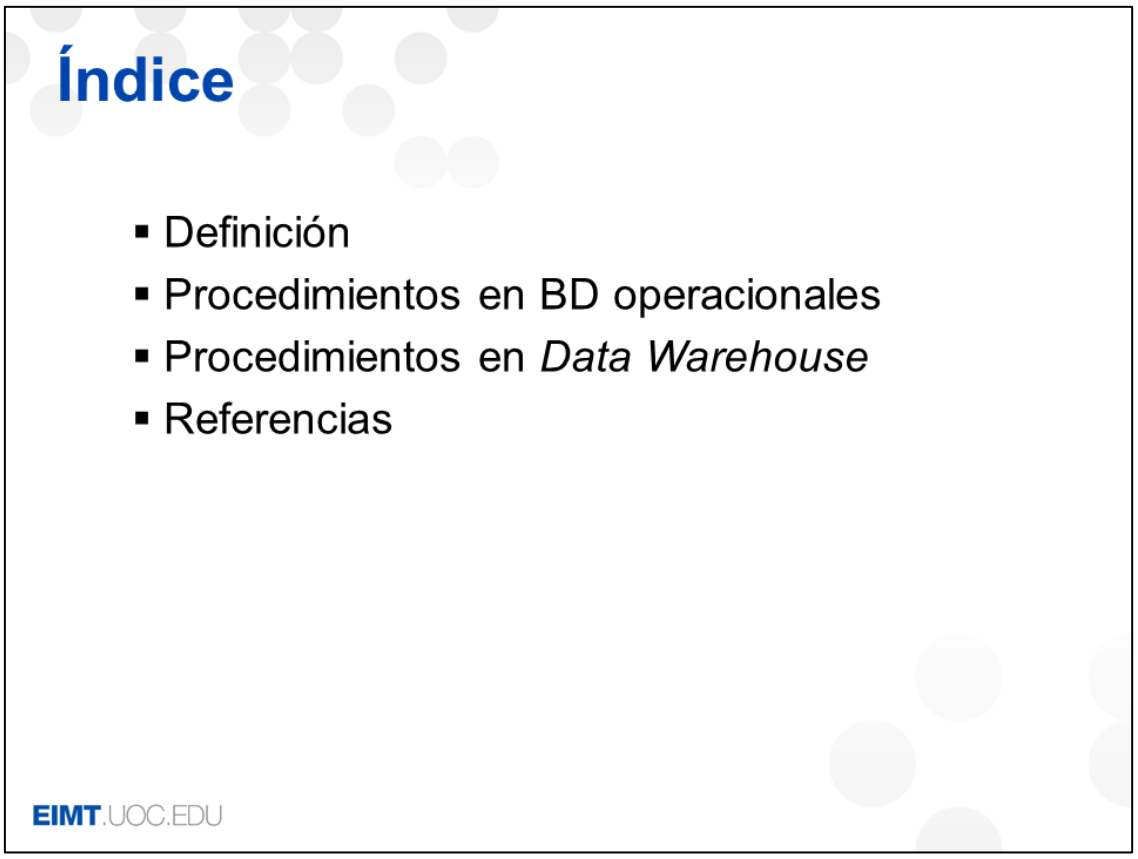
Contextualització de procediments

BD operacionals i *Data Warehouse*

PID_00253048

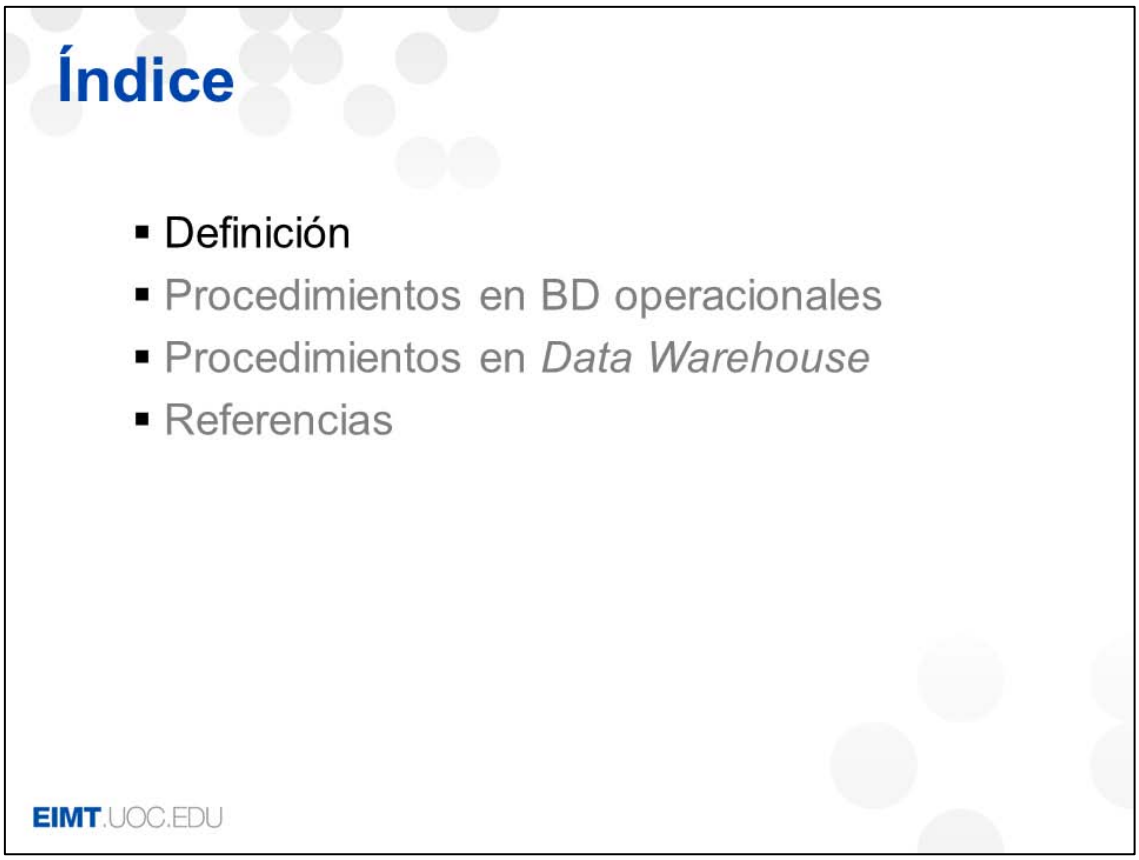
Alexandre Pereiras Magariños

Universitat Oberta de Catalunya

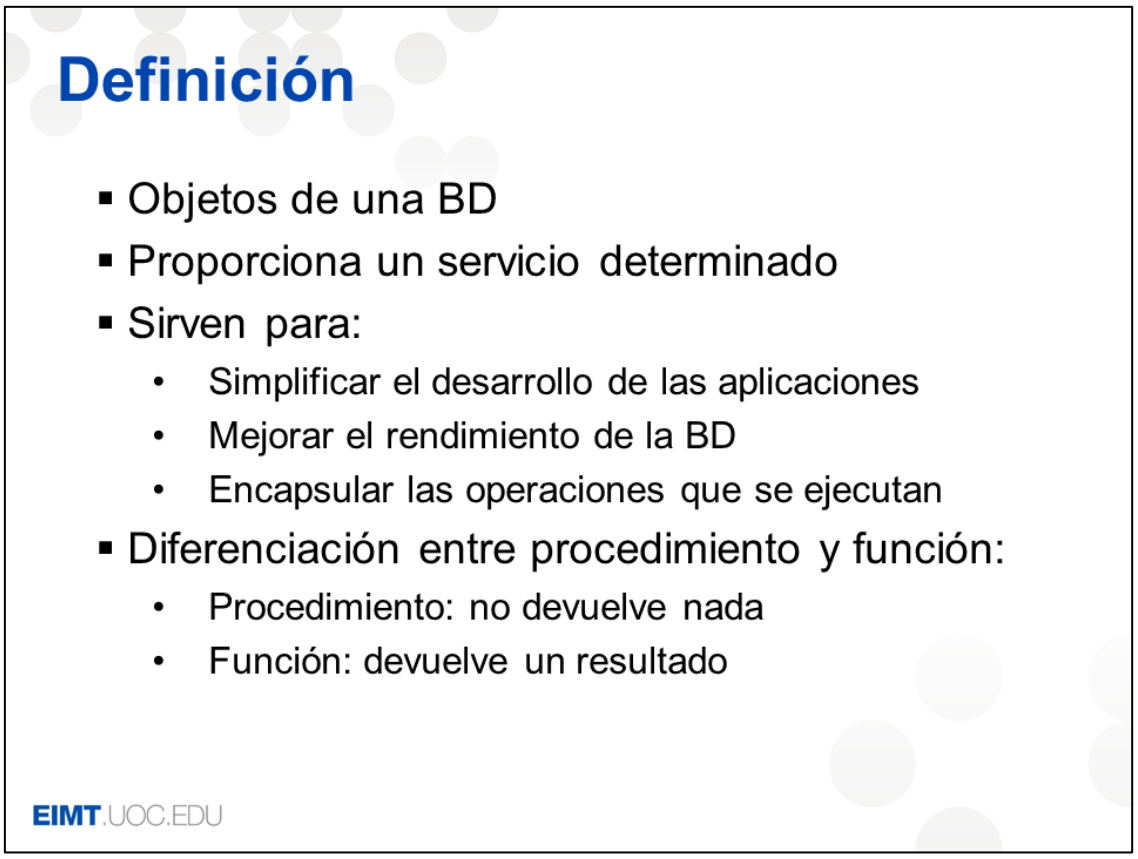


Benvinguts a aquest vídeo titulat *Contextualització de procediments*, en què comentarem i explicarem què són els procediments i com podem aplicar-los dins del marc de les bases de dades (BD) operacionals i de *data warehouse*.

Aquest vídeo introduirà breument el concepte de procediment emmagatzemat, per després proporcionar la seva contextualització des del punt de vista de les BD operacionals i del *data warehouse*, revisant els usos que aquests components ens poden proporcionar en tots dos tipus d'entorns. Des del punt de vista del *data warehouse*, també veurem una sèrie d'exemples de la gestió de transaccions i errors. Per últim, us proporcionarem les referències bibliogràfiques que hem utilitzat per a elaborar aquesta presentació.



Comencem, doncs, amb la definició de procediment emmagatzemat i les seves característiques.



Els procediments emmagatzemats són objectes de la base de dades, definits pels usuaris i que encapsulen un codi concret, proporcionant-los així un servei determinat. Aquests objectes es guarden en la BD i poden ser invocats en qualsevol moment per aquells usuaris que tinguin accés.

L'ús de procediments emmagatzemats té una sèrie d'objectius:

- En primer lloc, ens serveixen per a simplificar el desenvolupament de les aplicacions, de manera que aquestes solament hauran de realitzar una crida al procediment emmagatzemat, en lloc d'executar diferents sentències SQL una darrere l'altra. A més, la lògica relativa a la BD s'emmagatzemarà en el sistema gestor de bases de dades (SGBD), per la qual cosa s'afavoreix la reutilització del codi.

- Un altre dels objectius de l'ús dels procediments és que millora el rendiment de la BD. Hem mencionat anteriorment el cas en què s'executaven sentències SQL una darrere l'altra. L'execució d'aquestes seqüències provoca que la BD hagi de calcular l'estratègia d'execució de cadascuna d'aquestes just abans d'executar-les, mentre que en el cas dels procediments emmagatzemats, aquesta estratègia es calcula a l'hora de crear-les en la BD.

- Finalment, és important destacar que l'ús dels procediments emmagatzemats possibilita l'encapsulació del codi de les operacions que s'executen. D'aquesta

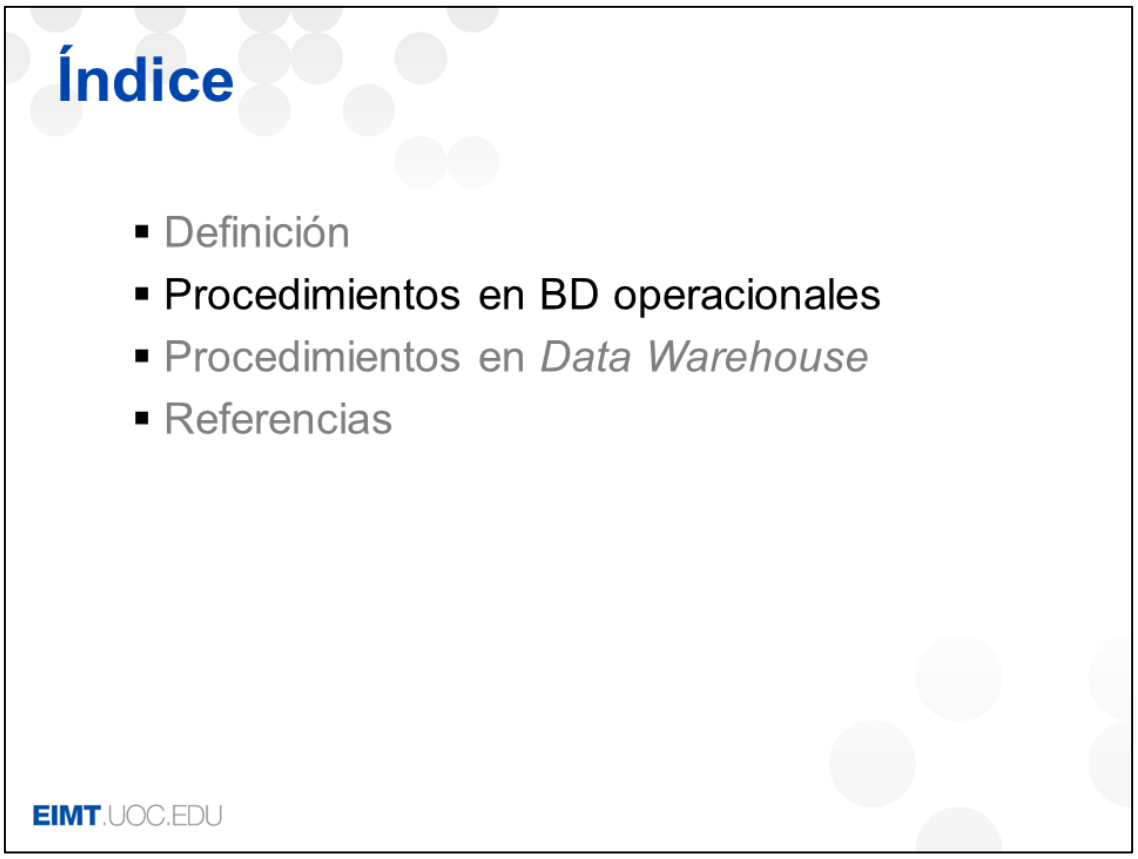
forma, l'usuari solament necessita conèixer el nom del procediment que proporciona el servei, evitant així que l'usuari hagi de conèixer les sentències SQL que el procediment emmagatzemat incorpora i els elements de la BD que manipulen aquestes sentències.

La codificació d'aquests procediments se sol realitzar mitjançant l'SQL, complementat amb un llenguatge procedimental. Per exemple, en PostgreSQL, els dos llenguatges procedimentals més utilitzats són PL/pgSQL i PL/Python.

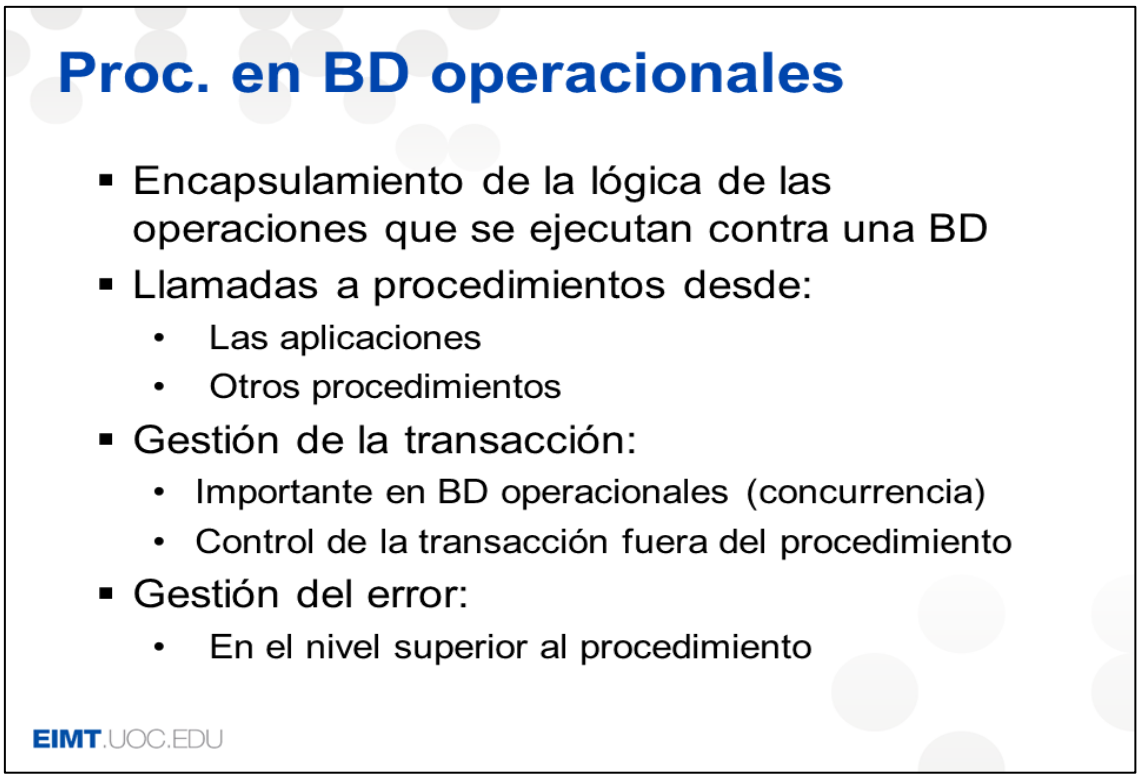
Finalment, és important esmentar que des del punt de vista de PostgreSQL, els procediments emmagatzemats es denominen funcions i poden o no retornar un resultat. Aquesta definició difereix pel que fa a altres SGBD, com ara Oracle, en què tots dos tipus requereixen clàusules SQL concretes (`CREATE PROCEDURE` o `CREATE FUNCTION`, en què la primera no retorna cap resultat i la segona sí) en oposició amb PostgreSQL (`CREATE FUNCTION [...] RETURNS [...]`). Amb la finalitat de diferenciar tots dos tipus en PostgreSQL, hem decidit denominar com a **procediment** aquells procediments emmagatzemats que no retornen cap resultat i com a **funció** aquells procediments emmagatzemats que sí que retornen un resultat.

En aquesta transparència, podem veure un exemple de procediment emmagatzemat en PostgreSQL creat amb el llenguatge procedimental PL/pgSQL.

El procediment mostrat, denominat `sp_carga_prestamo()`, realitza la càrrega de les dades dels préstecs des d'una taula anomenada `prestamo_tmp` a una altra taula denominada `prestamo_fact`. El procediment recorre fila a fila la taula `prestamo_tmp` i intenta inserir la fila en la taula de destí. En cas que l'operació d'inserció produeixi un error de clau única violada (és a dir, que la fila ja existeixi en la taula), el procediment s'encarregarà d'actualitzar aquesta fila concatenant al nom del llibre una cadena de caràcters formada per guions, a més d'actualitzar l'identificador i l'autor del llibre. Si l'error produït no és de clau única violada, llavors el procediment no fa res (codificat com un bloc buit amb el comentari «*Insertar en log*») i continua amb la fila següent, fins que es processi l'última fila de la taula, moment en què finalitza el procediment.



A continuació, vegem en quins supòsits podem usar els procediments des del punt de vista de les BD operacionals.



Dins del context de les BD operacionals, els procediments emmagatzemats, com hem esmentat anteriorment, ens resulten de molta utilitat per a encapsular la lògica de les operacions que s'executen contra una BD. Aquesta encapsulació permet disminuir el tràfic a la xarxa, ja que evita que cada sentència SQL s'enviï d'una manera individual per la xarxa i que els resultats de l'execució de cadascuna de les sentències siguin recollits per l'aplicació. Amb l'ús de procediments emmagatzemats, tota la lògica s'executarà en l'SGBD sense necessitat d'utilitzar la xarxa de forma innecessària, retornant els resultats una única vegada a l'aplicació. D'aquesta forma, els usuaris solament necessiten saber el nom del procediment que han d'executar amb la finalitat d'obtenir el servei esperat.

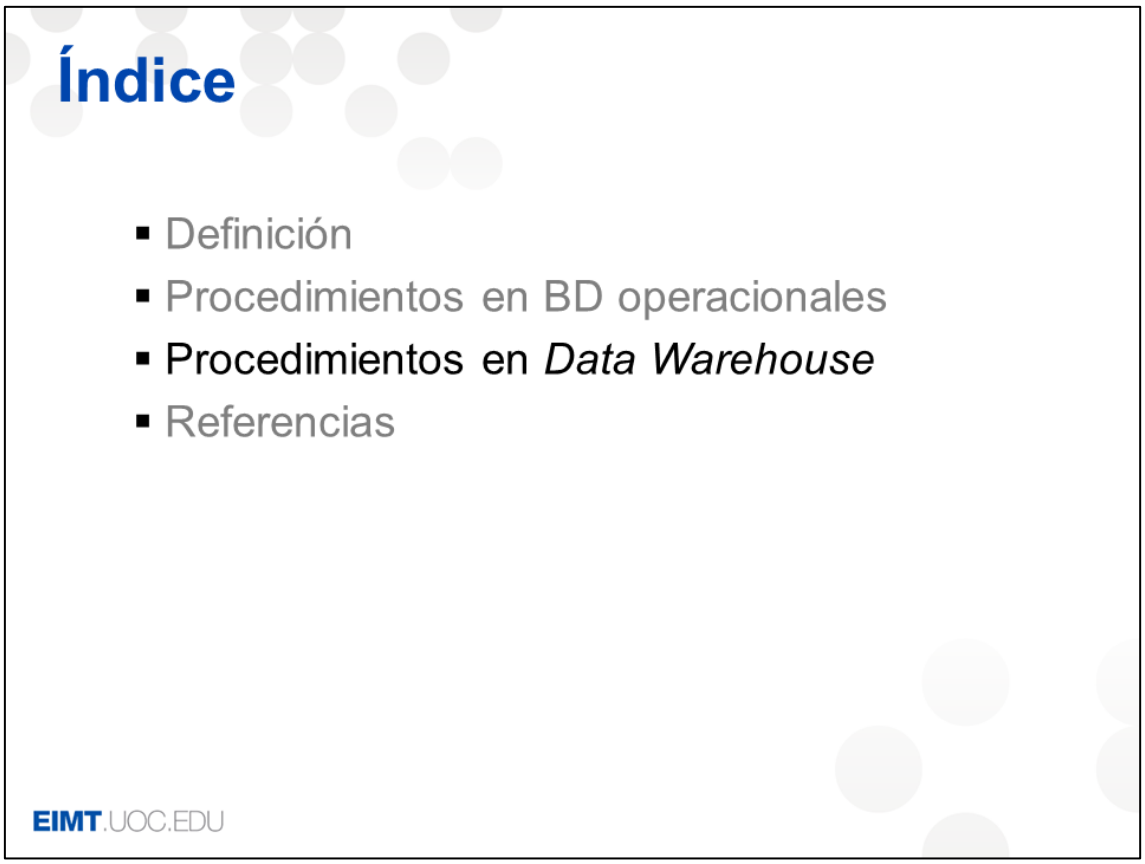
La crida a un procediment es pot realitzar de les formes següents, a més de l'opció de cridar-lo des de la consola o línia de comandaments que es connecta a una BD:

- Crides des de les aplicacions: en aquests casos, el codi de les aplicacions ha d'establir una connexió amb la base de dades i realitzar la crida al procediment emmagatzemat.

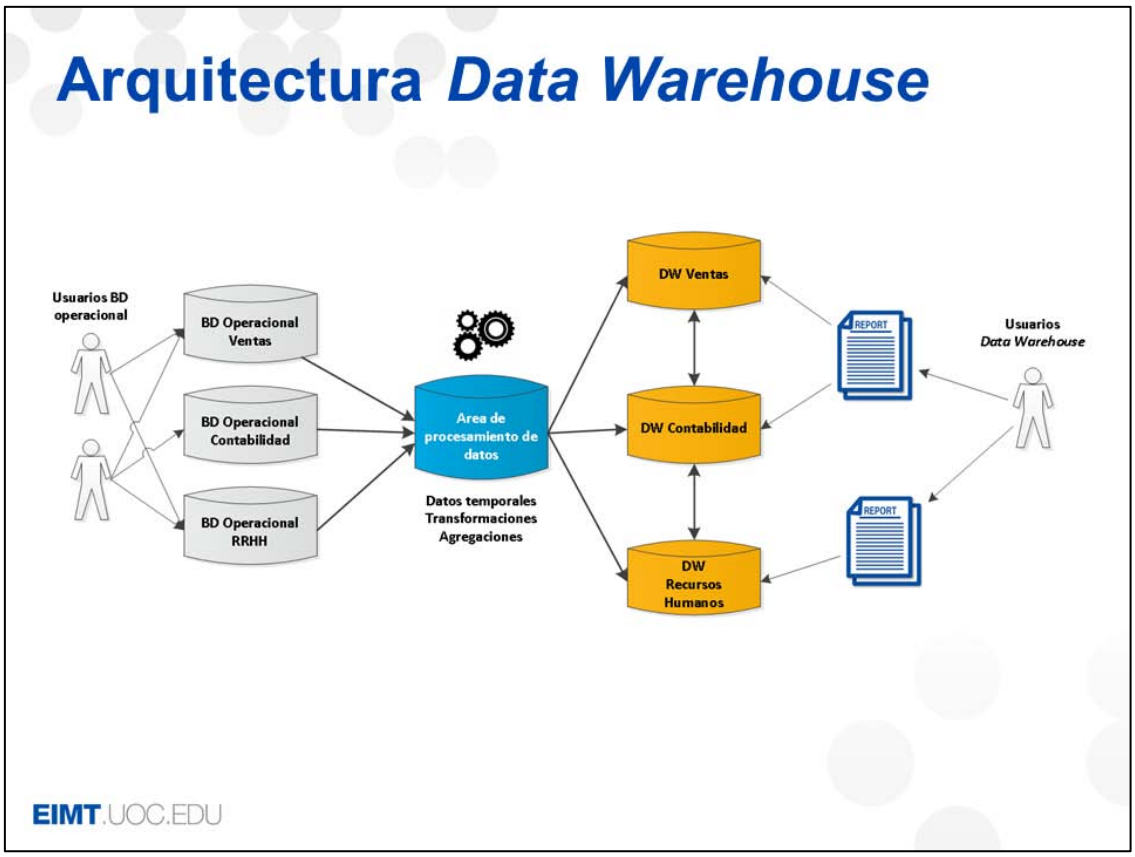
- Altres procediments: aquesta opció és comuna quan volem implementar un servei determinat que sigui utilitzat per múltiples procediments. Un exemple de crida a un procediment dins d'un altre procediment seria en el cas de generar informació d'auditoria (el procés de generació i actualització d'informació d'auditoria se sol implementar de forma separada mitjançant un conjunt de procediments que, al seu torn, són cridats per aquells procediments que s'encarreguen d'actualitzar la nostra BD).

Un punt molt important a tenir en compte en l'ús dels procediments emmagatzemats en les BD operacionals és la gestió de les transaccions. En aquest tipus de BD, és molt significativa la concurrència entre usuaris per a poder mantenir les dades consistents mentre aquests llancen transaccions alhora que poden llegir i actualitzar les dades al mateix temps. En aquest tipus d'entorns, és important que la transacció sempre es generi fora del procediment i que aquesta estigui gestionada per les aplicacions. És a dir, és responsabilitat del codi de l'aplicació iniciar i finalitzar la transacció, sia mitjançant la confirmació de les dades o desfent-les fins a tornar a l'estat inicial.

Un últim punt a considerar en l'ús dels procediments emmagatzemats en les BD operacionals és la gestió de l'error. Què passa si, per alguna raó, es produeix un error durant l'execució d'un procediment? Què fem quan ens trobem amb un error? En aquests casos, sempre hem de gestionar l'error en el nivell superior, independentment de si és un procediment o l'aplicació en si. El nivell superior ha de saber si l'execució del procediment emmagatzemat finalitza o no en una situació d'error, perquè en cas contrari, la BD pot quedar en un estat inconsistent. En cas que es produeixi una situació d'error, en general, el nivell superior haurà de cancel·lar la transacció que invoca l'execució del procediment emmagatzemat.



Una vegada contextualitzats els procediments emmagatzemats en les BD operacionals, vegem la contextualització d'aquests components des del punt de vista del *data warehouse*.



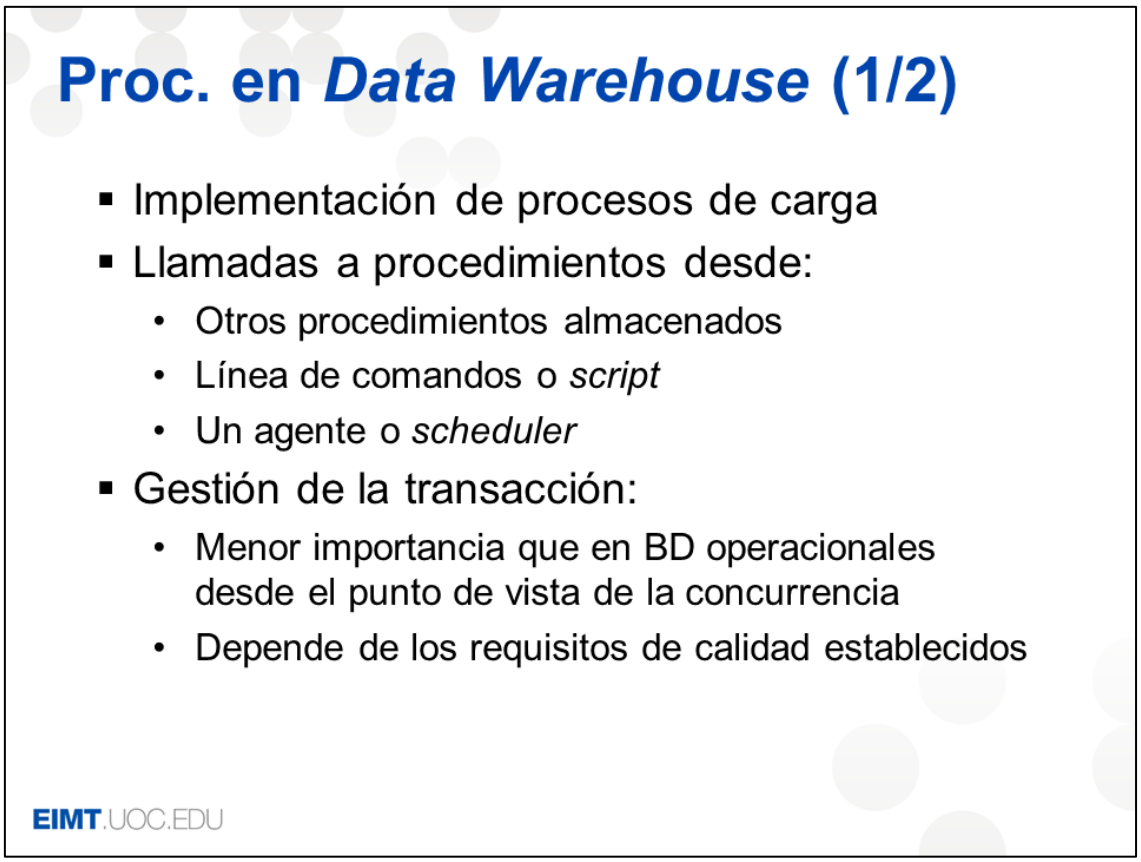
Abans de realitzar la contextualització dels procediments, vegem un exemple típic de l'arquitectura *data warehouse*, exemple que es presenta a alt nivell sense arribar a entrar en detalls. Els elements típics d'una arquitectura *data warehouse* són els següents:

- En primer lloc, tenim les BD operacionals. Aquestes BD són accedides per molts usuaris que realitzen milers de transaccions de lectura i actualització de dades, generalment afectant desenes o centenars de files. Aquestes BD que solen estar optimitzades per a permetre un grau de concurrència alt (milers d'usuaris hi accedeixen) i actuen com a origen de les dades per al nostre *data warehouse*, ja que contenen la informació sobre els processos de negoci de la nostra empresa. En alguns casos, en lloc de BD operacionals també podem tenir com a origen fitxers de text, XML o qualsevol altre format.

- Com a segon component, tenim el que denominem àrea de processament de dades. Es tracta d'una zona de treball intermèdia, que s'encarrega de llegir les dades de les BD operacionals i emmagatzemar-les de forma temporal. Aquestes tasques són dutes a terme pels processos de càrrega o ETL (*extract, transform and load*). En aquesta zona temporal, les dades són transformades, millorades segons els processos de qualitat establerts i, fins i tot, poden ser agregades. Cap usuari no hauria de tenir accés a aquesta àrea, ja que aquesta

està dedicada solament per a ser utilitzada pels processos de càrrega i, puntualment, pels operadors del *data warehouse*.

- El tercer component ja és el *data warehouse*, en què s'emmagatzemen les dades ja transformades i agregades, preparades per a ser consumides pels usuaris finals. El nombre d'usuaris d'un *data warehouse* sol ser molt menor comparat amb el nombre d'usuaris de les BD operacionals (de l'ordre del centenar, generalment), i aquests solament realitzen consultes de dades (lectures) sobre una gran quantitat de dades. Un usuari del *data warehouse* mai no podrà realitzar actualitzacions, ja que són els processos de càrrega o ETL els encarregats de realitzar aquestes tasques a partir de les BD operacionals, tal com hem vist anteriorment.



A l'hora de contextualitzar els procediments en un entorn *data warehouse,* podem dir que, des d'aquest punt de vista, els procediments emmagatzemats ens resulten molt útils per a desenvolupar i implementar processos de càrrega si no disposem d'una eina ETL apropiada. És molt comú codificar procediments emmagatzemats per a la càrrega de taules de dimensions i fets, a més de gestionar les càrregues de dades temporals, entre les BD origen (o BD operacionals) i destí (*data warehouse*).

La crida a procediments emmagatzemats en aquest tipus d'entorns se sol realitzar de les maneres següents:

- Crida des d'altres procediments: igual que hem comentat anteriorment, és possible que necessitem un servei concret des de diferents punts d'un procés de càrrega, com és el d'emmagatzemar dades d'auditoria. Per tant, la crida a procediments des d'altres procediments és un mecanisme bastant comú en el desenvolupament de processos de càrrega.

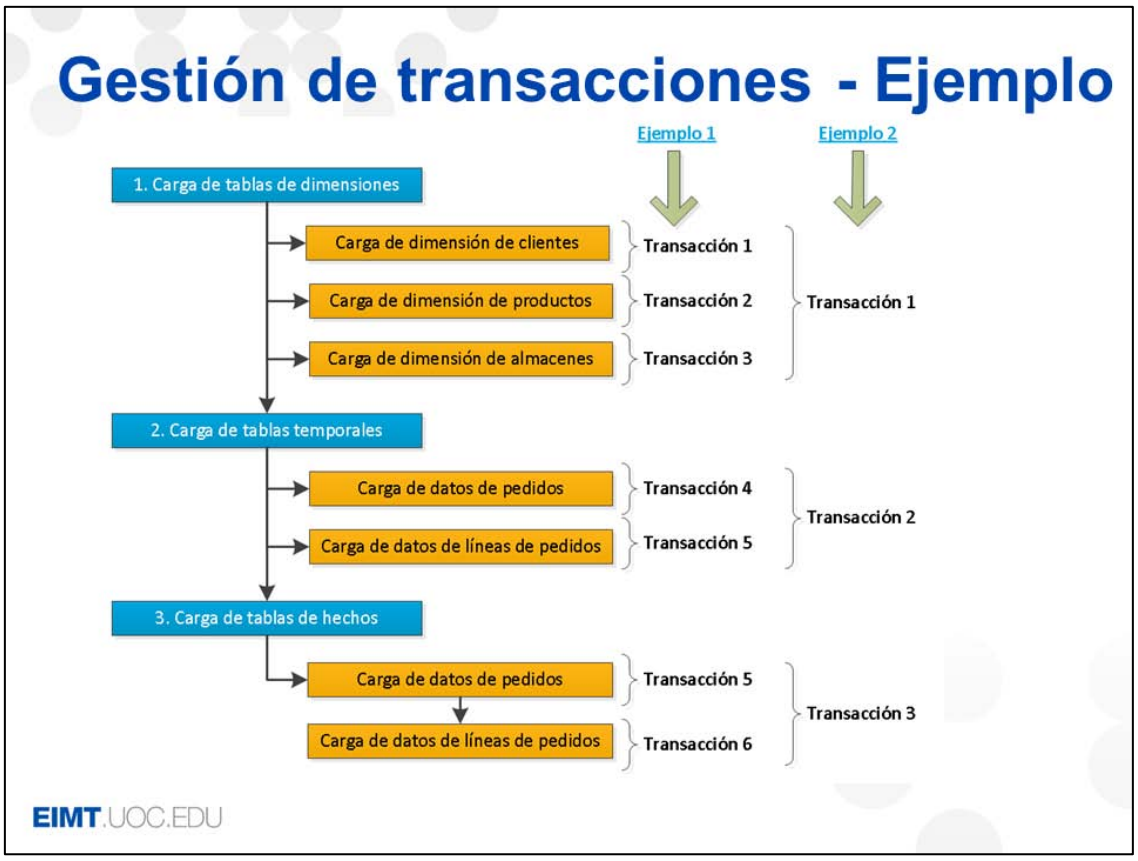
- Crida des de la línia de comandaments o des d'un *script*: la crida des de la consola sol ser comuna quan necessitem corregir càrregues de dades, és a dir, necessitem realitzar una tasca específica que és part del procés general. També es poden fer cridades des d'un *script* SQL, com a part d'una sèrie de sentències SQL.

- Crida des d'un agent o *scheduler*: els processos de càrrega de dades en *data warehouse* solen ser processos que estan programats per a ser llançats a una determinada hora del dia, sia a la nit quan no hi ha usuaris o, fins i tot, durant el dia a intervals constants de temps. En aquests casos, la programació del procés se sol realitzar mitjançant un agent o *scheduler*, que generalment ens permet executar sentències SQL i realitzar així crides a procediments emmagatzemats.

A diferència de l'esmentat anteriorment en les BD operacionals, la gestió de les transaccions en processos de càrrega és menys important des del punt de vista de la concurrència. Les transaccions, si bé són necessàries per a confirmar les dades en el *data warehouse*, poden ser implementades de manera que aquestes tinguin una durada més llarga o, fins i tot, amb un nivell d'aïllament inferior al que es podria requerir en una BD operacional. També solen ser utilitzades com a punts de control específics per a la recuperació de processos de càrrega. D'aquesta forma, arribats a un punt concret, sabem que podem rellançar la càrrega sense problemes i sense haver de manipular les dades de forma manual.

Recordem que, per naturalesa i com hem esmentat anteriorment, una BD operacional requereix accessos de lectura i escriptura per part de milers d'usuaris, mentre que en el *data warehouse*, l'accés a les dades és de lectura per part d'uns centenars d'usuaris com a molt. Aquí radica la diferència i perquè la gestió de les transaccions en *un data warehouse* es podria relaxar en comparació amb una BD operacional.

De totes maneres, la gestió de les transaccions en un procés de càrrega codificat mitjançant procediments emmagatzemats depèn, en gran manera, dels requisits de qualitat establerts en el projecte en què estiguem treballant.



Vegem un exemple de gestió de transaccions en un *data warehouse*.

Suposem que tenim un procés de càrrega en un *data warehouse* que executa cada nit els processos següents de forma seqüencial:

1. Càrrega de taules de dimensions: té com a tasques la càrrega de clients, productes i magatzems.

2. Càrrega de taules temporals: el seu objectiu és el d'obtenir les comandes i línies de comandes que han estat creades o actualitzades recentment (per exemple, en els últims 3 dies). Com a tasques té la càrrega de comandes i la càrrega de línies de comandes, tasques que es realitzen dins de l'àrea de processament de dades.

3. Càrrega de taules de fets: són les dues tasques que podem veure a la part inferior de la imatge.

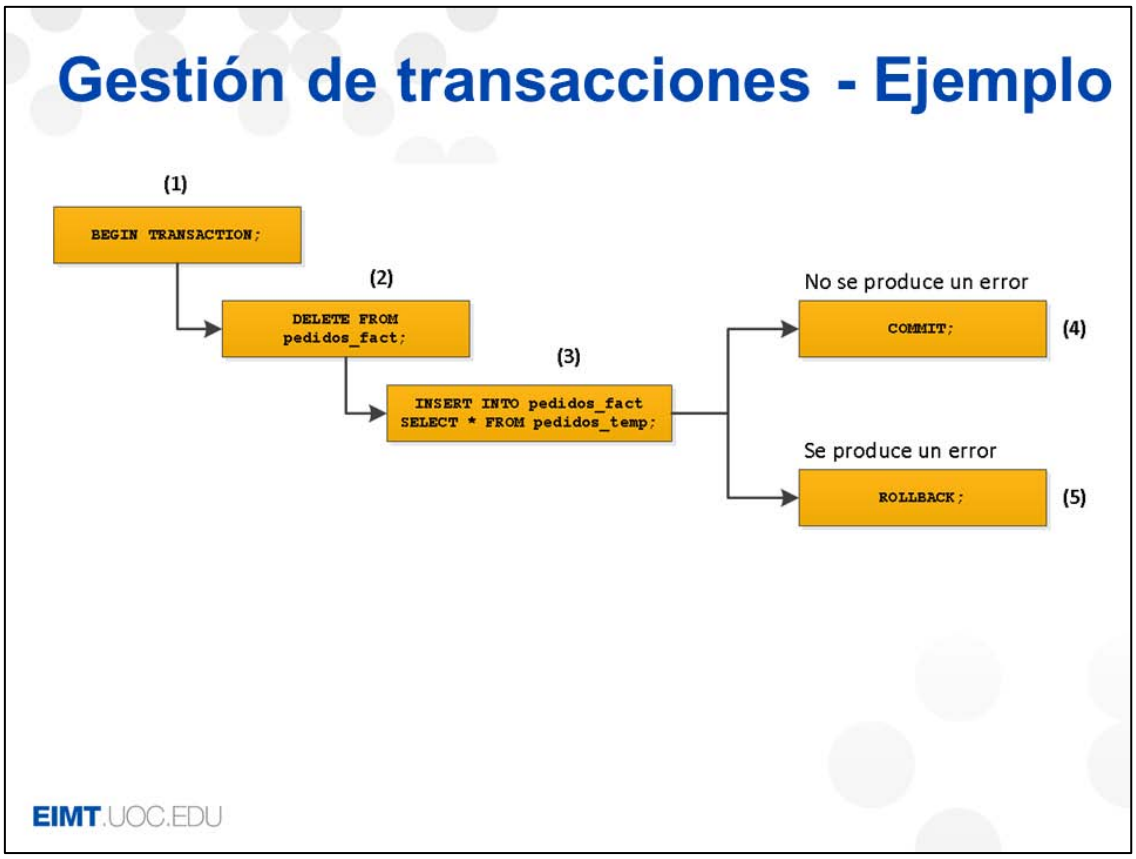
Des del punt de vista de la concurrència, podem suposar que no tindrem usuaris connectats mentre el procés de càrrega s'està duent a terme, per la qual cosa podem establir transaccions més llargues si així ho desitgem (seria l'exemple d'un procés de càrrega en mode *batch* dut a terme durant la nit, quan no tenim usuaris connectats). Si el que necessitem és implementar processos de càrrega en temps real (mentre els usuaris estan connectats), podria ser necessari realitzar transaccions més curtes per a no competir en recursos amb les lectures de dades massives per part dels usuaris.

Per tant, d'acord amb l'exemple presentat, podríem pensar en establir una transacció per a cadascuna de les tasques que s'executen en els processos anteriorment citats, com es veu en l'exemple número 1. D'aquesta manera, garantirem que les dades estiguin disponibles després de la finalització de cada tasca. En aquest cas, la transacció tendeix a ser curta en termes de tasques i temps, encara que depèn en gran manera de la quantitat de dades a actualitzar en cadascuna de les tasques implementades.

En l'exemple número 2, veiem que la transacció s'estableix per a totes les tasques de cadascun dels processos. Per tant, la transacció és més llarga i les dades solament estaran disponibles quan finalitzi el procés en qüestió, i no quan finalitza la tasca.

En qualsevol dels casos proposats, és necessari establir un equilibri entre la longitud de la transacció, el rendiment dels processos de càrrega i els requisits de qualitat establerts pels usuaris o pel projecte. És possible que el temps del procés duri menys si establim les transaccions per tasques (com en l'exemple número 1) que si establíssim una transacció per procés (com en l'exemple número 2), ja que en el primer cas, l'SGBD disposarà de més recursos lliures (com per exemple, espai temporal per a consultes) i les taules no romandran bloquejades per un espai de temps més llarg, com en el cas del segon exemple.

En la vida real, se sol implementar l'exemple número 1, fins i tot considerant una divisió de la tasca en múltiples transaccions, depenent del complicat que sigui el procés de càrrega implementat.



Un escenari especial en relació a les transaccions es produeix quan volem reemplaçar els continguts d'una taula per uns altres mitjançant la realització d'un esborrament i una càrrega completa (és a dir, eliminació completa de les dades i recàrrega completa amb les dades noves just a continuació). Aquest escenari es pot veure a la imatge mostrada.

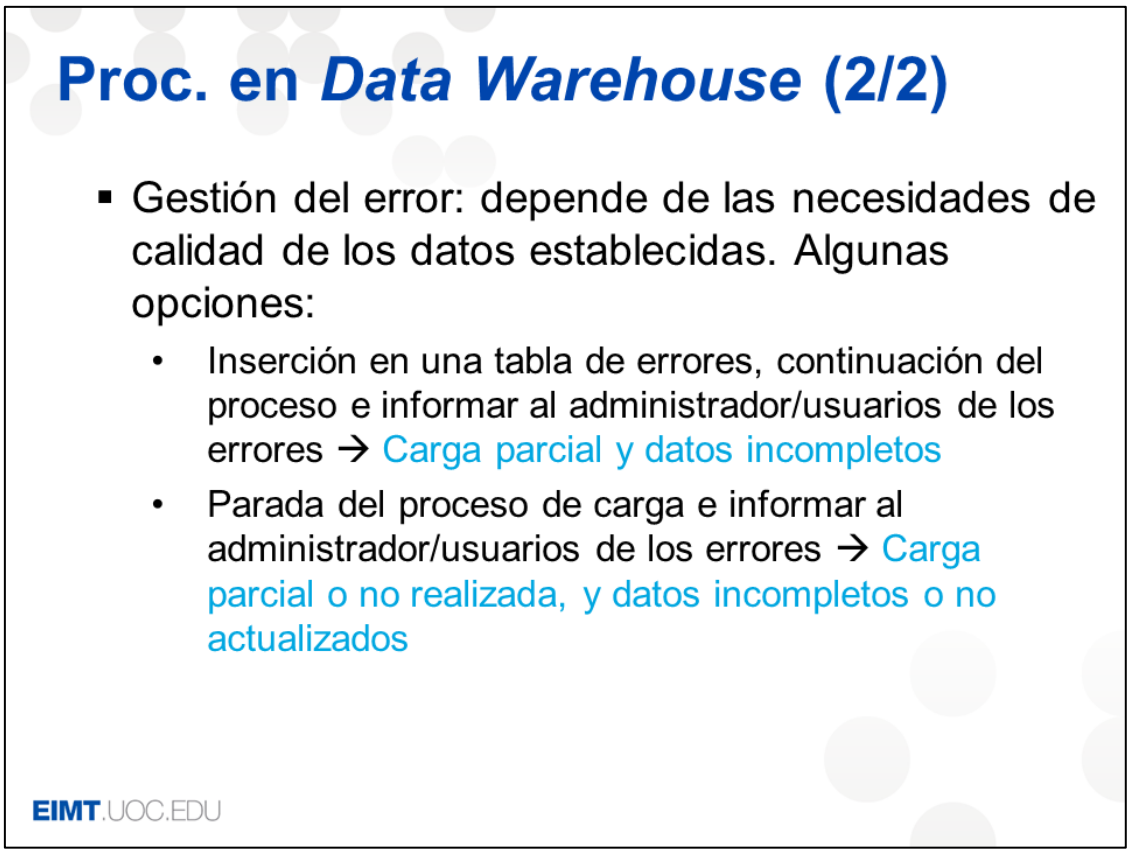
Moltes vegades en entorns *data warehouse* es requereix que la taula de fets sempre estigui carregada amb dades perquè els usuaris segueixin accedint a aquestes, sia com a requisit de l'aplicació o simplement per a disposar d'informació per a la realització d'informes. L'ús de les transaccions és molt necessari quan ens trobem amb aquesta situació.

Per a gestionar aquest escenari de manera eficient, la transacció se sol iniciar (punt 1) just abans d'eliminar les dades de la taula de fets (punt 2) i es confirma (punt 4) just després de recarregar les dades en la mateixa taula a partir d'una taula temporal (punt 3). En cas que es produeixi un error en la càrrega, la transacció desfaria els canvis (punt 5) i tornaria a l'estat inicial, garantint que mentre la transacció s'està duent a terme, la taula de fets seguirà disposant de les dades antigues perquè els usuaris segueixin fent ús d'aquestes, per la qual cosa la taula de fets mai no estarà buida.

Recordeu que en un *data warehouse*, l'accés a les dades per part dels usuaris és de lectura i mai d'escriptura, de manera que la transacció de recàrrega de les dades en la

taula de fets mai no competirà amb altres transaccions que modifiquin les dades en la mateixa taula.

Aquest tipus de càrrega de dades se sol denominar **tot o res** (és a dir, o es carrega tot el conjunt de dades o no es carrega res, tornant a l'estat inicial en cas que no es pugui realitzar una càrrega completa).

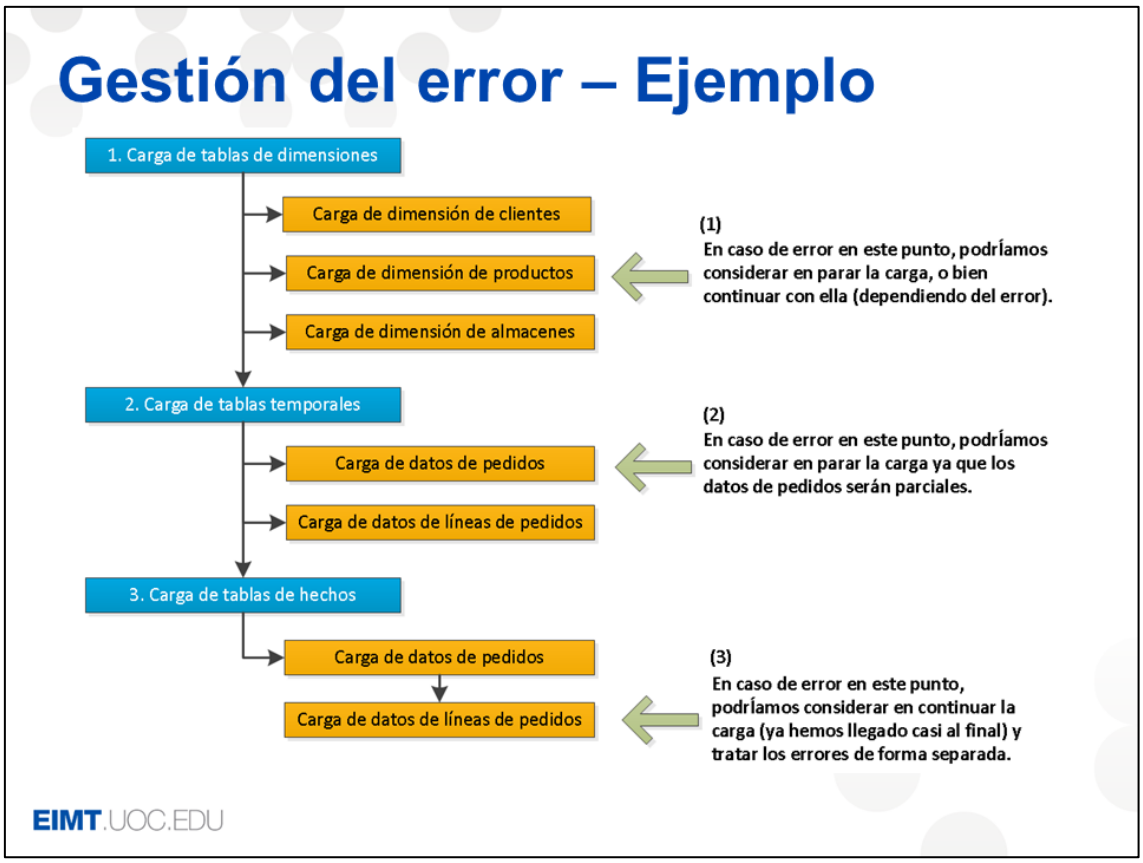


Continuant amb la contextualització dels procediments en un *data warehouse*, passem a tractar la gestió de l'error. Aquest punt és molt important en els processos de càrrega, ja que depèn en gran manera dels requisits de qualitat de les dades imposades com a part del projecte. Com a exemples de tractament de dades (que no són tots), proposem els següents:

- Capturar l'error en una taula d'errors i continuar la càrrega de dades. En aquests casos, s'informa l'administrador i els usuaris, i es gestionen els errors generalment de forma manual per part d'un operador. Aquests errors es poden produir per molts motius: falta de dades, problemes amb nuls, duplicats, etc., encara que es pot donar el cas de tractar-se d'errors amb la connectivitat de la xarxa o, fins i tot, errors en el maquinari. El fet de continuar amb la càrrega de dades (si és possible) produeix que la càrrega es consideri parcial i que les dades siguin incompletes (a l'espera de la correcció de les dades).

- Parar el procés de càrrega. En aquests casos, ens arrisquem a no realitzar cap actualització de dades, especialment si l'error es produeix a l'inici de la càrrega. Això, encara que sembli un problema, pot ser un avantatge sobre les càrregues parcials, ja que les dades no estan a mig fer sinó que simplement encara no han estat carregades. En moltes ocasions, n'hi ha prou amb corregir l'error i rellançar de nou el procés de càrrega. Si l'error es produeix cap al final,

llavors ens trobarem amb la problemàtica de la càrrega parcial i de presentar dades incompletes.



Vegem, utilitzant el mateix exemple de processos i tasques vist anteriorment, què passaria en els supòsits següents sobre la base del que hem comentat sobre la gestió dels errors:

1. Es produeix un error en la càrrega de productes. En aquests casos, podem realitzar el següent:

   a. Parar la càrrega i notificar-ho a l'administrador i als usuaris. Se sol seguir aquesta opció si, com a requisit, és necessari que hi hagi productes per a carregar les línies de comanda (és a dir, que la càrrega de línies de comanda generi un error en cas que un producte no es trobi en la dimensió de productes). En aquest cas, la càrrega de dades no està realitzada, per la qual cosa les noves dades de comandes no estaran disponibles per a l'usuari.

   b. Inserir les dades en una taula d'errors i continuar del procés. Els errors seran tractats una vegada notificats, però el procés de càrrega continua. Aquest mecanisme seria suficient si tenim una gestió de l'error similar en la càrrega de línies de comandes quan els productes no existeixen (és a dir, el procés de càrrega no es para si no tenim en la dimensió el producte associat a la línia de comanda). En aquest cas, disposarem d'una càrrega parcial, per tant, les dades estaran incompletes per als usuaris.

2. Es produeix un error en la càrrega de dades de comandes modificades. En aquests casos, podem realitzar el següent:

    a. Parar la càrrega i notificar-ho a l'administrador i als usuaris. Es recomana aquesta acció si la taula de fets de línies de comanda té una clau forana a la taula de fets de les comandes. També es recomana aquesta opció si es prefereix no disposar de dades incompletes de comandes.

    b. Igual que l'explicat anteriorment, podem inserir les dades en una taula d'errors i processar-les de forma diferent una vegada acabada la càrrega, deixant les dades de comandes com a incompletes.

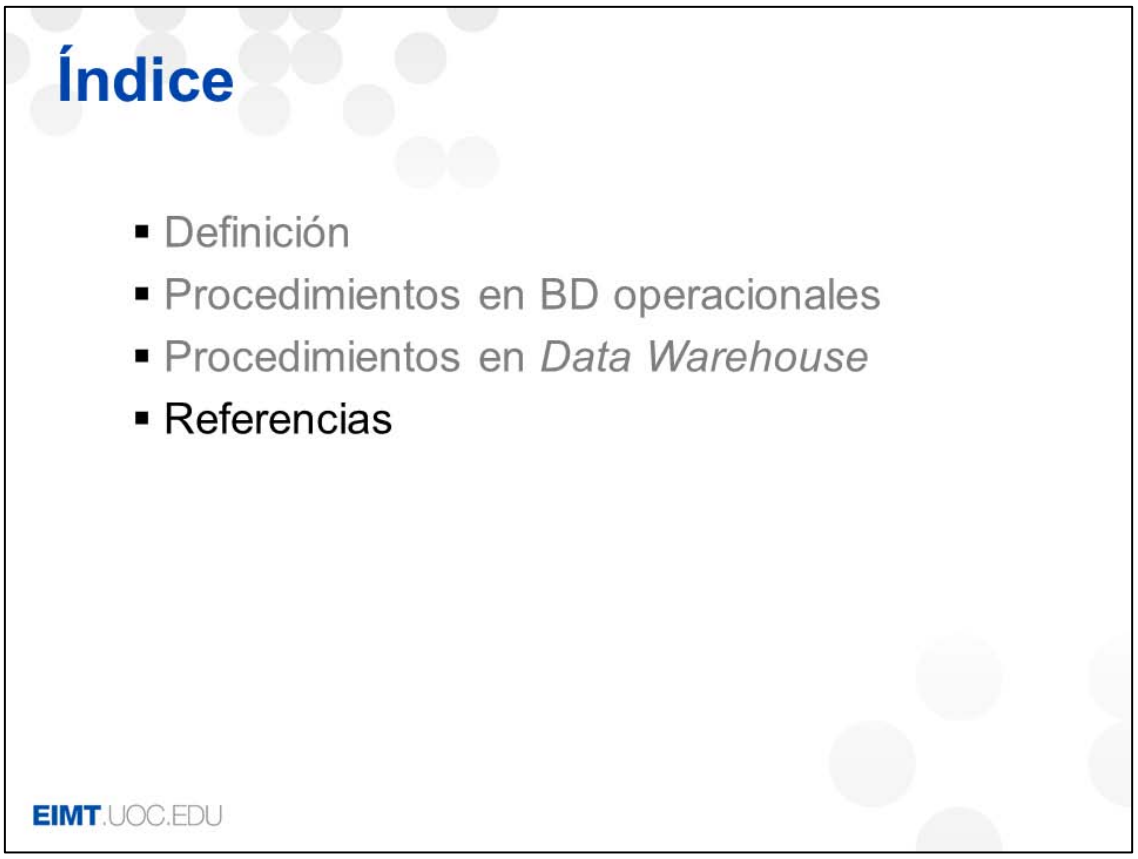
3. Finalment, es produeix un error en la càrrega de dades de les línies de comandes (fets). En aquests casos podem realitzar el següent:

    a. De nou, parar la càrrega i notificar-ho a l'administrador i als usuaris. Es recomana aquesta acció si desitgem no tenir dades parcials o incompletes. Es podria requerir, des del punt de vista de l'usuari, que es tornés a l'estat inicial (fet que suposaria guardar, d'alguna manera, l'estat anterior de la taula o dels registres modificats).

    b. Inserir les dades en una taula d'errors i processar-les de forma separada, continuant amb la càrrega. De nou, l'usuari pot requerir les dades en el seu estat inicial en cas d'error.

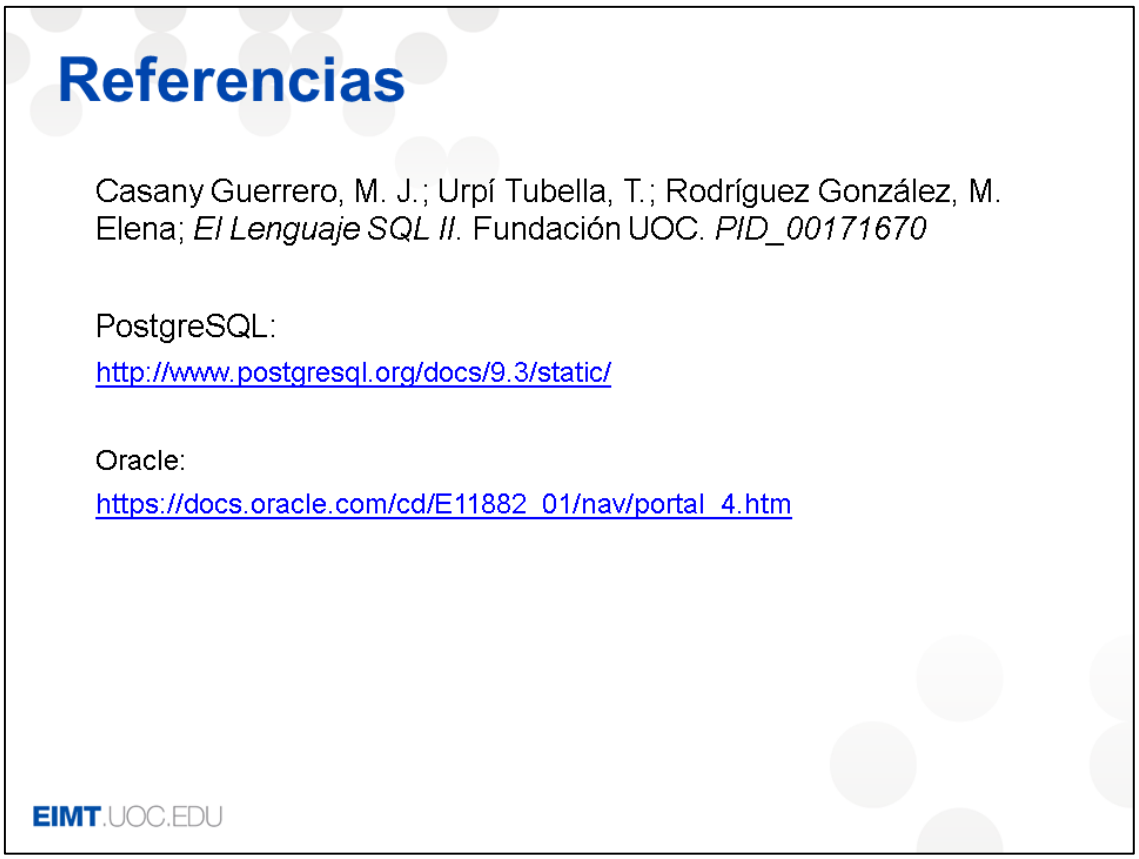
    En el cas de càrregues de taules de fets, és comú que, si l'error capturat està relacionat amb la manca de dades en les dimensions (per exemple, que no existeixi un producte en concret), s'insereixi la fila en la taula amb un identificador especial (generalment, amb valor -1 i la descripció DESCONEGUT), que significa que no és possible identificar en la dimensió el valor que s'ha intentat buscar. Aquesta tècnica és molt utilitzada en *data warehouse* per a assegurar que la manca de dades en origen no afecti els models dimensionals. D'aquesta forma, les càrregues de dades finalitzaran correctament i els usuaris podran veure les dades completes, això sí, amb una qualitat de les dades menor que si les dades estiguessin mapejades correctament.

Finalment, és important destacar que, generalment i com a part de la documentació lliurada en un projecte de *data warehouse*, se sol incloure un manual d'operador, que indica com es poden rellançar els processos de càrrega depenent d'on ha fallat i quines accions s'han de realitzar per a recuperar la consistència de les dades.



I fins aquí hem arribat amb aquest vídeo sobre la contextualització de procediments emmagatzemats. 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!