Oracle 数据库12c-21c 开发必用特性

公益讲座11：00准时开始，请大家先浏览云技术微信公众号技术文章。资料会在各群同步发布，已入群客户请勿重复入群！

20-22

数据库和云讲座群

甲骨文云技术公众号

B站专家系列课程

* 活动最终解释权归甲骨文公司所有

立即扫码进行 1V1 免费咨询

2023 年 10 月，MySQL 5.7 将终止官方支持和更新。

立刻升级至更快、更稳定、更安全的 MySQL 8.0 / MySQL Database Service，获取 300+ 项新特性，使开发更加灵活和高效，更好的满足业务发展需求。

免费咨询热线：

400-699-8888

Oracle 数据库 12c -- 21c 开发必用新特性

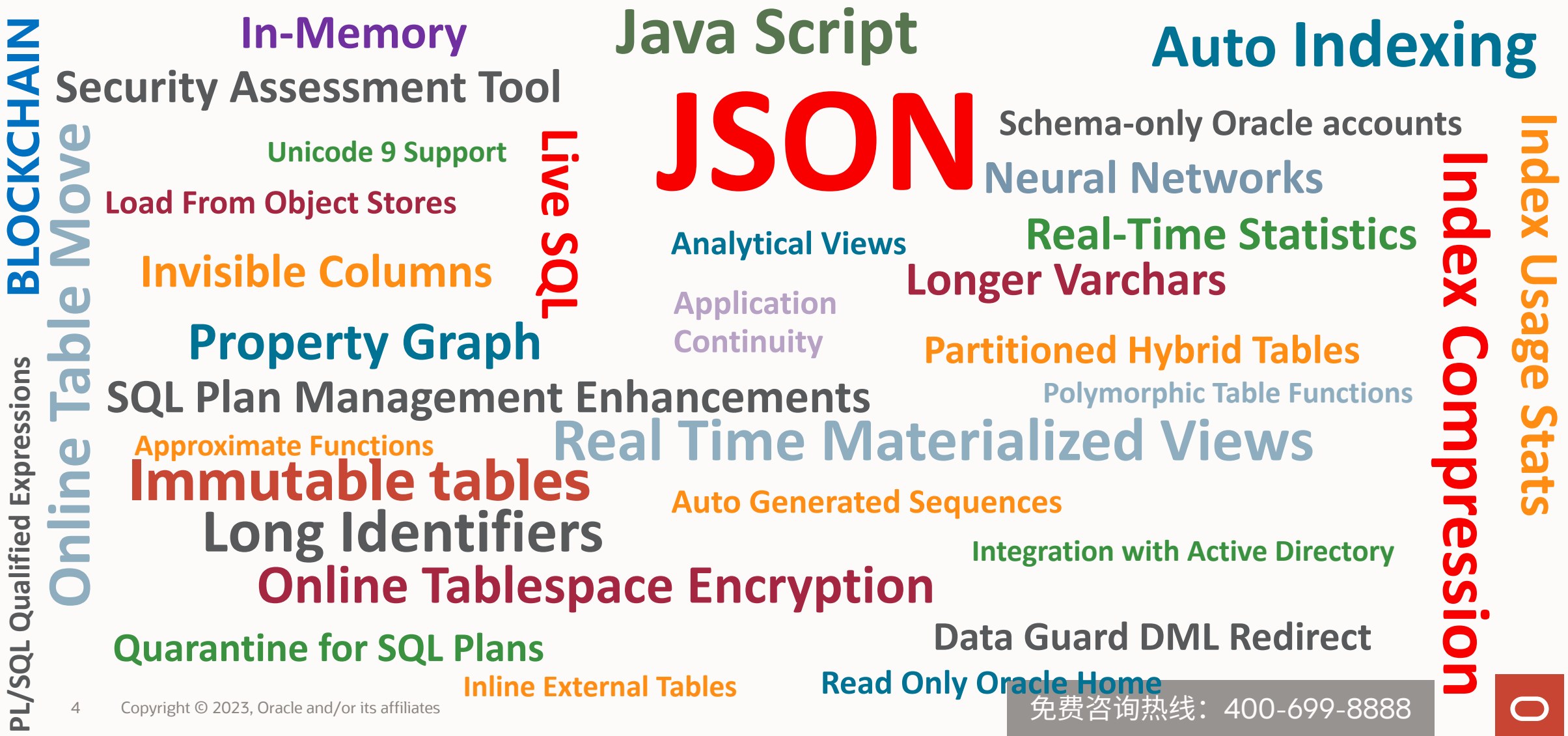
甲骨文技术公益课 - 数据库专场

2023 年 9 月 22 日 11:00

线上直播

阮蓉

Oracle DB 诸多特性



NEW IN 18c

Scalable Sequences
可扩展序列

可扩展序列 - Scalable Sequence

NEW IN 18c

通过在CREATE SEQUENCE或ALTER SEQUENCE语句中指定SCALE子句，可以使序列获得健壮的扩展性。

CREATE | ALTER SEQUENCE sequence_name ... SCALE [EXTEND | NOEXTEND] | NOSCALE ...

当 SCALE 语句被指定时, 一个 6 位数的数字被指定作为序列的前缀，末尾是正常的序列数字，两者联合成为新的序列：

scalable sequence number = 6 digit scalable sequence offset number || normal sequence number

在这里, 6 位数字前缀是如何生成的呢？正是由 实例号 和 会话号 生成的：

6 digit scalable sequence offset number = 3 digit *instance* offset number || 3 digit *session* offset number.

现在通过这种序列方式，能够真正将来自不同实例的数据分散开来，从而使RAC可扩展，而且可以把主键分散开索引竞争大大降低，从而提升了性能，使得序列变得可扩展。

```
create sequence scale_seq scale extend;

select scale_seq.nextval from dual
connect by level <= 5;
```

Sequence values

```
NEXTVAL
---------------------
101126000000000000000000000000001
101126000000000000000000000000002
101126000000000000000000000000003
101126000000000000000000000000004
101126000000000000000000000000005
```

Instance & session offset

```
alter sequence scale_seq scale noextend;

select scale_seq.nextval from dual
connect by level <= 5;

NEXTVAL
-------------------------
1011260000000000000000000006
1011260000000000000000000007
1011260000000000000000000008
1011260000000000000000000009
1011260000000000000000000010
```

Scalable Sequence -- 需要注意的细节

```
create sequence scale_noextend_seq start with 1
increment by 1 minvalue 1 maxvalue 9999999
scale noextend;


select scale_noextend_seq.nextval from dual
  NEXTVAL
----------
  1016281
。。。
  NEXTVAL
----------
  1016289

超过maxvalue 最大长度会报错。
SQL> /
select scale_noextend_seq.nextval from dual
ORA-64603: 无法为 SCALE_NOEXTEND_SEQ 实例化 NEXTVAL。
将队列放宽 1 位数或使用 SCALE EXTEND 变更队列。


alter sequence scale_noextend_seq scale extend
```

NEW IN 18c

私有临时表 Private Temporary Tables

私有临时表

NEW IN 18c

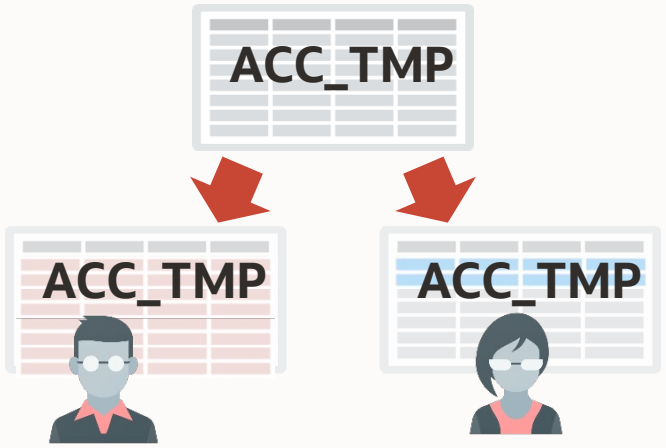
在Oracle数据库18c中，我们为临时表提供了更灵活的使用：私有临时表，一个在内存中用完即弃的临时对象。如果开发人员需要为不同的事务创建不同的临时表，或在只读数据库使用临时表时，都可以使用私有临时表。这种方法为开发人员提供了更大的灵活性。

属性	全局临时表	私有临时表
命名规则	跟普通表一样	必须以 ORA$PTT_为前缀
表定义可见	所有会话可见	仅仅建立该表的会话可见
表定义存储	硬盘	仅在内存

私有临时表

短暂存在的表适用于报表程序

NEW IN 18c

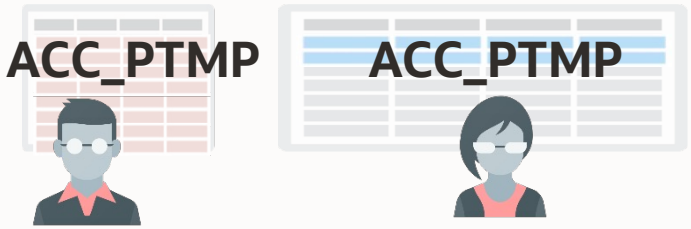


全局临时表

持久的，共享（全局）表定义

临时, 私有 (基于会话) 数据内容

- 数据物理存在会话或事务期间
- 私有会话统计



私有临时表(18c)

- 临时, 私有(基于会话) 表定义
  - 私有表名和定义
- 临时, 私有 (基于会话) 数据内容
  - 会话或事物期间

private_temp_table_prefix

```
create private temporary table
  ora$ptt_file_data (
  c1 varchar2(2),
  c2 varchar2(2),
  c3 varchar2(2)
);
```

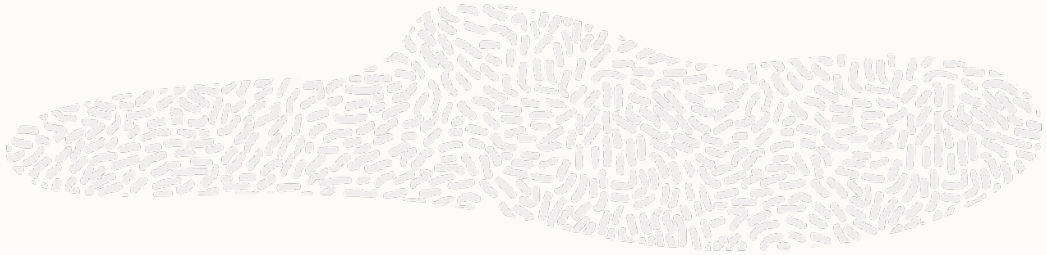
Only exists in your *session*

Session 1

```
create private temporary table
  ora$ptt_file_data (
  c1 varchar2(2),
  c2 varchar2(2),
  c3 varchar2(2)
);

insert into ora$ptt_file_data values
(
  'v1', 'v2', 'v3'
);
```

Session 2

```
create private temporary table
  ora$ptt_file_data (
  c1 integer,
  c2 date
);

insert into ora$ptt_file_data values
(
  1, sysdate
);
```

```
commit;

select * from ora$ptt_file_data;

ORA-00942: table or view does not exist
```

```
create private temporary table
  ora$ptt_<table_name> (
  <columns>
) on commit
  [ preserve | drop ] definition;
```

default

```
SQL> SHOW PARAMETER PRIVATE_TEMP_TABLE_PREFIX

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------
private_temp_table_prefix            string      ORA$PTT_
SQL> CREATE PRIVATE TEMPORARY TABLE ora$ptt_my_temp_table (
  id           NUMBER,
  description  VARCHAR2(20)
)
ON COMMIT DROP DEFINITION;  2    3    4    5

Table created.

SQL> INSERT INTO ora$ptt_my_temp_table VALUES (1, 'ONE');

1 row created.

SQL> commit;

Commit complete.

SQL> SELECT COUNT(*) FROM ora$ptt_my_temp_table;
SELECT COUNT(*) FROM ora$ptt_my_temp_table
                     *
ERROR at line 1:
ORA-00942: table or view does not exist
```

```
SQL> CREATE PRIVATE TEMPORARY TABLE ora$ptt_my_temp_table (
  id           NUMBER,
  description  VARCHAR2(20)
)
ON COMMIT PRESERVE DEFINITION;  2    3    4    5

Table created.

SQL> INSERT INTO ora$ptt_my_temp_table VALUES (1, 'ONE');

1 row created.

SQL> commit;

Commit complete.

SQL> SELECT COUNT(*) FROM ora$ptt_my_temp_table;

  COUNT(*)
----------
         1

SQL> conn test/test@orclpdb
Connected.
SQL> SELECT COUNT(*) FROM ora$ptt_my_temp_table;
SELECT COUNT(*) FROM ora$ptt_my_temp_table
                     *
ERROR at line 1:
ORA-00942: table or view does not exist
```

NEW IN 12c

Oracle自增字段 IDENTITY

自动产生主键PKs

Primary keys, document numbers, case numbers, trace IDs等等.

NEW IN 12c

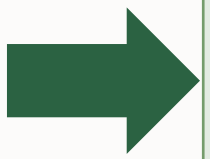
```
CREATE SEQUENCE movie_seq;


CREATE TRIGGER movie_ids
  BEFORE INSERT ON movie
  FOR EACH ROW
BEGIN
  IF :new.movie_id IS NULL THEN
    :new.movie_id :=
movie_seq.NEXTVAL;
  END IF;
END;
```

自动产生主键PKs - IDENTITY columns (ISO SQL 标准)

NEW IN 12c

Primary keys, document numbers, case numbers, trace IDs等等.

```
CREATE SEQUENCE movie_seq;


CREATE TRIGGER movie_ids
  BEFORE INSERT ON movie
  FOR EACH ROW
BEGIN
  IF :new.movie_id IS NULL THEN
    :new.movie_id := movie_seq.NEXTVAL;
  END IF;
END;
```



```
CREATE TABLE movie (
  movie_id   NUMBER
    GENERATED BY DEFAULT AS IDENTITY
    START WITH 100 INCREMENT BY 10,
  …
);


--set identity nextval > movie_id max value
ALTER TABLE movie
MODIFY movie_id
  GENERATED BY DEFAULT ON NULL AS IDENTITY
  START WITH LIMIT VALUE
```

```
create table transactions (
  transaction_id integer
    generated always as identity
    not null primary key,
    name varchar2(32)
);
```

```
create table transactions (
  transaction_id integer
    generated by default as identity
    not null primary key,
  ...
);
```

```
CREATE TABLE FIPR_MASTER (
FIPR_Id INT GENERATED ALWAYS AS IDENTITY
(START WITH 100 INCREMENT BY 1 MINVALUE
100 NOMAXVALUE NOCYCLE NOCACHE ORDER)
PRIMARY KEY, FIPR_NAME VARCHAR2(40) NOT
NULL,
BU_NAME VARCHAR2(40) NOT NULL,
LOCATION VARCHAR2(40) NOT NULL, …….
```

```
SQL> create table tab2(
  2  id integer generated by default as identity,
  3  name varchar2(32));

Table created.
```

```
insert into tab2 values(2, 'Vincent');
insert into tab2(name) values('Vicky');
insert into tab2(name) values('Grace');
```

```
SQL> insert into tab2 values(2, 'Vincent');

1 row created.

SQL> insert into tab2(name) values('Victor');

1 row created.

SQL> insert into tab2(name) values('Grace');

1 row created.
```

```
SQL> select * from tab2;

        ID NAME
---------- --------------------------------
         2 Vincent
         1 Victor
         2 Grace
```

```
SQL> create table transactions (
  transaction_id integer
    generated always as identity
    not null primary key,
    name varchar2(32)
);
  2    3    4    5    6
Table created.

SQL> insert into transactions values(1,'vincent');
insert into transactions values(1,'vincent')
*
ERROR at line 1:
ORA-32795: cannot insert into a generated always identity column
```

```
SQL> insert into transactions(name) values('vincent');

1 row created.

SQL> commit;

Commit complete.

SQL> select * from transactions;

TRANSACTION_ID NAME
-------------- --------------------------------
             1 vincent
```

NEW IN 12c

Bigger varchar2

32k VARCHAR2/NVARCHAR2

更长的字符串存储

- 在Oracle12c之前，不管使用什么字符语义，数据库中VARCHAR2、NVARCHAR2和RAW列的最大大小如下所示：
  - VARCHAR2 : 4000 bytes
  - NVARCHAR2 : 4000 bytes
  - RAW : 2000 bytes
- 随着扩展数据类型的引入，Oracle12c可以选择增加这些最大大小：
  - VARCHAR2 : 32767 bytes
  - NVARCHAR2 : 32767 bytes
  - RAW : 32767 bytes

- 启用32k VARCHAR2支持

```
ALTER SYSTEM set MAX_STRING_SIZE = EXTENDED
scope = SPFILE;
```

- 用32k VARCHAR2创建一个表

```
CREATE TABLE Applications
(id           NUMBER GENERATED AS IDENTITY,
 first_name   VARCHAR2(30),
 last_name    VARCHAR2(30),
 application  DATE,
 CV           VARCHAR2(32767)
);
```

32k VARCHAR2/NVARCHAR2

更长的字符串存储

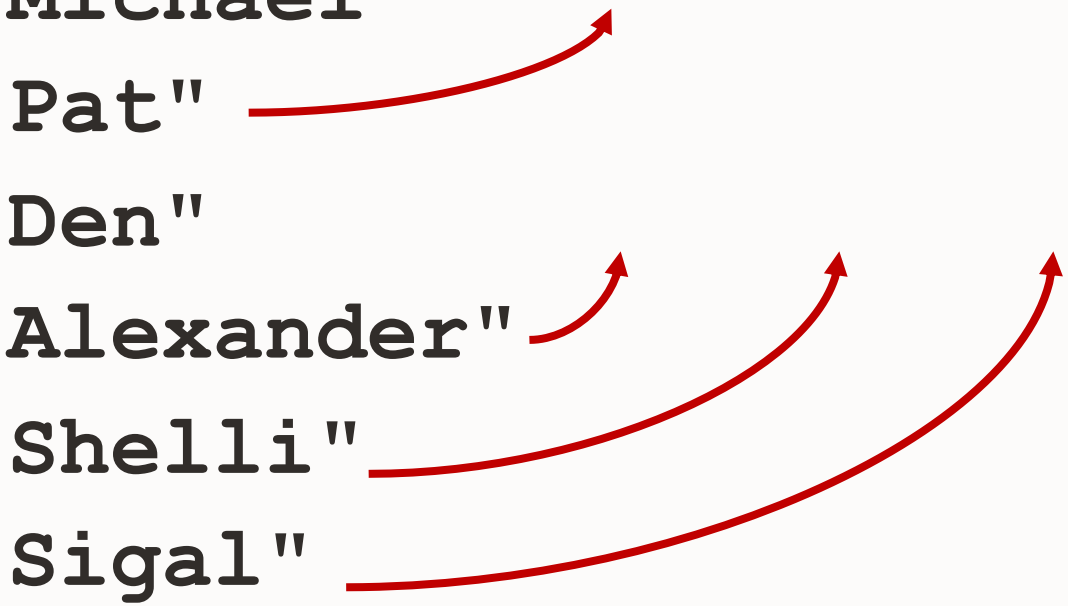
- 启用32k VARCHAR2支持

1.查看参数max_string_size默认值
SQL> show parameter max_string_size
NAME TYPE VALUE
---------- ---------- ----------
max_string_size string STANDARD
2. 关闭数据库
SQL> shutdown immediate;
3. 以升级模式重启数据库
SQL> startup upgrade;
3. 更改参数: ALTERSYSTEM SET MAX_STRING_SIZE=EXTENDED;
SQL> alter system set max_string_size=extended scope=spfile;
System altered.
4. 执行 utl32k.sql as sysdba
SQL> @$ORACLE_HOME/rdbms/admin/utl32k.sql
说明：如果环境是cdb+pdb，那么所有的cdb + pdb$seed + pdb 都需要在 startup upgrade; 状态下执行script脚本 @$ORACLE_HOME/rdbms/admin/utl32k.sql 。不然，只扩展完cdb后，pdb不能打开，报ORA-14694: database must in UPGRADE mode to begin MAX_STRING_SIZE migration
5. 检查一下有没有产生失效对象
SQL> select count(*) from dba_objects where status<>'VALID';
COUNT(*)

0
6. 关闭数据库
SQL> shutdown immediate;

listagg

```
"DEPARTMENT_ID","FIRST_NAME"
10,"Jennifer"
20,"Michael"
20,"Pat"
30,"Den"
30,"Alexander"
30,"Shelli"
30,"Sigal"
...
```



```
"DEPARTMENT_ID","FIRST_NAME"
10,Jennifer
20,Michael,Pat
30,Den,Alexander,Shelli,Sigal,…
…
```

```
select table_name,
       listagg(index_name, ',') within group
         (order by index_name) inds
from   user_indexes
group  by table_name;
```

```
ORA-01489: result of string concatenation is
too long
```

NEW IN 12c

listagg (
  improved on overfloooooow
)

```
select owner,
       listagg (
         object_type, ','
       on overflow truncate
       ) within group (
         order by object_type
       )
from   dba_objects
group  by owner;
```

```
SCOTT    …,TABLE,TABLE
SH       …,TABLE PARTITION,VIEW
SYS      …,JAVA CLASS,JAVA CLASS,...(48638)
SYSTEM   …,VIEW,VIEW
```

```
SCOTT    …,TABLE
SH       …,TABLE PARTITION,VIEW
SYS      …,JAVA CLASS,
SYSTEM   …,VIEW
```

NEW IN 19c

listagg (distinct)

```
select owner,
       listagg (
         distinct object_type, ','
         on overflow truncate
       ) within group (
         order by object_type
       )
from   dba_objects
group  by owner;
```

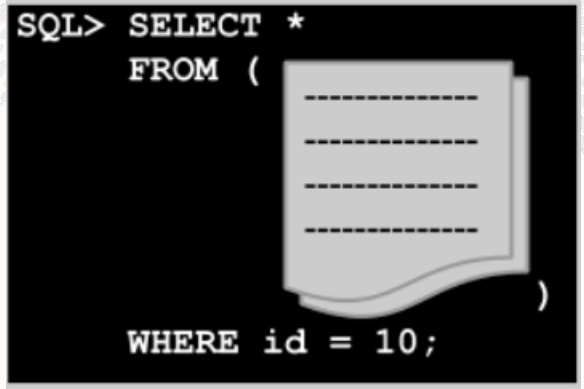
```
SCOTT   INDEX,TABLE
SH      …,TABLE,TABLE PARTITION,VIEW
SYS     …,UNIFIED AUDIT POLICY,VIEW,WINDOW
SYSTEM …,TABLE PARTITION,TYPE,VIEW
```

NEW IN 18c

内联外部表 <Inline> External Tables

内联外部表

透明地访问外部数据



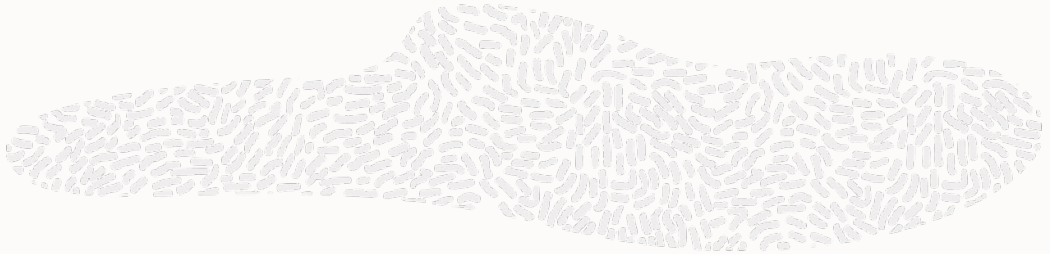
- 运行时提供的外部表定义
  - 类似于内联视图
- 不需要预先创建只使用一次的外部表
  - 提高开发人员的工作效率

```
CREATE TABLE sales_xt
    (prod_id number, … )
     TYPE ORACLE_LOADER
     …
     LOCATION 'new_sales_kw13')
     REJECT LIMIT UNLIMITED );

INSERT INTO sales SELECT * FROM
sales_xt;

DROP TABLE sales_xt;
```

```
INSERT INTO sales
SELECT sales_xt.*
FROM EXTERNAL(
      (prod_id number, … )
       TYPE ORACLE_LOADER
       …
       LOCATION 'new_sales_kw13')
       REJECT LIMIT UNLIMITED );
```

内联外部表

透明地访问外部数据



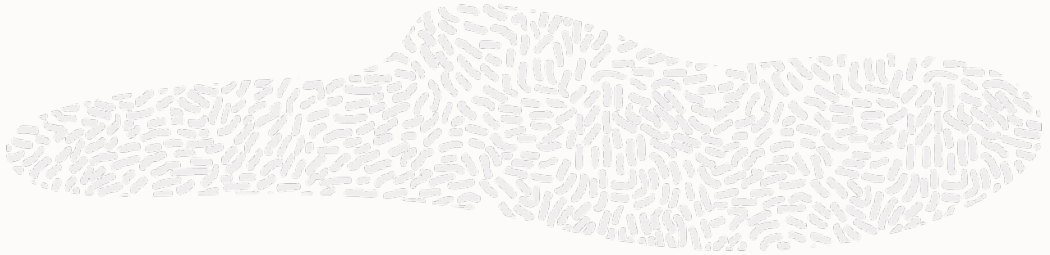
```
SELECT * FROM   EXTERNAL (
     (time_id          DATE NOT NULL,
      prod_id          INTEGER NOT NULL,
      quantity_sold    NUMBER(10,2),
      amount_sold      NUMBER(10,2))
     TYPE ORACLE_LOADER
     DEFAULT DIRECTORY data_dir1
     ACCESS PARAMETERS (
       RECORDS DELIMITED BY NEWLINE
       FIELDS TERMINATED BY '|')
    LOCATION ('sales_9.csv') REJECT LIMIT UNLIMITED)
sales_external;
```

NEW IN 12c

Top-N/pagination queries

Top-N/分页查询

Select only the first N rows



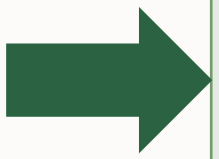
```
SELECT *
FROM   (
  SELECT title, movie_details,
         ROWNUM rn
  FROM   movie
  ORDER BY release_date
)
WHERE  rn <= 5;
```

Top-N/分页查询- Row limit (ISO SQL Standard)

NEW IN 12c

Select only the first N rows

```
SELECT *
FROM   (
  SELECT title, movie_details,
         ROWNUM rn
  FROM   movie
  ORDER BY release_date
)
WHERE  rn <= 5;
```



```
-- first 5 rows
SELECT   title, movie_details
FROM     movie
ORDER BY release_date
FETCH FIRST 5 ROWS ONLY;

-- first 5% rows with duplicates for 5%
SELECT   title, movie_details, list_price
FROM     custsales
ORDER BY list_price
FETCH FIRST 5 PERCENT ROWS WITH TIES;
```

Top-N查询处理

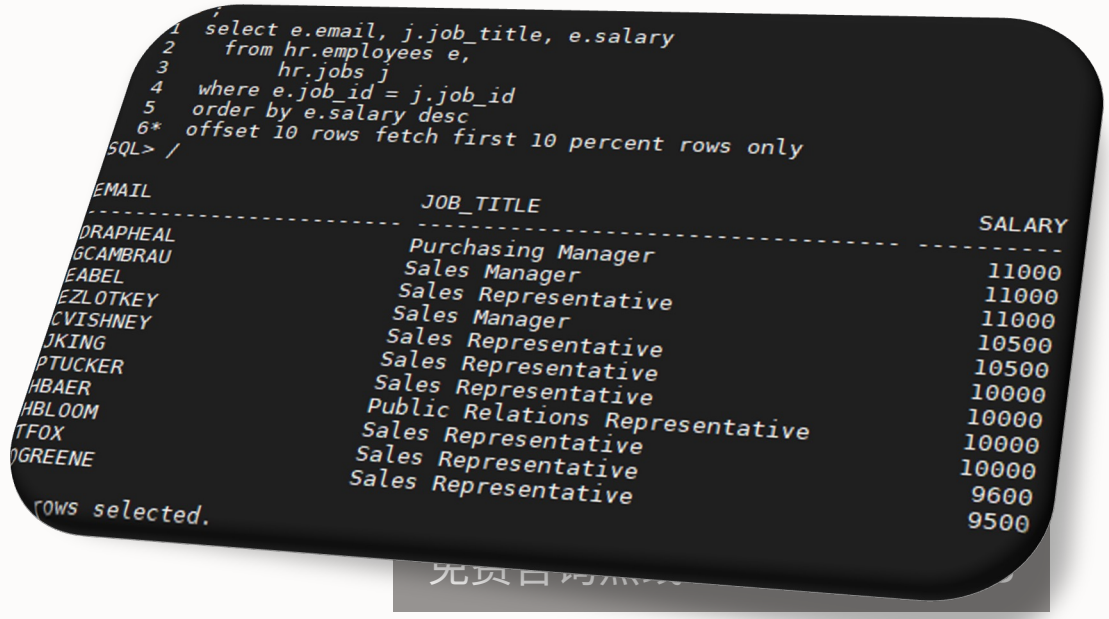
- 可以通过特定的行数或行的百分比来限制SQL查询返回的行数；
- 分页查询；

[OFFSET offset { ROW | ROWS }]
[FETCH { FIRST | NEXT } [{ rowcount | percent PERCENT }]
{ ROW | ROWS } { ONLY | WITH TIES }]

offset	指定跳跃多少行开始计数，即从结果集的第offset+1个记录开始返回
row/rows	这些关键字可以交替使用，并且是为了语义清晰而提供的;两者没区别
fetch	使用这个子句去指定返回行的个数或者返回行的百分比。如果没有指定，那么从offset+1行开始的所有行都会被返回，可以用作分页查询
frist/next	使语句的语义更清晰，构成语法的完整性，两者没有区别
rowcount	指定返回记录的数量
precent	指定返回记录的数量占整个结果集的百分比
only	指定only会返回明确的行数或者是百分比的行数
with ties	如果指定with ties子句，那么拥有和最后一行相同的排序键值的行都会被fetch。如果指定了with ties子句，那么必须指定order by 。如果没有指定order by，那么不会有附加的行被返回。

举例：请找出公司除top 10收入员工外，前10%的员工 (公司考虑给他们加10%的工资)：

```
SQL> SELECT e.email, j.job_title, e.salary
        FROM hr.employees e, hr.jobs j
      WHERE e.job_id = j.job_id
       ORDER BY e.salary DESC
       OFFSET 10 ROWS FETCH FIRST 10
PERCENT ROWS ONLY;
```



NEW IN 18c

Top-N 近似值聚合计算

Approximate Top-N Query Processing

Top-N 近似值聚合计算

减少TB级数据响应时间

NEW IN 18c

- 常见Top-N查询近似结果
  - 上周前五大博客文章，有大致多少次浏览量?
  - 每个地区的前50名客户，他们的大概消费是多少?
- 速度更快，处理精度更高 (准确度通常为 97%，错误率< 0.5%)，5-50x性能提升
- 新添加近似函数 APPROX_COUNT(), APPROX_SUM(), APPROX_RANK()

Top 5 blogs with approximate hits

```
SELECT blog_post, APPROX_COUNT(*)
FROM weblog
GROUP BY blog_post
FETCH FIRST 5 ROWS ONLY;
```

Top 50 customers per region with approximate spending

```
SELECT region, customer_name,
       APPROX_RANK(PARTITION BY region
                   ORDER BY APPROX_SUM(sales) DESC) appr_rank,
       APPROX_SUM(sales) appr_sales
FROM sales_transactions
GROUP BY region, customer_name
HAVING APPROX_RANK(..) <=50;
```

```
SELECT department,
       record_type,
       APPROX_SUM(record_value) AS approx_sum_val,
       APPROX_RANK(PARTITION BY department ORDER BY APPROX_SUM(record_value) DESC) AS approx_rank_val
FROM   t1
GROUP BY department, record_type
HAVING APPROX_RANK(PARTITION BY department ORDER BY APPROX_SUM(record_value) DESC) <= 2
ORDER BY 1, 4;
```

```
DEPARTMENT RECORD_TYPE  APPROX_SUM_VAL APPROX_RANK_VAL
---------- ------------ -------------- ---------------
        10 FIVE                 128894               1
        10 FOUR                 128618               2
        20 THREE                128108               1
        20 TWO                  127481               2
        30 FIVE                 130705               1
        30 TWO                  130174               2
        40 FIVE                 128375               1
        40 ONE                  127721               2
```

NEW IN 21c

SQL Macro

在查询中使用函数

- 可以在查询中使用函数
  - 需要在 SQL 和 PL/SQL 引擎之间进行上下文切换
  - 减慢查询速度

```
create or replace function
  is_president (p_mgr number)
return varchar2
as
begin
  return( case when p_mgr is null
               then 1 else 0 end );
end;
```

```
SQL> select ename, job, is_president(mgr)
  2> from emp;

ENAME  JOB           IS_PRESIDENT
====== ============ ============
KING   PRESIDENT    1
BLAKE  MANAGER      0
CLARK  MANAGER      0
JONES  MANAGER      0
SCOTT  ANALYST      0
```

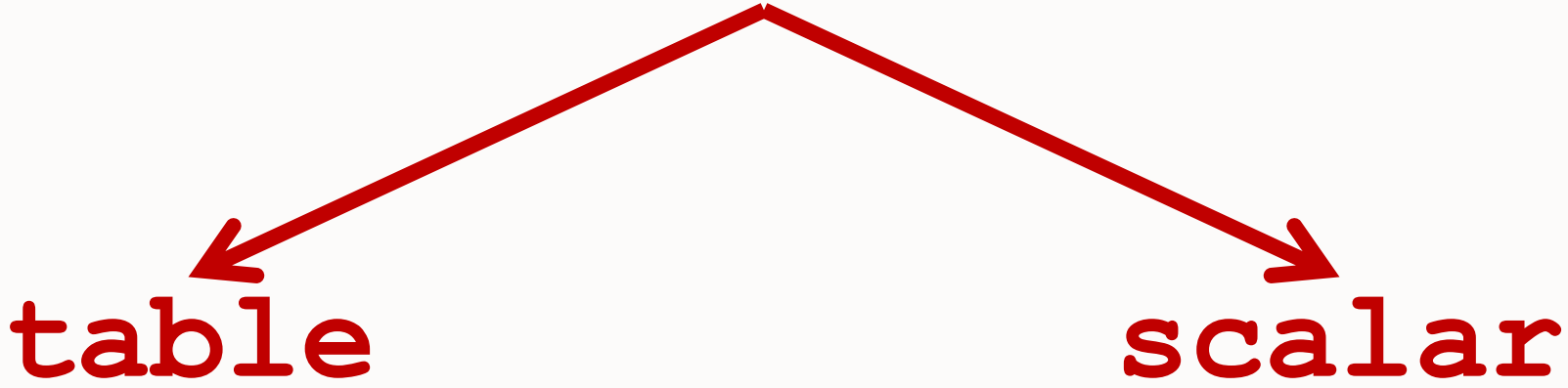
为标量函数引入 SQL 宏

- 首先将函数的返回值交换给查询
  - 充当 SQL 预处理器
  - 可重用的代码对 SQL Optimizer 透明
  - 轻松创建，可重用且可移植的代码
  - 没有昂贵的上下文切换

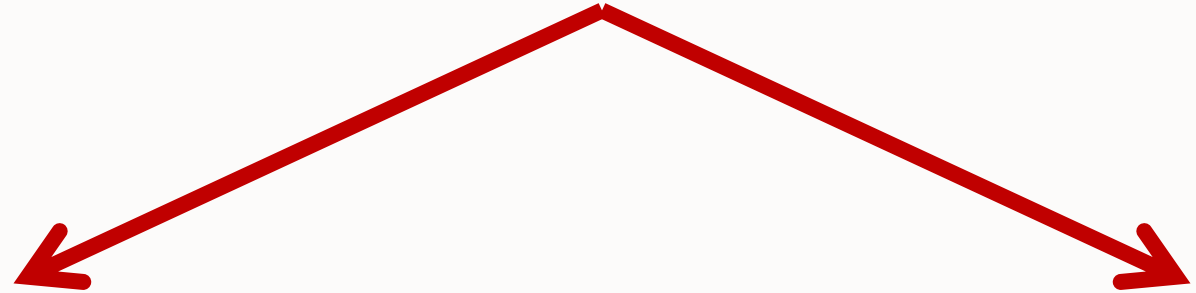
```
create function
  is_president (p_mgr number)
return varchar2 SQL_MACRO(SCALAR)
as
begin
  return('case when p_mgr is null
              then 1 else 0 end');
end;
```

```
SQL> select ename, job, is_president(mgr)
  2> from emp;

ENAME  JOB           IS_PRESIDENT
====== ============ ============
KING   PRESIDENT    1
BLAKE  MANAGER      0
CLARK  MANAGER      0
JONES  MANAGER      0
SCOTT  ANALYST      0
```

SQL Macros



SQL Macros



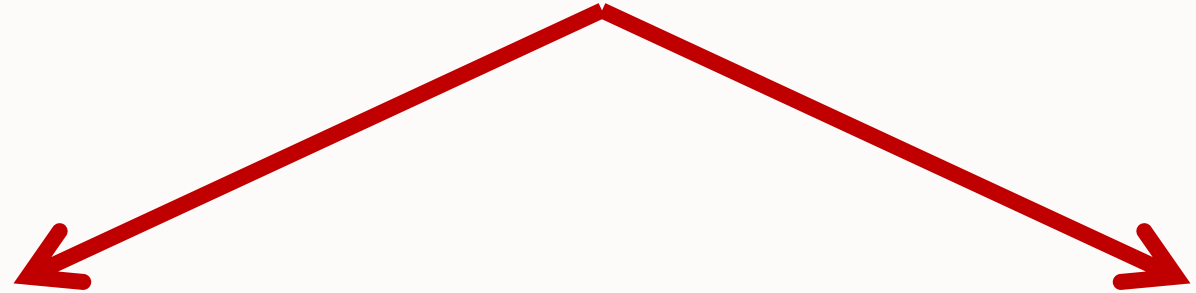
`table`

in from clause

`scalar`

"everywhere" else

SQL Macros



`table`

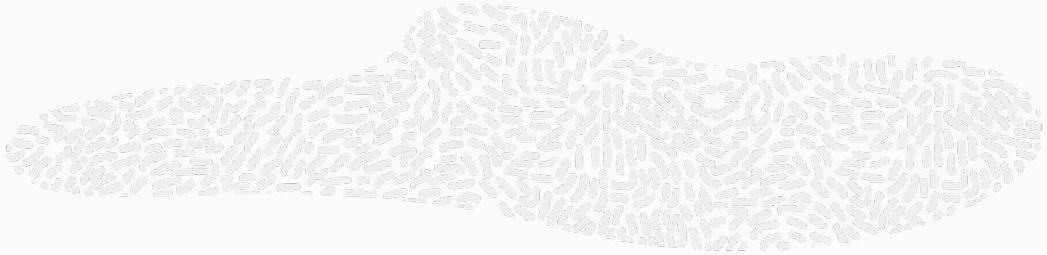
`scalar`

in from clause

only these
in 19.7

标量的SQL Macros

标量宏提供了一种封装复杂SQL表达式的简单方法



- 可以用于 SELECT 列表、WHERE/HAVING、GROUP BY/ORDER BY子句

```
create or replace function
       sales_tax(unit_cost number,
                 unit_type varchar)
return varchar2 SQL_MACRO(SCALAR) is
begin
return q'[case when unit_type = 'FOOD'
         then unit_cost
         else unit_cost * 1.2 end]';
end;
```

FUNCTION RETURNS STRING

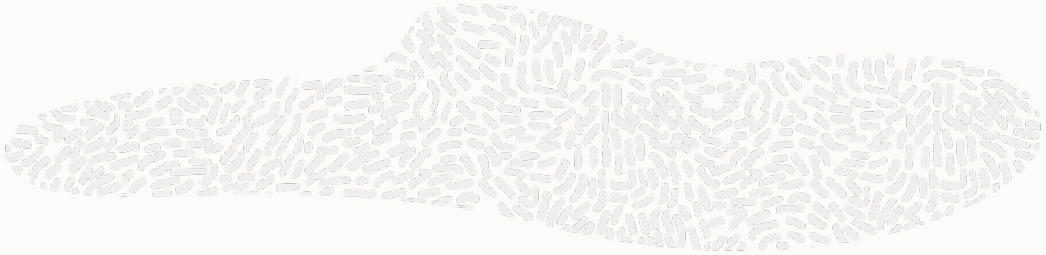
```
SQL> select sales_tax(20,'WINE') from dual;

SALES_TAX(20,'WINE')
--------------------
                  24
```

EQUIVALENT TO

```
SQL> select case
            when 'WINE' = 'FOOD' then 20
            else 20*1.2
        end
     from dual;
```

表的SQL Macros

SQL表宏封装了FROM子句中使用的SQL



```
CREATE OR REPLACE FUNCTION budget
return varchar2 SQL_MACRO
IS
BEGIN
RETURN q'( select department_id,
sum(salary) budget
from hr.employees
group by department_id )';
END;
```

```
SQL> SELECT * FROM budget() WHERE
department_id IN (10,50);
```

```
DEPARTMENT_ID BUDGET
------------- ----------
50            156400
10            4400
```

(19.7)

```
select department_id, sum ( salary )
from   employees
where  job_id = dept_job_salaries.job
group  by department_id
```

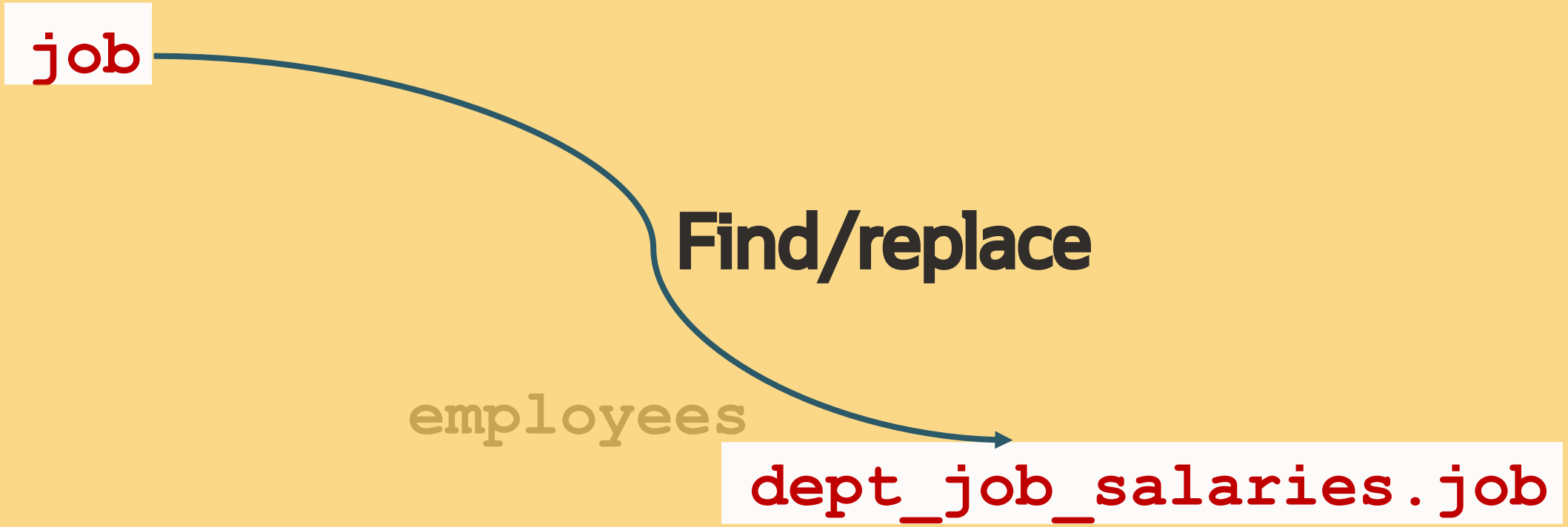
(19.7)

```
return '
    select department_id, sum ( salary )
    from   employees
    where  job_id = dept_job_salaries.job
    group  by department_id';
```

(19.7)

```
create function dept_job_salaries (
   job varchar2
) return varchar2 sql_macro is
begin
   return '
      select department_id, sum ( salary )
      from   employees
      where  job_id = dept_job_salaries.job
      group  by department_id';
end;
```

(19.7)



(19.7)

```
select *
from   dept_job_salaries ( 'MANAGER' );
```

(19.7)

```
select *
from   dept_job_salaries ( 'MANAGER' );

select department_id, sum ( salary )
from   employees
where  job_id = 'MANAGER'
group  by department_id;
```

(19.7)

```
select *
from   dept_job_salaries ( :job );
```

(19.7)

```
select *
from   dept_job_salaries ( :job );

select department_id, sum ( salary )
from   employees
where  job_id = :job
group  by department_id;
```

(19.7)

SQL Macro
多态视图
(Polymorphic Views)

(19.7ish)

```
select * from top_n.tab
fetch first top_n.num_rows
   rows only
```

(19.7)

```
create function top_n (
   num_rows number, tab dbms_tf.table_t
) return varchar2 sql_macro is
begin
   return
      'select * from top_n.tab
       fetch first top_n.num_rows
          rows only';
end;
```

(19.7)

(19.7)

```
from top_n (
   5, employees
)
```

(19.7)

```
from top_n (
   5, employees
)
```

```
from employees
fetch first 5
   rows only
```

(19.7)

```
from top_n (
   5, employees
)
```

```
from employees
fetch first 5
   rows only
```

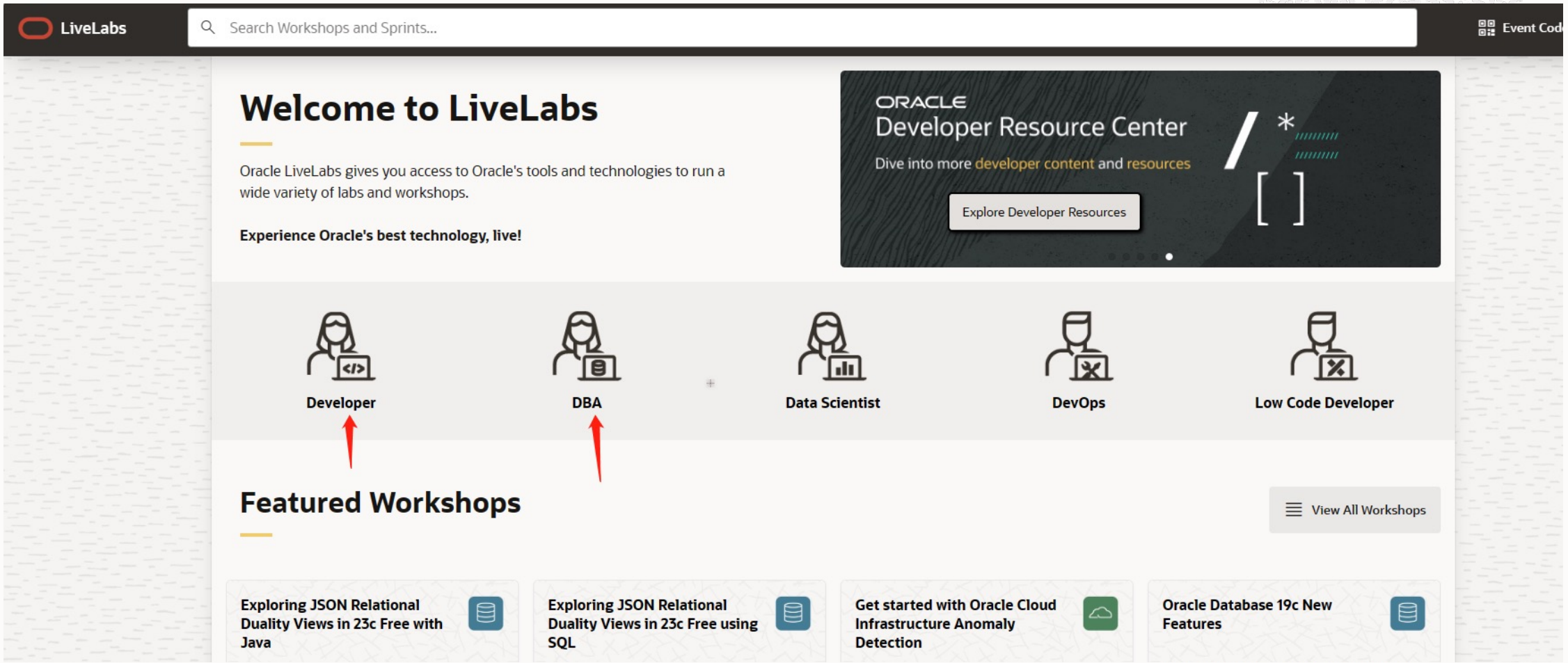
```
from top_n (
   :bind, jobs
)
```

```
from jobs
fetch first :bind
   rows only
```

(19.7)

LiveLabs

[LiveLabs Home (oracle.com)](https://apexapps.oracle.com/pls/apex/r/dbpm/livelabs/home) https://apexapps.oracle.com/pls/apex/r/dbpm/livelabs/home



Clear Search & Filters

Sort By
Most Popular

Number of Workshops: 833

Level

- [ ] Beginner (455)
- [ ] Intermediate (343)
- [ ] Advanced (37)

Workshop Type

- [ ] Paid Credits (520)
- [ ] Sprints (303)
- [ ] Run on LiveLabs (184)
- [ ] Run on Gov Cloud
- [ ] ADB for Free

Workshop Series

- [ ] Autonomous DB Serverless (21)
- [ ] DBA Masterclass Series (17)
- [ ] Database Security Series (15)
- [ ] DBA Essentials (12)
- [ ] MovieStream Series (11)

Show All

Role

- [ ] IT (465)

Workshops and Sprints

Get Started with Oracle Cloud Infrastructure Core Services

Explore basic OCI services: Networking, Compute, Storage and more.

2 hrs, 30 mins — 123121 Views

Load and Analyze Your Data with Autonomous Database

Perform basic tasks on Autonomous Database: provision, load data, create data visualizations.

1 hr, 30 mins — 99361 Views

Getting started with OCI Data Science

Explore concepts used in the Oracle Cloud Infrastructure Data Science service to speed up your (..)

1 hr, 30 mins — 72468 Views

Converting your Spreadsheet into a Cloud App using Oracle APEX

Create an application in minutes by simply uploading a spreadsheet!

45 mins — 59926 Views

Build a Starter Online Shopping App using Oracle APEX!

In this workshop, developers will learn how to take advantage of Oracle APEX components to create a (..)

2 hrs — 52594 Views

Create a Web Application in Oracle Visual Builder

Learn how to develop a web application that displays a user interface through which users can view, (..)

1 hr, 55 mins — 51905 Views

Launch Your First MySQL Database Service System

In this workshop understand the powerful union between MySQL Enterprise Edition and OCI, learn to (..)

1 hr, 30 mins — 51294 Views

Deploy an OKE Cluster Using Oracle Cloud Shell

Deploy an OKE cluster, connect to it and run a sample application all using OCI CLI.

2 hrs — 51232 Views

Simplify Microservices with Converged Oracle Database

Modernize and simplify your data-centric applications with microservices architecture using the (..)

1 hr, 15 mins — 48778 Views

Get Started with Oracle Data Safe Fundamentals

AI Services: Introduction to OCI Vision

Autonomous Database 15 Minute Quick Start

谢谢

甲骨文中国

* 活动最终解释权归甲骨文公司所有

立即扫码进行 1V1 免费咨询

2023 年 10 月，MySQL 5.7 将终止官方支持和更新。

立刻升级至更快、更稳定、更安全的 MySQL 8.0 / MySQL Database Service，获取 300+ 项新特性，使开发更加灵活和高效，更好的满足业务发展需求。

免费咨询热线：

400-699-8888

ORACLE 甲骨文

Oracle数据库性能之并行那些事儿

数据库和云系列公益讲座

赵靖宇

- 资深解决方案工程师
- 超过10年的Oracle数据库管理经验
- 专注于数据库领域，主要负责银行、保险行业客户的数据库相关技术咨询和解决方案设计

内容简介

Oracle数据库中的并行处理是提高性能的重要手段之一，但需要根据具体情况进行配置和管理，以确保系统能够充分利用并行处理的优势同时避免潜在的性能问题

Zoom直播

直播时间：10月13日 11:00 - 12:00
扫描二维码进入直播
Zoom ID：957 9669 6723
密码：20212023

微信扫一扫预约

20-22

数据库和云讲座群

甲骨文云技术公众号

技术专家1V1深入交流


ORACLE