Базы данных
Лекция 5

Партиционирование. Хранимые процедуры. Триггеры

Андрей Каледин

В предыдущих сериях…

В предыдущих сериях…

- Создание таблиц в БД
- Получение и изменение данных
- Индексы
- Транзакции

План лекции

1. Партиционирование
2. Хранимые процедуры
3. Триггеры
4. Как все сущности связаны друг с другом?

Партиционирование

Вопрос

В БД хотим сохранять все транзакции пользователей. В начале каждого месяца хотим составлять отчет о тратах за предыдущий месяц.

Как будем делать?

Партиционирование

Партиционирование — разделение большой таблицы на части (партиции). Каждая партиция хранится отдельно, но при этом воспринимается как часть единой таблицы.

Партиционирование

Партиционирование — разделение большой таблицы на части (партиции). Каждая партиция хранится отдельно, но при этом воспринимается как часть единой таблицы.

Преимущества:

1. Ускорение запросов: не ищем по данным, которые точно не интересны
2. Оптимизация индексов: свой индекс на каждую партицию
3. Удалять можно целыми партициями разом

Партиционирование

Когда использовать:

- Таблицы на десятки ГБ
- Много запросов на фильтрацию по ключу партиционирования

Партиционирование

Когда использовать:

- Таблицы на десятки ГБ
- Много запросов на фильтрацию по ключу партиционирования

Когда не использовать:

- Маленькие таблицы
- Частый поиск сразу по всем партициям

Партиционирование: пример

Пример:

Храним транзакции пользователей в партициях. В каждой партиции транзакции за конкретный месяц

transactions_04_2025

Id	user_id	shop_id	amount	created_at
98765	63565	675321	113.43	10:42 28.**04**.2025
12431	63565	73353	199.99	09:42 09.**04**.2025
51243	526145	164142	51451.00	12:42 13.**04**.2025
…				

transactions_05_2025

Id	user_id	shop_id	amount	created_at
42415	526145	675321	4154.31	11:12 12.**05**.2025
86767	63565	73353	5141.12	08:22 06.**05**.2025
90868	526145	123423	432.12	16:52 01.**05**.2025
…				

Партиционирование: пример

Пример:

Храним транзакции пользователей в партициях. В каждой партиции транзакции за конкретный месяц

transactions_04_2025

Id	user_id	shop_id	amount	created_at
98765	63565	675321	113.43	10:42 28.**04**.2025
12431	63565	73353	199.99	09:42 09.**04**.2025
51243	526145	164142	51451.00	12:42 13.**04**.2025
…				

```
CREATE TABLE transactions (
    id SERIAL,
    user_id INT NOT NULL,
    shop_id INT NOT NULL,
    amount NUMERIC(18, 2) NOT NULL,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),

    PRIMARY KEY (id, created_at)
) PARTITION BY RANGE (created_at);

CREATE INDEX ON transactions (created_at);

CREATE TABLE transactions_2025_04 PARTITION OF transactions
    FOR VALUES FROM ('2025-04-01') TO ('2025-05-01');

CREATE TABLE transactions_2025_05 PARTITION OF transactions
    FOR VALUES FROM ('2025-05-01') TO ('2025-06-01');
```

Партиционирование: виды

Вид	Когда использовать?	Запрос на создание

Range (Диапазонное)	Деление по датам / упорядоченным ID	`CREATE TABLE transactions (`
		`  ...`
		`) PARTITION BY RANGE (transaction_date);`
		`CREATE TABLE transactions_2023_01 PARTITION OF transactions`
		`  FOR VALUES FROM ('2023-01-01') TO ('2023-02-01');`

Партиционирование: виды

Вид	Когда использовать?	Запрос на создание

Range (Диапазонное)	Деление по датам / упорядоченным ID	`CREATE TABLE transactions (`
		`  ...`
		`) PARTITION BY RANGE (transaction_date);`
		`CREATE TABLE transactions_2023_01 PARTITION OF transactions`
		`  FOR VALUES FROM ('2023-01-01') TO ('2023-02-01');`

List (Списковое)	Деление по категориям	`CREATE TABLE users (`
		`  ...`
		`) PARTITION BY LIST (country);`
		`CREATE TABLE users_eu PARTITION OF users`
		`  FOR VALUES IN ('DE', 'FR', 'IT');`

Партиционирование: виды

Вид	Когда использовать?	Запрос на создание

Range (Диапазонное)	Деление по датам / упорядоченным ID	`CREATE TABLE transactions (`
		`  ...`
		`) PARTITION BY RANGE (transaction_date);`
		`CREATE TABLE transactions_2023_01 PARTITION OF transactions`
		`  FOR VALUES FROM ('2023-01-01') TO ('2023-02-01');`

List (Списковое)	Деление по категориям	`CREATE TABLE users (`
		`  ...`
		`) PARTITION BY LIST (country);`
		`CREATE TABLE users_eu PARTITION OF users`
		`  FOR VALUES IN ('DE', 'FR', 'IT');`

Hash (Хэшевое)	Равномерное распределение по партициям	`CREATE TABLE users (`
		`  ...`
		`) PARTITION BY HASH (id);`
		`CREATE TABLE users_0 PARTITION OF users`
		`  FOR VALUES WITH (MODULUS 4, REMAINDER 0);`

Партиционирование: виды

Вид	Когда использовать?	Запрос на создание

Range (Диапазонное)	Деление по датам / упорядоченным ID	`CREATE TABLE transactions (`
		`  ...`
		`) PARTITION BY RANGE (transaction_date);`
		`CREATE TABLE transactions_2023_01 PARTITION OF transactions`
		`  FOR VALUES FROM ('2023-01-01') TO ('2023-02-01');`

List (Списковое)	Деление по категориям	`CREATE TABLE users (`
		`  ...`
		`) PARTITION BY LIST (country);`
		`CREATE TABLE users_eu PARTITION OF users`
		`  FOR VALUES IN ('DE', 'FR', 'IT');`

Hash (Хэшевое)	Равномерное распределение по партициям	`CREATE TABLE users (`
		`  ...`
		`) PARTITION BY HASH (id);`
		`CREATE TABLE users_0 PARTITION OF users`
		`  FOR VALUES WITH (MODULUS 4, REMAINDER 0);`

Composite (Комбинированное)	Сочетание нескольких методов сразу	`CREATE TABLE users (`
		`  ...`
		`) PARTITION BY LIST (region);`
		`CREATE TABLE users_eu PARTITION OF users`
		`  FOR VALUES IN ('EU')`
		`  PARTITION BY RANGE (created_date);`

Хранимые процедуры

Хранимые процедуры

Хранимая процедура (Stored Procedure) – набор SQL-инструкций, который компилируется один раз и хранится на сервере БД.

Хранимые процедуры

Хранимая процедура (Stored Procedure) – набор SQL-инструкций, который компилируется один раз и хранится на сервере БД.

Характеристики:

1. Переиспользование кода
2. Входные и выходные параметры
3. Ограничение доступа
4. Транзакции внутри процедуры

Хранимые процедуры

```
CREATE OR REPLACE PROCEDURE UPDATE_USER_EMAIL(
    IN user_id INT,
    IN new_email VARCHAR
)
AS $$
BEGIN
    UPDATE users
    SET email = new_email
    WHERE id = user_id;

    IF NOT FOUND THEN
        RAISE EXCEPTION 'User with id % not found', user_id;
    END IF;
END;
$$ LANGUAGE plpgsql;


CALL UPDATE_USER_EMAIL(5, 'erikcartman@yandex.ru');
```

Хранимые процедуры

Плюсы	**Минусы**
Логика внутри БД вместо кода приложения	

Хранимые процедуры

Плюсы	**Минусы**
Логика внутри БД вместо кода приложения	
Производительность (компилируется и кэшируется в СУБД)	

Хранимые процедуры

Плюсы	**Минусы**
Логика внутри БД вместо кода приложения	
Производительность (компилируется и кэшируется в СУБД)	
Поддерживает условия, циклы, транзакции	

Хранимые процедуры

Плюсы	**Минусы**
Логика внутри БД вместо кода приложения	
Производительность (компилируется и кэшируется в СУБД)	Сложно дебажить
Поддерживает условия, циклы, транзакции	

Хранимые процедуры

Плюсы	**Минусы**
Логика внутри БД вместо кода приложения	
Производительность (компилируется и кэшируется в СУБД)	Сложно дебажить
Поддерживает условия, циклы, транзакции	Приколачиваете проект к конкретной СУБД

Хранимые процедуры: триггеры

Триггер (Trigger) – хранимая процедура, автоматически выполняющаяся при наступлении определенного события.

!!! Уменьшают прозрачность происходящего с данными, использовать с осторожностью !!!

Хранимые процедуры: триггеры

```
CREATE TABLE users_audit (
   user_id INT,
   old_email VARCHAR,
   new_email VARCHAR,
   changed_at TIMESTAMP DEFAULT NOW()
);
```

Хранимые процедуры: триггеры

```
CREATE TABLE users_audit (
   user_id INT,
   old_email VARCHAR,
   new_email VARCHAR,
   changed_at TIMESTAMP DEFAULT NOW()
);

CREATE OR REPLACE FUNCTION log_email_change()
RETURNS TRIGGER AS $$
BEGIN
   IF OLD.email <> NEW.email THEN
      INSERT INTO users_audit (user_id, old_email, new_email)
      VALUES (OLD.id, OLD.email, NEW.email);
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Хранимые процедуры: триггеры

```
CREATE TABLE users_audit (
   user_id INT,
   old_email VARCHAR,
   new_email VARCHAR,
   changed_at TIMESTAMP DEFAULT NOW()
);

CREATE OR REPLACE FUNCTION log_email_change()
RETURNS TRIGGER AS $$
BEGIN
   IF OLD.email <> NEW.email THEN
      INSERT INTO users_audit (user_id, old_email, new_email)
      VALUES (OLD.id, OLD.email, NEW.email);
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER users_email_audit
BEFORE UPDATE ON users
FOR EACH ROW
EXECUTE FUNCTION log_email_change();
```

Хранимые процедуры: триггеры

```
UPDATE users
SET email = 'spinner@gmail.com'
WHERE id = 5;


SELECT * FROM users_audit;
```

Хранимые процедуры: триггеры

```
UPDATE users
SET email = 'spinner@gmail.com'
WHERE id = 5;
```

```
SELECT * FROM users_audit;
```

	user_id	old_email	new_email	changed_at
1	5	erikcartman@yandex.ru	spinner@gmail.com	2025-04-13 22:09:30.243

Автоматическое создание партиций

С помощью триггеров и хранимых процедур можно настроить автоматическое создание партиций

Автоматическое создание партиций

С помощью триггеров и хранимых процедур можно настроить автоматическое создание партиций

```
CREATE OR REPLACE FUNCTION create_transaction_partition()
RETURNS TRIGGER AS $$
DECLARE
   name TEXT; partition_start DATE; partition_end DATE;
BEGIN
   name := format('transactions_%s', to_char(NEW.created_at, 'YYYY_MM'));

   partition_start := date_trunc('month', NEW.created_at);
   partition_end := partition_start + INTERVAL '1 month';

   IF NOT EXISTS ( SELECT 1 FROM pg_tables
      WHERE schemaname = 'public' AND tablename = name
   ) THEN
      EXECUTE format('CREATE TABLE %I PARTITION OF transactions ' ||
         'FOR VALUES FROM (%L) TO (%L)', name, partition_start, partition_end);
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Автоматическое создание партиций

С помощью триггеров и хранимых процедур можно настроить автоматическое создание партиций

```
CREATE OR REPLACE FUNCTION create_transaction_partition()
RETURNS TRIGGER AS $$
DECLARE
   name TEXT; partition_start DATE; partition_end DATE;
BEGIN
   name := format('transactions_%s', to_char(NEW.created_at, 'YYYY_MM'));

   partition_start := date_trunc('month', NEW.created_at);
   partition_end := partition_start + INTERVAL '1 month';

   IF NOT EXISTS ( SELECT 1 FROM pg_tables
      WHERE schemaname = 'public' AND tablename = name
   ) THEN
      EXECUTE format('CREATE TABLE %I PARTITION OF transactions ' ||
         'FOR VALUES FROM (%L) TO (%L)', name, partition_start, partition_end);
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trg_create_transaction_partition
BEFORE INSERT ON transactions
FOR EACH ROW EXECUTE FUNCTION create_transaction_partition();
```

Автоматическое создание партиций

С помощью триггеров и хранимых процедур можно настроить автоматическое создание партиций

```
CREATE OR REPLACE FUNCTION create_transaction_partition()
RETURNS TRIGGER AS $$
DECLARE
   name TEXT; partition_start DATE; partition_end DATE;
BEGIN
   name := format('transactions_%s', to_char(NEW.created_at, 'YYYY_MM'));

   partition_start := date_trunc('month', NEW.created_at);
   partition_end := partition_start + INTERVAL '1 month';

   IF NOT EXISTS ( SELECT 1 FROM pg_tables
      WHERE schemaname = 'public' AND tablename = name
   ) THEN
      EXECUTE format('CREATE TABLE %I PARTITION OF transactions ' ||
         'FOR VALUES FROM (%L) TO (%L)', name, partition_start, partition_end);
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trg_create_transaction_partition
BEFORE INSERT ON transactions
FOR EACH ROW EXECUTE FUNCTION create_transaction_partition();
```

```
INSERT INTO transactions (created_at, amount)
VALUES ('2024-01-15', 100.00); --> Создаст transactions_2024_01

INSERT INTO transactions (created_at, amount)
VALUES ('2024-02-05', 200.00); --> Создаст transactions_2024_02
```

Что мы сегодня узнали?

1. Осознали концепцию партиций и подходы к работе с ними
2. Научились писать хранимые процедуры
3. Разобрались с триггерами

Все переплетено

Все переплетено

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

Все переплетено

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Все переплетено

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Все переплетено

Liquibase

Меняем структуру данных и сами данные в условиях командной разработки

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Все переплетено

Liquibase

Меняем структуру данных и сами данные в условиях командной разработки

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

Индексы

Оптимально манипулируем данными

B-tree, Bitmap, Hash, …

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Все переплетено

Liquibase

Меняем структуру данных и сами данные в условиях командной разработки

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

Индексы

Оптимально манипулируем данными

B-tree, Bitmap, Hash, …

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Партиции

Делим данные на части

Range, List, Hash

Все переплетено

Liquibase

Меняем структуру данных и сами данные в условиях командной разработки

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

Индексы

Оптимально манипулируем данными

B-tree, Bitmap, Hash, …

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Партиции

Делим данные на части

Range, List, Hash

Хранимки

Блоки с кодом внутри БД

Все переплетено

Liquibase

Меняем структуру данных и сами данные в условиях командной разработки

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

Индексы

Оптимально манипулируем данными

B-tree, Bitmap, Hash, …

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Партиции

Делим данные на части

Range, List, Hash

Триггеры

Автоматические обработчики событий

Хранимки

Блоки с кодом внутри БД

Все переплетено

Liquibase

Меняем структуру данных и сами данные в условиях командной разработки

1

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

Индексы

Оптимально манипулируем данными

B-tree, Bitmap, Hash, …

Связи между сущностями БД:

1. Через Liquibase создаем сущности в БД

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Партиции

Делим данные на части

Range, List, Hash

Триггеры

Автоматические обработчики событий

Хранимки

Блоки с кодом внутри БД

Все переплетено

Liquibase

Меняем структуру данных и сами данные в условиях командной разработки

1

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

2

Индексы

Оптимально манипулируем данными

B-tree, Bitmap, Hash, …

Связи между сущностями БД:

1. Через Liquibase создаем сущности в БД
2. DDL создает Индексы

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Партиции

Делим данные на части

Range, List, Hash

Триггеры

Автоматические обработчики событий

Хранимки

Блоки с кодом внутри БД

Все переплетено

Liquibase

Меняем структуру данных и сами данные в условиях командной разработки

1

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

2

Индексы

Оптимально манипулируем данными

B-tree, Bitmap, Hash, …

3

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Партиции

Делим данные на части

Range, List, Hash

Триггеры

Автоматические обработчики событий

Хранимки

Блоки с кодом внутри БД

Связи между сущностями БД:

1. Через Liquibase создаем сущности в БД
2. DDL создает Индексы
3. DDL создает Партиции

Все переплетено

Liquibase

Меняем структуру данных и сами данные в условиях командной разработки

1

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

2

Индексы

Оптимально манипулируем данными

B-tree, Bitmap, Hash, …

3

Партиции

Делим данные на части

Range, List, Hash

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

4

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Триггеры

Автоматические обработчики событий

Хранимки

Блоки с кодом внутри БД

Связи между сущностями БД:

1. Через Liquibase создаем сущности в БД
2. DDL создает Индексы
3. DDL создает Партиции
4. DML использует транзакции для обеспечения ACID

Все переплетено

Liquibase

Меняем структуру данных и сами данные в условиях командной разработки

1

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

2

Индексы

Оптимально манипулируем данными

B-tree, Bitmap, Hash, …

3

5

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

4

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Партиции

Делим данные на части

Range, List, Hash

Триггеры

Автоматические обработчики событий

Хранимки

Блоки с кодом внутри БД

Связи между сущностями БД:

1. Через Liquibase создаем сущности в БД
2. DDL создает Индексы
3. DDL создает Партиции
4. DML использует транзакции для обеспечения ACID
5. DML использует индексы для взаимодействия с данными

Все переплетено

Liquibase

Меняем структуру данных и сами данные в условиях командной разработки

1

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

2

Индексы

Оптимально манипулируем данными

B-tree, Bitmap, Hash, …

3

5

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

4

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

6

Партиции

Делим данные на части

Range, List, Hash

Триггеры

Автоматические обработчики событий

Хранимки

Блоки с кодом внутри БД

Связи между сущностями БД:

1. Через Liquibase создаем сущности в БД
2. DDL создает Индексы
3. DDL создает Партиции
4. DML использует транзакции для обеспечения ACID
5. DML использует индексы для взаимодействия с данными
6. Данные могут быть распределены между несколькими партициями

Все переплетено

Liquibase

Меняем структуру данных и сами данные в условиях командной разработки

DDL

Определяем структуру данных в БД

CREATE TABLE/INDEX/VIEW, ALTER TABLE

Индексы

Оптимально манипулируем данными

B-tree, Bitmap, Hash, …

TCL

Транзакционно работаем с данными

ACID, WAL, MVCC, Autovacuum

DML

Манипулируем данными

SELECT, INSERT, UPDATE, DELETE, JOIN

Партиции

Делим данные на части

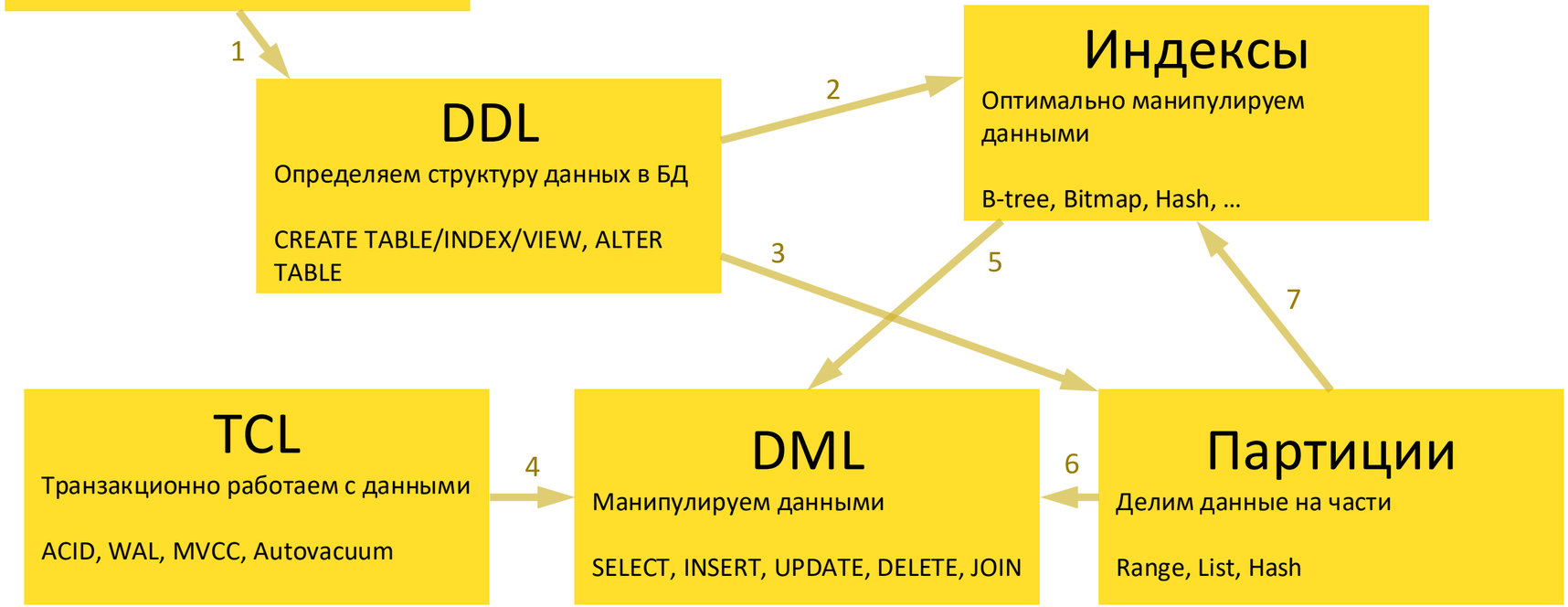
Range, List, Hash

Триггеры

Автоматические обработчики событий

Хранимки

Блоки с кодом внутри БД



Связи между сущностями БД:

1. Через Liquibase создаем сущности в БД
2. DDL создает Индексы
3. DDL создает Партиции
4. DML использует транзакции для обеспечения ACID
5. DML использует индексы для взаимодействия с данными
6. Данные могут быть распределены между несколькими партициями
7. Каждая партиция имеет свой индекс

Все переплетено

Связи между сущностями БД:

1. Через Liquibase создаем сущности в БД
2. DDL создает Индексы
3. DDL создает Партиции
4. DML использует транзакции для обеспечения ACID
5. DML использует индексы для взаимодействия с данными
6. Данные могут быть распределены между несколькими партициями
7. Каждая партиция имеет свой индекс
8. Хранимые процедуры могут менять данные

Все переплетено

Связи между сущностями БД:

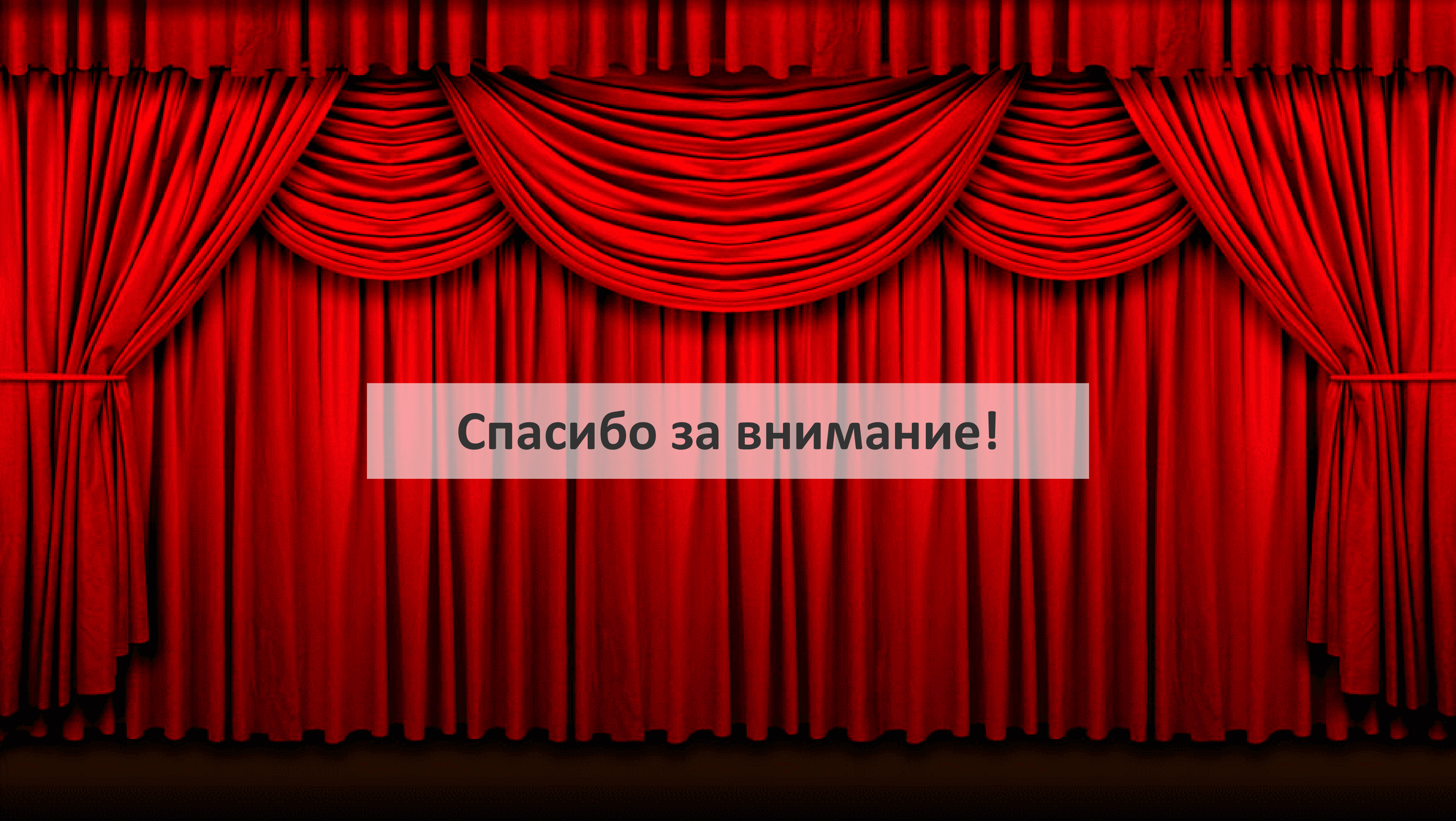
1. Через Liquibase создаем сущности в БД
2. DDL создает Индексы
3. DDL создает Партиции
4. DML использует транзакции для обеспечения ACID
5. DML использует индексы для взаимодействия с данными
6. Данные могут быть распределены между несколькими партициями
7. Каждая партиция имеет свой индекс
8. Хранимые процедуры могут менять данные
9. Хранимые процедуры могут создавать партиции

Все переплетено

Связи между сущностями БД:

1. Через Liquibase создаем сущности в БД
2. DDL создает Индексы
3. DDL создает Партиции
4. DML использует транзакции для обеспечения ACID
5. DML использует индексы для взаимодействия с данными
6. Данные могут быть распределены между несколькими партициями
7. Каждая партиция имеет свой индекс
8. Хранимые процедуры могут менять данные
9. Хранимые процедуры могут создавать партиции
10. Для запуска хранимых процедур используются триггеры

ВСЁ!


Спасибо за внимание!