

Hello,

KDT 웹 개발자 양성 프로젝트

1기! 44th

with





App.js

분기 처리!



```
function App() {  
  const page = useSelector((state) => state.mbti.page);  
  const survey = useSelector((state) => state.mbti.survey);  
  
  return (  
    <>  
      <GlobalStyle />  
      <Main>  
        {page === 0 ? (  
          <Start />  
        ) : page !== survey.length + 1 ? (  
          <Mbti />  
        ) : (  
          <Show />  
        )}  
      </Main>  
    </>  
  );  
}
```

Src/App.js





SQL

(Structured Query Language)



https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

SQL Statement:

```
SELECT * FROM Customers;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Click "**Run SQL**" to execute the SQL statement above.

W3Schools has created an SQL database in your browser.

The menu to the right displays the database, and will reflect any changes.

Feel free to experiment with any SQL statement.

You can restore the database at any time.





SELECT



SELECT, DB 에서 원하는 데이터 가져오기

- 갓춰진 DB 에서 데이터를 뽑아 올 때 사용하는 구문 입니다!
- 일단
- `SELECT * FROM Customer;`



원하는 컬럼의 값만 가지고 올 때?

- 모든 컬럼 값이 필요 한게 아니라면!?
- `SELECT 컬럼명 FROM table`
- `SELECT City, Country FROM Customers`

DB에는 영향 없이 원하는 데이터 추가 할 때!



- 필요한 값을 임의로 추가해서 가져오고 싶을 땐?
- `SELECT` 컬럼명, 원하는 데이터 `FROM` table
- `SELECT` City, 1, '원하는 문자열', NULL `FROM` Customers

원하는 조건을 충족하는 Row 만 가져 올 때!



- 원하는 조건을 충족 하는 값을 찾고 싶을 땐? **WHERE**
- **SELECT * FROM table WHERE 조건**
- **SELECT * FROM OrderDetails WHERE Quantity > 5;**



Row 를 정렬해서 가져 올 때!

- 선택한 값들을 정렬해서 가지고 올 때는? ORDER BY
- SELECT * FROM Customers ORDER BY ContactName;
- SELECT * FROM Customers ORDER BY ContactName DESC;
- SELECT * FROM Customers ORDER BY CustomerName ASC,
ContactName DESC;



Row 의 개수를 지정 하고 싶을 때!

- 가지고 오는 ROW의 수를 지정하고 싶을 때? **LIMIT**
- **SELECT * FROM Customers LIMIT 10;**
- **SELECT * FROM Customers LIMIT 30, 10;**
 - 원하는 Row 순번, 자르는 개수

Column 명을 변경해서 가지고 오고 싶을 때?



- Column 명을 변경해서 가지고 올 때? **AS**
- **SELECT** CustomerId **AS** id, **CustomerName AS name** **FROM**
Customers;



테이블 합치기!

JOIN

여러 테이블의 값을 하나로 합치고 싶다면!?



- 여러 테이블을 하나로 붙이고 싶을 땐? JOIN
- `SELECT * FROM Products P JOIN Categories C ON P.CategoryID = P.CategoryID;`

SQL Statement:

SELECT * FROM Products P JOIN Categories C ON P.CategoryID = P.CategoryID;

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 616

ProductID	ProductName	SupplierID	CategoryID	Unit	Price	CategoryName	Description
1	Chais	1	1	10 boxes x 20 bags	18	Beverages	Soft drinks, coffees, teas, beers, and ales
1	Chais	1	2	10 boxes x 20 bags	18	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
1	Chais	1	3	10 boxes x 20 bags	18	Confections	Desserts, candies, and sweet breads
1	Chais	1	4	10 boxes x 20 bags	18	Dairy Products	Cheeses
1	Chais	1	5	10 boxes x 20 bags	18	Grains/Cereals	Breads, crackers, pasta, and cereal
1	Chais	1	6	10 boxes x 20 bags	18	Meat/Poultry	Prepared meats
1	Chais	1	7	10 boxes x 20 bags	18	Produce	Dried fruit and bean curd
1	Chais	1	8	10 boxes x 20 bags	18	Seafood	Seaweed and fish
2	Chang	1	1	24 - 12 oz bottles	19	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Chang	1	2	24 - 12 oz bottles	18	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings





Local 에서
연습 시작!

이제 Local 에 Mysql DB 를 구축해 봅시다!



- `mysql -u root -p`
- 설정한 비밀 번호를 치고 들어가기

```
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 53  
Server version: 8.0.28 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2022, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> _
```



MySQL Workbench



수업을 위한 DB 만들기

DB 생성!



- `CREATE SCHEMA `mydb` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;`

A screenshot of a SQL query editor window. The title bar says "Query 1". Below the title bar is a toolbar with various icons for file operations, execution, and search. The main area shows a single line of SQL code: `CREATE SCHEMA `mydb` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;`. The line number "1" is visible on the left.

```
Query 1 x
1 CREATE SCHEMA `mydb` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
```

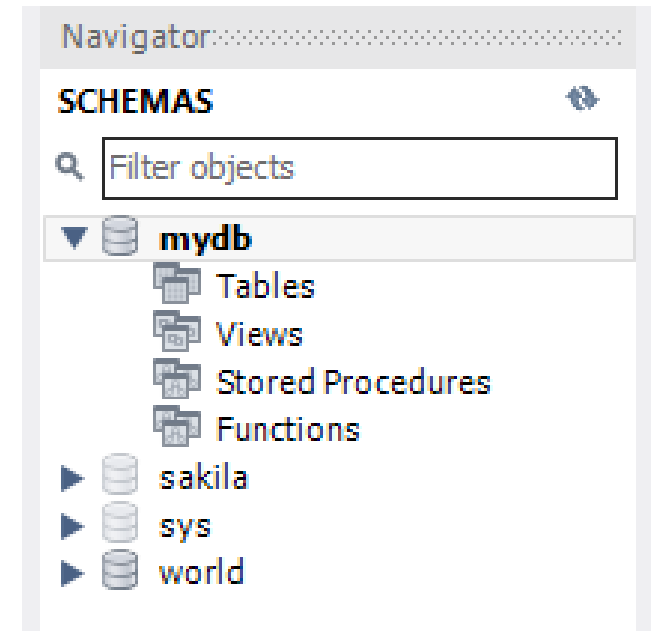


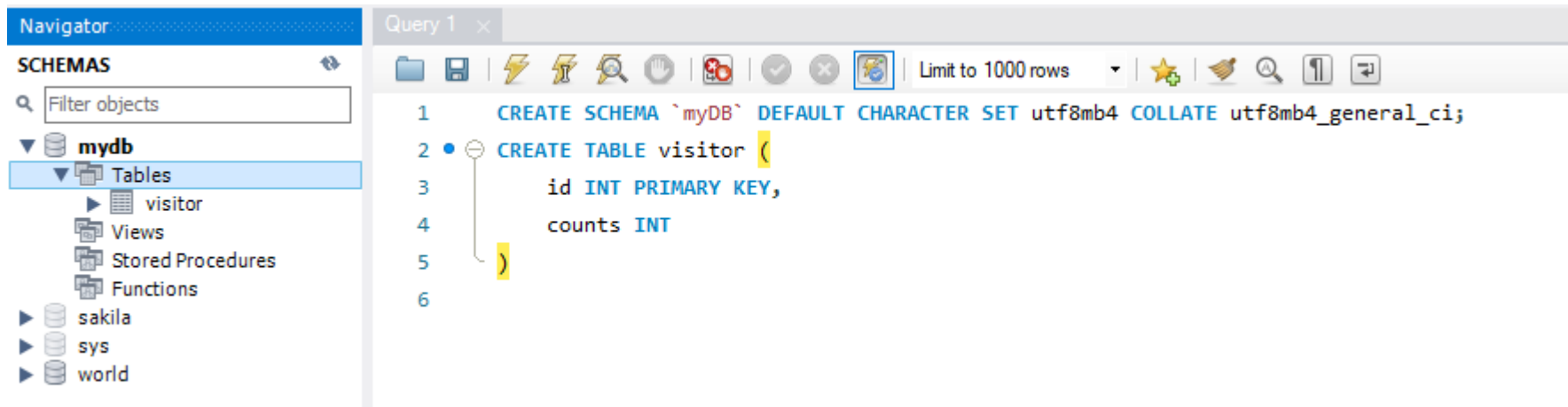


TABLE 생성!

DB 의 TABLE 생성!



- CREATE TABLE visitor (id INT PRIMARY KEY, counts INT);





DATA 삽입하기



생성한 TABLE 에 DATA 삽입!

- **INSERT INTO** visitor (id, counts) **VALUES** (1, 0);

The screenshot shows a database query editor window titled "Query 1". The query text is as follows:

```
1 • SELECT * from mydb.visitor;  
2 • insert into visitor (id, counts) VALUES (1, 0);
```

Below the query editor, the "Result Grid" is displayed, showing the results of the executed query. The grid has two columns: "id" and "counts". The first row contains the values "1" and "0". The second row contains "NULL" and "NULL".

	id	counts
▶	1	0
*	NULL	NULL



DATA 삭제

DATA 삭제!





- DELETE FROM visitor WHERE id = 2;

3 • DELETE FROM visitor WHERE id = 2;

<

Result Grid

Filter Rows:

Edit:  

	id	counts
▶	1	5
✱	NULL	NULL



DATA 수정



DATA 수정(Update)!



- UPDATE visitor SET counts = counts + 1 WHERE id = 1;

3 • DELETE FROM visitor WHERE id = 2;

<

Result Grid

Filter Rows:

Edit:  

	id	counts
▶	1	5
✱	NULL	NULL



Table 수정 및

삭제하기



ALTER TABLE - 테이블 변경

```
-- 테이블명 변경
ALTER TABLE people RENAME TO friends,
-- 컬럼 자료형 변경
CHANGE COLUMN person_id person_id TINYINT,
-- 컬럼명 변경
CHANGE COLUMN person_name person_nickname VARCHAR(10),
-- 컬럼 삭제
DROP COLUMN birthday,
-- 컬럼 추가
ADD COLUMN is_married TINYINT AFTER age;
```

DROP TABLE - 테이블 삭제

```
DROP TABLE friends;
```



DB 통신용 서버 구축하기



Express 서버 코드 작성!

```
const express = require('express');
const cors = require('cors');
const PORT = 4000;

const app = express();

app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cors());

const dataRouter = require('./routes/data');
app.use('/data', dataRouter);

app.listen(PORT, () => {
  console.log(`데이터 통신 서버가 ${PORT}에서 작동 중`);
});
```

Data-server/server.js



MySQL 서버 통신용 모듈 작성

- dbConnect.js 라는 DB 통신용 모듈을 작성해 봅시다!

```
const mysql = require('mysql');

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '1234',
  port: '3306',
  database: 'mydb',
});
```

```
connection.connect();
```

```
module.exports = connection;
```

Data-server/dbConnect.js



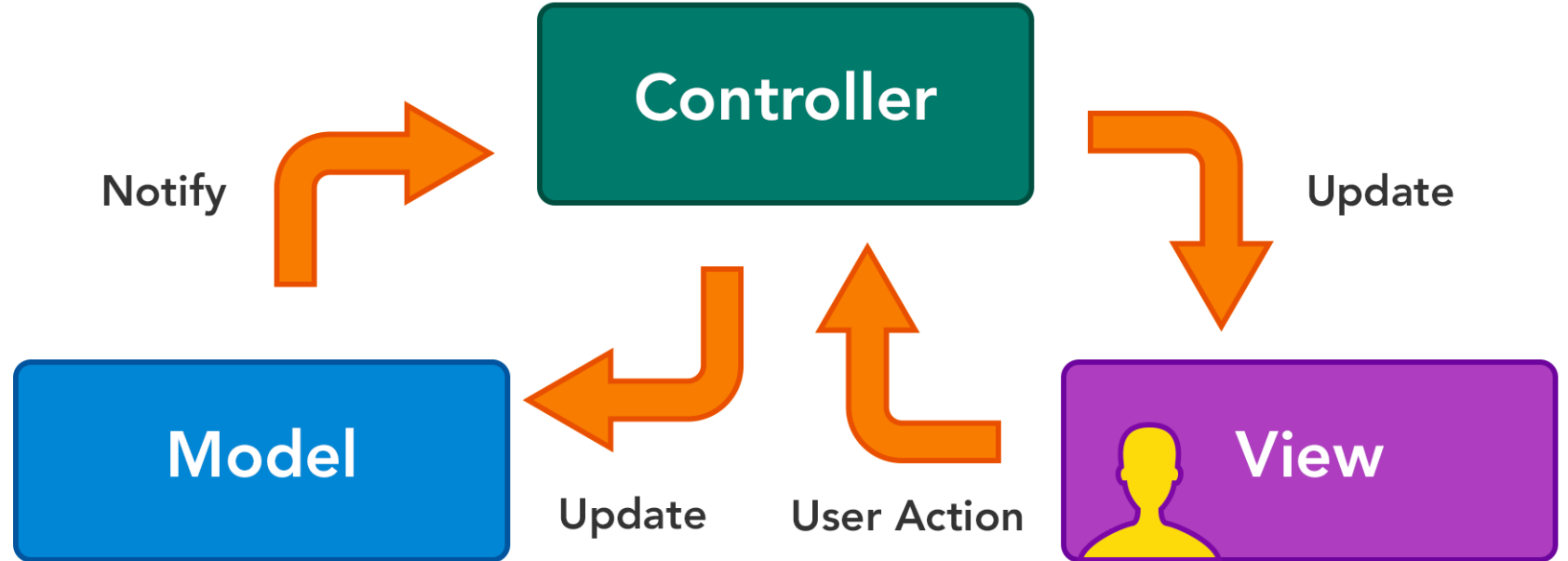
DB 통신용

컨트롤러 작성



DB 통신 컨트롤러 작성

- DB와 통신을 해서 우리가 원하는 데이터를 만들어 주는 Controller 를 작성해 봅시다





```
const connection = require('../dbConnect');

const db = {
  getCounts: (cb) => {
    connection.query('SELECT counts FROM mydb.visitor;', (err, data) => {
      if (err) throw err;
      cb(data);
    });
  },
  incCounts: (cb) => {
    connection.query(
      'UPDATE mydb.visitor SET counts = counts + 1 WHERE id = 1;',
      (err) => {
        if (err) throw err;
        cb(JSON.stringify('업데이트 성공'));
      }
    );
  },
};

module.exports = db;
```

Data-server/controlers/dataController.js



주소별

Routing 처리



```
const express = require('express');
const router = express.Router();

const db = require('../controllers/dataController');

router.get('/count', (req, res) => {
  db.getCounts((data) => {
    res.send(data);
  });
});

router.post('/inccount', (req, res) => {
  db.incCounts((msg) => {
    res.send(msg);
  });
});

module.exports = router;
```

Data-server/routes/data.js



React 앱에서 데이터 요청!

```
export default function Start() {
  const [counts, setCounts] = useState(0);
  const dispatch = useDispatch();

  useEffect(() => {
    async function fetchData() {
      const resCount = await fetch('http://localhost:4000/data/count');
      if (resCount.status === 200) {
        const num = await resCount.json();
        if (num[0].counts !== 0) setCounts(num[0].counts);
      } else {
        throw new Error('통신 이상');
      }
    }
    fetchData();
  }, [counts]);
```

Src/component/Start.js





통신 에러 시 대처 방법



```
D:\git\developer-mbti\node_modules\mysql\lib\protocol\Parser.js:437
  throw err; // Rethrow non-MySQL errors
  ^
```

```
Error: ER_NOT_SUPPORTED_AUTH_MODE: Client does not support authentication protocol requested by server;
consider upgrading MySQL client
    at Handshake.Sequence._packetToError (D:\git\developer-mbti\node_modules\mysql\lib\protocol\sequences\Sequence.js:47:14)
    at Handshake.ErrorPacket (D:\git\developer-mbti\node_modules\mysql\lib\protocol\sequences\Handshake.js:123:18)
    at Protocol.parsePacket (D:\git\developer-mbti\node_modules\mysql\lib\protocol\Protocol.js:281:23)
```

<https://1mini2.tistory.com/88>

3



조운호

2018.07.13 오후 4:33

<https://stackoverflow.com/questions/50093144/mysql-8-0-client-does-not-support-authentication-protocol-requested-by-server>

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '사용할패스워드'

뭔가 이렇게 하니까 됐어요~~

3 • ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '1234';



마지막 페이지에서

Count + 1 요청



```
export default function Show() {
  const result = useSelector((state) => state.mbti.mbtiResult);
  const explanation = useSelector((state) => state.mbti.explanation[result]);
  const dispatch = useDispatch();

  const incCount = async () => {
    const resInc = await fetch('http://localhost:4000/data/inccount', {
      method: 'POST',
    });
    if (resInc.status === 200) {
      console.log(await resInc.json());
    } else {
      throw new Error('통신 이상');
    }
  };
};
```

Src/component/Show.js



```
return (  
  <>  
    <Header>당신의 개발자 MBTI 결과는?</Header>  
    <Explanation>{explanation.text}</Explanation>  
    <Result>{result}</Result>  
    <Additional>이건 재미로 읽어 보세요!</Additional>  
    <AdditionalImg src={explanation.img} alt="팩폭" />  
    <PinkButton  
      text="다시 검사하기"  
      clickEvent={() => {  
        incCount();  
        dispatch(reset());  
      }}  
    />  
  </>  
>);  
}
```

Src/component/Show.js



wookimla.blog.me

자~ 이제 시작이야(내꿈을)



Redux

InitialData



Redux 에서 사용할 초기 데이터 구축!

- Redux 에서 사용할 초기 데이터를 위한 Table 을 만들어 봅시다!
- 모든 데이터를 하나의 테이블에 넣어도 되지만, DB Table 구성을 할 때에는 정규화(Normalization)을 따라 주는 것이 좋습니다!



정규화



정규화?

- DB 설계에 있어서 중복을 최소화 하기 위해 데이터를 구조화 하는 과정
- 크고 조직화 되지 않은 테이블 → 작고, 잘 조직된 테이블로 변경
- 이렇게 작성을 해야만, 데이터 추가 및 삭제 시에 이상 현상(Abnormal)을 예방 할 수 있습니다!



제 1 정규형(1NF)

- 하나의 컬럼은 반드시 하나의 속성만을 가져야 하는 법칙

학생번호	이름	과목
101	아이유	운영체제, DB
102	한지민	자바
103	한효주	C, C++

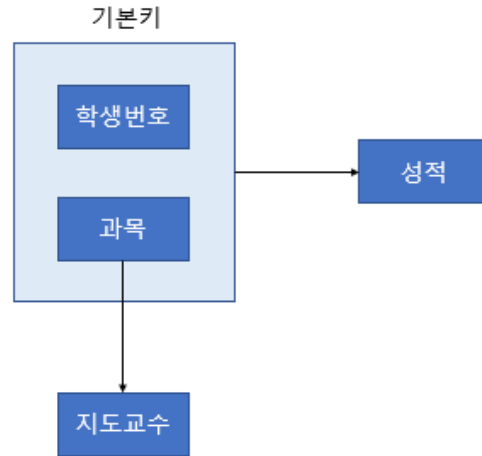
학생번호	이름	과목
101	아이유	운영체제
101	아이유	DB
102	한지민	자바
103	한효주	C
103	한효주	C++



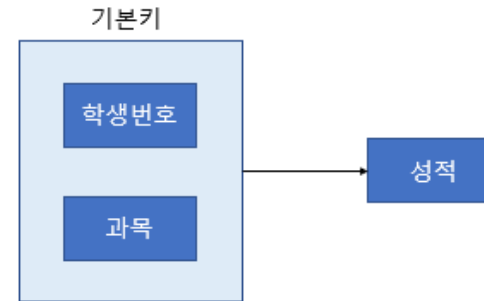
제 2 정규형(2NF)

- 모든 컬럼에 대한 **부분 종속**이 없어야 한다

학생번호	과목	지도교수	성적
101	운영체제	김운체	100
101	DB	조디비	60
102	자바	박자바	70
103	C	김씨	80
103	C++	이씨플	90



학생번호	과목	성적
101	운영체제	100
101	DB	60
102	자바	70
103	C	80
103	C++	90



과목	지도교수
운영체제	김운체
DB	조디비
자바	박자바
C	김씨
C++	이씨플

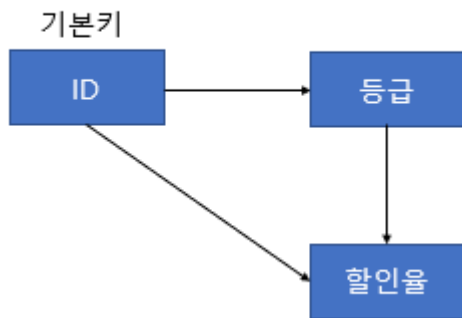


제 3 정규형(3NF) – 딱히 중요하지 않습니다!

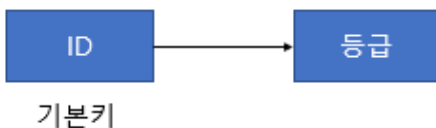


- 이행 종속성($A = B, B = C$ 여서 $A = C$ 인 경우)이 없어야 한다

ID	등급	할인율
101	Vip	40%
102	Gold	20%
103	Bronze	10%



ID	등급
101	Vip
102	Gold
103	Bronze



등급	할인율
Vip	40%
Gold	20%
Bronze	10%





Foreign Key

(외래 키)



외래 키(Foreign Key)

- 정규화를 하게 되면 테이블은 최소한의 단위로 쪼개지게 됩니다!
- 하지만 데이터를 불러 들일 때에는 한꺼번에 많은 값을 가지는 테이블을 JOIN 해서 가지고 와야 하는 경우가 많습니다.
- 앞서 배운 JOIN 을 사용하여 테이블을 합치게 되는데요. 보통 기준이 되는 값을 통해서 테이블을 합쳐 주게 됩니다! → 이 때 기준이 되는 값(서로 공유하고 있는 값)을 외래 키라고 합니다!



기본키(Primary Key)

외래키(Foreign Key)

기본키(Primary Key)

선수번호	이름	팀 코드	포지션	등번호	키
1	김남일	K03	DF	33	177
2	박지성	K07	MF	7	178
3	이영표	K02	MF	22	176

〈선수 테이블〉

팀 코드	팀 명	연고지
K03	스틸러스	포항
K07	드래곤즈	전남
K02	블루윙즈	수원

〈구단 테이블〉

선수번호	이름	팀 코드	포지션	등번호	키	팀 명	연고지
1	김남일	K03	DF	33	177	스틸러스	포항
2	박지성	K07	MF	7	178	드래곤즈	전남
3	이영표	K02	MF	22	176	블루윙즈	수원



InitialData

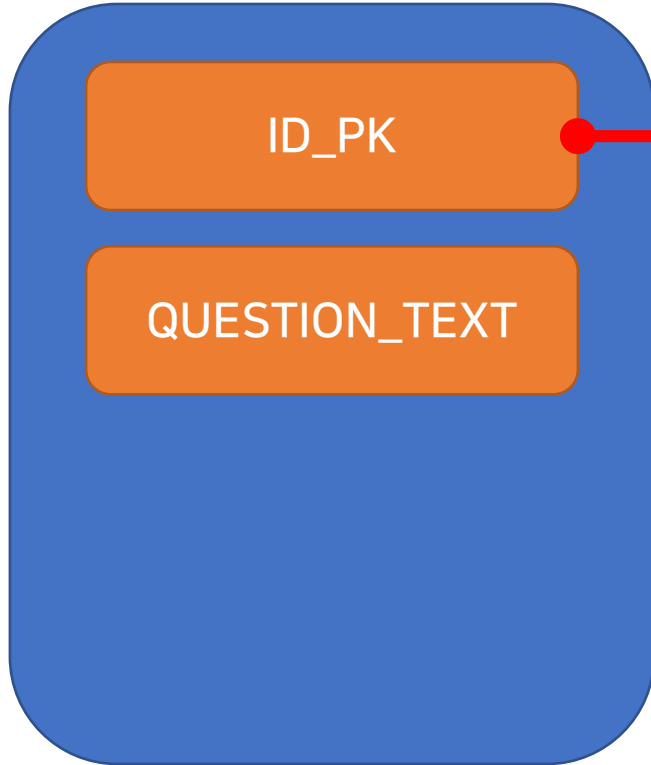
테이블 구성



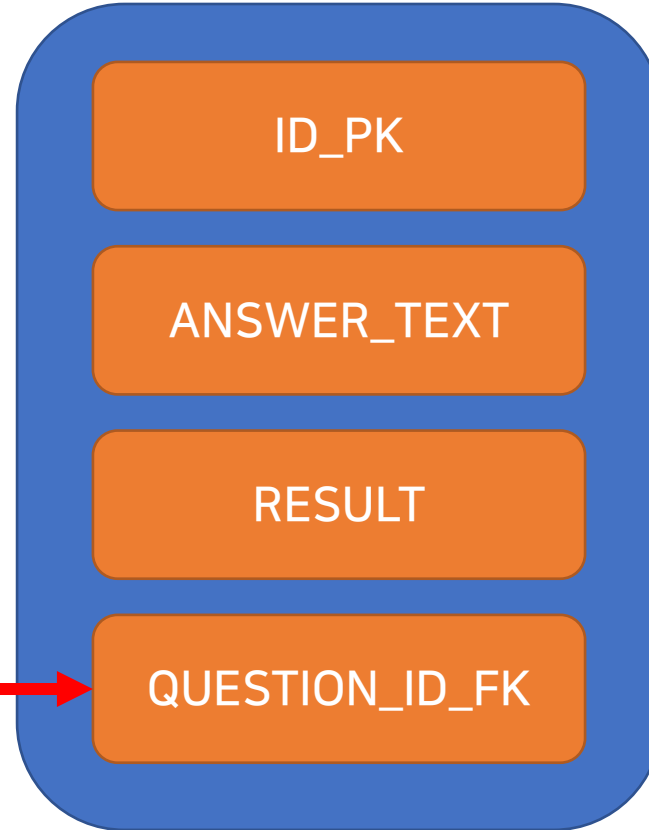
```
// 초기 상태 설정
const initState = {
  mbtiResult: '',
  page: 0, // 0: 인트로 페이지, 1 ~ n: 선택 페이지, n+1: 결과 페이지
  survey: [
    {
      question:
        '퇴근 직전에 동료로부터 개발자 모임에 초대를 받은 나!\n\n퇴근 시간에 나는?',
      answer: [
        {
          text: '그런 모임을 왜 이제서야 알려 준거야! 당장 모임으로 출발한다',
          result: 'E',
        },
        {
          text: '1년 전에 알려줬어도 안갔을 건데 뭘... 더 빠르게 집으로 간다',
          result: 'I',
        },
      ],
    },
    {
      question:
        '새로운 서비스 개발 중에, 동료가 새로 나온 신기술을 쓰는게 더 편할거라고 추천을 해',
      answer: [
        {
          text: '원소리여, 그냥 하던 대로 개발하면 되는거지! 기존 생각대로 개발한다',
          result: 'S',
        },
        {
          text: '오호? 그런게 있어? 일단 구글을 찾아본다',
          result: 'N',
        },
      ],
    },
  ],
},
```



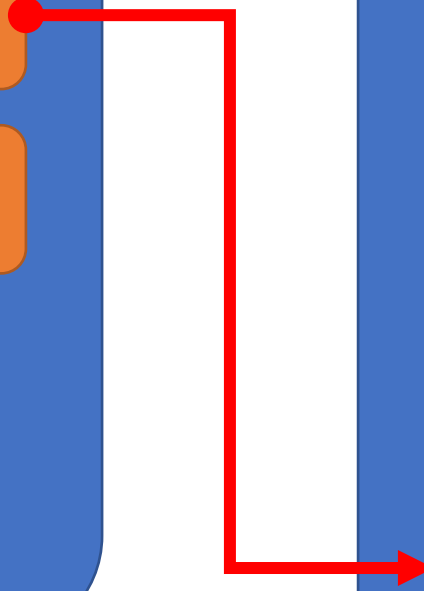
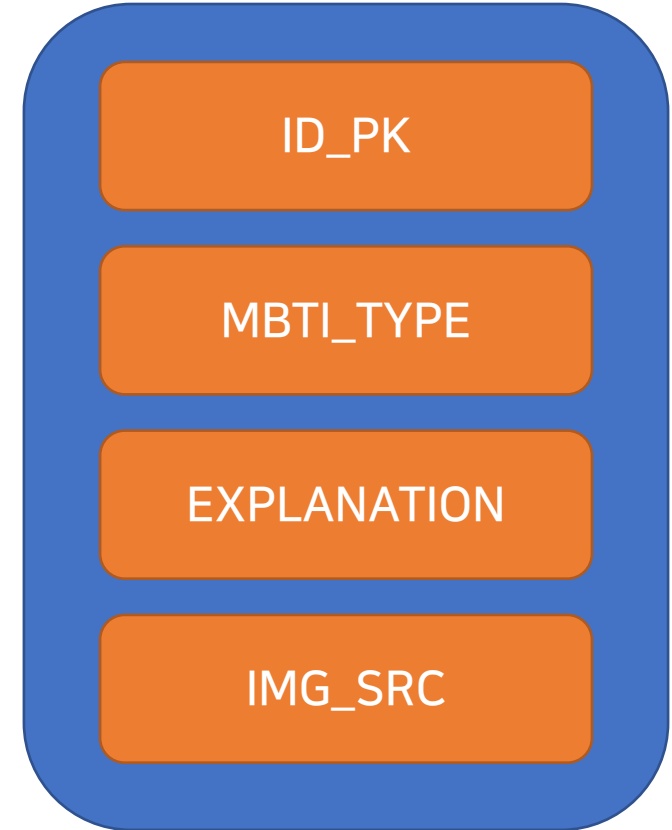
QUESTION
TABLE



ANSWER
TABLE



EXPLANATION
TABLE



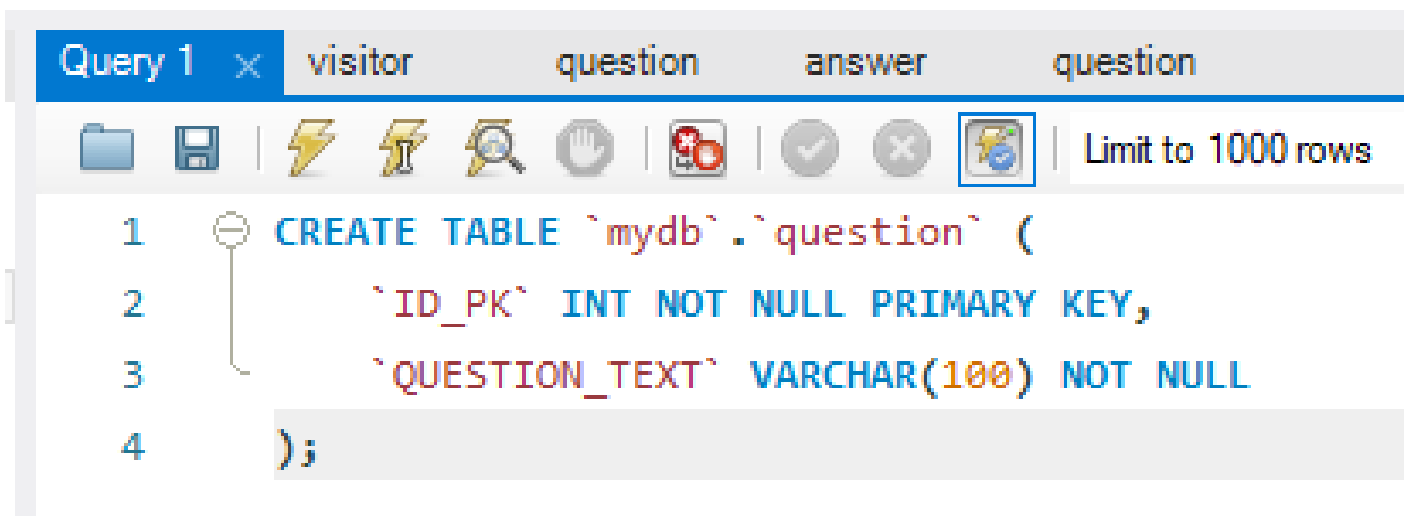


Question Table



Question Table 생성

- 먼저 Question 이 제일 상위에 필요 하므로 Question Table 만들기!
- Question Table 은 고유 Primary Key 역할을 할 ID_PK 값과
Question 의 질문 내용을 담고 있는 QUESTION_TEXT 로 구성 됩니다!

[illegible]



Answer Table



Answer Table 생성

- Answer 값은 Question 의 값에 종속 되므로, Question Table 의 ID_PK 값을 외래키로 사용하여 생성 합니다!
- 그리고 각각의 선택에 따른 결과 값도 가지고 있어야 합니다!



Query 1 visitor question answer question answer - Table x

Table Name:

Schema: **mbti**

Charset/Collation:

Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID_PK	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
ANSWER_TEXT	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
RESULT	VARCHAR(2)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
QUESTION_ID_FK	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name:

Charset/Collation:

Comments:

Data Type:

Default:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply

Revert

Query 1 visitor question answer question **answer - Table** x

Table Name: Schema: **mbti**

Charset/Collation: Engine:

Comments:

Foreign Key Name	Referenced Table
QUESTION_ID	`mbti`.`question`

Column	Referenced Column
<input type="checkbox"/> ID_PK	
<input type="checkbox"/> ANSWER_TEXT	
<input type="checkbox"/> RESULT	
<input checked="" type="checkbox"/> QUESTION_ID_FK	ID_PK

Foreign Key Options

On Update:

On Delete:

☐ Skip in SQL generation

Foreign Key Comment

Columns Indexes **Foreign Keys** Triggers Partitioning Options

Apply Revert

Foreign Key 설정!





Explanation Table



Explanation Table 생성

- Explanation Table 은 ID 값을 Primary Key, Auto Inc 로 설정해서 만들면 됩니다!
- 설명이 들어가는 EXPLANATION, 이미지 주소 값이 들어가는 IMG_SRC 컬럼을 만들어 주면 됩니다~!



Query 1 explanation visitor question answer question explanation explanation - Table x

Table Name: Schema: **mbti**

Charset/Collation: Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID_PK	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
MBTI_TYPE	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
EXPLANATION	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
IMG_SRC	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>

```
1 CREATE TABLE `mbti`.`explanation` (  
2   `ID_PK` INT NOT NULL AUTO_INCREMENT,  
3   `MBTI_TYPE` VARCHAR(45) NOT NULL,  
4   `EXPLANATION` VARCHAR(100) NOT NULL,  
5   `IMG_SRC` VARCHAR(45) NOT NULL,  
6   PRIMARY KEY (`ID_PK`));  
7
```



Question Table

Data 삽입!



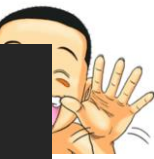
Question Table 에 Data 를 삽입!

- **INSERT INTO** 스키마명.테이블명 (컬럼명, 컬럼명) **VALUES** (값, 값);
- **INSERT INTO** 를 이용하여 값을 입력해 봅시다!

```
2 • INSERT INTO mbti.question (ID_PK, QUESTION_TEXT) VALUES
3     (0, '퇴근 직전에 동료로부터 개발자 모임에 초대를 받은 나!\n\n퇴근 시간에 나는?'),
4     (1, '새로운 서비스 개발 중에, 동료가 새로 나온 신기술을 쓰는게 더 편할거라고 추천을 해준다!\n\n나의 선택은!?'),
5     (2, '서비스 출시 이틀 전 야근 시간, 갑자기 동료가 어!? 를 외쳤다!\n\n나의 선택은?'),
6     (3, '팀장님이 XX씨 그전에 말한 기능 내일 오후까지 완료 부탁드립니다라고 말했다!\n\n나의 선택은?');
7
```

INSERT INTO mydb.question (ID_PK, QUESTION_TEXT) VALUES

- (0, '퇴근 직전에 동료로부터 개발자 모임에 초대를 받은 나!₩₩₩₩퇴근 시간에 나는?'),
- (1, '새로운 서비스 개발 중에, 동료가 새로 나온 신기술을 쓰는게 더 편할거라고 추천을 해준다!₩₩₩₩나의 선택은!?'),
- (2, '서비스 출시 이틀 전 야근 시간, 갑자기 동료가 어!? 를 외쳤다!₩₩₩₩나의 선택은?'),
- (3, '팀장님이 xx씨 그전에 말한 기능 내일 오후까지 완료 부탁드립니다라고 말했다!₩₩₩₩나의 선택은?');



Result Grid			Filter Rows:	Edit:
	ID_PK	QUESTION_TEXT		
▶	0	퇴근 직전에 동료로부터 개발자 모임에 초대를 ...		
	1	새로운 서비스 개발 중에, 동료가 새로 나온 신...		
	2	서비스 출시 이틀 전 야근 시간, 갑자기 동료가 ...		
	3	팀장님이 xx씨 그전에 말한 기능 내일 오후까지...		
●	NULL	NULL		



Answer Table

Data 삽입!



```
INSERT INTO mydb.answer (ANSWER_TEXT, RESULT, QUESTION_ID_FK) VALUES
('그런 모임을 왜 이제서야 알려 준거야! 당장 모임으로 출발한다', 'E', 0),
('1년 전에 알려줬어도 안갔을 건데 뭘... 더 빠르게 집으로 간다', 'I', 0),
('뭘소리여, 그냥 하던 대로 개발하면 되는거지! 기존 생각대로 개발한다', 'S', 1),
('오호? 그런게 있어? 일단 구글을 찾아본다', 'N', 1),
('무슨 버그가 발생한 거지? 아마 DB 관련 버그가 아닐까? 빠르게 동료의 자리로 달려간다', 'T', 2),
('아... 내일도 야근 각이구나 ㅠㅠ! 일단 동료의 자리로 가 본다', 'F', 2),
('일단 빠르게 개발 완료하고, 나머지 시간에 논다', 'J', 3),
('그거 내일 아침에 와서 개발해도 충분 하겠는데? 일단 논다', 'P', 3);
```

	ID_PK	ANSWER_TEXT	RESULT	QUESTION_ID_FK
▶	1	그런 모임을 왜 이제서야 알려 준거야! 당장 모...	E	0
	2	1년 전에 알려줬어도 안갔을 건데 뭘... 더 빠르...	I	0
	3	뭘소리여, 그냥 하던 대로 개발하면 되는거지! ...	S	1
	4	오호? 그런게 있어? 일단 구글을 찾아본다	N	1
	5	무슨 버그가 발생한 거지? 아마 DB 관련 버그가 ...	T	2
	6	아... 내일도 야근 각이구나 ㅠㅠ! 일단 동료의 ...	F	2
	7	일단 빠르게 개발 완료하고, 나머지 시간에 논다	J	3
	8	그거 내일 아침에 와서 개발해도 충분 하겠는데...	P	3
•	NULL	NULL	NULL	NULL



Explanation Table

Data 삽입!



```
INSERT INTO mydb.explanation (MBTI_TYPE, EXPLANATION, IMG_SRC) VALUES
('ESTJ', '무리한 개발 일정만 아니라면 일정을 철저하게 지킬 당신의 MBTI 는!', '/images/estj.jpg'),
('ISTJ', '스스로 하고싶은 분야를 끝까지 파고 들어서 끝내 성공 시킬 당신의 MBTI 는!', '/images/istj.jpg'),
('ENTJ', '미래의 능력 켜는 개발 팀장님으로 개발팀을 이끌 당신의 MBTI 는!', '/images/entj.jpg'),
('INTJ', '혼자서 모든 것을 다 해내는 원맨 캐리의 표본! 당신의 MBTI 는!', '/images/intj.jpg'),
('ESFJ', '개발팀의 분위기 메이커이자 아이디어 뱅크가 될 당신의 MBTI 는!', '/images/esfj.jpg'),
('ISFJ', '개발팀의 마더 테레사, 고민 상담소 역할을 자처하는 당신의 MBTI 는!', '/images/isfj.jpg'),
('ENFJ', '당신이 있는 팀은 언제나 올바른 곳을 향하고 있습니다! 팀원은 물론 팀의 방향을 챙기는 당신의 MBTI 는!', '/images/enfj.jpg'),
('INFJ', '예리한 통찰력으로 모든 것을 내다보면서 완벽하게 개발을 할 당신의 MBTI 는!', '/images/infj.jpg'),
('ESTP', '쿨하게 자신이 할 것을 하면서 논리적인 개발을 할 당신의 MBTI 는!', '/images/estp.jpg'),
('ISTP', '단시간에도 효율적으로 개발하여 모든 것을 완성할 당신의 MBTI 는!', '/images/istp.jpg'),
('ENTP', '스스로 흥미만 생긴다면 당장에 페이스북도 만들어 버릴 당신의 MBTI 는!', '/images/entp.jpg'),
('INTP', '확실한 주관과 뛰어난 지능을 바탕으로 논리적 개발을 할 당신의 MBTI 는!', '/images/intp.jpg'),
('ESFP', '개발팀의 에너지이저! 개발팀 특유의 서먹함을 깨는 당신! 당신의 MBTI 는!', '/images/esfp.jpg'),
('ISFP', '뛰어난 호기심과 예술적 감각으로 개발팀의 부족함을 채워갈 당신! 당신의 MBTI 는!', '/images/isfp.jpg'),
('ENFP', '자유로운 영혼으로 개발팀의 윤택유 및 활력소가 되어줄 당신의 MBTI 는!', '/images/enfp.jpg'),
('INFP', '개발팀의 그 어떤 트러블도 당신 앞에서는 사르르 녹을뿐, 팀의 근간을 다져주는 당신의 MBTI 는!', '/images/infp.jpg');
```

	ID_PK	MBTI_TYPE	EXPLANATION	IMG_SRC
▶	1	ESTJ	무리한 개발 일정만 아니라면 일정을 철저하게 ...	/images/estj.jpg
	2	ISTJ	스스로 하고싶은 분야를 끝까지 파고 들어서 끝...	/images/istj.jpg
	3	ENTJ	미래의 능력 켜는 개발 팀장님으로 개발팀을 이...	/images/entj.jpg
	4	INTJ	혼자서 모든 것을 다 해내는 원맨 캐리의 표본! ...	/images/intj.jpg
	5	ESFJ	개발팀의 분위기 메이커이자 아이디어 뱅크가 ...	/images/esfj.jpg
	6	ISFJ	개발팀의 마더 테레사, 고민 상담소 역할을 자...	/images/isfj.jpg
	7	ENFJ	당신이 있는 팀은 언제나 올바른 곳을 향하고 있...	/images/enfj.jpg
	8	INFJ	예리한 통찰력으로 모든 것을 내다보면서 완벽...	/images/infj.jpg
	9	ESTP	쿨하게 자신이 할 것을 하면서 논리적인 개발을 ...	/images/estp.jpg
	10	ISTP	단시간에도 효율적으로 개발하여 모든 것을 완...	/images/istp.jpg
	11	ENTP	스스로 흥미만 생긴다면 당장에 페이스북도 만...	/images/entp.jpg
	12	INTP	확실한 주관과 뛰어난 지능을 바탕으로 논리적 ...	/images/intp.jpg
	13	ESFP	개발팀의 에너지이저! 개발팀 특유의 서먹함을 ...	/images/esfp.jpg
	14	ISFP	뛰어난 호기심과 예술적 감각으로 개발팀의 부...	/images/isfp.jpg
	15	ENFP	자유로운 영혼으로 개발팀의 윤택유 및 활력소...	/images/enfp.jpg
	16	INFP	개발팀의 그 어떤 트러블도 당신 앞에서는 사르...	/images/infp.jpg



Foreign Key

동작 확인하기!




Foreign Key 동작 확인하기




- 아까 Answer Table 생성시 설정한 Foreign Key 가 정상 작동하는지 봅시다!
- 먼저, Question Table 의 ID_PK 를 변경 해보기!




	ID_PK	QUESTION_TEXT
...	7	퇴근 직전에 동료로부터 개발자 모임에 초대를 ...
	1	새로운 서비스 개발 중에, 동료가 새로 나온 신...
	2	서비스 출시 이틀 전 야근 시간, 갑자기 동료가 ...
	3	팀장님이 xx씨 그전에 말한 기능 내일 오후까지...
●	NULL	NULL

Result Grid

 Filter Rows:

Edit:   

Export/Import: 

	ID_PK	ANSWER_TEXT	RESULT	QUESTION_ID_FK
▶	1	그런 모임을 왜 이제서야 알려 준거야! 당장 모...	E	7
	2	1년 전에 알려줬어도 안갔을 건데 뭘... 더 빠르...	I	7
	3	웬소리여, 그냥 하던 대로 개발하면 되는거지! ...	S	1
	4	오호? 그런게 있어? 일단 구글을 찾아본다	N	1
	5	무슨 버그가 발생한 거지? 아마 DB 관련 버그가 ...	T	2
	6	아... 내일도 야근 각이구나 ππ! 일단 동료의 ...	F	2
	7	일단 빠르게 개발 완료하고, 나머지 시간에 논다	J	3
	8	그거 내일 아침에 와서 개발해도 충분 하겠는데...	P	3
●	NULL	NULL	NULL	NULL



Foreign Key 동작 확인하기

- 그럼 이번에는 Answer Table 의 QUESTION_ID_FK 를 변경해 봅시다!



Result Grid

Filter Rows:

Edit: Export/Import: Wrap Cell Content: ☐

ID_PK	ANSWER_TEXT	RESULT	QUESTION_ID_FK
1	그런 모임을 왜 이제서야 알려 준거야! 당장 모...	E	15
2	1년 전에 알려줬어도 안갔을 건데 똥... 더 빠르...	I	7
3	원소리며, 그냥 하던 대로 개발하면 되는거지! ...	S	1
4	오호? 그런게 있어? 일단 구글을 찾아본다	N	1
5	무슨 버그가 발생한 거지? 아마 DB 관련 버그가 ...	T	2
6	아... 내일도 야근 각이구나 ㅠㅠ! 일단 동료의 ...	F	2
7	일단 빠르게 개발 완료하고, 나머지 시간에 논다	J	3
8	그거 내일 아침에 와서 개발해도 충분 하겠는데...	P	3
* NULL	NULL	NULL	NULL

answer 8 x

Apply Revert

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

❌ Execute SQL Statements

Error: There was an error while applying the SQL script to the database.



Redux InitialData를
SQL DB로 부터 받기!



Redux InitialData 받기

- 이제 DB Table 구성을 마쳤으니, 해당 DB로 부터 원하는 데이터를 받아 봅시다!
- 기존의 Backend 서버의 Controller 와 Router 를 수정하고, 리액트 App 에서 데이터 요청을 보내 봅시다!



Survey

컨트롤러 추가!



```
// 초기 상태 설정
const initState = {
  mbtiResult: '',
  page: 0, // 0: 인트로 페이지, 1 ~ n: 선택 페이지, n+1: 결과 페이지
  survey: [
    {
      question:
        '퇴근 직전에 동료로부터 개발자 모임에 초대를 받은 나!\n\n퇴근 시간에 나는?',
      answer: [
        {
          text: '그런 모임을 왜 이제서야 알려 준거야! 당장 모임으로 출발한다',
          result: 'E',
        },
        {
          text: '1년 전에 알려줬어도 안갔을 건데 뭘... 더 빠르게 집으로 간다',
          result: 'I',
        },
      ],
    },
  ],
  {
    question:
      '새로운 서비스 개발 중에, 동료가 새로 나온 신기술을 쓰는게 더 편할거라고 추천을 해',
    answer: [
      {
        text: '원소리여, 그냥 하던 대로 개발하면 되는거지! 기존 생각대로 개발한다',
        result: 'S',
      },
      {
        text: '오호? 그런게 있어? 일단 구글을 찾아본다',
        result: 'N',
      },
    ],
  },
],
},
```



설문 조사를 위한 Data 만들기!

- 설문 조사는 survey 라는 객체에서 질문 내용과, 답변 내용, 그리고 그에 대한 결과를 받아서 처리 했습니다.
- 하지만 현재 MySQL 에 데이터는 정규화를 위해서 테이블이 분리 된 상태이기 때문에, Question Table 과 Answer Table 을 동시에 가지고 와야 합니다!



설문 조사를 위한 Data 만들기!

- Question Table 과 Answer Table 을 Foreign Key 인 Question ID 값을 이용하여 합쳐 봅시다!
- `SELECT * FROM mydb.question q LEFT JOIN mydb.answer a ON q.ID_PK=a.QUESTION_ID_FK`

Result Grid



Filter Rows:

Export:



Wrap Cell Content:



	ID_PK	QUESTION_TEXT	ID_PK	ANSWER_TEXT	RESULT	QUESTION_ID_FK
▶	0	퇴근 직전에 동료로부터 개발자 모임에 초대를 ...	1	그런 모임을 왜 이제서야 알려 준거야! 당장 모...	E	0
	0	퇴근 직전에 동료로부터 개발자 모임에 초대를 ...	2	1년 전에 알려줬어도 안갔을 건데 뭘... 더 빠르...	I	0
	1	새로운 서비스 개발 중에, 동료가 새로 나온 신...	3	원소리여, 그냥 하던 대로 개발하면 되는거지! ...	S	1
	1	새로운 서비스 개발 중에, 동료가 새로 나온 신...	4	오호? 그런게 있어? 일단 구글을 찾아본다	N	1
	2	서비스 출시 이틀 전 야근 시간, 갑자기 동료가 ...	5	무슨 버그가 발생한 거지? 아마 DB 관련 버그가 ...	T	2
	2	서비스 출시 이틀 전 야근 시간, 갑자기 동료가 ...	6	아... 내일도 야근 각이구나 ㅠㅠ! 일단 동료의 ...	F	2
	3	팀장님이 xx씨 그전에 말한 기능 내일 오후까지...	7	일단 빠르게 개발 완료하고, 나머지 시간에 논다	J	3
	3	팀장님이 xx씨 그전에 말한 기능 내일 오후까지...	8	그거 내일 아침에 와서 개발해도 충분 하겠는데...	P	3





JOIN 으로 합쳐진 데이터 받기!

- 기존의 Controller 모듈에 getSurvey 메소드를 추가하고, JOIN 으로 합친 데이터의 값을 전달해 봅시다!

```
getSurvey: (cb) => {  
  connection.query(  
    'SELECT * FROM mydb.question q LEFT JOIN mydb.answer a ON q.ID_PK=a.QUESTION_ID_FK',  
    (err, data) => {  
      if (err) throw err;  
      cb(data);  
    }  
  );  
},
```

Data-server/controllers/dataController.js



Survey

Router 설정



Router 설정

- `/survey` 라는 주소로 GET 방식 요청이 들어오면 합쳐진 테이블의 값을 받아서 전달해 주는 Router 만들기 + 데이터 출력 해보기!!

```
router.get('/survey', (req, res) => {  
  db.getSurvey((data) => {  
    console.log(data);  
    res.send(data);  
  });  
});
```

Data-server/routes/data.js



```
[
  RowDataPacket {
    ID_PK: 1,
    QUESTION_TEXT: '퇴근 직전에 동료로부터 개발자 모임에 초대를 받은 나!\n\n\n퇴근 시간에 나는',
    ANSWER_TEXT: '그런 모임을 왜 이제서야 알려 준거야! 당장 모임으로 출발한다',
    RESULT: 'E',
    QUESTION_ID_FK: 0
  },
  RowDataPacket {
    ID_PK: 2,
    QUESTION_TEXT: '퇴근 직전에 동료로부터 개발자 모임에 초대를 받은 나!\n\n\n퇴근 시간에 나는',
    ANSWER_TEXT: '1년 전에 알려줬어도 안갔을 건데 뭘... 더 빠르게 집으로 간다',
    RESULT: 'I',
    QUESTION_ID_FK: 0
  },
  RowDataPacket {
    ID_PK: 3,
    QUESTION_TEXT: '새로운 서비스 개발 중에, 동료가 새로 나온 신기술을 쓰는게 더 편할거라고 추천을 해준다!\n\n\n나의 선택은!?',
    ANSWER_TEXT: '원소리여, 그냥 하던 대로 개발하면 되는거지! 기존 생각대로 개발한다',
    RESULT: 'S',
    QUESTION_ID_FK: 1
  },
  RowDataPacket {
    ID_PK: 4,
    QUESTION_TEXT: '새로운 서비스 개발 중에, 동료가 새로 나온 신기술을 쓰는게 더 편할거라고 추천을 해준다!\n\n\n나의 선택은!?',
    ANSWER_TEXT: '오호? 그런게 있어? 일단 구글을 찾아본다',
    RESULT: 'N',
  },
]
```




React 에서 데이터 받기!

React 앱에서 데이터 요청 및 데이터 받기!



- Start 컴포넌트에서 <http://localhost:4000/data/survey> 주소로 요청을 보내서 데이터가 잘 들어오는지 확인해 봅시다!



```
useEffect(() => {
  async function fetchData() {
    // counts 값 받아오기
    const resCount = await fetch('http://localhost:4000/data/count');
    if (resCount.status === 200) {
      const num = await resCount.json();
      if (num[0].counts !== 0) setCounts(num[0].counts);
    } else {
      throw new Error('통신 이상');
    }
  }

  // survey 값을 위한 JOIN Table 의 데이터 받아오기
  const resSurvey = await fetch('http://localhost:4000/data/survey');
  if (resSurvey.status === 200) {
    const surveyData = await resSurvey.json();
    console.log(surveyData);
  } else {
    throw new Error('통신 이상');
  }
}

fetchData();
}, []);
```

Src/component/Start.js



Start.js:117

▼ (8) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}] ⓘ

▼ 0:

ANSWER_TEXT: "그런 모임을 왜 이제서야 알려 준거야! 당장 모임으로 출발한다"

ID_PK: 1

QUESTION_ID_FK: 0

QUESTION_TEXT: "퇴근 직전에 동료로부터 개발자 모임에 초대를 받은 나!\n\n\n퇴근 시간에 나는"

RESULT: "E"

▶ [[Prototype]]: Object

- ▶ 1: {ID_PK: 2, QUESTION_TEXT: '퇴근 직전에 동료로부터 개발자 모임에 초대를 받은 나!\n\n\n퇴근 시간에 나는', ANSWER_TEXT: '1년 전에 알려줬어도 안갔을 2
 - ▶ 2: {ID_PK: 3, QUESTION_TEXT: '새로운 서비스 개발 중에, 동료가 새로 나온 신기술을 쓰는게 더 편할거라고 추천을 해준다!\n\n\n나의 선택은!?', ANSWER_TEXT
 - ▶ 3: {ID_PK: 4, QUESTION_TEXT: '새로운 서비스 개발 중에, 동료가 새로 나온 신기술을 쓰는게 더 편할거라고 추천을 해준다!\n\n\n나의 선택은!?', ANSWER_TEXT
 - ▶ 4: {ID_PK: 5, QUESTION_TEXT: '서비스 출시 이틀 전 야근 시간, 갑자기 동료가 어!? 를 외쳤다!\n\n\n나의 선택은?', ANSWER_TEXT: '무슨 버그가 발생한 거지?
 - ▶ 5: {ID_PK: 6, QUESTION_TEXT: '서비스 출시 이틀 전 야근 시간, 갑자기 동료가 어!? 를 외쳤다!\n\n\n나의 선택은?', ANSWER_TEXT: '아... 내일도 야근 각이구
 - ▶ 6: {ID_PK: 7, QUESTION_TEXT: '팀장님이 xx씨 그전에 말한 기능 내일 오후까지 완료 부탁드립니다라고 말했다!\n\n\n나의 선택은?', ANSWER_TEXT: '일단 빠르게 개
 - ▶ 7: {ID_PK: 8, QUESTION_TEXT: '팀장님이 xx씨 그전에 말한 기능 내일 오후까지 완료 부탁드립니다라고 말했다!\n\n\n나의 선택은?', ANSWER_TEXT: '그거 내일 아침
- length: 8

• Table 의 데이터는 잘 들어 오는 것을 확인!



Explanation

컨트롤러 추가!



Explanation 데이터 받기!

- explanation 테이블의 값을 받아서 전달 하는 getExplanation 메소드 추가!

```
getExplanation: (cb) => {  
  connection.query('SELECT * FROM mydb.explanation', (err, data) => {  
    if (err) throw err;  
    cb(data);  
  });  
},
```

Data-server/controllers/dataController.js



Explanation

Router 설정



Router 설정

- `/explanation` 라는 주소로 GET 방식 요청이 들어오면 DB의 값을 받아서 전달해 주는 Router 만들기 + 데이터 출력 해보기!!

```
router.get('/explanation', (req, res) => {  
  db.getExplanation((data) => {  
    console.log(data);  
    res.send(data);  
  });  
});
```

Data-server/routes/data.js

데이터 통신 서버가 4000에서 작동 중

```
[
  RowDataPacket {
    ID_PK: 1,
    MBTI_TYPE: 'ESTJ',
    EXPLANATION: '무리한 개발 일정만 아니라면 일정을 철저하게 지킬 당신의 MBTI 는!',
    IMG_SRC: '/images/estj.jpg'
  },
  RowDataPacket {
    ID_PK: 2,
    MBTI_TYPE: 'ISTJ',
    EXPLANATION: '스스로 하고싶은 분야를 끝까지 파고 들어서 끝내 성공 시킬 당신의 MBTI 는!',
    IMG_SRC: '/images/istj.jpg'
  },
  RowDataPacket {
    ID_PK: 3,
    MBTI_TYPE: 'ENTJ',
    EXPLANATION: '미래의 능력 켜는 개발 팀장님으로 개발팀을 이끌 당신의 MBTI 는!',
    IMG_SRC: '/images/entj.jpg'
  },
  RowDataPacket {
    ID_PK: 4,
    MBTI_TYPE: 'INTJ',
    EXPLANATION: '혼자서 모든 것을 다 해내는 원맨 캐리의 표본! 당신의 MBTI 는!',
    IMG_SRC: '/images/intj.jpg'
  },
  RowDataPacket {
```





React 에서 데이터 받기!



React 앱에서 데이터 요청 및 데이터 받기!

- Start 컴포넌트에서 <http://localhost:4000/data/explanation> 주소로 요청을 보내서 데이터가 잘 들어오는지 확인해 봅시다!



```
useEffect(() => {
  async function fetchData() {
    const resCount = await fetch('http://localhost:4000/data/count');
    if (resCount.status === 200) {
      const num = await resCount.json();
      if (num[0].counts !== 0) setCounts(num[0].counts);
    } else {
      throw new Error('통신 이상');
    }
    const resSurvey = await fetch('http://localhost:4000/data/survey');
    if (resSurvey.status === 200) {
      const surveyData = await resSurvey.json();
      // explanation Table 의 데이터 받아오기
      const resExplanation = await fetch(
        'http://localhost:4000/data/explanation'
      );
      if (resExplanation.status === 200) {
        const explanationData = await resExplanation.json();
        console.log(explanationData);
      } else {
        throw new Error('통신 이상');
      }
    } else {
      throw new Error('통신 이상');
    }
  }
}
```



```

▼ (16) [{-}, {-}, {-}, {-}, {-}, {-}, {-}, {-}, {-}, {-}, {-}, {-}, {-}, {-}, {-}, {-}] ⓘ
  ▼ 0:
    EXPLANATION: "무리한 개발 일정만 아니라면 일정을 철저히 지킬 당신의 MBTI 는!"
    ID_PK: 1
    IMG_SRC: "/images/estj.jpg"
    MBTI_TYPE: "ESTJ"
    ▶ [[Prototype]]: Object
  ▶ 1: {ID_PK: 2, MBTI_TYPE: 'ISTJ', EXPLANATION: '스스로 하고싶은 분야를 끝까지 파고 들어서 끝내 성공 시킬 당신의 MBTI 는!', IMG_SRC: '/images/istj.jp'
  ▶ 2: {ID_PK: 3, MBTI_TYPE: 'ENTJ', EXPLANATION: '미래의 능력 켜는 개발 팀장님으로 개발팀을 이끌 당신의 MBTI 는!', IMG_SRC: '/images/entj.jpg'}
  ▶ 3: {ID_PK: 4, MBTI_TYPE: 'INTJ', EXPLANATION: '혼자서 모든 것을 다 해내는 원맨 캐리의 표본! 당신의 MBTI 는!', IMG_SRC: '/images/intj.jpg'}
  ▶ 4: {ID_PK: 5, MBTI_TYPE: 'ESFJ', EXPLANATION: '개발팀의 분위기 메이커이자 아이디어 뱅크가 될 당신의 MBTI 는!', IMG_SRC: '/images/esfj.jpg'}
  ▶ 5: {ID_PK: 6, MBTI_TYPE: 'ISFJ', EXPLANATION: '개발팀의 마더 테레사, 고민 상담소 역할을 자처하는 당신의 MBTI 는!', IMG_SRC: '/images/isfj.jpg'}
  ▶ 6: {ID_PK: 7, MBTI_TYPE: 'ENFJ', EXPLANATION: '당신이 있는 팀은 언제나 올바른 곳을 향하고 있습니다! 팀원은 물론 팀의 방향을 챙기는 당신의 MBTI 는!',
  ▶ 7: {ID_PK: 8, MBTI_TYPE: 'INFJ', EXPLANATION: '예리한 통찰력으로 모든 것을 내다보면서 완벽하게 개발을 할 당신의 MBTI 는!', IMG_SRC: '/images/infj.j'
  ▶ 8: {ID_PK: 9, MBTI_TYPE: 'ESTP', EXPLANATION: '쿨하게 자신이 할 것을 하면서 논리적인 개발을 할 당신의 MBTI 는!', IMG_SRC: '/images/estp.jpg'}
  ▶ 9: {ID_PK: 10, MBTI_TYPE: 'ISTP', EXPLANATION: '단시간에도 효율적으로 개발하여 모든 것을 완성할 당신의 MBTI 는!', IMG_SRC: '/images/istp.jpg'}
  ▶ 10: {ID_PK: 11, MBTI_TYPE: 'ENTP', EXPLANATION: '스스로 흥미만 생긴다면 당장에 페이스북도 만들어 버릴 당신의 MBTI 는!', IMG_SRC: '/images/entp.jpg'
  ▶ 11: {ID_PK: 12, MBTI_TYPE: 'INTP', EXPLANATION: '확실한 주관과 뛰어난 지능을 바탕으로 논리적 개발을 할 당신의 MBTI 는!', IMG_SRC: '/images/intp.jpg'
  ▶ 12: {ID_PK: 13, MBTI_TYPE: 'ESFP', EXPLANATION: '개발팀의 에너지아저! 개발팀 특유의 서먹함을 깨는 당신! 당신의 MBTI 는!', IMG_SRC: '/images/esfp.jp'
  ▶ 13: {ID_PK: 14, MBTI_TYPE: 'ISFP', EXPLANATION: '뛰어난 호기심과 예술적 감각으로 개발팀의 부족함을 채워갈 당신! 당신의 MBTI 는!', IMG_SRC: '/images/'
  ▶ 14: {ID_PK: 15, MBTI_TYPE: 'ENFP', EXPLANATION: '자유로운 영혼으로 개발팀의 윤택유 및 활력소가 되어줄 당신의 MBTI 는!', IMG_SRC: '/images/enfp.jpg'
  ▶ 15: {ID_PK: 16, MBTI_TYPE: 'INFP', EXPLANATION: '개발팀의 그 어떤 트러블도 당신 앞에서는 사르르 녹을뿐, 팀의 균간을 다져주는 당신의 MBTI 는!', IMG_SI
    length: 16
  ▶ [[Prototype]]: Array(0)

```

- Explanation 데이터도 잘 들어 오는 것을 확인!



**받아온 데이터를
꾸미기!**



받은 데이터 꾸미기!

- 지금 받은 데이터는 애초에 Redux 에서 세팅 하였던 initialState 랑은 구조가 다릅니다!
- 따라서, 지금 받은 데이터를 initialState 에 맞게 꾸며줘야 합니다!
- 현재 받은 데이터는 surveyData 와 ExlanationData 에 들어 있으므로, 해당 데이터를 가공해서 만들어 봅시다!
- 저는 makeData() 라는 함수를 만들어서 만들 생각입니다!



```
function makeData(survey, explanation) {
  const initData = { survey: [], explanation: {} };
  if (initData.survey.length === 0) {
    for (let i = 0; i < survey.length; i = i + 2) {
      initData.survey.push({
        question: survey[i].QUESTION_TEXT,
        answer: [
          {
            text: survey[i].ANSWER_TEXT,
            result: survey[i].RESULT,
          },
          {
            text: survey[i + 1].ANSWER_TEXT,
            result: survey[i + 1].RESULT,
          },
        ],
      });
    }

    for (let i = 0; i < explanation.length; i++) {
      initData.explanation[explanation[i].MBTI_TYPE] = {
        explanation: explanation[i].EXPLANATION,
        img: explanation[i].IMG_SRC,
      };
    }
  }
  return initData;
}
```

Src/component/Start.js



```

▼ {survey: Array(4), explanation: {...}} ⓘ
  ▼ explanation:
    ▶ ENFJ: {explanation: undefined, img: '/images/enfj.jpg'}
    ▶ ENFP: {explanation: undefined, img: '/images/enfp.jpg'}
    ▶ ENTJ: {explanation: undefined, img: '/images/entj.jpg'}
    ▶ ENTP: {explanation: undefined, img: '/images/entp.jpg'}
    ▶ ESFJ: {explanation: undefined, img: '/images/esfj.jpg'}
    ▶ ESFP: {explanation: undefined, img: '/images/esfp.jpg'}
    ▶ ESTJ: {explanation: undefined, img: '/images/estj.jpg'}
    ▶ ESTP: {explanation: undefined, img: '/images/estp.jpg'}
    ▶ INFJ: {explanation: undefined, img: '/images/infj.jpg'}
    ▶ INFP: {explanation: undefined, img: '/images/infp.jpg'}
    ▶ INTJ: {explanation: undefined, img: '/images/intj.jpg'}
    ▶ INTP: {explanation: undefined, img: '/images/intp.jpg'}
    ▶ ISFJ: {explanation: undefined, img: '/images/isfj.jpg'}
    ▶ ISFP: {explanation: undefined, img: '/images/isfp.jpg'}
    ▶ ISTJ: {explanation: undefined, img: '/images/istj.jpg'}
    ▶ ISTP: {explanation: undefined, img: '/images/istp.jpg'}
    ▶ [[Prototype]]: Object
  ▼ survey: Array(4)
    ▼ 0:
      ▼ answer: Array(2)
        ▶ 0: {text: '그런 모임을 왜 이제서야 알려 준거야! 당장 모임으로 출발한다', result: 'E'}
        ▶ 1: {text: '1년 전에 알려줬어도 안갔을 건데 뭐... 더 빠르게 집으로 간다', result: 'I'}
        length: 2
        ▶ [[Prototype]]: Array(0)
        question: "퇴근 직전에 동료로부터 개발자 모임에 초대를 받은 나!\n\n\n퇴근 시간에 나는"
        ▶ [[Prototype]]: Object
      ▶ 1: {question: '새로운 서비스 개발 중에, 동료가 새로 나온 기술을 쓰는게 더 편할거라고 추천을 해준다!\n\n\n나의 선택은!?', answer: Array(2)}
      ▶ 2: {question: '서비스 출시 이틀 전 야근 시간, 갑자기 동료가 어!? 를 외쳤다!\n\n\n나의 선택은?', answer: Array(2)}
      ▶ 3: {question: '팀장님이 xx씨 그전에 말한 기능 내일 오후까지 완료 부탁드립니다라고 말했다!\n\n\n나의 선택은?', answer: Array(2)}
        length: 4
        ▶ [[Prototype]]: Array(0)
      ▶ [[Prototype]]: Object

```



Redux

수정하기!



InitData 를 redux 에 전달!

- 이제 초창기 redux 에서 세팅 했던 initData 와 동일한 Data 를 만들었으므로, 해당 data 를 redux 에 전달하여 초기 값으로 만들어 봅시다!
- Data 를 초기화 시켜주는 작업도 하나의 Action 이므로 redux 에 Action Type 설정 → Action 생성 함수 만들기 → 리듀서 작업의 순서로 진행해 봅시다!



Action 설정(Type, 생성 함수)

```
// 액션 타입에 INIT 추가
const INIT = 'mbti/INIT';
const CHECK = 'mbti/CHECK';
const NEXT = 'mbti/NEXT';
const RESET = 'mbti/RESET';
```

Src/store/modules/Show.js

```
// 액션 생성 함수 추가
export function init(data) {
  return {
    type: INIT,
    payload: data,
  };
}
```

Src/store/modules/Show.js

Redux 최초 로딩 시에 전달한 초기값 만들기!



- 리액트 앱이 최초 실행 될 때, reducer 에 전달할 빈 초기 값 설정을 해봅시다!

```
const initStateEmpty = {  
  mbtiResult: '',  
  page: 0,  
  survey: [],  
  explanation: {},  
};
```

Src/store/modules/Show.js



리듀서 작업하기!

- 빈 초기 값을 Reducer 의 State 에 전달하고, INIT 액션에 대한 동작을 구현해 봅시다!

```
export default function mbti(state = initStateEmpty, action) {  
  switch (action.type) {  
    case INIT:  
      return {  
        ...state,  
        survey: action.payload.survey,  
        explanation: action.payload.explanation,  
      };  
  }  
}
```

Src/store/modules/Show.js



만들어진 Data
전달하기!



Data 전달하기!

- 만들어진 Data 를 dispatch 를 통해 init() 함수에 담아서 전달해 봅시다!

```
import { next, init } from '../store/modules/mbti';
```




```
import { next, init } from '../store/modules/mbti';

// 중간 생략

// survey 값을 위한 JOIN Table 의 데이터 받아오기
const resSurvey = await fetch('http://localhost:4000/data/survey');
if (resSurvey.status === 200) {
  const surveyData = await resSurvey.json();

  const resExplanation = await fetch(
    'http://localhost:4000/data/explanation'
  );
  if (resExplanation.status === 200) {
    const explanationData = await resExplanation.json();
    const madeData = makeData(surveyData, explanationData);
    dispatch(init(madeData));
  } else {
    throw new Error('통신 이상');
  }
} else {
  throw new Error('통신 이상');
}
}
```

Src/component/Start.js



```
useEffect(() => {
  async function fetchData() {
    const resCount = await fetch('http://localhost:4000/data/count');
    if (resCount.status === 200) {
      const num = await resCount.json();
      if (num[0].counts !== 0) setCounts(num[0].counts);
    } else {
      throw new Error('통신 이상');
    }
  }

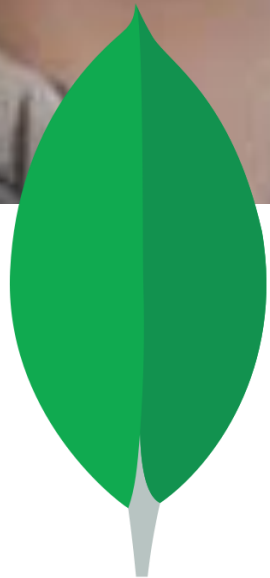
  const resSurvey = await fetch('http://localhost:4000/data/survey');
  if (resSurvey.status === 200) {
    const surveyData = await resSurvey.json();

    const resExplanation = await fetch(
      'http://localhost:4000/data/explanation'
    );
    if (resExplanation.status === 200) {
      const explanationData = await resExplanation.json();
      const madeData = makeData(surveyData, explanationData);
      dispatch(init(madeData));
    } else {
      throw new Error('통신 이상');
    }
  } else {
    throw new Error('통신 이상');
  }
}

fetchData();
}, []);
```

Src/component/Start.js 중 useEffect 전체 코드





mongoDB[®]



MongoDB로 구현하기

MongoDB와 React를 같이 사용해 봅시다!



- 이제 DB 를 MySQL 이 아닌 MongoDB 를 사용해 봅시다!



MongoDB

모듈 설치 및 세팅



MongoDB 모듈 설치

- Npm install mongodb

```
tetz@DESKTOP-P7Q4OLL MINGW64 /d/git2/developer-mbti/data-server (main)
$ npm i mongodb
npm WARN config global `--global`, `--local` are deprecated. Use `--location`
added 15 packages, and audited 1641 packages in 3s

221 packages are looking for funding
  run `npm fund` for details

6 high severity vulnerabilities
```




MongoDB 접속 모듈 생성

- mongoConnect.js 에 Mongodb 에 접속하기 위한 모듈을 만들어 봅시다!

```
const { MongoClient, ServerApiVersion } = require('mongodb');

const DB_URI = 'mongodb://localhost:27017';
const DB_URI_ATLAS =
  'mongodb+srv://tetz:qwer1234@cluster0.sdiakr0.mongodb.net/?retryWrites=true&w=majority';

const uri = DB_URI_ATLAS;

const client = new MongoClient(uri, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
  serverApi: ServerApiVersion.v1,
});

module.exports = client;
```

Data-server/mongoConnect.js



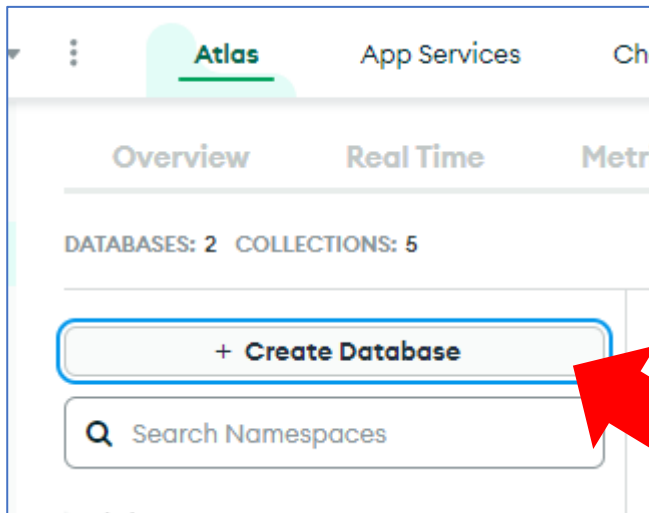
MongoDB

데이터 베이스 만들기!

Atlas 에 접속하여 mbti 용 DB를 만듭시다!



- Atlas 에 접속하여 mbti 용 DB 를 구축해 봅시다!

A screenshot of the 'Create Database' dialog box in MongoDB Atlas. The 'Database name' field contains 'mbti'. The 'Collection name' field contains 'counts'. Under 'Additional Preferences', there are two unchecked checkboxes: 'Capped Collection' and 'Time Series Collection'. At the bottom right, there are 'Cancel' and 'Create' buttons. A red arrow points to the 'Create' button from the right.



방문자 확인용 Collection

- 먼저 방문자 수 확인용 collection 에 document 를 추가해 봅시다!

mbti.counts

STORAGE SIZE: 4KB LOGICAL DATA SIZE: 0B TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE: 4KB

Find

Indexes

Schema Anti-Patterns 0

Aggregation

Search Indexes ●

FILTER { field: 'value' }

▶ OPTIMIZE

Apply

Reset

INSERT DOCUMENT

QUERY RESULTS: 0

Insert to Collection counts

VIEW

{ } ≡

1 _id: 634bde723a75d299fee9edfd
2 counts: 1

ObjectId
Int32

Cancel

Insert



Mbti 설문용 Collection

- 설문용 Data 는 Collection 을 만들고 하나하나 넣으면 될까요?
- 아니죠! 우리는 Redux 에 객체 형태로 데이터를 가지고 있으므로, 해당 데이터를 mogodb 의 insert 기능을 이용하여 바로 넣어 주면 됩니다!
- 테이블을 별도로 만드는 과정을 할 필요가 없고, JS의 객체와 mongodb 의 데이터가 바로 연동 되는 부분이 편리한 점 입니다! 😊
 - 물론 단점도 존재 합니다!



MongoDB

Controller 생성



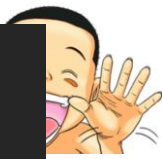
MongoDB 용 Controller 만들기!

- Mongodb 에서 데이터를 받아서 처리해줄 Controller 도 만들어 봅시다
- 먼저, 설문용 데이터를 DB에 저장 시켜주는 controller 부터 만들어 봅시다!
- 설문용 데이터는 Redux 에서 객체를 바로 가져 오면 됩니다!



```
// 초기 상태 설정
const initState = {
  mbtiResult: '',
  page: 0, // 0: 인트로 페이지, 1 ~ n: 선택 페이지, n+1: 결과 페이지
  survey: [ ...
  ],
  explanation: { ...
  },
};
```

kdtTetz, 14시간 전 • 수업 코드 수정 Data-server/controllers/mongoController.js



```
const mongoClient = require('../mongoConnect');
const _client = mongoClient.connect();

const mongoDB = {
  setData: async () => {
    const client = await _client;
    // collection 이 없으면 알아서 생성
    const db = client.db('mbti').collection('data');
    // redux 의 데이터를 그대로 mongodb 에 삽입
    const result = await db.insertOne(initState);
    if (result.acknowledged) {
      return '업데이트 성공';
    } else {
      throw new Error('통신 이상');
    }
  },
};
```

```
module.exports = mongoDB;
```

Data-server/controllers/mongoController.js



MongoDB 용 라우터 설정



MongoDB 용 라우터 설정

- 이제 mongoDB 에 데이터를 요청하는 주소 라우팅 설정을 해봅시다!
- 메인 서버에 **/mongo** 라는 주소 값을 처리하는 라우터 넣어주기
- Mongo.js 라는 라우터 모듈 생성



메인 서버 설정

```
const express = require('express');
const cors = require('cors');
const PORT = 4000;

const app = express();

app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cors());

const dataRouter = require('./routes/data');
const mongoRouter = require('./routes/mongo');
app.use('/data', dataRouter);
app.use('/mongo', mongoRouter);

app.listen(PORT, () => {
  console.log(`데이터 통신 서버가 ${PORT}에서 작동 중`);
});
```

Data-server/server.js



Mongo.js 라우터 구현

```
const express = require('express');
const router = express.Router();

const mongoDB = require('../controllers/mongoController');

router.post('/setdata', async (req, res) => {
  const msg = await mongoDB.setData();
  res.send(JSON.stringify(msg));
});

module.exports = router;
```

Data-server/routes/mongo.js



POSTMAN으로 요청 보내기



POSTMAN 으로 요청 보내기

- 이제 POST 방식으로 <http://localhost:4000/mongo/setdata> 에 요청을 보내 봅시다!



HomeWorkspaces ▾API Network ▾Explore

My Workspace

NewImport

Collections

APIs

Environments

Mock Servers

Monitors

Flows

History

+

☰

...

▼ New Collection

GET localhost:4000/users

POST localhost:4000/users

PUT localhost:4000/users

DEL localhost:4000/users

POST localhost:4000/posts

PUT localhost:4000/posts

DEL localhost:4000/posts

GET localhost:4000/users ●

POST localhost:4000/users ●

PUT localhost:4000/users

New Collection / localhost:4000/posts

POST ▾localhost:4000/mongo/setdata

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTest

Query Params

	KEY
	Key

BodyCookiesHeaders (8)Test Results

PrettyRawPreviewVisualizeHTML ▾

⌵

1

"업데이트 성공"



Overview Real Time Metrics Collections

DATABASES: 2 COLLECTIONS: 5

+ Create Database

Q Search Namespaces

▶ kdt1

▼ mbti

counts

| data

mbti.data

STORAGE SIZE: 20KB

Find

Indexe

FILTER

{ field: 'va

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('634be9aa42bdff49301db252')
mbtiResult: ""
page: 0
survey: Array
  0: Object
    question: "퇴근 직전에 동료로부터 개발자 모임에 초대를 받은 나!
           퇴근 시간에 나는?"
    answer: Array
      0: Object
        text: "그런 모임을 왜 이제서야 알려 준거야! 당장 모임으로 출발한다"
        result: "E"
      1: Object
        text: "1년 전에 알려줬어도 안갔을 건데 뭐... 더 빠르게 집으로 간다"
        result: "I"
      1: Object
      2: Object
      3: Object
    explanation: Object
```

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('634be9aa42bdff49301db252')
mbtiResult: ""
page: 0
> survey: Array
> explanation: Object
```



컨트롤러 기능 추가



컨트롤러 기능 추가!

- 이제 기존 MySQL 에서 하던 기능들을 컨트롤러에 추가해 봅시다!
- 방문자 수를 가져오는 getCounts
- 방문자 수를 증가 시키는 incCounts
- 데이터를 받아오는 getData

getCounts() 구현



```
getCounts: async () => {  
  const client = await _client;  
  const db = client.db('mbti').collection('counts');  
  const data = await db.find({}).toArray();  
  return data;  
},
```

Data-server/controllers/mongoController.js



incCounts() 구현

```
incCounts: async () => {  
  const client = await _client;  
  const db = client.db('mbti').collection('counts');  
  const result = await db.updateOne({ id: 1 }, { $inc: { counts: +1 } });  
  if (result.acknowledged) {  
    return '업데이트 성공';  
  } else {  
    throw new Error('통신 이상');  
  }  
},
```

Data-server/controllers/mongoController.js

getData() 구현



```
getData: async () => {  
  const client = await _client;  
  const db = client.db('mbti').collection('data');  
  const data = await db.find({}).toArray();  
  return data;  
},
```

Data-server/controllers/mongoController.js



Data-server/controllers/mongoController.js

전체 코드(initState 데이터는 제외)

```
const mongoClient = require('../mongoConnect');
const _client = mongoClient.connect();

const mongoDB = {
  getCounts: async () => {
    const client = await _client;
    const db = client.db('mbti').collection('counts');
    const data = await db.find({}).toArray();
    return data;
  },
  incCounts: async () => {
    const client = await _client;
    const db = client.db('mbti').collection('counts');
    const result = await db.updateOne({ id: 1 }, { $inc: { counts: +1 } });
    if (result.acknowledged) {
      return '업데이트 성공';
    } else {
      throw new Error('통신 이상');
    }
  },
  setData: async () => {
    const client = await _client;
    const db = client.db('mbti').collection('data');
    const result = await db.insertOne(initState);
    if (result.acknowledged) {
      return '업데이트 성공';
    } else {
      throw new Error('통신 이상');
    }
  },
  getData: async () => {
    const client = await _client;
    const db = client.db('mbti').collection('data');
    const data = await db.find({}).toArray();
    return data;
  },
};

module.exports = mongoDB;
```



mongoRouter

설정

mongoRouter 설정



- 컨트롤러에 추가된 기능을 사용하는 주소 값을 라우터에 추가해 봅시다!



```
const express = require('express');
const router = express.Router();
const mongoDB = require('../controllers/mongoController');
```

```
router.post('/setdata', async (req, res) => {
  const msg = await mongoDB.setData();
  res.send(JSON.stringify(msg));
});
```

```
router.get('/count', async (req, res) => {
  const counts = await mongoDB.getCounts();
  res.send(counts);
});
```

```
router.post('/inccount', async (req, res) => {
  const msg = await mongoDB.incCounts();
  res.send(JSON.stringify(msg));
});
```

```
router.get('/getdata', async (req, res) => {
  const data = await mongoDB.getData();
  res.send(data);
});
```

```
module.exports = router;
```

Data-server/routes/mongo.js



React 에서 데이터 요청 및 사용!



MySQL 데이터 관련 코드 정리!

- Start 컴포넌트에서 MySQL 로 부터 데이터를 받아오는 코드는 이제 하나의 함수로 묶어서 정리 합시다!



```
async function sqlFetchData() {
  const resCount = await fetch('http://localhost:4000/data/count');
  if (resCount.status === 200) {
    const num = await resCount.json();
    if (num[0].counts !== 0) setCounts(num[0].counts);
  } else {
    throw new Error('통신 이상');
  }

  const resSurvey = await fetch('http://localhost:4000/data/survey');
  if (resSurvey.status === 200) {
    const surveyData = await resSurvey.json();
    console.log(surveyData);
    const resExplanation = await fetch(
      'http://localhost:4000/data/explanation'
    );
    if (resExplanation.status === 200) {
      const explanationData = await resExplanation.json();
      const data = makeData(surveyData, explanationData);
      dispatch(init(data));
    } else {
      throw new Error('통신 이상');
    }
  } else {
    throw new Error('통신 이상');
  }
}
```

Src/component/Start.js



이제 필요할 때 호출해서 사용!

- useEffect HOOK 에 이제 필요할 때 호출해서 사용 합시다!

```
useEffect(() => {  
  // sqlFetchData();  
  mongoFetchData();  
}, []);
```

Src/component/Start.js



MongoDB용

코드 작성



MongoDB 의 데이터를 받는 함수 작성

- `sqlDataFetch()` 함수와 같이, `mongoFetchData()` 함수를 만들어서 백에서 작성한 코드에서 데이터가 잘 넘어오는지 확인해 봅시다!



```
async function mongoFetchData() {  
  const resMongoCount = await fetch('http://localhost:4000/mongo/count');  
  if (resMongoCount.status === 200) {  
    const num = await resMongoCount.json();  
    // counts 가 제대로 들어오는지 확인하기  
    console.log(num);  
  } else {  
    throw new Error('통신 이상');  
  }  
  const resMongoData = await fetch('http://localhost:4000/mongo/getdata');  
  if (resMongoData.status === 200) {  
    const data = await resMongoData.json();  
    // 설문용 data 가 제대로 들어오는지 확인하기  
    console.log(data);  
  } else {  
    throw new Error('통신 이상');  
  }  
}
```

Src/component/Start.js



들어온 데이터 확인!

- 전달된 데이터는 배열에 객체가 들어있는 형태로 들어오며, 0번 index 의 값에 우리가 원하는 데이터가 들어 있음을 확인!

```
▼ [{-}] ⓘ  
  ▼ 0:  
    counts: 8  
    id: 1  
    _id: "634bde723a75d299fee9edfd"  
    ▶ [[Prototype]]: Object  
    length: 1  
    ▶ [[Prototype]]: Array(0)  
  
▼ [{-}] ⓘ  
  ▼ 0:  
    ▶ explanation: {ESTJ: {-}, ISTJ: {-}, ENTJ: {-}, INTJ: {-}, ESFJ: {-}, ...}  
    mbtiResult: ""  
    page: 0  
    ▶ survey: (4) [{-}, {-}, {-}, {-}]  
    _id: "634cb653a123bd5eeafc1f35"  
    ▶ [[Prototype]]: Object  
    length: 1  
    ▶ [[Prototype]]: Array(0)
```



방문자수 값

전달



Counts 값 전달!

- Counts 는 값을 새로 받으면, Start 컴포넌트의 방문자 수 값을 Update 해줘야 하므로 useState 로 값을 선언 → State 값을 변경하는 Hook 에 원하는 값을 전달!



```
async function mongoFetchData() {
  const resMongoCount = await fetch('http://localhost:4000/mongo/count');
  if (resMongoCount.status === 200) {
    const num = await resMongoCount.json();
    if (num[0].counts !== 0) setCounts(num[0].counts);
  } else {
    throw new Error('통신 이상');
  }
  const resMongoData = await fetch('http://localhost:4000/mongo/getdata');
  if (resMongoData.status === 200) {
    const data = await resMongoData.json();
    console.log(data);
    if (data[0].survey.length !== 0) {
      dispatch(init(data[0]));
    }
  } else {
    throw new Error('통신 이상');
  }
}
```

Src/component/Start.js



Redux Initial

Data 전달



Redux Initial Data 전달

- Redux 의 InitialData 는 바로 사용할 일이 없으며, 다음 MbtI 컴포넌트가 사용하면 되므로 State 값을 사용할 필요가 X
- 기존에 만들어 놓은 init() 액션 생성 함수에 받아온 데이터만 넣어서 dispatch 로 전달하면 끝!



```
async function mongoFetchData() {  
  const resMongoCount = await fetch('http://localhost:4000/mongo/count');  
  if (resMongoCount.status === 200) {  
    const num = await resMongoCount.json();  
    if (num[0].counts !== 0) setCounts(num[0].counts);  
  } else {  
    throw new Error('통신 이상');  
  }  
  const resMongoData = await fetch('http://localhost:4000/mongo/getdata');  
  if (resMongoData.status === 200) {  
    const data = await resMongoData.json();  
    console.log(data);  
    // 리액트의 기본 동작으로 통신 전 빈 데이터가 전달 되는 것을 예방  
    if (data[0].survey.length !== 0) {  
      dispatch(init(data[0]));  
    }  
  } else {  
    throw new Error('통신 이상');  
  }  
}
```

Src/component/Start.js


```
useEffect(() => {  
  // sqlFetchData();  
  mongoFetchData();  
}, []);
```

Src/component/Start.js





YAY WE DID

**WORK TOGETHER, BELIVE IN
YOURSELF AND YOUR GROUP
MEMBER**

makeameme.org



수고하셨습니다!