From Data Sharing to Clean Rooms

Exploring Collaboration Options in Snowflake

Tom Manfredi
Senior Partner Sales Engineer
January 2025

What you will learn…

- Common governance challenges
- Fundamentals of Secure Data Sharing in Snowflake
- Spectrum of Privacy Options in Snowflake Collaboration
  - Simple Sharing w/RBAC
  - Privacy with Views
  - Dynamic/Conditional Data Masking
  - Tag-based Policies and Propagation
  - Row Access Policies
  - Projection Policies
  - Differential Privacy
  - Data Clean Rooms
- How to best balance collaboration goals
  - Analytic Value / Privacy / Ease of Use

DATA GOVERNANCE CHALLENGES

Data Is Everywhere

Must be able to eliminate silos inside and outside your organization

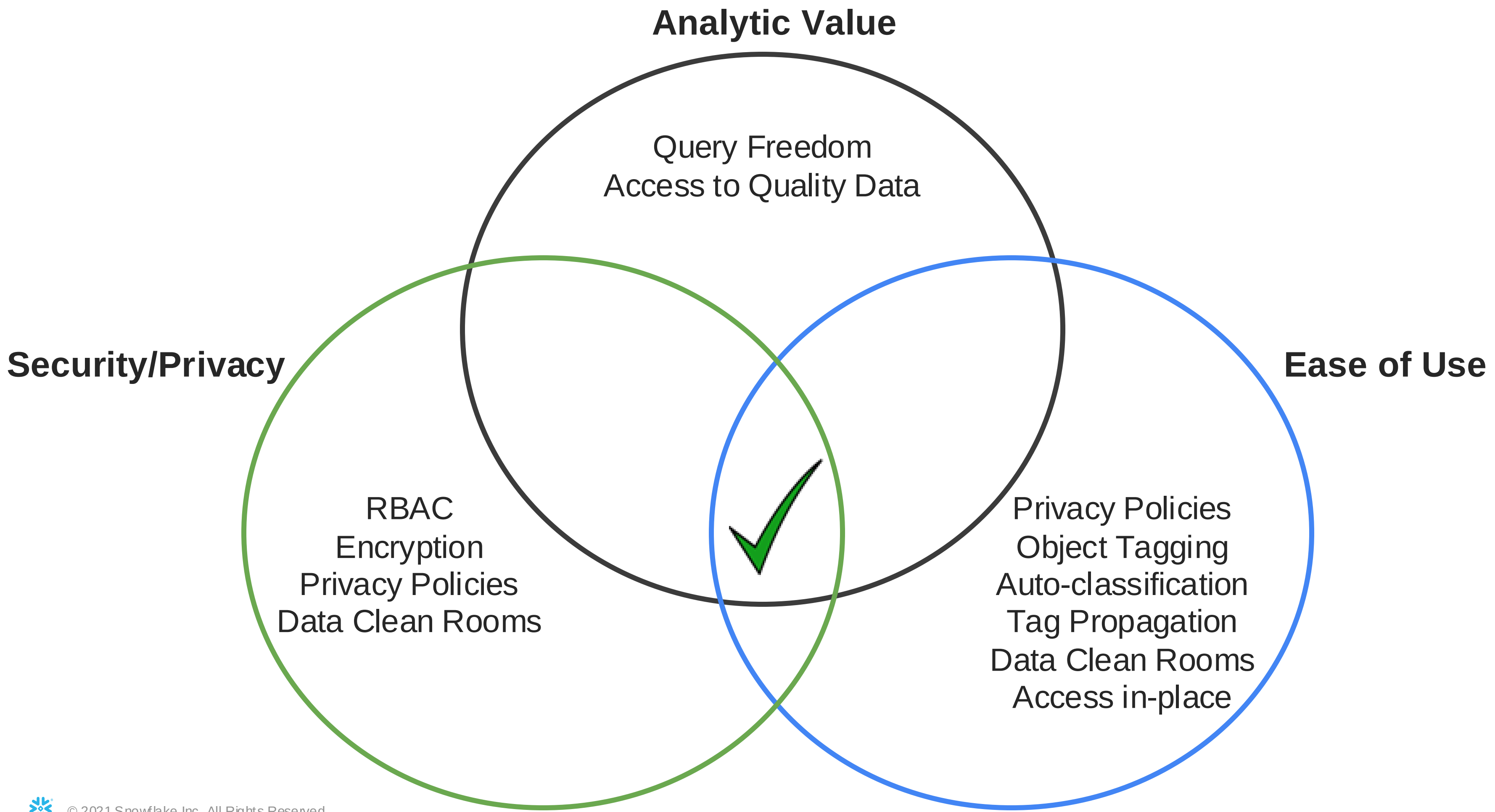
Managing Data Is Unnecessarily Complex

Knowing what your data is — and how it is being used — is hard

Security and Governance Are Inherently Rigid

Requires a flexible approach to managing risk, regardless of workload

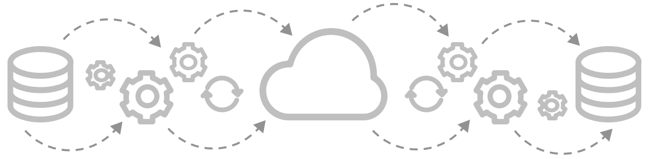
Goal is to balance…



Snowflake Makes Collaboration Easy

Traditional Methods

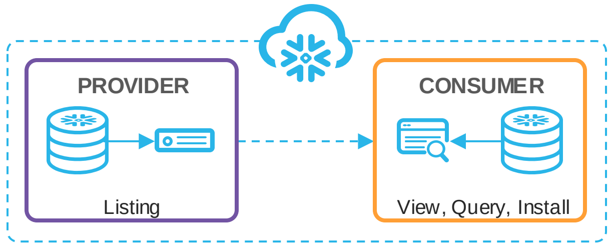
FTP | APIs | ETL | Federated Protocols



- ✗ **Costly to maintain** data pipeline infrastructure to share beyond a single cloud region, delaying access and exposing governance risk
- ✗ **Heavy data wrangling** required to train and distribute AI Models
- ✗ Integrating SaaS requires copying data to the app; **creating new silos**

Snowflake

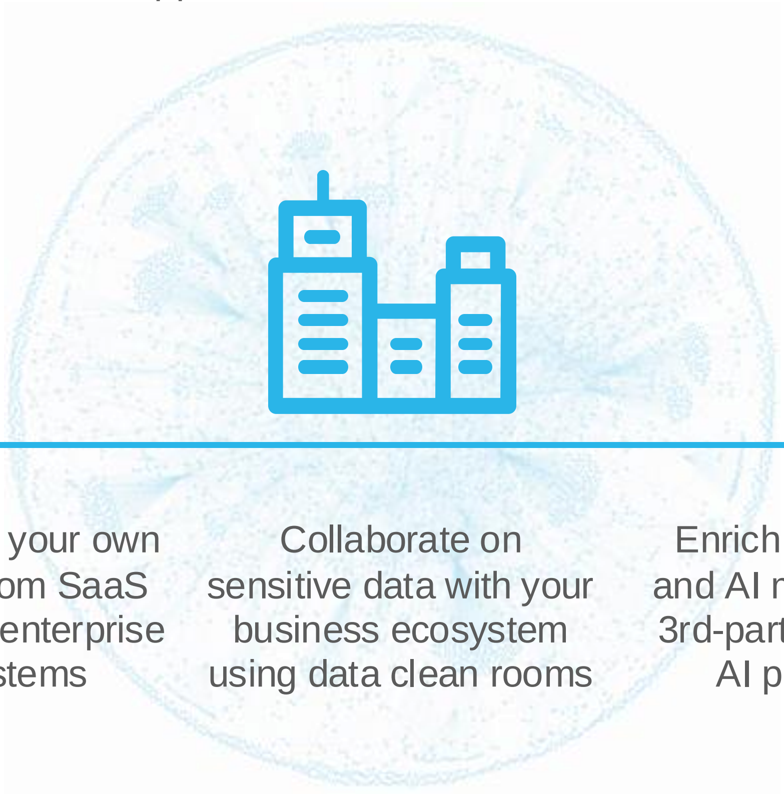
Privacy-Preserving Collaboration



- ✓ Share a **single managed copy** of data across cloud region; no ETL or copies giving you fast delivery and control with insights derived from shared data
- ✓ Direct access to governed data for AI model training and **easy distribution** to other teams, partners and customers
- ✓ Securely install apps directly into your Snowflake account to **bring the code to your data**

Collaboration in The AI Data Cloud

Discover, Share & Monetize Data, Apps and AI Products Across Clouds



Within organization ⟷ **Across organizations**

Share data, apps and AI models among teams and business units

Access your own data from SaaS apps & enterprise systems

Collaborate on sensitive data with your business ecosystem using data clean rooms

Enrich your data and AI models with 3rd-party data and AI products

Deliver and sell data and AI products via Snowflake

We start with a table…

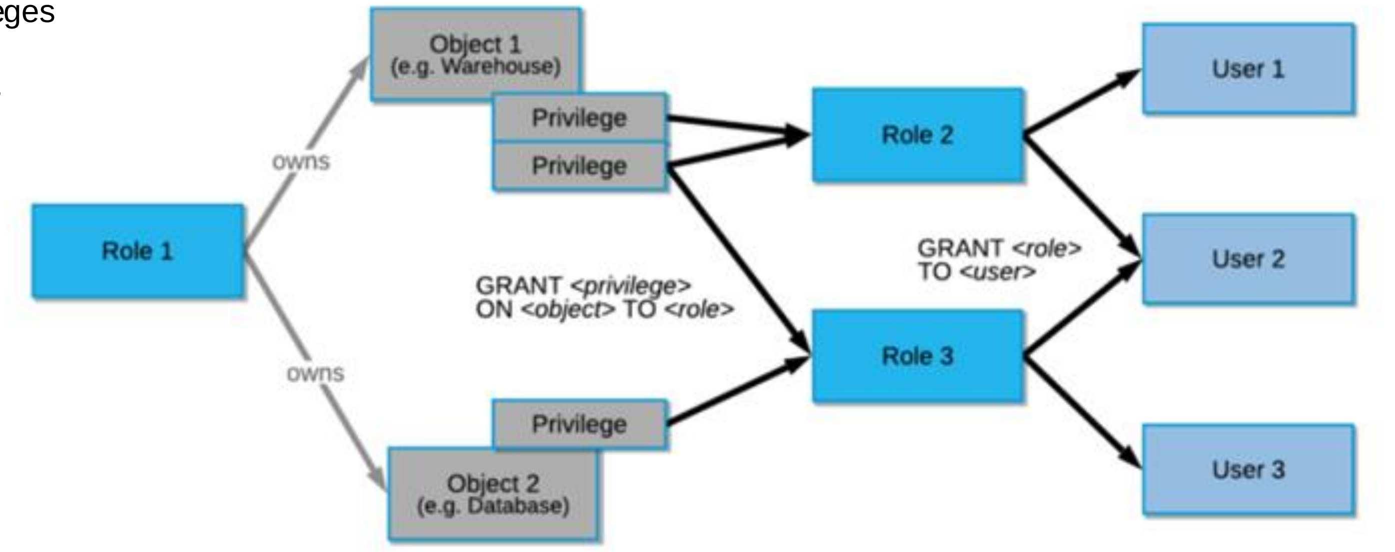
Customer

name	gender	age	zip_code	phone
John Smith	male	39	79007	123-555-1234
Jane Doe	female	50	77001	333-555-1236
Mary Taylor	female	46	77020	222-333-1111
Gene Marshall	non-binary	48	77042	555-555-1234
Michael Gaines	male	75	79003	666-666-1357

- The table has been instantiated from the encrypted, at-rest files (micro-partitions)
- The information in the table is opaque to Snowflake
- “Clear text” data is only visible to
  - Authenticated users to Snowflake
  - Assigned to an authorized role

Role-based Access Control (RBAC)

- Objects (data, applications, models…)
- Privileges
- Roles
- Users



Access privileges are assigned to roles, which are in turn assigned to users.

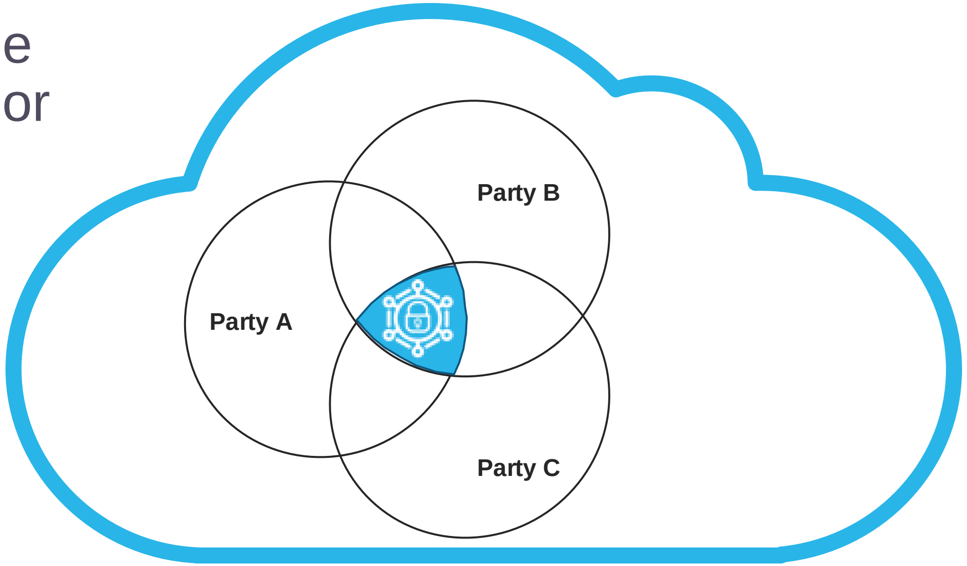
RBAC can help us down to the table/view level. Beyond that, we need something else

DEMO

Data Clean Rooms

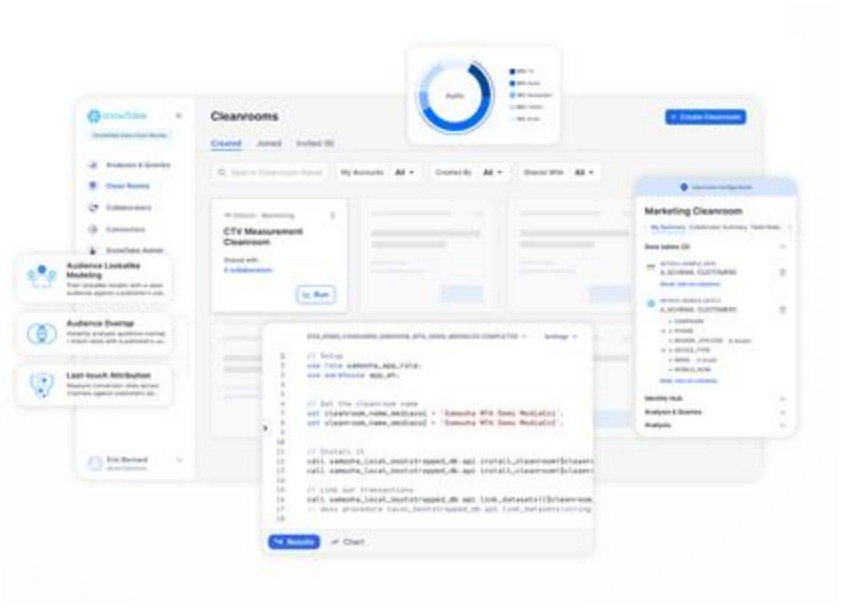
Data Clean Rooms

A secure environment where multiple parties can collaborate on sensitive or regulated data without exposing or moving the underlying data



Snowflake Data Clean Rooms

An app that makes data clean rooms easy to use & customizable for non-technical and technical users respectively.



- Allows for **secure collaboration on sensitive data** without exposing the underlying data
- Requires *no data movement*
- Snowflake Native App **available for free** on Snowflake Marketplace
- Enables **business users** with **ready-to-use industry-specific templates**
- **Customizable templates** using developer APIs

Snowflake Differentiators

Native Applications

Secure Data Sharing

Snowpark

Row Access Policies

Stored Procedures

Snowgrid

Conclusions: Balancing

Analytic Value
Security/Privacy
Ease of Use

Ease of use

Easy & flexible

More Complex

Level of trust Provider/Consumer

Trusted

Untrusted

Secure Data Sharing

Easy to setup and maintain
Provider can **audit** and **monitor** usage

+

Secure Views

The audience is trusted to be able to access and use data “freely”

All audiences

+

Provides **conditional** control over attributes (**columns**) and **rows** exposed to audience

Policy Constrained access

Row Access Policies
Dynamic Data Masking
Conditional Masking
Projection Policies
Aggregation Policies
Differential Privacy

No protection against (non friendly) sophisticated inspection

Internal & trusted audience

Flexible control over how data can be viewed (**projected**) and **aggregated** without pre-defined access (**templates**)

Protects against misuse or accidental misuse of the data - keeps usage within established boundaries

“Friendly” audience

+

Advanced setup and usage based on pre-defined templates

Pre-Defined Access

(templated)
Custom Native App
Data Clean Room

Protects against complex Data Clean Room “attacks”

External, semi- or untrusted audience

snowflake

THANK YOU

Backup Slides

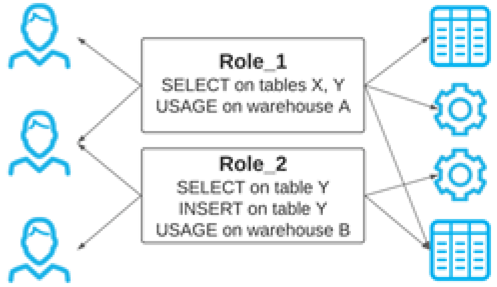
RBAC, DAC, Views, UDFs

RBAC & DAC

Views & UDFs

name	age	zip_code
J Smith	3-	790**
J Doe	5-	770**
M Taylor	4-	770**
M Gaines	7-	790**

name	gender	age	zip_code	phone
John Smith	male	39	79007	123-555-1234
Jane Doe	female	50	77001	333-555-1236
Mary Taylor	trans-fem	46	77020	222-333-1111
Gene Marshall	non-binary	48	77042	555-555-1234
Michael Gaines	male	75	79003	666-666-1357



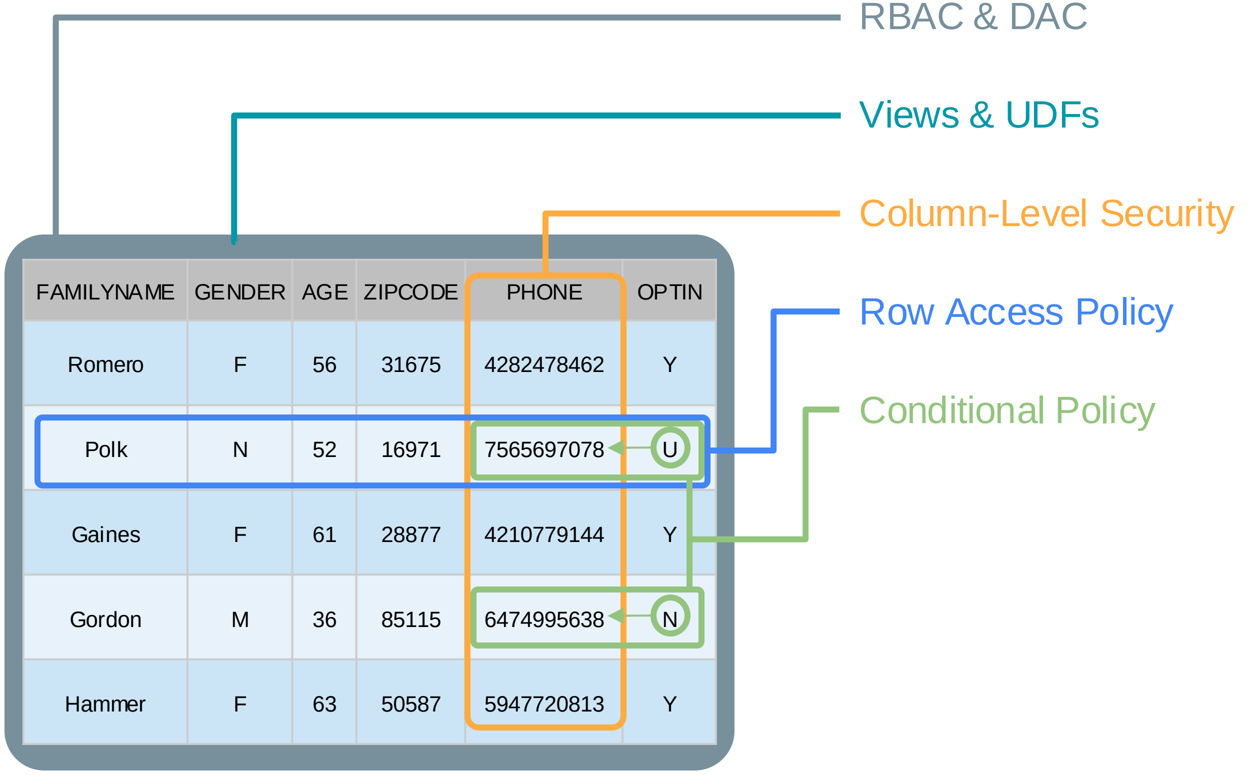
RBAC & DAC Protect the table

- Every object in Snowflake is subject to these controls, and they are at the whole-object level
- RBAC inheritance and other RBAC features apply
- The customer controls RBAC completely
- DAC (Discretionary Access Control) applies to the role that owns the object, unless the object is subject to Managed Schema Access

You may also create Views & UDFs

- These are mostly used to redact or transform rows, columns, or even cells, and create a new object
- The new object has RBAC and DAC controls

Column-Level Security / Row Access Policy



FAMILYNAME	GENDER	AGE	ZIPCODE	PHONE	OPTIN
Romero	F	56	31675	4282478462	Y
Polk	N	52	16971	7565697078	U
Gaines	F	61	28877	4210779144	Y
Gordon	M	36	85115	6474995638	N
Hammer	F	63	50587	5947720813	Y

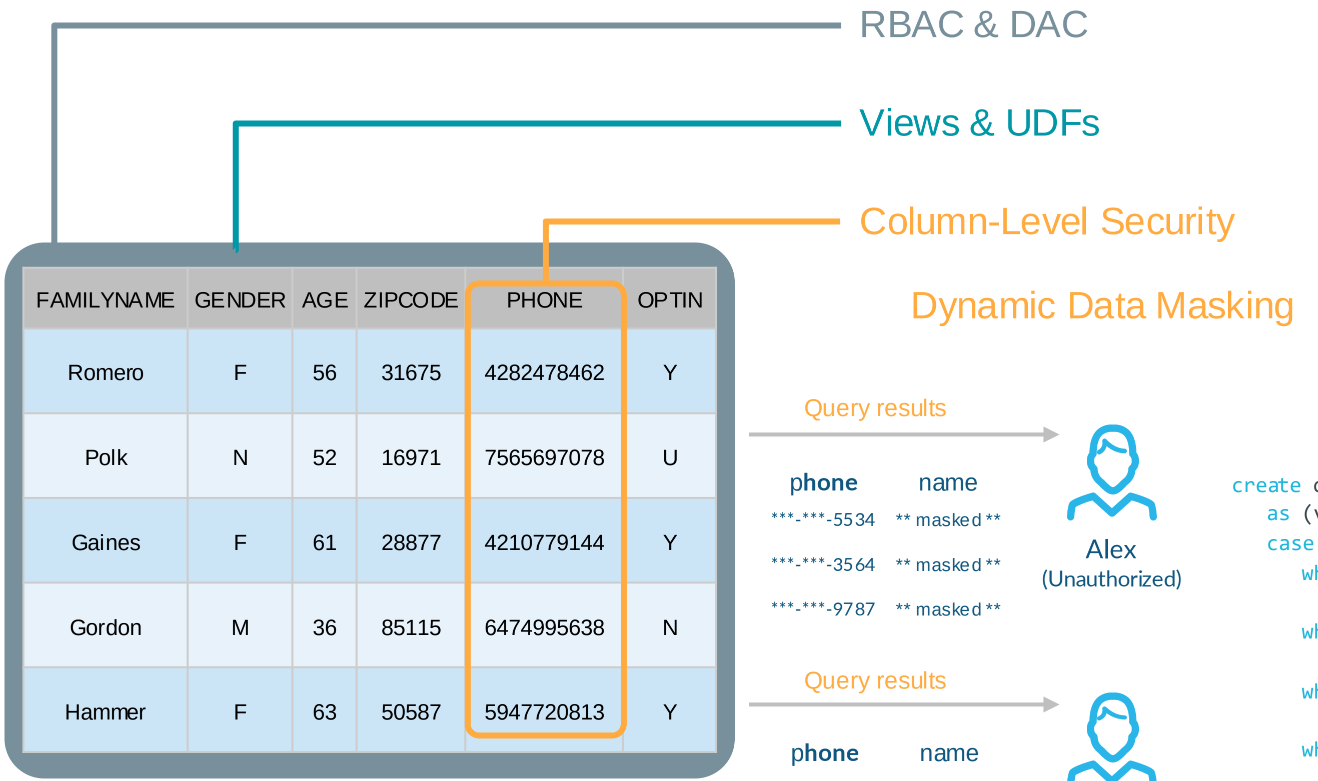
We can use Policy controls for Columns and Rows

- Prevent View/UDF explosion
- Table/View owners and privileged users (such as ACCOUNTADMIN) unauthorized to data by default
- Ensure controls are applied in any context where the object’s data is used

We get more ease of management

- Centrally manage policies
- Apply a single policy to multiple tables
- Built-in separation of duty: policy admins assign and users are subject to policy controls
- All application and use is fully audited

Dynamic Data Masking



FAMILYNAME	GENDER	AGE	ZIPCODE	PHONE	OPTIN
Romero	F	56	31675	4282478462	Y
Polk	N	52	16971	7565697078	U
Gaines	F	61	28877	4210779144	Y
Gordon	M	36	85115	6474995638	N
Hammer	F	63	50587	5947720813	Y

We can leverage Column-level Security to dynamically mask data at query time

- No change to the stored data
- Mask or partially mask using constant value, hash, and custom functions
- Unmask for authorized users only

Query results

phone	name
--5534	** masked **
--3564	** masked **
--9787	** masked **

Alex
(Unauthorized)

Query results

phone	name
408-123-5534	** masked **
510-335-3564	** masked **
214-553-9787	** masked **

Morgan
(Partially Authorized)

```
create or replace masking policy FOO
   as (val string) returns string ->
   case
      when is_granted_to_invoker_role('SEECLEAR')
           then val
      when current_role('ONLYPART')
           then regexp_replace(val, '[0-9]', '*', 7)
      when is_role_in_session('CRYPTO')
           then decrypt_raw(val, keyval, IV, ...)
      when is_role_in_session('BESPOKE')
           then user_defined_Func(val, baz, ...)
      else '** masked **'
   end;
```

```
alter table start_with_a_table modify column PHONE
      set masking policy FOO;
```

External Tokenization

RBAC & DAC

Views & UDFs

Column-Level Security

External Tokenization

FAMILYNAME	GENDER	AGE	ZIPCODE	PHONE	OPTIN
Romero	F	56	31675	awiufyaf873fg	Y
Polk	N	52	16971	78ybhsbbcvzd	U
Gaines	F	61	28877	984iuwrfgjffsss	Y
Gordon	M	36	85115	ciudsjhciasudg	N
Hammer	F	63	50587	8347ryfgvgshf	Y

Ingest protected (PII/PHI) data as Externally Tokenized

- Using tokenization provider functionality upstream from Snowflake

De-tokenize for authorized users at query time

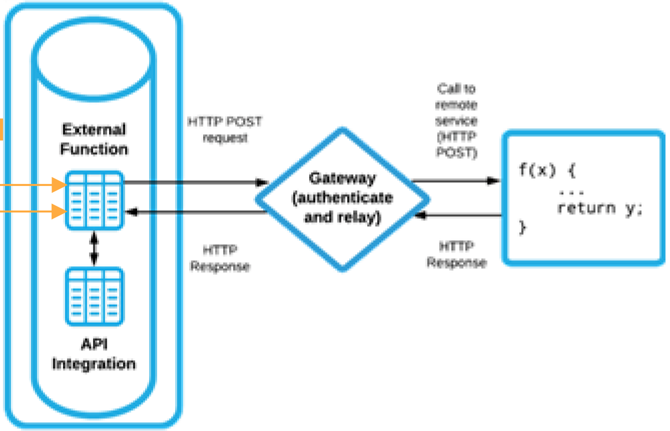
- The tokenization provider is called using a Snowflake External Function to de-tokenize data
- For unauthorized users, third-party service is not called
- Can be used in policy or outside

Example using policy:

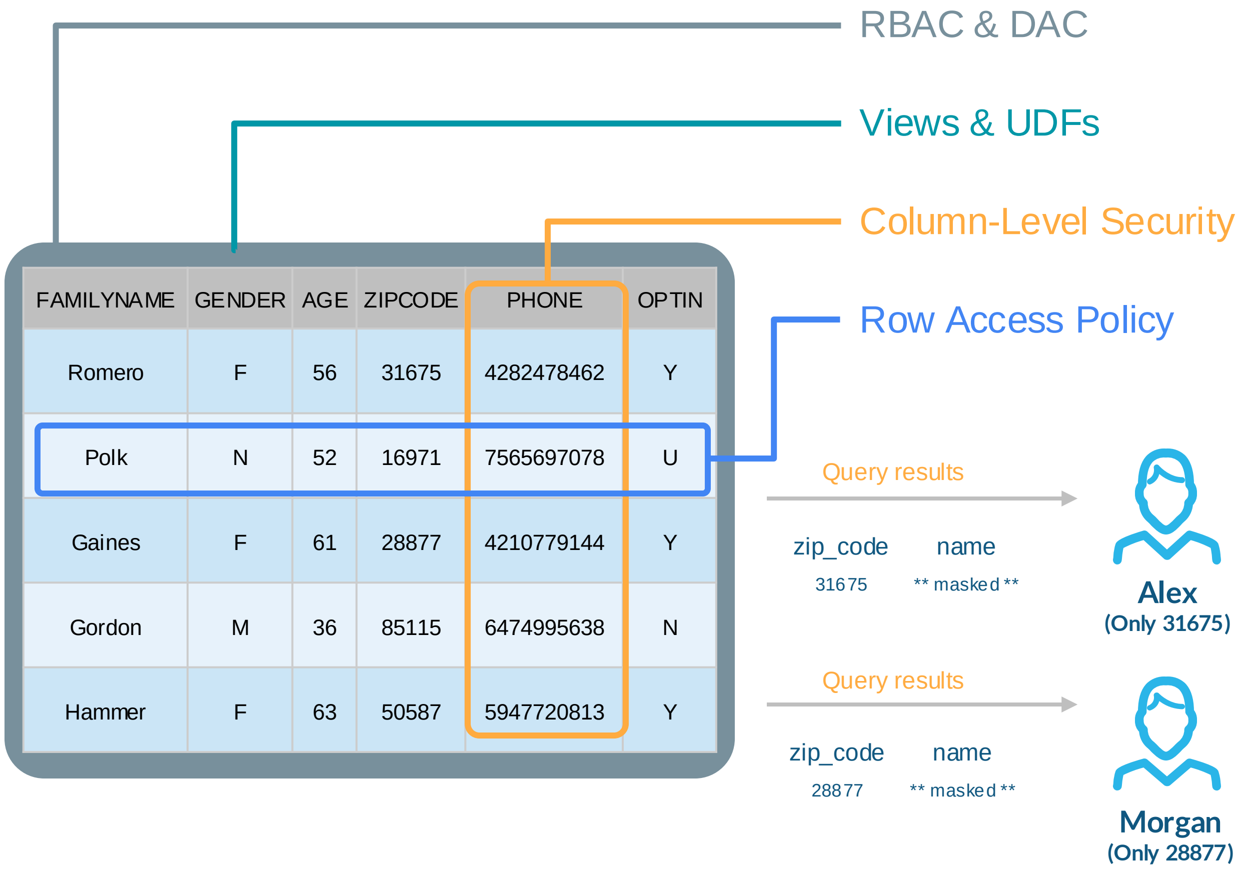
```
create or replace masking policy BAR as (val
string) returns string ->
case
   when is_granted_to_invoker_role('SEETOKENS')
     then val
   when current_role('GETREAL')
     then detok_ext_func(val, current_user(),…)
   else '** masked **'
end;
```

Example using SQL outside policy:

```
SELECT detok_ext_func(T1.phone) AS REAL_PHONE
      ,T1.GENDER
      ,T2.ZIP
  FROM T1
       JOIN T2
         ON T2.PHONE = T1.PHONE
;
```



Row Access Policy



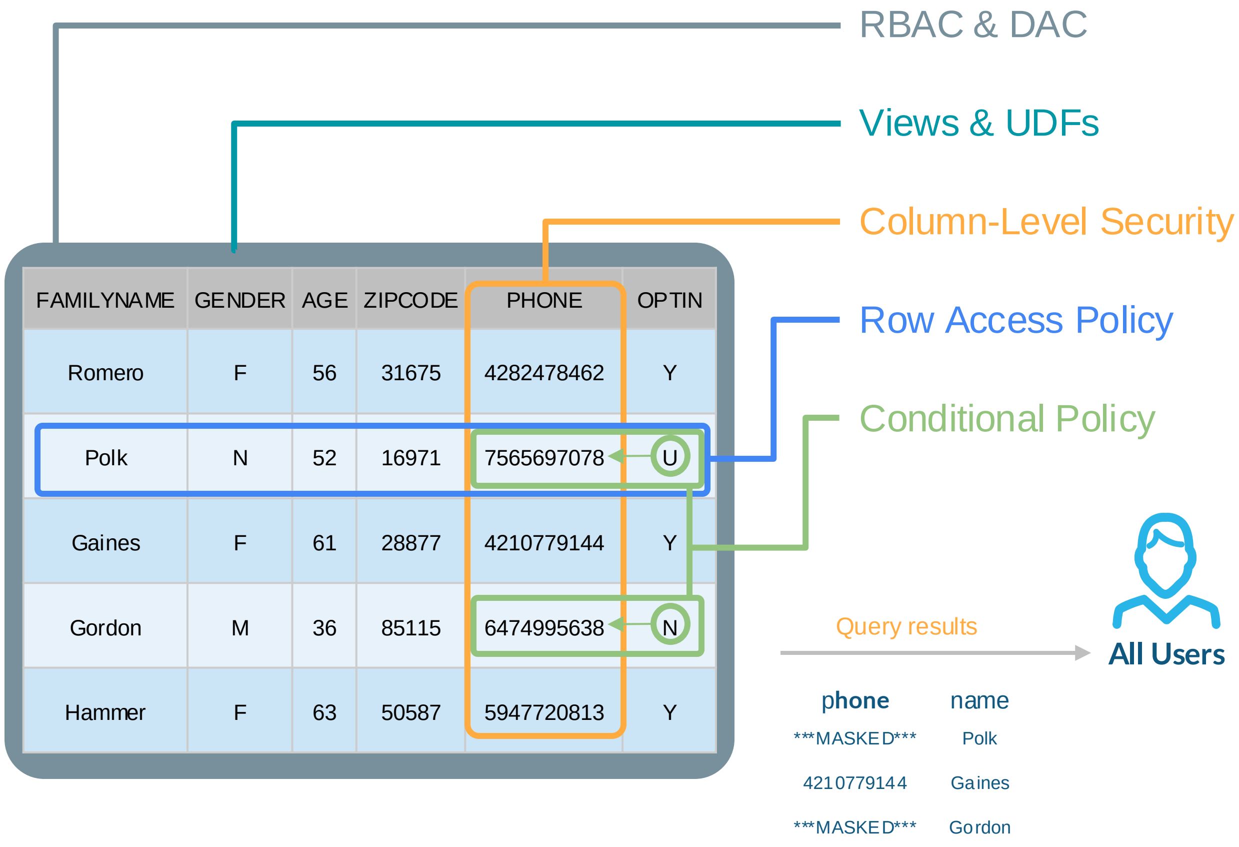
FAMILYNAME	GENDER	AGE	ZIPCODE	PHONE	OPTIN
Romero	F	56	31675	4282478462	Y
Polk	N	52	16971	7565697078	U
Gaines	F	61	28877	4210779144	Y
Gordon	M	36	85115	6474995638	N
Hammer	F	63	50587	5947720813	Y

Filter rows at query time based on user role and lookup table

- Policy contains condition(s) to allow or filter out rows
- Policy is applied to one or more table, view, or external table in an account
- Dynamically generated predicate filters out rows the user is not authorized to see at query time
- Can be combined with other controls

```
create or replace row access policy FOO
   as (this_zip varchar) returns boolean ->
      'all_seeing_role' = current_role()
   or
      exists (
   select 1 from zip_mapping_table
   where info_reader = current_role()
   and zip_code = this_zip);
```

Conditional Policy



FAMILYNAME	GENDER	AGE	ZIPCODE	PHONE	OPTIN
Romero	F	56	31675	4282478462	Y
Polk	N	52	16971	7565697078	U
Gaines	F	61	28877	4210779144	Y
Gordon	M	36	85115	6474995638	N
Hammer	F	63	50587	5947720813	Y

phone	name
MASKED	Polk
4210779144	Gaines
MASKED	Gordon

Conditional policy is a type of Column-level Security policy to dynamically mask data at query time in *one column* based on *the value of another*

- Row level policy for column access
- No change to the stored data
- Same properties as other column level policies

```
create or replace masking policy BAZ
   as (val string, optin string) returns string ->
   case
      when optin = 'Y' then val
      else '***MASKED***'
   end;


alter table start_with_a_table modify column PHONE set
   masking policy BAZ using (PHONE, OPTIN);
```

In Dev — Private — Public — ● GA

Projection Policies

Protect values of specific columns

WHAT IS IT

Block queries that enumerate values of designated columns while allowing them in operations like filter, group, and join

WHY USE IT

Consumers can discover insights from sensitive data, while the data in designated columns is protected from exposure

HOW TO USE IT

Apply Projection Policies on sensitive columns using SQL syntax

```
CREATE PROJECTION POLICY proj_policy_ssn
AS () RETURNS PROJECTION_CONSTRAINT ->
CASE
WHEN CURRENT_ROLE() = 'ADMIN' THEN
 PROJECTION_CONSTRAINT(ALLOW => true)
ELSE
 PROJECTION_CONSTRAINT(ALLOW => false)
END;
```

```
 ALTER TABLE hr.employees.directory
 MODIFY COLUMN social_security_number
    SET PROJECTION POLICY proj_policy_ssn;
```

✗
```
SELECT social_security_number
             FROM directory;
```

✓
```
SELECT count(*)
  FROM directory AS d JOIN customers AS
s
   ON d.social_security_number =
      s.social_security_number;
```

In Dev — Private — Public — ● GA

Aggregation Policies

Protect values of individual records

WHAT IS IT

Only allow aggregate queries that have more than a minimum number of rows

WHY USE IT

Consumers can discover insights from sensitive data, while individual rows are protected from exposure

HOW TO USE IT

Specify a minimum group size in a policy and then apply it to tables and views via SQL syntax

```
CREATE AGGREGATION POLICY employees_agg_policy
AS () RETURNS PROJECTION_CONSTRAINT ->
CASE
WHEN CURRENT_ROLE() = 'ADMIN' THEN
 NO_AGGREGATION_CONSTRAINT()
ELSE
 AGGREGATION_CONSTRAINT(MIN_GROUP_SIZE=>5)
END;
```

```
ALTER TABLE employees SET AGGREGATION
POLICY employees_agg_policy;
```

```
✗   SELECT * FROM employees;

✗   SELECT count(*)
                 FROM employees
                 WHERE name='Latanya
    Sweeney';
✓   SELECT count(*)
                 FROM employees
                 WHERE
    department='Sales';
```

○ In Dev ○ Private ○ Public ● GA

Entity-level Privacy for Aggregation Policies

Protect values of individual entities

WHAT IS IT

Only allow aggregate queries that have more than a minimum number of entities, e.g., people, locations, organizations

WHY USE IT

Consumers can discover insights from sensitive data, while the information of individual entities across multiple rows is protected from exposure

HOW TO USE IT

Specify an optional Entity Key clause in an Aggregation Policy via SQL syntax

txn_date				txn_amount
2022-04-05	John Smith	39	79007	$20.19
2022-04-06	John Smith	39	79007	$13.76
2022-04-15	John Smith	39	79007	$42.03
2022-04-21	John Smith	39	79007	$20.19
2022-04-22	Travis Ortega	65	79010	$61.89

⋮

Query:

```
SELECT average(age)
  FROM purchases
  WHERE name='John Smith';
```

Without entity-level privacy, simple aggregation policy doesn't block this sensitive query

With entity-level, the query is blocked to protect John Smith

Differential Privacy Policy

Protect against re-identification and privacy attacks

WHAT IS IT

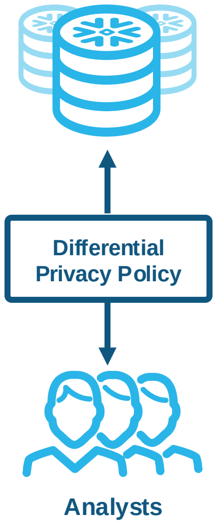
The highest standard of privacy protection: Only allow aggregate queries and add noise on sensitive data

WHY USE IT

Consumers can discover insights from granular, highly sensitive data, while protecting against re-identification and privacy attacks

HOW TO USE IT

Apply Differential Privacy Policies to tables and views via SQL syntax



Sensitive data: Full granularity, unmasked, not anonymized

✓ **Analysts can see:**

Statistics & aggregates

Feature eng. for AI/ML

✗ **Analysts can't see:**

See row-level data

Identify sensitive entities

Reverse-engineer proprietary info

In Dev · Private · Public · GA
SOON

Differential Privacy Policy: How it works

Mathematically tuned, noisy aggregates and privacy budget tracking

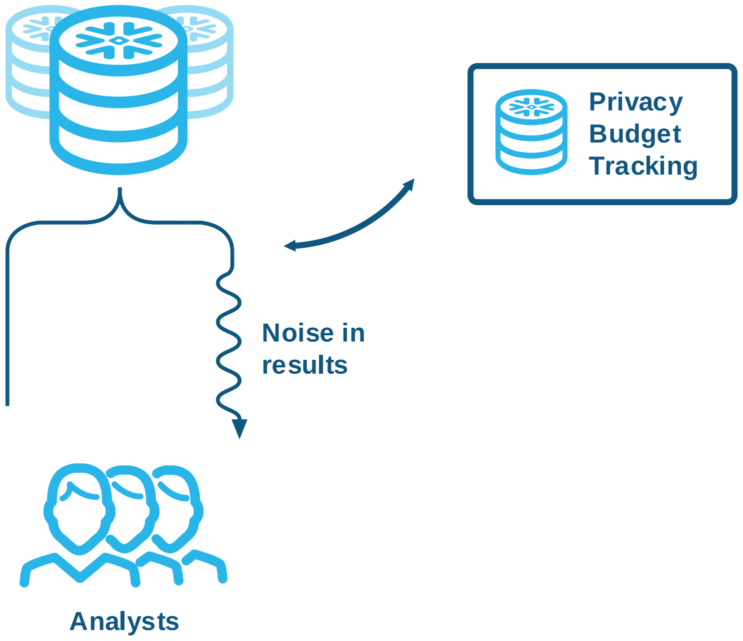
KEY ELEMENTS OF DIFFERENTIAL PRIVACY

NOISE

Noise is dynamically added to queries: more noise, higher protection

PRIVACY BUDGET

Blocks privacy attacks that occur across a history of queries. Consumers are cut off from running further queries before they can learn sensitive information


Privacy Budget Tracking
Noise in results
Analysts