Севастопольский государственный университет
Кафедра «Информационные системы»

Управление данными

курс лекций

лектор:
ст. преподаватель кафедры ИС Абрамович А.Ю.

Лекция 8

SQL — язык структурированных запросов

Формирование запросов к базе данных

ИСТОРИЯ РАЗВИТИЯ SQL

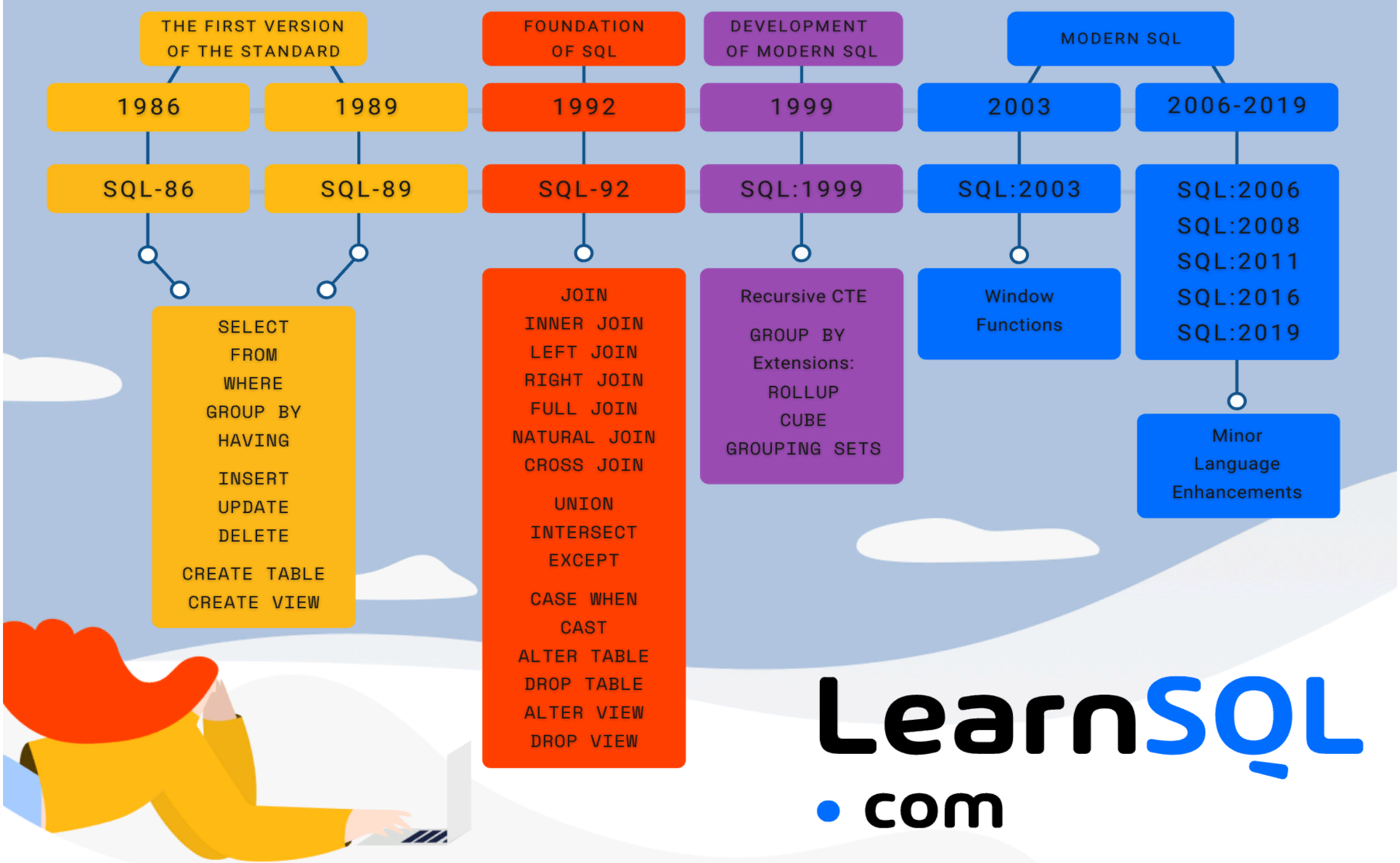
SQL (Structured Query Language) — **структурированный язык запросов** — стандартный язык запросов по работе с реляционными БД. Язык SQL **появился после реляционной алгебры**, и его **прототип был разработан в конце 70-х** годов в компании **IBM Research**. Он был реализован в первом прототипе реляционной СУБД фирмы IBM System R. В дальнейшем этот язык применялся во многих коммерческих СУБД и в силу своего широкого распространения постепенно стал стандартом «де-факто» для языков манипулирования данными в реляционных СУБД.

Первый международный стандарт языка SQL был принят в 1989 г. (SQL/89 или SQL1). Большинство доступных на рынке СУБД поддерживают этот стандарт полностью. Однако **развитие информационных технологий, связанных с базами данных, и необходимость реализации переносимых приложений потребовали в скором времени доработки и расширения первого стандарта SQL.** В **конце 1992 г. был принят новый международный стандарт языка SQL (SQL/92 или SQL2).** И он не лишен недостатков, но в то же время является существенно более точным и полным, чем SQL/89. В настоящий момент большинство производителей СУБД внесли изменения в свои продукты так, чтобы они в большей степени удовлетворяли стандарту SQL2.

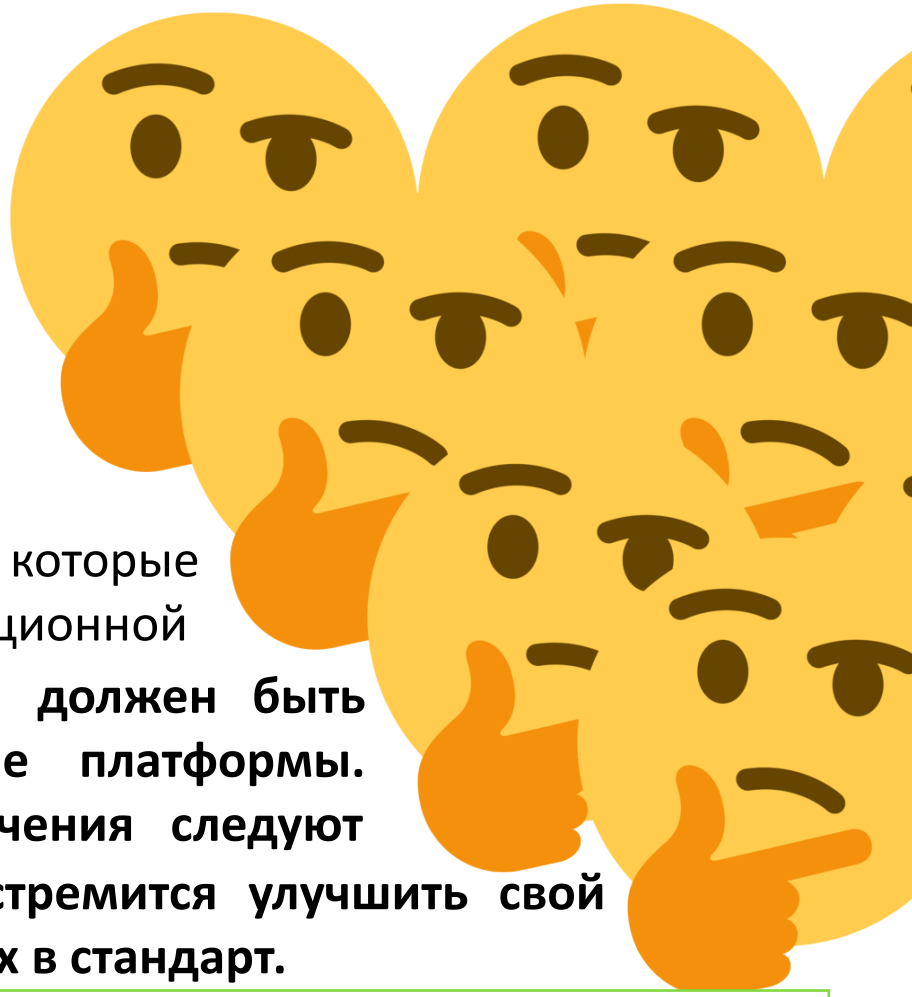
ИСТОРИЯ РАЗВИТИЯ SQL

В 1999 году появился новый стандарт, названный SQL3. В SQL3 введены новые типы данных, при этом предполагается возможность задания сложных структурированных типов данных, которые в большей степени соответствуют объектной ориентации. **Добавлен раздел, который вводит стандарты на события и триггеры, которые ранее не затрагивались** в стандартах, хотя давно уже широко использовались в коммерческих СУБД. В стандарте определены возможности четкой спецификации триггеров как совокупности события и действия. **В рамках управления транзакциями произошел возврат к старой модели транзакций,** допускающей точки сохранения (savepoints), и возможность указания в операторе отката ROLLBACK точек возврата позволит откатывать транзакцию не в начало, а в промежуточную ранее сохраненную точку.

Далее новые стандарты языка выходили в 2003 (SQL:2003), 2006 (SQL:2006), 2008 (SQL:2008), 2011 (SQL:2011), 2016 (SQL:2016). и последний стандарт вышел в 2019 году (SQL:2019). Стандарт SQL:2016 представил 44 новые опциональные функции. 22 из них принадлежат JSON функциональности, 10 других связаны с полиморфными табличными функциями. В стандарт SQL:2019 добавлена часть, определяющая работу с многомерными массивами.


THE FIRST VERSION OF THE STANDARD
1986
1989
SQL-86
SQL-89
SELECT
FROM
WHERE
GROUP BY
HAVING
INSERT
UPDATE
DELETE
CREATE TABLE
CREATE VIEW
FOUNDATION OF SQL
1992
SQL-92
JOIN
INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL JOIN
NATURAL JOIN
CROSS JOIN
UNION
INTERSECT
EXCEPT
CASE WHEN
CAST
ALTER TABLE
DROP TABLE
ALTER VIEW
DROP VIEW
DEVELOPMENT OF MODERN SQL
1999
SQL:1999
Recursive CTE
GROUP BY Extensions:
ROLLUP
CUBE
GROUPING SETS
MODERN SQL
2003
SQL:2003
Window Functions
2006-2019
SQL:2006
SQL:2008
SQL:2011
SQL:2016
SQL:2019
Minor Language Enhancements
LearnSQL
.com

А зачем вообще нужны эти стандарты?



Разработка любой информационной системы, ориентированной на технологию баз данных, **является трудоемким процессом, занимающим несколько десятков и даже сотен человеко-месяцев.** Развивается не только вычислительная техника, изменяются и реальные объекты, поведение которых моделируется использованием как самой БД, так и процедур обработки информации в ней, то есть конкретных приложений, которые составляют реальное наполнение разрабатываемой информационной системы. Именно поэтому **проект информационной системы должен быть рассчитан на расширяемость и переносимость на другие платформы. Большинство поставщиков аппаратуры и программного обеспечения следуют стратегии поддержки стандартов, но и каждый поставщик стремится улучшить свой продукт введением дополнительных возможностей, не входящих в стандарт.**

Выбор разработчиков: ориентироваться только на *экзотические* особенности данного продукта либо стараться в основном придерживаться стандарта. **Во втором случае весь интеллектуальный труд, вкладываемый в разработку, становится более защищенным, так как система приобретает свойства переносимости.** И в случае появления более перспективной платформы проект, ориентированный в большей степени на стандарты, может быть легче перенесен на нее.

ВВЕДЕНИЕ В SQL

Различают два вида языков запросов для реляционной модели данных. **Алгебраические языки** позволяют **выразить запросы средствами специальных операторов над отношениями** — операторов реляционной алгебры. **Языки реляционного исчисления** позволяют **описать запросы на языке исчисления предикатов** — языке математической логики.

Наиболее распространенным языком запросов реляционной модели данных является язык структурированных запросов SQL (Structured Query Language). SQL нельзя отнести к конкретному виду языков. Он содержит в себе возможности и языка реляционного исчисления (исчисления кортежей), и алгебраического языка и несомненно является реляционно полным.

SQL (Structured Query Language, или язык структурированных запросов) — это декларативный язык программирования (язык запросов), который используют для создания, обработки и хранения данных в реляционных БД.

СТРУКТУРА SQL

На чистом SQL нельзя написать программу — он предназначен только для взаимодействия с базами данных: получения, добавления, изменения и удаления информации в них, управления доступом и так далее.

Все SQL-команды делятся на четыре вида

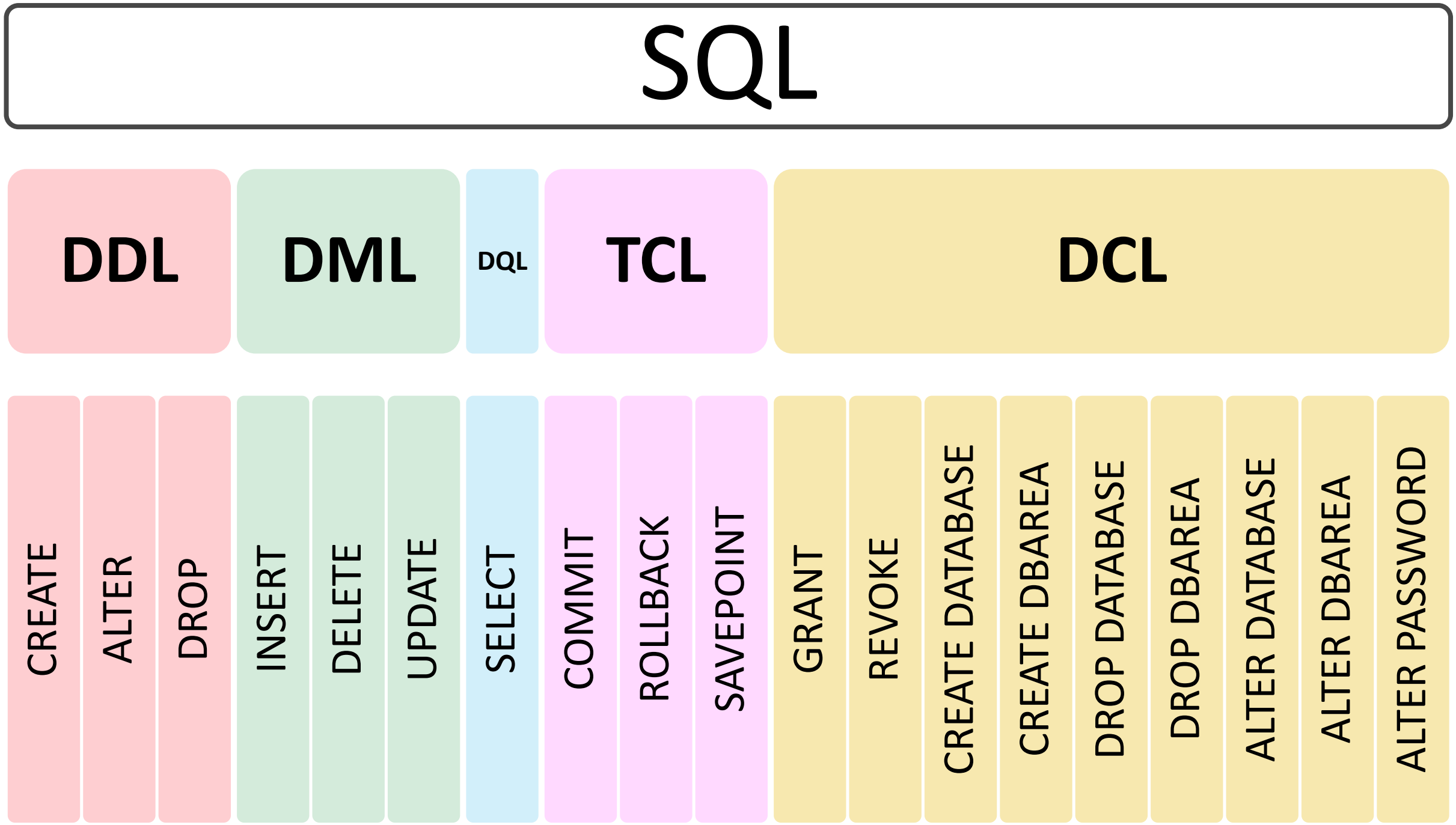
DDL (Data Definition Language, или язык описания данных). Их используют, чтобы создавать, изменять и удалять целые таблицы.

DML (Data Manipulation Language, или язык управления данными). Их применяют к содержимому таблиц, чтобы создавать, изменять, удалять атрибуты и записи.

DCL (Data Control Language, или язык контроля данных). Они нужны, чтобы выдавать конкретным пользователям доступ к базам данных и отзывать его.

Data Query Lanquaqe (Data Query Lanquaqe или язык запросов). Используется для выполнения запросов к данным.

TCL (Transaction Control Language, или язык контроля транзакций). Позволяет управлять транзакциями.


SQL
DDL
DML
DQL
TCL
DCL
CREATE
ALTER
DROP
INSERT
DELETE
UPDATE
SELECT
COMMIT
ROLLBACK
SAVEPOINT
GRANT
REVOKE
CREATE DATABASE
CREATE DBAREA
DROP DATABASE
DROP DBAREA
ALTER DATABASE
ALTER DBAREA
ALTER PASSWORD

DDL (ЯЗЫК ОПИСАНИЯ ДАННЫХ)

Описательный язык, который **позволяет описать и именовать сущности и атрибуты**, необходимые для работы некоторого приложения, а также **связи, имеющиеся между различными сущностями**, кроме того, **указать ограничения целостности** и защиты.

Оператор	Описание	Действие
`CREATE TABLE`	Создать таблицу	Создает новую таблицу в БД
`DROP TABLE`	Удалить таблицу	Удаляет таблицу из БД
`ALTER TABLE`	Изменить таблицу	Изменяет структуру существующей таблицы или ограничения целостности, задаваемые для данной таблицы
`CREATE VIEW`	Создать представление	Создает виртуальную таблицу, соответствующую некоторому SQL-запросу
`ALTER VIEW`	Изменить представление	Изменяет ранее созданное представление
`DROP VIEW`	Удалить представление	Удаляет ранее созданное представление
`CREATE INDEX`	Создать индекс	Создает индекс для некоторой таблицы для обеспечения быстрого доступа по атрибутам, входящим в индекс
`DROP INDEX`	Удалить индекс	Удаляет ранее созданный индекс

DML (ЯЗЫК МАНИПУЛИРОВАНИЯ ДАННЫМИ)

Язык, содержащий **набор операторов** для поддержки **основных операций манипулирования** содержащимися в базе данными. С помощью этих операторов **возможно добавлять, изменять, удалять**, т.е. манипулировать ими.

Оператор	Описание	Действие
DELETE	Удалить кортеж	Удаляет одну или несколько строк, соответствующих условиям фильтрации, из базовой таблицы. Применение оператора согласуется с принципами поддержки целостности, поэтому этот оператор не всегда может быть выполнен корректно, даже если синтаксически он записан правильно
INSERT	Вставить кортеж	Вставляет одну строку в базовую таблицу. Допустимы модификации оператора, при которых сразу несколько строк могут быть перенесены из одной таблицы или запроса в базовую таблицу
UPDATE	Обновить кортеж	Обновляет значения одного или нескольких столбцов в одной или нескольких строках, соответствующих условиям фильтрации

DQL (ЯЗЫК МАНИПУЛИРОВАНИЯ ДАННЫМИ)

Операторы DQL **используются для выполнения запросов к данным внутри объектов**. Цель команд DQL - получить некоторое отношение на основе переданного ей запроса. Язык включает в себя инструкцию SELECT.

Оператор	**Описание**	**Действие**
`SELECT`	Выбрать кортежи	Оператор, заменяющий все операторы реляционной алгебры и позволяющий сформировать результирующее отношение, соответствующее запросу

TCL (СРЕДСТВА УПРАВЛЕНИЯ ТРАНЗАКЦИЯМИ)

Транзакции группируют набор задач в единый блок выполнения. Каждая транзакция начинается с определенной задачи и заканчивается, когда все задачи в группе успешно завершены. Если какая-либо из задач завершается с ошибкой, транзакция завершается с ошибкой. Следовательно, транзакция имеет только два результата: успех или неудачу. **Для управления выполнением транзакции используются следующие команды TCL:**

Оператор	Описание	Действие
`COMMIT`	Завершить транзакцию	Завершить комплексную взаимосвязанную обработку информации, объединенную в транзакцию
`ROLLBACK`	Откатить транзакцию	Отменить изменения, проведенные в ходе выполнения транзакции
`SAVEPOINT`	Сохранить промежуточную точку выполнения транзакции	Сохранить промежуточное состояние БД, пометить его для того, чтобы можно было в дальнейшем к нему вернуться

DCL (ЯЗЫК КОНТРОЛЯ ДАННЫХ)

DCL включает в себя команды, которые в основном имеют дело с правами, разрешениями и другими элементами управления системой базы данных.

Оператор	Описание	Действие
GRANT	Предоставить права	Предоставить права доступа на ряд действий над некоторым объектом БД
REVOKE	Лишить прав	Лишить прав доступа к некоторому объекту или некоторым действиям над объектом
CREATE DATABASE	Создать БД	Создать новую базу данных, определив основные параметры для нее
CREATE DBAREA	Создать область хранения	Создать новую область хранения и сделать ее доступной для размещения данных
DROP DATABASE	Удалить БД	Удалить существующую базу данных (только в том случае, когда вы имеете право выполнить это действие)
DROP DBAREA	Удалить область хранения БД	Удалить существующую область хранения (если в ней на настоящий момент не располагаются активные данные)
ALTER DATABASE	Изменить БД	Изменить набор основных объектов в базе данных, ограничений, касающихся всей базы данных
ALTER DBAREA	Изменить область хранения БД	Изменить ранее созданную область хранения
ALTER PASSWORD	Изменить пароль	Изменить пароль для всей базы данных

ТИПЫ ДАННЫХ SQL

Типы данных SQL необходимы для организации работы всей базы. В таблице каждый столбец обязан иметь имя и указание на информацию, которая в нем содержится. Это определяет как язык, на котором написана БД, будет взаимодействовать с наполнением таблицы.

В каждой конкретной СУБД существуют свои типы данных. Более того, в некоторых системах **они могут иметь одинаковое название, но разные функции**, поэтому всегда **важно знакомиться с документацией** до начала работы.

Тип данных, которые будут храниться в столбце или в переменной, **необходимо определять заранее.** Это ограничит пользователя от возможности ввести любые неожиданные или недействительные данные.

ТИПЫ ДАННЫХ SQL

Типы данных

- Целочисленный тип данных SQL
- Вещественный тип данных SQL
- Типы данных даты и времени SQL
- Строковый тип данных SQL
- Текстовые и бинарные
- JSON

Типы данных в PostgreSQL 15

Типы данных в Firebird 4.0

Типы данных в Oracle Database (Release 19)

Типы данных в MySQL 8.0

ОПЕРАТОР ВЫБОРА SELECT

Язык запросов (Data Query Language) в SQL состоит из единственного оператора *SELECT*. Этот единственный оператор поиска реализует все операции реляционной алгебры. Один и тот же запрос может быть реализован несколькими способами, и, будучи все правильными, они, тем не менее, могут существенно отличаться по времени исполнения, и это особенно важно для больших баз данных.

Синтаксис оператора SELECT имеет следующий вид:

```
SELECT[ALL|DISTINCT](<Список полей>|*)
FROM      <Список таблиц>
[WHERE     <Предикат-условие выборки или соединения>]
[GROUP BY  <Список полей результата>]
[HAVING    <Предикат-условие для группы>]
[ORDER BY  <Список полей, по которым упорядочить вывод>]
```

Ключевое слово **ALL** означает, что **в результирующий набор строк включаются все строки**, удовлетворяющие условиям запроса.

Синтаксис оператора SELECT имеет следующий вид:

```
SELECT[ALL|DISTINCT](<Список полей>|*)
FROM      <Список таблиц>
[WHERE     <Предикат-условие выборки или соединения>]
[GROUP BY  <Список полей результата>]
[HAVING    <Предикат-условие для группы>]
[ORDER BY  <Список полей, по которым упорядочить вывод>]
```

Ключевое слово **DISTINCT** означает, что в **результирующий набор включаются только различные строки**, то есть дубликаты строк результата не включаются в набор.

Символ ***. (звездочка)** означает, что в **результирующий набор включаются все столбцы из исходных таблиц запроса.**

В разделе **FROM** задается **перечень исходных отношений** (таблиц) запроса.

В разделе **WHERE** задаются **условия отбора строк результата или условия соединения кортежей исходных таблиц**, подобно операции условного соединения в реляционной алгебре.

В выражении условий раздела WHERE могут быть использованы следующие предикаты:

- **предикаты сравнения** { =, <>, >,<, >=,<= },которые имеют традиционный смысл;
- **предикат `Between A and B`** —принимает значения между A и B. Предикат истинен, когда сравниваемое значение попадает в заданный диапазон, включая границы диапазона. *Одновременно в стандарте задан и противоположный предикат **Not Between A and B**, который истинен тогда, когда сравниваемое значение не попадает в заданный интервал, включая его границы*;
- **предикат вхождения в множество IN** (множество) истинен тогда, когда сравниваемое значение входит в множество заданных значений. *Одновременно существует противоположный предикат **NOT IN** (множество), который истинен тогда, когда сравниваемое значение не входит в заданное множество;*
- **предикаты сравнения с образцом `LIKE` и `NOT LIKE`.** Предикат `LIKE` требует задания шаблона, с которым сравнивается заданное значение, предикат истинен, если сравниваемое значение соответствует шаблону, и ложен в противном случае. Предикат **`NOT LIKE`** имеет противоположный смысл;
- **предикат сравнения с неопределенным значением IS NULL**;
- **предикаты существования `EXISTS` и несуществования `NOT EXISTS`.** Эти предикаты относятся к встроенным подзапросам.

Синтаксис оператора SELECT имеет следующий вид:

```
SELECT[ALL|DISTINCT](<Список полей>|*)
FROM      <Список таблиц>
[WHERE     <Предикат-условие выборки или соединения>]
[GROUP BY   <Список полей результата>]
[HAVING     <Предикат-условие для группы>]
[ORDER BY   <Список полей, по которым упорядочить вывод>]
```

В разделе **GROUP BY** задается **список полей группировки**.

В разделе **HAVING** задаются **предикаты-условия, накладываемые на каждую группу**.

В части **ORDER BY** задается **список полей упорядочения результата**, то есть список полей, который определяет порядок сортировки в результирующем отношении.

SELECT — ключевое слово, которое сообщает СУБД, что эта команда — запрос. Все запросы начинаются этим словом с последующим пробелом. **За ним может следовать способ выборки** — с удалением дубликатов (DISTINCT) или без удаления (ALL, подразумевается по умолчанию). **Затем следует список перечисленных через запятую столбцов, которые выбираются запросом из таблиц, или символ '*'** (звездочка) для выбора всей строки. Любые **столбцы, не перечисленные здесь, не будут включены в результирующее отношение**, соответствующее выполнению команды.

FROM — ключевое слово, подобно SELECT, которое должно быть представлено в каждом запросе. Оно сопровождается пробелом и затем **именами таблиц, используемых в качестве источника информации.** В случае **если указано более одного имени таблицы, неявно подразумевается, что над перечисленными таблицами осуществляется операция декартова произведения.** Таблицам можно присвоить имена-псевдонимы, что бывает полезно для осуществления операции соединения таблицы с самой собою или для доступа из вложенного подзапроса к текущей записи внешнего запроса (вложенные подзапросы здесь не рассматриваются).

Все последующие разделы оператора SELECT являются необязательными.

Алгоритм работы базового оператора SELECT:

1) запоминаются условия выборки кортежей из отношения;

2) проверяется, какие кортежи удовлетворяют указанным свойствам. Такие кортежи запоминаются;

3) на выход выводятся перечисленные в первой строчке базовой структуры оператора SELECT атрибуты со своими значениями.

ID	COMPANY
23	Темная сторона
12	НеМолоко
12	АГРОМИЛК
13	Рот Фронт

```
SQL> select distinct id from post;

          ID
============
          12
          13
          23
```

```
SQL> select * from post;

          ID CPMPANY
============ ==============
          23 temnya storona
          23 pechenky
          13 milka
          12 agromilk
          13 RotFront
          12 NeMoloko
          12 <null>
          12 alpro
          13 alpro
```

```
SQL> select all cpmpany from post;

CPMPANY
====================
temnya storona
pechenky
milka
agromilk
RotFront
NeMoloko
<null>
alpro
alpro
```

УНАРНЫЕ ОПЕРАЦИИ НА ЯЗЫКЕ СТРУКТУРИРОВАННЫХ ЗАПРОСОВ

Операция проекции

Операция селекции (выборки)

Операция переименования

При применении операции проекции выбираются не строки, а столбцы. Поэтому **достаточно перечислить заголовки нужных столбцов** (т. е. имена атрибутов), **без указания каких-либо посторонних условий.**

```
SELECT Список имен атрибутов FROM Имя отношения;
```

Операция селекции (выборки) на языке SQL реализуется оператором SELECT следующего вида:

```
SELECT Список имен атрибутов FROM Имя отношения
WHERE условие выборки;
```

УНАРНЫЕ ОПЕРАЦИИ НА ЯЗЫКЕ СТРУКТУРИРОВАННЫХ ЗАПРОСОВ

Операция переименования атрибутов на языке структурированных запросов осуществляется по следующему алгоритму:

1) в списке имен атрибутов оператора SELECT перечисляются те атрибуты, которые необходимо переименовать;

2) к каждому указанному атрибуту добавляется специальное ключевое слово AS;

3) после каждого вхождения слова AS указывается то имя соответствующего атрибута, на которое необходимо поменять имя исходное.

```
SELECT имя атрибута 1 as новое имя атрибута 1, …
FROM Имя отношения;
```

ID	COMPANY
23	Темная сторона
12	НеМолоко
13	Милка
13	Рот Фронт

```
SQL> select id as nomerochki, cpmpany as company from post;

  NOMEROCHKI COMPANY
============ ====================
          23 temnya storona
          23 pechenky
          13 milka
          12 agromilk
          13 RotFront
          12 NeMoloko
          12 <null>
```