# Тема №11 - Подключение базы данных POSTGRESQL к проекту VISUAL STUDIO.

**Наследование**

PostgreSQL реализует наследование таблиц, что может быть полезно для проектировщиков баз данных.

Стоит начать со следующего примера: предположим, что мы создаём модель данных для городов. В каждом штате есть множество городов, но лишь одна столица. Мы хотим иметь возможность быстро получать город-столицу для любого штата. Это можно сделать, создав две таблицы: одну для столиц штатов, а другую для городов, не являющихся столицами. Однако, что делать, если нам нужно получить информацию о любом городе, будь то столица штата или нет? В решении этой проблемы может помочь наследование. Мы определим таблицу capitals как наследника cities:

CREATE TABLE cities (

name text,

population float,

altitude int -- в футах

);

CREATE TABLE capitals (

state char(2)

) INHERITS (cities);

В этом случае таблица capitals наследует все столбцы своей родительской таблицы, cities. Столицы штатов также имеют дополнительный столбец state, в котором будет указан штат.

В PostgreSQL таблица может наследоваться от нуля или нескольких других таблиц, а запросы могут выбирать все строки родительской таблицы или все строки родительской и всех дочерних таблиц. По умолчанию принят последний вариант. Например, следующий запрос найдёт названия всех городов, включая столицы штатов, расположенных выше 500 футов:

SELECT name, altitude

FROM cities

WHERE altitude > 500;

А следующий запрос находит все города, которые не являются столицами штатов, но также находятся на высоте выше 500 футов:

SELECT name, altitude

FROM ONLY cities

WHERE altitude > 500;

Здесь ключевое слово ONLY указывает, что запрос должен применяться только к таблице cities, но не к таблицам, расположенным ниже cities в иерархии наследования. Многие операторы, которые мы уже обсудили, — SELECT, UPDATE и DELETE — поддерживают ключевое слово ONLY.

Вы также можете добавить после имени таблицы \*, чтобы явно указать, что должны включаться и дочерние таблицы:

SELECT name, altitude

FROM cities\*

WHERE altitude > 500;

Указывать \* не обязательно, так как теперь это поведение всегда подразумевается по умолчанию. Однако такая запись всё ещё поддерживается для совместимости со старыми версиями, где поведение по умолчанию могло быть изменено.

SELECT c.tableoid, c.name, c.altitude

FROM cities c

WHERE c.altitude > 500;

Механизм наследования не способен автоматически распределять данные команд INSERT или COPY по таблицам в иерархии наследования. Поэтому в нашем примере этот оператор INSERT не выполнится:

INSERT INTO cities (name, population, altitude, state)

VALUES ('Albany', NULL, NULL, 'NY');

INSERT всегда вставляет данные непосредственно в указанную таблицу. В некоторых случаях добавляемые данные можно перенаправлять, используя правила. Дочерние таблицы автоматически наследуют от родительской таблицы ограничения-проверки и ограничения NOT NULL. Все остальные ограничения (уникальности, первичный ключ и внешние ключи) не наследуются.

Таблица может наследоваться от нескольких родительских таблиц, в этом случае она будет объединять в себе все столбцы этих таблиц, а также столбцы, описанные непосредственно в её определении. Если в определениях родительских и дочерней таблиц встретятся столбцы с одним именем, эти столбцы будут «объединены», так что в дочерней таблице окажется только один столбец. Чтобы такое объединение было возможно, столбцы должны иметь одинаковый тип данных, в противном случае произойдёт ошибка. Наследуемые ограничения-проверки и ограничения NOT NULL объединяются подобным образом. Так, например, объединяемый столбец получит свойство NOT NULL, если какое-либо из порождающих его определений имеет свойство NOT NULL. Ограничения-проверки объединяются, если они имеют одинаковые имена; но, если их условия различаются, происходит ошибка.

Отношение наследования между таблицами обычно устанавливается при создании дочерней таблицы с использованием предложения INHERITS оператора CREATE TABLE. Другой способ добавить такое отношение для таблицы, определённой подходящим образом — использовать INHERIT с оператором ALTER TABLE. Для этого будущая дочерняя таблица должна уже включать те же столбцы (с совпадающими именами и типами), что и родительская таблица. Также она должна включать аналогичные ограничения-проверки (с теми же именами и выражениями). Удалить отношение наследования можно с помощью указания NO INHERIT оператора ALTER TABLE. Динамическое добавление и удаление отношений наследования может быть полезно при реализации секционирования таблиц.

Для создания таблицы, которая затем может стать наследником другой, удобно воспользоваться предложением LIKE оператора CREATE TABLE. Такая команда создаст новую таблицу с теми же столбцами, что имеются в исходной. Если в исходной таблице определены ограничения CHECK, для создания полностью совместимой таблицы их тоже нужно скопировать, и это можно сделать, добавив к предложению LIKE параметр INCLUDING CONSTRAINTS.

Родительскую таблицу нельзя удалить, пока существуют унаследованные от неё. При этом в дочерних таблицах нельзя удалять или модифицировать столбцы, или ограничения-проверки, унаследованные от родительских таблиц. Если вы хотите удалить таблицу вместе со всеми её потомками, это легко сделать, добавив в команду удаления родительской таблицы параметр CASCADE.

При изменениях определений и ограничений столбцов команда ALTER TABLE распространяет эти изменения вниз в иерархии наследования. Однако удалить столбцы, унаследованные дочерними таблицами, можно только с помощью параметра CASCADE. При создании отношений наследования команда ALTER TABLE следует тем же правилам объединения дублирующихся столбцов, что и CREATE TABLE.

В запросах с наследуемыми таблицами проверка прав доступа выполняется только в родительской таблице. Так, например, наличие разрешения UPDATE для таблицы cities подразумевает право на изменение строк и в таблице capitals, когда это изменение осуществляется через cities. Тем самым поддерживается видимость того, что данные находятся (также) в родительской таблице. Но изменить данные непосредственно в таблице capitals нельзя без дополнительного разрешения. Это правило имеет два исключения — команды TRUNCATE и LOCK TABLE, при выполнении которых всегда проверяются разрешения и для дочерних таблиц (то есть прямое обращение к таблицам и косвенное, через родительскую, обрабатываются одинаково).

Подобным образом, политики безопасности на уровне строк для родительской таблицы применяются к строкам, получаемым из дочерних таблиц при выполнении запроса с наследованием. Политики же дочерних таблиц, если они определены, действуют только когда такие таблицы явно задействуются в запросе; в этом случае все политики, связанные с родительскими таблицами, игнорируются.

Не все SQL-команды могут работать с иерархиями наследования. Команды, выполняющие выборку данных, изменение данных или модификацию схемы (например, SELECT, UPDATE, DELETE, большинство вариантов ALTER TABLE, но не INSERT и ALTER TABLE ... RENAME), обычно по умолчанию обрабатывают данные дочерних таблиц и могут исключать их, если поддерживают указание ONLY. Команды для обслуживания и настройки базы данных (например, REINDEX и VACUUM) обычно работают только с отдельными физическими таблицами и не поддерживают рекурсивную обработку отношений наследования.

Возможности наследования серьёзно ограничены тем, что индексы (включая ограничения уникальности) и ограничения внешних ключей относятся только к отдельным таблицам, но не к их потомкам. Некоторая функциональность, не реализованная для иерархий наследования, реализована для декларативного секционирования. Поэтому обязательно взвесьте все за и против, прежде чем применять в своих приложениях секционирование с использованием устаревшего наследования.