

Guía visual de bases de datos

8 tecnologías explicadas de forma clara, elegante y práctica

Relacionales, documentales, embebidas, en memoria y distribuidas.

UNA GUÍA ORIGINAL DE PALENTINO.ES



Qué encontrarás



Arquitectura y organización



Seguridad, copias y alta disponibilidad



Monitorización y herramientas



Operaciones básicas y ejemplos prácticos





Ideal para LinkedIn, formación y consulta rápida





Una visión rápida para **administradores, desarrolladores** y **estudiantes**

Creado por palentino.es · Oscar de la Cuesta

MySQL®

Administrar una base de datos relacional

Organiza tablas, protege el acceso y mantén los datos disponibles.

Cómo se organiza

Servidor → Bases de datos → Tablas

Filas: registros · Columnas: atributos
En MySQL, esquema y base de datos son sinónimos.
SQL es el lenguaje; MySQL es el servidor.

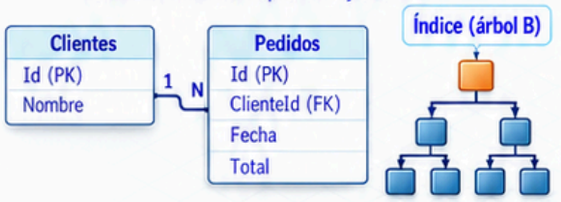




InnoDB: transacciones ACID, claves foráneas y bloqueo por filas.

01 Modelo e índices

Define claves primarias y foráneas.
Elige tipos de datos y utf8mb4.
Crea índices según tus consultas.
EXPLAIN muestra el plan de ejecución.



Más índices también encarecen las escrituras.

02 Seguridad

Usuarios, roles y mínimo privilegio.
TLS para cifrar las conexiones.
Acceso de red restringido.
Parches y credenciales bien gestionadas.



Consultas parametrizadas frente a inyección SQL.

03 Copias y recuperación

mysqldump: copia lógica de datos.
MySQL Shell: utilidades de copia y carga.
Base coherente + binlogs: recuperación temporal.
Copias externas y restauraciones probadas.



Conserva los binlogs necesarios y aplica hasta el punto deseado.

RPO: pérdida tolerable · RTO: tiempo de recuperación.

04 Replicación y disponibilidad

La replicación clásica es asíncrona.
La réplica puede tener retraso.
GTID identifica las transacciones.
InnoDB Cluster ofrece alta disponibilidad.



Group Replication + MySQL Router + MySQL Shell.

Replicar no sustituye a una copia de seguridad.

05 Monitorización y rendimiento

Performance Schema: actividad y esperas.
Slow query log: consultas lentas.
Vigila CPU, disco, conexiones y bloqueos.
Controla el buffer pool y el retraso de réplica.



- CPU y memoria
- Actividad de consultas
- Consultas lentas
- Conexiones y bloqueos
- Buffer pool (InnoDB)
- Retraso de réplica

06 Herramientas del administrador

mysql · cliente de consola SQL.
MySQL Shell · administración y scripts.
Workbench · consultas y modelado visual.
Event Scheduler · eventos SQL programados.



Comprueba la compatibilidad de tus herramientas.

07 · Operaciones básicas en SQL

CRUD: crear, leer, actualizar y eliminar registros.

En una base de pruebas ya creada y seleccionada:

```
CREATE TABLE Personas (
  Id INT PRIMARY KEY,
  Nombre VARCHAR(50),
  Ciudad VARCHAR(50)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

C · Crear

Inserta un registro.

```
INSERT INTO Personas
  (Id, Nombre, Ciudad)
VALUES (101, 'Ana', 'Palencia');
```

R · Leer

Consulta personas de Palencia.

```
SELECT Id, Nombre, Ciudad
FROM Personas
WHERE Ciudad = 'Palencia';
```

U · Actualizar

Cambia la ciudad del registro 101.

```
UPDATE Personas
SET Ciudad = 'Valladolid'
WHERE Id = 101;
```

WHERE filtra las filas. Sin WHERE, UPDATE y DELETE afectan a todas.

D · Eliminar

Elimina el registro con Id 101.

```
DELETE FROM Personas
WHERE Id = 101;
```

Comprueba la selección con SELECT antes de modificar o borrar.

Datos íntegros + acceso seguro + consultas eficientes + recuperación comprobada

Creado por palentino.es · Oscar de la Cuesta

Fuentes: documentación oficial de MySQL · dev.mysql.com/doc

SQL Server

Administrar una base de datos relacional

Organiza tablas, protege el acceso y conserva la integridad de los datos.

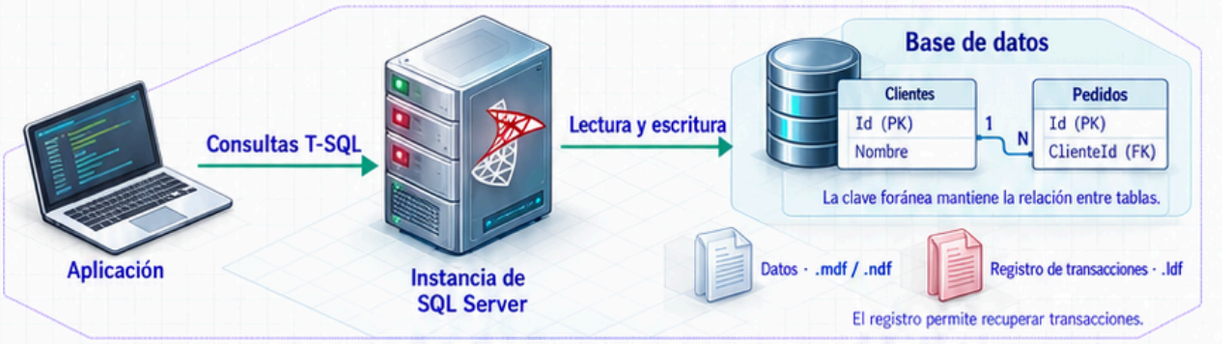
Cómo se organiza

Instancia → Base de datos → Esquema → Tablas

Filas: registros · Columnas: atributos

SQL es el lenguaje; SQL Server es el motor.





Transacciones ACID: atomicidad, consistencia, aislamiento y durabilidad.

01 Modelo e índices

Define claves primarias y foráneas.
Usa tipos de datos y restricciones adecuados.
Diseña índices según las consultas.
Revisa estadísticas y planes de ejecución.



Más índices también encarecen las escrituras.

02 Seguridad

Identidades, usuarios y roles.
Permisos de mínimo privilegio.
TLS para cifrar las conexiones.
Red restringida y parches al día.



Consultas parametrizadas frente a inyección SQL.

03 Copias y recuperación

Completa: base de la recuperación.
Diferencial: cambios desde su copia base.
Log: conserva la cadena de transacciones.
El modelo de recuperación define las opciones.



Según el modelo de recuperación.
Copias externas y restauraciones probadas.

RPO: pérdida tolerable · RTO: tiempo de recuperación.

04 Alta disponibilidad

Always On: réplicas de bases de datos.
Sincronización síncrona o asíncrona.
El failover automático exige condiciones.
Funciones según edición y configuración.



La alta disponibilidad no sustituye al backup.

05 Monitorización y mantenimiento

Query Store: historial de consultas y planes.
Vigila CPU, memoria, disco y bloqueos.
Controla el crecimiento de datos y del log.
DBCC CHECKDB: comprueba la integridad.



06 Herramientas del administrador

SSMS · gestión y consultas.
sqlcmd · consola y automatización.
SQL Server Agent · trabajos programados.
Agent no está incluido en Express.
Planifica copias, alertas y mantenimiento.



07 · Operaciones básicas en T-SQL

CRUD: crear, leer, actualizar y eliminar registros.

En una base de pruebas ya creada, prepara la tabla:

```
CREATE TABLE dbo.Personas (
   Id int PRIMARY KEY,
   Nombre nvarchar(50),
   Ciudad nvarchar(50)
);
```

C · Crear

Inserta un registro.

```
INSERT INTO dbo.Personas
  (Id, Nombre, Ciudad)
VALUES (101, N'Ana', N'Palencia');
```

R · Leer

Consulta personas de Palencia.

```
SELECT Id, Nombre, Ciudad
FROM dbo.Personas
WHERE Ciudad = N'Palencia';
```

U · Actualizar

Cambia la ciudad del registro 101.

```
UPDATE dbo.Personas
SET Ciudad = N'Valladolid'
WHERE Id = 101;
```

D · Eliminar

Elimina el registro con Id 101.

```
DELETE FROM dbo.Personas
WHERE Id = 101;
```

WHERE filtra las filas. Sin WHERE, UPDATE y DELETE afectan a todas.

Comprueba la selección con SELECT antes de modificar o borrar.

Datos íntegros + acceso seguro + consultas eficientes + recuperación comprobada

Creado por palentino.es · Oscar de la Cuesta

Fuentes: Microsoft Learn · learn.microsoft.com/sql

MongoDB

Administrar una base de datos documental

Organiza los datos, protege el acceso y mantén el servicio disponible.

Cómo se organiza

Base de datos → Colecciones → Documentos BSON

Esquema flexible: diseña los datos según tus consultas y valida su estructura.



BSON: representación binaria de documentos.

REPLICA SET · alta disponibilidad



Replicar no sustituye a una copia de seguridad.

01 Modelado e índices

Embede o referencia según el uso.
Crea índices para filtros y ordenación.
Comprueba el plan con `explain()`.



Más índices también encarecen las escrituras.

02 Seguridad

Autenticación y roles de mínimo privilegio.
TLS para cifrar las conexiones.
Acceso de red restringido y parches.



- Usuarios y roles (mínimo privilegio)
- Conexiones cifradas con TLS
- Acceso de red restringido (p. ej. listas de IP)
- Parches y actualizaciones

03 Copias y recuperación

Define RPO: pérdida de datos tolerable.
Define RTO: tiempo de recuperación.
Copias externas y restauraciones probadas.



`mongodump / mongorestore`

Elige un método coherente con tu despliegue.

04 Escalado con sharding

Reparte datos entre varios shards.
La clave de partición determina el reparto.
mongos dirige las consultas.

Cada shard se despliega como replica set.



05 Monitorización

Vigila latencia, consultas lentas y errores.
Controla RAM, disco y conexiones.
Alerta sobre retrasos de replicación.
Revisa logs y capacidad antes de crecer.



- **Rendimiento** (latencia, consultas lentas)
- **Recursos** (RAM, disco, conexiones)
- **Replicación** (retrasos y errores)
- **Logs** (revisa y planifica la capacidad)

06 Herramientas del administrador

mongosh · consola y automatización.
Compass · exploración visual de datos.
Atlas · servicio gestionado en la nube.
Autogestionado · tú operas la infraestructura.



07 · Operaciones básicas en mongosh

CRUD: crear, leer, actualizar y eliminar documentos.

En una base de pruebas, selecciona la base de datos: `use demo`

Al insertar el primer documento se crean la base y la colección si no existen.

C · Crear

Inserta un documento en personas.

```
db.personas.insertOne({
  _id: 101, nombre: "Ana",
  ciudad: "Palencia"
})
```

R · Leer

Busca personas de Palencia.

```
db.personas.find({
  ciudad: "Palencia"
})
```

`find({})` devuelve todos los documentos.

U · Actualizar

Cambia la ciudad del documento 101.

```
db.personas.updateOne(
  { _id: 101 },
  { $set: { ciudad: "Valladolid" } }
)
```

`$set` modifica el campo indicado.

D · Eliminar

Elimina el documento con `_id` 101.

```
db.personas.deleteOne({
  _id: 101
})
```

Comprueba el filtro con `find()` antes de borrar.

El filtro selecciona los documentos. `updateOne()` y `deleteOne()` afectan como máximo a uno.

Datos bien modelados + acceso seguro + rendimiento medido + recuperación comprobada

Creado por palentino.es · Oscar de la Cuesta

Fuentes: documentación oficial de MongoDB · mongodb.com/docs

ORACLE DATABASE

Administrar una base de datos relacional

Protege, organiza y recupera los datos con SQL y PL/SQL.

Cómo se organiza

Instancia → CDB (contenedora) → PDB (conectable) → Esquemas y tablas

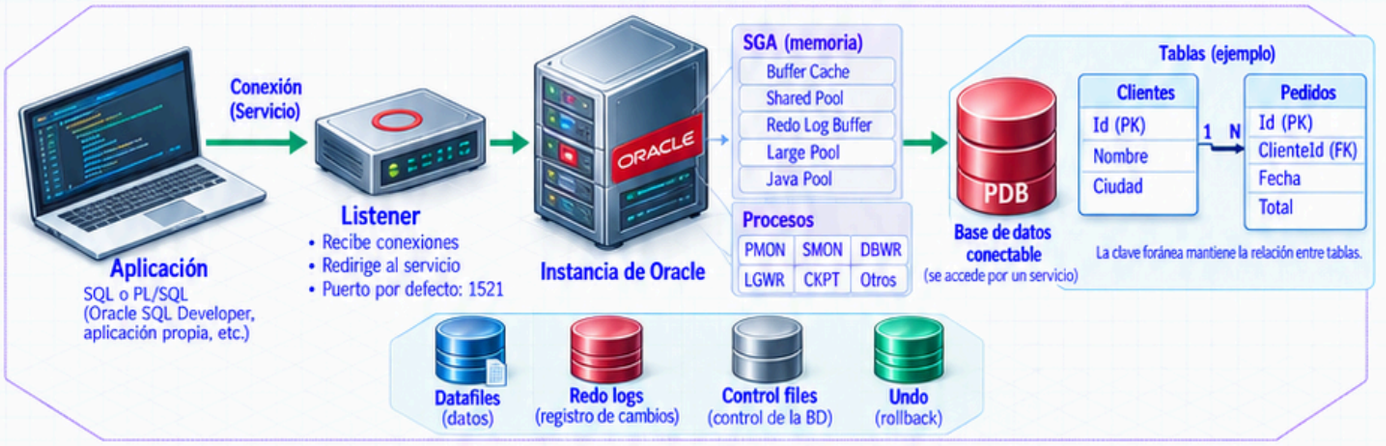


Instancia
SGA + procesos
- Memoria SGA
- Procesos (PMON, SMON, DBWR, LGWR, etc.)

CDB
Base de datos contenedora
- Gestiona PDBs
- Recursos comunes

PDB
Base de datos conectable
- Datos de la aplicación
- Se conecta mediante un servicio

Esquemas y tablas
- Esquemas (usuarios)
- Tablas, vistas, índices, etc.



Transacciones ACID: datos consistentes y recuperables.

01 Modelo e índices

Claves primarias y foráneas.
B-tree y planes de ejecución.
Tipos NUMBER, VARCHAR2 y DATE.



Los índices aceleran las consultas y también encarecen las escrituras.

02 Seguridad

Usuarios, roles y privilegio mínimo.
TLS y cifrado TDE según configuración.
Parches y auditoría.



- Usuarios, roles y privilegio mínimo
- TLS para cifrar las conexiones (cifrado en tránsito)
- TDE para cifrar los datos en disco (según configuración)
- Parches de seguridad y auditoría (AUDIT)

Protege los datos y controla el acceso a la información.

03 Copias y recuperación

RMAN para backup y restore.
Archived redo para recuperación.
Probar restauraciones.



Realiza copias periódicas y prueba la recuperación.

04 Alta disponibilidad

Oracle Data Guard: primaria → standby mediante redo.
RAC: varias instancias, según edición/licencia.
El backup no equivale a replicación.



Alta disponibilidad para minimizar el tiempo de inactividad.

05 Monitorización y rendimiento

Sesiones, esperas, CPU e I/O.
Vistas V$ y planes SQL.
AWR/ASH sujetos a licencia según edición.



- Sesiones, esperas, CPU e I/O
- Vistas V$ y planes SQL (PLAN_TABLE, EXPLAIN)
- AWR/ASH sujetos a licencia según edición
- Identifica cuellos de botella y optimiza consultas

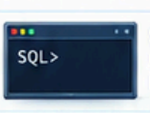
Monitoriza el estado de la base de datos y mejora el rendimiento.

06 Herramientas del administrador

SQL Developer. SQLcl. Oracle Enterprise Manager.
RMAN y Data Pump.



SQL Developer
Entorno gráfico para consultas y administración.



SQLcl
Consola moderna para SQL y PL/SQL.



Oracle Enterprise Manager
Monitorización, gestión y tareas administrativas.

RMAN
Copias y recuperación.

Data Pump
Exportación e importación de datos.

Herramientas oficiales para todas las tareas de administración.

07 · Operaciones básicas en Oracle SQL

CRUD: crear, leer, actualizar y eliminar registros.

En una base de pruebas crea la tabla:

```
CREATE TABLE Personas (
    id NUMBER PRIMARY KEY,
    nombre VARCHAR2(50),
    ciudad VARCHAR2(50)
);
```

C · Crear
Inserta un registro.

```
INSERT INTO Personas
  (id, nombre, ciudad)
VALUES (101, 'Ana', 'Palencia');
```

R · Leer
Consulta registros.

```
SELECT id, nombre, ciudad
FROM Personas
WHERE ciudad = 'Palencia';
```

U · Actualizar
Modifica registros.

```
UPDATE Personas
SET ciudad = 'Valladolid'
WHERE id = 101;
```

D · Eliminar
Elimina registros.

```
DELETE FROM Personas
WHERE id = 101;
```

WHERE evita actualizar o borrar todas las filas. COMMIT confirma la transacción.

Creado por palentino.es • Oscar de la Cuesta

Fuente: Documentación oficial de Oracle • docs.oracle.com

PostgreSQL

Administrar una base de datos relacional avanzada

Organiza datos, protege el acceso y aprovecha SQL avanzado y JSONB.

Cómo se organiza

Instancia → Base de datos → Esquemas → Tablas

Filas: registros · Columnas: atributos y tipos
PostgreSQL es el motor y SQL es el lenguaje.

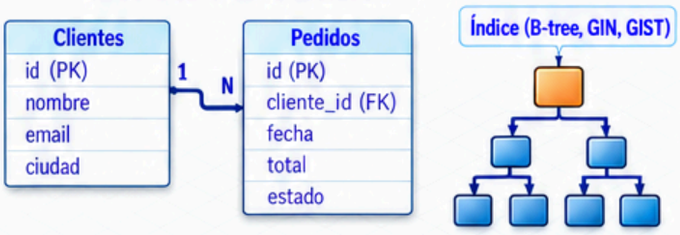




MVCC y transacciones ACID: concurrencia, consistencia y recuperación.

01 Modelo e índices

Define claves primarias y foráneas.
Tipos de datos: INTEGER, TEXT, DATE, JSONB, etc.
Índices B-tree, GIN y GiST. Usa EXPLAIN ANALYZE para analizar el plan de ejecución.



Los índices aceleran las consultas pero también ocupan espacio en disco.

02 Seguridad

Roles, usuarios y mínimo privilegio.
Conexiones cifradas con TLS.
Control de acceso con pg_hba.conf.
Parcheo y gestión de credenciales.

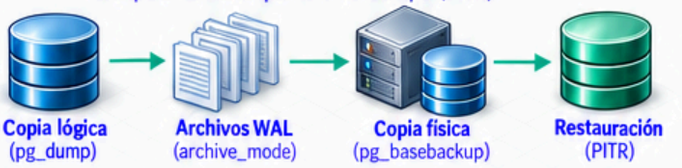


- Roles, usuarios y permisos (mínimo privilegio)
- Conexiones cifradas con TLS (SSL)
- Control de acceso pg_hba.conf
- Parcheo y actualizaciones
- Gestión segura de credenciales (passwords, certificados)

Restringe el acceso a la red, configura TLS y revisa permisos regularmente.

03 Copias y recuperación

pg_dump para copias lógicas.
pg_basebackup para copias físicas.
Archivos WAL y archivado.
Recuperación a un punto en el tiempo (PITR).



RPO: pérdida de datos tolerable · **RTO**: tiempo de recuperación.

04 Replicación y alta disponibilidad

Replicación en streaming: síncrona o asíncrona.
Failover automático con herramientas de clúster.
La replicación no sustituye a las copias de seguridad.



Usa replicación para alta disponibilidad y backups para recuperación ante desastres.

05 Monitorización y rendimiento

Vistas del sistema: pg_stat_* y pg_stat_statements.
VACUUM y autovacuum. Controla CPU, memoria, disco y bloqueos.
Detecta consultas lentas y revisa el plan de ejecución.



- Rendimiento (CPU, memoria, disco)
- Consultas y planes (pg_stat_statements)
- Consultas lentas
- Bloqueos y sesiones activas
- Autovacuum y mantenimiento
- Tamaño de tablas e índices

06 Herramientas del administrador

psql: consola de comandos.
pgAdmin: administración y diseño visual.
pg_dump / pg_restore: copias y restauración.
Herramientas de monitorización y mantenimiento.



07 · Operaciones básicas en SQL

CRUD: crear, leer, actualizar y eliminar registros.

En una base de datos ya creada, crea una tabla:

```
CREATE TABLE personas (
   id INTEGER PRIMARY KEY,
   nombre VARCHAR(50),
); ciudad VARCHAR(50)
```

C · Crear — Inserta un registro.

```
INSERT INTO personas
   (id, nombre, ciudad)
VALUES (101, 'Ana', 'Palencia');
```

R · Leer — Consulta registros.

```
SELECT id, nombre, ciudad
FROM personas
WHERE ciudad = 'Palencia';
```

U · Actualizar — Modifica un registro.

```
UPDATE personas
SET ciudad = 'Valladolid'
WHERE id = 101;
```

D · Eliminar — Elimina un registro.

```
DELETE FROM personas
WHERE id = 101;
```

Datos consistentes + acceso seguro + consultas potentes + recuperación comprobada

Creado por palentino.es · Oscar de la Cuesta
Fuente: documentación oficial de PostgreSQL · postgresql.org/docs

SQLite

Administrar una base de datos embebida

Ligera, portable y almacenada en un archivo local para aplicaciones y dispositivos.

Cómo se organiza

Aplicación → Biblioteca SQLite → Archivo .db → Tablas

Filas: registros · Columnas: atributos y tipos de datos.
No necesita un servidor independiente.

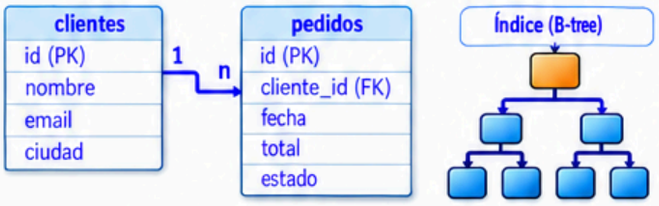




Transacciones ACID: integridad y simplicidad en un único archivo.

01 Modelo e índices

Define claves primarias y foráneas.
Tipos de datos: INTEGER, TEXT, REAL, BLOB.
Índices B-tree para mejorar el rendimiento.
Usa EXPLAIN QUERY PLAN para analizar consultas.



Los índices aceleran las consultas y ocupan espacio en el archivo.

02 Seguridad

Permisos del sistema de archivos
Controla quién puede leer y escribir el archivo .db.

Protección del dispositivo
Usa cifrado de disco y permisos del sistema operativo.

Copias cifradas (según solución utilizada)
El cifrado del archivo depende de la aplicación
o de extensiones específicas (p. ej. SQLCipher).

Acceso desde la aplicación
La app accede mediante la biblioteca SQLite.

Buenas prácticas
Restringe permisos, protege copias y verifica integridad.

03 Copias y recuperación

Copia el archivo .db directamente.
Realiza backups en frío o coordinados (modo WAL).
Incluye los archivos -wal y -journal si existen.
La restauración es tan simple como reemplazar el archivo.



Una copia consistente incluye el archivo .db y sus archivos WAL/journal (si existen).

04 Concurrencia y disponibilidad

Un único escritor y múltiples lectores.
El modo WAL mejora la concurrencia.
Ideal para aplicaciones locales y embebidas.
No está pensado para clústeres distribuidos.



Modo WAL
- Más concurrencia
- Lectores sin bloqueo
- Escritor exclusivo
- Ideal para entorno local
- No para clúster distribuido

WAL permite que los lectores no bloqueen al escritor, mejorando la disponibilidad.

05 Monitorización y rendimiento

Usa ANALYZE para actualizar estadísticas.
VACUUM para compactar y optimizar el archivo.
Controla el tamaño del archivo y el uso de índices.
Analiza consultas lentas con EXPLAIN QUERY PLAN.



- ANALYZE (estadísticas)
- VACUUM (compacta el archivo)
- Tamaño del archivo (.db)
- Índices y uso de espacio
- Consultas lentas (EXPLAIN QUERY PLAN)
- Rendimiento de la aplicación

Un archivo optimizado mejora el rendimiento y reduce el espacio en disco.

06 Herramientas del administrador

Línea de comandos, herramientas gráficas y bibliotecas.
Útiles para explorar, administrar y analizar la base de datos.



sqlite3
(línea de comandos)
Consultas y administración directa.

DB Browser for SQLite
(interfaz gráfica)
Explorar datos, ejecutar consultas y gestionar la BD.

Bibliotecas y drivers
(C, Python, Java, etc.)
Integración en aplicaciones mediante la biblioteca SQLite.

07 • Operaciones básicas en SQL

CRUD: crear, leer, actualizar y eliminar registros.

C · Crear
Crea una tabla.

```
CREATE TABLE personas (
  id INTEGER PRIMARY KEY,
  nombre TEXT,
  ciudad TEXT
);
```

R · Insertar
Inserta un registro.

```
INSERT INTO personas
  (id, nombre, ciudad)
VALUES (101, 'Ana',
        'Palencia');
```

R · Consultar
Consulta registros.

```
SELECT id, nombre, ciudad
FROM personas
WHERE ciudad = 'Palencia';
```

U · Actualizar
Modifica un registro.

```
UPDATE personas
SET ciudad = 'Valladolid'
WHERE id = 101;
```

D · Eliminar
Elimina un registro.

```
DELETE FROM personas
WHERE id = 101;
```

Simplicidad + portabilidad + acceso local + recuperación directa.

Creado por palentino.es • Oscar de la Cuesta

Fuente: documentación oficial de SQLite • sqlite.org/docs

Redis

Administrar una base de datos en memoria

Acelera aplicaciones con claves, estructuras de datos, caché y persistencia opcional.

Cómo se organiza

Cliente → Servidor Redis → Claves y estructuras de datos

- Opera principalmente en memoria RAM (muy rápido).
- Puede persistir los datos en disco con RDB y/o AOF.





Claves con TTL
Expiración automática de datos

Publicación / Suscripción
(Pub/Sub)
Mensajería en tiempo real

Colas (Lists)
LPUSH / RPOP
para tareas en segundo plano

Velocidad en memoria + persistencia opcional + replicación y alta disponibilidad.

01 Modelo y estructuras

Claves (strings) y múltiples estructuras de datos.
Soporta TTL para datos temporales.

Strings
SET
GET

Hashes
HSET
HGETALL

Lists
LPUSH
LRANGE

Sets
SADD
SMEMBERS

Sorted Sets
ZADD
ZRANGE

Streams
XADD
XRANGE

Claves y TTL

```
SET clave valor
EXPIRE clave segundos
TTL clave
PERSIST clave
DEL clave
```

TTL: permite que las claves expiren automáticamente.

02 Seguridad

Protege el acceso y la comunicación con Redis.



AUTH y ACL
Autenticación y control de permisos por usuario (ACL).

Conexiones cifradas con TLS
Protege la comunicación cliente-servidor.

Red restringida
Escucha solo en redes confiables.

Gestión de secretos
Contraseñas en variables de entorno o gestores de secretos.

Parcheo y actualizaciones
Mantén Redis actualizado.

03 Persistencia y recuperación

Guarda los datos en disco y permite su recuperación.



Recuperación y backups

Restauración
Desde RDB o AOF.

Estrategia de backup
Copias periódicas del RDB y/o AOF.

Pruebas de recuperación
Verifica que puedes restaurar los datos.

04 Replicación y disponibilidad

Replica datos y garantiza la alta disponibilidad.



Alta disponibilidad

Redis Sentinel
Monitorización y failover automático

Redis Cluster
Particionado y escalado horizontal

05 Monitorización y rendimiento

Supervisa el estado y optimiza el rendimiento.



Latencia
Tiempo de respuesta de comandos.

Uso de memoria
Memoria utilizada, fragmentación.

Hits / Misses
Efectividad del caché.

Claves expiradas
Número de expiraciones (TTL).

CPU y conexiones
Uso de CPU y clientes conectados.

06 Herramientas del administrador

Utilidades para gestionar, monitorear y operar Redis.

redis-cli
Cliente de línea de comandos. Gestiona, consulta y administra.

Redis Insight
Interfaz visual. Explora claves, analiza memoria y monitoriza.

Exportación e importación
RDB, AOF o herramientas como redis-cli --rdb/--aof.

Monitorización
INFO, SLOWLOG, comandos de estadísticas y herramientas externas (Prometheus, Grafana).

07 · Operaciones básicas en Redis

CRUD: crear, leer, actualizar y eliminar claves.

Ejemplos en redis-cli
Conéctate a tu servidor Redis y ejecuta los siguientes comandos.

```
$ redis-cli
>
```

C Crear

Crear claves y estructuras.

```
> SET persona:101:ciudad "Palencia"
OK
> HSET persona:101 nombre "Ana" \
      ciudad "Palencia"
  (integer) 2
```

SET: crea una clave string.
HSET: crea/actualiza campos en un hash.

R Leer

Obtener el valor de las claves.

```
> GET persona:101:ciudad
  "Palencia"
> HGETALL persona:101
  1) "nombre"
  2) "Ana"
  3) "ciudad"
  4) "Palencia"
```

GET: recupera el valor de una clave.
HGETALL: devuelve todos los campos del hash.

U Actualizar

Modificar valores existentes.

```
> HSET persona:101 ciudad \
    "Valladolid"
  (integer) 0
```

HSET: actualiza un campo del hash (si existe, lo sobrescribe).

D Eliminar

Eliminar claves.

```
> DEL persona:101:ciudad
  (integer) 1
> DEL persona:101
  (integer) 1
```

DEL: elimina una clave (string, hash, etc.).

Baja latencia + datos temporales + escalado + alta velocidad.

Cassandra

Administrar una base de datos distribuida

Diseñada para alta disponibilidad, escalado horizontal y grandes volúmenes de datos.

Cómo se organiza

Cluster → Datacenter → Keyspace → Tablas

Datos particionados y distribuidos en múltiples nodos.
Cada fila pertenece a una partición (*partition key*) y contiene columnas. Se replica entre nodos y *datacenters*.



Cluster (conjunto de nodos) → **Datacenter** (centro de datos) → **Keyspace** (espacio de claves) → **Tablas** (filas y columnas)



Escalado horizontal + replicación + tolerancia a fallos + consistencia configurable.

01 Modelado y particionado

Diseña las tablas según las consultas. Define la partition key y, opcionalmente, clustering columns. Desnormaliza y modela para evitar joins.

Tabla: clientes
id (PK)
nombre
email
ciudad

Tabla: pedidos
cliente_id (PK)
fecha (clustering)
total
estado

Partition key
(id o cliente_id)



Los datos se distribuyen por la clave de partición en distintos nodos.

02 Seguridad

Roles, usuarios y permisos. Conexiones cifradas con TLS.
Control de acceso a keyspaces y tablas. Red restringida.
Parcheo y cifrado según el despliegue.



- Roles y permisos (CREATE ROLE, GRANT)
- Conexiones cifradas con TLS/SSL
- Red restringida (firewall, VPC, subredes)
- Parcheo y actualizaciones
- Cifrado en tránsito y en reposo (según despliegue)

Si un nodo falla, las operaciones continúan gracias a la replicación.

03 Copias y recuperación

Snapshots de nodos (copia online de datos).
Herramienta *sstableloader* para restauraciones.
Backups periódicos y restauración planificada.



Snapshot (nodo) → **Archivos SSTable** (datos en disco) → **sstableloader** (restauración) → **Backup** (periódico) → **Restauración** (planificada)

Combina snapshots, copias externas y pruebas de restauración.

04 Replicación y alta disponibilidad

Replication factor (RF). Nodos y datacenters.
Niveles de consistencia (CONSISTENCY).
Failover implicito y tolerancia a fallos.



Si un nodo falla, las operaciones continúan gracias a la replicación.

05 Monitorización y rendimiento

Supervisa latencia, throughput, uso de disco, compaction, reparaciones y estado de nodos. Detecta hot partitions y ajusta el modelo de datos y la configuración.



- Latencia (lecturas/escrituras)
- Uso de CPU, memoria y disco
- Compaction (procesos y backlog)
- Reparaciones (nodetool repair)
- Estado de nodos y datacenters
- Detección de hot partitions
- Ajuste de configuración y rendimiento

06 Herramientas del administrador

cqlsh: consola de consultas CQL.
nodetool: administración y mantenimiento.
Herramientas de monitorización y operación.



cqlsh (consultas CQL) · **nodetool** (administración) · **Grafana** (monitorización) · **Herramientas** (backup, operación, alertas, etc.)

07 • Operaciones básicas en CQL

CRUD: crear, leer, actualizar y eliminar registros.

```
CREATE TABLE personas (
   id int,
   ciudad text,
   nombre text,
   PRIMARY KEY (id)
);
```

En Cassandra, las consultas deben apoyarse en la clave de partición o en el modelo de datos previsto para ser eficientes.

C • Crear — Crea la tabla (si no existe).

```
CREATE TABLE personas (
   id int,
   ciudad text,
   nombre text,
   PRIMARY KEY (id)
);
```

I • Insertar — Inserta un registro.

```
INSERT INTO personas
  (id, nombre, ciudad)
  VALUES (101, 'Ana',
          'Palencia' );
```

R • Consultar — Recupera un registro (por clave).

```
SELECT id, nombre, ciudad
FROM personas
WHERE id = 101;
```

U • Actualizar — Modifica un registro.

```
UPDATE personas
SET ciudad = 'Valladolid'
WHERE id = 101;
```

D • Eliminar — Elimina un registro.

```
DELETE FROM personas
WHERE id = 101;
```

Disponibilidad + escalado + replicación + resiliencia

Creado por palentino.es · Oscar de la Cuesta

Fuente: documentación oficial de Apache Cassandra · cassandra.apache.org/doc

Comparativa rápida de bases de datos

Cuándo usar cada tecnología y qué ofrece



- Diferentes necesidades
- Distintos escenarios
- Una solución para cada caso

Característica	MySQL	SQL Server	MongoDB	Oracle Database	PostgreSQL	SQLite	Redis	Cassandra
Tipo	Relacional	Relacional	No relacional	Relacional	Relacional	Relacional	No relacional	No relacional
Modelo de datos	Tablas (ACID)	Tablas (ACID)	Documental (JSON/BSON)	Tablas (ACID)	Tablas avanzadas (ACID)	Tablas (archivo local)	Clave-valor y estructuras	Wide-column (columnas anchas)
Lenguaje / interfaz	SQL	T-SQL	Consultas documentales (MQL)	SQL (PL/SQL)	SQL (PL/pgSQL)	SQL (ligero)	Comandos (Redis CLI)	CQL (Cassandra Query Language)
Punto fuerte	Sencillez y uso general	Integración con ecosistema Microsoft	Flexibilidad de esquema	Alta fiabilidad y funciones avanzadas	Potente, extensible y estándar	Ligereza y portabilidad	Velocidad en memoria	Escalado masivo y alta escritura
Escalado	Vertical y replicación	Vertical y clústeres	Horizontal (sharding)	Vertical y RAC (clúster)	Vertical y horizontal	Limitado (una sola máquina)	Horizontal (clústeres)	Horizontal (nativo)
Alta disponibilidad	Replicación y clúster	Always On y clústeres	Réplicas y sharding	RAC y Data Guard	Streaming replication	Limitada	Clúster y replicación	Multi-datacenter (nativo)
Caso de uso típico	Aplicaciones web y pymes	Entornos empresariales Microsoft	Aplicaciones modernas y datos flexibles	Sistemas críticos empresariales	Aplicaciones avanzadas y analítica	Aplicaciones móviles, IoT y embebidas	Caché, sesiones y tiempo real	Grandes volúmenes de datos

Lectura rápida: mejor para...

Una frase por tecnología para identificar su uso principal.

MySQL
Mejor para: Aplicaciones web, proyectos generales y pymes.

SQL Server
Mejor para: Entornos empresariales con ecosistema Microsoft.

MongoDB
Mejor para: Aplicaciones modernas y datos flexibles en formato JSON.

Oracle Database
Mejor para: Sistemas críticos de gran escala en entornos empresariales.

PostgreSQL
Mejor para: Aplicaciones avanzadas, analítica y proyectos exigentes.

SQLite
Mejor para: Aplicaciones ligeras, móviles, IoT y embebidas.

Redis
Mejor para: Caché, sesiones, tiempo real y aplicaciones de alta velocidad.

Cassandra
Mejor para: Grandes volúmenes de datos distribuidos y alta escritura.

Mapa mental: familias de bases de datos

Tecnologías agrupadas por su modelo y tipo de uso.

Relacionales

Datos en tablas, transacciones ACID y gran madurez.

MySQL · SQL Server · Oracle Database · PostgreSQL · SQLite

Desde proyectos ligeros hasta sistemas empresariales avanzados.

Documental

Documentos flexibles en formato JSON/BSON.

MongoDB

Datos flexibles y esquemas cambiantes.

En memoria

Clave-valor y estructuras de datos.

Redis

Máxima velocidad en tiempo real.

Distribuida

Wide-column y escalado horizontal.

Cassandra

Grandes volúmenes de datos y alta escalabilidad.

No hay una base de datos universal: el contexto define la elección.

Creado por palentino.es · Oscar de la Cuesta.