Statistics and Migrations

Well-Kept Secrets Revealed

Virtual Classroom Seminar #29 - June 2026

Photo by Stefan Steinbauer on Unsplash

Oracle

DBAs

run the world

Mike Dietrich

Vice President

mikedietrich

@mikedietrichde.com

https://mikedietrichde.com

Daniel Overby Hansen

Distinguished Product Manager

dohdatabase

@dohdatabase.com

https://dohdatabase.com

Rodrigo Jorge

Distinguished Product Manager

rodrigoaraujorge

@dbarj.com.br

https://www.dbarj.com.br

Alex Zaballa

Distinguished Product Manager

alexzaballa

@alexzaballa.bsky.social

https://alexzaballa.com

Alejandro Diaz

Product Manager

alejandrodiazs

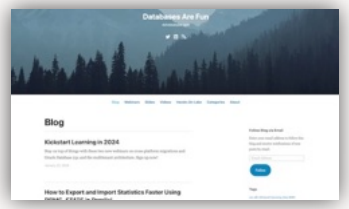
axdiaz.bsky.social

https://axdiaz.com

Find Slides and Much More on Our Blogs

MikeDietrichDE.com

Mike.Dietrich@oracle.com



dohdatabase.com

Daniel.Overby.Hansen@oracle.com

DBArj.com.br

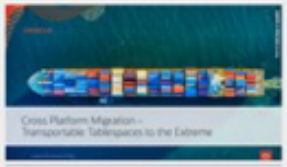
Rodrigo.R.Jorge@oracle.com

AlexZaballa.com

Alex.Zaballa@oracle.com

axdiaz.com

jorge.a.diaz@oracle.com

Virtual Classroom Seminars	Slides
Episode 16 (replaces Episode 1 from Feb 2021) Oracle Database Release and Patching Strategy for 19c and 23c *115 minutes – May 10, 2023*	
Episode 17 From SR to Patch – Insights into the Oracle Database Development process *55 minutes – June 22, 2023*	
Episode 18 Cross Platform Migration – Transportable Tablespaces to the Extreme *145 min – February 22, 2024*	
Episode 19 Move to Oracle Database 23ai – Everything you need to know about Multitenant PART 1 *145 min – May 16, 2024*	
Episode 20 Move to Oracle Database 23ai – Everything you need to know about Multitenant PART 2 *100 min – June 28, 2024*	
Episode 21 One-Button Patching with AutoUpgrade – Easing every DBA's life *55 min – October 24, 2024*	

Recorded Web Seminars

https://MikeDietrichDE.com/videos

More than 45 hours of technical content
On-demand, anytime, anywhere

Performance and Migrations

What's the Worst That Can Happen?

Photo by Ugi K. on Unsplash


Photo by Dave Hoefler on Unsplash

Photo by Craig Pattenaude on Unsplash


Photo by Urip Dunker on Unsplash

A Word About Statistics

Statistics

Object Statistics

Statistics Preferences

Column Usage Information

Statistics

Object Statistics

Statistics Preferences

Column Usage Information

```
SQL> select table_name, num_rows, blocks, empty_blocks, avg_row_len
     from   dba_tab_statistics;

   NUM_ROWS     BLOCKS     EMPTY_BLOCKS     AVG_ROW_LEN
___________ ___________ ________________ ________________
     571047        2638                0              27
```


```
SQL> select column_name, num_distinct, density, num_nulls, histogram
     from   dba_tab_col_statistics;
```

COLUMN_NAME	NUM_DISTINCT	DENSITY	NUM_NULLS	HISTOGRAM
RACEID	528	0.00189	0	HYBRID
DRIVERID	141	0.00000875584671664504	0	FREQUENCY
LAP	87	0.00000875584671664504	0	FREQUENCY
POSITION	24	0.00000875584671664504	0	FREQUENCY
TIME	76224	0.000013	0	HYBRID
MILLISECONDS	75328	0.000013	0	HYBRID

Statistics

Statistics Preferences

Object Statistics

Column Usage Information

```
BEGIN
   DBMS_STATS.SET_TABLE_PREFS (
      OWNNAME => 'APPUSER',
      TABNAME => 'SALES',
      PNAME   => 'TABLE_CACHED_BLOCKS',
      PVALUE  => '42'
   );
END;
```


Table 171-131 SET_TABLE_PREFS Procedure Parameters

Parameter	Description
ownname	Owner name
tabname	Table name
pname	Preference name. You can set the default value for following preferences: • APPROXIMATE_NDV_ALGORITHM • AUTO_STAT_EXTENSIONS • CASCADE • DEGREE • ESTIMATE_PERCENT • GRANULARITY • INCREMENTAL • INCREMENTAL_LEVEL • INCREMENTAL_STALENESS • METHOD_OPT • NO_INVALIDATE • OPTIONS • PREFERENCE_OVERRIDES_PARAMETER • PUBLISH • STALE_PERCENT • TABLE_CACHED_BLOCKS
pvalue	Preference value. If NULL is specified, it will set the Oracle default value.

PL/SQL Packages and Types Reference, Oracle Database 19c

```
BEGIN
   DBMS_STATS.SET_GLOBAL_PREFS (
      PNAME    => 'STALE_PERCENT',
      PVALUE   => '5'
   );
END;
```


You often use statistics preferences to solve a particular problem

- Evaluate whether that problem exists in the target environment

Statistics

Object Statistics

Statistics Preferences

Column Usage Information

Column Usage Information

- Information on how you join tables and use predicates
- Stored internally in `SYS.COL_USAGE$`

```
SQL> select * from col_usage$
```

OBJ#	INTCOL#	EQUALITY_PREDS	EQUIJOIN_PREDS	NONEQUIJOIN_PREDS	RANGE_PREDS	...
1392	1	54	10	0	0	...
104	1	55	8	0	0	...

Column Usage Information

- Used by the optimizer to determine when to create histograms

```
exec dbms_stats.gather_table_stats(
       ... ,
       METHOD_OPT => FOR ALL COLUMNS SIZE AUTO
   );
```

Without column usage information, statistics gathering creates no histograms

- Applicable to `METHOD_OPT` set to `SIZE AUTO`

Cost Based Optimizer needs accurate statistics to produce a good plan

PHYSICAL

LOGICAL

Generally, bring your statistics when you do physical migrations

PHYSICAL

LOGICAL

Your Options

1 Include statistics in Data Pump

2 Exclude statistics in Data Pump
Regather statistics after import

3 Exclude statistics in Data Pump
Import statistics using DBMS_STATS

1 Include statistics in Data Pump

2 Exclude statistics in Data Pump
Regather statistics after import

3 Exclude statistics in Data Pump
Import statistics using DBMS_STATS

Default

```
expdp ... include=statistics
```


Data Pump | Statistics

Data Pump includes:

- Object statistics
- Table level statistics preferences
- Column usage information

Data Pump does not include:

- Global statistics preferences

Your statistics settings don't match those of Autonomous AI Database

- Probably…

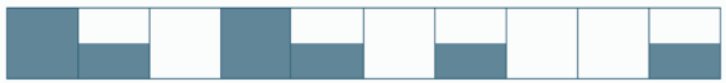
Importing statistics using Data Pump is a convenient, but slow option

- Not recommended

Importing or transporting statistics might be a bad idea…

When source and target database do not match

Statistics | When Importing Stats Is Bad



Fragmented table

Blocks	12000
Leaf blocks	11000
B-level	4
Clustering factor	10000

Compacted table

Blocks	12000
Leaf blocks	11000
B-level	4
Clustering factor	10000

```
DBMS_STATS.GATHER_TABLE_STATS(...
```

Blocks	5000
Leaf blocks	4000
B-level	2
Clustering factor	20000

Statistics | When Importing Stats Is Bad

- Potentially a problem
  - Fragmented tables
  - Changing block size
  - Changing character set
  - Compress or decompress
  - …
- Only a problem for table and index base statistics, column statistics remain accurate

1 Include statistics in Data Pump

2 Exclude statistics in Data Pump
Regather statistics after import

3 Exclude statistics in Data Pump
Import statistics using DBMS_STATS

```
expdp ... exclude=statistics
```


Gather statistics fast

- Blog post by Nigel Bayliss

```
begin
 dbms_stats.gather_database_stats(
       options => 'GATHER',
       cascade => TRUE,
       degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```


```
begin
 dbms_stats.gather_database_stats(
      options => 'GATHER',
      cascade => TRUE,
      degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```


```
begin
 dbms_stats.gather_database_stats(
      options => 'GATHER',
      cascade => TRUE,
      degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```

```
begin
 dbms_stats.gather_database_stats(
      options => 'GATHER',
      cascade => TRUE,
      degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```


You're missing statistics preferences

```
EXEC DBMS_STATS.EXPORT_TABLE_PREFS('APPUSER', 'SALES', 'STAGING_TAB');

expdp ... tables=APPUSER.STAGING_TAB
```

```
impdp ... tables=APPUSER.STAGING_TAB

EXEC DBMS_STATS.IMPORT_TABLE_PREFS('APPUSER', 'SALES', 'STAGING_TAB');
```


You're missing column usage information

- Statistics gathering creates no histograms

EXCLUDE

`EXCLUDE=STATISTICS`

`COL_USAGE$` empty

REGATHER

First time only

```
METHOD_OPT =>
... SIZE SKEWONLY
```

GO LIVE

Column usage information is updated

REGATHER

Use default

```
METHOD_OPT =>
... SIZE AUTO
```

```
begin
 dbms_stats.gather_database_stats(
      options    => 'GATHER',
      cascade    => TRUE,
      degree     => DBMS_STATS.AUTO_DEGREE,
      method_opt => FOR ALL COLUMNS SIZE SKEWONLY);
end;
/
```


Gather Statistics

Gathering stats with `SIZE SKEWONLY` probably:

- Creates more histograms than `SIZE AUTO`
- Takes a little longer time

But better too many, than too few histograms

- Missing histograms may lead to suboptimal plans

Further Information

Transporting statistics

- Blog post: How to Export and Import Statistics Faster Using DBMS_STATS in Parallel
- Blog post: If Importing Statistics Using DBMS_STATS Is Slow
- Blog post: Does Exporting Database Statistics Include the Dictionary Statistics?
- Video: Transporting optimizer statistics - pro tips
- Video: Transporting optimizer statistics - demo
- Video: Transporting optimizer statistics - the concept

1 Include statistics in Data Pump

2 Exclude statistics in Data Pump
Regather statistics after import

3 Exclude statistics in Data Pump
Import statistics using DBMS_STATS

```
EXEC DBMS_STATS.EXPORT_SCHEMA_STATS('APPUSER', 'STAGING_TAB');

expdp ... tables=APPUSER.STAGING_TAB
```

```
impdp ... tables=APPUSER.STAGING_TAB

EXEC DBMS_STATS.IMPORT_SCHEMA_STATS('APPUSER', 'STAGING_TAB');
```


The database considers imported statistics as *current*

- Be sure those statistics are accurate

Data Pump | Statistics

You're missing statistics preferences

You're missing column usage information

- Warm up the database before next stats gathering

Performance Stability Prescription

What if we had a *one-button* tool that points out the bad statements…

SQL Performance Analyzer

Database Licensing Information User Manual

1 Licensing Information

1.1 Introduction

This Licensing Information document is a part of the product or program documentation under the terms of your Oracle license agreement and is intended to help you understand the program editions, entitlements, restrictions, prerequisites, special license rights, and/or separately licensed third party technology terms associated with the Oracle software program(s) covered by this document (the "Program(s)"). Entitled or restricted use products or components identified in this document that are not provided with the particular Program may be obtained from the Oracle Software Delivery Cloud website (https://edelivery.oracle.com) or from media Oracle may provide. If you have a question about your license rights and obligations, please contact your Oracle sales representative, review the information provided in Oracle's Software Investment Guide (http://www.oracle.com/us/

Oracle Real Application Testing	Extra cost option: **EE-ES** Included option: **BaseDB EE**, **BaseDB EE-HP**, **BaseDB EE-EP**, **ExaDB**	Oracle Real Application Testing includes the following features: • Database Replay • SQL Performance Analyzer (SPA) • Database Migration Planner • Database Migration Workbench **Database Replay** • The Oracle Real Application Testing license is required on both capture and replay systems for Database Replay and is charged by the total number of CPUs on those systems. Licensing is also charged by the total number of CPUs on both systems when the capture is done on any read-only standby database and the workload is replayed on a True Cache. • Use of Capture and Replay ASH Analytics Reports, Compare Period ADDM Reports, and Replay Compare Period Reports also requires an Oracle Diagnostics Pack license. • An Oracle Real Application Testing license permits you to access Database Replay functionality through Oracle Enterprise Manager, as well as through the following database server command-line APIs: `DBMS_WORKLOAD_CAPTURE` package and `DBMS_WORKLOAD_REPLAY` package. The use of the `DBMS_WORKLOAD_REPLAY.COMPARE_PERIOD_REPORT()` function also requires a license for Oracle Diagnostics Pack. **SQL Performance Analyzer (SPA)** An Oracle Real Application Testing license permits you to access SQL Performance Analyzer functionality through Oracle Enterprise Manager, as w as through the following database server command-line API: `DBMS_SQLPA`

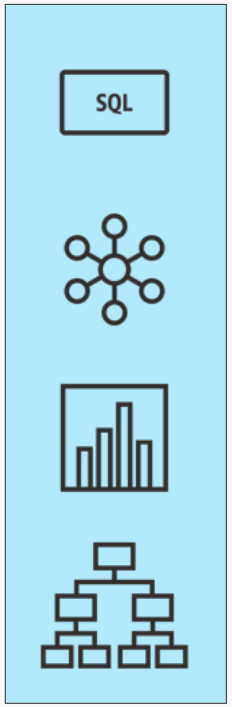
PERFORMANCE STABILITY


1
CAPTURE
2
ANALYZE
3
FIX
4
REMEDY


1
CAPTURE
Capture workload information into SQL Tuning Set
2
ANALYZE
3
FIX
4
REMEDY

SQL Tuning Set | Definition

SQL Tuning Set



SQL statement

Context

Statistics

Plans

```
--Capture directly from cursor cache at regular intervals
exec DBMS_SQLSET.CAPTURE_CURSOR_CACHE ( ... );

--One time sample from cursor cache
exec DBMS_SQLSET.SELECT_CURSOR_CACHE ( ... );
```


```
--Capture directly from cursor cache at regular intervals
exec DBMS_SQLSET.CAPTURE_CURSOR_CACHE ( ... );

--One time sample from cursor cache
exec DBMS_SQLSET.SELECT_CURSOR_CACHE ( ... );

--Capture from AWR
SQL> exec DBMS_SQLSET.SELECT_WORKLOAD_REPOSITORY ( ... );
```


```
SQL> select name, statement_count from dba_sqlset;

NAME                                STATEMENT_COUNT
_____________________________ ____________________
SALES_APP_WORKLOAD                             2343
```


```
SQL> select view_name from dba_views where view_name like
'DBA%SQLSET%';

VIEW_NAME

_____________________________

DBA_SQLSET
DBA_SQLSET_REFERENCES
DBA_SQLSET_STATEMENTS
DBA_SQLSET_BINDS
DBA_SQLSET_PLANS
```


```
SQL> select name, statement_count from dba_sqlset;

NAME                                STATEMENT_COUNT
_____________________________ ____________________
SALES_APP_WORKLOAD                             2343
SYS_AUTO_STS                                   6426
```

Automatic SQL Tuning Set is not meant to be transported

- Collect into your own SQL Tuning Set

Gather at least a full month of workload data

- Assist in testing your database
- Useful in solving post-migration performance problems

```
EXEC DBMS_SQLSET.PACK_STGTAB ( ... );

...

EXEC DBMS_SQLSET.UNPACK_STGTAB ( ... );
```


```
-- If you have many bind variables in your queries, increase the number of binds
-- to capture from 400 (default) to 3999 (max)

alter system set "_cursor_bind_capture_area_size"=3999;
```


Further Information

SQL Tuning Set

- Blog post: Oracle SQL Tuning Sets (STS) – The foundation for SQL Tuning
- Blog post: What is the automatic SQL tuning set?

1

CAPTURE

2

ANALYZE

Analyze performance after migration using SQL Performance Analyzer

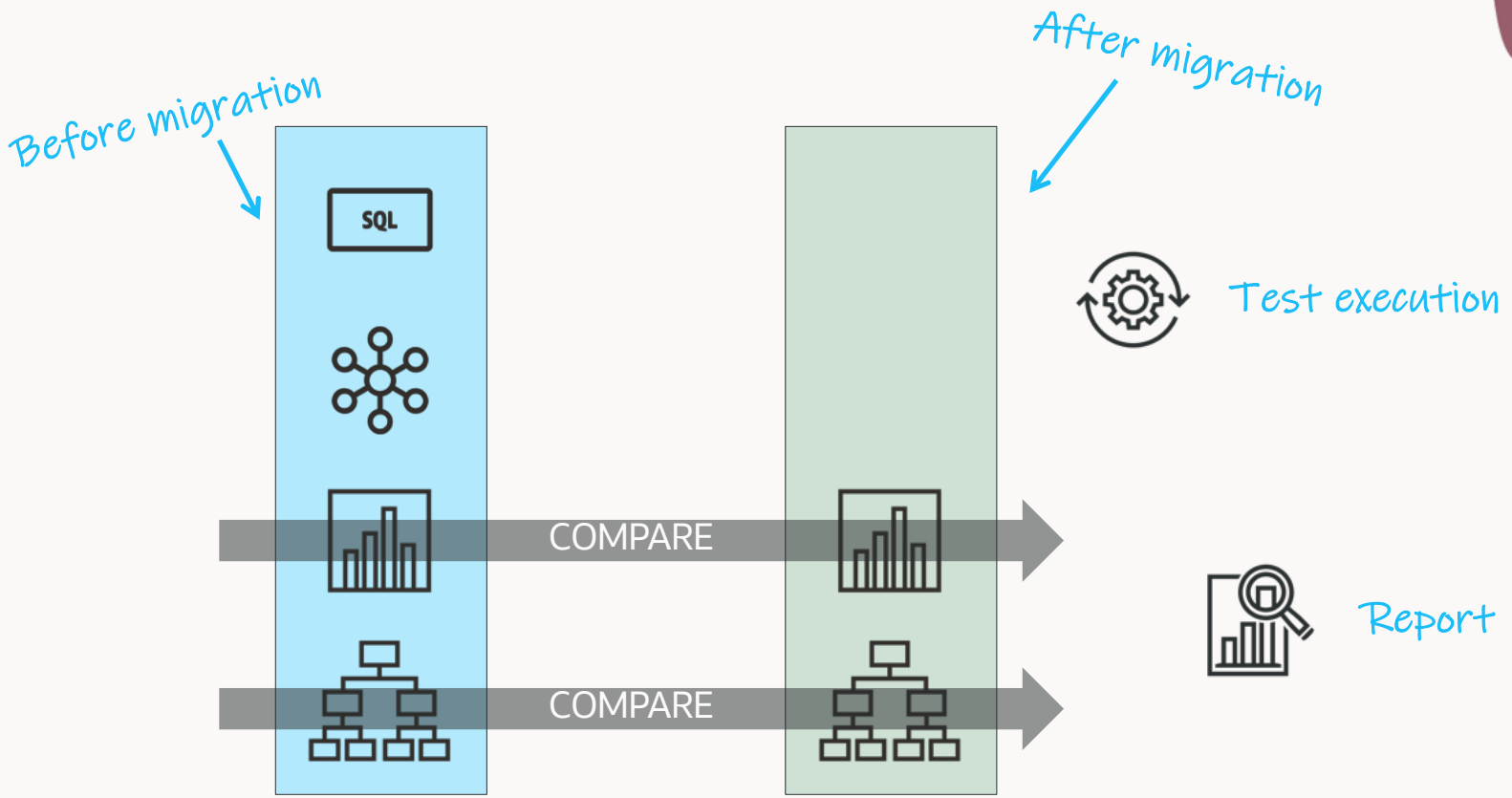
3

FIX

4

REMEDY

SQL Performance Analyzer



Report Summary

Projected Workload Change Impact:

Overall Impact	**4.64%**
Improvement Impact	**30.98%**
Regression Impact	**-26.35%**

SQL Statement Count

SQL Category	SQL Count	Plan Change Count
Overall	43	5
Improved	1	1
Regressed	1	1
Unchanged	30	3
Unsupported	11	0

Top 32 SQL Sorted by Absolute Value of Change Impact on the Workload

object_id	sql_id	Impact on Workload	Execution Frequency	Metric Before	Metric After	Impact on SQL	Plan Change
83	f90zn75aphu4w	30.98%	2828	139319.505304102	41950	69.89%	y
47	0cwuxyv314wcg	-26.35%	18254	981.459680070122	13809	-1306.99%	y
80	csv0xdm9c394t	-.36%	2734	4689.26664228237	5868	-25.14%	n
60	4hbzjyh4p336s	.21%	2818	668.862668559262	10	98.5%	n
76	a8ntu3081hfgw	-.18%	2818	262.609297374024	828	-215.3%	y

Report Summary

Projected Workload Change Impact:

Overall Impact : 4.64%
Improvement Impact : 30.98%
Regression Impact : -26.35%

SQL Statement Count

SQL Category	SQL Count	Plan Change Count
Overall	43	5
Improved	1	1
Regressed	1	1
Unchanged	30	3
Unsupported	11	0

Top 32 SQL Sorted by Absolute Value of Change Impact on the Workload

object_id	sql_id	Impact on Workload	Execution Frequency	Metric Before	Metric After	Impact on SQL	Plan Change
83	f90zn75aphu4w	30.98%	2828	139319.505304102	41950	69.89%	y
47	0cwuxyv314wcg	-26.35%	18254	981.459680070122	13809	-1306.99%	y
80	csv0xdm9c394t	-.36%	2734	4689.26664228237	5868	-25.14%	n
60	4hbzjyh4p336s	.21%	2818	668.862668559262	10	98.5%	n
76	a8ntu3081hfgw	-.18%	2818	262.609297374024	828	-215.3%	y

Report Summary

Projected Workload Change Impact:

Overall Impact	: 4.64%
Improvement Impact	: 30.98%
Regression Impact	: -26.35%

SQL Statement Count

SQL Category	SQL Count	Plan Change Count
Overall	43	5
Improved	1	1
Regressed	1	1
Unchanged	30	3
Unsupported	11	0

Top 32 SQL Sorted by Absolute Value of Change Impact on the Workload

object_id	sql_id	Impact on Workload	Execution Frequency	Metric Before	Metric After	Impact on SQL	Plan Change
83	f90zn75aphu4w	30.98%	2828	139319.505304102	41950	69.89%	y
47	0cwuxyv314wcg	-26.35%	18254	981.459680070122	13809	-1306.99%	y
80	csv0xdm9c394t	-.36%	2734	4689.26664228237	5868	-25.14%	n
60	4hbzjyh4p336s	.21%	2818	668.862668559262	10	98.5%	n
76	a8ntu3081hfgw	-.18%	2818	262.609297374024	828	-215.3%	y

SQL Details:

Object ID : 47
Schema Name : TPCC
Container Name : Unknown (con_dbid: 344460545)
SQL ID : 0cwuxyv314wcg
Execution Frequency : 18254
SQL Text : SELECT ROWID FROM CUSTOMER WHERE C_W_ID = :B3 AND C_D_ID = :B2 AND C_LAST = :B1

Bind Variables:

1 - (NUMBER): 3
2 - (NUMBER): 7
3 - (VARCHAR2): ESEEINGOUGHT

Execution Statistics:

Stat Name	Impact on Workload	Value Before	Value After	Impact on SQL
elapsed_time	-26.35%	.000981	.013809	-1306.99%
parse_time			.000477	
cpu_time	-106.5%	.000494	.013743	-2681.43%
user_io_time			0	
buffer_gets	-130.82%	253	7252	-2759.03%
cost	-745969.19%	255	1982	-677.25%
reads	0%	0	0	0%
writes	0%	0	0	0%

SQL Details:

Object ID : 47
Schema Name : TPCC
Container Name : Unknown (con_dbid: 344460545)
SQL ID : 0cwuxyv314wcg
Execution Frequency : 18254
SQL Text : SELECT ROWID FROM CUSTOMER WHERE C_W_ID = :B3 AND C_D_ID = :B2 AND C_LAST = :B1

Bind Variables:

1 - (NUMBER): 3
2 - (NUMBER): 7
3 - (VARCHAR2): ESEEINGOUGHT

Execution Statistics:

Stat Name	Impact on Workload	Value Before	Value After	Impact on SQL
elapsed_time	-26.35%	.000981	.013809	-1306.99%
parse_time			.000477	
cpu_time	-106.5%	.000494	.013743	-2681.43%
user_io_time			0	
buffer_gets	-130.82%	253	7252	-2759.03%
cost	-745969.19%	255	1982	-677.25%
reads	0%	0	0	0%
writes	0%	0	0	0%

Execution Plan Before Change:

Plan Hash Value : 612465046

Id	Operation	Name	Rows	Bytes	Cost	Time
0	SELECT STATEMENT				255	
1	SORT ORDER BY		2	92	255	00:00:01
2	TABLE ACCESS BY INDEX ROWID BATCHED	CUSTOMER	2	92	254	00:00:01
3	INDEX RANGE SCAN	CUSTOMER_I1	3000		10	00:00:01

Execution Plan After Change:

Plan Id : 545

Plan Hash Value : 4040750106

Id	Operation	Name	Rows	Bytes	Cost	Time
0	SELECT STATEMENT		2	92	1982	00:00:01
1	SORT ORDER BY		2	92	1982	00:00:01
* 2	TABLE ACCESS FULL	CUSTOMER	2	92	1981	00:00:01

Predicate Information (identified by operation id):

- 2 - filter("C_LAST"=:B1 AND "C_D_ID"=:B2 AND "C_W_ID"=:B3)

Execution Plan Before Change:

Plan Hash Value : 612465046

Id	Operation	Name	Rows	Bytes	Cost	Time
0	SELECT STATEMENT				255	
1	SORT ORDER BY		2	92	255	00:00:01
2	TABLE ACCESS BY INDEX ROWID BATCHED	CUSTOMER	2	92	254	00:00:01
3	INDEX RANGE SCAN	CUSTOMER_I1	3000		10	00:00:01

Execution Plan After Change:

Plan Id : 545

Plan Hash Value : 4040750106

Id	Operation	Name	Rows	Bytes	Cost	Time
0	SELECT STATEMENT		2	92	1982	00:00:01
1	SORT ORDER BY		2	92	1982	00:00:01
* 2	TABLE ACCESS FULL	CUSTOMER	2	92	1981	00:00:01

Predicate Information (identified by operation id):

- 2 - filter("C_LAST"=:B1 AND "C_D_ID"=:B2 AND "C_W_ID"=:B3)

You don't need to connect your app to use SQL Performance Analyzer

You can use SQL Performance Analyzer to test any change to your database

Schema changes may interfere with SQL Performance Analyzer

Further Information

SQL Performance Analyzer

- Blog post: Smooth transition to Autonomous Database using SPA


1
CAPTURE
2
ANALYZE
3
FIX
Fix regressing statements
4
REMEDY

Fixing Statements

Most cloud offerings have access to a number of tools:

- SQL Tuning Advisor
- Real-time SQL Monitoring
- SQL Access Advisor

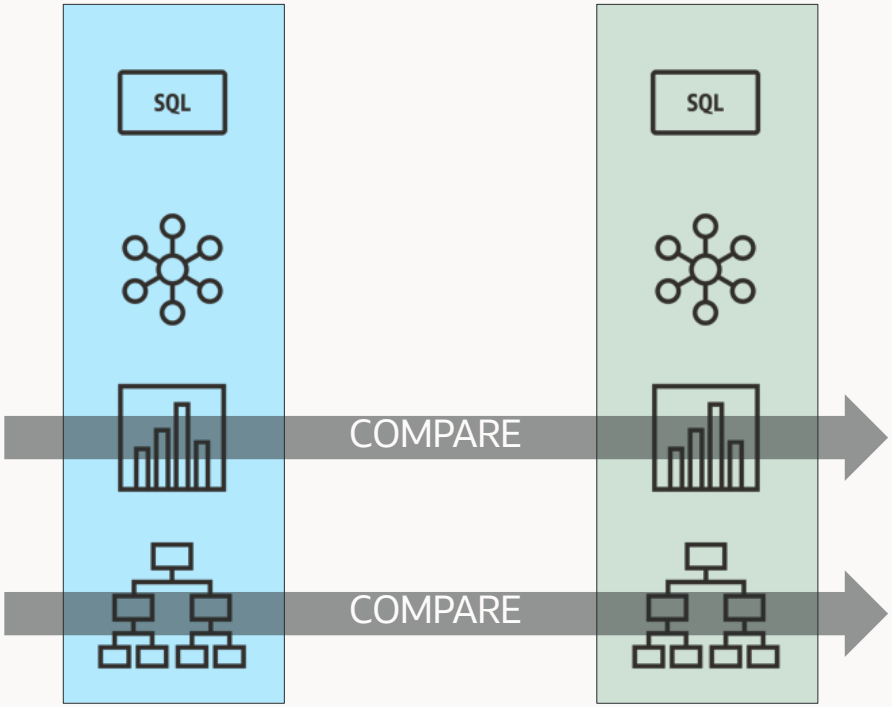
Perhaps you even have access to a DBA

You could also ask ChatGPT

What is the best fix?

- Does it have any side effects?

Continuous Improvement



Continuous Improvement

Implement change

Test execute

Check outcome

Repeat

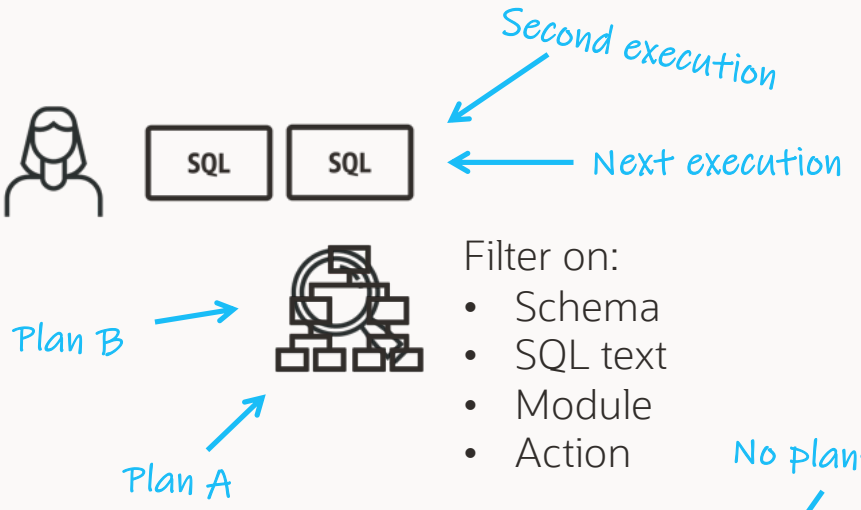

1
CAPTURE
2
ANALYZE
3
FIX
4
REMEDY
Stabilize performance
using
SQL Plan Management

SQL Plan Management is the best tool to ensure plan stability

```
-- Toggles creation of SQL plan baselines for all repeatable statements
-- Usually, not recommended to capture and create baselines for all statements

alter system set optimizer_capture_sql_plan_baselines=true;
```

SQL Plan Management

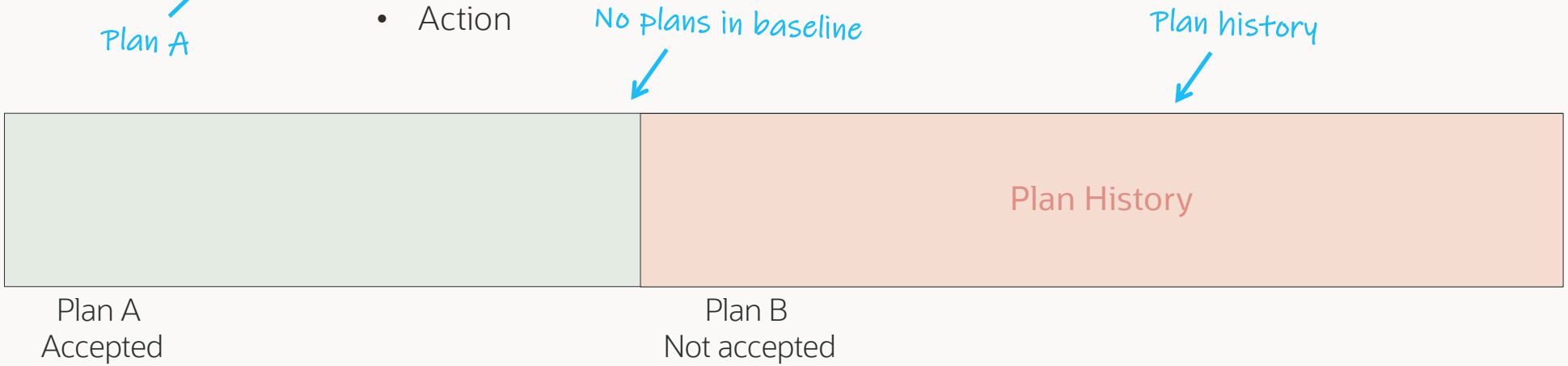


Filter on:

- Schema
- SQL text
- Module
- Action

Something changed:

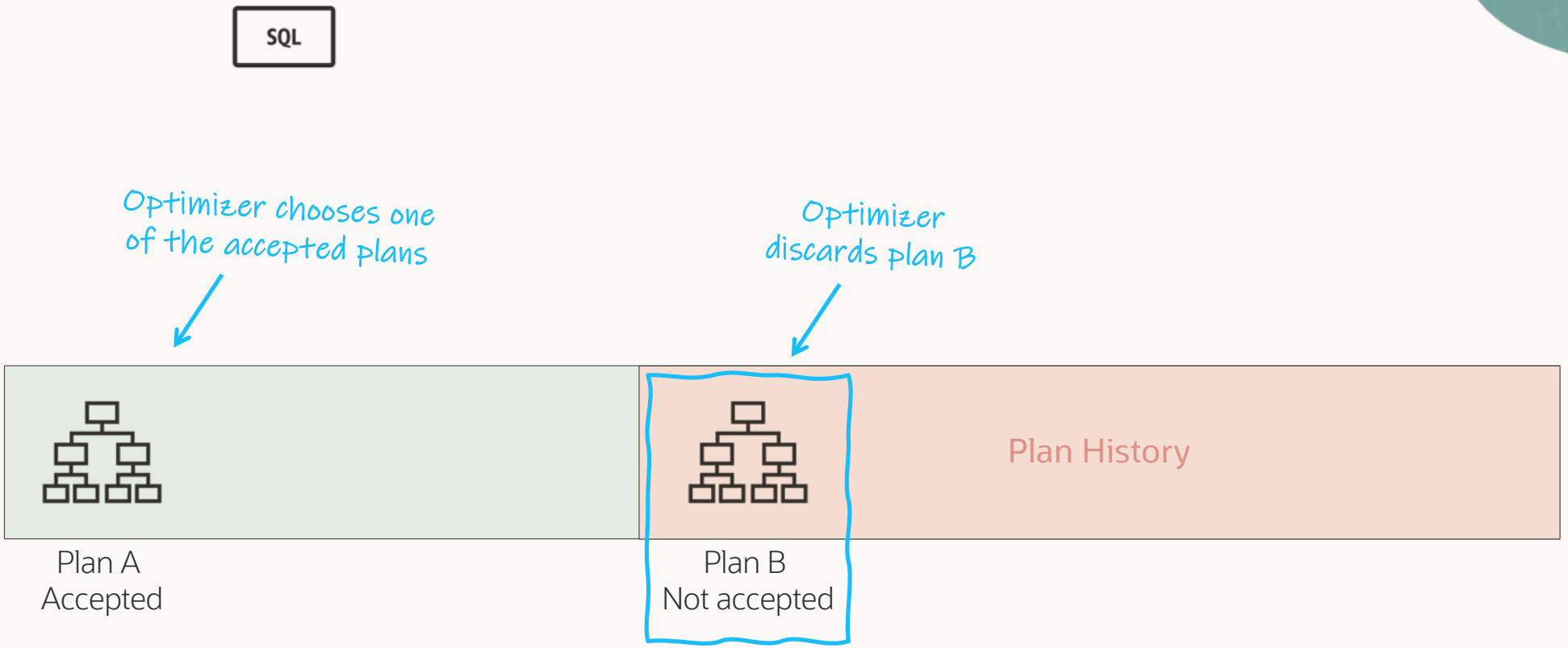
- New statistics
- New parameters
- Upgrade



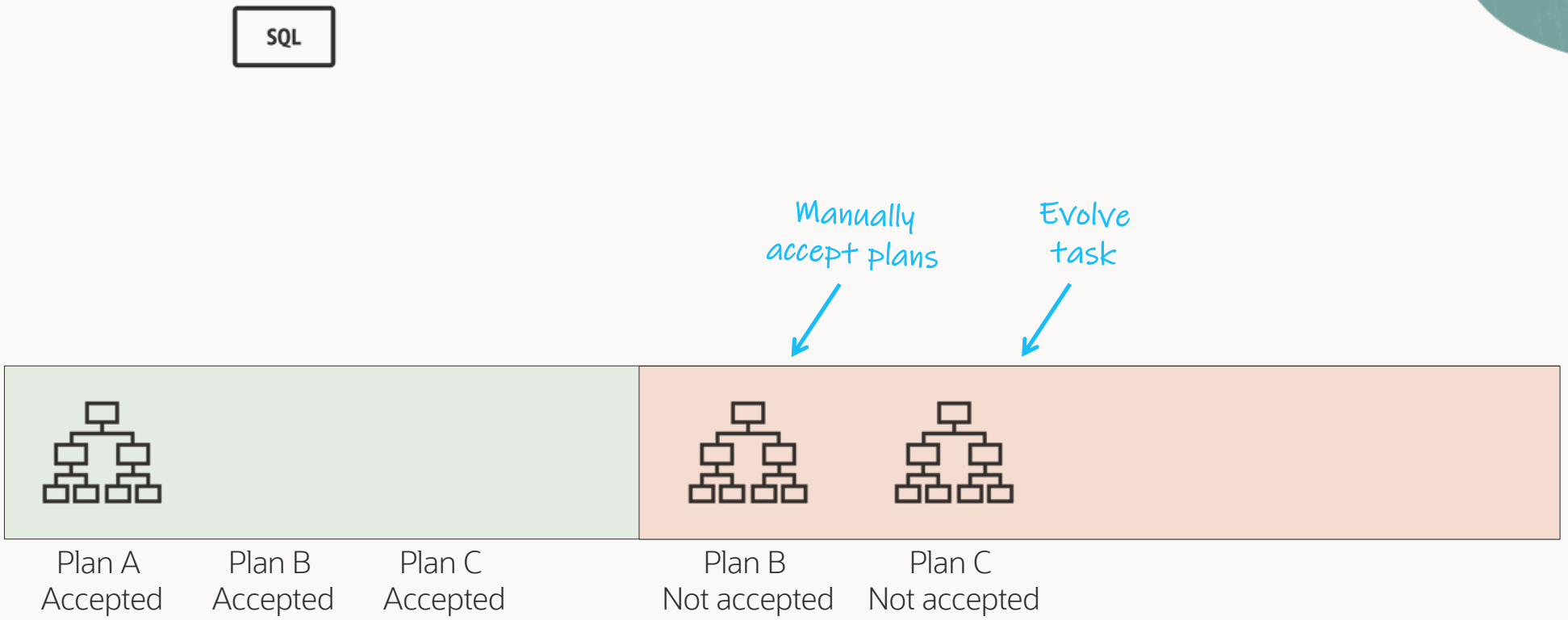
```
-- Restricts the optimizer to only use plans that are accepted
-- This is the default value

alter system set optimizer_use_sql_plan_baselines=true;
```


SQL Plan Management



SQL Plan Management



```
-- The database stores plans in the plan history for 53 weeks.
-- This might consume a lot of space. Consider lowering the limit.

exec dbms_spm.configure('plan_retention_weeks', 5);
```


You can manually create plan baselines for specific statements

```
-- Load all plans from a SQL tuning set into plan baselines
-- Plans are accepted automatically without test execution

var cnt number;
exec :cnt := DBMS_SPM.LOAD_PLANS_FROM_SQLSET(
                basic_filter => 'sql_id=''0cwuxyv314wcg''',
                ...
             );
```


Plan baselines are transportable; create in test, use in production

```
-- Pack baselines from "problematic" statements into staging table
exec DBMS_SPM.PACK_STGTAB_BASELINE(sql_handle => SQL_130f372d9ffe4df9, ...);
```


```
-- Pack baselines from "problematic" statements into staging table
exec DBMS_SPM.PACK_STGTAB_BASELINE(sql_handle => SQL_130f372d9ffe4df9, ...);
exec DBMS_SPM.PACK_STGTAB_BASELINE(sql_handle => SQL_51dc7232adc62849, ...);
exec DBMS_SPM.PACK_STGTAB_BASELINE(sql_handle => SQL_3f38bc33ae7086c9, ...);
```


```
-- Pack baselines from "problematic" statements into staging table
exec DBMS_SPM.PACK_STGTAB_BASELINE(sql_handle => SQL_130f372d9ffe4df9, ...);
exec DBMS_SPM.PACK_STGTAB_BASELINE(sql_handle => SQL_51dc7232adc62849, ...);
exec DBMS_SPM.PACK_STGTAB_BASELINE(sql_handle => SQL_3f38bc33ae7086c9, ...);

...

-- After production migration, import plan baselines to fix regressions
exec DBMS_SPM.UNPACK_STGTAB_BASELINE( ... );
```


Let's automate it

Automatic SQL Plan Management

- Background job (30 min)
- Finds resource-intensive SQLs
- Evaluate execution statistics against previous executions

Automatic SQL Plan Management

- Creates a plan baseline with the best plan
- Next execution uses an accepted plan

- Enterprise Edition, RU 19.22
- Exadata 19c
- Autonomous AI Database (on by default)

Real-Time SQL Plan Management

- User executes SQL
- Optimizer chooses a different execution plan
- Executes with new plan

Real-Time SQL Plan Management

- Compares execution statistics with history
- Comparison is performed in the user foreground session

Real-Time SQL Plan Management

- Creates a plan baseline with the best plan
- Next execution uses an accepted plan

- Oracle AI Database 26ai Enterprise Edition
- Autonomous AI Database 19c (on by default)

Here's one for the Top Gun DBA

```
--Loads all known plans from cursor cache, AWR,
--and automatic SQL tuning sets into a SQL plan baseline.
--Use Evolve Advisor to find the best plan and mark that as accepted.

var report clob;
exec :report := dbms_spm.add_verified_sql_plan_baseline('<sql_id>');
select :report report from dual;
```


Who said literals?

- SQL Plan Management is not a good fit for an application that doesn't use bind variables

```
--Application using literals creates many distinct statements
--You'd get 4 plan baselines
select * from sales where order_id=42;
select * from sales where order_id=56;
select * from sales where order_id=101;
select * from sales where order_id=220;
```


```
--Application using literals creates many distinct statements
--You'd get 4 plan baselines
select * from sales where order_id=42;
select * from sales where order_id=56;
select * from sales where order_id=101;
select * from sales where order_id=220;


--Ideally change the application to use literals
--You'd get only 1 plan baseline
select * from sales where order_id=:b1;
```


Generally, avoid setting CURSOR_SHARING=FORCE

- Advice from Real-World Performance Group

Use SQL Profiles

- Part of Tuning Pack
- Included in most cloud offerings

Further Information

SQL Plan Management

- Blog post: SQL Plan Management Cheat Sheet – Part 1
- Blog post: SQL Plan Management Cheat Sheet – Part 2
- Blog post: What is automatic SQL plan management and why should you care?
- Blog post: What is Real-time SQL plan Management?
- My Oracle Support: Things to Consider to Avoid SQL Plan Management (SPM) Related Problems on 19c (KB139467)


1
CAPTURE
2
ANALYZE
3
FIX
4
REMEDY

Hands-on Lab

Hitchhiker's Guide for upgrading to Oracle AI Database 26ai

It's better to fail in our lab, than in production

Access lab on Oracle Live Labs

Oracle Autonomous AI Database

You've been managing your database for many years.

Performance is tailored to your application,
your hardware and your needs.

Oracle Autonomous AI Database is an Oracle AI Database

.... but it is a different Oracle AI Database

Understand Your Environments

Hardware

- Runs on Exadata
- Smart Scan makes full table scans more attractive
- Analytic indexes might no longer be needed
- Other Exadata capabilities

Parameters

- Restrictions on parameters
- Less influence on optimizer
- Can't set `OPTIMIZER_FEATURES_ENABLE`
- Stick with the defaults in Autonomous AI Database

Statistics | Global Preferences

- Set with `DBMS_STATS.SET_GLOBAL_PREFS`
- Oracle recommends using default global statistics preferences
- Also applies to Oracle Autonomous AI Database
- Data Pump does not export them

Statistics | Table Preferences

- Set with `DBMS_STATS.SET_TABLE_PREFS`
- ZDM and DMS set `EXCLUDE=STATISTICS`
- Not moved to Autonomous AI Database

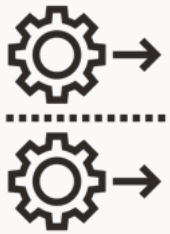
Does this problem occur in Oracle Autonomous AI Database?

- You often use statistics preferences to solve a particular problem

Statistics | Extended Statistics

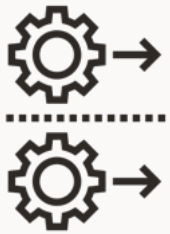
- Set with `DBMS_STATS.CREATE_EXTENDED_STATS`
- ZDM and DMS set `EXCLUDE=STATISTICS`
- Not moved to Autonomous AI Database

Parallel



- Do you set a parallel degree on your objects?
- Do you use **PARALLEL** hints?
- Autonomous AI Database applies parallel differently

Parallel



- Data Pump sets the object parallel degree according to the source
- Consider removing after migration

Collect and use this information as input to your testing/troubleshooting in ADB

Be open to changes…

Understand Autonomous AI Database

Autonomous AI Database Dedicated gathers stats like any other Exadata database

```
select   origin, start_time
from     dba_auto_stat_executions
order by start_time;

ORIGIN                  START_TIME
_____________________ ____________________________________________
AUTO_TASK               19-JUN-25 10.00.02.475575000 PM GMT
...
HIGH_FREQ_AUTO_TASK     20-JUN-25 08.35.38.149345000 AM GMT
HIGH_FREQ_AUTO_TASK     20-JUN-25 08.50.38.568564000 AM GMT
HIGH_FREQ_AUTO_TASK     20-JUN-25 09.05.39.171647000 AM GMT
HIGH_FREQ_AUTO_TASK     20-JUN-25 09.20.39.558415000 AM GMT
```

Autonomous AI Database Dedicated

```
select   operation, target, start_time, job_name
from     DBA_OPTSTAT_OPERATIONS
order by start_time;
```

OPERATION	TARGET	START_TIME	JOB_NAME
gather_database_stats (auto)	AUTO	19-JUN-25 10.00.02.295346000 PM GMT	ORA$AT_OS_OPT_SY_349
purge_stats		19-JUN-25 10.00.50.552518000 PM GMT	ORA$AT_OS_OPT_SY_349
purge_stats		19-JUN-25 10.00.51.783925000 PM GMT	ORA$AT_OS_OPT_SY_349
...			
gather_database_stats (auto)	AUTO	20-JUN-25 08.13.29.267822000 AM GMT	ORA$_ATSK_AUTOSMT
gather_database_stats (auto)	AUTO	20-JUN-25 08.28.29.846908000 AM GMT	ORA$_ATSK_AUTOSMT
gather_database_stats (auto)	AUTO	20-JUN-25 08.43.30.349803000 AM GMT	ORA$_ATSK_AUTOSMT
gather_database_stats (auto)	AUTO	20-JUN-25 08.58.30.842146000 AM GMT	ORA$_ATSK_AUTOSMT
gather_database_stats (auto)	AUTO	20-JUN-25 09.13.31.355042000 AM GMT	ORA$_ATSK_AUTOSMT
gather_database_stats (auto)	AUTO	20-JUN-25 09.28.31.933039000 AM GMT	ORA$_ATSK_AUTOSMT
gather_database_stats (auto)	AUTO	20-JUN-25 09.43.32.600796000 AM GMT	ORA$_ATSK_AUTOSMT
gather_database_stats (auto)	AUTO	20-JUN-25 09.58.32.941213000 AM GMT	ORA$_ATSK_AUTOSMT

Autonomous AI Database Serverless introduces new ways of gathering stats

AI Lakehouse Serverless does not use Automatic Statistics Gathering

AI Lakehouse Serverless

- In DWH after loading,
  DMLs are atypical or infrequent
- Gathering stats on such a workload
  is considered wasting resources
- Automatic Statistics Gathering
  is disabled

- Gathers statistics automatically
  following most direct load operations
- If you use conventional DML,
  consider gathering statistics manually

AI Lakehouse Serverless

Transaction Processing Serverless uses the regular Automatic Stats Gathering job

Dictionary Views

- The usual auto tasks views are disabled in Autonomous AI Database Serverless
  - `DBA_OPTSTAT%`
  - `DBA_AUTOTASK%`

- Use other dictionary views for information on stats gathering, like
  - `DBA_AUTO_STAT_EXECUTIONS`
  - `DBA_AUTO_STAT_OBJ_GATHER_DETAILS`

Transaction Processing Serverless

Automatic Statistics Gathering also updates stale dictionary stats

- For additional runs, file a service request

Autonomous AI Database Serverless creates histograms on all columns

- Serverless uses `FOR ALL COLUMNS SIZE 254`
- Dedicated uses `FOR ALL COLUMNS SIZE AUTO` (default)

```
--If stats gathering takes too long on a specific table due to histograms
--you can disable histograms using a table-specific preference

exec dbms_stats.set_table_prefs(
    ownname => 'SCOTT',
    tabname => 'EMP',
    pname   => 'METHOD_OPT',
    pvalue  => 'FOR ALL COLUMNS SIZE 1');
```

Autonomous AI Database Serverless

High-frequency Statistics Gathering complements Automatic Statistics Gathering

High-frequency Statistics Gathering

- Automated task that only gathers stale statistics
- Reduces the period where an objects might have stale statistics
- Helps avoiding cardinality misestimates on volatile tables

Autonomous AI Database All

```
select
   dbms_stats.get_prefs ('AUTO_TASK_STATUS') high_freq,
   dbms_stats.get_prefs ('AUTO_TASK_INTERVAL') run_interval,
   dbms_stats.get_prefs ('AUTO_TASK_MAX_RUN_TIME') max_run_time
from dual;
```

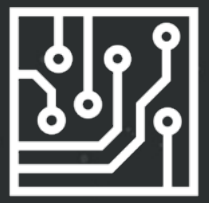
```
HIGH_FREQ RUN_INTERVAL MAX_RUN_TIME
--------- ------------ ------------
ON        900          3600
```

Autonomous AI Database All

```
--Shows the last result of auto stats gathering
--Works in ADB-S also
select status, elapsed_time, last_schedule_time
from   dba_autotask_schedule_control
where  task_name='Auto Statistics Management Task';
```

```
STATUS    ELAPSED_TIME LAST_SCHEDULE_TIME
--------- ------------ ---------------------------
SUCCEEDED            13 2026-06-18T14:05:34.942Z
```

Autonomous AI Database All



Real-Time Statistics updates the most essential stats during DML

- Disabled by default

Real-Time Statistics

- Augments traditional statistics collection
- When a DML operation is currently modifying a table, the most essential statistics are computed
- Does not gather statistics, but updates existing statistics, like
  - Number of rows
  - Number of table blocks
  - High/low values
- Recommendation: Leave at default value: Disabled

Autonomous AI Database
All

Statistics

- Manage Optimizer Statistics on Autonomous AI Database Serverless
- High Performance Features in Autonomous AI Database on Dedicated Exadata Infrastructure
- How to Use High-frequency Statistics Gathering
- Configuring High-Frequency Automatic Optimizer Statistics Collection
- Real-Time Statistics
- OPTIMIZER_REAL_TIME_STATISTICS

Autonomous AI Database automatically applies parallel degree

Parallel

The connection service defines the parallel capabilities:

- `HIGH` and `MEDIUM` has automatic parallelism
- `TP` and `LOW` has no parallelism
- `TPURGENT` can manually set parallelism

```
sqlplus appuser/password@tpurgent
```

```
sqlplus appuser/password@tpurgent
```

```
alter session set optimizer_ignore_hints=false;
alter session set optimizer_ignore_parallel_hints=false;
```


```
sqlplus appuser/password@tpurgent


alter session set optimizer_ignore_hints=false;
alter session set optimizer_ignore_parallel_hints=false;


select /*+ parallel */ ...
```


Parallel

- Database Service Names for Autonomous AI Database

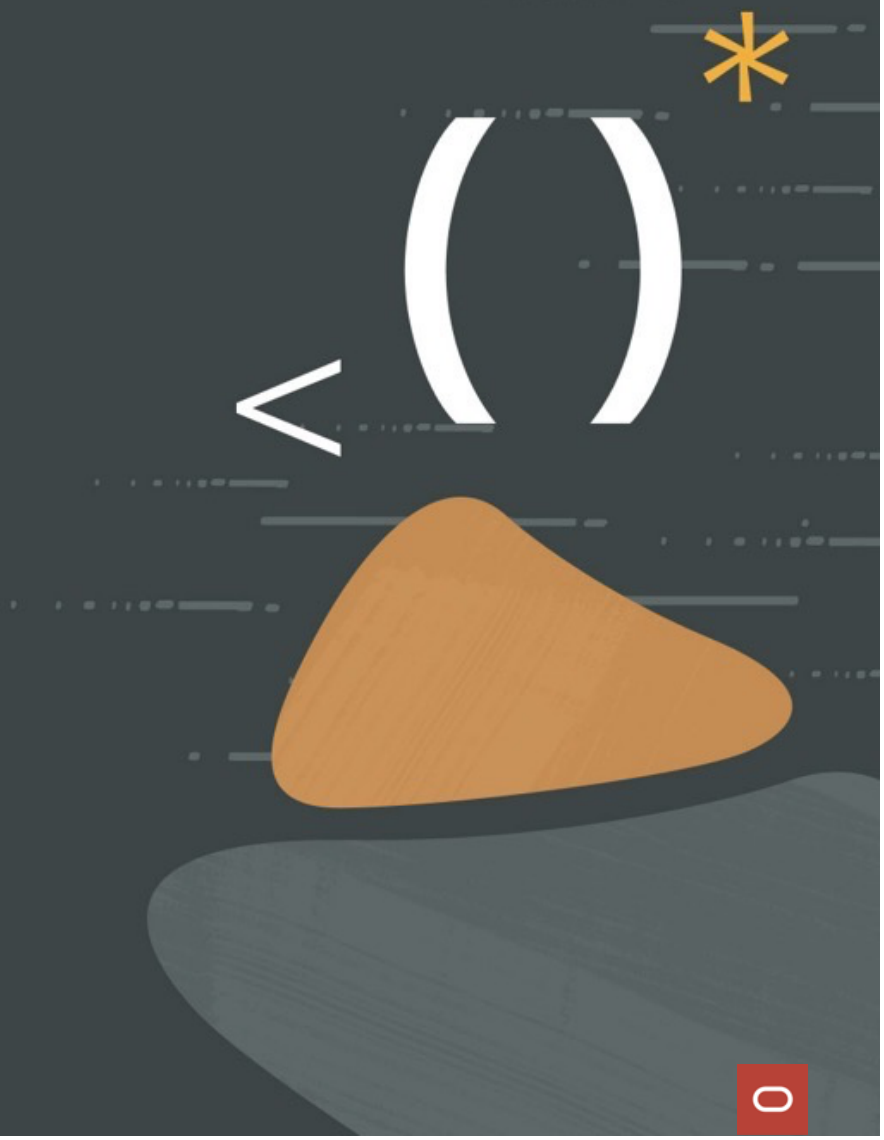
Autonomous AI Lakehouse ignores optimizer hints - including PARALLEL

- Applies to Serverless and Dedicated

```
select name, value, isses_modifiable
from   v$parameter
where  name like 'optimizer_ignore%';

NAME                                 VALUE ISSES_MODIFIABLE
------------------------------------ ----- ----------------
optimizer_ignore_hints               TRUE              TRUE
optimizer_ignore_parallel_hints      TRUE              TRUE
```

What To Do



In source database, capture workload information into SQL Tuning Set

Gather at least a full month of workload data

- Assist in testing your database
- Useful in solving post-migration performance problems

Use SQL Performance Analyzer to test performance of your workload

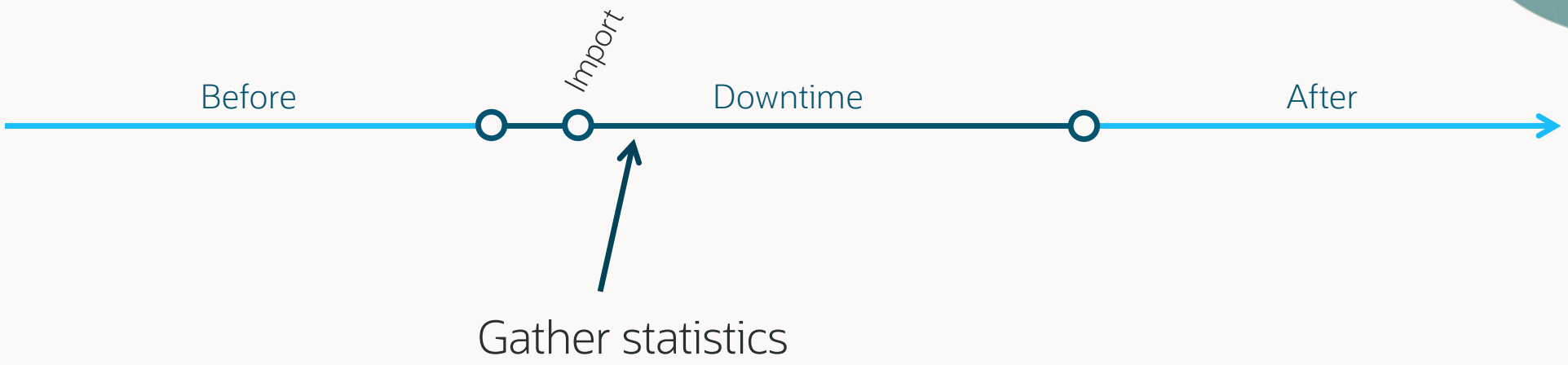
Further Information

SQL Tuning Set

- Blog post: Oracle SQL Tuning Sets (STS) – The foundation for SQL Tuning
- Blog post: What is the automatic SQL tuning set?

The No-Effort Solution

Offline Migration



```
begin
 dbms_stats.gather_schema_stats(
      ownname => 'SH',
      options => 'GATHER',
      cascade => TRUE,
      degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```

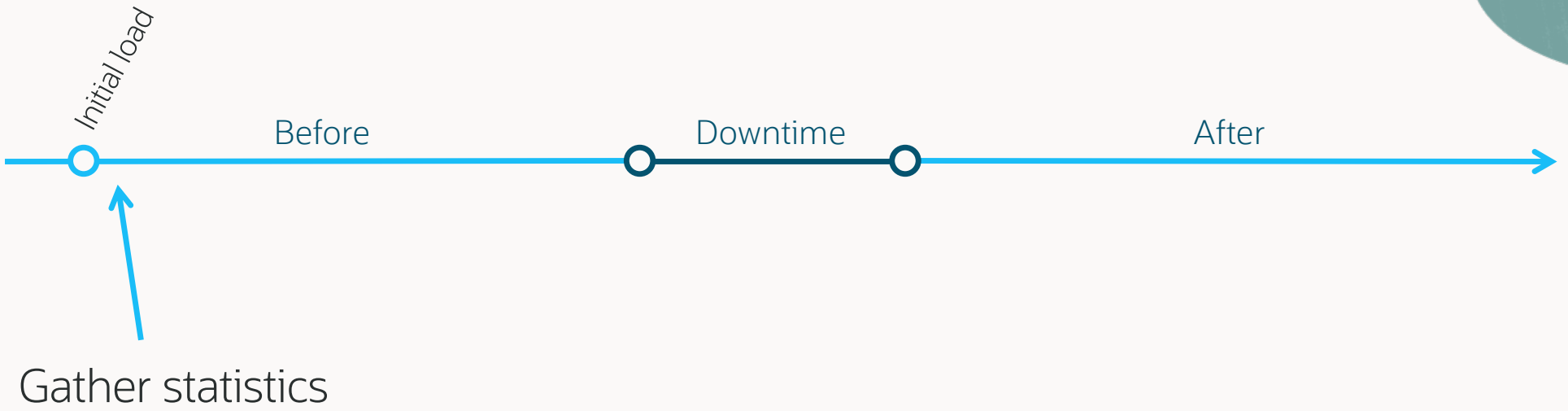

```
begin
 dbms_stats.gather_schema_stats(
      ownname => 'SH',
      options => 'GATHER',
      cascade => TRUE,
      degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```


```
begin
 dbms_stats.gather_schema_stats(
      ownname => 'SH',
      options => 'GATHER',
      cascade => TRUE,
      degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```


```
begin
 dbms_stats.gather_schema_stats(
      ownname => 'SH',
      options => 'GATHER',
      cascade => TRUE,
      degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```


```
begin
 dbms_stats.gather_schema_stats(
      ownname => 'SH',
      options => 'GATHER',
      cascade => TRUE,
      degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```


Online Migration

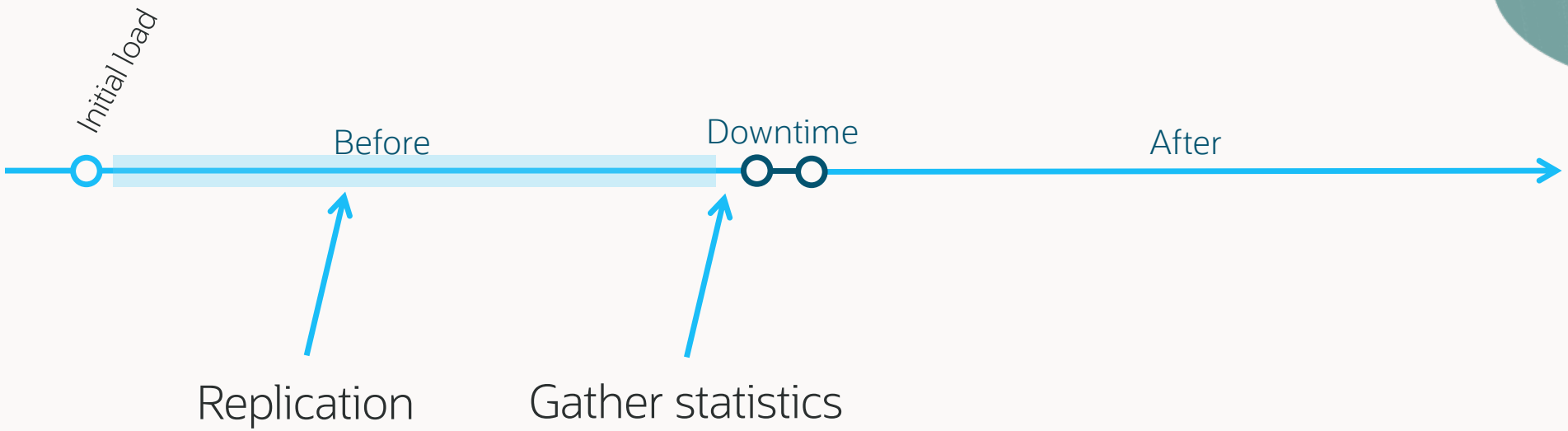


```
begin
 dbms_stats.gather_schema_stats(
      ownname => 'SH',
      options => 'GATHER',
      cascade => TRUE,
      degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```


```
begin
 dbms_stats.gather_schema_stats(
      ownname => 'SH',
      options => 'GATHER',
      cascade => TRUE,
      degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```


Replication changes your data and make statistics go stale

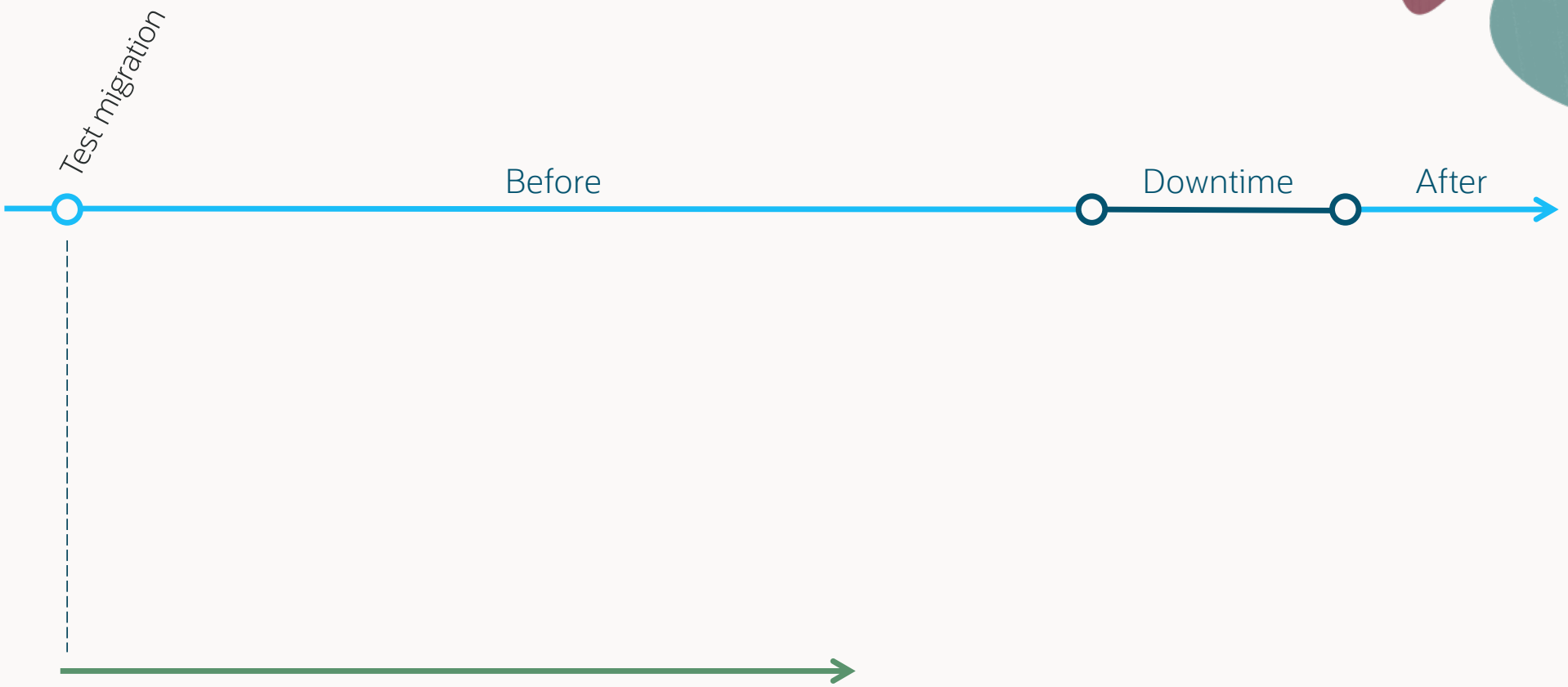
Online Migration



```
begin
 dbms_stats.gather_schema_stats(
      ownname => 'SH',
      options => 'GATHER AUTO',
      cascade => TRUE,
      degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```


The Advanced Solution

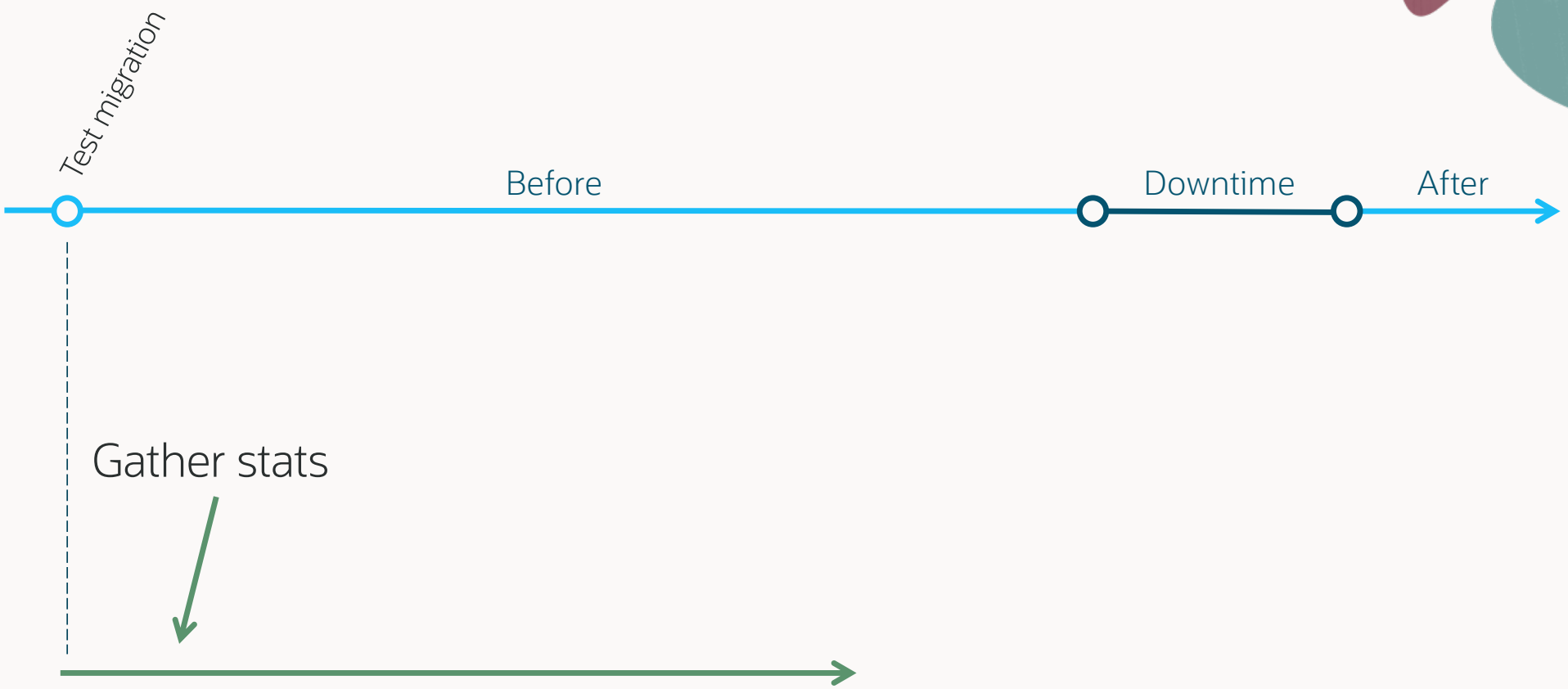
The Advanced Solution



Your test data must be an exact copy of the production data

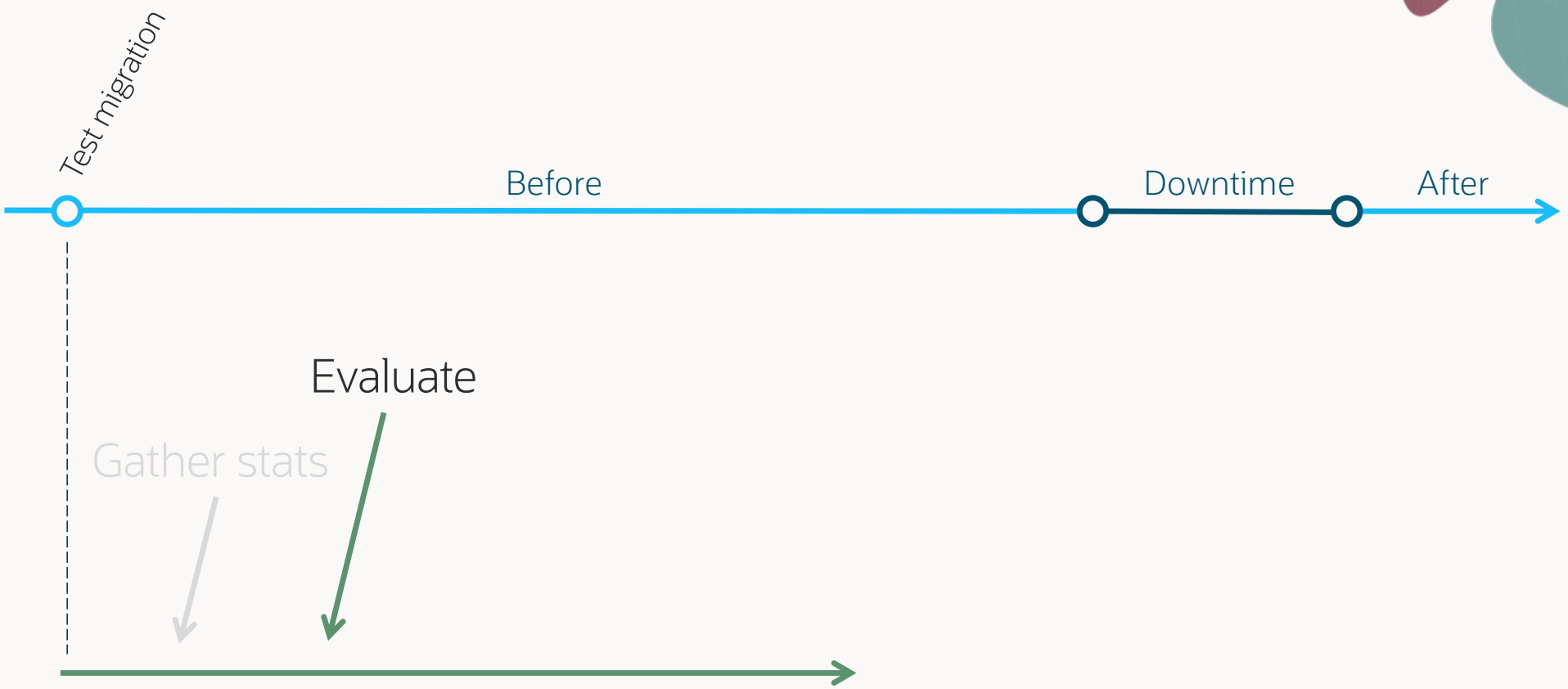
- Must be recent
- Only a few days old, not weeks

The Advanced Solution



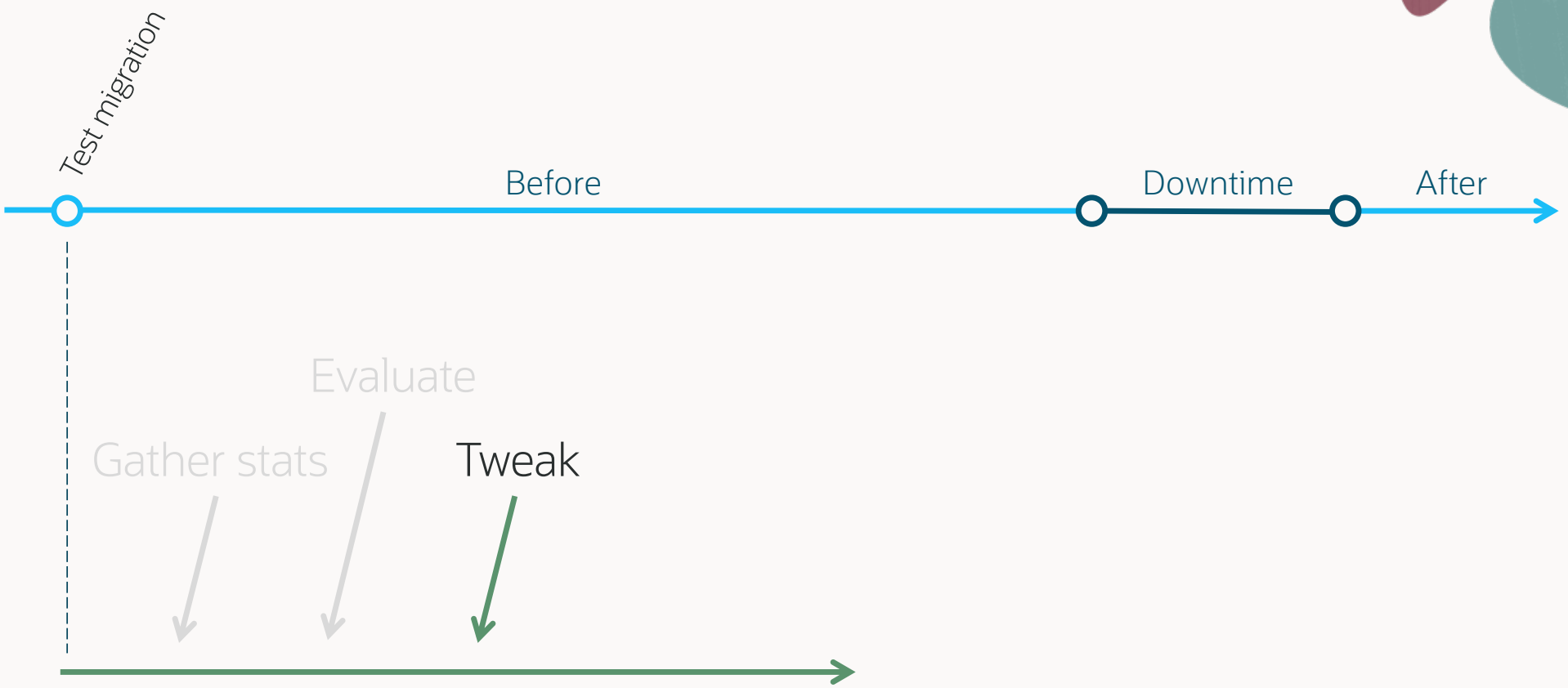
```
begin
 dbms_stats.gather_schema_stats(
      ownname => 'SH',
      options => 'GATHER',
      cascade => TRUE,
      degree  => DBMS_STATS.AUTO_DEGREE);
end;
/
```


The Advanced Solution

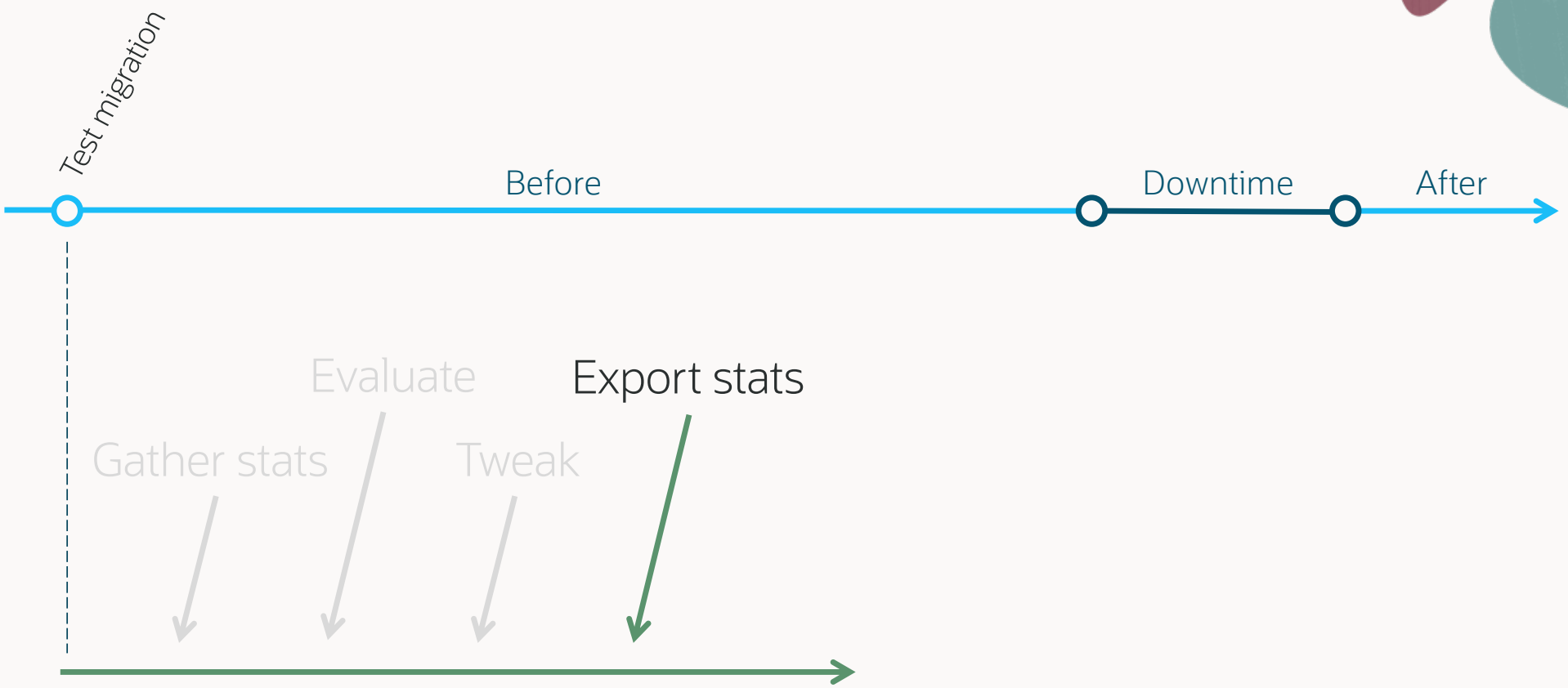


Use SQL Performance Analyzer to test workload from SQL Tuning Set

The Advanced Solution



The Advanced Solution

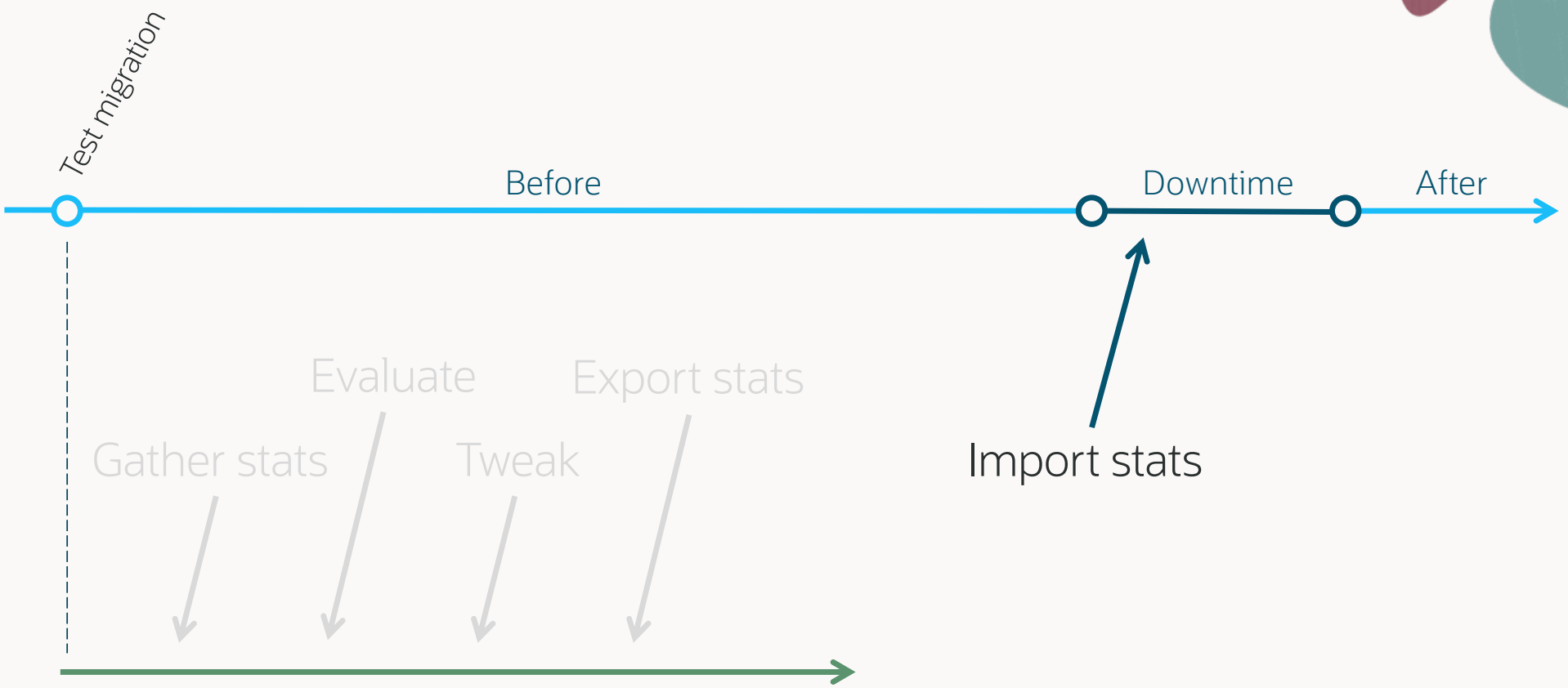


```
begin
    dbms_stats.export_schema_stats(
        ownname => 'APPUSER',
        stattab => 'DBSTATS');
end;
/
```


```
begin
   dbms_stats.export_schema_stats(
      ownname => 'APPUSER',
      stattab => 'DBSTATS');
end;
/
```

```
expdp ... tables=APPUSER.DBSTATS
```


The Advanced Solution

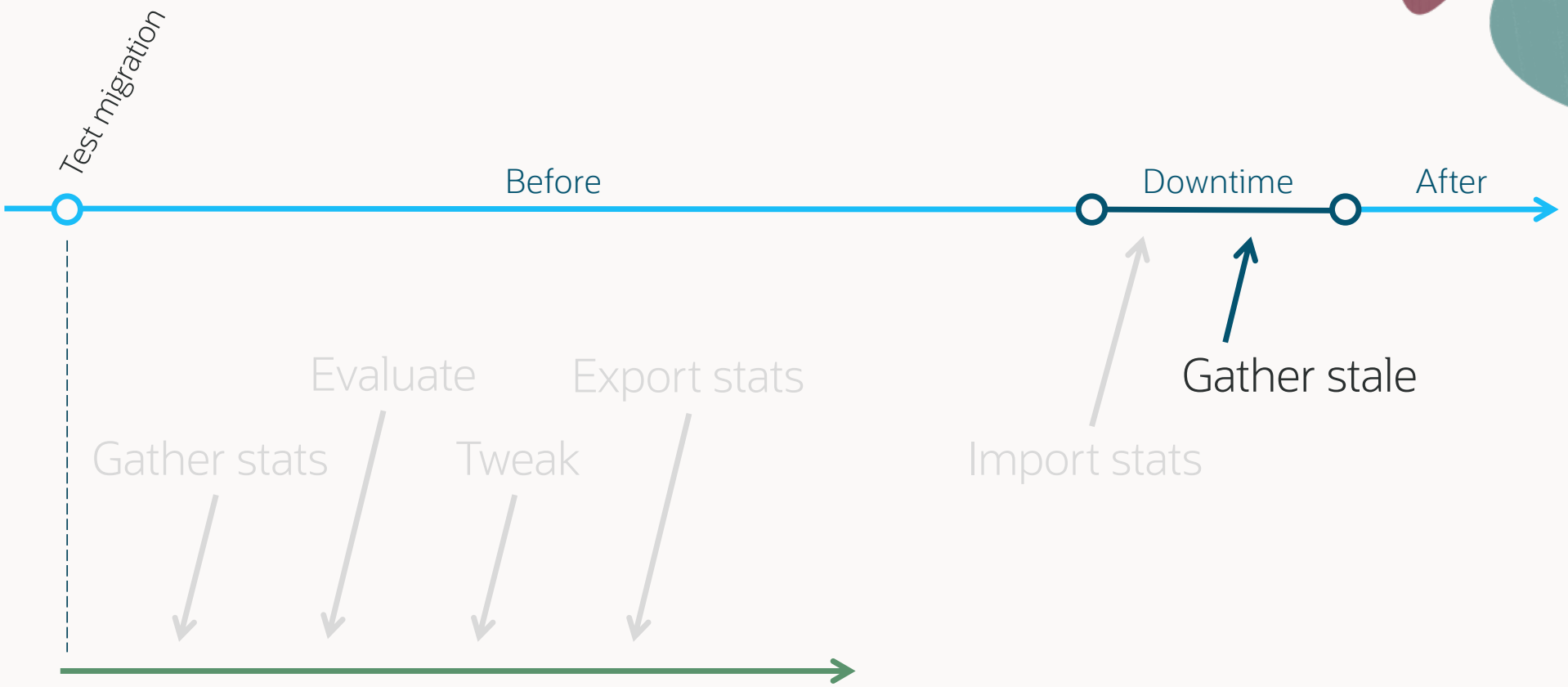


```
impdp ... tables=APPUSER.DBSTATS
```

```
impdp ... tables=APPUSER.DBSTATS
```

```
begin
    dbms_stats.import_schema_stats(
        ownname => 'APPUSER',
        stattab => 'DBSTATS');
end;
/
```


The Advanced Solution

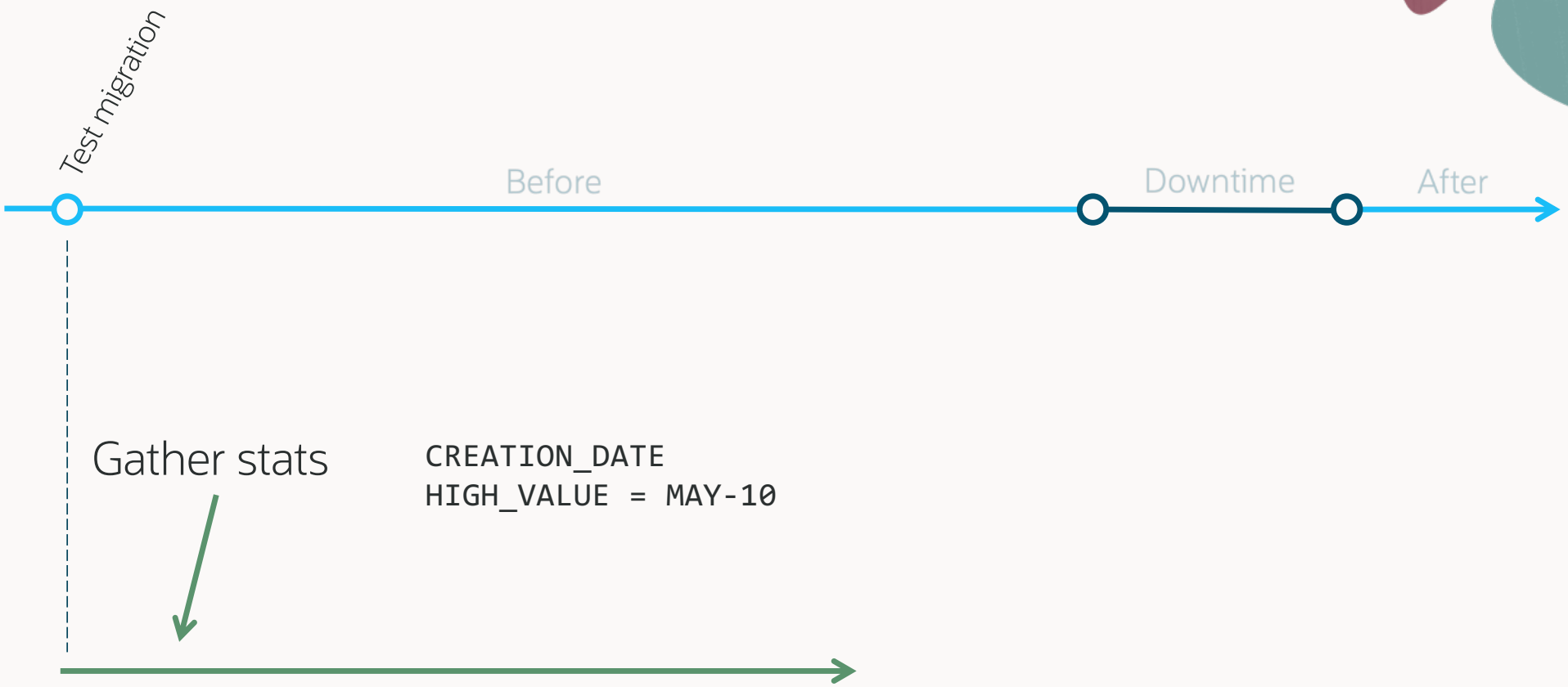


Imported statistics are considered current

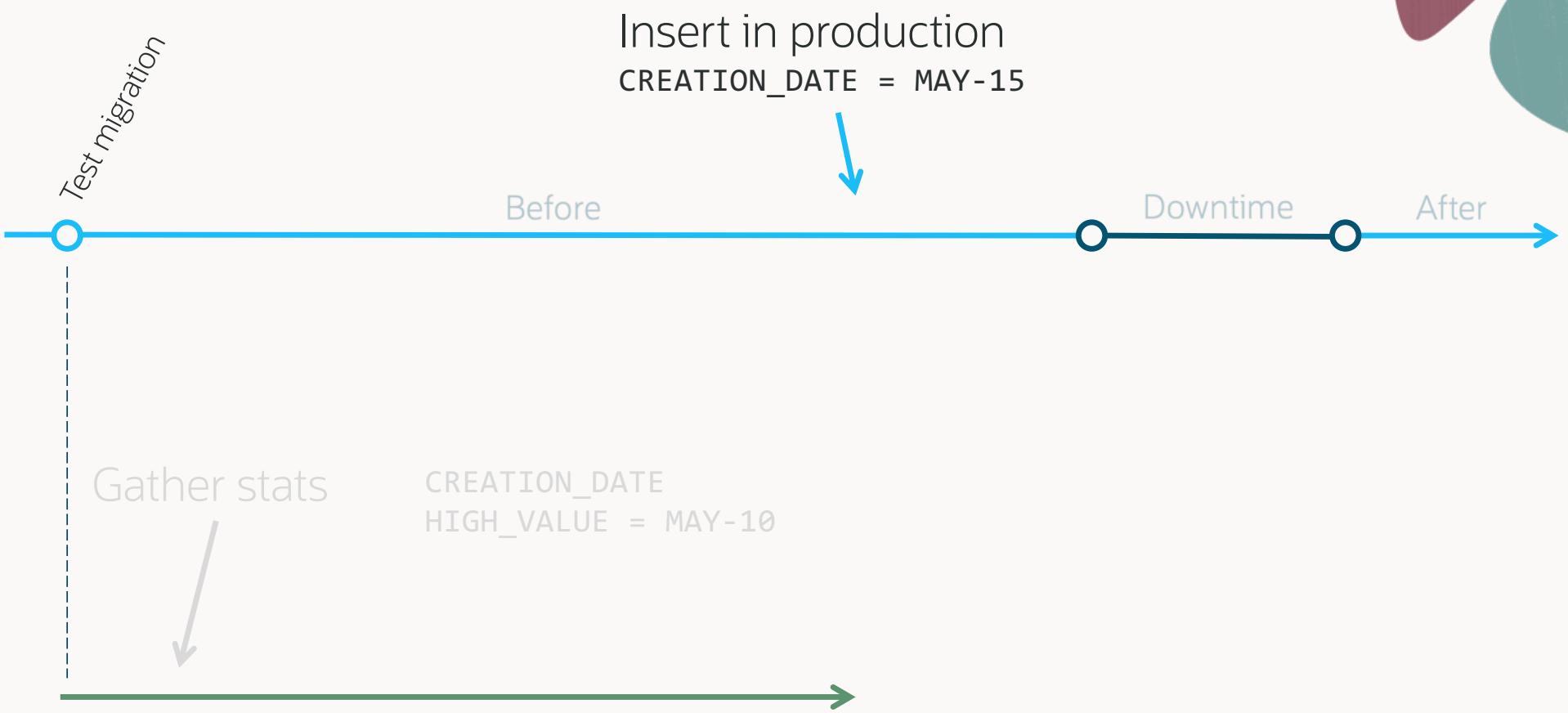
- Watch for out-of-range cardinality misestimates

```
create table sales (
    ...
    creation_date       DATE,
    ...
);
```

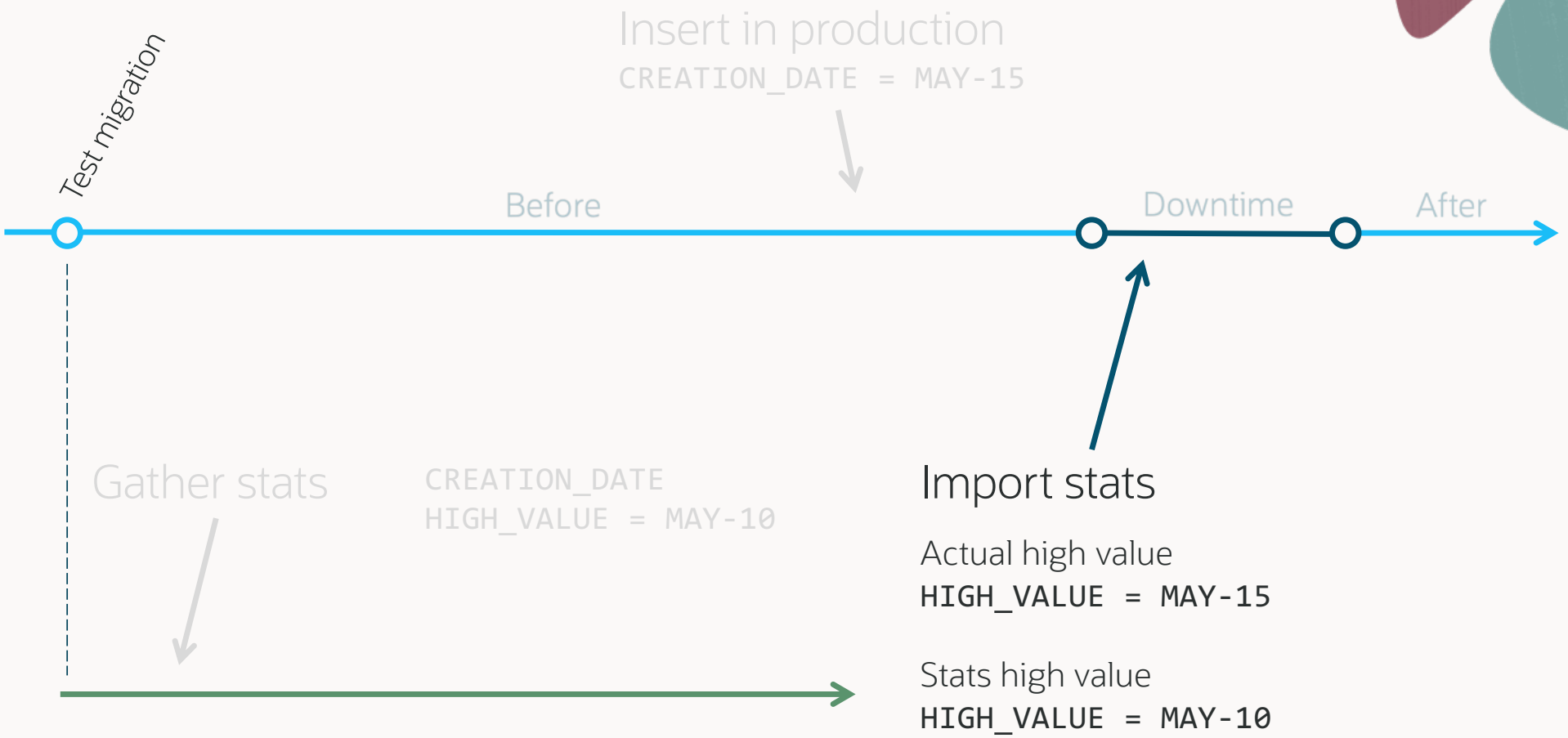
Out-of-range Misestimates



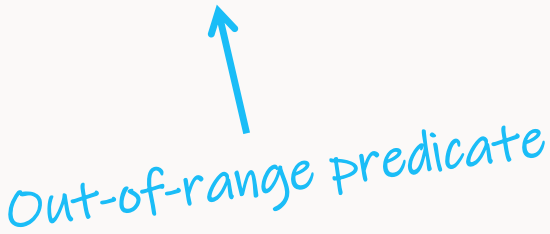
Out-of-range Misestimates



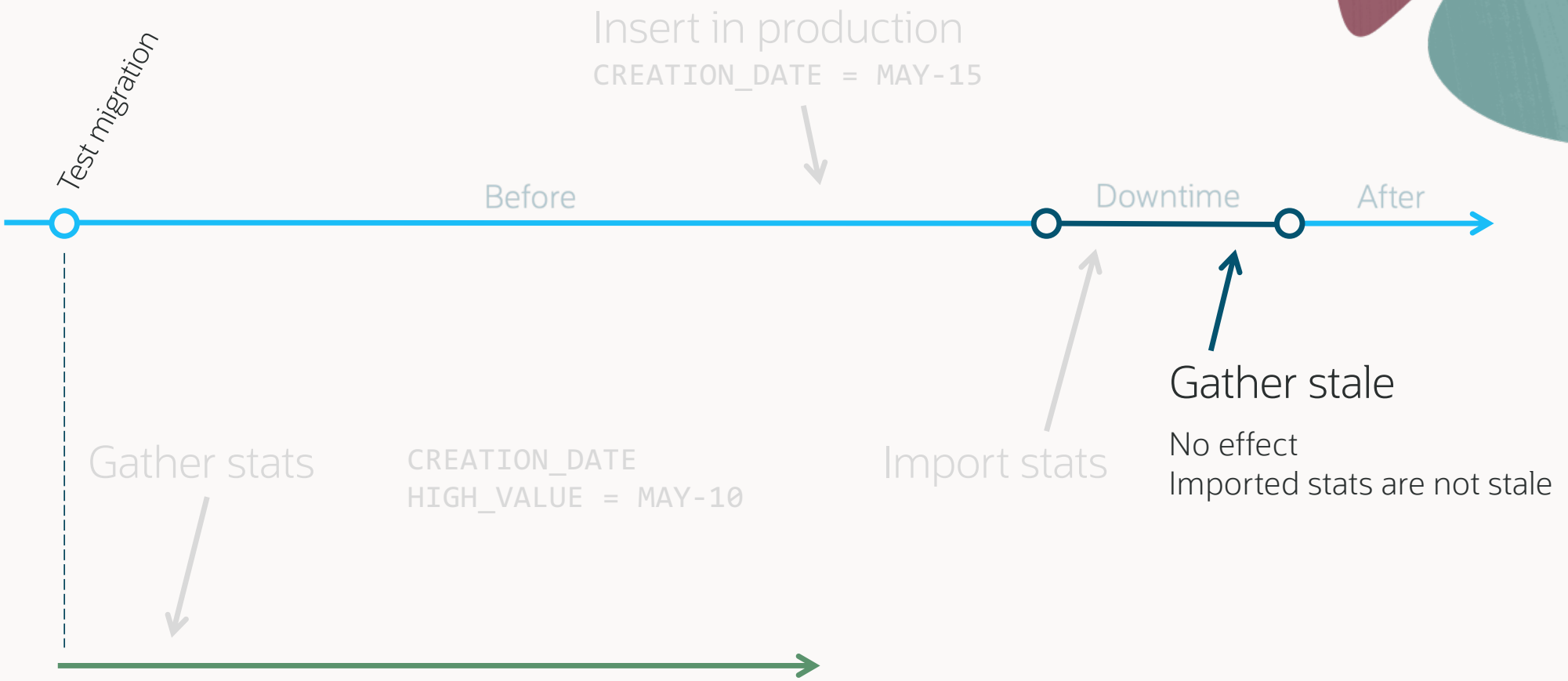
Out-of-range Misestimates



```
select *
from   sales
where  creation_date = MAY-15
```



Out-of-range Misestimates



Out-of-range Misestimates

Techniques to help avoid this:

- Dynamic sampling
- Real-time statistics
- Extended statistics

But nothing beats up-to-date statistics!

```
--Manually gather statistics on tables where queries
--are prone to out-of-range cardinality misestimates

exec dbms_stats.gather_table_stats(
      ownname => 'APPUSER',
      tabname => 'SALES',
      options => 'GATHER',
      degree  => DBMS_STATS.DEFAULT_DEGREE
);
```


Statistics

- How to Export and Import Statistics Faster Using DBMS_STATS in Parallel
- How to Gather Optimizer Statistics Fast!

Autonomous AI Databases collect statistics during the import

Statistics | Gather During Load

During import ADB gathers statistics

- Happens automatically during Data Pump loads
  - But only with direct path loads
  - So, Data Pump avoids external table loads
  - Which sometimes is faster

- Recommendation:
  - Disable stats gathering during import
  - Gather afterwards

```
-- Disable gather stats during load in Data Pump

impdp ... data_options=disable_stats_gathering
```


```
-- Disable gather stats during load in Zero Downtime Migration (ZDM)
-- Add parameter to the ZDM response file

DATAPUMPSETTINGS_DATAPUMPPARAMETERS_DISABLESTATSGATHERING = TRUE
```


```
-- Via Data Pump API

DBMS_DATAPUMP.SET_PARAMETER(
  handle => h1,
  name   => 'DATA_OPTIONS',
  value  => DBMS_DATAPUMP.KU$_DATAOPT_DISABLE_STATS_GATH);
```

Choosing the Right Solution

Most databases will be fine with the No-Effort solution

Use the Advanced solution only when needed

Advanced Solution

Suitable for:

- Critical databases
- Statistics gathering takes too long
- Non-default or custom statistics required
- Edge cases
- Testing shows a need for tweaks and customizations

Autonomous AI Database has a lot of automation to improve performance

You could do nothing, and ADB would sort it out after a while

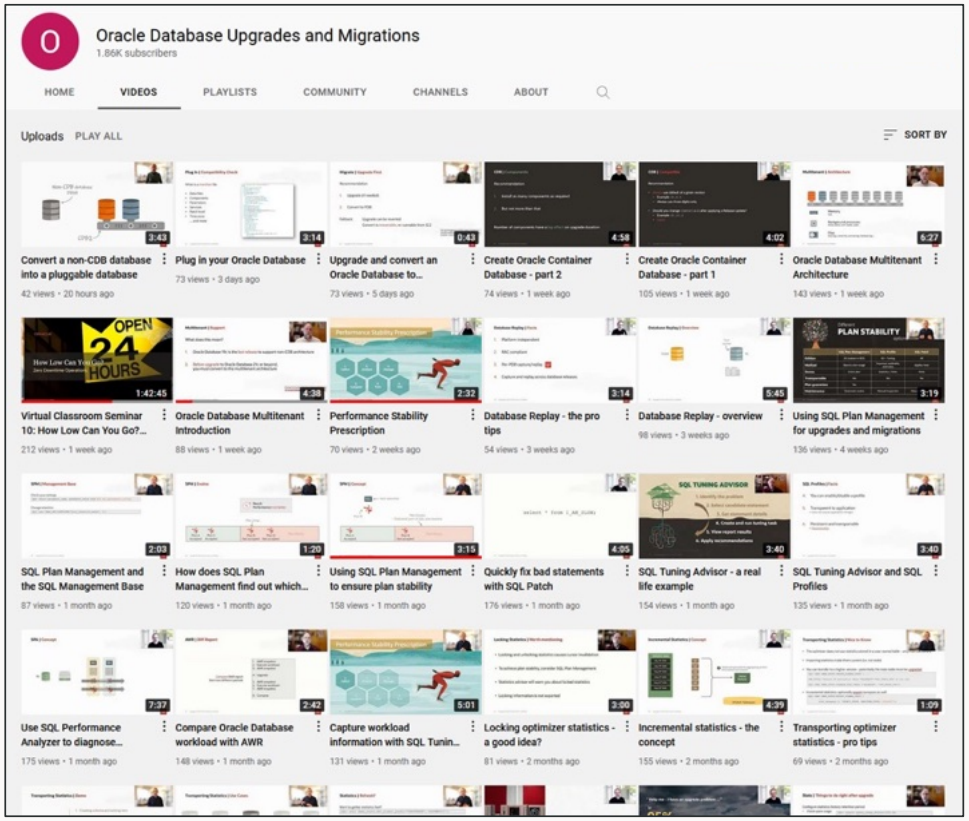
Our advice will give you a flying start

Oracle

DBAs

run the world

YouTube Channel



https://www.youtube.com/@upgradenow

- 300+ videos
- New videos every week
- No marketing
- No buzzword
- All tech

SCAN ME

Key Learnings

1. Choose the right method
2. Gather workload information
3. Use SQL Performance Analyzer

Thank You