCC since version 4
- Public interface for flashback query since 9.2 (UNDO only)
- Flashback data archive since 11.1 (beyond UNDO)
- Available in all editions since 11.2.0.4 (cost-free)
- Import/export capabilities since 12.1.0.1 (OTN provided package for 11.2)

Example

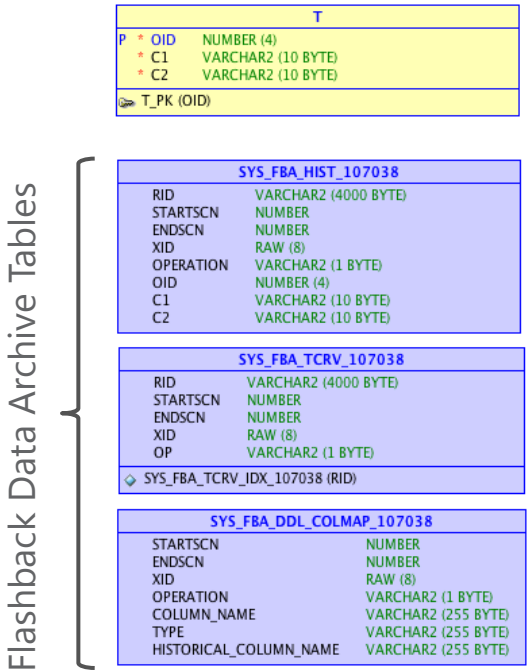
Enable FBDA

```
CREATE TABLE t (
   oid          NUMBER(4,0)  NOT NULL PRIMARY KEY,
   c1           VARCHAR2(10) NOT NULL,
   c2           VARCHAR2(10) NOT NULL
) FLASHBACK ARCHIVE fba;
```

Enforce creation of FBDA tables

```
BEGIN
   dbms_flashback_archive.disassociate_fba(owner_name => USER,
                                           table_name => 'T');
   dbms_flashback_archive.reassociate_fba(owner_name => USER,
                                          table_name => 'T');
END;
```

Flashback Archive Tables



- FBA tables asynchronously maintained by ora_fbda_<sid> process
- Recent changes are available in UNDO only
- SYS_FBA_HIST_<object_id>
  - Data history
  - Range partitioned by ENDSCN
- SYS_FBA_TCRV_<object_id>
  - Validity of latest rows in T
- SYS_FBA_DDL_COLMAP_<object_id>
  - Column history

Querying Latest Data

```
SELECT * FROM t;

-----------------------------------
| Id  | Operation          | Name |
-----------------------------------
|   0 | SELECT STATEMENT   |      |
|   1 |  TABLE ACCESS FULL| T    |
-----------------------------------
```

Flashback Query with FBA – Execution Plan

```
SELECT * FROM t AS OF TIMESTAMP SYSTIMESTAMP - INTERVAL '1' SECOND;

-----------------------------------------------------------
| Id  | Operation                    | Name                 |
-----------------------------------------------------------
|   0 | SELECT STATEMENT             |                      |
|   1 |  VIEW                        |                      |
|   2 |   UNION-ALL                  |                      |
|*  3 |    FILTER                    |                      |
|   4 |     PARTITION RANGE SINGLE   |                      |
|*  5 |      TABLE ACCESS FULL       | SYS_FBA_HIST_107038  |
|*  6 |    FILTER                    |                      |
|   7 |     MERGE JOIN OUTER         |                      |
|   8 |      SORT JOIN               |                      |
|*  9 |       TABLE ACCESS FULL      | T                    |
|* 10 |      SORT JOIN               |                      |
|* 11 |       TABLE ACCESS FULL      | SYS_FBA_TCRV_107038  |
-----------------------------------------------------------
```

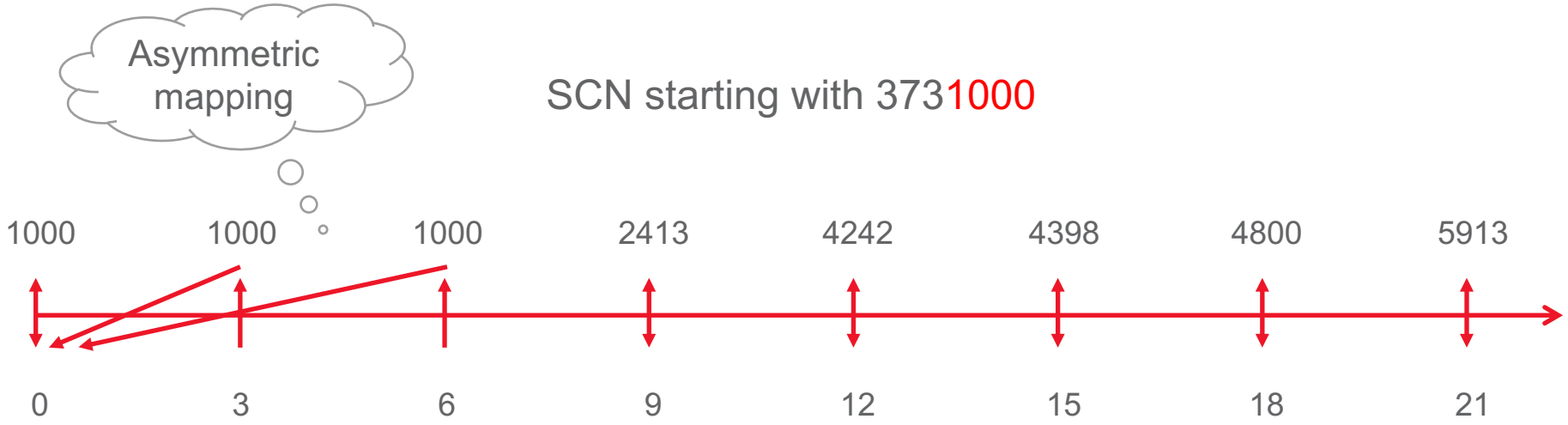
Based on history

Based on latest table

SCN – System Change Number

- Oracle uses its own time standard SCN for FBA
- Initialized during database creation and valid for a single Oracle instance
- Synchronized among database instances, e.g. when accessing data via a database link (see also MOS note 1376995.1)
- May not represent a date before 1988-01-01
- The time spent between two SCNs is varying,
  - It may be shorter when the database instance is executing a lot of transactions
  - It may be longer in more idle times or when the database instance is shut down

Mapping SCN to Timestamp – Example



Timestamp starting with 2014-03-28 08:00:00

Table SYS.SMON_SCN_TIME

- Map SCN to TIMESTAMP and vice versa through:
  - SCN_TO_TIMESTAMP
  - TIMESTAMP_TO_SCN
- Oracle states that the usual precision of the result value is 3 seconds
- A row handles up to 100 SCN using the column TIM_SCN_MAP
- The ora_smon_<sid> process maintains the content of the table
  - Creates a row every 5 minutes
  - Deletes old entries according UNDO and FBA configuration

SYS.SMON_SCN_TIME – Example

Column	Data Type	Description	Example
thread	number	Cluster key	0
time_mp	number	Number of seconds since 1970-01-01 (UTC)	1384332055
time_dp	date	Date of time_mp (UTC)	2013-11-13 08:40:55
scn_wrp	number	Number of times the scn_bas has exceeded its max. 32 bit value: floor(scn / power(2, 32))	1877
scn_bas	number	SCN as original 32-bit value: mod(scn, power(2, 32))	57569150
num_mappings	number	Number of mappings in tim_scn_map	84
tim_scn_map	raw(1200)	Array of mappings, 12 bytes for each entry, 4 bytes for time_mp, 4 bytes for scn_bas, 2 bytes for scn_wrp, last 2 bytes not yet known (example from a "Little Endian" system)	**1A3B8352**7F6F6E03**5507**C7B6 **1D3B8352**806F6E03**5507**0000 …
scn	number	System Change Number	8061711183742
orig_thread	number	Original thread, relevant for downgrade?	0

Export/Import FBA-enabled Tables?

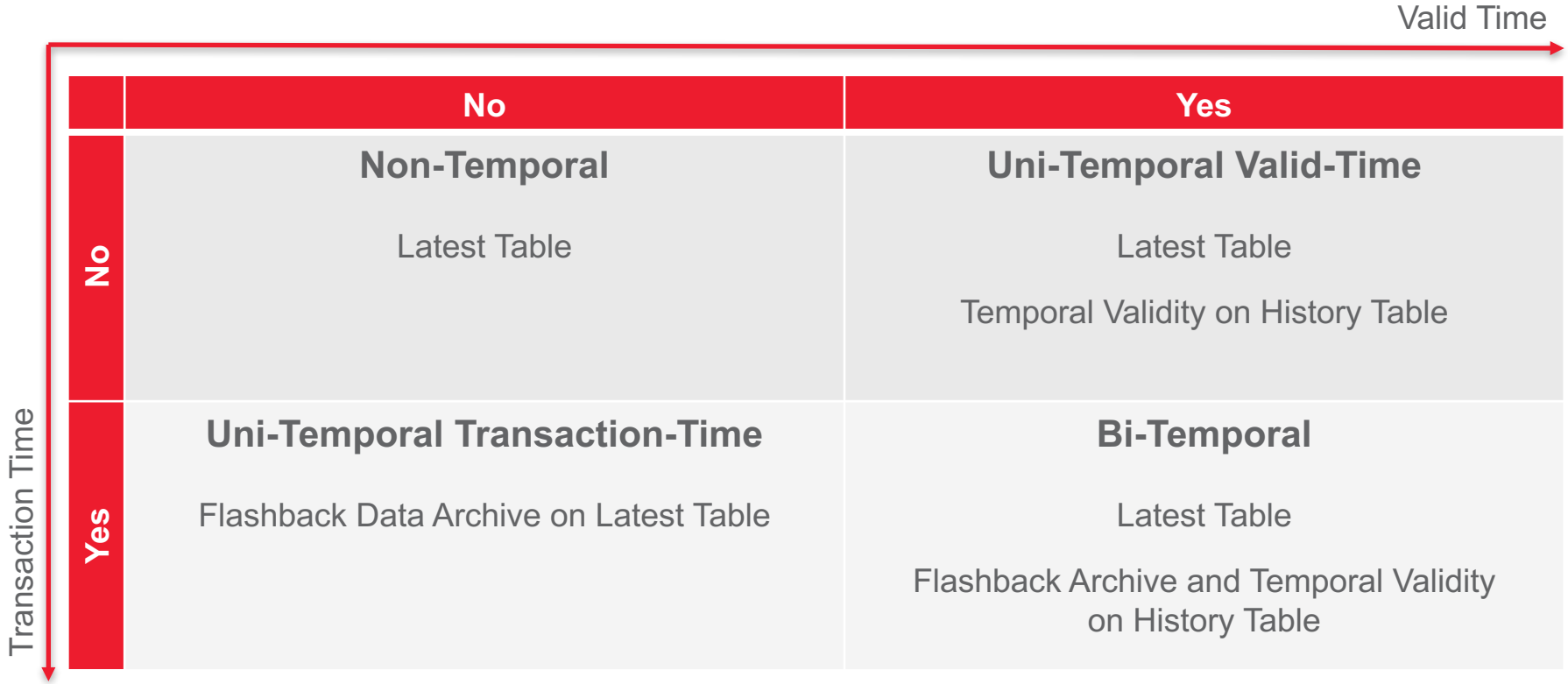
- Works, technically
- Result depending on
  - SYS.SMON_SCN_TIME in target database
  - Size of missing SCN ranges
- DBMS_FLASHBACK_ARCHIVE.EXTEND_MAPPINGS might add up to 10 mio. evenly distributed SCN before the first entry (347 days with a 3 seconds accuracy)
- Expect the history to be different
- What's the value of such a history?
- Copy data without history or use cloning technologies

Querying Views

- The *flashback_query* clause is applicable on views and tables
- Applied on views Oracle will consider UNDO only, no FBA!
- Scope of DBMS_FLASHBACK.ENABLE_AT_TIME might be too much
- Flashback queries on FBA-enabled tables must be applied on tables and not on views

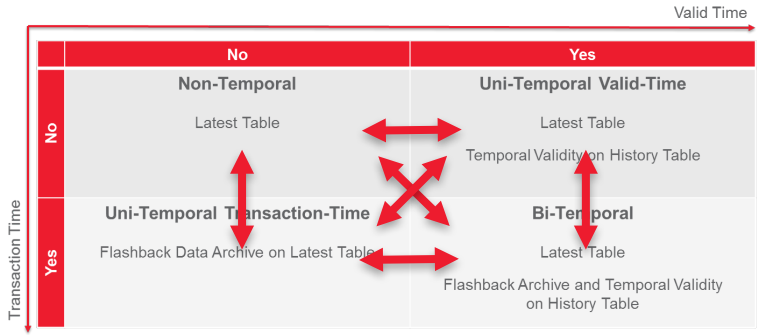
Bitemp Remodeler
Data Modeling Principles

Four Models – Use Database Features



Switching To Any Model

- 4 target models
- 12 from-to-combinations
- Keep data
  (if target model can store it)
- Compatible Table API for latest table
  (no application change necessary)
- Remaining part of Table API is target model specific
  (requires application change)



trivadis makes IT easier.

Latest Table

- For all model variants
- Data structure
  - As original non-temporal table
  - Adds a IS_DELETED$ column for uni-temporal valid-time and bi-temporal models
  - Foundation for Table API compatibility
- Meaning
  - One row per primary key for all model variants
  - This row holds data of the latest (newest, youngest) period
  - Latest row != actual valid row

History Table

- For uni-temporal valid-time and bi-temporal models
- Data structure as latest table plus
  - HIST_ID$ primary key, identity column, always generated
  - VT_START start of valid time period
  - VT_END end of valid time period
  - VT$ hidden vitual column, temporal validity period definition
  - IS_DELETED$ 1=period is marked as deleted, NULL=period is active

Temporal vs. Non-Temporal Columns

- All columns in a temporal table are temporal
- Simplifies model definition
  (no formal need to distinguish between temporal and non-temporal columns)
- Missing information in the Oracle data dictionary
  - Why is a table temporal?
  - Could be documented in table and/or column comments
- Additionally produced periods

Periods – Temporal Constraints

- For uni-temporal valid-time and bi-temporal models
- No gaps between periods
  - First vt_start IS NULL
  - Last vt_end IS NULL
  - Deleted periods are identified with the condition "IS_DELETED$ = 1"
- No overlapping periods
- No invalid periods
- No duplicate periods
- Merge identical, connected periods immediately

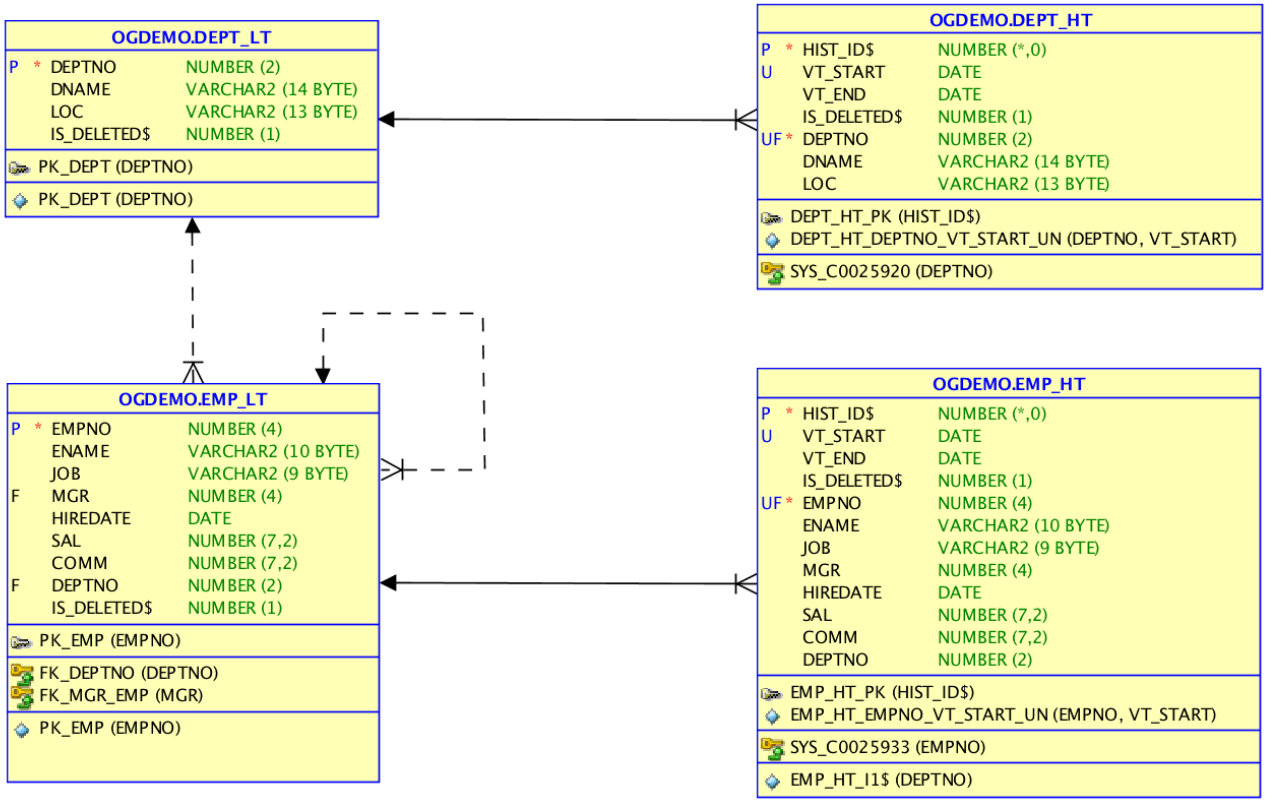
Original Primary Key

- Identification over all periods
- Foreign key constraint on history table to latest table
- Unique constraint for original primary key columns plus VT_START in history table
- Must not be changed (use surrogate key, if this is not acceptable)

Original Foreign Keys

- No foreign key constraints on history table for original foreign keys
- Non-unique indexes on original foreign keys in history table

Example Data Model



Temporal DML

Generated Table API on Table EMP

Object Type	Object Name	Description
View	EMP(_LV)	Latest view, latest rows only, updateable
	EMP_HV	History view, all but deleted rows, updateable
	EMP_FHV	Full history view, all rows, FBA version columns, readonly
Trigger	EMP_TRG	Instead-of-trigger on EMP, calls EMP_API (ins, upd, del)
	EMP_HV_TRG	Instead-of-trigger on EMP_HV, calls EMP_API (ins, upd, del)
Package	EMP_API	API package specification (ins, upd, del, init_load, delta_load, …)
	EMP_HOOK	Package specification for pre-/post ins/upd/del hooks, no body
Package Body	EMP_API	API package body (ins, upd, del, init_load, delta_load, …)
Type	EMP_OT	Object type for EMP_HT columns
	EMP_CT	Collection type, table of emp_ot
Type Body	EMP_OT	Type body with default constuctor implementation

Temporal INSERT

- Adds a period to a new or existing object
  - Deletes enclosing, existing periods
  - Adjusts validity of existing, overlapping periods
- SYSTIMESTAMP is used for vt_start for inserts on latest view
- Enforces temporal constraints
- Keeps history and latest table in sync

Temporal UPDATE

- Period changed only (vt_start, vt_end)
  - Adjust validity of overlapping periods
  - Update all columns in affected periods
  - Requires period to be enlarged to have an impact
- Application columns changed (ename, job, mgr, hiredate, sal, comm, deptno)
  - Adjust validity of overlapping periods
  - Update changed columns in all affected periods
- Enforces temporal constraints
- Keeps history and latest table in sync

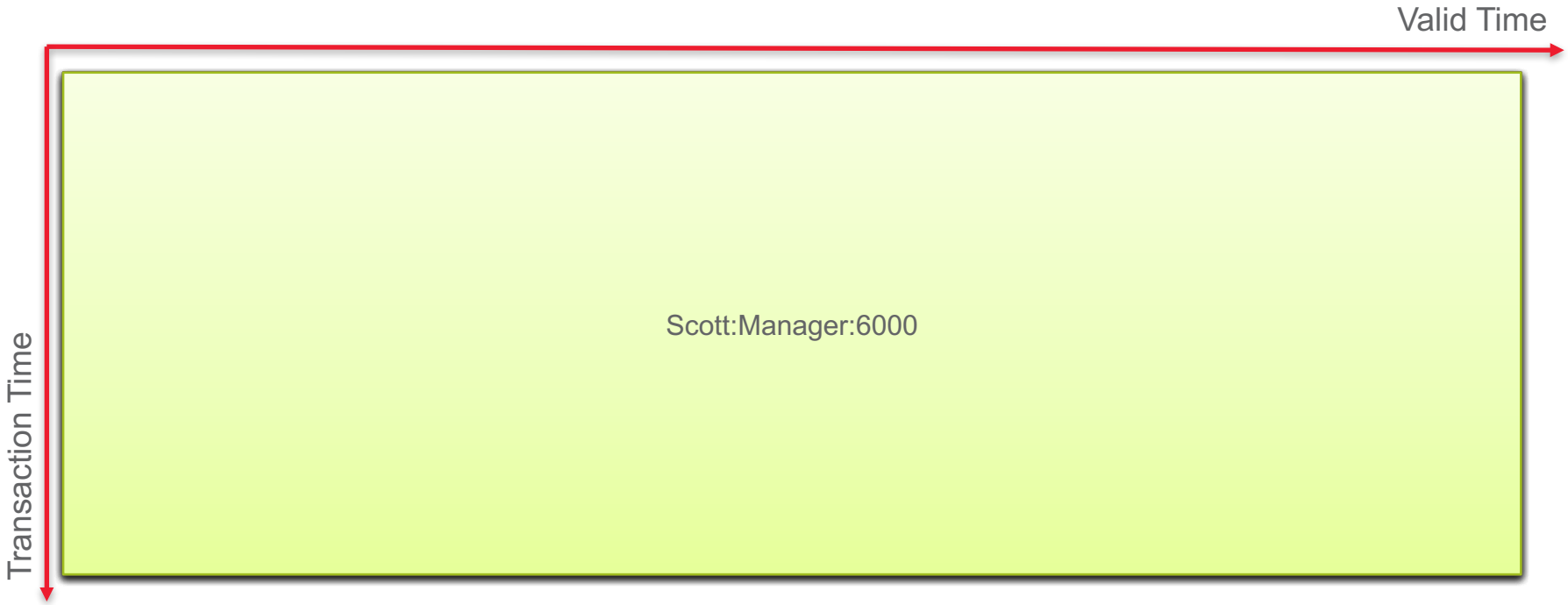
Temporal DELETE

- Delete period from an existing object
  - Adjust validity of overlapping periods
  - Set IS_DELETED$ to 1 in affected periods
- Enforces temporal constraints
- Keeps history and latest table in sync
- Deleting a non-existent period is supported via EMP_API.DEL or EMP_API.INS
- Deleted periods are not shown in updateable latest/history view
- Deleted periods are visible in read-only full history view

Events

No	Transaction Time (TT)	Valid Time (VT)	Action
1	1		Initial load from SCOTT.EMP table
2	2	1990-01-01	Change name from SCOTT to Scott
3	3	1991-04-01	Scott leaves the company
4	4	1991-10-01	Scott rejoins
5	5	1989-01-01	Change job from ANALYST TO Analyst
6	6	2017-01-01	Change job to Manager and double salary

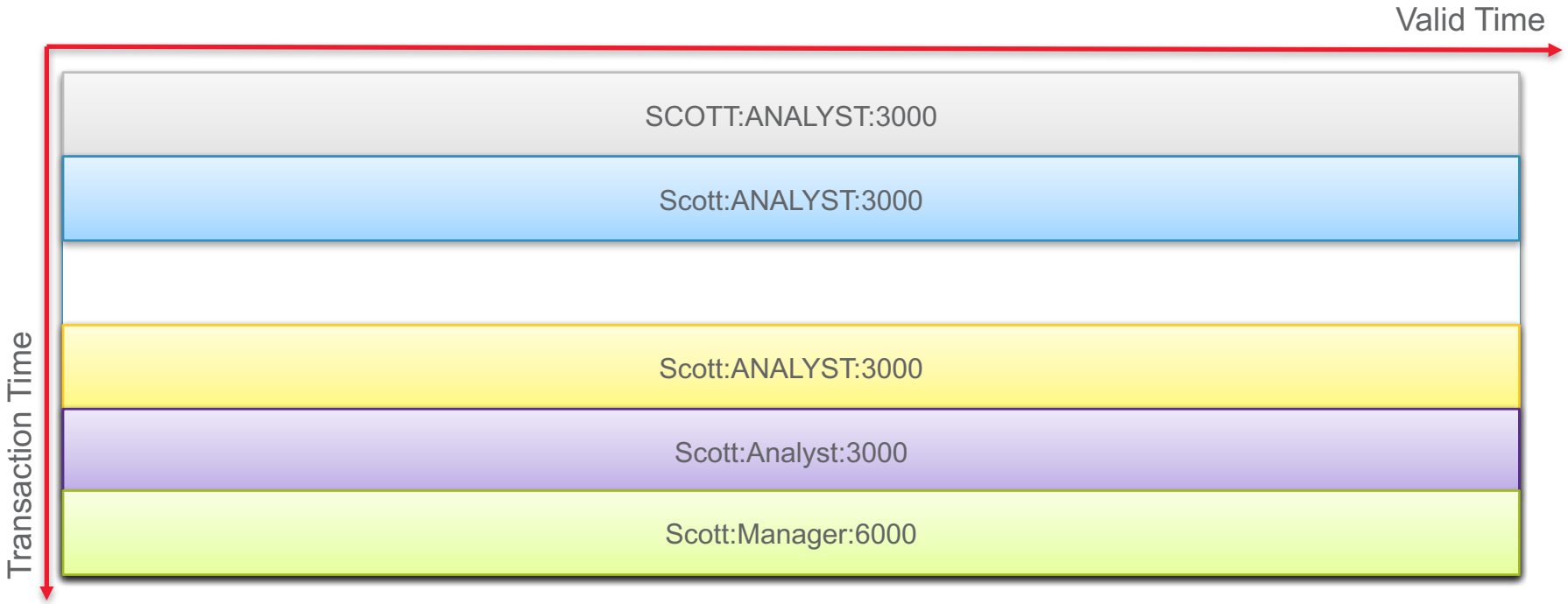
Data in Non-Temporal Model



DML for Non-Temporal Model

1. INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, deptno)
   VALUES (7788, 'SCOTT', 'ANALYST', 7566, DATE '1987-04-19', 3000, 20);
2. UPDATE emp SET ename = 'Scott' WHERE empno = 7788;
3. DELETE FROM emp WHERE empno = 7788;
4. INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, deptno)
   VALUES (7788, 'Scott', 'ANALYST', 7566, DATE '1987-04-19', 3000, 20);
5. UPDATE emp SET job = 'Analyst' WHERE empno = 7788;
6. UPDATE emp SET job = 'Manager', sal = 6000 WHERE empno = 7788;

Periods in Uni-Temporal Model (TT)



DML for Uni-Temporal Model (TT)

- Using Flashback Data Archive
- 100% identical with non-temporal model

Periods in Uni-Temporal Model (VT)

DML for Uni-Temporal Model (VT)

1
```
INSERT INTO emp_lt (empno, ename, job, mgr, hiredate, sal, deptno) VALUES (7788, 'SCOTT', 'ANALYST', 7566, DATE '1987-04-19', 3000, 20);
INSERT INTO emp_ht (empno, ename, job, mgr, hiredate, sal, deptno) SELECT empno, ename, job, mgr, hiredate, sal, deptno FROM emp_lt WHERE empno = 7788;
```

2
```
UPDATE emp_lt SET ename = 'Scott' WHERE empno = 7788;
UPDATE emp_ht SET vt_end = DATE '1990-01-01' WHERE empno = 7788;
INSERT INTO emp_ht (vt_start, empno, ename, job, mgr, hiredate, sal, deptno)
  SELECT vt_end, empno, 'Scott', job, mgr, hiredate, sal, deptno FROM emp_ht WHERE vt_end = DATE '1990-01-01';
```

3
```
UPDATE emp_lt SET is_deleted$ = 1 WHERE empno = 7788;
UPDATE emp_ht SET vt_end = DATE '1991-04-01' WHERE empno = 7788 AND vt_end IS NULL;
INSERT INTO emp_ht (vt_start, empno, ename, job, mgr, hiredate, sal, deptno, is_deleted$)
  SELECT vt_end, empno, ename, job, mgr, hiredate, sal, deptno, 1 FROM emp_ht WHERE vt_end = DATE '1991-04-01';
```

4
```
UPDATE emp_lt SET is_deleted$ = NULL WHERE empno = 7788;
UPDATE emp_ht SET vt_end = DATE '1991-10-01' WHERE empno = 7788 AND vt_end IS NULL;
INSERT INTO emp_ht (vt_start, empno, ename, job, mgr, hiredate, sal, deptno)
  VALUES (DATE '1991-10-01', 7788, 'Scott', 'ANALYST', 7566, DATE '1987-04-19', 3000, 20);
```

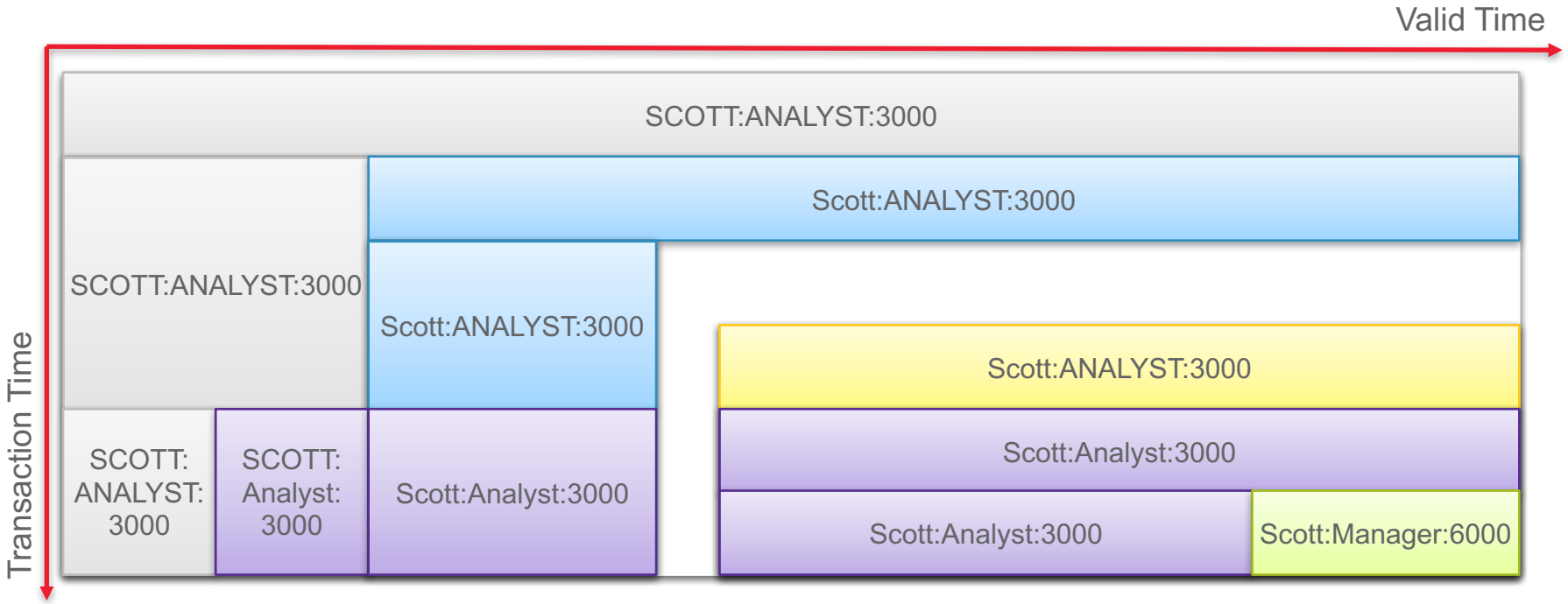
5
```
UPDATE emp_lt SET job = 'Analyst' WHERE empno = 7788;
UPDATE emp_ht SET vt_end = DATE '1989-01-01' WHERE empno = 7788 AND vt_start IS NULL;
INSERT INTO emp_ht (vt_start, vt_end, empno, ename, job, mgr, hiredate, sal, deptno)
  SELECT vt_end, DATE '1990-01-01', empno, ename, 'Analyst', mgr, hiredate, sal, deptno FROM emp_ht WHERE empno = 7788 AND vt_end = DATE '1989-01-01';
UPDATE emp_ht SET job = 'Analyst' WHERE empno = 7788 AND vt_start >= DATE '1990-01-01' AND is_deleted$ IS NULL;
```

6
```
UPDATE emp_lt SET job = 'Manager', sal = 6000 WHERE empno = 7788;
UPDATE emp_ht SET vt_end = DATE '2017-01-01' WHERE empno = 7788 AND vt_end IS NULL;
INSERT INTO emp_ht (vt_start, empno, ename, job, mgr, hiredate, sal, deptno)
  SELECT vt_end, empno, ename, 'Manager', mgr, hiredate, 6000, deptno FROM emp_ht WHERE empno = 7788  AND vt_end = DATE '2017-01-01';
```

Temporal DML for Uni-Temporal Model (VT)

1. INSERT INTO emp_hv (empno, ename, job, mgr, hiredate, sal, deptno)
   VALUES (7788, 'SCOTT', 'ANALYST', 7566, DATE '1987-04-19', 3000, 20);
2. UPDATE emp_hv SET ename = 'Scott', vt_start = DATE '1990-01-01',
   vt_end = DATE '1991-04-01' WHERE empno = 7788 AND vt_end IS NULL;
3. DELETE FROM emp_hv WHERE empno = 7788 AND vt_start = DATE '1991-04-01';
4. INSERT INTO emp_hv (vt_start, empno, ename, job, mgr, hiredate, sal, deptno)
   VALUES (DATE '1991-10-01', 7788, 'Scott', 'ANALYST', 7566, DATE '1987-04-19', 3000, 20);
5. UPDATE emp_hv SET job = 'Analyst', vt_start = DATE '1989-01-01'
   WHERE empno = 7788 AND vt_end IS NULL;
6. UPDATE emp_hv SET job = 'Manager', sal = 6000, vt_start = DATE '2017-01-01'
   WHERE empno = 7788 AND vt_end IS NULL;

Periods in Bi-Temporal Model

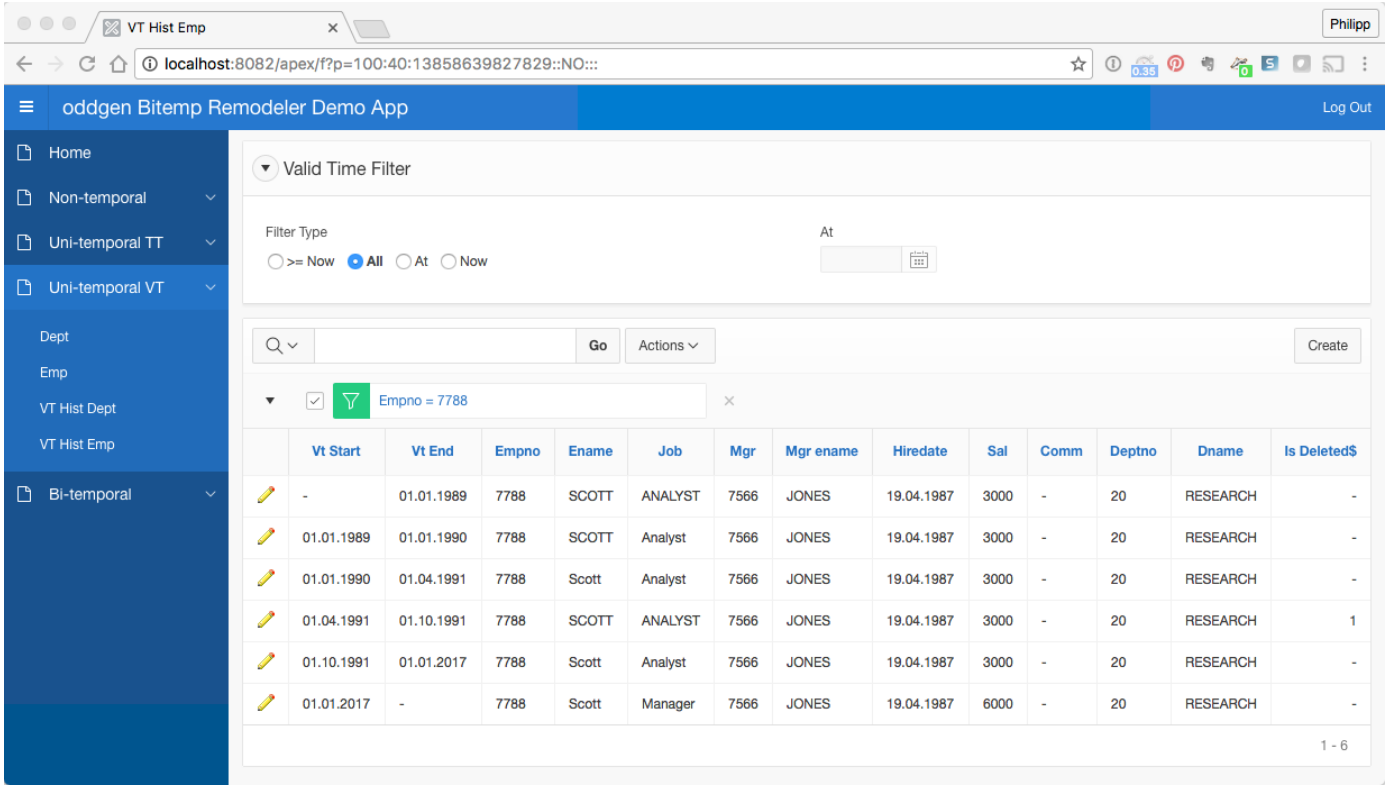


Temporal DML for Bi-Temporal Model

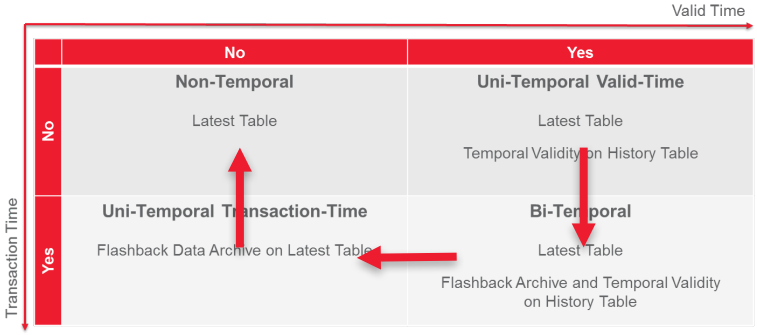
- Using Flashback Data Archive and Temporal Validity
- 100% identical with uni-temporal valid-time model

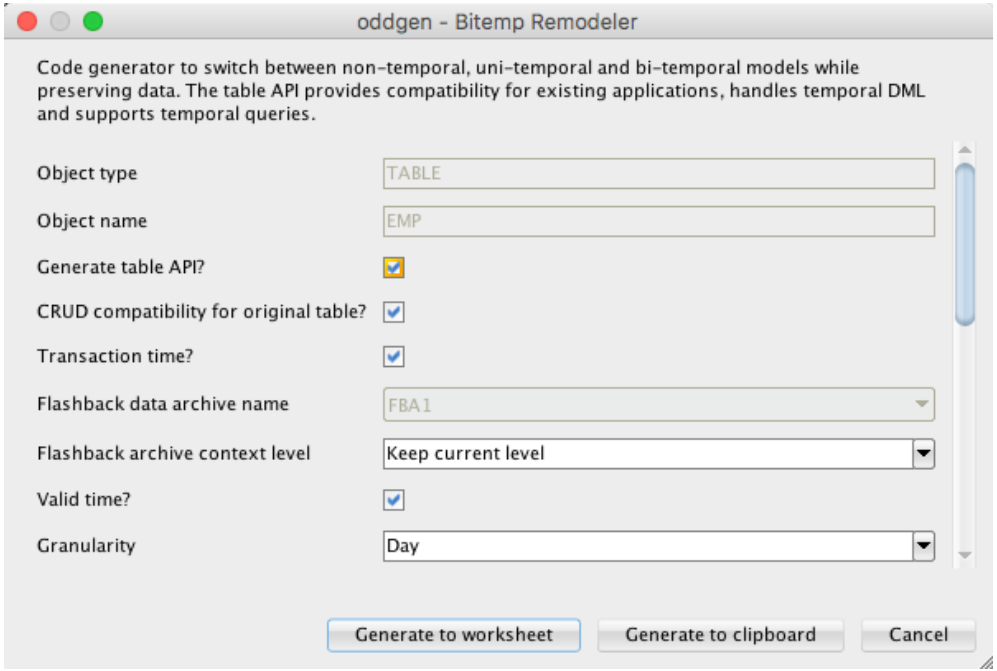
Switching Models

Dept/Emp APEX 5 Application



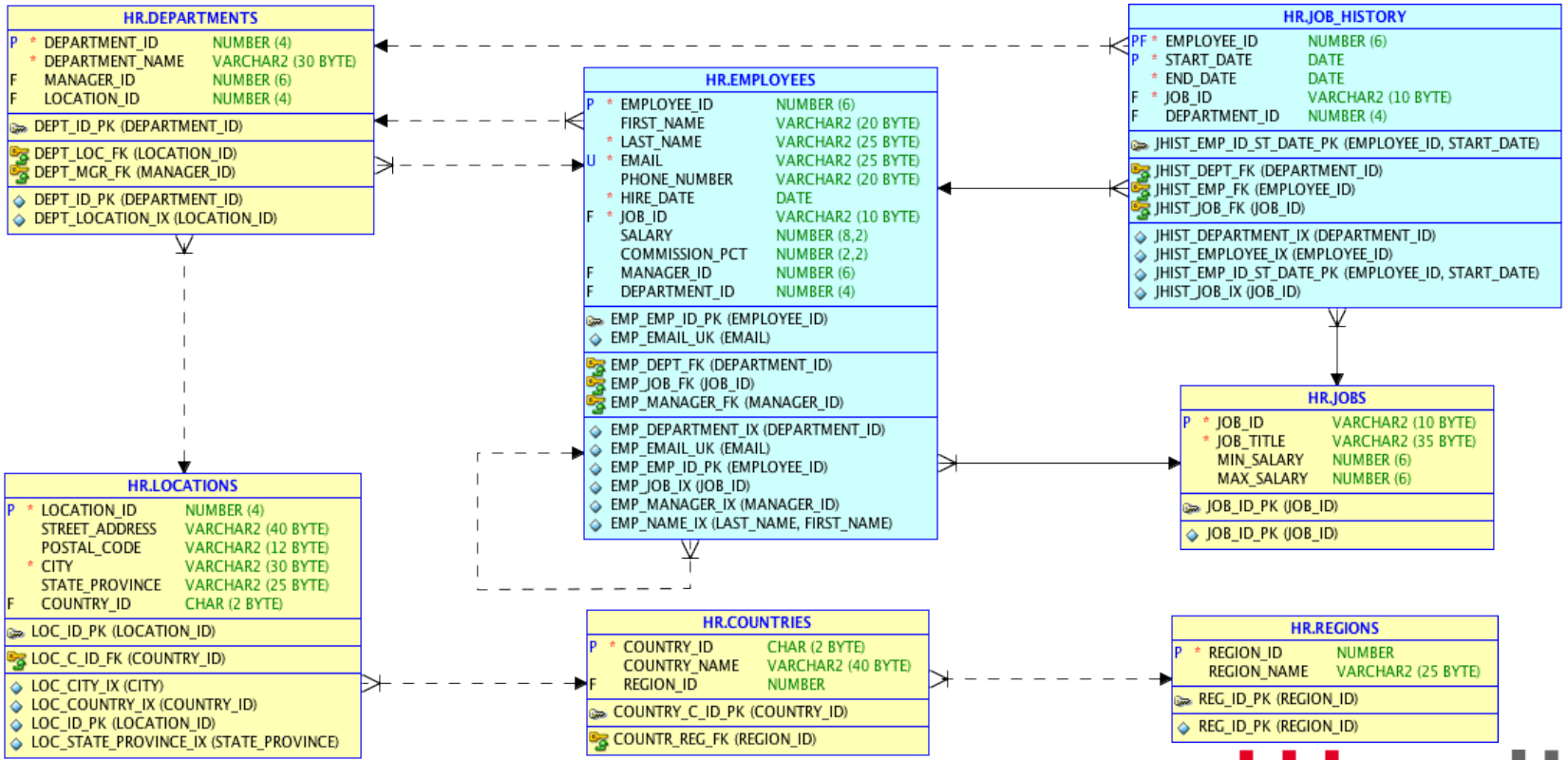
Switching EMP Models



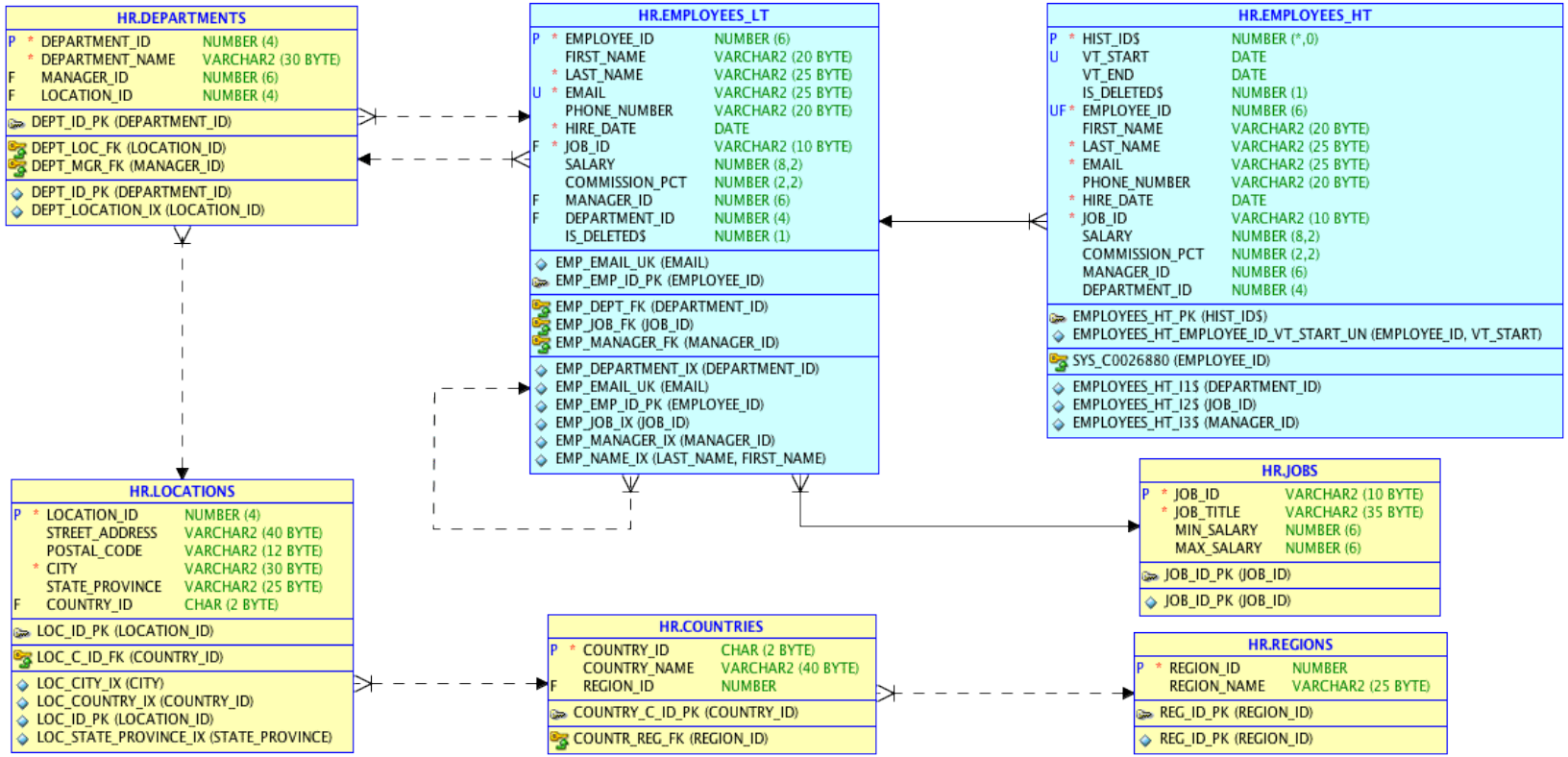


trivadis makes IT easier.

HR Model – Original Example (Source)



HR Model – Uni-Temporal Valid-Time (Target)



From Source to Target Model

- Generate target model using Bitemp Remodeler
- Apply generated code
- Create staging and error log table
- Populate staging table
- Delta Load
- Cleanup

Bulk Processing

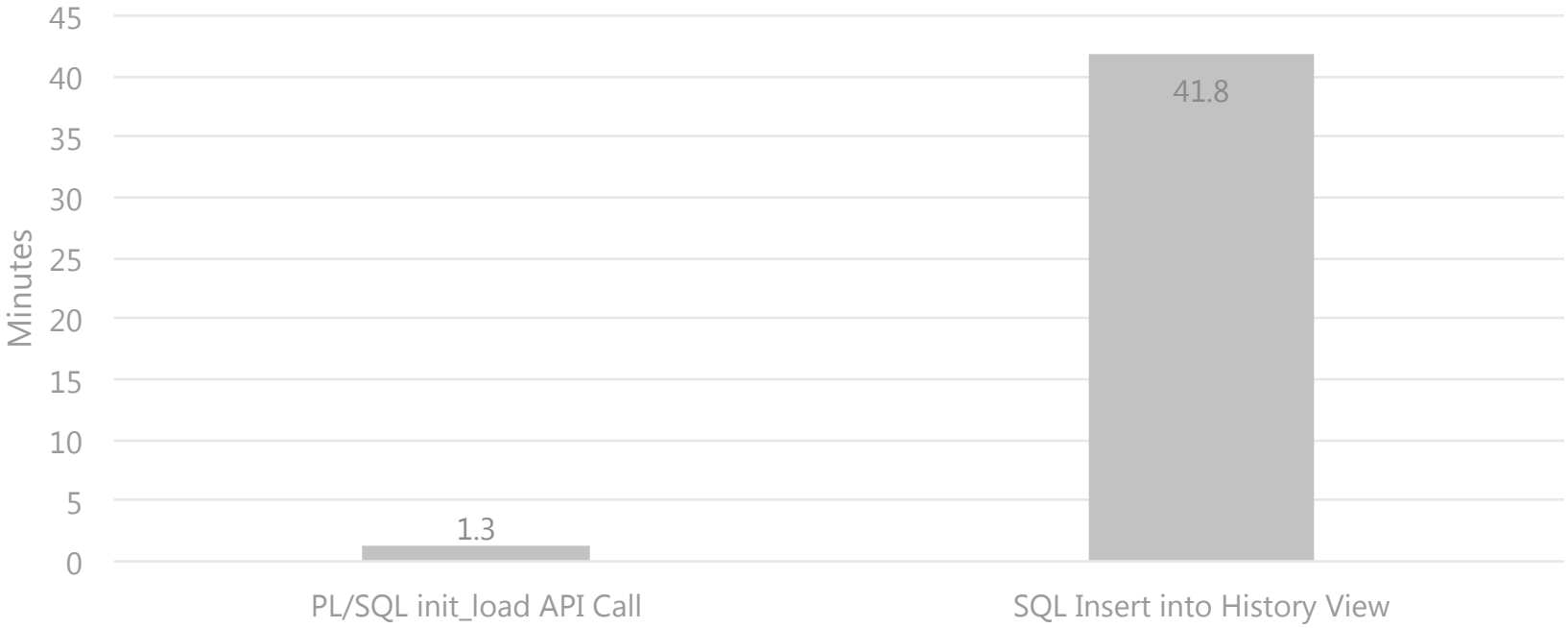
Initial Load API

- Use Cases
  - BI/DWH environment (Core, Data Marts)
  - Data migration
- Scope / Features
  - Uni-temporal valid-time and bi-temporal models
  - Temporal INSERT from staging table
  - Error log and REJECT_LIMIT according DML *error_logging_clause*
- Limitations
  - Table API hooks are not processed
  - No parallel processing (yet)

About Data for Initial Load

- Uni-temporal valid-time target table with columns deptno (PK, enabled) and dname
- 1'000'000 periods in staging table
- 551'914 unique deptno
  - 262'939 deptno with 1 period
  - 287'059 deptno with 2 .. 5 periods
  - 1913 deptno with 6 .. 9 periods
  - 3 deptno with 10 periods
- 551'914 rows loaded into latest table
- 1'615'123 periods loaded into history table

Initial Load



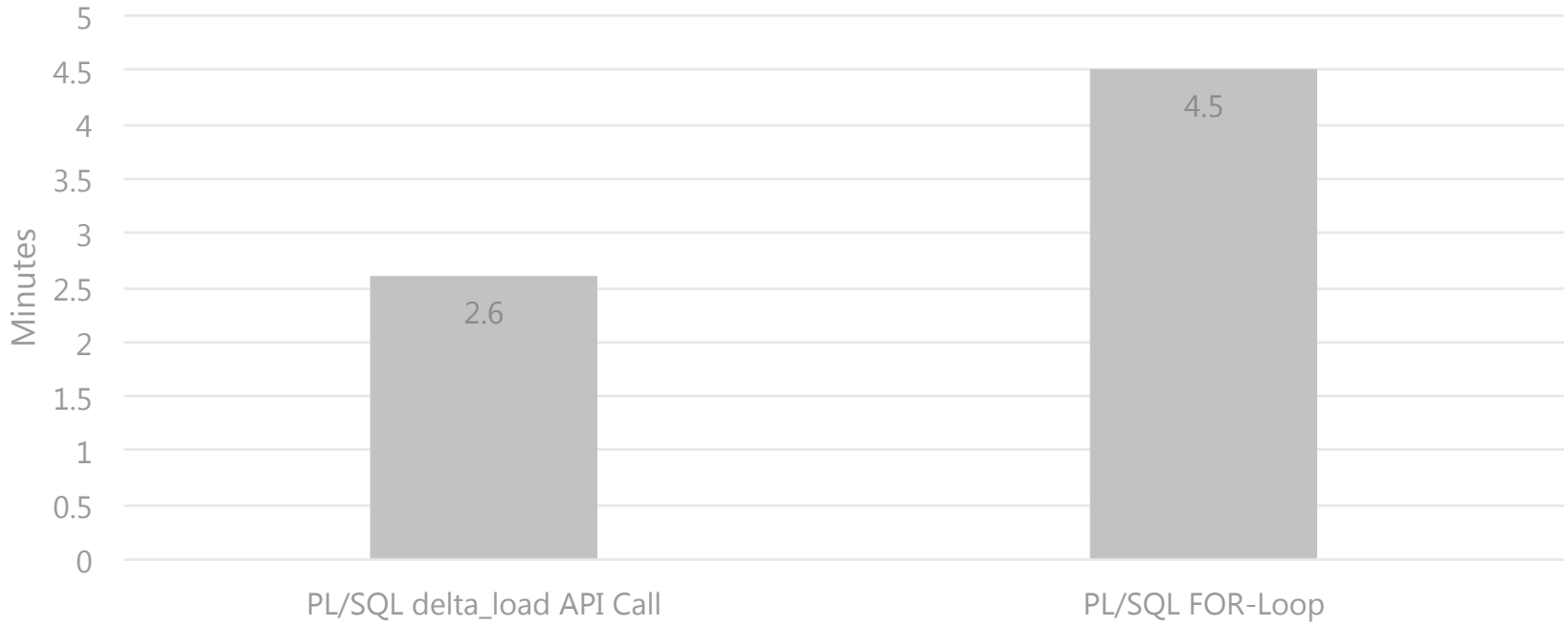
Delta Load API

- Use Cases
  - BI/DWH environment (Core, Data Marts)
- Scope / Fatures
  - Uni-temporal valid-time and bi-temporal models
  - Temporal INSERT, UPDATE and DELETE from staging table
    (PK columns, vt_start, vt_end, is_deleted$ required for delete operation)
  - Process full or reduced data sets
  - Error log and REJECT_LIMIT according DML *error_logging_clause*
- Limitations
  - Performance depends on staging table size and change ratio (partial bulk)

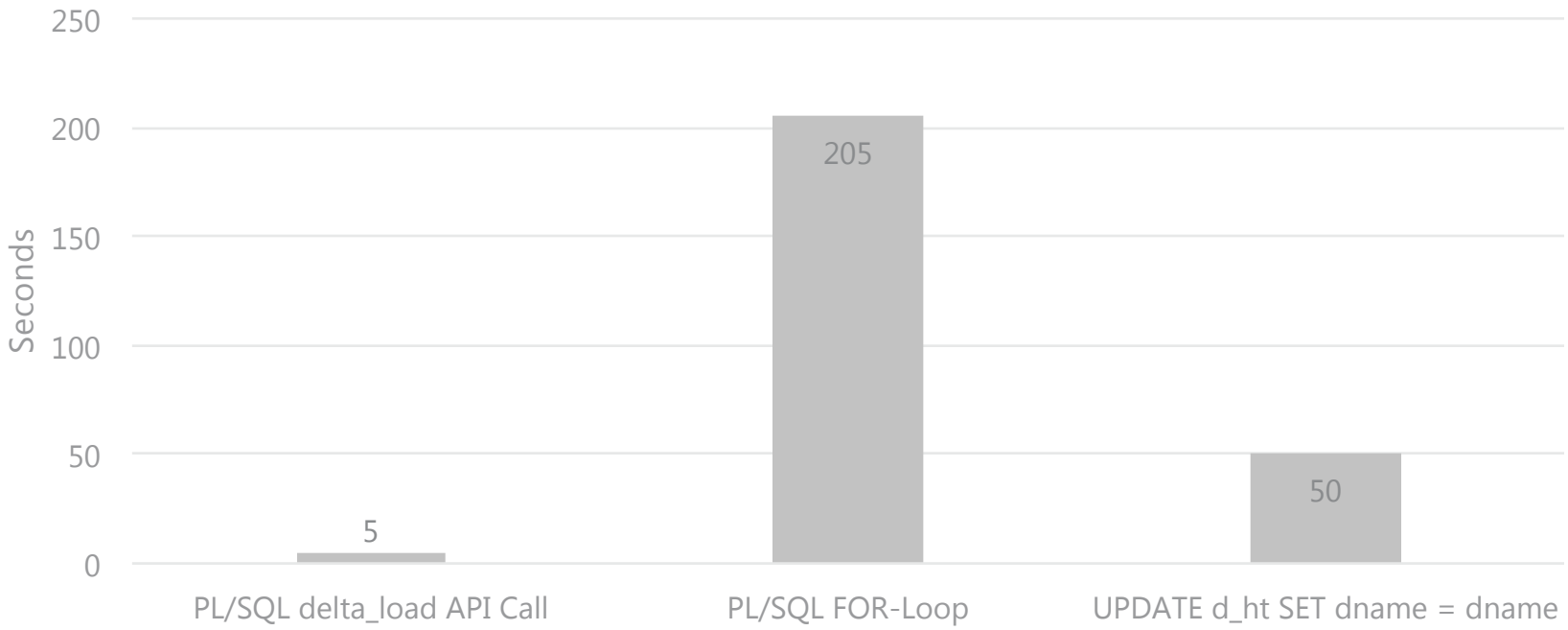
About Data for Delta Load

- Uni-temporal valid-time target table with columns deptno (PK, enabled) and dname
- 1'000'000 periods in staging table (full set, e.g. new snapshot delivery)
- Same distribution as for initial load
- 59'649 periods changed (~6%)
- 32'905 rows to be updated in latest table
- 59'649 periods to be updated in history table
- No deletes, no inserts

Delta Load – First Run (6% Changes)



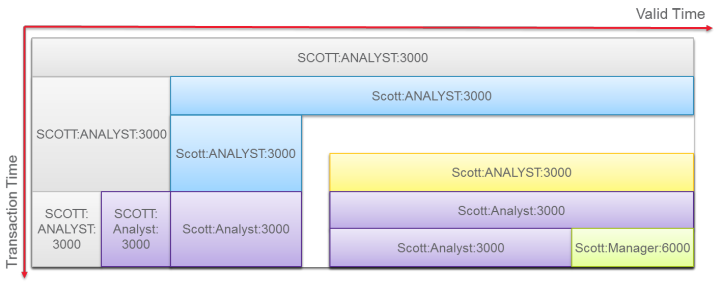
Delta Load – Second Run (0% Changes)



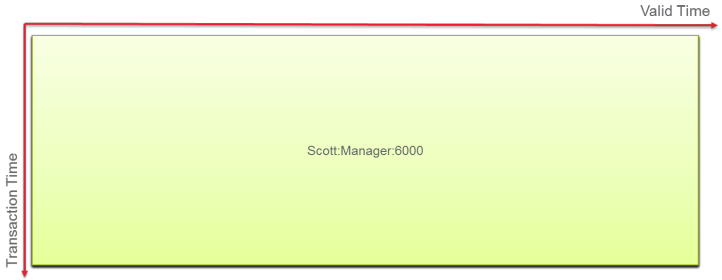
Core Messages

Temporal Models Are not Simple

- Do not create temporal tables "just in case"
- Use temporal tables only if you know which columns require a temporal view
- Start with a non-temporal table if you are unsure



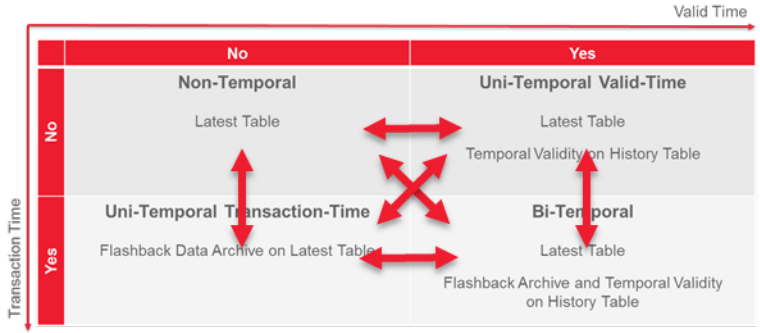
vs.



trivadis makes IT easier.

Use oddgen Bitemp Remodeler

- Reduces the problem-solution gap
- Makes model switching easier



trivadis makes IT easier.

Questions and Answers

Philipp Salvisberg
Senior Principal Consultant

Tel. +41 58 459 52 31
philipp.salvisberg@trivadis.com

trivadis
makes IT easier.

Further Information

- https://www.oddgen.org/
- https://github.com/oddgen/

trivadis
makes IT easier.

oddgen Bitemp Remodeler Prerequisites

- Products
  - Oracle 12.1.0.2 Standard Edition or higher
  - Oracle SQL Developer 4.0.2 or higher
  - oddgen for SQL Developer 0.2.5 or higher
- Oracle Database based applications
  - Table with defined primary key (does not need to be enabled)
  - Table name length <= database maximal length – 5 characters

trivadis makes IT easier.