Beispiel:

Preisgruppen dienen im Beispiel der Datenbank `Faradiso` zur Zusammenfassung von Fahrrädern mit gleichem Verleihpreis.

Der Verleihpreis der Fahrräder, die vor 2016 angeschafft wurden, soll um eine Preisgruppe vermindert werden. Die Änderungen sollen in den betroffenen Datensätzen gespeichert werden.

Lösung:

```
UPDATE Fahrraeder
SET Preisgruppe = Preisgruppe - 1
WHERE YEAR(Anschaffungsdatum) < 2016;
```

Hier wird in allen Datensätzen der Tabelle `Fahrraeder`, die der `WHERE`-Bedingung entsprechen und vor dem Jahr 2016 angeschafft wurden, der Inhalt des Feldes `Preisgruppe` um 1 vermindert.

6.5 Konsistenz der Datenbank

Die **Konsistenz** einer Datenbank beschreibt die Korrektheit der internen Speicherungsstrukturen und der Zugriffspfade, um die Datenbestände stets widerspruchsfrei nutzen zu können.

Da Datenbanken in der Regel von mehreren Anwendern gleichzeitig verwaltet werden, können z. B. Löschvorgänge von Daten nicht mehr rückgängig gemacht werden. Denn nach der Löschung könnten sich weitere Anwender auf die inzwischen geänderten Datenbestände bereits verlassen haben. Durch diese Verhinderung des Rückgängigmachens werden somit Anomalien (logische Widersprüche) im Datenbestand verhindert, die Daten sind weiterhin konsistent (widerspruchsfrei).

Beispiel:

Die Datenbankstruktur muss sicherstellen, dass eine einmal gespeicherte Kundenadresse nicht einer weiteren Adressangabe dieses Kunden widerspricht. Sonst wäre das Zusenden einer Rechnung nicht mehr möglich.

Bereits bei der Planung einer Datenbank zu verhindernde Anomalien sind Löschanomalien, Änderungsanomalien und Einfügeanomalien.

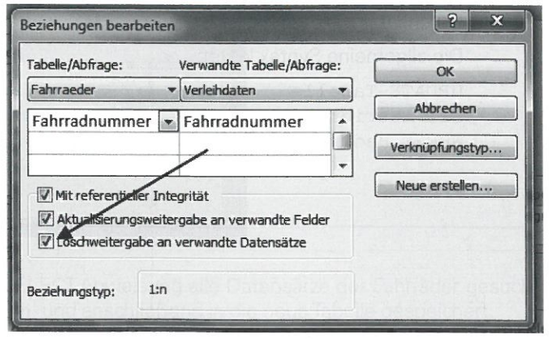
Löschanomalien entstehen, wenn ein Datensatz gelöscht wird, auf dessen Primärschlüsselwert sich ein abhängiger Datensatz einer anderen Tabelle bezieht. Würde z. B. der Datensatz des Kunden mit der Kundennummer 15 gelöscht werden, so könnten Verleihdaten, die sich auf die Kundennummer 15 beziehen, nicht mehr zugeordnet werden.

Änderungsanomalien entstehen z. B. durch Mehrfachspeicherung von Daten. Werden die Daten (z. B. die Kundenadresse) in einer Tabelle geändert, so widersprechen sie den Daten in einer anderen Tabelle.

Einfügeanomalien entstehen dann, wenn nach dem Einfügen von Datensätzen Werte mehrfach in der Datenbank vorkommen und sich dadurch widersprechen.

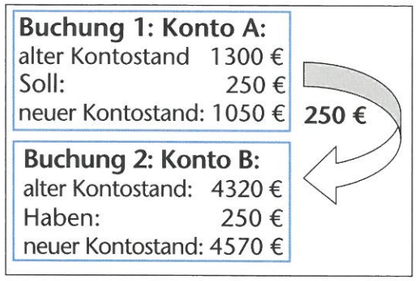
Um **Anomalien** zu verhindern, werden bereits beim Entwurf der Datenbank die Beziehungen so festgelegt, dass Löschvorgänge überwacht werden.

Wird z. B. in der Tabelle `Fahrraeder` ein Fahrrad gelöscht, auf dessen Fahrradnummer ein Datensatz in der Verleihtabelle verweist, so sollen ebenfalls die Verleihdatensätze dieser Fahrradnummer gelöscht werden. Dies geschieht z. B. in Access durch Auswahl der Checkbox `Löschweitergabe an verwandte Datensätze` im Fenster `Beziehungen bearbeiten`:



6.6 Transaktionen

Datenbanken müssen zu jedem Zeitpunkt, auch nach einem Hardware- oder Softwarefehler, konsistent sein. Bei einer Überweisung zwischen Konten zweier Banken darf die Überweisung nur dann gültig sein, wenn das Konto A um den Betrag vermindert worden ist und dem Konto B dieser Betrag gutgeschrieben worden ist.



Erfolgt nach der Buchung 1 ein Stromausfall, so wären die Kontostände nicht mehr konsistent, da der Kontostand A bereits um 250 € vermindert, der Kontostand B aber noch nicht erhöht wurde.

Um sicher zu gehen, dass die Überweisung vollständig durchgeführt wird, werden die beiden Buchungen in Form einer Transaktion ausgeführt.

Definition:

Unter einer **Transaktion** versteht man eine Folge von SQL-Anweisungen, die logisch zusammengehören und den Datenbestand konsistent erhalten.

Hinweis:

Beim Ausführen einer Transaktion muss sichergestellt werden, dass die Transaktion entweder komplett ausgeführt wird oder im Fehlerfall völlig rückgängig gemacht wird.

Nachfolgende Anweisungen werden nur dann ausgeführt, wenn die vorhergehenden erfolgreich durchgeführt wurden.

Um Datenbestände immer gültig zu erhalten, wird in einzelnen Schritten vorgegangen:

Schritte	Aufgabe
1. Lesen der Daten	Die Daten werden vom Speichermedium eingelesen.
2. Merken der bisherigen Daten	Die zu ändernden Daten werden in die Logdatei geschrieben (Before-Image).
3. Ändern der Daten	Ändern der Daten im Arbeitsspeicher, Sperren dieser Einträge für andere Benutzer (Datensatz loggen).
4. Merken der geänderten Daten	Die geänderten Daten werden in die Logdatei geschrieben (After-Image).
5. Transaktionsende mit `COMMIT` (= Bestätigung)	Schreiben aller Images und Metadaten in die Logdatei. Transaktionsende in der Logdatei vermerken. Sperren freigeben.
6. Transaktionsende mit `ROLLBACK` (= zurückdrehen)	Rücksetzen der Metadaten der Transaktion. Geänderte Daten, die bereits in die Datenbank geschrieben wurden, werden für ungültig erklärt. Sperren freigeben.
7. Änderungen speichern	Die geänderten Daten werden in die Datenbank geschrieben.