Технічні науки

УДК 004.054

Токарський Андрій Олегович
студент
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»
Токарский Андрей Олегович
студент
Национального технического университета Украины
«Киевский политехнический институт имени Игоря Сикорского»
Tokarskyi Andrii
Student of the
National technical university of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»

ОБ'ЄКТНО-ОРІЄНТОВАНІ ВЛАСТИВОСТІ БАЗИ ДАНИХ POSTGRESQL
ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ СВОЙСТВА БАЗЫ ДАННЫХ POSTGRESQL
THE OBJECT-ORIENTED PROPERTIES OF POSTGRESQL

Анотація. Огляд об'єктних можливостей бази даних PostgreSQL.
Ключові слова: SQL, Postgres, Дані, ООП, База даних.
Аннотация. Обзор объектных возможностей базы данных PostgreSQL.
Ключевые слова: SQL, Postgres, Дані, ООП, База данных.
Summary. The review of object-oriented abilities of PostgreSQL.
Key words: SQL, Postgres, Data, OOP, Database.

Однією із найпопулярніших вільних баз даних є об'єктно-реляційна база даних PostgreSQL. Не зважаючи на те, що ця БД дозволяє формувати

сховище даних на основі реляційної алгебри, є і можливість створити базу даних, що може в деякій мірі повторити об'єктну архітектуру програми [1].

Інкупсуляції, як такої, немає. Є лише можливість сховати стовпець таблиці при запиті типу "SELECT *".

Невід'ємною частиною об'єктно-орієнтованої архітектури є наслідування. Наслідування (inheritance) – це концепція ООП, згідно із якою тип даних може наслідувати дані і функціонал деякого існуючого типу.

Також існує і множинне наслідування (multiple inheritance) – це вид наслідування, при якому клас-потомок може мати більше одного батьківського класу. Так, у базі даних PostgreSQL є можливість для таблиць створювати таблиць-потомків. Розглянемо, яким чином це реалізовано у PostgreSQL.

Для цього створимо схему бази даних, заповнимо її таблиці даними та виконаємо ряд SELECT-запитів, як це зображено на рисунку 1.

```
1.
2.      --- Створення таблиць
3.      CREATE TABLE Human (
4.          firstName TEXT,
5.          lastName  TEXT
6.      );
7.
8.      CREATE TABLE Programmer (
9.          progLanguage TEXT
10.     ) INHERITS (Human); --- Таблиця Programmer наслідує Human
11.
12.
13.     - Заповнення даними
14.     INSERT INTO Programmer (firstName, lastName, progLanguage)
15.          VALUES ('Jon', 'Skeet', 'C#');
16.
17.     INSERT INTO Human(firstName, lastName)
18.          VALUES ('John', 'Smith');
19.
20.     INSERT INTO Programmer(firstName, lastName, progLanguage)
21.          VALUES ('Bill', 'Gates', 'Visual Basic');
22.
23.     INSERT INTO Human(firstName, lastName)
24.          VALUES ('Roman', 'Ivanov');
25.
26.     -- Виконання SELECT-запитів
27.
28.     SELECT * FROM Human; ---- #1
29.     SELECT * FROM Programmer; ---- #2
30.     SELECT * FROM ONLY Human; ---- #3
31.     SELECT * FROM ONLY Programmer; ---- #4
32.     SELECT Human.tableoid, ---- #5
33.            Human.*
34.       FROM Human;
35.
36.     ---- #6
37.     SELECT pg_class.oid,
38.            pg_class.relname,
39.            Human.*
40.       FROM Human
41.            JOIN pg_class
42.              ON pg_class.oid = Human.tableoid;
43.
44.     ---- #7
45.     SELECT * FROM pg_inherits;
```

Рис. 1. Набір запитів, що демонструє роботу наслідування в PostgreSQL

Результат виконання цього ряду запитів зображений на рисунку 2.

	firstname	lastname
1	John	Smith
2	Roman	Ivanov
3	Jon	Skeet
4	Bill	Gates

	firstname	lastname	proglanguage
1	Jon	Skeet	C#
2	Bill	Gates	Visual Basic

	firstname	lastname
1	John	Smith
2	Roman	Ivanov

	firstname	lastname	proglanguage
1	Jon	Skeet	C#
2	Bill	Gates	Visual Basic

	tableoid	firstname	lastname
1	1196801	John	Smith
2	1196801	Roman	Ivanov
3	1196807	Jon	Skeet
4	1196807	Bill	Gates

	oid	relname	firstname	lastname
1	1196801	human	John	Smith
2	1196801	human	Roman	Ivanov
3	1196807	programmer	Jon	Skeet
4	1196807	programmer	Bill	Gates

	inhrelid	inhparent	inhseqno
1	1196807	1196801	1

Рис. 2. Результат виконання тестових запитів

Із результатів, отриманих із виконання першого запиту, видно, що всі записи із таблиці Programmer одночасно знаходяться і в таблицю Human. Як видно із запиту 3, із допомогою ключового слова ONLY можна вибрати ті записи, які є у даній таблиці, але яких немає у жодній із потомків.

Запити 5 і 6 дозволяють визначити кінцеву таблицю, у якій знаходяться записи. Це здійснено із допомогою колонки tableoid, яку мають всі таблиці в PostgreSQL.

Таблиця pg_inherits в СУБД PostgreSQL зберігає в собі список усіх наслідувань. У нашому прикладі (результат виконання запиту №7) видно, що таблиця із ідентифікатором 1196807 – потомок таблиці із ідентифікатором 1196801.

Детальніше роботу таблиці pg_inherits можна розглянути на іншому прикладі. Розглянемо наступний SQL-код, зображений на рисунку 3.

```
1.
2.      CREATE TABLE Humanoid (
3.          humanoid_data TEXT
4.      );
5.
6.      CREATE TABLE AI (
7.          ai_data TEXT
8.      );
9.
10.     CREATE TABLE Human () INHERITS (Humanoid);
11.     CREATE TABLE Droid () INHERITS (Humanoid, AI);
12.     CREATE TABLE C3PO() INHERITS (Droid);
13.
14.     SELECT * FROM pg_inherits;
```

	inhrelid	inhparent	inhseqno
1	1196825	1196813	1
2	1196831	1196813	1
3	1196831	1196819	2
4	1196837	1196831	1

Рис. 3. Демонстрація роботи таблиці pg_inherits в PostgreSQL

У цьому прикладі також продемонстровано multiple inheritance. Є дві базових таблиці – Humanoid та AI. Human наслідується із Humanoid, а Droid – із Humanoid та AI (це приклад множинного наслідування). Також, в свою чергу, Droid є батьківською таблицею для C3PO.

Розглянемо колонки таблиці - pg_inherits. inhrelid та inhparent вказують, яка таблиця наслідується із якої відповідно. Якщо у таблиці-потомка є кілька батьківських таблиць, то число inhseqno визначає порядок, у якому будуть розташовуватись стовпці наслідуваних таблиць.

Таким чином, користуючись даними із таблиці pg_inherits, можемо отримати граф, на якому зображено схему наслідування у базі. Цей граф зображений на рисунку 4.

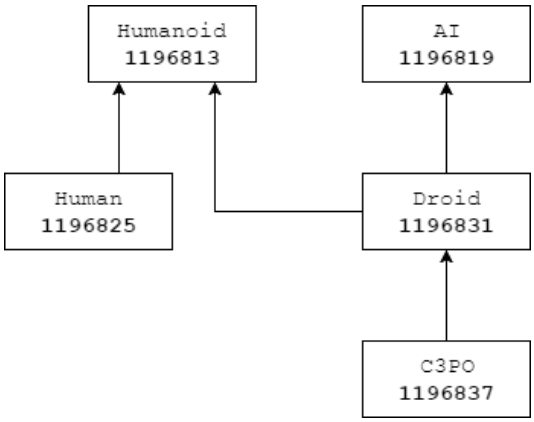

Рис. 4. Граф наслідування

Отже, можна зробити висновок, що для забезпечення базі даних можливістю наслідування потрібно в тому чи іншому вигляді зберігати граф

наслідування. У СУБД PostgreSQL дані про наслідування зберігаються у окремій службовій таблиці pg_inherits.

Слід звернути увагу, що не всі команди SQL можуть працювати з ієрархією наслідування. Команди, які використовуються для запитів отримання даних, зміни даних або зміни схем (наприклад, SELECT, UPDATE, DELETE, більшість варіантів ALTER TABLE, але не INSERT і ALTER TABLE ... RENAME) зазвичай за умовчанням включають таблиці-нащадки і підтримують нотацію ONLY для виключення цих таблиць-нащадків. Команди, які обслуговують базу даних і виконують тонкі настройки (наприклад, REINDEX, VACUUM) зазвичай працюють тільки з окремими, фізичними таблицями і не підтримують рекурсивную обробку ієрархії наслідування [2].

Серйозне обмеження можливості наслідування полягає в тому, що індекси (включаючи обмеження унікальності) і зовнішні ключі застосовуються тільки до одиноких таблиць, а не до нащадків. Це обмеження справедливо як для посилаються так і для посилальних сторін зовнішнього ключа.

Література

1. S. K. Singh. Database Systems: Concepts, Design and Applications / S. K. Singh – London : Dorling Kinderslay. – 2011.
2. PostgreSQL. Tutorial inheritance. – Режим доступа: https://www.postgresql.org/docs/8.4/static/tutorial-inheritance.html/. – Дата доступа: 27.07.2017.