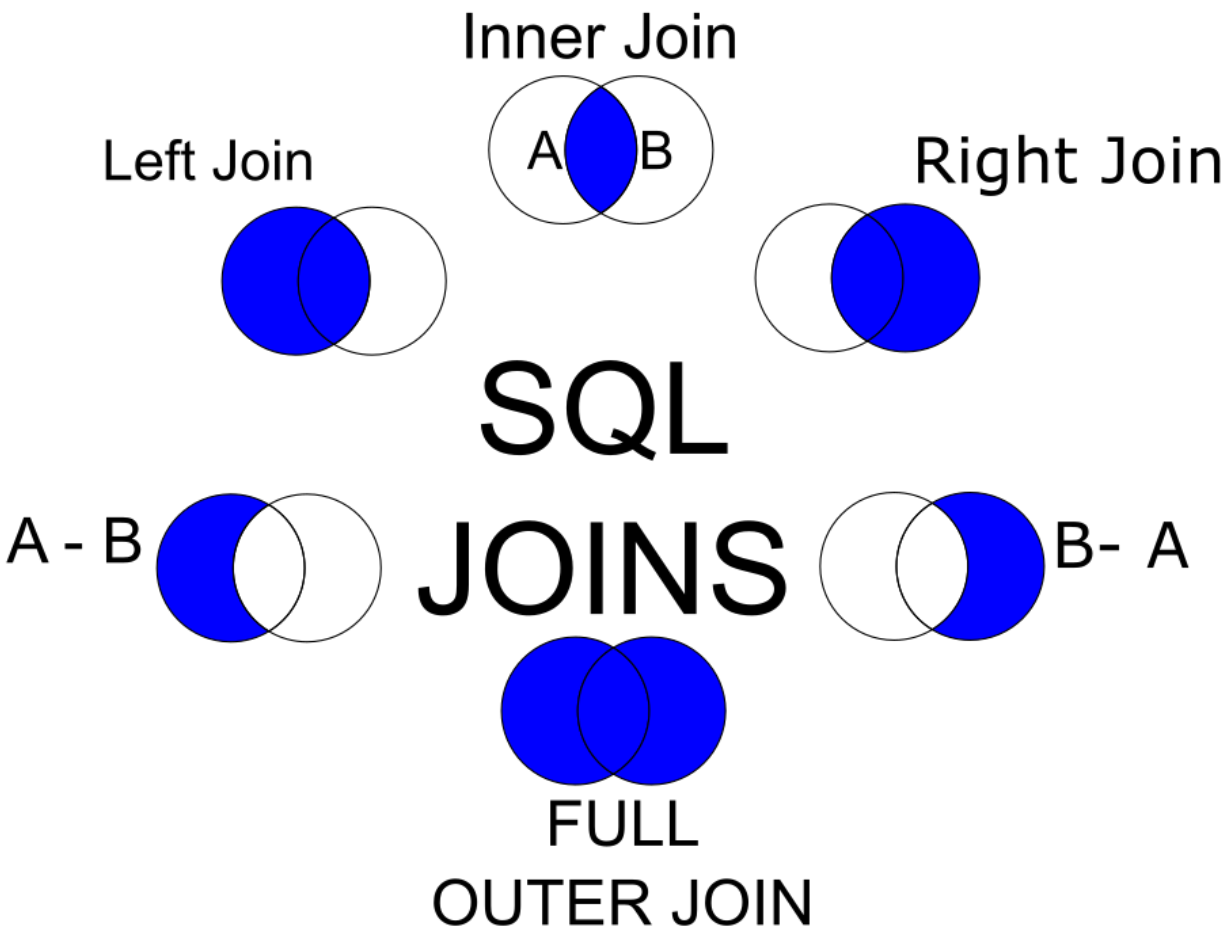
Joins

In [2]:
```
# Hier ist nur Code zum Initialisieren der Umgebeung, bitte gehen Sie weiter, es gibt nich

# Keine langen Fehlermeldungen
import sys
ipython = get_ipython()

def exception_handler(exception_type, exception, traceback):
    print("%s: %s" % (exception_type.__name__, exception), file=sys.stderr)

ipython._showtraceback = exception_handler

# Lade die Erweiterung, damit wir SQL Befehle nutzen können
%reload_ext sql
# Verbinde Dich zu einer in - Memory Datenbank
%sql sqlite:///
%sql PRAGMA foreign_keys = ON
```

Out[2]:
```
 * sqlite:///
Done.
[]
```

Die verschiedenen Joins

Es gibt verschiedene Arten Joins zu erklären. Ich bevorzuge immer noch folgendes Bild:



In [3]:
```
%%sql

DROP TABLE IF EXISTS eins;
DROP TABLE IF EXISTS zwei;

CREATE TABLE eins(id INT PRIMARY KEY, val CHAR(1));
INSERT INTO eins VALUES(1,'A'), (2,'B'), (3,'C');
CREATE TABLE zwei(id INT PRIMARY KEY, val2 CHAR(1));
INSERT INTO zwei VALUES(2,'D'), (3,'E'), (4,'F');
```

```
 * sqlite:///
Done.
Done.
Done.
3 rows affected.
Done.
3 rows affected.
```

Out[3]:
```
[]
```

Aufgabe 1

Führen sie an den Tabellen `eins, zwei` verschiedene `JOINS` durch, um ein Gefühl für die veränderte Ergebnissmenge zu erhalten. Nachdem beide Tabellen gleich benannte `id` Attribute haben können Sie bei einem `JOIN` auf diese einen `NATURAL JOIN` oder `JOIN using(id)` oder `JOIN ON(eins.id = zwei.id)` verwenden.

Aufgabe 1.1 Implementieren Sie einen `INNER` Join in den drei oben genannten Varianten

In [4]:
```
%%sql
SELECT *
FROM eins
NATURAL JOIN zwei;
```

```
 * sqlite:///
Done.
```

Out[4]:

id	val	val2
2	B	D
3	C	E

In [5]:
```
%%sql
SELECT *
FROM eins
JOIN zwei USING(id);
```

```
 * sqlite:///
Done.
```

Out[5]:

id	val	val2
2	B	D
3	C	E

In [6]:
```
%%sql
SELECT *
FROM eins
JOIN zwei ON(eins.id = zwei.id);
```

```
 * sqlite:///
Done.
```

Out[6]:

id	val	id_1	val2
2	B	2	D
3	C	3	E

Aufgabe 1.2 Implementieren Sie einen `LEFT Join`

In [7]:

```
%%sql
SELECT *
FROM eins
LEFT JOIN zwei USING(id);
```

```
 * sqlite:///
Done.
```

Out[7]:

id	val	val2
1	A	None
2	B	D
3	C	E

Aufgabe 1.3 Implementieren Sie einen `RIGHT Join`

In [8]:

```
%%sql
SELECT *
FROM eins
-- Das geht nicht, da es diesen Befehl nicht gibt
RIGHT JOIN zwei
USING(id);
```

```
 * sqlite:///
(sqlite3.OperationalError) RIGHT and FULL OUTER JOINs are not currently supported
[SQL: SELECT * FROM eins
-- Das geht nicht, da es diesen Befehl nicht gibt
RIGHT JOIN zwei
USING(id);]
(Background on this error at: https://sqlalche.me/e/14/e3q8)
```

In [9]:

```
%%sql
SELECT id, val, val2
FROM zwei
LEFT JOIN eins USING(id);
```

```
 * sqlite:///
Done.
```

Out[9]:

id	val	val2
2	B	D
3	C	E
4	None	F

Aufgabe 1.4 Implementieren Sie `A – B`

In [10]:

```
%%sql
SELECT *
```

```
FROM eins
LEFT JOIN zwei USING(id)
WHERE val2 IS NULL;
```

```
 * sqlite:///
Done.
```

Out[10]:

id	val	val2
1	A	None

Aufgabe 1.5 Implementieren Sie `B - A`

In [11]:

```
%%sql
SELECT *
FROM zwei
LEFT JOIN eins USING(id)
WHERE val IS NULL;
```

```
 * sqlite:///
Done.
```

Out[11]:

id	val2	val
4	F	None

Aufgabe 1.5 Implementieren Sie einen `FULL OUTER JOIN`

In [12]:

```
%%sql
SELECT DISTINCT *
FROM
 (
    SELECT id, val, val2
    FROM eins
    LEFT JOIN zwei
    USING(id)
    UNION ALL
    SELECT id, val, val2
    FROM zwei
    LEFT JOIN eins
    USING(id)
 ) AS gemeinsam
ORDER BY id;
```

```
 * sqlite:///
Done.
```

Out[12]:

id	val	val2
1	A	None
2	B	D
3	C	E
4	None	F

Aufgabe 2

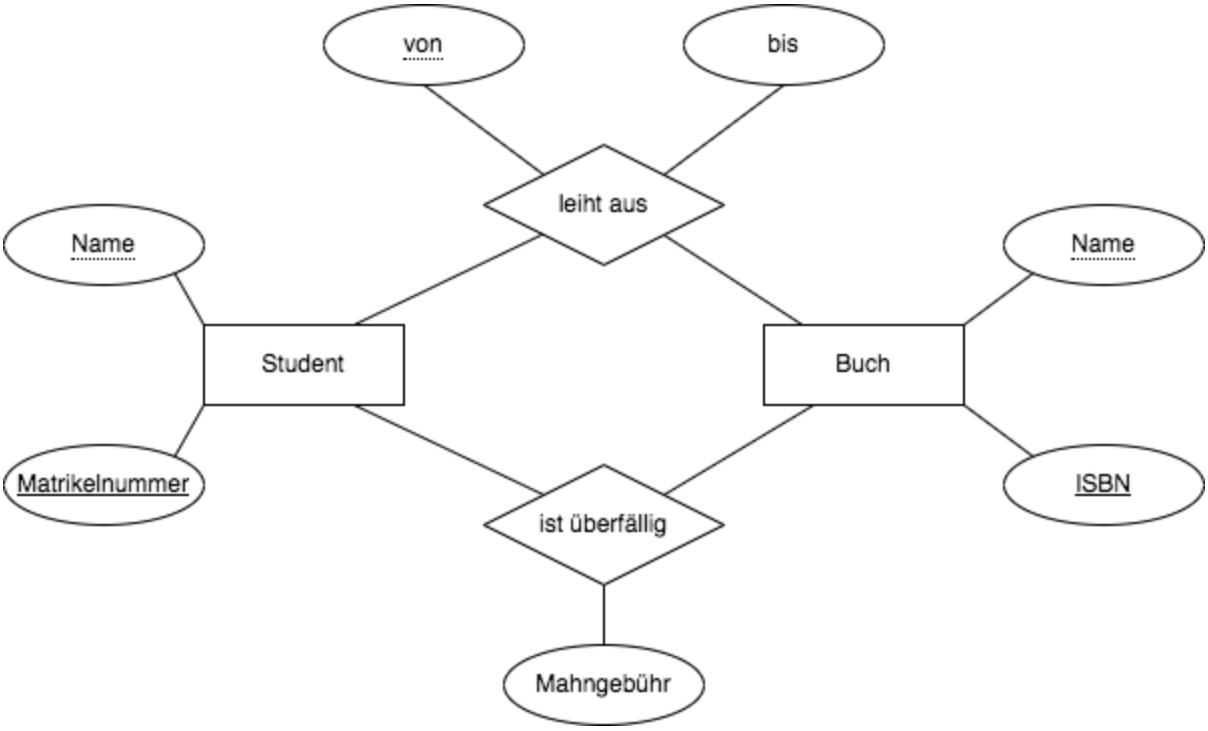
Die Beispiele aus Aufgabe 1 waren bewusst abstrakt gewählt. Was heißt die jeweilige Form von Joins denn auf einen konkreten Use-Case? Was würde jede einzelne Join form bedeuten, wenn es um Stundenten und Bücher ginge, die Studenten ausleihen können?

- Inner Join:
- Left Outer Join:
- Right Outer Join:
- Student - Buch:
- Buch - Student:
- FULL OUTER JOIN:

- Inner Join: *Studenten, die gerade Bücher ausgeliehen haben und welche Bücher das sind*
- Left Outer Join: *Wie INNER JOIN, nur inkl. aller Studenten, auch die, die keine Bücher ausgeliehen haben*
- Right Outer Join: *WIE INNER JOIN nur inkl. aller Bücher in der Bib*
- Student - Buch: *Studenten, die kein Buch ausgeliehen haben*
- Buch - Student: *Bücher, die nicht ausgeliehen sind*
- FULL OUTER JOIN: *Alles studenten und alle Bücher*

Aufgabe 3

Betrachten wir folgene Modellierung:



In [52]:

```
%%sql

DROP TABLE IF EXISTS leiht_aus;
DROP TABLE IF EXISTS ist_ueberfaellig;
DROP TABLE IF EXISTS student;
DROP TABLE IF EXISTS buch;

CREATE TABLE student(
    matrikelnummer INT PRIMARY KEY NOT NULL,
    name TEXT NOT NULL
);

CREATE TABLE buch(
    isbn TEXT PRIMARY KEY NOT NULL,
    name TEXT NOT NULL
);
```

```
CREATE TABLE leiht_aus(
    matrikelnummer INT NOT NULL,
    isbn TEXT NOT NULL,
    von DATE NOT NULL,
    bis DATE,
    PRIMARY KEY(matrikelnummer, isbn, von),
    FOREIGN KEY(matrikelnummer) REFERENCES student(matrikelnummer),
    FOREIGN KEY(isbn) REFERENCES buch(isbn)
);

CREATE TABLE ist_ueberfaellig(
    matrikelnummer INT NOT NULL,
    isbn TEXT NOT NULL,
    gebuer INT,
    PRIMARY KEY(matrikelnummer, isbn),
    FOREIGN KEY(matrikelnummer) REFERENCES student(matrikelnummer),
    FOREIGN KEY(isbn) REFERENCES buch(isbn)
);
```

```
 * sqlite:///
Done.
Done.
Done.
Done.
Done.
Done.
Done.
Done.
```

Out[52]:
```
[]
```

In [53]:
```
%%sql
INSERT INTO student VALUES(1,'Elser'), (2, 'Maier'), (3, 'Huber');
INSERT INTO buch VALUES(1,'Datenbanken'), (2, 'Ulysses'), (3, 'Das Niebelungenlied');
INSERT INTO ist_ueberfaellig VALUES(1,2,10), (2,2,15), (2,3,22);
INSERT INTO leiht_aus VALUES(1,1,'01.10.2021',NULL), (1,2,'01.11.2021',NULL), (2,3,'01.11.
```

```
 * sqlite:///
3 rows affected.
3 rows affected.
3 rows affected.
3 rows affected.
```

Out[53]:
```
[]
```

Aufgabe 3.1

Entwickeln Sie einen JOIN, der angibt welche Studenten welche Bücher ausgeliehen haben und welche Gebüren da noch zu zahlen sind.

Achtung die Lösungen sind Vorschläge, die JOIN Prädikate demonstrieren sollen, es gibt deutlich mehr Lösungen für das Problem

In [56]:
```
%%sql
SELECT s.name, b.name, l.von, l.bis, u.gebuer
FROM student s
LEFT JOIN leiht_aus l USING(matrikelnummer)
LEFT JOIN buch b USING(isbn)
LEFT JOIN ist_ueberfaellig u ON(s.matrikelnummer = u.matrikelnummer AND b.isbn = u.isbn)
```

```
 * sqlite:///
Done.
```

Out[56]:

name	name_1	von	bis	gebuer
Elser	Datenbanken	01.10.2021	None	None
Elser	Ulysses	01.11.2021	None	10
Maier	Das Niebelungenlied	01.11.2021	11.11.2021	22
Huber	None	None	None	None

Aufgabe 3.2

Geben Sie nur ausleihen aus, die > 20 Euro Gebüren haben

In [58]:

```
%%sql
SELECT s.name, b.name, l.von, l.bis, u.gebuer
FROM student s
LEFT JOIN leiht_aus l USING(matrikelnummer)
LEFT JOIN buch b USING(isbn)
JOIN ist_ueberfaellig u ON(s.matrikelnummer = u.matrikelnummer AND b.isbn = u.isbn AND u.g
```

```
 * sqlite:///
Done.
```

Out[58]:

name	name_1	von	bis	gebuer
Maier	Das Niebelungenlied	01.11.2021	11.11.2021	22

In [ ]: