Constraints in SQL

1. Key constraints (PRIMARY KEY and UNIQUE)
2. Referential integrity constraints (FOREIGN KEY … REFERENCES)
3. Constraints on values of attributes:
   - NOT NULL
   - attribute-based CHECK constraints

Attribute-Based CHECK Constraints

- Use keyword CHECK followed by a parenthesized condition that must be true for every value of this attribute.
- In practice (good style): condition is simple limit on values.
- In general (not necessarily good style): condition can be anything that could follow WHERE in a SQL query.
Attribute-Based CHECK Constraints

Example of good-style use:

CREATE TABLE Products ( 
    product_no INTEGER, 
    name CHAR(255), 
    price REAL 
    PRIMARY KEY (product_no) 
);
Attribute-Based CHECK Constraints

Example of good-style use:

CREATE TABLE Products (  
    product_no INTEGER,  
    name CHAR(255),  
    price REAL CHECK (price > 0),  
    PRIMARY KEY (product_no)  
);
Attribute-Based CHECK Constraints

When is an attribute-based CHECK constraint checked by the DBMS?

• **Only** when a tuple gets a new value for the attribute to which the constraint is attached:
  • update of the corresponding attribute
  • insertion of a new tuple in the table where the constraint is defined

• If constraint is violated, then update or insertion is **rejected**
(Bad-Style) Attribute-Based CHECK Constraints

Attribute-based CHECK constraints can mention other attributes or relations (bad style): need a subquery for this
(Bad-Style) Attribute-Based CHECK Constraints

Example of bad-style use: Suppose that we want to simulate (incorrectly) a foreign-key constraint:
Example of bad-style use: Suppose that we want to simulate \textit{(incorrectly)} a foreign-key constraint:

```
CREATE TABLE Works_In (  
  ssn INTEGER,  
  did INTEGER CHECK (did IN (SELECT D.did  
                             FROM Departments D)),  
  PRIMARY KEY (ssn, did),  
  FOREIGN KEY (ssn) REFERENCES Employees);
```

Why doesn’t this work?
(Bad-Style) Attribute-Based CHECK Constraints

- **Answer:** CHECK is only associated with modifications of attribute in question (i.e., did in Works_In), *not* with every relation and attribute mentioned.

- So deletions and updates in the Departments table are a problem, because they don’t cause the DBMS to verify that the CHECK constraint still holds …
Constraints in SQL

1. Key constraints (PRIMARY KEY and UNIQUE)
2. Referential integrity constraints (FOREIGN KEY ... REFERENCES)
3. Constraints on values of attributes:
   - NOT NULL
   - attribute-based CHECK constraints
4. Constraints on values of tuples of a relation:
   - tuple-based CHECK constraints (or table constraints)
Tuple-Based CHECK Constraints

Like attribute-based CHECK constraints, but associated with each tuple of one table.
Tuple-Based CHECK Constraints

Example of good-style use:

```
CREATE TABLE Products (  
    product_no INTEGER,  
    name CHAR(255),  
    price REAL,  
    discounted_price REAL,  
    PRIMARY KEY (product_no)  
);```
Tuple-Based CHECK Constraints

Example of good-style use:

CREATE TABLE Products (  
    product_no INTEGER,  
    name CHAR(255),  
    price REAL,  
    discounted_price REAL,  
    CHECK (price > discounted_price),  
    PRIMARY KEY (product_no)  
);
Tuple-Based CHECK Constraints

When is a tuple-based CHECK constraint checked by the DBMS?

- **Only** when a tuple is inserted into the corresponding table and when a tuple is updated in the corresponding table, and the CHECK constraint is evaluated only for the new or updated tuple

- If constraint is violated, then insertion or update is **rejected**
(Bad-Style) Tuple-Based CHECK Constraints

Tuple-based CHECK constraints can mention other relations (bad style): need a subquery for this
(Bad-Style) Tuple-Based CHECK Constraints

Example of bad-style use: Suppose that we want to enforce (incorrectly) the total participation constraint of Employees in Works_In:

CREATE TABLE Employees (  
  ssn INTEGER,  
  name CHAR(30),  
  PRIMARY KEY (ssn),  
)
(Bad-Style) Tuple-Based CHECK Constraints

Example of bad-style use: Suppose that we want to enforce (incorrectly) the total participation constraint of Employees in Works_In:

CREATE TABLE Employees (  
  ssn INTEGER,  
  name CHAR(30),  
  PRIMARY KEY (ssn),  
  CHECK (ssn IN (SELECT W.ssn  
              FROM Works_In W)));

Why doesn’t this work?
(Bad-Style) Tuple-Based CHECK Constraints

- **Answer:** CHECK is only associated with insertions or updates of tuples in relation where the CHECK constraint is defined (i.e., Employees), **not** with every relation mentioned.

- So deletions or updates in the Works_In table are a problem, because they don’t cause the DBMS to verify that the CHECK constraint still holds ...
Constraints in SQL

1. Key constraints (PRIMARY KEY and UNIQUE)
2. Referential integrity constraints (FOREIGN KEY … REFERENCES)
3. Constraints on values of attributes:
   - NOT NULL
   - attribute-based CHECK constraints
4. Constraints on values of tuples of a relation:
   - tuple-based CHECK constraints (or table constraints)
5. Schema-level constraints:
   - assertions
Assertions

- To enforce any condition (that can follow `WHERE`)
- Not associated with tables or attributes
- **Guaranteed to always hold:** any database modification, anywhere, that would make an assertion false is rejected
- **Expensive; not implemented universally** (PostgreSQL …)
Assertions

We can now finally model (correctly) the total participation constraint of Employees in Works_In:
Assertions

We can now finally model (correctly) the total participation constraint of Employees in Works_In:

CREATE ASSERTION TotalPartEmpWorksIn CHECK (NOT EXISTS (SELECT * FROM Employees E WHERE E.ssn NOT IN (SELECT W.ssn FROM Works_In W)));

![Diagram showing the relationship between Employees, Works_In, and Departments with attributes ssn: integer, name: string, did: string, dname: string.](image-url)
## Summary

<table>
<thead>
<tr>
<th>Type of constraint</th>
<th>Where declared</th>
<th>When activated</th>
<th>Guaranteed to hold?</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute-based CHECK</td>
<td>with attribute</td>
<td>on insertion to relation or attribute update</td>
<td>not if subqueries</td>
</tr>
<tr>
<td>tuple-based CHECK</td>
<td>with relation schema</td>
<td>on insertion to relation or tuple update</td>
<td>not if subqueries</td>
</tr>
<tr>
<td>assertion</td>
<td>with database schema</td>
<td>on change to any mentioned relation</td>
<td>yes</td>
</tr>
</tbody>
</table>