Введение в реляционные базы данных

Лекция 9: Транзакции

Артем Толканев

November 29, 2024

Работа с СУБД

Система управления базой данных (СУБД) представляет собой программное обеспечение, которое управляет всем доступом к базе данных

Система управления базой данных (СУБД) представляет собой программное обеспечение, которое управляет всем доступом к базе данных

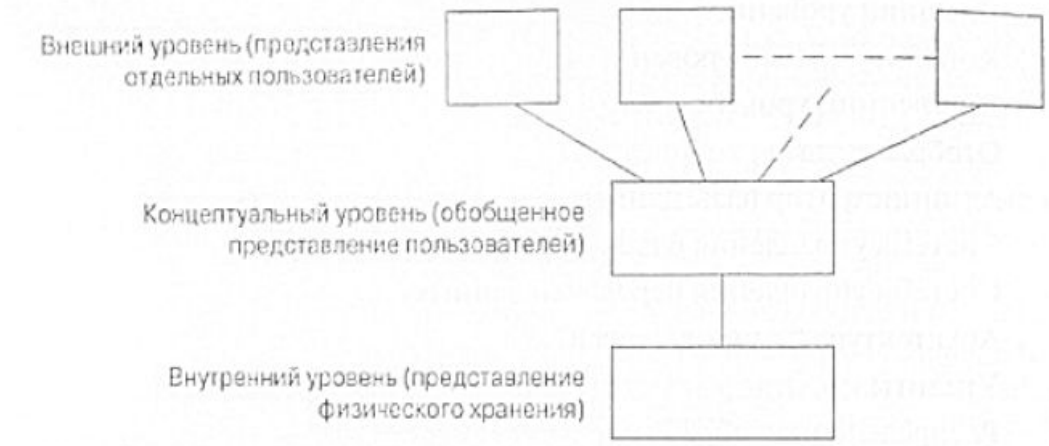
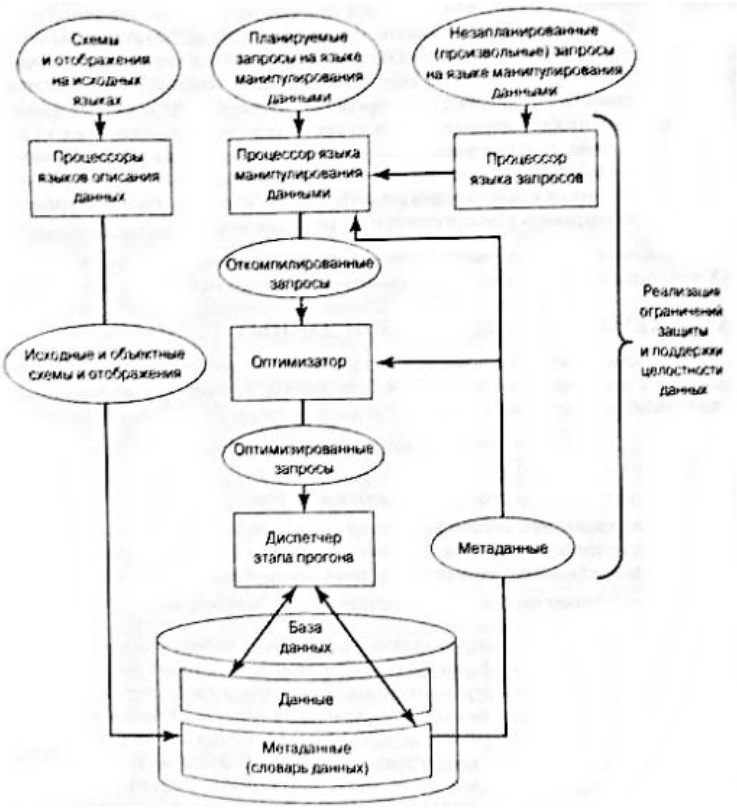


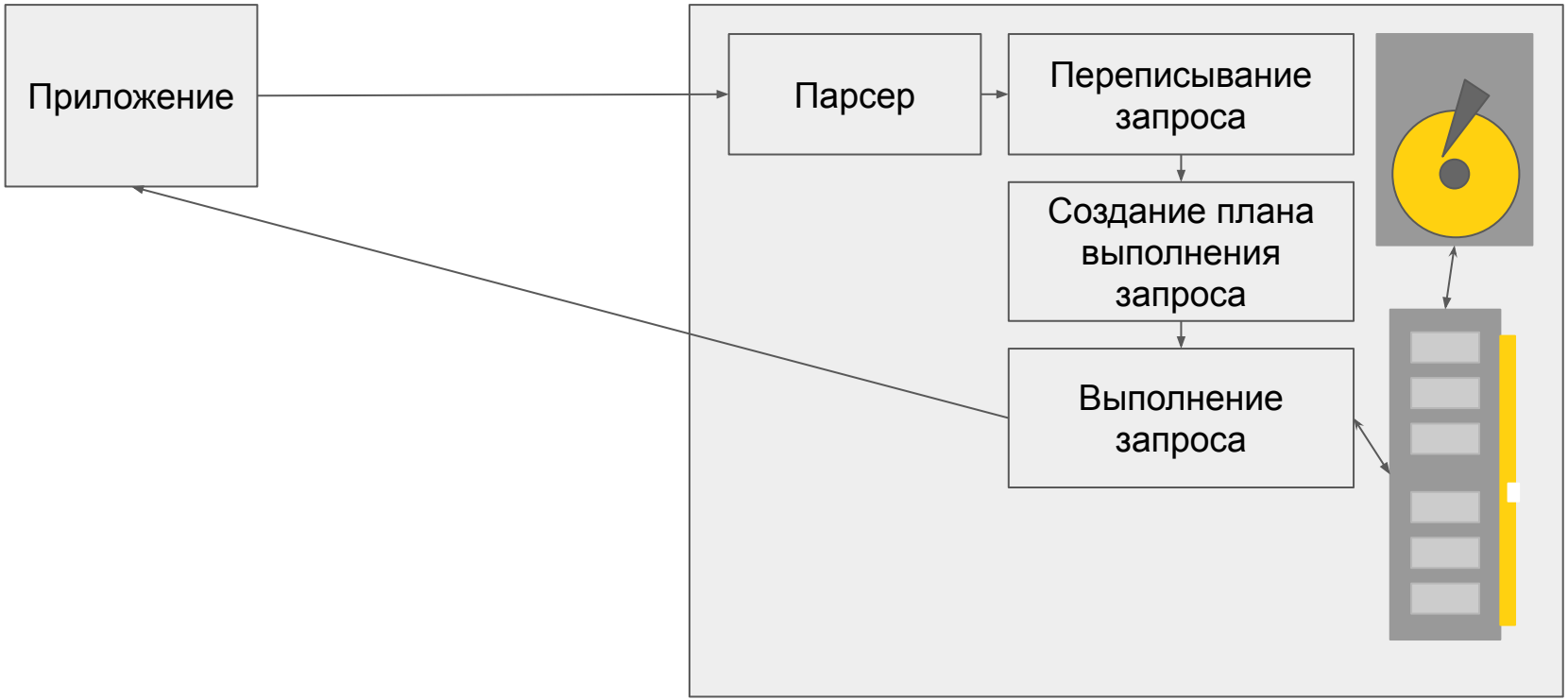
Рис. 2.1. Три уровня архитектуры ANSI/SPARC

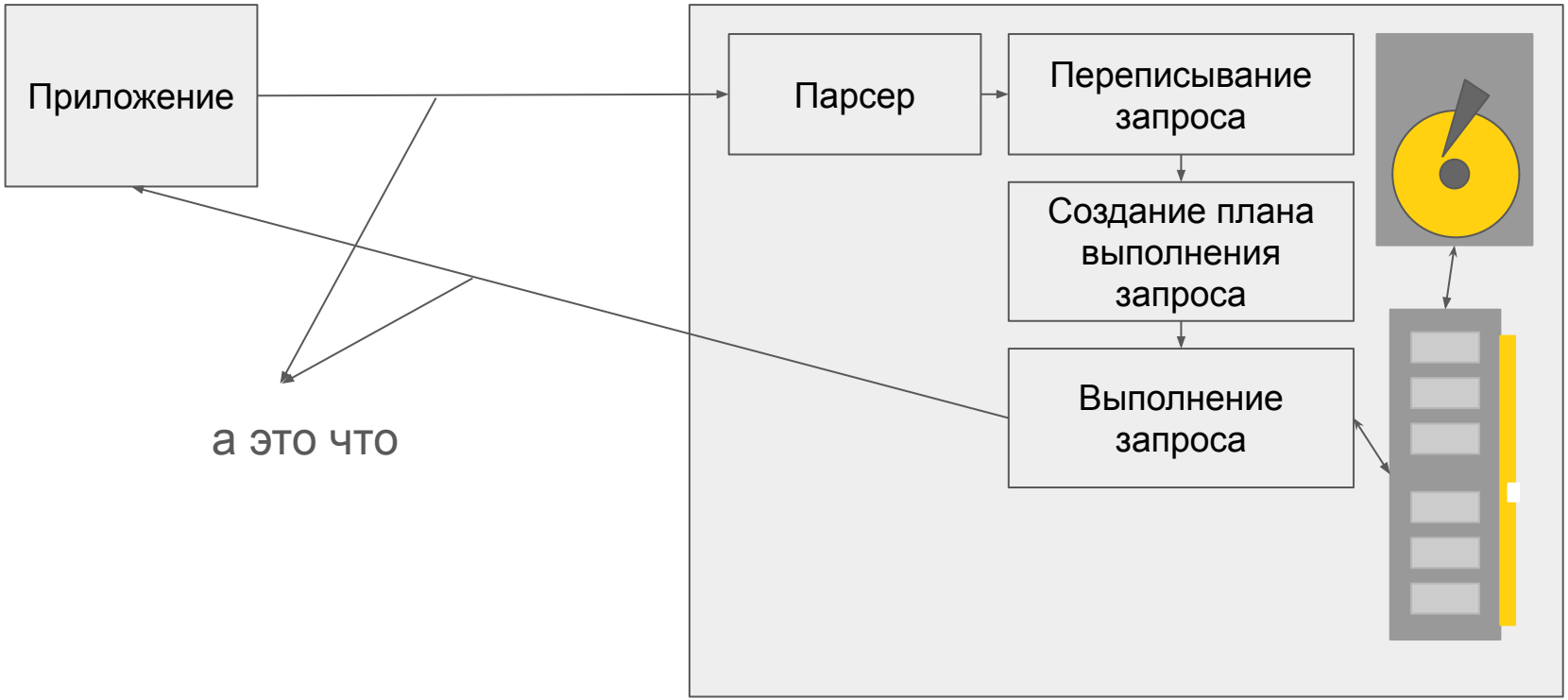
Система управления базой данны

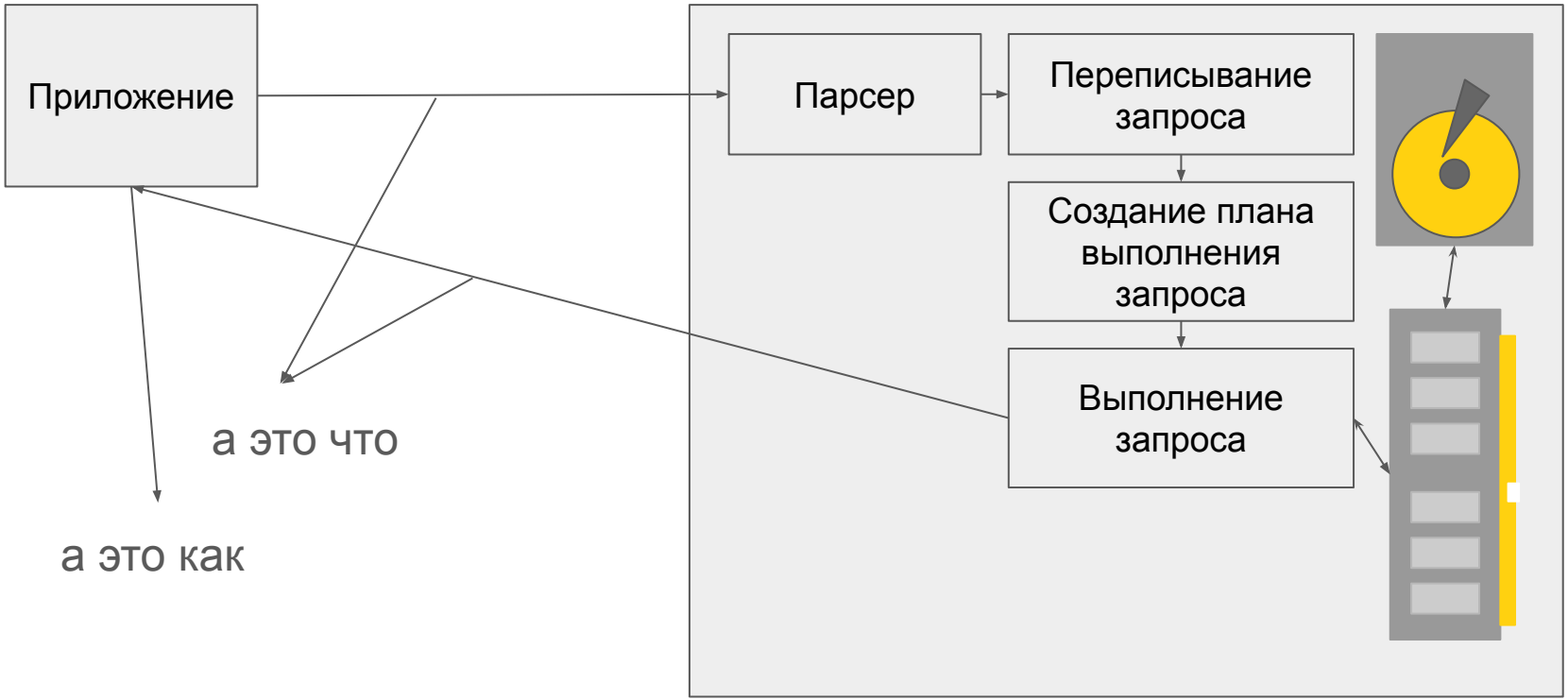
программное обеспечение, которое ых

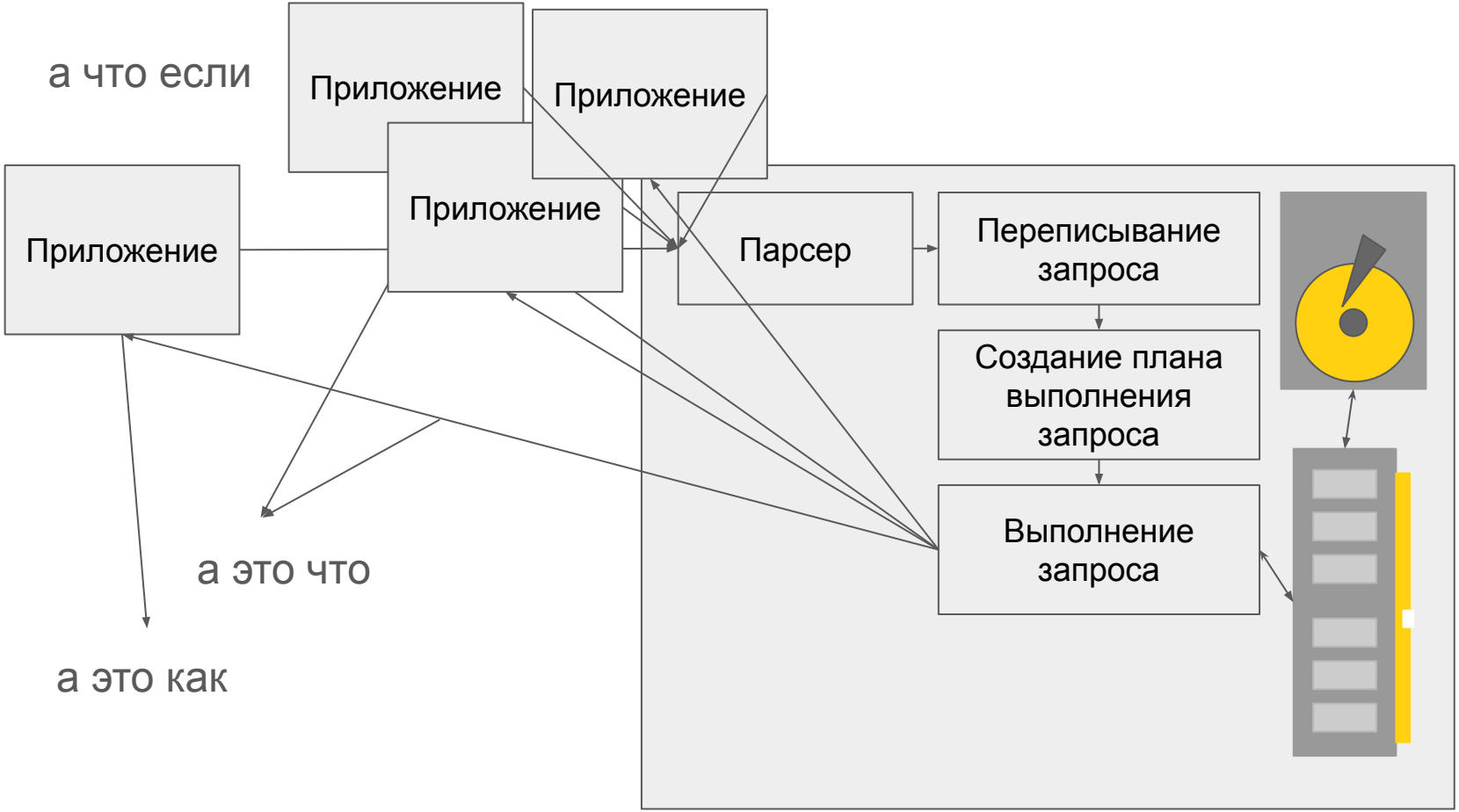
Рис. 2.1. Три уровня архитектуры ANSI/SPARC

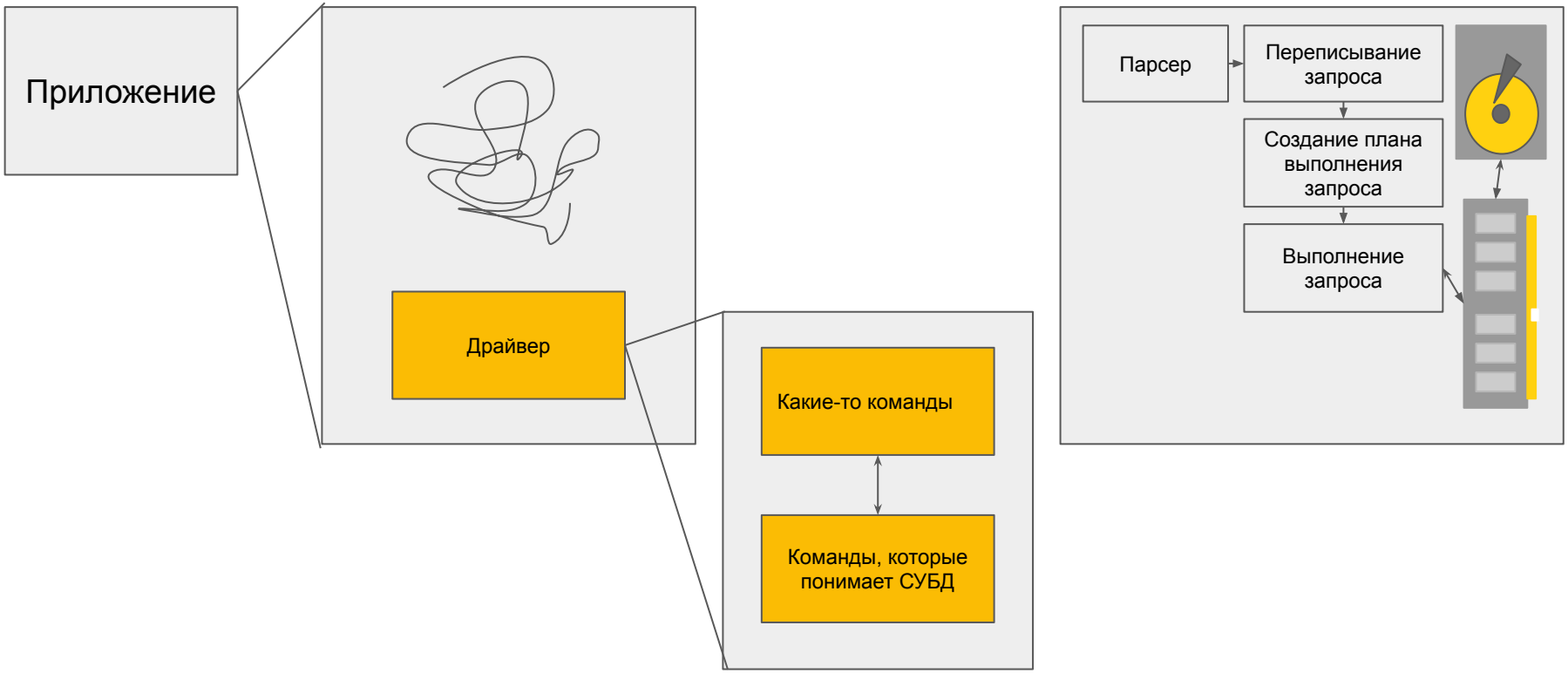


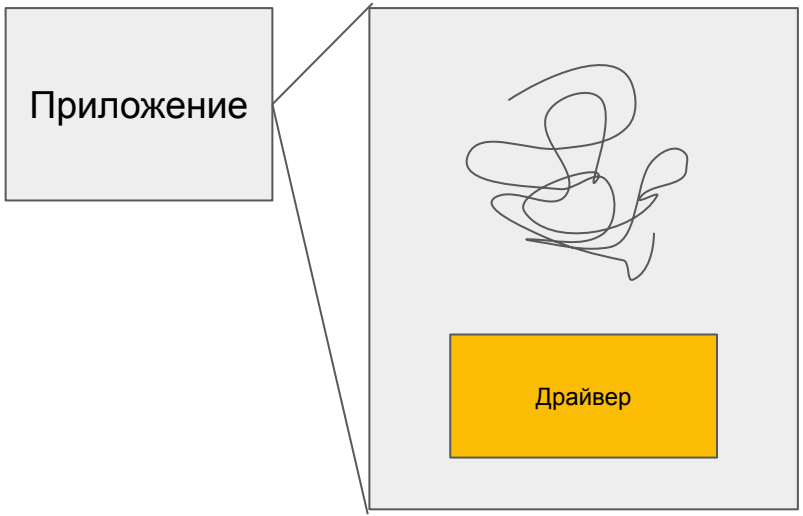

Приложение
Парсер
Переписывание запроса
Создание плана выполнения запроса
Выполнение запроса


Приложение
Парсер
Переписывание запроса
Создание плана выполнения запроса
Выполнение запроса
а это что

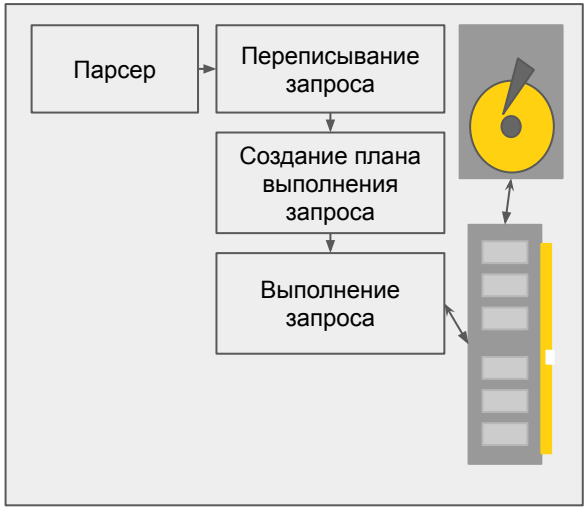

Приложение
Парсер
Переписывание запроса
Создание плана выполнения запроса
Выполнение запроса
а это что
а это как

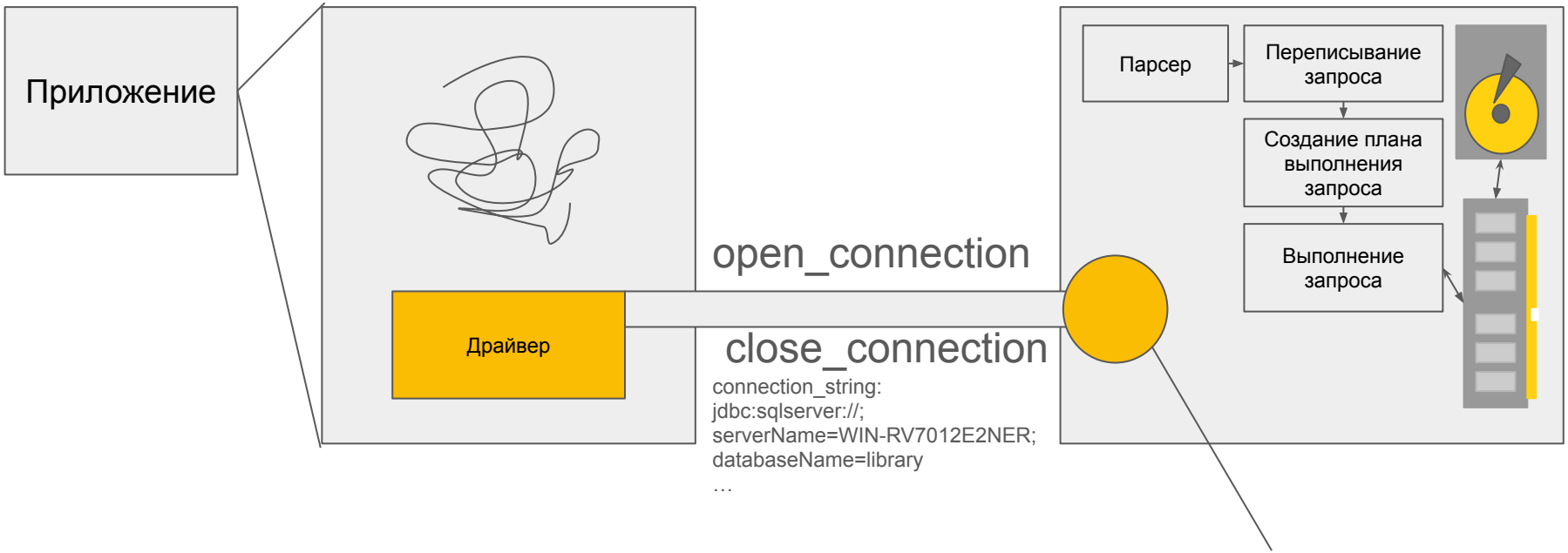

а что если
Приложение
Приложение
Приложение
Приложение
Парсер
Переписывание запроса
Создание плана выполнения запроса
Выполнение запроса
а это что
а это как


Приложение
Драйвер
Какие-то команды
Команды, которые понимает СУБД
Парсер
Переписывание запроса
Создание плана выполнения запроса
Выполнение запроса

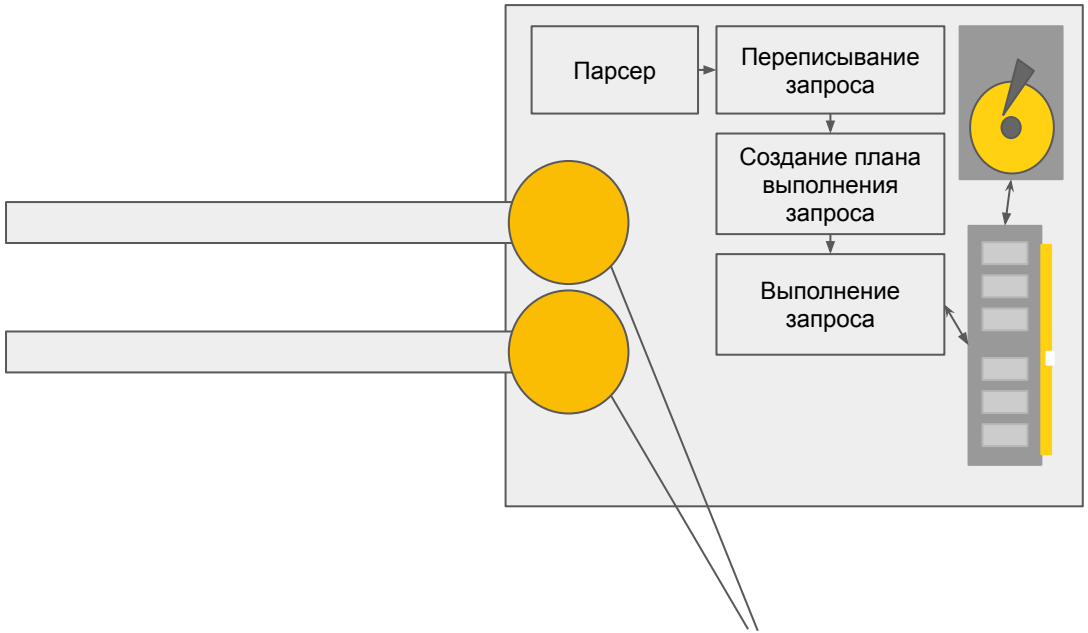


connection_string:
jdbc:sqlserver://;
serverName=WIN-RV7012E2NER;
databaseName=library
…

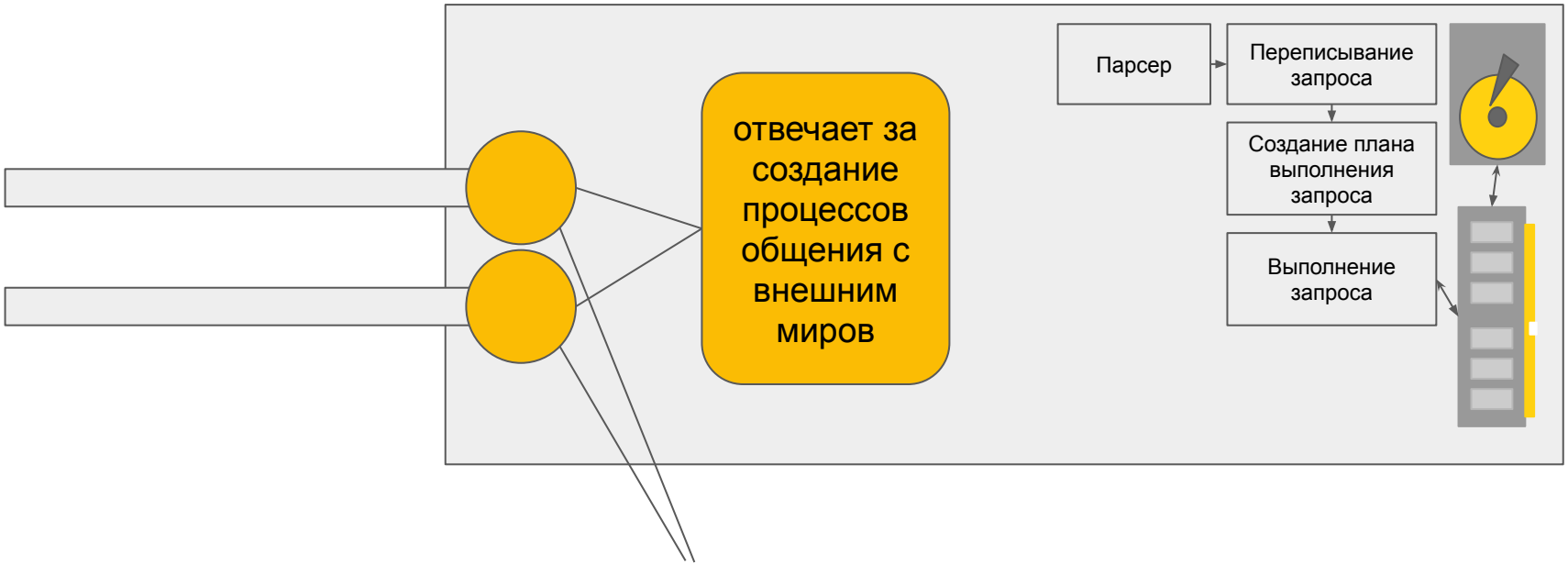




специальный процесс на стороне СУБД

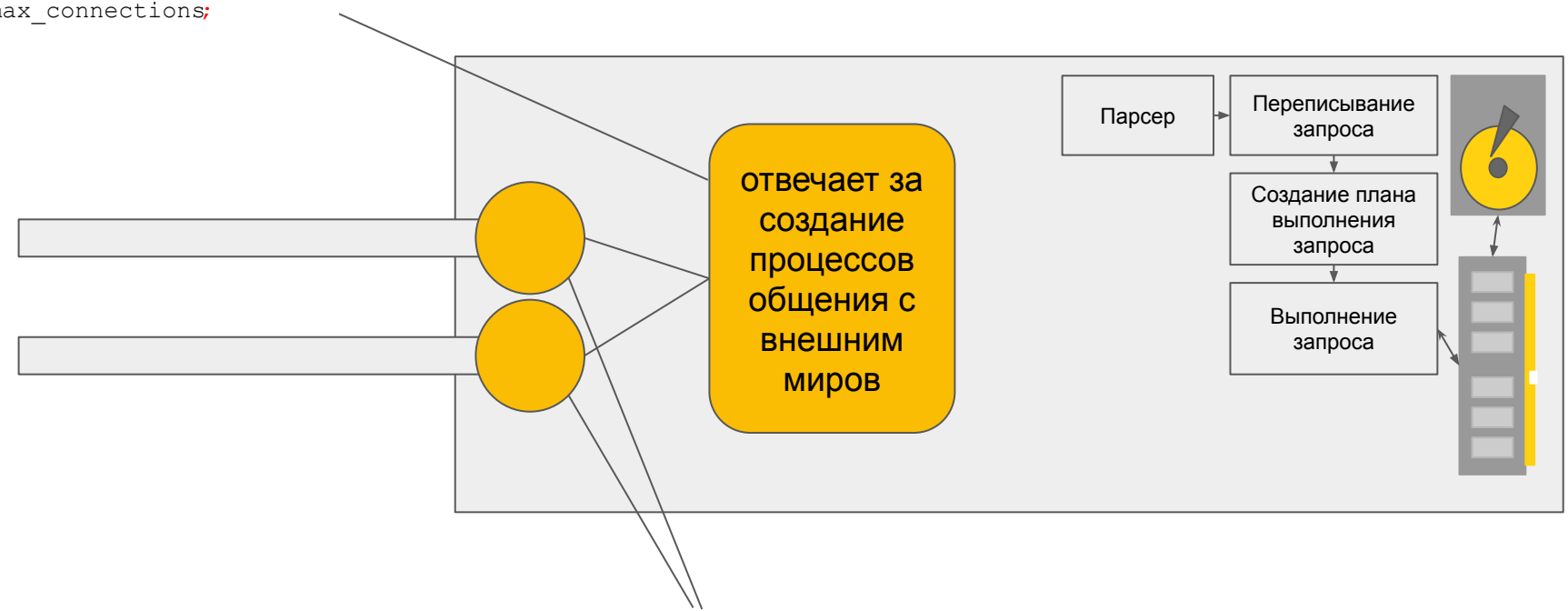


специальный процесс на стороне СУБД

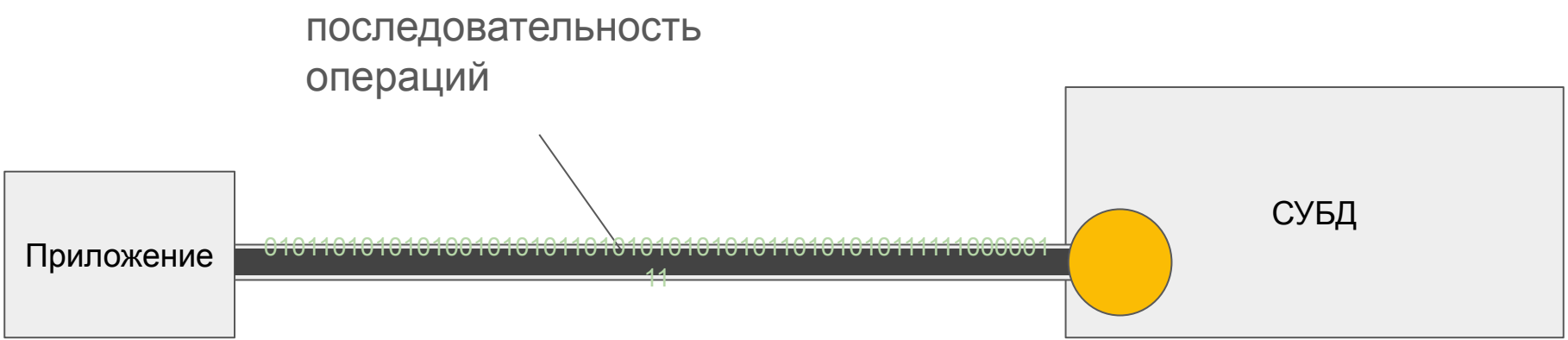


специальный процесс на стороне СУБД

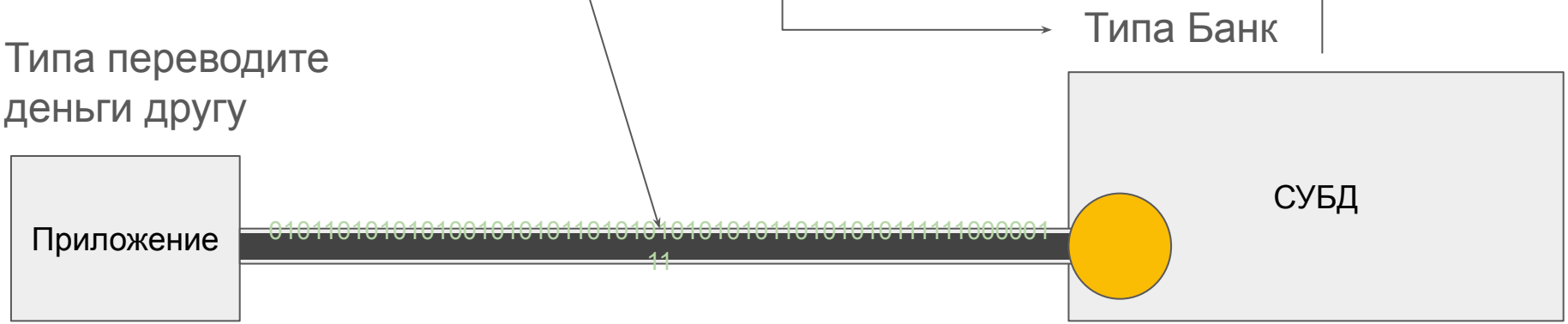
```
SELECT @@MAX_CONNECTIONS;
SHOW VARIABLES LIKE "max_connections";
SHOW max_connections;
```

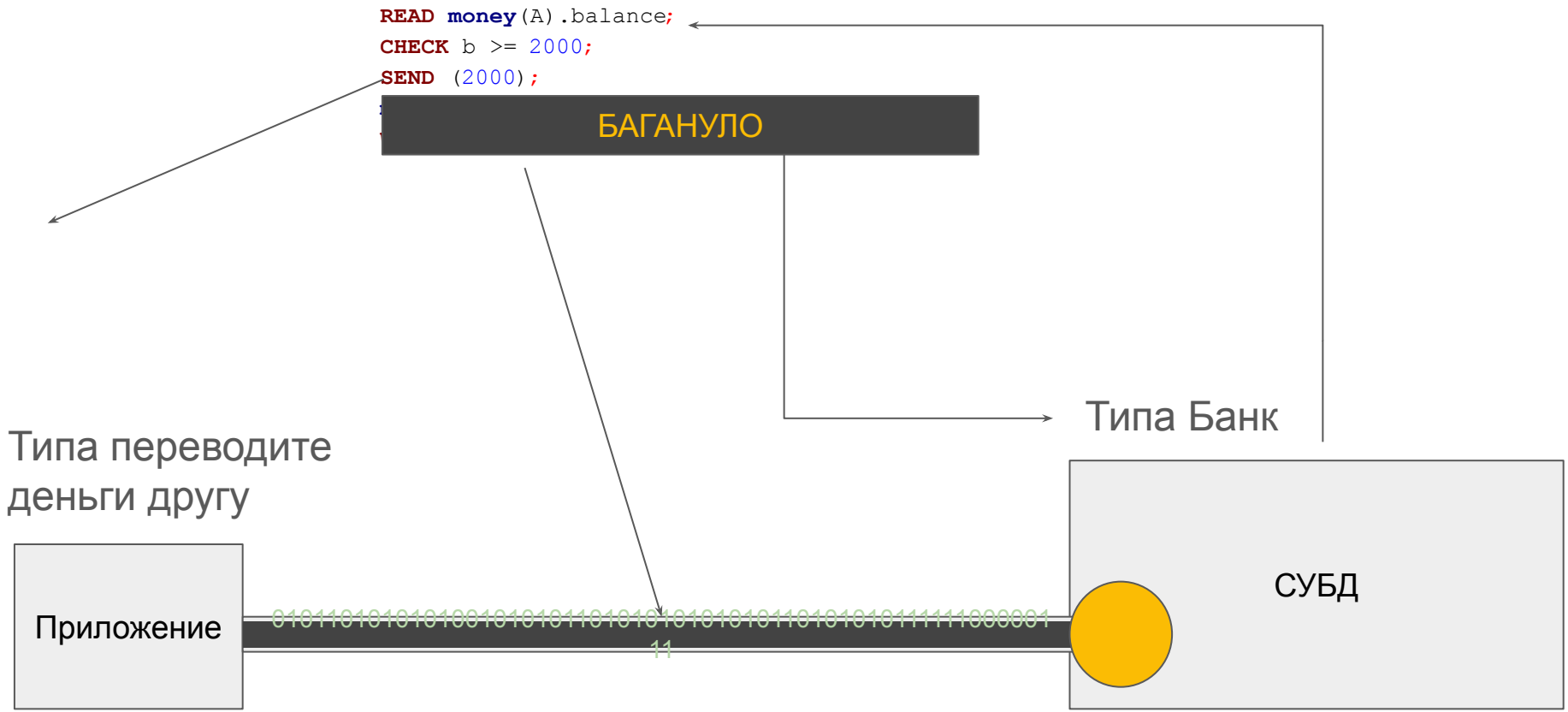


специальный процесс на стороне СУБД


последовательность операций
Приложение
СУБД

```
READ money(A).balance;
CHECK b >= 2000;
SEND (2000);
money(A).balance = money(A).balance - 2000;
WRITE(A);
```

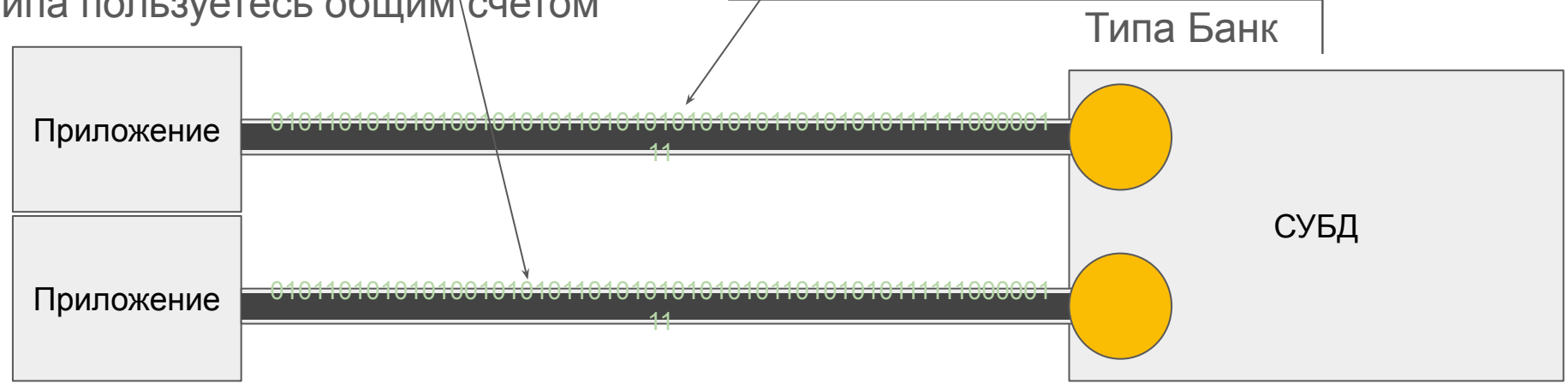



READ money(A).balance;
CHECK b >= 2000;
SEND (2000);
БАГАНУЛО
Типа переводите деньги другу
Типа Банк
Приложение
СУБД

```
READ money(A).balance = 10000;
CHECK b >= 2000;
SEND (2000);
money(A).balance = money(A).balance - 2000;
WRITE(A); (8000)
```

```
READ money(A).balance = 10000;
CHECK b >= 2000;
SEND (2000);
money(A).balance = money(A).balance - 2000;
WRITE(A); (8000)
```

типа пользуетесь общим счетом



Транзакции

Есть последовательности инструкций запроса и/или изменения данных.

Набор этих операций, которые можно считать в конкретном случае логически одной, то есть полностью произойдут или полностью отменятся, называют **транзакцией**.

Соответственно по результату транзакции происходит

COMMIT
фиксирует текущую транзакцию, то есть делает обновления, выполняемые транзакцией, постоянными в базе данных.

ROLLBACK
вызывает откат текущей транзакции, то есть отменяет все обновления, выполненные инструкциями SQL в транзакции. Таким образом, состояние базы данных восстанавливается таким, каким оно было до выполнения первой инструкции транзакции

База данных обеспечивает абстрагирование транзакции делая её неделимой. Либо фиксирует действия транзакции после завершения всех ее этапов, либо откатывает все ее действия в случае, если транзакция не смогла успешно выполнить все свои действия.

После выполнения транзакции база данных остается “адекватной”.

Причем внутри транзакции при её работе может все что угодно нарушать, главное чтобы по итогу все ограничения целостности совпадали.

Во время работы транзакции другие транзакции не должны оказывать влияние на результаты её работы.

Внезависимости от того, что где-либо что-то происходит - если мы получили Ок после выполнения транзакции, то это действительно Ок - все изменения произведены даже если будет сбой.

Вот эта концепция получила название **ACID**:

Atomicity (**Атомарность**)

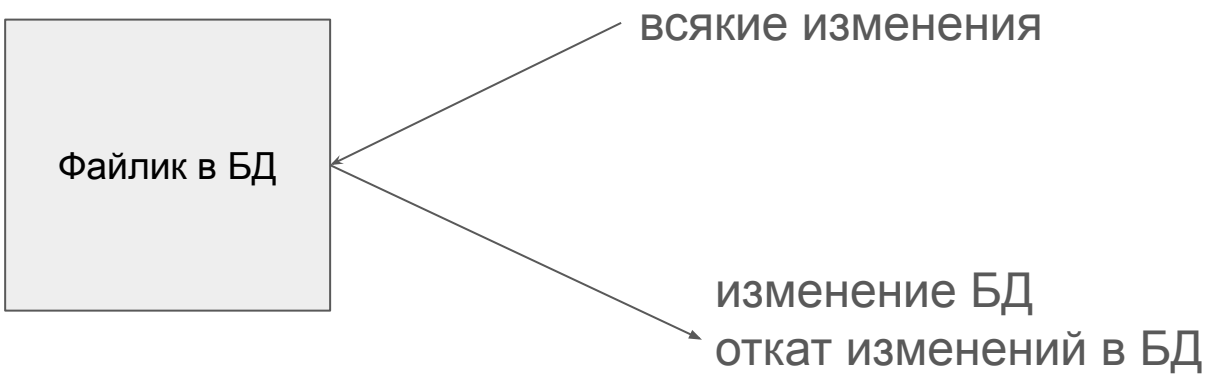
Consistency (**Согласованность**)

Isolation (**Изоляционность**)

Durability (**Устойчивость**)

Atomicity (**Атомарность**)

СУБД для записи изменений, произведенных с данными использует механизм transaction log:



Isolation (**Изоляционность**)

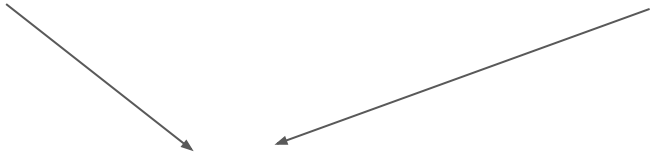
Пользователи отправляют транзакции, и каждый транзакция выполняется так, как если бы она выполнялся сам по себе.

Но СУБД обеспечивает параллелизм, чередуя действия (чтение/запись объектов базы данных) используемых внутри транзакций. Соответственно нужен способ чередовать действия внутри транзакции, но при этом создавать впечатление, что они выполняются независимо.

```
DECLARE
A = 2000
B = 2000
CHECK A + B = 4000;
```

```
T1
BEGIN
UPDATE A = A - 500
UPDATE B = B + 500
END
```

```
T2
BEGIN
UPDATE A = A * 1.5
UPDATE B = B * 1.5
END
```



```
CHECK A + B = 6000;
```


```
DECLARE
A = 2000
B = 2000
CHECK A + B = 4000;
```

T1 BEGIN UPDATE A = A - 500 UPDATE B = B + 500 END	T2 BEGIN UPDATE A = A * 1.5 UPDATE B = B * 1.5 END

```
CHECK A + B = 6000;
```

T1 BEGIN UPDATE A = A - 500 UPDATE B = B + 500 END	T2 BEGIN UPDATE A = A * 1.5 UPDATE B = B * 1.5 END

```
CHECK A + B = 6000;
```


```
DECLARE
A = 2000
B = 2000
CHECK A + B = 4000;
```

T1	T2
BEGIN UPDATE A = A - 500 UPDATE B = B + 500 END	BEGIN UPDATE A = A * 1.5 UPDATE B = B * 1.5 END

T1	T2
BEGIN UPDATE A = A - 500 UPDATE B = B + 500 END	BEGIN UPDATE A = A * 1.5 UPDATE B = B * 1.5 END

```
CHECK A + B = 6000;
```


```
DECLARE
A = 2000
B = 2000
CHECK A + B = 4000;
```

T1	T2
BEGIN	BEGIN
UPDATE A = A - 500	UPDATE A = A * 1.5
UPDATE B = B + 500	UPDATE B = B * 1.5
END	END

CHECK A + B = 6000;

T1	T2
BEGIN	BEGIN
UPDATE A = A - 500	UPDATE A = A * 1.5
	UPDATE B = B * 1.5
UPDATE B = B + 500	END
END	

CHECK A + B = 5750.0 ?????

Есть вариант пускать транзакции последовательно.

Как понять, что работа параллельных транзакций выполнено правильно?

Есть вариант пускать транзакции последовательно.

если =

Как понять, что работа параллельных транзакций выполнено правильно?

Есть вариант пускать транзакции последовательно.

если =

Как понять, что работа параллельных транзакций выполнено правильно?

R(A) - чтение объекта A базы данных

W(A) - запись объекта A базы данных

Соответственно возможны аномалии:

- Read-Write (R-W)
- Write-Read (W-R)
- Write-Write (W-W)

R(A) - чтение объекта A базы данных

W(A) - запись объекта A базы данных

Соответственно возможны аномалии:

Read-Write (R-W) (Unrepeatable Read)

Write-Read (W-R)

Write-Write (W-W)

T1	T2
BEGIN R(A) = 2000	BEGIN R(A) = 2000 W(A) = 3000 COMMIT
R(A) = 3000 COMMIT	

Соответственно возможны аномалии:

Read-Write (R-W)

Write-Read (W-R) Dirty Read

Write-Write (W-W)

T1	T2
BEGIN R(A) = 2000 W(A) = 3000 **ROLLBACK**	**BEGIN** R(A) = 3000 W(A) = 3000 - 500 **COMMIT**

Соответственно возможны аномалии:

- Read-Write (R-W)
- Write-Read (W-R)
- **Write-Write (W-W)** Lost Update

T1	T2
BEGIN	BEGIN
R(A) = 2000	R(A) = 2000
	W(A) = 2000 - 500
	COMMIT
W(A) = 2000 - 500	
COMMIT