ORACLE PARTITIONING FOR DBAS AND DEVELOPERS

WHY, WHEN AND HOW TO DO IT

Franky Weber Faust
May 2022

Pythian

Oracle Certified Expert, Oracle Database 12c: Performance Management and Tuning
Issued by Oracle

Oracle Exadata Database Machine and Cloud Service 2017 Certified Implementation Specialist
Issued by Oracle

Oracle Database 12c Administrator Certified Professional
Issued by Oracle

Oracle Linux 6 Certified Implementation Specialist
Issued by Oracle

Oracle Real Application Clusters 12c Certified Implementation Specialist
Issued by Oracle

Oracle Database SQL Certified Expert
Issued by Oracle

Oracle Database 11g Administrator Certified Associate
Issued by Oracle

Oracle Autonomous Database Cloud 2019 Certified Specialist
Oracle

ORACLE ACE

FRANKY WEBER FAUST

- Lead Database Consultant at Pythian
- 31 years old
- Based in Brazil
- Writer at OTNLA and Lore Data Blog
- Speaker at conferences around the world
- High Availability specialist
- Performance researcher
- Exadata, RAC, DataGuard, GoldenGate
- AcroYoga practitioner
- Guitar player

loredata.com.br

Keep in touch

E-mail: faust@pythian.com or franky@loredata.com.br

Blog: http://loredata.com.br/blog

Facebook: https://facebook.com/08Franky.Weber

Instagram: https://www.instagram.com/frankyweber/

Twitter: https://twitter.com/frankyweber

LinkedIn: https://linkedin.com/in/frankyweber/en

Oracle ACE: https://bit.ly/2YxU6bK

25

Years in Business

450+

Pythian Experts
in 35 Countries

350+

Current Clients
Globally

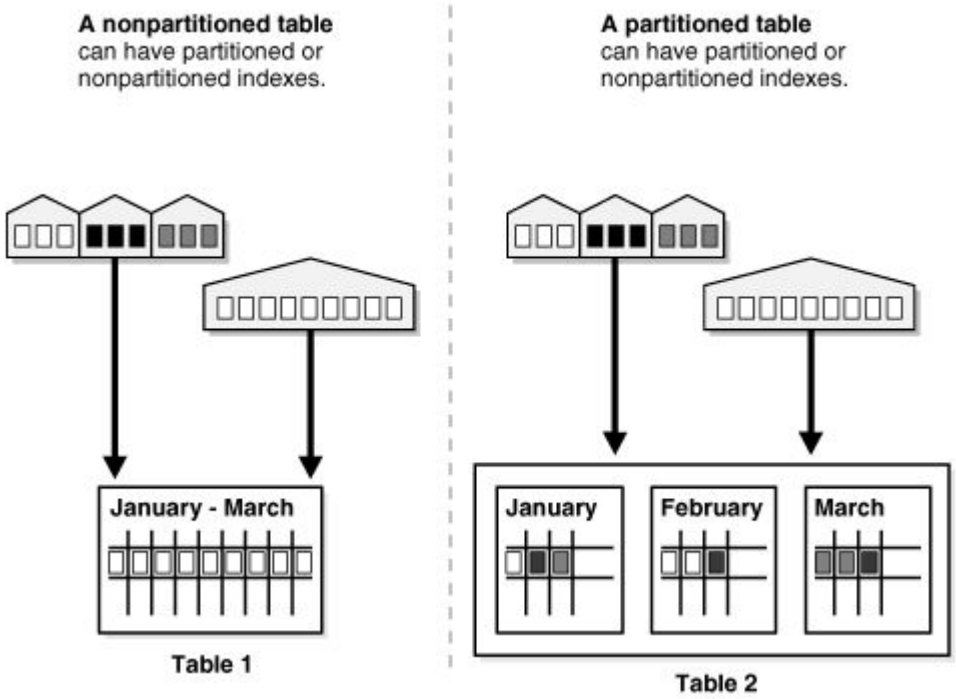
WHY THIS PRESENTATION?

AGENDA

- Partitioning concept
- Background
- Partitioning benefits (Why)
- Implementation scenarios (When)
- Types of partitioning (When)
- Range, List, Interval e Hash partitioning (How)
- The execution plan

PARTITIONING CONCEPTS

- Break down a table/index into smaller pieces
- Partition Key
  - Single or multicolumn
  - Up to 16 columns
  - Unique values are not required
- Every partition is a segment
- Tables
  - Heap tables
  - Index-organized tables
- Indexes
- Materialized Views
- Clusters
- Performance
  - Partition Pruning
  - Concurrency on DML



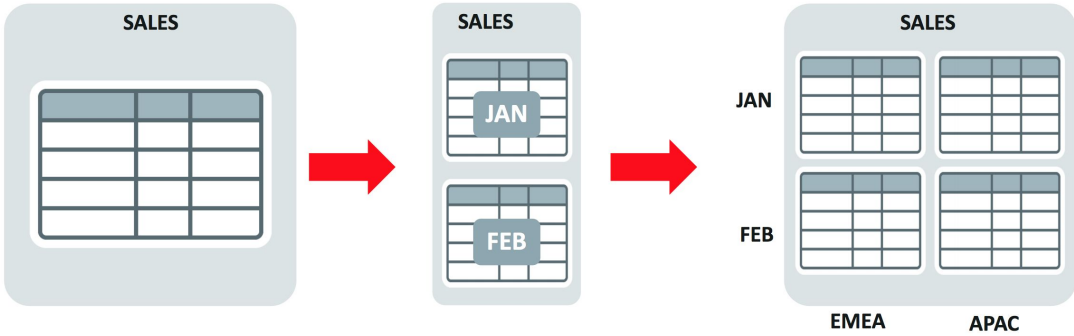
BACKGROUND HISTORY

- Since Oracle 8 (1997)
- Requires extra licensing and EE
- Manual partitioning using views
- New features every new release
- **19c** Hybrid partitioning (Hive, HDFS, External file)

	Core Functionality	Performance	Manageability
Oracle 8.0	Range partitioning Global range indexes	Static partition pruning	Basic maintenance: ADD,DROP,EXCHANGE
Oracle 8*i*	Hash partitioning Range-hash partitioning	Partition-wise joins Dynamic pruning	Expanded maintenance: MERGE
Oracle 9*i*	List partitioning		Global index maintenance
Oracle 9*i* R2	Range-list partitioning	Fast partition SPLIT	
Oracle 10*g*	Global hash indexes		Local index maintenance
Oracle 10*g* R2	1M partitions per table	Multidimensional pruning	Fast DROP TABLE
Oracle 11*g*	Virtual column based partitioning More composite choices REF partitioning		Interval partitioning Partition Advisor Incremental stats management
Oracle 11*g* R2	Hash-hash partitioning	"AND" pruning	Multibranch execution Segment creation on demand*
Oracle 12*c*	Interval-reference partitioning Partial Indexes for partitioning		Partition maintenance on multiple partitions / Online move partition
Oracle 12cR2	Auto-list and Multicolumn list partitioning, Partitioned external tables		Table creation for partition exchange, read-only partitions, Filtered partition maintenance, Online table conversion to partitioned table
Oracle 18c	Modifying the Partitioning Strategy	Parallel Partition-Wise SQL Operations	Online Merging of Partitions and Subpartitions

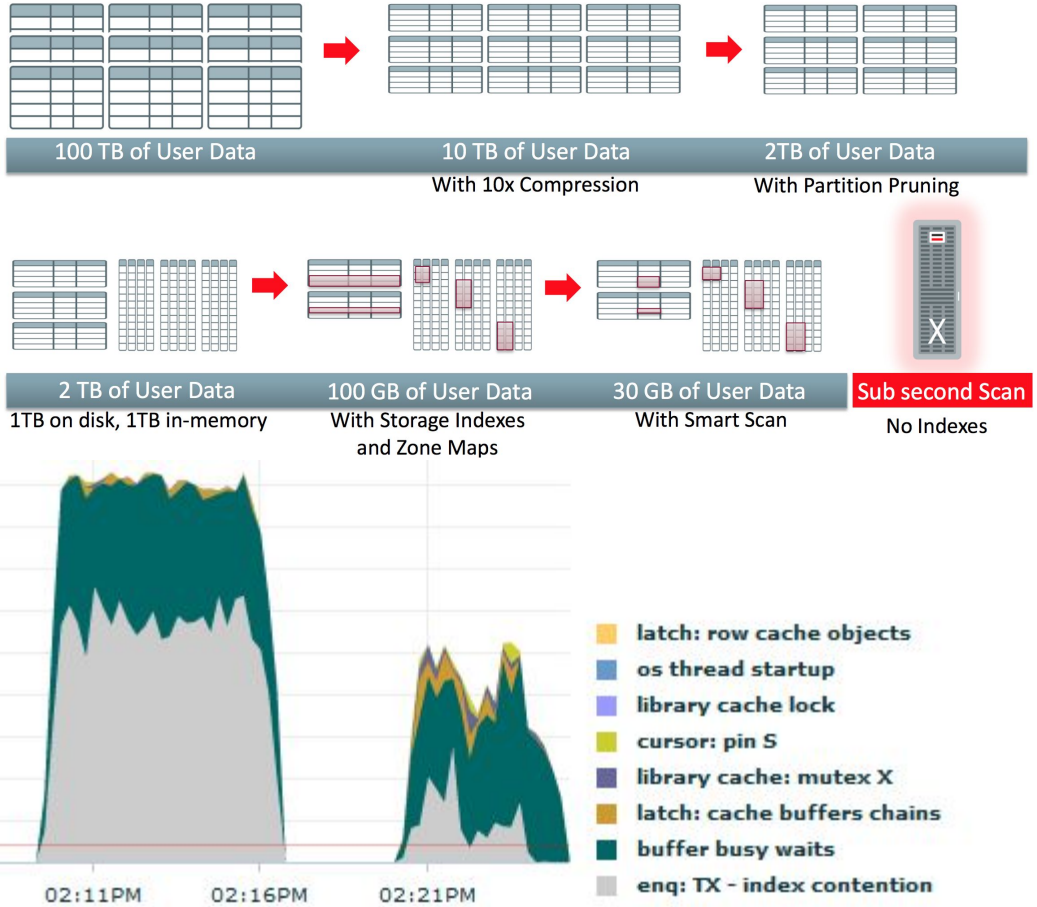
PARTITIONING BENEFITS (WHY)

- Increases availability of data
- Eases administration of large segments
- Might improve the performance of some queries
- Might reduce contention on hot OLTP segments



IMPLEMENTATION SCENARIOS (WHEN)

- Large volume of data when administration becomes difficult
- Query performance on large tables suffers
- Buffer busy waits on high-concurrency tables
- Horizontal scalability in RAC
- Decreased storage cost
- Compression and row archival



TYPES OF PARTITIONING (HOW)

- Range Partitioning
  - Dates and numbers are good candidates
  - Upper boundary
- Hash Partitioning
  - High-concurrency OLTP segments
  - Data is evenly distributed across partitions
- List Partitioning
  - Categories, colors, sports, brands
  - Default values accepted
- Interval Partitioning
  - Automated range partitions
- Reference Partitioning
  - Relational model PK-FK
- Interval Reference Partitioning
  - All child tables will be automatically maintained
- Virtual Column Based Partitioning
- Interval List Partitioning
  - Automated list partitions
- Composite Partition
  - Partition + Subpartition
  - E.g.: Range + Hash
- **Hybrid Partitioning (19c)**

RANGE PARTITIONING (HOW)

```
CREATE TABLE sales_part (
    id              NUMBER,
    flag            NUMBER,
    product         CHAR(14),
    channel_id      NUMBER,
    cust_id         NUMBER,
    amount_sold     NUMBER,
    order_date      DATE,
    ship_date       DATE
)
    PARTITION BY RANGE ( order_date )
    (
    PARTITION before_2017     VALUES LESS THAN ( TO_DATE('01/01/2017','dd/mm/yyyy') ),
    PARTITION p2017_01        VALUES LESS THAN ( TO_DATE('01/02/2017','dd/mm/yyyy') ),
    PARTITION p2017_02        VALUES LESS THAN ( TO_DATE('01/03/2017','dd/mm/yyyy') ),
    PARTITION p2017_03        VALUES LESS THAN ( TO_DATE('01/04/2017','dd/mm/yyyy') ),
    PARTITION p2017_04        VALUES LESS THAN ( TO_DATE('01/05/2017','dd/mm/yyyy') ),
    PARTITION p2017_05        VALUES LESS THAN ( TO_DATE('01/06/2017','dd/mm/yyyy') ),
    PARTITION p2017_06        VALUES LESS THAN ( TO_DATE('01/07/2017','dd/mm/yyyy') ),
    PARTITION p2017_07        VALUES LESS THAN ( TO_DATE('01/08/2017','dd/mm/yyyy') ),
    PARTITION p2017_08        VALUES LESS THAN ( TO_DATE('01/09/2017','dd/mm/yyyy') ),
    PARTITION p2017_09        VALUES LESS THAN ( TO_DATE('01/10/2017','dd/mm/yyyy') ),
    PARTITION p2017_10        VALUES LESS THAN ( TO_DATE('01/11/2017','dd/mm/yyyy') ),
    PARTITION p2017_11        VALUES LESS THAN ( TO_DATE('01/12/2017','dd/mm/yyyy') ),
    PARTITION p2017_12        VALUES LESS THAN ( TO_DATE('01/01/2018','dd/mm/yyyy') ),
    PARTITION p2018_01        VALUES LESS THAN ( TO_DATE('01/02/2018','dd/mm/yyyy') ),
    PARTITION p2018_02        VALUES LESS THAN ( TO_DATE('01/03/2018','dd/mm/yyyy') ),
    PARTITION p2018_03        VALUES LESS THAN ( TO_DATE('01/04/2018','dd/mm/yyyy') ),
    PARTITION p2018_04        VALUES LESS THAN ( TO_DATE('01/05/2018','dd/mm/yyyy') ),
    PARTITION p2018_05        VALUES LESS THAN ( TO_DATE('01/06/2018','dd/mm/yyyy') ),
    PARTITION p2018_06        VALUES LESS THAN ( TO_DATE('01/07/2018','dd/mm/yyyy') )
    );
```

RANGE PARTITIONING (HOW)

PNAME	POS	HIGH_VALUE	NUM_ROWS
BEFORE_2017	1	TO_DATE(' 2017-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	19024
P2017_01	2	TO_DATE(' 2017-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
P2017_02	3	TO_DATE(' 2017-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	56
P2017_03	4	TO_DATE(' 2017-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
P2017_04	5	TO_DATE(' 2017-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
P2017_05	6	TO_DATE(' 2017-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
P2017_06	7	TO_DATE(' 2017-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
P2017_07	8	TO_DATE(' 2017-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
P2017_08	9	TO_DATE(' 2017-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
P2017_09	10	TO_DATE(' 2017-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
P2017_10	11	TO_DATE(' 2017-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
P2017_11	12	TO_DATE(' 2017-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
P2017_12	13	TO_DATE(' 2018-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
P2018_01	14	TO_DATE(' 2018-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
P2018_02	15	TO_DATE(' 2018-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	56
P2018_03	16	TO_DATE(' 2018-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
P2018_04	17	TO_DATE(' 2018-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
P2018_05	18	TO_DATE(' 2018-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	6
P2018_06	19	TO_DATE(' 2018-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	0

LIST PARTITIONING (HOW)

```
CREATE TABLE cars_part (
    id      NUMBER,
    car     VARCHAR(100),
    model   VARCHAR(30),
    brand   VARCHAR(100),
    price   NUMBER
)
    PARTITION BY LIST ( brand ) ( PARTITION bmw VALUES ( 'BMW' ),
        PARTITION ford VALUES ( 'FORD' ),
        PARTITION fiat VALUES ( 'FIAT' ),
        PARTITION ferrari VALUES ( 'FERRARI' ),
        PARTITION toyota VALUES ( 'TOYOTA' ),
        PARTITION honda VALUES ( 'HONDA' ),
        PARTITION kia VALUES ( 'KIA' ),
        PARTITION mercedes VALUES ( 'MERCEDES' ),
        PARTITION landrover VALUES ( 'LANDROVER' ),
        PARTITION mini VALUES ( 'MINI' ),
        PARTITION vw_ag VALUES ( 'AUDI','LAMBORGHINI','BUGATTI','BENTLEY','PORSCHE','SEAT','VOLKSWAGEN' ),
        PARTITION all_others VALUES ( DEFAULT )
    );
```

LIST PARTITIONING (HOW)

```
insert into cars_part values (1, 'FOCUS','mmmmmmmmm','FORD',50000);
insert into cars_part values (2, 'A3','mmmmmmmmm','AUDI',50000);
insert into cars_part values (3, 'C 180','mmmmmmmmm','MERCEDES',50000);
insert into cars_part values (4, 'CERATO','mmmmmmmmm','KIA',50000);
insert into cars_part values (5, 'HILUX','mmmmmmmmm','TOYOTA',50000);
insert into cars_part values (6, 'DEFENDER','mmmmmmmmm','LANDROVER',50000);
insert into cars_part values (7, 'CIVIC','mmmmmmmmm','HONDA',50000);
insert into cars_part values (8, 'BRAVO','mmmmmmmmm','FIAT',50000);
insert into cars_part values (9, 'COOPER S','mmmmmmmmm','MINI',50000);
insert into cars_part values (10,'COOPER','mmmmmmmmm','MINI',50000);
insert into cars_part values (11,'VEYRON','mmmmmmmmm','BUGATTI',50000);
insert into cars_part values (12,'VYPER','mmmmmmmmm','DODGE',50000);
insert into cars_part values (13,'300C','mmmmmmmmm','CHRYSLER',50000);
insert into cars_part values (14,'GOLF','mmmmmmmmm','VOLKSWAGEN',50000);
insert into cars_part values (15,'JETTA','mmmmmmmmm','VOLKSWAGEN',50000);
insert into cars_part values (16,'POLO','mmmmmmmmm','VOLKSWAGEN',50000);
insert into cars_part values (17,'AMAROK','mmmmmmmmm','VOLKSWAGEN',50000);
insert into cars_part values (18,'TIGUAN','mmmmmmmmm','VOLKSWAGEN
```

PNAME	POS	HIGH_VALUE	NUM_ROWS
BMW	1	'BMW'	0
FORD	2	'FORD'	1
FIAT	3	'FIAT'	1
FERRARI	4	'FERRARI'	0
TOYOTA	5	'TOYOTA'	1
HONDA	6	'HONDA'	1
KIA	7	'KIA'	1
MERCEDES	8	'MERCEDES'	1
LANDROVER	9	'LANDROVER'	1
MINI	10	'MINI'	2
VW_AG	11	'AUDI', 'LAMBORGHINI', 'BUGATTI', 'BENTLEY', 'PORSCHE', 'SEAT', 'VOLKSWAGEN'	7
ALL_OTHERS	12	DEFAULT	2

HASH PARTITIONING (HOW)

```
CREATE TABLE cars_hashpart (
    id      NUMBER,
    car     VARCHAR(100),
    model   VARCHAR(30),
    brand   VARCHAR(100),
    price   NUMBER
)
    PARTITION BY HASH ( brand ) (
    PARTITION p1,
    PARTITION p2,
    PARTITION p3,
    PARTITION p4,
    PARTITION p5,
    PARTITION p6,
    PARTITION p7 );
```

HASH PARTITIONING (HOW)

PNAME	POS	HIGH_VALUE	NUM_ROWS
P1	1	–	2
P2	2	–	1
P3	3	–	5
P4	4	–	7
P5	5	–	2
P6	6	–	1
P7	7	–	0

HASH PARTITIONING (HOW)

```
CREATE TABLE cars_hashpart2 (
    id      NUMBER,
    car     VARCHAR(100),
    model   VARCHAR(30),
    brand   VARCHAR(100),
    price   NUMBER
)
    PARTITION BY HASH ( brand ) (
    PARTITION p1,
    PARTITION p2,
    PARTITION p3,
    PARTITION p4,
    PARTITION p5,
    PARTITION p6 );
```

HASH PARTITIONING (HOW)

PNAME	POS	HIGH_VALUE	NUM_ROWS
P1	1	–	2
P2	2	–	1
P3	3	–	5
P4	4	–	7
P5	5	–	2
P6	6	–	1

HASH PARTITIONING (HOW)

```
CREATE TABLE cars_hashpart3 (
    id      NUMBER,
    car     VARCHAR(100),
    model   VARCHAR(30),
    brand   VARCHAR(100),
    price   NUMBER
)
    PARTITION BY HASH ( brand ) (
    PARTITION p1,
    PARTITION p2,
    PARTITION p3,
    PARTITION p4,
    PARTITION p5,
    PARTITION p6,
    PARTITION p7,
    PARTITION p8 );
```

HASH PARTITIONING (HOW)

PNAME	POS	HIGH_VALUE	NUM_ROWS
P1	1	–	2
P2	2	–	1
P3	3	–	5
P4	4	–	6
P5	5	–	2
P6	6	–	1
P7	7	–	0
P8	8	–	1

HASH PARTITIONING (HOW)

```
CREATE TABLE cars_hashpart4 (
    id      NUMBER,
    car     VARCHAR(100),
    model   VARCHAR(30),
    brand   VARCHAR(100),
    price   NUMBER
)
    PARTITION BY HASH ( id ) (
    PARTITION p1,
    PARTITION p2,
    PARTITION p3,
    PARTITION p4,
    PARTITION p5,
    PARTITION p6,
    PARTITION p7,
    PARTITION p8 );
```

HASH PARTITIONING (HOW)

PNAME	POS	HIGH_VALUE	NUM_ROWS
P1	1	–	2
P2	2	–	1
P3	3	–	1
P4	4	–	3
P5	5	–	1
P6	6	–	3
P7	7	–	3
P8	8	–	4

HASH PARTITIONING (HOW)

```
CREATE TABLE sales_hashpart (
    id              NUMBER,
    flag            NUMBER,
    product         CHAR(14),
    channel_id      NUMBER,
    cust_id         NUMBER,
    amount_sold     NUMBER,
    order_date      DATE,
    ship_date       DATE
)
    PARTITION BY HASH ( id ) PARTITIONS 8;
```

HASH PARTITIONING (HOW)

PNAME	POS	HIGH_VALUE	NUM_ROWS
SYS_P5579	1	-	2482
SYS_P5580	2	-	2512
SYS_P5581	3	-	2563
SYS_P5582	4	-	2469
SYS_P5583	5	-	2437
SYS_P5584	6	-	2472
SYS_P5585	7	-	2557
SYS_P5586	8	-	2508

HASH PARTITIONING (HOW)

PNAME	POS	HIGH_VALUE	NUM_ROWS
SYS_P5587	1	–	2482
SYS_P5588	2	–	2512
SYS_P5589	3	–	2563
SYS_P5590	4	–	4977
SYS_P5591	5	–	2437
SYS_P5592	6	–	2472
SYS_P5593	7	–	2557

INTERVAL PARTITIONING (HOW)

```
CREATE TABLE sales_intervalpart (
    id              NUMBER,
    flag            NUMBER,
    product         CHAR(14),
    channel_id      NUMBER,
    cust_id         NUMBER,
    amount_sold     NUMBER,
    order_date      DATE,
    ship_date       DATE
)
    PARTITION BY RANGE ( order_date ) INTERVAL ( numtoyminterval(1,'month') )
    (
    PARTITION before_2017
        VALUES LESS THAN ( TO_DATE('01/01/2017','dd/mm/yyyy') )
    );
```

INTERVAL PARTITIONING (HOW)

PNAME	POS	HIGH_VALUE	NUM_ROWS
BEFORE_2017	1	TO_DATE(' 2017-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	19024
SYS_P5611	2	TO_DATE(' 2017-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5612	3	TO_DATE(' 2017-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	56
SYS_P5613	4	TO_DATE(' 2017-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5614	5	TO_DATE(' 2017-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5615	6	TO_DATE(' 2017-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5616	7	TO_DATE(' 2017-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5617	8	TO_DATE(' 2017-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5618	9	TO_DATE(' 2017-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5619	10	TO_DATE(' 2017-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5620	11	TO_DATE(' 2017-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5621	12	TO_DATE(' 2017-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5622	13	TO_DATE(' 2018-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5623	14	TO_DATE(' 2018-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5624	15	TO_DATE(' 2018-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	56
SYS_P5625	16	TO_DATE(' 2018-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5626	17	TO_DATE(' 2018-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5627	18	TO_DATE(' 2018-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	6

INTERVAL PARTITIONING (HOW)

```
insert into sales_intervalpart values (31497139,31497140,'Apple MacBook', 8, 198, 12475, '10/07/2018', '14/10/2018' );
insert into sales_intervalpa
insert into sales_intervalpa
insert into sales_intervalpa
insert into sales_intervalpa
insert into sales_intervalpa
insert into sales_intervalpa
insert into sales_intervalpa
insert into sales_intervalpa
commit;
```

PNAME	POS	HIGH_VALUE	NUM_ROWS
BEFORE_2017	1	TO_DATE(' 2017-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	19024
SYS_P5611	2	TO_DATE(' 2017-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5612	3	TO_DATE(' 2017-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	56
SYS_P5613	4	TO_DATE(' 2017-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5614	5	TO_DATE(' 2017-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5615	6	TO_DATE(' 2017-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5616	7	TO_DATE(' 2017-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5617	8	TO_DATE(' 2017-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5618	9	TO_DATE(' 2017-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5619	10	TO_DATE(' 2017-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5620	11	TO_DATE(' 2017-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5621	12	TO_DATE(' 2017-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5622	13	TO_DATE(' 2018-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5623	14	TO_DATE(' 2018-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5624	15	TO_DATE(' 2018-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	56
SYS_P5625	16	TO_DATE(' 2018-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5626	17	TO_DATE(' 2018-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5627	18	TO_DATE(' 2018-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	6
SYS_P5628	19	TO_DATE(' 2018-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	2
SYS_P5629	20	TO_DATE(' 2018-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5630	21	TO_DATE(' 2018-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5631	22	TO_DATE(' 2018-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5632	23	TO_DATE(' 2018-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5633	24	TO_DATE(' 2019-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5634	25	TO_DATE(' 2019-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5635	26	TO_DATE(' 2019-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1

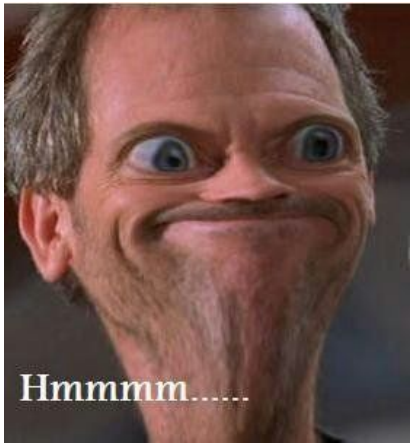
INTERVAL PARTITIONING (HOW)

```
insert into sales_intervalpart values (5149908712,5149908713,'Apple MacBook', 9, 191, 12475, '10/02/2021', '14/02/2021' );
commit;
```

PNAME	POS	HIGH_VALUE	NUM_ROWS
BEFORE_2017	1	TO_DATE(' 2017-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	19024
SYS_P5611	2	TO_DATE(' 2017-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5612	3	TO_DATE(' 2017-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	56
SYS_P5613	4	TO_DATE(' 2017-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5614	5	TO_DATE(' 2017-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5615	6	TO_DATE(' 2017-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5616	7	TO_DATE(' 2017-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5617	8	TO_DATE(' 2017-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5618	9	TO_DATE(' 2017-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5619	10	TO_DATE(' 2017-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5620	11	TO_DATE(' 2017-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5621	12	TO_DATE(' 2017-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5622	13	TO_DATE(' 2018-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5623	14	TO_DATE(' 2018-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5624	15	TO_DATE(' 2018-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	56
SYS_P5625	16	TO_DATE(' 2018-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5626	17	TO_DATE(' 2018-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5627	18	TO_DATE(' 2018-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	6
SYS_P5628	19	TO_DATE(' 2018-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	2
SYS_P5629	20	TO_DATE(' 2018-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5630	21	TO_DATE(' 2018-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5631	22	TO_DATE(' 2018-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5632	23	TO_DATE(' 2018-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5633	24	TO_DATE(' 2019-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5634	25	TO_DATE(' 2019-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5635	26	TO_DATE(' 2019-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5636	27	TO_DATE(' 2021-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1

INTERVAL PARTITIONING (HOW)

```
insert into sales_intervalpart
commit;
```



PNAME	POS	HIGH_VALUE	NUM_ROWS
BEFORE_2017	1	TO_DATE(' 2017-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	19024
SYS_P5611	2	TO_DATE(' 2017-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5612	3	TO_DATE(' 2017-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	56
SYS_P5613	4	TO_DATE(' 2017-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5614	5	TO_DATE(' 2017-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5615	6	TO_DATE(' 2017-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5616	7	TO_DATE(' 2017-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5617	8	TO_DATE(' 2017-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5618	9	TO_DATE(' 2017-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5619	10	TO_DATE(' 2017-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5620	11	TO_DATE(' 2017-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5621	12	TO_DATE(' 2017-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5622	13	TO_DATE(' 2018-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5623	14	TO_DATE(' 2018-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5624	15	TO_DATE(' 2018-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	56
SYS_P5625	16	TO_DATE(' 2018-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	62
SYS_P5626	17	TO_DATE(' 2018-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	60
SYS_P5627	18	TO_DATE(' 2018-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	6
SYS_P5628	19	TO_DATE(' 2018-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	2
SYS_P5629	20	TO_DATE(' 2018-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5630	21	TO_DATE(' 2018-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5631	22	TO_DATE(' 2018-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5632	23	TO_DATE(' 2018-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5633	24	TO_DATE(' 2019-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5634	25	TO_DATE(' 2019-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5635	26	TO_DATE(' 2019-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5637	27	TO_DATE(' 2020-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1
SYS_P5636	28	TO_DATE(' 2021-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	1

```
'10/02/2020', '14/02/2021' );
```

THE EXECUTION PLAN

PNAME	POS	HIGH_VALUE	NUM_ROWS
BEFORE_2017	1	TO_DATE(' 2017-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	45115
P2017_01	2	TO_DATE(' 2017-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	155
P2017_02	3	TO_DATE(' 2017-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	140
P2017_03	4	TO_DATE(' 2017-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	155
P2017_04	5	TO_DATE(' 2017-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	150
P2017_05	6	TO_DATE(' 2017-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	155
P2017_06	7	TO_DATE(' 2017-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	150
P2017_07	8	TO_DATE(' 2017-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	155
P2017_08	9	TO_DATE(' 2017-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	155
P2017_09	10	TO_DATE(' 2017-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	150
P2017_10	11	TO_DATE(' 2017-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	155
P2017_11	12	TO_DATE(' 2017-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	150
P2017_12	13	TO_DATE(' 2018-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	155
P2018_01	14	TO_DATE(' 2018-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	155
P2018_02	15	TO_DATE(' 2018-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	140
P2018_03	16	TO_DATE(' 2018-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	155
P2018_04	17	TO_DATE(' 2018-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	150
P2018_05	18	TO_DATE(' 2018-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	155
P2018_06	19	TO_DATE(' 2018-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	150
PMAX	20	MAXVALUE	2155

THE EXECUTION PLAN

ID	FLAG	PRODUCT	CHANNEL_ID	CUST_ID	AMOUNT_SOLD	ORDER_DATE	SHIP_DATE
215	216	Oracle Exadata X8-2 Full Rack	0	215	5000	20-NOV-92	21-NOV-92
216	217	Oracle Exadata X8-2 Full Rack	1	216	5000	21-NOV-92	22-NOV-92
217	218	Oracle Exadata X8-2 Full Rack	2	217	5000	22-NOV-92	23-NOV-92
218	219	Oracle Exadata X8-2 Full Rack	3	218	5000	23-NOV-92	24-NOV-92
219	220	Oracle Exadata X8-2 Full Rack	4	219	5000	24-NOV-92	25-NOV-92
220	221	Oracle Exadata X8-2 Full Rack	0	220	5000	25-NOV-92	26-NOV-92
221	222	Oracle Exadata X8-2 Full Rack	1	221	5000	26-NOV-92	27-NOV-92
222	223	Oracle Exadata X8-2 Full Rack	2	222	5000	27-NOV-92	28-NOV-92
223	224	Oracle Exadata X8-2 Full Rack	3	223	5000	28-NOV-92	29-NOV-92
224	225	Oracle Exadata X8-2 Full Rack	4	224	5000	29-NOV-92	30-NOV-92
225	226	Oracle Exadata X8-2 Full Rack	0	225	5000	30-NOV-92	01-DEC-92
226	227	Oracle Exadata X8-2 Full Rack	1	226	5000	01-DEC-92	02-DEC-92
227	228	Oracle Exadata X8-2 Full Rack	2	227	5000	02-DEC-92	03-DEC-92
228	229	Oracle Exadata X8-2 Full Rack	3	228	5000	03-DEC-92	04-DEC-92
229	230	Oracle Exadata X8-2 Full Rack	4	229	5000	04-DEC-92	05-DEC-92
230	231	Oracle Exadata X8-2 Full Rack	0	230	5000	05-DEC-92	06-DEC-92
231	232	Oracle Exadata X8-2 Full Rack	1	231	5000	06-DEC-92	07-DEC-92
232	233	Oracle Exadata X8-2 Full Rack	2	232	5000	07-DEC-92	08-DEC-92
233	234	Oracle Exadata X8-2 Full Rack	3	233	5000	08-DEC-92	09-DEC-92
234	235	Oracle Exadata X8-2 Full Rack	4	234	5000	09-DEC-92	10-DEC-92

THE EXECUTION PLAN

```
PLAN_TABLE_OUTPUT
SQL_ID  20ydmgyuhwp61, child number 0
-------------------------------------
select /*My statement*/ /*+ gather_plan_statistics */ sum(amount_sold)
  from sales_part    where order_date between    '01/Feb/2015' and
'31/Jul/2015'

Plan hash value: 3843654642

-----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation               | Name       | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |            |      1 |        |       |   161 (100)|          |       |       |      1 |00:00:00.01 |     471 |
|   1 |  SORT AGGREGATE         |            |      1 |      1 |    11 |            |          |       |       |      1 |00:00:00.01 |     471 |
|   2 |   PARTITION RANGE SINGLE|            |      1 |    910 | 10010 |   161   (1)| 00:00:01 |     1 |     1 |    905 |00:00:00.01 |     471 |
|*  3 |    TABLE ACCESS FULL    | SALES_PART |      1 |    910 | 10010 |   161   (1)| 00:00:01 |     1 |     1 |    905 |00:00:00.01 |     471 |
-----------------------------------------------------------------------------------------------------------------------------------
```

THE EXECUTION PLAN

```
PLAN_TABLE_OUTPUT
SQL_ID  f40w4vpjta20p, child number 0
-------------------------------------
select /*My statement*/ /*+ gather_plan_statistics */ sum(amount_sold)
  from sales_part    where order_date between    '01/Jul/2015' and
'31/Jul/2015'

Plan hash value: 3843654642

-----------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name       | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |            |      1 |        |       |   161 (100)|          |       |       |      1 |00:00:00.01 |     471 |
|   1 |  SORT AGGREGATE          |            |      1 |      1 |    11 |            |          |       |       |      1 |00:00:00.01 |     471 |
|   2 |   PARTITION RANGE SINGLE|            |      1 |    160 |  1760 |   161   (1)| 00:00:01 |     1 |     1 |    155 |00:00:00.01 |     471 |
|*  3 |    TABLE ACCESS FULL     | SALES_PART |      1 |    160 |  1760 |   161   (1)| 00:00:01 |     1 |     1 |    155 |00:00:00.01 |     471 |
-----------------------------------------------------------------------------------------------------------------------------------------
```

THE EXECUTION PLAN

```
PLAN_TABLE_OUTPUT
SQL_ID  17f15ybps0g29, child number 0
-------------------------------------
select /*My statement*/ /*+ gather_plan_statistics */ sum(amount_sold)
  from sales_part    where order_date between    '30/Jul/2015' and
'31/Jul/2015'

Plan hash value: 3843654642

-----------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name       | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |            |      1 |        |       |   161 (100)|          |       |       |      1 |00:00:00.01 |     471 |
|   1 |  SORT AGGREGATE          |            |      1 |      1 |    11 |            |          |       |       |      1 |00:00:00.01 |     471 |
|   2 |   PARTITION RANGE SINGLE|            |      1 |     15 |   165 |   161   (1)| 00:00:01 |     1 |     1 |     10 |00:00:00.01 |     471 |
|*  3 |    TABLE ACCESS FULL     | SALES_PART |      1 |     15 |   165 |   161   (1)| 00:00:01 |     1 |     1 |     10 |00:00:00.01 |     471 |
-----------------------------------------------------------------------------------------------------------------------------------------
```

THE EXECUTION PLAN

```
PLAN_TABLE_OUTPUT
SQL_ID  17f15ybps0g29, child number 0
-------------------------------------
select /*My statement*/ /*+ gather_plan_statistics */ sum(amount_sold)
  from sales_part    where order_date between    '30/Jul/2015' and
'31/Jul/2015'

Plan hash value: 3843654642

-----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name       | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |            |      1 |        |       |   161 (100)|          |       |       |      1 |00:00:00.01 |     471 |
|   1 |  SORT AGGREGATE          |            |      1 |      1 |    11 |            |          |       |       |      1 |00:00:00.01 |     471 |
|   2 |   PARTITION RANGE SINGLE|            |      1 |     15 |   165 |   161   (1)| 00:00:01 |     1 |     1 |     10 |00:00:00.01 |     471 |
|*  3 |    TABLE ACCESS FULL     | SALES_PART |      1 |     15 |   165 |   161   (1)| 00:00:01 |     1 |     1 |     10 |00:00:00.01 |     471 |
-----------------------------------------------------------------------------------------------------------------------------------
```

THE EXECUTION PLAN

```
PLAN_TABLE_OUTPUT
SQL_ID  b07ngs4mjfh2n, child number 0
-------------------------------------
select /*My statement*/ /*+ gather_plan_statistics */ sum(amount_sold)
  from sales_part    where order_date between    '01/Feb/2017' and
'31/Jul/2017'

Plan hash value: 308944665

-----------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name       | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |            |      1 |        |       |    11 (100)|          |       |       |      1 |00:00:00.01 |      36 |
|   1 |  SORT AGGREGATE          |            |      1 |      1 |    11 |            |          |       |       |      1 |00:00:00.01 |      36 |
|   2 |   PARTITION RANGE ITERATOR|           |      1 |    910 | 10010 |    11   (0)| 00:00:01 |     3 |     8 |    905 |00:00:00.01 |      36 |
|* 3 |    TABLE ACCESS FULL      | SALES_PART |      6 |    910 | 10010 |    11   (0)| 00:00:01 |     3 |     8 |    905 |00:00:00.01 |      36 |
-----------------------------------------------------------------------------------------------------------------------------------------
```

THE EXECUTION PLAN

```
PLAN_TABLE_OUTPUT
SQL_ID  2c101v6jmtp3q, child number 0
-------------------------------------
select /*My statement*/ /*+ gather_plan_statistics */ sum(amount_sold)
  from sales_part    where order_date between    '01/Jul/2017' and
'31/Jul/2017'

Plan hash value: 3843654642

-----------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name       | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |            |      1 |        |       |     3 (100)|          |       |       |      1 |00:00:00.01 |       6 |
|   1 |  SORT AGGREGATE          |            |      1 |      1 |    11 |            |          |       |       |      1 |00:00:00.01 |       6 |
|   2 |   PARTITION RANGE SINGLE|            |      1 |    155 |  1705 |     3   (0)| 00:00:01 |     8 |     8 |    155 |00:00:00.01 |       6 |
|*  3 |    TABLE ACCESS FULL     | SALES_PART |      1 |    155 |  1705 |     3   (0)| 00:00:01 |     8 |     8 |    155 |00:00:00.01 |       6 |
-----------------------------------------------------------------------------------------------------------------------------------------
```

THE EXECUTION PLAN

```
PLAN_TABLE_OUTPUT
SQL_ID  8tgc0jhwg1w31, child number 0
-------------------------------------
select /*My statement*/ /*+ gather_plan_statistics */ sum(amount_sold)
  from sales_part    where order_date between    '30/Jul/2017' and
'31/Jul/2017'

Plan hash value: 3843654642

-----------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name       | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |            |      1 |        |       |     3 (100)|          |       |       |      1 |00:00:00.01 |       6 |
|   1 |  SORT AGGREGATE          |            |      1 |      1 |    11 |            |          |       |       |      1 |00:00:00.01 |       6 |
|   2 |   PARTITION RANGE SINGLE|            |      1 |     10 |   110 |     3   (0)| 00:00:01 |     8 |     8 |     10 |00:00:00.01 |       6 |
|*  3 |    TABLE ACCESS FULL     | SALES_PART |      1 |     10 |   110 |     3   (0)| 00:00:01 |     8 |     8 |     10 |00:00:00.01 |       6 |
-----------------------------------------------------------------------------------------------------------------------------------------
```

THE EXECUTION PLAN

SQL Worksheet

```
1  create index idx_salesp_od on sales_part(order_date);
2  exec dbms_stats.gather_table_stats('','SALES_PART',cascade=>true,no_invalidate=>false);
```

THE EXECUTION PLAN

```
PLAN_TABLE_OUTPUT
SQL_ID  5up8xsq8msytt, child number 0
-------------------------------------
select /*My statement*/ /*+ gather_plan_statistics index(sales_part
idx_salesp_od) */ sum(amount_sold)    from sales_part    where
order_date between    '30/Jul/2017' and '31/Jul/2017'

Plan hash value: 1182382279

-----------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name          | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |               |      1 |        |       |    17 (100)|          |       |       |      1 |00:00:00.01 |       6 |
|   1 |  SORT AGGREGATE                     |               |      1 |      1 |    11 |            |          |       |       |      1 |00:00:00.01 |       6 |
|   2 |   TABLE ACCESS BY GLOBAL INDEX ROWID BATCHED| SALES_PART    |      1 |      9 |    99 |    17   (0)| 00:00:01 |     8 |     8 |     10 |00:00:00.01 |       6 |
|*  3 |    INDEX RANGE SCAN                 | IDX_SALESP_OD |      1 |     15 |       |     2   (0)| 00:00:01 |       |       |     10 |00:00:00.01 |       2 |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
```

THE EXECUTION PLAN

SQL Worksheet

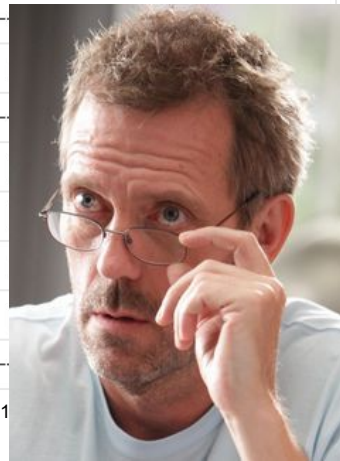
```
1  create index idx_salesp_od_l on sales_part(order_date) local;
2  exec dbms_stats.gather_table_stats('','SALES_PART',cascade=>true,no_invalidate=>false);
```

THE EXECUTION PLAN

```
PLAN_TABLE_OUTPUT
SQL_ID  4yy14f8mzau1b, child number 0
-------------------------------------
select /*My statement*/ /*+ gather_plan_statistics index(sales_part
idx_salesp_od_l) */ sum(amount_sold)    from sales_part    where
order_date between    '30/Jul/2017' and '31/Jul/2017'

Plan hash value: 3760996097

-----------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name           | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                |      1 |        |       |     5 (100)|          |       |       |      1 |00:00:00.01 |       5 |
|   1 |  SORT AGGREGATE                          |                |      1 |      1 |    11 |            |          |       |       |      1 |00:00:00.01 |       5 |
|   2 |   PARTITION RANGE SINGLE                 |                |      1 |      9 |    99 |     5   (0)| 00:00:01 |     8 |     8 |     10 |00:00:00.01 |       5 |
|   3 |    TABLE ACCESS BY LOCAL INDEX ROWID BATCHED| SALES_PART     |      1 |      9 |    99 |     5   (0)| 00:00:01 |     8 |     8 |     10 |00:00:00.01 |       5 |
|*  4 |     INDEX RANGE SCAN                     | IDX_SALESP_OD_L |      1 |      9 |       |     1   (0)| 00:00:01 |     8 |     8 |     10 |00:00:00.01 |       1 |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
```

THE EXECUTION PLAN

```
PLAN_TABLE_OUTPUT
SQL_ID  8tgc0jhwg1w31, child number 0
-------------------------------------
select /*My statement*/ /*+ gather_plan_statistics */ sum(amount_sold)
  from sales_part    where order_date between    '30/Jul/2017' and
'31/Jul/2017'

Plan hash value: 3843654642

-----------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name       | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |
-----------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |            |      1 |        |       |     3 (100)|          |       |       |      1 |
|   1 |  SORT AGGREGATE          |            |      1 |      1 |    11 |            |          |       |       |      1 |
|   2 |   PARTITION RANGE SINGLE|            |      1 |     10 |   110 |     3   (0)| 00:00:01 |     8 |     8 |     10 |
|*  3 |    TABLE ACCESS FULL     | SALES_PART |      1 |     10 |   110 |     3   (0)| 00:00:01 |     8 |     8 |     10 |
-----------------------------------------------------------------------------------------------------------------------------
```



PARTITIONING MAINTENANCE OPERATIONS

```
select segment_name, PARTITION_NAME, segment_type, tablespace_name, bytes/1024/1024 MB from user_segments;

SQL> SQL>
SEGMENT_NAME         PARTITION_NAME       SEGMENT_TYPE       TABLESPACE_NAME                     MB
-------------------- -------------------- ------------------ ------------------------------ ----------
SALES_PART           P2018_05             TABLE PARTITION    USERS                              .0625
SALES_PART           P2018_04             TABLE PARTITION    USERS                              .4375
SALES_PART           P2018_03             TABLE PARTITION    USERS                              .4375
SALES_PART           P2018_02             TABLE PARTITION    USERS                              .4375
SALES_PART           P2018_01             TABLE PARTITION    USERS                              .4375
SALES_PART           P2017_12             TABLE PARTITION    USERS                              .4375
SALES_PART           P2017_11             TABLE PARTITION    USERS                              .4375
SALES_PART           P2017_10             TABLE PARTITION    USERS                              .4375
SALES_PART           P2017_09             TABLE PARTITION    USERS                              .4375
SALES_PART           P2017_08             TABLE PARTITION    USERS                              .4375
SALES_PART           P2017_07             TABLE PARTITION    USERS                              .4375
SALES_PART           P2017_06             TABLE PARTITION    USERS                              .4375
SALES_PART           P2017_05             TABLE PARTITION    USERS                              .4375
SALES_PART           P2017_04             TABLE PARTITION    USERS                              .4375
SALES_PART           P2017_03             TABLE PARTITION    USERS                              .4375
SALES_PART           P2017_02             TABLE PARTITION    USERS                              .4375
SALES_PART           P2017_01             TABLE PARTITION    USERS                              .4375
SALES_PART           BEFORE_2017          TABLE PARTITION    USERS                                120
SALES_INDEXED                             TABLE              USERS                                128
SALES_IDX01                               INDEX              USERS                                 43
SALES                                     TABLE              USERS                                127

21 rows selected.
```


```
alter table sales_part
merge partitions
BEFORE_2017, P2017_01, P2017_02, P2017_03, P2017_04, P2017_05, P2017_06, P2017_07, P2017_08, P2017_09,P2017_10, P2017_11, P2017_12
into partition BEFORE_2018;
```

online

PARTITIONING MAINTENANCE OPERATIONS

```
select segment_name, PARTITION_NAME, segment_type, tablespace_name, bytes/1024/1024 MB from user_segments;
SQL> SQL>
SEGMENT_NAME         PARTITION_NAME       SEGMENT_TYPE       TABLESPACE_NAME                     MB
-------------------- -------------------- ------------------ ------------------------------ ----------
SALES_PART           P2018_05             TABLE PARTITION    USERS                              .0625
SALES_PART           P2018_04             TABLE PARTITION    USERS                              .4375
SALES_PART           P2018_03             TABLE PARTITION    USERS                              .4375
SALES_PART           P2018_02             TABLE PARTITION    USERS                              .4375
SALES_PART           P2018_01             TABLE PARTITION    USERS                              .4375
SALES_PART           BEFORE_2018          TABLE PARTITION    USERS                                127
SALES_INDEXED                             TABLE              USERS                                128
SALES_IDX01                               INDEX              USERS                                 43
SALES                                     TABLE              USERS                                127

9 rows selected.
```

PARTITIONING MAINTENANCE OPERATIONS

online

```
SQL> alter table sales_part move partition BEFORE_2018 row store compress advanced tablespace comp_data;

Table altered.

SQL> select segment_name, PARTITION_NAME, segment_type, tablespace_name, bytes/1024/1024 MB from user_segments;

SEGMENT_NAME         PARTITION_NAME       SEGMENT_TYPE       TABLESPACE_NAME                        MB
-------------------- -------------------- ------------------ ------------------------------ ----------
SALES_PART           P2018_05             TABLE PARTITION    USERS                                .0625
SALES_PART           P2018_04             TABLE PARTITION    USERS                                .4375
SALES_PART           P2018_03             TABLE PARTITION    USERS                                .4375
SALES_PART           P2018_02             TABLE PARTITION    USERS                                .4375
SALES_PART           P2018_01             TABLE PARTITION    USERS                                .4375
SALES_INDEXED                             TABLE              USERS                                 128
SALES_IDX01                               INDEX              USERS                                  43
SALES                                     TABLE              USERS                                 127
SALES_PART           BEFORE_2018          TABLE PARTITION    COMP_DATA                              80

9 rows selected.
```

CONCLUSION

- Range Partitioning, Hash Partitioning, List Partitioning, Interval Partitioning
- 12cR2 onwards can perform most partitioning operations online
- Performance problems in read or write operations can get better or worse with partitioning, so plan well
- New releases -> New features, it's been like that for years.
- **Questions?**

Stay in touch!

- E-mail: franky@loredata.com.br or faust@pythian.com
- Blog: http://loredata.com.br/blog
- Facebook: https://facebook.com/08Franky.Weber
- Instagram: https://www.instagram.com/frankyweber/
- Twitter: https://twitter.com/frankyweber
- LinkedIn: https://linkedin.com/in/frankyweber/en
- Oracle ACE: https://bit.ly/2YxU6bK