Организация данных

Основные объекты БД

Темы

Таблицы, типы данных и ограничения целостности

Последовательности

Индексы

Функции пользователя

Триггеры

Представления и материализованные представления

Таблицы

Обычные

Нежурналируемые

- не происходит запись в журнал
- восстановление невозможно; при сбое очищаются (слой init)

Временные

- хранят данные до конца транзакции (ON COMMIT DELETE ROWS) или сеанса (ON COMMIT PRESERVE ROWS)
- существуют на время сеанса или транзакции (ON COMMIT DROP)
- не журналируются
- не попадают в общий буферный кэш
- активное использование вредит системному каталогу

Типы данных

Postgres PROFESSIONAL

Числовые

целые, вещественные с плавающей и фиксированной запятой

Символьные

с фиксированной и переменной длиной

Даты

даты с временной зоной и без, время, интервалы

Логический

трехзначная логика (true, false, null)

Составные

записи, аналогичные строкам таблиц

Массивы

в том числе многомерные

Типы данных

Двоичные

байтовые и битовые массивы

Перечислимые

заданный набор значений

Диапазоны

пара значений «от … до …»

Геометрические

точка, прямая, отрезок, прямоугольник, ломаная, многоугольник, круг

Сложносоставные

XML, JSON, JSONB

…

Последовательности

Генерация уникальных номеров

Нетранзакционное изменение

- не гарантируется отсутствие пропусков

Возможность кэширования на уровне сеансов

- не гарантируется строгая последовательность

Псевдотип serial

- для полей с автоувеличением

Ограничения целостности

Первичный ключ (primary key) и уникальность (unique)

- поддерживаются индексом

Обязательность (not null)

Внешний ключ (foreign key)

Проверка (check)

- на уровне одной строки

Исключение (exclude)

- на уровне всех строк таблицы
- поддерживается специальным индексом

Индексы-B-деревья

Задачи

- ускорение доступа к небольшой части данных
- выдача результата в порядке сортировки
- поддержка ограничений целостности (уникальность)

Побочные эффекты

- накладные расходы на поддержание
- занимаемое место

Вариации

- индексы по нескольким столбцам
- индексы по выражениям
- частичные индексы

Другие индексы

GiST — обобщенное сбалансированное дерево

R-деревья; поиск ближайших соседей...

SP-GiST — обобщенное несбалансированное дерево

деревья квадрантов; префиксные деревья...

GIN — инвертированный список

полнотекстовый поиск...

BRIN (9.5) — диапазоны блоков

для больших физически упорядоченных таблиц

Bloom (9.6) — фильтр Блума

быстрое исключение строк с ложными позитивными срабатываниями

Функции пользователя

Хранятся и выполняются на сервере, рядом с данными

Различные языки программирования

- чистый SQL
- процедурные языки: PL/pgSQL, а также PL/Python, PL/Perl, PL/Tcl и др.
- Си (статически или динамически подключаемые)

Богатые возможности

- перегруженные функции
- полиморфные типы
- переменное число параметров
- функции, возвращающие наборы строк
- пользовательские агрегатные и оконные функции

Триггеры

Postgres PROFESSIONAL

Функции, срабатывающие при определенных событиях

- можно писать на любом языке, кроме SQL
- дополнительные ограничения для гарантии согласованности данных; логика изменения данных (с осторожностью)

Табличный триггер

- вставка, изменение, удаление или очистка строк таблиц и представлений
- срабатывает до, после или вместо операции
- срабатывает для всей операции или для каждой строки

Событийный триггер

- создание, изменение или удаление объекта, выдача или отзыв привилегий и др.
- срабатывает до или после команды

Представления

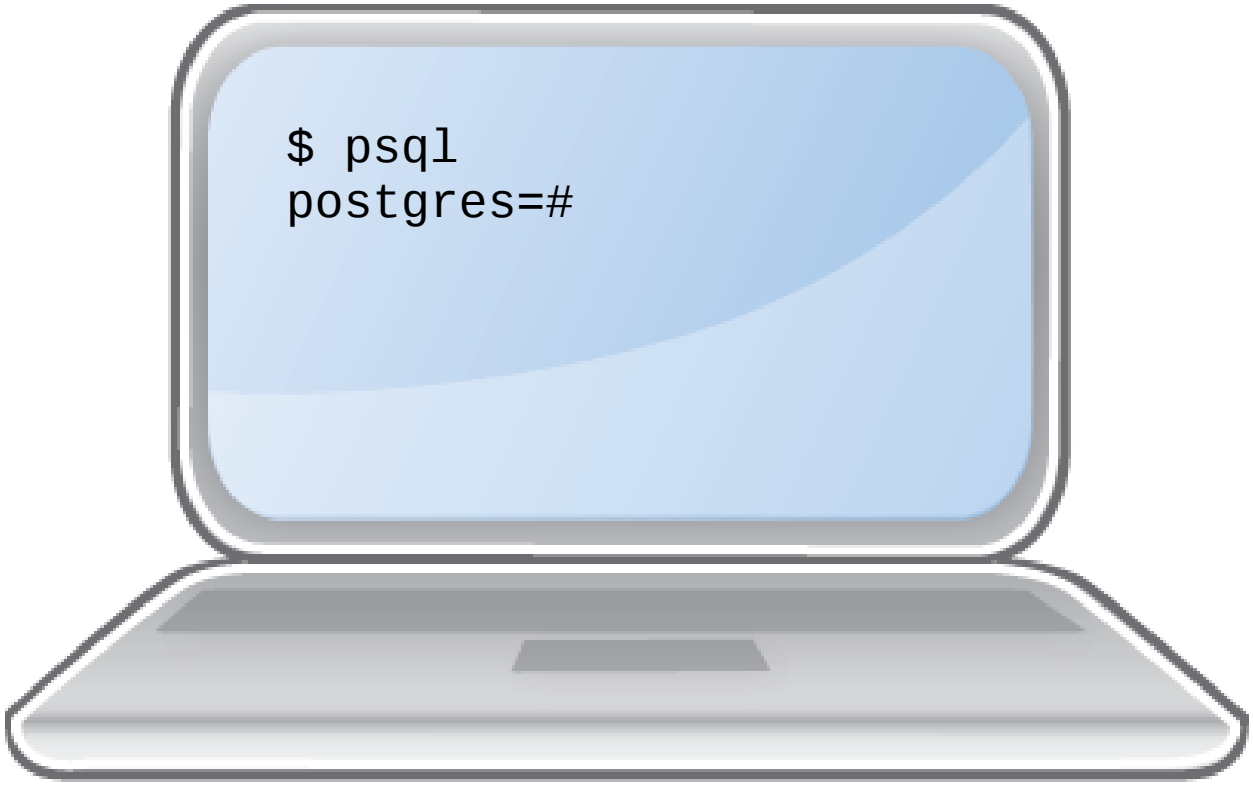
Представления

- хранимый текст запроса, подставляется с помощью правил
- используются (почти) так же, как обычные таблицы,
  но данные не дублируются
- могут использоваться для разграничения доступа

Материализованные представления

- результат запроса предварительно вычисляется и сохраняется в таблице
- обновление таблицы происходит по требованию
- используются для оптимизации,
  но требуют синхронизации с базовыми таблицами

Демонстрация



Итоги

PostgreSQL предоставляет как стандартный набор объектов, так и некоторые особые возможности

Расширяемость системы позволяет увеличивать функционал

Практика

1. В новой базе данных создайте таблицу observations для ведения наблюдений за температурой. Таблица должна содержать поля:
   - date_taken — дата измерений, не пустое
   - temperature — температура, от −60°C до +60°C
2. Добавьте в таблицу несколько произвольных записей; убедитесь, что ограничения действуют.
3. Создайте представление observ_fahrenheit, показывающее температуру в градусах Фаренгейта (°F = °C·9/5 + 32).
4. Проиндексируйте таблицу по дате измерений.

Авторские права
© Postgres Professional, 2017 год.
Авторы: Егор Рогов, Павел Лузанов

Использование материалов курса
Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Отказ от ответственности
Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или непрямым, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Темы

Таблицы, типы данных и ограничения целостности

Последовательности

Индексы

Функции пользователя

Триггеры

Представления и материализованные представления

В презентации содержатся основные теоретические сведения. Практическое применение рассматривается в демонстрации.

Таблицы

Postgres PROFESSIONAL

Обычные

Нежурналируемые

- не происходит запись в журнал
- восстановление невозможно; при сбое очищаются (слой init)

Временные

- хранят данные до конца транзакции (ON COMMIT DELETE ROWS) или сеанса (ON COMMIT PRESERVE ROWS)
- существуют на время сеанса или транзакции (ON COMMIT DROP)
- не журналируются
- не попадают в общий буферный кэш
- активное использование вредит системному каталогу

Таблицы — основной объект базы данных. Кроме обычных таблиц (с которыми вы должны быть знакомы из любого курса SQL), поддерживаются две их разновидности.

Таблица может быть нежурналируемой. Изменения, связанные с такой таблицей, не записываются в журнал упреждающей записи. Это позволяет обходиться меньшими накладными расходами, но в случае сбоя восстановить содержимое такой таблицы невозможно. Поэтому для нежурналируемых таблиц хранится дополнительный init-слой, который при сбое копируется поверх основного, стирая его содержимое — таблица становится пустой. Аналогично обстоит дело и с индексами, созданными для нежурналируемой таблицы.

Временные таблицы предназначены для хранения данных, которые должны быть доступны только текущему сеансу (и только на время его жизни, или даже на время текущей транзакции).

Временные таблицы также являются нежурналируемыми. Кроме того, страницы таких таблиц не попадают в общий буферный кэш. Все это позволяет экономить вычислительные ресурсы, но постоянное создание и удаление временных таблиц приводит к изменению таблиц системного каталога, что требует частой обработки процессом очистки.

https://postgrespro.ru/docs/postgresql/9.6/ddl-basics.html

Типы данных

Числовые

целые, вещественные с плавающей и фиксированной запятой

Символьные

с фиксированной и переменной длиной

Даты

даты с временной зоной и без, время, интервалы

Логический

трехзначная логика (true, false, null)

Составные

записи, аналогичные строкам таблиц

Массивы

в том числе многомерные

Столбцы таблиц должны иметь определенный тип данных. Тип определяет, какие значения могут храниться в этом столбце и какие операции определены над этими значениями.

Поддерживается множество различных типов данных, основные из которых приведены на этом слайде, а некоторые дополнительные — на следующем.

https://postgrespro.ru/docs/postgresql/9.6/datatype.html

https://postgrespro.ru/docs/postgresql/9.6/functions.html

Типы данных

Двоичные

байтовые и битовые массивы

Перечислимые

заданный набор значений

Диапазоны

пара значений «от … до …»

Геометрические

точка, прямая, отрезок, прямоугольник, ломаная, многоугольник, круг

Сложносоставные

XML, JSON, JSONB

…

Геометрические типы могут, например, используются в гео-информационных системах, а полноценная поддержка XML и JSON позволяет использовать PostgreSQL в области NoSQL, не жертвуя при этом ни эффективностью, ни транзакционностью.

В целом, благодаря расширяемой архитектуре PostgreSQL, можно реализовать и подключить любые типы данных; многие можно найти в сторонних расширениях.

Последовательности

Генерация уникальных номеров

Нетранзакционное изменение

- не гарантируется отсутствие пропусков

Возможность кэширования на уровне сеансов

- не гарантируется строгая последовательность

Псевдотип serial

- для полей с автоувеличением

Последовательности удобно использовать для генерации уникальных номеров, в частности, для первичных ключей таблиц (для этого случая удобно воспользоваться псевдотипом serial или bigserial — при этом создается последовательность и поле будет автоматически получать из нее значения).

Внутренняя реализация последовательностей использует специальную однострочную таблицу, которая хранит текущий номер.

В общем случае последовательность гарантирует только уникальность номеров. Если разрешить кэширование, то каждый сеанс будет получать сразу диапазон значений и выдавать их по одному; при этом параллельные сеансы будут выдавать номера не в общем порядке. Таким образом повышается эффективность, так как обращение к таблице происходит не каждый раз.

Также не гарантируется отсутствие пропусков в номерах — если взять из последовательности номер и не использовать его (или, например, если транзакция будет отменена), уже выданный номер не будет выдан повторно.

https://postgrespro.ru/docs/postgresql/9.6/functions-sequence.html

Ограничения целостности

Первичный ключ (primary key) и уникальность (unique)

- поддерживаются индексом

Обязательность (not null)

Внешний ключ (foreign key)

Проверка (check)

- на уровне одной строки

Исключение (exclude)

- на уровне всех строк таблицы
- поддерживается специальным индексом

Для таблиц могут быть заданы ограничения целостности. PostgreSQL поддерживает весь стандартный набор ограничений, включая ссылочную целостность.

Специфичным является «исключение», которое позволяет задать ограничение не на уровне отдельной строки, а для всей таблицы сразу.

Ограничения могут проверяться постоянно, либо откладываться до конца транзакции. В любом случае PostgreSQL гарантирует свойство согласованности: транзакция должна переводить базу данных из одного целостного состояния в другое.

Дополнительные ограничения согласованности данных, не представимые в виде стандартных ограничений целостности, можно реализовать с помощью триггеров.

https://postgrespro.ru/docs/postgresql/9.6/ddl-constraints.html

Индексы-B-деревья

Задачи

- ускорение доступа к небольшой части данных
- выдача результата в порядке сортировки
- поддержка ограничений целостности (уникальность)

Побочные эффекты

- накладные расходы на поддержание
- занимаемое место

Вариации

- индексы по нескольким столбцам
- индексы по выражениям
- частичные индексы

Индексы используются для получения быстрого доступа к (обычно) небольшой части имеющихся данных.

Стандартный, наиболее часто используемый вид индекса — B-дерево. Этот индекс упорядочивает данные, что позволяет не только быстро находить нужные значения, но и выдавать результаты сразу в отсортированном виде.

Этот тип индекса также поддерживает ограничение уникальности; соответствующий индекс создается автоматически при объявлении первичного или уникального ключа.

Индексы могут строиться как по одному, так и нескольким столбцам. Кроме того, можно индексировать не столбец, а выражение. Частичные индексы позволяют индексировать не все строки, уменьшая таким образом размер и увеличивая эффективность.

https://postgrespro.ru/docs/postgresql/9.6/indexes.html

Другие индексы

GiST — обобщенное сбалансированное дерево

R-деревья; поиск ближайших соседей...

SP-GiST — обобщенное несбалансированное дерево

деревья квадрантов; префиксные деревья...

GIN — инвертированный список

полнотекстовый поиск...

BRIN (9.5) — диапазоны блоков

для больших физически упорядоченных таблиц

Bloom (9.6) — фильтр Блума

быстрое исключение строк с ложными позитивными срабатываниями

Кроме B-деревьев, PostgreSQL предлагает множество других специализированных индексов.

Они могут понадобиться, если требуется индексировать значения, для которых не определен порядок (например, прямоугольники на плоскости); находить ближайших соседей к заданной точке; пользоваться полнотекстовым поиском и т. д.

Подробно эти типы индексов здесь не рассматриваются.

Функции пользователя

Хранятся и выполняются на сервере, рядом с данными

Различные языки программирования

- чистый SQL
- процедурные языки: PL/pgSQL, а также PL/Python, PL/Perl, PL/Tcl и др.
- Си (статически или динамически подключаемые)

Богатые возможности

- перегруженные функции
- полиморфные типы
- переменное число параметров
- функции, возвращающие наборы строк
- пользовательские агрегатные и оконные функции

Хранимые функции позволяют пользователям расширить набор имеющихся встроенных функций и расположить логику обработки данных на сервере, рядом с самими данными.

Функции могут быть написаны как на SQL, так и на ряде процедурных языков, из которых особо важен PL/pgSQL. Там, где необходима крайняя эффективность, можно написать функцию и на языке Си.

Поддерживается перегрузка функций, то есть допускаются функции с одинаковыми именами, отличающиеся типом и числом параметров. Также поддерживаются функции, работающие с полиморфными типами данных.

Можно создавать функции с переменным числом параметров.

Функции могут возвращать не только отдельные значения, но и наборы строк, что позволяет использовать их в запросах наравне с таблицами.

Можно создавать и собственные агрегатные и оконные функции для использования в запросах с GROUP BY и PARTITION BY.

https://postgrespro.ru/docs/postgresql/9.6/xfunc.html

Триггеры

Postgres PROFESSIONAL

Функции, срабатывающие при определенных событиях

- можно писать на любом языке, кроме SQL
- дополнительные ограничения для гарантии согласованности данных; логика изменения данных (с осторожностью)

Табличный триггер

- вставка, изменение, удаление или очистка строк таблиц и представлений
- срабатывает до, после или вместо операции
- срабатывает для всей операции или для каждой строки

Событийный триггер

- создание, изменение или удаление объекта, выдача или отзыв привилегий и др.
- срабатывает до или после команды

Триггеры — это функции, которые срабатывают при определенных событиях.

Триггер может быть задан для таблицы (в том числе внешней; эта тема рассматривается в курсе DBA2) или представления. Он срабатывает при вставке, изменении, удалении или очистке.

Триггер может срабатывать как перед действием, так и после или даже вместо него; а также для каждой строки или только один раз для всей операции.

Триггеры могут (с известной осторожностью и аккуратностью) использоваться для обеспечения таких ограничений целостности, которые невозможно реализовать декларативно, а также (с еще большей аккуратностью) для реализации определенной логики изменения данных.

https://postgrespro.ru/docs/postgresql/9.6/triggers.html

Событийные триггеры аналогичны обычным триггерам, но не связаны с какой-либо таблицей или представлением. Они срабатывают при событиях DDL (изменяющих определение данных), например, при выполнении команд create index, alter table, drop type и т. п.

https://postgrespro.ru/docs/postgresql/9.6/event-triggers.html

Представления

Postgres PROFESSIONAL

Представления

- хранимый текст запроса, подставляется с помощью правил
- используются (почти) так же, как обычные таблицы,
  но данные не дублируются
- могут использоваться для разграничения доступа

Материализованные представления

- результат запроса предварительно вычисляется и сохраняется в таблице
- обновление таблицы происходит по требованию
- используются для оптимизации,
  но требуют синхронизации с базовыми таблицами

Представления — это хранимый текст запроса. Когда в запросе происходит обращение к представлению, его имя заменяется на запомненный текст.

Представления могут использоваться, чтобы предоставить удобный доступ к «нижележащим» таблицам или ввести ограничение доступа. По большому счету, представление и не отличимо от таблицы с точки зрения SQL, пока дело не касается изменения данных (но и в этом случае представление можно «замаскировать» с помощью instead-of-триггеров).

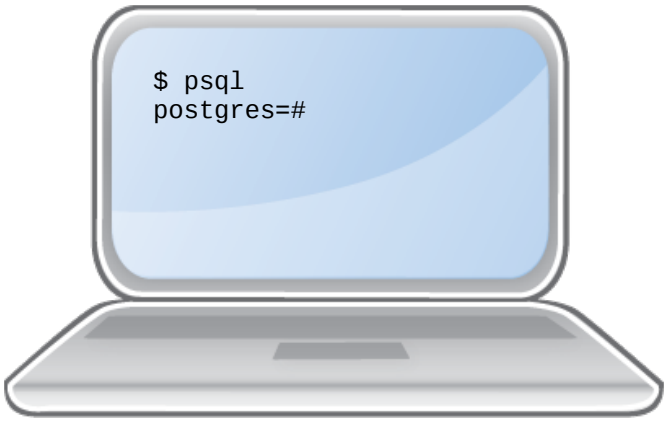
Материализованное представление отличаются от обычного тем, что при его создании запрос выполняется и его результат сохраняется в специальную таблицу. При обращении к материализованному представлению обращение фактически происходит к заранее подготовленной таблице. Обновление материализованного представления происходит по требованию.

Использование материализованных представлений может существенно улучшить производительность, но следует иметь в виду устаревание собранных данных, накладные расходы на их пересчет и необходимость выполнять такой пересчет на периодической основе.

http://www.postgresql.org/docs/current/static/rules-views.html

http://www.postgresql.org/docs/current/static/rules-materializedviews.html

Демонстрация



Итоги

PostgreSQL предоставляет как стандартный набор объектов, так и некоторые особые возможности

Расширяемость системы позволяет увеличивать функционал

Практика

1. В новой базе данных создайте таблицу observations для ведения наблюдений за температурой. Таблица должна содержать поля:
   - date_taken — дата измерений, не пустое
   - temperature — температура, от −60°C до +60°C
2. Добавьте в таблицу несколько произвольных записей; убедитесь, что ограничения действуют.
3. Создайте представление observ_fahrenheit, показывающее температуру в градусах Фаренгейта (°F = °C·9/5 + 32).
4. Проиндексируйте таблицу по дате измерений.