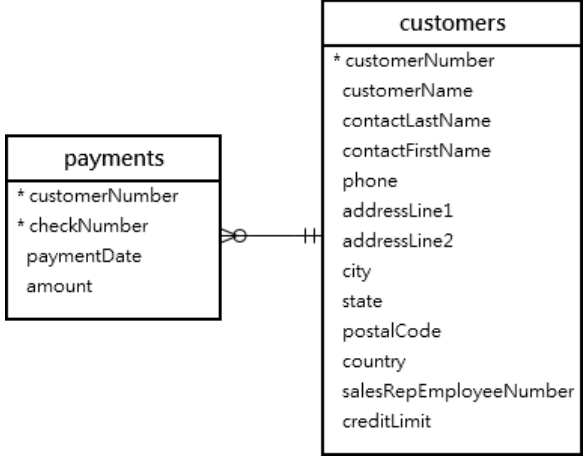


VISTES

Definició	Exemple	
<p>MySQL Views (mysqltutorial.org)</p> <p>Una vista (VIEW) és una query emmagatzemada a la BD amb un nom.</p> <p>Per crear una vista la sentència és:</p> <pre>CREATE VIEW nom AS SELECT</pre> <p>A vegades a les vistes se les anomena taules virtuals.</p> <p>Una vista no emmagatzema físicament les dades. Quan es fa un SELECT contra una vista, el que s'executa és la query especificada en la definició de la vista.</p>	<p>Taules customers i payments, a partir de les quals crearem la vista customerPayments</p>  <pre>CREATE VIEW customerPayments AS SELECT customerName, checkNumber, paymentDate, amount FROM customers INNER JOIN payments USING (customerNumber);</pre>	
	<p>Sense la vista la query per obtenir els clients i les seus pagaments és:</p> <pre>SELECT customerName, checkNumber, paymentDate, amount FROM customers INNER JOIN payments USING (customerNumber);</pre>	<p>La mateixa query de l'esquerre es pot fer a partir de la vista:</p> <pre>SELECT customerName, checkNumber, paymentDate, amount FROM customerPayments;</pre>

Definició	Exemple
<p>Algunes de les raons i els avantatges de fer servir vistes es mostren a la dreta:</p>	<ol style="list-style-type: none"> 1) Simplificar queries complexes Si tot sovint es fa servir una query complicada, pots crear una vista basada en aquesta query. Així pots fer la query simplement apuntant a la vista en comptes de picar cada vegada la query complexe sencera. 2) Donar consistència a la lògica de negoci Suposem que sempre repetim una formula a cada query (ex: concatenem cognoms i nom separats per coma), o bé repetim un càlcul complicat (ex: dies que manquen per arribar a cap d'any). Per a fer aquestes operacions consistents i que retornin el mateix resultat en qualsevol select que les demani es pot tenir una vista que les implementi. 3) Afegir nivells de seguretat Una taula pot exposar un munt de dades, inclosa informació sensible com per exemple comptes bancaris. Fent servir vistes i privilegis, es pot limitar quina part de la informació d'una taula veu cada usuari. Per exemple: <ul style="list-style-type: none"> • la taula d'empleats pot tenir: número del INSS, adreça de contacte ..., a la qual només ha de tenir accés el departament de RH. • pel departament d'Administració crearem una vista que mostri part de la informació de la taula d'empleats, aquella a la que poden accedir: nom, cognoms i departament. • els usuaris del departament d'Administració no tindran privilegis per consultar la taula d'empleats sinó la vista d'empleats que s'ha creat per ells. 4) Fer compatibles canvis amb versions anteriors (<i>Enable backward compatibility</i>) Suposem que voleu normalitzar una taula gran en moltes de més petites. I no voleu afectar les aplicacions actuals que fan referència a la taula. En aquest cas, podeu crear una vista, el nom de la qual sigui el mateix de la taula gran a partir de la qual s'han creat les taules noves, de manera que totes les aplicacions puguin fer referència a la vista com si fos una taula.

Creació de vistes actualitzables

view-updatability

Per crear una vista que pugui servir per modificar informació d'una taula de la BD. La SELECT que defineix la vista no pot contenir cap dels elements següents:

- Aggregate functions (SUM(), MIN(), MAX(), COUNT(), and so forth)
- DISTINCT
- GROUP BY, HAVING
- UNION or UNION ALL
- Subquery in the select list
- Nondependent subqueries in the select list fail for INSERT, but are okay for UPDATE, DELETE. For dependent subqueries in the select list, no data change statements are permitted.
- Certain joins (see additional join discussion later in this section)
- Reference to nonupdatable view in the FROM clause
- Subquery in the WHERE clause that refers to a table in the FROM clause
- Refers only to literal values (in this case, there is no underlying table to update)
- Multiple references to any column of a base table (fails for INSERT, okay for UPDATE, DELETE)

Per comprovar si una vista es actualitzable pot fer la següent consulta en MySql:

```
select table_schema, table_name, is_updatable
from information_schema.views
where table_schema = 'hr'
```

El resultat s'assemblarà al següent:

TABLE_SCHEMA	TABLE_NAME	IS_UPDATABLE
hr	emp_details_view	YES

Les que tinguin valor YES a la columna «is_updatable» són les vistes actualitzables.

CREATE VIEW.....WITH CHECK OPTION	Exemple
<p>Per assegurar la consistència d'una vista, es a dir, que els usuaris només puguin veure o actualitzar les dades que són visibles segons la definició de la vista es fa servir la clàusula WITH CHECK OPTION.</p> <p>Una vista que conté aquesta clàusula aplica restriccions a l'hora de fer un UPDATE, INSERT o DELETE a partir de la vista.</p> <p>Understand LOCAL & CASCADED in WITH CHECK OPTION (mysqltutorial.org)</p> <p>Si no s'indica el tipus de CHECK, per defecte es CASCADED, però hi ha altres tipus:</p> <ul style="list-style-type: none"> ◆ <i>If a view uses a WITH LOCAL CHECK OPTION, MySQL checks the rules of views that have a WITH LOCAL CHECK OPTION and a WITH CASCADED CHECK OPTION.</i> ◆ <i>If a view uses a WITH CASCADED CHECK OPTION MySQL checks the rules of all dependent views.</i> 	<div data-bbox="1093 272 2074 911"> <pre> CREATE OR REPLACE VIEW vps AS SELECT employeeNumber, lastName, firstName, jobTitle, extension, email, officeCode, reportsTo FROM employees WHERE jobTitle LIKE '%VP%' WITH CHECK OPTION; INSERT INTO vps(employeeNumber, firstName, lastName, jobTitle, extension, email, officeCode, reportsTo) VALUES(1703, 'Lily', 'Bush', 'IT Manager', 'x9111', 'lilybush@classiccars.com', 1, 1002); </pre> </div> <p>Amb la clàusula WITH CHECK OPTION, aquest insert a partir de la vista VPS donaria error perquè el valor que es vol donar a la columna "jobTitle" no compleix la condició definida sobre els elements de la vista:</p> <ul style="list-style-type: none"> • <i>jobTitle LIKE '%VP%'</i> <p>L'error que donaria seria semblant al següent:</p> <p>Error Code: 1369. CHECK OPTION failed 'classicmodels.vps'</p>

INDEXS

Definició i ús

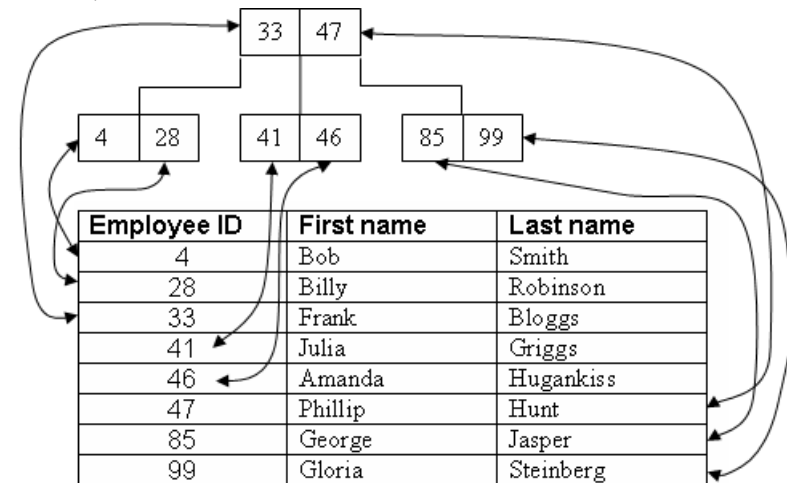
Un índex és una estructura de dades, organitzada per exemple com un B-Tree, que serveix per millorar la velocitat de recuperació de dades de les taules de la BD. El cost de tenir un índex és:

- increment d'espai necessari per a la BD, a banda de l'espai que ocupa una taula, també ocupen espai els indexos que creem per a la taula.
- les operacions d'actualització són més farragoses, es. quan inserim una fila nova hem d'inserir dades a la taula i a l'índex.

Quan es crea una taula amb una clau primària, automàticament es crea un índex per la columna o columnes que formen la clau primària.

Exemple

B-Tree,



When B-Tree is used in database index, each node holds the index value and the pointer to the row it comes from.

*For example, if the **index is on the column employeeID**, then the tree will hold both the value "47" and a pointer to the row where employeeID=47. When we search for the row where employeeID=47, we can quickly find the value 47 in the B-Tree and then follow the pointer to get the actual row*

Definició i ús

What are indexes for ?

- Speed up access in the database
- Help to enforce constraints (UNIQUE, FOREIGN KEY)
- Queries can be ran without any indexes, **but it can take a really long time**
 - MySQL uses indexes to quickly find rows with specific column values. Without an index, MySQL must scan the whole table to locate the relevant rows. The larger table, the slower it searches.
 - Response time is usually longer because records are stored randomly, and search queries have to find the needed data by looping through the entire stored records. Indexes make this search easier and faster.

A book index can be used as an example. Instead of looking for a page by going through all the pages in a book, the book index can be checked to find the page one is looking for.

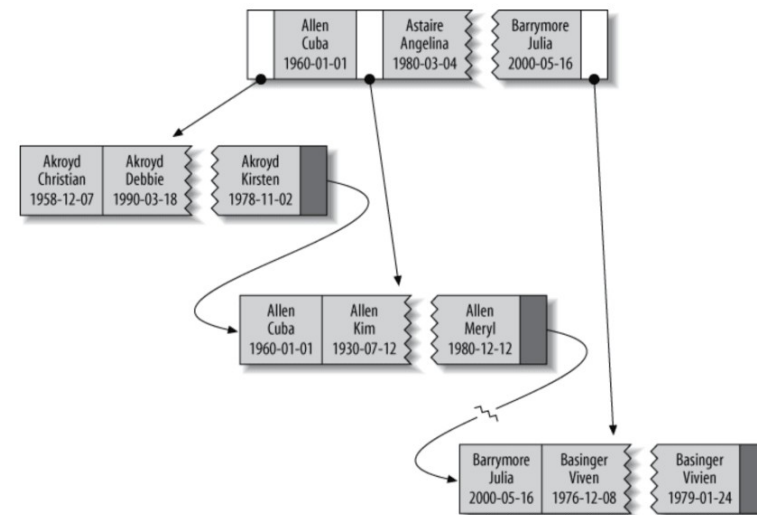
In most database systems, high and fast performance is needed. Many businesses invest huge sums of money on hardware for faster data retrievals and manipulations. But by optimizing the database, the costs can be reduced.

Exemple

Exemple d'índex múltiple, per més d'una columna;

```
last_name  varchar(50)    not null,
first_name varchar(50)    not null,
dob        date         not null,
gender     enum('m', 'f') not null,
key(last_name, first_name, dob) );
```

L'índex contindrà els valors de les columnes last_name, first_name i dob per a cada fila de la taula.



hi ha dues persones amb el mateix nom però amb dates de naixement diferents, a l'índex estan ordenades per data de naixement.

Definició i ús	Exemple																																			
Habitualment els índexs es creen quan es crea la taula, a la dreta es mostra un exemple de creació d'un index UNIQUE	<pre>CREATE TABLE Persons (ID int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Age int, CONSTRAINT UC_Person UNIQUE (ID,LastName));</pre>																																			
Els índexs també es poden afegir una vegada està creada una taula, la sentència per fer-ho és: • CREATE INDEX	<pre>CREATE INDEX index_name ON table_name (column_list)</pre>																																			
Per veure els índex que hi ha sobre una taula, en MySQL es fa servir SHOW INDEXES FROM ex: show indexes from employees;																																				
<table><tr><th>Table</th><th>Non_unique</th><th>Key_name</th><th>Seq_in_index</th><th>Column_name</th><th>Cardinality</th><th>Index_type</th></tr><tr><td>employees</td><td>0</td><td>PRIMARY</td><td>1</td><td>employee_id</td><td>107</td><td>BTREE</td></tr><tr><td>employees</td><td>1</td><td>employees_jobs_job_id</td><td>1</td><td>job_id</td><td>19</td><td>BTREE</td></tr><tr><td>employees</td><td>1</td><td>employees_departments_department_id</td><td>1</td><td>department_id</td><td>12</td><td>BTREE</td></tr><tr><td>employees</td><td>1</td><td>employees_employees_employee_id</td><td>1</td><td>manager_id</td><td>19</td><td>BTREE</td></tr></table>		Table	Non_unique	Key_name	Seq_in_index	Column_name	Cardinality	Index_type	employees	0	PRIMARY	1	employee_id	107	BTREE	employees	1	employees_jobs_job_id	1	job_id	19	BTREE	employees	1	employees_departments_department_id	1	department_id	12	BTREE	employees	1	employees_employees_employee_id	1	manager_id	19	BTREE
Table	Non_unique	Key_name	Seq_in_index	Column_name	Cardinality	Index_type																														
employees	0	PRIMARY	1	employee_id	107	BTREE																														
employees	1	employees_jobs_job_id	1	job_id	19	BTREE																														
employees	1	employees_departments_department_id	1	department_id	12	BTREE																														
employees	1	employees_employees_employee_id	1	manager_id	19	BTREE																														
Per forçar que el valor d'una columna, o un conjunt de columnes, tingui un valor únic, sense repetits, es fa servir la restricció PRIMARY KEY. A vegades però es necessita que altres columnes tinguin també valor únic, però com només pot haver-hi una clau primària, per aquesta altre columna o conjunt de columnes, es definirà un índex únic UNIQUE INDEX Pot haver-hi més d'un índex únic en una taula.	Per crear l'index: ALTER TABLE Persons ADD CONSTRAINT UC_Person UNIQUE (ID,LastName); Per esborrar l'index: ALTER TABLE Persons DROP INDEX UC_Person;																																			

Com funcionen els INDEXS EN CONSULTES

TIPUS DE CONSULTA	Exemples
<p>Using Indexes for Data Lookups</p> <p>Cercar dades a partir dels valors de certes columnes.</p> <p>A la dreta es mostra en quin cas el motor de BD farà servir l'índex i en quins no</p>	<ul style="list-style-type: none"> The classical use of <i>index on (LAST_NAME)</i> <pre>SELECT * FROM EMPLOYEES WHERE LAST_NAME="Smith"</pre> Multiple column indexes <pre>SELECT * FROM EMPLOYEES WHERE LAST_NAME="Smith" AND DEPT="Accounting"</pre> will use <i>index on (DEPT, LAST_NAME)</i> <p>Un índex per múltiples columnes: Index (A,B,C)</p> <ul style="list-style-type: none"> es fa servir en en cerques com: <ul style="list-style-type: none"> A > 5 A = 5 AND B > 6 A = 5 AND B = 6 AND C = 7 A = 5 AND B IN (1,2) AND C > 5 NO es fa servir en en cerques com: <ul style="list-style-type: none"> B > 5, <i>no fa referencia a la columna A que encapçala l'índex</i> B = 6 AND C = 7, <i>no fa referencia a la columna A que encapçala l'índex</i> fa servir part de l'índex: <ul style="list-style-type: none"> A > 5 AND B = 6, <i>aplicarà un rang sobre la 1era columna</i> A = 5 AND B > 6 AND C = 7, <i>aplicarà un rang sobre la 2ona columna</i>

TIPUS DE CONSULTA	Exemples
<p>Using Index for Sorting</p> <p>Per ordenar un resultat</p> <ul style="list-style-type: none"> A partir de la versió 8.x de MySQL, es pot especificar l'ordre d'emmagatzematge d'una columna en un índex. Això pot ser beneficiós si es necessari mostrar la columna en un ordre concret. Per defecte, l'ordre és ascendent. 	<pre>SELECT * FROM PLAYERS ORDER BY SCORE DESC LIMIT 10</pre> <ul style="list-style-type: none"> Will use index <i>on SCORE column</i> Without index MySQL will do "filesort" (external sort) which is very costly <pre>SELECT * FROM PLAYERS WHERE country='US' ORDER BY SCORE DESC LIMIT 10</pre> <ul style="list-style-type: none"> Best served by <i>Index on (COUNTRY,SCORE)</i> Combines using Index for lookup and for sort
<p>Multi Column indexes for efficient sorting</p>	<p>Un índex per múltiples columnes: Index (A,B)</p> <ul style="list-style-type: none"> es fa servir per ordenar: <ul style="list-style-type: none"> ORDER BY A, ordenar per la primera columna A=5 ORDER BY B, filtrar per la 1era columna de l'índex i ordenar per la 2ona ORDER BY A DESC, B DESC, ordenar per 2 columnes en el mateix ordre de l'índex. A>5 ORDER BY A, aplicar un rang a la columna que encapçala l'índex i ordenar per la mateixa. NO es fa servir en ordenacions com: <ul style="list-style-type: none"> ORDER BY B, <i>ordenar per una columna que NO encapçala l'índex.</i> A>5 ORDER BY B, <i>seguim intentant ordenar per una columna que NO encapçala l'índex.</i> ORDER BY A ASC, B DESC, <i>ordenar en ordre diferent al de l'índex.</i>

TIPUS DE CONSULTA	Exemples
<p>Un índex de cobertura, «covering index», és un cas especial d'índex on tots els camps necessaris per a una consulta s'inclouen a l'índex; és a dir, el propi índex conté les dades necessàries per executar les consultes sense haver d'executar lectures addicionals.</p> <ul style="list-style-type: none"> • Llegir dades de l'índex en comptes d'una taula, els índexs són més petits que les taules. • per exemple en una taula pot haver-hi 15 columnes i a l'índex només les columnes 2 i 3, que són les que es recuperen en moltes consultes. 	<p>Covering Indexes in MySQL</p> <p><i>A covering index is a special kind of index in InnoDB. When a covering index is in use, all the required fields for a query are included, or “covered”, by the index meaning that you can also reap the benefits of reading only the index instead of the data. If nothing else helps, a covering index could be your ticket to improved performance. Some of the benefits of using covering indexes include:</i></p> <ul style="list-style-type: none"> • <i>One of the main scenarios where a covering index might be of use include serving queries without additional I/O reads on big tables.</i> • <i>MySQL can also access less data due to the fact that index entries are smaller than the size of rows.</i> • <i>Most storage engines cache indexes better than data.</i> <p><i>Creating covering indexes on a table is pretty simple – simply cover the fields accessed by SELECT, WHERE and GROUP BY clauses</i></p>

SEQÜENCIES

Definició	Exemple																
<p>Una seqüència es un conjunt d'enters 1, 2, 3, ... que són generats en ordre a quan es fa una petició específica d'un número de seqüència.</p> <p>Les seqüències es fan servir tot sovint en les BD perquè moltes aplicacions requereixen que cada fila d'una taula tingui un identificador únic i les seqüències són una manera senzilla d'implementar-ho, de generar un número únic per a cada fila</p> <p><i>The simplest way in MySQL to use Sequences is to define a column as AUTO_INCREMENT and leave the remaining things to MySQL to take care.</i></p> <p>Using MySQL Sequences - Tutorialspoint</p>	<pre>mysql> CREATE TABLE insect -> (-> id INT UNSIGNED NOT NULL AUTO_INCREMENT, -> PRIMARY KEY (id), -> name VARCHAR(30) NOT NULL, # type of insect -> date DATE NOT NULL, # date collected -> origin VARCHAR(30) NOT NULL # where collected); Query OK, 0 rows affected (0.02 sec) mysql> INSERT INTO insect (id,name,date,origin) VALUES -> (NULL,'housefly','2001-09-10','kitchen'), -> (NULL,'millipede','2001-09-10','driveway'), -> (NULL,'grasshopper','2001-09-10','front yard'); Query OK, 3 rows affected (0.02 sec) Records: 3 Duplicates: 0 Warnings: 0 mysql> SELECT * FROM insect ORDER BY id;</pre> <table><tr><th>id</th><th>name</th><th>date</th><th>origin</th></tr><tr><td>1</td><td>housefly</td><td>2001-09-10</td><td>kitchen</td></tr><tr><td>2</td><td>millipede</td><td>2001-09-10</td><td>driveway</td></tr><tr><td>3</td><td>grasshopper</td><td>2001-09-10</td><td>front yard</td></tr></table> <pre>3 rows in set (0.00 sec)</pre>	id	name	date	origin	1	housefly	2001-09-10	kitchen	2	millipede	2001-09-10	driveway	3	grasshopper	2001-09-10	front yard
id	name	date	origin														
1	housefly	2001-09-10	kitchen														
2	millipede	2001-09-10	driveway														
3	grasshopper	2001-09-10	front yard														

Webgrafia

Enllaços web	
Using MySQL Sequences - Tutorialspoint	Using MySQL Sequences
MySQL - INDEXES - Tutorialspoint Types of queries that can & can't use a B-Tree index	MySQL - INDEXES
MySQL Views (mysqldata.org)	MySQL Views