Tabellarische Gegenüberstellung von SQL- und MongoDB-Befehlen für CRUD mit Beispielen

Bei der Erstellung des Dokuments kam Grok als KI zum Einsatz.

Quelle für den Vergleich: Link

Dür die Beispiele wird angenommen, dass es eine **Tabelle** in einer SQL-Datenbank gibt, die `user` heißt und über die Spalten `name`und `age` verfügt. In MongoDB ist `users` eine collection, in der jedes Dokument Felder für `name` und `age` hat.

Tabellarische Gegenüberstellung

CRUD-Operation	SQL (z. B. MySQL)	MongoDB	Beispiele
Create (nicht mit SQL create zu verwechseln!)	`INSERT INTO table_name (column1, column2) VALUES (value1, value2);`	`db.collection.insertOne({ "field1": value1, "field2": value2 });`	**SQL**: `INSERT INTO users (name, age) VALUES ('Alice', 25);` **MongoDB**: `db.users.insertOne({ "name": "Alice", "age": 25 });`
	`INSERT INTO table_name (column1, column2) VALUES (v1, v2), (v3, v4);`	`db.collection.insertMany([{ "field1": v1, "field2": v2 }, { "field1": v3, "field2": v4 }]);`	**SQL**: `INSERT INTO users (name, age) VALUES ('Bob', 30), ('Charlie', 28);` **MongoDB**: `db.users.insertMany([{ "name": "Bob", "age": 30 }, { "name": "Charlie", "age": 28 }]);`
Read	`SELECT * FROM table_name;`	`db.collection.find({});`	**SQL**: `SELECT * FROM users;` **MongoDB**: `db.users.find({});`

CRUD-Operation	SQL (z. B. MySQL)	MongoDB	Beispiele
	`SELECT column1, column2 FROM table_name WHERE condition;`	`db.collection.find({ "field": condition }).project({ "field1": 1, "field2": 1 });`	**SQL**: `SELECT name, age FROM users WHERE age > 25;` **MongoDB**: `db.users.find({ "age": { $gt: 25 } }).project({ "name": 1, "age": 1, "_id": 0 });`
	`SELECT * FROM table_name WHERE column1 = value LIMIT 1;`	`db.collection.findOne({ "field1": value });`	**SQL**: `SELECT * FROM users WHERE name = 'Alice' LIMIT 1;` **MongoDB**: `db.users.findOne({ "name": "Alice" });`
Update	`UPDATE table_name SET column1 = value1 WHERE condition;`	`db.collection.updateOne({ "condition": value }, { $set: { "field1": value1 } });`	**SQL**: `UPDATE users SET age = 26 WHERE name = 'Alice';` **MongoDB**: `db.users.updateOne({ "name": "Alice" }, { $set: { "age": 26 } });`
	`UPDATE table_name SET column1 = value1 WHERE condition;` (mehrere Zeilen)	`db.collection.updateMany({ "condition": value }, { $set: { "field1": value1 } });`	**SQL**: `UPDATE users SET age = 30 WHERE age > 28;` **MongoDB**: `db.users.updateMany({ "age": { $gt: 28 } }, { $set: { "age": 30 } });`
Delete	`DELETE FROM table_name WHERE condition;`	`db.collection.deleteOne({ "condition": value });`	**SQL**: `DELETE FROM users WHERE name = 'Alice';` **MongoDB**: `db.users.deleteOne({ "name": "Alice" });`
	`DELETE FROM table_name WHERE condition;` (mehrere Zeilen)	`db.collection.deleteMany({ "condition": value });`	**SQL**: `DELETE FROM users WHERE age > 30;` **MongoDB**: `db.users.deleteMany({ "age": { $gt: 30 } });`

Erklärungen zu den Beispielen und weitere Hinweise

- **Read**: SQL verwendet `SELECT` für Abfragen,
  MongoDB `find` oder `findOne`. MongoDBs `project` entspricht der Spaltenauswahl in SQL. Eine 1 bedeutet hier, dass die Spalte ausgewählt wird. Eine 0, dass diese nicht angezeigt wird. Entpsrechend unterdrückt `_id: 0` das Standardfeld `_id` in der Ausgabe.
- MongoDB-Befehle können flexibel für unterschiedliche Dokumente angewendet werden. In den obigen Beispielen ist es wichtig, dass die Felder enthalten sind, nach denen abgefragt wird. SQL ist auf strikte Tabellenstrukturen angewiesen und bei SELECT bedingt flexibel.
- MongoDBs Operatoren wie `$gt` (greater than) ersetzen SQL-Bedingungen wie `>`.