데이터베이스

- 1. 데이터베이스란?
- 2. 어떤 데이터베이스를 사용할까
  관계형 데이터베이스
- 3. 실습! 데이터베이스를 사용하자

데이터베이스란?

데이터베이스란?

- 사전적 정의?
  Organized Collection of Data.
  체계적으로 데이터를 정리한 목록

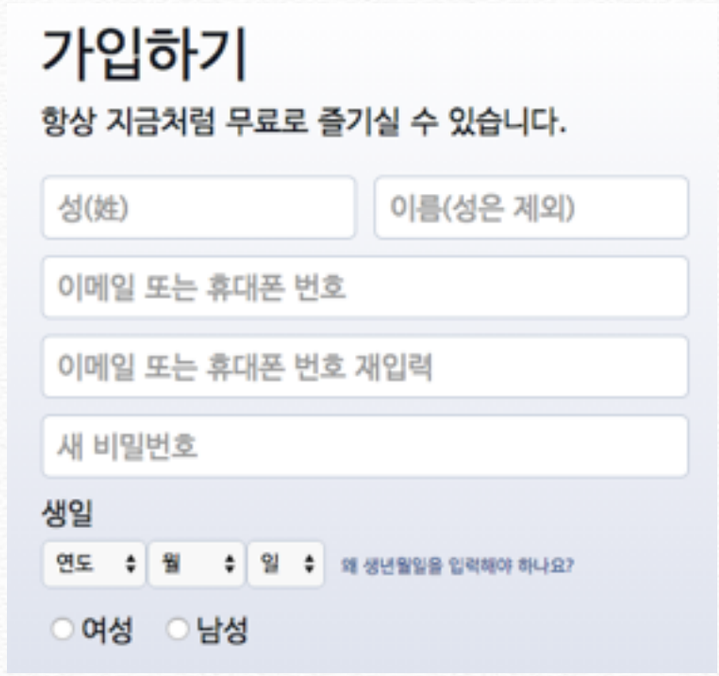
- 일상 생활에서의 예 - 표로 정리되어 있는 모든 것들!
  ex) 실험 결과, 수강편람, 성적표...

- 그렇다면 우리가 만들 웹서비스에서는 어떻게 사용될까?

웹서비스에서는…

- 웹 서비스들에서는 직접적으로 데이터베이스가 드러나진 않는다.
- 하지만 기능을 살펴보면 데이터베이스에 어떤 내용이 있는지 생각해 볼 수 있다.

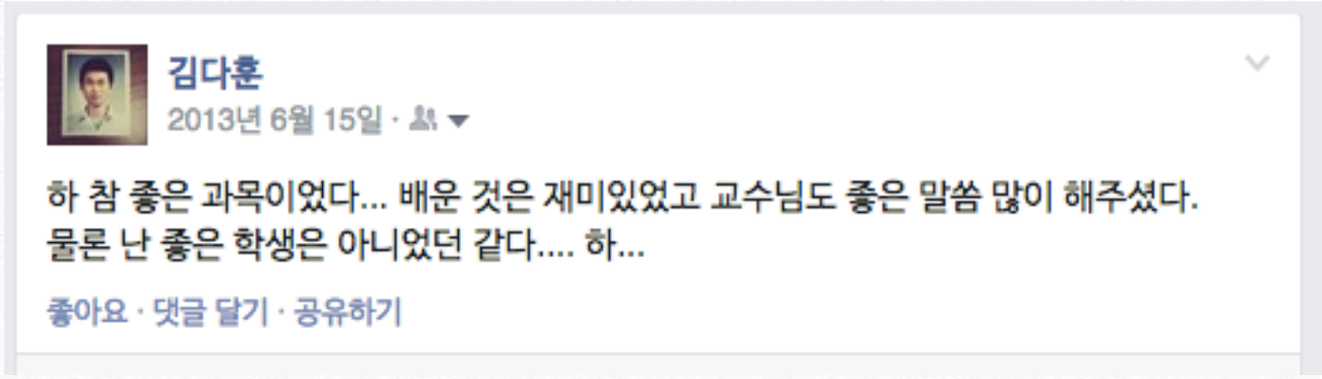
페이스북을 예로 들어



사용자

성	이름	이메일	휴대폰 번호	비밀번호	생일	성별
김	다훈	salqueng@gmail.com	010-xxxx-xxxx	xxxxxxx	1992/06/03	남
주	현탁	momamene@gmail.com	010-xxxx-xxxx	xxxxxxxx	1991/10/04	남

페이스북을 예로 들어 (2)



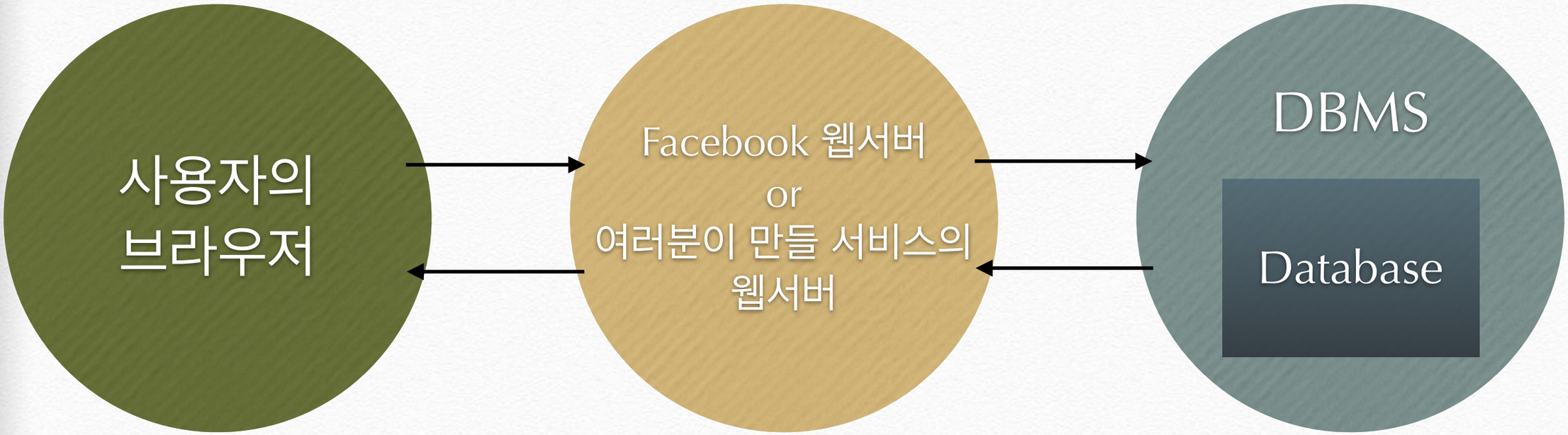
포스트(Post)

작성자	작성일	공개범위	내용
김다훈	2013-06-15	친구만	하 좋은 과목이었다…
주현탁	2015-01-01	전체공개	피로그래밍은 왜 피로그래밍인가요?!

데이터베이스로 할 수 있는 일

- 항목 추가하기 (사용자 추가, 포스트 추가)
- 항목 삭제하기 (회원 탈퇴, 포스트 지우기)
- 항목 수정하기
- 특정한 조건을 만족하는 항목 찾기 (검색, 친구의 포스팅 찾기)
- 표 만들기, 수정하기, 지우기
- 이 모든 것들을 직접 구현하는 건 어렵다. 그리고 필요없다! -> DBMS (Database Management System)가 알아서 해준다.

큰 그림



어떤 데이터베이스를 사용할까?

관계형 데이터베이스

관계형 데이터베이스

- 데이터베이스 내에 여러 표들이 있고 이들 사이에 관계가 존재
- 대부분의 서비스들은 관계형 데이터베이스를 이용!
- 표 = 테이블(Table)
- 테이블 아래의 각 항목 = 레코드(Record)
- 테이블의 각 열 = 필드(Field)

사용자

성	이름	이메일	휴대폰 번호	비밀번호	생일	성별
김	다훈	salqueng@gmail.com	010-xxxx-xxxx	xxxxxxx	1992/06/03	남
주	현탁	momamene@gmail.com	010-xxxx-xxxx	xxxxxxxx	1991/10/04	남

포스트(Post)

작성자	작성일	공개범위	내용
김다훈	2013-06-15	친구만	하 좋은 과목이었다…
주현탁	2015-01-01	전체공개	피로그래밍은 왜 피로그래밍인가요?!
김다훈	2015-01-06	친구만	아 점심먹었는데 배고프다

포스트 테이블에는 어떤 사용자가 작성자인지 저장되어있다.
사용자와 포스트는 1:N 관계

그런데 작성자의 이름을 포스트에 두는 것이 괜찮을까?
(동명이인이 있다면?)
사용자 테이블에 각 항목마다 고유한 번호를 두고, 이를 포스트에서 사용!

사용자

번호	성	이름	이메일	휴대폰 번호	비밀번호	생일	성별
1000	김	다훈	salqueng@gmail.com	010-xxxx-xxxx	xxxxxxx	1992/06/03	남
1001	주	현탁	momamene@gmail.com	010-xxxx-xxxx	xxxxxxxx	1991/10/04	남
1002	김	다훈	kim@tntcrowd.com	010-xxxx-xxxx	xxxxxxx	1999/09/09	여

포스트(Post)

번호	작성자	작성일	공개범위	내용
123456	1000	2013-06-15	친구만	하 좋은 과목이었다…
123457	1001	2015-01-01	전체공개	피로그래밍은 왜 피로그래밍인가요?!
123458	1000	2015-01-06	친구만	아 점심먹었는데 배고프다

- 기본키(Primary Key)
  테이블에서 레코드를 식별하는 고유한 필드
  테이블 마다 하나씩만 존재!
  ex) 순서대로 매긴 번호, 학번, 주민등록번호

- 외부키(Foreign Key)
  다른 테이블 레코드를 가리키는 필드 (ex. 포스트에서 작성자)

Q. M:N 관계는 어떻게 표현할까 (사람 - 과일 (좋아하는 과일))

번호	이름
1	김다훈
2	주현탁
3	박태영
4	

번호	이름
1	사과
2	배
3	귤
4	자몽

번호	사람	과일
1	1	3
2	1	4
3	2	1
4	2	2
5	2	3

실습!

데이터베이스를 사용하자

PostgreSQL

- 우리가 사용할 DBMS
- 관계형 데이터베이스를 관리
- 다루기 위해 SQL 이라는 언어를 사용
- 그러나 SQL을 직접적으로 사용하는 일은 없을 예정!

SQLAlchemy

- SQL + 연금술
  SQL을 직접 사용할 필요없이 Python으로 DBMS를 사용할 수 있다

- PostgreSQL, Flask와 같이 사용하기 좋다!

- http://docs.sqlalchemy.org/en/rel_0_9/orm/tutorial.html

- sudo pip install sqlalchemy pyscopg2 ipython

- 예제로 사용자 + 주소를 만들어보자

Database 만들기 & 연결하기

- “test”라는 이름의 데이터베이스 만들기

```
$ createdb test
```

- python에서 PostgreSQL 연결하기

```
$ ipython
>>> from sqlalchemy import create_engine
>>> engine = create_engine('postgresql://localhost:
5432/test')
```

테이블 만들기

```
In [34]: from sqlalchemy.ext.declarative import declarative_base

In [35]: Base = declarative_base()

In [36]: from sqlalchemy import Column, Integer, String

In [37]: class User(Base):
    ....:     __tablename__ = 'users'
    ....:     id = Column(Integer, primary_key=True)
    ....:     name = Column(String)
    ....:     fullname = Column(String)
    ....:     password = Column(String)
    ....:     def __repr__(self):
    ....:         return "<User(name='%s', fullname='%s', password='%s')>" % (
    ....:         self.name, self.fullname, self.password)
    ....:

In [39]: Base.metadata.create_all(engine)
```

users

id (primary key)	name (문자열)	fullname (문자열)	password (문자열)

세션(Session)

- 테이블에 레코드를 추가, 삭제, 수정, 검색을 도와주는 통로
- commit을 하기 전까지는 데이터베이스에 반영되지 않음
- rollback을 하면 변경사항이 반영되지 않고 초기화
- 세션 만들기
  from sqlalchemy.orm import session maker
  Session = sessionmaker(bind=engine)
  session = Session()

추가하기

- 사용자 만들기

  new_user = User(name=‘dahoon’, fullname-=‘Dahoon Kim’, password=‘asdf’)

- 세션에 만든 사용자 추가하기

  session.add(new_user)

추가하기 (2)

- 한번에 여러명 추가하기

```
session.add_all([
    User(name='wendy', fullname='Wendy Williams', password='foobar'),
    User(name='mary', fullname='Mary Contrary', password='xxg527'),
    User(name='fred', fullname='Fred Flinstone', password='blah')
])
```

- 세션을 적용

```
session.commit()
```

검색하기

- 사용자 목록 얻기

  user_list = session.query(User).order_by(User.id)
  사용자 목록을 id 순서대로 얻어라

- 사용자 목록에서 사용자를 하나씩 얻기

  for user in user_list:
      print(user)

  앞에서 추가한 사용자의 목록이 잘 나와야 합니다!

검색하기 (2)

- 특정 조건을 만족하는 사용자 찾기

```
user_filter_list = session.query(User).filter(name='dahoon')
```

or

```
user_filter_list = session.query(User).filter_by(User.name == 'dahoon')
```

filter 또는 filter_by 의 안에 검색 조건을 넣는다.

- 이들 중에서 첫번째 사용자 얻기

```
filtered_user = user_filter_list.first()
```

- 이들 중에서 하나만 얻기 (조건을 만족하는게 여러개 있으면 에러가 발생해요!)

```
filtered_user = user_filter_list.one()
```

- 자세한 filter, filter_by 사용법은 튜토리얼 참고!

http://docs.sqlalchemy.org/en/rel_0_9/orm/tutorial.html#common-filter-operators

수정하기, 삭제하기

- filtered_user를 수정하자

```
filtered_user.password = 'new_password'

session.commit()
```

변경을 한 후에는 commit을 해야 반영된다!

- filtered_user를 삭제하자

```
session.delete(filtered_user)
session.commit()
```

1:N 관계

```
In [40]: from sqlalchemy import ForeignKey

In [41]: from sqlalchemy.orm import relationship, backref

In [42]: class Address(Base):
    ....:     __tablename__ = 'addresses'
    ....:     id = Column(Integer, primary_key=True)
    ....:     email_address = Column(String, nullable=False)
    ....:     user_id = Column(Integer, ForeignKey('users.id'))
    ....:     user = relationship("User", backref=backref('addresses', order_by=id))
    ....:     def __repr__(self):
    ....:         return "<Address(email_address='%s')>" % self.email_address
    ....:

In [43]: Base.metadata.create_all(engine)
```

addresses

id	email_address	user_id

user_id는 Foreign Key이며
이 값을 이용하여 address와 연결된 user를 찾는다.

- 사용자를 만들고 주소를 추가하자

```
>>> jack = User(name='jack', fullname='Jack Bean',
password='gjffdd')
>>> jack.addresses
[] (아무것도 없다)
>>> jack.addresses = [
    Address(email_address='jack@google.com'),
    Address(email_address='j25@yahoo.com')
]
>>> session.add(jack)
>>> session.commit()
```

- 사용자와 주소가 잘 추가되었는지 확인해보자!

```
>>> jack =
session.query(User).filter(name='jack').one()
>>> jack.addresses
```

(주소가 두 개 있어야 합니다!)

users

id (primary key)	name (문자열)	fullname (문자열)	password (문자열)
(jack의 id)	jack	Jack Bean	gjffdd

addresses

id	email_address	user_id
(어떤 id)	jack@google.com	(jack의 id)
(어떤 id)	j25@yahoo.com	(jack의 id)

Join

- 두 테이블의 레코드를 합치는 연산(?)
- 필요한 경우를 찾아보자?
  위의 사용자 + 주소 예제에서 “이메일이 jack@google.com인 사용자를 찾기”?

Join (2)

- 방법 1: 이메일이 일치하는 주소를 찾고, 주소로부터 사용자를 얻어내자!

```
>>> address = session.query(Address).filter(email_address='jack@google.com').one()
>>> address.user
```

- 더 좋은 방법이 있다? Join을 이용하자!

Join (3)

- 방법 2: Join을 이용하여 users 테이블에 address 테이블을 합치고, email_address가 'jack@google.com'인 사용자를 바로 찾자.

- ```
  >>>
  session.query(User).join(Address).filter(Address.
  email_address == 'jack@google.com').one()
  ```

- 이 방법이 더 빠르다.
  ```