DB II B: DBMS-Implementierung

Übung 4: Systemkatalog, Zugriffsrechte (Teil II)

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2021/22

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

Inhalt

Vorlesung am 11.11.2021 fällt aus!

- Die Vorlesung in der nächsten Woche, am 11.11.2021, muss leider ausfallen.

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

- Die Übung findet statt!
- Ich versuche noch, eine Aufzeichnung am Mittwoch zu machen, so dass es ein Video gibt.
- Möglicherweise würde ich da dann die C++ Einführung machen.

Zur elektronischen Klausur

- Bei der elektronischen Klausur gab es offenbar ein Missverständnis. Geplant ist keine Online-Klausur, sondern eine Klausur am Rechner in Präsenz.

  Wahrscheinlich im Lührmann-Gebäude (ehemals Karstadt). Das LLZ baut da gerade ein Prüfungszentrum auf.

- Ich habe eine Erfahrung mit einer Online-Klausur, die wegen Corona unumgänglich war.
  Dabei gab es nachweisbare Täuschungsversuche.

  Und das war vermutlich nur die Spitze des Eisbergs. Diese Studenten haben es besonders unclever angestellt. Daher sehe ich die Integrität der Prüfung bei diesem Format gefährdet. Es ist aber mein Job, dafür zu sorgen.

- Vorteil für Sie: Es gibt einen Mitarbeiter des LLZ, der technische Probleme löst.

  Und alle Teilnehmer haben die gleiche Rechner-Ausstattung.

Oracle Accounts

- 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.

Inhalt

Blatt 2, Aufgabe a) (1)

Data Dictionary Abfrage: Erhaltene Rechte

- Schreiben Sie eine Anfrage, die Ihnen alle Tabellen listet,
  - an denen Ihnen direkt Zugriffsrechte gegeben wurden (also nicht über eine Rolle).
  - Drucken Sie den Besitzer der Tabelle (`OWNER`), den Tabellen-Namen (`TABLE_NAME`) und das Recht (`PRIVILEGE`).
- Lösung:

```
SELECT OWNER, TABLE_NAME, PRIVILEGE
FROM   USER_TAB_PRIVS
WHERE  GRANTEE = USER -- Oder z.B. 'BRASS'
```

`USER` liefert den Namen des aktuell eingeloggten Nutzers. Das ist Oracle-spezifisch, nach dem Standard ist es `CURRENT_USER`. Beachten Sie, dass man bei dieser nullstelligen Funktion keine `()` angeben darf.

Blatt 2, Aufgabe a) (2)

- Die Tabelle `USER_TAB_PRIVS` enthält alle Rechte-Vergaben, die Sie (den aktuellen Nutzer) betreffen:
  - Rechte, die Sie bekommen haben.

    Das war bei dieser Aufgabe gewünscht.
  - Rechte, die Sie vergeben haben.
  - Rechte, die andere an Ihren Tabellen vergeben haben (das geht nur, wenn Sie diesen Nutzern das Recht `WITH GRANT OPTION` gegeben haben).
- Das Folgende reicht also nicht:

```
SELECT OWNER, TABLE_NAME, PRIVILEGE
FROM   USER_TAB_PRIVS
```

Blatt 2, Aufgabe a) (3)

- Es gibt aber eine spezielle Sicht, die nur empfangene Rechte enthält:

```
SELECT OWNER, TABLE_NAME, PRIVILEGE
FROM   USER_TAB_PRIVS_RECD
```

- Die folgende Abfrage hat micht verblüfft:

```
SELECT OWNER, TABLE_NAME, PRIVILEGE
FROM   USER_TAB_PRIVS_RECD
WHERE  TYPE = 'TABLE'
```

  Ich wusste nicht, dass es eine Spalte `TYPE` in dieser Tabelle gibt.

Blatt 2, Aufgabe a) (4)

- Erster Versuch:

```
SELECT * FROM DICT_COLUMNS
WHERE  TABLE_NAME = 'USER_TAB_PRIVS'
```

  Leider sind alle `COMMENTS` Null.

- Link zur Oracle 18c Dokumentation:
  [https://docs.oracle.com/en/database/oracle/
  /oracle-database/18/
  Sie brauchen die „Database Reference“ (unter „Administration“), dann „Static Data Dictionary Views“.
  [https://docs.oracle.com/en/database/oracle/
  oracle-database/18/refrn/index.html]

- Die ausführliche Erklärung finden Sie bei `DBA_TAB_PRIVS` und `ALL_TAB_PRIVS`.

Blatt 2, Aufgabe a) (5)

- Die Spalte HIERARCHY kam bei Version 9i hinzu, die Spalten COMMON und TYPE bei Version 12c, INHERITED bei 18c.

  Oracle 12c: [https://docs.oracle.com/database/121/]

  Oracle 11g: [https://docs.oracle.com/cd/E11882_01/index.htm]

  Oracle 10g: [https://docs.oracle.com/cd/B19306_01/index.htm]

  Oracle 9i: [https://docs.oracle.com/cd/B10501_01/index.htm]

  Oracle 8i: [https://docs.oracle.com/cd/A87860_01/doc/server.817/index.htm]

- HIERARCHY gehört zu den objektrelationalen Features (ob der GRANT auch für „Unterklassen-Sichten“ gilt).

  „The phrase WITH HIERARCHY OPTION grants a specified object privilege on all subobjects of the object.“

- COMMON wurde wegen der „Multitenant-Architektur“ eingeführt — es ist YES, wenn der GRANT für alle „pluggable databases“ und die Wurzel-DB gilt (ganze CDB).

Blatt 2, Aufgabe a) (6)

- Wenn es Sie interessiert, können Sie auch den `GRANT`-Befehl in der „SQL Language Reference“ nachschlagen:

  [https://docs.oracle.com/en/database/oracle/oracle-database/18/sqlrf/

  Direkter Link zum `GRANT`-Befehl:

  [https://docs.oracle.com/en/database/oracle/oracle-database/18/sqlrf/GRANT.html]

- Das Kapitel 7 im „Database Administrator's Guide“ beschäftigt sich mit Sicherheit:

  [https://docs.oracle.com/en/database/oracle/oracle-database/18/admin/]

- Und es gibt noch den „Database Security Guide“:

  [https://docs.oracle.com/en/database/oracle/oracle-database/18/dbseg/]

Blatt 2, Aufgabe a) (7)

- Auch verblüfft hat mich:

```
select unique owner, table_name, privilege
from   user_tab_privs_recd
where  type = 'TABLE'
```

- Es heisst im Standard `DISTINCT`. Oracle akzeptiert `UNIQUE` als Synonym.

  Die Aussage, dass sie Synonyme sind, steht in der SQL Language Reference:
  [https://docs.oracle.com/en/database/oracle/oracle-database/18/sqlrf/SELECT.html]

- Die SQL-2016 Grammatik findet man z.B. hier:

  [https://jakewheat.github.io/sql-overview/sql-2016-foundation-grammar.html]

  `SELECT` steht unter 7.16 „query specification“. „set quantifier“ ist nur `DISTINCT` oder `ALL`.

Blatt 2, Aufgabe a) (8)

- Normalerweise ist Portabilität wichtig:
  - Man zahlt einen Preis (mehr Arbeit, falls man jemals auf ein anderes DBMS wechseln muss).
  - Dafür sollte man einen Vorteil (Gegenwert) bekommen (kürzere Anfrage, bessere Performance).
- Data Dictionary Anfragen sind aber immer system-spezifisch.

  (Etwas) Portabilität könnte man nur erreichen, wenn man das „`INFORMATION_SCHEMA`“ aus dem SQL-Standard verwendet, aber das unterstützt Oracle nicht. Viele interessante Daten sind auch inhärent system-spezifisch und fehlen im `INFORMATION_SCHEMA`.
- Aber: Es gibt auch den Aspekt der Lesbarkeit.
  Ich (und alle kundigen SQL-Entwickler) kennen `DISTINCT`.
  Bei `UNIQUE` musste ich erst nachschlagen und länger lesen.

Blatt 2, Aufgabe a) (9)

- Ich habe die Duplikat-Elimination zuerst für überflüssig gehalten.
- `OWNER` und `TABLE_NAME` identifizieren die Tabelle eindeutig.

  Die `DBA_*`, `ALL_*` und `USER_*` Tabellen beziehen sich alle nur auf den aktuellen Container, bei dem man angemeldet ist (also unsere „Pluggable Database“ `XEPDB1`). Wenn man beim Root-Container angemeldet ist und spezielle Rechte hat, kann man auch auf `CDB_*` zugreifen, und würde dort noch eine Container-ID brauchen zur eindeutigen Identifikation.
- Tatsächlich ist es aber möglich, dass man das gleiche Recht an der gleichen Tabelle von verschiedenen Nutzern als „`GRANTOR`“ bekommen hat.

  Insofern wäre die Anfrage mit `DISTINCT` die korrekte Lösung.

Blatt 2, Aufgabe a) (10)

- Noch eine andere Lösung:

```
SELECT OWNER, TABLE_NAME, PRIVILEGE
FROM   USER_TAB_PRIVS
WHERE  GRANTEE =
       SYS_CONTEXT('USERENV', 'SESSION_USER');
```

- Die Funktion `SYS_CONTEXT` erlaubt die Abfrage vieler Daten der Sitzung.

  [https://docs.oracle.com/en/database/oracle/oracle-database/18/sqlrf/SYS_CONTEXT.html]

Inhalt

Blatt 2, Aufgabe b) (1)

- Unter diesen Tabellen müsste auch eine Tabelle des Nutzers BRASS sein, für die Sie sogar Einfüge-Rechte haben.

  Die Aufgabe ist leider mehrdeutig: Man kann es so verstehen, dass man den Tabellennamen in die Anfrage einsetzen darf, oder dass man den Join berechnen muss. Als SCOTT gab es auch zusätzlich noch die Tabelle HA3 vom letzten Jahr. Bei den persönlichen Accounts nicht.

- Schreiben Sie eine SQL-Anfrage, die Ihnen die Spalten dieser Tabelle (COLUMN_NAME, COLUMN_ID) mit Datentyp (DATA_TYPE, DATA_LENGTH, DATA_PRECISION) anzeigt.

- Sortieren Sie die Spalten in der deklarierten Reihenfolge (also nach der COLUMN_ID).

Blatt 2, Aufgabe b) (2)

- Lösung mit (Semi-)Join:

```
SELECT C.COLUMN_NAME, C.COLUMN_ID,
       C.DATA_TYPE, C.DATA_LENGTH,
       C.DATA_PRECISION
FROM   ALL_TAB_COLUMNS C
WHERE  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')
ORDER  BY C.COLUMN_ID
```

Man hätte unter `SELECT` die Tupelvariable `C` auch weglassen können, es gibt hier ja nur eine deklarierte Tupelvariable. Siehe auch nächste Folie.

Blatt 2, Aufgabe b) (3)

- Weil es mehrere Tabellen geben könnte (und für `SCOTT` auch gibt), sollte man in diesem Fall entgegen der Aufgabenstellung auch den `TABLE_NAME` mit ausgeben.

  Wenn Sie von der Aufgabenstellung abweichen, schreiben Sie einen Kommentar dazu, warum Sie das tun. Zumindest muss dann bei der Korrektur nicht vermutet werden, dass Sie die Aufgabenstellung nicht genau gelesen haben. Sie können natürlich auch fragen.

- Durch die Lösung mit `EXISTS` gibt es keine Duplikate, auch wenn das gleiche Zugriffsrecht auf verschiedenen Wegen empfangen wurde.

  Beispiel: `BRASS` gibt Ihnen direkt das Zugriffsrecht, und er gibt dem Nutzer `BRASS_DBA` das Zugriffsrecht `WITH GRANT OPTION`, und dieser Nutzer gibt Ihnen das Recht nochmals. Wenn Sie `TABLE_NAME` nicht angeben, könnte es deswegen natürlich Duplikate geben. Die wären aber wohl erwünscht (man merkt dann, dass es zwei Tabellen gibt).

Blatt 2, Aufgabe b) (4)

- Lösung mit eingesetztem Tabellennamen:

```
SELECT COLUMN_NAME, COLUMN_ID,
       DATA_TYPE, DATA_LENGTH,
       DATA_PRECISION
FROM   ALL_TAB_COLUMNS
WHERE  OWNER = 'BRASS'
AND    TABLE_NAME = 'H2'
ORDER  BY C.COLUMN_ID
```

- Das Schlüsselwort `ASC` beim `ORDER BY` ist überflüssig (aber nicht falsch):

```
ORDER BY C.COLUMN_ID ASC
```

- Dagegen entspricht es nicht der Aufgabenstellung, nach `COLUMN_NAME` zu sortieren.

Blatt 2, Aufgabe b) (5)

- Ein natürlicher Verbund funktioniert offenbar:

```
FROM  ALL_TAB_COLUMNS NATURAL JOIN
      USER_TAB_PRIVS_RECD
```

- Man muss die Spalten der beiden Tabellen genau prüfen.
- Wie wir gesehen haben, werden die Data Dictionary Tabellen manchmal erweitert.

  Man muss im Normalfall davon ausgehen, dass das bei jeder Tabelle irgendwann in der Zukunft geschehen kann. Dann würde sich die Bedeutung der Anfrage ändern (sie würde höchtswahrscheinlich falsch werden). Und es gibt keine Fehlermeldung, nur ein falsches Ergebnis.

- Verwenden Sie besser `USING`, das ist zukunftssicher:

```
FROM  ALL_TAB_COLUMNS JOIN
      USER_TAB_PRIVS_RECD
      USING (OWNER, TABLE_NAME)
```

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'
```

Inhalt

Blatt 2, Aufgabe d) (1)

- Anschließend suchen Sie passende Werte aus, die Sie einfügen können, und fügen die Zeile ein als Beleg Ihres Erfolgs.
- Geben Sie das `INSERT`-Statement ab.
- Die Tabelle war so deklariert:

```
CREATE TABLE H2 (
    USERNAME VARCHAR(20) NOT NULL PRIMARY KEY,
    ZAHL     NUMERIC(5)  NOT NULL,
    CONSTRAINT ZAHL_OK
    CHECK(ZAHL < 200 AND MOD(ZAHL,50) IN (7,23)
          OR ZAHL BETWEEN 200 AND 500
             AND MOD(ZAHL-1,50) IN (19,42)
          OR ZAHL > 500 AND MOD(ZAHL-500,30) IN
             (1,3,7,11,13,17,19,23,29) ) );
```

Blatt 2, Aufgabe d) (2)

- Leider habe ich vergessen, `ZAHL` als `UNIQUE` zu deklarieren.
- So konnte jeder die gleiche Zahl nehmen. Geplant war natürlich, dass jeder Teilnehmer eine neue Zahl finden muss, die die Bedingung erfüllt.
- Die am häufigsten verwendeten Zahlen waren:

Zahl	wie oft
7	10
23	7
57	6
123	2

Außerdem wurden 73, 157, 320, 501, 531, 533 je ein Mal verwendet.

Blatt 2, Aufgabe d) (3)

- Eine Lösung ist also:

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

- Man konnte natürlich auch den Nutzernamen direkt einsetzen:

```
INSERT INTO BRASS.H2
                VALUES('Mein Name', 220);
```

- Wenn die Anweisung in einem Programm oder Skript steht, und auch noch funktionieren soll, nachdem die Tabelle um weitere Spalten erweitert wurde, müsste man die Spaltennamen explizit angeben:

```
INSERT INTO BRASS.H2(USERNAME, ZAHL)
                VALUES(USER, 220);
```

Inhalt

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 2, Aufgabe f) (1)

- Geben Sie den Nutzern BRASS und SCOTT Lese-Rechte und das UPDATE-Recht an Ihrer Tabelle (dagegen sollen Sie INSERT und DELETE nicht erlauben).
- Geben Sie den entsprechenden SQL-Befehl ab.
- Lösung:

```
GRANT SELECT, UPDATE ON H2 TO BRASS, SCOTT
```

- Einzelne Befehle sind auch möglich, aber umständlicher:

```
GRANT SELECT ON H2 TO BRASS, SCOTT
GRANT UPDATE ON H2 TO BRASS, SCOTT
```

Man kann auch noch für jeden der beiden Benutzer einzeln einen Befehl schreiben.

Blatt 2, Aufgabe f) (2)

- Einige Studierende haben ihren Nutzernamen vor dern Tabellennamen geschrieben:

  ```
  GRANT SELECT, INSERT ON MEIER.H2 TO BRASS, SCOTT
  ```

- Das ist überflüssig: Wenn man als `MEIER` eingelogged ist, beziehen sich alle Tabellennamen ohne Präfix automatisch auf die eigene Tabelle.

  Einzige Ausnahme wären „Public Synonyms“, wie sie z.B. beim Data Dictionary verwendet werden. Aber sie würden von einer eigenen Tabelle gleichen Namens verschattet werden.

- Der Nutzername ist nur notwendig, wenn man selbst nicht `MEIER` ist, aber das Recht `WITH GRANT OPTION` bekommen hat, und jetzt das Recht an der fremden Tabelle weitergeben will.