# バッチ処理にバインド変数はもうやめません？

## ～ バッチ処理の突発遅延を題材にして考えてみる ～

Oracle OpenWorld Unconference

presented by JPOUG Japan Oracle User Group

2012/4/6 株式会社コーソル 渡部亮太

CO-Solutions

CO-Solutions

# 今日お伝えしたいこと

**バッチ処理SQLを**
**バインド変数化するの**
**はやめませんか？**

OLTP処理SQLは
バインド変数化してOKなんだけどね・・・

CO-Solutions

# 自己紹介 + 所属企業の紹介

- **渡部 亮太（わたべ りょうた）**

  SE、PM を経験後、Oracle Databaseのエキスパートを目指して転職

  執筆　「プロとしてのOracleアーキテクチャ入門」

  「プロとしてのOracle運用管理入門」

  講演　Developers Summit 2009

  Oracle LOVERS シーズン2 第2回

  Oracle DBA & Developer Days 2010, 2011

- **株式会社コーソル**

  「CO-Solutions＝共に解決する」の理念のもと、Oracle技術に特化した事業を展開中。心あるサービスの提供とデータベースエンジニアの育成に注力している。社員数: 98名（エンジニアのほぼ全員がOracle Master 所有者）

CO-Solutions

# こんな現象を題材に考えます

- 稼動実績があるバッチ処理が、**突然パフォーマンスダウン**するケース
  - 突然SQLのパフォーマンスがダウン
  - バインド変数を使用したSQL
  - 試行錯誤していろいろ対処策を探っているうちに、なぜか通常のパフォーマンスに戻った
  - しかし、その後も不定期にパフォーマンスダウンが再発
  - 原因がつかめない・・・

# 正常稼働時

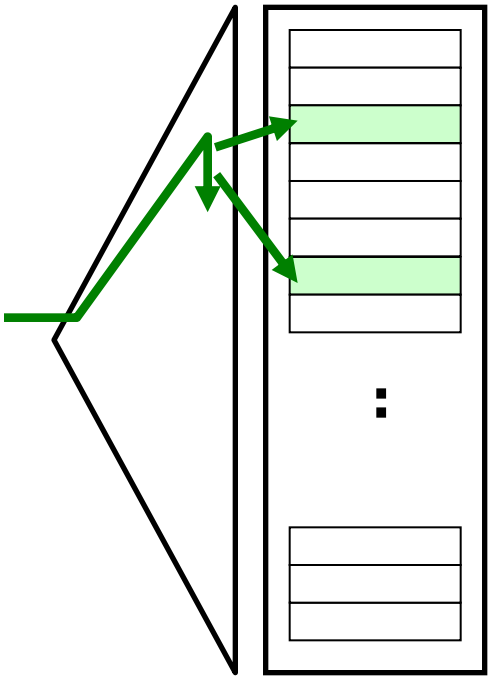
- 範囲検索条件をバインド変数化したSQL
- 検索対象行数は比較的少ない
- INDEX RANGE SCANが適切と想定される

```
SELECT * FROM tbl0
WHERE :sval < val AND val < :eval
```

検索対象行

1

10



# パフォーマンスダウン時

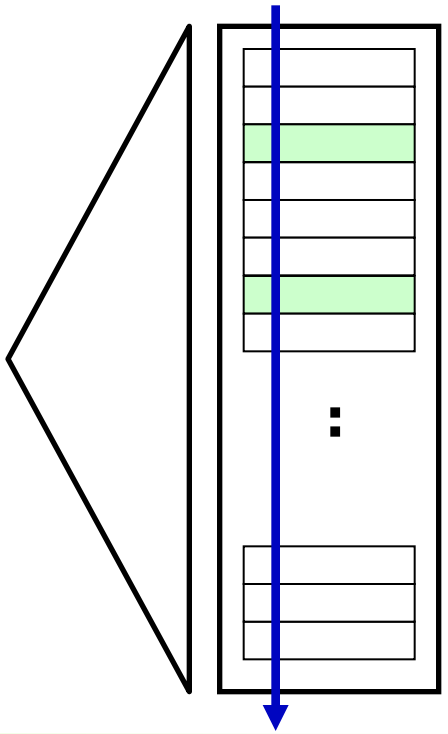
- TABLE FULL SCANが実行されている
  - しかし、現場的には実行計画の変化に気づいていない・・・
- 不要なブロックアクセスが大量に発生

```
SELECT * FROM tbl0
WHERE :sval < val AND val < :eval
```

検索対象行

1

10



# 問題発生のメカニズム

- なぜこのような現象が発生するのか？
- 以下の2つの動作が影響している
  - 1) バインドピーク機能
    - ハードパース時に指定されたバインド変数値を元に実行計画を作成する機能
  - 2) 共有カーソル（子カーソル）の再利用
    - 作成済みの共有カーソルが共有プールに存在する場合、2回目以降のSQL実行で再利用する仕組み
    - 共有カーソルには実行計画が含まれるため、実行計画も再利用される

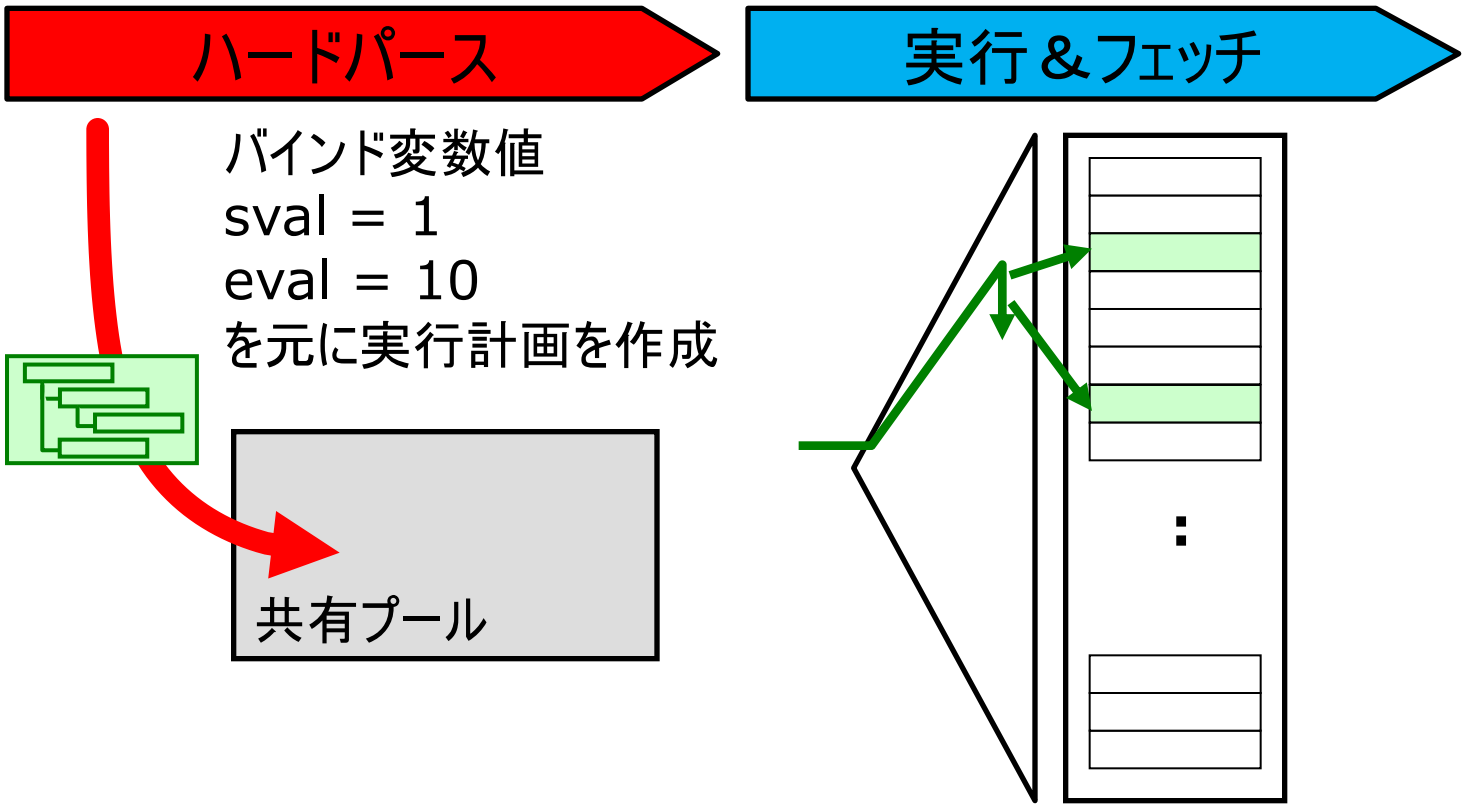
# 初回実行

```
SELECT * FROM tbl0
WHERE :sval < val AND val < :eval
```

検索対象行

1

10



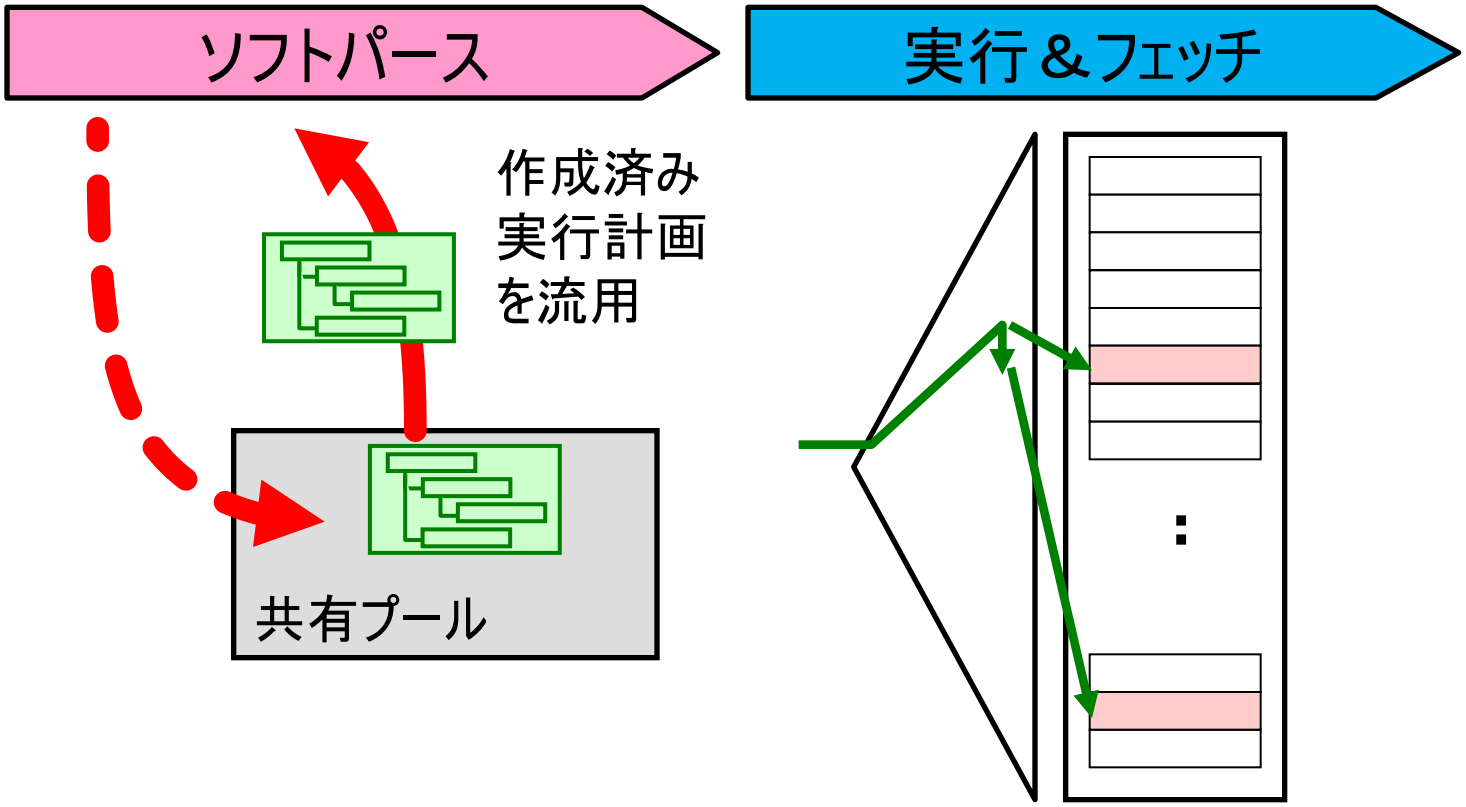
# 2回目以降の実行

```
SELECT * FROM tbl0
WHERE :sval < val AND val < :eval
```

検索対象行

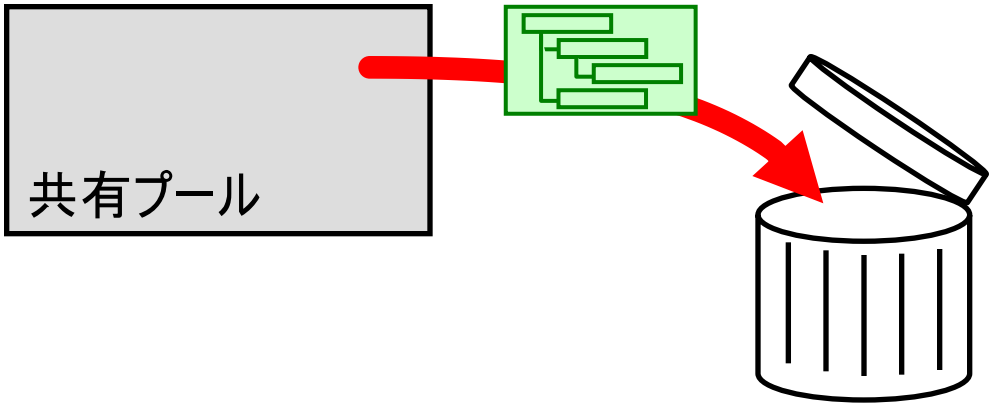
11

20



# 共有カーソルのage-out

共有プールの領域不足、統計情報の更新など
様々な理由で共有カーソルはage-outされる場合がある



# (age-out後の)初回実行

SELECT * FROM tbl0
WHERE :sval < val AND val < :eval

検索対象行

100

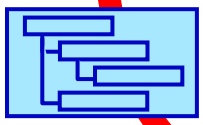
5000

注目

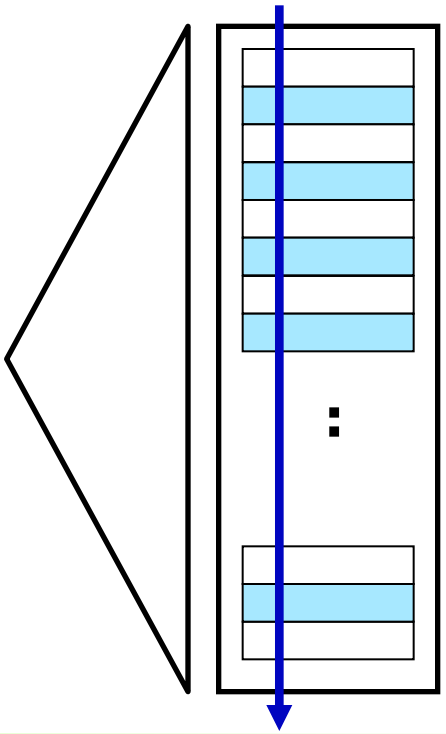
ハードパース

実行＆フェッチ

バインド変数値
sval = **100**
eval = **5000**
を元に実行計画を作成



共有プール



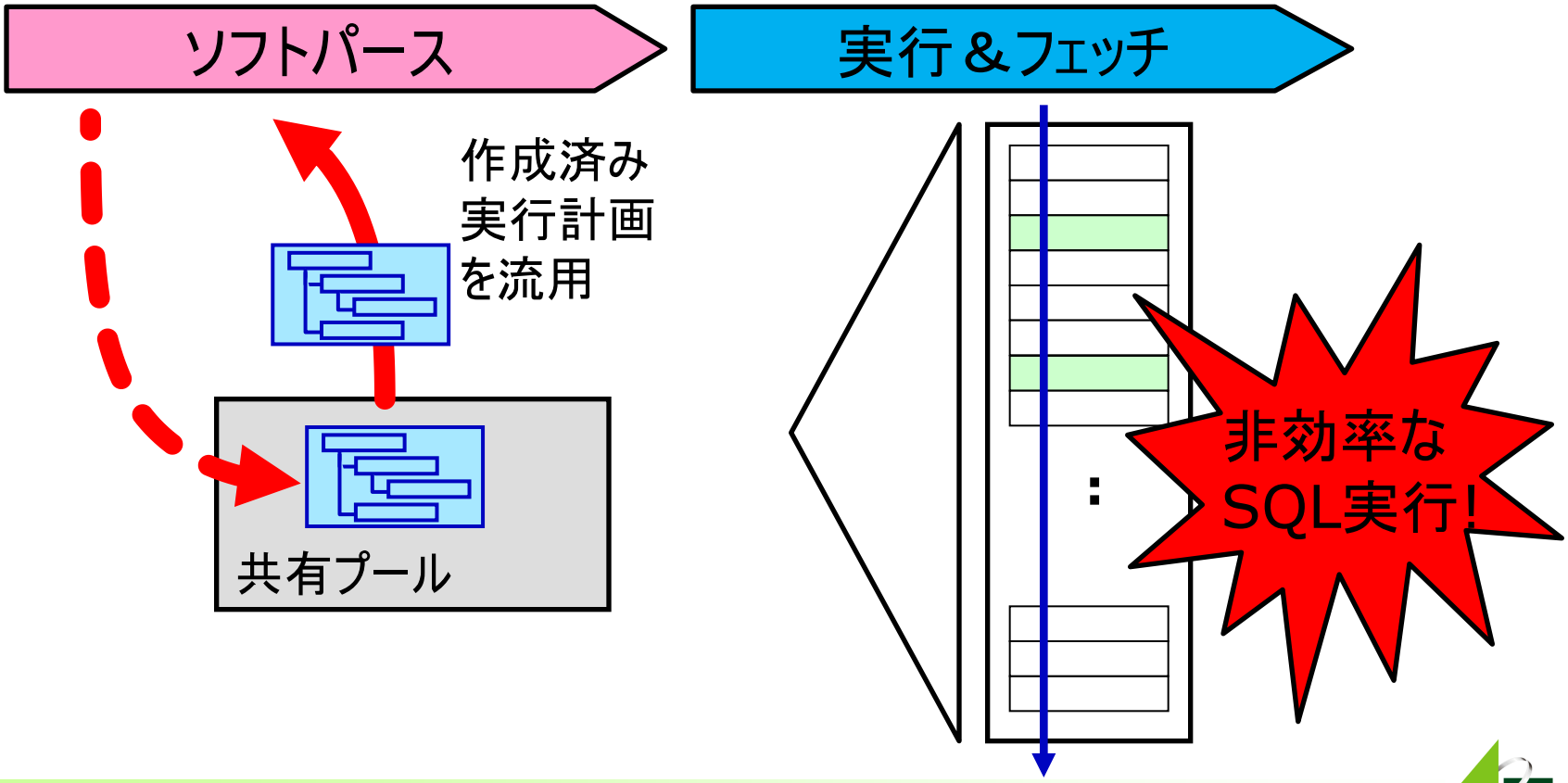
# (age-out後の) 2回目以降の実行

```
SELECT * FROM tbl0
WHERE :sval < val AND val < :eval
```

検索対象行

1

10



# 混乱した現場が・・・

- 共有プールの強制flush、統計情報の再収集、インスタンスの再起動などを実行すると・・・
- 共有カーソルのage-out相当の処理が実行される形になる
- ハードパースが再実行され、想定通りの実行計画となる

- （問題のメカニズムが分かっていれば、上記の動作は当たり前の話なのだが・・・）現場的には、なぜかよくわからないが問題が解消したように見える
- が、以後も偶発的に同様の現象が発生・・・

# 対処策は？

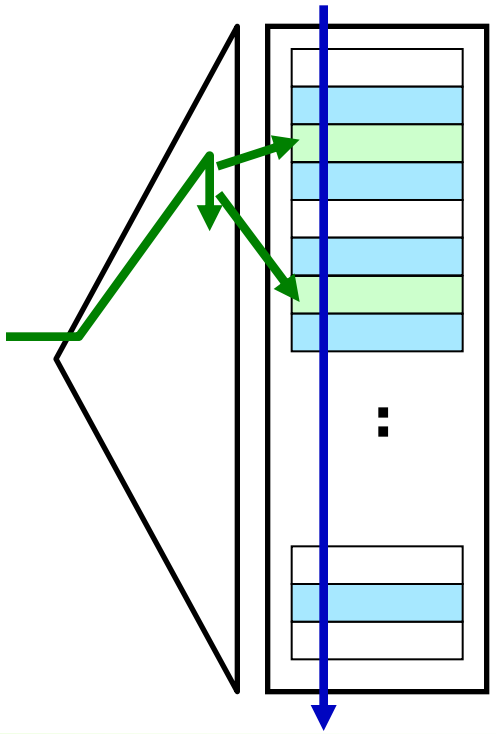
| 対処策 | 説明 |
| --- | --- |
| SQLをリテラル（バインド変数を使用しない）に修正 | リテラル値（≒WHERE条件）毎に実行計画を作成<br>→ 異なるWHERE条件が指定されたSQLに対して、1つの実行計画を使いまわさないようになる |
| バインドピークを無効化(_optim_peek_user_binds=false) | デフォルト値を基準に実行計画を作成する<br>→ 実行計画がバインド変数値に依存しないようになる |
| 実行計画を誘導（ヒント、アウトライン、SQL実行計画管理） | 指定した実行計画が作成される<br>→実行計画がバインド変数値に依存しないようになる |

# 対処策1) SQLをリテラルに修正

- WHERE条件ごとに実行計画が作成される
  - 子カーソル(≒実行計画)が共有されない
  - 個々のWHERE条件に最適な実行計画が作成される

```
SELECT * FROM tbl0
WHERE  1 < val AND val < 10
```
検索対象行

```
SELECT * FROM tbl0
WHERE  100 < val AND val < 5000
```
検索対象行



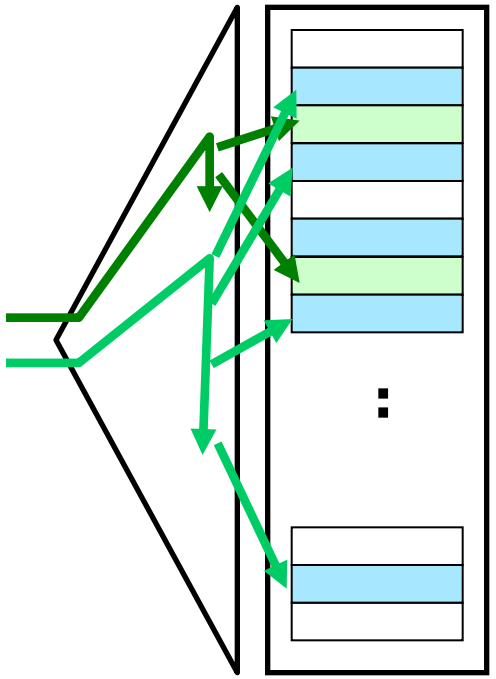
CO-Solutions

# 対処策2) バインドピークを無効化

- _optim_peek_user_binds=false
- デフォルト値(*1)を基準に実行計画が作成される
  - 子カーソル(≒実行計画)が共有される
  - ある意味・・・平均的な実行計画が作成される

SELECT * FROM tbl0
WHERE :sval < val AND val < :eval
検索対象行
1
10

SELECT * FROM tbl0
WHERE :sval < val AND val < :eval
検索対象行
100
5000



(*1) デフォルト値の例
'=', 'LIKE' の選択率 : 1/NUM_DISTINCT
そのほかの選択率 : 0.05

# 対処策3) 実行計画を誘導

- 指定した実行計画が作成される
  - 子カーソル(≒実行計画)が共有される
  - どの実行計画に誘導するか？という問題が残る

```
SELECT /*+INDEX(tbl0) */
* FROM tbl0
WHERE :sval < val AND val < :eval
```

検索対象行

1

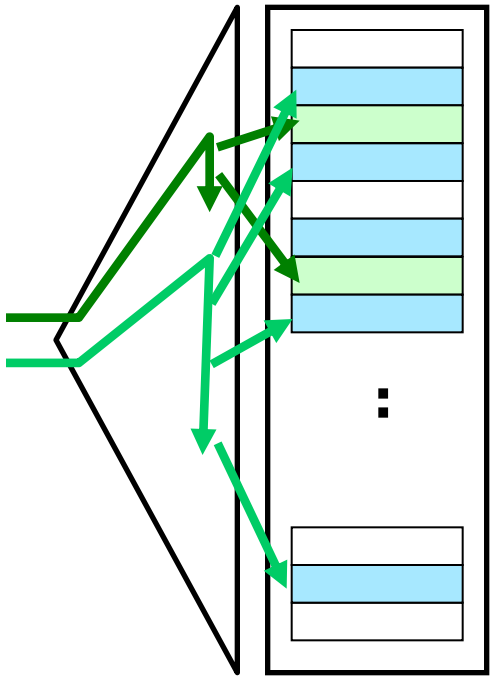
10

```
SELECT /*+INDEX(tbl0) */
* FROM tbl0
WHERE :sval < val AND val < :eval
```

検索対象行

100

5000




CO-Solutions

# 「そもそも論」で考える

- 「そもそも」SQLをバインド変数化する目的とは？
  - バインド変数化するメリット、デメリット
  - バインド変数化が「ある意味」盲目的に推奨されている背景
- 「そもそも」実行計画の共有/非共有が、「するか しないかの２択」なのはいかがなものか
  - もう少し賢い仕組みがあってしかるべきでは
  - というわけでAdaptive Cursor Sharing (11.1-)

# 「そもそも論」で考える

- 「そもそも」SQLをバインド変数化する目的とは？
  - バインド変数化するメリット、デメリット
  - バインド変数化が「ある意味」盲目的に推奨されている背景
- 「そもそも」実行計画の共有/非共有が、「するか しないかの２択」なのはいかがなものか
  - もう少し賢い仕組みがあってしかるべきでは
  - というわけでAdaptive Cursor Sharing (11.1-)

# バインド変数化のメリット・デメリット

| ○/× | 案 | 説明 |
|---|---|---|
| ○ | ハードパース実行回数の削減によるCPU使用率の削減 | 異なるWHERE条件を指定した多くのSQLが発行される場合は**効果大** |
| ○ | 共有プール使用量の削減 | 異なるWHERE条件を指定した多くのSQLが発行される場合は**効果大** |
| × | WHERE条件により最適な実行計画が異なる場合でも同一の実行計画を使用 | どのようなWHERE条件でも**最適な実行計画が同じ**場合は問題とならない |

# バッチ処理SQLとバインド変数化

| ○/× | 案 | バッチ処理SQLの場合 |
|---|---|---|
| ○ | ハードパース実行回数の削減によるCPU使用率の削減 | 異なるWHERE条件を指定した多くのSQLが発行されないため、**効果小** |
| ○ | 共有プール使用量の削減 | 異なるWHERE条件を指定した多くのSQLが発行されないため、**効果小** |
| × | WHERE条件により最適な実行計画が異なる場合でも同一の実行計画を使用 | WHERE条件により**最適な実行計画が異なる**場合が多い |

⇒ バインド変数化の利点は、バッチ処理SQLの特性に適合しない

# バインド変数化を盲目的に推奨する風潮？

- バインド変数化の有無は、開発者がそれぞれのSQLごとに判断する必要がある
  - 現在のOracle Databaseでは・・・残念ながら
- しかし、盲目的に**「バインド変数化＝善」**と判断している風潮が見られる
  - コーディング規約でのルール化
  - DBアクセスロジックを過度に共通化

突発的なパフォーマンスダウンを避けるため、盲目的にバインド変数を使用することは避けてほしい

# 「そもそも論」で考える

- 「そもそも」SQLをバインド変数化する目的とは？
  - バインド変数化するメリット、デメリット
  - バインド変数化が「ある意味」盲目的に推奨されている背景
- 「そもそも」実行計画の共有/非共有が、「するか しないかの２択」なのはいかがなものか
  - もう少し賢い仕組みがあってしかるべきでは
  - というわけでAdaptive Cursor Sharing (11.1-)

# 実行計画の共有/非共有の判断

- WHERE条件が異なるSQLと実行計画の共有/非共有
  - nのSQLについて1つの実行計画を共有するか
  - nのSQLについてそれぞれnの実行計画を作成する
- → 実行計画を共有するかしないか
- 「そもそも」もうすこしインテリジェントな仕組みがあっても良いのでは？

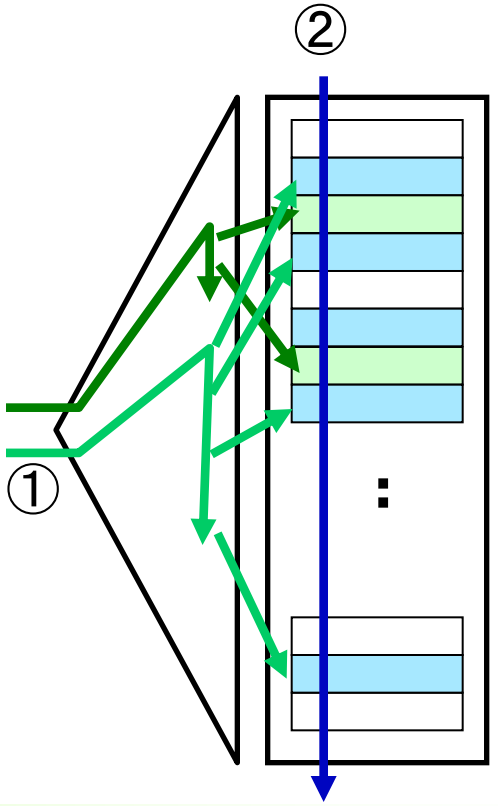
# Adaptive Cursor Sharing (11.1-)

- 作成済みの実行計画(≒子カーソル)が不適切な場合は、自動的に新規に子カーソルを作成する機能

SELECT * FROM tbl0 検索対象行
WHERE :sval < val AND val < :eval
1 10

①
SELECT * FROM tbl0 検索対象行
WHERE :sval < val AND val < :eval
**100** **5000**

②
SELECT * FROM tbl0 検索対象行
WHERE :sval < val AND val < :eval
**100** **5000**




CO-Solutions

# Adaptive Cursor Sharingで万事OK?

- 残念ながらそんなことはない
  - Feedback-Baseなので、最低1回は痛い目にあわないとダメ
  - （実行イメージ）作成済みの実行計画で実行
    →（パフォーマンスダウン＋）予測値と実測値に乖離
    →新規に実行計画を作成
- （大量データを処理する）バッチ処理の場合、痛い目にあってからでは遅い・・・
  - したがって、バッチ処理の場合は、SQLのリテラル化がやっぱりオススメ

# ご参加いただきありがとうございました

Oracle OpenWorld Unconference

presented by JPOUG Japan Oracle User Group

ひきつづき鼓動をお楽しみください

CO-Solutions

# 中間的な特性を持つSQLでは？

- バッチ処理とOLTPの中間的な特性を持つSQLでは、どういうアプローチが適切か
  - WHERE条件により最適な実行計画が異なる
  - 実行頻度はそれなりに高い、条件の種類も多い
- 対処策
  - たまに発生する実行計画の作成ミスには目をつぶり、ACSに頼る
  - 最適な実行計画が異なるバインド変数値を洗い出し、その場合のみリテラル化 or 実行計画を誘導（それなりに大変なはず）

# CURSOR_SHARING=SIMILAR

- CURSOR_SHARING
  - リテラルSQLを自動的にバインド変数化する機能
- CURSOR_SHARING=SIMILAR
  - SQLの種類とオプティマイザ統計の取得状況が条件に合致する場合、WHERE条件毎に別の子カーソル（≒実行計画）を作成する
  - 意図しない実行計画の共有を回避できる反面、子カーソル数の肥大化を招く場合があるため、適用は慎重に
  - 11.2以降で非推奨、11.2.0.3で使用不可