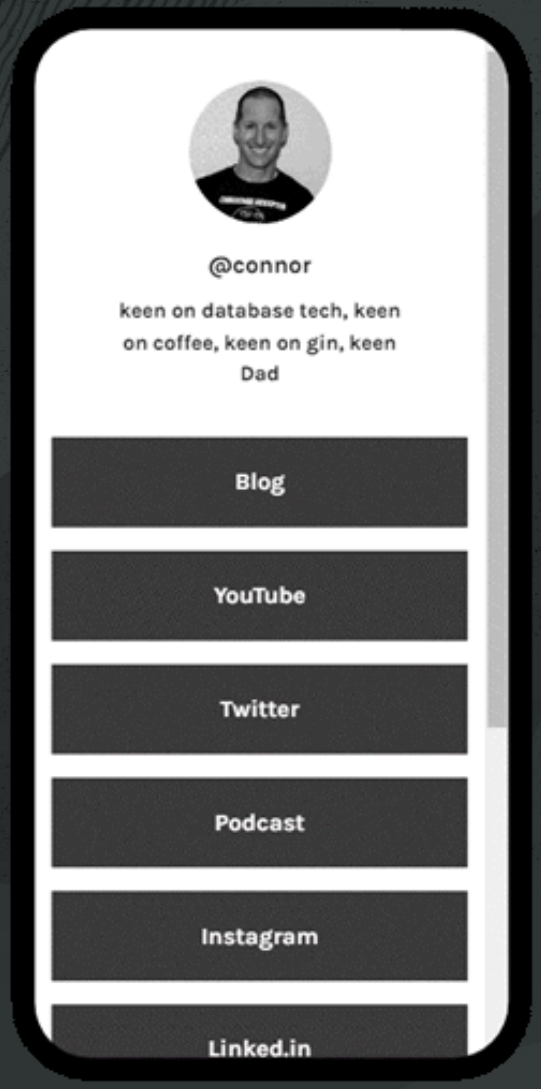
Getting in touch is easy...



@connor_mc_d

https://linktr.ee/connor

before we begin






EXTRA FLAVOUR
B21 - B32
B17, B19
B18, B20
دورات المياه
Toilets
PANDORA

thank you for your support of user groups



5 cool things you might not be using

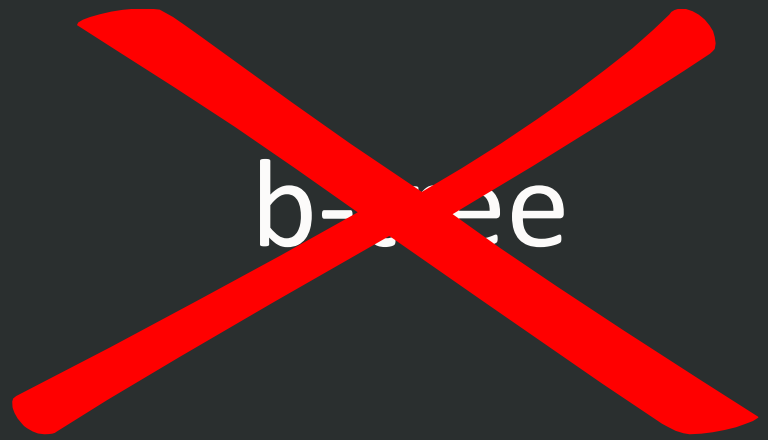


1

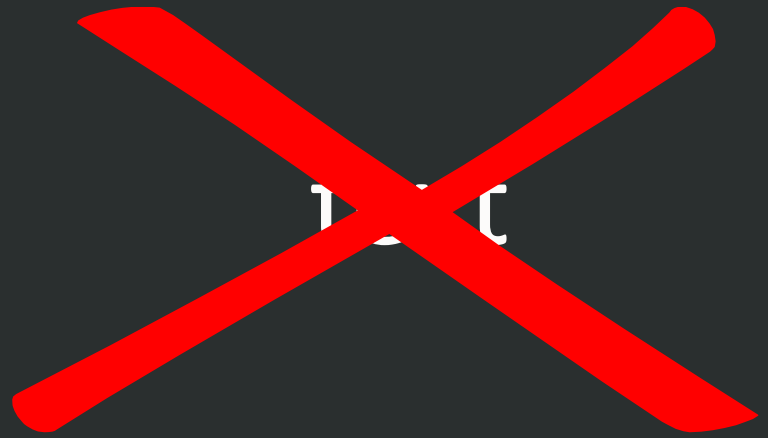
indexing




DUDE
I AM USING INDEXES !


b-tree


bi ap




in-memory


reverse

still lots to talk about



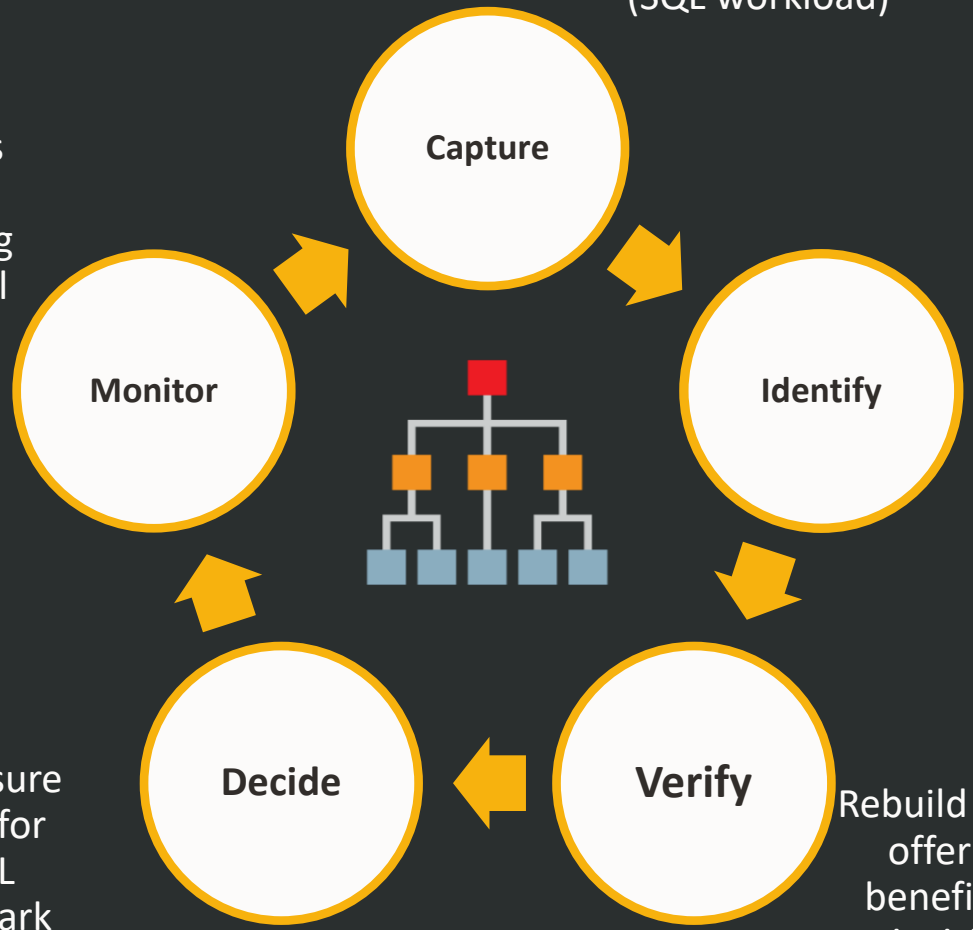
19c



automatic indexing



Identify index candidates based on column usage (SQL workload)



Monitor all indexes (automatic and manual) for ongoing usage and potential removal

Create indexes unusable, invisible and evaluate new execution plans *without parse costs*

Rebuild indexes that offer potential benefit (status as invisible). Test execute each SQL statement

Use SPM to ensure no regression for excluded SQL statements. Mark indexes as visible

only Exadata / Autonomous



"I need a new index
on this 6TB table..."



"will it speed up things ?"



"I think so"



indexes = high risk to build



virtual indexes



```
SQL> create index TEST on
  2  MYTAB ( SALES_DATE ) NOSEGMENT;

Index created.
```



data dictionary only

```
SQL> begin
  2     dbms_stats.generate_stats(
  3           'SCOTT',
  4           'MYTAB');
  5  end;
  6  /
```

```
SQL> alter session
  2    set "_use_nosegment_indexes" = true;

SQL> explain plan for
  2  select * from MYTAB
  3  where sales_date > sysdate-7;
```



```
SQL> alter session
  2    set "_use_nosegment_indexes" = true;

SQL> explain plan for
  2  select * from MYTAB
  3  where sales_date > sysdate-7;

-----------------------------------------------------------------------------
| Id  | Operation                    | Name  | Rows  | Bytes | Cost (%CPU)|
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |       |   950 | 90109 |   962   (34)|
|   1 |  TABLE ACCESS BY INDEX ROWID| MYTAB |   950 | 90109 |   962   (34)|
|*  2 |   INDEX RANGE SCAN           | TEST  |   950 |       |   962   (50)|
-----------------------------------------------------------------------------
```

indexes = high risk to use



invisible indexes



```
SQL> create table T
  2  as select * from all_objects;

Table created.

SQL> create index OBJ_IX on T ( OBJECT_ID );

Index created.

SQL> exec dbms_stats.gather_table_stats('','T')

PL/SQL procedure successfully completed.
```



```
SQL> set autotrace traceonly explain

SQL> select * from T
  2  where  OWNER = 'SCOTT'
  3  and    created > sysdate - 1
  4  /

----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)|
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     6 |   546 |   281   (1)|
|*  1 |  TABLE ACCESS FULL| T    |     6 |   546 |   281   (1)|
----------------------------------------------------------------------------
```

"Can I index OWNER.."



"will it speed up things ?"



"I know so"



```
SQL> create index NEW_IX on T ( OWNER );

Index created.

SQL> select * from T
  2  where  OWNER = 'SCOTT'
  3  and    created > sysdate - 1
  4  /

-----------------------------------------------------------------------------
| Id  | Operation                    | Name   | Rows  | Bytes | Cost (%CPU)|
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |        |     6 |   546 |   122   (0)|
|*  1 |  TABLE ACCESS BY INDEX ROWID| T      |     6 |   546 |   122   (0)|
|*  2 |   INDEX RANGE SCAN           | NEW_IX |  4107 |       |    10   (0)|
-----------------------------------------------------------------------------
```





all the other queries





before the new index



```
SQL> set autotrace traceonly explain

SQL> select * from T
  2  where  OWNER = 'SYS'
  3  and    OBJECT_ID between 10 and 8000
  4  /

----------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost (%CPU)|
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |   462 | 42042 |   143   (0)|
|*  1 |  TABLE ACCESS BY INDEX ROWID| T       |   462 | 42042 |   143   (0)|
|*  2 |   INDEX RANGE SCAN           | OBJ_IX  |  7851 |       |    20   (0)|
----------------------------------------------------------------------------
```



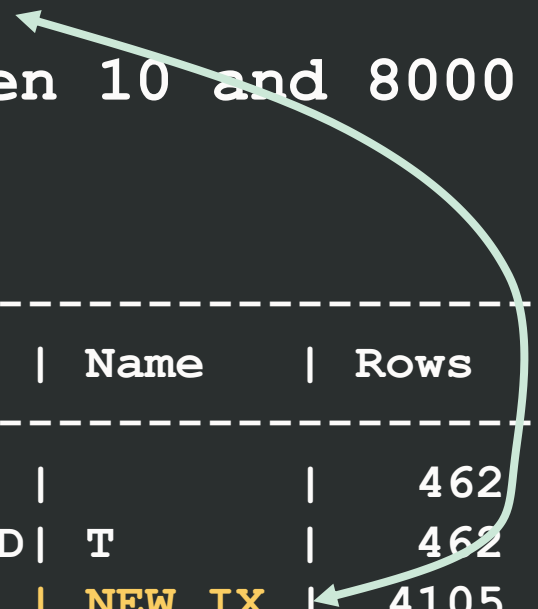
```
SQL> set autotrace traceonly stat
SQL> select * from T
  2  where  owner = 'SYS'
  3  and    object_id between 10 and 8000;

4967 rows selected.

Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
        784  consistent gets
         15  physical reads
          0  redo size
```



after the new index



```
SQL> select * from T
  2  where  owner = 'SYS'
  3  and    object_id between 10 and 8000
  4  /

-------------------------------------------------------------------------------
| Id  | Operation                    | Name   | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |        |   462 | 42042 |   122   (0)|
|*  1 |  TABLE ACCESS BY INDEX ROWID| T      |   462 | 42042 |   122   (0)|
|*  2 |   INDEX RANGE SCAN           | NEW_IX |  4105 |       |    10   (0)|
-------------------------------------------------------------------------------
```



```
SQL> set autotrace traceonly stat
SQL> select * from T
  2  where  owner = 'SYS'
  3  and    object_id between 10 and 8000;

4967 rows selected.

Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
       1522  consistent gets
         62  physical reads
          0  redo size
```





the solution ?



invisible indexes



```
SQL> alter index NEW_IX invisible;

Index altered.
```



```
SQL> select * from T
  2  where  owner = 'SYS'
  3  and    object_id between 10 and 8000
  4  /

-------------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |   462 | 42042 |   143   (0)|
|*  1 |  TABLE ACCESS BY INDEX ROWID| T       |   462 | 42042 |   143   (0)|
|*  2 |   INDEX RANGE SCAN           | OBJ_IX  |  7851 |       |    20   (0)|
-------------------------------------------------------------------------------
```



"what about my code ???"



```
SQL> set autotrace traceonly explain

SQL> select * from T
  2  where  OWNER = 'SCOTT'
  3  and    created > sysdate - 1
  4  /

----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)|
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     6 |   546 |   281   (1)|
|*  1 |  TABLE ACCESS FULL| T    |     6 |   546 |   281   (1)|
----------------------------------------------------------------------------
```

```
SQL> alter session set
  2       optimizer_use_invisible_indexes = true;

Session altered.

SQL> select * from T
  2  where  OWNER = 'SCOTT'
  3  and    created > sysdate - 1
  4  /

-------------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |   462 | 42042 |   122   (0)|
|*  1 |  TABLE ACCESS BY INDEX ROWID| T       |   462 | 42042 |   122   (0)|
|*  2 |   INDEX RANGE SCAN           | NEW_IX  |  4105 |       |    10   (0)|
-------------------------------------------------------------------------------
```



wrap your SQL to protect others



statement level hint

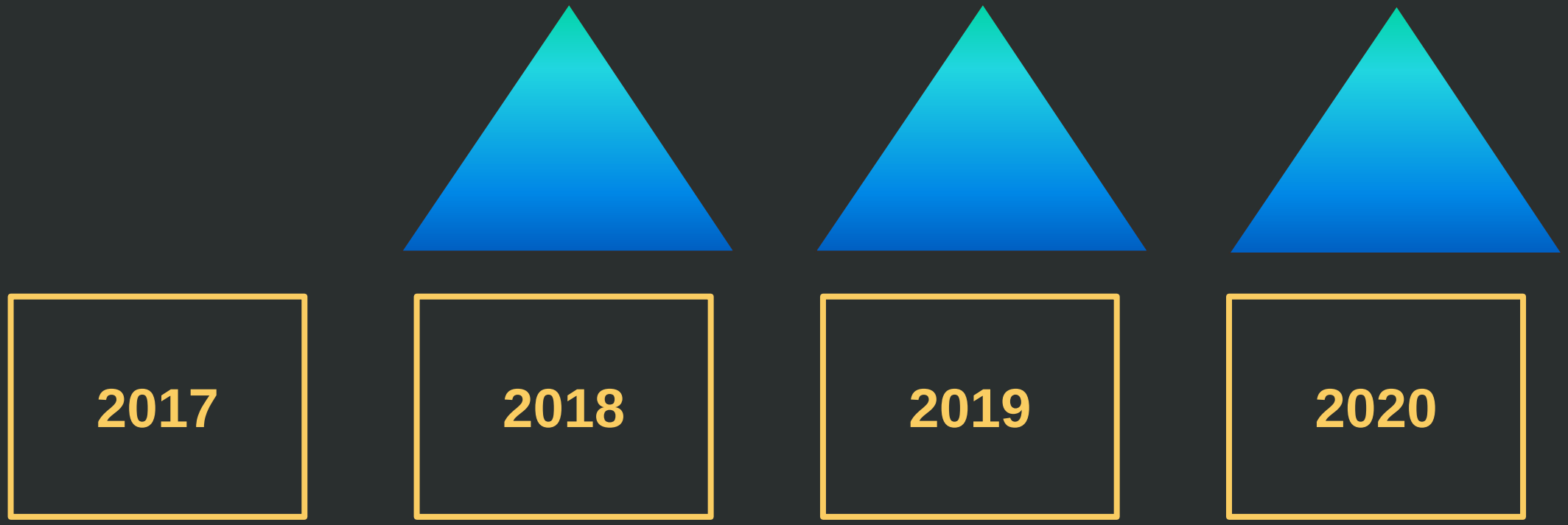
USE_INVISIBLE_INDEXES



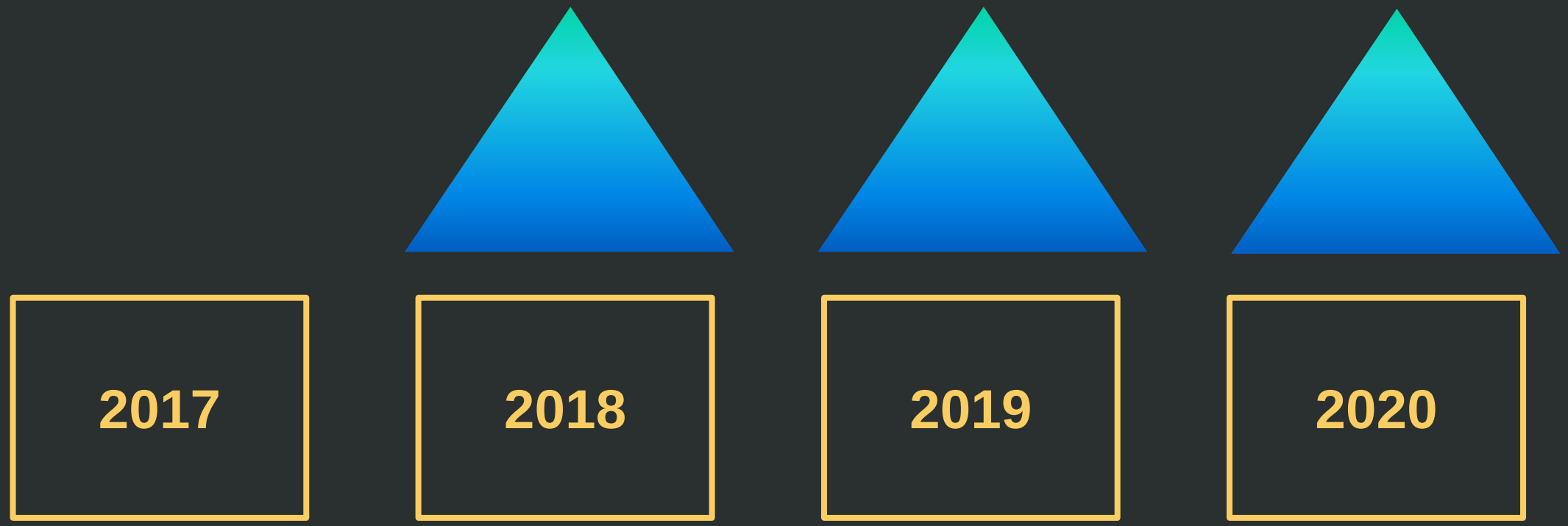
partial indexing



```
SQL> create table SALES
  2  ( trans_id        varchar2(10) not null,
  3    ...
  4  )
  5  INDEXING OFF
  6  PARTITION BY LIST (fin_year)
  7  (
  8   PARTITION P_2015  VALUES (2015) INDEXING OFF,
  9   PARTITION P_2016  VALUES (2016) INDEXING ON,
 10   PARTITION P_2017  VALUES (2017) INDEXING ON,
 11   PARTITION P_2018  VALUES (2018) INDEXING ON,
 12  )
 13  /

Table created.
```



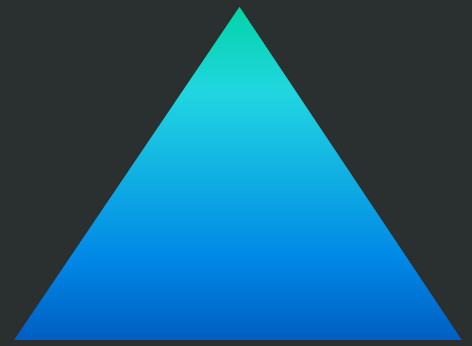

```
SQL> create index SALES_IX on SALES ( ... ) LOCAL
  2     INDEXING PARTIAL;
```




```
SQL> alter table SALES modify partition SALES_2017
  2    INDEXING OFF;
```

why ?





2017

2018

2019

2020

trends

txns

or the opposite






```
SQL> alter table SALES modify partition SALES_2020
  2     INDEXING OFF;
```

especially with 19c



partial indexing

non-partitioned



function based indexes



(entire) nulls not indexed



```
SQL> select ship_status, count(*)
  2  from ORDERS
  3  group by ship_status;

SHIP_STATUS COUNT(*)
----------- --------
SHIPPED        80000
READY             80

SQL> create index IX
  2  on ORDERS ( ship_status );

Index created.

SQL> exec  dbms_stats.gather_table_stats(user,'ORDERS');

PL/SQL procedure successfully completed.
```



```
SQL> set autotrace traceonly explain

SQL> select * from ORDERS
  2  where ship_status = 'READY';

-----------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)|
-----------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         | 41009 |  1503K|     74   (2)|
|*  1 |  TABLE ACCESS FULL| ORDERS  | 41009 |  1503K|     74   (2)|
-----------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("ship_status"='READY')
```



50% = SHIPPED
50% = READY

"no problem"



```
SQL> begin
  2     dbms_stats.gather_table_stats(user,'ORDERS',
  3        method_opt=>'for all indexed columns size auto');
  4  end;
  5  /

PL/SQL procedure successfully completed.
```



```
SQL> select * from T
  2  where ship_status = 'READY';

-------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Cost (%CPU)|
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |    82 |     3  (34)|
|   1 |  TABLE ACCESS BY INDEX ROWID| ORDERS  |    82 |     3  (34)|
|*  2 |   INDEX RANGE SCAN           | IX      |    82 |     2  (50)|
-------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("ship_status"='READY')
```



```
SQL> select leaf_blocks
  2  from   user_indexes
  3  where  index_name = 'IX';

LEAF_BLOCKS
-----------
        139
```



still indexing the "SHIPPED" rows



```
SQL> drop index IX;

Index dropped.

SQL> create index IX on ORDERS
  2  ( case when ship_status = 'READY' then 'Y' end );

Index created.

SQL> exec  dbms_stats.gather_table_stats(user,'ORDERS');

PL/SQL procedure successfully completed.
```



```
SQL> select * from ORDERS
  2  where case when ship_status = 'READY' then 'Y' end = 'Y';

-----------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost (%CPU)|
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |    54 |  3132 |     3  (34)|
|   1 |  TABLE ACCESS BY INDEX ROWID| ORDERS  |    54 |  3132 |     3  (34)|
|*  2 |   INDEX RANGE SCAN           | IX      |    54 |       |     2  (50)|
-----------------------------------------------------------------------------

SQL> select leaf_blocks
  2  from user_indexes
  3  where index_name = 'IX';

LEAF_BLOCKS
-----------
          1
```



"what about my app code?"



virtual columns



```
SQL> alter table ORDERS as
  2  add my_virtual_col varchar2(1) generated always as
  3    ( case when ship_status = 'READY' then 'Y' end );

Table altered.

SQL> select *
  2  from ORDERS
  3  where my_virtual_col = 'Y';
```



optional: make the column invisible



2

clustering



faster queries ...



faster queries ... no code changes



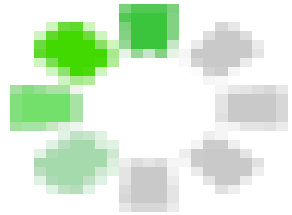
```
SQL> create table cust_trans
  2    ( txn_id     int,
  3      txn_date   date,
  4      cust_id    int,
  5      amt        number(10,2)
  6    );
```

```
SQL> insert /*+ APPEND */ into cust_trans
  2  select rownum,
  3         sysdate-1000+rownum/1000  txn_date,
  4         trunc(dbms_random.value(1,1000)) cust_id,
  5         dbms_random.value(1,100) amount
  6  from
  7  ( select 1 from dual
  8    connect by level <= 10000000 );

10000000 rows created.
```

```
SQL> insert /*+ APPEND */ into cust_trans
  2  select rownum,
  3         sysdate-1000+rownum/1000  txn_date,
  4         trunc(dbms_random.value(1,1000)) cust_id,
  5         dbms_random.value(1,100) amount
  6  from
  7  ( select 1 from dual
  8    connect by level <= 10000000 );

10000000 rows created.

SQL> create index cust_trans_ix
  2    on cust_trans (cust_id );

Index created.
```

```
SQL> select max(amt)
  2  from   cust_trans
  3  where  cust_id = 123;
```



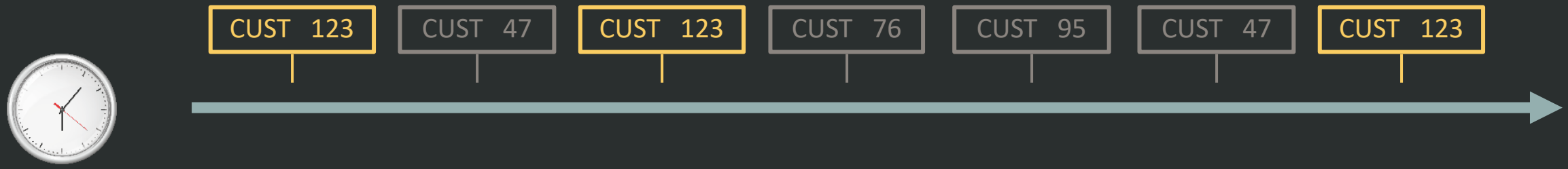


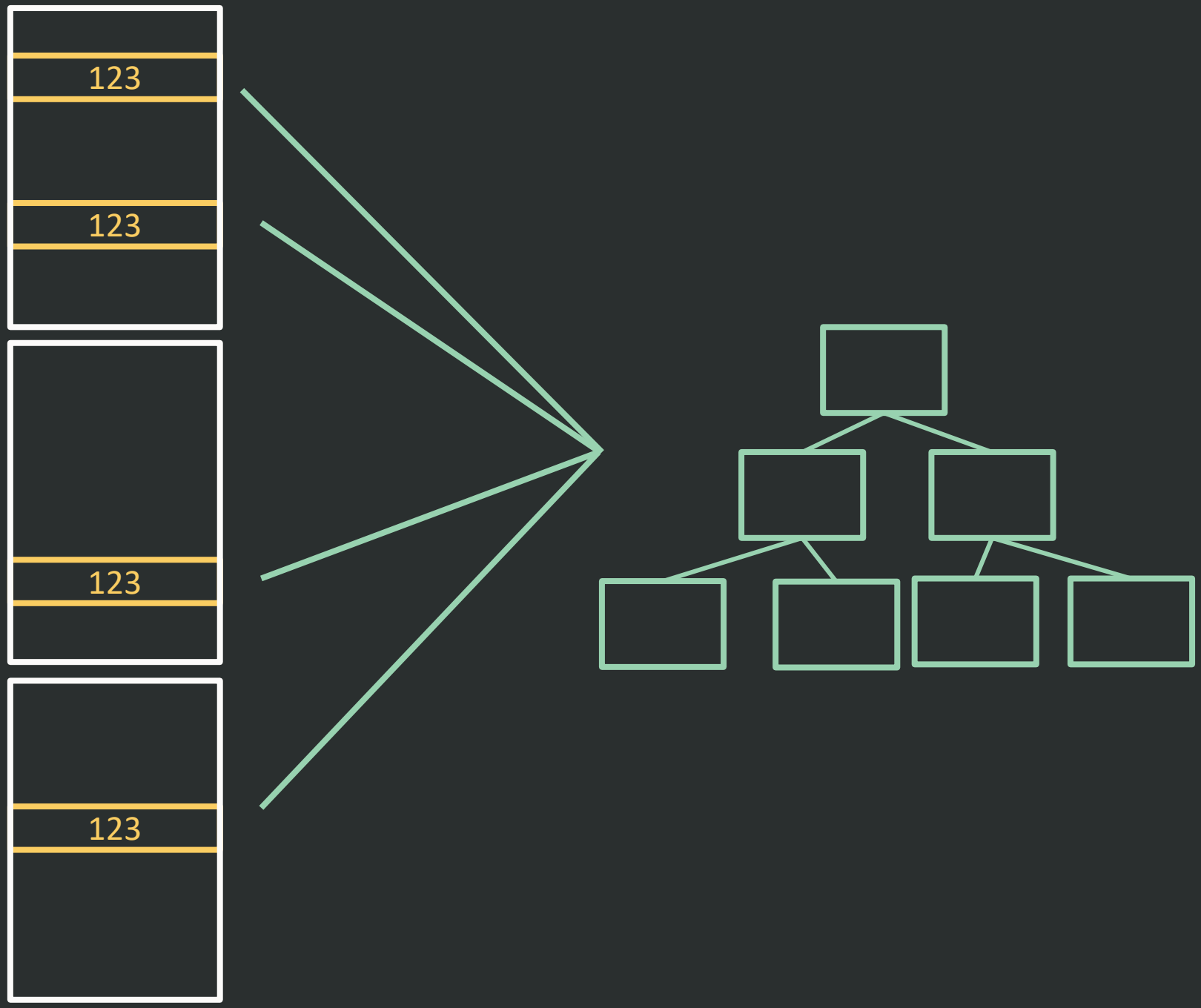
```
SQL> select max(amt)
  2  from   cust_trans
  3  where  cust_id = 123;

-------------------------------------------------------------------------------
| Id  | Operation                                | Name           | Rows  | Bytes |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                |     1 |     8 |
|   1 |  SORT AGGREGATE                          |                |     1 |     8 |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| CUST_TRANS     | 10010 | 80080 |
|*  3 |    INDEX RANGE SCAN                      | CUST_TRANS_IX  | 10010 |       |
-------------------------------------------------------------------------------
```



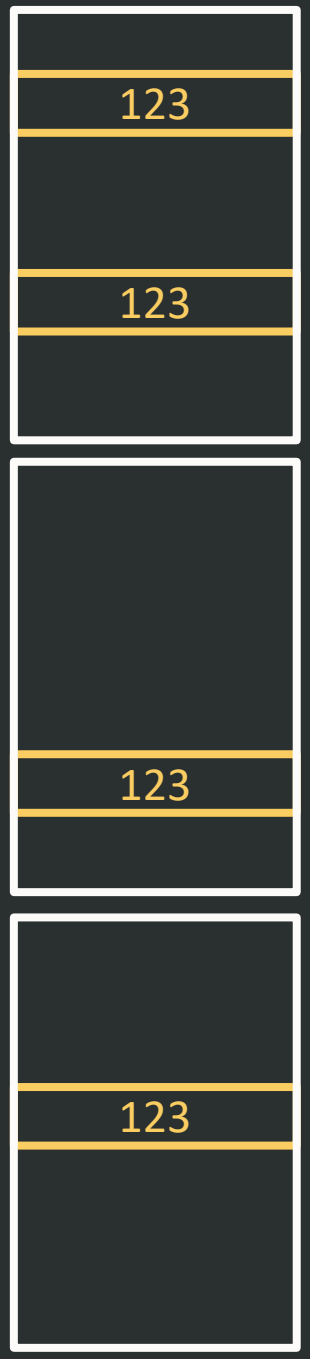
```
SQL> select max(amt)
  2  from   cust_trans
  3  where  cust_id = 123;

-----------------------------------------------------------------------------
| Id  | Operation                           | Name           | Rows  | Bytes |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                |     1 |     8 |
|   1 |  SORT AGGREGATE                     |                |     1 |     8 |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| CUST_TRANS     | 10010 | 80080 |
|*  3 |    INDEX RANGE SCAN                 | CUST_TRANS_IX  | 10010 |       |
-----------------------------------------------------------------------------

Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
       8867  consistent gets
       8854  physical reads
          0  redo size
        543  bytes sent via SQL*Net to client
        607  bytes received via SQL*Net from client
```

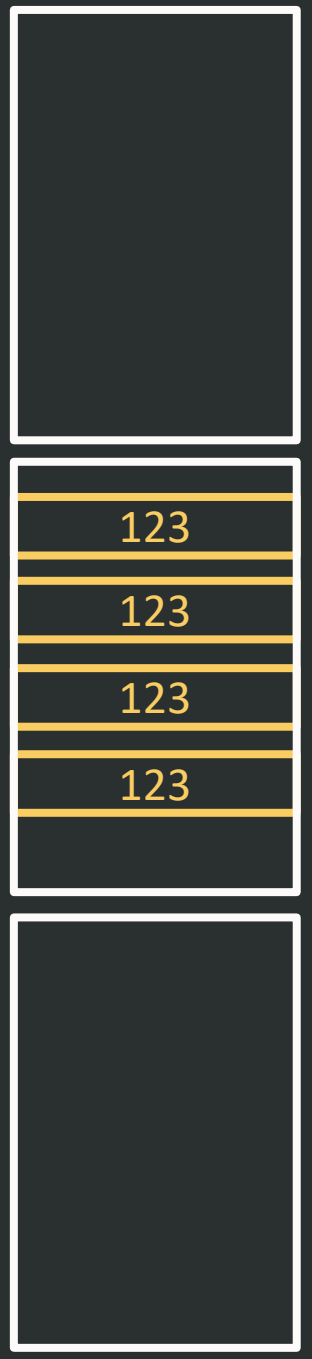


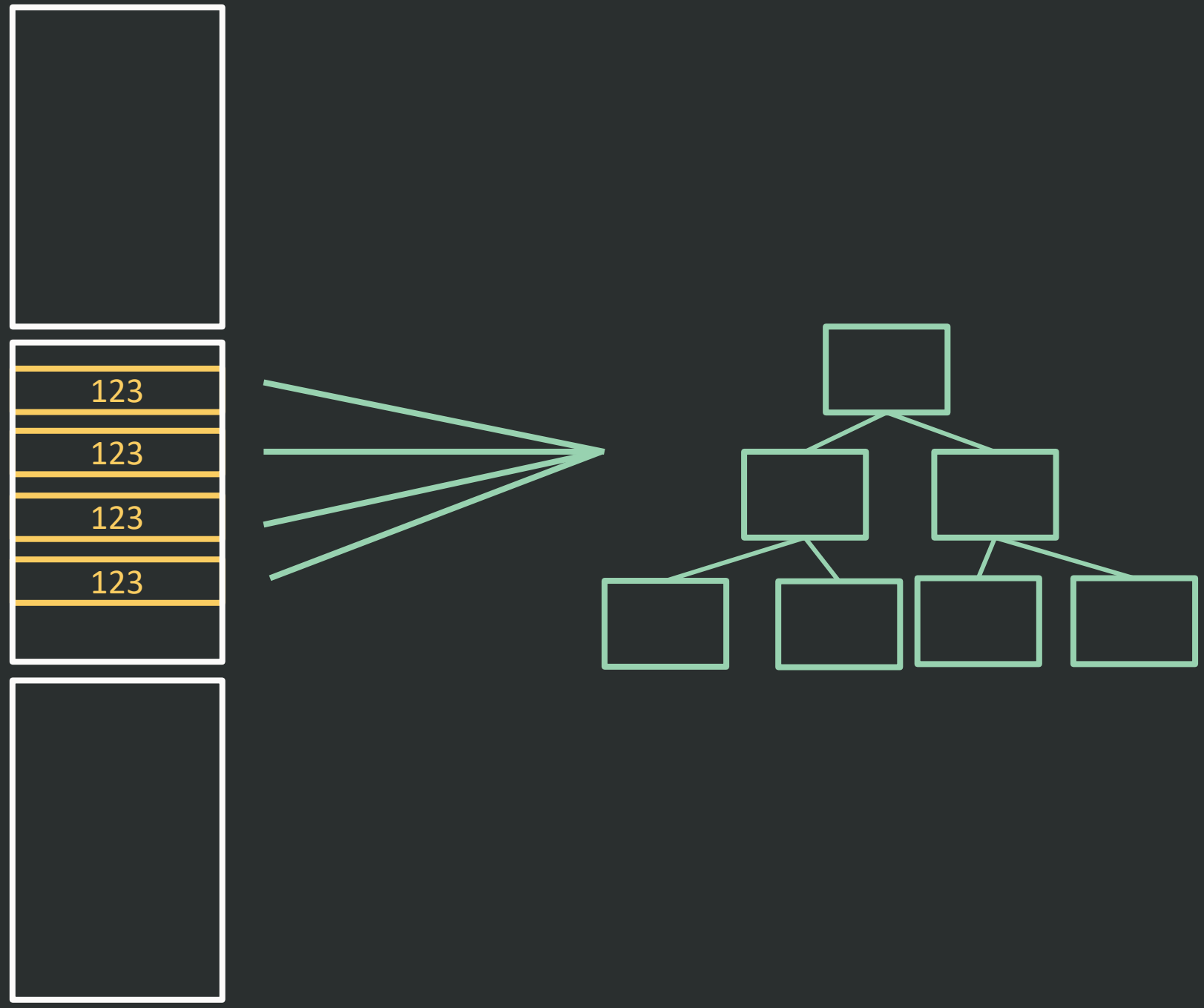

CUST 123
CUST 47
CUST 123
CUST 76
CUST 95
CUST 47
CUST 123


123
123
123
123

```
SQL> alter table cust_trans
  2     add clustering by linear order(cust_id);

Table altered.


SQL> alter table cust_trans move online;

Table altered.
```


123
123
123
123


123
123
123
123


123
123
123
123

```
SQL> select max(amt)
  2  from   cust_trans
  3  where  cust_id = 123;

-------------------------------------------------------------------------------
| Id  | Operation                            | Name           | Rows  | Bytes |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |                |     1 |     8 |
|   1 |  SORT AGGREGATE                      |                |     1 |     8 |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| CUST_TRANS     | 10010 | 80080 |
|*  3 |    INDEX RANGE SCAN                  | CUST_TRANS_IX  | 10010 |       |
-------------------------------------------------------------------------------
```



```
SQL> select max(amt)
  2  from   cust_trans
  3  where  cust_id = 123;

-------------------------------------------------------------------------------
| Id  | Operation                            | Name           | Rows  | Bytes |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |                |     1 |     8 |
|   1 |  SORT AGGREGATE                      |                |     1 |     8 |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| CUST_TRANS     | 10010 | 80080 |
|*  3 |    INDEX RANGE SCAN                  | CUST_TRANS_IX  | 10010 |       |
-------------------------------------------------------------------------------

Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
         62  consistent gets
         23  physical reads
          0  redo size
        543  bytes sent via SQL*Net to client
        607  bytes received via SQL*Net from client
```



de-clustering



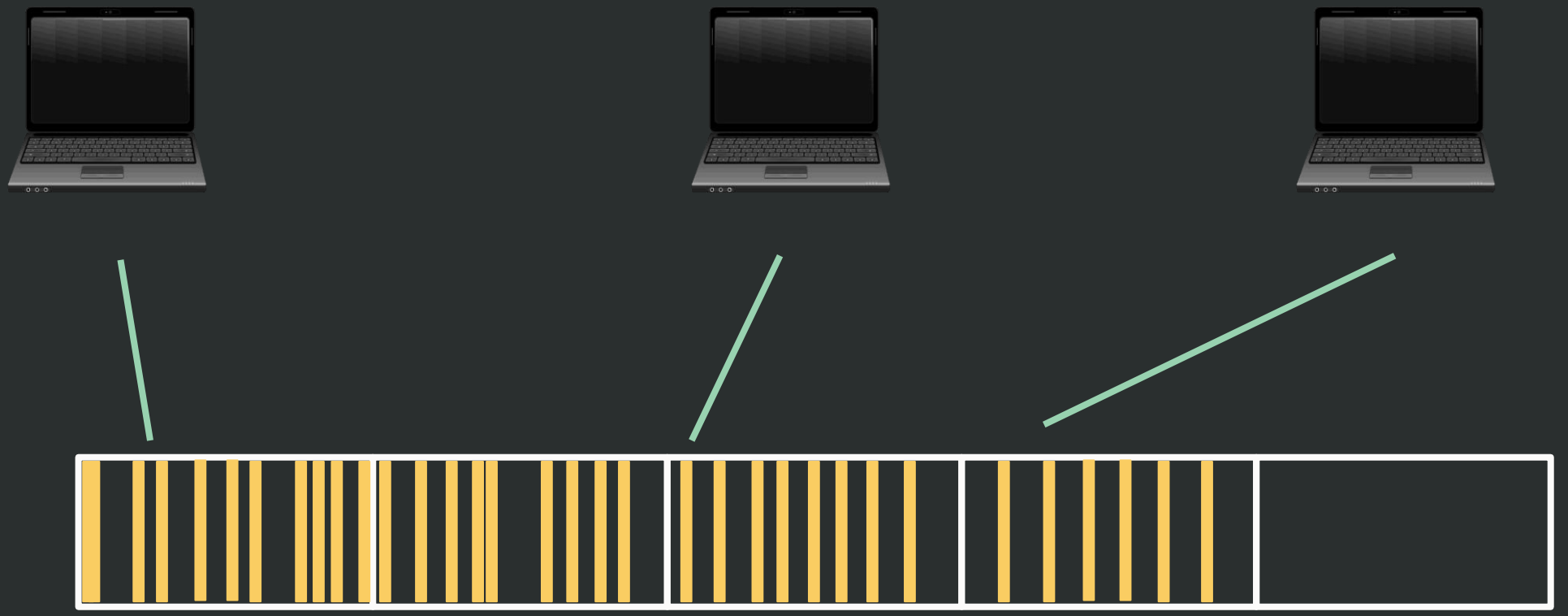


table contention is a solved problem



ASSM



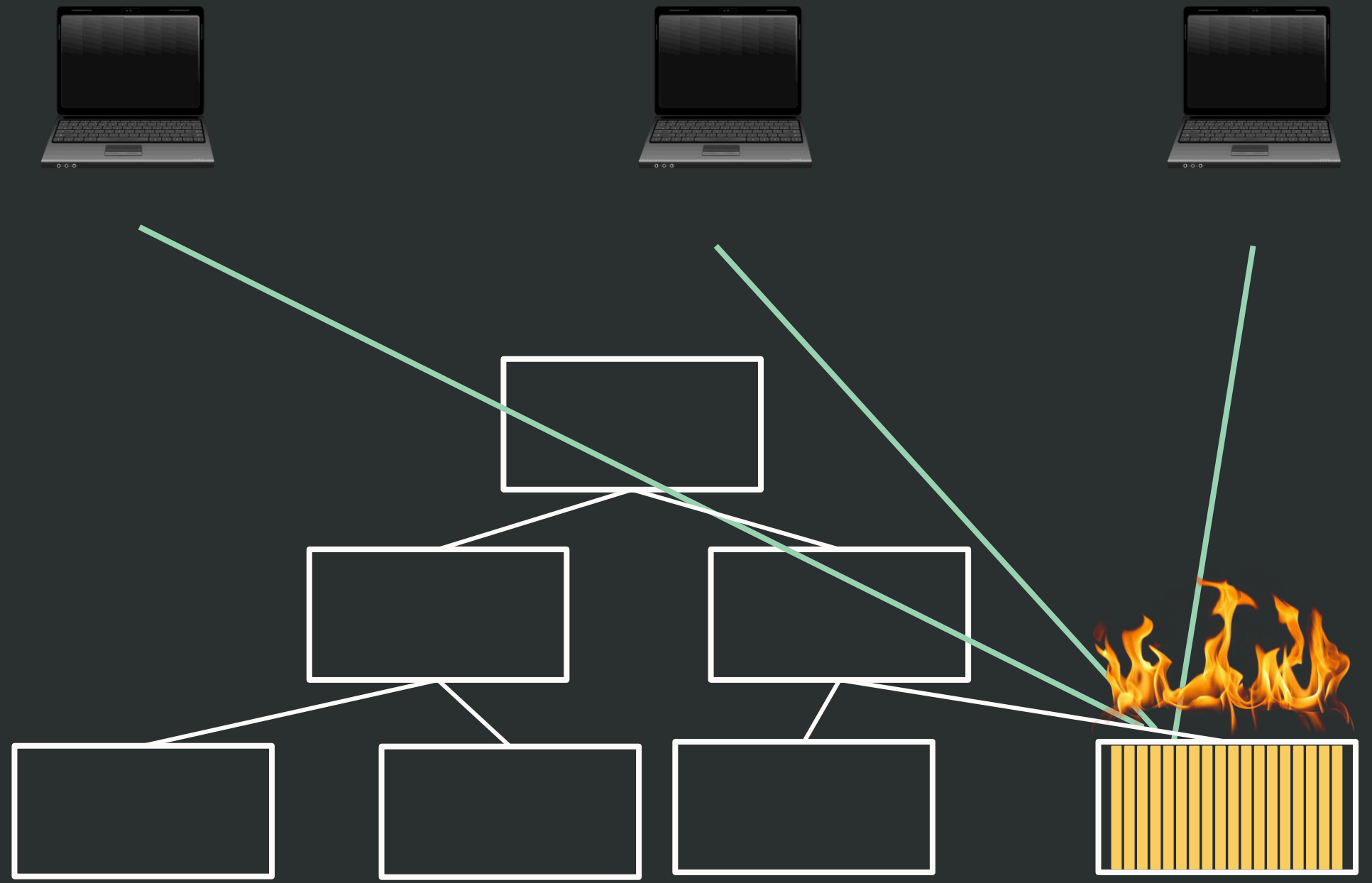


see my podcast

indexes a problem

ascending key

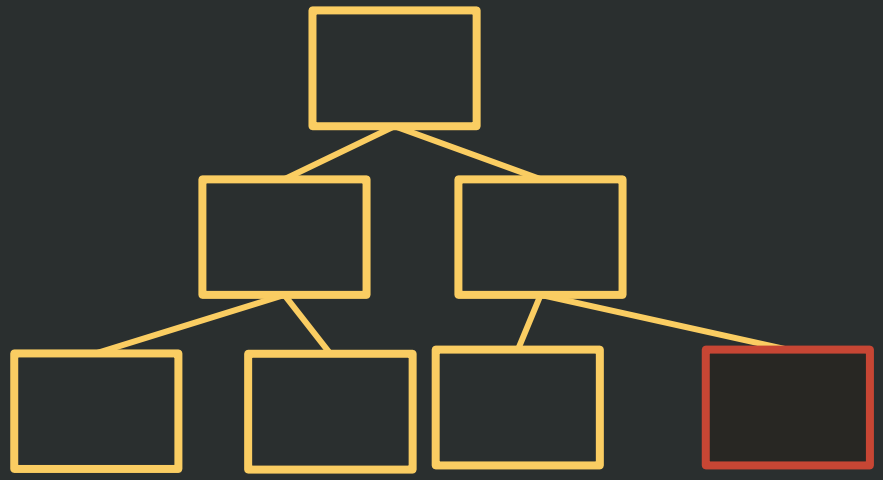


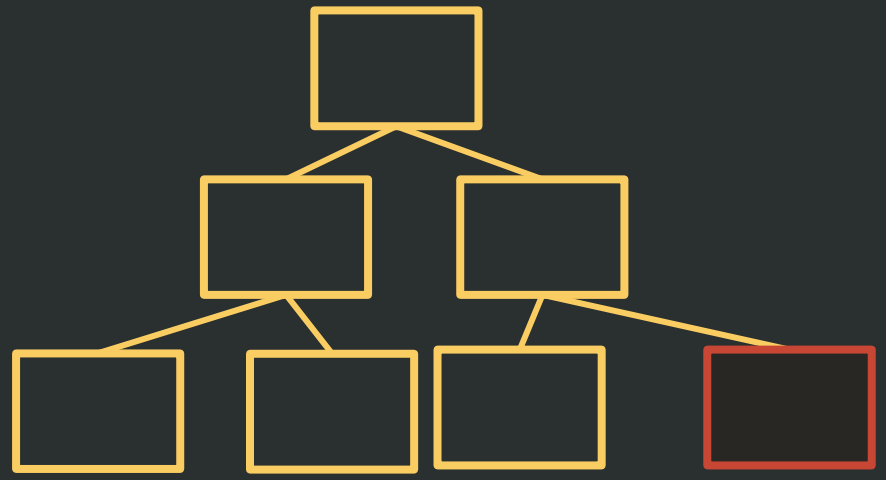


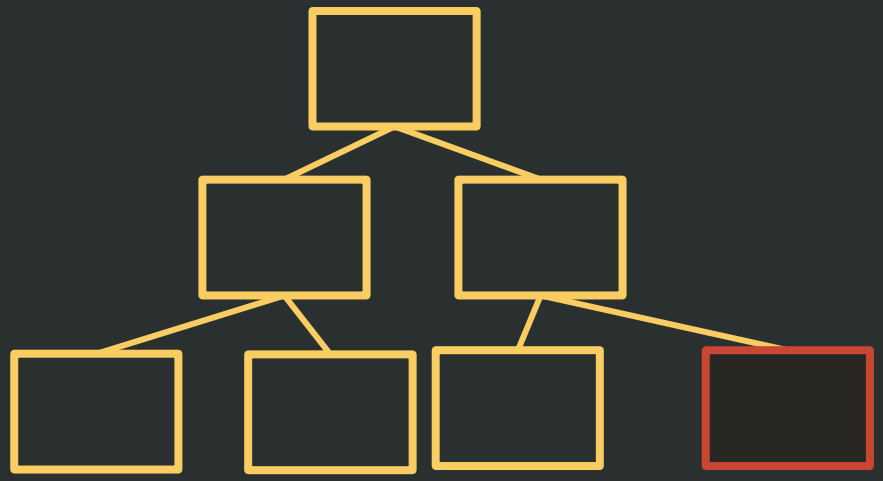
hash partitioned indexes

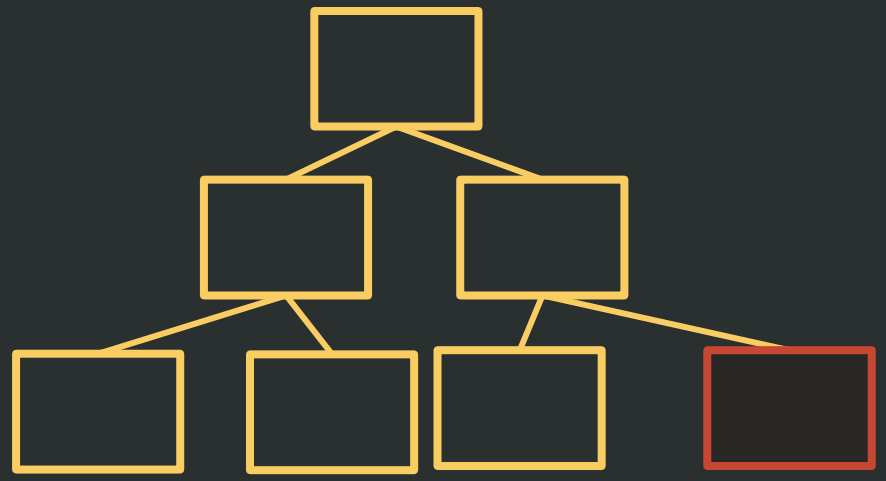


```
SQL> create index SALES_PK on SALES( TXN_ID )
  2  global partition by hash ( TXN_ID )
  3  partitions 8
  4  /

Index created.
```









3

online operations



STOP

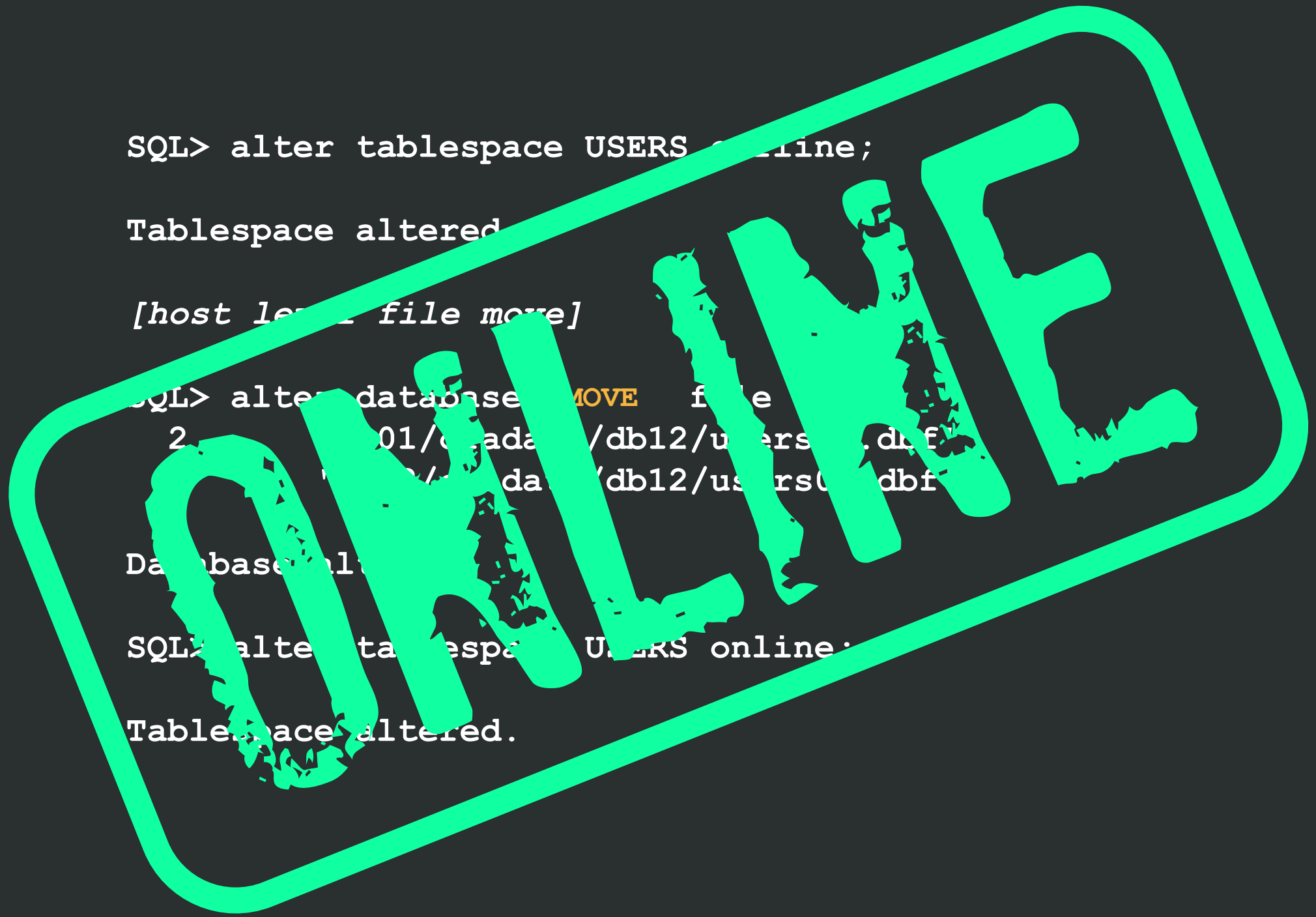
TAKING

OUTAGES

online datafiles

```
SQL> alter tablespace USERS offline;

Tablespace altered.

[host level file move]

SQL> alter database rename file
  2       '/u01/oradata/db12/users01.dbf'
  3   to '/u02/oradata/db12/users01.dbf';

Database altered.

SQL> alter tablespace USERS online;

Tablespace altered.
```

```
SQL> alter tablespace USERS online;

Tablespace altered.

[host level file move]

SQL> alter database MOVE file
  2.        01/oradata/db12/users0.dbf
          /oradata/db12/users0.dbf

Database alt

SQL> alter tablespace USERS online;

Tablespace altered.
```



tempfiles ... just drop/recreate

something very relevant

Here's what we know about the Optus cyber attack, and what you can do to protect your data

By Evlin DuBose · Last updated Monday, 31 October 2022 · 11 min read



At the end of September, Optus announced it'd been the
targeting customer data as far back as 2017. Here's what
breach and how to protect your information.

Australia News

Medibank hack: Urgent action 9.7million Aussies must take after stolen personal data published on the dark web on Wednesday

What 9.7 million Medibank customers need to do immediately after Russian hackers leaked stolen data on the dark web under "good" and "naughty" lists on Wednesday.



Stacey Eldridge Digital Reporter

2 min read November 9, 2022 - 5:35PM sky news .COM.AU

16 comments



transparent data encryption

```
SQL> create table MY_IMPORTANT_STUFF
  2    ( name           varchar2(50),
  3      credit_card    varchar2(20),
  4      ssn            varchar2(20)
  5    )
  6  tablespace users;

Table created.

SQL> insert into  MY_IMPORTANT_STUFF
  2  values ('Connor McDonald',
  3          '4555-6543-8765-1234','666-44-5555');

1 row created.
```

```
[oracle@host122 ~]$ strings users01.dbf
}|{z
2DB122
USERS
AAAAAAAA
Connor McDonald
4555-6543-8765-1234
666-44-5555
...
```

TDE now online as well

```
SQL> select name from v$datafile;

NAME
--------------------------------------------------------------------------------
/u01/oradata/mydb/system01.dbf
/u01/oradata/mydb/sysaux01.dbf
/u01/oradata/mydb/undo01.dbf
/u01/oradata/mydb/users01.dbf

SQL> ALTER TABLESPACE users ENCRYPTION ONLINE USING 'AES192'
  2  ENCRYPT FILE_NAME_CONVERT= ('users', 'users_enc');

SQL> select name from v$datafile;

NAME
--------------------------------------------------------------------------------
/u01/oradata/mydb/system01.dbf
/u01/oradata/mydb/sysaux01.dbf
/u01/oradata/mydb/undo01.dbf
/u01/oradata/mydb/users_enc01.dbf
```

```
[oracle@host122 ~]$ strings users_enc01.dbf
}|{z
2DB122
USERS
AAAAAAAA
siO+
#Hcc
M       ]Q
f\tz
@#cs
ZRk+s C
24Dk
Ah!X
...
```

online schema changes

```
SQL> select bytes/1024/1024 mb
  2  from   user_segments
  3  where  segment_name = 'SALES';

        MB
----------
     52496

SQL> alter table SALES
  2  add TAX_CODE varchar2(20) default 'USA-TAX' not null;

Table altered.

Elapsed: 00:00:00.36
```

```
SQL> select bytes/1024/1024 mb
  2  from   user_segments
  3  where  segment_name = 'SALES';

        MB
----------
     52496          <<== no change
```

```
SQL> insert into sales (...)
  2  values (...);

1 row created.
```

```
SQL> alter table sales modify owner invisible;

Table altered.
```

same with … drop constraint

same with ... drop index

same with … rebuild index

online statistics

```
SQL> create table T as
  2  select *
  3  from my_huge_external_table;

Table created.

Elapsed: 07:13:12.42

SQL> exec dbms_stats.gather_table_stats('','T')

Elapsed: 02:45:07.19
```

```
SQL> create table T as
  2  select *
  3  from my_huge_external_table;

Table created.

Elapsed: 07:13:12.42

SQL> select num_rows, blocks
  2  from   user_tables
  3  where  table_name = 'T';

  NUM_ROWS     BLOCKS
---------- ----------
 734129892     496174
```

create table as select

direct load INSERT

on empty table

online table/index data

reclaim in place





```
SQL> alter table sales enable row movement;

Table altered.

SQL> alter table sales shrink space cascade;

Table altered.
```





our aim: online everything

heap tables

```
SQL> create table t as
  2  select *
  3  from dba_Objects;

Table created.

SQL> create index ix
  2  on t ( object_id );

Index created.
```

```
SQL> insert into t values (...);

1 row created.

....
```

```
SQL> alter table t move online;

[working]
```

```
SQL> commit;

Commit complete.
```

```
SQL> alter table t move online;

[working]

Table altered.

SQL> select index_name, status
  2  from user_indexes;

INDEX_NAME                     STATUS
------------------------------ --------
IX                             VALID
```

composite operations

```
SQL> alter table MY_SALES move
  2    compress
  3    tablespace users
  4      update indexes
  5        (sales_ix1 tablespace indexes01,
  6         sales_ix2 tablespace indexes02)
  7      ONLINE;

Table altered.
```

data subsetting

```
SQL> alter table MY_SALES move
  2    compress
  3    including rows
  4    where SALES_DATE > date '2020-01-01';

Table altered.
```

online structural

partition existing table

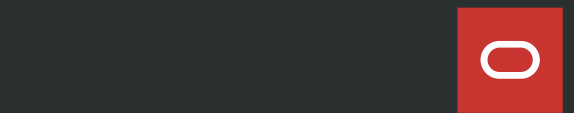
one line command

```
SQL> create table T as
  2  select d.*
  3  from dba_Objects d,
  4       ( select 1 from dual
  5         connect by level <= 20 )
  6  where d.object_id is not null;

Table created.

SQL> create index IX on t ( object_id );

Index created.

SQL> create index IX2 on t ( created, object_name );

Index created.
```



```
SQL> alter table T modify
```

compress



anything to anything

```
SQL> create table t
  2  partition by hash ( owner )
  3  partitions 16
  4  as select * from dba_objects;

Table created.

SQL> alter table T modify
  2  partition by list (owner) automatic
  3  (
  4    partition p1 values ('SYS')
  5  ) online
  6  update indexes;

Table altered.
```



when in doubt ... try online

STOP

TAKING

OUTAGES

4

DataGuard



controversial statement



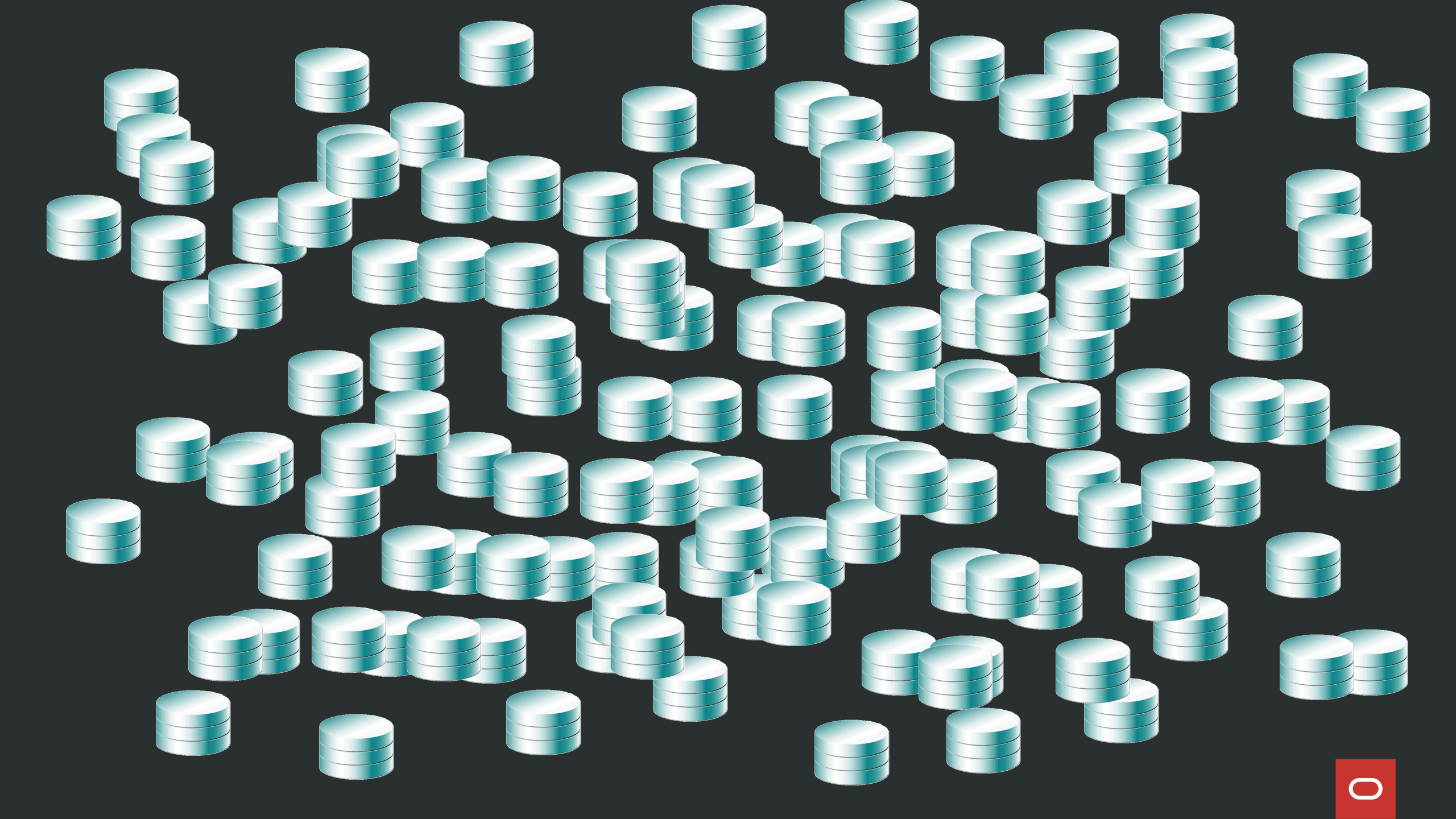
FREE

audience poll









= 1 server



DataGuard is "free"



"I don't need a DR solution"

"I already have a DR solution"

DR is the least benefit



production "playground" is huge benefit



```
SQL> ALTER DATABASE CONVERT TO SNAPSHOT STANDBY;
```



open standby read/write



```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;

SQL> ALTER DATABASE RECOVER MANAGED STANDBY
  2      DATABASE DISCONNECT;
```

flashback to resetlogs



resume managed recovery



archives still transmitted



5

SQL Plan Management



apology #1




ADVISOR
TUNING SET
BASELINES
OUTLINES
PROFILES
TUNING TASK
DIRECTIVES
MANAGEMENT

rarely used

evolve

accepted

capture

apology #2

signature

repeatable

unaccepted

working good ? Keep it

found something better ? Tell me

close to our utopia

found something worse ? Don’t use it



not new

11g



```
SQL> select a.data, b.data
  2  from   tab1 a, tab2
  3  where  b.tab1_id = a.id
  4  and    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 |
---------------------------------------------------------------------
```



parsing continues as per normal



```
SQL> select a.data, b.data
  2  from   tab1 a, tab2
  3  where  b.tab1_id = a.id
  4  and    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 |
-----------------------------------------------------------------------
```



```
SQL> select a.data, b.data
  2  from   tab1 a, tab2
  3  where  b.tab1_id = a.id
  4  and    a.code = :1
```

but we save this !

```
-----------------------------------------------------------------------
|   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 |
-----------------------------------------------------------------------
```





maybe this plan is better ?

```
-----------------------------------------------------------------------
|   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 |
-----------------------------------------------------------------------
```





evolve a plan



DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE



```
SQL> select a.data, b.data
  2  from   tab1 a, tab2
  3  where  b.tab1_id = a.id
  4  and    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 |
-----------------------------------------------------------------------
```





great for upgrades



```
SQL> alter system set
  2     optimizer_capture_sql_plan_baselines = true
```



```
SQL> exec DBMS_SPM.PACK_STGTAB_BASELINE

(datapump)

SQL> exec DBMS_SPM.UNPACK_STGTAB_BASELINE
```



tuning specific SQLs

```
SQL> SELECT p.prod_name,SUM(s.amount_sold)
  2  FROM   sh.products p, sh.sales s
  3  WHERE  p.prod_id = s.prod_id
  4  AND    p.supplier_id = :sup_id
  5  GROUP  BY p.prod_name;

PROD_NAME                                  SUM(S.AMOUNT_SOLD)
-----------------------------------------  ------------------
Baseball trouser Kids                                      91
Short Sleeve Rayon Printed Shirt $8.99                     32
```

```
Plan hash value: 504757596

-------------------------------------------------------------------------------
| Id  | Operation                 | Name     | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |          |    71 |  3550 |  4455   (3)|
|   1 |  HASH GROUP BY            |          |    71 |  3550 |  4455   (3)|
|*  2 |   HASH JOIN               |          |    72 |  3600 |  4454   (3)|
|   3 |    VIEW                   | VW_GBC_5 |    72 |  1224 |  4451   (3)|
|   4 |     HASH GROUP BY         |          |    72 |   648 |  4451   (3)|
|   5 |      PARTITION RANGE ALL|          |   918K|  8075K|  4388   (1)|
|   6 |       TABLE ACCESS FULL | SALES    |   918K|  8075K|  4388   (1)|
|*  7 |    TABLE ACCESS FULL     | PRODUCTS |    72 |  2376 |     3   (0)|
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."PROD_ID"="ITEM_1")
   7 - filter("P"."SUPPLIER_ID"=TO_NUMBER(:SUP_ID))
```



```
SELECT sql_id,
       sql_fulltext
FROM   v$sql
WHERE  sql_text LIKE 'SELECT p.prod_name, %';

SQL_ID          SQL_FULLTEXT
-------------   ------------------------------------
akuntdurat7yr   SELECT p.prod_name, SUM(s.amount_sold)
                FROM   sh.products p, sh.sales s
                WHERE  p.prod_id = s.prod_id
                AND    p.supplier_id = :sup_id
                GROUP  BY p.prod_name;
```



```
SQL> var cnt number
SQL> begin
  2    :cnt := dbms_spm.load_plans_from_cursor_cache(
  3                  sql_id=>'akuntdurat7yr');
  4  end;

PL/SQL procedure successfully completed.


SQL> SELECT sql_handle, plan_name, enabled
  2  FROM   dba_sql_plan_baselines
  3  WHERE  sql_text LIKE 'SELECT p.prod_name, %';

SQL_HANDLE           PLAN_NAME                      ENA
-------------------- ------------------------------ ---
SQL_8f876d84821398cf SQL_PLAN_8z1vdhk11766g42949306 YES
```



```
SQL> var cnt number
SQL> begin
  2     :cnt := dbms_spm.alter_sql_plan_baseline(
  3                     sql_handle=>'SQL_8f876d84821398cf',
  4                     plan_name=>'SQL_PLAN_8z1vdhk11766g42949306',
  5                     attribute_name => 'enabled',
  6                     attribute_value => 'NO');
  7  end;

PL/SQL procedure successfully completed.
```

```
SQL> SELECT /*+ index(p) */ p.prod_name,SUM(s.amount_sold)
  2  FROM   sh.products p, sh.sales s
  3  WHERE  p.prod_id = s.prod_id
  4  AND    p.supplier_id = :sup_id
  5  GROUP  BY p.prod_name;

PROD_NAME                                  SUM(S.AMOUNT_SOLD)
-----------------------------------------  ------------------
Baseball trouser Kids                                      91
Short Sleeve Rayon Printed Shirt $8.99                     32
```



```
SELECT sql_id,
       plan_hash_value,
       sql_fulltext
FROM   v$sql
WHERE  sql_text LIKE 'SELECT /*+ index(p) %';

SQL_ID         PLAN_HASH_VALUE  SQL_FULLTEXT
-------------  ---------------- ------------------------------
avph0nnq5pfc2  2567686925       SELECT /*+ index(p) ...
```



```
SQL> var cnt number
SQL> begin
  2     :cnt := dbms_spm.load_plans_from_cursor_cache(
  3                   sql_id=>' 'avph0nnq5pfc2',
  4                   plan_hash_value=>'2567686925',
  5                   sql_handle=>'SQL_8f876d84821398cf');
  4  end;

PL/SQL procedure successfully completed.
```

hinted SQL/plan

original SQL signature

```
SQL> SELECT sql_handle, plan_name, enabled
  2  FROM   dba_sql_plan_baselines
  3  WHERE  sql_text LIKE 'SELECT p.prod_name, %';

SQL_HANDLE                     PLAN_NAME                                ENA
------------------------------ ---------------------------------------- ---
SQL_8f876d84821398cf SQL_PLAN_8z1vdhk11766g42949306  NO
SQL_8f876d84821398cf SQL_PLAN_8z1vdhk11766ge1c67f67  YES
```



```
Plan hash value: 2567686925

-------------------------------------------------------------------------------
| Id  | Operation                                | Name         | Rows  | Bytes |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |              |    71 |  3550 |
|   1 |  HASH GROUP BY                           |              |    71 |  3550 |
|*  2 |   HASH JOIN                              |              |    72 |  3600 |
|   3 |    VIEW                                  | VW_GBC_5     |    72 |  1224 |
|   4 |     HASH GROUP BY                        |              |    72 |   648 |
|   5 |      PARTITION RANGE ALL                 |              |   918K|  8075K|
|   6 |       TABLE ACCESS FULL                  | SALES        |   918K|  8075K|
|*  7 |    TABLE ACCESS BY INDEX ROWID BATCHED| PRODUCTS     |    72 |  2376 |
|   8 |     INDEX FULL SCAN                      | PRODUCTS_PK  |    72 |       |
-------------------------------------------------------------------------------

Note
-----
- SQL plan baseline SQL_PLAN_8z1vdhk11766ge1c67f67 was used for this statement
```

wrap up





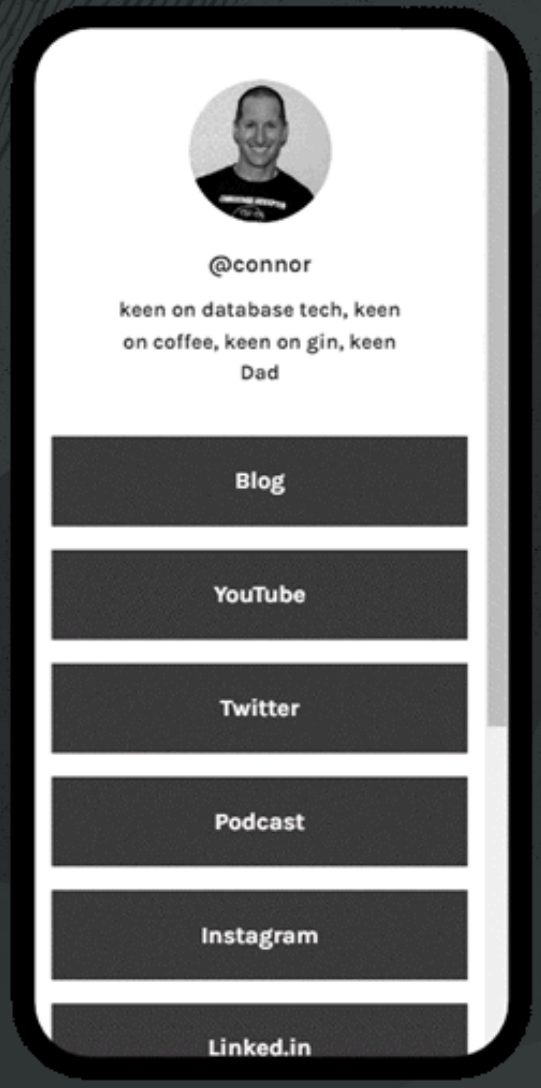
is scary big



reach out to others



Let's connect



@connor_mc_d

https://linktr.ee/connor