Modèle relationnel
Langage de requêtes

SITE :http://www.univ-orleans.fr/lifo/Members/Mirian.Halfeld/

Les requêtes

- Exemple des requêtes.

STUDENTS $[Number, Name, Address]$
INSCRIPTION $[StudNb, CourseCode, Year, Time]$
COURSE $[Code, Title, WebSite]$

1. Quel est le numéro de l'étudiant *Jean Martin*?
2. Quel est l'adresse de *Marie Leblanc*?
3. Dans quel cours l'étudiant *Yves Laurant* est inscrit en 2009?
4. Quels sont les étudiants inscrits dans le cours de BD et non inscrits dans le cours de Java?

Les requêtes

- **Le résultat d'une requête**:
  - Une table: schéma + instance.
  - Schéma: La requête définie le schéma de la relation résultat.
  - Instance: Une instance de relation obtenue à partir de l'instance de la base de données.
- **Comment écrire une requête?**

  Langages de requêtes:
  - Théoriques (proposés avec le modèle relationnel de Codd): Calcul relationnel, algèbre relationnelle, datalog, etc
  - Commerciales: SQL

Déclaratif × Procédural

- Dans nos requêtes nous écrivons ce que nous voulons avoir dans le résultat!.
- Nous n'indiquons pas comment obtenir ce résultat.
- Nous souhaitons ainsi avoir des langages de requêtes déclaratifs.
- Les SGBD décident comment obtenir le résultat.
- Les SGBD travaillent en interne avec des différents langages procéduraux qui spécifient comment obtenir le résultat.

Déclaratif $\times$ Procédural: un exemple

STUDENTS $[Number, Name, Address]$
INSCRIPTION $[StudNb, CourseCode, Year, Time]$

Dans quel cours l'étudiant *Yves Laurant* est inscrit en 2009?

Déclaratif

$\{X_{CourseCode} \mid \langle X_{StudNb}, X_{CourseCode}, 2009, X_{Time}\rangle$ is a tuple in INSCRIPTION and $\langle X_{StudNb}$, *Yves Laurant*, $X_{Address}\rangle$ is a tuple in STUDENTS$\}$

Procédural

```
for each tuple T1= (x1,z1,y1, h1) in relation INSCRIPTION do {
 if y1 = '2009'
  then  for each tuple T2= (x2,z2,y2) in relation STUDENTS do {
         if x1 = x2 and z2 = 'Yves Laurant'
                then output z1
                 }
}
```

Langages de requêtes relationnels

- **Langages de requêtes (*Query languages*)**: Permettent la manipulation et l'obtention des données de la base.
- Le modèle relationnel admet des languages de requêtes puissants et simples.
- **Les languages de requêtes sont différents des langages de programmation**
  - QL ne font pas des calculs compliqués.
  - QL permettent un accès facile (déclaratif) aux données dans une base.
    $\Rightarrow$ Les langages de requêtes spécifient ce que la sortie doit contenir, mais ils ne disent pas comment obtenir la sortie.

Langages de requêtes formels

- La base pour des langages ”réels” comme SQL.
  - **Algèbre relationnelle**: plus opérationnel, très utile pour faire les plans d’exécution d’une requête.
  - **Calcul relationnel**: l’utilisateur décrit ce qu’il cherche et non comment l’obtenir (plus déclaratif). Basé sur le calcul de prédicats du premier ordre.

EXEMPLE

STUDENTS $[Number, Name, Address]$

Algèbre relationnelle: $\Pi_{Number}(\sigma_{Name='CaroleDupuis'}(STUDENTS))$

Calcul relationnel : $\{x \mid \exists y STUDENTS(x, CaroleDupuis, y)\}$

Algèbre Relationnelle

- Langage pour définir des nouvelles relations à partir des relations existantes.
- Les expressions dans l'algèbre relationnelle commencent par des noms de relations ou par des tuples (valeurs constantes). Nous pouvons construire des expressions plus complexes de façon récursive, en appliquant des opérateurs (définis ensuite).
- Deux approches: Algèbre dans le point de vue *non étiqueté* et algèbre dans le point de vue *étiqueté*. Les opérateurs peuvent changer selon l' approche adoptée.
- Une *requête* est une expression de l'algèbre relationnelle.

SQL

- SQL : *Structured Query Language*.
- Principal langage de requêtes dans les bases de données relationnelles.
- Langage normalisé.
- Fondé sur l'algèbre et le calcul relationnels.

Les requêtes conjonctives

- Requêtes simples, limitées, mais extrêmement naturelles et fréquentes.
- Exemples:
  Soit la base de données

  CINEMA[Film [Title, Director, Year], Theater [Title, Director, TheaterName]]

  1. Qui est le metteur en scène (*director*) de *The Cameraman*?
  2. Quels sont les cinémas qui passent *The Cameraman*?

Comment écrire une requête conjonctive?

En algèbre relationnelle

- Algèbre - point de vue étiqueté: algèbre SPJR
  - Sélection
  - Projection
  - Jointure
  - Renommage
- Algèbre - point de vue NON étiqueté: algèbre SPC
  - Sélection
  - Projection
  - Produit Cartésien

Les langage algébrique: Requêtes de base

FILM	Director	Title	Year
	Woody Allen	Annie Hall	1977
	Alfred Hitchcock	Rear Window	1951
	Clint Eastwood	Million Dollar Baby	2005
	Clint Eastwood	Grand Torino	2008

En algèbre: FILM

En SQL :

```
SELECT * FROM Film;
```

Sélection en algèbre

FILM	Director	Title	Year
	Woody Allen	Annie Hall	1977
	Alfred Hitchcock	Rear Window	1951
	Clint Eastwood	Million Dollar Baby	2005
	Clint Eastwood	Grand Torino	2008

Quels sont les films de Clint Eastwood?

Algèbre SPJR: $\sigma_{Director=ClintEastwood}(FILM)$

Algèbre SPC: $\sigma_{1=ClintEastwood}(FILM)$

FILM	Director	Title	Year
	Clint Eastwood	Million Dollar Baby	2005
	Clint Eastwood	Grand Torino	2008

Sélection: définition formelle

- **Algèbre étiquetée**:
  - L'opérateur σ prend en argument une instance I d'une relation R et une condition de sélection ϕ. Il renvoie une instance de relation **ayant le même sort** et définie par

$$\sigma_\phi(R) = \{t \in I(R) \mid \phi\}$$

    La condition de sélection ϕ est définie comme suit:
    - Si $A, B \in$ **att**, $a \in$ **dom** et op un opérateur dans $\{=, \geq, \leq, <, >, \neq\}$, alors
      $A\ op\ a$ (en supposant que $A \in sort(R)$) et
      $A\ op\ B$ (en supposant que $A, B \in sort(R)$)
      sont des conditions de sélection.
    - Si f et f' sont des conditions de sélection alors: (f), $\neg f$, $f \vee f'$, $f \wedge f'$ sont des conditions de sélection.
    - L'algèbre SPJR est définie SANS négation et SANS disjonction.
- **Algèbre non-étiquetée**: définie de façon similaire en prenant en compte l'arité au lieu du *sort*.

Comment faire une sélection en SQL?

```
SELECT *
FROM Film
WHERE Director =  'Clint Eastwood' ;
```

- La sélection est un opérateur horizontal qui sélectionne des tuples.
- OBS: SQL mélange l'approche étiquetée et non étiquetée (rappelez vous des *insert*). **Le mieux est de toujours utiliser l'approche étiquetée** pour plus de lisibilité et facilité de maintenance.

Projection en algèbre

FILM	Director	Title	Year
	Woody Allen	Annie Hall	1977
	Alfred Hitchcock	Rear Window	1951
	Clint Eastwood	Million Dollar Baby	2005
	Clint Eastwood	Grand Torino	2008

Quels sont les directeurs dans la base de données?

Algèbre SPJR: $\Pi_{Director}(FILM)$

Algèbre SPC: $\Pi_1(FILM)$

FILM	Director
	Woody Allen
	Alfred Hitchcock
	Clint Eastwood

Projection: définition formelle

- **Algèbre étiquetée**
  - Notation:

$$\pi_{A_1,\ldots,A_n}(R), n \geq 0$$

    (les répétitions ne sont pas permises).
  - L'opérateur π prend en argument une instance I d'une relation R **dont le sort contient** $\{A_1, \ldots, A_n\}$. Il renvoie une instance avec sort $\{A_1, \ldots, A_n\}$ et définie par:

$$\pi_{A_1,\ldots,A_n}(R) = \{\langle t(A_1) \ldots t(A_n)\rangle \mid t \in I(R)\}.$$

- **Algèbre non-étiquetée**: définie de façon similaire en prenant en compte l'arité au lieu du *sort*. La position des attributs est utilisée à la place du nom.

Comment faire une projection en SQL?

```
SELECT Director
FROM Film;
```

- La projection est un opérateur vertical qui sélectionne des colonnes dans la table.
- ATTENTION: quel est le résultat de notre requête dans notre base exemple? Avons nous de doublons? Pourquoi?

Exemples de requêtes: combinaisons des opérateurs

Quel est le metteur en scène du film *Rear Window*?

Algèbre étiquetée:

$\pi_{Director}(\sigma_{Title=RearWindow}FILM)$

Algèbre non-étiquetée:

$\pi_1(\sigma_{2=RearWindow}FILM)$

SQL:

```
SELECT Director
FROM Film
WHERE Title = 'Rear Window' ;
```

Produit cartésien en algèbre

- Opération binaire définie dans **l'algèbre NON étiquetée**.
- **Définition formelle**: Opérateur permet d'associer des relations
  L'opérateur prend en entrée un couple de relations, d'arités quelconques n et m, et renvoie une relation d'arité $n + m$
  Si $arity(R) = n$ et $arity(S) = m$ alors:

$$R \times S = \{\langle t(1), \ldots, t(n), s(1), \ldots, s(m)\rangle \mid t \in I(R) \; et \; s \in J(S)\}$$

FILM	1	2	3
	Woody Allen	Annie Hall	1977
	Alfred Hitchcock	Rear Window	1951
	Clint Eastwood	Million Dollar Baby	2005
	Clint Eastwood	Grand Torino	2008

Quel est le résultat de $FILM \times FILM$?

Comment faire le produit cartésien en SQL?

- Pour indiquer $FILM \times FILM$ nous devons utiliser un alias, un autre nom pour la table FILM.

```
SELECT *
FROM Film f1, FILM f2;
```

- Dans la requête, $f1$ et $f2$ sont des alias pour les tuples dans FILM. Cela corresponds à voir le schéma du produit cartésien comme ayant les attributs $f1.Director$, $f1.Title$, $f1.Year$, $f2.Director$, $f2.Title$, $f2.Year$

```
DIRECTOR  TITLE  YEAR DIRECTOR  TITLE   YEAR
Clint Eastwood Grand Torino 2008 Clint Eastwood
Grand Torino 2008
Clint Eastwood Grand Torino 2008 Clint Eastwood
Million Dollar Baby 2005
...
Alfred Hitchcock  Rear Window 1951 Clint Eastwood
Grand Torino 2008
Alfred Hitchcock  Rear Window 1951 Clint Eastwood
Million Dollar Baby 2005
```

Exemple

FILM	1	2	3
	Woody Allen	Annie Hall	1977
	Woody Allen	Match Point	2005
	Alfred Hitchcock	Rear Window	1951
	Clint Eastwood	Million Dollar Baby	2005
	Clint Eastwood	Grand Torino	2008

Lister des couples de films de la même année.

SQL:

```
SELECT f1.Title , f2.Title
FROM  Film f1 , Film f2
WHERE  f1.Year = f2.Year AND f1.Title <> f2.Title;
```

Algèbre non-étiquetée:

$\pi_{2,5}(\sigma_{3=6 \wedge 2 \neq 5}(FILM \times FILM))$

Et en algèbre étiquetée??

Jointure en algèbre étiquetée

- Opération définie dans l'**algèbre étiquetée**.
- Entrée: instances quelconques I (de R) et J (de S) avec sorts V et W respectivement.
- Sortie: instance avec sort $V \cup W$.

$$R \bowtie S = \{t \text{ sur } V \cup W \mid \text{ pour un } v \in I(R) \; et \; w \in J(S), \; t[V] = v \; et \; t[W] = w\}$$

R:

A	B
1	2
3	4

S:

B	C	D
2	5	6
4	7	8
9	10	11

$R \bowtie S$:

A	B	C	D
1	2	5	6
3	4	7	8

- **Comment écrire donc la requête** *Lister des couples de films de la même année?*

Renommage

- Renommage d'attributs d'un ensemble fini U: une bijection de U dans **att**.
- Renommage d'attributs f sur U: définition de l'ensemble de couples $(A, f(A))$ pour lesquels $A \neq f(A)$.
- Notation: $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_n$ pour indiquer que $f(A_i) = B_i$ pour tout $i \in [1, n]$, $n \geq 0$.
- Un opérateur de renommage pour les attributs de U: expression δ_f, où f est un renommage d'attributs de U. Il envoie les résultats sur $f[U]$
- Pour une instance I de R, sur U:

$$\delta_f(R) = \{v \text{ sur } f[U] \mid \text{ pour un } u \in I(R),$$
$$v(f(A)) = u(A) \text{ pour tout } A \in U\}$$

Exemple

FILM	Director	Title	Year
	Woody Allen	Annie Hall	1977
	Woody Allen	Match Point	2005
	Alfred Hitchcock	Rear Window	1951
	Clint Eastwood	Million Dollar Baby	2005
	Clint Eastwood	Grand Torino	2008

Lister des couples de films de la même année.

$$\pi_{Title,Title1}(FILM \bowtie \delta_{Director,Title \rightarrow Director1,Title1}(FILM))$$

Comment faire la jointure en SQL?

Similaire au produit cartésien: il est toujours nécessaire d'établir les conditions.

Spécifier clairement les attributs dont les valeurs doivent être identiques, i.e., sur lesquels se fait la jointure.

Lister des couples de films de la même année.

```
SELECT  f1.Title, f2.Title
FROM Film f1, Film f2
WHERE  f1.Title <> f2.Title AND  f1.Year = f2.Year
```

Encore un exemple

FILM	Director	Title	Year
	Woody Allen	Annie Hall	1977
	Alfred Hitchcock	Rear Window	1951
	Clint Eastwood	Million Dollar Baby	2005
	Woody Allen	Match Point	2005
	Clint Eastwood	Grand Torino	2008

THEATER	Title	Director	TheaterName
	Annie Hall	Woody Allen	Le Champo
	Million Dollar Baby	Clint Eastwood	Le Pathe
	Grand Torino	Clint Eastwood	Action Christine
	Match Point	Woody Allen	Action Christine

Comment faire la jointure en SQL?

Spécification des conditions

Lister les salles de cinéma affichant des films de 2005

```
SELECT  t.TheaterName
FROM Film f, Theater t
WHERE  f.Title =  t.Title
AND f.Director = t.Director
AND f1.Year = 2005;
```

Comment faire la jointure en SQL?

- Avec JOIN: plusieurs types d'opérateurs JOIN.
- Ces opérateurs n'existaient pas de manière explicite les premiers standards SQL.
- NATURAL JOIN est similaire à la jointure de l'algèbre relationnelle.
- JOIN peut être utiliser quand les attributs ont des noms différents.

Lister les salles de cinéma affichant des films de 2005

```
SELECT  TheaterName
FROM Film NATURAL JOIN Theater
WHERE  Year = 2005;
```

Autre exemple

STUDENTS $[Number, Name, Address]$
INSCRIPTION $[StudNb, CourseCode, Year, Time]$

Lister les noms des étudiants inscrits dans le cours BD1-2009

```
SELECT  Name
FROM Students  JOIN Inscription ON StudNb = Number
WHERE  CourseCode = 'BD1-2009' ;
```

Révision: l'algèbre SPJR

- **Algèbre étiquetée - SPJR**
  - Sélection, projection, jointure, renommage
  - Pas de négation ou de disjonction dans la condition de la sélection
  - **Langage algébrique SPJR**:
    - Requêtes de base: R ou $\{\langle A : a\rangle\}$ où R est un schéma de relation, A un attribut dans **att** et $a \in$ **dom**.
    - Pour toutes requêtes SPJR q_1 et q_2, les expressions:
      - $\pi_{A_1,\ldots,A_n} q_1$ avec $A_1 \ldots A_n$ des attributs dans $sort(q_1)$.
      - $\sigma_\phi q_1$, avec ϕ une condition de sélection SPJR.
      - $\delta_f(q_1)$ où f est une fonction de renommage.
      - $q_1 \bowtie q_2$

      sont des requêtes SPJR.

Exercices

Soit le schéma de base de données CINÉPHILE :

FILM	[Titre, Metteur-en-scène, Acteur]
CINÉMA	[Nom-Cinéma, Adresse, Téléphone]
PARISCOPE	[Nom-Cinéma, Titre, Heure]

Exprimer les requêtes ci-dessous en algèbre SPJR:

1. Qui est le metteur en scène de *Les temps modernes*?
2. Donnez les adresses des cinémas où on peut voir un film de Hitchcock.
3. Donnez la liste contenant les paires d'acteurs ayant joué dans les mêmes films.

Révision: l'algèbre SPC

- **Algèbre non- étiquetée - SPC**
  - Sélection, projection, produit cartésien
  - Pas de négation ou de disjonction dans la condition de la sélection
  - **Langage algébrique SPC**:
    - Requêtes de base: R ou $\{\langle a \rangle\}$ où R est un schéma de relation et $a \in$ **dom**.
    - Pour toutes requêtes SPC q_1 et q_2, les expressions:
      - $\pi_{1,\ldots,n} q_1$ avec $|q_1| \geq n$.
      - $\sigma_\phi q_1$, avec ϕ une condition de sélection SPC.
      - $q_1 \times q_2$

      sont des requêtes SPC.

Exercices

Soit le schéma de base de données CINÉPHILE :

FILM	[Titre, Metteur-en-scène, Acteur]
CINÉMA	[Nom-Cinéma, Adresse, Téléphone]
PARISCOPE	[Nom-Cinéma, Titre, Heure]

Exprimer les requêtes ci-dessous en algèbre SPC:

1. Qui est le metteur en scène de *Les temps modernes*?
2. Donnez les adresses des cinémas où on peut voir un film de Hitchcock.
3. Donnez la liste contenant les paires d'acteurs ayant joué dans les mêmes films.

Révision: les bases des requêtes SQL

Format général:

`select` $A_1, \ldots, A_n$
`from` $R_1, \ldots, R_m$
`where` ϕ

- La clause *select* va spécifier les attributs $A_1, \ldots, A_n$ dont les valeurs nous cherchons, i.e., elle va spécifier une projection.
- La clause *from* va lister les schémas de relations $R_1, \ldots, R_m$.
- La clause *where* va spécifier la condition de sélection ϕ.
- La clause *where* est facultative.

La commande SQL est donc traduite par le système comme l'expression suivante de l'algèbre:

$$\pi_{1,\ldots,n}(\sigma_{\phi}(R_1 \times \ldots \times R_m)$$

Exercices

Soit le schéma de base de données CINÉPHILE :

FILM	[Titre, Metteur-en-scène, Acteur]
CINÉMA	[Nom-Cinéma, Adresse, Téléphone]
PARISCOPE	[Nom-Cinéma, Titre, Heure]

Exprimer les requêtes ci-dessous en SQL:

1. Qui est le metteur en scène de *Les temps modernes*?
2. Donnez les adresses des cinémas où on peut voir un film de Hitchcock.
3. Donnez la liste contenant les paires d'acteurs ayant joué dans les mêmes films.

Quelques compléments de SQL

- Comparaison approximative de chaînes: `like`
- Format:

```
SELECT attribute_name(s)
FROM relation_name
WHERE attribute_name LIKE pattern
```

Le *pattern* est une chaîne de caractères qui peut contenir des symboles spéciaux, à savoir:

- % qui est un *joker*, remplace 0 ou plusieurs caractères.
- _ remplace 1 seul caractère.
- Nous pouvons aussi utiliser NOT LIKE

Exemple

FILM	Director	Title	Year
	Woody Allen	Annie Hall	1977
	Alfred Hitchcock	Rear Window	1951
	Clint Eastwood	Million Dollar Baby	2005
	Clint Eastwood	Grand Torino	2008

Trouver tous les metteurs en scène avec un t dans leurs nom.

```
SELECT Director
FROM FILM
WHERE Director  LIKE '%t%'
```

Modification de l’affichage des colonnes

```
SELECT Director AS  MetteurScene
FROM FILM
WHERE Director  LIKE ’%t%’
```

- Le mot AS est facultatif
- Même principe pour la création d’un alias (voir exemple précédant) pour permettre l’utilisation d’une même relation dans une requête.

L’ordre du résultat

- Il est possible de donner le résultat d’une requête de manière ordonnée.
- L’ordre est établi par rapport à la valeur d’une attribut. Un deuxième attribut peut être utilisé pour résoudre les cas de valeurs identiques et ainsi de suite.
- Format: `ORDER BY <list of attributes>`
- Par défaut l’ordre est croissant mais nous pouvons demander l’ordre décroissant avec l’instruction DESC.
- Cette clause est toujours la dernière d’une requête

Films dont le nom du metteur en scène a un t, dans l’ordre alphabétique.

```
SELECT *
FROM FILM
WHERE  Director  LIKE ’%t%’
ORDER BY Director , Title;
```

Élimination des doublons

- Le résultat d'une requête SQL est un *bag*.
- Utilisation de l'instruction DISTINCT pour supprimer les doublons.

FILM	Director	Title	Year
	Woody Allen	Annie Hall	1977
	Alfred Hitchcock	Rear Window	1951
	Clint Eastwood	Million Dollar Baby	2005
	Clint Eastwood	Grand Torino	2008

Trouver tous les metteurs en scène avec un t dans leurs nom.

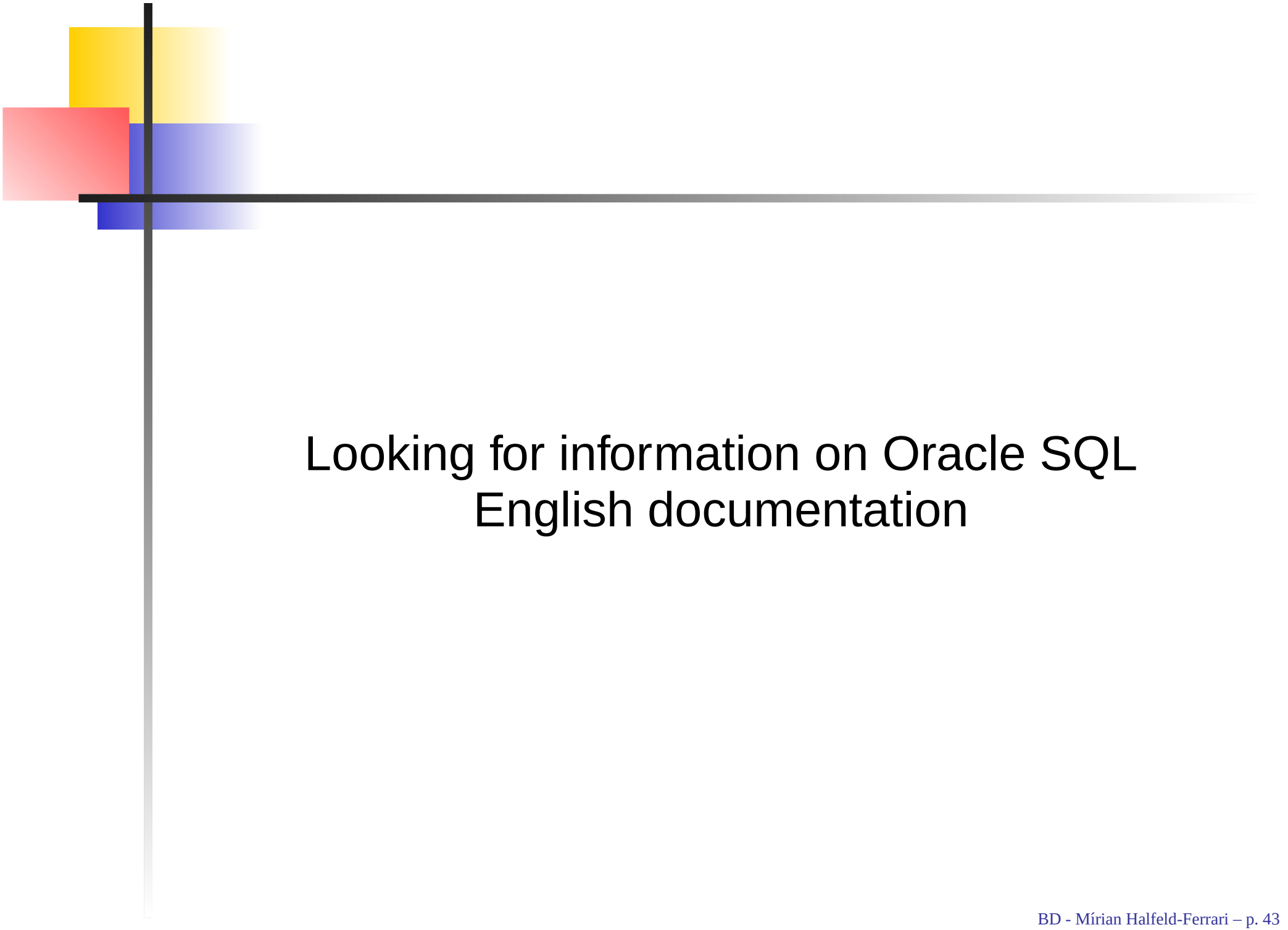
```
SELECT DISTINCT Director
FROM FILM
WHERE Director  LIKE '%t%'
```

Calcul dans le select

Les colonnes du select peuvent être le résultat d'un calcul entre des colonnes des tables de la requête.

Supposons une base contenant des informations sur des étudiants et leurs notes.

```
SELECT  NomEtud, Note*Coef
FROM  Etudiant ...
```



Looking for information on Oracle SQL English documentation

Oracle Dates and Times

- Oracle supports both date and time.
- Rather than using two separate entities, date and time, Oracle only uses one, DATE.
- The DATE type is **stored in a special internal format that includes not just the month, day, and year, but also the hour, minute, and second**.
- The DATE type is used in the same way as other built-in types such as INT. For example, the following SQL statement creates a relation with an attribute of type DATE:

```
create table x(a int, b date);
```

Date format

- When a DATE value is displayed, Oracle must first **convert that value from the special internal format to a printable string**.
- The conversion is done by a function `TO_CHAR`, according to a DATE format. Oracle's default format for DATE is `"DD-MM-YY"`. Therefore, when you issue the query

```
select b from x;
```

you will see something like:

```
B
---------
01-APR-98
```

Changing format

- Whenever a DATE value is displayed, Oracle will call `TO_CHAR` automatically with the default DATE format.
- **You may override the default behavior** by calling `TO_CHAR` explicitly with **your own** DATE format. For example,

```
SELECT TO_CHAR(b, 'YYYY/MM/DD') AS b
FROM x;
```

returns the result:

```
B
---------------
1998/04/01
```

- The general usage of `TO_CHAR` is:

```
TO_CHAR(<date>, '<format>')
```

where the `<format>` string can be formed from over 40 options.

Input date value

- Function `TO_DATE` converts a string to a DATE value, again according to the DATE format.
- Normally, you do not have to call `TO_DATE` explicitly: Whenever Oracle expects a DATE value, it will automatically convert your input string using `TO_DATE` according to the default DATE format `"DD-MON-YY"`.
- Example, to insert a tuple with a DATE attribute, type:

```
insert into x values(99, '31-may-98');
```

- Alternatively, you may use `TO_DATE` explicitly:

```
insert into x
values(99,to_date('1998/05/31:12:00:00AM','yyyy/mm/dd:hh:mi:ssa
```

- The general usage of `TO_DATE` is:

```
TO_DATE(<string>, '<format>')
```

where the `<format>` string has the same options as in `TO_CHAR`.

Changing Oracle default

- You can change the default DATE format of Oracle from `"DD-MON-YY"` to something you like by issuing the following command in `sqlplus`:

  ```
  alter session set NLS_DATE_FORMAT='<my_format>';
  ```

- The change is only valid for the current `sqlplus` session.

The Current Time

- The built-in function SYSDATE returns a DATE value containing the current date and time on your system.
- Example,

```
select to_char(sysdate, 'Dy DD-Mon-YYYY HH24:MI:SS') as "Curren
from dual;
```

returns

```
Current Time
---------------------------------------------
Fri 11-Sep-2009 11:57:18
```

which is the time when this document was been prepared.

The Current Time: remarks

- Two interesting things to note here:
  - You can use **double quotes to make names case sensitive** (by default, SQL is case insensitive), or to force spaces into names.
    Oracle will treat everything inside the double quotes literally as a single name. In this example, if `"Current Time"` is **not quoted**, it would have been interpreted as two case insensitive names CURRENT and TIME, which would actually cause a **syntax error**.
  - `DUAL` is **built-in relation in Oracle** which serves as a dummy relation to put in the FROM clause when nothing else is appropriate. For example, try `select 1+2 from dual;`
  - Another name for the built-in function `SYSDATE` is `CURRENT_DATE`. Be aware of these special names to avoid name conflicts.

Operations on DATE

- You can compare DATE values using the standard comparison operators such as $=, != , >$, etc.
- **You can subtract two DATE values**, and the result is a FLOAT which is the number of days between the two DATE values. In general, the result may contain a fraction because DATE also has a time component.
- For obvious reasons, **adding, multiplying, and dividing two DATE values are not allowed**.
- **You can add and subtract constants to and from a DATE value**, and these numbers will be **interpreted as numbers of days**. For example, `SYSDATE+1` will be tomorrow.
- With the help of `TO_CHAR`, string operations can be used on DATE values as well. For example,

  `to_char(<date>, 'DD-MON-YY') like '%JUN%'`

  evaluates to true if `<date>` is in June.