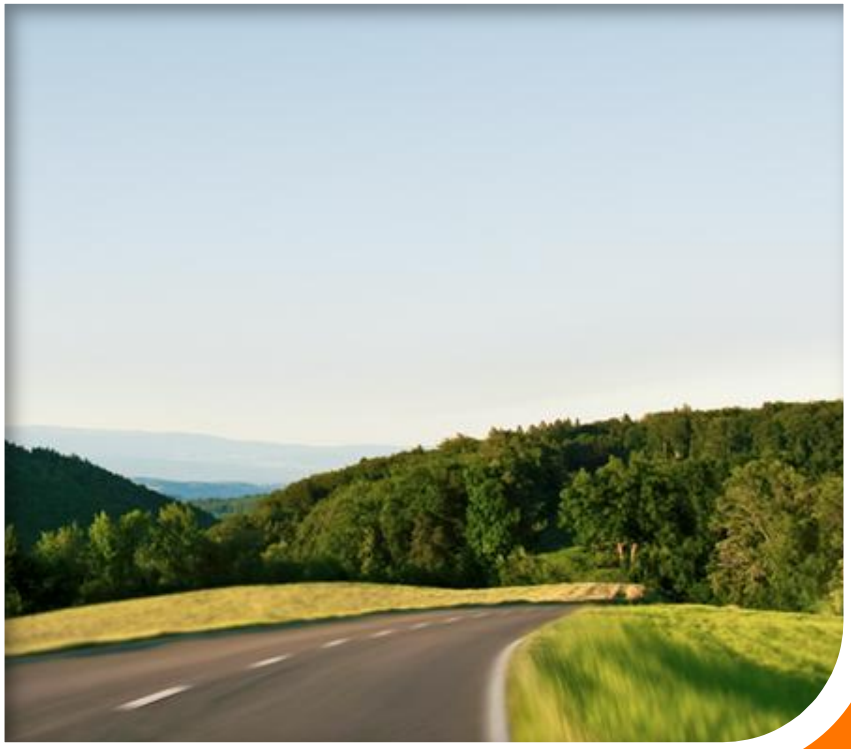
# True Production Databases for Impatient Developers

AUTO SCOUT 24

DOAG Konferenz

20. – 22.11.2012, Nürnberg

www.autoscout24.de



# Inhaltsverzeichnis

# AutoScout24 – HIER IST ALLES AUTO.

Rund 2 Millionen Fahrzeugangebote

15,6 Millionen Nutzer jährlich in Deutschland

Europaweit größter Online-Automarkt

Über 100.000 gebrauchte Nutzfahrzeuge

Über 10 Millionen Menschen europaweit nutzen AutoScout24 jeden Monat

Rund 99.000 Motorräder

Mehr als 300 Millionen virtuelle Fahrzeug-besichtigungen pro Monat

Rund 40.000 Händlerkunden europaweit

# Das Problem

## Stufe 1

Entwicklungsumgebung und Produktionsumgebung laufen mit der Zeit auseinander

- Auf der Entwicklung bricht die Anwendung. Nach langer Suche stellt sich heraus, dass es an einer Tabelle liegt, die es in der Produktion nicht gibt.
- Das Skript lief schnell auf der Entwicklungsdatenbank, warum braucht es auf der Produktion so lange? Weil die Tabelle dort 1000mal mehr Einträge hat.
- Wir haben ein Problem auf Live, können es aber auf der Testumgebung nicht nachstellen. Diese Datenkombination kommt da nicht vor.

# Die ad-hoc Lösung

## Stufe 2

Alle sind sich einig, dass die Livedatenbank in die Entwicklungs-umgebung transferiert werden muss

- Aber nicht in dieser Woche! Dann ist die Entwicklungs-datenbank einen ganzen Tag lang nicht verfügbar, und wir können nicht arbeiten. Außerdem ist danach wieder unsere ganze Arbeit weg …

## Stufe 3?

# Livenahe Umgebungen für die Entwickler bereitstellen

## Die Aufgabe

- Die Entwicklungs- und Testumgebungen sollen so produktionsnah wie möglich sein
- Das betrifft Struktur und Inhalt
- Aber wir sollen damit nicht Arbeit unterbrechen, die Entwicklung zurückwerfen, oder die Tests stören.

## Nicht die Aufgabe

- Der Code auf den Webservern

---

Wie heißt diese Aufgabe? Ich nenne sie in diesem Vortrag „Refresh“.

---



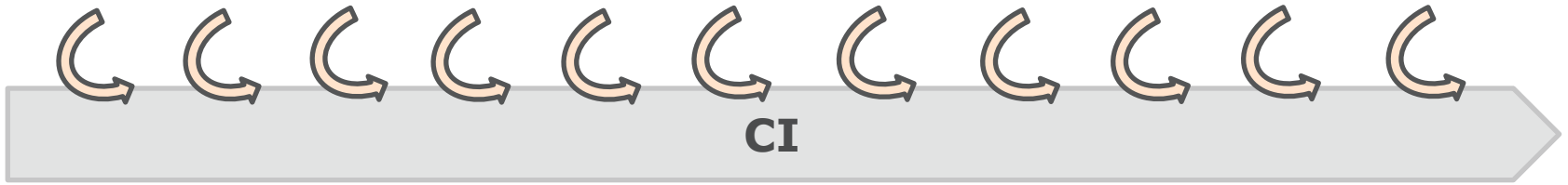
AUTO SCOUT 24

# Refresh, ohne die Entwicklung zu stören

Wie machen wir das?

# Continuous Integration bei Autoscout24

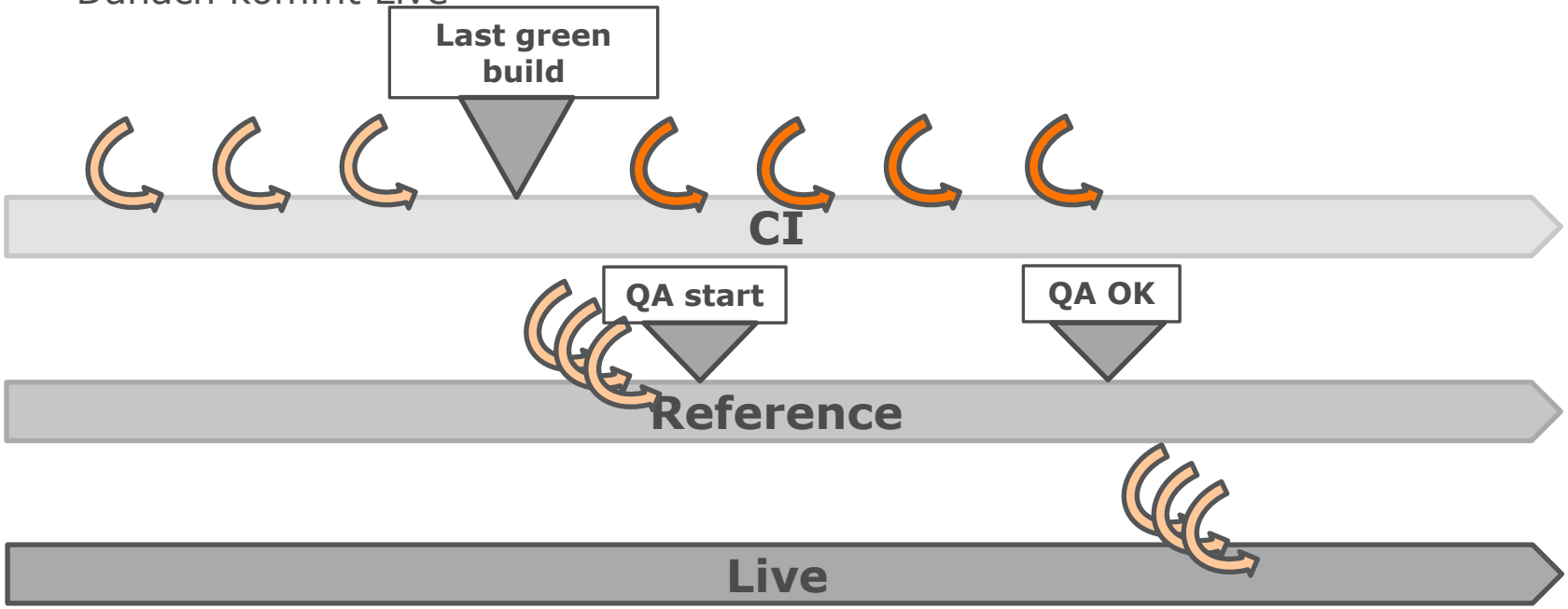
- Bei Autoscout24 wird mit „Continuous Integration“ entwickelt.
- Für die Datenbankskripte bedeutet das bei uns, dass sie gegen die Entwicklungsumgebung laufen, sobald sie im Codeverwaltungssystem als fertig committed werden.
- Ein selbstgeschriebenes Tool führt sie per SQL*plus auf den Datenbanken aus. Dabei tragen sie sich in eine Changelogtabelle ein. Ein Constraint verhindert, dass ein Skript mehrmals läuft.
- Die Datenbank wird nicht versioniert. Der Code auch nicht.
- Schnelle Tests zeigen, ob die Applikation noch funktioniert („Green Build“)




**Es kommen ein paar Skripte pro Tag hinzu. Das kann alles sein, von neuen Datenbankobjekten bis zu Datenmigrationen.**

# Der Entwicklungszyklus bei Autoscout24

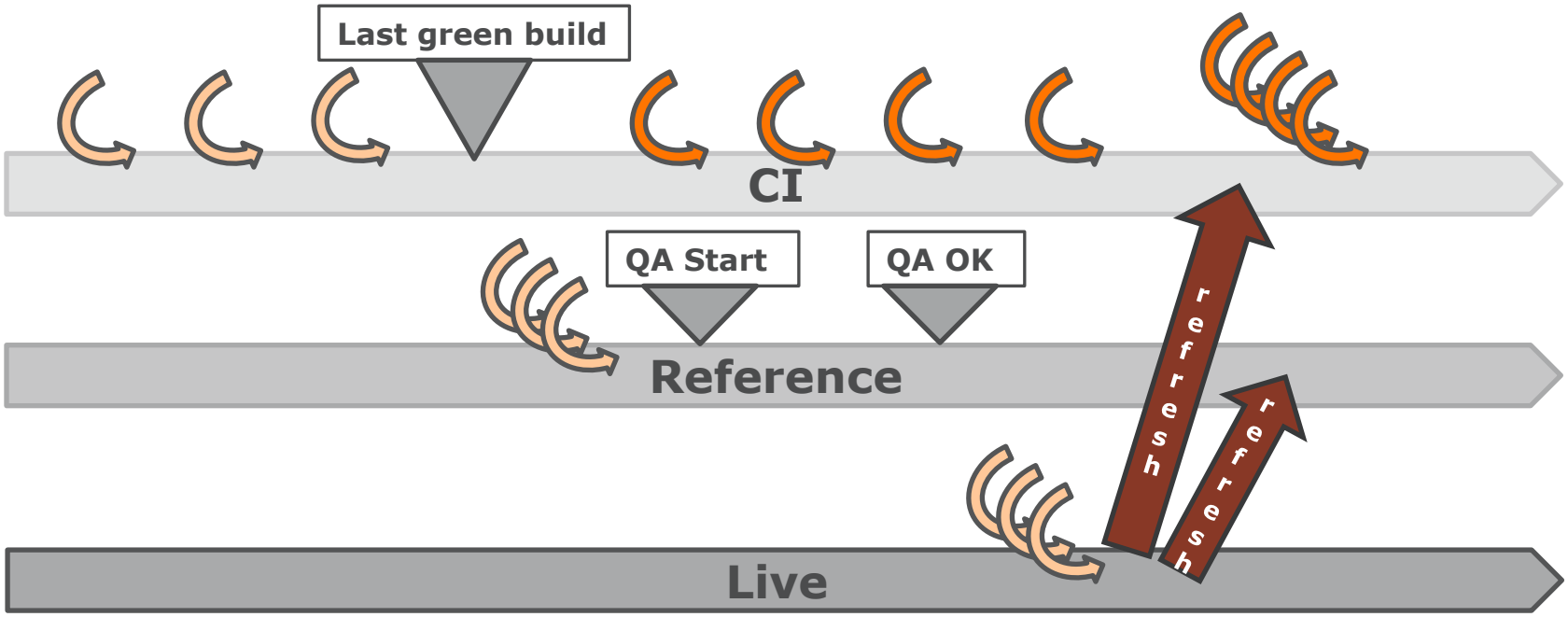
- „CI“ ist die erste Entwicklungsumgebung
- „Stable“ ist wie CI, es laufen aber mehrstündige Tests. In der Zeit werden keine Skripte eingespielt. Diese Phase ähnelt der folgenden und wird deshalb in den Bildern ausgelassen.
- „Referenz“: Hier wird ein Stand von der QA freigegeben
- Danach kommt Live



# Wie wir den aktuellen Stand herstellen

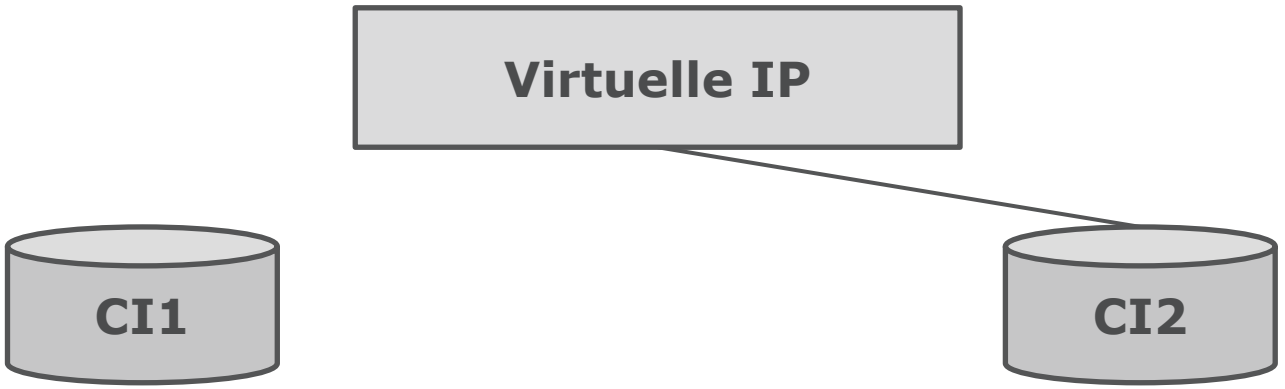
Wenn ein Skriptstand auf Live ist, ist er für CI und Stable bereits veraltet. Um diese Lücke nach einem Refresh zu füllen, laufen anschließend alle Skripte, die in der Zwischenzeit comitted wurden. Erst dann wird die Umgebung freigegeben.

Betankung und Release gehören also zusammen. Sie werden aus demselben Tool gesteuert (Team City).



# Wie wir die Downtime kurz halten

Die Betankung mit neuen Daten braucht länger als ein paar Minuten. Deshalb machen wir das im Hintergrund.
Jede Datenumgebung gibt es zweimal. Die inaktive Umgebung wird mit Livedaten betankt, während die andere benutzt wird.




Betankung mit frischen Daten,
Skripte im Hintergrund,
Wartungsarbeiten

Applikation,
Entwicklung, Tests

# Nach außen merkt man fast nichts

Die Entwicklungsumgebung droht zu veralten, oder ein Skript hat sie zerstört? Kein Problem. In fünf Minuten haben die Entwickler eine neue, livenahe Datenlandschaft mit dem neuesten Skriptstand.

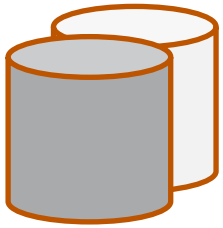
Nur die Tester haben leider ein Problem: Ihre Testdaten sind weg. Deshalb können sie die Referenzumgebung reservieren.

Die Betankung mit frischen Daten braucht eine Stunde. Der Switch dauert etwa 5 Minuten.

# Die Daten sind nicht nur in einer Datenbank

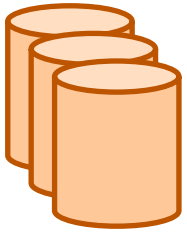
# Unsere Datenlager

Die Kerndatenbank: Eine Oracle Datenbank mit Lesefarm (vergl. Vortrag „Active-DataGuard bei Autoscout24: eine Lesefarm im Praxiseinsatz“)



110 GB

Drei weitere Oracle DBs

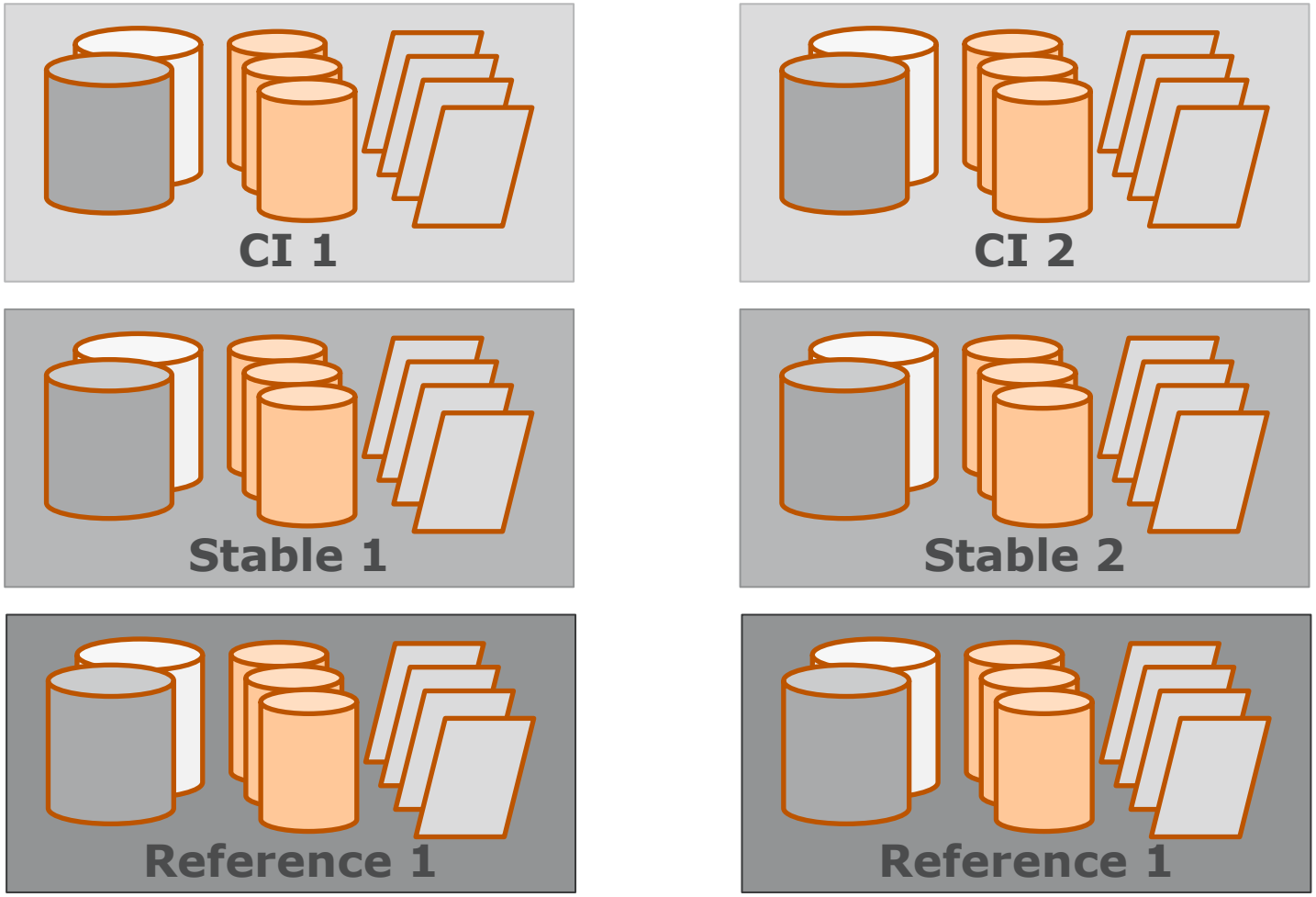


100 GB + 70 GB + 17 GB

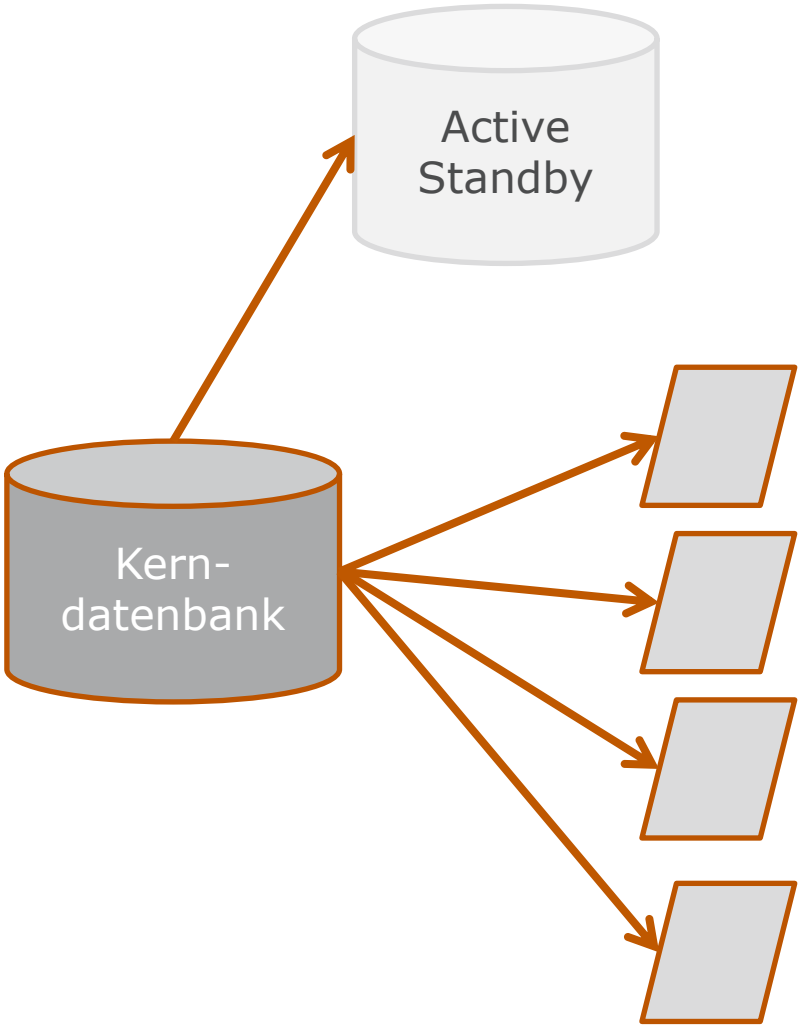
4 Suchindexe: Drei Endecasysteme und ein Key-Value-Store.

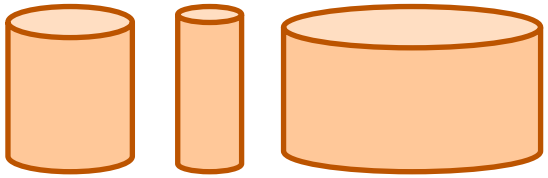
5 GB + 4 GB + 3 GB + 4 GB

# Die Datenlager in drei doppelten Umgebungen



# Die Datenstände müssen zusammenpassen



Drei unabhängige Datenbanken

Die Indexsysteme ziehen Daten von der Kerndatenbank

# Unsere Skripte schnüren ein Datenpaket mit Livedaten

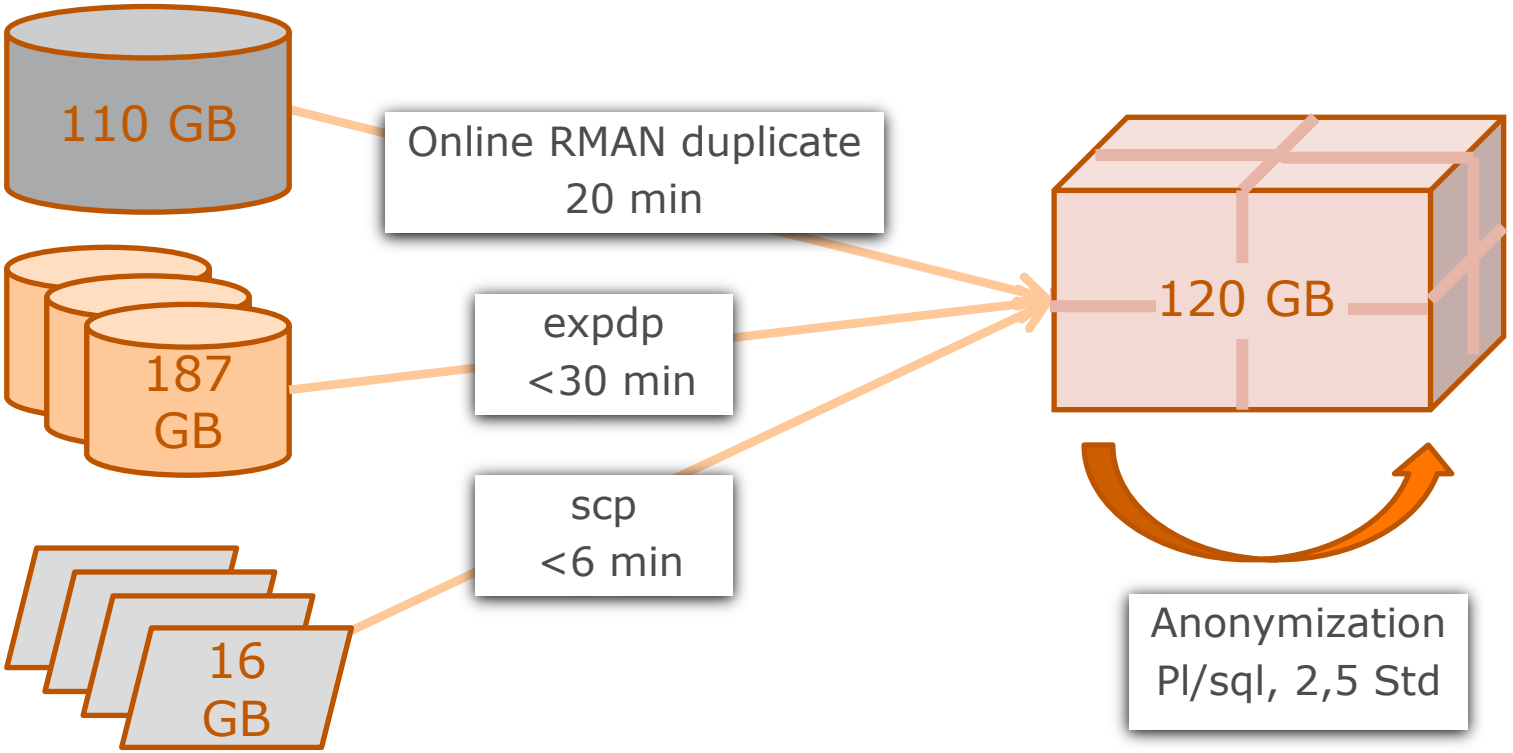
## Um aus den Produktionsdaten einen konsistenten Stand zu bekommen:

1. Stoppen und kopieren wir die Indexsysteme
2. Setzen einen Zeitstempel
3. Starten parallel drei Data Pump Exports der Randdatenbanken, mit Konsistenz auf den Zeitstempel. Einige Tabellen werden exkludiert, einige mit Where-Klausel verkleinert.
4. Ebenfalls parallel wird mit Active RMAN duplicate eine Kopie der Kerndatenbank erzeugt. Nur hier gibt es personenbezogene Daten. Sie werden mit PL/SQL-Prozeduren anonymisiert. Anschließend stoppen die Skripte die Datenbank.

Das Ergebnis ist ein Verzeichnis auf einem NFS-Server mit Indexkopien, Data Pump Exports und offline Datenfiles.

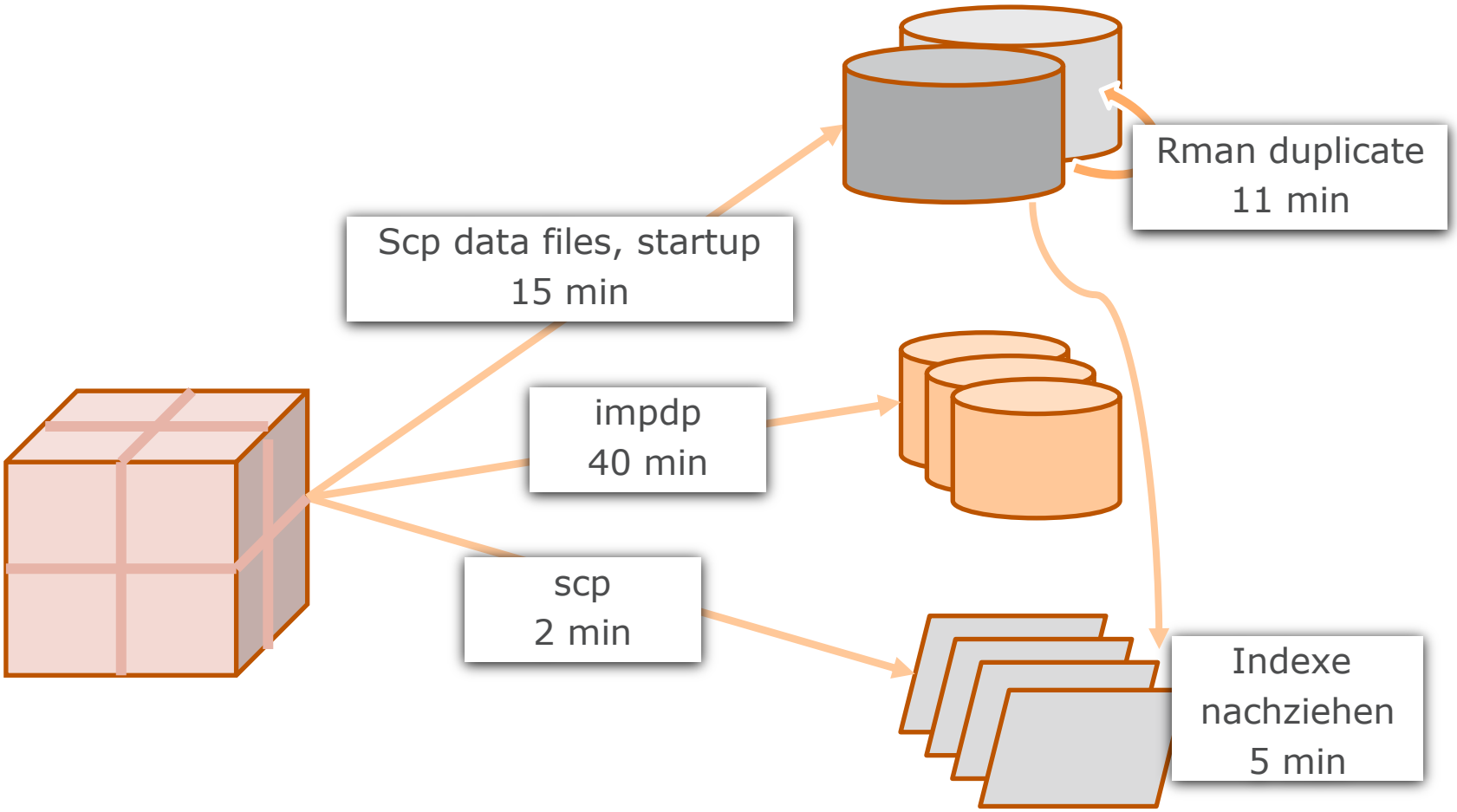
Verworfene Techniken: SAN Snapshot, Flashback and Roll Forward, Snapshot Standby

# Wir packen ein Paket aus den Livedaten …




Keine spürbare Last auf den Live-Systemen

# … und packen es in der Entwicklungsumgebung wieder aus.



# Unsere Skripte betanken damit die Entwicklungsumgebung

## Um daraus eine Entwicklungsumgebung zu bekommen:

1. Finden wir die inaktive Umgebung
2. Kopieren wir die Indexe auf die Endeca Server
3. Starten wir den Master der Kerndatenbank aus offline Datafiles, erzeugen die Data Guard Broker Konfiguration und die Standby-Datenbank (DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE NOFILENAMECHECK)
4. Füllen wir die anderen drei Oracledatenbanken parallel mit Data Pump Import
5. Ziehen wir die Indexe nach
6. Rufen Customizing-Scripte auf

# Automatisierung und Steuerung

# Unser Refreshprozess hat drei Schritte

1. Ein Paket aus Produktionsdaten packen
2. Mit diesem Paket eine inaktive Entwicklungsumgebung betanken
3. Diese Umgebung aktiv schalten (Switch)

Das machen 70 Shell- und SQL-Skripte mit über 10.000 Zeilen unter Einsatz verschiedenster Techniken

# Wir drücken nur noch auf den Knopf – oder nicht einmal das

**Switch und Refresh werden nach Regeln gestartet, oder manuell**

- Die Regeln verhindern, dass veraltete Umgebungen benutzt werden: Refresh nach jedem Switch, Switch vor jedem Release u.s.w.
- Und die Regeln erzwingen, dass der Prozess regelmäßig und damit stabil läuft: Wöchentlich mindestens einmal

- Außerdem gibt es Knöpfe in „Team City“, auf die Datenbankadministratoren und die Release Engineers drücken können.

# Fazit

- Die Betankung läuft vollautomatisiert, regelmäßig und stabil seit einem halben Jahr. Wir sind in der täglichen Arbeit entlastet
- Wir haben keine Probleme mehr durch unterschiedliche Stände der Umgebungen
- Die Belastung der Produktion durch RMAN duplicate und Data Pump ist nicht spürbar
- Die Unterbrechung durch den Switch ist sehr kurz, und die Akzeptanz durch die Entwicklung ist entsprechend hoch. Aber der Switch stört Tests, die Daten anlegen
- Wir pflegen eine Menge Systeme, und jedes neue System in Live zieht sechs weiter nach sich
- Die Hoffnung, die ganze Sache einfach zu halten, hat sich nicht erfüllt

Der Aufwand hat sich gelohnt, und es geht weiter

www.autoscout24.de

# Vielen Dank für Ihre Aufmerksamkeit!

**Kontakt:**
AutoScout24 GmbH
Dingolfinger Straße 1-15
81673 München

Suny Kim
Fon +49 89 444 56-1507
suny.kim@autoscout24.com