



WhereScape RED
Builds data warehouses, fast.

The WhereScape Pragmatic Data Warehousing™ Methodology:

An Overview

Wayne Richmond, Chief Technology Officer
WhereScape Software

ABSTRACT

This paper describes in summary fashion the models, methods and processes that make up the essence of WhereScape's Pragmatic Data Warehousing™ methodology as practiced by WhereScape designers and as implemented in the WhereScape RED data warehouse life cycle management product.

OVERVIEW

The WhereScape Pragmatic Data Warehousing™ Methodology, or PDW, has been in development by several dozen practitioners associated with WhereScape for more than a decade.

Born largely out of frustration with the boom-bust cycles through which data warehousing seems to pass – and the incessant name-changing and euphemism-making that accompanies this boom-bust cycle – PDW is focused on several (as we see it) self-evident objectives for a data warehousing or data marting project:

- > Build the warehouse or mart in a fashion that involves business analysts in the process, builds their confidence in IT, and increases their commitment to the finished product: the warehouse or mart under development.
- > Treat data sources as what they in fact are – constraints on the outer bounds of the functionality of any data warehouse or mart – rather than as the central design problem, which they assuredly are not.
- > Treat the data warehouse or mart as a process, not a project – focus on iterative releases and rollouts that follow one another in quick succession, keeping the warehouse or mart evergreen, instead of treating the warehouse as a one-time, big-bang project that produces a finished monolith that endures in that form for ever more.
- > Build warehouses and marts that present the dominant sets of required information for decision-making in the most appropriate form for the end-user analysts and their chosen tool sets (in that order), saving edge-cases, arcane issues and special needs for subsequent iterations.
- > Focus energy and technology on the three areas of the data warehouse life cycle that research and practice clearly show are the leading sources of failed warehouses and marts: poor schema design that distort, disguise or neglect critical information; overly lengthy deployment cycles that cannot keep pace with business change; and near-total lack of enhancement to warehouses and marts due to their overly brittle implementation.

In a nutshell: **do it fast, do it properly, and then do it again. And again. And again.**

At the outset, it's important to note a few critical things about PDW, in order for the proper context to be established:

- > We did not develop the PDW to sell it, per se. We developed it to improve our own skills and cycle times as practitioners.

- > WhereScape RED, our data warehouse life cycle management software product, has its genesis in PDW. RED began as set of marginally-connected private tools several of us used to perform what we saw as basic activities associated with good data warehouse design, deployment and enhancement, and evolved into an integrated data warehouse lifecycle management product as our methods and practices began to cohere and form a coherent whole.
- > Any methodology will produce better results, consistently, than no methodology or merely intuitive design and development behavior.

This last point we cannot emphasize enough. Methodology, per se, when embedded in software products, creates suspicion in the minds of designers and developers, due in part no doubt to the draconian ways in which so-called "upper CASE" tools popular in the 1980s and early 1990s demanded full and unthinking compliance with their ways of doing things. We are not interested in enforcing our practice; we are interested in seeing a common practice emerge across data warehousing projects within a given organization, because that common practice will produce predictable results.

Additionally, data warehouse design is not as mature as the transactional application design CASE tools tried to automate– not by a long stretch. There are still many valid instances in which 'normal' practice is either insufficient or not applicable to the problem at hand, and – as pragmatists – we believe that the situation at hand should always determine the methods used. In some cases, we have to tackle novel design or deployment problems for which no existing methods will suffice.

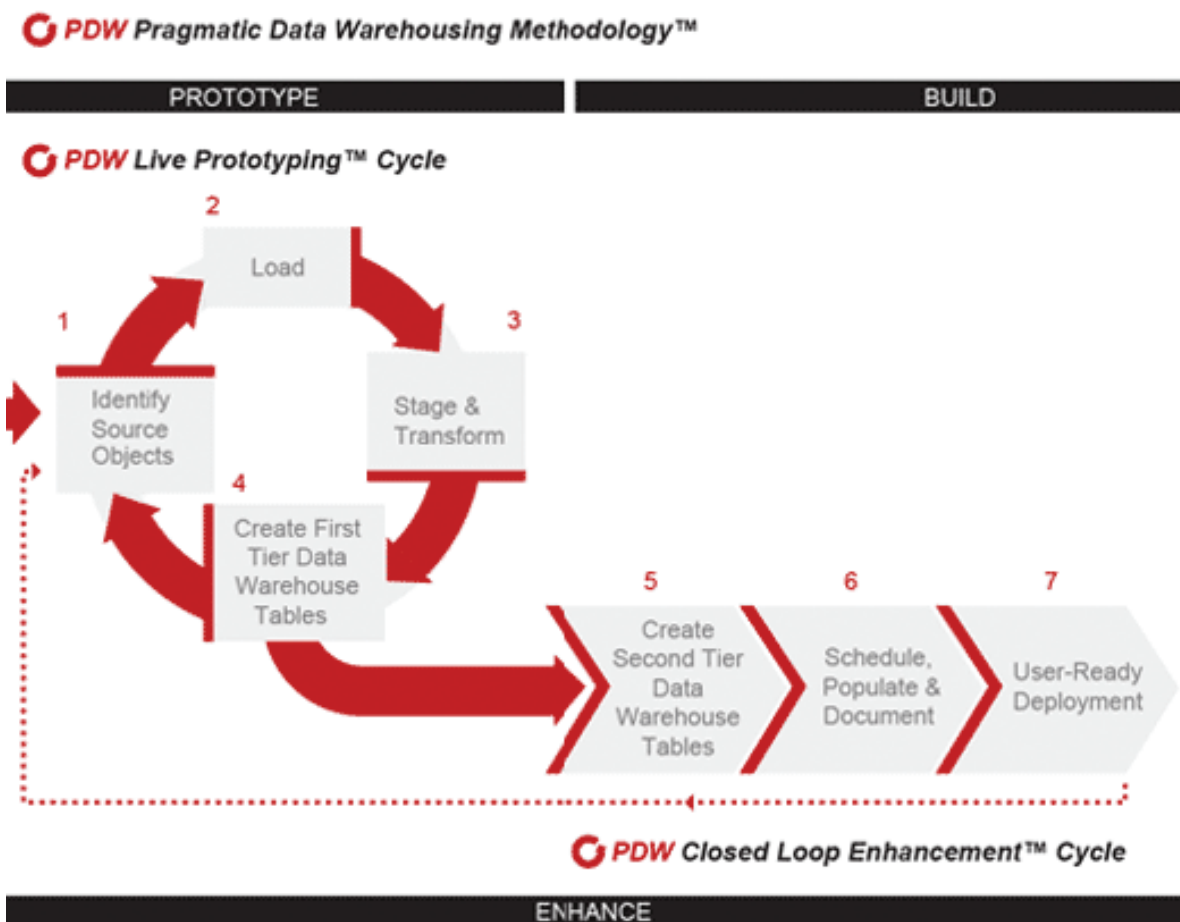
Finally, we believe that gentle suggestion is better than draconian enforcement. The techniques and methods that make up PDW can be extracted from PDW and transplanted into other methodologies – including the classic standard development life cycle (SDLC) waterfall-style project – with little or no rework.¹ We practice PDW rigorously, and we believe our customers' uniformly high levels of satisfaction and success are the best metrics of that consistent practice and the intrinsic value of PDW. But, when we came to embody PDW in WhereScape

RED, we made the conscious decision to suggest PDW techniques, but not to enforce them. WhereScape RED has been, and can be, used quite successfully in data warehousing projects that employ a project methodology or a design methodology other than PDW.²

As a final, introductory observation, we need to acknowledge that PDW it trades heavily in ideas first elaborated by Bill Inmon and Ralph Kimball. We freely and fully acknowledge our debts to those two pioneers.

THE PRAGMATIC DATA WAREHOUSING™ METHODOLOGY IN A NUTSHELL

The PDW consists of three interlinked cycles, or sets of activities, as described in the diagram below, and the text that follows.

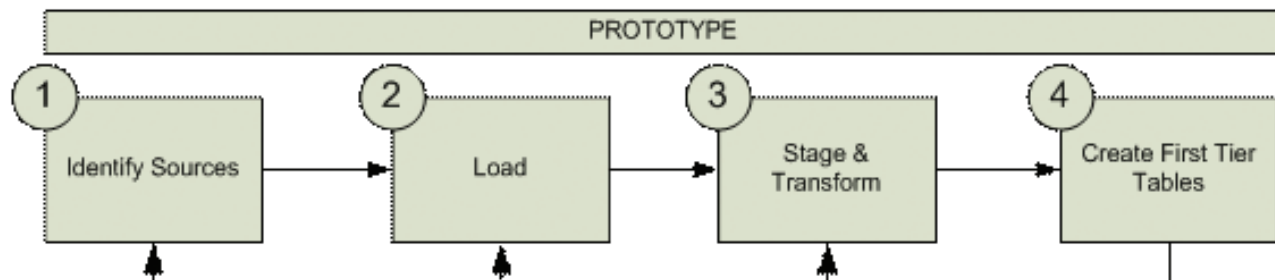


These three cycles of activity serve three overarching purposes:

- > First, they focus designers and implementers on the three areas of data warehouse practice that are collectively the source of most poor data warehouse or mart implementations: poor, untested schema designs that do not meet the actual needs of analysts and business decision-makers; overly lengthy deployment cycles that ensure that schema no longer meet the needs of the business by the time they are populated and in production; and fairly to refresh and redesign the warehouse or mart as business needs and market conditions change, often radically and discontinuously, and within relatively short time periods.
- > Secondly, they provide the basis for measurement. Data warehousing teams can and should measure their effectiveness in terms of the time it takes a team to (a) prototype and test warehouse and mart schema, (b) move prototyped and tested schema into production, and (c) modify, extend and enhance those schema. The best PDW practitioners measure themselves, in each cycle, in days or weeks, against industry norms of months and (in some unpleasant cases) years.
- > Thirdly, they mirror the way in which successful data warehousing practitioners iteratively and consciously build the confidence and commitment of the business analysts and users they serve, and who in most cases fund their projects: ultimately, the single most important factor in successful data warehousing projects. First, PDW makes certain that analysts' actual requirements are surfaced, vetted against available data, and tested with that data by the analysts themselves, making requirements effective, rather than a documentation exercise. Users get what they really want, and know what they get and why. Then, PDW quickly transforms that requirements-gathering and prototyping exercise into a real, useful production environment. Users get what they want, as they want it, when they still need it, fast. And finally PDW explicitly calls out the evergreen 'refresh' cycle that accrues in all successful data warehousing projects, as business requirements and analyst constituencies change, often rapidly and discontinuously, over time. Users get to change their requirements, whenever it makes business sense to do so.

Ultimately, PDW makes warehousing projects time-oriented and effective: fast, in every sense of that word.

THE LIVE PROTOTYPING™ CYCLE: FAIL FAST, FAIL EARLY



PDW Live Prototyping™ Cycle

We developed PDW's Live Prototyping discipline in response to several related phenomena we encounter in our work with customers:

- > The oft-remarked-on inability of even seasoned business professionals to clearly articulate their actual informational requirements in advance of seeing those business requirements embodied in data-rich schema.
- > The inability of business professionals to differentiate between data and its visual or paper-based representation, leading to common dysfunctional modes of requirements communication centering on, for example, batteries of reports required by users.
- > The ways in which prematurely hardened schema designs produce brittle data warehouses that cannot be changed when those schema are demonstrated, late and in production, to be inadequate for analysts' needs.
- > The desire on the part of analysts and decision-makers for decisional data that cannot in any reasonable way be extracted, or manufactured using business rules, from available source systems, and the ways in which those thwarted desires translate into analysts' negative opinions about the value of the data warehouse.

- > The desire of many project sponsors, funding organizations and risk managers to have failure occur early, in the design process, when costs are modest and investment low. From a commercial perspective, formalizing and setting expectations with end-users about specific information deliverables in a data warehousing project without an understanding of their cost, complexity and feasibility is an organizationally dangerous behavior if (as is always the case) resources, time and money are limited.³

Live Prototyping™ addresses these issues by, in essence, centering on the creation of data-rich prototypes of candidate production schema, populated in real time or near real-time with actual source data, in response to stated user requirements, in front of representative users in joint design sessions.

If you like, Live Prototyping™ is the iterative, joint refinement of the target schema, done by DW teams with representatives of their business users.

As such, Live Prototyping™ begins with a thorough investigation and inventory of the source systems implicated in the project. The project is bounded, at the outer edges, by available source data, which are either of value in native form, or as the inputs to business rules that transform native elements into synthetic forms of use to analysts.

Once source systems have been interrogated – particularly at the metadata level – and their assets are inventoried and understood, source system data and metadata sets likely to be useful in the design process are extracted and loaded into the data warehousing environment in unprocessed form.

Then, target schema design begins, using standard question-elicitation techniques, in a face-to-face, hands-on setting, with particularly expert and/or particularly representative members of the user constituencies for whom the warehouse or mart is being constructed. As discussion around design progresses, and possible target schema emerge in outline form, those schema are implemented (ideally in exactly the same technology infrastructure in which they will be deployed), and loaded with representative samples of source data (usually after appropriate transforms have taken place, again within the

database). Then, the user representatives are asked to evaluate the effectiveness and completeness of the populated design.

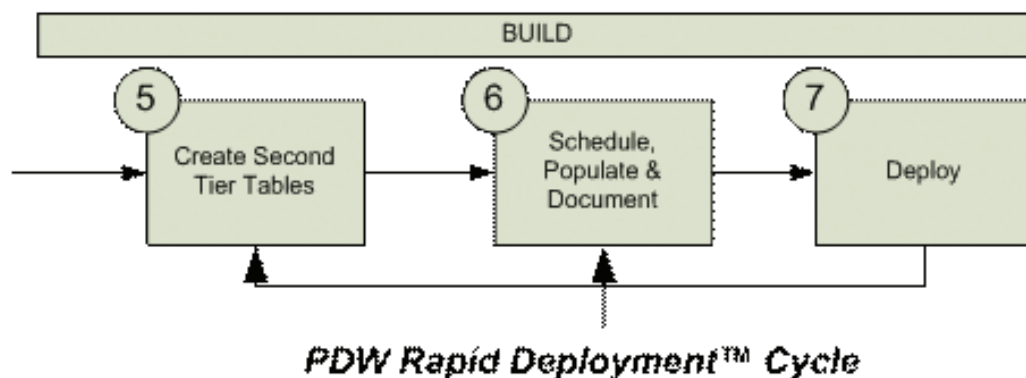
It is worth noting that, during this activity, user requests for data elements that designers would recognize as roll-up, aggregate, drill-through or drill-across elements are documented and tabled, for the time being, as the primary objective of Live Prototyping is to elicit and stabilize what PDW refers to as the first-tier schema – the fundamental schema of the warehouse or mart, populated with the lowest level of aggregation likely to be required by the analyst community served by it.

This design-populate-critique cycle is repeated until user requirements, feedback and design guidance are exhausted, or until the time allotted for the design activity has elapsed.

At that point, the first tier schema is reviewed a final time, and, in PWD parlance, “hardened”.

In its software incarnation within WhereScape RED, PDW fully supports Live Prototyping™ - RED does not require a target schema to start the design process, its ability to automatically generate extraction, transformation and loading code “on the fly” means that iteration cycles are tight (minutes or hours), and it has built-in support for common data warehouse design models, including stars, snowflakes and relational forms.

THE RAPID DEPLOYMENT™ CYCLE: ONCE IT’S BAKED, SERVE IT IMMEDIATELY



Every data warehouse or mart has, effectively, a set of first-tier (usually fine-grained) tables, as described in the previous section. Those first-tier tables hold both the data set that represents the outer bound of the warehouse's analytical support, and the design that – when Live Prototyping™ is employed – represents most accurately the current requirements of end-user analyst communities.

The key to successful deployment is: speed. Every day that the tested design produced in the first cycle is not deployed to business analysts is a day in which their needs, desires and expectations change, and in which the tested design diverges from those needs and expectations.

In PDW parlance, once a prototype has been signed off⁴, PDW emphasizes taking the prototype directly into production, in as little time as possible.⁵

Time being of the essence, the material objectives that must be accomplished in the period between finalization of the first tier schema and production deployment are fairly well-understood and straightforward:

- > Second-tier data warehouse tables, providing high value to the data warehouse user by aggregating, summarizing or otherwise elucidating the baseline first-tier data set, have to be produced. All client tool-specific metadata and all OLAP cubes, whether shared or personal, should be considered second-tier data warehouse tables.
- > The entire schema set embodied by the warehouse has to be populated, and its population regularly scheduled and automated to the fullest extent possible, with appropriate logging, commitment, concurrency and recovery mechanisms, and (in these days of oversight and scrutiny) auditability.
- > The entire warehouse environment has to be documented for two distinct audiences: analysts and end-users, on the one hand, who require documentation to help them make best use of the data warehouse's data resources, and warehouse administrators, who are responsible for the data warehouse's in-production stability (and perhaps its enhancement as well).

There may be related activities that must also be concluded during this cycle, including in many cases formal or informal end-user education for the client-side query and reporting infrastructure that will be used to access the data warehouse. There may also be formal production transition procedures that have to be executed as part of an organization's standard project management methodology.

In its software incarnation within WhereScape RED, PDW fully supports aggregate tables, views, join indexes for Teradata and OLAP cubes as second tier tables. These are created by simple drag-and-drop operations within the interface to the RED Repository™. Aggregations and transformations are carried out within the data warehouse database, without the need for a proprietary transformation engine. Additionally, because of the integrated nature of the WhereScape RED product, scheduling, populating and documenting the data warehouse is dead simple, and RED warehouses are self-documenting because the RED Repository contains all the information necessary to produce end-user and administrator documentation.

THE CLOSED-LOOP ENHANCEMENT™ CYCLE: KEEP IT HOT AND FRESH

As we mentioned in the introduction, the PDW mantra is: **do it fast, do it properly, and then do it again. And again. And again.**

Again, and again. The key to long-term success for data warehousing projects is, as any successful practitioner will tell you: keeping it hot and fresh.

Unfortunately, closed-loop enhancement of existing data warehouses is, all too often, impossible. The reasons for this are fairly well-understood:

- > The separation of design from implementation, and the inability to do iterative design in traditional ETL and data warehousing tools, in effect cements the first generation data warehouse schema in place for all time. To contemplate changes to those schema is, in effect, to contemplate abandoning the existing data warehouse or mart and starting the design-build-deploy cycle all over again – a proposition that, in terms of time and cost, few organizations have the stomach for.

- > The divergence of in-production data warehouses from actual, current business needs leads to (a) a vocal lack of confidence in and support for the in-production warehouse and (b) surreptitious coping mechanisms employed by end-users that effectively obviate the need for the data warehouse. Both (a) and (b) make it difficult to garner support and funding for long-cycle, expensive "refresh" initiatives.
- > Waterfall-based methodologies very rarely contain explicit provisions for refresh cycles in any form, and were developed for (transactional system) projects that did not include enhancement cycles as a matter of necessity, as they certainly are in data warehousing environments, which are supposed to support volatile decision-making processes.

Closed-loop enhancement, as the PDW diagram suggests, can implicate any previous activity, all the way back to the inclusion of new source systems and the creation of new or substantially-revised target schema, as well as later-stage activities such as the metadata models created for specific client-side tools, or the schedule and strategy (drop-and-reload, incremental addition, row-wise update) for warehouse rebuilds and refreshes.

The key is, once again, to be fast: to respond to inbound requests for change and enhancement in daily increments by:

- > Acknowledging and tracking all the requests for change and enhancement
- > Prototyping and testing changes to the in-production data warehouse while the warehouse is in operation
- > Rolling significant new and improved functionality into the warehouse's next (typically daily) rebuild cycle.

In traditional environments, this requires (a) dedication to a formal process of receiving end-user RFCs, (b) a parallel development-test environment in which to prototype and test responses to those RFCs, and (c) a mechanism for rolling tested changes into the next production build. In modern metadata-repository

based environments – such as that provided by WhereScape RED, the mechanisms to capture, prototype, test and schedule incremental changes to the warehouse are built into the metadata repository, and no elaborate process model or dedicated test-and-development environment is required.

CODA

No methodology can ever replace a designer's understanding of the analytical problem at hand. Data warehousing requires smart designers.

And no methodology can prevent poor technology selections, move or remove the limits imposed by tools and technologies, or repurpose a technology beyond its competency. Bad toolsets produce bad data warehouses, no matter how smart the designer.

With a good methodology, and a toolset that comprehends the complete, complex data warehousing life cycle, most of the risks traditionally associated with data warehousing projects can be mitigated or eliminated.

PDW, we believe, represents a pragmatic and highly-effective approach to data warehouse prototyping, building and enhancement, and embodies key principles we believe every experienced data warehouse practitioner knows, instinctively or otherwise:

1. Do it fast, do it properly, and then do it again.
2. Fail fast, fail early.
3. Once it's baked, serve it immediately.
4. Keep it hot and fresh.

Used in any form, within any methodology, the key elements of PDW will improve the practice of data warehouse designers and implementers.

Used in its entirety, as embodied in WhereScape RED, PDW represents the only holistic and complete approach to data warehouse life cycle management available in the market today.

¹ In fact, we particularly recommend the use of Live Prototyping™ techniques in standard SDLC-style projects, for risk mitigation reasons. Given that poor quality schema are a common cause of data warehouse and mart project failures, and given the long-time-cycle nature of SDLC-style models and their utter dependency on analysts' ability to fully define and articulate their needs in advance of seeing any significant amount of data, we know from experience that Live Prototyping™ significantly de-risks SDLC-style projects and raises the probability of success in such projects significantly – by half or more.

² See, for example, the use of WhereScape RED for Teradata-based projects, where the PDW methodology, as a whole, is simply not applicable.

³ Yet, as most of us know from cruel experience, this is precisely what SDLC waterfall-style requirements gathering methodologies lead to.

⁴ In our experience, formal approval activities are often little more than poor substitutes for committed end-users. Like so many legal documents, formal approvals presuppose a future conflict, for which they are supposed to be a prophylactic or antidote. PDW is indifferent to whether sign-off is a formal procedure, or an informal agreement to proceed between a DW design team and its user constituencies.

⁵ This is one instance in which the close connection between methodology and technology becomes rather obvious. In many traditional methodologies, design and deployment are two distinct activity sets, executed by distinct teams, and the design team often uses either (a) no technology at all to complete its design work or (b) technology that is unintegrated or incommensurate with the deployment technology infrastructure, requiring a (typically long) period of rework, modification and (all too often) compromise, as a tested design is retrofitted to a technology infrastructure of which it was unaware.

© 2006 WhereScape Software Limited. All rights reserved.

WhereScape is a trademark or registered trademarks of WhereScape Software Limited.

Brands or product names are the trademarks or registered trademarks of their respective owners.

For more information on WhereScape RED or to schedule a web demonstration visit www.wherescape.com