Relevant DBSII

# Kapitel 1

## Schnittstellen

Eine Datenbankschnittstelle ist eine Programmierschnittstelle , die dem Zugriff auf und den Datenaustausch mit einer Datenbank regelt, d.h. die Kommunikation zwischen den Beiden ermöglicht.

Konflikte, die aus Strukturunterschieden zwischen beiden Systemen entstehen (zwischen imperative Programmiersprache und Relationale Datenbank).
==Abbildung der unterschiedlichen Datenmodelle und Zugriffsparadigmen zwischen Programmiersprache und DBMS → **Impedance Mismatch**

## Aufgaben von Schnittstellen

1. Kapselung der Datenfunktionalität (Verbindung zum DBMS, Zugriff auf konkrete Datenbank, Absetzen von Anfragen)
2. Zugriff auf Ergebnisse (Geeignete Datenstrukturen für mengenwertige Anfrageergebnisse, Zugriff auf imperative Programmiersprache, Zugriff auf Metadaten == Beschreibung von Tabellen)
3. Umgesetzt als Bibliotheken, die auf Treiber und Protokoll zur Kommunikation mit DBMS Server abbilden.

## Beschreibung der wichtigsten CLI-Funktionen in OBDC == 3-4 Bibliotheken/Methoden kennen

1. SQLAllocHandle legt ein Umgebungs-, Verbindungs-, Anweisungs- oder Deskriptorhandle an.

*SQLRETURN SQLAllocHandle(SQLSMALLINT HandleType, SQLHANDLE InputHandle, _Out_ SQLHANDLE *OutputHandle);*

2. SQLSetEnvAttr definiert sogenannte Umgebungsattribute ENVIRONMENT ATTRIBUTES

*SQLRETURN SQLSetEnvAttr(SQLHENV EnvironmentHandle, SQLINTEGER Attribute, SQLPOINTER ValuePtr, SQLINTEGER StringLength);*

3. SQLSetConnectAttr definiert sogennante Verbindungsattribute

*SQLRETURN SQLSetConnectAttr(SQLHDBC ConnectionHandle, SQLINTEGER Attribute, SQLPOINTER ValuePtr, SQLINTEGER StringLength);*

4. SQLConnect baut eine Verbindung zur Datenquelle auf

```
SQLRETURN SQLConnect(SQLHDBC hdbc,
                     SQLCHAR* ServerName, SQLSMALLINT NameLength1,
                     SQLCHAR* UserName, SQLSMALLINT NameLength2,
                     SQLCHAR* Aunthentication, SQLSMALLINT Namelength3);
```

Bestimmte Bibliotheken und Methoden in OBDC – siehe oben
NOCH ZU TUN → fehlerhafter Programmcode in C korrigieren oder die Methoden allgemein in die richtige Reihenfolge bringen

Allgemeine grobe Reihenfolge:

```
retcode = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
retcode = SQLSetEnvAttr(...);
retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
.
.
retcode = SQLConnect(...);
retcode = SQLAllocHandle(...);
.
.
char* selectStmt = "SELECT Kunr, Ort FROM Kunde";
retcode = SQLExecDirect(hstmt, (SQLCHAR *)selectStmt, SQL_NTS);
printf("Performed: %s\n", selectStmt);
retcode = SQLBindCol(hstmt, 1, SQL_C_CHAR, szKunr, 100, &cbKunr); // Bind columns 1
retcode = SQLBindCol(hstmt, 2, SQL_C_CHAR, szOrt, ORT_LEN, &cbOrt); // Bind columns 2
```

for schleife – fetch – print break

```
SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
SQLDisconnect(hdbc);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return 0;  }
```

## Kapitel 2

**Datensicherung** = Gewährleistung der Richtigkeit, Vollständigkeit und logischen Widerspruchsfreiheit der Daten, Gewährleistung der Integrität/Konsistenz der Daten
**Zugriffsschutz** = Schutz der Daten vor unberechtigter Benutzung
**Integritätssicherung** = Unterstützung des Datenbanknutzers bei der Gewährleistung von Konsistenz und Integrität der gespeicherten Daten
**Integrität** bedeutet ganz allgemein inhaltliche Korrektheit/Richtigkeit und Vollständigkeit der Daten = semantische Integrität, operationale Integrität, physische Integrität
**Konsistenz** bedeutet logische Übereinstimmung und innere Widerspruchsfreiheit der Daten

### Gefahrenbereiche ----- die 4 muss man wissen

1. Schadenursache Personal:
   - ➔ Fehlerhafte Bedienung / Versehentliches Löschen von Daten
   - ➔ Fehler bei der Dateneingabe
   - ➔ Unterlassung notwendiger Handlungen (Vergessen des Backups)
2. Schadenursache Hardware:
   - ➔ Ausfall von Hardwarekomponenten (Festplatte)
   - ➔ Störung des Rechners-/Kommunikationsnetzes
3. Schadenursache Software:
   - ➔ Fehlerhafte Anwendungssoftware
   - ➔ Probleme bei Update/Upgrades
   - ➔ Computerviren
4. Schadenursache „Höhere Gewalt":
   - ➔ Feuer / Brand
   - ➔ Wasser / Hochwasser
   - ➔ Immission
   - ➔ Blitzschlag, Stromausfall, Erdbeben

### Maßnahmen zur Begegnung der Gefahren

1. Personelle Maßnahmen = Auswahl des Personals; Sicherung des Zugangs
2. Bauliche Maßnahmen = Trennung des EDV-Bereiches von übrigen Arbeitsbereichen; Auswahl und Sicherung geeigneter Räumlichkeiten (Hochwasserschutz, Brandschutztüren)
3. Technische Maßnahmen = Auswahl zuverlässiger Hardware, Notstromaggregate, elektronische Verriegelung
4. Organisatorische / Softwareseitige Maßnahmen: Programmierte Maßnahmen(Prüfen); Regelung von Zugriffsberechtigung, Anlegen von Sicherheitskopien, File- und Record-Locking, Prüfziffern.

### Was ist eine Transaktion?

Eine Transaktion ist eine Folge an SQL-Anweisungen, die wie eine einzelne Anweisung behandelt werden. Eine Transaktion garantiert, dass Änderungen entweder vollständig oder gar nicht ausgeführt werden.

- Transaktions-Kontrollanweisungen werden vom Programmierer verwendet um sicherzustellen, dass eine Reihe von Anweisungen als Einheit behandelt wird.
- Der Server protokolliert alle Modifizierungen, um die Möglichkeit der Rekonstruktion zu garantieren

- Der Server sperrt Tabellenseiten die während einer Transaktion verändert werden, so dass andere Anwender keinen Zugriff auf „vorübergehende“ Daten haben.

**Sytanx der Transaktions-Kontrollanweisungen**

Beginn :: BEGINN TRANSACTION
Bestätigen :: COMMIT [TRANSACTION]
Zurücksetzen :: ROLLBACK [TRANSACTION]

## ACID PRINZIP – Eigenschaften einer Transaktion

1. Atomarität (atomicity) == Unteilbarkeit (BEGIN bis COMMIT) = alles oder nichts == Der Nutzer weiß stets, in welchem Zustand die Datenbank ist
2. Konsistenzhaltung (consistency) = eine erfolgreich ausgeführte Transaktion erklärt ihre Änderungen für gültig und gewährleistet die Konsistenz der Datenbank.
3. Isolation (isolation) = Zwischenzustände der Daten, die eine Transaktion bearbeitet, sind möglicherweise inkonsistent und dürfen von parallel ablaufenden Transaktionen NICHT benutzt werden. Umgekehrt darf eine Transaktion selbst auch keine Zwischenzustände anderer Transaktionen benutzen.
4. Dauerhaftigkeit (durability) = die Ergebnisse vollständig ausgeführter Transaktionen dürfen nicht durch Fehler verloren gehen

**A**tomicity – **C**onsistency – **I**solation – **D**urability

## Semantische Integritätssicherung

Sicherung der semantischen Integrität: die Gewährleistung der „Richtigkeit“ und „Korrektheit“ der Daten.

Möglichkeiten / Formen zur Sicherung der semantischen Integrität

- ➔ Formatkontrolle: sichert, dass die definierten Datenformate eingehalten werden, z.B. keine Alphazeichen in Integerspalten eingetragen werden können
- ➔ CHECK_Klauseln: anwendungsbedingte Einschränkung der Wertebereiche der Attribute.
- ➔ Domain: Festlegung der Wertebereiche der Attribute. Stellt eine Art Datentyp dar, Beschreibung einer Spalte durch Name, Datentyp, Länge, Standardwert und Integritätsbedingung
- ➔ Nutzerdefiniert Datentypen: es werden neue Datentypen entsprechend den Nutzererfordernissen kreiert (Analogien zu Domainen)
- ➔ Starke Primärschlüssel und UNIQUE-Klausel: Beides dient der Zurückweisung von Duplikaten.

**Referentielle Integrität** = Sicherung logischer Widerspruchsfreiheit inhaltlich in Beziehung stehenden Spalten verschiedener Tabellen.
**Trigger** = daten- und ereignisorientierte Prüfbedingungen und Sicherungsmaßnahmen, die die semantische Integrität auf prozeduralem Wege sichern.

**Spalten- und Tabellenbezogene Integritätsbedingungen**

➔ **Constraint =** Bedingung/Einschränkung , die an eine Spalte und/oder Tabelle gebunden ist.

| Schlüsselwort in SQL | Wirkungsweise | Wirkungskreis | |
|---|---|---|---|
| | | Spalte | Tabelle |
| NOT NULL | Missing Value werden nicht zugelassen | x | |
| UNIQUE | Dublikate werden abgelehnt | x | x |
| PRIMARY KEY | Duplikate werden abgelehnt | x | x |
| CHECK | Sichert Einhaltung d.Prüfbedingungen | x | x |
| REFERENCES | sichert referentielle Integrität | x | x |
| FOREIGN KEY | sichert referentielle Integrität | | x |

➔ **Standardwerte/DEFAULTS** = Definition an der Spalte DEFAULT <Wert>
ALTER TABLE Kunde ADD Bundesland CHAR(20) CONSTRAINT DF_Kunde_Budesland **DEFAULT** ‚Sachsen'

ALTER TABLE Kunde ADD **CONSTRAINT** DF_Kunde_ort DEFAULT ‚Dresden' FOR Ort

➔ **Einschränkungen/CHECK** = Definition an einer Tabelle CHECK (<logischer Ausdruck>)
ALTER TABLE Kunde ADD **CONSTRAINT** CK_Kunde_geschl **CHECK**(Geschl IN (‚M', ‚W'))

## Referentielle Integrität

Das problem der Sicherung des inhaltlichen „Zusammenpassens“ zwischen Eigen- und Fremdschlüsselwert.

### Foreign Key Klausel

```
foreign key (spaltenname) references tabelle
[on delete {restrict | cascade | set null | set default}]
[on update {restrict | cascade | set null | set default}]
```

- Restrict = ein Versuch, eine Zeile von T1 zu löschen fehlgeschlagen wird, wenn irgendwelche referenzierte Zeilen in T2 existieren.
- Cascade = die referenzierte Zeile in T2 ebenfalls gelöscht wird
- Set null = Wert von T“.FK in den referenzierten Zeilen von T2 auf Null gesetzt werden (T2:FK darf in diesem Fall nicht als NOT NULL spezifiziert werden)
- Set default = Werte von T2.FK in den referenzierten Zeilen von T“ aus den anwendbaren Default_Wert gesetzt werden

### Trigger (wird praktisch gefragt)

Trigger lösen beimEintreten Bestimmter Ereignisse für bestimmte Datenobjekte eine Folge von Aktionen aus. – Bei der Benutzung reagieren die Datenbanken nach dem Eintreten bestimmter Bedingungen von allein, sie sind aktiv. Das sind die Datenbanken natürlich nur, wenn der Entwickler vorher entsprechende Trigger programmiert hat.

**Syntax:**

```
CREATE TRIGGER <trigger_name> ON <table_name>
{FOR {INSERT, UPDATE, DELETE} AS SQL_statements|
FOR {INSERT, UPDATE} AS IF UPDATE (column_name)
[{ AND|OR} UPDATE (column_name)]
SQL_statements}
```

### Beispiele der Arbeit mit Triggern (2)

2. Änderungen in den Mitteln der Tabelle Project sind nur zulässig, wenn die Erhöhung der Gesamtmittel kleiner als 50% ist

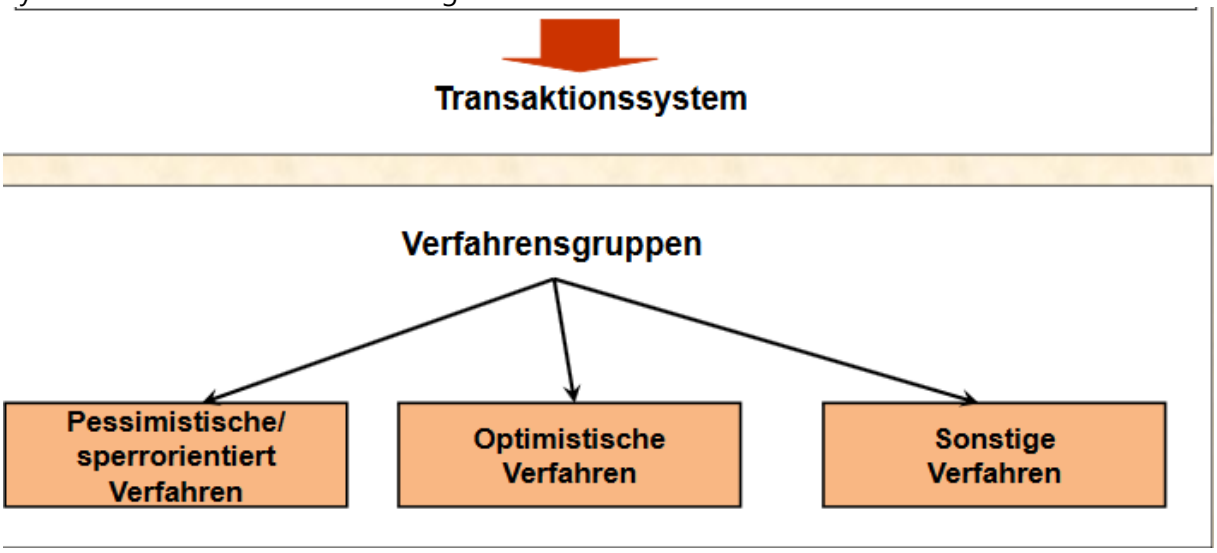
**Tabelle Project**

| Projnr | Proj_bezeich | Mittel | .... | ... |
|---|---|---|---|---|
| 1234 | DV-Projekt CAD-Hausbau | 150.000,00 | ... | |
| 1235 | Statistik-Projekt | 125.000,00 | ... | |

```
CREATE TRIGGER Mittel_pruef ON Project
   FOR UPDATE AS
   IF UPDATE(Mittel)
   BEGIN
      DECLARE @alte_summe float
      DECLARE @neue_summe float
      SELECT @alte_summe = (SELECT SUM(Mittel) FROM DELETED)
      SELECT @neue_summe = (SELECT SUM(Mittel) FROM INSERTED)
      IF @neue_summe  > @alte_summe * 1.5
          BEGIN
          ROLLBACK TRANSACTION
          PRINT "Die Änderung der Projektmittel nicht ausgeführt!"
          END
      ELSE
          PRINT "Die Änderung der Projektmittel wurde ausgeführt"
```

## Sicherung der operationalen Integrität

Sicherung der operationalen Integrität heißt, Vermeidung von Nebenwirkungen des Mehrnutzerbetriebes durch die Realisierung eines fiktiven Einzelnutzerbetriebes und die Gewährleistung der Eigenschaft der Isolation einer Transaktion. Das wird auch als Synchronisation oder Serialisierung bezeichnet.




## Sperrverfahren_Pessimistische Sperrverfahren

Bei den Sperrverfahren werden Schreib- bzw. Lesesperren als Mechanismus zur Gewährleistung der Serialisirungbarkeit benutzt.
5 Bedingungen werden dafür genannt:  siehe Skript

**Statisches Sperren:** werden zu Beginn der Transaktionen alle Sperren angefordert Man sperrt alles was man „gebrauchen könnte". Also manchmal auch etwas mehr.
**Dynamisches Sperren:** beim Dynamischen Sperren werden während der Transaktion (aber nicht in der Schrumpfungsphase) Sperren nach Bedarf angefordert, Dadurch kann es zu Verklemmungen (Deadlocks) kommen, heißt, dass es auf beide Tabellen gesperrt wurde und beide auf einander zugreifen möchten und darauf dann unendlich warten müssen.

**Sperrmodi – S-Sperre(shared) – Lesesperre:** Dabei können Datenobjekte, die mit einer S-Sperre belegt sind, auch von anderen Transaktionen mit einer S-Sperre versehen und von diesen gelesen werden. Schreiben und das Setzen von Schreibesperren wird abgewiesen.
**Sperrmodi – X-Sperre (exclusive) – Schreibsperre:** Dabei kann auf Datenobjekte, die mit einer X-Sperre versehen sind, von keiner anderen Transaktion (weder lesend noch schreibend) zugegriffen werden.

**Stufen der Isolation paralleler Transaktionen:** Es wird zwischen kurzen und langen Sperren unterschieden. Kurze Sperren wurden sofort nach Gebrauch des Datenobjekts, also vor EOT freigegeben. Lange Sperren werden bis EOT gehalten.

## Pessimistische und optimistische Verfahren vergleichen