


DM CERN IT Department

Database Design Tips & Tricks



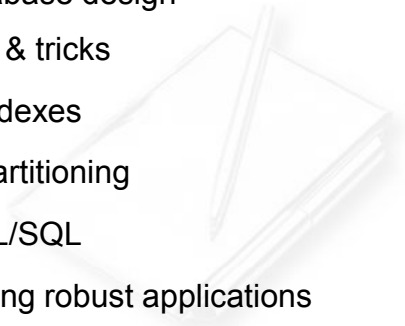
Database Developers' Workshop
CERN, July 8th, 2008
Eva Dafonte Pérez, CERN IT-DM
DB Design based on slides by Dawid Wójcik

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM CERN IT Department

Outline

- Database design
- Tips & tricks
 - Indexes
 - Partitioning
 - PL/SQL
- Writing robust applications
- Q&A



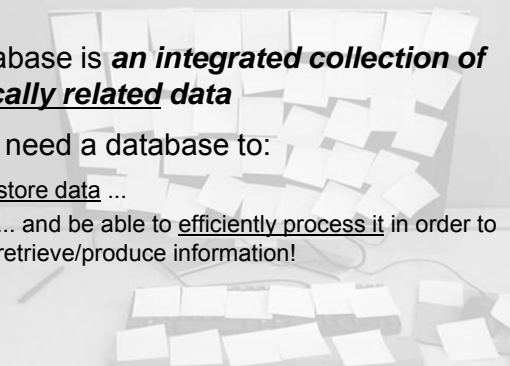
CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM CERN IT Department

Database design

"It's a Database, Not a Data Dump"

- Database is ***an integrated collection of logically related data***
- You need a database to:
 - store data ...
 - ... and be able to efficiently process it in order to retrieve/produce information!



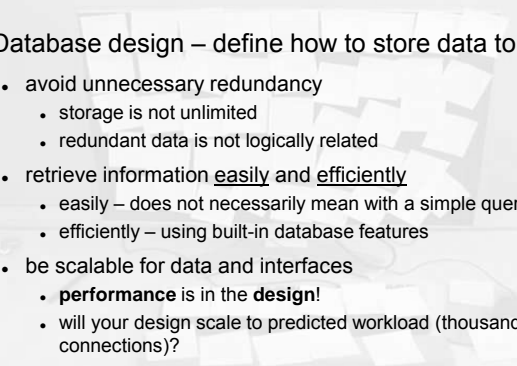
CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM CERN IT Department

Database design – goals

"It's a Database, Not a Data Dump"

- Database design – define how to store data to:
 - avoid unnecessary redundancy
 - storage is not unlimited
 - redundant data is not logically related
 - retrieve information **easily and efficiently**
 - easily – does not necessarily mean with a simple query
 - efficiently – using built-in database features
 - be scalable for data and interfaces
 - **performance** is in the **design**!
 - will your design scale to predicted workload (thousands of connections)?



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Conceptual design

CERN IT Department

- Process of constructing a model of the information used in an enterprise
- Is a conceptual representation of the data structures
- Is independent of all physical considerations

Database requirements

Conceptual model

Database Developers' Workshop, CERN, July 2008 - 5

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Conceptual design – practice

CERN IT Department

- The **Entity-Relationship model (ER)** is most common conceptual model for database design:
 - describes the data in a system and how data is related
 - describes data as **entities**, **attributes**, and **relationships**
 - can be easily translated into many database implementations

Entity

Relationship

Attribute

Department

id
* name
o location

Employee

id
* name
o age

composed of

assigned to

Database Developers' Workshop, CERN, July 2008 - 5

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Modeling relationships - example

CERN IT Department

- Many – to – many (M:N)
 - a student can be registered on any number of courses (including zero)
 - a course can be taken by any number of students (including zero)

Student

student_id
* last_name
* first_name
o date_of_birth

Course_enrollment

student_id
course_id
* enrollment_date

Course

course_id
* course_name
* start_date
* end_date

cannot be represented by the relational model

Database Developers' Workshop, CERN, July 2008 - 7

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Logical design

CERN IT Department

- Translate the conceptual representation into the logical structure
- Logical model – normalized form

Conceptual Model (ERD)

Relational model

Database Developers' Workshop, CERN, July 2008 - 8

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Third Normal Form (3NF) CERN IT Department

- 2NF
- No transitive dependency for non-key attributes
 - any non-key attribute cannot be dependent on another non-key attribute

Class(CID, CNAME, CLEVEL, ROOM, CAPACITY)

```

graph TD
    CID((CID)) --> CNAME((CNAME))
    CID --> CLEVEL((CLEVEL))
    CID --> ROOM((ROOM))
    CID --> CAPACITY((CAPACITY))
    CLEVEL --> ROOM
    CLEVEL --> CAPACITY
  
```

Violation of the 3NF!

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Normalization to 3NF CERN IT Department

- For each non-key attribute that is transitive dependent on a non-key attribute, create a table

Class(CID, CNAME, CLEVEL, ROOM, CAPACITY)

```

graph LR
    CID((CID)) --> CNAME((CNAME))
    CID --> CLEVEL((CLEVEL))
    CID --> ROOMID((ROOMID))
    CID --> CAPACITY((CAPACITY))
    CLEVEL --> CAPACITY
  
```

Class(CID, CNAME, CLEVEL, ROOMID)
Room(ROOMID, CAPACITY)

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Integrity constraints - PK CERN IT Department

- Primary keys (PK)
 - enforce entity integrity
 - attribute or set of attributes that uniquely identifies each record in the table
 - every entity in the data model must have a primary key that:
 - is a non-null value
 - is unique
 - it does not change or become null during the table life time (time invariant)

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Integrity constraints - FK CERN IT Department

- Foreign keys (FK)
 - maintains consistency between two tables with a relation
 - must have a value that matches a primary key in the other table or be null
 - an attribute in a table that serves as primary key of another table
 - use foreign keys!
 - foreign keys with indexes on them improve performance of selects, but also inserts, updates and deletes

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Schema design – best practices CERN IT Department

- Use generic structures
 - whenever there's a need of higher flexibility
 - do not use it to model “everything”

```

graph TD
    Parameter --> Object
    Parameter --> Parameter_type
    Object --> Object_type
  
```

Parameter
parameter_id
* parameter_type_id
* object_id
o value

Object
object_id
* object_type_id
o object_name

Parameter_type
parameter_type_id
o parameter_name

Object_type
object_type_id
o type_name

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Schema design – best practices CERN IT Department

- Column types and sizing columns
 - VARCHAR2(4000) is not the universal column type
 - high memory usage on the client
 - it makes data dump, not database
 - you cannot index a key longer than ≈6400 bytes
 - use proper data types
 - “nullable” columns at the end of the table

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Schema design – best practices CERN IT Department

- Estimate future workload
 - read intensive?
 - write intensive?
 - transaction intensive?
 - mixture? – estimate the amount of each type
- Design indexes knowing the workload
 - what will users query for?
 - minimize number of indexes using proper column order in the indexes
 - create views, stored procedures (PL/SQL) to retrieve the data in the most efficient way – easier to tune in a running system
 - what is the update/insert/delete pattern?
 - create indexes on foreign keys

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM CERN IT Department

Tips & tricks

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM

CERN IT

Department

Indexes – tips & tricks

- Selective indexes
 - suppose we have huge table with jobs, most of them already processed (processed_flag = 'Y'), we want only to index non-processed jobs
 - use bitmap index – very bad idea on frequently updated tables
 - create selective view
 - create function based index:

```
create index my_indx on tbl_t1 (
  case when processed_flag = 'N' then 'N'
       else NULL
end);
```
 - save index space, improve queries performance

CERN IT Department

CH-1211 Genève 23

Switzerland

www.cern.ch/it

DM

CERN IT

Department

Indexes – tips & tricks

- Reversed indexes
 - suppose we have many concurrent programs that insert into the same table
 - the table has a primary key column populated by an increasing sequence
 - no range scans on that column
 - data is deleted from time to time according to some rules which leave some old data undeleted in the table
 - create reversed index:

```
alter index index_name rebuild reverse;
```
 - reversed index decreases contention on the index, especially in RAC environment – improves insert/update performance
 - you can no longer make range scans using reversed index

CERN IT Department

CH-1211 Genève 23

Switzerland

www.cern.ch/it

DM

CERN IT

Department

Partitioning – tips & tricks

- Investigate partitioning your application
 - partitioning by time, subdetector, subsystem, etc
 - benefits:
 - increased availability – in case of loosing one tablespace/partition
 - easier administration – moving smaller objects if necessary, easier deletion of history, easier online operations on data (ALTER TABLE ... EXCHANGE PARTITION)
 - increased performance – use of local and global indexes, less contention in RAC environment

CERN IT Department

CH-1211 Genève 23

Switzerland

www.cern.ch/it

DM

CERN IT

Department

PL/SQL – tips & tricks

- Query parse types
 - Hard parse
 - optimizing execution plan of a query
 - high CPU consumption
 - Soft parse
 - reusing previous execution plan
 - low CPU consumption, faster execution
- Reduce the number of hard parses
 - put top executed queries in PL/SQL packages/procedures/functions
 - put most common queries in views
 - it also makes easier to tune bad queries in case of problems

CERN IT Department

CH-1211 Genève 23

Switzerland

www.cern.ch/it

DM

PL/SQL – tips & tricks

CERN IT
Department

- Reduce the number of hard parses
 - use bind variables
 - instead of:


```
select ... from users where user_id=12345
```
 - use:


```
select ... from users where user_id=:uid
```
 - using bind variables protects from sql injection:
 - before:


```
sql = "select count(*) from users where
username='"+user+"' and password='"+pass+"'"
user="hacker"    pass="aaa' or 1=1"
```
 - after:


```
sql = "select count(*) from users where username=:user
and password=:pass"
```

CERN IT Department

CH-1211 Genève 23

Switzerland

www.cern.ch/it

DM

PL/SQL – tips & tricks

CERN IT
Department

- Beware of bind variables peeking
 - optimizer peeks at bind variable values before hard parsing for the first time
 - suppose we have huge table with jobs, most of them already processed (processed_flag = 'Y')
 - using bind variable on processed_flag may change query behavior, depending on which query is processed first after DB startup (with bind variable set to 'Y' or 'N')
 - on a low cardinality column which distribution can significantly vary in time – do not use bind variable only if doing so will result in just a few different queries, otherwise use bind variables

CERN IT Department

CH-1211 Genève 23

Switzerland

www.cern.ch/it

DM

PL/SQL – tips & tricks

CERN IT
Department

- Reduce the number of hard parses
 - Prepare once, execute many
 - use prepared statements
 - dynamic SQL executed thousands of times – consider dbms_sql package instead of execute immediate
 - use bulk inserts whenever possible
- Use fully qualified names
 - instead of:


```
select ... from table1 ...
```
 - use:


```
select ... from schema_name.table1 ...
```
 - known bugs – execution in a wrong schema

CERN IT Department

CH-1211 Genève 23

Switzerland

www.cern.ch/it

DM

Writing robust applications

CERN IT
Department

- Use different level of account privileges
 - application owner (full DDL and DML)
 - writer account (grant read/write rights to specific objects)
 - reader account (grant read rights)
 - directly grant object rights or use roles
 - caution – roles are switched off in PL/SQL code, one must set them explicitly
 - passwords in code get exposed very easily
 - exposing reader password may result in DoS attacks
 - exposing other accounts' passwords may result in data loss

CERN IT Department

CH-1211 Genève 23

Switzerland

www.cern.ch/it

DM

Writing robust applications

CERN IT Department

- Use connection pooling
 - connect once and keep a specific number of connections to be used by several client threads (pconnect in OCI)
 - test if the connection is still open before using it, otherwise try reconnecting
 - log connection errors, it may help DBAs to resolve any potential connection issues
- Connection management
 - handle server or network failures
 - in case of server failure (or rolling patch), connection should be automatically moved to an available server
 - other errors might happen – review documentation/wiki

CERN IT Department

CH-1211 Genève 23

Switzerland

www.cern.ch/it

DM

Writing robust applications

CERN IT Department

- Error logging and retrying
 - trap errors
 - check transactions for errors, try to repeat failed transactions, log any errors (including SQL that failed and application status – it might help to resolve the issue)
 - some parts of a transaction (bulk insert) may fail – consider using error logging clause:
 - CREATE TABLE raises (emp_id NUMBER, sal NUMBER CONSTRAINT check_sal CHECK(sal > 8000));
 - EXECUTE DBMS_ERRLOG.CREATE_ERROR_LOG('raises', 'errlog');
 - INSERT INTO raises SELECT employee_id, salary*1.1 FROM employees WHERE commission_pct > .2 LOG ERRORS INTO errlog ('my_bad') REJECT LIMIT 10;
 - SELECT ORA_ERR_MSG\$, ORA_ERR_TAG\$, emp_id, sal FROM errlog;

ORA_ERR_MSG\$	ORA_ERR_TAG\$	EMP_ID	SAL
ORA-02290: check constraint my_bad (HR.SYS_C004266) violated		161	7700

CERN IT Department

CH-1211 Genève 23

Switzerland

www.cern.ch/it

DM

Writing robust applications

CERN IT Department

- Design, test, design, test ...
- Try to prepare a testbed system – workload generators, etc
- Do NOT test changes on a live production system
- IT-DM provides test and integration system (preproduction) with the same Oracle setup as on production clusters
 - contact PhyDB.Support to obtain accounts and ask for imports/exports

CERN IT Department

CH-1211 Genève 23

Switzerland

www.cern.ch/it

DM

Writing robust applications

CERN IT Department

Q & A

<https://twiki.cern.ch/twiki/bin/view/PSSGroup/GeneralAdvices>

<https://twiki.cern.ch/twiki/bin/view/PSSGroup/ConnectionManagement>

CERN IT Department

CH-1211 Genève 23

Switzerland

www.cern.ch/it