Database

hyuk

Database

- 여러 사람이 공유하여 사용할 목적으로 체계화해 통합, 관리하는 데이터의 집합이다.
- 중복된 데이터를 없애고, 자료를 구조화하여, 효율적인 처리를 할 수 있도록 관리함
- 데이터베이스는 응용 프로그램과는 다른 별도의 미들웨어에 의해 관리됨
- DBMS(Database Management System) : DB를 관리하는 미들웨어

Database 특징

1. 사용자의 질의에 대하여 즉각적인 처리와 응답이 이루어집니다.
2. 생성, 수정, 삭제를 통하여 항상 최신의 데이터를 유지합니다.
3. 사용자들이 원하는 데이터를 동시에 공유할 수 있습니다.
4. 사용자가 원하는 데이터를 주소가 아닌 내용에 따라 참조 할 수 있습니다.
5. 응용프로그램과 데이터베이스는 독립되어 있으므로, 데이터의 논리적 구조와 응용프로그램은 별개로 동작됩니다.

Database Model

- 데이터 베이스가 어떻게 구조화되고 어떻게 쓰이는지 알려주는 모델
- 이거에 따라서 크게 3개의 세대로 나뉨
- 내비게이셔널 데이터베이스, SQL/관계형 데이터 베이스, 관계형 이후 데이터베이스
- 주가 되는 2개의 내비게이셔널 데이터 모델은 IBM의 IMS 시스템의 전형적 본보기가 되는 계층형 모델, 그리고 IDMS와 같은 수많은 제품들에 구현된 CODASYL 모델 (네트워크 모델)이다.

SQL : Structured Query Language

- 구조화된 쿼리 언어 => 관계형 데이터베이스 라는 의미로 쓰임

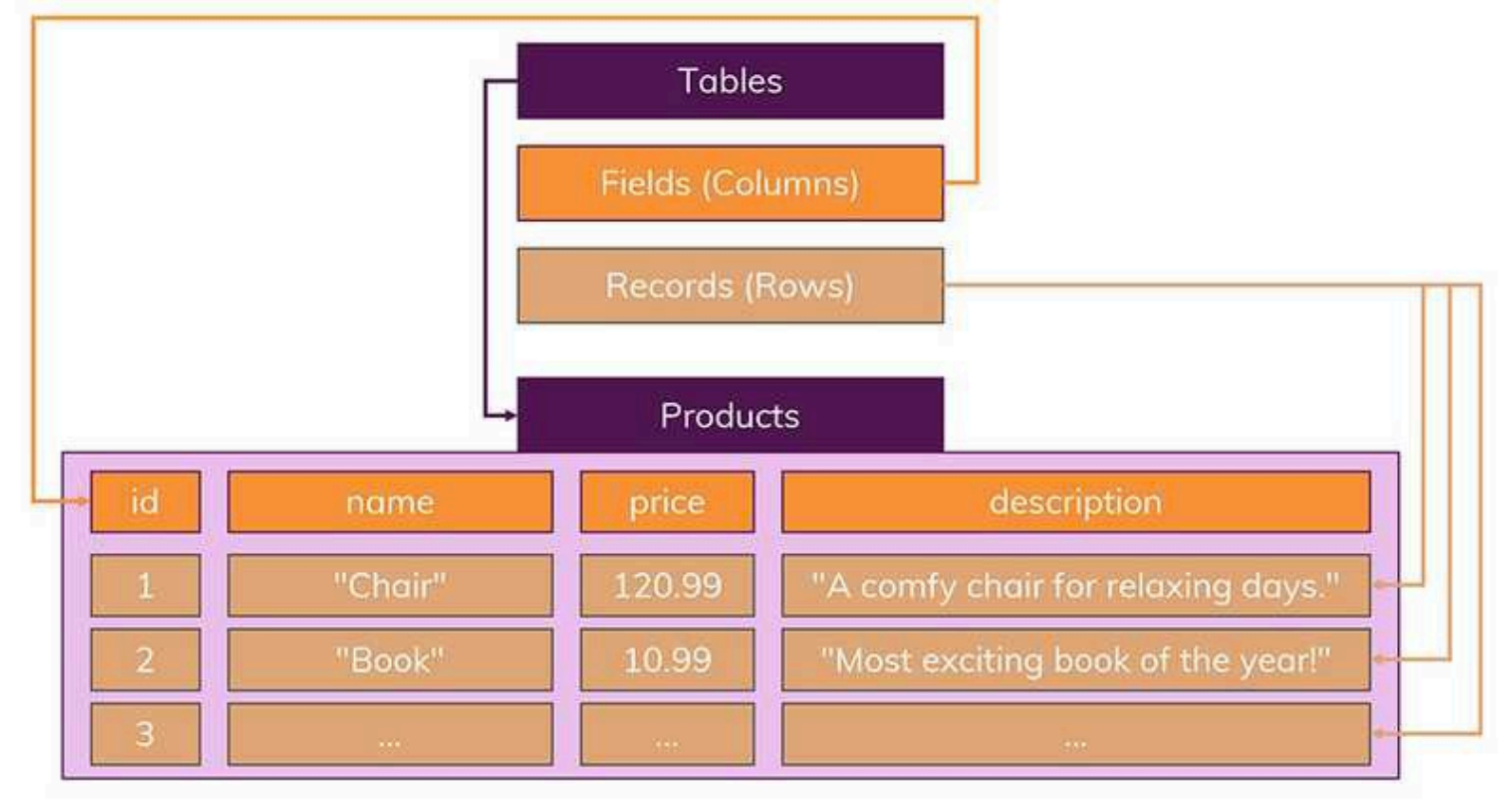
- 관계형 데이터베이스
  - 데이터는 정해진(엄격한) 데이터 스키마 (= structure)를 따라 데이터베이스 테이블에 저장
  - 데이터는 관계를 통해서 연결된 여러개의 테이블에 분산됩니다.

Ex) mySQL

SQL

- 엄격한 스키마

데이터는 table에 record로 저장되고, 각 테이블에는 명확하게 정의된 structure가 있다



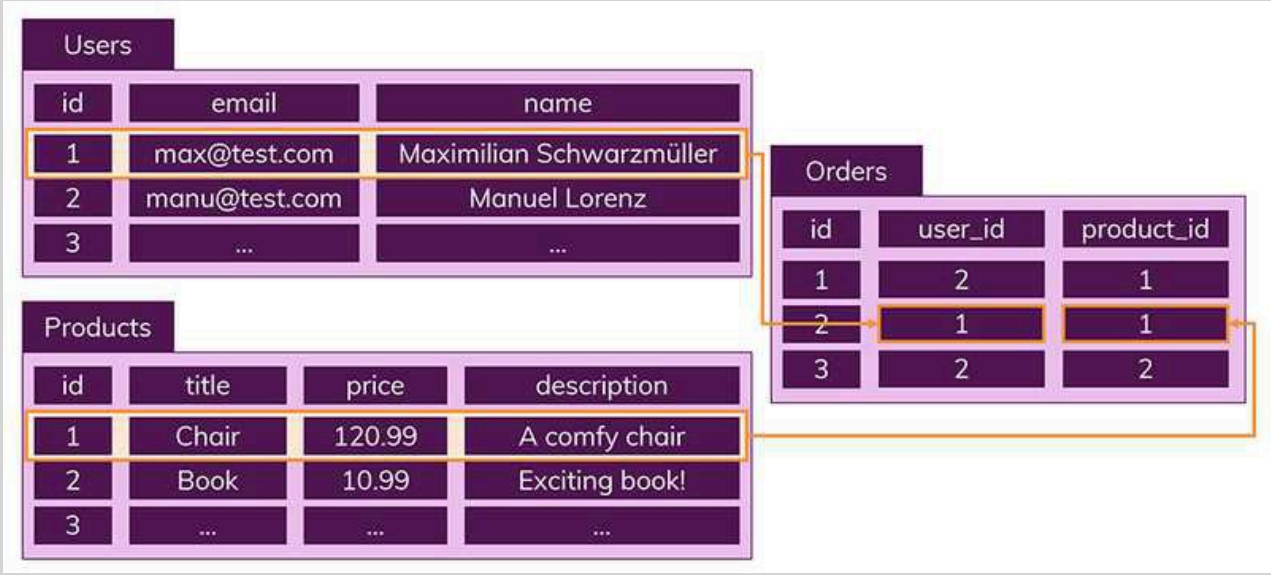
SQL : Structured Query Language

- 관계

데이터들을 여러개의 테이블에 나누어서, 데이터들의 중복을 피할 수 있다

만약 사용자가 구입한 상품들을 나타내기 위해서는, Users(사용자), Products(상품), Orders(주문한 상품) 여러 테이블을 만들어야 하지만, 각각의 테이블들은 다른 테이블에 저장되지 않은 데이터 만을 가지고 있다. (중복된 데이터없음)

장점 : 하나의 테이블에서 중복없이 하나의 데이터만을 관리하기 때문에, 다른 테이블에서 부정확한 데이터를 다룰 위험이 없습니다.



NoSQL (non SQL)

- 스키마 없음
- 관계 없음

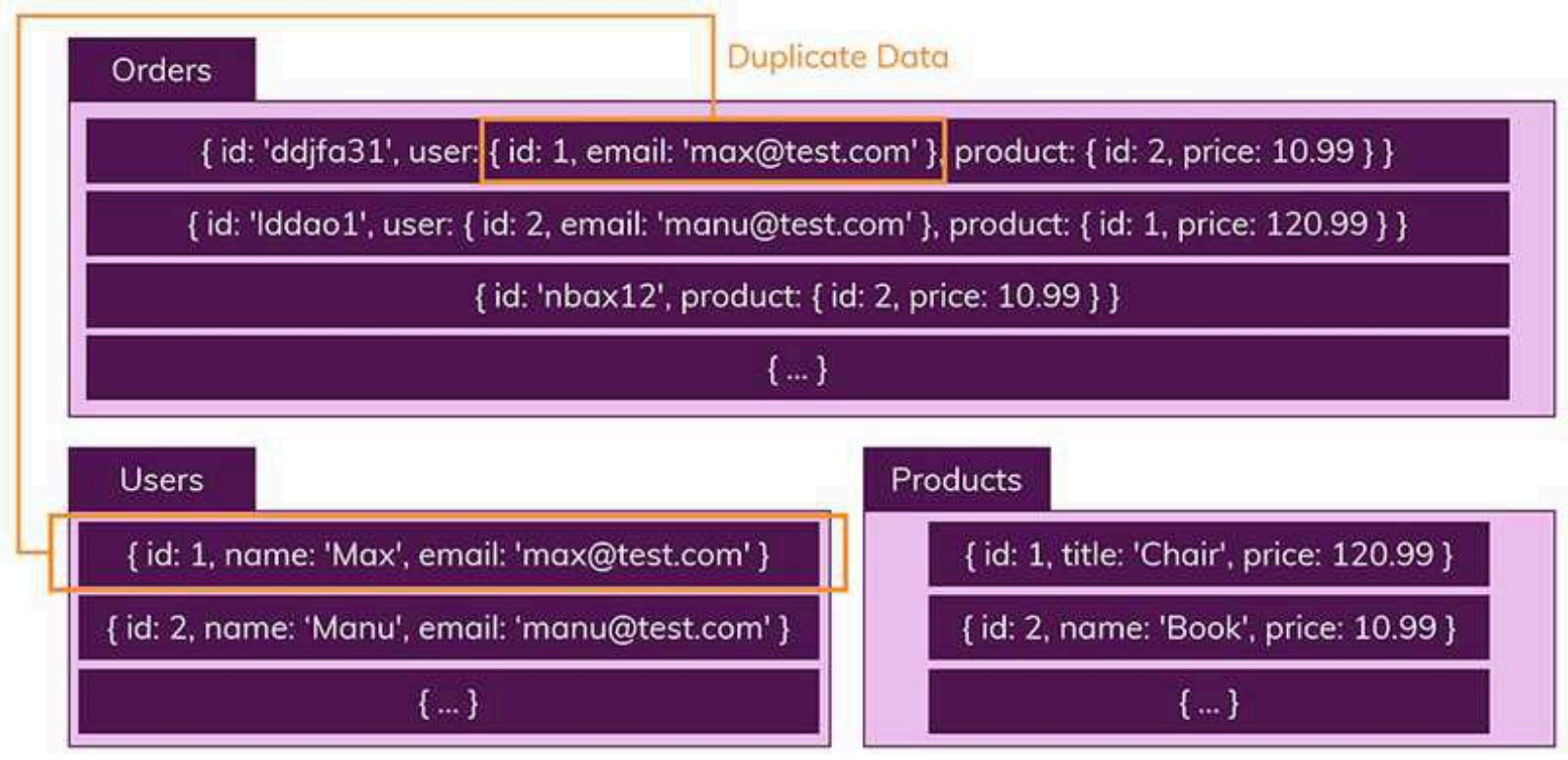
- 어떻게 저장? => 보통 documents 형식(보통 Json)

Ex) mongoDB

NoSQL

- SQL과 다르게 같은 collection에 다른 구조의 데이터 저장 가능

NoSQL



단점 : 정보가 바뀌면 전부 바꿔주어야 한다
장점 : order만 봐도 다 알 수 있다 => 자주 수정 x일수록 장점이 커진다

Scaling (확장)

- 데이터베이스의 서버의 확장성

- 수직적 확장 : DB 서버의 성능 확장 (ex CPU업그레이드)
    - SQL에서 일반적으로 수직적 확장만 지원

- 수평적 확장 : 서버가 추가되고 DB가 분산됨
    - NoSQL데이터 에서만 가능

장단점

	SQL	NoSQL
장점	✓ 명확하게 정의된 스키마 ✓ 데이터의 무결성 ✓ 중복 없는 데이터	✓ 훨씬 유연하다 ✓ 애플리케이션이 원하는 방식으로 데이터 저장 => 속도가 훨씬 빠르다 ✓ 수직 및 **수평확장**이 가능하므로 모든 요청 처리 가능
단점	✓ 데이터 스키마는 사전에 계획 (수정 힘듬) ✓ JOIN문이 많은 매우 복잡한 쿼리가 생길 수 있음 ✓ 수평적 확장이 어렵고, 성장 한계	✓ 수정 해야 하는 경우 모든 컬렉션에서 수정해야함

결론

- SQL
  - 관계를 갖고 있는 데이터가 자주 변경 되는 경우
  - 변경될 여지가 없고 명확한 스키마가 중요한 경우

- NoSQL
  - 정확한 데이터 구조를 알 수 없거나 변경/확장 될 수 있는 경우
  - 데이터를 자주 읽지만 수정을 하지 않는 경우
  - 막대한 양의 데이터를 다뤄야 하는 경우

상황에 맞게 잘 쓰면 된다!!!!

실습

- mongoDB
- 설치
- https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/

- $ wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add –
- $ echo "deb [arch=amd64,arm64] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list
- $ sudo apt-get update
- $ sudo apt-get install -y mongodb-org

실습

- 실행
  - $ mongo
- Db생성
  - $ use [name]
  - wheelSeminar 사용
- Db 리스트 확인
  - $ show dbs
  - 없다 => 하나 이상의 collection이 있어야 함

```
> use wheelSeminar
switched to db wheelSeminar
> show dbs
admin     0.000GB
conduit   0.000GB
config    0.000GB
local     0.000GB
>
```

실습

- Collection 생성
  - db.createCollection([name])
- Collection list 확인
  - show collections
- 다시 db list 확인

```
> db.createCollection("users")
{ "ok" : 1 }
> show collections
users
> show dbs
admin          0.000GB
conduit        0.000GB
config         0.000GB
local          0.000GB
wheelSeminar   0.000GB
```

실습

- document 추가
- $ db.[collection_name].insert([document])
- [document]는 JSON형식으로

```
> db.users.insert([
... {"name":"kjy","nickname":"panya"},
... {"name":"cjh","nickname":"ball"}
... ])
```

```
> db.users.insert({"name":"jsh","nickname":"hyuk"})
WriteResult({ "nInserted" : 1 })
```

실습

- document 조회
  - $ db.[collection_name].find([document])
  - .find()이면 전부 보여줌
  - .pretty()는 예쁘게 보여줌

```
> db.users.find()
{ "_id" : ObjectId("5f17daee8e411cc81102492b"), "name" : "jsh", "nickname" : "hyuk" }
{ "_id" : ObjectId("5f17dc348e411cc81102492c"), "name" : "kjy", "nickname" : "panya" }
{ "_id" : ObjectId("5f17dc348e411cc81102492d"), "name" : "cjh", "nickname" : "ball" }
> db.users.find().pretty()
{
        "_id" : ObjectId("5f17daee8e411cc81102492b"),
        "name" : "jsh",
        "nickname" : "hyuk"
}
{
        "_id" : ObjectId("5f17dc348e411cc81102492c"),
        "name" : "kjy",
        "nickname" : "panya"
}
{
        "_id" : ObjectId("5f17dc348e411cc81102492d"),
        "name" : "cjh",
        "nickname" : "ball"
}
> db.users.find({"name":"jsh"}).pretty()
{
        "_id" : ObjectId("5f17daee8e411cc81102492b"),
        "name" : "jsh",
        "nickname" : "hyuk"
}
```

실습

- .find({ ”age": { $gt: 20, $lt: 22 } })
- .find({ $and: [{ “name”: “jsh” }, { “gender”: “M” }] })

operator	설명
$eq	(equals) 주어진 값과 일치하는 값
$gt	(greater than) 주어진 값보다 큰 값
$gte	(greather than or equals) 주어진 값보다 크거나 같은 값
$lt	(less than) 주어진 값보다 작은 값
$lte	(less than or equals) 주어진 값보다 작거나 같은 값
$ne	(not equal) 주어진 값과 일치하지 않는 값
$in	주어진 배열 안에 속하는 값
$nin	주어빈 배열 안에 속하지 않는 값

operator	설명
$or	주어진 조건중 하나라도 true 일 때 true
$and	주어진 모든 조건이 true 일 때 true
$not	주어진 조건이 false 일 때 true
$nor	주어진 모든 조건이 false 일때 true

실습

- Update
- .update([누구],{$set:[뭘]})
- 추가도 가능

```
> db.users.update({"nickname":"hyuk"},{$set:{"name":"csh"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.users.find().pretty()
{
        "_id" : ObjectId("5f17daee8e411cc81102492b"),
        "name" : "csh",
        "nickname" : "hyuk"
}
{
        "_id" : ObjectId("5f17dc348e411cc81102492c"),
        "name" : "kjy",
        "nickname" : "panya"
}
{
        "_id" : ObjectId("5f17dc348e411cc81102492d"),
        "name" : "cjh",
        "nickname" : "ball"
}
```

```
> db.users.update({"nickname":"hyuk"},{$set:{"gender":"M"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.users.find().pretty()
{
        "_id" : ObjectId("5f17daee8e411cc81102492b"),
        "name" : "csh",
        "nickname" : "hyuk",
        "gender" : "M"
}
```

실습

- 삭제
- db.[collection_name].remove([criteria])

```
> db.users.find().pretty()
{
        "_id" : ObjectId("5f17daee8e411cc81102492b"),
        "name" : "csh",
        "nickname" : "hyuk",
        "gender" : "M"
}
{
        "_id" : ObjectId("5f17dc348e411cc81102492c"),
        "name" : "kjy",
        "nickname" : "panya",
        "gender" : "F"
}
{
        "_id" : ObjectId("5f17dc348e411cc81102492d"),
        "name" : "cjh",
        "nickname" : "ball",
        "gender" : "M"
}
> db.users.remove({"gender":"M"})
WriteResult({ "nRemoved" : 2 })
> db.users.find().pretty()
{
        "_id" : ObjectId("5f17dc348e411cc81102492c"),
        "name" : "kjy",
        "nickname" : "panya",
        "gender" : "F"
}
> db.users.remove({})
WriteResult({ "nRemoved" : 1 })
> db.users.find()
```

실습

- Collection 삭제
  - db.[collection_name].drop()
- DB 삭제
  - db.dropDatabase()

```
> db.createCollection("hi")
{ "ok" : 1 }
> show collections
hi
users
> db.hi.drop()
true
> show collections
users
```

```
> db.dropDatabase()
{ "dropped" : "wheelSeminar", "ok" : 1 }
> show dbs
admin    0.000GB
conduit  0.000GB
config   0.000GB
local    0.000GB
```

제출

- 생성,수정,삭제 한 부분 사진

감사합니다