Datenbanksysteme 1 | Dbs1

Zusammenfassung

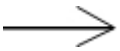
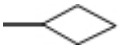
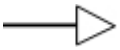
INHALTSVERZEICHNIS

1. GLOSSAR

Begriff	*Bedeutung*
Entität	Ein eindeutig identifizierbares Objekt oder Konzept, z. B. eine Person oder ein Produkt.
Entitätsmenge	Eine Gruppe von ähnlichen Entitäten, die durch gemeinsame Eigenschaften gekennzeichnet sind.
Kardinalität von Beziehungen	Gibt an, wie viele Entitäten an einer Beziehung beteiligt sind (z. B. 1:1, 1:n, n:m).
Kardinalität von Attributen	Bezieht sich auf die Anzahl der Werte, die ein Attribut annehmen kann.
Basisdatentyp	Die grundlegenden Datentypen in einer Datenbank, wie Integer, String oder Boolean.
Mehrwertiges Attribut	Ein Attribut, das mehrere Werte für ein einzelnes Entitätsobjekt speichern kann, z. B. Telefonnummern.
Schwache Entity Menge	Eine Menge von Entitäten, die ohne eine zugehörige starke Entität nicht existieren kann.
Spezialisierung	Der Prozess, bei dem von einer allgemeinen Entität eine spezifischere Entität abgeleitet wird.
Surrogate Key	Ein künstlicher Schlüssel, der verwendet wird, um eine Entität zu identifizieren, ohne natürliche Eigenschaften zu verwenden.
Zusammengesetztes Attribut	Ein Attribut, das aus mehreren Unterattributen besteht, z. B. eine Adresse (Straße, Stadt, PLZ).
Zweitschlüssel	Ein Attribut oder eine Kombination von Attributen, die nicht primär, aber eindeutig identifizieren können.
Datenbank	Eine organisierte Sammlung von Daten, die in einem DBMS verwaltet wird.
System-Katalog	Ein internes Repository, das Metadaten über die Datenbankobjekte enthält, z. B. Tabellen und Schemata.
Datenbankschema	Die logische Struktur einer Datenbank, die die Organisation der Daten und deren Beziehungen beschreibt.
Datenbasis	Der physische Speicherort, an dem die Daten der Datenbank tatsächlich gespeichert sind.
Datenunabhängigkeit	Die Fähigkeit, Daten in einer Datenbank zu ändern, ohne dass Änderungen an den Anwendungen erforderlich sind.
DBMS (Datenbankmanagementsystem)	Software, die die Erstellung, Verwaltung und Nutzung einer Datenbank ermöglicht.
DBS (Datenbanksystem)	Ein allgemeiner Begriff für eine Kombination aus DBMS und den dazugehörigen Datenbanken.
Implementierungsschema	Die spezifische Struktur und Organisation der Daten auf physischer Ebene innerhalb eines DBMS.
Impedance-Mismatch	Ein Problem, das auftritt, wenn zwei Systeme unterschiedliche Datenmodelle verwenden, wodurch die Interaktion erschwert wird.

Begriff	Bedeutung
Konsistenz	Der Zustand, in dem die Datenbankregeln eingehalten werden, und die Daten integrativ und fehlerfrei sind.
ODBMS (Objektorientiertes Datenbankmanagementsystem)	Ein DBMS, das objektorientierte Programmierkonzepte für die Datenverwaltung nutzt.
ORDBMS (Objekt-relationales Datenbankmanagementsystem)	Eine Kombination aus objektorientierten und relationalen Prinzipien in der Datenbankverwaltung.
Persistenz	Die Fähigkeit von Daten, über die Laufzeit eines Programms hinaus zu bestehen.
RDBMS (Relationales Datenbankmanagementsystem)	Ein DBMS, das Daten in tabellarischer Form speichert und relationalen Zugriff ermöglicht.

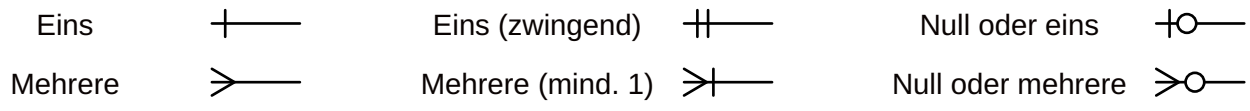
2. UML

Begriff	Bedeutung
Assoziation 	Eine Beziehung zwischen zwei oder mehr Klassen, die zeigt, wie Objekte dieser Klassen miteinander verbunden sind.
Aggregation 	Eine spezielle Form der Assoziation, die eine «Teil-von» Beziehung darstellt, bei der die Lebensdauer des Teils unabhängig von dem Ganzen ist.
Komposition 	Eine stärkere Form der Aggregation, bei der die Lebensdauer des Teils von der des Ganzen abhängt; wenn das Ganze gelöscht wird, wird auch das Teil gelöscht.
Vererbung 	Ein Mechanismus, durch den eine Klasse Eigenschaften und Verhalten (Methoden) von einer anderen Klasse erben kann, was die Wiederverwendbarkeit fördert.
Multiplizität 1 | 0..1 | 1..*	Gibt an, wie viele Objekte einer Klasse mit einem Objekt einer anderen Klasse in Beziehung stehen können (z. B. 1:1, 1:n, n:m).
Notiz	Eine Anmerkung oder ein Kommentar innerhalb eines UML-Diagramms, die zusätzliche Informationen oder Erklärungen zu Elementen oder Beziehungen bereitstellt.

2.1. VERERBUNG ZUSÄTZLICHE NOTIZEN

Begriff	Bedeutung
Disjoint	Objekt ist Instanz von genau einer Unterklasse.
Overlapping	Objekt kann Instanz von mehreren überlappenden Unterklassen sein; Objekt kann im Laufe seines Lebens die Unterklassen-Zugehörigkeit wechseln.
Complete	Alle Subklassen sind definiert
Incomplete	Zusätzliche Subklassen sind erlaubt

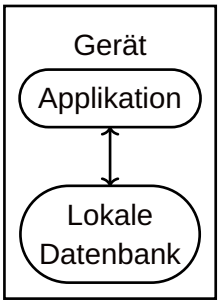
2.2. KRÄHENFUSSNOTATION



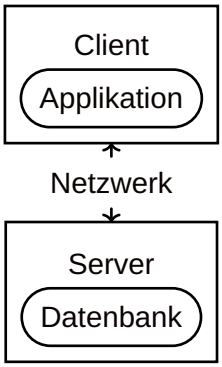
3. DATENBANKMODELLE

Begriff	Bedeutung
Hierarchisch	Ein Datenbankmodell, das Daten in einer baumartigen Struktur anordnet, bei der jede Entität einen Elternknoten hat und mehrere Kindknoten haben kann.
Netzwerk	Ein Datenbankmodell, das eine flexiblere Struktur als das hierarchische Modell bietet, indem es mehrere Pfade zwischen Entitäten erlaubt.
Objektorientiert	Ein Datenbankmodell, das objektorientierte Programmierprinzipien anwendet, um Daten und deren Verhalten in Form von Objekten zu speichern und zu verwalten.
Objektrelational	Kombiniert objektorientierte und relationale Prinzipien, um komplexe Datenstrukturen zu unterstützen und erweitert die relationale Modellierung um Objekte.
Relational	Ein weit verbreitetes Datenbankmodell, das Daten in Tabellen (Relationen) speichert und die Beziehungen zwischen diesen Tabellen durch Schlüssel verwaltet.

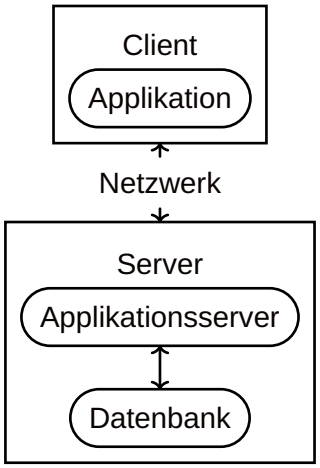
1-Tier



2-Tier



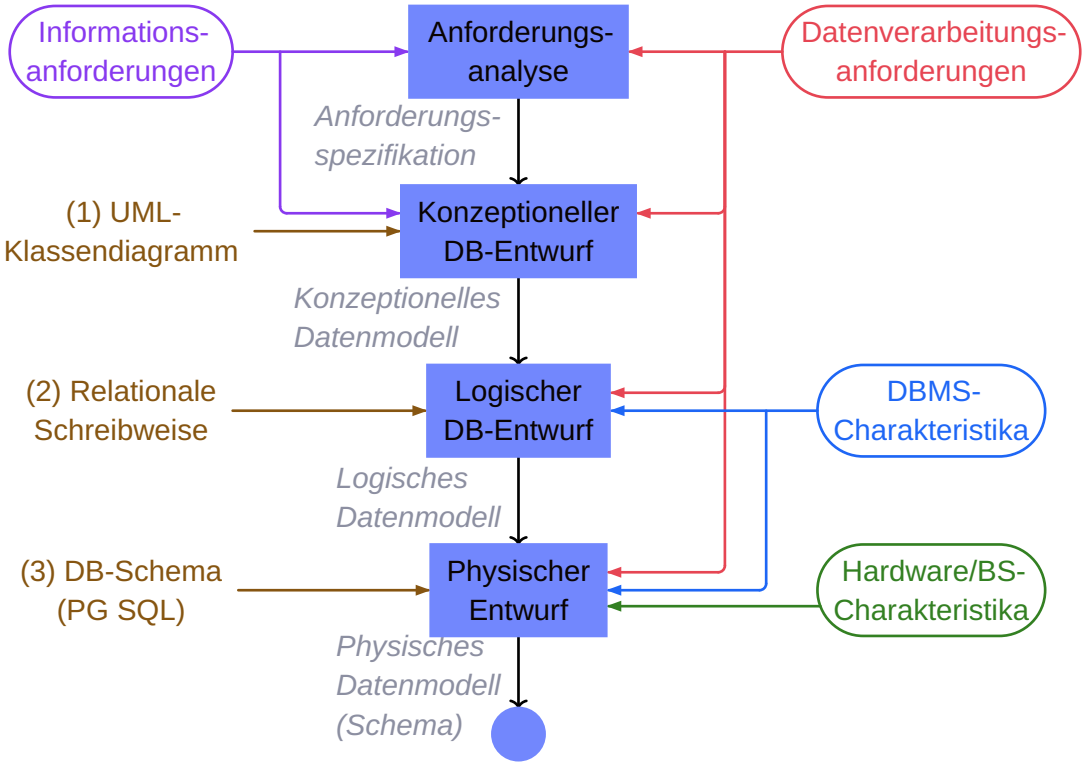
3-Tier



4. ANSI-MODELL

Begriff	Bedeutung
Logische Ebene	Logische Struktur der Daten, Definition durch logisches Schema «Trägermodell» (Zugriff auf die Daten durch DBMS von Speichermedium)
Interne Ebene	Speicherstrukturen, Definition durch internes Schema (Beziehungen zwischen den Daten, Tabellen etc.)
Externe Ebene	Sicht einer Benutzerklasse auf Teilmenge der DB, Definition durch externes Schema (Daten, auf die der Benutzer zugreifen kann)
Mapping	Zwischen den Ebenen ist eine mehr oder weniger komplexe Abbildung notwendig

5. DATENBANK-ENTWURFSPROZESS



5.1. KONZEPTIONELLES MODELL

In dieser Phase wird eine abstrakte Sicht auf die Datenbank erstellt, die die Anforderungen der Benutzer erfasst. Es werden Entitäten, Attribute und deren Beziehungen definiert, ohne sich um technische Details zu kümmern.

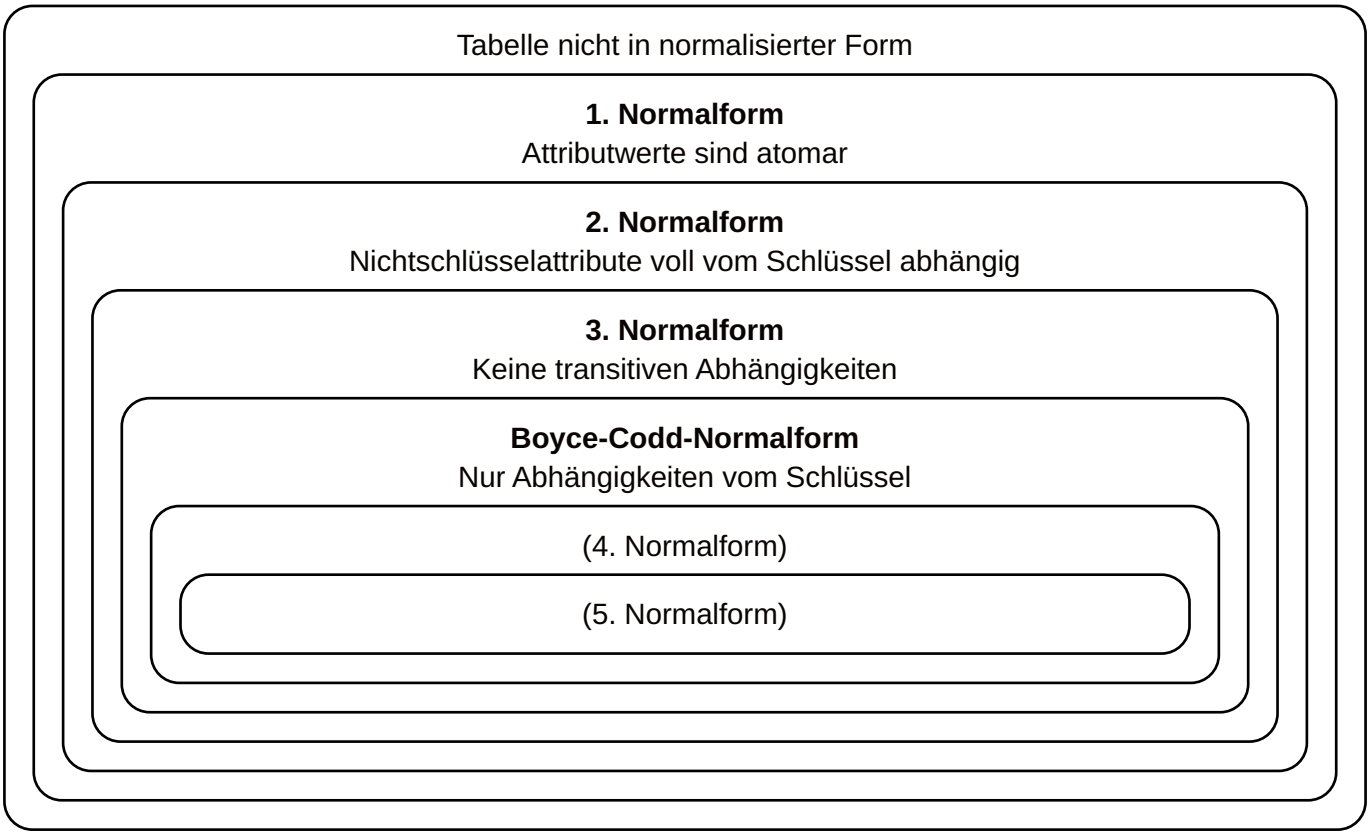
5.2. LOGISCHES MODELL

Hier wird das konzeptionelle Modell in eine spezifische Datenbankstruktur übersetzt, ohne sich um physische Aspekte zu kümmern. Es wird festgelegt, wie Daten organisiert sind (z. B. in Tabellen) und welche Regeln für Integrität bestehen.

5.3. PHYSISCHES MODELL

In dieser Phase werden die spezifischen Implementierungsdetails festgelegt, einschließlich der Speicherstruktur und der Indexierung. Das Modell wird optimiert, um Leistung, Effizienz und Datenzugriff zu verbessern.

6. NORMALFORMEN



Begriff	*Bedeutung*
Funktionale Abhängigkeit	Eine Beziehung zwischen zwei Attributen, bei der ein Attribut (B) von einem anderen Attribut (A) abhängt; zu jedem Wert von A existiert genau ein Wert von B, dargestellt als $A \rightarrow B$.
Atomar	Ein Attribut ist atomar, wenn es nicht weiter in Unterattribute zerlegt werden kann; es enthält also nur einen einzelnen Wert.
Voll funktional abhängig	Ein Attribut (B) ist voll funktional abhängig von einem anderen Attribut (A), wenn es von A abhängt und sich nicht durch andere Attribute ableiten lässt.
Teilweise funktional abhängig	Ein Attribut (B) ist teilweise funktional abhängig von einem anderen Attribut (A), wenn B von A abhängt, aber auch von einem Teil eines zusammengesetzten Schlüssels.
Transitiv abhängig	Ein Attribut (C) ist transitiv abhängig von einem anderen Attribut (A), wenn A auf B und B auf C verweist, wodurch eine indirekte Abhängigkeit zwischen A und C entsteht.

6.1. NF1

– Attributwerte sind atomar

6.2. NF2

– Nichtschlüsselatribute voll vom Schlüssel abhängig
– Besteht der PK nur aus einem einzigen Attribut (ist er also nicht zusammengesetzt), so bereits automatisch die 2. NF gegeben.
– Eine Tabelle ist NICHT in 2. NF, wenn sie einen zusammengesetzten Schlüssel hat und mind. ein Nichtschlüsselattribut von nur einem Teil des Schlüssels funktional abhängig ist.

6.3. NF3
– Keine transitiven Abhängigkeiten

6.4. BCNF
– Keine Abhängigkeiten vom Schlüssel

7. RELATIONALE ALGEBRA

https://users.informatik.uni-halle.de/~brass/db23/slides/dd_relal.pdf

Begriff		*Bedeutung*	*Beispiel*	*SQL*
Projektion	π	Extrahieren ausgewählter Spalten einer Relation	$\pi_{R1,R4}(R)$	`SELECT R1,R4 FROM R;`
Selektion	σ	Filtern der Datensätze. σ steht für $=, >, <, \neq, \leq, \geq$	$\sigma_{R1>30}(R)$	`SELECT * FROM R WHERE R1 > 30;`
Umbenennung	ρ	Umbenennung der Attribute	$\rho_{a \leftarrow R}$	`SELECT * FROM R AS a;`
Kartesisches Produkt	$\times$	Kombinieren jeder Zeile einer Relation mit jeder Zeile einer anderen Relation `CROSS JOIN`	$(R \times S)$	`SELECT * FROM R,S;`
Verbund/Join	$\bowtie$	Voraussetung: Attribute r[a] und r[b] sind vereinigungsverträglich (typkompatibel)	$R \bowtie S$	
Theta-join	$\bowtie_\theta$	θ steht für $=, >, <, \neq, \leq, \geq$	$R[a\theta b]S$	`SELECT R.* FROM R JOIN S ON R.A :theta S.B;`
Equi-join	$\bowtie_=$	θ ist $=$	$R \underset{A=B}{\bowtie} S$ $= \rho_{A=B}(R \times S)$	`SELECT * FROM R JOIN S ON R.A = S.B;`
Natural join	$\bowtie$	Equi join ohne doppelte spalten `ON R.A = S.A` kann ausgelassen werden, implizit werden gleich heissende columns verglichen		`SELECT * FROM R NATURAL JOIN S ON R.A = S.B;`
Outer join	$⟗$ $⟕$ $⟖$	Verbindet Tupel auch ohne Übereinstimmung		
Semi-join	$⋉$ $⋊$	«left outer join ohne join»		

8. (POSTGRE)SQL

8.1. GLOSSAR

Begriff	*Bedeutung*
Operatorbaum	Eine hierarchische Struktur, die die Beziehung zwischen Operatoren und Operanden in einer Abfrage darstellt; wird in der Analyse von SQL-Ausdrücken verwendet.

Begriff	*Bedeutung*
Semantische Integrität	Die Gewährleistung, dass die Daten innerhalb der Datenbank nicht nur syntaktisch korrekt, sondern auch inhaltlich sinnvoll sind, wodurch die Gültigkeit der Daten sichergestellt wird.
Relation	In PostgreSQL bezeichnet eine Relation eine Tabelle, die aus Zeilen und Spalten besteht, wobei jede Zeile einem Datensatz und jede Spalte einem Attribut entspricht.
Surrogate Key	Ein künstlich generierter Schlüssel, der zur eindeutigen Identifizierung von Datensätzen in einer Tabelle verwendet wird und keine natürliche Bedeutung hat; oft als Sequenz implementiert.

8.2. DDL (DATA DEFINITION LANGUAGE)

Postgres Dokumentation

Definiert Befehle für das Kreieren, Löschen und Modifizieren von Tabellendefinitionen, von externen Sichten (Views), von Constraints und von einigen anderen Datenbankobjekten (Index, Cluster) für die Optimierung der Abbildung von der logischen auf die interne Ebene.

8.2.1. Wichtigste Befehle

Zusätzlich kann `IF (NOT) EXISTS` hinzugefügt werden oder der `CREATE` mit `CREATE OR REPLACE` ausgetauscht werden. Beim `DROP` Befehl kann `CASCADE` hinzugefügt werden.

```
CREATE SCHEMA s;     DROP SCHEMA s;
CREATE DOMAIN d;     DROP DOMAIN d;
CREATE TABLE t;      DROP TABLE t;      ALTER TABLE t;  TRUNCATE TABLE
t;
CREATE VIEW v;       DROP VIEW v;
CREATE INDEX i;      DROP INDEX i;
CREATE DATABASE db; DROP DATABASE db;
```

8.2.2. Beispiel CREATE TABLE

```
CREATE TABLE schema_name.table_name (
  counter SERIAL NOT NULL,
  important INT NOT NULL,
  field INT NOT NULL UNIQUE,
  very_long_name VARCHAR(666) DEFAULT "Based PostgreSQL user",
  age INT CHECK (age > 18),
  reference INT NOT NULL,
  someday DATE DEFAULT SYSDATE,
  another_ref INT FOREIGN KEY REFERENCES other_table.id ON UPDATE
RESTRICT,
  PRIMARY KEY (counter, important),
  FOREIGN KEY (reference) REFERENCES third_table.id ON DELETE CASCADE,
  CHECK (field BETWEEN 69 AND 420)
)
```

8.2.3. Datentypen

Postgres Dokumentation

Sinnvolle Konversionen und Rundungen werden implizit durchgeführt.

Begriff	*Bedeutung*
INTEGER/INT	Integer (4 bytes)
BIGINT	Large integer (8 bytes)
SMALLINT	Small integer (2 bytes)
REAL	Single precision floating-point number (4 bytes)
NUMERIC(precision,scale)	Exact numeric of selectable precision
DOUBLE PRECISION	Double precision floating-point number (8 bytes)
SERIAL	Auto-incrementing integer (4 bytes)
BIGSERIAL	Auto-incrementing large integer (8 bytes)
SMALLSERIAL	Auto-incrementing small integer (2 bytes)
CHARACTER/CHAR(size)	Fixed-length, blank-padded string
VARCHAR(size)	Variable-length, non-blank-padded string
TEXT	Variable-length character string
BOOLEAN	Logical Boolean (true/false)
DATE	Calendar date (year, month, day)
TIME	Time of day (no time zone)
TIMESTAMP	Date and time (no time zone)
TIMESTAMP WITH TIME ZONE	Date and time with time zone
INTERVAL	Time interval
JSON	JSON data
UUID	Universally unique identifier
ARRAY OF base_type	Array of values

8.2.3.1. Type casting

Explizit

```
CAST(5 AS float8) = 5::float8
SELECT 'ABCDEFG'::NUMERIC;                    -- error
SELECT SAFE_CAST('ABCDEFG' AS NUMERIC);       -- NULL
```

Implizit

```
SELECT 5 + 3.2;                               -- cast to 5.0 (numeric)
SELECT 'Number ' || 42;                       -- cast to '42'
SELECT true AND 1;                            -- treated as true
SELECT CURRENT_TIMESTAMP + INTERVAL '1 day'; -- cast to DATE
SELECT '100'::text + 1;                       -- cast to 100
```

8.2.4. Contraints

Postgres Dokumentation

Begriff	*Bedeutung*
PRIMARY KEY	Attribut ist Primärschlüssel und damit «UNIQUE» und «NOT NULL»
NOT NULL	Attributwerte müssen immer einen definierter Wert haben (default ist NULL)
UNIQUE	Attributwerte aller Tupels der Tabelle müssen eindeutig sein. UNIQUE in Kombination mit NOT NULL definiert einen Sekundärschlüssel. UNIQUE in Kombination mit NULL bedeutet, dass alle Attributwerte ungleich NULL eindeutig sein müssen.
CHECK	erlaubt die Definition von weiteren Einschränkungen (siehe Beispiele)
DEFAULT	setzt einen Defaultwert, dieser gilt wenn beim Einfügen eines Tupels mit INSERT kein Attributwert angegeben wird.
REFERENCES	Attribut ist Fremdschlüssel

8.2.5. ALTER TABLE

Postgres Dokumentation

Sollte bei foreign Keys bevorzugt werden, da somit rekursive Referenzen einfacher umgesetzt werden können.

```
ALTER TABLE table_name
  ADD CONSTRAINT constraint_name
  CHECK (counter < 1337);

ALTER TABLE other_table
  ADD CONSTRAINT fk_that
  FOREIGN KEY (that) REFERENCES table_name (counter);
```

8.2.6. Referentielle Integrität

Der Fremdschlüssel in einer abhängigen Tabelle muss als Wert entweder einen aktuellen Wert des Schlüssels der referenzierten Tabelle (1:n) oder NULL (1c:n) aufweisen. Diese referentielle Integritätsbedingung kann bei Einfüge-, Lösch- und Änderungsoperationen verletzt werden und sollte daher vom DBMS überprüft werden.

Trigger:

```
ON UPDATE
ON DELETE
```

Aktion:

```
CASCADE
RESTRICT
SET DEFAULT
SET NULL
```

Siehe «Beispiel CREATE TABLE» (Seite 9) .

8.2.7. Index

Blogpost Stefan Keller

Use the index, luke

Ein Index ist eine Hilfsdatenstruktur, die zu einem gegebenen Attributwert die Adressen der Tupel mit diesem Attributwert liefert.

Begriff	*Bedeutung*
B-Baum	Ein selbstbalancierender Baum, der für schnelle Datenzugriffe in Datenbanken verwendet wird; er ermöglicht effizientes Einfügen, Löschen und Suchen von Datensätzen.
B+-Baum	Es befinden sich alle Daten nur in den Blattknoten. Schneller als B-Baum.
Bitmap-Index	Ein Index-Typ, der eine Bitmap für jeden Wert eines Attributs verwendet, um große Datenmengen effizient abzufragen und besonders für Spalten mit wenigen eindeutigen Werten geeignet ist.
Cluster	Eine Methode, die Daten in einer Tabelle physisch zusammenzuhalten basierend auf einem bestimmten Index, wodurch der Zugriff auf mehrere zusammengehörende Datensätze verbessert wird.
Füllgrad	Der Prozentsatz des Speicherplatzes eines Indexes, der tatsächlich mit Daten gefüllt ist; beeinflusst die Effizienz bei der Index-Suche und Datenspeicherung.
Hash	Ein Index, der eine Hash-Funktion verwendet, um den Zugriff auf Datensätze zu beschleunigen; besonders nützlich für Gleichheitsabfragen bei sehr großen Datenmengen.
Heap	Die Standard-Speicherstruktur in einer relationalen Datenbank, bei der Datensätze in beliebiger Reihenfolge gespeichert werden, ohne einen spezifischen Index.
Indizes	Strukturen, die die Suche nach Datensätzen in einer Datenbank beschleunigen, indem sie eine schnelle Zugriffsart auf Daten bereitstellen, ohne alle Datensätze scannen zu müssen.
ISAM	Indexed Sequential Access Method; ein älteres Indexierungssystem, das eine Kombination aus sequenzieller und indizierter Speicherung verwendet, um den Datenzugriff zu optimieren.
Physische Speicherstruktur	Die Art und Weise, wie Daten in der Datenbank physisch auf dem Speicher abgelegt sind, einschließlich der Verwendung von Indizes für eine schnellere Datenabfrage.
Überlaufseite	Eine spezielle Seite, die verwendet wird, um zusätzliche Daten zu speichern, wenn die ursprüngliche Indexseite voll ist; wird häufig bei B-Bäumen verwendet.
Zusammengesetzter Index	Ein Index, der auf zwei oder mehr Attributen basiert, um schnellere Suchoperationen für Kombinationen dieser Attribute zu ermöglichen, anstatt nur auf ein einzelnes Attribut zuzugreifen.
Partieller Index	Ein Index, der nur auf einen Teil der Tabelle angewendet wird, basierend auf einer definierten Bedingung; ermöglicht effiziente Abfragen für häufig verwendete Teilmengen der Daten.
Funktionaler Index	Basiert auf den Ergebnissen einer Funktion, die auf den Werten einer oder mehrerer Spalten angewendet wird.

Zwei Datenstrukturen:
– Data Pages (Heaps)
– Suchbaum

Zugriff Baum:

– Durchwandern des Baumes

– Verfolgen der Blattknoten-Kette

– Tabellenzugriff (falls nötig)

– Primär-Index vs Sekundär-Index

– Sekundärer Index vs Integrierter (Clustered) Index

```
CREATE INDEX mytable_col_idx ON mytable (col);

SELECT id,nr,txt FROM test;
CREATE INDEX magic_idx ON test (nr,id) INCLUDE txt;

CREATE EXTENSION btree_gist;
CREATE INDEX mytable_col_idx2 ON mytable (col) USING gist (col);

DROP INDEX mytable_col_idx;

CREATE INDEX mytable_col_idx ON mytable (UPPER(col));
```

8.2.7.1. B-Tree

8.2.8. Schema

Postgres Dokumentation

Ein Schema ist ein Menge von DB-Objekten, welche zu einer logischen Datenbank gehören. Ein Schema kann folgende Objekte beinhalten: Tabellen, Sichten, Zusicherungen (Assertions), Berechtigungen etc.

8.2.9. Permissions

Postgres Dokumentation

8.3. DML (DATA MANIPULATION LANGUAGE)

Postgres Dokumentation

8.3.1. Ausschnitt des Syntax

```
<select> := [ 'WITH' [ 'RECURSIVE' ] <with_query> [, ...] ]
'SELECT' [ 'ALL' | 'DISTINCT' [ 'ON' ( <expression> [, ...] ) ] ]
    [ { * | <expression> [ [ 'AS' ] <output_name> ] } [, ...] ]
    [ 'FROM' <from_item> [, ...] ]
    [ 'WHERE' <condition> ]
    [ 'GROUP BY' [ 'ALL' | 'DISTINCT' ] <grouping_element> [, ...] ]
    [ 'HAVING' <condition> ]
    [ 'WINDOW' <window_name> 'AS' ( <window_definition> ) [, ...] ]
    [ { 'UNION' | 'INTERSECT' | 'EXCEPT' } [ 'ALL' | 'DISTINCT' ] <select> ]
    [ 'ORDER BY' <expression> [ 'ASC' | 'DESC' | 'USING' <operator> ]
[ 'NULLS' { 'FIRST' | 'LAST' } ] [, ...] ]
    [ 'LIMIT' { <count> | 'ALL' } ]
    [ 'OFFSET' <start> [ 'ROW' | 'ROWS' ] ]

<from_item> := <table_name> [ * ] [ [ 'AS' ] <alias> [ ( <column_alias> [, ...] ) ] ]
    [ 'LATERAL' ] ( <select> ) [ [ 'AS' ] <alias> [ ( <column_alias> [, ...] ) ] ]
    <with_query_name> [ [ 'AS' ] <alias> [ ( <column_alias> [, ...] ) ] ]
    <from_item> <join_type> <from_item> { 'ON' <join_condition> |
'USING' ( <join_column> [, ...] ) [ 'AS' <join_using_alias> ] }
    <from_item> 'NATURAL' <join_type> <from_item>
```

```
    <from_item> 'CROSS JOIN' <from_item>

<with_query> := <with_query_name> [ ( <column_name> [, ...] ) ] 'AS' ( <select> |
<values> | <insert> | <update> | <delete> | <merge> )
        [ 'USING' <cycle_path_col_name> ]
```

8.3.2. INSERT

Postgres Dokumentation

```
INSERT INTO table_name (important, field, reference) VALUES (3, 99,
7);

INSERT INTO other_table VALUES (20, "Goodbye world.") RETURNING
counter;
```

8.3.2.1. INSERT ... SELECT

```
INSERT INTO combined_table
  SELECT t.age, t.very_long_name FROM table_name AS t
  INNER JOIN other_table AS o ON t.reference = o.id;
```

8.3.3. UPDATE

Postgres Dokumentation

Mit dem Update-Befehl können bestehende Tupel in der Datenbank modifiziert werden.

```
UPDATE table_name
  SET field = field * 2
  WHERE field IN (71,73,74);
```

8.3.4. DELETE

Postgres Dokumentation

```
DELETE FROM table_name
  WHERE field > 300;
```

8.3.5. SELECT

Postgres Dokumentation

```
SELECT field FROM table_name;

SELECT important, CONCAT(very_long_name, age) AS personal_info
  FROM table_name
  WHERE counter > 21
  GROUP BY personal_info
  ORDER BY age, counter DESC
  LIMIT 77;
```

8.3.5.1. Prädikate (WHERE)

Postgres Dokumentation

```
BETWEEN ... AND ...
IN (..., ...)
LIKE '___%'          -- 3 chars or more
```

```
LIKE '%asd'          -- ending with "asd"
AND
OR
IS (NOT) NULL
```

8.3.5.2. Aggregatfunktionen

Postgres Dokumentation

```
MAX()
MIN()
AVG()
SUM()
COUNT()
```

8.3.5.3. Gruppierung (GROUP BY and HAVING)

Postgres Dokumentation

GROUP BY teilt die Resultattabelle in Gruppen auf, die in der GROUP BY - Spalte gleiche Werte aufweisen. NULL-Werte einer GROUP-BY Spalte werden als separate Gruppe behandelt.

Die ***HAVING*** Klausel kann nur nach einer GROUP-BY Klausel stehen. Sie erlaubt die Auswahl von Zeilen, die durch die Anwendung der GROUP BY Bedingung entstehen (analog der WHERE-Klausel). Die Bedingung der HAVING-Klausel muss mit einer Funktion beginnen, welche in der SELECT-Klausel vorkommen muss.

```
SELECT very_long_name, age, COUNT(*)
  FROM table_name
  GROUP BY age, very_long_name
  HAVING COUNT(*) > 2;
```

8.3.5.4. Join

Postgres Dokumentation , Visuelle Beispiele Atlassian , Visuelle Beispiele CodeCademy

Die Join-Operation verbindet Tabellen über Spalten mit dem gleichen Datentyp. Damit können Beziehungen zwischen den Tabellen (realisiert über Fremdschlüssel) aufgelöst werden.

8.3.5.4.1. CROSS JOIN

Gibt das kartesische Produkt der beiden Tabellen zurück, d.h. jede Zeile aus der ersten Tabelle wird mit jeder Zeile aus der zweiten Tabelle kombiniert.

```
SELECT a.*, b.*
  FROM a
  CROSS JOIN b;
```

8.3.5.4.2. INNER JOIN

Gibt nur die Zeilen zurück, die in beiden Tabellen übereinstimmen.

```
SELECT a.*, b.*
  FROM a
  INNER JOIN b ON a.id = b.id;
```

8.3.5.4.3. LEFT JOIN

Gibt alle Zeilen aus der linken Tabelle zurück und die übereinstimmenden Zeilen aus der rechten Tabelle. Wenn es keine Übereinstimmung gibt, werden NULL-Werte für die rechte Tabelle zurückgegeben.

8.3.5.4.4. RIGHT JOIN

Wie left join, nur umgekehrt.

8.3.5.4.5. FULL OUTER JOIN

Gibt alle Zeilen zurück, die in einer der beiden Tabellen vorhanden sind. Wenn es keine Übereinstimmung gibt, werden NULL-Werte für die Tabelle zurückgegeben, in der keine Übereinstimmung gefunden wurde.

8.3.5.4.6. JOIN LATERAL

Ein spezieller Typ von Join, der es ermöglicht, eine Unterabfrage zu verwenden, die auf Zeilen der äußeren Abfrage basiert. Dies ist nützlich für komplexe Abfragen, bei denen die Ergebnisse einer Unterabfrage von den Ergebnissen der äußeren Abfrage abhängen.

```
SELECT a.*, b.*
  FROM table_a AS a,
    JOIN LATERAL (SELECT * FROM table_b WHERE table_b.id = a.id) AS b;
```

8.3.5.5. Mengenoperationen

Postgres Dokumentation

Jede dieser Operationen kann mit `ALL` postfixed werden, um die Duplikate beizubehalten.

8.3.5.5.1. UNION

Kombiniert die Ergebnisse zweier oder mehrerer SELECT-Abfragen und gibt alle Zeilen zurück, ausschließlich Duplikate.

```
SELECT r1, r2
FROM A
UNION ALL
SELECT r1, r2
FROM B;
```

8.3.5.5.2. INTERSECT

Gibt die gemeinsamen Zeilen aus zwei SELECT-Abfragen zurück. Das Ergebnis enthält nur die Zeilen, die in beiden Abfragen vorhanden sind.

8.3.5.5.3. EXCEPT (MINUS)

Gibt die Zeilen aus der ersten SELECT-Abfrage zurück, die nicht in der zweiten SELECT-Abfrage vorhanden sind.

8.3.5.6. Unterabfragen

Postgres Dokumentation

Eine Unterabfrage darf nur einen Spaltennamen oder einen Ausdruck und keine ORDER BY - Klausel enthalten.

Verschachteltes Beispiel:

```
SELECT name
  FROM mitarbeiter
  WHERE id IN
  (SELECT mitarbeiterNr
    FROM projektZuteilung
    WHERE projNr IN
    (SELECT projNr
```

```
      FROM projekt INNER JOIN mitarbeiter
      ON projekt.projLeiter = mitarbeiter.id
      WHERE name = 'Kropotkin, Peter'
    )
  );
```

8.3.5.6.1. =

Vergleicht den Wert einer Spalte mit dem Ergebnis einer Unterabfrage. Unterabfrage darf nur einen Wert zurückliefern.

```
SELECT *
  FROM mitarbeiter
  WHERE gehalt = (SELECT MAX(gehalt) FROM mitarbeiter);
```

8.3.5.6.2. IN

Überprüft, ob der Wert einer Spalte in der Menge der Ergebnisse einer Unterabfrage enthalten ist.

```
SELECT *
  FROM bestellungen
  WHERE kundenID IN (SELECT id FROM kunden WHERE land = 'CH');
```

8.3.5.6.3. EXISTS

Überprüft, ob eine Unterabfrage mindestens eine Zeile zurückgibt.

```
SELECT name
  FROM mitarbeiter
  WHERE EXISTS
  (SELECT * FROM projektZuteilung WHERE mitarbeiterNr =
mitarbeiter.id);
```

8.3.5.6.4. ANY

Überprüft, ob der Wert einer Spalte irgendeinen Wert in der Menge der Ergebnisse einer Unterabfrage erfüllt.

```
SELECT *
  FROM mitarbeiter
  WHERE gehalt > ANY (SELECT gehalt FROM manager);
```

8.3.5.6.5. ALL

Überprüft, ob der Wert einer Spalte alle Werte in der Menge der Ergebnisse einer Unterabfrage erfüllt.

8.3.5.7. Window functions

Postgres Dokumentation

Window Functions sind spezielle SQL-Funktionen, die Berechnungen über eine Menge von Zeilen durchführen, die mit der aktuellen Zeile in Beziehung stehen.

```
SELECT
  mitarbeiter,
  gehalt,
  RANK() OVER (ORDER BY gehalt DESC) as gehaltsrang
FROM
```

```
  mitarbeitertabelle;

SELECT
  persnr,
  name,
  LAG(salaer, 1) OVER (
    PARTITION BY abtnr
  ORDER BY
    salaer DESC
  ) - salaer AS differenz
FROM
  angestellter;
```

8.3.5.7.1. Funktionen

Postgres Dokumentation

Begriff	*Bedeutung*
RANK()	Vergibt Rangpositionen mit Berücksichtigung von Gleichständen. Bei mehreren gleichen Werten erhalten diese den gleichen Rang, und die nächste Position wird übersprungen
DENSE_RANK()	Wie RANK nur ohne Lücken
PERCENT_RANK()	Berechnet den prozentualen Rang eines Datensatzes innerhalb eines Fensters
ROW_NUMBER()	Weist jeder Zeile eine eindeutige fortlaufende Nummer zu. Auch bei gleichen Werten erhält jede Zeile eine unterschiedliche Nummer
LAG(value, offset)	Greift auf den Wert einer vorherigen Zeile im Fenster zu
LEAD(value, offset)	Greift auf den Wert einer nachfolgenden Zeile im Fenster zu
FIRST_VALUE(value)	Liefert den ersten Wert in der definierten Fenstermenge. Nützlich für Vergleiche mit dem Anfangswert einer Partition
LAST_VALUE(value)	Liefert den letzten Wert in der definierten Fenstermenge
NTH_VALUE(value, n)	Gibt den Wert des n-ten Datensatzes innerhalb eines Fensteres zurück
NTILE(n)	Teilt die Datensätze in n gleich grosse Gruppen auf

8.3.5.7.2. OVER Klausel

Begriff	*Bedeutung*
ORDER BY	Sortiert die Zeilen innerhalb des Fensters nach einem oder mehreren Spalten. Bestimmt die Reihenfolge, in der Berechnungen für Window Functions durchgeführt werden
PARTITION BY	Teilt das Resultset in Partitionen, auf die Window Functions separat angewendet werden. Ermöglicht Berechnungen innerhalb definierter Gruppen, ohne die Gesamtergebnismenge zu ändern

8.4. VIEWS

Eine View ist eine virtuelle Tabelle, welche auf eine oder mehrere Tabellen oder Views abgebildet wird. Die Abbildung wird mit einer Select-Anweisung definiert. Die Daten der View werden erst zur Ausführungszeit aus den darunter liegenden Tabellendaten hergeleitet.

Die Views erlauben es, dass verschiedene Benutzer die Daten unterschiedlich strukturiert sehen. Mit Views kann die Formulierung von Abfragen, die sich über mehrere Tabellen erstrecken, verein- facht werden. Views erlauben einen wirksamen Zugriffsschutz, da es möglich ist, Spalten der da- runterliegenden Tabellen auszublenden.

```
CREATE VIEW mitarbeiter_public (id, name, tel) AS
  SELECT id, name, tel
  FROM mitarbeiter;
```

8.4.1. Common table expressions

Hilfs-Query in einer WITH-Klausel (Temporäre Tabellen während des Statements). Query-Name immer im FROM.

```
WITH queryName AS ( SELECT * FROM myTable )
SELECT * FROM queryName;

WITH tmptable(name, bezeichnung, zeitanteil) AS (
  SELECT name, bezeichnung, zeitanteil
  FROM angestellter a
  JOIN projektzuteilung pz ON pz.persnr=a.persnr
  JOIN projekt p ON p.projnr=pz.projnr
)
SELECT name AS "Mitarb.", bezeichnung AS "Projekt", zeitanteil AS
"Zeit" FROM tmptable;
```

8.4.1.1. Recursive

Ein rekursives CTE referenziert sich selbst, um Teilmengen der Daten zurückzugeben, bis alle Ergebnisse erhalten sind.

```
WITH RECURSIVE untergebene(persnr, name, chef) AS (
  SELECT A.persnr, A.name, A.chef FROM angestellter A
  WHERE A.chef = 1010 UNION ALL -- recursive term
    SELECT A.persnr, A.name, A.chef FROM angestellter A
  INNER JOIN untergebene B ON B.persnr = A.chef
)
SELECT * FROM untergebene ORDER BY chef, persnr;
```

8.5. FUNKTIONEN

```
CREATE PROCEDURE
EXEC
```

8.6. DCL (DATA CONTROL LANGUAGE)

8.6.1. Benutzerverwaltung

Postgres Dokumentation Postgres Dokumentation

ROLE: Oberbegriff für Benutzer (***USER***) oder Gruppe (***GROUP***).

```
CREATE ROLE user_name
WITH LOGIN PASSWORD 'password';

-- set current user
```

```
SET ROLE user_name;

DROP ROLE user_name;
```

8.6.2. Berechtigungen

Postgres Dokumentation

8.6.2.1. System

```
GRANT INSERT ON TABLE table_name TO user_name;
REVOKE INSERT ON TABLE table_name TO user_name;
```

8.6.2.2. Objekt

```
ALTER ROLE user_name CREATEROLE, CREATEDB, INHERIT;
```

8.6.3. Gruppen

```
CREATE ROLE manager;
GRANT SELECT ON table_name TO manager;
GRANT manager TO user_name;
```

8.6.4. Read-only user

```
-- creating
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
CREATE ROLE readonlyuser WITH LOGIN ENCRYPTED PASSWORD 'readonlyuser'
NOINHERIT;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO readonlyuser;
-- assign permissions to read all newly tables created in the future
(by others):
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO
readonlyuser;
-- deleting
REVOKE SELECT ON ALL TABLES IN SCHEMA public FROM readonlyuser;
ALTER DEFAULT PRIVILEGES IN SCHEMA public
  REVOKE SELECT ON TABLES FROM readonlyuser;
DROP USER readonlyuser;
```

8.6.5. RBAC (Role-Based Access Control)

Ist das, was oben erklärt wird.

8.6.6. RLS (Row-Level Security)

– Check-Constraint auf Tabelle pro Rolle (sozusagen ein zusätzliches WHERE)
– (Tipp: Aufpassen bei TRUNCATE, REFERENCES und VIEWS)

```
CREATE TABLE exams (
  id SERIAL,
  student_name TEXT,
  grade DECIMAL(2,1),
  added TIMESTAMP DEFAULT current_timestamp,
  teacher_pguser VARCHAR(60) DEFAULT current_user
);
```

```
CREATE POLICY policy_teachers_see_own_exams ON exams
FOR ALL
TO PUBLIC
USING (teacher_pguser = current_user);

ALTER TABLE exams
ENABLE ROW LEVEL SECURITY;
```

8.7. PREPARED STATEMENTS

Postgres Dokumentation

8.8. TRANSAKTIONEN

Begriff	*Bedeutung*
Fault Tolerance	Bei Server-Crash kann Operation wiederholt werden oder wird ganz gecancelt (nicht nur Hälfte durchgeführt)
Concurrency	Isolation der Transaktionen, Parallelität wird ermöglicht.
MVCC (Multi-Version Concurrency Control)	Mehrere Transaktionen gleichzeitig verwalten, ohne dass es zu Konflikten kommt
Locking	Ein Mechanismus zur Steuerung des Zugriffs auf Daten in einer Datenbank, der sicherstellt, dass nur eine Transaktion gleichzeitig auf bestimmte Daten zugreifen oder diese ändern kann, um Inkonsistenzen zu vermeiden.
2-Phasen-Sperrprotokoll	Ein Protokoll, das Transaktionen zwingt, zuerst alle erforderlichen Sperren zu erhalten (Phase 1) und danach keine neuen Sperren mehr zu setzen (Phase 2). Dies garantiert die Serialisierbarkeit der Transaktionen.
Lost-Update	Ein Problem, das auftritt, wenn zwei Transaktionen denselben Datensatz lesen und ändern. Die Änderungen der ersten Transaktion gehen verloren, da die zweite Transaktion die Änderungen überschreibt, ohne sie zu berücksichtigen.
Non-Repeatable Read	Eine Situation, in der eine Transaktion einen Datensatz liest, und während dieser Transaktion ändert eine andere Transaktion denselben Datensatz. Ein späterer Lesevorgang in der ersten Transaktion könnte einen anderen Wert zurückgeben.
Optimistisches Lockverfahren	Ein Ansatz, bei dem Transaktionen ohne anfängliche Sperren operieren und am Ende überprüfen, ob ihre Änderungen gültig sind. Wenn Konflikte auftreten, werden Änderungen zurückgesetzt.
Pessimistisches Lockverfahren	Ein Ansatz, bei dem Transaktionen sofort Sperren anfordern, um sicherzustellen, dass andere Transaktionen nicht gleichzeitig auf dieselben Daten zugreifen oder diese ändern.
Semantische Integrität	Die Sicherstellung, dass die Daten in der Datenbank nicht nur syntaktisch, sondern auch inhaltlich sinnvoll und korrekt sind, insbesondere nach Transaktionsoperationen.
SLOCK (Shared Lock)	Ein Sperrtyp, der mehreren Transaktionen das gleichzeitige Lesen von Daten erlaubt, jedoch das Schreiben während einer laufenden Transaktion verhindert.
XLOCK (Exclusive Lock)	Ein Sperrtyp, der einer Transaktion das exklusive Lesen und Schreiben eines Datensatzes ermöglicht und allen anderen Transaktionen verbietet, darauf zuzugreifen, bis die Sperre aufgehoben wird.

Begriff	Bedeutung
UNLOCK	Der Befehl oder die Aktion, um eine zuvor gesetzte Sperre aufzuheben, wodurch anderen Transaktionen der Zugriff auf die gesperrten Daten wieder ermöglicht wird.
Starvation	Eine Situation, in der eine Transaktion aufgrund von Sperren oder anderen Contention-Mechanismen niemals die Möglichkeit erhält, ihre Arbeit abzuschließen, da sie ständig blockiert wird.
A Atomicity	Vollständig oder gar nicht
C Consistency	Konsistenter Zustand bleibt erhalten
I Isolation	Transaktion soll von anderen isoliert sein
D Durability	Alle Änderungen sind persistent

```
BEGIN;

COMMIT;
BEGIN;

ROLLBACK;
```

8.8.1. Transaktionsisolation

https://pgdash.io/blog/postgres-transactions.html

```
READ UNCOMMITTED
READ COMMITTED
REPEATABLE READ
SERIALIZABLE
```

8.8.1.1. Fehlertypen

Begriff	Bedeutung
Dirty Read	Ein Dirty Read tritt auf, wenn eine Transaktion Daten liest, die von einer anderen, noch nicht abgeschlossenen Transaktion geändert wurden. Dies kann zu inkonsistenten Daten führen, da die ersten Änderungen möglicherweise später zurückgesetzt werden.
Fuzzy Read	Bei einem Fuzzy Read liest eine Transaktion Daten, die während ihrer Ausführung von anderen Transaktionen geändert werden. Dadurch kann die Transaktion unterschiedliche Werte lesen, wenn sie die gleiche Abfrage mehrmals ausführt.
Phantom Read	Ein Phantom Read tritt auf, wenn eine Transaktion eine Abfrage ausführt, die aufgrund von Änderungen in einer anderen Transaktion neue Datensätze zurückgibt. Dies geschieht häufig, wenn neue Datensätze eingefügt werden, nachdem die Abfrage ursprünglich ausgeführt wurde.
Serialization Anomaly	Diese Anomalie tritt auf, wenn die Ergebnisse von Transaktionen, die in einer bestimmten Reihenfolge ausgeführt werden, anders sind als die, die auftreten würden, wenn die Transaktionen seriell nacheinander in einer anderen Reihenfolge ausgeführt werden. Dies kann zu inkonsistenten Ergebnissen führen.

8.8.1.2. READ COMMITTED

Standart-Isolationsstufe in PostgreSQL.
Jede Abfrage sieht nur Daten, die vor Beginn der Abfrage committed wurden. Verhindert «Dirty Reads»

8.8.1.3. REPEATABLE READ

Verhindert «Non-Repeatable Reads». Verwendet Snapshot-Isolation. Höherer Speicherbedarf durch Snapshots.

8.8.1.4. SERIALIZABLE

Falls die gleiche Spalte in mehreren Transaktionen UPDATEd wird, werden die Transaktionen abgebrochen (ausser die erste).

8.8.2. Savepoints

Postgres Dokumentation

```
BEGIN;
...
SAVEPOINT savepoint1;
...
ROLLBACK TO SAVEPOINT savepoint1;
COMMIT;
```

8.8.3. Serialisierbarkeit

https://www.youtube.com/watch?v=01MDhIXiXIY

Begriff	***Bedeutung***
Seriell	Wenn alle Transaktionen in einem Schedule geordnet sind
Konfliktäquivalent	Wenn die Rehenfolge aller Paare von konfligierenden Aktionen in beiden Schedules gleich ist
Konfliktserialisierbar	Wenn ein Schedule konfliktäquivalent zu einem seriellen Schedule ist

8.9. BACKUP UND RECOVERY

Begriff	***Bedeutung***
WAL (Write-Ahead-Logging)	Ein Protokollierverfahren, das sicherstellt, dass alle Änderungen an der Datenbank zunächst in einem Log gespeichert werden, bevor sie in die Datenbank geschrieben werden. Dies ermöglicht eine Wiederherstellung im Falle eines Fehlers
Physischer Backup	Eine Sicherung, die eine Kopie der gesamten Datenbankdateien (einschließlich der Logdateien) erstellt. Diese Art von Backup ist nützlich für die vollständige Wiederherstellung einer Datenbank
Logischer Backup	Eine Sicherung, die die Struktur und den Inhalt der Datenbank auf einer höheren Ebene abbildet, z. B. durch Exportieren von Tabellen als SQL-Skripte. Diese Art von Backup ist nützlich für die Migration zwischen Datenbanken
Full Backup	Eine vollständige Sicherung, die alle Daten einer Datenbank zu einem bestimmten Zeitpunkt sichert. Dies ist die umfassendste Art von Backup und ermöglicht eine vollständige Wiederherstellung
Incremental Backup	Eine Sicherung, die nur die Änderungen sichert, die seit dem letzten Backup (vollständig oder inkrementell) vorgenommen wurden. Dies spart Speicherplatz und verkürzt die Backup-Zeit