//слайд 1

**Определение.** //слайд 2

PostgreSQL — это платформа с открытым исходным кодом, объектно-ориентированная система баз данных. Это одна из самых популярных баз данных, используемых сегодня, с готовыми функциями и совместимостью с рядом языков программирования. Имеет открытый исходный код и базируется на языке SQL.

PostgreSQL является объектно-реляционной СУБД, что означает, что она поддерживает и объектный, и реляционный подход.

Реляционная СУБД -- данные, которые в них хранятся, представляются в виде записей, связанных друг с другом отношениями.

Объектная СУБД -- данные представляются в виде объектов, их атрибутов, методов и классов. Объектная модель поддерживает принципы ООП (инкапсуляция, наследование, полиморфизм, абстракция) Также такие СУБД называют сокращенно ООСУБД

**Сравнение синтаксиса MS SQL Server и PostgreSQL** //слайд 3

**Сопоставление типов** //слайд 4

|  |  |  |
| --- | --- | --- |
| Тип | PostgreSQL | MsSQL |
| целочисленный тип данных | BIGINT | BIGINT, INT8 |
| хранение фиксированного количества байтов | BINARY(n) | BYTEA |
| используется для хранения двоичных данных | VARBINARY(n) | BYTEA //слайд 5 |
| используется для хранения переменного количества двоичных данных | VARBINARY(max) | BYTEA |
| используется для контроля версий и оптимистической блокировки. | ROWVERSION | BYTEA |
| используется для хранения двоичных данных, | IMAGE | BYTEA |
| используется для хранения и манипулирования иерархическими путями данных | FIELDHIERARCHYID | BYTEA, LTREE(расширение) |
| может принимать значения TRUE или FALSE, представленные соответственно 1 и 0 | BIT | BOOLEAN, BOOL |
| хранит строку фиксированного размера, где "n" указывает на количество символов. | CHAR(n), n<=8000 | TEXT //слайд 6 |
| аналогично CHAR(n), но для хранения символов Юникода | NCHAR(n), n<=4000 | TEXT |
| для хранения переменной длины символо | VARCHAR(n), n<=8000 | TEXT |
| аналогично VARCHAR(n), но для хранения символов Юникода (Unicode characters). | NVARCHAR(n), n<=4000 | TEXT |
| тип данных для хранения переменной длины символов с максимальным размером до 2^31-1 символов. | VARCHAR(max) | TEXT |
| аналогично VARCHAR(max), но для хранения символов Юникода | NVARCHAR(max) | TEXT |
| устаревший тип данных для хранения переменной длины текстовых данных, с максимальным размером до 2^31-1 символов. | TEXT | TEXT //слайд 7 |
| устаревший тип данных для хранения символов Юникода переменной длины, с максимальным размером до 2^30-1 символов. | NTEXT | TEXT |
| тип данных для хранения чисел с плавающей точкой, где "n" указывает на количество битов, используемых для хранения числа. | FLOAT(n) | DOUBLE PRECISION, FLOAT8 |
| это тип данных для хранения денежных значений с малой точностью. Хранит денежные значения от -214,748.3648 до 214,748.3647. | SMALLMONEY | MONEY |
| это тип данных для хранения денежных значений с большей точностью. Хранит денежные значения от -922,337,203,685,477.5808 до 922,337,203,685,477.5807. | MONEY | MONEY //слайд 8 |
| это целочисленный тип данных, предназначенный для хранения целых чисел. INT имеет размер 4 байта и может хранить числа от -2,147,483,648 до 2,147,483,647. | INT, INTEGER | INT, INTEGER, INT4 |
| это целочисленный тип данных меньшего размера, чем INT. SMALLINT имеет размер 2 байта и может хранить числа от -32,768 до 32,767. | SMALLINT | SMALLINT, INT2 |
| это тип данных для хранения чисел с фиксированной точностью и масштабом. "n" указывает на общую разрядность числа, а "m" указывает на количество разрядов после запятой. | NUMERIC(n,m) | NUMERIC(n,m) |
| это тип данных для хранения чисел с фиксированной точностью и масштабом. "n" указывает на общую разрядность числа, а "m" указывает на количество разрядов после запятой. | DEC(n,m), DECIMAL(n,m) | DEC(n,m), DECIMAL(n,m)  //слайд 9 |
| это целочисленный тип данных малого размера. TINYINT имеет размер 1 байт и может хранить числа от 0 до 255. | TINYINT | SMALLINT, INT2 |
| это тип данных для хранения чисел с плавающей точкой одинарной точности. REAL имеет размер 4 байта. | REAL | REAL, FLOAT4 |
| это тип данных для хранения уникального идентификатора (GUID). Используется для создания глобально уникальных идентификаторов. | UNIQUEIDENTIFIER | CHAR(16), UUID  //слайд 10 |
| это тип данных для хранения только даты без времени. | DATE | DATE |
| это тип данных для хранения только времени. "n" указывает на количество разрядов после запятой. | TIME(n) | TIME(n) |
| это тип данных для хранения даты и времени. Он представляет собой комбинацию даты и времени в формате гггг-мм-дд чч:мм:сс. | DATETIME | TIMESTAMP(3) |
| это тип данных для хранения даты и времени с более высокой точностью. "n" указывает на количество разрядов после запятой. | DATETIME2(n) | TIMESTAMP(m) |
| это тип данных для хранения даты, времени и смещения временной зоны. "n" указывает на количество разрядов после запятой. | DATETIMEOFFSET(n) | TIMESTAMP(n) WITH TIME  ZONE, TIMESTAMPTZ  //слайд 11 |
| это тип данных для хранения даты и времени с меньшей точностью, чем DATETIME. SMALLDATETIME используется для хранения даты и времени с точностью до минуты. | SMALLDATETIME | TIMESTAMP(0) |
| это тип данных для хранения XML-документов. XML может содержать структурированные данные в формате XML. | XML | XML |

**Сопоставление синтаксиса MS SQL Server и PostgreSQL**

**I. Регистрозависимое обращение к схемам, таблицам (представлениям) и их полям и другим объектам базы данных** //слайд 12

|  |  |  |
| --- | --- | --- |
|  | PostgreSQL | MsSQL |
| Код |  |  |
| Объяснение отличий в синтаксисе | В PostgreSQL при обращениях к объектам используются двойные кавычки (они обязательны, только если в названии объекта присутствуют заглавные буквы или есть недопустимые символы в названии объекта или его поля): | В MS SQL Server при обращениях к объектам можно использовать квадратные скобки (они обязательны, только если в названии объекта или его поля присутствуют недопустимые символы) |

**II. Выборка заданных N данных** //слайд 13

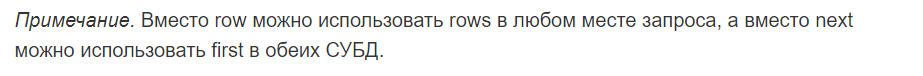
|  |  |  |
| --- | --- | --- |
|  | PostgreSQL | MsSQL |
| Код |  |  |
| Объяснение отличий в синтаксисе | В PostgreSQL используется LIMIT | В MS SQL Server используется TOP |

**III. Постраничная загрузка данных (скользящее окно)** //слайд 14

Предположим, у нас есть задача: извлечь 100 строк начиная с 202-й строки включительно по возрастанию даты рождения. Тогда:

|  |  |  |
| --- | --- | --- |
|  | PostgreSQL | MsSQL |
| Код |  |  |

|  |  |
| --- | --- |
| Объяснение кода | С помощью **select \* from tbl** выбираем все столбцы из таблицы **tbl**. Т.к. используем order by, результаты выборки будут отсортированы по полю **BirthDate**. Т.к. используем **asc**, сортировка будет по возрастанию. Используем **OFFSET 201** **ROW**– пропускаем 201 строку перед началом выборки. **FETCH NEXT 100 ROWS** – выбираем следующие 100 строк. **ONLY** –мы выбираем только 100 строк из результирующего набора. |

****

**IV. Тернарный оператор IIF** //слайд 15

|  |  |  |
| --- | --- | --- |
|  | PostgreSQL | MsSQL |
| Код | case when <условие> then <выражение\_если\_условие\_истинно> else <выражение\_если\_условие\_ложно> end |  |

**V. Выражения CASE** //слайд 16

|  |  |  |
| --- | --- | --- |
|  | PostgreSQL | MsSQL |
| Код |  |  |

**VI.Работа с переменными** //слайд 17

|  |  |  |
| --- | --- | --- |
|  | PostgreSQL | MsSQL |
| Объявление переменной |  |  |
| Присвоение переменной значения |  |  |
| Вывод значения на консоль |  |  |

**VII.Цикл while** //слайд 18

|  |  |  |
| --- | --- | --- |
|  | PostgreSQL | MsSQL |
| Код |  |  |

**VIII.Логическое ветвление** //слайд 19

|  |  |  |
| --- | --- | --- |
|  | PostgreSQL | MsSQL |
| Код |  |  |

**Возможности и производительность** /слайд 20

1. ***Высокая производительность.*** PostgreSQL обеспечивает эффективную обработку, индексирование и оптимизацию запросов с поддержкой параллельного выполнения запросов. //слайд 21
2. ***Совместимость.*** PostgreSQL легко интегрируется с различными системами и может быть легко интегрирован в существующие или новые системы. //слайд 22
3. ***Расширяемость.*** PostgreSQL допускает модификацию кода, позволяя пользователям добавлять собственные функции без дополнительных лицензионных сборов. Активное сообщество разработчиков PostgreSQL предоставляет поддержку, ресурсы для устранения неполадок и постоянные улучшения для оптимизации производительности. //слайд 23
4. ***Масштабируемость.*** PostgreSQL предлагает горизонтальную масштабируемость с помощью таких методов, как секционирование таблиц, сегментирование и потоковая репликация, гарантируя эффективную обработку больших наборов данных. //слайд 24
5. ***Целостность данных и контроль параллелизма.*** PostgreSQL обеспечивает целостность данных и эффективно обрабатывает несколько одновременных транзакций. //слайд 25
6. ***Богатый набор функций.*** PostgreSQL предлагает обширные функции, включая поддержку JSON, полнотекстовый поиск, обработку пространственных данных и расширенные возможности манипулирования данными. //слайд 26

**Сравнение базового функционала MsSQL и PostgreSQL** //слайд 27

|  |  |  |
| --- | --- | --- |
|  | PostgreSQL | MsSQL |
| Временные таблицы  //слайд 28 | Временные таблицы делятся на локальные и глобальные и настраиваются с помощью гибких переменных. | Можно создавать локальные и глобальные временные таблицы, а также контролировать и создавать переменные. |
| Индексы  //слайд 29 | Поддерживает организацию таблиц на основе индексов, но ранние версии не включают автоматические обновления индексов. Решение также позволяет искать много индексов в одном поиске, а это значит, что вы можете найти много информации | Предлагает широкие автоматизированные функциональные возможности для управления индексами. Они могут объединяться в кластеры и поддерживать правильный порядок строк без ручного вмешательства. Решение также поддерживает поиск по нескольким индексам и частичные индексы. Наличие гибких настроек индекса позволяет быстрее искать информацию и организовывать несколько данных одновременно. |
| Таблицы, оптимизированные для памяти (Memory-optimized tables)  //слайд 30 | Не поддерживает создание базы данных в памяти | Использует оптимистичную стратегию для обработки оптимизированных для памяти таблиц, что означает, что они могут участвовать в транзакциях наряду с обычными таблицами. Транзакции на основе памяти быстрее, чем обычные, и это позволяет резко увеличить скорость приложения. |
| Поддержка JSON  //слайд 31 | поддерживает JSON-файлы, а также их индексацию и частичные обновления. Пользователи могут загружать определенные типы, геопространственные данные, создавать многомерные массивы. | Обеспечивает полную поддержку документов JSON, их обновлений, функциональности и обслуживания. Он имеет множество дополнительных функций для данных GPS, пользовательских типов, иерархической информации и т. д. |
| Разделение  //слайд 32 | позволяет создавать разделы LIST и RANGE, где индекс раздела создается вручную. Разработчикам необходимо определить дочерний и родительский столбцы, прежде чем назначать для них раздел. | предоставляет доступ к секционированию RANGE, где раздел назначается всем значениям, попадающим в определенный диапазон. Если данные находятся в пределах порогового значения, они будут перемещены в секцию. |
| Экосистема СУБД  //слайд 33 | Сообщество PostgreSQL предлагает множество инструментов для масштабирования и оптимизации программного обеспечения. Интеграция позволяет разработчикам выполнять кластеризацию, интеграцию ИИ, совместную работу, отслеживание проблем, улучшение отображения объектов и охватывать многие другие важные функции. | ориентирована на большие инфраструктуры. Это дороже, чем конкуренты с открытым исходным кодом, но в конце концов пользователи получают доступ к часто обновляемой официальной экосистеме и активной поддержке клиентов. |

**На что обратить внимание при выборе системы управления базами данных?** //слайд 34

1. Производительность
2. Масштабируемость \*1
3. Безопасность
4. Надежность и доступность
5. Целостность данных и контроль параллелизма \*2
6. Совместимость \*3
7. Простота в использовании
8. Гибкость
9. Поддержка и сообщество

10) Экономическая эффективность \*4

\*1 **Масштабируемость:** Это возможность системы или приложения управлять ростом нагрузки путем добавления ресурсов, таких как серверы, хранилища данных или сетевые узлы, с целью поддержания высокой производительности.

**\*2 Целостность данных и контроль параллелизма:** Это обеспечение корректности данных и избежание их повреждения или потери при одновременном доступе нескольких пользователей или процессов к базе данных.

**\*3 Совместимость:** Это способность системы или приложения работать вместе с другими системами, программным обеспечением или устройствами, обеспечивая совместимость форматов данных, протоколов и интерфейсов.

**\*4 Экономическая эффективность:** Это обеспечение оптимального соотношения между затратами на разработку, внедрение и поддержку системы или приложения и полученными выгодами или результатами, такими как повышение производительности, сокращение расходов или улучшение бизнес-показателей.

**Варианты использования СУБД** //слайд 35

**Варианты использования PostgreSQL:**

1. **Веб-приложения:** надежная, масштабируемая и многофункциональная база данных для веб-приложений.
2. **Геопространственные приложения:** расширенная поддержка геопространственных данных в ГИС и сервисах, основанных на местоположении.
3. **Хранилище данных и бизнес-аналитика: э**ффективно обрабатывать большие наборы данных для хранения данных и бизнес-аналитики.
4. **Системы управления контентом (CMS):** Надежное и стабильное хранилище данных для платформ CMS.

**Варианты использования SQL Server:**

1. **Корпоративные приложения:** комплексные функции и масштабируемость для приложений корпоративного уровня.
2. **Бизнес-приложения:** интеграция с технологиями Microsoft для финансов, управления персоналом, управления запасами и CRM.
3. **Анализ данных и отчетность:** интегрированные инструменты бизнес-аналитики для анализа данных и составления отчетов.
4. **Электронная коммерция и онлайн-торговля:** транзакционные возможности для платформ электронной коммерции и онлайн-торговли с высоким трафиком.

**Какая СУБД лучше?** //слайд 36

Какая база данных лучше из двух рассматриваемых сегодня в презентации в конечном итоге сводится к сравнению функциональности, вариантов использования и экосистемам. Компании, которые отдают приоритет гибкости, экономической эффективности и инновациям, обычно выбирают решения с открытым исходным кодом. Они могут быть интегрированы с несколькими бесплатными дополнениями, иметь активные сообщества пользователей и постоянно обновляться.

Для корпораций, которые предпочитают традиционные коммерческие решения, программное обеспечение, такое как MsSQL. Они имеют доступ к постоянной технической поддержке, персонализированной помощи и профессиональным инструментам управления.