WACKER

DATENBANKREPLIKATION IN EINEM GLOBALEN LÖSUNGSVERBUND

Stefan Brandl, Wacker Chemie AG

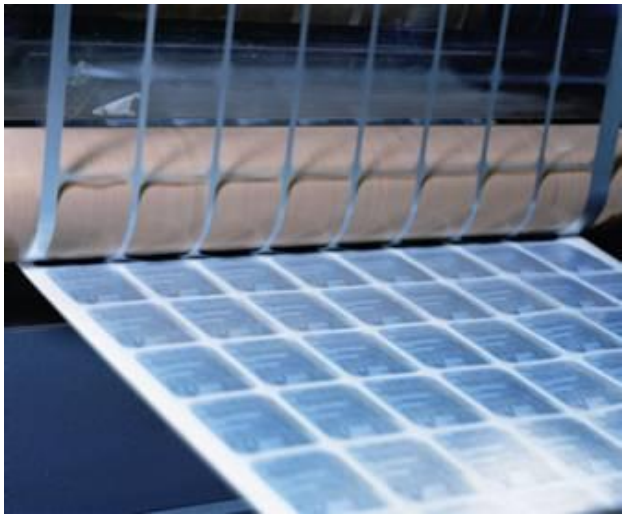
CREATING TOMORROW'S SOLUTIONS

AGENDA

- Kurzvorstellung Wacker Chemie
- Projektbeschreibung
- Umsetzung
- Fazit

KURZVORSTELLUNG WACKER CHEMIE

PRODUKTE UND LÖSUNGEN FÜR DIE GLOBALEN SCHLÜSSELINDUSTRIEN



- WACKER ist ein Technologieführer der Chemie und Halbleiterindustrie.
- Wir treiben technische Innovationen und die Entwicklung neuer Produkte für die globalen Schlüsselindustrien voran.
- Wir bieten Lösungen und Innovationen für ein breites Branchenspektrum.

- Automotive & Transport
- Bau
- Energie, Elektro&Elektronik
- Coatings & Paints
- Adhesives & Sealants
- Consumer Care
- Elastomere, Kunststoffe
- Papier, Folien & NIP
- Photovoltaik
- Textil, Leder & Faser
- Halbleiter
- Life Science

etc.

ERFOLGREICH SEIT FAST 100 JAHREN

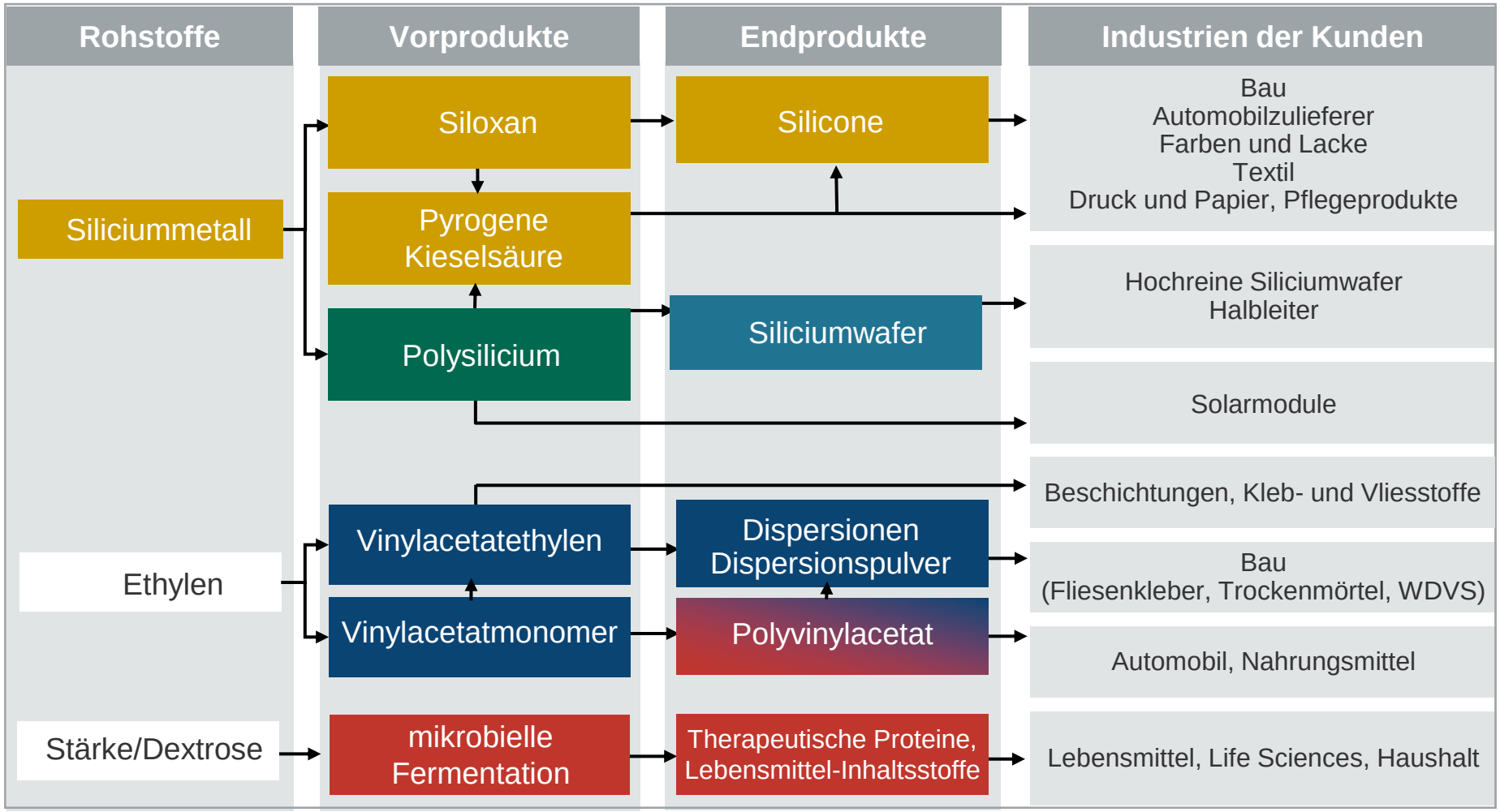
Wacker Chemie AG

- Gegründet 1914 von Dr. Alexander Wacker
- Konzernsitz ist München

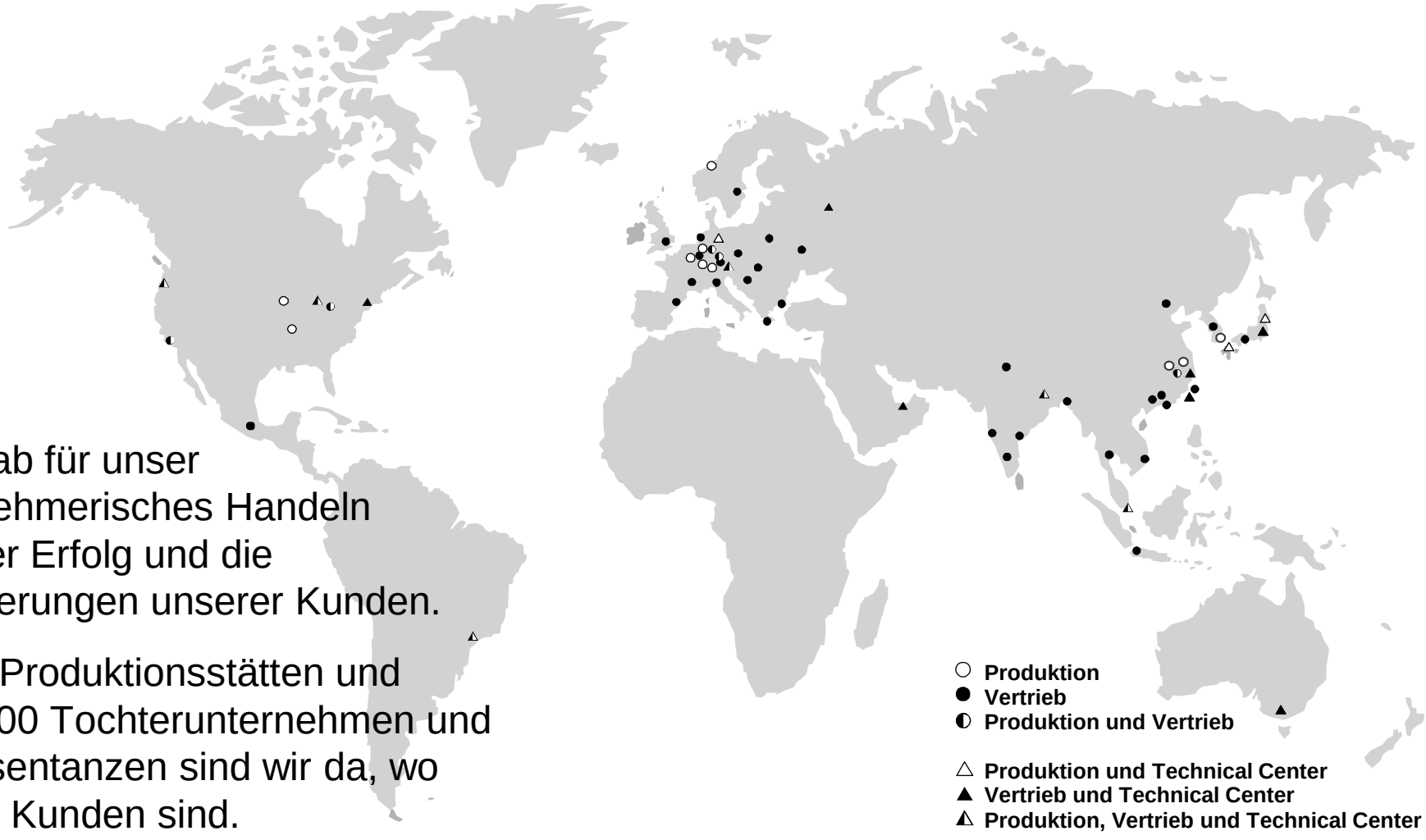
WACKER- Konzern (2011)

• Umsatz:	4,91	Mrd. €
• EBITDA:	1,1	Mrd. €
• F&E:	173	Mio. €
• Investitionen:	981	Mio. €
• Beschäftigte:	17.168	

WACKER: HOCH INTEGRIERTE PRODUKTION AUF BASIS VON DREI ROHSTOFFEN



WELTWEIT PRÄSENT UND NAH BEIM KUNDEN: DIE WELT VON WACKER



Maßstab für unser unternehmerisches Handeln sind der Erfolg und die Anforderungen unserer Kunden.

Mit 24 Produktionsstätten und etwa 100 Tochterunternehmen und Repräsentanzen sind wir da, wo unsere Kunden sind.

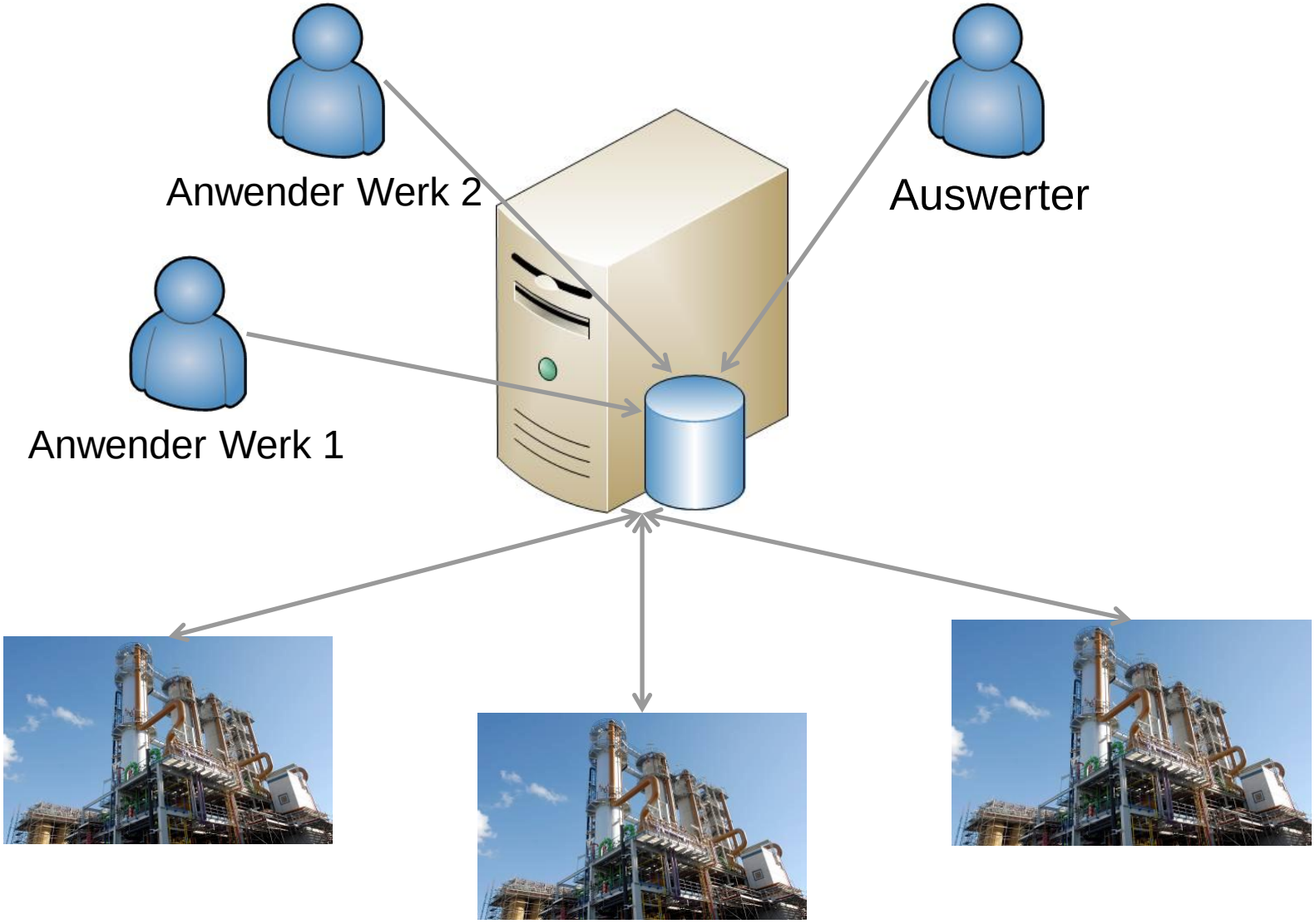
DAS PROJEKT

DAS PROJEKT

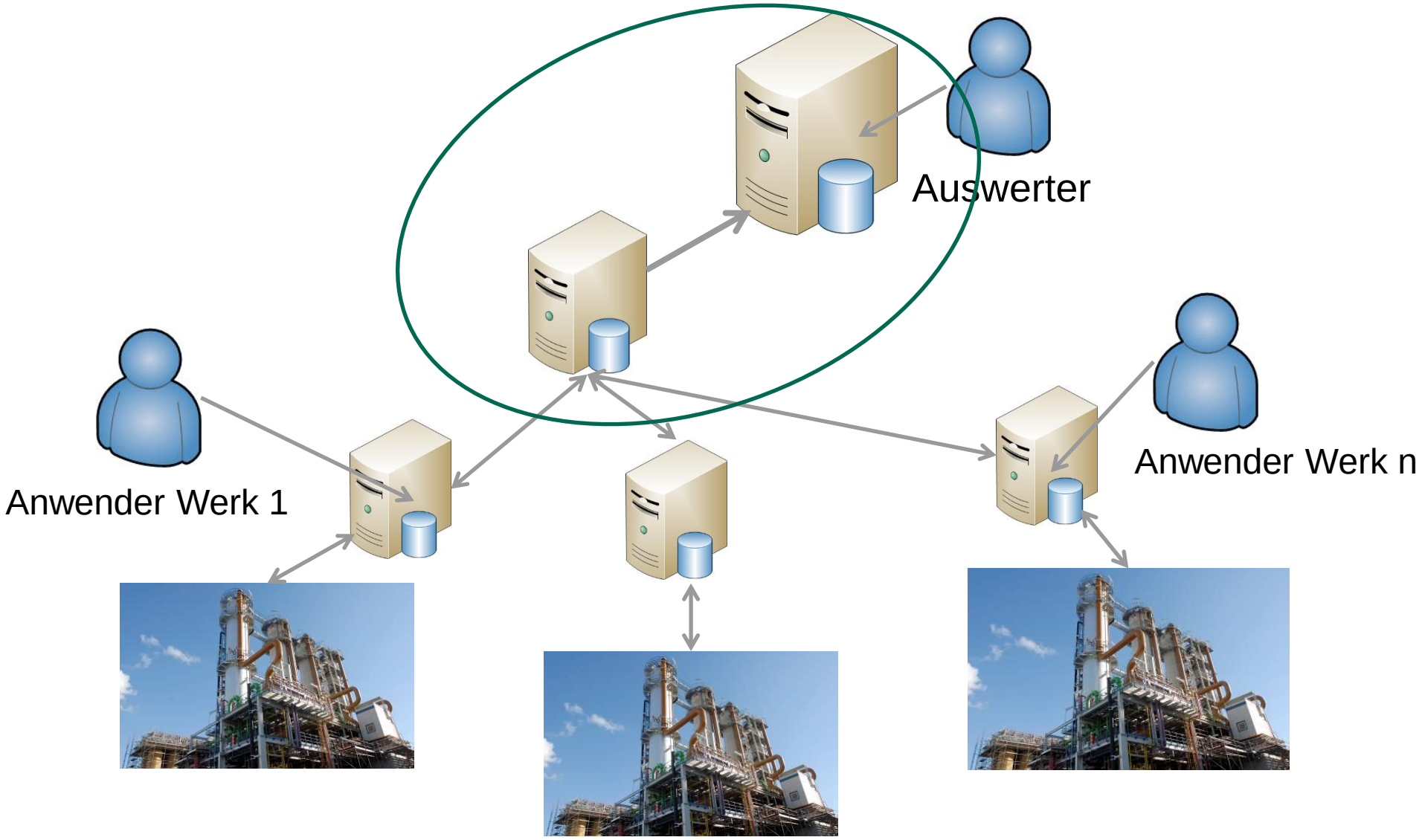
- Ausgangssituation: zentrale Datenbank
  - produktionssteuernd
  - Auswertungen
  - Datenbanknahe Applikationen
- Ziel:
  - Aufteilung der Datenbankstruktur mit Datenaustausch
  - Vorgabe: Reporting soll unverändert bleiben

➔ Datenbankreplikationslösung wird notwendig.

GEWACHSENE ANWENDUNGSSTRUKTUR



ZIELARCHITEKTUR



EINFÜHREN EINER NEUEN DATENBANKREPLIKATIONSLÖSUNG

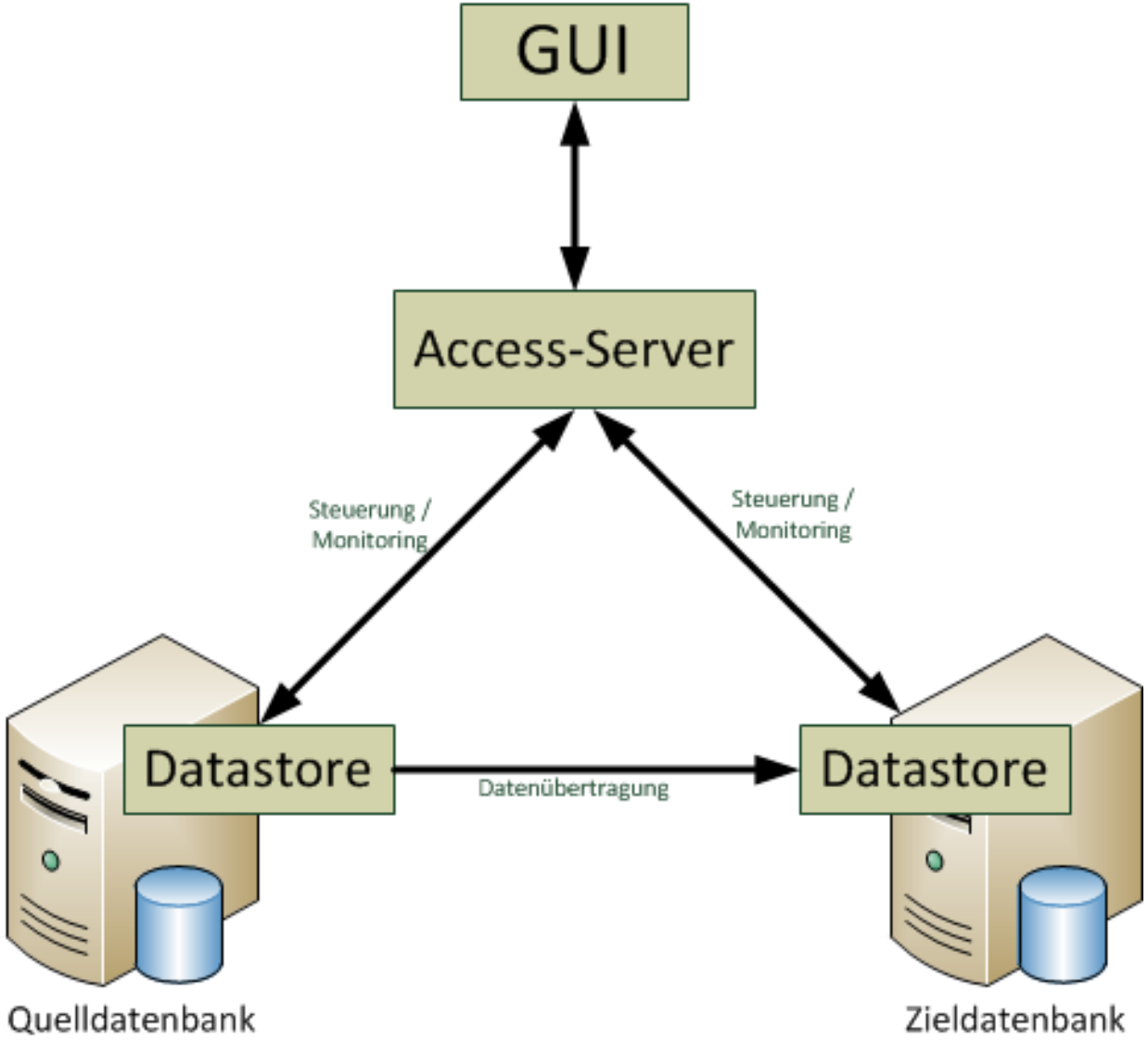
- Anforderungen
  - 1:1 Replikation
  - besondere Oracle Datentypen (XML Type, CLOB, …)
  - einfaches Starten nach DB-Recovery, bzw. nach Ausfall der Replikationslösung

UMSETZUNG

FUNKTIONSWEISE REPLIKATION

- Lesen der Redologdateien
- Applizieren der registrierten Änderungen am Ziel
- Datenbankuser mit SYSDBA Rolle wird zum Schreiben am Ziel benötigt.
- Möglichkeiten zur Datenmanipulation sollen im aktuellen Projekt nicht genutzt werden.

BASISARCHITEKTUR DER REPLIKATIONSLÖSUNG



LOGGING

- Enablen Supplemental Logging für replizierte Tabellen
  - `ALTER TABLE SCOTT.EMP ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;`
- Automatische Funktion des Tools nicht nutzbar.
- Vorsicht: Lock-Situation!
- Vorhaltezeit für Redologdateien: 7 Tage

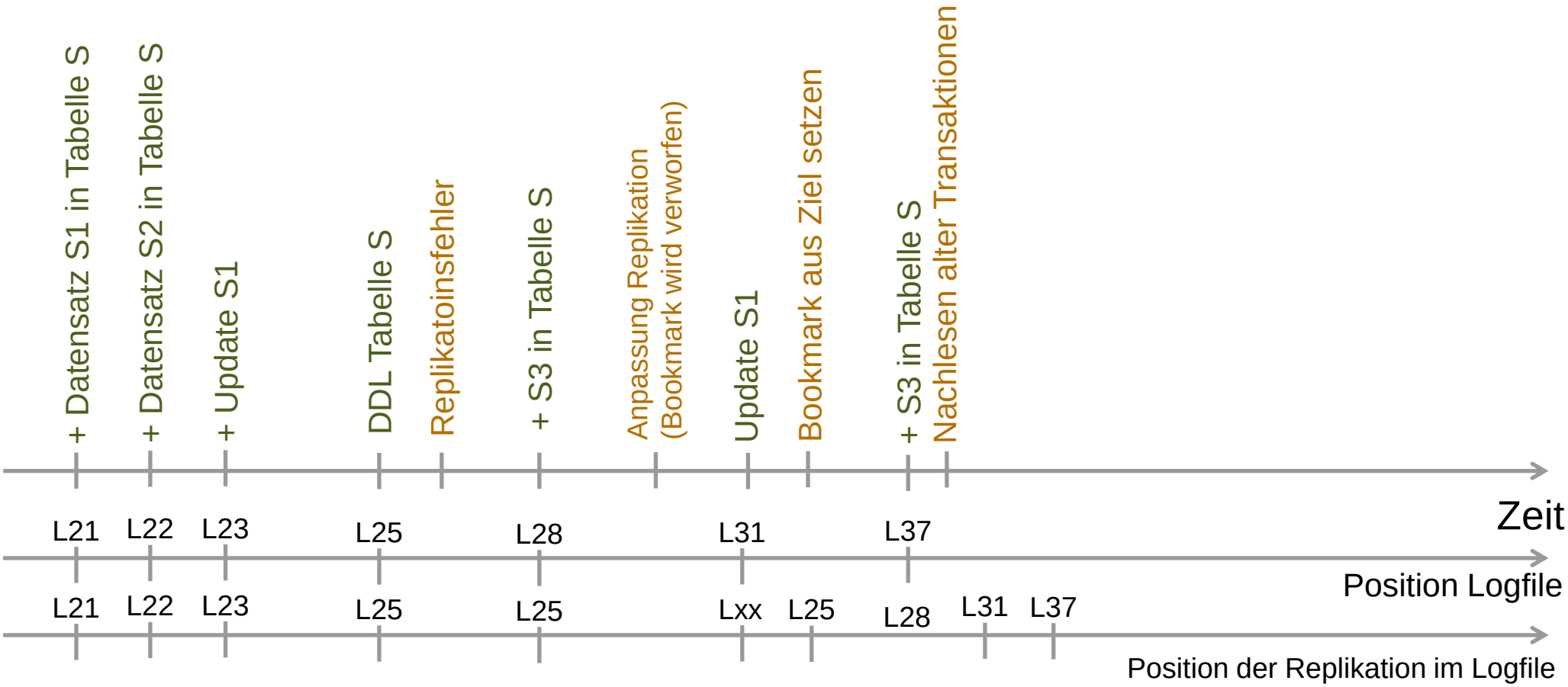
➔ stark angestiegener Speicherbedarf der Datenbank selbst

CHANGEMANAGEMENT

- Änderungen an Tabellen müssen koordiniert ausgeführt werden.
  - neue Spalten
  - umbenannte Spalten
  - Datentypänderungen
  - neue Tabellen
- Verschiedene Tätigkeiten werden von verschiedenen Abteilungen ausgeführt.

CHANGEMANAGEMENT

- Vorgehen bei DDL auf der Quelle ohne Downtime.

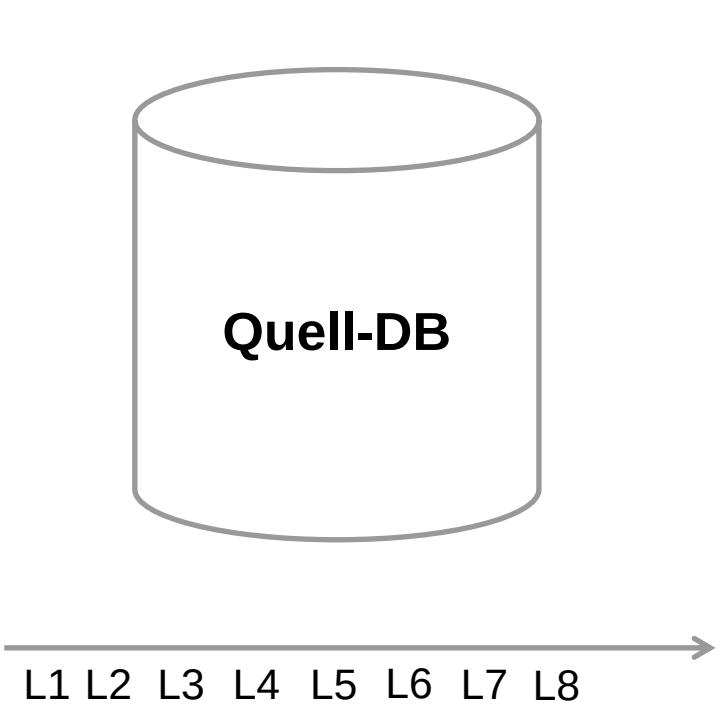


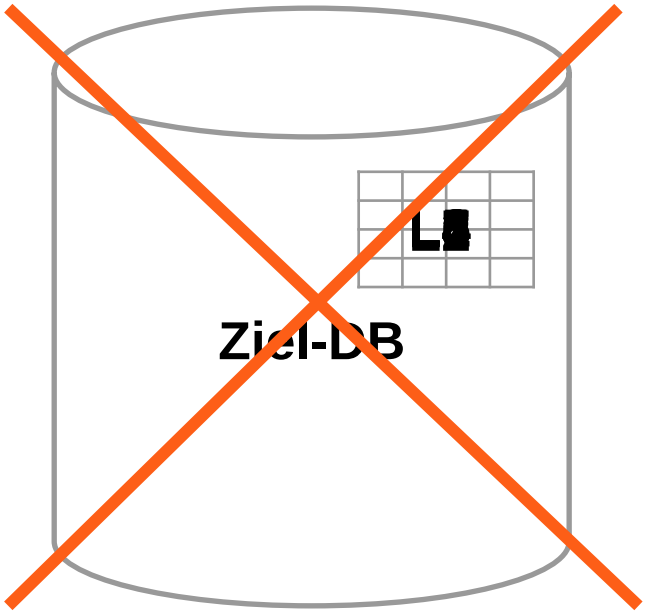
WIEDERANLAUF IM FEHLERFALL

- Aktuelle Logposition von der Quelle wird im Ziel gespeichert.
- Bei Recovery des Ziels (auch unvollständig), wird die zum Wiederherstellzeitpunkt gesetzte Bookmarkposition benutzt, um den Aufsatzpunkt in der Quelle zu finden.
- Das Starten der Replikation kann in diesem Fall ohne zusätzliche Einstellungen erfolgen.
- Dasselbe gilt, falls die Replikation ausfällt.

WIEDERANLAUF IM FEHLERFALL

- Ausfall des Ziels.





WIEDERANLAUF IM FEHLERFALL

- Ausfall der Quelle.
  - ➔ Ziel muss auf einen Zeitpunkt **VOR** dem Wiederherstellzeitpunkt der Quelle gesetzt werden.
  - Grund dafür: Statements werden in diesem Fall 1 : 1 repliziert.

ARCHIVIERUNG

- Filterung einzelner Datenbankuser
- Ziel: beinhaltet die gesamte Historie
- Quelle: historische Datensätze werden gelöscht

DESIGN DER REPLIKATIONSCONTAINER

- Innerhalb eines Replikationscontainers bleibt die Reihenfolge der Statements (meistens!) gewahrt.

  ➔ alle Tabellen, die per FK voneinander abhängen, müssen in einem Container definiert werden.

  ➔hohe Anzahl an Tabellen in Quelldatenbank (> 1100), 638 davon miteinander verbunden ➔ diese mussten in einen Replikationscontainer. Insgesamt wurden 6 Container erstellt.

  ➔Initialbefüllung des Ziels dauerte ca. 6 Stunden, dabei müssen die FKs deaktiviert sein.

- Problem: selbstreferenzierende Tabellen, die per ‚connect by' – Konstrukt gelöscht werden ➔ Lösung: deferred constraints.

PRÜFUNG DER REPLIKATION

- PL/SQL Package, welches die Daten aller replizierten Tabellen via SQL-Minus über Datenbanklink mit der Quelltabelle vergleicht.

FEHLERBEHANDLUNG

- Neuladen der Daten (Refresh)
  - Vorgehen: FK-Constraints auf die Tabelle müssen deaktiviert werden, da ein „truncate“ – Befehl abgesetzt wird. Indizes werden ebenfalls neu aufgebaut.
- Alternative: Setzen der Tabelle auf „aktiv“ (Bookmark setzen)

BESONDERHEITEN

- Materialized Views
  - bei Aktualisierung würden einzelne delete Statements repliziert ➔ extrem hohe Latenzzeit, alle MVs, welche im Ziel benötigt werden, werden auch nur noch im Ziel aufgebaut.
- Datenbanklinks von / zu anderen Systemen wurden ebenso meist auf das Replikationsziel umgelegt.

VERFÜGBARKEIT

- Verfügbarkeit der Replikation: 5 * 8
- Verfügbarkeit der Zieldatenbank: 7 * 24
- Latenzzeit darf maximal 2 Stunden betragen
- Durchschnittliche Latenzzeit < 1 min.
- bisher erst 1 mal Überschreitung der max. Latenzzeit.
- Besonderheit: bisherige Archivlösung setzt auf Partitionierung bzgl. Archivflag. Beim Archivieren von großen Datenmengen gibt es Auswirkungen auf die Latenzzeit.

FAZIT

FAZIT

- Es gab trotz „einfacher“ 1:1 Replikation einige Klippen zu umschiffen.
- Projektlaufzeit konnte trotzdem eingehalten werden.
- Wertvolle Erfahrungen konnten mit der Replikationslösung für den späteren sehr zentralen Einsatz im globalen Lösungsverbund gesammelt werden.
- Verfügbarkeitsanforderungen an die Replikation werden zukünftig höher sein.

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT.

WACKER

CREATING TOMORROW'S SOLUTIONS