

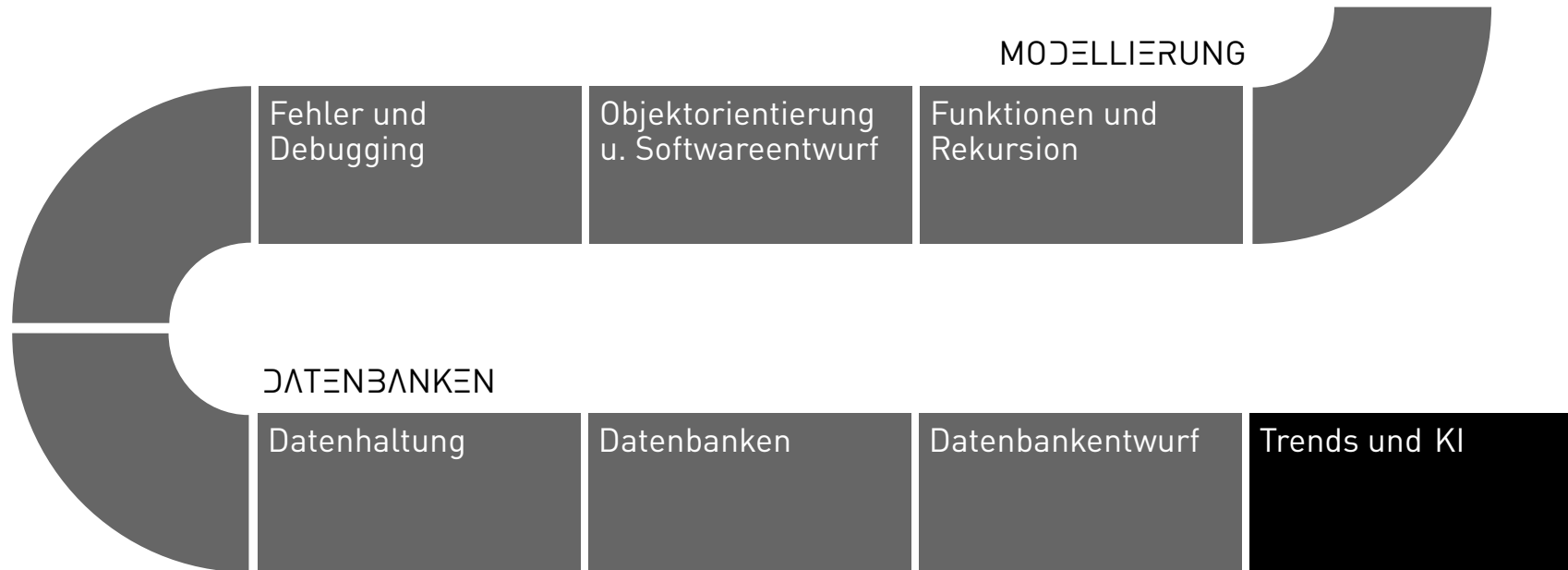
programmierung und datenbanken

Joern Ploennigs

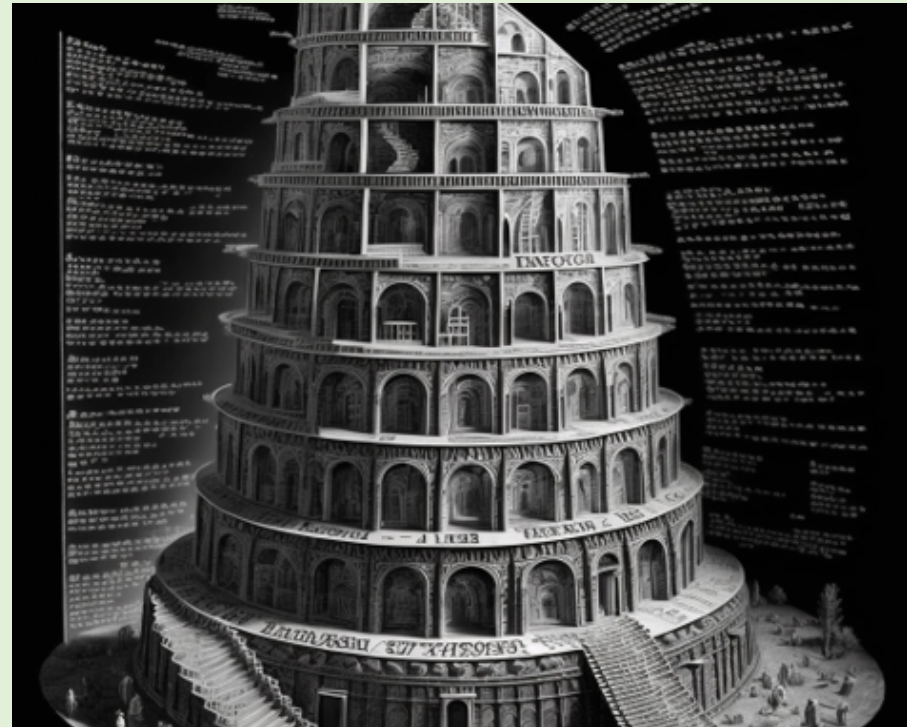
Tabellen Anlegen

MIDJOURNEY: THE CREATION OF SQL, REF. MICHELANGELO

ABLAUF



TABELLEN MIT SQL ANLEGEN



Midjourney: Construction plan of the tower of babel

LETZTER SCHRITT: IMPLEMENTIERUNG DES ER-DIAGRAMMES

Abbilden des ER-Modells auf SQL-Befehle zum Erzeugen der Tabellen

- Erstellen der Tabellen
- Für jedes Attribut jeder Relation einen sinnvollen Datentyp finden
- Festlegen ob Attribute Nullbar oder Nicht Null sind
- Festlegen des Primärschlüssel
- Festlegen von Fremdschlüssel
- Deklarieren zusätzlicher Suchindexe zur Performanceoptimierung
- Definition von Zugriffsrechten

Um die Tabelle zu erstellen, benötigt man den `CREATE TABLE SQL`-Befehl.
Dieser fügt noch keine Daten ein, sondern erstellt nur ein leeres Grundgerüst.

DDL - DATA DEFINITION LANGUAGE

Erlaubt Operationen auf den Definitionen von Datenstrukturen:

- **CREATE** – Tabellen erzeugen
- **ALTER** – Tabellen verändern
- **DROP** – Tabellen löschen
- **TRUNC** – Daten aber nicht die Tabelle löschen

Die möglichen Datenstrukturen sind dabei:

- **TABLE** – Tabellen zum speichern von Daten
- **VIEW** – Dynamisch generierte Ansichten auf Tabellen
- **MATERIALIZED VIEW** – Tabellen die sich aus anderen Tabellen ableiten

TABELLEN ERSTELLEN IN SQL

```
CREATE TABLE table_name (  
    attribute1_name attribute1_type attribute1_constraints,  
    attribute2_name attribute2_type attribute2_constraints,  
    .../  
    table_constraints  
)
```

DATENTYPEN IN SQLITE VS. PYTHON

SQLite	Python	Beschreibung
Boolean (INTEGER)	bool	Wahrheitswerte
INTEGER	int	Ganzzahl
REAL	float	Gleitkommazahl
NUMERIC	-	Interpretiert beliebige Dateneingaben als Zahl
TEXT	str	Text-String, gespeichert in UTF-8 oder UTF-16
BLOB	bytes	Beliebiger Block an Binärdaten
NULL	None	Keine Daten

BEISPIEL: TABELLEN ERSTELLEN IN SQLITE

Für unser Geometriebeispiel ergeben sich folgende CREATE TABLE Statements:

```
CREATE TABLE Points (  
    point_id INTEGER,  
    x REAL,  
    y REAL  
);
```

```
CREATE TABLE Lines (  
    lines_id INTEGER,  
    start INTEGER,  
    end INTEGER  
);
```


SCHLÜSSEL UND CONSTRAINTS IN SQL

- Primär- und Sekundärschlüssel werden meistens durch INTEGER-Spalten abgebildet.
- Der Primärschlüssel jeder Relation muss speziell markiert werden durch den PRIMARY KEY Constraint.
- Weitere Constraints die beim Einfügen neuer Daten eingehalten werden:
- **NOT NULL** – Werte dürfen nicht null sein
- **AUTOINCREMENT** – Numerische Primärschlüssel werden automatisch berechnet
- **UNIQUE** – Werte müssen eindeutig sein, keine Dopplungen erlaubt
- **CHECKED** – Zusätzliche logische Bedingung für neue Werte
- **FOREIGN KEY** – Wert eines existierenden Fremdschlüssels

BEISPIEL: PRIMÄRSCHLÜSSEL, FREMDSCHLÜSSEL UND CONSTRAINTS

```
CREATE TABLE Points (  
    point_id INTEGER PRIMARY KEY AUTOINCREMENT,  
    x REAL NOT NULL,  
    y REAL NOT NULL  
);  
  
CREATE TABLE Lines (  
    lines_id INTEGER PRIMARY KEY AUTOINCREMENT,  
    start INTEGER NOT NULL,  
    end INTEGER NOT NULL,  
    FOREIGN KEY(start) REFERENCES Points(point_id),  
    FOREIGN KEY(end) REFERENCES Points(point_id)  
);
```

CREATE TABLE ERWEITERTE FUNKTIONEN

Zusätzliche Optionen:

- **DEFAULT** : Standardwert für neue Elemente festlegen
- **WITHOUT ROWID** : Tabellen die nicht durchnummeriert werden
- **STRICT** : Tabelle forciert Datentypen strikt

DATEN ZU TABELLEN HINZUFÜGEN

Daten werden in Tabellen mit dem SQL-Befehl **INSERT** hinzugefügt

Daten hinzufügen in der Reihenfolge der Spaltennamen:

```
INSERT INTO Points VALUES (1,  
54.083336, 12.108811);  
INSERT INTO Points VALUES (2,  
12.094167, 54.075211);  
INSERT INTO Lines VALUES (1, 1, 2);
```

oder mit expliziten Spaltennamen:

```
INSERT INTO Points (x, y)  
VALUES (54.083336, 12.108811);  
INSERT INTO Points (x, y)  
VALUES (12.094167, 54.075211);  
INSERT INTO Lines (start, end)  
VALUES (1, 2);
```

TABELLE AUS BESTEHENDEN DATEN ERSTELLEN

```
CREATE TABLE tablename AS  
SELECT ... FROM ... WHERE ...
```

Hat keinen Primärschlüssel, kann nur nachträglich definiert werden

ALTER TABLE - STATEMENT

Ändern von Tabellen und Spaltennamen, Hinzufügen und Löschen von Spalten

```
ALTER TABLE tablename TO newtablename
```

Beispiel:

```
ALTER TABLE Points RENAME TO Punkte
```

ALTER TABLE - BEISPIEL

Besitzer zu Kunde umbenennen, Spalte für Kontonummer hinzufügen

```
ALTER TABLE tablename ADD COLUMN column_name column_type
```

Beispiel:

```
ALTER TABLE PolygonType ADD COLUMN maxPoints INTEGER
```

DROP TABLE - STATEMENT

Das Löschen von kompletten Tabellen.

```
DROP TABLE tablename
```

Beispiel:

```
DROP TABLE Points
```


fragen?

