Correctness

Knowledge Objectives

1. Define four logic properties of integrity constraints (i.e. schema satisfiability, liveliness, constraint redundancy and state reachability)
2. Exemplify the three necessary conditions for summarizability

Application Objectives

1. Find and eventually fix the problems related to the five logical properties of integrity constraints in a given relational schema (with at most 6 tables and views)

Problems in the constraints

- Contradictory constraints generate empty tables/views
  - May even result in empty databases

    CHECK (a<10 AND a>20)

- Redundant constraints slow down DBMS performance
  1. CHECK (a>10)
  2. CHECK (a>20)

Logic properties of constraints

- *Schema-satisfiability*: A schema is satisfiable if there is at least one consistent DB state containing tuples (i.e. each and every constraint is fulfilled)
- *Liveliness*: A table/view is lively if there is at least one consistent DB state, so that the table/view contains tuples
- *State-reachability*: A given set of tuples is reachable if there is at least one consistent DB state containing those tuples (and maybe others)
- *Redundancy*: a constraint is redundant if it is a logic consequence of other constraints

Example of Schema-satisfiability

```
CREATE TABLE employees (
    id CHAR(9) PRIMARY KEY,
    dpt VARCHAR(4) NOT NULL
    );

CREATE TABLE departments (
    id VARCHAR(4) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    basicSalary INT                 ,
    CONSTRAINT ckMinSalary CHECK (basicSalary>2000),
    CONSTRAINT ckMaxSalary CHECK (basicSalary<1000)
    );
```

Example of Schema-satisfiability

```
CREATE TABLE employees (
    id CHAR(9) PRIMARY KEY,
    dpt VARCHAR(4) NOT NULL REFERENCES departments (ID)
    );

CREATE TABLE departments (
    id VARCHAR(4) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    basicSalary INT                 ,
    CONSTRAINT ckMinSalary CHECK (basicSalary>2000),
    CONSTRAINT ckMaxSalary CHECK (basicSalary<1000)
    );
```

Example of Schema-satisfiability

```
CREATE TABLE employees (
    id CHAR(9) PRIMARY KEY,
    dpt VARCHAR(4) NOT NULL REFERENCES departments (ID)
    );

CREATE TABLE departments (
    id VARCHAR(4) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    basicSalary INT NOT NULL ,
    CONSTRAINT ckMinSalary CHECK (basicSalary>2000),
    CONSTRAINT ckMaxSalary CHECK (basicSalary<1000)
    );
```

Example of Liveliness

```
CREATE TABLE departments (
    id VARCHAR(4) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    basicSalary INT NOT NULL,
    CONSTRAINT ckMinSalary CHECK (basicSalary>2000));

CREATE TABLE employees (
    id CHAR(9) PRIMARY KEY,
    dpt VARCHAR(4)              REFERENCES departments (id));

CREATE VIEW unassigned AS (
    SELECT *
    FROM employees e
    WHERE NOT EXISTS (          SELECT *
                                FROM departments d
                                WHERE d.id=e.dpt));
```

Example of Liveliness

```
CREATE TABLE departments (
    id VARCHAR(4) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    basicSalary INT NOT NULL,
    CONSTRAINT ckMinSalary CHECK (basicSalary>2000));

CREATE TABLE employees (
    id CHAR(9) PRIMARY KEY,
    dpt VARCHAR(4) NOT NULL REFERENCES departments (id));

CREATE VIEW unassigned AS (
    SELECT *
    FROM employees e
    WHERE NOT EXISTS (          SELECT *
                                FROM departments d
                                WHERE d.id=e.dpt));
```

Example of Redundancy

```
CREATE TABLE departments (
    id VARCHAR(4) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    basicSalary INT NOT NULL,
    CONSTRAINT ckMinSalary CHECK (basicSalary>2000),
    CONSTRAINT ckDeptName CHECK (id<>'CS')
    );


CREATE TABLE employees (
    id CHAR(9) PRIMARY KEY,
    dpt VARCHAR(4) NOT NULL REFERENCES departments (ID),
    CONSTRAINT ckEmpName CHECK (dpt<>'CS')
    );
```

Example of Redundancy

```
CREATE TABLE departments (
    id VARCHAR(4) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    basicSalary INT NOT NULL,
    CONSTRAINT ckMinSalary CHECK (basicSalary>2000),
    CONSTRAINT ckDeptName CHECK (id<>'CS')
    );


CREATE TABLE employees (
    id CHAR(9) PRIMARY KEY,
    dpt VARCHAR(4) NOT NULL REFERENCES departments (ID),

    );
```

Example of State-reachability

```
CREATE TABLE departments (
    id VARCHAR(4) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    basicSalary INT NOT NULL,
    CONSTRAINT ckMinSalary CHECK (basicSalary>2000));


CREATE TABLE employees (
    id CHAR(9) PRIMARY KEY,
    dpt VARCHAR(4) NOT NULL REFERENCES departments (ID));
```

```
Employees(id,    dpt); Departments(    id,    name, basicSalary)
              1      CS                      CS     Compu…    10000
              2      MK
```

Example of State-reachability

```
CREATE TABLE departments (
    id VARCHAR(4) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    basicSalary INT NOT NULL,
    CONSTRAINT ckMinSalary CHECK (basicSalary>2000));


CREATE TABLE employees (
    id CHAR(9) PRIMARY KEY,
    dpt VARCHAR(4) NOT NULL REFERENCES departments (ID));
```

Employees(id,	dpt);	Departments(	id,	name,	basicSalary)
1	CS		CS	Compu…	10000
2	MK		MK	Market…	2001

Aggregation problems (I)

- Number of students per department and year, assuming the students follow a two-year program

	1994	1995	1996	All
Informatics	15	17	13	28
Statistics	10	15	11	21
All	25	32	24	49

Aggregation problems (I)

- Number of students per department and year, assuming the students follow a two-year program

	1994	1995	1996	All
Informatics	15	17	13	28
Statistics	10	15	11	21
All	25	32	24	49

- Number of students per department and year, assuming the students follow a two-year program where there are inter-department courses

	1994	1995	1996	All
Informatics	15	17	13	28
Statistics	10	15	11	21
All	23	30	24	47

Aggregation problems (II)

- Number of car accidents per province chief town and year

	1994	1995	1996	All
Barcelona	5	6	3	14
Tarragona	1	0	1	2
Lleida	0	2	1	3
Girona	3	5	6	14
Catalunya	20	23	22	65

Aggregation problems (III)

	Cumulative	State	Value per unit
min	No problem	No problem	No problem
max	No problem	No problem	No problem
sum	No problem	Non-temporal	Never
avg	No problem	No problem	No problem

Summary

- Logic properties of constraints
  - Schema satisfiability
  - Lifeliness
  - Redundancy
  - State-reachability
- Aggregation problems
  - Summarizability necessary conditions

Bibliography

- Ernest Teniente, et al. *SVT: Schema Validation Tool for Microsoft SQL-Server*. Conference on Very Large Databases (VLDB), 2004
- H. J. Lenz and A. Shoshani. *Summarizability in OLAP and statistical databases*. In *Proceedings of SSDBM'1997*. IEEE, 1997