

2교시.

Data/Query/Index 서비스 소개 및 실습

- 1 UI 설명 및 Data 서비스 소개
- 2 Query/Index 서비스 소개
- 3 실습
- A JSON 데이터 모델



2-1. UI 설명 및 Data 서비스 소개



Flexibility of JSON | Relaxed Normalization

Relational Tables

USERS		
ID	First	Last
1	Jeffrey	Mackay

USERS_SKILLS	
UserID	Skill Name
1	SQL
1	JavaScript
1	NoSQL

USERS_EXPERIENCE		
ID	Role	Company
1	Solutions Architect	AppMax Inc
1	Solutions Engineer	Couchbase

JSON Document

```
{
  "UserID": "1",
  "firstName": "Jeffrey",
  "lastName": "Mackay",
  "skills": ["SQL", "JavaScript", "NoSQL"],
  "experience": [
    {
      "role": "Solutions Architect",
      "company": "AppMax Inc",
    },
    {
      "role": "Solutions Engineer",
      "company": "Couchbase",
    }
  ]
}
```

- Rigid Schema, Fixed length columns
- Normalization means a lot of costly joins

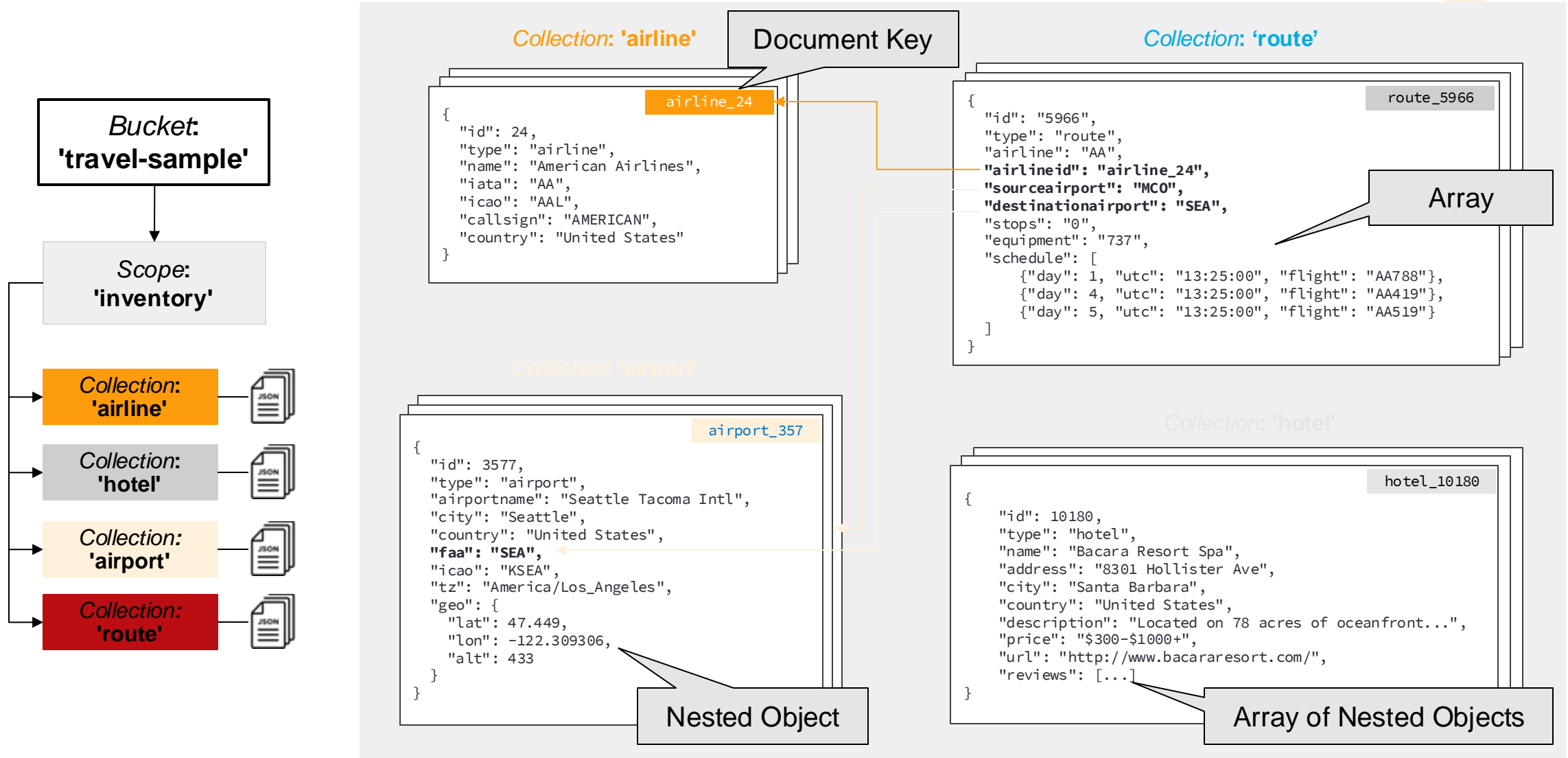
- Document format aligns to application needs
- Less joins than RDBMS, improving performance

Flexibility of JSON | Simple mapping from RDBMS to Couchbase

Relational	Couchbase
Server	Cluster
Database	Bucket
Schema	Scope
Table	Collection
Row	Document

- Distributed highly available and scalable data platform
- Buckets are similar to a relational database
- Scopes define namespaces
- Collections are tables with flexible schemas
- JSON Documents provide a flexible way to store data

Flexibility of JSON | Travel Sample Example



Couchbase UI 설명

작업

계정

activity

Administrator



cbdb > Dashboard

Enterprise Edition 7.6.3 build 4200 · IPv4 · © 2024 Couchbase, Inc.

Dashboard

Servers

Buckets

Backup

XDCR

Security

Settings

Logs

Documents

Query

Indexes

Search

Analytics

Eventing

Views

Choose Dashboard or create your own

Cluster Overview

Stat Interval

Minute

Bucket

travel-sample

Nodes

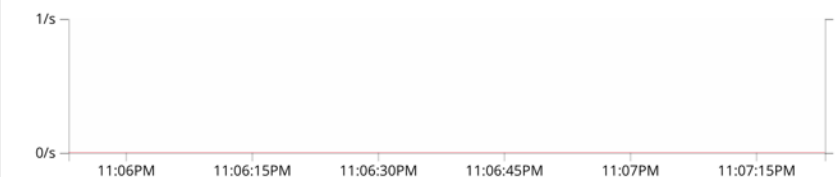
All Server Nodes (1)

Reset

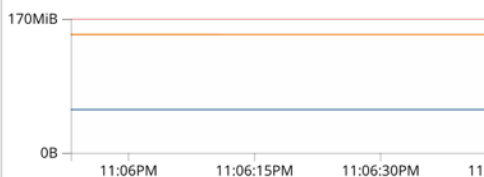
Cluster Overview

Indexes #primary

Total Ops, N1QL Request Rate, Search Query Rate, Temp OOM Rate, Cache Miss Ratio, Gets, Sets, ...



Data Total RAM Used, Low Water Mark, High Water Mark



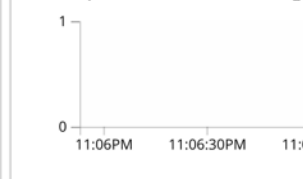
Active Items, Replica Items, Active Resident Ratio, Replica Resident Ratio, Docs Fragmentation



Disk Write Queue, Data Total Disk Size



DCP Replication Items Remaining



Disk Read Failures, Disk Write Failures, N1QL Error Rate, Search Query Error Rate, Failed Function ...



Queries > 250ms, Queries > 500ms, Queries > 1000ms, Queries > 5000ms



XDCR Total Outbound Mutations, Index Mutations Remaining, Search Mutations Remaining

Data - 1 bucket Index Query Search Analytics Eventing XDCR
1 active node 0 failed-over nodes 0 nodes pending rebalance 0 inactive nodes

관리
항목

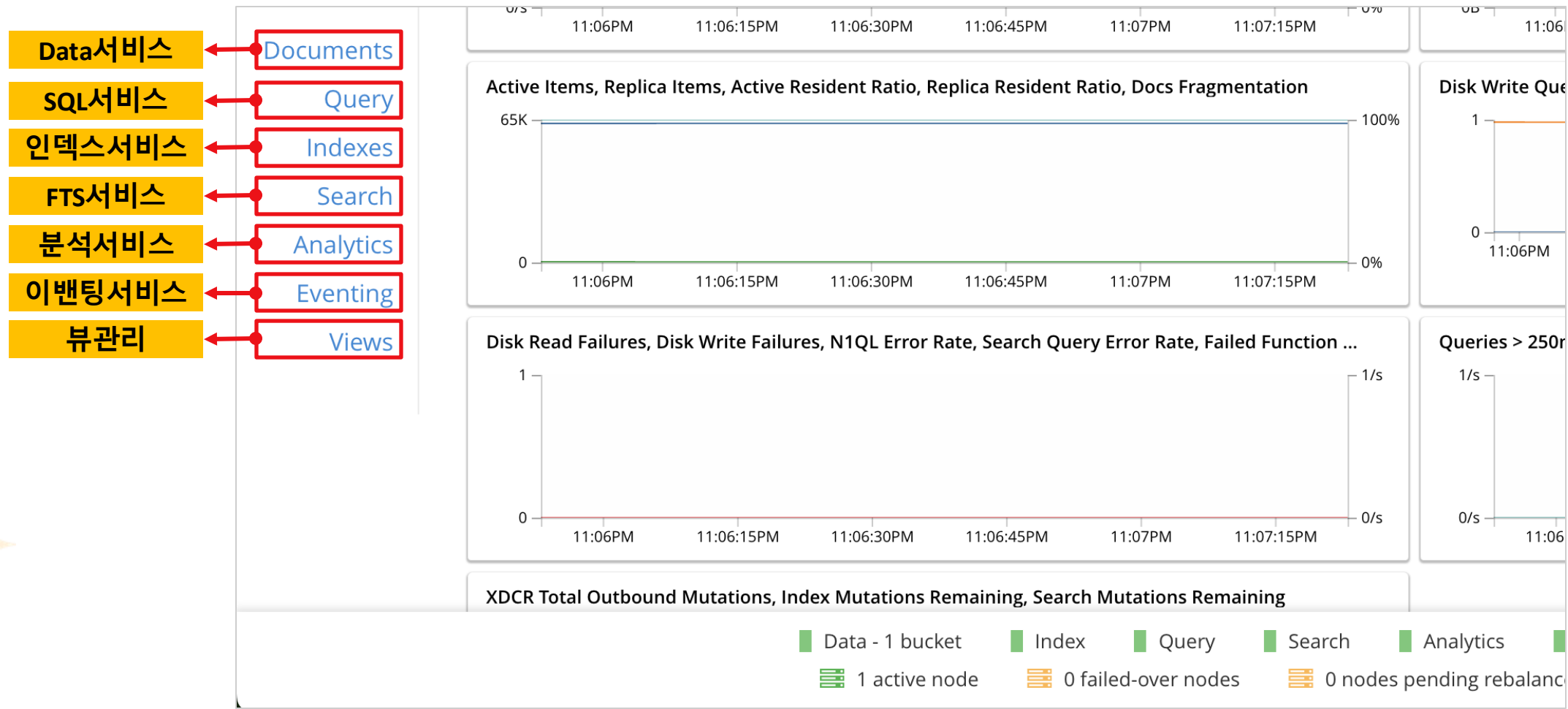
서비스
항목



Couchbase UI 설명

The screenshot shows the Couchbase UI Dashboard. On the left, a vertical sidebar contains yellow buttons with Korean labels: '모니터링' (Monitoring), '서버관리' (Server Management), '버킷관리' (Bucket Management), '백업관리' (Backup Management), 'DR관리' (DR Management), '계정관리' (Account Management), '성능관리' (Performance Management), and '로그관리' (Log Management). Red arrows point from these labels to the corresponding menu items in the sidebar: 'Dashboard', 'Servers', 'Buckets', 'Backup', 'XDCR', 'Security', 'Settings', and 'Logs'. The main content area has a blue header with the Couchbase logo and 'cbdb > Dashboard'. Below the header, there are filters: 'Choose Dashboard' (set to 'Cluster Overview'), 'Stat Interval' (set to 'Minute'), 'Bucket' (set to 'travel-sample'), and 'Nodes' (set to 'All Server Nodes (1)'). The 'Cluster Overview' section contains two charts. The left chart, titled 'Total Ops, N1QL Request Rate, Search Query Rate, Temp OOM Rate, Cache Miss Ratio, Gets, Sets, ...', shows a line graph with a y-axis from 0/s to 1/s and an x-axis from 11:06PM to 11:07PM. The right chart, titled 'Data Total RAM Used, Low Water Mark, H', shows a line graph with a y-axis from 0B to 170MiB and an x-axis from 11:06PM to 11:07PM.

Couchbase UI 설명



Couchbase UI 설명 - Bucket/Scope/Collection 생성/삭제

참고 : <https://docs.couchbase.com/server/current/manage/manage-buckets/create-bucket.html>

cbdb > Buckets

activity help Administrator

ADD BUCKET

Dashboard

Servers

Buckets

Backup

XDCR

Security

Settings

Logs

Documents

Query

filter buckets...

name

beer-sample

pos

travel-sample

items

7,306

2,109,414

63,349

resident

100%

52.8%

100%

ops/sec

0

0

0

RAM used/quota

5.95MiB / 200MiB

782MiB / 1GiB

50.8MiB / 200MiB

disk used

5.23MiB

1.22GiB

43.9MiB

Documents

Scopes & Collections

Documents

Scopes & Collections

Documents

Scopes & Collections

버킷 생성

Add Data Bucket

Name

Test_bucket

authorized users

Bucket Type

☒ Couchbase ☐ Memcached ☐ Ephemeral

Storage Backend

☒ CouchStore ☐ Magma

Memory Quota in mebibytes per server node

200 MiB

other buckets (1.39GiB) this bucket (200MiB) available (424MiB)

Advanced bucket settings

Replicas

☒ Enable 1 Number of replica (backup) copies

Warning: you do not have enough data servers or server groups to support this number of replicas.

☐ Replicate view indexes

Bucket Max Time-To-Live

☐ Enable 0 seconds

Compression Mode

☐ Off ☒ Passive ☐ Active

Conflict Resolution

☒ Sequence number ☐ Timestamp

Ejection Method

☒ Value-only ☐ Full

Bucket Priority

☒ Default ☐ High

Minimum Durability Level

none

Auto-Compaction

☐ Override the default auto-compaction settings?

Flush

☐ Enable

Cancel Add Bucket

버킷을 클릭하여 상세 정보 확인

cbdb > Buckets

ADD BUCKET

filter buckets...

name

beer-sample

pos

travel-sample

items

7,306

2,109,414

63,349

resident

100%

52.8%

100%

ops/sec

0

0

0

RAM used/quota

5.95MiB / 200MiB

782MiB / 1GiB

50.8MiB / 200MiB

disk used

5.23MiB

1.22GiB

43.9MiB

Documents

Scopes & Collections

Documents

Scopes & Collections

Documents

Scopes & Collections

Type: Couchbase

Bucket RAM Quota: 200MiB

Cluster RAM Quota: 2GiB

Replicas: disabled

Server Nodes: 1

Ejection Method: Full

Conflict Resolution: Sequence Number

Compaction: Not active

Compression: Passive

Storage Backend: CouchStore

Minimum Durability Level: none

Memory

other buckets (1.19GiB) this bucket (200MiB) available (624MiB)

Disk

other buckets (1.22GiB) this bucket (43.9MiB) available (432GiB)

Drop Compact Edit

Couchbase UI 설명 - Bucket/Scope/Collection 생성/삭제

cbdb > Buckets

filter buckets...

name	items	resident	ops/sec	RAM used/quota	disk used	
beer-sample	7,306	100%	0	5.95MiB / 200MiB	5.23MiB	Documents Scopes & Collections
pos	2,109,414	52.8%	0	782MiB / 1GiB	1.22GiB	Documents Scopes & Collections
travel-sample	63,349	100%	0	50.8MiB / 200MiB	43.9MiB	Documents Scopes & Collections

Type: Couchbase
Bucket RAM Quota: 200MiB
Cluster RAM Quota: 2GiB
Replicas: disabled

Memory

Disk

cluster quota (2GiB)

total cluster storage (460GiB)

Scope & Collection 클릭하여
상세 정보 확인

cbdb > Buckets > Scopes & Collections

travel-sample

filter scopes

scope name	collections	items	memory used	disk utilization	ops/sec	
_default	1	31.5K	20.6MiB	19.8MiB	0	Documents Add Collection
_system	2	61	129KiB	119KiB	0	Documents Add Collection
inventory			20.6MiB	19.8MiB	0	Drop Documents Add Collection

filter collections

collection name	ttl	items	memory used	disk utilization	ops/sec	
airline	0	187	34.6KiB	35.3KiB	0	Edit TTL Drop Documents
airport	0	1.96K	503KiB	455KiB	0	Edit TTL Drop Documents
hotel	0	917	3.83MiB	3.81MiB	0	Edit TTL Drop Documents
landmark	0	4.49K	2.98MiB	2.86MiB	0	Edit TTL Drop Documents
route	0	24K	13.3MiB	12.7MiB	0	Edit TTL Drop Documents

10

< prev 1 next >

tenant_agent_00

Drop Documents Add Collection

Scope 생성

Collection 생성
Document 서비스
화면으로 이동

Scope 선택하면 collection 정보 확인

Scope 삭제

```
CREATE SCOPE `travel-sample`.events  
CREATE COLLECTION `travel-sample`.inventory.city
```

Scope & Collection 생성은 SQL 로도 가능

Couchbase UI 설명 - Documents (Data Service)

The screenshot shows the cbdb Documents interface with several annotations:

- Bucket > Scope > Collection 선택**: Points to the "Keyspace" dropdown menu.
- 검색 조건**: Points to the "Document ID" and "N1QL WHERE" input fields.
- 신규 문서 추가(Add)**: Points to the "ADD DOCUMENT" button.
- 문서 수정(Update)**: Points to the "Edit Document" modal window.
- 삭제(Remove)**: Points to the delete icon in the document list.
- Key**: Points to the "id" column in the document list.
- Value (Json Documents)**: Points to the JSON document content in the list.

Couchbase UI 설명 - Query Service

Query 실행 정보 Query 편집 창 Query 모니터링 사용자 정의 함수 데이터 가져오기/내보내기

Bucket > Scope 지정

Query 실행 정보

Query 결과 보기 옵션/실행 계획 및 인덱스 권고

Schema 확인

Query 결과 확인

Query Editor

```
1 SELECT route.airline, route.schedule, route.sourceairport
2 FROM route
3 WHERE route.sourceairport = "JFK"
```

Execute Run as TX Index Advisor Explain success 4 min ago | 49.9ms | 5 docs | 13371 bytes

Results

airline	schedule	sourceairport
4M	day flight utc	JFK
0	4M762 14:13:00	
1	4M986 00:15:00	
1	4M977 13:58:00	
2	4M856 22:50:00	
2	4M745 00:03:00	
2	4M210 08:34:00	
2	4M778 16:05:00	
2	4M162 19:52:00	
3	4M107 02:23:00	
3	4M332 18:04:00	
3	4M965 02:36:00	
3	4M814 12:55:00	
4	4M965 11:45:00	
4	4M667 11:31:00	
4	4M808 04:04:00	
4	4M025 14:07:00	
5	4M349 16:12:00	
6	4M635 02:40:00	
6	4M406 04:33:00	

Explore Your Data

Common document types, field names, and sample values from your data - by bucket, scope, collection.

beer-sample

- pos
- travel-sample (16 collections)
 - _default
 - inventory (187 docs)
 - airline (100%)
 - callsign (null,string)
 - country (string)
 - iata (null,string)
 - icao (string)
 - id (number)
 - name (string)
 - type (string)
 - Indexes
 - airport (1968 docs)
 - hotel (917 docs)
 - landmark (4495 docs)
 - route (24024 docs)
 - tenant_agent_00
 - tenant_agent_01
 - tenant_agent_02

Couchbase UI 설명 - Query Monitoring

수행 중 Query 현황 Query 모니터링

The screenshot shows the Couchbase Query Monitoring interface. The top navigation bar includes 'activity', 'help', and 'Administrator'. The main header is 'cbdb > Query'. Below this, there are tabs for 'Workbench', 'Monitor' (highlighted with a red box), and 'UDF'. On the left, a sidebar lists various system components: Dashboard, Servers, Buckets, Backup, XDCR, Security, Settings, Logs, Documents, Query (selected), Indexes, Search, Analytics, Eventing, and Views. The main content area is titled 'Active Queries' with a timestamp '10:57:10 GMT+9' and a 'pause' button. It displays a table of active queries. The first row is highlighted with a red box. To the right of the table, there are filters for 'Active', 'Completed', and 'Prepared' (highlighted with a red box). Below the table, there are buttons for 'Plan', 'Edit', and 'Cancel' (highlighted with a red box).

query	node	duration	request ID	state	user
select active_requests.*, meta().plan from syste...	127.0.0.1:8091	00:00.029	6693b6e6-17be-4f14-971e-2db4b44b42bb	running	builtin:Admini...

선택(실행중,완료,준비)

실행 계획 보기(Plan)
Query 수정(Edit)
Query강제 종료(Cancel)

Couchbase UI 설명 - User Defined Function

사용자 정의 함수

JavaScript 함수 생성

사용자 정의 함수 생성

The screenshot displays the Couchbase UI's 'UDFs' (User Defined Functions) section. The top navigation bar includes 'activity', 'help', and 'Administrator'. The main header shows 'cbdb > UDFs' with tabs for 'Workbench', 'Monitor', and 'UDF'. The left sidebar lists various database management options: Dashboard, Servers, Buckets, Backup, XDCR, Security, Settings, Logs, Documents, Query, Indexes, Search, Analytics, Eventing, and Views. The main content area is divided into two sections: 'Javascript Function Libraries' and 'User-Defined Functions'. Both sections have a '+ add' button. Red dashed boxes and arrows highlight these buttons, with labels indicating they lead to 'JavaScript 함수 생성' (Create JavaScript Function) and '사용자 정의 함수 생성' (Create User-Defined Function) respectively. Below these sections, two modal windows are shown: 'Add Function' and 'Add Library'. The 'Add Function' modal includes fields for 'Namespace' (set to 'bucket.scope'), 'Function Name', 'Parameters', 'Function Type' (set to 'inline'), and 'Expression' (with a hint: 'N1QL expression, e.g. args[0] + 1'). The 'Add Library' modal includes a 'Namespace' field, a 'Library Name' search field, and a text area for the function code, which contains a sample JavaScript function:

```
1 /* a UDF library contains one or more javascript functions */
2 function add(a,b) {
3   return(a+b);
4 }
5
```

Couchbase UI 설명 - Index Service

Bucket > Scope 선택

cbdb > Indexes

Dashboard
Servers
Buckets
Backup
XDCR
Security
Settings
Logs
Documents
Query
Indexes
Search
Analytics
Eventing
Views

Bucket & Scope
travel-sample inventory view by index filter indexes...

index name	requests/sec	resident ratio	items	data size	keyspace	status
adv_activity_id_name_city_address	0	100%	4.49K	956KiB	travel-sample.inventory.land...	ready

Definition CREATE INDEX `adv_activity_id_name_city_address` ON `travel-sample`.`inventory`.`landmark` (`activity`,`id`,`name`,`city`,`address`)
Storage Mode Standard GSI
Last Scanned null

Open in Workbench Drop

Index Stats Minute

Index Mutations Remaining
1
0
11:36:30AM 11:37AM 11:37:30AM

Index Drain Rate
1/s
0/s
11:36:30AM 11:37AM 11:37:30AM

Index Resident Percent
100%
0%
11:36:30AM 11:37AM 11:37:30AM

Index RAM Used
430KiB
0B
11:36:30AM 11:37AM 11:37:30AM

Indexed Items
4.5K
0
11:36:30AM 11:37AM 11:37:30AM

Index Data Size
960KiB
0B
11:36:30AM 11:37AM 11:37:30AM

Index Disk Size
2MiB
0B
11:36:30AM 11:37AM 11:37:30AM

Index Item Size
180B
0B
11:36:30AM 11:37AM 11:37:30AM

Index Scan Latency
0.006s
0s
11:36:30AM 11:37AM 11:37:30AM

Index Request Rate
1/s
0/s
11:36:30AM 11:37AM 11:37:30AM

Index Returned Items Rate
1/s
0/s
11:36:30AM 11:37AM 11:37:30AM

Index Scan Bytes
200K/s
0/s
11:36:30AM 11:37AM 11:37:30AM

Index Service RAM Quota 1GiB
RAM Used/Remaining 335MiB/688MiB
Index Service RAM Percent 32.7%

Total Scan Rate 0/sec
Indexes Fragmentation 78.6%

beer-sample
Indexes Data Size 411KiB
Indexes Disk Size 832KiB

인덱스 확인

인덱스
사용 현황 모니터링

Couchbase UI 설명 - Query Plan 확인

Bucket > Scope 선택

Query Plan 확인

Query Plan 결과

Query Editor

```
1 SELECT route.airline, route.schedule, route.sourceairport
2 FROM route
3 WHERE route.sourceairport = "JFK"
```

Results


Indexes
route.def_inventory_route_sourceairport

Buckets
travel-sample.inventory.route route

Fields
travel-sample.inventory.route.sourceairport travel-sample.inventory.route.airline travel-sample.inventory.route.schedule route.airline

Query Plan 결과

```
graph RL
    IndexScan3[IndexScan3  
route.def_inventory_route_sourceair...] --> Fetch[Fetch  
route]
    Fetch --> Filter[Filter  
(("route"."sourceairport") = "JFK")]
    Filter --> Project[Project  
3 terms]
    Project --> Order[Order  
("route"."airline")  
5]
    Order --> Limit[Limit  
5]
```

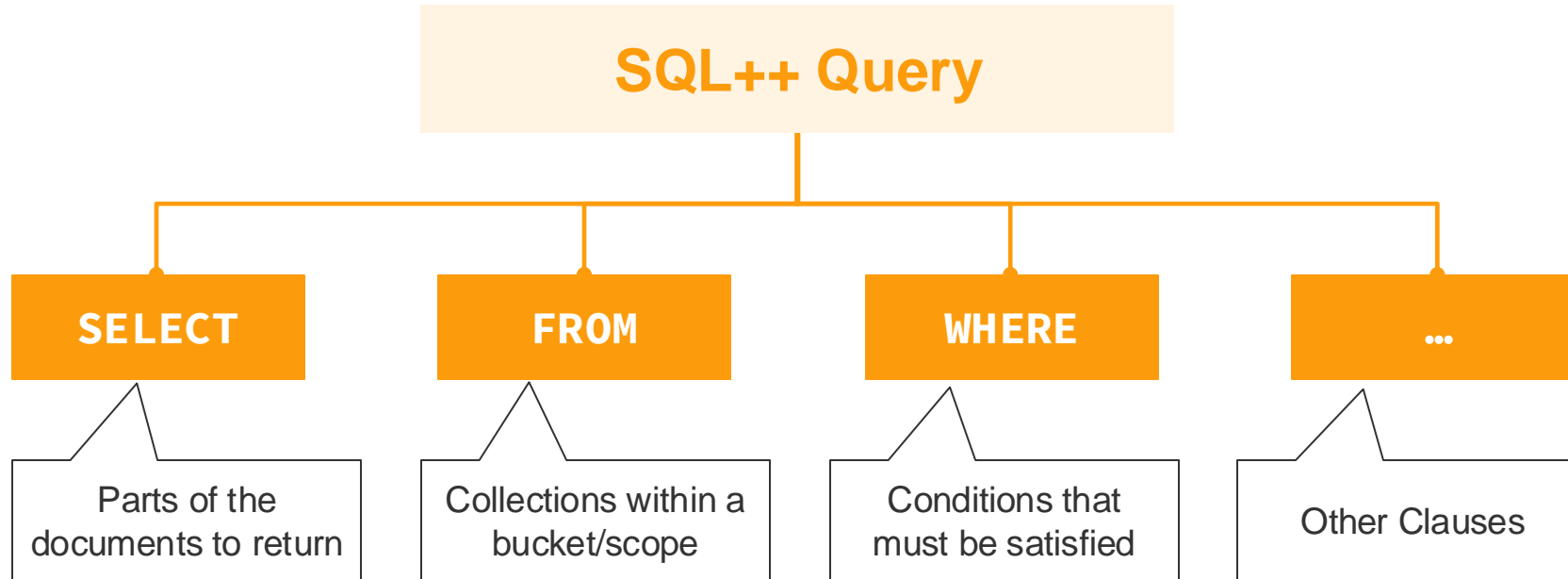



2-2.

Query/Index 서비스 소개



SQL-like queries | Looks familiar?



```
SELECT airportname, city, country
FROM `travel-sample`.inventory.airport
WHERE country = "France"
LIMIT 4;
```

OR

```
# Query Context: `travel-sample`.inventory
SELECT airportname, city, country
FROM airport
WHERE country = "France"
LIMIT 4;
```

You can set the query context in the UI or the SDK. It will resolve partial keyspace reference and simplify the request

SQL++ Queries design

Modelling data using Collections simplifies queries and indexes

```
set query_context = "default:CRM._default"
```

Set query context. Here _default scope of bucket CRM.

```
CREATE COLLECTION Orders;  
CREATE COLLECTION Individuals;
```

Create collections in that context

```
INSERT INTO Individuals (key,value)  
VALUES ("Ind:110", { "Id": "110","Name": "Mary Joe",  
                    "Email": "mj@email.com" });  
  
INSERT INTO Orders (key,value)  
VALUES ("Ord:123", { "Id": "123", "Indid": "110",  
                    "Items": [ "Qty": "1","Name": "Shoes XX"}] });
```

Insert Json docs into collections.
There is no type field.

```
CREATE INDEX Orders_Indid ON Orders(Indid);  
CREATE INDEX Individuals_Id ON Individuals(Id);
```

Create indexes into a collection.
There is no WHERE type='Orders'.

```
SELECT i.Name, o.Items  
FROM Orders o  
JOIN Individuals i ON i.Id = o.Indid;
```

Query with JOIN between collections

SQL-like queries | Includes DML (Data Manipulation Language)

DML Statement	Examples	Description
INSERT	INSERT INTO `travel-sample`.inventory.hotel (KEY, VALUE) VALUES ("hotel_9999", { "type" : "hotel", "name" : "new hotel" });	Create one or more new documents into a collection. Each INSERT statement requires a unique document key and well-formed JSON as values.
DELETE	DELETE FROM `travel-sample`.inventory.airline f WHERE f.callsign = "AIR-X";	Immediately removes the specified document from your collection.
UPDATE	UPDATE `travel-sample`.inventory.airport SET city = "San Francisco" WHERE lower(city) = "san francisco";	Replaces a document that already exists with updated values.
UPSERT	UPSERT INTO `travel-sample`.inventory.hotel (KEY, VALUE) VALUES ("key1", { "type" : "hotel", "name" : "new hotel" });	Insert a new record or update an existing one. If the document doesn't exist it will be created. UPSERT is a combination of INSERT and UPDATE.
MERGE	(See Next slide)	

DML statements allow you to create, delete, and modify the **data stored in JSON documents** by specifying and executing simple commands



SQL-like queries | MERGE

Finds all routes for airline BA whose source airport is in France.

- *If any flights are using equipment 319, they are updated to use equipment 797.*
- *If any flights are using equipment 757, they are deleted.*

Collection: 'route'

```
{
  "id": "14452",
  "type": "route",
  "airline": "BA",
  "airlineid": "airline_1355",
  "sourceairport": "BOD",
  "destinationairport": "LGW",
  "equipment": "734 319",
  "stops": "0",
  "distance": 99.89861063028253
  "schedule": [
    {"day": 0, "utc": "19:41:00", "flight": "BA977"},
    {"day": 0, "utc": "14:03:00", "flight": "BA938"},
    {"day": 1, "utc": "08:40:00", "flight": "BA394"},
    {"day": 1, "utc": "06:54:00", "flight": "BA427"},
    {"day": 1, "utc": "06:00:00", "flight": "BA916"}
  ]
}
```

Collection: 'airport'

```
{
  "id": 1264,
  "type": "airport",
  "airportname": "Merignac",
  "city": "Bordeaux",
  "country": "France",
  "faa": "BOD",
  "icao": "LFBD",
  "tz": "Europe/Paris",
  "geo": {
    "lat": 44.828335,
    "lon": -0.715556,
    "alt": 162
  }
}
```

Before updating or deleting some routes,
you first need to JOIN route and airport because the
country of the source airport is in the airport collection

SQL-like queries | MERGE

```
MERGE INTO `travel-sample`.inventory.route
USING `travel-sample`.inventory.airport
ON route.sourceairport = airport.faa
WHEN MATCHED THEN
  UPDATE
    SET route.old_equipment = route.equipment,
        route.equipment = "797"
    WHERE airport.country = "France" AND route.airline = "BA" AND CONTAINS(route.equipment, "319")
WHEN MATCHED THEN
  DELETE
    WHERE airport.country = "France" AND route.airline = "BA" AND CONTAINS(route.equipment, "757")
RETURNING route.old_equipment, route.equipment, airport.faa;
```

Collection to modify

Collection to join with

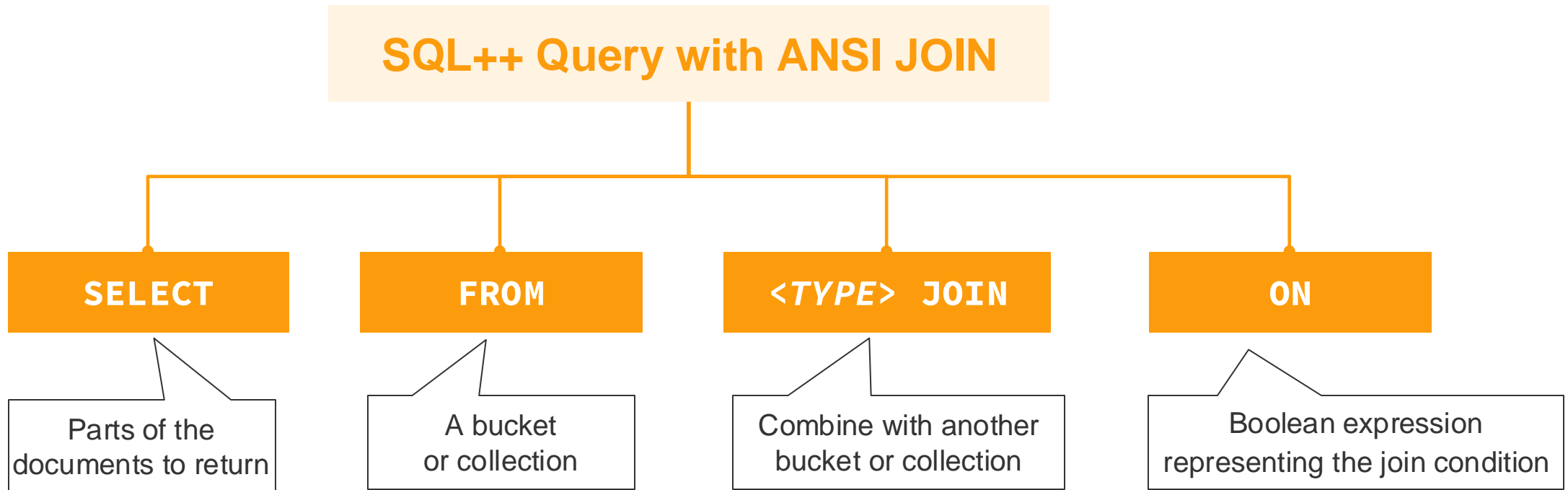
JOIN predicate

Update the routes using equipment 319

Delete routes having 757 equipment

- Provides the ability to modify a collection based on the results of a join with another collection.
- Based on a match or no match in the join, actions can be INSERT, UPDATE, DELETE.
- Multiple actions can be specified.

SQL-like queries | ANSI JOINS



SQL-like queries | ANSI JOINS

JOIN TYPE	Examples
ANSI JOIN	SELECT .. FROM customer c JOIN orders o ON o.customer_id = c.id
ANSI JOIN Complex	SELECT .. FROM airline JOIN route ON route.airlineid = "airline_" toString(airline.id) AND route.type = "route"
ANSI JOIN with IN CLAUSE	SELECT .. FROM `travel-sample` route JOIN airport ON airport.faa IN [route.sourceairport, route.destinationairport] AND airport.type = "airport"
ANSI LEFT OUTER	SELECT .. FROM airport LEFT JOIN route ON airport.faa = route.sourceairport AND route.type = "route"
ANSI JOIN with HASH JOIN	SELECT .. FROM airport JOIN route USE HASH (build) ON airport.faa = route.sourceairport AND route.type = "route"

ANSI JOIN can join arbitrary fields of the documents and can be chained together.

SQL-like queries | Arrays

Retrieve the details of KL flight schedules from Albuquerque (ABQ) to Atlanta (ATL) if any of the flights are after 23:40.

```
SELECT *
FROM `travel-sample`.inventory.route
WHERE airline="KL"
      AND sourceairport="ABQ"
      AND destinationairport="ATL"
      AND ANY departure IN schedule
          SATISFIES departure.utc > "23:40" END;
```



```
[
  {
    "travel-sample": {
      "airline": "KL",
      "airlineid": "airline_3090",
      "destinationairport": "ATL",
      "distance": 2038.3535078909663,
      "equipment": "757 320",
      "id": 36159,
      "schedule": [
        {"day": 0, "flight": "KL938", "utc": "03:54:00"},
        // ...
        {"day": 5, "flight": "KL169", "utc": "23:41:00"},
        // ...
        {"day": 6, "flight": "KL636", "utc": "17:40:00"}
      ],
      "sourceairport": "ABQ",
      "stops": 0,
      "type": "route"
    }
  }
]
```

- Range predicates (ANY, EVERY) enable you to evaluate expressions over every element in an array
- They are particularly useful when used to evaluate expressions over an array of objects

SQL-like queries | Subqueries

Find total number of airports by country where each city has more than 5 airports.

```
SELECT t1.country,  
       array_agg(t1.city),  
       sum(t1.city_cnt) as apnum  
FROM  
  (SELECT city,  
          city_cnt,  
          array_agg(airportname) as apnames,  
          country  
   FROM `travel-sample`.inventory.airport  
   GROUP BY city,  
            country LETTING city_cnt = count(city)  
  ) AS t1  
WHERE t1.city_cnt > 5  
GROUP BY t1.country;
```



```
[  
  {  
    "$1": [  
      "Paris"  
    ],  
    "apnum": 9,  
    "country": "France"  
  },  
  {  
    "$1": [  
      "London"  
    ],  
    "apnum": 13,  
    "country": "United Kingdom"  
  },  
  {  
    "$1": [  
      "Houston",  
      "New York",  
      "San Diego"  
    ],  
    "apnum": 22,  
    "country": "United States"  
  }  
]
```

- A subquery is a query within another query
- Subqueries can be embedded anywhere a valid expression can go

SQL-like queries | Built-in Functions

Category	Functions
Aggregate functions	ARRAY_AGG() ARRAY_AGG(DISTINCT) AVG(), AVG(DISTINCT) COUNT() COUNT(DISTINCT) MAX() MIN() SUM() SUM(DISTINCT)
Object functions	OBJECT_LENGTH() OBJECT_NAMES() OBJECT_PAIRS() OBJECT_INNER_PAIRS() OBJECT_VALUES() OBJECT_INNER_VALUES() OBJECT_ADD() OBJECT_PUT() OBJECT_REMOVE() OBJECT_UNWRAP()
Array functions	ARRAY_APPEND() ARRAY_AVG() ARRAY_CONCAT() ARRAY_CONTAINS() ARRAY_COUNT() ARRAY_DISTINCT() ARRAY_IFNULL() ARRAY_LENGTH() ARRAY_MAX() ARRAY_MIN() ARRAY_POSITION() ARRAY_PREPEND() ARRAY_PUT() ARRAY_RANGE() ARRAY_REMOVE() ARRAY_REPEAT() ARRAY_REPLACE() ARRAY_REVERSE() ARRAY_SORT() ARRAY_SUM()
Comparison functions	GREATEST() LEAST()
Conditional functions	IFMISSING() IFMISSINGORNULL() IFNULL() MISSINGIF() NULLIF() IFINF() IFNAN() IFNANORINF() NANIF() NEGINFIF() POSINFIF()
Number functions	ABS() ACOS() ASIN() ATAN() ATAN2() CEIL() COS() DEGREES() E() EXP() LN() LOG() FLOOR() PI() POWER() RADIANS() RANDOM() ROUND() SIGN() SIN() SQRT() TAN() TRUNC()
Date functions	CLOCK_MILLIS() CLOCK_STR() DATE_ADD_MILLIS() DATE_ADD_STR() DATE_DIFF_MILLIS() DATE_DIFF_STR() DATE_PART_MILLIS() DATE_PART_STR() DATE_TRUNC_MILLIS() DATE_TRUNC_STR() STR_TO_MILLIS() MILLIS_TO_STR() MILLIS_TO_UTC() MILLIS_TO_ZONE_NAME() NOW_MILLIS() NOW_STR() STR_TO_MILLIS() STR_TO_UTC() STR_TO_ZONE_NAME()
JSON functions	DECODE_JSON() ENCODE_JSON() ENCODED_SIZE() POLY_LENGTH()
Meta and UUID functions	BASE64() BASE64_ENCODE() BASE64_DECODE() META() UUID()
Pattern-matching functions	REG_CONTAINS() REG_LIKE() REG_POSITION() REG_REPLACE()
String functions	CONTAINS() INITCAP() LENGTH() LOWER() LTRIM() POSITION() REPEAT() REPLACE() RTRIM() SPLIT() SUBSTR() TITLE() TRIM() UPPER()
Type-Checking Functions	ISARRAY() ISATOM() ISBOOLEAN() ISNUMBER() ISOBJECT() ISSTRING() TYPE()
Type-Conversion Functions	TOARRAY() TOATOM() TOBOOLEAN() TONUMBER() TOOBJECT() TOSTRING()

SQL-like queries | User Defined Functions (UDF)

Simple Inline Example

```
CREATE FUNCTION to_meters(...) {  
  args[0] * 0.3048  
};
```

```
SELECT airportname, ROUND(to_meters(geo.alt)) AS mamsl  
FROM `travel-sample`.inventory.airport  
LIMIT 5;
```



```
[  
  {"airportname": "Calais Dunkerque", "mamsl": 4},  
  {"airportname": "Peronne St Quentin", "mamsl": 90},  
  {"airportname": "Les Loges", "mamsl": 130},  
  {"airportname": "Couterne", "mamsl": 219},  
  {"airportname": "Bray", "mamsl": 111}  
]
```

Inline Example with SQL++ query inside

```
CREATE FUNCTION locations(vActivity) {  
  (SELECT id, name, address, city  
   FROM `travel-sample`.inventory.landmark  
   WHERE activity = vActivity) };
```

```
SELECT l.name, l.city  
FROM locations("eat") AS l  
WHERE l.city = "Gillingham";
```



```
[  
  {"city": "Gillingham", "name": "Hollywood Bowl"},  
  {"city": "Gillingham", "name": "Thai Won Mien"},  
  {"city": "Gillingham", "name": "Spice Court"},  
  {"city": "Gillingham", "name": "Beijing Inn"},  
  {"city": "Gillingham", "name": "Ossie's Fish and  
  Chips"}  
]
```

- Inline functions are defined using SQL++ expressions, including subqueries.
- You can name and reuse complex or repetitive expressions in order to simplify your queries.

SQL-like queries | UDF with JavaScript & SQL++

Create a Javascript function in a Library

JavaScript library

```
Edit Library X
demodml Q
1 /* a UDF library contains one or more javascript functions */
2 = function getairportname(airportcode)
3   var query = SELECT a.airportname FROM 'travel-sample'.inventory.airport
4   a WHERE a.icao=$airportcode;
5   let acc = [];
6   for (const row of query) {
7     acc.push(row);
8   }
9
10  return(acc);
11 }
```

Create a UDF Function that references it

```
Edit Function X
Function Name
getname
Namespace
global
Parameters
--
Function Type
javascript
Javascript Library
demodml
Library Function Name
getairportname
Cancel Save Function
```

Use the UDF function in SQL++ statements

Query Editor

```
1 select getname(a.icao) from 'travel-sample'.inventory.airport a limit 1;
```

Execute Run as TX Index Advisor Explain

success 2 min ago 247.5ms | 1 docs | 111 bytes

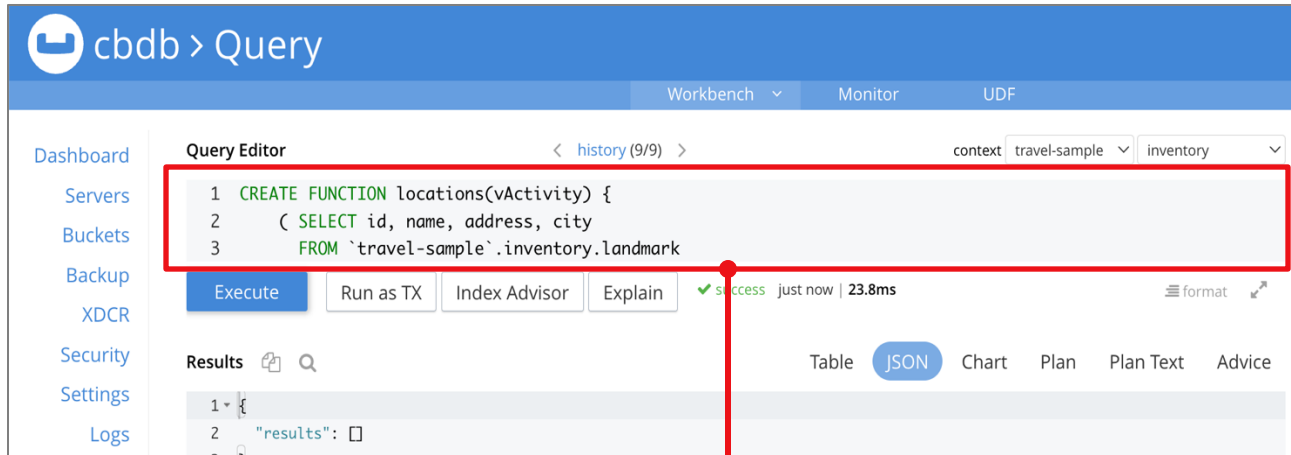
Results Table JSON Chart Plan Plan Text Advice

```
1 = {
2 = {
3 =   "$1": [
4 =   {
5 =     "airportname": "Calais Dunkerque"
6 =   }
7 = ]
}
```

- UDFs with JavaScript allows developers to provide custom functions to extend SQL++ capabilities

SQL-like queries | UDF with SQL++

참고 : <https://docs.couchbase.com/server/current/n1ql/n1ql-language-reference/userfun.html>



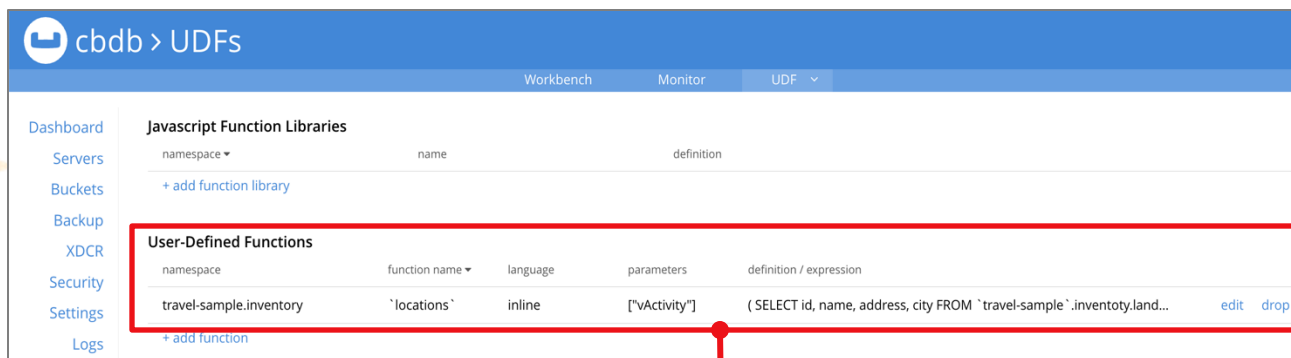
The screenshot shows the Couchbase Query Editor interface. The 'Query Editor' tab is active, and the 'context' is set to 'travel-sample' and 'inventory'. The query editor contains the following SQL code:

```
1 CREATE FUNCTION locations(vActivity) {  
2   ( SELECT id, name, address, city  
3     FROM `travel-sample`.inventory.landmark
```

The 'Execute' button is highlighted, and the status bar shows 'success just now | 23.8ms'. The 'Results' section shows a single row with the value 'results': [].

1. UDF 생성

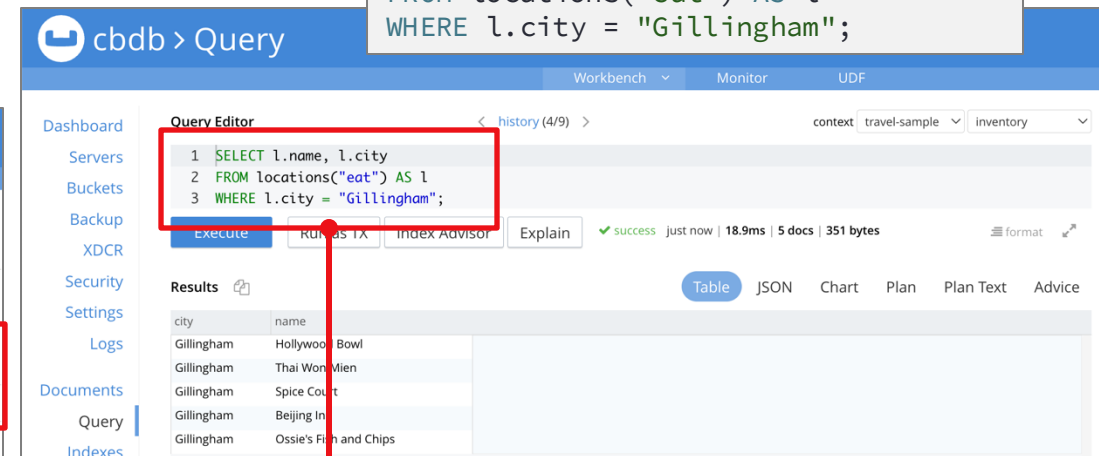
```
CREATE FUNCTION locations(vActivity) {  
  ( SELECT id, name, address, city  
    FROM landmark  
    WHERE activity = vActivity) };
```



The screenshot shows the Couchbase UDFs page. The 'User-Defined Functions' section is highlighted, showing a table with the following information:

namespace	function name	language	parameters	definition / expression
travel-sample.inventory	'locations'	inline	["vActivity"]	(SELECT id, name, address, city FROM `travel-sample`.inventory.land...

2. UDF 확인



The screenshot shows the Couchbase Query Editor interface. The 'Query Editor' tab is active, and the 'context' is set to 'travel-sample' and 'inventory'. The query editor contains the following SQL code:

```
1 SELECT l.name, l.city  
2 FROM locations("eat") AS l  
3 WHERE l.city = "Gillingham";
```

The 'Execute' button is highlighted, and the status bar shows 'success just now | 18.9ms | 5 docs | 351 bytes'. The 'Results' section shows a table with the following data:

city	name
Gillingham	Hollywood Bowl
Gillingham	Thai Worn Mien
Gillingham	Spice Court
Gillingham	Beijing In
Gillingham	Ossie's Fish and Chips

3. UDF 실행

SQL-like queries | Search Functions

Find the name of the hotels in United Kingdom where the reviews match the term 'bathrobes'

```
SELECT t1.name, meta().id
FROM `travel-sample`.inventory.hotel AS t1
WHERE SEARCH(t1, {
  "match": "bathrobes",
  "field": "reviews.content",
  "analyzer": "standard"
})
AND country="United Kingdom"
LIMIT 3;
```



```
[
  {
    "id": "hotel_12068",
    "name": "Castle Hotel"
  },
  {
    "id": "hotel_18819",
    "name": "Bistro Prego With Rooms"
  },
  {
    "id": "hotel_3622",
    "name": "Premier Inn Birmingham Central East"
  }
]
```

- Search functions enable you to use full text search (FTS) queries directly within a SQL++ query.
- It is recommended that you create suitable full text indexes for the searches that you need to perform.

SQL-like queries | Prepared Statement

Positional Parameters

```
PREPARE NumParam AS  
SELECT * FROM `travel-sample`.inventory.hotel  
WHERE city=$1 AND country=$2;
```

```
EXECUTE NumParam  
USING ["Paris", "France"];
```

Named Parameters

```
PREPARE NameParam AS  
SELECT * FROM `travel-sample`.inventory.hotel  
WHERE city=$city AND country=$country;
```

```
EXECUTE NameParam  
USING {"city": "Paris", "country": "France"};
```

- You can add placeholder parameters to a statement, so that you can supply variable values when you run the statement.
- If you need to run a statement more than once, you can prepare the execution plan for the statement.

SQL-like queries | Transactions

Ensures database consistency when multiple documents are updated in a single or multiple SQL++ statements

START TRANSACTION;

```
UPDATE customer SET balance = balance + 100 WHERE cid = 4872;  
SELECT cid, name, balance FROM customer;
```

SAVEPOINT s1;

```
UPDATE customer SET balance = balance - 100 WHERE cid = 1924;  
SELECT cid, name, balance FROM customer;
```

ROLLBACK WORK TO SAVEPOINT s1;

```
SELECT cid, name, balance FROM customer;
```

COMMIT;

Couchbase provides the following statements for transactions:

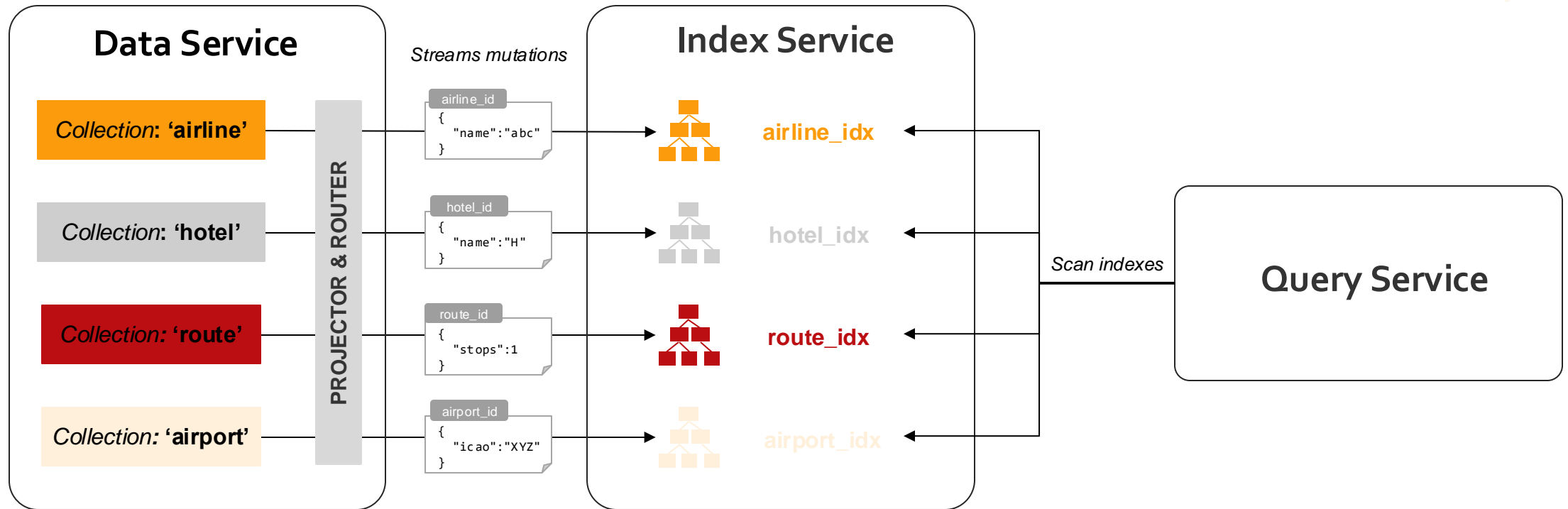
- BEGIN TRANSACTION
- SET TRANSACTION (optional)
- SAVEPOINT
- ROLLBACK TRANSACTION
- COMMIT TRANSACTION

Built-In Indexer | For better queries performance

Indexes will efficiently look up documents meeting specified criteria

Index Type	Definition
Primary	Index on the document key on a whole bucket or collection, to support full scans
Named Primary	Primary index with an assigned name, to allow multiple primary indexes in a cluster
Secondary	Index on a field (key-value pair) or document-key
Composite	Index on more than one field
Functional	Index on values resulting from a function of expression applied to a field
Array	Index on individual elements of array fields
Partial	Index on filtered subset of documents in a bucket
Covering	Describes any index which fully responds to a query without the need for a document fetch
Adaptive	Describes an indexing feature providing an arrayed approach to generically indexing all or specific doc fields, to support increased query flexibility

Built-In Indexer | Continuous Ingestion from the Data Service



- Data Service streams copies of all mutations that occur to documents to the relevant Indexes.
- Indexes are constantly and automatically updated. They are *eventually consistent* with changes on documents.
- Queries are forwarded to the relevant indexes and done based on indexed informations.

Built-In Indexer | Query Consistency

Indexes are eventually consistent with respect to data mutations. You can specify consistency-options per query.

not_bounded

(default) Will not wait for any indexes to finish updating before running the query and returning results. The results are returned **quickly**, but the query will not return any documents that are yet to be indexed.

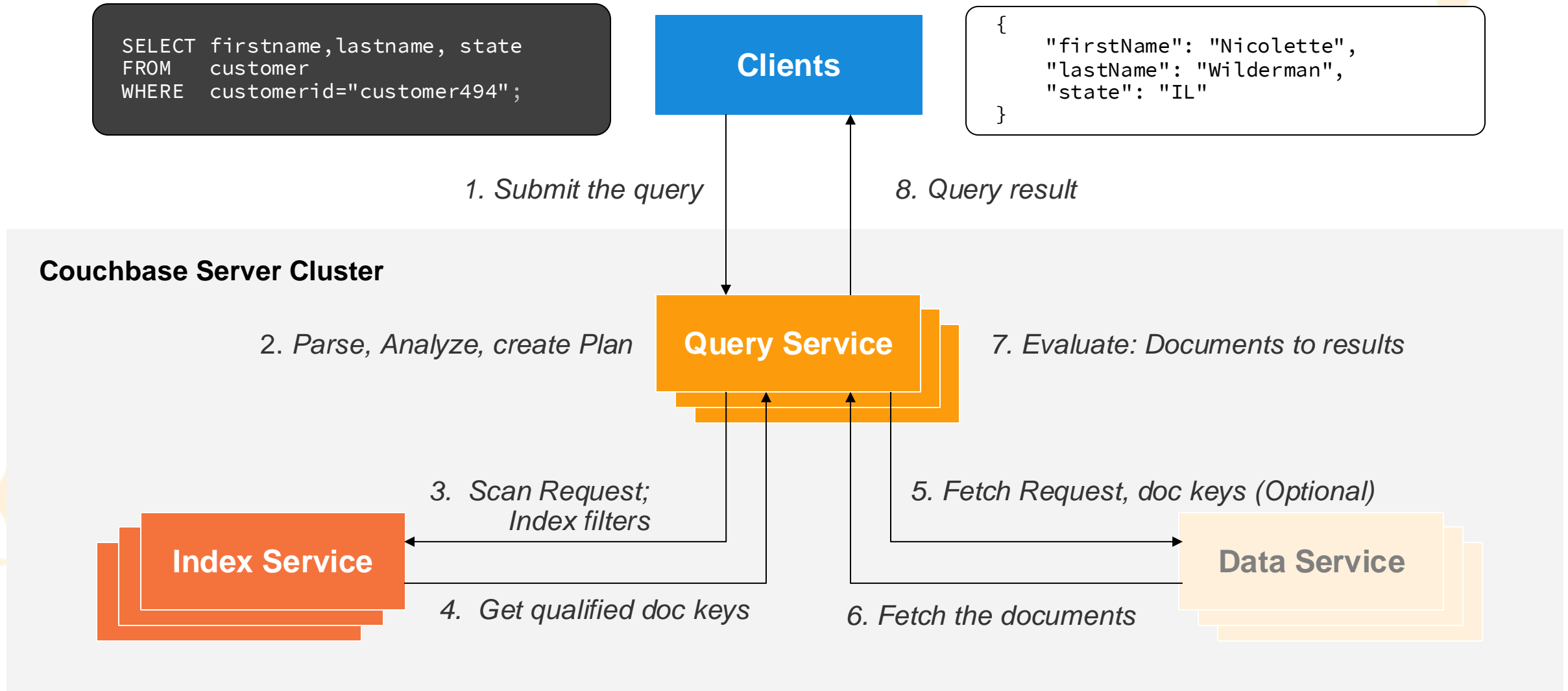
request_plus

All document changes and index updates are processed before the query is run. Select this when **consistency** is always more important than performance.

at_plus

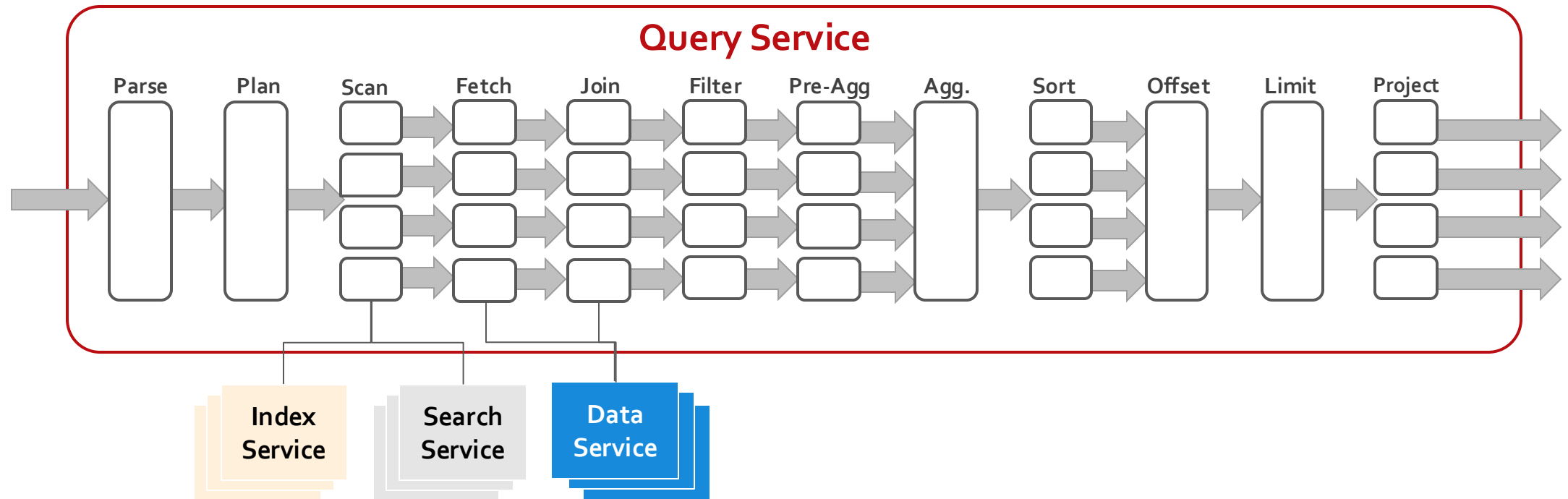
A "read your own write" (**RYOW**) option, which means it just waits for the new documents that you specify to be indexed, rather than an entire index of multiple documents.

Queries & Indexes | SQL++ Query Execution Flow



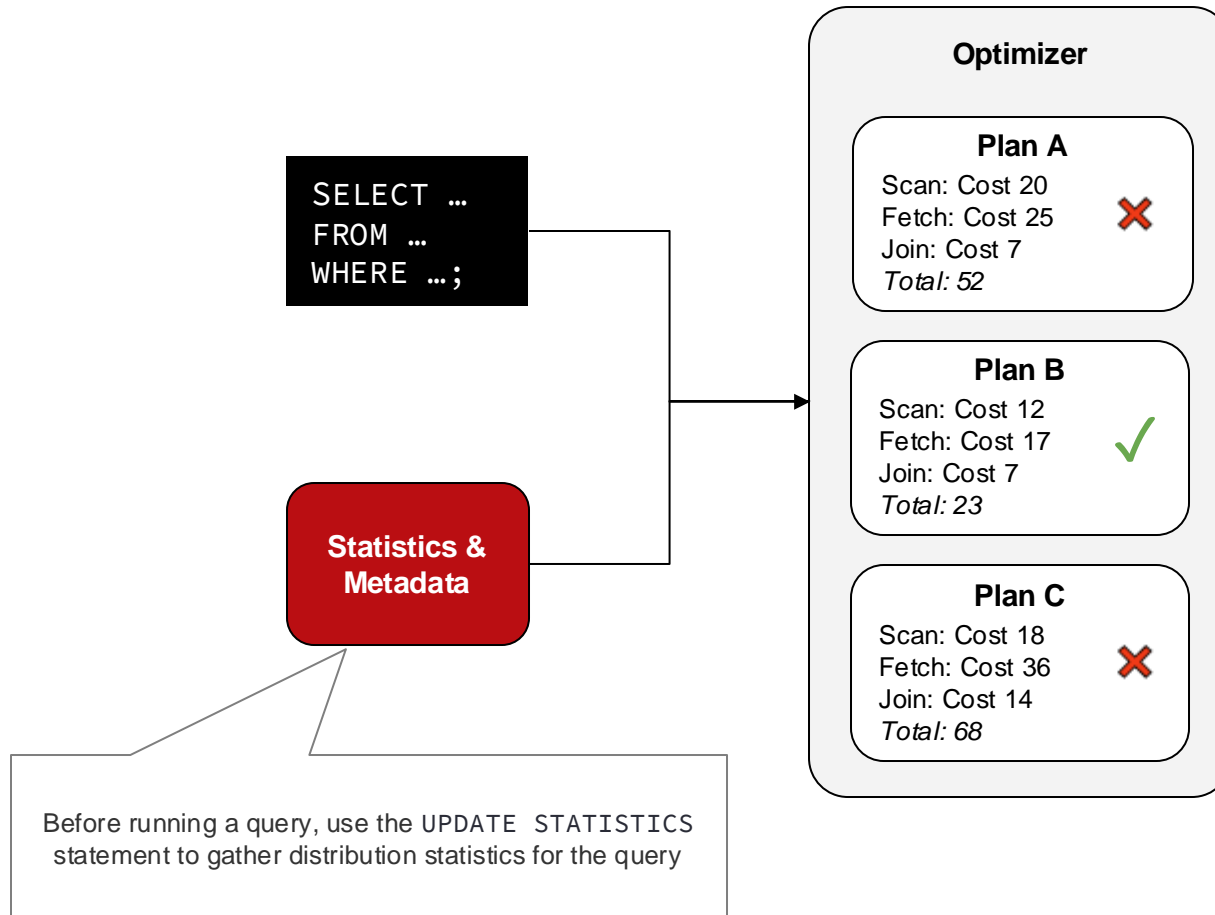
High Performance Query Engine

The degree of parallelism is represented by the vertically aligned groups of right-pointing arrows.



Cost Based Optimizer

CBO generates the optimal access path based on statistics without the need to provide optimizer hint

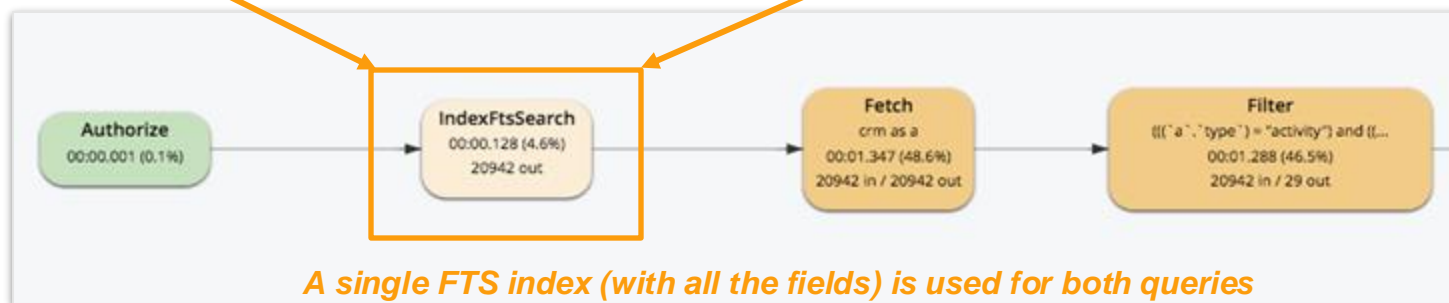


1. Estimate the cost of each plan based on the available statistics on data indexes
2. Choose the one that optimize the query latency

Flex indexing

```
SELECT *  
FROM crm a  
USE INDEX (USING FTS)  
WHERE a.type='activity'  
AND ( a.dept = 'iA88' OR a.region > '59416' )  
AND a.priority = 'High'  
AND ( a.act_date BETWEEN '2022-01-01' AND '2022-08-31'  
      OR a.event.location = 'Moscone Center' )  
AND ( a.account.id = 'acc100' OR a.owner.name = 'Amanda Morrison')
```

```
SELECT *  
FROM crm a  
USE INDEX (USING FTS)  
WHERE a.type='activity'  
AND a.priority = 'High'  
AND a.act_date > '2020-01-31'  
AND a.owner.name = 'Amanda Morrison')
```



- Flex Index allows query service to **leverage Search index, using only the standard SQL++ predicates**
- Can greatly **reduce the number of indexes** and resource needed to support ad hoc queries.
- Existing applications can benefit without any query modifications (parameter use_fts)

2-3. 실습



실습

	실습 항목	상세 실습 내용	기타
1	Data Export	1. Query Workbench 2. SELECT meta().id, * FROM `travel-sample`.inventory.airline 실행 3. 우측 상단의 Export 선택 후, airline.json 파일로 export 수행	airline, airport, hotel, route, landmark
2	Bucket, Scope, Collection 생성	1. travel 버킷 생성 2. inventory Scope 생성 3. airline, airport, hotel, route, landmark Collection 생성	Bucket 메모리 200MB
3	Data Import	1. Document 페이지 이동. 2. Import 선택 3. 위 1항에서 Export 한 json 파일 Import 수행, Document ID 는 id 로 지정	airline, airport, hotel, route, landmark
4	Index 생성	• travel-sample 버킷에 있는 index 참고하여, 생성.	
5	Query 수행	• 43, 44 페이지 SQL 수행	
6	Query Plan 확인	• 5항 SQL 수행 수 Query Plan 확인	

Export Query / Data / History
Export 화면

Choose Export

- ☒ Current query results (JSON)
- ☐ Current results as tab-separated (text)
- ☐ Query history (JSON)
- ☐ Query history including results (JSON)
- ☐ Current Query Statement (txt)

Filename
.json

Cancel Save

cbdb > Documents
Import 화면

Dashboard
Servers
Buckets
Backup
XDCR
Security
Settings
Logs
Documents
Query
Indexes
Search
Analytics
Eventing
Views

Select File to Import...
tab/comma delimited or JSON list/lines
file format details

Parse File As
JSON List
found 187 records

Keyspace
bucket.scope.collection
travel
inventory
airline

Import With Document ID
☒ UUID
☒ Value of Field:
id

For faster performance, or datasets greater than 100MB, use the command-line tool cbinport.

```
cbinport json --format list -c http://localhost:8091 -u <login> -p <password> -d 'file://airport.json' -b 'travel' --scope-collection-exp 'inventory.airline' -g %id%
```

command generated from your file, destination, and ID selections on the left

File Contents
Raw File
Parsed Table
Parsed JSON

airline	country	iata	icao	id	name	type	id
MILE-AIR	United States	Q5	MLA	10	40-Mile Air	airline	airline_10
callsign	country	iata	icao	id	name	type	airline_10123
TXW	United States	TQ	TXW	10123	Texas Wings	airline	airline_10226
callsign	country	iata	icao	id	name	type	airline_10642
atfly	United States	A1	A1F	10226	Atfly	airline	airline_10748
callsign	country	iata	icao	id	name	type	airline_10765
null	United Kingdom	null	JRB	10642	Jc royal.britannica	airline	airline_109
callsign	country	iata	icao	id	name	type	airline_112
LOCAIR	United States	ZQ	LOC	10748	Locair	airline	
callsign	country	iata	icao	id	name	type	
SASQUATCH	United States	KS	SOH	10765	SeaPort Airlines	airline	
callsign	country	iata	icao	id	name	type	
ACE AIR	United States	KO	AER	109	Alaska Central Express	airline	
callsign	country	iata	icao	id	name	type	

SQL Sample from Capella Playground

```
SELECT route.airline, route.schedule, route.sourceairport
FROM route
WHERE route.sourceairport = "JFK"
ORDER BY route.airline LIMIT 5;
```

```
SELECT DISTINCT airline.name, airline.callsign, route.schedule,
route.sourceairport, route.destinationairport
FROM route
INNER JOIN airline
ON route.airlineid = META(airline).id
WHERE route.sourceairport = "JFK"
AND route.destinationairport = "SFO"
AND airline.callsign = "UNITED";
```

```
SELECT h.name, h.address
FROM hotel h
WHERE h.city = 'San Francisco'
AND ANY r IN h.reviews
SATISFIES r.ratings.Overall > 3 END
```

```
SELECT h.city, AVG(r.ratings.Overall) AS hotel_rating
FROM hotel AS h
UNNEST h.reviews AS r
WHERE h.country = "United Kingdom"
GROUP BY h.city
HAVING AVG(r.ratings.Overall) > 4.9
```

```
SELECT h.city, h.name,
AVG(r.ratings.Overall) AS avg_rating,
RANK() OVER (PARTITION BY h.city ORDER BY AVG(r.ratings.Overall) DESC) as city_rank
FROM hotel as h
UNNEST h.reviews r
WHERE h.country = "France"
GROUP BY h.city, h.name
```

SQL Sample from Capella Playground

```
WITH AirlineDestinations AS (  
  SELECT r.destinationairport,  
  ARRAY_COUNT(r.schedule) AS flights  
  FROM `travel-sample`.inventory.route AS r  
  WHERE r.airline = "UA"  
  GROUP BY r.destinationairport, r.schedule  
)  
SELECT ad.destinationairport, SUM(ad.flights) AS total_flights  
FROM AirlineDestinations AS ad  
GROUP BY ad.destinationairport  
ORDER BY ad.destinationairport;
```

```
select a.name, count(1) as numRoutes  
from route r  
join airline a on r.airlineid = meta(a).id  
group by a.name  
having count(1) > 2000  
order by count(1) desc;
```

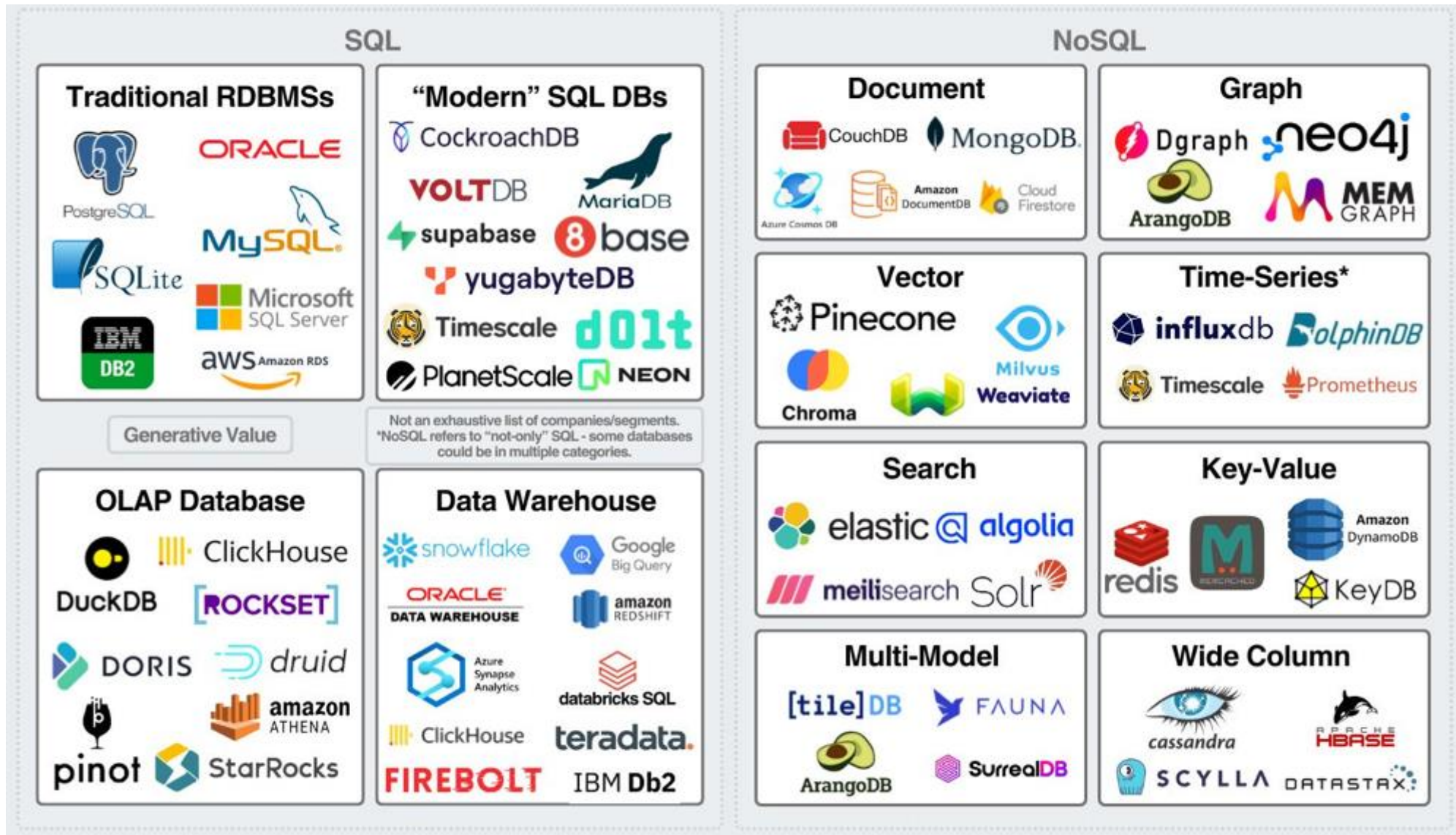
```
SELECT h.city,  
  ARRAY_AGG(h.name)  
  FILTER (WHERE ANY AND EVERY r IN h.reviews  
    SATISFIES r.ratings.Overall >= 3 END) AS good_hotels,  
  ARRAY_AGG(h.name)  
  FILTER (WHERE ANY r IN h.reviews  
    SATISFIES r.ratings.Overall < 3 END) AS other_hotels  
FROM hotel as h  
GROUP BY h.city;
```

```
select meta().id as _id, * from route
```

Appendix. JSON 데이터 모델



데이터 모델 : SQL(Table)과 NoSQL(Real World)



데이터 모델 : JSON 도큐먼트

JSON은 텍스트로 이루어져 있으므로, 사람과 기계 모두 읽고 쓰기 쉽다. 프로그래밍 언어와 플랫폼에 독립적이므로, 서로 다른 시스템 간에 객체를 교환하기에 좋다.

- JSON 도큐먼트의 장점
 - 단일 도큐먼트 내에 다양한 정보를 계층 구조를 활용하여 저장
 - 정보 추가/삭제가 유연한 구조 제공
 - 데이터 전달을 위한 표준 인터페이스 역할
- RDB와 차별점
 - 여러 테이블로 분리, 저장되는 데이터를 단일 도큐먼트에 저장
 - 테이블 간 조인을 최소화하여 데이터 처리 속도 향상

SQL, Table

이 력 서

1. 기초자료

성명	홍길동	성별	남	생년월일	1990-01-01
주민등록번호	000000-000000000				
E-mail	bizforms@bizforms.co.kr				
전화번호	02-000-0000	휴대폰	000-123-4567		
우편번호	00000	주소	00번지		
주	00	시	00번지		
본	00	구	00번지		
호적관계	호주	관계	장남		

2. 학력사항

년/월/일	학	교	명	비	고
2000-02	00	정보	공업고등학교 졸업		

3. 경력사항

000.00	담당
--------	----

4. 개인능력

관련내용	비고
리눅스(레드햇기반)운영	
samba서버 운영(nfs기반)	
apache웹서버 운영	
sendmail서버 운영	
shell 스크립트(bash) 작성 가능	
perl 및 C도 기본적으로 이해 가능	
DNS서버 운영	
WINDOWS2000 서버 운영 및 BS, AD(분산파일시스템), DNS	
NAT, DHCP, SMTP, POP3 등의 service 운영가능	
전반적인 네트워크구성 이해(TCP/IP기반)	

기초정보

학력사항

경력사항

개인능력

하드웨어에 맞춘 데이터 모델

NoSQL, JSON 도큐먼트

KEY : 1001

```
{
  "성명": "홍길동",
  "주소": "서울시 00구 00동 000-000",
  "E-mail": "HongKildong@couchbase.com",
  ...
  "학력사항": [
    {
      "졸업년도": "2019년",
      "학교명": "00정보 공업고등학교"
    }
  ],
  "경력사항": [
    {
      "기간": "2019 ~ 현재",
      "관련내용": "XX글로벌 IT팀 Unix 서버 담당"
    }
  ],
  "개인능력": [
    {
      "관련내용": "리눅스 운영"
    },
    {
      "비고": NULL
    }
  ],
  ...
}
```

실제 세계에 맞춘 데이터 모델

논리 / 물리 모델

- RDBMS와 유사한 구조의 논리 계층 구조로 구성하여 편리한 데이터 관리
- Data 서비스를 완전 메모리DB로 사용도 가능하며 용도에 따라 물리 저장 방식을 선택할 수 있음

RDBMS	Couchbase
Server	Cluster
Database	Bucket
Schema	Scope
Table	Collection
Row	Document (JSON)
Value	Sub-Document, Array

Feature	Ephemeral Bucket	Couchbase Bucket	Magma Bucket
Bucket memory quota (per node)	Min 256MB	Min 256MB	Min 1024MB
Max Object Size	20MB	20MB	20MB
Persistence	no	yes	yes
Replication and XDCR	yes	yes	yes
Encrypted data access	yes	yes	yes
Rebalance	yes	yes	yes
N1QL, Seach, Analytics, Eventing	yes	yes	yes
Indexing	yes	yes	yes
Backup	yes	yes	yes

관계형 vs. 다큐멘터 데이터 모델



관계형 데이터 모델

Required 정규화

- Schema enforced by the database (스키마가 데이터베이스에 의해 강제됨)
- Same fields required in all records (모든 레코드는 같은 필드를 가져야 함)



- Optimized for data entry(데이터 구성에 최적화)
- Reduced duplicated data (중복 감소, 제거)
- Minimize data inconsistencies (데이터 불일치 최소화)

다큐멘트 데이터 모델

Relaxed 정규화

- Schema inferred from structure (시키마는 구조로부터 추론됨)
- Fields may vary, be duplicate or missing (필드에 어떤 제약도 없음)



- Optimized based on access patterns (사용 패턴에 최적화)
- Flexible and agile development(유연하고 빠른 개발)
- Supports clustered, scalable architecture(확장성 보장)

Real World Data 를 JSON으로 표현

Customer		
CustomerID	Name	DoB
CBL2017	Jane Smith	1990-01-30

Customer Document Key: CBL2017

```
{  
  "Name" : "Jane Smith",  
  "DOB"  : "1990-01-30"  
}
```

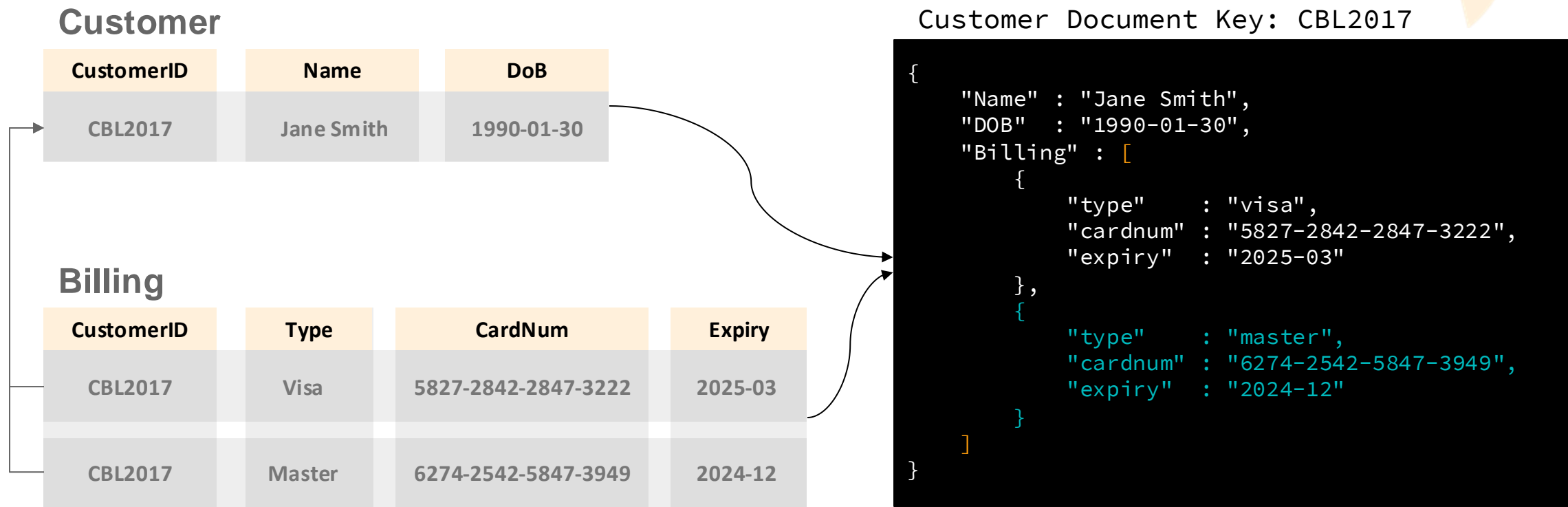
OR

Customer Document Key: CBL2017

```
{  
  "Name" : {  
    "fname": "Jane",  
    "lname": "Smith"  
  },  
  "DOB"  : "1990-01-30"  
}
```

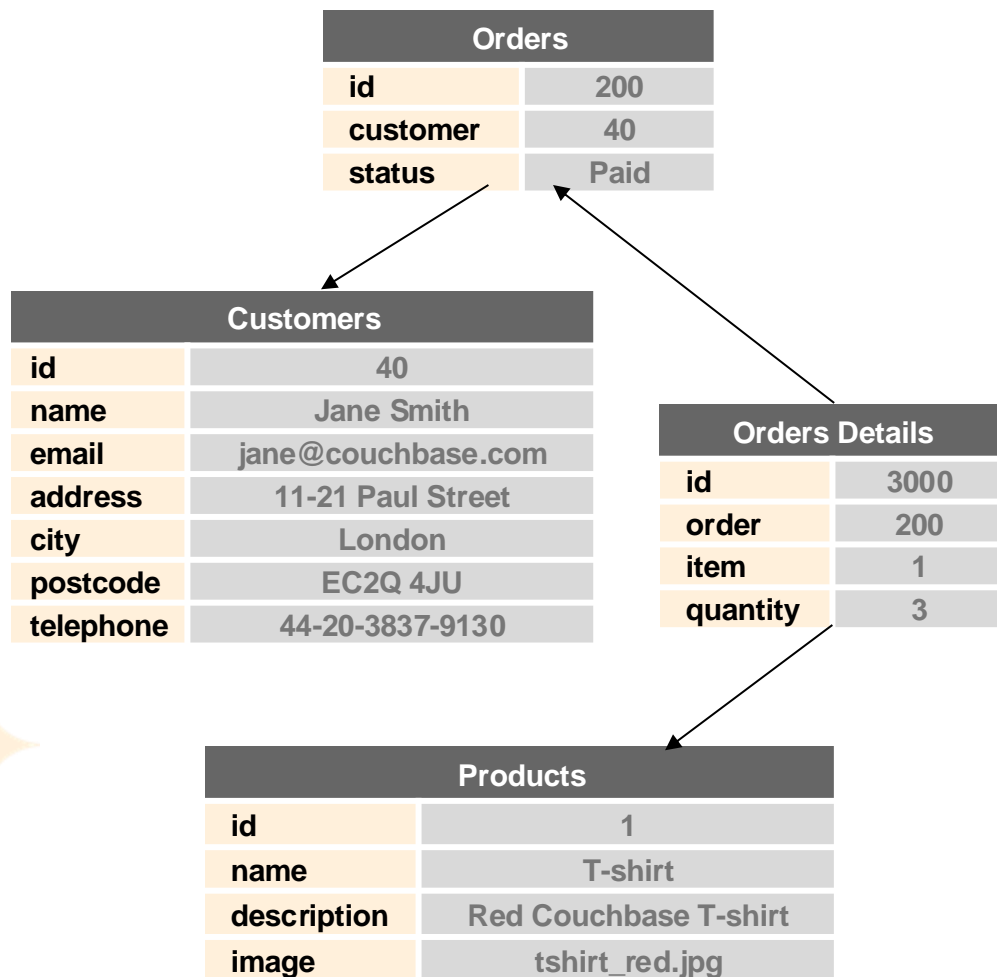
- The Primary Key becomes the Document Key
- Column name-Column value become KEY-VALUE pair

고객의 카드 정보를 JSON으로 표현



- Denormalization simplifies data access and offers the best performance
비정규화는 데이터 사용을 단순화 해주고, 최고의 성능을 제공함
- Value evolution: Simply add additional array element or update a value

테이블(RDBMS)을 컬렉션(NoSQL)에 매핑 | eCommerce 예제



Bucket: ecommerce | Scope: _default

Collection:
'Products'

```
{
  "name": "T-shirt",
  "description": "Red Couchbase T-shirt",
  "image": "tshirt_red.jpg"
}
```

Doc ID: 1

Collection:
'Customers'

```
{
  "name": "Jane Smith",
  "email": "jane@couchbase.com",
  "address": "11-21 Paul Street",
  "city": "London",
  "postCode": "EC2A 4JU",
  "telephone": "44-20-3837-9130"
}
```

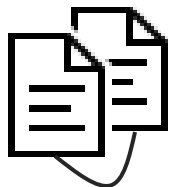
Doc ID: 40

Collection:
'Orders'

```
{
  "customer": {
    "id": 40,
    "name": "Jane smith",
    "email": "jane@couchbase.com"
  },
  "status": "Paid",
  "orderDetails": [
    { "productId": 1, "name": "T-shirt", "quantity": 3 },
  ]
}
```

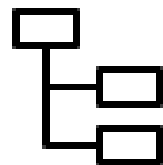
Doc ID: 200

JSON 데이터 모델시 고려사항



EMBED WHEN

- There is an Ownership Relationship
오너쉽 관계가 명확할 때
- Both docs are frequently accessed together
두 문서가 동시에 사용되는 경우가 많을 때
- Reads greatly outnumber writes
읽기가 쓰기에 비해 많을 때
- Data is small
데이터 사이즈가 작을 때



REFER WHEN

- There is not an Ownership Relationship
오너쉽 관계가 명확하지 않을 때
- Both docs are not frequently accessed together
두 문서가 동시에 사용되는 경우가 작을 때
- Document is updated frequently
문서가 자주 변경될 때
- Need to reduce the document size
문서의 크기를 줄일 필요가 있을 때

Try to embed first, refer when it makes sense
먼저 Embed를 시도해 보고, 그 다음 Refer 고려

Relationships에 따른 Embed 와 Refer

- 1-1

Embed Example: Satellite and Manufacturer

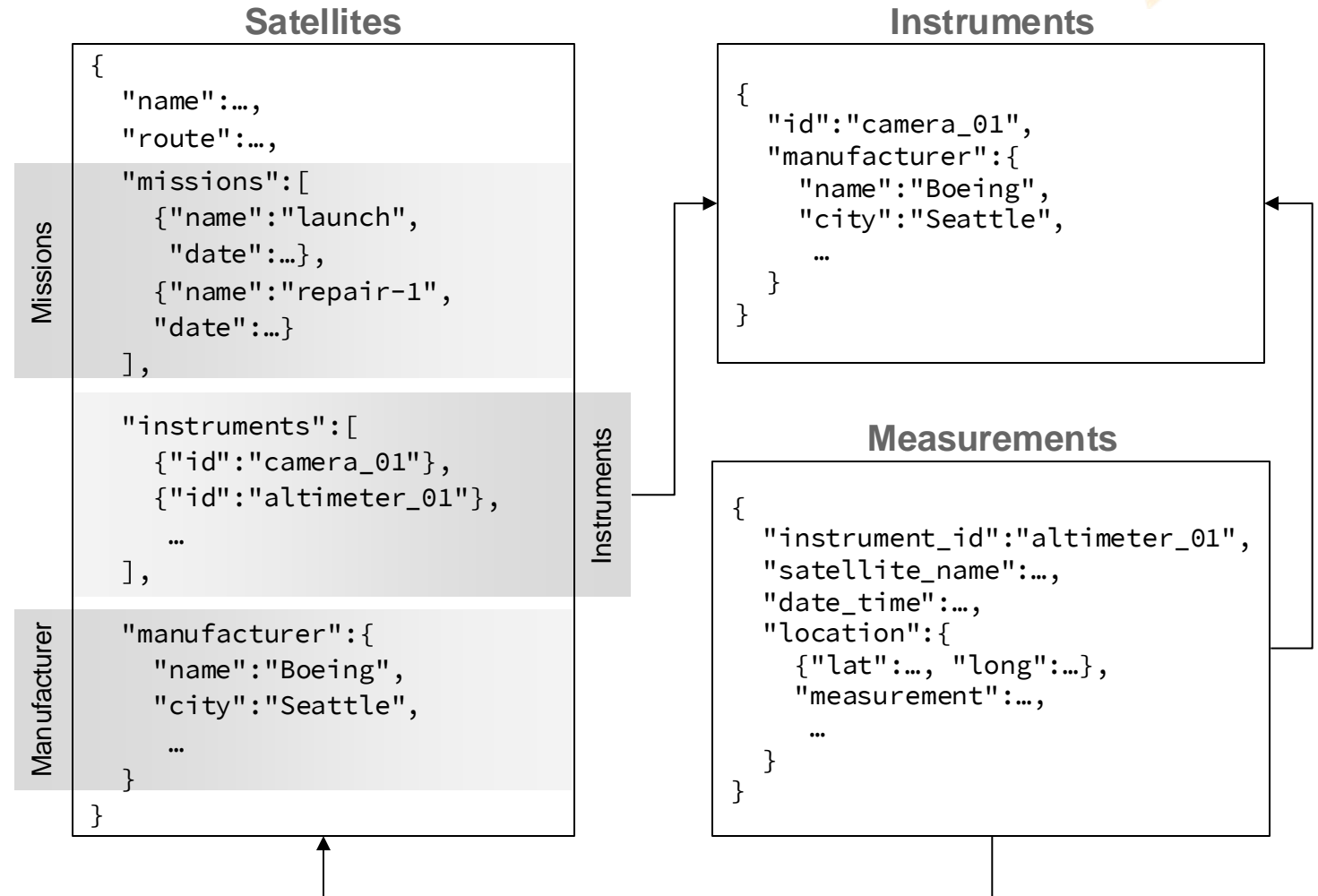
- 1-Many

Embed Example: Satellite and Missions

Reference Example: Measurements, Satellites and Instruments

- Many-Many

Reference Example: Satellite and Instruments





수고하셨습니다.



paul.son@couchbase.com

www.couchbase.com

cloud.couchbase.com



Couchbase