Praktikum „Integritätsbedingungen“, Phase 2

Henning Basold Igor Zerr

22. Mai 2011

1 Fehlerhafte Assertions

1.1 Fehlendes Schlüsselwort „Assertion“

```
CREATE HOUSE_STREET_DISJOINT CHECK (
    NOT EXISTS ( SELECT * FROM
         Haus AS h, Strasse AS s
        WHERE path(h.umriss) ?# s.verlauf
    )
);
```

```
Parse error: Parse error at 1:8:
     Expected token "ASSERTION" but got "H"
```

1.2 Fehlendes Semikolon

```
CREATE ASSERTION HOUSE_STREET_DISJOINT CHECK (
    NOT EXISTS ( SELECT * FROM
         Haus AS h, Strasse AS s
        WHERE path(h.umriss) ?# s.verlauf
    )
)
```

```
Parse error: Parse error at 8:1:
    Expected ";"
```

1.3 Fehlende schließende Klammer

```
CREATE ASSERTION HOUSE_STREET_DISJOINT CHECK (
    NOT EXISTS ( SELECT * FROM
         Haus AS h, Strasse AS s
        WHERE path(h.umriss) ?# s.verlauf
    )
;
```

```
Error in assertion HOUSE_STREET_DISJOINT on line 1:
    error in predicate at position 132: Syntaxfehler am Ende der Eingabe
```

1.4 Syntaktisch inkorrekter Bezeichner

```
CREATE ASSERTION 1234 CHECK (
    NOT EXISTS ( SELECT * FROM
         Haus AS h, Strasse AS s
        WHERE path(h.umriss) ?# s.verlauf
    )
);
```

```
Parse error: Parse error at 1:18:
    Expected identifier but got "1234"
```

1.5 Schlüsselwort als Bezeichner

```
CREATE ASSERTION select CHECK (
    NOT EXISTS ( SELECT * FROM
         Haus AS h, Strasse AS s
        WHERE path(h.umriss) ?# s.verlauf
    )
);
```

```
Error in assertion select on line 1:
    "select" is an invalid SQL identifier
```

1.6 Syntaktisch ungültiges Prädikat

```
CREATE ASSERTION HOUSE_STREET_DISJOINT CHECK (
    CREATE Inv (Field INTEGER)
);
```

```
Error in assertion HOUSE_STREET_DISJOINT on line 1:
    syntactic error in predicate at position 32 near "CREATE".
```

1.7 Fehlende Tabelle in Prädikat

```
CREATE ASSERTION HOUSE_STREET_DISJOINT CHECK (
    NOT EXISTS ( SELECT * FROM
        XYZ AS h, Strasse AS s
        WHERE path(h.umriss) ?# s.verlauf
    )
);
```

```
Error in assertion HOUSE_STREET_DISJOINT on line 1:
    error in predicate at position 67: Relation "xyz" does not exist.
```

1.8 Fehlende Spalte in Prädikat

```
CREATE ASSERTION HOUSE_STREET_DISJOINT CHECK (
    NOT EXISTS ( SELECT * FROM
         Haus AS h, Strasse AS s
        WHERE path(h.xyz) ?# s.verlauf
    )
);
```

```
Error in assertion HOUSE_STREET_DISJOINT on line 1:
    error in predicate at position 110: Spalte "h.xyz" does not exist.
```

1.9 Ungültiger Operator in Prädikat

```
CREATE ASSERTION invPred CHECK (
    NOT EXISTS ( SELECT * FROM
        Haus AS h, Strasse AS s
        WHERE path(h.umriss) ?+ s.verlauf
    )
);
```

```
Error in assertion invPred on line 1:
    error in predicate at position 120: Operator existiert nicht: path ?+ path
  Hinweis: Kein Operator stimmt mit dem angegebenen Namen und den Argumenttypen überein.
  Sie müssen möglicherweise ausdrückliche Typumwandlungen hinzufügen.
```

1.10 Mehrfaches Einfügen

Die erste Assertion ist bereits in der Datenbank eingetragen, hat aber das gleiche Prädikat. Die zweite hat ein anderes Prädikat, aber den gleichen Namen.

```
CREATE ASSERTION HOUSE_STREET_DISJOINT CHECK (
    NOT EXISTS ( SELECT * FROM
        Haus AS h, Strasse AS s
        WHERE path(h.umriss) ?# s.verlauf
    )
);

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

```
Warning: assertion HOUSE_STREET_DISJOINT on line 1 already exists with same predicate.
    It is not inserted again.
Error in assertion HOUSE_STREET_DISJOINT on line 8:
    assertion already exists with another predicate.
```

2 Korrekte Assertions

```
CREATE ASSERTION HOUSE_STREET_DISJOINT CHECK (
    NOT EXISTS ( SELECT * FROM
        Haus AS h, Strasse AS s
        WHERE path(h.umriss) ?# s.verlauf
    )
);

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

```
CREATE ASSERTION HOUSE_LAKE_DISJOINT CHECK (
    NOT EXISTS ( SELECT * FROM
        Haus AS h, See AS l
        WHERE h.umriss && l.umriss
    )
);

CREATE ASSERTION STREET_LAKE_DISJOINT CHECK (
    NOT EXISTS ( SELECT * FROM
        Strasse AS s, See AS l
        WHERE s.verlauf ?# path(l.umriss)
    )
);

CREATE ASSERTION STREET_LANDUSE_DISJOINT CHECK (
    NOT EXISTS ( SELECT * FROM
        Strasse AS s, Landnutzung AS l
        WHERE s.verlauf ?# path(l.umriss)
    )
);

CREATE ASSERTION STREET_PLAYGROUND_DISJOINT CHECK (
    NOT EXISTS ( SELECT * FROM
        Strasse AS s, Spielplatz AS p
        WHERE s.verlauf ?# path(p.umriss)
    )
);

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 ASSERTION TRAFFIC_LIGHT_AT_STREET CHECK (
    NOT EXISTS ( SELECT * FROM
        Ampel AS t
        WHERE NOT EXISTS ( SELECT * FROM
            Strasse AS s
            WHERE approx_circle(t.position, 1) ?# s.verlauf
        )
    )
);

CREATE ASSERTION PARKING_REACHABLE CHECK (
    NOT EXISTS ( SELECT * FROM
        Parkplatz AS p
        WHERE NOT EXISTS ( SELECT * FROM
            Strasse AS s
            WHERE approx_circle(p.position, 1) ?# s.verlauf
```

```
          )
     )
);
```

```
Warning: assertion HOUSE_STREET_DISJOINT on line 1 already exists with same predicate.
    It is not inserted again.
Warning: assertion HOUSE_HOUSE_DISJOINT on line 8 already exists with same predicate.
    It is not inserted again.
Warning: assertion HOUSE_LAKE_DISJOINT on line 15 already exists with same predicate.
    It is not inserted again.
Warning: assertion STREET_LAKE_DISJOINT on line 22 already exists with same predicate.
    It is not inserted again.
Warning: assertion STREET_LANDUSE_DISJOINT on line 29 already exists with same predicate.
    It is not inserted again.
Warning: assertion STREET_PLAYGROUND_DISJOINT on line 36 already exists with same predicate.
    It is not inserted again.
Warning: assertion NO_STANDALONE_STOP on line 43 already exists with same predicate.
    It is not inserted again.
Warning: assertion TRAFFIC_LIGHT_AT_STREET on line 53 already exists with same predicate.
    It is not inserted again.
Warning: assertion PARKING_REACHABLE on line 63 already exists with same predicate.
    It is not inserted again.
Assertions have been successfully checked and saved.
```

Datenbankinhalt:

```
pg_dump -a -t AssertionSysRel rdb_praktikum

--
-- PostgreSQL database dump
--

SET statement_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = off;
SET check_function_bodies = false;
SET client_min_messages = warning;
SET escape_string_warning = off;

SET search_path = public, pg_catalog;

--
-- Data for Name: assertionsysrel; Type: TABLE DATA; Schema: public;
    Owner: ***
--

COPY assertionsysrel (assertionname, bedingung, implementiert) FROM
    stdin;
HOUSE_STREET_DISJOINT   NOT EXISTS ( SELECT * FROM\n          Haus AS h,
      Strasse AS s\n        WHERE path(h.umriss) ?# s.verlauf\n     )    f
HOUSE_HOUSE_DISJOINT    NOT EXISTS ( SELECT * FROM\n          Haus AS h1
    , Haus AS h2\n        WHERE h1.umriss && h2.umriss\n     ) f
HOUSE_LAKE_DISJOINT     NOT EXISTS ( SELECT * FROM\n          Haus AS h,
      See AS l\n        WHERE h.umriss && l.umriss\n     )       f
```

```
STREET_LAKE_DISJOINT      NOT EXISTS ( SELECT * FROM\n          Strasse AS
     s, See AS l\n          WHERE s.verlauf ?# path(l.umriss)\n     )     f
STREET_LANDUSE_DISJOINT NOT EXISTS ( SELECT * FROM\n          Strasse AS
     s, Landnutzung AS l\n          WHERE s.verlauf ?# path(l.umriss)\n
       )     f
STREET_PLAYGROUND_DISJOINT       NOT EXISTS ( SELECT * FROM\n
    Strasse AS s, Spielplatz AS p\n          WHERE s.verlauf ?# path(p.
    umriss)\n     )      f
NO_STANDALONE_STOP        NOT EXISTS ( SELECT * FROM\n
    Haltestelle AS b\n          WHERE NOT EXISTS ( SELECT * FROM\n
                Strasse AS s\n              WHERE approx_circle(b.position
    , 1) ?# s.verlauf\n          )\n     )f
TRAFFIC_LIGHT_AT_STREET NOT EXISTS ( SELECT * FROM\n          Ampel AS t
    \n          WHERE NOT EXISTS ( SELECT * FROM\n              Strasse AS
    s\n              WHERE approx_circle(t.position, 1) ?# s.verlauf\n
          )\n     )        f
PARKING_REACHABLE         NOT EXISTS ( SELECT * FROM\n          Parkplatz
    AS p\n          WHERE NOT EXISTS ( SELECT * FROM\n              Strasse
     AS s\n              WHERE approx_circle(p.position, 1) ?# s.verlauf\
    n          )\n     )     f
\.


--
-- PostgreSQL database dump complete
--
```

3 Prüfung der Bezeichner

3.1 Prüfung beim Parsen

```
identifierPattern = Pattern.compile("[_a-zA-Z][_a-zA-Z0-9]*");

⋮

private String
parseIndentifier(InputStreamIterator in) throws AssertionParseError {
    int oldLine = line;
    int oldColumn = column;

    StringBuffer word = new StringBuffer();

    if(!isWS(next)){
        word.appendCodePoint(next.intValue());
    }

    while(in.hasNext()) {
        nextChar(in);
        if(!isWS(next)){
            word.appendCodePoint(next.intValue());
        }
        else{
            break;
        }
    }

    Matcher idMatcher = identifierPattern.matcher(word);
    if(idMatcher.matches()){
        return word.toString();
    }
    else{
        throw new AssertionParseError("Expected␣identifier␣but␣got␣\""
                                        + word + "\"", oldLine, oldColumn);
    }
}
```

3.2 Prüfung auf Verträglichkeit mit SQL

```
syntaxErrorParser
    = Pattern.compile(".*␣Syntaxfehler␣bei␣»(.*)«\\s+ Position: (\\d+).*");
```

⋮

```
private String checkName(String name) throws SQLException {
    Statement create = sql.createStatement();
    try {
        create.executeUpdate("CREATE␣TABLE␣" + name + "␣(Attribut␣INTEGER)");
        create.executeUpdate("DROP␣TABLE␣" + name);
    }
    catch (SQLException e) {
        // Leider funktionieren die Fehlercodes etc.
        // mit dem Postgres-Backend scheinbar nicht...
        Matcher m = syntaxErrorParser.matcher(e.getMessage());
        if(m.matches()){
            return "\"" + name + "\"␣is␣an␣invalid␣SQL␣identifier";
        }
        else{
            throw e;
        }
    }
    finally {
        create.close();
    }

    return null;
}
```