1. **Lệnh trace function chưa commit;**

select t.inst\_id

,s.sid

,s.serial#

,s.username

,s.machine

,s.status

,s.lockwait

,t.used\_ublk

,t.used\_urec

,t.start\_time

from gv$transaction t

inner join gv$session s on t.addr = s.taddr;

alter system kill session 'sid,serial#';

alter system kill session '1353,36856'**;**

1. **Check lock object in oracle ( using in table, pkg)**

SELECT s.inst\_id, s.sid,s.serial#,p.spid,s.username,s.program,s.SQL\_ID,s.INST\_ID

,'ALTER SYSTEM KILL SESSION '''||s.sid||','||s.serial#||',@'||s.INST\_ID||''' IMMEDIATE;'

,'ALTER SYSTEM DISCONNECT SESSION '''||s.sid||','||s.serial#||',@'||s.INST\_ID||''' IMMEDIATE;'

FROM gv$session s

JOIN gv$process p ON p.addr = s.paddr AND p.inst\_id = s.inst\_id

WHERE s.type != 'BACKGROUND' and s.SQL\_ID is not null

order by s.username,s.program,s.SQL\_ID

;

SELECT B.Owner, B.Object\_Name, A.Oracle\_Username, A.OS\_User\_Name ,a.SESSION\_ID,

'alter system kill session '''||s.SID||','||s.serial#||',@'||s.INST\_ID||''' IMMEDIATE;',

s.SQL\_ID

FROM V$Locked\_Object A, All\_Objects B, gv$session s

WHERE A.Object\_ID = B.Object\_ID

and a.SESSION\_ID = s.SID

--and B.Object\_Name = 'PROM\_HISTORY\_COUNT'

;

1. **Check table space:**

SELECT df.tablespace\_name tablespace,totalusedspace use\_mb,(df.totalspace - tu.totalusedspace) free\_mb, df.totalspace total\_mb,ROUND(100 \*((df.totalspace - tu.totalusedspace)/df.totalspace)) free\_ps FROM (SELECT tablespace\_name,ROUND(SUM(bytes) / 1048576) TotalSpace FROM dba\_data\_files GROUP BY tablespace\_name) df,(SELECT ROUND(SUM(bytes)/(1024\*1024)) totalusedspace,tablespace\_name FROM dba\_segments GROUP BY tablespace\_name) tu WHERE df.tablespace\_name = tu.tablespace\_name order by df.tablespace\_name

1. **Index trong Oracle:**

1. Cách chọn index

Index là một tính năng cực kì quan trọng khi thao tác với cơ sở dữ liệu. Việc lựa chọn index sao cho hợp lí phụ thuộc phần lớn vào số lượng dữ liệu và môi trường cài đặt của cơ sở dữ liệu.

Tuy nhiên cũng có một vài quy tắc cơ bản khi lựa chọn index trong Oracle như sau:

* Chọn những column được dùng nhiều trong mệnh đề WHERE.
* Chọn những column được dùng nhiều để JOIN các bảng.
* Chọn những column có selectivity cao. Selectivity của index là phần trăm của số row trong bảng có giá trị unique, trên tổng số record của bảng. Sẽ tốt hơn với index khi có ít row có cùng dữ liệu (selectivity tiến gần đến 1). (Note: Oracle tự động tạo hoặc dùng index có sẵn với các column mà có constraint là unique, hoặc primary key). Việc chọn index những column có selectivity thấp hữu ích trong trường hợp ta muốn một hoặc hai giá trị được lựa chọn ít hơn các giá trị khác.
* Không dùng standard B-tree index với những cột có ít giá trị khác biệt. Những cột này thường có selectivity thấp nên performance sẽ không hiệu quả.
* Không chọn những cột mà giá trị thường xuyên được update. Với những câu lệnh UPDATE, INSERT, DELETE thì index cũng sẽ được maintain nên việc có index sẽ làm câu lệnh mất nhiều thơì gian hơn để thực thi so với không có index.
* Không dùng những index column chỉ xuất hiện trong WHERE với function hoặc operator. Câu điều kiện WHERE sử dụng function (ngoại trừ MIN hoặc MAX), hoặc operator sẽ không thể cung cấp access path đến index đó, ngoài trừ các index dạng function-based.
* Xem xét việc chọn index những foreign key hoặc các key sử dụng để JOIN trong trường hợp các cấu UPDATE, INSERT, DELETE, access cả bảng chính và bảng phụ. Những index này cho phép UPDATE, DELETE ở bảng chính mà không share locking bảng phụ.

2. Lưu ý khi chọn index

* Khi lựa chọn index, nên xem xét xem liệu phần performance đạt được khi query có đáng với phần performance mất đi khi INSERT, UPDATE, DELETE, và phần tài nguyên phải dùng để lưu trữ index. Bạn nên so sánh perfomance giữa trường trường hợp có/không có index để đưa ra được phương án tốt nhất.
* Khi lựa chọn composite index (index được tạo nên bởi 2 hoặc nhiều column khác nhau), tổ hợp của các column có selectivity thấp có thể tạo nên index có selectivity cao. Ngoài ra cũng nên xem xét lựa chọn các column thường được chọn làm điều kiện query ở phần WHERE, hoặc các column dùng để JOIN, để làm composite key.

# **CRONTAB IN LINUX**

This tutorial shows you how to use the crontab command on Oracle Linux.

Background

Oracle Linux can run programs automatically as scheduled tasks or jobs. You can either schedule programs to run as system-level tasks by editing cron configuration in /etc/cron\*; or you can schedule programs to run as jobs within your user crontab. If you do not have system administrator access on a system or the programs that you wish to schedule are specific to your user account, the crontab utility provides a good mechanism to run programs on a regular schedule under your own user credentials. The crontab allows you to schedule jobs to run as often as every minute or as infrequently as once a year; however you should note that cron job will not run if the system is down during the time that the job was scheduled to run.

This tutorial is targeted at Oracle Linux 8 users, but the crontab is also available for Oracle Linux 7 and the contents of this tutorial should equally apply on that platform. Note that cron configuration is standard on most Linux systems and has been available for Unix-like systems since 1975. More complex scheduling solutions are included in Systemd in the form of timers and users are encouraged to explore Systemd timers in more depth to solve specific requirements. However, this tutorial provides foundational training that is more suitable for the average user.

What Do You Need?

A system with Oracle Linux installed

Objectives

In this tutorial you learn to:

List the contents of an existing crontab

Create and edit crontab entries

Access crontab output

Remove the full crontab for a user

(Hands-on Lab) Connect to the Compute Instance

**Note:** This step is specific to the Oracle provided free lab environment.

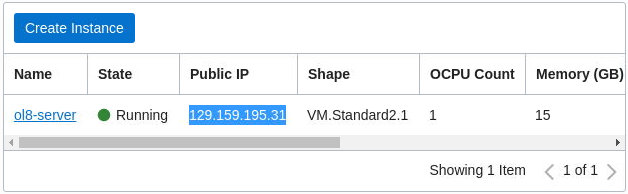
The Desktop environment will display before the instance(s) are ready. Deployment of this environment can take two to five minutes, depending on the number of resources and provisioning steps needed.

First, to access the lab compute instance(s), connect to the Oracle Cloud Console and copy the compute instance Public IP address.

Sign in to Oracle Cloud Console, and select your Compartment.

Click **Instances**.

Copy the Public IP to a temporary location (such as a text file) on your computer.



To copy, highlight the IP address with the mouse and press Ctrl+C.

Right-click the Virtual Desktop and select **Open Terminal Here**.

Connect to the instance.

Copyssh oracle@<IP\_ADDRESS\_OF\_COMPUTE\_INSTANCE>

Where <IP\_ADDRESS\_OF\_COMPUTE\_INSTANCE> is the IP address copied from the Oracle Cloud Console.

Accept the ECDSA key fingerprint by typing yes at the prompt.

You are now connected to the compute instance for this lab.

If the connection fails with the Permission denied (publickey,gssapi-keyex,gssapi-with-mic) message, wait a bit longer for the provisioning process to complete and try making the ssh connection again.

List the contents of your crontab file

Use the crontab -l command to list the contents of your crontab file.

Copy crontab -l

If no crontab entry exists for the user, the following message is displayed:

Copyno crontab for oracle

Create or edit your crontab

Use the crontab –e command to open your crontab in the default editor:

Copy crontab -e

The default editor in Oracle Linux 8 is the vim editor.

Add the following line to the end of the file to create a crontab job that runs the echo command every minute:

Copy **\*** **\*** **\*** **\*** **\*** echo "Hello World"

In vim:

You can jump to the bottom of the file by pressing ‘Shift-g’.

Hit the ‘i’ key to enter insert mode.

Enter the crontab line.

Hit the ‘Esc’ key to exit insert mode.

Enter ‘:wq’ to write the file and to quite the editor.

This step installs the crontab and makes it active immediately:

Copy crontab: installing new crontab

List the entries in your crontab to display the new entry:

Copycrontab -l

Each job appears on its own line in the crontab. The crontab has 5 space separated fields that define the different time periods that can be configured, followed by the command that is run for the job. The command can be equivalent to any command that you can run on the command line on the system, allowing you to run your own scripts or to take advantage of pipes and redirection operators.

Check your cron entry output

No Mail Transport Agent (MTA) is installed on a minimal Oracle Linux system or on an Oracle Linux platform image, by default. When no MTA is found, the cron daemon directs the output from commands specified in the crontab to the Syslog daemon. On a new install of Oracle Linux 8, you can view crontab output in the log file at /var/log/cron.

Run the following command to view the output generated by your new crontab entry:

Copysudo tail /var/log/cron

Note that the cron task only runs every minute, so you may need to wait for a minute before you run the command. The output for this cron entry should appear similar to:

CopyJun 30 12:39:02 ol8-server CROND[68564]: (oracle) CMD (echo "Hello World")

Jun 30 12:39:02 ol8-server CROND[68551]: (oracle) CMDOUT (Hello World)

This output shows the user that the command is run as, the command that was run and then shows the command output.

If an MTA, such as **postfix** or **sendmail** is installed, cron directs the output from the crontab entry to the mail spool for the user that the crontab entry runs as. This is discussed further, later in this tutorial.

Crontab time sequence options

Crontab time fields are defined in the following sequence:

minute: From 0 to 59

hour: From 0 to 23

day: From 1 to 31

month: From 1 to 12, or the name of the month

day-of-week: From 0 to 7, or the abbreviated name of day. Note that 0 and 7 both represent Sunday.

Special characters can be used within any of the time fields:

Asterisk (\*): Specifies that the command should be run for every instance of the field

Hyphen (-): Can be used to indicate a range. For example, in the day-of-week field, you might specify 1-5 to schedule the task to run from Monday to Friday, but not to run on Saturday or Sunday.

Comma (,): Can be used to specify a list of values for a field. For example, in the hour field, you might specify 6,18 to schedule the task to run at 6am and 6pm.

Forward slash (/): Can be used to specify step values. For example, you could specify \*/15 in the minutes field to schedule a task to run every 15 minutes.

The crontab also allows you to use a special shorthand extension to replace the time fields for general time requirements. Importantly, this shorthand also includes an option to run a command after reboot:

@reboot : Run once after reboot.

@yearly : Run once a year, ie. “0 0 1 1 \*”.

@annually : Run once a year, ie. “0 0 1 1 \*”.

@monthly : Run once a month, ie. “0 0 1 \* \*”.

@weekly : Run once a week, ie. “0 0 \* \* 0”.

@daily : Run once a day, ie. “0 0 \* \* \*”.

@hourly : Run once an hour, ie. “0 \* \* \* \*”.

Note that the crontab only provide 1 minute granularity. It is not straightforward to set a crontab entry that runs more frequently than every minute. If you are looking for a tool that can handle this, consider looking into Systemd timer units.

Edit the crontab to try different running commands at different time sequences

Use the crontab –e command to open your crontab in the default editor:

Copycrontab -e

Add the following entries to the crontab:

Copy**\***/15 **\*** **\*** **\*** **\*** echo "This crontab entry runs every 15 minutes"

10 **\*** **\*** **\*** **\*** echo "This crontab entry runs at 10 minutes past every hour"

30 13 **\*** **\*** 3 echo "This crontab entry runs at 13h30 on every Wednesday"

@reboot echo "This command runs once after a reboot event"

Save the crontab file and exit.

Crontab command syntax

The crontab command runs under the credentials of the user that the crontab belongs to and runs with some default environment variables set, including:

HOME: The home directory for the user that the crontab runs as

LOGNAME: Equivalent to the user name of the user that the crontab runs as

USER: The user name of the user that the crontab runs as

SHELL: The shell binary that is used to run commands. This is set to /bin/sh by default.

PATH: The path available to the crontab utility. This is set to /usr/bin:/bin by default.

You can set alternate environment variables by specifying them as lines within the crontab that do not include time fields, for example if you would prefer to use the bash shell and you want to regularly import all of the variables and settings withing your personal .bashrc configuration, you may edit your crontab to look as follows:

CopySHELL**=**/bin/bash

**\*** **\*** **\*** **\*** **\*** source $HOME/.bashrc;

Since the command run for a crontab job is executed within a shell environment, standard shell syntax applies and can be used to pipe or redirect output. For example to redirect all output to /dev/null, disabling any mailed output for the job as a result, use standard shell redirects for the STDOUT and STDERR output on the command:

Copy**\*** **\*** **\*** **\*** **\*** /path/to/command **>** /dev/null 2>&1

Alternately, you could redirect output to append to a file that you could use to track command output for each time it is run:

Copy**\*** **\*** **\*** **\*** **\*** /path/to/command **>>** $HOME/command.log

Edit your crontab to experiment with command syntax

Open the crontab in an editor using the crontab -e command.

Add the following entries:

Copy**\***/15 **\*** **\*** **\*** **\*** echo "This 'silent' crontab entry runs every 15 minutes but output is redirected to /dev/null" **>** /dev/null 2>&1

**\*** **\*** **\*** **\*** **\*** echo "The date and time is $(date +\%Y-\%m-\%d\ \%H:\%M:\%S)" **>>** $HOME/crontab.log

Note that the % characters in the date command are escaped because % characters are changed into newline characters by the crontab utility, and all data after the first % is sent to the command as standard input. See man 5 crontab for more information.

Save the file and exit.

Within a minute, a crontab.log file should appear in the $HOME directory. You can view the contents of this file by running:

Copytail $HOME/crontab.log

Access crontab output in the local mail spool

If an MTA is intaslled, all crontab output is mailed to the user’s system mail spool, by default.

**Note:** The following instructions are for example purposes only, and do describe not a complete configuration for a mail transfer agent (MTA). The configuration defaults may not be appropriate for your installation. If you are enabling postfix on a system, please also read [Oracle Linux: Install the Postfix Email Server](https://docs.oracle.com/en/operating-systems/oracle-linux/8/obe-postfix-install) for information on configuring it properly.

For crontab mail handling to work correctly, you must install, configure and enable an MTA correctly. For the purpose of this exercise, you can simply perform the following steps to enable the **postfix** MTA to handle local mail:

Install the **postfix** package:

Copysudo dnf install -y postfix

Enable and start the postfix service. The default configuration enables postfix as a local delivery MTA only:

Copysudo systemctl enable --now postfix.service

Restart the crond service, so that it starts using the postfix MTA to handle mail delivery:

Copy sudo systemctl restart crond.service

If you have not configured your crontab to redirect output to an alternate mail address or to discard output for your cron jobs, the output for each job is automatically delivered into the local mail spool for the user. The mail spool is usually stored as a single mail file that can be accessed at /var/spool/mail/$USER. If you are simply monitoring recent activity, it is easiest to tail this file to see mail as it is delivered:

Copytail -f /var/spool/mail/$USER

To review the output of each cron job in a manageable interface, install the **mailx** package and use the mail command to view the contents of the mail spool:

Copysudo dnf install -y mailx

Copymail

Mail is listed in the following way:

CopyMail version 12.5 7/5/10. Type ? for help.

"/var/spool/mail/oracle": 1 messages 1 new

>N 1 (Cron Daemon) Wed Aug 5 23:48 30/1436 "Cron oracle

&

Enter the number of the mail message to view. For example, enter 1 to view the first message in the queue:

Copy& 1

Hello World

Use the header command to return to the mail headers listing, or use the quit command to exit the application.

You can change the email address that is used to mail cron output by setting the MAILTO environment variable in your crontab, but your system **must** be configured to handle external mail delivery. If you set the MAILTO environment to an empty value, output is discarded and no mail is sent for any cron job in the crontab.

Remove the entire crontab

Use the crontab -r command to remove the entire crontab for a user.

Copycrontab -r

This command removes all cron job entries that you have specified for your user. If you only want to remove a specific job entry, use crontab -e to edit the crontab and remove the line for the job that you want to remove.

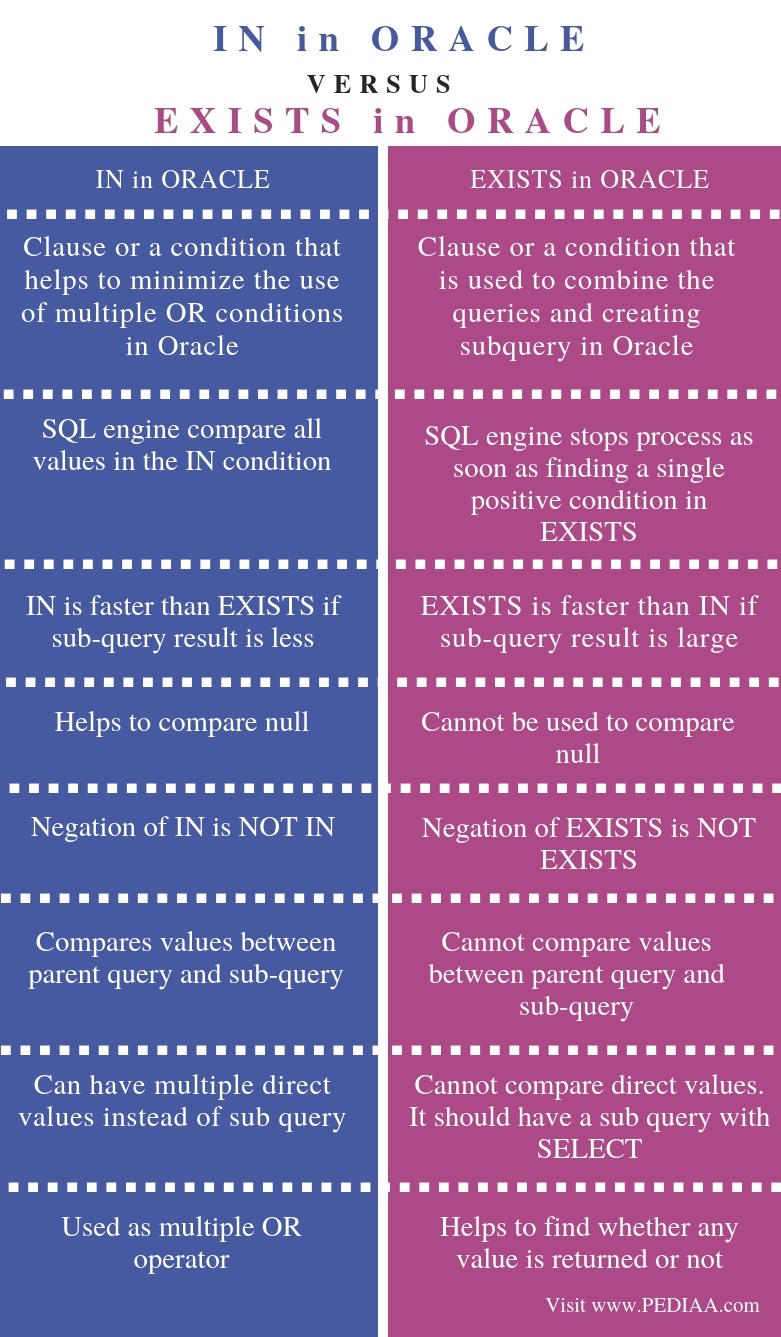
1. What is the Difference Between IN and EXISTS in Oracle

The **main difference** between IN and EXISTS in Oracle is that the **SQL engine compares all values in the IN condition while the SQL engine stops the process as soon as finding a single positive condition in EXISTS.**

[Oracle](https://pediaa.com/what-is-the-difference-between-oracle-and-db2-architecture/#Oracle%20Architecture) database is an [RDBMS](https://pediaa.com/what-is-the-difference-between-rdbms-and-ordbms/#RDBMS) designed by Oracle Corporation. It is commonly used in [data warehousing](https://pediaa.com/what-is-the-difference-between-data-mining-and-data-warehousing/#Data%20Warehousing) and online transaction processing. The new version of Oracle, which is Oracle 19c, is available on cloud or in a hybrid cloud environment. Programmers can write [SQL](https://pediaa.com/what-is-the-difference-between-sql-and-tsql/#SQL) queries to perform operations on the data stored in the database. IN and EXISTS are two commands or conditions we can use when writing SQL queries.

**Key Areas Covered**

**1.**[**What is IN in Oracle**](https://pediaa.com/what-is-the-difference-between-in-and-exists-in-oracle/#IN%20in%20Oracle)  
     -Definition, Functionality  
**2.**[**What is EXISTS in Oracle**](https://pediaa.com/what-is-the-difference-between-in-and-exists-in-oracle/#EXISTS%20in%20Oracle)  
     -Definition, Functionality  
**3.**[**Difference Between IN and EXISTS in Oracle**](https://pediaa.com/what-is-the-difference-between-in-and-exists-in-oracle/#IN%20and%20EXISTS%20in%20Oracle%20-%20Comparison%20of%20key%20differences)  
     -Comparison of Key Differences



**What is IN in Oracle**

IN is a condition in Oracle that allows filtering the data in the WHERE clause. It helps to limit the data. Furthermore, it avoids using multiple OR clauses in the SQL statement. Furthermore, the statement with IN makes the query simpler.

The syntax is as follows. The expression denotes the name of the column to obtain values.

Expression IN (value1, value2, …., value n)

For example, assume a table called student. The SQL statement with ‘IN’ is as follows.

SELECT \* from student WHERE name IN (‘Ann’, ‘Peter’, ‘Tom’);

It will provide the records that have the names Ann, Peter and Tom. In other words, it gives the values of all the columns in the table in which the names are ‘Ann’ or ‘Peter’ or ‘Tom’.

**What is EXISTS in Oracle**

The EXISTS is a condition that is used to combine queries and create subquery.

The syntax is as follows. The subquery denotes a select statement which returns at least one record.

WHERE EXISTS (subquery)

For example, table1 and table2 are two tables. The SQL statement with EXISTS is as follows.

SELECT name from table1 WHERE EXISTS (SELECT \* FROM table2 WHERE table1.id = table2.id)

It will give a list of names from table1 after executing the query inside the braces.

**Difference Between IN and EXISTS in Oracle**

**Definition**

IN is a clause or a condition that helps to minimize the use of multiple OR conditions in Oracle while EXISTS is a clause or a condition that is used to combine the queries and create subquery in Oracle.

**Process**

SQL engine compares all values in the IN condition. On the other hand, the SQL engine stops the process as soon as finding a single positive condition in EXISTS.

**Execution speed**

Moreover, IN is faster than EXISTS if the sub-query result is less whereas EXISTS is faster than IN if the sub-query result is large.

**Null**

IN helps to compare null while EXISTS cannot be used to compare null.

**Negation**

The negation of IN is NOT IN while negation of EXISTS is NOT EXISTS.

**Parent and subquery**

IN compares values between parent query and sub-query while EXISTS cannot compare values between parent query and sub-query.

**Direct values**

Furthermore, IN condition can have multiple direct values instead of a subquery. On the other hand, EXISTS condition cannot compare direct values. It should have a subquery with select.

**Usage**

IN is used as multiple OR operator whereas EXISTS helps to find whether any value is returned or not.

**Conclusion**

Oracle is a popular database management system. Programmer can write various queries to perform operations on the data stored in the databases. In brief, the two conditions available in Oracle are IN and EXISTS. The main difference between IN and EXISTS in Oracle is that the SQL engine compares all values in the IN condition while SQL engine stops the process as soon as finding a single positive condition in EXISTS.

**References:**

1.“Oracle IN – Javatpoint.” Www.javatpoint.com, [Available here](http://www.javatpoint.com/oracle-in).  
2.“Oracle EXISTS – Javatpoint.” Www.javatpoint.com, [Available here](http://www.javatpoint.com/oracle-exists).

**Image Courtesy:**

1.”The logo of Oracle Corporation” By Oracle Corporation – the German Wikipedia where it was uploaded by Afrank99 (Public Domain) via [Commons Wikimedia](https://commons.wikimedia.org/w/index.php?curid=11777903)