Базы Данных

Лекция 4

Домашнее задание

Для каждого уникального товара вывести количество купленных товаров в заказах. В выводе должно быть (DESCRIPTION, SUM_QTY)

Домашнее задание

```
select DESCRIPTION, sum(SUM_QTY) as SUM_QTY
from (
    select MFR, PRODUCT, sum(SUM_QTY) as SUM_QTY
    from (
            /* Все уникальные товары из таблицы PRODUCTS */

        select MFR_ID as MFR, trim(PRODUCT_ID) as PRODUCT, 0 as SUM_QTY
        from PRODUCTS
        union
        /* Все уникальные товары с суммой по количеству в заказах по каждому товару из таблицы ORDERS */
        select MFR, trim(PRODUCT) as PRODUCT, sum(QTY) as SUM_QTY
        from ORDERS
        group by MFR, PRODUCT
        )
    group by MFR, PRODUCT
    ) join PRODUCTS on (trim(PRODUCT_ID) = PRODUCT and MFR = MFR_ID)
group by MFR, PRODUCT;
```

Домашнее задание

SQLITE:

```
SELECT DISTINCT DESCRIPTION, SUM(QTY) AS
sum_qty
FROM ORDERS, PRODUCTS
WHERE (trim(PRODUCT) = trim(PRODUCT_ID))
AND (trim(MFR) = trim(MFR_ID))
GROUP BY PRODUCT_ID, MFR_ID;
```

Домашнее задание

ORACLE:

```
SELECT DISTINCT DESCRIPTION, SUM(QTY) AS
sum_qty
FROM ORDERS, PRODUCTS
WHERE PRODUCT = PRODUCT_ID
AND trim(MFR) = MFR_ID
GROUP BY PRODUCT_ID, MFR_ID, DESCRIPTION;
```

Домашнее задание

Для каждого сотрудника определить, что больше: отношение квоты сотрудника к квоте офиса или продаж сотрудника к продажам офиса и вывести наибольшее из них. Если определить нельзя, то вывести ноль. Также вывести город сотрудника; Если офис для сотрудника не определен, то вывести ‘N/A’; В выводе должно быть (Name, OFFICE, res)

Домашнее задание

```
select NAME as Name, COALESCE(CITY, 'N/A') as OFFICE,
    case
        when (TARGET is null or TARGET = 0 or OFFICES.SALES =
0)
            then 0
        when (QUOTA*1.0/TARGET >
SALESREPS.SALES*1.0/OFFICES.SALES)
            then QUOTA*1.0/TARGET
        when (QUOTA*1.0/TARGET <
SALESREPS.SALES*1.0/OFFICES.SALES)
            then SALESREPS.SALES*1.0/OFFICES.SALES
        else 0
    end as res
from SALESREPS left join OFFICES on (REP_OFFICE = OFFICE);
```

Домашнее задание

Необходимо рассмотреть только те заказы сотрудника, сумма которых больше средней суммы заказов данного сотрудника, которые были совершены в 2007 году. Из полученных заказов сформировать итоговой вывод, который для каждого сотрудника указывает количество таких заказов. В выводе должно быть (NAME, cnt_of_orders)

Домашнее задание 3 ORACLE

```
SELECT NAME, COUNT(*) AS cnt_of_orders
FROM ORDERS O
    INNER JOIN SALESREPS
        ON O.REP = SALESREPS.EMPL_NUM
WHERE
  AMOUNT > (SELECT avg(AMOUNT) FROM ORDERS
WHERE O.REP = ORDERS.REP
              AND (EXTRACT(YEAR FROM ORDER_DATE) =
2007))
GROUP BY REP, NAME;
```

Домашнее задание 3 SQLITE

```
SELECT NAME, COUNT(*) AS cnt_of_orders
FROM ORDERS O
    INNER JOIN SALESREPS
        ON O.REP = SALESREPS.EMPL_NUM
WHERE
    AMOUNT > (SELECT avg(AMOUNT) FROM ORDERS
                WHERE O.REP = ORDERS.REP
                AND (strftime('%Y', ORDER_DATE) = '2007'))
GROUP BY REP, NAME;
```

Домашнее задание 4

Вывести только те товары, для которых количество на складе меньше на 20%, чем общее количество этого товара, которое было сделано в заказах. Вывести (DESCRIPTION)

Домашнее задание 4 sqlite

```
select DESCRIPTION
from PRODUCTS join (
    select MFR, PRODUCT, sum(QTY) as sum_qty
    from ORDERS
    group by MFR, PRODUCT
) on (MFR_ID = MFR and trim(PRODUCT_ID) = trim(PRODUCT))
where (QTY_ON_HAND = 0.8 * sum_qty);
```

Домашнее задание 4 sqlite

```
SELECT DESCRIPTION
FROM PRODUCTS
WHERE QTY_ON_HAND = 0.8 * (SELECT SUM(QTY)
FROM ORDERS
                        WHERE (MFR_ID = MFR
                          AND trim(PRODUCT_ID) =
trim(PRODUCT)));
```

Домашнее задание 4 oracle

```
SELECT DESCRIPTION
FROM PRODUCTS
WHERE QTY_ON_HAND = 0.8 * (SELECT SUM(QTY)
FROM ORDERS
                    WHERE (MFR_ID = MFR
                      AND PRODUCT_ID = PRODUCT));
```

Домашнее задание 5

Для каждого дня, когда был совершен заказ или принят сотрудник, рассчитать абсолютную разницу между количеством сотрудником и количеством заказов, сделанных в этот день. В выводе должно быть (date, abs_diff)

Домашнее задание 5 oracle

```
select date, abs(cnt) as abs_diff
from (
    select HIRE_DATE as date, count(*) as cnt
    from SALESREPS
    group by HIRE_DATE
union
    select ORDER_DATE as date, -count(*) as cnt
    from ORDERS
    group by ORDER_DATE
)
group by date;
```

Домашнее задание 6

* Учитывая условия задачи 5 найти даты с минимальными и максимальными разницами в заказах. В выводе должно быть (date, abs_diff).

Домашнее задание 6

```
select date, abs(sum(val_hire)-sum(val_order)) as abs_diff
from (
    select HIRE_DATE as date, count(*) as val_hire, 0 as val_order
    from SALESREPS
    group by HIRE_DATE
union
    select ORDER_DATE as date, 0 as val_hire, count(*) as
val_order
    from ORDERS
    group by ORDER_DATE
)
group by date
-- продолжение на след слайде
```

Домашнее задание 6

```
having abs_diff in (
        /* Поиск минимума среди абсолютной разности */
        select min(abs_diff) as val
        from (
            select date, abs(sum(val_hire)-sum(val_order)) as abs_diff
            from (
                select HIRE_DATE as date, count(*) as val_hire, 0 as val_order
                from SALESREPS
                group by HIRE_DATE
            union
                select ORDER_DATE as date, 0 as val_hire, count(*) as val_order
                from ORDERS
                group by ORDER_DATE
            )
            group by date
        ) – продолжение на след слайде
```

Домашнее задание 6

```
union
/* Поиск максимума среди абсолютной разности */
select max(abs_diff) as val
from (
    select date, abs(sum(val_hire)-sum(val_order)) as abs_diff
    from (
        select HIRE_DATE as date, count(*) as val_hire, 0 as val_order
        from SALESREPS
        group by HIRE_DATE
    union
        select ORDER_DATE as date, 0 as val_hire, count(*) as val_order
        from ORDERS
        group by ORDER_DATE
    )
    group by date
)
);
```

Домашнее задание 6

```
WITH DIFF_CALC
AS
(
	select date, abs(cnt) as abs_diff
	from (
		select HIRE_DATE as date, count(*) as cnt
		from SALESREPS
		group by HIRE_DATE
	union
		select ORDER_DATE as date, -count(*) as cnt
		from ORDERS
		group by ORDER_DATE
	)
	group by date
)
SELECT date, abs_diff
FROM DIFF_CALC
WHERE abs_diff = (SELECT max(abs_diff) FROM DIFF_CALC)
OR abs_diff = (SELECT min(abs_diff) FROM DIFF_CALC);
```

Домашнее задание 6

```
SELECT date, abs_diff
FROM hw3_t5_tmp
WHERE abs_diff = (SELECT max(abs_diff) FROM hw3_t5_tmp)
OR abs_diff = (SELECT min(abs_diff) FROM hw3_t5_tmp);
```

Домашнее задание 7

Вывести только те товары, по которым было совершенно менее, чем 3 заказа и сумма каждого из заказов была не более, чем 4000, а общая сумма заказов менее 9000;

Домашнее задание 7

```
SELECT DESCRIPTION
FROM PRODUCTS
WHERE NOT EXISTS
(
SELECT 1
FROM ORDERS
WHERE MFR_ID = MFR
AND TRIM(PRODUCT_ID) = TRIM(PRODUCT)
AND AMOUNT > 4000
)
AND 3 > (SELECT COUNT(*)
      FROM ORDERS
      WHERE MFR_ID = MFR
      AND TRIM(PRODUCT_ID) = TRIM(PRODUCT) )
AND 9000 > (SELECT SUM(AMOUNT)
      FROM ORDERS
      WHERE MFR_ID = MFR
      AND TRIM(PRODUCT_ID) = TRIM(PRODUCT) );
```

Обработка транзакций

Часто база данных должна обрабатывать события, которые приводят более чем к одному изменению в Базе Данных.

Обработка транзакций

Рассмотрим событие – прием нового заказа от клиента

- Добавление нового заказа в таблицу ORDERS
- Обновление фактического объема продаж для служащего, принявшего заказ
- Обновление фактического объема продаж для офиса, в котором работает данный служащий
- Обновление количества товара, имеющегося в наличие

Обработка транзакций

Для избегания нарушений целостности базы данных, четыре указанных изменения следует выполнить как единое целое.

Обработка транзакций

Для избегания нарушений целостности базы данных, четыре указанных изменения следует выполнить как единое целое.

Допустим произошел системный сбой или другая ошибка.

Тогда одна часть изменений была внесена, а другая нет. Это приводит к нарушению целостности хранимых данных и при последующих вычислениях результаты окажутся неверными.

Обработка транзакций

Изменения базы данных, которые были вызваны одним событием, необходимо вносить по принципу “Всё или ничего”.

SQL обеспечивает такое поведение посредством возможностей обработки транзакций.

Обработка транзакций

Транзакция – это несколько последовательных инструкций SQL, которые вместе образуют логическую единицу работы (unit of work).

Обработка транзакций

Транзакция – это несколько последовательных инструкций SQL, которые вместе образуют логическую единицу работы (unit of work).

Инструкции, входящие в транзакцию, обычно тесно связаны между собой и выполняют взаимосвязанные действия.

Обработка транзакций

Группировка инструкций в единую транзакцию указывает СУБД, что вся их последовательность должна выполняться так, чтобы пройти так называемый ACID-тест.

- **Atomic (атомарность)**
- **Consistent (целостность)**
- **Isolated (изолированность)**
- **Durable (постоянство)**

Обработка транзакций

Atomic **(Атомарность).** “Все, или ничего”. Выполняются успешно либо все операции транзакции, либо не выполняется ни одна из них. Если выполнены лишь некоторые инструкции, то транзакция оказывается неуспешной, и в результате будет выполнен откат выполненных инструкций. Только если все инструкции выполнены успешно, то тогда транзакция может рассматриваться как завершенная, а ее результаты фиксируются в БД.

Обработка транзакций

***Consistent (*Целостность).** Транзакций должна переводить БД из одного согласованного состояния в другое. База данных должна быть в согласованном состоянии по окончании каждой транзакции, а это означает, что должны выполняться все правила и ограничения. Ни один пользователь не должен иметь доступ к данным, несогласованным из-за незавершенности транзакции.

Обработка транзакций

Isolated **(Изолированность).** Каждая транзакция должна выполняться сама по себе, без взаимодействия с другими транзакциями. Для этого ни одна транзакция не должна работать с изменениями, вносимыми другой транзакцией, пока та не будет завершена.

Обработка транзакций

Durable (Постоянство). По завершении транзакции все внесенные ею изменения должны быть сохранены. Данные должны быть в согласованном состоянии, даже если по окончании транзакции произойдет аппаратный или программный сбой. В ООП программировании это называется *персистентность* (persistence).

Обработка транзакций

Пример транзакций:

- **Прием заказа.** Программа ввода заказа должна:

a) Выполнить запрос к Products и проверить наличие товара на складе.

b) Добавить заказ в таблицу ORDERS

c) Обновить таблицу PRODUCTS, вычтя заказанное кол-во товара из кол-ва товара, имеющегося в наличии;

Обработка транзакций

Пример транзакций:

- **Прием заказа.** Программа ввода заказа должна:

(d) Обновить таблицу SALESREPS, добавив стоимость заказа к объему продаж служащего, принявшего заказ;

(e) Обновить таблицу OFFICES, добавив стоимость заказа к объему продаж офиса, в котором работает служащий.

Обработка транзакций

Пример транзакций:

- **Отмена заказа.**

a) Удалить заказ из таблицы ORDERS.

b) Обновить таблицу PRODUCTS, откорректировав кол-во товара, имеющегося в наличии.

c) Обновить таблицу SALESREPS, вычтя стоимость заказа из объема продаж служащего;

d) Обновить таблицу OFFICES, вычтя стоимость заказа из объема продаж офиса.

Обработка транзакций

Пример транзакций:

- **Перевод клиента**.

a) Обновить таблицу CUSTOMERS;

b) Обновить таблицу ORDERS, изменив имя служащего, ответственного за заказы данного клиента;

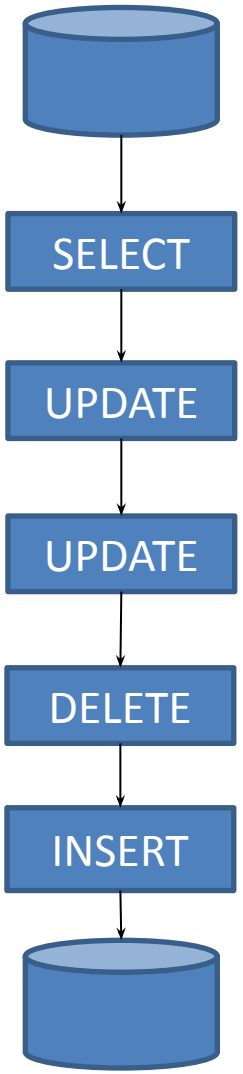
c) Обновить таблицу SALESREPS, уменьшив план для служащего, теряющего клиента;

d) Обновить таблицу SALESREPS, увеличив план служащего, приобретающего клиента.

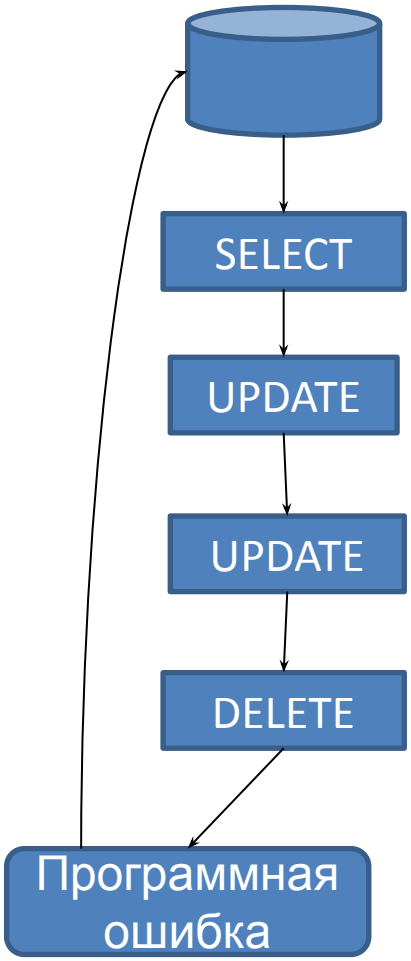
Обработка транзакций

Инструкции, входящие в транзакцию, выполняются атомарно, как единое неделимое целое. Либо все инструкции будут выполнены успешно, либо ни одна из них не должны быть выполнена.

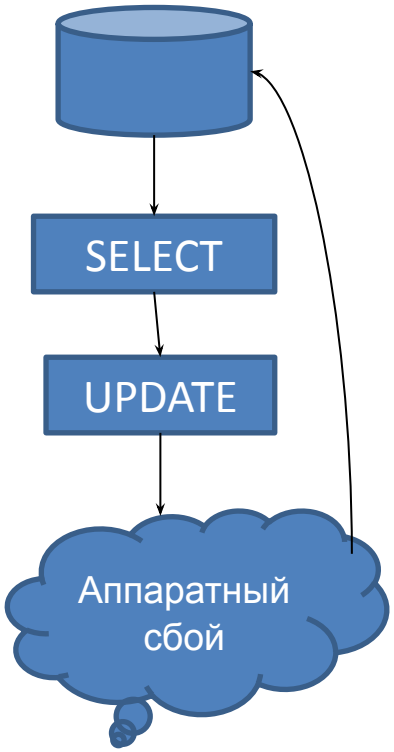
Обработка транзакций



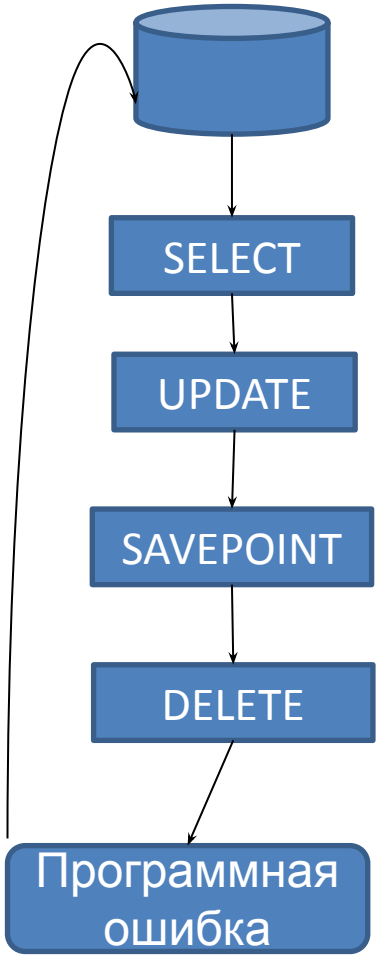
Состояние БД после транзакции



СУБД отменяет все изменения



СУБД отменяет все изменения



СУБД отменяет все изменения

Модель транзакций ANSI/ISO SQL

В стандарте ANSI/ISO определена *модель транзацкций* SQL, а также семь инструкций для поддержки работы с транзакциями

Модель транзакций ANSI/ISO SQL

- **START TRANSACTION.** Устанавливает свойства новой транзакции и запускает транзакцию

Модель транзакций ANSI/ISO SQL

- **START TRANSACTION.**
- **SET TRANSACTION.** Устанавливает свойства очередной выполняемой транзакции. Не влияет на текущую выполняемую транзакцию.

Модель транзакций ANSI/ISO SQL

- **START TRANSACTION.**
- **SET TRANSACTION.**
- **SET CONSTRAINTS.** Устанавливает режим ограничений в текущей транзакции. Режим ограничений управляет тем, применяется ли ограничение немедленно или откладывается до более позднее момента.

Модель транзакций ANSI/ISO SQL

- **START TRANSACTION.**
- **SET TRANSACTION.**
- **SET CONSTRAINTS.**
- **SAVEPOINT.** Создает точку сохранения в пределах транзакции. Точка сохранения представляет собой место в посл-ти событий транзакции, к-ое может выступать в качестве промежуточной точки восстановления. Откат текущей транзакции может быть выполнен не к началу транзакции, а к точке сохранения.

Модель транзакций ANSI/ISO SQL

- **START TRANSACTION.**
- **SET TRANSACTION.**
- **SET CONSTRAINTS.**
- **SAVEPOINT.**
- **RELEASE SAVEPOINT.** Освобождает точку сохранения и все ресурсы, которые она могла захватить.

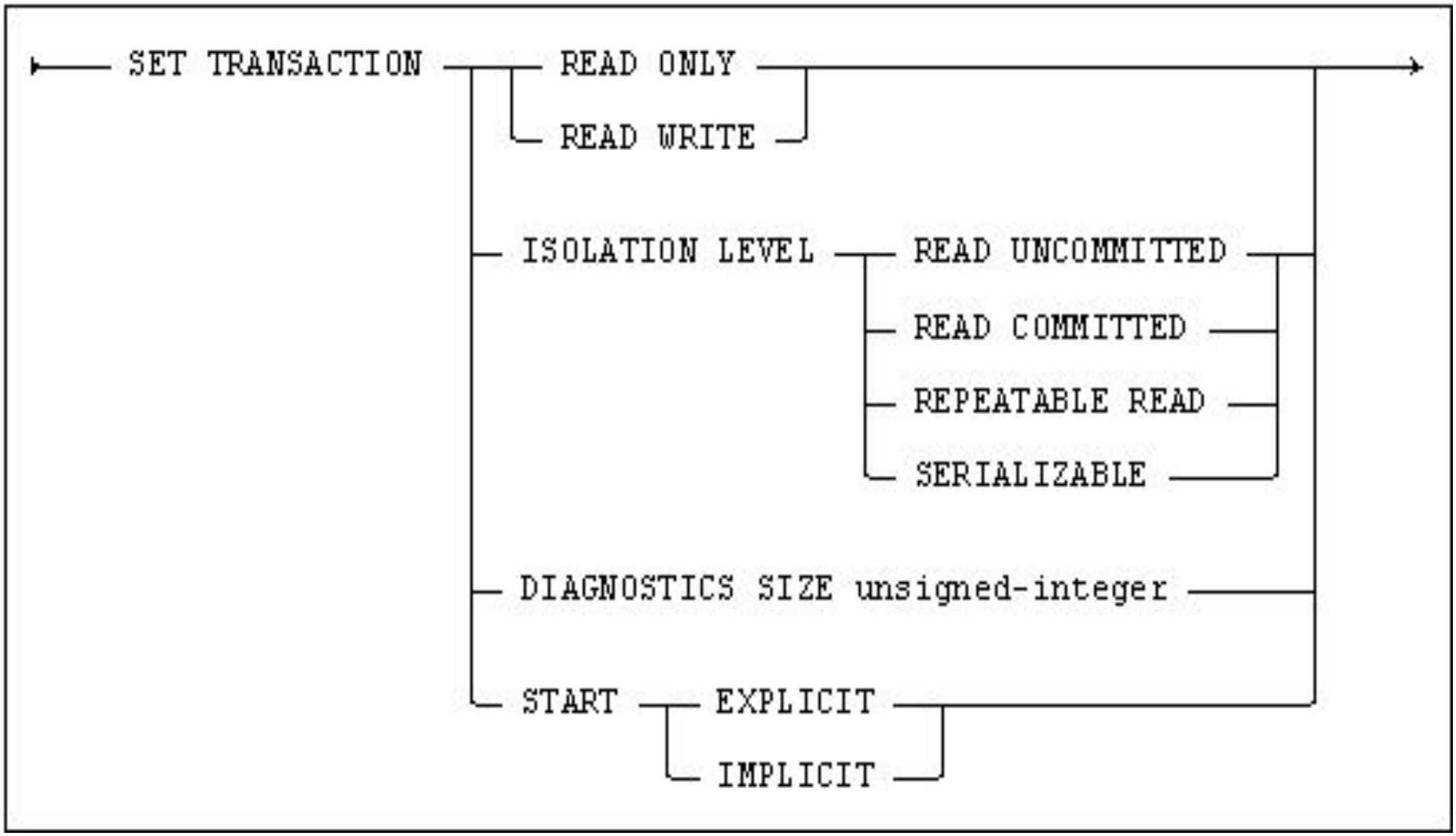
Модель транзакций ANSI/ISO SQL

- **COMMIT.** Завершает успешную транзакцию и сохраняет все внесенные изменения в базе данных.

Модель транзакций ANSI/ISO SQL

- **ROLLBACK.** При использовании без точки сохранения прекращает неудачную транзакцию и выполняет откат всех изменений к началу транзакции, по сути, возвращая БД к ее согласованному состоянию, имевшему место до начала транзакции (как если бы транзакция никогда не выполнялась). При использовании с т. сохранения выполняет откат транзакции к именованной точке сохранения, но допускает продолжение выполнения транзакции.

Инструкции SET TRANSACTION и START TRANSACTION



Инструкции SAVEPOINT и RELEASE SAVEPOINT

SAVEPOINT *имя_точки_сохранения;*

Преимущество – возможность отката части транзакции в случае небольших и потенциально восстановительных ошибок.

Пример – приложение для ввода заказов может создавать точку сохранения после каждой введенной строки заказа. Если добавление новой строки из заказа приводит к превышению лимита кредита, приложение может выполнить откат к т. сохранения, установленной непосредственно перед

Инструкции SAVEPOINT и RELEASE SAVEPOINT

SAVEPOINT *имя_точки_сохранения;*

Недостаток – потенциальное использование большого кол-ва ресурсов.

RELEASE SAVEPOINT *имя_точки_сохранения;*

Инструкции COMMIT и ROLLBACK

- **COMMIT [WORK] [AND [NO] CHAIN]**
- **ROLLBACK [WORK] [AND [NO] CHAIN] [TO SAVEPOINT *имя_точки_сохранения*]**

WORK. Ключевое слово. Включено в стандарт для совместимости.

AND [NO] CHAIN. Передача свойств следующей транзакции.

TO SAVEPOINT. Указание в какое место произвести откат

Пример COMMIT

Изменить объем заказа 113051 с 4 до 10 единиц, что повышает его сумму с $1458 до $3550. Заказ на товар QSA-XK47 был принят Ларри Фитчем (ид 108), к-ый работает в Л.А. (21).

```
UPDATE ORDERS
SET QTY = 10, AMOUNT = 3550.00
WHERE ORDER_NUM = 113051;

UPDATE SALESREPS
SET SALES = SALES – 1458.00 + 3550.00
WHERE EMPL_NUM = 108;
```

Пример COMMIT

```
UPDATE OFFICES
SET SALES = SALES – 1458.00 + 3550.00
WHERE OFFICE = 21;

UPDATE PRODUCTS
SET QTY_ON_HAND = QTY_ON_HAND + 4 – 10
WHERE MFR_ID = ‘QSA’
AND PRODUCT_ID = ‘XK47’;
-- подтверждение
COMMIT WORK;
```

Пример ROLLBACK

```
UPDATE ORDERS
SET QTY = 10, AMOUNT = 3550.00
WHERE ORDER_NUM = 113051;

UPDATE SALESREPS
SET SALES = SALES – 1458.00 + 3550.00
WHERE EMPL_NUM = 108
```

Пример ROLLBACK

```
UPDATE OFFICES
SET SALES = SALES – 1458.00 + 3550.00
WHERE OFFICE = 21;

UPDATE PRODUCTS
SET QTY_ON_HAND = QTY_ON_HAND + 4 – 10
WHERE MFR_ID = ‘QAS’
AND PRODUCT_ID = ‘XK47’
-- производитель не QAS, а QSA
ROLLBACK WORK;
```

Транзакции и работа в многопользовательском режиме

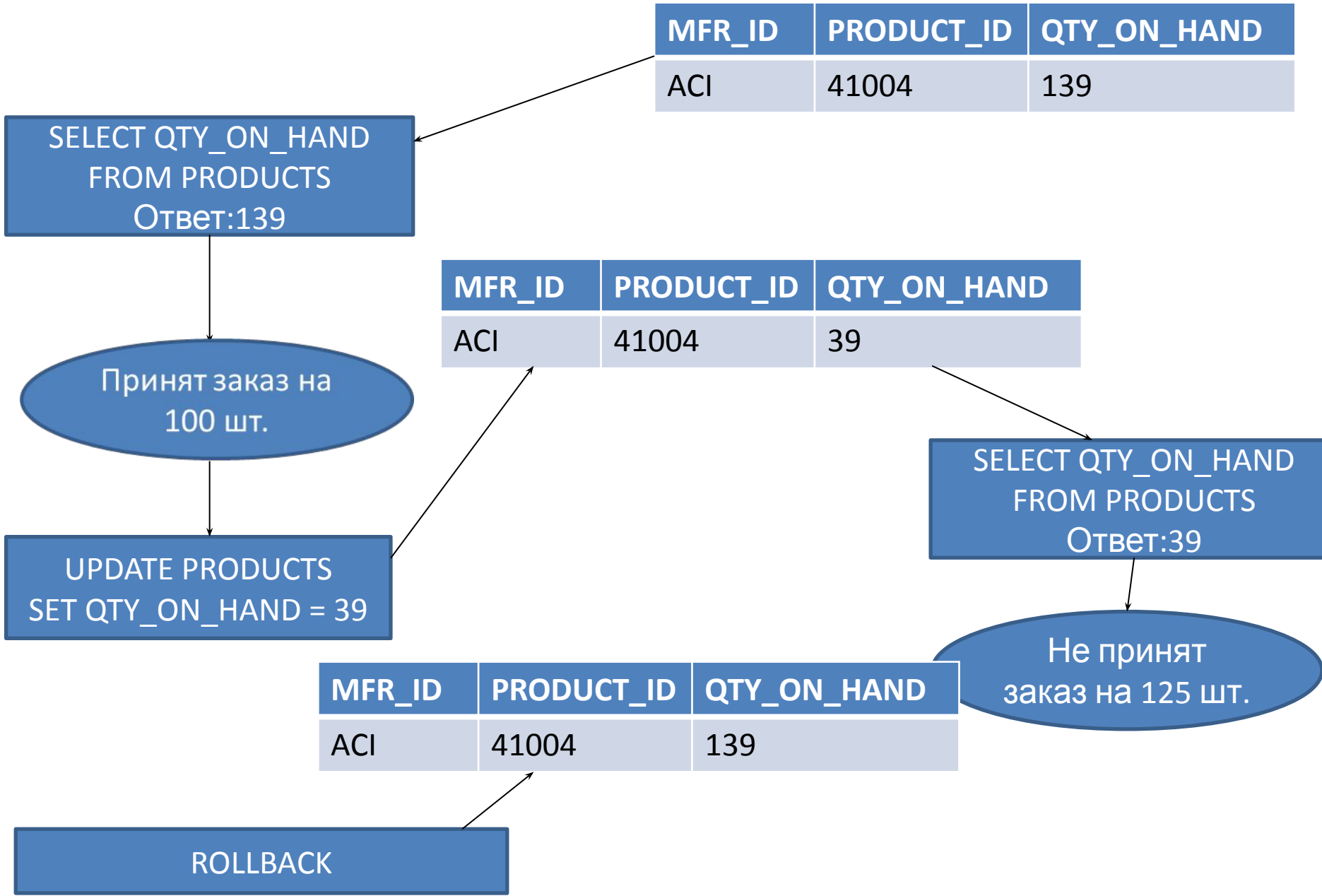
Если с БД работают одновременно двое или более пользователей, СУБД должна не только осуществлять восстановление базы данных после отмены транзакций, но и гарантировать, что пользователи не будут мешать друг другу.

В идеальном случае каждый пользователь должен работать с БД так, как если бы он не имел к ней многопользовательский доступ, и не должен беспокоиться о действиях других пользователей.

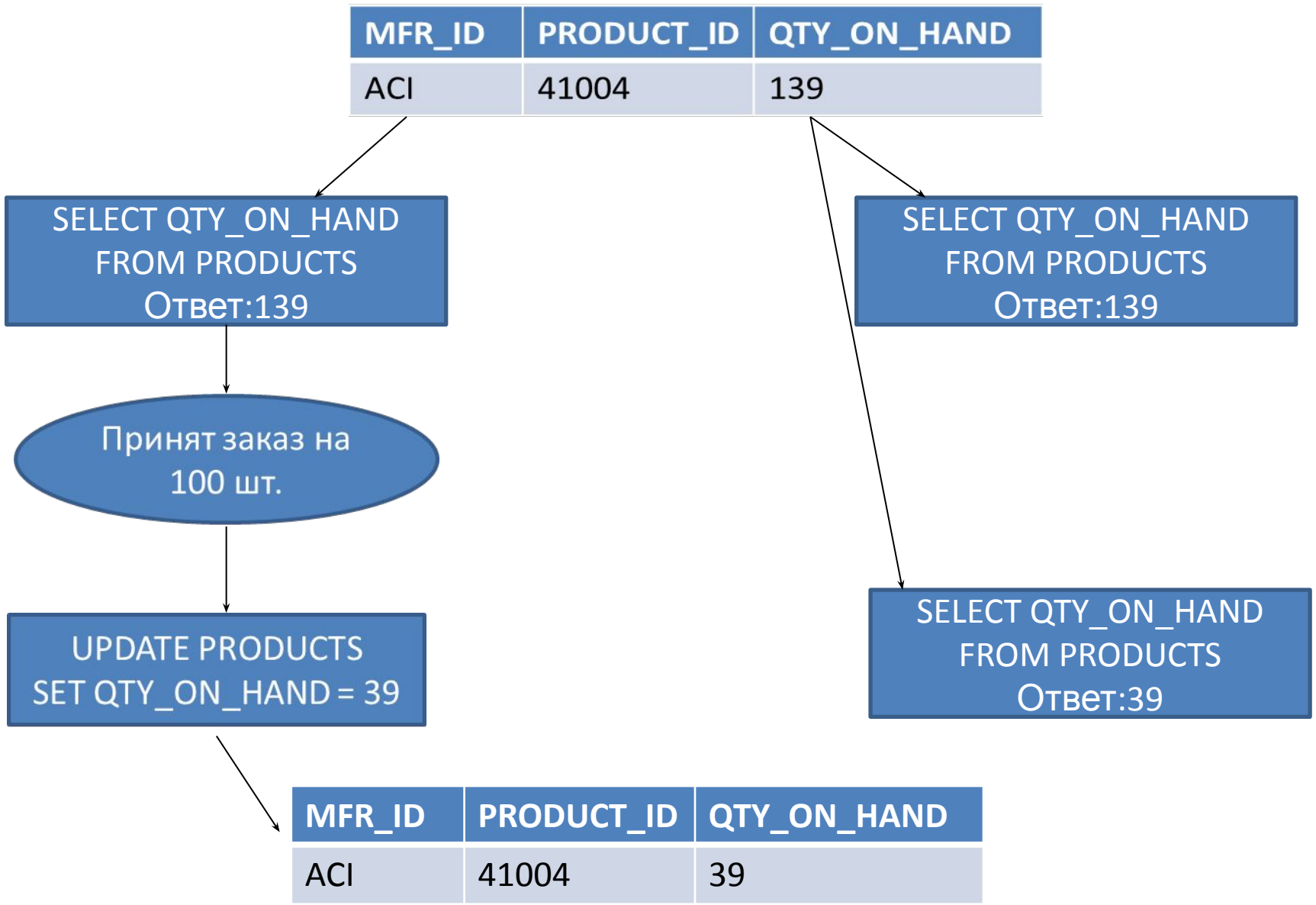
DIRTY WRITE P0

Транзакция T1 модифицирует строку. Другая транзакция T2 также модифицирует эту строку до COMMIT или ROLLBACK от T1. Если T1 или T2 произведет ROLLBACK не ясно, какие данные должны быть корректны.

DIRTY READ P1



P2 NONREPEATABLE READ



P3 PHANTOM

ORDER_NUM	AMOUNT
112961	31500.00
113012	3745.00

SELECT *
FROM ORDERS

112962,31500

113012,3745

INSERT INTO VALUES
(118102, .., 5.000)

COMMIT

ORDER_NUM	AMOUNT
112961	31500.00
118102	5000.00
113012	3745.00

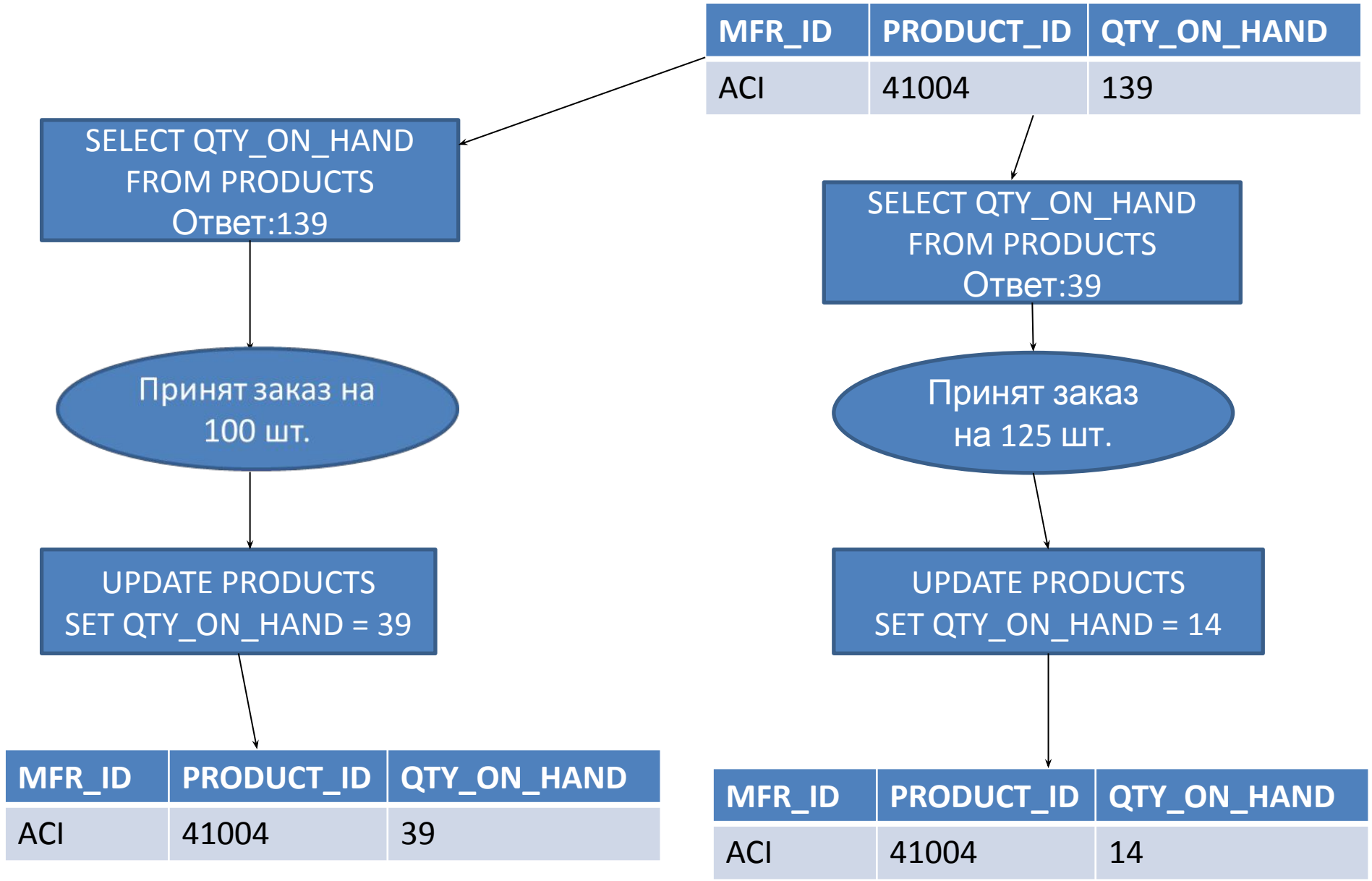
SELECT *
FROM ORDERS

112962,31500

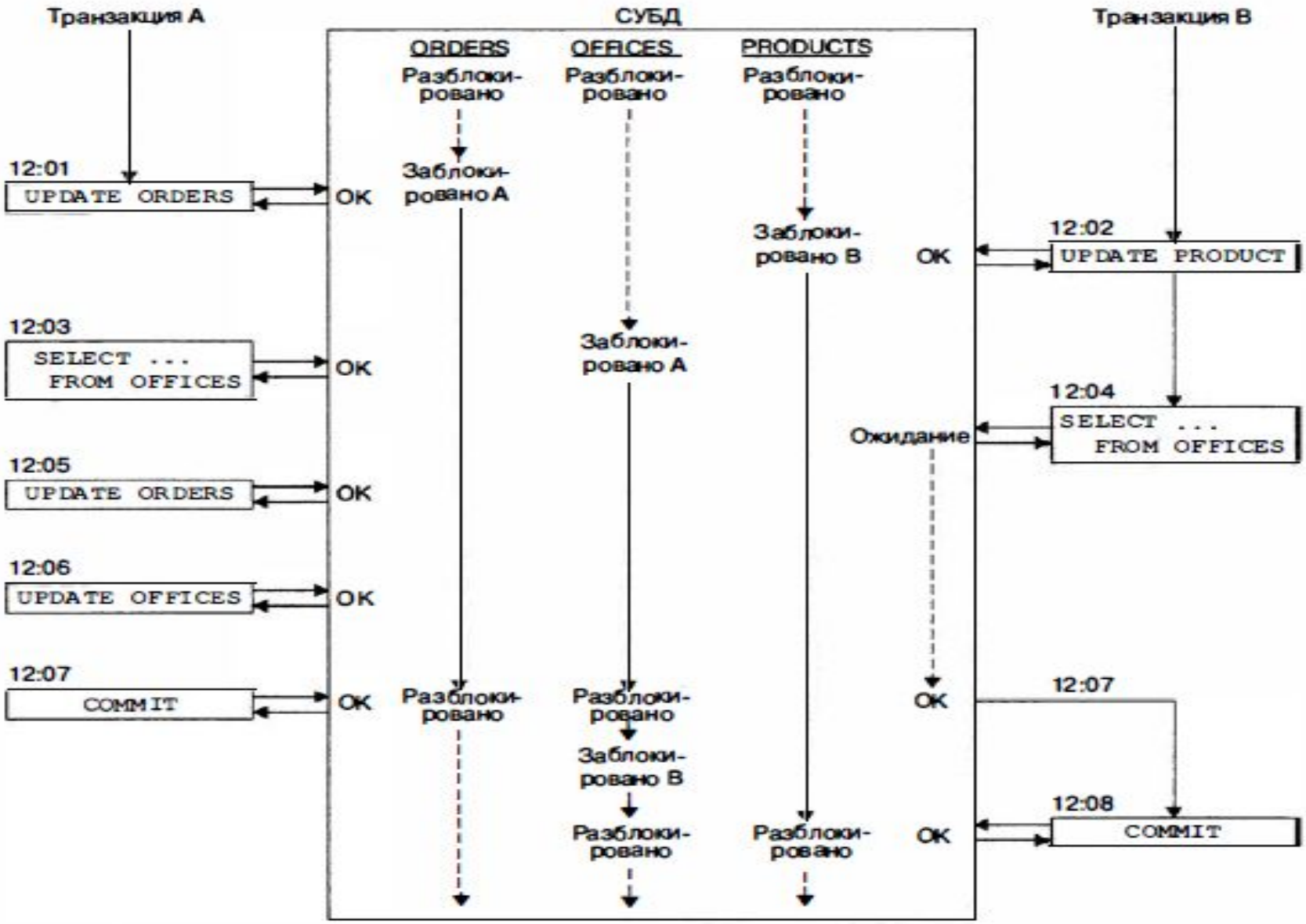
118102,5000

113012,3745

P4 LOST UPDATE



Блокировка при параллельном выполнении 2 транзакций



Уровни блокировки

- На уровне БД
- На уровне таблицы
- На уровне страниц
- На уровне строк

Виды блокировок

- **Блокировка с обеспечением совместного доступа**, или блокировка без монополизации (shared lock). Когда транзакция извлекает информацию из базы данных, СУБД применяет блокировку без монополизации. При этом другие транзакции, выполняемые параллельно, могут извлекать те же данные.

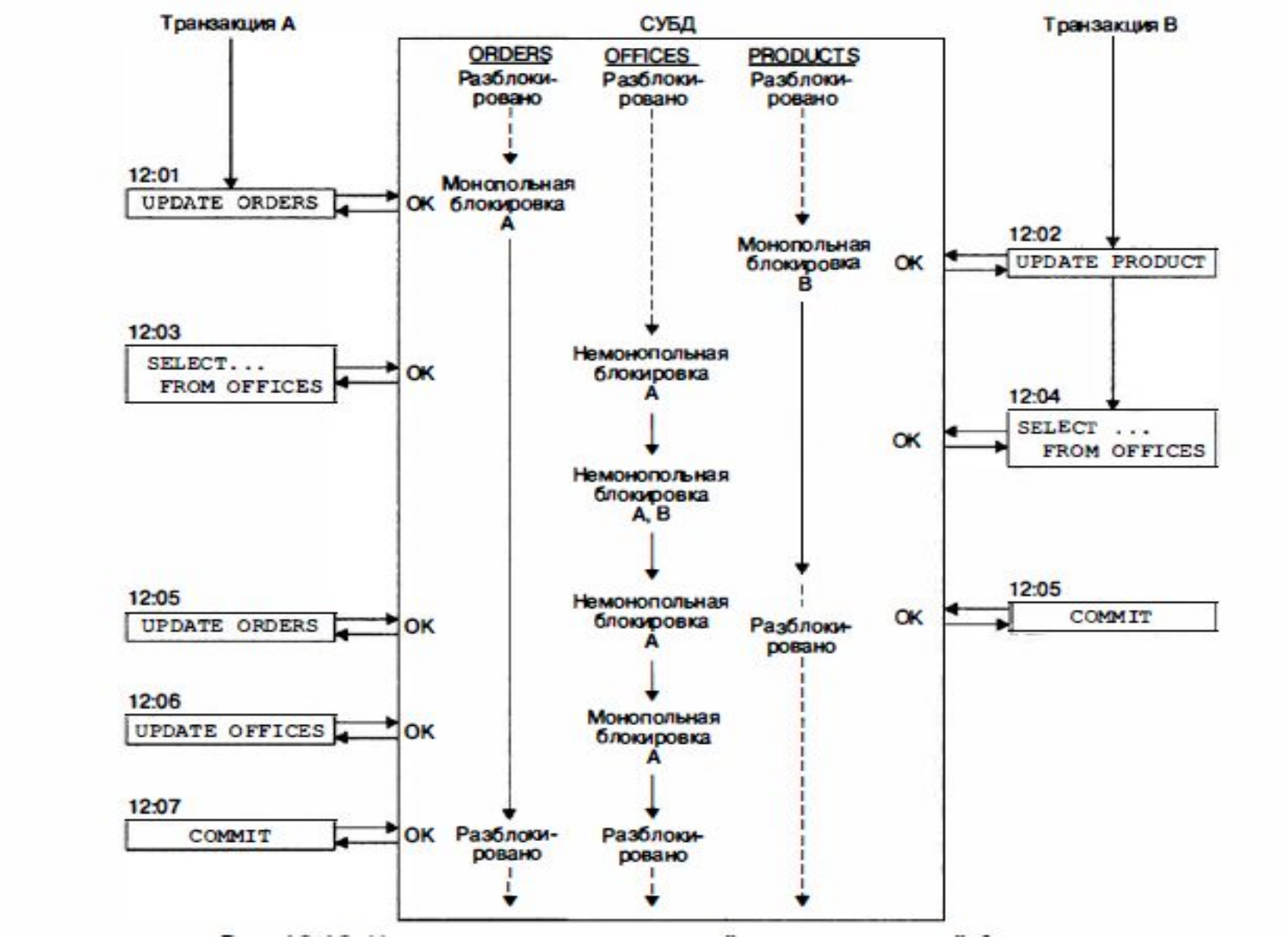
Виды блокировок

- **Монопольная**, или исключающая блокировка (exclusive lock). Когда транзакция обновляет информацию в БД, СУБД применяет исключающую блокировку. Если транзакция монопольно заблокировала какие-либо данные, другие транзакции не могут обращаться к ним ни для выборки, ни для записи.

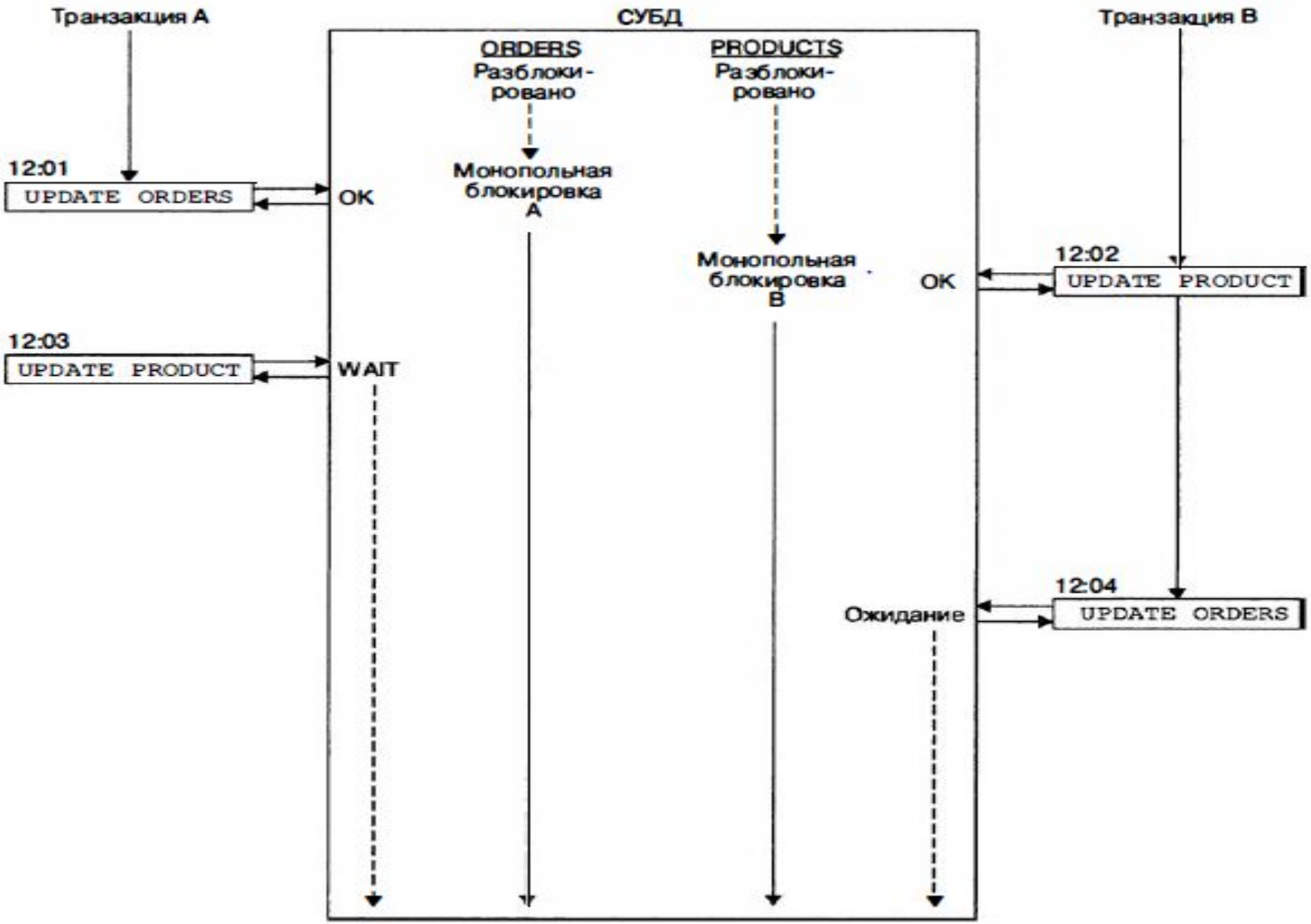
Правила применения блокировок

		Транзакция B		
		Без блокировки	Блокировка без монополизации	Монопольная блокировка
Транзакция A	Без блокировки	Да	Да	Да
	Блокировка без монополизации	Да	Да	Нет
	Монопольная блокировка	Да	Нет	Нет

Виды блокировок



Виды блокировок



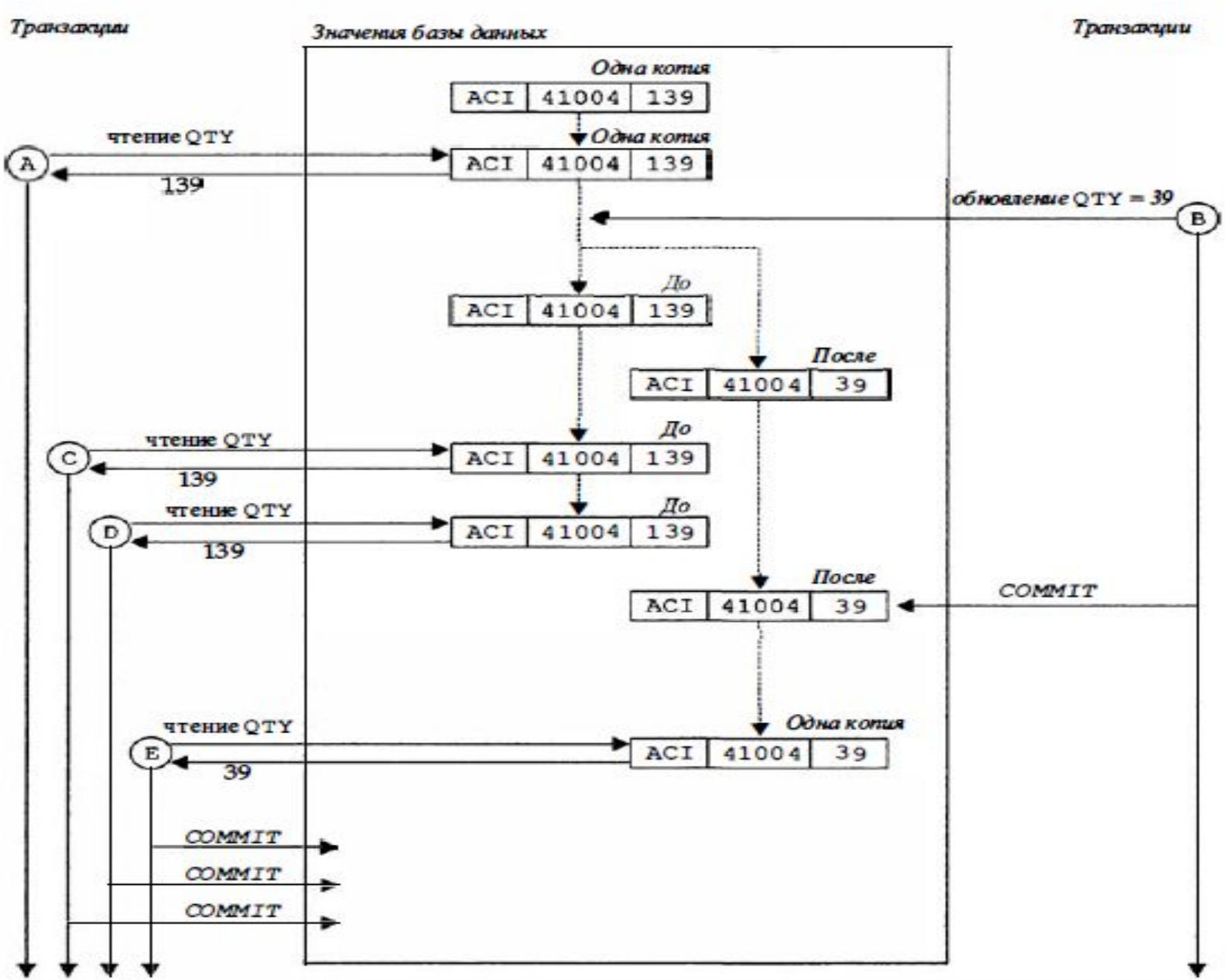
Усовершенствованные методы блокировок

- **Явная блокировка.**
- **Уровни изоляции.**
- **Параметры блокировки.**

Уровни изоляции

Уровень изоляции	Пропавшие обновления	Промежуточные данные	Несогласованные данные	Строки - призраки
SERIALIZABLE	-	-	-	-
REPEATABLE READ	-	-	-	+
READ COMMITED	-	-	+	+
READ UNCOMMITED	-	+	+	+

Архитектура управления версиями



Резюме

- В РСУБД транзакция представляет собой логическую единицу работы. Транзакция состоит из последовательности инструкций SQL, которые СУБД выполняет как одно целое.
- Инструкции SET TRANSACTION и START TRANSACTION могут использоваться для установки уровня изоляции и уровня доступа транзакций
- Инструкций SAVEPOINT создает промежуточную точку восстановления внутри транзакции.

Резюме

- Инструкция RELEASE SAVEPOINT удаляет точку сохранения и освобождает захваченные ею ресурсы.
- Инструкция COMMIT сообщает об успешном завершениии.
- Инструкция ROLLBACK предлагает СУБД отменить транзакцию и все изменения, уже внесенные в базу данных данной транзакции.
- Транзакции играют ключевую роль при восстановлении базы данных после системного сбоя.

Резюме

- Транзакции играют ключевую роль при параллельном доступе к данным в многопользовательской базе данных.
- Иногда конфликт с другой параллельной транзакцией может привести к отмене транзакции не по ее вине. Приложение должно быть готово к решению этой проблемы в случае ее возникновения.
- Одной из наиболее сложных областей использования и настройки большей большой базы данных является управление транзакциями и их влияние на производительность СУБД.

Резюме

- Многие СУБД для обработки параллельных транзакций применяют методику блокировки. Изменение параметров блокировок и инструкции явной блокировки обеспечивают возможность тонкой настройки обработки транзакций и повышения производительности баз данных
- Альтернативой блокировкам служит поддерживаемый рядом СУБД метод управления версиями

Создание базы данных

- Инструкции SELECT, INSERT, UPDATE, COMMIT, ROLLBACK, DELETE предназначены для обработки данных. Эти инструкции называются языком обработки данных или DML (Data Manipulation Language). Инструкции DML могут модифицировать информацию, хранимую в базе данных, но не могут модифицировать ее структуру.

Создание базы данных

- Для изменения структуры базы данных предназначен другой набор инструкций SQL, так называемый язык *определения данных* или DDL (Data Definition Language).

DDL

- Определить структуру новой таблицы и создать ее;
- Удалить таблицу, которая больше не нужна;
- Изменить определение существующей таблицы;
- Определить виртуальную таблицу (или представление) данных;
- Обеспечить безопасность базы данных;
- Создать индекс для ускорения доступа к таблице
- Управлять физическим размещением данных

DDL

Ядро языка определения данных образуют три команды:

- **CREATE** (создать), позволяющая определить и создать объект базы данных;
- **DROP** (удалить), служащая для удаления существующего объекта базы данных;
- **ALTER** (изменить), посредством которой можно изменить определение объекта базы данных;

Создание базы данных

CREATE DATABASE

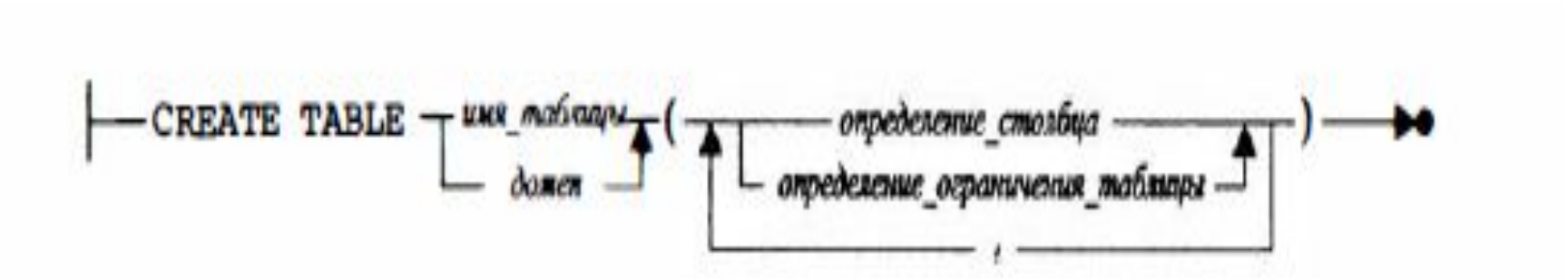
DROP DATABASE

Определение таблиц

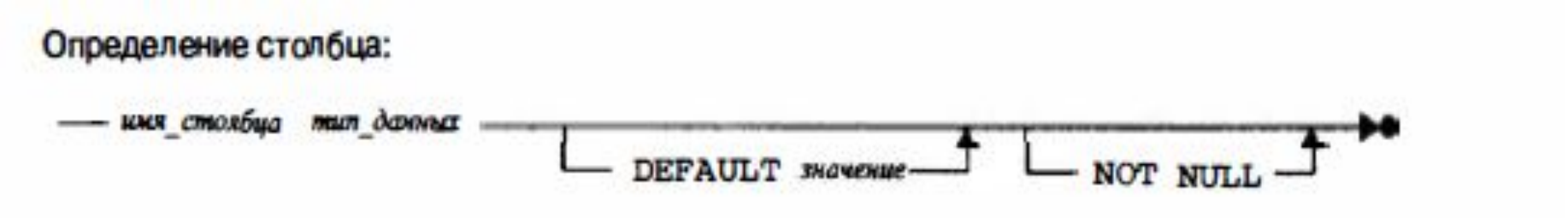
В реляционной базе данных наиболее важным элементом ее структуры является таблица

Таблица является проекцией отношения из реляционной алгебры на реальные базы данных.

Создание таблицы (CREATE TABLE)

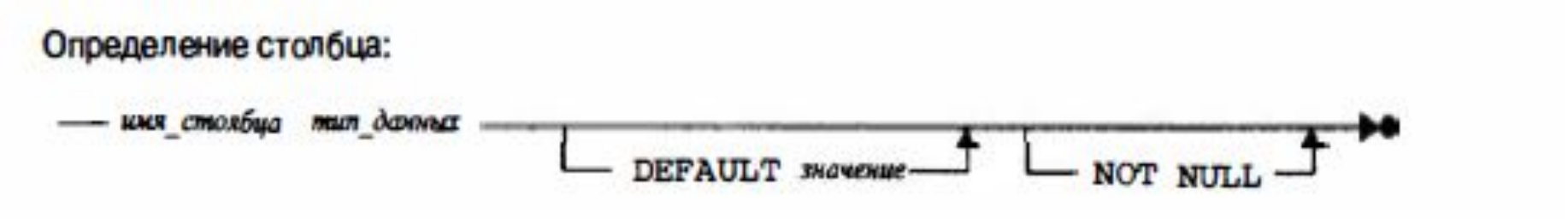


Определения столбцов



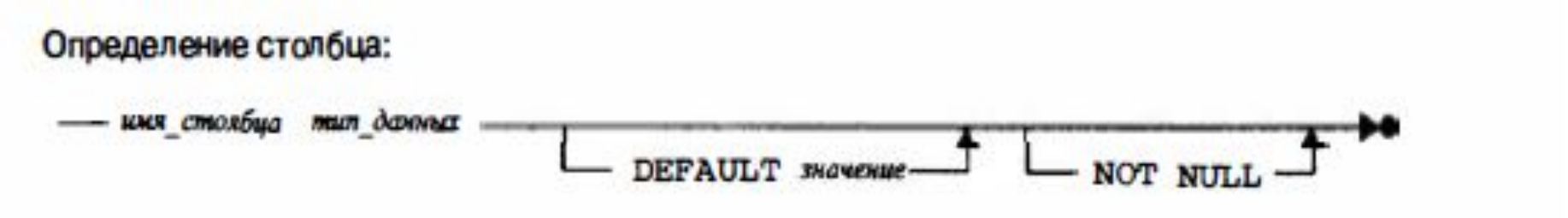
- **Имя столбца.** Используется для обращения к столбцу в инструкциях SQL. Каждый столбец в таблице должен иметь уникальное имя, но в разных таблицах имена столбцов могут совпадать.

Определения столбцов



- **Тип данных.** Указывает тип столбца. Иногда указывается доп. информация, такая как длина или число десятичных разрядов.

Определения столбцов



- **Обязательность данных.** Определяет, допускаются ли в данном столбце значения NULL.
- **Значения по умолчанию.** Это необязательное значение *по умолчанию*, которое заносится в том случае, если в инструкции INSERT для таблицы не указано значение для данного столбца.

Пример ORACLE

```
CREATE TABLE OFFICES
(
    OFFICE NUMERIC(10, 0) NOT NULL,
    CITY VARCHAR2(15)    NOT NULL,
    REGION VARCHAR2(10)  NOT NULL,
    MGR NUMERIC (10,0),
    TARGET NUMERIC(10, 2),
    SALES NUMERIC(10, 2) NOT NULL
);
```

Пример ORACLE

```
CREATE TABLE ORDERS
(
    ORDER_NUM NUMERIC(10, 0) NOT NULL,
    ORDER_DATE DATE NOT NULL,
    CUST NUMERIC(10, 0) NOT NULL,
    REP NUMERIC(10, 0),
    MFR CHAR(3) NOT NULL,
    PRODUCT CHAR(5) NOT NULL,
    QTY NUMERIC(10, 0) NOT NULL,
    AMOUNT NUMERIC(9,2) NOT NULL
);
```

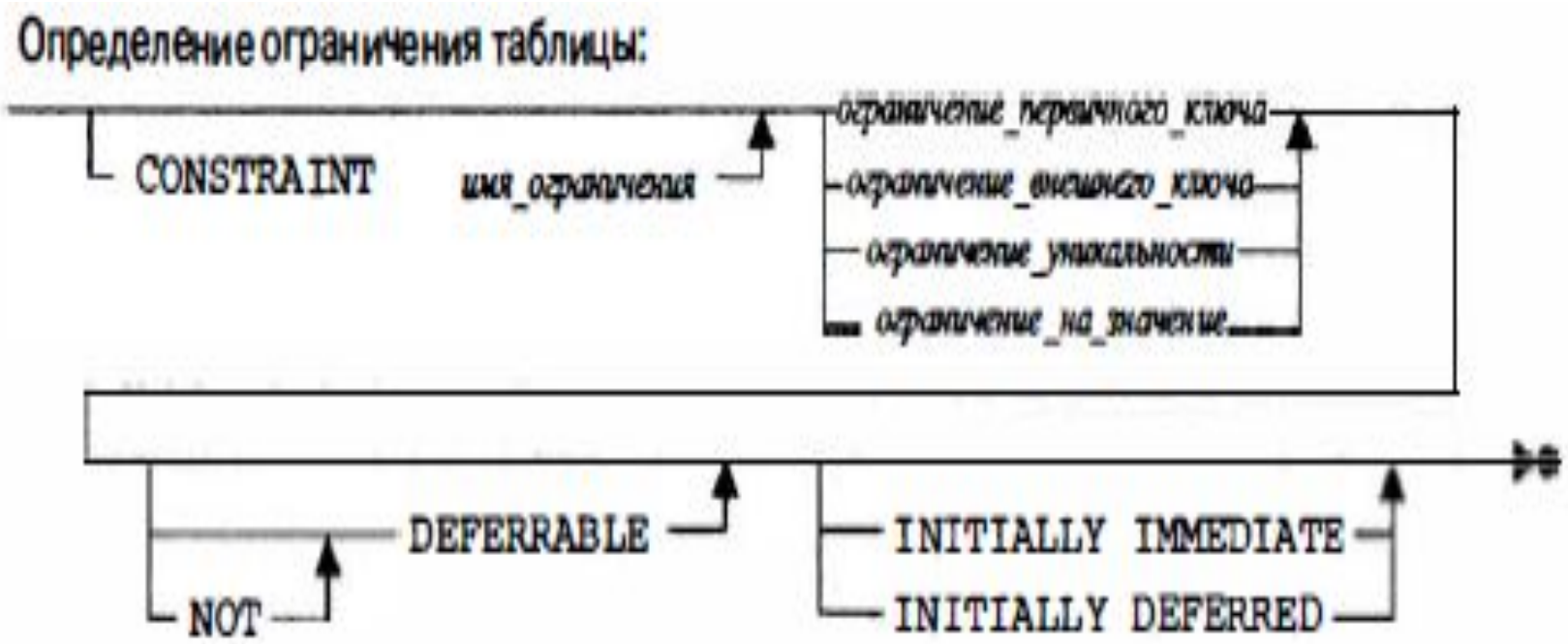
Пример ORACLE

```
CREATE TABLE ORDERS
(
    ORDER_NUM NUMERIC(10, 0) NOT NULL,
    ORDER_DATE DATE NOT NULL,
    CUST NUMERIC(10, 0) NOT NULL,
    REP NUMERIC(10, 0),
    MFR CHAR(3) NOT NULL,
    PRODUCT CHAR(5) NOT NULL,
    QTY NUMERIC(10, 0) NOT NULL,
    AMOUNT NUMERIC(9,2) NOT NULL
);
```

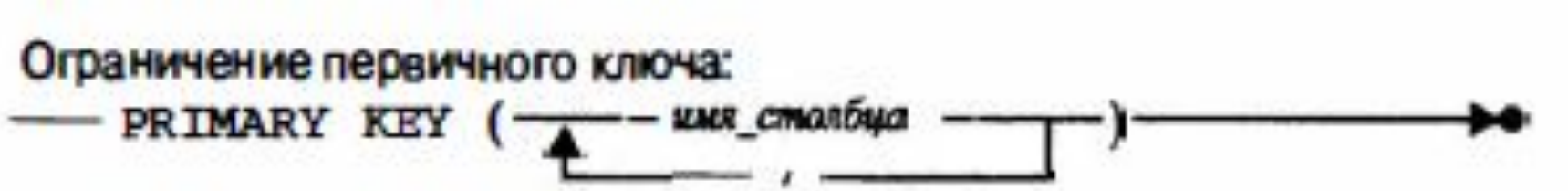
Значения по умолчанию и отсутствующие значения

```
CREATE TABLE OFFICES
(
    OFFICE NUMERIC(10,0) NOT NULL,
    CITY VARCHAR2(15)  NOT NULL,
    REGION VARCHAR2(10) NOT NULL DEFAULT
'Eastern',
    MGR NUMERIC(10,0) DEFAULT 106,
    TARGET NUMERIC(9,2) DEFAULT NULL,
    SALES NUMERIC(9,2) NOT NULL DEFAULT 0.00
);
```

Ограничения

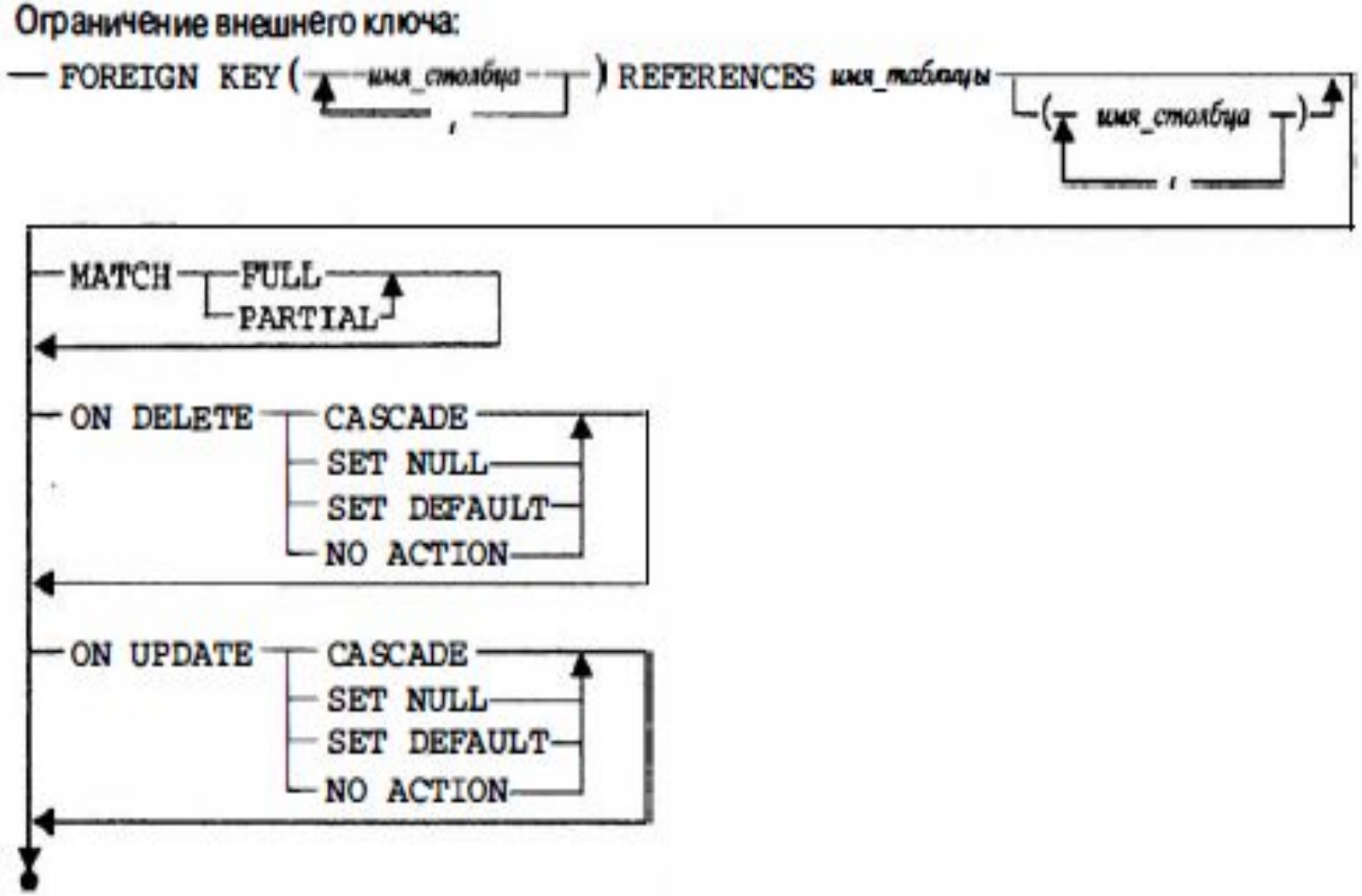


Определение первичного ключа



В предложении PRIMARY KEY задается столбец или столбцы, которые образуют первичный ключ таблицы. Этот столбец служит в качестве уникального идентификатора строк таблицы. СУБД автоматически следит, чтобы первичный ключ содержал уникальные значения. Также должно быть указано, что значение NOT NULL

Определение внешнего ключа



Определение внешнего ключа

- Стоблец или столбцы создаваемой таблицы, которые создают внешний ключ.
- Таблица, связь с которой создает внешний ключ. Это родительская таблица; Определяемая таблица в данном отношении является дочерней.
- Необязательный список имен столбцов родительской таблицы, которые соответствуют столбцам внешнего ключа определяемой таблицы. Если имена столбцов опущены, в родительской таблице обязаны быть столбцы с именами, идентичными именам столбцов во внешнем ключе.

Определение внешнего ключа

- Необязательное имя для этого отношения; оно не используется в инструкциях SQL, но может появляться в сообщениях об ошибках и потребуется в дальнейшем, если будет необходимо удалить внешний ключ;
- Как СУБД должна трактовать значения NULL в одном или нескольких столбцах внешнего ключа при связывании его со строками таблицы-предка

Определение внешнего ключа

- Необязательное правило удаления для данного отношения (CASCADE, SET NULL, SET DEFAULT, NO ACTION), которое определяет действие, предпринимаемое при удалении строки родительской строки;
- Необязательное правило обновления для данного отношения, которое определяет действие, предпринимаемое при обновлении первичного ключа в строке родительской таблицы;

Определение внешнего ключа

- Необязательное условие на значения, которое ограничивает данные в таблице так, чтобы они отвечали определенному критерию отбора.

Пример с PRIMARY и FOREIGN KEY

```
CREATE TABLE ORDERS
(
    ORDER_NUM INTEGER NOT NULL,
    ORDER_DATE DATE NOT NULL,
    CUST INTEGER NOT NULL,
    REP INTEGER,
    MFR CHAR(3) NOT NULL,
    PRODUCT CHAR(5) NOT NULL,
    QTY INTEGER NOT NULL,
    AMOUNT DECIMAL(9,2) NOT NULL,
```

Пример с PRIMARY и FOREIGN KEY

```
PRIMARY KEY (ORDER_NUM),
 CONSTRAINT PLACEDBY FOREIGN KEY (CUST)
  REFERENCES CUSTOMERS(CUST_NUM)
   ON DELETE CASCADE,
 CONSTRAINT TAKENBY FOREIGN KEY (REP)
  REFERENCES SALESREPS(EMPL_NUM)
   ON DELETE SET NULL,
 CONSTRAINT ISFOR FOREIGN KEY (MFR,
PRODUCT)
  REFERENCES PRODUCTS(MFR_ID,
PRODUCT_ID));
```

Условия уникальности

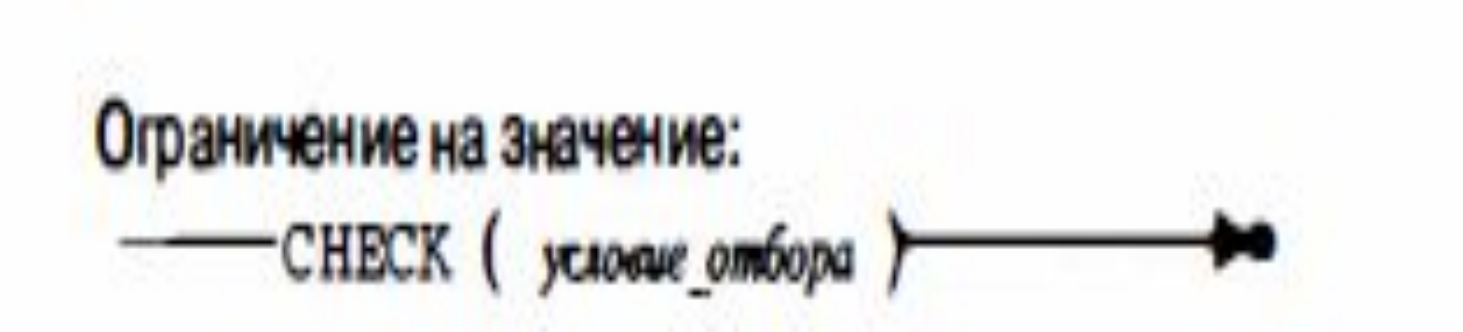
Стандарт SQL определяет, что условия уникальности также задаются в инструкции CREATE TABLE, с применением предложения UNIQUE.

Пример с UNIQUE

```
CREATE TABLE OFFICES
(    OFFICE INTEGER NOT NULL,
   CITY VARCHAR(15) NOT NULL,
   REGION VARCHAR(10) NOT NULL,
   MGR INTEGER,
   TARGET DECIMAL(9,2),
   SALES DECIMAL(9,2) NOT NULL,
 PRIMARY KEY (OFFICE),
 CONSTRAINT HASMGR
     FOREIGN KEY (MGR)
          REFERENCES SALESREPS (EMPL_NUM)
     ON DELETE SET NULL,
     UNIQUE(CITY)) ;
```

Ограничения на значения столбцов

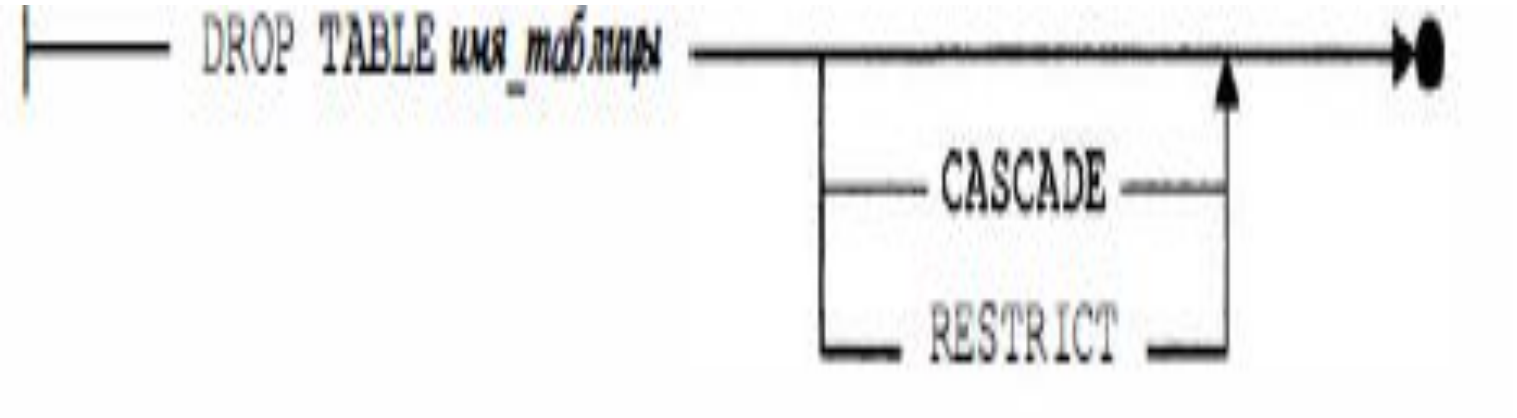
Ограничение на значения столбцов связано с инструкцией CHECK.



Пример с CHECK

```
CREATE TABLE OFFICES
    (OFFICE INTEGER NOT NULL,
      CITY VARCHAR(15) NOT NULL,
     REGION VARCHAR(10) NOT NULL,
       MGR INTEGER,
     TARGET DECIMAL(9,2),
      SALES DECIMAL(9,2) NOT NULL,
PRIMARY KEY (OFFICE),
CHECK (TARGET >= 0.0));
```

Удаление таблицы



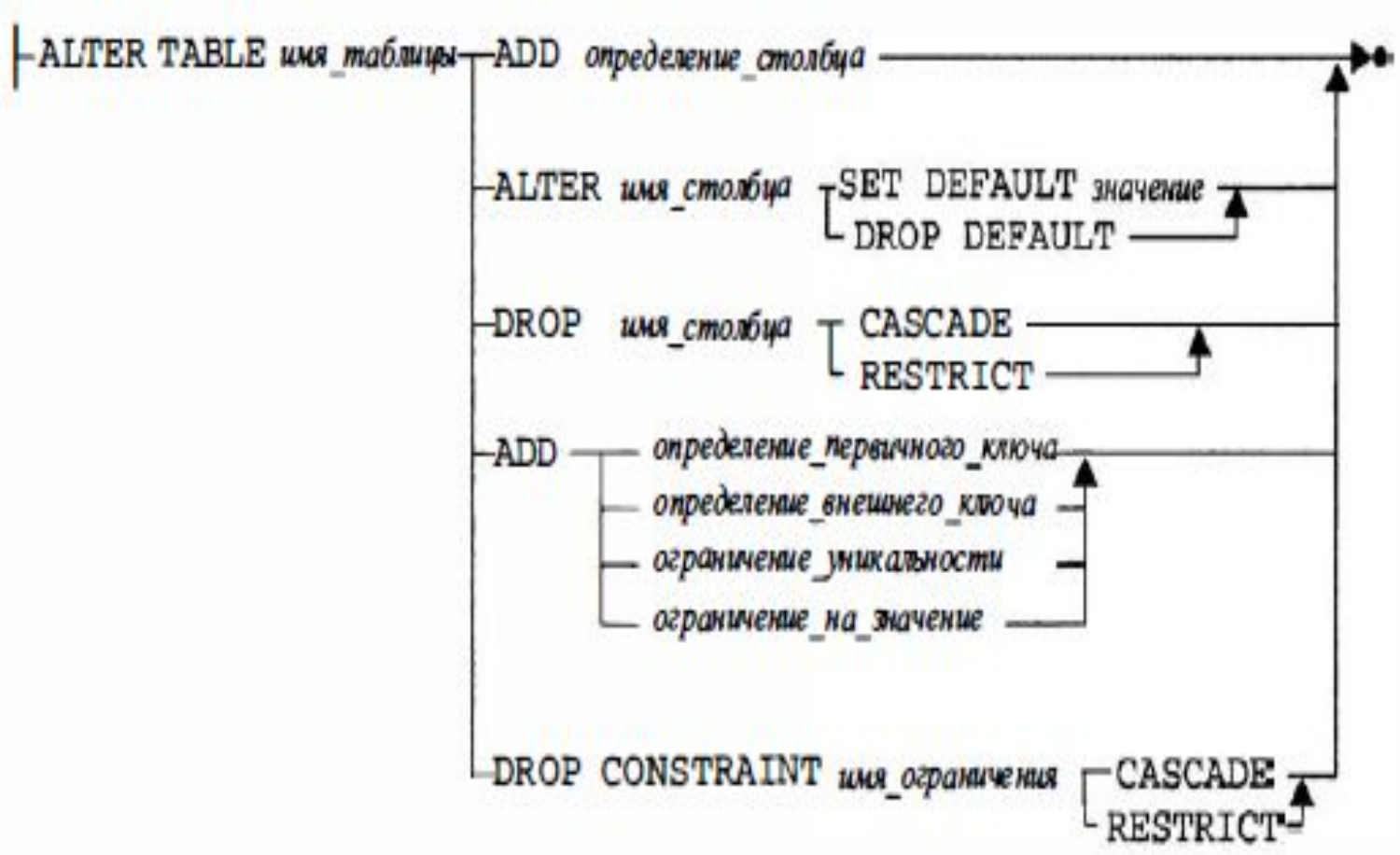
Изменение определения таблицы

- Добавить в каждую строку таблицы CUSTOMERS имя и номер телефона служащего компании – клиента, через которого поддерживается контакт, если необходимо использовать эту таблицу для связи с клиентами;
- Добавить в таблицу PRODUCTS столбец с указанием минимального количества на складе, чтобы база данных могла автоматически предупреждать о том, что запас какого-либо товара стал меньше допустимого предела.

Изменение определения таблицы

- Сделать столбец REGION таблицы OFFICES внешним ключом для вновь созданной таблицы REGIONS, первичным ключом которой является название региона;
- Удалить определение внешнего ключа для столбца CUST таблицы ORDERS, связывающего ее с таблицей CUSTOMERS, и заменить определениями двух внешних ключей, связывающих столбец CUST с двумя вновь созданными таблицами CUST_INFO и ACCOUNT_INFO

Изменение определения таблицы



Изменение определения таблицы

- Добавить в таблицу определение столбца;
- Удалить столбец из таблицы;
- Изменить значение по умолчанию для какого-либо столбца;
- Добавить или удалить первичный ключ таблицы;
- Добавить или удалить внешний ключ таблицы;
- Добавить или удалить условие уникальности;
- Добавить или удалить условие на

Добавление в таблицу столбцов

```
ALTER TABLE CUSTOMERS
ADD CONTACT_NAME VARCHAR(30);

ALTER TABLE CUSTOMERS
ADD COLUMN CONTACT_PHONE CHAR(10);

ALTER TABLE PRODUCTS
ADD MIN_QTY INTEGER NOT NULL DEFAULT 0;
```

Удаление столбцов

```
ALTER TABLE CUSTOMERS
DROP CONTACT_NAME;
```

Изменения первичных и внешних ключей

```
ALTER TABLE OFFICES
ADD CONSTRAINT INREGION
FOREIGN KEY (REGION)
REFERENCES REGIONS;

ALTER TABLE SALESREPS
    DROP CONSTRAINT WORKSIN;

ALTER TABLE OFFICES
    DROP PRIMARY KEY;
```

Основные понятия ER-диаграмм

Определение 1. ***Сущность*** - это класс однотипных объектов, информация о которых должна быть учтена в модели.

Основные понятия ER-диаграмм

http://citforum.ru/database/dblearn/dblearn08.shtml

Статья по разделу

Основные понятия ER-диаграмм

- *Определение 2*. ***Экземпляр сущности*** - это конкретный представитель данной сущности.

Например, представителем сущности "Сотрудник" может быть "Сотрудник Иванов".

Экземпляры сущностей должны быть *различимы*, т.е. сущности должны иметь некоторые свойства, уникальные для каждого экземпляра этой сущности.

Основные понятия ER-диаграмм

- *Определение 3*. ***Атрибут сущности*** - это именованная характеристика, являющаяся некоторым свойством сущности.

Наименование атрибута должно быть выражено существительным в единственном числе (возможно, с характеризующими прилагательными).

Примерами атрибутов сущности "Сотрудник" могут быть такие атрибуты как "Табельный номер", "Фамилия", "Имя", "Отчество", "Должность", "Зарплата" и т.п.

Основные понятия ER-диаграмм

Основные понятия ER-диаграмм

- *Определение 4*. ***Ключ сущности*** - это *неизбыточный* набор атрибутов, значения которых в совокупности являются *уникальными* для каждого экземпляра сущности. Неизбыточность заключается в том, что удаление любого атрибута из ключа нарушается его уникальность.

Сущность может иметь несколько различных ключей.

Основные понятия ER-диаграмм

Основные понятия ER-диаграмм

- *Определение 5*. ***Связь*** - это некоторая ассоциация между *двумя* сущностями. Одна сущность может быть связана с другой сущностью или сама с собою.

Связи позволяют по одной сущности находить другие сущности, связанные с нею.

Например, связи между сущностями могут выражаться следующими фразами - "СОТРУДНИК может иметь несколько ДЕТЕЙ", "каждый СОТРУДНИК обязан числиться ровно в одном ОТДЕЛЕ".

Графически связь изображается линией, соединяющей две сущности:

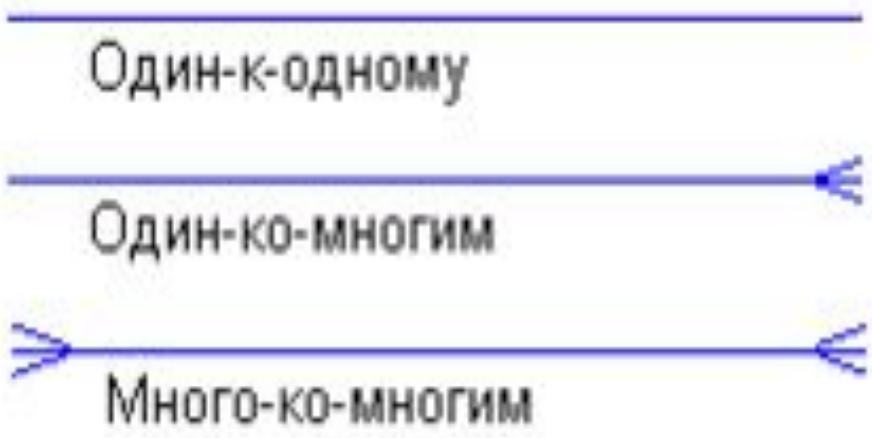
Основные понятия ER-диаграмм

Основные понятия ER-диаграмм

- Каждая связь имеет два конца и одно или два наименования. Наименование обычно выражается в неопределенной глагольной форме: "иметь", "принадлежать" и т.п. Каждое из наименований относится к своему концу связи. Иногда наименования не пишутся ввиду их очевидности.

Каждая связь может иметь один из следующих ***типов связи***:

Основные понятия ER-диаграмм



Основные понятия ER-диаграмм

- Связь типа ***один-к-одному*** означает, что один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой). Связь один-к-одному чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

Основные понятия ER-диаграмм

- Связь типа ***один-ко-многим*** означает, что один экземпляр первой сущности (левой) связан с несколькими экземплярами второй сущности (правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны "один") называется ***родительской***, правая (со стороны "много") - ***дочерней***

Основные понятия ER-диаграмм

- Связь типа ***много-ко-многим*** означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности.

Основные понятия ER-диаграмм

Каждая связь может иметь одну из двух ***модальностей связи***:

Основные понятия ER-диаграмм

- Модальность "***может***" означает, что экземпляр одной сущности *может быть связан* с одним или несколькими экземплярами другой сущности, *а может быть и не связан* ни с одним экземпляром.
- Модальность "***должен***" означает, что экземпляр одной сущности *обязан быть связан не менее чем с одним* экземпляром другой сущности.
- Связь может иметь *разную модальность* с разных концов

Основные понятия ER-диаграмм

При разработке ER-моделей мы должны получить следующую информацию о предметной области:

- Список сущностей предметной области.
- Список атрибутов сущностей.
- Описание взаимосвязей между сущностями.

Основные понятия ER-диаграмм

Предположим, что перед нами стоит задача разработать информационную систему по заказу некоторой оптовой торговой фирмы. В первую очередь мы должны изучить предметную область и процессы, происходящие в ней.

Основные понятия ER-диаграмм

Например, выяснилось, что проектируемая система должна выполнять следующие действия:

- Хранить информацию о покупателях.
- Печатать накладные на отпущенные товары.
- Следить за наличием товаров на складе.

Основные понятия ER-диаграмм

Выделим все существительные в этих предложениях - это будут потенциальные кандидаты на сущности и атрибуты, и проанализируем их (непонятные термины будем выделять знаком вопроса):

- *Покупатель* - явный кандидат на сущность.
- *Накладная* - явный кандидат на сущность.
- *Товар* - явный кандидат на сущность
- *(?)Склад* - а вообще, сколько складов имеет фирма? Если несколько, то это будет кандидатом на новую сущность.
- *(?)Наличие товара* – это, скорее всего, атрибут, но атрибут какой сущности?

Основные понятия ER-диаграмм

Основные понятия ER-диаграмм

Формируем атрибуты сущности:

- Каждый покупатель является юридическим лицом и имеет наименование, адрес, банковские реквизиты.
- Каждый товар имеет наименование, цену, а также характеризуется единицами измерения.
- Каждая накладная имеет уникальный номер, дату выписки, список товаров с количествами и ценами, а также общую сумму накладной. Накладная выписывается с определенного склада и на определенного покупателя.

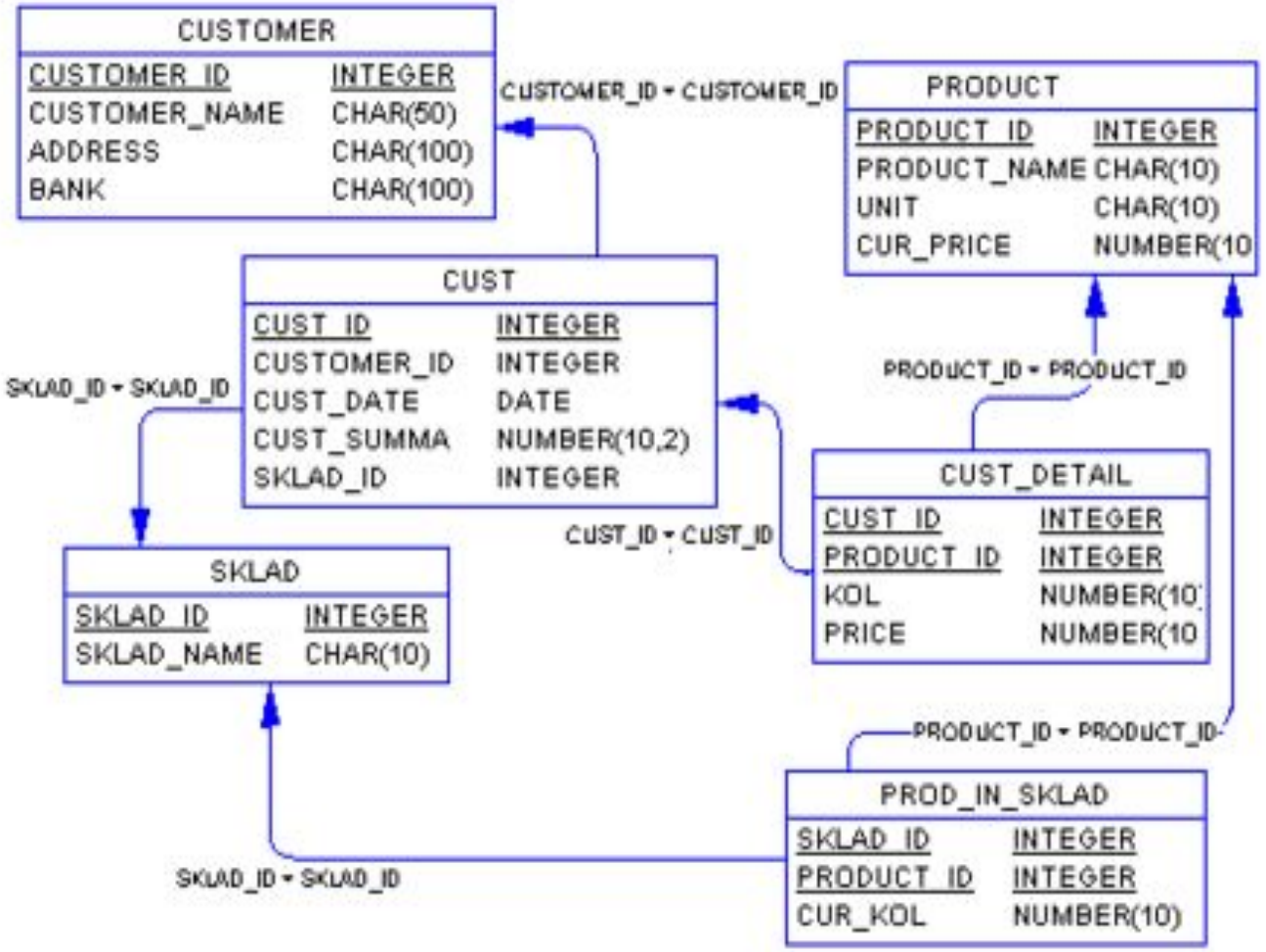
И т.п.

Основные понятия ER-диаграмм

Основные понятия ER-диаграмм

Разработанный выше пример ER-диаграммы является примером ***концептуальной диаграммы***. Это означает, что диаграмма *не учитывает* особенности конкретной СУБД. По данной концептуальной диаграмме можно построить ***физическую диаграмму***, которая уже будут учитываться такие особенности СУБД, как допустимые типы и наименования полей и таблиц, ограничения целостности и т.п

Основные понятия ER-диаграмм



Нормальные формы

Статья - https://habrahabr.ru/post/254773/

Нормальные формы

- Используемые термины

Атрибут — свойство некоторой сущности. Часто называется полем таблицы.

Домен атрибута — множество допустимых значений, которые может принимать атрибут.

Кортеж — конечное множество взаимосвязанных допустимых значений атрибутов, которые вместе описывают некоторую сущность (строка таблицы).

Отношение — конечное множество кортежей (таблица).

Схема отношения — конечное множество атрибутов, определяющих некоторую сущность. Иными словами, это структура таблицы, состоящей из конкретного набора полей.

Нормальные формы

Проекция — отношение, полученное из заданного путём удаления и (или) перестановки некоторых атрибутов.

Функциональная зависимость между атрибутами (множествами атрибутов) X и Y означает, что для любого допустимого набора кортежей в данном отношении: если два кортежа совпадают по значению X, то они совпадают по значению Y. Например, если значение атрибута «Название компании» — Canonical Ltd, то значением атрибута «Штаб-квартира» в таком кортеже всегда будет Millbank Tower, London, United Kingdom. Обозначение: {X} -> {Y}.

Нормальная форма — требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц).

Метод нормальных форм (НФ) состоит в сборе информации о объектах решения задачи в рамках одного отношения и последующей декомпозиции этого отношения на несколько взаимосвязанных отношений на основе процедур нормализации отношений.

Нормальные формы

Цель нормализации: исключить избыточное дублирование данных, которое является причиной аномалий, возникших при добавлении, редактировании и удалении кортежей(строк таблицы).

Аномалией называется такая ситуация в таблице БД, которая приводит к противоречию в БД либо существенно усложняет обработку БД. Причиной является излишнее дублирование данных в таблице, которое вызывается наличием функциональных зависимостей от не ключевых атрибутов.

Аномалии-модификации проявляются в том, что изменение одних данных может повлечь просмотр всей таблицы и соответствующее изменение некоторых записей таблицы.

Аномалии-удаления — при удалении какого либо кортежа из таблицы может пропасть информация, которая не связана на прямую с удаляемой записью.

Аномалии-добавления возникают, когда информацию в таблицу нельзя поместить, пока она не полная, либо вставка записи требует дополнительного просмотра таблицы.

Первая нормальная форма

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

Фирма	Модели
BMW	M5, X5M, M1
Nissan	GT-R

Первая нормальная форма

Фирма	Модели
BMW	M5
Nissan	GT-R
BMW	X5M
BMW	M1

Вторая нормальная форма

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа(ПК).

Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.

Вторая нормальная форма

Модель	Фирма	Цена	Скидка
M5	BMW	550000	5%
X5M	BMW	750000	5%
M1	BMW	250000	5%
GT-R	Nissan	500000	10%

Таблица находится в первой нормальной форме, но не во второй. Цену машины зависит от модели и фирмы. Скидка зависят от фирмы, то есть зависимость от первичного ключа неполная. Исправляется это путем декомпозиции на два отношения, в которых не ключевые атрибуты зависят от ПК.

Вторая нормальная форма

Модель	Фирма	Цена
M5	BMW	550000
X5M	BMW	750000
M1	BMW	250000
GT-R	Nissan	500000

Фирма	Скидка
BMW	5%
Nissan	10%

Третья нормальная форма

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

Третья нормальная форма

Модель	Магазин	Телефон
BMW	Риал-авто	87-33-98
Audi	Риал-авто	87-33-98
Nissan	Некст-авто	94-54-12

Третья нормальная форма

Модель	Магазин	Телефон
BMW	Риал-авто	87-33-98
Audi	Риал-авто	87-33-98
Nissan	Некст-авто	94-54-12

Таблица находится во 2НФ, но не в 3НФ.
В отношении существуют следующие функциональные зависимости:
Модель → Магазин, Магазин → Телефон, Модель → Телефон.
Зависимость Модель → Телефон является транзитивной, следовательно, отношение не находится в 3НФ.

Третья нормальная форма

Модель	Магазин
BMW	Риал-авто
Audi	Риал-авто
Nissan	Некст-авто

Магазин	Телефон
Риал-авто	87-33-98
Некст-авто	94-54-12

Нормальная форма Бойса-Кодда (НФБК) (частная форма третьей нормальной формы)

Определение 3НФ не совсем подходит для следующих отношений:
1) отношение имеет две или более потенциальных ключа;
2) два и более потенциальных ключа являются составными;
3) они пересекаются, т.е. имеют хотя бы один атрибут.

Для отношений, имеющих один потенциальный ключ (первичный), НФБК является 3НФ.

Отношение находится в НФБК, когда каждая нетривиальная и неприводимая слева функциональная зависимость обладает потенциальным ключом в качестве детерминанта.

BCNF

Номер стоянки	Время начала	Время окончания	Тариф
1	09:30	10:30	Бережливый
1	11:00	12:00	Бережливый
1	14:00	15:30	Стандарт
2	10:00	12:00	Премиум-B
2	12:00	14:00	Премиум-B
2	15:00	18:00	Премиум-A

возможны следующие составные первичные ключи: {Номер стоянки, Время начала}, {Номер стоянки, Время окончания}, {Тариф, Время начала}, {Тариф, Время окончания}

BCNF

Отношение находится в 3НФ. Требования второй нормальной формы выполняются, так как все атрибуты входят в какой-то из потенциальных ключей, а неключевых атрибутов в отношении нет. Также нет и транзитивных зависимостей, что соответствует требованиям третьей нормальной формы. Тем не менее, существует функциональная зависимость Тариф → Номер стоянки, в которой левая часть (детерминант) не является потенциальным ключом отношения, то есть отношение не находится в нормальной форме Бойса — Кодда.

Недостатком данной структуры является то, что, например, по ошибке можно приписать тариф «Бережливый» к бронированию второй стоянки, хотя он может относиться только к первой стоянки

BCNF

Тариф	Номер стоянки	Имеет льготы
Бережливый	1	Да
Стандарт	1	Нет
Премиум-А	2	Да
Премиум-В	2	Нет

Тариф	Время начала	Время окончания
Бережливый	09:30	10:30
Бережливый	11:00	12:00
Стандарт	14:00	15:30
Премиум-В	10:00	12:00
Премиум-В	12:00	14:00
Премиум-А	15:00	18:00

Домашнее задание

1. Описать вашу БД для курсовой, как ее можно применять, какие задачи она должна выполнять.
2. Выделить Атрибуты, Сущности их связи. Построить логическую E-R диаграмму.
3. Построить физическую E-R диаграмму для вашей базы данных.
4. Привести вашу схему к нормальной форме Бойса - Кодда
5. Написать DLL скрипты для генерации ваших таблиц. В них должны быть различные ограничения, первичные ключи, внешние ключи. Также привести примеры с ALTER TABLE.

Домашнее задание

6. Написать DML скрипты для INSERT, DELETE, UPDATE.

7. Привести пример транзакции. Описать словами, почему данная DML – последовательность должна быть транзакцией.