Database

Relational database vs. Non-relational database

Relational database - using stuctured query language to access data

Non-relational - stores data as document

- data can be accessed through mongoDB’s query language

(close to JS)

Relational Model

Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack Sealing

Key = 24

Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

Date	Activity Code	Route No.
01/12/01	24	I-95
01/15/01	23	I-495
02/08/01	24	I-66

table - composed of fields and records that hold data

record - a record is composed of fields and contains all the data about one item

- record appear as row

field - field is part of the record and contains a single piece of data for the subject of record

- field appear as a column


{
_id: <ObjectId1>,
username: "123xyz",
contact: {
phone: "123-456-7890",
email: "xyz@example.com"
},
access: {
level: 5,
group: "dev"
}
}
Embedded sub-document
Embedded sub-document

- In MongoDB we store documents in collections
- The collection holds documents in JSON format
- this data model maps naturally to objects in application code, making it simple for developers to learn and use
- documents give you to the ability to represent hierarchical relationships

Create Read Update Delete

Create:	Read:	Update:	Delete:
mySQL: CREATE / INSERT (table) VALUE () mongoDB: db.createCollection()/ object.save()	mySQL: SELECT * FROM (table) mongoDB: collection.find()	mySQL: SET () WHERE () mongoDB: collection.update()	MySQL: DELETE () WHERE () mongoDB: collection.remove()

ORM

- object relational mapping
- an ORM library is a completely ordinary library written in your language of choice
- that encapsulates the code needed to manipulate the data
- you do not need to use SQL language anymore
- you interact directly with an object in the same language you are using

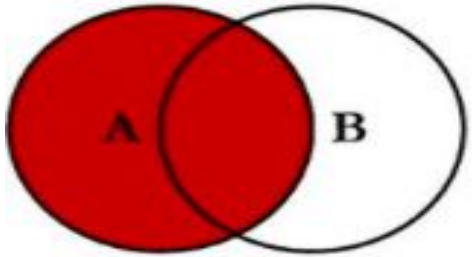
SQL syntax

-structured query language

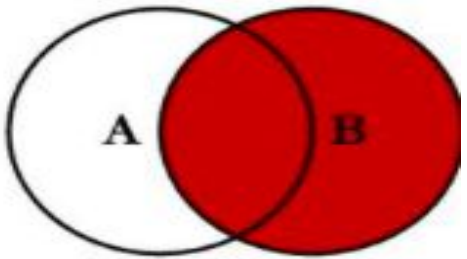
- all SQL statements start with : SELECT / INSERT / UPDATE / DELETE / ALTER / DROP / CREATE / USE / SHOW and ends with **;**

- case insensitive and works with tables

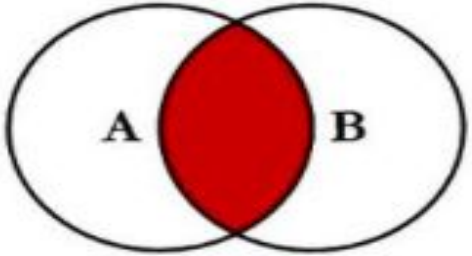
SQL JOINS



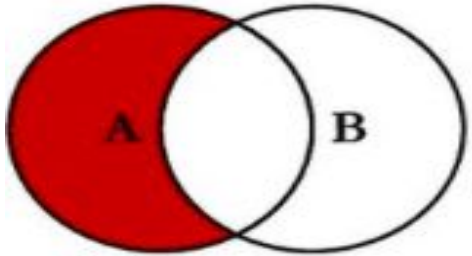

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



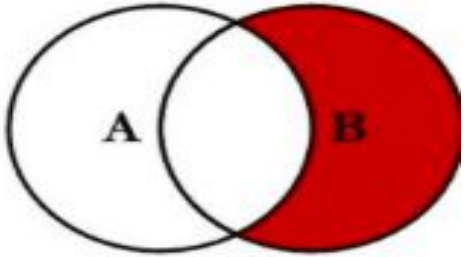

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



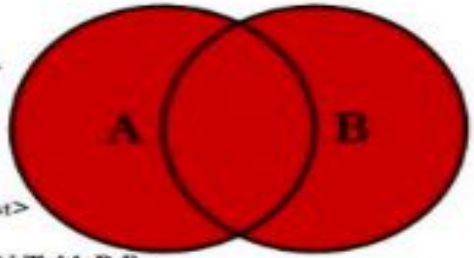

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



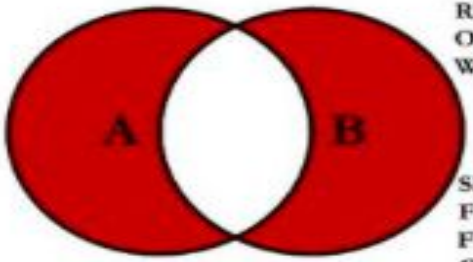

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Joined models and foreign keys

- we join tables based on the related data (same data exist in the other table)
- foreign keys point to the primary key of the table

```
1  select order_date, order_amount
2  from customers
3  join orders
4     on customers.customer_id = orders.customer_id
5  where customer_id = 3
```