

Базы данных

Лекция 12

SQLite



Характеристики СУБД

- ▶ архитектура;
- ▶ носители данных;
- ▶ SQL/noSQL;
- ▶ стандарты;
- ▶ Модель данных;
- ▶ ограничения целостности;
- ▶ объекты БД;
- ▶ встроенная процедурная система программирования;
- ▶ программные объекты;
- ▶ программные интерфейсы;
- ▶ типы данных;



SQLite

- ▶ встроенная, однопользовательская
- ▶ файловая система: БД – это обычно 1 файл
- ▶ ограниченный SQL
- ▶ ограниченный SQL92
- ▶ не строго типизируемая
- ▶ есть некоторые объекты (индексы, view)
- ▶ встроенной системы программирования нет
- ▶ программные объекты: триггеры
- ▶ программный интерфейс: библиотека функций



DbBrowser for SQLite

The screenshot shows the interface of DbBrowser for SQLite. At the top, there's a menu bar with 'Файл' (File), 'Редактирование' (Edit), 'Вид' (View), and 'Справка' (Help). Below the menu is a toolbar with icons for creating a new database, opening an existing one, saving changes, and undoing changes.

The main window has tabs at the top: 'Структура БД' (Structure), 'Данные' (Data), 'Прагмы' (Pragms), and 'SQL'. The 'Структура БД' tab is selected, showing a tree view of the database structure:

- Таблицы (5):
 - CUSTOMERS: CREATE TABLE CUSTOMERS (CUST_N...
 - OFFICES: CREATE TABLE OFFICES (OFFICE INTI...
 - ORDERS: CREATE TABLE ORDERS (ORDER_NUM...
 - PRODUCTS: CREATE TABLE PRODUCTS (MFR_ID C...
 - SALESREPS: CREATE TABLE SALESREPS (EMPL_NU...
- Индексы (0)
- Представления (0)
- Тrigгеры (0)

To the right of the tree view is a large text area for editing SQL or text. Above this area are buttons for 'Режим:' (Mode) set to 'Текст' (Text), 'Импортировать' (Import), 'Экспортировать' (Export), and 'Присвоить NULL' (Set to NULL). Below the text area, it says 'Тип данных в ячейке: NULL' (Data type in cell: NULL) and '0 байтов' (0 bytes).

At the bottom right is a 'Применить' (Apply) button. Below the main window is a 'Remote' panel with a table:

| Имя | Commit | Last modified | Size |
|-----|--------|---------------|------|
| | | | |

At the very bottom of the interface are tabs for 'Журнал SQL' (SQL Log), 'График' (Scheduler), 'Схема БД' (Schema), and 'Remote'. On the far right, it says 'UTF-8'.

Класс хранения

- ▶ Не типы данных, а класс хранения
- ▶ В любом классе можно хранить данные любого типа
- ▶ Можно указывать любые типы данных
- ▶ Или не указывать никакой тип



SQLite классы хранения

- ▶ **NULL** (пустое значение)
- ▶ **INTEGER** (1,2,3,4,5,8 – байтов, целые числа)
- ▶ **REAL** (IEEE-752)
- ▶ **TEXT** (UTF-8, UTF-16)
- ▶ **BLOB**



Аффинированные типы

- ▶ Аффинированный тип — это тот, который является рекомендуемым для сохраняемых в столбце значений
- ▶ Не является обязательным
- ▶ Аффинированность столбца определяется неявно во время создания таблицы при объявлении типа данных, который будет храниться в столбце



Аффинированные типы

- ▶ Определение столбца содержит:
 - ▶ TEXT (CLOB, CHAR, TEXT)
 - ▶ INTEGER (INT)
 - ▶ REAL (REAL, FLOAT, DOUBLE)
 - ▶ NONE (BLOB или не указан)
 - ▶ NUMERIC(в остальных случаях)



TEXT

- ▶ предназначены для хранения значений с классом NULL, TEXT или BLOB
- ▶ Если добавить в столбец с аффинированным типом данных TEXT числовое значение, то оно будет преобразовано в строку



NUMERIC, INTEGER, REAL

- ▶ Могут использоваться для хранения значений всех классов хранения
- ▶ Если поместить текстовое значение, то SQLite постарается преобразовать его в INTEGER или REAL
- ▶ Если преобразование не может произойти, то SQLite сохранит значение с классом TEXT
- ▶ Для REAL – целочисленные значения будут преобразованы в числа с плавающей точкой



BLOB

- ▶ не предпочитает какой-либо класс хранения
- ▶ не предпринимает попыток преобразования данных из одного класса в другой



Аффинированные типы

- ▶ Допустимо сравнивать значения с разными типами данных
- ▶ Нет булевого типа данных – целые числа 0 (`false`) и 1 (`true`)
- ▶ Нет типа данных даты и времени – хранится тип `TEXT` в виде строки формата: «`YYYY-MM-DD HH:MM:SS.SSS`»
- ▶ `REAL` как числа юлианского календаря - число дней с полуночью 24 ноября 4714 г. до н.э.
- ▶ `INTEGER` как время Unix - количество секунд с 1970-01-01 00:00:00 UTC.



Операторы сравнения

- ▶ «=», «==», «<», «<=», «>», «>=», «!=», «<>», «BETWEEN», «IN», «IS», «NOT IN» и «IS NOT»
- ▶ Значения с классом NULL считаются меньше любого другого значения, даже другого значения NULL
- ▶ Значение INTEGER или REAL считается меньше, чем значение TEXT или BLOB
- ▶ При сравнении INTEGER или REAL используется числовое сравнение
- ▶ Значение TEXT меньше значения BLOB
- ▶ Сравнение значений BLOB определяется с использованием функции memcmp()



Операторы сравнения

- ▶ Выражение, которое является ссылкой на значение столбца, имеет такую же аффинированность, как и столбец.
- ▶ Выражение "CAST (*выражение AS тип*)" имеет ту же аффинированность
- ▶ {INTEGER REAL NUMERIC} и {TEXT BLOB} → NUMERIC;
- ▶ {TEXT} и { } → TEXT;
- ▶ В остальных случаях преобразование не применяется



Операторы сравнения

- ▶ Все математические операторы преобразуют operandы в NUMERIC до выполнения операций
- ▶ Преобразование происходит даже в том случае, если оно необратимое и с потерями
- ▶ Operand NULL приводит к пустому (NULL) результату любую математическую операцию
- ▶ Operand в математической операции, который не приводится каким-либо образом к числовому и не является NULL, преобразуется в 0 или 0.0.



Операторы сравнения

- ▶ $a \text{ BETWEEN } b \text{ AND } c$
 - ▶ $a \geq b \text{ AND } a \leq c$
 - ▶ в каждом из сравнений применяются различные аффинированности



SELECT

Структура БД Данные Прагмы SQL

SQL 1 X

```
1 select * from SALESREPS
```

| | EMPL_NUM | NAME | AGE | REP_OFFICE | TITLE | HIRE_DA |
|---|----------|-------------|-----|------------|-----------|-----------|
| 1 | 106 | Sam Clark | 52 | 11 | VP Sales | 2006-06-1 |
| 2 | 109 | Mary Jones | 31 | 11 | Sales Rep | 2007-10-1 |
| 3 | 104 | Bob Smith | 33 | 12 | Sales Mgr | 2005-05-1 |
| 4 | 108 | Larry Fitch | 62 | 21 | Sales Mgr | 2007-10-1 |
| 5 | 105 | Bill Adams | 37 | 13 | Sales Rep | 2006-02-1 |
| 6 | 102 | Sue Smith | 48 | 21 | Sales Rep | 2004-12-1 |
| 7 | 101 | Dan Roberts | 45 | 12 | Sales Rep | 2004-10-2 |

10 строки возвращены за 0мс из: select * from SALESREPS

SQLite: последовательность секций

SELECT DISTINCT список столбцов

INTO новая таблица

FROM источник строк

WHERE логическое условие – фильтр строк

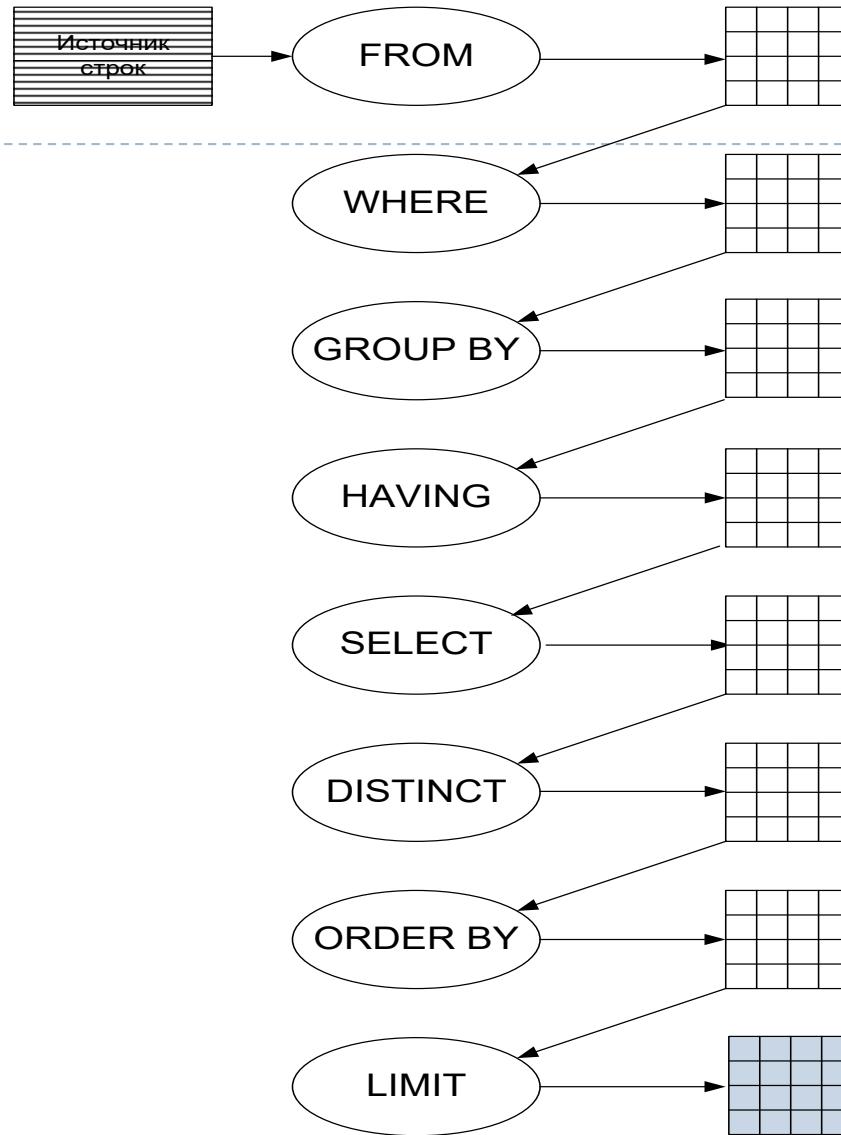
GROUP BY выражение для группировки строк

HAVING логическое условие – фильтр строк

ORDER BY выражение для сортировки ASC DESC

LIMIT количество строк





SQLite

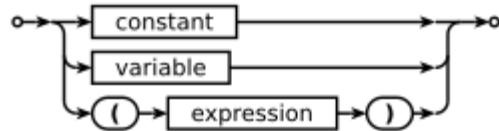
Ограничения целостности



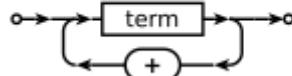
Railroad diagram

- ▶ Синтаксическая диаграмма — это направленный граф с одним входным ребром и одним выходным ребром и помеченными вершинами

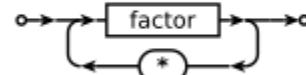
factor:



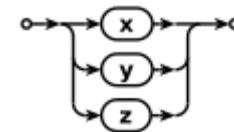
expression:



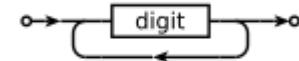
term:



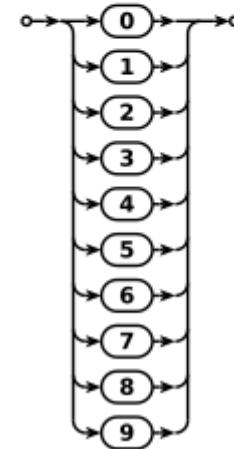
variable:



constant:



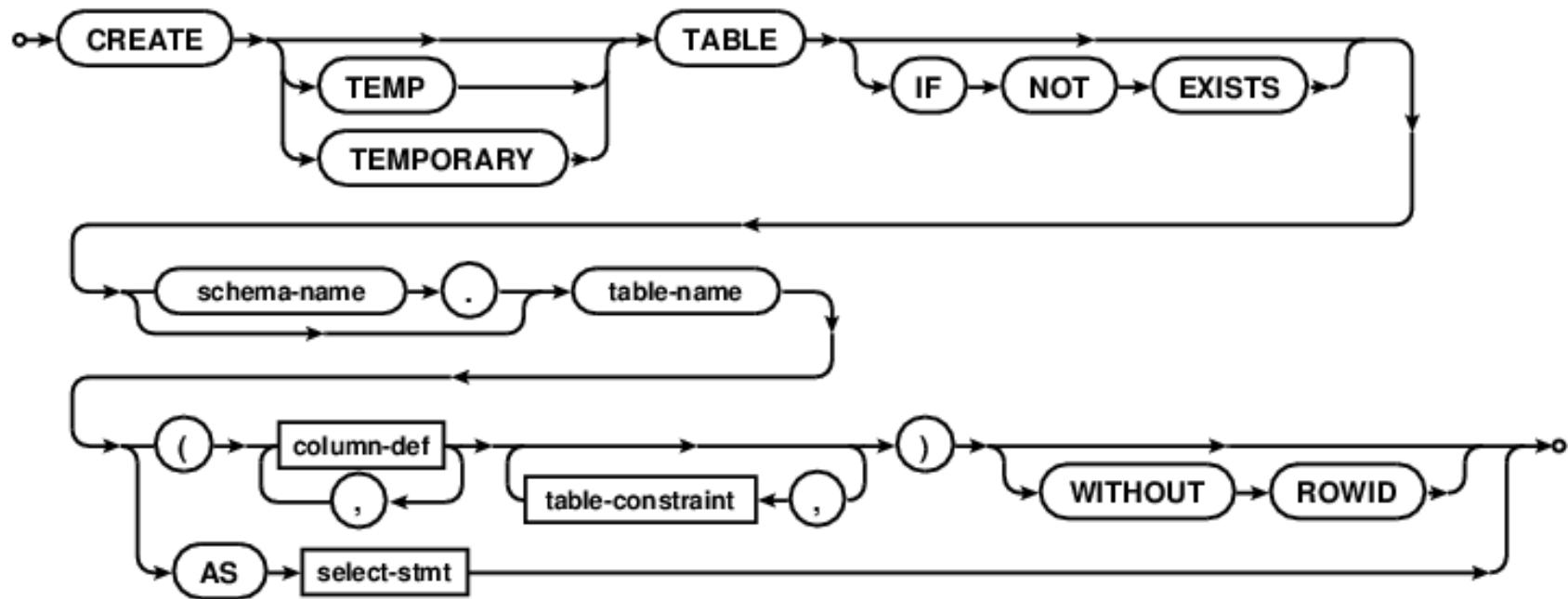
digit:



Create table

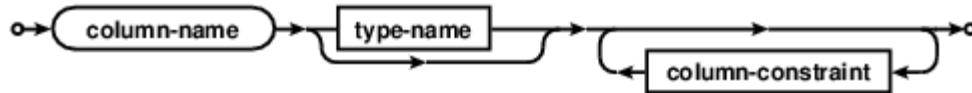
create-table-stmt:

hide



column-def:

hide

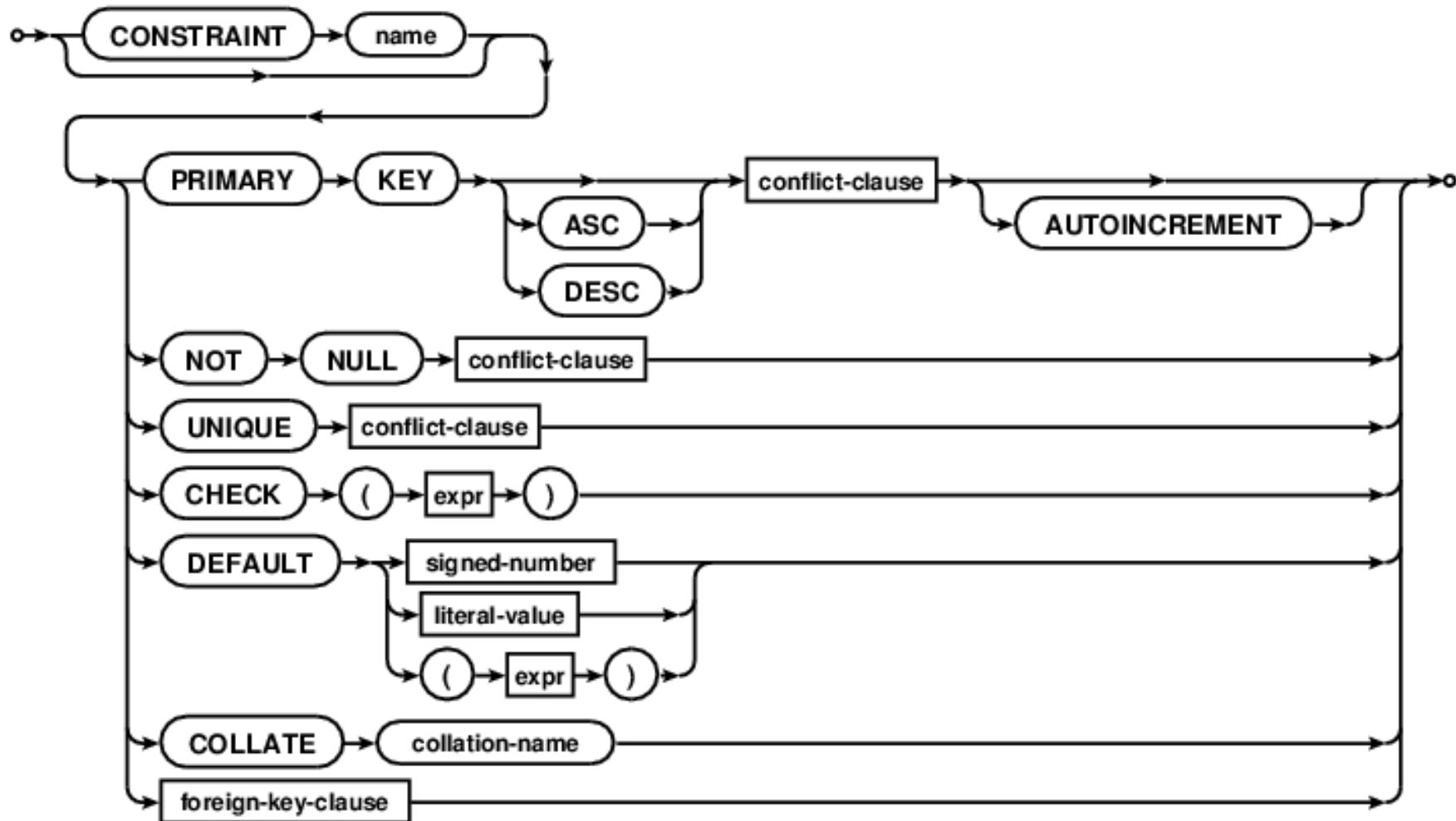


Create table

- ▶ Обязательно имя таблицы
- ▶ Имена с `sqlite_` зарезервированы
- ▶ `TEMP` или `TEMPORARY` создает временную таблицу
- ▶ `IF NOT EXISTS` создает, если такого нет
- ▶ Нельзя таблицу, представление и индекс с одинаковыми именами, а триггер можно
- ▶ Таблица удаляется `DROP TABLE`



CONSTRAINT



ROWID

- ▶ ROWID – уникальный номер записи в каждой таблице
- ▶ В запросе `SELECT * from table` не показывается
- ▶ Можно посмотреть значение по имени rowid
- ▶ Записи в таблице – B-дерево по rowid, => есть индекс

The screenshot shows the Oracle SQL Developer interface. The top menu bar has tabs: Структура БД, Данные, Прагмы, and SQL. The SQL tab is selected. Below the menu is a toolbar with icons for new connection, save, refresh, and execute. A SQL editor window titled "SQL 1" contains the query: `select rowid, empl_num from SALESREPS`. To the right of the editor is a results grid displaying the data from the query. The grid has two columns: "rowid" and "EMPL_NUM". The data is as follows:

| | rowid | EMPL_NUM |
|---|-------|----------|
| 1 | 7 | 101 |
| 2 | 6 | 102 |
| 3 | 9 | 103 |
| 4 | 3 | 104 |
| 5 | 5 | 105 |
| 6 | 1 | 106 |
| 7 | 10 | 107 |
| 8 | 4 | 108 |

At the bottom of the results grid, a message states: "10 строки возвращены за 1мс из: select rowid, empl_num from SALESREPS".

PRIMARY KEY

- ▶ UNIQUE+NOT NULL
- ▶ По умолчанию автоинкремент на единицу
- ▶ Значения ключа совпадают с ROWID
- ▶ Может быть составным



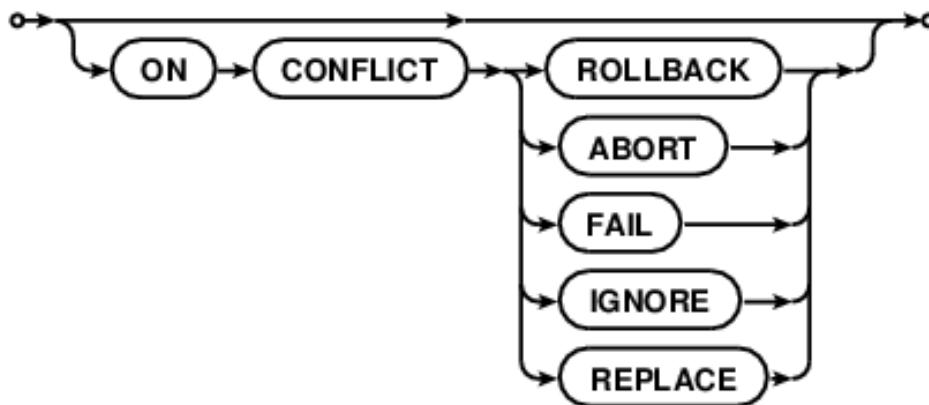
WITHOUT ROWID

- ▶ Можно создать таблицу с опцией WITHOUT ROWID
- ▶ Каждая таблица WITHOUT ROWID должна иметь PRIMARY KEY
- ▶ Нет механизма автоинкремента
- ▶ Используются для оптимизации хранения и обработки
- ▶ Нельзя использовать ROWID в SELECT - ошибка



ON CONFLICT

- ▶ Метод разрешения конфликтов при вставке
 - ▶ ROLLBACK
 - ▶ ABORT
 - ▶ FAIL
 - ▶ IGNORE
 - ▶ REPLACE



ROLLBACK

- ▶ Генерирует ошибку
- ▶ Выполняет ROLLBACK для всей транзакции



ABORT

- ▶ Генерирует ошибку
- ▶ Выполняет ROLLBACK только для текущего оператора



FAIL

- ▶ Генерирует ошибку
- ▶ Выполняет ROLLBACK только последней операции в рамках оператора
 - ▶ если UPDATE сделал 99 обновлений и на 100 сработал констрайнт, то 99 обновлений сохраняются



IGNORE

- ▶ Сбрасывается текущая операция
- ▶ Оператор продолжает свою работу



REPLACE

- ▶ Для ограничений PRIMARY KEY и UNIQUE: при INSERT или UPDATE удаляет конфликтную строку и добавляет новую
- ▶ Для ограничения NOT NULL: заменяет на значение по умолчанию, если нет такого значения, то ABORT
- ▶ Для ограничения CHECK: ABORT



UNIQUE

- ▶ уникальные значения в столбце или NULL
- ▶ допускается 2 и более NULL в одном столбце – все NULL разные



COLLATE

- ▶ COLLATE позволяет задать способ сравнения данных



COLLATE

- ▶ **BINARY** — побайтовое сопоставление строк данных независимо от кодировки текста
- ▶ **NOCASE** — так же, как BINARY, кроме 26-ти прописных букв ASCII, которые перед сортировкой переводятся в свои эквиваленты в нижнем регистре
- ▶ **RTRIM** — так же, как BINARY, но игнорируются пробелы в конце строки



COLLATE

- ▶ Каждый столбец каждой таблицы имеет связанную с ним функцию сортировки
- ▶ Если функция не определена явно, то по умолчанию используется BINARY
- ▶ Можно сравнивать строки в разных *collation*, но не нужно



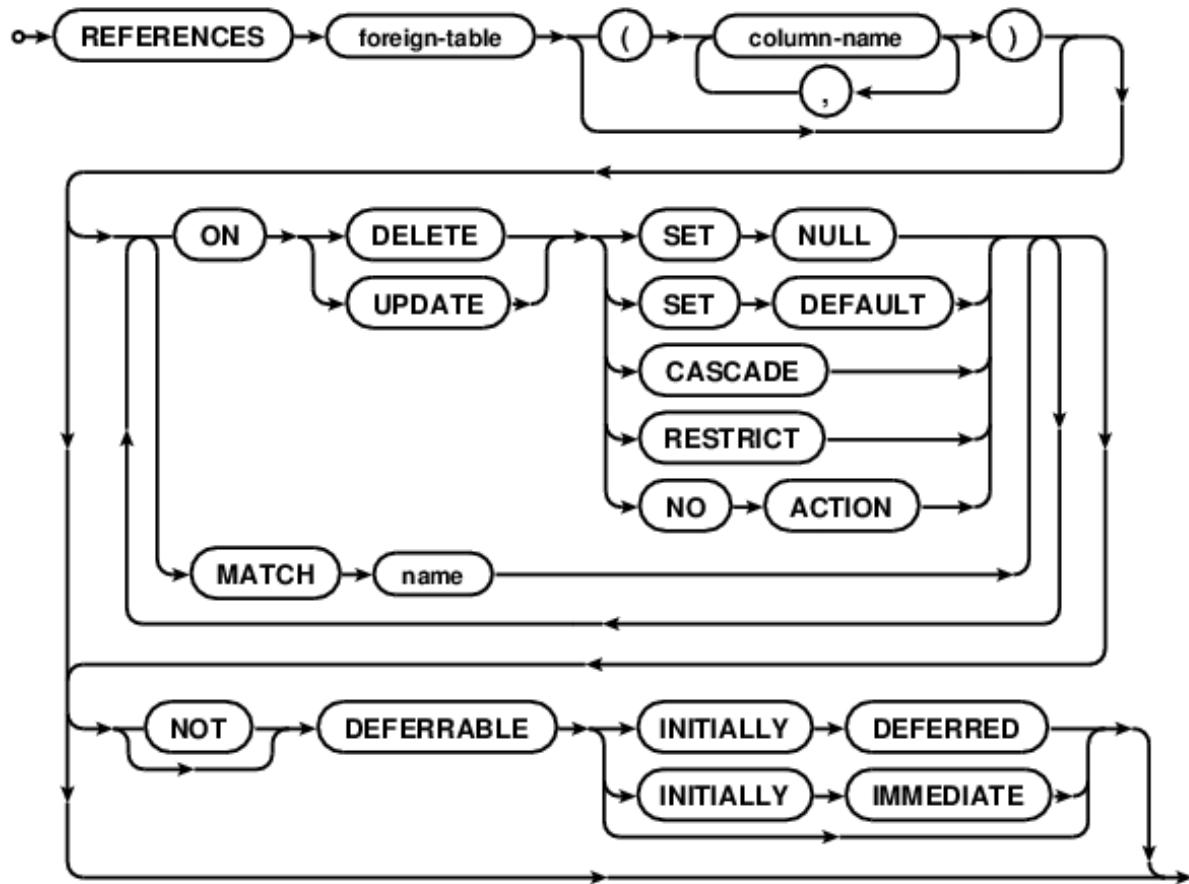
FOREIGN KEY

- ▶ FOREIGN KEY – ограничение уровня таблицы
- ▶ Нужны для обеспечения целостности данных
- ▶ Внешний ключ объявляется при помощи конструкции FOREIGN KEY
- ▶ Таблица и столбец, на которые ссылается внешний ключ указывается после ключевого слова REFERENCE



FOREIGN KEY

[foreign-key-clause:](#) hide



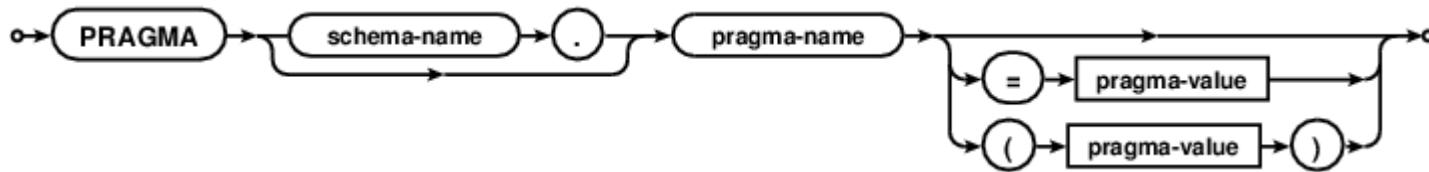
PRAGMA foreign_keys

- ▶ Внешние ключи по умолчанию отключены для обратной совместимости
- ▶ PRAGMA foreign_keys – определить, включены ли в данный момент внешние ключи



PRAGMA

- ▶ Оператор PRAGMA является расширением SQL, специфичным для SQLite
- ▶ Используется для изменения работы библиотеки SQLite
- ▶ Нет гарантии обратной совместимости для утверждений PRAGMA
- ▶ Сообщения об ошибках не генерируются, если выдается неизвестная PRAGMA

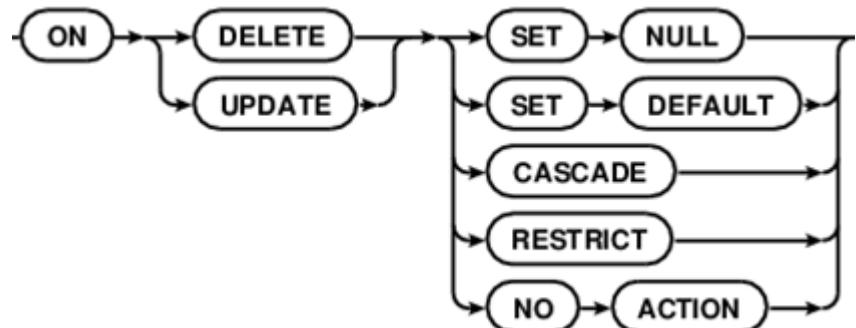


PRAGMA foreign_keys



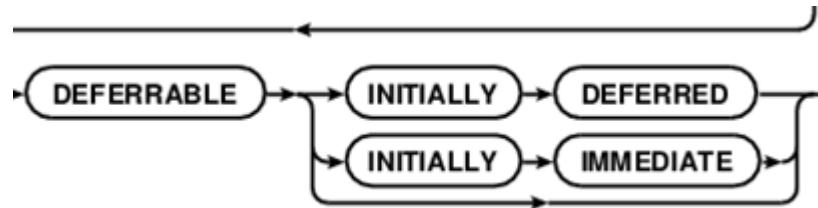
FOREIGN KEY ON DELETE/UPDATE

- ▶ RESTRICT – явное указание на запрет изменять или удалять PRIMARY KEY значение, если на него ссылаются FOREIGN KEY – действует по умолчанию
- ▶ NO ACTION – явное указание на то, что никакие не предпринимать при изменении или удалении PRIMARY KEY значения (фактически отсутствует ограничение)
- ▶ SET NULL
- ▶ SET DEFAULT
- ▶ CASCADE



FOREIGN KEY DEFERRABLE

- ▶ DEFERRABLE – отложенная проверка ограничения
 - ▶ DEFERRED – проверка осуществляется при COMMIT
 - ▶ IMMEDIATE – проверка осуществляется сразу



FOREIGN KEY MATCH

- ▶ Способ сравнения FOREIGN KEY и PRIMARY KEY составных ключей
 - ▶ MATCH SIMPLE – если одно из значений составного ключа NULL, то проверка на ограничение не проверяется
 - ▶ MATCH FULL – допускаются (проверка не осуществляется) составные ключи состоящие полностью из NULL
 - ▶ MATCH PARTIAL – если в составном FOREIGN KEY есть NULL-значение, то должен существовать хотя бы один родительский ключ у которого есть значение для NULL-позиции FOREIGN KEY – ключа



Внутренние таблицы

- ▶ SQLITE_MASTER
- ▶ SQLITE_SEQUENCE
- ▶ Таблицы SQLITE_STAT



Вопросы?



Базы данных

Лекция 13



SQLite



SELECT

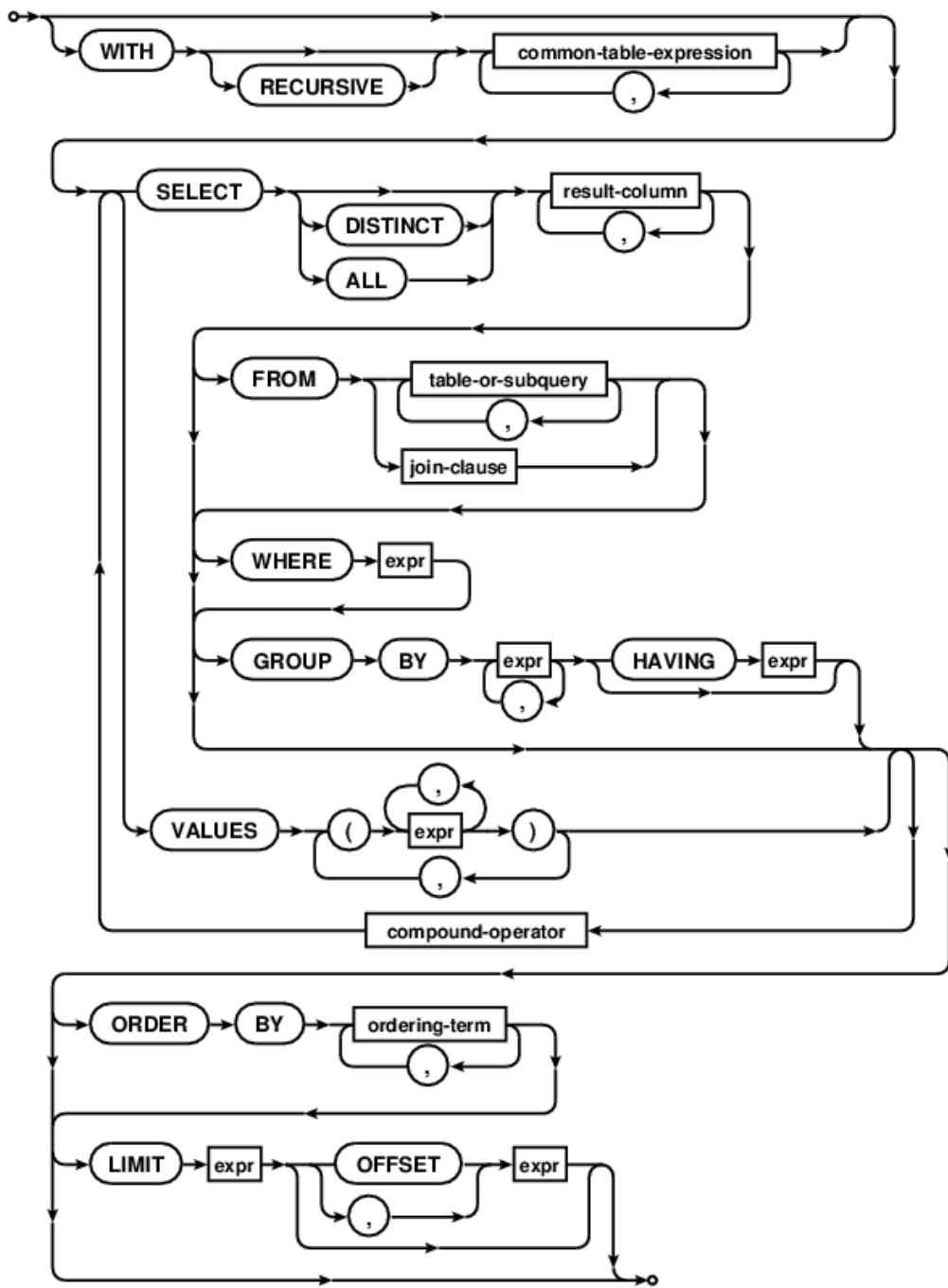
Рассматриваемые темы

- ▶ Выборка с различными условиями
- ▶ Использование сортировки
- ▶ Использование группировки
- ▶ Операторы и функции для работы со строками и датами
- ▶ Подзапросы
- ▶ Соединение таблиц
- ▶ Объединение выборок
- ▶ Common Table Expressions – CTE



Выборка с различными условиями

select-stmt: [hide](#)



Использование сортировки

Сортирующие последовательности

- ▶ **BINARY** — сопоставляет строки побайтово независимо от кодировки
- ▶ **NOCASE** — так же, как BINARY, за исключением 26-ти прописных букв ASCII, которые перед сортировкой переводятся в свои эквиваленты в нижнем регистре
- ▶ **RTRIM** — так же, как BINARY, но игнорируются пробелы в конце строки



Сортирующие последовательности

Tables

```
CREATE TABLE `Table_B` ( `Field1` TEXT NOT NULL collate nocase, PRIMARY KEY(`Field1`) )
CREATE TABLE `Table_R` ( `Field1` TEXT NOT NULL collate rtrim, PRIMARY KEY(`Field1`) )
```

Table: Table_B

| Field1 |
|--------|
| Фильтр |
| 1 Data |
| 2 Дата |
| 3 дата |
| 4 data |

DB Browser for SQLite

Ошибка изменения данных:
UNIQUE constraint failed: Table_B.Field1

OK

Table: Table_R

| Field1 |
|--------|
| Фильтр |
| 1 Data |
| 2 data |
| 3 Data |

DB Browser for SQLite

Ошибка изменения данных:
UNIQUE constraint failed: Table_R.Field1

OK

Операторы и функции для работы со строками и датами

Операторы сравнения

- ▶ **LIKE**
- ▶ **GLOB**
- ▶ **REGEXP**



LIKE

- ▶ Оператор LIKE – сравнение текста с образцом
- ▶ По умолчанию ‘а’ LIKE ‘А’ = TRUE для ASCII
- ▶ PRAGMA case_sensitive_like
- ▶ % - ноль или больше символов
- ▶ _ - точно один символ

```
1 select rowid, empl_num, name from SALESREPS where name like '%smith';
```

| | rowid | EMPL_NUM | NAME |
|---|-------|----------|-----------|
| 1 | 3 | 104 | Bob Smith |
| 2 | 6 | 102 | Sue Smith |

GLOB

- ▶ Оператор GLOB – побайтовое сравнение текста с образцом
- ▶ Все не NULL конвертируются в текст
- ▶ По умолчанию ‘а’ GLOB ‘A’ = FALSE
- ▶ * - ноль или больше символов, ? - точно один символ
- ▶ [...] – набор значений, например, [a-zA-Z0-9], ^ – исключение из набора, например, [^0-9] исключает все цифры



GLOB

SQL 1 

```
1 select rowid, empl_num, name from SALESREPS where name glob '*smith';
```

<

0 строки возвращены за 0мс из: select rowid, empl_num, name from SALESREPS where name glob '*smith';

```
1 select rowid, empl_num, name from SALESREPS where name glob '*Smith';
```

<

| | rowid | EMPL_NUM | NAME |
|---|-------|----------|-----------|
| 1 | 3 | 104 | Bob Smith |
| 2 | 6 | 102 | Sue Smith |

2 строки возвращены за 0мс из: select rowid, empl_num, name from SALESREPS where name glob '*Smith';

```
1 select rowid, empl_num, name from SALESREPS where name glob '[S-Z]*';
```

<

| | rowid | EMPL_NUM | NAME |
|---|-------|----------|------------|
| 1 | 1 | 106 | Sam Clark |
| 2 | 6 | 102 | Sue Smith |
| 3 | 8 | 110 | Tom Snyder |

Работа с датами тип хранения

- ▶ TEXT в виде строки формата: «YYYY-MM-DD HH:MM:SS.SSS»
- ▶ REAL как числа юлианского календаря - число дней с полуночью 24 ноября 4714 г. до н.э.
- ▶ INTEGER как время Unix - количество секунд с 1970-01-01 00:00:00 UTC (POSIX)

```
1 select order_num, order_date, amount from ORDERS where order_date > '2007-12-12';
```

| | ORDER_NUM | ORDER_DATE | AMOUNT |
|---|-----------|------------|--------|
| 1 | 112961 | 2007-12-17 | 31500 |
| 2 | 112963 | 2007-12-17 | 3276 |
| 3 | 113012 | 2008-01-11 | 3745 |
| 4 | 113013 | 2008-01-14 | 652 |
| 5 | 113036 | 2008-01-30 | 22500 |
| 6 | 113045 | 2008-02-02 | 45000 |
| 7 | 113055 | 2008-02-15 | 150 |

Работа с датами – вопросы

- ▶ Временные зоны
- ▶ Смещение на летнее/зимнее время
- ▶ Сравнение дат
- ▶ Выборки дат по условию
- ▶ Смещение даты и времени на определенный период



Работа с датами

- ▶ Временные зоны не поддерживаются
- ▶ Смещение на летнее/зимнее время не поддерживается
- ▶ Поддерживается формат UTC – Universal Coordinated Time – практически идентичен времени по Гринвичу



Работа с датами – литералы времени

- ▶ CURRENT_TIME – HH:MM:SS
- ▶ CURRENT_DATE – YYYY-MM-DD
- ▶ CURRENT_TIMESTAMP – YYYY-MM-DD HH:MM:SS

```
1 select current_date, current_time, current_timestamp;
```

| | current_date | current_time | current_timestamp |
|---|--------------|--------------|---------------------|
| 1 | 2023-04-12 | 11:10:44 | 2023-04-12 11:10:44 |

Работа с датами – функции

- ▶ `date(timestring, modifier, modifier...)`
- ▶ `time(timestring, modifier, modifier...)`
- ▶ `datetime(timestring, modifier, modifier...)`



Работа с датами – функции

```
1 select date(order_date), time(order_date) from ORDERS;
```

| | date(order_date) | time(order_date) |
|---|------------------|------------------|
| 1 | 2007-12-17 | 00:00:00 |
| 2 | 2007-12-17 | 00:00:00 |
| 3 | 2007-10-12 | 00:00:00 |
| 4 | 2007-10-12 | 00:00:00 |
| 5 | 2007-11-04 | 00:00:00 |
| 6 | 2008-01-11 | 00:00:00 |
| 7 | 2008-01-14 | 00:00:00 |
| 8 | 2008-01-30 | 00:00:00 |

```
1 select date(current_time), date(current_date), date(current_timestamp), time(current_timestamp);
```

| | date(current_time) | date(current_date) | date(current_timestamp) | time(current_timestamp) |
|---|--------------------|--------------------|-------------------------|-------------------------|
| 1 | 2000-01-01 | 2023-04-12 | 2023-04-12 | 11:13:55 |



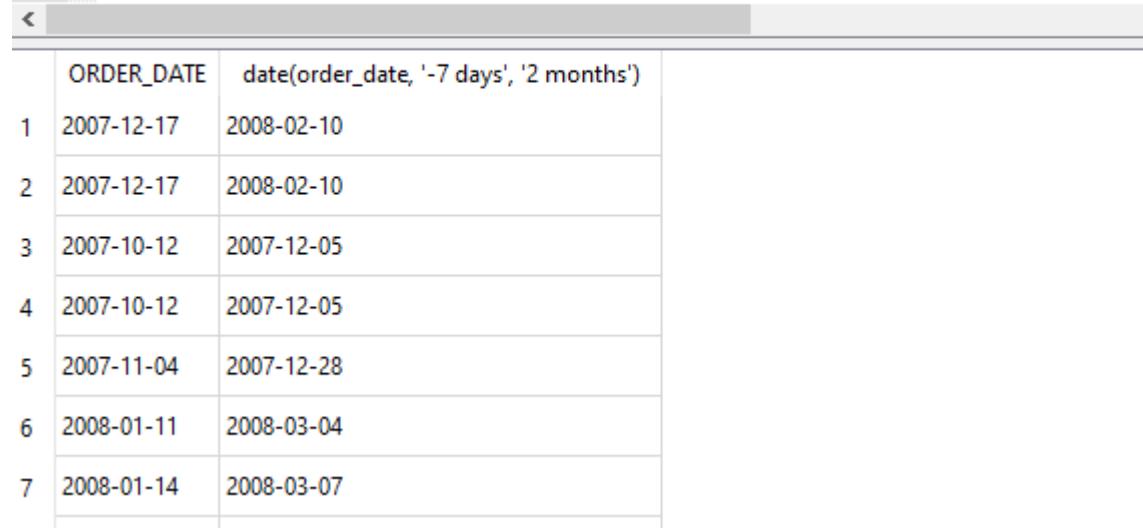
Работа с датами – функции

- ▶ [+/-] NNN day[s]
- ▶ [+/-] NNN hour[s]
- ▶ [+/-] NNN minute[s]
- ▶ [+/-] NNN second[s]
- ▶ [+/-] NNN.nnn second[s]
- ▶ [+/-] NNN month[s]
- ▶ [+/-] NNN year[s]



Работа с датами – функции

```
1 select order_date, date(order_date, '-7 days', '2 months') from ORDERS;
```



| | ORDER_DATE | date(order_date, '-7 days', '2 months') |
|---|------------|-----------------------------------------|
| 1 | 2007-12-17 | 2008-02-10 |
| 2 | 2007-12-17 | 2008-02-10 |
| 3 | 2007-10-12 | 2007-12-05 |
| 4 | 2007-10-12 | 2007-12-05 |
| 5 | 2007-11-04 | 2007-12-28 |
| 6 | 2008-01-11 | 2008-03-04 |
| 7 | 2008-01-14 | 2008-03-07 |

Работа с датами – функции

- ▶ start of month
- ▶ start of year
- ▶ start of day
- ▶ weekday N
(начиная с воскресенья 0-6)

```
1 select order_date,
2       date(order_date, 'start of month'),
3       date(order_date, 'start of year')
4   from ORDERS;
```

| | ORDER_DATE | date(order_date, 'start of month') | date(order_date, 'start of year') |
|---|------------|------------------------------------|-----------------------------------|
| 1 | 2007-12-17 | 2007-12-01 | 2007-01-01 |
| 2 | 2007-12-17 | 2007-12-01 | 2007-01-01 |
| 3 | 2007-10-12 | 2007-10-01 | 2007-01-01 |
| 4 | 2007-10-12 | 2007-10-01 | 2007-01-01 |
| 5 | 2007-11-04 | 2007-11-01 | 2007-01-01 |
| 6 | 2008-01-11 | 2008-01-01 | 2008-01-01 |
| 7 | 2008-01-14 | 2008-01-01 | 2008-01-01 |

Работа с датами – распознаваемые форматы даты

- ▶ YYYY-MM-DD
- ▶ YYYY-MM-DD HH:MM
- ▶ YYYY-MM-DD HH:MM:SS
- ▶ YYYY-MM-DD HH:MM:SS.sss
- ▶ YYYY-MM-DDTHH:MM
- ▶ YYYY-MM-DDTHH:MM:SS
- ▶ YYYY-MM-DDTHH:MM:SS.sss
- ▶ HH:MM
- ▶ HH:MM:SS
- ▶ HH:MM:SS.sss
- ▶ now
- ▶ DDDDDDDD
- ▶ DDDDDDDD.ddddddd



Работа с датами – функция strftime()

- ▶ `strftime(format, time, modifier, modifier...)`
 - ▶ %d — day of the month (*DD*), 01-31
 - ▶ %f — seconds with fractional part (*SS.sss*), 00-59 plus decimal portion
 - ▶ %H — hour (*HH*), 00-23
 - ▶ %j — day of the year (*NNN*), 001-366
 - ▶ %J — Julian day number (*DDDDDDDD.dddddddd*)
 - ▶ %m — month (*MM*), 01-12
 - ▶ %M — minute (*MM*), 00-59
 - ▶ %s — seconds since 1970-01-01 (POSIX time value)
 - ▶ %S — seconds (*SS*), 00-59
 - ▶ %w — day of the week (*N*), 0-6, starting with Sunday as 0
 - ▶ %W — week of the year (*WW*), 00-53
 - ▶ %Y — year (*YYYY*)



Работа с датами – функция strftime()

```
1 select order_date,  
2      strftime('%d', order_date) as Day,  
3      strftime('%m', order_date) as Month,  
4      strftime('%Y', order_date) as Year  
5 from ORDERS;
```

| | ORDER_DATE | Day | Month | Year |
|---|------------|-----|-------|------|
| 1 | 2007-12-17 | 17 | 12 | 2007 |
| 2 | 2007-12-17 | 17 | 12 | 2007 |
| 3 | 2007-10-12 | 12 | 10 | 2007 |
| 4 | 2007-10-12 | 12 | 10 | 2007 |
| 5 | 2007-11-04 | 04 | 11 | 2007 |
| 6 | 2008-01-11 | 11 | 01 | 2008 |

```
1 select order_date,  
2      strftime('%d/%m/%Y', order_date) as Date  
3 from ORDERS;
```

| | ORDER_DATE | Date |
|---|------------|------------|
| 1 | 2007-12-17 | 17/12/2007 |
| 2 | 2007-12-17 | 17/12/2007 |
| 3 | 2007-10-12 | 12/10/2007 |
| 4 | 2007-10-12 | 12/10/2007 |
| 5 | 2007-11-04 | 04/11/2007 |
| 6 | 2008-01-11 | 11/01/2008 |

Подзапросы

```
1 select order_num, order_date, amount
2 from ORDERS
3 where cust = ( SELECT cust_num from customers where company = 'Holm Landis');|
```

A screenshot of a database query results window. At the top, there is a toolbar with a back arrow icon. Below the toolbar is a header row with three columns: ORDER_NUM, ORDER_DATE, and AMOUNT. A single data row is shown below the header, consisting of three cells with the values 1, 113055, and 2008-02-15 respectively.

| | ORDER_NUM | ORDER_DATE | AMOUNT |
|---|-----------|------------|--------|
| 1 | 113055 | 2008-02-15 | 150 |



JOINS

- ▶ INNER JOIN
- ▶ CROSS JOIN
- ▶ JOIN ... USING
- ▶ NATURAL JOIN
- ▶ Self JOIN
- ▶ OUTER JOIN



Запись соединений

- ▶ Поддерживаются два стиля записи соединения:
- ▶ ANSI:
- ▶ `SELECT ... FROM TABLE_1 JOIN TABLE_2`
- ▶ `ON (TABLE_1.ID = TABLE_2.ID)`

- ▶ `SELECT ... FROM TABLE_1, TABLE_2`
- ▶ `WHERE TABLE_1.ID = TABLE_2.ID`



INNER JOIN

- ▶ JOIN по умолчанию трактуется как INNER JOIN
- ▶ Если нет условия (ON), то трактуется как CROSS JOIN

```
1 select order_num, order_date, amount, company
2   from ORDERS o join customers c
3     on o.cust = c.cust_num
4   where credit_limit = 55000;
```

| | ORDER_NUM | ORDER_DATE | AMOUNT | COMPANY |
|---|-----------|------------|--------|-------------|
| 1 | 113055 | 2008-02-15 | 150 | Holm Landis |

JOIN ... USING

```
create table t1 (x int, y int);
create table t2 (x int, y int);
insert into t1 values (1,1);
insert into t1 values (1,1);
insert into t1 values (2,2);
insert into t1 values (3,3);

insert into t2 values (1,1);
insert into t2 values (1,2);
insert into t2 values (2,2);
insert into t2 values (4,4);
```

```
1 select t1.x, t2.x, t1.y, t2.y from t1 join t2 using (x);
```

| | x | y |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 2 | 2 |
| 4 | 3 | 3 |

| | x | y |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 2 |
| 3 | 2 | 2 |
| 4 | 4 | 4 |

```
1 select t1.x, t2.x, t1.y, t2.y from t1 join t2 using (y);
```

| | x | x | y | y |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 2 |
| 3 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 2 |
| 5 | 2 | 2 | 2 | 2 |

```
1 select t1.x, t2.x, t1.y, t2.y from t1 join t2 using (y);
```

| | x | x | y | y |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 1 |
| 3 | 2 | 1 | 2 | 2 |
| 4 | 2 | 2 | 2 | 2 |



NATURAL JOIN

```
1 select t1.x, t2.x, t1.y, t2.y from t1 natural join t2;
```

The screenshot shows a database interface with a query window containing the SQL command for a natural join. Below the query window is a results grid. The results grid has four columns labeled 'x', 'x', 'y', and 'y'. It contains three rows of data, each with values 1, 1, 1, 1. A horizontal scrollbar is visible above the results grid.

| | x | x | y | y |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 1 |
| 3 | 2 | 2 | 2 | 2 |



Self JOIN

```
1 select t11.x, t12.x, t11.y, t12.y  
2 from t2 as t11 join t2 as t12  
3 on t11.x = t12.y;
```

<

| | x | x | y | y |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 2 | 1 |
| 3 | 2 | 1 | 2 | 2 |
| 4 | 2 | 2 | 2 | 2 |
| 5 | 4 | 4 | 4 | 4 |



OUTER JOIN

- ▶ Поддерживается только LEFT OUTER JOIN
- ▶ Поддерживаются все конструкции USING, NATURAL, ON
- ▶ Как получить FULL OUTER JOIN?



OUTER JOIN

```
1 select t1.x, t1.y, t2.x, t2.y  
2 from t1 left join t2  
3 on t1.x = t2.y;
```

< [REDACTED]

| | x | y | x | y |
|---|---|---|------|------|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 1 |
| 3 | 2 | 2 | 1 | 2 |
| 4 | 2 | 2 | 2 | 2 |
| 5 | 3 | 3 | NULL | NULL |

```
1 select t1.x, t1.y, t2.x, t2.y  
2 from t2 left join t1  
3 on t1.x = t2.y;
```

< [REDACTED]

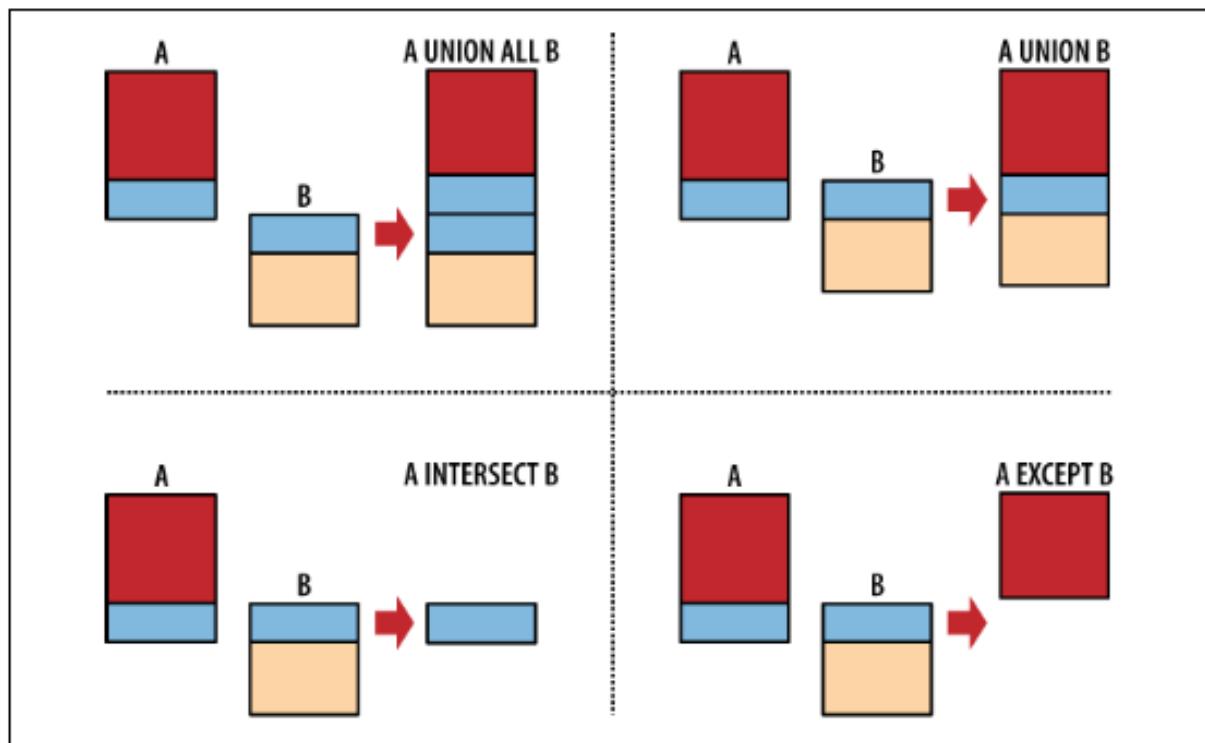
| | x | y | x | y |
|---|------|------|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 1 |
| 3 | 2 | 2 | 1 | 2 |
| 4 | 2 | 2 | 2 | 2 |
| 5 | NULL | NULL | 4 | 4 |



Объединение выборок

Объединения

- ▶ UNION ALL
- ▶ UNION
- ▶ INTERSECT
- ▶ EXCEPT



UNION

```
1 select t1.x, t1.y  
2 from t1  
3 union  
4 select t2.x, t2.y|  
5 from t2;
```

| | x | y |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 2 |
| 3 | 2 | 2 |
| 4 | 3 | 3 |
| 5 | 4 | 4 |

```
1 select t1.x, t1.y  
2 from t1  
3 union all  
4 select t2.x, t2.y|  
5 from t2;
```

| | x | y |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 2 | 2 |
| 4 | 3 | 3 |
| 5 | 1 | 1 |
| 6 | 1 | 2 |
| 7 | 2 | 2 |
| 8 | 4 | 4 |

```
1 select t1.x, t1.y  
2 from t1  
3 intersect  
4 select t2.x, t2.y|  
5 from t2;
```

| | x | y |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 2 |

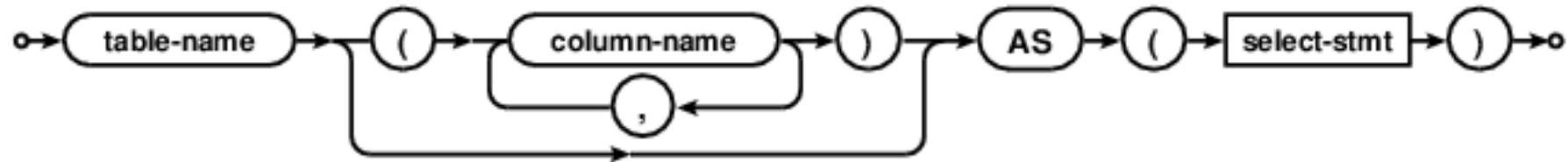
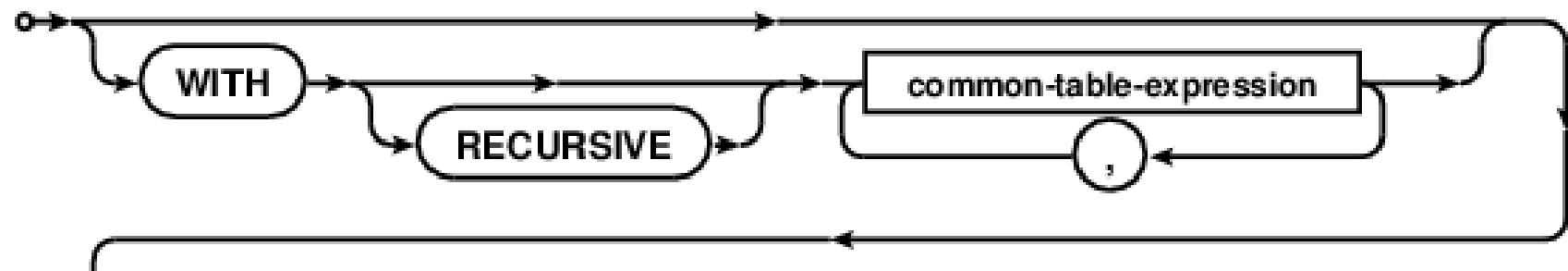
```
1 select t1.x, t1.y  
2 from t1  
3 except  
4 select t2.x, t2.y|  
5 from t2;
```

| | x | y |
|---|---|---|
| 1 | 3 | 3 |



Common Table Expressions – CTE

Common Table Expressions



Common Table Expressions

- ▶ Обобщенные табличные выражения – временные результирующие наборы, определенные в области выполнения единичных инструкций SELECT
- ▶ Не сохраняются в базе данных в виде объектов
- ▶ Время жизни ограничено продолжительностью запроса
- ▶ Могут ссылаться сами на себя
- ▶ Один и тот же запрос может ссылаться на СТЕ несколько раз



Common Table Expressions

- ▶ WITH expression_name [(column_name [,...n])]
 - ▶ AS (CTE_query_definition)
-
- ▶ SELECT <column_list> FROM expression_name;



Common Table Expressions

```
1  with avg_cte as
2      (select avg(amount) as avg_amount from ORDERS)
3  select *
4    from ORDERS as o cross join avg_cte
5   where o.amount > avg_amount;
```

| | ORDER_NUM | ORDER_DATE | CUST | REP | MFR | PRODUCT | QTY | AMOUNT | avg_amount |
|---|-----------|------------|------|-----|-----|---------|-----|--------|------------|
| 1 | 112961 | 2007-12-17 | 2117 | 106 | REI | 2A44L | 7 | 31500 | 11366.1 |
| 2 | 113036 | 2008-01-30 | 2107 | 110 | ACI | 4100Z | 9 | 22500 | 11366.1 |
| 3 | 113045 | 2008-02-02 | 2112 | 108 | REI | 2A44R | 10 | 45000 | 11366.1 |

Common Table Expressions

- ▶ Рекурсивность – обращение через СТЕ к самому себе

```
1  with inf_rec_cte as
2    (select 1
3     union all
4     select * from inf_rec_cte)
5   select * from inf_rec_cte;
```

Common Table Expressions

```
1   with inf_rec_cte as
2     (select 1 as rec_data
3      union all
4      select rec_data + 1 from inf_rec_cte limit 5)
5      select * from inf_rec_cte;
```

| | rec_data |
|---|----------|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |



Common Table Expressions

- ▶ Создания рекурсивных запросов
- ▶ Многократных ссылок на результирующую таблицу из одной и той же инструкции
- ▶ Замены представлений
- ▶ Группирования по столбцу, производного от скалярного подзапроса выборки или функции, которая недетерминирована



Common Table Expressions

- ▶ Позволяет значительно повысить читаемость и упростить работу со сложными запросами
 - ▶ Можно составлять промежуточные СТЕ
 - ▶ Могут быть определены в пользовательских подпрограммах
-
- ▶ Повышает ли СТЕ производительность запроса?



SQLITE VIRTUAL TABLE

- ▶ Виртуальная таблица – механизм, позволяющий создавать таблицу-модуль, зарегистрированный в SQLite



SQLITE VIRTUAL TABLE

- ▶ Создается программный модуль, для которого определяется необходимый набор методов
- ▶ Модуль регистрируется в SQLite
- ▶ Может быть создана виртуальная таблица, основанная на таком объекте
- ▶ С этой таблицей можно работать, как с любой другой
- ▶ Модули могут быть внешние и внутренние

iVersion
xCreate()
xConnect()
xBestIndex()
xDisconnect()
xDestroy()
xOpen()
xClose()
xFilter()
xNext()
xEof()
xColumn()
xRowid()
xUpdate()
xBegin()
xSync()
xCommit()
xRollback()
xFindFunction()
xRename()

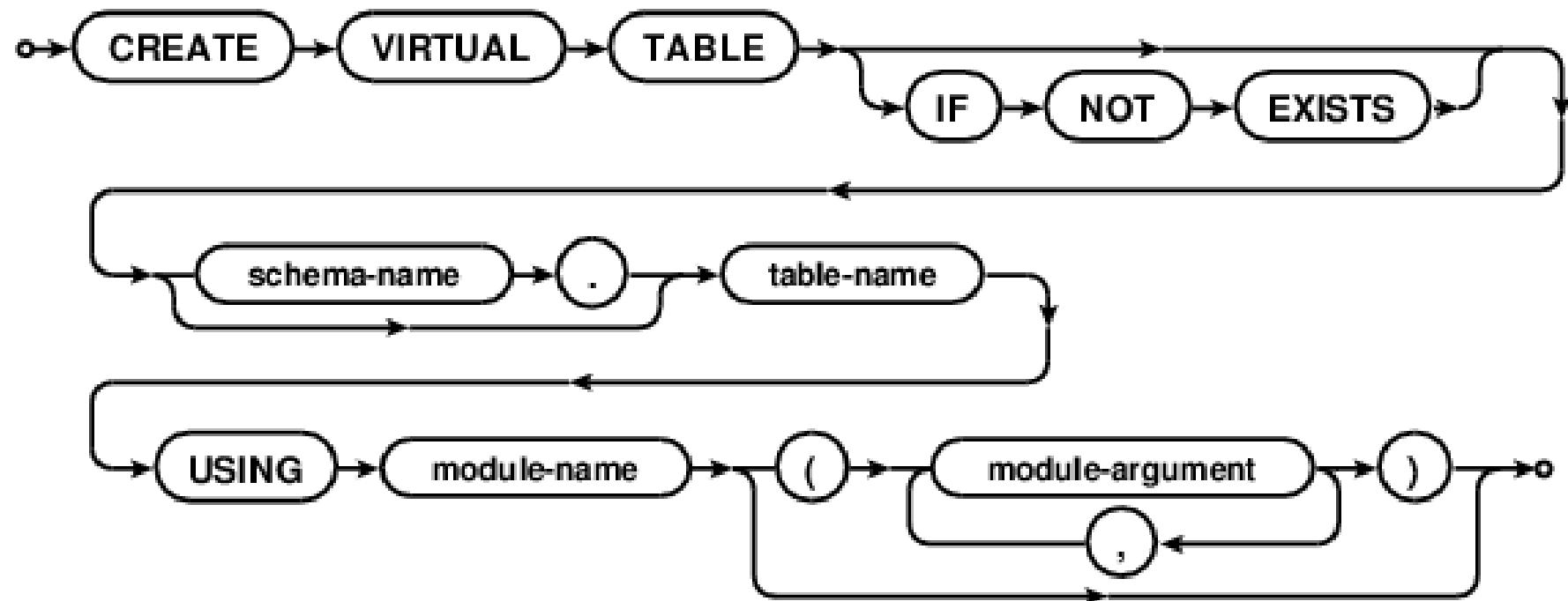


SQLITE VIRTUAL TABLE

- ▶ Для SQLITE существуют готовые модули Full-Text Search Module FTS3, FTS4, предназначенные для полнотекстового поиска документов



SQLITE VIRTUAL TABLE



SQLITE VIRTUAL TABLE

- ▶ Нельзя создать триггер на виртуальной таблице
- ▶ Нельзя создавать дополнительные индексы на виртуальной таблице, используя инструкции CREATE INDEX
- ▶ Виртуальные таблицы могут иметь индексы, но они должны быть встроены в реализацию виртуальной таблицы
- ▶ Нельзя выполнить ALTER TABLE ... ADD COLUMN



SQLITE VIRTUAL TABLE USING fts4()

```
1 CREATE TABLE search_table(
2     ID INTEGER PRIMARY KEY ,
3     NAME text,
4     VALUE text
5 );
6     INSERT INTO search_table (ID,NAME,VALUE)
7     VALUES (1,"Hello","World"),
8     (2,"Hello","my World"),
9     (3,"Hello","mmy World"),
10    (4,"Hello","my World myWorld"),
11    (5,"Hello","Some words in this sentences some strange to me");
12
13    CREATE VIRTUAL TABLE search_table_fts USING fts4(ID,NAME,VALUE);
14
15    INSERT INTO search_table_fts(ID,NAME,VALUE)
16    SELECT ID,NAME,VALUE FROM search_table;
17
18    SELECT ID,NAME,VALUE FROM search_table;
19    SELECT ID,NAME,VALUE FROM search_table_fts;
```

| | ID | NAME | VALUE |
|---|----|-------|-------------------------------------------------|
| 1 | 1 | Hello | World |
| 2 | 2 | Hello | my World |
| 3 | 3 | Hello | mmy World |
| 4 | 4 | Hello | my World myWorld |
| 5 | 5 | Hello | Some words in this sentences some strange to me |

SQLITE VIRTUAL TABLE USING fts()

```
21 SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'myWorld';
22 SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'my*';
23
24 SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'my NEAR some ';
25 SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'this NEAR some ';
```

| ID | VALUE |
|----|--------------------|
| 1 | 4 my World myWorld |

```
20
21 SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'myWorld';
22 SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'my*';
23
24 SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'my NEAR some ';
25 SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'this NEAR some ';
```

| ID | VALUE |
|----|--------------------|
| 1 | 2 my World |
| 2 | 4 my World myWorld |



SQLITE VIRTUAL TABLE USING fts4()

```
23
24     SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'my NEAR some ';
25     SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'this NEAR some ';
26
```

0 строки возвращены за 0мс из: SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'my NEAR some ';

```
24     SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'my NEAR some ';
25     SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'this NEAR some ';
26
```

| | ID | VALUE |
|---|----|-------------------------------------------------|
| 1 | 5 | Some words in this sentences some strange to me |

1 строки возвращены за 0мс из: SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'this NEAR some ';

SQLITE VIRTUAL TABLE USING fts4()

```
26
27   SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'words OR some';
28
<


| ID | VALUE                                             |
|----|---------------------------------------------------|
| 1  | 5 Some words in this sentences some strange to me |


1 строки возвращены за 0мс из: SELECT ID,VALUE FROM search_table_fts WHERE VALUE MATCH 'words OR some';
```

Применение виртуальных таблиц

- ▶ Полнотекстовой поиск
- ▶ Пространственные индексы
- ▶ Просмотр статистики в файле базы данных
- ▶ Чтение и / или запись содержимого файла с разделителями-запятыми (CSV)
- ▶ Доступ к файловой системе компьютера, как если бы это была таблица базы данных



Внутренние таблицы SQLite

- ▶ **SQLITE_MASTER**
- ▶ **SQLITE_SEQUENCE**
- ▶ **SQLITE_STATN**



SQLITE_MASTER

| Столбец | Тип хранения | Значения |
|----------|--------------|-------------------------------------------|
| type | text | Тип объекта – table, ... |
| name | text | Имя объекта |
| tbl_name | text | Имя таблицы или представления для объекта |
| rootpage | integer | Номер корневой страницы |
| sql | text | Порождающий SQL запрос |

SQLITE_MASTER

```
1 select * from sqlite_master;
```

| | type | name | tbl_name | rootpage | sql |
|---|-------|------------------------------|-------------------|----------|----------------------------------------------|
| 1 | table | PRODUCTS | PRODUCTS | 2 | CREATE TABLE PRODUCTS (MFR_ID CH... |
| 2 | index | sqlite_autoindex_PRODUCTS_1 | PRODUCTS | 3 | NULL |
| 3 | table | OFFICES | OFFICES | 4 | CREATE TABLE OFFICES (OFFICE INT NO... |
| 4 | index | sqlite_autoindex_OFFICES_1 | OFFICES | 5 | NULL |
| 5 | table | SALESREPS | SALESREPS | 6 | CREATE TABLE SALESREPS (EMPL_NUM I... |
| 6 | index | sqlite_autoindex_SALESREPS_1 | SALESREPS | 7 | NULL |
| 7 | table | CUSTOMERS | CUSTOMERS | 8 | CREATE TABLE CUSTOMERS (CUST_NUM ... |
| 8 | table | ORDERS | ORDERS | 9 | CREATE TABLE ORDERS (ORDER_NUM IN... |
| 9 | view | v_order_less_5000 | v_order_less_5000 | 0 | CREATE VIEW v_order_less_5000 asselect * ... |



DESCRIBE TABLE

```
1 pragma table_info('PRODUCTS');
```

| | cid | name | type | notnull | dflt_value | pk |
|---|-----|-------------|-------------|---------|------------|----|
| 1 | 0 | MFR_ID | CHAR(3) | 1 | NULL | 1 |
| 2 | 1 | PRODUCT_ID | CHAR(5) | 1 | NULL | 2 |
| 3 | 2 | DESCRIPTION | VARCHAR(20) | 1 | NULL | 0 |
| 4 | 3 | PRICE | MONEY | 1 | NULL | 0 |
| 5 | 4 | QTY_ON_HAND | INTEGER | 1 | NULL | 0 |

Вопросы?



Базы данных

Лекция 14

Объекты SQLite

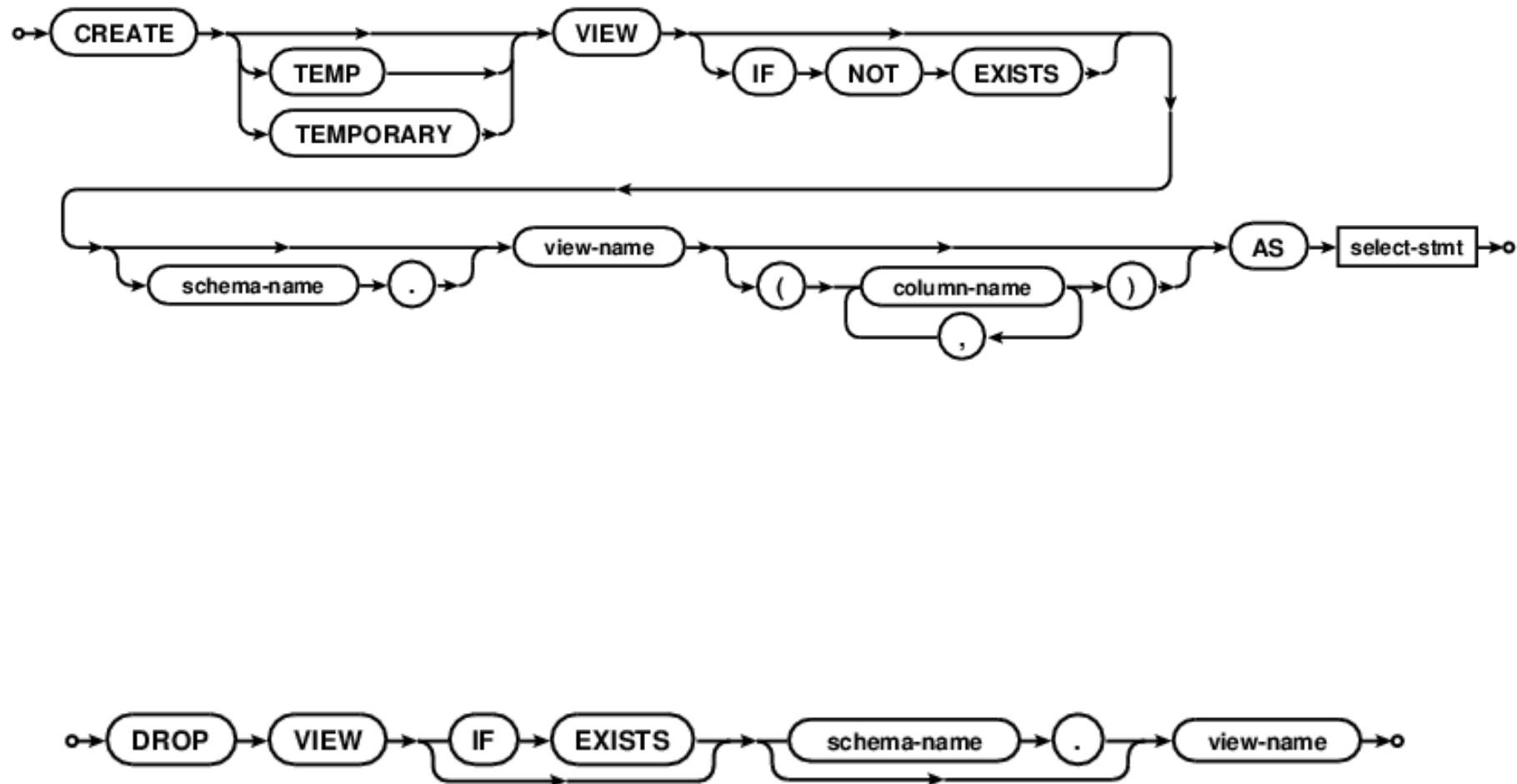


Представление

- ▶ Представление – VIEW – поименованный SELECT-запрос
- ▶ каким образом создается?
- ▶ в каких случаях применяется?
- ▶ повышает ли представление производительность запроса?
- ▶ разрешается ли выполнение операций INSERT, DELETE, UPDATE?
- ▶ допускается ли применение опции CHECK OPTION?



CREATE VIEW



CREATE TEMP VIEW

- ▶ CREATE TEMP /TEMPORARY/ VIEW – создание временного представления
- ▶ Существует в рамках одного соединения
- ▶ Аналогично и для таблиц



CREATE VIEW

```
1  create view v_order_less_5000 as
2      select * from orders
3      where amount < 5000;
4
5      select * from v_order_less_5000;
6      |
```

| | ORDER_NUM | ORDER_DATE | CUST | REP | MFR | PRODUCT | QTY | AMOUNT |
|---|-----------|------------|------|-----|-----|---------|-----|--------|
| 1 | 112963 | 2007-12-17 | 2103 | 105 | ACI | 41004 | 28 | 3276 |
| 2 | 112968 | 2007-10-12 | 2102 | 101 | ACI | 41004 | 34 | 3978 |
| 3 | 112975 | 2007-10-12 | 2111 | 103 | REI | 2A44G | 6 | 2100 |
| 4 | 112992 | 2007-11-04 | 2118 | 108 | ACI | 41002 | 10 | 760 |
| 5 | 113012 | 2008-01-11 | 2111 | 105 | ACI | 41003 | 35 | 3745 |
| 6 | 113013 | 2008-01-14 | 2118 | 108 | BIC | 41003 | 1 | 652 |
| 7 | 113055 | 2008-02-15 | 2108 | 101 | ACI | 4100X | 6 | 150 |

CREATE TEMP VIEW

```
1 create temp view tv_order_less_5000 as
2   select * from orders
3   where amount < 5000;
4
5   select * from tv_order_less_5000;
```

| | ORDER_NUM | ORDER_DATE | CUST | REP | MFR | PRODUCT | QTY | AMOUNT |
|---|-----------|------------|------|-----|-----|---------|-----|--------|
| 1 | 112963 | 2007-12-17 | 2103 | 105 | ACI | 41004 | 28 | 3276 |
| 2 | 112968 | 2007-10-12 | 2102 | 101 | ACI | 41004 | 34 | 3978 |
| 3 | 112975 | 2007-10-12 | 2111 | 103 | REI | 2A44G | 6 | 2100 |
| 4 | 112992 | 2007-11-04 | 2118 | 108 | ACI | 41002 | 10 | 760 |
| 5 | 113012 | 2008-01-11 | 2111 | 105 | ACI | 41003 | 35 | 3745 |
| 6 | 113013 | 2008-01-14 | 2118 | 108 | BIC | 41003 | 1 | 652 |
| 7 | 113055 | 2008-02-15 | 2108 | 101 | ACI | 4100X | 6 | 150 |

Имя

- Таблицы (5)
 - CUSTOMERS
 - OFFICES
 - ORDERS
 - PRODUCTS
 - SALESREPS
- Индексы (0)
- Представления (2)
 - tv_order_less_5000
 - v_order_less_5000
- Триггеры (0)

Имя

- Таблицы (5)
 - CUSTOMERS
 - OFFICES
 - ORDERS
 - PRODUCTS
 - SALESREPS
- Индексы (0)
- Представления (1)
 - v_order_less_5000
- Триггеры (0)

Индекс

- ▶ Что такое индекс?
- ▶ Каким образом применяется?
- ▶ В каких случаях создается автоматически?



Индекс

- ▶ Что получаем при добавлении индекса?
- ▶ Чем за это платим?
- ▶ Для каких объектов можно создать индекс?
- ▶ Для каких столбцов создаем индекс?



Индекс

- ▶ Что такое простой индекс?
- ▶ Что такое составной индекс?
- ▶ Что такое уникальный / не уникальный индекс?
- ▶ Какой порядок значений (ASC, DESC)?
- ▶ Что такое индекс покрытия?
- ▶ Что такое фильтрующий индекс?



Индекс

- ▶ Интеграция индекса с таблицей:
 - ▶ кластеризованные
 - ▶ некластеризованные



Эффективность индекса

- ▶ Эффективность индекса оценивает оптимизатор запросов, основываясь на характеристиках запроса:
 - ▶ Селективность
 - ▶ Плотность
 - ▶ Распределение значений



Селективность

- ▶ Селективность – отношение числа выбираемых записей к общему числу записей
- ▶ Чем выше селективность – чем больше записей выбирается – тем хуже



Плотность

- ▶ Плотность – отношение числа дубликатов значений к общему числу значений
- ▶ Лучшая плотность – уникальные значения



Распределение значений

- ▶ Распределение значений – показатель, как значения ключей индекса распределены по всему интервалу возможных значений
- ▶ Больше слов на букву А, чем на букву Й



Индекс

- ▶ Структура индекса:
 - ▶ деревья (tree-index)
 - ▶ частичные
 - ▶ функциональные
 - ▶ двоичные таблицы (bitmap, Oracle)
 - ▶ пространственные индексы (spatial)
 - ▶ полнотекстовые индексы (full text)
 - ▶ XML-индексы
 - ▶ колоночные индексы (для OLAP-приложений)



Индекс

- ▶ Что такое фрагментация индекса?
- ▶ В чем заключается обслуживание индекса?

- ▶ Как принимается решение о применении индекса?
- ▶ Каковы критерии принятия решения?



Поиск и сортировка в запросах в SQLite

- ▶ Полный скан таблицы
- ▶ Поиск по RowId
- ▶ Поиск по индексу (поиск RowId, переход по RowId)
- ▶ Поиск по нескольким условиям (И)
- ▶ Поиск по нескольким условиям (ИЛИ)
- ▶ Покрывающий индекс (включены все поля поиска)



Поиск и сортировка в запросах в SQLite

▶ Полный скан таблицы

| rowid | fruit | state | price |
|-------|------------|-------|-------|
| 1 | Orange | FL | 0.85 |
| 2 | Apple | NC | 0.45 |
| 4 | Peach | SC | 0.60 |
| 5 | Grape | CA | 0.80 |
| 18 | Lemon | FL | 1.25 |
| 19 | Strawberry | NC | 2.45 |
| 23 | Orange | CA | 1.05 |

```
SELECT price FROM fruitsforsale WHERE fruit='Peach';
```

The diagram illustrates a full table scan for the query `SELECT price FROM fruitsforsale WHERE fruit='Peach';`. A red vertical arrow on the left points downwards through all rows of the table. A red circle is placed over the 'fruit' column of the fourth row (Peach). A red box highlights the 'price' value '0.60' in the same row. A red arrow points from the highlighted cell to the right.

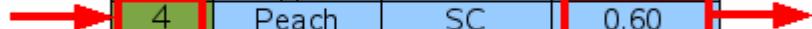
| rowid | fruit | state | price |
|-------|------------|-------|-------|
| 1 | Orange | FL | 0.85 |
| 2 | Apple | NC | 0.45 |
| 4 | Peach | SC | 0.60 |
| 5 | Grape | CA | 0.80 |
| 18 | Lemon | FL | 1.25 |
| 19 | Strawberry | NC | 2.45 |
| 23 | Orange | CA | 1.05 |

Поиск и сортировка в запросах в SQLite

▶ Поиск по RowId

```
SELECT price FROM fruitsforsale WHERE rowid=4;
```

| rowid | fruit | state | price |
|-------|------------|-------|-------|
| 1 | Orange | FL | 0.85 |
| 2 | Apple | NC | 0.45 |
| 4 | Peach | SC | 0.60 |
| 5 | Grape | CA | 0.80 |
| 18 | Lemon | FL | 1.25 |
| 19 | Strawberry | NC | 2.45 |
| 23 | Orange | CA | 1.05 |



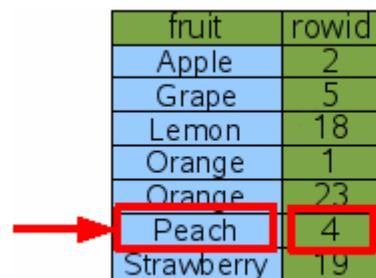
Поиск и сортировка в запросах в SQLite

▶ Поиск по индексу (поиск RowId, переход по RowId)

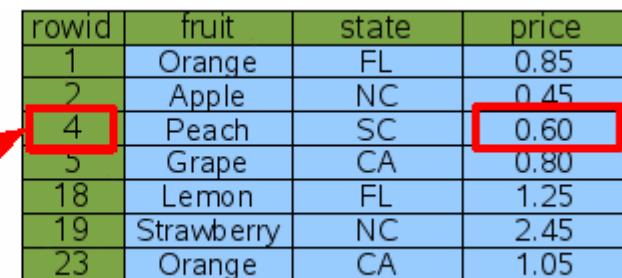
```
CREATE INDEX Idx1 ON fruitsforsale(fruit);
```

| fruit | rowid |
|------------|-------|
| Apple | 2 |
| Grape | 5 |
| Lemon | 18 |
| Orange | 1 |
| Orange | 23 |
| Peach | 4 |
| Strawberry | 19 |

```
SELECT price FROM fruitsforsale WHERE fruit='Peach';
```



| fruit | rowid |
|------------|-------|
| Apple | 2 |
| Grape | 5 |
| Lemon | 18 |
| Orange | 1 |
| Orange | 23 |
| Peach | 4 |
| Strawberry | 19 |

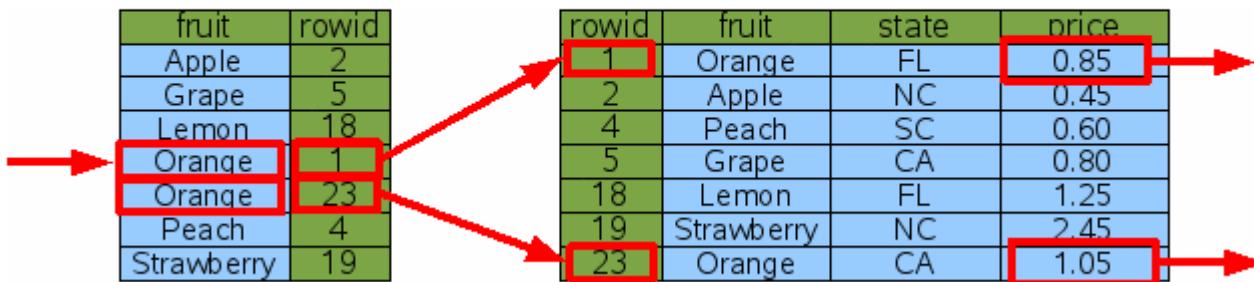


| rowid | fruit | state | price |
|-------|------------|-------|-------|
| 1 | Orange | FL | 0.85 |
| 2 | Apple | NC | 0.45 |
| 4 | Peach | SC | 0.60 |
| 5 | Grape | CA | 0.80 |
| 18 | Lemon | FL | 1.25 |
| 19 | Strawberry | NC | 2.45 |
| 23 | Orange | CA | 1.05 |

Поиск и сортировка в запросах в SQLite

- ▶ Поиск по индексу (поиск RowId, переход по RowId)

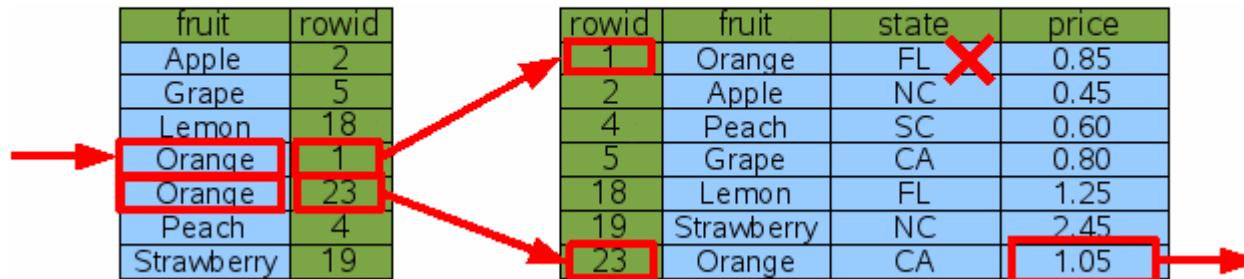
```
SELECT price FROM fruitsforsale WHERE fruit='Orange'
```



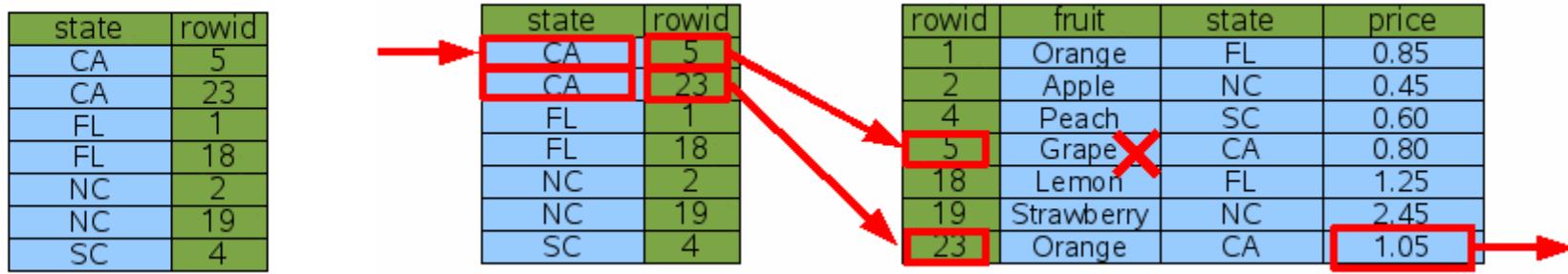
Поиск и сортировка в запросах в SQLite

▶ Поиск по нескольким условиям (И)

```
SELECT price FROM fruitsforsale WHERE fruit='Orange' AND state='CA'
```



```
CREATE INDEX Idx2 ON fruitsforsale(state);
```



Поиск и сортировка в запросах в SQLite

▶ Поиск по нескольким условиям (И)

```
CREATE INDEX Idx3 ON FruitsForSale(fruit, state);
```

| fruit | state | rowid |
|------------|-------|-------|
| Apple | NC | 2 |
| Grape | CA | 5 |
| Lemon | FL | 18 |
| Orange | CA | 23 |
| Orange | FL | 1 |
| Peach | SC | 4 |
| Strawberry | NC | 19 |

```
SELECT price FROM fruitsforsale WHERE fruit='Orange' AND state='CA'
```

| fruit | state | rowid |
|------------|-------|-------|
| Apple | NC | 2 |
| Grape | CA | 5 |
| Lemon | FL | 18 |
| Orange | CA | 23 |
| Orange | FL | 1 |
| Peach | SC | 4 |
| Strawberry | NC | 19 |

| rowid | fruit | state | price |
|-------|------------|-------|-------|
| 1 | Orange | FL | 0.85 |
| 2 | Apple | NC | 0.45 |
| 4 | Peach | SC | 0.60 |
| 5 | Grape | CA | 0.80 |
| 18 | Lemon | FL | 1.25 |
| 19 | Strawberry | NC | 2.45 |
| 23 | Orange | CA | 1.05 |



```
SELECT price FROM fruitsforsale WHERE fruit='Peach'
```

| fruit | state | rowid |
|------------|-------|-------|
| Apple | NC | 2 |
| Grape | CA | 5 |
| Lemon | FL | 18 |
| Orange | CA | 23 |
| Orange | FL | 1 |
| Peach | SC | 4 |
| Strawberry | NC | 19 |

| rowid | fruit | state | price |
|-------|------------|-------|-------|
| 1 | Orange | FL | 0.85 |
| 2 | Apple | NC | 0.45 |
| 4 | Peach | SC | 0.60 |
| 5 | Grape | CA | 0.80 |
| 18 | Lemon | FL | 1.25 |
| 19 | Strawberry | NC | 2.45 |
| 23 | Orange | CA | 1.05 |



Поиск и сортировка в запросах в SQLite

- ▶ Покрывающий индекс (включены все поля поиска)

```
CREATE INDEX Idx4 ON FruitsForSale(fruit, state, price);
```

| fruit | state | price | rowid |
|------------|-------|-------|-------|
| Apple | NC | 0.45 | 2 |
| Grape | CA | 0.80 | 5 |
| Lemon | FL | 1.25 | 18 |
| Orange | CA | 1.05 | 23 |
| Orange | FL | 0.85 | 1 |
| Peach | SC | 0.60 | 4 |
| Strawberry | NC | 2.45 | 19 |

```
SELECT price FROM fruitsforsale WHERE fruit='Orange' AND state='CA';
```

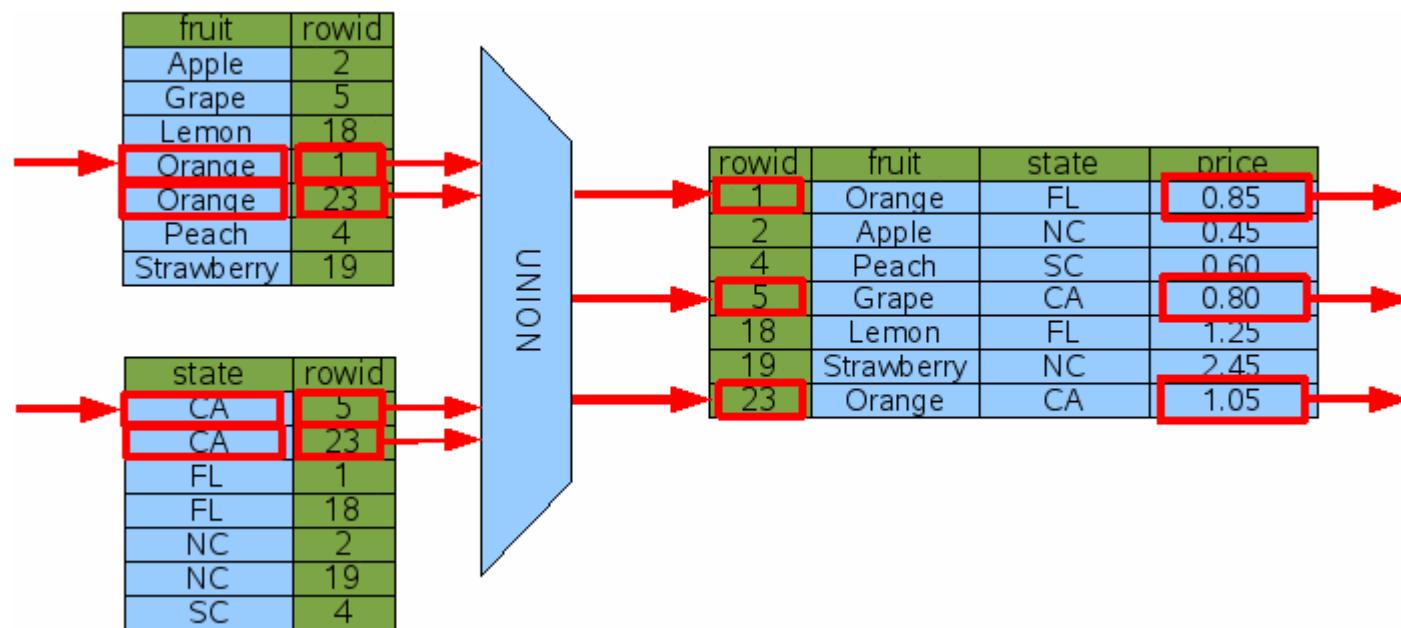


| fruit | state | price | rowid |
|------------|-------|-------|-------|
| Apple | NC | 0.45 | 2 |
| Grape | CA | 0.80 | 5 |
| Lemon | FL | 1.25 | 18 |
| Orange | CA | 1.05 | 23 |
| Orange | FL | 0.85 | 1 |
| Peach | SC | 0.60 | 4 |
| Strawberry | NC | 2.45 | 19 |

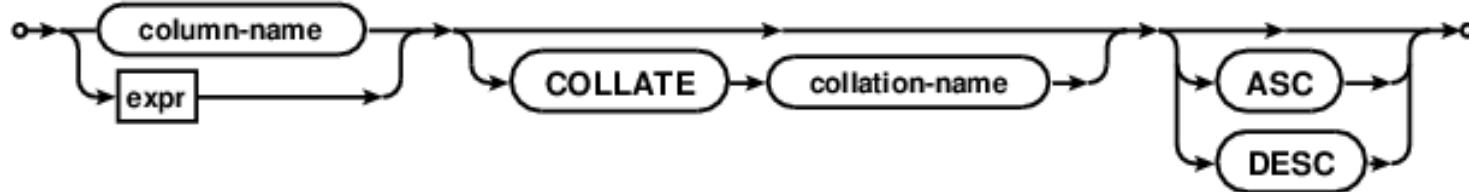
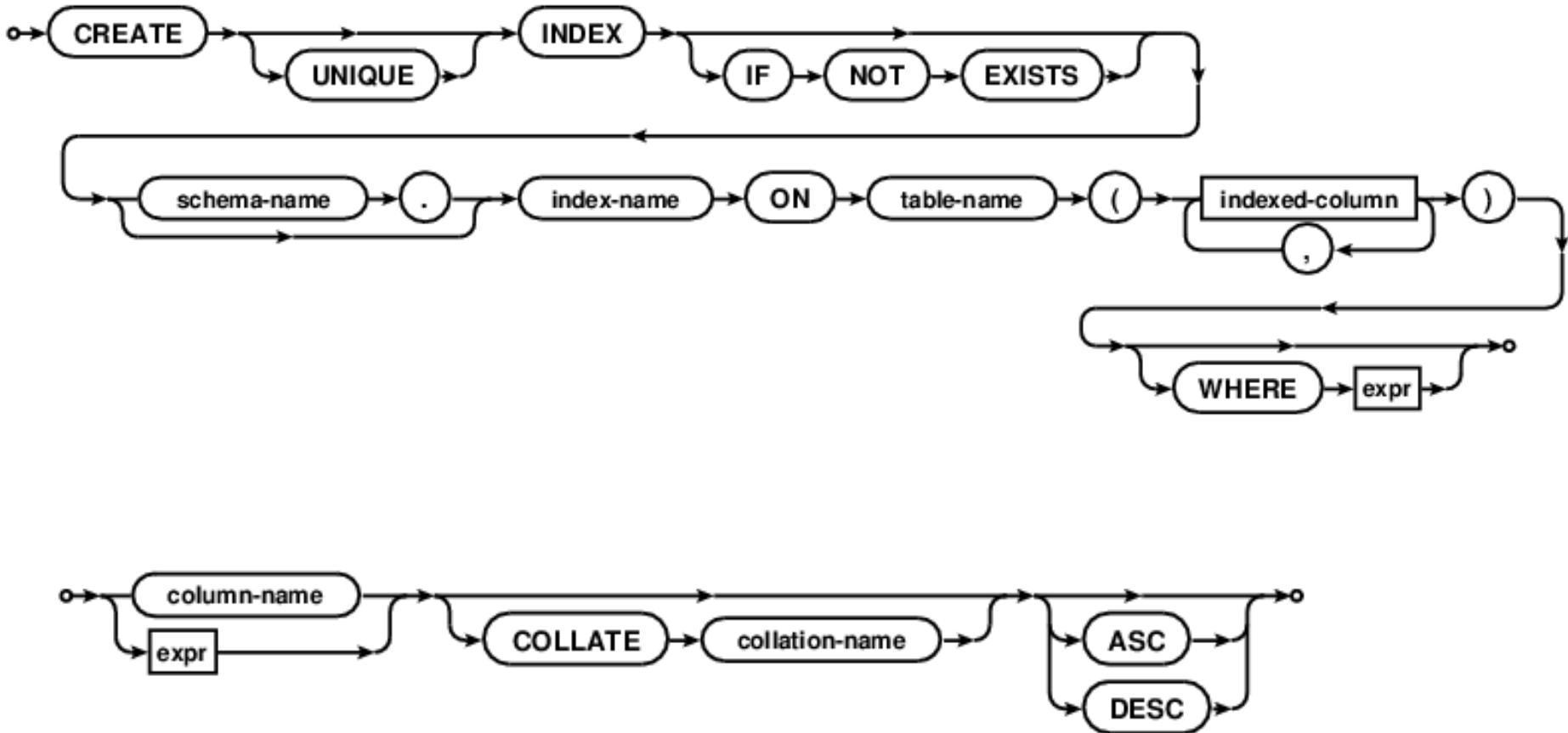
Поиск и сортировка в запросах в SQLite

▶ Поиск по нескольким условиям (ИЛИ)

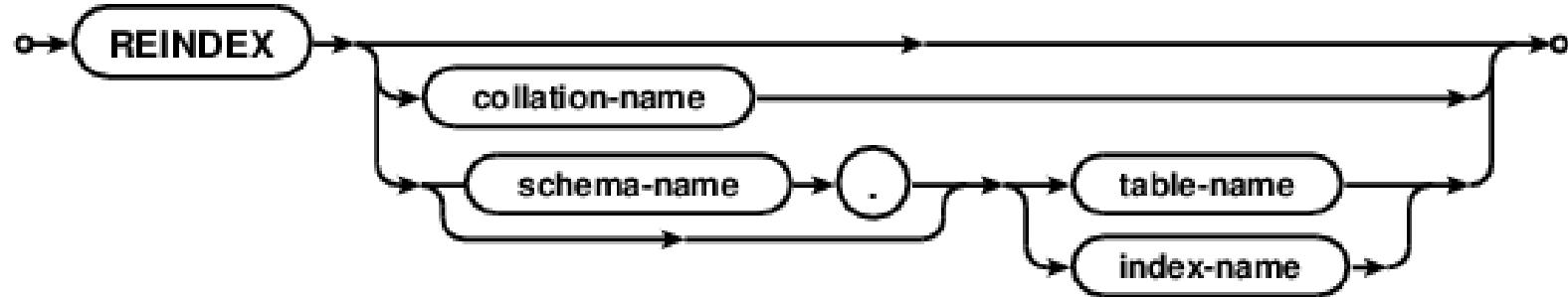
```
SELECT price FROM FruitsForSale WHERE fruit='Orange' OR state='CA';
```



Создание индекса



Пересоздание и удаление индекса



Именование индекса

- ▶ Стандартный префикс `idx_`
- ▶ Имя таблицы
- ▶ Имя столбца (столбцов)

The screenshot shows the MySQL Workbench interface. On the left, there's a tree view with a node labeled 'Индексы (1)' expanded, showing 'idx_Customers_Cu'. To the right, the SQL editor displays the command:

```
CREATE UNIQUE INDEX `idx_Customers_Cu` ON `Customers` (`CustomerName`)
```



Создание индекса

- ▶ Индекс строится на столбцах одной таблицы
- ▶ На представлении индекс построить нельзя
- ▶ На виртуальной таблице индекс построить нельзя
- ▶ Нет ограничений на количество индексов для одной таблицы
- ▶ Количество столбцов в индексе ограничено
- ▶ Если используется ключевое слово **UNIQUE** дублирование записей индекса не допускается



Автоматическое создание индекса

- ▶ При указании UNIQUE и PRIMARY KEY
- ▶ Не могут быть удалены DROP INDEX
- ▶ Показаны в sqlite_master
- ▶ Индекс по RowID не показывается в sqlite_master

| Таблицы (21) | | |
|--------------|--------|------|
| BookTable | Title | TEXT |
| BookTable | Author | TEXT |
| Books | Title | TEXT |
| Books | Author | TEXT |

CREATE TABLE "BookTable" (`Title` TEXT, `Author` TEXT UNIQUE, PRIMARY KEY(`Title`))
`Title` TEXT
`Author` TEXT UNIQUE
CREATE TABLE "Books" (`Title` TEXT UNIQUE, `Author` TEXT NOT NULL, PRIMARY KEY(`Title`)) WITHOUT ROWID
`Title` TEXT UNIQUE
`Author` TEXT NOT NULL

Автоматическое создание индекса

```
1 select * from sqlite_master where type = 'index';
2
```

| | type | name | tbl_name | rootpage | sql |
|---|-------|--------------------------------------------|-------------------------|----------|----------------------------------------------------|
| 1 | index | sqlite_autoindex_Table_B_1 | Table_B | 13 | NULL |
| 2 | index | sqlite_autoindex_Table_R_1 | Table_R | 15 | NULL |
| 3 | index | sqlite_autoindex_search_table_fts_segdir_1 | search_table_fts_segdir | 21 | NULL |
| 4 | index | idx_Customers_Cu | Customers | 17 | CREATE UNIQUE INDEX `idx_Customers_Cu` ON `Cust... |
| 5 | index | sqlite_autoindex_BookTable_1 | BookTable | 25 | NULL |
| 6 | index | sqlite_autoindex_BookTable_2 | BookTable | 31 | NULL |

```
drop index sqlite_autoindex_BookTable_1;
```

```
index associated with UNIQUE or PRIMARY KEY constraint cannot be dropped: drop index sqlite_autoindex_BookTable_1;
```

```
reindex sqlite_autoindex_BookTable_1;
```

```
Запрос успешно выполнен: reindex sqlite_autoindex_BookTable_1; (заняло 44мс)
```

Индекс на представление

```
Представления (1)
> Customers_view
CREATE VIEW Customers_view as select Customers.CustomerName from Customers
```

```
CREATE UNIQUE INDEX `idx_Customers_Cu` ON `Customers_view` (
    `CustomerName`
);
```

```
views may not be indexed: CREATE UNIQUE INDEX `idx_Customers_Cu` ON `Customers_view` (
    `CustomerName`
);
```

```
sqlite> .limit
      length 1000000000
      sql_length 1000000000
      column 2000
      expr_depth 1000
      compound_select 500
      vdbe_op 250000000
      function_arg 127
      attached 10
like_pattern_length 50000
variable_number 999
trigger_depth 1000
worker_threads 0
```

Индексы WHERE

- ▶ Частичный индекс - это индекс над подмножеством строк таблицы
- ▶ Например, частичный индекс может опускать записи, для которых индексируемый столбец NULL
- ▶ Назначение - уменьшение файла базы данных



Индексы WHERE

- ▶ Если в индексе используется OR, то в запросе, использующем индекс, может быть указано одно из условий
- ▶ Если в индексе используется точное равенство (=a), то запрос будет использовать индекс при таком же условии (between a and a – не подойдет)
- ▶ Если в индексе указано IS NOT NULL, то в запросе, использующем <> индекс не применяется

```
CREATE INDEX `idx_Goods_Descr_Price` ON `Goods` (
    `GoodDescr`,
    `Price`
) WHERE GoodDescr is not null or Price > 0;
```

Составные индексы

- ▶ Используются для ускорения выполнения запросов по нескольким полям
- ▶ Порядок расположения столбцов важен!



Индексы

```
CREATE INDEX `idx_Orders_Date_IdCustomer` ON `Orders` (
    `OrderDate` ASC,
    `idCustomer`      ASC
) WHERE OrderDate > '2016-12-31';
```

Запрос успешно выполнен: CREATE INDEX `idx_Orders_Date_IdCustomer` ON `Orders` (
 `OrderDate` ASC,
 `idCustomer` ASC
) WHERE OrderDate > '2016-12-31'; (заняло 0мс)

Индексы с указанием порядка сортировки

- ▶ Указание порядка хранения значений ключей оправдано для запроса с предложением ORDER BY

```
create index `idx_Goods_Descr` on 'Goods' ('GoodDescr' desc);
```

```
Запрос успешно выполнен: create index `idx_Goods_Descr` on 'Goods' ('GoodDescr' desc); (заняло 37мс)
```

Индексы

- ▶ COLLATE определяет последовательность сортировки для текстовых записей

```
CREATE INDEX `idx_Orders_Details` ON `Orders` (
    `OrderDetails`  collate rtrim ASC);
```

Индексы в выражениях

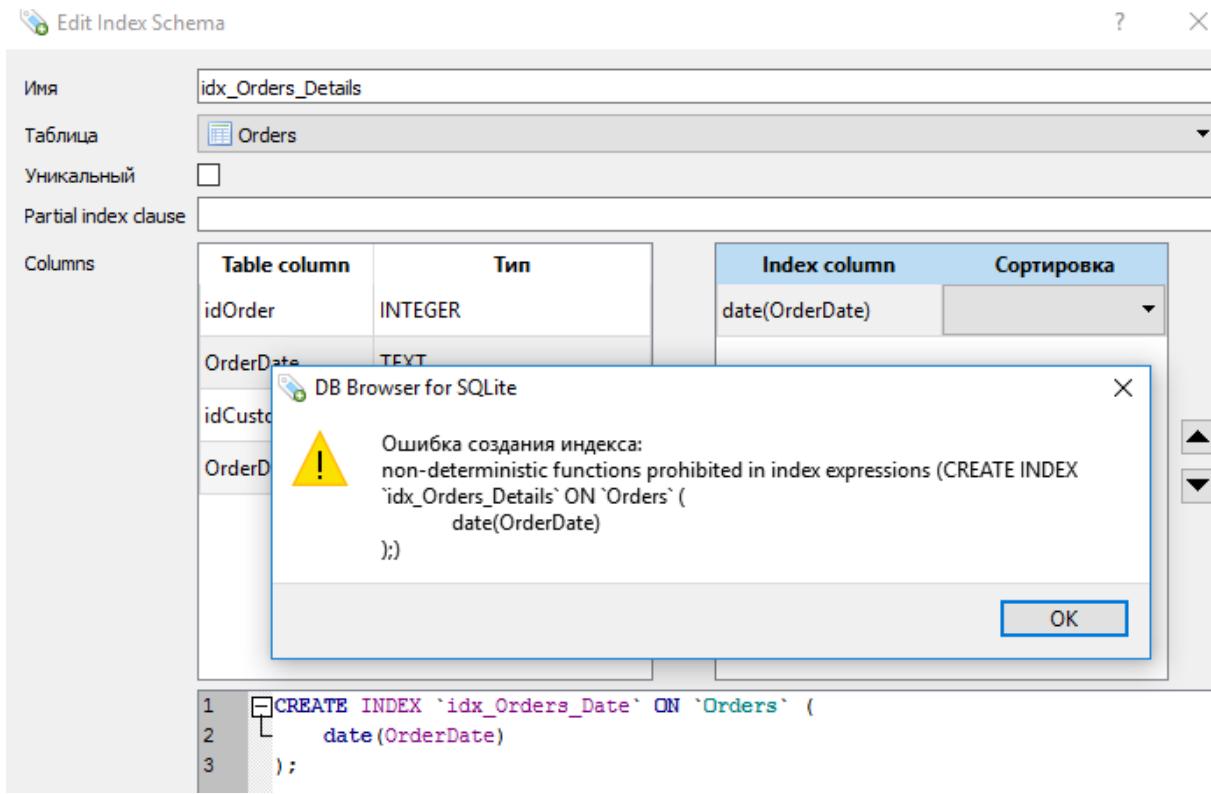
- ▶ В индексе можно использовать выражение от столбца/набора столбцов



CREATE INDEX `idx_BookTable_Title` ON `BookTable` (substr(`Title`, 1))

Индексы в выражениях

- Выражения в индексе не могут ссылаться на другие таблицы и использовать подзапросы и функции, результат которых может измениться



Оптимизация запросов

- ▶ Используются только для оперативного анализа
- ▶ ANALYZE
- ▶ EXPLAIN
- ▶ EXPLAIN QUERY PLAN

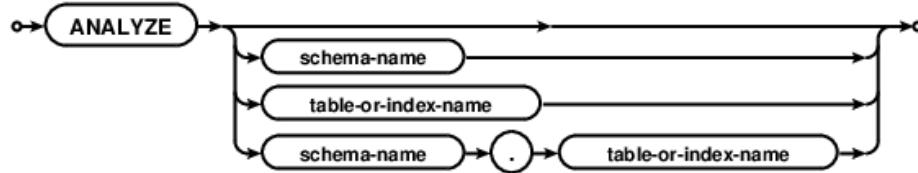


ANALYZE

- ▶ Сбор статистики о таблицах и индексах
- ▶ Хранится в `sqlite_stat1`
- ▶ Оптимизатор запросов может получить доступ к информации и использовать ее, чтобы помочь улучшить выбор планирования запросов



ANALYZE



```
1 analyze
2
3 analyze CUSTOMERS;
4
5 select * from sqlite_stat1;
6
```

| | tbl | idx | stat |
|---|-----------|------------------------------|--------|
| 1 | ORDERS | NULL | 10 |
| 2 | SALESREPS | sqlite_autoindex_SALESREPS_1 | 10 1 |
| 3 | OFFICES | sqlite_autoindex_OFFICES_1 | 5 1 |
| 4 | PRODUCTS | sqlite_autoindex_PRODUCTS_1 | 25 5 1 |
| 5 | CUSTOMERS | NULL | 21 |

EXPLAIN

- ▶ Пошаговое исполнение оператора
- ▶ Оператор при этом не выполняется

```
1   explain
2   select o.order_num, o.order_date, c.company, o.amount
3   from CUSTOMERS as c join ORDERS as o
4   on c.cust_num = o.CUST
5   where amount < 30000;
6
```

| | addr | opcode | p1 | p2 | p3 | p4 | p5 | comment |
|---|------|-----------|----|----|----|----------|----|---------|
| 1 | 0 | Init | 0 | 17 | 0 | | 00 | NULL |
| 2 | 1 | OpenRead | 1 | 9 | 0 | 8 | 00 | NULL |
| 3 | 2 | OpenRead | 0 | 8 | 0 | 2 | 00 | NULL |
| 4 | 3 | Rewind | 1 | 14 | 0 | | 00 | NULL |
| 5 | 4 | Column | 1 | 7 | 1 | | 00 | NULL |
| 6 | 5 | Ge | 2 | 13 | 1 | (BINARY) | 53 | NULL |
| 7 | 6 | Column | 1 | 2 | 3 | | 00 | NULL |
| 8 | 7 | SeekRowid | 0 | 13 | 3 | | 00 | NULL |

EXPLAIN

| | addr | opcode | p1 | p2 | p3 | p4 | p5 | comment |
|----|------|-------------|-------|----|----|-----------|----|---------|
| 9 | 8 | Rowid | 1 | 4 | 0 | | 00 | NULL |
| 10 | 9 | Column | 1 | 1 | 5 | | 00 | NULL |
| 11 | 10 | Column | 0 | 1 | 6 | | 00 | NULL |
| 12 | 11 | Copy | 1 | 7 | 0 | | 00 | NULL |
| 13 | 12 | ResultRow | 4 | 4 | 0 | | 00 | NULL |
| 14 | 13 | Next | 1 | 4 | 0 | | 01 | NULL |
| 15 | 14 | Close | 1 | 0 | 0 | | 00 | NULL |
| 16 | 15 | Close | 0 | 0 | 0 | | 00 | NULL |
| 17 | 16 | Halt | 0 | 0 | 0 | | 00 | NULL |
| 18 | 17 | Transaction | 0 | 0 | 11 | 0 | 01 | NULL |
| 19 | 18 | TableLock | 0 | 9 | 0 | ORDERS | 00 | NULL |
| 20 | 19 | TableLock | 0 | 8 | 0 | CUSTOMERS | 00 | NULL |
| 21 | 20 | Integer | 30000 | 2 | 0 | | 00 | NULL |
| 22 | 21 | Goto | 0 | 1 | 0 | | 00 | NULL |



EXPLAIN QUERY PLAN

- ▶ Показывает, каким образом будет проводится поиск в таблице

```
1 explain query plan
2 select o.order_num, o.order_date, c.company, o.amount
3 from CUSTOMERS as c join ORDERS as o
4 on c.cust_num = o.CUST
5 where amount < 30000;
6
```

| | selectid | order | from | detail |
|---|----------|-------|------|----------------------------------------|
| 1 | 0 | 0 | 1 | SCAN TABLE ORDERS AS o |
| 2 | 0 | 1 | 0 | SEARCH TABLE CUSTOMERS AS c USING I... |

EXPLAIN QUERY PLAN

```
1 explain query plan
2 select o.order_num, o.order_date, c.company, o.amount
3 from CUSTOMERS as c join ORDERS as o
4 on c.cust_num = o.CUST
5 where amount < 30000;
6
7
8 create index idx_Orders_Amount on ORDERS(amount);
9 create index idx_Orders_Cust on ORDERS(cust);
10 |
```

| | selectid | order | from | detail |
|---|----------|-------|------|-------------------------------------------------------------------|
| 1 | 0 | 0 | 1 | SEARCH TABLE ORDERS AS o USING INDEX idx_Orders_Amount (AMOUNT<?) |
| 2 | 0 | 1 | 0 | SEARCH TABLE CUSTOMERS AS c USING INTEGER PRIMARY KEY (rowid=?) |

PRAGMA

- ▶ **INDEX_INFO** (*indexname*) – возвращает одну строку для каждого столбца ключа в названном индексе
- ▶ **INDEX_LIST** (*tablename*) – возвращает одну строку для каждого индекса, связанного с данной таблицей
- ▶ **INDEX_XINFO** (*indexname*) – возвращает информацию о каждом столбце в индексе



PRAGMA

```
11
12     pragma index_list(Orders);
13
```

| | seq | name | unique | origin | partial |
|---|-----|-------------------|--------|--------|---------|
| 1 | 0 | idx_Orders_Cust | 0 | c | 0 |
| 2 | 1 | idx_Orders_Amount | 0 | c | 0 |

```
11
12     pragma index_info(idx_Orders_Amount);
```

| | seqno | cid | name |
|---|-------|-----|--------|
| 1 | 0 | 7 | AMOUNT |

Триггер

- ▶ Триггер – это особая разновидность хранимых процедур, исполняемых в ответ на какое-то событие
- ▶ Используется для:
 - ▶ Обеспечения целостности данных
 - ▶ Реализации сложной бизнес-логики
 - ▶ Аудита изменений
 - ▶ Каскадного удаления/обновления данных



Триггер – классификация

- ▶ По группам событий (DML, DDL, системные)
 - ▶ По событию (INSERT, UPDATE, DELETE)
 - ▶ По времени события (BEFORE, AFTER, INSTEAD OF)
 - ▶ Операторные или строчные
-
- ▶ Как срабатывает триггер – до констрайнта или после?
 - ▶ Триггер INSTEAD OF
 - ▶ Проверка ограничений
 - ▶ Триггер AFTER



Триггер – часть транзакции

- ▶ Триггер и операция, его вызвавшая, выполняются в одной транзакции:
 - ▶ Если не выполняется операция – не выполняется и триггер
 - ▶ Если триггер не может выполниться – то и операция откатится



Триггер DML

- ▶ Каков механизм доступа к данным до изменения и после изменения?



Триггер DML

- ▶ Как использовать вложенные и рекурсивные триггеры?
 - ▶ Вложенный триггер возникает при выполнении триггером действия, вызывающим другой триггер
 - ▶ 32 уровня вложенности
 - ▶ Прямая и косвенная рекурсии

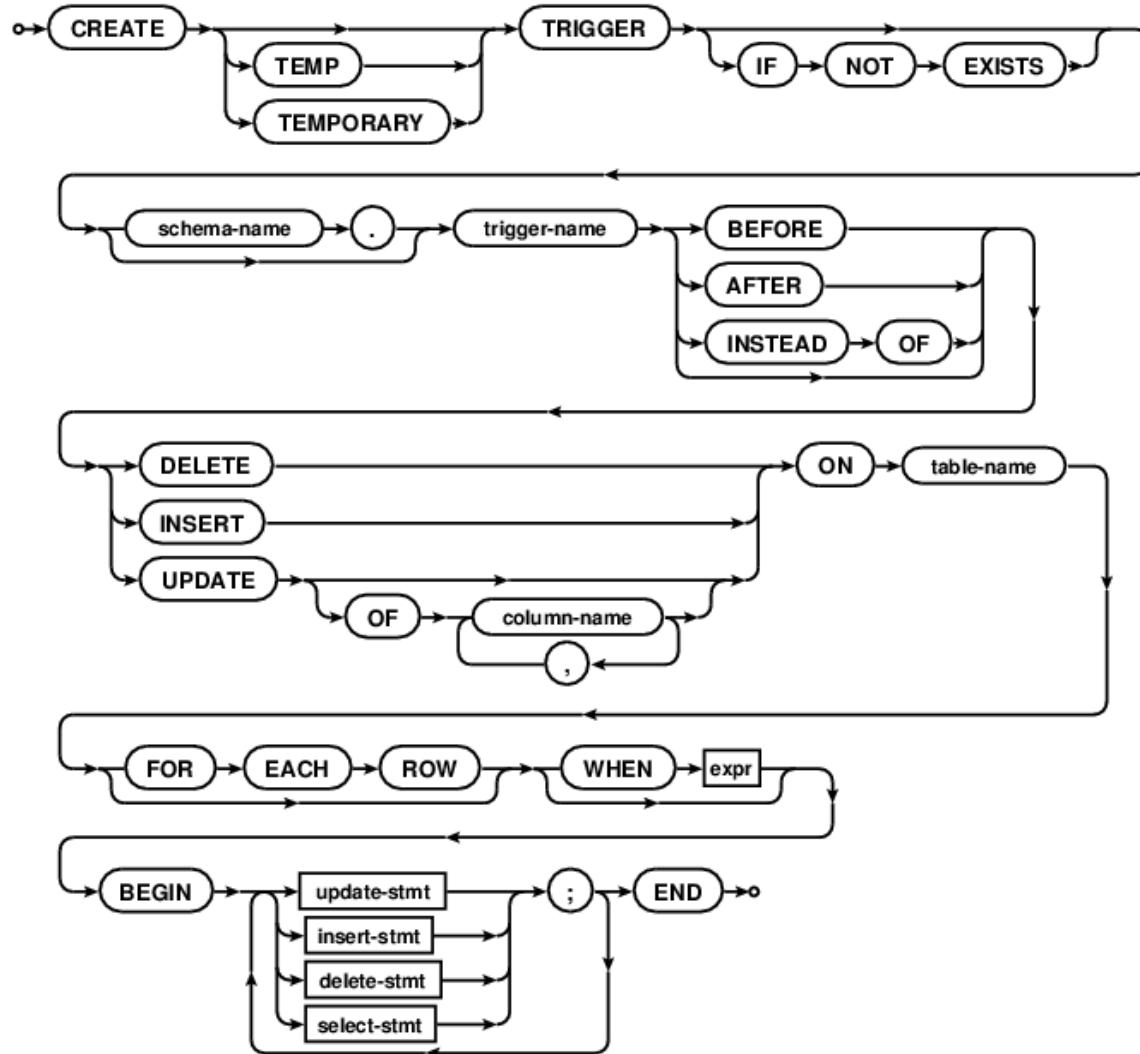


Триггер DML

- ▶ Можно ли создавать несколько разных триггеров одного типа?
- ▶ В каком порядке они будут выполняться?



CREATE TRIGGER



Триггер SQLite

- ▶ TEMP / TEMPORARY – временный триггер будет существовать до тех пор, пока пользователь не разорвет соединение
- ▶ Могут быть созданы не только для временных таблиц, но и для обычных таблиц



Триггер SQLite

- ▶ Начинается тело триггера оператором BEGIN
- ▶ Заканчивается – оператором END

```
create table Audit_Tbl (tmstmp);

drop table Audit_Tbl;
drop trigger tr_Aft_Update_Orders;

create trigger if not exists tr_Aft_Update_Orders
after update
on ORDERS
begin
    insert into Audit_Tbl values ('Orders' || current_timestamp);
end;
```

```
update ORDERS set cust = 2125 where order_num = 113055;
```

| | tmstmp |
|---|---------------------------|
| 1 | Orders2023-05-02 14:49:53 |

Триггер SQLite – FOR EACH ROW

- ▶ **FOR EACH ROW** – в текущей версии поддерживаются только строчные триггеры
- ▶ при написании триггера **FOR EACH ROW** можно пропускать

```
update ORDERS set cust = 2125;
```

```
15 select * from Audit_Tbl;  
16
```

| | tmstmp |
|----|---------------------------|
| 1 | Orders2023-05-02 14:49:53 |
| 2 | Orders2023-05-02 14:51:18 |
| 3 | Orders2023-05-02 14:51:18 |
| 4 | Orders2023-05-02 14:51:18 |
| 5 | Orders2023-05-02 14:51:18 |
| 6 | Orders2023-05-02 14:51:18 |
| 7 | Orders2023-05-02 14:51:18 |
| 8 | Orders2023-05-02 14:51:18 |
| 9 | Orders2023-05-02 14:51:18 |
| 10 | Orders2023-05-02 14:51:18 |
| 11 | Orders2023-05-02 14:51:18 |

Триггер SQLite - BEFORE

```
create trigger if not exists tr_Bef_Insert_Customers
before insert
on CUSTOMERS
begin
    insert into Audit_Tbl values ('CUSTOMERS' || current_timestamp);
end;

insert into CUSTOMERS values (2127, 'Eternity', 107, 50000);
```

| | | |
|----|-----------|----------------|
| 12 | CUSTOMERS | 2023-05-02 ... |
|----|-----------|----------------|



Триггер SQLite

- ▶ Можно использовать фразу WHEN для уточнения условия, при котором триггер выполнится
- ▶ Условие WHEN интерпретируется как логическое
- ▶ Могут применяться OLD и NEW префиксы

```
create trigger if not exists tr_Bef_Insert_Customers_limit_more_50000
before insert
on CUSTOMERS
when new.Credit_Limit > 50000
begin
    insert into Audit_Tbl values ('CUSTOMERS_more_50000' || current_timestamp);
end;

insert into CUSTOMERS values (2128, 'Local JSC',      106, 50000);
insert into CUSTOMERS values (2129, 'MegaCosm',     106, 60000);
```

- ▶ Триггеры выполняются в неопределенном порядке

| | | |
|----|----------------------|---------------------|
| 12 | CUSTOMERS | 2023-05-02 14:54:09 |
| 13 | CUSTOMERS | 2023-05-02 15:00:12 |
| 14 | CUSTOMERS_more_50000 | 2023-05-02 15:00:15 |
| 15 | CUSTOMERS | 2023-05-02 15:00:15 |

Триггер SQLite – момент выполнения ограничения целостности

- ▶ При нарушении ограничения целостности триггер не выполняется
- ▶ Повторно добавляем строки для нарушения ограничения целостности

```
insert into CUSTOMERS values (2128, 'Local JSC', 106, 50000);
insert into CUSTOMERS values (2129, 'MegaCosm', 106, 60000);
```

| | | |
|----|-----------|----------------|
| 12 | CUSTOMERS | 2023-05-02 ... |
| 13 | CUSTOMERS | 2023-05-02 ... |
| 14 | CUSTOMERS | _more_5000... |
| 15 | CUSTOMERS | 2023-05-02 ... |



Триггер SQLite - INSTEAD OF

```
create trigger
if not exists
tr_Customers_View_Instead_Delete
instead of delete on Customers_view
begin
    insert into Audit (DateOfChange, TextOfChange) values (datetime(), 'Попытка удаления из представления');
end;
```

```
delete from Customers_view
```

| | | |
|----|---------------------|-----------------------------------|
| 11 | 2017-10-12 19:49:29 | 16.0 |
| 12 | 2017-10-12 19:55:30 | Попытка удаления из представления |
| 13 | 2017-10-12 19:55:30 | Попытка удаления из представления |
| 14 | 2017-10-12 19:55:30 | Попытка удаления из представления |
| 15 | 2017-10-12 19:55:30 | Попытка удаления из представления |
| 16 | 2017-10-12 19:55:30 | Попытка удаления из представления |
| 17 | 2017-10-12 19:55:30 | Попытка удаления из представления |
| 18 | 2017-10-12 19:55:30 | Попытка удаления из представления |

Триггер SQLite

- ▶ Для команды UPDATE можно указать столбец, который будет отслеживать триггер

```
create trigger if not exists tr_Aft_Update_Orders_Amount
after update of Amount
on ORDERS
begin
    insert into Audit_Tbl values ('Amount_of_Order' || current_timestamp);
end;
```

```
update ORDERS set cust = 2127 where order_num = 113055;
update ORDERS set cust = 2128, amount = 12000 where order_num = 113045;
```

16 Orders2023-05-02 15:11:43

17 Amount_of_Order2023-05-02 15:11:45

18 Orders2023-05-02 15:11:45

Триггер SQLite – разрешение конфликтов

- ▶ RAISE () – специальная функция обработки ошибки в триггере
- ▶ Два аргумента – действие и комментарий

```
create trigger if not exists tr_Aft_Delete_Customers
after delete
on CUSTOMERS
begin
    select RAISE(ABORT, 'You can''t delete records from CUSTOMERS');
end;
```

```
You can't delete records from CUSTOMERS: delete from CUSTOMERS;
```

Триггер SQLite – RAISE ()

- ▶ IGNORE – игнорировать строку, породившую конфликт и продолжить выполнение последующих операций
- ▶ ROLLBACK – откатить все операции к исходному состоянию
- ▶ ABORT – отменить не все выполненные ранее операции, а только ту, при которой возник конфликт
- ▶ FAIL – прервать выполнение текущей операции, сохранить результаты успешных операций, остальные не выполнять



Триггер SQLite – вложенность и рекурсия

- ▶ **SQLITE_LIMIT_TRIGGER_DEPTH = 1000**
- ▶ Нет рекурсивности при 0

A screenshot of a SQLite command-line interface. The top part shows the command: `51 pragma recursive_triggers;`. The bottom part shows the result of the command: a table named `recursive_triggers` with one row containing values `1 0`.

| | recursive_triggers |
|---|--------------------|
| 1 | 0 |

Триггер SQLite – просмотр в БД

```
65 select * from sqlite_master;
```

| # | type | name | tbl_name | rootpage | sql |
|----|---------|------------------------------------------|--------------|----------|----------------------------------------|
| 9 | view | v_order_less_5000 | v_order_l... | 0 | CREATE VIEW v_order_less_5000 assel... |
| 10 | table | Audit_Tbl | Audit_Tbl | 10 | CREATE TABLE Audit_Tbl (tmstmp) |
| 11 | trigger | tr_Aft_Update_Orders | ORDERS | 0 | CREATE TRIGGER tr_Aft_Update_Order... |
| 12 | trigger | tr_Bef_Insert_Customers | CUSTOM... | 0 | CREATE TRIGGER tr_Bef_Insert_Custom... |
| 13 | trigger | tr_Bef_Insert_Customers_limit_more_50000 | CUSTOM... | 0 | CREATE TRIGGER tr_Bef_Insert_Custom... |
| 14 | trigger | tr_Aft_Update_Orders_Amount | ORDERS | 0 | CREATE TRIGGER tr_Aft_Update_Order... |
| 15 | trigger | tr_Aft_Update_Customers_NoRep | CUSTOM... | 0 | CREATE TRIGGER tr_Aft_Update_Custo... |
| 16 | trigger | tr_Aft_Delete_Customers | CUSTOM... | 0 | CREATE TRIGGER tr_Aft_Delete_Custo... |

Триггер SQLite – удаление таблиц

- ▶ При удалении таблицы все связанные триггеры также удаляются
- ▶ Если удалена таблица, на которую есть ссылка в теле триггера, то возвращается ошибка, триггер надо переопределить

```
drop table Audit_Tbl;  
update ORDERS set cust = 2125 where order_num = 113055;
```

```
no such table: main.Audit_Tbl: update ORDERS set cust = 2125 where order_num = 113055;
```

Вопросы?



Базы данных

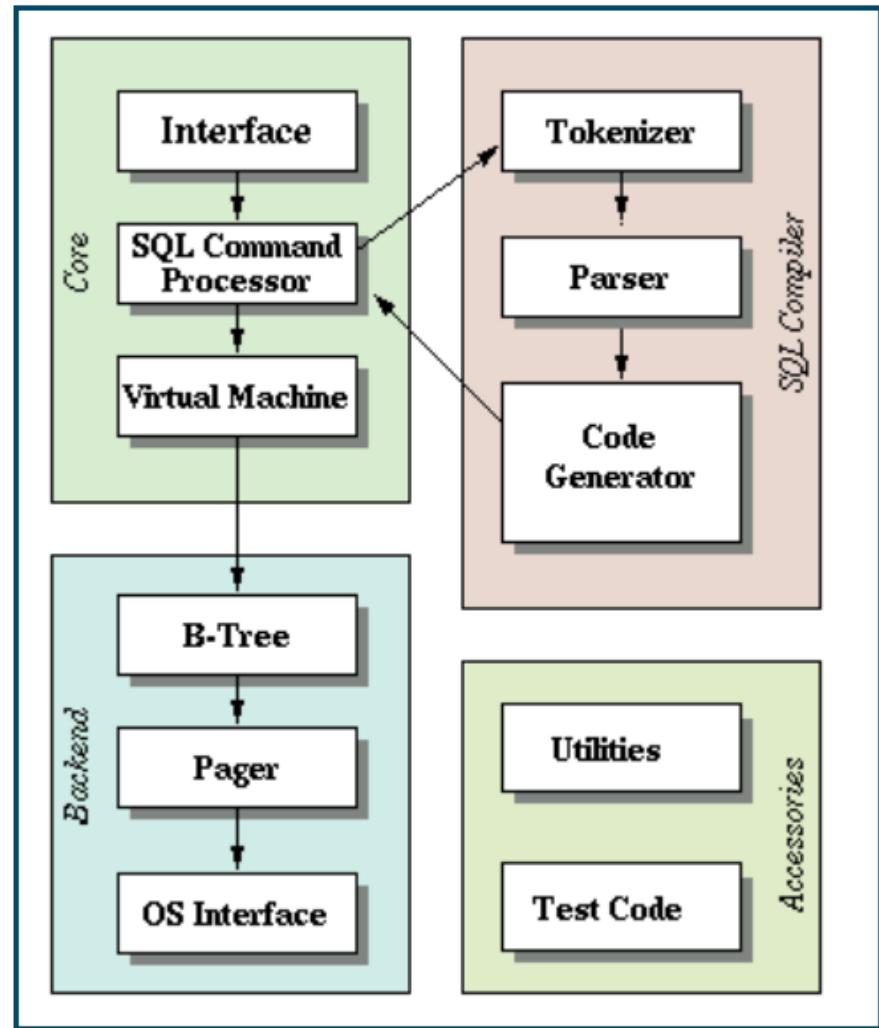
Лекция 15

Устройство БД SQLite и совместный доступ



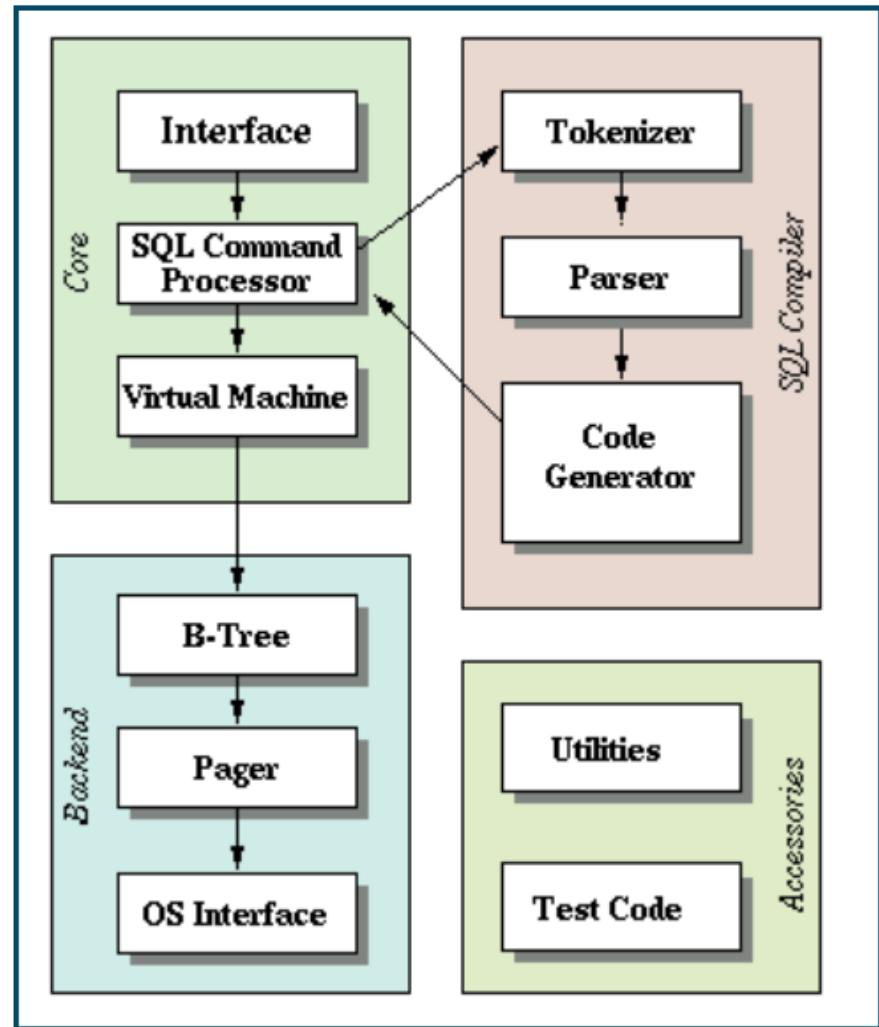
SQLite – архитектура – Core

- ▶ Библиотека
- ▶ Написана на С
- ▶ Amalgamation – объединенный в один файл набор файлов библиотеки
- ▶ Компилирует текст в байткод, затем запускает его на виртуальной машине Virtual Database Engine (Vdbe)



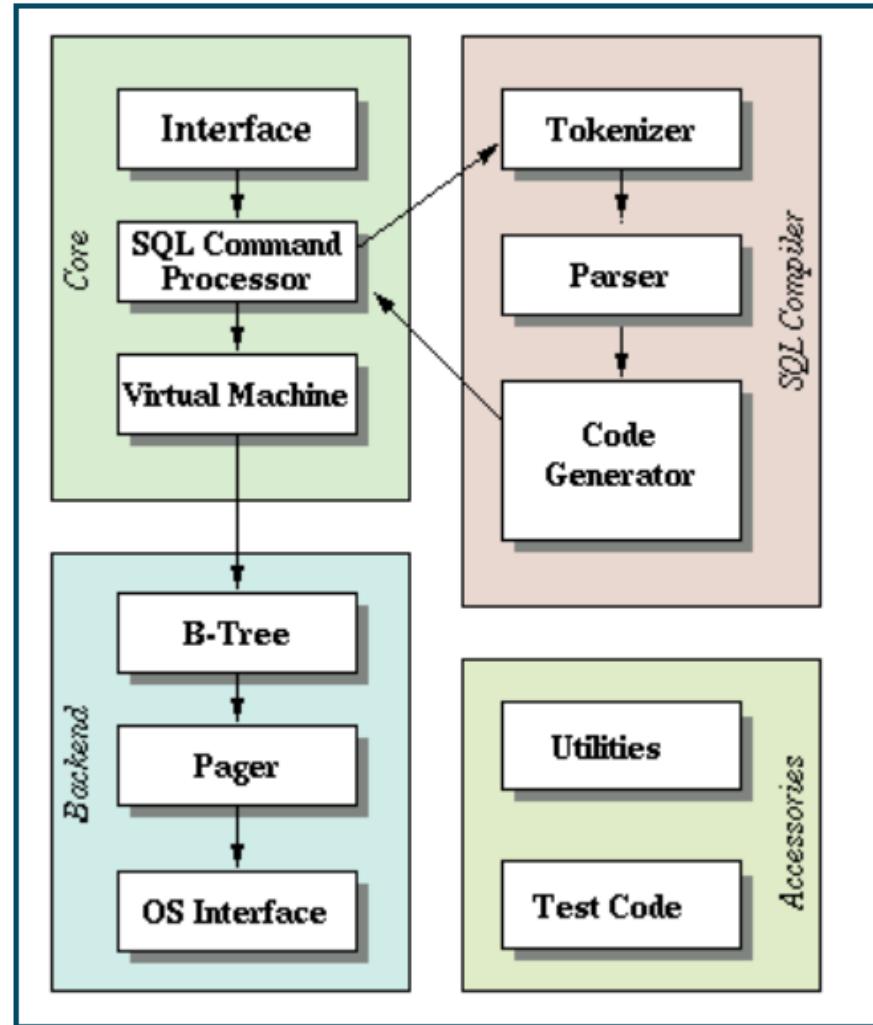
SQLite – архитектура – SQL Compiler

- ▶ Синтаксический разбор в токенизаторе
- ▶ Парсер собирает токены в дерево разбора
- ▶ Генератор кода – планировщик запросов



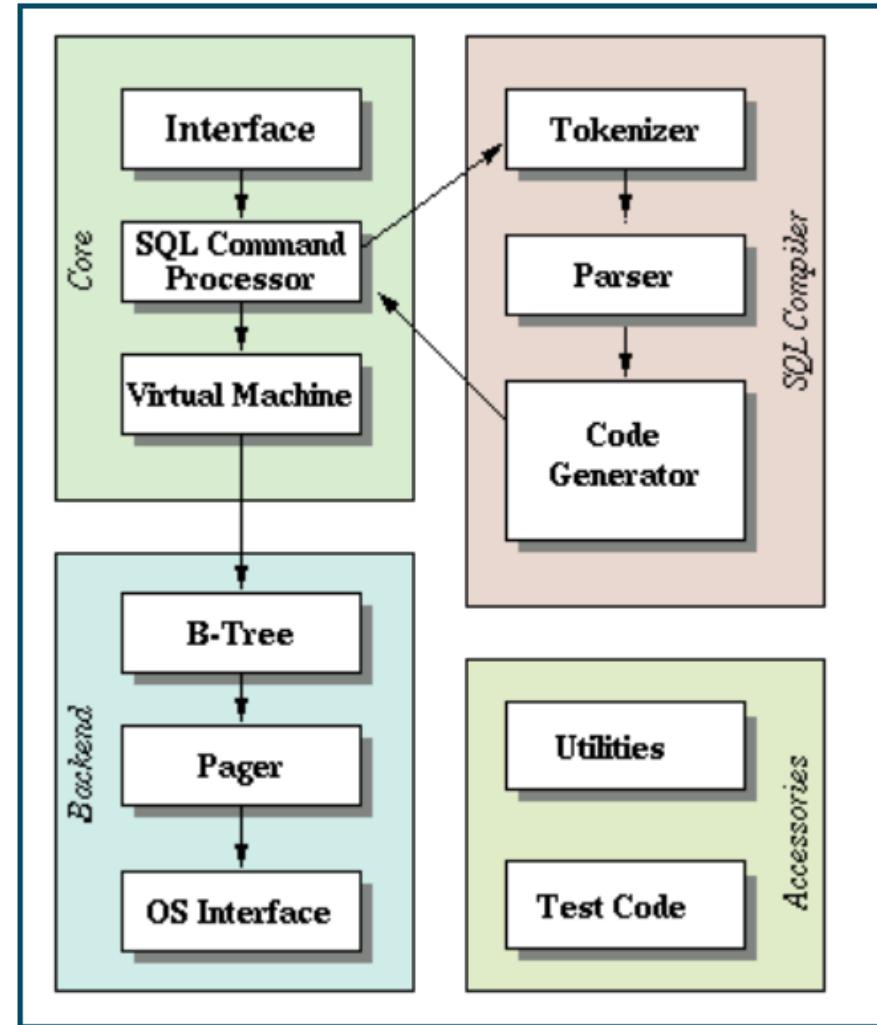
SQLite – архитектура – Backend

- ▶ Данные хранятся в виде B-tree
- ▶ Pager – модуль обработки страниц данных (кэш)
- ▶ OS Interface – абстракция обращения к операционной системе



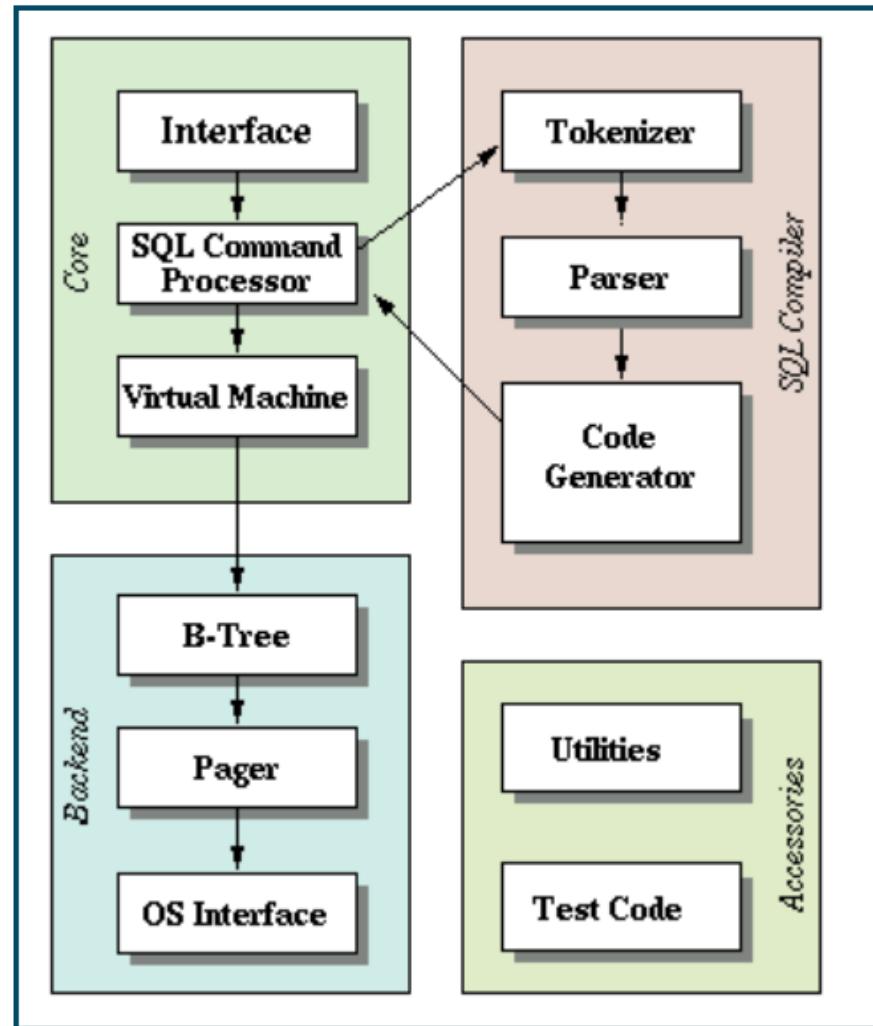
SQLite – архитектура – B-tree

- ▶ Все B-деревья хранятся в одном файле диска
- ▶ Для каждой таблицы и индекса используется отдельное B-дерево
- ▶ Модуль B-tree запрашивает с диска информацию фиксированного размера в страницах данных



SQLite – архитектура – PAGER

- ▶ Размер страницы – 4096 байт, [512 – 65536] байт
- ▶ Отвечает за чтение, запись и кеширование страниц
- ▶ Обеспечивает фиксацию и откат транзакции, обеспечивает блокировку файла



Файлы базы данных

- ▶ Состоит из страниц
 - ▶ Размер от 512 байт до 65536
 - ▶ Все страницы одного размера
-
- ▶ Каждая страница может быть:
 - ▶ Страницей блокировки
 - ▶ Свободной страницей
 - ▶ Страницей В-дерева
 - ▶ Страницей переполнения
 - ▶ Страницей указателей



Заголовок файла базы данных

- ▶ Стока заголовка
- ▶ Размер страницы базы данных в байтах
- ▶ Вид журнала, используемый при чтении или записи (обычный или WAL)
- ▶ Счетчик изменений файлов
- ▶ Размер файла в страницах
- ▶ Свободные страницы
- ▶ Размер кеша страницы по умолчанию
- ▶ Кодировка текста
- ▶ Версия SQLite и другие



Архитектура разделения одновременного доступа

- ▶ Одновременно несколько соединений могут читать БД
- ▶ Записывать в данный момент времени может только одно соединение

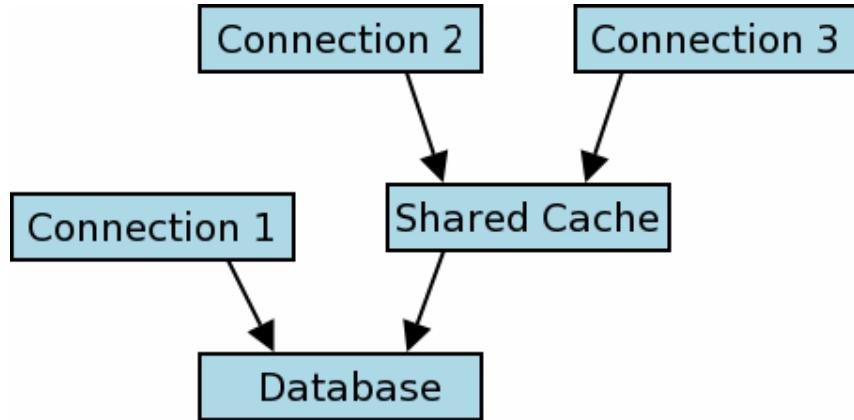


Архитектура разделения одновременного доступа

- ▶ Вначале в журнал отката записываются неизмененные данные и размер файла БД
- ▶ Затем изменяются данные в БД
- ▶ Затем стирается журнал



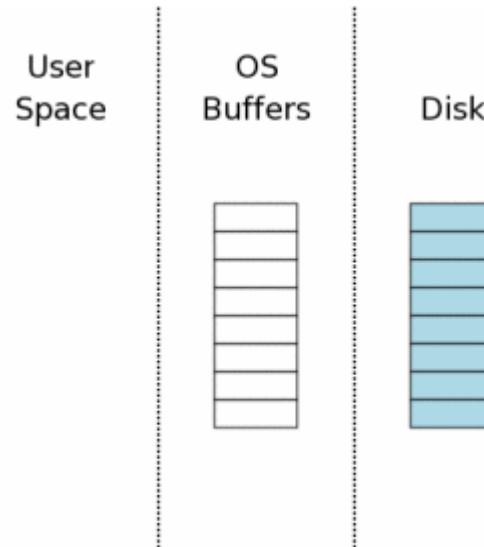
Разделяемый кеш



- ▶ По умолчанию отключен
- ▶ enable_shared_cache
- ▶ Блокировки:
 - ▶ блокировка уровня транзакции
 - ▶ блокировка уровня таблицы
 - ▶ блокировка уровня схемы



Начальное состояние

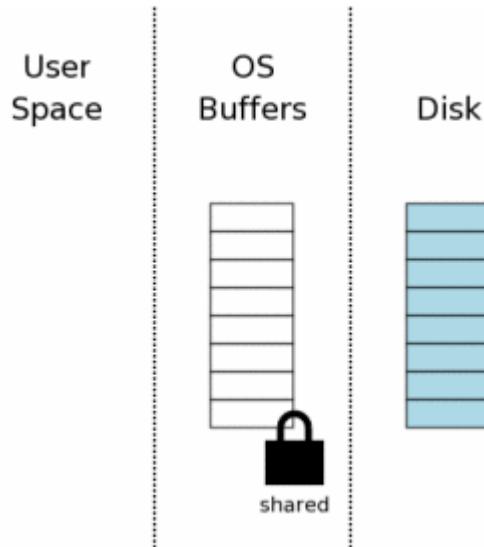


- ▶ Перед установлением соединения:
- ▶ Данные на диске неизменны
- ▶ Буферный кэш пуст
- ▶ Процесс SQLite пуст



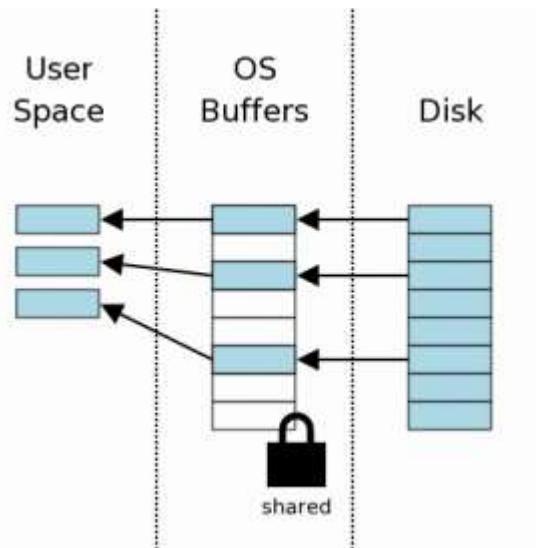
Соединение с базой данных

- ▶ Получение разделяемой блокировки на процесс чтения, блокировки файлов - обычно флаги внутри ядра ОС

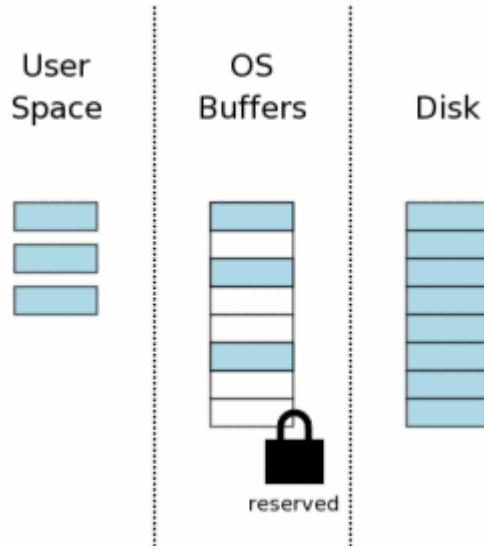


Чтение из базы данных

- ▶ Считано 3 страницы
- ▶ При последующих считываниях данные могут быть взяты из кэша



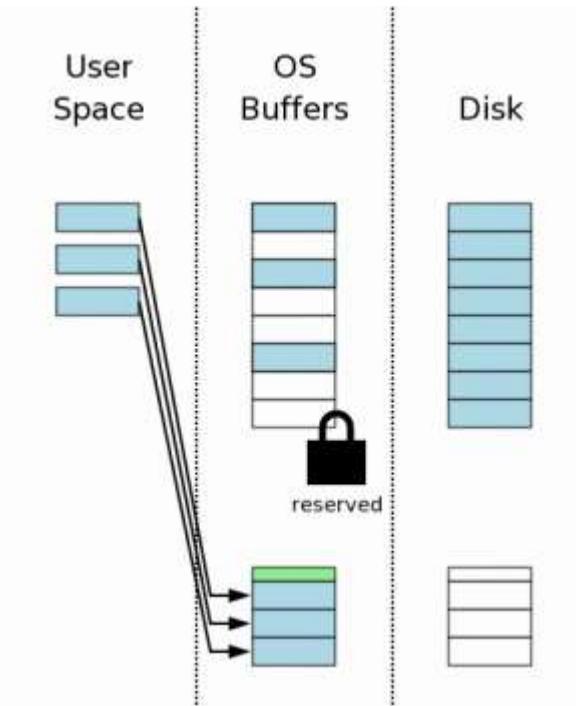
Резервирование перед записью



- ▶ Процесс намеревается изменить файл базы данных в ближайшем будущем
- ▶ Еще не начал вносить изменения
- ▶ Другие процессы могут продолжать читать
- ▶ Ни один процесс не должен начинать попытки записи в базу данных

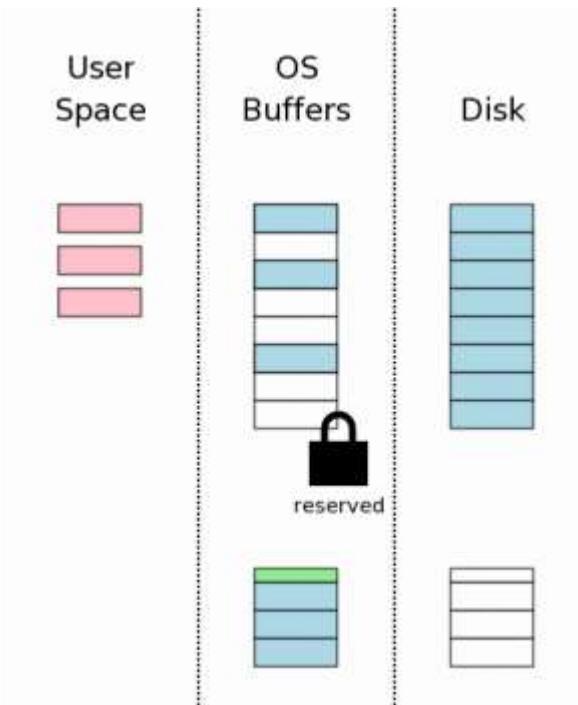


Журнал отката



- ▶ Создается отдельный журнал отката в буфере
- ▶ Зеленым указан заголовок, содержит исходный размер файла базы данных
- ▶ Когда будет время ОС сбросит буфер на диск

Изменение пользовательских данных

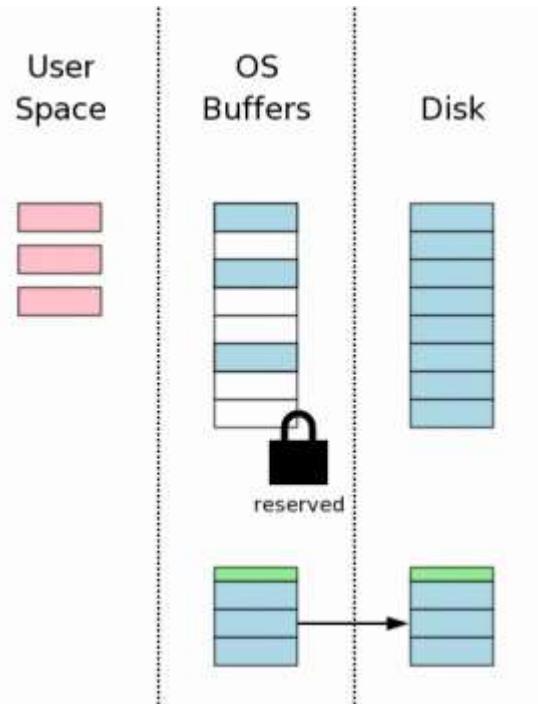


- ▶ Производится изменение в пользовательском подключении к базе данных
- ▶ Другие соединения могут продолжать видеть то, что было

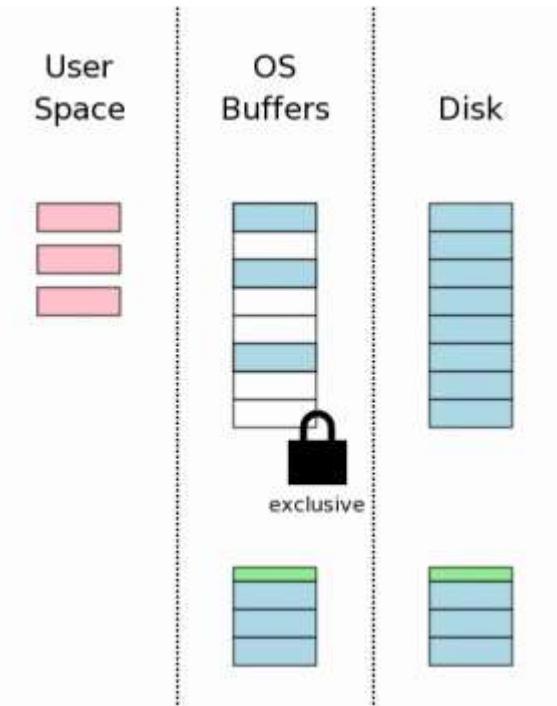


Сброс в файл журнала

- ▶ Сброс буфера в файл журнала на диске
- ▶ Вначале содержимое, потом заголовок

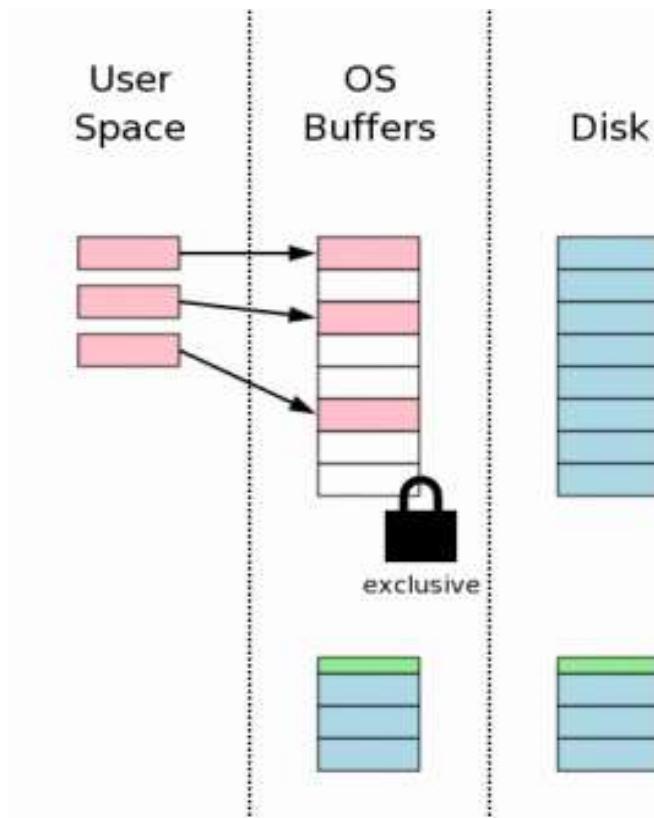


Захват монопольной блокировки



- ▶ Двухэтапный процесс – вначале блокировка ожидания (*pending*), затем монопольная (*exclusive*)
- ▶ При блокировке ожидания другие процессы дочитывают, новые не могут подключиться

Фиксация изменений

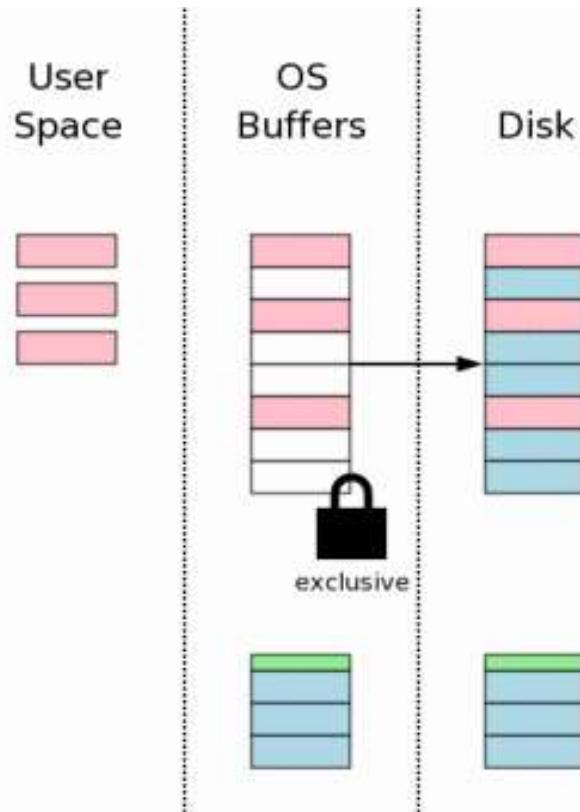


▶ Вначале изменения вносятся в буфер

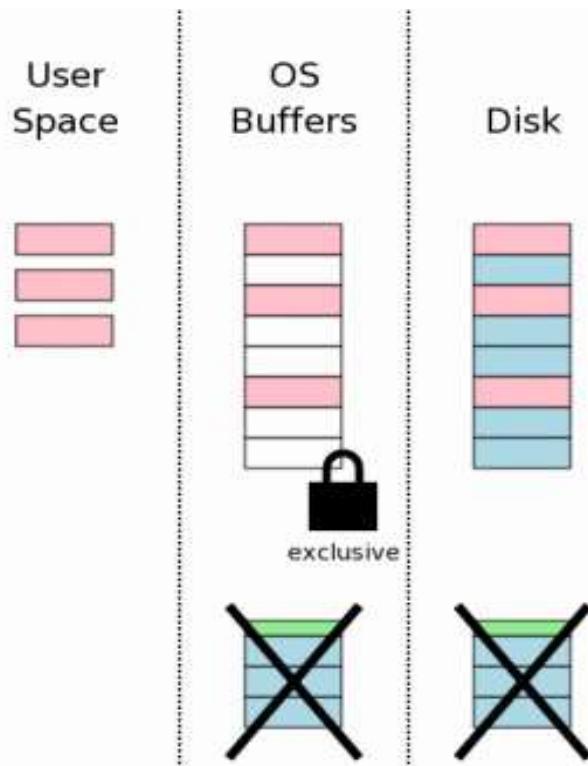


Фиксация изменений

▶ Затем данные записываются на диск



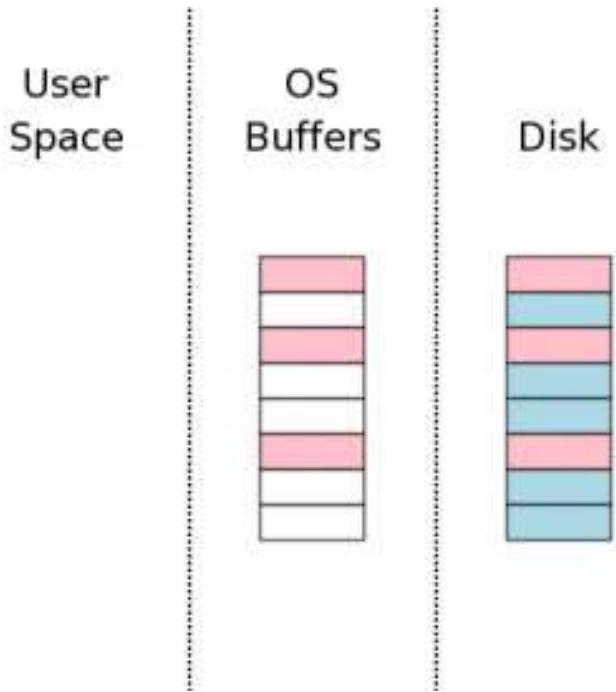
Удаление журнала отката



- ▶ Commit
- ▶ Если до этого момента происходит сбой, то транзакция откатывается
- ▶ Дорогая операция, поэтому иногда заменяется усечением или обнулением части (заголовка)



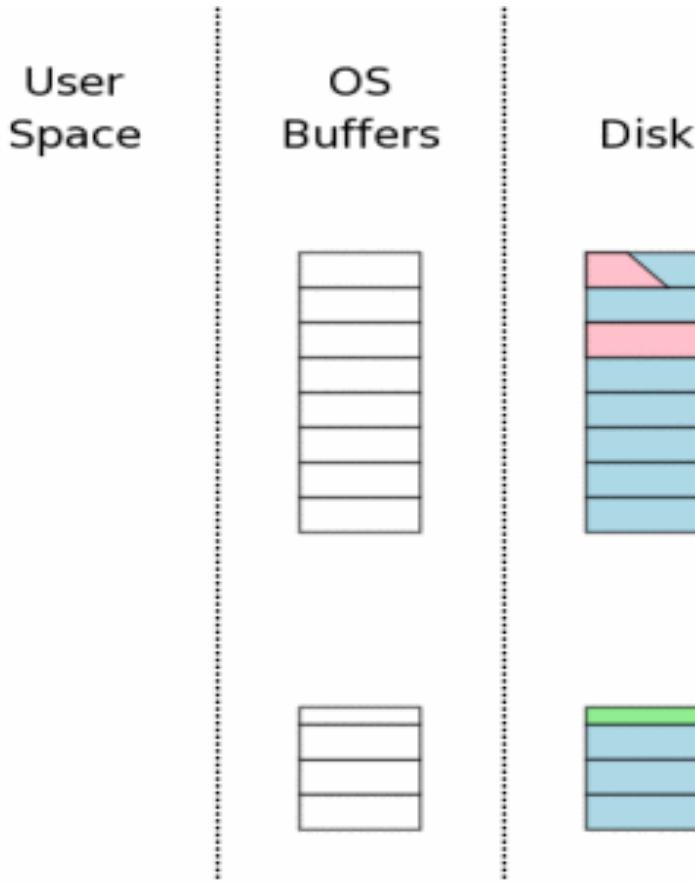
Снятие блокировок



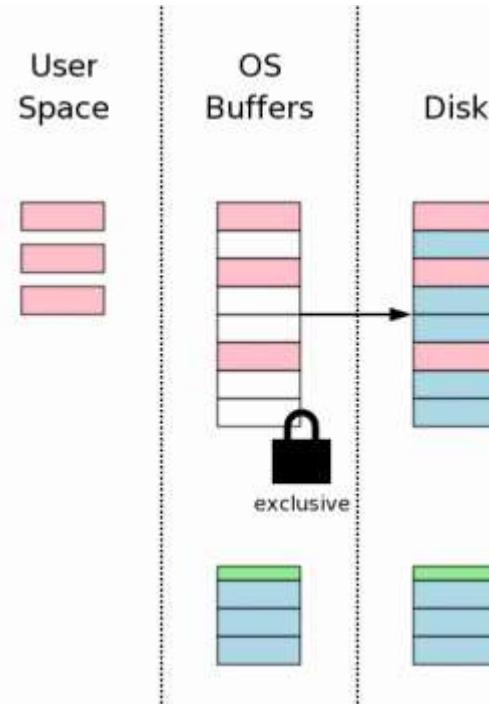
- ▶ Снятие монопольной блокировки, чтобы другие процессы могли произвести чтение
- ▶ Пользовательские данные остаются
- ▶ Счетчик изменений базы данных на 1 странице



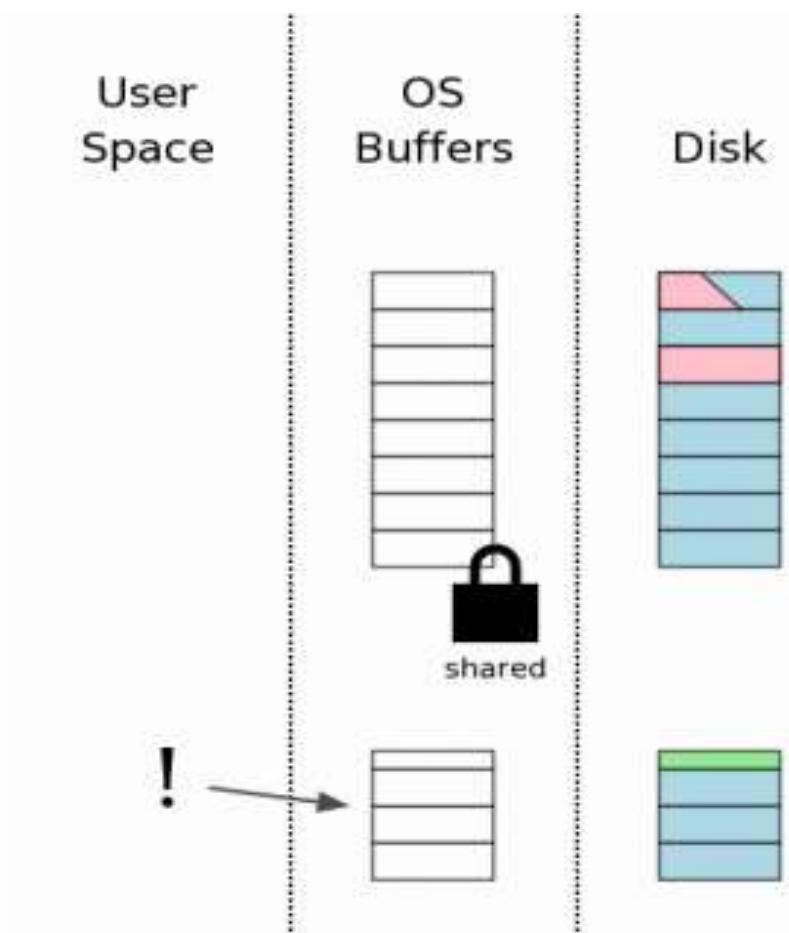
Откат



▶ Если во время записи из буфера на диск произошел сбой



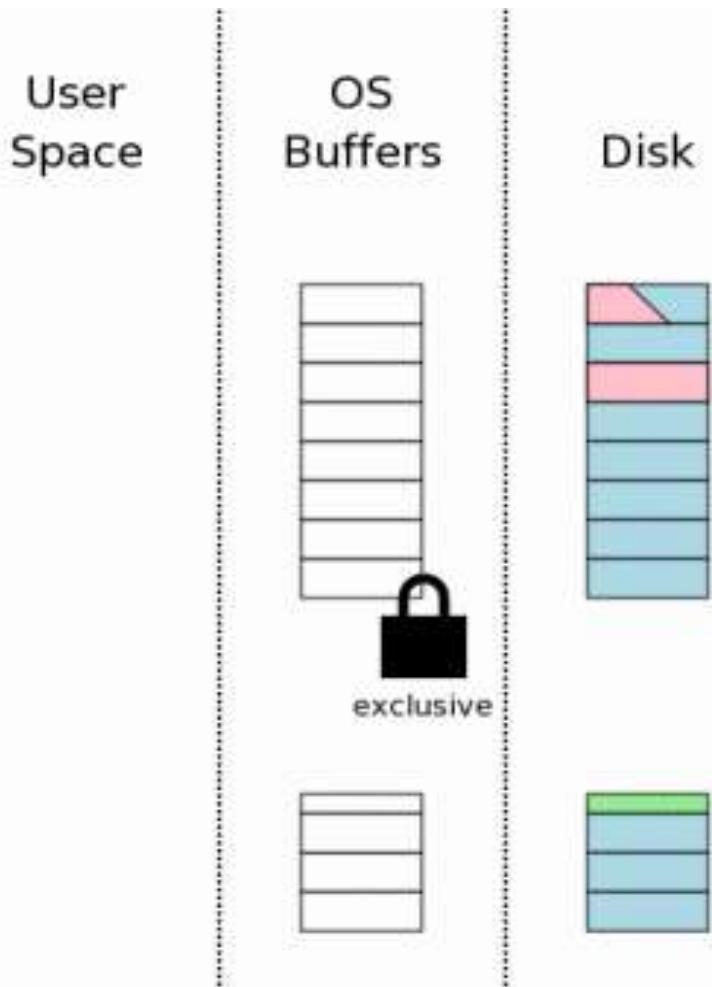
Откат



- ▶ Проверка – горячий журнал?
- ▶ существует;
- ▶ не пуст;
- ▶ в файле базы данных нет зарезервированной блокировки;
- ▶ верный заголовок.

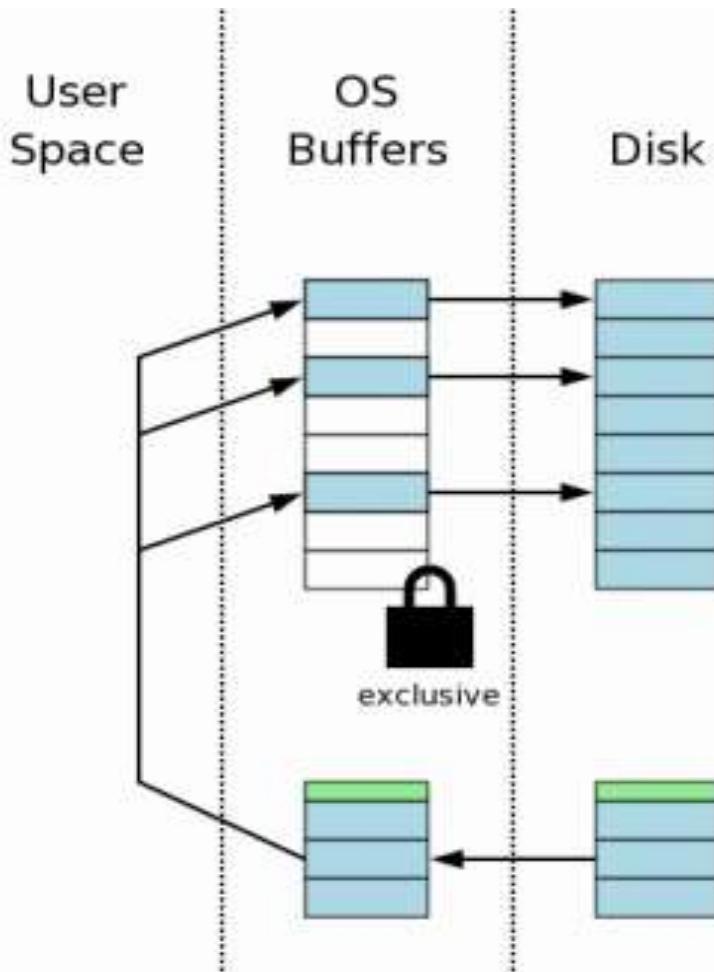


Откат



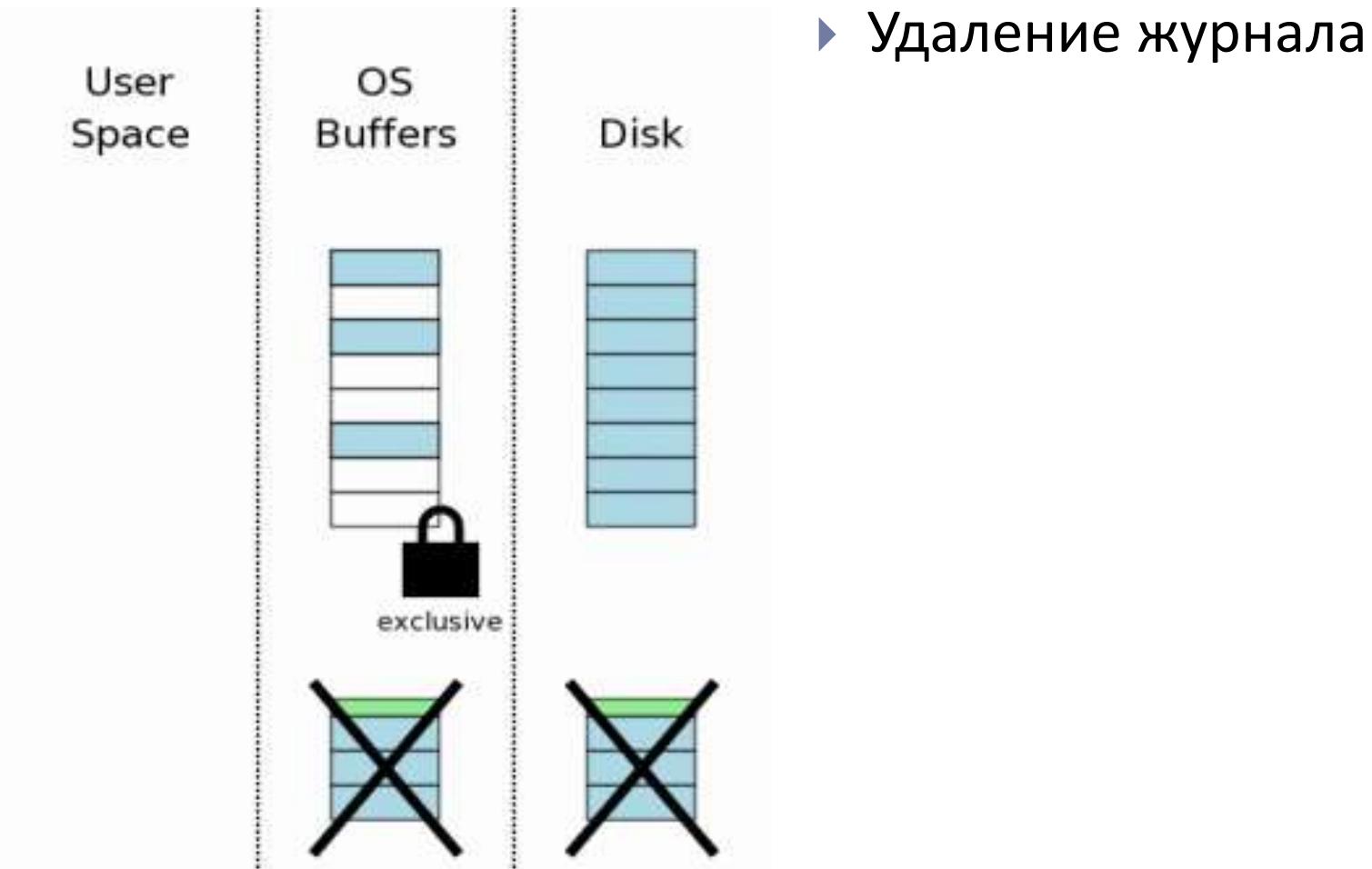
- ▶ Получение монопольной блокировки, чтобы только один процесс производил откат

Откат

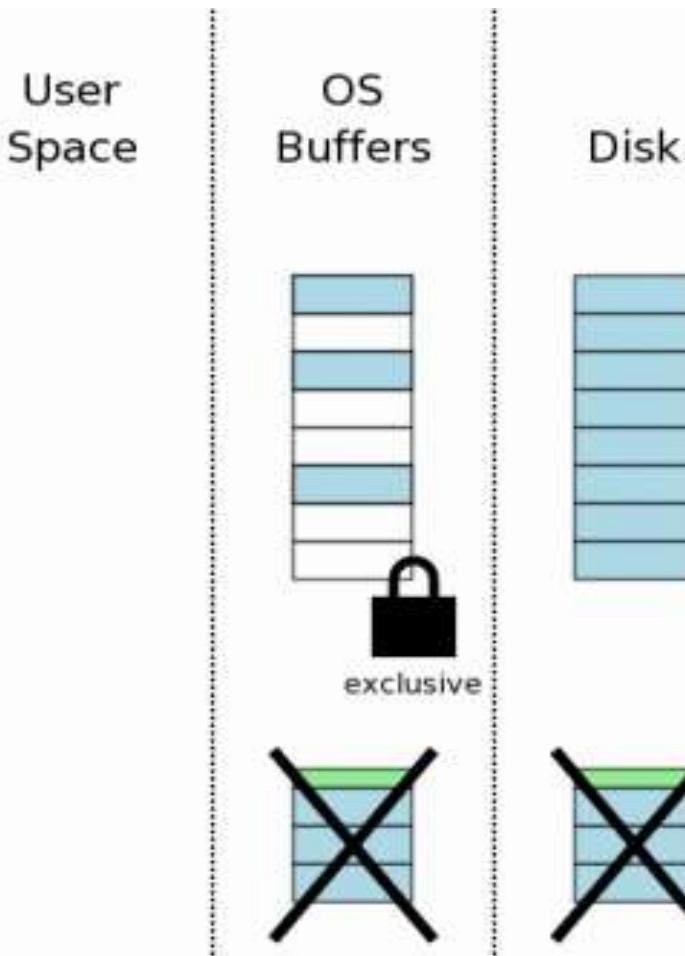


▶ Считывание и запись
данных из журнала в
буфер и в файл

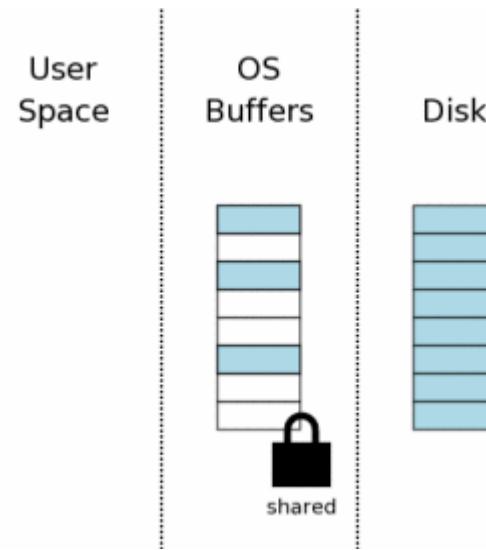
Откат



Откат



- ▶ Удаление журнала
- ▶ Состояние возвращается к начальному



Транзакция

- ▶ Группа операций, которая может быть выполнена или не выполнена вместе
 - ▶ TCL: COMMIT, ROLLBACK
 - ▶ Наличие автокоммита?
 - ▶ ACID: Atomicity, Consistency, Isolation, Durability
 - ▶ Точка сохранения – контрольная точка – SAVEPOINT
-
- ▶ Фиксация и откат внешних и внутренних транзакций
 - ▶ Автономные транзакции



Транзакция - ACID

- ▶ Изолированность – свойство независимости результата выполнения транзакций от параллельно работающих транзакций
 - ▶ READ UNCOMITED
 - ▶ READ COMMITTED
 - ▶ REPEATABLE READ
 - ▶ SERIALIZABLE



Транзакция – блокировки

- ▶ Блокировки – механизм обеспечения согласованности данных в случае одновременного обращения к данным нескольких пользователей
 - ▶ Разделяемая (shared lock)
 - ▶ Монопольная (exclusive lock)
 - ▶ Обновления (update lock)
- ▶ Гранулярность блокировки определяет, какой объект блокируется
- ▶ Процесс преобразования большого числа блокировок уровня строки, страницы или индекса в одну блокировку уровня таблицы называется эскалацией блокировок



Транзакция – блокировки

Разделяемая блокировка

Резервирует ресурс только для чтения
Может быть несколько разделяемых блокировок

Монопольная блокировка

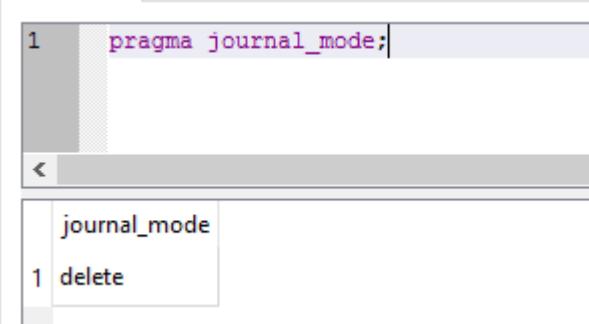
Резервирует страницу или строку для монопольного
использования одной транзакции
Применяется при INSERT, UPDATE и DELETE

Блокировка обновления

При COMMIT транзакции обновления, блокировка
обновления преобразовывается в монопольную
блокировку
Можно устанавливать на объекты с разделяемой
блокировкой
Нельзя устанавливать при наличии на нем другой
блокировки обновления или монопольной
блокировки
Может быть только одна блокировка обновления



Архитектура разделения одновременного доступа



```
1 pragma journal_mode;
```

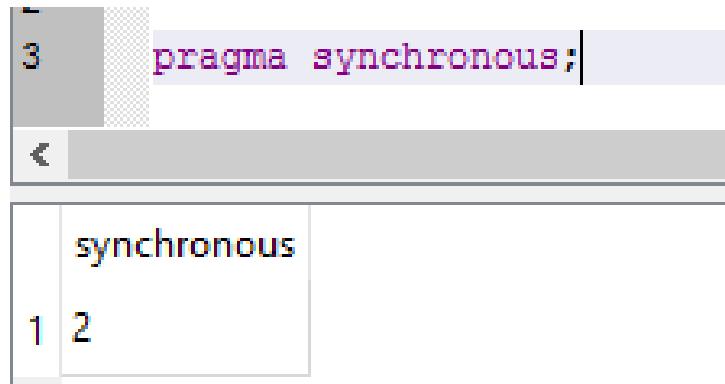
```
journal_mode
1 delete
```

A screenshot of a terminal window. The top part shows the command 'pragma journal_mode;'. Below it, the output 'journal_mode' is shown, followed by the number '1' and the word 'delete'.

- ▶ DELETE – файл журнала удаляется
- ▶ TRUNCATE - файл журнала обрезается
- ▶ PERSIST - размер остается, содержание заменяется нулями
- ▶ MEMORY - журнал ведется в памяти
- ▶ OFF – журнал отключен
- ▶ WAL – другой способ записи в журнал



PRAGMA SYNCHRONOUS



A screenshot of a code editor window. The main area shows the line of code "pragma synchronous;". To the left of the code, there is a vertical column with line numbers: 3 at the top, followed by a blank line, then 1 and 2. Below the code editor, there is a small preview pane showing the word "synchronous" in a blue font.

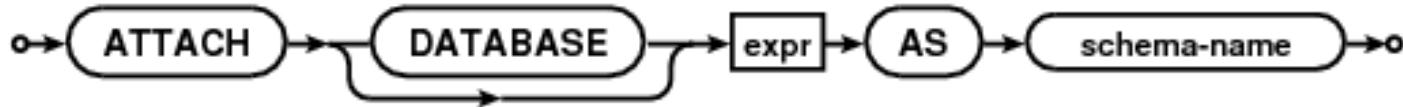
0 / OFF - данные считаются записанными, как только переданы ОС

1 / NORMAL (или 1) - компромиссный режим
2 / FULL - гарантирует целостность

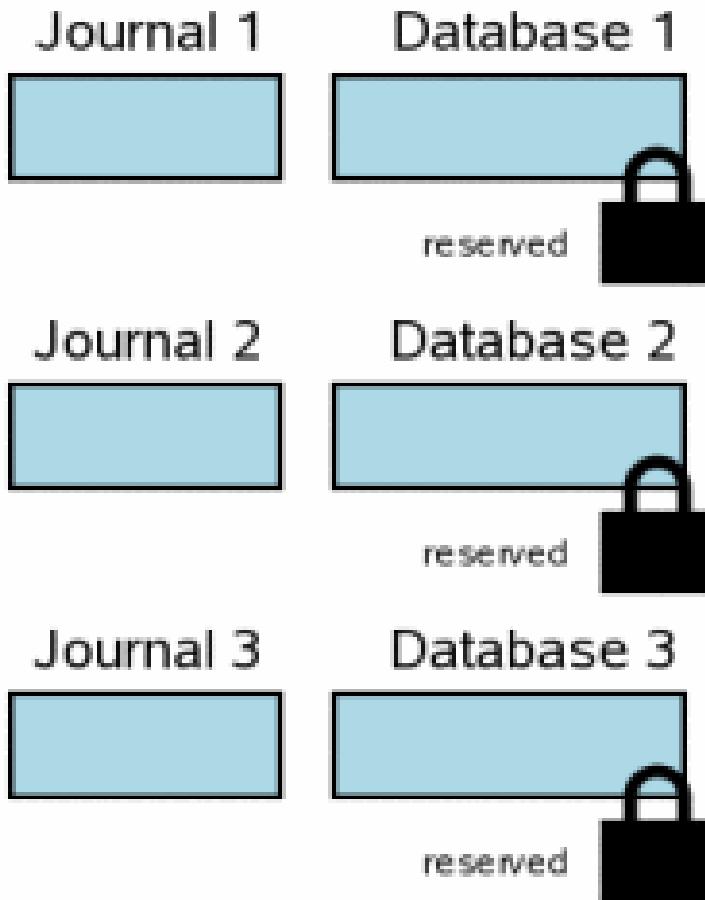
- ▶ предполагается, что вначале увеличивается размер файла, а затем добавляются данные
- ▶ фиксируется количество страниц в заголовке журнала, вначале 0
- ▶ буфер сбрасывается на диск
- ▶ записывается количество страниц в журнал

Транзакции в нескольких базах данных

- ▶ Можно установить соединение с несколькими базами данных
- ▶ В одной транзакции либо все файлы обновляются, либо не один не обновляется
- ▶ ATTACH DATABASE



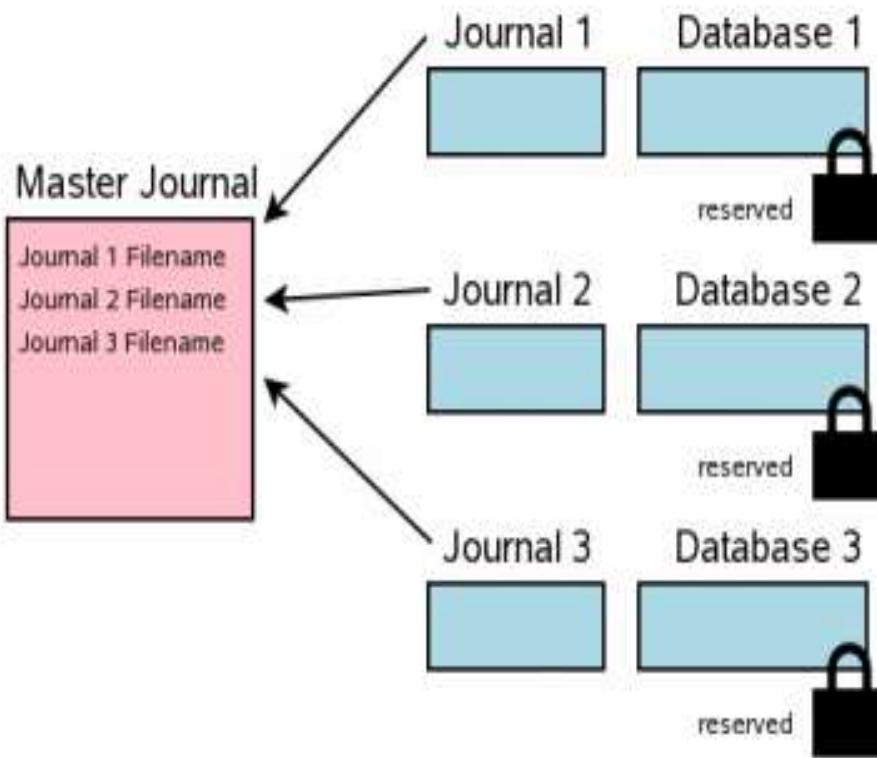
Транзакции в нескольких базах данных



- ▶ У каждой базы свой журнал
- ▶ Состояние перед началом транзакции

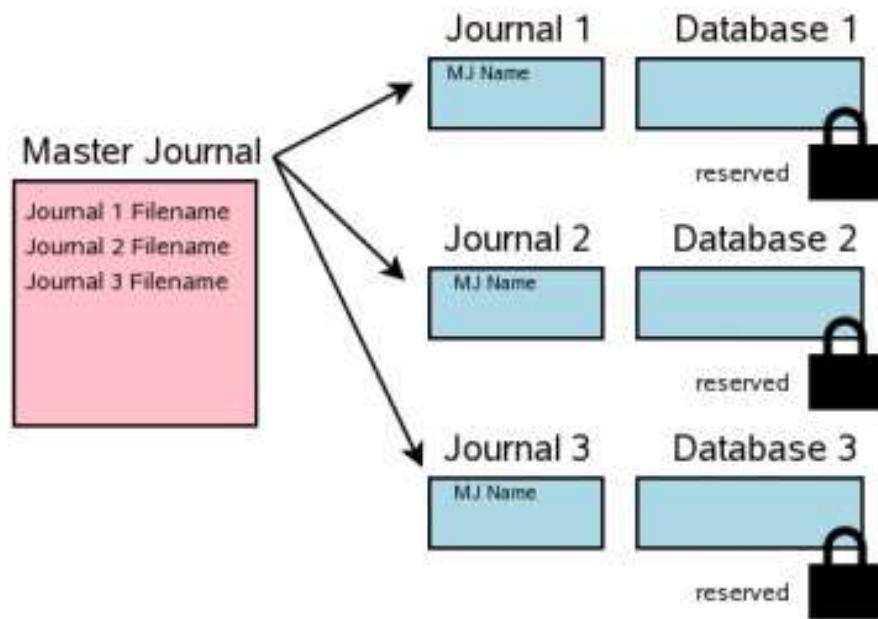


Транзакции в нескольких базах данных



- ▶ Создание главного журнала
- ▶ Имя совпадает с именем исходного файла
- ▶ Содержит полные пути к журналам
- ▶ Содержимое сбрасывается на диск, кроме случаев:
 - ▶ PRAGMA synchronous = OFF
 - ▶ PRAGMA journal_mode = MEMORY

Транзакции в нескольких базах данных

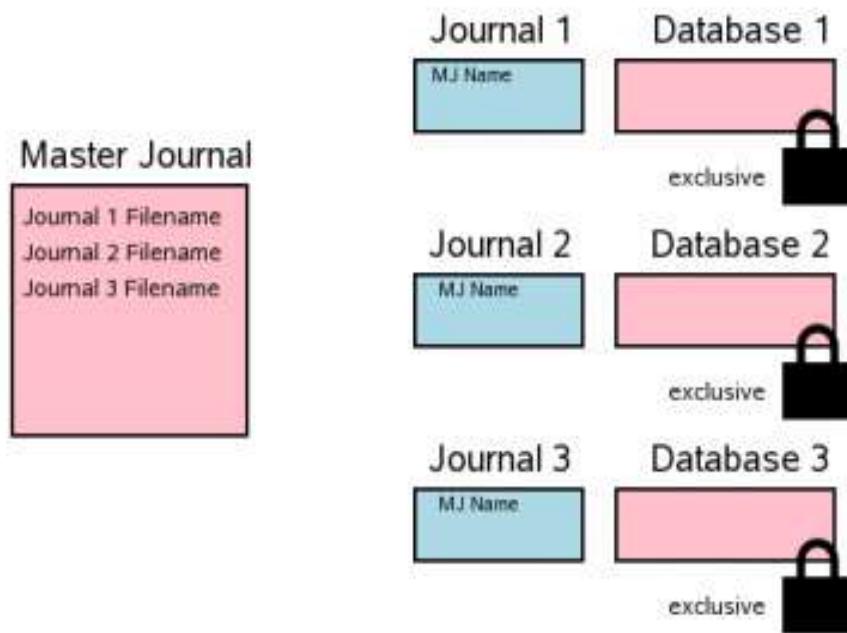


- ▶ Журналы сбрасывается на диск
- ▶ Обновление заголовков для журналов в буфере
- ▶ Журналы снова сбрасывается на диск

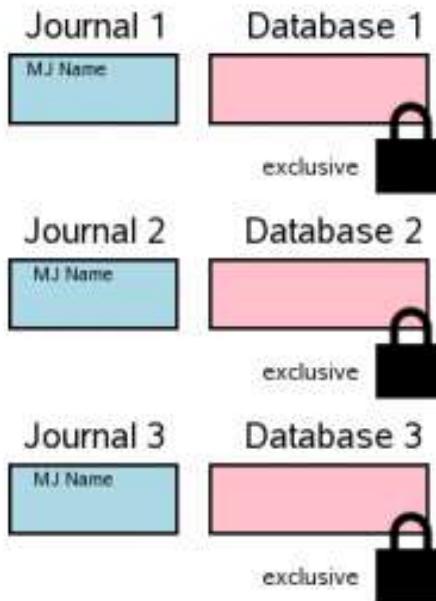
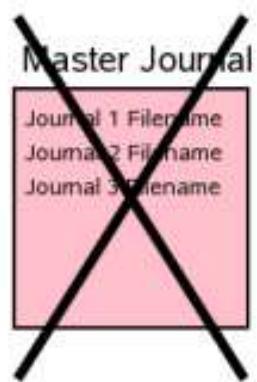


Транзакции в нескольких базах данных

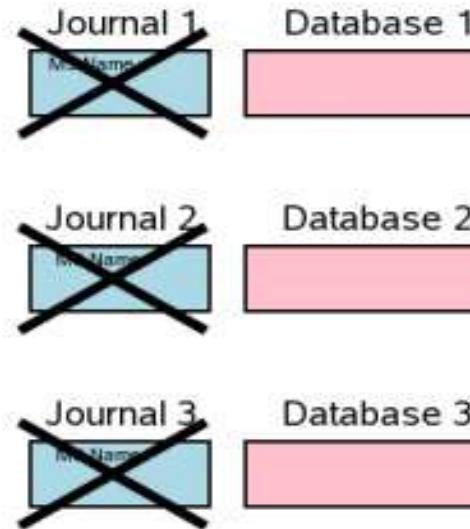
- ▶ Буферы файлов баз данных сбрасываются на диск



Транзакции в нескольких базах данных



- ▶ Удаляется главный журнал
- ▶ commit

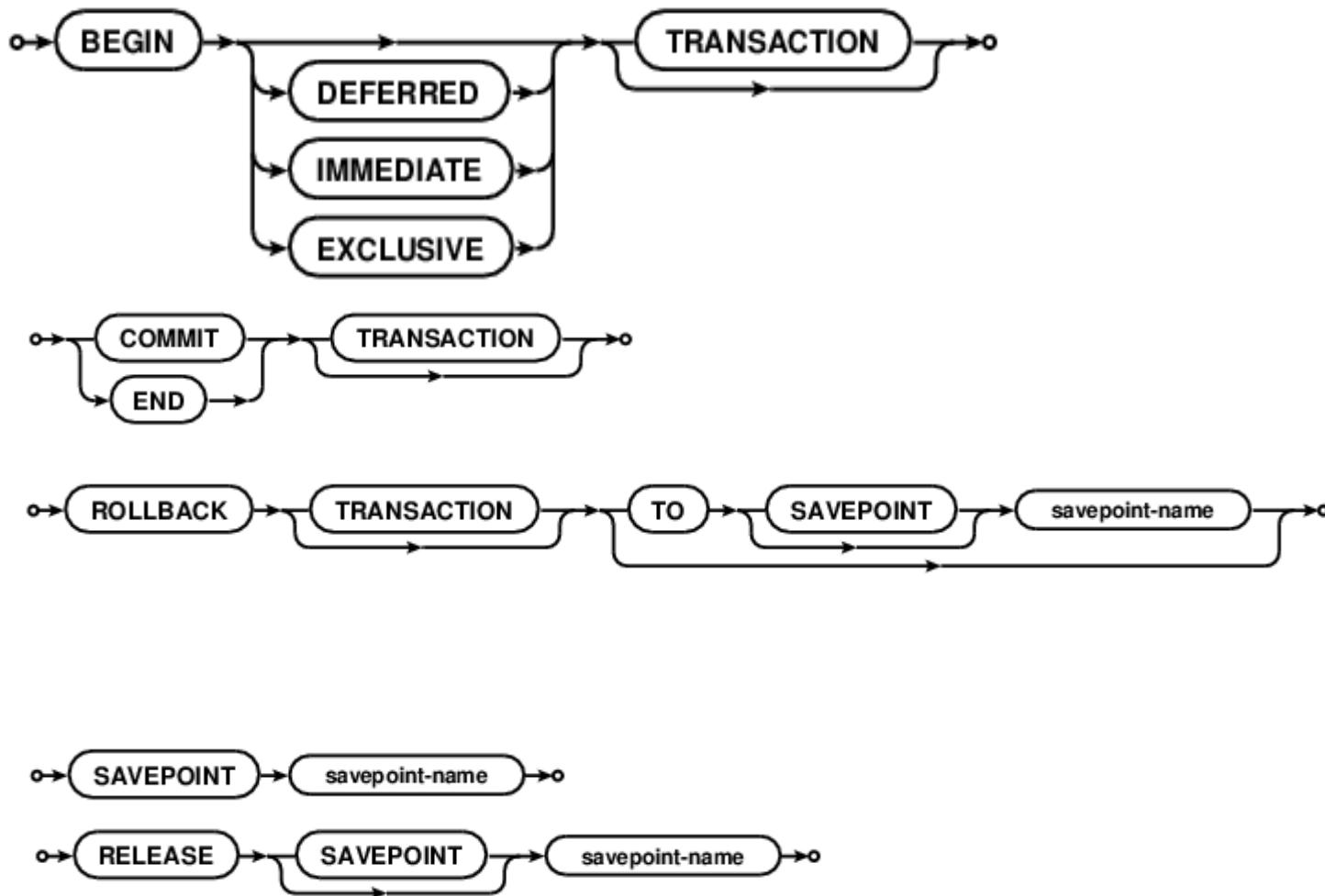


WAL – Write Ahead Log

- ▶ Изменения добавляются в отдельный файл WAL
- ▶ Исходные данные сохраняется в файле базы данных
- ▶ При COMMIT запись добавляется к WAL
- ▶ При CHECKPOINT данные переносятся в БД
 - ▶ Автоматически при 1000 страницах



Транзакция – SQLite



Выбор режима блокировки DEFERRED

- ▶ БД блокируется на запись при достижении первого оператора чтения или записи, который находится внутри тела транзакции
- ▶ БД будет доступна другим пользователям только для чтения
- ▶ DEFERRED (отложенный) является режимом по умолчанию



Выбор режима блокировки IMMEDIATE

- ▶ Выполнение блокировки БД после ключевого слова BEGIN
- ▶ БД будет доступна другим пользователям только для чтения



Выбор режима блокировки EXCLUSIVE

- ▶ Блокировка БД не только на запись, но и на чтение
- ▶ Блокировка происходит после ключевого слова BEGIN
- ▶ Во время транзакции – появляется файл журнала

| | | | | |
|--|----------------------|------------------|--------------------|-------|
| | SQLite_Db.db | 26.04.2023 16:23 | Data Base File | 36 КБ |
| | SQLite_Db.db-journal | 02.05.2023 16:33 | Файл "DB-JOURN..." | 0 КБ |

```
1 begin exclusive transaction;
2     insert into customers values (2125,'Brothers Chabb', 107, 40000);
3     update ORDERS set cust = 2125 where order_num = 113055;
4     commit;
5
6
```

- ▶ После транзакции – остается только файл БД

| | | | | |
|--|--------------|------------------|----------------|-------|
| | SQLite_Db.db | 02.05.2023 17:05 | Data Base File | 36 КБ |
|--|--------------|------------------|----------------|-------|

Транзакция – SQLite

- ▶ Каждый оператор идет в неявной транзакции с автокоммитом
- ▶ Можно транзакцию задать явно
- ▶ Разница между 127 вставками данных в таблицу в неявных транзакциях и в одной явной транзакции:

```
Запрос успешно выполнен: commit; (заняло 106мс)
```

```
Запрос успешно выполнен: insert into Table3 values ('Tran1', 1000.001); (заняло 155мс, 1 строк изменено)
```

Транзакция

- ▶ Транзакция закончится ROLLBACK, если база данных закрылась
- ▶ Транзакция закончится ROLLBACK, если возникает ошибка и указан алгоритм разрешения конфликтов ROLLBACK
- ▶ END TRANSACTION - COMMIT



Транзакция

- ▶ Механизма вложенных транзакций нет
- ▶ BEGIN TRANSACTION / COMMIT

| | | |
|--------|------------|------|
| 113045 | 2008-02-02 | 2112 |
| 113055 | 2008-02-15 | 2125 |

```
1 begin transaction A;
2   update ORDERS set cust = 2124 where order_num = 113055;
3   begin transaction B;
4     update ORDERS set cust = 2125 where order_num = 113045;
5     commit transaction B;
6   commit transaction A;
```

```
cannot start a transaction within a transaction:
begin transaction B;
```

Транзакция

- ▶ BEGIN TRANSACTION / COMMIT
- ▶ Контрольные точки SAVEPOINT
- ▶ Фиксация точки – RELEASE

| | | |
|--------|------------|------|
| 113045 | 2008-02-02 | 2112 |
| 113055 | 2008-02-15 | 2125 |

```
1 begin transaction A;
2     update ORDERS set cust = 2124 where order_num = 113055;
3     savepoint A;
4     update ORDERS set cust = 2125 where order_num = 113045;
5     release savepoint A;
6     commit transaction A;
```

| | | | |
|----|--------|------------|------|
| 9 | 113045 | 2008-02-02 | 2125 |
| 10 | 113055 | 2008-02-15 | 2124 |



Транзакция

- ▶ Откат транзакции на контрольную точку – ROLLBACK TO SAVEPOINT

| | | | |
|----|--------|------------|------|
| 9 | 113045 | 2008-02-02 | 2125 |
| 10 | 113055 | 2008-02-15 | 2124 |

```
1 begin transaction A;
2     update ORDERS set cust = 2103 where order_num = 113055;
3     savepoint A;
4     update ORDERS set cust = 2112 where order_num = 113045;
5     rollback to savepoint A;
6     commit transaction A;
```

| | | | |
|----|--------|------------|------|
| 9 | 113045 | 2008-02-02 | 2125 |
| 10 | 113055 | 2008-02-15 | 2103 |

Транзакция

- ▶ Откат транзакции – ROLLBACK
- ▶ При откате транзакции контрольные точки не фиксируются

| | | | |
|----|--------|------------|------|
| 9 | 113045 | 2008-02-02 | 2125 |
| 10 | 113055 | 2008-02-15 | 2103 |

```
1 begin transaction A;
2     update ORDERS set cust = 2125 where order_num = 113055;
3     savepoint A;
4     update ORDERS set cust = 2124 where order_num = 113045;
5     release savepoint A;
6     rollback transaction A;
7
```

| | | | |
|----|--------|------------|------|
| 9 | 113045 | 2008-02-02 | 2125 |
| 10 | 113055 | 2008-02-15 | 2103 |



Вопросы?

