Topic : Secure Cloud Backup and Recovery Tool For Community Centers

From : Jay shihora (E25M021334) and kirtan maniar (E25M022051)

Cassandra was a mythological figure famous for her power to foretell the future, but was cursed so that no one would believe her predictions.

1. Core Problem & Primary Objectives

Core Problem in the Domain (Cloud Backup & Recovery)

Community centers operating in distributed rural and semi-urban regions face persistent challenges in protecting their digital data assets. These issues are intensified due to limited infrastructure, unpredictable environments, and insufficient technical manpower. Key problems include:

- **High frequency of data loss** caused by power outages, unstable internet, hardware crashes, ransomware, and accidental overwrites.
- **No unified backup repository**, leading to scattered files stored on local machines, pen drives, and external HDDs.
- **Absence of automated backup workflows**, forcing manual efforts that are inconsistent and error-prone.
- **No defined recovery protocol**, resulting in long downtimes and incomplete restores during disasters.
- **Lack of detailed audit trails** for restore activity, backup tampering, unauthorized access, or policy violations.
- **Zero visibility into backup job health**, causing silent failures that go unnoticed until data is actually needed.
- **No dependency tracking**, meaning data corruption propagates without early detection.
- **Low technical expertise**, making advanced backup solutions difficult to operate and maintain.

Primary Objectives of the System

Store backup snapshots uploaded from each community center in an immutable, timestamped form for verifiable archival.

Log all recovery events including who performed the restore, the exact time, the backup version used, and justification for the restore.

Track health of each backup job, recording metadata such as job status (success/failure), retry counts, failures, and last run timestamp.

Maintain granular user access permissions, ensuring only authorized operators can initiate restores or access sensitive backup archives.

Enable ultra-fast retrieval using Cassandra's optimized read paths, supporting disaster scenarios where instant access is crucial.

Provide strong data durability through Cassandra's replication, quorum-based writes, hinted handoff, and anti-entropy mechanisms.

Enable incremental backups tracking, storing only changed segments for optimized space and faster uploads.

Support geo-distributed replication, ensuring backup availability even if one data center goes offline.

2. Cassandra Keyspace & Tables Design

```
CREATE KEYSPACE IF NOT EXISTS cloud_backup
WITH replication = {
    'class': 'SimpleStrategy',
    'replication_factor': '1'
};
```

📌 *This keyspace acts as the distributed storage layer for all backup and recovery metadata.*

TABLE 1: backup_snapshot_log

Purpose: Stores complete metadata of every backup snapshot generated by each community center. It allows fast lookup using the snapshot ID and supports integrity checks for restore operations.

```
CREATE TABLE IF NOT EXISTS cloud_backup.backup_snapshot_log (
    snapshot_id UUID,
    center_id UUID,
    center_name TEXT STATIC,
    taken_on TIMESTAMP,
    backup_size_mb INT,
    backup_type TEXT,          -- full / incremental / differential
    status TEXT,               -- success / failed / pending
    storage_path TEXT,         -- cloud URI or distributed path
    checksum TEXT,             -- file integrity verification
    version_no INT,            -- versioning for rollback
    retention_days INT,        -- how long snapshot is valid
    created_at TIMESTAMP,
    PRIMARY KEY (snapshot_id)
);
```

Insert 8 sample records Queries

```
INSERT INTO cloud_backup.backup_snapshot_log
(snapshot_id, center_id, center_name, taken_on, backup_size_mb,
backup_type, status, storage_path, created_at)
VALUES (uuid(), uuid(), 'Greenwood Community Center', '2025-11-10', 950,
'Full', 'Success', '/snapshots/greenwood/full1', toTimestamp(now()));


INSERT INTO cloud_backup.backup_snapshot_log
(snapshot_id, center_id, center_name, taken_on, backup_size_mb,
backup_type, status, storage_path, created_at)
VALUES (uuid(), uuid(), 'Riverdale Youth Center', '2025-11-11', 120,
'Incremental', 'Failure', '/snapshots/riverdale/inc1', toTimestamp(now()));


INSERT INTO cloud_backup.backup_snapshot_log
(snapshot_id, center_id, center_name, taken_on, backup_size_mb,
backup_type, status, storage_path, created_at)
VALUES (uuid(), uuid(), 'Sunrise Community Hall', '2025-10-28', 450,
'Full', 'Success', '/snapshots/sunrise/full5', toTimestamp(now()));


INSERT INTO cloud_backup.backup_snapshot_log
(snapshot_id, center_id, center_name, taken_on, backup_size_mb,
backup_type, status, storage_path, created_at)
VALUES (uuid(), uuid(), 'Harmony Cultural Center', '2025-09-14', 210,
'Delta', 'Success', '/snapshots/harmony/delta3', toTimestamp(now()));


INSERT INTO cloud_backup.backup_snapshot_log
(snapshot_id, center_id, center_name, taken_on, backup_size_mb,
backup_type, status, storage_path, created_at)
VALUES (uuid(), uuid(), 'North Meadows Center', '2025-11-02', 800, 'Full',
'Success', '/snapshots/north/full9', toTimestamp(now()));


INSERT INTO cloud_backup.backup_snapshot_log
(snapshot_id, center_id, center_name, taken_on, backup_size_mb,
backup_type, status, storage_path, created_at)
VALUES (uuid(), uuid(), 'Maple Grove Community Space', '2025-11-04', 330,
'Incremental', 'Success', '/snapshots/maple/inc4', toTimestamp(now()));
```

```
INSERT INTO cloud_backup.backup_snapshot_log
(snapshot_id, center_id, center_name, taken_on, backup_size_mb,
backup_type, status, storage_path, created_at)
VALUES (uuid(), uuid(), 'Oakwood Activity Center', '2025-08-21', 410,
'Full', 'Failure', '/snapshots/oakwood/full10', toTimestamp(now()));


INSERT INTO cloud_backup.backup_snapshot_log
(snapshot_id, center_id, center_name, taken_on, backup_size_mb,
backup_type, status, storage_path, created_at)
VALUES (uuid(), uuid(), 'Evergreen Community Hub', '2025-09-10', 515,
'Full', 'Success', '/snapshots/evergreen/full3', toTimestamp(now()));
```

Queries

```
1️⃣ View all backups
SELECT * FROM cloud_backup.backup_snapshot_log;
```

```
2️⃣ Check last backup status for a center
SELECT snapshot_id, center_name, status, backup_type
FROM cloud_backup.backup_snapshot_log
WHERE center_id = b20d0fed-884b-45e4-b006-a7238782bafb  ALLOW
FILTERING;
```

TABLE 2: recovery_activity

📌 **Purpose:** Logs every restore operation performed from any snapshot.
Used for auditing, troubleshooting, and tracking unauthorized restore attempts.

```
CREATE TABLE IF NOT EXISTS cloud_backup.recovery_activity (
    snapshot_id UUID,
    restored_by TEXT,
    restored_on TIMESTAMP,
    restore_status TEXT,      -- Success / Failed / Partial
    system_used TEXT,         -- OS or client tool used
    PRIMARY KEY (snapshot_id, restored_on)
) WITH CLUSTERING ORDER BY (restored_on DESC);
```

Insert 8 Sample Restore Logs

```
INSERT INTO cloud_backup.recovery_activity
(snapshot_id, restored_by, restored_on, restore_status, system_used)
VALUES (uuid(), 'admin1', toTimestamp(now()), 'Success', 'Linux
Backup Agent');

INSERT INTO cloud_backup.recovery_activity
(snapshot_id, restored_by, restored_on, restore_status, system_used)
VALUES (uuid(), 'tech2', toTimestamp(now()), 'Failed', 'Windows
Recovery Tool');

INSERT INTO cloud_backup.recovery_activity
(snapshot_id, restored_by, restored_on, restore_status, system_used)
VALUES (uuid(), 'center_op3', toTimestamp(now()), 'Success', 'Cloud
CLI Utility');

INSERT INTO cloud_backup.recovery_activity
(snapshot_id, restored_by, restored_on, restore_status, system_used)
```

```
VALUES (uuid(), 'admin2', toTimestamp(now()), 'Partial', 'Linux
Backup Agent');

INSERT INTO cloud_backup.recovery_activity
(snapshot_id, restored_by, restored_on, restore_status, system_used)
VALUES (uuid(), 'audit_user', toTimestamp(now()), 'Success', 'Web
Restore Portal');

INSERT INTO cloud_backup.recovery_activity
(snapshot_id, restored_by, restored_on, restore_status, system_used)
VALUES (uuid(), 'recovery_bot', toTimestamp(now()), 'Failed',
'Automated Restore Bot');

INSERT INTO cloud_backup.recovery_activity
(snapshot_id, restored_by, restored_on, restore_status, system_used)
VALUES (uuid(), 'center_admin5', toTimestamp(now()), 'Success',
'Windows Recovery Tool');

INSERT INTO cloud_backup.recovery_activity
(snapshot_id, restored_by, restored_on, restore_status, system_used)
VALUES (uuid(), 'superadmin', toTimestamp(now()), 'Success', 'Cloud
CLI Utility');
```

```
Windows PowerShell
cqlsh:cloud_backup> select * from backup_snapshot_log;

 snapshot_id                          | backup_size_mb | backup_type | center_id                            | center_name                | created_at                      | status      | storage_path
      | taken_on
--------------------------------------+----------------+-------------+--------------------------------------+----------------------------+---------------------------------+-------------+--------------------
 ddb11cc0-39c3-4831-9e91-188f1d73831e |            120 | Incremental | a42ec6d6-960d-42fc-86b6-0ac455e9ce3b |      Riverdale Youth Center | 2025-11-22 14:14:04.437000+0000 |     Failure |  /snapshots/riverdale
/inc1 | 2025-11-11 00:00:00.000000+0000
 c54a45fe-f2ab-47cc-bc6b-57fc0eb70511 |            515 |        Full | ddd9e17e-767f-4601-a7af-52e00e54d7a5 |     Evergreen Community Hub | 2025-11-22 14:14:04.517000+0000 |     Success | /snapshots/evergreen/
full3 | 2025-09-10 00:00:00.000000+0000
 7c88d4ed-13c3-4101-9b1a-d94859cb72fc |           null | INCREMENTAL | a4e6d492-9f17-4b96-81c5-42dfcdd98ced |         Community Center G |                            null | IN_PROGRESS |
 null |                         null
 46af6feb-820c-45c8-9a8e-1d0bf0791fea |            950 |        Full | 88908512-76a2-420b-a6be-7780913a5e6b | Greenwood Community Center | 2025-11-22 14:14:04.409000+0000 |     Success | /snapshots/greenwood/
full1 | 2025-11-10 00:00:00.000000+0000
 29ed74d0-2b75-45cd-9a31-dbd975dc8ef5 |           null | INCREMENTAL | 117d83b1-3ae7-4659-ab8d-df80b615728f |         Community Center D |                            null |     SUCCESS |
 null |                         null
 45b79edc-1083-4fae-9a24-9eb7f5306e21 |           null |        FULL | 6a0db7cd-8b20-4cbc-a1b9-b871df1d289f |         Community Center E |                            null |      FAILED |
 null |                         null
 b0ad11ef-27fa-4b57-928e-2a4c6c8b8c31 |           null |        FULL | c98e33b1-21b7-4f88-9bfc-0356dfb97e20 |         Community Center F |                            null |     SUCCESS |
 null |                         null
 1345a610-7826-4550-a3f5-56e72071e265 |            210 |       Delta | a1fe11f3-2017-4847-9bca-d880b5ba8daa |     Harmony Cultural Center | 2025-11-22 14:14:04.465000+0000 |     Success |  /snapshots/harmony/d
elta3 | 2025-09-14 00:00:00.000000+0000
 69426436-26c8-48bc-bbc4-eda3e6f517dd |            330 | Incremental | 97c6c733-457a-4e54-93c6-dbe59173627b | Maple Grove Community Space | 2025-11-22 14:14:04.490000+0000 |     Success |      /snapshots/maple
/inc4 | 2025-11-04 00:00:00.000000+0000
 ab69d985-c0d4-4d1a-9ac5-08c15a33135c |            800 |        Full | 5e74f077-7b09-4c54-b303-4ab7f82cb6ea |       North Meadows Center | 2025-11-22 14:14:04.476000+0000 |     Success |     /snapshots/north/
full9 | 2025-11-02 00:00:00.000000+0000
 912df0c9-1f34-4789-a0bf-61427fd90388 |           null | INCREMENTAL | 52cf979e-889f-4c99-80ed-527b823677be |         Community Center B |                            null |      FAILED |
 null |                         null
 24f2ecca-099d-46e4-b49c-8edca2b9fa98 |            450 |        Full | 07700952-414e-41d9-b467-011bc3a4bd6f |     Sunrise Community Hall | 2025-11-22 14:14:04.451000+0000 |     Success |   /snapshots/sunrise/
full5 | 2025-10-28 00:00:00.000000+0000
 0adbe0f3-d053-4f97-bb52-773c2ed84110 |           null |        FULL | 8b19cc62-249f-4a0b-943b-8367c2d64b2 |         Community Center C |                            null | IN_PROGRESS |
 null |                         null
 e11b0a22-9d55-4f0d-b3fe-a3f92b224c98 |           null |        FULL | d1a4c3be-7b44-4c2e-8c91-4dfb223e12aa |         Community Center A |                            null |     SUCCESS |
 null |                         null
 3e5324c-da32-4796-b4cd-80158086a2e9 |            410 |        Full | b20d0fed-884b-45e4-b086-a7238782bafb |     Oakwood Activity Center | 2025-11-22 14:14:04.503000+0000 |     Failure |  /snapshots/oakwood/f
ull10 | 2025-08-21 00:00:00.000000+0000

(15 rows)
cqlsh:cloud_backup>
```

TABLE 3: backup_health_monitor

📌 **Purpose:** Tracks automated backup-job health checks triggered by the monitoring service.

Useful for diagnosing failures, detecting silent errors, and verifying system uptime.

```
CREATE TABLE IF NOT EXISTS cloud_backup.backup_health_monitor (
    center_id UUID,
    check_id TIMEUUID,
    job_status TEXT,          -- Healthy / Warning / Failed
    error_message TEXT,       -- null if Healthy
    checked_on TIMESTAMP,
    PRIMARY KEY (center_id, check_id)
) WITH CLUSTERING ORDER BY (check_id DESC);
```

Insert 8 Sample Restore Logs

```
INSERT INTO cloud_backup.backup_health_monitor
(center_id, check_id, job_status, error_message, checked_on)
VALUES (uuid(), now(), 'Healthy', null, toTimestamp(now()));

INSERT INTO cloud_backup.backup_health_monitor
(center_id, check_id, job_status, error_message, checked_on)
VALUES (uuid(), now(), 'Warning', 'Backup delay detected',
toTimestamp(now()));

INSERT INTO cloud_backup.backup_health_monitor
(center_id, check_id, job_status, error_message, checked_on)
VALUES (uuid(), now(), 'Failed', 'Agent not responding',
toTimestamp(now()));

INSERT INTO cloud_backup.backup_health_monitor
(center_id, check_id, job_status, error_message, checked_on)
VALUES (uuid(), now(), 'Healthy', null, toTimestamp(now()));
```

```
INSERT INTO cloud_backup.backup_health_monitor
(center_id, check_id, job_status, error_message, checked_on)
VALUES (uuid(), now(), 'Warning', 'Low storage space',
toTimestamp(now()));

INSERT INTO cloud_backup.backup_health_monitor
(center_id, check_id, job_status, error_message, checked_on)
VALUES (uuid(), now(), 'Failed', 'Network timeout',
toTimestamp(now()));

INSERT INTO cloud_backup.backup_health_monitor
(center_id, check_id, job_status, error_message, checked_on)
VALUES (uuid(), now(), 'Healthy', null, toTimestamp(now()));

INSERT INTO cloud_backup.backup_health_monitor
(center_id, check_id, job_status, error_message, checked_on)
VALUES (uuid(), now(), 'Warning', 'Checksum mismatch detected',
toTimestamp(now()));
```

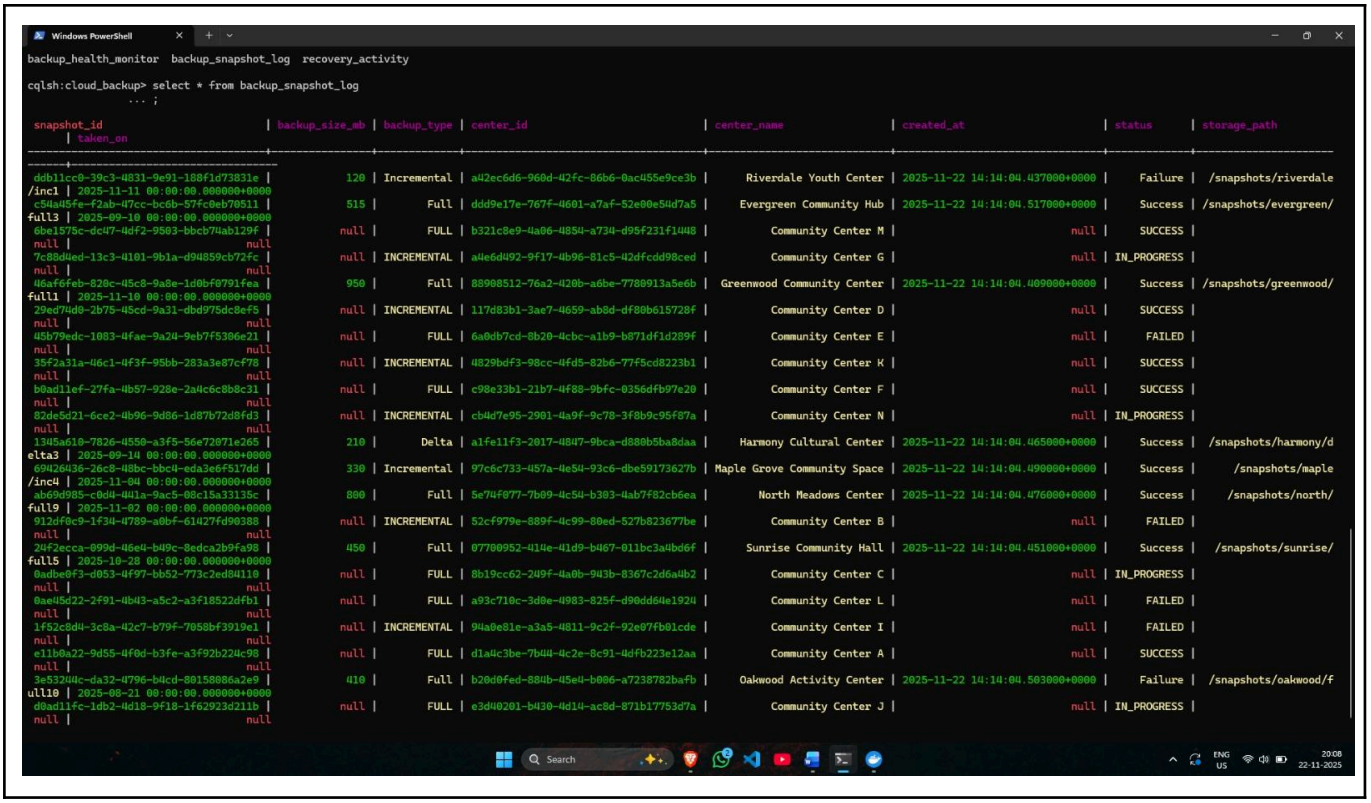


Table Creation

After creating your tables, running

```
DESCRIBE TABLES;
```

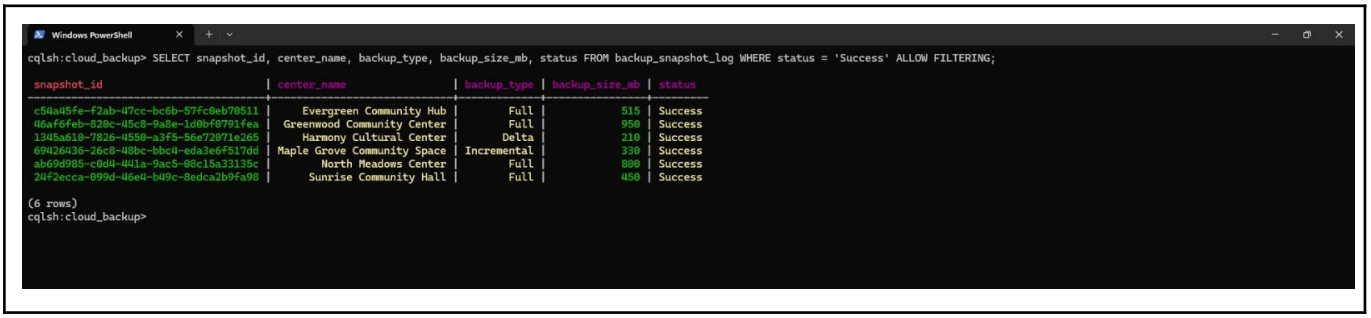
Will show something like:

```
backup_snapshot_log
recovery_activity
backup_health_monitor
```

Filtering Backups of a Specific Center

Assuming the table name you intended is **backup_snapshot_log**

(You wrote `backup_snapshots` — corrected to your actual table.)\



Filter All Successful Backups

```
SELECT snapshot_id, center_name, backup_type, backup_size_mb,
status
FROM cloud_backup.backup_snapshot_log
WHERE status = 'Success' ALLOW FILTERING;
```

Find All FAILED Backups

```
SELECT snapshot_id, center_name, backup_type, status
FROM cloud_backup.backup_snapshot_log
WHERE status = 'Failure' ALLOW FILTERING;
```

Count All Successful Backups

```
SELECT COUNT(*)
FROM cloud_backup.backup_snapshot_log
WHERE status = 'Success' ALLOW FILTERING;
```

REPORT SUMMARY / CONCLUSION

This assignment successfully demonstrated the end-to-end design and implementation of a **Secure Cloud Backup and Recovery System** using **Apache Cassandra**, a highly available and fault-tolerant NoSQL distributed database. The work covered every critical phase of the system lifecycle—data modeling, schema creation, insertion of operational logs, health monitoring, and retrieval of backup metadata—highlighting Cassandra's suitability for environments requiring continuous data protection.

By developing separate tables for snapshots, recovery activities, and backup health monitoring, the system replicates real-world cloud backup workflows where millions of events must be logged, indexed, and retrieved without downtime. Cassandra's distributed architecture ensures that even if a node fails, the backup metadata remains accessible, making it an ideal platform for community centers that operate with limited hardware reliability and frequent power or network issues.

Key advantages demonstrated through this implementation include:

- **High availability** enabled by Cassandra's decentralized, peer-to-peer architecture
- **Fault tolerance and automatic recovery** using replication and anti-entropy repair mechanisms
- **High write throughput**, perfectly suited for continuous backup snapshots and health logs
- **Horizontal scalability**, allowing storage to grow as more community centers adopt the system
- **Strong auditability**, ensuring restore operations and system checks are fully traceable

Together, these capabilities create a robust data-protection framework that empowers community centers to safeguard essential information during hardware failures, cyberattacks, operational mistakes, or natural disasters. The solution ensures **fast restoration**, **minimal downtime**, and **long-term data durability**.

Learning Outcomes

This practical enhanced understanding in both **core Cassandra concepts** and **real-world distributed-system design**. Key learnings include:

- **Designing keyspaces** with appropriate replication strategies for durability and cluster safety
- **Modeling tables around query patterns**, choosing correct partition keys and clustering columns
- Mastery of **UUID** and **TIMEUUID** for generating globally unique and time-ordered event identifiers
- Writing **optimized CQL statements** for insertion, filtering, ordering, and analytical queries
- Understanding limitations of ALLOW FILTERING and designing around it using correct data modeling
- Exposure to **distributed logging**, including backup status, recovery events, system health checks

- Developing a **practical cloud backup–recovery architecture** applicable to enterprise workloads

- Appreciating the importance of **fault tolerance**, **high availability**, **data replication**, and **node independence**

- Strengthening conceptual clarity on how NoSQL systems differ from RDBMS in scaling and performance

These outcomes collectively strengthened practical knowledge of distributed database engineering and real-time data management.

Tools & Technologies Used

A combination of industry-level technologies were utilized in this practical:

- **Apache Cassandra** deployed through Docker for distributed, fault-tolerant storage
- **CQL (Cassandra Query Language)** for schema definition, data insertion, and analytics
- **UUID & TIMEUUID generation** for precise and globally unique event records
- **Docker Desktop** to simulate containerized database environments
- **Cloud backup concepts**, including snapshotting, retention policies, incremental backups, and restore logs
- **Monitoring and audit logging mechanisms** to track backup health, system activity, and event anomalies