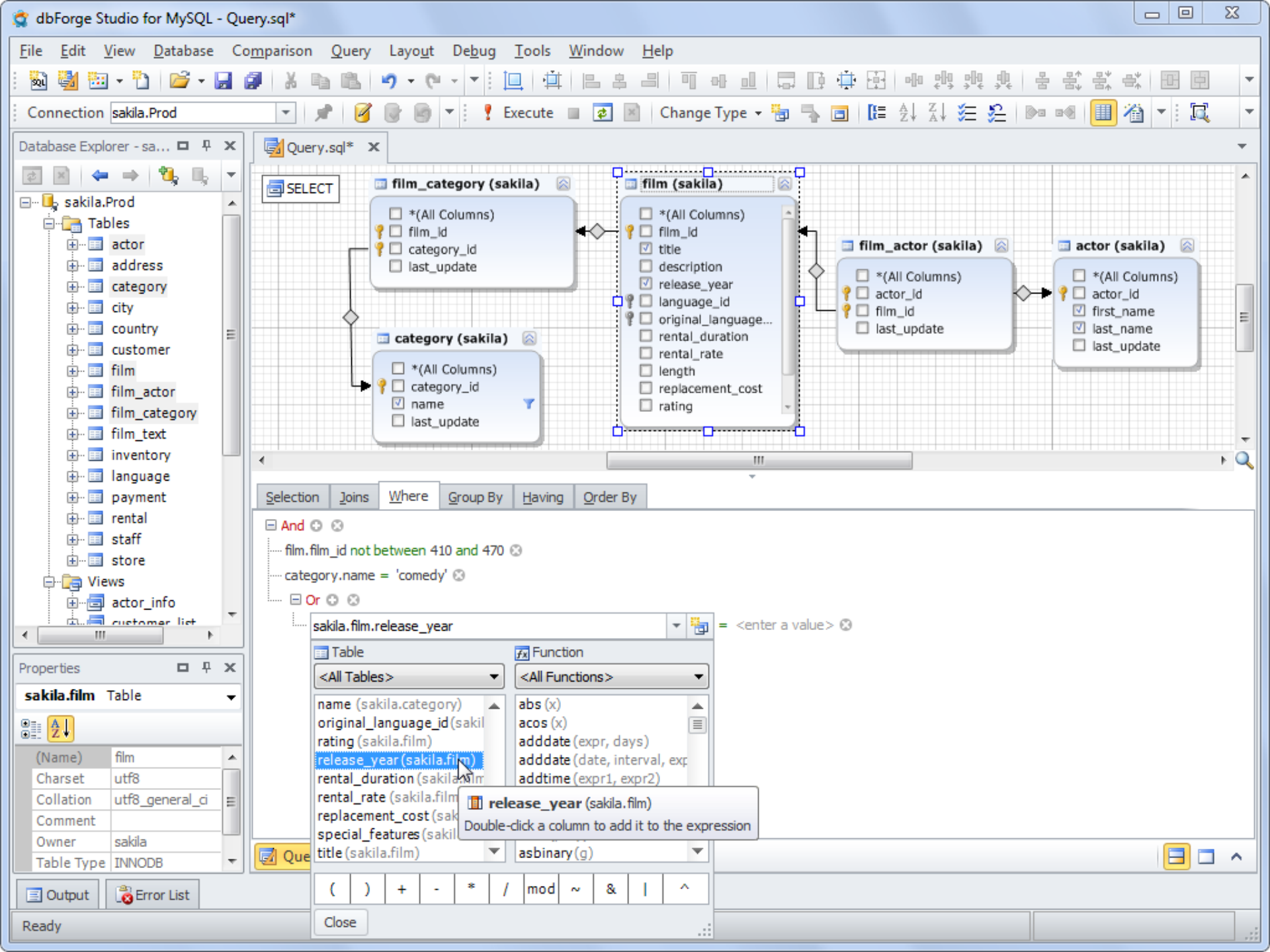
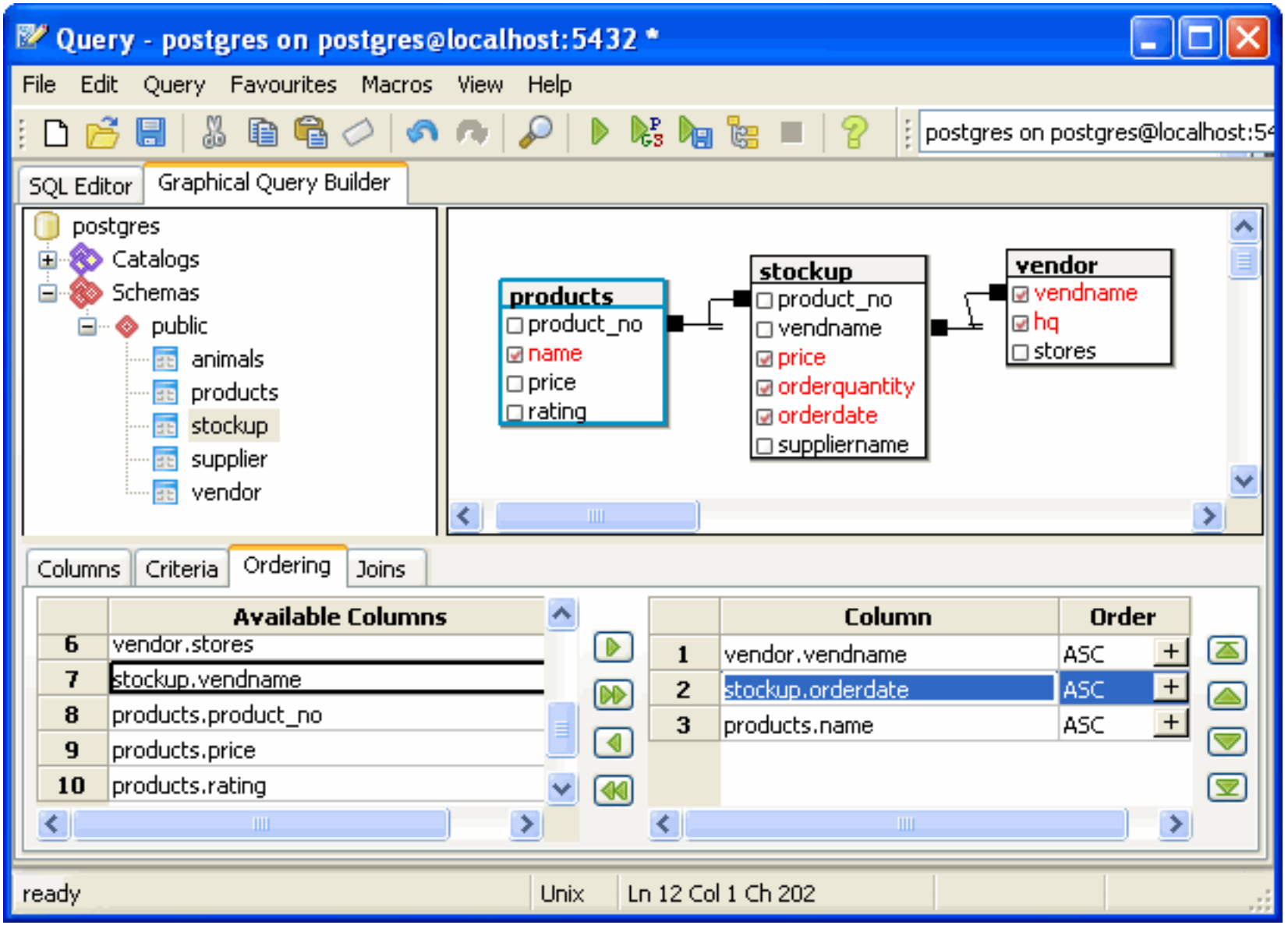
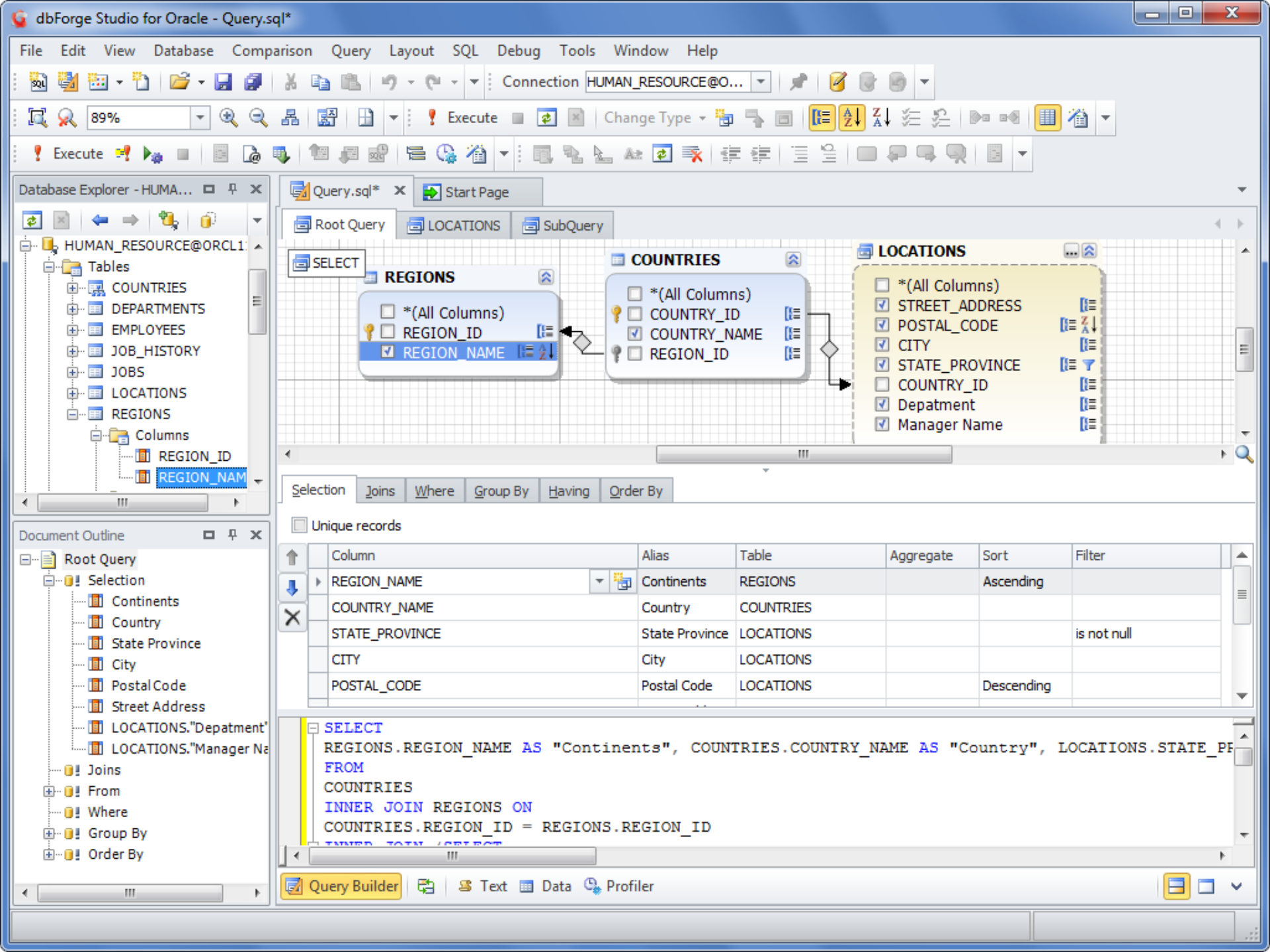
Other Query Interfaces and Languages

Davood Rafiei


dbForge Studio for MySQL - Query.sql*
File Edit View Database Comparison Query Layout Debug Tools Window Help
Connection sakila.Prod
Execute
Change Type
Database Explorer - sa...
sakila.Prod
Tables
actor
address
category
city
country
customer
film
film_actor
film_category
film_text
inventory
language
payment
rental
staff
store
Views
actor_info
Query.sql*
SELECT
film_category (sakila)
*(All Columns)
film_id
category_id
last_update
film (sakila)
*(All Columns)
film_id
title
description
release_year
language_id
original_language...
rental_duration
rental_rate
length
replacement_cost
rating
film_actor (sakila)
*(All Columns)
actor_id
film_id
last_update
actor (sakila)
*(All Columns)
actor_id
first_name
last_name
last_update
category (sakila)
*(All Columns)
category_id
name
last_update
Selection Joins Where Group By Having Order By
And
film.film_id not between 410 and 470
category.name = 'comedy'
Or
sakila.film.release_year
= <enter a value>
Table
Function
<All Tables>
<All Functions>
name (sakila.category)
original_language_id (sakil
rating (sakila.film)
release_year (sakila.film)
rental_duration (sakila.film
rental_rate (sakila.film
replacement_cost (sak
special_features (sakil
title (sakila.film)
abs (x)
acos (x)
adddate (expr, days)
adddate (date, interval, exp
addtime (expr1, expr2)
asbinary (g)
release_year (sakila.film)
Double-click a column to add it to the expression
( ) + - * / mod ~ & | ^
Close
Properties
sakila.film Table
(Name) film
Charset utf8
Collation utf8_general_ci
Comment
Owner sakila
Table Type INNODB
Output
Error List
Ready


Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
postgres on postgres@localhost:5
SQL Editor
Graphical Query Builder
postgres
Catalogs
Schemas
public
animals
products
stockup
supplier
vendor
products
product_no
name
price
rating
stockup
product_no
vendname
price
orderquantity
orderdate
suppliername
vendor
vendname
hq
stores
Columns
Criteria
Ordering
Joins
Available Columns
6 vendor.stores
7 stockup.vendname
8 products.product_no
9 products.price
10 products.rating
Column
Order
1 vendor.vendname ASC
2 stockup.orderdate ASC
3 products.name ASC
ready
Unix
Ln 12 Col 1 Ch 202


dbForge Studio for Oracle - Query.sql*
File Edit View Database Comparison Query Layout SQL Debug Tools Window Help
Connection HUMAN_RESOURCE@O...
Execute
Change Type
89%
Database Explorer - HUMA...
HUMAN_RESOURCE@ORCL1
Tables
COUNTRIES
DEPARTMENTS
EMPLOYEES
JOB_HISTORY
JOBS
LOCATIONS
REGIONS
Columns
REGION_ID
REGION_NAM
Document Outline
Root Query
Selection
Continents
Country
State Province
City
Postal Code
Street Address
LOCATIONS."Depatment"
LOCATIONS."Manager Na
Joins
From
Where
Group By
Order By
Query.sql*
Start Page
Root Query
LOCATIONS
SubQuery
SELECT
REGIONS
*(All Columns)
REGION_ID
REGION_NAME
COUNTRIES
*(All Columns)
COUNTRY_ID
COUNTRY_NAME
REGION_ID
LOCATIONS
*(All Columns)
STREET_ADDRESS
POSTAL_CODE
CITY
STATE_PROVINCE
COUNTRY_ID
Depatment
Manager Name
Selection
Joins
Where
Group By
Having
Order By
Unique records
Column
Alias
Table
Aggregate
Sort
Filter
REGION_NAME
Continents
REGIONS
Ascending
COUNTRY_NAME
Country
COUNTRIES
STATE_PROVINCE
State Province
LOCATIONS
is not null
CITY
City
LOCATIONS
POSTAL_CODE
Postal Code
LOCATIONS
Descending
SELECT
REGIONS.REGION_NAME AS "Continents", COUNTRIES.COUNTRY_NAME AS "Country", LOCATIONS.STATE_PF
FROM
COUNTRIES
INNER JOIN REGIONS ON
COUNTRIES.REGION_ID = REGIONS.REGION_ID
Query Builder
Text
Data
Profiler

Graphical Query Interfaces

- Oracle SQL query builder
- MySQL query builder
- SQL Server visual query builder
- Postgress query builder
- SQLite query builder
- Microsoft Access query interface

Roots

- Based on relational calculus
  - So is SQL
- Declarative
- Known as Query by Example (QBE)
- Limited expressive power

QBE: Query by Example

- Declarative query language, like SQL
- Developed in 1970s at IBM
- Based on DRC
- Visual
- Other visual query languages (MS Access, Paradox) are just incremental improvements

QBE - Examples

Q1. Print all professors' names in the Math department

Professor	*Id*	*Name*	*DeptId*
		P.	Math

Q2. Print all professors' names who taught c291 in Fall 2002.

Professor	*Id*	*Name*	*DeptId*
	_123	**P.**	

Teaching	*ProfId*	*CrsCode*	*Semester*
	_123	C291	F2002

Condition Boxes

- Some conditions are too complex to be placed directly in table columns

Transcript	*StudId*	*CrsCode*	*Semester*	*Grade*
	P.	CS532		_Gr

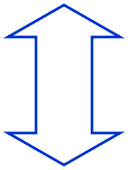
Conditions
_Gr = ‘A’ OR _Gr = ‘B’

- Students who took CS532 & got A or B

Connection to Rel. Calculus

- A graphical representation of DRC

Transcript	*StudId*	*CrsCode*	*Semester*	*Grade*
	_123	_CS532	F2002	A



DRC: Transcript(x, y, ‘F2002’, ‘A’)

TRC: Transcript(t) AND t.Semester=‘F2002’ AND t.Grade = ‘A’

Relational Calculus

- Two flavors
  - Tuple Relational Calculus (TRC)
    - Variables range over tuples (e.g. SQL)
  - Domain Relational Calculus (DRC)
    - Variables range over domains (e.g. QBE)
- Query ~ formula
- Answer ~ an assignment that makes the formula true

Examples

TRC:

{t | Transcript(*t*) AND t.Semester='F2002' AND t.Grade = 'A'}

{s | Students(s) AND ∃e∈Enroll (s.sid = e.sid AND e.cid='291')}

{s | Students(s) AND ∃c∈Courses (c.instructor='John Smith'
==> ∃e∈Enroll (e.sid = s.sid AND e.cid=c.cid))}

DRC:

{x,y | Transcript(*x*, *y*, 'F2002', 'A')}

Relational Calculus

a base for other query languages

Relation Between TRC and SQL

- List the names of all professors who have taught MGT123
  - In TRC:

    {p.*Name* | Professor(p) AND ∃t ∈ Teaching
    (p.*Id* = t.*ProfId* AND t.*CrsCode* = 'MGT123') }

  - In SQL:

```
SELECT   p.Name
FROM     Professor p, Teaching t
WHERE    p.Id = t.ProfId  AND  t.CrsCode = 'MGT123'
```

Core of SQL is merely a syntactic sugar on top of TRC

What Happened to Quantifiers in SQL?

- SQL has no quantifiers: how come? It uses conventions:
  - *Convention* 1. Universal quantifiers are not allowed (but SQL:1999 introduced a limited form of explicit ∀)
  - *Convention* 2. Make existential quantifiers *implicit*: Any tuple variable that does not occur in SELECT is assumed to be implicitly quantified with ∃
- Compare:

{p.*Name* | Professor(p) AND ∃t ∈Teaching … }

and

```
SELECT   P.Name
FROM     Professor p, Teaching t
… … …
```

Implicit ∃

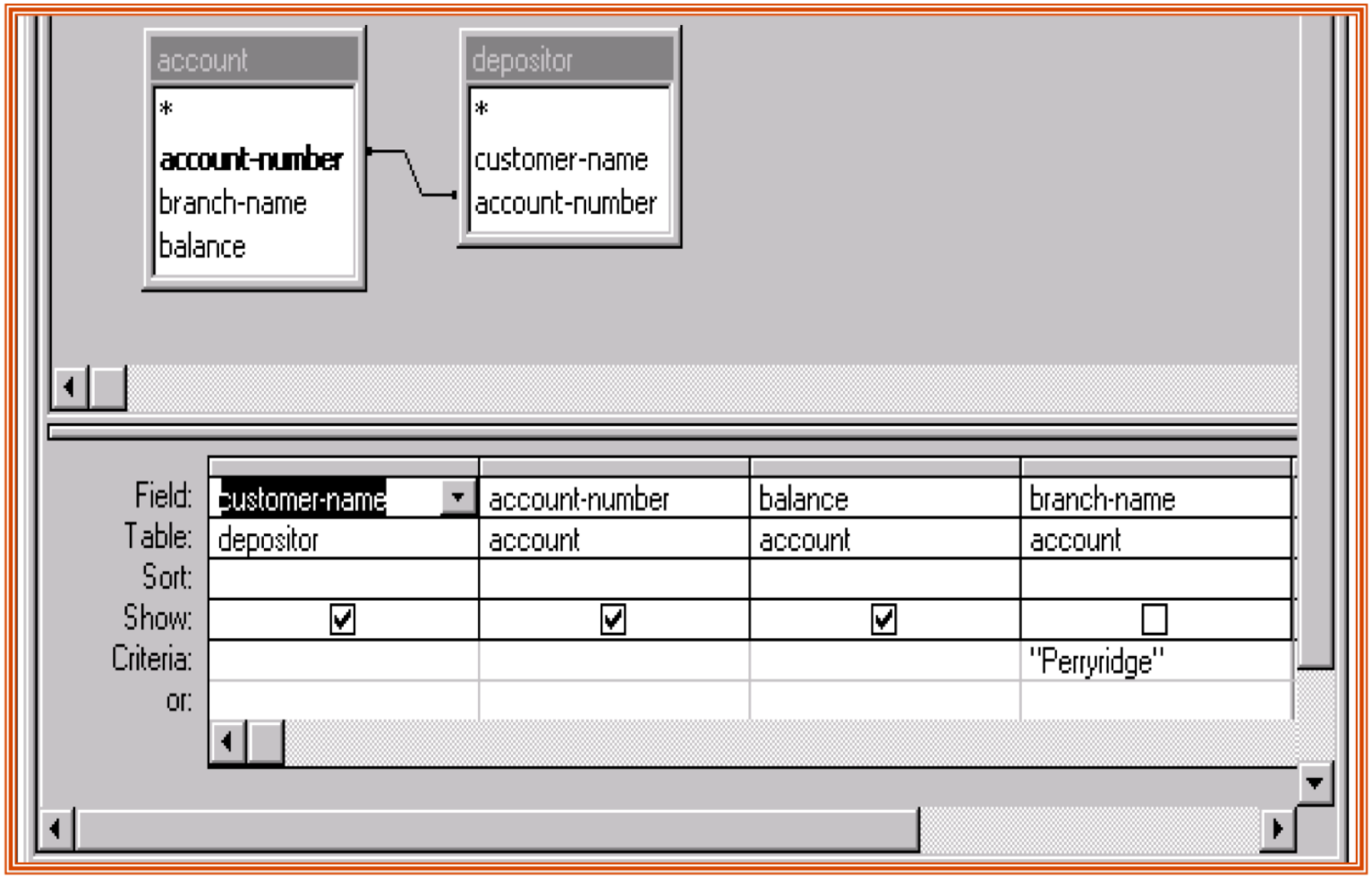
Relation Between TRC and SQL (cont'd)

- SQL uses a subset of TRC with simplifying conventions for quantification
- Restricts the use of quantification and negation (so TRC is more general in this respect)
- SQL uses aggregates, which are absent in TRC (and relational algebra, for that matter). But aggregates can be added
- SQL is extended with relational algebra operators (MINUS, UNION, JOIN, etc.)
  - This is just more syntactic sugar, but it makes queries easier to write

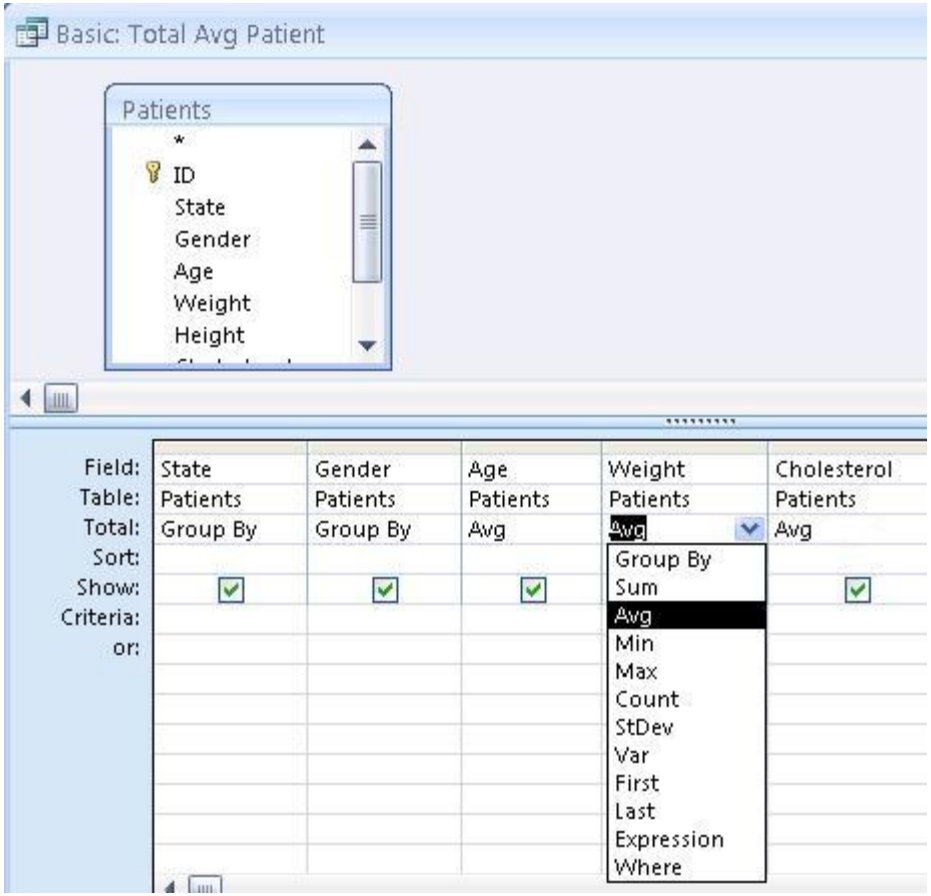
Graphical Interfaces

Of PC Databases

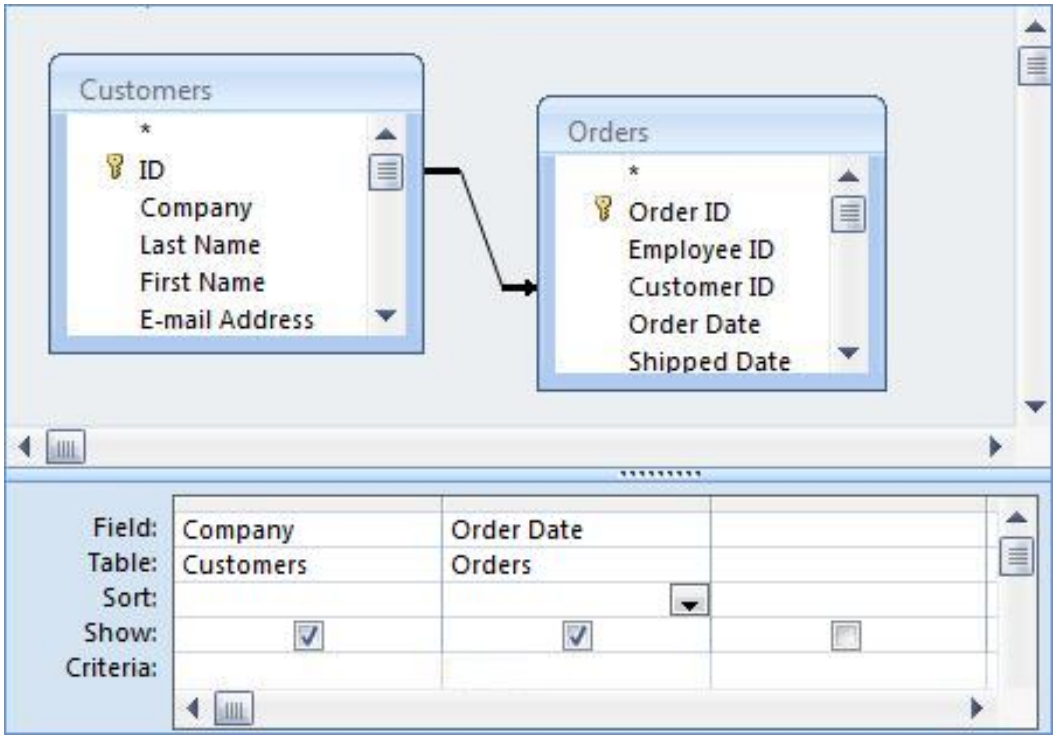
Microsoft Access



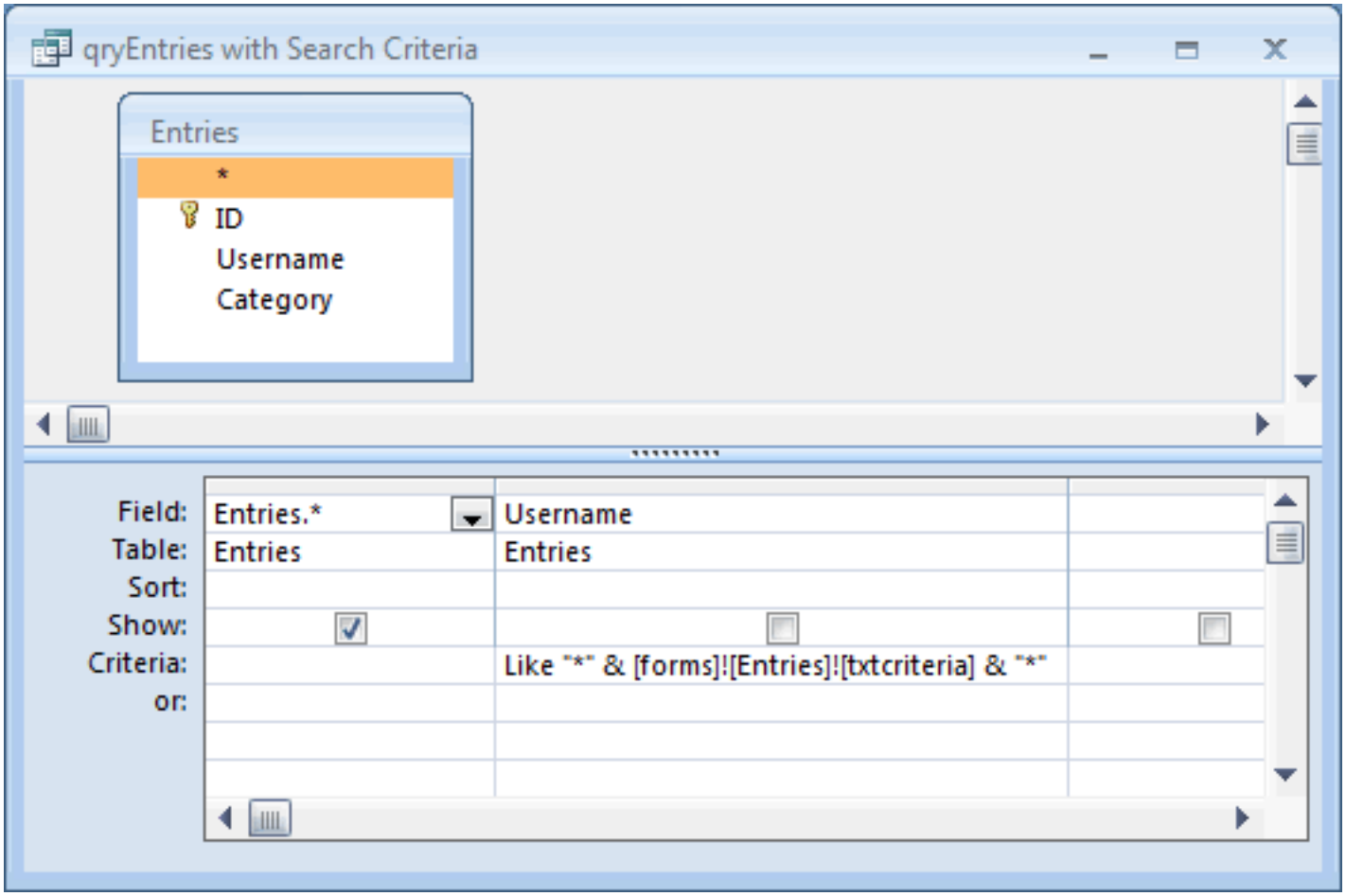
Microsoft Access (Cont)



Microsoft Access (Cont)



Microsoft Access (Cont)



PC Databases

- A spruced up version of QBE (better interface)
- Be aware of implicit quantification
- Beware of negation pitfalls
  - Sec. 13.4 gives some of the pitfalls under the heading “the price of free lunch”