Zürcher Hochschule für Angewandte Wissenschaften

School of Engineering

Datenbanken

1 Grundlagen

1.1 Daten, Informationen, Wissen

- Lateinisch datum: gegebenes
- Angaben über Dinge und Sachverhalte, die elektronisch gespeichert und verarbeitet werden können
- Daten sind strukturierte Zeichen
- Daten in einem Kontext sind Informationen
- Verknüpfte Informationen zur intellektuellen Einbettung ist Wissen
- Dies ist keine präzise Definition

1.2 Datenmanagement

- Methodische, konzeptionelle, organisatorische und technische Massnahmen und Verfahren zur Behandlung der Ressource Daten
- Daten mit ihrem maximalen Nutzungspotenzial in die Geschäftsprozesse einzubringen
- Im laufenden Betrieb die optimale Nutzung der Daten gewährleisten

1.3 Strukturierte Daten

- Feste vorgegebene Struktur
- Mehrer gleichartige Datensätze mit identischem Aufbau
- Gut tabellarisch darstellbar
- Relationale Datenbanken

1.3.1 Unstrukturierte Daten

- Keine explizite Struktur
- Implizite Struktur möglich (Syntax)
- Texte, Bilder, Filme, Audiodaten

1.3.2 Semistrukturierte Daten

- Teilweise irreguläre bzw. unvollständige Strukturen
- Können sich öfters ändern
- Schema kann aus zusätzlichen Informationen rekonstruiert werden
- Suchergebnise, E-Mail, XML, JSON

1.4 Persistenz

- Lateinisch persistere: beharren
- Daten über lange Zeit bereitzuhalten
- Benötigt nichtflüchtige Speichermedien
- Partnerbegriff: Volatilität

1.5 Datenverwaltung mittels Dateisystemen

- Speicherung von Daten in Dateien
- Anwendungen/Programme lesen/schreiben Daten direkt
- Logische und physische Datenabhängigkeit

1.5.1 Vorteile

- Einfach, auf Anwendung angepasst, effizient implementierbar
- Anwendung muss keine Rücksicht nehmen auf „andere“
- Proprietäre (nicht allgemein anerkannten Standards entsprechend) Formate möglich

1.5.2 Nachteile

- Probleme bei Mehrfachverwendung der Daten für unterschiedliche Zwecke
- Datenstrukturänderung bedeutet i.d.R. Programmänderung
- Gleichzeitiger Zugriff aufwändig zu realisieren
- Abgestufte Zugriffsrechte aufwändig zu realisieren
- Daten werden oft mehrfach gespeichert
- Datenaustausch, -integration komplex

1.6 Datenbanksystem DBS

- Ein Datenbanksystem DBS ist ein Datenbankverwaltungssystem DBMS inklusive Datenbank
- Ein DBS mit Anwendungsprogramm AP ergeben ein Informationssystem IS

1.6.1 Vorteile

- Datenkonsistenz (= Datenintegrität) einfacher sicherzustellen
- Mehrere Anwendungen können gleichzeitig auf dieselben Daten zugreifen
- Anwendungen unabhängig von physischer Datenstruktur
- Anwendungen unabhängig von Erweiterungen der Daten
- Verwaltung und Nutzung sehr grosser Datenmengen
- Deklarativer und mengenorientierter Zugriff
- Einmalige, zentrale Datendefinition
- Automatisierung wichtiger Aufgaben (Integritätskontrolle, Redundanzverwaltung, Zugriffskontrolle, Zugriffsoptimierung, GleichzeitigerZugriff, Datensicherung und -wiederherstellung)

1.6.2 Nachteile

- Aufbau und Betrieb sind anspruchsvoll und teuer

1.7 Datenbankverwaltungssystem DBMS

- Effiziente und flexible Verwaltung grosser Mengen persistenter Daten
- Werden verwendet um strukturierte Daten zu verwalten
- Hoher Grad an Datenunabhängigkeit
- Hohe Leistung und Skalierbarkeit
- Mächtige Datenmodelle und Abfragesprachen / leichte Handhabbarkeit
- Transaktionskonzept (ACID), Datenkontrolle
- Ständige Betriebsbereitschaft (hohe Verfügbarkeit und Fehlertoleranz)

1.8 Relationale Datenbanken

- Werden verwendet um strukturierte Daten zu verwalten
- Bedeutenste Datenbanktechnologie in der Praxis

1.8.1 Drei Ebenen Architektur

- Konzeptionell: Logische Gesamtsicht
- Extern: Sicht einer Anwendung
- Intern: Speicherung, Datenorganisation, Zugriffsstruktur
- Realisiert durch ein RDBMS
- Datenbeschreibungen ebenfalls in der Datenbank gespeichert
- Zwischen den verschiedenen Ebenen (Schemas) erfolgen Transformationen
- Logische Datenunabhängigkeit: Externes Schema ändert → modifizieren der Transformationsregeln zum konzeptionellen Schema
- Physische Datenunabhängigkeit: Internes Schema ändert → modifizieren der Transformationsregeln zum konzeptionellen Schema

1.9 Datenbankentwurf/-betrieb

1.9.1 Aufbau

- Erstellen von verschiedenen Schemas
- Iterativer Prozess
- Ausbau und Umbau im laufenden Betrieb typisch
- ERM (Entity-Relationship-Model)
- DDL (Data Definition Language)

1.9.2 Betrieb, Benutzung, Verwaltung

- Abfragen, Einfügen, Ändern und Löschen von Daten
- Sicherung und Wiederherstellung
- Überwachung und Tuning
- Benutzerverwaltung und Rechtevergabe
- DML (Data Manipulation Language)
- DQL (Data Query Language)
- DCL (Data Control Language)

2 Relationenmodell

2.1 Wertebereich (Domäne)

- Menge einfacher/atomarer Werte
- Entspricht im wesentlichen einem Datentyp einer höheren Programmiersprache
- Mengen von Werten sind nicht zulässig

2.2 Attribut

- Besteht aus Bezeichnung/Name und Domäne/Wertebereich
- Ein Attribut nimmt konkrete Werte an
- Attributwerte können sich im Laufe der Zeit ändern
- Attribute (Name, Bedeutung und Wertebereich) bleiben konstant

2.3 Tupel (n-Tupel)

- Sammlung von als zusammengehörig betrachteter Attribute
- Feste Zahl von Komponenten
- Beliebige Anordnung
- Der Attributwert entstammt einer für jedes Attribut festgelegten Domäne
- Eine Menge von gleichartig strukturierten Tupeln bildet eine Relation

2.4 Relation

- Besteht aus Relationsschema und Ausprägung
- Relationsschema/Format: Menge der Attribute mit Namen und Domäne
- Ausprägung: Menge von Tupeln
- Eine Relation ist eine Teilmenge des kartesischen Produktes von n endlichen Wertebereichen
- Relationen sind Mengen, enthalten also keine doppelten Elemente und sind ungeordnet
- Relationen werden oft als Tabellen dargestellt (Domänen werden dabei oft weggelassen)

2.5 Kurzschreibweisen

- Kurzschreibweise Format/Schema: R$(A_1, A_2, ..., A_n)$
- Kurzschreibweise Tupel: {<Max, Muster, Blau>,<Beni, Beispiel, Rot>,<Fritz, Müller, Schwarz>}

2.6 Äquivalenz

- Enthalten zwei Relationen die gleichen Attribute inklusive Domänen, sind sie äquivalent
- R1 $\sim$ R2

2.7 Schlüssel

2.7.1 Schlüsselkandidat

- Eine Teilmenge von Attributen K
- Es gibt nicht zwei Tupel mit denselben Schlüsselattributwerten in K
- In K kann nichts weglassen werden
- Wenn mehrere Schlüsselkandidaten zur Verfügung stehen, muss eine Auswahl getroffen werden

Orders

OrdNo	CNo	PNo	Qty	Amount	Status	ValidDate
1	1	1	100	1800.00	'paid'	2010-07-16
2	1	1	100	1800.00	'paid'	2010-07-21
3	1	2	4	9000.00	'paid'	2010-09-30

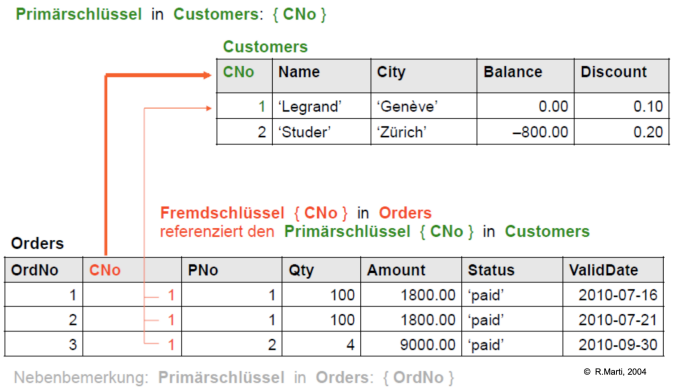
Schlüsselkandidaten: { OrdNo }, evt. { CNo, PNo, ValidDate }

2.7.2 Primärschlüssel

- Ein ausgewählter Schlüsselkandidat, der explizit als Primärschlüssel bezeichnet wird
- Attributwerte sollten sich möglichst wenig ändern
- Eindeutigkeit der Werte eines Primärschlüssels sollte über die Zeit gelten
- Attribute sollten möglichst wenig Speicherplatz benötigen
- Wenn nichts passt Surrogatschlüssel "künstlicher Schlüssel" definieren

2.7.3 Fremdschlüssel

- Eine Menge von Attributen in einer Relation S zu der es eine Relation R gibt, deren Primärschlüssel von diesen Attributen in S referenziert werden
- Referentielle Integrität: Ein Fremdschlüssel (in S) kann, muss aber nicht Schlüssel in S sein
- Primärschlüssel/Fremdschlüssel-Beziehungen müssen explizit deklariert werden



3 Relationale Algebra

- Operatoren und Rechenregeln mit Relationen
- Inhalt der Datenbank (Relationen) sind Operanden
- Operatoren definieren Funktionen zum Berechnen von Anfrageergebnissen
- Rangfolge: $\sigma, \pi, \rho \Rightarrow \times, \bowtie \Rightarrow \cap \Rightarrow \cup, \setminus$

3.1 Selektion σ

- Unärer Operator (d.h. nur ein Operand)
- Erzeugt neue Relation mit gleichem Schema aber einer Teilmenge der Tupel
- Nur Tupel, die der sogenannten Selektionsbedingung entsprechen, werden übernommen
- Ergebnis kann eine ohne Tupel/leere Relation sein
- Selektionsbedingung: Kombination von logischen Ausdrücken bestehend aus Attributen und/oder Konstanten
- Prüft Selektionsbedingung für jedes Tupel der Relation
- $R' = \sigma_{\text{Selektionsbedingung}}(R)$
- $r' = \sigma_{A=0 \vee C=2*B}(r)$
- $r' = \sigma_{A=0 \vee C=2*B}(r)$

3.2 Projektion π

- Unärer Operator (d.h. nur ein Operand)
- Erzeugt neue Relation mit einer Teilmenge der ursprünglichen Attribute
- Es können Duplikate entstehen, die entfernt werden müssen
- Die Projektion wählt eine Menge von Spalten aus
- Mit einer Projektion lässt sich auch die Reihenfolge der Spalten anpassen

- Die Menge der Attribute muss im Format vorhanden sein, sonst Fehler
- $R' = \pi_{\text{A1, A3, A12}}(R)$
- $r' = \pi_{\text{Name, Farbe}}(r)$

3.3 kartesisches Produkt $\times$

- Kreuzprodukt zweier Relationen R und S
- Menge aller Tupel, die entsteht, wenn jeder Tupel aus R mit jedem Tupel aus S kombiniert wird
- Schema hat ein Attribut für jedes Attribut aus R und S
- Bei Namensgleichheit wird kein Attribut weggelassen, stattdessen: Umbenennen oder qualifizieren

R

A	B
1	2
3	4

S

B	C	D
2	5	6
4	7	8
9	10	11

R × S

A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

3.4 Verbund, natural Join $\bowtie$

- Statt im Kreuzprodukt alle Paare zu bilden, sollen nur die Tupelpaare gebildet werden, deren Tupel übereinstimmen

R

A	B	C
1	2	3
6	7	8
9	7	8

S

B	C	D
2	5	6
2	3	5
7	8	10

R ⋈ S

A	B	C	D
1	2	3	5
6	7	8	10
9	7	8	10

3.5 Theta Join $\bowtie_P$

- Verallgemeinerung des natürlichen Joins, viel flexibler, darum in der Praxis der Normalfall
- Verknüpfungsbedingung kann frei gestaltet werden
- $R \bowtie_P S = \sigma_P(R \times S)$

R

A	B	C
1	2	3
6	7	8
9	7	8

S

B	C	D
2	5	6
2	3	5
7	8	10

R ⋈$_{A<D}$ S

A	R.B	R.C	S.B	S.C	D
1	2	3	2	5	6
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

R ⋈$_{A<D \text{ AND } R.B \neq S.B}$ S

A	R.B	R.C	S.B	S.C	D
1	2	3	7	8	10

3.6 Outer Join

Natürlicher join:

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⋈

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1

Left outer join:

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⋈

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	NULL	NULL

Right outer join:

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⋈

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
NULL	NULL	c_3	d_2	e_2

Full outer join:

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⋈

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	NULL	NULL
NULL	NULL	c_3	d_2	e_2

3.7 Mengenoperator

Vereinigung $\cup$

- $R \cup S$
- Müssen gleiches Schema haben
- Tupel die in R oder S vorkommen ohne Duplikate

Differenz $\setminus$

- R \ S oder R - S
- Müssen gleiches Schema haben
- Tupel die in R aber nicht in S vorkommen

Durchschnitt $\cap$

- R ∩ S
- Müssen gleiches Schema haben
- Tupel die in R und S vorkommen

3.8 Qualifizierung, Namenskonflikte ρ

- $\rho_{S(D,E)}(R(A,B))$
- Aus R(A,B) wird S(D,E)

3.9 Äquivalenzen

$\sigma_\Phi(\sigma_\psi(r)) = \sigma_\psi(\sigma_\Phi(r))$ **Kommutativität**

$\pi_A(\sigma_\Phi(r)) = \sigma_\Phi(\pi_A(r))$ falls Φ nur Attribute aus der Menge A referenziert

r ⋈ s = s ⋈ r (Achtung: Relationenformat ist verschieden!)

---

r ⋈ (s ⋈ t) = (r ⋈ s) ⋈ t **Assoziativität**

$\pi_A(\pi_C(r)) = \pi_A(r)$ falls $A \subseteq C$

$\sigma_\Phi(\sigma_\psi(r)) = \sigma_{\Phi \wedge \psi}(r)$ **Idempotenz**

$\pi_A(r \cup s) = \pi_A(r) \cup \pi_A(s)$

$\sigma_\Phi(r \cup s) = \sigma_\Phi(r) \cup \sigma_\Phi(s)$ **Distributivität**

$\sigma_\Phi(r \bowtie s) = \sigma_\Phi(r) \bowtie s$ falls Φ nur Attribute von r referenziert

$\pi_{A,B}(r \bowtie s) = \pi_A(r) \bowtie \pi_B(s)$ falls für die Joinattribute J gilt: $J \subseteq A \cap B$

r ⋈ (s ∪ t) = (r ⋈ s) ∪ (r ⋈ t)

3.10 Aggregat-Funktionen

3.10.1 Summe Σ

- R(A,B,X) und S(A)
- $r = \{< a1, b1, 2 >, < a1, b2, 3 >, < a2, b1, 4 >\}$
- $\Sigma_X(r) = \{< 9 >\}$
- Gruppieren: $\Sigma_{S,X}(r) = \{< a1, 5 >, < a2, 4 >\}$

3.10.2 Aggregats-Operator F

- F_{COUNT}: Anzahl Tupel
- F_{MAX}: Grösster Wert des betrachteten Attributs
- F_{MIN}: Kleinster Wert des betrachteten Attributs
- F_{SUM}: Summe des betrachteten Attributs
- F_{AVG}: Durchschnitt des betrachteten Attributs

4 Entity-Relationship Design

4.1 3-Phasen

- Konzeptionelle Datenmodell: weitgehend technologieunabhängige Spezifikation der Daten
- Logische Datenmodell: Übersetzung des konzeptionellen Schemas in Strukturen, die mit einem konkreten DBMS implementiert werden können
- Physische Datenmodell: Anpassungen, die nötig sind um eine befriedigende Leistung im Betrieb zu erreichen (Datenverteilung, Performanz, Sicherheit, ...)

4.2 Konzeptioneller Entwurf

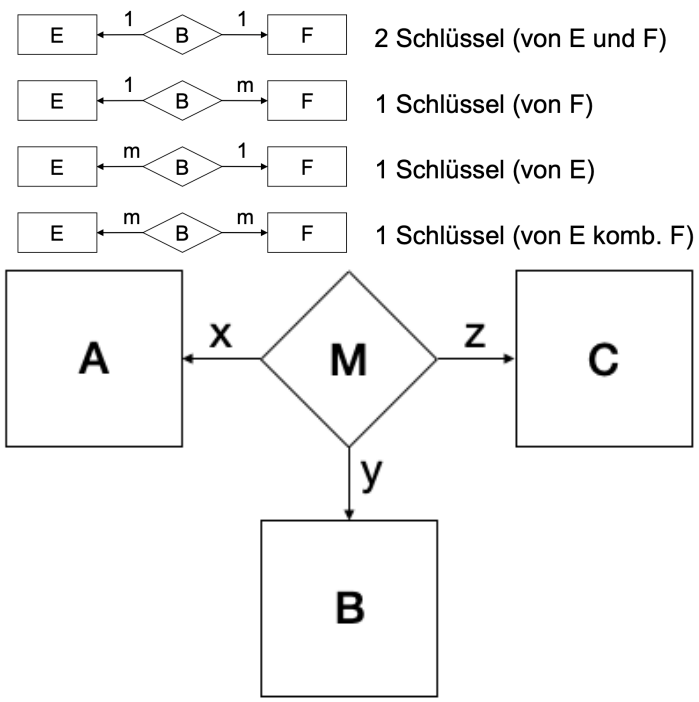
- Klassifikation: Identifikation von gleichen oder ähnlichen Eigenschaften
- Aggregation: Zusammenfassen von diesen Eigenschaften
- Generalisierung und Spezialisierung: Verallgemeinerung (Student → Person)

4.3 Entität

- Konkretes oder abstraktes Objekt, welches eindeutig identifiziert werden kann
- Ein **Entitätstyp** steht für Mengen von gleichartigen Entitäten
- Darstellung durch Rechteck mit eindeutigem Namen
- Ein Entitätstyp wird später in eine Tabelle mit Schlüsseln abgebildet, die Entitäten werden die Zeilen der Tabelle sein
- Entitätstypen haben Eigenschaften (Attribute)
- Entitäten haben Attributwerte
- Attribute werden als Ovale dargestellt
- Unterstrichene Attribute sind Primärschlüssel
- Jeder unabhängige Entitätstyp erhält einen oder mehrere Schlüssel
- Falls der Entitätstyp eingehende Pfeile hat, wird ein Primärschlüssel bestimmt

4.4 Beziehungstyp

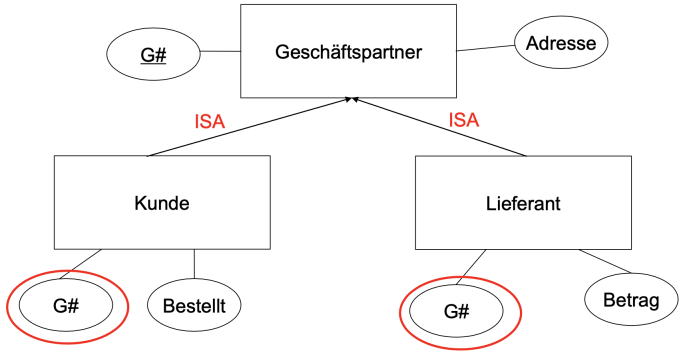
- Wird durch einen Rhombus dargestellt
- Erbt die Primärschlüsselattribute der Entitätstypen von denen er abhängig ist
- m bedeutet beliebig viele, auch 0



x	y	z	Schlüssel
1	1	1	{A,B} und {A,C} und {B,C}
1	1	m	{A,C} und {B,C}
1	m	1	{A,B} und {B,C}
1	m	m	{B,C}
m	1	1	{A,B} und {A,C}
m	1	m	{A,C}
m	m	1	{A,B}
m	m	m	{A,B,C}

4.5 ISA-abhängiger Entitätstyp

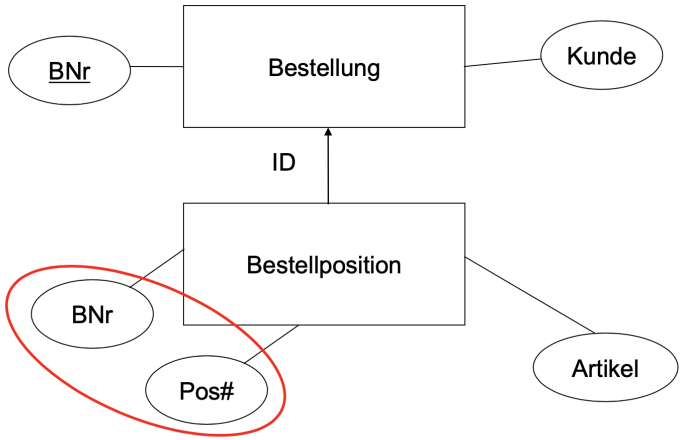
- Ein ISA-abhängiger Entitätstyp ist eine Untergruppe eines anderen Entitätstyps
- 1:1 Relation



4.6 ID-abhängiger Entitätstyp

- Ein ID-abhängiger Entitätstyp ist eine Untergruppe eines anderen Entitätstyps

- Komplexe Attribute führen zu ID-abhängigen Entitätstypen
- 1:M Relation



4.7 Zusammengesetzter Entitätstyp

- Wird verwendet, wenn an Beziehungstypen Entitätstypen angehängt werden wollen
- Es wird ein Rechteck um den Beziehungstyp gezeichnet

4.8 Anomalien

4.8.1 Update-Anomalien

- Sachverhalt in der Realität ändert sich
- Mehrere Änderungen in einer Relation sind nötig

4.8.2 Delete-Anomalien

- Sachverhalt in der Realität ändert sich
- Information in einer Relation verschwindet

4.8.3 Insert-Anomalien

- Sachverhalt der Realität möchte abgebildet werden
- Information kann nicht erfasst werden

4.8.4 Normalisieren

- Bei der Normalisierung werden Relationen aufgeteilt
- Ist ein mathematischer Prozess

5 SQL

5.1 Mängel

- Mangelnde Performanz (in den siebziger Jahren) führte zu Kompromissen
- Verzicht auf eine rein mengenmässige Verarbeitung
- SQL lässt Duplikate zu
- Die Behandlung von NULL
- Trotz Standardisierungsbemühungen ist eine Vielzahl von Dialekten entstanden
- Die Sprache enthält sehr viel Redundanz

5.2 ER-Schema zu Relationenformat

- Jeder Entitäts- und jeder Beziehungstyp ergibt ein Relationenformat, unabhängige Entitätstypen zuerst
- Alle Fremdschlüssel-Attribute müssen im Relationenformat aufgeführt werden
- Primärschlüssel-Attribute werden auch bei der Dokumentation des Relationenformats unterstrichen
- Es ist empfehlenswert jeder Tabelle einen Primärschlüssel zuzuweisen

5.3 Relationenformat zu Tabellen

- Jedes Relationenformat wird zu einer Datenbank-Tabelle
- Die Tupel in einer Tabelle sind **nicht** geordnet
- ¡A, B¿ und ¡B, A¿ sind verschiedene Schlüssel
- Unique-Schlüssel ist nötig
- Es ist empfehlenswert, immer einen Primärschlüssel zu definieren

5.4 Data Definition Language DDL

- Erzeugen, Ändern, Löschen von Datenbankobjekten
- CREATE, ALTER, DROP,...

5.4.1 CREATE TABLE

- Es kann höchstens einen PK pro Tabelle geben
- Es sind mehrere Unique-Klauseln pro Tabelle möglich
- Es ist abzuraten, andere Attribute als den PK als FK zu referenzieren
- Anhand der Referenz sichert das DBMS bei Dateneingabe, aber auch bei Löschen von Tabellen, die referentielle Integrität
- Trigger: ON DELETE, ON UPDATE wenn ein Tupel in der referenzierten Tabelle gelöscht, geändert oder eingefügt wird
- Trigger: CASCADE Werte des Fremdschlüssels werden bei Ändern des PK-Werts der referenzierten Tabelle automatisch angepasst

```
CREATE TABLE Produkt
(
  ProduktNr    char(4)       NOT NULL,
  Bezeichnung varchar(100) NOT NULL,

  CONSTRAINT PK_Produkt PRIMARY KEY(ProduktNr)
  -- auch möglich: PRIMARY KEY(ProduktNr)
);
CREATE TABLE Verkauf
(
  PNr          char(4)        NOT NULL,
  Kundennummer integer        NOT NULL,
  Menge        decimal(10,2) NOT NULL,
  CONSTRAINT FK_Produkt FOREIGN KEY(PNr)
  REFERENCES Produkt(ProduktNr)
);
```

5.4.2 ALTER TABLE

- Wenn eine Datenbank sauber implementiert und richtig genutz wird, kann sie im laufenden Betrieb erweitert werden
- ALTER TABLE Verkauf ADD Datum date NOT NULL;
- ALTER TABLE Verkauf ADD CONSTRAINT FK_Kunde FOREIGN KEY (KundenNummer) REFERENCES Kunde(KNr);

5.4.3 DROP TABLE

- Löschen von Tabellen
- DROP TABLE Verkauf;

5.5 Data Manipulation Language DML

- Datenändern (Einfügen,Ändern,Löschen)
- INSERT, UPDATE, DELETE, ...

5.5.1 INSERT

- Anzahl und Datentypen müssen zueinander passen
- Die Attributnamen können weggelassen werden, ist aber schlechter Stil
- Bei fehlenden Attributwerten wird der Default-Wert (oder NULL) eingetragen
- Bei unzulässigen Daten wird nichts eingefügt
- INSERT INTO Student (SNo,SName,Adresse) VALUES ('87-604-1','Meier','Basel');
- Es kann auch das Resultat einer Abfrage eingefügt werden

- INSERT INTO Employees (EmpFirstName, EmpLastName) SELECT Customers.CustFirstName, Customers.CustLastName FROM Customers WHERE Customers.CustomerID IN (1,4,9);

5.5.2 UPDATE

- UPDATE wird in der Regel mit einer Selektion verbunden
- UPDATE Student SET Adresse = 'Zürich' WHERE SNo = '87-604-1';
- Es können mehrere Attributwerte in einer Anweisung gleichzeitig geändert werden
- Wenn die WHERE-Klausel weggelassen wird, werden ALLE Tupel geändert
- Bei unzulässigen Daten wird nichts geändert
- Für die neuen Attributwerte können auch Berechnungen (*, /, -, +) mit bereits bestehenden Attributwerten gemacht werden
- UPDATE Belegt SET Note = Note + 0.5 WHERE SNo = '87-604-1';

5.5.3 DELETE

- DELETE löscht immer ganze Tupel
- DELETE wird in der Regel mit einer Selektion verbunden
- Ohne Search Condition/WHERE-Klausel wird die ganze Tabelle geleert
- DELETE FROM Salaer WHERE Betrag > 100000;
- Bei Verletzung von Schlüsselbedingungen oder wenn die WHERE-Klausel keine Treffer ergibt, wird nichts gelöscht

5.6 Data Query Language DQL

- Daten lesen (Anfragen an Datenbank stellen)
- SELECT – FROM – WHERE
- Bei Duplikaten erhält man einen relationalen Bag

5.6.1 SELECT

- Entspricht der Projektion
- Reihenfolge: FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY
- DISTINCT elimieniert Duplikate, da aber eine Sortierung nötig ist (n*log(n)) nur anwenden, wenn nötig
- Suchprädikate werden mit einer dreiwertigen Logik ausgewertet (false, true, unknown)
- Bereichsvariable: SELECT * FROM (SELECT Name FROM Person) AS x WHERE x.Name = 'Müller';

LIKE

- '_' steht für ein einzelnes, beliebiges Zeichen
- '%' steht für eine beliebige Anzahl Zeichen
- SELECT name FROM Mitarbeiter NATURAL JOIN Person WHERE name LIKE 'M__er';
- Besucher mit gleichem Vornamen: SELECT * FROM Besucher AS x WHERE EXISTS (SELECT 1 FROM Besucher AS y WHERE x.Vorname = y.Vorname AND x.Name <> y.Name);

Mengen

- Concatenation: UNION ALL
- Durchschnitt: INTERSECT ALL
- Differenz: EXCEPT ALL
- SELECT * FROM Besucher1 EXCEPT ALL SELECT * FROM Besucher2;

Aggregatfunktionen

- COUNT, MAX, MIN, SUM, AVG
- SELECT COUNT (DISTINCT TITEL) FROM CD;
- Aggregatfunktionen ohne Gruppierung liefern eine Tabelle mit einem Tupel und einem Attribut
- SELECT kdNr, COUNT(*) AS AnzahlBestellungen FROM Kaufhistorie GROUP BY kdNr;
- Liefert eine Tabelle mit einem Tupel pro kdNr, d.h. ein Tupel pro Kunde
- HAVING schliesst ganze Gruppen aus
- ... HAVING SUM(menge * preis)1 ¿ 500;
- ... WHERE Name IN ('Meier', 'Müller', 'Sonderegger');

Join

- Natural join: A $\bowtie$ B $\rightarrow$ SELECT ... FROM A NATURAL JOIN B WHERE ...;
- Theta join: A $\bowtie_P$ B $\rightarrow$ SELECT ...FROM A JOIN B ON P – Joinprädikat WHERE ... – Selektionsprädikat;
- Kreuzprodukt: A $\times$ B $\rightarrow$ SELECT ... FROM A, B WHERE ...;

Null

- Ein Wert ist unbekannt, nicht anwendbar, nicht definiert, die leere Menge oder existiert nicht
- Null ist kein Wert
- WHERE-Klauseln liefern in SQL nur Tupel deren Selektionsprädikat auf wahr ausgewertet wird
- SELECT ... FROM ... WHERE A = 5 OR A IS NULL;

a und b			
a \ b	f	u	w
f	f	f	f
u	f	u	u
w	f	u	w

a oder b			
a \ b	f	u	w
f	f	u	w
u	u	u	w
w	w	w	w

nicht a	
a	¬ a
f	w
u	u
w	f

5.7 Data Control Language DCL

- Rechtevergabe, Datensicherung, ...
- GRANT, REVOKE, ...

	Relationenmodell	SQL D	SQL E
Struktur	Relation	Tabelle	table
	Attribut	Spalte, Kolonne	column
	Tupel	Zeile	row
Auswahl	Relationenalgebra Ausdruck	SELECT Anweisung	SELECT statement
	Projektion	SELECT Klausel	SELECT clause
	Selektion	WHERE Klausel	WHERE clause

5.8 Syntaxregeln

- SQL ist keine Mengensprache, sondern eine Sprache für den Umgang mit relationalen Bags
- In SQL spielt Gross-/Kleinschreibung nur innerhalb von Text-Konstanten eine Rolle
- Konvention: Schlüsselworte gross schreiben (z.B. CREATE TABLE)
- Konvention: Schemaelemente klein schreiben (ausser am Wortanfang)
- Ein Name (TableName, TableAlias, ColumnName, ColumnAlias, ...) muss mit einem Buchstaben beginnen, gefolgt von Buchstaben, Ziffern oder der Unterlänge (underscore) _
- Jede Anweisung ist mit einem ; (Semikolon) abzuschliessen.

6 Technische Aspekte

- SQL ist eine deskriptive Sprache
- Bei SQL wurden Eigenschaften für eine prozedurale Sprache hinzugefügt
- Stored procedures werden in der Datenbank abgelegt und ausgeführt