ISO
Query By Example

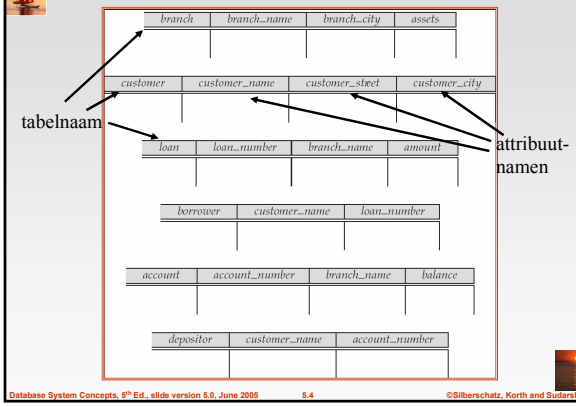
Prof. dr. Paul De Bra

Gebaseerd op:
Database System Concepts, 5th Ed.
©Silberschatz, Korth and Sudarshan

QBE — waarom nog een query taal?

- de relationele algebra en SQL geven niet alleen een specificatie van een query-resultaat, maar ook een (mogelijk) recept om de query te berekenen
- in QBE geven we alleen een specificatie van het resultaat, en geen recept voor de berekening
- QBE is "grafisch": je *tekent* hoe het resultaat van de query er moet uit zien en wat met wat moet overeenkomen
- **je geeft als het ware een *voorbeeld* van het resultaat**

QBE "skelet" voor de bank database:



Queries over 1 tabel

- geef alle nummers van leningen die geopend zijn bij het filiaal Perryridge.

loan	loan_number	branch_name	amount
	P._x	Perryridge	

  - _x is een variabele (is in bovenstaande query nog niet nodig)
  - P. betekent print (toon op scherm)
  - het resultaat is een verzameling: mogelijke dubbele rijen worden automatisch verwijderd
  - om de dubbels te behouden: gebruik P.ALL

loan	loan_number	branch_name	amount
	P.ALL.	Perryridge	

Queries op 1 tabel (cont.)

- toon de loan tabel (met alle attributen):
  - Methode 1:

loan	loan-number	branch-name	amount
	P._x	P._y	P._z

  - Methode 2: kortschrift (vergelijk met * in SQL)

loan	loan_number	branch_name	amount
P.			

Queries op 1 tabel (cont.)

- geef de nummers van alle leningen met een bedrag hoger dan 700

loan	loan_number	branch_name	amount
	P.		>700

- geef de namen van alle filialen die **niet** in Brooklyn gevestigd zijn

branch	branch_name	branch_city	assets
	P.	¬ Brooklyn	

Queries op 1 tabel (cont.)

- geef de nummers van de leningen die Smith en Jones samen hebben aangegaan

borrower	*customer_name*	*loan_number*
	Smith	P._x
	Jones	_x

- geef alle klanten die in dezelfde stad wonen als Jones

customer	*customer_name*	*customer_street*	*customer_city*
	P._x		_y
	Jones		_y

Queries meer dan 1 tabel

- geef de namen van alle klanten die een lening hebben bij het filiaal Perryridge

loan	*loan_number*	*branch_name*	*amount*
	_x	Perryridge	

borrower	*customer_name*	*loan_number*
	P._y	_x

Queries op meer dan 1 tabel (cont.)

- geef de namen van de klanten die een rekening en een lening hebben

depositor	*customer_name*	*account_number*
	P._x	

borrower	*customer_name*	*loan_number*
	_x	

De “not exists” in QBE

- geef de namen van alle klanten die een rekening hebben maar geen lening

depositor	*customer_name*	*account_number*
	P._x	

borrower	*customer_name*	*loan_number*
¬	_x	

¬ betekent dus “er bestaat geen”

“verschillend van” in SQL

- geef alle klanten met twee of meer rekeningen

depositor	*customer_name*	*account_number*
	P._x	_y
	_x	¬ _y

¬ betekent hier “verschillend van”

De “Condition Box”

- soms wil je een conditie (zoals in de “where” in SQL) die niet (makkelijk) kan uitgedrukt worden binnen een tabel-skelet
- complexe condities kunnen in een “condition box” worden geplaatst (er mogen verschillende “condition boxes” worden gebruikt)
- voorbeeld: geef het nummer van de leningen van Smith, Jones of beide samen

borrower	*customer_name*	*loan_number*
	_n	P._x

conditions

_n = Smith **or** _n = Jones

Condition Box (cont.)

- QBE laat een “vereenvoudigde” syntax toe voor het uitdrukken van logische expressies

branch	branch_name	branch_city	assets
	P.	_x	

conditions
_x = (Brooklyn **or** Queens)

Condition Box (cont.)

- geef de nummers van alle rekeningen met een saldo van minstens 1300 en hoogstens 1500

account	account_number	branch_name	balance
	P.		_x

conditions
_x ≥ 1300
_x ≤ 1500

- geef de nummers van alle rekeningen met een saldo van minstend 1300 en hoogstens 2000 maar niet exact 1500

account	account_number	branch_name	balance
	P.		_x

conditions
_x = (≥ 1300 **and** ≤ 2000 **and** ¬ 1500)

Condition Box (cont.)

- bij de volgende vraag is de condition box echt nodig: geef alle filialen die grotere assets hebben dan tenminste 1 filiaal uit Brooklyn

branch	branch_name	branch_city	assets
	P._x	Brooklyn	_y _z

conditions
_y > _z

de resultaat (result) tabel

- Soms is het antwoord op een query niet een deel van een van de bestaande tabellen, maar samengesteld uit verschillende tabellen:
- geef de *customer_name*, *account_number*, en *balance* voor alle rekeningen bij het filiaal Perryridge
  - wat moeten we doen:
    - de *depositor* en *account* tabel verbinden (“join”)
    - projecteren op *customer_name, account_number* en *balance*.
  - dit kan alleen maar door:
    - een tabel-skelet te maken met de structuur van het resultaat: met attributen *customer_name, account_number,* en *balance*.
    - de query op te schrijven in de bestaande tabel-skeletten en het nieuwe skelet.

de resultaat (result) tabel (cont.)

- de query wordt dan:

account	account_number	branch_name	balance
	_y	Perryridge	_z

depositor	customer_name	account_number
	_x	_y

result	customer_name	account_number	balance
P.	_x	_y	_z

aggregatie functies (zoals bij SQL)

- de aggregatie functies zijn AVG, MAX, MIN, SUM, en CNT
- omdat QBE standaard altijd dubbels uit resultaten verwijdert moeten de operatoren de toevoeging ALL krijgen om dit te voorkomen, dus SUM.ALL._x of AVG.ALL._x, etc.
- voorbeeld: geef de som van alle saldi van de rekeningen bij het filiaal Perryridge

account	account_number	branch_name	balance
		Perryridge	P.SUM.ALL.

aggregatie functies (cont.)

- UNQ geeft expliciet aan dat dubbels worden verwijderd (dit is standaard maar het is duidelijker om het toch te vermelden)
- geef het totaal aantal klanten met een rekening:

depositor	*customer_name*	*account_number*
	P.CNT.UNQ.	

aggregatie en group-by

- maak een lijst van alle filialen met per filiaal het gemiddelde saldo op de rekeningen bij dat filiaal

account	*account_number*	*branch_name*	*balance*
		P.G.	P.AVG.ALL._x

- de "G" in "P.G" betekent dat gegroepeerd wordt per filiaal *(branch_name)*
- de "ALL" in "P.AVG.ALL" verzekert dat dubbels niet worden verwijderd
- om alleen filialen te krijgen waar het gemiddelde saldo hoger is dan 1200 kan je een condition box toevoegen

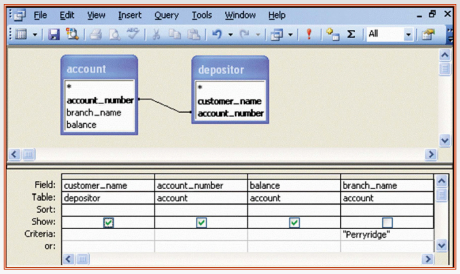
conditions
AVG.ALL._x > 1200

Microsoft Access QBE

- Microsoft Access biedt een variant (slap aftreksel) van QBE aan: Graphical Query By Example (GQBE)
- GQBE verschilt van "echte" QBE onder andere op volgende punten:
  - in de "design view" geven de "skeletten" de lijst van attributen verticaal weer in plaats van horizontaal
  - lijnen in de view worden gebruikt om aan te duiden dat waarden (voor bepaalde attributen) gelijk moeten zijn
    - een aantal lijnen worden automatisch gegenereerd, voor gelijknamige attributen (je moet ze weer weghalen als ze ongewenst zijn)
    - andere verbanden (ongelijk, groter, kleiner) kunnen niet met lijnen worden uitgedrukt, maar moeten in een "design grid" (een soort uitgebreide condition box)
  - de **design grid** is niets meer of minder dan een plek om SQL **where**, **group by** en **having** condities in te vullen, en met een vinkje de attributen voor de "select" aan te duiden

Access QBE voorbeeld

- geef de *customer_name*, *account_number* en *balance* voor alle rekeningen bij het filiaal Perryridge



een Aggregatie Query in Access QBE

- geef de *name*, *street* en *city* van alle klanten die meer dan 1 rekening hebben bij de bank