無料電子ブック

学習

Oracle Database

Free unaffiliated eBook created from
Stack Overflow contributors.

#oracle

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: oracle-database

It is an unofficial and free Oracle Database ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Oracle Database.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: Oracle Databaseの

Oracleは、70にLarry Ellison、Bob Miner、Ed Oatesによってにされたリレーショナル·データベースシステムRDBMSです。これはIBMのシステムRとがあることをしていました。

バージョン

バージョン	
バージョン1	1978-01-01
Oracle V2	1979-01-01
Oracleバージョン3	1983-01-01
Oracleバージョン4	1984-01-01
Oracleバージョン5	1985-01-01
Oracleバージョン6	1988-01-01
Oracle7	1992-01-01
Oracle8	1997-07-01
Oracle8i	1999-02-01
Oracle9i	2001-06-01
Oracle 10g	2003-01-01
Oracle 11g	2007-01-01
Oracle 12c	2013-01-01

Examples

こんにちは

```
SELECT 'Hello world!' FROM dual;
```

OracleのSQLでは、「デュアルはただのせの」です。 もともと JOINをしてを2にすることをしていましたが、 `DUMMY`が 'X'のが1つまれています。

こんにちはテーブルから

シンプルなをする

```
create table MY_table (
   what varchar2(10),
   who varchar2(10),
   mark varchar2(10)
);
```

をしますすべてのにをするは、ターゲットをできます

```
insert into my_table (what, who, mark) values ('Hello', 'world', '!' );
insert into my_table values ('Bye bye', 'ponies', '?' );
insert into my_table (what) values('Hey');
```

Oracleはトランザクションをするため、コミットする

```
commit;
```

あなたのデータをしてください

```
select what, who, mark from my_table where what='Hello';
```

SQLクエリ

にまれた5ドルをいでいるをリストアップしてください。、、をアルファベットにべます。

```
SELECT employee_name, date_of_birth, salary
FROM   employees
WHERE  salary > 50000
   AND date_of_birth >= DATE '2000-01-01'
ORDER BY employee_name;
```

なくとも5のをつののをします。にのをリストアップしてください。

```
SELECT department_id, COUNT(*)
FROM   employees
GROUP BY department_id
HAVING COUNT(*) >= 5
ORDER BY COUNT(*) DESC;
```

PL / SQLのHello World

```
/* PL/SQL is a core Oracle Database technology, allowing you to build clean, secure,
```

```
   optimized APIs to SQL and business logic. */

set serveroutput on

BEGIN
   DBMS_OUTPUT.PUT_LINE ('Hello World!');
END;
```

オンラインでOracle Databaseのをむ https://riptutorial.com/ja/oracle/topic/558/oracle-databaseの

2: DUALテーブル

DUALには、 VARCHAR2(1)とxつが1つだけされたDUMMYが1つあります。

DUALは、データベースのにSYSスキーマににされます。のスキーマからアクセスできます。

DUALテーブルはできません。

DUALテーブルをして、SQLからをびすことができます。 1つのしかたず、オラクル·オプティマイザがすべてをっているのでです。

Examples

のでは、のオペレーティングシステムのと

```
select sysdate from dual
```

のでは、start_valueとend_valueののをします。

```
select :start_value + level -1 n
from dual
connect by level <= :end_value  - :start_value + 1
```

オンラインでDUALテーブルをむ https://riptutorial.com/ja/oracle/topic/7328/dualテーブル

3: NULLの

き

そののデ―タになく、にがないはNULLです。このをしてNULLをNULLとしないでください。はUNKNOWNになるため、 `a = NULL`してください。わりに`a IS NULL`または`a IS NOT NULL`を`a IS NULL` `a IS NOT NULL`。 NULLはNULLとしくないnullがするのある2つのをするには、でするのいずれかをします。をくすべてのは、そのオペランドの1つがNULLのはNULLをします。たとえば、 `3 * NULL + 5`のはNULLです。

NULLは、PRIMARY KEYまたはNOT NULLによってされたにはされません。 はNOVALIDATEをつしいです

Examples

のデ―タのにNULLをめることができます

```
SELECT 1 NUM_COLUMN, 'foo' VARCHAR2_COLUMN from DUAL
UNION ALL
SELECT NULL, NULL from DUAL;
```

NUM_COLUMN	VARCHAR2_COLUMN
1	foo
ヌル	ヌル

がNULLです

```
SELECT 1 a, '' b from DUAL;
```

A	B
1	ヌル

NULLをむは、をいてNULLです。

```
SELECT 3 * NULL + 5, 'Hello ' || NULL || 'world'   from DUAL;
```

3 * NULL + 5	'HELLO' || NULL || 'WORLD'
ヌル	こんにちは

ヌルをきえるNVL

```
SELECT a column_with_null, NVL(a, 'N/A') column_without_null FROM
  (SELECT NULL a FROM DUAL);
```

COLUMN_WITH_NULL	COLUMN_WITHOUT_NULL
ヌル	N / A

NVLは、NULLをむことができる2つのをするのにです。

```
SELECT
    CASE WHEN a = b THEN 1 WHEN a <> b THEN 0 else -1 END comparison_without_nvl,
    CASE WHEN NVL(a, -1) = NVL(b, -1) THEN 1 WHEN NVL(a, -1) <> NVL(b, -1) THEN 0 else -1 END
comparison_with_nvl
  FROM
    (select null a, 3 b FROM DUAL
     UNION ALL
     SELECT NULL, NULL FROM DUAL);
```

COMPARISON_WITHOUT_NVL	COMPARISON_WITH_NVL
-1	0
-1	1

NVL2は、がnullのとはなるをる

のパラメータがNOT NULLの、NVL2は2のパラメータをします。それのは、3のがされます。

```
SELECT NVL2(null, 'Foo', 'Bar'), NVL2(5, 'Foo', 'Bar') FROM DUAL;
```

NVL2NULL、'FOO'、'BAR'	NVL25、'FOO'、'BAR'
バー	フー

のNULLをすCOALESCE

```
SELECT COALESCE(a, b, c, d, 5) FROM
    (SELECT NULL A, NULL b, NULL c, 4 d FROM DUAL);
```

COALESCEA、B、C、D、5
4

によっては、2つのパラメータをつCOALESCEをするが、2つのパラメータがでないにNVLをする

よりもになるがあります。 NVLはにのパラメータをします。 COALESCEは、したのNULLでします。つまり、のがNULLでない、COALESCEはよりになります。

オンラインでNULLのをむ https://riptutorial.com/ja/oracle/topic/8183/nullの

4: Oracle Database 12Cでのな

き

クエリをすると、テーブルののなにづいてデータをできます

Examples

Caluseでする

```
SELECT E.EMPLOYEE_ID,E.LAST_NAME,E.MANAGER_ID FROM HR.EMPLOYEES E
CONNECT BY PRIOR E.EMPLOYEE_ID = E.MANAGER_ID;
```

とマネージャのをするCONNECT BY。

トップダウンからのクエリのの

```
SELECT E.LAST_NAME|| ' reports to ' ||
PRIOR E.LAST_NAME "Walk Top Down"
FROM HR.EMPLOYEES E
START WITH E.MANAGER_ID IS NULL
CONNECT BY PRIOR E.EMPLOYEE_ID = E.MANAGER_ID;
```

オンラインでOracle Database 12Cでのなをむ https://riptutorial.com/ja/oracle/topic/8777/oracle-database-12cでのな

5: Oracle MAF

Examples

バインディングからをするには

```
        ValueExpression ve = AdfmfJavaUtilities.getValueExpression(<binding>, String.class);
       String <variable_name> = (String) ve.getValue(AdfmfJavaUtilities.getELContext());
```

ここで、「」とは、そのがられるべきELをす。

"variable_name"は、バインディングからのがされるパラメータです。

をバインドにするには

```
        ValueExpression ve = AdfmfJavaUtilities.getValueExpression(<binding>, String.class);
        ve.setValue(AdfmfJavaUtilities.getELContext(), <value>);
```

ここで、「」は、がされるELをす。

"value"はバインディングにするましいです

バインディングからメソッドをびすには

```
    AdfELContext adfELContext = AdfmfJavaUtilities.getAdfELContext();
    MethodExpression me;
    me = AdfmfJavaUtilities.getMethodExpression(<binding>, Object.class, new Class[] { });
    me.invoke(adfELContext, new Object[] { });
```

「バインディング」は、びされるメソッドをびすELをします。

javaScriptをびすには

```
AdfmfContainerUtilities.invokeContainerJavaScriptFunction(AdfmfJavaUtilities.getFeatureId(),
<function>, new Object[] {
                                                    });
```

"function"はびしたいjsです

オンラインでOracle MAFをむ https://riptutorial.com/ja/oracle/topic/6352/oracle-maf

- `dbms_aqadm.create_queue_table`されたにして、DDLまたはDMLをしないでください。dbms_aqadmおよびdbms_aqをしてこれらのテーブルをしてください。オラクルは、あなたがづいていないいくつかのサポート、などをすることがあります。にしてDDLまたはDMLをでると、Oracleサポートでをするためにおよびキューをおよびするがあるシナリオにつながるがあります。

- `dbms_aq.forever`をオプションとしてしないことをくおめします。これは、なキューをするために、なのワーカー・ジョブのスケジューリングをするがあるため、にがしていますOracle Doc ID 2001165.1を。

- バージョン10.1では、AQ_TM_PROCESSESパラメータをしないことをおめします。にキューをするためになQMONバックグラウンドジョブをにするので、これをゼロにしないでください。のコマンドをしてこのをOracleのデフォルトにリセットし、データベースをすることができます。 `alter system reset aq_tm_processes scope=spfile sid='*';`

Examples

シンプルプロデューサー/コンシューマー

メッセージをできるキューをします。オラクルはストアド・プロシージャにメッセージがエンキューされていることをし、するがあります。また、にできるサブプログラムをいくつかして、メッセージのデキューをやめ、デキューをして、なバッチジョブをしてすべてのメッセージをします。

これらのサンプルは、Oracle Database 12c Enterprise Editionリリース12.1.0.2.0 - 64bit Productionでテストされました。

キューをする

メッセージタイプ、メッセージをできるキューテーブル、およびキューをします。キューのメッセージは、にでデキューされ、にエンキューになります。メッセージのにがし、デキューがロールバックされると、AQはメッセージを3600にデキューにします。キューをするに、これを48します。

```
create type message_t as object
   (
   sender  varchar2 ( 50 ),
   message varchar2 ( 512 )
   );
/
-- Type MESSAGE_T compiled
begin dbms_aqadm.create_queue_table(
     queue_table        => 'MESSAGE_Q_TBL',
```

```
     queue_payload_type => 'MESSAGE_T',
     sort_list          => 'PRIORITY,ENQ_TIME',
     multiple_consumers =>  false,
     compatible         => '10.0.0');
  end;
/
-- PL/SQL procedure successfully completed.
begin dbms_aqadm.create_queue(
     queue_name          => 'MESSAGE_Q',
     queue_table         => 'MESSAGE_Q_TBL',
     queue_type          =>  0,
     max_retries         =>  48,
     retry_delay         =>  3600,
     dependency_tracking =>  false);
  end;
/
-- PL/SQL procedure successfully completed.
```

メッセージをくができたので、キューのメッセージをしてさせるパッケージをできます。

```
create or replace package message_worker_pkg
is
   queue_name_c constant varchar2(20) := 'MESSAGE_Q';

   -- allows the workers to process messages in the queue
   procedure enable_dequeue;

   -- prevents messages from being worked but will still allow them to be created and enqueued
   procedure disable_dequeue;

   -- called only by Oracle Advanced Queueing.  Do not call anywhere else.
   procedure on_message_enqueued (context        in raw,
                                  reginfo        in sys.aq$_reg_info,
                                  descr          in sys.aq$_descriptor,
                                  payload        in raw,
                                  payloadl       in number);

   -- allows messages to be worked if we missed the notification (or a retry
   -- is pending)
   procedure work_old_messages;

end;
/

create or replace package body message_worker_pkg
is
   -- raised by Oracle when we try to dequeue but no more messages are ready to
   -- be dequeued at this moment
   no_more_messages_ex          exception;
   pragma exception_init (no_more_messages_ex,
                          -25228);

   -- allows the workers to process messages in the queue
   procedure enable_dequeue
   as
   begin
      dbms_aqadm.start_queue (queue_name => queue_name_c, dequeue => true);
   end enable_dequeue;

   -- prevents messages from being worked but will still allow them to be created and enqueued
```

```
procedure disable_dequeue
as
begin
   dbms_aqadm.stop_queue (queue_name => queue_name_c, dequeue => true, enqueue => false);
end disable_dequeue;

procedure work_message (message_in in out nocopy message_t)
as
begin
   dbms_output.put_line ( message_in.sender || ' says ' || message_in.message );
end work_message;

-- called only by Oracle Advanced Queueing.  Do not call anywhere else.

procedure on_message_enqueued (context        in raw,
                               reginfo        in sys.aq$_reg_info,
                               descr          in sys.aq$_descriptor,
                               payload        in raw,
                               payloadl       in number)
as
   pragma autonomous_transaction;
   dequeue_options_l      dbms_aq.dequeue_options_t;
   message_id_l           raw (16);
   message_l              message_t;
   message_properties_l   dbms_aq.message_properties_t;
begin
   dequeue_options_l.msgid         := descr.msg_id;
   dequeue_options_l.consumer_name := descr.consumer_name;
   dequeue_options_l.wait          := dbms_aq.no_wait;
   dbms_aq.dequeue (queue_name           => descr.queue_name,
                    dequeue_options      => dequeue_options_l,
                    message_properties   => message_properties_l,
                    payload              => message_l,
                    msgid                => message_id_l);
   work_message (message_l);
   commit;
exception
   when no_more_messages_ex
   then
      -- it's possible work_old_messages already dequeued the message
      commit;
   when others
   then
      -- we don't need to have a raise here.  I just wanted to point out that
      -- since this will be called by AQ throwing the exception back to it
      -- will have it put the message back on the queue and retry later
      raise;
end on_message_enqueued;

-- allows messages to be worked if we missed the notification (or a retry
-- is pending)
procedure work_old_messages
as
   pragma autonomous_transaction;
   dequeue_options_l      dbms_aq.dequeue_options_t;
   message_id_l           raw (16);
   message_l              message_t;
   message_properties_l   dbms_aq.message_properties_t;
begin
   dequeue_options_l.wait       := dbms_aq.no_wait;
   dequeue_options_l.navigation := dbms_aq.first_message;
```

```
      while (true) loop -- way out is no_more_messages_ex
         dbms_aq.dequeue (queue_name           => queue_name_c,
                          dequeue_options      => dequeue_options_1,
                          message_properties   => message_properties_1,
                          payload              => message_1,
                          msgid                => message_id_1);
         work_message (message_1);
         commit;
      end loop;
   exception
      when no_more_messages_ex
      then
         null;
   end work_old_messages;
end;
```

に、メッセージがMESSAGE_Qにエンキューされコミットされたとき、たちのプロシージャにするときに、それがうがあることをAQにえます。 AQはこれをするためにのセッションでジョブをします。

```
begin
  dbms_aq.register (
     sys.aq$_reg_info_list (
        sys.aq$_reg_info (user || '.' || message_worker_pkg.queue_name_c,
                          dbms_aq.namespace_aq,
                          'plsql://' || user || '.message_worker_pkg.on_message_enqueued',
                          hextoraw ('FF'))),
     1);
  commit;
end;
```

キューをしてメッセージをする

```
declare
   enqueue_options_1      dbms_aq.enqueue_options_t;
   message_properties_1   dbms_aq.message_properties_t;
   message_id_1           raw (16);
   message_1              message_t;
begin
   -- only need to do this next line ONCE
   dbms_aqadm.start_queue (queue_name => message_worker_pkg.queue_name_c, enqueue => true ,
dequeue => true);

   message_1 := new message_t ( 'Jon', 'Hello, world!' );
   dbms_aq.enqueue (queue_name           => message_worker_pkg.queue_name_c,
                    enqueue_options      => enqueue_options_1,
                    message_properties   => message_properties_1,
                    payload              => message_1,
                    msgid                => message_id_1);
   commit;
end;
```

オンラインでOracleアドバンスト・キューイングAQをむ
https://riptutorial.com/ja/oracle/topic/4362/oracleアドバンスト-キューイング-aq-

7: WITHAKAテーブルをしたサブクエリファクタリング

サブクエリ·ファクタリングは、Oracle 11g R2でできます。

Examples

シンプルなジェネレータ

クエリ

```
WITH generator ( value ) AS (
  SELECT 1 FROM DUAL
UNION ALL
  SELECT value + 1
  FROM   generator
  WHERE  value < 10
)
SELECT value
FROM   generator;
```

```
VALUE
-----
    1
    2
    3
    4
    5
    6
    7
    8
    9
   10
```

りきの

サンプルデータ

```
CREATE TABLE table_name ( value VARCHAR2(50) );

INSERT INTO table_name ( value ) VALUES ( 'A,B,C,D,E' );
```

クエリ

```
WITH items ( list, item, lvl ) AS (
  SELECT value,
         REGEXP_SUBSTR( value, '[^,]+', 1, 1 ),
         1
```

```
  FROM   table_name
UNION ALL
  SELECT value,
         REGEXP_SUBSTR( value, '[^,]+', 1, lvl + 1 ),
         lvl + 1
  FROM   items
  WHERE  lvl < REGEXP_COUNT( value, '[^,]+' )
)
SELECT * FROM items;
```

```
LIST      ITEM LVL
--------- ---- ---
A,B,C,D,E    A   1
A,B,C,D,E    B   2
A,B,C,D,E    C   3
A,B,C,D,E    D   4
A,B,C,D,E    E   5
```

オンラインでWITHAKAテーブルをしたサブクエリファクタリングをむ
https://riptutorial.com/ja/oracle/topic/3506/with-akaテーブル-をしたサブクエリファクタリング

8: インデックス

き

ここでは、をしてなるインデックスをします。インデックスのクエリパフォーマンスの、インデックスのDMLパフォーマンスのなど

Examples

Bツリーインデックス

```
CREATE INDEX ord_customer_ix ON orders (customer_id);
```

デフォルトでは、もしなければ、oracleはb-treeとしてをします。しかし、いつうべきかをっておくがあります。 Bツリーインデックスは、バイナリツリーでデータをします。たちがっているように、indexは、インデックスきカラムのにしてらかのエントリをするスキーマ·オブジェクトです。したがって、これらのでがわれるたびに、そのレコードのなをでチェックして、にアクセスします。インデックスにするいくつかの

- インデックスのエントリをするには、らかのバイナリアルゴリズムをします。
- データのカーディナリティがいは、b-treeインデックスがするのにです。
- はDMLをくします。レコードとに、きのには1つのエントリがです。
- したがって、でないは、をしないでください。

ビットマップ

```
CREATE BITMAP INDEX
emp_bitmap_idx
ON index_demo (gender);
```

- ビットマップは、 データ·カーディナリティがいにされます。
- ここで、 **Gender**はがいというがあります。は、、などです。
- ですから、にこの3つののバイナリツリーをすると、なトラバースがします。
- ビットマップでは、けされるのにして1つのをつ2がされます。は、ビットマップののをします。この2は、インデックスのにテーブルのをけたをします。
- のに、Oracleはビットマップをデータ·バッファにして、するがすばやくスキャンされるようにします。これらのするは、Row-IDリストのでOracleにされ、これらのRow-IDはなにアクセスできます。

ベースインデックス

```
CREATE INDEX first_name_idx ON user_data (UPPER(first_name));
```

```
SELECT *
FROM   user_data
WHERE  UPPER(first_name) = 'JOHN2';
```

- ベースのインデックスは、についてインデックスをすることをします。
- whereでは、にがされる、そのについてをするがいでしょう。
- ここでは、このでは、のためにUpperがされています。したがって、をしてインデックスをするがよいでしょう。

オンラインでインデックスをむ https://riptutorial.com/ja/oracle/topic/9978/インデックス

9: ウィンドウ

- Ratio_To_ReportexprOVERquery_partition_clause

Examples

Ratio_To_Report

ウィンドウのすべてのにするののののをします。

```
--Data
CREATE TABLE Employees (Name Varchar2(30), Salary Number(10));
INSERT INTO Employees Values ('Bob',2500);
INSERT INTO Employees Values ('Alice',3500);
INSERT INTO Employees Values ('Tom',2700);
INSERT INTO Employees Values ('Sue',2000);
--Query
SELECT Name, Salary, Ratio_To_Report(Salary) OVER () As Ratio
FROM Employees
ORDER BY Salary, Name, Ratio;
--Output
NAME                           SALARY      RATIO
------------------------------ ---------- ----------
Sue                                  2000 .186915888
Bob                                  2500  .23364486
Tom                                  2700 .252336449
Alice                                3500 .327102804
```

オンラインでウィンドウをむ https://riptutorial.com/ja/oracle/topic/6669/ウィンドウ

10: エラーログ

Examples

データベースへのきみのエラー·ロギング

のEXAMPLEのOracleエラー·ログERR $ _EXAMPLEをします。

```
EXECUTE DBMS_ERRLOG.CREATE_ERROR_LOG('EXAMPLE', NULL, NULL, NULL, TRUE);
```

SQLできみをう

```
insert into EXAMPLE (COL1) values ('example')
LOG ERRORS INTO ERR$_EXAMPLE reject limit unlimited;
```

オンラインでエラーログをむ https://riptutorial.com/ja/oracle/topic/3505/エラーログ

11: キーワードまたはのり

Examples

をしてまたはをります

firm's_addressから*をします。

* firm's_addressから*をしてください。

であるまたはのをる

あなたがテーブルというのテーブルをっているとします。あるいは、テーブルをキーワードとするテーブルをしたいは、テーブルを "table"

テーブルから*をします。のクエリはエラーでします。のクエリはにされます。

"table"から*をしてください。

オンラインでキーワードまたはのりをむ https://riptutorial.com/ja/oracle/topic/6553/キーワードまたはのり

12: クエリによってされるのページネーション

Examples

でのNをする

FETCHは、Oracle 12c R1でされました。

```
SELECT   val
FROM     mytable
ORDER BY val DESC
FETCH FIRST 5 ROWS ONLY;
```

のバージョンでもするFETCHをしない

```
SELECT * FROM (
   SELECT   val
   FROM     mytable
   ORDER BY val DESC
) WHERE ROWNUM <= 5;
```

SQLでのページネーション

```
SELECT val
FROM   (SELECT val, rownum AS rnum
        FROM   (SELECT val
                FROM   rownum_order_test
                ORDER BY val)
        WHERE rownum <= :upper_limit)
WHERE  rnum >= :lower_limit ;
```

このようにして、Webデータページとにテーブルデータをページすることができます

テーブルからNのレコードをする

rownumをってからのをすることができます

```
select * from
(
  select val from  mytable
) where rownum<=5
```

またはのレコードがなは、についてをすクエリのorder byがです。

の5つのレコード

```
select * from
```

```
(
    select val from  mytable order by val desc
) where rownum<=5
```

の**5**つのレコード

```
select * from
(
    select val from  mytable order by val
) where rownum<=5
```

くのから**NM**をする**Oracle 12c**より

row_numberをします。

```
with t as (
  select col1
  , col2
  , row_number() over (order by col1, col2) rn
  from table
)
select col1
, col2
from t
where rn between N and M; -- N and M are both inclusive
```

Oracle 12cでは、これを`OFFSET`と`FETCH`よりにします。

いくつかのをスキップしてから

Oracle 12g +では、

```
SELECT Id, Col1
FROM TableName
ORDER BY Id
OFFSET 20 ROWS FETCH NEXT 20 ROWS ONLY;
```

のバージョン

```
SELECT Id,
   Col1
 FROM (SELECT Id,
           Col1,
           row_number() over (order by Id) RowNumber
      FROM TableName)
WHERE RowNumber BETWEEN 21 AND 40
```

からいくつかのをスキップする

Oracle 12g +では、

```
SELECT Id, Col1
FROM TableName
ORDER BY Id
OFFSET 5 ROWS;
```

のバージョン

```
SELECT Id,
   Col1
 FROM (SELECT Id,
           Col1,
           row_number() over (order by Id) RowNumber
      FROM TableName)
WHERE RowNumber > 20
```

オンラインでクエリによってされるのページネーションをむ
https://riptutorial.com/ja/oracle/topic/4300/クエリによってされるの-ページネーション-

13: コンテキストの

- CREATE [OR REPLACE] CONTEXTUSING [schema。]パッケージ。
- CREATE [OR REPLACE] CONTEXTUSING [スキーマ。]パッケージINITIALIZED EXTERNALLY;
- CREATE [OR REPLACE] CONTEXTUSING [スキーマ。]パッケージINITIALIZED GLOBALLY;
- CREATE [OR REPLACE] CONTEXTUSING [スキーマ。]パッケージACCESSED GLOBALLY;

パラ メーター

パラ メータ	
OR REPLACE	のコンテキストをする
	コンテキストの - これはSYS_CONTEXTへのびしのです
スキーマ	パッケージの
パッケージ	コンテキストをまたはリセットするデータベースパッケージ。コンテキストをするためにデータベースパッケージがするはありません。
INITIALIZED	コンテキストをできるOracle Databaseのエンティティをします。
EXTERNALLY	OCIインタフェースでコンテキストをできるようにします。
GLOBALLY	セッションをするときにLDAPディレクトリがコンテキストをできるようにします。
ACCESSED GLOBALLY	インスタンスでコンテキストにアクセスできるようにします。じクライアントIDをつり、のセッションでをできます。

Oracleのマニュアル12cR1 http ://docs.oracle.com/database/121/SQLRF/statements_5003.htm

Examples

コンテキストをする

```
CREATE CONTEXT my_ctx USING my_pkg;
```

これにより、データベースパッケージmy_pkgのルーチンによってのみできるコンテキストがされます。たとえば、のようになります。

```
CREATE PACKAGE my_pkg AS
  PROCEDURE set_ctx;
END my_pkg;

CREATE PACKAGE BODY my_pkg AS
  PROCEDURE set_ctx IS
  BEGIN
    DBMS_SESSION.set_context('MY_CTX','THE KEY','Value');
    DBMS_SESSION.set_context('MY_CTX','ANOTHER','Bla');
  END set_ctx;
END my_pkg;
```

、セッションがこれをう

```
my_pkg.set_ctx;
```

これでキーのをできるようになりました

```
SELECT SYS_CONTEXT('MY_CTX','THE KEY') FROM dual;

Value
```

オンラインでコンテキストのをむ https://riptutorial.com/ja/oracle/topic/2088/コンテキストの

14: シーケンス

- CREATE SEQUENCE SCHEMA.SEQUENCE {INCREMENT BY INTEGER | INTEGERをして| MAXVALUE INTEGER | NOMAXVALUE INTEGER | MINVALUE INTEGER | NOMINVALUE INTEGER | CYCLE INTEGER | NOCYCLE INTEGER |キャッシュ| NOCACHE |オーダー| NOODER}

パラメーター

パラメータ	
スキーマ	スキーマ
インクリメントする	の
りに	のがです
マックスバリュー	シーケンスの
ノーマックスバリュー	はデフォルトです
	シーケンスの
	がデフォルトになります
サイクル	このにしたににリセットする
ノーサイクル	デフォルト
キャッシュ	りて
ノーキャッシュ	デフォルト
	のをする
いいえ	デフォルト

Examples

シーケンスの

CREATE SEQUENCEステートメントをして、のユーザーがのをするデータベース·オブジェクトであるシーケンスをします。シーケンスをしてキーをにすることができます。

シーケンスがされると、コミットまたはロールバックされたトランザクションとはして、シーケンスがインクリメントされます。 2のユーザがじシーケンスをにインクリメントすると、シーケンスがのユーザによってされているため、ユーザがするシーケンスにギャップがじることがあります。 1のユーザは、のユーザによってされたシーケンスをしてることができない。あるユーザーによってシーケンスがされた、そのユーザーは、シーケンスがのユーザーによってされているかどうかにかかわらず、そのにききアクセスできます。

シーケンスはテーブルとはにされるため、1つまたはのテーブルにしてじシーケンスをできます。々のシーケンスは、にロールバックされるトランザクションでされてされるため、スキップされているようにえるがあります。さらに、のユーザは、のユーザがじシーケンスからしていることをしないことがある。

シーケンスがされた、シーケンスののをするCURRVALまたはシーケンスをインクリメントしてしいをするNEXTVALをして、SQLステートメントのそのにアクセスできます。

のスキーマにシーケンスをするには、CREATE SEQUENCEシステムがです。

のユーザーのスキーマにシーケンスをするには、CREATE ANY SEQUENCEシステムがです。

ののは、サンプル・スキーマoeにcustomers_seqをします。このシーケンスは、がcustomersにされたときにIDをするためにできます。

```
CREATE SEQUENCE customers_seq
START WITH     1000
INCREMENT BY   1
NOCACHE
NOCYCLE;
```

customers_seq.nextvalへののは1000をします。2のは1001をします。そののは、のよりもきな1をします。

オンラインでシーケンスをむ https://riptutorial.com/ja/oracle/topic/3709/シーケンス

15: ジョイン

Examples

クロスジョイン

`CROSS JOIN`は、なをしない2つののの`CROSS JOIN`し、2つののデカルトになります。デカルトは、1つののがの2ののとされることをします。たとえば、 `TABLEA`に20、 `TABLEB`に20がある、は`20*20 = 400`になります。

```
SELECT *
FROM TABLEA CROSS JOIN TABLEB;
```

これはのようにくこともできます

```
SELECT *
FROM TABLEA, TABLEB;
```

に、2つのテーブルのSQLでのクロスジョインのをします。

サンプルテーブル TABLEA

```
+-------+---------+
| VALUE |   NAME  |
+-------+---------+
|   1   |   ONE   |
|   2   |   TWO   |
+-------+---------+
```

サンプルテーブル TABLEB

```
+-------+--------+
| VALUE |  NAME  |
+-------+--------+
|   3   |  THREE |
|   4   |  FOUR  |
+-------+--------+
```

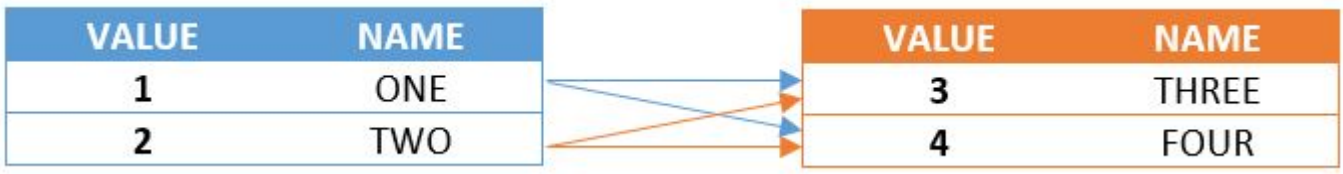
さて、クエリをすると

```
SELECT *
FROM TABLEA CROSS JOIN TABLEB;
```

```
+-------+--------+-------+--------+
| VALUE |  NAME  | VALUE |  NAME  |
+-------+--------+-------+--------+
|   1   |   ONE  |   3   |  THREE |
|   1   |   ONE  |   4   |  FOUR  |
```

```
|   2   |   TWO  |   3   |  THREE |
|   2   |   TWO  |   4   |  FOUR  |
+-------+--------+-------+--------+
```

これは、2つのテーブルでクロスがわれるです。



クロス·ジョインの Oracleのドキュメント

INNER JOIN

INNER JOINは、なをできるJOINです。

TableExpression [INNER] JOINテーブル{ON booleanExpression | USING}

ブールでONをして、をすることができます。

ONののには、のと、のSELECTにするせブロックのがまれます。のでは、ONはのテーブルをします。

```
-- Join the EMP_ACT and EMPLOYEE tables
-- select all the columns from the EMP_ACT table and
-- add the employee's surname (LASTNAME) from the EMPLOYEE table
-- to each row of the result
SELECT SAMP.EMP_ACT.*, LASTNAME
 FROM SAMP.EMP_ACT JOIN SAMP.EMPLOYEE
 ON EMP_ACT.EMPNO = EMPLOYEE.EMPNO
-- Join the EMPLOYEE and DEPARTMENT tables,
-- select the employee number (EMPNO),
-- employee surname (LASTNAME),
-- department number (WORKDEPT in the EMPLOYEE table and DEPTNO in the
-- DEPARTMENT table)
-- and department name (DEPTNAME)
-- of all employees who were born (BIRTHDATE) earlier than 1930.
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
 FROM SAMP.EMPLOYEE JOIN SAMP.DEPARTMENT
 ON WORKDEPT = DEPTNO
 AND YEAR(BIRTHDATE) < 1930

-- Another example of "generating" new data values,
-- using a query which selects from a VALUES clause (which is an
-- alternate form of a fullselect).
-- This query shows how a table can be derived called "X"
-- having 2 columns "R1" and "R2" and 1 row of data
SELECT *
FROM (VALUES (3, 4), (1, 5), (2, 6))
AS VALUESTABLE1(C1, C2)
JOIN (VALUES (3, 2), (1, 2),
(0, 3)) AS VALUESTABLE2(c1, c2)
ON VALUESTABLE1.c1 = VALUESTABLE2.c1
```

```
-- This results in:
-- C1          |C2          |C1          |2
-- -----------------------------------------------
-- 3           |4           |3           |2
-- 1           |5           |1           |2


-- List every department with the employee number and
-- last name of the manager

SELECT DEPTNO, DEPTNAME, EMPNO, LASTNAME
FROM DEPARTMENT INNER JOIN EMPLOYEE
ON MGRNO = EMPNO

-- List every employee number and last name
-- with the employee number and last name of their manager
SELECT E.EMPNO, E.LASTNAME, M.EMPNO, M.LASTNAME
FROM EMPLOYEE E INNER JOIN
DEPARTMENT INNER JOIN EMPLOYEE M
    ON MGRNO = M.EMPNO
    ON E.WORKDEPT = DEPTNO
```

LEFT OUTER JOIN

`LEFT OUTER JOIN`は、なをとするが、しないをのからしない2つののLEFT OUTER JOINします。

```
SELECT
      ENAME,
      DNAME,
      EMP.DEPTNO,
      DEPT.DEPTNO
FROM
      SCOTT.EMP LEFT OUTER JOIN SCOTT.DEPT
      ON EMP.DEPTNO = DEPT.DEPTNO;
```

ANSIがされるであっても、ににのにするがあります。あるで(+)をすると、のどちらがとみなされるかがまります。

```
SELECT
      ENAME,
      DNAME,
      EMP.DEPTNO,
      DEPT.DEPTNO
FROM
      SCOTT.EMP,
      SCOTT.DEPT
WHERE
      EMP.DEPTNO = DEPT.DEPTNO(+);
```

に、2つのテーブルののをします。

サンプルテーブル EMPLOYEE

```
+-----------+---------+
|    NAME   |  DEPTNO |
```

```
+-----------+---------+
|     A     |    2    |
|     B     |    1    |
|     C     |    3    |
|     D     |    2    |
|     E     |    1    |
|     F     |    1    |
|     G     |    4    |
|     H     |    4    |
+-----------+---------+
```

サンプルテーブル DEPT

```
+---------+--------------+
|  DEPTNO |   DEPTNAME   |
+---------+--------------+
|    1    |  ACCOUNTING  |
|    2    |    FINANCE   |
|    5    |   MARKETING  |
|    6    |      HR      |
+---------+--------------+
```

さて、クエリをすると

```
SELECT
      *
FROM
      EMPLOYEE LEFT OUTER JOIN DEPT
      ON EMPLOYEE.DEPTNO = DEPT.DEPTNO;
```

```
+-----------+---------+---------+--------------+
|    NAME   |  DEPTNO |  DEPTNO |   DEPTNAME   |
+-----------+---------+---------+--------------+
|     F     |    1    |    1    |  ACCOUNTING  |
|     E     |    1    |    1    |  ACCOUNTING  |
|     B     |    1    |    1    |  ACCOUNTING  |
|     D     |    2    |    2    |    FINANCE   |
|     A     |    2    |    2    |    FINANCE   |
|     C     |    3    |         |              |
|     H     |    4    |         |              |
|     G     |    4    |         |              |
+-----------+---------+---------+--------------+
```

のジョイン

RIGHT OUTER JOINは、なをとするが、2のからしないをしない2つののRIGHT OUTER JOINします。

```
SELECT
      ENAME,
      DNAME,
      EMP.DEPTNO,
      DEPT.DEPTNO
FROM
      SCOTT.EMP RIGHT OUTER JOIN SCOTT.DEPT
      ON EMP.DEPTNO = DEPT.DEPTNO;
```

SCOTT.DEPTしないはまれていますが、 SCOTT.EMPのSCOTT.DEPTしないはまれていないため、はLEFT OUTER JOINをするのとじです。

```
SELECT
      ENAME,
      DNAME,
      EMP.DEPTNO,
      DEPT.DEPTNO
FROM
      SCOTT.DEPT RIGHT OUTER JOIN SCOTT.EMP
      ON DEPT.DEPTNO = EMP.DEPTNO;
```

に、2つのテーブルののをします。

サンプルテーブル EMPLOYEE

```
+-----------+---------+
|    NAME   |  DEPTNO |
+-----------+---------+
|     A     |    2    |
|     B     |    1    |
|     C     |    3    |
|     D     |    2    |
|     E     |    1    |
|     F     |    1    |
|     G     |    4    |
|     H     |    4    |
+-----------+---------+
```

サンプルテーブル DEPT

```
+---------+--------------+
|  DEPTNO |   DEPTNAME   |
+---------+--------------+
|    1    |  ACCOUNTING  |
|    2    |    FINANCE   |
|    5    |   MARKETING  |
|    6    |      HR      |
+---------+--------------+
```

さて、クエリをすると

```
SELECT
      *
FROM
      EMPLOYEE RIGHT OUTER JOIN DEPT
      ON EMPLOYEE.DEPTNO = DEPT.DEPTNO;
```

```
+-----------+---------+---------+--------------+
|    NAME   |  DEPTNO |  DEPTNO |   DEPTNAME   |
+-----------+---------+---------+--------------+
|     A     |    2    |    2    |    FINANCE   |
|     B     |    1    |    1    |  ACCOUNTING  |
|     D     |    2    |    2    |    FINANCE   |
```

```
|     E     |    1    |    1    |  ACCOUNTING  |
|     F     |    1    |    1    |  ACCOUNTING  |
|           |         |    5    |   MARKETING  |
|           |         |    6    |      HR      |
+-----------+---------+---------+--------------+
```

クエリにするOracle+はのとおりです。

```
SELECT *
FROM EMPLOYEE, DEPT
WHERE EMPLOYEE.DEPTNO(+) = DEPT.DEPTNO;
```

フル・アウター・ジョイン

`FULL OUTER JOIN`は、なをとするが、どちらのでもしないをしない2つののFULL OUTER JOINします。つまり、テーブルのすべてのをします。

```
SELECT
      *
FROM
      EMPLOYEE FULL OUTER JOIN DEPT
      ON EMPLOYEE.DEPTNO = DEPT.DEPTNO;
```

に、2つのテーブルののをします。

サンプルテーブル EMPLOYEE

```
+-----------+---------+
|    NAME   |  DEPTNO |
+-----------+---------+
|     A     |    2    |
|     B     |    1    |
|     C     |    3    |
|     D     |    2    |
|     E     |    1    |
|     F     |    1    |
|     G     |    4    |
|     H     |    4    |
+-----------+---------+
```

サンプルテーブル DEPT

```
+---------+--------------+
|  DEPTNO |   DEPTNAME   |
+---------+--------------+
|    1    |  ACCOUNTING  |
|    2    |    FINANCE   |
|    5    |   MARKETING  |
|    6    |      HR      |
+---------+--------------+
```

さて、クエリをすると

```
SELECT
      *
FROM
      EMPLOYEE FULL OUTER JOIN DEPT
      ON EMPLOYEE.DEPTNO = DEPT.DEPTNO;
```

```
+-----------+---------+---------+--------------+
|    NAME   |  DEPTNO |  DEPTNO |   DEPTNAME   |
+-----------+---------+---------+--------------+
|     A     |    2    |    2    |    FINANCE   |
|     B     |    1    |    1    |  ACCOUNTING  |
|     C     |    3    |         |              |
|     D     |    2    |    2    |    FINANCE   |
|     E     |    1    |    1    |  ACCOUNTING  |
|     F     |    1    |    1    |  ACCOUNTING  |
|     G     |    4    |         |              |
|     H     |    4    |         |              |
|           |         |    6    |      HR      |
|           |         |    5    |   MARKETING  |
+-----------+---------+---------+--------------+
```

ここでは、しないはNULLになっています。

アンティン

antijoinは、のからするがないのからをします。のサブクエリとしないNOT INをします。

```
SELECT * FROM employees
   WHERE department_id NOT IN
   (SELECT department_id FROM departments
       WHERE location_id = 1700)
   ORDER BY last_name;
```

に、2つのテーブルのアンチのをします。

サンプルテーブル EMPLOYEE

```
+-----------+---------+
|    NAME   |  DEPTNO |
+-----------+---------+
|     A     |    2    |
|     B     |    1    |
|     C     |    3    |
|     D     |    2    |
|     E     |    1    |
|     F     |    1    |
|     G     |    4    |
|     H     |    4    |
+-----------+---------+
```

サンプルテーブル DEPT

```
+---------+--------------+
|  DEPTNO |   DEPTNAME   |
```

```
+---------+--------------+
|    1    |  ACCOUNTING  |
|    2    |    FINANCE   |
|    5    |   MARKETING  |
|    6    |      HR      |
+---------+--------------+
```

さて、クエリをすると

```
SELECT
      *
FROM
      EMPLOYEE WHERE DEPTNO NOT IN
      (SELECT DEPTNO FROM DEPT);
```

```
+-----------+---------+
|    NAME   |  DEPTNO |
+-----------+---------+
|     C     |    3    |
|     H     |    4    |
|     G     |    4    |
+-----------+---------+
```

には、DEPTにDEPTNOがしなかったEMPLOYEEののみがされます。

SEMIJOIN

えば、Semijoinクエリをして、が2500をえるがなくとも1いるすべてのをつけることができます。

```
SELECT * FROM departments
   WHERE EXISTS
   (SELECT 1 FROM employees
       WHERE departments.department_id = employees.department_id
       AND employees.salary > 2500)
   ORDER BY department_name;
```

これは、のとしてが2500をえるがあることをするwhereをえることで、じをもすことができるため、なのよりもです。に3000の$_n$のがいれば、`select * from departments, employees`し、idsにな`select * from departments, employees`とwhereでを$_n$すとします。

ジョイン

`JOIN`は、2つののの`JOIN`をします。ののしないはきます。 Oracle 9iから、 `JOIN`は`INNER JOIN`にです。このには、 `CROSS JOIN`および`NATURAL JOIN`とはに、ながです。

```
select t1.*,
       t2.DeptId
  from table_1 t1
  join table_2 t2 on t2.DeptNo = t1.DeptNo
```

Oracleのマニュアル

- 10g
- 11g
- 12g

NATURAL JOIN

NATURAL JOINでは、はありません。されたテーブルにじのすべてのフィールドにづいて1つをします。

```
create table tab1(id number,  descr varchar2(100));
create table tab2(id number,  descr varchar2(100));
insert into tab1 values(1, 'one');
insert into tab1 values(2, 'two');
insert into tab1 values(3, 'three');
insert into tab2 values(1, 'ONE');
insert into tab2 values(3, 'three');
```

はのテーブルにのフィールドIDとDESCRでわれます。

```
SQL> select *
  2  from tab1
  3        natural join
  4       tab2;

        ID DESCR
---------- ----------
         3 three
```

なるのは、JOINではされません。

```
SQL> select *
  2  from (select id as id, descr as descr1 from tab1)
  3       natural join
  4       (select id as id, descr as descr2 from tab2);

        ID DESCR1     DESCR2
---------- ---------- ----------
         1 one        ONE
         3 three      three
```

されたにのがない、のないJOINがされます。

```
SQL> select *
  2  from (select id as id1, descr as descr1 from tab1)
  3       natural join
  4       (select id as id2, descr as descr2 from tab2);

       ID1 DESCR1            ID2 DESCR2
---------- ---------- ---------- ----------
         1 one                 1 ONE
         2 two                 1 ONE
```

```
3 three                1 ONE
1 one                  3 three
2 two                  3 three
3 three                3 three
```

オンラインでジョインをむ https://riptutorial.com/ja/oracle/topic/4192/ジョイン

16: データベースリンク

Examples

データベースリンクの

```
CREATE DATABASE LINK dblink_name
CONNECT TO remote_username
IDENTIFIED BY remote_password
USING 'tns_service_name';
```

リモートDBは、のでアクセスできます。

```
SELECT * FROM MY_TABLE@dblink_name;
```

リンクされたデータベースのオブジェクトをらなくてもデータベースリンクのをテストするには、のクエリをします。

```
SELECT * FROM DUAL@dblink_name;
```

リンクされたデータベースサービスのドメインをにするには、ドメインが`USING`ステートメントにされます。えば

```
USING 'tns_service_name.WORLD'
```

ドメインがにされていない、Oracleはリンクがされているデータベースのドメインをします。

データベース・リンクのためのOracleのマニュアル

- 10g https : //docs.oracle.com/cd/B19306_01/server.102/b14200/statements_5005.htm
- 11g https : //docs.oracle.com/cd/B28359_01/server.111/b28310/ds_concepts002.htm
- 12g https : //docs.oracle.com/database/121/SQLRF/statements_5006.htm#SQLRF01205

データベースリンクの

「ORA1」と「ORA2」という2つのデータベースがあるとします。データベースリンクをして、データベース "ORA1"から "ORA2"のオブジェクトにアクセスできます。

データベース・リンクをするには、 `CREATE DATABASE LINK`がです。プライベートデータベースリンクをするには、 `CREATE PUBLIC DATABASE LINK`がです。

* Oracle Netはのインスタンスにするがあります。

データベースリンクをする

ORA1から

```
SQL> create <public> database link ora2 connect to user1 identified by pass1 using <tns name
of ora2>;
```

データベースリンクがされました。

DBリンクがされたので、ORA1からのコマンドをすることでできます。

```
SQL> Select name from V$DATABASE@ORA2; -- should return ORA2
```

ユーザ`user1`がORA2のオブジェクトのTABLE1などにして`SELECT`をっている、 "ORA1"から "ORA2"のDBオブジェクトにアクセスすることもできます。

```
  SELECT COUNT(*) FROM TABLE1@ORA2;
```

- のデータベースがしていなければなりませんかれている。
- のデータベース·リスナーがしているがあります。
- TNSをしくするがあります。
- ORA2データベースにユーザーuser1がし、パスワードをしてするがあります。
- ユーザーuser1には、なくとも`SELECT`がです。ORA2のオブジェクトにアクセスするには、ユーザーがです。

オンラインでデータベースリンクをむ https://riptutorial.com/ja/oracle/topic/3859/データベースリンク

17: データポンプ

き

に、データポンプのインポート/エクスポートをするをします。

Examples

Datapumpジョブをする

データポンプジョブは、

1.データビュー

```
   select * from dba_datapump_jobs;
   SELECT * FROM DBA_DATAPUMP_SESSIONS;
   select username,opname,target_desc,sofar,totalwork,message from V$SESSION_LONGOPS where
username = 'bkpadmin';
```

2.データポンプの

- インポート/エクスポート·ログまたはデータ·ディクショナリからジョブをメモし、
- する**attach**コマンド
- インポート/エクスポートプロンプトでのステータスの

```
impdp <bkpadmin>/<bkp123> attach=<SYS_IMPORT_SCHEMA_01>
Import> status
```

Ctrl / Cをして、インポート/エクスポートプロンプトからけす

ステップ3/6ディレクトリをする

```
create or replace directory DATAPUMP_REMOTE_DIR as '/oracle/scripts/expimp';
```

7エクスポートコマンド

コマンド

```
expdp <bkpadmin>/<bkp123>  parfile=<exp.par>
```

ごのにじて<>のデータをなにきえてください。にじてパラメータを/できます。のでは、りのすべてのパラメータがにすようにパラメータファイルにされています。

- エクスポートタイプ ユーザーエクスポート

- スキーマをエクスポートする
- パラメータファイルの[exp.par]

```
schemas=<schema>
directory= DATAPUMP_REMOTE_DIR
dumpfile=<dbname>_<schema>.dmp
logfile=exp_<dbname>_<schema>.log
```

- エクスポートの きなスキーマのユーザーエクスポート
- なデータセットのスキーマをエクスポートするここでエクスポートダンプファイルはされ、されます。ここではがされていますをすると、サーバーの*CPU*がします
- パラメータファイルの[exp.par]

```
schemas=<schema>
directory= DATAPUMP_REMOTE_DIR
dumpfile=<dbname>_<schema>_%U.dmp
logfile=exp_<dbname>_<schema>.log
compression = all
parallel=5
```

- エクスポートタイプ テーブルエクスポート [テーブルのエクスポートセット]
- パラメータファイルの[exp.par]

```
tables= tname1, tname2, tname3
directory= DATAPUMP_REMOTE_DIR
dumpfile=<dbname>_<schema>.dmp
logfile=exp_<dbname>_<schema>.log
```

---

9 インポートコマンド

- ユーザーのインポートのに、インポートされたスキーマまたはをドロップすることをおめします。

コマンド

```
impdp <bkpadmin>/<bkp123>  parfile=<imp.par>
```

ごのにじて<>のデータをなにきえてください。にじてパラメータを/できます。のでは、りのすべてのパラメータがにすようにパラメータファイルにされています。

- インポートの ユーザーの インポート
- スキーマをインポートする
- パラメータファイルの[say imp.par]

```
schemas=<schema>
directory= DATAPUMP_REMOTE_DIR
```

```
dumpfile=<dbname>_<schema>.dmp
logfile=imp_<dbname>_<schema>.log
```

- インポートの きなスキーマのユーザーインポート
- なデータセットのスキーマをインポートするがここでされますをすると、サーバーの*CPU*がします
- パラメータファイルの[say imp.par]

```
schemas=<schema>
directory= DATAPUMP_REMOTE_DIR
dumpfile=<dbname>_<schema>_%U.dmp
logfile=imp_<dbname>_<schema>.log
parallel=5
```

- インポートの テーブルの インポート [テーブルの インポート]
- パラメータファイルの[say imp.par]

```
tables= tname1, tname2, tname3
directory= DATAPUMP_REMOTE_DIR
dumpfile=<dbname>_<schema>.dmp
logfile=exp_<dbname>_<schema>.log
TABLE_EXISTS_ACTION= <APPEND /SKIP /TRUNCATE /REPLACE>
```

1.データポンプステップ

ソース サーバー[データのエクスポート]	ターゲット サーバ[データの インポート]
1.エクスポートダンプファイルをするデータポンプフォルダをします	4.インポートダンプファイルをするデータポンプフォルダをします
2.エクスポートをするデータベース･スキーマにログイン します。	5.インポートをするデータベース･スキーマにログイン します。
3.1をすディレクトリをします。	6.4をすディレクトリをします。
7.エクスポートステートメントをします。	
8.ダンプファイルをターゲット サーバにコピー/ SCPします。	
	9. Importをする
	10.データをチェックし、なオブジェクトをコンパイル し、するをする

なるスキーマとスペースでをコピーする

```
 expdp <bkpadmin>/<bkp123> directory=DATAPUMP_REMOTE_DIR dumpfile=<customer.dmp>


 impdp <bkpadmin>/<bkp123> directory=DATAPUMP_REMOTE_DIR dumpfile=<customer.dmp>
remap_schema=<source schema>:<target schema> remap_tablespace=<source tablespace>:<target
tablespace>
```

オンラインでデータポンプをむ https://riptutorial.com/ja/oracle/topic/9391/データポンプ

18: データ

カタログビューともばれるデータディクショナリビューでは、データベースのをリアルタイムでできます。

ビューUSER_ 、 ALL_ 、およびDBA_ 、あなたがしているスキーマ·オブジェクトにするをしUSER_ あなたがアクセスできる、 ALL_ 、またはSYSDBAをつユーザーがアクセスできるDBA_ 。 たとえば、ビューALL_TABLESは、をつすべてのがされます。

V$ビューには、パフォーマンスのがされます。

_PRIVSビューには、ユーザー、ロール、およびオブジェクトのさまざまなみわせにするがされます。

Oracleのドキュメントカタログ·ビュー/データ·ディクショナリ·ビュー

Examples

されたオブジェクトのテキストソース

USER_SOURCEは、のユーザーがするストアド·オブジェクトのテキスト·ソースをします。このビューには、 OWNERはされません。

```
select * from user_source where type='TRIGGER' and lower(text) like '%order%'
```

ALL_SOURCEは、のユーザーがアクセスできるストアド·オブジェクトのテキスト·ソースをします。

```
select * from all_source where owner=:owner
```

DBA_SOURCEは、データベースにされているすべてのオブジェクトのテキスト·ソースをします。

```
select * from dba_source
```

Oracleのすべてのテーブルのリストをする

```
select owner, table_name
from all_tables
```

ALL_TAB_COLUMNSは、ユーザーがアクセスな、ビューおよびクラスタのをします。 COLSはUSER_TAB_COLUMNSです。

```
select *
from all_tab_columns
where table_name = :tname
```

ユーザーにされたすべてのロール。

```
select *
from dba_role_privs
where grantee= :username
```

ユーザーにされた

1. システム

```
select *
from dba_sys_privs
where grantee = :username
```

2. オブジェクトグラント

```
select *
from dba_tab_privs
where grantee = :username
```

ロールにえられたパーミッション。

のロールにされたロール。

```
select *
from role_role_privs
where role in (select granted_role from dba_role_privs where grantee= :username)
```

1. システム

```
select *
from role_sys_privs
where role in (select granted_role from dba_role_privs where grantee= :username)
```

2. オブジェクトグラント

```
select *
from role_tab_privs
where role in (select granted_role from dba_role_privs where grantee= :username)
```

Oracleバージョン

```
select *
from v$version
```

データベースのすべてのオブジェクトについてします。

```
select *
```

```
from dba_objects
```

アクセスなすべてのデータ·ディクショナリ·ビューをするには

```
select * from dict
```

オンラインでデータをむ https://riptutorial.com/ja/oracle/topic/7347/データ

19: ヒント

パラ メーター

パラ メーター	
DOP	これは、クエリをくための/プロセスのです。それは2,4,8,16などです。
テーブル	ヒントがされるテーブルの。

Examples

パラレルヒント

ステートメントレベルのヒントがもです。

```
SELECT /*+ PARALLEL(8) */ first_name, last_name FROM employee emp;
```

オブジェクトレベルのヒントは、よりくのをしますが、エラーがしやすくなります。はオブジェクトのわりにエイリアスをすることをれたり、オブジェクトをいくつかめることをれることがあります。

```
SELECT /*+ PARALLEL(emp,8) */ first_name, last_name FROM employee emp;
```

```
SELECT /*+ PARALLEL(table_alias,Degree of Parallelism) */ FROM table_name table_alias;
```

パラレルヒントをせずにクエリをするのに100かかるとしましょう。じクエリにしてDOPを2にすると、 にはヒントをつじクエリに50かかることになります。にDOPを4としてするは25かかります。

には、はのくのにし、にはしません。これは、パラレル・オーバーヘッドがのパラレル・サーバーでされたのよりもきいさなのににてはまります。

USE_NL

ネストループをする。

```
use_nl(AB)
```

このヒントは、ネストされたループメソッドをしてテーブルAとBをするようエンジンにします。これは、ごとのです。ヒントはのをするものではなく、にNLをめます。

```
SELECT /*+use_nl(e d)*/ *
FROM Employees E
```

```
JOIN Departments D on E.DepartmentID = D.ID
```

APPENDヒント

"しいをするためにDIRECT PATHメソッドをする"。

APPENDヒントは、 ダイレクト·パス·ロードをするようにエンジンにします 。つまり、エンジンはメモリーとロックをするのをしませんが、データをスペースにきむことになります。にテーブルのセグメントにされるしいブロックをします。これはですが、いくつかのがあります

- トランザクションをコミットまたはロールバックするまで、じセッションでしたテーブルをみることはできません。
- にトリガがされている、Oracle はダイレクト·パスをしません sqlldrロードのはのです。
- その

。

```
INSERT /*+append*/ INTO Employees
SELECT *
FROM Employees;
```

USE_HASH

ハッシュメソッドをして、のテーブルをするようにエンジンにします。

```
use_hash(TableA [TableB] ... [TableN])
```

くの でされて いるように 、「HASHでは、Oracleはされたのうちさいのにアクセスし、キーのハッシュ·テーブルをメモリーにします。 1、それにマッチするハッシュテーブルをべます。

テーブルがきい、インデックスがでないなどは、ネストループメソッドにしてされます。

ヒントはのをするものではなく、にハッシュメソッドをします。

```
SELECT /*+use_hash(e d)*/ *
FROM Employees E
JOIN Departments D on E.DepartmentID = D.ID
```

FULL

FULLヒントは、をできるかどうかにかかわらず、されたにしてフル·テーブル·スキャンをするようOracleにします。

```
create table fullTable(id) as select level from dual connect by level < 100000;
create index idx on fullTable(id);
```

ヒントがないは、インデックスがされます。

```
select count(1) from fullTable f where id between 10 and 100;
-------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |     1 |    13 |     3   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE   |      |     1 |    13 |            |          |
|*  2 |   INDEX RANGE SCAN| IDX  |     2 |    26 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

FULLヒントはフルスキャンをします

```
select /*+ full(f) */ count(1) from fullTable f where id between 10 and 100;
-----------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |     1 |    13 |    47   (3)| 00:00:01 |
|   1 |  SORT AGGREGATE    |           |     1 |    13 |            |          |
|*  2 |   TABLE ACCESS FULL| FULLTABLE |     2 |    26 |    47   (3)| 00:00:01 |
-----------------------------------------------------------------------------------
```

キャッシュ

Oracle *11g* では、SQLクエリをSGAにキャッシュしてしてパフォーマンスをさせることができます。データベースではなくキャッシュからデータをします。でじクエリをすると、データがキャッシュからされるため、よりにできます。

```
SELECT /*+ result_cache */ number FROM main_table;
```

-

```
Number
------
   1
   2
   3
   4
   5
   6
   7
   8
   9
   10

Elapsed: 00:00:02.20
```

もうじクエリをすると、のにされたキャッシュからデータがされるため、がされます。

-

```
Number
------
   1
   2
```

```
    3
    4
    5
    6
    7
    8
    9
    10

Elapsed: 00:00:00.10
```

どのようにが**2.20**から**0.10**にされたかにしてください。

> キャッシュは、データベースのデータが//されるまでキャッシュをします。によってキャッシュがされます。

オンラインでヒントをむ https://riptutorial.com/ja/oracle/topic/1490/ヒント

20: リアルアプリケーションセキュリティ

き

Oracle Real Application SecurityはOracle 12cでされました。ユーザーロールモデル、アクセス、アプリケーションとデータベース、エンドユーザーセキュリティ、とレベルのセキュリティのようなくのセキュリティトピックをしています

Examples

アプリケーションをデータベースのかにけるには、3つのながあります。

アプリケーションアプリケーションは、 `SELECT` 、 `INSERT` 、 `UPDATE` 、 `DELETE`などのをします。アプリケーションは、としてすることができます。

```
XS$PRIVILEGE(
    name=>'privilege_name'
    [, implied_priv_list=>XS$NAME_LIST('"SELECT"', '"INSERT"', '"UPDATE"', '"DELETE"')]
)

XS$PRIVILEGE_LIST(
    XS$PRIVILEGE(...),
    XS$PRIVILEGE(...),
    ...
);
```

アプリケーションユーザ

シンプルなアプリケーションユーザー

```
BEGIN
    SYS.XS_PRINCIPAL.CREATE_USER('user_name');
END;
```

ダイレクトログインアプリケーションユーザ

```
BEGIN
    SYS.XS_PRINCIPAL.CREATE_USER(name => 'user_name', schema => 'schema_name');
END;

BEGIN
    SYS.XS_PRINCIPAL.SET_PASSWORD('user_name', 'password');
END;
CREATE PROFILE prof LIMIT
    PASSWORD_REUSE_TIME 1/4440
    PASSWORD_REUSE_MAX 3
    PASSWORD_VERIFY_FUNCTION Verify_Pass;

BEGIN
    SYS.XS_PRINCIPAL.SET_PROFILE('user_name', 'prof');
```

```
END;

BEGIN
    SYS.XS_PRINCIPAL.GRANT_ROLES('user_name', 'XSONNCENT');
END;
```

オプション

```
BEGIN
    SYS.XS_PRINCIPAL.SET_VERIFIER('user_name', '6DFF060084ECE67F', XS_PRINCIPAL.XS_SHA512‟);
END;
```

アプリケ―ションロ―ル

のアプリケ―ションの

```
DECLARE
    st_date TIMESTAMP WITH TIME ZONE;
    ed_date TIMESTAMP WITH TIME ZONE;
BEGIN
    st_date := SYSTIMESTAMP;
    ed_date := TO_TIMESTAMP_TZ('2013-06-18 11:00:00 -5:00','YYYY-MM-DD HH:MI:SS');
    SYS.XS_PRINCIPAL.CREATE_ROLE
        (name => 'app_regular_role',
        enabled => TRUE,
        start_date => st_date,
        end_date => ed_date);
END;
```

アプリケ―ションにづいてにになる

```
BEGIN
    SYS.XS_PRINCIPAL.CREATE_DYNAMIC_ROLE
        (name => 'app_dynamic_role',
        duration => 40,
        scope => XS_PRINCIPAL.SESSION_SCOPE);
END;
```

みのアプリケ―ションロ―ル

レギュラ―

- XSPUBLIC
- XSBYPASS
- XSSESSIONADMIN
- XSNAMESPACEADMIN
- XSPROVISIONER
- XSCACHEADMIN
- XSDISPATCHER

:(アプリケ―ションユ―ザのにする

- DBMS_AUTH ログオンまたはそののデ―タベ―ス
- EXTERNAL_DBMS_AUTH

ログオンまたはのデータベースおよびユーザーはです

- `DBMS_PASSWD` パスワードきダイレクトログオン
- `MIDTIER_AUTH` :(アプリケーションによる
- `XSAUTHENTICATED` またはアプリケーション
- `XSSWITCH` ユーザーがプロキシユーザーからアプリケーションユーザーにりえた

オンラインでリアルアプリケーションセキュリティをむ
https://riptutorial.com/ja/oracle/topic/10864/リアルアプリケーションセキュリティ

21: レコードをするさまざまな

- UPDATE table-Name [[AS]] SET column-Name =[、=}] * [WHERE]
- UPDATEテーブル - SETカラム - =[、カラム - =] *の

Examples

をしてをする

の

```
UPDATE
      TESTTABLE
SET
      TEST_COLUMN= 'Testvalue',TEST_COLUMN2= 123
WHERE
      EXISTS
          (SELECT MASTERTABLE.TESTTABLE_ID
           FROM MASTERTABLE
           WHERE ID_NUMBER=11);
```

インライインビューをした

インライン·ビューをする**Oracle**によってとなされる

キーエラーがしたは、インデックスをしてそれをしてにします

```
UPDATE
      (SELECT
            TESTTABLE.TEST_COLUMN AS OLD,
            'Testvalue' AS NEW
       FROM
                TESTTABLE
            INNER JOIN
                MASTERTABLE
            ON TESTTABLE.TESTTABLE_ID = MASTERTABLE.TESTTABLE_ID
       WHERE ID_NUMBER=11) T
SET
      T.OLD        = T.NEW;
```

Mergeをした

マージの

```
MERGE INTO
      TESTTABLE
USING
      (SELECT
            T1.ROWID AS RID,
```

```
            T2.TESTTABLE_ID
       FROM
                TESTTABLE T1
            INNER JOIN
                MASTERTABLE T2
            ON TESTTABLE.TESTTABLE_ID = MASTERTABLE.TESTTABLE_ID
       WHERE ID_NUMBER=11)
ON
      ( ROWID = RID )
WHEN MATCHED
THEN
    UPDATE SET TEST_COLUMN= 'Testvalue';
```

サンプルデータとのマージ

```
drop table table01;
drop table table02;

create table table01 (
       code int,
       name varchar(50),
       old int
);

create table table02 (
       code int,
       name varchar(50),
       old int
);

truncate table table01;
insert into table01 values (1, 'A', 10);
insert into table01 values (9, 'B', 12);
insert into table01 values (3, 'C', 14);
insert into table01 values (4, 'D', 16);
insert into table01 values (5, 'E', 18);

truncate table table02;
insert into table02 values (1, 'AA', null);
insert into table02 values (2, 'BB', 123);
insert into table02 values (3, 'CC', null);
insert into table02 values (4, 'DD', null);
insert into table02 values (5, 'EE', null);

select * from table01 a order by 2;
select * from table02 a order by 2;

--

merge into table02 a using (
      select b.code, b.old from table01 b
) c on (
  a.code = c.code
)
when matched then update set a.old = c.old
;

--
```

```
select a.*, b.* from table01 a
inner join table02 b on a.code = b.codetable01;

select * from table01 a
where
       exists
       (
         select 'x' from table02 b where a.code = b.codetable01
       );

select * from table01 a where a.code in (select b.codetable01 from table02 b);

--

select * from table01 a
where
       not exists
       (
         select 'x' from table02 b where a.code = b.codetable01
       );

select * from table01 a where a.code not in (select b.codetable01 from table02 b);
```

オンラインでレコードをするさまざまなをむ https://riptutorial.com/ja/oracle/topic/4193/レコードをするさまざまな

22: レベルクエリ

いくつかののについてNのダミーレコードをするがあります。

Examples

Nのレコードをする

```
SELECT ROWNUM NO FROM DUAL CONNECT BY LEVEL <= 10
```

レベルクエリのはほとんどありません

/ *これはのをすることができるなクエリです。のでは、1..100 * /

```
select level from dual connect by level <= 100;
```

/ *のクエリは、されたからのシーケンスをするなど、さまざまなシナリオでにちます。のクエリは、する10のをします* /

```
select to_date('01-01-2017','mm-dd-yyyy')+level-1 as  dates from dual connect by level <= 10;
```

01-JAN-17
02-JAN-17
03117
04-JAN-17
05-JAN-17
06-JAN-17
07117
08-JAN-17
0919
10110

オンラインでレベルクエリをむ https://riptutorial.com/ja/oracle/topic/6548/レベルクエリ

23:

Examples

Oracleのしいでキーをする

があり、こののIDの1つをしたいとします。のscrpitをすることができます。プライマリIDは「PK_S」です

```
begin
  for i in (select a.table_name, c.column_name
              from user_constraints a, user_cons_columns c
             where a.CONSTRAINT_TYPE = 'R'
               and a.R_CONSTRAINT_NAME = 'PK_S'
               and c.constraint_name = a.constraint_name) loop


         execute immediate 'update ' || i.table_name || ' set ' || i.column_name ||
                     '=to_number(''1000'' || ' ||  i.column_name || ') ';



  end loop;

end;
```

Oracleのするすべてのキーをにする

テーブルT1があり、それがくのテーブルとのをち、プライマリ･キーのが「pk_t1」で、これらのキーをにしたいとします。

```
Begin
    For I in (select table_name, constraint_name from user_constraint t where
r_constraint_name='pk_t1') loop

Execute immediate ' alter table ' || I.table_name || ' disable constraint ' ||
i.constraint_name;

   End loop;
End;
```

オンラインでをむ https://riptutorial.com/ja/oracle/topic/6040/

24: SQL

き

SQLをすると、にSQLクエリコードをアセンブルできます。このにはいくつかのがあり、にくするがあります。に、よりなロジックをすることができます。 PL / SQLでは、コードでされるすべてのオブジェクトがコンパイルにし、であるがあります。そのため、PL / SQLでDDLをすることはできませんが、SQLではこれをできます。

いくつかのな

1. をしてクエリにをしたり、わりにパラメータをしたりしないでください。これはっています

```
execute immediate 'select value from my_table where id = ' ||
     id_valiable into result_variable;
```

そして、これはしいです

```
execute immediate 'select value from my_table where id = :P '
    using id_valiable into result_variable;
```

これには2つのがあります。 1つはセキュリティです。はSQLインジェクションをにします。のクエリでは、に`1 or 1 = 1`がまれる、 `UPDATE`ステートメントはテーブルのすべてのをします。

```
execute immediate 'update my_table set value = ''I have bad news for you'' where id = '
|| id;
```

2のはパフォーマンスです。 Oracleは、するたびにパラメータなしでせをしますが、パラメータきせはセッションで1のみされます。

2. データベースエンジンがDDLをするとき、にのコミットをすることにしてください。

Examples

SQLでを

ユーザーがなるテーブルからデータをしたいとします。テーブルはユーザーによってされます。

```
function get_value(p_table_name varchar2, p_id number) return varchar2 is
   value varchar2(100);
 begin
   execute immediate 'select column_value from ' || p_table_name ||
                     ' where id = :P' into value using p_id;
   return value;
```

```
  end;
```

どおりこのをびします。

```
declare
  table_name varchar2(30) := 'my_table';
  id number := 1;
begin
  dbms_output.put_line(get_value(table_name, id));
end;
```

テストするテーブル

```
create table my_table (id number, column_value varchar2(100));
insert into my_table values (1, 'Hello, world!');
```

SQLにをする

のは、ののテーブルにvalueをします。

```
declare
  query_text varchar2(1000) := 'insert into my_table(id, column_value) values (:P_ID,
:P_VAL)';
  id number := 2;
  value varchar2(100) := 'Bonjour!';
begin
  execute immediate query_text using id, value;
end;
/
```

SQLのをする

のからテーブルをしましょう

```
declare
  query_text varchar2(1000) := 'update my_table set column_value = :P_VAL where id = :P_ID';
  id number := 2;
  value varchar2(100) := 'Bonjour le monde!';
begin
  execute immediate query_text using value, id;
end;
/
```

DDLをする

このコードはテーブルをします

```
begin
  execute immediate 'create table my_table (id number, column_value varchar2(100))';
end;
/
```

ブロックをする

ブロックをできます。このは、SQLからをすもしています。

```
declare
  query_text varchar2(1000) := 'begin :P_OUT := cos(:P_IN); end;';
  in_value number := 0;
  out_value number;
begin
  execute immediate query_text using out out_value, in in_value;
  dbms_output.put_line('Result of anonymous block: ' || to_char(out_value));
end;
/
```

オンラインでSQLをむ https://riptutorial.com/ja/oracle/topic/10905/sql

25: られたの

Examples

サブクエリファクタリングをしたの

サンプルデータ

```
CREATE TABLE table_name ( id, list ) AS
SELECT 1, 'a,b,c,d' FROM DUAL UNION ALL -- Multiple items in the list
SELECT 2, 'e'       FROM DUAL UNION ALL -- Single item in the list
SELECT 3, NULL      FROM DUAL UNION ALL -- NULL list
SELECT 4, 'f,,g'    FROM DUAL;          -- NULL item in the list
```

クエリ

```
WITH bounds ( id, list, start_pos, end_pos, lvl ) AS (
  SELECT id, list, 1, INSTR( list, ',' ), 1 FROM table_name
UNION ALL
  SELECT id,
         list,
         end_pos + 1,
         INSTR( list, ',', end_pos + 1 ),
         lvl + 1
  FROM   bounds
  WHERE  end_pos > 0
)
SELECT id,
       SUBSTR(
         list,
         start_pos,
         CASE end_pos
           WHEN 0
           THEN LENGTH( list ) + 1
           ELSE end_pos
         END - start_pos
       ) AS item,
      lvl
FROM   bounds
ORDER BY id, lvl;
```

```
        ID ITEM           LVL
---------- ------- ----------
         1 a                1
         1 b                2
         1 c                3
         1 d                4
         2 e                1
         3 (NULL)           1
         4 f                1
         4 (NULL)           2
         4 g                3
```

PL / SQLファンクションをしたの

PL / SQLファンクション

```
CREATE OR REPLACE FUNCTION split_String(
  i_str    IN  VARCHAR2,
  i_delim  IN  VARCHAR2 DEFAULT ','
) RETURN SYS.ODCIVARCHAR2LIST DETERMINISTIC
AS
  p_result       SYS.ODCIVARCHAR2LIST := SYS.ODCIVARCHAR2LIST();
  p_start        NUMBER(5) := 1;
  p_end          NUMBER(5);
  c_len CONSTANT NUMBER(5) := LENGTH( i_str );
  c_ld  CONSTANT NUMBER(5) := LENGTH( i_delim );
BEGIN
  IF c_len > 0 THEN
    p_end := INSTR( i_str, i_delim, p_start );
    WHILE p_end > 0 LOOP
      p_result.EXTEND;
      p_result( p_result.COUNT ) := SUBSTR( i_str, p_start, p_end - p_start );
      p_start := p_end + c_ld;
      p_end := INSTR( i_str, i_delim, p_start );
    END LOOP;
    IF p_start <= c_len + 1 THEN
      p_result.EXTEND;
      p_result( p_result.COUNT ) := SUBSTR( i_str, p_start, c_len - p_start + 1 );
    END IF;
  END IF;
  RETURN p_result;
END;
/
```

サンプルデ―タ

```
CREATE TABLE table_name ( id, list ) AS
SELECT 1, 'a,b,c,d' FROM DUAL UNION ALL -- Multiple items in the list
SELECT 2, 'e'       FROM DUAL UNION ALL -- Single item in the list
SELECT 3, NULL      FROM DUAL UNION ALL -- NULL list
SELECT 4, 'f,,g'    FROM DUAL;          -- NULL item in the list
```

クエリ

```
SELECT t.id,
       v.column_value AS value,
       ROW_NUMBER() OVER ( PARTITION BY id ORDER BY ROWNUM ) AS lvl
FROM   table_name t,
       TABLE( split_String( t.list ) ) (+) v
```

```
        ID ITEM           LVL
---------- ------- ----------
         1 a                 1
         1 b                 2
         1 c                 3
         1 d                 4
         2 e                 1
         3 (NULL)            1
```

```
         4 f                1
         4 (NULL)           2
         4 g                3
```

テーブルをしたの

サンプルデータ

```
CREATE TABLE table_name ( id, list ) AS
SELECT 1, 'a,b,c,d' FROM DUAL UNION ALL -- Multiple items in the list
SELECT 2, 'e'       FROM DUAL UNION ALL -- Single item in the list
SELECT 3, NULL      FROM DUAL UNION ALL -- NULL list
SELECT 4, 'f,,g'    FROM DUAL;          -- NULL item in the list
```

クエリ

```
SELECT t.id,
       v.COLUMN_VALUE AS value,
       ROW_NUMBER() OVER ( PARTITION BY id ORDER BY ROWNUM ) AS lvl
FROM   table_name t,
       TABLE(
         CAST(
           MULTISET(
             SELECT REGEXP_SUBSTR( t.list, '([^,]*)(,|$)', 1, LEVEL, NULL, 1 )
             FROM   DUAL
             CONNECT BY LEVEL < REGEXP_COUNT( t.list, '[^,]*(,|$)' )
           )
           AS SYS.ODCIVARCHAR2LIST
         )
       ) v;
```

```
        ID ITEM           LVL
---------- ------- ----------
         1 a                1
         1 b                2
         1 c                3
         1 d                4
         2 e                1
         3 (NULL)           1
         4 f                1
         4 (NULL)           2
         4 g                3
```

クエリをしたの

サンプルデータ

```
CREATE TABLE table_name ( id, list ) AS
SELECT 1, 'a,b,c,d' FROM DUAL UNION ALL -- Multiple items in the list
SELECT 2, 'e'       FROM DUAL UNION ALL -- Single item in the list
SELECT 3, NULL      FROM DUAL UNION ALL -- NULL list
SELECT 4, 'f,,g'    FROM DUAL;          -- NULL item in the list
```

クエリ

```
SELECT t.id,
       REGEXP_SUBSTR( list, '([^,]*)(,|$)', 1, LEVEL, NULL, 1 ) AS value,
       LEVEL AS lvl
FROM   table_name t
CONNECT BY
       id = PRIOR id
AND    PRIOR SYS_GUID() IS NOT NULL
AND    LEVEL < REGEXP_COUNT( list, '([^,]*)(,|$)' )
```

```
        ID ITEM           LVL
---------- ------- ----------
         1 a                1
         1 b                2
         1 c                3
         1 d                4
         2 e                1
         3 (NULL)           1
         4 f                1
         4 (NULL)           2
         4 g                3
```

XMLTableとFLWORをしたの

このソリューションでは、Oracle 11からな`ora:tokenize` XQueryをします。

サンプルデータ

```
CREATE TABLE table_name ( id, list ) AS
SELECT 1, 'a,b,c,d' FROM DUAL UNION ALL -- Multiple items in the list
SELECT 2, 'e'       FROM DUAL UNION ALL -- Single item in the list
SELECT 3, NULL      FROM DUAL UNION ALL -- NULL list
SELECT 4, 'f,,g'    FROM DUAL;          -- NULL item in the list
```

クエリ

```
SELECT t.id,
       x.item,
       x.lvl
FROM   table_name t,
       XMLTABLE(
         'let $list := ora:tokenize(.,","),
              $cnt := count($list)
          for $val at $r in $list
          where $r < $cnt
          return $val'
       PASSING list||','
       COLUMNS
         item VARCHAR2(100) PATH '.',
         lvl FOR ORDINALITY
       ) (+) x;
```

```
        ID ITEM           LVL
```

```
---------- ------- ----------
         1 a                1
         1 b                2
         1 c                3
         1 d                4
         2 e                1
         3 (NULL)       (NULL)
         4 f                1
         4 (NULL)           2
         4 g                3
```

CROSS APPLYOracle 12cをしたの

サンプルデ―タ

```
CREATE TABLE table_name ( id, list ) AS
SELECT 1, 'a,b,c,d' FROM DUAL UNION ALL -- Multiple items in the list
SELECT 2, 'e'       FROM DUAL UNION ALL -- Single item in the list
SELECT 3, NULL      FROM DUAL UNION ALL -- NULL list
SELECT 4, 'f,,g'    FROM DUAL;          -- NULL item in the list
```

クエリ

```
SELECT t.id,
       REGEXP_SUBSTR( t.list, '([^,]*)($|,)', 1, l.lvl, NULL, 1 ) AS item,
       l.lvl
FROM   table_name t
       CROSS APPLY
       (
         SELECT LEVEL AS lvl
         FROM   DUAL
         CONNECT BY LEVEL <= REGEXP_COUNT( t.list, ',' ) + 1
       ) l;
```

```
        ID ITEM           LVL
---------- ------- ----------
         1 a                1
         1 b                2
         1 c                3
         1 d                4
         2 e                1
         3 (NULL)           1
         4 f                1
         4 (NULL)           2
         4 g                3
```

XMLTableをしてられたをする

サンプルデ―タ

```
CREATE TABLE table_name ( id, list ) AS
SELECT 1, 'a,b,c,d' FROM DUAL UNION ALL -- Multiple items in the list
SELECT 2, 'e'       FROM DUAL UNION ALL -- Single item in the list
SELECT 3, NULL      FROM DUAL UNION ALL -- NULL list
```

```
SELECT 4, 'f,,g'    FROM DUAL;          -- NULL item in the list
```

クエリ

```
SELECT t.id,
       SUBSTR( x.item.getStringVal(), 2 ) AS item,
       x.lvl
FROM   table_name t
       CROSS JOIN
       XMLTABLE(
         ( '"#' || REPLACE( t.list, ',', '","#' ) || '"' )
         COLUMNS item XMLTYPE PATH '.',
                 lvl  FOR ORDINALITY
       ) x;
```

*#*をしやすくするためにされる*NULL*を、それは、にしてされる*SUBSTR(item, 2)*。 *NULL*は、なされていないクエリをし、これをすることができます。

```
        ID ITEM           LVL
---------- ------- ----------
         1 a                1
         1 b                2
         1 c                3
         1 d                4
         2 e                1
         3 (NULL)           1
         4 f                1
         4 (NULL)           2
         4 g                3
```

オンラインでられたのをむ https://riptutorial.com/ja/oracle/topic/1968/られたの

26: PL / SQLブロック

がけられていないので、ブロックはのプログラムですることはできません。

Examples

ブロックの

```
DECLARE
    -- declare a variable
    message varchar2(20);
BEGIN
  -- assign value to variable
  message := 'HELLO WORLD';

  -- print message to screen
  DBMS_OUTPUT.PUT_LINE(message);
END;
/
```

オンラインでPL / SQLブロックをむ https://riptutorial.com/ja/oracle/topic/6451/pl---sqlブロック

27:

Examples

||またはconcat

Oracle SQLとPL / SQL `||`をすると、2つのをすることができます。

の`customers`テーブルをします。

```
 id  firstname    lastname
---  -----------  ----------
  1  Thomas       Woody
```

クエリ

```
 SELECT firstname || ' ' || lastname || ' is in my database.' as "My Sentence"
   FROM customers;
```

```
My Sentence
--------------------------------
Thomas Woody is in my database.
```

Oracleは、のSQL `CONCAT(str1, str2)`もサポートしています。

クエリ

```
 SELECT CONCAT(firstname, ' is in my database.') from customers;
```

```
Expr1
--------------------------------
Thomas is in my database.
```

アッパー

UPPERをすると、のすべてのをにできます。

```
SELECT UPPER('My text 123!') AS result FROM dual;
```

```
RESULT
------------
MY TEXT 123!
```

INITCAP

INITCAPは、がでまり、すべてののがになるように、の/をします。

```
SELECT INITCAP('HELLO mr macdonald!') AS NEW FROM dual;
```

```
NEW
-------------------
Hello Mr Macdonald!
```

LOWER

LOWERは、のすべてのをにします。

```
SELECT LOWER('HELLO World123!') text FROM dual;
```

テキスト
こんにちはworld123

だけを2できえたいとしましょう。では(\d\d)

```
SELECT REGEXP_REPLACE ('2, 5, and 10 are numbers in this example', '(\d\d)', '#')
FROM dual;
```

```
'2, 5, and # are numbers in this example'
```

テキストのをれえるは、 \1 、 \2 、 \3をってするをびします。

```
 SELECT REGEXP_REPLACE ('swap around 10 in that one ', '(.*)(\d\d )(.*)', '\3\2\1\3')
 FROM dual;
```

SUBSTR

SUBSTRは、するのとSUBSTRをしてのをします。

```
SELECT SUBSTR('abcdefg',2,3) FROM DUAL;
```

り

```
bcd
```

のからえるには、 SUBSTRは2のパラメータとしてのをけります。

```
SELECT SUBSTR('abcdefg',-4,2) FROM DUAL;
```

り

```
de
```

ののをするには `SUBSTR(mystring,-1,1)`

LTRIM / RTRIM

`LTRIM`および`RTRIM`は、のまたはからをします。するには、1つのセットをすることができますデフォルトはスペ―スです。

えば、

```
select LTRIM('<===>HELLO<===>', '=<>')
      ,RTRIM('<===>HELLO<===>', '=<>')
from dual;
```

り

```
HELLO<===>
<===>HELLO
```

オンラインでをむ https://riptutorial.com/ja/oracle/topic/1518/

28:

Examples

コンポーネントなしのの

すべてのDATEはがあります。しかし、をめるのないを//がゼロすなわちにしてすることがである。

ANSI DATEリテラルをする ISO 8601のを。

```
SELECT DATE '2000-01-01' FROM DUAL;
```

TO_DATE()をしてリテラルからします。

```
SELECT TO_DATE( '2001-01-01', 'YYYY-MM-DD' ) FROM DUAL;
```

モデルのは、*Oracle*のマニュアルをしてください。

または

```
SELECT TO_DATE(
         'January 1, 2000, 00:00 A.M.',
         'Month dd, YYYY, HH12:MI A.M.',
         'NLS_DATE_LANGUAGE = American'
       )
FROM   DUAL;
```

などののをするは、3の*nlsparam*パラメータを*TO_DATE()*に*TO_DATE()*、されるをすることをおめします。

コンポーネントをしたの

TO_DATE()をしてリテラルからします。

```
SELECT TO_DATE( '2000-01-01 12:00:00', 'YYYY-MM-DD HH24:MI:SS' ) FROM DUAL;
```

または、 TIMESTAMPリテラルをしTIMESTAMP。

```
CREATE TABLE date_table(
  date_value DATE
);

INSERT INTO date_table ( date_value ) VALUES ( TIMESTAMP '2000-01-01 12:00:00' );
```

Oracleは、のDATEにTIMESTAMPをするときに、にTIMESTAMPをDATEにキャストしTIMESTAMP。ただし、にをDATE CAST()することができます。

```
SELECT CAST( TIMESTAMP '2000-01-01 12:00:00' AS DATE ) FROM DUAL;
```

の

Oracleでは、 DATEデータにはフォーマットがありません。 Oracleがクライアントプログラム(SQL / Plus、SQL / Developer、Toad、Java、Pythonなど)にDATEをすると、をす7バイトまたは8バイトがされます。

テーブルにされていないつまり、 SYSDATEによってされ、 DUMP()コマンドをするときに "タイプ13" をつ DATEは、8バイトでされていますのは2012-11-26 16:41:092012-11-26 16:41:09

```
BYTE VALUE                          EXAMPLE
---- ------------------------------ ---------------------------------------
1    Year modulo 256                220
2    Year multiples of 256          7   (7 * 256 + 220 = 2012)
3    Month                          11
4    Day                            26
5    Hours                          16
6    Minutes                        41
7    Seconds                        9
8    Unused                         0
```

テーブルにされているDATE DUMP()コマンドをするは「タイプ12」は7バイトでされていますのは2012-11-26 16:41:09です

```
BYTE VALUE                          EXAMPLE
---- ------------------------------ ---------------------------------------
1    ( Year multiples of 100 ) + 100 120
2    ( Year modulo 100 ) + 100      112 ((120-100)*100 + (112-100) = 2012)
3    Month                          11
4    Day                            26
5    Hours + 1                      17
6    Minutes + 1                    42
7    Seconds + 1                    10
```

にのをするは、つまりをつものにするがあります。 SQLクライアントはにこれをうか、にTO_CHAR(date, format_model, nls_params)をTO_CHAR(date, format_model, nls_params) てをにできます。

をにする

TO_CHAR(date [, format_model [, nls_params]])ます。

フォーマット・モデルがされていないは、 NLS_DATE_FORMATセッション・パラメータがデフォルト・フォーマット・モデルとしてされますが、これはすべてのセッションでなるがありますので、するはありません。

```
CREATE TABLE table_name (
  date_value DATE
);
```

```
INSERT INTO table_name ( date_value ) VALUES ( DATE '2000-01-01' );
INSERT INTO table_name ( date_value ) VALUES ( TIMESTAMP '2016-07-21 08:00:00' );
INSERT INTO table_name ( date_value ) VALUES ( SYSDATE );
```

に

```
SELECT TO_CHAR( date_value, 'YYYY-MM-DD' ) AS formatted_date FROM table_name;
```

```
FORMATTED_DATE
--------------
2000-01-01
2016-07-21
2016-07-21
```

そして

```
SELECT TO_CHAR(
         date_value,
         'FMMonth d yyyy, hh12:mi:ss AM',
         'NLS_DATE_LANGUAGE = French'
       ) AS formatted_date
FROM   table_name;
```

```
FORMATTED_DATE
-----------------------------
Janvier   01 2000, 12:00:00 AM
Juillet   21 2016, 08:00:00 AM
Juillet   21 2016, 19:08:31 PM
```

デフォルトのモデルの

Oracleが`DATE`からににする、または`TO_CHAR()`または`TO_DATE()`がモデルなしでにびされたは、`NLS_DATE_FORMAT`セッション･パラメータがのフォーマット･モデルとしてされます。リテラルがモデルとしない、がします。

このパラメータはのでできます。

```
SELECT VALUE FROM NLS_SESSION_PARAMETERS WHERE PARAMETER = 'NLS_DATE_FORMAT';
```

のセッションでこのをするには、のようにします。

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS';
```

のユーザーのはされません。

`TO_DATE()`または`TO_CHAR()`フォーマット･マスクをするために`NLS_DATE_FORMAT`をするは、このがされたにがしてもかないはずです。

SQL / PlusまたはSQL Developerによるのの

SQL / PlusまたはSQL Developerでがされると、デフォルトのモデルをしてへのながされます「のモデルの」のを。

NLS_DATE_FORMATパラメータをすると、のをできます。

- 、、および/またはののい

オラクルでは、2つのDATEのおよび/またはそのは、をしてつけることができます。

```
SELECT DATE '2016-03-23' - DATE '2015-12-25' AS difference FROM DUAL;
```

2つののをします。

```
DIFFERENCE
----------
        89
```

そして

```
SELECT TO_DATE( '2016-01-02 01:01:12', 'YYYY-MM-DD HH24:MI:SS' )
         - TO_DATE( '2016-01-01 00:00:00', 'YYYY-MM-DD HH24:MI:SS' )
         AS difference
FROM   DUAL
```

2つののをします。

```
DIFFERENCE
----------
    1.0425
```

、、のがで、このをけることによりめることができる24 、 24*60または24*60*60それぞれ。

のをすると、のように2つのの、、、をできます。

```
SELECT TRUNC( difference                        ) AS days,
       TRUNC( MOD( difference * 24,       24 ) ) AS hours,
       TRUNC( MOD( difference * 24*60,    60 ) ) AS minutes,
       TRUNC( MOD( difference * 24*60*60, 60 ) ) AS seconds
FROM   (
  SELECT TO_DATE( '2016-01-02 01:01:12', 'YYYY-MM-DD HH24:MI:SS' )
         - TO_DATE( '2016-01-01 00:00:00', 'YYYY-MM-DD HH24:MI:SS' )
         AS difference
  FROM   DUAL
```

;

のいをしくするには、 *FLOOR()* ではなく *TRUNC()* をします。

```
DAYS HOURS MINUTES SECONDS
---- ----- ------- -------
   1     1       1      12
```

のは、 `NUMTODSINTERVAL()` をしてのをにすることによってもできます。

```
SELECT EXTRACT( DAY    FROM difference ) AS days,
       EXTRACT( HOUR   FROM difference ) AS hours,
       EXTRACT( MINUTE FROM difference ) AS minutes,
       EXTRACT( SECOND FROM difference ) AS seconds
FROM   (
  SELECT NUMTODSINTERVAL(
           TO_DATE( '2016-01-02 01:01:12', 'YYYY-MM-DD HH24:MI:SS' )
             - TO_DATE( '2016-01-01 00:00:00', 'YYYY-MM-DD HH24:MI:SS' ),
           'DAY'
         ) AS difference
  FROM   DUAL
);
```

- またはののい

2つののののいは、 `MONTHS_BETWEEN( date1, date2 )` をして `MONTHS_BETWEEN( date1, date2 )` できます。

```
SELECT MONTHS_BETWEEN( DATE '2016-03-10', DATE '2015-03-10' ) AS difference FROM DUAL;
```

```
DIFFERENCE
----------
        12
```

にながまれているは、**31**についてそののをします。

```
SELECT MONTHS_BETWEEN( DATE '2015-02-15', DATE '2015-01-01' ) AS difference FROM DUAL;
```

```
DIFFERENCE
----------
 1.4516129
```

`MONTHS_BETWEEN`ため、1かに31`MONTHS_BETWEEN`すると、のがなくなるがあるため、のにまたがるのがなることがあります。

```
SELECT MONTHS_BETWEEN( DATE'2016-02-01', DATE'2016-02-01' - INTERVAL '1' DAY ) AS "JAN-FEB",
       MONTHS_BETWEEN( DATE'2016-03-01', DATE'2016-03-01' - INTERVAL '1' DAY ) AS "FEB-MAR",
       MONTHS_BETWEEN( DATE'2016-04-01', DATE'2016-04-01' - INTERVAL '1' DAY ) AS "MAR-APR",
       MONTHS_BETWEEN( DATE'2016-05-01', DATE'2016-05-01' - INTERVAL '1' DAY ) AS "APR-MAY"
FROM   DUAL;
```

```
JAN-FEB FEB-MAR MAR-APR APR-MAY
------- ------- ------- -------
0.03226 0.09677 0.03226 0.06452
```

のいはを12でってめられます。

、、、、またはのコンポーネントをする

DATEデータの、、のコンポーネントは、 `EXTRACT( [ YEAR | MONTH | DAY ] FROM datevalue )` をしてできます。

```
SELECT EXTRACT (YEAR  FROM DATE '2016-07-25') AS YEAR,
       EXTRACT (MONTH FROM DATE '2016-07-25') AS MONTH,
       EXTRACT (DAY   FROM DATE '2016-07-25') AS DAY
FROM DUAL;
```

```
YEAR MONTH DAY
---- ----- ---
2016     7  25
```

、のコンポーネントは、のいずれかによってできます。

- `CAST( datevalue AS TIMESTAMP )` するDATEにTIMESTAMPして、そのし、 `EXTRACT( [ HOUR | MINUTE | SECOND ] FROM timestampvalue )` または
- `TO_CHAR( datevalue, format_model )` をしてをとしてします。

えば

```
SELECT EXTRACT( HOUR   FROM CAST( datetime AS TIMESTAMP ) ) AS Hours,
       EXTRACT( MINUTE FROM CAST( datetime AS TIMESTAMP ) ) AS Minutes,
       EXTRACT( SECOND FROM CAST( datetime AS TIMESTAMP ) ) AS Seconds
FROM   (
  SELECT TO_DATE( '2016-01-01 09:42:01', 'YYYY-MM-DD HH24:MI:SS' ) AS datetime FROM DUAL
);
```

```
HOURS MINUTES SECONDS
----- ------- -------
    9      42       1
```

タイムゾーンと

DATEデータは、タイムゾーンやのをしません。

どちらか

- `TIMESTAMP WITH TIME ZONE` データをしTIMESTAMP WITH TIME ZONE。 または
- アプリケーションロジックのをします。

DATEはUTCとしてし、のようにのセッションのタイムゾーンにできます。

```
SELECT FROM_TZ(
         CAST(
           TO_DATE( '2016-01-01 12:00:00', 'YYYY-MM-DD HH24:MI:SS' )
```

```
            AS TIMESTAMP
          ),
          'UTC'
        )
        AT LOCAL AS time
FROM    DUAL;
```

`ALTER SESSION SET TIME_ZONE = '+01:00';`はのようになります。

```
TIME
-----------------------------------
2016-01-01 13:00:00.000000000 +01:00
```

および`ALTER SESSION SET TIME_ZONE = 'PST';`はのようになります。

```
TIME
-----------------------------------
2016-01-01 04:00:00.000000000 PST
```

うるう

Oracle はをしません 。は、My Oracleサポート・ノート`2019397.2`および`730795.1`をしてください。

のをする

`TO_CHAR( date_value, 'D' )`をしてをできます。

ただし、これは`NLS_TERRITORY`セッション・パラメータにします。

```
ALTER SESSION SET NLS_TERRITORY = 'AMERICA';        -- First day of week is Sunday
SELECT TO_CHAR( DATE '1970-01-01', 'D' ) FROM DUAL;
```

```
5
```

```
ALTER SESSION SET NLS_TERRITORY = 'UNITED KINGDOM'; -- First day of week is Monday
SELECT TO_CHAR( DATE '1970-01-01', 'D' ) FROM DUAL;
```

アウトプット`4`

これを`NLS`とはにうには、のの0をりてのかのをする、のiso-weekににまりますのをりてます。

```
SELECT TRUNC( date_value ) - TRUNC( date_value, 'IW' ) + 1 FROM DUAL
```

オンラインでをむ https://riptutorial.com/ja/oracle/topic/2087/

29: の

Examples

オラクルでは、 `DATE` もいまでのをむおよび`TIMESTAMP` までのをむデータがサポートされており、およびがネイティブにです。えば

になるには

```
select to_char(sysdate + 1, 'YYYY-MM-DD') as tomorrow from dual;
```

をするには

```
select to_char(sysdate - 1, 'YYYY-MM-DD') as yesterday from dual;
```

のに5をするには

```
select to_char(sysdate + 5, 'YYYY-MM-DD') as five_days_from_now from dual;
```

のに5をするには

```
select to_char(sysdate + (5/24), 'YYYY-MM-DD HH24:MI:SS') as five_hours_from_now from dual;
```

のに10をするには

```
select to_char(sysdate + (10/1440), 'YYYY-MM-DD HH24:MI:SS') as ten_mintues_from_now from
dual;
```

のに7をするには

```
select to_char(sysdate + (7/86400), 'YYYY-MM-DD HH24:MI:SS') as seven_seconds_from_now from
dual;
```

`hire_date`が30のをするには

```
select * from emp where hire_date < sysdate - 30;
```

`last_updated`カラムがの1にあるをするには

```
select * from logfile where last_updated >= sysdate - (1/24);
```

また、Oracleは、えば1.5、36、2カなどをすビルトイン･データ`INTERVAL`をします。これらは、`DATE`および`TIMESTAMP`のでもでき`TIMESTAMP` 。えば

```
select * from logfile where last_updated >= sysdate - interval '1' hour;
```

Add_months

```
add_months(p_date, integer) return date;
```

Add_monthsは、amtをp_dateのにします。

```
SELECT add_months(date'2015-01-12', 2) m FROM dual;
```

M
2015-03-12

またsubstractケマイナスの`amt`

```
SELECT add_months(date'2015-01-12', -2) m FROM dual;
```

M
2014-11-12

されたのがしたよりない、されたのがされます。

```
SELECT to_char( add_months(date'2015-01-31', 1),'YYYY-MM-DD') m FROM dual;
```

M
2015-02-28

オンラインでのをむ https://riptutorial.com/ja/oracle/topic/768/の

30: による

き

なSOをむとはに、Oracleはをしてをします。しかし、いくつかのかなりながあります。たちはがうまくいかないか、そしてなをしてがこるかをします。じことをするのは、MERGEステートメントです。

Examples

するものとしないもの

```
create table tgt ( id, val ) as
  select 1, 'a' from dual union all
  select 2, 'b' from dual
;

Table TGT created.

create table src ( id, val ) as
  select 1, 'x' from dual union all
  select 2, 'y' from dual
;

Table SRC created.

update
  ( select t.val as t_val, s.val as s_val
    from   tgt t inner join src s on t.id = s.id
  )
set t_val = s_val
;

SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved table
01779. 00000 -  "cannot modify a column which maps to a non key-preserved table"
*Cause:    An attempt was made to insert or update columns of a join view which
           map to a non-key-preserved table.
*Action:   Modify the underlying base tables directly.
```

`src.id`のに`1`がされていて、 `src.val`がなる、がこるかをしてみて`src.val`。らかに、はをなさないでしょうANYデータベースで - それはなです。、 たちはにはがないことをって`src.id`、しかし、Oracleのエンジンはそれをらない-それはをっています。おそらくこれが、くのがOracleに「でUPDATEしていない」とえているです。

オラクルがしているのは、 `src.id`はであるべきであり、Oracleはこれをにっているということです。にのがのをするがあるは、のにあるキーでもじことがくことにしてください。には、 `src.id`はPKであり、 `tgt.id`はこのPKをしFKかもしれませんが、これはによるにはありません。していることはユニークです。

```
alter table src add constraint src_uc unique (id);

Table SRC altered.

update
  ( select t.val as t_val, s.val as s_val
    from   tgt t inner join src s on t.id = s.id
  )
set t_val = s_val
;

2 rows updated.

select * from tgt;

ID  VAL
--  ---
 1  x
 2  y
```

MERGEステートメントのドキュメンテーションをしてくださいでもじがられますが、はこれらのケースでにMERGEをむが、そのは「Oracleはによるをしない」ということではない。このにすように、Oracle はによるをいます。

オンラインでによるをむ https://riptutorial.com/ja/oracle/topic/8061/による

31:

Examples

ののの

Oracle 10gのMEDIANは、いやすいです。

```
SELECT MEDIAN(SAL)
FROM EMP
```

のをします

DATETIMEでもします。

> MEDIANのは、にをべえることによってされます。グループのとしてNをすると、OracleはRN =1 +0.5 *N-1をしてRNをします。CRN = CEILINGRNおよびFRN = FLOORRNのからののの。

Oracle 9iでは、 PERIENLE_CONTをすることができます。これは、MEDIANとじきをち、パーセンタイルのデフォルトは0.5です

```
SELECT PERCENTILE_CONT(.5) WITHIN GROUP(order by SAL)
FROM EMP
```

は、セットがそのからどのくらいがっているかをします。なから、それはそのから2されたであり、よりくになるほどそのはきくなる。

のは、のをします

```
SELECT name, salary, VARIANCE(salary) "Variance"
FROM employees
```

STDDEV

STDDEVは、exprのサンプル、つまりのをします。とのとしてできます。 STDDEV_SAMPは、1のデータしかたないはSTDDEVがゼロをし、STDDEV_SAMPはNULLをすでSTDDEV_SAMPとはなります。

Oracle Databaseでは、がVARIANCEにされたのとしてされます。

このは、のデータまたはにデータにできるデータをとしてとります。このは、のデータとじデータをします。

DISTINCTをすると、analytic_clauseのquery_partition_clauseのみをできます。 order_by_clause

およびwindowing_clauseはされていません。

のは、サンプル**hr.employees**ののをします。

hrはスキーマで、employeesはテーブルです。

```
SELECT STDDEV(salary) "Deviation"
FROM employees;

Deviation
----------
3909.36575
```

ののクエリは、サンプルテーブルhr.employeesの80ののをhire_dateでします。

```
SELECT last_name, salary,
STDDEV(salary) OVER (ORDER BY hire_date) "StdDev"
FROM employees
WHERE department_id = 30;

LAST_NAME                     SALARY     StdDev
------------------------- ---------- ----------
Raphaely                       11000          0
Khoo                            3100 5586.14357
Tobias                          2800  4650.0896
```

オンラインでをむ https://riptutorial.com/ja/oracle/topic/2283/

32: トランザクション

トランザクションのなはのとおりです。

1. のでしたエラーロギングフレームワークのようなロギングフレームワークをするため。
2. トランザクションのステータスCOMMITまたはROLLBACKになく、のトリガーでDMLをします。

Examples

ロギングエラーのためのトランザクションの

のは、なエラーログテーブルにアプリケーションのすべてのエラーをするためにされるなです。

```
CREATE OR REPLACE PROCEDURE log_errors
(
  p_calling_program IN VARCHAR2,
  p_error_code IN INTEGER,
  p_error_description IN VARCHAR2
)
IS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  INSERT INTO error_log
  VALUES
  (
  p_calling_program,
  p_error_code,
  p_error_description,
  SYSDATE,
  USER
  );
  COMMIT;
END log_errors;
```

のPLSQLブロックは、log_errorsプロシージャのびしをしています。

```
BEGIN
   DELETE FROM dept WHERE deptno = 10;
EXCEPTION
   WHEN OTHERS THEN
      log_errors('Delete dept',sqlcode, sqlerrm);
      RAISE;
END;

SELECT * FROM error_log;

CALLING_PROGRAM    ERROR_CODE    ERROR_DESCRIPTION
ERROR_DATETIME         DB_USER
Delete dept        -2292         ORA-02292: integrity constraint violated - child record found
08/09/2016             APEX_PUBLIC_USER
```

オンラインでトランザクションをむ https://riptutorial.com/ja/oracle/topic/6103/トランザクション

33: のパーティション

き

パーティショニングは、テーブルとインデックスをよりさなにするです。パフォーマンスをさせ、よりさいをにするためにされます。パーティションキーは、がどのパーティションにされるかをするまたはのです。 オラクルのドキュメントのパーティションの

パーティショニングはコストオプションで、Enterprise Editionでのみできます。

Examples

ハッシュパーティショニング

これは、このではstore idにハッシュでられたテーブルをします。

```
CREATE TABLE orders (
    order_nr NUMBER(15),
    user_id VARCHAR2(2),
    order_value NUMBER(15),
    store_id NUMBER(5)
)
PARTITION BY HASH(store_id) PARTITIONS 8;
```

ハッシュ・パーティションのには2のをするがあります。これにより、パーティション・サイズになながられます。

レンジ

このでは、でられたがされます。

```
CREATE TABLE orders (
    order_nr NUMBER(15),
    user_id VARCHAR2(2),
    order_value NUMBER(15),
    store_id NUMBER(5)
)
PARTITION BY RANGE(order_value) (
    PARTITION p1 VALUES LESS THAN(10),
    PARTITION p2 VALUES LESS THAN(40),
    PARTITION p3 VALUES LESS THAN(100),
    PARTITION p4 VALUES LESS THAN(MAXVALUE)
);
```

のパーティションを

スキーマののパーティションをする

```
SELECT * FROM user_tab_partitions;
```

リストのパ―ティション

このでは、ストアIDにリストでパ―ティションされたテ―ブルがされます。

```
CREATE TABLE orders (
    order_nr NUMBER(15),
    user_id VARCHAR2(2),
    order_value NUMBER(15),
    store_id NUMBER(5)
)
PARTITION BY LIST(store_id) (
    PARTITION p1 VALUES (1,2,3),
    PARTITION p2 VALUES(4,5,6),
    PARTITION p3 VALUES(7,8,9),
    PARTITION p4 VALUES(10,11)
);
```

ドロップパ―ティション

```
ALTER TABLE table_name DROP PARTITION partition_name;
```

パ―ティションからデ―タをする

パ―ティションからデ―タをする

```
SELECT * FROM orders PARTITION(partition_name);
```

パ―ティションをりてる

```
ALTER TABLE table_name TRUNCATE PARTITION partition_name;
```

パ―ティションのをする

```
ALTER TABLE table_name RENAME PARTITION p3 TO p6;
```

パ―ティションをのにする

```
ALTER TABLE table_name
MOVE PARTITION partition_name TABLESPACE tablespace_name;
```

しいパ―ティションをする

```
ALTER TABLE table_name
ADD PARTITION new_partition VALUES LESS THAN(400);
```

をする

いくつかのパ―ティションをのをつ2つのパ―ティションにします。

```
ALTER TABLE table_name SPLIT PARTITION old_partition
    AT (new_high_bound) INTO (PARTITION new_partition TABLESPACE new_tablespace,
    PARTITION old_partition)
```

パ―ティションをマ―ジする

2つのパ―ティションを1つにマ―ジする

```
ALTER TABLE table_name
  MERGE PARTITIONS first_partition, second_partition
  INTO  PARTITION  splitted_partition TABLESPACE new_tablespace
```

パ―ティションをする

パ―ティションをパ―ティションされていないテ―ブルに/します。これは、がDDLであるため、データ·セグメントのデ―タのな「」「...」または「の...」とはなりますをにしますパ―ティションはデ―タDMLきなUNDO / REDOオ―バ―ヘッドではなく、のデ―タをすることなくをします。

もな

1. パ―ティションされていないテ―ブルテ―ブル "B"をテ―ブル "A"のパ―ティションにします。

テ―ブル「A」はパ―ティション「OLD_VALUES」にデ―タをまず、テ―ブル「B」はデ―タをむ

```
ALTER TABLE "A" EXCHANGE PARTITION "OLD_VALUES" WITH TABLE "B";
```

デ―タは、テ―ブル「B」のデ―タをまないからパ―ティション「OLD_VALUES」に「」され、

2. パ―ティションをパ―ティションテ―ブルにする

テ―ブル「A」はパ―ティション「OLD_VALUES」のデ―タをみ、テ―ブル「B」はデ―タをまない

```
ALTER TABLE "A" EXCHANGE PARTITION "OLD_VALUES" WITH TABLE "B";
```

デ―タはパ―ティション「OLD_VALUES」のデ―タをまないからテ―ブル「B」に「」され、

このにはさらにくのオプション、、があります

については、このリンクをしてください。---> "

https://docs.oracle.com/cd/E11882_01/server.112/e25523/part_admin002.htm#i1107555 "セクション「パーティションの」

オンラインでのパーティションをむ https://riptutorial.com/ja/oracle/topic/3955/のパーティション

クレジット

S. No		Contributors
1	Oracle Databaseの	Community, J. Chomel, Jeffrey Kemp, Jon Ericson, Kevin Montrose, Mark Stewart, Sanjay Radadiya, Steven Feuerstein, tonirush
2	DUALテーブル	Slava Babin
3	NULLの	Dalex, JeromeFr
4	Oracle Database 12Cでのな	Muntasir, Vahid
5	Oracle MAF	Anand Raj
6	Oracleアドバンスト・キューイングAQ	Jon Theriault
7	WITHAKAテーブルをしたサブクエリファクタリング	B Samedi, MT0
8	インデックス	smshafiqulislam
9	ウィンドウ	Leigh Riffel
10	エラーログ	zygimantus
11	キーワードまたはのり	dev
12	クエリによってされるのページネーション	Ahmed Mohamed, Martin Schapendonk, Matas Vaitkevicius, Sanjay Radadiya, tonirush, trincot
13	コンテキストの	Jeffrey Kemp
14	シーケンス	Pranav Shah, SriniV
15	ジョイン	Aleksej, B Samedi, Bakhtiar Hasan, Daniel Langemann, Erkan Haspulat, Pranav Shah, Robin James, SriniV, Sumner Evans
16	データベースリンク	carlosb, Daniel Langemann, g00dy, kasi

17	データポンプ	Vidya Thotangare
18	データ	Mark Stewart, Pancho, Slava Babin
19	ヒント	Aleksej, Florin Ghita, Jon Heller, Mark Stewart, Pirate X
20	リアルアプリケーションセキュリティ	Ben H
21	レコードをするさまざまな	nimour pristou, Nogueira Jr, SriniV
22	レベルクエリ	Sanjay Radadiya, TechEnthusiast
23		SSD
24	SQL	Dmitry
25	られたの	Arkadiusz Łukasiewicz, MT0
26	PL / SQLブロック	Jon Heller, Skynet, Zohar Elkayam
27		carlosb, Eric B., Florin Ghita, Francesco Serra, J. Chomel, J.Hudler, Jeffrey Kemp, Mark Stewart, SriniV, Thunder, walen, zhliu03
28		carlosb, MT0, Roman, tonirush
29	の	David Aldridge, Florin Ghita, Jeffrey Kemp, Mark Stewart, tonirush, zygimantus
30	による	mathguy
31		Evgeniy K., Matas Vaitkevicius, ppeterka, Pranav Shah
32	トランザクション	phonetic_man
33	のパーティション	BobC, carlosb, ivanzg, JeromeFr, Kamil Islamov, Stephen Leppik, tonirush