Datenbanken

Relationale Algebra

Operation	Symbol
Selektion	$\sigma_B(R)$
Projektion	$\pi_{A_1,A_2,...,A_n}(R)$
Vereinigung	$R \cup S$
Bag-Concatenation	$R \sqcup S$
Schnittmenge	$R \cap S$
Differenz	$R - S$ oder $R \backslash S$
Division (Alle A_i in R haben zugehörige B_j in S)	$R \div S$
Kartesisches Produkt	$R \times S$
Natural Join	$R \bowtie_B S$
Full Outer Join	$R \ \overset{\bowtie}{\bowtie}_B \ S$
Left Outer Join	$R \ \overset{\bowtie}{\bowtie}_B \ S$
Right Outer Join	$R \ \overset{\bowtie}{\bowtie}_B \ S$
Umbenennung	$\rho_{S(B_1,B_2,...,B_n)}(R)$
AND-Verknüpfung	$\sigma_{B_1 \wedge B_2}(R)$
OR-Verknüpfung	$\sigma_{B_1 \vee B_2}(R)$
Negation	$\sigma_{\neg B}(R)$
Duplikatelöschung	$\delta(R)$

B: Bedingung, A_i: Attribut, R, S: Relationen

Bag Algebra bedeutet, dass Duplikate erlaubt sind.

Äquivalenzregeln

- Selektion und Projektion sind kommutativ:

$$\sigma_{B_1}(\sigma_{B_2}(R)) = \sigma_{B_2}(\sigma_{B_1}(R))$$
$$\pi_{A_1,A_2}(\pi_{A_2,A_1}(R)) = \pi_{A_1,A_2}(R)$$

- Selektion verteilt sich über Vereinigung, Schnittmenge und Differenz:

$$\sigma_B(R \cup S) = \sigma_B(R) \cup \sigma_B(S)$$
$$\sigma_B(R \cap S) = \sigma_B(R) \cap \sigma_B(S)$$
$$\sigma_B(R - S) = \sigma_B(R) - \sigma_B(S)$$

- Projektion über Vereinigung:

$$\pi_{A_1,A_2}(R \cup S) = \pi_{A_1,A_2}(R) \cup \pi_{A_1,A_2}(S)$$

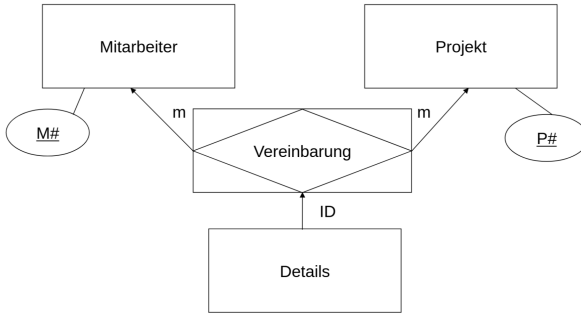
- Selektion über Join:

$$\sigma_B(R \bowtie S) = \sigma_B(R) \bowtie S$$

(wenn B nur Attribute von R betrifft)

Entity-Relationship-Modell

- **ISA-Beziehung**: Vererbung von Attributen und Beziehungen
  - Genauere Spezifikation
  - Vermeidung von Redundanzen
- **ID-Abhängigkeit**: Existenzabhängigkeit eines Entities von einem anderen
  - Ist ein Entity, dass ohne das übergeordnete Entity nicht existieren kann
- **Zusammengesetzter Entitätstyp**: Entity-Typ, der aus mehreren Entity-Typen besteht
  - Entsteht durch die Kombination Beziehungstypen und Entity-Typen
  - Wird durch eine Raute in einem Rechteck dargestellt



Normalformen

- **1. Normalform (1NF)**: Alle Attribute sind atomar
- **2. Normalform (2NF)**: Jedes Nicht-Schlüsselattribut ist voll funktional abhängig vom gesamten Primärschlüssel
- **3. Normalform (3NF)**: Kein Nicht-Schlüsselattribut ist transitiv abhängig vom Primärschlüssel
- **Boyce-Codd-Normalform (BCNF)**: Alle Attribute hängen nur vom Primärschlüssel ab

SQL Befehle

Create Table

```
CREATE TABLE table_name (
    column1 datatype [NOT NULL, UNIQUE],
    column2 datatype [NOT NULL, UNIQUE],
    ...
    PRIMARY KEY (column_name)
);
```

Create Foreign Key

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    ...
    FOREIGN KEY (col) REFERENCES other_tbl(col)
);
```

Delete

```
DELETE FROM table_name
WHERE condition [CASCADE];
```

Insert

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

Update

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition [CASCADE];
```

Select

```
  SELECT [DISTINCT] [AGG] column1, column2, ...
FROM table_name
WHERE [EXISTS] condition;
```

Join

```
SELECT columns
FROM table1
JOIN table2
ON table1.column = table2.column;
```

Vergleichsoperatoren

Operator	Bedeutung
=	Gleich
<> oder ! =	Ungleich
>	Größer als
<	Kleiner als
>=	Größer oder gleich
<=	Kleiner oder gleich
LIKE	"_" für ein Zeichen, "%" für beliebig viele Zeichen
IN	Wert ist in einer Liste von Werten (kombinierbar mit ANY)
BETWEEN	Wert liegt in einem Bereich
IS NULL	Wert ist NULL
IS NOT NULL	Wert ist nicht NULL
=~	Ähnlich (Regulärer Ausdruck)

Aggregatfunktionen

- **COUNT()**: Zählt die Anzahl der Zeilen
- **SUM()**: Berechnet die Summe der Werte in einer Spalte
- **AVG()**: Berechnet den Durchschnitt der Werte in einer Spalte
- **MIN()**: Gibt den kleinsten Wert in einer Spalte zurück
- **MAX()**: Gibt den größten Wert in einer Spalte zurück

Gruppierung

```
SELECT column1, AGG(column2)
FROM table_name
GROUP BY column1;
```

Sortierung

```
SELECT column1, column2
FROM table_name
ORDER BY column1 [ASC|DESC];
```

HAVING (Gruppierungsbedingung)

```
SELECT column1, AGG(column2)
FROM table_name
GROUP BY column1
HAVING AGG(column2) condition;
```

CASE

```
SELECT column1,
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    ELSE resultN
END AS new_column
FROM table_name;
```

Constraints

- **BASIC**: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY
- **CHECK**: Stellt sicher, dass alle Werte in einer Spalte eine bestimmte Bedingung erfüllen
- **DEFAULT**: Legt einen Standardwert für eine Spalte fest, wenn kein Wert angegeben ist

Triggers and Stored Procedures

- **Stored Procedures**: Vorgefertigte SQL-Befehle, die aufgerufen werden können
- **Triggers**: Automatisch ausgelöste Aktionen bei bestimmten Ereignissen (INSERT, UPDATE, DELETE)

Stored Procedures

```
CREATE PROCEDURE procedure_name (parameters)
BEGIN
    SQL statements;
[EXCEPTION
    WHEN condition THEN
        SQL statements; ]
END;

CALL procedure_name (arguments);
```

Triggers

```
CREATE TRIGGER trigger_name
BEFORE|AFTER INSERT|UPDATE|DELETE
ON table_name
FOR EACH ROW
BEGIN
    SQL statements;
END;
```

Datenorganisation

- **Heap-Dateien**: Unsortierte Dateien, schnelle Einfügungen, langsame Suche
- **Sortierte Dateien**: Sortierte Daten, schnelle Suche, langsame Einfügungen
- **Hash-Dateien**: Schnelle Suche, langsame Einfügungen und Löschungen
- **ISAM**: Index-sequentielle Dateien, schnelle Suche und Einfügungen
- **B-Bäume**: Balancierte Bäume, schnelle Suche, Einfügungen und Löschungen

Transaktionen

- **ACID-Prinzipien**:
  - **Atomicity**: Transaktionen sind unteilbar
  - **Consistency**: Datenbank bleibt konsistent
  - **Isolation**: Transaktionen beeinflussen sich nicht gegenseitig
  - **Durability**: Änderungen sind dauerhaft
- **Sperrprotokolle**: Verhindern Inkonsistenzen durch gleichzeitige Transaktionen
- **Deadlocks**: Situationen, in denen Transaktionen sich gegenseitig blockieren

```
BEGIN TRANSACTION;
-- SQL-Befehle
COMMIT; -- oder ROLLBACK;
```