Couchbase Server — Cheat Sheet

SQL++ / KV • verified against docs.couchbase.com/server/current • irurueta.com

SHELLS & CONNECT

```
cbq -e couchbase://HOST -u Administrator -p pw
cbsh                        # Couchbase Shell
couchbase-cli host-list -c HOST -u .. -p ..
```

```
// SDK: one long-lived Cluster
var cluster = Cluster.connect(
  "couchbase://host1,host2",
  "user","pass");
var col = cluster.bucket("travel-sample")
  .scope("inventory").collection("airline");
```

TLS: `couchbases://` • Console :8091 • Query :8093 • FTS :8094

KEYSPACE

- **bucket** → **scope** → **collection** → document
- defaults: `_default` scope & collection; `_system` scope (7.6)
- types: **Couchbase** (persisted), **Ephemeral** (RAM-only), **Memcached** (legacy)

```
CREATE SCOPE `bkt`.inventory;
CREATE COLLECTION `bkt`.inventory.airline;
-- path in SQL++:  `travel-
sample`.inventory.airline
```

KEY-VALUE API

`insert(id,doc)`	fails if exists
`upsert(id,doc)`	create or replace
`replace(id,doc,{cas})`	fails if missing
`get(id,{project,withExpiry})`	
`remove(id,{cas})`	• exists(id)
`getAndTouch(id,ttl)`	• touch(id,ttl)

errors: DocumentNotFound / DocumentExists / CasMismatch

COUNTERS, BINARY, BULK

```
binary().increment(id,{delta:1,initial:0})
binary().decrement(id,{delta:1})
binary().append(id, bytes)   prepend(id, bytes)
```

- KV Range Scan (7.6): `scan(RangeScan(from,to))` / `PrefixScan` / `SamplingScan`
- bulk: reactive/async API; `ReactiveCollection`, `AsyncCollection`

SUB-DOCUMENT

```
col.lookupIn(id,[ get("addr.city"),
                  exists("phones[0]") ])
col.mutateIn(id,[
  upsert("addr.zip","94040"),
  arrayAppend("phones",p),
  arrayAddUnique("tags","vip"),
  increment("visits",1),
  remove("tmp") ], {storeSemantics:UPSERT})
```

flags: `createPath`, `xattr` (system/user attrs)

CAS & LOCKING

```
// optimistic: retry loop
for (;;) {
  var r = col.get(id);
  var d = mutate(r.contentAs(...));
  try { col.replace(id,d,{cas:r.cas()}); break;
}
  catch (CasMismatchException e) { /* retry */
}
}
```

```
// pessimistic
var r = col.getAndLock(id,
Duration.ofSeconds(5));
col.replace(id, d, {cas: r.cas()});   // or
col.unlock(id, r.cas())
```

CAS doubles as an HTTP **ETag**

DURABLE WRITES

`majority`	in RAM on majority of replicas
`majorityAndPersistToActive`	+ on disk on active
`persistToMajority`	on disk on majority

```
col.upsert(id,doc,{durability:
PERSIST_TO_MAJORITY})
```

supersedes observe-based `PersistTo`/`ReplicateTo`

SQL++ — SELECT

```
SELECT a.name, COUNT(*) AS n
FROM `travel-sample`.inventory.airline a
WHERE a.country = "France"
GROUP BY a.name HAVING COUNT(*) > 1
ORDER BY n DESC LIMIT 10 OFFSET 0;
```

- `SELECT RAW expr` / `SELECT VALUE` → unwrapped array
- `UNNEST a.schedule s` flatten array; `NEST` to group
- paths: `a.geo.lat`, `a.tags[0]`, `a.tags[*]`

SQL++ — JOINS & WITH

```
-- lookup join (key equi-join)
SELECT r.*, a.name FROM route r
  JOIN airline a ON KEYS r.airlineid;
-- ANSI join
SELECT * FROM route r JOIN airline a
  ON r.airlineid = META(a).id;
-- CTE / recursive
WITH RECURSIVE parts AS (
  SELECT * FROM bom WHERE id = "root"
  UNION SELECT c.* FROM bom c JOIN parts p ON
c.parent = META(p).id)
SELECT * FROM parts;
```

SQL++ — DML

```
INSERT INTO coll (KEY,VALUE) VALUES ("k",
{"a":1});
UPSERT INTO coll (KEY,VALUE) VALUES ("k",
{"a":2});
UPDATE coll USE KEYS "k" SET a = a+1, tags =
ARRAY_APPEND(tags,"x")
  UNSET tmp RETURNING META().id;
DELETE FROM coll WHERE expired = true;
MERGE INTO tgt t USING src s ON t.id = s.id
  WHEN MATCHED THEN UPDATE SET t.v = s.v
  WHEN NOT MATCHED THEN INSERT (KEY s.id, VALUE
s);
```

INDEXES (GSI)

```
CREATE PRIMARY INDEX ON coll;          -- dev
only
CREATE INDEX ix_name ON coll(country, name);
CREATE INDEX ix_part ON coll(name) WHERE
type="airline";
CREATE INDEX ix_arr ON coll(DISTINCT ARRAY
s.day
  FOR s IN schedule END);
CREATE INDEX ix_def ON coll(x) WITH
{"defer_build":true};
BUILD INDEX ON coll(ix_def);
```

- `EXPLAIN` / `ADVISE` <stmt> • `UPDATE STATISTICS` for CBO
- covering: all referenced keys in the index → no fetch
- `... PARTITION BY HASH(k)` • `WITH {"num_replica":1}`

FULL-TEXT SEARCH

```
SELECT META().id, SEARCH_SCORE() AS score
FROM hotel AS h
WHERE SEARCH(h, {"query":
{"match":"spa","field":"description"},
                "size":10});
```

query types: match, match_phrase, prefix, wildcard, regexp, fuzzy (`fuzziness`), term, numeric/date range, geo (`distance`, bounding box), boolean, conjuncts/disjuncts, query_string, **knn** (vector / hybrid, 7.6).

EVENTING FUNCTION

```
function OnUpdate(doc, meta) {
  if (doc.type !== "order") return;
  tgt[meta.id] = { total: doc.qty * doc.price
};  // bucket binding "tgt"
  createTimer(cb, DATE, meta.id, ctx);
// timers
}
function OnDelete(meta, options) { delete
tgt[meta.id]; }
```

feed boundary: **From now** / **From beginning** • cURL bindings for HTTP

TRANSACTIONS (ACID)

```
// SDK lambda — auto retry/rollback
cluster.transactions().run(ctx -> {
  var a = ctx.get(col, "acc:1");
  ctx.replace(a, debit(a.contentAs(...)));
  ctx.insert(col, "log:x", entry);
});
```

```
BEGIN WORK;  SET TRANSACTION ISOLATION LEVEL
READ COMMITTED;
UPDATE acc USE KEYS "1" SET bal = bal - 10;
SAVEPOINT s1;  COMMIT WORK;   -- or ROLLBACK
[TO SAVEPOINT s1]
```

Read Committed • staged in XATTRs + ATR docs • doc < 10 MB • NTP-synced

CLUSTER OPS

```
couchbase-cli server-add   -c H --server-add H2
...
couchbase-cli rebalance    -c H -u .. -p ..
couchbase-cli failover     -c H --server-
failover H2 --hard
couchbase-cli setting-autofailover --enable-
auto-failover 1 --auto-failover-timeout 120
```

```
# XDCR
couchbase-cli xdcr-setup    --create --xdcr-
cluster-name dr --xdcr-hostname H2 ...
couchbase-cli xdcr-replicate --create --xdcr-
from-bucket b --xdcr-to-bucket b --xdcr-
cluster-name dr
```

conflict resolution: **seqno** (revID) or **LWW** timestamp (set at bucket creation)

BACKUP, IMPORT & MONITOR

```
cbbackupmgr config  --archive /bk --repo r
cbbackupmgr backup  --archive /bk --repo r --
cluster couchbase://H -u .. -p ..
cbbackupmgr restore --archive /bk --repo r ...
cbimport json --format lines -d file:///d.json
-c H -b bkt -u .. -p ..
cbexport json -c H -b bkt -o out.json -u .. -p
..
```

stats: `cbstats HOST all` • `/pools/default` • Prometheus `/metrics` • legacy `cbbackup`/`cbtransfer` removed

MODEL: EMBED VS REFERENCE

`Embed`	read together • bounded array • changes with parent • 1:1 / 1:few
`Reference`	queried alone • unbounded / large • written independently • 1:many / many:many

keys: predictable `type::id`, counters (`increment`), or UUID • version with a `schema` field • a collection per entity type