IBM i

# Più unico che raro

## *generare un ID univoco con RPG e SQL*

Marco Riva

MKT

www.markonetools.it

IBMCHAMPION 2021 Power Systems area

Ultimo aggiornamento: 31/10/2021

# DB2: identity e row id

- **`identity`**: il db può generare automaticamente dei **valori numerici sequenziali**. Il db non verifica che il valore sia univoco a meno che non esista un indice univoco che specifichi solo la colonna identity. Il tipo dati deve essere un intero o un numero definito senza decimali.

```
My_ID bigint generated by default as identity
      (start with 1 increment by 1 no cycle cache 50 order)
```

- **`rowid`**: è un valore (**varchar 40 byte**) che identifica in maniera univoca un record di una tabella. Il db si preoccupa di mantenere il valore univoco anche a seguito della riorganizzazione della tabella. Il valore **non è sequenziale**. E' indipendente dal CCSID.

```
My_ID rowid generated by default
```

# DB2: generazione automatica

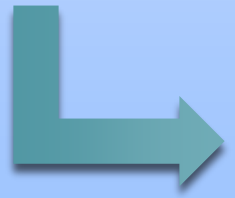
- per i campi di tipo rowid o identity è possibile specificare la clausola **`generated`**. Il DB **genera automaticamente** dei valori per popolare il campo sulle operazioni di inserimento
  - `always`: il db genera sempre il valore. Non si può specificare un valore nell'istruzione di inserimento a meno di usare la clausola `overriding`.
  - `by default`: il db genera il valore a meno che non venga esplicitamente fornito un valore nell'istruzione di inserimento

cfr. anche Generated Always Columns – Approfondimenti (IT) in https://blog.faq400.com/it/database-db2-for-i/howto-generated-always-colums/

# DB2: Esempio 1

```
create or replace table MYTAB1
   (
    My_ID bigint generated by default as identity
(start with 1 increment by 1 no cycle order),
    My_RID rowid generated by default,
    My_Desc char(10));

insert into MYTAB1 values(default, default, 'REC1');
insert into MYTAB1 (My_Desc) values('REC2');
insert into MYTAB1 (My_ID, My_Desc) values(3, 'REC3');
insert into MYTAB1 (My_Desc) values('REC4');
```



| MY_ID | MY_RID | MY_DESC |
|---|---|---|
| 1 | A725CB7A5D852001F2F110C4F9C6C5E640400000000000000001 | REC1 |
| 2 | A725CB94E3631001F2F110C4F9C6C5E640400000000000000001 | REC2 |
| 3 | A725CB9FB4C9F001F2F110C4F9C6C5E640400000000000000001 | REC3 |
| 3 | A725CBA4C4B04001F2F110C4F9C6C5E640400000000000000001 | REC4 |

# DB2: Esempio 2

```
create or replace table MYTAB2
  (
   My_ID bigint generated always as identity (start with 1 increment by 1 no
cycle order),
   My_RID rowid generated by default,
   My_Desc char(10));

insert into MYTAB2 values(default, default, 'REC1');
insert into MYTAB2 (My_Desc) values('REC2');
insert into MYTAB2 (My_ID, My_Desc) values(5, 'REC3');
insert into MYTAB2 (My_ID, My_Desc) overriding system value
  values(5, 'REC3');
insert into MYTAB2 (My_Desc)  values('REC4');
insert into MYTAB2 (My_Desc)  values('REC5');
insert into MYTAB2 (My_Desc)  values('REC6');
```

errore SQL0798

| MY_ID | MY_RID | MY_DESC |
|---|---|---|
| 1 | A72A77C7C4AC2001F2F110C4F9C6C5E640400000000000000001 | REC1 |
| 2 | A72A77C853342001F2F110C4F9C6C5E640400000000000000001 | REC2 |
| 5 | A72A77D6CD52B001F2F110C4F9C6C5E640400000000000000001 | REC3 |
| 3 | A72A77DD72BBE001F2F110C4F9C6C5E640400000000000000001 | REC4 |
| 4 | A72A78070DA07001F2F110C4F9C6C5E640400000000000000001 | REC5 |
| 5 | A72A7813D0059001F2F110C4F9C6C5E640400000000000000001 | REC6 |

# DB2: Esempio 3

```
create or replace table MYTAB3
   (
    My_ID bigint generated by default as identity (start with
1 increment by 1 no cycle order) constraint MYTAB3U unique,
    My_RID rowid generated by default,
    My_Desc char(10));

insert into MYTAB3 values(default, default, 'REC1');
insert into MYTAB3 (My_Desc) values('REC2');
insert into MYTAB3 (My_ID, My_Desc) values(3, 'REC3');
insert into MYTAB3 (My_Desc)  values('REC4');
insert into MYTAB3 (My_Desc)  values('REC4');
```

errore SQL0803 chiave duplicata perché My_ID calcolato è 3

la medesima istruzione ripetuta va a buon fine perché My_ID calcolato ora è 4

| MY_ID | MY_RID | MY_DESC |
|---|---|---|
| 1 | A725CE5297383001F2F110C4F9C6C5E640400000000000000001 | REC1 |
| 2 | A725CE5CD912D001F2F110C4F9C6C5E640400000000000000001 | REC2 |
| 3 | A725CE5E0A522001F2F110C4F9C6C5E640400000000000000001 | REC3 |
| 4 | A725CEA8B9791001F2F110C4F9C6C5E640400000000000000001 | REC4 |

# Generare un GUID/UUID

- Il **GUID** (**G**lobal **U**nique **ID**entifier) o un **UUID** (**U**niversally **U**nique **ID**entifier) è un valore binario di 16 bytes che si suppone essere univoco in senso "assoluto" (quindi anche su tabelle diverse e su database diversi)
- Per generare un **GUID/UUID** è possibile utilizzare l'API **GENUUID** che restituisce un valore simile a

  UUID = ÷ÞØÎÊª3 Ðþê

  che può essere convertito in caratteri "umani" (32 car.) con la funzione SQL `hex`

  GUID = E1AE8006761D1A729AF30004AC1C8E52

# API _GENUUID/1

- Prototipo

```
dcl-pr GenUUID extproc('_GENUUID');
  *n pointer value;
end-pr;
```

- Struttura dati per parametro

```
dcl-ds dsUUID;
  // Input: byte utilizzati (deve essere almeno 32)
  BytesProv uns(10:0) inz(%size(dsUUID));
  // Output: byte restituiti
  BytesAvail uns(10:0);
  // Input: versione: 0 versione 1 DCE
  Version uns(3:0) inz(0);
  // riservato da inizializzare a zero
  *n char(7) inz(*allx'00');
  // Output: UUID restituito (formato binario)
  UUID char(16);
end-ds;
```

# API _GENUUID/2

- chiamata API (16 car.)

```
UUID = *allx'00';
GenUUID(%addr(dsUUID));
```

- conversione UUID binario in alfanumerico (32 car.)

```
dcl-s GUID like(UUID:+16);

exec sql
  set :GUID = hex(:UUID);
```

- formattazione UUID secondo formato standard (36 car.)

```
dcl-s GUID like(UUID:+20);

GUID = %subst(GUID:1:8) + '-' +
              %subst(GUID:9:4) + '-' +
              %subst(GUID:13:4) + '-' +
              %subst(GUID:17:4) + '-' +
              %subst(GUID:21:12);
```

# Funzione RPG %timestamp

da 7.3, 7.4

- La funzione RPG **%timestamp** consente di restituire un timestamp unico specificando il parametro *unique
- **ATTENZIONE**: questo timestamp NON va considerato come una data/ora con una precisione della parte frazionale di 12 cifre. La parte frazionale dei secondi è composta dalle prime 6 cifre che sono effettivamente i microsecondi e le ultime 6 cifre vengono generate solo per rendere univoco il timestamp

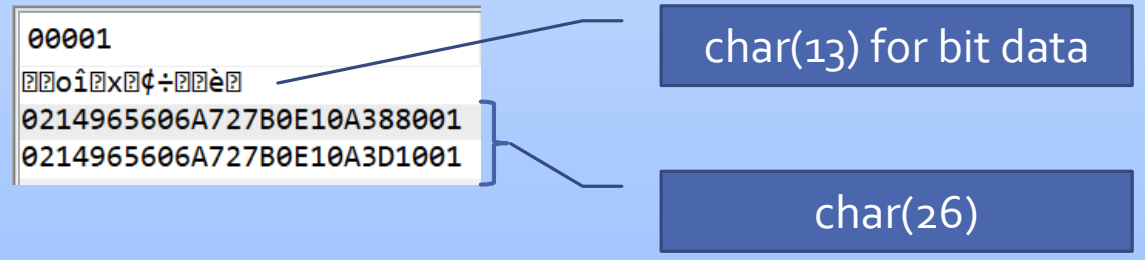
```
dcl-s TimeUnique timestamp(12);

TimeUnique = %timestamp(*unique);
```

# Funzione SQL generate_unique()

- La funzione SQL **`generate_unique()`** restituisce una stringa di tipo `char(13) for bit data` che è univoca rispetto ad ogni altra chiamata della medesima funzione
- La stringa restituita è ottenuta dal timestamp e dal numero seriale
- La funzione restituisce risultati diversi per ogni esecuzione anche all'interno della stessa istruzione

```
values generate_unique(), hex(generate_unique()), hex(generate_unique());
```



# Riferimenti

- E-mail aziendale: mriva@sirio-is.it
- Blog: www.markonetools.it
- E-mail blog: info@markonetools.it
- Linkedin: www.linkedin.com/in/marcoriva-mk1
- Twitter: @MarcoRiva73
- Facebook: https://www.facebook.com/markonetools/
- YouTube: https://www.youtube.com/channel/UCb47YJQJCzU-5x4nnGzDu-w