Snowflake 필기(1~5)

CHAPTER 01. Intoduction

Snoflake는 SaaS이다. 즉 서비스(응용프로그램)를 클라우드로 제공한다.
데이터 저장 및 처리, 분석에 도움되며,
새로운 Sql 쿼리 엔진과 클라우드를 위해 혁신적인 아키텍처를 결합한 것.
즉 Snowflake는 다음에 강점이 있음
- Data Warehouse
- Data lake
- Data Exchange
- Data Apps
- Data Science
- Data Engineering

Snowflake 에디션은 - 표준. 엔터프라이즈. 비즈니스 크리터컬 등이 있음

Snowflake는 세 클라우드 공금자들에 의해 Snowflake를 사용할 수 있음
즉 데이터 같은 것들은 이 3 업체에서 저장하고 해야됨
1. aws
2. Azure
3. Google Cloud

US 프라이버시와 보안 표준 FIPS 140-2 및 FedRAMP 준수가 필요한 정부 기관의 경우는 AWS와 Azure만을 지원함.
또한 이런 정부 지역에 만족하는 것은 Business Critical Edition에서만 지원함

Snowflake와 연결할 때 웹 인터페이스나 콘솔을 통해 연결0 할 수 있음
물론 드라이버를 설치해야함
1. SnowSQL -〉 CLI 클라이언트로 해야됨 - MFA 지원
2. ODBC - MFA 지원
3. JDBC - MFA 지원
4. SDK(Node, Pyhon, Kafka, Spark, Go 등)

chapter 01 예상 문제

Is Snowflake available on-premise? 답) No
- 스노우플레이크는 SaaS 즉 소프트웨어를 제공하므로 온프레미스 불가능

What are the three Snowflake editions offered when you sign up for a Snowflake trial account? - standard, Enterphrise, business critical

Which Snowflake edition supports private communication between Snowflake and your other VPCs through AWS PrivateLink? - Business Critical

Which cloud provider is not supported by Snowflake?
- except AWS, Azure, Google Cloud

Does Snowflake automatically stores data in encrypted form in all editions?
- true 스노우플레이크는 추가 비용 없이 기본적으로 모든 고객의 정보를 암호화 한다

In which Snowflake edition is Tri-Secret Secure option available?
- Only Business Critical edition
※Tri-Secret Secure는 데이터를 복합 마스터 키로 관리하는 기능임.

Can we use **Multi-factor Authentication** to connect to Snowflake via the Snowflake JDBC driver? - True 당연히 이 드라이버를 설치해 사용 가능
- 더 안정적인 인증방법을 JDBC를 통해서도 연결할 수 있다.

A client has ODBC or JDBC available in their systems, but they do not have the Snowflake drivers. Is the client able to connect to Snowflake?
- False. ODBC/JDBC가 있다고 하더라도 결국 Snowflake 전용 ODBC/JDBC 드라이버를 설치해야 사용 가능하다. UI 인터페이스를 통해 다운받을 수 있음

What is the name of Snowflake's Command Line Interface tool?
- SnowSQL 이름은 SQL이지만 이는 CLI 툴이다. 뭐 SQL 쓰는 데긴 함.

멀티 클러스터는 다중 쿼리에 의한 성능 저하를 예방하기 위한 것이다.
Scale-up : 사이즈 업을 하는 거임
Scale-out : 동시성에 의한 부하를 줄이기 위해서 사이즈보다 개수를 늘리는 것

최대 스케일 아웃은 보통 10개인 듯. 근데 결국 비용에 영향을 준다.

쿼리 액셀레레이션은 자동으로 스케일업하게 만들어주는 것인데 특정 목적에만 사용함

CHAPTER 02. Architecture

Snowflake의 아키텍처는 **Shared-disk**와 **Shared-nothing**의 하이브리드 형태이다.
- Shared-disk의 특징적 요소 : 중앙 데이터 저장소를 두어 모든 컴퓨트 노드에서 접근 가능하다는 점.
- Shared Nothing의 특징적 요소 : 가상 웨어하우스(서버라고 보면됨)를 사용하여 쿼리를 처리하며, 클러스터의 각 노드가 전체 **데이터 집합의 일부**를 로컬로 저장합니다.

이 두가지 특징으로 인해 데이터의 관리 간편성(일관성일 듯) --〉 shared-disk의 장점
성능 및 확장성 ---〉 shared Nothing의 장점을 제공

Snowflake는 3가지의 층(Layer)을 가지고 있음. 각 층마다 별도 과금임.
1. Centralized Storage (중앙 집중형 스토리지) : 데이터를 넣으면, 내부에서 최적화하여 압축하여 Column 지향으로 재구성. 데이터 저장에 모든 측면 관리
2. Compute (컴퓨팅) : 쿼리 실행은 가상 DW를 통해 수행됨. 이 가상 DW는 클라우드 제공업체에서 스노우플레이크가 할당한 여러 컴퓨트 노드들로 구성된 **MPP**(Massively Parallel Processing, 대규모 병렬 처리) **Compute Cluster**임
3. Cloud Service : 스노우플레이크 전체에서 활동을 조정하는 서비스. 다음 5가지 포함

- Authentication (인증)
- Infrastructure management (인프라 관리)
- Metadata management (메타데이터 관리)
- Query parsing and optimization. Access Control (쿼리 구문 분석 및 최적화,액세스 제어)

++ Cloud Agnostic Layer (클라우드 중립 레이어) : 4번째 레이어라고도 할 수 있음
- 이는 처음 공급업체를 선택할때만 사용함. -〉 그래서 과금이랑 크게 상관X
- 이는 각 클라우드 마다 작업마다 다른 사용법이나 방법들을 추상화하여 일관된 형태(API) 등으로 제공하기 위해 존재하는 Layer와 같다.

Snowflake의 중요한 오브젝트들 (간단히 설명)
1. Account : 계정 이름은 어느 지역에서든 유일하며, 조직 내에서도 유일해야함.
- 우리가 로그인할 때 개인 어카운트 ID가 적힌 링크로 접속을 한다.

2. Warehouse : 이는 일반적인 DW가 아닌 주어진 쿼리를 실행하기 위한 가상 머신(엔진)의 집합.

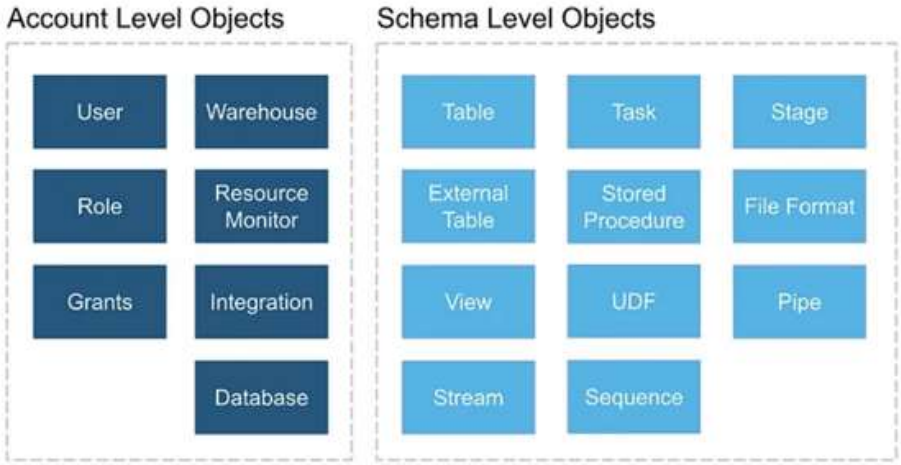
3. Database : 논리적인 스키마의 컬렉션. 이는 뭐 다른 DB와 동일하다. 다른 오브젝트를 저장하는 논리적인 컨테이너로 사용되며, 스키마들을 저장

4. Schema : 이것도 오브젝트의 논리적인 컬렉션이며, DB가 생성되면 2개의 스키마가 생성. 하나는 생성된 모든 오브젝트를 저장하는 Public 스키마이고, 다른 하나는 메타데이터 정보를 저장하는 Information_Schema

스키마에 포함될 수 있는 오브젝트들

1. 테이블 : DB내에 모든 데이터를 포함하는 DB 오브젝트
2. 뷰 : 쿼리로 정의된 가상 테이블
3. 스테이지 : 클라우드 저장소 내의 데이터 파일의 위치
4. 파일 형식 : 미리 정의된 구조로, CSV,JSON, AVRO, ORC, PARQUET 및 XML 입력 유형의 데이터를 Snowflake 테이블에 액세스하거나 로드하기 위한 스테이지된 데이터 집합을 설명하는 오브젝트
5. 시퀀스 : 시퀀스를 사용하여 고유한 번호를 생성 (카운터와 비슷)
6. **파이프** : 스테이지에서 사용 가능한 파일이 있을 때 자동으로 데이터를 로드하는 Snowflake의 특수하고 독특한 유형의 오브젝트
7. 저장 프로시저 및 사용자 정의 함수(UDF) : SQL 및 JS에서 작업을 수행하기 위해 시스템을 확장할 수 있도록 함

오브젝트는 계정 레벨 또는 스키마 레벨에 존재할 수 있음. 이 두 차이는 그 오브젝트를 저장하는 위치이다.



- Role 사용자 임의의 역할을 부여할 수 있다.

챕터 2에서는 각 3 계층의 하는 역할을 아는 것이 좋다.

chapter 02 예상 문제

The three main layers of Snowflake are… :

- centralized Storage, compute, cloud Service

Select the term that is associated with the compute layer

--〉 당연히 쿼리를 실행하는 것일 것이다. 즉, Query Processing

Which of the following services are provided by the Cloud Services Layer?

- **메타데이터/인프라 관리, 쿼리 구문 분석(파싱) 및 최적화, 액세스 컨트롤, 인증**

What statement is true about Snowflake's unique architecture?

1. One Node Shared Data
2. **Multi-Cluster Shared Data** ---〉 Shared-disk and shared Nothing의 하이브리드 형태
3. One Node Private Data

What is the storage hierarchy in Snowflake? (※hierarchy : 계층)

: Account - DB - Schema - Object

Can two Virtual Warehouses access the same data simultaneously without any contention issues?

- True : 이는 Shared-disk의 특징을 가져왔따고 볼 수 있다.

Are the interactions with data initialized through the services layer? 질문!!!

- true : 여기서 Service layer는 Cloud Service layer라는 것이고, 데이터의 상호 작용, 즉 메타데이터를 관리하는 Service 층을 통해 초기화 된다.

In which layer of Snowflake architecture is stored all security-related information?

--〉 Cloud Service : Cloud Service의 특징인 액세스 컨트롤은 다양한 보안 관리 요소를 가지고 있음.

Can the table functions in INFORMATION_SCHEMA be used to return **account-level** usage and historical information for storage, warehouses, user logins, and queries?

- **TRUE : INFORMATION_SCHEMA**는 거의 대부분의 정보를 갖고 있어 위의 말이 맞음

CHAPTER 03. pricing

snowflake의 비용은 실제 사용한 시간에 따라 측정된다.
스토리지의 양을 조절하거나, 컴퓨팅 할때의 비용은 서로 각각 (따로따로) 측정된다.

1. 저장 비용 : 저장 장치의 비용은 모든 고객의 데이터를 **압축 후의 양에 대한 평균**을 따져 비용을 산정한다. 또한 다음을 포함한다
- Data stored in tables, including historical data for Time Travel
- Fail-Safe historical data
- Internal Stages

2. 컴퓨팅 비용 : Snowflake 크레딧을 이용해 가상 웨어하우스의 비용 지불
- 이 크레딧은 Snowflake 에디션과 snowflake 계정의 지역, 공급자에 따라 비용이 다름
- 비용은 웨어하우스의 크기, 클러스터의 개수, 각 서버의 클러스터 가동 시간에 따라 결정
- 처음 키면 무조건 1분의 사용량은 지불, 그 후 초 단위로 사용량 지불
- XS S M L XL .... 6XL. 1부터 2배씩 증가.

3. Cloud Service 비용 : 이것도 크레딧을 통해 지불하지만, 컴퓨팅 크레딧 비용의 최대 10%는 무료로 제공하기 때문에 보통 이 서비스에 대한 추가 비용은 지불하지 않음

데이터 전송 비용 : 데이터를 다른 지역이나 클라우드에 복사하거나 이동시킬 때 개별의 데이터 전송 요금을 받음. 보통 같은 지역 동일한 클라우드 전송은 무료이지만, 지역이 달라지거나 클라우드도 달라지면 요금이 비싸진다.

용량 옵션 : Snowflake 서비스 구매 방법은 **on-demand** 방식과 **pre-paid** 방식이 있음
1. ON-demand : 매월 사용한 만큼의 고정 비용을 지부라는 방식.
2. pre paid : 용량을 선결제 하는 방식. 구매한 용량은 매월 사용되며, 더 낮은 가격 및 장기적인 가격 보장 등의 이점 제공

chapter 03 예상 문제

What influences Snowflake pricing?--〉 based on usage&storage

Compute cost in Snowflake depends on… (즉 쿼리 수나 복잡도에 구애X)
--〉The warehouse size and how long the warehouse runs.

What are the two major cost categories in Snowflake? -〉storage, compute

How is the data storage cost computed for Snowflake?
-〉Based on the average daily amount of **compressed** data stored.

Which type of data incur Snowflake storage cost?

1. Data Stored in permanent tables.
2. Data Stored in temporal tables.
3. Cache results.
4. Data retained for Fail-Safe & Time-Travel.

---〉 1,2,4 (테이블에 저장된 데이터, 시간마다 저장된 데이터까지 비용 정산)

Do tables with Fail-Safe turned on incur additional storage costs compared to tables where Fail-Safe is turned off?
- True : 저장비용 카테고리 확인.

Which factors influence the unit cost of Snowflake credits and data storage?]

1. Snowflake Edition.
2. Region of the Snowflake account.
3. On-Demand or Pre-Paid account.
4. Users on Snowflake.

--〉 비용에 영향을 미치는 요소 : 1,2,3

CHAPTER 04. Micro-partitions - 내부적 저장 관점

Snowflake의 모든 테이블은 자동적으로 Micro-partition으로 분할되어 저장된다.
Micro-partition은 **연속된 공간**으로 **50~500MB**의 **압축되지 않은** 데이터로, **컬럼 단위**로 구성이 된다. (즉 열의 값으로 파티션을 구분)
- 또한 이 파티션은 테이블의 물리적 구조이다.
- 마이크로 파티션은 변경할 수 없으므로, 한번 생성되면 **변경할** 수 없음.
- 대신 행이 업데이트 되면, 그 행이 포함된 마이크로 파티션이 새 마이크로 파티션으로 복사되며, 업데이트된 행이 삽입된다.
- 오래되거나 삭제될 파티션은 마킹되어 **오직 삭제 대상으로 표시**된다.

Snowflake는 효율적으로 검색 데이터를 반환하기 위해 PRUNING Process를 거치게 된다
- 이 기법은 모든 마이크로-파티션을 살펴보지 않아도 필요한 모든 데이터를 검색하여 결과를 반환하는데 시간을 많이 절약할 수 있다.
- 이 기법은 쿼리를 최적화하여 쓸모없는 데이터를 스캔하지 않거나, 데이터의 범위를 바로 줄여서 결과를 반환하는 것이다.
- 즉 자동적으로 쿼리 최적화해서 검색 결과 반환을 단축시킨다 보면됨.
- 이는 파티션 키를 이용해 검색 범위를 줄이기 때문이다.
- 스노우플레이크는 각 열 이름별로 메타데이터를 만들어서 그 열(속성)의 값들을 모아둔다.
- 그러면 Select를 사용할때 보통 열로 지정을 하니까 그 속성만 참조한여 필터링한다.

각 열 이름별로 메타데이터를 저장한다고 했는데, 그 속성에 맞는 모든 행에 대한 메타데이터를 저장하는 것이다. 이에 포함되는 것은 다음 것들이 있다.
- 각 열에 대한 값의 범위
- 고유 값의 수
- 추가 속성. -〉 효율적 쿼리 처리를 위해 사용

추가 속성에는 메타데이터 키, 파티션키, 행크기의 수, 생성 일자 및 수정일자 등이 있다.

마이크로 파티션은 데이터를 로드하는 방식으로 기록되어, 쓰기의 임계값을 초과하면 더 많은 파티션으로 분할된다.

chapter 04 예상 문제

What technique does Snowflake use to limit the number of micro-partitions retrieved as part of a query?
--〉 pruning

Which statements are correct about micro-partitions in Snowflake?

1. Contiguous units of storage
2. Non-contiguous units of storage
3. 50 and 500MB of compressed data
4. 50 and 500MB of uncompressed data
5. Organized in a columnar way

-〉 1, 4, 5

Which options are correct regarding the data that is stored in micro-partition metadata?

- the range of values for each columns in the micro-partition
- the number of distinct values
- additional properties used for both optimization and efficient quety processing

먼저 Micro-partition 의 PURNIG이 먼저 수행되고, 그 다음 열 방향으로써의 속성을 택함

CHAPTER 05. Clustering - 테이블 데이터 최적화

데이터를 정렬할 방법을 지정하는 것이다. 정확히는 그룹화가 맞을 듯.

Snowflake는 기본적으로 자연 차원(날짜나 지리적 위치)를 기준으로 정렬된다.

당연히 정렬이 안되어있으면 성능에 영향을 미치기 때문

클러스터링 메타데이터는 각 Micro-partition 내에 존재 하므로, 쿼리 처리 중 불필요한 스캐닝을 피하여 성능을 높임

테이블의 마이크로 파티션에 대해 수집되는 클러스터링 메타데이터

1. 테이블을 구성하는 총 마이크로 파티션의 수
2. 하위에서 서로 중복된 값이 들어가 있는 마이크로 파티션의 수
3. 중복되는 마이크로 파티션의 깊이

클러스터링 깊이는 테이블의 지정된 열에 대해 겹치는 마이크로 파티션의 평균 깊이(1이상)을 측정. 평균 깊이가 작을수록 지정된 열에 대해 테이블이 더 잘 클러스터링됨.

- 파티션이 없는 테이블은 클러스터링 깊이가 0 이라고 볼 수 있지만, 테이블에는 하나 이상의 마이크로 파티션이 포함되므로 이는 불가능함.

이 커맨드로 클러스터링 깊이를 알수 있음

- SYSTEM$CLUSTERING_DEPTH
- SYSTEM$CLUSTERING_INFORMATION

즉 한테이블에서 제일 많이 중첩된 파티션의 수가 클러스터링 깊이가 된다.

아무런 중첩이 없으면 클러스터링 깊이=1로, 쿼리 성능에 어떠한 영향도 없다

깊어질수록 퀄리 성능이 저하되거나 시간이 오래가 걸린다.

클러스터링 키 :**실제 데이터**를 Micro-partition에 **어떻게 정렬하여 배치할** 때 사용

- 이는 순서가 꼬였기나 광범위한 삽입 때문에 클러스터링이 저하되는 경우 유용
- 클러스터링 키는 보통 **Where/JOINS/ORDER 에 많이 쓰이는 속성(열)**에 배치된다.
- 또한 단순한 열이 아니라 특정 표현식의 부분집합이 될 수도 있음

클러스터링 키에 적합한 후보는 다음과 같다.

1. 예측한거보다 느리게 작동하거나, 눈의 띄게 성능이 저하된 쿼리.
2. 클러스터링 깊이가 매우 큰 경우

RECLUSTERING

: 가끔 클러스터된 테이블에서 DML(Insert, update, delete, merge, copy)이 수행되면, 테이블에 데이터가 덜 클러스터될 수가 있다.

- 왜냐하면 클러스터링 키를 날짜로 지정하여서 특정 날짜의 데이터를 넣으려고 하는데, 넣을 공간이 없어 새로운 Micro-partiion에 넣게된다. -〉 순서가 꼬이게됨. -〉 덜 클러스터됐다고 한다.

이를 위해 Snowflake는 최적의 클러스터링을 위한 주기적이고 자동적인 **ReCLustering**을 지원한다.

처음 기본적인 클러스터링이든 Reclustering이든 결국 데이터를 옮기거나 재정렬하는 것이기 때문에 크레딧과 스토리지 비용을 소모하게 된다.

즉,테이블의 클러스터를 유지하는 것은 매우 많은 비용이 든다.

따라서, 자주 쿼리처리하는 테이블이나, 잘 변경되지 않는 테이블의 경우 더 효율적이다.

-그래서 자동적으로 Reclustering을 하지만 깊이가 깊거나 성능저하가 있는 경우만 알아서 해주는 것이다. 필요하면 내가 Rclusting을 요청할 수 있다.

chapter 05 예상 문제

What techniques would you consider to improve the performance of a query th at takes a lot of time to return any result?

1. Define partition keys
2. Create cluster keys & turn auto clustering on the table --〉 answer
3. Create an index on the search result

Which of the following clustering metadata for the micro-partitions is maintaine d bySnowflake in a table?

1. The number of micro-partitions that comprise the table.
2. The number of micro-partitions containing values that overlap with each oth er.
3. The depth of the overlapping micro-partitions.
4. None of the above. 1,2,3

Which of the below will you consider while choosing a cluster key

1. Columns that are typically used in the selective filters. 1,2
2. Columns are frequently used in join predicates
3. Columns with extremely high cardinality.
4. Columns with extremely low cardinality

Does re-clustering in Snowflake require manual configuration

: NO – automatically and periodic active

Is re-clustering in Snowflake only triggered if the table would benefit from the operation?

- Yes

What can you easily check to see if a large table will benefit from explicitly de fining a clustering key? 1

1. Clustering depth
2. Clustering ratio
3. Values in a table

Which system functions are available in Snowflake to view/monitor the clusteri ng metadata for a table? 1,2

1. SYSTEM$CLUSTERING_DEPTH
2. SYSTEM$CLUSTERING_INFORMATION
3. SYSTEM$CLUSTERING_METADATA

Is clustering generally more cost-effective for frequently queried tables and do notchange often? : True

- But it's more expensive to keep be clustered the frequently-queried-tables

Which of the below columns is usually a good choice for clustering keys? 4

1. UUID column from a Customer in a 10TB table. --〉 카티널리티 너무 높음
2. Gender male/female in a 20TB table. --〉 카티널티가 너무 낮음
3. Timestamp in a 10TB table. --〉 카디널리티가 너무 높아 클러스터에 도움X
4. Store_id in a 2TB table

마이크로 파티션과 데이터 클러스터링은 모두 쿼리의 최적화를 위해 사용된다,

마이크로 파티션은 보다 더욱 데이터가 저장된 위치에 효율적으로 접근하기 위해 파티션을 나눈거고

클러스터링은 데이터 자체를 열 값으로 정렬하여 효울적으로 저장한 것이 주요 관점이다.

특히 이로써, 필터링/정렬/조인 작업에 큰 성능 향상을 보임.

중복되는 값은 Snowflake의 클러스터링 메커니즘이 마이크로 파티션 내에서 유사한 클러스터링 키 값을 가진 데이터를 그룹화하기 때문에 발생

여러개의 클러스터링 키가 있으면 평가하는 순서는 동적으로 결정한다.