Базы данных

Лекция 3

Домашнее задание

Вывести номер заказа, описание продукта и имя сотрудник, принявшего заказ. В выводе должно быть 3 столбца с названиями (ORDER_NUM, DESCRIPTION, NAME).

Домашнее задание

Вывести номер заказа, описание продукта и имя сотрудник, принявшего заказ. В выводе должно быть 3 столбца с названиями (ORDER_NUM, DESCRIPTION, NAME).

```
SELECT ORDER_NUM, DESCRIPTION, NAME
FROM PRODUCTS
  INNER JOIN ORDERS
    ON  PRODUCTS.product_id = ORDERS.product
    AND PRODUCTS.mfr_id = ORDERS.mfr
  INNER JOIN SALESREPS
    ON ORDERS.REP = SALESREPS.EMPL_NUM;
```

Домашнее задание

Вывести номер заказа, имя сотрудника, принявшего заказ, а также город, в котором работает сотрудник. Если для сотрудника офис не определен, то вывести ‘N/A’. В выводе должно быть 3 столбца с названиями (ORDER_NUM, NAME, OFFICE).

Домашнее задание

Вывести номер заказа, имя сотрудника, принявшего заказ, а также город, в котором работает сотрудник. Если для сотрудника офис не определен, то вывести ‘N/A’. В выводе должно быть 3 столбца с названиями (ORDER_NUM, NAME, OFFICE).

```
SELECT ORDER_NUM, NAME, CASE WHEN REP_OFFICE IS NULL THEN 'N/A' ELSE CITY END AS OFFICE
FROM ORDERS
  INNER JOIN SALESREPS
    ON ORDERS.REP = SALESREPS.EMPL_NUM
  LEFT JOIN OFFICES
    ON SALESREPS.REP_OFFICE = OFFICES.OFFICE;
```

Домашнее задание

Вывести номер заказа, дату заказа, название компании и ее кредитный лимит для тех заказов, которые были сделаны в 2007 году и для тех компаний, чей кредитный лимит менее 53000$. В выводе должно быть 4 столбца с названиями (ORDER_NUM, ORDER_DATE, COMPANY, CREDIT_LIMIT).

Домашнее задание

Вывести номер заказа, дату заказа, название компании и ее кредитный лимит для тех заказов, которые были сделаны в 2007 году и для тех компаний, чей кредитный лимит менее 53000$. В выводе должно быть 4 столбца с названиями (ORDER_NUM, ORDER_DATE, COMPANY, CREDIT_LIMIT).

```
SELECT ORDER_NUM, ORDER_DATE, COMPANY, CREDIT_LIMIT
FROM ORDERS
  INNER JOIN CUSTOMERS
    ON ORDERS.CUST = CUSTOMERS.CUST_NUM
WHERE EXTRACT(YEAR FROM ORDER_DATE) = 2007
AND CREDIT_LIMIT < 53000;
```

Домашнее задание

Вывести номер заказа, имя сотрудника, принявшего заказа, а также отношение суммы заказа к квоте сотрудника, в процентах. Если квота не определена, то вывести 0. В выводе должно быть 3 столбца с названиями (ORDER_NUM, NAME, PROC_OF_QUOTA).

Домашнее задание

Вывести номер заказа, имя сотрудника, принявшего заказа, а также отношение суммы заказа к квоте сотрудника, в процентах. Если квота не определена, то вывести 0. В выводе должно быть 3 столбца с названиями (ORDER_NUM, NAME, PROC_OF_QUOTA).

```
SELECT ORDER_NUM, NAME,
COALESCE(AMOUNT/QUOTA, 0.0)* 100 AS
PROC_OF_QUOTA
FROM ORDERS
  INNER JOIN SALESREPS
    ON ORDERS.REP = SALESREPS.EMPL_NUM;
```

Домашнее задание

Вывести город, в котором расположен офис, и кол-во сотрудников в офисе для всех офисов, в которых кол-во сотрудников не более 2. Если у сотрудника офис не определен, то название города должно быть ‘N/A’ и такие сотрудники также должны учитываться при выводе запроса. В выводе должно быть 2 столбца с названиями (CITY, cnt_city).

Домашнее задание

```
SELECT CASE WHEN REP_OFFICE IS NULL THEN 'N/A'
ELSE CITY END AS city, COUNT(*) AS cnt_city
FROM SALESREPS
  LEFT JOIN OFFICES
    ON SALESREPS.REP_OFFICE = OFFICES.OFFICE
GROUP BY REP_OFFICE, CITY
HAVING COUNT(*) <= 2;
```

Домашнее задание

Вывести номер заказа, идентификатор продукта в заказе, сумму заказа и описание продуктов для всех тех заказов, количество в заказе которых меньше среднего количества по всем заказам, а также для всех тех продуктов, количество которых на складе не меньше среднего количества по всем продуктам. В выводе должно быть 3 столбца (ORDER_NUM, MFR, PRODUCT, DESCRIPTION, AMOUNT).

Домашнее задание

```
SELECT ORDER_NUM, MFR, PRODUCT, DESCRIPTION,
AMOUNT
FROM ORDERS
  INNER JOIN PRODUCTS
    ON  ORDERS.PRODUCT = PRODUCTS.PRODUCT_ID
    AND ORDERS.MFR = PRODUCTS.MFR_ID
WHERE ORDERS.QTY < (SELECT AVG(QTY) FROM
ORDERS)
AND PRODUCTS.QTY_ON_HAND >= (SELECT
AVG(QTY_ON_HAND) FROM PRODUCTS);
```

Домашнее задание

Для каждого сотрудника, нанятого до 1 января 2008 года, по каждому продукту, подсчитать количество уникальных дат, в которых был сделан заказ. Вывести все строки, для которых количество таких дат, не менее двух. В выводе должно быть 3 столбца (NAME, DESCRIPTION, COUNT_OF_DATES).

Домашнее задание

```
SELECT SALESREPS.NAME, PRODUCTS.DESCRIPTION,
COUNT(DISTINCT ORDER_DATE) AS COUNT_OF_DATES
FROM ORDERS
  INNER JOIN PRODUCTS
    ON  ORDERS.MFR = PRODUCTS.MFR_ID
    AND ORDERS.PRODUCT = PRODUCTS.PRODUCT_ID
  INNER JOIN SALESREPS
    ON ORDERS.REP = SALESREPS.EMPL_NUM
WHERE HIRE_DATE < TO_DATE('2008.01.01',
'yyyy.mm.dd')
GROUP BY SALESREPS.EMPL_NUM, SALESREPS.NAME,
PRODUCTS.DESCRIPTION
HAVING COUNT(DISTINCT ORDER_DATE) >= 2;
```

Домашнее задание

Вывести описания тех продуктов, для которых существуют заказы, сделанные для компаний, которые обслуживаются сотрудником Paul Cruz. В выводе должен быть 1 столбец (DESCRIPTION).

Домашнее задание

```
SELECT DESCRIPTION
FROM PRODUCTS
  INNER JOIN ORDERS
    ON  PRODUCTS.PRODUCT_ID = ORDERS.PRODUCT
    AND PRODUCTS.MFR_ID = ORDERS.MFR
WHERE ORDERS.CUST IN (SELECT CUST_NUM
                      FROM CUSTOMERS
                        INNER JOIN SALESREPS
                          ON CUSTOMERS.CUST_REP =
SALESREPS.EMPL_NUM
                      WHERE SALESREPS.NAME = 'Paul Cruz');
```

Домашнее задание

Вывести описание товаров, для которых имеются заказы на сумму менее 35000, кроме тех заказов, которые были обработаны сотрудником Sue Smith. В выводе должен быть 1 столбец(DESCRIPTION).

Домашнее задание

Вывести описание товаров, для которых имеются заказы на сумму менее 35000, кроме тех заказов, которые были обработаны сотрудником Sue Smith. В выводе должен быть 1 столбец(DESCRIPTION).

```
SELECT DISTINCT PRODUCTS.DESCRIPTION
FROM PRODUCTS
  INNER JOIN ORDERS
    ON  PRODUCTS.PRODUCT_ID = ORDERS.PRODUCT
    AND PRODUCTS.MFR_ID = ORDERS.MFR
  INNER JOIN SALESREPS
    ON ORDERS.REP = SALESREPS.EMPL_NUM
WHERE ORDERS.AMOUNT < 35000
AND SALESREPS.NAME <> 'Sue Smith';
```

Домашнее задание

Пусть если кредитный лимит для компании более 60000$, то ее рейтинг ‘A’, если больше 45000$, но менее 60000$, то – ‘B’, если больше 25000$, но менее 45000$, то – 'C', иначе – ‘D’. Вывести сумму заказов по каждому из кредитных рейтингов. В выводе должно быть 2 столбца (CREDIT_RATING, SUM_OF_AMOUNT)

Домашнее задание

```
SELECT CASE WHEN CREDIT_LIMIT > 60000 THEN 'A'
            WHEN CREDIT_LIMIT > 45000 THEN 'B'
            WHEN CREDIT_LIMIT > 25000 THEN 'C'
            ELSE 'D' END AS CREDIT_RATING,
  SUM(AMOUNT) AS SUM_OF_AMOUNT
FROM CUSTOMERS
  INNER JOIN ORDERS
    ON CUSTOMERS.CUST_NUM = ORDERS.CUST
GROUP BY CASE WHEN CREDIT_LIMIT > 60000 THEN 'A'
            WHEN CREDIT_LIMIT > 45000 THEN 'B'
            WHEN CREDIT_LIMIT > 25000 THEN 'C'
            ELSE 'D' END ;
```

Обновление данных

- Внесение изменений в базу данных
- Целостность данных
- Обработка транзакций

Внесение изменений в БД

- INSERT
- UPDATE
- DELETE
- MERGE

Добавление новых данных

- Однострочная инструкция INSERT
- Многострочная инструкция INSERT
- Пакетная загрузка

Добавление новых данных

- Если вы принимаете на работу нового служащего, в таблицу SALESREPS необходимо добавить новую строку с данными о нем.
- Если служащий заключает договор с новым клиентом, в таблицу CUSTOMERS должна быть добавлена новая строка, представляющего этого клиента
- Если клиент делает заказ, в таблицу ORDERS требуется добавить новую строку, содержащую информацию об этом заказе

Однострочная инструкция INSERT

Имя	Henry Jacobsen
Возраст	36
Идентификатор	111
Должность	Sales Mgr
Офис	Атланта (13)
Дата приема	25.07.2008
Личный план продаж	Еще не установлен
Объем продаж на текущую дату	$0.00

Однострочная инструкция INSERT

Имя	Henry Jacobsen
Возраст	36
Идентификатор	111
Должность	Sales Mgr
Офис	Атланта (13)
Дата приема	25.07.2008
Личный план продаж	Еще не установлен
Объем продаж на текущую дату	$0.00

```
INSERT INTO SALESREPS (NAME, AGE, EMPL_NUM, SALES, TITLE, HIRE_DATE,
                       REP_OFFICE)
VALUES ('Henry Jacobsen', 36, 111, 0.00, 'Sales Mgr', '2008-07-25', 13);
```

Однострочная инструкция INSERT

Добавить информацию о новом клиенте и заказе для служащего Якобсена

```
INSERT INTO CUSTOMERS(COMPANY, CUST_NUM,
CREDIT_LIMIT, CUST_REP)
VALUES (‘InterCorp’, 2126, 15000.00, 111);

INSERT INTO ORDERS (AMOUNT, MFR, PRODUCT,
QTY, ORDER_DATE, ORDER_NUM, CUST, REP)
VALUES(2340.00, ‘ACI’, ‘41004’, 20,
CURRENT_DATE, 113069, 2126, 111);
```

Вставка значений NULL

При добавлении в таблицу новой строки данных всем столбцам, имена которых отсутствуют в списке столбцов инструкции INSERT, автоматически присваивается значение NULL.

INSERT INTO SALESREPS (NAME, AGE, EMPL_NUM, SALES, TITLE, HIRE_DATE, REP_OFFICE)

VALUES (‘Henry Jacobsen’, 36, 111, 0.00, ‘Sales Mgr’, ‘2008-07-25’, 13)

Опущены столбцы QUOTA и MANAGER.

Вставка всех столбцов

Для удобства в SQL разрешается не исключать список столбцов в инструкцию INSERT.

```
INSERT INTO SALESREPS
VALUES (111, ‘Henry Jacobsen’, 36, 13, ‘Sales Mgr’,
‘2008-07-25’, NULL, NULL, 0.00)
```

Многострочная инструкция INSERT

```
INSERT INTO OLDORDERS (ORDER_NUM, ORDER_DATE, AMOUNT)
    SELECT ORDER_NUM, ORDER_DATE, AMOUNT
    FROM ORDERS
    WHERE ORDER_DATE < ‘2008-01-01’
```

Удаление существующих данных

- Если клиент отменяет заказ, необходимо удалить соответствующую строку из таблицы ORDERS.
- Если служащий увольняется из компании, должна быть удалена соответствующая строка из таблицы SALESREPS.
- Если ликвидируется офис, необходимо удалить соответствующую строку из таблицы OFFICES; в случае, когда служащие этого офиса увольняются, их строки в таблице SALESREPS также должны быть удалены; если служащие переводятся в другой офис, то соответствующие значения в столбце OFFICE

Инструкция DELETE

Инструкция DELETE удаляет выбранные записи из одной таблицы.

Предположим, что недавно принятый на работу Генри Якобсен решил уволиться из компании. Вот инструкция DELETE, удаляющая относящуюся к служащему строку из таблицы SALESREPS.

Инструкция DELETE

Удалить информацию о Генри Якобсене из базы данных

```
DELETE FROM SALESREPS
WHERE NAME = ‘Henry Jacobsen’;
```

Инструкция DELETE

Удалить информацию о Генри Якобсене из базы данных

```
DELETE FROM SALESREPS
WHERE NAME = ‘Henry Jacobsen’;
```

Удалить все заказы компании InterCorp (2126)

```
DELETE FROM ORDERS
WHERE CUST = 2126;
```

Инструкция DELETE

Удалить данные о всех клиентах, обслуживаемых Биллом Адамсом, Мери Джонс и Дэном Робертсом

```
DELETE FROM CUSTOMERS
WHERE CUST_REP IN (105, 109, 101);
```

Удалить данные о всех служащих, принятых на работу до июля 2006 года и еще не имеющих личного плана

```
DELETE FROM SALESREPS
WHERE HIRE_DATE < TO_DATE('20060701',
'yyyymmdd') AND QUOTA IS NULL;
```

Удаление всех строк

```
DELETE FROM ORDERS;
```

Хотя в результате выполнения приведенной инструкции DELETE таблица ORDERS становится пустой, из базы данных она не удаляется. Определение таблицы ORDERS и ее столбцов остается в базе данных.

Инструкция DELETE с подзапросом

Удалить все заказы, принятые Сью Смит

Ошибка:

```
DELETE FROM ORDERS, SALESREPS
WHERE REP = EMPL_NUM
AND NAME = ‘Sue Smith’;
```

Верно:

```
DELETE FROM ORDERS
WHERE REP = ( SELECT EMPL_NUM
              FROM SALESREPS
              WHERE NAME = ‘Sue Smith’);
```

Инструкция DELETE с подзапросом

Удалить данные о всех клиентах, обслуживаемых служащими, у которых фактический объем продаж меньше 80 процентов их плана.

Инструкция DELETE с подзапросом

Удалить данные о всех клиентах, обслуживаемых служащими, у которых фактический объем продаж меньше 80 процентов их плана.

```
DELETE FROM CUSTOMERS
WHERE CUST_REP IN (SELECT EMPL_NUM
                   FROM SALESREPS
                   WHERE SALES < (.8 * QUOTA));
```

Инструкция DELETE с подзапросом

Удалить данные о всех служащих, у которых сумма текущих заказов меньше двух процентов их личного плана.

```
DELETE FROM SALESREPS
WHERE (.02 * QUOTA) > (SELECT SUM(AMOUNT)
                       FROM ORDERS
                   WHERE REP = EMPL_NUM);
```

Инструкция DELETE с подзапросом

Удалить данные о всех клиентах, которые не делали заказов с 10 ноября 2007 года

```
DELETE FROM CUSTOMERS
WHERE NOT EXISTS (SELECT *
                  FROM ORDERS
                   WHERE CUST = CUST_NUM
                 AND ORDER_DATE > ‘2007-11-10’)
```

Обновление существующих данных

- Если клиент изменяет количество заказанного товара, в соответствующей строке таблицы ORDERS должен быть обновлен столбец QTY.
- Если руководитель переходит из одного офиса в другой, столбец MGR таблицы OFFICES и столбец REP_OFFICE таблицы SALESREPS необходимо обновить, чтобы отобразить новое назначение
- Если личные планы продаж в нью-йоркском офисе увеличиваются на пять процентов, значения столбцов QUOTA в соответствующих

Обновление существующих данных

UPDATE <имя_таблицы>

SET <имя_столбца = выражение>

[, имя_столбца = выражение]

[WHERE условие_отбора]

Обновление существующих данных

Увеличить предельный кредит для компании Acme Manufacturing до $60000 и закрепить ее за Мэри Джонс (ид. 109).

```
UPDATE CUSTOMERS
SET CREDIT_LIMIT = 60000.00, CUST_REP = 109
WHERE COMPANY = ‘Acme Mfg.’;
```

Обновление существующих данных

Перевести всех служащих из чикагского офиса (ид. 12) в нью-йорский офис (ид. 11) и понизить их личные планы на 10%.

Обновление существующих данных

Перевести всех служащих из чикагского офиса (ид. 12) в нью-йорский офис (ид. 11) и понизить их личные планы на 10%.

```
UPDATE SALESREPS
SET REP_OFFICE = 11, QUOTA = 0.9 * QUOTA
WHERE REP_OFFICE = 12;
```

Обновление существующих данных

Перевести всех клиентов, обслуживаемых служащими с ид. 105, 106, 107 к служащему 102.

```
UPDATE CUSTOMERS
SET CUST_REP = 102
WHERE CUST_REP IN (105, 106, 107);
```

Обновление существующих данных

Установить личный план продаж в 100000$ всем служащим, не имеющим в настоящий момент плана.

```
UPDATE SALESREPS
SET QUOTA = 100000.00
WHERE QUOTA IS NULL;
```

Обновление существующих данных

Обновление всех строк.

Увеличить все планы на 5%.

```
UPDATE SALESREPS
SET QUOTA = 1.05 * QUOTA.
```

Инструкция UPDATE с подзапросом

Увеличить на 5000$ лимит кредита для тех клиентов, которые сделали заказ на сумму более 25000$.

```
UPDATE CUSTOMERS
SET CREDIT_LIMIT = CREDIT_LIMIT + 5000.00
WHERE CUST_NUM IN (SELECT DISTINCT CUST
                   FROM ORDERS
                   WHERE AMOUNT > 25000.00);
```

Инструкция UPDATE с подзапросом

Переназначить клиентов, обслуживаемых служащими, чей объем продаж меньше 80 процентов их личного плана, служащему с ид 105.

Инструкция UPDATE с подзапросом

Переназначить клиентов, обслуживаемых служащими, чей объем продаж меньше 80 процентов их личного плана, служащему с ид 105.

```
UPDATE CUSTOMERS
SET CUST_REP = 105
WHERE CUST_REP IN (SELECT EMPL_NUM
                   FROM SALESREPS
                   WHERE SALES < (.8 * QUOTA));
```

Инструкция UPDATE с подзапросом

Всех служающих, обслуживающих более трех клиентов, подчинить непосредственно Сэму Кларку (ид. 106).

```
UPDATE SALESREPS
SET MANAGER = 106
WHERE 3 < (SELECT COUNT(*)
        FROM CUSTOMERS
        WHERE CUST_REP = EMPL_NUM);
```

Резюме

- Однострочная инструкция INSERT добавляет в таблицу одну строку данных. Значения новой строки задаются в инструкции в виде констант.
- Многострочная инструкция INSERT добавляет в таблицу нуль или более строк данных. Значения новых строк берутся из запроса, являющегося частью инструкции SELECT.
- Инструкция DELETE удаляет из таблицы нуль или более строк данных. Удаляемые строки задаются с помощью условий отбора.
- Инструкция UPDATE обновляет значения одного или более столбцов в нуле или более строках таблицы. Обновляемые строки задаются с помощью условия отбора. Обновляемые столбцы и выражения, определяющие новые значения, задаются в инструкции UPDATE

Целостность данных

Термин *целостность данных* относится к правильности и полноте информации, содержащейся в базе данных. При изменении содержимого базы данных с помощью инструкций INSERT, DELETE или UPDATE может произойти нарушение целостности содержащихся в ней данных.

Целостность данных

- В базу могут быть внесены неправильные данные, скажем, заказ, в котором указан несуществующий товар.
- Имеющимся данным, в результате изменения, могут быть присвоены некорректные значения, например служащий может быть назначен в несуществующий офис.
- Изменения, внесенные в базу данных, могут быть утеряны из-за системной ошибки или сбоя в электропитании либо они могут быть внесены лишь частично; например, заказ на товар может быть добавлен без учета изменения количества товара, имеющегося

Условия целостности данных

Для сохранения непротиворечивости и правильности хранимой информации в РСУБД устанавливается одно или несколько *условий (ограничений) целостности данных.* Эти условия определяют, какие значения могут быть записаны в базу данных в результате добавления или обновления данных.

Условия целостности данных

- **Обязательное наличие данных**. Некоторые столбцы в БД должны содержать значения в каждой строке; в таких столбцах не могут содержаться значения NULL или не содержаться никакие значения. Например, в учебной базе данных для каждого заказа должен существовать соответствующий клиент, сделавший этот заказ. Поэтому столбец CUST в таблице ORDERS является обязательным. Можно указать СУБД, что запись значения NULL в такие столбцы недопустима.

Условия целостности данных

- **Условие на значение**. У каждого столбца в БД есть свой *домен,* т.е. набор значений, которые допускается хранить в данном столбце. В учебной базе данных заказы нумеруются, начиная с числа 100001, поэтому доменом столбца ORDER_NUM являются положительные числа больше 100000. Аналогично идентификаторы служащих в столбцах EMPL_NUM должны находиться в диапазоне от 101 до 999. Можно указать СУБД, что запись значений, не входящих в определенный диапазон, в такие столбцы недопустима.

Условия целостности данных

- **Логическая целостность данных**. Первичный ключ таблицы должен в каждой строке иметь уникальное значение, отличное от значений во всех остальных строках. Например, каждая строка таблицы PRODUCTS имеет уникальную комбинацию значений в столбцах MFR_ID и PRODUCT_ID, которая однозначно идентифицирует товар, представляемый данной строкой. Повторяющиеся значения в этих столбцах недопустимы, поскольку тогда БД не сможет отличить один товар от другого. Можно потребовать от СУБД обеспечения логической целостности данных.

Условия целостности данных

- **Ссылочная целостность**. В реляционной БД каждая строка дочерней таблицы связана с помощью внешнего ключа со строкой родительской таблицы, содержащей первичный ключ, значение которого равно значению внешнего ключа. В учебной базе данных значение столбца REP_OFFICE таблицы SALESREPS связывает служащего с офисом, в котором он работает. Столбец REP_OFFICE *обязан* содержать значение из столбца OFFICE таблицы OFFICES; в противном случае служащий будет закреплен за несуществующим офисом. Можно указать СУБД, чтобы она обеспечивала

Условия целостности данных

- **Другие соотношения между данными**. Моделируемая БД ситуация реального мира зачастую накладывает собственные ограничения на корректность данных, которые могут храниться в БД. Например, в нашей учебной БД вице-президент по продажам может захотеть, чтобы план продаж каждого офиса не превышал суммарных планов продаж сотрудников этого офиса. СУБД можно предупредить о том, что при внесении изменений в планы продаж офиса и служащих должно выполняться указанное выше ограничение.

Условия целостности данных

- **Бизнес-правила**. Обновление информации в БД может быть ограничено бизнес-правилами, которым подчиняются сделки, представляемые подобными обновлениями. Например, компания, использующая учебную БД, может установить бизнес-правило, запрещающее принимать заказы на товар в количествах, превышающих количество товара на складе. Можно указать СУБД, что следует проверять каждую новую строку, добавляемую в таблицу ORDERS, и убеждаться, что значение и столбец QTY не нарушает установленное бизнес-правило.

Условия целостности данных

- **Непротиворечивость**. Многие реальные деловые операции вызывают в БД несколько изменений одновременно. Например, операция “прием заказа” может включать в себя добавление строки в таблицу ORDERS, увеличение значения столбца SALES в таблице SALESREPS для служащего, принявшего заказ, и увеличение значения столбца SALES в таблице OFFICES для офиса, за которым закреплен этот служащий. Одна инструкция INSERT и две инструкции UPDATE – все они должны быть выполнены для того, чтобы БД осталась в правильном непротиворечивом состоянии. Можно указать СУБД, что следует

Обязательность данных

Это наиболее простое, условие целостности данных требует, чтобы некоторые столбцы не содержали значений NULL. Само условие задается как часть инструкций CREATE TABLE в виде ограничения NOT NULL.

Обязательность данных

Если на столбец наложено ограничение NOT NULL, то для выполнения этого условия СУБД обеспечивает следующее:

- Ни в одной инструкции INSERT, добавляющей в таблицу строку или строки, нельзя указывать значение NULL для этого столбца; попытка добавить строку, содержащую (явно или неявно) значение NULL для такого столбца, вызовет ошибку.
- Ни в одной инструкции UPDATE, обновляющей столбец, нельзя присваивать столбцу значение NULL; попытка обновить такой столбец, присвоив ему значение NULL, вызовет ошибку.

Ограничения на значения столбца

В SQL *ограничения на значения столбца* аналогично условию отбора в предложении WHERE, которое возвращает значение TRUE или FALSE.

Если для столбца задано ограничение, то при каждом добавлении новой строки или обновлении старой СУБД автоматически проверяет, выполняется ли ограничение для значения в этом столбце.

Ограничения на значения столбца

Если условие не выполняется, то инструкция INSERT или UPDATE завершается ошибкой.

Ограничение на значение столбца задается при определении столбцов в инструкции CREATE TABLE.

```
CREATE TABLE SALESREPS
(
EMPL_NUM INTEGER NOT NULL
    CHECK (EMPL_NUM BETWEEN 101 AND 199),
AGE INTEGER
    CHECK (AGE >= 21)
)
```

Домены

Домен в SQL обобщает понятие ограничения на значения столбца и позволяет применять одно и то же ограничение для различных столбцов в БД.

Домен представляет собой множество допустимых значений

Стандарт SQL:

```
CREATE DOMAIN VALID_EMPLOYEE_ID INTEGER
CHECK (VALUE BETWEEN 101 AND 999)
```

Целостность данных

Каждая строка таблицы должна иметь уникальное значение первичного ключа, иначе база данных потеряет свою целостность и перестанет быть адекватной моделью внешнего мира.

По этой причине требование, чтобы первичные ключи имели уникальное значения, называется условием *целостности таблицы.*

Целостность данных

При указании первичного ключа в определении таблицы СУБД автоматически проверяет уникальность его значений при выполнении каждой инструкции INSERT или UPDATE.

Попытка добавить строку с уже имеющимся значением первичного ключа или обновить строку таким образом, что первичный ключ потеряет свою уникальность, завершится выдачей сообщения об ошибке.

Прочие условия уникальности столбцов

Иногда требуется, чтобы столбец, не являющийся первичным ключом таблицы, все же содержал уникальные значения во всех строках.

Прочие условия уникальности столбцов

Предположим, например, что требуется ограничить данные в таблице SALESREPS таким образом, чтобы не было двух служащих с одинаковыми именами.

Достичь этой цели можно, наложив условие *уникальности* на столбце NAME.

СУБД обеспечивает это условие точно так же, как обеспечивает уникальность первичного ключа.

Любая попытка добавить или обновить строку, нарушающая условие уникальности,

Прочие условия уникальности столбцов

Фундаментальные отличия между первичным ключом и условием уникальности:

- Таблица может иметь только один первичный ключ, в то время как ограничений уникальности может быть наложено несколько.
- Столбцы, указанные в первичном ключе, должны быть определены как NOT NULL, в то время как столбцы, включенные в условие уникальности, могут быть определены и как NULL, и как NOT NULL.

Уникальность и значения NULL

Значения NULL создают проблемы в столбце первичного ключа таблицы или в столбце, для которого задано условие уникальности.

Предположим, что вы пытаетесь добавить в таблицу строку с первичным ключом, имеющим значение NULL.

Из-за значения NULL СУБД не может однозначно решить, является ли первичный ключ дубликатом уже имеющегося ключа в таблице.

Из-за этого стандарт SQL требует, чтобы любой столбец был с ограничением NOT NULL.

Уникальность и значения NULL

Но стандарт SQL не накладывает такое ограничение на столбцы в условии уникальности, хотя часть реализаций накладывают ограничения.

Однако существуют различные расхождения в том, как разные реализации SQL обеспечивают выполнение условия уникальности в случае столбцов, допускающих значения NULL.

Уникальность и значения NULL

```
CREATE TABLE ADVISOR_ASSIGNMENTS
(
    STUDENT_NAME VARCHAR(25),
    ADVISOR_NAME VARCHAR(25),
    UNIQUE (STUDENT_NAME, ADVISOR_NAME));
```

Уникальность и значения NULL

Номер строки	STUDENT_NAME	ADVISOR_NAME	ORACLE	SQL SERVER	MySQL
1	NULL	NULL	Допустимая строка	Допустимая строка	Допустимая строка
2	NULL	NULL	Допустимая строка	Копия строки 1	Допустимая строка
3	Bill	NULL	Допустимая строка	Допустимая строка	Допустимая строка
4	Bill	NULL	Копия строки 3	Копия строки 3	Допустимая строка
5	Sue	Harrison	Допустимая строка	Допустимая строка	Допустимая строка
6	Sue	Harrison	Копия строки 5	Копия строки 5	Копия строки 5

Ссылочная целостность

Столбец OFFICE является первичным ключом таблицы OFFICES и уникальным образом идентифицирует каждую строку в этой таблице.

Столбец REP_OFFICE таблицы SALESREPS представляет собой внешний ключ для таблицы OFFICES.

Он идентифицирует офис, за которым закреплен каждый служащий

Ссылочная целостность

Столбцы REP_OFFICE и OFFICE создают между строками таблиц OFFICES и SALESREPS отношение “предок-потомок”.

Для каждой строки таблицы OFFICES (предок) существует нуль или более строк таблицы SALESREPS (потомки) с таким же идентификатором офиса.

Для каждой строки таблицы SALESREPS (потомок) существует ровно одна строка таблицы OFFICES (предок) с таким же идентификатором офиса.

Ссылочная целостность

Предположим, что вы пытаетесь вставить в таблицу SALESREPS новую строку, содержащую недопустимый идентификатор офиса, как в следующем примере:

```
INSERT INTO SALESREPS (EMPL_NUM, NAME,
REP_OFFICE, AGE, HIRE_DATE, SALES)
VALUES (115, ‘George Smith’, 31, 37, ‘2008-04-01’,
0.00)
```

Ссылочная целостность

Допустимое значение для столбца REP_OFFICE должно быть равно одному из значений, содержащихся в столбце OFFICE.

Это правило известно как ограничение *ссылочной целостности*. Оно обеспечивает целостность отношений “предок-потомок”, создаваемых внешними и первичными ключами.

Ссылочная целостность (проблемы)

- **Добавление новой дочерней строки**. Когда происходит добавление новой строки в дочернюю таблицу (SALESREPS), значение ее внешнего ключа (REP_OFFICE) *должно* быть равно одному из значений первичного ключа (OFFICE) в родительской таблице (OFFICES). Если значение внешнего ключа не равно ни одному из значений первичного ключа, то добавление такой строки повредит БД, поскольку в ней появится потомок без предка (“сирота”)

Ссылочная целостность (проблемы)

- **Обновление внешнего ключа в дочерней структуре**. Это та же проблема, что и в предыдущей ситуации, но выраженная в иной форме. Если внешний ключ (REP_OFFICE) обновляется инструкцией UPDATE, то его новое значение должно быть равно одному из значений первичного ключа (OFFICE) в родительской таблице (OFFICES). В противном случае обновленная строка окажется “сиротой”

Ссылочная целостность (проблемы)

- **Удаление родительской строки**. Если из родительской таблицы (OFFICES) будет удалена строка, у которой есть хотя бы один потомок (в таблице SALESREPS), то дочерние структуры станут “сиротами”. Значения внешних ключей (REP_OFFICE) в этих строках больше не будут равны ни одному из значений первичного ключа (OFFICE) родительской таблицы.

Ссылочная целостность (проблемы)

- **Обновление первичного ключа в родительской строке**. Это иная форма проблемы, рассмотренной в предыдущем пункте. Если происходит изменение первичного ключа (OFFICE) в родительской таблице OFFICES, все существующие потомки этой строки становятся “сиротами”, поскольку их внешние ключи больше не равны не одному первичному ключу.

Ссылочная целостность (проблемы)

Первая проблема (добавление строки в дочернюю таблицу) решается путем проверки значений в столбцах внешнего ключа перед выполнением инструкции INSERT.

Если они не равны ни одному из значений первичного ключа, то инструкция INSERT отбрасывается и выдается сообщение об ошибки.

Ссылочная целостность (проблемы)

Вторая проблема (обновление дочерней таблицы) решается аналогично, путем проверки нового значения внешнего ключа.

Если нет ни одного равного ему значений первичного ключа, инструкция UPDATE отбрасывается с выдачей сообщения об ошибке.

Ссылочная целостность (проблемы)

Третья проблема (удаление родительской строки) является более сложной.

Предположим, что вы закрыли офис в Лос-Анджелесе и хотите удалить соотв. строку из таблицы OFFICES. Что при этом при этом должно произойти с двумя дочерними строками в таблице SALESREPS, которые представляют служащих, закрепленных за офисом в Лос-Анджелесе?

Ссылочная целостность (проблемы)

- Не удалять из БД офис до тех пор, пока служащие не будут переведены в другой офис.
- Автоматически удалить двух служащих из таблицы SALESREPS.
- В столбце REP_OFFICE установить для этих двух служащих значения NULL, показывая тем самым, что идентификатор их офиса неизвестен.
- В столбце REP_OFFICE для этих двух служащих установить по умолчанию некоторое значение, например идентификатор главного офиса в Нью-Йорке, указывая тем самым, что

Ссылочная целостность (проблемы)

Аналогичные сложности существуют и в 4 ситуации (обновление первичного ключа в родительской таблице).

Допустим необходимо изменить идентификатор офиса в Лос-Анджелесе с 21 на 23.

Возникает вопрос: как поступить с двумя дочерними строками в таблице SALESREPS, представляющими служащих офиса в Л.А.

Ссылочная целостность (проблемы)

- Не изменять идентификатор офиса до тех пор, пока служащие не будут переведены в другой офис; в таком случае в таблицу OFFICES следует вначале добавить строку с новым идентификатором офиса в Л.А., затем обновить таблицу SALESREPS и, наконец, удалить строку со старым идентификатором офиса в Л.А.

Ссылочная целостность (проблемы)

- Автоматически обновить идентификатор офиса этих двух служащих в таблице SALESREPS для того, чтобы их строки были по-прежнему связаны с лос-анжелесской строкой в таблице OFFICES через ее идентификатор офиса;
- В столбце REP_OFFICE установить для этих двух служащих значение NULL, показывая тем самым, что идентификатор из офиса неизвестен;

Ссылочная целостность (проблемы)

- В столбце REP_OFFICE установить по умолчанию для этих двух служащих некоторое значение, например идентификатор главного офиса в Нью-Йорке, указывая тем самым, что служащие автоматически переводятся в этот офис.

Правила удаления и обновления

Для каждого отношения “предок-потомок” в БД, создаваемого внешним ключом, стандарт SQL позволяет указать связанные с ним правило удаления и правило обновления. Правило удаления определяет те действия, которые СУБД выполняет, когда пользователь пытается удалить строку из родительской таблицы.

Правила удаления и обновления

Можно задать одно из 4 правил:

- **RESTRICT –** запрещает удаление строки из родительской строки, если строка имеет потомков. Инструкция *DELETE*, пытающаяся удалить такую строку, отвергается, и выдается сообщение об ошибке. Таким образом, из родительской таблицы можно удалять только строки, не имеющие потомков. “Нельзя удалить офис, если в нем кто-то работает”.

Правила удаления и обновления

- **CASCADE** – определяет, что при удалении родительской строки все дочерние строки *также* автоматически удаляются из дочерней строки. “При удалении офиса его служащие автоматически удаляются”.

Правила удаления и обновления

- **SET NULL** – определяет, что при удалении родительской строки внешним ключам во всех ее дочерних строках автоматически присваивается значение NULL. Таким образом, удаление строки из родительской таблицы вызывает установку значений NULL в некоторых столбцах дочерней таблицы.

“При расформировании офиса служащие остаются в резерве, т.е. их офис неизвестен”

Правила удаления и обновления

- **SET DEFAULT** - указывает, что при обновлении значения первичного ключа в родительской строке внешним ключам во всех ее дочерних строках присваивается значение по умолчанию, установленное для данного столбца. Таким образом, изменение первичного ключа в родительской таблице вызывает выполнение установки значения по умолчанию в некоторых столбцах дочерней таблицы.

“При изменении идентификатора офиса все его служащие переводятся в офис по

Правила удаления и обновления

```
ALTER TABLE OFFICES
    ADD CONSTRAINT HASMGR
    FOREIGN KEY (MGR)
        REFERENCES SALESREPS(EMPL_NUM)
            ON UPDATE CASCADE
            ON DELETE SET NULL;
```

Каскадные удаления и обновления

Правило RESTRICT является одноуровневым в том смысле, что в отношении “предок-потомок” оно затрагивает только родительскую таблицу.

Правило CASCADE может быть многоуровневым.

Удаляем офис Л.А. из таблицы OFFICES => СУБД удалит из таблицы SALESREPS все строки, относящиеся к офису в Л.А. => Удаление всех строк, относящихся к таблице ORDERS.

Таким образом, удаление офиса вызывает каскадное удаление соотв. записей о служащих, что, в свою очередь, вызывает каскадное удаление заказов.

Каскадные удаления и обновления

Правила удаления и обновления SET NULL и SET DEFAULT являются двухуровневыми; их влияние заканчивается на дочерней таблице.

При удалении офиса в L.A. СУБД установит в столбце REP_OFFICE таблицы SALESREPS значение NULL в тех строках, где был идентификатор офиса 21.

Ссылочные циклы

В учебной БД таблица SALESREPS содержит столбец REP_OFFICE – внешний ключ для таблицы OFFICES. Таблица OFFICES, в свою очередь, содержит столбец MGR – внешний ключ для таблицы SALESREPS.

Эти два отношения образуют ссылочный цикл.

Ссылочные циклы представляют особую проблему для ссылочной целостности независимо от количества таблиц в них.

Ссылочные циклы

Предположим, что в 2 таблицах не допускаются значения NULL.

INSERT INTO SALESREPS (EMPL_NUN, NAME, REP_OFFICE, HIRE_DATE, SALES)

VALUES (115, ‘Ben Adams’, 14, ‘2008-01-01’, 0.00);

INSERT INTO OFFICE (OFFICE, CITY, REGION, MGR, TARGET, SALES)

VALUES (14, ‘Detroit’, ‘Eastern’, 115, 0.00, 0.00);

ОШИБКА!

Ссылочные циклы

```
INSERT INTO SALESREPS  (EMPL_NUN, NAME,
REP_OFFICE, HIRE_DATE, SALES)
VALUES (115, ‘Ben Adams’, NULL, ‘2008-01-01’, 0.00);

INSERT INTO OFFICE (OFFICE, CITY, REGION, MGR,
TARGET, SALES)
VALUES (14, ‘Detroit’, ‘Eastern’, 115, 0.00, 0.00);

UPDATE SALESREPS
SET REP_OFFICE = 14
WHERE EMPL_NUM = 115;
```

Внешние ключи и значения NULL

В отличие от первичных ключей, внешние ключи в реляционной базе данных могут содержать значения NULL.

Пример REP_OFFICE – NULL (Tom Snyder)

Расширенные возможности ограничений

- Ограничения столбцов
- Домены
- Ограничения таблиц
- Утверждения

Типы ограничений SQL

- Ограничения NOT NULL
- Ограничение PRIMARY KEY
- Ограничение UNIQUE
- Ограничение FOREIGN KEY
- Ограничение CHECK

Каждому ограничению может быть присвоено имя.

Курсовая

1. О чем ваша база данных
2. Описание всех таблиц и ограничений целостности (ERP – диаграмма и описание)
3. Скрипт по созданию таблиц
4. Скрипты по добавлению и обновлению данных в таблице
5. Скрипты запросов к базе данных
6. Индексы (Где эти индексы будут проставлены и куда)
7. * Работа с базой данных через ЯП