Inhalt

[ACID 2](#_Toc95153663)

[Integrität/Konsistenz 2](#_Toc95153664)

[Semantische Integrität 3](#_Toc95153665)

[Referentielle Integrität 3](#_Toc95153666)

[Sicherstellung durch Datenintegrität und Constraints 4](#_Toc95153667)

[Quellen 7](#_Toc95153668)

# ACID

**Atomicity oder Atomarität:** Eine Transaktion besteht aus einer Sequenz einzelner Aktionen. Eine Sequenz muss so ablaufen, dass entweder alle Einzelschritte komplett oder gar nicht ausgeführt werden. Wenn während einer Sequenz Fehler auftreten, muss dafür gesorgt werden, dass alle erfolgten Änderungen zurückgenommen werden. Eine Transaktion ist erst gültig, wenn sie alle erfolgreich abgeschlossen wurden. Durch Logging aller durchgeführten Aktionen wird die Atomarität realisiert.

**Consistency oder Konsistenz:** Wenn eine Transaktion erfolgreich abgeschlossen wurde, muss sie in der zuvor konsistenten Datenbank einen wieder konsistenten Zustand hinterlassen werden. Die Konsistenz muss vor und nach der Transaktion sichergestellt werden, allerdings dürfen während einer Transaktion durchaus inkonsistente Zustände auftreten. Wenn eine Transaktion gegen die Konsistenzbedingungen verstößt, wird diese zurückgewiesen und sämtliche Daten werden auf dem Zustand vor der Transaktion zurückgesetzt. Solche Bedingungen können die Einhaltung bestimmter Wertebereiche, das Vorhandsein von Schlüsseleigenschaften oder die Eindeutigkeit von Beziehungen sein.

**Isolation oder Abgrenzung:** Das man eine Datenbank nur für sich allein hat, ist eher die Seltenheit, deswegen muss auch auf den Mehrbenutzerbetrieb Rücksicht genommen werden. Durch die Isolation wird sichergestellt, dass sich die Benutzer nicht negativ gegenseitig beeinflussen. Datenbanksysteme führen die Isolation mithilfe von Sperrverfahren.

**Durability oder Dauerhaftigkeit**: Wurde eine Transaktion ausgeführt und ist konsistent, sind die Informationen dauerhaft in der Datenbank gespeichert. Speicherausfälle, Systemabstürze oder andere zukünftige Fehler dürfen nicht dazu führen, dass Daten gelöscht werden und nicht mehr hergestellt werden können.

# Integrität/Konsistenz

Die **Datenintegrität/Integrität** kann durch sogenannte Constraints geregelt werden. Diese Regeln bestimmen im DBMS wie die Daten verändert werden dürfen.

Unter **Konsistenz** versteht man die Korrektheit der gespeicherten Daten, alle Clients haben die gleiche Sicht auf den Datenbestand -auch im Fall von Updates.

Datenintegrität lässt sich auf zwei verschiedene Arten in SQL-Server implementieren.

* Deklarativ
* Prozedural

**Entitätenintegrität**

Hierbei wird sichergestellt, dass jede Zeile einer Tabelle sich eindeutig identifizieren lässt. Dies kann mit folgenden Constraints bewerkstelligt werden.

* Primary Key (Surrogate Key generierter Schlüssel, Composite Key zusammengesetzter Schlüssel)
* Unique
* Index

# Semantische Integrität

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



# Referentielle Integrität

Dazu werden Regeln aufgestellt, wie und unter welchen Bedingungen ein Datensatz in die Datenbank eingetragen wird. Bei der referentiellen Integrität können Datensätze, die einen Fremdschlüssel aufweisen nur dann gespeichert werden, wenn der Wert des Fremdschlüssels einmalig in der referenzierten Tabelle existiert.

**“Warum wird die Referentielle Integrität benötigt?“**

Es können Anomalien im Datenbestand auftreten, die verschiedene Formen annehmen. Man spricht hier von Einfüge-, Lösch- und Änderungsanomalien. Tritt eine oder mehrerer dieser Anomalien auf, kann das zur Verfälschung oder Löschung von Informationen führen.

**Einfüge-Anomalien in einer Datenbank**

Trifft auf, wenn keiner oder kein eindeutiger Primärschlüssel vorliegt.

Ein Bild, das Text, Gerät, Anzeige, Screenshot enthält.

Automatisch generierte Beschreibung

Die Tabelle OrderPerson verweist auf die Person Tabelle, und es versucht eben diese Person zu löschen, hier trifft eine Speicheranomalie auf.

# Sicherstellung durch Datenintegrität und Constraints

Siehe import.sql

# Deklarative Datenintegrität (not null, unique, primary key, index,  
# define length, check, FOREIGN KEY, Delete, update, insert  
  
  
## Semantische Integrität  
drop table Student;  
create table Student  
(  
 id int(30),  
 name int  
);  
describe Student;  
  
insert into Student(id, name)  
VALUES (1, 'test');  
  
select \* from Student;  
  
# Entitätenintegrität  
select \*  
from Customer;  
DESCRIBE Customer;  
drop table Customer;  
  
# not null  
create table Customer  
(  
 ID int,  
 LastName varchar,  
 FirstName varchar,  
 Age int  
);  
  
insert into Customer(ID, LastName, FirstName, Age)  
VALUES (null, null, null, null);  
  
  
# unique attribute (darf allerdings null sein, im Gegensatz zum Primary Key)  
create table Customer  
(  
 ID int unique,  
 LastName varchar,  
 FirstName varchar,  
 Age int  
);  
  
  
  
insert into Customer(ID, LastName, FirstName, Age)  
values (1, 'Hieselmayr', 'David', 1);  
insert into Customer(ID, LastName, FirstName, Age)  
values (1, 'Berger', 'David', 1);  
  
# primary key  
alter table Customer  
 add primary key (ID);  
  
# primary key composite key  
alter table Customer  
 add primary key (ID, LastName);  
  
# remove primary key  
alter table Customer  
 drop PRIMARY KEY;  
  
# create index  
create index idx\_lastname on Customer (LastName);  
  
  
  
drop table Customer;  
# primary key  
create table Customer  
(  
 ID int primary key auto\_increment,  
 LastName varchar,  
 FirstName varchar,  
 Age int  
);  
  
insert into Customer(LastName, FirstName, Age)  
values ('Dieselmayr', 'David', 10);  
insert into Customer(LastName, FirstName, Age)  
values ('Benzinmayer', 'David', 21);  
insert into Customer(LastName, FirstName, Age)  
values ('Emayer', 'David', 22);  
  
  
# Domänenintegrität  
  
# define length  
CREATE TABLE Person  
(  
 ID int primary key auto\_increment,  
 LastName varchar(2) NOT NULL,  
 FirstName varchar(2),  
 Age int,  
 City varchar(255) DEFAULT 'Bad Hall'  
);  
  
describe Person;  
  
insert into Person(LastName, FirstName, Age)  
VALUES ('Gruber', 'Hans', 2);  
insert into Person(LastName, FirstName, Age, City)  
VALUES ('Gruber', 'Hans', 2, 'Steyr');  
  
alter table Person  
 modify FirstName varchar(50);  
alter table Person  
 modify LastName varchar(60);  
  
# check constraint  
drop table Person;  
CREATE TABLE Person  
(  
 ID int primary key auto\_increment,  
 LastName varchar(60) NOT NULL,  
 FirstName varchar(50),  
 City varchar(255) DEFAULT 'Bad Hall',  
 Age int,  
 check ( Age >= 18)  
);  
  
insert into Person(LastName, FirstName, Age)  
values ('Linsenmayr', 'David', 17);  
insert into Person(LastName, FirstName, Age)  
values ('Linsenmayr', 'David', 18);  
  
select \*  
from Person;  
  
  
#FOREIGN KEY  
CREATE TABLE OrderPerson  
(  
 OrderPersonID int NOT NULL PRIMARY KEY,  
 name varchar(40) not null,  
 personId Int,  
 CONSTRAINT person\_fk foreign key (personId) references Person (id)  
);  
  
describe OrderPerson;  
  
select \*  
from Person;  
  
insert into OrderPerson(OrderPersonID, name, personId)  
values (1, 'LKW', 1);  
  
select O.OrderPersonID, O.name, P.FirstName  
from OrderPerson O  
 left join Person P on O.personId = P.ID;  
  
  
  
## Referentielle Integrität  
  
# passt  
DELETE  
From OrderPerson  
where OrderPersonID = 1;  
select \*  
from OrderPerson;  
insert into OrderPerson(OrderPersonID, name, personId)  
values (1, 'LKW', 1);  
  
# ALARM  
DELETE  
From Person  
where ID = 1;  
# GEHT  
UPDATE Person  
set FirstName='Wieselmayr'  
where ID = 1;  
;

# Quellen

<https://www.bigdata-insider.de/was-ist-acid-a-776182/>

<https://www.datenbanken-verstehen.de/datenmodellierung/referentielle-integritaet/>

<https://www.insidesql.org/blogs/frankkalis/2004/07/16/datenintegritaet>