

MySQL y Apache CouchDB: análisis comparativo de sistemas de gestión de bases de datos relacionales y NoSQL

MySQL and Apache CouchDB: Comparative analysis of relational and NoSQL database management systems

ANDRÉS CAMILO VIVAS BAQUERO¹ - LAURA NATALIA PERILLA QUINTERO²

1. Estudiante de ingeniería de Sistemas
Universidad Escuela Colombiana de Ingeniería Julio Garavito.
ORCID: <https://orcid.org/0009-0003-4664-6325>
andres.vivas-b@mail.escuelaing.edu.co
2. Estudiante de ingeniería de Sistemas
Universidad Escuela Colombiana de Ingeniería Julio Garavito.
ORCID: <https://orcid.org/0009-0004-3767-4938>
laura.perilla-q@mail.escuelaing.edu.co

Recibido: 14/03/2026 Aceptado: 5/04/2026

Disponible en http://www.escuelaing.edu.co/es/publicaciones_revista
<http://revistas.escuelaing.edu.co/index.php/revi>

Resumen

En el presente artículo se hace un análisis comparativo entre MySQL 8.x, sistema de gestión de bases de datos relacional de mayor adopción mundial, y Apache CouchDB 3.x, motor NoSQL, orientado a documentos y basado en JSON. El objetivo es proporcionar una referencia técnica detallada que cubra los principales aspectos de ambos sistemas: arquitectura interna, modelo de datos, lenguaje de consultas, mecanismos de concurrencia (MVCC), seguridad, respaldo, replicación, escalabilidad, mantenimiento y casos de uso ideales. La novedad del trabajo reside en la integración de un análisis técnico comparativo exhaustivo de estos dos paradigmas contrastantes dentro del contexto académico colombiano, complementado con ejemplos prácticos de operaciones CRUD y una tabla comparativa estructurada.

La metodología empleada es de naturaleza documental-comparativa: se revisaron fuentes primarias (documentación oficial de MySQL y CouchDB), bibliografía académica especializada y estudios de rendimiento publicados, articulando los hallazgos en un marco comparativo sistemático.

Los resultados indican que MySQL sobresale en entornos transaccionales con datos altamente estructurados, garantizando consistencia ACID multifila, mientras que CouchDB ofrece ventajas significativas en escenarios de alta distribución, sincronización *offline* y esquemas dinámicos, priorizando disponibilidad y consistencia eventual.

La elección entre ambos motores depende del tipo de problema, el volumen de datos, los requisitos de consistencia y la arquitectura del

sistema, y no constituye una relación de superioridad sino de idoneidad contextual.

Palabras claves: MySQL, CouchDB, bases de datos relacionales, NoSQL.

Abstract

This article presents a comparative analysis between MySQL 8.x, the world's most widely adopted relational database management system, and Apache CouchDB 3.x, a JSON-based NoSQL document-oriented database engine. The objective is to provide a detailed technical reference covering the main aspects of both systems: internal architecture, data model, query language, concurrency mechanisms (MVCC), security, backup, replication, scalability, maintenance, and ideal use cases. The novelty of this work lies in the integration of an exhaustive comparative technical analysis of these two contrasting paradigms within the Colombian academic context, complemented by practical CRUD operation examples and a structured comparison table.

The methodology is documentary-comparative in nature: primary sources (official MySQL and CouchDB documentation), specialized academic literature, and published performance studies were reviewed, articulating findings in a systematic comparative framework.

Results indicate that MySQL excels in transactional environments with highly structured data, guaranteeing multi-row ACID consistency, while CouchDB offers significant advantages in high-distribution

scenarios, offline synchronization and dynamic schemas, prioritizing availability and eventual consistency.

The choice between both engines depends on problem type, data volume, consistency requirements and system architecture, and does not constitute a relationship of superiority but of contextual suitability.

Keywords: MySQL, CouchDB, relational databases, NoSQL.

INTRODUCCIÓN

El crecimiento exponencial del volumen de datos en la era digital ha llevado a la industria tecnológica a desarrollar soluciones de almacenamiento cada vez más diversas. Desde la consolidación del modelo relacional en la década de los setenta hasta la explosión de las bases de datos NoSQL a principios del siglo XXI, los sistemas de gestión de bases de datos (SGBD) han evolucionado para responder a necesidades específicas de rendimiento, flexibilidad y escalabilidad (Date, 2004).

En el contexto del curso de Sistemas de Gestión de Bases de Datos de la Universidad Escuela Colombiana de Ingeniería Julio Garavito, se seleccionaron dos motores con filosofías de diseño contrastantes: MySQL, el SGBD relacional de código abierto más popular del mundo, y Apache CouchDB, un motor NoSQL orientado a documentos que abraza los principios de la web y la computación distribuida. Esta selección permite ilustrar la dicotomía SQL/NoSQL, los fundamentos de ambos paradigmas y las implicaciones prácticas de elegir uno u otro en proyectos reales.

El objetivo de este artículo es proporcionar una referencia técnica detallada que cubra los temas abordados a lo largo del semestre: instalación, modelado de datos, operaciones CRUD, lenguaje de consultas, concurrencia, seguridad, respaldo, replicación, estructura de alta disponibilidad, mantenimiento y optimización. Al final, se muestra una tabla comparativa y se hacen recomendaciones de uso según el tipo de problema.

El artículo se estructura de la siguiente manera: en la sección 2 se presentan la revisión de la bibliografía y el estado del arte; en la sección 3 se describe la metodología empleada; en la sección 4 se desarrolla el análisis técnico comparativo detallado de ambos motores, incluyendo arquitectura, modelo de datos, concurrencia, seguridad, respaldo, replicación, optimización y casos de uso, y en la sección 5 se sintetizan las conclusiones del trabajo.

REVISIÓN DE LA BIBLIOGRAFÍA (ESTADO DEL ARTE)

La distinción entre bases de datos relacionales y NoSQL se ha estudiado ampliamente desde la proliferación de los sistemas distribuidos a gran escala. Pokorny (2013) señala que las bases de datos NoSQL no surgieron como remplazo de las relacionales, sino como respuesta a las limitaciones de escalabilidad horizontal y flexibilidad de esquema que el modelo relacional impone en entornos web de alta demanda. En la misma línea, Fowler y Sadalage (2013) introducen el concepto de persistencia polígota, según el cual una arquitectura moderna puede combinar múltiples motores de bases de datos, según las necesidades específicas de cada componente del sistema.

El teorema CAP, formulado por Brewer (2000), constituye el marco teórico fundamental para comprender las diferencias de diseño entre MySQL y CouchDB: ningún sistema distribuido puede garantizar simultáneamente consistencia, disponibilidad y tolerancia a particiones de red. MySQL, en configuración *standalone* o con Group Replication, da prioridad a consistencia y disponibilidad (CA), mientras que CouchDB, diseñado para entornos distribuidos, prioriza disponibilidad y tolerancia a particiones (AP), además de que garantiza consistencia eventual.

Kaur y Rani (2013) analizaron las capacidades de modelado y consulta de los sistemas NoSQL frente a los relacionales, y concluyeron que la elección del paradigma se debe basar en el análisis de los patrones de acceso, la naturaleza de los datos y los requisitos de consistencia. Por su parte, Chickerur, Goudar y Kinnerkar (2015) compararon el rendimiento de bases de datos relacionales con MongoDB en aplicaciones de *big data*, y evidenciaron ventajas de los sistemas documentales en inserción masiva de registros, mientras que los sistemas relacionales superan en consultas complejas con múltiples *joins*.

Patil, Sonsale y Kamble (2020), en un estudio publicado en *Applied Sciences*, hicieron un análisis de rendimiento específico entre CouchDB y MySQL, en el que se confirma que las diferencias se estrechan para conjuntos de datos pequeños (menos de 100.000 registros) y se amplían significativamente en cargas de trabajo masivas, con ventajas de CouchDB en inserción distribuida y de MySQL en eliminación y consultas complejas. IBM (2024) refuerza esta perspectiva al describir CouchDB como una solución madura para aplicaciones

offline-first y sincronización multirregión, en tanto que MySQL mantiene su dominio en sistemas transaccionales empresariales, de acuerdo con DB-Engines (2025).

La documentación oficial de MySQL (Oracle Corporation, 2023) y la de Apache CouchDB (Apache Software Foundation, 2023) constituyen las fuentes primarias más actualizadas sobre la arquitectura interna, el modelo de datos y los mecanismos de concurrencia de cada motor; además, sirven como referencia técnica base para el análisis hecho en este artículo.

METODOLOGÍA

En el presente artículo se adopta una metodología documental-comparativa de carácter cualitativo y técnico. El análisis se desarrolló en tres fases articuladas.

En la primera fase se hizo una revisión sistemática de fuentes primarias: la documentación oficial de MySQL 8.x (Oracle Corporation, 2023) y de Apache CouchDB 3.x (Apache Software Foundation, 2023), complementada con bibliografía académica especializada indexada en bases de datos como IEEE Xplore y Google Scholar, al igual que con informes técnicos de organizaciones de referencia (IBM, DB-Engines). En los criterios de selección de fuentes se dio prioridad a publicaciones de los últimos doce años, pertinencia técnica directa con los motores analizados y disponibilidad de acceso abierto.

En la segunda fase se construyó un marco comparativo estructurado en doce dimensiones técnicas: historia y evolución, arquitectura interna, modelo de datos y lenguaje de consultas, instalación y configuración, gestión de objetos y almacenamiento, control de concurrencia (MVCC), seguridad, respaldo y recuperación, replicación y alta disponibilidad, optimización y mantenimiento, migración e importación de datos, y casos de uso. Para cada dimensión se documentaron ejemplos prácticos de operaciones CRUD, así como fragmentos de configuración extraídos de la documentación oficial, con el fin de ilustrar las diferencias de paradigma en forma concreta.

En la tercera fase se sintetizaron los hallazgos en una tabla comparativa estructurada (tabla 1) y se contrastaron con los estudios de rendimiento disponibles en la bibliografía, al tiempo que se elaboraron criterios de selección aplicables a distintos tipos de proyectos. Este enfoque metodológico es coherente con estudios comparativos previos del área, como los de Kaur y

Rani (2013) y Patil et al. (2020), que emplearon marcos dimensionales similares para comparar sistemas de bases de datos heterogéneos.

DISCUSIÓN Y RESULTADOS

En esta sección se desarrolla el análisis comparativo de MySQL 8.x y Apache CouchDB 3.x en las doce dimensiones técnicas definidas en la metodología. Para cada dimensión se presentan las características de cada motor y se discuten las implicaciones prácticas de sus diferencias de diseño.

Historia y evolución

MySQL nació en Suecia en 1995, de la mano de Michael “Monty” Widenius, David Axmark y Allan Larsson, quienes fundaron la empresa MySQL AB. El nombre MySQL combina el prefijo “My” (en honor de la hija de Widenius) con la sigla SQL. El proyecto surgió de la necesidad de un sistema de bases de datos más rápido y flexible que mSQL (MySQL AB, 1995).

En el año 2000, MySQL se publicó íntegramente bajo la licencia GPL v2, lo que impulsó su adopción masiva como componente del *stack* LAMP (Linux, Apache, MySQL, PHP/Perl/Python). En 2008, Sun Microsystems adquirió MySQL AB; dos años después, Oracle Corporation completó la compra, lo que motivó a un grupo de desarrolladores originales a crear en 2009 el *fork* MariaDB como alternativa comunitaria. Desde la versión 5.0 (2005) hasta la actual 8.0/8.4 LTS, MySQL ha incorporado soporte UTF-8, particionamiento, procedimientos almacenados, soporte JSON, replicación GTID e InnoDB Cluster. La versión 9.0 de 2024 introdujo programas almacenados en JavaScript y las capacidades de HeatWave GenAI.

Damien Katz, exdesarrollador de Lotus Notes en IBM, creó Apache CouchDB en abril de 2005 con la visión de construir una base de datos orientada a documentos que adoptara los principios de la web. El nombre CouchDB es un acrónimo de “Cluster Of Unreliable Commodity Hardware”, filosofía que refleja su diseño para tolerar fallos de nodos individuales. Desarrollado inicialmente en C++, el proyecto migró a Erlang en 2006 e ingresó al incubador de la Apache Software Foundation en 2008. La versión 2.0.0 (2016) incorporó *clustering* nativo, heredado del proyecto BigCouch. La versión 3.x (vigente a 2025) consolidó

el Mango Query Server y ofreció mayores garantías de seguridad por defecto.

Arquitectura interna

MySQL emplea una arquitectura de tres capas bien definidas (Schwartz et al., 2012):

- Capa de conexión y autenticación, que gestiona conexiones de clientes mediante un *pool* de hilos.
- Capa de servicios del servidor, que contiene el *parser*, el optimizador de consultas y los objetos de servidor (procedimientos, disparadores, vistas).
- Capa de motores de almacenamiento, que expone una API abstracta para motores intercambiables.

El motor predeterminado es InnoDB, que utiliza un índice agrupado en páginas de 16 KB almacenadas en tablespaces, con un *buffer pool* en memoria y un *redo log* para garantizar la durabilidad (WAL). MySQL escucha en el puerto 3306 (TCP); el proceso principal se denomina *mysqld*.

CouchDB está construido sobre Erlang/OTP y sigue el principio “everything is HTTP”: toda interacción se hace mediante una API RESTful (Lehnardt, 2014). Internamente, cada base de datos se almacena en un único archivo B-tree de *solo-append*, lo que simplifica la recuperación ante fallos y habilita MVCC sin bloqueos de lectura. A partir de la versión 2.0, CouchDB implementa *clustering* nativo mediante un anillo de *hashing* consistente, con *sharding* y réplicas configurables por base de datos. CouchDB escucha en los puertos 5984 (HTTP) y 6984 (HTTPS); el proceso principal es *beam.smp*.

Modelo de datos y lenguaje de consultas

MySQL organiza los datos en tablas con esquema rígido predefinido (DDL), siguiendo el modelo relacional de E. F. Codd (1970). Las relaciones entre tablas se expresan mediante claves foráneas que habilitan operaciones JOIN eficientes. El lenguaje de consultas es SQL (SQL:2023), con extensiones propietarias para operaciones como REPLACE INTO, GROUP_CONCAT y la familia de funciones JSON_*. A partir de la versión 8.0, MySQL soporta expresiones de tabla comunes (CTE) y funciones de ventana. A continuación, se ilustra un ciclo CRUD completo en MySQL:

```
-- CREATE TABLE CREATE TABLE estudiantes (id INT AUTO_INCREMENT PRIMARY KEY,
```

```
nombre VARCHAR(100) NOT NULL, materia VARCHAR(80), salon VARCHAR(20), created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP); -- INSERT INSERT INTO estudiantes (nombre, materia, salon) VALUES ('Ana García', 'Bases de Datos', 's-301'); -- SELECT con JOIN SELECT e.nombre, m.nombre AS materia FROM estudiantes e JOIN materias m ON e.materia_id = m.id WHERE m.nombre = 'Bases de Datos'; -- UPDATE UPDATE estudiantes SET salon = 's-302' WHERE id = 1; -- DELETE DELETE FROM estudiantes WHERE id = 1;
```

CouchDB almacena los datos como documentos JSON independientes, sin esquema predefinido. Cada documento tiene un identificador único (*_id*) y un número de revisión (*_rev*) gestionado automáticamente. El lenguaje de consultas principal utiliza el paradigma MapReduce con funciones JavaScript; la versión 2.0 introdujo el Mango Query Server, que permite hacer consultas mediante documentos JSON similares a MongoDB. Las operaciones se realizan íntegramente mediante peticiones HTTP, como se ilustra a continuación:

```
# CREATE DATABASE PUT http://localhost:5984/estudiantes_db # INSERT PUT http://localhost:5984/estudiantes_db/est001 Body: {"nombre": "Ana García", "materia": "Bases de Datos", "salon": "S-301"} # READ GET http://localhost:5984/estudiantes_db/est001 # UPDATE (requiere _rev) PUT http://localhost:5984/estudiantes_db/est001 Body: {"_rev": "1-abc123", "salon": "S-302"} # DELETE DELETE http://localhost:5984/estudiantes_db/est001?rev=2-def456 # MANGO QUERY POST http://localhost:5984/estudiantes_db/_find Body: {"selector": {"materia": "Bases de Datos"}}
```

Instalación y configuración inicial

MySQL está disponible en los repositorios oficiales de Ubuntu y a través del repositorio APT de Oracle. La instalación en Ubuntu 22.04 LTS se realiza con `sudo apt install -y mysql-server`, seguida del asistente de seguridad `mysql_secure_installation`. El servicio se gestiona con `systemctl`, el puerto por defecto es 3306 y la configuración principal reside en `/etc/mysql/mysql.conf.d/mysqld.cnf`. Para entornos en nube, MySQL está disponible como servicio gestionado en AWS (RDS), Azure (Azure Database for MySQL) y Oracle Cloud (HeatWave). También se puede desplegar mediante Docker: `docker run --name mysql-db -e`

`MYSQL_ROOT_PASSWORD=secreto -p 3306:3306 -d mysql:8.0.`

CouchDB requiere la adición del repositorio oficial de Apache. La instalación en Ubuntu 22.04 LTS incluye la importación de la clave GPG y la adición del repositorio antes de ejecutar `sudo apt install -y couchdb`; durante el proceso se selecciona el modo (*standalone* o *cluster*) y se establece la contraseña del administrador. El servicio escucha en el puerto 5984 y la configuración reside en `/opt/couchdb/etc/local.ini`. CouchDB ofrece la interfaz gráfica Fauxton, accesible desde `http://localhost:5984/_utils`, que permite crear bases de datos, insertar documentos y ejecutar consultas Mango sin herramientas adicionales.

Arquitectura de objetos y gestión de almacenamiento

La jerarquía de objetos en MySQL sigue este orden: instancia → esquema/base de datos → tablas → columnas/índices. Los objetos principales son bases de datos, tablas, índices B-Tree/Hash/Full-text, vistas, procedimientos y funciones almacenadas, disparadores, y usuarios y roles. El almacenamiento de InnoDB utiliza páginas de 16 KB en tablespaces; los archivos .ibd residen en `/var/lib/mysql`. El diccionario de datos en MySQL 8.0 es transaccional y se almacena en InnoDB, reemplazando los archivos .frm de versiones anteriores.

CouchDB tiene una jerarquía más simple: instancia → bases de datos → documentos → campos. Los componentes principales son bases de datos (archivos .couch de solo-append), documentos JSON con `_id` y `_rev`, documentos de diseño (`_design`) con vistas MapReduce y funciones de validación, índices Mango para el lenguaje de consultas JSON, y usuarios gestionados en la base de datos especial `_users`. La administración se realiza mediante las bases de datos del sistema: `_global_changes`, `_replicator` y `_users`.

Control de concurrencia (MVCC)

InnoDB implementa MVCC combinado con bloqueos de fila. Internamente, cada fila almacena los campos ocultos `DB_TRX_ID` (identificador de la última transacción modificadora) y `DB_ROLL_PTR` (puntero al Undo Log). El nivel de aislamiento por defecto es REPEATABLE READ, que utiliza *next-key locks* para prevenir lecturas fantasma (Oracle Corporation, 2023). Los tipos de candados incluyen bloqueos compartidos (S), exclusivos (X), de intención (IS, IX) y de *gap*. El

comando `SHOW ENGINE INNODB STATUS` permite inspeccionar bloqueos activos y detectar *deadlocks*.

CouchDB implementa MVCC mediante *shadow paging* en el archivo B-tree de *solo-append*: las escrituras agregan una nueva versión del documento al final del archivo sin sobrescribir datos existentes, eliminando la necesidad de bloqueos en lecturas (Apache Software Foundation, 2023). No existe el concepto de transacciones multidocumento; cada escritura es atómica a nivel de documento único con semántica ACID. En entornos distribuidos, CouchDB garantiza consistencia eventual: el campo `_rev` actúa como control de versiones optimista y los conflictos de escritura simultánea se gestionan mediante el código de error 409 Conflict.

Seguridad

MySQL implementa un modelo de seguridad basado en usuarios, *hosts* y privilegios. Los componentes claves son autenticación mediante los *plugins* `caching_sha2_password` (recomendado en 8.0), LDAP y Kerberos; autorización granular en el servidor, base de datos, tabla, columna y rutina mediante `GRANT/REVOKE`; cifrado TLS/SSL para conexiones en tránsito e InnoDB Transparent Data Encryption (TDE) para datos en reposo desde la versión 5.7, y auditoría mediante el *plugin* Enterprise o Percona Audit Log Plugin. El *hardening* básico incluye ejecutar `mysql_secure_installation`, configurar `require_secure_transport=ON` y limitar el *bind-address* a la interfaz necesaria.

En CouchDB 3.x, la seguridad se gestiona a través de la API RESTful con el modo “Admin Party” deshabilitado por defecto. Soporta autenticación básica HTTP, *cookies* de sesión, *tokens* JWT y OAuth 2.0. La autorización se define por base de datos mediante el documento `_security` con *arrays* `admins` y `members`. El cifrado TLS/SSL es nativo para conexiones; el cifrado en reposo se debe implementar en el sistema de ficheros (LUKS). Las funciones `validate_doc_update` en documentos de diseño permiten rechazar escrituras que no cumplan reglas de negocio. El *hardening* incluye configurar `require_valid_user=true` y habilitar TLS en `local.ini`.

Respaldo y recuperación

MySQL ofrece tres mecanismos principales de respaldo. El respaldo lógico mediante `mysqldump` genera archivos SQL portables con DDL + DML, soportando la

opción `--single-transaction` para consistencia *online* con InnoDB. El respaldo físico mediante MySQL Enterprise Backup o Percona XtraBackup (gratuito) copia los archivos de InnoDB en caliente con soporte para respaldos incrementales. Los Binary Logs registran todas las modificaciones y habilitan la recuperación *point-in-time* (PITR) aplicando `mysqlbinlog` sobre los *logs* archivados tras restaurar un *backup full*.

El modelo de *solo-append* de CouchDB simplifica las estrategias de respaldo. La herramienta oficial *couchbackup* (`npm install -g @cloudant/couchbackup`) exporta documentos como JSON a un archivo NDJSON con soporte *full* e incremental. La replicación incremental nativa se puede utilizar como mecanismo de *backup* continuo hacia una instancia destino. También es posible realizar *snapshots* del sistema de ficheros del archivo *.couch* para *backups offline*. La operación de compactación (`POST /{db}/_compact`) reduce el espacio en disco, eliminando revisiones obsoletas y se puede configurar para ejecutarse automáticamente.

Replicación y alta disponibilidad

MySQL implementa replicación asíncrona y semisíncrona nativa desde la versión 3.23. Los modelos disponibles incluyen la replicación maestro-réplica clásica, los GTID (Global Transaction Identifiers, disponibles desde la versión 5.6) que simplifican la configuración y el *failover* automático, e InnoDB Cluster basado en MySQL Group Replication (activo-activo con cuórum), MySQL Router y MySQL Shell, que proporciona *failover* automático y tolerancia a fallos de nodos. También soporta topologías en cascada y *multi-source* para consolidación de datos.

La replicación es una de las características más diferenciadas de CouchDB. Su modelo *multi-master* bidireccional permite que múltiples nodos reciban escrituras simultáneamente y se sincronicen después mediante replicación incremental basada en el *_changes* feed. Esto habilita patrones *offline-first*, donde un dispositivo trabaja sin conexión y sincroniza al recuperar la red. Cuando dos réplicas modifican el mismo documento en forma independiente, CouchDB marca el documento como conflictuado (*_conflicts*) y delega la resolución a la aplicación. En modo clúster, los datos se distribuyen automáticamente entre nodos mediante *hashing*, consistente con soporte para adición y eliminación de nodos en caliente.

Optimización y mantenimiento

MySQL incorpora un optimizador basado en costos (CBO) que evalúa múltiples planes de ejecución. Las estadísticas se almacenan en *information_schema* e InnoDB Statistics y se visualizan con `EXPLAIN` o `EXPLAIN ANALYZE` (desde MySQL 8.0). Las tareas de mantenimiento regulares incluyen `ANALYZE TABLE` (actualizar estadísticas), `OPTIMIZE TABLE` (reorganizar el *tablespace*), `CHECK TABLE` (verificar integridad) y la configuración del parámetro `innodb_buffer_pool_size` (idealmente, 70-80 % de la RAM disponible).

En CouchDB, el rendimiento depende principalmente de los índices de vistas y los índices Mango. Las vistas MapReduce se construyen en forma incremental; la primera consulta puede ser lenta si hay muchos documentos sin indexar. La operación de compactación elimina revisiones obsoletas del archivo de base de datos y de los archivos de vistas. La monitorización se hace por medio del *endpoint* `GET /_stats` (estadísticas del servidor) y `GET /{db}/_design/{ddoc}/_info` (estado de una vista específica).

Migración, exportación e importación de datos

MySQL ofrece `SELECT ... INTO OUTFILE / LOAD DATA INFILE` para transferencia de datos en CSV, `mysqldump` para exportación completa en SQL, y la interfaz gráfica MySQL Workbench, con asistentes para múltiples formatos. Para migraciones complejas, herramientas ETL como Pentaho Data Integration, Apache Spark y AWS Database Migration Service se conectan mediante JDBC.

La naturaleza HTTP de CouchDB facilita la integración con *pipelines* de datos. La herramienta *couchbackup/couchrestore* exporta e importa documentos en formato NDJSON. El *endpoint* `_all_docs` con `include_docs=true` devuelve todos los documentos y es consumible directamente por *scripts* Python o Node.js. Mediante el *_changes* feed es posible construir *pipelines* de *streaming* que consuman cambios en tiempo real e integrarlos con Apache Kafka o Apache Spark.

Tabla comparativa

A renglón seguido, se resumen los aspectos técnicos más relevantes de ambos motores analizados en las secciones anteriores (tabla 1).

Tabla 1
Comparativa técnica entre MySQL 8.x y Apache CouchDB 3.x

Criterio	MySQL 8.x	Apache CouchDB 3.x
Tipo de BD	Relacional (RDBMS)	Documental NoSQL
Modelo de datos	Tablas, filas, columnas, esquema rígido	Documentos JSON, esquema flexible, sin <i>joins</i>
Lenguaje de consultas	SQL estándar (SQL:2023) + extensiones Oracle	MapReduce (JS) + Mango Query (JSON)
Lenguaje de programación	C, C++	Erlang/OTP
Puerto por defecto	3306 (TCP)	5984 (HTTP) / 6984 (HTTPS)
Proceso principal	Mysqld	beam.smp (Erlang)
ACID por transacción	Sí (InnoDB, a nivel multifila)	Sí, como documento único
Consistencia distribuida	Fuerte (por defecto)	Eventual (configurable)
Concurrencia	MVCC + row-level locking + gap locks	MVCC puro (<i>append-only</i> , sin <i>read locks</i>)
Replicación	Maestro-réplica, Group Replication, GTID	<i>Multi-master</i> bidireccional, incremental
Escalabilidad horizontal	InnoDB Cluster, NDB Cluster (limitado)	Nativa, <i>sharding</i> con <i>hashing</i> consistente
Índices	B-Tree, Hash, Full-text, Espacial	B-Tree (vistas), índices Mango
Particionamiento	RANGE, LIST, HASH, KEY (nivel de tabla)	<i>Sharding</i> automático en clúster
Backup	mysqldump, XtraBackup, Binary Log (PITR)	<i>couchbackup</i> , replicación como <i>backup</i>
Interfaz administrativa	MySQL Workbench, consola mysql	Fauxton (web), API REST
Soporte Docker	Sí (imagen oficial mysql:8.0)	Sí (imagen oficial apache/couchdb)
Soporte nubes	AWS RDS, Azure DB, GCP SQL, OCI HeatWave	IBM Cloudant, AWS (manual), Azure (manual)
Licencia	GPL v2 (Community) / Propietaria (<i>Enterprise</i>)	Apache License 2.0
Uso de disco	Moderado; <i>tablespaces</i> estructurados en páginas	Mayor (<i>append-only</i> + versiones); compactación periódica
Casos de uso ideales	ERP, e-commerce, banca, apps web transaccionales	Offline-first, móvil, IoT, CMS, sync multi-región

Fuente: Elaboración propia, basada en documentación oficial y bibliografía académica.

Criterios de selección y casos de uso

MySQL es la elección más adecuada cuando el proyecto requiere:

- Transacciones ACID que involucren múltiples tablas y entidades relacionadas (sistemas bancarios, ERP, plataformas de *e-commerce* con inventario y pedidos).
- Datos con estructura bien definida y estable, con relaciones complejas entre entidades (múltiples JOIN, restricciones de integridad referencial).
- Equipos con experiencia en SQL y ecosistema maduro de herramientas (ORM, *frameworks* de migración).
- Carga de trabajo intensiva en lecturas con réplicas de lectura, o consultas analíticas complejas con GROUP BY, *subqueries* y funciones de ventana.

CouchDB es la elección más adecuada en los siguientes escenarios:

- Aplicaciones *offline-first* para dispositivos móviles o entornos con conectividad intermitente que necesiten sincronizar datos al recuperar la red.
- Sistemas que requieren sincronización multirregión sin un punto central de verdad y donde la consistencia eventual es aceptable.
- Proyectos con esquemas de datos que cambian frecuentemente o documentos con estructuras heterogéneas que no encajan en tablas relacionales.
- Aplicaciones IoT que necesitan ingerir grandes volúmenes de documentos independientes de manera distribuida, sin *joins* complejos.

Investigaciones como la de Pokorny (2013) y Kaur y Rani (2013) demuestran que los sistemas NoSQL no se diseñaron para remplazar a los relacionales, sino para abordar escenarios donde el paradigma relacional impone restricciones de escalabilidad o flexibilidad insuperables. En el estudio de Chickerur et al. (2015) se muestra que CouchDB supera a MySQL en inserción

de documentos a gran escala, mientras que MySQL es más eficiente en consultas complejas con múltiples *joins*. Patil et al. (2020) confirman que las diferencias se estrechan para conjuntos de datos pequeños y se amplían en cargas masivas. Desde la perspectiva del teorema CAP de Brewer (2000), se debe considerar esta diferencia fundamental al evaluar requisitos de SLA y arquitectura del sistema.

CONCLUSIONES

A partir del análisis comparativo desarrollado en este artículo, se establecen las siguientes conclusiones:

- MySQL es el SGBD relacional de referencia para aplicaciones transaccionales. Su arquitectura de tres capas con motores de almacenamiento intercambiables, el motor InnoDB con MVCC y bloqueos de fila, así como el soporte completo de SQL, lo hacen ideal para la mayoría de las aplicaciones empresariales convencionales.
- Apache CouchDB representa una solución NoSQL madura para escenarios distribuidos. Su arquitectura basada en Erlang, el modelo de datos JSON flexible, la replicación *multi-master* bidireccional y la API RESTful universal lo convierten en una herramienta poderosa para aplicaciones *offline-first* y sistemas multirregión.
- Ambos motores implementan MVCC, pero con filosofías distintas: InnoDB combina versiones con bloqueos de *gap* para garantizar consistencia fuerte, mientras que CouchDB usa *append-only* sin bloqueos de lectura, priorizando disponibilidad sobre consistencia instantánea.
- La elección del motor se debe basar en el análisis de la naturaleza de los datos (estructurados vs. semiestructurados), los requisitos de consistencia (ACID estricto vs. eventual), los patrones de acceso, la escala y distribución esperada, y la experiencia del equipo de desarrollo.
- En el mercado laboral colombiano e internacional, ambos conocimientos son valiosos: MySQL domina en sistemas legados, ERP y aplicaciones web tradicionales, mientras que CouchDB y otros motores NoSQL son requeridos en proyectos de IoT, aplicaciones móviles y arquitecturas *cloud-native*.
- Como trabajo futuro se recomienda implementar pruebas de carga con sysbench (MySQL) o YCSB para medir el rendimiento en escenarios reales de alta concurrencia, y explorar la integración con PouchDB para patrones *offline-first* en Progressive Web Apps.

REFERENCIAS

- Apache Software Foundation (2023). *Apache CouchDB 3.3 documentation*. <https://docs.couchdb.org/en/stable/>.
- Brewer, E. A. (2000). *Towards robust distributed systems* (p. 7). Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (PODC), 7.
- Chickerur, S., Goudar, A., & Kinnerkar, A. (2015). *Comparison of relational database with document-oriented database (MongoDB) for big data applications* (pp. 41-47). 2015 8th International Conference on Advanced Software Engineering & Its Applications.
- Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), 377-387. <https://doi.org/10.1145/362384.362685>.
- Date, C. J. (2004). *An introduction to database systems* (8th ed.). Addison-Wesley.
- DB-Engines (2025). *DB-Engines ranking*. <https://db-engines.com/en/ranking>.
- Fowler, M., & Sadalage, P. J. (2013). *NoSQL distilled: A brief guide to the emerging world of polyglot persistence*. Addison-Wesley.
- GeeksforGeeks (2023). *Difference between CouchDB and MySQL*. <https://www.geeksforgeeks.org/difference-between-couchdb-and-mysql/>
- IBM (2024a). *SQL vs. NoSQL databases: What's the difference?* IBM Think. <https://www.ibm.com/think/topics/sql-vs-nosql>.
- IBM (2024b). *What is CouchDB?* IBM Think. <https://www.ibm.com/think/topics/couchdb>.
- Kaur, K., & Rani, R. (2013). *Modeling and querying data in NoSQL databases* (pp. 1-7). 2013 IEEE International Conference on Big Data.
- Lehnardt, J. (2014). *Apache CouchDB: The definitive introduction*. InfoQ. <https://www.infoq.com/articles/apache-couchdb-the-definitive-introduction/>.
- MySQL AB (1995). *MySQL reference manual – History of MySQL*. <https://dev.mysql.com/doc/refman/8.0/en/history.html>.
- Oracle Corporation (2023). *MySQL 8.0 reference manual – InnoDB multi-versioning*. <https://dev.mysql.com/doc/refman/8.0/en/innodb-multi-versioning.html>.
- Patil, M., Sonsale, A., & Kamble, A. (2020). Performance analysis of NoSQL and relational databases with CouchDB and MySQL for application's data storage. *Applied Sciences*, 10(23), 8524. <https://doi.org/10.3390/app10238524>.
- Pokorny, J. (2013). NoSQL databases: A step to database scalability in web environment. *International Journal of Web Information Systems*, 9(1), 69-82.
- Schwartz, B., Zaitsev, P., & Tkachenko, V. (2012). *High performance MySQL* (3rd ed.). O'Reilly Media.
- Wikipedia (2025a). *Apache CouchDB*. https://en.wikipedia.org/wiki/Apache_CouchDB.
- Wikipedia (2025b). *MySQL*. <https://es.wikipedia.org/wiki/MySQL>.