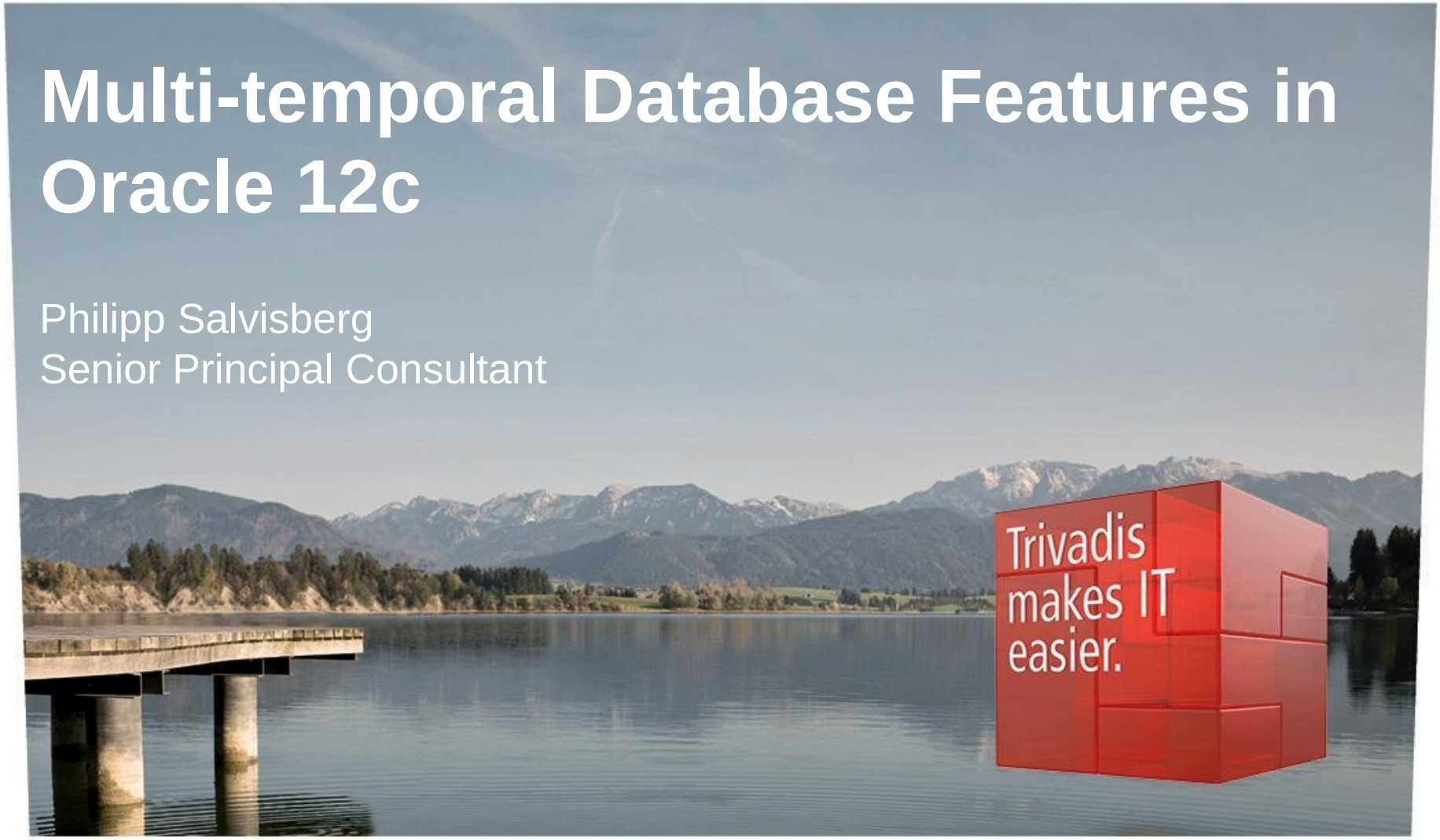
Multi-temporal Database Features in Oracle 12c

Philipp Salvisberg
Senior Principal Consultant



BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MUNICH STUTTGART VIENNA

trivadis
makes IT easier.

About Me

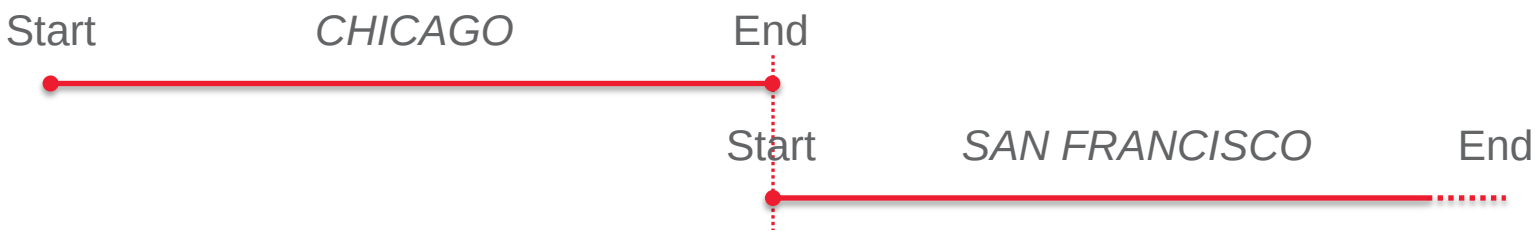
- With Trivadis since April 2000
  - Senior Principal Consultant, Partner
  - Member of the Board of Directors
  - philipp.salvisberg@trivadis.com
  - www.salvis.com/blog
- Member of the trivadis performanceteam

- Main focus on database centric development with Oracle DB
  - Application Development
  - Business Intelligence
  - Application Performance Management
- Over 20 years experience in using Oracle products

AGENDA

1. Introduction
2. Temporal Validity
3. Temporal Flashback Query
4. Tri-Temporal Model in Action
5. Core Messages

Period, Semantics and Granularity



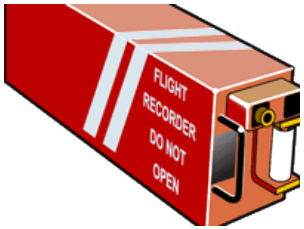
1981-02-20	2014-01-01	NULL (∞)
1981-02-20 00:00:00	2014-01-01 00:00:00	NULL (∞)
1981-02-20 00:00:00.000000	2014-01-01 00:00:00.000000	NULL (∞)

Use of Temporal Periods

Model	Data Category	Appropriate?
Entity Relationship	Master Data	Yes
	Reference Data	Yes
	Position Data	-
	Transaction Data	-
Dimensional	Dimension (SCD2)	Yes
	Fact	-

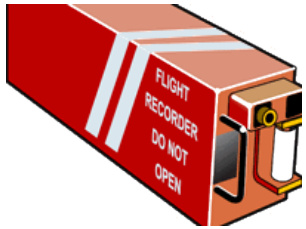
Temporal Database – When Do You Want To Be?

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End	System timestamp when the data has been outdated.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	Job description change from Analyst to Manager. Was entered into the system on 2013-04-15 15:42:42. Preceding job description was 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 FBDA). Transaction awareness required for a consistent picture of the past!

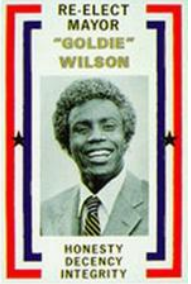
Consistent Picture of the Past with FBDA?



SCN	Session A	Session B
1	INSERT INTO emp (empno, ename, job, sal, deptno) VALUES (4242, 'CARTER', 'CLERK', '2400', 20);	
2	SELECT COUNT(*) FROMemp; -- 15 rows	
3		SELECT COUNT(*) FROMemp; -- 14 rows
4	COMMIT;-- delayed FBDA 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	Job description change from Analyst to Manager. Becomes Valid on 2014-01-01. Preceding job description 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.

Decision Time (DT)

Aka	DT
Start	Date or Timestamp the decision has been made.
End	Date or Timestamp the decision has been revised.
Example	Job description change from Analyst to Manager. Decision has been made on 2013-03-24. Preceding job description has become void on the same date.
Charac-teristics	Data is available about previous and current states. Backdated changes are possible. Future changes are not possible. Requires a decision-time-aware data model.

Historization Types

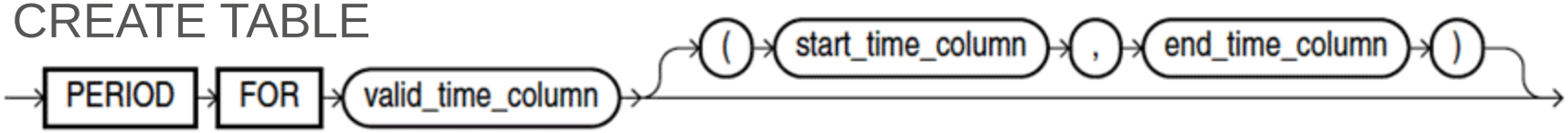
- Non-temporal Model
  - E.g. EMP and DEPT in schema SCOTT
- Uni-temporal Model
  - Typically TT or VT
- Bi-temporal Model
  - Typically TT and VT
- Multi-temporal Model
  - Typically TT and VT and DT (tri-temporal)
  - Further combination possible using application specific temporal periods with a total of at least three temporal periods

AGENDA

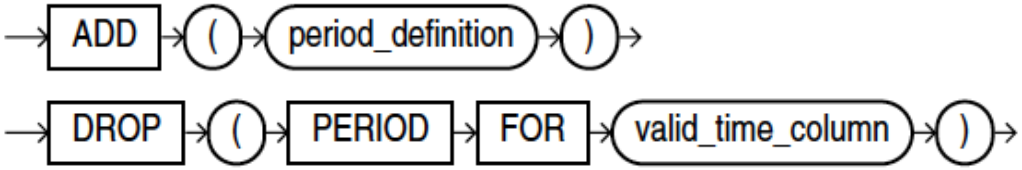
1. Introduction
2. Temporal Validity
3. Temporal Flashback Query
4. Tri-Temporal Model in Action
5. Core Messages

DDL Enhancements

- CREATE TABLE



- ALTER TABLE



- Columns
  - *valid_time_column* is the name of temporal period
    - virtual and hidden
    - default prefix for *start_time_column* and *end_time_column*
  - *start_time_column* and *end_time_column* (NULL = ∞)
    - hidden by default

Explicit Period

- Visible period start and end columns

```
SQL> ALTER TABLE dept ADD (
  2     vt_start DATE,
  3     vt_end   DATE,
  4     PERIOD FOR vt (vt_start, vt_end)
  5  );
```

```
SQL> SELECT * FROM dept;

    DEPTNO DNAME          LOC           VT_START   VT_END
---------- -------------- ------------- ---------- ----------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

Period Definition

- Where's the period defined in the data dictionary?

```
SQL> SELECT * FROM sys.sys_fba_period WHERE obj# IN
  2     (SELECT object_id
  3        FROM dba_objects
  4       WHERE owner = 'SCOTT' AND object_name = 'DEPT');

 OBJ#       PERIODNAME       FLAGS PERIODSTART PERIODEND       SPARE
---------- ---------- ---------- ----------- --------- ----------
     91661 VT                    0 VT_START    VT_END
```

- Flags
  - 1 = implicit period (hidden period start and end columns)
  - 0 = explicit period (visible period start and end columns)

Constraints

- Defined Constraints

```
SQL> SELECT constraint_name, search_condition_vc
  2    FROM user_constraints
  3   WHERE table_name = 'DEPT';

CONSTRAINT_NAME                SEARCH_CONDITION_VC
------------------------------ --------------------------------
PK_DEPT
VT7AD5E3                       (VT_START < VT_END) and (VT > 0)
```

- Basic temporal integrity constraint
- No object identifier (non-temporal identifier)
  - Required to check for overlapping periods
  - Required to allow/prohibit gaps between periods
  - Required to check for orphaned children and parents (temporal integrity)

DML

- No DML extension nor stored objects for DML support, e.g.
  - Insert, update, delete for a given period
  - Update a subset of columns across multiple periods
  - Merge of connected and identical periods after a DML operation
- Do it yourself

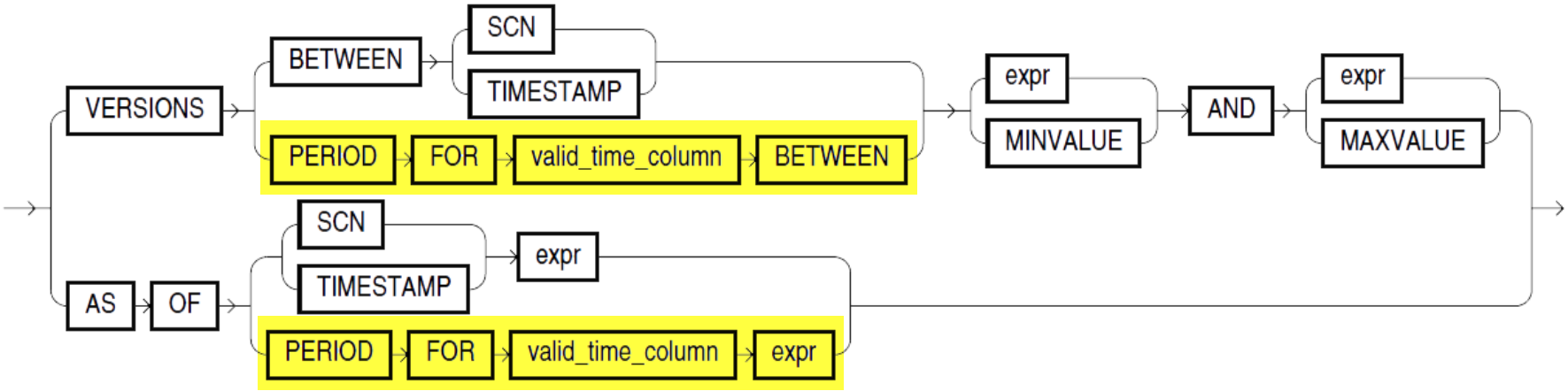
```
SQL> ALTER TABLE emp DISABLE CONSTRAINT fk_deptno;
SQL> ALTER TABLE dept DISABLE PRIMARY KEY;
SQL> UPDATE dept SET vt_end = DATE '2014-01-01' WHERE deptno = 30;
SQL> INSERT INTO dept (deptno, dname, loc, vt_start)
 2        VALUES (30, 'SALES', 'SAN FRANCISCO', DATE '2014-01-01');
SQL> SELECT * FROM dept WHERE deptno = 30 ORDER BY vt_start NULLS FIRST;

 DEPTNO    DNAME          LOC            VT_START   VT_END
---------- -------------- ------------- ---------- ----------
        30 SALES          CHICAGO                  2014-01-01
        30 SALES          SAN FRANCISCO 2014-01-01
```

AGENDA

1. Introduction
2. Temporal Validity
3. Temporal Flashback Query
4. Tri-Temporal Model in Action
5. Core Messages

Select Enhancements

- flashback_query_clause



- Part of the table_reference_clause as in previous releases

Native Query

- As Of Period Query

```
SQL> SELECT *
  2    FROM dept AS OF PERIOD FOR vt DATE '2015-01-01'
  3   ORDER BY deptno;

    DEPTNO DNAME          LOC           VT_START   VT_END
---------- -------------- ------------- ---------- ----------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          SAN FRANCISCO 2014-01-01
        40 OPERATIONS     BOSTON
```

Native Query Plan

- Explain Plan of "As Of Period Query"

```
-----------------------------------
| Id  | Operation           | Name |
-----------------------------------
|   0 | SELECT STATEMENT    |      |
|   1 |  SORT ORDER BY      |      |
|*  2 |   TABLE ACCESS FULL| DEPT |
-----------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter((("T"."VT_START" IS NULL OR "T"."VT_START" <=
              TO_DATE(' 2015-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
              AND ("T"."VT_END" IS NULL OR "T"."VT_END" >
              TO_DATE(' 2015-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))))
```

- The filter is the value of the flashback_query_clause

DBMS_FLASHBACK_ARCHIVE Enhancements

- ENABLE_AT_VALID_TIME
  (level IN VARCHAR2, query_time IN TIMESTAMP SYSTIMESTAMP)
  - Level 'ALL' – no filter
  - Level 'CURRENT' – filter records to display currently valid data
  - Level 'ASOF' – filter records to display valid data as of <query_time>
- DISABLE_ASOF_VALID_TIME
  - Clears an ASOF filter
  - Same as "ENABLE_AT_VALID_TIME('ALL');"

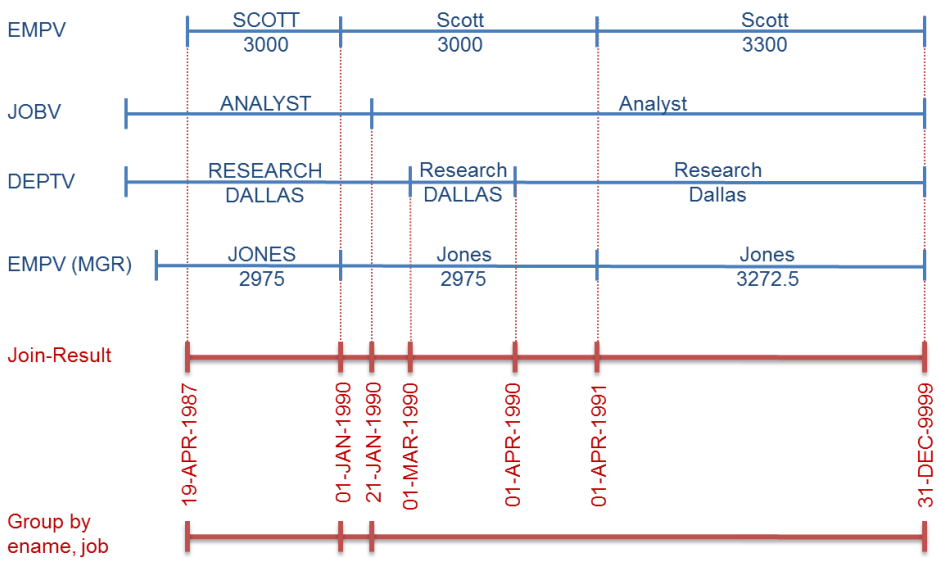
Context Query

- DBMS_FLASHBACK_ARCHIVE As Of Query

```
SQL> BEGIN
  2     dbms_flashback_archive.enable_at_valid_time(
  3        level      => 'ASOF',
  4        query_time => DATE '2015-01-01'
  5     );
  6  END;
  7  /

SQL> SELECT * FROM dept ORDER BY deptno;

    DEPTNO DNAME          LOC           VT_START    VT_END
---------- -------------- ------------- ---------- ----------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          SAN FRANCISCO 2014-01-01
        40 OPERATIONS     BOSTON
```

Limitations

- Temporal validity is not supported with a multitenant container database
  - DDL and DML works
  - Temporal flashback query filters are applied in a non-CDB instance only
- flashback_query_clause
  - Temporal flashback query filters are not applied on views
- No support for
  - Temporal join
  - Temporal aggregation (group by)



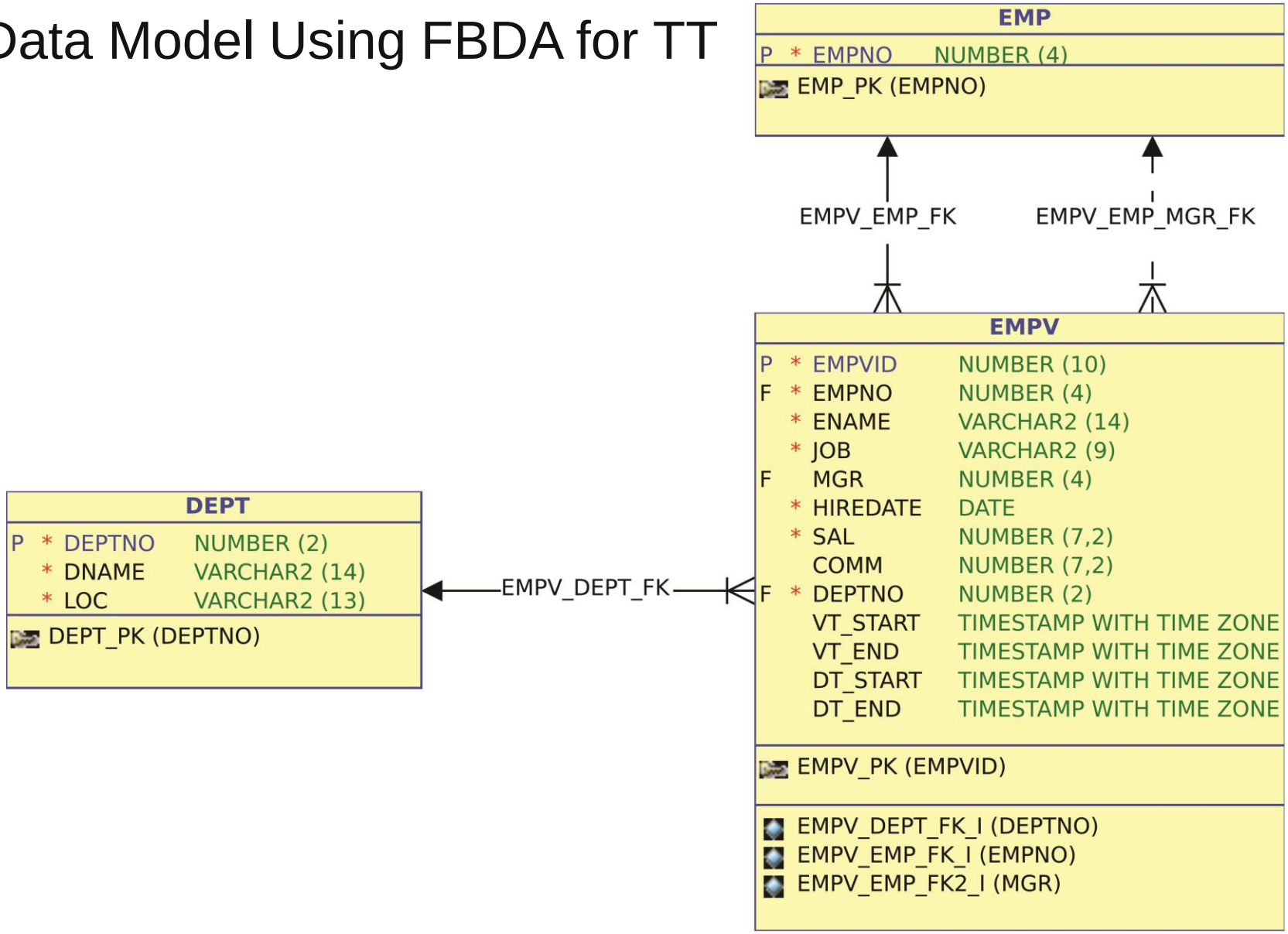
Bugs

- flashback_query_clause
  - More than one flashback_query_clause is not supported (Bug 16793968)
  - Traditional "AS OF" flashback query combined with one valid_time_column works, but be aware that FBDA tables are never accessed and such a query will cause an ORA-1555 if data is not available in UNDO
- PL/SQL package dbms_flashback_archive
  - Missing period parameter in enable_at_valid_time (Bug 16793933)
  - Wrong result when calling enable_at_valid_time multiple times (16793845)

AGENDA

1. Introduction
2. Temporal Validity
3. Temporal Flashback Query
4. Tri-Temporal Model in Action
5. Core Messages

Data Model Using FBDA for TT



load.sql
alltt.sql
alldt.sql
allvt.sql
combi.sql

Events

No	Transaction Time (TT)	Valid Time (VT)	Decision Time (DT)	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	2014-01-01	2013-03-24	Change job to Manager and double salary

All Versions after Event #6

```
SQL> SELECT dense_rank() OVER(ORDER BY versions_startscn) event_no, empno, ename, job,
  2         sal, versions_startscn tt_start, versions_endscn tt_end,
  3         to_char(vt_start,'YYYY-MM-DD') vt_start, to_char(vt_end,'YYYY-MM-DD') vt_end,
  4         to_CHAR(dt_start,'YYYY-MM-DD') dt_start, to_char(dt_end,'YYYY-MM-DD') dt_end
  5    FROM empv VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
  6   WHERE empno = 7788 AND versions_operation IN ('I','U')
  7   ORDER BY tt_start, vt_start NULLS FIRST, dt_start NULLS FIRST;

 # EMPNO ENAME JOB       SAL TT_START   TT_END VT_START   VT_END     DT_START   DT_END
-- ----- ----- ------- ----- -------- -------- ---------- ---------- ---------- ----------
 1  7788 SCOTT ANALYST  3000  2366310  2366356
 2  7788 SCOTT ANALYST  3000  2366356  2366559            1990-01-01
 2  7788 Scott ANALYST  3000  2366356  2366408 1990-01-01
 3  7788 Scott ANALYST  3000  2366408  2366559 1990-01-01 1991-04-01
 4  7788 Scott ANALYST  3000  2366424  2366559 1991-10-01
 5  7788 SCOTT ANALYST  3000  2366559                     1989-01-01
 5  7788 SCOTT Analyst  3000  2366559          1989-01-01 1990-01-01
 5  7788 Scott Analyst  3000  2366559          1990-01-01 1991-04-01
 5  7788 Scott Analyst  3000  2366559  2366670 1991-10-01
 6  7788 Scott Analyst  3000  2366670          1991-10-01                     2013-03-24
 6  7788 Scott Analyst  3000  2366670          1991-10-01 2014-01-01 2013-03-24
 6  7788 Scott Manager  6000  2366670          2014-01-01            2013-03-24

12 rows selected.
```

Current TT, VT as of 2014-01-01, DT as of '2013-03-15'

```
SQL> SELECT empno, ename, job, sal,
  2         to_char(vt_start,'YYYY-MM-DD') AS vt_start,
  3         to_char(vt_end,'YYYY-MM-DD') AS vt_end,
  4         to_CHAR(dt_start,'YYYY-MM-DD') AS dt_start,
  5         to_char(dt_end,'YYYY-MM-DD') AS dt_end
  6    FROM empv AS OF period FOR dt DATE '2013-03-15'
  7   WHERE empno = 7788 AND
  8         (vt_start <= DATE '2014-01-01' OR vt_start IS NULL) AND
  9         (vt_end > DATE '2014-01-01' OR vt_end IS NULL)
 10   ORDER BY vt_start NULLS FIRST, dt_start NULLS FIRST;

EMPNO ENAME JOB       SAL VT_START   VT_END     DT_START   DT_END
----- ----- ------- ----- ---------- ---------- ---------- ----------
 7788 Scott Analyst  3000 1991-10-01                       2013-03-24

1 row selected.
```

Current TT, VT as of 2014-01-01, DT as of '2013-04-01'

```
SQL> SELECT empno, ename, job, sal,
  2         to_char(vt_start,'YYYY-MM-DD') AS vt_start,
  3         to_char(vt_end,'YYYY-MM-DD') AS vt_end,
  4         to_CHAR(dt_start,'YYYY-MM-DD') AS dt_start,
  5         to_char(dt_end,'YYYY-MM-DD') AS dt_end
  6    FROM empv AS OF period FOR dt DATE '2013-04-01'
  7   WHERE empno = 7788 AND
  8         (vt_start <= DATE '2014-01-01' OR vt_start IS NULL) AND
  9         (vt_end > DATE '2014-01-01' OR vt_end IS NULL)
 10   ORDER BY vt_start NULLS FIRST, dt_start NULLS FIRST;

EMPNO ENAME JOB        SAL VT_START   VT_END      DT_START   DT_END
----- ----- ------- ----- ---------- ---------- ---------- ----------
 7788 Scott Manager  6000 2014-01-01             2013-03-24

1 row selected.
```

AGENDA

1. Introduction
2. Temporal Validity
3. Temporal Flashback Query
4. Tri-Temporal Model in Action
5. Core Messages

Core Messages – New Temporal 12c Features

Temporal Validity & Flashback Query

- Sound concepts but incomplete
- We miss
  - Temporal DML API
  - Temporal integrity constraints
  - Temporal join
  - Temporal aggregation
- Start using 12c temporal semantics for your valid-time or decision-time periods

Core Messages – Modeling

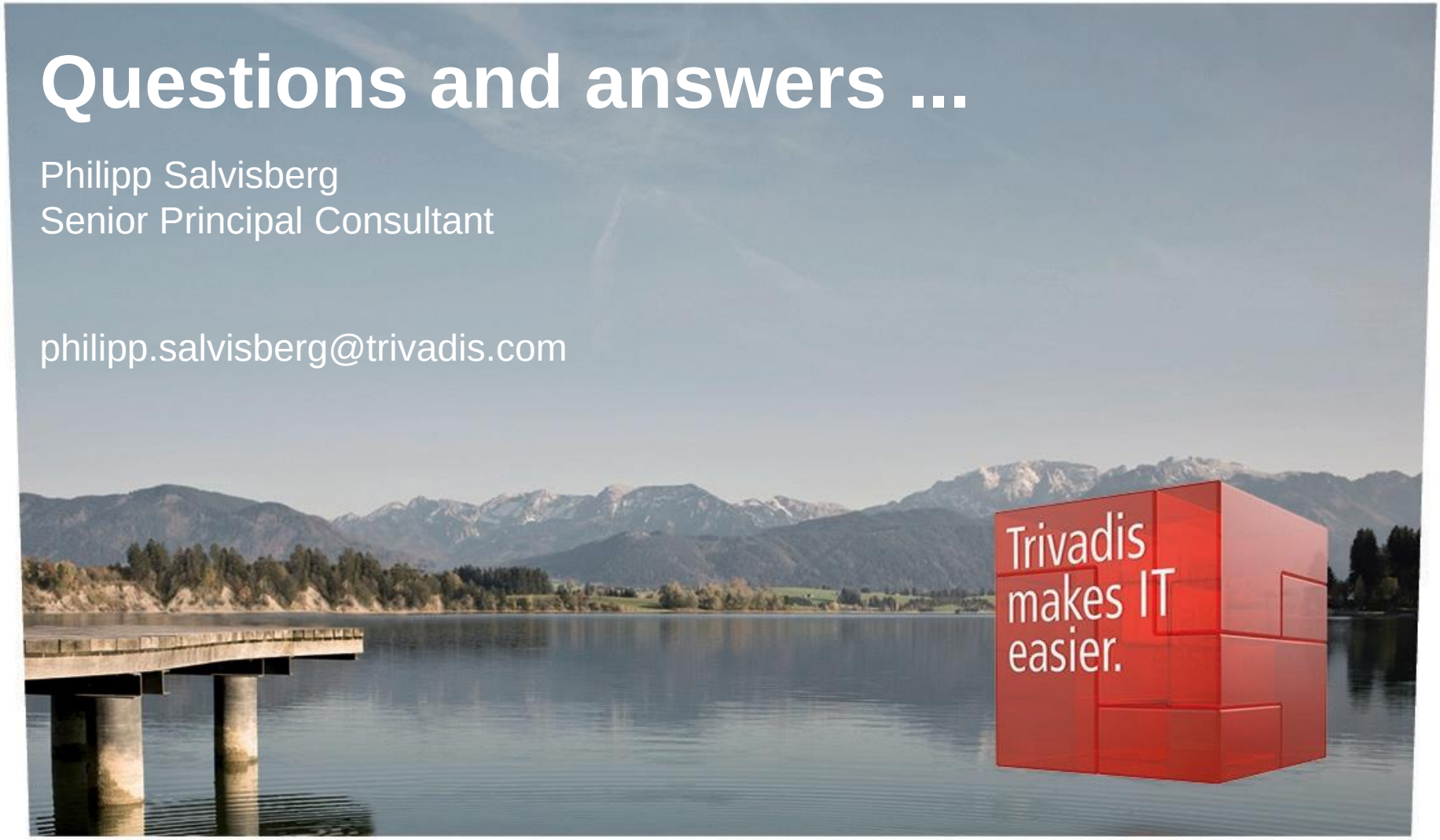
Multi-temporal Models

- In the real world we use multiple temporal periods
- A temporal period makes a model more complex
- Models are simplifications of the real world, based on requirements and budget
- Use standardized tooling to apply DML on temporal data
- Use Flashback Data Archive (Total Recall) to model the transaction time

Questions and answers ...

Philipp Salvisberg
Senior Principal Consultant

philipp.salvisberg@trivadis.com



BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MUNICH STUTTGART VIENNA

trivadis
makes IT easier.