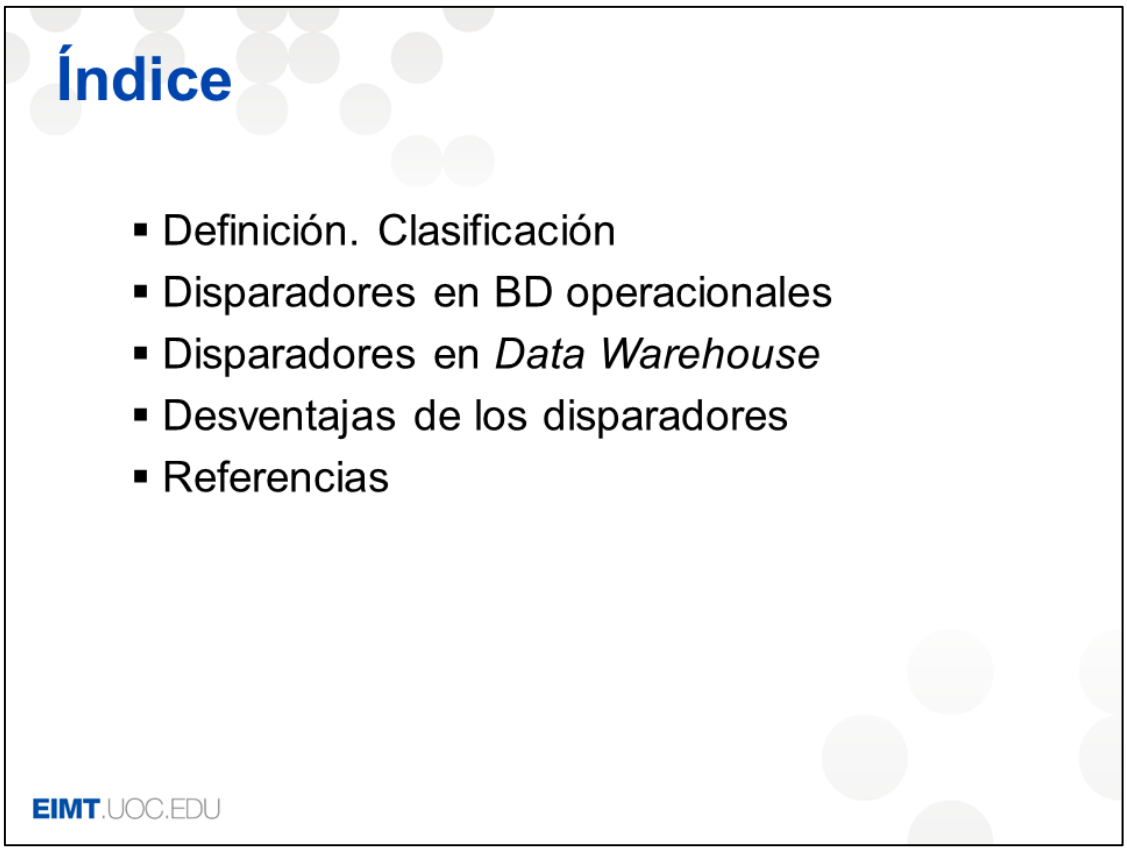
Contextualització de disparadors

PID_00253049

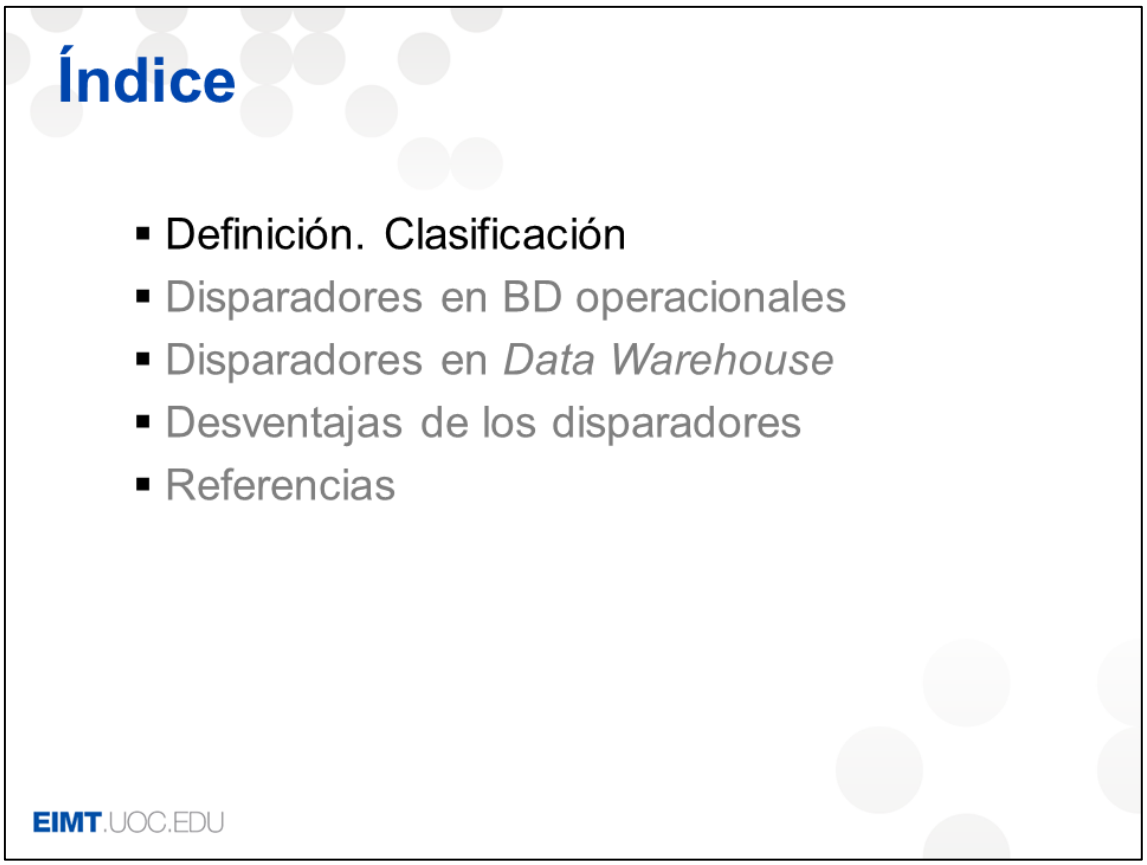
Alexandre Pereiras Magariños

Universitat Oberta de Catalunya

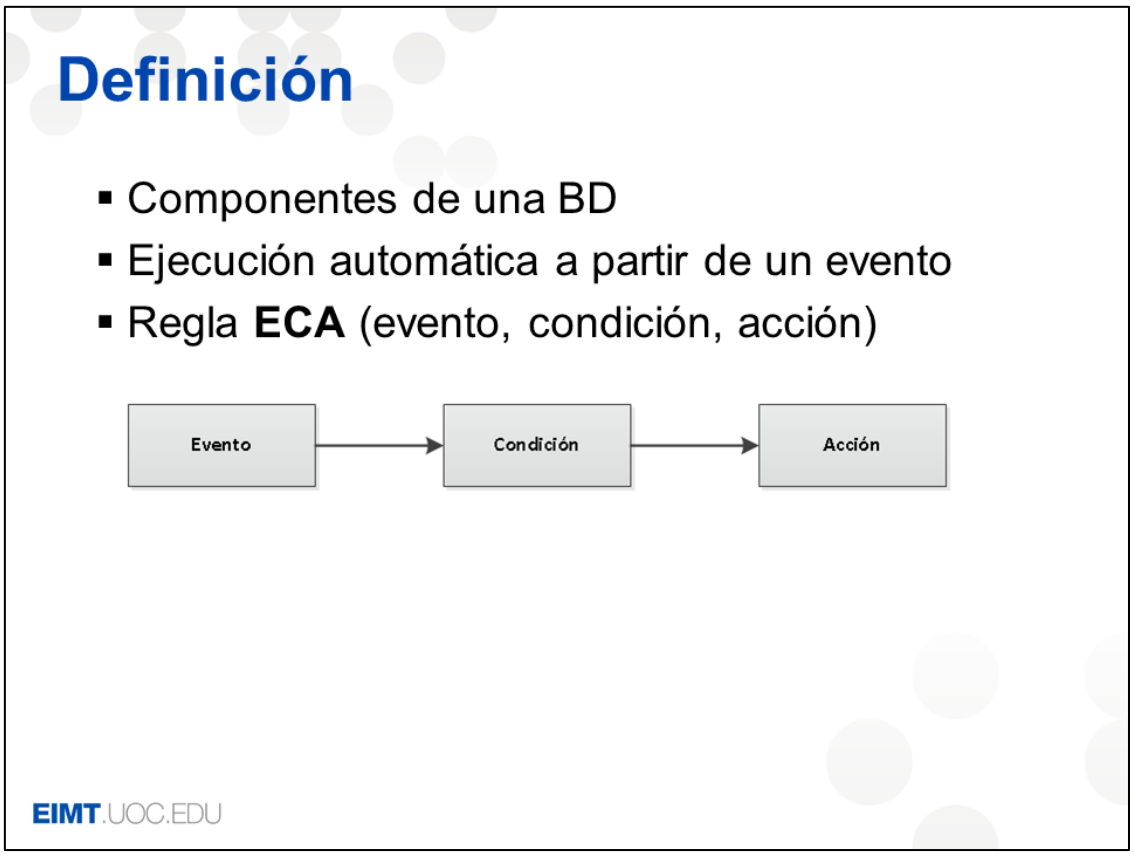


Benvinguts a aquest vídeo titulat *Contextualització de disparadors*, en què comentarem i explicarem què són els disparadors i com podem aplicar-los dins del marc de les bases de dades (BD) operacionals i de *data warehouse*.

Aquest vídeo se centrarà, primer, a introduir breument els disparadors i, a continuació, es contextualitzarà els disparadors des del punt de vista de les BD operacionals i dels d*ata warehouse*, revisant els usos que aquests components ens poden proporcionar en aquest tipus d'entorns. També veurem una sèrie de desavantatges que implica l'ús dels disparadors i, per últim, proporcionarem les referències bibliogràfiques que hem utilitzat per a elaborar aquesta presentació.

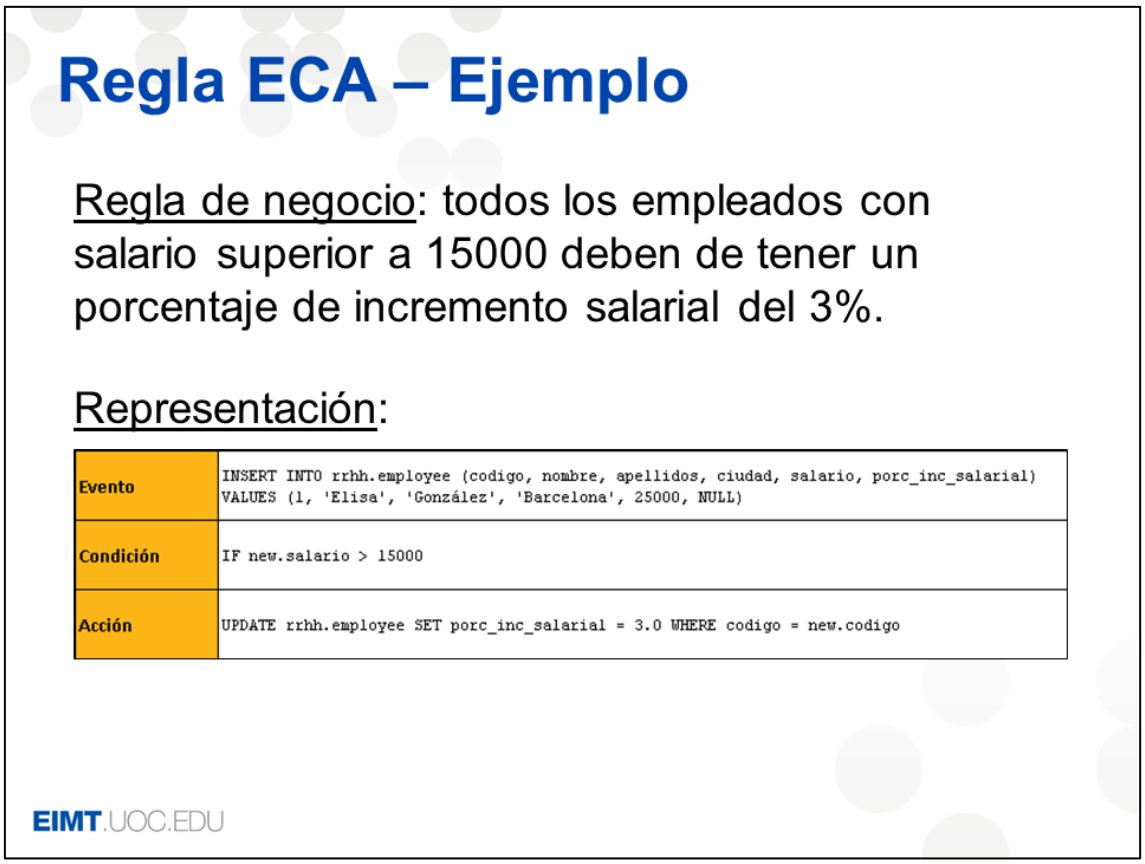


Comencem, doncs, amb la definició del disparador i les classificacions d'aquests.



Els disparadors són components específics d'una base de dades (BD) que s'associen a una taula o a una vista, i que tenen un comportament particular: aquests objectes s'executen de forma automàtica, reaccionant davant un esdeveniment sobre la taula o vista definida per a realitzar així una acció concreta. Els diferents esdeveniments a què un disparador reacciona són les operacions d'`INSERT`, `UPDATE` o `DELETE`.

Aquest tipus de mecanismes esdeveniment–acció se sol conèixer, de forma genèrica, com a mecanismes o **regles ECA**, on l'E representa l'esdeveniment, la C representa la condició i l'A representa l'acció. Les regles ECA són, per tant, regles que reaccionen davant un esdeveniment, avaluen una condició i, depenent d'aquesta avaluació, executen una acció concreta.

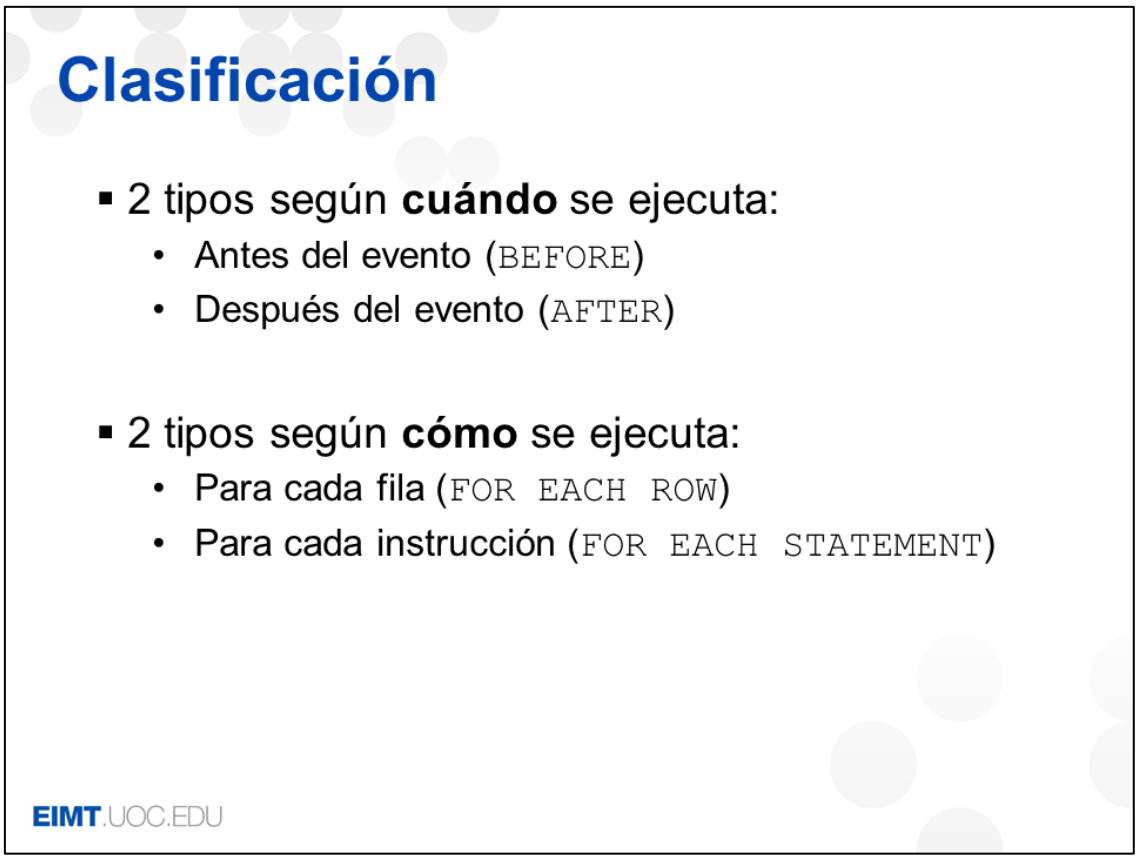


Vegem un exemple de regla ECA implementada en un sistema gestor de bases de dades (SGBD).

Imagineu que disposem d'una BD de Recursos Humans amb una taula d'empleats (*employee*) en què es guarda la informació següent dels empleats d'una empresa: codi d'empleat, nom, cognoms, ciutat, salari i percentatge d'increment salarial. Volem implementar una regla de negoci utilitzant una regla ECA, de manera que cada vegada que s'insereixi un nou empleat a la taula, s'ha de comprovar si el salari és superior a 15.000 unitats, llavors el percentatge d'increment salarial assignat ha de ser del 3% (3.0).

Aquesta regla es pot representar de la manera següent, segons el gràfic que es mostra a la part inferior d'aquesta diapositiva: veiem que es detalla l'esdeveniment com la inserció de les dades de l'empleat Elisa González, amb un salari de 25.000 unitats. Una vegada s'ha capturat l'esdeveniment d'inserció, la regla avalua la condició (`IF new.salari > 15000`). Si l'avaluació d'aquesta condició és positiva, llavors es duu a terme l'acció, que és la d'actualitzar el percentatge d'increment salarial d'aquest empleat al 3%.

Vegeu que en l'exemple proposat s'ha utilitzat `new` per a poder referenciar aquells valors de la fila que s'han inserit com a part de l'esdeveniment.



De forma genèrica, els disparadors es poden definir de diferents maneres tenint en compte dos criteris ortogonals entre si: **quan** s'executa i **com** s'executa.

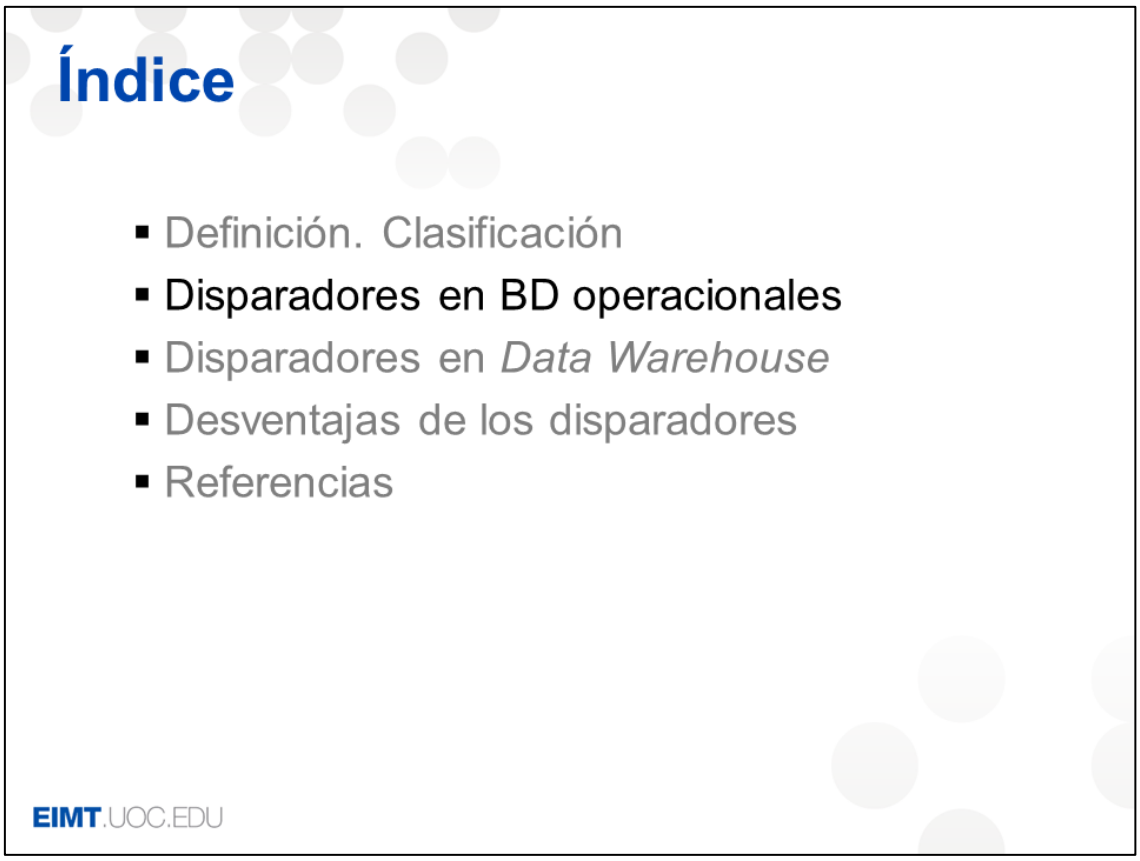
Des del punt de vista temporal, es poden definir disparadors que s'executin **abans** de l'esdeveniment (l'avaluació de la condició i l'execució de l'acció es duran a terme abans de la sentència SQL associada a l'esdeveniment) o bé **després** de l'esdeveniment (cas en què l'avaluació de la condició i l'execució de l'acció es duran a terme després de la sentència SQL associada a l'esdeveniment). Per a poder definir aquests dos tipus de disparadors s'utilitzen les clàusules SQL `BEFORE` i `AFTER` respectivament.

Des del punt de vista funcional, es poden definir disparadors que s'executin **per a cada fila** (l'avaluació de la condició i l'execució de l'acció es duran a terme de forma individual per a cada fila modificada per la sentència SQL de l'esdeveniment) o bé **per a cada instrucció** (cas en què l'avaluació de la condició i l'execució de l'acció es porti a terme una única vegada per a la instrucció especificada). Per a poder definir aquests dos tipus de disparadors s'utilitzen les clàusules SQL `FOR EACH ROW` i `FOR EACH STATEMENT` respectivament.

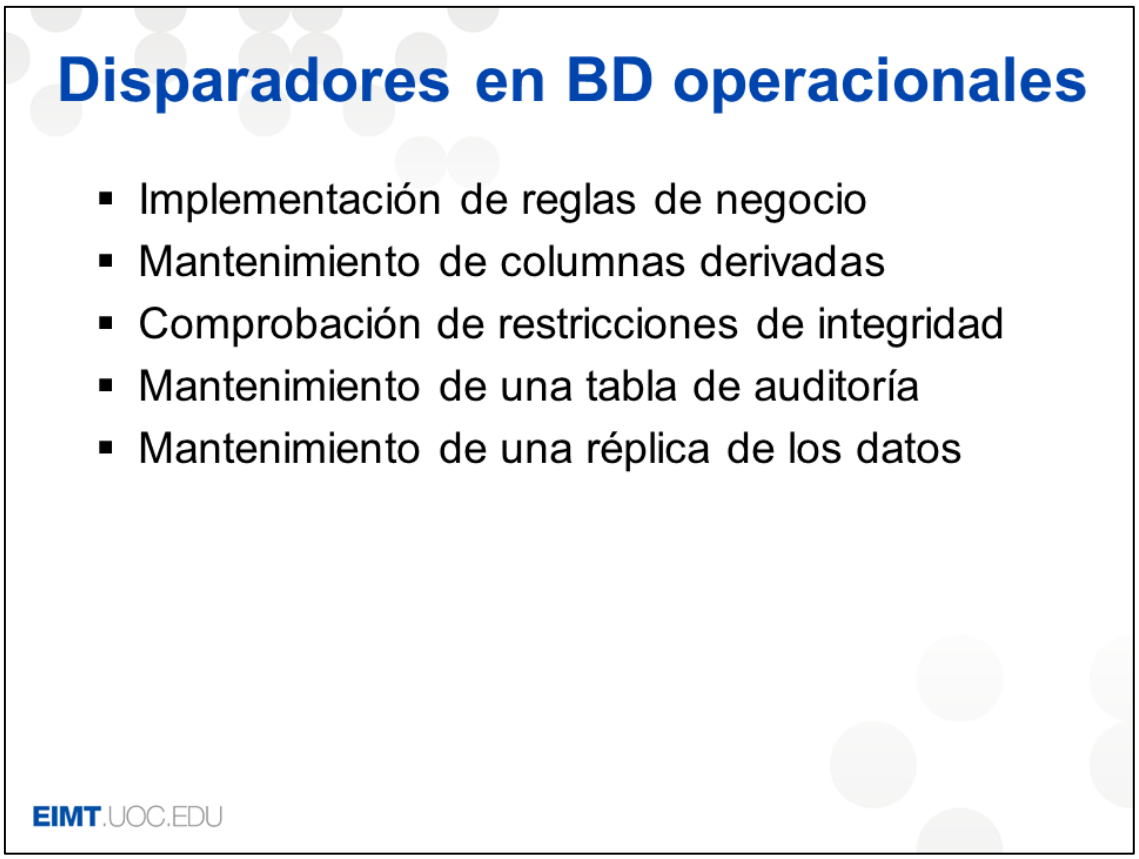
Per tant, i com podeu imaginar, podem combinar aquests dos criteris a l'hora de definir disparadors. Per exemple, podem definir un disparador `BEFORE / FOR EACH ROW` que s'executarà abans de l'esdeveniment i per a cada fila, o bé `AFTER/FOR EACH STATEMENT` que s'executarà després de l'esdeveniment i una única vegada per a tota la instrucció.

Segons l'SGBD, poden haver-hi extensions a les opcions vistes o, fins i tot, característiques especials, per la qual cosa recomanem que es revisin els manuals tècnics proporcionats per cada SGBD. En el cas de PostgreSQL, recomanem revisar la referència bibliogràfica següent, a més dels manuals en línia de PostgreSQL 9.3 (http://www.postgresql.org/docs/9.3/static/), ambdues proporcionades al final d'aquest vídeo:

Casany Guerrero, M. J.; Urpí Tubella, T.; Rodríguez Gonzalez, M. E. *El Llenguatge SQL II*. Fundació UOC. *PID_00171670*



A continuació, veurem en quins supòsits podem usar els disparadors en les BD operacionals.



Els disparadors, en l'àmbit de les BD operacionals, poden ser utilitzats per a diferents propòsits. En aquesta transparència esmentem alguns d'aquests:

- **Implementació de regles de negoci**: com per exemple, la regla de negoci proposada en aquest mateix vídeo per a definir les regles ECA.

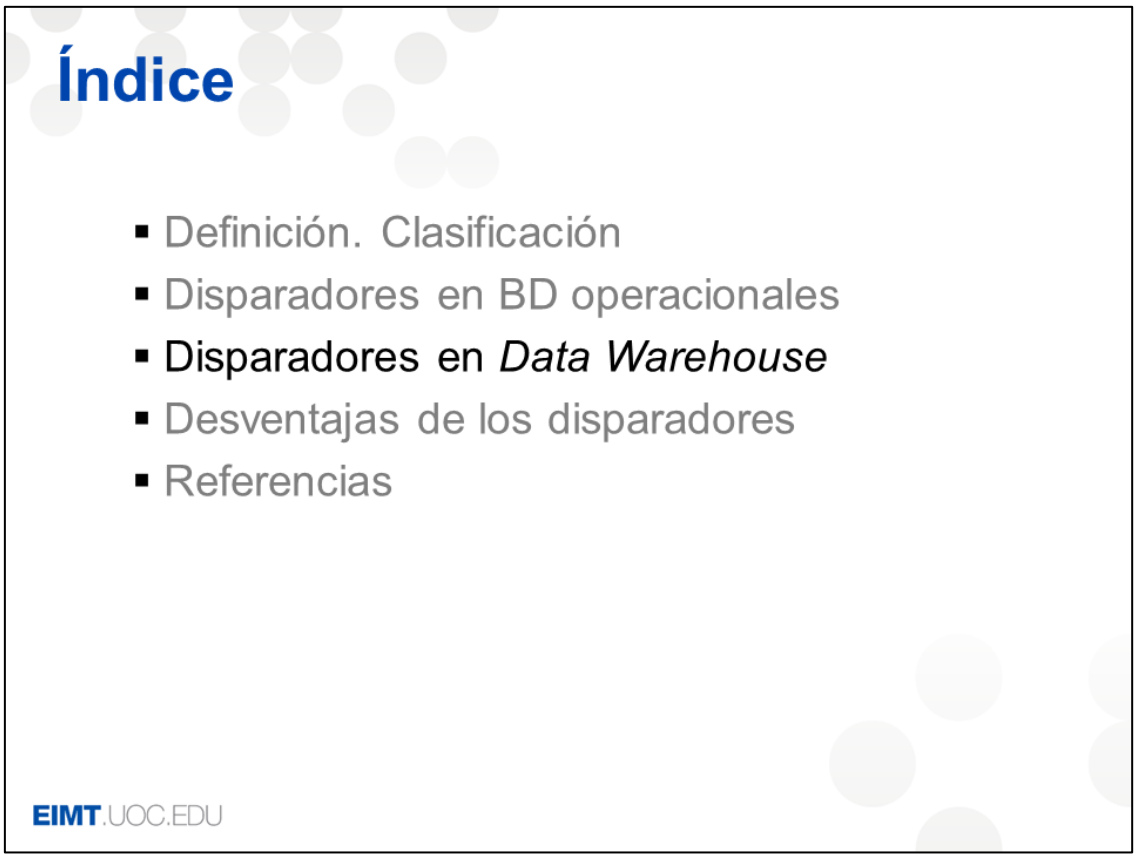
- **Manteniment de columnes derivades**: el valor d'una columna derivada en una taula es calcula a partir del valor d'altres columnes dins de la mateixa taula i, fins i tot, d'altres taules. Per tant, quan es modifiquen les columnes base, s'ha de recalcular d'una manera automàtica el valor de la columna derivada per a assegurar que les dades siguin consistents.

- **Comprovació de restriccions d'integritat referencial**: com és el cas de les regles dinàmiques, que poden necessitar referenciar-se a l'estat anterior i posterior d'una modificació. Per exemple, la implementació de la regla *la nota d'examen d'un estudiant que sol·licita una revisió no pot experimentar una baixada*. Aquesta regla no podria ser implementada mitjançant una restricció de taula o columna `CHECK`, per la qual cosa la implementació d'aquesta mitjançant un disparador sol ser una bona solució.

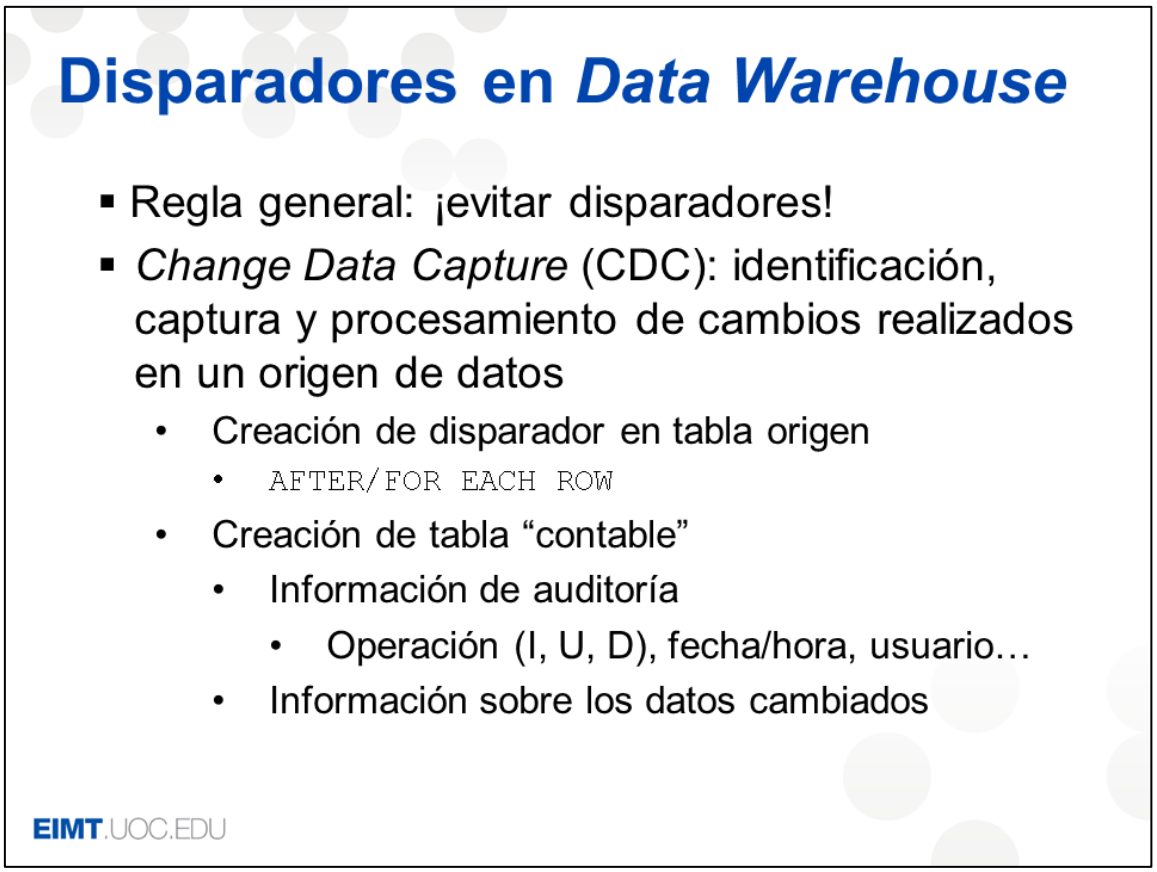
- **Manteniment automàtic d'una taula d'auditoria de l'activitat en la BD**: el propòsit d'aquesta taula d'auditoria és la de registrar, d'una manera automàtica, els canvis que es fan en una taula o taules determinades. Les dades

recopilades en aquesta taula d'auditoria podrien ser utilitzades per a realitzar estadístiques o, fins i tot, com a origen de dades per a un *data warehouse*. Aquest últim punt el comentarem en la secció següent.

- **Manteniment d'una rèplica de dades**: hi poden haver casos en què és necessari mantenir una rèplica de les dades d'una taula de forma automàtica. Per exemple, si volem emmagatzemar una còpia d'una taula de clients, amb la mateixa estructura que l'original, i que tingui les dades sincronitzades en temps real. Per a això, l'ús de disparadors podria ser una bona solució.



Una vegada hem conegut alguns dels diferents usos dels disparadors en les BD operacionals, vegem l'ús d'aquests components en *l'entorn data warehouse*.



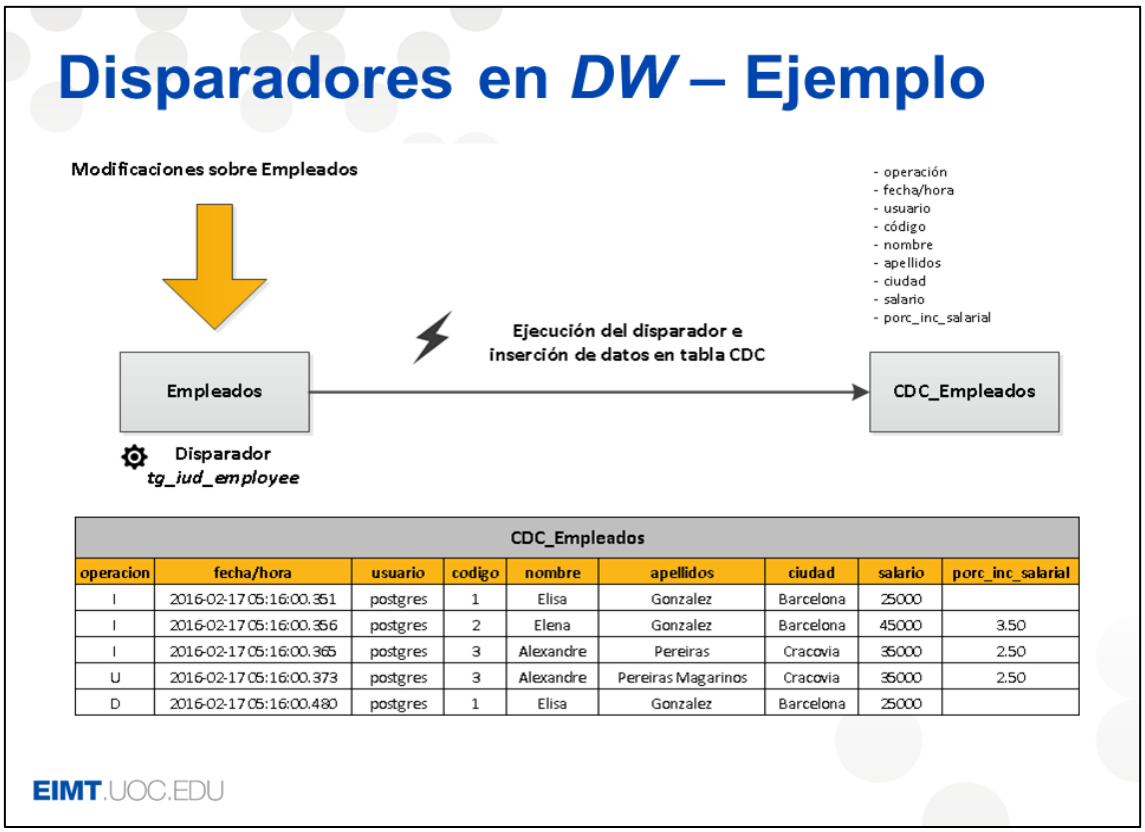
Com a regla general, en un *data warehouse* hem d'evitar els disparadors! La raó principal es deu a la naturalesa d'aquest tipus d'entorns: totes les operacions de càlcul i regles implementades es realitzen a través de processos de càrrega o ETL (en anglès, *extract, transform and load*), processos que són llançats amb una freqüència diària i que, normalment, s'encarreguen de llegir dades en els orígens, realitzar transformacions i càlculs, i carregar-les en *l'entorn data warehouse*. En general, podríem dir que l'ús assignat als disparadors en una BD operacional no ha de ser donat en un entorn *data warehouse*.

Hi ha, en canvi, una excepció a la norma comentada i per a això és necessari presentar el concepte de *change data capture* (CDC). *Change data capture* és un mecanisme d'identificació, captura i processament de dades molt utilitzat en entorns *data warehouse* per a identificar, capturar i processar canvis realitzats en els orígens de les dades, de manera que les dades que canvien en origen puguin ser actualitzades en el nostre *data warehouse*.

Hi ha diferents formes d'implementar un CDC. Entre les dues més comunes, hi ha l'ús d'una columna de tipus data/hora a cada taula que registri el moment en què una fila es modifica i l'ús d'una taula *comptable*. Les dues alternatives proposades es basen en l'ús de disparadors.

En el primer cas, la columna de tipus data/hora es podria actualitzar mitjançant la creació d'un disparador que asseguri que, per cada fila modificada, s'actualitzi el camp amb la data i hora actuals. Aquest mecanisme faria ús d'un disparador de tipus `AFTER/FOR EACH ROW` i té el desavantatge que no permet registrar les files esborrades, ja que una vegada eliminada la fila, deixa d'existir en la taula i no la podem actualitzar.

L'ús de la taula *comptable*, el segon cas esmentat, soluciona aquest desavantatge. Mitjançant l'ús d'un disparador similar al descrit en el cas anterior (`AFTER/FOR EACH ROW`), podem capturar en una taula d'auditoria o taula *comptable* els canvis realitzats en les taules origen. En aquesta taula *comptable*, es registraria l'operació realitzada (`INSERT`, `UPDATE` o `DELETE`), l'usuari i la data/hora en què s'ha realitzat l'operació (entre d'altres), a més de la informació actual (i, fins i tot, anterior, si així es desitja) de la fila afectada. D'aquesta forma, podem utilitzar aquesta informació per a identificar i capturar els canvis realitzats, sigui quina sigui l'operació, i processar-los segons les regles del nostre *data warehouse*.



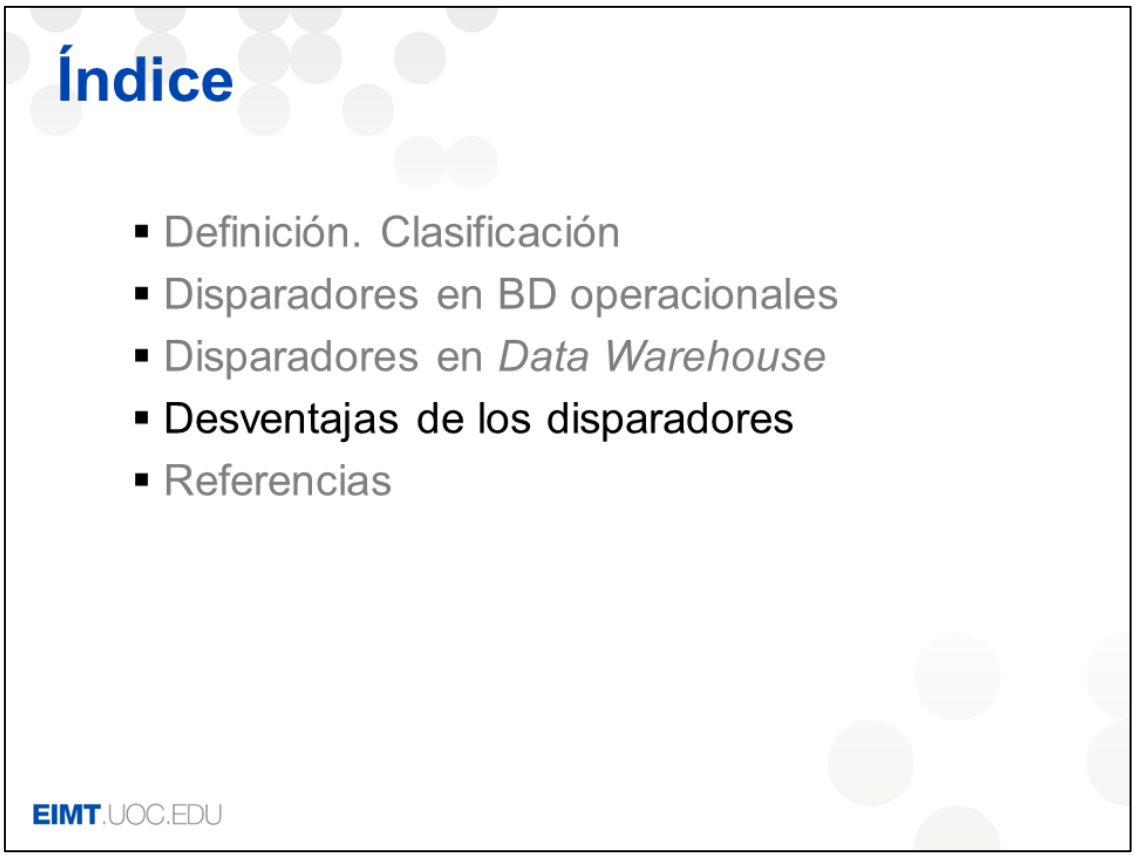
CDC_Empleados								
operacion	fecha/hora	usuario	codigo	nombre	apellidos	ciudad	salario	porc_inc_salarial
I	2016-02-17 05:16:00.351	postgres	1	Elisa	Gonzalez	Barcelona	25000	
I	2016-02-17 05:16:00.356	postgres	2	Elena	Gonzalez	Barcelona	45000	3.50
I	2016-02-17 05:16:00.365	postgres	3	Alexandre	Pereiras	Cracovia	35000	2.50
U	2016-02-17 05:16:00.373	postgres	3	Alexandre	Pereiras Magarinos	Cracovia	35000	2.50
D	2016-02-17 05:16:00.480	postgres	1	Elisa	Gonzalez	Barcelona	25000	

Vegem un exemple d'implementació d'un CDC mitjançant un disparador i una taula *comptable*, exemple que podem veure a la imatge superior.

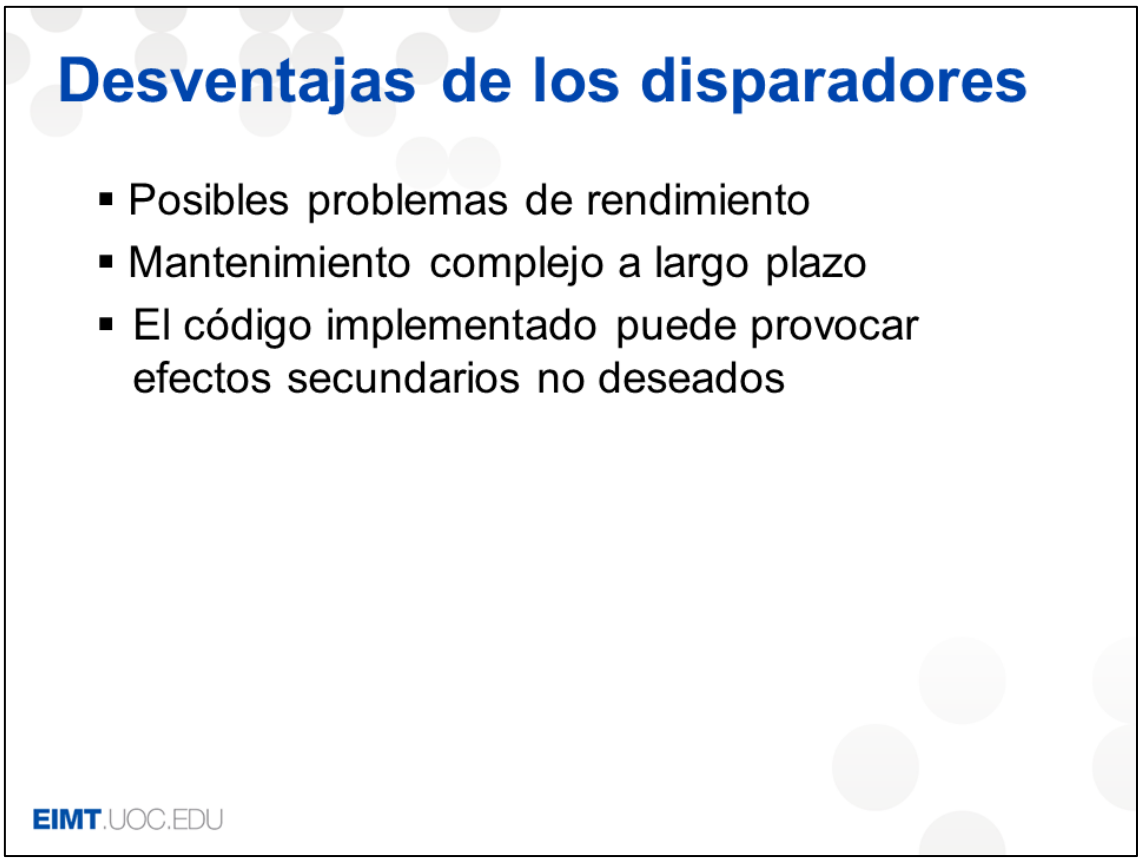
Suposem una taula d'empleats amb un disparador de tipus `AFTER/FOR EACH ROW` anomenat *tg_iud_employee* que captura totes les operacions sobre la taula d'empleats. Per a cada fila modificada en la taula d'empleats, s'executarà el disparador i s'emmagatzemarà la informació dels canvis (en el cas d'`INSERT` i `UPDATE`, la nova informació després de l'operació, i en el cas de `DELETE`, la informació actual esborrada) en una taula *comptable* denominada *CDC_Empleados*. En aquesta taula comptable, s'emmagatzema la informació de l'operació (mitjançant un caràcter, `I`, `U` o `D`), la data i hora del canvi, l'usuari de la BD que ha realitzat el canvi i totes les dades de les columnes de la taula empleats.

A la imatge inferior, podem veure el resultat de realitzar cinc operacions consecutives sobre la taula d'empleats utilitzant l'arquitectura proposada, assumint que la taula estava inicialment buida:

- Primer, s'afegeixen tres empleats (Elisa, Elena i Alexandre) mitjançant tres operacions d'`INSERT`.
- A continuació, s'actualitzen els cognoms d'Alexandre, mitjançant una operació d'`UPDATE`.
- Finalment, s'elimina l'usuari Elisa, mitjançant una operació de `DELETE`.



Finalment, presentarem una sèrie de desavantatges de l'ús de disparadors, tant en les BD operacionals com en entorns de *data warehouse*.

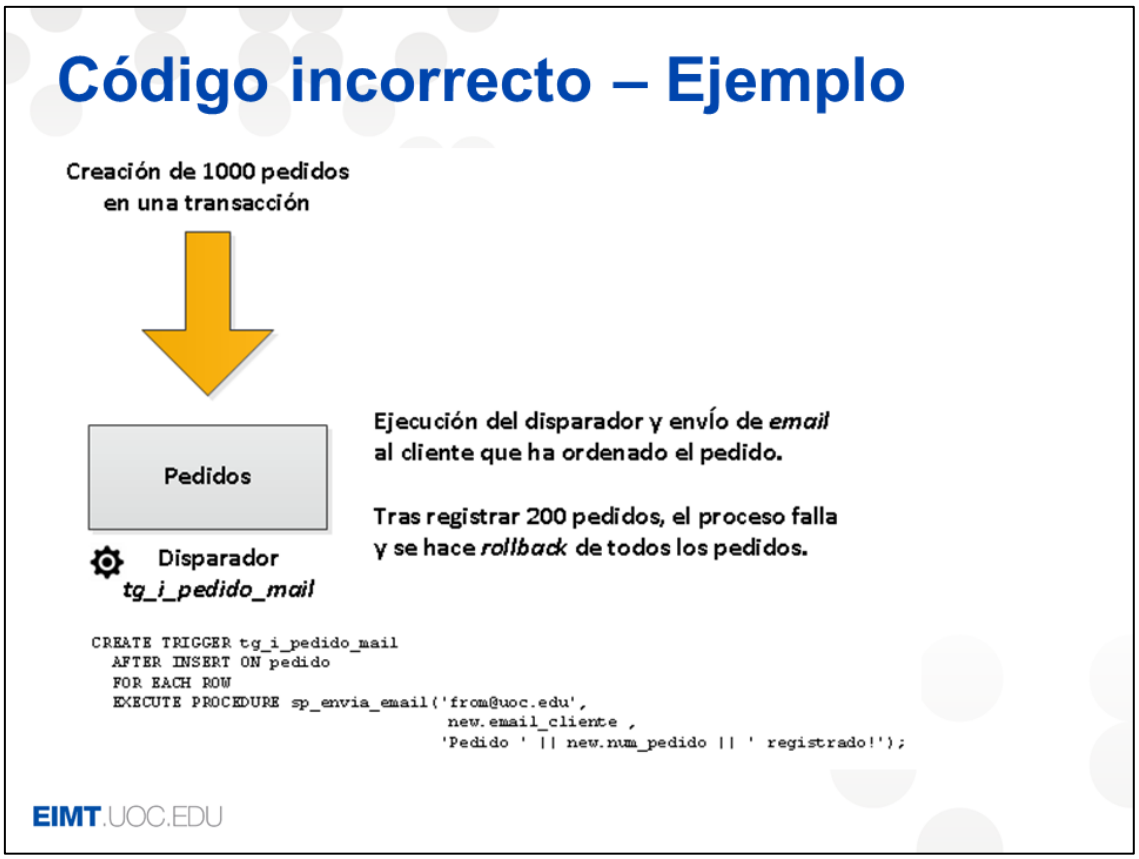


Un primer desavantatge que podem esmentar és que els disparadors poden causar problemes de rendiment si les operacions realitzades sobre la taula en què està definit el disparador afecten una gran quantitat de files. Aquest escenari es pot donar en entorns *data warehouse* quan implementem mecanismes de CDC, com hem vist anteriorment. En cas que les operacions en la BD d'origen impliquin actualitzacions massives de dades, la implementació de disparadors per a capturar les dades canviades pot no ser la millor opció, ja que poden causar problemes de rendiment que afectin les operacions sobre la BD d'origen, perjudicant els usuaris i l'activitat diària de l'empresa.

Un altre desavantatge de l'ús dels disparadors és que el manteniment a llarg termini és molt complex. Imagineu que hereteu el manteniment d'una BD operacional plena de disparadors que gestionen regles de negoci. Aquest escenari es pot convertir en un malson autèntic, ja que qualsevol operació en la base de dades pot desencadenar una multitud d'actualitzacions automàtiques i incontrolables, en molts casos sense que l'usuari o administrador ho sàpiga. Si un no està al corrent de l'existència d'aquests components o de la lògica implementada per aquests, pot resultar un problema greu en la gestió de les dades.

Finalment, és important esmentar que el codi implementat en els disparadors pot provocar efectes secundaris no desitjats si el codi no està escrit correctament des d'un punt de vista de negoci i no sintàctic o, simplement, si no s'han anticipat les

conseqüències d'aquest codi dins del context de les regles de negoci. Aquests casos solen succeir quan en el codi implementat en els disparadors invoquen funcions auxiliars proporcionades per l'SGBD, funcions per les quals no podem desfer els canvis mitjançant el `ROLLBACK`.

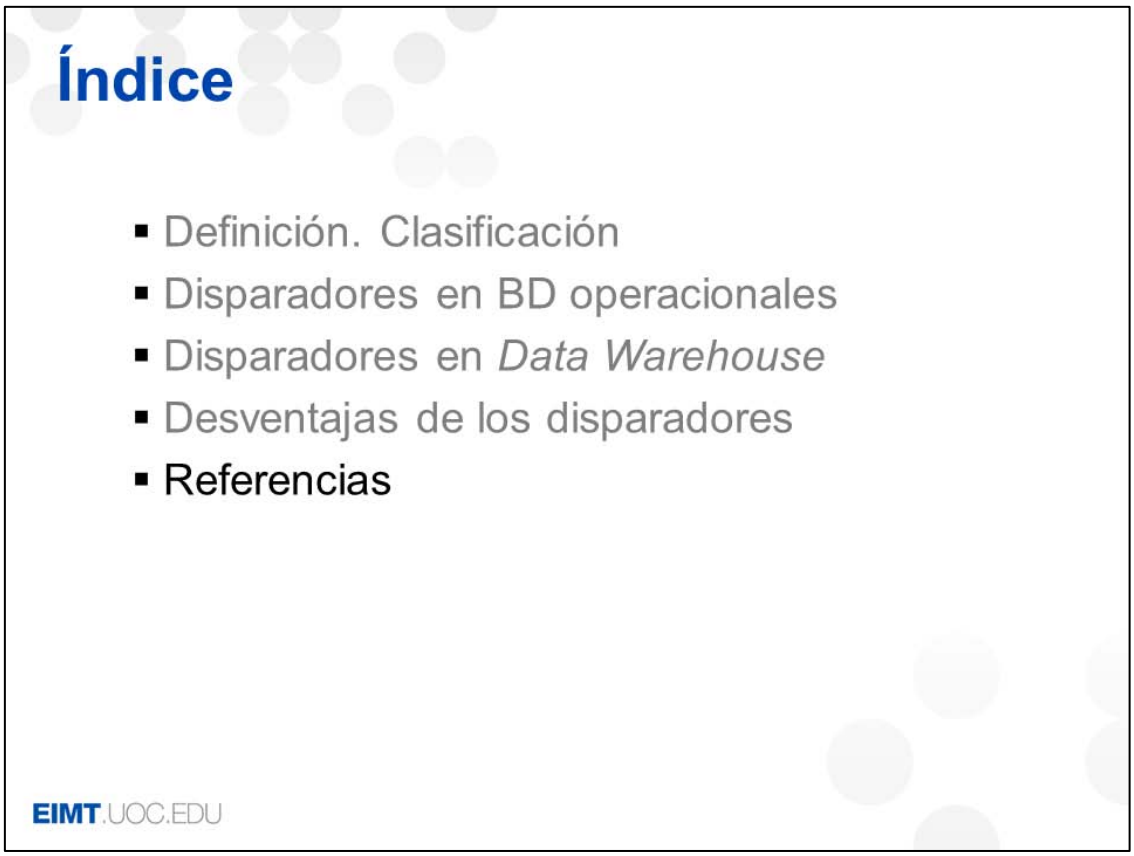


Vegem un exemple dels efectes secundaris provocats per un codi implementat dins d'un disparador.

Suposem que un procés ha de crear 1.000 comandes en la taula de comandes, on tenim un disparador creat que executa un procediment emmagatzemat que envia un *e-mail* al client una vegada que la comanda està registrada en la taula. La creació d'aquestes comandes es realitza dins d'una mateixa transacció, és a dir, o bé s'insereixen les 1.000 comandes correctament o bé no se n'insereix cap.

Quan ja s'han inserit 200 comandes i, per tant, enviat els seus corresponents *e-mails*, el procés falla i realitza un *rollback* de les comandes generades, per la qual cosa la taula de comandes tornarà a l'estat inicial. El problema rau ara en el fet que ja hem enviat *e-mails* a molts clients que les seves comandes han estat registrades!

Per tant, com podeu veure, hem de ser molt acurats amb el codi que s'implementa dins d'un disparador, analitzant minuciosament els possibles efectes secundaris que aquest pot tenir quan treballem amb transaccions en la nostra base de dades.



I fins aquí hem arribat amb aquest vídeo sobre la contextualització dels disparadors. Esperem que us hagi agradat la presentació i que aquesta us hagi servit de molta ajuda.

Ara us presentarem un conjunt d'enllaços i referències d'interès sobre aquest tema.

Que tingueu un bon dia!