GoldenGate

How to start such a project?

Mathias Zarick
Nuremberg, Nov. 17th 2015

Introduction – Mathias Zarick

- Principal Consultant at Trivadis Delphi GmbH in Vienna
- Graduated from University of Rostock / Computer Science
- Trainer
  - Data Guard, Architecture and Internals for advanced DBAs, Maximum Availability Architecture Workshop, Grid Infrastructure
- E-Mail: Mathias.Zarick@trivadis.com
- Main focus:
  - Oracle database
  - Oracle high availability projects (Real Application Clusters, Data Guard, Maximum Availability Architecture, Replication with Streams and GoldenGate)
  - Backup/Recovery
  - Development Lead of Trivadis Toolbox
  - Developer of TVD-Standby
  - Research projects in Trivadis Technology Center (TTC)

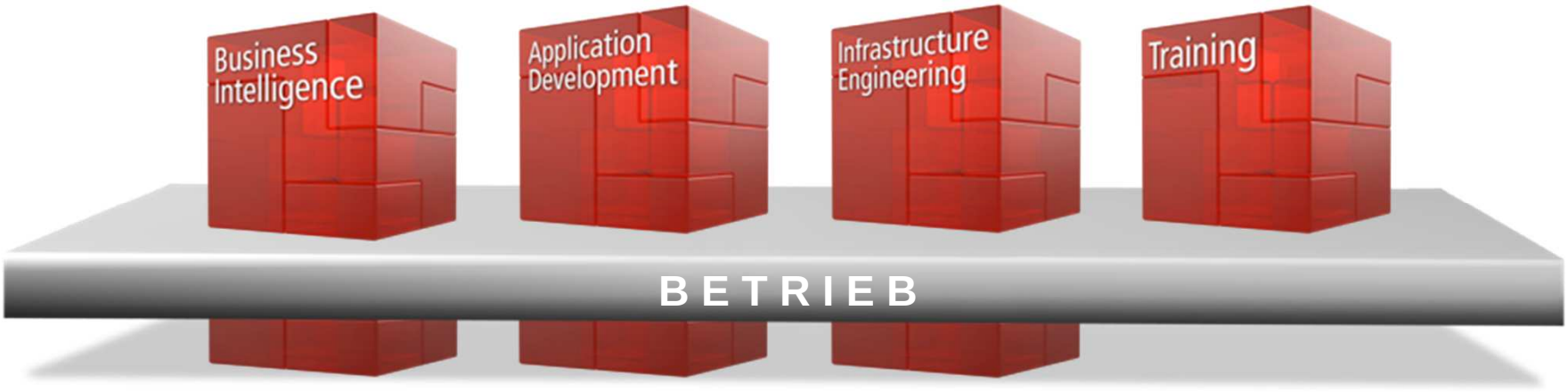
ORACLE®
Certified Master
Oracle Database 11*g*
Administrator

ORACLE®
Certified Professional

trivadis
makes IT easier.

Unser Unternehmen.

Trivadis ist führend bei der IT-Beratung, der Systemintegration, dem Solution Engineering und der Erbringung von IT-Services mit Fokussierung auf ORACLE - und Microsoft -Technologien in der Schweiz, Deutschland, Österreich und Dänemark. Trivadis erbringt ihre Leistungen aus den strategischen Geschäftsfeldern:



Trivadis Services übernimmt den korrespondierenden Betrieb Ihrer IT Systeme.

trivadis makes IT easier.

Mit über 600 IT- und Fachexperten bei Ihnen vor Ort.

- 14 Trivadis Niederlassungen mit über 600 Mitarbeitenden.
- Über 200 Service Level Agreements.
- Mehr als 4'000 Trainingsteilnehmer.
- Forschungs- und Entwicklungsbudget: CHF 5.0 Mio.
- Finanziell unabhängig und nachhaltig profitabel.
- Erfahrung aus mehr als 1'900 Projekten pro Jahr bei über 800 Kunden.

trivadis
makes IT easier.

Agenda

1. **What is Replication and GoldenGate?**
2. Possible Topologies and Usecases
3. Rules for Successful Replication Setups
4. Conflict Resolution
5. Some More Tips
6. Conclusion

What is Replication and GoldenGate?

What is Replication? (1)

- Wikipedia:
  - sharing information so as to ensure consistency between redundant resources
  - improve reliability, fault-tolerance, or accessibility
- Types with Oracle database
  - physical replica:
    - physical standby database with media recovery (Active Data Guard / ADG)
    - easier to maintain/operate
  - logical replica:
    - apply SQL (using own programs or provided tools/technology)
    - harder to maintain/operate

What is Replication? (2)

- Evolution at Oracle:
  - Materialized view (snapshot) replication
  - Advanced replication
  - Logical Standby
  - Oracle Streams
  - Oracle GoldenGate
- Third Party
  - Shareplex (Quest now Dell)
  - DataMirror / Rocket iCluster

Why Replication?

- (Zero-Downtime) Migrations
  - change OS/platform
  - change DB Version or even DB vendor
  - endianness changes
  - character set changes
  - etc.
- High availability / run several replica
  - to be able to failover to a standby system
  - or even to be able to update all of them simultaneously to avoid SPOFs introduced by one central database (Multimaster replication)
- Offload productive transactional systems
  - read-only reporting systems
  - staging area for a data warehouse
- Other
  - historify
  - centralization / consolidation of data
  - …

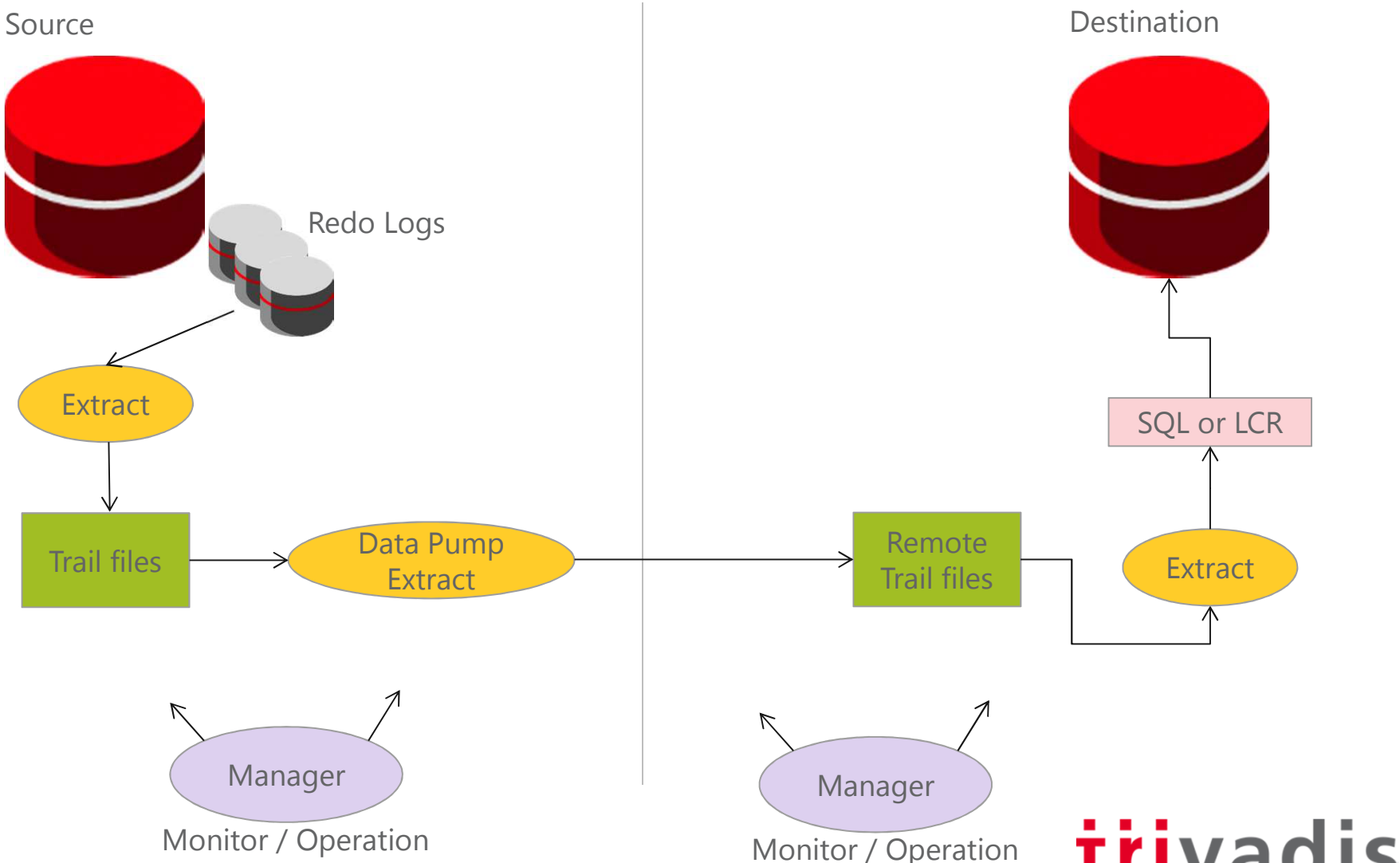
Why GoldenGate?

- All following pros and cons are summed up in comparison to former Oracle database replication technologies
- Pros
  - current and most up-to-date replication technology by Oracle (Streams is deprecated)
  - better transaction throughput / performance in comparison to Streams: but in most cases only if using classic processes
  - heterogeneous replication support: e.g. replicate data from DB/2 to Oracle
  - only committed transactions are transferred to target (in contrast to Streams)
- Cons
  - separately licensable product (Oracle Fusion Middleware product suite)
  - own proprietary interface and language
  - can get more complex easily
  - external processes to be run and operated
- Hint: see “Oracle Streams to GoldenGate Migration Utility (Doc ID 1912338.1)”

GoldenGate Architecture

- 3 types of processes / similar to Streams
- Capture – Extract
  - capture transactions/changes
  - can be classic/integrated
  - downstream configurations are supported (transfer redo logs and capture on another system/database)
  - writes to local (recommended) or remote trails
- Propagation – Data Pump Extract
  - optional but recommended
  - reads local trails and writes remote trails
  - no database link but an own TCP/IP protocol based communication is used
- Apply – Replicat
  - can be classic/integrated
  - reads data from trails
  - construct and process SQL (classic) or LCR (integrated)
- Transformations, mappings, filters can be applied on all those processes

GoldenGate Architecture (2)



Classic vs. Integrated Capture
Why? What to choose?

- Classic Capture reads online redo logs directly
  - good performance!
  - restrictions: some data types / structures are unsupported, e.g. compressed tables, CDBs
  - no good integration with other HA technologies, e.g. RAC and Data Guard
  - special DDL triggers (DDL support objects) needed
- Integrated Capture internally creates a Streams-like capture which starts LogMiner
  - LCRs are created and written to queues, those LCRs are transformed to trails
  - everything which was already resolved with Streams is resolved now as well
  - LogMiner Performance
  - less restrictions, e.g. CDB support, one extract can read from several PDBs
  - better integration with RAC/ASM, Data Guard, Transparent Data Encryption
- In most setups you would choose integrated, because
  - it offers better support and has less limitations
  - it allows to combine with HA components easier
  - Oracle always emphasizes this technology regarding future developments

Classic vs. Integrated Apply
Why? What to choose?

- Nonintegrated/Classic Apply = Replicat reads trails, constructs SQL and performs it via OCI connection
  - parallelization with coordinated replicat configuration
- Integrated Replicat internally creates a Streams-like apply
  - trails → LCR → AQ Queue → Apply (with parallelism)
  - better suited for heavy workloads (parallelization with preserving of integrity and atomicity of source transactions)
  - better integration with RAC/ASM, Data Guard, Transparent Data Encryption
- In most setups you would choose integrated, because
  - it offers better support and has less limitations
  - it allows to combine with HA components easier
  - Oracle always emphasizes this technology regarding future developments
- In any case you need one replicat for one target database, in a Multitenant configuration, one replicat can only write to one PDB

Okay – You want a Setup Example with Syntax?

```
GGSCI (zam32) 1> DBLOGIN USERID ggadm@nad
GGSCI (zam32) 2> ADD TRANDATA zam.tab1
GGSCI (zam32) 3> EDIT PARAMS epn

    EXTRACT epn
    USERID ggadm@nad, PASSWORD *********
    TRANLOGOPTIONS EXCLUDEUSER GGADM
    LOGALLSUPCOLS
    UPDATERECORDFORMAT COMPACT
    DDL INCLUDE MAPPED
    EXTTRAIL /home/oracle/ggtrail/NAD/ep
    TABLE zam.*;

GGSCI (zam32) 4> EDIT PARAMS de

    EXTRACT de
    USERID ggadm@nad, PASSWORD *********
    RMTHOST zam33, MGRPORT 7809
    RMTTRAIL /home/oracle/ggtrail/ZERATUL/rp
    TABLE zam.*;
```

Syntax Example Continuation

```
GGSCI (zam32) 5> REGISTER EXTRACT epn DATABASE
GGSCI (zam32) 6> ADD EXTRACT epn, INTEGRATED TRANLOG, BEGIN NOW
GGSCI (zam32) 7> ADD EXTTRAIL /…/ggtrail/NAD/ep, EXTRACT epn
GGSCI (zam32) 8> ADD EXTRACT de, EXTTRAILSOURCE
                 /…/ggtrail/NAD/ep
GGSCI (zam32) 9> ADD RMTTRAIL /…/ggtrail/ZERATUL/rp, EXTRACT de


GGSCI (zam33) 1> DBLOGIN USERID ggadm@zeratul
GGSCI (zam33) 2> EDIT PARAMS app

    REPLICAT app
    USERID ggadm@zeratul, PASSWORD *********
    ASSUMETARGETDEFS
    MAP zam.*, TARGET zam.*,
    COMPARECOLS (ON UPDATE KEYANDMOD, ON DELETE ALL);;

GGSCI (zam33) 3> ADD REPLICAT app, INTEGRATED,
                 EXTTRAIL /home/oracle/ggtrail/ZERATUL/rp
```

Agenda

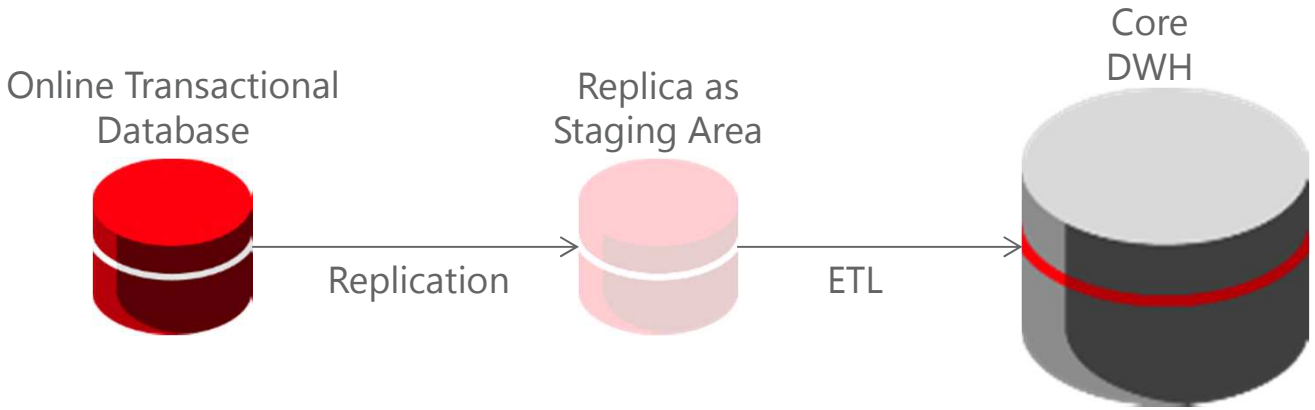
1. What is Replication and GoldenGate?
2. **Possible Topologies and Usecases**
3. Rules for Successful Replication Setups
4. Conflict Resolution
5. Some More Tips
6. Conclusion

Possible Topologies and Usecases

Topologies

- There are infinite possibilities for GoldenGate topologies
- Remember: flexible is good, but you need to operate and therefore understand it as well
- Complexity is the biggest enemy of high availability
- So try to design it as simple as possible
- KISS: keep it small and simple

Staging Area for Data Warehouse



- Often used like that with Streams or GoldenGate
- But there is a better solution, which uses physical replication instead of logical: Combination of Data Guard Snapshot Standby and Transportable Tablespaces

trivadis
makes IT easier.

Solution for Staging Area in Near Real-Time DWH – Efficient in Refresh and Easy to Operate

Technical White Paper

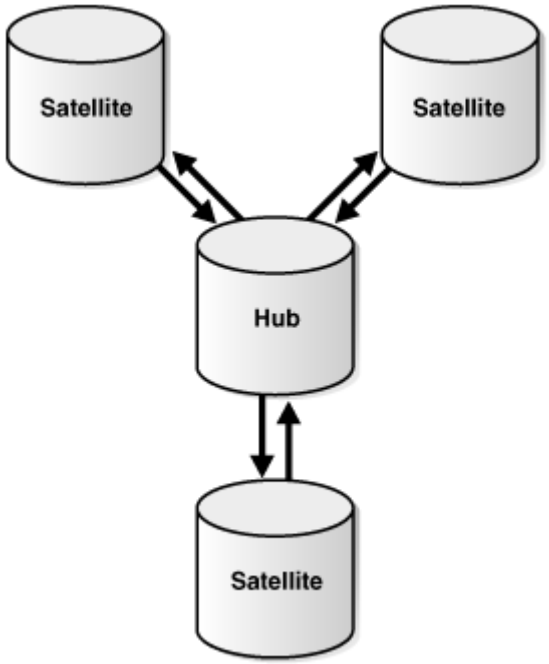
Mathias Zarick, Karol Hajdu
Senior Consultants
March-2011

- See http://www.trivadis.com/sites/default/files/downloads/WhitePaper_Solution_for_Staging_Area_01.pdf

trivadis
makes IT easier.

Hub and Spoke Replication

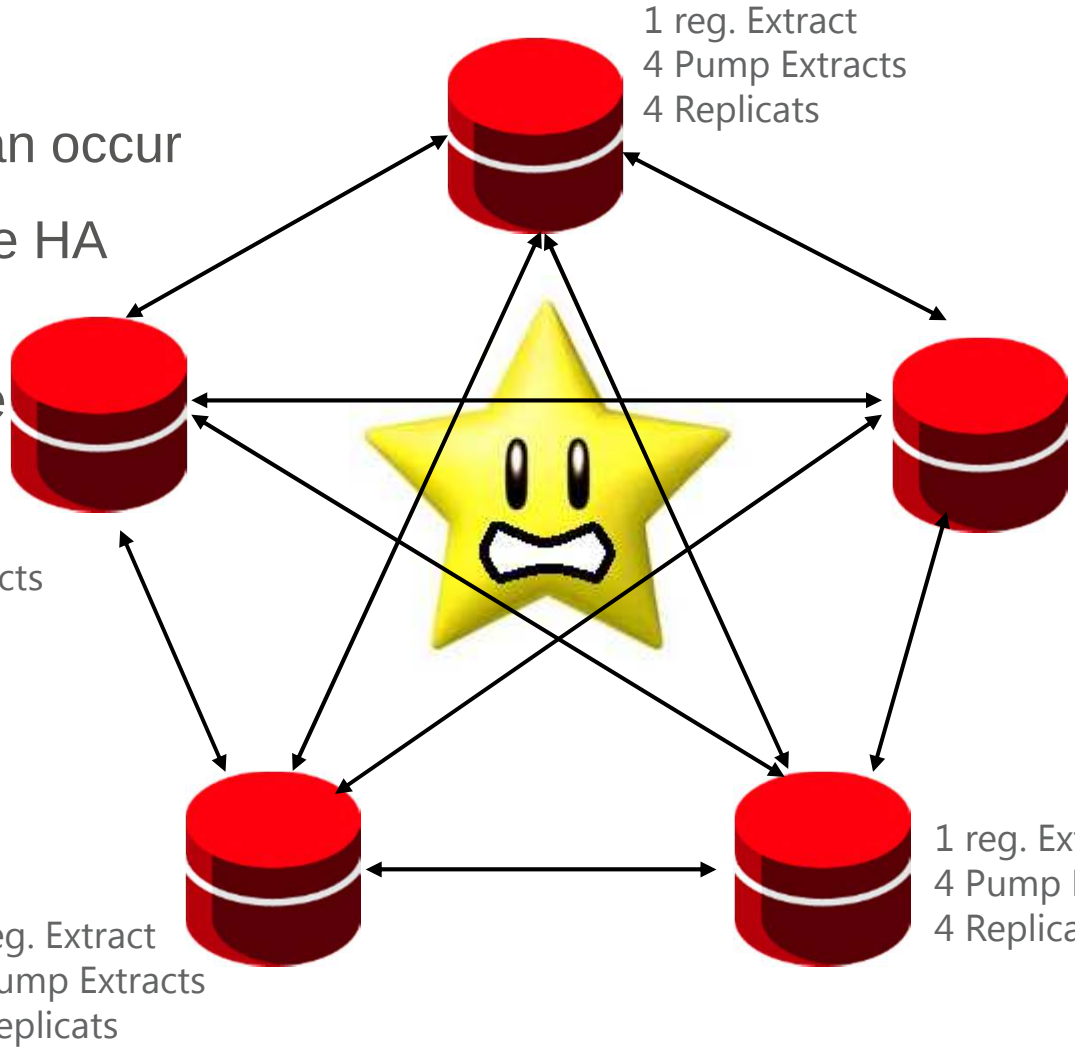
- Spokes or satellites are connected to central hub
- Conflicts can occur – it is a variant of a multi-master replication
- Can provide HA
- Overall availability dependent on hub availability



trivadis makes IT easier.

Multi-master Replication

- Conflicts can occur
- Can provide HA
- Docs have an example for DBFS

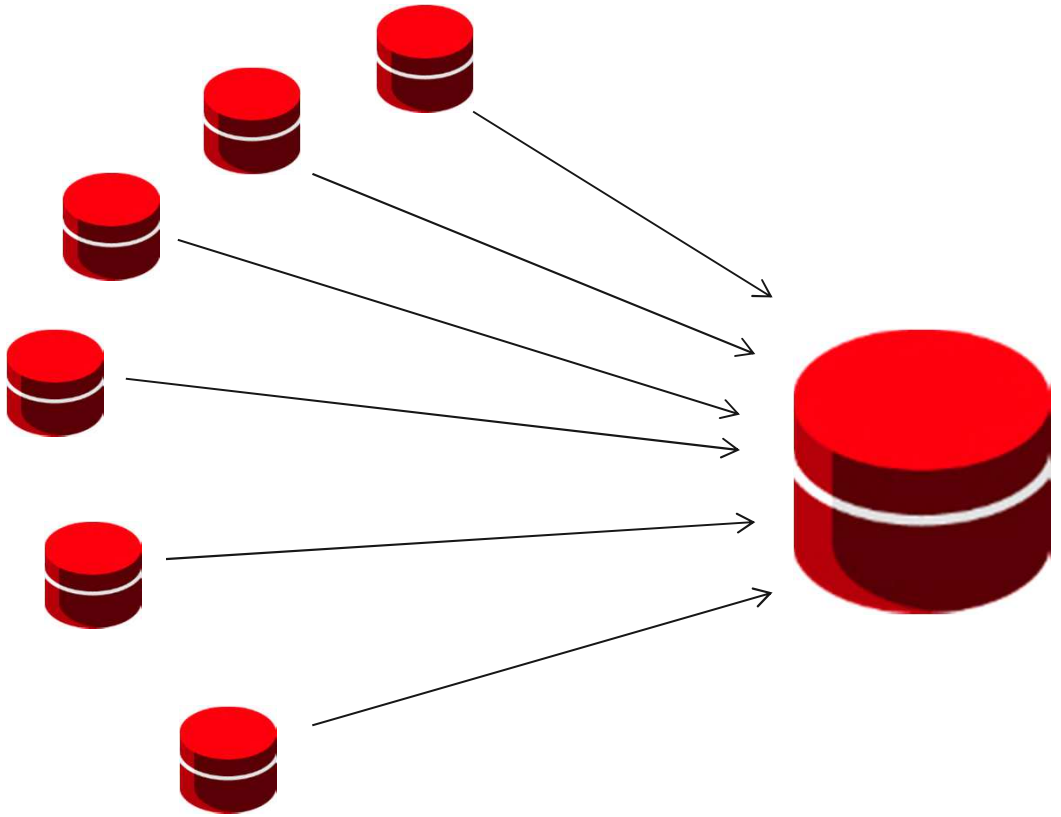


DO NOT FEAR! ☺

trivadis makes IT easier.

Consolidation of Data

- Data is transferred to a central location
- One central reporting / history database
- Deletes might not be transferred (historify)



Other Topologies

- Multi-master
- Reader farms to reduce workload on source: better use ADG (physical replication)
- Consolidation of data to central reporting database
- Historization of changes on OLTP database
  - DELETE on source is INSERT on destination
- Bidirectional setups for migrations
- Cascaded configurations / directed networks with intermediary systems/trails

Agenda

1. What is Replication and GoldenGate?
2. Possible Topologies and Usecases
3. **Rules for Successful Replication Setups**
4. Conflict Resolution
5. Some More Tips
6. Conclusion

Rules for Successful Replication Setups

Project Plan and Operation

1. Define scope
2. Run a “Proof of concept”
3. Design a suitable topology and setup
4. Define rules and processes for the environment, e.g. how to handle DDL or extend the replication setup (new schemas, tables, columns etc.)
5. Prepare a proven set of scripts, programs, documentation etc. for setup and operation (see later)
6. Bring to operation

Operational Loop

1. Detect weaknesses in setup and monitoring
2. Improve the setup more and more with the help of suitable test systems

Typical Situation after a PoC

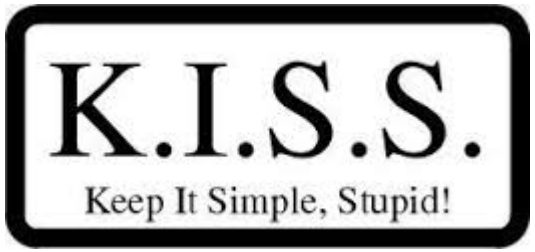
- It is believed, that usecase is suitable and replication process works as intended
- Some application tests show success
- Some scripts for replication setup are created
- Scripts are run, replication is online and productive

- And now?
- Is this enough?
- How to prove that everything is running as intended, even after some weeks?
- Monitoring?
- Are you prepared for the first runtime error? What to do then?

To be Successful, Certain Rules should be Considered

- Following 8 Rules are based on experience with a lot of replication setups with different technologies
  - Advanced Replication
  - Materialized View Replication
  - Own PL/SQL based replications
  - Streams
  - GoldenGate

1. KISS – KISS – KISS !!!



- Keep It Small and Simple!
- Keep It Simple, Stupid!
  - make sure you really need it that way you design it, e.g.
    - Is the asynchronous transmission acceptable?
    - Do you really need a read write database at destination?
  - make sure, that all replicated tables have a primary key
  - avoid DDL (even if supported), or at least define a tested process to perform DDL
  - avoid incremental changes to your replication setup, or at least define a tested process to do so

2. Configure Databases Correctly and Avoid Unnecessary Overhead

- Capture database:
  - Archivelog mode
  - force logging
  - minimal supplemental logging
  - As of 11.2.0.4: SET enable_goldengate_replication=TRUE;
- Table Level: Any additional needed supplemental logging

```
add trandata hr.employees
add trandata hr.heartbeat
...
```

- Worst practice:

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA
(PRIMARY KEY, UNIQUE, FOREIGN KEY) COLUMNS;
```

- Transmit only data you really need at target database

3. Implement Critical Components

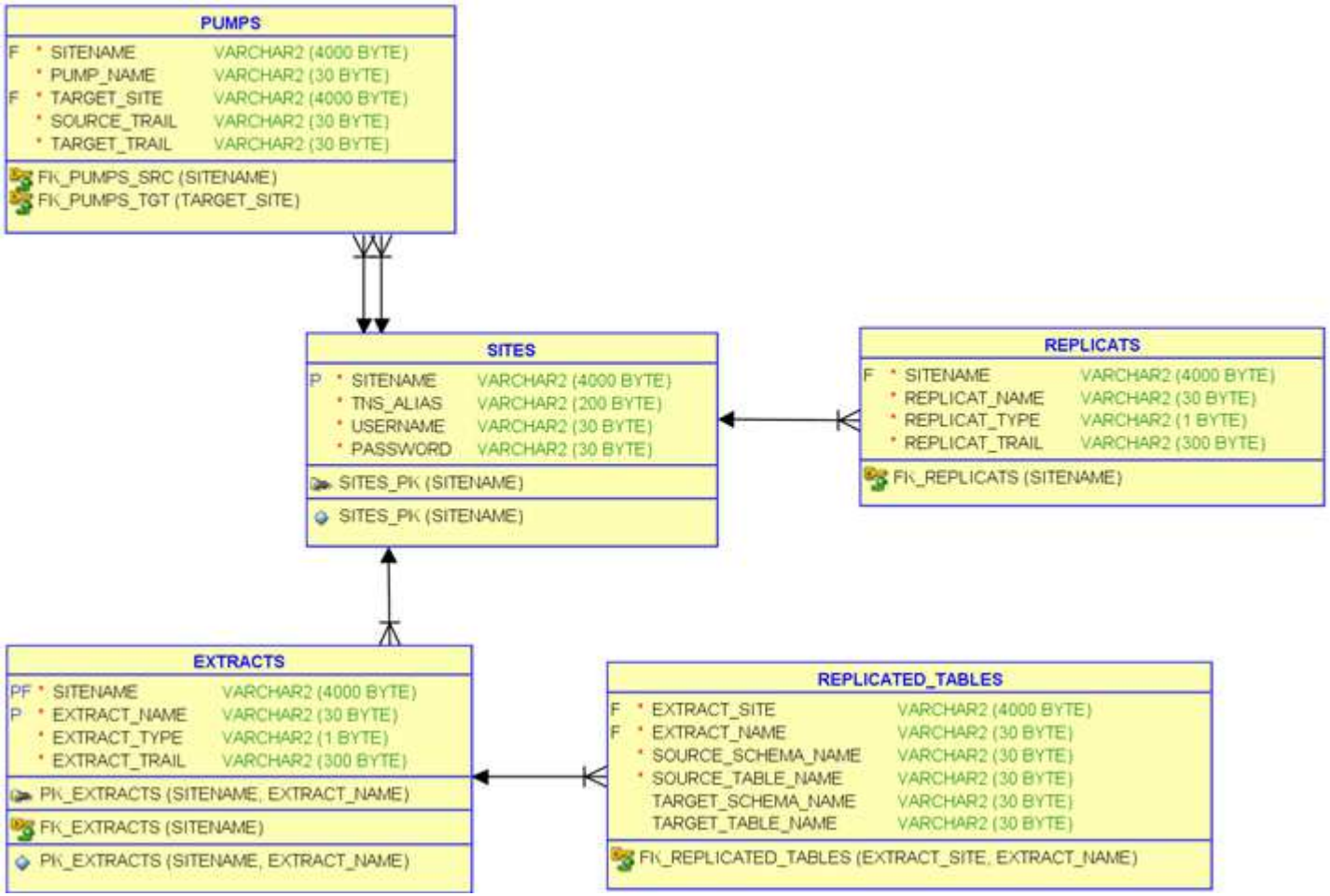
a. Latest recommended patches, see “Oracle GoldenGate -- Oracle RDBMS Server Recommended Patches (Doc ID 1557031.1)”

b. It is recommended to have an own repository (e.g. in GGADM schema), which describes your setup and can be used to generate scripts, which are used for
   - (re-)setup of all processes (for all objects)
   - deinstallation
   - verify
   - rectify

c. Implement a heartbeat table, job and monitor

d. Several different monitors
   - processes (manager, extract, data pump, replicat)
   - log files
   - resource utilization
   - transport lag
   - apply lag

Example Repo



- Actual design is dependent on complexity of the replication
- Again, consider KISS!

Heartbeat

- It is also a best practice to include a heartbeat table in the setup, which is updated again and again
- This table is suitable for monitoring purposes

```
CREATE TABLE heartbeat (
  site VARCHAR2(4000) CONSTRAINT heartbeat_pk PRIMARY KEY,
  stamp DATE);
INSERT INTO heartbeat (site, stamp)
SELECT dbms_reputil.global_name, sysdate FROM dual;
COMMIT;
BEGIN
  dbms_scheduler.create_job(
    job_name        => 'heartbeat_update_job',
    job_type        => 'plsql_block',
    job_action      => 'UPDATE heartbeat SET stamp = sysdate
                        WHERE site = dbms_reputil.global_name;',
    repeat_interval => 'freq=minutely;bysecond=0,15,30,45',
    comments        => 'heartbeat for goldengate replication',
    enabled         => TRUE
  );
END;
/
```

4. Be Prepared to Verify the Replicated Data

- Compare data between source and destination
- Most setups aim to have same data on different sites (though sometimes this is not the case, sometimes it is intentionally different (transformations))
- So keeping the data in sync can be a considered as a general wish in many setups
- You should be able to check for synchronicity/consistency and know how to synchronize/rectify
- If data differs at different sites we call it: data diverges, we will have data divergence
- Often we want to have the opposite: convergent data, convergence

Compare Tables

- Compare 2 tables like this (base algorithm)
- $(A \setminus B) \cup (B \setminus A)$

```
SELECT count(*) FROM (
(SELECT * FROM tab@vizrtdb1
MINUS
SELECT * FROM tab@vizrtdb2)
UNION ALL
(SELECT * FROM tab@vizrtdb2
MINUS
SELECT * FROM tab@vizrtdb1)
)
```

The brackets matter ☺

- Does not work for LOB and LONG columns (but there are workarounds if needed)
- If too much data is transferred – think about
  - comparing only PKs and “important” columns
  - generate hashes based on row data to reduce network bandwidth
  - partition the data and checks, e.g. compare only latest month

Example: Generate Compare Code – Compare (nearly) everything

- Function code is PL/SQL and generated by PL/SQL
  - LOBs are excluded, LONG does not exist
  - 1 reference site, 4 comparison sites, 51 tables

```
CREATE OR REPLACE FUNCTION compare RETURN NUMBER AS
  v_sum NUMBER; v_count NUMBER;
BEGIN
  v_sum := 0;
  SELECT count(*) INTO v_count FROM (( SELECT ... FROM
  PILOT.CATEGORY@VIZRTDB1 MINUS SELECT ... FROM
  PILOT.CATEGORY@VIZRTDB2 ) UNION ALL ( SELECT ... FROM
  PILOT.CATEGORY@VIZRTDB2 MINUS SELECT ... FROM
  PILOT.CATEGORY@VIZRTDB1)); v_sum := v_sum + v_count;
  ... /* 204 (51 x 4) selects in total */
  RETURN v_sum;
END;
/
-- runs for some time depending on data amount
-- should return zero and be included in monitoring
SELECT compare FROM dual;
```

Other Verification and Converge Options/Tools

- Basic approach is always the same (see previous slide)

- Available tools and packages:
  - Oracle GoldenGate Veridata: GUI and CLI, extra licensable, complex
  - DBMS_COMPARISON: integrated in DB / PL/SQL
  - Older: DBMS_RECTIFIER_DIFF

5. Be able to Synchronize Replicated Data and to Re-Setup

- You should be able to rectify (make convergent) divergent data incrementally
- In some cases you will even need a complete re-setup (deinstall and install) possibly with new data instantiation
- one-button-approach would be aimed, but is hard to realize with GoldenGate

- You can use converge tools as mentioned on previous slide to overcome need of complete instantiations
- Use them in combination with the process description in the section “Resynchronizing an out-of-sync table” of “GoldenGate Configuration for Stability and Recovery (Doc ID 1451514.1)”
  - bypass table in question at regular replicat
  - manual rectify of table
  - create a temporary replicat which handles the rectified object (*HANDLECOLLISIONS*)
  - several adaptions of replicats and restarts of replicats and extract
  - removal of temporary replicat

6. Perform Different Kind of Recoveries with Involved Databases

- Complete recovery of source / target databases
- Incomplete recovery of source / target databases
- Repair GoldenGate / some scenarios would need a re-setup
- Document this process carefully

7. Training for Operation

- Install and deinstall the replication again and again before going to production and later on the test system
- Establish a process for cloning the production environment to test system again and again with a representative set of data and replication components
- Do failure tests
- Provoke some conflicts
- Repair some error situations, e.g. truncate a table at target
- Get familiar with log messages in ggserr.log
- Get familiar with your scripts and improve continuously

8. Documentation and Operation Manual

- It is essential to have an up-to-date documentation and manual
- All setup steps
- All about monitoring
- Howto’s
  - (Re-)setup
  - (Re-)instantiate
  - Check for convergence
  - Rectify / Resynchronize
  - Clone to test system
  - Recoveries: Complete and PITR

Agenda

1. What is Replication and GoldenGate?
2. Possible Topologies and Usecases
3. Rules for Successful Replication Setups
4. **Conflict Resolution**
5. Some More Tips
6. Conclusion

Conflict Resolution

Conflict Detection and Resolution

- What is a conflict?
  - mismatch between old data (before image of row) and actual data at destination database (update / delete)
  - typically introduced if same data is changed at almost the same time at different sites
  - uniqueness conflicts can happen if using same IDs/PKs at different sites
  - can also happen just because of asynchronous transmission

In case you are dealing with setups, which bear a risk for conflicts, consider the following

- Avoid conflicts by application
  - Partition your data
  - Avoid insert/uniqueness conflicts by generating unique numbers, e.g. SYS_GUID or sequences with certain increment
- Configure a good and exact conflict detection
  - this was a matter of course for Advanced Replication and Streams but is not for GoldenGate!
  - GoldenGate: in most (default) setups only PKs are compared

Conflict Detection – The Lazy and the Exact Way

- Following statement at origin

```
UPDATE employees SET job_id='SH_CLERK', salary=salary*1.2
WHERE first_name='TJ' AND last_name='Olson'
```

Is transported to destination database exactly or kind of lazy

- This is exact way (like Advanced Replication and Streams always did by default)

```
UPDATE employees SET job_id='SH_CLERK', salary=2520
WHERE employee_id=132
AND job_id='ST_CLERK'
AND salary=2100
```

- We could even be more accurate by including all old column values
- The default way of most GoldenGate deployments – which I also call the "do not care" logic: only PKs are compared to match the row

```
UPDATE employees SET job_id='SH_CLERK', salary=2520
WHERE employee_id=132
```

Exact Conflict Detection in GoldenGate

- To get an accurate conflict detection in GoldenGate – which is recommended – you need to configure
  - *UPDATERECORDFORMAT COMPACT* in extract
  - Setting of
    - *LOGALLSUPCOLS* in extract (database level) or
    - *GETBEFORECOLS (ON UPDATE KEYANDMOD, ON DELETE ALL)* in extract (table level)
  - *COMPARECOLS (ON UPDATE KEYANDMOD, ON DELETE ALL)* in replicat
  - This works for Oracle databases as of release 11.2.0.4 and for GoldenGate as of release 12.1.2
- For details see "**My conflict with the default conflict detection of Oracle GoldenGate**" http://blog.trivadis.com/b/mathiaszarick/archive/2015/04/17/my-conflict-with-the-default-conflict-detection-of-oracle-goldengate.aspx

Conflict is Detected – And Now?

- Let it crash?

```
2015-09-03 17:45:50  WARNING OGG-01004  Oracle GoldenGate Delivery for Oracle,
rep_c_v.prm:  Aborted grouped transaction on 'S.HEARTBEAT', Database error 1403
(OCI Error ORA-01403: no data found, SQL <UPDATE "S"."HEARTBEAT" x SET
x."STAMP" = :a3 WHERE x."SITE" = :b0 AND x."STAMP" = :b1>).
…
2015-09-03 17:45:50  ERROR   OGG-01668  Oracle GoldenGate Delivery for Oracle,
rep_c_v.prm:  PROCESS ABENDING.
```

- Or continue?

1. We are used to have an error logged in databases (accessible by SQL) and
2. The option to stop application of newly arriving data (at least with Streams)

1. Can be done with exception table(s)
2. Can be done with *REPERROR (DEFAULT, ABEND)* which is default, error is logged in ggserr.log only

Exception Table Management Can Get Painful

- An exception table for each target table?
  - *MAP* statement for each table in replicat's param file
  - What has to be done after DDL, e.g. column add or modify?
  - How to query on this from a central point of view?
  - only useful chance to log column values from failed transaction in the database
- A single generic exception table
  - *MAPEXCEPTION* method, you need to have all possible columns in a very wide table → in most cases absolutely impractical
  - handle exceptions with a *MACRO* → here you have a central queryable table, but you do not have logged all column values of the failed transaction, only the SQL, which shows only bind variables and maybe gets truncated (4000 Bytes restriction)

- You set *REPERROR (DEFAULT, EXCEPTION)* to implement this.
  Problem: after an error, the replicat continues ☹
  "Log to DB and stop afterwards" is impossible (at least I have to clue how to do this)

Example: Exception Table for a Specific Target Table

- Create an exception table for each target table

```
CREATE TABLE hr.employees_exception
AS SELECT * FROM hr.employees WHERE 1=2;
ALTER TABLE hr.employees_exception ADD (
  error_date    DATE,          -- protocolled date and time of error occurrence
  optype        VARCHAR2(20),  -- operation type
  errno         NUMBER,        -- error number
  errmsg        VARCHAR2(4000) -- error message and failing SQL
);
```

- Map the failed transaction to exception table

```
REPERROR (DEFAULT, EXCEPTION)
MAP hr.employees, TARGET hr.employees,
COMPARECOLS (ON UPDATE KEYANDMOD, ON DELETE ALL);
MAP hr.employees, TARGET hr.employees_exception,
EXCEPTIONSONLY,
INSERTALLRECORDS
COLMAP (USEDEFAULTS,
ERROR_DATE = @DATENOW (),
optype = @GETENV ('LASTERR', 'OPTYPE'),
errno  = @GETENV ('LASTERR', 'DBERRNUM'),
errmsg = @GETENV ('LASTERR', 'DBERRMSG'));
```

Before image from trail is not logged!

Expansion of Example: Before Images are also Logged

- Add the BEFORE columns to exceptions table

```
ALTER TABLE hr.employees_exception ADD (
 b_employee_id    NUMBER(6),
 ...
 b_department_id  NUMBER(4)
);
```

- Map to them using @BEFORE

```
REPERROR (DEFAULT, EXCEPTION)
MAP hr.employees, TARGET hr.employees,
COMPARECOLS (ON UPDATE KEYANDMOD, ON DELETE ALL);
MAP hr.employees, TARGET hr.employees_exception,
EXCEPTIONSONLY, INSERTALLRECORDS
COLMAP (USEDEFAULTS,
ERROR_DATE = @DATENOW (),
optype = @GETENV ('LASTERR', 'OPTYPE'),
errno  = @GETENV ('LASTERR', 'DBERRNUM'),
errmsg = @GETENV ('LASTERR', 'DBERRMSG')
b_employee_id = @BEFORE (employee_id),
...
b_department_id = @BEFORE (department_id)
);
```

Single Generic Exception Table

- You can use a macro to avoid repetition of same instructions for all tables

```
MACRO #exc_handler
BEGIN
TARGET ggadm.exceptions
, EXCEPTIONSONLY , INSERTALLRECORDS
, COLMAP (
  replicat_name   = @GETENV ('GGENVIRONMENT', 'GROUPNAME')
, table_name      = @GETENV ('GGHEADER',      'TABLENAME')
, optype          = @GETENV ('LASTERR',       'OPTYPE')
, errno           = @GETENV ('LASTERR',       'DBERRNUM')
, errmsg          = @GETENV ('LASTERR',       'DBERRMSG')
, errtype         = @GETENV ('LASTERR',       'ERRTYPE')
, logrba          = @GETENV ('GGHEADER',      'LOGRBA')
, logposition     = @GETENV ('GGHEADER',      'LOGPOSITION')
, committimestamp = @GETENV ('GGHEADER',      'COMMITTIMESTAMP')
);
END;

MAP hr.employees, TARGET hr.employees,
COMPARECOLS (ON UPDATE KEYANDMOD, ON DELETE ALL);
...
MAP hr.employees, #exc_handler() ;
MAP hr.departments, #exc_handler() ;
...
```

Conflict Resolution Methods are there, But Configuring them Can Easily Get Hard Work

- Example using a max timestamp column for resolution (largest wins)

```
MAP hr.employees, TARGET hr.employees,
COMPARECOLS (ON UPDATE KEYANDMOD, ON DELETE ALL),
RESOLVECONFLICT (UPDATEROWEXISTS, (DEFAULT, USEMAX (stamp)))
;
```

- Confirm the successful resolution in GGSCI

```
GGSCI (zam33) 6> stats replicat rep_c_v, reportcdr
…
*** Latest statistics since 2015-09-08 11:38:39 ***
        Total inserts                                       0.00
        Total updates                                       7.00
        Total deletes                                       0.00
        Total discards                                      0.00
        Total operations                                    7.00
        Total CDR conflicts                                 2.00
        CDR resolutions succeeded                           2.00
        CDR UPDATEROWEXISTS conflicts                       2.00
…
```

Other Conflict Resolution Methods

- INSERTROWEXISTS
  - Uniqueness problem, should not be necessary, better avoid
- UPDATEROWEXISTS
  - Classic update conflict
- UPDATEROWMISSING
  - There is no row which matches PK from trail, another variant of update conflict
- DELETEROWEXISTS
  - Classic delete conflict (introduced by concurrent delete or update)
- DELETEROWMISSING
  - There is no row which matches PK from trail, another variant of delete conflict
- SQLEXEC
  - Custom action for a trail which can contain also conflict handling

And Now? – Any Recommendations?

- Recommendation regarding conflict resolution!
  - No Multi-master replication → use exact conflict detection, no exception table, no automatic conflict resolution, leave default setting of *REPERROR (DEFAULT, ABEND)*
  - Multi-master replication → use exact conflict detection and following iterative approach
    1. start completely without automatic conflict resolution
    2. get to know the different variants of your application's conflicts and understand why they happen, try to avoid those conflicts from application side
    3. work with dedicated exception tables for affected tables
    4. resolve conflicts which cannot be avoided by application in a first step manually and document this carefully
    5. ONLY for those types of conflicts
       - for which you know why they happen and
       - which cannot be avoided by application design and
       - for which you know how to resolve them correctly

         → implement an automatic resolution

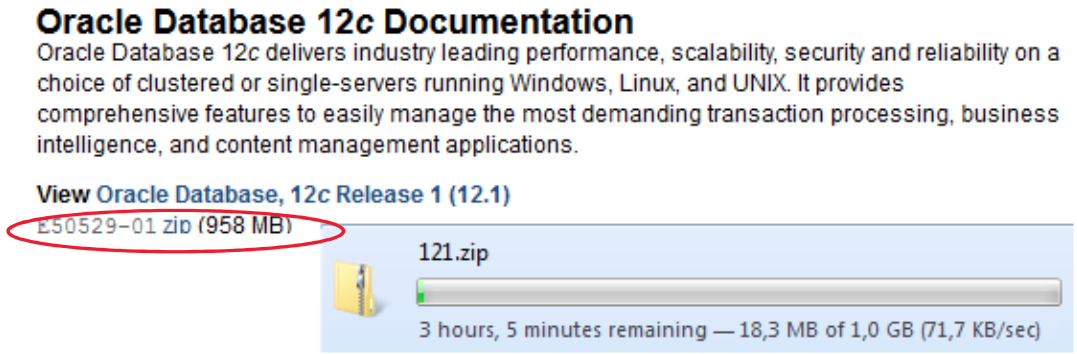
Agenda

1. What is Replication and GoldenGate?
2. Possible Topologies and Usecases
3. Rules for Successful Replication Setups
4. Conflict Resolution
5. **Some More Tips**
6. Conclusion

Some More Tips

Documentation Download?

- For database docs we are used to be able to download it for offline browsing



- But for GoldenGate this is not possible like this (only for older releases)
- Workaround:

```
wget \
     --recursive \
     --no-clobber \
     --page-requisites \
     --html-extension \
     --convert-links \
     --restrict-file-names=windows \
     --domains docs.oracle.com \
     --no-parent \
     http://docs.oracle.com/goldengate/1212/gg-winux/index.html
```

Ggsci Call is Kind of Strange

- Isn't this strange / non-intuitive?

```
# which ggsci
/u00/app/ggadm/product/ogg12.1.2/ggsci
# ggsci
Cannot load ICU resource bundle 'ggMessage', error code 2 - No
such file or directory
Aborted
```

- My solution:
  I use ggh

```
vi ${GG_HOME}/ggh
#!/bin/sh
oldpwd=$(pwd)
cd ${GG_HOME}
rlwrap -i -f ${GG_HOME}/ggsci.key ./ggsci
cd ${oldpwd}

chmod +x ggh
```

- Key file for rlwrap

```
grep '[A-Z][A-Z]' help.txt | tr ' ' '\n' | \
grep '[A-Z][A-Z]' | tr '[A-Z]' '[a-z]' | \
sort | uniq > ${GG_HOME}/ggsci.key
vi ${GG_HOME}/ggsci.key # manual removals of nonsense
                        # and special characters
```

Implement Restarts for Crashed GoldenGate Processes

- By default crashed processes do not try to restart, e.g. a network outage will crash a data pump process

```
ERROR   OGG-01232  Receive TCP params error: TCP/IP error
104 (Connection reset by peer), endpoint: zam33:7819.
```

- You should implement *AUTORESTART*, maybe also *AUTOSTART*
- Example for manager params file mgr.prm

```
AUTORESTART ER *,RETRIES 3, WAITMINUTES 4
AUTOSTART ER *
```

Agenda

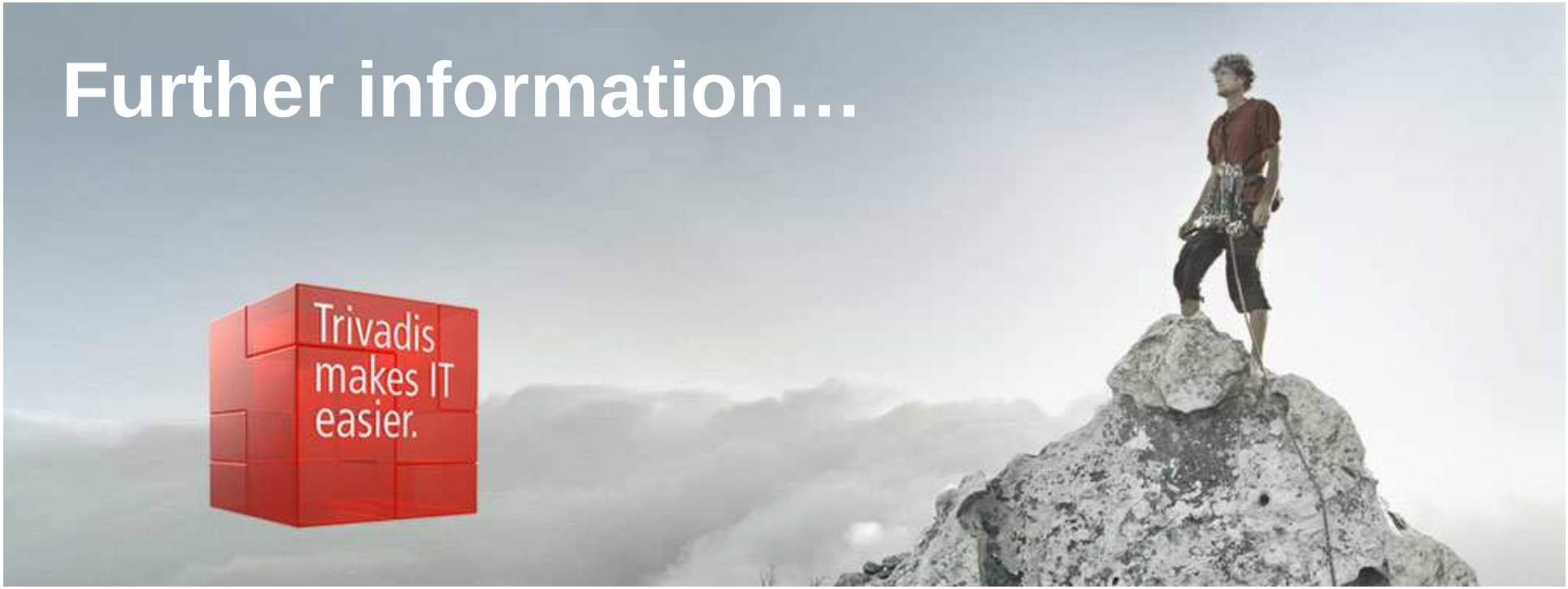
1. What is Replication and GoldenGate?
2. Possible Topologies and Usecases
3. Rules for Successful Replication Setups
4. Conflict Resolution
5. Some More Tips
6. **Conclusion**

Conclusion

Conclusion

- GoldenGate is an advanced replication technology
- Some things that just worked easily with Streams / Advanced Replication seem complicated with GoldenGate
- Follow the 8 mentioned rules for successful setups and operation: KISS
- Conflict detection and resolution:
  - Use exact detection!
  - Better avoid conflicts by application → if impossible use an iterative approach to get closer and closer right to the perfect resolution

Further information…



- Oracle GoldenGate 12c (12.1.2)

  http://docs.oracle.com/goldengate/1212/gg-winux/index.html
- Oracle GoldenGate 12c Implementer's Guide – John P Jeffries
- Oracle Goldengate 11g Complete Cookbook – Ankur Gupta
- Expert Oracle GoldenGate – Ben Prusinski, Steve Phillips, Shing Chung

trivadis
makes IT easier.

Questions and Answers

Mathias Zarick
Principal Consultant

Mathias.Zarick@trivadis.com

trivadis
makes IT easier.

Trivadis an der DOAG 2015

Ebene 3 - gleich neben der Rolltreppe

Wir freuen uns auf Ihren Besuch.

Denn mit Trivadis gewinnen Sie immer.