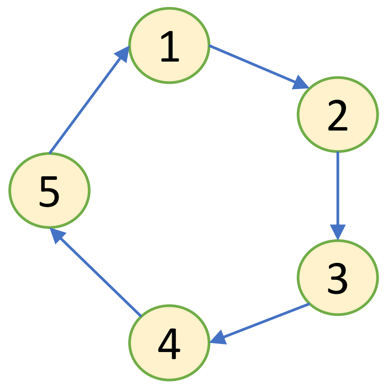
Cassandra

A distributed Open Source NoSQL Database

Cassandra CQL	vs **SQL**
CREATE **KEYSPACE** myDatabase WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};	CREATE **DATABASE** myDatabase;
USE myDatabase;	--"--
CREATE TABLE IF NOT EXISTS myTable (id INT PRIMARY KEY); -- (synonyms in cql: COLUMNFAMILY=TABLE) **NB table need primary key** in CQL.	--"--
ALTER TABLE myTable ADD myField INT;	--"--
CREATE INDEX myIndex ON myTable (myField);	--"--
INSERT INTO myTable (id, myField) VALUES (1, 7);	--"--
SELECT * FROM myTable WHERE myField = 7;	--"--
SELECT COUNT(*) FROM myTable;	--"--
DELETE FROM myTable WHERE myField = 7;	--"--

CQL	**SQL**
- No support for things like **JOIN, GROUP BY, or FOREIGN KEY**. Leaving these features out is important because it makes writing and retrieving data from Cassandra much more efficient.	**JOIN, GROUP BY, FOREIGN KEY**
Writes are cheap. Write everything the way you want to read it. CQL does not perform a read while inserting. Without a read, there is no way to know if the data being inserted is replacing an existing record. This means that both inserts and updates are extremely fast.	
UPDATE myTable SET myField = 2 WHERE id = 6; - However, if the row does not exist, it will still get created. Similarly as unintuitive, an INSERT statement will actually replace data if it exists. In where-clause, only primary key column can be used. Under the hood, INSERT and UPDATE are treated the same by Cassandra ("Upserts"), except for Counter columns/tables. Both INSERT and UPDATE require complete PRIMARY KEY.	--"--
Transaction Control Language (TCL) - **Not in CQL**	COMMIT – It saves the work done SAVEPOINT – It identifies a point in a transaction to which you can later roll back ROLLBACK – It restores database to original since the last COMMIT
Data Retrieval/Query Language (DRL/DQL): **Simple transactions** (Relation between database objects is not possible): - Where clause: only on primary key or secondary indexes! - Can use only AND operator, There are no OR and NOT operators.	Data Retrieval/Query Language (DRL/DQL): **Full transactions.**



Nodes →

Partition# = hash(PrimKey)

Keyspace->

Replicas:3

	1	2	3	4	5
1	SE	SE	SE		
2		DE	DE	DE	
3			NO	NO	NO
4	DK			DK	DK
5	UK	UK			UK

Replicas:1

	1	2	3	4
1	SE			
2		DE		
3			NO	
4				DK

Replicas:2

	1	2
1	SE	SE
2	DE	DE
3	NO	NO
4	DK	DK

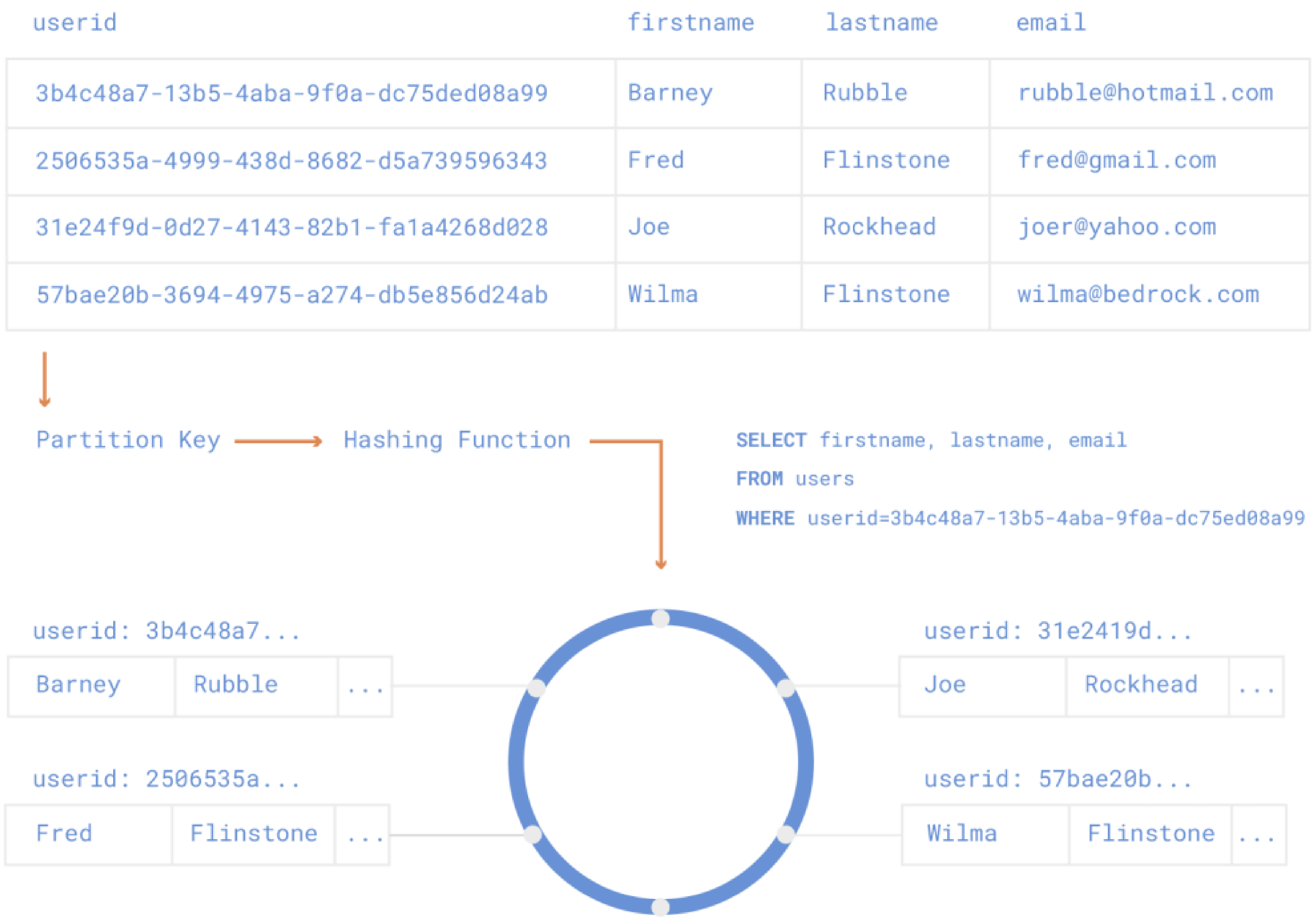
Replicas:1

	1	2
1	SE	
2		DE
3	NO	
4		DK

Replicas:1

	1
1	SE
2	DE
3	NO
4	DK

	~~1~~	~~2~~	~~3~~	~~4~~
~~1~~	~~SE, DE~~	~~SE, DE~~		
~~2~~		~~NO, DK~~	~~NO, DK~~	



userid	firstname	lastname	email
3b4c48a7-13b5-4aba-9f0a-dc75ded08a99	Barney	Rubble	rubble@hotmail.com
2506535a-4999-438d-8682-d5a739596343	Fred	Flintstone	fred@gmail.com
31e24f9d-0d27-4143-82b1-fa1a4268d028	Joe	Rockhead	joer@yahoo.com
57bae20b-3694-4975-a274-db5e856d24ab	Wilma	Flintstone	wilma@bedrock.com

```
SELECT firstname, lastname, email
FROM users
WHERE userid=3b4c48a7-13b5-4aba-9f0a-dc75ed08a99
```

FIGURE 1 How Cassandra stores data