DB II B: DBMS-Implementierung

Übung 5: Oracle Architektur

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2021/22

http://www.informatik.uni-halle.de/~brass/dbi21/

Inhalt

Vorlesung heute fällt aus!

- Wie bereits mehrfach angekündigt, fällt die Vorlesung heute leider aus.

  Ich bin Senatsberichtserstatter in einer Berufungskommission der Medizin. Die hat eine Sitzung ab 15^{45}.

- Es gibt eine Aufzeichnung in StudIP, die ich am Dienstag-Abend gemacht habe.

- Es ist der Anfang einer Einführung in C++ für Java-Programmierer.

  Schauen Sie sich das an. Wir werden beim nächsten Mal darüber abstimmen, ob Fortsetzungen gewünscht sind (ggf. auch zusätzlich zu den regulären Vorlesungen).

Zur Klausur

- Wir hatten ja eine Abstimmung über elektronische Klausur vs. Papierklausur gemacht, die zugunsten der elektronischen Klausur ausgegangen ist.
- Dann kam die Diskussion auf, dass vielleicht „elektronische Klausur“ mit „Online-Klausur“ verwechselt wurde.

  Aufgrund schlechter Erfahrungen mit Täuschungsversuchen mache ich keine Online-Klausur.

- Zweite Abstimmung: Papierklausur (relativ knapp).
- Nachdem wir es jetzt überschlafen konnten, machen wir noch eine dritte Abstimmung.
- Das Ergebnis gilt — es sei denn, ich bekomme keinen Raum an dem Termin (10.03.2022, 10-12).

  Für die elektronische Klausur habe ich schon einen Raum.

Oracle Accounts (1)

- Sie sollten zwei eigene Oracle-Accounts haben:
  - Einen normalen Account (`Nachname`).
  - Einen Account mit unbeschränktem Lese-Zugriff (`Nachname_dba`)
- Außerdem gibt es den Test-Account „`SCOTT`“:

  https://dbs.informatik.uni-halle.de/db2b/adminer?oracle=oracle-18.4-xe-db2b%2FXEPDB1&username=scott&db=USERS
- Nicht benutzte Accounts werden gelöscht!

  Wahrscheinlich werde ich zur Sicherheit `SCOTT` sperren, sobald wir mit den Zugriffsrechten fertig sind.

Oracle Accounts (2)

- Folgende Anfrage liefert 31:

```
SELECT COUNT(*)
FROM   DBA_USERS
WHERE  CREATED >= '01-JAN-21'
AND    LAST_LOGIN IS NULL
```

- Als Bedingung für das Datum könnte man auch vom aktuellen Datum 30 Tage zurück schreiben (die Datenbank war seit 2020 nicht mehr benutzt):

```
SELECT COUNT(*)
FROM   DBA_USERS
WHERE  CREATED >= SYSDATE - 30
AND    LAST_LOGIN IS NULL
```

- Das liefert auch 31.

Oracle Accounts (3)

- Man kann noch nach den Nicht-DBA Accounts fragen:

```
SELECT COUNT(*)
FROM   DBA_USERS
WHERE  CREATED >= '01-JAN-21'
AND    LAST_LOGIN IS NULL
AND    USERNAME NOT LIKE '%\_DBA' ESCAPE '\'
```

- Das Ergebnis ist 14. Entsprechend gibt es 17 DBA-Accounts, die nie benutzt wurden.
- Insgesamt habe ich jeweils 51 Accounts von beiden Typen angelegt (für die damals in StudIP eingetragenen Studenten).
- Aktuell in StudIP eingetragen: 47.

Oracle Accounts (4)

- Ich habe gestern abend 21^{00} ausgeführt:

```
ALTER USER SCOTT ACCOUNT LOCK;
```

- Jetzt werde ich das Ergebnis folgender Anfrage mit Copy&Paste das Adminer-Eingabefeld einfügen und ausführen:

```
SELECT 'ALTER USER ' || USERNAME ||
       ' ACCOUNT LOCK;'
FROM   DBA_USERS
WHERE  CREATED >= '01-JAN-21'
AND    LAST_LOGIN IS NULL
```

- Auch möglich: `DROP USER ...`.

  Es gibt noch einen Zusatz `CASCADE`, wenn auch Nutzer mit Tabellen gelöscht werden sollen. Das ist hier nicht nötig.

Links zur Oracle-Dokumentation

- Link zur Oracle 18c Dokumentation:
  [https://docs.oracle.com/en/database/oracle/
  /oracle-database/18/
- Oracle 18c Database Reference:
  [https://docs.oracle.com/en/database/oracle/
  oracle-database/18/refrn/index.html]
- Oracle SQL Language Reference:
  [https://docs.oracle.com/en/database/oracle/
  oracle-database/18/sqlrf/]
- Oracle Administrator's Guide:
  [https://docs.oracle.com/en/database/oracle/
  oracle-database/18/admin/]

Inhalt

Blatt 2, Aufgabe c) (1)

- Sie sollen eine Zeile in diese Tabelle einfügen, bei denen die erste Spalte Ihren Nutzernamen enthält.

  Die Pseudospalte `USER` ist immer der Name des aktuellen Nutzers.

- Damit die Sache nicht so einfach ist, sind auf der Tabelle Integritätsbedingungen formuliert.

- Schreiben Sie eine Anfrage, die die Integritätsbedingungen anzeigt.

- Leider ist die Aufgabenstellung wieder nicht eindeutig (warum hat niemand gefragt?).
  - Ich dachte nur an `CHECK`-Constraints.
  - Wenn man tatsächlich auch Schlüssel und Fremdschlüssel wollte, müsste man auch die Spalten ausgeben (anders strukturiertes Ergebnis).

Blatt 2, Aufgabe c) (2)

- Lösung mit (Semi-)Join, nur CHECK-Constraints:

```
SELECT C.TABLE_NAME, C.SEARCH_CONDITION
FROM   ALL_CONSTRAINTS C
WHERE  C.CONSTRAINT_TYPE = 'C'
AND    EXISTS(SELECT *
              FROM   USER_TAB_PRIVS_RECD P
              WHERE  C.OWNER = P.OWNER
              AND    C.TABLE_NAME = P.TABLE_NAME
              AND    C.OWNER = 'BRASS'
              AND    P.PRIVILEGE = 'INSERT')
```

Wie oben erläutert, kann es mehrere Tabellen geben, die die Bedingung erfüllen. Daher wird der Tabellen-Name mit ausgegeben.

Blatt 2, Aufgabe c) (3)

- Leider werden auch die `NOT NULL` constraints mit ausgegeben in der Form:

  ```
  "USERNAME" IS NOT NULL
  ```

- Theoretisch könnte man das ausschliessen mit

  ```
  AND SEARCH_CONDITION NOT LIKE
                        '"%" IS NOT NULL'
  ```

  Das wäre nicht absolut sicher, da zwischen den `"..."` beliebige Zeichen stehen könnten, z.B. auch weitere solche Anführungszeichen. Man würde dann also zu viel ausschließen. Oracle hat seit Version 10g reguläre Ausdrücke mit Bedingungen der Form `REGEXP_LIKE(`*s*`,`*r*`)` — leider nicht `SIMILAR TO` wie im Standard (die regulären Ausdrücke sind auch POSIX-kompatibel und nicht wie im SQL-Standard).

  ```
  REGEXP_LIKE(SEARCH_CONDITION, '^"[^"]*" IS NOT NULL$')
  ```

- Man bekommt aber einen Typfehler, siehe nächste Folie.

Blatt 2, Aufgabe c) (4)

- Das Problem ist, dass `SEARCH_CONDITION` vom Datentyp `LONG` ist.

  Das war eine Lösung für lange Zeichenketten, auch ganze Dateien.

  Der Typ `VARCHAR` ist auf 4000 Bytes begrenzt. Inzwischen gibt es einen Konfigurationsparameter `MAX_STRING_SIZE`, mit dem man das Limit auf 32 KByte erhöhen kann.

- Werte vom Datentyp `LONG` kann man ausgeben, aber nichts sonst damit machen, und insbesondere nicht `LIKE` oder `REGEXP_LIKE` darauf anwenden.

  Oracle empfiehlt schon lange, `LONG` nicht mehr zu verwenden, sondern `CLOB` zu nutzen. Leider hält sich Oracle beim eigenen Data Dictionary nicht daran.

- Eine Typ-Umwandlung und direkte Weiterverwendung in der Anfrage scheint nicht möglich zu sein.

  [http://www.morganslibrary.org/hci/hci010.html]

Blatt 2, Aufgabe c) (5)

- Man muss den Umweg über eine temporäre Tabelle gehen:

```
CREATE TABLE H2_CHECK(COND CLOB NOT NULL);
```

- Dort kann man die Integritäts-Bedingungen einfügen:

```
INSERT INTO H2_CHECK
SELECT TO_LOB(SEARCH_CONDITION)
FROM   ALL_CONSTRAINTS
WHERE  CONSTRAINT_TYPE = 'C'
AND    TABLE_NAME = 'H2' AND OWNER = 'BRASS'
```

  Hier funktioniert die Typ-Umwandlung, während eine direkte Verwendung von `TO_LOB(...)` in einer `LIKE`-Bedingung einen Typfehler liefert.

- Nun löscht man die umgeformten `NOT NULL`-Constraints:

```
DELETE FROM H2_CHECK
WHERE  COND LIKE '"%" IS NOT NULL'
```

Temporäre Tabellen

- Oracle hat temporäre Tabellen:

```
CREATE GLOBAL TEMPORARY TABLE H2_CHECK
                              (COND CLOB NOT NULL);
```

- Wir können sie über den Adminer nicht verwenden, der der Inhalt spätestens am Ende einer „Session“ gelöscht wird, und der Adminer scheint für jede Anfrage eine neue Sitzung zu öffnen:

```
SELECT SYS_CONTEXT('USERENV', 'SESSIONID')
FROM   DUAL
```

Normal wird der Inhalt sogar am Ende der Transaktion gelöscht. Man kann aber am Ende der `CREATE TABLE` Anweisung (nach der Klammer zu) schreiben: `ON COMMIT PRESERVE ROWS`.
[https://oracle-base.com/articles/misc/temporary-tables]

Blatt 2, Aufgabe e) (1)

Aufgabe e)

- Legen Sie selbst eine Tabelle an mit gleichem Namen, Spalten, Datentypen und Primärschlüssel wie die Tabelle der obigen Aufgaben.
- Als `CHECK`-Constraint wählen Sie dagegen eine andere Bedingung (gerne eine ganz einfache Bedingung).
- Geben Sie das `CREATE TABLE`-Statement ab.
- Fügen Sie eine Zeile in Ihre Tabelle ein.

Blatt 2, Aufgabe e) (2)

- Man muss noch den Primärschlüssel dieser Tabelle herausfinden:

```
SELECT COL.POSITION, COL.COLUMN_NAME
FROM   ALL_CONSTRAINTS CON,
       ALL_CONS_COLUMNS COL
WHERE  CON.OWNER = 'BRASS'
AND    CON.TABLE_NAME = 'H2'
AND    CON.CONSTRAINT_TYPE = 'P'
AND    CON.CONSTRAINT_NAME = COL.CONSTRAINT_NAME
AND    CON.OWNER = COL.OWNER
ORDER  BY COL.POSITION
```

Diese Anfrage musste man nicht abgeben. Man braucht aber das Ergebnis. Außerdem hätte man bei der `ZAHL`-Spalte zur Sicherheit noch `DATA_SCALE` abfragen müssen (es ist aber 0, wie der Constraints auch nahelegt).

Blatt 2, Aufgabe e) (3)

- Lösung:

```
CREATE TABLE H2 (
    USERNAME VARCHAR(20) NOT NULL PRIMARY KEY,
    ZAHL     NUMERIC(5)  NOT NULL,
    CONSTRAINT ZAHL_OK CHECK(ZAHL = 7)
);
```

- Zeile einfügen (für Updates, siehe nächste Teilaufgabe):

```
INSERT INTO H2 VALUES(USER, 7);
```

Blatt 2, Aufgabe e) (4)

- Der Datentyp

  `NUMERIC(22,5)`

  für die ZAHL-Spalte ist falsch:

  - `DATA_LENGTH` hatte den Wert 22, aber ist die maximale Speicherlänge in Bytes.

    „Maximal“ bezieht sich hier auf sämtliche `NUMERIC`-Typen. Die maximale Speichergröße von `NUMERIC(5)` ist viel kleiner (zufällig 5 Bytes, es ist aber nicht einfach die Anzahl der Ziffern, dies wird später in der Vorlesung erläutert).

  - `DATA_LENGTH` ist nützlich für `VARCHAR`-Typen, dort ist es der Typ-Parameter.
  - Für `NUMERIC`-Typen braucht man dagegen `DATA_PRECISION`, um die Anzahl Ziffern herauszufinden.

Blatt 2, Aufgabe e) (5)

- Die Datentypen `VARCHAR2` und `NUMBER` sind Oracle-spezifische Synonyme für `VARCHAR` und `NUMERIC`:
  - Natürlich zählt es als korrekt, die im Data Dictionary eingetragenen Datentypen `VARCHAR2` und `NUMBER` zu verwenden.
  - Aber `VARCHAR` und `NUMERIC` wären auch korrekt (und sind standard-konform).
- Der Typ `INT` statt `NUMERIC(5)` ist auch falsch.
- Oft fehlte der Primärschlüssel.
- Ein Primärschlüssel aus beiden Spalten ist auch falsch.

Inhalt

Blatt 3, Aufgabe a) (1)

Aufgabe:

- Drucken Sie eine Liste aller Systemprivilegien, die Sie haben — entweder direkt, oder über eine Rolle.
- Sie können dabei `GRANT`s von Rollen an Rollen ignorieren, müssen also keine rekursive Anfrage schreiben.

  Wenn Sie Spaß daran haben, dürfen Sie das aber tun.

Blatt 3, Aufgabe a) (2)

- Mögliche Lösung:

```
SELECT PRIVILEGE
FROM   USER_SYS_PRIVS
UNION
SELECT S.PRIVILEGE
FROM   USER_ROLE_PRIVS R, ROLE_SYS_PRIVS S
WHERE  R.GRANTED_ROLE = S.ROLE
```

- Der Join in der zweiten Hälfte ist vermutlich überflüssig. Das Handbuch sagt zu `ROLE_SYS_PRIVS`: „Information is provided only about roles to which the user has access.“

```
UNION
SELECT PRIVILEGE
FROM   ROLE_SYS_PRIVS
```

Inhalt

Blatt 3, Aufgabe b) (1)

Aufgabe:

- Finden Sie heraus, wie groß die bei unserer Oracle-Datenbank verwendete Blockgröße ist.
- Geben Sie die Anfrage ab, die die Blockgröße liefert mit einem Kommentar für das Ergebnis.

Blatt 3, Aufgabe b) (2)

- Die Aufgabenstellung ist leider nicht mehr passend für neuere Oracle-Versionen.
- Seit Oracle 9i kann jeder Tablespace eine eigene Blockgröße haben.
- Die Auswahl ist offenbar beschränkt auf 2 KByte, 4 KByte, 8 KByte, 16 KByte und 32 KByte.

  Wenn man eine Blockgröße verwendet, braucht man auch einen Puffer/Cache für diese Blockgröße. Hierfür gibt es Parameter wie `db_32k_cache_size`.
  [https://oracle-base.com/articles/9i/multiple-block-sizes]

Blatt 3, Aufgabe b) (3)

- Mögliche Lösung:

```
SELECT BLOCK_SIZE
FROM   DBA_TABLESPACES
WHERE  TABLESPACE_NAME = 'USERS'
```

  Man könnte auch mit einer Unteranfrage Tablespaces auswählen, in denen man Tabellen hat.

- Den Parameter für die Standard-Blockgröße (verwendet für den `SYSTEM`-Tablespace) kann man so abfragen:

```
SELECT VALUE
FROM   V$PARAMETER
WHERE  NAME = 'db_block_size'
```

  Beachten Sie, dass Parameternamen in dieser Tabelle klein geschrieben werden.

Inhalt

Blatt 3, Aufgabe c) (1)

- Schreiben Sie eine Anfrage, die Dateien unserer Oracle-Datenbank auflistet.
- Es sollen Dateien von mindestens drei verschiedenen Typen sein, die im Skript genannt werden (mehr wären willkommen). Das Ergebnis soll vier Spalten haben:
  - Eine Nummer, mit der Sie bei der Sortierung eine sinnvolle Reihenfolge bekommen.
  - Den Typ der Datei als lesbare Bezeichnung.
  - Den Dateinamen (bzw. den ganzen Pfadnamen).
  - Ggf. Zusatzinformationen, z.B. den Tablespace bei den Dateien mit den Tabellendaten (Datenbank-Dateien).
- Sortieren Sie die Ausgabe sinnvoll.

Blatt 3, Aufgabe c) (2)

- Mögliche Lösung:

```
SELECT 1 AS SEQ, 'Controlfile' AS FILETYPE,
       NAME, TO_CHAR(NULL) AS INFO
FROM   V$CONTROLFILE
UNION  ALL
SELECT 2 AS SEQ, 'Datafile' AS FILETYPE,
       F.NAME, T.NAME AS INFO
FROM   V$DATAFILE F, V$TABLESPACE T
WHERE  F.TS# = T.TS#
UNION  ALL
SELECT 3 AS SEQ, 'Logfile' AS FILETYPE,
       MEMBER, TO_CHAR(GROUP#) AS INFO
FROM   V$LOGFILE
ORDER  BY SEQ;
```

Blatt 3, Aufgabe c) (3)

- Die Dateien für temporäre Daten fehlen noch:

```
SELECT NAME
FROM   V$TEMPFILE
```

  Über `TS#` kann man es mit `V$TABLESPACE` joinen.

- Weitere Daten aus `V$PARAMETER`:
  - `spfile`: Server Parameter File.

```
SELECT VALUE
FROM   V$PARAMETER
WHERE  NAME = 'spfile'
```

  - `background_dump_dest`

    Auch `user_dump_dest`, `core_dump_dest`, `diagnostic_dest`.

  - `log_archive_dest%` (bei uns `NULL`)
  - `audit_file_dest`