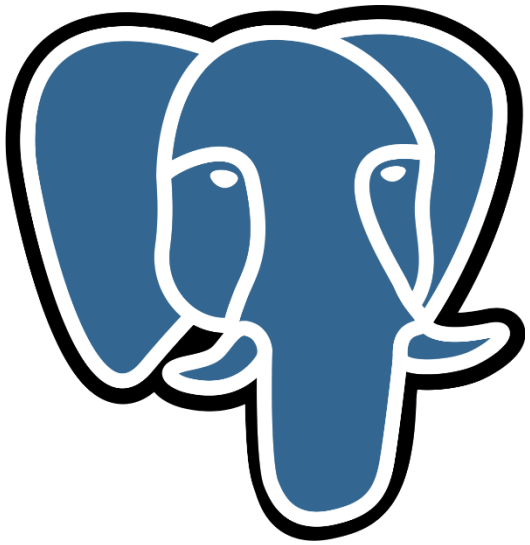
SQL Queries

Database Systems
DataLab, CS, NTHU
Spring, 2024



PostgreSQL

- Download and install
- For Mac users, try PostgreSQL.app

Using PostgreSQL

```
$ createdb <db>
$ psql <db> [user]
> \h or \?
> SELECT now(); -- SQL commands
```

- Multiple lines until ‘;’
- ‘--’ for comments
- ***Case insensitive***
  - Use “” to distinguish lower and upper cases
  - E.g., `SELECT "authorId" FROM posts;`

Structured Query Language (SQL)

- Data Definition Language (DDL) on schema
  - CREATE TABLE …
  - ALTER TABLE …
  - DROP TABLE …
- Data Manipulation Language (DML) on records
  - INSERT INTO … VALUES …
  - SELECT … FROM … WHERE …
  - UPDATE … SET … WHERE …
  - DELETE FROM … WHERE …

Creating Tables/Relations

```
CREATE TABLE users (
  id              serial PRIMARY KEY NOT NULL,
  name            varchar(50) NOT NULL,
  Karma           integer NOT NULL
);
```

- Column types:
  - Integer, bigint, real, double, etc.
  - varchar(10), text, etc.
- Non-null constraint

Creating Tables/Relations

```
CREATE TABLE users (
  id              serial PRIMARY KEY NOT NULL,
  name            varchar(50) NOT NULL,
  Karma           integer NOT NULL
);
```

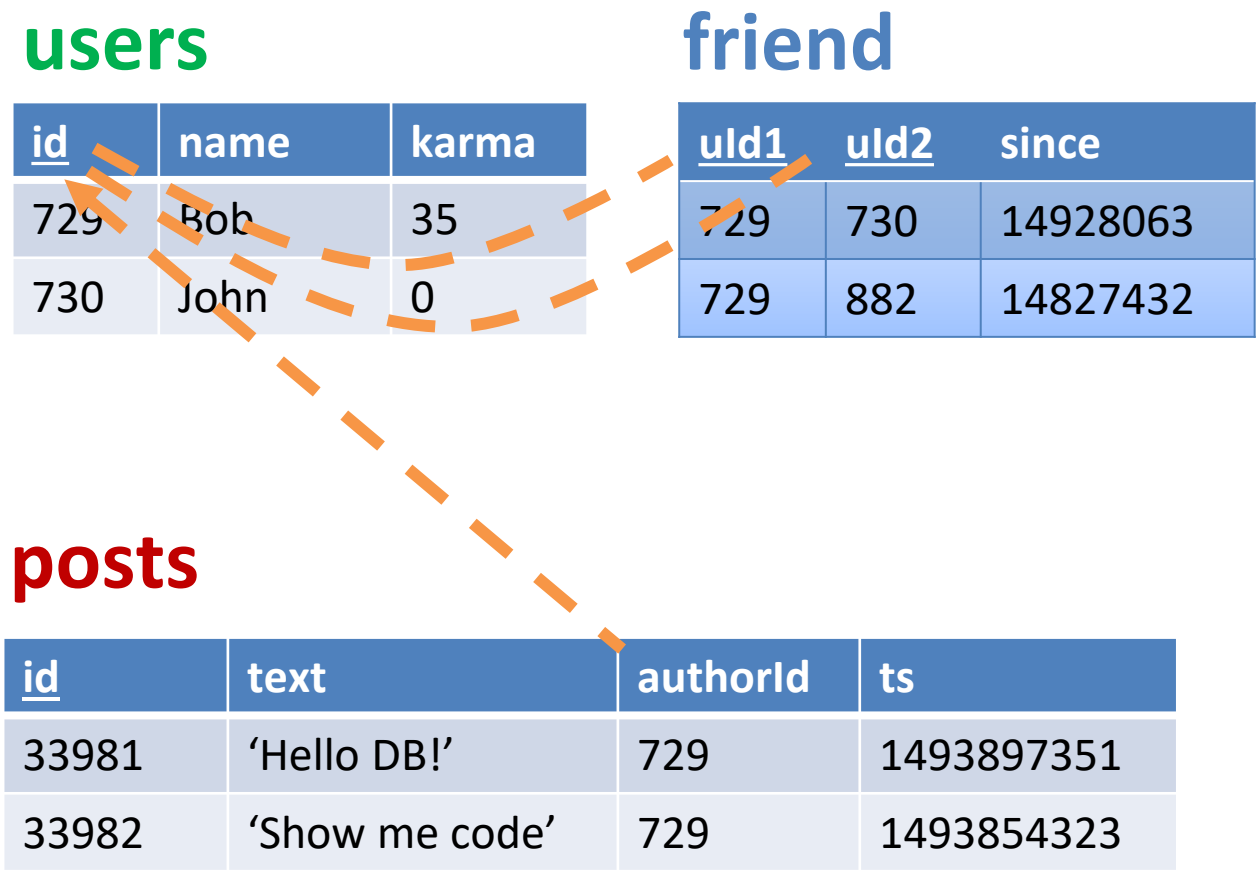
- Primary key:
  - Unique (no duplicate values among rows)
  - Usually of type “serial” (auto-filled integer)
  - Index automatically created

Creating Tables/Relations

```
CREATE TABLE posts (
  id              serial PRIMARY KEY NOT NULL,
  text            text NOT NULL,
  "authorId"      integer NOT NULL
                  REFERENCES users ON DELETE CASCADE,
  ts              bigint NOT NULL
                  DEFAULT (extract(epoch from now()))
);
```

- Foreign key: posts.authorId must be a valid users.id
- When deleting a user (row):
  - NO ACTION (default): user not deleted, error raised
  - CASCADE: user ***and all referencing posts*** deleted

Schema


users
friend
id
name
karma
729
Bob
35
730
John
0
uId1
uId2
since
729
730
14928063
729
882
14827432
posts
id
text
authorId
ts
33981
'Hello DB!'
729
1493897351
33982
'Show me code'
729
1493854323

Queries

```
SELECT * FROM users;
```

- Aggregate function:

```
SELECT COUNT(*) FROM users;

SELECT AVG(karma) FROM users;

SELECT MIN(karma) FROM users;
```

- Often used with the GROUP BY

Queries

```
SELECT *
FROM users
WHERE id<5 AND name ILIKE '%User%'
ORDER BY id DESC
LIMIT 2;
```

- To see how a query is processed:

```
EXPLAIN ANALYZE -- show plan tree
SELECT *
FROM users
WHERE id<5 AND name ILIKE '%User%'
ORDER BY id DESC
LIMIT 2;
```

(Batch) Updating Rows

```
UPDATE users SET karma = karma + 10 WHERE name =
'Bob';
```

- ***All*** rows satisfying the WHERE clause will be updated

Assigned Reading

- SQL Tutorial