ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  |  |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| ТРИГГЕРЫ. ОБЕСПЕЧЕНИЕ АКТИВНОЙ ЦЕЛОСТНОСТИ ДАННЫХ БАЗЫ ДАННЫХ |
| по курсу: ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4132 |  |  |  | Е.И. Куконен |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

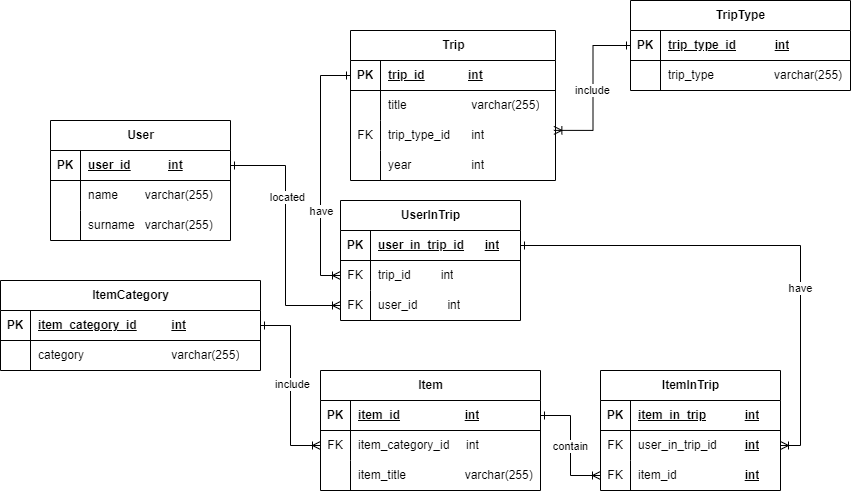
**Цель**: Создание триггеров.

**Задание**:

По аналогии с примерами, приведенными в п. 1 реализовать для своей базы данныхтриггеры для всех событий (insert,delete,update) до и после. Часть из которых будет обеспечивать ссылочную целостность, остальные могут иметь другое назначение из 3 предложенных. Вычислимые поля можно добавить при необходимости.

**Ход работы**:

Диаграмма:



**INSERT BEFORE**

Проверка года Trip

DROP TRIGGER IF EXISTS validate\_insert\_trip\_triger ON Trip;

DROP FUNCTION IF EXISTS validate\_insert\_trip();

CREATE OR REPLACE FUNCTION validate\_insert\_trip()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.year < 1900 THEN

RAISE EXCEPTION 'Invalid year';

END IF;

IF NOT EXISTS ( SELECT \* FROM TripType WHERE id = NEW.type\_id) THEN

RAISE EXCEPTION 'нет такого triptype';

END IF;

RETURN NEW;

END;

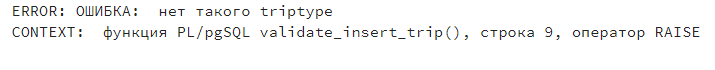
$$ LANGUAGE plpgsql;

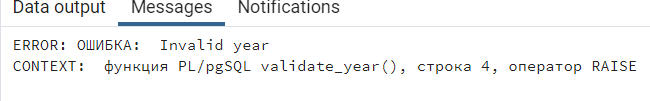
CREATE TRIGGER validate\_insert\_trip\_triger

BEFORE INSERT ON trip

FOR EACH ROW

EXECUTE FUNCTION validate\_insert\_trip();





**INSERT AFTER**

Создание логов при создании поездки

DROP TABLE IF EXISTS Logs;

CREATE TABLE Logs (

id SERIAL NOT NULL,

target\_id INT NOT NULL,

target\_table VARCHAR(255) NOT NULL,

operation\_type VARCHAR(255) NOT NULL,

operation\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (id)

);

DROP TRIGGER IF EXISTS log\_inserted\_trip\_triger ON Trip;

DROP FUNCTION IF EXISTS log\_inserted\_trip();

CREATE OR REPLACE FUNCTION log\_inserted\_trip()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO Logs (target\_id, target\_table, operation\_type, operation\_time)

VALUES (NEW.id, 'trip', 'INSERT', CURRENT\_TIMESTAMP);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER log\_inserted\_trip\_triger

AFTER INSERT ON Trip

FOR EACH ROW

EXECUTE FUNCTION log\_inserted\_trip();



**UPDATE BEFORE**

Поддержание ссылочной целостности

CREATE OR REPLACE FUNCTION check\_having\_trip\_type()

RETURNS TRIGGER AS $$

BEGIN

IF NOT EXISTS(SELECT \* FROM TripType WHERE id = NEW.type\_id) THEN

RAISE EXCEPTION 'Invalid trip type';

END IF;

RETURN NEW;

END;

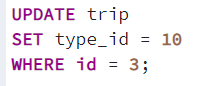
$$ LANGUAGE plpgsql;

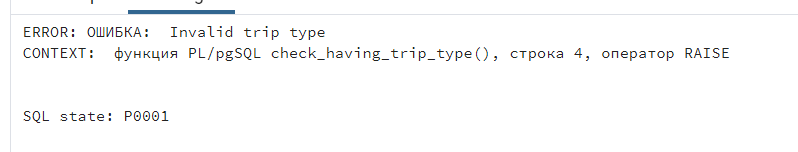
CREATE TRIGGER check\_having\_trip\_type\_trigger

BEFORE UPDATE ON Trip

FOR EACH ROW

EXECUTE FUNCTION check\_having\_trip\_type();





**UPDATE AFTER**

Создание логов обновления trip и сохранение резервной копии

CREATE TABLE Logs (

id SERIAL NOT NULL,

target\_id INT NOT NULL,

target\_table VARCHAR(255) NOT NULL,

operation\_type VARCHAR(255) NOT NULL,

operation\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (id)

);

CREATE TABLE TripModifi (

id SERIAL NOT NULL ,

title VARCHAR(255) NOT NULL,

year INT NOT NULL,

type\_id INT NOT NULL

);

DROP TRIGGER IF EXISTS log\_updated\_trip\_triger ON Trip;

DROP FUNCTION IF EXISTS log\_updated\_trip();

CREATE OR REPLACE FUNCTION log\_updated\_trip()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO Logs (target\_id, target\_table, operation\_type, operation\_time)

VALUES (NEW.id, 'trip', 'UPDATE', CURRENT\_TIMESTAMP);

INSERT INTO TripModifi ("id", title, "year", "type\_id")

VALUES (OLD."id", OLD.title, OLD."year", OLD."type\_id");

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER log\_updated\_trip\_triger

AFTER UPDATE ON Trip

FOR EACH ROW

EXECUTE FUNCTION log\_updated\_trip();



**DELETE BEFORE**

DROP TRIGGER IF EXISTS prevent\_delete\_trip\_trigger ON Trip;

DROP FUNCTION IF EXISTS prevent\_delete\_trip();

CREATE OR REPLACE FUNCTION prevent\_delete\_trip()

RETURNS TRIGGER AS $$

BEGIN

IF EXISTS (SELECT \* FROM UserInTrip WHERE trip\_id = OLD.id) THEN

RAISE EXCEPTION 'trip используется в UserInTrip';

END IF;

RETURN OLD;

END;

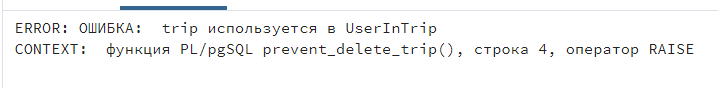
$$ LANGUAGE plpgsql;

CREATE TRIGGER prevent\_delete\_trip\_trigger

BEFORE DELETE ON Trip

FOR EACH ROW

EXECUTE FUNCTION prevent\_delete\_trip();



**DELETE AFTER**

Удаление trip с очистной справочника, логированием и бэкапом

DROP TRIGGER IF EXISTS clear\_trip\_trigger ON Trip;

DROP FUNCTION IF EXISTS clear\_trip();

CREATE OR REPLACE FUNCTION clear\_trip()

RETURNS TRIGGER AS $$

BEGIN

IF NOT EXISTS (SELECT \* FROM Trip WHERE type\_id = OLD.type\_id) THEN

DELETE FROM TripType WHERE id = OLD.type\_id;

END IF;

INSERT INTO Logs (target\_id, target\_table, operation\_type, operation\_time)

VALUES (OLD.id, 'trip', 'DELETE', CURRENT\_TIMESTAMP);

INSERT INTO TripModifi ("id", title, "year", "type\_id")

VALUES (OLD."id", OLD.title, OLD."year", OLD."type\_id");

RETURN OLD;

END;

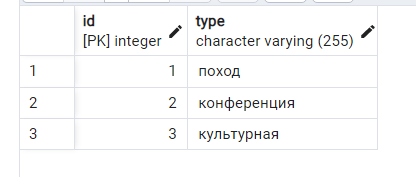
$$ LANGUAGE plpgsql;

CREATE TRIGGER clear\_trip\_trigger

AFTER DELETE ON Trip

FOR EACH ROW

EXECUTE FUNCTION clear\_trip();



**Вывод**: были созданы триггеры.