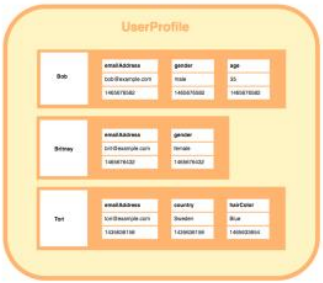
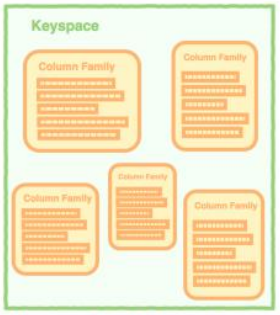
35. Sloupcové databáze – koncept, sloupcově orientovaný model, výhody a nevýhody. Cassandra – architektura, distribuce dat a replikace, sekundární index

Sloupcové databáze – koncept, sloupcově orientovaný model, výhody a nevýhody

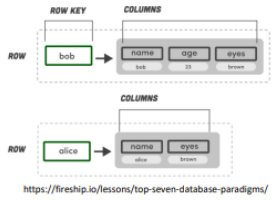
- sloupcově orientovaný model
  - základem je keyspace
    - volně odpovídá schématu v relačních databázích
    - dává představu o struktuře databáze
  - keyspace obsahuje column families
    - volně odpovídají relacím v relačních databázích



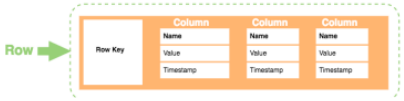


https://database.guide/what-is-a-column-store-database/

- sloupcově orientovaný model
  - column family obsahuje seřazené řádky (rows)
    - jednoznačně identifikovány klíčem (row key)
  - každý řádek obsahuje sloupce (columns)
    - sloupce patří jen svému řádku
      - na rozdíl od relačních databází
    - počet sloupců se mezi řádky může lišit
    - stejně tak se mohou lišit názvy sloupců i jejich datové typy
  - každý sloupec obsahuje pár jméno – hodnota a časovou značku



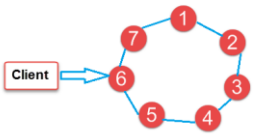
https://fireship.io/lessons/top-seven-database-paradigms/



- výhody sloupcových databází
  - horizontální škálování a replikace
  - volné schéma
  - rychlé načítání dat, dotazování a agregace, komprese dat
- hlavní použití
  - množství zápisů je výrazně vyšší než čtení, malé množství aktualizací
    - ukládání dat ze senzorů, časových řad
    - ukládání historických záznamů
  - čtení podle primárního klíče
    - sledování statusů, balíčků
- významnější zástupci
  - Cassandra, HBase, Azure Cosmos DB

Cassandra – architektura, distribuce dat a replikace, sekundární index

- sloupcová NoSQL databáze (od 2008)
  - od Apache
  - open source
    - napsaná v Javě
    - založená na Amazon Dynamo a Google Big Table
  - distribuovaná
    - horizontální (lineární) škálování, replikace
  - technologie pro big data
  - vysoká dostupnost
    - velmi často na velkém množství uzlů přes více datových center
    - laditelná konzistence
  - odolnost vůči chybám
    - každý uzel má stejnou funkcionalitu (vs. master – slave)



https://www.guru99.com/cassandra-tutorial.html

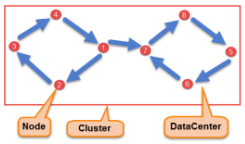
- možnosti aplikace
  - katalogy produktů / playlisty
    - Netflix, Comcast, Hulu
  - doporučovací systémy
    - Outbrain, eBay
  - detekce podvodů
    - Barracuda Networks, Instagram
  - platformy pro posílání zpráv
  - senzorová data / IoT
  - herní průmysl
  - služby založené na poloze

ARCHITEKTURA CASSANDRY

- základem je uzel (node)
  - ukládá data
  - každý uzel plní stejnou roli
    - neexistují slaves ani master
  - uzly jsou replikované
  - neexistuje jeden bod selhání


https://www.udemy.com/course/apache-cassandra

- příbuzné uzly tvoří datová centra
  - fyzické i virtuální
  - replikace na úrovni datových center
  - prstencová struktura
- datová centra tvoří cluster
  - dostupnost dat i při výpadku centra



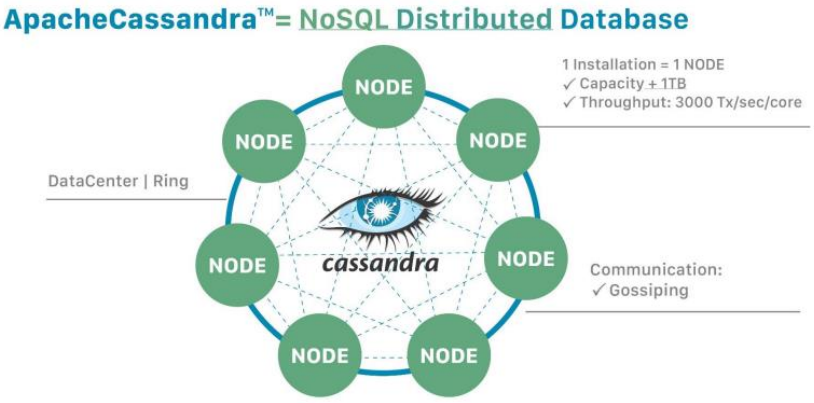
https://www.guru99.com/cassandra-architecture.html

- jak o sobě ale uzly vědí?
  - neexistuje master a všechny uzly také zastávají stejnou roli…
  - snitch
    - poskytuje uzlům informaci o topologii sítě, routuje requesty
    - definuje skupiny uzlů do datových center a racků
      - replikační strategie využívá skupiny pro umístění replik
    - konfigurována při vytvoření clusteru
    - konfigurační soubor uložený na všech uzlech
    - SimpleSnitch (základní)
      - použití jen pro clustery s jedním datovým centrem
    - PropertyFileSnitch
    - GossipingPropertyFileSnitch
      - doporučená pro produkční nasazení
      - používá soubor přiřazený ke každému uzlu popisující jeho lokaci, uzly si tuto informaci vyměňují
    - všechny typy v základu používají dynamickou snitch vrstvu (DynamicEndpointSnitch)
      - monitoruje výkon a volí nejlepší repliku pro čtení

```
130.77.100.147 =DC1:RAC1
130.77.100.148 =DC1:RAC1
130.77.100.165 =DC1:RAC1
130.77.200.109 =DC1:RAC2
130.77.200.110 =DC1:RAC2
130.77.200.111 =DC1:RAC2

155.23.100.128 =DC2:RAC1
155.23.100.129 =DC2:RAC1
155.23.200.107 =DC2:RAC2
155.23.200.108 =DC2:RAC2
```
https://www.udemy.com/course/apache-cassandra

- jak spolu uzly komunikují?
  - gossip
    - peer-to-peer komunikační protokol pro zjišťování a sdílení informací o ostatních uzlech
      - umístění a stav
    - každou vteřinu každý uzel komunikuje až se třemi dalšími uzly
    - uzly si vyměňují informace o sobě a o všech ostatních uzlech, o kterých mají informace
    - každá gossip zpráva také obsahuje verzi pro snadnou aktualizaci
  - starší zprávy jsou přepsány novějšími
  - jedná se o interní komunikační metodu
    - pro externí komunikaci je používáno CQL
      - aplikace – Cassandra

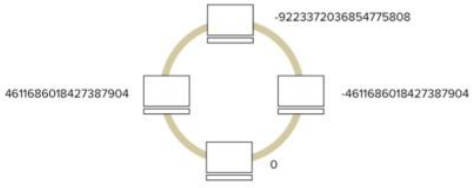


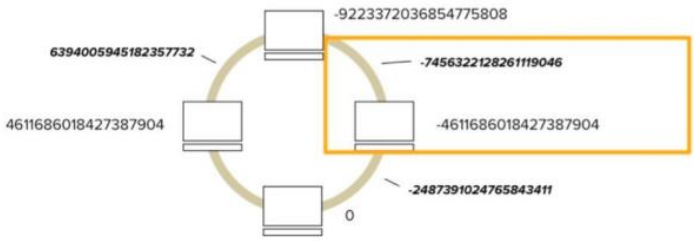
Distribuce dat

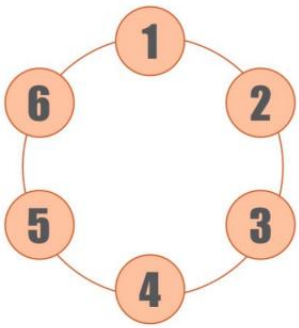
- prováděna pomocí konzistentního hashování
  - cílem je rovnoměrné rozložení dat na uzlech v clusteru
  - záznamy (řádky) tabulky jsou distribuovány mezi uzly clusteru
    - snaha o rovnoměrné rozložení zátěže při načítání dat tabulky
- pro distribuci řádků je použit partitioner
  - každý řádek je definovaný primárním klíčem, ze kterého je odvozen partition klíč
- využívá algoritmus přidělující každému řádku odpovídající token (a uzel)
- v základu Murmur3 partitioner
- Murmur3 vezme partition klíč (celý primární klíč, případně jeho první část) a vygeneruje unikátní číslo v rozmezí -2^{63} a 2^{63}

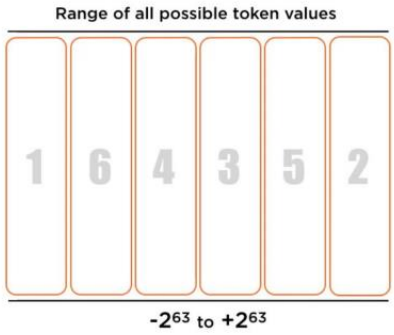
- Každý uzel má svůj rozsah (range)

H01033638 -> **-7456322128261119046**
H01545551 -> **-2487391024765843411**
H00999943 -> **6394005945182357732**



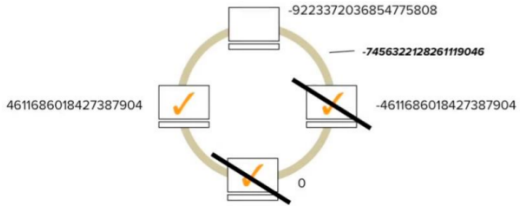






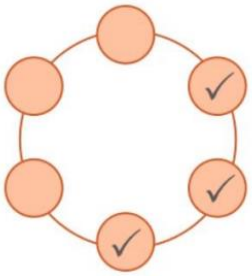
Replikace

- při definování databáze je potřeba specifikovat replikační faktor
  - udává počet instancí dat v dané databázi
    - 1 bez replikace
    - 3 pro zajištění neexistence jednoho bodu selhání
      - výpadek uzlu neznamená ztrátu dat
- o umístění další repliky rozhoduje replikační strategie



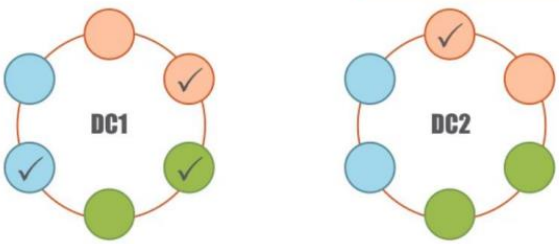
1. SimpleStrategy
   - většinou pro vývoj, případně pro cluster v jediném datacentru
   - synergie se SimpleSnitch

```
create keyspace pluralsight with replication =
  {'class': 'SimpleStrategy', 'replication_factor': 3};
```



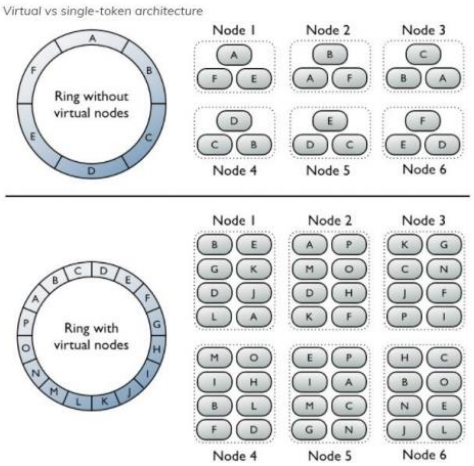
2. NetworkTopologyStrategy
   - i pro clustery ve více datových centrech
   - synergie s GossipingPropertyFileSnitch

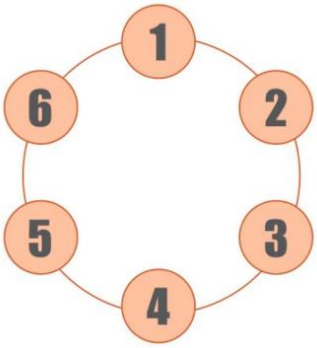
```
create keyspace pluralsight with replication =
  {'class': 'NetworkTopologyStrategy', 'DC1': 3, 'DC2', 1};
```

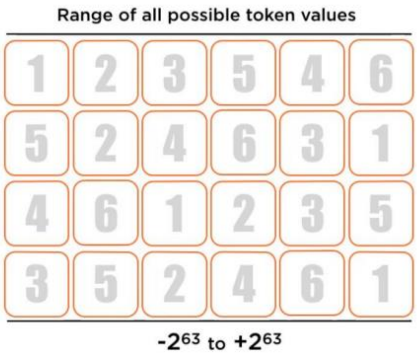


Virtuální uzly

- již v základním nastavení
- alternativní způsob přiřazení rozsahů uzlům
- uzel je místo jednoho rozsahu zodpovědný za mnoho menších rozsahů
  - v základu 256
- umožňují nastavit více rozsahů lepším strojům a méně horším
- vytvořeny pro usnadnění přidávání nových uzlů a zároveň udržení rovnovážného rozložení dat v clusteru
- nový uzel po přidání obdrží mnoho menších rozsahů z ostatních uzlů







CQL

CQL

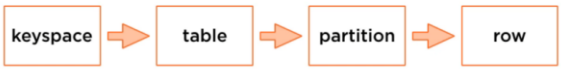
- Cassandra Query Language
  - dotazovací jazyk podobný SQL
  - dokumentace
  - není case sensitive
  - nemá veškeré možnosti jako SQL
    - z důvodu distribuované povahy Cassandry
    - např. chybí joiny
      - byly by neefektivní

```
SELECT home_id, datetime, event, code_used FROM activity;
```

vybere sloupce home_id, …, code_used z tabulky activity

Cassandra prakticky

- v Cassandře je databáze definována jako keyspace
- keyspace obsahuje tabulky (tables)
- všechna data mají přiřazený partition klíč
  - určuje jejich umístění v clusteru
  - všechna data v partition uložena spolu
- data v partition reprezentována jako jeden nebo více řádků



Primární klíč a Partition key

- primární klíč
  - v případě definice jen nad jedním sloupcem, dva způsoby zápisu

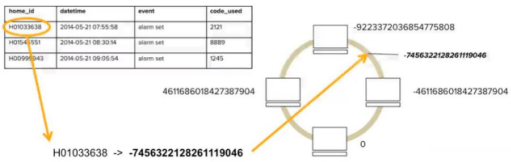
```
CREATE TABLE home (
home_id text,
address text,
city text,
state text,
zip text,
owner text,
phone text,
alt_phone text,
email text,
phone_password text,
main_code text,
guest_code text,
PRIMARY KEY (home_id)
);
```

```
CREATE TABLE home (
home_id text PRIMARY KEY,
address text,
city text,
state text,
zip text,
owner text,
phone text,
alt_phone text,
email text,
phone_password text,
main_code text,
guest_code text
);
```

https://www.udemy.com/course/apache-cassandra

- vytvoření tabulky
  - partition klíč
    - zodpovídá za distribuci dat na uzly
      - a přístup k nim
    - partitioner hodnotu klíče hashuje a na základě výsledku přiřadí partition (a rozsah a uzel)
    - odvozený z primárního klíče
      - z jednoduchého primárního klíče je partition klíč celý primární klíč
      - ze složeného primárního klíče je partition klíč první část primárního klíče
      - zbylé sloupce se nazývají clustering columns a slouží k setřídění partition
    - ➢ v praxi všechny CQL řádky se stejným partition klíčem jsou uloženy ve stejné partition

```
CREATE TABLE activity (
home_id text,
datetime timestamp,
event text,
code_used text,
PRIMARY KEY (home_id, datetime)
...
```



https://www.udemy.com/course/apache-cassandra

- composite partition klíč
  - využití více sloupců jako partition klíč
  - více záznamů v jedné partition
  - zpravidla se používá v případech, kdy data uložená v partitions jsou příliš velká
    - rozdělení dat na části

```
cqlsh> CREATE TABLE cycling.rank_by_year_and_name (
race_year int,
race_name text,
cyclist_name text,
rank int,
PRIMARY KEY ((race_year, race_name), rank)
);
```

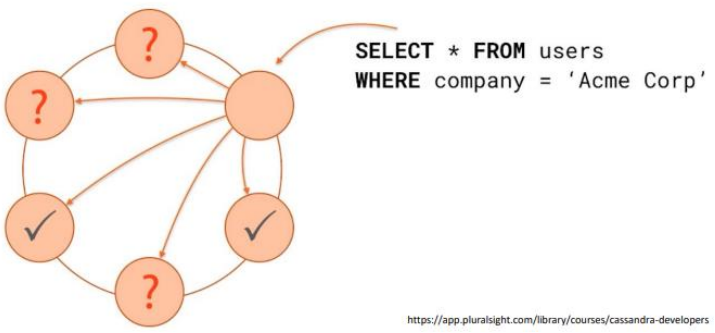
https://docs.datastax.com/en/cql-oss/3.3/cql/cql_using/useKSQualifier.html

- co je potřeba si uvědomit?
  - Cassandra je distribuovaná databáze
    - neexistují joiny
    - veškerá data k danému dotazu musí být jen v jedné tabulce
    - vede k denormalizovaným datům
      - zápisy jsou rychlé, redundance není až takový problém
- je tedy potřeba si dobře rozmyslet, které dotazy budou potřeba
  - a následně podle nich navrhnout tabulky
  - partition key, clustering columns a sekundární indexy
  - kontrast oproti klasickým datovým přístupům v relačních databázích

Sekundární indexy

- sekundární indexy
  - možnost definovat nad vybranými sloupci
    - kromě partition key, který už indexovaný je
  - umožňují filtrování nad danými sloupci
    - použití WHERE klauzule
  - pro každý sekundární index je vytvořena skrytá tabulka na každém uzlu
    - pro přístup
  - nezrychlují prohledávání, ale umožňují jej i nad ostatními sloupci
    - pro zrychlení je možnost vytvořit tabulku přímo odpovídající dotazu
  - CREATE INDEX <jméno indexu> ON <jméno tabulky> (<jméno sloupce>)

- sekundární indexy



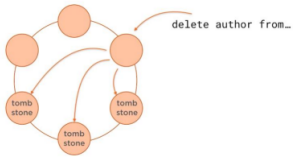
https://app.pluralsight.com/library/courses/cassandra-developers

Update

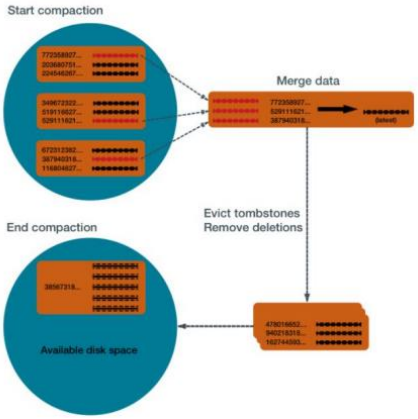
- jak ale update probíhá?
  - situace je odlišná od relačních databází
    - neplatí vyhledání záznamu na disku, jeho aktualizace a uložení
  - ➢ update je v podstatě další zápis
    - zápis do memtable
    - po naplnění memtable uložení do nové SSTable
    - konflikty jsou řešeny pomocí časových značek
    - při čtení Cassandra prochází všechny memtables a SSTables
    - ➢ vrácený záznam je ten s poslední časovou značkou

Odstranění dat

- co se stane při odstranění?
  - při zadání příkazu pro odstranění je vytvořen tzv. tombstone
  - ten označuje data pro smazání
  - proč ale nejsou data smazána ihned?
    - důvodem je distribuovaná povaha Cassandry
    - při okamžitém smazání by mohlo dojít ke zpětné replikaci z jiného uzlu
      - který nemusel v době smazání být k dispozici
    - ➢ uzly (i ty momentálně neběžící) tak dostávají čas, aby se o označení dozvěděly
  - doba mezi označením a možným smazáním dána ve vlastnostech tabulky
    - gc_grace_seconds
    - v základu 10 dní (864 000 sekund)
    - po uplynutí této doby je data možné smazat



- co se stane při odstranění?



- time to live (TTL)
  - způsob, kterým lze vkládaným datům nastavit expirační lhůtu
  - pomocí USING TTL

```
INSERT INTO location (vehicle_id, date, time, latitude,
longitude) VALUES ('AZWT3RSKI', '2014-05-20', '2014-05-20
11:23:55', 34.872689, -111.757373) USING TTL 30;
```

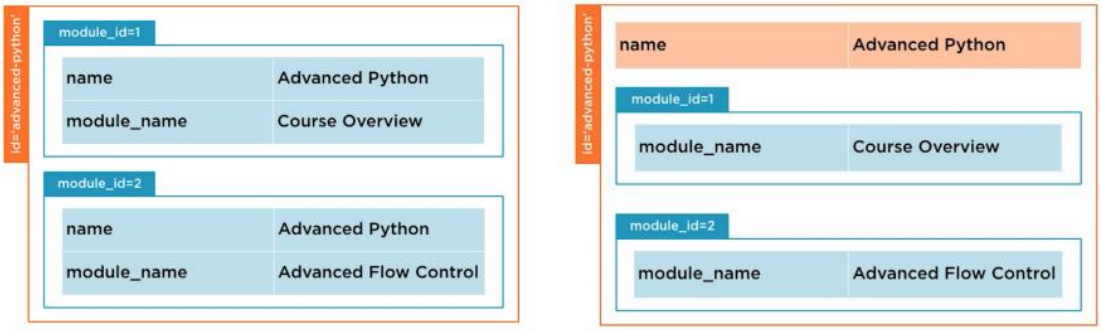
https://www.udemy.com/course/apache-cassandra

  - v sekundách (např. 30 vteřin)
  - po vypršení TTL nastává standardní procedura pro smazání
    - tombstone
    - gc_grace_period
    - compaction

Statické sloupce

- sloupce, jejichž hodnoty jsou konstantní pro všechny řádky v partition
- STATIC

```
CREATE TABLE courses (
  id varchar,
  name varchar STATIC,
  module_id int,
  module_name varchar,
  PRIMARY KEY (id, module_id)
);
```



- základní funkce jsou vestavěné
  - minimum (min)
  - maximum (max)
  - průměr (avg)
  - suma (sum)

```
id                                   | race_points | firstname | lastname
-------------------------------------+-------------+-----------+----------
e3b19ec4-774a-4d1c-9e5a-dececle30aac |           6 |   Giorgia | BRONZINI
e3b19ec4-774a-4d1c-9e5a-dececle30aac |          75 |   Giorgia | BRONZINI
e3b19ec4-774a-4d1c-9e5a-dececle30aac |         120 |   Giorgia | BRONZINI
```

```
cqlsh> SELECT sum(race_points) FROM cycling.cyclist_points WHERE id=e3b19ec4-774a-4d1c-9e5a-dececle30aac;

 system.sum(race_points)
-------------------------
                     201
```

  - počet (count)
    - př. počet soutěžících z Belgie

```
country | cyclist_name | flag
--------+--------------+------
Belgium |       Andre  |    1
Belgium |     Jacques  |    1
 France |       Andre  |    3
 France |      George  |    3
```

```
cqlsh> SELECT count(cyclist_name) FROM cycling.country_flag WHERE country='Belgium';

 system.count(cyclist_name)
----------------------------
                          2
```

- možnost definovat vlastní agregační funkce

FUNKCE DEFINOVANÉ UŽIVATELEM

- uživatelské funkce (UDF)
  - aplikované na data v tabulce v rámci dotazu
  - musí být vytvořeny před použití v SELECT dotazu
  - aplikovány na všechny záznamy
  - nutno povolit v cassandra.yaml
  - v základu Java nebo Javascript
    - možnost rozšířit např. o Python, Ruby nebo Scalu
  - př. logaritmus

```
CREATE FUNCTION IF NOT EXISTS fLog (input double)
  CALLED ON NULL INPUT
  RETURNS double
  LANGUAGE java AS '
    return Double.valueOf(Math.log(input.doubleValue()));
    ';
```

https://docs.datastax.com/en/cql-oss/3.3/cql/cql_using/useCreateUDF.html

  - CALLED ON NULL INPUT – zajistí, že funkce bude vždy provedena
  - RETURNS NULL ON NULL INPUT – vrátí pro vstupní NULL výstupní NULL
  - RETURNS – definuje vracený datový typ

- vybraná témata
  - statické sloupce
    - sloupce, jejichž hodnoty jsou konstantní pro všechny řádky v partition
  - počítadla
  - agregační funkce
  - funkce definované uživatelem
  - lightweight transakce
    - IF NOT EXISTS, IF
  - materialized view
    - speciální tabulka s daty, která jsou vkládána a aktualizována podle zdrojové tabulky
    - tabulky se liší primárním klíčem a vlastnostmi
    - ➢ umožňuje provádět jiné optimalizované dotazy
    - ➢ a zároveň udržuje data aktuální podle zdrojové tabulky

```
CREATE TABLE courses (
  id varchar,
  name varchar STATIC,
  module_id int,
  module_name varchar,
  PRIMARY KEY (id, module_id)
);
```

Insert

```
INSERT INTO users (id, first_name, last_name)
VALUES ('john-doe', 'John','Doe')
IF NOT EXISTS;
```

https://app.pluralsight.com/library/courses/cassandra-developers