SQL (Structured Query Language) Структурированный язык запросов

Состоит из трех частей:

- DDL (Data Definition Language) – язык определения данных. Предназначен для создания базы данных (таблиц, индексов и т.д.) и редактирования ее схемы.
- DCL (Data Control Language) – язык управления данными. Содержит операторы для разграничения доступа пользователей к объектам базы данных.
- DML (Data Manipulation Language) – язык обработки данных. Содержит операторы для внесения изменений в содержимое таблиц базы данных.

Синтаксис оператора SELECT

SELECT [ALL/DISTINCT] <список атрибутов>/*

FROM <список таблиц>

[WHERE <условие выборки>]

[ORDER BY <список атрибутов>]

[GROUP BY <список атрибутов>]

[HAVING <условие>]

[UNION <выражение с оператором SELECT>]

В квадратных скобках указываются элементы, которые могут в запросе отсутствовать

Использование языка SQL для выбора информации из таблицы

В *теории реляционных баз данных* существуют формальные методы построения реляционной модели *базы данных*, в которой отсутствует *избыточность* и *аномалии обновления*, удаления и включения.

Построение рационального варианта схем отношений осуществляется путем так называемой нормализации схем отношений. *Нормализация* производится в несколько этапов. На начальном этапе схема отношений должна находиться в первой *нормальной форме* (*1НФ*).

Отношение находится в первой нормальной форме, если все атрибуты отношения принимают простые значения (атомарные или неделимые), не являющиеся множеством или кортежем из более элементарных составляющих.

Последовательный переход от одной нормальной формы к другой при нормализации схем отношений реализуется через декомпозицию. Основной операцией, с помощью которой осуществляется декомпозиция, является проекция.

Декомпозицией схемы отношения R = {A1, A2, ...,An} называется замена ее совокупностью подмножеств R, таких, что их объединение дает R. При этом допускается, чтобы подмножества были пересекающимися.

При нормализации необходимо выбирать такие декомпозиции, которые обладают свойством соединения без потерь. В этом случае, декомпозиция должна обеспечить то, что запросы (выборка данных по условию) к исходному отношению и отношениям, получаемым в результате декомпозиции, дадут одинаковый результат.

Рассмотрим следующий пример.

Таблица представляет сущность ЭКЗАМЕНАЦИОННАЯ ВЕДОМОСТЬ

Код студента	Фамилия	Код экзамена	Предмет и дата	Оценка
1	Сергеев	1	Математика 5.06.18	4
2	Иванов	1	Математика 5.06.18	5
1	Сергеев	2	Физика 9.06.18	5
2	Иванов	2	Физика 9.06.18	5

Преобразуем таблицу к первой нормальной форме:

Код студента	Фамилия	Код экзамена	Предмет	Дата	Оценка
1	Сергеев	1	Математика	5.06.18	4
2	Иванов	1	Математика	5.06.18	5
1	Сергеев	2	Физика	9.06.18	5
2	Иванов	2	Физика	9.06.18	5

Отношение приведено к первой *нормальной форме.* Ключом данного отношения будет совокупность атрибутов – Код студента и Код экзамена.

Для более краткой записи процесса нормализации введем следующие обозначения:

КС – код студента, КЭ – код экзамена, Ф – фамилия, П – предмет, Д – дата, О - оценка.

Выпишем *функциональные зависимости*

- КС, КЭ -> Ф, П, Д, О
- КС, КЭ -> Ф
- КС, КЭ -> П
- КС, КЭ -> Д
- КС, КЭ -> О
- КЭ -> П
- КЭ -> Д
- КС -> Ф

- Рассмотрим отношение R(КС, Ф, КЭ, П, Д, О).

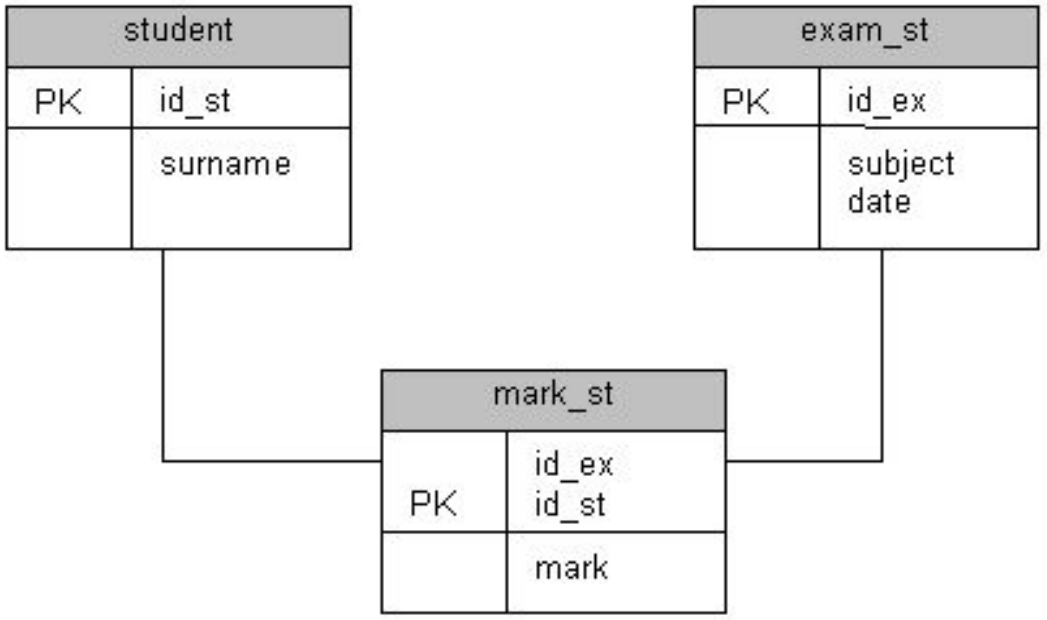
Отношение находится во второй нормальной форме (2НФ), если оно находится в 1НФ и каждый не ключевой атрибут зависит от первичного ключа и не зависит от части ключа.

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы, в отдельные таблицы.
https://habr.com/ru/post/254773/

Здесь атрибуты П, Д, Ф зависят от части ключа. Чтобы избавиться от этих зависимостей необходимо произвести декомпозицию отношения.

В результате исходное отношение будет приведено к трем отношениям, каждое из которых находится в 3НФ (третьей нормальной форме) R1(КС, Ф), R2(КЭ, П, Д), R3(КС, КЭ, О)

ER-диаграмма предметной области



Отношение `R1` представляет *объект* `student` с атрибутами `id_st` (*первичный ключ*), surname.

Отношение `R2` представляет *объект* `exam_st` с атрибутами `id_ex` (*первичный ключ*), *subject*, *date*.

Отношение `R3` представляет *объект* `mark_st` с атрибутами `id_st` (*внешний ключ*), `id_ex` (*внешний ключ*), *mark*. *Первичный ключ* здесь `id_st`, `id_ex`.

Рассмотрим реализацию запросов для этого примера

Выдать список всех студентов

```
SELECT *
FROM student
```

или

```
SELECT id_st, surname
FROM student
```

Если добавить к данному запросу предложение ORDER BY surname, то список будет упорядочен по фамилии.

По умолчанию подразумевается, что сортировка производится по возрастанию.

Если необходимо упорядочение по убыванию, после имени атрибута добавляется слово DESC.

Выдать список оценок, которые получил студент с кодом "1"

```
SELECT id_st, mark

FROM mark_st

WHERE id_st = 1
```

Выдать список кодов студентов, которые получили на экзаменах хотя бы одну двойку или тройку

```
SELECT id_st, mark

FROM mark_st

WHERE ( mark >= 2 ) AND ( mark <= 3 )
```

В предложении WHERE можно записывать выражение с использованием арифметических операторов сравнения (<, >, и т.д.) и логических операторов (AND, OR, NOT) как и в обычных языках программирования.

Наряду с операторами сравнения и логическими операторами для составления условий в языке SQL (из-за специфики области применения) существуют ряд специальных операторов, которые, как правило, не имеют аналогов в других языках. Это операторы:

- IN – вхождение в некоторое множество значений;
- BETWEEN – вхождение в некоторый диапазон значений;
- LIKE – проверка на совпадение с образцом;
- IS NULL – проверка на неопределенное значение.

Выдать список кодов студентов, которые получили на экзаменах хотя бы одну двойку или тройку

```
SELECT id_st, mark
FROM mark_st
WHERE ( MARK >= 2 ) AND ( MARK <= 3 )
```

```
SELECT id_st, mark
FROM mark_st
WHERE mark IN (2,3)
```

```
SELECT id_st, mark
FROM mark_st
WHERE mark BETWEEN 2 AND 3
```

Выдать список всех студентов, фамилии которых начинаются с буквы А

```
SELECT id_st, surname
FROM student
WHERE surname LIKE 'A%’
```

Оператор LIKE применим исключительно к символьным полям и позволяет устанавливать, соответствует ли значение поля образцу.

Образец может содержать специальные символы:

_ (символ подчеркивания) – замещает любой одиночный символ;

% (знак процента) – замещает последовательность любого числа символов.

АГРЕГАТНЫЕ ФУНКЦИИ SQL

- MIN – минимальное значение в столбце;
- MAX – максимальное значение в столбце;
- SUM – сумма значений в столбце;
- AVG – среднее значение в столбце;
- COUNT – количество значений в столбце, отличных от NULL

запрос считает среднее среди всех баллов, полученных студентами на экзаменах

```
SELECT  AVG(mark)
FROM   mark_st
```

запрос вычислит средний балл студента с кодом 100 по результатам всех сданных им экзаменов

```
SELECT AVG(mark)
FROM  mark_st
WHERE id_st = 100
```

запрос вычислит средний балл студентов по результатам сдачи экзамена с кодом 10

```
SELECT  AVG(mark)
FROM  mark_st
WHERE id_ex = 10
```

В дополнение к рассмотренным механизмам язык SQL предоставляет мощный аппарат для вычисления агрегатных функций не для всей таблицы результатов запроса, а для разных значений по группам. Для этого в SQL существует специальная конструкция GROUP BY, предназначенная для указания того столбца, по значениям которого будет производиться группировка. Группировка может производиться более чем по одному полю

запрос вычисляет средний балл по всем экзаменам для каждого студента

```
SELECT   id_st, AVG(mark)
FROM     mark_st
GROUP BY id_st
```

запрос, который вычисляет средний балл по оценкам, полученным на экзамене с кодом 100, для каждого студента

```
SELECT  id_st, AVG(mark)
FROM  mark_st
WHERE id_ex = 100
GROUP BY  id_st
```