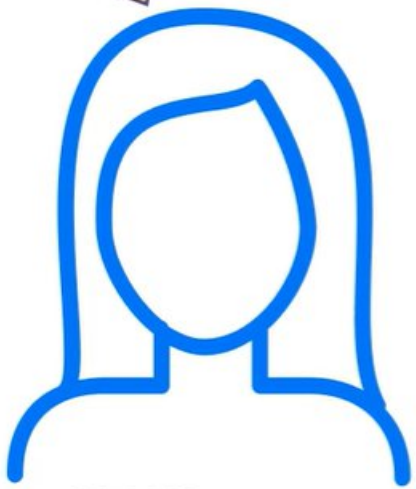
VERTICA
by opentext™

Demo – Access Policy

Fine-grained data access control

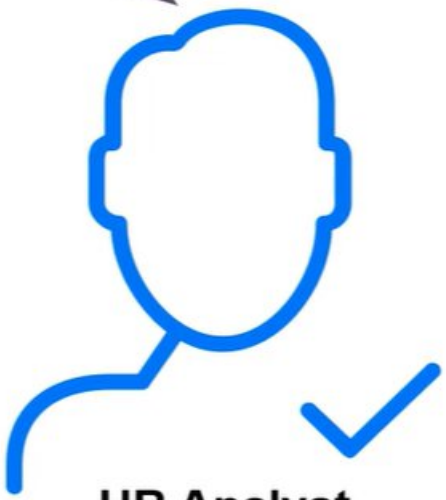
```
SELECT * FROM PEOPLE;
```



HR Manager

Name	SSN
Dave	123-45-6789
Holly	456-78-9123
Maria	314-15-9265
Kevin	667-40-8311

```
SELECT * FROM PEOPLE;
```



HR Analyst

Name	SSN
Holly	9123
Dave	6789

```
SELECT * FROM PEOPLE;
```

Developer

Name	SSN
Maria	

Column access privileges

table			
A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d2
a3	b3	c3	d3

```
CREATE ACCESS POLICY ON table FOR COLUMN column [ENABLE | DISABLE];
```

role1

table			
A	B	C	D
a1	b1		d1
a2	b2		d2
a3	b3		d3

or

role2

table			
A	B	C	D
a1	b1	X	d1
a2	b2	X	d2
a3	b3	X	d3

Column access privileges

```
CREATE ACCESS POLICY ON table FOR COLUMN column

    CASE
        WHEN ENABLED_ROLE('role1') [AND condition] THEN column_value1
        WHEN ENABLED_ROLE('role2') [AND condition] THEN column_value2
        WHEN ENABLED_ROLE('role3') [AND condition] THEN column_value3
        …
        {ELSE column_valueN}
    END

ENABLE;
```

```
CREATE ACCESS POLICY ON table FOR COLUMN column

    WHERE
        (ENABLED_ROLE('role1') [AND condition]) OR
        (ENABLED_ROLE('role2') [AND condition])
    column_value1

    WHERE
        (ENABLED_ROLE('role3') [AND condition])
        …
    column_value2

ENABLE;
```

Row access privileges

table			
A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d2
a3	b3	c3	d3
a4	b4	c4	d4

```
CREATE ACCESS POLICY ON table FOR ROWS [ENABLE | DISABLE];
```

role1

table			
A	B	C	D
a1	b1	c1	d1
a4	b4	c4	d4

role2

table			
A	B	C	D
a2	b2	c2	d2
a3	b3	c3	d3
a4	b4	c4	d4

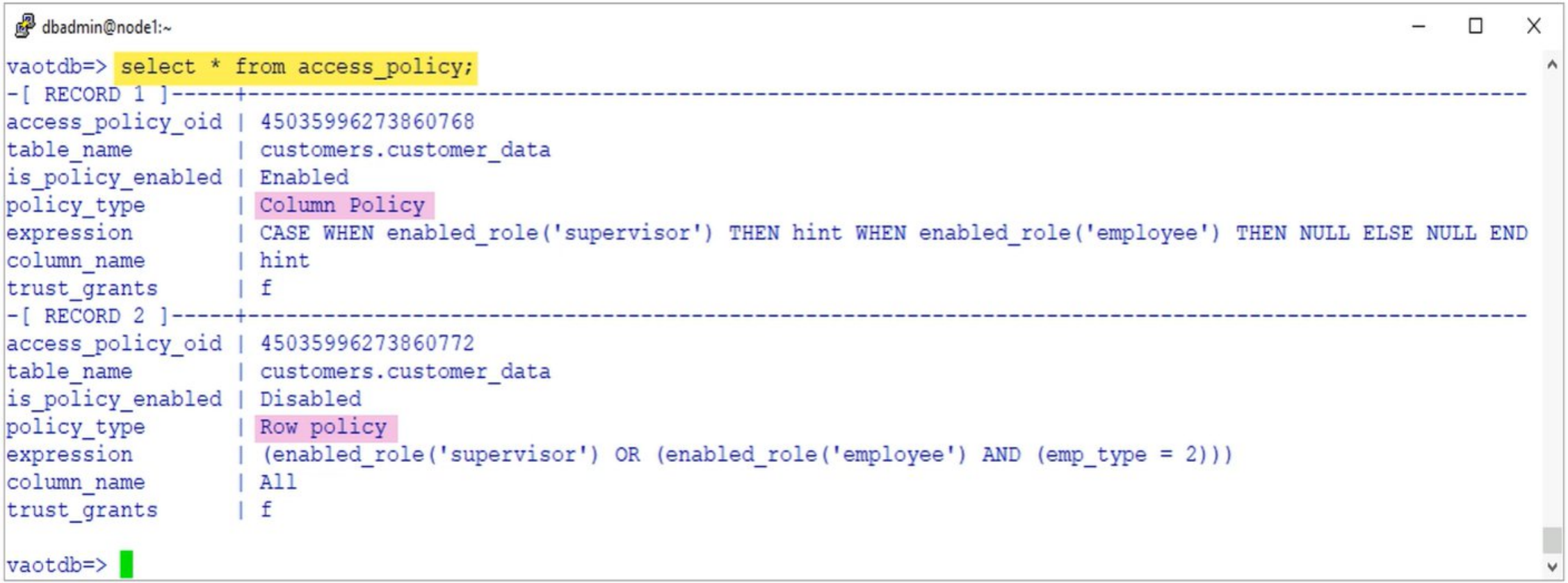
Row access privileges

```
CREATE ACCESS POLICY ON table FOR ROWS

   WHERE
      (ENABLED_ROLE('role1') [AND condition]) OR
      (ENABLED_ROLE('role2') [AND condition]) OR
      (ENABLED_ROLE('role3') [AND condition]) OR
       ...

ENABLE;
```

System table: ACCESS_POLICY



Altering access policies

Alter expression / change policy status:

```
ALTER ACCESS POLICY ON table FOR [COLUMN column_name | ROWS]
    [new expression]
[ENABLE | DISABLE];
```

Permanently remove policy:

```
DROP ACCESS POLICY ON table FOR [COLUMN column_name | ROWS];
```

Column Access Policy

```
CREATE TABLE customer_dimension (
    customer_id IDENTITY,
    customer_type VARCHAR(50),
    customer_name VARCHAR(100),
    customer_gender VARCHAR(10),
    customer_address VARCHAR(200),
    customer_city VARCHAR(100)
);
create role supervisor;
create role developer;

create user mark identified by 'password';
create user miguel identified by 'password';

grant supervisor to mark;
grant developer to miguel;

grant select on public.customer_dimension to mark,miguel;
```

Column Access Policy

```
CREATE ACCESS POLICY ON public.customer_dimension FOR COLUMN customer_address
CASE
WHEN ENABLED_ROLE('supervisor') THEN customer_address
ELSE '**************'
END ENABLE;

alter user mark default role supervisor;
\c - mark;
\c
\c - miguel;
\c
select * from access_policy;
select * from public.customer_dimension limit 10;
```

Row Access Policy

```
CREATE TABLE store_sales (
    store_id INT,
    store_region VARCHAR(100),
    manager VARCHAR(100),
    total_sales NUMERIC(18,2)
);

grant select on public.store_sales to mark,miguel;

CREATE ACCESS POLICY ON public.store_sales FOR ROWS WHERE
   (ENABLED_ROLE('supervisor')) AND (public.store_sales.manager =CURRENT_USER())
   OR ENABLED_ROLE('dbadmin')
ENABLE;

select * from store_sales;
```

Restrictions

The following limitations apply to access policies:
A column can have only one access policy.
Column access policies cannot be set on columns of complex types other than native arrays.
Column access policies cannot be set for materialized columns on flex tables. While it is possible to set an access policy for the __raw__ column, doing so restricts access to the whole table.
Row access policies are invalid on temporary tables and tables with aggregate projections.
Access policy expressions cannot contain:
Subqueries
Aggregate functions
Analytic functions
User-defined transform functions (UDTF)
If the query optimizer cannot replace a deterministic expression that involves only constants with their computed values, it blocks all DML operations such as INSERT.

Access policies and DML operations

Row access

On tables where a row access policy is enabled, you can only perform DML operations when the condition in the row access policy evaluates to TRUE.

Column access

On tables where a column access policy is enabled, you can perform DML operations if you can view the entire column in its originally defined type.

Access policies and query optimization

Projection designs

When Database Designer creates projections for a given table, it takes into account access policies that apply to the current user. The set of projections that Database Designer produces for the table are optimized for that user's access privileges, and other users with similar access privileges.

Query rewrite

The Vertica optimizer enforces access policies by rewriting user queries in its query plan, which can affect query performance. For example, the clients table has row and column access policies, both enabled. When a user queries this table, the query optimizer produces a plan that rewrites the query so it includes both policies.

Managing access policies

By default, you can only manage access policies on tables that you own. You can optionally restrict access policy management to superusers with the AccessPolicyManagementSuperuserOnly parameter (false by default):

```
ALTER DATABASE DEFAULT SET PARAMETER AccessPolicyManagementSuperuserOnly = 1;
```

Viewing access policies

1. Query system table ACCESS_POLICY.
2. Export table DDL from the database catalog with EXPORT_TABLES, EXPORT_OBJECTS, or EXPORT_CATALOG

Modifying access policy expression

```
ALTER ACCESS POLICY ON public.customer_dimension FOR COLUMN customer_address
   CASE WHEN enabled_role('dbadmin') THEN customer_address
            WHEN enabled_role('administrator') THEN customer_address
       ELSE '**************' END ENABLE;

DROP ACCESS POLICY ON public.customer_dimension FOR COLUMN customer_address;
```

Enabling and disabling access policies

Row access policies

ALTER ACCESS POLICY ON [schema.]table FOR ROWS { ENABLE | DISABLE }

Column access policies

ALTER ACCESS POLICY ON [schema.]table FOR COLUMN column { ENABLE | DISABLE }

Copying access polices

You copy access policies from one table to another as follows. Non-superusers must have ownership of both the source and destination tables:

ALTER ACCESS POLICY ON [schema.]table { FOR COLUMN column | FOR ROWS } COPY TO TABLE table

opentext™

Thank you