
Kapitel 1

Einführung

Folien zum Datenbankpraktikum
Wintersemester 2012/13 LMU München

© 2008 Thomas Bernecker, Tobias Emrich
unter Verwendung der Folien des Datenbankpraktikums aus dem Wintersemester 2007/08 von Dr. Matthias Schubert

Welche DBMS gibt es?

Marktführer (Marktanteile 2011 der gängigsten Systeme)

Quelle: Market Share: All Software Markets, Worldwide 2011

- Oracle: 48,2%
- DB2 (IBM): 20,2%
- MS SQL (Microsoft): 17,0%
- SAP/Sybase: 4,6%

Open Source, z.B.

- MySQL
- PostgreSQL
- SQLite

Übersicht

- 1.1 Datenbank-Architektur im Praktikum
- 1.2 Datenbank-Design
- 1.3 ER-Modell und relationales Modell
- 1.4 Datendefinition in SQL
- 1.5 Data Dictionary

Oracle: Objekt-Relationales Datenbanksystem (Version 10g)

Client-Server-Architektur

- DBMS läuft auf einem dedizierten Serverrechner (flores.dbs.ifi.lmu.de)
- Anwendungsprogramme laufen auf Client-Rechner (Linux, Solaris, Windows)
- Kommunikation über Netzwerk für den Benutzer transparent
- Viele Anwender greifen gleichzeitig auf die Datenbank zu (Transaktionsschutz)

Zugriff auf die Datenbank über ...

- Anfragesprache SQL
- Prozedurale Erweiterung PL/SQL (Oracle-spezifisch)
- SQL in Hostsprachen (z.B. C oder Java)

Übersicht

1.1 Datenbank-Architektur im Praktikum

1.2 Datenbank-Design

1.3 ER-Modell und relationales Modell

1.4 Datendefinition in SQL

1.5 Data Dictionary

Datenbanksystem (DBS): System zur Beschreibung, Speicherung und Wiedergewinnung umfangreicher Datenmengen, die von verschiedenen Anwendungsprogrammen benutzt werden.

Ein DBS besteht aus:

- Datenbank: Sammlung aller gespeicherten Daten
 - Es existiert ein Ziel des Entwurfs und eine Vorstellung über die Nutzung.
 - Es existieren logische Zusammenhänge zwischen den Daten in einer DB.

- Datenbank-Management-System
 - Programmsystem, das die DB verwaltet, fortschreibt und Zugriffe darauf regelt

Abstraktionsebenen eines Datenbanksystems

Schichtenmodell: Jede Schicht kennt nur die darunterliegende

Externe Sicht

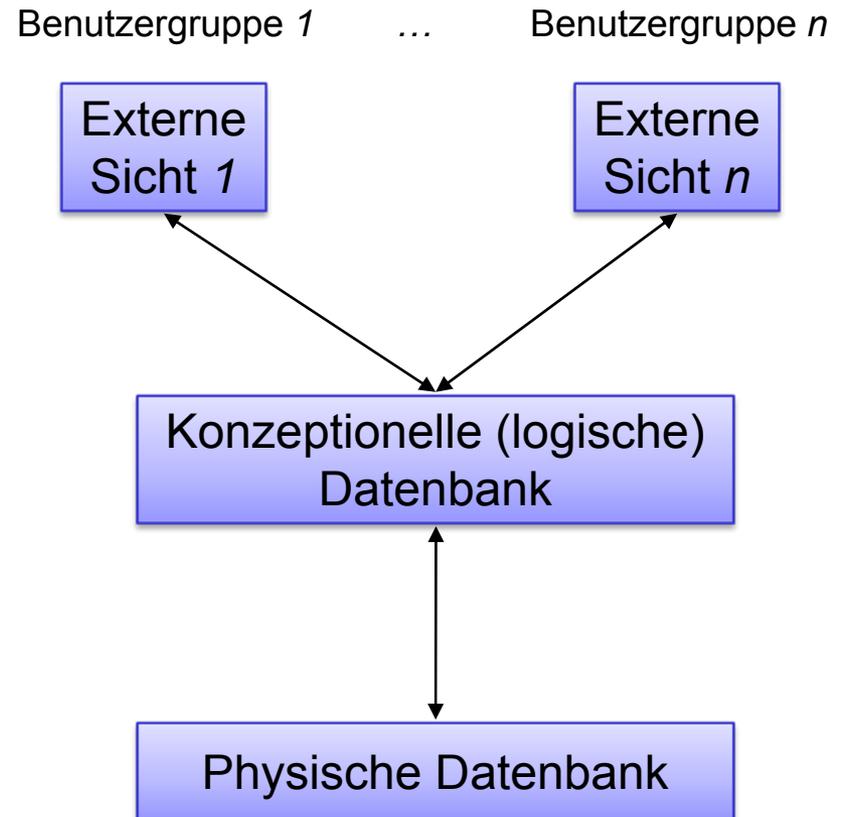
- Views
- Ausschnitte aus dem konzeptionellen Schema

Konzeptionelle Sicht

- einheitliche Darstellung aller Daten
- DBS bietet dazu Datenmodell mit entsprechendem Schema (definiert über Data Definition Language (DDL))

Interne Sicht

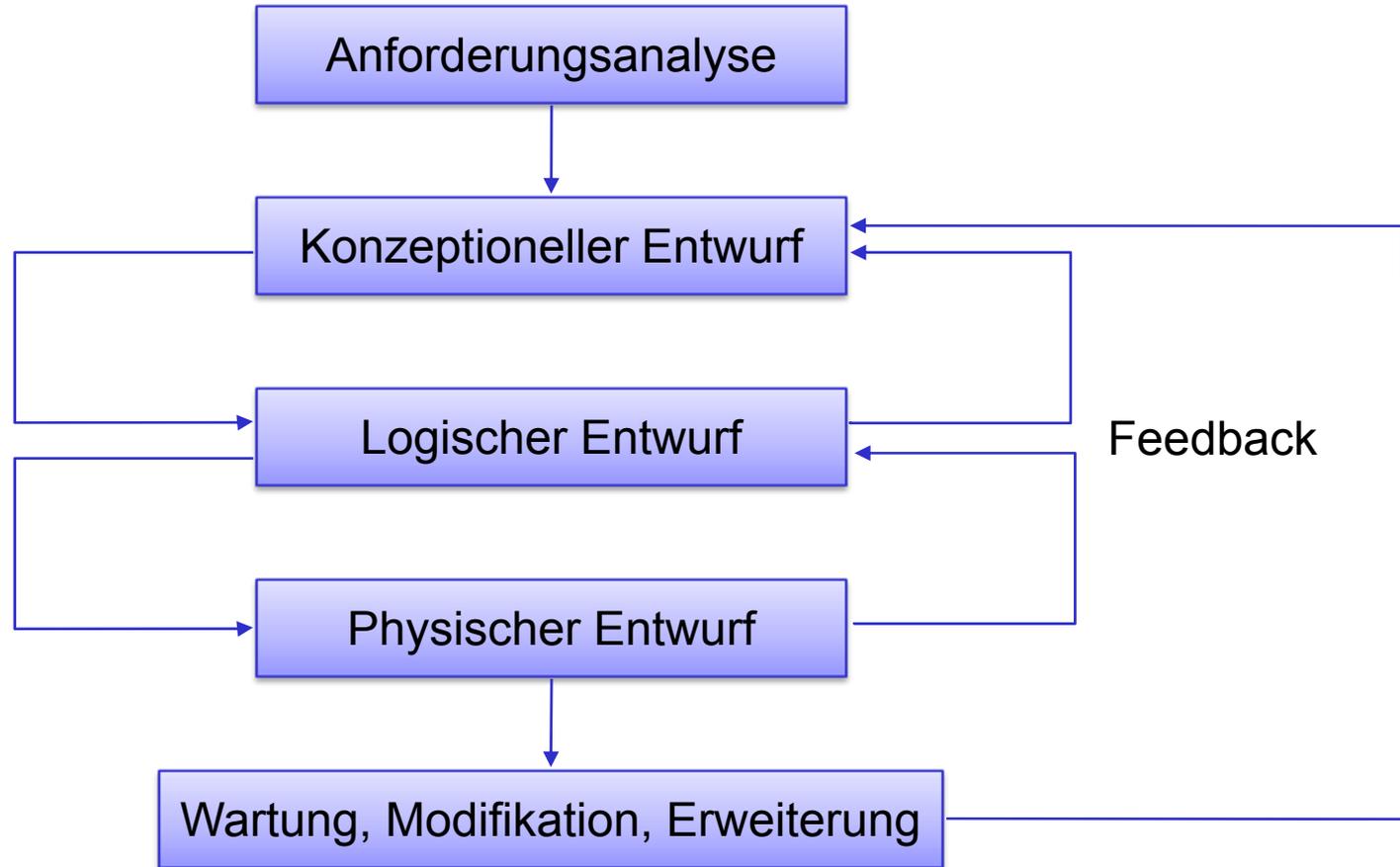
- Implementierung der konzep. DB
- Dateien, Zugriffspfade, etc.



Ansätze für das Datenbank-Design

- *Daten-orientiert*: Die Entwicklung der Datenbank orientiert sich an den Daten, die im Fokus der Anwendungen stehen.
- *Funktions-orientiert*: Die Entwicklung der Datenbank orientiert sich an den Funktionen (Anwendungen), die durch das Datenbanksystem unterstützt werden sollen.
- *Objekt-orientiert*: Die Entwicklung der Datenbank orientiert sich an der Menge der interagierenden Objekte, die in dem zu modellierenden Ausschnitt der realen Welt auftreten. Objekte vereinen datenspezifische Eigenschaften und funktionsspezifisches Verhalten.

Entstehungszyklus



Anforderungsanalyse

- Analyse und Spezifikation von
 - Daten
 - Datenbeziehungen
 - Transaktionen (Funktionen auf den Daten, die von den Applikationen benötigt werden)

- Randbedingungen
 - Leistungsanforderungen (Antwortzeiten etc.)
 - Sicherheit
 - HW/SW-Plattformen
 - Anwendungsrichtlinien

- Beispielansätze
 - Der DB-Experte interviewt die zukünftigen Anwender und versucht deren Anforderungen festzustellen.
 - Der DB-Experte und der zukünftige Anwender entwickeln das Design als Team.

Konzeptioneller Entwurf

- Basierend auf der Anforderungsanalyse werden die essentiellen Konzepte extrahiert und in einem konzeptionellen Datenmodell (DB-Schema) beschrieben.
- Modell (z.B. Entity-Relationship, UML)
 - Formale Beschreibung der Datenobjekte und ihrer Beziehungen
 - Integration verschiedener Sichten
- Beschreibung der Transaktionen (Eingabe, Änderung, Suche, ...)
- Festlegung von Integritätsbedingungen für die Daten und für die Transaktionen

Logischer Entwurf

- Ausgehend vom DB-Schema (z.B. ER-Modell) wird ein DBS-spezifisches Datenmodell (z.B. Relationales Datenmodell) entwickelt.
- Wichtige Schritte bei der Transformation des DB-Schemas in das Datenmodell:
 - Normalisierung und Denormalisierung
 - Primärschlüssel festlegen
 - Formulierung von Integritätsbedingungen und Transaktionen in DBS-spezifischem Datenmodell (relationale Anfragen, rel. Algebra etc.)
 - View-Definitionen (externe Sichten)
 - Zugriffsrechte

Physischer Entwurf (Umsetzung des Datenmodells)

- konkrete Domäne für Attribute, z.B. CHAR(30), VARCHAR(20), NUMBER(4), NUMBER(7 , 2), DATE, ...
- Erzeugen der Relationen und evtl. Laden mit vorhandenen Daten
- Einrichten von Views, Benutzern, Zugriffsrechten
- Eingabe der Integritätsbedingungen (Anfragen, Prüfprogramme, ...)
- Definition geeigneter Indexe

Wartung, Modifikation und Erweiterung

- Beseitigung von Fehlern
- Erstellen von Anwendungsprogrammen
- Re-Engineering, Integration
- Erweiterung

Übersicht

- 1.1 Datenbank-Architektur im Praktikum
- 1.2 Datenbank-Design
- 1.3 ER-Modell und relationales Modell**
- 1.4 Datendefinition in SQL
- 1.5 Data Dictionary

Das ER-Modell beschreibt einen Ausschnitt der realen Welt durch:



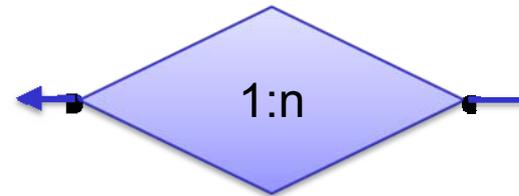
Set	Menge gleichartiger Objekte	Beziehungs- möglichkeit	Merkmal, Eigenschaft
Instanz	bestimmtes Objekt	Konkrete Beziehung zwischen zwei Objekten	Eigenschaft eines Objekts

Kardinalität von Relationships:

1:1-Beziehung (“one-to-one”):



1:n-Beziehung (“one-to-many”):

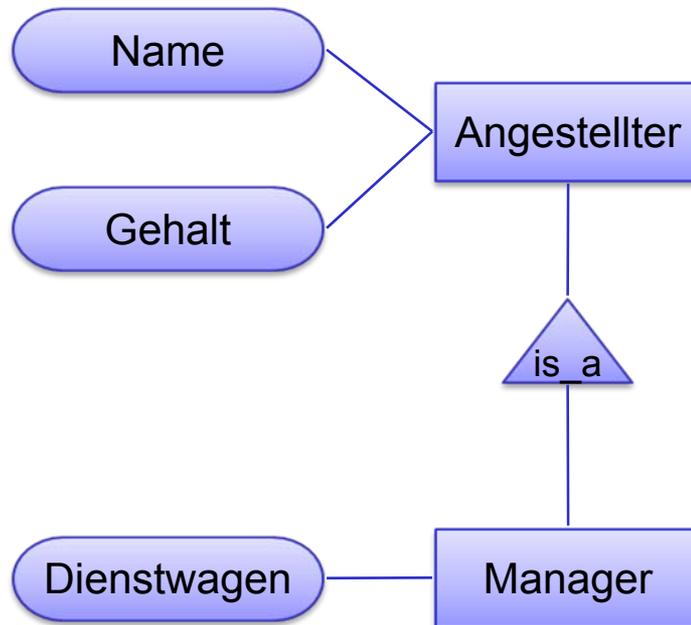


n:m-Beziehung (“many-to-many”):



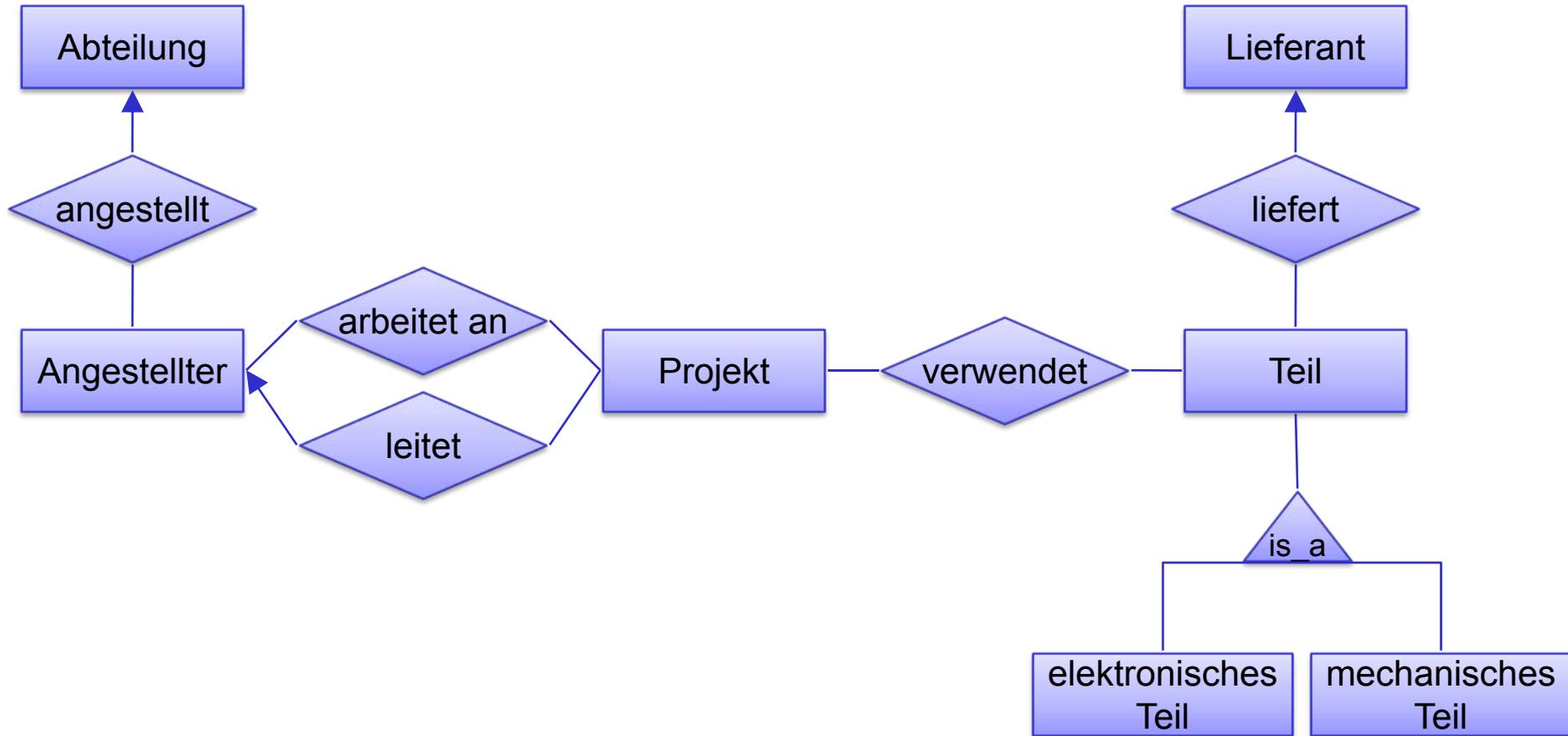
Erweiterung: ISA-Beziehungen

Beispiel:



Semantik dieser ISA-Beziehung:
Der Manager erbt alle Attribute
Und Beziehungen vom Angestellten,
auch den Schlüssel.

Beispiel:



Modellierungsentscheidungen

1. Attribut oder Entity?
2. Attribut zu welchem Entity?
3. Schlüsselattribute?
4. Entity oder Relationship?
5. Relationships nur zwischen zwei Entities?
6. Attribut oder Relationship?
7. Einsatz von ISA-Beziehungen?

1. Attribut oder Entity?



Kriterien:

- Ist das Attribut ein atomarer Wert oder aus mehreren Werten zusammengesetzt?
- Kann der Wert des Attributs fehlen?
- Kann das Attribut mehrere Werte gleichzeitig annehmen?
- Ist die Domäne des Attributs wichtig (bzgl. der Korrektheit des Wertes)?

2. Attribut zu welcher Entity?



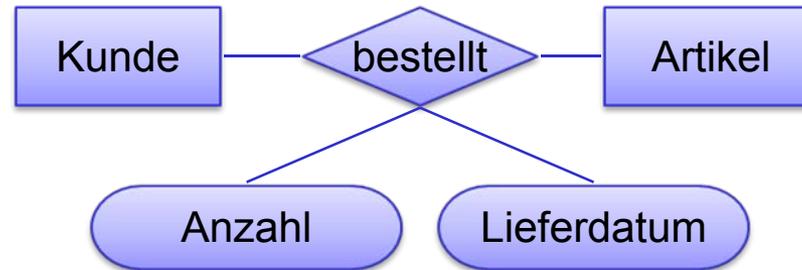
Kriterien:

- Natürlichkeit
- Änderungsfreundlichkeit

3. Schlüssel-Attribute

- Dürfen auch mehrere sein (zusammengesetzter Schlüssel)
- Künstliche IDs (*Surrogate*) nur dann einsetzen, wenn es keinen konstanten Schlüssel innerhalb des Entity-Typs gibt oder der Schlüssel aus sehr vielen Attributen zusammengesetzt wäre

4. Entity oder Relationship?



ODER

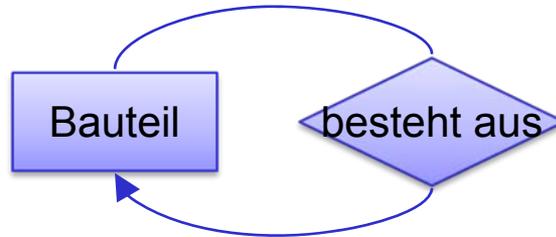


Kriterien:

- Beziehung benötigt Schlüsselattribute → modelliere Entity
- Beziehung hat viele Attribute → modelliere Entity

5. Relationships nur zwischen zwei Entities?

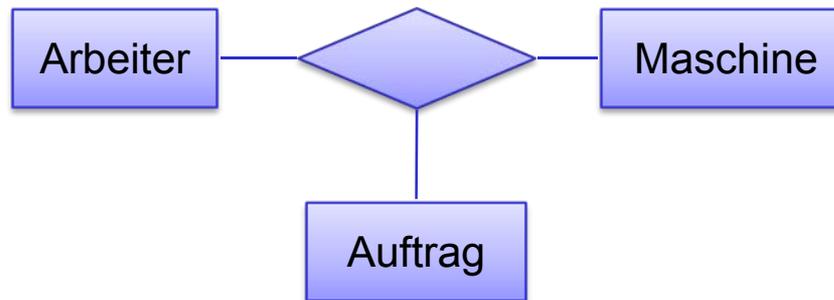
unär:



binär:

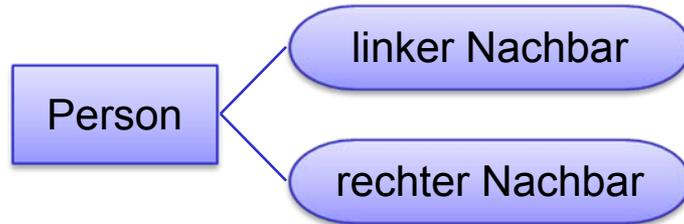


ternär:

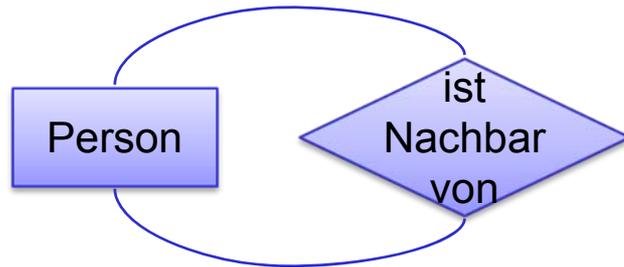


(selten, meist
schlechte
Modellierung)

6. Attribut oder Relationship?



Schlechte Modellierung im ER-Modell:
Fremdschlüssel vermeiden

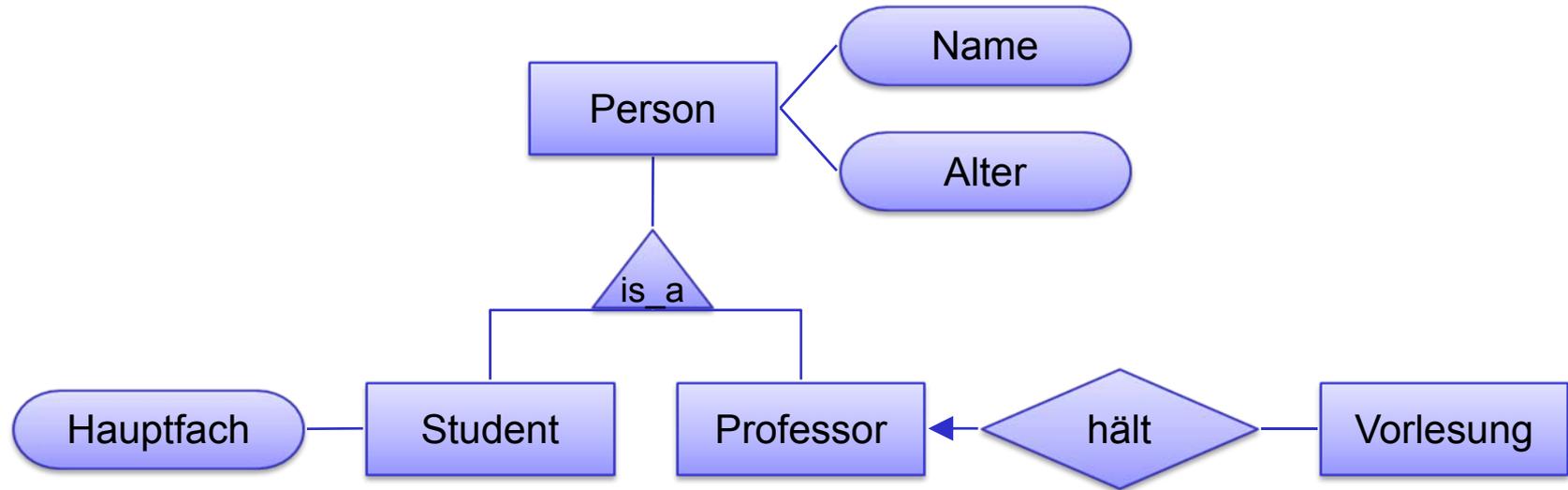


besser als Relationship modelliert



Aus anderen Attributen berechenbar
(hier: Nachbarschaft über
Hausnummer):
keine Relationship verwenden

7. Einsatz von ISA-Beziehungen?



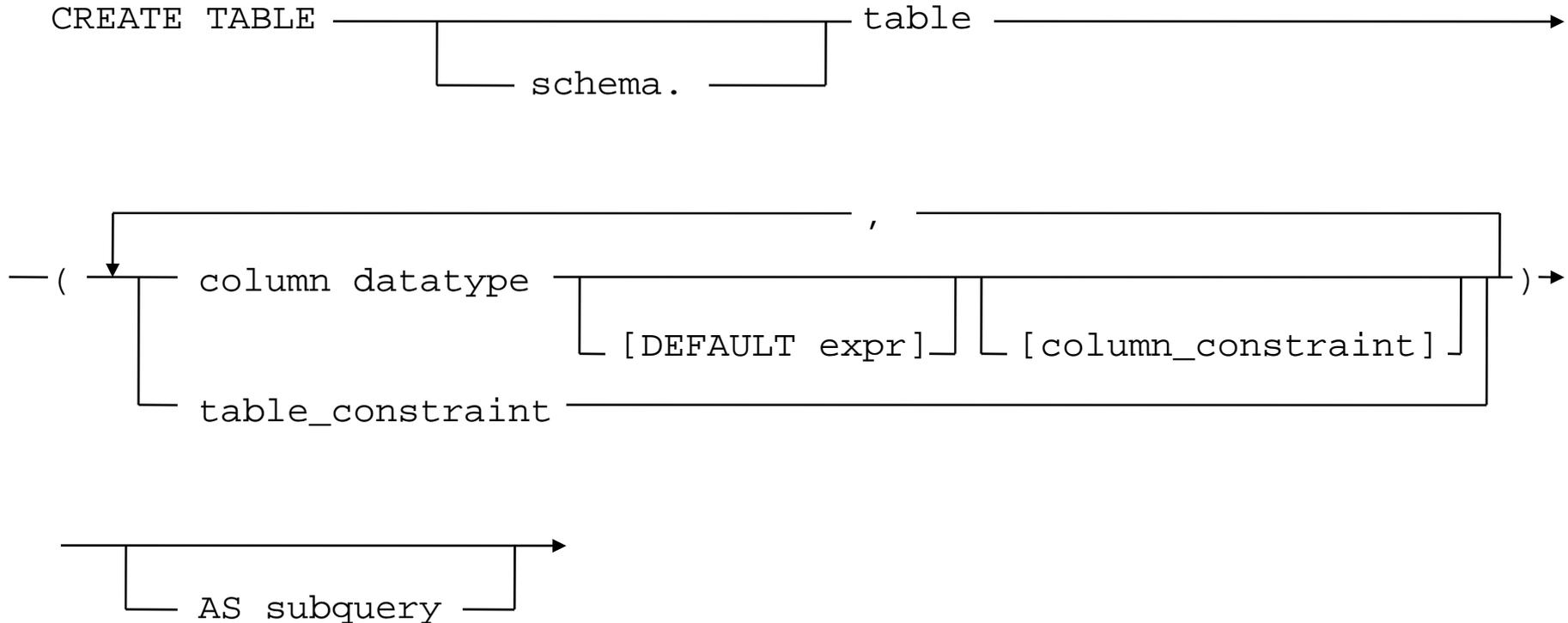
Kriterien:

- Hat die Oberklasse gemeinsame Beziehungen / Attribute?
- Haben die Unterklassen unterschiedliche Beziehungen / Attribute?

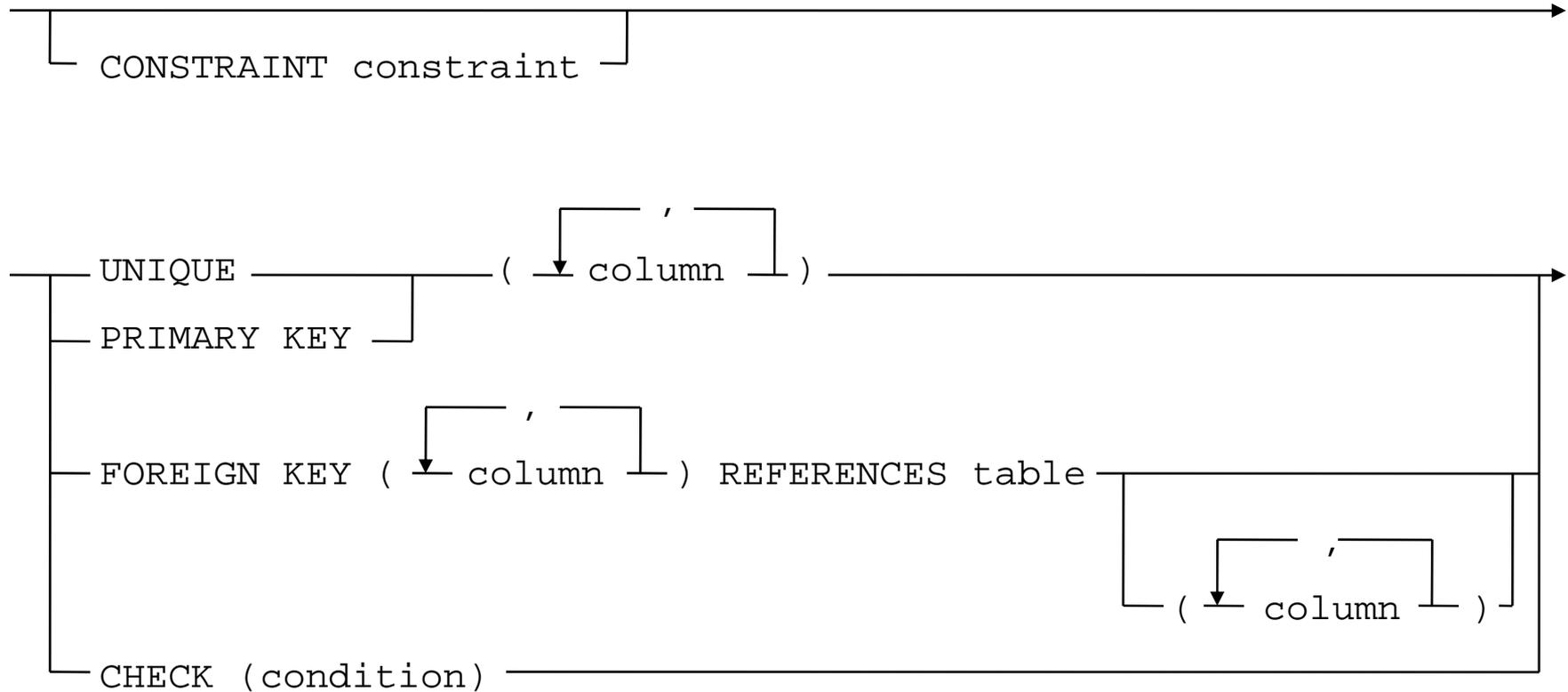
Übersicht

- 1.1 Datenbank-Architektur im Praktikum
- 1.2 Datenbank-Design
- 1.3 ER-Modell und relationales Modell
- 1.4 Datendefinition in SQL**
- 1.5 Data Dictionary

Der Befehl CREATE TABLE (teilweise) in Oracle SQL



```
table_constraint ::=
```



Datentypen in ORACLE (auszugsweise)

Data Type	Description
CHAR(size)	Fixed length character data of length size. Maximum size is 255.
VARCHAR2(size)	Variable length character string having maximum length size bytes. Maximum size is 4000.
NUMBER(p, s)	Number having precision p and scale s.
LONG	Character data of variable length up to 2 gigabytes.
DATE	Valid dates range from January 1, 4712 BC to December 31, 4712 AD.
RAW(size)	Raw binary data of length size bytes. Maximum size is 2000.
LONG RAW	Raw binary data of variable length up to 2 gigabytes.
BLOB	A binary large object. Maximum size is 4 gigabytes.

Relationales Modell (Wiederholung)

- Einziger Grundbaustein: Relation (Tabelle)
 - wird beschrieben durch Relationenschema: $R(A_1 : D_1, \dots, A_k : D_k)$
 - **Attribut A_i** : Spalte, Name eindeutig
 - **Domäne D_i** : Wertebereich, Datentyp des Attributs
 - **Tupel**: Zeile, Element einer Relation
- Relationales Datenbankschema: Menge aller Relationenschemata
- Relationale Datenbank:
Menge aller Relationen = Relationales DB-Schema + Ausprägung
- Relationen sind Mengen
 - keine Duplikate
 - Reihenfolge der Tupel belanglos

Transformation ER in Relationales Modell

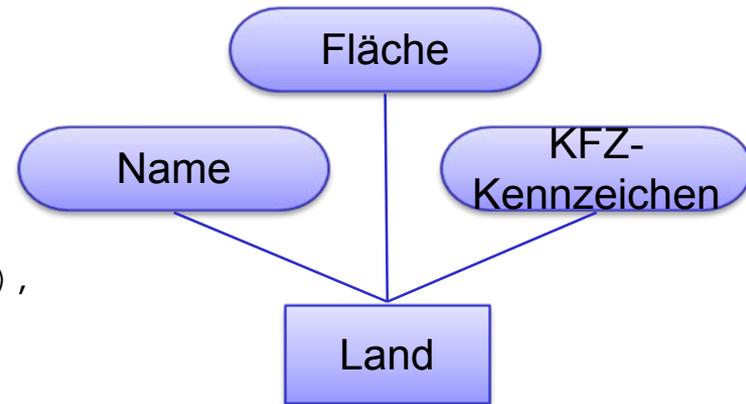
- Bezeichnungen umwandeln
 - **ER**: freie Bezeichnungen und Beschreibung
 - **SQL**: feste Notation (Namensgebung) z.B. bezügl. Sonderzeichen, Schlüsselwörter

- Datentypen für Attribute festlegen
 - **ER**: frei
 - **SQL**: atomare Datentypen (Oracle-spezifisch)
CHAR, VARCHAR2, NUMBER, LONG, DATE, RAW, LONGRAW, ...

Die Auswahl des Datentyps ist abhängig von den Operationen, die ausgeführt werden sollen.

Transformation von Entities

```
CREATE TABLE land (  
    name varchar2(25) NOT NULL,  
    flaeche number(10,2),  
    kfz_kennz char(4) NOT NULL,  
    CONSTRAINT pk_name PRIMARY KEY (name),  
    UNIQUE (kfz_kennz)  
);
```



Transformation von Relationships (n:m)

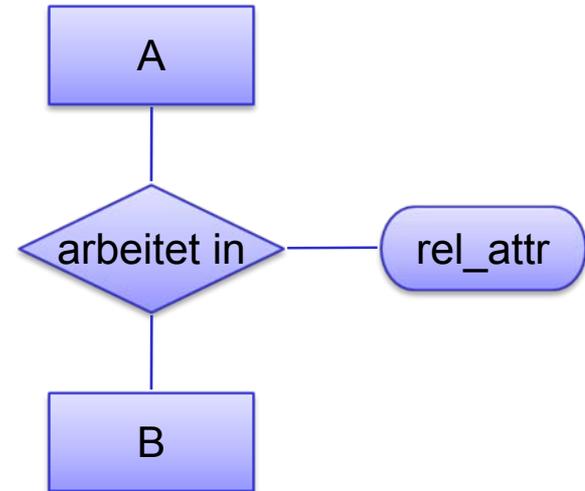
Entity A und B haben wir schon überführt:

```
CREATE TABLE A ( ... PRIMARY KEY a_id ...)
```

```
CREATE TABLE B ( ... PRIMARY KEY b_id ...)
```

Hinzu kommt:

```
CREATE TABLE rel (  
  a_id ... NOT NULL,  
  b_id ... NOT NULL,  
  rel_attr ... ,  
  PRIMARY KEY (a_id, b_id),  
  FOREIGN KEY (a_id) REFERENCES A,  
  FOREIGN KEY (b_id) REFERENCES B  
);
```



Transformation von Relationships (1:n, 1:1)

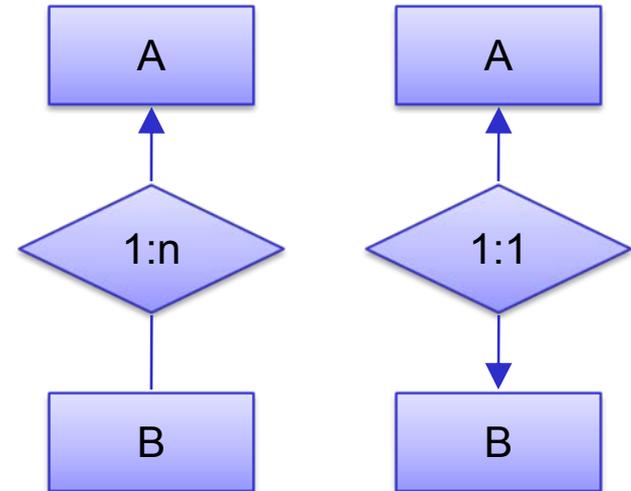
Keine separate Relation für die Beziehung,

sondern:

1:n → Aufnahme des Primärschlüssels von A in
die Entity-Relation B

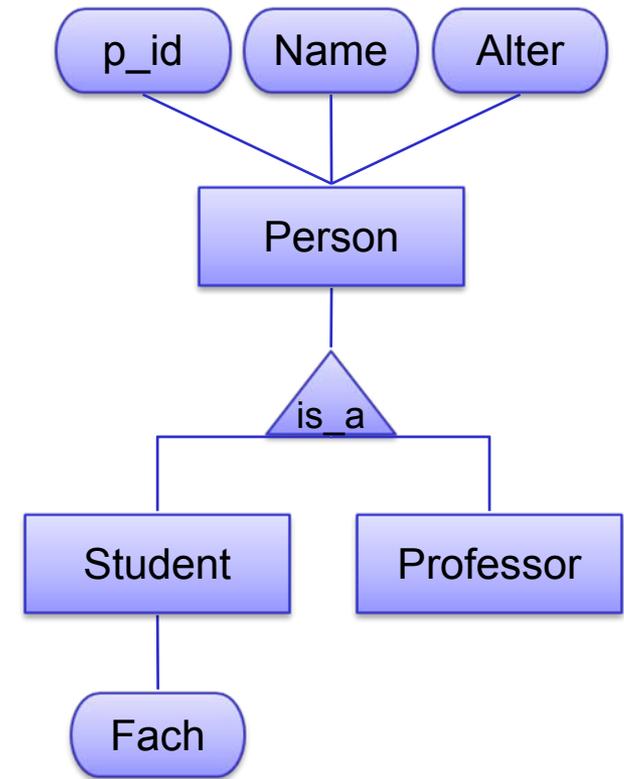
1:1 → Aufnahme des Primärschlüssels von B in
die Entity-Relation A oder umgekehrt

```
CREATE TABLE B (  
    b_id ... NOT NULL,  
    b_rest ... ,  
    a_id ... NOT NULL,  
    PRIMARY KEY (b_id),  
    FOREIGN KEY (a_id) REFERENCES A  
);
```



Transformation von ISA-Beziehung

```
CREATE TABLE person (  
    p_id char(8) NOT NULL,  
    name ... ,  
    alter ... ,  
    PRIMARY KEY (p_id) );  
  
CREATE TABLE student (  
    p_id char(8) NOT NULL,  
    fach ... ,  
    PRIMARY KEY (p_id),  
    FOREIGN KEY (p_id) REFERENCES person )  
  
CREATE TABLE professor (  
    p_id char(8) NOT NULL,  
    ... ,  
    PRIMARY KEY (p_id),  
    FOREIGN KEY (p_id) REFERENCES person );
```



Alternative:

```
CREATE TABLE person_x (  
    p_id char(8) NOT NULL,  
    name ... ,  
    alter ... ,  
    PRIMARY KEY (p_id)  
)
```

```
CREATE TABLE student (  
    p_id char(8) NOT NULL,  
    name ... ,  
    alter ... ,  
    fach ... ,  
    PRIMARY KEY (p_id)  
)
```

```
CREATE TABLE professor ( ... );
```

```
CREATE VIEW person (p_id, name,  
    alter) AS (  
    (SELECT p_id, name, alter FROM  
    person_x)  
    UNION  
    (SELECT p_id, name, alter FROM  
    student)  
    UNION  
    (SELECT p_id, name, alter FROM  
    professor)  
);
```

Normalisierung des erzeugten Relationalen Schemas durch Herstellung der 3. Normalform (z.B. mit dem Synthesealgorithmus aus dem DBS-Skript).

Ein Relationenschema ist in der 3. Normalform (3NF), wenn für alle Abhängigkeiten $X \rightarrow A$ mit $X \subseteq R$, $A \in R$, $A \notin X$ gilt:

- X enthält einen Schlüssel von R , oder
- A ist prim.

Seien A und B Attributmengen des Relationenschemas R ($A, B \subseteq R$). B ist von A *funktional abhängig* oder A *bestimmt B funktional*, geschrieben $A \rightarrow B$, gdw. zu jedem Wert in A genau ein Wert in B gehört:

$$A \rightarrow B \Leftrightarrow t_1, t_2 \in r(R): t_1[A]=t_2[A] \Rightarrow t_1[B]=t_2[B]$$

für alle real möglichen Relationen $r(R)$. Dabei bezeichnet $r(R)$ eine Relation r über dem Schema R .

Ein Attribut heißt *prim*, wenn es Teil eines Schlüsselkandidaten ist, sonst *nicht prim*.

Übersicht

- 1.1 Datenbank-Architektur im Praktikum
- 1.2 Datenbank-Design
- 1.3 ER-Modell und relationales Modell
- 1.4 Datendefinition in SQL
- 1.5 Data Dictionary**

Neben den Daten **in** den Relationen benötigt ein Datenbanksystem Informationen **über** die Relationen (und über andere Objekte), sog. “Meta-Daten”. Die Meta-Daten werden im “Data Dictionary” oder “System-Katalog” gespeichert.

- Relationenverwaltung

- Relationennamen
- Attributnamen für jede Relation
- Domäne der Attribute
- Viewnamen und Viewdefinitionen
- Integritätsbedingungen

- Benutzerverwaltung (Datenschutz)

- Namen berechtigter (autorisierter) Benutzer
- Speicherbereich und Speicherobergrenze für jeden Benutzer
- Zugriffs- und andere Rechte einzelner Benutzer

○ Anfragebearbeitung

- Anzahl der Tupel in jeder Relation
- Indexnamen
- Indizierte Relationen
- Indizierte Attribute
- Art des Index

Häufig: Data Dictionary ist selbst Teil der Datenbank. Meta-Information ist in Systemrelationen gespeichert. Systemrelationen sind wie alle anderen Relationen zugänglich.

○ Verwaltete Objekte:

- Relation
- View
- Synonym
- Link
- Sequenz
- Schnappschuss
- PL/SQL-Programme
- Benutzer

Namensgebung für ORACLE-Systemrelationen

	USER_	ALL_	DBA_
TABLES	alle Tabellen, die der Benutzer angelegt hat	alle Tabellen, auf die der Benutzer Zugriff hat	alle Tabellen des gesamten Systems
TAB_COLUMNS	alle Spalten derjenigen Tabellen, die der Benutzer angelegt hat	alle Spalten derjenigen Tabellen, auf die der Benutzer Zugriff hat	alle Spalten aller Tabellen des gesamten Systems
INDEXES	alle Indexe, die der Benutzer angelegt hat	alle Indexe, die über Tabellen erstellt wurden, auf die der Benutzer Zugriff hat	alle Indexe des gesamten Systems
VIEWS	alle Views, die der Benutzer angelegt hat	alle Views, auf die der Benutzer Zugriff hat	alle Views des gesamten Systems
TAB_PRIVS	Zugriffsrechte auf alle Tabellen, die der Benutzer angelegt hat	Zugriffsrechte auf alle Tabellen, auf die der Benutzer Zugriff hat	Zugriffsrechte auf alle Tabellen des gesamten Systems

Beispielanfragen an das Data Dictionary

```
SQL> describe dictionary;
```

Name	Null?	Type
-----	-----	-----
TABLE_NAME		VARCHAR2(30)
COMMENTS		VARCHAR2(2000)

```
SQL> select count(*) from dictionary;
```

```
COUNT(*)
```

```
-----
```

```
157
```

```
SQL> select TABLE_NAME, COMMENTS from dictionary where TABLE_NAME like  
      '%TABLE%';
```

TABLE_NAME	COMMENTS
-----	-----
ALL_TABLES	Description of tables accessible to the user
DBA_TABLES	Description of all tables in the database
DBA_TABLESPACES	Description of all tablespaces
.....	

```
SQL> select TABLE_NAME, OWNER from ALL_TABLES;
```

TABLE_NAME	OWNER
-----	-----
ACCESS\$	SYS
AUD\$	SYS
....	