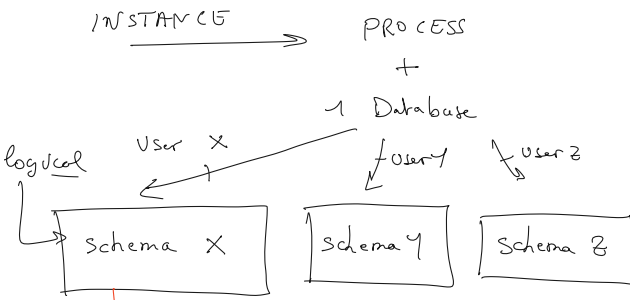
Oracle architecture
No Sql Databases –



Define own storage = own TABLESPACE (not by default)

1) Create 1 Tablespace (logical) (Reference to many datafiles) using 1 Datafile

2) Create new user (uses a system Tablespace by default)

3) Setup the user to user our Tablespace by Default

RDBMS / NoSql

Big Data

Mainframe
Cobol
RPG

New Tech
Java
.net
mssql
mariadb
oracle

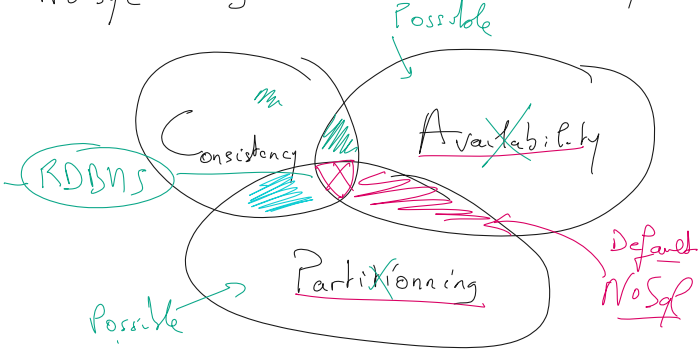
React/HTML5
REST API
NoSql DB.

- Slow upgrade

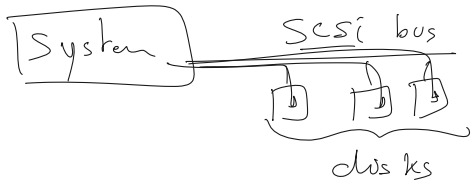
- easy to scale
- Upgrade

RDBMS → guarantee integrity

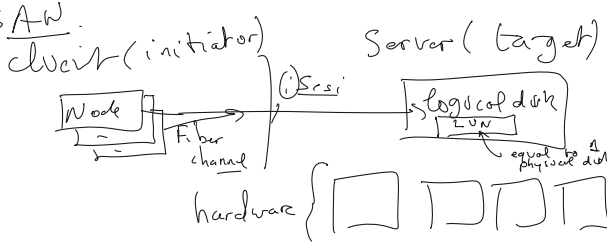
NoSql → guarantee Availability

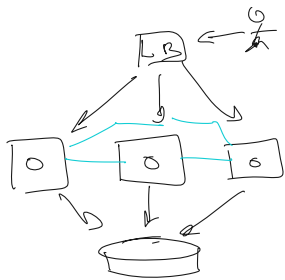


Before:


System
Scsi bus
disks

SAN:


client (initiator)
Server (target)
Node
Fiber channel
iSCSI
logical disk
LUN
equal to physical disk
hardware


LB

"Table like" ≈ SQL → Cassandra

→ no Foreign Keys.

→ Normalization → Anti' patterns.

"Key-values" → Key (unique) → Value (Complex)

→ MongoDb ↳ unmanaged

"Document" → Key → value

↳ manageable

(indexed column)

soft schema

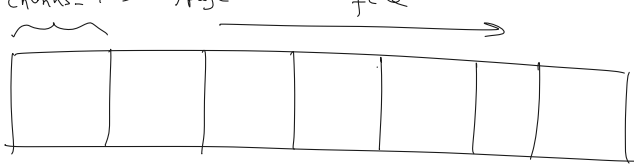
"Graph"

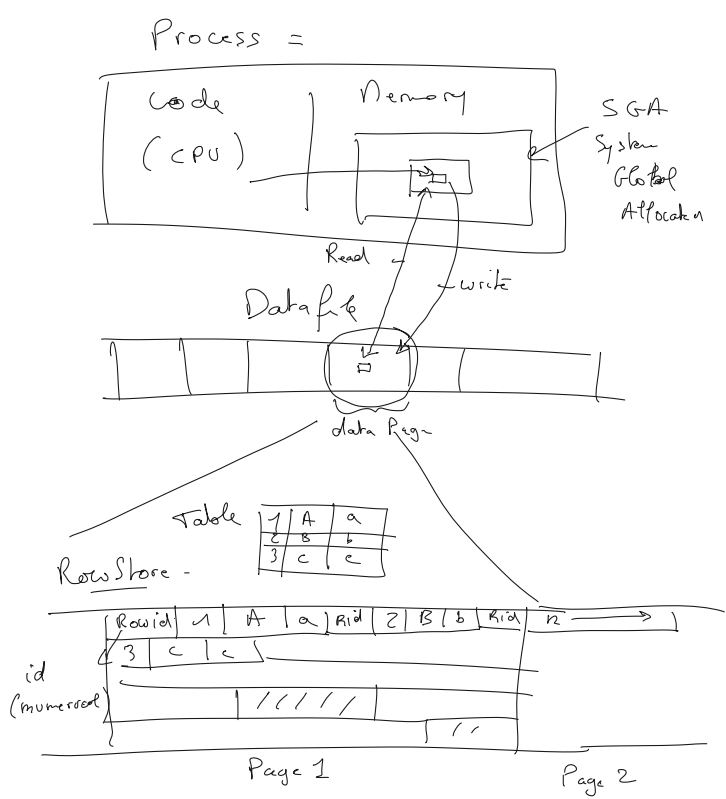
↳ focus on relations (ex: Facebook)

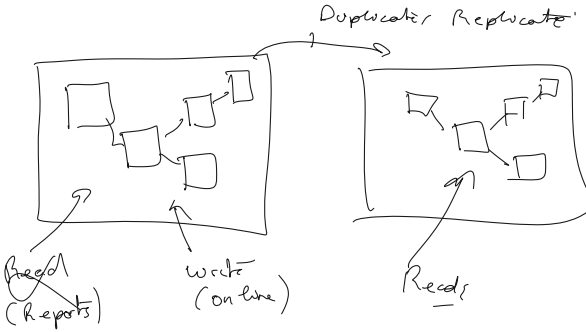
Most Common RDBMS are "row stores"

→ since 70's

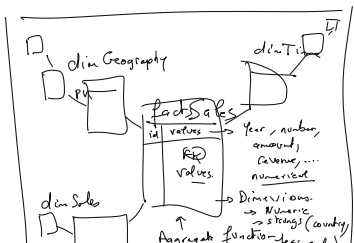
chunks = 1-32 Kb/page




Process =
Code
(CPU)
Memory
SGA
System
Global
Allocation
Read
write
Datafile
data Page
Table
1 A a
2 B b
3 c c
RowStore -
Rowid 1 A a Rid 2 B b Rid n
3 c c
id
(numerical)
Page 1
Page 2

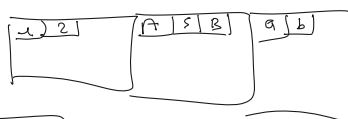


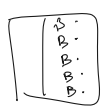
Star - schema

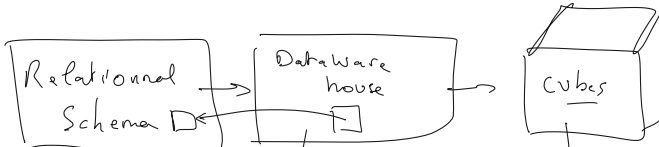


good: column store

Column Store







Columnstore

SQL Language

MDX
DMX

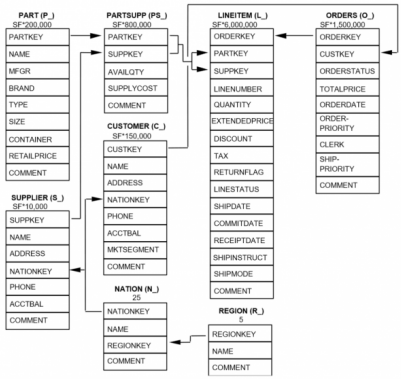
Select * FROM CUST where C = ?

→ ① Plan

where C='FR'
C='BE'
} n

A	tomicity ← Plan / query
C	onsistency -
I	solation -
D	urability -

Exercices :
User : DBT3
pass : password
stored in : /u02/orcl



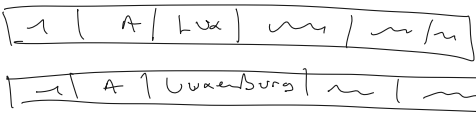
CHAR (50) → 50 character

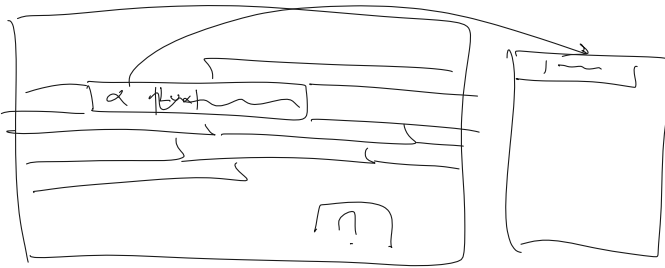
VARCHAR (150) → variable sized
↳ up to 50

Lux → (update) → Luxembourg

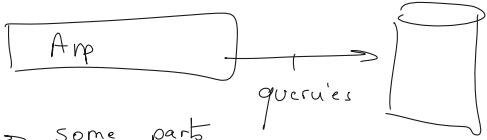
char → Problem → Size = 50/3

Varchar → Ø waste → good !





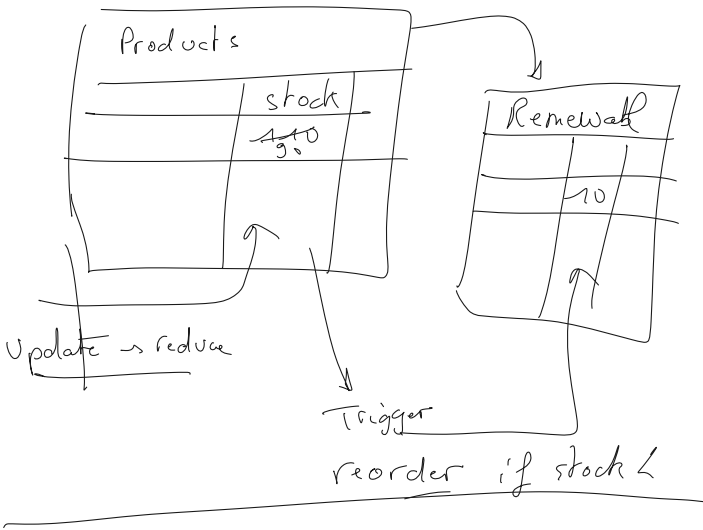
Tuning: The main

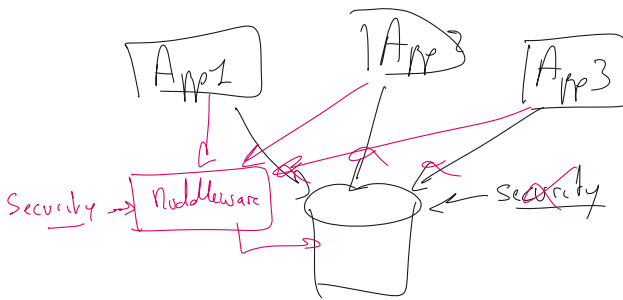


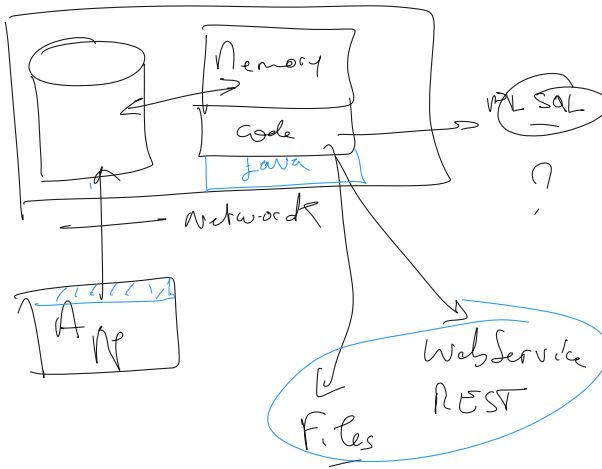
→ some parts
are too slow

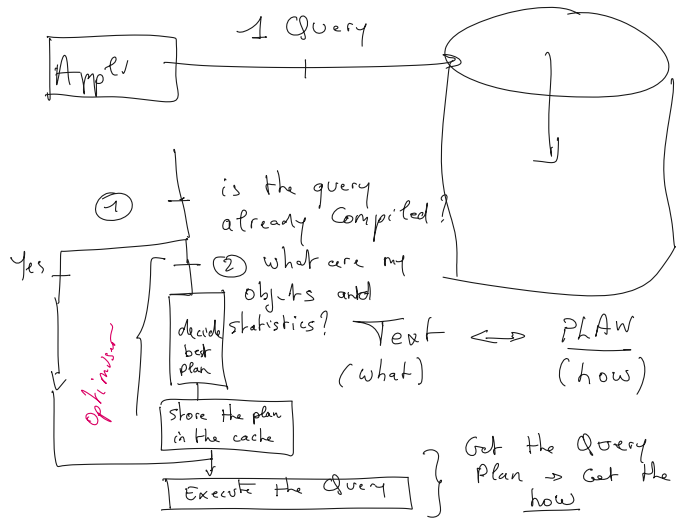
2 Kinds → small queries / many times
→ big queries / few time

→ Gathering query Times
→ Query by execution time
→ Cumulated time by query
} statistics


Products
stock
10
9
Renewal
10
Update → reduce
Trigger
reorder if stock

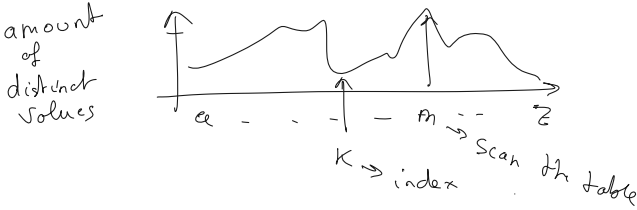

App1
App2
App3
Security
Middleware
Security


Memory
Code
Java
PL SQL
?
Network
API
WebService
REST
Files


1 Query
Appli
1
is the query already Compiled?
Yes
2
what are my objects and statistics?
decide best plan
Optimizer
Store the plan in the cache
Execute the Query
Text
(what)
PLAN
(how)
Get the Query Plan → Get the how

Statistics → amount of rows per table
" of pages
" free space / page

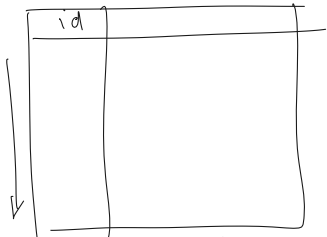
histogram of data



Updating → Asynchronous
→ based on a sample
} somehow, pretty reliable

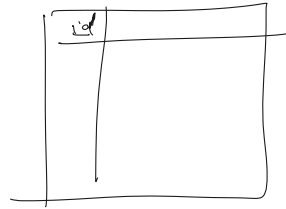
---

no PK

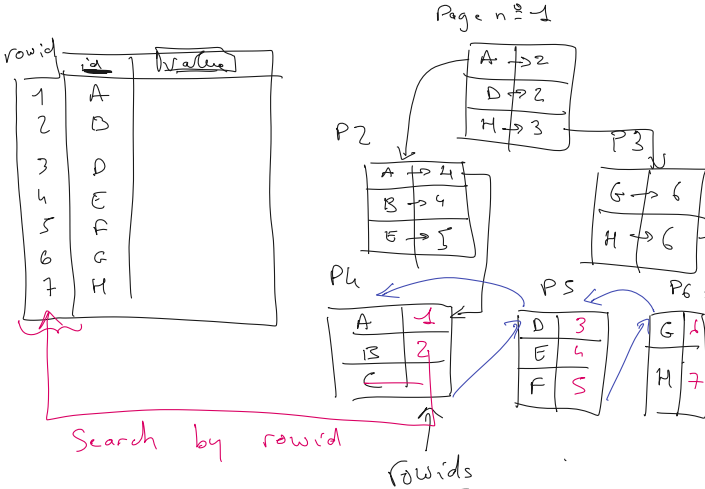


full table scan (always)
(where id=1)

with PK



→ search using the PK index



Leaf value of an index that is not a PK = ?

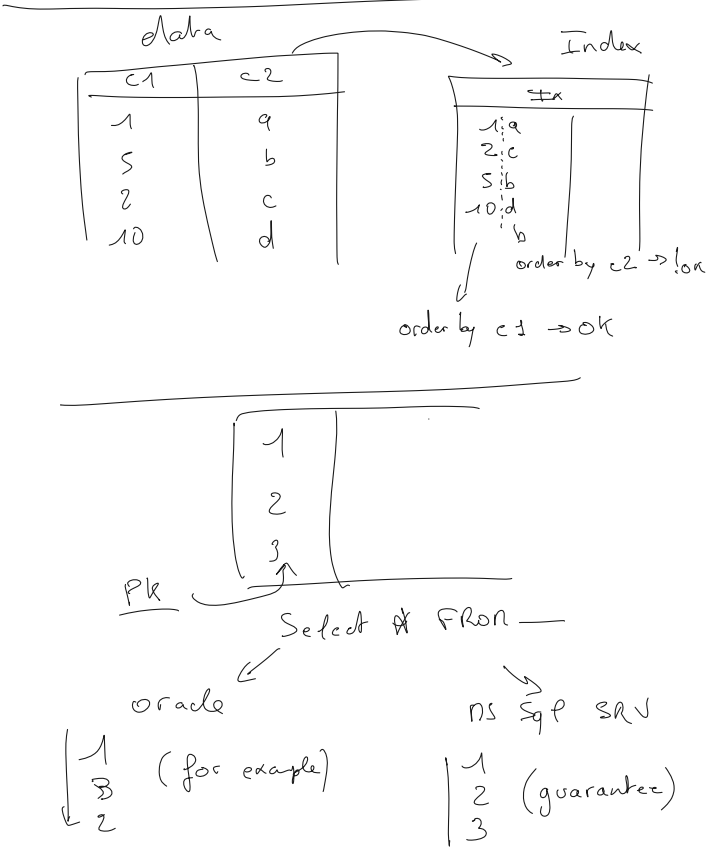
→ Table can have PK or not _

~~Ix~~ → PK Leaf → Ref to PK

Ix → ~~PK~~ Leaf → Ref Page

Exercice : Compare joins between tables with and without indexes (time, and query plan)

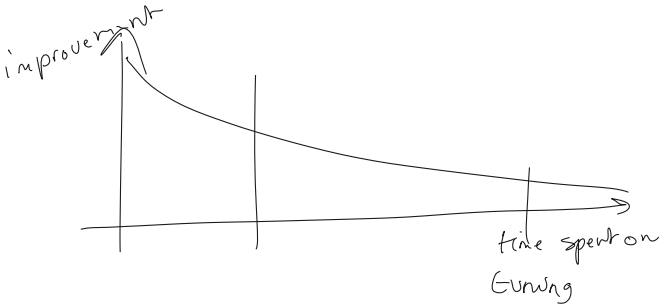
Access paths :
https://docs.oracle.com/database/121/TGSQL/tgsql_optop.htm#TGSQL228

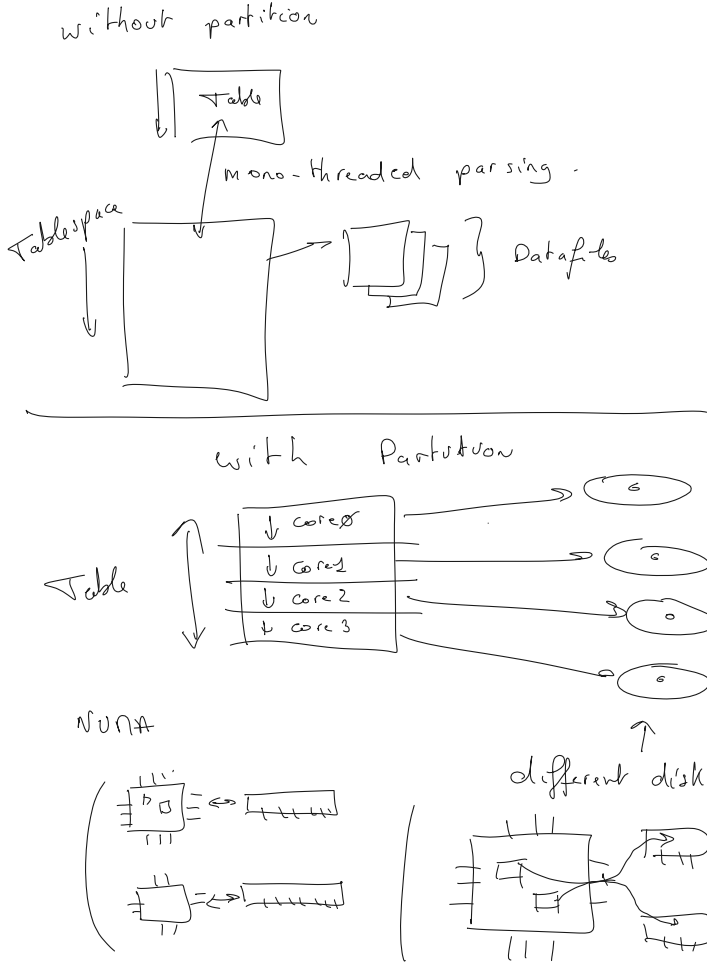


id	firstname	Lastname

Queries:

- Select count(*) from — where fn = —
  ln = —
- " "
- where fn = — AND
  ln = —




without partition
Table
mono-threaded parsing
Tablespace
Datafiles
with Partition
Table
core0
core1
core2
core3
NUMA
different disks

```
CREATE TABLE P_LINEITEM
(
  L_ORDERKEY NUMBER(*, 0) NOT NULL
, L_PARTKEY NUMBER(*, 0) NOT NULL
, L_SUPPKEY NUMBER(*, 0) NOT NULL
, L_LINENUMBER NUMBER(*, 0) NOT NULL
, L_QUANTITY NUMBER(15, 2) NOT NULL
, L_EXTENDEDPRICE NUMBER(15, 2) NOT NULL
, L_DISCOUNT NUMBER(15, 2) NOT NULL
, L_TAX NUMBER(15, 2) NOT NULL
, L_RETURNFLAG CHAR(1 BYTE) NOT NULL
, L_LINESTATUS CHAR(1 BYTE) NOT NULL
, L_SHIPDATE DATE NOT NULL
, L_COMMITDATE DATE NOT NULL
, L_RECEIPTDATE DATE NOT NULL
, L_SHIPINSTRUCT CHAR(25 BYTE) NOT NULL
, L_SHIPMODE CHAR(10 BYTE) NOT NULL
, L_COMMENT VARCHAR2(44 BYTE) NOT NULL
, CONSTRAINT MYPK PRIMARY KEY
  (
    L_ORDERKEY
  , L_PARTKEY
  , L_SUPPKEY
  )
)
PARTITION BY RANGE (L_QUANTITY)
    (PARTITION part1 VALUES LESS THAN (10) TABLESPACE TBS1,
    PARTITION part2 VALUES LESS THAN (15) TABLESPACE TBS2,
    PARTITION part3 VALUES LESS THAN (MAXVALUE) TABLESPACE TBS3)

insert into P_LINEITEM select * from LINEITEM



ALTER TABLE parts MOVE PARTITION depot2
   TABLESPACE ts094 NOLOGGING COMPRESS;

ALTER DATABASE DATAFILE '/u01/app/oracle/oradata/DB11G/reclaim01.dbf' RESIZE 24M;
```

hash (value) → result (numerical)

numerical, string, blob ...

Compute (one way)

hash functions
- md5 → performance — Collision → very few
- sha1, 2(56), 512 → no collision
- CRC
- checksum →

function → bring collision

collision 2 ≠ values → same result

ex: CRC

A = 65, B = 66

AB = 131 / BA = 131

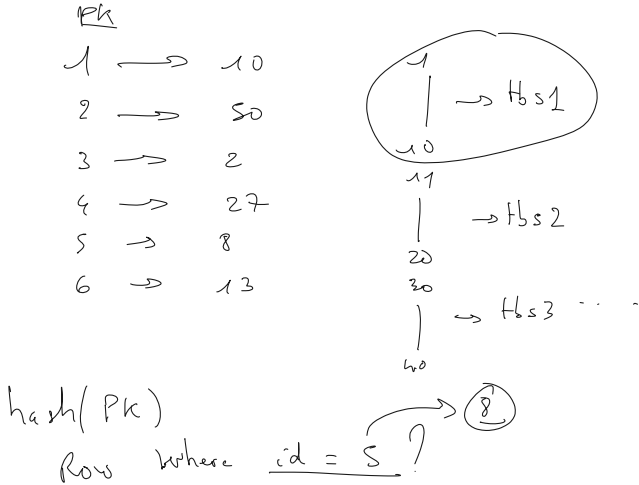
intel Xeon → sha2, md5 encryption } hardware accel.

80286

287 → math acceleration

md5 (salt)

hash → value




```
CREATE TABLE table
(col1 type_donnée1,
col2 type_donnée2,
...,
coln type_donnéen)
PARTITION BY HASH (col)
PARTITIONS n
[STORE IN (tbs_part1, ..., tbs_partn)];
```

Enable // :
ALTER SESSION [ENABLE | DISABLE | FORCE] PARALLEL {DML | DDL | n};

SELECT username, pdml_status, pddl_status, pq_status, pdml_enabled
FROM v$session
WHERE username IS NOT NULL;
USERNAME PDML_STA PDDL_STA PQ_STATU PDM
------------------------------ -------- -------- -------- ---
SYSTEM ENABLED ENABLED ENABLED YES

Hint (to force) :
SELECT /*+PARALLEL(table,n)*/ col1, col2, ...
FROM table
WHERE . . .;
UPDATE /*+PARALLEL(table,n)*/ table
SET col1=val1, col2=val2, ...
WHERE . . .;
DELETE /*+PARALLEL(table,n)*/
FROM table
WHERE . . .;

Check :
SELECT *
FROM v$pq_sesstat
WHERE statistic LIKE '%Parallel%';
STATISTIC LAST_QUERY SESSION_TOTAL
------------------------------ ---------- -------------
Queries Parallelized 0 0
DML Parallelized 0 0
DDL Parallelized 1 1

1) enable // to your session
2) execute Select command
3) check your query plan and the statistics
4) Compare using hints to force more //

```
INSERT /*+PARALLEL (emp_part_plages_parallel,3)*/
INTO emp_part_plages_parallel
SELECT /*+PARALLEL (e_client,3) */ no, nom, SYSDATE
FROM e_client;
15 ligne(s) créée(s).
SELECT *
FROM v$pq_sesstat
WHERE statistic LIKE '%Parallel%';
STATISTIC LAST_QUERY SESSION_TOTAL
------------------------------ ---------- -------------
Queries Parallelized 0 0
DML Parallelized 1 1
DDL Parallelized 0 1
```

Statistics

```
ANALYZE {TABLE table_name | INDEX index_name}
{COMPUTE|ESTIMATE|DELETE} STATISTICS [SAMPLE n {ROWS|PERCENT}];
```

Compute=whole values

DBA_TABLES et DBA_INDEXES

```
ANALYZE {TABLE table_name | INDEX index_name}
{COMPUTE|ESTIMATE|DELETE} STATISTICS [SAMPLE n {ROWS | PERCENT}]
FOR {COLUMNS col SIZE h1 [col SIZE h2]| ALL INDEXED COLUMNS SIZE h3}
```

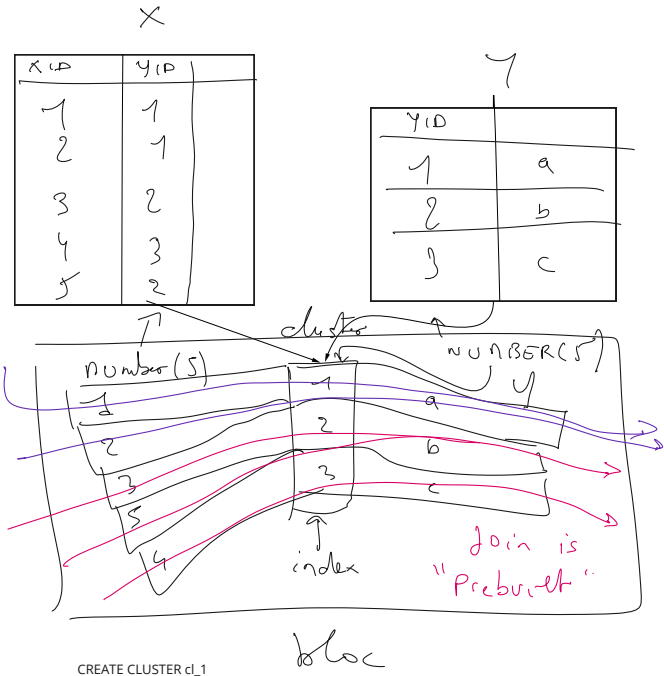
Update statistics of LINEITEM
Update statistics of an INDEX
Check them

Compare execution plan with and without (deleted) statistics on a table and indexes
check the table stats before and after a query.

Use plan Baseline to setup Query plans :

https://docs.oracle.com/database/121/TGSQL/tgsql_spm.htm#TGSQL94621

```
DECLARE
tune_sql CLOB;
tune_task VARCHAR2(30);
BEGIN
tune_sql := 'select l_orderkey from dbt3.p_lineitem';
tune_task := DBMS_SQLTUNE.CREATE_TUNING_TASK(
sql_text
=> tune_sql
,user_name => 'DBT3'
,scope
=> 'COMPREHENSIVE'
,time_limit => 60
,task_name => 'TUNE1'
,description => 'Calling SQL Tuning Advisor for one statement'
);
END;

-- 2 :
exec dbms_sqltune.execute_tuning_task(task_name=>'TUNE1');

-- 3 :
set long 10000
set longchunksize 10000
set lines 132
set pages 200
select dbms_sqltune.report_tuning_task('TUNE1') from dual;

-- 4 : delete the plan
-- a : choose :
select name, type, status, sql_text from dba_sql_profiles;
-- b : delete :
exec dbms_sqltune.drop_sql_profile('SYS_SQLPROF_016d72b5d1470000');

-- 4' : drop a task :
begin
   DBMS_SQLTUNE.DROP_TUNING_TASK( 'TUNE1' );
end;

-- 3' : do what the optimiser says :
begin
   dbms_stats.gather_table_stats(ownname => 'DBT3', tabname =>
        'P_LINEITEM', estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE,
        method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade => TRUE);
end;

-- 5  : plan baseline exist ?
set pages 100 linesize 150
col sql_handle form a20
col plan_name form a30
col sql_text form a40
col created form a20
--
SELECT sql_handle, plan_name, enabled
,accepted, created, optimizer_cost, sql_text
FROM dba_sql_plan_baselines;

-- 6 : create a baseline :
-- a : execute the query :
select l_orderkey from dbt3.p_lineitem;
-- b : find the id :
select sql_id, plan_hash_value, sql_text
from v$sql
where sql_text like 'select l_orderkey from dbt3.p_lineitem'
and sql_text not like '%v$sql%';
-- c: load the plan :
DECLARE
plan1 PLS_INTEGER;
BEGIN
   plan1 := DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE(sql_id => '0j33fc04s4gn7');
END;
-- d: display baselines :
select sql_handle, plan_name, sql_text from dba_sql_plan_baselines;
-- e: check if query uses the plan baseline : ( no rows mean no execution with the plan ) :
select sql_id, child_number, sql_plan_baseline, sql_text from v$sql
   where sql_plan_baseline is not null
   and sql_text like 'select l_orderkey%';
```




```
CREATE CLUSTER cl_1
(dept_no number(3));

-- Création d'un index (Obligatoire)
CREATE INDEX cl_1_ind ON CLUSTER cl_1;

-- Création des tables en cluster
CREATE TABLE salarie
(no NUMBER(4) NOT NULL,
nom VARCHAR2(10) NOT NULL,
emploi VARCHAR2(12),
dept_no NUMBER(3))
CLUSTER cl_1 (dept_no);
CREATE TABLE departement
(no NUMBER(3) NOT NULL,
nom VARCHAR2(14) NOT NULL,
localite VARCHAR2(13))
CLUSTER cl_1 (no);
```