Script & Sample Queries to Create & Manipulate the URISA Proceedings Database in ORACLE

/*--- Create the URISA database schema --------------- */
drop table authors;
drop table titles;
drop table keywords;
drop table match;

create table authors
(
    lastname varchar2(25),
    fnamemi varchar2(15),
    paper number(10, 0)
);

create table titles
(
    title varchar2(140),
    paper number(10, 0)
);

create table keywords
(
    code number(10, 0),
    keyword varchar2(45),
    major number(10, 0),
    category varchar2(20)
);

create table match
(
    code number(10, 0),
    major number(10, 0),
    paper number(10, 0)
);

create index apaper on authors (paper);
create index tpaper on titles (paper);
create index mpaper on match (paper);
create index cmatch on match (code);
create index kcode on keywords (code);

/*--- Authors ------------------------------------------------------*/
drop view v1hold;
create view v1hold as
select lastname, fnamemi, count(*) papers
from authors
group by lastname, fnamemi;
/* Alphabetical list of authors & count of papers */
select * from v1hold
order by lastname;

/* Authors sorted by number of papers */
select fnamemi, lastname, papers from v1hold
where papers > 1
order by papers DESC, lastname ASC, fnamemi ASC;

/* Alphabetical list of authors & papers */
select distinct lastname, fnamemi, paper
from authors
where lastname like 'FER%'
order by lastname, fnamemi, paper asc;

/*--- Keywords ------------------------------------------------------*/
drop view v1hold1;
create view v1hold1 as
select k.code, keyword, count(*) papers
from keywords k, match m
where k.code = m.code
group by k.code, keyword;
/* alphabetical list of keywords & count of papers*/
select distinct code, keyword, papers from v1hold1
order by code, keyword;
/* keywords sorted by count of papers*/
select distinct code, keyword, papers from v1hold1
where papers > 10
order by papers desc, code, keyword;
/* count of papers using *EITHER* of two particular keywords */
select code, count(*) papers
from match
where code = 106
or code = 229
group by code;

/* count of papers using *BOTH* of two particular keywords. Doing 'AND' combinations is trickier since any one row in the 'match' table associates a paper with a single keyword. Here's a 'self-join' that finds all papers associated with both keyword code 106 AND keyword 229 */
select m.code, n.code, count(*) papers
from match m, match n
where m.paper = n.paper
and (m.code = 106 AND n.code = 229)
group by m.code, n.code;
/* Here's a variation that joins in the 'keywords' table to lookup the keyword description and then counts papers that use either code=106 OR a keyword with 'TRANS' in it. */
select m.code, keyword, count(*) papers
from match m, keywords k
where m.code = k.code and (m.code = 106
   or k.keyword like 'TRANS%')
group by m.code, k.keyword;

/* Now do the query with an *AND* condition. */
select m.code, substr(k1.keyword,1,20) keyword1,
       n.code, substr(k2.keyword,1,20) keyword2,
       count(*) papers
from match m, match n, keywords k1, keywords k2
where m.paper = n.paper and
   m.code = k1.code and
   n.code = k2.code
   and (m.code = 106 AND k2.keyword like 'TRANS%')
group by m.code, k1.keyword, n.code, k2.keyword;

/* Another AND case with a join to the titles table */
select t.paper, substr(title,1,50)
from titles t, match m, match n
where m.paper = n.paper and m.paper = t.paper
   and (m.code = 229 AND n.code = 106)
order by paper;

/****** More Complex Queries *******/

What other keywords were most frequently used for GIS-related papers?
---------------------------------------------------------------------

/* Keywords related to GIS and mapping: */
drop view v1gispapers;

CREATE view v1gispapers AS
SELECT m.code, keyword, m.paper
FROM keywords k, match m
WHERE m.code = k.code AND
   (keyword LIKE '%GIS%' OR
    keyword LIKE '%GEOGRAPHIC INFORMATION%' OR
    keyword LIKE '%MAPPING%');

/* Counts of papers using these keywords */

SELECT m.code, k.keyword, count(distinct t.paper) papers
from match m, titles t, keywords k
where m.paper = t.paper AND k.code = m.code AND
   m.code IN (select distinct code from v1gispapers)
group by m.code, k.keyword
order by m.code;

/****************************
Determine the frequency of use of OTHER keywords among those papers (in the 'tlgispapers' tables) that we've categorized as 'GIS/Mapping' papers
CREATE VIEW v1combo as
SELECT m1.paper, m1.code giscode, m2.code othercode
FROM match m1, match m2
WHERE m1.code IN (select distinct code from v1gispapers) AND
    m2.code NOT IN (select distinct code from v1gispapers) AND
    m1.paper = m2.paper;

/* The above 'v1combo' view lists every paper that used a
   'gis/mapping' keyword (i.e., codes that show up in the
   'v1gispapers' view) as well as some non-gis keywords
   (i.e., codes that do NOT show up in the 'v1gispaper' view.
   There is one row for every combination of gis and non-gis
   keyword paring. If a paper used three of the gis keywords
   and two other keywords, it would appear in this list
   3x2 = 6 times. That's why there are so many rows in combo.
   To get the list of unique 'other' keywords
   used by each paper, let's create view 'combo2' from
   the 'combo' view: */

CREATE VIEW v1combo2 AS
SELECT DISTINCT paper, othercode, keyword
FROM combo c, keywords k
WHERE c.othercode = k.code;

SELECT * from v1combo2
ORDER BY paper, othercode;

/* Okay, there were 738 rows in 'combo', 608 rows in 'combo2',
   but only 185 distinct papers appearing in each table.
   Does this may sense to you?

   Now, finally, let's determine the frequency of use
   of each of the non-GIS keywords by the 'v1gispapers'.
   119 'other' keywords are used by these papers,
   and 76 of these are used more than once. */

SELECT othercode, keyword, count(paper) papers
FROM v1combo2
GROUP BY othercode, keyword
ORDER BY papers DESC, othercode;

/* From the listing, we see a few other keywords (such
   as 909 = Land Information Systems) that are more or
   less synonyms for GIS. We could go back and redefine
   't1gispapers' to include those using these keywords
   as well. Then the identical subsequent queries from above
   could be rerun to do the revised 'othercode' analysis. */