Принципы поддержки целостности в реляционной модели данных

- поддержка *структурной целостности*
- поддержка *языковой целостности*
- поддержка *ссылочной целостности*
- поддержка *семантической целостности.*
-

поддержка *структурной целостности*

- реляционная СУБД должна допускать работу только с однородными структурами данных типа «реляционное отношение» т.е.
- отсутствие дубликатов кортежей,
- соответственно обязательное наличие первичного ключа,
- отсутствие понятия упорядоченности кортежей.

поддержка *языковой целостности*

Реляционная СУБД должна обеспечивать языки описания и манипулирования данными не ниже стандарта SQL.

- Не должны быть доступны иные низкоуровневые средства манипулирования данными, не соответствующие стандарту.

поддержка *ссылочной целостности*

- кортежи подчиненного отношения уничтожаются при удалении кортежа основного отношения, связанного с ними.
- кортежи основного отношения модифицируются при удалении кортежа основного отношения, связанного с ними, при этом на месте ключа родительского отношений ставится неопределенное Null значение.

Семантическая поддержка целостности.

- Семантическая поддержка может быть обеспечена двумя путями:
- Декларативным и
- процедурным путем.

Включение ограничений

- Ограничения обеспечивают декларативную поддержку целостности.
- Что такое ограничения?
- Создание и сопровождение ограничений

Что такое ограничения?

- Ограничения обеспечивают выполнение правил на уровне таблицы.
- Ограничения предотвращают удаление таблицы при наличии подчиненных данных в других таблицах.
- В Oracle допускаются следующие виды ограничений:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK

Указания по ограничениям

- Присваивайте имена ограничениям сами; в противном случае сервер Oracle присвоит имя в формате SYS_C*n*.
- Создавайте ограничения:
  - При создании таблицы
  - После создания таблицы
- Устанавливайте ограничения на уровне столбца или таблицы.
- Просматривайте ограничения в словаре данных.

Определение ограничений

```
CREATE TABLE [schema.]table
            (column datatype [DEFAULT expr]
            [column_constraint],
            ...
            [table_constraint]);
```

```
CREATE TABLE emp(
             empno   NUMBER(4),
             ename   VARCHAR2(10),
             ...
             deptno  NUMBER(7,2) NOT NULL,
             CONSTRAINT emp_empno_pk
                        PRIMARY KEY (EMPNO));
```

Определение ограничений

- Ограничение на уровне столбца

```
column [CONSTRAINT constraint_name] constraint_type,
```

- Ограничение на уровне таблицы

```
column,...
   [CONSTRAINT constraint_name] constraint_type
   (column, ...),
```

Ограничение NOT NULL

Предотвращает появление неопределенных значений в столбце

EMP

EMPNO	ENAME	JOB	...	COMM	DEPTNO
7839	KING	PRESIDENT			10
7698	BLAKE	MANAGER			30
7782	CLARK	MANAGER			10
7566	JONES	MANAGER			20
...					

Ограничение NOT NULL (ни одна строка не может содержать неопределенное значение в этом столбце)

Отсутствие ограничения NOT NULL (любая строка может содержать неопределенное значение в этом столбце)

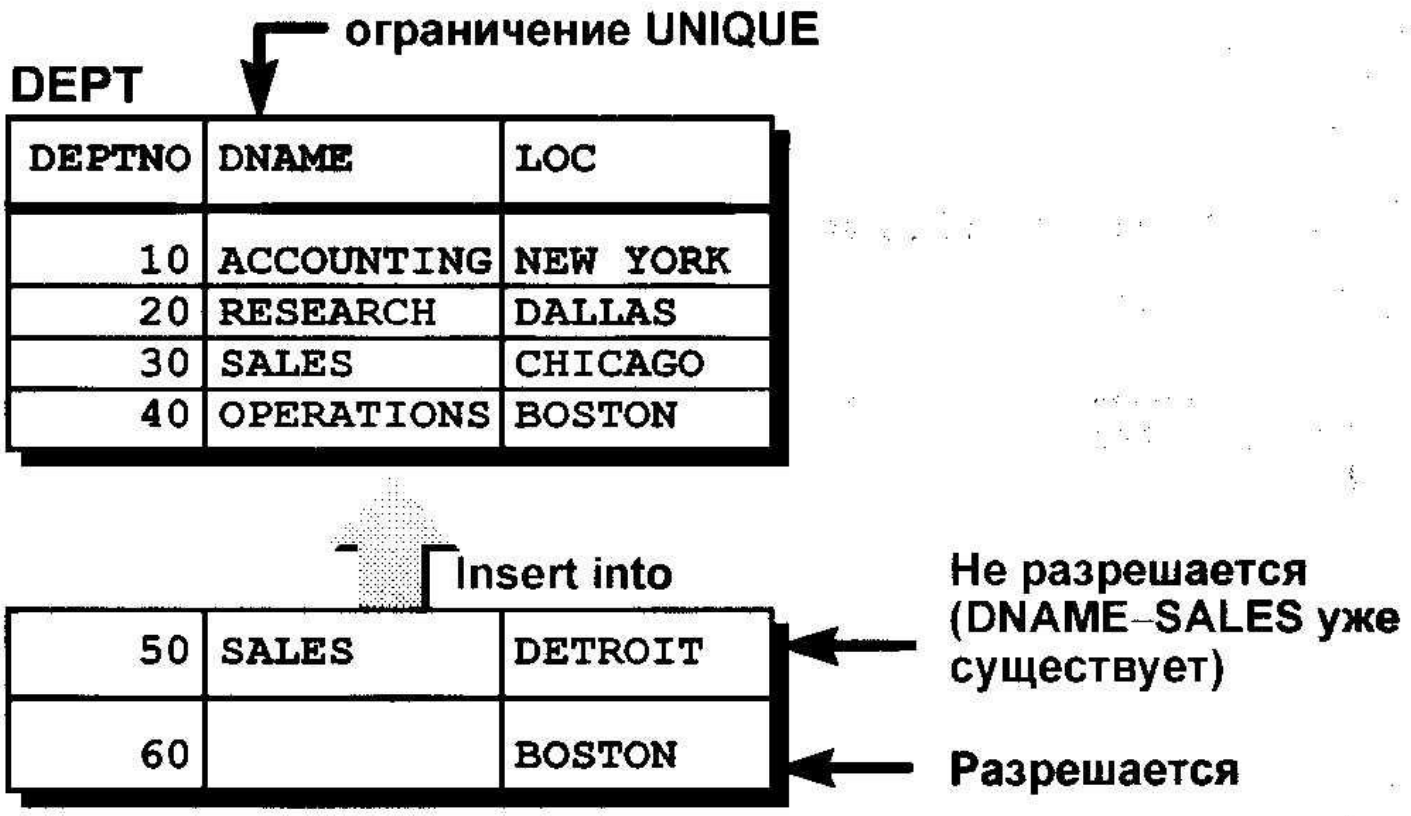
Ограничение NOT NULL

Ограничение NOT NULL

Может быть задано только для столбца

```
SQL> CREATE TABLE emp(
  2       empno     NUMBER(4),
  3       ename     VARCHAR2(10) NOT NULL,
  4       job       VARCHAR2(9),
  5       mgr       NUMBER(4),
  6       hiredate  DATE,
  7       sal       NUMBER(7,2),
  8       comm      NUMBER(7,2),
  9       deptno    NUMBER(7,2) NOT NULL);
```

Ограничение UNIQUE

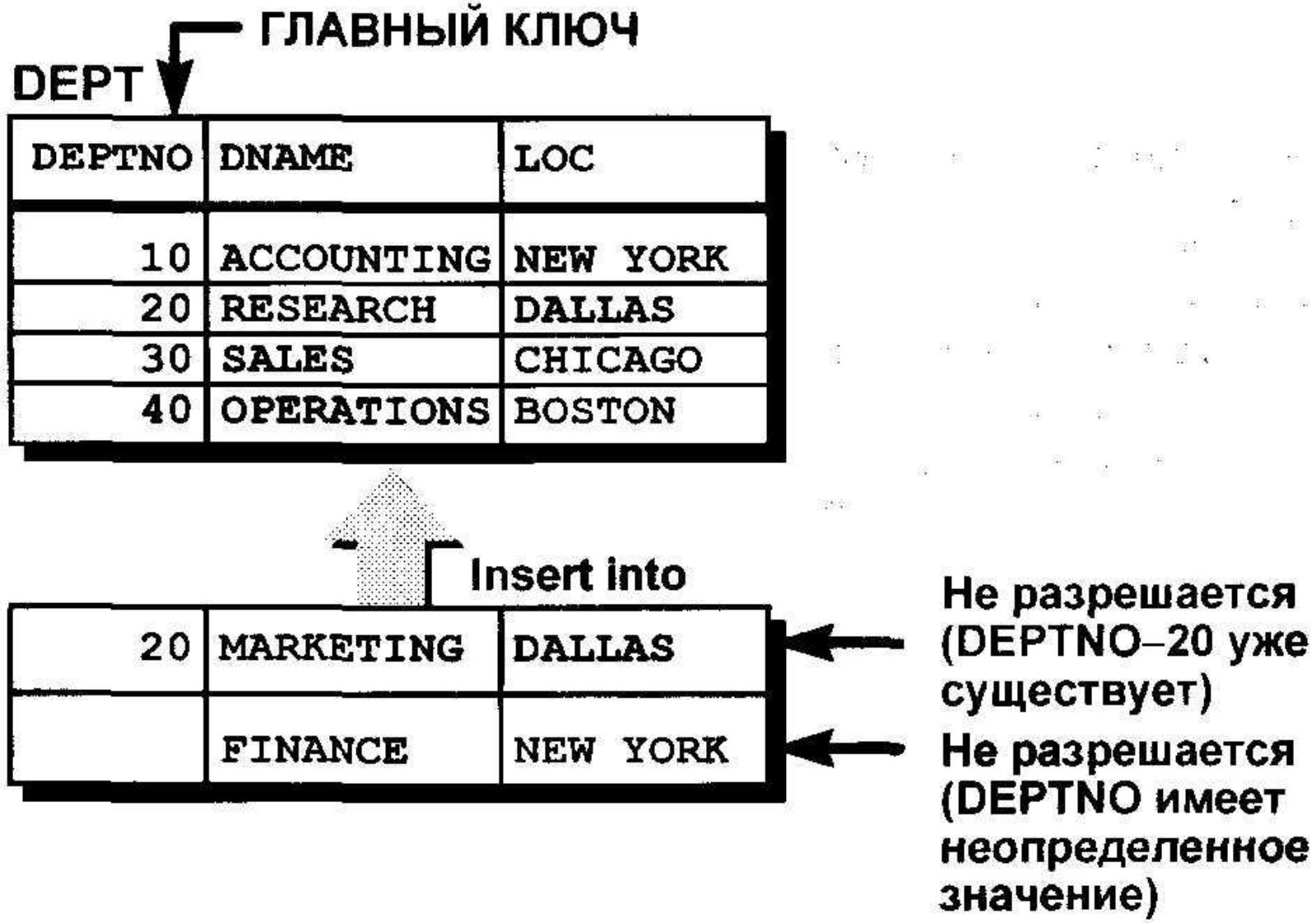


Ограничение UNIQUE

Может быть задано на уровне столбца или таблицы

```
SQL> CREATE TABLE   dept(
  2       deptno      NUMBER(2),
  3       dname       VARCHAR2(14),
  4       loc         VARCHAR2(13),
  5       CONSTRAINT dept_dname_uk UNIQUE(dname));
```

Ограничение PRIMARY KEY

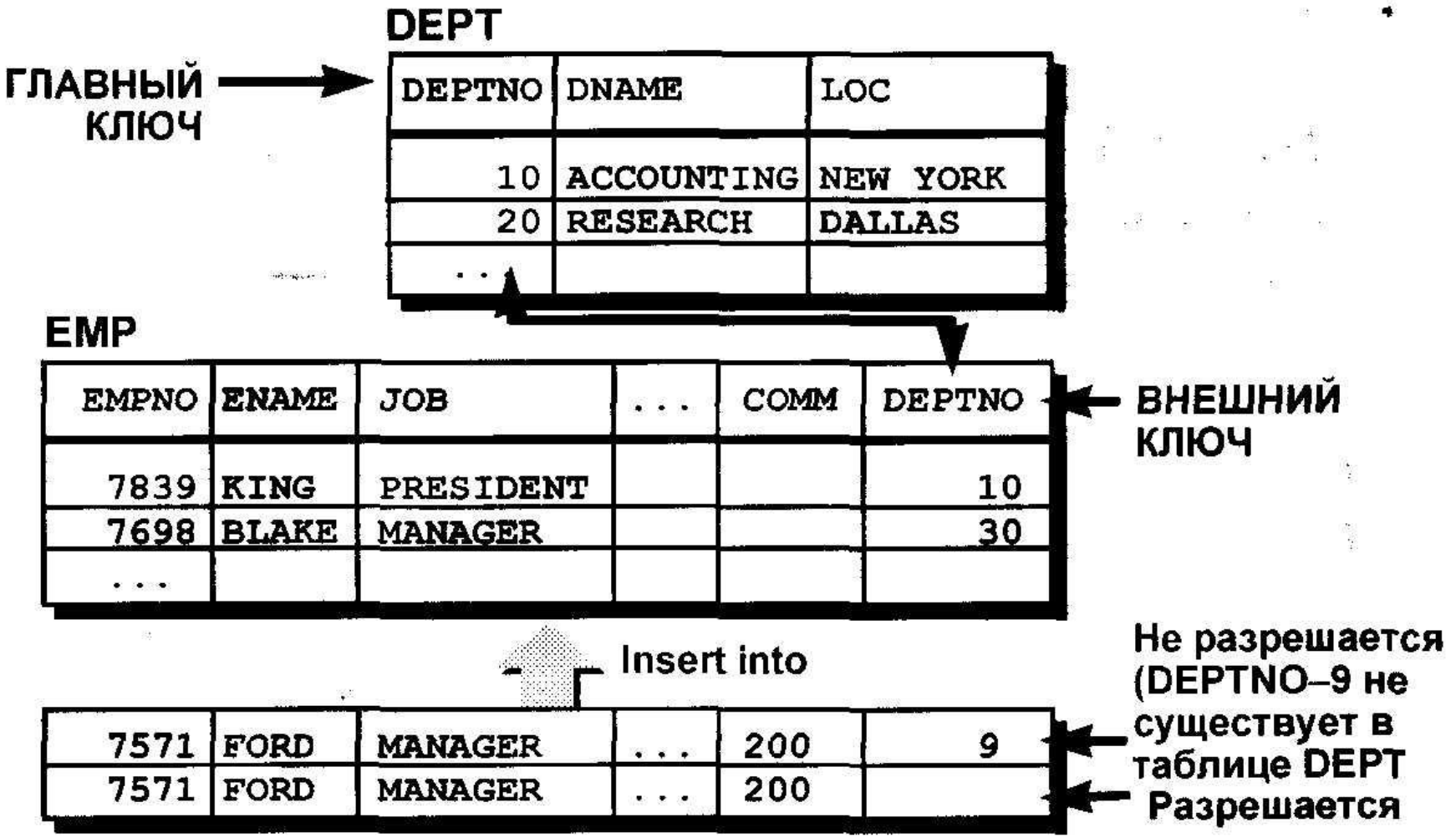


Ограничение PRIMARY KEY

Может быть задано на уровне таблицы или столбца

```
SQL> CREATE TABLE   dept(
  2       deptno      NUMBER(2),
  3       dname       VARCHAR2(14),
  4       loc         VARCHAR2(13),
  5       CONSTRAINT dept_dname_uk UNIQUE (dname),
  6       CONSTRAINT dept_deptno_pk PRIMARY KEY(deptno));
```

Ограничение FOREIGN KEY



Ограничение FOREIGN KEY

Может быть задано на уровне таблицы или столбца

```
SQL> CREATE TABLE emp(
  2       empno     NUMBER(4),
  3       ename     VARCHAR2(10) NOT NULL,
  4       job       VARCHAR2(9),
  5       mgr       NUMBER(4),
  6       hiredate  DATE,
  7       sal       NUMBER(7,2),
  8       comm      NUMBER(7,2),
  9       deptno    NUMBER(7,2) NOT NULL,
 10       CONSTRAINT emp_deptno_fk FOREIGN KEY (deptno)
 11                 REFERENCES dept (deptno));
```

Ключевые слова ограничения FOREIGN KEY

- **FOREIGN KEY**

  Определяет столбец в подчиненной таблице, используемый в качестве внешнего ключа

- **REFERENCES**

  Определяет родительскую таблицу и столбец в ней

- **ON DELETE CASCADE**

  Разрешает удаление в родительской таблице с одновременным удалением зависимых строк в подчиненной таблице

Пример

- Оператор создания таблицы BOOKS из базы данных «Библиотека».
- Бизнес-правила:
- *Шифр книги* — последовательность символов длиной не более 14, однозначно определяющая книгу, значит, это — фактически первичный ключ таблицы BOOKS.
- *Название книги* — последовательность символов, не более 120. Обязательно должно быть задано.
- *Автор* — последовательность символов, не более 30, может быть не задан.
- *Соавтор* — последовательность символов, не более 30, может быть не задан.
- *Год издания* — целое число, не менее 1960 и не более текущего года. По умолчанию ставится текущий год.
- *Издательство* — последовательность символов, не более 20, может отсутствовать.
- *Количество страниц* — целое число не менее 5 и не более 1000.

Оператор

```
CREATE TABLE BOOKS

    ISBN       varchar(14)  NOT NULL PRIMARY KEY,
    TITLE      varchar(120) NOT NULL,
    AUTOR      varchar (30) NULL,
    COAUTOR    varchar(30)  NULL,
    YEAR_PUBL smallint DEFAULT Year(GetDate()) CHECK(YEAR_PUBL >= 1960 AND
       YEAR_PUBL <= YEAR(GetDate())),
   PUBLICH   varchar(20)  NULL,
   PAGES     smallint CHECK(PAGES > = 5 AND PAGES <= 1000)
);
```

Дополнительное ограничение для таблицы

```
CREATE TABLE BOOKS
(
   ISBN       varchar(14)     NOT NULL PRIMARY KEY,
   TITLE      varchar(120)    NOT NULL,
   AUTOR      varchar (30)    NULL,
   COAUTOR    varchar(30)     NULL,
   YEAR_PUBL smallint DEFAULT Year(GetDate()) CHECK(YEAR_PUBL >= 1960 AND
      YEAR_PUBL <= YEAR(GetDate())),
   PUBLICH    varchar(20)     NULL,
   PAGES      smallint        CHECK(PAGES > = 5 AND PAGES <= 1000),
   CHECK (NOT (AUTOR IS NULL AND COAUTOR IS NOT NULL))
);
```

Именованные ограничения

- Для анализа ошибок целесообразно именовать все ограничения, особенно если таблица содержит несколько ограничений одного типа.
- Для именования ограничений используется ключевое слово CONSTRAINT

Создание BOOKS с именованными ограничениями

```
CREATE TABLE BOOKS
(
ISBN ,     varchar(14)      NOT NULL.
TITLE      varchar(120)     NOT NULL.
AUTOR      varchar (30)     NULL.
COAUTOR    varchar(30)      NULL.
YEAR_PUBL  smallint         NOT NULL.
PUBLICH    varchar(20)      NULL.
PAGES      smallint         NOT NULL.
CONSTRAINT PK_BOOKS PRIMARY KEY (ISBN).
CONSTRAINT DF_ YEAR_PUBL DEFAULT (Year(GetDate()).
CONSTRAINT CK_ YEAR_PUBL CHECK (YEAR_PUBL >= 1960 AND
   YEAR_PUBL <= YEAR(GetDate())).
CONSTRANT CK_PAGES CHECK (PAGES > = 5 AND PAGES <= 1000).
CONSTRAINT CK_BOOKS CHECK (NOT (AUTOR IS NULL AND COAUTOR IS NOT NULL))
);
```

Таблица READERS:

- *Номер читательского билета* - это целое число в пределах 32 000 и он уникально определяет читателя.
- *Имя, фамилия* читателя — это последовательность символов, не более 30.
- *Адрес* — это последовательность символов, не более 50.
- *Номера телефонов рабочего и домашнего* — последовательность символов, не более 12.
- *Дата рождения* — календарная дата. В библиотеку принимаются читатели не младше 17 лет.

Оператор

```
CREATE TABLE READERS
(
READER_ID       Smallint(4)  PRIMARY KEY,
   FIRST_NAME   char(30)     NOT NULL,
   LAST_NAME    char(30)     NOT NULL,
   ADRES        char(50),
   HOME_PHON    char(12),
   WORK_PHON    char(12),
   BIRTH_DAY date CHECK(DateDiff(year, GetDate(),BIRTH_DAY) >=17)
);
```

Таблица Examplar

```
CREATE TABLE EXEMPLAR
(
   EXEMPLAR_ID  INT          IDENTITY PRIMARY KEY,
   ISBN         varchar(14)  NOT NULL FOREIGN KEY references BOOKS(ISBN),
   READER_ID    Smallint(4)  NULL FOREIGN KEY references READERS (READER_ID),
   DATA_IN      date,
   DATA_OUT     date,
   EXIST        Logical,
);
```

Порядок создания таблиц

- В нашем примере с библиотекой порядок описания таблиц следующий:

1. Таблица BOOKS
2. Таблица READERS
3. Таблица CATALOG (системный каталог)
4. Таблица EXEMPLAR
5. Таблица RELATION_1 (дополнительная связующая таблица между книгами и системным каталогом).

Средства определения схемы базы данных

- В СУБД ORACLE база данных создается в ходе установки программного обеспечения собственно СУБД. Все таблицы пользователей помещаются в единую базу данных.
- Однако они могут быть разделены на группы, объединенные в подсхемы.
- Понятие подсхемы не стандартизировано в SQL и не используется в других СУБД.

Семантическое обеспечение целостности данных

Процедуры и триггеры

Хранимые процедуры

- Хранимые процедуры пишутся на специальном встроенном языке программирования, они могут включать любые операторы SQL, а также включают некоторый набор операторов, управляющих ходом выполнения программ

Синтаксис

- CREATE [OR REPLACE]
- (“аргумент” IN | OUT | IN OUT “Тип данных” [,..])
- IS | AS
- “Тело процедуры PL/SQL”

Функция получения ip-адреса

- **create or replace function**
  client_ip_address
  return varchar2 is
  begin
    return dbms_standard.client_ip_address;
  end;

Пример процедуры

- create or replace procedure update_debtsis
- Begin
- update computation c set n_pay=(select sum(n_sum)
- from payment
- where n_client=clients.n_client and d_pay between dates.d_computation and add_months(dates.d_computation,1))
- end update_debts;

Триггеры

- Фактически триггер — это специальный вид хранимой процедуры, которую SQL Server вызывает при выполнении операций модификации соответствующих таблиц.
- Триггер автоматически активизируется при выполнении операции, с которой он связан.
- Триггеры связываются с одной или несколькими операциями модификации над одной таблицей.

два типа триггеров

- В СУБД Oracle определены два типа триггеров:
- триггеры, которые могут быть запущены перед реализацией операции модификации, они называются BEFORE-триггерами,
- и триггеры, которые активизируются после выполнения соответствующей модификации, аналогично триггерам MS SQL Server, — они называются AFTER-триггерами.

Синтаксис

- CREATE [OR REPLACE] TRIGGER <имя_триггера> BEFORE | AFTER
- ON <имя_таблицы>
- FOR { [INSERT] [,UPDATE] [, DELETE] }
- FOR EACH ROW
- WHEN (условие)
- AS
- SQL-операторы (Тело триггера)

Пример1

- **create or replace trigger** add_author **AFTER INSERT OR UPDATE OF** C_AUTHOR
- **ON** T_CLAUSES
  FOR EACH ROW

- **DECLARE**
  id_cl **int**;
  aut **varchar2**(500);
  res **int**;

- **BEGIN**
  id_cl := :new.N_ID_CL;
  aut := :new.C_AUTHOR;
  res := ANALIZ_AUT(id_cl, aut);
  END;

Пример2

- create or replace trigger "BI_COMPUTATION"
- before insert on "COMPUTATION"
- for each row
- begin
- select "COMPUTATION_SEQ".nextval into :NEW.N_COMPUTATION
- from dual;
- :NEW.D_COMPUTATION := SYSDATE();
- end;

Ограничения

- Нельзя использовать в теле триггера операции создания объектов БД (новой БД, новой таблицы, нового индекса, новой хранимой процедуры, нового триггера, новых индексов, новых представлений),
- Нельзя использовать в триггере команду удаления объектов DROP для всех типов базовых объектов БД.
- Нельзя использовать в теле триггера команды изменения базовых объектов ALTER TABLE, ALTER DATABASE.
- Нельзя изменять права доступа к объектам БД, то есть выполнять команду GRAND или REVOKE.
- Нельзя создать триггер для представления (VIEW).
- В отличие от хранимых процедур, триггер не может возвращать никаких значений, он запускается автоматически сервером и не может связаться самостоятельно ни с одним клиентом.