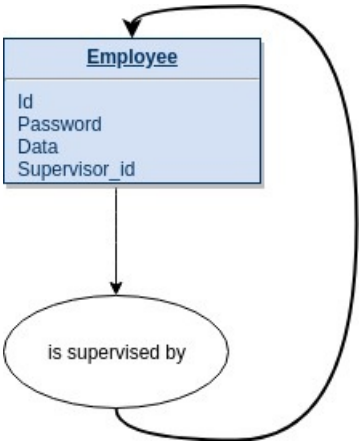
Model konceptualny

Wojciech Fica

W bazie danych *student* będzie jedna tabela - *employee*.
Implementuję wersje zadania 0 - częste zapytania o hierarchię pracowników.



Automatycznie wygenerowany opis tabeli

Table "public.employee"

Column	Type	Modifiers
id	integer	not null
passwd	text	not null
data	text	
supervisor_id	integer	

Indexes:
"employee_pkey" PRIMARY KEY, btree (id)

Foreign-key constraints:
"employee_supervisor_id_fkey" FOREIGN KEY (supervisor_id) REFERENCES employee(id) ON DELETE CASCADE

Referenced by:
TABLE "employee" CONSTRAINT "employee_supervisor_id_fkey" FOREIGN KEY (supervisor_id) REFERENCES employee(id) ON DELETE CASCADE

Uprawnienia użytkowników bazy danych *student*

Uprawnienia użytkownika *init* są nadawane przez *superuser* poleceniem:

```
GRANT ALL PRIVILEGES ON DATABASE student TO init
```

Uprawnienia użytkownika *app* są nadawane przez *init* poleceniem:

```
GRANT SELECT, DELETE, UPDATE, INSERT ON employee TO app
```

Komentarz do implementacji funkcji API

Program będzie napisany w Pythonie 3.5 z rozszerzeniem psycopg2. Hasła będą przechowywane w bazie danych z pomocą rozszerzenia **pgcrypto**.

Wywołanie w trybie *--init*:

Program wykona polecenia z wejścia dodając wpisy o nowych pracownikach.

Wywołanie w trybie *app*:

Program zanim przystąpi do wykonywania poleceń przejdzie po drzewie pracowników dfsem, dla każdego wierzchołka licząc czasy wejścia i wyjścia. Umożliwi to odpowiadanie w czasie stałym na pytania o hierarchię dwóch pracowników.

Wszystkie funkcje przed wykonaniem zmian do bazy danych sprawdzają zgodność wejścia ze specyfikacją. Dokonywana jest autentykacja *admina* wprowadzającego zmiany oraz sprawdzenie czy *admin* ma uprawnienia do wywołania funkcji z konkretnymi parametrami. Jeśli uprawnienia się zgadzają to zostanie wywołane odpowiednie zapytanie SQL i zwrócony wynik. W przypadku niepowodzenia nie jest podejmowana próba ponownego wywołania zapytania i zwracany jest komunikat { "status": "ERROR" }.