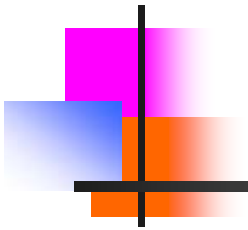


Esercitazione 5
DB2 – Cataloghi e Indici

Sistemi Informativi L-B

Home Page del corso:
http://www-db.deis.unibo.it/courses/SIL-B/

Versione elettronica: esercitazione5.pdf

Cataloghi di sistema

- Sono insieme di tables e views che descrivono la struttura logica e fisica degli oggetti di un DB, ovvero contengono metadati

- Gli schemi relativi sono:
  - SYSIBM: tables a uso interno di DB2
  - **SYSCAT**: viste definite sulle tables in SYSIBM, a uso degli utenti
  - **SYSSTAT**: statistiche sul DB

- Le viste in SYSCAT (in DB2 sono 142!) costituiscono il cosiddetto INFORMATION SCHEMA e alcune di queste sono standardizzate
  - Non sono direttamente modificabili
- Le cosiddette statistiche, sono invece (in parte) modificabili utilizzando le viste dello schema SYSSTAT (9 in totale)

SYSCAT.SCHEMATA

- Ogni tupla descrive uno schema
- Principali attributi:

Name	Description
SCHEMANAME	Nome dello schema
OWNER	Utente che ha creato lo schema[1]
CREATE_TIME	Timestamp di creazione

[1]Il campo **OWNER**, in questo catalogo così come nei successivi, è stato introdotto in DB2 a partire dalla versione 9, soppiantando il campo **DEFINER** ormai deprecato.

SYSCAT.TABLES

- Ogni tupla descrive una table o view (“oggetti”) del DB
- Tables e views dei cataloghi sono anch’esse incluse
- Principali attributi:

Name	Description
TABSCHEMA	Schema dell’oggetto
TABNAME	Nome dell’oggetto
OWNER	Proprietario dell’oggetto[1]
TYPE	Tipo dell’oggetto (‘T’ tabella, ‘V’ vista)
CREATE_TIME	Timestamp in cui l’oggetto è stato creato
COLCOUNT	Numero di colonne
CHILDREN	Numero di vincoli di foreign key che fanno riferimento a una key di questo oggetto
PARENTS	Numero di foreign key definite per questo oggetto
SELFREFS	Numero di foreign key autoreferenziali
CHECKCOUNT	Numero di check per questo oggetto
KEYCOLUMNS	Numero di colonne nella primary key

SYSCAT.VIEWS

- Ogni tupla fornisce dettagli aggiuntivi per una vista
- Principali attributi:

Name	Description
VIEWSCHEMA	Schema della vista
VIEWNAME	Nome della vista
OWNER	Utente che ha creato la vista
VIEWCHECK	Tipo di check applicato per le modifiche alla vista (‘C’ = cascaded check option, ‘L’ = local check option, ‘N’ = no check option)
READONLY	‘N’ = la vista può essere modificata dagli utenti con gli opportuni privilegi, ‘Y’ = la vista non può essere aggiornata a causa della sua definizione
TEXT	CLOB contenente la definizione SQL della vista

SYSSTAT.TABLES

- Ogni tupla fornisce le statistiche per una table o view
- Principali attributi:

Name	Description
TABSCHEMA	Schema dell'oggetto
TABNAME	Nome dell'oggetto
CARD	Cardinalità (numero di tuple)
NPAGES	Numero totale di pagine in cui esiste almeno una tupla
FPAGES	Numero totale di pagine (del file)
OVERFLOW	Numero di record in overflow

SYSCAT.COLUMNS

- Ogni tupla descrive una colonna di una vista o di una tabella
- Principali attributi:

Name	Description
TABSCHEMA	Schema dell'oggetto cui appartiene la colonna
TABNAME	Nome dell'oggetto cui appartiene la colonna
COLNAME	Nome della colonna
TYPENAME	Nome del tipo della colonna (VARCHAR, SMALLINT, ecc.)
LENGTH	Massima lunghezza dei valori della colonna
DEFAULT	Valore di default se definito, altrimenti NULL
NULLS	'Y' se la colonna ammette NULL, altrimenti 'N'
KEYSEQ	Posizione numerica della colonna all'interno della primary key

SYSSTAT.COLUMNS

- Ogni tupla descrive le statistiche di una colonna di una vista o di una tabella
- Principali attributi:

Name	Description
TABSCHEMA	Schema dell'oggetto cui appartiene la colonna
TABNAME	Nome dell'oggetto cui appartiene la colonna
COLNAME	Nome della colonna
COLCARD	Numero di valori distinti
LOW2KEY	Secondo valore minore
HIGH2KEY	Secondo valore maggiore
NUMNULLS	Numero di valori nulli
AVGCOLLEN	Numero medio di byte di un valore

SYSCAT.INDEXES

- Ogni tupla descrive un indice
- Principali attributi:

Name	Description
INDSCHEMA	Schema dell'indice
INDNAME	Nome dell'indice
TABSCHEMA	Schema della table su cui è costruito l'indice
TABNAME	Nome della table su cui è costruito l'indice
COLCOUNT	Numero di colonne su cui è costruito l'indice

SYSCAT.INDEXCOLUSE

- Ogni tupla descrive su quali colonne è costruito un indice
- Principali attributi:

Name	Description
INDSCHEMA	Schema dell'indice
INDNAME	Nome dell'indice
COLNAME	Nome della colonna
COLSEQ	Posizione della colonna (1,2,…)
COLORDER	Ordinamento dei valori ('A'=ascending; 'D'=descending)

SYSSTAT.INDEXES

- Ogni tupla descrive le statistiche di un indice
- Principali attributi:

Name	Description
INDSCHEMA	Schema dell'indice
INDNAME	Nome dell'indice
TABSCHEMA	Schema della table su cui è costruito l'indice
TABNAME	Nome della table su cui è costruito l'indice
COLNAME	Nome della colonna
COLCARD	Numero di valori distinti
NLEAF	Numero di pagine foglia dell'indice
NLEVELS	Numero di livelli dell'indice
FULLKEYCARD	Numero di valori distinti di chiave nell'indice
AVGCOLLEN	Numero medio di byte di un valore

Altri cataloghi

- Nel seguito viene riportata la definizione di altri cataloghi importanti, che però non utilizziamo in questa esercitazione

SYSCAT.TABCONST

- Ogni tupla descrive un constraint di tipo CHECK, UNIQUE, PRIMARY KEY o FOREIGN KEY
- Principali attributi:

Name	Description
CONSTNAME	Nome del constraint
TABSCHEMA	Schema della tabella cui si applica questo constraint
TABNAME	Nome della tabella cui si applica questo constraint
OWNER	Utente che ha creato il constraint
TYPE	Tipo di constraint (‘F’ = foreign key, ‘K’ = check, ‘P’ = primary key, ‘U’ = unique)
ENFORCED	‘Y’ se il constraint è attivo, altrimenti ‘N’

SYSCAT.KEYCOLUSE

- Ogni tupla descrive una colonna coinvolta nella definizione di una chiave primaria, chiave o foreign key
- Principali attributi:

Name	Description
CONSTNAME	Nome del constraint
TABSCHEMA	Schema di appartenenza
TABNAME	Nome della tabella
COLNAME	Nome della colonna coinvolta
COLSEQ	Posizione della colonna nella definizione della chiave

SYSCAT.REFERENCES

- Ogni tupla descrive una foreign key
- Utile da utilizzare assieme a SYSCAT.KEYCOLUSE!
- Principali attributi:

Name	Description
CONSTNAME	Nome del constraint
TABSCHEMA	Schema di appartenenza della tabella dipendente (quella che contiene la FK)
TABNAME	Nome della tabella dipendente
OWNER	Utente che ha creato il constraint
REFTABSCHEMA	Schema della tabella cui si fa riferimento
REFTABNAME	Nome della tabella cui si fa riferimento
REFKEYNAME	Nome del constraint nella tabella cui si fa riferimento (SYSCAT.KEYCOLUSE.CONSTNAME)
COLCOUNT	Numero di colonne coinvolte nella foreign key
DELETERULE	Politica per la cancellazione (‘A’ = no action, ‘C’ = cascade, ‘N’ = set null, ‘R’ = restrict)
UPDATERULE	Politica per l’aggiornamento (‘A’ = no action, ‘R’ = restrict)

SYSCAT.CHECKS

- Ogni tupla descrive un check constraint
- Principali attributi:

Name	Description
CONSTNAME	Nome del check
OWNER	Utente che ha creato il check
TABSCHEMA	Schema di appartenenza
TABNAME	Nome della tabella cui è applicato il check
CREATE_TIME	Timestamp di creazione
TEXT	Campo CLOB contenente la definizione del check

SYSCAT.COLCHECKS

- Ogni tupla descrive una colonna coinvolta in un check constraint
- Utile da utilizzare assieme a SYSCAT.CHECKS!
- Principali attributi:

Name	Description
CONSTNAME	Nome del check
TABSCHEMA	Schema di appartenenza
TABNAME	Nome della tabella cui è applicato il check
COLNAME	Nome della colonna coinvolta
USAGE	Uso della colonna, vale ‘R’ per i check, altri valori per altri usi non trattati nel corso

SYSCAT.TRIGGERS

- Ogni tupla descrive un trigger
- Principali attributi:

Name	Description
TRIGSCHEMA	Schema del trigger
TRIGNAME	Nome del trigger
TABSCHEMA	Schema della table su cuì è definito il trigger
TABNAME	Nome della table su cuì è definito il trigger
TEXT	CLOB contenente la definizione SQL del trigger

Esempi di query su cataloghi

```
SELECT TABSCHEMA, COUNT(*) FROM SYSCAT.TABLES GROUP BY TABSCHEMA
-- numero di table/views in ogni schema

SELECT TABNAME FROM SYSCAT.TABLES
WHERE TABSCHEMA = ‘SYSCAT;          -- i cataloghi in SYSCAT

SELECT INDNAME from SYSCAT.INDEXES
where INDSCHEMA = ‘SILB001’;       -- indici di uno schema

SELECT TABNAME,NCARD,NPAGES,FPAGES,OVERFLOW FROM SYSSTAT.TABLES
WHERE TABSCHEMA = ‘SILB001;        -- statistiche delle table di uno schema

SELECT TRIGNAME, CAST(TEXT AS VARCHAR(500)) AS SQLDEF
FROM SYSCAT.TRIGGERS
WHERE TABSCHEMA = ‘SILB001;        -- definizioni SQL dei trigger di uno schema
-- il CAST serve per dimensionare il campo SQLDEF (TEXT è un CLOB!)
```

Collezionare le statistiche

- Le informazioni statistiche sulle tabelle vengono utilizzate principalmente dall'ottimizzatore per eseguire in modo efficiente le interrogazioni
- Non vengono aggiornate automaticamente modificando una tabella, ma vanno aggiornate con un apposito comando **RUNSTATS**
- Esistono molte varianti, noi usiamo:

```
RUNSTATS ON TABLE MySchema.TableName WITH DISTRIBUTION
ON ALL COLUMNS AND DETAILED INDEXES ALL
```

- La forma **WITH DISTRIBUTION** forza DB2 a collezionare statistiche dettagliate sulle distribuzioni dei valori delle varie colonne (quante volte è ripetuto il valore più frequente, quante volte il secondo, ecc.)

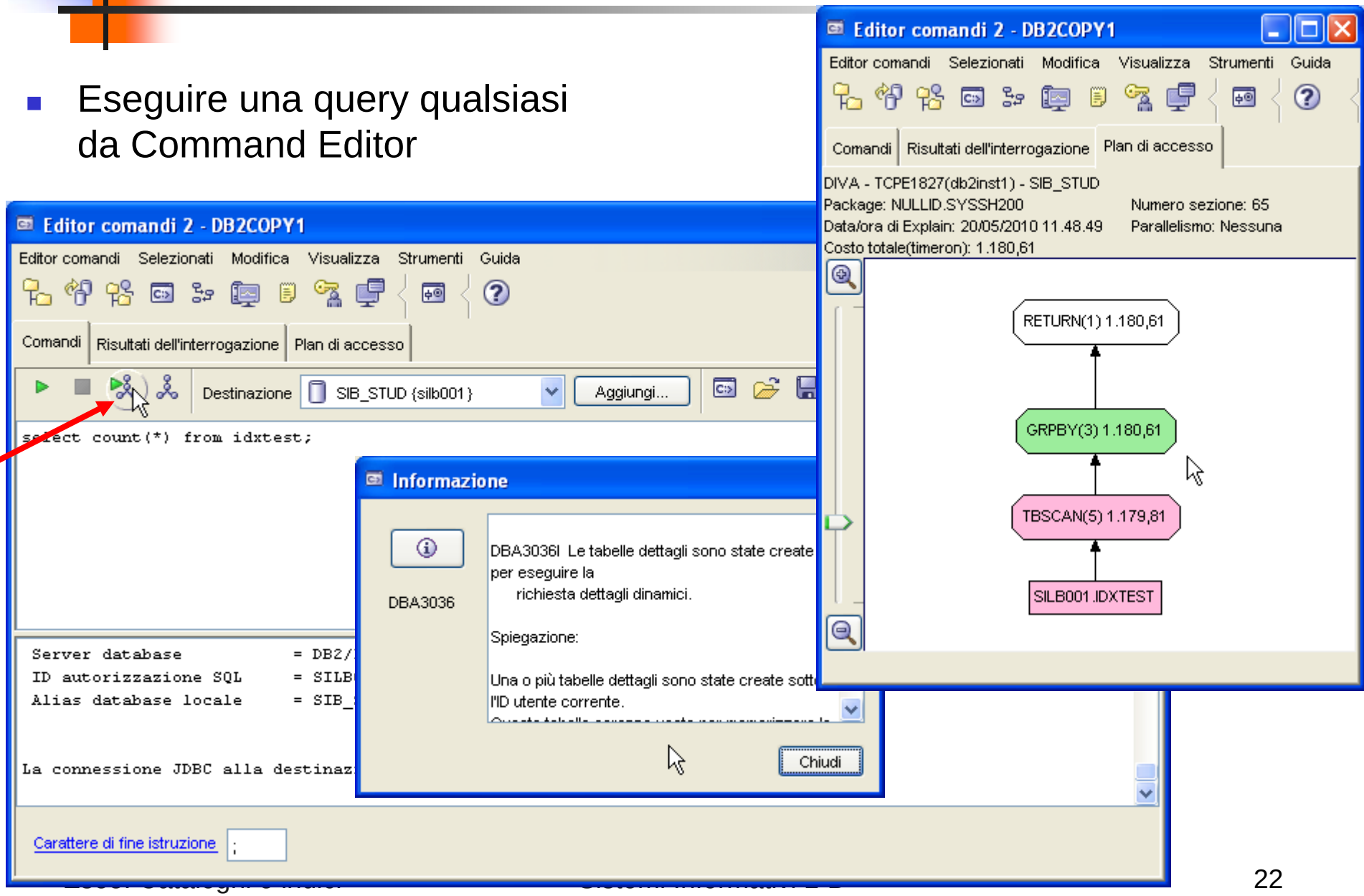
Explain tables

- Ogni volta che DB2 elabora uno statement SQL produce un cosiddetto "piano di accesso" (access plan), che fornisce dettagli su come la richiesta viene effettivamente eseguita
- Informazioni dettagliate su tali piani di accesso possono essere mantenute in un insieme di tabelle chiamate Explain tables
- Le Explain tables possono essere esaminate mediante
  - tool grafico del Command Editor (Visual Explain)
  - interrogazioni SQL
- Lo schema associato alle Explain table coincide con quello dello userid (es. SILB001): quindi ogni utente del sistema genera un insieme distinto di Explain table
- Prima di poter usare le Explain table occorre crearle
  - modalità grafica (Control Center o Command Editor)
  - a linea di comando (…solo lato server)

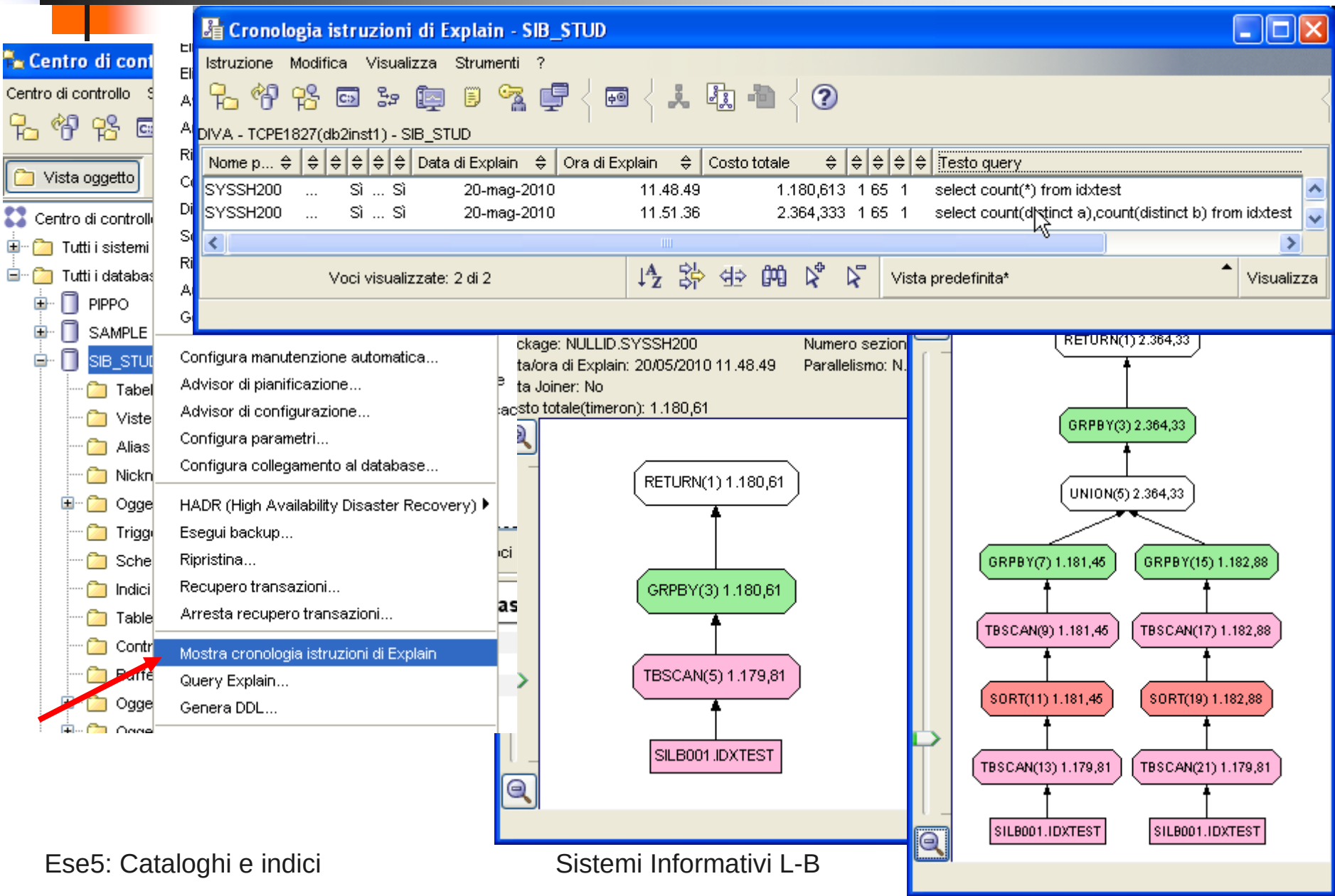
```
db2 -tf EXPLAIN.DDL (sqllib/misc/EXPLAIN.DDL)
```

Creazione Explain table: modalità grafica

- Eseguire una query qualsiasi da Command Editor



Explained statements history



Esercizio 1: prestazioni degli indici

- Proviamo a usare DB2 per cercare di capire come viene scelto l'indice migliore da usare nell'esecuzione di una query
- A tale scopo andremo a creare e popolare una tabella (**IDXTEST**) con solo 2 valori nel campo **A**, ma più valori distinti nel campo **B**
- Avendo due indici, uno su A e uno su B, e una query

  ```
  SELECT * FROM IDXTEST WHERE A = :a AND B = :b
  ```

  i due predicati avranno un diverso "fattore di selettività" (**FILTER_FACTOR**), e ciò influirà sulla scelta dell'indice scelto da DB2
- Eseguiamo le query dal Command Editor di DB2 per poter vedere l'access plan
- Cominciamo creando la tabella **IDXTEST**:

  ```
  CREATE TABLE IDXTEST(A INT, B INT, C CHAR(250))
  ```
- Il campo **C** serve solo a creare record "grandi" e quindi una relazione con molte pagine

Esercizio 1: Popolare la tabella di test (1)

- Per fare un test minimamente attendibile occorre una tabella con molti record (altrimenti DB2 non usa gli indici!)
- La prima possibilità è manuale:

```
insert into idxtest (a, b, c) values (1, 1, '1st record');
insert into idxtest (a, b, c) values (2, 2, '2nd record');
insert into idxtest (a, b) select a, 2*b+1 from idxtest;
insert into idxtest (a, b) select a, 2*b+2 from idxtest;
insert into idxtest (a, b) select a, 2*b+3 from idxtest;
insert into idxtest (a, b) select a, 2*b+4 from idxtest;
insert into idxtest (a, b) select a, 2*b+5 from idxtest;
insert into idxtest (a, b) select a, 2*b+6 from idxtest;
insert into idxtest (a, b) select a, 2*b+7 from idxtest;
...
```

Colonne con diverso n. di valori

```
SELECT A,B FROM IDXTEST
```

A	B
1	1
2	2
1	3
2	5
1	4
2	6
1	8
2	12
1	5
2	7
1	9
2	13
1	11
2	15
1	19
2	27
1	6

- Ora nella tabella ci sono 256 record:
  - nella colonna A sono presenti solo i valori 1 e 2;
  - nella colonna B i valori sono molto più vari

Esercizio 1: Popolare la tabella di test (2)

- Per generare molti più record possiamo usare un trigger!
- L'idea è definire un after trigger ricorsivo che si attivi a fronte di un inserimento in **`IDXTEST`**
- Bisogna però considerare che:
  - DB2 supporta al massimo 16 livelli di ricorsione
  - Non permette "ricorsione multipla" (ovvero non possiamo avere più di 1 azione di inserimento nel trigger)
- E quindi?

Esercizio 1: Popolare la tabella di test (3)

SOLUZIONE: usare uno Statement trigger che esegue un inserimento multiplo (come nell'approccio manuale)

```
CREATE TRIGGER AUTO_INSERT
AFTER INSERT ON IDXTEST
FOR EACH STATEMENT
WHEN (10000 > (SELECT COUNT(*) FROM IDXTEST))
INSERT INTO IDXTEST(A,B)
    SELECT A,MOD(3*B+1,50) FROM IDXTEST;
```

- Il trigger va creato dopo l'inserimento del 1° record e prima dell'inserimento del 2°
- Così si generano 16384 record! (16384 = 2+2+4+8+16+ … +8192)

Esercizio 1: Test (1)

- Consideriamo la query :

```
SELECT * FROM IDXTEST
WHERE A = :a AND B = :b
```

- Gli indici che possono influenzare l'esecuzione sono:
  1) indice solo su A (idxA)
  2) indice solo su B (idxB)
  3) indice su A e B (in quest'ordine) (idxAB)
  4) indice su B e A (in quest'ordine) (idxBA)

- Faremo quindi vari test: il primo senza nessun indice e i successivi con diverse configurazioni di indici
- Il numero di configurazioni è 16 (2^4), provarne almeno 6, ad es.:

  ∅, {idxA}, {idxB}, {idxA,idxB}, {idxA,idxB,idxAB}, {idxA,idxB,idxAB,idxBA}

Esercizio 1: Test (2)

- Per ogni test:
  1. Scegliere la configurazione di test (creando/eliminando indici)
  2. Aggiornare le statistiche
  3. Eseguire la query da Command Editor
  4. Vedere l'access plan e il costo stimato (nodo RETURN)
  5. Cliccare sull'eventuale nodo IXSCAN per vedere dettagli sull'indice usato
  6. Provare con diversi valori di B e vedere se cambia qualcosa
     (è utile la query `select b, count(*) from idxtest group by b`
     per capire quali sono i valori più o meno frequenti)

- Al termine, trarre le conclusioni determinando qual è l'indice più conveniente

Esercizio 2: riorganizzazione dei dati

- In questo esercizio vediamo alcuni strumenti di base per capire meglio aspetti relativi alla rappresentazione fisica dei dati
- Un comando estremamente utile allo scopo è **REORGCHK**

```
REORGCHK ON TABLE MySchema.TableName
```

  che fornisce alcuni indicatori utili per capire se una table deve essere riorganizzata fisicamente (esegue anche **RUNSTATS**)
- Se necessario (può richiedere tempo!), la table può essere riorganizzata con:

```
REORG TABLE MySchema.TableName
```

  che riorganizza anche tutti gli indici
- Per riorganizzare solo gli indici:

```
REORG INDEXES ALL FOR TABLE MySchema.TableName
REORG INDEX AnIndex FOR TABLE MySchema.TableName
```

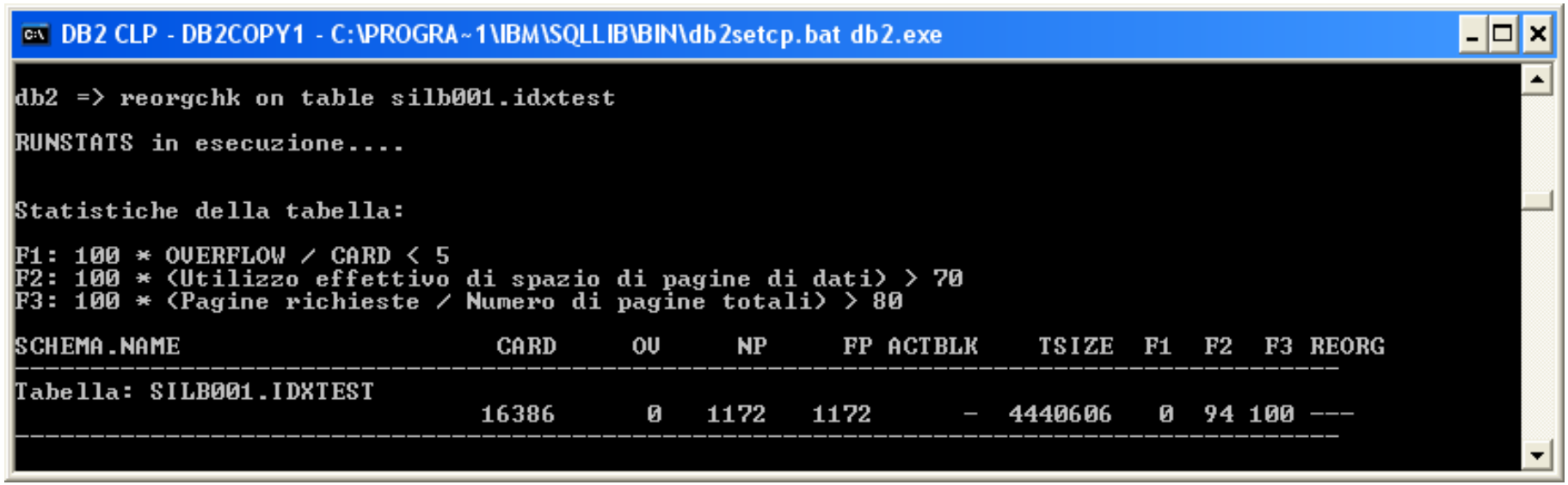
Info da REORGCHK per una table

- Vengono forniti tre indicatori, con relativi valori di soglia che, se superati, settano un relativo flag che suggerisce la riorganizzazione

F1: % di record in overflow (< 5%)

F2: % di spazio utilizzato nelle pagine allocate (> 70%)

F3: NPAGES/FPAGES (> 80%)



Esercizio 2: frammentare i dati

- Per generare una situazione in cui la riorganizzazione può rendersi necessaria riapplichiamo il trigger ricorsivo
  - Molto probabilmente questo genera frammentazione, che possiamo verificare con **REORGCHK**
- Quindi inseriamo ancora manualmente un record, ma modificando la soglia del trigger da 10000 a 30000 (cancellare il trigger e ricrearlo)
- Proviamo anche a cancellare tutte le tuple con b>14
- Ripetiamo quindi le prove con gli indici e verifichiamo se e cosa è cambiato…

- Al termine eliminare **IDXTEST,** gli indici e le explain tables

  NB: per rimuovere più oggetti in un colpo solo si può usare il Control Center, selezionandoli tutti
- Per cancellare solo i dati delle explain tables:

```
delete from explain_instance;
```

  - Eliminando un record dalla tabella explain_instance verranno eliminati tutti i record nelle altre explain tables

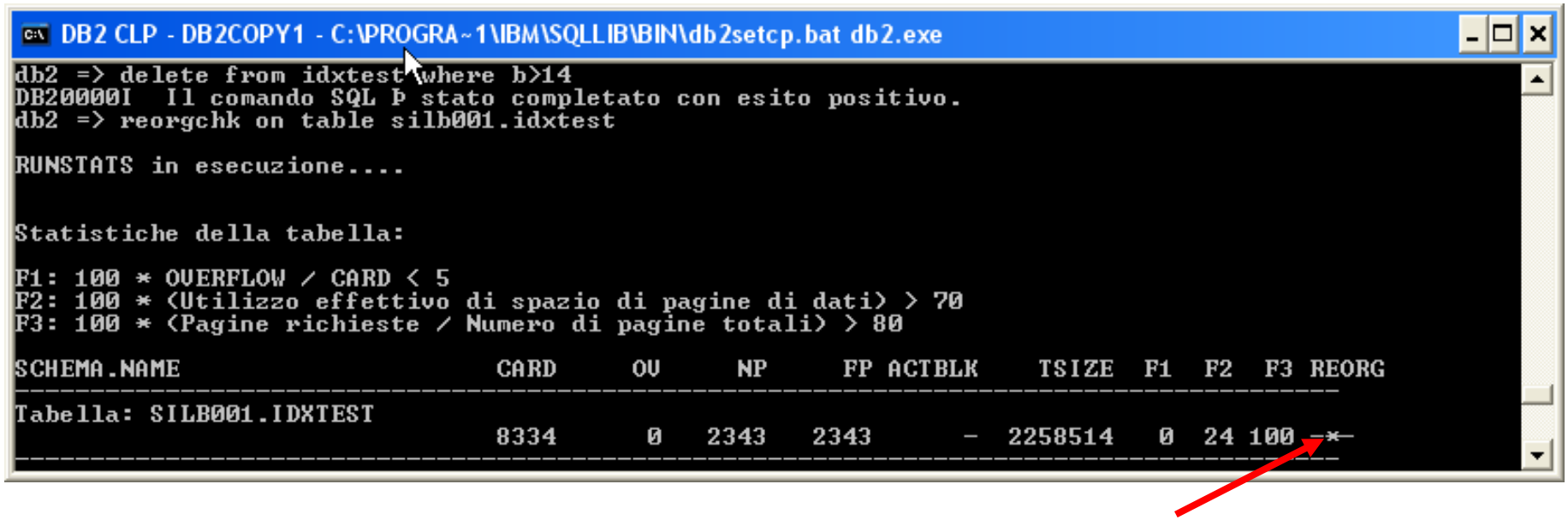
Info aggiornate da REORGCHK: prima…

- Cancellando tuple, l'indicatore F2 è sceso al valore 24%, il che suggerisce di riorganizzare la table…

F1: % di record in overflow (< 5%)

F2: % di spazio utilizzato nelle pagine allocate (> 70%)

F3: NPAGES/FPAGES (> 80%)



…e dopo la riorganizzazione