Lost Write und Große Indexe (Re)build

MIRACLE

Das versprochene Abstract

- Lost Write bedeutet verlorene Daten und wahrscheinlich Korruption. Der erste Teil des Vortrags erklärt das Wesen von Lost Write und wie Data Guard hilft es zu erkennen. Die zweite Hälfte beschreibt den nachfolgenden Wiederaufbau, TBs große Indizien mit maximum Verfügbarkeit, das heißt fast ohne Unterbrechung der Benutzer.

Ich ?!

- **Michael Möller** (aka. M-square oder M²)
- Im EDV/IT Bereich seit 50 Jahren (habe früh angefangen)
- Oracle DBA bei Oracle und Miracle mehr als 20 Jahre.
- Oracle Certified Master, OakTable Mitglied
- Obwohl ich viel lieber ein Steampunk Tesla-Spulen Kanonier auf einem Dampf-Luftschiff sein möchte ...

MIRACLE

Inhaltsverzeichnis

Zwei Vorträge

- Lost Write (Schmerzhaft)
  - Wie entsteht es
  - Was passiert in Oracle
  - ”Lost Write Detection”
  - Ist Abhilfe möglich?
- Großer Index (Re)Build
  - Wie Groß ist Groß?
  - Allgemeiner Rebuild Online nicht möglich
  - Materialized View anders benützt

M2@Miracle.dk DOAG 2017

Lost Write

Ein unangenehmes Erlebnis

MIRACLE

Eine lange Geschichte – Was (aber nicht wo)

- Das System ist groß genug.
  - Viele Benützer, komplizierte Applikation
  - Terrabyte
  - Rechner (mit VmWare), SAN
  - DataGuard
- Est ist wichtig
  - 24x7 – wie zB. Flughafen, Krankenhaus, Bankkarten Zahlung

Eine Lange Geschichte – mehrere Fehler

- Kunde har ORA-1578 (Corrupted Block; read DBA↔ expected DBA mismatch)
  - Wird behoben mit recovery
- Standby kriegt ORA-00600 [3020] (Redo ↔ Block mismatch)
  - Vielleicht wegen der recovery?
- Versuchen mehrmals Standby Reinstantiation, von Backups (von vorhin)
  - (Unangenehm: Backups sind nicht mehr brauchbar ?!)
  - Reinstantiation mit neuer Kopie gelingt (Die schlechtest mögliche Wahl ! ☹)
- Benützer sehen ORA-08102 (Index ↔ Rowdata mismatch)
  - „Bloß“ Index rebuild machen (was nicht geht)

Eine Lange Geschichte – Hindernisse

- Applikation läuft – und bleibt live (fast) die ganze Zeit.
- Switchover / Experimente
- Untersuchungen waren nicht wiederholbar
- Index rebuild „nicht möglich“ – die sind zu groß.
  - Versuche mit nicht-super-großem scheitern manchmal

Eine Lange Geschichte – Indizien

- Falsche Theorie (SB Abbruch, Restore Scheitern): Fehler in Redo Dateien
- O/S Untersuchung findet Oracle Blocke am Anfang der Plattenspeicher
  - Analyse der Corrupted Blocks zeigen die sind OK
- Die korrupte Blocks sind auch OK – bloß am verkehrten Platz
- ?? .. !!
- Ah ha – „Lost Write“ oder „Misplaced Write“

Lost Write – was ist das?

- Ein Datenblock ist geändert
- Oracle bittet die O/S zum Plattenspeicher zu schreiben
- Der Block wird **nicht** geschrieben (oder im verkehrten Platz)
- Die O/S meldet zur Oracle „Alles Gut”
- Datenverlust, und Recovery wird unmöglich

Wie/Wo ensteht es

- Hardware – Memory, Rechner, Verbindungen
- Software – OS, Filesystem, Virtualisierung
- Dazu Caches, Load share und ”striping”.
- Es wird zillionen Schreibbefehle gegeben. Es gibt viele ”parity” und ”checksum” Kontrollen auf allen Ebenen –weil eben Fehler entstehen.
  - Fehler werden normalerweise automatisch behoben oder gemeldet
- Irgendwo entsteht ein zwei-/dreifach Fehler
  - Wird nicht bei dem Schuldingen entdeckt, aber in Oracle.

Was passiert in Oracle

- Nichts besonderes.
  - Wie immer, schreibt Oracle einen Redo Record, mit den Vor/Nach Beschreibung und SCN
  - Wie immer ist der Block mit neuer SCN im cache, der später in einem ”batch” (Scatter write) geschrieben wird
  - Der WRITE geht immer gut (obwohl hier der Fehler entsteht)
  - Wenn man die Daten bloß lesen will, geht es auch gut
    - Man sieht bloß die **alten** Daten
  - Man kann auch Rows im Block weiter/wieder updaten
    - Hier entsteht unwiderufbar **Lost Data**

Wie entdeckt Oracle es ? (Standard)

- Vor einem weiteren Write auf dem Block : nicht entdeckt
  - Aber Datenverlust!
- Nach weitere Writes : auch kein Fehler

Aber

- Beim Recovery wenn der Redo der zweite Write gelesen wird:
  Redo Record und Daten Block stimmen nicht mehr – **stuck recovery**
- Das gleiche wenn man DataGuard läuft
  - Ist ja eine Recovery auf dem Standby

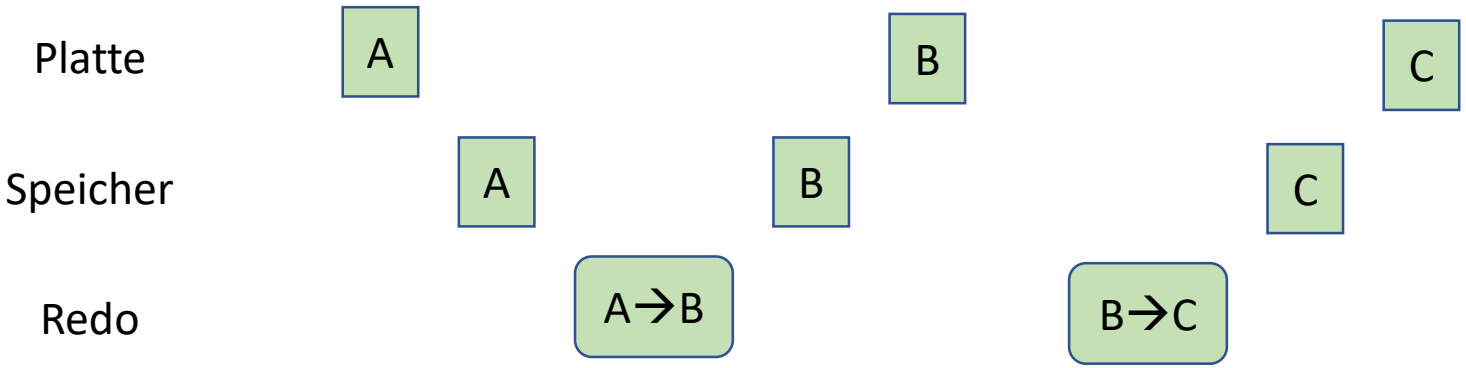
Redo Inhalt

- Redo enthält für jede Änderung
  - Daten vor der Änderung
  - Daten nach der Änderung
  - Block Adresse, Zeitpunkt, andre Referenzen
  - Andre Daten von der Tabelle wenn man Supplemental Loggning benützt
- Redo hat auch andere als nur Dataenblock Änderungen
  - Commit Record
  - Manche DBWR Operationen (für Fast Crash Recovery)
  - Checkpoint Punkte
  - "LostWrite records" – für den Standby zur Entdeckung desgleichen
  - …

…(redo Inhalt)

- Redo Vector ist eine „Automische Änderung“ die sich öfter über mehrere Blöcke streckt, zB. TabellenBlock und UndoBlock
  - Ein Vector hat mehrere RedoRecords. Ein Record ist bloß für einen Block
  - Ein Vector/Record für Nicht-Block-Änderungen har andre Strukturen
- In dieser Erklärung ist Redo und Archive das Gleiche

Normaler Ablauf



Anfang; A nach B

Block zum Speicher

Blockänderung von A nach B

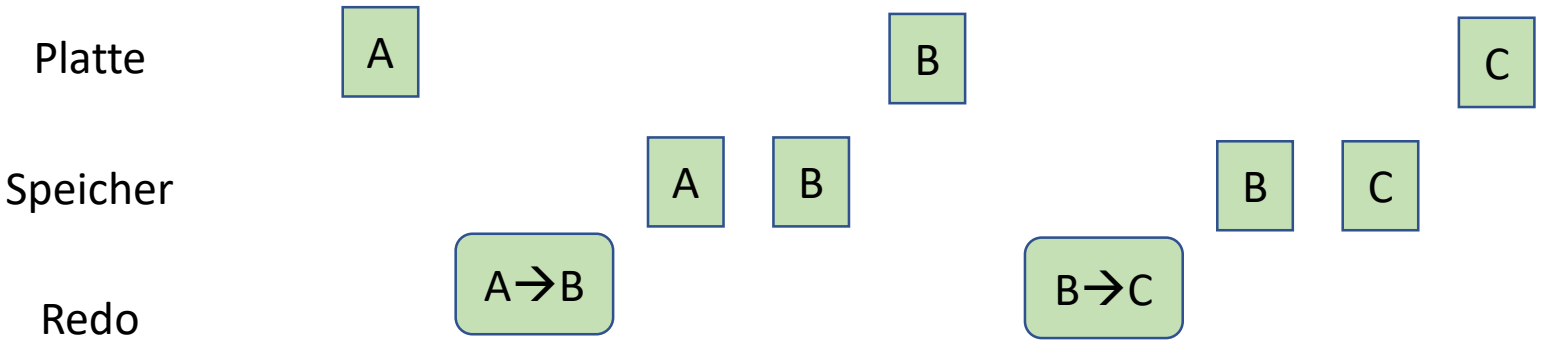
Blockänderung ausführen

Block schreiben (viel später)

Wiederholen wenn Änderung nach C,
B einlesen, B→C Vektor, C schreiben

MIRACLE

Normaler Standby oder Recovery



Redo verlangt A zu B

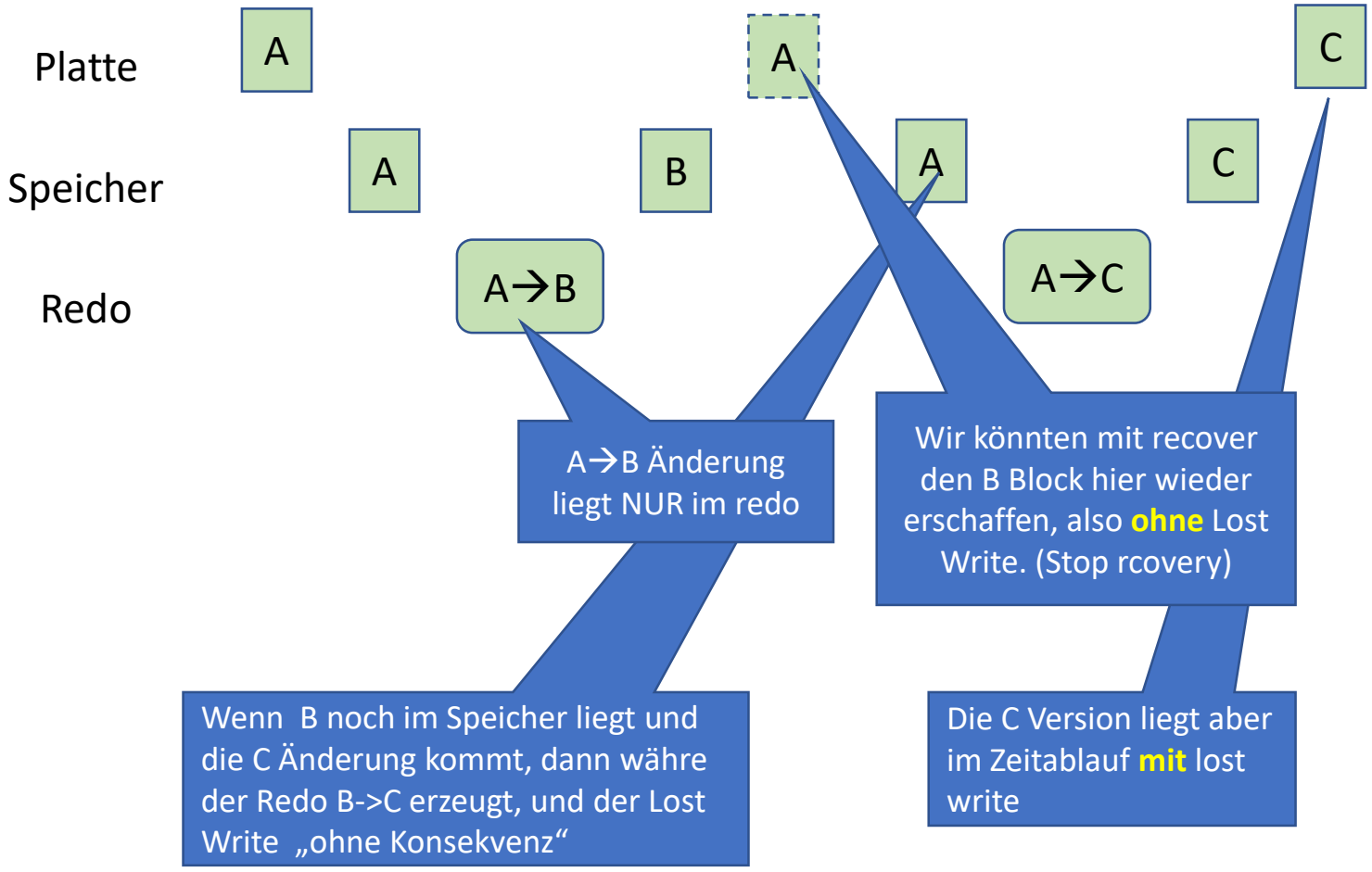
Block lesen

Kontrollieren ob es A ist, dann ändern
Block schreiben (etwas später)

Wiederholen wenn Änderung nach C,
B einlesen, B→C Kontrolle, C schreiben

MIRACLE

Eine ”Lost Write”, eine gelungene Schreibung



Anfange Erste, ? zu B

Block zum Speicher

Blockänderung von A nach B

Blockänderung ausführen

Block Schreibung verschwindet

Anfang Zweite, ? zu C

Block zum Speicher

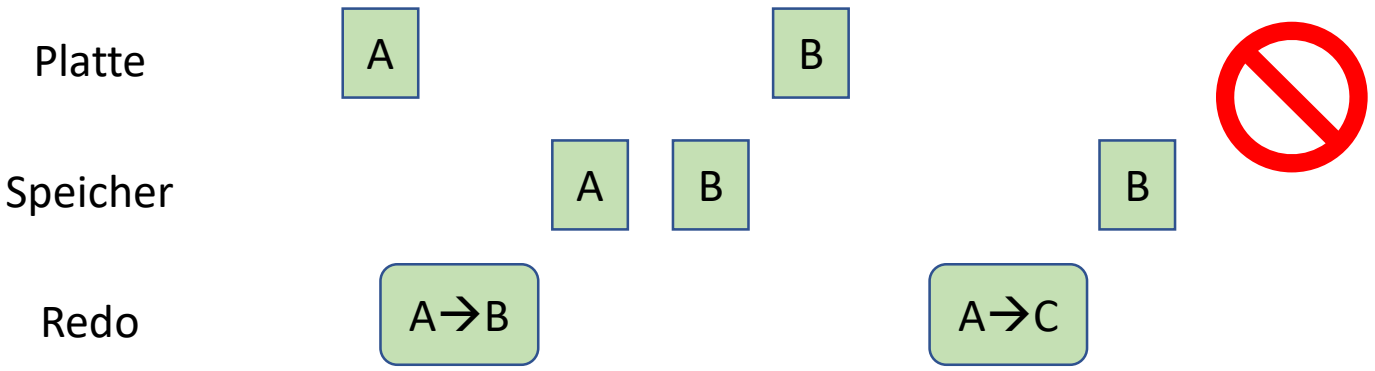
Blockändrung von A nach C

Blockänderung ausführen

Block schreiben (viel später)

MIRACLE

Standby oder Recovery stürzt ab



Redo verlangt A zu B

Block lesen

Kontrollieren ob es A ist, dann ändern

Block schreiben (etwas später)

Redo verlangt A zu C

Block lesen

ORA 600 [3020]; A erwartet

MIRACLE

Wie entdeckt Oracle es ? (Lost Write Detect)

- **`DB_LOST_WRITE_PROTECT=TYPICAL`** (NONE, FULL)
- Schreibt für jeden READ im Redo was gelesen wird (DBA,SCN)
- Also: Nichts wird entdeckt (!)

- Bei einem Recover wird es bei dem ersten Read nach einem Lost Write entdeckt.
  - (Ohne DB_LOST_WRITE_PROTECT, erst bei der zweiten write)
- Also: Laufend recovery durchführen … zB. im Standby

Data Guard Lost Write Detection

- Der fatale Fehler, und die Detection, sind davon abhängig, wie und wie oft der Block mit dem einem LostWrite nachfolgend benützt wird.
- Hauptsächlich, wenn der Primary einen Block liest, liest der Standby den gleichen Block bei sich, und vergleicht ihn mit dem im Primary.
  - SCN/DBA, nicht ganze Blöcke.
- Mehr IO, mehr Redo

Was kann man tun

- Wenn es zu spät ist, ist es zu spät.
  - Nach dem zweiten Write ist es zu spät ☹
- Wenn man es **vor** der zweiten Schreibung entdeckt, kann man Blockrecovery auf den Block (evtl. vom Standby) machen.
- `DB_LOST_WRITE_PROTECT` – mit den Standby – signalisiert es beim ersten Read nach dem Fehler.
- Mit Redo (=Recovery) die Block Änderung wiederholen

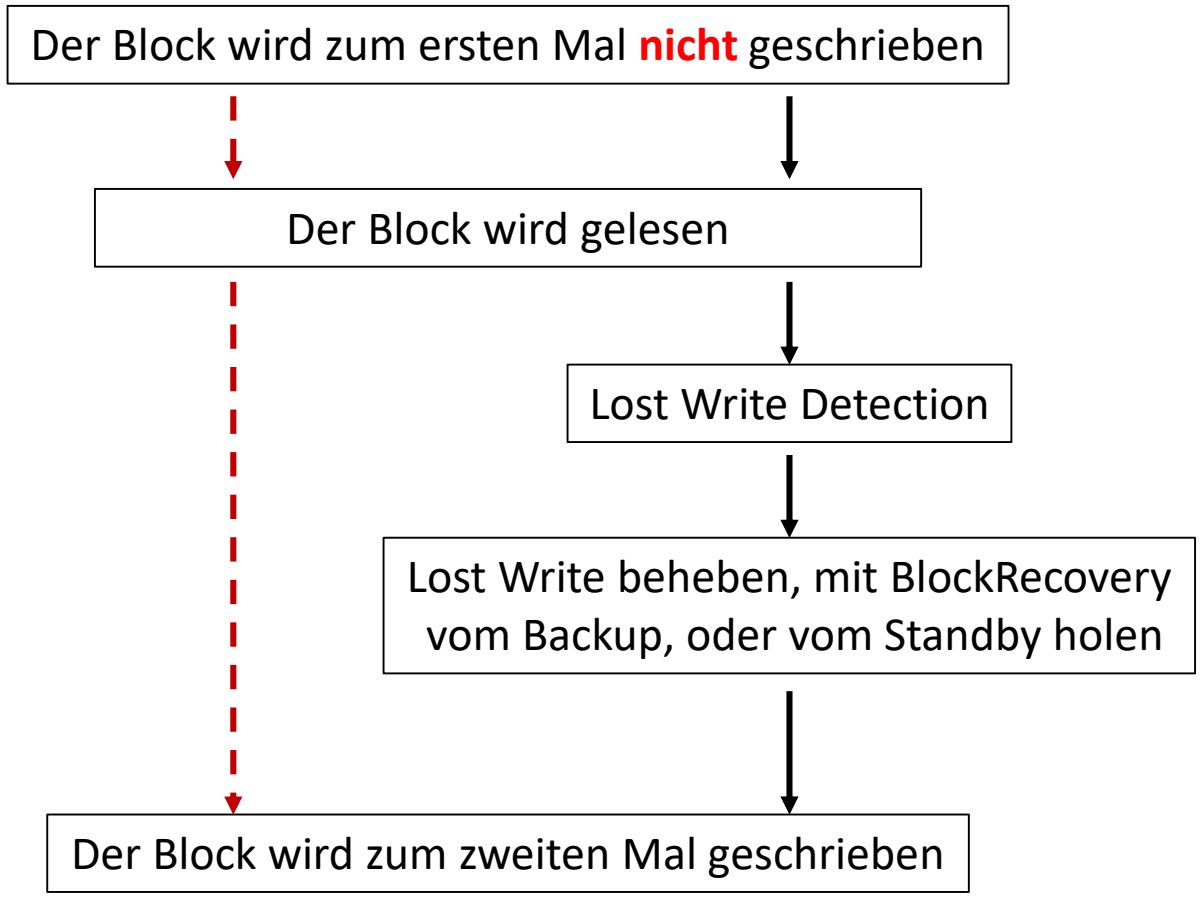
Block Repair (≡Restore/Recover)

- Guten Block vom Backup holen (EE, block recovery)
- Guten Block vom Standby holen (Active DG)
- **Active** Data Guard Lost Write Detection **and Repair**
  - Detection ist „mit“, Repair
  - War aber nicht aktiv in diesem Fall.

Lost Write kontra Corrupted

- In diesen Fall ging der Write im verkehrten Platz
  - Ein Block wird Corrupted (wenn er in eine Datei schreibt)
  - Und der Orginal Block ist Lost Write
- Corruptionen werden beim ersten Read gemeldet
  - „Schnell“ behoben mit Recovery, evt BlockLevelRecovery

Also …



MIRACLE

„Demo“

`http://www.oracle.com/technetwork/database/availability/lostwriteprotection-1867677.exelostwriteprotection-1867677.exe`

Zeigt den LostWrite Detection auf der Standby

`http://www.oracle.com/technetwork/database/features/availability/activedataguardautoblockrepair-180595.exe`

Der demo zeigt Block Corruption, nicht Lost Write

`standby-autoblockrepair-1881372.exe`

Zeigt das gleiche, aber auf der Standby

MIRACLE

Wie geht es beim Kunden?

- Wir wissen einige Tabellen Blöcke sind verloren
  - Hoffnungslose Korruption mit Repair/CTAS entfernt
  - Unbekannter Daten Verlust (innerhalb eines Zeit Fensters)
- Der Lost Write ist nicht mehr geschehen
  - Aber die Hardware ist jetzt eine andere

- Und viele/große Indexes die erneuert werden müssen.

Großer Index (Re)build

Neue Indexe gegen Corrupt/LostWrite Fehler

- Die Corrupt/LostWrite Fehler im Index mit neuen Indexen beheben.
- Die Indexe waren „groß“ (z.B. ½Tb Tabelle, 4 x 50Gb Index).
- Rebuild war nicht möglich, es mußte Drop&Create Index sein
  - Bug 22241601
- „Online“ blockiert nicht DML,
  aber die Applikation geht nicht ohne Indexe.
- Dauer >1 Tag (insgesamt)
- Mit Test Datenbank versucht und geübt.

Gleichzeitig einige andere Probleme (Nebengeschichte)

- Wegen vielen „Migrated Rows“ war Create Index (besonders parallel) langsam → ein CTAS vor einem Index Aufbau erwünscht.
- Viele CTAS und Create Index um
  - Zeitaufwand für neue Index zu messen
  - optimale Einstellungen
  - den Lost Write Fehler nochmals zu sehen
- Mehrere Standbys, Sandbox und Failover Rechner (mit/ohne VM)
- LostWrite traf uns noch einmal
  - Wir glauben 3 Mal insgesamt, aber in einem „Sturm“ mit vielen Blöcken

Materialized View – ein bisschen anders benützt

- Als eine Kopie Tabelle
- Mit Fast Refresh möglich „auf Stand“ zu bringen
- Index auf Materialized View ist unabhängig von Quellen Tabelle
- Materialized View kann „abgeschaltet“ werden und die Kopie bleibt

Ablauf

PREPARE

- Create MV Log auf Quelle
- Create MV … Select *
  - Erst als CTAS, dann MV auf PreBuild
- Indexe auf der MV bauen
  - Parallel? SortArea?
- Fast Refresh (mehrmals)

DOWNTIME

- Letztes Fast Refresh
  - Logtabelle - Kontrolle
- Drop Materialized View … Preserve Table
- Rename Table (und Index Name, Grants, usw.)

M2@Miracle.dk DOAG 2017

```
Create Table DATA_MV as Select * From DATA Where 1=2 ;
```

(Tablespace u.ä. hier nicht erwähnt)

```
Create Materialized View Log On DATA With Primary Key ;

Create Materialized View DATA_MV On Prebuilt Table Refresh Fast On Demand As
  Select * From DATA ;

Create Index DATA_N1_MV On DATA_MV (ITEM_ID, TYPE_CODE) ;

Exec Dbms_Mview.Refresh('DATA_MV',Method=> 'F');
```

Downtime Anfang

```
Exec Dbms_Mview.Refresh('DATA_MV',Method=> 'F');
Drop Materialized View DATA_MV Preserve Table ;
Drop Materialized View Log On DATA ;
Alter Table DATA Rename To DATA_Old ;
Alter Table DATA_MV Rename To DATA ;
Alter Index DATA_N1 to DATA_N1_OLD ;
Alter Index DATA_N1_MV to DATA_N1 ;
Grant Select, Update On DATA to USER23 ;
```

Downtime Schluß

Aufräumen

Abendessen

Oder habt Ihr Fragen ?

Michael Möller, Miracle A/S

MIRACLE

We Dare - We Share - We Care