Database. Relational and no-sql concepts. Basics of SQL. MySQL and MongoDb.

Speaker:

Roman Tikhiy

MySQL

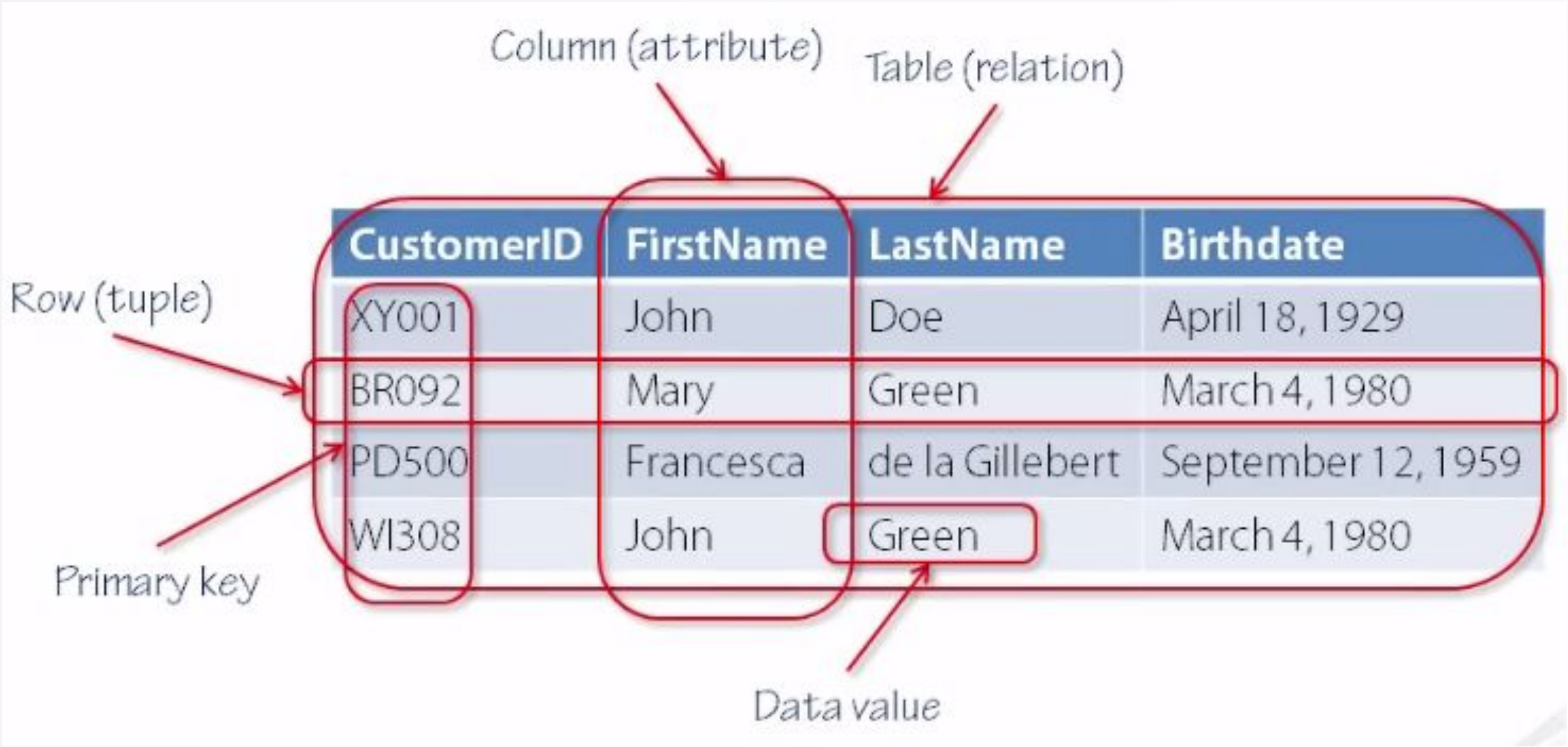
mongoDB

Agenda

- Relational data model
- Types of relations
- Keys
- What is ERD?
- Basics of SQL (DDL & DML)
- Fulltext search index
- Stored procedures
- Transactions
- MongoDb
- Data structure in mongo
- Sample queries
- MySQL vs MongoDb

Relational model terminology

Table – Relation (Entity)
Column – Attribute (Property)
Row – Tuple (Instance)


Column (attribute)
Table (relation)
Row (tuple)
Primary key
Data value
CustomerID
FirstName
LastName
Birthdate
XY001
John
Doe
April 18, 1929
BR092
Mary
Green
March 4, 1980
PD500
Francesca
de la Gillebert
September 12, 1959
WI308
John
Green
March 4, 1980

Basic steps for relational data modeling

1. - Define entities and their attributes.
2. - Define primary keys for each table.
3. - Define foreign keys and types of relations among tables.
4. - Optimization and improving.

Atomic attributes

Tip: split complex objects(attributes) to small atomic properties.

Example:
FullName -> FirstName, LastName
Address -> Street, City, Country...

What is attribute domain?

Set of values allowed in an attribute.

```
Rooms in hotel (1-300)
Age (1-99)
Married (yes or no)
Nationality (Nepalese, Indian, American, or British)
Colors (Red, Yellow, Green)
```

NULL

Primary key

It is the unique identifier of row(tuple, instance).
It MUST:
- be unique all the time
- always has value. Can`t be NULL.
- not change the value.

Alternate keys

Some entities like `user` could have several unique `not null` properties, like `phone` or `email`.

These properties are potential keys and theoretically they can be primary key.

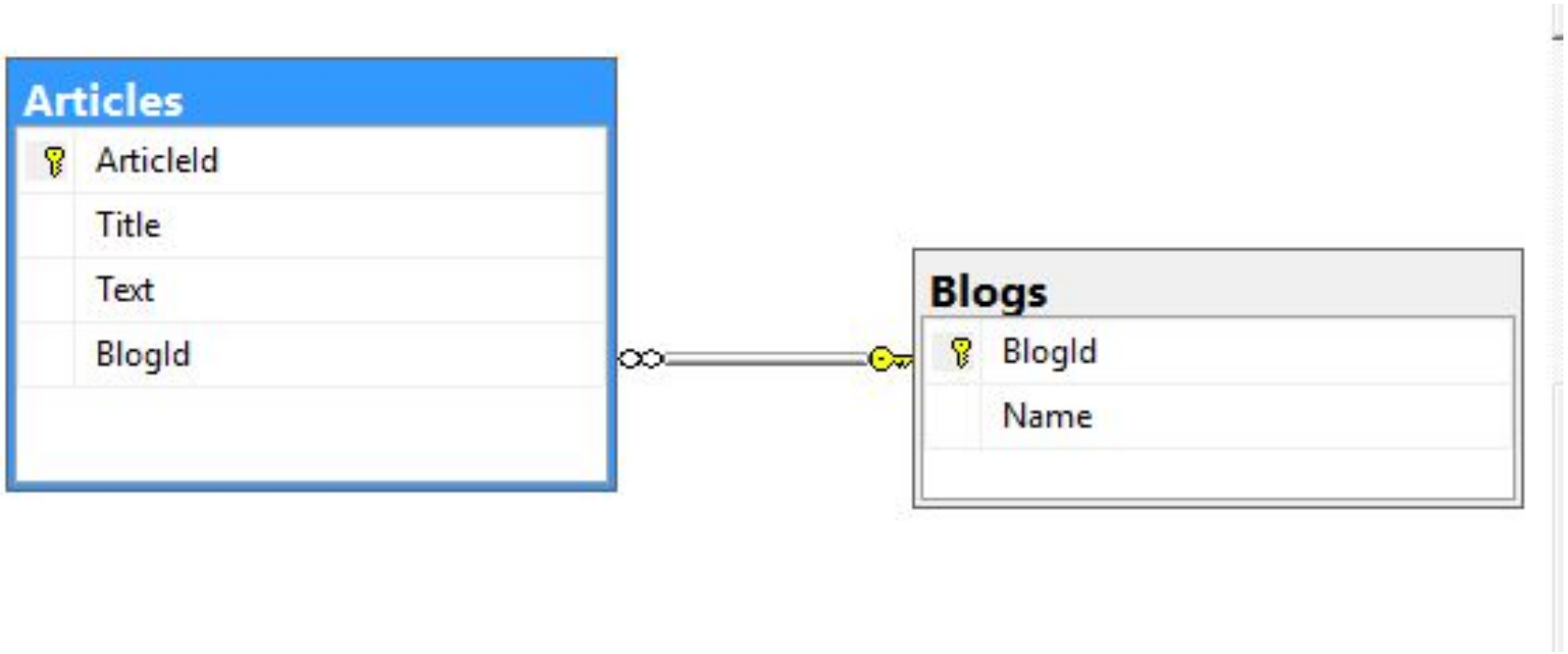
Bu they can be changed, so to avoid this issue, you need to define separate property for PK.

Foreign key

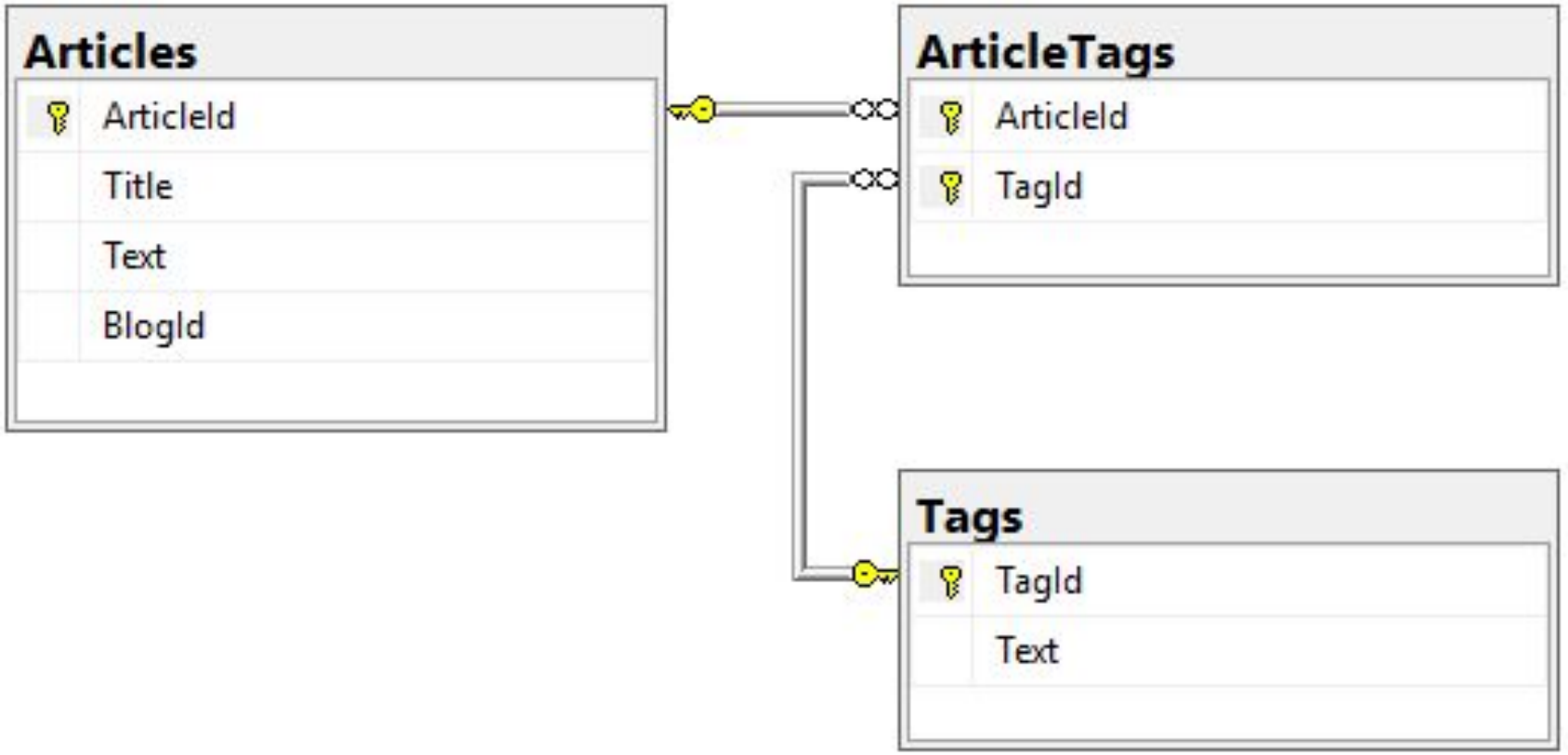
Attribute or set of attributes that refers to primary key of relation.

One to one

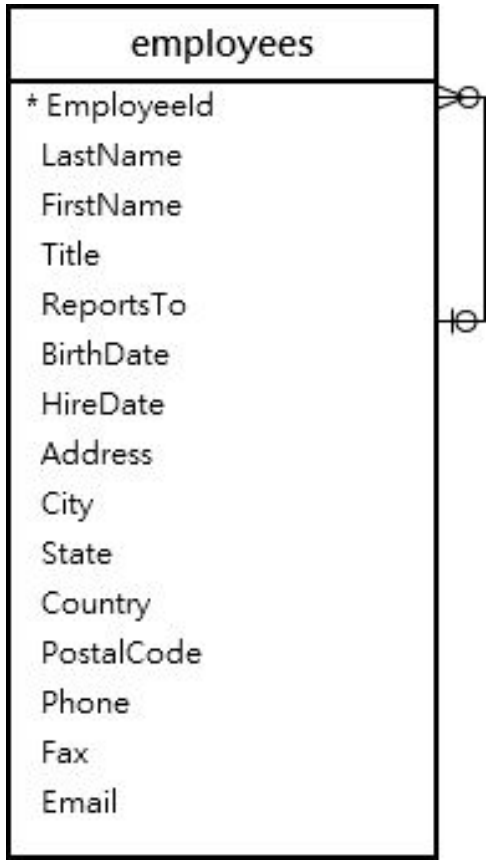
One to many



Many to many


Articles
ArticleId
Title
Text
BlogId
ArticleTags
ArticleId
TagId
Tags
TagId
Text

Self referencing

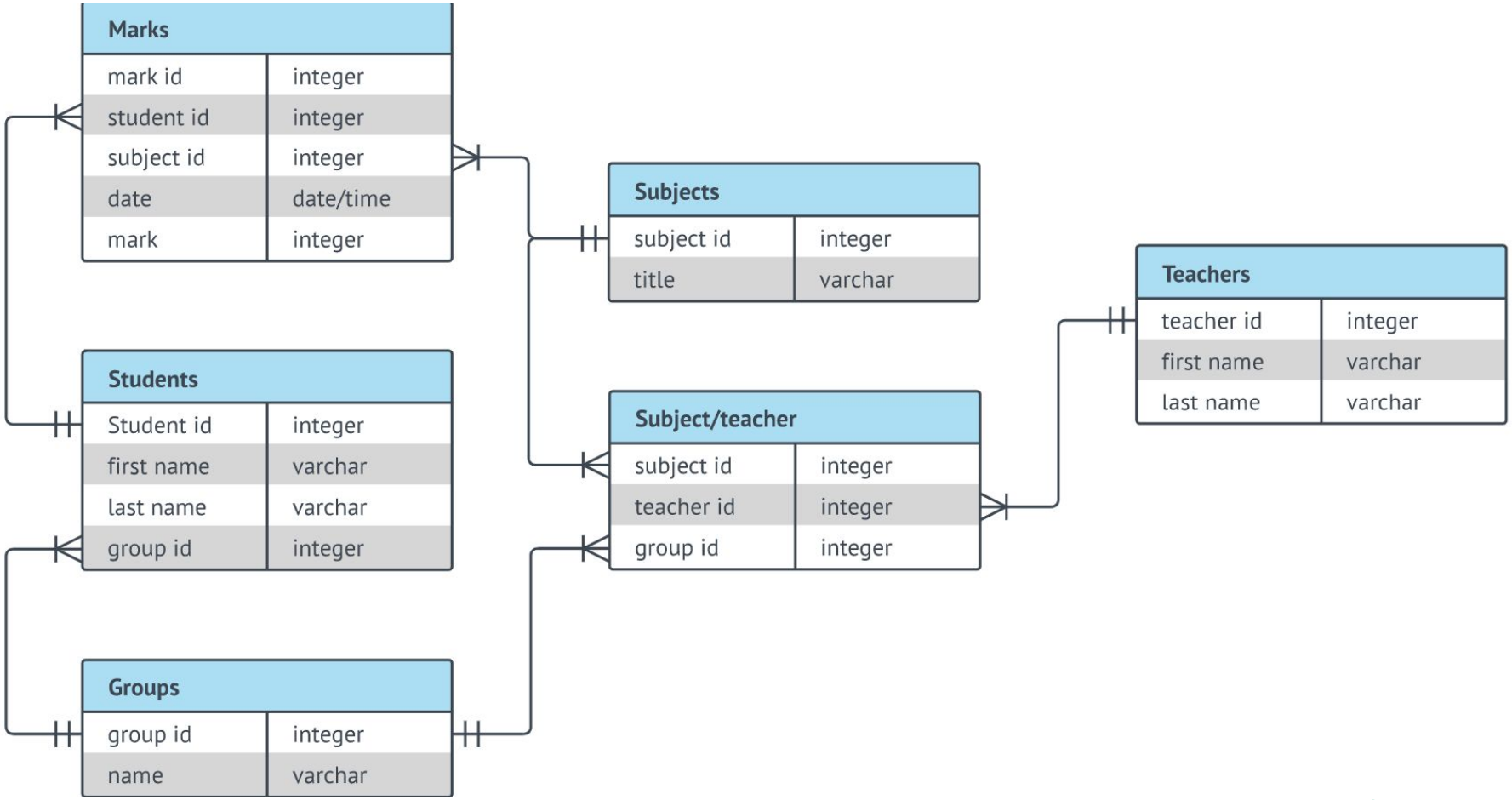


Delete behaviors

Cascade
Set NULL
Set default

ERD

An Entity Relationship (ERD) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.



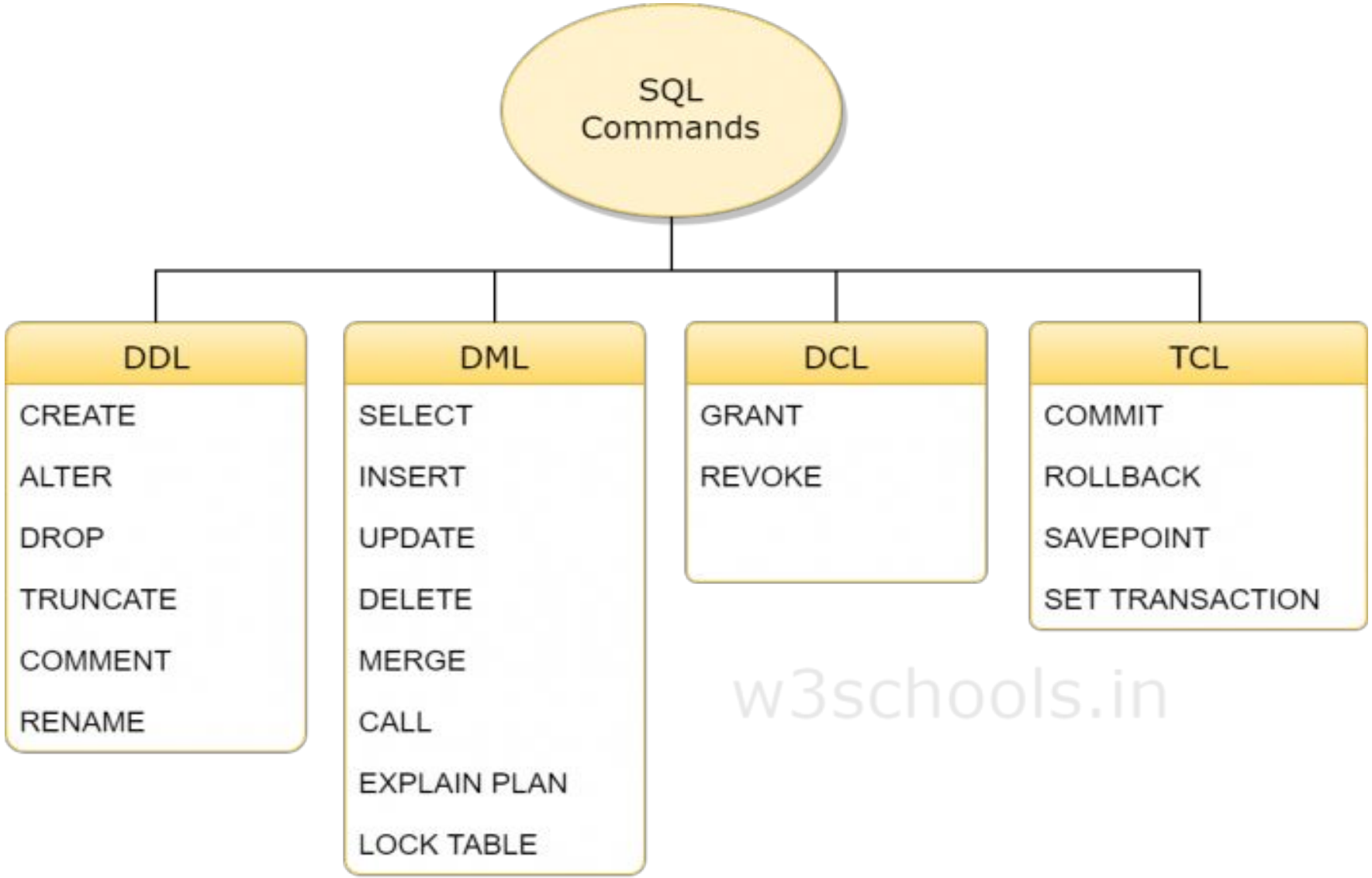
What is RDBMS?

RDBMS — (relational database management system) software package containing utilities that enable the administration of and access to relational databases.

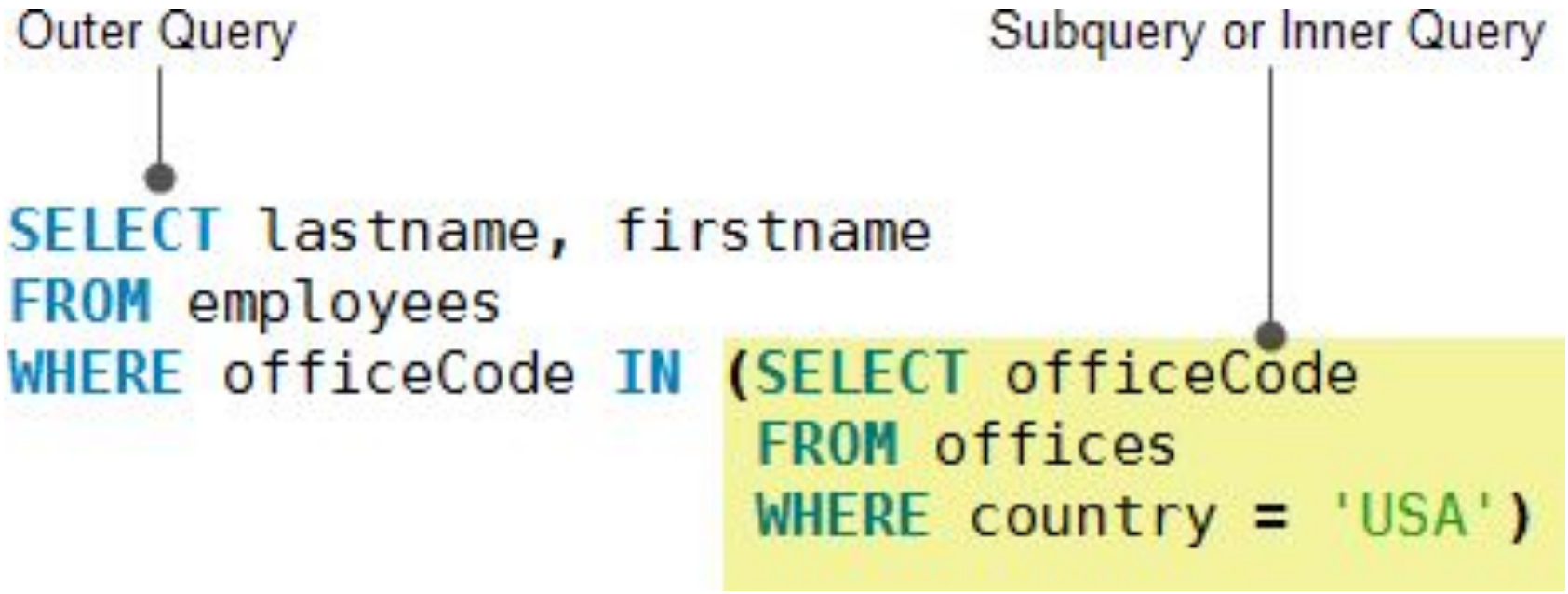
Open-source RDBMS

MySQL DATA TYPES

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-92233720368547758 08 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)


SQL Commands
DDL
CREATE
ALTER
DROP
TRUNCATE
COMMENT
RENAME
DML
SELECT
INSERT
UPDATE
DELETE
MERGE
CALL
EXPLAIN PLAN
LOCK TABLE
DCL
GRANT
REVOKE
TCL
COMMIT
ROLLBACK
SAVEPOINT
SET TRANSACTION
w3schools.in

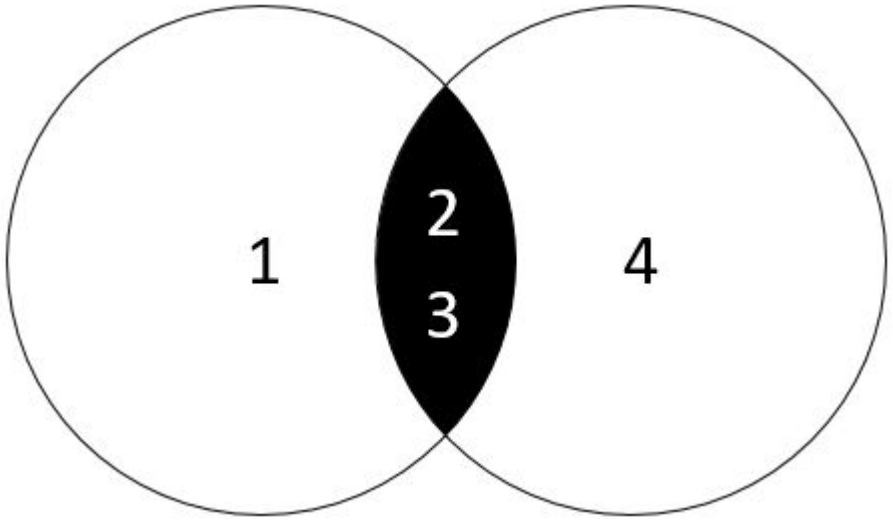
Subquery


Outer Query
Subquery or Inner Query
SELECT lastname, firstname
FROM employees
WHERE officeCode IN (SELECT officeCode
FROM offices
WHERE country = 'USA')

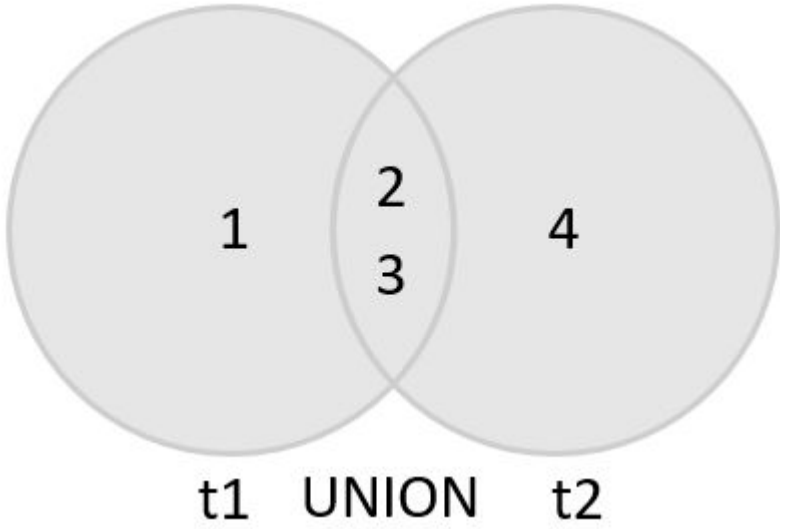
Filtering

Operator	Description	Sample
<	Less than	WHERE `price` < 100 …
>	Greater than	WHERE `price` > 100 …
<=	Less than or equals to	WHERE `price` <= 100 …
>=	Greater than or equals to	WHERE `price` >= 100 …
=	Equals to	WHERE `price` = 100 …
!= (<>)	Not equals to(alternate syntax)	WHERE `price` != 100 …
IN	Comparison function used for checking membership against a list of values or expressions.	WHERE `price` IN (100, 200, 300) …
NOT IN	Comparison function used for checking membership not found in a list of values or expressions.	WHERE `city` NOT IN (`Ternopil`, `Lviv`) .
BETWEEN	Operator to determine whether a value is in a range of values.	WHERE `price` BETWEEN 120 AND 350
LIKE	Operator to query data based on a specified pattern.	WHERE `email` LIKE `t%`
LIMIT	Clause to constrain the number of rows returned by a query.	LIMIT 5 – takes first 5 tuples LIMIT 10, 20 – skips 10, takes 20
ORDER BY	Sorts a data set in ascending order.	ORDER BY `lastName`
ORDER BY DESC	Sorts a data set in descending order.	ORDER BY `lastName`, `age` DESC
DISTINCT	Eliminates duplicate rows in a result set.	SELECT DISTINCT `username` FROM …

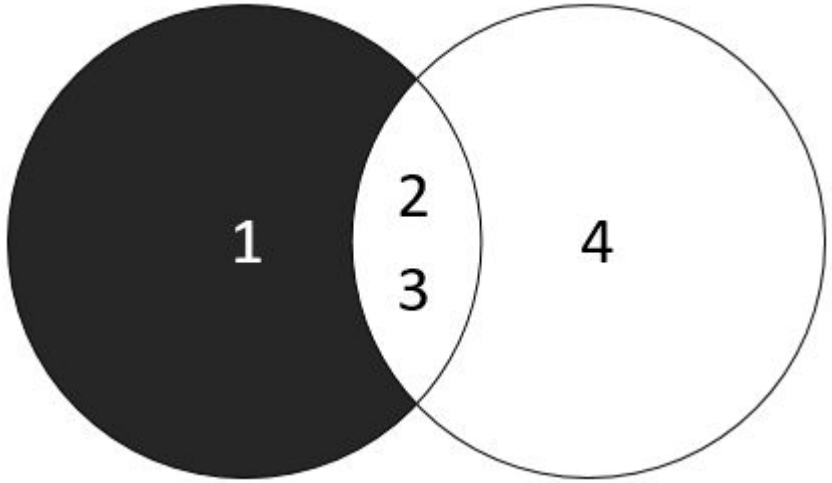
Set operations




```
(SELECT column_list
FROM table_1)
INTERSECT
(SELECT column_list
FROM table_2);
```




```
SELECT id
FROM t1
UNION
SELECT id
FROM t2;
```




```
SELECT id FROM
t1
MINUS
SELECT id FROM
t2;
```

Json data type

Create Table

```
CREATE TABLE events(
  id int auto_increment primary key,
  event_name varchar(255),
  visitor varchar(255),
  properties json,
  browser json
);
```

```
SELECT id, browser->'$.name' browser
FROM events;
```

```
+----+---------+
| id | browser |
+----+---------+
|  1 | Safari  |
|  2 | Firefox |
|  3 | Safari  |
+----+---------+
6 rows in set (0.00 sec)
```

Insert data

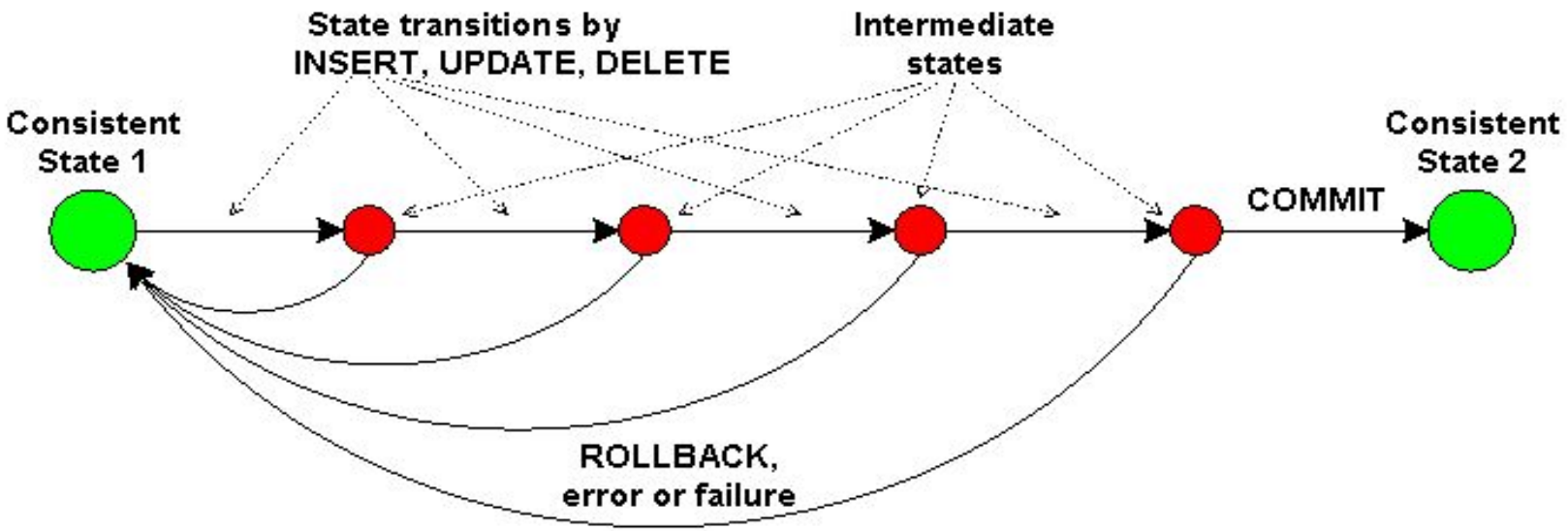
```
INSERT INTO
events(event_name,
visitor,properties, browser)
VALUES (
  'pageview',
   '1',
   '{ "page": "/" }',
   '{ "name": "Safari", "os":
"Mac", "resolution": { "x": 1920,
"y": 1080 } }'),
('pageview',
  '2',
  '{ "page": "/contact" }',
  '{ "name": "Firefox", "os":
"Windows", "resolution": { "x":
2560, "y": 1600 } }'),
(
  'pageview',
  '1',
  '{ "page": "/products" }',
  '{ "name": "Safari", "os":
"Mac", "resolution": { "x": 1920,
```

FULLTEXT index

```sql
CREATE TABLE articles (
        id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
        title VARCHAR(200),
        body TEXT,
        FULLTEXT (title,body)
    );
```

```sql
SELECT * FROM articles WHERE MATCH (title,body)
AGAINST ('Терноп*' IN BOOLEAN MODE);
```

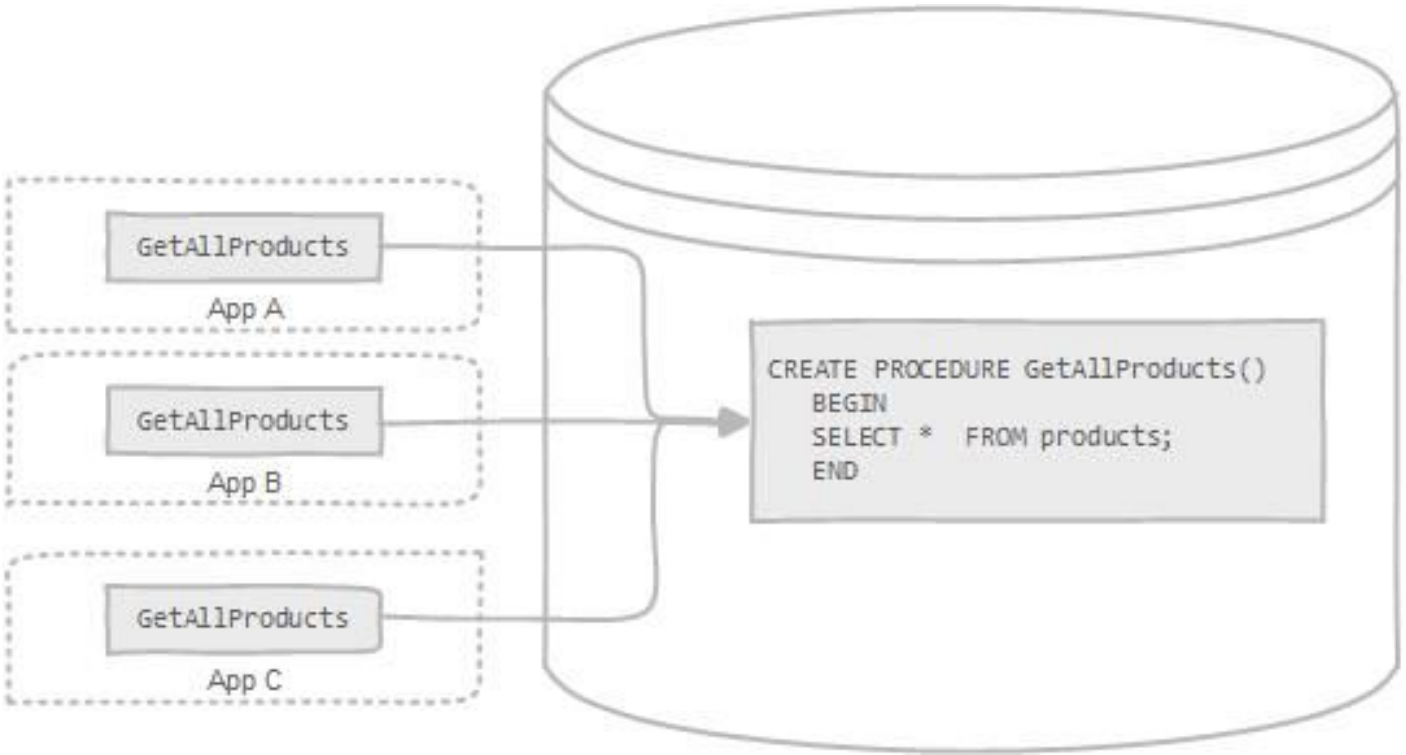
Transactions



Transaction sample

```sql
-- 1. start a new transaction
START TRANSACTION;

-- 2. Get the latest order number
SELECT
    @orderNumber:=MAX(orderNUmber)+1
FROM
    orders;

-- 3. insert a new order for customer 145
INSERT INTO orders(orderNumber,
                   orderDate,
                   requiredDate,
                   shippedDate,
                   status,
                   customerNumber)
VALUES(@orderNumber,
       '2005-05-31',
       '2005-06-10',
       '2005-06-11',
       'In Process',
        145);

-- 4. Insert order line items
INSERT INTO orderdetails(orderNumber,
                         productCode,
                         quantityOrdered,
                         priceEach,
                         orderLineNumber)
VALUES(@orderNumber,'S18_1749', 30, '136', 1),
      (@orderNumber,'S18_2248', 50, '55.09', 2);

-- 5. commit changes
COMMIT;
```

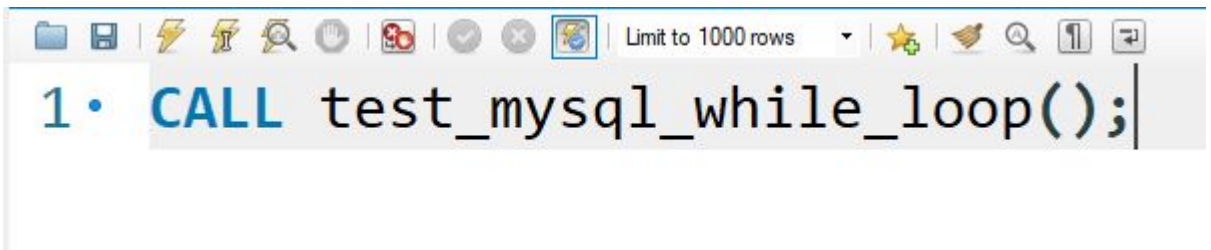
Stored procedures




```
1  DELIMITER //
2  CREATE PROCEDURE GetAllProducts()
3    BEGIN
4    SELECT *  FROM products;
5    END //
6  DELIMITER ;
```

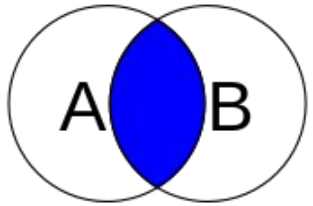
Stored procedure sample

```
1  DELIMITER $$
2  DROP PROCEDURE IF EXISTS test_mysql_while_loop$$
3  CREATE PROCEDURE test_mysql_while_loop()
4  BEGIN
5  DECLARE x  INT;
6  DECLARE str  VARCHAR(255);
7
8  SET x = 1;
9  SET str =  '';
10
11 WHILE x  <= 5 DO
12 SET  str = CONCAT(str,x,',');
13 SET  x = x + 1;
14 END WHILE;
15
16 SELECT str;
17 END$$
18 DELIMITER ;
```

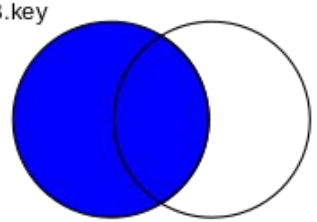


SQL JOINS

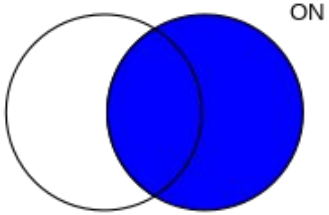
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key



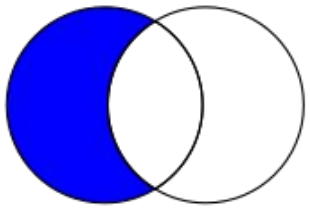
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key



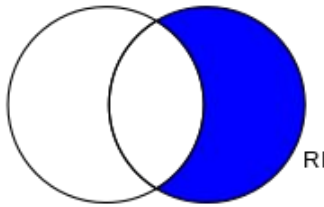
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key



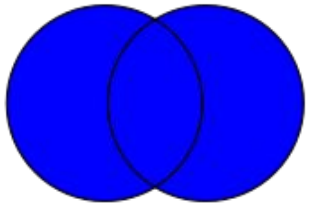
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL



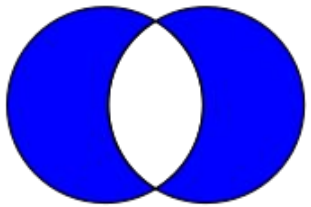
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE a.key IS NULL



SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key

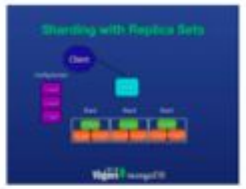


SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key iIS NULL



Faster process

Open Source



Sharding

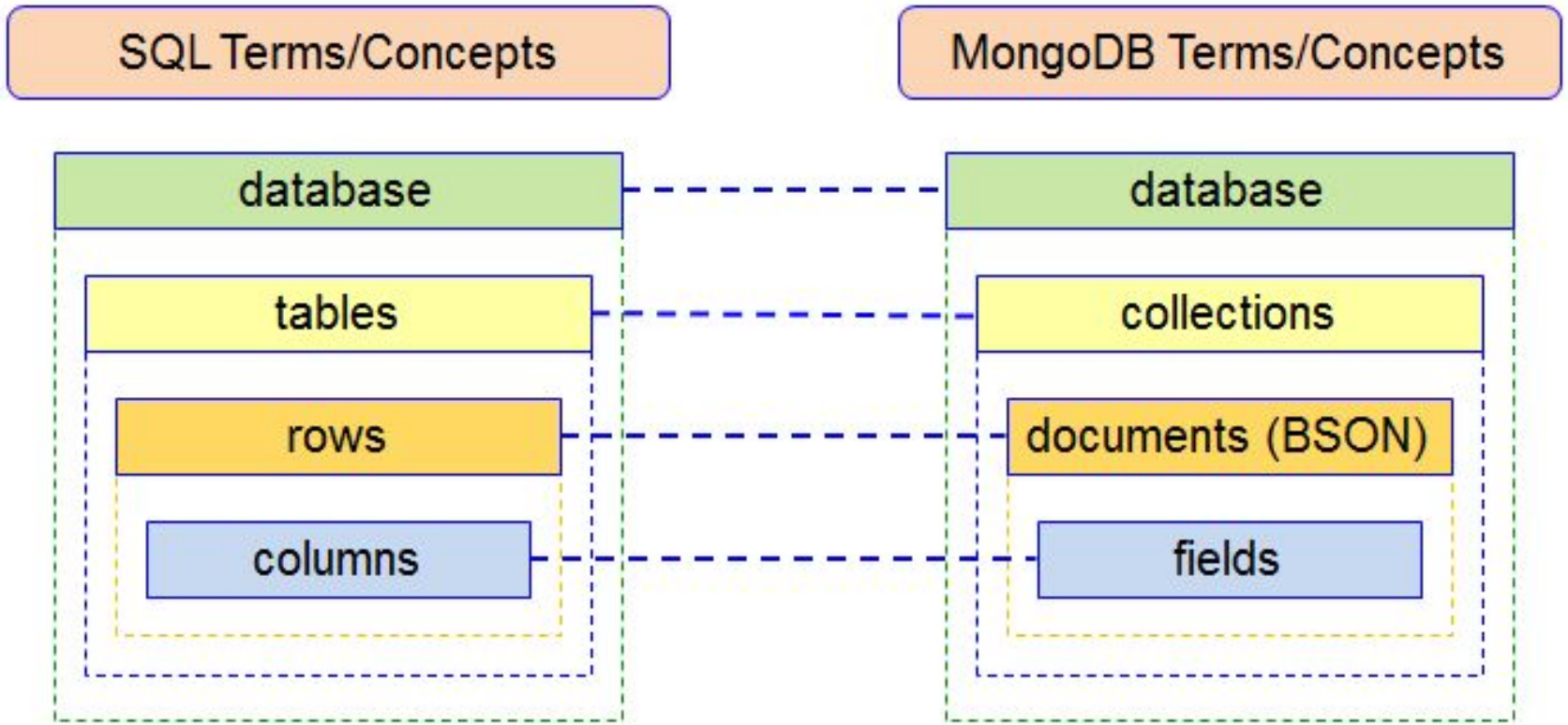
mongoDB

Schemaless

Document based

No SQL Injection

Terms


SQL Terms/Concepts
MongoDB Terms/Concepts
database
tables
rows
columns
database
collections
documents (BSON)
fields

Document & collection

Collection

Document

```
db.users.insert(
                {
                    name: "sue",
                     age: 26,
                  status: "A",
                  groups: [ "news", "sports" ]
                }
              )
```

Document

```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

insert

Collection

```
{ name: "al", age: 18, ... }
{ name: "lee", age: 28, ... }
{ name: "jan", age: 21, ... }
{ name: "kai", age: 38, ... }
{ name: "sam", age: 18, ... }
{ name: "mel", age: 38, ... }
{ name: "ryan", age: 31, ... }
{ name: "sue", age: 26, ... }
```

users

RDBMS & Mongo

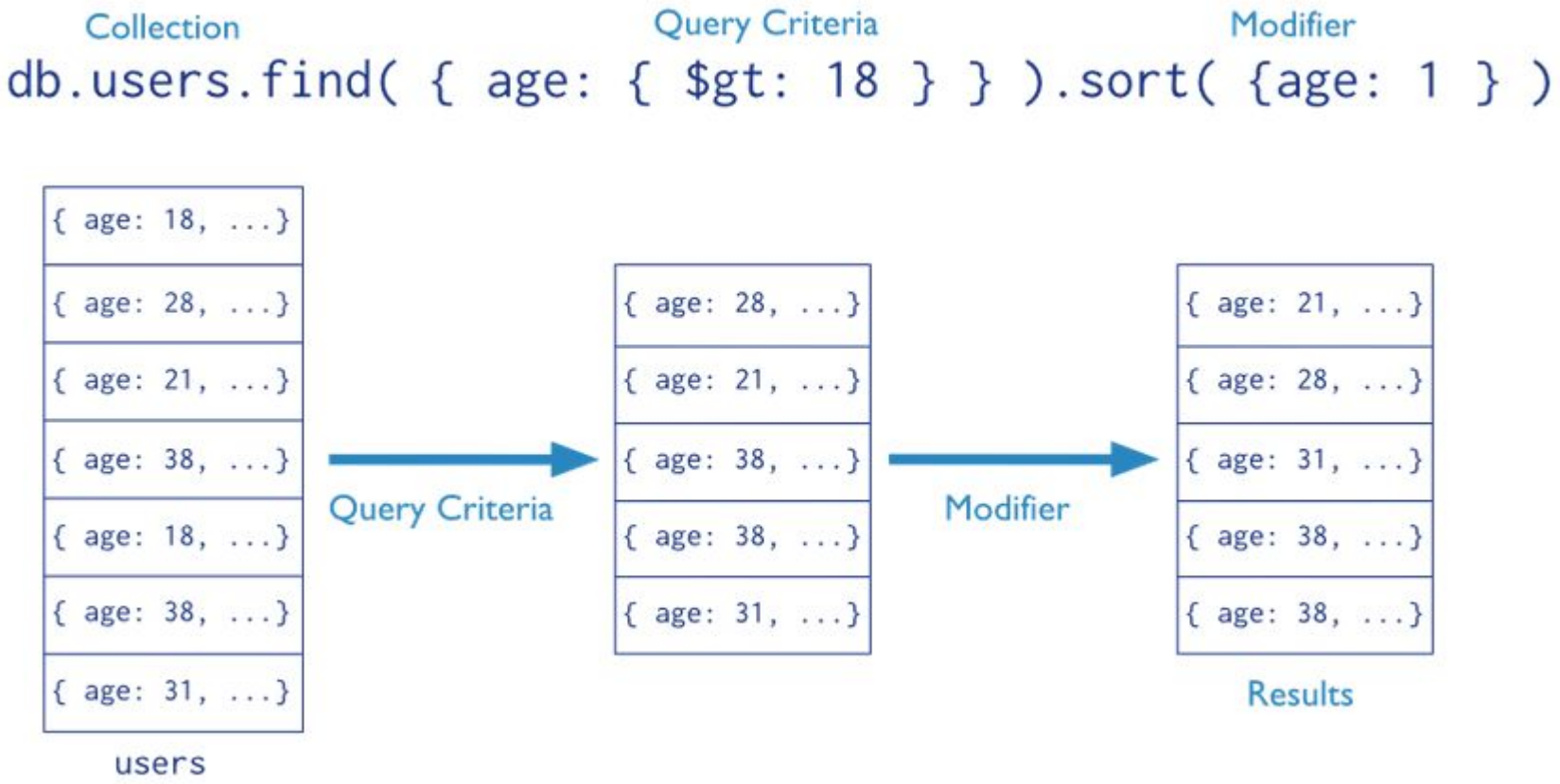
Relational Database

Student_Id	Student_Name	Age	College
1001	Chaitanya	30	Beginnersbook
1002	Steve	29	Beginnersbook
1003	Negan	28	Beginnersbook

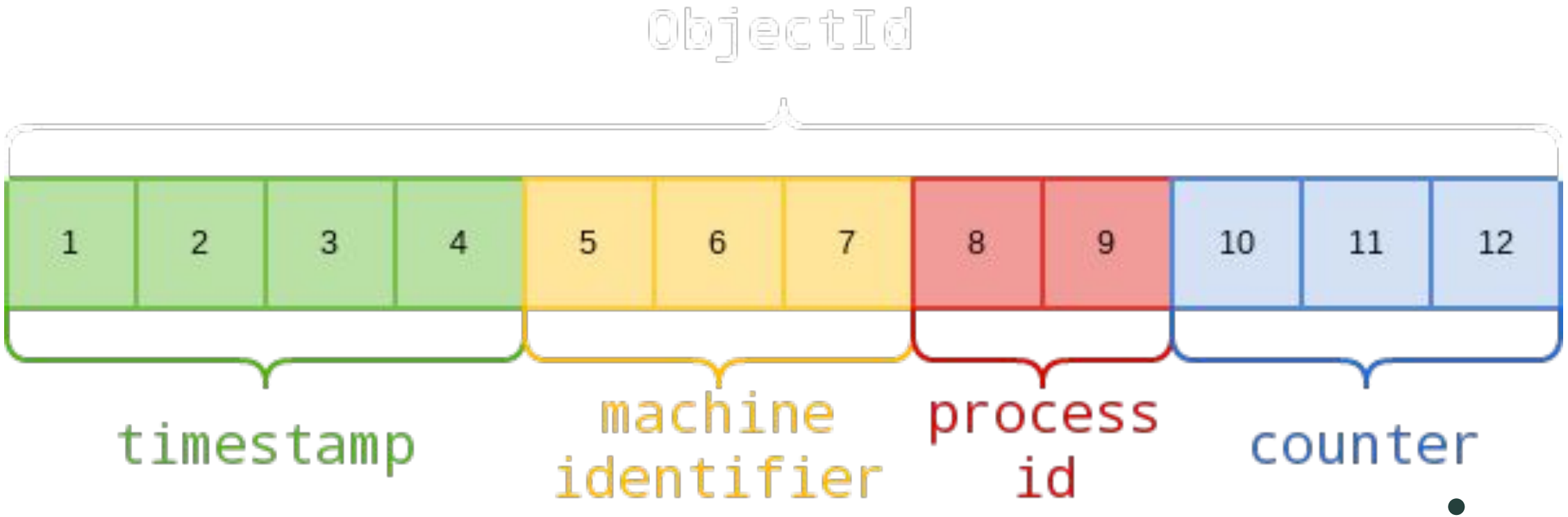
MongoDB

```
{
 "_id": ObjectId("....."),
 "Student_Id": 1001,
 "Student_Name": "Chaitanya",
 "Age": 30,
 "College": "Beginnersbook"
}
{
 "_id": ObjectId("....."),
 "Student_Id": 1002,
 "Student_Name": "Steve",
 "Age": 29,
 "College": "Beginnersbook"
}
{
 "_id": ObjectId("....."),
 "Student_Id": 1003,
 "Student_Name": "Negan",
 "Age": 28,
 "College": "Beginnersbook"
}
```

Sample query



_id


ObjectId
1
2
3
4
5
6
7
8
9
10
11
12
timestamp
machine identifier
process id
counter

Thank you!