Praktikum „Integritätsbedingungen“, Phase 3

Henning Basold Igor Zerr

25. Juni 2011

1 Generierte Funktionen und Trigger

1.1 HOUSE_HOUSE_DISJOINT

```
CREATE ASSERTION HOUSE_HOUSE_DISJOINT CHECK (
    NOT EXISTS ( SELECT * FROM
        Haus AS h1, Haus AS h2
        WHERE h1.umriss && h2.umriss
    )
);

CREATE FUNCTION check_house_house_disjoint() RETURNS trigger
    LANGUAGE plpgsql
    AS $$Declare res RECORD;
BEGIN
  SELECT INTO res COUNT(*) AS num
  FROM TestSysRel
  WHERE NOT (
    NOT EXISTS ( SELECT * FROM
        Haus AS h1, Haus AS h2
        WHERE h1.umriss && h2.umriss
    )
  )
;  IF (res.num > 0)
  THEN RAISE EXCEPTION
    'ASSERTION_CHECK_HOUSE_HOUSE_DISJOINT_violated!';
  END IF;
  RETURN NEW;
END;$$;

CREATE TRIGGER check_house_house_disjoint_haus
    AFTER INSERT OR DELETE OR UPDATE ON haus
    FOR EACH ROW
    EXECUTE PROCEDURE check_house_house_disjoint();

CREATE TRIGGER check_house_lake_disjoint_haus
    AFTER INSERT OR DELETE OR UPDATE ON haus
    FOR EACH ROW
    EXECUTE PROCEDURE check_house_lake_disjoint();
```

1.2 NO_STANDALONE_STOP

```
CREATE ASSERTION NO_STANDALONE_STOP CHECK (
    NOT EXISTS ( SELECT * FROM
        Haltestelle AS b
        WHERE NOT EXISTS ( SELECT * FROM
            Strasse AS s
            WHERE approx_circle(b.position, 1) ?# s.verlauf
        )
    )
);

CREATE FUNCTION check_no_standalone_stop() RETURNS trigger
    LANGUAGE plpgsql
    AS $$Declare res RECORD;
BEGIN
  SELECT INTO res COUNT(*) AS num
  FROM TestSysRel
  WHERE NOT (
    NOT EXISTS ( SELECT * FROM
        Haltestelle AS b
        WHERE NOT EXISTS ( SELECT * FROM
            Strasse AS s
            WHERE approx_circle(b.position, 1) ?# s.verlauf
        )
    )
  )
; IF (res.num > 0)
  THEN RAISE EXCEPTION
    'ASSERTION CHECK NO_STANDALONE_STOP violated!';
  END IF;
  RETURN NEW;
END;$$;

CREATE TRIGGER check_no_standalone_stop_haltestelle
    AFTER INSERT OR DELETE OR UPDATE ON haltestelle
    FOR EACH ROW
    EXECUTE PROCEDURE check_no_standalone_stop();

CREATE TRIGGER check_no_standalone_stop_strasse
    AFTER INSERT OR DELETE OR UPDATE ON strasse
    FOR EACH ROW
    EXECUTE PROCEDURE check_no_standalone_stop();
```

2 Datenimport

3 Bestimmen der betroffenen Relationen

```
Set<String>
getAffectedTables(Connection sql, Assertion a) throws SQLException {
    Pattern tableExtraction
        = Pattern.compile(".* Scan.* on (\\w+) .*");

    Set<String> affectedTables
        = new CopyOnWriteArraySet<String>();
```

```
        Statement check = sql.createStatement();
        ResultSet pred
            = check.executeQuery(
                "EXPLAIN SELECT * FROM TestSysRel WHERE " + a.predicate);
        while(pred.next()){
            Matcher match
                = tableExtraction.matcher(pred.getString("QUERY PLAN"));
            if(match.find()){
                String table = match.group(1);
                if(!table.equalsIgnoreCase("TestSysRel")){
                    affectedTables.add(table);
                }
            }
        }

        return affectedTables;
    }
```