

# Data Advanced

# H2. Database objects

Koen Bloemen

Sander De Puydt



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)



## H2. Database Objects

Iets dat data vasthoudt en kan manipuleren:

- Sequence
- Index
- Synonym
- ... (Er zijn er nog veel meer)

# Sequence

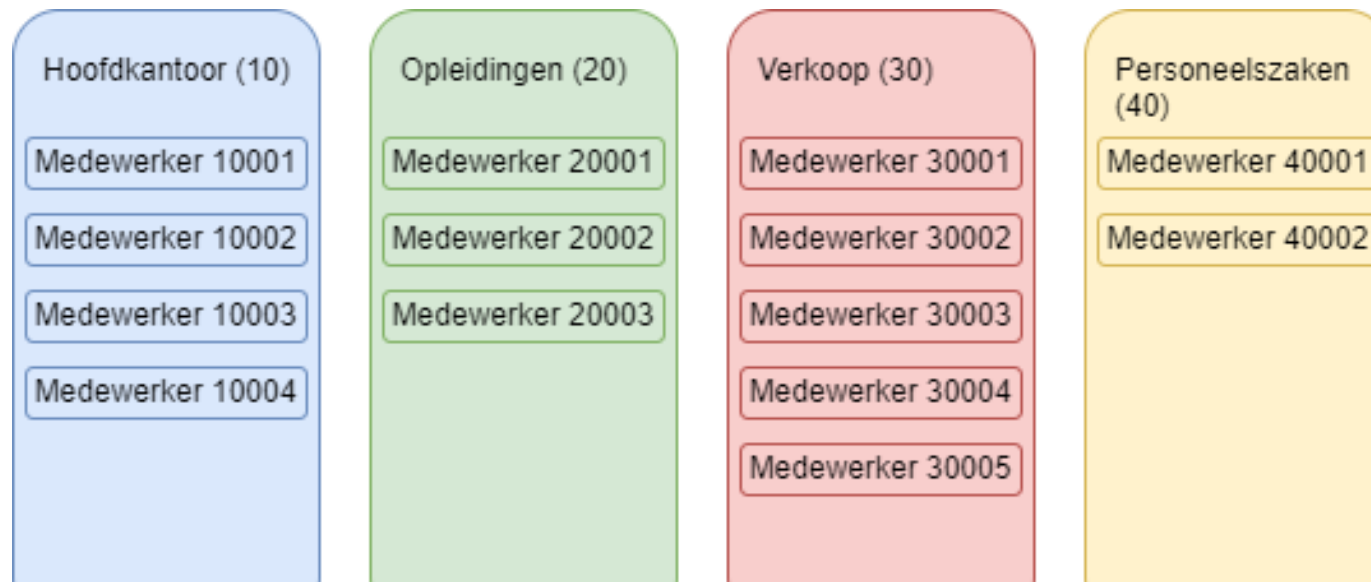
- Doel
- CREATE Syntax
- ALTER Syntax
- DROP Syntax
- Testing





# Sequence – Doel

- Use case: We willen een patroon gebruiken
- We willen de medewerkers codes geven op basis van hun afdeling. Elk van de medewerkers hebben unieke codes nodig die op elkaar opvolgen. Hoe zorgen we er voor dat we deze codes kunnen oproepen zonder de tabellen te moeten uitlezen (en zonder conflicten)?



# Sequence – Doel

- Met een Sequence kunnen we er voor zorgen dat elke rij een unieke waarde heeft.
- We kunnen zelf bepalen wat het patroon is dat de Sequence moet volgen.
  - Dalend / Stijgend
  - In sprongen
  - Min / Max
- We kunnen inzicht krijgen in het verloop van de sequence tijdens het gebruik ervan door de huidige waarde op te vragen.
- Sequences kunnen gebruikt worden onafhankelijk van een tabel

# Sequence – CREATE Syntax

```
CREATE SEQUENCE sequence_name  
START WITH initial_value  
INCREMENT BY increment_value  
MINVALUE minimum value  
MAXVALUE maximum value  
CYCLE|NOCYCLE;
```

# Sequence – CREATE Syntax

*START WITH* *initial\_value*

- Eerste waarde die gegenereerd wordt door de sequence
- Let op: Is **niet de waarde** waar de **cyclus** naar toe gaat, indien de sequence **herhaald** wordt.

*INCREMENT BY* *increment\_value*

- Bepaalt het interval tussen twee getallen in een sequence
- Default = 1

# Sequence – CREATE Syntax

*MINVALUE* *minimum value*

- *MINVALUE* moet kleiner of gelijk zijn aan *START WITH*
- *MINVALUE* moet kleiner zijn dan *MAXVALUE*

*MAXVALUE* *maximum value*

- *MAXVALUE* moet groter of gelijk zijn aan *START WITH*
- *MAXVALUE* moet groter zijn dan *MINVALUE*

*CYCLE|NOCYCLE* ;

- *CYCLE*: geeft aan dat de sequence getallen blijft genereren na het bereiken van een limiet
- *Default = NOCYCLE*



# Sequence – CREATE Syntax

Voorbeeld:

```
CREATE SEQUENCE numbers_seq  
START WITH 1  
INCREMENT BY 1;
```

# Sequence – ALTER Syntax

```
ALTER SEQUENCE sequence_name  
START WITH initial_value  
INCREMENT BY increment_value  
MINVALUE minimum value  
MAXVALUE maximum value  
RESTART  
CYCLE|NOCYCLE;
```

# Sequence – ALTER Syntax

MINVALUE *minimum value*

MAXVALUE *maximum value*

- Oracle controleert of een nieuwe MINVALUE of MAXVALUE in conflict is met het huidige sequence number

## RESTART

- Gebruik RESTART om de NEXTVAL in te stellen op MINVALUE voor een ascending sequence en op MAXVALUE voor een descending sequence.
- Indien je wil herstarten op een ander getal, dan **combineer** je RESTART met START WITH

# Sequence – ALTER Syntax

Voorbeeld:

```
ALTER SEQUENCE numbers_seq  
START WITH 100  
INCREMENT BY -1  
MINVALUE 0  
RESTART  
CYCLE;
```



# Sequence – DROP Syntax

*DROP SEQUENCE `sequence_name`;*

Voorbeeld:

*DROP SEQUENCE numbers\_seq;*

# Sequence – Testing

- Hoe kunnen we onze sequence uitproberen?
  - **DUAL tabel**
    - De DUAL tabel zit standard in je Oracle Database en kan gebruikt worden door elke USER.
    - De DUAL tabel heeft één kolom, DUMMY, met één rij met de waarde “X”.
    - De DUAL tabel wordt gebruikt om constante expressions met het SELECT statement uit te voeren. Aangezien er maar één rij is, wordt de constante waarde slechts één keer teruggegeven. (*Probeer het uit!*)
  - nextval
  - currval

```
SQL> SELECT * FROM DUAL;
```

```
D  
-  
X
```

# Sequence – Testing

- Hoe kunnen we onze sequence uitproberen?
  - DUAL tabel
  - **nextval**
    - Geeft de eerstvolgende waarde in de sequence terug.
    - Elke keer je de **nextval** oproept wordt de **current value** van de sequence aangepast (ongeacht of je de waarde gebruikt hebt in een insert statement).
  - **currval**
    - Geeft de huidige waarde terug van de sequence.

Voorbeeld:

*SELECT numbers\_seq.currval FROM DUAL;*

```
SQL> SELECT numbers_seq.currval FROM DUAL;
```

```
  CURRVAL
```

```
-----  
      53
```

# Index

- Doel
- CREATE Syntax
- ALTER Syntax
- DROP Syntax
- Testing
- Extra



## TABLE OF CONTENTS.

| MAPS.                                    | INDEX.   | MAPS.  | INDEX.   |
|--|----------|--|----------|
| Abyssinia.....                           | 340      | Montana.....                                 | 210, 211 |
| Africa.....                              | 342, 343 | New Brunswick.....                           | 248, 249 |
| Alabama.....                             | 162, 163 | New Hampshire.....                           | 10, 11   |
| Alaska.....                              | 372, 373 | New York.....                                | 26, 27   |
| Algoma.....                              | 388, 389 | New Mexico.....                              | 198, 199 |
| Arizona.....                             | 194, 195 | New Jersey.....                              | 48, 49   |
| Arkansas.....                            | 174, 175 | Norfolk.....                                 | 116, 117 |
| Asia.....                                | 330, 331 | Nevada.....                                  | 302, 303 |
| Australia.....                           | 341      | North America.....                           | 148, 149 |
| Austria.....                             | 323      | North Carolina.....                          | 264, 265 |
| Belgium.....                             | 318      | Northwest Territory.....                     | 248, 249 |
| British Columbia.....                    | 268, 269 | Nova Scotia.....                             | 252, 253 |
| California.....                          | 208, 207 | Nova Scotia, Large Scale, Western Half.....  | 256, 257 |
| Central America.....                     | 293      | Nova Scotia, Large Scale, Eastern Half.....  | 317      |
| Central Asia.....                        | 332      | Norway.....                                  | 340      |
| China.....                               | 335      | Nubia.....                                   | 70, 71   |
| Colorado.....                            | 120, 121 | Ohio.....                                    | 192      |
| Connecticut.....                         | 18       | Oklahoma.....                                | 334, 335 |
| Cuba.....                                | 290      | Ontario.....                                 | 330, 331 |
| Dakota.....                              | 132, 133 | Oregon.....                                  | 338      |
| Delaware.....                            | 65       | Palestine.....                               | 44, 45   |
| Denmark.....                             | 318      | Pennsylvania.....                            | 36, 37   |
| East Indies.....                         | 336, 337 | Pennsylvania, Large Scale, Western Half..... | 40, 41   |
| Egypt.....                               | 340      | Pennsylvania, Large Scale, Eastern Half..... | 333      |
| England.....                             | 312      | Persia.....                                  | 320      |
| Europe.....                              | 314, 315 | Portugal.....                                | 240, 241 |
| Florida.....                             | 168, 169 | Prince Edward Island.....                    | 14, 15   |
| France.....                              | 321      | Quebec.....                                  | 329      |
| Georgia.....                             | 154, 155 | Rhode Island.....                            | 291      |
| Germany.....                             | 324, 325 | Russia.....                                  | 316      |
| Greece.....                              | 318      | San Domingo.....                             | 301      |
| Holland.....                             | 322, 323 | Scotland.....                                | 148, 149 |
| Illinois.....                            | 84, 85   | South America.....                           | 320      |
| Illinois, Large Scale Southern Half..... | 86, 87   | South Carolina.....                          | 317      |
| Illinois, Large Scale Northern Half..... | 90, 91   | Sweden.....                                  | 322      |
| Indiana.....                             | 82, 83   | Switzerland.....                             | 142, 143 |
| Indian Empire.....                       | 334      | Tennessee.....                               | 186, 187 |
| Indian Territory.....                    | 190, 191 | Texas.....                                   | 178, 179 |
| Iowa.....                                | 100, 101 | Texas, Large Scale, Western Half.....        | 182, 183 |
| Ireland.....                             | 313      | Texas, Large Scale, Eastern Half.....        | 327      |
| Islands in the Atlantic Ocean.....       | 308      | Turkey in Europe.....                        | 339      |
| Islands in the Pacific Ocean.....        | 309      | Turkish Empire.....                          | 305, 307 |
| Italy.....                               | 326      | United States.....                           | 214, 215 |
| Jamaica.....                             | 291      | Utah.....                                    | 10, 11   |
| Japan.....                               | 335      | Vermont.....                                 | 60, 61   |
| Kansas.....                              | 112, 113 | Virginia.....                                | 52, 53   |
| Kentucky.....                            | 142, 143 | Virginia, Large Scale, Western Half.....     | 56, 57   |
| Louisiana.....                           | 170, 171 | Virginia, Large Scale, Eastern Half.....     | 226, 227 |
| Maine.....                               | 6, 7     | Washington Territory.....                    | 294, 295 |
| Manitoba.....                            | 260, 261 | West Indies.....                             | 60, 61   |
| Maryland.....                            | 64, 65   | West Virginia.....                           | 52, 53   |
| Massachusetts.....                       | 14, 15   | West Virginia, Large Scale.....              | 124, 125 |
| Mexico.....                              | 298, 299 | Wisconsin.....                               | 310, 311 |
| Michigan.....                            | 26, 27   | World.....                                   | 218, 219 |
| Mississippi.....                         | 166, 167 | Yellowstone Park.....                        | 212      |
| Missouri.....                            | 106, 107 |  |          |
| Minnesota.....                           | 128, 129 |  |          |

## MAPS OF CITIES.

|                  | PAGE. |                    | PAGE.    |
|------------------|-------|--------------------|----------|
| Atlanta.....     | 288   | Minneapolis.....   | 305      |
| Baltimore.....   | 270   | Montreal.....      | 255      |
| Boston.....      | 267   | New Orleans.....   | 281      |
| Brooklyn.....    | 266   | New York.....      | 282, 283 |
| Chicago.....     | 274   | Philadelphia.....  | 286, 287 |
| Cincinnati.....  | 275   | Portland.....      | 280      |
| Cleveland.....   | 279   | St. Louis.....     | 276      |
| Columbus.....    | 278   | St. Paul.....      | 304      |
| Detroit.....     | 280   | San Francisco..... | 292      |
| Kansas City..... | 284   | Toronto.....       | 254      |
| Milwaukee.....   | 285   | Washington.....    | 271      |

## DIAGRAMS.

|  | PAGE. |   | PAGE. |
|--|-------|---|-------|
| Areas of the principal countries of the world..... | 365   | Orchard products of the United States.....                | 367   |
| Areas of lakes, seas and oceans.....               | 365   | Population of principal countries.....                    | 365   |
| Average density of population.....                 | 365   | Public lands of the United States.....                    | 369   |
| Armies of the world.....                           | 370   | Railroads in the United States, Canada and the world..... | 368   |
| Commerce of different countries.....               | 366   | Religious sects of the world.....                         | 368   |
| Debts of different nations.....                    | 366   | Revenues of nations.....                                  | 366   |
| Expenditures of nations.....                       | 366   | Silver product of the United States.....                  | 367   |



# Index – Doel

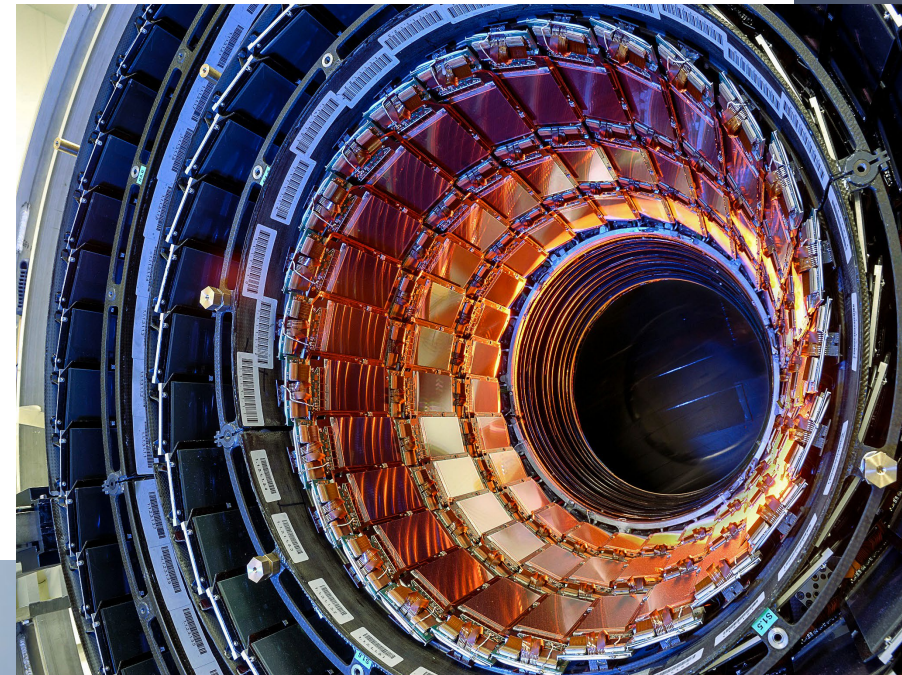
- Use case:
- Ik heb honderdduizenden films en wil weten **welke + hoeveel** films zijn **uitgekomen in 1994** en oorspronkelijk in het **Frans** zijn geproduceerd en ik wil ze **gesorteerd** zien in de ranking van IMDB.
- Grote datasets, zoals IMDB, eisen veel tijd voor een query.
  - We kunnen de query sneller maken met een Index

The IMDb logo is displayed in a bold, black, sans-serif font. It is centered within a solid yellow rectangular background that occupies the bottom right portion of the slide.

**IMDb**

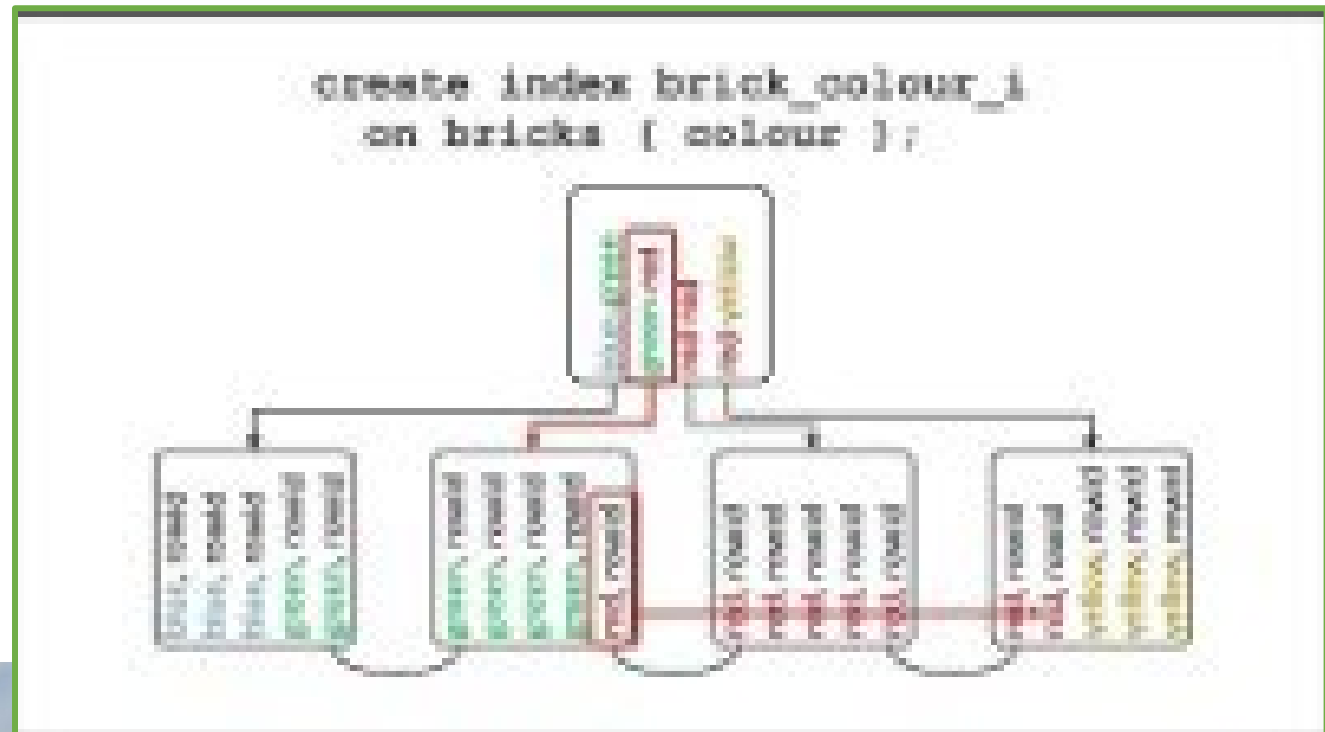
# Index – Doel

- Databases kunnen bijzonder groot worden.
  - Large Hadron Collider in CERN genereert [gigantische hoeveelheden aan data](#)
  - $10^9$  collisions/s x 1 Mbyte/collision =  $10^{15}$  bytes/s = 1 PB/s (1 Petabyte/second)
- Hoe groter de database, hoe langer de zoektijd van een query wordt.
- Met een Index kunnen we onze database performant maken.



# Index – Doel

- Duidelijke video over werking van indexen in databases
  - Geeft meer uitleg over multi-kolom indexen
  - Extra: geeft uitleg over function-indexen
- Video **voor thuis** —→



# Index – Doel

- Hoe werkt een index?
- Zonder index de naam 'Zack' opzoeken in een tabel.
  - Eén per één vergelijken
  - Overloopt volledige tabel
  - (full scan)
  - Traag

```
SELECT * FROM friends WHERE name = 'Zack';
```

| friends |        |               |
|---------|--------|---------------|
| id      | name   | city          |
| 1       | Matt   | San Francisco |
| 2       | Dave   | Oakland       |
| 3       | Andrew | Blacksburg    |
| 4       | Todd   | Chicago       |
| 5       | Blake  | Atlanta       |
| 6       | Evan   | Detroit       |
| 7       | Nick   | New York City |
| 8       | Zack   | Seattle       |



# Index – Doel

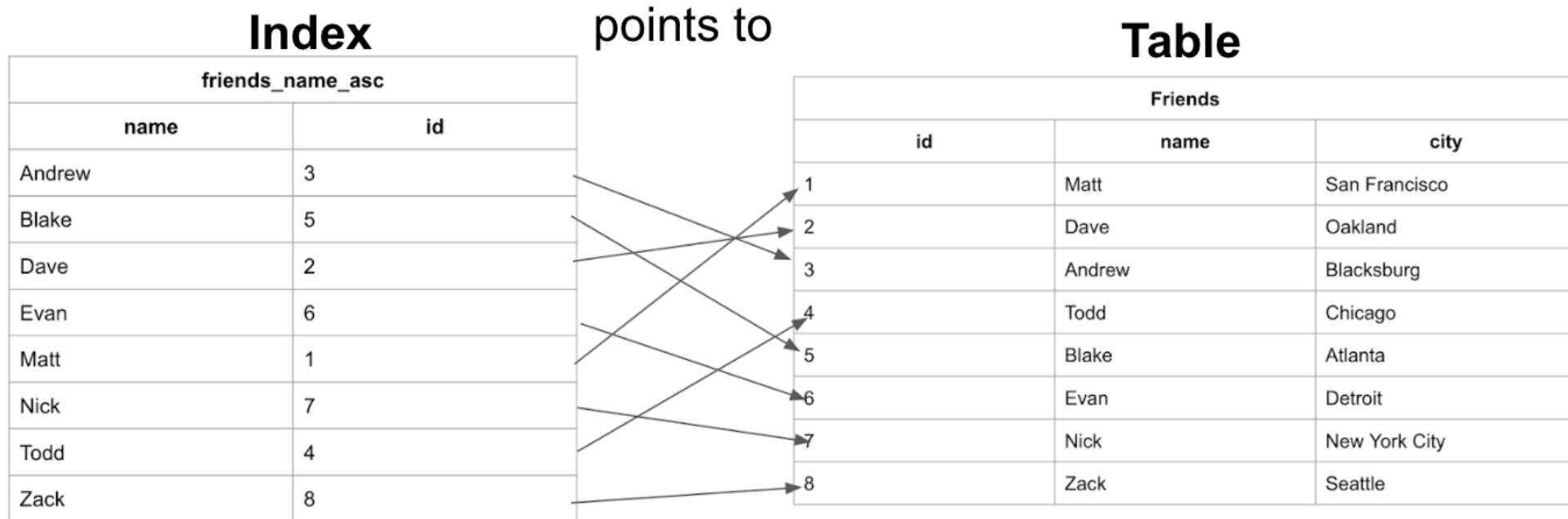
- Hoe werkt een index?
- Met index de naam 'Zack' opzoeken in een tabel.
  - Slaat veel records over
  - Geen full scan
  - Snel
  - ... Maar hoe?

```
SELECT * FROM friends WHERE name = 'Zack';
```

| friends_name_asc |       |
|------------------|-------|
| Name             | Index |
| Andrew           | 3     |
| Blake            | 5     |
| Dave             | 2     |
| Evan             | 6     |
| Matt             | 1     |
| Nick             | 7     |
| Todd             | 4     |
| Zack             | 8     |

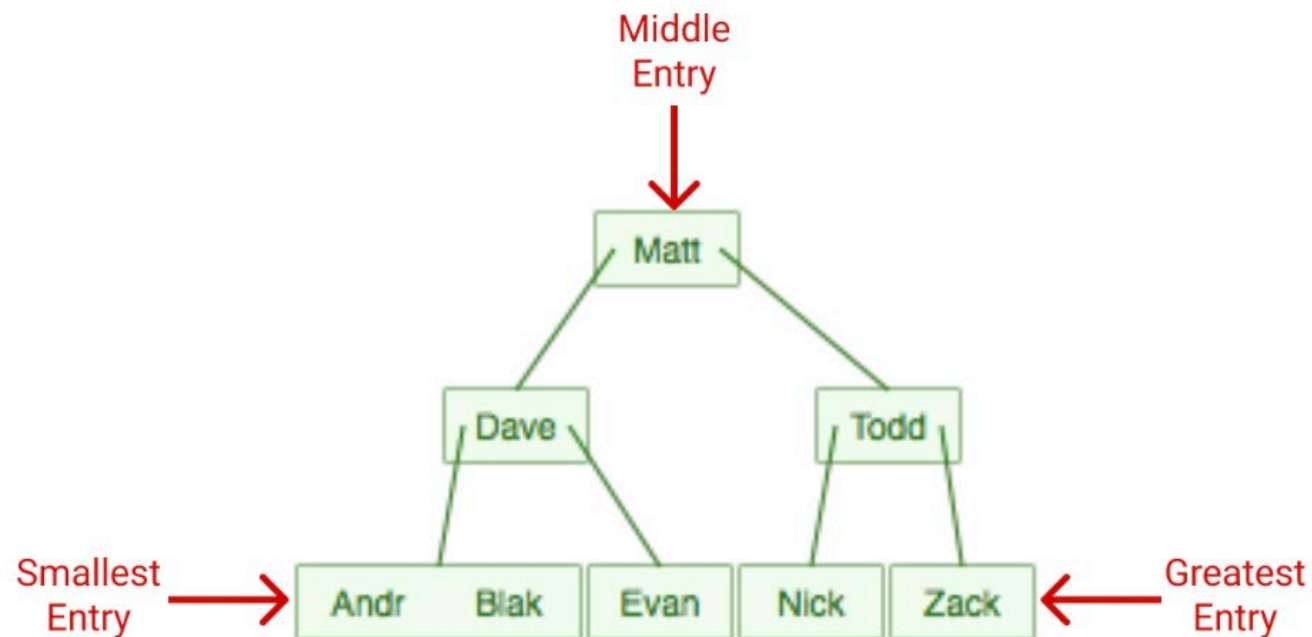
# Index – Doel

- Hoe werkt een index?
- Data wordt gestructureerd rond de geïndexeerde kolom



# Index – Doel

- Hoe werkt een index?
- Het doorlopen van deze data gebruikt een boomstructuur



# Index – CREATE Syntax

- Het aanmaken van een index is "makkelijk"
  - Syntax = makkelijk
- Het kiezen van de index strategie is "moeilijk"
  - Strategie = moeilijk

```
CREATE INDEX index_name  
ON table_name  
(kolom_naam1, kolom_naam2, ...);
```



# Index – CREATE Syntax

- De index maak je aan op basis van een kolom. De index kan een query versnellen, **als** de query een vraag stelt over de geïndexeerde kolom.
  - Je kan meerdere indexes maken per tabel, maar
  - Hoe meer indexes, hoe trager **inserts en deletes zijn (die rekening moeten houden met het onderhoud van de index)**
  - *CREATE INDEX* *index\_name* ON *table\_name* (*kolom\_naam1*);
- Wanneer je meer complexe queries maakt die condities hebben voor meerdere kolommen, dan kan je een multi-kolom index maken
  - **Meerdere index != Multi-kolom index**
  - Volgorde van de kolommen in CREATE bepaalt de volgorde waarin gezocht wordt
  - *CREATE INDEX* *index\_name* ON *table\_name* (*kolom\_naam1*, *kolom\_naam2*, ...);

# Index – CREATE Syntax

- Samengevat:
  - Multiple indexes kan slecht zijn voor database performantie.
    - Wees voorzichtig met het aanmaken hiervan
  - Multi-kolom index is beter, maar je moet op voorhand goed weten welke queries vaak uitgevoerd worden.
    - Je index kan enkel specifieke queries op de gegeven kolommen versnellen.

# Index – ALTER Syntax

- Past een bestaande index aan:

```
ALTER INDEX index_name  
ON table_name  
(kolom_naam1, kolom_naam2, ...);
```

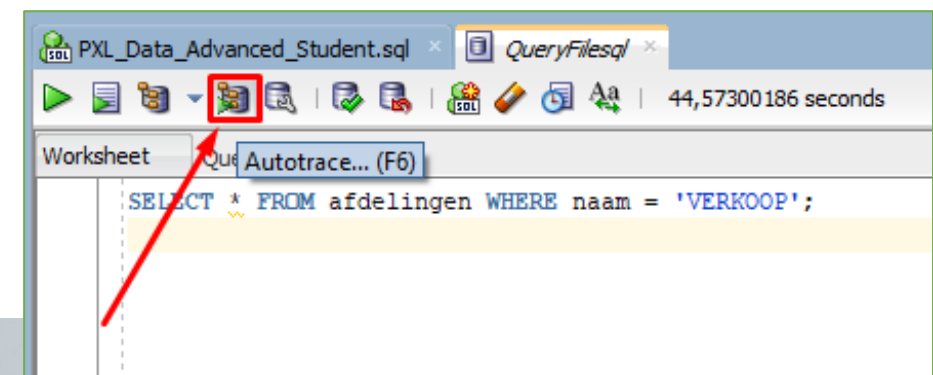
# Index – DROP Syntax

- Verwijdert een bestaande index:

*DROP INDEX **index\_name**;*

# Index – Testing

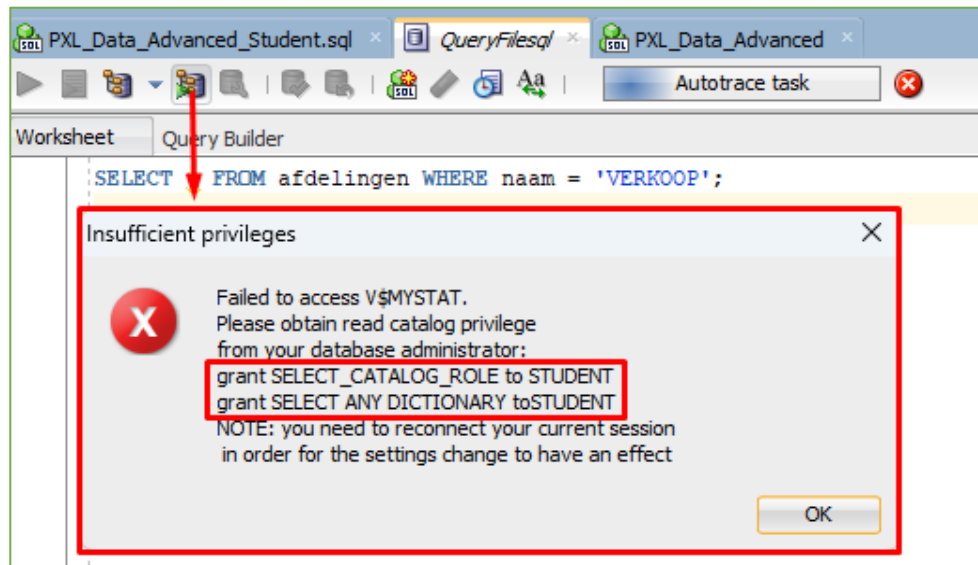
- Wanneer we aan SQL optimalisatie doen, hoe kunnen we testen of onze optimalisatie succesvol is.
  - Achterliggend wordt er een EXECUTION PLAN gebruikt
  - Dit plan bepaalt welke tabellen worden aangesproken en in welke volgorde
  - Controleer de SCAN OPTIONS
- We kunnen dit plan voor een bepaalde query terugvinden in SQL Developer onder "Autotrace"





# Index – Testing

- Wanneer je Autotrace activeert, dan kan het zijn dat je nog rechten mist. Voeg deze toe als "system" user + Reconnect.

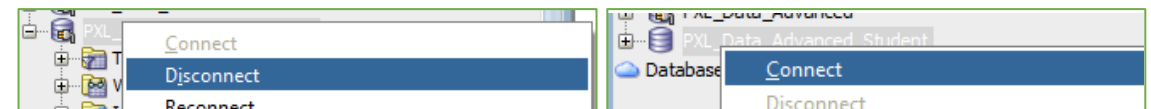


```
SQL> connect system/pxl
Connected.
SQL> grant SELECT_CATALOG_ROLE to student;

Grant succeeded.

SQL> grant SELECT ANY DICTIONARY to student;

Grant succeeded.
```



# Index – Testing

- Autotrace

The screenshot shows the Oracle SQL Developer interface. The top pane displays the query: `SELECT * FROM afdelingen WHERE naam = 'VERKOOP';`. The bottom pane shows the Autotrace results, which are highlighted with a red border. The Autotrace results include a table with columns: OPERATION, OBJECT\_NAME, OPTIONS, CARDINALITY, COST, and LAST\_CR.

| OPERATION        | OBJECT_NAME | OPTIONS        | CARDINALITY | COST | LAST_CR |
|------------------|-------------|----------------|-------------|------|---------|
| SELECT STATEMENT |             |                |             |      | 1       |
| TABLE ACCESS     | AFDELINGEN  | BY INDEX ROWID |             | 1    | 1       |
| INDEX            | A2_NAAM_LN  | UNIQUE SCAN    |             | 1    | 0       |

Below the table, there are sections for Access Predicates, Filter Predicates, and Other XML. The Access Predicates section shows `NAAM='VERKOOP'`. The Filter Predicates section shows `UPPER(NAAM)='VERKOOP'`. The Other XML section shows an XML structure with information about the database version (11.2.0.2), parse schema ('STUDENT'), and other details.

Hoe kunnen we het gebruik van onze index interpreteren?

# Index – Testing

- Autotrace: Geef een query in op de MEDEWERKERS tabel
  - Er wordt een index unique scan gebruikt.

Worksheet Query Builder

```
SELECT * FROM MEDEWERKERS WHERE NAAM = 'CASPER';
```

Script Output x Explain Plan x Autotrace x Query Result x

SQL HotSpot | 0,059 seconds

| OPERATION        | OBJECT_NAME | OPTIONS        | CARDINALITY |
|------------------|-------------|----------------|-------------|
| SELECT STATEMENT |             |                |             |
| TABLE ACCESS     | ADEDELINGEN | BY INDEX ROWID | 1           |
| INDEX            | A2_NAAM_UN  | UNIQUE SCAN    | 1           |

Access Predicates  
NAAM='VERKOOP'

Filter Predicates  
UPPER(NAAM)='VERKOOP'

# Index – Testing

- Autotrace: Geef een query in op de MEDEWERKERS tabel
  - Er wordt een index unique scan gebruikt.
  - Maar ... we hebben nooit een index aangemaakt voor MEDEWERKERS ?!
- **Let op:** Oracle maakt een index automatisch aan wanneer een unique key of primary key wordt gebruikt!



# Index – Testing

- Voorbeeld zonder unique primary key constraint

```
/*  
CREATE TABLE Blocks (BlockID int, Color varchar(20), Shape varchar(20));  
INSERT INTO blocks VALUES (1, 'red', 'square');  
INSERT INTO blocks VALUES (2, 'red', 'triangle');  
INSERT INTO blocks VALUES (3, 'red', 'square');  
INSERT INTO blocks VALUES (4, 'blue', 'square');  
INSERT INTO blocks VALUES (5, 'blue', 'circle');  
INSERT INTO blocks VALUES (6, 'blue', 'square');  
INSERT INTO blocks VALUES (7, 'yellow', 'circle');  
INSERT INTO blocks VALUES (8, 'blue', 'square');  
INSERT INTO blocks VALUES (9, 'red', 'triangle');  
*/
```

```
select * from blocks where shape = 'circle';
```

# Index – Testing

- Voorbeeld zonder unique primary key constraint
  - Er wordt **FULL SCAN** uitgevoerd
  - Er is geen index gebruikt

```
select * from blocks where shape = 'circle';
```

| OPERATION         | OBJECT_NAME | OPTIONS |
|-------------------|-------------|---------|
| SELECT STATEMENT  |             |         |
| TABLE ACCESS      | BLOCKS      | FULL    |
| Filter Predicates |             |         |
| SHAPE='circle'    |             |         |

# Index – Testing

- Voorbeeld na aanmaken van index

```
create index blocks_index on blocks (color, shape);
select * from blocks where color = 'red';
```

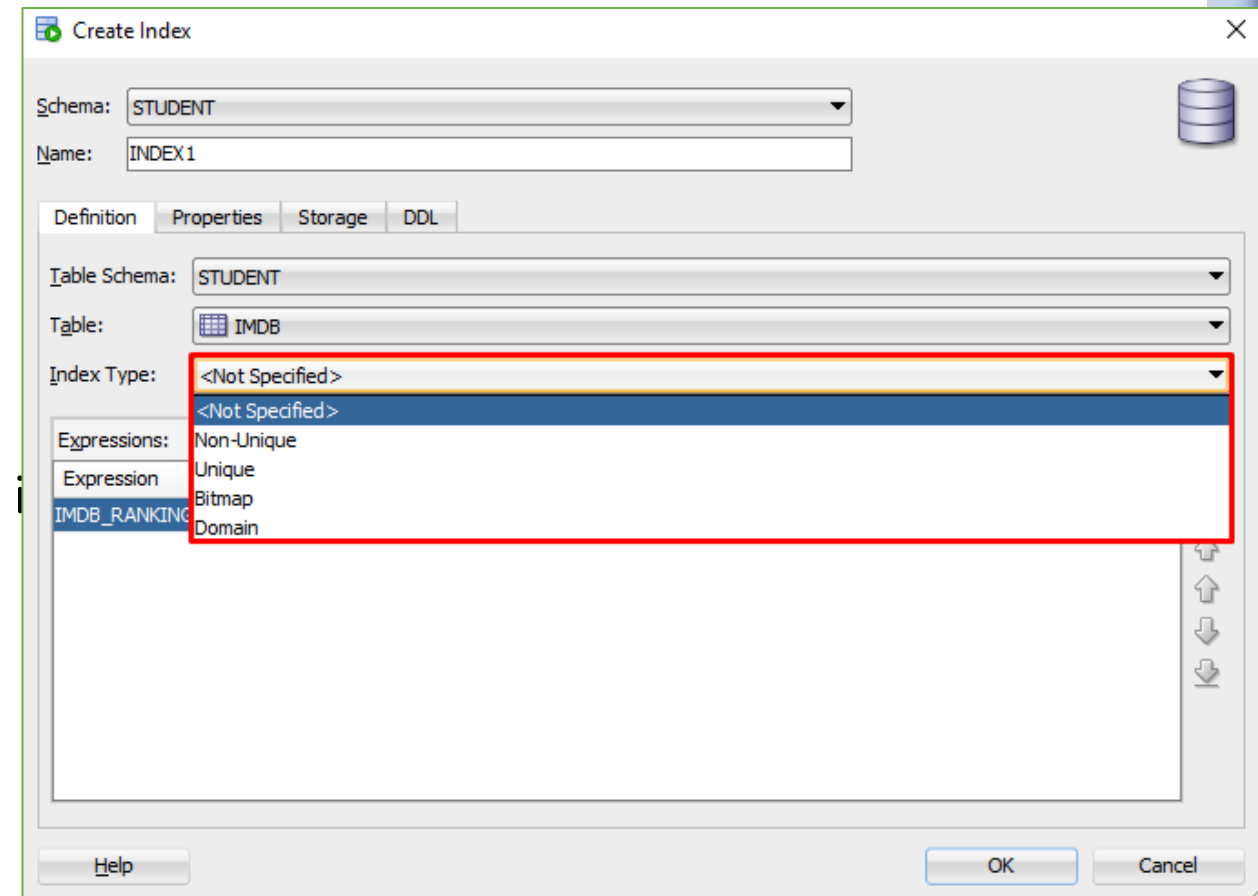
```
create index blocks_index on blocks (color, shape);
select * from blocks where color = 'red';
```

SQL HotSpot | 0,066 seconds

| OPERATION         | OBJECT_NAME  | OPTIONS        |
|-------------------|--------------|----------------|
| SELECT STATEMENT  |              |                |
| TABLE ACCESS      | BLOCKS       | BY INDEX ROWID |
| INDEX             | BLOCKS_INDEX | RANGE SCAN     |
| Access Predicates |              |                |
| COLOR='red'       |              |                |

# Index – Extra

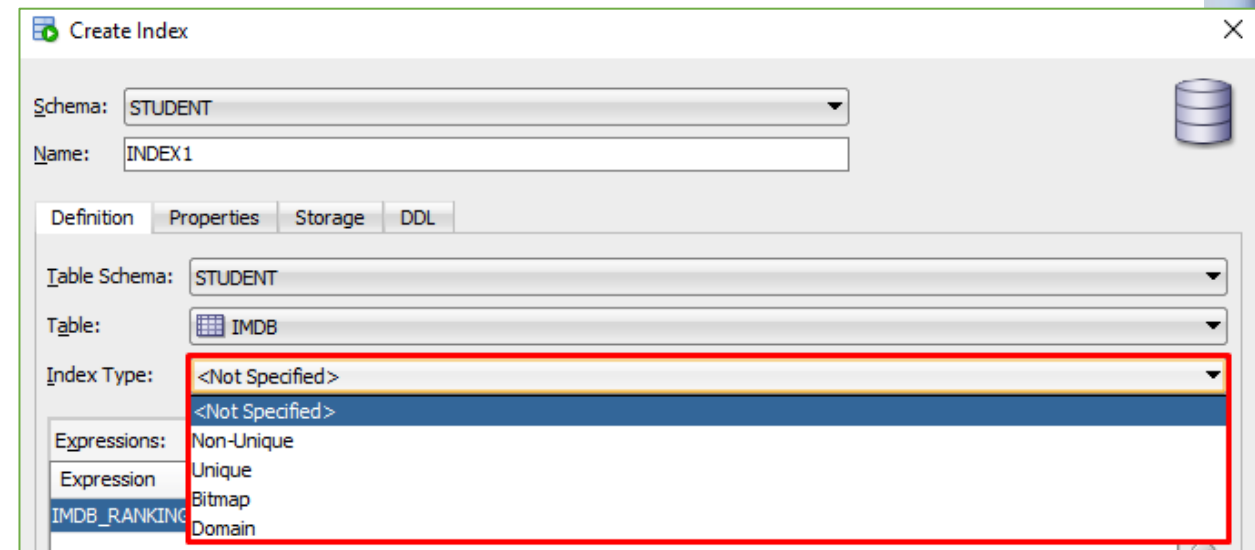
- Als je SQL Developer gebruikt om een Index aan te maken, dan zal je zien dat je kan kiezen voor een type.
  - Unique:
    - Elke waarde in de kolom(men) is **uniek**
  - Non-Unique: default





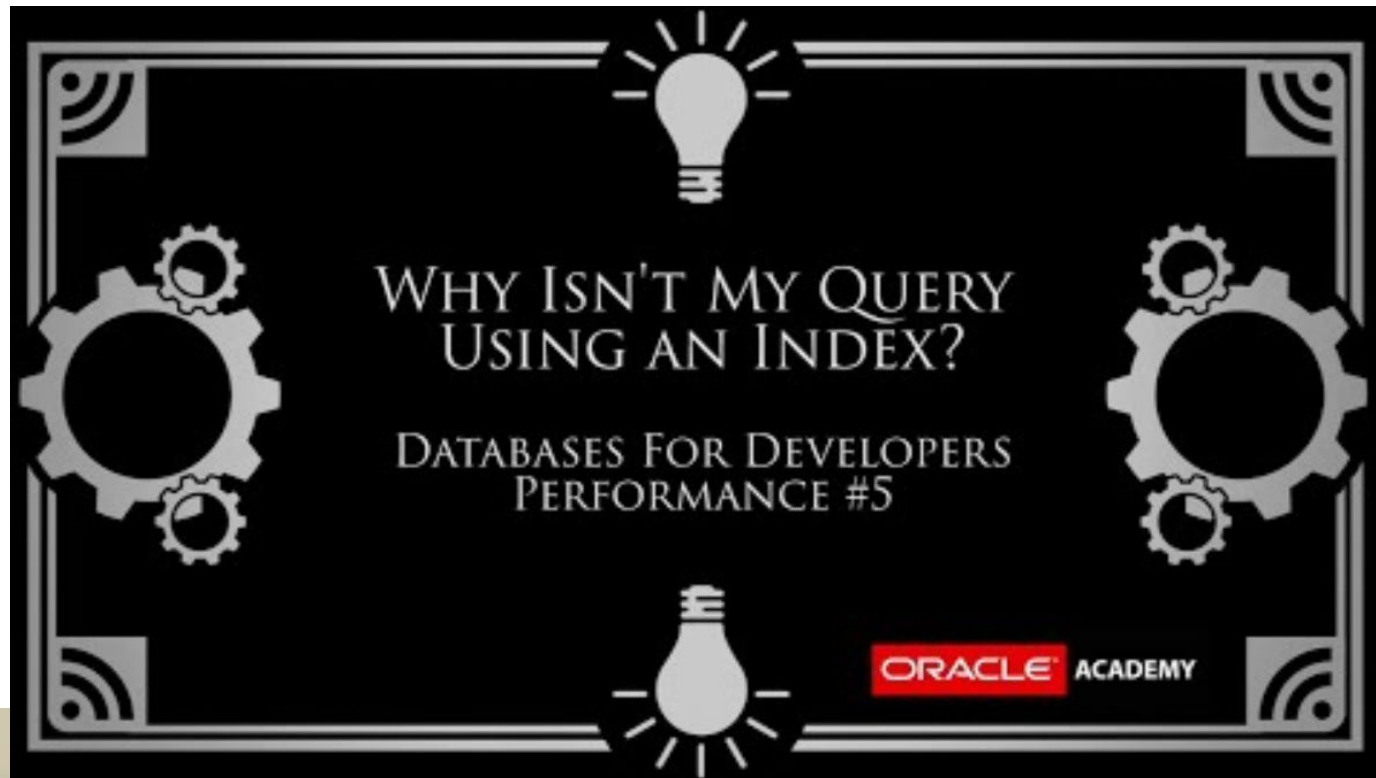
# Index – Extra

- Als je SQL Developer gebruikt om een Index aan te maken, dan zal je zien dat je kan kiezen voor een type.
  - Het gebruik van de alternatieve types (bitmap en domain) vallen buiten de scope van dit vak:
    - Bitmap:** beschrijft een **range van rijen** en slaat op welke rijen in de range wel of niet een waarde hebben in de kolom in een bitmap (voor elke mogelijke waarde van de kolommen).
    - Bitmaps zijn enkel goed voor datasets die nooit aangepast worden (data warehouse), aangezien het **write concurrency moet blokkeren** om **dirty writes** te voorkomen
    - Domain:** specialiseerd in indexeren van **ruimtelijke data** of **afbeeldingen**



# Index – Extra (niet kennen)

- Er zijn uitzonderingen waarin er toch geen index wordt gebruikt.
  - Ontdek waarom:



# Synonym

- Doel
- CREATE Syntax
- ALTER Syntax
- DROP Syntax

GOOD

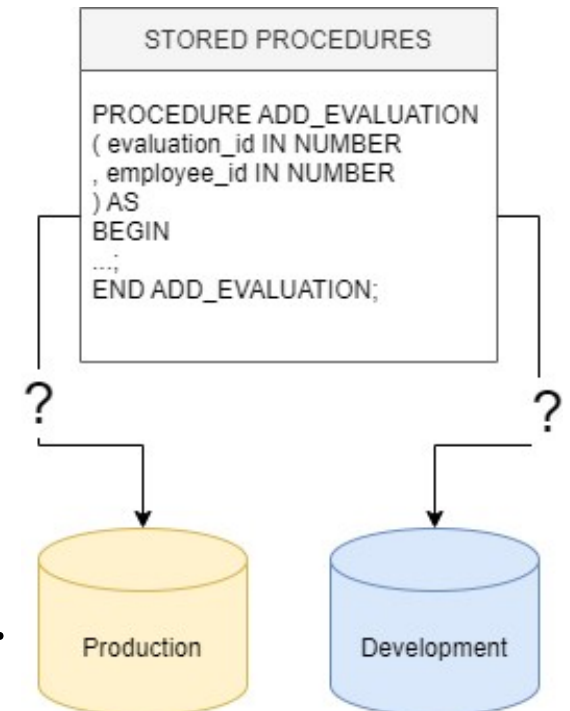


-

*excellent, fine, superior, wonderful, marvelous, qualified, suited, suitable, apt, proper, capable, generous, kindly, friendly, pleasant, satisfactory, well-behaved, obedient, honorable, reliable, top-notch, worthy, grand, salubrious, noble, great, beneficial, splendid, first-rate, genuine, ample, estimable, valid, helpful, expedient, righteous, advantageous, sterling, superb, respectable, edifying, trustworthy, gracious, obliging, agreeable, safe, profitable, beneficial*

# Synonym – Doel

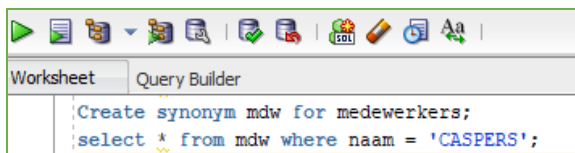
- Use case:
- Er is een database object met een lange of onduidelijke naam. Hoe kunnen we een alternatieve naam introduceren?
- We hebben stored procedures (sql opdrachten) die gebruik maken van de productie database (met *prod\_tables*), maar we willen graag testen uitvoeren op de development database (met *dev\_tables*) zonder alle queries in de stored procedures aan te moeten passen.  
Hoe lossen we dit op?
  - = Abstractie laag om productie applicaties te beschermen.



# Synonym – CREATE Syntax

CREATE SYNONYM *synonym\_voor\_data\_object*  
FOR *naam\_van\_data\_object*;

- *Indien je geen rechten hebt om een synonym aan te maken, dan zal je deze als system user moeten toevoegen aan student.*
  - Zie [Prerequisites](#)
  - Test een synonym uit!



```
SQL> connect system/pxl
Connected.
SQL> grant create synonym to student;

Grant succeeded.

SQL> grant create any synonym to student;

Grant succeeded.
```



# Synonym – ALTER Syntax

*ALTER SYNONYM synonym\_voor\_data\_object  
FOR naam\_van\_data\_object **COMPILE**;*

- **COMPILE:**
  - Een synonym plaatst een afhankelijkheid op het target data object en wordt onbruikbaar (*invalid*) wanneer het target data object aangepast of verwijderd wordt.
  - Wanneer je een onbruikbaar (*invalid*) synonym compiled, dan wordt het terug bruikbaar (*valid*).

# Synonym – DROP Syntax

- Verwijdert een bestaande synonym:

*DROP SYNONYM synonym\_voor\_data\_object*

# Oefeningen!

