Oracle Database 12c

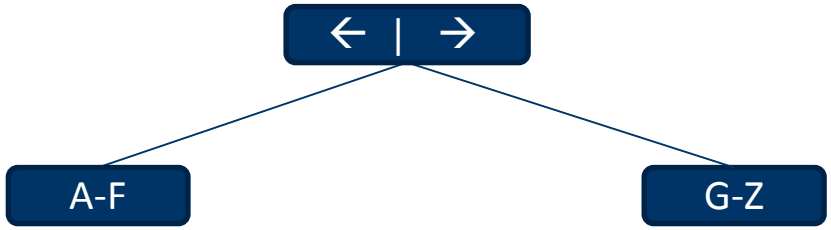
Was Sie immer schon über Indexe wissen wollten

Marco Mischke, 08.09.2015 – DOAG Regionaltreffen

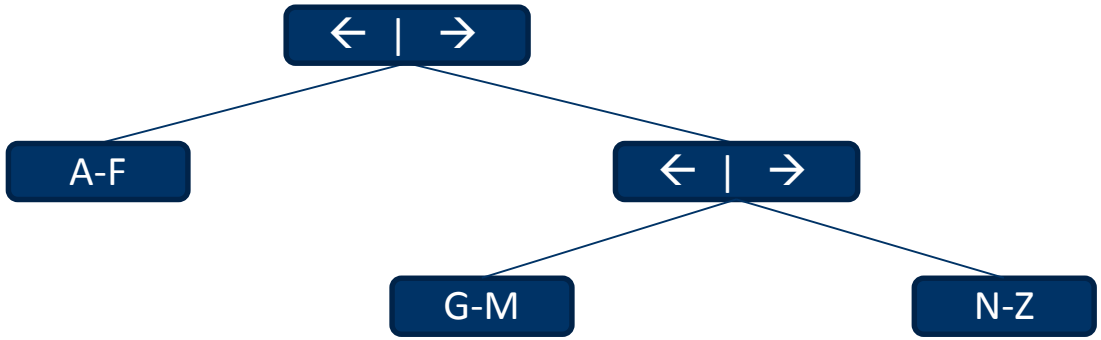
B* Indexe - Aufbau

- 0-Level Index

- 1-Level Index



- 2-Level



Redundante Indexe

- Optimizer kann Indexe mit mehreren Spalten verwenden, wenn die WHERE-Prädikate am Anfang des Index stehen
  - Index 1 auf TTT (ID1, ID2, Wert)
  - Index 2 auf TTT (ID1, ID2)
  - Select … from TTT where ID1= :1 and ID2 = :2
  - INDEX RANGE SCAN auf Index 1 und Index 2 möglich
    - Index 2 ist daher unnötig

`red_idx.sql`

Indexe und NULL Werte

- **Problem:** NULL Werte werden nicht indiziert
  - Create index AAA on BBB (XXX);
  - Select from BBB where XXX is null → Full Table Scan

- **Lösung:** NULL Werte von Spalten in zusammengesetzten Indizes werden indiziert, sofern eine der nachfolgenden Spalten NOT NULL ist
  - Create index AAA on BBB (XXX, 0);
  - Select from BBB where XXX is null → Index Range Scan

idxnull.sql

Indexe und NOT NULL Constraints

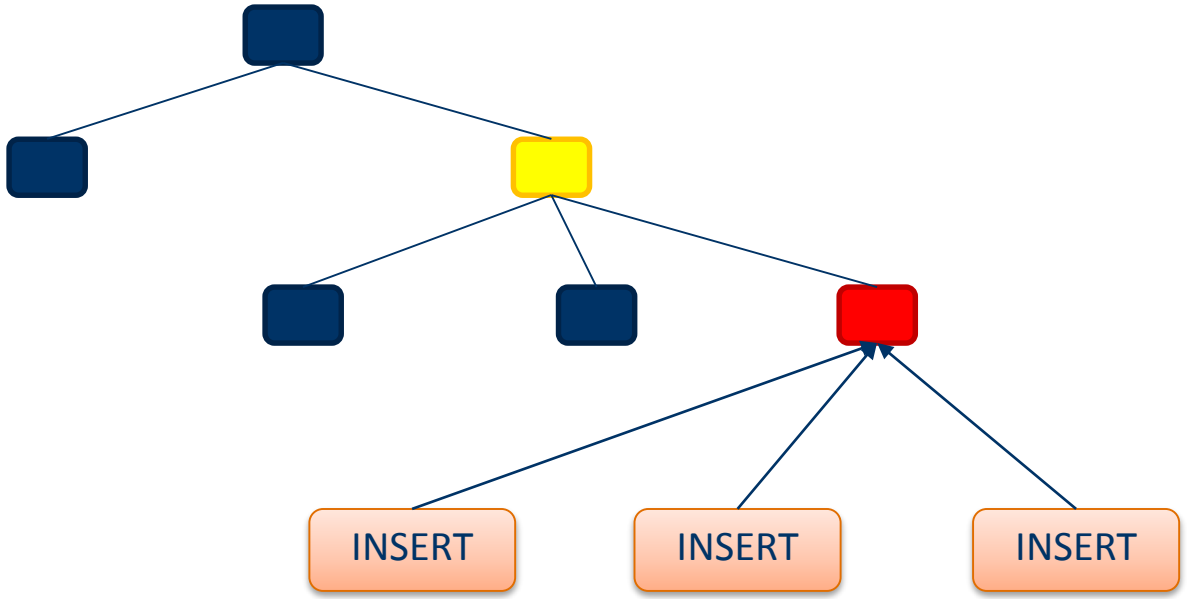
- **Problem:** NULL Werte werden nicht indiziert
  - Create index AAA on BBB (XXX);
  - Spalte XXX enthält keine NULL Werte
  - Select from BBB where XXX is null → Full Table Scan

- **Lösung:** NOT NULL Constraint auf Spalte anlegen
  - Alter table BBB modify(XXX not null);
  - Select from BBB where XXX is null → Index Scan

idxnull2.sql

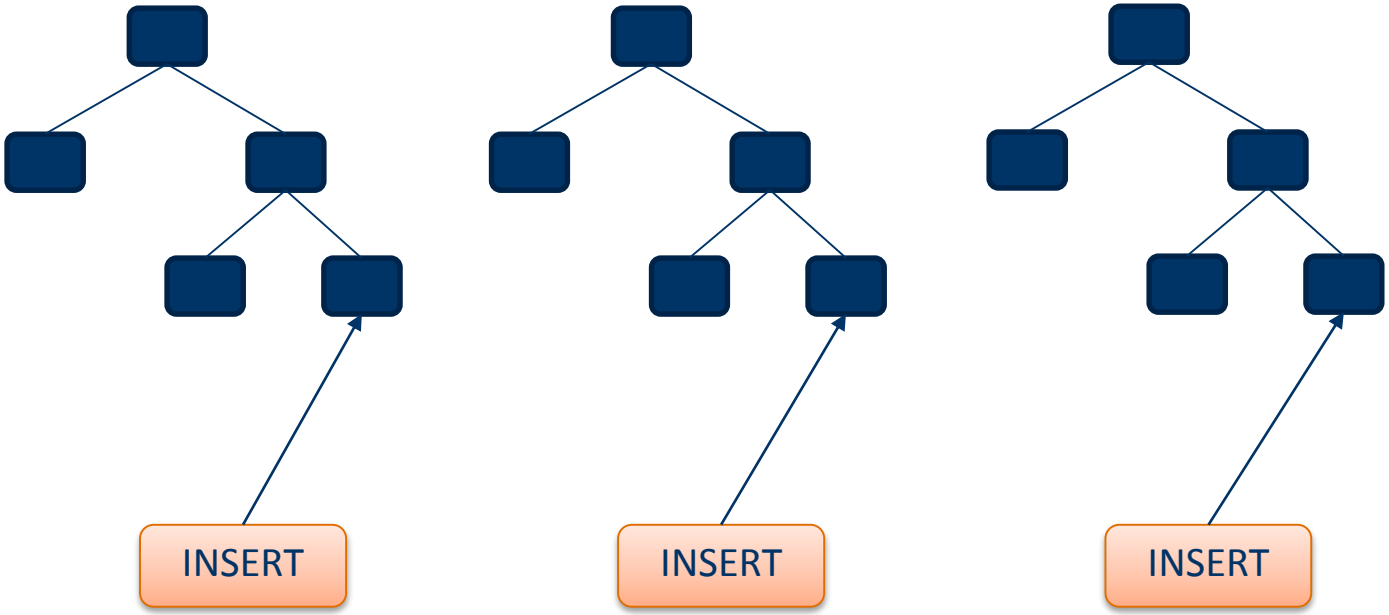
Monoton wachsender Index

- **Problem:** Monoton wachsender Index (ID befüllt mit Sequence) wait event = “enq: TX – index contention”



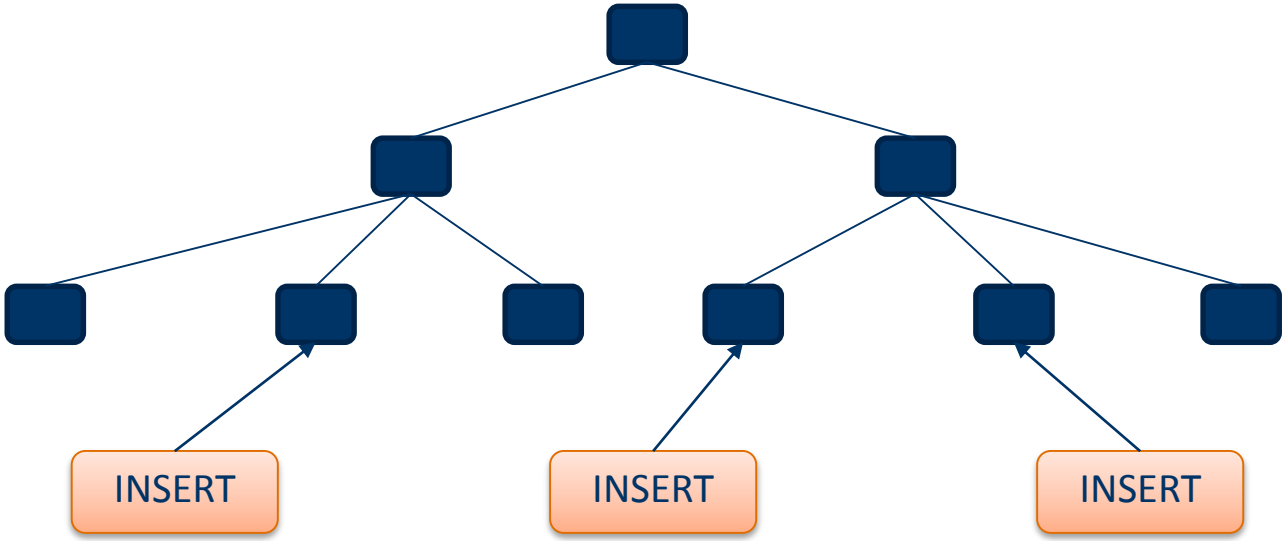
Monoton wachsender Index

- **Lösung 1:** Hash partitioning (EE)
  - Jede ID landet in einem anderen Index Segment
  - Kontra: Abfragen ohne ID-Prädikat vervielfachen die Index Scans



Monoton wachsender Index

- **Lösung 2:** Reverse key index
  - ID wird binär rückwärts indiziert
    187 = 10111011 → 11011101 = 221
    188 = 10111100 → 00111101 = 61
  - Kontra: Keine Index Range Scans oder Sortierungen mehr möglich



Function Based Indexes and NULLs

- **Problem:** Eine Tabelle mit einer Flag-Spalte
  - 99% = ‚Y‘
  - 1% = ‚N‘
- Es wird immer nur nach Flag=‚N‘ gesucht
- Bitmap Index ungünstig weil
  - Enterprise Edition Feature
  - Die Tabelle wird ständig geändert
  - Bitmap Index würde parallele Bearbeitungen behindern
  - Der Bitmap Index würde enorm wachsen
- **Lösung:** Function Based Index

```
create index on TAB(case when flag=‚N‘ then ‚N‘ else null end)
```

selective.sql

Indexe und MIN/MAX

- select MIN() from …
  - Index Range Scan MIN/MAX auf ersten Leaf Block
- select MAX() from …
  - Index Range Scan MIN/MAX auf letzten Leaf Block

- **Problem:** select MIN(), MAX() from …
  - Index Full Scan oder Full Table Scan

- **Lösung:** Gruppenfunktionen in Sub-Select auslagern

idx_range.sql

Index Only Zugriffe

- Index enthält alle Spalten, die im SELECT vorkommen
  - Tabellenzugriff wird komplett eliminiert
- Ist das immer der Fall? → Index Organized Table
  - z.B. für Lookup Tabellen mit ID, WERT

```
create table LOOKUP (
  ID    number,
  WERT varchar2(10),
  constraint PK_LOOKUP primary key (ID)
) organization index;
```

idx_only.sql

Index zum Sortieren

- Index ist sortiert gespeichert
  - ORDER BY kann Index benutzen
- NLS gerechtes Sortieren
  - Index ist normalerweise BINARY sortiert
  - Function Based Index mit NLSSORT() anlegen
  - Bedingungen:
    - optimizer_mode = first_rows
    - query_rewrite_enabled = true
    - nls_sort = <passend>

idx_sort.sql

Index für Datenintegrität

- **Problem:** Kombination von Werten soll unter bestimmten Bedingungen eindeutig sein
  - Trigger zur Prüfung ungeeignet
    - Können Daten anderer Sessions nicht sehen
    - Müssten aufwändig serialisiert werden
    - Kosten vergleichsweise viel Rechenzeit
- **Lösung:** Unique function based index
  - Erinnerung: Zeile landet nur im Index, wenn alle Werte NOT NULL sind

```
create unique index on TAB(
  case when flag=‚A‘ then ID1 else null end,
  case when flag=‚A‘ then ID2 else null end
)
```

idx_integr.sql

Unindexed Foreign Keys

- **Problem:** Ungewöhnliche Sperren auf komplette Tabellen
- **Ursache:**
  - Nicht indizierte Fremdschlüssel
  - Operationen auf Master Tabelle sperren immer die ganze Child Tabelle
- **Lösung:** Fremdschlüssel in der Child Tabelle indizieren
- Kleine Verbesserung 12c: INSERTs sind nun unabhängig

`fkey02_1.sql`

Bitmap Indexe und Locking

- Bitmap Indexe sind ungeeignet für Tabellen mit viel DML (besonders single row DML)
- **Ursache:** Lock Mechanismus für Bitmap Indexe
  - Es gibt eine Bitmap pro Schlüsselwert
  - Jedes Bit in der Bitmap repräsentiert eine ROWID, wenn das Bit gesetzt ist, passt die Zeile zum Schlüssel
  - Sperren betreffen mehrere Tabellenzeilen da immer eine ganze Bitmap Reihe gesperrt wird

01_bitmap.sql

Clustering Factor 1

- Maß für die Sortierung der Daten in der Tabelle
- Clustering Factor gibt an wie viele I/O Operationen zum Lesen der gesamten Tabelle per Index nötig sind
- Clustering Factor geht gegen Anzahl Zeilen
  - Viele verschiedene Datenblöcke pro Indexblock → Datenblöcke werden mehrfach gelesen
  - Eher Full Table Scan

Clustering Factor 2

- Clustering Factor geht gegen Anzahl Datenblöcke
  - Ein oder wenige Datenblöcke pro Indexblock → jeder Datenblock wird nur einmal gelesen
  - Eher Index (Range) Scan
- Index Rebuild hilft nicht → Tabelle muss reorganisiert werden

clusfact.sql

Index Rebuilds 1

- Indexe werden ihre ursprüngliche Größe annehmen weil:
  - Durch Updates in der Tabelle können Schlüsselwerte an andere Stellen im Index wandern
  - An der neuen Stelle ist aber kein Platz durch das REBUILD
  - Der Indexblock muss gesplittet werden um Platz zu schaffen
  - An der alten Stelle entsteht dann freier Platz
  - Mit der Zeit sind die Indexblöcke dann mehr oder weniger halb-leer
- Das ist gut so, es entstehen durch weitere Updates keine Splits mehr

Index Rebuilds 2

- REBUILD als Mittel zum Speicherplatz sparen
- Erfahrungswerte, was kann passieren:
  - der Index wächst danach wieder
  - Es wird mehr Redo generiert
  - Mehr I/O Operationen
  - Langsamere Abfragen
- REBUILD sollte die Ausnahme sein, nicht die Regel
  - Bitmap Indexe nach vielen Änderungen
  - B*Tree Indexe nach großen einmaligen Löschoperationen
-

Marco Mischke
Senior Systemberater Oracle Support

Telefon: 0351 25859-2884
marco.mischke@robotron.de

www.robotron.de

Fragen?