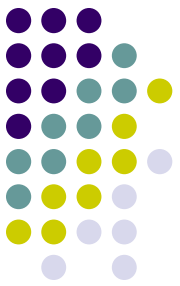


# Ограничение целостности в Oracle

---

Подготовил: слушатель группы 7342  
ряд.Евдокимов В.А.





# Целостность базы данных

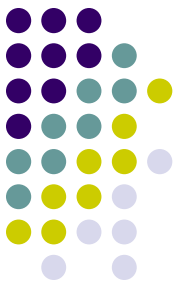
**Целостность базы данных (DATABASE INTEGRITY)** - соответствие информации в БД её структуре и правилам. Каждое правило, налагающее ограничение на состояние базы данных, называется ограничением целостности (integrity constraint).

Примеры ограничений:

- возраст сотрудника не может быть меньше 18 и больше 65 лет.
- каждый сотрудник имеет уникальный табельный номер.
- сотрудник обязан числиться в одном отделе.

# Работу системы по проверке ограничений





# Транзакции

- Транзакция – последовательность операций с данными, выполняющаяся как единое целое.
- Свойства:

Атомарность

Долговечность

Согласованность

Изоляция

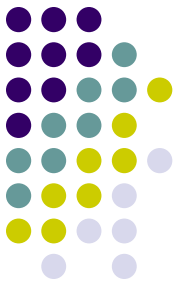
# Команды управления транзакциями



Транзакция начинается автоматически с момента присоединения пользователя к СУБД и продолжается, пока не произойдет одно из событий:

- Подана команда `COMMIT WORK` (зафиксировать транзакцию).
- Подана команда `ROLLBACK WORK` (откатить транзакцию).
- Произошло отсоединение пользователя от СУБД.
- Произошел сбой системы.

# Свойства и классификация ограничений целостности



Ограничения целостности обладают следующими свойствами:

- навязывают правила на уровне таблицы,
- предотвращают удаления строк в случаях, если на эти строки в таблицы наложены ограничения,
- ограничения к таблице могут быть наложены на момент создания таблицы, а также после создания таблицы

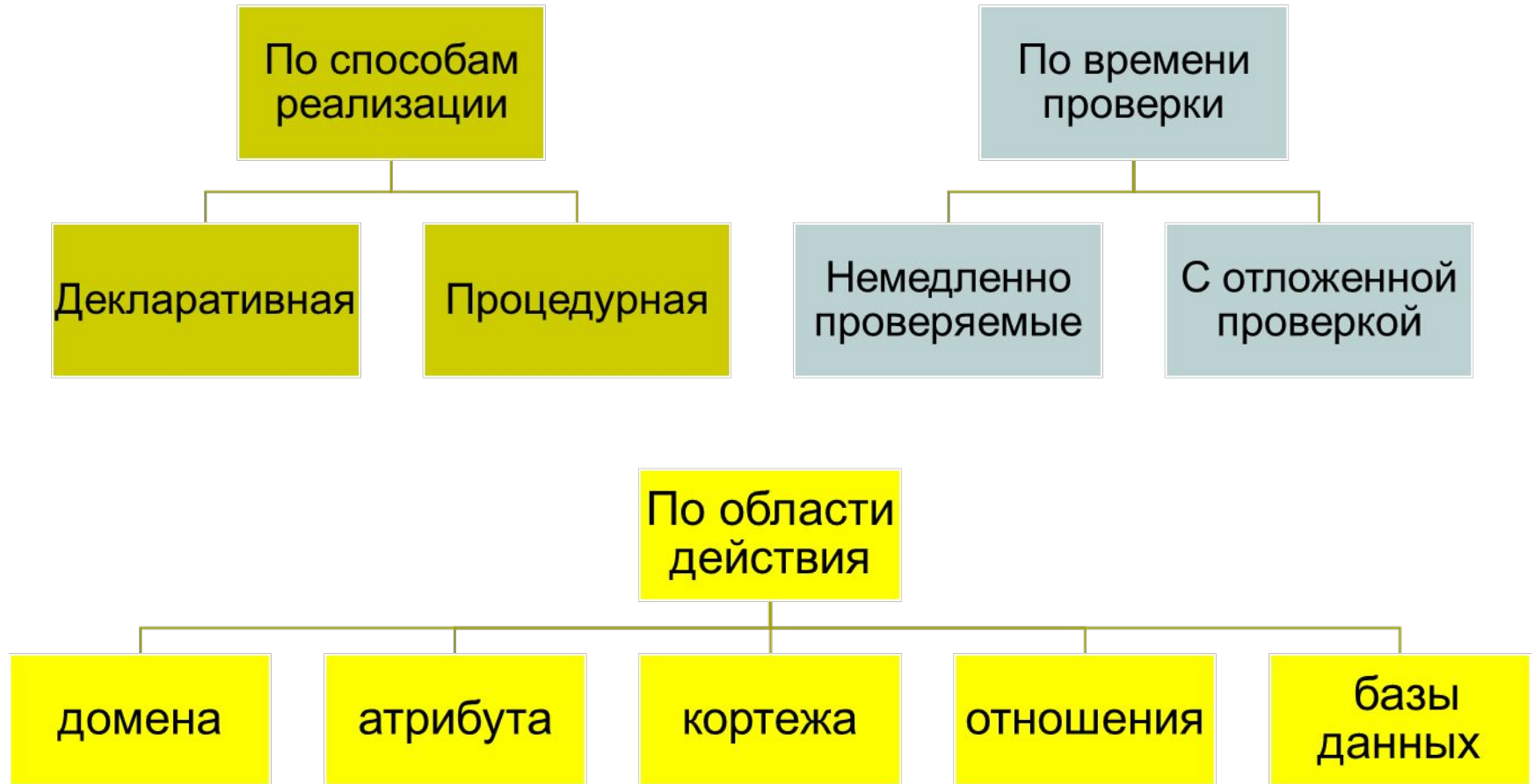
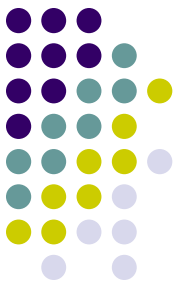
## Классификация

По способам реализации

По времени проверки

По области действия

# Классификация ограничений целостности



# Задание декларативных ограничений



Стандарт SQL позволяет задавать декларативные ограничения следующими способами:

- Как ограничения домена.
- Как ограничения, входящие в определение таблицы.
- Как ограничения, хранящиеся в базе данных в виде независимых утверждений (assertion).



# Способы определения



**CONSTRAINT** Имя ограничения]

```
{ PRIMARY KEY (Имя столбца, ...)}  
  {UNIQUE (Имя столбца, ...)}  
  {FOREIGN KEY (Имя столбца, ...) REFERENCES Имя таблицы [(Имя столбца, ...)]  
[Ссылочная спецификация]}  
  {Ограничение check }  
[Атрибуты ограничения]
```

Ограничения столбца:

[**CONSTRAINT** Имя ограничения]

```
{ {NOT NULL}  
  {PRIMARY KEY}  
  {UNIQUE}  
  {REFERENCES Имя таблицы [(Имя столбца)] [Ссылочная спецификация]}  
  { Ограничение check }  
}
```

Ссылочная спецификация:

[**MATCH** {FULL | PARTIAL}]

[**ON UPDATE** {CASCADE | SET NULL | SET DEFAULT | NO ACTION}]

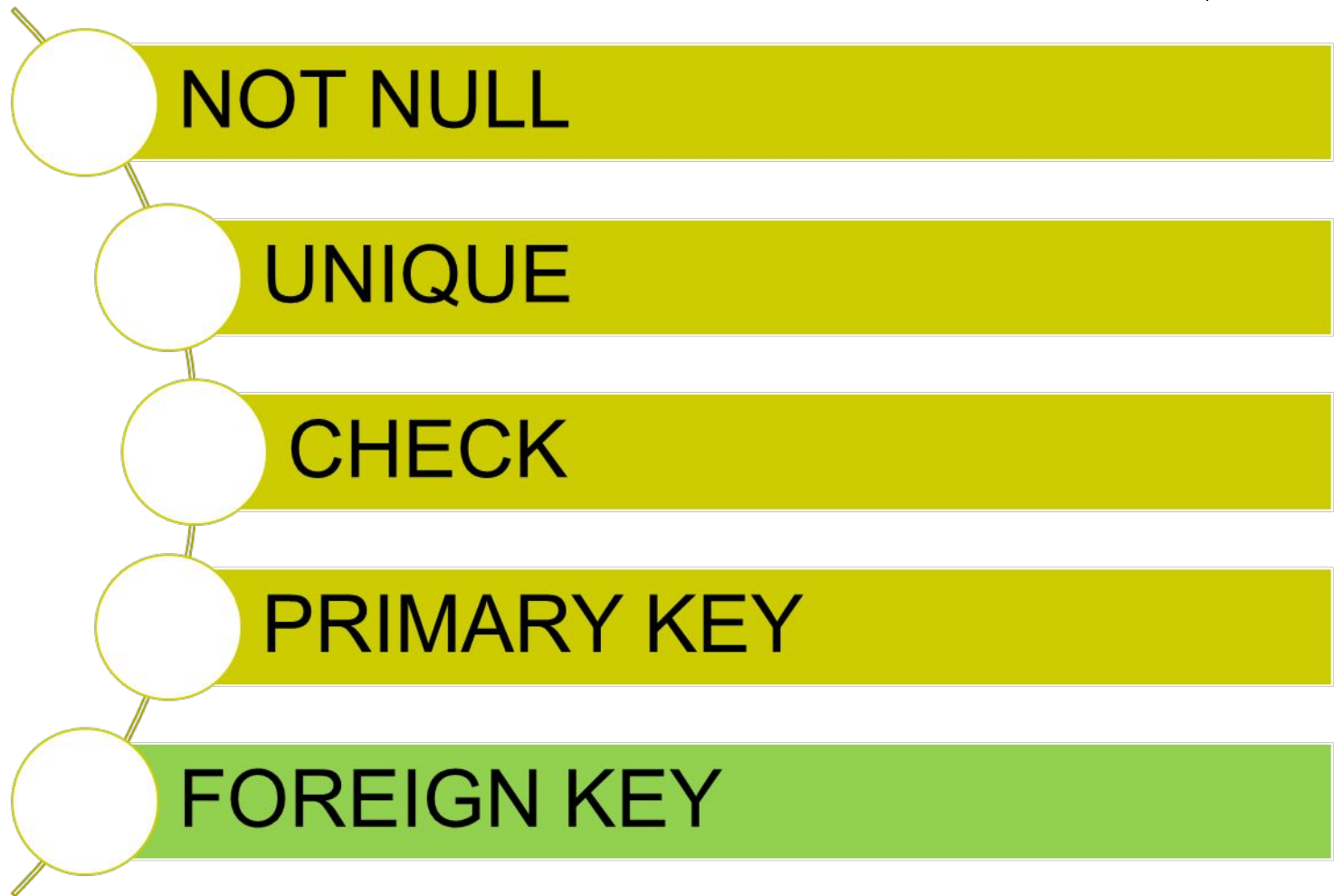
[**ON DELETE** {CASCADE | SET NULL | SET DEFAULT | NO ACTION}]

Атрибуты ограничения:

{**DEFERRABLE** [INITIALLY DEFERRED | INITIALLY IMMEDIATE]}

{**NOT DEFERRABLE**}

# Типы ограничений





# NOT NULL

Ограничение NOT NULL используется для тех столбцов таблицы, которые требуют, чтобы значение было всегда.

Таблица EMP:

ID	ENAME	JOB	HIREDATE	SAL	ID_DEPT
7329	SMITH	CEO	16.12.85	9000	20
7499	ALLEN	VP-SALES	02.04.90	7500	30
7521	WARD	MANAGER	23.08.91	5000	30
7566	JONES	SALEMAN	12.01.90	3700	30



NOT NULL



НЕ NOT NULL

**Замечание:** NOT NULL можно лишь определить на уровне столбца

# Определение ограничения NOT NULL



При создании таблицы:

```
CREATE TABLE EMP  
(  
ID number(20,0) primary key,  
ENAME varchar (50) not null,  
JOB varchar (50),  
...  
);
```

На имеющуюся таблицу, :

```
ALTER TABLE EMP  
MODIFY ENAME NOT NULL
```



# UNIQUE

Ограничения UNIQUE проверяет столбец на уникальность строк.

Таблица DEPT:

ID	NAME	LOC
20	RESEARCH	NEW YORK
30	SALES	BOSTON
40	MARKETING	DALLAS
50	SALES	BOSTON
60		TORONTO

Строка SALES уже есть

NAME пусто

# Определение ограничения UNIQUE



Пример при создании таблицы на уровне столбцов:

```
CREATE TABLE DEPT  
(ID number(20, 0) primary key ,  
NAME varchar2 (50) UNIQUE,  
LOC varchar2 (50));
```

На уровне таблицы:

```
CREATE TABLE DEPT  
(ID number(20, 0) primary key ,  
NAME varchar2 (50),  
LOC varchar2 (50),  
constraint my_con_uq UNIQUE(NAME);
```

На имеющуюся таблицу:

```
ALTER TABLE DEPT  
ADD CONSTRAINT con_uq UNIQUE(NAME [, column_name, ...])
```

# PRIMARY KEY



PRIMARY KEY используется для тех столбцов таблицы, которые позволяют идентифицировать каждую строку и гарантирует, что не будет повторяющихся.

## PRIMARY KEY = UNIQUE + NOT NULL

Ограничение PRIMARY KEY обладает следующими свойствами:

- ❑ в одной таблице может быть только одно ограничение типа PRIMARY KEY
- ❑ ограничение проверяет строки на уникальность и на то, чтобы они не были NOT NULL
- ❑ оно может быть определено, как на уровне столбца так и на уровне таблицы
- ❑ поиск по столбцу с ограничением PRIMARY KEY является быстрым поиском (так как по ним автоматически создается индекс)

# Определение ограничения PRIMARY KEY



Пример при создании таблицы на уровне столбцов:

```
CREATE TABLE DEPT  
(ID number(20, 0) primary key ,  
NAME varchar2 (50) UNIQUE,  
LOC varchar2 (50));
```

На уровне таблицы:

```
CREATE TABLE DEPT  
(ID number(20, 0),  
NAME varchar2 (50) unique not null,  
LOC varchar2 (50) not null,  
PRIMARY KEY(ID));
```

На имеющуюся таблицу:

```
ALTER TABLE DEPT  
ADD PRIMARY KEY (ID)
```



# CHECK

Ограничения CHECK позволяет ставить ограничение на значение столбцов.

Пример ограничения при создании таблицы:

```
CREATE TABLE EMP
```

```
(
```

```
ID number(20,0) primary key,
```

```
ENAME varchar (50) not null,
```

```
SAL number CHECK(SAL >= 1000)
```

```
...
```

```
);
```

При этом система сама автоматически назначит имя нашему ограничению.



# Определение ограничения CHECK



В этом случае мы сами явно задаем имя нашему ограничению:

```
CREATE TABLE EMP
(  
  ID number(20,0) primary key,  
  ENAME varchar(50) not null,  
  SAL number,  
  constraint con_chk CHECK (SAL >= 1000).  
);
```

На имеющуюся таблицу:

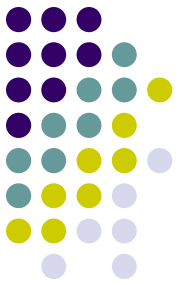
```
ALTER TABLE EMP  
ADD CONSTRAINT con_chk CHECK (SAL >= 0);
```

# Ссылочная целостность



В базе существуют связи между таблицами, к примеру таблицы DEPT и EMP связаны через столбец ID\_DEPT. Это означает:

- все сотрудники распределены по имеющимся отделам,
- невозможно взять сотрудника, зарегистрировав его на несуществующий отдел,
- можно создать отдел, и в нем могут быть пока не зарегистрированные сотрудники,
- при увольнение сотрудника – просто удаляем запись про него
- при закрытие(удаление) отдела – сперва нужно проверить, нет ли сотрудников, работающих в нем. Если нет, то удаляем отдел, а если есть то:
  - а) не разрешать удаление этого отдела, пока есть хотя бы один сотрудник привязанный к этому отделу
  - б) удалить отдел и всех сотрудников, которые в нем зарегистрированы
  - в) удалить отдел, при этом сотрудником назначить пустой (null) отдел



# Интерпретация

А теперь переведем верхнее сказанное на наш язык:

- каждой строки EMP должен быть определено поле ID\_DEPT, определенное в таблице DEPT в поле ID, или NULL,
- невозможно INSERT INTO EMP указав значение ID\_DEPT, которое отсутствует в таблице DEPT в поле ID (за искл. NULL),
- можно запустить INSERT INTO DEPT – указав любое значение полю ID,
- DELETE FROM EMP – срабатывает успешно
- при DELETE FROM DEPT WHERE ID=XXX – если в EMP нет строк, у которых поле ID\_DEPT имеет значение XXX, то DELETE сработает, а если же есть хотя бы одна строка из EMP, у которой поле ID\_DEPT имеет значение XXX, то
  - а) DELETE не сработает, пока все строки из EMP, у которых поле ID\_DEPT имеет значение XXX не будут удалены
  - б) сработает DELETE и автоматически запустится DELETE FROM EMP WHERE ID\_DEPT=XXX
  - в) сработает DELETE и автоматически запустится UPDATE EMP SET ID\_DEPT = NULL WHERE DEPTNO=XXX

# Родительские и дочерние таблицы

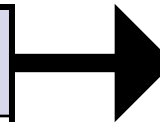


- Таблица EMP получается зависимой от DEPT.
- Таблицы, которые ссылаются на значения других таблиц называются дочерними, а те таблицы на значение, которых ссылаются называются родительскими.

EMP - дочерняя

DEPT - родительская

ID	ENAME	JOB	HIREDATE	SAL	ID_DEPT
7329	SMITH	CEO	16.12.85	9000	20
7499	ALLEN	VP-SALES	02.04.90	7500	30
7521	WARD	MANAGER	23.08.91	5000	30
7566	JONES	SALEMAN	12.01.90	3700	30



ID	NAME	LOC
20	RESEARCH	NEW YORK
30	SALES	BOSTON
40	MARKETING	DALLAS

# FOREIGN KEY



FOREIGN KEY - это ссылочное ограничение, устанавливается на дочерней таблице, которое ссылается на столбец в родительской таблице.

Свойства:

- FOREIGN KEY может принимать значение из родительского столбца или NULL
- родительский столбец должен иметь ограничение PRIMARY KEY или UNIQUE.

Назначать FOREIGN KEY можно как на уровне таблице, так и на уровне столбца.

Пример определения:

```
CREATE TABLE EMP
(ID number(20,0) primary key,
ENAME varchar(50) not null,
SAL number(10) check(SAL >= 1000),
ID_DEPT number not null
REFERENCES DEPT(ID)
[NULL | ON DELETE CASCADE | ON DELETE SET NULL],
);
```

# Определение ограничения FOREIGN KEY



```
CREATE TABLE EMP
(ID number(20,0) primary key,
ENAME varchar(50) not null,
SAL number(10) check(SAL >= 1000),
ID_DEPT number not null
CONSTRAINT dept_fkey FOREIGN KEY (ID_DEPT)
REFERENCES DEPT(ID)
[NULL | ON DELETE CASCADE | ON DELETE SET NULL]
);
```

**NULL** - это значение по умолчанию, то есть при удалении родительской записи, удаление разрешается только если нет дочерних записей ссылающихся на родительскую запись,

**ON DELETE CASCADE** - при удалении родительской записи автоматически удаляются все дочерние записи,

**ON DELETE SET NULL** - при удалении родительской записи, автоматически все дочерние записи обновляются на значение NULL.

# Управление ограничениями



Для добавление ограничений к уже созданной таблице используется синтаксис:

```
ALTER TABLE table_name  
ADD [CONSTRAINT constraint_name] type (column);
```

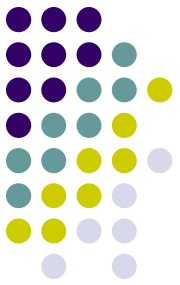
Для удаление ограничений используется синтаксис:

```
ALTER TABLE table_name  
DROP PRIMARY KEY | UNIQUE (column) |  
CONSTRAINT constraint_name [CASCADE]
```

Для просмотра всех имеющихся ограничений, используется представление **USER\_CONSTRAINTS**:

```
SELECT * FROM USER_CONSTRAINTS
```





# Список литературы:

- <http://www.sql.az/>
- <http://oracle.bulldogss.com/>
- <http://citforum.ru/>
- <http://oarcle.livejournal.com/>
- <http://www.ngpedia.ru/>