

Oracle 19^c Performance New Features

Oracle **19^c** Performance New Features

SQL Quarantine

Automatic Indexing

Real-Time Statistics

High-Frequency Statistics Collection

Automatic SPM

High-Frequency Automatic SPM

Real-Time SQL Monitoring for Developers

19c New Features Licensing Information

19c New Features Licensing Information

19c New Features 중 Exadata 기반에서 사용가능한 기능

Feature / Option / Pack	SE2	EE	EE-ES	DBCS SE	DBCS EE	DBCS EE-HP	DBCS EE-EP	ExaCS	Notes
Automatic Indexing	N	N	Y	N	N	N	N	Y	EE-ES: Available on Exadata. Not available on Oracle Database Appliance.
SQL Quarantine	N	N	Y	N	N	N	N	Y	EE-ES: Available on Exadata. Not available on Oracle Database Appliance.
Real-Time Statistics	N	N	Y	N	N	N	N	Y	EE-ES: Available on Exadata. Not available on Oracle Database Appliance.
High-Frequency Automatic Optimizer Statistics Collection	N	N	Y	N	N	N	N	Y	EE-ES: Available on Exadata. Not available on Oracle Database Appliance.
High-Frequency Automatic SPM Evolve Advisor Task	N	N	Y	N	N	N	N	Y	You can enable this feature by setting the <code>AUTO_SPM_EVOLVE_TASK</code> parameter of the <code>DBMS_SPM.CONFIGURE</code> procedure. EE-ES: Available on Exadata. Not available on Oracle Database Appliance.
Automatic SQL Plan Management	N	N	Y	N	N	N	N	Y	EE-ES: Available on Exadata. Not available on Oracle Database Appliance.

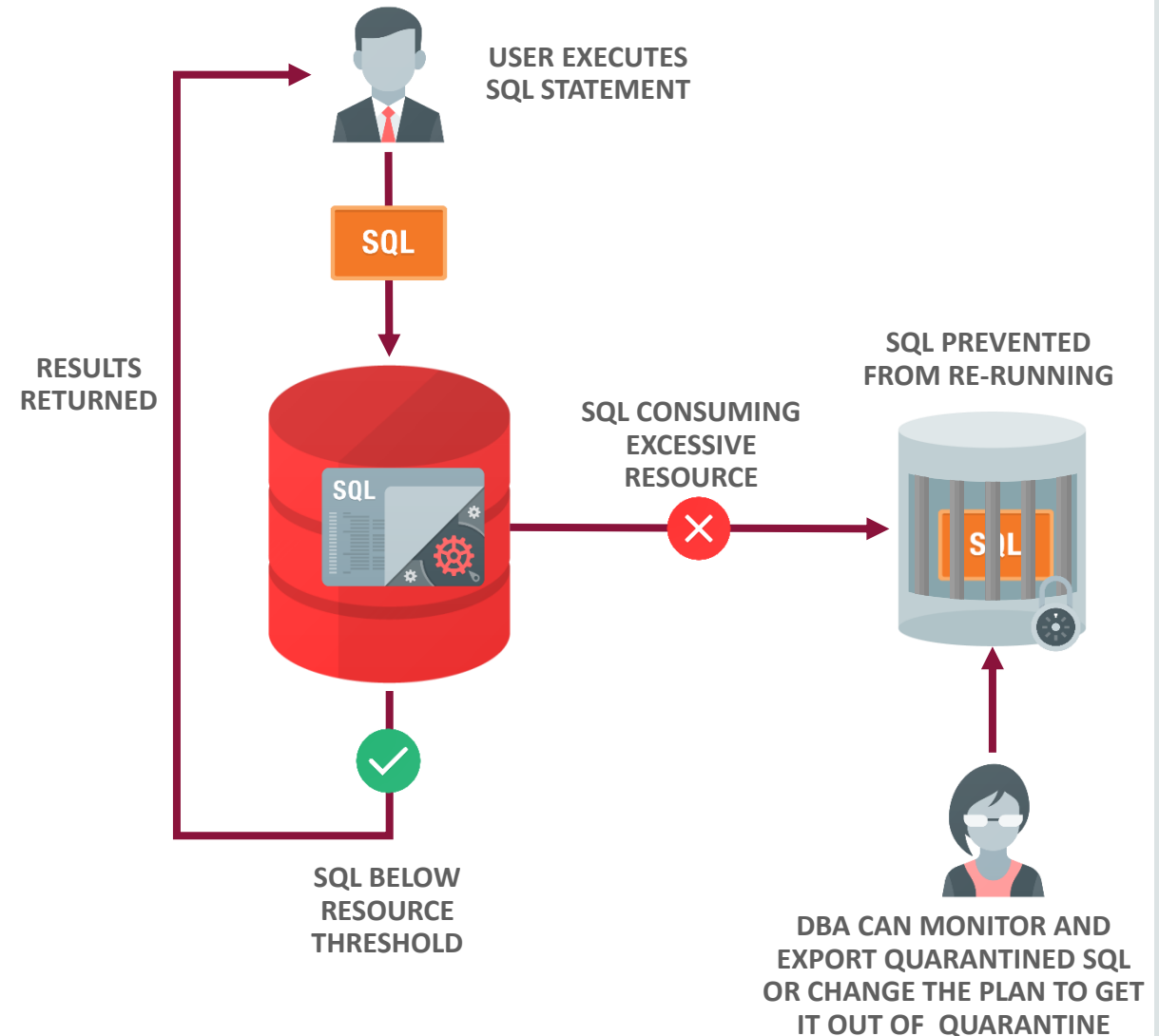
- Automatic Indexing
- SQL Quarantine
- Real-Time Statistics
- High-Frequency Automatic Optimizer Statistics
- Automatic SPM
- High-Frequency Automatic SPM
- 위의 기능들은 Exadata 기반에서만 사용할 수 있음 (19c 엔터프라이즈 에디션이나 DBCS에서 사용 불가능)

악성 SQL의 재실행을 방지하라

SQL Quarantine

SQL Quarantine (SQL 격리)

- CPU 및 I/O 리소스를 과도하게 사용한 SQL은 리소스 매니저에 의해 실행이 중지됨
- 과도한 리소스를 소비하는 SQL 실행 계획이 격리됨
- 일단 격리된 SQL은 다시 실행할 수 없음
- 반복 실행되는 악성 SQL 문을 중지하여 시스템의 전반적인 응답성능을 향상시킴
- DBA는 V\$SQL에서 격리된 구문을 모니터링 할 수 있음
- DBMS_SQLQ 혹은 DBMS_RESOURCE_MANAGER 패키지를 사용하여 Control 함

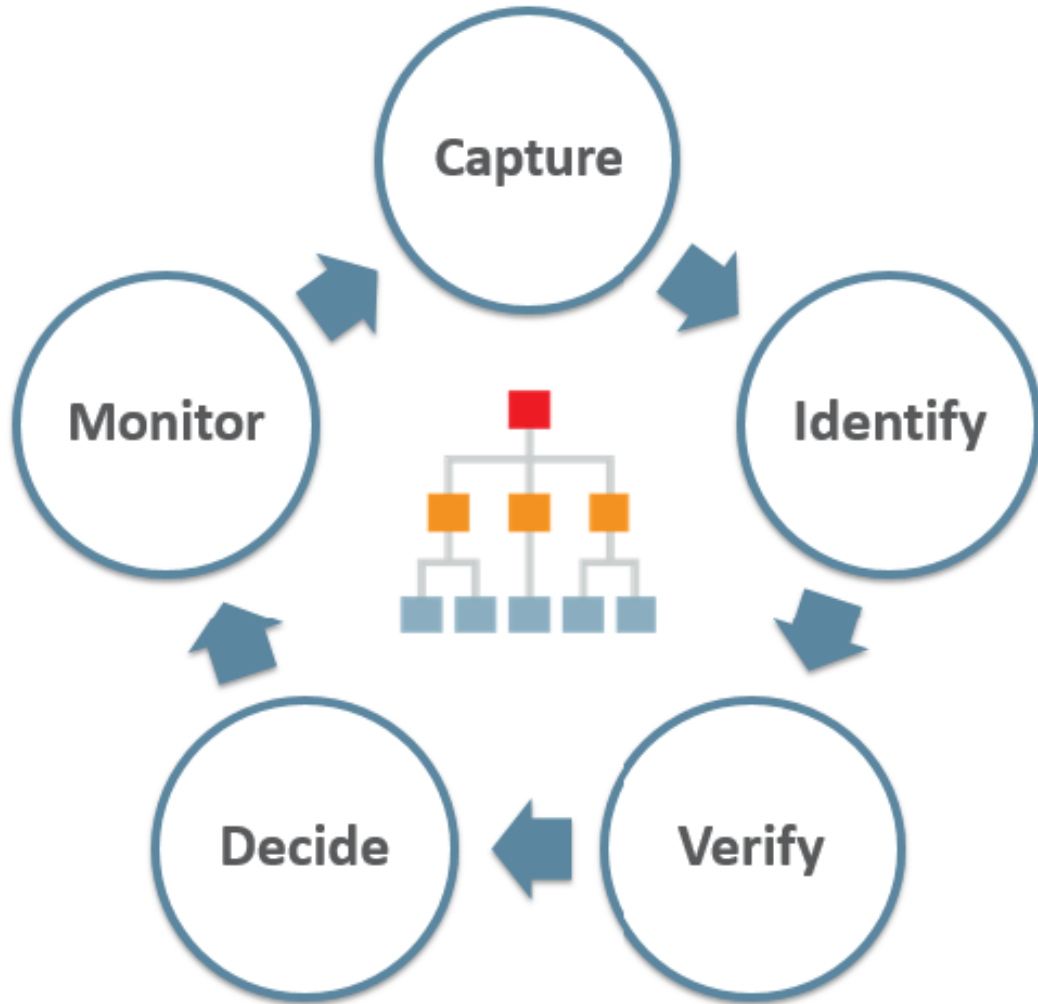


자동으로 적절한 인덱스를 생성하라

Automatic Indexing

Automatic Indexing

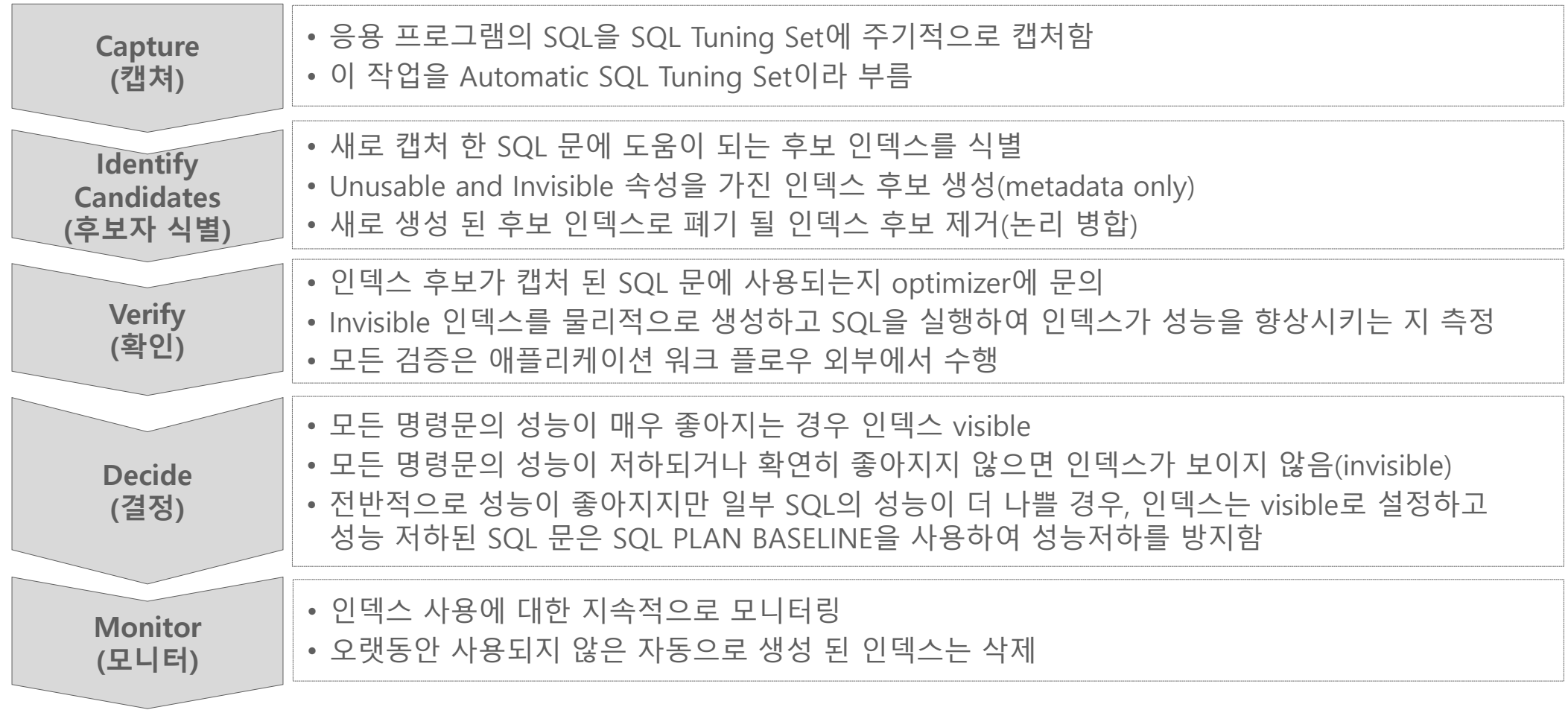
NEW IN
19^c



- Automatic Indexing 방법론은 수동 SQL 튜닝에 대한 일반적인 접근법을 기반으로 함
- 후보 인덱스를 식별하고 **적용하기 전에 유효성을 검사** 함
- 전체 프로세스는 완전히 automatic
- 정교한 자동화와 마찬가지로 투명성도 중요함
 - 모든 튜닝 활동은 reporting를 통해 감사가 가능함

Automatic Indexing - 방법론

NEW IN
19^c



Automatic Indexing- DBMS_AUTO_INDEX 패키지(1/3)



스키마 및 테이블스페이스 설정

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_SCHEMA', NULL, TRUE);
```

전체 스키마에 Automatic Index 적용

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_SCHEMA', 'SH', FALSE);  
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_SCHEMA', 'HR', FALSE);
```

특정 스키마에 Automatic Index 미적용

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_DEFAULT_TABLESPACE', 'TBS_AUTO');
```

Automatic Index 를 위한
Default Tablespace

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_SPACE_BUDGET', '5');
```

Default Tablespace에 5%까지 할당가능

Automatic Indexing- DBMS_AUTO_INDEX 패키지(2/3)



모드 설정 (적용/미적용/보고서)

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','IMPLEMENT');
```

Automatic Index 적용

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','OFF');
```

Automatic Index 미적용

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','REPORT ONLY');
```

Automatic Index 보고서(Invisible Index)

Automatic Indexing- DBMS_AUTO_INDEX 패키지(3/3)

NEW IN
19c

Retention 및 Index Drop 설정

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_REPORT_RETENTION', '60');
```

60일 후에 Report를 위한 로그삭제
(Default는 31일)

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_RETENTION_FOR_AUTO', '90');
```

90일 후에 미사용 인덱스 제거(Default는 373일)
AUTO_INDEX_RETENTION_FOR_MANUAL 을
사용하면 Manual로 생성한 미사용 인덱스 제거

```
begin
    dbms_auto_index.drop_secondary_indexes;
end;
```

전체 스키마의 index 삭제
단 constraint와 연관된 인덱스는 유지

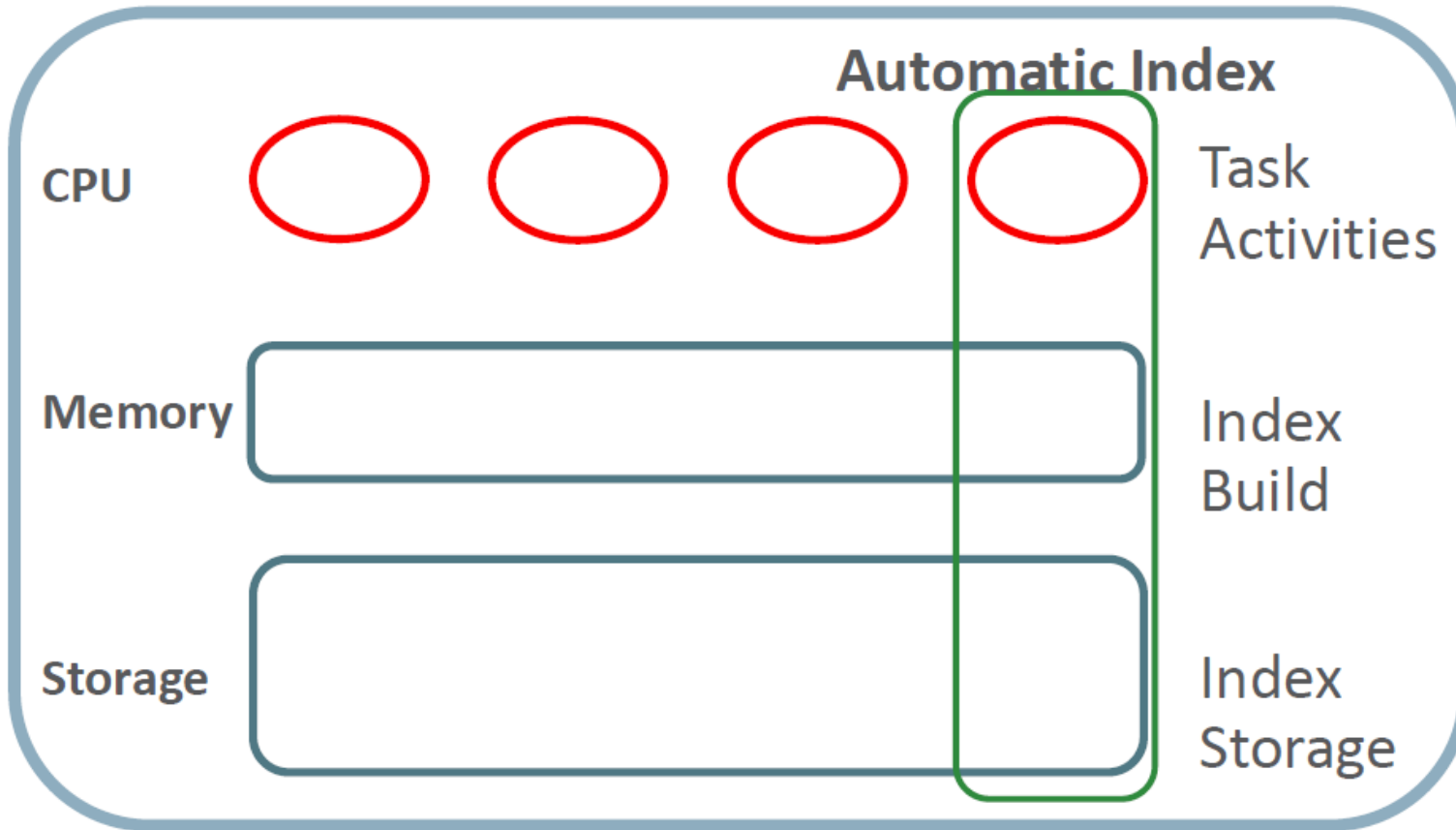
```
begin
    dbms_auto_index.drop_secondary_indexes('HR', 'EMP');
end;
```

HR 스키마의 EMP 테이블 index 삭제
단 constraint와 연관된 인덱스는 유지

Deployment – Inline Automatic Indexing

NEW IN
19^c

Production Database



- Automatic Indexing 작업은 CPU 메모리 및 Storage를 사용함
- Resource manager plan이 작업을 1 CPU 로 제한함
- 인덱스를 만들 때 사용하는 Temporary Tablespace를 지정 가능함
- Default Tablespace를 지정하고 사용량에 제한을 둘 수 있음

Using Automatic Indexing Hints

```
SELECT /*+ USE_AUTO_INDEXES */ emp_id, emp_name, dept_id  
FROM employees  
WHERE dept_id > 50;
```

```
SELECT /*+ NO_USE_AUTO_INDEXES */ emp_id, emp_name, dept_id  
FROM employees  
WHERE dept_id > 50;
```

Auditing Automatic Indexing

NEW IN
19^c

삭제되기전 log는 아래의 view에서 볼 수 있음

- `DBA_AUTO_INDEX_EXECUTIONS` – Shows history automatic indexing tasks executions
- `DBA_AUTO_INDEX_STATISTICS` – Shows statistics related to automatic indexes
- `DBA_AUTO_INDEX_IND_ACTIONS` – Shows actions performed on automatic indexes
- `DBA_AUTO_INDEX_SQL_ACTIONS` – Shows actions performed on SQL to verify automatic indexes
- `DBA_AUTO_INDEX_CONFIG` – Shows history of configuration settings related to automatic indexes

Automatic Indexing 보고서(1/4)

NEW IN
19^c

```
declare
  report clob := null;
begin
  report := DBMS_AUTO_INDEX.REPORT_ACTIVITY(
    activity_start => TO_TIMESTAMP('2018-11-01', 'YYYY-MM-DD'),
    activity_end   => TO_TIMESTAMP('2018-12-01', 'YYYY-MM-DD'),
    type           => 'HTML',
    section        => 'SUMMARY',
    level          => 'BASIC');
end;
```

특정 일자와 type을 지정하는 경우

```
declare
  report clob := null;
begin
  report := DBMS_AUTO_INDEX.REPORT_ACTIVITY();
end;
```

특정 일자와 type을 지정하지 않는 경우
(지난 24시간과 text type이 자동으로 설정됨)

Automatic Indexing 보고서(2/4)



GENERAL INFORMATION

Activity start	: 29-AUG-2018 12.20.40
Activity end	: 30-AUG-2018 12.20.40
Executions completed	: 13
Executions interrupted	: 3
Executions with fatal error	: 1

SUMMARY (AUTO INDEXES)

Index candidates	: 53
Indexes created (visible / invisible)	: 12 (12 / 0)
Space used (visible / invisible)	: 3.48 MB (3.48 MB / 0 B)
Indexes dropped	: 0
SQL statements verified	: 16
SQL statements improved (improvement factor)	: 16 (3x)
SQL statements disallowed from auto indexes	: 0
Overall improvement factor	: 3x

SUMMARY (MANUAL INDEXES)

Unused indexes (visible / invisible)	: 10 (8 / 2)
Space used (visible / invisible)	: 100 MB (76 MB / 24 MB)
Unusable indexes	: 0

Automatic Indexing 보고서(3/4)



INDEX DETAILS

1. The following indexes were created*: invisible

Owner	Table	Index	Key	Type	Properties
OPT	T_10K_CP1	SYS_AI_3cpm0ahgt469g	ROWID_UNIQUE	B-TREE	NONE
OPT	T_10K_CP1	SYS_AI_3rk4h2m9d49b5	CHAR_UNIQUE	B-TREE	NONE
OPT	T_10K_CP1	SYS_AI_5cq2h6jhmznc9	DATE_UNIQUE	B-TREE	NONE
OPT	T_10K_CP1	SYS_AI_6vg5wr5nwcqxs	THOUSAND	B-TREE	NONE
OPT	T_10K_CP1	SYS_AI_agnvzczmz4z0a	TEN, UNIQUE1, UNIQUE2	B-TREE	NONE
OPT	T_10K_CP1	SYS_AI_bcms9qy98nq1c	VCHAR_UNIQUE	B-TREE	NONE
OPT	T_5K_CP	SYS_AI_0urcv8chmxu20	VCHAR_UNIQUE	B-TREE	NONE
OPT	T_5K_CP	SYS_AI_2pvk34mqdh7pa	TEN, UNIQUE1, UNIQUE2	B-TREE	NONE
OPT	T_5K_CP	SYS_AI_428hqd6qu531y	THOUSAND	B-TREE	NONE
OPT	T_5K_CP	SYS_AI_5d2cukrm2gju2	DATE_UNIQUE	B-TREE	NONE
OPT	T_5K_CP	SYS_AI_97zrtmcmn5tz6	CHAR_UNIQUE	B-TREE	NONE
OPT	T_5K_CP	SYS_AI_cn9fsv12paxcb	ROWID_UNIQUE	B-TREE	NONE

Automatic Indexing 보고서(4/4)

NEW IN
19^c

VERIFICATION DETAILS

1. The performance of the following statements improved:-----

Schema Name : OPT
SQL ID : 2vy3tr5kyg88z
SQL Text : select count(*) from t_5k_cp where vchar_unique ='MAN'
Improvement Factor : 2x

PLANS SECTION

Original

Plan Hash Value : 3944640934

Id	Operation	Name	Rows	Bytes	Cost	Time
0	SELECT STATEMENT					
1	SORT AGGREGATE					
2	TABLE ACCESS FULL	T_5K_CP				

With Auto Indexes

Plan Hash Value : 2541075899

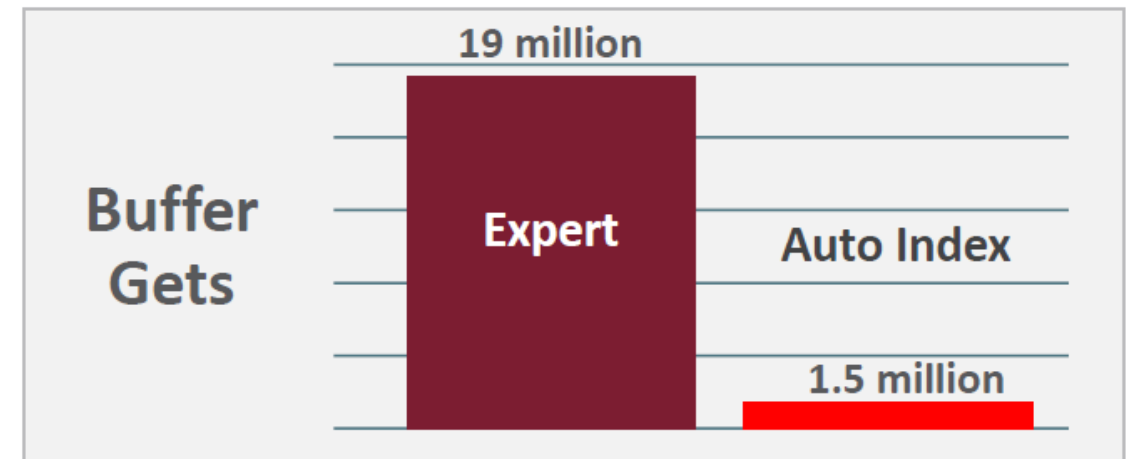
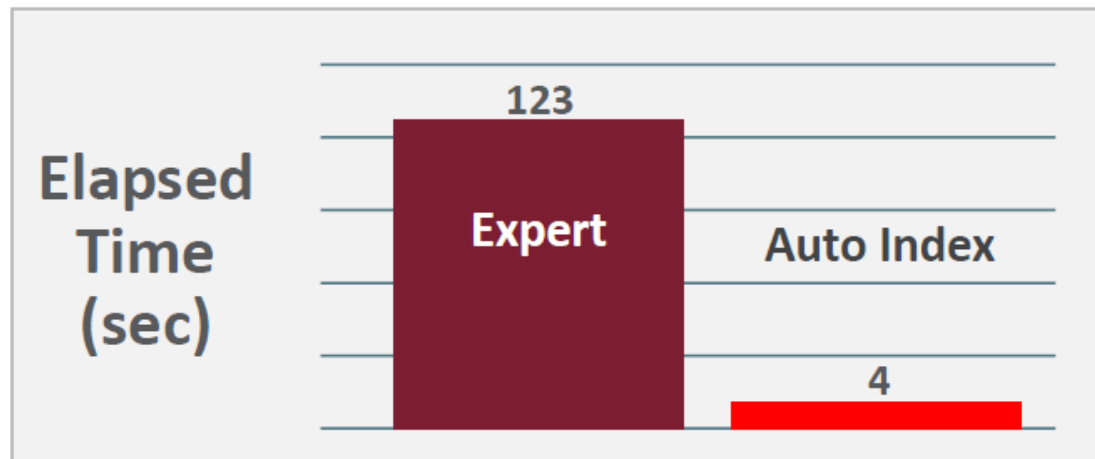
Id	Operation	Name	Rows	Bytes	Cost	Time
0	SELECT STATEMENT					
1	SORT AGGREGATE					
* 2	INDEX RANGE SCAN	SYS_AI_0urcv8chmxu20				

Predicate Information (identified by operation id):

* 2 - access ("VCHAR_UNIQUE"='MAN')

Validation – **Accounts Receivable**

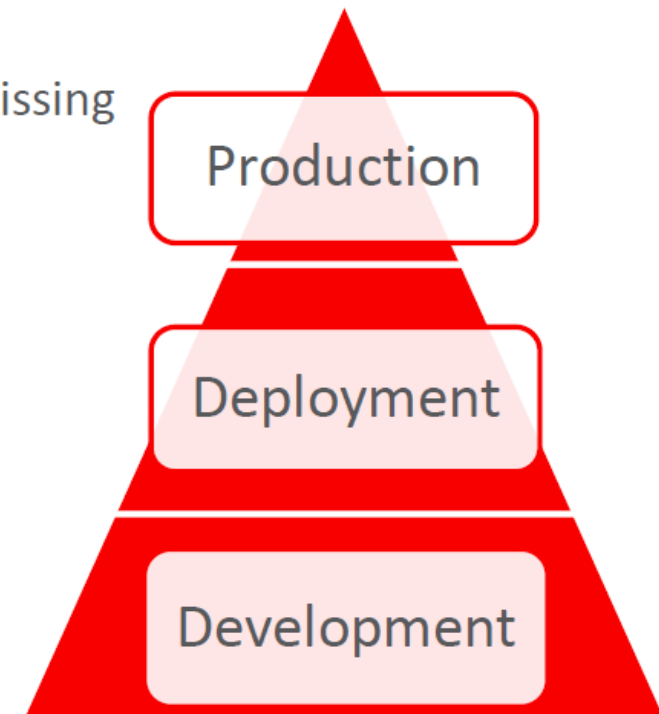
- Workload: 4,889 SQL statements
- Indexes:
 - Experts created 49 indexes of which 17 were used
 - Automatic indexing created 5 indexes, *all* of which were used



Automatic Indexing – Scope

NEW IN
19^c

- Useful for OLTP, DW, Mixed workloads but very **critical** for OLTP
- Applies to tuned and un-tuned applications
 - Tuned
 - Existing secondary indexes may be outdated or important ones can be missing
 - Some secondary indexes can be dropped and auto indexes can be added
 - Un-tuned
 - Existing indexes support primary or unique key constraints
- Applicable to all stages of an application lifecycle
- Supports
 - Single and concatenated indexes
 - Function-based indexes
 - Compression Advanced Low



OLTP성 DML도 통계정보를 실시간으로 관리하라

Real-Time Statistics

Online Statistics Gathering for Direct Path Load

- Oracle 12.1 부터 Append Insert와 CTAS등 Direct Path Load에 대해 자동으로 통계를 수집함.
- 파라미터 `_optimizer_gather_stats_on_load` 값이 Default로 True임

```
DROP TABLE tab1 PURGE;

CREATE TABLE tab1 AS
SELECT level AS id,
       'Description of ' || level AS description
FROM   dual
CONNECT BY level <= 1000;
```

```
COLUMN table_name FORMAT A20
```

```
SELECT table_name,
       num_rows
FROM   user_tables
WHERE  table_name = 'TAB1';
```

TABLE_NAME	NUM_ROWS
TAB1	1000

CTAS 결과 1000건의
통계정보가 자동으로 생성됨

```
TRUNCATE TABLE tab1;

INSERT /*+ APPEND */ INTO tab1
SELECT level AS id,
       'Description of ' || level AS description
FROM   dual
CONNECT BY level <= 500;
COMMIT;
```

```
COLUMN table_name FORMAT A20
```

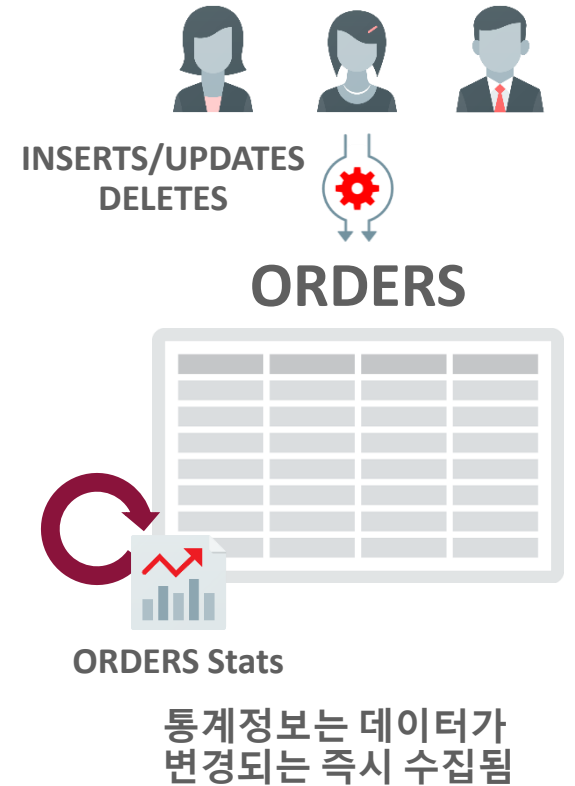
```
SELECT table_name,
       num_rows
FROM   user_tables
WHERE  table_name = 'TAB1';
```

TABLE_NAME	NUM_ROWS
TAB1	500

APPEND INSERT 결과 500건의
통계정보가 자동으로 생성됨

Real-Time Statistics

- Conventional DML(Insert/update/merge)에 대해 실시간 통계 수집
- DML 중에 수집된 통계는 오버헤드가 무시될 정도로 빠를 필요가 있음
- 필수 통계만 수집하여 치명적인 SQL 실행 계획으로 인한 성능 저하를 방지함 (e.g. avoiding out-of-range conditions)
 - Min, Max, num_rows, etc.
- 나머지 상세 통계는 지연되어 수집됨
 - maintenance window에 의한 자동 통계수집 작업으로 히스토그램등의 상세 통계가 수집됨



Online Statistics Gathering for conventional DML

NEW IN
19c

- Oracle19c 부터 OLTP 성 DML에 대해 자동으로 통계를 수집함.
- 파라미터 `_optimizer gather_stats_on_conventional_dml`로 Control 함. Default로 True임
- 통계 생성시 성능 오버 헤드를 최소화하기 위해 가장 필수적인 통계만 생성됨. 나머지 상세한 통계(예:NDV)는 수동으로 `DBMS_STATS`를 실행하거나 Automatic optimizer statistics collection(11g 매일 수집) 혹은 High-Frequency Statistics Collection(19c)에서 수집됨

```
SQL> set serveroutput on
```

```
declare
```

```
  v_inicio number;
```

```
  v_fin number;
```

```
begin
```

```
  v_inicio:=DBMS_UTILITY.get_time;
```

```
  for i in 2000001..2500000 loop
```

```
    insert into TEST values (i,'T' || to_char(i));
```

```
  end loop;
```

```
  commit;
```

```
  v_fin:=DBMS_UTILITY.get_time;
```

```
  DBMS_OUTPUT.put_line('Duracion: ' || to_char((v_fin-v_inicio)/100) || ' seg');
```

```
end;
```

```
/
```

```
Duracion: 23.66 seg
```

```
PL/SQL procedure successfully completed.
```

200만건의 test 테이블에
50만건의 insert를 추가로 발생시킴

```
SQL> select NOTES,NUM_ROWS,STALE_STATS
       from DBA_TAB_STATISTICS
       where owner='FRICCIO' and table_name='TEST';
```

NOTES	NUM_ROWS	STALE_STATS
-------	----------	-------------

-----	-----	-----
-------	-------	-------

	2000000	YES
--	---------	-----

STATS_ON_CONVENTIONAL_DML	2500000	
---------------------------	---------	--

50만건의 통계가 추가됨

Online Statistics Gathering for conventional DML



```
SQL> select CAMPO2 from TEST where campol=217002;
```

CAMPO2

T217002

```
SQL> select * from table(dbms_xplan.display_cursor(format=>'TYPICAL'));
```

PLAN_TABLE_OUTPUT

SQL_ID 35q795p6a3dhm, child number 0

select CAMPO2 from TEST where campol=217002

Plan hash value: 480040

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT				4 (100)	
1	TABLE ACCESS BY INDEX ROWID BATCHED	TEST	1	13	4 (0)	00:00:01
* 2	INDEX RANGE SCAN	SYS_AI_ayv3vppgrvmw8	1		3 (0)	00:00:01

Predicate Information (identified by operation id):

2 - access("CAMPOL"=217002)

Note

- dynamic statistics used: statistics for conventional DML

옵티마이저가 Conventional DML에 의한 통계정보
변경사항을 인식하여 실행계획을 작성함

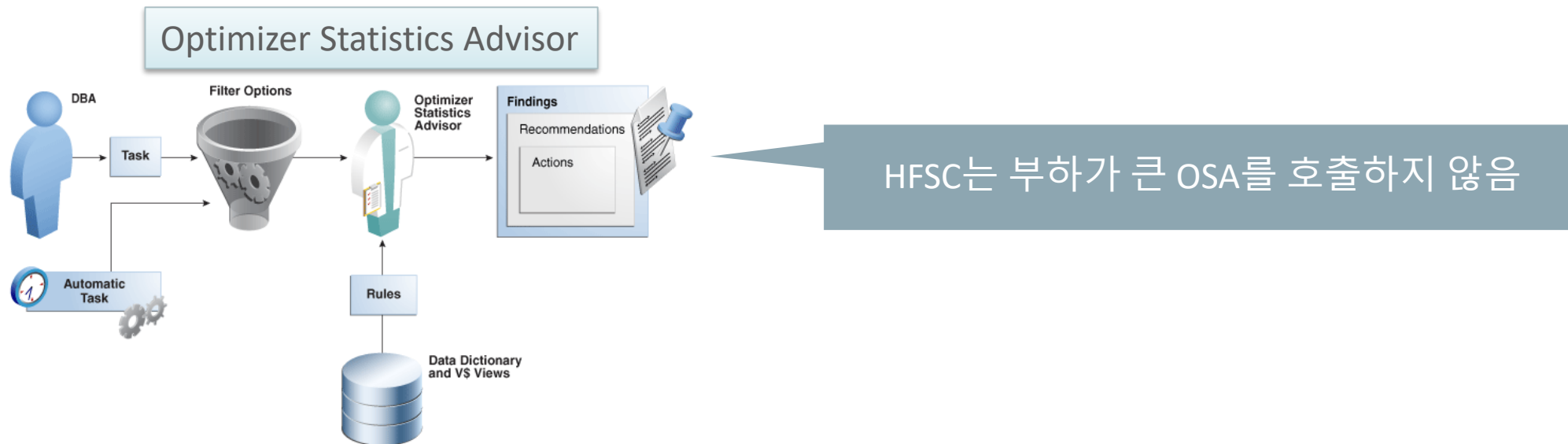
부실 통계를 빈번하게 보완하라

High-Frequency Statistics Collection

High-Frequency Statistics Collection(HFSC)

NEW IN
19^c

- 통계 수집 후 다음 번 통계를 수집할 때 까지 DML이 많이 발생하면 부실(stale) 상태가 될 수 있음
- 부실한 통계는 악성 실행계획을 만들 수 있음
- 따라서 통계 수집 후 다음 번 통계를 수집할 때 까지 데이터의 변경이 크다면 통계를 수집해야 될 필요가 있음
- 하지만 일반적인 통계수집은 부하가 크기 때문에 빈번히 실행 할 수 없음
- HFSC는 " 경량" 통계 수집 작업이며, 부실(STALE)한 경우에만 통계만 수집함
- HFSC는 존재하지 않는 오브젝트에 대한 통계를 삭제하지 않으며, Optimizer Statistics Advisor를 호출하지 않음
- 19c의 Real Time Statistics는 필수적인 통계 정보만 수집함으로 HFSC로 보완함



High-Frequency Statistics Collection

NEW IN
19c

- 아래의 예제는 SALES, CUSTOMER 테이블에 DML을 발생시켜 통계를 STALE로 만들고 HFSC를 실행시킴

```
SELECT TABLE_NAME, PARTITION_NAME, NUM_ROWS, STALE_STATS
FROM   DBA_TAB_STATISTICS
WHERE  OWNER = 'SH'
AND    TABLE_NAME IN ('CUSTOMERS', 'SALES')
ORDER BY TABLE_NAME, PARTITION_NAME;
```

TABLE_NAME	PARTITION_NAME	NUM_ROWS	STA
CUSTOMERS		55500	NO
SALES	SALES_1995	0	NO
SALES	SALES_1996	0	NO
SALES	SALES_H1_1997	0	NO
SALES	SALES_H2_1997	0	NO
SALES	SALES_Q1_1998	43687	NO
SALES	SALES_Q1_1999	64186	NO
SALES	SALES_Q1_2000	62197	NO
SALES	SALES_Q1_2001	60608	NO
SALES	SALES_Q1_2002	0	NO
SALES	SALES_Q1_2003	0	NO
SALES	SALES_Q2_1998	35758	NO
SALES	SALES_Q2_1999	54233	NO
SALES	SALES_Q2_2000	55515	NO
SALES	SALES_Q2_2001	63292	NO
SALES	SALES_Q2_2002	0	NO
SALES	SALES_Q2_2003	0	NO
SALES	SALES_Q3_1998	50515	NO
SALES	SALES_Q3_1999	67138	NO
SALES	SALES_Q3_2000	58950	NO
SALES	SALES_Q3_2001	65769	NO
SALES	SALES_Q3_2002	0	NO
SALES	SALES_Q3_2003	0	NO
SALES	SALES_Q4_1998	48874	NO
SALES	SALES_Q4_1999	62388	NO
SALES	SALES_Q4_2000	55984	NO
SALES	SALES_Q4_2001	69749	NO
SALES	SALES_Q4_2002	0	NO
SALES	SALES_Q4_2003	0	NO
SALES		918843	NO

STALE_STATS 값이 NO 임으로
부실 통계 가 없음

SALES 테이블 건수가 100% 증가
CUSTOMER 테이블은 1건 증가

```
-- insert 918K rows in sales
INSERT INTO sh.sales SELECT * FROM sh.sales;
-- update around 15% of sales rows
UPDATE sh.sales SET amount_sold = amount_sold + 1 WHERE amount_sold > 100;
-- insert 1 row into customers
INSERT INTO sh.customers(cust_id, cust_first_name, cust_last_name,
    cust_gender, cust_year_of_birth, cust_main_phone_number,
    cust_street_address, cust_postal_code, cust_city_id,
    cust_city, cust_state_province_id, cust_state_province,
    country_id, cust_total, cust_total_id)
VALUES(188710, 'Jenny', 'Smith', 'F', '1966', '555-111-2222',
    '400 oracle parkway', '94065', 51402, 'Redwood Shores',
    52564, 'CA', 52790, 'Customer total', '52772');
COMMIT;
```

통계를 DISK에 저장

```
EXEC DBMS_STATS.FLUSH_DATABASE_MONITORING_INFO;
```

High-Frequency Statistics Collection — DML에 의한 STALE 통계조회

NEW IN
19c

- DML로 인한 부실 통계 확인

```
SELECT TABLE_NAME, PARTITION_NAME, NUM_ROWS, STALE_STATS
FROM   DBA_TAB_STATISTICS
WHERE  OWNER = 'SH'
AND    TABLE_NAME IN ('CUSTOMERS', 'SALES')
ORDER BY TABLE_NAME, PARTITION_NAME;
```

TABLE_NAME	PARTITION_NAME	NUM_ROWS	STA
CUSTOMERS		55500	NO
SALES	SALES_1995	0	NO
SALES	SALES_1996	0	NO
SALES	SALES_H1_1997	0	NO
SALES	SALES_H2_1997	0	NO
SALES	SALES_Q1_1998	43687	YES
SALES	SALES_Q1_1999	64186	YES
SALES	SALES_Q1_2000	62197	YES
SALES	SALES_Q1_2001	60608	YES
SALES	SALES_Q1_2002	0	NO
SALES	SALES_Q1_2003	0	NO
SALES	SALES_Q2_1998	35758	YES
SALES	SALES_Q2_1999	54233	YES
SALES	SALES_Q2_2000	55515	YES
SALES	SALES_Q2_2001	63292	YES
SALES	SALES_Q2_2002	0	NO
SALES	SALES_Q2_2003	0	NO
SALES	SALES_Q3_1998	50515	YES
SALES	SALES_Q3_1999	67138	YES
SALES	SALES_Q3_2000	58950	YES
SALES	SALES_Q3_2001	65769	YES
SALES	SALES_Q3_2002	0	NO
SALES	SALES_Q3_2003	0	NO
SALES	SALES_Q4_1998	48874	YES
SALES	SALES_Q4_1999	62388	YES
SALES	SALES_Q4_2000	55984	YES
SALES	SALES_Q4_2001	69749	YES
SALES	SALES_Q4_2002	0	NO
SALES	SALES_Q4_2003	0	NO
SALES		1837686	
SALES		918843	YES

SALES 통계는 STALE
CUSTOMER 통계는 정상

변경된 파티션 데이터는 STALE

Insert 때문에 REAL TIME STATISTICS가 실행됨

High-Frequency Statistics Collection - 설정

NEW IN
19c

- 통계가 STALE 상태가 되었음으로 HFSC를 실행함

HFSC를 ENABLE 시킴
OFF는 DISABLE이며 Default값임

```
EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO_TASK_STATUS','ON');  
EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO TASK MAX RUN TIME','180');  
EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO TASK INTERVAL','240');
```

maximum run time을
3분으로 설정
Default는 1시간 임

재수행 시간을 4분으로 설정
Default는 15분 임

High-Frequency Statistics Collection - 모니터링

- Stale 통계가 존재함으로 HFSC가 INTERVAL에 의해 반복 실행됨
- HSFC는 maintenance window에 의한 자동 통계수집 작업(하루에 한번)중에 실행될 수 없음(제약사항)

```
SELECT OPID, ORIGIN, STATUS, TO_CHAR(START_TIME, 'DD/MM HH24:MI:SS' ) AS BEGIN_TIME,  
       TO_CHAR(END_TIME, 'DD/MM HH24:MI:SS') AS END_TIME, COMPLETED, FAILED,  
       TIMED_OUT AS TIMEOUT, IN_PROGRESS AS INPROG  
FROM DBA_AUTO_STAT_EXECUTIONS  
ORDER BY OPID;
```

HFSC와 SASC 를 모니터링 함

HFSC가 4분(INTERVAL)단위로
2번 실행됨

OPID	ORIGIN	STATUS	BEGIN_TIME	END_TIME	COMPLETED
790	HIGH_FREQ_AUTO_TASK	COMPLETED	03/10 14:54:02	03/10 14:54:35	338
793	HIGH FREQ AUTO TASK	COMPLETED	03/10 14:58:11	03/10 14:58:45	193
794	AUTO TASK	COMPLETED	03/10 15:00:02	03/10 15:00:20	52

standard automatic statistics
collection(SASC)이 한번 실행됨

실행계획관리를 자동화 하라

Automatic SPM

SQL Plan Management (SPM)


11g

- SPM을 사용하면 기본적으로 승인된 PLAN으로만 실행된다.
- 새로운 PLAN이 생성되면 바로 적용하지 않고, PLAN과 수행시간 및 I/O량을 저장한다.
- SPM은 OLD Plan과 New Plan의 수행시간 및 I/O량을 비교하여 월등히 나은 경우만 New Plan으로 실행한다.
- 더 좋은 실행계획을 적용(evolve) 하는 것은 Auto로 할수도 있고 Manual로 할수도 있다.


Accepted
“SQL Plan Baseline”

A SQL plan baseline is an accepted plan.

```
SELECT a.data,  
       b.data  
FROM   tab1 a  
JOIN   tab2 b ON  
       b.tab1_id = a.id  
WHERE  a.code = :1
```



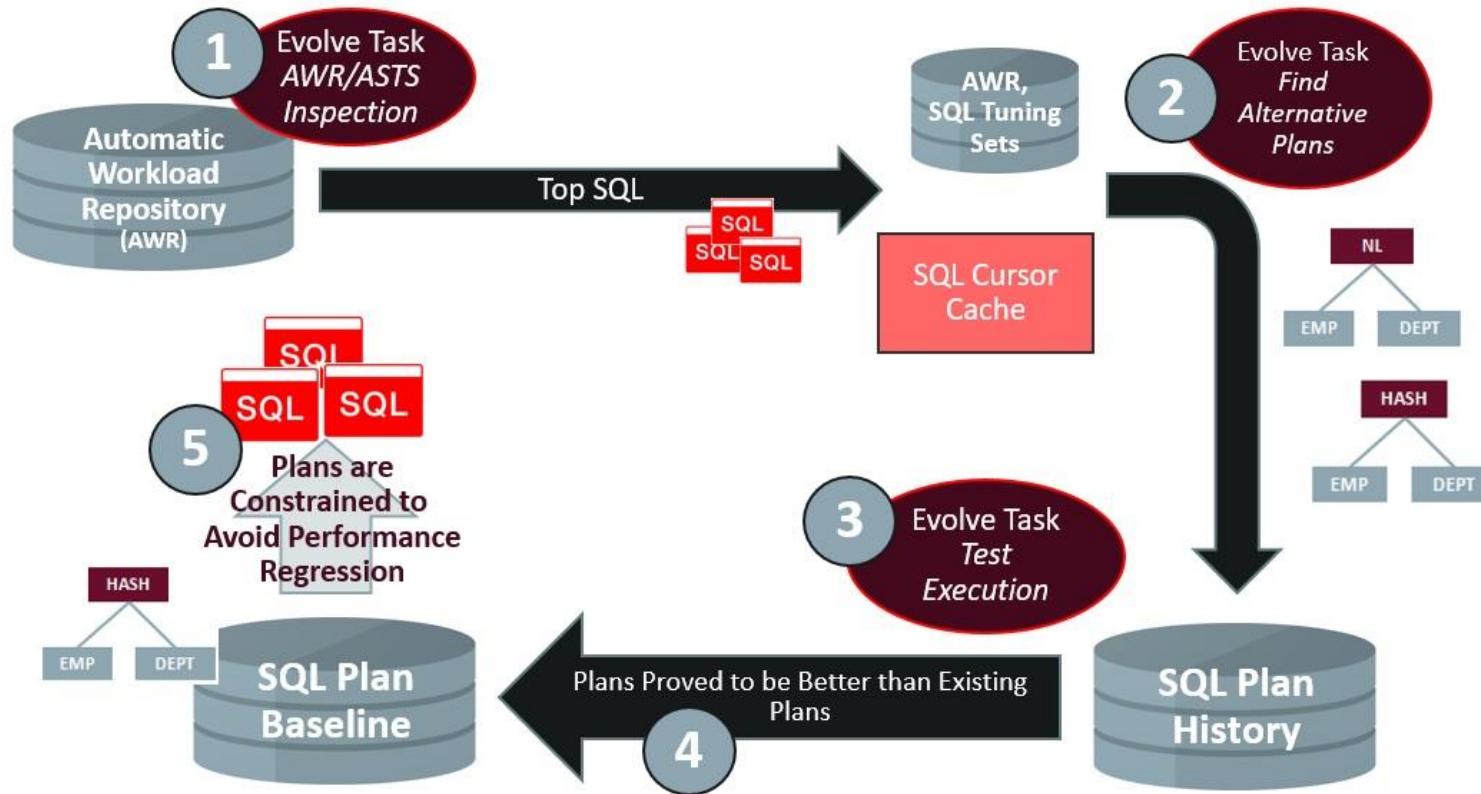
Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT			
1	NESTED LOOPS		25	425
2	NESTED LOOPS		25	425
* 3	TABLE ACCESS FULL	TAB1	1	11
* 4	INDEX RANGE SCAN	TAB2_TAB1_FKI	25	
5	TABLE ACCESS BY INDEX ROWID	TAB2	25	150



Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT			
* 1	HASH JOIN		25	425
* 5	TABLE ACCESS FULL	TAB1	1	11
8	TABLE ACCESS FULL	TAB2	25	150

Automatic SPM

NEW IN
19^c



1. AWR은 시스템 자원을 많이 소비하는 SQL 실행 계획을 검사함. 또한 사용 가능한 경우 Automatic SQL tuning set (ASTS)를 검사함. (ASTS는 주로 automatic indexing을 위해 관리되는 하스 튜닝 세트임)
2. 데이터베이스는 AWR, SQL 튜닝 세트 및 커서 캐시와 같은 다양한 소스에서 대체 SQL 실행 계획을 찾음. 식별된 계획이 SQL Plan 히스토리에 추가됨.
3. SPM evolution advisor는 테스트(대체 계획을 실행하고 성능을 비교)를 실행함
4. Evolve advisor는 어떤 계획이 가장 잘 수행되는지 결정하고 이를 SQL plan baseline에 추가함
5. SQL plan baseline은 'regressed' 실행 계획이 사용되는 것을 방지함

Enable/Disable Automatic SPM

Enable

```
BEGIN
  DBMS_SPM.SET_EVOLVE_TASK_PARAMETER(
    task_name => 'SYS_AUTO_SPM_EVOLVE_TASK' ,
    parameter => 'ALTERNATE_PLAN_BASELINE',
    value      => 'AUTO');
END;
/

BEGIN
  DBMS_SPM.SET_EVOLVE_TASK_PARAMETER(
    task_name => 'SYS_AUTO_SPM_EVOLVE_TASK',
    parameter => 'ALTERNATE_PLAN_SOURCE',
    value      => 'AUTO');
END;
/
```

Disable

```
BEGIN
  DBMS_SPM.SET_EVOLVE_TASK_PARAMETER(
    task_name => 'SYS_AUTO_SPM_EVOLVE_TASK' ,
    parameter => 'ALTERNATE_PLAN_BASELINE',
    value      => '');
END;
/

BEGIN
  DBMS_SPM.SET_EVOLVE_TASK_PARAMETER(
    task_name => 'SYS_AUTO_SPM_EVOLVE_TASK',
    parameter => 'ALTERNATE_PLAN_SOURCE',
    value      => '');
END;
/
```

확인

```
SELECT PARAMETER_NAME, PARAMETER_VALUE
FROM   DBA_ADVISOR_PARAMETERS
WHERE  TASK_NAME = 'SYS_AUTO_SPM_EVOLVE_TASK';
```

개선된 실행 계획을 수동으로 승인하려는 경우

```
BEGIN
  DBMS_SPM.SET_EVOLVE_TASK_PARAMETER(
    task_name => 'SYS_AUTO_SPM_EVOLVE_TASK' ,
    parameter => 'ACCEPT_PLANS',
    value      => FALSE);
END;
/
```

SPM 보고서 및 test 결과를 볼 수 있도록 automatic SPM evolve advisor task를 허용한 상태임

최적의 실행계획을 위해 Automatic SPM을 자주 실행하라

High-Frequency Automatic SPM

High-Frequency Automatic SPM Evolve Advisor



- 기본적으로 Automatic SPM Evolve Advisor는 maintenance window에서 하루에 한번 실행됨
- 데이터가 자주 변경된다면 하루 사이에도 비효율적인 실행계획이 만들어 질 수 있음
- 19c에서는 정규 maintenance window 시간 이외에 더 자주 실행되도록 구성 할 수 있음
- 이를 제어하기 위해 DBMS_SPM.CONFIGURE 프로시저는 새로운 매개 변수를 지원함

```
EXEC DBMS_SPM.CONFIGURE('AUTO_SPM_EVOLVE_TASK', 'ON');
```

- Default는 OFF이며 ON과 AUTO의 의미는 동일함
- ON 혹은 AUTO로 지정되면 SPM Evolve Advisor가 한시간에 한번씩 실행되며, 최대 수행시간은 30분임

Enable High-Frequency Automatic SPM Evolve Advisor

NEW IN
19c

sys로 로그인하여 DBMS_SPM.CONFIGURE 프로시저를 실행해야함

(1)

```
SELECT PARAMETER_NAME, PARAMETER_VALUE
FROM   DBA_SQL_MANAGEMENT_CONFIG
WHERE  PARAMETER_NAME LIKE '%SPM%';
```

PARAMETER_NAME	PARAMETER_VALUE
AUTO_SPM_EVOLVE_TASK	OFF
AUTO_SPM_EVOLVE_TASK_INTERVAL	3600
AUTO_SPM_EVOLVE_TASK_MAX_RUNTIME	1800

Default로 off이며
주기는 한시간
최대 수행시간은 30분

(2)

```
EXEC DBMS_SPM.CONFIGURE('AUTO_SPM_EVOLVE_TASK', 'ON');
```

(3)

```
SELECT PARAMETER_NAME, PARAMETER_VALUE
FROM   DBA_SQL_MANAGEMENT_CONFIG
WHERE  PARAMETER_NAME = 'AUTO_SPM_EVOLVE_TASK';
```

PARAMETER_NAME	PARAMETER_VALUE
AUTO_SPM_EVOLVE_TASK	ON

High-Frequency
Automatic SPM Evolve
Advisor 활성화

Monitoring High-Frequency Automatic SPM Evolve Advisor

NEW IN
19c

활성화 후 몇시간이 흐른뒤 advisor의 활동 내역을 조회함

```
SELECT TASK_NAME, EXECUTION_NAME, STATUS
FROM   DBA_ADVISOR_EXECUTIONS
WHERE  TASK_NAME LIKE '%SPM%'
AND    (EXECUTION_NAME LIKE 'SYS_SPM%' OR EXECUTION_NAME LIKE 'EXEC_%')
ORDER BY EXECUTION_END;
```

TASK_NAME	EXECUTION_NAME	STATUS
SYS_AUTO_SPM_EVOLVE_TASK	SYS_SPM_2019-06-03/13:15:26	COMPLETED
SYS_AUTO_SPM_EVOLVE_TASK	SYS_SPM_2019-06-03/14:16:04	COMPLETED
SYS_AUTO_SPM_EVOLVE_TASK	EXEC_6	COMPLETED
SYS_AUTO_SPM_EVOLVE_TASK	SYS_SPM_2019-06-03/15:16:32	COMPLETED
SYS_AUTO_SPM_EVOLVE_TASK	SYS_SPM_2019-06-03/16:17:00	COMPLETED
...		

하루에 한번씩
실행되는 정규 작업임

한시간에 한번씩
Automatic SPM Evolve
Advisor가 작동함

개발자에게 성능 모니터링 권한을 부여하다

Real-Time SQL Monitoring for Developers

Real-Time SQL Monitoring for Developers



기능의 등장 배경

- 18c까지는 SELECT_CATALOG_ROLE 권한이 있는(주로 SYS 및 SYSTEM) 유저만 Real-Time SQL Monitoring이 가능 했음
- Database 개발자의 주요 책임은 SQL 문을 작성하고 튜닝 하는 것임. SQL Monitor reports를 볼 수 있으면 데이터베이스 관리자 권한 없이도 이러한 작업을 수행 할 수 있음.



- 19c 부터 특정 권한이없는 모든 데이터베이스 사용자는 자신이 실행한 SQL 문에 대한 실시간 모니터링 보고서를 생성 할 수 있음
- 모니터링 된 명령문과 실행 방법을 확인하기 위한 새로운 뷰가 제공됨
 - V\$ALL_SQL_MONITOR
 - V\$ALL_SQL_PLAN_MONITOR
 - V\$ALL_SQL_PLAN
 - V\$ALL_ACTIVE_SESSION_HISTORY

Real-Time SQL Monitoring for Developers(1/2)

NEW IN
19c

SELECT_CATALOG_ROLE 권한 없이 유저 생성

```
SQL> create user dev1 identified by dep1;  
User created.
```

```
SQL> grant connect, resource to dev1;  
Grant succeeded.
```

```
SQL> grant select on dev.tab1 to dev1;  
Grant succeeded.
```

```
SQL> grant select on dev.tab2 to dev1;
```

SQL 실행

```
SQL> conn dev1/dep1
```

```
select /*+ monitoring */ count(*)  
  from dev.tab1 a, dev.tab2 b  
 where a.c1=b.c1;
```

```
SQL> SELECT DBMS_SQLTUNE.report_sql_monitor_list(type => 'html', report_level => 'ALL') AS report FROM dual;
```

SQL List Monitoring 실행

SQL Monitoring List

Status	Duration	SQL Id or DBOP Name	User	Dop	DB Time	IOs	Start	End	SQL Text
EXECUTING	71s	6mjrxjg7a8guk	DEV1		70s		06/11/2019 16:57:48		select /*+ monitoring */ count(*) from dev.tab1 a, dev.tab2 b where a....
EXECUTING	2843s	4f48nwrxfd0m4	DEV1		2842s	905	06/11/2019 16:11:36		select /*+ monitoring */ count(*) from dev.tab1 a, dev.tab2 b where a....

```
SQL> spool off
```

Real-Time SQL Monitoring for Developers(2/2)

NEW IN
19^c

```
SQL> select dbms_sqltune.report_sql_monitor(type=>'html') from dual;
```

SQL Monitoring Report

SQL Text

SQL Monitoring 실행

```
select /*+ monitoring */ count(*) from dev.tab1 a, dev.tab2 b where a.c1=b.c1
```

Global Information: EXECUTING

Instance ID : 1
Session : DEV1 (18040:17462)
SQL ID : 6mjnxjg7a8guk
SQL Execution ID : 16777216
Execution Started : 06/11/2019 16:57:48
First Refresh Time : 06/11/2019 16:57:54
Last Refresh Time : 06/11/2019 17:01:26
Duration : 218s
Module/Action : SQL*Plus/-
Service : testus
Program : sqlplus@hpcexa01.de.oracle.com
(TNS V1-V3)

Buffer Gets	Database Time	Wait Activity
9141	220s	

SQL Plan Monitoring Details (Plan Hash Value=2530687813)

Id	Operation	Name	Estimated Rows	Cost	Active Period (218s)	Execs	Rows	Memory	Temp	IO Requests	CPU Activity	Wait Activity
0	SELECT STATEMENT					1	0					
1	TEMP TABLE TRANSFORMATION					1						
2	LOAD AS SELECT (CURSOR DURATION MEMORY)	SYS_TEMP_0FD9D664C_96FDF78				1	1	1.0KB				
3	HASH GROUP BY		1	1327		1	1					
4	KEY VECTOR CREATE BUFFERED	:KV0000	1	1319		1	1					
5	TABLE ACCESS STORAGE FULL	TAB2	320K	1318		1	320K					
6	SORT AGGREGATE		1			1						
7	HASH JOIN		1	63141		1		1.1MB				
8	VIEW	VW VT 377C5901	1	63139		1						
9	VECTOR GROUP BY		1	63139		1		15.0KB				
-> 10	HASH GROUP BY		1	63139		1	0					
-> 11	KEY VECTOR USE	:KV0000	15M	63138		1	964M	2.9MB				
-> 12	TABLE ACCESS STORAGE FULL	TAB1	15M	63137		1	603K					
13	TABLE ACCESS STORAGE FULL	SYS_TEMP_0FD9D664C_96FDF78	1	2								

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

ORACLE®