SQLPlus

User: st03

Pass: cs514 (default, change after Lab1 graded)

Second account: st03 + 210 = st213

Login from Rohan: sqlplus username/password@cfedb.sdsu.edu

Change Password

ALTER USER <user\_name> IDENTIFIED BY <new\_password>;

ALTER USER st01 IDENTIFY BY st99;

Set page line height: set pages 200

Run last valid SQL command: /

Directives such as describe don’t need an ending semi-colon: desc empbb02

Directives talk directly to SQL Plus

SQL requires an ending semi-colon: SELECT \* FROM empbb02;

CREATE TABLE sampsamp

(

one VARCHAR2(5),

two VARCHAR2(5) NOT NULL

);

Drop table and purge trash bin: DROP TABLE name PURGE;

INSERT INTO sampsamp VALUES

(

‘abe’,

‘LF’

);

-- two dashes is the start of a SQL comment

SQL Flavors

1. DDL (Data Definition Language)
   1. CREATE/DROP VIEW
   2. CREATE/DROP TABLE
   3. CREATE/DROP TRIGGER
2. DML (Data Manipulation Language)
   1. INSERT INTO
   2. DELETE FROM
   3. UPDATE SET
3. DCL (Data Control Language)
   1. Authentication
   2. Authorization
   3. Grant
   4. Revoke

Database Properties

* Persistence
  + Difficult to delete data permanently
* Integrity
  + Constraints are in place to prevent insertion of invalid data
* Concurrency
* Independence

SQL Transactions

* Commit
  + Permanently commit changes made thus far in the session
  + Entering ‘quit’ also does this, but it ends the session as well
* Rollback
  + Used to undo changes of current transaction

Generation Languages (GL)

* 1GL – Machine
* 2GL – Assembly Language
* 3GL – High Level Programming Language
  + Command = Imperative Language
* 4GL – SQL
  + Descriptive Language

Backup & Recovery

* Create duplicate tables to save a backup copy

CREATE TABLE sampsamp\_backup AS SELECT \* FROM sampsamp;

Table Keys (ID Columns)

* Primary Key (PK)
  + A selected CK
* Candidate Key (CK)
  + A key that can serve as a PK
  + A column that can be part of a set of CK’s
* Set of Candidate Keys
  + The minimal set of columns that enables the identification of any row in a valid set of rows for a table
  + A set of minimal columns is such that if any CK column is removed from the set, then the first property is no longer true

CREATE TABLE tnt  
(

a NUMBER PRIMARY KEY,

b NUMBER

);

CREATE TABLE tnt

(

a NUMBER,

b NUMBER,

PRIMARY KEY(a, b)

);

ALTER TABLE <table name> ADD CONSTRAINT <[table name]\_[column name]\_[constraint type abbreviated]> empbb02\_empno\_pk PRIMARY KEY(empno);

ALTER TABLE empbb02 ADD CONSTRAINT empbb02\_empno\_pk PRIMARY KEY(empno);

UNIQUE

CHECK

FOREIGN KEY

afiedt.buf

* SQL Plus default edit save file

set time on

set time off

Abbreviations

* DA (Data Administrator)
  + Logical design
* DBA (Database Administrator)
  + Physical design

User Stuff

desc user\_constraints

desc user\_cons\_columns

desc all\_constraints

desc user\_catelog

desc user\_tables

SELECT owner, constraint\_name, constraint\_type FROM user\_constraints;

Instance

* An instance of Oracle

Schema

* User accounts (st13)

Foreign Keys (FK)

* Table columns that point to PK in other tables to form a parent-child relationship

|  |  |  |
| --- | --- | --- |
| **Table** | empbb02 | infobb02 |
| **Foreign Key (Column)** | empno | empno |
| **Role** | Parent | Child |

ALTER TABLE infobb02 ADD CONSTRAINT infobb02\_empno\_fk FOREIGN KEY(empno) REFERENCES empbb02(empno);

Equijoin or Natural Join – Using a foreign key

SELECT \* FROM empbb02 e, infobb02 i WHERE e.empno = i.empno;

SELECT \* FROM empbb02 NATURAL JOIN infobb02;

Constraint Types

* C (check constraint – ex: NOT NULL)
* P (primary key)
* U (unique key)
* R (referential integrity – foreign key)
* V (check option on a view)
* O (read only on a view)

ALTER TABLE <table name> DROP CONSTRAINT <constraint name>

ALTER TABLE <table name> ADD CONSTRAINT <constraint name> UNIQUE (col1, col2, ...)

ALTER TABLE <table name> ADD CONSTRAINT <constraint name> CHECK (sal > 1500)

Concatenation

SELECT ename || ‘ makes ‘ || sal || ‘ dollars’ FROM empbb02;

Column Aliases

SELECT ename AS “Last Name” FROM empbb02;

Double quotes for column aliases

* Used to preserve character case and whitespace
* LastName for example would not require the quotes

Representing NULL

SELECT sal, NVL(incentives, 0) AS “Incentives” FROM empbb02;

Experimenting

SELECT ROUND(1234.56789, 3) FROM dual; // dual is shorthand for a sandbox calculator for testing

SELECT EXP(1) FROM dual; // value of e^1

SELECT LN(1) FROM dual;

SELECT LOG(9, 3) FROM dual;

SELECT POWER(2, 10) FROM dual;

SELECT UPPER(ename) FROM empbb02;

SELECT LOWER(ename) FROM empbb02;

SELECT INITCAP(ename) FROM empbb02; // sets the first character in each word to uppercase

SELECT CONCAT(ename, ‘dummy’) FROM empbb02;

SELECT LTRIM(‘abcdef’, ‘ab’) FROM dual;

SELECT RTRIM(‘abcdef’, ‘cd’) FROM dual;

SELECT SUBSTR(‘abcdefgh’, 2, 4) FROM dual; // search string, start, length

INSTR function // returns the position in the string that is the first character of a specified occurrence of the substring

SELECT INSTR('Corporate Floor','or', 3) FROM dual; // search string, look for, position to start searching

SELECT INSTR('Corporate Floor','or', 3, 2) FROM dual; // search string, look for, position to start searching, occurrence to find

Database Layers/Levels

1. Internal
   1. How the actual data is physically structured
2. Conceptual / Logical
   1. Conceptual = design
   2. Logical = tables
3. External
   1. Views – snapshots of specific views of a table (ex: the first 10 rows)
   2. 1 table can have many views

SELECT LPAD(sal, 8, '\*') FROM empbb02; // 8 is the total length

SELECT RPAD(sal, 8, '\*') FROM empbb02;

SELECT REPLACE('hiphop, 'h', 'fl') FROM dual; // output: flipflop

SELECT TRUNC(hiredate) FROM empbb02;

SELECT ROUND(hiredate, 'month') FROM empbb02;

SELECT LAST\_DAY(hiredate) FROM empbb02;

SELECT NEXT\_DAY(hiredate, 'friday') FROM empbb02;

SELECT ADD\_MONTHS(hiredate, 8) FROM empbb02;

SELECT MONTHS\_BETWEEN(hiredate, sysdate) FROM empbb02; // sysdate means today

SELECT sysdate FROM dual;

SELECT systimestamp FROM dual;

SELECT GREATEST(3, 77, 99, 44) FROM dual; // greatest means youngest, and least means oldest

SELECT TO\_CHAR(sysdate, 'mon dd, yy') FROM dual;

SELECT TO\_CHAR(sysdate, 'mon dd, yyyy') FROM dual;

SELECT TO\_CHAR(sysdate, 'mm dd, yy') FROM dual;

SELECT TO\_CHAR(sysdate, 'month dd, yy') FROM dual;

SELECT TO\_CHAR(sysdate, 'mon dd, year') FROM dual;

SELECT TO\_CHAR(sysdate, 'mon day, yy') FROM dual;

SELECT TO\_CHAR(sysdate, 'mon dy, yy') FROM dual;

SELECT TO\_CHAR(sysdate, 'mon dd, yy am') FROM dual;

SELECT TO\_CHAR(sysdate, 'mon dd, yy hh:mi:ss') FROM dual;

SELECT TO\_CHAR(sysdate, 'mon dd, yy hh24:mi:ss') FROM dual;

SELECT TO\_CHAR(1234.56789, '99999999999999') FROM dual;

SELECT TO\_CHAR(1234.56789, '0009900.00') FROM dual;

SELECT TO\_CHAR(1234.56789, '$999,999,900.00') FROM dual;

SELECT TO\_CHAR(1234.56789, 'L999,999,900.00') FROM dual; // L means local currency symbol

SELECT ename, sal DECODE(deptno, 10, sal+56,

20, sal+70,

30, sal+80,

sal) // else

FROM empbb02;

SELECT sal,

CASE

WHEN sal <= 2000 THEN 'Lo Pay'

WHEN sal <= 60000 THEN 'Medium Pay'

WHEN sal < 100000 THEN 'Hi Pay'

ELSE 'Over Paid'

END

FROM empbb02;

Relational Algebra (RA)

Relational databases are based on useful and mathematical theory. A database is a cross-product of various sets.

Theory: rows and columns are an unordered set.

Basic set of operators enhanced by composition:

1. Rename
   1. Rename R to S
2. Union
   1. ∪
   2. Assumes union compatibility (columns have identical names and types)
3. Intersect
   1. ∩
   2. Assumes union compatibility (columns have identical names and types)
4. Relative Complement
   1. – (minus)
   2. Assumes union compatibility (columns have identical names and types)
5. Select
   1. σ (little sigma)
   2. σ ename, sal (empbb02)
6. Restrict / Project (where)
   1. ρ (little rho) boolean(emp) (ex: deptno < 50)
   2. π (pi)
7. Times / Product
   1. T1 x T2
8. Natural Join
   1. ⋈
9. Divide
   1. ÷
   2. Defined by the other 8
   3. A = B ÷ C
   4. Theoretical operator
   5. Relative complement of the columns in B & C
   6. For every, for all, for any

--------------------------

Example of Natural Join:

T3 = T1 ⋈ T2

Let T1 have the attributes: {A1…An, C1…Ck}

And T2 have the attributes: {B1…Bn, C1…Ck}

Then T3 has the attributes: {A1…An, B1…Bn, C1…Ck} where the {C1…Ck} from T1 and T2 match

---------------------------

Find lname, fname, pos, uniform# for department’s ≠ 40

ρ deptno ≠ 40 (σ ename, fname, pos, uni (empbb02 ⋈ infobb02))

---------------------------

CREATE TABLE t117 AS SELECT ename, sal FROM empbb02

WHERE deptno < 40;

CREATE TABLE t118 AS SELECT ename, sal FROM empbb02

WHERE deptno > 20;

SELECT \* FROM t117 UNION SELECT \* FROM t118;

SELECT \* FROM t117 INTERSECT SELECT \* FROM t118;

SELECT \* FROM t117 MINUS SELECT \* FROM t118;

Relational Calculus (RC)

Domain

Tuple (this is the type we care about)

Find enames and salaries for deptno < 40

{x.ename, x.sal: empbb02(x) ∩ x.deptno < 40}

Types of WFF (Well-Formed Formula):

1. Atomic
2. NOT(WFF) // NOT(x.a > 30)
3. WFF(1) OR WFF(2)
4. ∃x(WFF) // there exists, replaces ⋈

Atomic WFF’s:

1. Tablename(x)
2. x.a Θ C // Θ : { ≠, =, <, >, ≤, ≥ }
3. x.a Θ x.b

AND ∧

OR ∨

Find last and first names for all infielders using RC // this should be for dept 10 for this solution

{x.ename, y.fname: empbb02(x) ∧ ∃y[infobb02(y) ∧ y.empno = x.empno] ∧ y.deptno = 10}

Joins!!!!!

Natural

SELECT \* FROM empbb02 NATUARL JOIN infobb02;

Equijoin

SELECT \* FROM empbb02 e, infobb02 i WHERE e.empno = i.empno;

Inner

SELECT \* FROM empbb02 e INNER JOIN infobb02 i ON e.empno = i.empno;

Outer Join

Self

SELECT e.ename || ‘ works for ‘ || e2.ename AS “Who’s the Boss” FROM empbb02 e, empbb02 e2 WHERE e.boss = e2.empno;

Special Operators

SELECT \* FROM infobb02 WHERE average IS NULL;

SELECT \* FROM infobb02 WHERE average IS NOT NULL;

SELECT \* FROM infobb02 WHERE NOT(average IS NULL); -- Logical NOT

EXISTS NOT EXISTS

SELECT ename, sal FROM empbb02 WHERE ename LIKE ‘b%’; -- enames starting with ‘b’

SELECT ename, sal FROM empbb02 WHERE ename LIKE ‘%b%’; -- enames containing ‘b’

SELECT ename, sal FROM empbb02 WHERE ename LIKE ‘\_\_\_\_’; -- enames of length 4

Also NOT LIKE

SELECT ename, sal FROM empbb02 WHERE sal BETWEEN 35000 AND 70000;

SELECT ename, sal FROM empbb02 WHERE sal IN (40000, 42000);

Subselect

SELECT ename, sal FROM empbb02 WHERE sal < (SELECT sal FROM empbb02 WHERE ename = ‘brown’);

SELECT ename, sal, pos FROM empbb02 WHERE (sal, pos) IN (SELECT sal, pos FROM empbb02 WHERE deptno = 10);

SELECT ename, sal, pos FROM empbb02 WHERE sal <ALL (SELECT sal FROM empbb02 WHERE deptno = 30);

SELECT ename, sal, pos FROM empbb02 WHERE sal <ANY (SELECT sal FROM empbb02 WHERE deptno = 30);

Views

CREATE VIEW empview3 AS SELECT ename, sal FROM empbb02 WHERE sal > 40000;

UPDATE empview3 SET sal = 52000 WHERE sal = 50000;

Changing values in a view also changes the underlying table. Unless...

CREATE VIEW empview4 AS SELECT ename, sal FROM empbb02 WITH READ ONLY;

DROP VIEW empview3;

Other Stuff

CREATE SYNONYM jerks FOR empbb02;

RENAME empbb02 TO jerks;

Column Formatting

COLUMN capital format a10

COLUMN capital heading capital -- IDK if quotes are needed for “capital” or ‘capital’

COLUMN capital clear -- clear formatting of this column

Syntactic Sugar

.. FOREIGN KEY .. REFERENCES .. ON DELETE CASCADE

* Drops the selected FK when it prevents a table from being dropped.
* Strictly for FK only

DROP TABLE empbb02 PURGE CASCADE CONSTRAINTS

* all referential integrity constraints will be dropped as well

INSERT INTO deptbb02 VALUES(56, &n, &r, &l, &m);

* Substitution/runtime variables
* Good to use for data entry

ACCEPT num PROMPT ‘Please enter a number: ‘

INSERT INTO deptbb02 VALUES(&num, &n, &r, &l, &m);

* Single quotes around the &num if needed

Indexes

CREATE INDEX empbb02\_ename\_ndx ON empbb02(ename);

DROP INDEX empbb02\_ename\_ndx;

Sequences

A sequence is an object in Oracle that is used to generate a number sequence. This can be useful when you need to create a unique number to act as a primary key

CREATE SEQUENCE emp\_seq

MINVALUE 1

MAXVALUE 999999999999999999999999999

START WITH 1

INCREMENT BY 1

CACHE 20;

INSERT INTO emps (emp\_id, emp\_name) VALUES

(

emp\_seq.NEXTVAL,

'Katy’

);

Data Types

* Varchar2(n)
* Char(n)
* Number
* Number(n)
* Number(n, d)
* Date
* Long
* CLOB
  + Character Large Object
* RAW, LongRaw
  + Binary
* BLOB
  + Binary Large Object
* BFILE
* XML
* NCLOB
  + National Character set
  + For non-English characters
* Binary\_Integer
* Binary\_Float
* Binary\_Double

Aggregate Functions

* AVG
* SUM
* MIN
* MAX
* COUNT
  + Can do COUNT(\*)
  + All but this one ignore NULL values

Grouping Results

SELECT AVG(sal), deptno FROM empbb02 GROUP BY deptno;

-- After the keywords ‘GROUP BY’ a WHERE clause is declared as ‘HAVING’

SELECT AVG(sal), deptno FROM empbb02 GROUP BY deptno HAVING deptno <> 50;

Random Stuff

* Tools of ignorance
  + nickname for a catcher's protective equipment
* How to catch a knuckleball…
  + wait until it stops rolling and then pick it up
* Know player nicknames
* E.F. Codd
  + Invented the relational model for database management, the theoretical basis for relational databases

MIDTERM

Ways to copy tables (lab 3)

1. Oracle keyword: COPY
   1. Complex syntax
2. Export / Import
   1. Exports to a example.dmp file
   2. Can only import Oracle exported files
3. SQLLDR
   1. SQL Loader
   2. Assumes to start with CSV files

Transactions

SAVEPOINT spot1;

...

ROLLBACK TO SAVEPOINT spot1;

Commit;

PL/SQL

Pseudo Ops

* Set serveroutput on/off
  + Allows for displaying PL/SQL output

Boolean Type

* True AND NULL = NULL
* False AND NULL = False
* True OR NULL = True
* False OR NULL = NULL

End of PL/SQL Block (I don’t know why...)

.

/

Anchored Types

* %TYPE
* %ROWTYPE

Table Type Index Methods (return an index)

* First
* Last
* Next(index)
* Prior(index)

Cursors

* A set of rows from a table with an iterative method for processing

Cursor Attributes

* SQL%ROWCOUNT
* Found
* Notfound
* Isopen

Cursor Commands

* Open
* Cursor
* Fetch
* Close