

# ***Perl Database Interface (DBI)***

Thorsten Huber, Hendrik Brückner

Berufsakademie Stuttgart

# Agenda

- Perl – Eine kurze Einführung
- Was ist Perl DBI? – Grundlagen
  - Konzept & Architektur
- Beispiel: Abfrage mit Perl DBI
- DBI Programmierung
  - Connection
  - Query
  - Transaction
- DBI in der Praxis
- Praktische Übung

# *Practical **E**xtraction and **R**eporting **L**anguage*

## *Introduction*

# Perl – Die Scriptsprache

## Sprache:

- Kontextsensitive Interpretersprache
- Objekt-Orientiert (seit Perl  $\geq 5$ )
- Plattformunabhängig

## Anwendung:

- **Textmanipulation**
- System Administration
- Web Development
- Network Programming

# Perl – Variablen und Datentyp

Perl besitzt 3 bedeutende Datentypen:

- **Scalar: Zahlen und Zeichenfolgen**

```
$var = "skalärer_Inhalt";
```

- **List (Array): Liste von Scalars**

```
@list = ( 123, 'ele1', 'ele2' );
```

- **Hash: Struktur von Key-/Value-Pairs**

```
%myHash = ( 'key1' => val1, 'key2' => 123 );
```

## Perl – Variablen und Kontext

Wert der Variable abhängig vom Kontext der Umgebung

Mögliche Kontexte sind:

- Void Context
- Scalar Context
- List Context
- Hash Context, Boolean Context, ...

## Perl – Reguläre Ausdrücke

### Textverarbeitung in Perl – Reguläre Ausdrücke

#### Beispiel: Umwandlung von Klein- zu Großbuchstaben

```
$text =~ tr/a-z/A-Z/;
```

#### Beispiel: CVS Revision

```
$version = do { my @r = (q$Revision: 1.6 $ =~ /\d+/g); sprintf "%d"."%02d" x $#r, @r };
```

#### Beispiel: CGI URL Decoding

```
$value =~ s/%([\dA-Fa-f][\dA-Fa-f])/pack( "C", hex( $1 ) )/eg;
```

# *Perl DataBase Interface*

## *Introduction*

# Was ist Perl DBI?

- Perl **DataBase Interface**
- Datenbank-unabhängige Programmierschnittstelle
- Perl Modul DBI
- Sammlung von Database Drivers (DBD)

## Vorteile von Perl DBI

- *Unabhängigkeit*
  - Perl und DBI sind für viele Plattformen verfügbar
  - Unterstützung einer Vielzahl von Datenbanken
- *Perl's Stärke in der Textanalyse*
  - Professionelle Datenanalyse und Datenselektion
  - Data Warehousing, Data Mining, ETL
- *Web Anwendungen*
  - Dynamische Web Anwendungen mit CGI
  - Apache Webserver Integration

# DBI Komponenten

DBI besteht aus zwei Software-Komponenten:

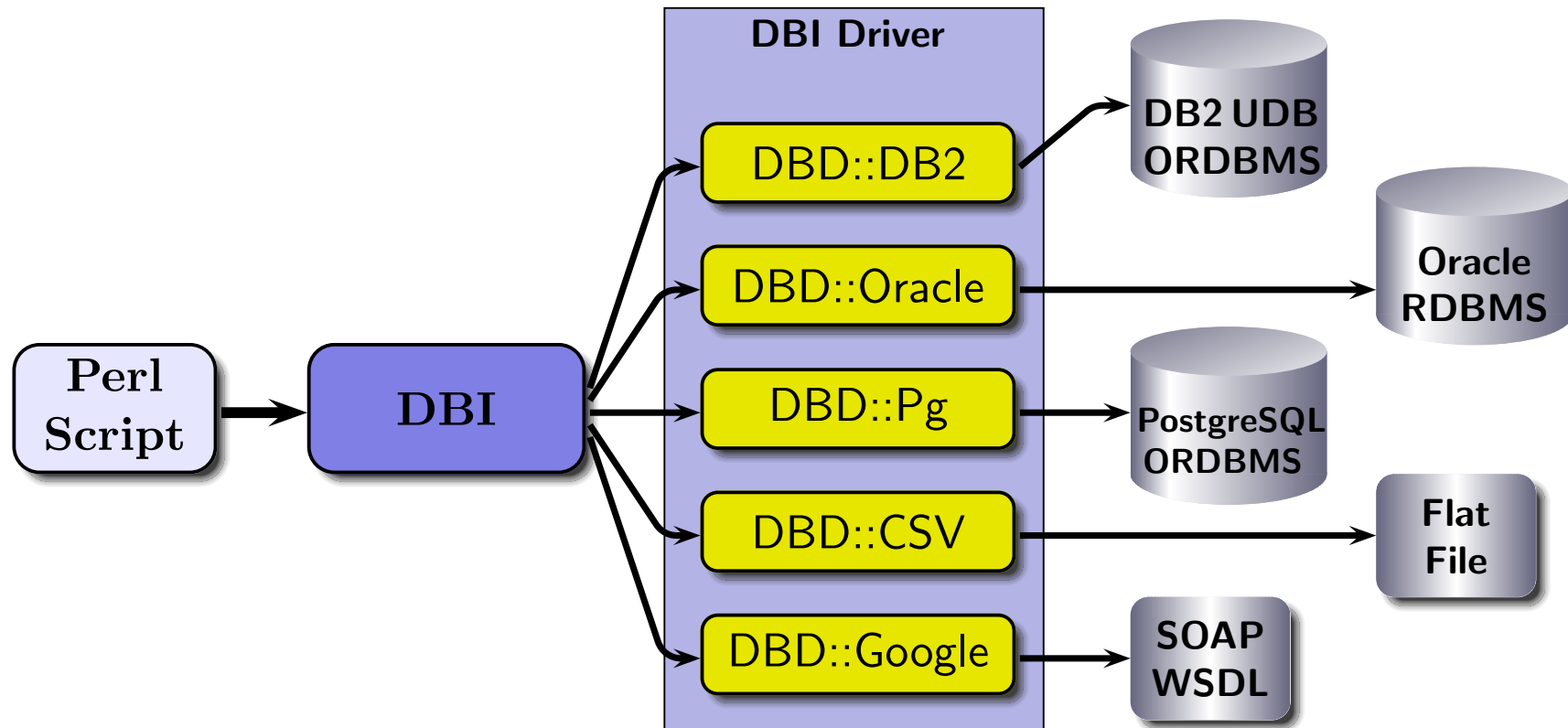
## Database Interface

- Database Independent
- Schnittstelle zwischen Perl Script und Database Driver

## Database Drivers

- Database Dependent
- Kommunikation mit dem Datenbank Managementsystem

# DBI Architektur

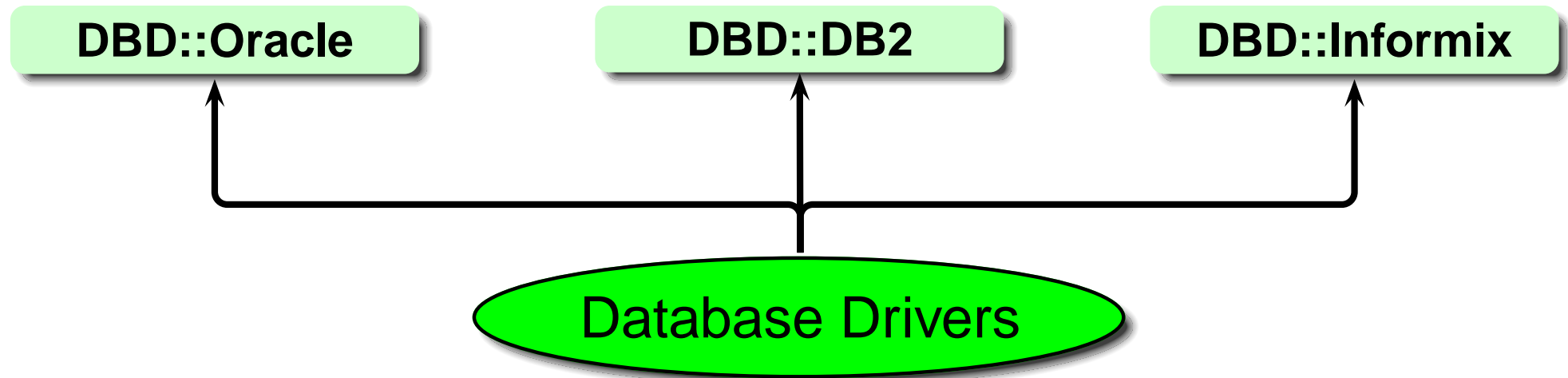


# DBI Database Drivers

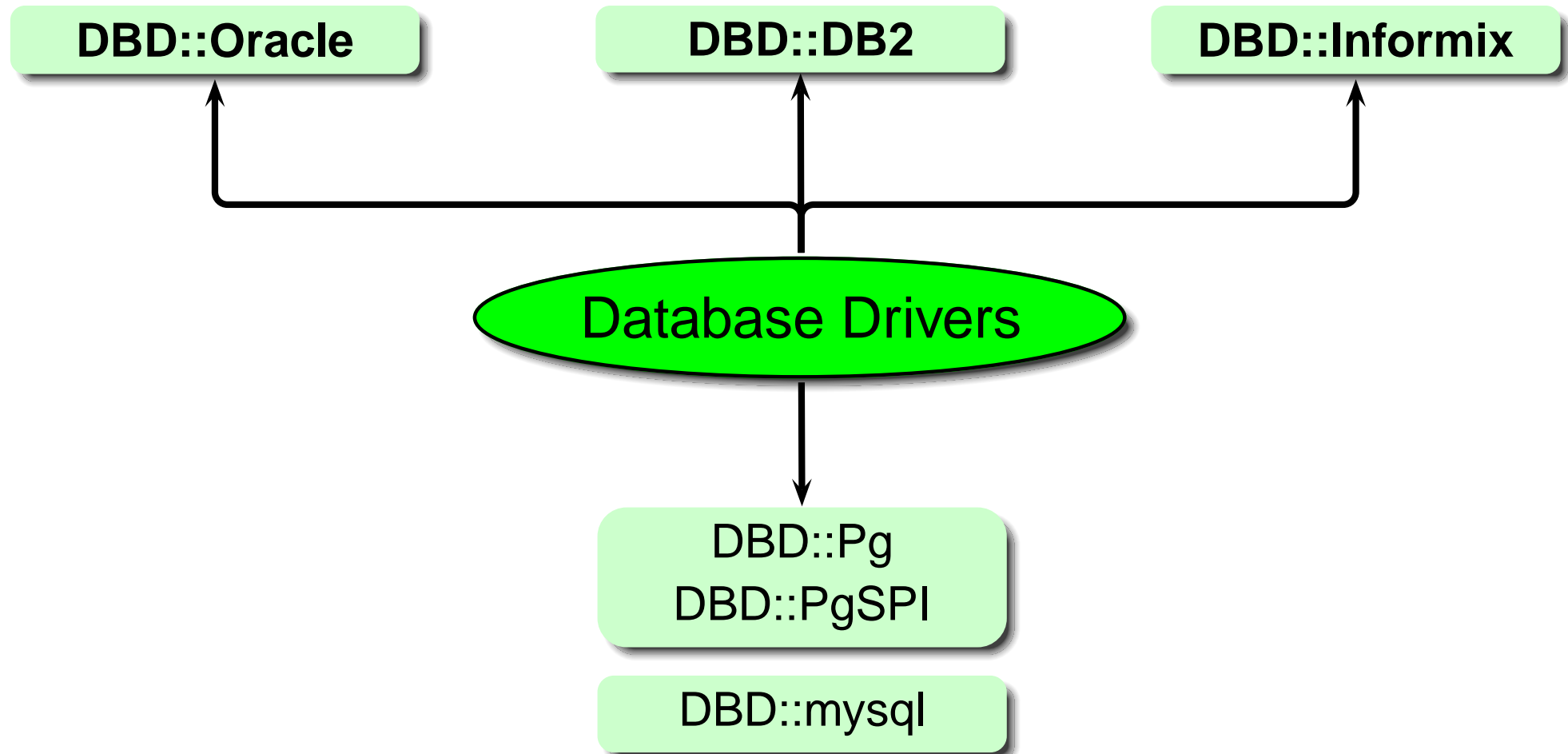


Database Drivers

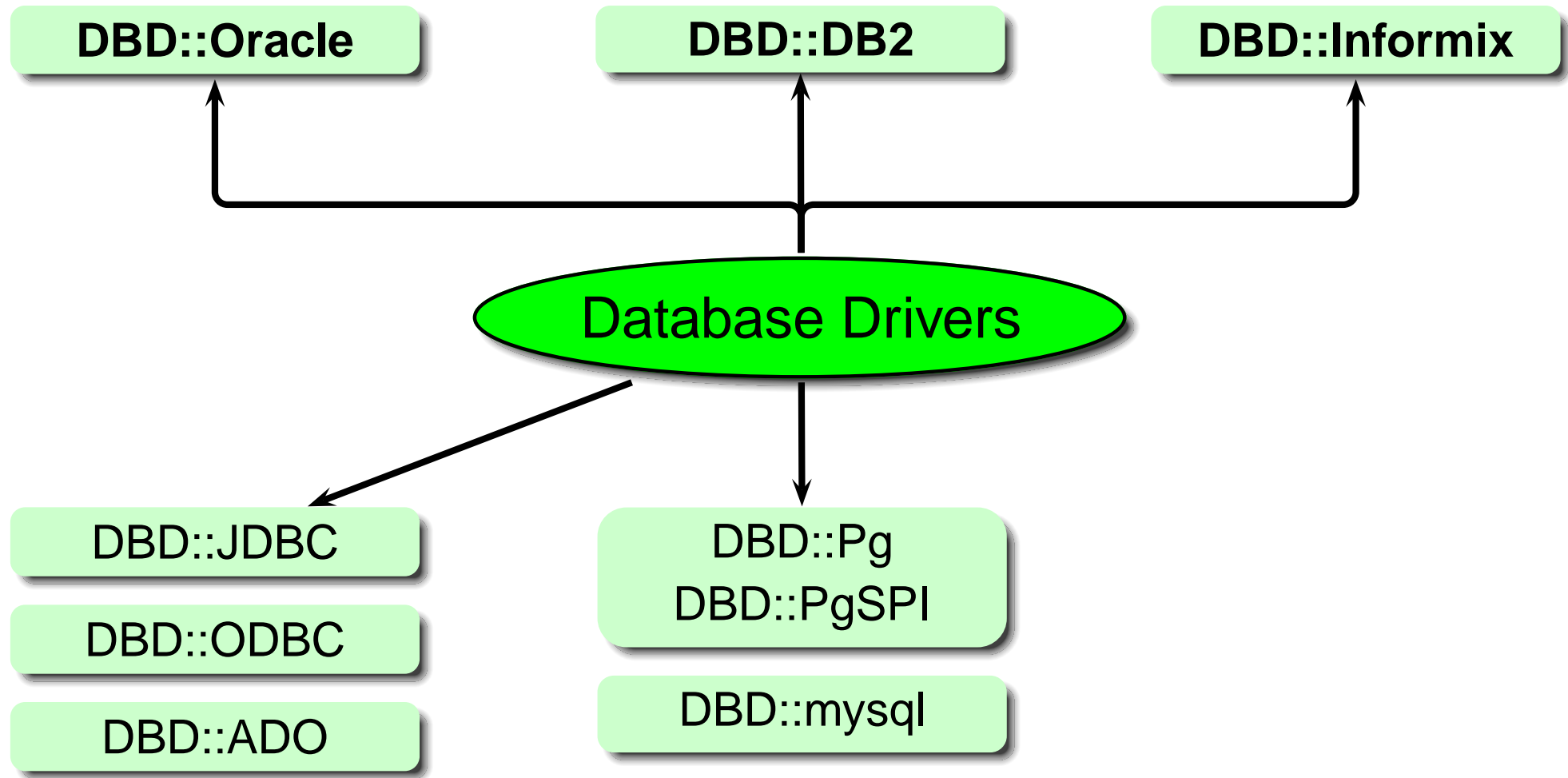
# DBI Database Drivers



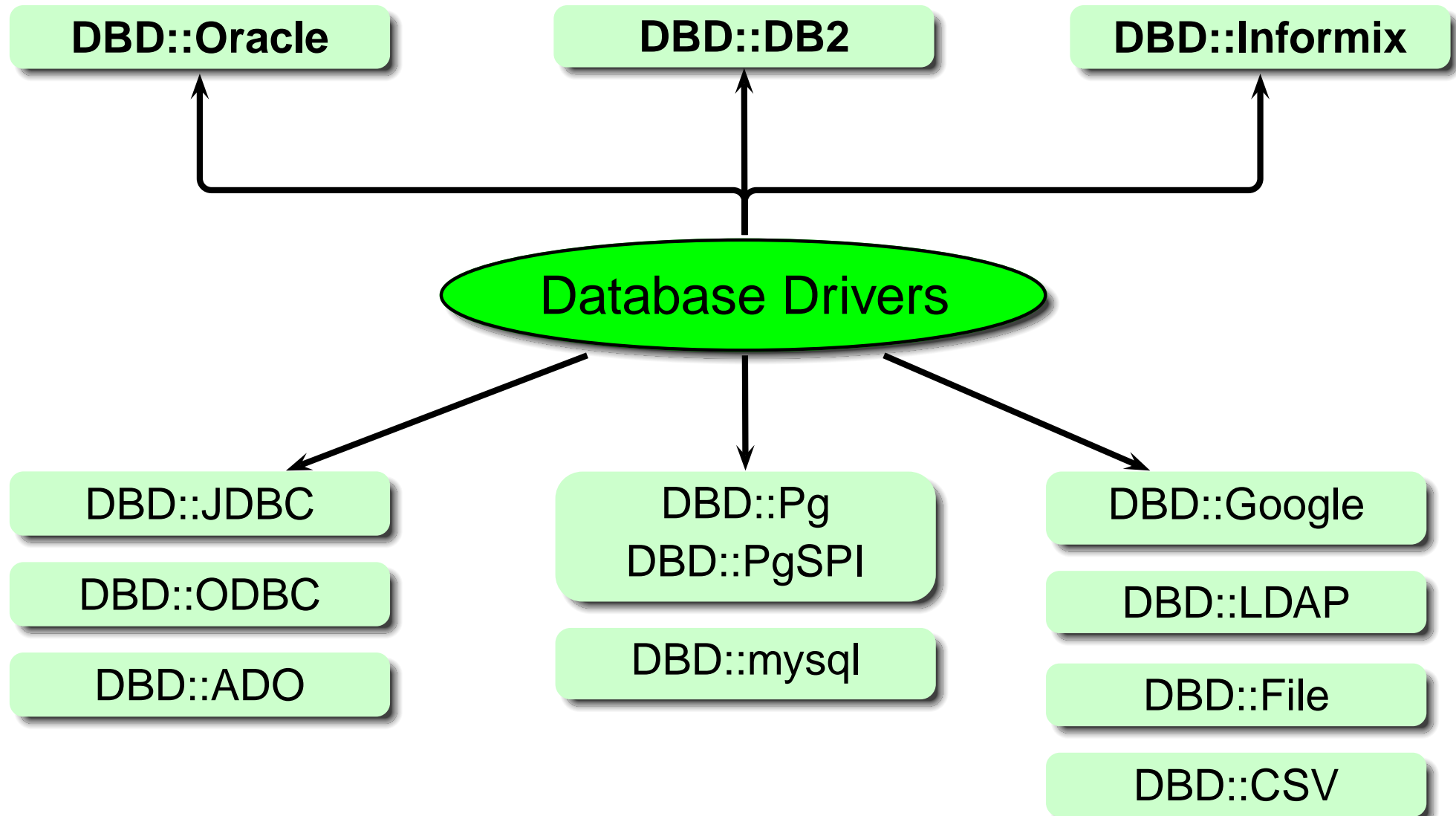
# DBI Database Drivers



# DBI Database Drivers



# DBI Database Drivers



# *Perl DBI*

## *Beispiel Abfrage*

# *Perl DBI Programmierung*

# Handles

## Database Handles

```
$datasource = "DBD:DriverName";  
$dbh = DBI->connect($datasource, ...);
```

## Statement Handles

```
$sth = $dbh->prepare("SELECT _name, _vorname _".  
                    "FROM _tabelle _WHERE _".  
                    "id _= _?");  
$rsh = $sth->execute(2543);
```

# Connection & Disconnection

## Übergabe von Attributen beim Verbindungsaufbau

```
$dbh = DBI->connect(  
    "dbi:Pg:dbname=phonebook;host=database.somedoma.in;port=5432;",  
    "gast" , "gast" ,{ AutoCommit => 0}  
);
```

```
# Datenbankzugriffe
```

```
# ...
```

```
$dbh->disconnect();
```

## Query Data

### Aktion ohne Rückgabewerte:

```
$dbh->do("DELETE FROM person WHERE vorname = 'Ernst'.  
        "AND name = 'Sorgenfrei'");
```

### Prepared Statements mit Variablenbindung:

```
$sth = $dbh->prepare("DELETE FROM person WHERE vorname = ?".  
                    "AND name = ?");  
  
$sth->bind_param(1,"Ernst");  
$sth->bind_param(2,"Sorgenfrei");  
$sth->execute();
```

# Fetching Data

## Auslesen von Rückgabewerten:

```
$sth = $dbh->prepare("SELECT _id_ FROM _person_ where _vorname_ = _?_".  
                    "AND _name_ = _?");  
  
$sth->bind_param(1,"Ernst");  
$sth->bind_param(2,"Sorgenfrei");  
$sth->execute();  
while ( ($id) = $sth->fetchrow_array() ){  
    print "id=$id\n";  
}
```

# Transaktionen in DBI – Voraussetzungen

Folgende Voraussetzungen müssen erfüllt sein:

- Datenbank muss Transaktionen unterstützen
- Deaktivierung des *AutoCommit*-Mechanismus

```
my $dbh = DBI->connect( "dbi:Pg:dbname=phonebook", "username", "password" , {  
    AutoCommit => 0  
});
```

# Transaktionen in DBI – commit & rollback

„Festlegen“ einer Transaktion:

```
$dbh->commit();
```

„Zurückrollen“ einer Transaktion:

```
$dbh->rollback();
```

# *Perl DBI in der Praxis*

# Datenbank Proxy – DBI::Proxy

## Eigenschaften von DBI::Proxy

- zusätzlicher Abstraktionslayer
  - eigenes Sicherheitskonzept
  - schlankes Protokoll
- Zugriff auf Datenbanken ohne Netzwerkfähigkeiten
- Kein Datenbanktreiber auf Client notwendig
  - geringerer Administrationsaufwand
  - evtl. Einsparung von Lizenzkosten
  - Plattformunabhängigkeit

# DBI::Proxy – Architektur

Architektur:

Applikation

DBI Proxy

(Netzwerkverbindung)

DBI ProxyServer

DBI Datenbanktreiber

Datenbanktreiber

(MySQL, PostgreSQL, Oracle, JDBC, ODBC, ...)

Datenbank (lokal o. remote)

# Integration in den Apache WebServer

Integration von Perl in Apache – *mod\_perl*:

- Performance Steigerung
- Precompiled Perl-Scripts

Apache DBI Erweiterungen:

- Authentifizierung & Autorisierung – Apache::AuthDBI
- Apache::DBI

## Apache::DBI – Besonderheiten

DBI in Apache bietet folgende Features:

- Aufbau von Datenbank Verbindungen beim Start von Apache
- Persistente Datenbank Verbindungen
- Konfigurierbares Rollback
- Verifizierung von Datenbank Verbindungen

*Praktische Übung*

*PhoneBook Query*

*DBI Proxy und PostgreSQL*

???

# FRAGEN

???