Элементы языка SQL (Structured Query Language)

Язык SQL появился в середине 70-х и в настоящее время является стандартным реляционным языком и поддерживается практически всеми продуктами работы с данными.

SQL включает два подъязыка:

- **DDL (data definition language) - язык определения данных (ЯОД)**

  Предназначен для определения структуры данных, которая фиксируется в виде схемы базы данных. Схема представляет собой метаданные , хранится как часть базы данных и может быть изменена без ущерба для фактографических данных, относящихся к предметной области.

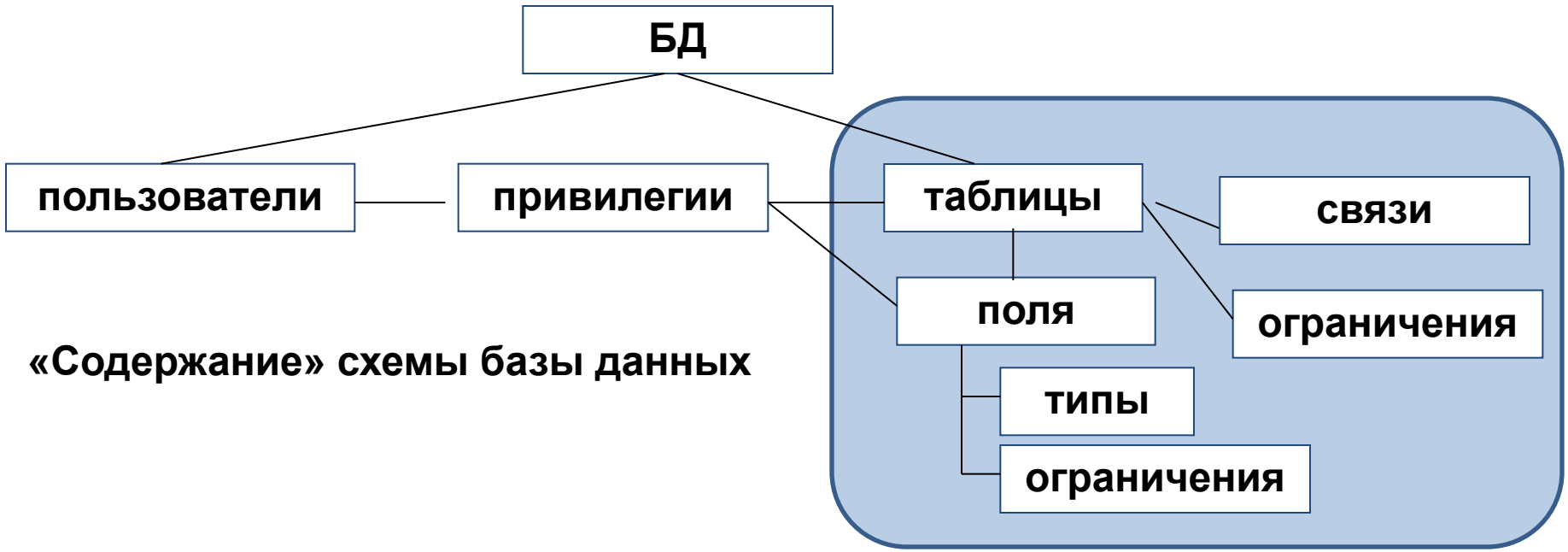
- **DML (data manipulation language) - язык манипулирования данными (ЯМД)**

  Предназначен для ведения (пополнение, изменения удаления) данных и реализации целевого регламентированного доступа к данным.

Схема БД

Схема базы данных представляет собой структурированную совокупность данных о структуре базы, заданных ограничениях и распределении привилегий (прав) пользователе на доступ как к самим данным, так и к структуре базы

В различных СУБД поддержка схемы реализована различным образом.



«Содержание» схемы базы данных

Основные типы данных:

В SQL существуют три основных типа данных:

Числа:

Целые числа могут иметь размер от байта до двойного машинного слова в зависимости от конкретного подтипа

Числа с плавающей запятой - представляют собой приблизительные значения, могут иметь одинарную и двойную точность.

Десятичные числа - имеют фиксированное количество цифр после запятой. Тип используется для хранения значений величин, для которых погрешности округлений недопустимы.

Строки (последовательности символов) бывают трех типов (***в MySQL***):
ASCII_строки (ограничены 255 символами)
большие двоичные объекты (может содержать до 16 Мбайт текста)
Перечисления (строки с предопределенным набором возможных значений).

Значения даты и времени – имеют ряд подтипов.
Наиболее общий подтип DATETIME (ГГГГ-ММ-ДД ЧЧ:ММ:СС) ,

ЯОД : Определение данных

Для определения структуры базы данных, просмотра и удаления компонентов структуры предназначены следующие операторы SQL

Инструкции работы с БД как единым объектом:

CREATE DATABASE *library* - создание ***пустой*** базы данных с именем *library*

DROP DATABASE - удаление БД вместе со всеми таблицами. **Необратимое действие !!!**

SHOW DATABASES – вывод списка имен существующих (доступных) баз данных

USE *library* – «активизация» (использование по умолчанию) базы данных с именем *library*

Инструкция создания таблиц БД **CREATE TABLE** в качестве аргументов требует имя таблицы и ее полное определение, состоящее из определений отдельных полей.

Полное имя таблицы **::=** ***Имя БД.Имя таблицы*** *(library. books)*

Полное имя поля **::=** ***Имя БД.Имя таб.Имя Поля*** *(library. books. Author)*

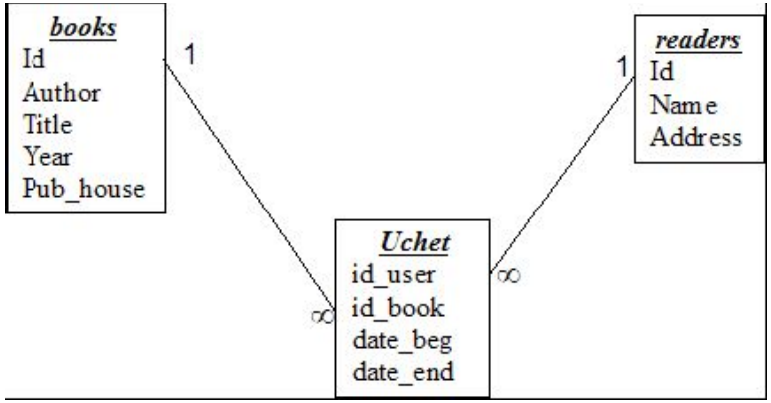
- создание ***пустой*** базы данных с именем *library*

Общий формат инструкции CREATE TABLE:

CREATE TABLE *имя_таблицы*

(определение столбца, …ограниченя)

Определение столбца включает в себя имя столбца и спецификацию его типа.



Структура БД ***library***

Пример создание таблицы средствами SQL

```
CREATE TABLE books (
id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
A_ship INTEGER UNSIGNED NOT NULL,
title VARCHAR(256) NOT NULL,
year INTEGER UNSIGNED NOT NULL,
pub_house VARCHAR(40) NOT NULL,
PRIMARY KEY(id))
```

Имя таблицы

Определения полей

Определение первичного ключа таблицы

```
CREATE TABLE uchet (
id_user INTEGER UNSIGNED NOT NULL,
id_book INTEGER UNSIGNED NOT NULL,
date_beg DATE NOT NULL,
date_end DATE NOT NULL,
PRIMARY KEY(id_user, id_book, date_beg),
FOREIGN KEY(id_user) REFERENCES readers(id)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY(id_book) REFERENCES books(id)
ON DELETE CASCADE ON UPDATE CASCADE
)
```

Определение внешних ключей таблицы

Описание ограничений:

Первичный ключ задается выражением
PRIMARY KEY *(имя поля [имя поля2,…])*

Для таблицы может быть задан только один первичный ключ, который может быть простым (одно поле) или составным (состоять из нескольких полей).

Потенциальный ключ задается выражением
UNIQUE (*имя поля [имя поля2,…]*)

Таблица может иметь несколько потенциальных ключей.

Внешний ключ задается выражением
FOREIGN KEY *(имя поля)* REFERENCES *имя_таблицы(имя_поля*)
ON DELETE *опция* ON UPDATE *опция*
Возможные ***опции*** : **NO ACTION; CASCADE; SET DEFAULT; SET NULL**

Используется для установления связи между таблицами

Проверочные условия могут задаваться с помощью фразы
CHECK *(условное выражение).*

Используются, как правило, для проверки ограничений, накладываемых предметной областью (бизнес-правилами)

Просмотр и изменение структуры БД:

Для просмотра существующей структуры данных используется инструкция SHOW

SHOW DATABASES - вывод списка имен существующих (доступных) баз данных

SHOW TABLES - возвращает список таблиц, существующих в указанной базе данных:

SHOW [OPEN] TABLES [FROM *база_данных*]

SHOW COLUMNS - возвращает описание столбцов таблицы.

SHOW COLUMNS FROM *таблица* [FROM *база_данных*]

SHOW CREATE TABLE - возвращает описание запроса, который необходим для создания указанной таблицы.

SHOW GRANTS выводит список привилегий, предоставленных пользователю:

SHOW GRANTS FOR *пользователь*

Для изменения структуры данных используется инструкция ALTER

ALTER TABLE *таблица спецификация_изменений*
[, спецификация изменений]

Спецификация изменений может касаться любого элемента спецификации таблицы, но для выполнения инструкции пользователь должен обладать соответствующими привилегиями и изменения не должны приводить к нарушению целостности

ЯМД : Манипулирование данными

В СУБД операции над данными выполняются с помощью **запросов**

Результатом выполнения запроса, в большинстве случаев, является отношение

Основными инструкциями SQL , используемыми в запросах являются следующие

SELECT извлекает строки из одной или нескольких таблиц

INSERT вставляет кортежи в отношение

DELETE удаляет кортежи из отношения

UPDATE обновляет значения указанных полей в кортеже.

Формат «полной» инструкции:

SELECT *имя_столбца,...* *(список выборки)*
[ALL | DISTINCT] *(опция «все» или «без повторов»)*
FROM *имя_таблицы, ...* *(источник строк)*
WHERE условие_отбора_1 *(ограничения)*
GROUP BY *имя_столбца,...* *(группировка для выполнения стат. функций)*
HAVING условие_отбора _2
ORDER BY *имя_столбца, ...* *(упорядочение вывода результата)*
LIMIT *лимит* *(ограничения числа выводимых строк результата)*

Обработка данных

Предложение **GROUP BY** - **Группировка результатов запроса**

Позволяет группировать записи, вошедшие в результаты запроса, с тем чтобы над ними можно было выполнять статистические функции. Все элементы списка возвращаемых столбцов должны иметь одно значение для каждой группы строк. В предложении SELECT запроса с группировкой разрешается указывать только ***столбцы группировки*** и ***статистические функции***, а также выражения с ними.

Предложение **HAVING** – **дополнительное условие, накладываемое на сгруппированные данные**

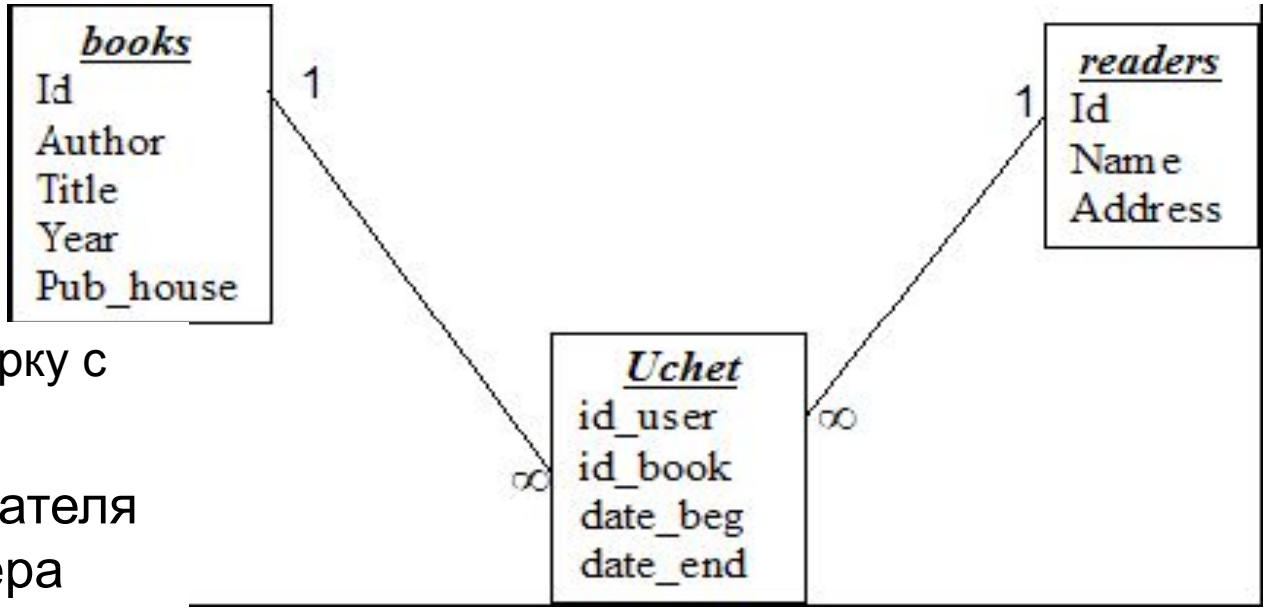
Предложение **ORDER BY** – **Упорядочение результатов запроса**

Содержит имена столбцов, по которым осуществляется сортировка результатов запроса. Если указано несколько столбцов, то их порядок определяет приоритет сортировки.

Предложение **LIMIT** – **Ограничение числа возвращаемых записей**

LIMIT 10 – выводятся не более 10-ти строк, начиная с 1-й;
LIMIT 7, 10 – выводятся не более 10-ти строк, начиная с 7-й

Обработка данных (примеры запросов)



Структура БД ***library***

Простой запрос на выборку с условием:

«Узнать фамилию читателя под номером 2 и номера книг, которые он брал»

Варианты реализации запроса:

```
SELECT id_book, name FROM uchet, readers
 WHERE readers.id=uchet.id_user AND readers.id=2;
```

```
SELECT id_book, name
FROM readers JOIN uchet ON readers.id=uchet.id_user
WHERE readers.id=2;
```

Обработка данных (примеры запросов)

Простой запрос на выборку с условием:

«Выбрать авторов и названия книг, которые брали читатели с номерами 9 или 19»

Варианты реализации запроса:

```
SELECT DISTINCT id_book, author, title
FROM from uchet, readers, books
WHERE readers.id=uchet.id_user AND books.id=uchet.id_book AND
                        (readers.id=9 OR readers.id=19);
```

```
SELECT DISTINCT id_book, author, title
FROM (readers JOIN uchet ON readers.id=uchet.id_user)
        JOIN books ON books.id=uchet.id_book
WHERE readers.id=9 OR readers.id=19;
```

В представленных вариантах имеется избыточность!

Использование вложенных запросов (подзапросы)

Вложенный оператор **SELECT** может использоваться только в частях **WHERE** и **HAVING** другого запроса !

«Получить фамилии и номера читателей, которые брали те же книги, что и читатель под номером 19, а также и номера этих книг»

Вариант реализации запроса:

```
SELECT id_book, id, name
FROM uchet, readers
WHERE (readers.id=uchet.id_user AND readers.id<>19 AND id_book IN
    (SELECT id_book
        FROM uchet, readers
        WHERE readers.id=uchet.id_user AND readers.id=19));
```

*Вложенный запрос возвращает отношение, имеющее одно поле **id_book** содержащее номера книг, которые брал читатель с номером 19*

Запросы с использованием вычисляемых полей

Использование вычисляемых полей позволяет получать в результате запроса данные, которые не хранятся в БД, но могут быть вычислены на основе хранимых данных

Пример: «Определить, на какое количество дней брались книги »

Вариант реализации запроса:

```
SELECT  id_book, date_beg, date_end, datediff (date_end, date_beg)
              AS delta
FROM uchet
```

id_book	date_beg	date_end	delta
24	10-09-2001	24-09-2001	14
126	18-03-2004	04-04-2004	17
235	01-10-2002	11-10-2002	10

*Поле **delta** вычисляется с помощью функции **datediff**, возвращающей количество дней, прошедших между двумя датами*

Запросы с использованием шаблонов

При использовании в условиях отбора операторов LIKE / NOT LIKE разрешается использовать шаблоны с символами-заместителями. В стандарте SQL символ «**%**» (*процент*) представляет произвольную последовательность символов, а символ «_» (*подчеркивание*) – любой одиночный символ.

Пример:

Выбрать авторов книг, выпущенных после 2000 года при условии, что поле author начинается с набора символов «Фред»

Вариант реализации запроса:

```
SELECT author
FROM books
WHERE author LIKE 'Фред%' AND year >2000
```

Author
Фредерик Браун
Фредерик Пол
Фред Саберхаген
Фредерик Браун и Мак Рейнольдс

Запросы с использованием обобщающих функций

COUNT (*имя_поля*) – подсчитывает количество записей в отношении;
MAX (*имя_поля*) – возвращает максимальное значение из заданного поля;
MIN (*имя_поля*) – возвращает минимальное значение из заданного поля;

Следующие функции используются только с данными числового типа

SUM (*имя_поля*) – возвращает сумму значений в заданном поле;
AVG (*имя_поля*) – возвращает среднее значение, вычисленное на основе данных заданного поля.

Использование обобщающих функций возможно только совместно с выражением **GROUP BY** *имя_поля*

Пример: «Получить список авторов, для которых имеется более 10 наименований книг, и дату издания самой ранней из представленных книг для каждого из этих авторов»

```
SELECT author, MIN(year), COUNT(id)
FROM books
GROUP BY author
HAVING COUNT(id)>10;
```

Author	MIN(year)	COUNT(ID)
"Айзек Азимов"	1952	13
"Гарри Гаррисон"	1953	16
"Роберт Шекли"	1954	39
"Роджер Желязны"	1950	17

Сложный запрос с обобщающей функцией и подзапросом

«Определить, сколько раз брали книги, которые брал читатель № 19. Учитывать только те книги, которые «брались» более одного раза»

Вариант реализации запроса:

```
SELECT id_book, COUNT(uchet.id_user) AS pop
FROM uchet, readers
WHERE readers.id=uchet.id_user
GROUP by id_book
HAVING pop>1 AND id_book IN
    (SELECT id_book
        FROM uchet, readers
        WHERE readers.id=uchet.id_user AND readers.id=19));
```

*Вложенный запрос возвращает отношение, имеющее одно поле **id_book** содержащее номера книг, которые брал читатель с номером 19*